70 PAGES OF ESSENTIAL RASPBERRY PI KNOWLEDGE & KNOW-HOW

The MagPi

The official Raspberry Pi magazine Issue 32 Apr 2015 raspberrypi.org/magpi

Win! 10 RasPi Model A+s FROM THEPIHUT.COM

BUILD A STROBE LIGHT
Get the party started with a Pi

ANIMATE GRAPHICS IN PYTHON
How to make your Pygame projects move

FACIAL RECOGNITION MADE EASY
Using OpenCV and the camera module

RASPBERRY PI WEATHER STATION
The Foundation & Oracle team up for giveaway

CROWDFUNDING'S GREATEST HITS
The biggest projects powered by Raspberry Pi & funded by you

Also inside:

- › UNICEF PUTS PIS IN LEBANON
- › CONQUER THE COMMAND LINE
- › MAKE AN ELECTRONIC DOOR LOCK
- › MORE OF YOUR BRILLIANT PROJECTS

NATURE BYTES!
Get closer to nature with this awesome wildlife camera kit

Plus **SONIC PI: MAKE AMAZING MUSIC WITH OUR ESSENTIAL TIPS & TRICKS**

Weaved and the Power of Pi.

Add the power of IoT and mobile in 15 minutes.

LEARN MORE

Weaved™ IoT Kit

The Internet of Things for Everyone.

The Weaved IoT Kit for Raspberry Pi gives developers the tools they need to transform any Raspberry Pi application into an IoT application. Here's what you get:

Weaved Embedded Cloud Stack
This super-light, downloadable software package is installed on your Pi to communicate with the Weaved Cloud Services.

Weaved Cloud Services
Weaved operates secure Cloud Services so you can incorporate IoT features into your Pi project like iOS/ Android Push Notifications and remote Mobile Device connections to your Pi over the Internet.

Weaved App for iOS/Android
Download the Weaved App to create a real IoT mobile App experience for your Pi project. The Weaved App receives standard Push Notifications from your Pi based on trigger events you define. Use Weaved to monitor and control your Pi from anywhere through a Web GUI hosted on your Pi.

Weaved Developers Portal
Register as a developer at developer.weaved.com to receive full access to our downloadable embedded software, documentation, how-to videos and sample projects.

Weaved™

Weaved · Available on the App Store · Weaved IoT Kit

©2015 Weaved, Inc. All rights reserved. The Weaved App, 'The Internet of Things for Everyone,' Weaved and the Weaved logo are trademarks of Weaved, Inc. All other logos and names are trademarks or registered trademarks of their respective companies. Raspberry Pi is a trademark of the Raspberry Pi Foundation.

341 Hawthorne Avenue, Palo Alto, CA 94301 • Tel 650.262.0320 • www.weaved.com

WELCOME TO ISSUE 32!

Thanks for reading the latest issue of the official Raspberry Pi magazine. We've got another 70 Pi-packed pages, made up of the latest news, product reviews, step-by-step tutorials, and community features. We've also got another great competition this month, from our friends at **ThePiHut.com**, who have 10 Raspberry Pi Model A+s up for grabs. For your chance to win, all you have to do is head over to page 66 and tell us what sort of articles you enjoy reading in the magazine. We've listed a few categories for you to rank in order, but if you've got other ideas, we'd love to hear them – this is your magazine, after all. As an added incentive, a mini slice of Pi could be winging its way to you in a few weeks' time!

Thanks for all your amazing comments and feedback on the new-look magazine – we've all been totally blown away by your reaction and can't wait to share some of the plans we've laid for future issues. You might not have noticed yet, but we've also made a few technical upgrades in this issue that some of you asked for, too. Everything is single page-width layout (for easy viewing and printing) and all the URLs and email addresses are interactive. If you see a bold link like The Pi Hut one in the paragraph above, or our email address below, you can click it.

Don't forget that we want to hear what you've been doing with your Raspberry Pi – we can't make the magazine without your input! Speaking of which, if you'd like to write for the magazine, just get in touch via **magpi@raspberrypi.org**. I'm really looking forward to hearing from you!

Russell Barnes

70 PAGES OF ESSENTIAL RASPBERRY PI KNOWLEDGE & KNOW-HOW
The MagPi
The official Raspberry Pi magazine
Win! 10 RasPi Model A+s
BUILD A STROBE LIGHT
FACIAL RECOGNITION MADE EASY
ANIMATE GRAPHICS IN PYTHON
RASPBERRY PI WEATHER STATION
CROWDFUNDING'S GREATEST HITS
Also inside:
NATURE BYTES!
Plus SONIC PI: MAKE AMAZING MUSIC WITH OUR ESSENTIAL TIPS & TRICKS

THIS MONTH:

FIND US ONLINE raspberrypi.org/magpi

GET IN TOUCH magpi@raspberrypi.org

EDITORIAL
Managing Editor: **Russell Barnes**
russell@raspberrypi.org
Technical Editor: **David Whale**
Sub Editors: **Phil King, Laura Clay**

DESIGN
Critical Media: **criticalmedia.co.uk**
+44 (0)1202 399824
Head of Design: **Dougal Matthews**
Designers: **Mike Kay, Lee Allen**

PUBLISHING
For advertising & licensing:
magpi@raspberrypi.org
Publisher: **Liz Upton**
CEO: **Eben Upton**

With thanks to this month's contributors: **Sam Aaron, David Crookes, Liam Fraser, Gareth Halfacree, Phil King, Willem Koopman, Simon Long, Simon Monk, Les Pounder, Matt Richardson, Richard Smedley, Sean M Tracey,** and **Robin Withers.**

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 2051-9990.

70 PAGES OF ESSENTIAL RASPBERRY PI KNOWLEDGE & KNOW-HOW
The MagPi
Win! 10 RasPi Model A+s
BUILD A STROBE LIGHT
ANIMATE GRAPHICS IN PYTHON
FACIAL RECOGNITION MADE EASY
RASPBERRY PI WEATHER STATION
CROWDFUNDING'S GREATEST HITS
Also inside:
NATURE BYTES!
Plus SONIC PI: MAKE AMAZING MUSIC WITH OUR ESSENTIAL TIPS & TRICKS

Contents

Issue 32 April 2015 raspberrypi.org/magpi

SAVE 45% with a Newsstand subscription

Available now for smartphones & tablets

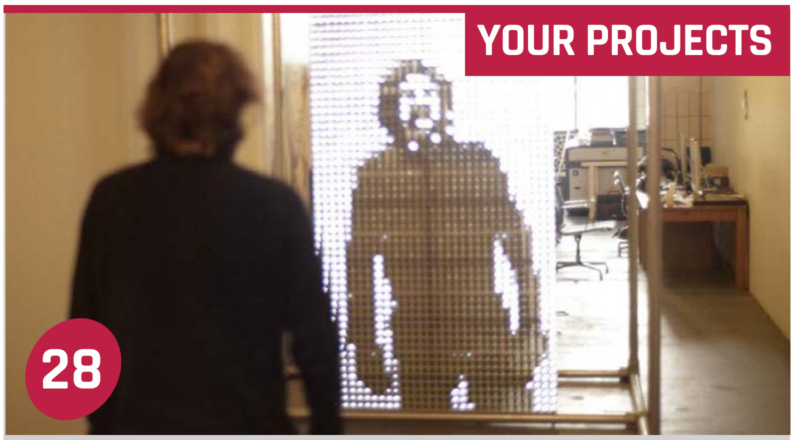
Get it on Google play

Available on the App Store

The MagPi

10 RASPBERRY PI Model A+s MUST BE WON! 66

What do you want to see more of in the mag? Let us know for your chance to win!



Key to keeping costs down, and the scheme portable, has been the use of HDMIPi screens

UNICEF TEACHES REFUGEES WITH PI

UNICEF innovator James Cranwell-Ward is using the Pi to teach displaced Syrian children in Lebanon essential numeracy and coding skills...

Often, children have a love-hate relationship with learning. While most undoubtedly understand its worth, there is also that great urge to get outside and play. But for the children who gather together in a section of a tent within a secured camp school compound in the Bekaa valley, near the Lebanese city of Zahlé, learning represents something else entirely: it gives a sense of normality.

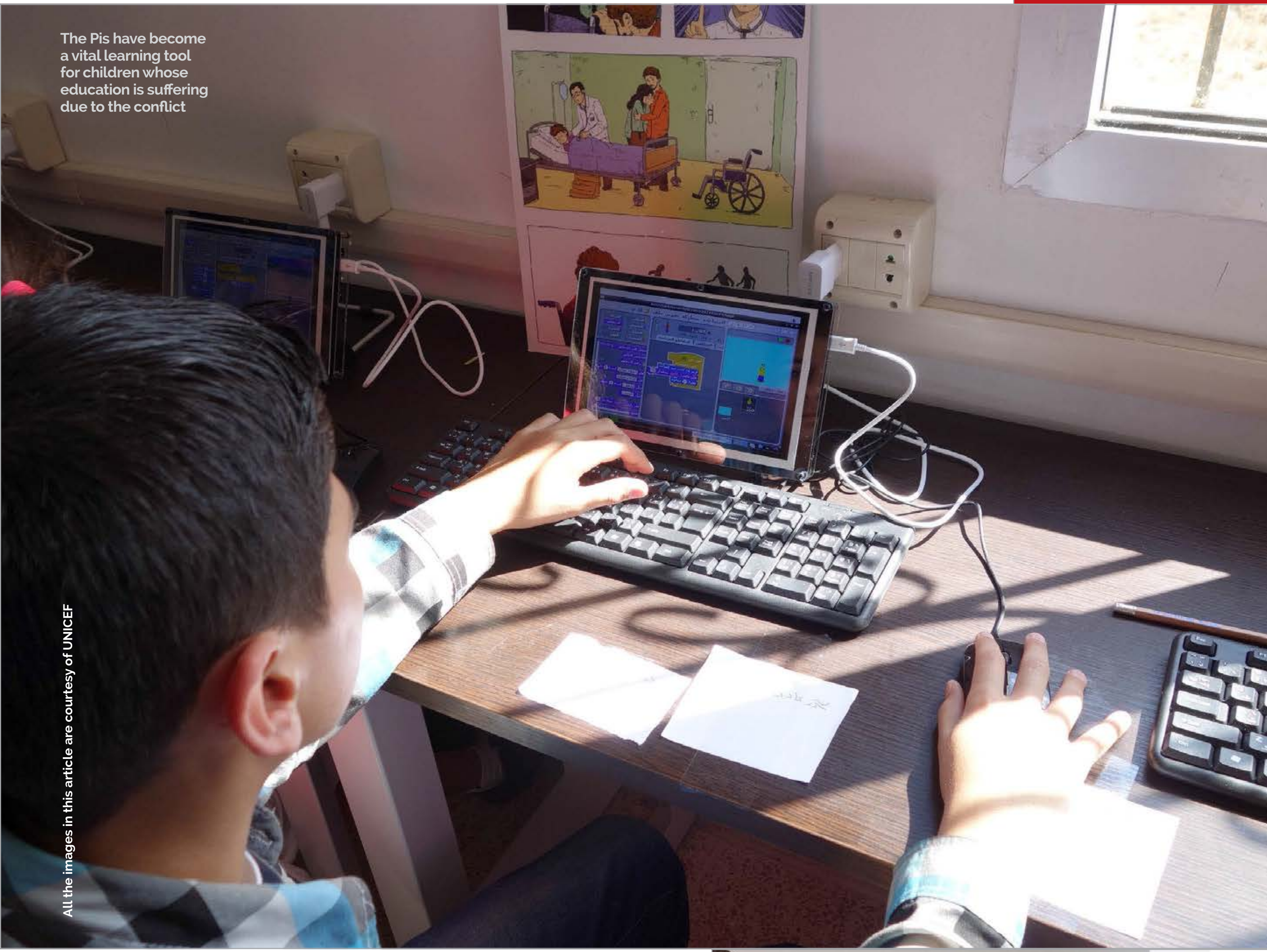
Ever since the Arab Spring reached Syria in March 2011, more than six million people have been displaced, a large number of them children. As well as fleeing over the Syrian borders to Iraq, Jordan, and Turkey, some 1.5 million refugees have come to reside in Lebanon. This has had a huge impact on a country which is little more than half the size of Wales, with an original population of four million.

For not only are many of the refugees undernourished, in need of medical attention, and living in unheated tents, the education of a good number of the 465,000 Syrian child refugees in Lebanon is suffering. But thanks to people like James Cranwell-Ward, UNICEF's Lebanon innovation specialist, things are starting to look up.

Raspberry Pi for Learning

James is a computer science graduate from the University of Manchester who has been working directly with Syrian refugees in Lebanon. One of his key tasks has been to investigate ways of providing non-formal education to millions of displaced children and he believes he has found the perfect solution – a pilot programme called Raspberry Pi for Learning, or Pi4L for short.

Devised by James and his team, the programme puts Raspberry Pis at the heart of the learning environment. A six-week-long course encourages children to learn basic skills in maths and science, before progressing to visual programming with Scratch. As well as giving children coding skills, the Scratch courses have been designed around UNICEF core themes.


The Pis have become a vital learning tool for children whose education is suffering due to the conflict

All the images in this article are courtesy of UNICEF

"So we have games based around the prevention of polio and importance of hygiene, and all kinds of facts of life and key issues affecting the population we are working with," explains James. "That has given it a solid ground around UNICEF's core messages, which was a nice touch as well." What's more, since many of the children have never even touched a computer previously, it brings them up to speed with modern technology. "It's why it is important they get some kind of access and important they do some kind of learning," adds James.

Before the project could get under way last November, James had his work cut out persuading UNICEF of its educational worth. But he was able to point towards a local non-government organisation (NGO) which had already put Pis in a Lebanese school. "I was really excited to see that," says James, who was looking for a local partner on the Pi4L project.

"UNICEF implements change through our partners and empowers local NGOs to deliver the work we want to do," James adds. "So with the NGO we had the technology and experience. And because their mandate was teaching training in technology for education, and because they were developing their courses with The College of Teachers in London, they were the perfect people to deal with."

Above James Cranwell-Ward demonstrates a Pi4L system

> We have games based around the prevention of polio and importance of hygiene and all kinds of facts...

Above James says the children have surpassed what the teachers can teach them and have reacted positively to the project

The price is right

James also had to make Pi4L economically viable. He had a figure of £100 per device in mind and he wanted a computer, display, keyboard, controller, and cables included in that price. Luckily, he knew about the HDMIPi Kickstarter project and believed that its inexpensive, 9-inch, high-definition screen would be a perfect fit for Pi4L. After rattling off an email to HDMIPi's co-creator Dave Mellor, James found himself meeting him in a coffee shop in London's Euston railway station.

"At that point Dave just had the screen but he was on board straight away," reveals James. "He gave me one of only three prototypes that they had and I went to Lebanon to present it to the UNICEF team. I showed them the screen in a Perspex box with a Pi inside and a wireless keyboard attached, but [they] took some convincing. A lot of people didn't know what a Pi was, never mind considering rolling it out across a country, but I told them of the possible uses and they agreed."

Since then, James and his team have sent dozens of Raspberry Pis, and just as many HDMIPis, to Lebanon. "We were the recipients of the very first batch of HDMIPis and we were receiving lots of free samples to play around with," James says, adding that the cables, keyboards and other electronics are sourced locally.

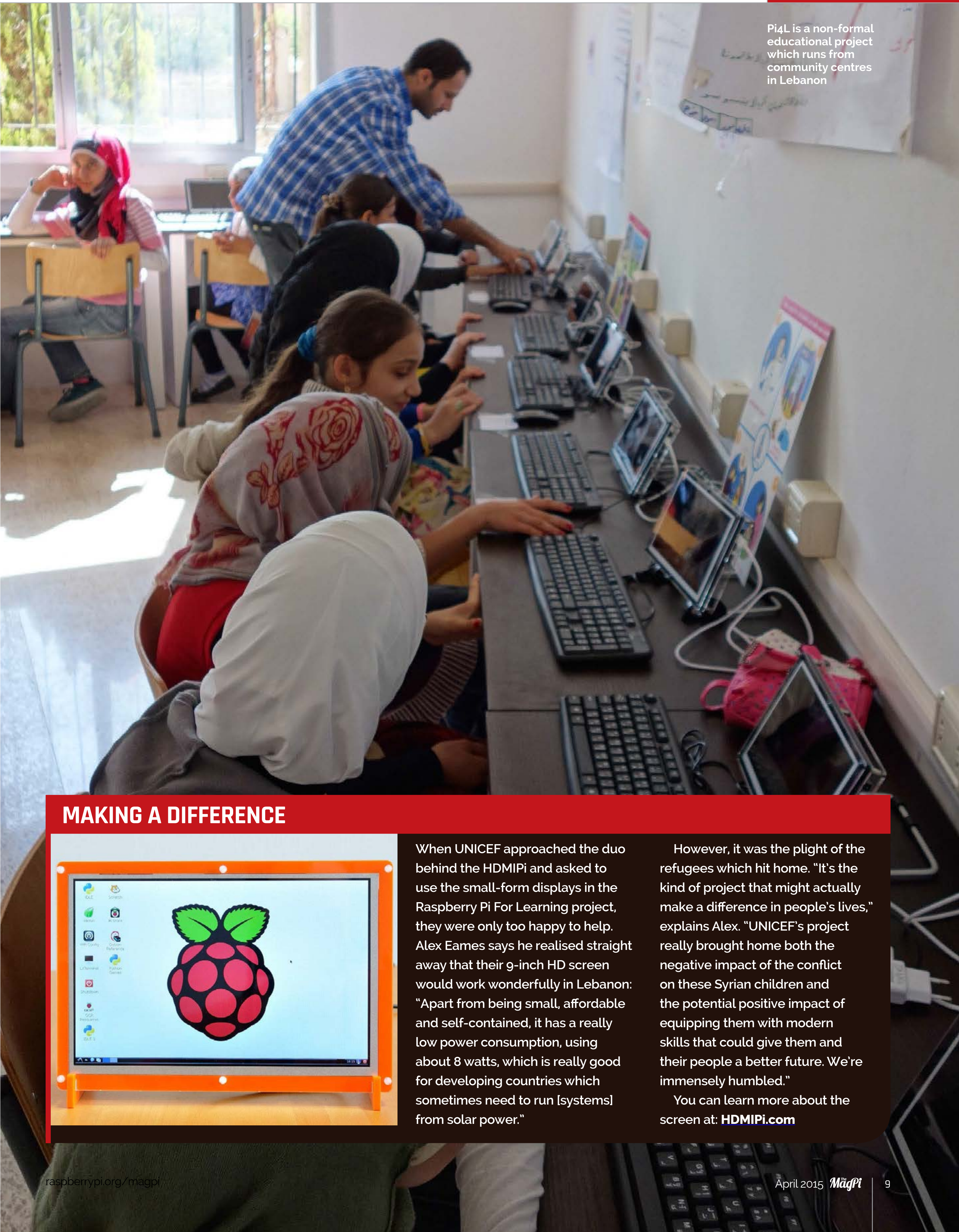
Reversing the model

James then looked at a setup created by the Foundation for Learning Equality in San Francisco, which was using Raspberry Pis as a server. "I flipped that model on its head, so that we use a laptop as the server and the Pis as the terminals to access content," says James. "The system works without constant need for the internet, which is very important because we can use local Wi-Fi networks instead and allow children to access through a browser. The laptops are updated via a 3G dongle."

Even so, it hasn't been a totally easy ride. With no electricity in some labs, UPS power supplies have been needed, but it has come together well. Indeed, so far a total of 158 children have benefited in three community centres and one informal settlement – in Zahlé, Beirut, Tyre, and Nabatieh – as part of the pilot program. A further 150 kids will be enrolled for the second phase. The idea is to continue rolling out the project in other areas of Lebanon and eventually elsewhere, each time giving students a chance to gain parity with the rest of the world.

"If you think about it, you have kids in developed countries using Raspberry Pis and learning different kinds of skills, and you have some of the most vulnerable Syrian refugees learning the same sets of skills and the same tech," enthuses James. "It's very rare to have this situation and it gets us around the problem of using second-rate computers too."

To learn more about the plight of Syria's children and make a donation via maintenance, visit: **unicef.org.uk.**



Pi4L is a non-formal educational project which runs from community centres in Lebanon

MAKING A DIFFERENCE

When UNICEF approached the duo behind the HDMIPi and asked to use the small-form displays in the Raspberry Pi For Learning project, they were only too happy to help. Alex Eames says he realised straight away that their 9-inch HD screen would work wonderfully in Lebanon: "Apart from being small, affordable and self-contained, it has a really low power consumption, using about 8 watts, which is really good for developing countries which sometimes need to run [systems] from solar power."

However, it was the plight of the refugees which hit home. "It's the kind of project that might actually make a difference in people's lives," explains Alex. "UNICEF's project really brought home both the negative impact of the conflict on these Syrian children and the potential positive impact of equipping them with modern skills that could give them and their people a better future. We're immensely humbled."

You can learn more about the screen at: **HDMIPi.com**

GET CLOSER TO NATURE WITH THE WILDLIFE CAM KIT

Build your own Pi-powered camera trap to capture stunning wildlife photos

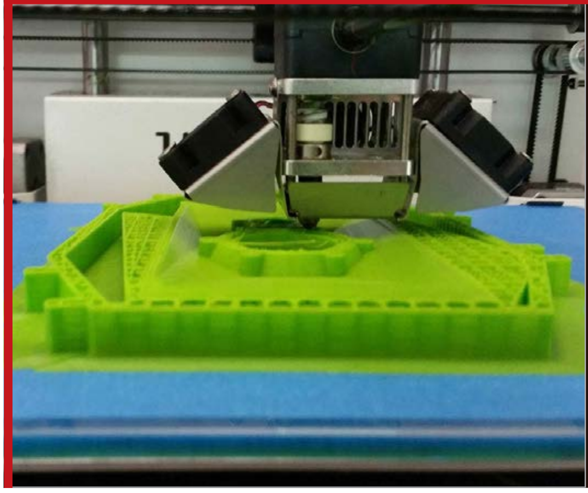
Wildlife camera trapping enables people to take amazing photographs of wild animals, including those that are normally camera shy. However, to be a camera trapper, you either need to invest in expensive off-the-shelf solutions or rig up your own digital camera and elaborate triggering mechanism before you can even get started. Naturebytes aims to change all that with its Wildlife Cam Kit, a project supported by the Raspberry Pi Foundation and Nesta. We spoke to Naturebytes' three co-founders – Alasdair Davies, Stephen Mowat and Jon Fidler – about how their Pi-powered camera kit will help users to reconnect with wildlife and contribute to conservation projects.

"It's a kit that can be built on your dining room table by the whole family, placed in your garden or local park, attached to your bird feeder, or simply hidden under the garden shed to look for badgers and foxes. You'll be automatically snapping photographs of wildlife and contributing to actual conservation projects by monitoring garden birds, urban wildlife, or participating in special Naturebytes challenges."

Powered by Pi

The Wildlife Cam Kit comprises a 3D-printed case which houses a Raspberry Pi Model A+, Camera module, PIR (passive infrared sensor), and LiPo rechargeable

3D-PRINTED PROTOTYPES



A major part of the prototyping process has been the use of Naturebytes' Ultimaker 2 3D printer, which has enabled the team to print of multiple versions of a case design quickly, and cheaply. "The great thing is that our Ultimaker allows us to print a case for around £3 in material costs, in as little as two days. This has significantly cut down our prototyping production time, and saved considerable costs versus traditional prototyping services."

The case even features a triangulated design to suit the 3D printing process: "Triangles are used to create the files when 3D printing (called STL files). When you look at these files, on closer inspection you will notice that they can be made up of thousands of triangles that then help the software create the tool paths used by the machines. The triangulated design is very much a theme that runs throughout our branding."

battery. "We selected the A+ model for its power efficiency, speed and fantastic supporting Camera module (for either daylight or night-time photography by switching to the PiNOIR), so we could construct a kit using a PIR sensor on the GPIO pins to detect a warm-blooded animal walking past, capture a high-resolution photograph within a few milliseconds, and then, by adding a real-time clock, record the exact time the event occurred."

The ~9600mAh battery will provide enough power for up to 24 hours before needing a recharge (by just plugging it in), although there's room in the case for a larger battery if required. Images are saved to the Pi's SD card, or an attached USB memory stick. You can either remove that or, since the hinged case opens up to reveal all of the ports on the Pi, just plug the latter into your TV to view the photos. Naturebytes is also encouraging users to hack the kit; so, for instance, you could easily add a Wi-Fi dongle to send the images to your network in real-time.

Open-and-shut case

Much of Naturebytes' development work on the kit has focused on the design of the hinged case. The use of SolidWorks 3D design software enabled the team to virtually lay out all the kit's components, which meant they could design the housing around them rather than vice versa. The prototypes were then 3D-printed with an Ultimaker 2 and used to "visualise the idea, test if the components fit, and actually test [them] in our own gardens." The concepts were also shown to select focus groups, including young festival-goers at last year's Camp Bestival. The design that was liked the most was chosen and the process repeated.

Another important aspect of the project is the creation of "a vibrant digital making community for wildlife." To this end, Naturebytes is introducing several unique modifiable features to the kit, the first of which is a removable insert. "This means that the kit can be used with all existing and future models of the Raspberry Pi family, as a new laser-cut insert can be 3D-printed, modified by a user, and reinserted into the protective case." The weatherproof case also allows for different lenses to be added to the design: "We are also going to release pod attachments, so users can attach IR LEDs / solar panels and any user-generated extras that the community think the kit needs."

With the ability to download community-created software, games, and wildlife-related activities, the Wildlife Cam Kit has been described as a 'Kano for nature', "in that it's a fully working computer, case, and feature-rich experience for the end user – only this time it's all based around supporting wildlife conservation and exploring the natural world."

Naturebytes is aiming to launch the Wildlife Cam Kit by the middle of 2015. For further details, and to sign up to receive updates, visit **naturebytes.org** or follow **@naturebytesuk** on Twitter.

> "It's a kit that can be built on your dining room table by the whole family"

WILDLIFE ACTION SHOTS

A variety of wildlife has already been photographed during initial testing of the camera kit, from garden birds like robins, and woodpeckers, to larger animals such as curious foxes. "Along with pictures of the usual suspects – such as the neighbours' dog escaping into the garden again – a snap of a muntjac (a small deer) was a nice surprise for one beta tester, who didn't ever expect to find one exploring their garden."

Above Young festival-goers got to assemble and play with a prototype at Camp Bestival

THE RASPBERRY PI WEATHER STATION

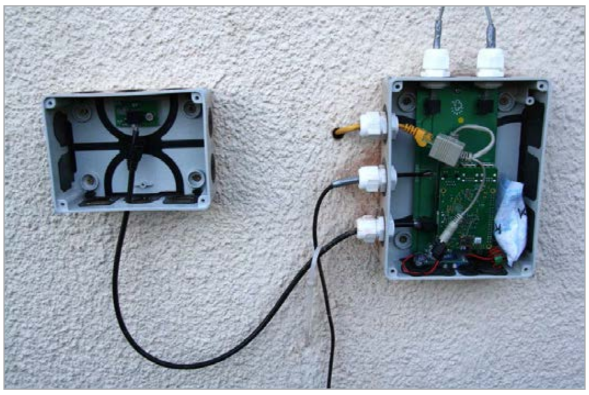
The Raspberry Pi Foundation and Oracle are giving away 1,000 weather station kits to schools around the world…

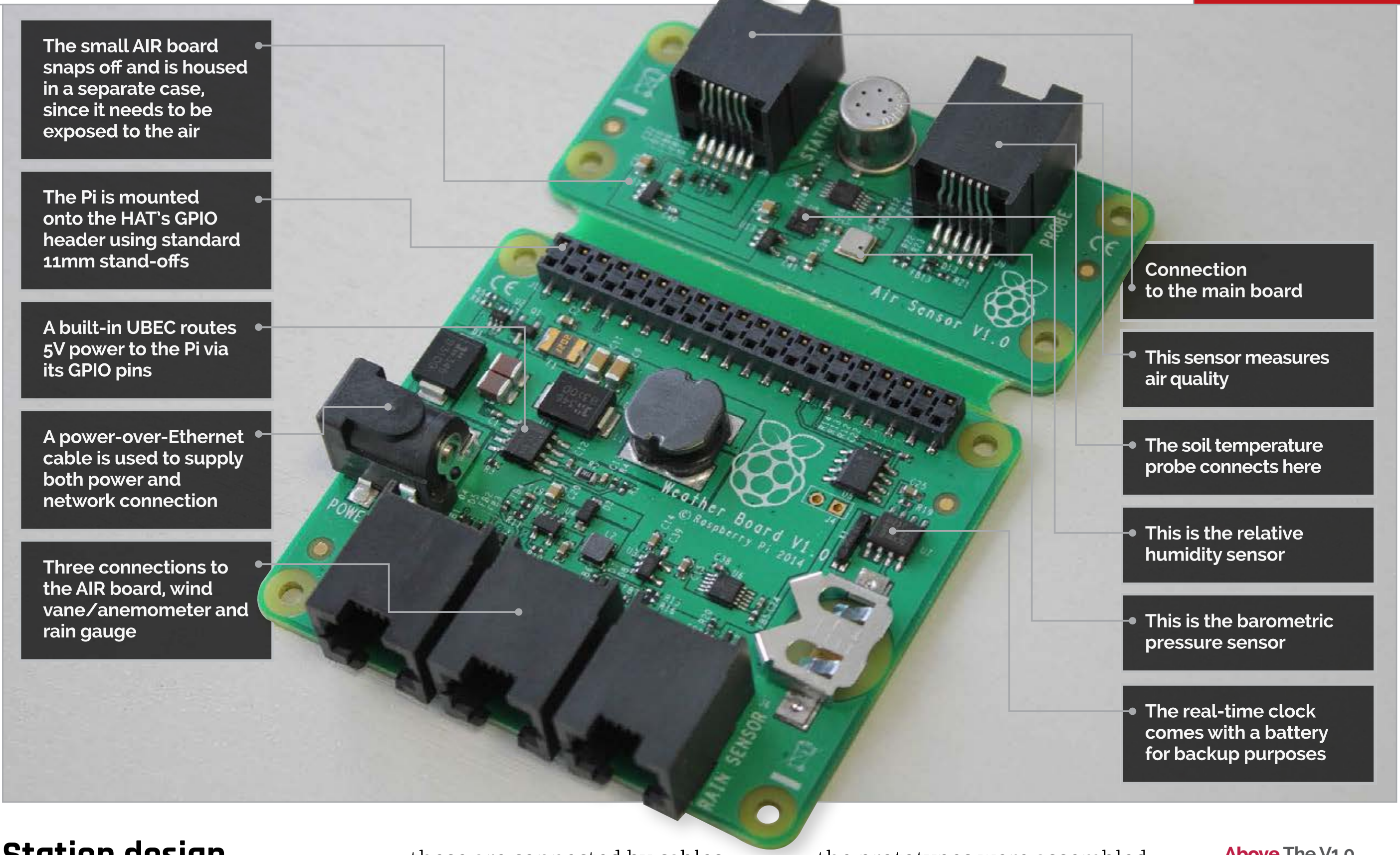
Why buy an off-the-shelf weather station when you can build your own and learn a whole lot more in the process? That's the idea behind the Raspberry Pi Foundation's latest educational project, Weather Station for Schools. Students will get to build a Pi-powered weather station from the supplied kit, write code to interface with its sensors, analyse the data collected, upload it to a database, and share it with others via a community website.

According to project leader and kit designer Dave Honess, it all came about when the Foundation was approached with funding by Oracle in FY2014: "They wanted us to develop a weather station around the Raspberry Pi, along with a scheme of work to go with it. The main aims are to provide kids with the opportunity to take part in cross-curricular computing and science projects that cover everything from embedded hardware to networking protocols, databases, and even big data. The plan is to roll out 1,000 free kits in the first year and hopefully more in the second year."

The goal has always been to have the Pi controlling everything, "to leverage learning opportunity: helping kids to learn about writing code to interface directly with the sensors, as well as displaying and analysing collected data." To this end, the kit's entire functionality can be accessed and modified by the user. "Data collection is done in Python; the measurements are stored in a local MySQL database which is then synchronised with a cloud Oracle system that will be free for everyone to use. There will be miles and miles of different kinds of learning opportunities that you can realise with this setup, from learning some electronics by interfacing the anemometer with the GPIO pins, to developing your own PHP/JavaScript website to display your local weather data. You could even pull data down from the Oracle DB and start making proper weather maps using the Google Maps API."

Below Two cases hold the air sensors (left) and the main board with Pi




The small AIR board snaps off and is housed in a separate case, since it needs to be exposed to the air
The Pi is mounted onto the HAT's GPIO header using standard 11mm stand-offs
A built-in UBEC routes 5V power to the Pi via its GPIO pins
A power-over-Ethernet cable is used to supply both power and network connection
Three connections to the AIR board, wind vane/anemometer and rain gauge
Connection to the main board
This sensor measures air quality
The soil temperature probe connects here
This is the relative humidity sensor
This is the barometric pressure sensor
The real-time clock comes with a battery for backup purposes

Above The V1.0 board. Note that the final design of the snap-off air sensor part will look slightly different

Station design

Not only was the project Dave's debut assignment upon joining the Pi Foundation but, remarkably, his first ever attempt at designing hardware. He eventually settled on a set of sensor measurements to include in the weather station: rainfall, wind speed, wind gust speed, wind direction, ambient temperature, soil temperature, barometric pressure, relative humidity, air quality, plus a real-time clock for data logging purposes. External sensors comprise an anemometer, wind vane, rain gauge, and soil probe; these are connected by cables to the main HAT board, which is plugged into the Raspberry Pi's GPIO pins and protected by a watertight case. A snap-off 'AIR' board featuring sensors for air quality, pressure, and humidity is housed in a separate case, since those sensors need to be exposed to the air.

