NEW! YOUR OFFICIAL RASPBERRY PI MAGAZINE

# The MagPi

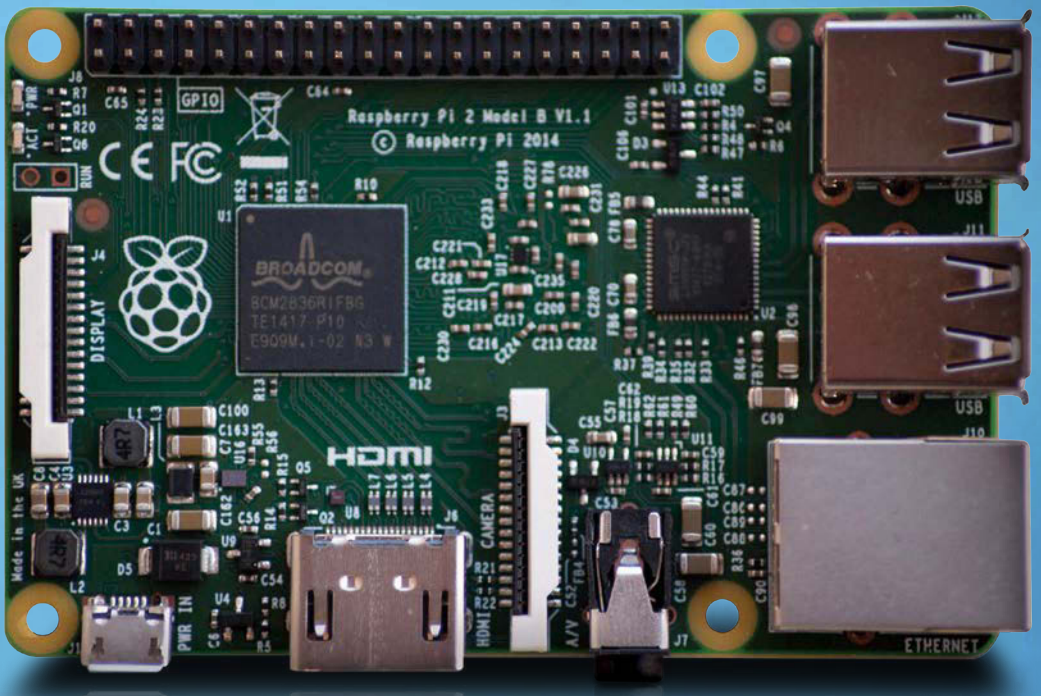
100 pages of hacking & making

Issue 36 August 2015 raspberrypi.org/magpi

Win! 10 RASPI 2s THANKS TO ELEMENT14.COM

## GET STARTED WITH RASPBERRY PI

How to unbox your credit card-sized PC and start hacking today!



**RETRO CLASSICS**
The very best games of yesteryear you can bring back to life...

**SIMULATE PHYSICS WITH PYTHON**
Create a whole solar system on your Raspberry Pi

**3D PRINTING WITH RASPBERRY PI**
Control and view your printer with OctoPrint

**CREATE A START SCREEN IN SCRATCH**
How to put the finishing touches to your latest creation

*Also inside:*

- CONQUER THE COMMAND LINE
- BUILD YOUR OWN MOOD LIGHT
- MAKE A SIMON SAYS ELECTRONIC GAME
- IQAUDIO: THE ULTIMATE AUDIO ADD-ONS

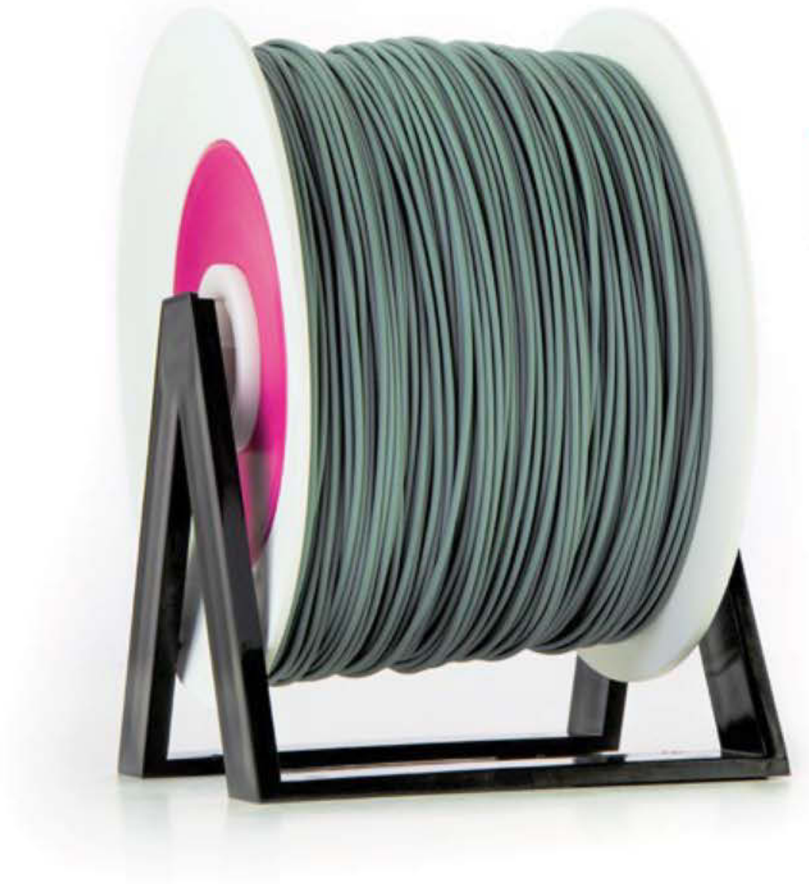
**MINECRAFT SPLAT**
Recreate Nintendo's brilliant Splatoon on your Raspberry Pi

Issue 36 • Aug 2015 • £5.99
08
9 772051 998001

THE ONLY MAGAZINE WRITTEN BY THE COMMUNITY, FOR THE COMMUNITY

EUMAKERS

the world has a new dimension



3D PRINTING

FILAMENT

OVER 60 COLORS AVAILABLE

TRY FOR FREE OUR QUALITY

GET A FREE 3D FILAMENT SAMPLE

VISIT OUR ONLINE STORE AT

WWW.EUMAKERS.COM

Made in Italy

PROUD SUPPORTERS OF: NATIONAL STEM CENTRE

# WELCOME TO YOUR OFFICIAL PI MAGAZINE!

The Raspberry Pi is now officially the second-best-selling computer ever to come out of the UK. Despite its massive audience, most people don't realise the Raspberry Pi Foundation is actually a charity, and all the profits from Pi sales are channelled into realising the Foundation's goals. The aim is to make affordable, programmable computers available for everyone, all over the world. This £15/$20 device is designed to allow children and adults from all walks of life access to the internet, games and applications, and have the opportunity to learn to code.

It's hoped it will help jump-start another paradigm shift in computer education, too, like we enjoyed when the first affordable computers of the 1980s arrived in our homes.

Just like the Raspberry Pi, the proceeds from sales of the official Raspberry Pi magazine also go towards achieving the Foundation's charitable aims. It's also all about celebrating the incredible community that's sprung up in the UK, America, and around the world. So if you've done something cool with a Raspberry Pi, or know someone who has, we'd love to hear from you.

I hope you enjoy our first print edition!

**Russell Barnes**

## THIS MONTH:

**16** GET STARTED WITH RASPBERRY PI
It's easier to unbox and set up your Pi than you might think!

**50** PRINT IN 3D WITH RASPBERRY PI
Introducing OctoPrint – the open-source way to make in 3D

**58** MAKE A TINY SOLAR SYSTEM
Use physics and forces to simulate the planets on your Pi

**66** MINECRAFT SPLAT!
We recreate Nintendo's Wii U classic, Splatoon, in Minecraft!

**FIND US ONLINE** raspberrypi.org/magpi

**GET IN TOUCH** magpi@raspberrypi.org

The MagPi

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

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

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

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

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

Contributors: **Tim Anderson, Mike Cook, David Crookes, Gareth Halfacree, Lucy Hattersley, Gina Häussge, Richard Hayler, Phil King, Ian McAlpine, Sean McManus, Simon Monk, Martin O'Hanlon, Les Pounder, Matt Richardson, Richard Smedley, Sean M Tracey** & **Robin Withers.**

Available on the App Store

Get it on Google play

BY NC SA

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

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

# Contents

Issue 36 August 2015 raspberrypi.org/magpi

## TUTORIALS

> BUILD A MOOD LIGHT 44
Join Dr Simon Monk for more Everyday Engineering fun!

> MASTER THE COMMAND LINE 48
Learn more command prompt skills with Richard Smedley

> 3D PRINT WITH OCTOPRINT 50
Start and monitor your creations with your Raspberry Pi

> MAKE A SCRATCH TITLE SCREEN 52
Author and MagPi regular Sean McManus shows us how

> MIKE'S PI BAKERY: STRING PONG 54
Author and hacker Mike Cook creates another corker

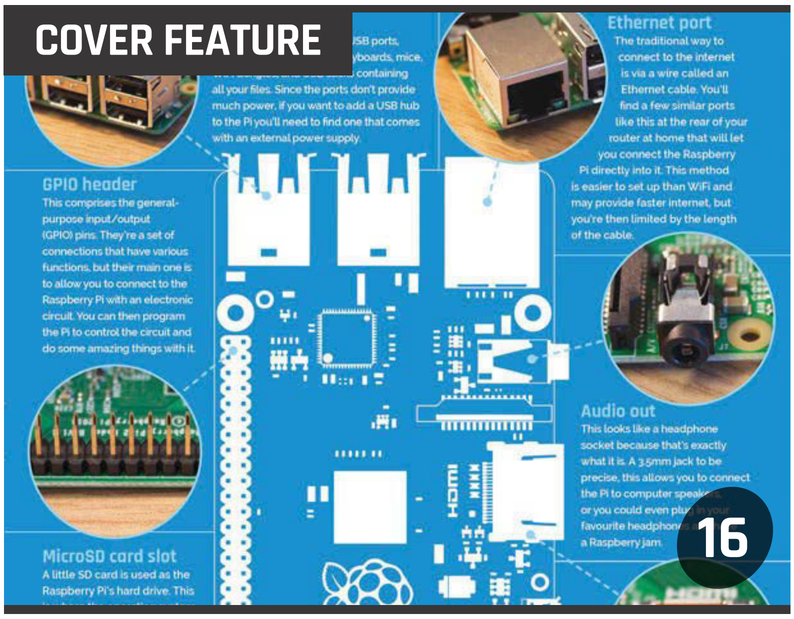
> PYTHON PHYSICS AND FORCES 58
How to build a solar system inside your Raspberry Pi

> BUILD A SIMON SAYS GAME 64
Test your memory with this Explorer HAT-inspired creation

> MINECRAFT SPLAT 66
Martin 'Minecraft' O'Hanlon recreates Splatoon!

> LIVE CODING WITH SONIC PI 72
The creator of Sonic Pi starts a new tutorial series

## COVER FEATURE



UNBOX YOUR RASPBERRY PI
It's easy to get started with the world's favourite credit card-sized PC!

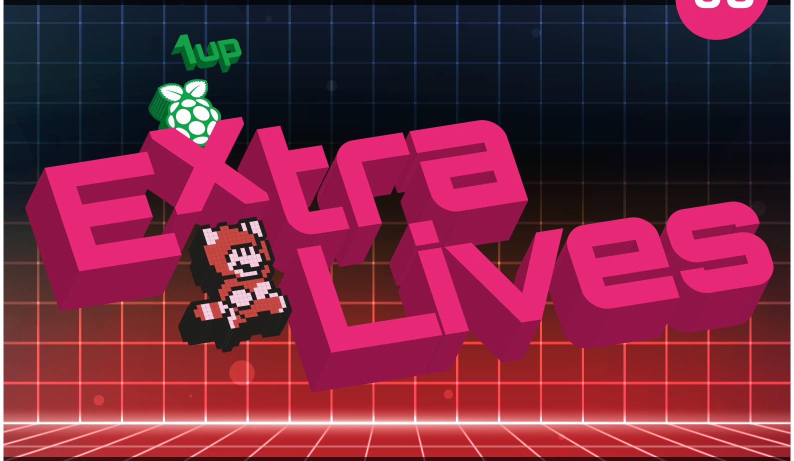
## IN THE NEWS

6
THE CREATIVE TECHNOLOGISTS
We get together with the young people exploring and using tech in new ways, thanks to Raspberry Pi

QUIET PLEASE! HISTORIANS AT PLAY
The MagPi speaks to The British Library Labs about a new project to do exciting things with useful data
8

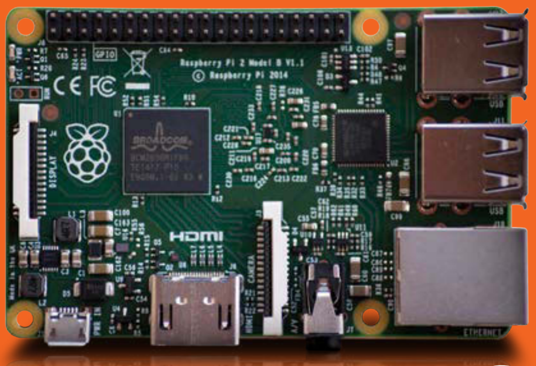
12
INSIDE LIVERPOOL MAKEFEST
Daleks, textiles, Minecraft, robots, and musical fruit and veg. The first ever Liverpool Makefest was anything but boring

THE BIG FEATURE 38



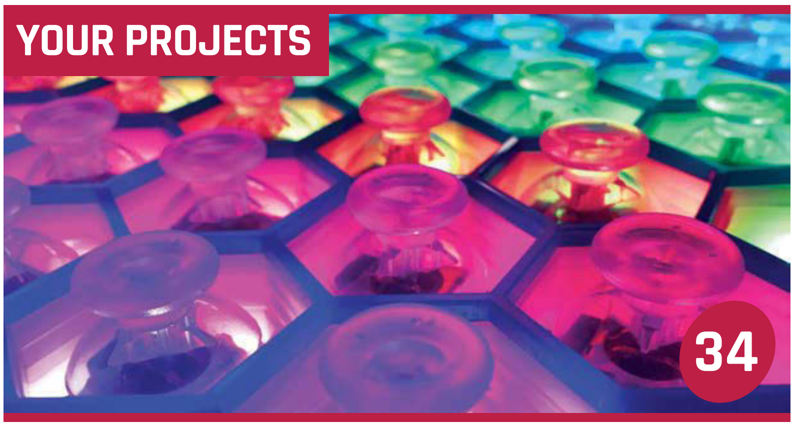
The Raspberry Pi is helping to revive some age-old gaming classics. The MagPi investigates...

10 Raspberry Pi 2s MUST BE WON!



Don't miss your chance to win a Raspberry Pi 2 in this month's reader competition... 91

YOUR PROJECTS

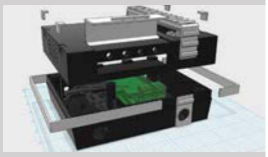


34

THE JOYTONE
Learn all about this amazing and totally unique kind of musical instrument

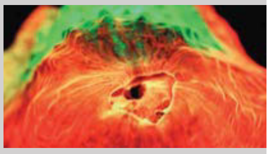
Pi Notebook 26
Adafruit's Ruiz brothers do it again, this time creating a tiny 3D-printed retro laptop

Nin10do 28
Retro games consoles don't get much better than this. Find out how it was made...



PiScan 30
One project builder created this rather clever Amazon Dash scanner. Here's how...

Kudu 32
Find out what Kudu can do that media centre software like Kodi can't



REGULARS

> NEWS 6
Keep up to date with the biggest stories from the world of Pi

> TECHNICAL FAQS 74
Got a problem? Our experts answer all your questions

> BOOK REVIEWS 84
The latest computer books reviewed and rated

> THE FINAL WORD 96
Matt Richardson considers the creative uses of the Pi

COMMUNITY

> EVENTS 88
Find a community gathering near you in the coming weeks

> THE BIG QUESTION 92
Can the Raspberry Pi help primary school children?

> LETTERS 94
Have your say on the magazine and the community

REVIEWS

> IQAUDIO DIGIAMP+ 76
> SWANKY PAINT 78
> THE FUZE 79
> 3D PRINTING PEN 80
> ADAFRUIT HDMI BACKPACK 82
> WITTY PI 83

# MEET THE CREATIVE TECHNOLOGISTS

The first in-house Raspberry Pi young people's programme is launched

When the Raspberry Pi Foundation's Rachel Rayns and Ben Nuttall announced the Creative Technologists (CTs) programme in February, little did they know how stressful the deadline for submissions would prove. "We were blown away [by the standard of entries], but it was one of the most nerve-wracking evenings of my time at Raspberry Pi," recalls creative producer Rachel. "We hadn't realised everyone would leave it to the very last minute to hit that Submit button!"

Applicants were asked to submit a 90-second video and single-page PDF, with no hint of what was expected. Ben says this enabled entrants to showcase their creativity. "We wanted a mix of tech people and those with art and design backgrounds, and that's exactly what we got. The idea is for the tech ones to discover their creative side, and the arty ones to find a way to incorporate technology into their work."

Rachel adds, "Finding the right balance of personalities, interests and experience within the group was one of the most important elements of the selection process. We wanted a mix of people who would push each other in new directions."

## Technology and creativity

The main aim of the year-long programme is to give an opportunity to creative young people to explore technology and creativity in new and interesting ways. To aid them, a wide range of partner organisations are providing mentoring and site visits: the Victoria and Albert

## YOUNG CREATIVES

**The nine successful applicants have wide-ranging backgrounds and skills**

**Andrew**
The creator of PiNet, Andrew also runs the Northern Ireland Raspberry Jam and is studying Computer Science at Queen's University in Belfast.

**Bawar**
West London sixth-form student Bawar found out about the programme the day before entries closed, and stayed up all night making his video.

**Connor**
Working in operations at Ragworm (PCB prototyping), Connor is exposed to the maker community and regularly attends Maker Faires and hackathons.

**Hannah**
Studying Creative Writing and Theatre at Lancaster University, Hannah has an interest in exploring video game scriptwriting.

Museum Digital Programmes, Writers' Centre Norwich, FutureEverything, Pimoroni, Saladhouse, and Hellicar & Lewis.

As well as costs for food, travel, and accommodation, each CT receives a Raspberry Pi 2 starter kit and a £300 materials grant. In return, CTs are expected to take part in three hours of online video calls per month, and spend at least four hours a week working independently on their projects.

A 16-21 age range was chosen for the programme: "We were interested to catching young people in a transitional time in their lives," explains Rachel. "The younger of the range would be choosing A-level and university courses; the older end would be finishing a degree or have worked for a few years out of college/school."

> "Applicants were asked to submit a 90-second video"

## First-year guinea pigs

"This first cohort are our guinea pigs," continues Rachel. "We want to figure out the most effective ways of supporting young people in developing new creative technology projects. We are doing this through a series of field trips and mentoring sessions."

The first event was an induction weekend held at Raspberry Pi HQ in Cambridge, enabling the new CTs to get to know each other, and (for those unfamiliar with it) the Raspberry Pi. After everyone had done quick 20-slide Pecha Kucha presentations, there was a workshop using the Pi with CamJam EduKits and the Camera Module. "For some, it was their first experience with a Pi," says Ben. This was followed by a field trip to the Maker Faire at Newcastle's Baltic Centre.

Further planned events include a visit to Pimoroni for some tutoring, along with a workshop in openFrameworks from mentor Joel Lewis. While all the events will take place in the UK, one of the new CTs, Javier Vila, lives in northwest Spain. "At first I thought it would be more difficult to attend field trips than it has been," he admits, but was pleased to find that it only took him six hours to travel from his home to Cambridge for the induction.

As this is the first year, the team feel that meet-ups are an essential aspect. However, while there are no plans to roll the programme out to other countries in its current form, Rachel tells us that they are looking at ways it can scale in the UK and abroad, which may include an exclusively online mentoring format.

## A QUICK CHAT WITH YASMIN

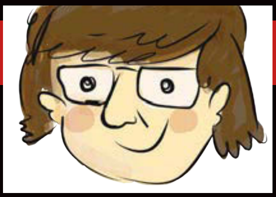
We chatted to one of the new Creative Technologists, 21-year-old Yasmin Curren...

**Did you have any previous experience of using the Raspberry Pi?**
Nope. I knew of Raspberry Pi and their involvement within the tech community, especially within education, but had never had the chance to play with one myself. So it's all a new and exciting experience for me!

**How did the induction weekend go?**
Everyone who worked at Raspberry Pi, as well as the mentors, oozed enthusiasm, really seeming to love what they do and being passionate about what they had to say, which led to one of the most positive and inspiring weekends of my life.

**What do you think of the other Creative Technologists?**
They're the most wacky, intelligent, fun, and friendly bunch of people that you'll come across and I'm so glad to know them!

To register your interest for the 2015-16 Creative Technologists programme, visit:
**raspberrypi.org/creatives/apply**



### Javier
**Living and going to school in northwest Spain, Javier has been programming since he was ten. A Pi enthusiast, he also likes to disassemble gadgets.**

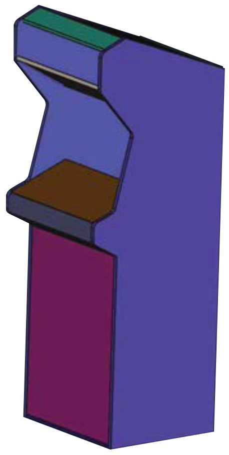
### Maddy
**Studying Visual Effects at college in Nottingham, Maddy spent three months creating an animated music video for a local band.**

### Milton
**A web developer in London, Milton wants to create worlds within worlds and explore how people interact with technology.**

### Owen
**A sixth-form student in Lewes, Owen is lightning-fast with a Rubik's cube, does magic tricks, and is keen to make things himself.**

### Yasmin
**A front-end web developer from Devon, Yasmin is a keen YouTuber, games enthusiast, content creator, storyteller, and musician.**

# HUSH! HISTORIANS AT PLAY



Organising a million images sounds like a daunting task, but thanks to an inspired bit of thinking – and a Raspberry Pi 2 – the job looks set to be a blast...

**Above right** A CAD drawing for the cabinet plans. The drawing was produced in the open-source app FreeCAD

From *Pac-Man* to *Pong*, *Street Fighter* to *Space Invaders*, arcade machines once attracted large and appreciative crowds of gamers thanks to their attractive visuals and addictive gameplay. They became a staple presence in seaside resorts and city centres, usually residing in dimly lit rooms soaked with the sweat of frustration and exhilaration.

But now a couple of them are going to be placed in the most unusual of spots – Britain's national library. Containing Raspberry Pi 2s and created from scratch, the hope is that they will be used to aid researchers in what could so easily be a mundane task.

Having noticed the power of arcade machines to inspire curiosity and turn people's heads, British Library Labs wants to use them to encourage visitors to help make sense of the British Library's huge amount of materials. "The machines will be 1980s-style cabinets with heavy-duty buttons and joysticks," says Ben O'Steen, the technical lead of British Library Labs, "but they will have fewer cigarette stains and garish colours."

The idea to use arcade machines for such a purpose came from the winning entrant of this year's prestigious British Library Labs competition. Every year, researchers are asked to pitch ideas that they would like to pursue with the organisation. Winner Adam Crymble suggested using repurposed arcade machines for crowdsourcing. By playing games, he suggested, visitors will be able to help to build a catalogue of metadata relating to the million images that have been uploaded to the British Library's Flickr account.

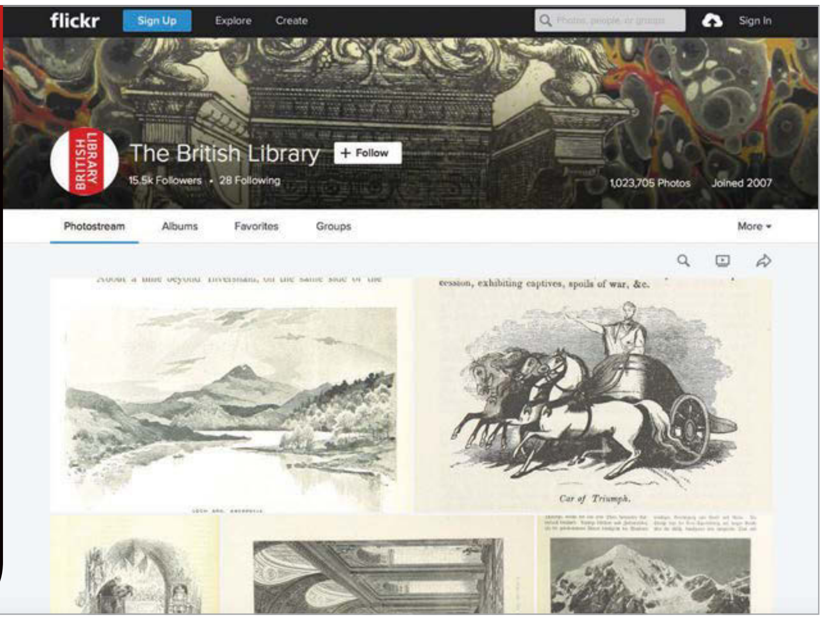
Adam is a lecturer of digital history at the University of Hertfordshire and his work involves exploring how digital data changes the way we can analyse history and engage audiences with the past. It was during an idle lunch break that inspiration hit. With a ban on

**Below right** The images on Flickr that British Library Labs needs help in organising

## GET JAMMING

British Library Labs will be holding a Game Jam in September to encourage coders to come up with amazing ideas for its forthcoming arcade crowdsourcing machine. The goal is to bring people together to build games that will make it fun to classify and categorise information about the one million images in the British Library Flickr collection at **flickr.com/photos/britishlibrary**.

If you are interested, email Adam Crymble at **a.crymble@herts.ac.uk**. Those taking part will need to use Phaser.io, an HTML5 game framework. Jammers are limited to a joystick and two buttons as an interface, as well as the Raspberry Pi 2, of course. "The big challenges are making something that's fun, and at the same time can gather and validate the information that's coming in," says Adam.



## WHAT IS CROWDSOURCING?

Sometimes a task needs doing that would be too costly, too unmanageable and too time-sapping if it involved hired, paid help to work on it full-time. Rather than abandon all hope, many individuals and organisations have turned to the power of the crowd. By spreading out tedious work among large groups of people, each of whom can operate whenever they have a spare moment, the idiom 'many hands make light work' becomes true. There are many examples of successful crowdsourcing, including Wikipedia and JustGiving. A branch called crowdfunding enables ideas to get off the ground by collecting small amounts of money from many individuals. This is seen to great effect on Kickstarter.

**Above left** Adam Crymble came up with the idea during a lunch break

taking food anywhere near the books in the British Library, he retired to the lobby for a sandwich. There, he noticed lots of scholars doing the same, each staring off into the distance.

"It's unproductive downtime really," he says. "I realised that no one was using the time to contribute to crowdsourcing initiatives. And yet, my fellow lunch-eaters were exactly the type of people these crowdsourcing initiatives are targeting: they're scholars, interested in contributing to the greater good, and they've got a few minutes to spare."

Adam realised that, despite the internet being awash with crowdsourcing projects, their ubiquity meant they had a tendency to be ignored during periods of inactivity. "It occurred to me that we couldn't expect people to go to the crowdsourcing project; we had to bring it to them and put it right in front of their noses in places where they're sitting around with time to kill. The idea to make it into an arcade game was really just a matter of trying to come up with a way to get people to go play with it."