Dave's first task was to produce 20 prototypes for use in field trials. "The electronics side of it was reasonably straightforward, but it was quite challenging sourcing all the components I needed from various suppliers… A couple of the prototypes were assembled incorrectly too, with capacitors on backwards, and these caused a nice puff of smoke when we turned them on." Once the prototype design was complete, Dave handed over his schematic to the engineers at Raspberry Pi Trading, "who then redid it properly with diodes, inductors, and all the nice electrical safety best practices that my prototypes don't really obey."

Dave opted to power the weather station using Power-over-Ethernet (PoE) because most school Wi-Fi networks are notoriously

EXTERNAL MEASURING EQUIPMENT

ANEMOMETER

This simple anemometer measures average wind speed and gusts.

WIND VANE

Like the other external parts, this wind direction sensor connects to the main board.

RAIN GAUGE

Used to measure precipitation over a period of time, the rain gauge is self-emptying.

SOIL PROBE

Once placed into the ground, this simple probe measures the temperature of the soil.

overloaded, with many mobile devices competing for service. "We wanted them to not have to worry about the connectivity to the weather station, because they would probably be hitting it from all the Windows PCs in the school pulling data off for their web development lessons. The weather station also needs a clear route to the internet so that it can upload its measurements to Oracle. So the connection needed to be reliable. It also needs to have a reliable power source, so PoE offers a nice way to kill both birds with one stone." He adds that anyone concerned about the possibility of lightning strikes can purchase an inexpensive Ethernet surge protector.

However, Dave notes that schools are not required to follow this setup and "modifying it for their individual needs can be quite a fun and engaging activity for students to do themselves. "One of the prototypes is actually going to be installed up a mountain in Northern Ireland with a 5.8GHz radio transceiver link to the school at a distance of several miles." In addition, schools are free to share the collected weather data with any other community-based projects, such as the one run by the Met Office: "Adapting the system to farm the local measurements out to other systems would be quite a nice programming challenge for students."

Although still in development, the project's hub website will show the geographical location of all weather stations in the programme. "You'll then be able to overlay different sensor data, show heat maps, and zoom down into different areas and interrogate each weather station individually. This will all be facilitated by the Oracle REST API that will be free for the public to use." The hub site will also enable users to locate the websites of other participating schools, interact with other participants about their experiences, blog, and get online technical support. "Part of the scheme of work will involve using the Oracle DB to explore weather measurements from nearby schools, which will cross over into the geography curriculum where students can try to explain the differences."

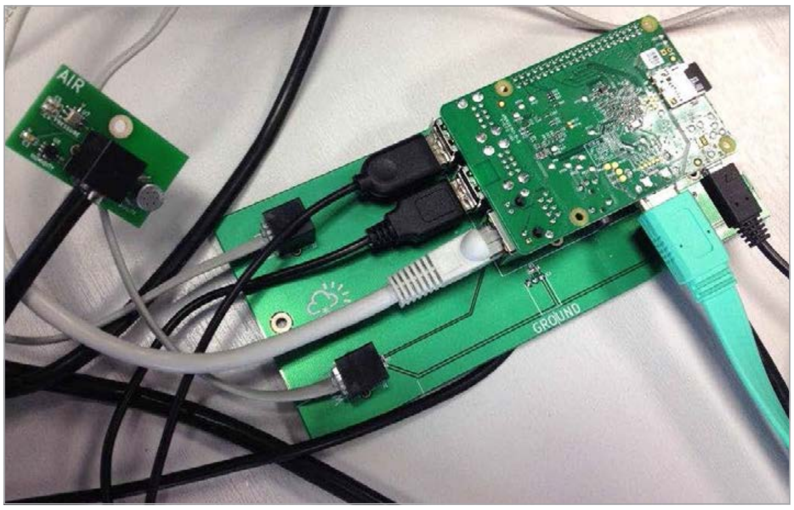
Lesson plans and other learning resources will be made available, along with source code for data logging and a demo display website (which can be found at **github.com/raspberrypi/weather-station**).

Global interest

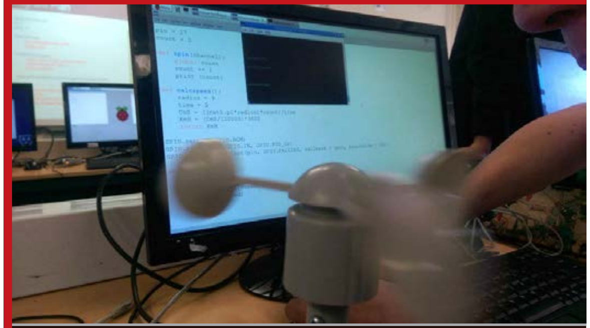
Already, the response from interested schools has been much greater than expected. At the time of writing, over 1,000 have applied to be part of the programme. "We have applications from every continent, and we've even had one from a school which is aboard a ship sailing in the southern Indian Ocean. We've said they can have one but they wouldn't be able to submit their data to the Oracle database, as that requires the weather station to have a fixed latitude/longitude."

While the project is aimed mainly at KS3/KS4 students, primary schools are free to apply. Since demand is now set to exceed supply, the assignment of kits will be "based on a number of factors, including number of pupils reached and geographical distribution, to make sure the network has a good amount of coverage globally... [but] we're hopeful that Oracle

Below Dave's prototype board hooked up to a Pi and various sensors



SCHOOL REPORT

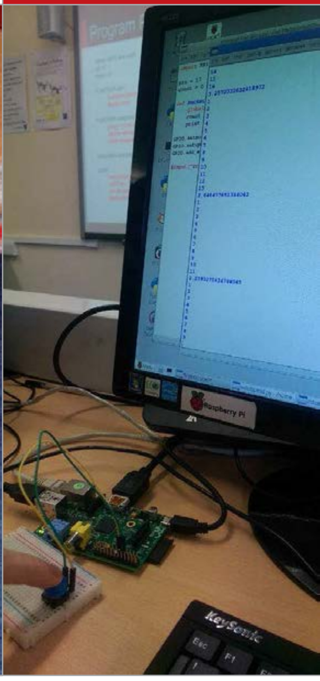


Soham Village College was one of several schools and coding clubs that got to try out a prototype version of the weather station. Led by computing teacher (and member of the Pi Foundation's education team) James Robinson, the Year 10 pupils got to build the kit and write code for it. James says assembly was straightforward, apart from having to cut holes in the rubber seals for the sensor wires: "I recommend a sharp knife and patience!" Apart from that, the hardest part was finding a location for the station that was "close enough to get power, but isolated enough to not get damaged. In the end we got help from the school's workshop to build a custom mounting bracket so that I could fix it to my classroom wall."

He says the students really engaged with the project and got a lot from it. For initial coding and testing purposes, they used a breadboard with jumper wires and a button to simulate the sensors. Year 10 pupil Harry particularly enjoyed employing the wind speed sensor: "We had to work out how the revolutions worked and how every time it spun twice we got a detection of the input."

The project also involves a lot of cross-curricular links, including elements of maths, physics, and geography. Harry would recommend it for other schools: "It's a good, practical idea for a subject and gets kids thinking about what to do." Fellow student Jamie thinks that due to their fully-tested code and the correct mathematical procedures employed, the data collected is valid and "could be used in any national studies [or by] the Met Office or other organisations."

Dave Honess created 20 prototypes to send out to testers



Above Students test their code by simulating sensor input using a simple push button

> "While the project is aimed mainly at KS3/KS4 students, primary schools are free to apply"

might want to make some more weather stations if this first lot of 1,000 goes well."

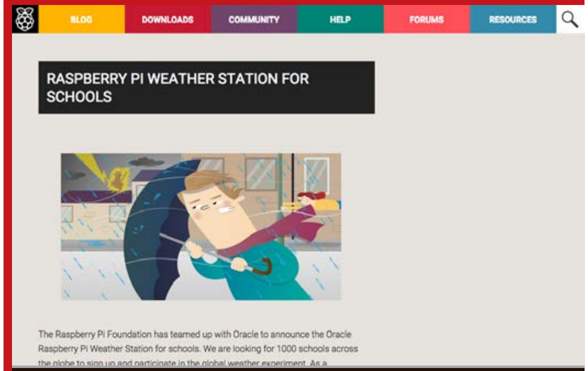
Currently only schools are eligible to apply, in order to fulfil the requirements of Oracle's grant proposal. "We have observed a lot of demand for it to be a product on general sale," says Dave, "but we'll have to wait and see if that can happen… It may be that we hand over the Gerbers and BOM [CAD files] to a third party who will continue to manufacture the HAT and kit in the future as their own product. However, no final decisions have been made here yet." If so, he would expect the price of such a kit (including a Pi) to be between £100 and £150.

Feedback from the prototype trial has been very positive, particularly regarding the cross-curricular opportunities afforded by the project. "When [computer science] is taught in context like this, it's more likely to engage more pupils," explains James Robinson, a computing teacher at Soham Village College (see 'School report' boxout). "In the initial work we did, we used aspects of science, maths, and geography to inform the computing we were doing. Once set up, the weather station is producing data sets that could be analysed in [those subjects]."

STEM ambassador and software engineer David Whale concurs: "I particularly like the fact that there are lots of opportunities for children to do science experiments and maths to understand the principles of reading data from sensors… [Children] have a complete system design in front of them to experiment with, and learn through real practical experience how to apply a range of techniques and skills they are already learning in class. Additionally, through the use of a data sharing website, children can compare results with others, learn and apply analytical techniques such as data interpretation [and] accuracy, and understand the practical limits of data returned from sensors."

Little wonder so many schools are applying to be part of such a versatile and educationally useful project. If your school is interested, applications are still being accepted via the website (see 'How to apply' boxout). At the time of writing, Dave Honess hopes the Foundation will be able to send out the first kits sometime in early April.

HOW TO APPLY



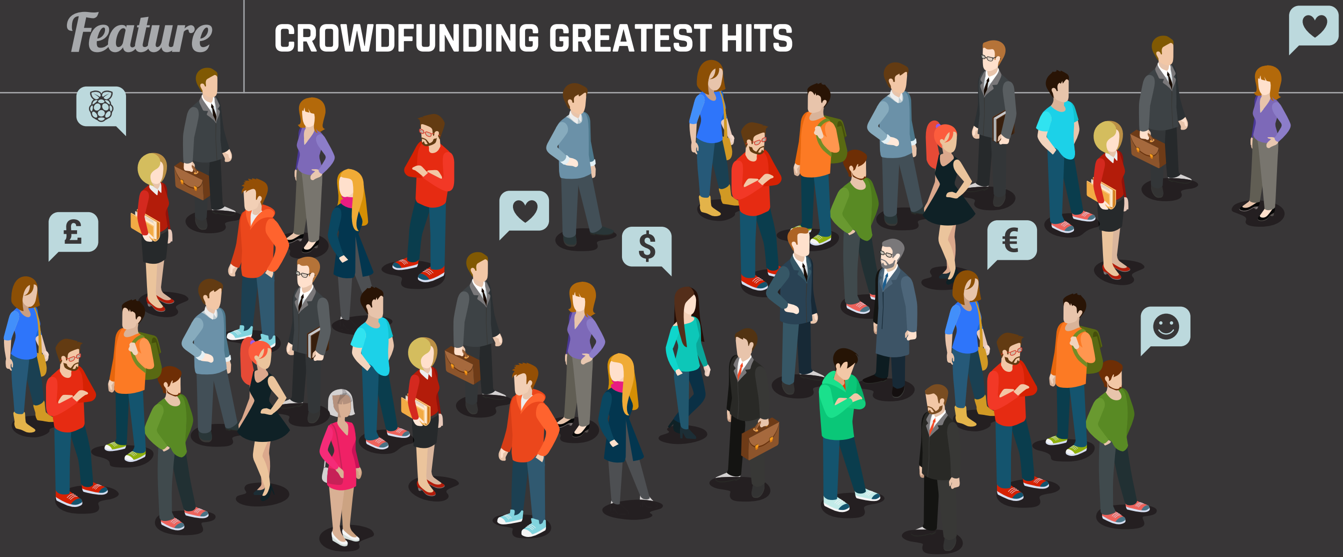
RASPBERRY PI WEATHER STATION FOR SCHOOLS

The Raspberry Pi Foundation has teamed up with Oracle to announce the Oracle Raspberry Pi Weather Station for schools. We are looking for 1000 schools across the globe to sign up and participate in the global weather experiment. As a

To find out more about the Weather Station for Schools project and register your school's interest, visit: **raspberrypi.org/education/weather-station**

Although the project is targeted at KS3/4 (i.e. secondary school) students, applications are welcomed from all schools from around the world. Participating schools will receive a free weather station kit for students to build, supported with a range of teaching materials covering computing and weather-related topics.

For more information about the development of the project, visit Dave Honess's blog post: **raspberrypi.org/school-weather-station-project**



CROWDFUNDING'S GREATEST HITS

The Raspberry Pi is the perfect maker tool, so it makes sense that it has played a pivotal role in some of the best crowdfunding stories ever told...

The low cost of the Raspberry Pi, coupled with its appeal to hackers and tinkerers, has launched an explosion of hardware projects from simple plastic cases, to entire spin-off devices powered by a Pi at their heart. For many, the only thing stopping them from taking their project from the workbench to market is a lack of funds, something crowdfunding sites like Kickstarter and Indiegogo aim to solve.

Crowdfunding serves several valuable purposes for smaller businesses and individual makers: it allows them to raise funds to bring a product to market without having to go through traditional funding channels, like bank loans or venture capital; it enables them to quickly gauge the demand for a particular product and receive feedback on its design ahead of mass-production; and it can help build buzz around a product launch.

For customers – 'backers,' in crowdfunding parlance – it provides access to hardware that would otherwise not be readily available, along with a feeling of being part of the journey to market that doesn't come from simple pre-orders or registrations of interest on a corporate website. While only around two in five Kickstarter campaigns reach their funding goal, the right idea at the right time can earn a fortune, as proven by our rundown of the biggest successes from the Raspberry Pi community.

It's a heady mixture that has featured some stunning successes, along with a perhaps surprisingly small number of failures. If you've ever wondered about the secret of crowdfunding success, we've spoken to those who have been there and done that, to bring you the advice you need to know in order to succeed, while avoiding the pitfalls that have caused others before you serious upset along the way.

KANO

The top-grossing crowdfunding campaign in Raspberry Pi history by quite some margin, Kano raised nearly £1 million in November 2013, with a premise that at first seems laughably basic: a Raspberry Pi starter kit bundle.

Positioning the project as ‘a computer you make yourself’, Kano bundled an off-the-shelf Raspberry Pi with carefully-selected accessories, including an eye-catching orange keyboard and trackpad, and a customised operating system. Its real potential came with the unveiling of Kano Blocks, a bundled graphical programming language inspired by MIT’s Scratch and Google’s Blockly, along with built-in lessons to walk the user through everything from assembling the kit to programming a game.

Despite its record-breaking funding run, Kano – as is common with crowdfunding projects – ran into some trouble during production. The first Kano units were due to ship in June and July 2014, but problems with the HDMI cables, keyboard battery, flashing the SD cards, the power supplies, and even the plastic case, meant that the first kits didn’t ship until September and it was October before the majority of backers received their rewards.

Since shipping Kano, the team behind it has continued to build on its success with the creation of Kano Challenges, educational contests designed to further encourage children to get involved with programming and computing.

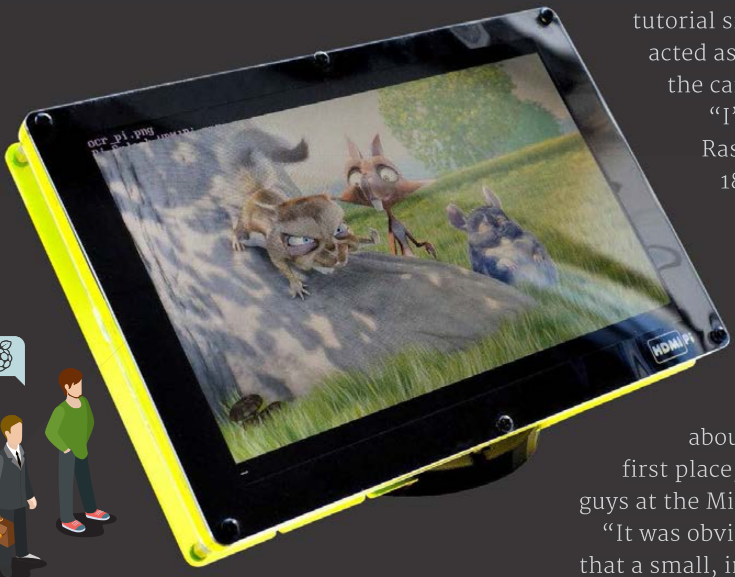
HDMIPi

The HDMIPi campaign raised £261,250 – far in excess of its £55,000 goal – from 2,523 backers to produce a low-cost high-resolution display. HDMIPi is notable for another reason than its sky-high funding, though: the project was a joint venture between electronics professional Dave Mellor of Cyntech and Alex Eames of review and tutorial site RasPi.TV, who acted as the public face of the campaign.

“I’d been doing RasPi.TV full-time for 18 months, with no income from it, before HDMIPi,” Alex explains. “It was Dave Mellor of Cyntech who approached me about the project in the first place, after talking to the guys at the Milton Keynes Jam.

“It was obvious to all of us that a small, inexpensive, portable screen was needed for the Pi – we all wanted one – but there wasn’t anything out there under £100. We thought it ought to be possible to make that happen, and we were right, but it wasn’t easy.”

While the crowdfunding campaign was a great success, the project hit difficulties during fulfilment, which led to some backers receiving their displays almost a year late – a common theme in hardware campaigns. “The actual driver board design came partly from ideas [that] backers suggested during the campaign,” Alex recalls. “We liked them because they made the product unique, but they were probably the biggest cause of the delays; in hindsight, we could have probably delivered months earlier if we hadn’t listened.”



SLICE

The creation of FiveNinjas – a group comprising members of Sheffield-based Pi accessories maker Pimoroni, the Raspberry Pi Foundation itself, and music producer and technology trainer Mo Volans – Slice raised an impressive £227,480 to build a media playback set-top box based around the at-the-time new Raspberry Pi Compute module.

Rather than taking the retail Raspberry Pi hardware as the basis for the project, as many other campaigns have done, the Compute module – an industrial Raspberry Pi variant based around the SODIMM form factor, designed to act as a plug-in computer-on-module for a custom-designed carrier board – allowed FiveNinjas greater control over the final design and layout of the project.

The result is a sleek aluminium box featuring a smart LED strip under user control, a wireless remote control, and an internal hard drive for storage. The Slice focuses more on playback of local content rather than streaming, although Wi-Fi connectivity was unlocked as a stretch goal when the campaign hit £100,000. It includes a slick user interface and excellent compatibility with various file formats, along with features such as SATA storage connectivity, missing from the standard Raspberry Pi hardware.

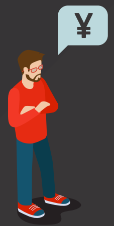
Like the majority of crowdfunded projects, Slice has been hit by numerous delays. The rewards were scheduled to be with backers by November 2014, but it took until the end of February 2015 for the first Slice packages to begin shipping.

FLOTILLA

Flotilla, in fourth place with £146,680 raised, is the second Kickstarter campaign to come directly from the Sheffield warehouse of Pimoroni, and takes aim at the educational market with a range of smart input and output modules which can be easily programmed from a Pi.

Pimoroni made a name for itself in crowdfunding circles by launching the first project to go live on Kickstarter UK: the Picade. "We actively wanted to be first. We'd heard that Kickstarter was coming in the UK, so we made sure that our campaign was ready and [that] we were there ready to press the button," explains Jon Williamson. "We just stood there at ten to midnight, pressing F5 until the live button became live, and we were the first to press it," adds a laughing Paul Beech.

Despite not having the benefit of being first, Flotilla easily surpassed the £74,134 raised by Picade. The company has a few ideas for avoiding the delays that plagued its original project, too. "We spent a year working on Flotilla before we put it on Kickstarter," Jon reveals. "With Flotilla, we've already been through six or seven iterations of the design of the system; the software has been in development for six months. Really, it's much closer to being finished this time around, and I guess that's what we've learned: the more stuff that you leave unfinished, the more things that can go wrong to trying to finish it."



PI-TOP LAPTOP

The only top-ten grossing Raspberry Pi crowdfunding campaign to use Indiegogo rather than the more well-known Kickstarter platform, London-based Jesse Lozano and Ryan Dunwoody's Pi-Top project looked to encourage people to learn computing by building their own Raspberry Pi-powered laptop. With £112,130 raised, more than double its original goal, the campaign certainly caught the interest of the community.

While much of the publicity surrounding the campaign focused on the hardware side – including the team's decision to 3D-print the chassis' master, then injection-mould for mass-production – Pi-Top promised more: like Kano, the project positions itself as the ideal platform for teaching beginners about computing, but with a focus on hardware rather than software.

The Pi-Top campaign closed in December 2014, and since then, the team behind it has been working to keep backers informed. In the most recent campaign update, which gave all backers a free upgrade to the Raspberry Pi 2 from the planned first-generation model, the team claimed that it was still on track to ship the devices to backers in May – which, if achieved, would make it one of the few crowdfunding campaigns in history to stick to its originally published schedule.

As with most crowdfunding projects, the Pi-Top is due to go on general sale once the backers who funded its journey from prototype to production have received their rewards.

Top Ten

RASPBERRY PI CROWD-FUNDING CAMPAIGNS

- Kano
  £987,976
  (1,522% funded)
- HDMIPi
  £261,250
  (475% funded)
- Slice
  £227,480
  (253% funded)
- Flotilla,
  £146,6W80
  (448% funded)
- Atlas 3D
  £141,940
  (7,333% funded)
- Pi-Top Laptop
  £112,130
  (217% funded)
- BrickPi
  £82,286
  (6,752% funded)
- Ziphius
  £80,873
  (102% funded)
- Rapiro Robot
  £75,099
  (375% funded)
- Picade
  £74,134
  (226% funded)

ATLAS 3D

Atlas 3D, from Kentucky-based Murobo LLC, is an example of the Raspberry Pi providing the power, rather than inspiration, for a project. Raising £141,940 – a record-smashing 7,333% of its modest £1,944 goal – the Atlas 3D is a 3D scanner built around a 3D-printed chassis and the use of lasers to measure the shape of the target object.

"The Raspberry Pi was an easy choice since it has an excellent 5-megapixel camera add-on, is able to drive lasers and motors, and has enough memory and CPU power to perform a 3D scan," project creator Uriah Liggett explained about his creation during its crowdfunding campaign last year. "All of the software runs on board the Raspberry Pi, so there are no required drivers or software packages to install."

Uriah's project promises to provide benefit not only to its backers, but to the community at large, too. The driving software, FreeLLS, is available under the GNU General Public License as an open source project, while the electronic design files are open hardware. Although the design files for the printed circuit boards and 3D-printed components were not available at the time of writing, Uriah indicated that they would be published once backers' rewards begin shipping in April 2015.

While Uriah took a risk by promising to release his creation under an open licence, since potential backers could have been put off by the idea of paying for something that others could build themselves from free designs, his impressive funding run proves that crowdfunding and open source are not mutually exclusive.

CROWDFUNDING FAILURES

Not every crowdfunding project ends in success, of course. Overall, 61 per cent of Kickstarter campaigns fail to make their funding goal; of these, more than half never even reach one-fifth of the way. These, however, are the lucky ones, failing gracefully and without anyone ending up out-of-pocket. More notable are those projects that reach their goal, collect backers' pledges, and then fail to deliver on their promises.

Azorean's Ziphius caught the community's attention back in 2013 with the promise of an aquatic drone powered by a Raspberry Pi and controlled from a smartphone app. The campaign ended with nearly £81,000 to produce the device, with a self-imposed deadline of March 2014.

“Projects that reach their goal, collect pledges, then fail to deliver”

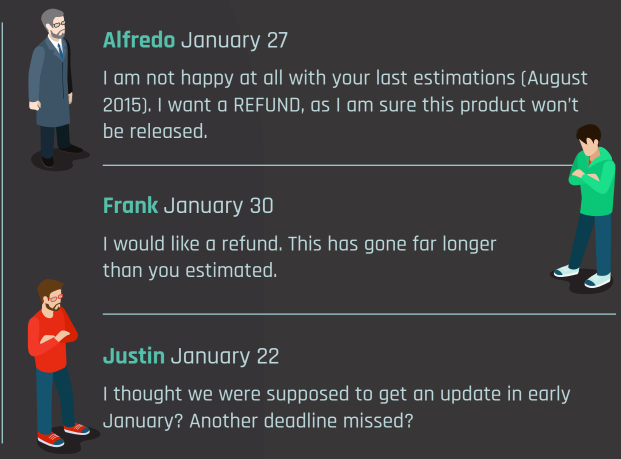
Not going as planned

By March, only the control boards were completed; in May, the company was still working on prototypes of the drone's chassis and control systems. A revised delivery plan suggested an October release date but in January 2015, Azorean admitted that it would be August before devices shipped – leaving backers clamouring for refunds.

“The delays were caused mainly by the difficulty moving from the prototype that we had at the time and the need to adapt it to the new tools of mass-production,” Azorean's Cristina Gouveia explained. “We spent more, much more, time than we were expecting.”

Those issues may now be resolved, but there's a more serious problem on the horizon: a lack of funds. “That delay has caused us some financial struggles, so we need to get additional investment,” Cristina admitted. As to whether Azorean will be able to ship its promised rewards to backers if investment fails to materialise: “If we don't [receive investment], I don't think we can.”

Above The Azorean Ziphius was a runaway crowdfunding success in 2013, but is still yet to deliver



Alfredo January 27

I am not happy at all with your last estimations (August 2015). I want a REFUND, as I am sure this product won't be released.

Frank January 30

I would like a refund. This has gone far longer than you estimated.

Justin January 22

I thought we were supposed to get an update in early January? Another deadline missed?

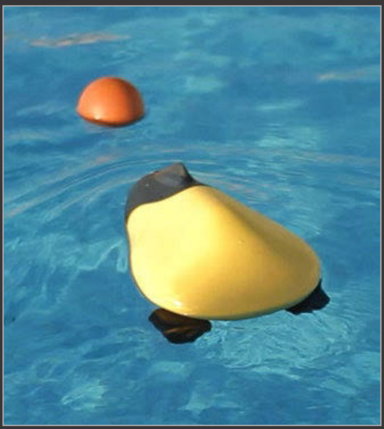
Still taking money

At the time of writing, Azorean was still accepting pre-orders for the Ziphius through its own website, priced at $269 plus shipping. Those placing an order receive an email warning of the new August delivery date, but with no mention of the company's financial struggles, nor that production of the Pi-powered aquatic drone may not take place if additional investment cannot be found.

Azorean's case may be an extreme one, but delays to the schedules originally proposed by crowdfunding newcomers are all too common. "The skills required for prototyping and production at moderate scale are quite different," Saar Drimer, of electronics design consultancy Boldport, explains. "On top of that, the ease by which prototyping is done today can be very misleading. 3D-printing a nice enclosure is pretty easy, whilst sourcing injection-moulded enclosures in volume from China is a completely different matter. Another example is sourcing components: if you're not aware of the many pitfalls of availability and obsolescence of components, you're going to get burned."

Pimoroni's Jon Williamson can certainly attest to the latter, having run into exactly that pitfall during the Picade campaign's delayed fulfilment. "The screens were a nightmare. We picked a screen that was basically end-of-life. It was a great screen, but they were like hen's teeth. We'd get odd batches of them; we might get 20 in Hong Kong one week, and we'd just grab them and we'd have 20 screens, great, but we needed 500. It was not a pleasant experience."

Communicating with a contractor overseas, often the only way a campaign can produce hardware at an affordable level, has its own challenges, too. "The biggest problem we had was communication with China," reveals Alex Eames about the HDMIPi campaign. "Getting things done the way we wanted, meeting our specifications, and being of usable quality was really difficult."



CROWDFUNDING GLOSSARY

Backer Someone who contributes money to a crowdfunding campaign, known as a 'funder' on some sites.

Campaign An attempt to raise funds through one or more crowdfunding sites, usually limited to a 30-day period.

Creator The individual or company behind a crowdfunding campaign.

Early Bird Reward A strictly limited number of rewards offered at a discount, designed to help a campaign build momentum in its early stages.

Fees Crowdfunding sites take a cut of all money raised by a campaign, usually around 10 per cent, in exchange for providing a platform and handling payments from backers.

Flexible Funding A feature of Indiegogo which allows campaigns to receive backers' funds even if the campaign does not reach its goal.

Goal The amount of money required by a crowdfunding campaign to be successful. If this is not raised, the funds are usually returned to backers.

Limited Rewards Campaigns can set limits of the number of rewards on offer, either to boost hype through artificial scarcity, or to meet an upper limit on manufacturing runs.

Pledge The money contributed to a crowdfunding campaign by a backer.

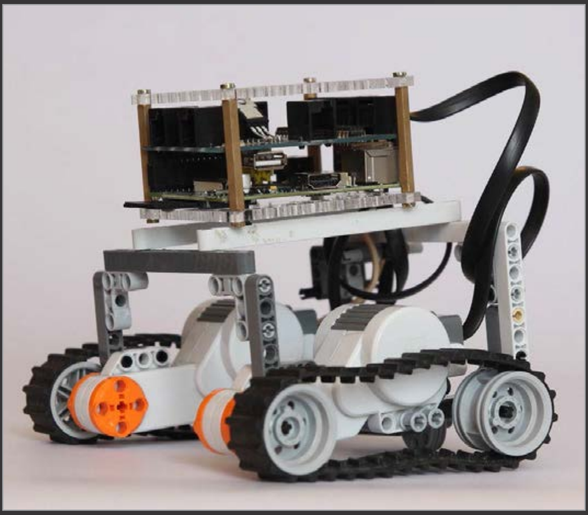
Reward The promised return for a backer's pledge, ranging from an email or T-shirt at the lower levels, to multiple units of the item and even dinner engagements with the creators at the upper end. Known as a 'perk' on some sites.

Story The pitch for the campaign, which typically includes a video presentation, a write-up of the project, information about the creator, a planned delivery schedule, and a section on potential risks.

Stretch Goal A campaign that raises more than its original goal may use the extra money to improve the product, offering a wider choice of colours, free extras, or upgraded features to backers.

Updates A good crowdfunding campaign will post regular progress reports to its page, both throughout the campaign and leading up to reward fulfilment.

FINDING CROWDFUNDING SUCCESS


Above The Dexter Laboratories' BrickPi raised over 67 times its original target

For those who want to follow in the footsteps of campaigns like Kano, HDMIPi, and Slice, it's all too easy to be blinded by the sums of money on offer and leap in without proper preparation. Convincing people to part with their cash is only the start of the process.

"You [need] to have your plan together for manufacturing, and then have a second one in the waiting if things go really, really well, before you launch your campaign," advises Dexter Industries' John Cole, whose LEGO-based BrickPi robotics kit raised £82,286 – over 67 times its original target. "That should not be an afterthought – that should actually be baked into the campaign. It ties into really concrete things, like what you price your product at and what you promise for a delivery date.

"It also factors into the softer side of developing a product, which is who are you going to be selling it to, what are their tastes, wants and needs? You should understand that clearly before you start the campaign and communicate that with your manufacturer, because they'll play a big role in whether you have a successful campaign or not."

PLEDGING FOR THE SKY

Crowdfunding campaigns typically offer a variety of reward levels, from the pound donation that gets a backer heartfelt thanks, to rewards valued at thousands of pounds. Here are some of the biggest reward levels seen...

Kano Academy
£6,490 to receive workshops for 100 people, bespoke tutorials, guides, and everything an establishment needs to become a Kano Academy, plus a T-shirt. Amazingly, one generous backer snapped the offer up.

BrickPi Namesake
£6,489 to rename the BrickPi's case to anything your heart desires, so long as it's suitable to repeat in polite company. Unsurprisingly, nobody backed the project at this level.

Rapiro Custom Design
£5,000 to have a Rapiro robot customised to your precise requirements. With the base kit priced at just £229, nobody took creator Shota Ishiwatari up on his offer.

Ziphius Prototype
One backer gave Azorean £4,858 to receive a pre-existing Ziphius drone prototype, as used when the company entered the Engadget Expand Insert-Coin Competition.

Slice Full Custom Shop
£2,999 to work with the FiveNinjas design team to construct an entirely custom Slice media player. As with Rapiro, the cost over the £179 standard version was considered too high for anyone to pledge.