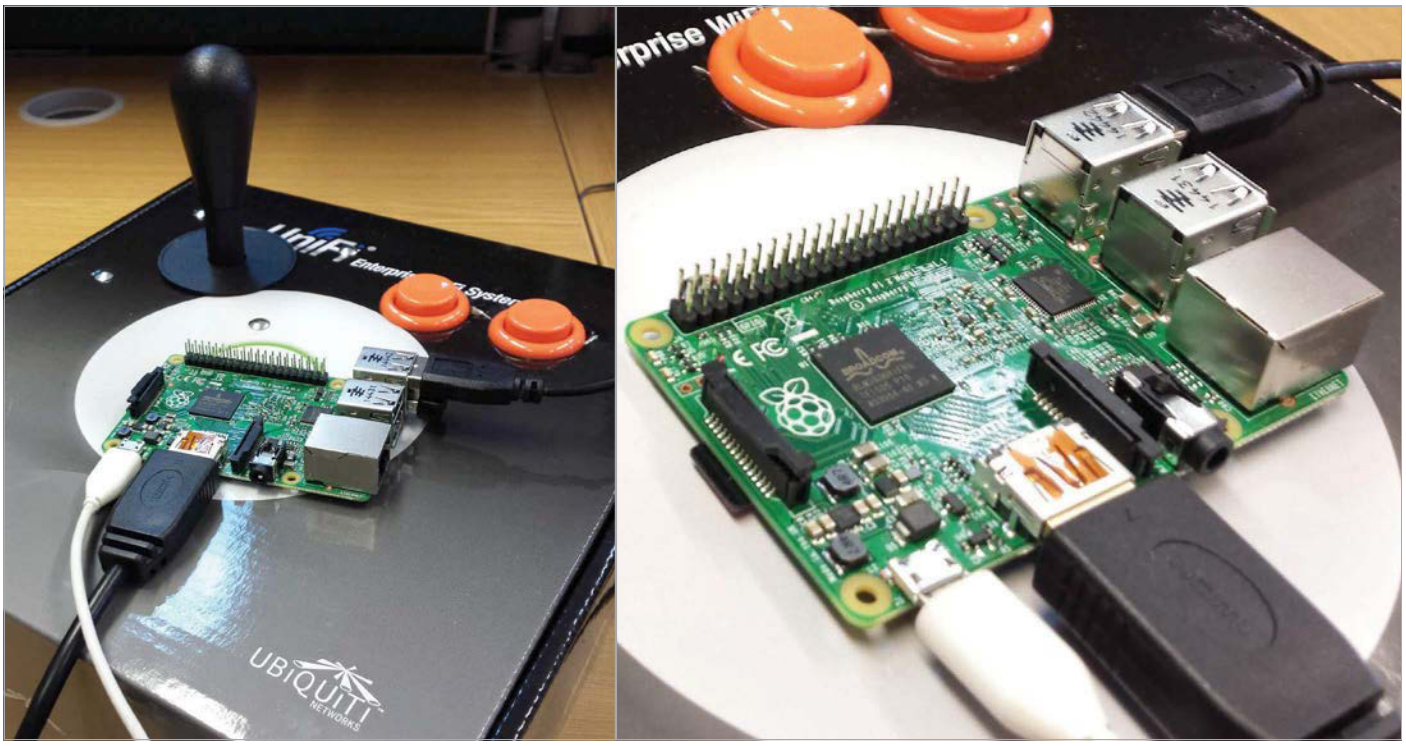
The project could save the British Library a lot of time. The images that will be organised were extracted automatically from digitised books from the 19th century, but it is difficult for researchers to know what these images are or what they represent. "It makes it incredibly difficult to find an image that's relevant to your needs," says Ben O'Steen. "It's like finding a needle in a haystack. By using these games and the help of the players to classify the material, this give us a much better idea of what is in this collection."

At the same time, the arcade machines will give the researchers a strong idea about how people play. British Library Labs will monitor how visitors use the machine, such as whether they stand and watch, or push on the glass and ignore the joystick. "We want to see whether people understand what the box is and whether they use it to play the games we put on there," explains Ben. "We're not just tracking what they end up doing, but how close they get to the machine, whether they poke at the glass, how long they stand there, and so on. We'll be doing this by using pressure and other basic sensors, certainly not by videoing or recording audio."

Still, the project is in its very early stages, with British Library Labs finalising its plans for the rest of the year. "We're dealing with the immense amount of bridge-building and bureaucracy that this will entail," says Ben. Yet they have already settled on the hardware and they have a working setup on Ben's desk: a Raspberry Pi 2 with arcade controls. "I am just working on the code to log interactions such as pressure sensors on the glass and activity in-game," he adds.

The games for the machine are being created with Phaser.io, a desktop and mobile framework which is designed for building HTML5-based games. The Raspberry Pi 2 can handle this,

> "The machines will be 1980s-style cabinets with heavy-duty buttons and joysticks, but they will have fewer cigarette stains and garish colours"



"The Raspberry Pi means we can give clear instructions on how others can put together a copy of what we've created, to extend or improve upon it…"

**Above** The arcade components have been fitted to a large box for the time being and connected to a keyboard encoder (ipac2). It can be connected via USB to any computer

allowing the project to come in at a reasonable cost. "We needed something that could do the job and that didn't cost the earth," says Adam Crymble. "We're also hoping to encourage others to contribute to the project, so it was important to us that we could use hardware that others could feasibly afford so that they could replicate our setup. That made the Raspberry Pi 2 a natural choice."

## HOW CAN ORGANISING DATA BECOME A GAME?

"Games are fundamentally a series of decisions someone has to make that if done well, with a bit of luck thrown in, will take them towards their goal," says Adam Crymble. "You jump on that, go in there, avoid that thing, and then you save the princess. Crowdsourcing is the same: a series of decisions. They seem like natural bedfellows to me. We'll either take the fun out of games, or we'll take the tedium out of crowdsourcing."

Using an alternative to the Pi – perhaps an old computer – would, Adam tells us, have made for a "clunkier and probably less reliable" machine. "The Raspberry Pi means we can give clear instructions on how others can put together a copy of what we've created, to extend or improve upon it," he continues. "It also completely blows the minds of my fellow academics when I tell them we're building an arcade machine with a £30 computer that fits in their pocket. The wood we'll need to build the cabinet will be considerably more expensive."

Although the British Library Labs team will be making a number of prototype crowdsourcing games itself, Ben O'Steen is keen to open things up. A Game Jam is being planned for the second week of September, which will allow people to contribute or work on games that can run on the arcade machines. "It should be in the building at the same time as the Alice in Wonderland exhibition," discloses Ben, "so there is scope to have games with that theme as well."

By using games to encourage people to participate in crowd-generated data collection, the aim is to tap into energy that is currently reserved for play, and so enable new knowledge to be built. Bringing together physical computing and historical research excites Adam Crymble, who cites it as a rare combination. "There is so much potential once we start thinking about what our needs are and how computers like Raspberry Pi can help us achieve them," he says. "We don't have to wait for other people to build the things we'd like to have. It's so levelling. As someone who spent much of my youth playing video games, I'm excited about the possibility of building my own games and putting those to serious use."

DEXTER INDUSTRIES

SAVE 15% "MagPi15" discount code

## BrickPi

Build a LEGO robot with your Raspberry Pi!

## GoPiGo

Everything you need to build a Raspberry Pi robot!

## GrovePi

Connect hundreds of sensors to your Raspberry Pi!

www.dexterindustries.com

# LIVERPOOL MAKEFEST

**Above** The hard-working team behind Liverpool MakeFest: which is the odd one out?

**Below** Liverpool Central Library

The first Liverpool MakeFest enthused more than a thousand visitors with Daleks, textiles, Minecraft, a host of robots – and musical fruit and veg

The Liverpool MakeFest banner outside the recently refurbished Liverpool Central Library was a clue that something more than books could be found inside on this day. Stepping through the neoclassical facade, visitors stopped in their tracks as a very active Dalek moved towards them. Adding to the mild sense of alarm was the Dalek voice, from effects company Sciphonix and its voice modulator, based on the original Dalek analogue ring modulator used by the BBC in the 1960s and 1970s. Next to that, under the full-height elliptical atrium topped by a glass dome, were the Doctor Who monster masks of Ray Phillips.

If that subconscious frisson of remembered childhood fear stoppped you staying while your children tried for a Dalek selfie, before you continued further into the building you could take up a Nerf gun to try to beat the scores gained by DoES members' ping-pong ball firing machines. The maker community in Liverpool and its environs is huge: at the heart of this is DoES (Do Extraordinary Stuff), a makerspace and co-working place run as a social enterprise to support the creative endeavours of local makers. Amid the displayed projects, such as a laser-cut pinhole camera, was DoES Tower. This scale model of the St John Tower changed colour according to Twitter requests monitored by its Raspberry Pi board, provided that they were HTML5 shades, and tweeted back pictures of the colour choice.

Step further inside and there were makers distributed around Liverpool Central Library, displaying and demonstrating everything from robot arms and ROVs to jewellery-making and woodcutting, and running hands-on sessions with Raspberry Pis, clay, and sewing machines – although not necessarily ones which combined all three!

## Circle of cool

In a multilevel circular space, beneath the historic Victorian Picton Reading Room, was the "Cool Kit" area. Here, despite the intense heat as the library's air conditioning struggled to cope with the crowds, people were drawn down to the displays of games, robotics, and some very hands-on technology. This included Laura Pullig's Tactile Electronics, where thermochromic pigments that change colour with heat were controlled with electronic circuits. Here you could control the servos of the MeArm, which garnered a five-star review in *The MagPi* #34, manually: it's something of a challenge, so it's good to know that it's also programmable in Scratch.

> Children tried for a Dalek selfie before taking up a Nerf gun to beat the ping-pong ball firing machines

CodeBug, the cute and easily programmable light-up wearable, features buttons, 25 LEDs, USB connection and, where you'd expect to find its legs, half a dozen connectors for crocodile clips. It's designed for entry-level programming in physical computing, with quick progression from triggering the LEDs to controlling attached devices. The North West's hackspaces and makerspaces were well represented, with Tamarisk from HacMan, Manchester's hackspace, helping people to build swarms of small, cheap, and strangely charming bugs with light-up eyes. Funky Aardvaark, who run Chester's Area 51 hackspace, brought along sumo robots, but also generated interest with freshly made board games, laser-cut by hackspace members.

A number of educational stands linked the joyful creativity of the maker community with a practical career in STEM (science, technology, engineering, and mathematics) subjects, via local colleges which are embracing the maker movement. Particularly close to this movement is Liverpool John Moores University (LJMU) – once a polytechnic, and still retaining the industry-led practicality of that form of tertiary education – which drew crowds with its strawberry drum kit, banana keyboard, and other Makey Makey orchestra vegetable-based fun. The Makey board was also to be seen controlling *Minecraft* on a Raspberry Pi.

In addition, LJMU was drawing many to its GameJam, and to the 3D scanning and printing from its FabLab, where the traditional hourglass of waiting during these 3D operations was replaced with a countdown of "magic happening". 3D printing and robotic arms were also part of Edge Hill University's contribution, along with involvement in Raspberry Pi and *Minecraft* workshops, and inventors' workshops with cheap components.

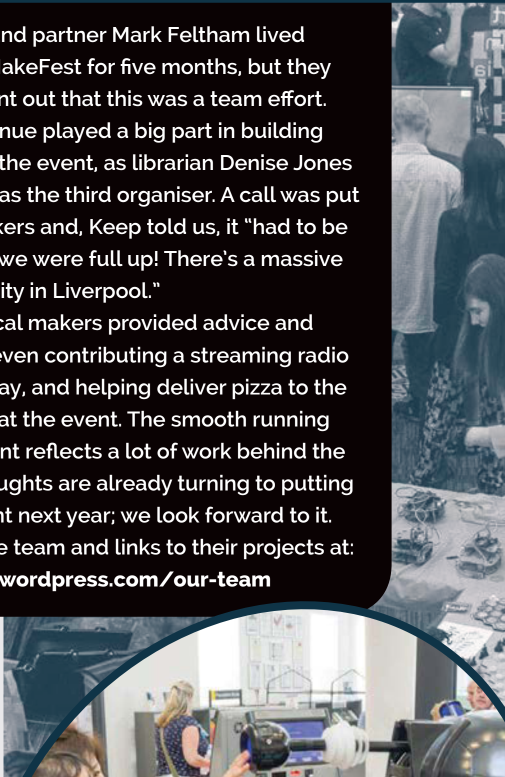
Students from Future Tech Studio in Warrington, a college for 14-19 year olds, showed many robots they had built and coded, as well as helping with the organisation of the event, having been drafted in by their teacher, MakeFest organiser Caroline Keep. Also attracting pupils in that age range is The Studio School in Liverpool, specialising in gaming and digital technologies, which brought along a Linux-based NAO Bot, from Aldebaran Robotics: "a companion, assistant and research platform" which charmed everyone with its dance moves.

## ORGANISING A MAKEFEST

Caroline Keep and partner Mark Feltham lived and breathed MakeFest for five months, but they are quick to point out that this was a team effort. Securing the venue played a big part in building momentum for the event, as librarian Denise Jones came on board as the third organiser. A call was put out to local makers and, Keep told us, it "had to be closed early as we were full up! There's a massive maker community in Liverpool."

A board of local makers provided advice and practical help, even contributing a streaming radio station on the day, and helping deliver pizza to the hungry makers at the event. The smooth running of the day's event reflects a lot of work behind the scenes, but thoughts are already turning to putting on another event next year; we look forward to it.

See the whole team and links to their projects at: **lpoolmakefest.wordpress.com/our-team**



**Right** If there's one thing a Dalek cannot do, it's inconspicuously blend in…even among the wonders of a MakeFest

**Below** A dancing robot at The Studio: every school should have one!

The crafters area was also home to this very cool Adana printing press

## MAKERS ALL

Spread out among the bookshelves of the ground floor, either side of the route to the workshops, were crafters working with older technologies ranging from clay dragons (make and paint your own), via woodworking, to sewing with recycled materials, as old T-shirts were transformed into bags. Fabric-inspired tech was on display elsewhere, as Gemma Latham's Minecrafting with Textiles featured a punchcard reader – like those used with Jacquard looms, which morphed into program storage for early computers – for transferring textile patterns into *Minecraft* builds. Try the work-in-progress Python code, developed with David Whale, on your Pi: **github.com/whaleygeek/punchcard_reader**

As well as putting the electronic and computing activities in a broader context of centuries of makers (in other words, geeks aren't on their own), the crafters enabled visitors to participate with traditional skills they already had – or to learn those they lacked – and added in more craft activities for a generation who will grow up seeing making a Pi-based robot as something in the same category as baking a pie.

## Bots and cyberpunks

One corner of the room was set aside for "the bots", including the open-source 3D-printed Cannybots, set up here in a Scalextric formation, but programmable from your Pi (or other boards) to solve maze challenges or take part in a jousting tournament. Ohbot is a face, a creative robot system that is modelled on the children that will use it in primary school classrooms. The kit is easy to assemble, can be programmed in Scratch, and comes with a range of learning-through-making projects that teach programming in a fun way, but are compatible with the new UK schools' Curriculum for Computing. BigLesP brought along several wonderful Linux bots to inspire and amaze.

Colourful characters were found throughout the event: the maker community may have distinctive personalities, but here we're talking about fancy dress, as attendees were invited to dress up in cyberpunk or other costumes, as you can see from the pictures. Our favourite was the girl in the Raspberry Pi logo dress who was with the Edge Hill University's PGCE course, "Geek Squad".

Liverpool Libraries was on hand with its business advice service, as was Liverpool SOUP, an organisation that brings together new businesses to give pitches for their ideas. The best of these will receive a few hundred pounds to help them on their way. Every inventor needs to have access to a laser cutter nowadays, and demonstrating the devices used in many makerspaces was Dominic Morrow of NottingHack, with his new company Just Add Sharks and its line-up of impressively powerful cutters. ScraperWiki was also there to talk open data, and Liverpool Girl Geeks were putting everyone in the frame with a frame to be photographed in, and promoting their excellent coding workshops: the latest details can be found **@lpoolgirlgeeks**.

"LJMU drew crowds with its strawberry drum kit, banana keyboard, and other Makey Makey orchestra vegetable-based fun..."

One of the scary masks!

## Down to business

Up on the first floor, a salmagundi of maker businesses and services included Ironbird and its "remotely operated aircraft and the stabilised camera technology", or drones as they're inevitably called. RepublicIoT [see "Internet of Things That Spy On You" boxout] demonstrated its low-cost shields for connecting your sensor device to the internet. At the next table, local Android app company Novoda showed the early result of its move into connected devices, as well as demonstrating Google Cardboard, and calling for ideas for new devices, with a Nexus 9 as a prize for the team's favourite.

The optimism of the maker movement was reflected in conversations at the MakeFest, which ranged from Hyperloop transport to teaching of code coming back to UK schools. The revamped Central Library also drew a lot of praise, with visitors enjoying

## MQTT

Dig a little, and you'll find MQTT mentioned as an essential part of many Liverpool MakeFest projects, from 4th Platform to the DoES Tower. Message Queuing Telemetry Transport is a machine-to-machine (M2M) connectivity protocol for the Internet of Things. It's lightweight and efficient enough for use in sensors communicating with one or many receivers via anything from satellite link, to mobile or even dial-up connections.

All images – J R Peterson for Liverpool MakeFest, https://lpoolmakefest.wordpress.com

the roof terrace and browsing the great collection of books and sheet music; this is not an old-fashioned keep-quiet-and-no-food library, though, as there's a cafe that lets you take your coffee with you as you browse the bookshelves.

### Plenty to make

A mini-classroom at the back of the library, reached via the crafters (see "Makers all" boxout), became a Raspberry Jam room for the day, with introductory Pi sessions and plenty of *Minecraft Pi*, while more physical workshops took place on the top floor, where makers were also drawn out onto the roof terrace for views across to North Wales.

Here the emphasis was hands-on: sewing your own signal-blocking bag for your mobile telephone, for example, and learning to podcast. Dark Water Exploration, a Liverpool-based foundation "taking open-source technology to the depths", was helping to make Lego ROVs (remotely operated vehicles), "a guerilla attempt to get people to do underwater robotics" as they put it, and even had a water tank for you to test your ROV in. Aimed at the young, it was very popular with all ages, and prizes were awarded for best designs, as well as fastest ROV in recovering pirate treasure in a fish tank. If you want to make your own, the custom control board design is online, or you can buy it from **darkwater.io**: just add Lego bricks.

Equally popular was the workshop from the Raspberry Pi Foundation's creative producer Rachel Rayns and education developer advocate Ben Nuttall, working with Liverpool's FACT (Foundation for Art and Creative Technology) on Dots boards, a Pi HAT which replaces soldering (not always classroom friendly) with dot-to-dot by conductive paint for an introduction to programming and electronics. "This is our first third-party Dots board," Rachel told us; the challenge is "to make meaningful hardware and software for the very young: those aged four and up." They're very low-cost (a Pi is the significant expense) so within reach of almost all, and they have the potential for "different colours and designs", to produce something like trading cards.

The next day, Rachel was up the hill at FACT Gallery, running a course on Robot Greenhouses with the Pi. Further maker activities continue all summer at FACT (**fact.co.uk**) if you're in the area. If not, get ready for a journey: we've a strong feeling that next year will bring an equally splendid follow-on MakeFest in Liverpool. We'll then have a chance to make it up to any of the great projects and makers we've accidentally missed from this report.

## INTERNET OF THINGS THAT SPY ON YOU

### A QUICK LOOK AT THE OPTIONS FOR CONNECTING YOUR BOARD AND ITS SENSORS

A couple of years ago, we heard DoES's Adrian McEwen speaking to the Open Source Hardware User Group about the limitations of connecting your sensors and devices, as the platforms available were all proprietary, with necessary limits to the sustainability of the companies. This was a theme taken up at last year's IndieTech conference as acquisitions by internet giants were discussed in the context of those companies which werer bought up, either leaving users without a service, or finding that the data privacy clauses in those services were suddenly revised.

Enter RepublicOfThings.com with its 4th Platform – "open, massively scalable and flexible, fit and agile", which, Republic's Andrew Bechener tells us, "aims to bring European social mores around privacy and data to the Internet of Things". It also significantly reduces costs involved in connecting up your sensor device or board, with a £3 shield (a fraction of a Zigbee's cost) and free and open-source software for the mesh network stack and encryption, which should run on any Raspberry Pi from the A+ upwards.

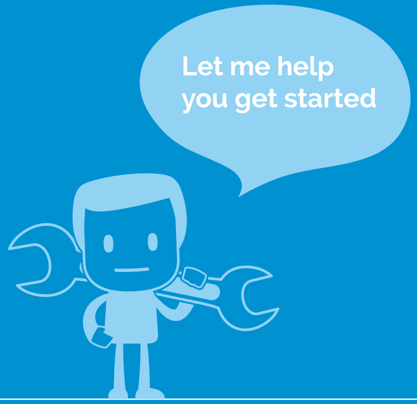
The next target is certification. Meanwhile, just a few tables away, Patrick John McGee was proposing an end to centralised databases in a Web of Things, since embedded data could instead be carried within images. For more details, see **#MOBWOT** on **slideshare.net/PatrickJohnMcGee**

Raspberry Pi

# GET STARTED WITH RASPBERRY PI

Learn everything you need to know to become pro with your Pi

1 The ingredients you need
2 Set up your Raspberry Pi
3 A tour of Raspbian
4 Installing & updating software
5 Use the GPIO pins



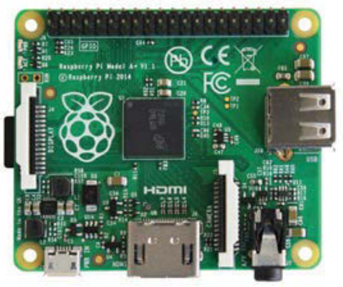
Congratulations! You've got yourself a brand new Raspberry Pi and you're ready to start using it for learning code, creating amazing projects, or just simply to power a home theatre. While the Raspberry Pi is generally very easy to use once you know how, it's that initial learning experience that can be a bit tricky for some.

Have no fear, though: we've put together the ultimate guide to getting started with your very own Raspberry Pi, from learning what all the ports and pins on your Raspberry Pi are for, to actually getting it up and running with your own monitor, mouse, and keyboard.

Whatever you want to use your Raspberry Pi for, you need to start here with the basics.

## The little Model A+

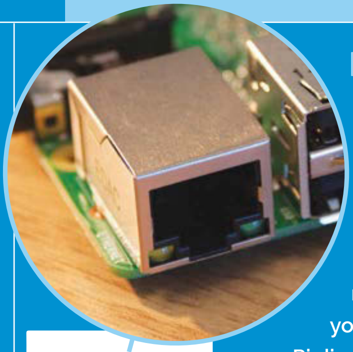
As well as the 'standard' Raspberry Pi, the Raspberry Pi 2 Model B, there's also a smaller version of the Raspberry Pi that you can use. The Raspberry Pi Model A+ is a cut-down version of the original Raspberry Pi, with a little less power at its disposal and fewer connections on it. It's favoured by people who like to make big physical projects, due to its diminutive size and low power requirements. It also has only one USB port and no Ethernet port, making it slightly less useful to some.



# THE INGREDIENTS FOR A RASPBERRY PI 2

## USB ports

The Raspberry Pi 2 has four USB ports, allowing you connect it to keyboards, mice, WiFi dongles, and USB sticks containing all your files. Since the ports don't provide much power, if you want to add a USB hub to the Pi you'll need to find one that comes with an external power supply.

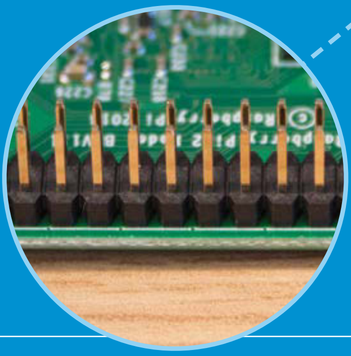


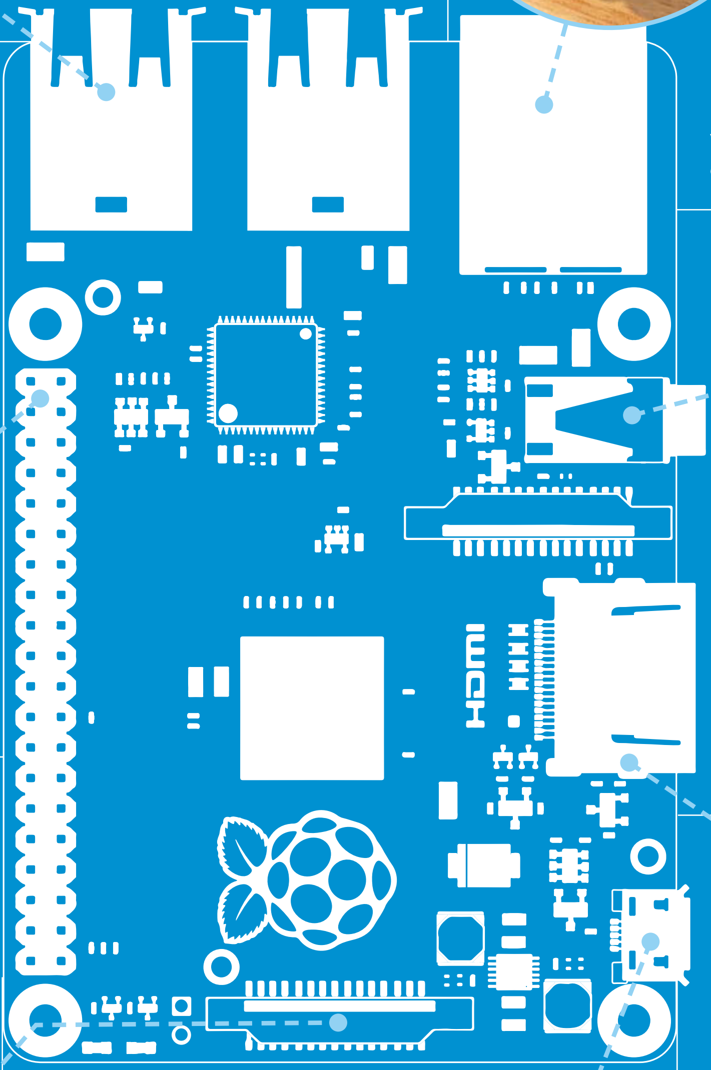
## Ethernet port

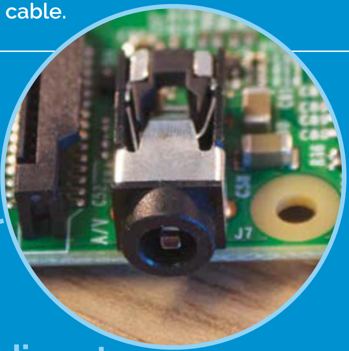
The traditional way to connect to the internet is via a wire called an Ethernet cable. You'll find a few similar ports like this at the rear of your router at home that will let you connect the Raspberry Pi directly into it. This method is easier to set up than WiFi and may provide faster internet, but you're then limited by the length of the cable.

## GPIO header

This comprises the general-purpose input/output (GPIO) pins. They're a set of connections that have various functions, but their main one is to allow you to connect to the Raspberry Pi with an electronic circuit. You can then program the Pi to control the circuit and do some amazing things with it.





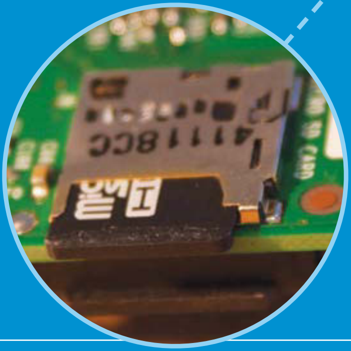


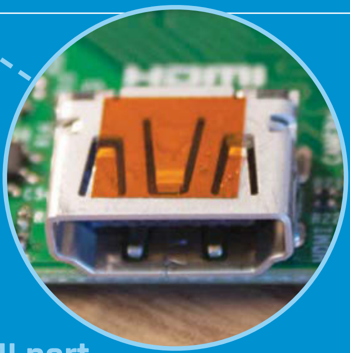
## Audio out

This looks like a headphone socket because that's exactly what it is. A 3.5mm jack to be precise, this allows you to connect the Pi to computer speakers, or you could even plug in your favourite headphones and have a Raspberry jam.

## MicroSD card slot

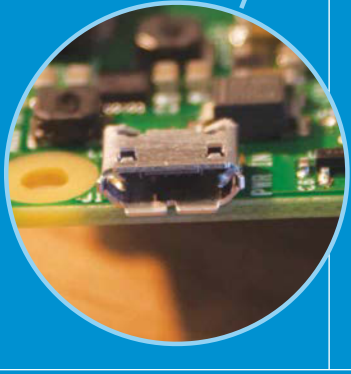
A little SD card is used as the Raspberry Pi's hard drive. This is where the operating system will live once you've put it on there. Most computers won't be able to directly connect to a microSD card, but you can get an adaptor that plugs into normal SD card slots.





## Power

This is the kind of small charging port you might find in your smartphone. This micro-USB port means you can power the Pi with the right kind of mobile phone charger or directly from your PC – however, it's best to use the official Raspberry Pi power supply to make sure the Pi is getting enough power.