Pi-Top One of a Kind
£1,622 to discuss the creation of a custom Raspberry Pi laptop, complete with the promise of automatic hinge and integrated light show. Despite planning for five potential backers, the Pi-Top project was left with no takers for this top-tier reward level.

"There have been quite a lot of Raspberry Pi-based crowdfunded projects – even a couple of screens – that have not succeeded because the community has never heard of the people behind them," adds Alex Eames, who was chosen to front the HDMIPi campaign specifically for his recognisability in the Raspberry Pi community. "The takeaway message from this is make sizeable deposits into the community before you try to make a withdrawal. If you look at who's done well in high-value crowdfunding projects, there aren't many who've managed it without significant input into the community over a sustained period of time before they had their 'overnight success'. You won't get money out of the community unless the community thinks you deserve it and can be trusted with it."

> "If you've got a hardware idea, keep it simple and true to its core"

Minimise complications

The lessons Pimoroni's Jon and Paul are taking into the Flotilla fulfilment process are simple. "Minimise the number of complications. If you're thinking 'should I add this feature?', then I think you should err on the side of caution," Jon advises. "If you start adding features part-way through the design process, you'll risk overshooting by a month, minimum, maybe more."

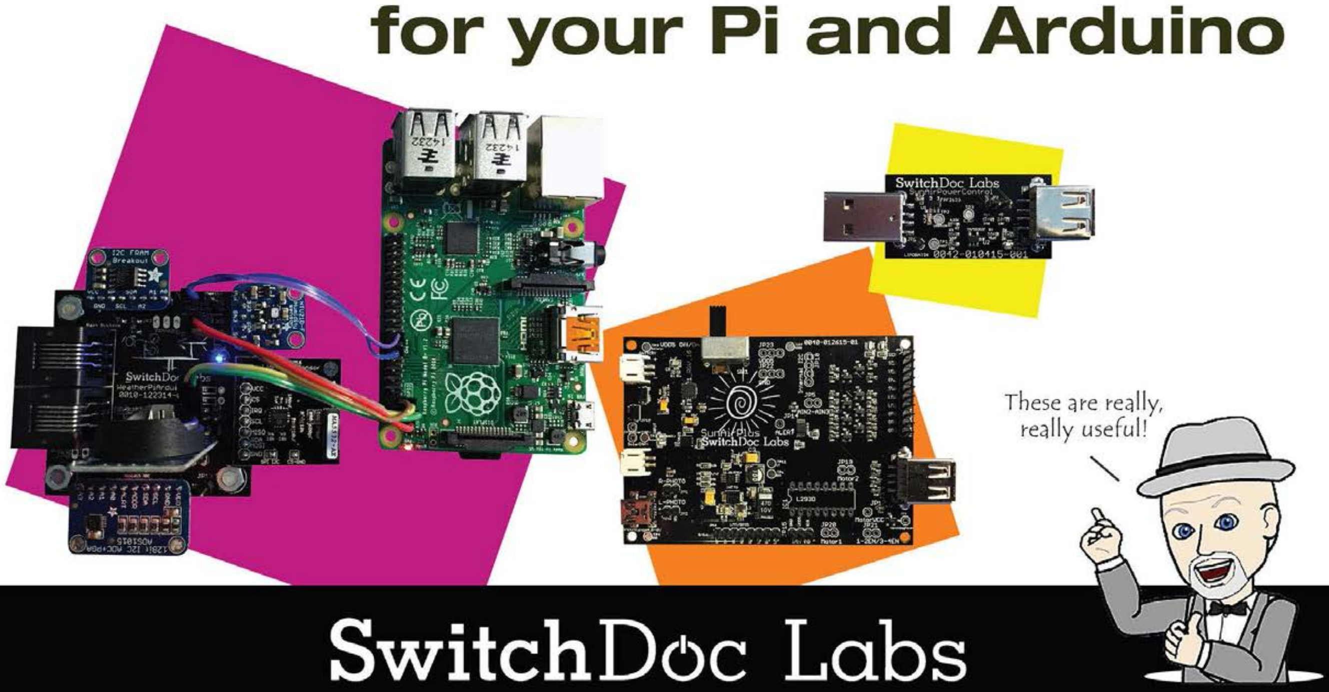
"If you've got a hardware idea," agrees Paul, "keep it simple, keep it true to what the core of it is, make sure you've got an audience, and make sure you've had lots of good feedback and criticism of it so it's polished."

Experience is key, adds Saar Drimer, and all too often lacking in those turning to Kickstarter to get their project off the ground. "Multiply the time you think it would take by three, then by your level of confidence," he advises, "where one is 'very confident', and three is 'not confident at all.'"

For those on the other side of the table, who are looking to back projects, Saar has a few words of warning. "Your payment is a gamble, and that's what Kickstarter was meant for. If something really appeals to you, support it – and treat the experience of actually getting it at all as a pleasant surprise."

Gareth Halfacree

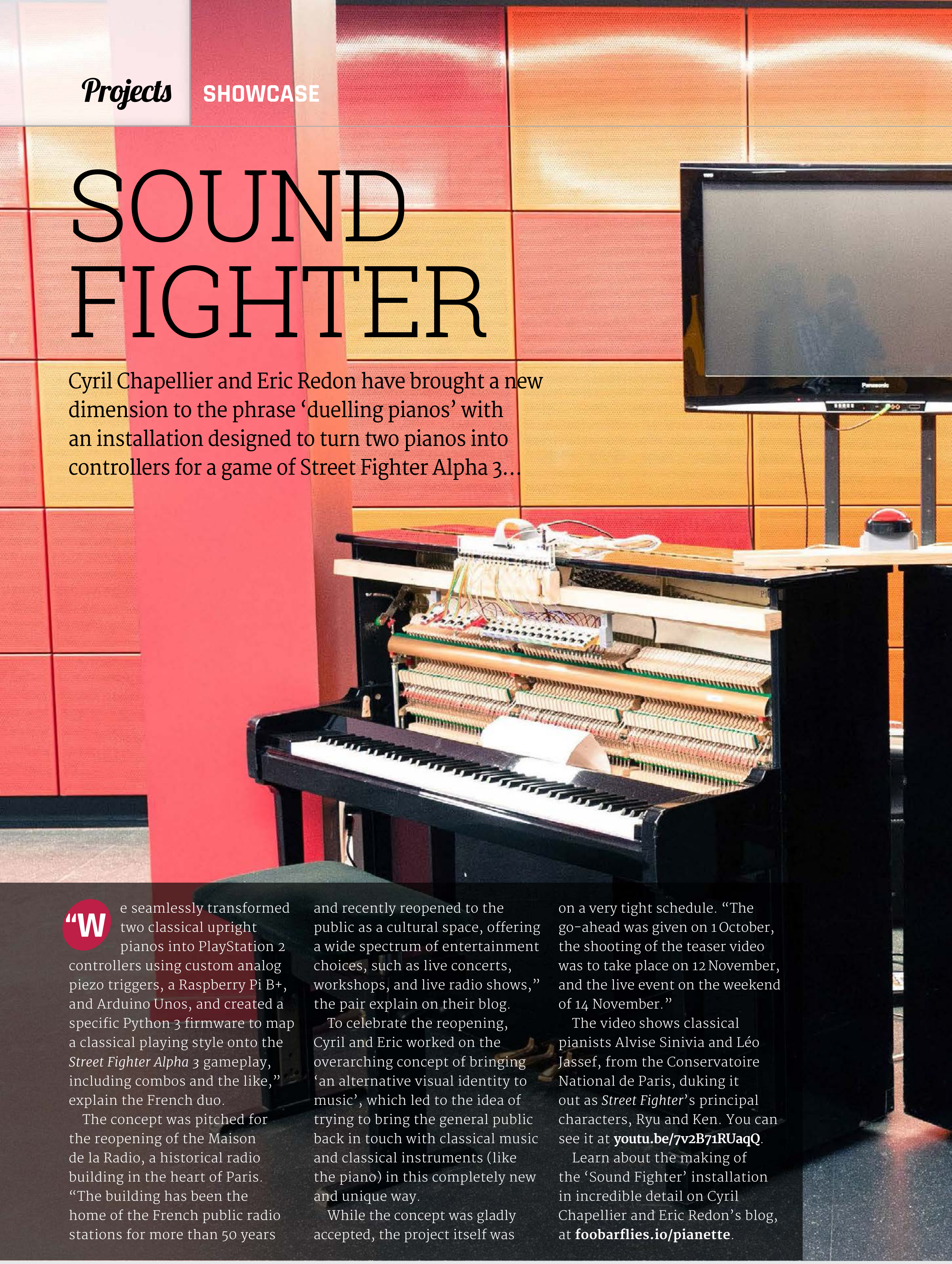
Solar Power, Weather and Breakout Boards for your Pi and Arduino



SwitchDoc Labs

switchdoclabs.com

© 2015 SwitchDoc Labs, LLC



SOUND FIGHTER

Cyril Chapellier and Eric Redon have brought a new dimension to the phrase 'duelling pianos' with an installation designed to turn two pianos into controllers for a game of Street Fighter Alpha 3…

"We seamlessly transformed two classical upright pianos into PlayStation 2 controllers using custom analog piezo triggers, a Raspberry Pi B+, and Arduino Unos, and created a specific Python 3 firmware to map a classical playing style onto the *Street Fighter Alpha 3* gameplay, including combos and the like," explain the French duo.

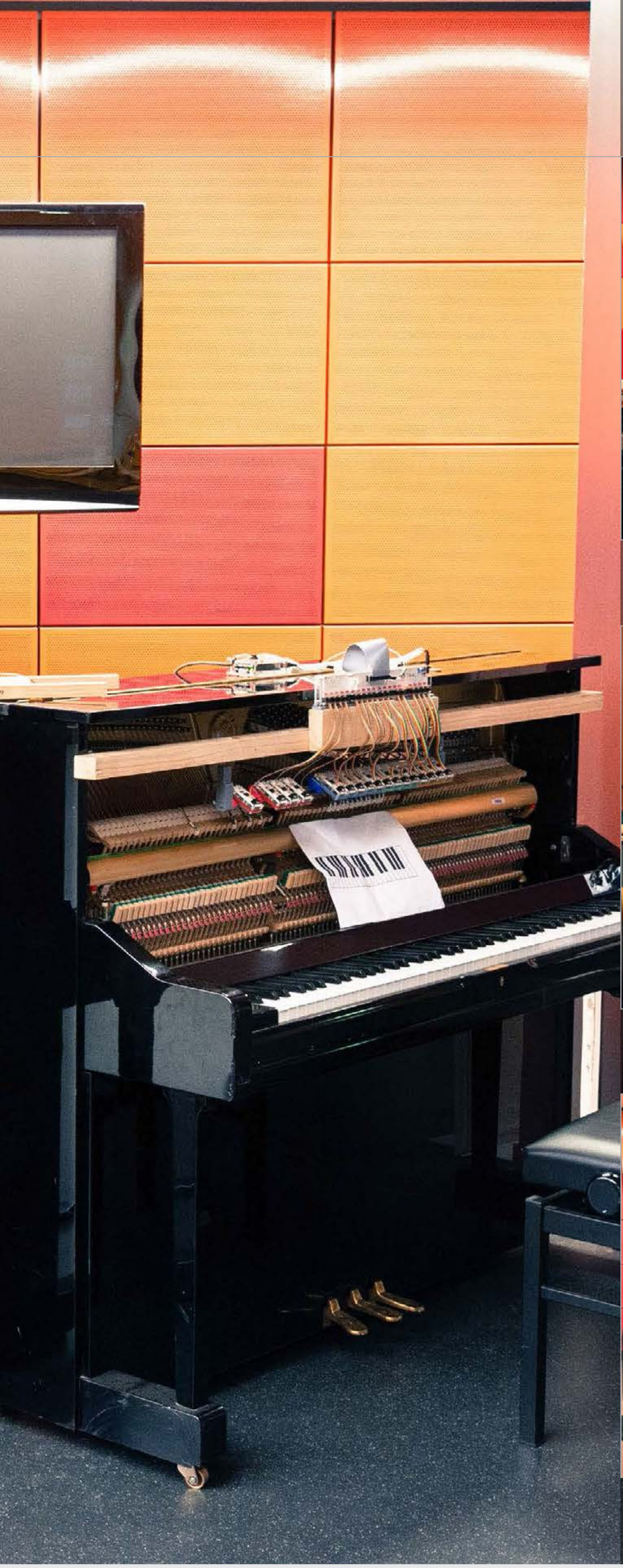
The concept was pitched for the reopening of the Maison de la Radio, a historical radio building in the heart of Paris. "The building has been the home of the French public radio stations for more than 50 years and recently reopened to the public as a cultural space, offering a wide spectrum of entertainment choices, such as live concerts, workshops, and live radio shows," the pair explain on their blog.

To celebrate the reopening, Cyril and Eric worked on the overarching concept of bringing 'an alternative visual identity to music', which led to the idea of trying to bring the general public back in touch with classical music and classical instruments (like the piano) in this completely new and unique way.

While the concept was gladly accepted, the project itself was on a very tight schedule. "The go-ahead was given on 1 October, the shooting of the teaser video was to take place on 12 November, and the live event on the weekend of 14 November."

The video shows classical pianists Alvise Sinivia and Léo Jassef, from the Conservatoire National de Paris, duking it out as *Street Fighter*'s principal characters, Ryu and Ken. You can see it at **youtu.be/7v2B71RUaqQ**.

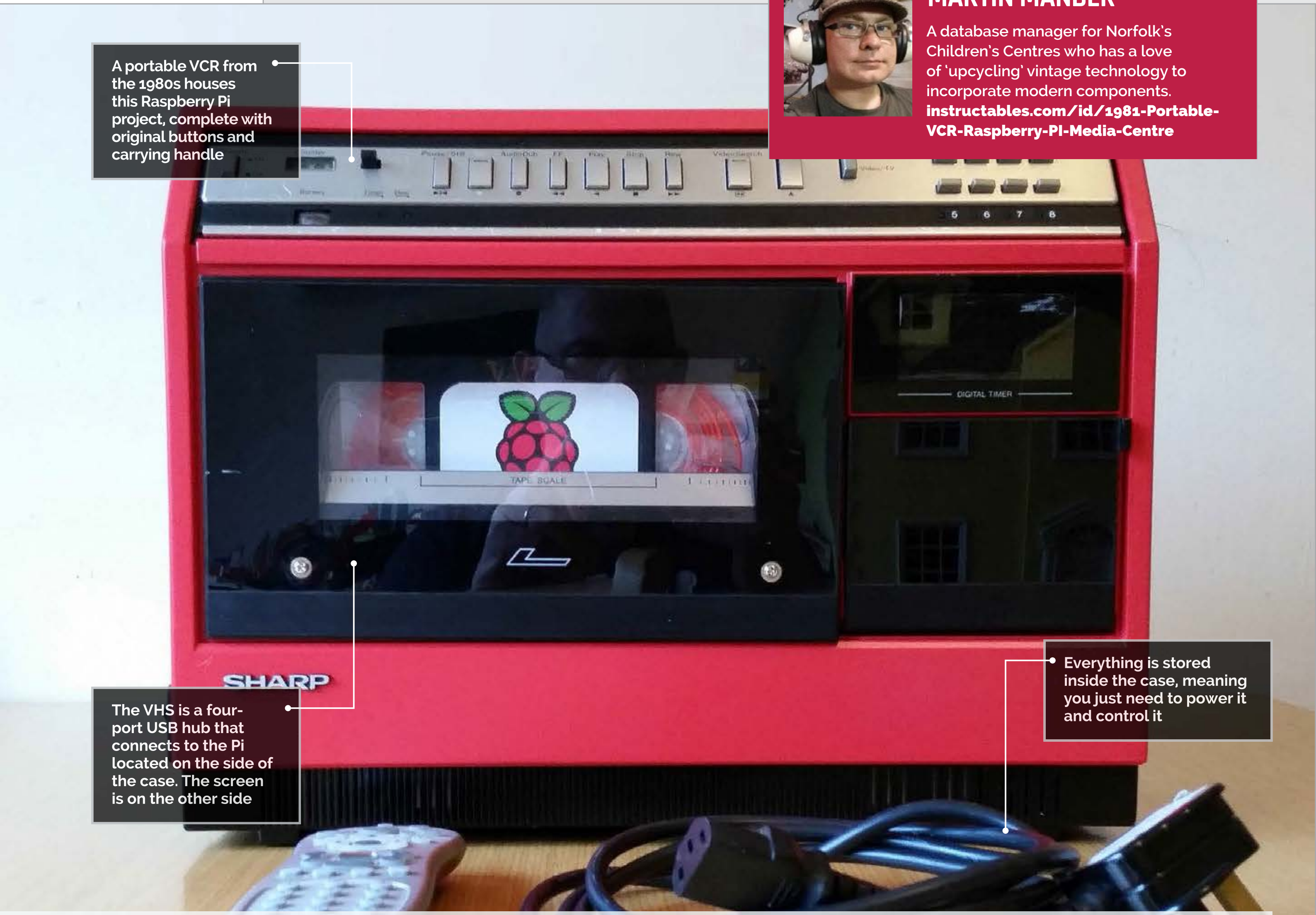
Learn about the making of the 'Sound Fighter' installation in incredible detail on Cyril Chapellier and Eric Redon's blog, at **foobarflies.io/pianette**.



Above The project cleverly interfaces a piano to a PlayStation 2 controller so the game can be 'played'

Above The public were invited to test this novel take on 'duelling pianos'

Above Alvise Sinivia and Léo Jassef, from the Conservatoire National de Paris, demonstrate the installation



A portable VCR from the 1980s houses this Raspberry Pi project, complete with original buttons and carrying handle

The VHS is a four-port USB hub that connects to the Pi located on the side of the case. The screen is on the other side

Everything is stored inside the case, meaning you just need to power it and control it

MARTIN MANDER

A database manager for Norfolk's Children's Centres who has a love of 'upcycling' vintage technology to incorporate modern components.
instructables.com/id/1981-Portable-VCR-Raspberry-PI-Media-Centre

Quick Facts

- The project took six months to complete
- Martin has many similar broken devices ready to upcycle
- His next project is to upgrade a Seventies cassette player
- The Pi community helped a lot during construction
- This is Martin's first major Pi project

RASPBERRY Pi VCR

An Eighties video player or portable digital entertainment centre? Spoilers: it's the latter, and it looks truly amazing...

Portable video playing is a modern concept, right? Before magical internet-connected phones in our pockets, there were portable DVD players in cars. Laptops had DVD drives and video software a bit before that as well, so perhaps it's a little older than you might initially consider.

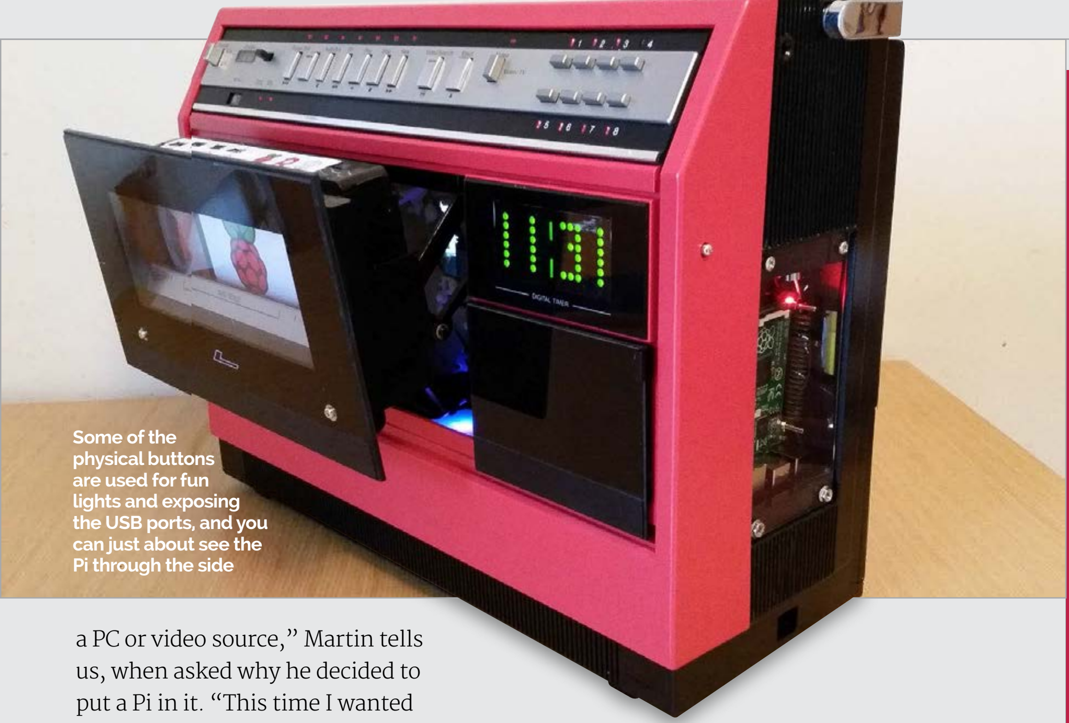
Meet the Sharp VC2300H and have your world turned upside down. This contraption from the ancient past of 1981 was able to play video stored on magnetic tape and housed in a big plastic rectangle – also known as a VHS. If you're not old enough to know what one of those is, ask your parents (or better yet, Siri). You did have to hook it up to a TV, but being able to lug it around was quite a novelty really, taking inspiration from portable stereos of the age.

This version is slightly different, though, as its creator Martin Mander explains very succinctly:

"I picked [this] up for 'spare or repair' on eBay for £6 – a top-loading VCR that unusually stands upright and has a carrying handle. I stripped out all of the internal circuits and replaced them with modern tech, with the Pi running the show, a powered USB hub housed in a pop-out VHS tape, an Arduino-powered clock, and a 15˝ HD TV panel integrated into the back of the unit."

The whole setup runs Raspbmc, the XBMC spin of Raspbian, and also allows you to stream from places like YouTube and the BBC iPlayer via Wi-Fi. There's even a built-in IR sensor for media remotes.

"I'd made several other projects combining retro TVs and LCD panels before, but these were always tied to


Some of the physical buttons are used for fun lights and exposing the USB ports, and you can just about see the Pi through the side

a PC or video source," Martin tells us, when asked why he decided to put a Pi in it. "This time I wanted to make a much more interactive all-in-one device. The Raspberry Pi looked like the ideal solution on form factor alone and when I nosed around at the fan and support sites, I was impressed by the scale and enthusiasm of the community. I'd experimented with APC (Android PC) boards in the past, but a lack of updates and support turned me off taking these any further."

Although all the physical buttons on the original machine still mostly worked, Martin was unable to get them all working with the Raspberry Pi: "I spent several weeks looking into different ways of connecting up the hardware buttons to the Pi via USB, working my way through a series of stripped down gamepads and keyboards. One evening I wondered if I could maybe use an Arduino to mimic key presses, then the more I researched, it struck me that the Pi itself had GPIO ports and the buttons could possibly be connected directly... Ultimately, I think the 30+ year-old switch circuit was the reason I couldn't get this to work; in retrospect, I should have replaced the old push buttons with modern microswitches – next time! With the case already painted and the TV panel installed, I settled for just integrating the basic Play/Rewind/Fast-Forward buttons with the Pi, which I did using the circuit from a pound shop USB mouse."

> I stripped out all of the internal circuits and replaced them with modern tech, with the Pi running the show

While some more work may be required to perfect this fascinating project, overall it works really well, according to Martin. So if you ever find an old portable VCR at a car boot sale, maybe you can make yourself a cheap, portable media player with retro charm.

Below There's a custom paint job and a bit of Pi branding, but really you'd have to take a close look to realise it's different

GET WATCHING!

>STEP-01
Turn it on!

There are several buttons required to turn the full thing on. First the switch for the mains supply, then the switch to activate the lights and clock and other fun things, and finally the switch for the Pi.

>STEP-02
Add peripherals

Need to add some storage or a USB keyboard/mouse? Eject the cassette. That's where the USB ports are for the Pi – don't worry, you don't need to put it back in.

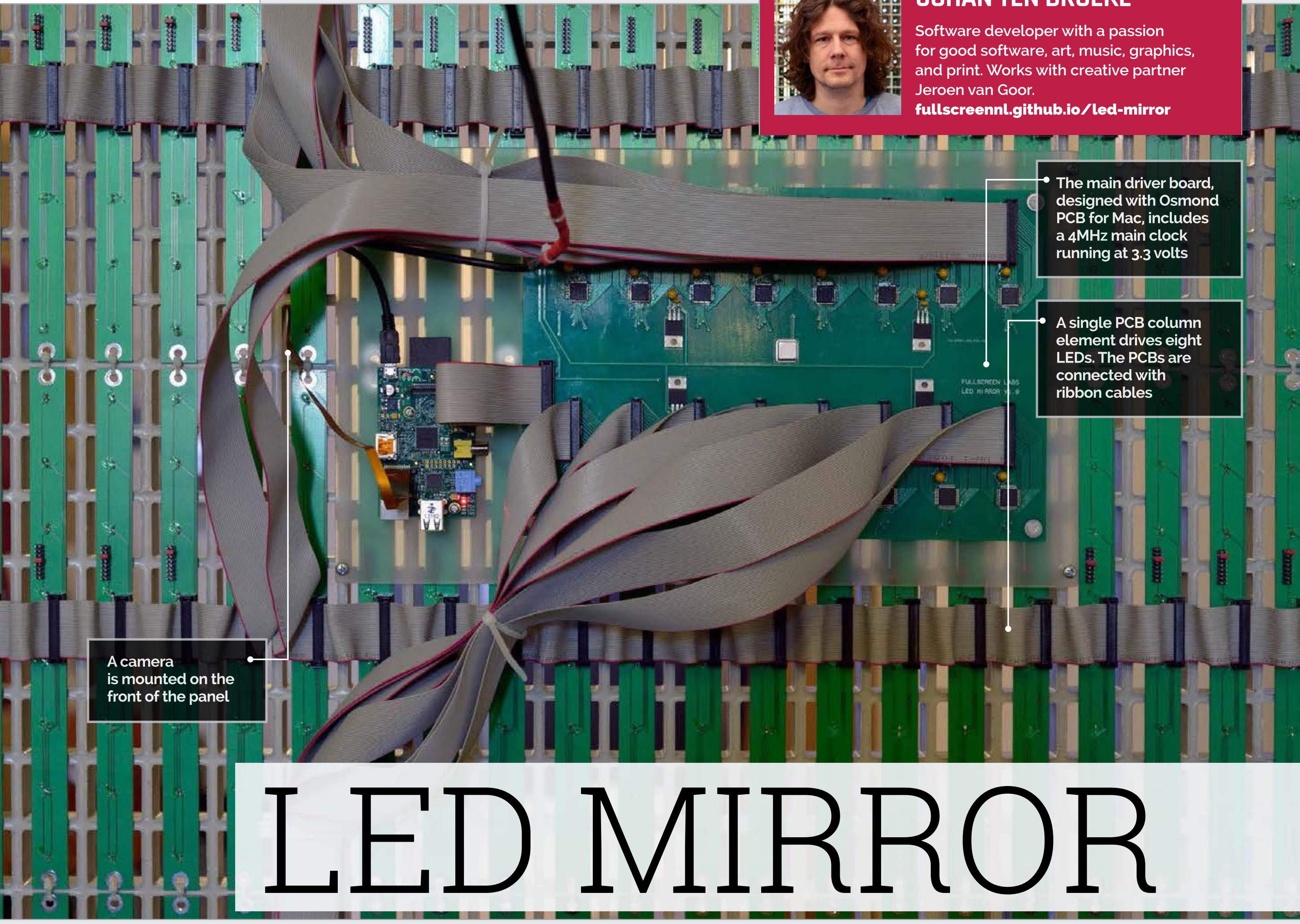
>STEP-03
Select a show

Raspbmc is running on the screen installed on the rear. Using an IR remote, you can select from local media or streamable videos on any supported service.

JOHAN TEN BROEKE

Software developer with a passion for good software, art, music, graphics, and print. Works with creative partner Jeroen van Goor.

fullscreennl.github.io/led-mirror



The main driver board, designed with Osmond PCB for Mac, includes a 4MHz main clock running at 3.3 volts

A single PCB column element drives eight LEDs. The PCBs are connected with ribbon cables

A camera is mounted on the front of the panel

LED MIRROR

It's not every day that you are able to see yourself in a whole new light, but the LED mirror has managed to crack it

Quick Facts

- The panel is just 50mm thick
- The camera is smaller than a coin
- More than 272 connectors were clamped
- The mirror is displayed at Fullscreen.nl's Netherlands office
- Fullscreen.nl created BlackStripes, a Pi-based drawing bot

We wouldn't say Johan ten Broeke was a vain man, but he does spend an inordinate amount of time in front of a mirror. Catch his ghostly white, silhouetted reflection, though, and you will see exactly why – for this is no ordinary mirror, but a large and rather special panel powered by a Raspberry Pi.

Created as an art installation, the mirror consists of a staggering 2,048 LEDs. When someone stands in front of it, a camera picks up their movement and creates a series of snazzy effects. "Myself and Jeroen van Goor are imaging enthusiasts and we got the idea for the project when we were playing around with a RaspiCam camera module," Johan says. "We thought it would be an entertaining device."

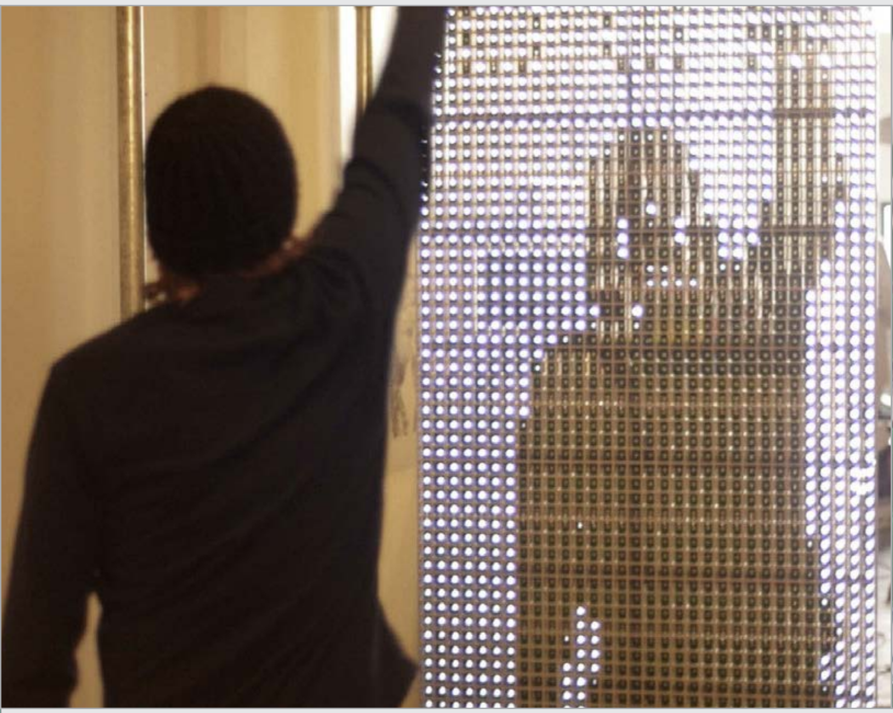
Eating & sleeping LEDs

The project began on a small scale. "We built a prototype of several small 8×8 LED dot-matrix digits," Johan explains. It soon grew. Before long, the pair were connecting LED-strip PCBs together using ribbon cables, strapping them to an off-the-shelf reinforced floor grate using cable ties, and hand-soldering hundreds upon hundreds of LEDs at a pitch of 38mm apart, in a hugely repetitive process which took them both three days.

The effect is nothing less than stunning, however. "Although the displayed images are very abstract, spectators identify themselves instantaneously due to the feedback of their motions. This triggers people to move and wave in front of the mirror," says Johan.

Best of all, the mirror is adaptable. The horizontal spacing of the LED grid can be varied to compensate for narrow viewing angles, while the slim design of the LED modules allows the lights to be applied on curved surfaces. It gives the mirror wide-ranging potential.

"Some people have suggested the LED mirror belongs in a nightclub-like environment, close to a DJ set or something like that," enthuses Johan. "It certainly draws a lot of attention, so perhaps shopping



windows and interiors would be a good setting. It could be used for game-assisted physical therapy, for people who are recovering from injury and need exercise. Patients could 'hit' moving virtual sensors and the software app could produce progress reports to the therapist."

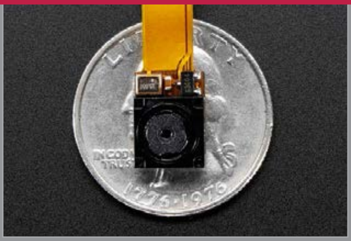
> Some people have suggested the LED mirror belongs in a nightclub-like environment, close to a DJ set or something like that

There's an app for that

Johan and Jeroen have written four main apps to accompany the mirror. The home app has 4×4 pixel sensors which are triggered when the change in pixel value exceeds a threshold; the recording app retains 10 seconds of display data at 15 frames per second and plays them back in reverse; the difference app displays a silhouette of spectators; and the drawing app only allows the brightest pixels to come through to the screen, allowing users to draw in white on a black canvas (or vice versa), with a flashlight or similar device.

"It's really adaptable and it could be made even bigger than the current screen size of 122cm by 244cm," Johan says. "The Maxim chips allow for a maximum of 256 cascaded devices on a single SPI bus and we use 16 chips right now. You could use a second SPI device in the Pi, allowing for a second 256-chip chain, making it 32 times as big." To which, on reflection, we can only stand in amazement.

USING THE LED MIRROR



> STEP-01

Plug in and play

The LED mirror is plugged into the mains and once it is switched on, you need to stand in front of it. There is a tiny camera mounted in the centre of the installation.

> STEP-02

Select a mode

The home app gives a generic view of the camera and accesses the other apps. Recording mode lets you create an animation. The difference mode lets you admire the view.

> STEP-03

Create a picture

In the drawing mode, grab a flashlight and create cool pictures on the mirror's 32×64 pixel canvas. After a while, the recording, difference, and drawing apps return to the home app.

Top Left As the user moves in front of the mirror, the LEDs power off to reflect the motion

Top Right A total of 2,048 LEDs were needed for the Raspberry Pi LED mirror project

JACK SMITH

A sales and IT assistant in the manufacturing industry, Jack knows his way around computers, but this was his first Pi project.

RASPBERRY Pi ARCADE

Save money at the arcade and shoot down space invaders from the comfort of your own home…

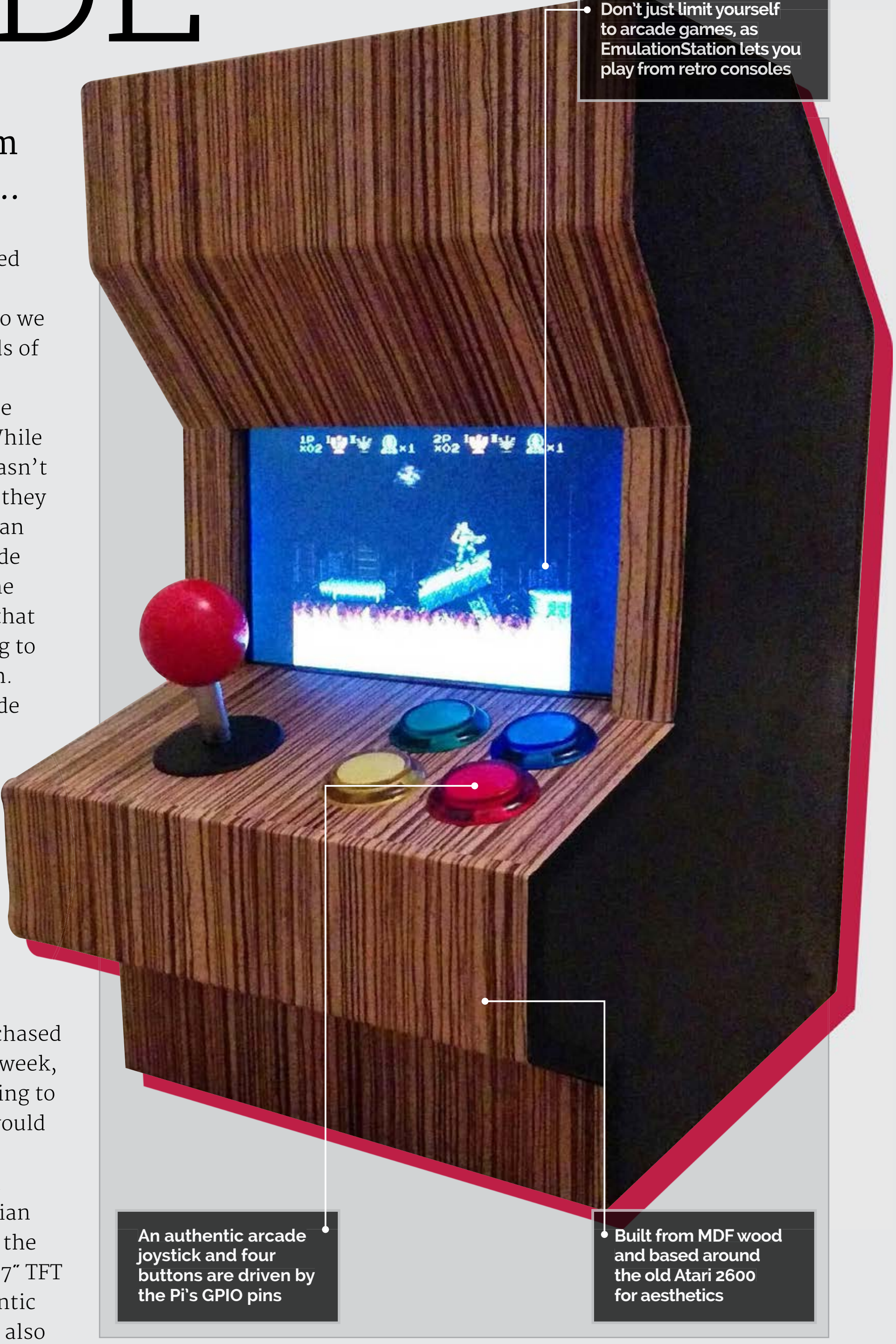
As kids, we all dreamed about having arcade machines at home so we wouldn't have to spend loads of money at the arcade to beat *X-Men* or figure out if Reptile was real in *Mortal Kombat*. While home consoles existed, it wasn't until the mid-Nineties that they came close to the quality of an arcade experience. The arcade was an experience, albeit one now lost to time; however, that hasn't stopped people trying to recreate it with modern tech. Enter the Raspberry Pi Arcade by Jack Smith.

"The idea was heavily inspired by the iCade by ION," Jack tells us, referring to the iPad dock that makes it look like an arcade cabinet. "I really liked the idea, but I don't have an iPad and I wanted something much more open. I had purchased a Raspberry Pi the previous week, which I had already been using to play games, and figured it would be the perfect match."

The Pi Arcade itself uses a Raspberry Pi running Raspbian with EmulationStation over the popular RetroPie, and has a 7˝ TFT display along with an authentic joystick and four buttons. It also

Quick Facts

- The project took two weeks to complete
- *OutRun 2* is Jack's favourite arcade game
- The original plan was to gut a Game Boy or SNES
- His next project is a light-gun arcade cabinet
- Jack's favourite game to play on the Pi Arcade is *Contra III*



Don't just limit yourself to arcade games, as EmulationStation lets you play from retro consoles

An authentic arcade joystick and four buttons are driven by the Pi's GPIO pins

Built from MDF wood and based around the old Atari 2600 for aesthetics

> I really wanted that retro look, similar to that of an Atari 2600 or a Moog synthesizer

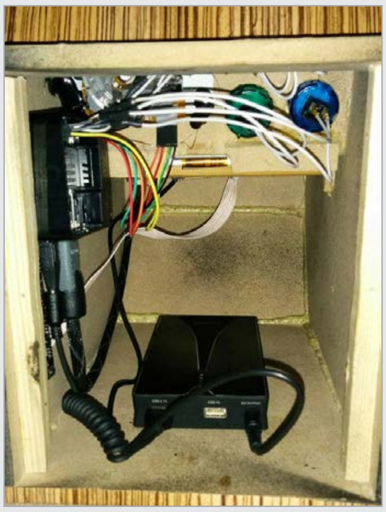
connects wirelessly for Jack to maintain it and add games. As for the cabinet itself, Jack made it himself:

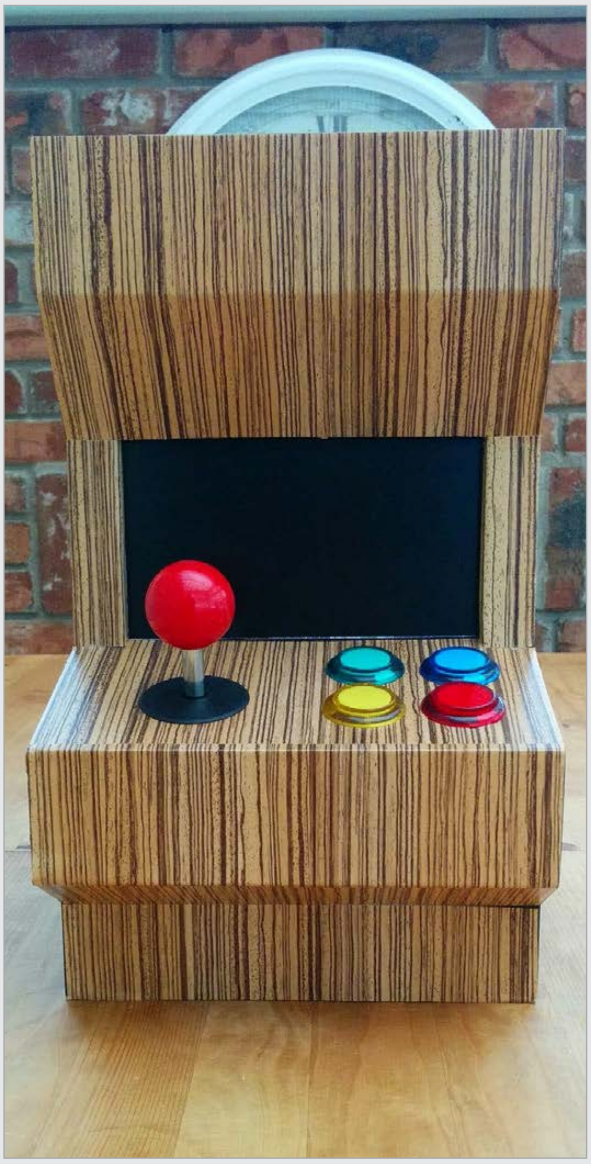
"I designed the cabinet from scratch. I used Sketchup for the plan, mainly because it's quick and easy to use and also gave me the exact measurements required to make it out of wood. I'm not the best craftsman, so a friend of mine who makes children's toys from wood was a big help. The cabinet is made from MDF, but I really wanted that retro look, similar to that of an Atari 2600 or a Moog synthesizer, so I used black paint and some rolls of Fablon to achieve that.

"It works surprisingly well," continues Jack. "The one issue, as many people have said, is that it only has four buttons. I intended it to have six, but I had miscalculated the size of the buttons and could only fit four. The controls that are on it, however, feel responsive and very similar to the real thing. Most of the emulators work very well; however, it does struggle with PS1 and some MAME games, but hopefully the newer model of the Pi will give it the extra muscle it needs."

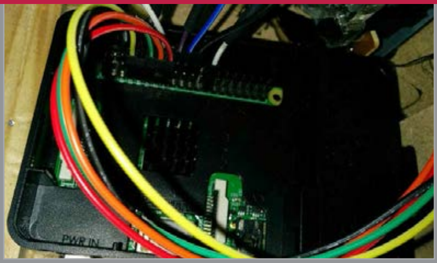
While it may not have quite enough buttons on the cabinet, you can plug in an Xbox 360 controller to have a little more control over the games – although it may ruin the experience a little.

"It was a great feeling [finishing the Pi Arcade]. The finished product really did take me back to the old bartop arcade machines I remember playing on holiday when I was a kid."


Above A hollowed-out interior houses the Raspberry Pi and the wires for the authentic arcade parts

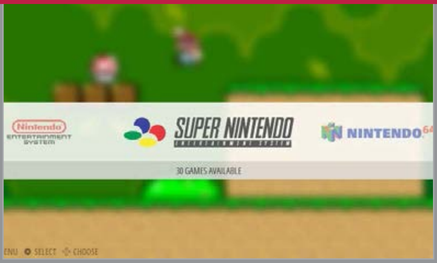

Above Small but mighty – the Pi Arcade can play games from over a 25-year time span

INSERT COIN TO PLAY



>STEP-01
Power it up

The machine is powered by a portable battery – plug it into the Pi and activate it for the arcade machine to start turning on. It should run fine for a few hours.



> STEP-02
Choose your emulator

After loading EmulationStation, you can choose what to play on as long as you have some appropriate games to load on that particular emulator.

> STEP-03
Just have fun

Play using the controls or by plugging an Xbox 360 controller into it, and just enjoy the retro arcade experience on your sofa with no sweaty men to jostle with.

BIG BIRTHDAY BASH

Here's what happened when 1,300 people helped the Raspberry Pi celebrate its third birthday where it all started, at the Cambridge Computer Lab...

An incredible weekend of people, Pi projects, and pizza. At least that's how one partygoer described Raspberry Pi's huge third birthday bash, and we'd have to agree. While previous birthday gatherings have always been impressive, February's bash at Cambridge University's Computer Laboratory couldn't have been better. After all, what better place to celebrate its success than where our favourite credit card-sized PC acquired its name?

While lots of people had a hand in making it memorable, it wouldn't have happened without the hard work of two particular Raspberry Jam aficionados – Michael Horne and Tim Richardson. Mike and Tim have been organising Raspberry Jams in Cambridge (**CamJam.me**) since the early days, and they know a thing or two about putting on a great event. They really excelled themselves here, too, with a staggering array of talks, workshops, and truly unforgettable Raspberry Pi projects.

Image: Ben Nuttall, bennuttall.com

Visitors were able to announce their arrival in style

Eben managed to maintain an air of sophistication, even under extreme stress

The amazing throne of wonder

Arguably, the crowning glory of all the Raspberry Pi projects on show over the weekend was the Announcing Throne, which took pride of place directly opposite the entrance in the Computer Lab's foyer. The installation was originally commissioned as part of Kensington Palace's Christmas celebrations, inspired by a series of parties and events held in 1699 by King William III, in which visitors could be named king or queen for a day. This Pi-powered throne was created by Henry Cooke and Tim Burrell-Saward of **elkworks.co.uk** and does much more than look rather special. Thanks to its Gen 2 RFID technology (making it able to read lots of tags simultaneously and over longer distances), users can pick a number of random words mounted on RFID-backed name cards to compose their ideal introduction that slides into an ornate gold frame. Pete Lomas, for example, opted for 'The messy inventor of wonderland'. Sitting on the throne triggers the playback of your chosen title to announce your arrival. A very clever project and a very memorable part of the celebrations.

Another Pi-powered project that Raspberry Pi creator Eben Upton is unlikely to forget any time soon was BigHak, a fantastic life-size tribute to the classic '80s toy, BigTrak. Constructed by members of Hitchin Hackspace (**HackHitchin.org.uk**), it was a particularly fitting tribute because, for many 80s kids (us included), the BigTrak represented their first taste of programming, since it required the user to type commands to get it to move, much like a 'turtle bot'. Using a combination of electric wheelchair parts and Arduino and Raspberry Pi electronics, the BigHak made a fantastic attraction for younger vistitors, not least because it could be steered by a remote control held by a third party as well as the driver. Eben certainly managed to build up quite a head of steam when he took it outside for a test drive. The Vine that Liz captured of Eben's journey is sure to become the stuff of legend. You can see it at **vine.co/v/O2lg67jp0l5**.

THE BIRTHDAY BASH IN NUMBERS

1,300
Attendees over two days

110
Pizzas snaffled on Saturday

24
Entertaining talks

02
Regularly packed lecture theatres

14
Amazing workshops

1,000
FREE Raspberry Pi cases!

Ben Croston, creator or RPi.GPIO, pulling a pint of his specially brewed Irration Ale

Image: Alex Eames, RasPi.TV

Getting the party started

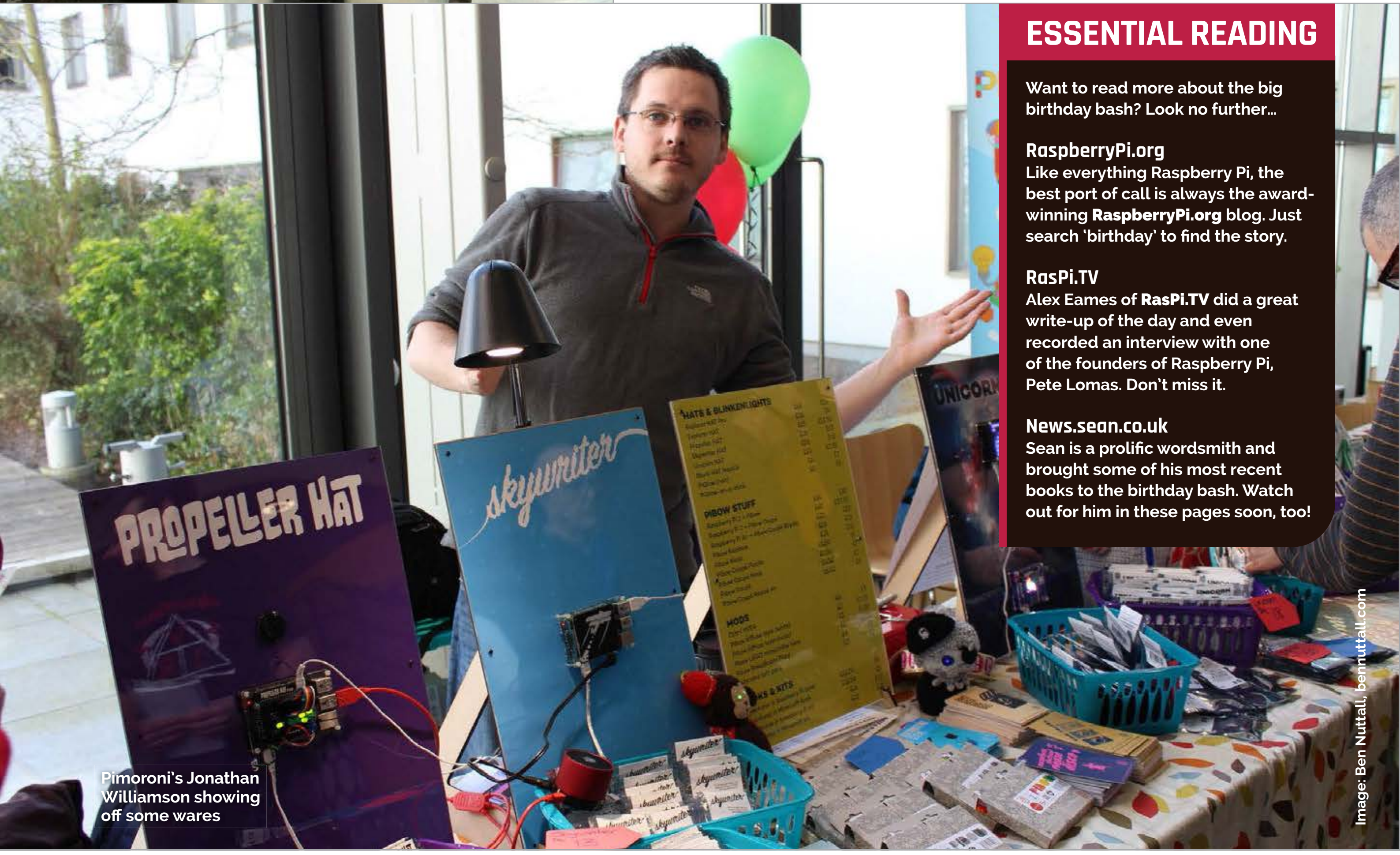
So how do you get a Raspberry Pi party started? The promise of alcohol is a pretty sure-fire hit, and one of the biggest treats for partygoers on the Saturday night (at least, the adults among them) was a rather fantastic pint of beer, courtesy of Ben Croston. His

Ben runs Fuzzy Duck Brewery (**fuzzyduckbrewery.co.uk**), where he uses the Raspberry Pi to help manage the beer brewing process. He's also the creator of **RPi.GPIO**, the Python package we all use to control the general-purpose input/output (GPIO) pins on the Raspberry Pi. Of course, it's

> "Ben Croston's Irration Ale went down a treat with all who tried it"

Irration Ale (a neat play on words, since pi is an irrational number) went down a treat with all who tried it. PiBorg's Claire Pollard is a bit of a beer connoisseur, as it turns out, and she certainly enjoyed it: "Not overly sweet from the raspberry, but you could definitely taste it. Shame it was limited to the birthday bash!"

never wise to drink on an empty stomach, which is why a grand total of 110 pizzas were consumed at breakneck pace, along with a fully-fledged birthday cake and a selection of brilliant Raspberry Pi-themed cupcakes. We can't wait for next year's bash to celebrate the Pi's first 'real' birthday on 29th February 2016!



Pimoroni's Jonathan Williamson showing off some wares

Image: Ben Nuttall, bennuttall.com

ESSENTIAL READING

Want to read more about the big birthday bash? Look no further…

RaspberryPi.org
Like everything Raspberry Pi, the best port of call is always the award-winning **RaspberryPi.org** blog. Just search 'birthday' to find the story.

RasPi.TV
Alex Eames of **RasPi.TV** did a great write-up of the day and even recorded an interview with one of the founders of Raspberry Pi, Pete Lomas. Don't miss it.

News.sean.co.uk
Sean is a prolific wordsmith and brought some of his most recent books to the birthday bash. Watch out for him in these pages soon, too!

BIRTHDAY WORKSHOPS

Here's a small sample of the 14 workshops, covering everything from setting up a Raspberry Pi for the first time, to creating Internet of Things devices with Node.js, and everything in between…

Introduction to Minecraft: Pi
Aimed at beginners, the Introduction to *Minecraft: Pi* helped kids and adults learn how to 'hack' *Minecraft*, while learning more about the popular programming language Python.

Scratch Hackathon
Working in teams, participants of this hackathon used the visual programming language Scratch (and Scratch GPIO) to make games and physical computing projects.

PiCamera workshop
Dave Jones, creator of the excellent PiCamera Python library to control the Raspberry Pi Camera, did an excellent job of teaching people how to take snaps with the Camera module.

Sonic Pi
Participants learnt the basics of making music with Sonic Pi, before being introduced to the concept of 'live coding' – something Sonic Pi's creator, Sam Aaron, demonstrated excellently on Saturday night.

Basic electronics
Using the CamJam EduKit (**camjam.me/edukit**), this workshop demonstrated beginner GPIO projects with the affordable, easy-to-use electronics kit.

Below Left Martin O'Hanlon and Sam Aaron

Below Darby and Matt Timmons-Brown

```
Last login: Wed May 28 12:04:11 EDT 2014 on
ttyAMA0
Linux raspberrypi 3.10.25+ #622 PREEMPT Fri
Jan 3 18:41:00 GMT 2014 armv6l

The programs included wipi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
pi@raspberrypi:~$
```

PiConsole

A clever solution to Command Raspberry Pi Console from your Android or iPhone!

Pi-Pan

a Complete Pan-Tilt kit for Raspberry Pi Camera

10% OFF !!!

Use Code at checkout: MAGPIQE4WE

OpenElectrons.com

EVERYDAY ENGINEERING PART 2

SIMON MONK

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

simonmonk.org
monkmakes.com

BUILD YOUR OWN

WEB DOOR LOCK

You'll Need

- Half-size breadboard
- 2x male-to-female jumper wires
- 2x male-to-male jumper wires
- 4x 470 ohm resistors
- TIP120 transistor (CPC: SC10999)
- 12V DC door latch (CPC: SR04745)
- DC barrel socket to screw adaptor (CPC: CN19646)
- 12V DC 2A power supply
- 1N4001 Diode
- Red LED

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

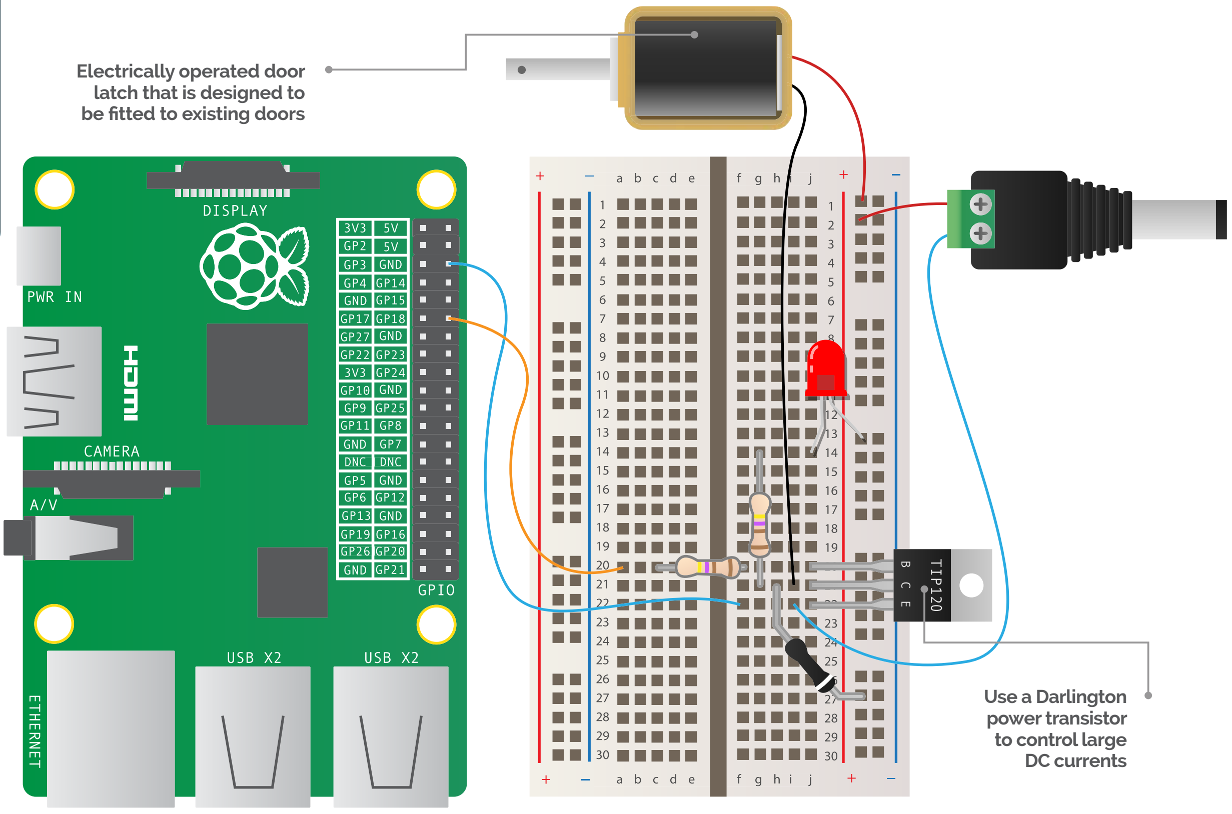
In the second of our series on Everyday Engineering, we show you how to use a Raspberry Pi to control a door latch. The Raspberry Pi runs a web server, to which you can connect using any device equipped with a browser. For instance, as long as you are in wireless network range of your home, you can stand outside your house and connect to the web server with a smartphone, enter a password, and unlock your door.

As you'll see from the list of required components on the left, this project is built on a breadboard and uses a TIP120 Darlington transistor to switch the power to an electric door latch.

We used a TIP120 power Darlington transistor, as these devices are very cheap and reliable. You could also use a logic-level N-channel power MOSFET, such as the FQP30N06, and the pin arrangement is such that you can just put it onto the breadboard in exactly the same position as the TIP120.

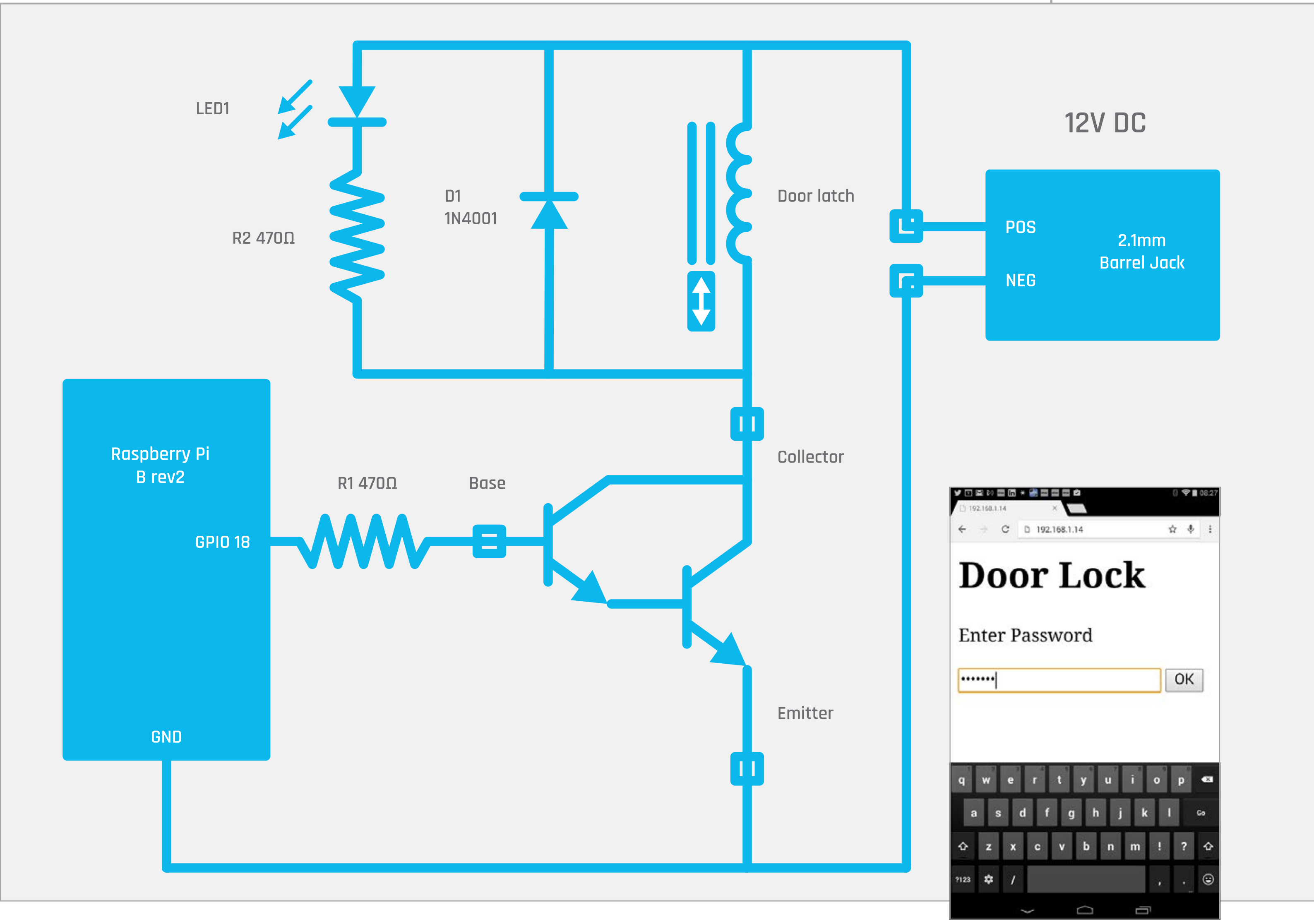
The diode can be pretty much any diode that you would find in a basic kit of components.

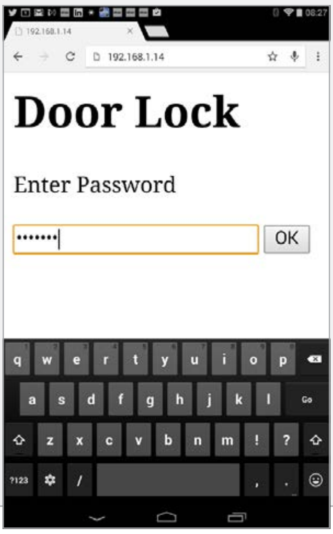
The door latch is of the type designed to replace the existing lock mortice (the hole into which the latch fits) with one that has one hinged edge, which is released when you energise an electromagnet



Electrically operated door latch that is designed to be fitted to existing doors

Use a Darlington power transistor to control large DC currents





Above Unlock your door with your phone

within the latch. While power is applied to the latch, the locked door can then escape through the open side. This has the advantage that, should the electronics stop working, or the Raspberry Pi crash, you won't be locked out of your house, because you can still use your ordinary key to unlock the door.

The DC barrel jack to screw adaptor is a handy little device that lets you connect wires to a standard DC power supply, without having to chop off the lead. However, if you prefer, you can just cut off the plug from the power supply (unplug it first!) – in which case, you don't need the two male-to-male jumper wires, as these are just used to connect the 12V power supply to the breadboard.

Note that the 12V power supply is just for the door latch – you will still need a 5V USB power supply for the Raspberry Pi.

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

How the electronics work

The GPIO pins of the Raspberry Pi are designed to supply a maximum of about 3mA. The door latch needs about 200mA to hold the latch open, with a much higher current when it first energises. A transistor is used, to allow the Raspberry Pi's weedy GPIO pin to control the current to the door latch. The type of transistor used here is called a power Darlington and is actually made up of two transistors, one piggybacking on the other to provide greater current amplification than just one transistor could supply on its own. However, it's all just on one transistor type package and you can use it just as if it were a single transistor.

When the GPIO 18 pin is HIGH, a small current (about 1.5mA) will flow through R2 and into the 'Base' of the transistor. This effectively switches the transistor on, so that a much larger current can flow from the positive side of the 12V power supply through the door latch's electromagnet, releasing the latch. The diode D1 is there to protect the transistor against voltage spikes that happen as the latch is energised.

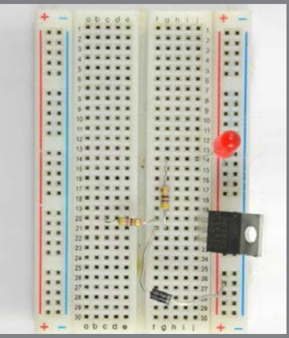
LED1 and R2 are also switched by the transistor, so that when the latch is enabled, the LED will also light, to tell you that the door is now unlocked.