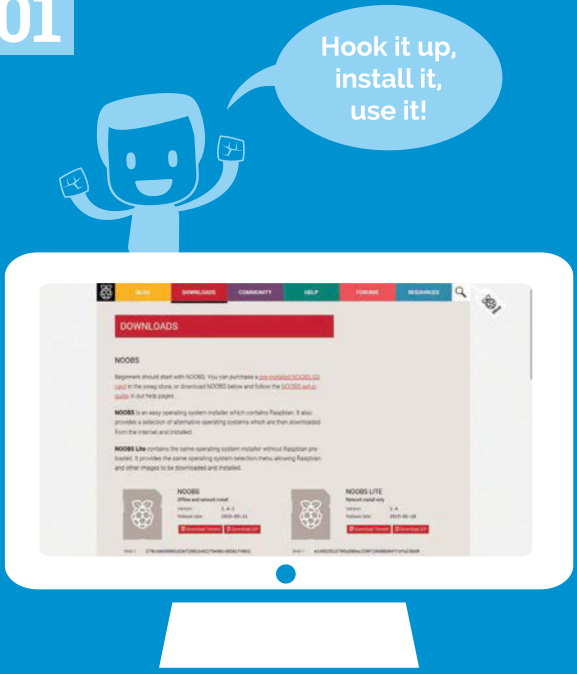


## HDMI port

This is an HDMI port, the kind you'll find on the back of most modern TVs and computer monitors. Use a standard HDMI cable to connect your Raspberry Pi to your chosen screen, to see (and hear) whatever it's doing. You'll definitely need to plug it in to set up the Pi.

## SET UP YOUR RASPBERRY PI

### 01 DOWNLOAD NOOBS



The Raspberry Pi comes with many operating systems you can use, which you could manually install yourself if you wish. There's a much easier way to install these OSes, though, and that's via the New Out Of Box Software, or NOOBS. It holds all the latest versions of the Raspberry Pi operating systems and you can grab it from the download page at: **raspberrypi.org/downloads**

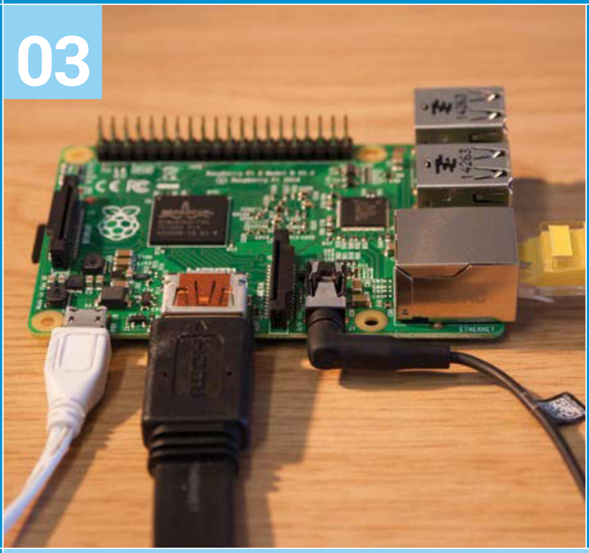
We prefer to use the full version of NOOBS, as it comes with Raspbian already downloaded, making the process slightly faster than with NOOBS Lite. However, all the other operating systems will be downloaded as they install, on both versions of NOOBS.

### 02 INSTALL SD CARD

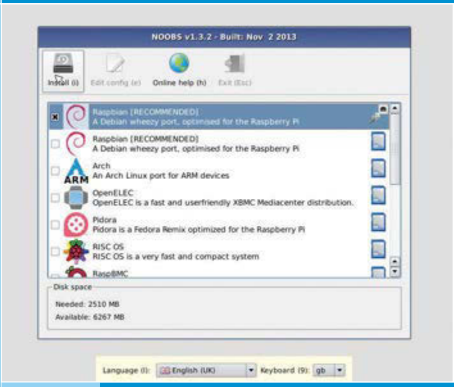
While that's downloading, you'll need to get your SD card ready to work on your Raspberry Pi. This will require you to format it, so if there are any files on the card you want to keep, now's the time to take them off. You'll need to install the SD Card Formatter 4.0 tool to prepare the card, which can be downloaded from here: **bit.ly/1alC3Wp**

Once you've formatted your SD card, extract the files from the NOOBS ZIP folder and put them all on the card. That's it: NOOBS is installed to your SD card and ready to use!

### 03 CONNECT THE CABLES



Take the SD card adaptor out, retrieve the microSD card, and slot it into the Raspberry Pi; this is very important, as the Raspberry Pi won't be able to turn on properly otherwise. To start with, you'll need to plug in an HDMI cable between the Raspberry Pi and your screen, an Ethernet cable for your router (or a USB WiFi dongle), along with a mouse and keyboard. Finally, when everything you need is plugged in, you can attach the power cable to the Raspberry Pi.



NOOBS v1.3.2 - Built: Nov 2 2013

Install (i) | Edit config (e) | Online help (h) | Exit (Esc)

- [✓] Raspbian [RECOMMENDED] – A Debian wheezy port, optimised for the Raspberry Pi
- Raspbian [RECOMMENDED] – A Debian wheezy port, optimised for the Raspberry Pi
- Arch – An Arch Linux port for ARM devices
- OpenELEC – OpenELEC is a fast and userfriendly XBMC Mediacenter distribution.
- Pidora – Pidora is a Fedora Remix optimized for the Raspberry Pi
- RISC OS – RISC OS is a very fast and compact system
- RaspBMC

Disk space
Needed: 2510 MB
Available: 6267 MB

Language (l): English (UK) Keyboard (9): gb

### 04 INSTALL RASPBIAN

The Raspberry Pi will turn on and display some text on the screen – you can ignore this until it gets to a menu which lists all the available operating systems. It allows you to select multiple OSes at once, but right now we just want to use the one that's called Raspbian. This is the main operating system for the Raspberry Pi, with all the official apps, software, and learning documents. Upon selecting Raspbian, click on Install and it'll begin the Raspbian installation process, which you can see on the following page.

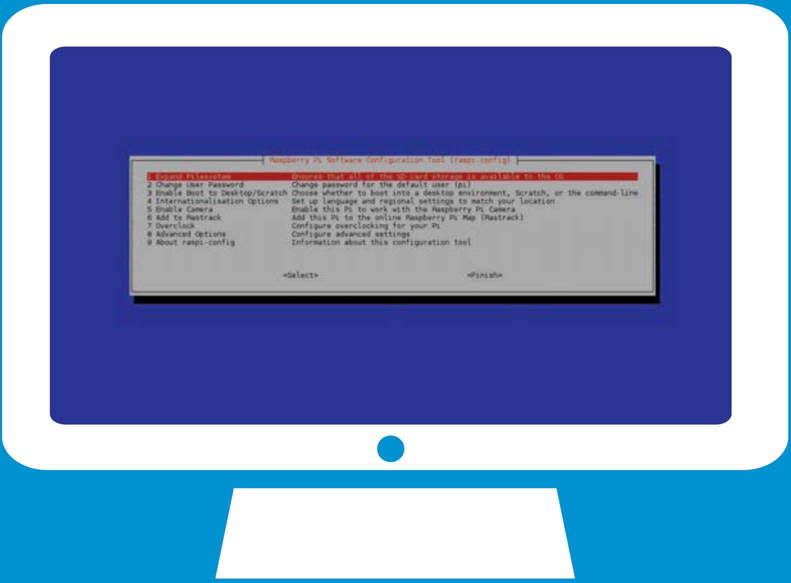
### Alternative operating systems

**PIDORA**
A bit like Raspbian, but based on a different core operating system. This is something people a bit techy can use for a slightly different Raspberry Pi.

**ARCH**
A very basic operating system that works entirely from a command line, no mouse and keyboard required. You'll really need to know your computers to start with this.

**OPENELEC**
An OS to turn the Raspberry Pi into a home theatre PC, complete with the Kodi software that plays music and videos as well as web video.

**RISC OS**
A throwback to your school days, RISC OS is what used to be on old Acorn computers. The Raspberry Pi is in many ways derived from them.



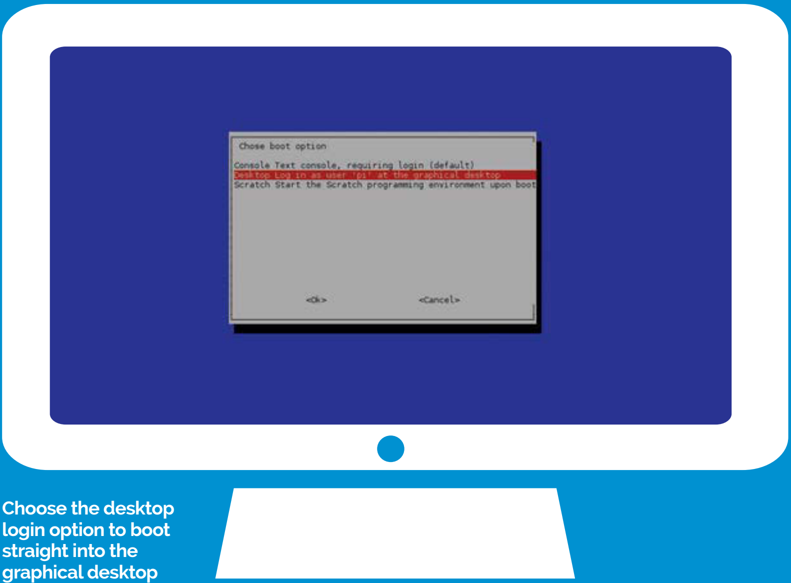
```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem              Ensures that all of the SD card storage is available to the OS
2 Change User Password           Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch Choose whether to boot into a desktop environment, Scratch, or the command-line
4 Internationalisation Options   Set up language and regional settings to match your location
5 Enable Camera                  Enable this Pi to work with the Raspberry Pi Camera
6 Add to Rastrack                Add this Pi to the online Raspberry Pi Map (Rastrack)
7 Overclock                      Configure overclocking for your Pi
8 Advanced Options               Configure advanced settings
9 About raspi-config             Information about this configuration tool

                  <Select>                                   <Finish>
```

## SOFTWARE CONFIGURATION TOOL

The blue configuration screen will pop up to let you make some changes to Raspbian before you start using it. There are some important tweaks to make here so that you have the best experience with Raspbian.

First of all, make sure to use the 'Expand filesystem' option. Raspbian doesn't take up much space to begin with, so you need to tell it to use your entire SD card if you want to use all your free space. After that, go to the 'Boot to desktop' option and select the option for a graphical desktop so that you're not stuck in a command line. You can have a look at the other settings, but you only require those two to make sure Raspbian is in top shape for you.

Now select 'Finish'; the Raspberry Pi will restart and load up into the Raspbian desktop, ready to use. If you ever need to access that blue screen again without reinstalling, open up a terminal command prompt and type **`sudo raspi-config`**.



```
 Chose boot option

Console Text console, requiring login (default)
Desktop Log in as user 'pi' at the graphical desktop
Scratch Start the Scratch programming environment upon boot

          <Ok>                  <Cancel>
```

Choose the desktop login option to boot straight into the graphical desktop

### [1] EXPAND FILESYSTEM

Raspbian only uses so much of your SD card by default. You can use this option so it makes use of the entire space on the SD card, allowing you to put more on your Raspberry Pi.

### [3] BOOT TO DESKTOP

By default, Raspbian will turn on and go to a command line rather than a normal graphical interface that you can use a mouse and keyboard on. This option allows you to go straight to the desktop instead.

### [5] ENABLE CAMERA

You can get a little camera for the Raspberry Pi that plugs directly into the Pi itself – if you want to use it, you'll have to turn it on here.

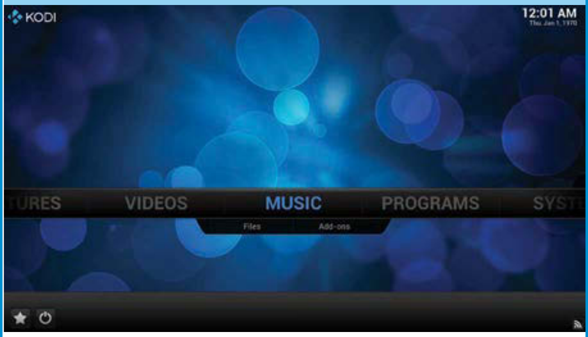
### [7] OVERCLOCK

This is where you can make the Raspberry Pi run a bit faster by giving it more power. Only do this if you really need to, though, since the Pi has plenty of power to start with.

### [8] ADVANCED OPTIONS

Here you can update this menu, change the way it displays on your TV, mess around with the memory, and alter many other things you shouldn't really touch unless you know what you're doing.

## Setting up a media centre



We touched on OpenELEC on the previous page, and how it can be used to make a home theatre PC. This is a PC that hooks up to your TV and powers all your media needs. You can find OpenELEC on NOOBS, and installing it is very similar to Raspbian: you select OpenELEC and hit Install!

OpenELEC runs on Kodi – software that lets you connect to your other computers over the network, as well as some online web services such as YouTube. It can play just about anything, but you need to show it where the files are. When adding folder locations to either Video or Music, you can find any shared folders via the SMB option, or you can simply plug in a USB hard drive full of videos and music and play them straight from the menus.

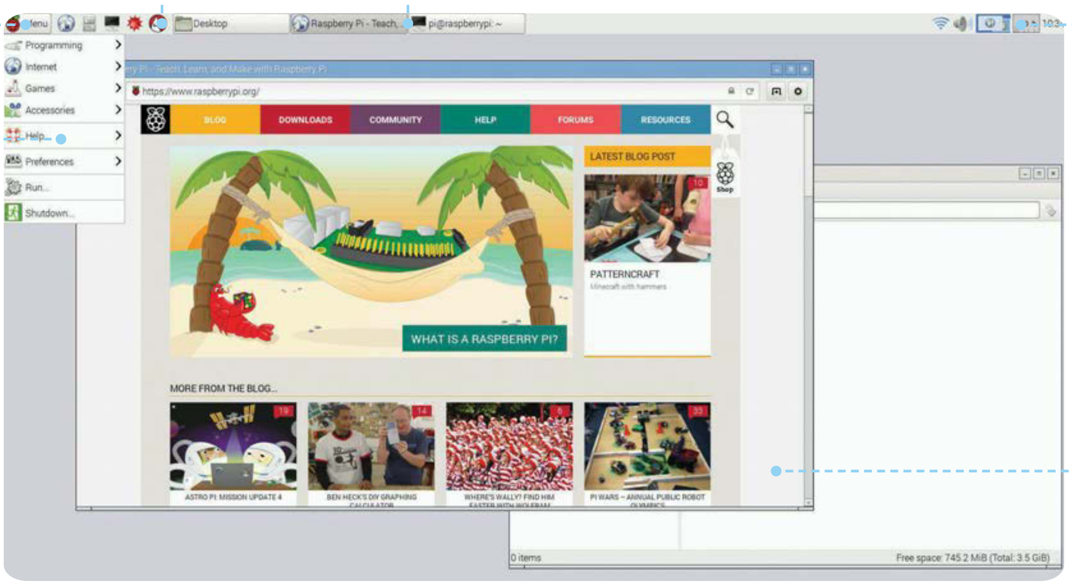
## A TOUR OF RASPBIAN

Access the various menus, programs and settings for Raspbian; almost everything you do will start here

These icons let you quickly launch certain programs, such as the browser, the terminal, and the Mathematica programs for hardcore maths and graphing

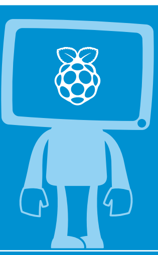
The various open windows are listed on here; much like in other operating systems, you can click between them when you need to change location

This area offers quick access to tools such as WiFi to ensure your Raspberry Pi is running just fine



Programs and apps are categorised to make them easier to find – if you can't find the app you're looking for, you might need to go through all of them

Raspbian should feel familiar to most PC users



Windows here work just like any other kind of operating system: you can drag them, change their shape, and close them using your mouse

### Top right icons

**WIRELESS INTERNET**
This shows the state of your internet connection; solid blue lines means it's connected!

**VOLUME**
Control the volume of your Raspberry Pi from here. This will work whether you're getting sound from the HDMI port or via headphones.

**PROCESSING POWER**
The Raspberry Pi, while small, has a lot of processing power. This tells you how much is in use, so if it's running a little slow and this gauge is at 100%, you'll know why.

**TIME**
Set the time to be anywhere in the world! The Raspberry Pi relies on the internet to tell it what time it is.

Raspbian looks and works very similarly to the kind of operating systems you're used to, except that the menu is now at top of the screen! Raspbian is based on something called Debian, which is a version of Linux, a highly customisable operating system that can be tweaked enough to run on the Raspberry Pi. It works extremely well, and even on the tiny Raspberry Pi it will almost feel like using a normal computer!

There are a few important icons on the top panel that you should make sure you're aware of.

The Menu is where all the programs and apps live; just like in any other operating system, you can access them from here and they'll open up in a new window. You'll find all the settings in here as well, in case you want to tweak the way Raspbian looks and works.

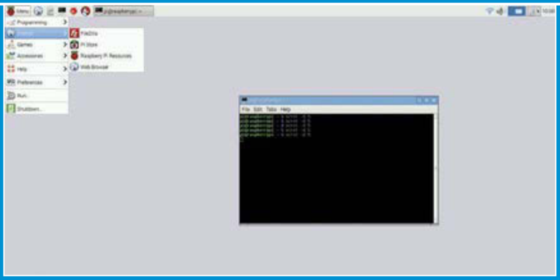
Next to the menu is a row of quick-start icons to quickly launch software. The globe picture is the Raspberry Pi browser, your access point to the internet. The cabinet represents the file system of Raspbian, allowing you to browse any documents or images you have saved onto your Raspberry Pi. The picture of the screen is the terminal, and it's what you use to run commands via text on the system,

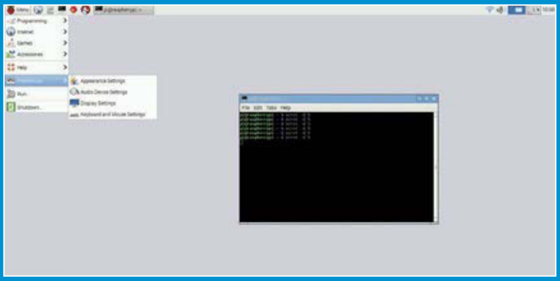
"The cabinet represents the file system of Raspbian" something you might have to do for more advanced projects.

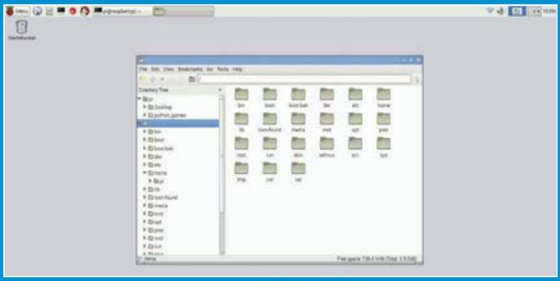
The file system of Raspbian is a little different from something like Windows. Instead of having a C:/ drive with a My Documents folder and programs kept in Program Files, everything is spread out in multiple folders on the root, or top of, the file system. What could be considered My Documents is a folder called Pi in the Home folder on the root. You may see it referred to as the 'home directory', and that's why.

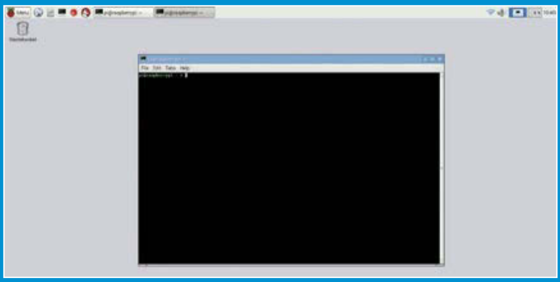
To turn off the Raspberry Pi when you're done for the day, you go to the Menu and press Shutdown. This makes sure everything is safely disconnected and turned off before the entire computer turns itself off. As the Raspberry Pi doesn't have a power switch, you'll have to manually unplug the Pi to fully turn it off standby, and you'll have to unplug it and plug it back in to turn it on again.

Raspbian is quite a simple interface, then, very similar to how you may have used computers in the past. You're now ready to start learning how to code and create your own excellent projects!


Look through the menu categories to find the software you want to use right now


Settings and preferences allow you to modify the look and feel of Raspbian

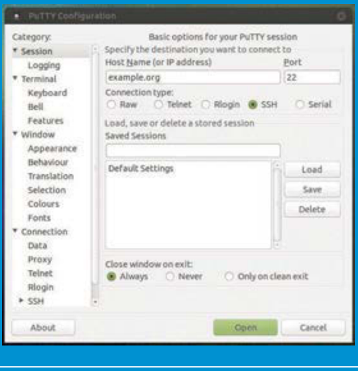

Browse files and folders on the Raspberry Pi, and use the same drag, drop, copy, and paste functions of other operating systems


Access the command line and control the Raspberry Pi with text commands. Almost like a hacker… almost

## REMOTELY CONNECT TO YOUR RASPBERRY PI

The Raspberry Pi is extremely flexible due to its design, and because of this it will let you connect to it from another computer via a system called SSH (Secure Shell). All you need to connect to it from another PC is the IP address of the Raspberry Pi and a way to access SSH. For the latter you can get PuTTY, a piece of software specifically made to let you easily connect to another system via SSH.

To find out the IP address of the Raspberry Pi, you simply open up a terminal window and type in **`ifconfig`**. It will list all the details of your network connections, including the IP address; this is listed as 'inetaddress'and may look something like 192.168.0.20. To connect to it, you need to use 'pi@ 192.168.0.20', give the password of 'raspberry', and then you can control your Pi from the command line.


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): example.org  Port: 22

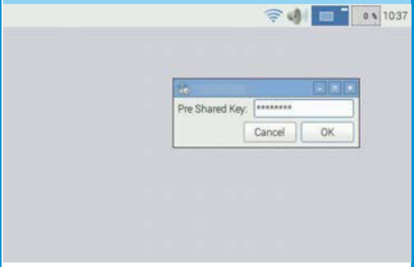
Connection type: Raw  Telnet  Rlogin  [✓ SSH]  Serial

Load, save or delete a stored session

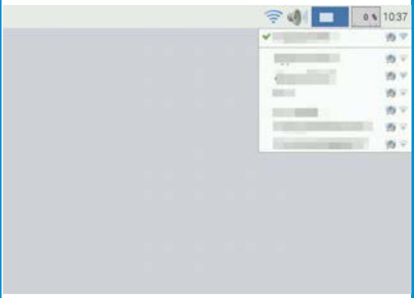
Saved Sessions: Default Settings

Close window on exit: [✓ Always]  Never  Only on clean exit

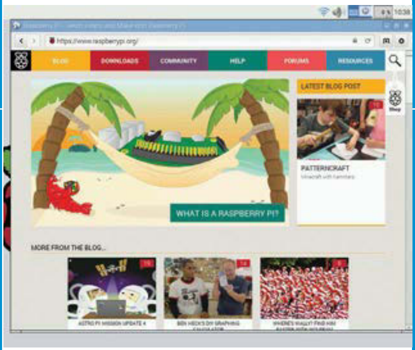
## Connect to WiFi



In the right corner of the top panel, you'll find access to WiFi. If you have a compatible WiFi dongle for the Pi, clicking on this will drop down a menu that shows you all the available wireless networks you can connect to.



Click on the wireless network you want to use and you'll be shown a box that lets you put in your password. It will actually display what you're typing in, which should make it easier to type, but make sure your neighbour isn't peering through the window trying to copy it down!



The Raspberry Pi should now be connected to the internet! It will automatically set all its options from your router that will let it talk online. Open the browser and go to your favourite website to make sure it works. The Pi will remember your wireless details and connect whenever it's on.

## INSTALLING AND UPDATING SOFTWARE

Expand and maintain Raspbian for a long-lasting Raspberry Pi experience

### INSTALL NEW SOFTWARE

You're not limited to the software that's just on Raspbian when you install it. Raspbian has access to thousands of different programs that you can download and install, just as you would with smartphone apps. Raspbian doesn't have an app store, though, so you need to install them using the terminal.

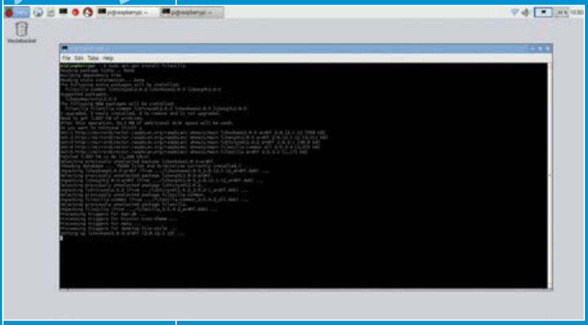
This does require you to already know what the software is called to install it in the terminal, since you can't browse the software in the same way as on your phone. If you're not sure of the exact name of the software you want, you may have to Google it. Otherwise, if you're looking for a specific kind of app, you can use a command like the following to search for it:

```
$ apt-cache search ftp
```

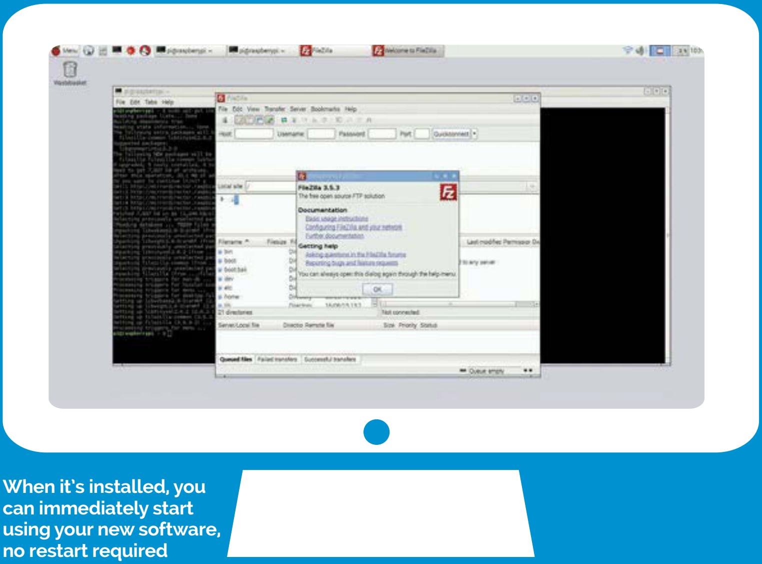
It will return a list of packages and their details. The package name is how you install the software; in our case, FileZilla comes back as an FTP client. Its package name is 'filezilla'. So, to install it, we use:

```
$ sudo apt-get install
filezilla
```

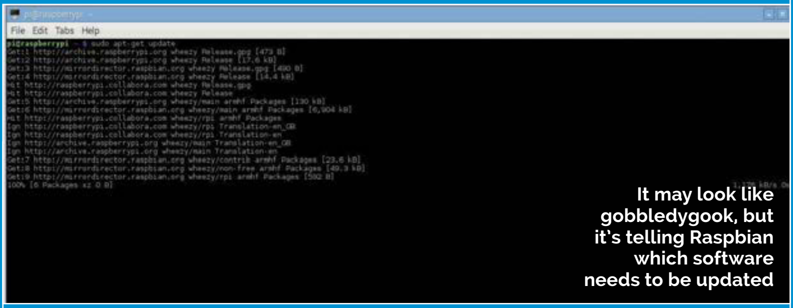
This will download the package and any other necessary software it needs to run, and install it to Raspbian.



Raspbian asks you to agree to an installation with a simple press of Y



When it's installed, you can immediately start using your new software, no restart required



```
pi@raspberrypi ~ $ sudo apt-get update
Get:1 http://archive.raspberrypi.org wheezy Release.gpg [473 B]
Get:2 http://archive.raspberrypi.org wheezy Release [17.6 kB]
Get:3 http://mirrordirector.raspbian.org wheezy Release.gpg [490 B]
Get:4 http://mirrordirector.raspbian.org wheezy Release [14.4 kB]
Hit http://raspberrypi.collabora.com wheezy Release.gpg
Hit http://raspberrypi.collabora.com wheezy Release
Get:5 http://archive.raspberrypi.org wheezy/main armhf Packages [130 kB]
Get:6 http://mirrordirector.raspbian.org wheezy/main armhf Packages [6,904 kB]
Hit http://raspberrypi.collabora.com wheezy/rpi armhf Packages
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en_GB
Ign http://raspberrypi.collabora.com wheezy/rpi Translation-en
Ign http://archive.raspberrypi.org wheezy/main Translation-en_GB
Ign http://archive.raspberrypi.org wheezy/main Translation-en
Get:7 http://mirrordirector.raspbian.org wheezy/contrib armhf Packages [23.6 kB]
Get:8 http://mirrordirector.raspbian.org wheezy/non-free armhf Packages [49.3 kB]
Get:9 http://mirrordirector.raspbian.org wheezy/rpi armhf Packages [592 B]
100% [6 Packages xz 0 B]
```

It may look like gobbledygook, but it's telling Raspbian which software needs to be updated

### UPDATE YOUR SOFTWARE AND OS

The software on Raspbian will be periodically updated online, bringing with it bug fixes and security updates. Those don't automatically sync with the Raspberry Pi, though, and you should regularly check to see if there are any updates for your system. This is handled entirely in the terminal again, much like the software installation.