Building your web door lock

As with all projects, it is a good idea to test the project out and get everything working while the parts are all out on your workspace. Once you know all is well, you can box things up and deploy them for real.

BUILDING THE PROJECT

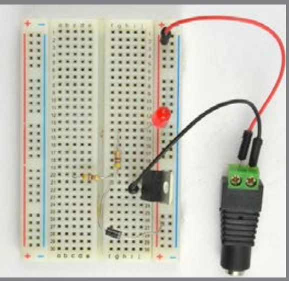
Follow the annotated breadboard diagram, taking care to ensure that all the components and jumper wires fit into the correct rows on the breadboard. You also need to make sure that the transistor, LED, and diode are all the right way around – especially the diode, which may well die and take out the transistor with it if it's the wrong way around.



>STEP-01

Build the breadboard

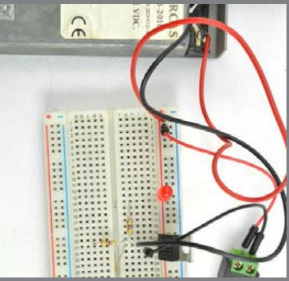
Plug the components into the breadboard, as shown in the breadboard diagram. The longer lead of the LED is the positive lead, and this needs to go to the right of the breadboard. The diode has a stripe at one end, and this end must also go to the right. The transistor is oriented with the flat, bare-metal side to the right. If you want to make the layout a bit neater, then shorten the resistor leads so that they don't stick up too high above the breadboard.



>STEP-02

Add the power connections

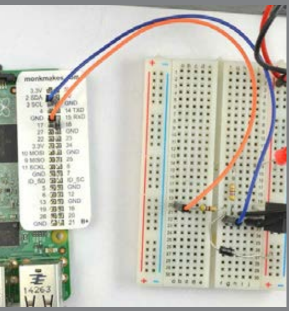
Attach the male-to-male jumper wires to the screw terminals of the barrel jack adaptor. Use red and black leads, and make sure that you attach the red lead to the screw terminal marked +. Then, plug the other ends of the jumper leads into the breadboard. The red lead goes to the right-hand red 'power bus' of the breadboard; the black lead to the bottom of the three transistor leads.



>STEP-03

Attach the door latch

The door latch leads can just be plugged directly into the breadboard, especially during testing. However, when it comes to deploying the project for real, you will probably want to add some longer leads to connect from the breadboard to the door latch. Twin bell cable, or even speaker cable, will work just fine for this. You can join the longer leads to the door latch leads with a terminal block.



>STEP-04

There are just two links to be made between the Raspberry Pi and the breadboard. For these connections, you will need to use male-to-female jumper leads. One lead goes from GND on the Raspberry Pi GPIO connector to the breadboard row at the bottom of the three transistor leads. The other lead goes from GPIO 18 to the left-hand side of R1. Power up your Raspberry Pi, connect the 12V power supply to the breadboard, and turn it on.

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.

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 from your Raspberry Pi command line, using the command:

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

Before you run the program, open **door_lock_server.py** in an editor (such as nano) and change the value of the **IP_ADDRESS** variable to match the IP address of your Raspberry Pi (type **ifconfig** at the command line). You might well also want to change the value of the **PASSWORD** variable from 'letmein' to something a little more secure.

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

```
cd /home/pi/pi_magazine/02_door_lock
sudo python door_lock_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. For our Pi, it was 192.168.1.14. If all is well, a webpage will be displayed, inviting you to enter a password to unlock the door. Enter your password and the LED should light, and the door latch click and unlock.

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**, **RPi.GPIO** and **time** libraries that it needs. Then there are some constants that you may need to tweak. **PASSWORD** is (fairly obviously) the password that must be typed before the door will open, so set it to something fairly obscure.

The constant **MAX_ATTEMPTS** is set initially to 5. This is the maximum number of incorrect password tries before the program will lock you out of any further attempts to open the door. The **OPEN_TIME** constant specifies how many seconds the latch should remain unlocked before it automatically locks again. It should just be open long enough for you to open the door and then shut it behind you.

Next, the GPIO output that will control the lock is set up and the **attempts** variable set to 0, in order to indicate that there have not yet been any unsuccessful attempts to log in.

The **unlock_door** function sets the **LOCK_PIN HIGH** to unlock the latch, delays for the number of seconds specified in **OPEN_TIME**, and then sets **LOCK_PIN LOW** to lock the latch again.

The remainder of the code supplies the web interface. Bottle uses the **@route** directive to indicate that the functions that immediately follow it are handlers 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 all the HTML templates used in the project contained in the code directory for the project. The homepage contains a form with a password field and a Submit button.

The second handler is for the **/unlock** page, and it is this page that the form from the index page is posted to. This function is where most of the action takes place.

First of all, the password that was submitted in the form is retrieved from the request parameter and assigned to the variable **pwd**. Next, the number of attempts is compared with the maximum number allowed and, if it has been exceeded, the **lockout** page is returned, to prevent any further login attempts.

After this, the password from the form is compared with **PASSWORD** and, if they match, then **unlock_door** is called and the number of failed **attempts** reset to 0. If, on the other hand, the password is wrong, 1 is added to **attempts** and the **failed.tpl** webpage displayed.

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 door lock

The first thing to say is that any even vaguely competent security expert reading this probably has tears of laughter coursing down their cheeks at the gross insecurity of this system and the numerous vulnerabilities that could allow someone to gain access to your home. However, keeping the Pi behind your firewall means that the web interface on which an attack could be mounted is only accessible if someone has access to your LAN. Only you can judge if this is secure enough for your home. Don't blame us if someone walks off with all your stuff!

Finally, if you enjoy a bit of webpage design, then 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 files. These are:

home.tpl – homepage containing the password form.

failed.tpl – the page displayed after an incorrect password has been entered.

lockout.tpl – the page displayed after the maximum number of login attempts has been exceeded.

opened.tpl – the page displayed after the door has been unlocked for the specified time.

Door_lock.py

Language
>PYTHON

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

# Change these for your setup.
PASSWORD = 'letmein'
MAX_ATTEMPTS = 5  # before you are locked out
OPEN_TIME = 5     # number of seconds for lock to stay open
IP_ADDRESS = '192.168.1.14' # of your Pi

# Configure the GPIO pin
GPIO.setmode(GPIO.BCM)
LOCK_PIN = 18
GPIO.setup(LOCK_PIN, GPIO.OUT)

# Initialize the number of failed login attempts to 0
attempts = 0

# Unlock the door by setting LOCK_PIN high for
# OPEN_TIME seconds
def unlock_door():
    GPIO.output(LOCK_PIN, True)
    time.sleep(OPEN_TIME)
    GPIO.output(LOCK_PIN, False)

# Handler for the home page
@route('/')
def index(name='time'):
    return template('home.tpl')

# Handler for the 'unlock' page
@route('/unlock', method='POST')
def new_item():
    global attempts
    # Get the password entered in the password field of the form
    pwd = request.POST.get('password', '').strip()
    # check to see if tehre have been too many failed attemts
    if attempts >= MAX_ATTEMPTS:
        return template('lockout.tpl')
    # unlock the door if the password is correct
    if pwd == PASSWORD:
        attempts = 0
        unlock_door()
        return template('opened.tpl')
    else:
        attempts += 1
        return template('failed.tpl')

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

NEXT MONTH

In the next project in this series, we will make a super-loud alarm clock that's capable of even waking teenagers.

LIAM FRASER

Liam is the creator of the RaspberryPiTutorials YouTube channel. He is currently studying Computer Science at the University of York and has a special interest in embedded systems.
liamfraser.co.uk

BUILD A STROBE LIGHT WITH A TRANSISTOR

Using a transistor, it's easy to control lots of LEDs. In this tutorial, we're creating a strobe to help get the party started with our Raspberry Pi…

You'll Need

- White LEDs (anywhere from 1 to 20)
- Breadboard
- A PNP transistor (2N2907)
- 2x push buttons
- Jumper cables
- Resistors (2x 1k ohm & 22 ohm for LEDs)

This guide follows on rather neatly from the tutorial last issue, where we emulated a candle using pulse-width modulation. In this guide we'll step things up a bit by throwing push buttons, a transistor, and lots of LEDs into the mix in an effort to create a strobe light that you can manually speed up and slow down. A strobe is a great accessory to any party and, with enough blue tack or duct tape, you should be able to mount this on a wall. For best effect, place it up high in a corner of a room, right where the two walls meet the ceiling.

>STEP-01

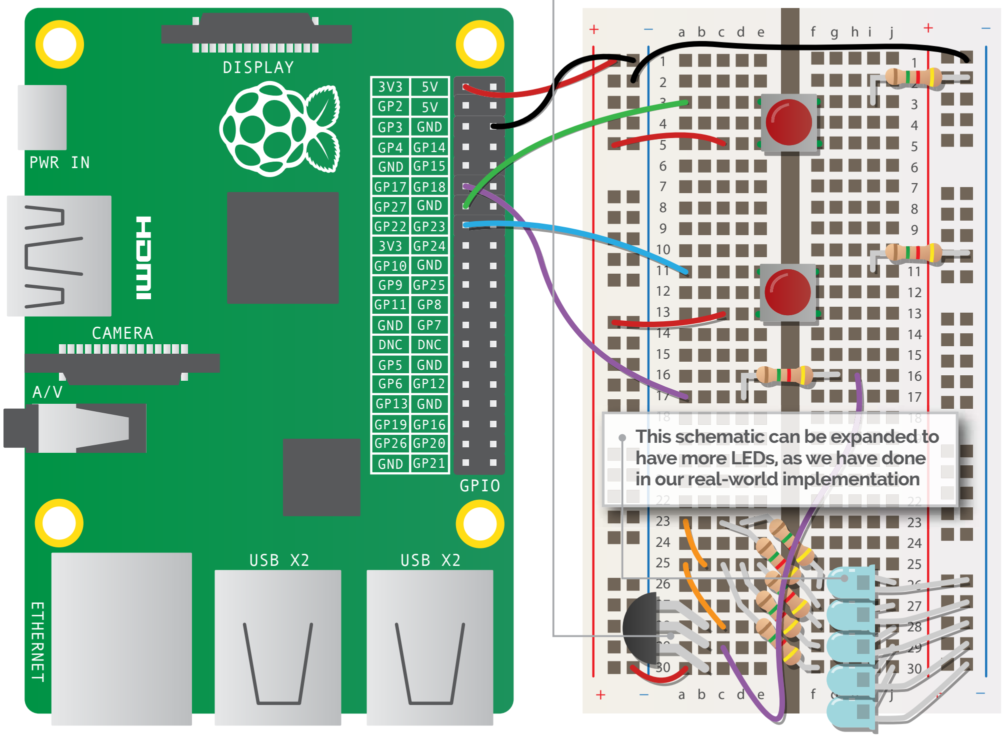
Using transistors

An LED typically uses 20mA of current. Since GPIO pins can't provide enough current to light lots of LEDs simultaneously, we're using a transistor to help us.

The 2N2907 transistor can pass 600mA, which is plenty for our needs. A PNP transistor has three pins: an emitter, a base, and a collector. Here, the emitter is connected to 3.3V, and the base is connected to a GPIO pin via a 1K ohm resistor. This pin is used to switch the transistor on and off. When the pin is 0V, current can flow; when it is 3.3V, no current can flow. Hence, this circuit is inverting: this means that when the GPIO is low, the transistor is on, and the LEDs light.

Since transistors have an amplification factor based on the current that goes through the base, we want to limit it with a resistor. Finally, the collector pin of the transistor is connected to the LEDs and eventually ends up at ground.

The Pins are c, b, e, from top to bottom. The transistor must be inserted this way for the circuit to work correctly



This schematic can be expanded to have more LEDs, as we have done in our real-world implementation

>STEP-02

The circuit path

Each LED path goes from the transistor, through a 22 ohm resistor to limit the current through the LED, and then into ground. Referring to the LED resistance equation from the last tutorial, we only need a small resistor because white LEDs have a forward voltage of about 3V and a current of 20mA: **R = (POWER_RAIL – LED VOLTAGE) / LED CURRENT.**

So: **R = (3.3V – 3V) / 0.020** (which is 15 ohms, rounded up to the nearest practical value here).

WARNING!

This project uses strobe lighting. Do not use it if you or anyone are known to suffer from photosensitivity, never look directly at the LEDs, and never run it for prolonged periods or any faster than four flashes per second.

>STEP-03
Working with push buttons

Electricity always takes the path of least resistance. In the circuit diagram, you can see that a GPIO pin is connected to a leg of each push button. The leg is also connected to ground (0V) via a 1K resistor. This is termed a pull-down resistor because when the button isn't pressed, the signal is pulled down to ground. The other leg is connected to 3.3V. When the button is pressed, the two sides are connected again. There is less resistance to 3.3V and the GPIO input goes high, which we can detect in our code listing on the right.

>STEP-04
Wire up the circuit

Let's get the circuit built, but don't forget to turn your Pi off first. It's important to remember that each LED is connected to the transistor in parallel. This means that they are all effectively connected to the output of the transistor (and not chained in any way). This being the case, you can scale up the design on the breadboard as much as you want, within the specification of the transistor and power supply. In the case of our setup, that would be 20 LEDs or less.

>STEP-05
Event detection

The code for this tutorial is similar to the Candle code in the previous tutorial, with one key difference: we're using events. The functions to speed up or slow down our strobe light will only be called when a rising edge (a transition from 0V to 3.3V) is detected on the button pins. This is a convenient way to handle button presses without having to keep checking them in your main loop.

>STEP-06
Test it out

Create and save the code on the right, then test it out using **`sudo python2 strobe.py`**. Pressing the button on the right will increase the speed of the strobe. Pressing the button on the left will slow the strobe down. Note that because the delay change is small, you'll have to press the buttons quite a few times. As always, you can start the script when you turn on your Pi, by adding it to rc.local:

```
sudo nano /etc/rc.local
```

Add the following line (before **`exit 0`**) and then save the changes:

```
python2 /home/pi/strobe.py &
```

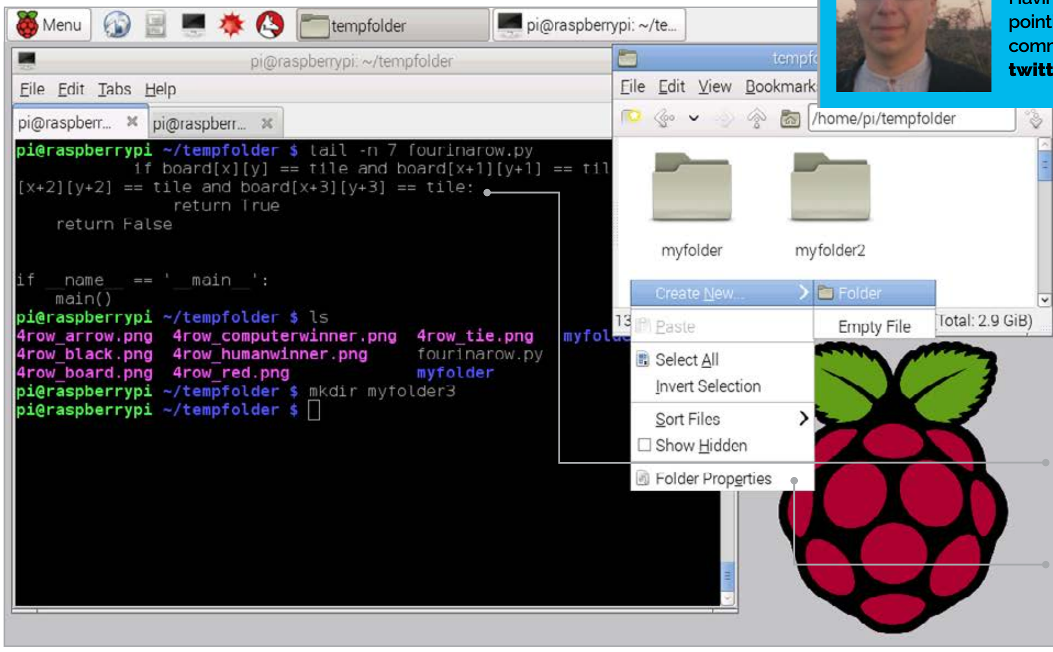
Strobe.py

Language: >PYTHON

```
# Import the GPIO and time library
import RPi.GPIO as GPIO
import time

# First we initialise some constants and variables
TRANSISTOR = 17
BTN_SPEED_UP = 27
BTN_SLOW_DOWN = 22
DELAY_CHANGE = 0.005

# Never use a strobe light any faster than 4 flashes per sec
DELAY_MIN = 0.125 # 1/8 = '4 on 4 off' flashes
delay = 0.2
def setup():
        # Next we initialise setup of the GPIO pins
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(TRANSISTOR, GPIO.OUT)
        GPIO.setup(BTN_SPEED_UP, GPIO.IN)
        GPIO.setup(BTN_SLOW_DOWN, GPIO.IN)

        # This will call a function when the speed up or slow down
        # buttons are pressed
        GPIO.add_event_detect(BTN_SPEED_UP, GPIO.RISING)
        GPIO.add_event_callback(BTN_SPEED_UP, speed_up)
        GPIO.add_event_detect(BTN_SLOW_DOWN, GPIO.RISING)
        GPIO.add_event_callback(BTN_SLOW_DOWN, slow_down)

def speed_up(channel):
        global delay
        # Take away the delay change value from the delay time.
        # Make sure the delay doesn't go less than the minimum
        # safe rate for use of stroboscopic lighting.
        delay = delay - DELAY_CHANGE
        if delay < DELAY_MIN:
                delay = DELAY_MIN

def slow_down(channel):
        global delay
        # Add the delay change value to the current delay
        delay = delay + DELAY_CHANGE

def loop():
        # The try statement makes sure we clean up properly
        # on a keyboard interrupt (Ctrl+C)
        try:
                # loop until the user presses Ctrl+C
                while True:
                        # Turn the strobe on, then wait for the
                        delay time
                        GPIO.output(TRANSISTOR, False)
                        time.sleep(delay)
                        # Turn the strobe off, then wait for the
                        delay time
                        GPIO.output(TRANSISTOR, True)
                        time.sleep(delay)
        except KeyboardInterrupt:
                pass
        finally:
                GPIO.cleanup()

# Now we setup the hardware, and start the main loop of the program

setup()

loop()
```

RICHARD SMEDLEY

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



```
pi@raspberrypi ~/tempfolder $ tail -n 7 fourinarow.py
            if board[x][y] == tile and board[x+1][y+1] == til
[x+2][y+2] == tile and board[x+3][y+3] == tile:
                return True
    return False


if __name__ == '__main__':
    main()
pi@raspberrypi ~/tempfolder $ ls
4row_arrow.png  4row_computerwinner.png  4row_tie.png   myfolde
4row_black.png  4row_humanwinner.png     fourinarow.py
4row_board.png  4row_red.png             myfolder
pi@raspberrypi ~/tempfolder $ mkdir myfolder3
pi@raspberrypi ~/tempfolder $ 
```

The command line offers tools to get text from different parts of a file, like skipping to the conclusion

Create and name files and directories with keystrokes, rather than mouse-clicks *and* keystrokes

COMMAND LINE Pi PART 2: READ/WRITE TEXT

Richard Smedley presents a cut-out-and-keep guide to using the command line on the Raspberry Pi. In part two, we get working on files...

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.

Now we can navigate folders and list files, it's time to learn how to create, view, and alter both files and folders. Once more, it's not that the task is difficult, rather that the forms of the commands (particularly editing) are unfamiliar, coming from an era before Common User Access (CUA) standards were created to ease switching between applications and operating systems.

Stick with it – with just the first two parts under your belt, you'll be able to do plenty of work at the command line, and start getting comfortable there.

Creating a directory

We're going to dive straight into working with files and folders by creating a new directory. Assuming you already have a terminal open (see 'Instant applications' box on the next page), and you're in your home directory (**pwd** to check, **cd ~** to get back there if necessary), type **mkdir tempfolder** and have a look with **ls**.

mkdir, as you've probably guessed, creates a new directory or folder. Let's use it to experiment with altering one of the included Python games – don't worry, we're not going to be programming Python, just making a small change by way of illustration. **cd tempfolder** (use tab completion, **cd t** then hit the TAB key). We'll copy over the files from the games directory:

```
cp ../python_games/fourinarow.py .
cp ../python_games/4row_* .
```

Wildcard

You may remember that the **..** refers to the directory above. The **.** (dot) at the end of the commands refers to 'just here' – which is where we want the files copied. Lastly, **4row_*** is read by the Pi as 'every file beginning **4row_**' – the ***** is known as a wildcard, and this one represents any number of characters (including none); there are other wildcards, including **?**, which means any single character.

Before we make any changes, try **python fourinarow.py** (after running **startx**) and you'll see you can run the local copy of the game (the Python Games part of the menu still reaches the original copy, of course). To change the game, we need an editor: there is a long and honourable tradition in the Unix and Linux world of having strong opinions on the merits or otherwise of various text editors – after all, they are an important tool for anyone who works daily with command scripts – but we'll sidestep the debate and use the Pi's built-in editor: nano. Unless you've previously used the pico editor, which accompanied the pine email

MORE INFO

Many utilities have info pages, giving far more information than their man page. If you're feeling brave, try **info nano** for a comprehensive guide to nano.

client on many university terminals in the 1980s and 1990s, it will seem a little odd (**Fig 1**). That's because its conventions predate the **CTRL+C** for copy type standards found in most modern programs. Bear with us.

Editing and paging

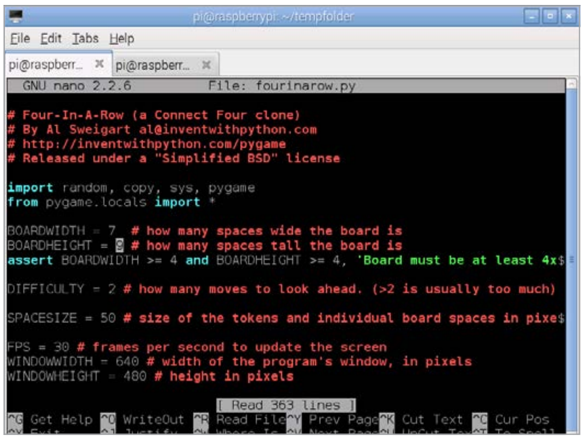
`nano fourinarow.py` will open the game for editing; use the arrow keys to go down nine lines, and along to the **BOARDHEIGHT** value of **6**. Change it to **10** (both the **BACKSPACE** and **DELETE** keys will work in nano). The last two lines of the screen show some shortcuts, with **^** (the caret symbol) representing the **CTRL** key: **CTRL+O**, followed by **RETURN** will 'write out' (save) the file; then use **CTRL+X** to exit. Now, **`python fourinarow.py`** will open an oversize board, giving you more time to beat the computer, should you need it – except there's no room to drag the token over the top of the board: go back and change the **BOARDHEIGHT** value to **9**, with nano.

If you want to take a look through the **`fourinarow.py`** listing without entering the strange environment of nano, you can see the entire text of any file using **`cat`**: eg **`cat fourinarow.py`**. Unfortunately, a big file quickly scrolls off the screen; to look through a page at a time, you need a 'pager' program. **`less fourinarow.py`** will let you scroll up and down through the text with the **PAGE UP** and **PAGE DOWN** keys. Other keys will do the same job, but we'll leave you to discover these yourself. To exit **`less`**, hit **Q** (which also works from man and info pages, which use a pager to display text).

Cats, heads & tails

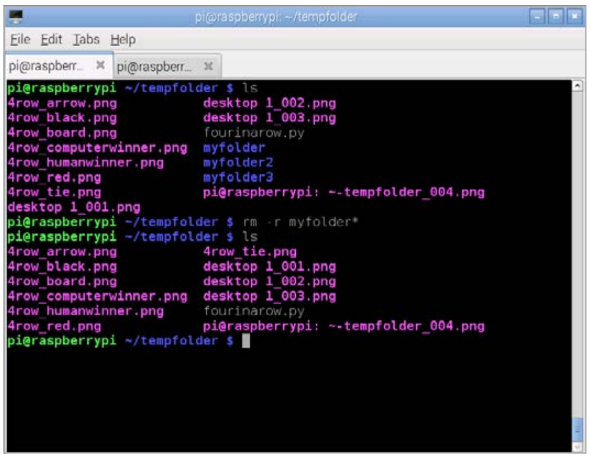
If editor wars are a Unix tradition we can safely ignore, there's no getting away from another tradition: bad puns. **`less`** is an improvement over **`more`**, a simple pager; the respective man pages will show you the differences. One advantage the more primitive **`more`** has is that at the end of a file it exits automatically, saving you reaching for the **Q** button. Not a huge advantage, admittedly, and you can always use **`cat`**.

Fortunately, **`cat`** is not a feline-based pun, but simply short for 'concatenate' – use it with more than one



```
  GNU nano 2.2.6          File: fourinarow.py

# Four-In-A-Row (a Connect Four clone)
# By Al Sweigart al@inventwithpython.com
# http://inventwithpython.com/pygame
# Released under a "Simplified BSD" license

import random, copy, sys, pygame
from pygame.locals import *

BOARDWIDTH = 7  # how many spaces wide the board is
BOARDHEIGHT = 9 # how many spaces tall the board is
assert BOARDWIDTH >= 4 and BOARDHEIGHT >= 4, 'Board must be at least 4x$

DIFFICULTY = 2 # how many moves to look ahead. (>2 is usually too much)

SPACESIZE = 50 # size of the tokens and individual board spaces in pixe$

FPS = 30 # frames per second to update the screen
WINDOWWIDTH = 640 # width of the program's window, in pixels
WINDOWHEIGHT = 480 # height in pixels

                              [ Read 363 lines ]
^G Get Help  ^O WriteOut  ^R Read File^Y Prev Page^K Cut Text  ^C Cur Pos
```

Fig 1 The default editor, nano, has unusual command shortcuts, but they're worth learning, as you'll find nano installed on virtually all Linux boxes, such as your web host



```
pi@raspberrypi ~/tempfolder $ ls
4row_arrow.png           desktop 1_002.png
4row_black.png           desktop 1_003.png
4row_board.png           fourinarow.py
4row_computerwinner.png  myfolder
4row_humanwinner.png     myfolder2
4row_red.png             myfolder3
4row_tie.png             pi@raspberrypi: ~-tempfolder_004.png
desktop 1_001.png
pi@raspberrypi ~/tempfolder $ rm -r myfolder*
pi@raspberrypi ~/tempfolder $ ls
4row_arrow.png           4row_tie.png
4row_black.png           desktop 1_001.png
4row_board.png           desktop 1_002.png
4row_computerwinner.png  desktop 1_003.png
4row_humanwinner.png     fourinarow.py
4row_red.png             pi@raspberrypi: ~-tempfolder_004.png
pi@raspberrypi ~/tempfolder $ 
```

Fig 2 rm is a powerful removal tool – use with great care!

file and it concatenates them together. Used with no argument – type **`cat`** – it echoes back what you type after each **ENTER**. Hit **CTRL+C** to get out of this when you've finished typing in silly words to try it. And remember that **CTRL+C** shortcut – it closes most command-line programs, in the same way that **ALT+F4** closes most windowed programs.

You can peek at the first or last few lines of a text file with **`head`** and **`tail`** commands. **`head fourinarow.py`** will show you the first ten lines of the file. **`head -n 5 fourinarow.py`** shows just five lines, as does **`tail -n 5 fourinarow.py`** with the last five lines. On the Pi, **`head -5 fourinarow.py`** will also work.

Remove with care

`nano afile.txt` will create a new file if afile.txt does not already exist – try it, and see if it works when you exit the file before writing and saving anything. We've done a lot already (at least, nano makes it feel like a lot), but it's never too early to learn how to clean up after ourselves. We'll remove the files we've created with **`rm`**. The remove tool should always be used with care: it has some built-in safeguards, but even these are easy to override (**Fig 2**). In particular, *never* let anyone persuade you to type **`rm -rf /`** – this will delete the entire contents of your Pi, all the programs, everything, with little to no chance of recovery.

Have a look at what files we have – if you're still in the tempfolder/ you made, then **ls** will show you the Four-in-a-Row files you copied here. Remove the program, then the .png files with careful use of the ***** wildcard.

```
rm fourinarow.py
rm 4row_*.png
```

`cd ..` to get back to /home/pi and **`rm -r tempfolder`** will remove the now empty folder. The **`-r`** (recursive) option is necessary for directories, and will also remove the contents if any remain.

Next month, we'll delve into file permissions and updating your Pi's software from the command line.

SWITCHING HELP

You don't need to wade through the man page to see what switches are available – type **`--help`** after the command to be shown your options. Eg **`rm --help`**

INSTANT APPLICATIONS

Although you can open the terminal emulator from the menu – Accessories > Terminal – for this, and any other app, just hit **ALT+F2** and type its command name: **`lxterminal`**.

SAM AARON

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

SONIC PI π))) TIPS & TRICKS

The creator of Sonic Pi, **Sam Aaron**, shares some of his top tips for budding electronic musicians of all ages...

IMPORTANT

· Sonic Pi 2.4 is required.

To update to the latest version, type these two commands into the terminal:

```
· sudo apt-get update
· sudo apt-get install sonic-pi
```

THERE ARE NO MISTAKES

This is the most important lesson. The best way to learn is to just try. Try lots of different things; stop worrying whether your code sounds good and start experimenting with as many different synths, notes, FX, and parameters as possible. You'll discover a lot of things that make you laugh, because they sound awful, and some real gems that sound truly amazing. Just drop the things you don't like and keep the things you do. The more 'mistakes' you allow yourself to make, the quicker you'll learn and discover your own sound.

USE THE FX

Once you've mastered the basics of making sounds with sample and play, you might be wondering what's next. Did you know Sonic Pi supports over 27 studio FX to change the sound of your code? FX are like fancy image filters in drawing programs, except that instead of blurring or making something black and white, you can add things like reverb, distortion, and echo to your sound. Think of it like plugging the cable from your guitar into an effects pedal of your choice and then into the amplifier, but Sonic Pi makes it much easier! All you need to do is to choose which section of your code you'd like the FX added to and wrap it with the FX code.

```
sample :loop_garzul

16.times do
  sample :bd_haus
  sleep 0.5
end
```

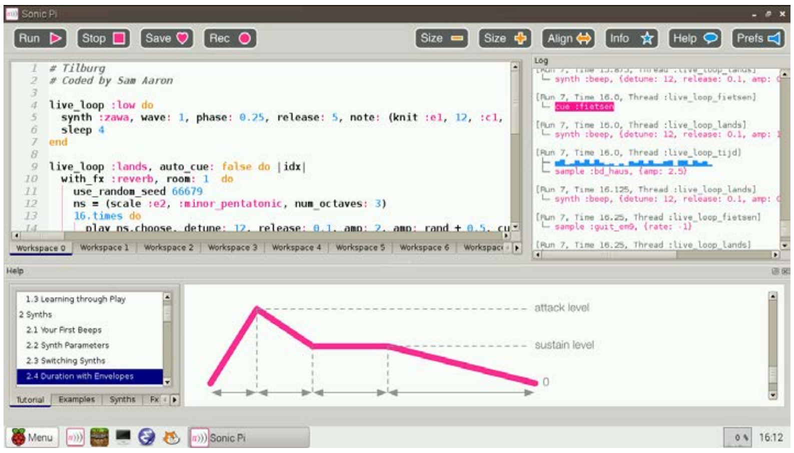
If you wanted to add FX to the **:loop_garzul** sample, you'd just tuck it inside a **with_fx** block, like this:

```
with_fx :flanger do
  sample :loop_garzul
end

16.times do
  sample :bd_haus
  sleep 0.5
end
```

Now, if you wanted to add FX to the bass drum, go and wrap that with **with_fx**, too:

```
with_fx :flanger do
  sample :loop_garzul
end

with_fx :echo do
  16.times do
    sample :bd_haus
    sleep 0.5
  end
end
```

Remember, you can wrap any code within **with_fx** and any sounds created will pass through that FX.



```
# Tilburg
# Coded by Sam Aaron

live_loop :low do
  synth :zawa, wave: 1, phase: 0.25, release: 5, note: (knit :e1, 12, :c1,
  sleep 4
end

live_loop :lands, auto_cue: false do |idx|
  with_fx :reverb, room: 1  do
    use_random_seed 66679
    ns = (scale :e2, :minor_pentatonic, num_octaves: 3)
    16.times do
      play ns.choose, detune: 12, release: 0.1, amp: 2, amp: rand + 0.5, cu
```

```
 └ synth :beep, {detune: 12, release: 0.1, amp:

[Run 7, Time 16.0, Thread :live_loop_fietsen]
 └ cue :fietsen

[Run 7, Time 16.0, Thread :live_loop_lands]
 └ synth :beep, {detune: 12, release: 0.1, amp: 1

[Run 7, Time 16.0, Thread :live_loop_tijd]
 ├
 └ sample :bd_haus, {amp: 2.5}

[Run 7, Time 16.125, Thread :live_loop_lands]
 └ synth :beep, {detune: 12, release: 0.1, amp:

[Run 7, Time 16.25, Thread :live_loop_fietsen]
 └ sample :guit_em9, {rate: -1}

[Run 7, Time 16.25, Thread :live_loop_lands]
```

PARAMETERISE YOUR SYNTHS

In order to really discover your coding sound, you'll soon want to know how to modify and control synths and FX. For example, you might want to change the duration of a note, add more reverb, or change the time between echoes. Sonic Pi gives you enough control to do exactly this, with special things called optional parameters. Copy this code into a workspace and hit Run:

```
sample :guit_em9
```

It's a great guitar sound. Let's change its rate:

```
sample :guit_em9, rate: 0.5
```

What's that **`rate: 0.5`** bit we just added at the end? That's called a parameter. All of Sonic Pi's synths support them and there's loads to play around with. They're also available for FX:

```
with_fx :flanger, feedback: 0.6 do
  sample :guit_em9, rate: 0.5
end
```

Now, try increasing that feedback to 1 to hear some crazy sounds! Read the docs for full details on all the many parameters available to you.

LIVE CODE

The best way to quickly experiment and explore Sonic Pi is to live-code. This allows you to start off some code and continually change and tweak it while it's still playing. So, if you don't know what the **`cutoff`** parameter does to a sample, just play around with it. Copy this code into one of your Sonic Pi workspaces:

```
live_loop :experiment do
  sample  :loop_amen, cutoff: 70
  sleep 1.75
end
```

Now, hit Run and you'll hear a slightly muffled drum break. Now, change the **`cutoff:`** value to 80 and hit Run again. Can you hear the difference? Once you get the hang of using **`live_loop`**, you'll never go back. If you were ever to do a live coding gig in the future, you'd find yourself relying on **`live_loops`** as much as a drummer relies on their sticks. To learn more about live coding, check out Section 9 of Sonic Pi's built-in tutorial.

SURF THE RANDOM STREAMS

One thing that's really fun to try is cheat by getting Sonic Pi to compose things for you. A really great way to do this is using randomisation. It might sound complicated, but it really isn't. Try this:

```
live_loop :rand_surfer do
  use_synth :dsaw
  notes = (scale :e2, :minor_pentatonic, num_octaves: 2)
  16.times do
    play notes.choose, release: 0.1, cutoff: rrand(70, 120)
    sleep 0.125
  end
end
```

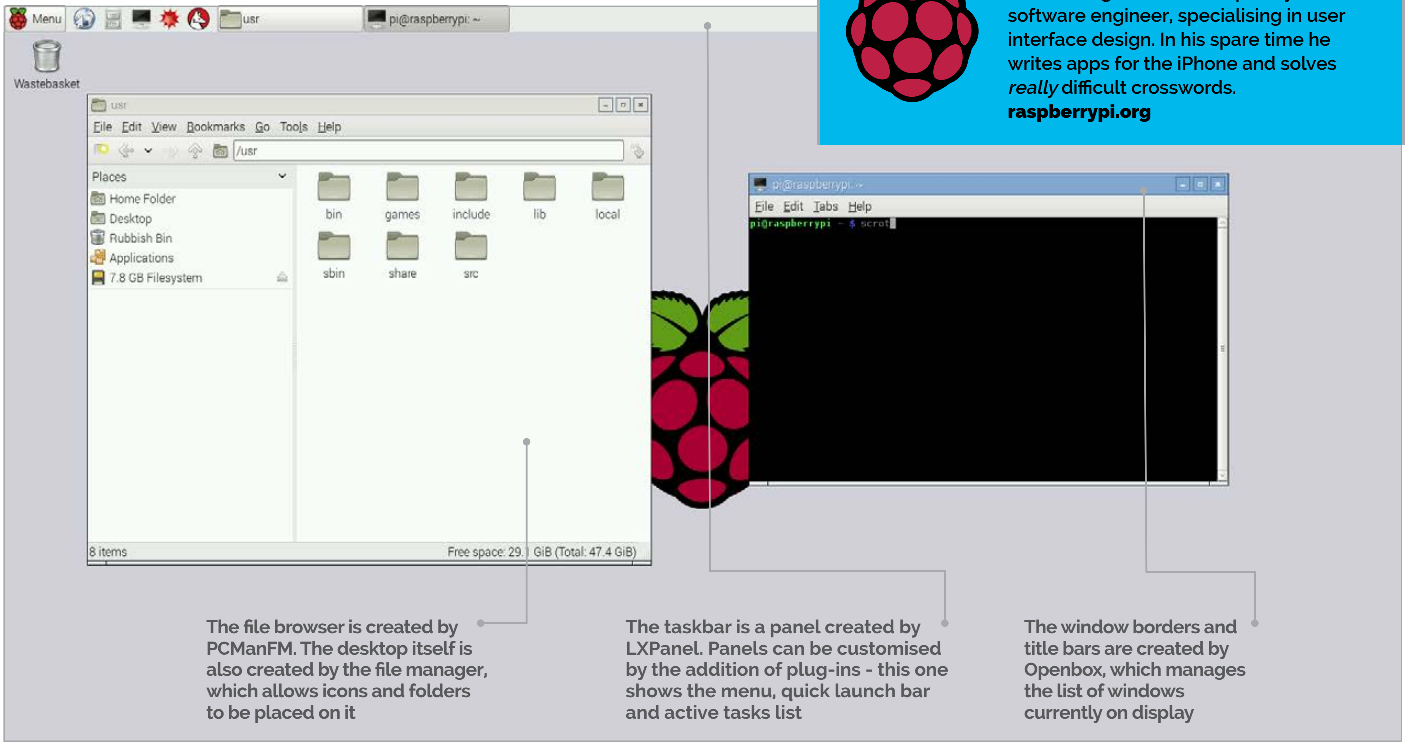
When you play this, you will hear a constant stream of random notes from the scale **`:e2 :minor_pentatonic`** played with the **`:dsaw`** synth. It might not sound like a melody, but that's the first part of the trick: every time we go round the

> "You can explore as many melodic combinations as you can imagine"

`live_loop`, we can tell Sonic Pi to reset the random stream to a known point. It's like going back to a particular point in time and space with the TARDIS. Let's try it. Add the line **`use_random_seed 1`** to the **`live_loop`**:

```
live_loop :rand_surfer do
  use_random_seed 1
  use_synth :dsaw
  notes = (scale :e2, :minor_pentatonic, num_octaves: 2)
  16.times do
    play notes.choose, release: 0.1, cutoff: rrand(70, 120)
    sleep 0.125
  end
end
```

Now, every time the **`live_loop`** loops around, the random stream is reset. This means it chooses the same 16 notes every time, giving you an instant melody. Here's the really exciting bit: change the **`seed`** value from 1 to another number – 4923, say – and it will give you another melody. So, just by changing one number (the random seed), you can explore as many melodic combinations as you can imagine, and that's the magic of code.



```
pi@raspberrypi ~ $ scrot
```

The file browser is created by PCManFM. The desktop itself is also created by the file manager, which allows icons and folders to be placed on it

The taskbar is a panel created by LXPanel. Panels can be customised by the addition of plug-ins - this one shows the menu, quick launch bar and active tasks list

The window borders and title bars are created by Openbox, which manages the list of windows currently on display

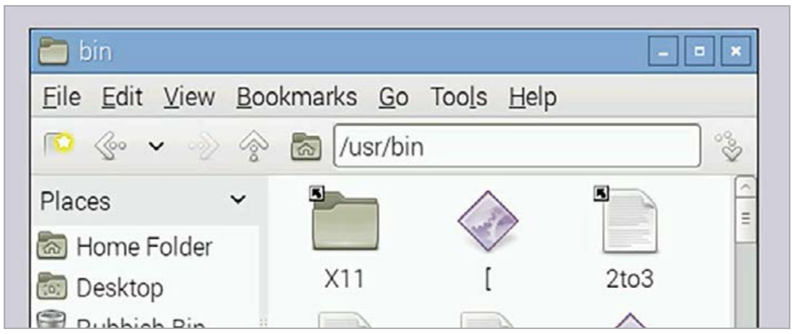
SIMON LONG

Simon Long works for Raspberry Pi as a software engineer, specialising in user interface design. In his spare time he writes apps for the iPhone and solves *really* difficult crosswords.
raspberrypi.org

HACKING THE RASPBIAN DESKTOP PART 1: WHAT IS LXDE?

In the first part of his new series, **Simon Long** talks us through Raspbian's desktop environment, LXDE...

Below Openbox manages the appearance of window title bars, including the buttons to minimise, maximise, and close a window



If you've used the desktop environment on Raspbian, rather than the command line, you've seen LXDE. The Lightweight X Desktop Environment is the software which creates the desktop graphic user interface (GUI) that appears when you type **`startx`** from the Raspbian command line – the desktop, the windows, the taskbar; all are parts of LXDE.

LXDE is a user interface which sits on top of a system called X, a client-server windowing system. When an application starts, it requests a window in which to work from the X server program; the server also takes care of things like detecting mouse and keyboard input, and putting windows on the screen.

But X itself provides only the barest elements of a GUI – there are numerous environments that can sit on top of it and make the result look nicer, and LXDE is one of these. It's a very good fit for the Raspberry Pi because, as the name suggests, it is lightweight in terms of processor and memory usage, and so works well on a lower-powered device like the Raspberry Pi.

LXDE itself consists of several different pieces of software, all with specific tasks. The more important of these are described in the 'LXDE key components' box on the next page.

LXDE'S KEY COMPONENTS

Openbox - The window manager

When you launch an application or open a window, the window manager is responsible for providing a window of the appropriate size, in which the application can appear. While the application itself determines what appears inside the window, the window manager is responsible for the overall appearance of the window – the border, the title bar at the top, the buttons to close, minimize, or resize the window – and this is why all windows, from whatever application, look the same in these respects. The window manager is also responsible for the layout of multiple windows on the screen, managing how windows appear on top of others when they overlap, and maintaining which window currently has focus, ie responds to the mouse and keyboard.

LXPanel - The desktop panel controller

This is responsible for creating the taskbar and for managing its contents. In fact, it can create multiple taskbars and put them anywhere on the screen, but that might look a bit confusing! Each taskbar is customisable by the addition of various plug-ins. Some of these – such as the task switcher, the system menu, and the quick launch icons – are standard and are built in to LXPanel, but you can also write and add others yourself.

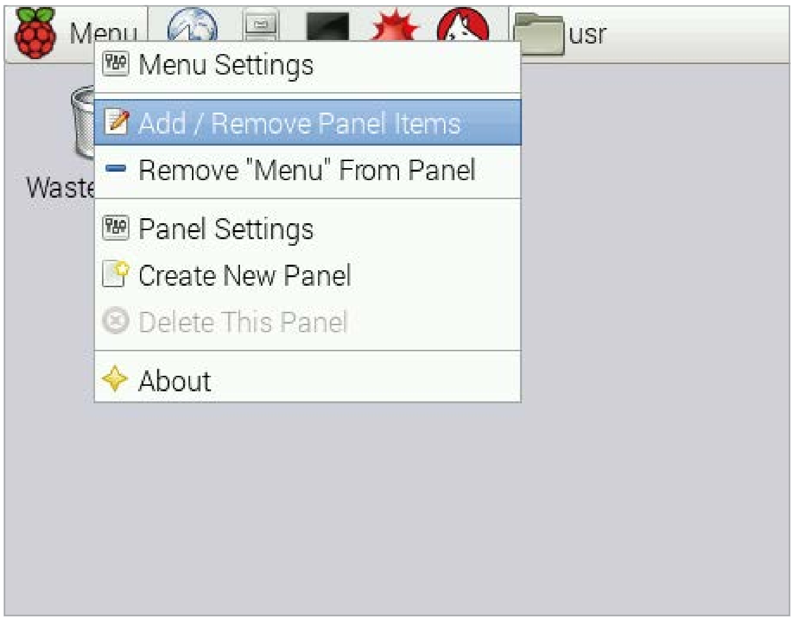
PCManFM - The file manager

This is responsible for providing the directory windows that allow you to browse the file system on the Pi, as well as things like the ability to drag and drop files from one directory to another. It takes care of associating files with applications, so that when you double-click a file with a certain extension, it opens in the right application. The desktop itself is effectively a file manager window, so the file manager also controls things like desktop icons and the wastebasket.

LXSession - The session manager

When you start LXDE, the session manager is responsible for controlling which applications are launched, including those listed above. Also, the whole of LXDE is written using a graphics toolkit called GTK, and the appearance of every element in the toolkit – buttons, menus, and the like – can be set for the entire system, from what is called a theme. LXSession controls which theme is applied to LXDE, and thus can be used to customise the appearance of every application that runs under it.

Below LXPanel provides a pop-up menu to modify panel settings – more on this in the next instalment



Look under the bonnet

Naturally, like the rest of Raspbian, LXDE is open source and under active development. It is written in C, and the code for all the components can be freely downloaded, modified, and rebuilt, even on the Raspberry Pi itself. If you are a keen hacker, it is worth downloading the code and having a look at it, as much of it is pretty straightforward – have a look at **lxde.org** for all the details, downloads and wiki articles, among other things.

However, before adopting drastic measures, it is worth knowing that a lot of the overall appearance

"Like the rest of Raspbian, LXDE is open source and under active development"

and behaviour of LXDE is controlled by a set of preference files that sit behind the scenes, and by editing these – they are plain text – you can make quite significant changes to how LXDE looks and works without changing a single line of code. There are dedicated applications to control some of these settings provided as part of LXDE – **obconf** is a controller for the Openbox preferences, and **lxappearance** adjusts the themes used by LXSession – but these don't necessarily control everything you might want to change, and it isn't always obvious where the setting you want to adjust can be found.

So that's what the rest of this series of articles is going to cover: how you can modify the LXDE settings to customise your Raspberry Pi to look the way you want it to.

NEXT MONTH

In the next part of this series, we will look at how to customise LXPanel to change the appearance of the taskbar, and how to add and remove plug-ins.

WILLEM KOOPMAN

Peripatetic sysadmin, providing grumpy solutions to grumpy problems; also maker of **whatcaniseefromtheshard.com**. Stumbled upon the field of computer vision while working in the magical world of visual effects.
secretbatcave.co.uk

FACE DETECTION WITH OPENCV

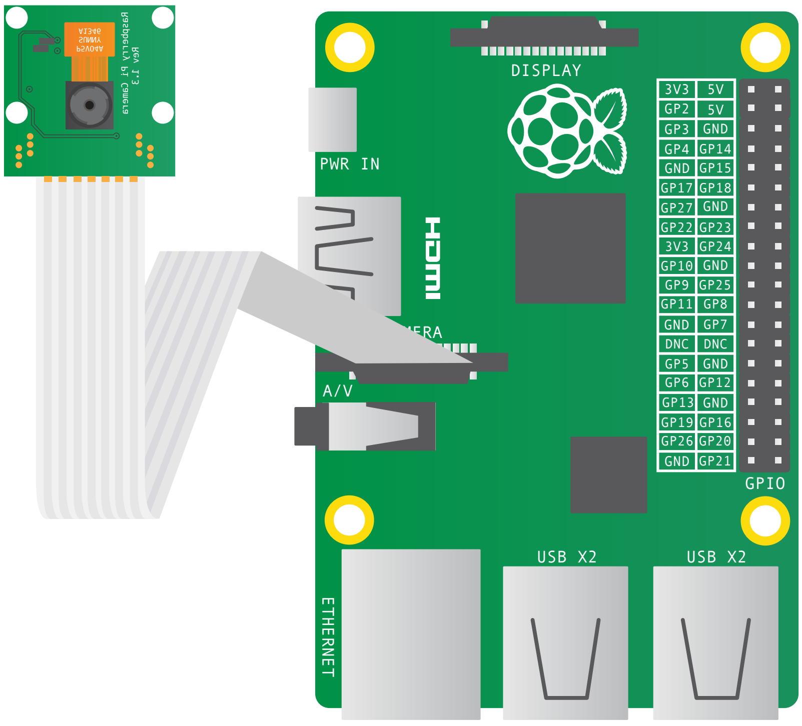
You've got a Raspberry Pi Camera module? You've taken a few images? Let's do something really clever and use them to detect faces...

You'll Need

- An internet connection
- Camera module (or webcam)
- OpenCV

You've set up your motion-triggered webcam, but that pesky dog keeps triggering it. How do you figure out if that alert is someone poking around or Fido searching for socks again? In this tutorial, we'll show you how to get your Raspberry Pi to separate the dogs from the faces, using the computer vision library, OpenCV.

We're going to make a simple Python script that will work its way through a directory of pictures, copying the ones that have faces in them. Not only that, it'll also draw a box around each face.



>STEP-01

Install OpenCV

By default, OpenCV isn't shipped with Raspbian. Never fear – everything is a simple **`apt-get`** away. First, we need to install OpenCV. In a terminal, type: **`sudo apt-get update`**, press ENTER, then: **`sudo-apt-get install python-opencv libopencv-dev`** and follow the instructions. We'll see if it's installed correctly by running the Python interactive interpreter and loading the OpenCV module. Type: **`python`** (and ENTER), then **`import cv`** (and ENTER again). If everything is installed correctly, you should see an empty prompt. If you see something along the lines of **`ImportError`**, go back and see if the **`apt-get`** worked correctly.

>STEP-02

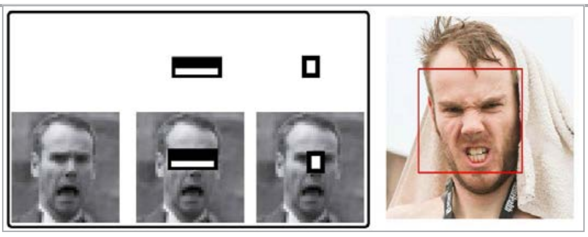
Understanding Haar-like features

We're going to use an algorithm called a 'Haar cascade'. Because computers have no understanding of what a face looks like, we have to give it a rule book. In this case, a Haar cascade describes the 'brightness signature'. A face contains two eyes surrounded by skin. The area surrounding the eye is a different intensity to the eye itself. A Haar cascade describes these patterns to provide us with a way to detect faces (and other objects).

>STEP-03

Begin the code!

Enough chit-chat! Let's write some code. We need to import the various libraries we are going to use and set some sensible defaults for the Haar detector. These defaults provide a trade-off between speed and accuracy. First, we set **`minSize`** to limit the smallest detectable face to a 20-pixel square. **`imageScale`** scales the image before we feed it into the detector; smaller


Above How the face detector works under the hood: high-contrast boxes are mapped to parts of the face

images mean faster detect times, but less accuracy. **minNeighbors** tells the detector that a match must be made up of a minimum number. Finally, **haarFlags** are special flags telling the detector what bits to ignore.

>STEP-04

Prepare the image

The first function we're going to create is **detectFace()**. Because the Haar detector only works with greyscale images, we make a greyscale copy. **grey** is resized and copied into the **small_img** container. Lastly, we equalise the histogram (using **EqualizeHist**). This evens out the contrast, making the Haar detector more effective. We pass the variables, **small_img** and **cascade**, along with the rest of the defaults we defined at the very start, into the function **cv.HaarDetectObjects**. It then spits out a list of objects, with attached coordinates, and dumps it into the variable **faces**.

>STEP-05

Mark them Faces

We then 'iterate' through all the objects and extract the 'bounding box' (the area where the object detector thinks there is a face.) Now, this is where we do something vaguely confusing. Remember in the previous step, we made a few copies of the original image? Well, we didn't throw away the original. We can mark where we think the faces are on the original, so that we can save full-sized images in full colour!

We've scaled up the coordinates so that we can accurately tell **cv.Rectangle** where the top-left and bottom-right corners of the box should be.

>STEP-06

Final Touches

readDirectory() goes through the directory supplied as a command-line argument and extracts files ending with '.jpg'. It then opens the image and passes it to **detectFace()**. If it finds some faces, it'll save out the marked images into a new file using **cv.SaveImage()**. To use your new program, you first need to find a Haar cascade XML file and put the path into **cv.Load()**. They can be found in /usr/share/opencv/haarcascades/. Running the program is as simple as storing some jpg files in the folder and typing **python facedetect.py .** With any luck, you'll see something like the following:

```
samples/ has:
  Analysing 292942_10151131251926133.jpg:
        Detected  2  object(s)
        Time = 1268.761ms
```

Facedetect.py

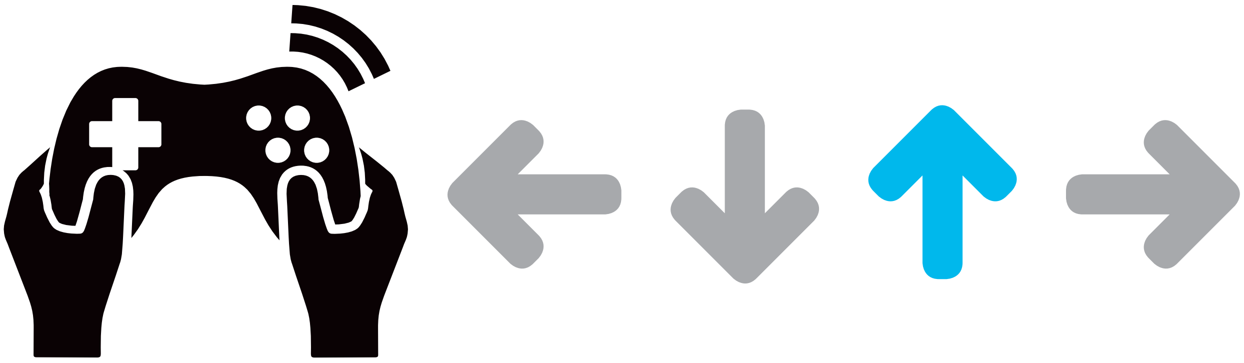
Code Language
>PYTHON

```
import os, sys, time
import cv2.cv as cv

minSize = (20, 20)
imageScale = 1
haarScale = 2
minNeighbors = 3
haarFlags = cv.CV_HAAR_DO_CANNY_PRUNING

def detectFace(img, cascade):
  # allocate temporary images
  gray = cv.CreateImage((img.width,img.height), 8, 1)
  small_img = cv.CreateImage((cv.Round(img.width /
imageScale),cv.Round (img.height / imageScale)), 8, 1)
  # convert color input image to grayscale
  cv.CvtColor(img, gray, cv.CV_BGR2GRAY)
  # scale input image for faster processing
  cv.Resize(gray, small_img, cv.CV_INTER_LINEAR)
  cv.EqualizeHist(small_img, small_img)
  faces = cv.HaarDetectObjects(small_img, cascade,
cv.CreateMemStorage(0),haarScale, minNeighbors, haarFlags, minSize)

  if faces:
    print "\tDetected ", len(faces), " object(s)"
    for ((x, y, w, h), n) in faces:
      #the input to cv.HaarDetectObjects was resized, scale the
      #bounding box of each face and convert it to two CvPoints
      pt1 = (int(x * imageScale), int(y * imageScale))
      pt2 = (int((x + w) * imageScale), int((y + h) *
imageScale))
      cv.Rectangle(img, pt1, pt2, cv.RGB(255, 0, 0), 3, 8, 0)
    return img

  else:
    return False

# scan all directories and subdirectories for jpg images
def readDirectory(fileLocation, cascade):
  for root, dirs, files in os.walk(fileLocation):
    print root, "has:"

    for name in files:
      if name.find(".jpg") >=1 :
        #sequentially loop, load and detect.
        print "Analysing " + name +":"
        #measure how long it takes
        t = cv.GetTickCount()
        #load in the image
        image = cv.LoadImage(os.path.join(root,name), 1)
        match = detectFace(image, cascade)

        if match:
          #save a new image with a box round each face
          cv.SaveImage( fileLocation + "/face_" + name, match)
        t = cv.GetTickCount() -t
        print "\tTime = %gms" %(t/(cv.GetTickFrequency()*1000.0))

if __name__ == '__main__':
  cdir = "/usr/share/opencv/haarcascades/"
  cascade = cv.Load(cdir + "haarcascade_frontalface_default.xml")

  if len(sys.argv) != 2:
    print 'please provide a directory to read'
    sys.exit(1)
  readDirectory(sys.argv[1], cascade)
```

PART 2
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



ANIMATING SHAPES & PATHS

In the second part of this series, we'll learn how to move shapes around the screen in different directions, speeds, and patterns...

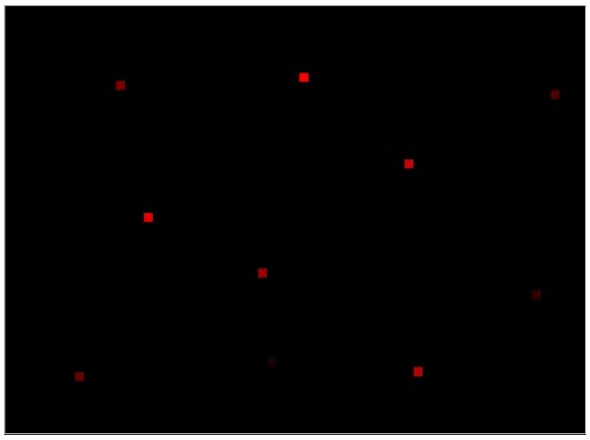
QUICK TIP

When we run our games, our window is given the title 'Pygame window'. We can set that to any string (series of characters) we like with `pygame.display.set_caption('Pygame Shapes!')`

In the first part of this set of tutorials, last issue, we looked at creating different kinds of shapes of different sizes and colours. Now we're going to be looking at different ways of moving and manipulating those shapes over time. Once we've covered the fundamentals of moving shapes with code, we can then jump into using keyboard and mouse events to control how and when things move (next issue). In this tutorial, we don't have one single Pygame program – instead, we have a couple of different code chunks, each demonstrating a different concept. Each complete program will consist of the **top** code, followed by one of the code chunks, and then finished with the **bottom** code. You can use the same file to test the code, or you can create a different file for each chunk; the result will be the same.

Things to notice

Before we jump into the animation, take a quick look at the import statements on lines 2 and 3 of the **top** code on the next spread). In the last tutorial, we imported all of Pygame and used some of its methods for drawing shapes. That was fine, because we were only drawing static things that didn't take user inputs, but from now on, we're going to include Pygame's locals and events constants. These are special variables that Pygame includes to help us write more readable code, as well as take away some of the complexity of interacting with the system that we're running our code on. The **`pygame.locals`** variable mostly contains properties that describe system and game state, so we've called it



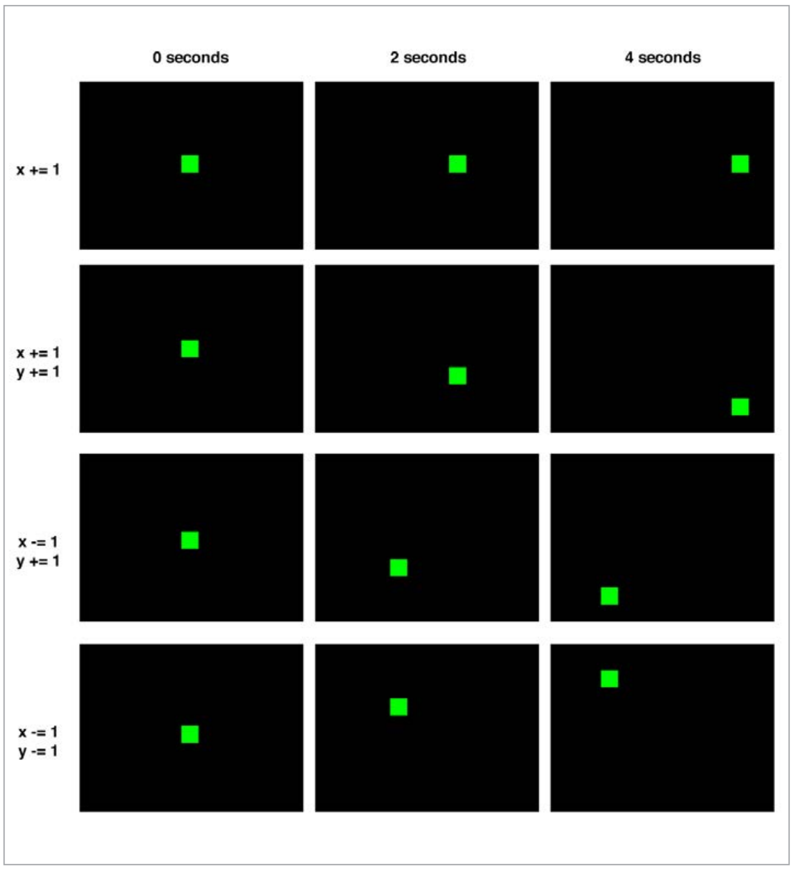
Above A simulated screenshot showing the random placement of red squares in our window

GAME_GLOBALS to reflect this. **pygame.events** includes a list of events, like keyboard events or system events that have happened since Pygame last updated its view; that's why we've imported it as **GAME_EVENTS**. In a future tutorial, we'll go over what this means exactly; for now, all we're going to use it for in the **bottom** code is to check whether or not our player tried to quit the game as it was running (in this case, by trying to close the window), and then close our program properly.

Moving shapes in time and space

When we think of animation, our minds might drift to thoughts of cartoons we watched as children (or still do) – just a bunch of lines and colours moving around a screen to trick our brains into seeing movement where there is none. It's no different with computers: whenever you move a mouse or minimise a window, nothing has actually been moved; instead, pixels have been drawn, updated, refreshed, and then drawn again, with everything in its new place.

If you run **chunk 01** (put the **top** code, **chunk 01** code and the **bottom** code together in one file) without uncommenting anything, you'll see a bunch of red squares appearing and disappearing all around the screen. Don't worry, nothing is broken! This is just to demonstrate Pygame drawing, destroying and redrawing things in a window. Add a **#** to the start of the line that starts **surface.fill()**. We use this code to clear the pixel data from the previous frame. Without it, what



Above This table demonstrates how different motions affect the position of a shape over time

> We can change the variable so that when it's next drawn, it will look slightly different

we see is all of the different frames built up one on top of the other as time passes. **surface.fill()** is like the paint that we use to cover old wallpaper before we add the new one: it creates a blank slate for us to work with.

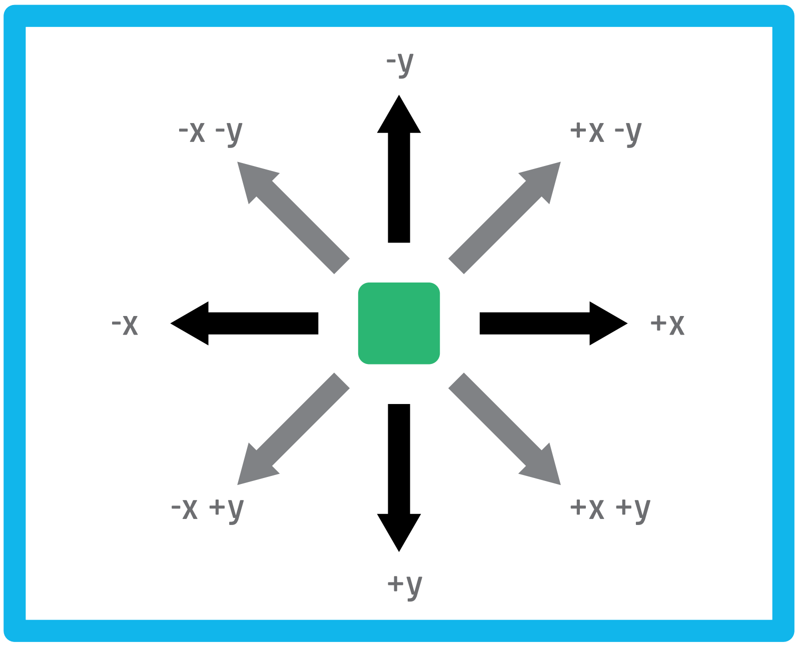
But that's not very useful, is it? Let's replace **chunk 01** code with **chunk 02** and you'll see a green square moving slowly to the right of the screen.

So, what's making our square move? When we were following the first tutorial, we were drawing shapes like this using numbers that we would pass through to Pygame, like **pygame.draw.rect(surface, (255,0,0), (20, 50, 40, 30))** – and that's all well and good, providing you never want to change anything about that shape. What if we wanted to change the height, width, or colour of this shape? How could we tell Pygame to change the numbers that we've already entered? Well, this is where variables come in. Rather than passing through numbers to **pygame.draw.rect()**, we pass in variables instead. After we've drawn the shapes, we can change the variable so that when it's next drawn, it will look slightly different. With **chunk 02**, every time we draw our green square, we add 1 to the variable we use to define its X coordinate (how far it is from the left of the screen), **greenSquareX**. We do this with **+=**, which basically says 'take the current value of the variable and then add whatever number comes after it'.

If we change that line to read **greenSquareX += 5**, every time we draw our square, it will be 5 pixels to the right of where it was the last time it was drawn. This gives the illusion of the shape moving faster than before. If we changed the number we add to **greenSquareX** to 0, our shape would never move; and if we changed it to -5, it would move backwards.

QUICK TIP

If we want to subtract values from a variable, we don't always have to use -= for subtraction and += for addition. We can use += for both; simply add a negative number to take away numbers... e.g. 4 + -3 = 1.



Above This diagram shows the eight directions a shape can move in when using integers

Moving in all directions

So that's how we move left and right; if we can do that much, surely we can go up and down too? Comment out the **greenSquareX** line from **chunk 02** and uncomment (remove the **#** from) the line below. Our square will start to travel towards the bottom of the screen. Just like before, we're changing the variable that tells our shape where to go, **greenSquareY** (Y, not X), just a little bit each time to make it move. And, just as we saw by changing the X variable, we can make the green square go up by adding a negative number.

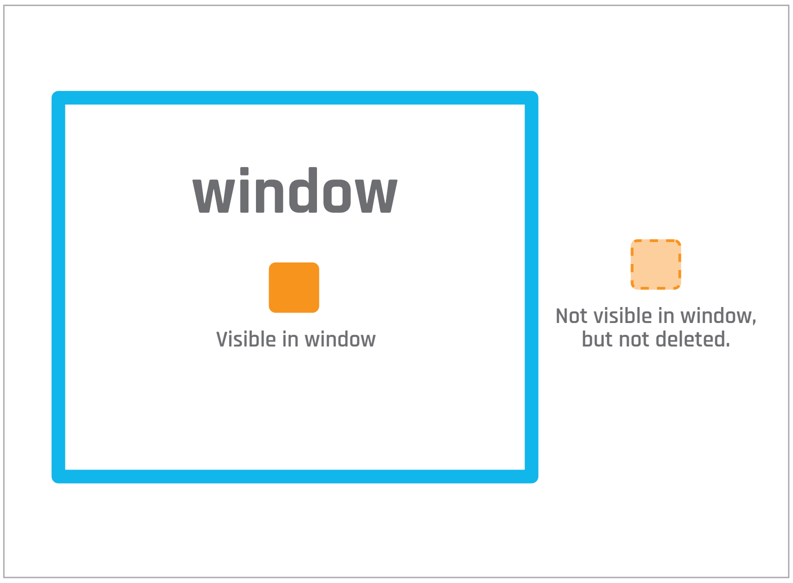
So now we can animate things moving in four directions, that's enough freedom to make so many classic games: *Pokémon*, *Legend Of Zelda*, *Space Invaders*, and more. These games would only move things horizontally and vertically, but never at the same time. What's that? How would we move things diagonally? Well, that's pretty simple, too.

If we uncomment both **greenSquareX** and **greenSquareY** in our code, then our shape will move to the right and down every time Pygame updates the screen. If we add to our X and Y values, our shapes will move to the right and down. If we add to our X value and subtract from our Y value, then our shapes will move to the right and up. If we subtract from our X value and add to our Y value, our shapes will move to the left and down. Finally, if we subtract from both our X and Y values, our shape will move to the left and upwards. That means we have eight directions that our objects can move in – assuming, that is, that we use numbers that are whole and equal to one another. If we used values that were different for our X and Y values, and we used floats (which are numbers with a decimal place, like 2.3 or 3.141) instead of integers (whole numbers), we could achieve a full 360 degrees of motion.

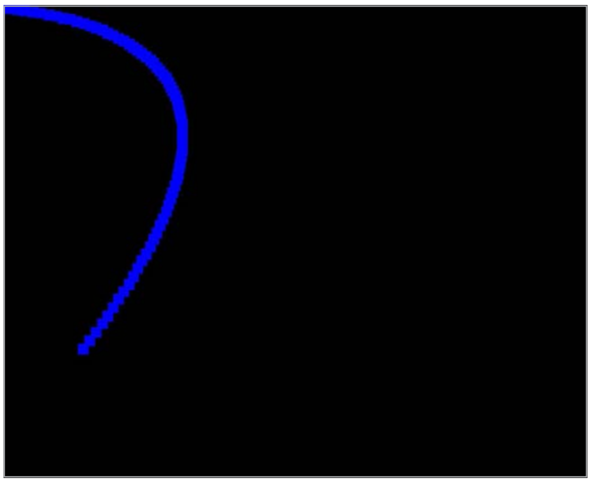
Let's play with numbers and decimals a little more. So far, the values we've used to animate our shapes around the screen have been integers that remain constant. With each frame, we would always add 1 (or some other arbitrary value) to move our object. But what happens if we change the values that we use to animate things? What if, instead of adding 1 to X/Y coordinates, we add 1, then 1.1, then 1.2, and so on?

Replace the code from **chunk 02** with the code from **chunk 03** (or create a new file with the **top** + **chunk 03** + **bottom** code). Now if we run that, what do we see? We're adding to both our X and Y values, so our square is moving down and to the right, but something is different from our previous bits of code: as our program continues to run, our square moves to the right a little more than it did in the previous frames. It's accelerating. This is because we're using variables that store a basic measure of speed. By using a variable to add a value to our X and Y coordinates, we can increase the amount of distance that is added in each frame, which gives the illusion of acceleration.

> "Moving in four directions – enough freedom to make so many classic games..."



Above The blue box is the viewport of a Pygame window


Above This is the path travelled by a shape moving across the window while accelerating

If we were to change our code so that it increased our speed variables (**blueSquareVX / blueSquareVY** in this case) through multiplication instead of addition or subtraction, our shapes would accelerate exponentially; we'd have hardly any time to see them before they ran off the screen.

Speaking of which, what happens to our shapes when they run off an edge and they're no longer on our screen? Are they deleted? Are they gone for ever? The answer is no. You can think of our window like an actual window in your house. If you look out of the window to see the postman and he then moves further down the street so you can no longer see him, he's not gone, he's just beyond your line of sight. If our shapes move further across our screen so that we can no longer see them, they don't stop moving or disappear, they keep on going for ever, or until you tell them to stop and come back.

Change the third line in **chunk 03** to read **blueSquareVX = 8**, change the penultimate line in **chunk 03** to **blueSquareVX -= 0.2**, and comment out the last line. Now when we run **chunk 03** for the last time, we see that our square moves to the right across our screen, before slowing to a stop and then coming back on itself, forming an arcing animation. This is because the **blueSquareVX** variable has entered minus numbers, but the **blueSquareVY** variable continues to increase. If we had subtracted the **VX** and **VY** variables in equal values, with equal starting speeds (both **VX** and **VY** being 8, for example), our shapes would have continued along their path, stopped, and then reversed along the exact same path, with the same rate of acceleration as it slowed. Play with theses values to see what effect they have on how our shape moves. If you like, you can comment out (add a **#** to) the **surface.fill** line and you'll see the path our shape takes trailing behind it.

TOP