The update process consists of two parts: first you need to update the repositories; this is the list of available software and their versions kept on your system. You do that by first entering the command:

```
$ sudo apt-get update
```

This will check online to see the state of the software repositories and report back to the Raspberry Pi, saving any changes. It will then determine what software can be and should be updated, but you then need to tell it to perform the update with this command:

```
$ sudo apt-get upgrade
```

Every now and then, there may be a major update to the Raspbian operating system, bringing with it big changes like a new interface or browser, etc. It's very rare, but when it happens, you can perform the upgrade with:

```
$ sudo apt dist-upgrade
```

## USE THE GPIO

Make your first small project with a bit of code and the GPIO pins

The GPIO port is one of the most powerful tools at the Raspberry Pi's disposal, allowing you to connect directly to an electronic circuit to control it. In such a system, the Pi is referred to as a microcontroller. This is what makes the Raspberry Pi great for big projects, as you can use it to program a machine or circuit, and even have it connect to the internet via the other Raspberry Pi functions so that it can control contraptions with web data.

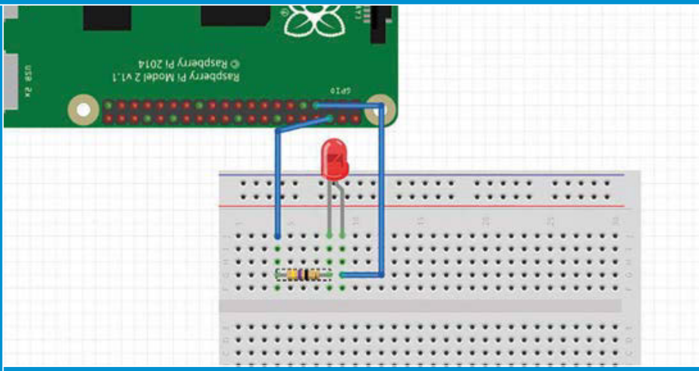
Each of the GPIO pins can do something different and very specific. At the basic core, though, you can have them provide power consistently to part of a circuit, program a power switch to one of the pins, and even have it sense a change over the pins (thanks to resistance). These three basic functions allow you to do a lot, and can be programmed with Python.

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)

GPIO.output(7,True)
time.sleep(1)
GPIO.output(7,False)
time.sleep(1)
GPIO.output(7,True)
time.sleep(1)
GPIO.output(7,False)

print “Done”

GPIO.cleanup()
```

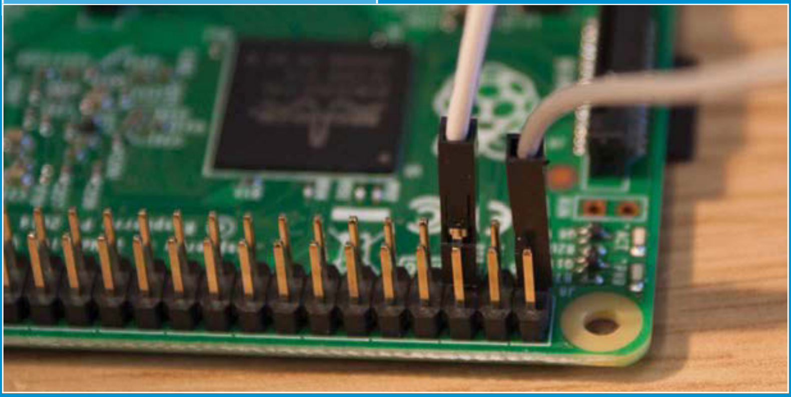
We're going to wire up an LED bulb to be programmable from the Raspberry Pi, to turn it on and off again a few times. For this, you will need a breadboard prototyping circuit board, an LED, a 50-ohm resistor, and some wires. Refer to our Fritzing diagram on the right, to see how it's wired up; the negative end of the LED goes to the ground rail on the Raspberry Pi (which is where the flow of electricity ends), and a programmable pin goes through the 50-ohm resistor to provide power to the LED when it's turned on.

Open up IDLE, the Python programming software, and create a New file. Save it as **led.py**, and input the code from the code listing. What the code does is first tell Python to use the GPIO module so we can connect to the GPIO pins, by **importing** the module. We then import the time module so we can create a delay between commands. We then tell the code to treat the GPIO pins as the number they are on the board, and to turn the seventh pin into an **output**. We alternate between **True** and **False** so that it turns the pin on and off. Once it's cycled a few times, it will **print** the message 'Done' into IDLE, and finally turn off the GPIO pins.

You can do a lot more with GPIO if you want to, and this is a good way to start before moving on to bigger projects.


Wire the circuit up just like this

| PIN 1 | | PIN 2 |
|---|---|---|
| +3V3 | 🟧 🟥 | +5V |
| GPIO2 / SDA1 | 🟦 🟥 | +5V |
| GPIO3 / SCL1 | 🟦 ⬛ | GND |
| GPIO4 | 🟩 🟨 | TXD0 / GPIO14 |
| GND | ⬛ 🟨 | RDX0 / GPIO15 |
| GPIO17 | 🟩 🟩 | GPIO18 |
| GPIO27 | 🟩 ⬛ | GND |
| GPIO22 | 🟩 🟩 | GPIO23 |
| +3V3 | 🟧 🟩 | GPIO24 |
| GPIO10 / MOSI | 🟪 ⬛ | GND |
| GPIO9 / MISO | 🟪 🟩 | GPIO25 |
| GPIO11 / SCLK | 🟪 🟪 | CE0# / GPIO8 |
| GND | ⬛ 🟪 | CE1# / GPIO7 |
| GPIO0 / ID_SD | 🟦 🟦 | ID_SC / GPIO1 |
| GPIO5 | 🟩 ⬛ | GND |
| GPIO6 | 🟩 🟩 | GPIO12 |
| GPIO13 | 🟩 ⬛ | GND |
| GPIO19 / MISO | 🟪 🟪 | CE2# / GPIO16 |
| GPIO26 | 🟩 🟪 | MOSI / GPIO20 |
| GND | ⬛ 🟪 | SCLK / GPIO21 |
| PIN 39 | | PIN 40 |


You can connect directly to the Raspberry Pi without needing any special slots over the pins

# SUBSCRIBE TODAY!

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

## Subscription benefits

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

100 PAGES OF RASPBERRY PI

SAVE UP TO 25%

THE OFFICIAL RASPBERRY PI MAGAZINE
NEW!
The MagPi
Win! 10 RASPI 2s THANKS TO ELEMENT14.COM
Issue 36 August 2015
100 pages of hacking & making
GET STARTED WITH RASPBERRY PI
AMAZING PI PROJECTS
WINDOWS 10 ON RASPBERRY PI
THE ONLY MAGAZINE WRITTEN BY THE COMMUNITY, FOR THE COMMUNITY

## Pricing

### Get the first six issues:

£30 (UK)

£45 (EU)

£50 (RoW)

### Subscribe for a year:

£55 (UK)

£80 (EU)

£90 (RoW)

### Direct Debit

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

### Three ways to subscribe:

- Visit **www.bit.ly/MagPiSubs**
- Call +44 (1)1202 586848
- Use the form on this page

Or search 'The MagPi' on your devices:

Available on the App Store

Get it on Google play

## SUBSCRIPTION FORM

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

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

Mag#36

YOUR DETAILS Mr ☐ Mrs ☐ Miss ☐ Ms ☐

First name ........................................ Surname ........................................

Address ..................................................................................................

..........................................................................................................

Postcode ........................................ Email ........................................

Daytime phone ........................................ Mobile ........................................

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

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

First name ........................................ Surname ........................................

Address ..................................................................................................

Postcode ........................................ Email ........................................

PAYMENT OPTIONS

**1 DIRECT DEBIT PAYMENT** £12.99 every 3 issues (UK only)
**Instruction to your bank or building society to pay by Direct Debit**

DIRECT Debit

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

Service user number 8 3 8 7 7 3

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

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

Address ..................................................................................................

..........................................................................................................

.................................................... Postcode ........................................

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

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

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

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

Signature ........................................ Date ☐☐/☐☐/☐☐

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

SUBSCRIPTION PRICING WHEN PAYING BY CHEQUE OR CREDIT/DEBIT CARD

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

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

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

Card number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

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

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

Signature ........................................ Date ☐☐/☐☐/☐☐

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

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

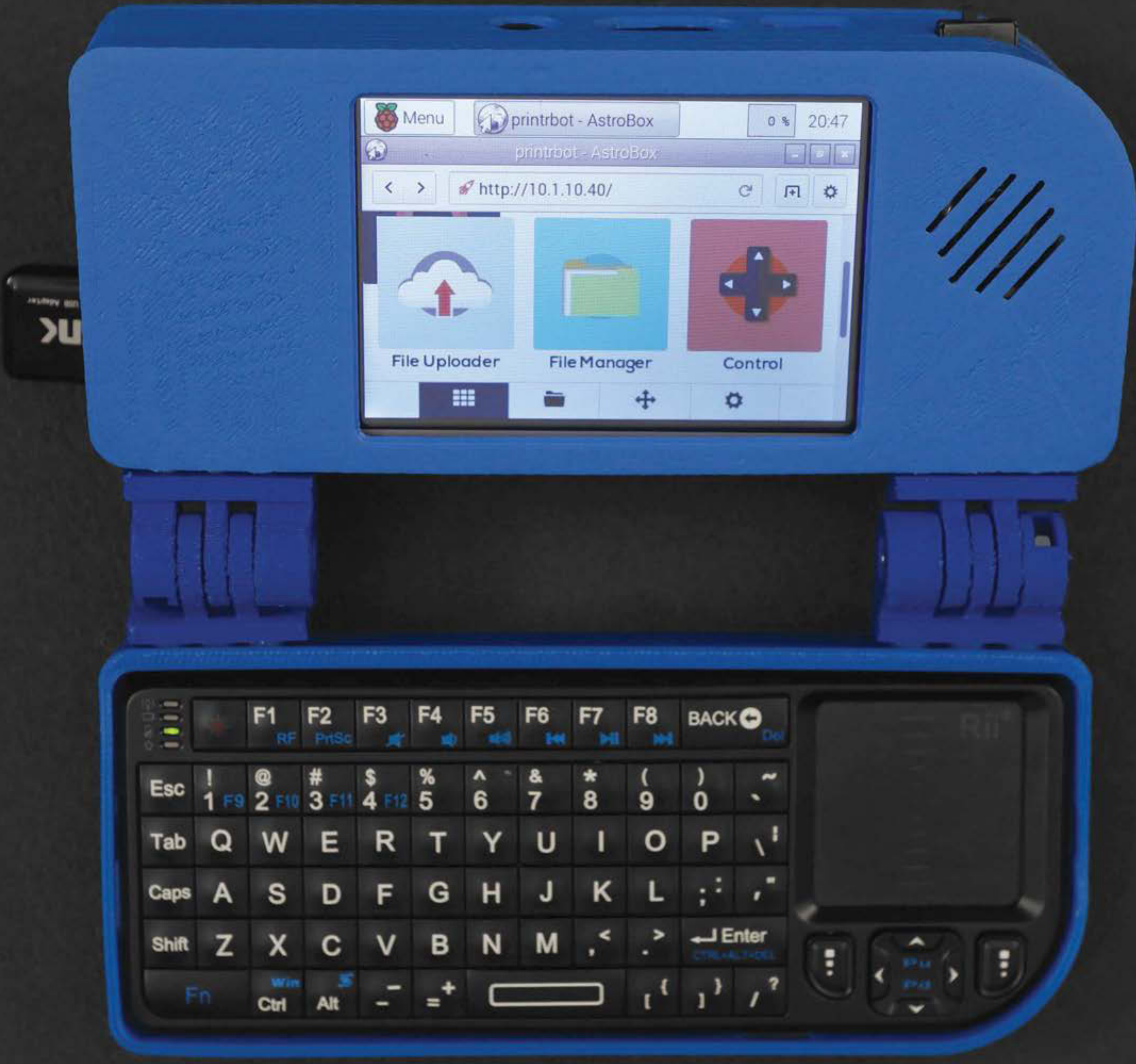
☐ Please tick this box if you DO NOT want to receive any other information from Select Publisher Services Ltd.

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

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

# RASPBERRY PI NOTEBOOK

Adafruit's star makers, the **Ruiz brothers**, are back with another stunning handheld Raspberry Pi project…



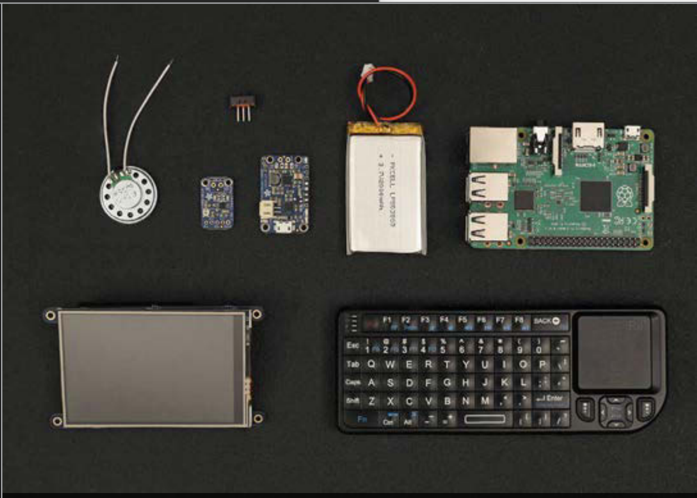
Here's a brilliant project for you or your family to test your hacking and making skills with this weekend. This beautiful retro-styled mini-notebook, built by Adafruit's Ruiz brothers (**bit.ly/1MkrxGe**), is powered by a Raspberry Pi 2 and an Adafruit 3.5″ PiTFT touchscreen and, frankly, not a great deal more! The project comes hot on the heels of the excellent Pocket PiGRRL that we covered in issue 34 (**raspberrypi.org/magpi/issues/34**), a home-brew Nintendo Game Boy build we're still swooning over three months later.

Besides the Raspberry Pi 2 and PiTFT, for control the project features a mini-chiclet keyboard with built-in trackpad. It's a widely available wireless input device that's both affordable and easy to use. The 200mAh rechargeable battery, which sits sandwiched between the Pi 2 and the screen, is managed by a PowerBoost 1000C, a load-sharing DC/DC boost converter capable of doling out 5.2V and charging the battery while your gadget is in use. Finally, a small amp is connected to a tiny speaker for audio output.

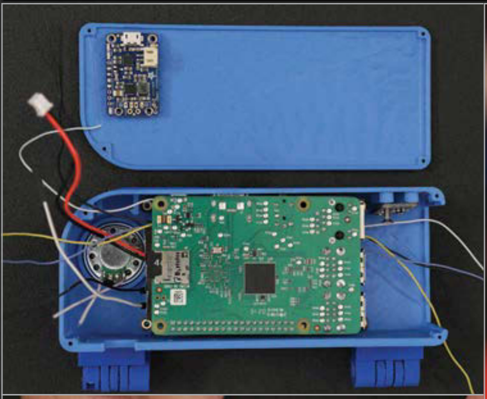
> "The project comes hot on the heels of the excellent Pocket PiGRRL we covered in issue 34"

While the hardware is the really exciting bit, the 3D-printed chassis is a work of art, too. Take, for example, its totally modular hinged design. While it works really well on this Raspberry Pi mini-notebook, you could reuse it for 101 different hardware projects.

Like all of Adafruit's excellent Raspberry Pi projects, you can find a full shopping list of parts, software and 3D printing files for the Raspberry Pi Notebook on the Adafuit Learning system at **learn.adafruit.com**.


**Above** All the components laid out – it's not particularly complex

**Above** The 3D-printed case, complete with modular hinges



**Above** It's a thing of beauty and a cracking weekend build, assuming you have access to a 3D printer

**DANIEL SPIES**

A Dutch magician who likes to build his own tricks, and has taken to more advanced electronics and microcontrollers to improve his shows.

**facebook.com/raspberrynin10do**

# NIN10DO

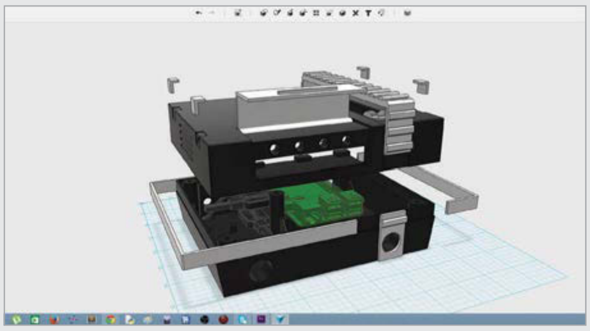
A 3D-printed Raspberry Pi-powered Nintendo Entertainment System that can also play Sega games…

**Quick Facts**

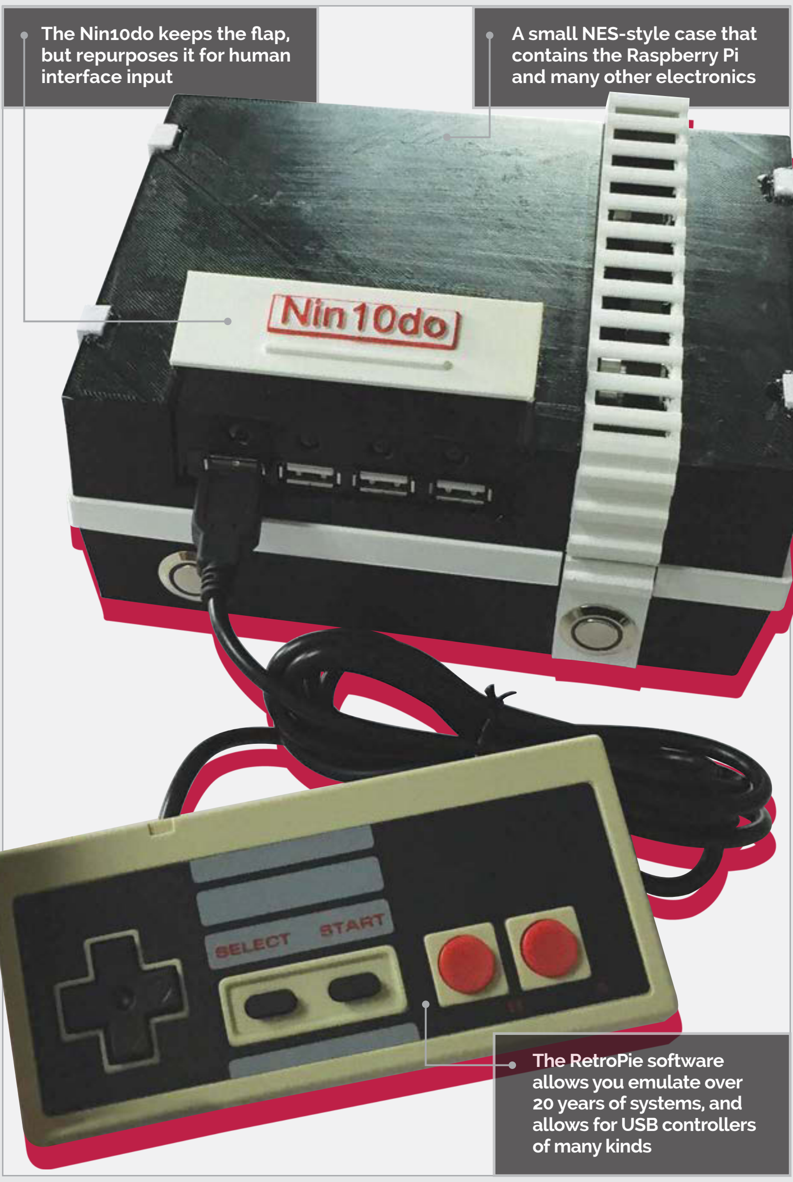
- The first version took about two months
- The case is made of XT-CO-Polyester instead of the normal ABS plastic
- Daniel now has plans and the confidence to develop his Pi box of tricks
- A future version will exist with laser-cut acrylic
- It's not a trick, it's an illusion

When you're a magician looking to make your tricks – sorry, *illusions* – better, you have several paths to go down. You could start buying or recreating other people's work; however, that's not entirely original, and you risk other people having seen it already. What truly great magicians do is to create their own illusions – original tricks that wow an audience with a wonderful show. Creating tricks and illusions for people-sized magic sounds similar to another hobby: that of being a maker. Makers love the Raspberry Pi, and this is where professional magician Daniel Spies found himself at the end of 2014.

"I wanted to start learning how to program in Python so I could integrate a Raspberry Pi in complex magic acts," Daniel tells us. "The Raspberry Pi would be great for starting special music



**Below** A 3D modelled and printed chassis allows for the perfect fit for the project



The Nin10do keeps the flap, but repurposes it for human interface input

A small NES-style case that contains the Raspberry Pi and many other electronics

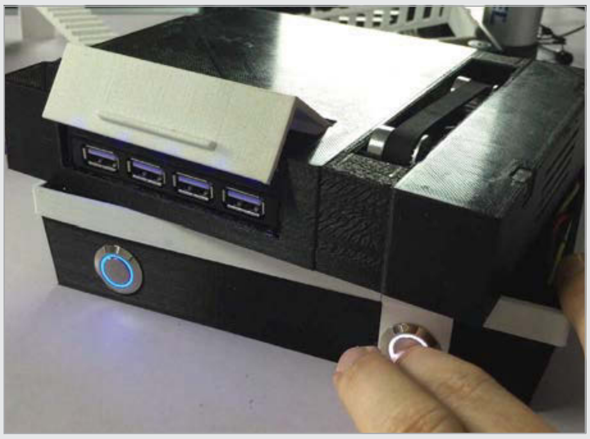
The RetroPie software allows you emulate over 20 years of systems, and allows for USB controllers of many kinds or sound effects, operating small smoke machines or even electromagnets. After I mastered the basics (servos, LEDs, etc.) it was time to build something as a practice project. It had to include as many different skills as possible, like CAD drawing, 3D printing, mechanical engineering, programming, and electronics. Then I saw guys on YouTube using their Raspberry Pi to emulate classic games on their TV. The idea was born."

Daniel decided to 3D-print his own custom NES case for the Raspberry Pi; it was smaller, sleeker and with a few more tricks than the 30-year-old console, including motors, lots of flashing lights, and the ability to play N64 games. There were some rules for the project to make it worthwhile as a test run for bigger things, though. "[These] rules included the Nin10do must look like a firm professional game console; it must run my Python script in the background but *not* sacrifice any speed or usability; it must have a option to be turned on and off without damaging the software or the SD card, and the cover must not damage itself if, for some reason, it is opened twice by the stepper motor."

With this in mind, Daniel went to work. He created full schematics for the electronics, making use of a series of LEDs, stepper motors, driver boards, timing belts, and lots of extenders to reach the I/O ports to make it actually usable. All of the components used were basic ones, so while there was a lot of soldering involved, he wasn't gutting any existing devices to make the project work.

With this and the coding in place (which you can find on his GitHub page – **bit.ly/1RkBirf**),

> Daniel designed and 3D-printed a case reminiscent of the NES that could house his Raspberry Pi

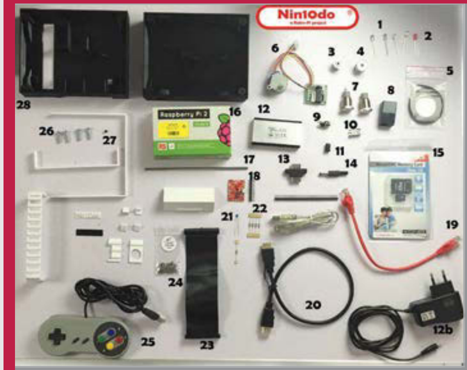

**Above** The flap is motorised, revealing the USB ports when turned on, thanks to a custom Python script

Daniel designed and 3D-printed a case reminiscent of the NES that could house his Raspberry Pi and electronics. The important question, though, was whether or not he succeeded...

"The latest version runs very stable!" Daniel reveals. "I added some parts a while ago (small capacitor in the second momentary switch) and changed the USB hub to a better version. This solved the last (minor) bugs."

The full build process, printable 3D models and code are all available online from Daniel, so if you want to give it a go yourself, the tools are there. There may even be a kit coming in the future...

## HOW TO CREATE A NIN10DO

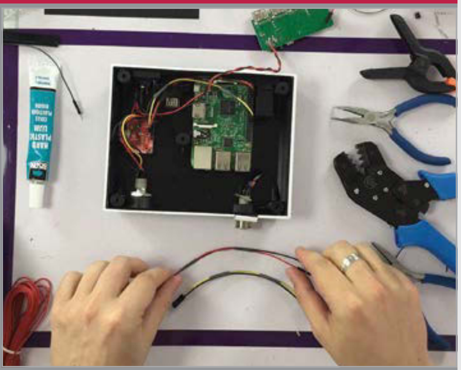


### >STEP-01
**Electronics**

Put together all the necessary components you need for the project. Apart from the Pi, this includes the LEDs, the other controllers, USB extensions, and the motors necessary for the mechanics section.

### >STEP-02
**Programming**

The main emulation is done by the RetroPie software. However, you also need to create a Python script that controls the buttons, motors and LEDs, depending on the situation.



### >STEP-03
**Mechanics**

Finally, fit everything into a custom, 3D-printed case. This step includes assembling the mechanised flap, along with installing the USB ports and physical buttons.

**DENIS PAPATHANASIOU**

Denis Papathanasiou is a managing director at Banrai LLC, an analytical technology firm that he co-founded in 2001.

**denis.papathanasiou.org**

# PISCAN

One project builder turned a Raspberry Pi into a home-made Amazon Dash scanner. **Lucy Hattersley** interviews the maker of PiScan

**Quick Facts**

- It uses the Open Product Data database to match products
- Unlike Amazon's Dash, you can order any product from Amazon with it
- It cost around $70 to build (including the cost of the Raspberry Pi)
- All the software and installation instructions are on GitHub
- A similar project called Oscar served as the initial inspiration

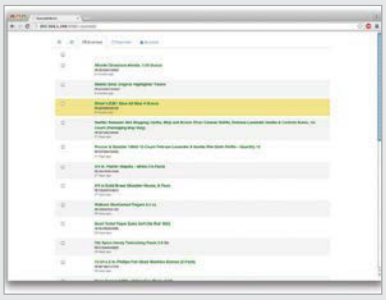
Wouldn't it be amazing if you could just scan a barcode and get another one of that particular product through your door the next day? Amazon clearly thought so; that's why it built a mini device called Dash that did just that. No more online shopping, just scan and go...

PiScan is an open-source version of Amazon Dash created by Denis Papathanasiou. With it, you can scan products using a Raspberry Pi and order them directly. It's great fun and incredibly practical, and it's even more powerful than Amazon's official device.

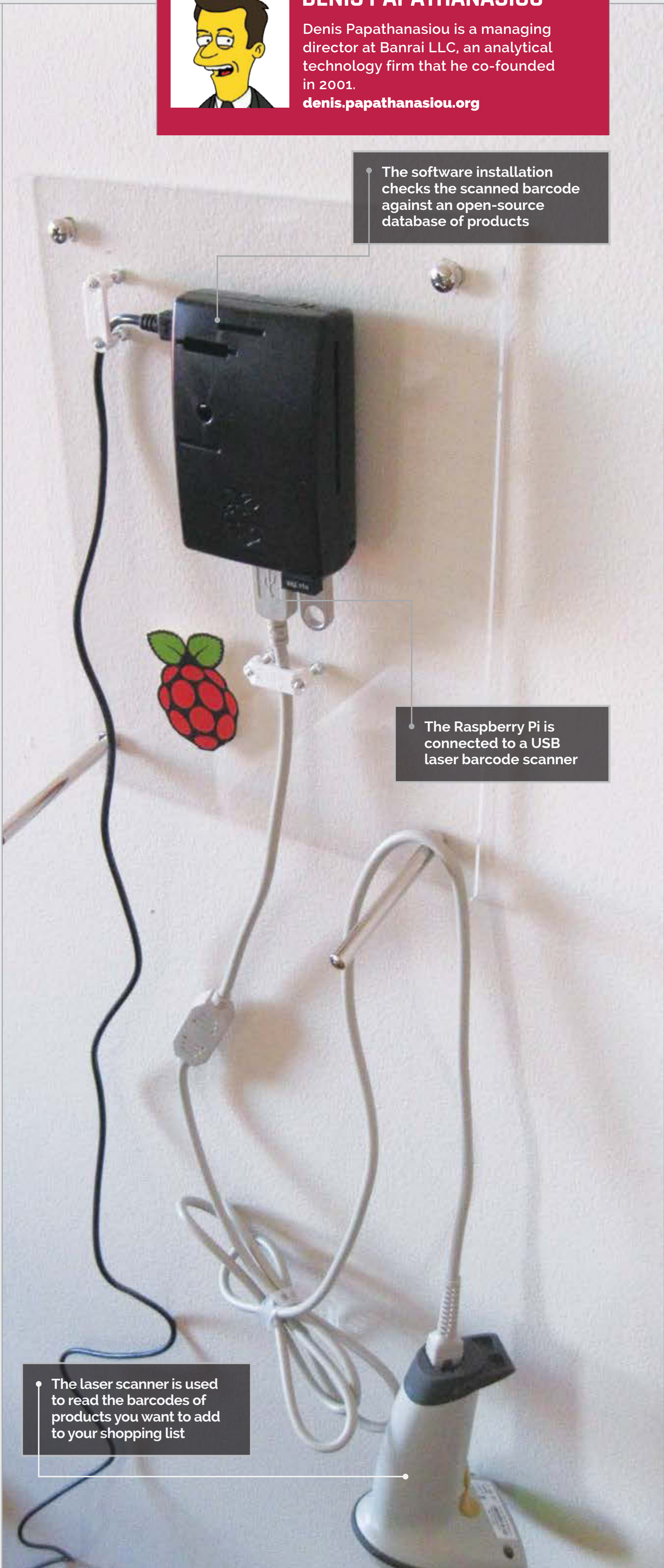
"It's the ultimate in lazy-person shopping," says Denis. "PiScan will read the barcode on any consumer product and order it for you from an online vendor."

The inspiration came from another Raspberry Pi project called Oscar. That just converted "product barcodes into a grocery list," explains Denis, "but I thought it would be nice to take it one step further."

PiScan converts scanned items into a list that you can use to order products, using Amazon's API.



**Right** A list of recently scanned products. Placing a tick next to the product enables you to shop for that item



The software installation checks the scanned barcode against an open-source database of products

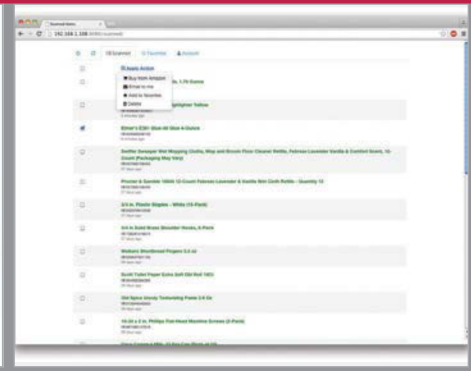
The Raspberry Pi is connected to a USB laser barcode scanner

The laser scanner is used to read the barcodes of products you want to add to your shopping list

## SETTING UP PI SCAN

### >STEP-01
**Laser USB scanner**

The main piece of hardware is a USB laser barcode scanner. This is used to read barcodes on products and send the digital information to the Raspberry Pi (which then matches it to a database).



### >STEP-02
**Software installation**

With the scanner connected, you install the software. Denis has put a pre-built ARM binary on GitHub (plus the source code). He's hosting an open barcode database (**saruzai.com**), or you can create one of your own.

### >STEP-03
**Scan and shop**

Use the barcode scanner to scan products. An open-source database of products is used to match the barcodes. You can then tick products in the list and shop for them automatically on Amazon.

In terms of hardware, PiScan is pretty basic. "I used a Raspberry Pi Model B with a WiFi dongle and a USB laser barcode scanner," says Denis. The scanner is the only extra hardware requirement to a regular setup, and you can pick one up from Amazon for less than £20.

"I wrote software for the Raspberry Pi to listen for input from the barcode scanner," adds Denis. "The scanner works just like I didn't need to use any of the Pi's GPIO pins... Most of the work went into the software design, to make sure that the input from the barcode scanner was being read correctly.

"It's great," he tells us. "The dedicated scanner device reports barcode numbers with high fidelity.

"I've been using it to buy staple products regularly."

> PiScan converts scanned items into a list that you can use to order products using Amazon

a keyboard, except its input comes in short bursts of characters."

The input is a 10- to 13-digit number matched to the Open Product Data database (**product-open-data.com**).

"If there's a match," explains Denis, "it will put the name of a product into a list." The Raspberry Pi delivers the list to you as an email, or you can tick items to add to your Amazon shopping cart.

Building the PiScan "was simple and straightforward," says Denis. "There was nothing to solder and I didn't need to use any of the Pi's GPIO pins..."

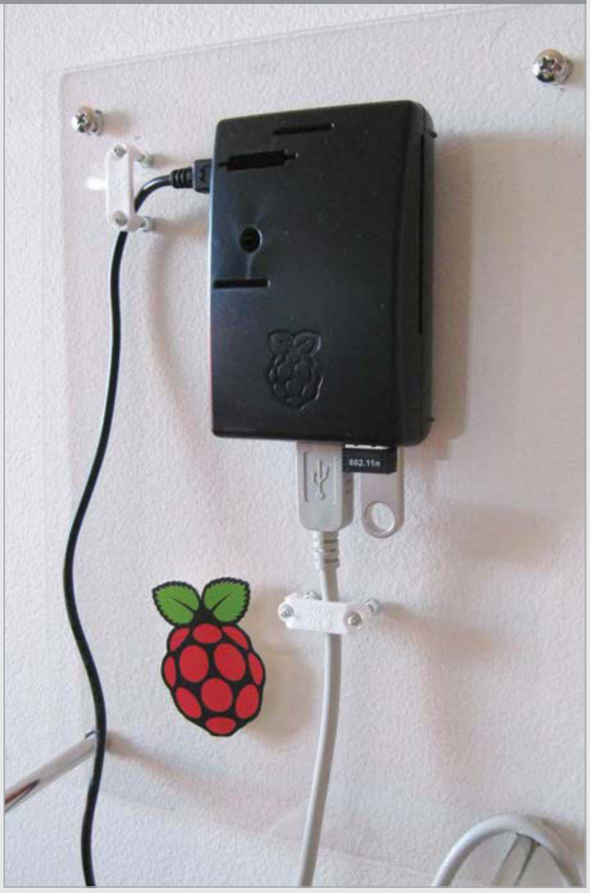
It's even better than Amazon's Dash scanner, claims Denis. "Dash only works with certain brand products, and it doesn't give you any say about quantity or price."

The only vendor so far is Amazon, but Denis is looking to expand to other retailers.

"Tesco in both the UK and Korea supposedly offer similar APIs," says Denis, "and there are probably other vendors out there that I'm not aware of."

If you fancy making your own PiScan personal product scanner, the software is freely available to download from GitHub (**github.com/Banrai/PiScan**), along with installation instructions.

"I've tried to make them simple to understand," says Denis, "but feedback is welcome."

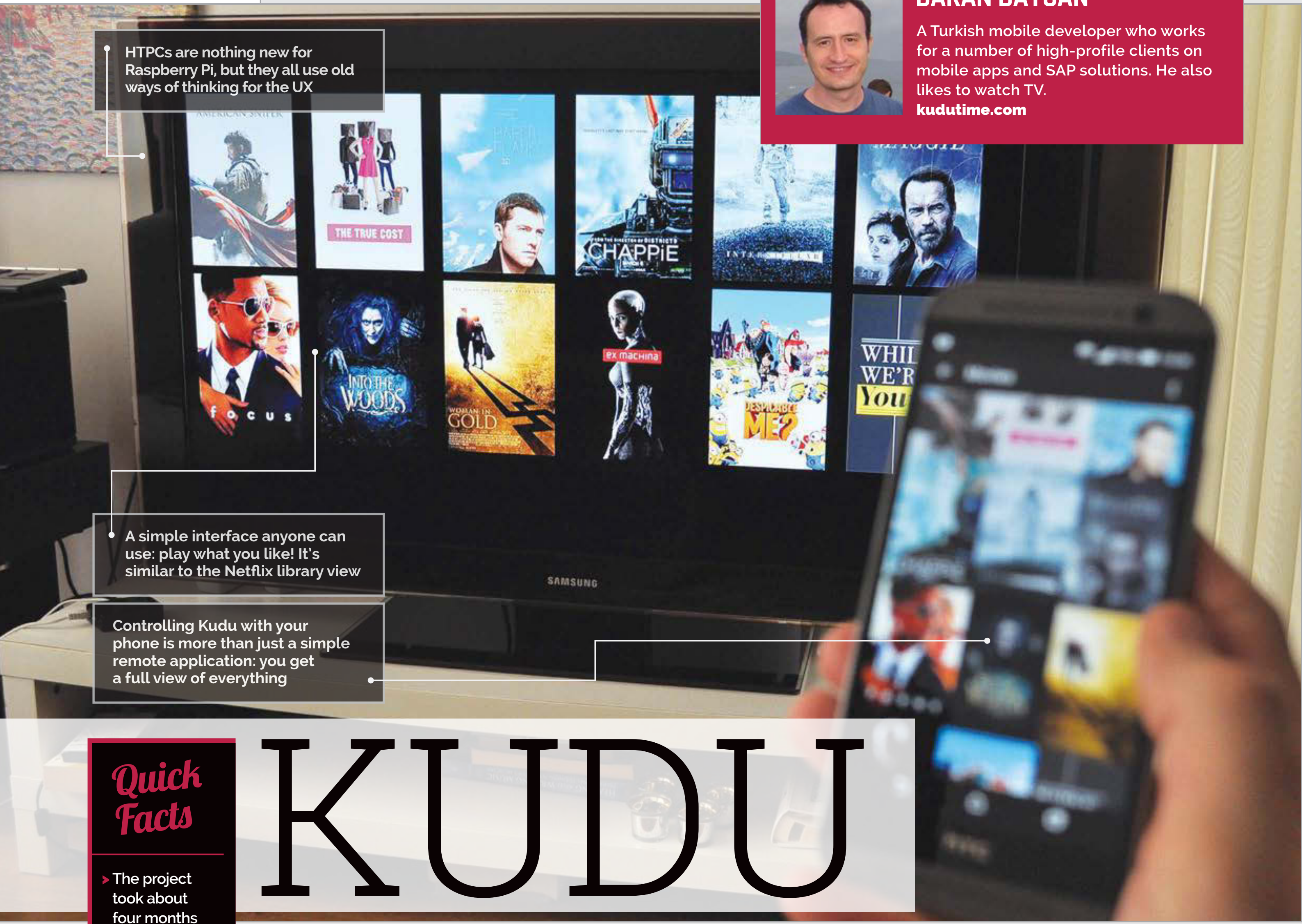


**Above** The finished PiScan device is attached to a wall and orders products online when you scan them

**BARAN BAYGAN**

A Turkish mobile developer who works for a number of high-profile clients on mobile apps and SAP solutions. He also likes to watch TV.

**kudutime.com**



HTPCs are nothing new for Raspberry Pi, but they all use old ways of thinking for the UX

A simple interface anyone can use: play what you like! It's similar to the Netflix library view

Controlling Kudu with your phone is more than just a simple remote application: you get a full view of everything

# KUDU

**Quick Facts**

- The project took about four months
- This is Baran's first Pi project
- Previously, Baran used a full-scale PC for media
- Baran worried the Pi would overheat while playing 1080p video
- After running it for weeks with no problems, he was happy

A new solution to the home theatre PC that uses the Raspberry Pi, but what does it do that Kodi can't?

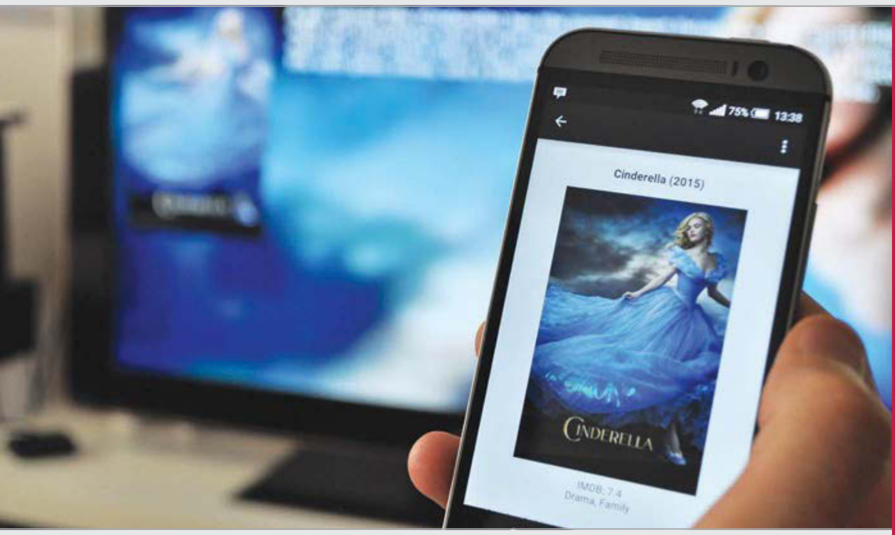
While many folks had grand plans for their Raspberry Pi when they first got it, a significant number ended up using it as a home theatre PC (HTPC). Still, the Pi is really good for that job, for all the reasons that make it so attractive for education purposes or maker projects: it's small, has low power consumption, and is easy to customise. For Baran Baygan, though, the standard solutions for HTPCs weren't quite good enough for his needs.

"I have tried several media centre approaches like OpenELEC, XBMC etc.," Baran tells us. "The problem with those is that they are mainly for geeks. My father or my mother, for instance, couldn't use OpenELEC. It is unnecessarily complicated. They are trying to solve too many problems. It makes their UX too complicated."

There are further issues with traditional solutions, as well, according to Baran: "Another problem is that no media centre is using the full potential of mobile devices. Mobile devices are going to replace remote controls. Remote controls with hard buttons are history. Content browsing can be done, and should be done, in mobile devices with soft controls. Most media centres still follow the old paradigm to browse the content on a TV screen. Using the remote control to move a cursor on the TV, or trying to select a tile with arrow keys, is very difficult. Browsing content on a mobile device is, on the other hand, very easy."

The final product is a lot more than just a modification for OpenELEC or Kodi. An entirely new app was developed by Baran for the Raspberry Pi itself, running on Raspbian.

"Once I was convinced that the Raspberry Pi could handle the workload, I had to move on to mobile app development, the remote control app," Baran explains to us. "Because I am working alone on the project, I had to pick either iOS or Android to start with. And because I use an iPhone, I started with the iOS version. The day I had the mobile



**Above** Your phone or tablet acts as an extra screen that not only controls Kudu, but also gives you extra information

> "Whenever you open YouTube in Kudu, you actually launch the website on your phone"

app on my phone and Raspberry Pi hooked up to TV, and was able to watch YouTube and almost everything else, I was convinced that Kudu was a real problem-solver for anyone."

With some extra development help from a friend, an Android version was created. Baran had one final step to perform: "Last but not least, there had to be a web part consisting of channels running in mobile apps. These channels are webpages, essentially, and run in WebViews in the mobile application… Whenever you open YouTube in Kudu, you actually launch the website on your phone. Websites and Kudu communicate through JavaScript. The mobile app gets the commands from websites and relays those commands to the Kudu device (i.e. the Raspberry Pi) via our real-time communication server. Every Kudu [instance] is connected to a server all the time."

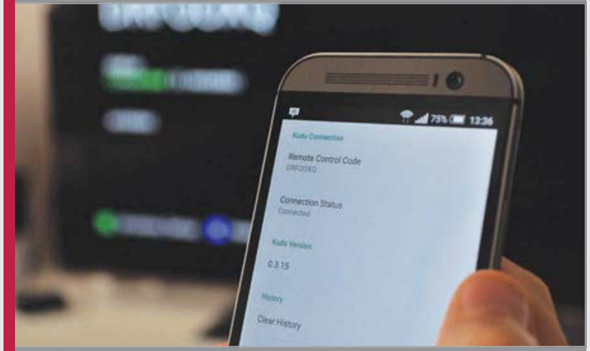
What Baran has created may be the future of how we watch media on TV. If you're interested in the project, there are kits that Baran is selling so you can create your own future home theatre.

**Above** Play music, movies and more via multiple sources on Kudu, just like any other HTPC

## HOW TO MAKE A NEW HTPC OS

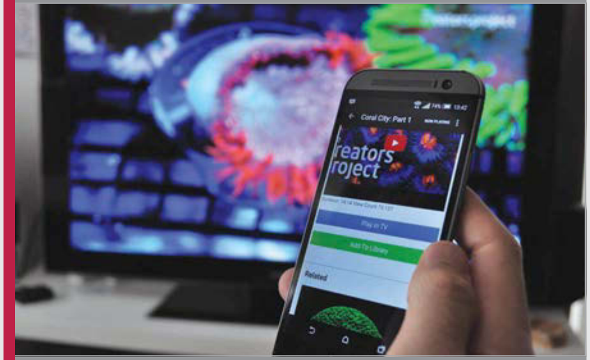
### >STEP-01
**Program your Pi**

At the very core, you need your Raspberry Pi to actually display the video and act as the hub for everything to work on. This means you start with creating software for the Pi.



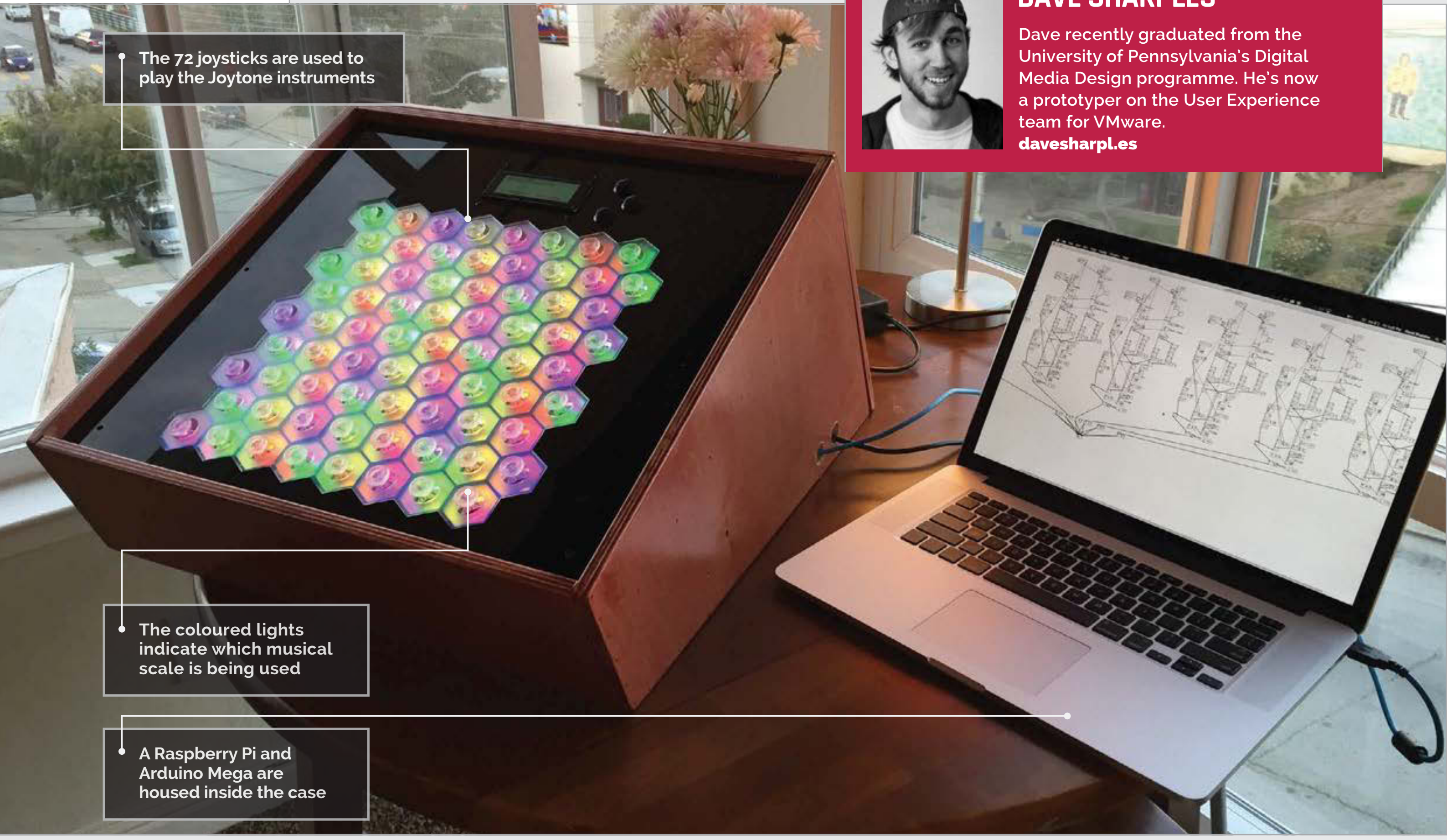
### >STEP-02
**Create your remote**

Unlike other HTPCs, Kudu is controlled from a remote screen, in this case a smartphone app that connects to the Raspberry Pi.



### >STEP-03
**Get web working**

Once you have each part working, you need to get web content to play on your Pi and be controlled on your device, and this requires WebViews.


The 72 joysticks are used to play the Joytone instruments
The coloured lights indicate which musical scale is being used
A Raspberry Pi and Arduino Mega are housed inside the case

**DAVE SHARPLES**

Dave recently graduated from the University of Pennsylvania's Digital Media Design programme. He's now a prototyper on the User Experience team for VMware.
**davesharpl.es**

**Quick Facts**

- Dave used Express PCB (**expresspcb.com**) to do the board layout
- The Joytone survived six weeks in the kids' area of the Toronto Film Festival
- The parts for the Joytone cost around $1,050
- Around 16,000 people saw it at the Toronto Film Festival

# JOYTONE

A unique musical instrument made from joysticks, lights, and powered by the Raspberry Pi. **Lucy Hattersley** talks to the Joytone's maker...

Every so often we come across a project so spectacular we have to share it. Joytone is one such creation. Designed and crafted by engineering expert Dave Sharples, the Joytone is a unique musical instrument played using an array of mini-joysticks.

"I've always wanted to be able to play a musical instrument," says Dave, "and a couple years ago I took a music theory class to see if that could help me learn piano."

Rather than learning the piano, Dave had a revelation about musical structures. "I became fascinated with the patterns associated with musical structures and realised how beautifully simple music can be," he reveals.

"Acoustic instruments are designed around the physical phenomena that produce sound," says Dave. "A violin is smaller than a cello because shorter strings make higher notes, not because it's convenient for the player."

Electronic instruments, like synthesizers and electric guitars, don't have to reproduce these limitations. There's no inherent reason for an electronic instrument to resemble the acoustic tool it is derived from.

So Dave set about creating a unique new musical instrument that made sense. "Joytone is a unique new musical instrument that features a hexagonal grid of 72 joysticks," he tells us. "The Joytone's hexagonal grid exposes musical patterns that are normally obscured by the quirks of common acoustic-style interfaces, like the white and black keys of a piano.

"Each joystick plays one note and the motion of the joystick affects the volume and character of the note," explains Dave. "The way the notes are distributed across the perfectly hexagonal grid means that all kinds of musical patterns become clear.

"Every major chord has the same finger shape, no matter what note you start on," he continues. "This is true of minor chords, scales or any other kind of musical structure, making [the Joytone] much easier to learn and play."

**Above** A laser cutter being used to cut the piece of acrylic.

"There's no inherent reason for an electronic instrument to resemble the acoustic tool it is derived from"

## Building the Joytone

The 'keys' of the Joytone are created using 72 joysticks. These are Xbox-style thumbsticks and were replacement parts that Dave found on eBay. They have clear plastic grips to let the LED lights shine through.

The lights (also sourced from eBay) are used to indicate which notes belong to the selected musical key. They are connected to NeoPixel (WS2812) strands from Adafruit (**adafruit.com**).

"Each joystick is really just a pair of potentiometers connected to a little plastic post," Dave explains. "One measures movement along the X-axis; the other measures movement along the Y-axis."

These are connected to an Arduino Mega via a series of custom circuit boards designed by Dave himself. "With two signals per joystick and eight joysticks per row, there are 16 analogue signals generated by each row of joysticks – a grand total of 140 for the whole instrument," he calculates.

"The Arduino only has 16 analogue inputs, so the Joytone makes use of multiplexers to handle all those analog signals. I designed custom circuit boards for the rows of joysticks, and at the end of each board is a 16-channel multiplexer.

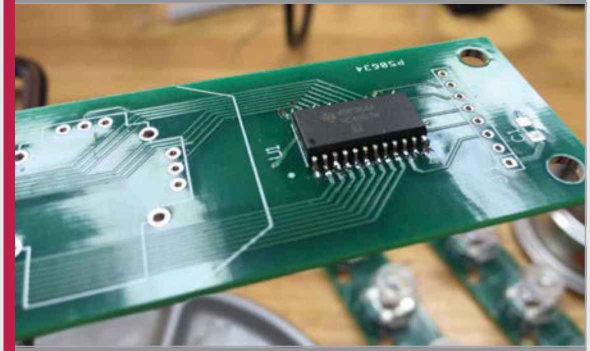
"A multiplexer is like a big switch," says Dave. "The output wires from the multiplexers are connected to analog inputs on the Arduino, [which] can set all the multiplexers to forward channel 0, then read all nine inputs. It then set the multiplexers to forward channel 1, then read all nine inputs again, and so on. Once it knows the positions of every joystick on the board, it can go through and figure out which ones are being moved.

"For each active joystick," continues Dave, "the Arduino looks up the MIDI note it represents, then bundles that information up with the two values coming from the joystick sensor and sends a little MIDI message to the Raspberry Pi."

## WHAT PARTS ARE USED?

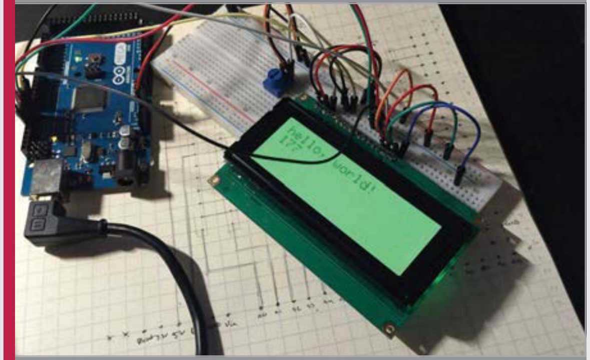
### >STEP-01
**72 joysticks**

The interface of the Joytone comprises these mini-joysticks (picked up from eBay). The clear plastic grips enable light from LEDs to shine through.



### >STEP-02
**Multiplexer and Arduino Mega**

An Arduino Mega and multiplexer are used to assess the stick positions. The Arduino only has 16 analogue inputs, so the multiplexer enables it to handle all 72 joystick signals.



### >STEP-03
**Screen and Raspberry Pi**

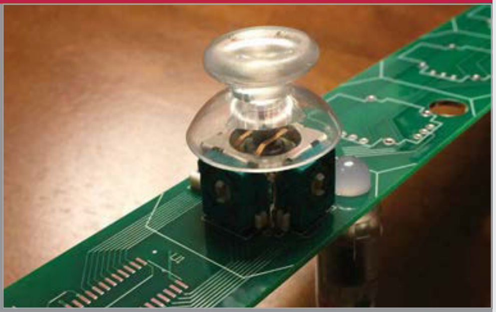
The screen is an RGB backlight positive LCD 20×40 from Adafruit (**adafruit.com**). The screen provides feedback to the user. The Arduino Mega sends MIDI information to the Raspberry Pi (which plays the audio).

## PUTTING JOYTONE TOGETHER

**>STEP-01**

**Custom circuit boards**

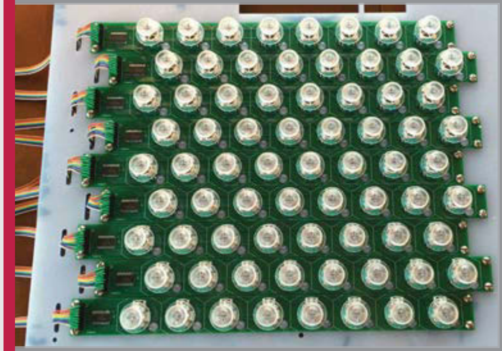
At the heart of the Joytone are nine printed circuit boards (PCBs). These make the Joytone more reliable and easier to fix than earlier models. Dave signed up for TechShop in San Francisco to learn how to design PCBs.



**>STEP-02**

**Fitting the joysticks**

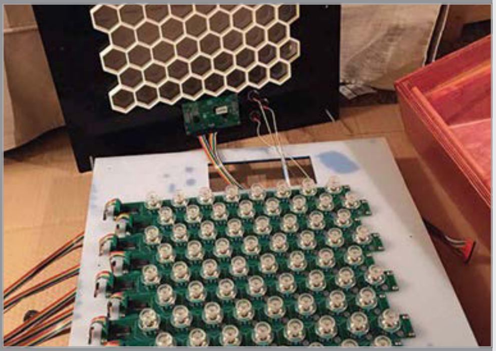
The clear joysticks and lights are fitted into the PCBs. Rather than use individual lights, the Joytone employs a strand of 25 Digital RGB LED Pixels (**adafruit.com**). These are easier to fit and more reliable than individual LEDs.



**>STEP-03**

**Arranging the boards**

Nine PCBs, each containing eight joysticks, are arranged into a square in this offset pattern. With two signals per joystick, there are 16 analogue signals generated by each row.



**>STEP-04**

**Multiplexer and Arduino**

Multiplexers are used to connect the PCBs to the Arduino Mega. They enable the Arduino to examine the position and movement of all 72 joysticks at once.

**>STEP-05**

**Raspberry Pi**

The Arduino looks up the MIDI note it represents and sends that information up to the Raspberry Pi, which then uses it to create the audio sound.

**>STEP-06**

**Played with sticks**

A sheet of acrylic is laser-cut into a honeycomb shape to hold the joysticks. A box holds all the equipment, and the device is ready to play. A single finger is used to play each stick and you can play up to six notes at once.

### The software

The Joytone depends on PureData (**puredata.info**): "You can connect blocks with lines to direct the flow of data through a series of mathematical operations, to produce all kinds of strange and delightful behaviour."

Dave built a PureData patch for the Joytone. "[It] receives MIDI messages, then unpacks them and passes the values inside into a group of blocks that produces a synthesized note," he explains.

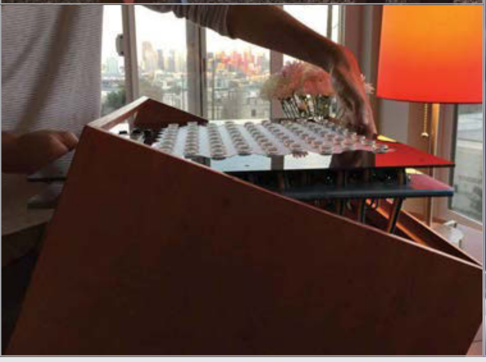
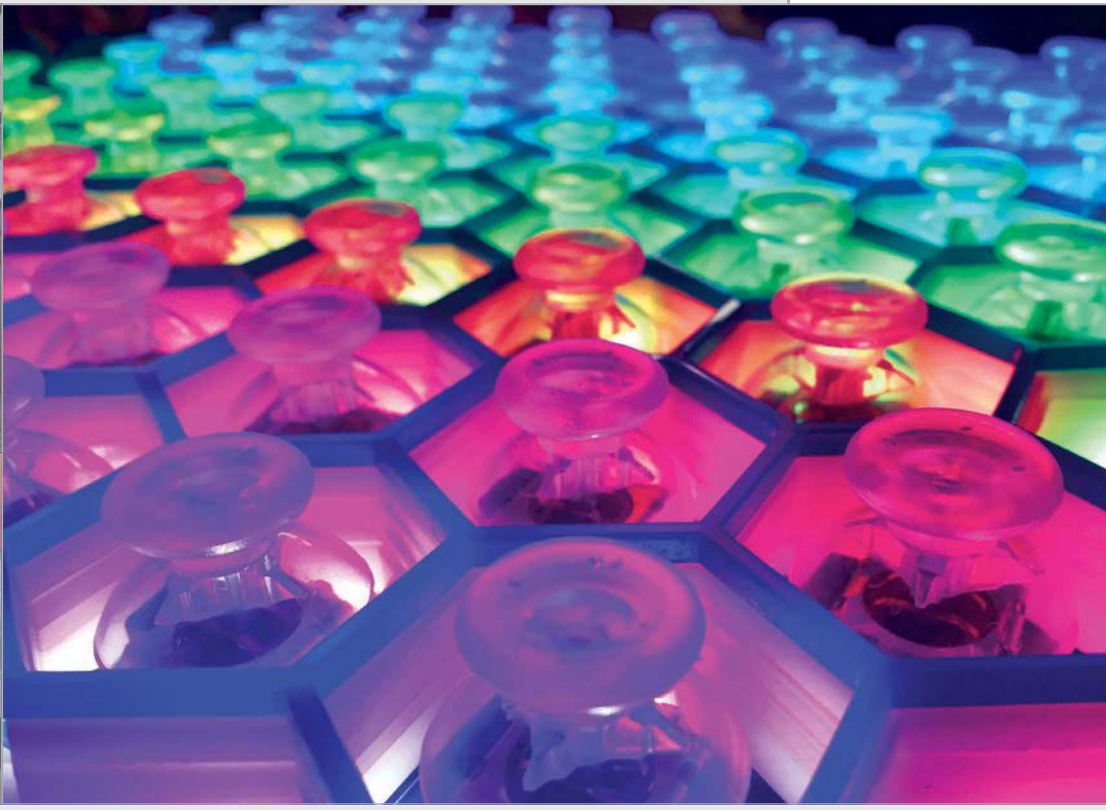
"The pitch is controlled by the joystick the user chose to push," says Dave. "One axis of the joystick controls the kind of waveform produced. It fades from a bright trumpet-like sawtooth wave to a darker triangle wave that sounds like a bell." The other axis of motion controls how flat or wide the note sounds by making a small tuning difference between the pair of oscillators that produces the note.

"All of that expressive potential is represented in the group of blocks in PureData," Dave tells us. "There are six of those groups in the patch, meaning the Joytone can play up to six notes simultaneously."

### Playing the Joytone

While the idea behind the Joytone is complex, playing it is surprisingly easy. "My friend who helped me build the first one in school is a very talented musician," affirms Dave,"and the first time we got it working, he played with a couple of the joysticks, then paused and thought for a second, then immediately played a Bach fugue he was familiar with. It was an awesome moment of success after a string of very long nights."

Playing the Joytone is remarkably straightforward. The joysticks are played with a single finger. "It's easy to hold your hands as if you were typing on a keyboard," advises Dave, "and play many notes at once. The joysticks are pretty close together, so the player has access to a large musical range with pretty limited hand motion."

## Integrating the Raspberry Pi

This build is the second iteration of Joytone and is the one that introduces the Raspberry Pi to the design. "The first Joytone had to function though a nightmarish rat's nest of wires inside," Dave recalls.

"In the two-brain design I used for the Joytone, the Arduino does all the analog-to-digital conversion and the Raspberry Pi does all the audio synthesis. I like to use it on a Raspberry Pi because I can just tuck it inside the enclosure and focus completely on the instrument. I taught myself a little bit about PCB routing, and with the help of some friends they came out perfectly," he says.

## Showing off the Joytone

Part of the inspiration for building a more powerful iteration of the Joytone was the Toronto International Film Festival (TIFF). Dave was asked to provide Joytone for part of the digiPlaySpace exhibition at the show.

"I had to design it to run continuously and withstand thousands of visitors over a period of six weeks," Dave says, "so I did some research at the Exploratorium in San Francisco to see how they build their exhibits.