```
import pygame, sys, random
import pygame.locals as GAME_GLOBALS
import pygame.event as GAME_EVENTS

pygame.init()
windowWidth = 640
windowHeight = 480
surface = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption('Pygame Shapes!')
```

CHUNK 01

```
while True:
    surface.fill((0,0,0))
    pygame.draw.rect(surface, (255,0,0), (random.randint    ↵
(0, windowWidth), random.randint(0, windowHeight), 10, 10 ) )
```

CHUNK 02

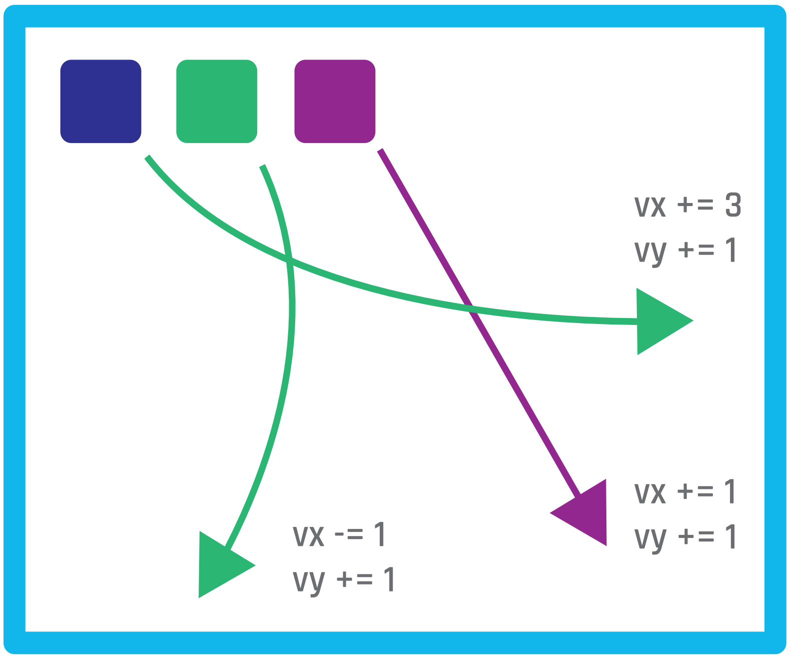
```
greenSquareX = windowWidth / 2
greenSquareY = windowHeight / 2

while True:
    surface.fill((0,0,0))
    pygame.draw.rect(surface, (0, 255, 0),
     (greenSquareX, greenSquareY, 10, 10))
    greenSquareX += 1
    #greenSquareY += 1
    pygame.draw.rect(surface, (0, 0, 255),
       (blueSquareX, blueSquareY, 10, 10))
```

CHUNK 03

```
blueSquareX = 0.0
blueSquareY = 0.0
blueSquareVX = 1
blueSquareVY = 1

while True:
    surface.fill((0,0,0))
    pygame.draw.rect(surface, (0, 0, 255),
       (blueSquareX, blueSquareY, 10, 10))
    blueSquareX += blueSquareVX
    blueSquareY += blueSquareVY
    blueSquareVX += 0.1
    blueSquareVY += 0.1
```

BOTTOM

```
    for event in GAME_EVENTS.get():
        if event.type == GAME_GLOBALS.QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```



Above This is demonstrating the varying effects of different acceleration values on shapes

Animating other properties

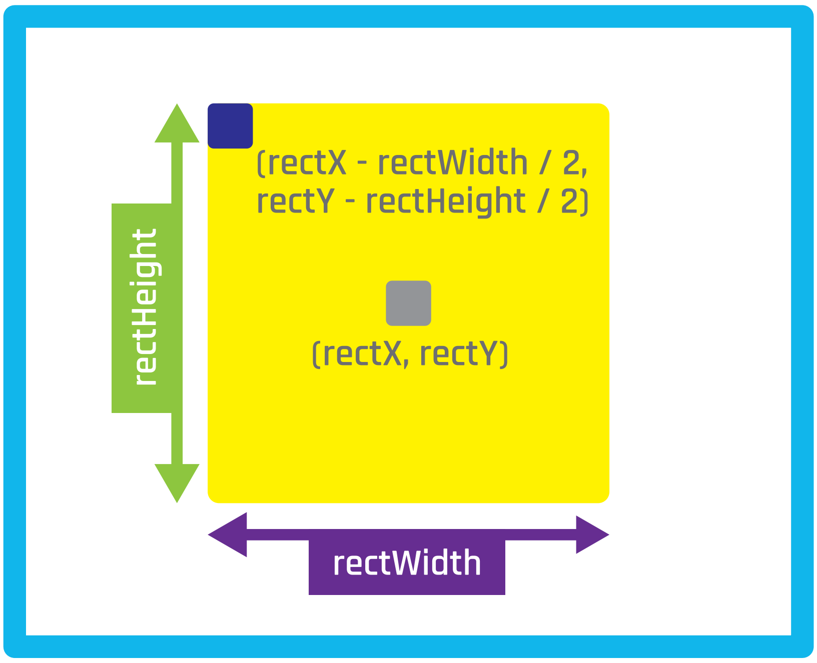
Animation isn't just about making things move: it's about making things change, too. Up until now, we've been animating shapes by moving them, but we can use the same approach of changing variables over time to affect other properties, like the dimensions of our shapes. For this, switch out the **chunk 03** code for **chunk 04**. Here, **`pygame.draw.rect`** draws a rectangle just the same as we've done before, but, as in other examples, we've replaced the parameters that determine the width and height of our rectangle with variables that we change.

We also do a little bit of maths in our code. As the square gets larger, the point from which it is drawn won't change, so the shape will get bigger, but it will do so off-centre from the rest of the window. By subtracting half of the width and half of the height from the coordinates that we draw our shape at, our square will remain in the centre of the window as it gets larger. The nice thing about using variables in our maths is that no matter how we change our variables, the shape created will always be in the centre of the window. Change the number on the **`rectWidth`** line to any other number between 2 and 10. Now, when our square enlarges, it becomes a rectangle, because its width increases faster than its height does, but it still remains in the centre.

The same effect works in the opposite direction. If we start off with a square that has a width and a height of 50, which we can do by setting the variables **`rectWidth`** and **`rectHeight`** to 50 and change the **`+=`** on those lines to **`-=`**, our square will decrease in size while remaining central to our window.

Something curious happens when our shape reaches a width and height of 0 – it starts to grow again. Why? When we hit 0, we start to draw our rectangle with negative numbers, which we are offsetting against with our maths. So, when we draw a shape with a negative width and then offset it against a negative number, our starting points become positive numbers again, albeit mirrored. We can't see the effect because we're using solid colours, but if we were to use the same expanding/shrinking code with an image, it would be flipped upside-down and back-to-front. That's just one of many little quirks that we'll explore in the future, but for now, we're going to finish up by changing the colours of our shapes over time, by moving onto our last code section, **chunk 05**.

Below Here are the different properties that allow us to centre a square as it enlarges/shrinks



> "This is a key concept of game development: how things respond"

Changing colour over time

Just like our previous bits of code, we're using variables in place of values to define what our shapes will look like with **`pygame.draw.rect`**. This code, however, has something a little different from the previous examples. Here, we're not just adding and subtracting values each and every time we draw our shapes – instead, we're checking the values that we have before we change them, using an **`if, else`** statement.

This is a key concept of game development – how things respond to you playing a game is a result of hundreds and thousands of these little checks going on every few milliseconds. Without them, there would be no kind of order to any game – it would be like our first bit of code, with the square simply appearing and disappearing at random positions, and there's not much fun in that! With these **`if, else`** checks, we're making sure that the red, green and blue values never go over 255 (the maximum value that these colours can display at, otherwise Pygame will throw an error).

If our values are about to go over 255, we assign them a random value between 0 and 255. The result? The colour of our square will change and will then continue to slowly work its way through the RGB colour palette by adding 1 to our R, G, and B variables (**`squaresRed`**, **`squaresGreen`** and **`squaresBlue`**) as our Pygame program runs. Just as before, if we added a larger number to each of our variables, we would cycle through the available colours more quickly; if we added less to each RGB value every time Pygame updates, we would cycle through all of the available colours more slowly. As well as a great learning device, it looks pretty groovy, too.

NEXT MONTH We'll be introducing control mechanisms so we can make our first mini-game!

TOP

```python
import pygame, sys, random
import pygame.locals as GAME_GLOBALS
import pygame.event as GAME_EVENTS
pygame.init()
windowWidth = 640
windowHeight = 480
surface = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption('Pygame Shapes!')
```

CHUNK 04

```python
rectX = windowWidth / 2
rectY = windowHeight / 2
rectWidth = 50
rectHeight = 50

while True:
    surface.fill((0,0,0))
    pygame.draw.rect(surface, (255,255,0), (rectX - rectWidth / 2, rectY - rectHeight / 2, rectWidth, rectHeight))
    rectWidth += 1
    rectHeight += 1
```

CHUNK 05

```python
squaresRed = random.randint(0, 255)
squaresBlue = random.randint(0, 255)
squaresGreen = random.randint(0, 255)

while True:
    surface.fill((0,0,0))
    pygame.draw.rect(surface, (squaresRed, squaresGreen, squaresBlue), (50, 50, windowWidth / 2, windowHeight / 2))
    if squaresRed >= 255:
        squaresRed = random.randint(0, 255)
    else:
        squaresRed += 1
    if squaresGreen >= 255:
        squaresGreen = random.randint(0, 255)
    else:
        squaresGreen += 1
    if squaresBlue >= 255:
        squaresBlue = random.randint(0, 255)
    else:
        squaresBlue += 1
```

BOTTOM

```python
    for event in GAME_EVENTS.get():
        if event.type == GAME_GLOBALS.QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```

pipsta.co.uk

£84 / $125

Maker Says

"The little printer with big ideas"
Pipsta

PIPSTA

Is this tiny, Raspberry Pi-friendly printer what your next IoT project is waiting for? **Russell Barnes** finds out…

It's not unusual for computers to be built into your computing experience. For example, Apple has done a roaring trade putting them behind monitors, and we probably wouldn't have the Raspberry Pi today if it weren't for classic home computers that were built behind keyboards, like the BBC Micro or the Amiga 500. But what about PCs built into printers? Is it taking things a little too far? Well, let's find out, because that's exactly what the Pipsta offers.

It's designed primarily as an Internet of Things gadget – a smart printer able to automagically print weather reports, Twitter feeds, or much-needed travel updates. It's not entirely unlike Adafruit's IoT Printer, though Pipsta is a little cheaper and easier to put together (it also features NFC capabilities). While Adafruit recommends its printer as a weekend project that requires soldering, Able Systems' Pipsta only takes about 45 minutes to put together. Since the printer workings are completely self-contained it only takes up a single USB port, so all your GPIO pins are left well alone.

The Pipsta is compatible with any Raspberry Pi, but the mounting holes in the clear acrylic chassis are designed for the current-generation models (B+, A+ and Pi 2 Model B). Every port of the Pi is fully accessible, so there's nothing stopping you setting it up inside the Pipsta for everyday use – even the GPIO and camera expansion ports are easily accessed.

Getting set up

It doesn't come with a printed manual, but it does (on the 12-month warranty card) offer a link to assembly instructions online. To give its maker Able Systems credit, the instructions are very robust and definitive, which is handy, because there's a lot of software cajoling involved.

Print quality is excellent and, like most thermal printers, it's quick and quiet in operation. It comes with two types of thermal printer rolls to get you started – a standard thermal paper roll and a label roll, which has a sticky back that can be applied immediately (no peeling required).

Overall, it's a great device and the team behind it are certainly doing their best to generate a community around it. There are a growing selection of projects created with the Pipsta at their core on their website, including printing ultrasonic readings, printing via HTML, NFC-powered projects, and more besides: **pipsta.co.uk/projects**.

Related

ADAFRUIT IOT PI PRINTER

Billed as a fun weekend project, this IoT printer kit requires a bit more build time and expertise than Pipsta (including some soldering).

£110 / $165
modmypi.com

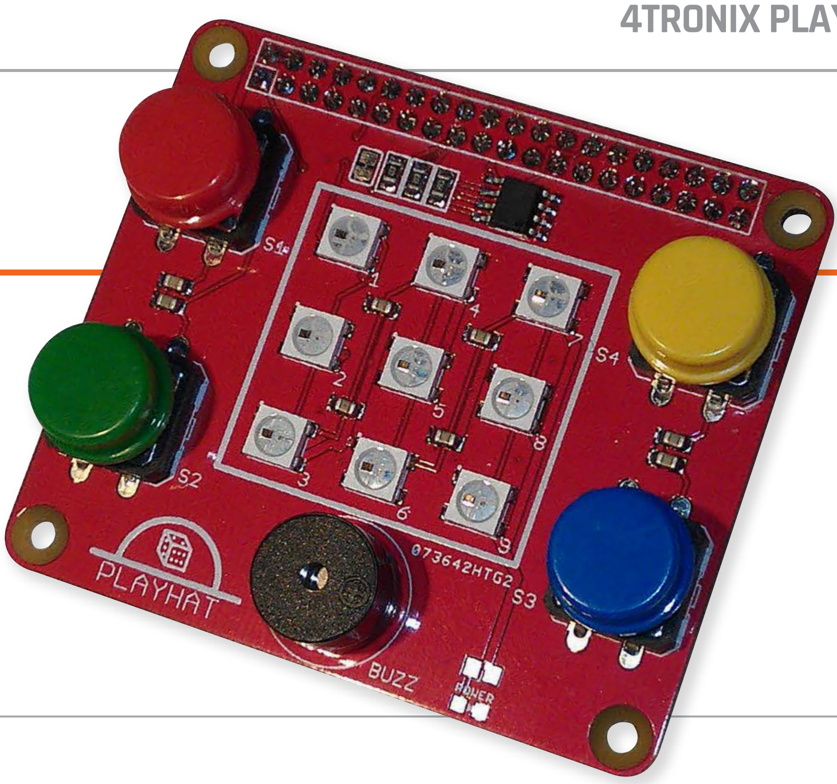
Last word

The Pipsta makes a brilliant intelligent print server and offers lots of new ground for Raspberry Pi project makers to explore. It's not cheap, but it is powerful.

★★★★☆

4tronix.co.uk

£12/$18



Maker Says

"A great way to learn GPIO, in either Python or Scratch"
4tronix

4TRONIX PLAYHAT

Les Pounder looks at an affordable add-on board aimed at introducing physical computing using simple inputs and outputs

4Tronix is perhaps best known for its range of Raspberry Pi-powered robots, such as Pi2Go and, more recently, the A+ powered Agobo. But 4Tronix is not just about robots. The Derby-based company also develops add-on boards to enable children to learn computing. It started with the PiDie, a 3×3 LED board that enabled users to play dice games and control traffic lights. Then we saw the ultra-cheap Pi Stop traffic-light sticks that clipped onto the GPIO.

The PlayHAT marks 4Tronix's first HAT board; a refinement of those earlier two LED products, it combines them into one neat package. Fitting neatly on top of the 40-pin GPIO present on the A+, B+, and Raspberry Pi 2, the PlayHAT boasts an impressive nine multicoloured NeoPixel LEDs, four push buttons, and a single buzzer. The board is colourful and well built, with components that have been thoughtfully picked with tiny hands in mind.

Setting up

Installation is remarkably easy, requiring only a download of the libraries, along with code examples from 4Tronix's GitHub repository. Upon installation, it is wise to run the strandtest.py script to ensure everything has been installed correctly. A quick word of warning, though: it would be wise to diffuse the NeoPixels with a piece of paper, as they really are very bright.

PlayHAT is programmed using Python, but rather than use a bespoke module, it employs the standard RPi.GPIO library to drive the buttons and buzzer. The NeoPixels, on the other hand, require the Adafruit NeoPixel library, contained in the GitHub download, to be used properly.

4Tronix has included example programs to illustrate how the PlayHAT works. The strandtest script runs various animation sequences on the NeoPixels, while the playhat script has further NeoPixel demos, but also comes with a simple dice game that illustrates how the board can be utilised to create a novel method of output.

This is a simple and easy-to use board, with which much hacking and making can be done.

Related

PIBRELLA

Pibrella offers LEDs, button, and buzzer just like the PlayHAT, but it also offers four inputs, and four outputs that can be used with external components, such as motors and sensors.

£10 / $15

pibrella.com

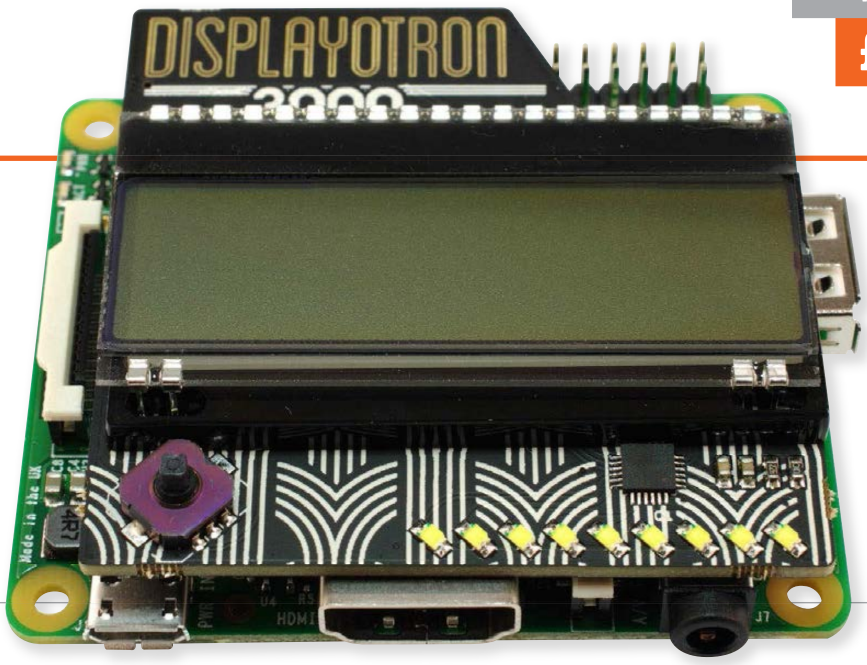
Last word

Not the most expansive or feature-rich board on the market, but what it lacks in features, it makes up for in its price and ease of use. This is a great 'getting started' board for physical computing with the Raspberry Pi.

★★★★☆

shop.pimoroni.com

£24/$30



Maker Says

“3x16 character display with three RGB backlights to give you D.I.S.C.O capability

Pimoroni

DISPLAY-O-TRON 3000

Has Pimoroni’s art deco-themed Display-o-Tron add-on done the impossible and dethroned the PiFace Control & Display? **Gareth Halfacree** finds out…

The PiFace Control & Display made quite a splash at launch, offering a simple plug-in board – long before a HAT was a thing – which provided embedded Pi projects with a simple two-line LCD display, a handful of buttons, and a small joystick. Since then, it has remained the king of the hill for any embedded project that doesn’t require a graphical display, but Sheffield-based Pimoroni could change all that.

The first thing that strikes you about the Display-o-Tron 3000 is its appearance. As soon as you remove it from the anti-static bag, you spot that it’s no ordinary device. The board has been designed with an art deco theme in mind, featuring visual embellishments to its black finish that you rarely see on rival products.

The second thing you notice is the size. Compared to the PiFace Control & Display it was so clearly inspired by, the Display-o-Tron is barely there. Measuring just over 12mm thick including the low-profile GPIO connector on the rear, it adds just 6mm to the height of any model of Raspberry Pi and is designed to stay within the footprint of the compact Model A+.

Despite this, the Display-o-Tron is brimming with features. Although it lacks the extra buttons of the PiFace C&D, the display is a larger three-line, 16-character display with three individually controllable RGB LED backlights. A five-way joystick provides directional control and push-to-click activation for your projects, while the spare GPIO lines are broken out into a nine-segment LED bar-graph along the bottom which illuminates on demand in a retina-searing white.

GPIO out

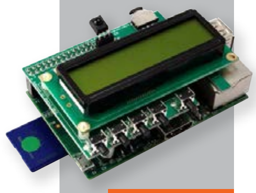
That feature list does reveal one drawback to the Display-o-Tron, however: it ties up the entire GPIO capabilities of the Raspberry Pi Models A and B. Those whose projects use the A+, B+, or Raspberry Pi 2 Model B – support for which was added to the RPi.GPIO Python library used by the Display-o-Tron around two weeks after launch – will still retain access to the extended header pins.

Programming the display, LEDs, and buttons is straightforward, with a quickly-installed library and handy example files for everything from controlling VLC to play internet radio, to playing a simple game.

Related

PIFACE CONTROL & DISPLAY

The PiFace Control & Display is undeniably bulkier than Pimoroni’s Display-o-Tron, but the extra buttons and cheaper price – in the UK, at least – could prove a worthy trade-off for some projects.



£14 / $29

pifacee.org.uk

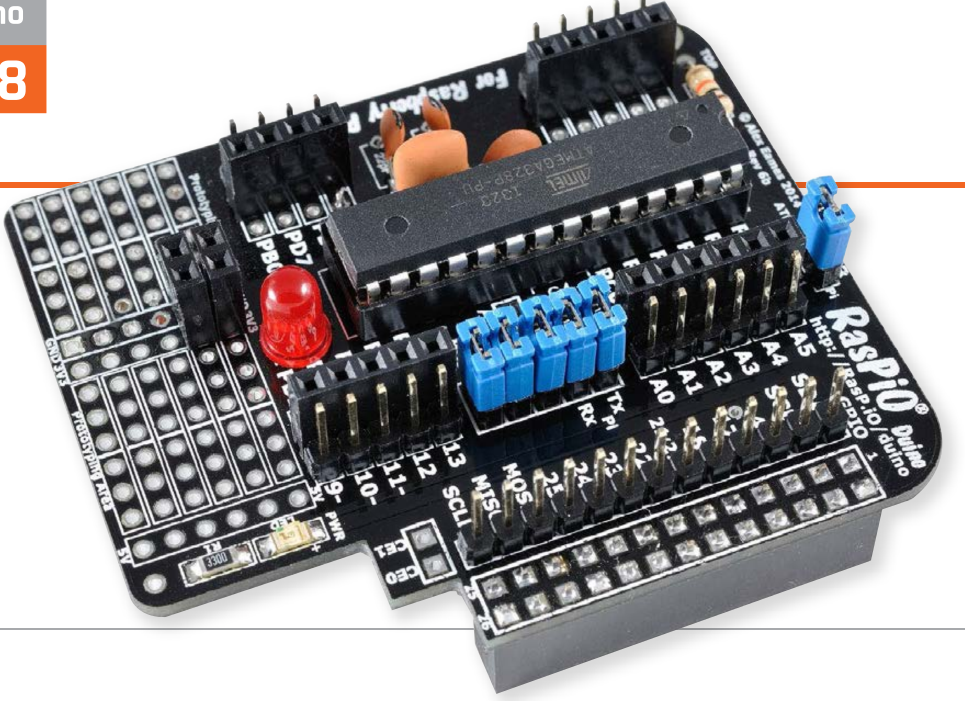
Last word

The Display-o-Tron is mightily impressive for its diminutive size and boasts capabilities better than its far more sizeable rivals. It’s not particularly cheap, however, and takes up the entire 26-pin header of older Pi models.

★★★★☆

rasp.io/duino

£12/$18



Maker Says

Learn Arduino programming on Raspberry Pi RasP.iO

RASPIO DUINO

The Arduino and Raspberry Pi joined into one glorious package? **Les Pounder** grabs a box of LEDs and buttons and starts hacking

Alex Eames is no stranger to Kickstarter: he was part of the team that brought us the excellent HDMIPi in 2014. Fresh from that success, he has launched another product based on a successful crowdfunding campaign. RasPiO Duino enables a Raspberry Pi to work harmoniously with the Arduino microcontroller board. Coming as a kit which requires around 30 minutes to solder, the Duino attaches to all models of Raspberry Pi via the GPIO pins and presents the familiar ATmega328P microcontroller found in the most popular Arduino, the Uno. The digital and analogue pins of the ATmega328P are broken out to a series of male and female headers around the board; providing both types of connection is a nice touch and really goes the extra mile for makers. Another nice touch is breaking out the I2C, SPI, and a selection of Raspberry Pi GPIO pins, as the Duino covers 26 of the 40 pins present on the Raspberry Pi 2 and the A+/B+ boards.

Installation of the software for the Duino was a little tricky and we did uncover one bug with permissions during installation – but, after a brief chat with Alex Eames, this issue was resolved and Alex is working to refine the install process for release.

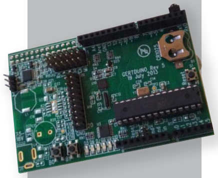
Arduino examples

So you will be thinking, can I use it just like an Arduino? Well, the answer is a resounding yes! We ran two tests that are the most common ones when first dabbling with Arduino: the Blink sketch and Button sketch. We used the Blink example found under Examples in the Arduino IDE and uploaded it to the board, ensuring that we first set up the Board and the Programmer via the Tools menu, and then used the Upload Using Programmer option in the File menu. After around 1 minute on a Raspberry Pi B+, the sketch had uploaded and our LED blinked into life. We then wired up the Button example, connecting the button to 5V power, Ground, and finally PD2, which is digital pin 2 on an Uno. Uploading the sketch to the Duino took less than 1 minute and we had a working push-button-controlled LED. Alex has done a great job of making the Arduino accessible to the Raspberry Pi community.

Related

GERTDUINO

The Gertduino is an Arduino Uno compatible board designed by Gert van Loo of Raspberry Pi fame. It features an Arduino shield-compatible layout, enabling add-on boards to be fitted.



£22.64 / $45

uk.farnell.com

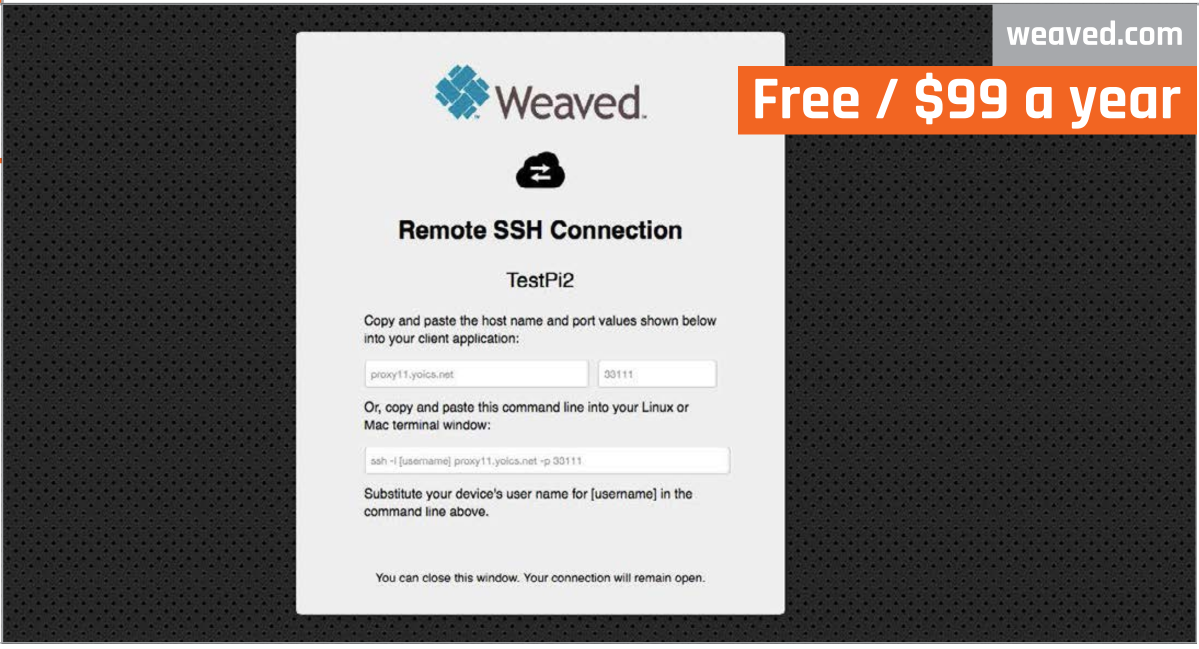
Last word

The Duino really does provide the best features of the Arduino and shows that the Arduino and Raspberry Pi can work together to make bigger and better projects. Alex has created a lovely-looking and efficient board.

★★★★☆

weaved.com

Free / $99 a year



Remote SSH Connection

TestPi2

Copy and paste the host name and port values shown below into your client application:

proxy11.yoics.net | 33111

Or, copy and paste this command line into your Linux or Mac terminal window:

```
ssh -l [username] proxy11.yoics.net -p 33111
```

Substitute your device's user name for [username] in the command line above.

You can close this window. Your connection will remain open.

Maker Says

Add the power of IoT and mobile in 15 minutes

Weaved

WEAVED IOT KIT

Can a package designed to take the fuss out of port forwarding be just what the Raspberry Pi needs? **Gareth Halfacree** tests the beta...

The Weaved IoT Kit for the Raspberry Pi is a grand name for a tool which promises to take the pain out of remote connectivity. While the Pi proves an easy platform for running various handy services, accessing these from outside your local network can prove tricky – and securing said access against third-party intrusion more difficult still.

This is, at its heart, the problem Weaved looks to solve. Its free membership tier offers control of five TCP network ports across a maximum of two devices, $25 a year extends this to unlimited ports across five devices, while $99 a year gives you support for 25 devices.

Installing the Weaved client on your Pi should be simple. During installation, however, we discovered that the password we had generated for the site wasn't supported in the client; sadly, resetting the password made no difference. We ended up creating an entirely separate account, an issue Weaved tracked down to an obscure bug which should now be resolved.

Configuration

Configuring Weaved is a mixture of simple and awkward. The installer offers four preset services – SSH, VNC, WebIOPi, and web – along with the option to forward any TCP port. Weaved doesn't install any services itself, though: while it'll be happy to forward WebIOPi for you, the service will only work if you've installed it manually. There's also no quick way to forward multiple ports at once beyond running the installer again.

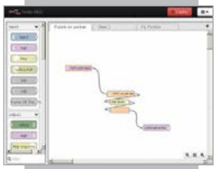
When forwarded, ports become accessible from Weaved's developer portal. Click on a Pi and it will open a new window while simultaneously triggering a proxied connection to your chosen port. While more secure and arguably easier than traditional port forwarding, this approach has an impact on performance. Our test SSH connections dropped to a tenth of their usual speed – another rare bug, Weaved told us.

An iOS mobile app provides even more functionality: as well as being able to trigger the connections to open while you're out and about, it works with the client software to receive push notifications from your applications. Free accounts are limited to 300 notifications a month, the $25 tier increases this to 1,500, while $99 removes the limit entirely. An Android app was in alpha at the time of writing, and did not support push notifications.

Related

THE THINGBOX PROJECT

Where Weaved is little more than a port-forwarding tool, ThingBox is a true IoT development kit – but one which, at present, doesn't support the new Raspberry Pi 2.



Free

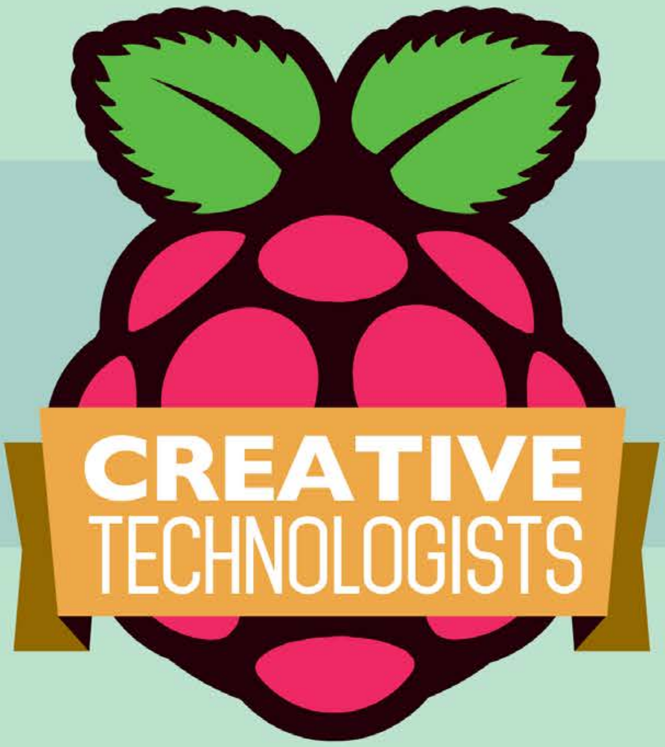
thethingbox.io

Last word

Weaved is an excellent idea, but there's still work to be done before its final release. We're looking forward to revisiting Weaved once it's out of beta.

★★★☆☆

RASPBERRY PI



12-month mentoring programme for 16 to 21 year-olds who are interested in creative technology

Mentoring, industry and creative field-trips, technical support, and equipment bursaries.

Applications by 9am, 30 March 2015

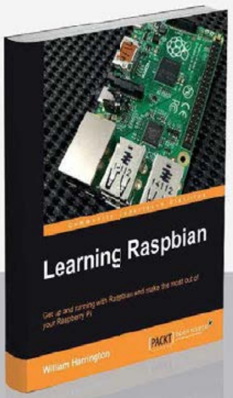
Find out more at:
www.raspberrypi.org/creatives
@raspberry_pi

Future Everything V&A WRITERS' CENTRE NORWICH National Centre *for* Writing PIMO RONI

RASPBERRY PI BESTSELLERS

Packt's 2015 releases for Raspberry Pi users feature three highlights to rival last year's wide-ranging selection…

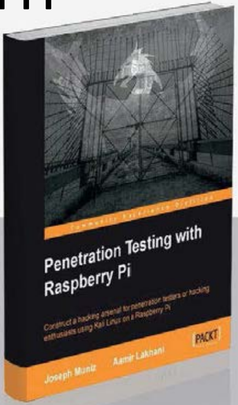
LEARNING RASPBIAN



Author: William Harrington
Publisher: Packt
Price: £15.99
ISBN: 978-1784392192
tinyurl.com/kk6dac5

Brief but useful 'missing manual' for Raspbian-equipped Pis. Assumes no Linux knowledge and covers getting started, installing and using software, the command line, and even interesting alternatives to Raspbian.

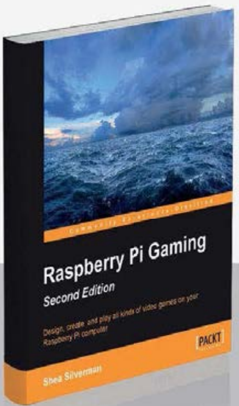
PENETRATION TESTING WITH RASPBERRY PI



Authors: Aamir Lakhani & Joseph Muniz
Publisher: Packt
Price: £18.99
ISBN: 978-1784396435
tinyurl.com/ksjdw33

Not a novelty – a serious look at the Pi's possibilities as a pen testing tool that can go where other computers cannot reach. Makes a useful networking security introduction, too.

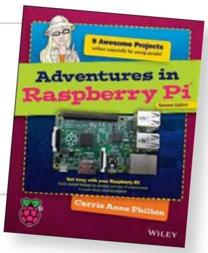
RASPBERRY PI GAMING 2ND EDITION



Authors: Shea Silverman
Publisher: Packt
Price: £15.99
ISBN: 978-1784399337
tinyurl.com/ncraeez

Gaming and entertainment on the Raspberry Pi: coding your own *Flappy Bird* clone in Scratch, emulators, classic Linux games, and connecting controllers – it's all covered in this highly informative book.

ADVENTURES IN RASPBERRY PI 2ND EDITION



Author: Carrie Anne Philbin
Publisher: Wiley
Price: £14.99
ISBN: 978-1119046028
tinyurl.com/p5lsdga

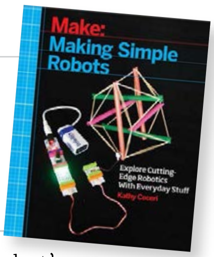
Philbin answers the question 'what can I do with my Pi?' – and does so with adventures! Ostensibly aimed at 11 to 15 year olds, with each chapter (or adventure, as they're labelled) a gentle introduction to an essential Pi topic, it can also be used to help younger learners into coding and learning about hardware, through the last two chapters.

Before hardware – and a GPIO-connected marshmallow (!) – readers start with the command line, the first essential for a generation brought up in GUI-only environments. Then Scratch is introduced in enough detail for those who haven't used it before, turtle graphics, Python, programming *Minecraft* worlds, and Sonic Pi for making music.

Learning resources on the companion website, including videos and achievement badges for each adventure completed, reinforce the lessons. Each written section is very well paced for the intended age group, with tips, new information, and additional challenges appearing to vary the pace along the way. The colourful layout makes for a far more digestible read than many of the books that come our way, perhaps one of the reasons that this book is also popular with readers older than the intended audience! Next steps for each adventure ensure the learning – and the fun – can continue. Recommended.

Score ★★★★★

MAKING SIMPLE ROBOTS



Author: Kathy Ceceri
Publisher: Maker Media
Price: £16.50
ISBN: 978-1457183638
tinyurl.com/pmdle7g

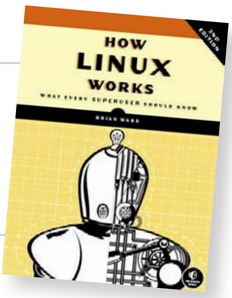
'Anyone can build a robot' is the enticing premise of Ceceri's new book. Building on her previous popular guide to no-tech and low-tech robotics, she pushes the boundaries to what can be achieved using everyday materials and just a smidgen of technology.

What is a robot? The book's definition of a machine which can 'sense, think, act' is a broad umbrella for projects ranging from an inflatable robot (à la Big Hero 6) to a swarm of vibrobots from recycled parts. Every project is designed to be manageable for those new to hardware geekery, although some will push the reader away from their comfort zone. Along the way, skills will be gained, knowledge of fascinating areas of robotic research learned, and fun had. Kathy Ceceri is a natural educator: the learning comes lightly from each page read, carried easily by the author's passion for the subject.

Many projects need no computer involvement – a couple of Arduino wearables and some littleBits-based builds are among the exceptions. However there is a Scratch-programmed chatbot project, which Ceceri suggests could be loaded onto a Raspberry Pi and fitted into a robotic body. Other than that, the low-tech angle really helps to keep the focus on simple, achievable projects which combine fun and learning in unusually successful forms.

Score ★★★★☆

HOW LINUX WORKS



Author: Brian Ward
Publisher: No Starch
Price: £26.50
ISBN: 978-1593275679
nostarch.com/howlinuxworks2

"You should never have to fight with a computer," says Brian Ward in his preface to this thoroughly revised and updated guide to the inner workings of our favourite kernel and operating system. Part of avoiding a struggle with your computer is understanding, and Linux is open to all with its plain text configuration files, open source code, and vast online resources.

If the Pi is your first Linux computer, you may look at the title and think this book is more than you need to know. Not so: the fractal layout – starting with basic information, and expanding in finer detail – gives you as much information as you wish to read in each chapter. Just stop when you've learned enough on a topic, then continue later when you're curious.

Ward sets out a more kernel-centred work than the first edition, walking the reader through devices, disk layout, boot order, user space, logging and other housekeeping, processes, and networking. There's something here for every level of user, and while well-written enough to work right through, drawing you onwards, it also earns its place on the shelf as an essential reference – particularly as previous Linux classics have not been updated to keep up with changes in the workings of recent kernels.

Score ★★★★☆

CLOJURE WEB DEVELOPMENT ESSENTIALS

Author: Ryan Baldwin
Publisher: Packt
Price: £27.99
ISBN: 978-1784392222
tinyurl.com/lva8c9r

If you've been intrigued by the promises of functional programming (see sidebar), but can't quite see the advantage for that most practical of environments, the web, this could be the introduction for you. Demanding perhaps minimal Clojure experience (those with some exposure to Common Lisp will quickly pick up the thread), Baldwin takes the reader though building a web application in Clojure from scratch, using the Luminus application template and the Ring web application library.

Using these two sensible default choices – which saves a lot of unnecessary discussion on alternatives – means the structural details of the stack and the build dependencies can be packed into the first two chapters. Stick with the book through these; it's worth the effort, even if you're a relative Clojure newbie. Now comes the fun part: really building the app.

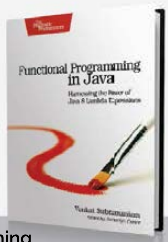
From simple routing with Compojure, through the Django-inspired Selmer, pages appear, then forms are handled, and the YeSQL library is introduced. Somewhere along the way – perhaps with bypassing the pain of ORM – readers are likely to find an 'aha' moment, as they discover how much easier things can be in Clojure than their previous framework and language of choice.

Score ★★★☆☆

ESSENTIAL READING: FUNCTIONAL PROGRAMMING

With multicore processors (like on the Pi 2) driving a functional programming revival, it's time to join a higher order of coding…

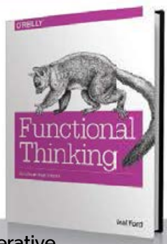
Functional Programming in Java



Author: Venkat Subramaniam
Publisher: Pragmatic
Price: £21.99
ISBN: 978-1937785468
tinyurl.com/lbdrwb7

Persuasive case for functional programming in Java. Dr Subramaniam shows the way for expressive, robust, parallelisable Java.

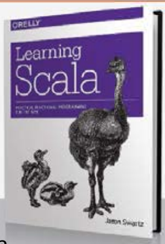
Functional Thinking: Paradigm Over Syntax



Author: Neal Ford
Publisher: O'Reilly
Price: £26.50
ISBN: 978-1449365516
tinyurl.com/pokj7l2

Takes Java programmers, used to an imperative coding style, through the functional concepts used in Java 8, Scala, Groovy, and Clojure.

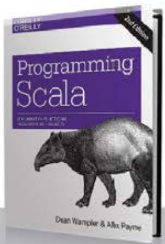
Learning Scala



Author: Jason Swartz
Publisher: O'Reilly
Price: £26.50
ISBN: 978-1449367930
tinyurl.com/k635sz6

Compact, indeed concentrated, introduction to the expressive, flexible Scala language, building on your OO knowledge – ideal for Ruby and Python coders.

Programming Scala 2nd Edition



Author: Dean Wampler & Alex Payne
Publisher: O'Reilly
Price: £33.50
ISBN: 978-1491949856
tinyurl.com/pp23jl7

Comprehensive guide to Scala, aimed mostly at professional Java programmers: a good guide and reference for the rest of us.

Seven Concurrency Models in Seven Weeks

Author: Paul Butcher
Publisher: Pragmatic
Price: £25.50
ISBN: 978-1937785659
tinyurl.com/og4caxo

Butcher walks us through seven models of parallelism and concurrency, using mostly – but not exclusively – Clojure.

RASPBERRY JAM EVENT CALENDAR

Find out what community-organised, Raspberry Pi-themed events are happening near you…

PUT YOUR EVENT ON THE MAP

List your forthcoming events at:
raspberrypi.org/jam

1 MELBOURNE RASPBERRY JAM
When: Sunday 29 March
Where: 5 Kent Lane, Hawthorn, Melbourne, Australia
bit.ly/1EYbNmm
A monthly geek gathering held from 1pm to 4pm at the Connected Community Hacker Space.

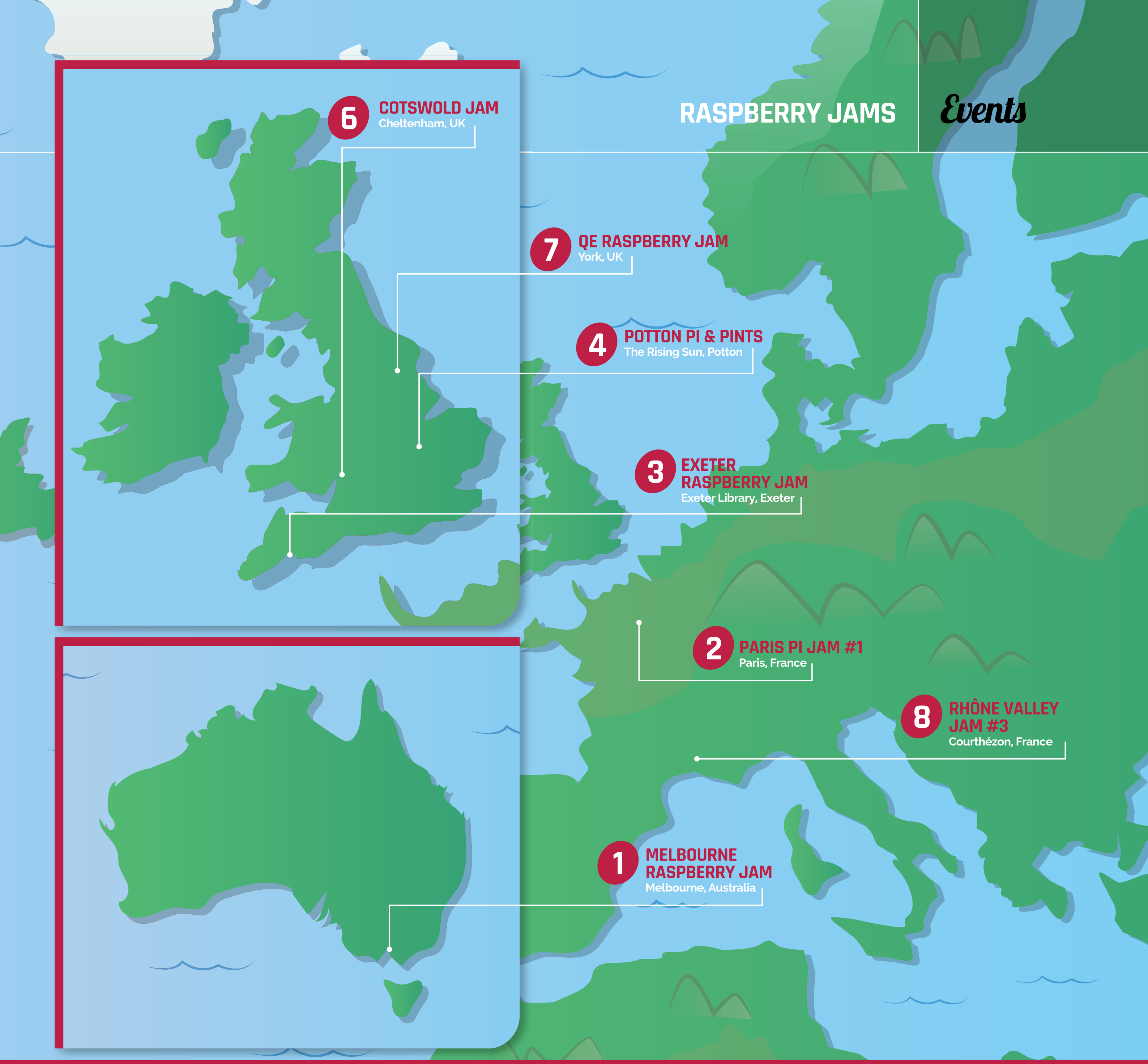
2 PARIS PI JAM #1
When: Monday 30 March
Where: 156 Boulevard Haussmann, Paris, France
meetup.com/Paris-Raspberry-Pi-Jam
It's the inaugural Paris Raspberry Jam and we've got Benoit Gaigal to thank for it. What a guy!

3 EXETER RASPBERRY JAM
When: Saturday 4 April
Where: Exeter Library, Exeter, Devon, UK
bit.ly/1BtsGle
Exeter Jam takes place between 10am and 12pm, and – as always – features plenty of projects, help, and advice!

4 POTTON PI & PINTS
When: Saturday 11 April
Where: The Rising Sun, Potton, Beds, UK
bit.ly/1FX0Um4
Enjoy a relaxing chat with like-minded Pi enthusiasts (from makers of CamJam).

5 SEATTLE RASPBERRY JAM
When: Wednesday 15 April
Where: 815 Seattle Blvd S, Suite 112, Seattle, USA
meetup.com/Jigsaw-Renaissance
It's sponsored by Jigsaw Renaissance, a non-profit maker space in Seattle's International District.

6 COTSWOLD JAM
When: Saturday 18 April
Where: 95 Promenade, Cheltenham, UK
cotswoldjam.org
This Jam is organised by Andy 'Python quadcopter' Baker and Andrew 'goblin detector' Oakley.



7

QE RASPBERRY JAM

When: Saturday 25 April
Where: Queen Ethelburga's Collegiate, York, UK
bit.ly/19UCkGX
Join in with QE's first family-friendly Jam for up to 200 people of all ages and levels of experience.

8

RHÔNE VALLEY JAM #3

When: Sunday 26 April
Where: 13 Avenue Jean Jaurès, 84350 Courthézon, France
bit.ly/19oFNT7
Alan McCullagh and Amaury Doucet are taking a third slice of Pi in Rhône Valley.

DON'T MISS: COTSWOLD JAM

When: **Saturday 18 April** Where: **Cheltenham**

Cotswold Jam is special in as much as it's organised by two favourites of the Raspberry Jam scene, Andy Baker and Andrew Oakley. Andy is well known for his excellent project blog, **blog.pistuffing.co.uk**, which follows his fascinating trials and tribulations of building a Raspberry Pi quadcopter from scratch. Andrew (**aoakley.com**) is known for his excellent Pi projects, often the highlight of CamJam. In short, it's worth going just to chat to these two! **cotswoldjam.org**

Happy 3rd Birthday, Raspberry Pi,
with love from Cotswold Raspberry Jam
RASPBERRY JAM Cotswold
www.cotswoldjam.org
Next event Sat 18 April, Cheltenham
See you at the Party - Andy

In association with:

The PiHut
www.ThePiHut.com

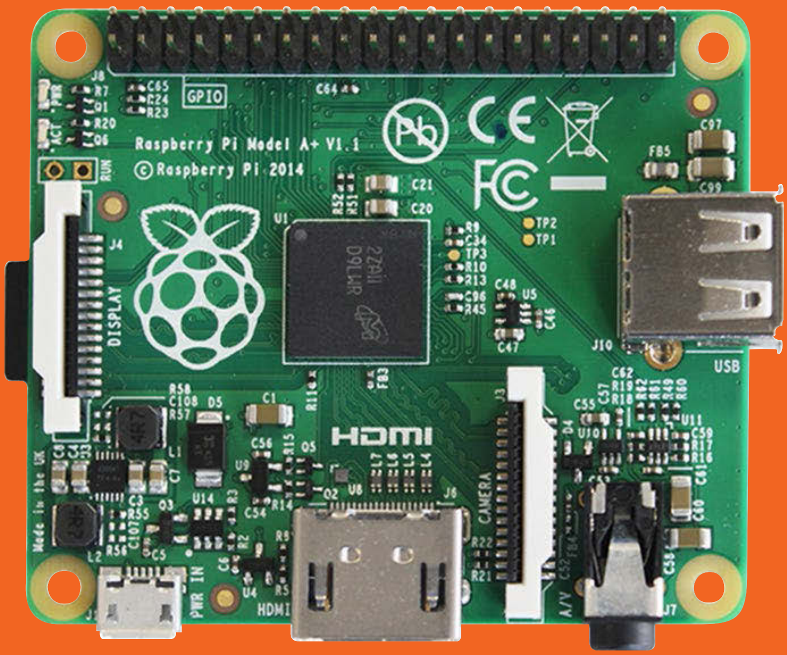
10 RASPBERRY PI Model A+s MUST BE WON!

WHAT WOULD YOU LIKE TO SEE MORE OF IN The MagPi?

Tell us by 28 April 2015 for a chance to win!

WHAT DO YOU LIKE?

- **News & features**
- **Project showcases**
- **Hardware tutorials**
- **Software guides**
- **Product reviews**



The Raspberry Pi Model A+. It's the connoisseur's choice, you know...

How to enter:

Simply email **competition@raspberrypi.org** ranking the above list in order of your favourite to least favourite.
Want to add an 'other' category?
Tell us about it in no more than 25 words.

Terms & Conditions

Competition closes 28 April 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.

DEXTER
INDUSTRIES

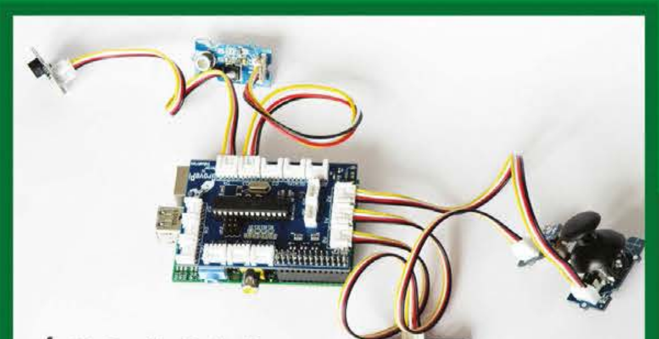
SAVE
10%
"MagPi15"
discount code

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

MATT RICHARDSON

Matt is Raspberry Pi's US-based evangelist. Before that he was co-author of *Getting Started with Raspberry Pi* and a contributing editor at *Make:* magazine.

MAKING FAILURE AFFORDABLE

In his column this month, the Raspberry Pi Foundation's **Matt Richardson** explores the beauty of 'affordable failure'

Whenever I tell people about Raspberry Pi, I usually say that the best feature is its price. This is especially true for the new Raspberry Pi 2, which packs about six times the performance compared to its predecessor, and yet keeps the $35 price tag. To say that Raspberry Pi is good value would be a huge understatement. But that's not the only reason why the price is so important.

In a 2012 BBC News interview, Linus Torvalds – the chief architect of the Linux kernel – praised the board, saying that it encourages a wider group of people to tinker with computers. But, more importantly, he pointed out that the price of Raspberry Pi allows people to "afford failure". Linus went on to explain that even if you bought a Raspberry Pi, but decide that tinkering with computers isn't right for you, you can afford to put it aside and say "I don't care."

"If it's cheap enough, you can afford to have a lot of 'don't cares', if then – every once in a while – you end up triggering even a fairly rare 'do care' case," said Linus.

Those 'do care' cases can have a massive impact on the long-term future of computing. I like to think that it's quite possible that the next Linus Torvalds is tinkering with a Raspberry Pi right now. But the point Linus makes is only one aspect of how Raspberry Pi enables 'affording failure'.

Avoiding the safe zone

For much of my own childhood, our family shared a single computer. I did my fair share of tinkering with it, but I stayed in the 'safe zone'. After all, if I messed something up and put the computer out of service, it meant it was unusable for the rest of the family as well. It was a risk I didn't take often.

Not that I want to minimise the impact and opportunity that our family computer offered me when I was growing up. But, I wonder how different things would be had I owned my very own Raspberry Pi as a kid. It could have been my technological sandbox where a failure is fine.

When I talk to curious parents about Raspberry Pi, I emphasise that it's meant to offer their children ownership of a computer. In turn, their kids can tinker with impunity. If something gets messed up, re-imaging the SD card will usually give them a fresh start. And if they somehow break their Raspberry Pi beyond repair, then it was only $35 and a lesson learned. Failure is not only fine, but it's also more affordable.

This idea of destigmatising failure isn't just for kids. In the business world – and especially among technology startups – failure has been incorporated into mantras like 'fail fast, fail often' or 'move fast and break things'. The latter comes from Facebook CEO Mark Zuckerberg in his letter for the site's IPO filing, where he describes Facebook's mission and ethos. He goes on to say that "the riskiest thing is to take no risks". Even if none of us is planning an IPO any time soon, it's advice that we can take to heart.

If you want to broaden your technological horizons, it doesn't matter whether you're failing fast or slow; the important thing is that you're doing *something* to get out of your safe zone. There's so much you can do with Raspberry Pi 2's quad-core 900MHz processor and 1GB of RAM. But to stay safe and comfortable with your Raspberry Pi is taking for granted one of the best specs of the board: $35.

Expand your Pi

Stackable expansion boards for the Raspberry Pi

Serial Pi Plus

RS232 serial communication board. Control your Raspberry Pi over RS232 or connect to external serial

Breakout Pi Plus

The Breakout Pi Plus is a useful and versatile prototyping expansion board for the Raspberry Pi

ADC Pi Plus

8 channel analogue to digital converter. I^{2}C address selection allows you to add up to 32 analogue channels to your Raspberry Pi.

IO Pi Plus

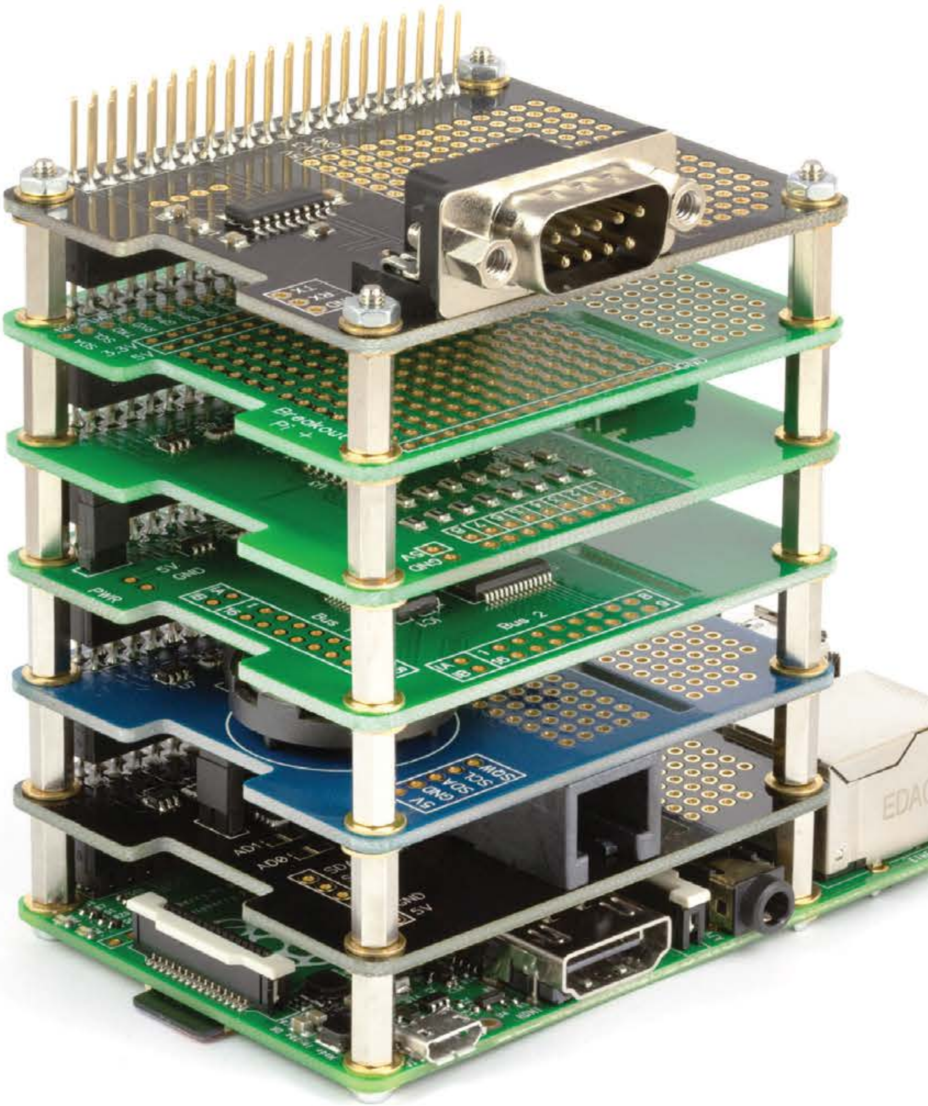
32 digital 5V inputs or outputs. I^{2}C address selection allows you to stack up to 4 IO Pi Plus boards on your Raspberry Pi.

RTC Pi Plus

Real-time clock with battery backup and 5V I^{2}C level converter for adding external 5V I^{2}C devices to your Raspberry Pi.

1 Wire Pi Plus

1-Wire® to I^{2}C host interface with ESD protection diode and I^{2}C address selection.



We also stock a wide range of expansion boards for the original Raspberry Pi models A and B

AB electronics UK

www.abelectronics.co.uk

Wyliodrin

ARDUINO

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 js Perl Java ARDUINO php