"[The Joytone] saw about 16,000 visitors this year and I've been showing it around to my friends. Everyone seems to have a good time with it, and I'm curious to see what could happen with some serious practising. My favourite moment so far was a write-up from a six-year-old blogger who visited the TIFF exhibit.

"I'm glad that the Joytone is back from the TIFF exhibition because I can practise on it now. I only finished it a day or two before I had to ship it out! Even with the relatively short amount of practice time I've had, I can play almost any scale or chord with ease."

Despite the apparent complexity, Dave claims that the Joytone was pretty simple to build. It was "extremely tedious because there are so many joysticks."

It isn't a cheap project to make, either: "The parts for the Joytone cost about $600, most of which is for the lights and joysticks." Dave also spent an additional $450 on the custom PCBs, though he tells us he's got enough leftover parts to build another instrument.

"I have lots of plans for other musical inventions," he says. "I'm going to iterate on the joystick idea but investigate some other form factors. I'd also like to make something more compact and portable."

## Making musical instruments

If you are interested in following in Dave's footsteps, it's worth knowing that you don't have to build a device as complex as the Joytone. "Anyone could wire up a joystick to an Arduino and start sending MIDI notes to their computer," says Dave, and "you could make a simple musical instrument in an hour or two."

Budding musical creators should just "go for it," he tells us. "There's a great community of people building new musical instruments, and lots of inspiring work already documented. PureData is a wonderful tool to experiment with quickly, and the Raspberry Pi is a friendly platform for musical inventions. Learning a little music theory is also really helpful."

**Above** Pushing a stick up controls the kind of waveform produced; left to right controls how flat or wide the note is

**Top left** The Joytone is a spectacular-looking modern musical instrument

**Left** A sturdy wooden case is used to house all of the components

# Extra Lives

The Raspberry Pi is helping to revive some age-old gaming classics. **David Crookes** investigates...

It may not feel like it today, but the computing world hasn't always been dominated by the PC and the Mac. Years before they became ubiquitous, scores of home machines left an indelible mark on computing, from the ZX Spectrum and BBC Micro in the UK, to the US-made Commodore 64. The Amstrad CPC did well in France, and that's before you get to the worldwide popularity of the Atari ST and Commodore Amiga.

Companies big and small entered the home computer market in droves, most of them allowing users to "get under the hood", just like the Raspberry Pi today. That made it possible for people to get their hands dirty with code and see what they could make their computers do. One of the great by-products of this was a flourishing games industry.

Programmers spent time in their bedrooms hunched over their keyboards as they contributed to the thousands upon thousands of titles that were released. Many of them subsequently made the switch to the flourishing home console market, which saw Nintendo and Sega go head-to-head with lots of wonderful games machines. But compatibility problems mean that playing those games today without having the original hardware is impossible. Impossible without using an emulator, that is.

And that's where the Raspberry Pi comes in. Emulators allow modern-day computers to behave like their predecessors, and you will be pleased to know that the Pi has a great many of them. It is possible for your Pi to pretend it is a Speccy or a C64. It can even mimic older consoles and arcade machines. With the right software and access to some gaming ROMs online, you can turn your tiny machine into a fully fledged retro games console. It's time to have some fun...

## DOS

WHAT: Before Windows, there was DOS. Short for "disk operating system", it was driven by command prompts rather than the graphical user interfaces to which most computer users are accustomed today.

# Beneath a Steel Sky

### Stats

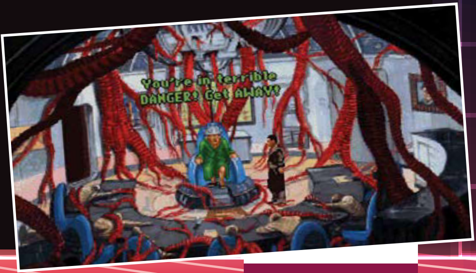
Developer: Revolution Software
Released: 1994
Formats: DOS, Amiga, Amiga CD32
Emulator: ScummVM

Even though *Beneath a Steel Sky* is celebrating its 21st anniversary, this cyberpunk point-and-click adventure will certainly enthral you today. Inspired by the *Mad Max* and *Blade Runner* films, it tells the story of Robert Foster, who is taken under the wing of indigenous Australians following a plane crash, only to later see his adopted family slaughtered by the army. Foster is flown to Union City, but he suffers a second crash which allows him to escape and search for answers with his robot pal, Joey. The game is a stunning tale of hope amid oppression and it will keep you engrossed for hours.

Players are expected to solve a series of intricate puzzles, each one developed to drive the narrative forward. As you look for clues and search for items that can be used to create often mind-bending solutions, you are introduced to a host of characters. By engaging in interactive conversation with them, the dialogue enriches the story and helps to unravel the mystery. But just as you think you may know what is going on, the game throws a major curveball and it will have you begging for more.

Much of the game is laden with humour, but there is also lots of drama, with Robert's backstory brought to the fore through the little quips between him and Joey. The game was made using Revolution Software's Virtual Theatre game engine, a "rival" of sorts to LucasArts' SCUMM engine. It helped lend the game a realistic edge and allowed the non-player characters to wander around rather than remaining fixed in one spot.

Revolution released the DOS-based version of *Beneath a Steel Sky* as freeware a few years ago, so it means you can legally play it on your Raspberry Pi for free. There is no doubt that you should.



### Why give BASS away?

The game's director, Charles Cecil, discusses his decision to make *Beneath a Steel Sky* a freebie.

"*BASS* was originally published for DOS, way back in 1993, so when Windows 98 stopped supporting DOS a few years later, it looked set for oblivion. We were approached by a group calling themselves ScummVM, looking to resurrect point-and-click adventures by converting the source code to run cross-platform. We duly provided them with all the assets.

"One of the leads, Joost Peters, later came to work for Revolution and is now our technical director. We felt since we were unable to sell the game, that we should give it away for free. I wish I could claim this was a stroke of marketing genius, but actually it was more about doing what we felt was fair. It has ensured the game is very widely played, particularly on Linux-based computers."

### Keep clicking

The ScummVM emulator allows you to play a host of point-and-click adventure games. It was originally based on the SCUMM engine created by LucasArts' Ron Gilbert and Aric Wilmunder. Learn more at **scummvm.org**. Here is the pick of the bunch:

#### The Secret of Monkey Island

Introducing hapless Guybrush Threepwood to the gaming world, this swashbuckling 1990 release was chock full of great humour, mind-bending puzzles, and sword fights that were a battle of sharp, insulting wit. It was no surprise that *Monkey Island* spun into an award-winning four-game series that is still loved today.

#### Broken Sword

Released a fair few years before Dan Brown's *The Da Vinci Code*, this game throws players deep into the legend of the Knights Templar. Starring American hero George Stobbart and French journalist Nico Collard, it takes players on a fantastic journey across France, Ireland, Syria, Spain, and Scotland.

#### King's Quest

*King's Quest* was a popular graphic adventure designed by Robert Williams, the co-founder of Sierra On-Line, using the Adventure Game Interpreter engine. The series followed the saga of the Kingdom of Daventry's royal family. Watch out for some memorable guest appearances from the likes of Dracula and Little Red Riding Hood.

#### GAMES GAMES GAMES

In order to play games on your Pi, you need to get hold of the gaming ROMs. It's worth checking out Emuparadise, but read the legal boxout on page 40 first.

# 8-bit Gaming

## Spectrum

It may have had just 16kB of memory to start with, and it may have run its games off slow-loading tapes, but the ZX Spectrum sold five million units. Its popularity ensured it was blessed with a wealth of brilliant games.

## Matthew Smith

Matthew Smith is revered today as one of the undisputed genius programmers of the ZX Spectrum. If you are emulating the Spectrum, you must play these two games…

### Manic Miner

*Manic Miner* was one of the first platform titles, its colourful graphics and fiendishly difficult gameplay catching the imagination. It had in-game music, being the first Spectrum game to do so, and animated toilets too! Try it to see how easy games really have become.

### Jet Set Willy

As the sequel to *Manic Miner*, this game reintroduced hero Willy, who by this point was a rich man tasked with cleaning up his mansion following a riotous party. More of an adventure than the first game, with a fresh structure, *Jet Set Willy* is every bit as addictive.

## Commodore 64

The Commodore 64 was the American rival to the British ZX Spectrum and it caused many a playground row to erupt. The C64 boasted more colours, but it also had amazing sound thanks to its SID chip. If digital music is your thing, then emulating C64 games will give you great pleasure.

## The classics

### Lemmings

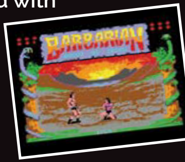
Made by DMA Design, which would go on to produce *Grand Theft Auto*, *Lemmings* became one of the best puzzle games ever made when it was released in 1991. Guide the tiny humanoid creatures to safety by putting a good number of them to good use: building bridges and digging tunnels, and by equipping some with umbrellas so they can float gently to the ground. Listen out for the cries of "oh no!" as lemmings explode in a shower of pixels.

### Rainbow Islands

*Rainbow Islands* was not only cute and addictive, but very popular too. Players need to make their way up the screen to avoid rising sea levels. This is achieved by laying down rainbows, allowing bad guys to be squished and giving you leverage to higher platforms. The game has four rounds of gameplay per island, after which you move on to another. Each time, you are treated to the (eventually annoying) strains of *Somewhere Over the Rainbow*.

### Barbarian: The Ultimate Warrior

Featuring decapitated heads and lots of blood and gore, *Barbarian* is a brutal beat-'em-up that keeps players nimbly moving their fingers in order to perform one of the 16 available moves. There is a follow-up to this game which you may also want to try, but before you do, give this one a go in two-player mode. Going head-to-head with a pal makes the game truly sing, providing a lot of fun in the process.



### Impossible Mission

As if to underline the excellence of the Commodore 64's sound capabilities, *Impossible Mission* starts off with the line, "Another Visitor. Stay a while. Stay Forever", delivered in a creepy voice that nevertheless thrilled gamers at the time. Exploring the underground lair of Professor Elvin Atombender and seeking clues in order to turn off a bomb, players were also amazed at the fluid animation. *Impossible Mission* was ground-breaking at the time and it remains a lot of fun today.

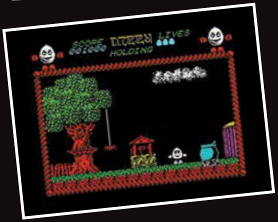
## LEGAL WORRIES

Using emulators to play games has long been a legal grey area. For some games publishers, the answer is clear: don't do it. For instance, Codemasters asked the website World of Spectrum to remove its retro games, while Nintendo states its position clearly at **nintendo.com/corp/legal.jsp** ("The introduction of emulators created to play illegally copied Nintendo software represents the greatest threat to date to the intellectual property rights of video game developers," it says). In general, the advice has been to use emulators to play games that you already own (so you can emulate a Spectrum game if you still have the cassette copy). In addition, you should certainly not pirate games or seek to profit from them in any way.

## OTHER CLASSICS

Alter Ego
Archon
Arkanoid: Revenge of D'oh
Atomino
Blue Max
Boulder Dash
Bubble Bobble
Buggy Boy
California Games
Creatures
Cybernoid
Dragon Wars
Elite
Emlyn Hughes Int. Soccer
Enforcer
Grand Prix Circuit
Head Over Heels
IK+
Jumpman
Laser Squad
Last Ninja 2
Law of the West
Leaderboard
Little Computer People
Lode Runner
Maniac Mansion
Mayhem in Monster Land
MicroProse Soccer
Midnight Resistance
Myth
Nebulus
Neuromancer
Paradroid
Park Patrol
Pirates!
Pitstop II
Platoon
Pool of Radiance
Prince of Persia
Project Firestart
Rick Dangerous
Silent Service
Skate or Die!
Space Taxi
Spherical
Spy vs Spy
Starquake
Steel Thunder
Stix
Stunt Car Racer
The Bard's Tale
The Sentinel
The Way of the Exploding Fist
Turrican
Ultima V
Uridium
Winter Games
Wizball
Yie Ar King Fu
Zak McKracken and the Alien Mindbenders

## Dizzy The Ultimate Cartoon Adventure



### Stats

Developer: The Oliver Twins
Released: 1987
Formats: Amstrad, ZX Spectrum,Commodore 64
Genre: Arcade-adventure

Once upon a time, there was a little egg-shaped character who lived in a fantasy world with his family and battled against an evil wizard. His name was Dizzy, and he went on to star in a multitude of adventures and a fair few spin-offs. Created by the Oliver Twins, Philip and Andrew, he became synonymous with British gaming in the 1980s and early 1990s, with best-selling games being made available on the Spectrum, Commodore 64, Amstrad CPC, Amiga, Atari ST, and NES.

The original game was made on the Spectrum and ported to the CPC, before the C64 version was made. "We only had 32K of RAM to store the whole game," says Philip Oliver, "plus the processor was very slow, meaning plotting graphics to the screen was extremely limited." The game involves collecting items to solve subsequent puzzles (using a grease gun to get a cart moving in a mine, for instance). If you play later games in the series, you will be introduced to Dizzy's Yolkfolk and engage in some chit-chat. There are some funny quirks involved, too: try kicking the Dozy's deckchair a few times in *Fantasy World Dizzy* to see what we mean.

### Why I love the Pi

"The Pi is a hobby computer designed to inspire people to code. We've always felt 2D games should be the first thing to attempt to write, and not scrolling, either, as that adds a lot of complexity. People shouldn't try 3D until they've really got the hang of 2D games.

Since retro games often means simple, fun, 2D, and there quite a few that don't require scrolling or many elements to make them fun: they make ideal material for Raspberry Pi."

**Philip Oliver**

## Nintendo

Harking back to the 8-bit days of Nintendo, notably the NES and the Game Boy, is arguably to take a trip down into the origins of the modern-day gaming scene. Nintendo's machines were blessed with amazingly well-designed games.

### Super Mario Bros

Often referred to as the greatest game of all time, *Super Mario Bros* is a side-scrolling 2D platformer which came bundled with the NES console. Set over a large playing area, the game has impeccable level design, lots of secrets to uncover, plus lovely graphics and music. It also popularised power-ups, such as Fire Flower, which allowed Mario to grow tall and hurl fireballs.

### Tetris

The Game Boy may have been a rather primitive handheld machine – it had a green screen and no backlight, and it paled in comparison to the Atari Lynx – but it had a killer app: *Tetris*. Designed by Russian programmer Alexey Pajitnov, this took the world by storm, with gamers transfixed by the falling blocks and the urge to tidy them away in neat rows.

### Hunt for the Red October

Many retro games have been based upon film franchises, usually published by Ocean Software and U.S. Gold. This game was based on the Cold War thriller that, in turn, was based on Tom Clancy's 1984 debut novel.

### The Yellow Submarine

One of the joys of retro gaming is uncovering titles that few people will have heard of. *The Yellow Submarine* was released in 1986 and it bypassed many a games reviewer at the time. Why not see why?

### Green Beret

Imagine Software went bust in the most spectacular fashion in the mid-1980s (just as documentary TV cameras were rolling), but it squeezed out this brilliant, fast, varied, horizontally scrolling shoot-'em-up before it did.

### Blue Thunder

With the aim being to fly a chopper off a ship and avoid being shot at as you travelled above islands, *Blue Thunder* not only looked good but behaved rather naturally too, with lovely sound effects. Get used to the controls and you'll love it.

## INTERVIEW

### Bare-metal Doom

First-year students at Imperial College London have produced a bare-metal partial clone of *Doom* for the first-generation Raspberry Pi in 9,800 lines (see hackaday.com/tag/bare-metal). Csongor Kiss tells us more about it…

MagPi: **Where did the idea come from?**
Csongor Kiss: As part of our first-year project, we were challenged to make whatever we could, limited by the constraint that it had to work on a Raspberry Pi, and be coded in bare-metal assembly. It turned out to be not so large a constraint.

MagPi: **What did the Pi bring to the table?**
Csongor Kiss: The Pi is more powerful than the machines running games like *Doom* on release, which let us use a higher screen resolution and better-quality textures than those of the original.

MagPi: **What was the process of making the conversion?**
Csongor Kiss: The only thing sourced from the original game were the textures. Everything else was built from scratch. We used the *Doom* wiki for inspiration for things like level design and gameplay, but coded it from the ground up, just as you would for any other project.

## PORTING A COIN-OP

Many arcade games were ported to home computers and consoles. Things didn't always go according to plan, however, as Ste Pickford, who worked on *Ghosts 'n Goblins*, recalls.

"I was given a pile of photographs taken from the screen of the arcade machine by the publisher, Elite, and asked to 'do the graphics' from that. I had no sense of the layout of the levels or anything. So I went to Stockport arcade with a sketchbook and stood next to its *Ghosts 'n Goblins* machine, watching people playing, and sketching the level from what I was seeing. People kept dying and starting over, but I had a decent sketch of the first level and needed someone to get to level two, but eventually the arcade owner spotted me and accused me of 'ripping off his machines'. I was barred.

"Programmer Nigel Alderton arranged to drive me to Elite's offices in Walsall to play their machine instead. It had been set to free play with infinite continues, but it took us hours and hours to get to the end. We finished around 5am and drove home to Stockport. But I was only 15 at the time, and it hadn't occurred to me to phone home. My parents were frantic and on the verge of reporting me as a missing person. Still, I had the level layouts, so everything was fine."

## Quake III Arena

### Stats

Developer: id Software
Released: 1999
Formats: Windows, Linux, OS X, Dreamcast, PS2
Genre: First-person shooter

If you love *Doom*, you'll adore *Quake III Arena*. It's a frantic, no-nonsense first-person arcade shooter which runs well on the Pi, pitting players against 30 artificial intelligence bots in a game that eschews a plot in favour of all-out fun. The design is deliberately minimalist, yet it looks amazing and it plays at a fast speed, giving gamers a good choice of weapons – check out the rocket launchers, shotguns, and the melting plasma gun – while thinking about ease of movement around the playing area. Accelerator Pads and Bounce Pads take the place of lifts and ladders, enabling a greater level of fluidity in the action.

One of the game's designers, Graeme Devine says he is thrilled the Pi has the power to open up *Quake III* to a new audience. "I have a Pi sitting right here on my desk," he tells us. "It dizzies me that something so small can do so much, and the fact that people can look through the *Quake III* source just opens up so much possibility all over the world. It feels like the games industry can explode again."

## ARCADE

### Best of the arcade

Standing for "Multiple Arcade Machine Emulator", MAME is a way of playing a host of coin-op arcade games, many of which were seen as cutting-edge in their day…

### Street Fighter II: The World Warrior

When *Street Fighter II* was released into the arcades in 1991, it sparked a huge boom in fighting games. Not only did it truly establish some well-known virtual fighters, including Ryu, Chun-Li, and Guile, this deeply strategic title also showed the importance in giving players special moves unique to each character.

### Metal Slug

For those who love run-and-gun platfomers, the ***Metal Slug*** series is nigh on perfect. The original has stood the test of time, giving players control of an eight-way joystick and letting them loose on six levels of shooting, bombing, and jumping. The game, while basic in premise, helped to establish the Neo Geo MVS arcade platform.

### Ghosts 'n Goblins

Capcom's *Ghosts 'n Goblins* series was a very hard and yet rewarding game, thanks to impeccable level design that makes it fun to play, regardless of how many times you are nobbled. Players assumed the role of a knight called Arthur in a zombie-strewn land, and you only needed two hits to lose a life.

## Chocolate Doom

### Stats

Developer: Simon "Fraggle" Howard
Released: 2005
Formats: DOS, Raspberry Pi
Genre: First-person shooter

Ported to the Raspberry Pi, *Chocolate Doom* takes the source code of the original game and replicates it faithfully on our favourite computer. The actual source port has been around since 2005, thanks to developer Simon "Fraggle" Howard, and it exists with all of the original engine's bugs and quirks for the most authentic experience of all. The Pi version supports upscaling to 1920×1080.

## Lakka

### Create a Pi console with Lakka

Jean-André Santoni has put together a lightweight Linux distribution called Lakka (**lakka.tv**), which lets you turn your Pi into a versatile games console.

MagPi: **Where did the idea for Lakka come from?**
Jean-André Santoni: I bought a Raspberry Pi when it [first] came out… and I discovered RetroArch. I loved the fact it was working without big dependencies like X11, and centralising the configuration of all emulators with that libretro API was a bright idea.

MagPi: **There is a certain familiarity to the UI. Why did you choose this?**
Jean-André Santoni: Ah yes, the PS3 XMB-like interface. I chose this because I wanted an intuitive interface. By looking at just one screenshot of the interface, the user can understand what we are offering: a list of controllers is displayed horizontally – they represent each emulated system – with a list of cartridges vertically, representing the games. Also, this kind of interface is not very difficult to implement, and is extensible.

MagPi: **Does the Pi make for the perfect retro machine?**
Jean-André Santoni: I like the fact that the bootloader is silent and fast. The resolution of the screen is easily detected. There are no major issues with the GPU driver. I would say that the Pi is a good compromise: its price, its documentation, and its overall quality makes it a serious choice for beginners.

MagPi: **What should gamers using a Pi bear in mind when using Lakka?**
Jean-André Santoni: Lakka is not yet finished, so people should expect some bugs in the interface. Configuring the Wi-Fi or the Bluetooth controllers will work only by using the command line. We maintain a page on our wiki explaining what is the best hardware to use, though. So far, the best controllers for Lakka are the official Xbox 360 ones.

## RetroPie

### Get your teeth into the classics

RetroPie supports a huge number of computers and consoles, including the Amiga, Apple II, Atari 2600, Amstrad CPC, C64, Game Boy, Game Gear, MAME, NES, N64, ScummVM, PS1, SNES, and Spectrum. We talk to its creator, Florian Müller…

MagPi: **Where did the idea for RetroPie come from?**
Florian Müller: I grew up with a whole bunch of 8-bit and 16-bit video game consoles. Home computers such as the Commodore 64, the Amstrad CPC, the Amiga 500, and the first IBM PCs were also parts of my childhood. The idea for RetroPie was born out of nostalgia.

MagPi: **How has the project evolved?**
Florian Müller: It started with a rudimentary Bash script that automated the installation of some emulators, and set up folders and configuration. This was the birth of the RetroPie-Setup script, which is still the core of RetroPie. One goal from the beginning was to achieve a keyboardless system. A big step towards that was the addition of EmulationStation, a graphical front-end developed by some great enthusiasts of RetroPie. By this time, more and more systems were being added to RetroPie, and supplementary functions such as a USB copy service, game metadata scrapers, splash screens, and configuration possibilities were included. There is a wiki and a forum, too.

MagPi: **What is the strength of RetroPie?**
Florian Müller: The volunteers who enjoy contributing in various ways. We have source-code maintainers and contributors, people who help each other in the forum and forum moderators, wiki maintainers, and people who post issues or feature proposals. There are people writing blog posts about specific features of RetroPie, and there are video tutorials. RetroPie truly is an open-source project and it evolves by the help of the community.

For more information about RetroPie, and SD card images, visit: **blog.petrockblock.com/retropie**

### RETROPIE INGREDIENTS

- A Raspberry Pi (preferably version 2 or model B+)
- A case
- A power supply (the official power supply does a great job)
- An SD card (preferably 8GB or larger)
- An HDMI cable
- A USB keyboard (at least for the configuration)
- And, of course, the current RetroPie image

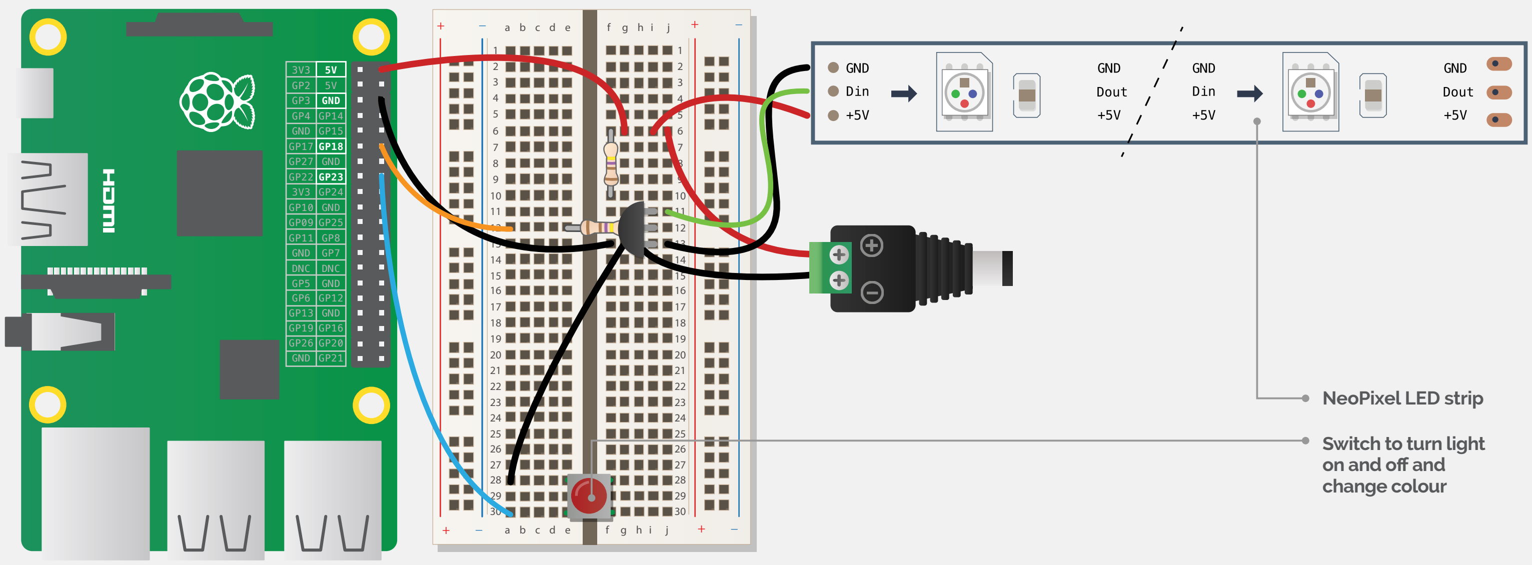
Go to the **bit.ly/1ecWcg** download page and download the correct image for your Pi model. Extract and copy the image to an SD card. If you have an SD card larger than 4GB, expand the root file system: press F4, run **`sudo raspi-config`** and choose "expand root fs". Restart the Pi. Search for game ROMs and copy them to a USB stick.

EVERYDAY ENGINEERING PART 6

SIMON MONK

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

**simonmonk.org**
**monkmakes.com**



# MOODLIGHT

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

**You'll Need**

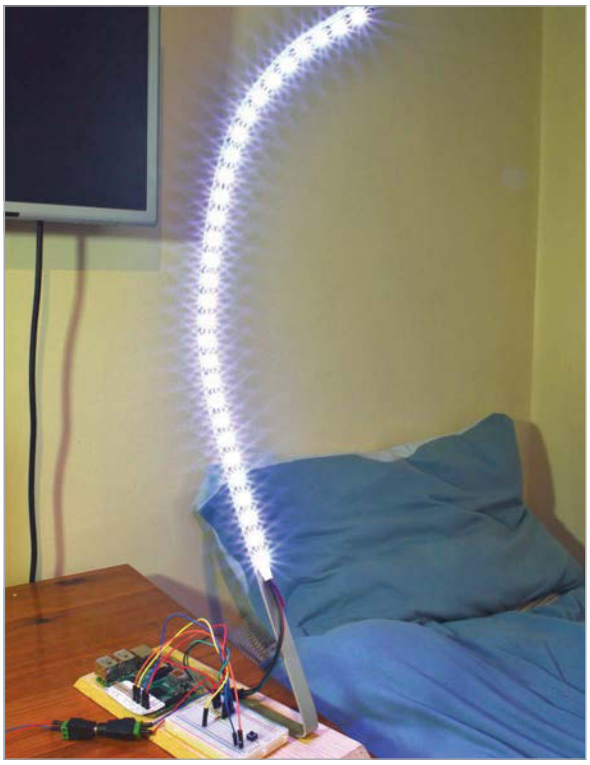
- Half-size breadboard
- 0.5m of 60 LED/m 5V WS2812B LED strip (eBay / Adafruit)
- 2× 470Ω resistor
- 2N7000 MOSFET transistor (CPC: SC06951)
- Tactile push switch
- 7× male-to-female jumper wires
- 3× male-to-male jumper wires
- DC barrel socket to screw adaptor (CPC: CN19646)
- 5V DC @ 3A Power supply

LED strips can be used both as displays and for lighting projects like this one. They look great and have a self-adhesive backing that means they can be stuck to just about anything. They also work well with a Raspberry Pi, requiring just one GPIO pin to control them.

In this project, 30 LEDs are used to provide light of different colours. You can change the colour and brightness of the lamp using a push button, as well as turning the light on and off. A long press of the push button causes the lamp to pick a colour based on the time of day. So, you can have a nice restful soft orange light at bedtime to get you in the mood for sleep!

As you'll see from the list of required components on the left, this project uses a breadboard to provide a solid base for the push switch, and for the transistor and associated resistors that are used to shift the signal level to the 5V needed by the display.

When you are buying your LED strip, make sure that it uses the WS2812B LEDs. They are available in a range of lengths and what's more, if it's not the length you need, then you can just cut some off with a pair of scissors. They are generally available as 30 or 60 LEDs per metre. We used 60/m in this project. You can buy generic reels of LED strip on eBay, and


Above The mood light

**⚠ WARNING!**

The 5V power supply is going to be supplying power to both the Raspberry Pi and the LED strip, both of which will be rapidly destroyed if you accidentally connect a 12V or 9V power supply.

Adafruit also sells these LED strips under the brand name of NeoPixels.

When all 30 LEDs are on and at maximum brightness, they consume almost 1.5A; add that to the 0.5A or more of a Raspberry Pi and the total current consumption will be around 2A. So, it would be wise to use a 3A power supply.

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

## LED strips (NeoPixels)

LED strips are made up of what are basically LED chips – red, green and blue LEDs and a controller integrated circuit, all in one neat little package. These are placed at regular intervals along a thin, flexible, self-adhesive PCB. Both the power and control pin of each NeoPixel are connected to the next. You will notice little arrows on the strip: the LEDs are connected together, one to the next, in the direction of the arrow. The data is blasted out as a stream of ones and zeros at 800,000 bits per second. When the stream of bits stops for a moment, the LED pixels all update themselves.

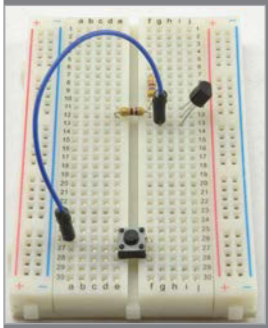
## Making your mood light

You could really go to town on the physical design of this project. We used a block of wood that was lying around and a strip of aluminium from the local hardware store that we bent into the shape of the number 2. But before you fix things down, it's always best to get the project working on your bench.

Above LED strips (NeoPixels)

## BUILDING THE PROJECT

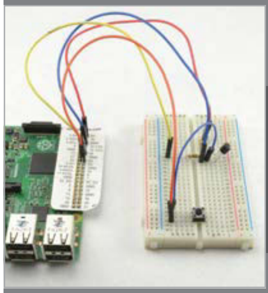
This is a really easy project to make. There is no soldering to be done and relatively few wires to connect up...



### >STEP-01
**Wire up the breadboard**

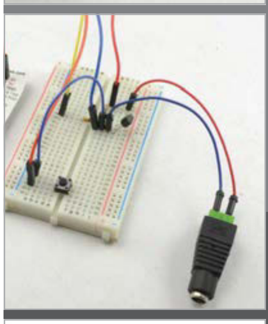
Fit the transistor, resistors and switch onto the breadboard in the positions shown. Make sure that the transistor is the correct way around, with the flat side towards the right of the breadboard. The switch will only fit the correct way around across the two banks of breadboard holes.

Link what will become the GND row of the breadboard to the top switch terminal using a male-to-male jumper wire, or, if you prefer, with a short length of solid core wire that will lie flat to the breadboard.



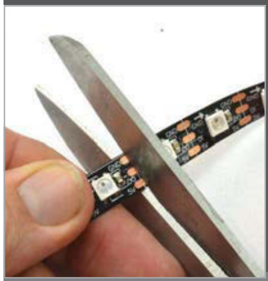
### >STEP-02
**Connect the breadboard to the Raspberry Pi**

Use the male-to-female jumper wires to connect the breadboard to the Raspberry Pi GPIO pins.



### >STEP-03
**Attach the power connector**

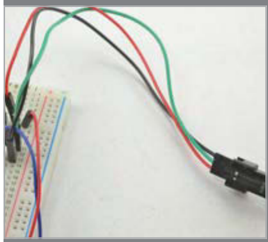
Fit male-to-male header wires to the screw terminals of the DC barrel adaptor. Ideally, use a red wire for the positive connection and blue or black for the negative. The other end of the header pins fit onto the breadboard as shown.



### >STEP-04
**Cut the LED strip**

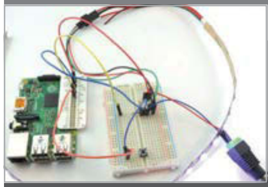
If your LED strip is longer than 30 LEDs, then you will need to cut it to size. If you are thinking of using a different number of LEDs you can, but you will need to adjust the program and also make sure that the power supply you use has a big enough current rating.

Count the LEDs in the direction of the arrow and cut the strip at the line marked between the LEDs. Do not throw away the rest of the strip: if you solder wires to the solder pads, you can use it for another project.



### >STEP-05
**Connect the LED strip to the breadboard**

Use male-to-female jumper wires to connect the LED strip plug to the breadboard. The LED strip may have a separate pair of power wires. You can ignore these or cut them off, as the power connections to the plug are more convenient. The female ends of the jumper wires may fit into the plug better one way around than the other.



### >STEP-06
**Testing the lamp**

The project is now ready for testing, but first it's time to set up the software on your Raspberry Pi.

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 Python library that, in turn, uses a C library that manages the high-speed communication. The software used here is based on this Adafruit tutorial: **learn.adafruit.com/neopixels-on-raspberry-pi**.

However, the C library used in the Adafruit tutorial is (at the time of writing) not compatible with the Raspberry Pi 2. Fortunately for any Pi 2 owners, Richard Hurst has made a version of the software that does work with Raspberry Pi 2. We have also tested it on a B+ and an original B.

> “'Colors' is where you can alter the colours of light that the lamp will cycle around”

Start by making sure that your package manager is up to date, using the command:

```
sudo apt-get update
```

Next, install the packages and library software that are needed, using:

```
sudo apt-get install build-essential python-dev git scons swig
```

The next step is where the directions diverge from the Adafruit tutorial, because you need to fetch the version of the software that has been fixed for Raspberry Pi 2. If you don't have a Raspberry Pi 2, you can just stick with the Adafruit tutorial if you prefer.

Fetch the modified NeoPixel code from GitHub using the command:

```
git clone https://github.com/richardghirst/rpi_ws281x.git
```

Change directory into the software that you just fetched from GitHub, using:

```
cd rpi_ws281x
```

...and then build the C code, using the command:

```
scons
```

When the C is compiled, you need to install the Python library that interfaces with the fast C code:

```
cd python
sudo python setup.py install
```

Finally, you can download the Python program for this project, using the commands:

```
cd /home/pi
git clone https://github.com/simonmonk/pi_magazine.git
```

This command will actually bring down the code for all the projects in this *MagPi* series, so if you have already issued this command for one of the earlier articles, change directory to **pi_magazine** and run the following command to update your directory with this project (06_mood_light).

```
git pull
```

## How the code works

The Python code for this project is pretty straightforward. All the hard work takes place in the libraries.

If you are interested in how the code works, load it up into an editor while we go through it.

The first section of code includes all of the libraries you need. This includes the **neopixel** library and also the **RPi.GPIO** library that is needed to detect whether the switch has been pressed.

The next section contains a load of variables that are used to configure the **neopixel** library so that it agrees with the actual hardware. If you wanted to use more or less LEDs, then change **LED_COUNT** to the number of LEDs in your strip. Another value that you might need to change is **LED_FREQ_HZ**. Some LED strips operate at 400,000Hz rather than the more usual 800,000Hz.

The section labelled 'Colors' is where you can alter the colours of light that the lamp will cycle around. The first one (**NO_COLOR**) needs to be unchanged or you won't be able to turn the lamp off using the button, but you can change the values of the other colours as you like. Remember to keep the variable **colors** up to date, adding any extra colours to the end of the list. The second, third and fourth colours (**MORNING_COLOR, DAY_COLOR**, and **NIGHT_COLOR**) are also special, because it is between these three colours that a choice will be made, based on the time of day when you hold down the button for more than half a second.

The **color_index** variable is used to indicate the index position 0 to 6 of the currently selected colour.

After this, there is a section to set up GPIO pin 23 as the switch input, and then we come to the start of the functions.

The function **set_all** will set every LED in the strip to the colour specified; **set_color_index_from_time** will pick the colour index 1, 2, or 3, based on whether it's morning, daytime, or evening.

Then the display is initialised and all the LED colours set to **colors[0]**, which means off, so that initially the lamp is off.

The main program loop is contained in a **try: finally:** block so that the GPIO pins will be tidied up when the program exits. This main loop only has to do anything when the switch is pressed. When this happens, the current time is stored in the variable **down_time** so that the length of the button press can later be calculated. There is then a sleep of 0.2 of a second to give the switch contacts time to settle. This is called 'debouncing' and without it, a single button press can register as multiple presses. The **while** loop that follows this ensures that nothing further happens until either the button is released or half a second (**LONG_PRESS_THRESHOLD**) has elapsed.

The duration of the button press is then calculated and if it's a long one, **set_color_index_from_time** is called to set the colour automatically; otherwise, 1 is added to the **color_index** variable that is returned to

0 when it gets to the end of the **colors** list. The LEDs are then set to the new colour.

The final **while** loop ensures that if the switch is still pressed after a long press, no further changes will occur until the button has been released.

## Using your mood light

Start the mood light program with the command:

```
sudo python mood_light.py
```

Initially, the LEDs should all be off. Actually, on our version, the first LED illuminates green, but we put this down to a bug in the **neopixel** library, or a problem with the transistor level shifter not quite operating quickly enough. In any case, by happy accident, this makes quite a handy power-on indicator.

Press the button and you should find that the lamp cycles through the various colours until it gets round to the 'off' setting again.

Once you are happy that everything is working, temporarily disconnect the LED strip and attach it to whatever you want to use to make the illuminating part of your lamp. At this point, you can also box up your design.

If you want the time feature of your Raspberry Pi to stay accurate, then you will need to give the Pi either a WiFi or wired network connection. If you are adding a network connection, then you may want to provide a web interface to the lamp so that you can turn it on and off from a browser on your phone. For some ideas on how you could do that, you may want to look at the 'Web Door Lock' article in issue 32.

Using a Raspberry Pi to control a lamp is definitely overkill. The Pi will be on all the time, whether the lamp is on or not. A Raspberry Pi 2 or B+ both use less power than a B and only actually use perhaps 1W of power most of the time. This amounts to approximately 9kW hours of electricity a year, which may cost £1 or £2.

Having a keyboard, mouse and monitor attached to your lamp is fine while you are constructing it, but it would be better to have the program start automatically when the Raspberry Pi first starts up. To do this, run the following command to make the program executable.

```
sudo chmod +x mood_light.py
```

Then, edit the file **/etc/rc.local** using the command:

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

Add the following line after the first block of comment lines that begin with '#'.

```
sudo /home/pi/pi_magazine/06_mood_light/
mood_light.py &
```

Restart your Raspberry Pi and this time the lamp program should start up automatically.

**NEXT MONTH**

In the next project in this series, we'll create a fridge monitor that uses IFTTT to notify you when your fridge is raided.

## Mood_light.py

**Language**
>PYTHON

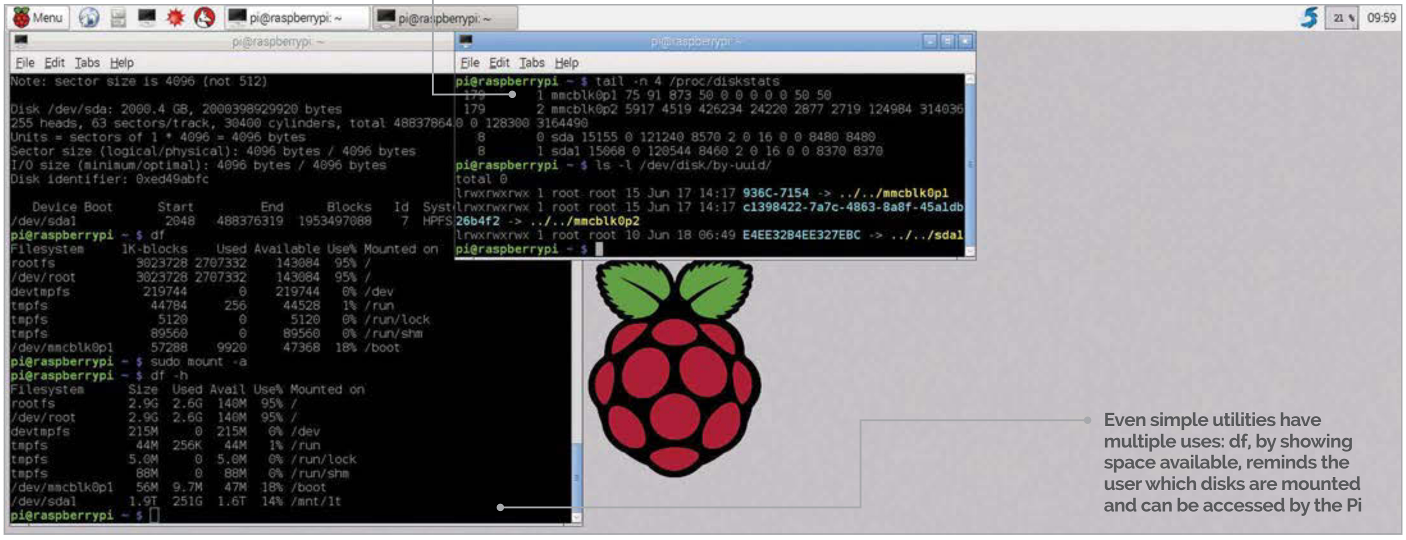
**DOWNLOAD:**
github.com/simonmonk/pi_magazine/tree/master/06_mood_light

```
import time
from neopixel import *
import RPi.GPIO as GPIO
import datetime

# LED strip
LED_COUNT      = 30      # Number of LED pixels
LED_PIN        = 18     # GPIO pin connected to 
the pixels (must support PWM!)
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 5       # DMA channel to use for generating signal (try 5)
LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = True    # True to invert the signal (when using NPN 
transistor level shift)

NO_COLOR = Color(0, 0, 0)
MORNING_COLOR = Color(190, 190, 255)
DAY_COLOR = Color(255, 255, 255)
NIGHT_COLOR = Color(255, 190, 100)
YELLOW = Color(255, 255, 0)
DIM = Color(10, 10, 10)
LONG_PRESS_THRESHOLD = 0.5

colors = [NO_COLOR, MORNING_COLOR, DAY_COLOR, NIGHT_COLOR, YELLOW, DIM]
color_index = 0

# GPIO
SWITCH_PIN = 23
GPIO.setmode(GPIO.BCM) # Configure the Pi to use the BCM (Broadcom) pin names
GPIO.setup(SWITCH_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Set all the LEDs to the color specified
def set_all(strip, color):
        for i in range(strip.numPixels()):
                strip.setPixelColor(i, color)
                strip.show()

# Set the color based on the time of day
def set_color_index_from_time():
    global color_index
    now = datetime.datetime.now()
    if now.hour < 10:
        color_index = 1
    elif now.hour < 21:
        color_index = 2
    else:
        color_index = 3

# Initialize the display
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_
INVERT, LED_BRIGHTNESS)
strip.begin()
set_all(strip, colors[0])
strip.show()

print 'Press Ctrl-C to quit.'

while True:
    if GPIO.input(SWITCH_PIN) == False: # button pressed
        down_time = time.time()
        time.sleep(0.2) # debounce
        # Wait for switch release or long press timeout
        while GPIO.input(SWITCH_PIN) == False and time.time() - down_time <= 
LONG_PRESS_THRESHOLD :
            time.sleep(0.1)
        press_duration = time.time() - down_time
        if press_duration > LONG_PRESS_THRESHOLD:
            set_color_index_from_time()
        else:
            color_index += 1
            if color_index > len(colors)-1:
                color_index = 0

        set_all(strip, colors[color_index])
        strip.show()
        while GPIO.input(SWITCH_PIN) == False:
            time.sleep(0.1)
```

**RICHARD SMEDLEY**

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

Raspbian, while presenting a simple surface, also lets you dig deep for information when you need to change default behaviour. That's real user-friendliness!



```
Note: sector size is 4096 (not 512)

Disk /dev/sda: 2000.4 GB, 2000398929920 bytes
255 heads, 63 sectors/track, 30400 cylinders, total 48837864
Units = sectors of 1 * 4096 = 4096 bytes
Sector size (logical/physical): 4096 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
Disk identifier: 0xed49abfc

   Device Boot      Start         End      Blocks   Id  Syst
/dev/sda1            2048   488376319  1953497088    7  HPFS
pi@raspberrypi ~ $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
rootfs           3023728 2707332    143084  95% /
/dev/root        3023728 2707332    143084  95% /
devtmpfs          219744       0    219744   0% /dev
tmpfs              44784     256     44528   1% /run
tmpfs               5120       0      5120   0% /run/lock
tmpfs              89560       0     89560   0% /run/shm
/dev/mmcblk0p1     57288    9920     47368  18% /boot
pi@raspberrypi ~ $ sudo mount -a
pi@raspberrypi ~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
rootfs          2.9G  2.6G  140M  95% /
/dev/root       2.9G  2.6G  140M  95% /
devtmpfs        215M     0  215M   0% /dev
tmpfs            44M  256K   44M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs            88M     0   88M   0% /run/shm
/dev/mmcblk0p1   56M  9.7M   47M  18% /boot
/dev/sda1       1.9T  251G  1.6T  14% /mnt/1t
pi@raspberrypi ~ $
```

```
pi@raspberrypi ~ $ tail -n 4 /proc/diskstats
 179       1 mmcblk0p1 75 91 873 50 0 0 0 0 0 50 50
 179       2 mmcblk0p2 5917 4519 426234 24220 2877 2719 124984 314036
0 0 128300 3164490
   8       0 sda 15155 0 121240 8570 2 0 16 0 0 8480 8480
   8       1 sda1 15068 0 120544 8460 2 0 16 0 0 8370 8370
pi@raspberrypi ~ $ ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx 1 root root 15 Jun 17 14:17 936C-7154 -> ../../mmcblk0p1
lrwxrwxrwx 1 root root 15 Jun 17 14:17 c1398422-7a7c-4863-8a8f-45a1db
26b4f2 -> ../../mmcblk0p2
lrwxrwxrwx 1 root root 10 Jun 18 06:49 E4EE32B4EE327EBC -> ../../sda1
pi@raspberrypi ~ $
```

Even simple utilities have multiple uses: df, by showing space available, reminds the user which disks are mounted and can be accessed by the Pi

# COMMAND LINE PI PART 6: CONNECTING DISKS

**Richard Smedley** presents your cut-out-and-keep guide to using the command line on the Raspberry Pi. In part 6, we tackle the management of removable storage

**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.

**IN DEPTH**

If you want to delve deeper into what goes on inside Raspbian and other GNU/Linux systems, try Brian Ward's *How Linux Works*, which we reviewed in *The MagPi #32*.

Although Raspbian will, when booted as far as the GUI, automatically mount any disk-type device (USB flash key, camera, etc.) plugged into the USB port and offer to open it for you (**Fig 1**), you may wish to get more direct control of the process. Or, as is more often the case, you may want to mount a disk when the Raspberry Pi is running a project that doesn't involve anyone getting as far as typing **`startx`** at the command line, as such graphical fripperies aren't necessary for most connected devices.

## Connected or mounted?

Plugging a drive or flash memory device into your Pi (*connecting* it to your computer) is not the same as making it available for the Pi to interact with (*mounting* it) so that Raspbian knows what's on it and can read, write, and alter files there. It's an odd concept to accept: the computer knows there's a disk plugged in, but its contents remain invisible until the Pi is told to mount it. It's a bit like seeing a book on your shelf, but not being allowed to open or read it.

Disks and disk-like devices are mounted by Raspbian on a virtual file system, and you'll rarely need to worry about what goes on beneath that layer of abstraction, but to see some of it, type **`mount`**. The information displayed is of the form *device* on *mount point*, *file-system type*, *options*. You'll see a lot of device "none" for various bits of the virtual system that you don't need to worry about; the devices that concern us start with **/dev/** and have names like /dev/mmcblk0p1 for partitions of the Pi's SD card, and /dev/sda1 for plugged-in USB drives.

Plug in a USB drive (remember that the Pi is not happy to power drives itself: either use a powered drive, or plug a USB flash drive into a powered USB hub). If you haven't yet typed **`startx`**, then the disk will not get automatically mounted; if you have, then you need to unmount it. **`mount`** will show an entry beginning something like "/dev/sda1 on /media/FLASH DRIVE..." and you can unmount with **`sudo umount /dev/sda1`** (yes, that is umount without an "n"). An error will result if the device is in use, so change directory and/or close apps using files from the device. Now we can mount it just the way we want.

## Finding the disk

The /dev/sda1 refers to the first (or only) partition on /dev/sda. The next device plugged in will be /dev/sdb1. You can see what's being assigned by running **`tail -f /var/log/messages`**, then plugging in the USB device. On other Linux systems, if /var/log/messages draws a blank, try **/var/log/syslog**. Stop the tail with

**CTRL+C**. Another way of seeing connected devices that aren't necessarily mounted is with fdisk, a low-level tool used to divide disks up into partitions, before creating file systems on those disks (see the "Format" boxout). Called with the list option **`sudo fdisk -l`**, it performs no partitioning, but simply lists partitions on those disks connected to your Pi. It also gives file-system information, which you need in order to mount the disk. Lastly, you need a mount point (somewhere to place the device on the file-system hierarchy) with appropriate permissions. Create one with:

```
sudo mkdir /media/usb
sudo chmod 775 /media/usb
```

You can then mount the disk with **`sudo mount -t vfat /dev/sda1 /media/usb`**, where vfat (or ntfs or ext2) is the file-system type.

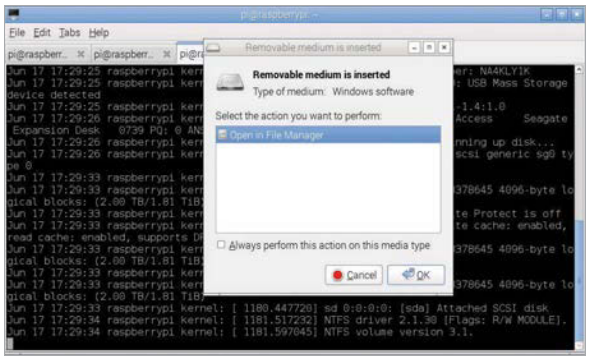
## File-system table

Raspbian knows which disks to mount at boot time by reading the file-system table (/etc/fstab), and we could put our /dev/sda1 in there, but if we start up with two drives plugged in, the wrong one may be selected. Fortunately, disks (or rather, disk partitions) have unique labels known as UUIDs randomly allocated when the partition is created. Find them with **`sudo blkid`**, which also helpfully tells you the label, if any, that often contains the make and model of external drives, or look in /dev/disk/by-uuid.

For an NTFS-formatted drive, we called **`sudo nano /etc/fstab`** and added the following to the end of the file:

```
/dev/disk/by-uuid/E4EE32B4EE327EBC  /media/
usb1t          ntfs    defaults  0    0
```

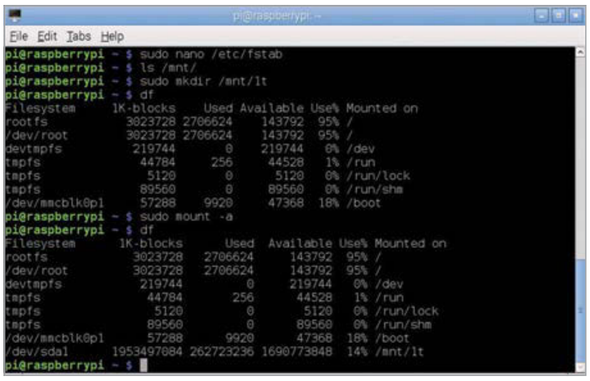
This gives the device name (yours will be different, of course), mount point, file-system type, options, and two numeric fields: the first of these should be zero (it relates to the unused dump backup program), while the second is the order of check and repair at boot: 1 for the root file system, 2 for other permanently mounted disks for data, and 0 (no check) for all others. **`man mount`** will tell you about possible options.


File Edit Tabs Help
pi@raspberr... × pi@raspberr... × pi@r
Removable medium is inserted
Open in File Manager
Cancel
OK

```
Jun 17 17:29:25 raspberrypi kern                                  er: NA4KLY1K
Jun 17 17:29:25 raspberrypi kern                                  : USB Mass Storage
device detected
Jun 17 17:29:25 raspberrypi kern                                  -1.4:1.0
Jun 17 17:29:26 raspberrypi kern                                  Access     Seagate
 Expansion Desk   0739 PQ: 0 ANS
Jun 17 17:29:26 raspberrypi kern                                  nning up disk...
Jun 17 17:29:26 raspberrypi kern                                  scsi generic sg0 ty
pe 0
Jun 17 17:29:33 raspberrypi kern
Jun 17 17:29:33 raspberrypi kern                                  378645 4096-byte lo
gical blocks: (2.00 TB/1.81 TiB)
Jun 17 17:29:33 raspberrypi kern                                  te Protect is off
Jun 17 17:29:33 raspberrypi kern                                  te cache: enabled,
read cache: enabled, supports DP
Jun 17 17:29:33 raspberrypi kern                                  378645 4096-byte lo
gical blocks: (2.00 TB/1.81 TiB)
Jun 17 17:29:33 raspberrypi kern
Jun 17 17:29:33 raspberrypi kern                                  378645 4096-byte lo
gical blocks: (2.00 TB/1.81 TiB)
Jun 17 17:29:33 raspberrypi kernel: [ 1180.447720] sd 0:0:0:0: [sda] Attached SCSI disk
Jun 17 17:29:34 raspberrypi kernel: [ 1181.517232] NTFS driver 2.1.30 [Flags: R/W MODULE].
Jun 17 17:29:34 raspberrypi kernel: [ 1181.597045] NTFS volume version 3.1.
```

**Removable medium is inserted**
Type of medium: Windows software
Select the action you want to perform:
☐ Always perform this action on this media type

**Fig 1** Raspbian wants to mount plugged-in disks, and take care of the details for you – note that the GUI tells you it's "Windows software" – while the command line beneath has information for you to take control when you need the job done in a particular way, telling you it's an NTFS file system



```
pi@raspberrypi ~ $ sudo nano /etc/fstab
pi@raspberrypi ~ $ ls /mnt/
pi@raspberrypi ~ $ sudo mkdir /mnt/1t
pi@raspberrypi ~ $ df
Filesystem     1K-blocks    Used Available Use% Mounted on
rootfs           3023728 2706624    143792  95% /
/dev/root        3023728 2706624    143792  95% /
devtmpfs          219744       0    219744   0% /dev
tmpfs              44784     256     44528   1% /run
tmpfs               5120       0      5120   0% /run/lock
tmpfs              89560       0     89560   0% /run/shm
/dev/mmcblk0p1     57288    9920     47368  18% /boot
pi@raspberrypi ~ $ sudo mount -a
pi@raspberrypi ~ $ df
Filesystem      1K-blocks      Used  Available Use% Mounted on
rootfs            3023728   2706624     143792  95% /
/dev/root         3023728   2706624     143792  95% /
devtmpfs           219744         0     219744   0% /dev
tmpfs               44784       256      44528   1% /run
tmpfs                5120         0       5120   0% /run/lock
tmpfs               89560         0      89560   0% /run/shm
/dev/mmcblk0p1      57288      9920      47368  18% /boot
/dev/sda1      1953497084 262723236 1690773848  14% /mnt/1t
pi@raspberrypi ~ $
```

**Fig 2** Once we've put our removable disk in the file-system table (/etc/fstab), `mount -a` will read the config from there to mount your disks, saving you from having to remember the details

## Editing with nano

We touched briefly on nano in part 2 of this series. Looking in a little more depth, the first thing to be aware of is the dozen shortcuts listed across the bottom two lines of the terminal: each is the **CTRL** key (represented by the caret ^) held at the same time as a single letter key. For example, ^R for ReadFile (*i.e. open*), ^O for WriteOut (in other words, *save*), and ^X for Exit. Remember those last two for now, and you'll be able to manage nano. However, if you learn more of them, you will really race through your editing tasks.

Nano lacks the power features of Emacs and Vim, its two main command-line code editor rivals, but has useful features such as a powerful Justify (^J), which will reassemble a paragraph of line-break strewn text into an unbroken paragraph, or apply line breaks at a fixed character length.This is a legacy of its development for email composition. ^K cuts the line of text the cursor is on, but it isn't just a delete function: each cut is added to a clipboard. ^U will paste the entire clipboard at the cursor position: it's great for gathering together useful snippets from a longer text.

Hit ^O to save fstab, and the shortcut listing changes, with many now beginning M instead of ^ – this is short for Meta, which means the **ALT** key on your keyboard (once upon a time, some computers had several modifier keys, such as Super and Hyper). One "hidden" shortcut after ^O is that at this point, ^T now opens a file manager to search for the file/directory where you want to save.

After saving, exit nano; now **`sudo mount -a`** will mount the external drive at the desired mount point (**Fig 2**), regardless of what else is plugged in. If you have other new entries in /etc/fstab, then **`sudo mount /media/usb1t`** (or whatever entry you put in fstab) will mount just that chosen device if you don't want to mount any of the others.

Having got inside connected disks, next time we'll be accessing all of the Pi, but remotely – from anywhere on the planet with an internet connection.

## DISK & DISK SPACE

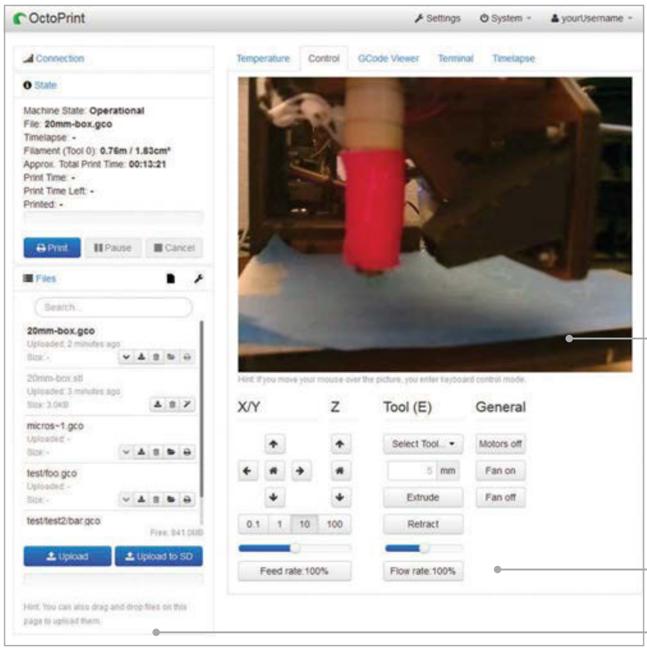
The *df* command shows you space on mounted drives: just type **df** and you'll also get a list of connected drives. It's more readable than **mount -l**, though lacking file type info. It's also quicker to type!

## FORMAT

Copying a disk image negates the need to format the disk. Should you need to format a new partition, or convert a disk to ext4 format, read the manual: **man mkfs** and for individual file-system types such as **man mkfs.ext4**

**GINA HÄUSSGE**

OctoPrint creator, now employed by BQ to work on it full-time! Loves tinkering and playing with both software and hardware in her free time.
**twitter.com/foosel**



Machine State: **Operational**
File: **20mm-box.gco**
Timelapse: -
Filament (Tool 0): **0.76m / 1.83cm³**
Approx. Total Print Time: **00:13:21**
Print Time: -
Print Time Left: -
Printed: -

20mm-box.gco
Uploaded: 2 minutes ago
Size: -

20mm-box.stl
Uploaded: 3 minutes ago
Size: 3.0KB

micros~1.gco
Uploaded: -
Size: -

test/foo.gco
Uploaded: -
Size: -

test/test2/bar.gco

Free: 841.0MB

Hint: If you move your mouse over the picture, you enter keyboard control mode.

Hint: You can also drag and drop files on this page to upload them.

Get a live view of your printer, right from within your browser

Manage your printable models and slice them, right on the Pi

Remotely control all aspects of your 3D printer just as if you were sitting next to it

**NEED HELP?**

You can find links to the FAQ, the docs, the G+ community and the mailing list at **octoprint.org**

# UNCHAINING YOUR 3D PRINTER WITH OCTOPRINT

We show you how to turn your 3D printer into a WiFi-enabled network printer that you can control through any web browser, using a pre-made Pi SD card image

**You'll Need**

- OctoPi image **github.com/guysoft/OctoPi**
- Camera Module **raspberrypi.org/products/camera-module**
- USB WiFi dongle, e.g. **raspberrypi.org/products/usb-wifi-dongle**
- Compatible 3D printer **bit.ly/1LXZf4b**
- Class 10 SD card (minimum 4GB)

Say goodbye to the cable salad of tethering your desktop PC or laptop to your 3D printer. With OctoPrint, there exists a 3D printer remote control that you can easily run on a Raspberry Pi and use from any web browser, giving you full control and monitoring capabilities. And thanks to the OctoPi distribution, getting this set up and ready to go is as easy as flashing an image to an SD card and booting your Pi from it.

We might try to convince ourselves to the contrary, but let's face it- 3D printers do smell a bit strange and can produce quite a racket while they are working multiple hours on your latest three-dimensional creation.

That's why I sat down a couple of years ago now to create OctoPrint, a free and open-source remote control and monitoring solution for 3D printers that is targeted at the Raspberry Pi. It allows you to control and monitor all aspects of your 3D printer and its print jobs just as if you were sitting in front of it, even if it's on the other side of your house in the garage – or on the other side of the world.

Let me show you how to get this up and running in no time, powered by OctoPi, the customised SD card image created by Guy Sheffer that merits the label 'batteries included'.

## Let's get started

First, go to **github.com/guysoft/OctoPi**, click on the download mirror linked there and download the most recent version of OctoPi. This might take a while. While the download is running, let's first prepare your desktop PC or laptop so that connecting to the server later will only require entering 'http://octopi.local' into your browser! OctoPi uses something called 'mDNS' (also known as 'Bonjour' or 'zeroconf') to broadcast this address on the local network and make it discoverable by other PCs that understand mDNS. Since not all systems support this out of the box, we now need to make sure yours will know what to do.

How to achieve that depends on what operating system you are running. Linux users should make sure that libnss-mdns is installed. On Debian and Ubuntu, a simple `sudo apt-get install libnss-mdns` should take care of that. Windows users need to install Bonjour Print Services for Windows, which can be downloaded from Apple at **support.apple.com/kb/DL999.** After installation, you'll need to make sure that your Windows Firewall allows traffic on UDP port 5353 and that mDNSresponder.exe has network access. Mac users are lucky; for them, mDNS should be supported out of the box by the operating system, without the need to install anything else.

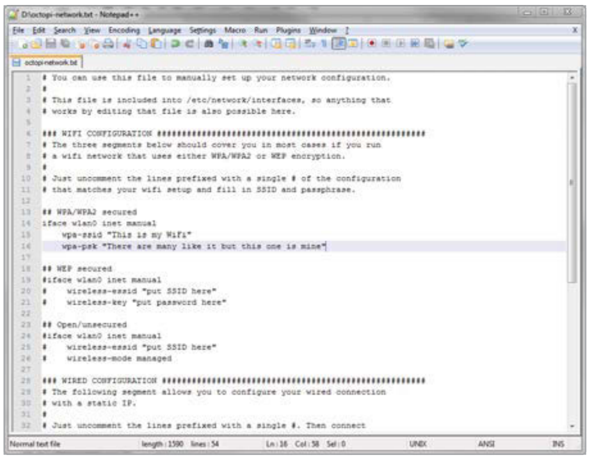
## Preparing the image

Once you have downloaded the ZIP file, unpack it on your computer and flash it to your SD card. After that's done, don't boot from it yet – we'll first configure your WiFi connection, which we can do quite easily without booting.

Plug the freshly flashed card into the SD card reader so your computer detects it just like it would a USB thumb drive. Open it up in your file manager and take a look there; you'll see a file called **octopi-network.txt**. Open it in a text editor. You'll see a bunch of example configuration snippets for various types of WiFi setups: encrypted with WPA/WPA2, encrypted with WEP, and unencrypted. Choose the one that matches your WiFi setup; if you are unsure, try the first one labelled 'WPA/WPA2', as that should be the most common type. Uncomment the three lines by removing the leading '#' and insert your WiFi SSID and password where it says so (**Fig 1**). Let's assume your WiFi has the SSID 'MyWifi', the password 'MySuperSecretPassword' and is WPA2 encrypted; if so, the four lines related to the WPA configuration snippet should look like this:

```
## WPA/WPA2 secured
iface wlan0 inet manual
    wpa-ssid “MyWifi”
    wpa-psk “MySuperSecretPassword”
```

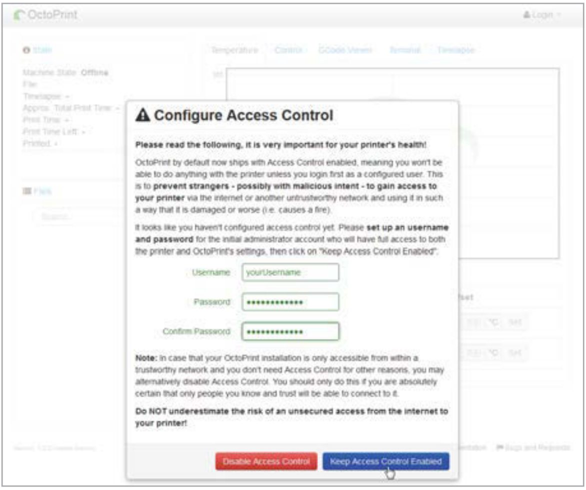
Save the file and securely remove the SD card reader. It's now time to boot up the Pi!


D:\octopi-network.txt - Notepad++

```
# You can use this file to manually set up your network configuration.
#
# This file is included into /etc/network/interfaces, so anything that
# works by editing that file is also possible here.

### WIFI CONFIGURATION ##############################################################
# The three segments below should cover you in most cases if you run
# a wifi network that uses either WPA/WPA2 or WEP encryption.
#
# Just uncomment the lines prefixed with a single # of the configuration
# that matches your wifi setup and fill in SSID and passphrase.

## WPA/WPA2 secured
iface wlan0 inet manual
    wpa-ssid "This is my Wifi"
    wpa-psk "There are many like it but this one is mine"

## WEP secured
#iface wlan0 inet manual
#     wireless-essid "put SSID here"
#     wireless-key "put password here"

## Open/unsecured
#iface wlan0 inet manual
#     wireless-essid "put SSID here"
#     wireless-mode managed

### WIRED CONFIGURATION #############################################################
# The following segment allows you to configure your wired connection
# with a static IP.
#
# Just uncomment the lines prefixed with a single #. Then connect
```

Normal text file length : 1590 lines : 54 Ln : 16 Col : 58 Sel : 0 UNIX ANSI INS

**Fig 1** Setting up your WiFi configuration through the octopi-network.txt file. Just enter your WiFi details and save


**Configure Access Control**

**Please read the following, it is very important for your printer's health!**

OctoPrint by default now ships with Access Control enabled, meaning you won't be able to do anything with the printer unless you login first as a configured user. This is to **prevent strangers - possibly with malicious intent - to gain access to your printer** via the internet or another untrustworthy network and using it in such a way that it is damaged or worse (i.e. causes a fire).

It looks like you haven't configured access control yet. Please **set up an username and password** for the initial administrator account who will have full access to both the printer and OctoPrint's settings, then click on "Keep Access Control Enabled".

Username: yourUsername
Password: •••••••••••••
Confirm Password: •••••••••••••

**Note:** In case that your OctoPrint installation is only accessible from within a trustworthy network and you don't need Access Control for other reasons, you may alternatively disable Access Control. You should only do this if you are absolutely certain that only people you know and trust will be able to connect to it.

**Do NOT underestimate the risk of an unsecured access from the internet to your printer!**

**Fig 2** Setting up your OctoPrint account on the first run. You should always enable access control if you plan to access your OctoPrint instance through the internet

## Final setup

Connect the camera, your printer and your WiFi dongle to your Raspberry Pi, then plug the SD card in and power your Pi up. The first bootup will take a while, but sooner or later your WiFi dongle should show some activity and your camera will switch on. If you've entered your WiFi credentials correctly and set up mDNS support properly, you should now be able to just SSH into the Pi via the address **octopi.local** (using the username **pi** with the password **raspberry**) to perform some final setup steps.

First of all, change the password with **`passwd`** – don't forget it! Then run **`sudo raspi-config`** and expand the file system on the SD card. After the obligatory reboot which follows, enter **http://octopi.local** into your browser. Depending on your Raspberry Pi model, this may take a little while to show you a page, but sooner or later OctoPrint's web interface will greet you with a prompt to set up your OctoPrint account. Choose a username and password, click 'Keep Access Control enabled' (**Fig 2**) and log into your shiny new OctoPrint instance via the Login button in the upper right corner. Then select your 3D printer's serial port and the appropriate baud rate from the Connection menu on the left, and click Connect. Once it says 'Connected' and your temperature graph starts showing your data, you are all set. Congratulations!

You can now fully control your printer from the web interface, upload files to print, remotely watch your print jobs, and create time-lapses. And if you take a look into the Plugin Manager inside the Settings dialog, there are also a bunch of very nice things to be discovered there. Have fun!

“Let's face it, 3D printers do smell a bit strange and can produce quite a racket while they are working multiple hours on your latest three-dimensional creation”

**CUSTOMISING OCTOPRINT**

You can add custom controls or get notified about finished print jobs – take a look at the docs!

**SEAN MCMANUS**

Sean McManus is a Code Club volunteer, and wrote the book *Scratch Programming in Easy Steps*. He co-wrote *Raspberry Pi For Dummies*.

**sean.co.uk**

**twitter.com/musicandwords**

# ADD A TITLE SCREEN TO YOUR SCRATCH GAME

To make a professional-looking game, follow these steps to add a title screen with instructions and a fun animation

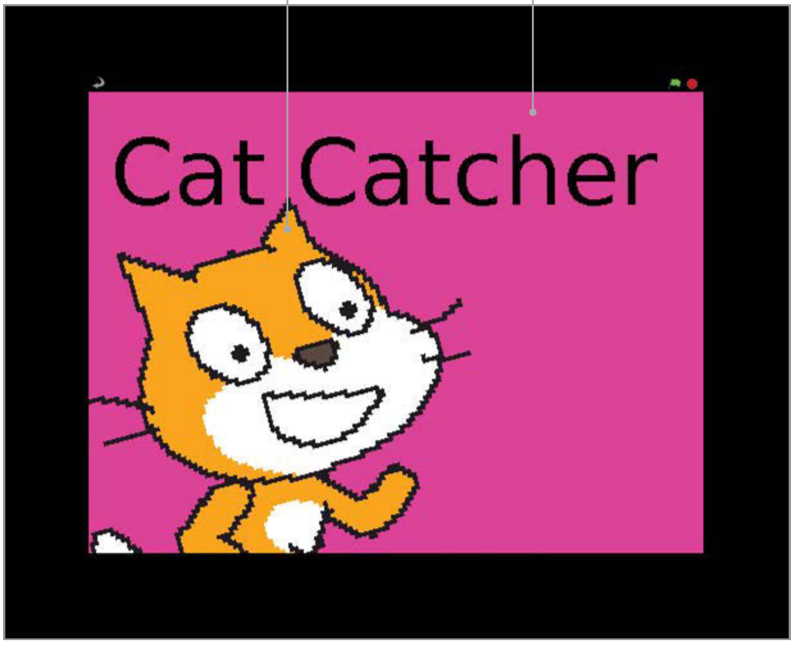
**You'll Need**

- One of your own Scratch games, or you can use our simple example here

A book has a cover, a film has its credits, and an album has its artwork. Only with the right presentation do these things feel professional and complete. In the same way, a great game starts with a title screen that draws players in and provides instructions. It's especially important if you want to share your game, as you won't be there to explain it when it's played. In this article, you'll see how you can add a title screen to a basic game. The same techniques will work for most simple games, so why not try adding a title screen to your own games, too?

Add an animated sprite to your title screen and use 'say' blocks to tell players how it works

Black text on hot pink: a timeless background design!



## >STEP-01

### Write your game

We recommend you try adding a title screen to our example game Cat Catcher before you add one to your own game. To make Cat Catcher, first bring in the sprite Gravity Marble from the Things folder. It comes with some scripts for controlling it with the cursor keys. Add **Listing 1** to your cat sprite. Together, these two sprites make a game where you're challenged to see how quickly you can catch the cat ten times with the marble. We've added the playing field background.

## >STEP-02

### Create your title screen background

Create a new background image that you will be using for your game's title screen. Ours is just a bright colour with the game title on it, but you could make something more elaborate if you like. On the background, add the scripts shown in **Listing 2**. They change the background between the title screen and the in-game background, and tell all the sprites to go into 'title screen' mode when the green flag is clicked. Ultimately, this should be the only time you use a `when green flag clicked` script.

## >STEP-03

### Create your title screen sprite

This is the sprite that will tell the player how to play, and it can be animated, too. For our game, we've brought in another cat sprite. Add **Listing 3** to it. There are three parts to this: one part displays the title animation and instructions; another part starts the game when the sprite is clicked; and a third part hides the sprite when the game begins. You'll need to make a variable called `game status`, which all sprites will use to tell whether the game is running or the title screen is on. You can add more sprites to your title screen. Include the **when I receive play game** script from **Listing 3** to hide them when the game begins. Use a **when I receive title screen** script to show them on the title screen.

## >STEP-04

### Replace your green flag scripts

Now, you need to go through your game sprites (the game cat and the marble in our example) and change their scripts so they don't start when the green flag is clicked any more. For each sprite and each of its scripts, replace the block **when green flag clicked** with the block **when I receive play game**. Add **Listing 4** to your game sprites to make them hide when the title screen is on, and appear when the game begins. If a sprite shouldn't be there at the start of the game, you can leave out the **show** script.

> A great game starts with a title screen that draws players in

## >STEP-05

### Replace the forever loops

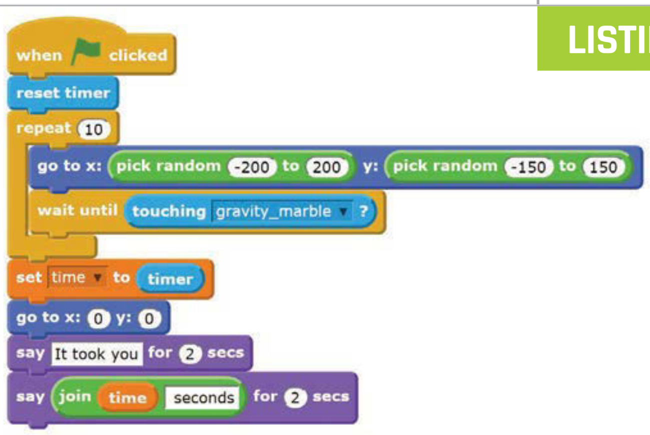
Some of your in-game sprites might have **forever** loops. These will keep running, even when the title screen is showing and the sprite is hidden. To avoid this causing unwanted results, replace the **forever** block on your in-game sprites with the **forever if** block. Give the block the condition **game status = game** (using your variable **game status** and the **= Operator** block). You might also have events that are triggered, such as when there is a key press. To stop these working on the title screen, wrap an **if** block around the entire script after the **when [space] key pressed** block and give it the condition **game status = game**, too.

## >STEP-06

### Start a new game

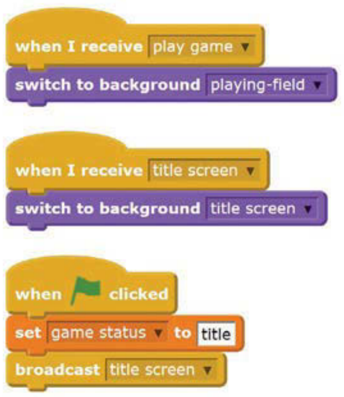
When your game finishes, you can show the title screen again by adding a Control block to broadcast **title screen**. For example, you could add it to the end of **Listing 1** in our game. Players can once again start a new game from the title screen. That will keep them in the game and encourage them to keep playing until they've got a score they can brag about! You might need to make some other tweaks for your game (each one is different, after all), but following these steps should enable you to add a title screen to most simple games, to make them look more polished.

**LISTING 1**



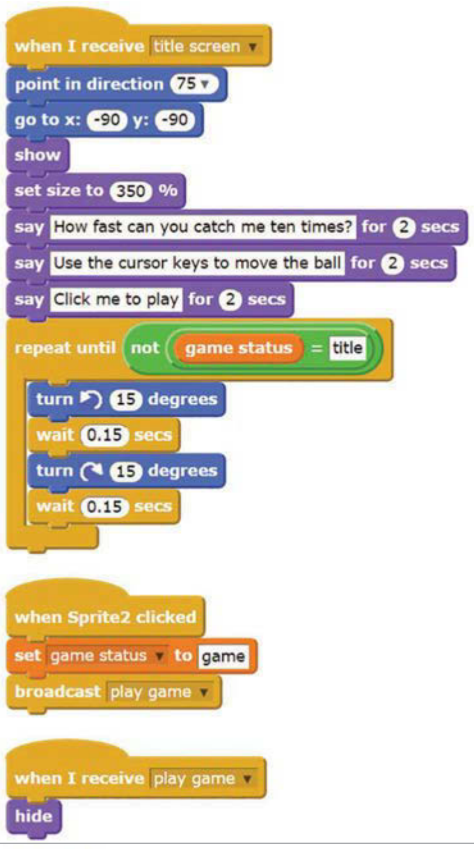
```
when green flag clicked
reset timer
repeat (10)
    go to x: (pick random (-200) to (200)) y: (pick random (-150) to (150))
    wait until <touching [gravity_marble v] ?>
set [time v] to (timer)
go to x: (0) y: (0)
say [It took you] for (2) secs
say (join (time) [ seconds]) for (2) secs
```

**LISTING 2**



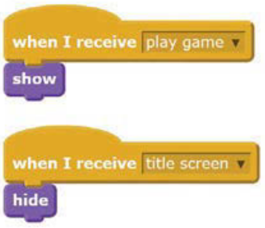
```
when I receive [play game v]
switch to background [playing-field v]

when I receive [title screen v]
switch to background [title screen v]

when green flag clicked
set [game status v] to [title]
broadcast [title screen v]
```

**LISTING 3**



```
when I receive [title screen v]
point in direction (75 v)
go to x: (-90) y: (-90)
show
set size to (350) %
say [How fast can you catch me ten times?] for (2) secs
say [Use the cursor keys to move the ball] for (2) secs
say [Click me to play] for (2) secs
repeat until <not <(game status) = [title]>>
    turn ↺ (15) degrees
    wait (0.15) secs
    turn ↻ (15) degrees
    wait (0.15) secs

when Sprite2 clicked
set [game status v] to [game]
broadcast [play game v]

when I receive [play game v]
hide
```

**LISTING 4**

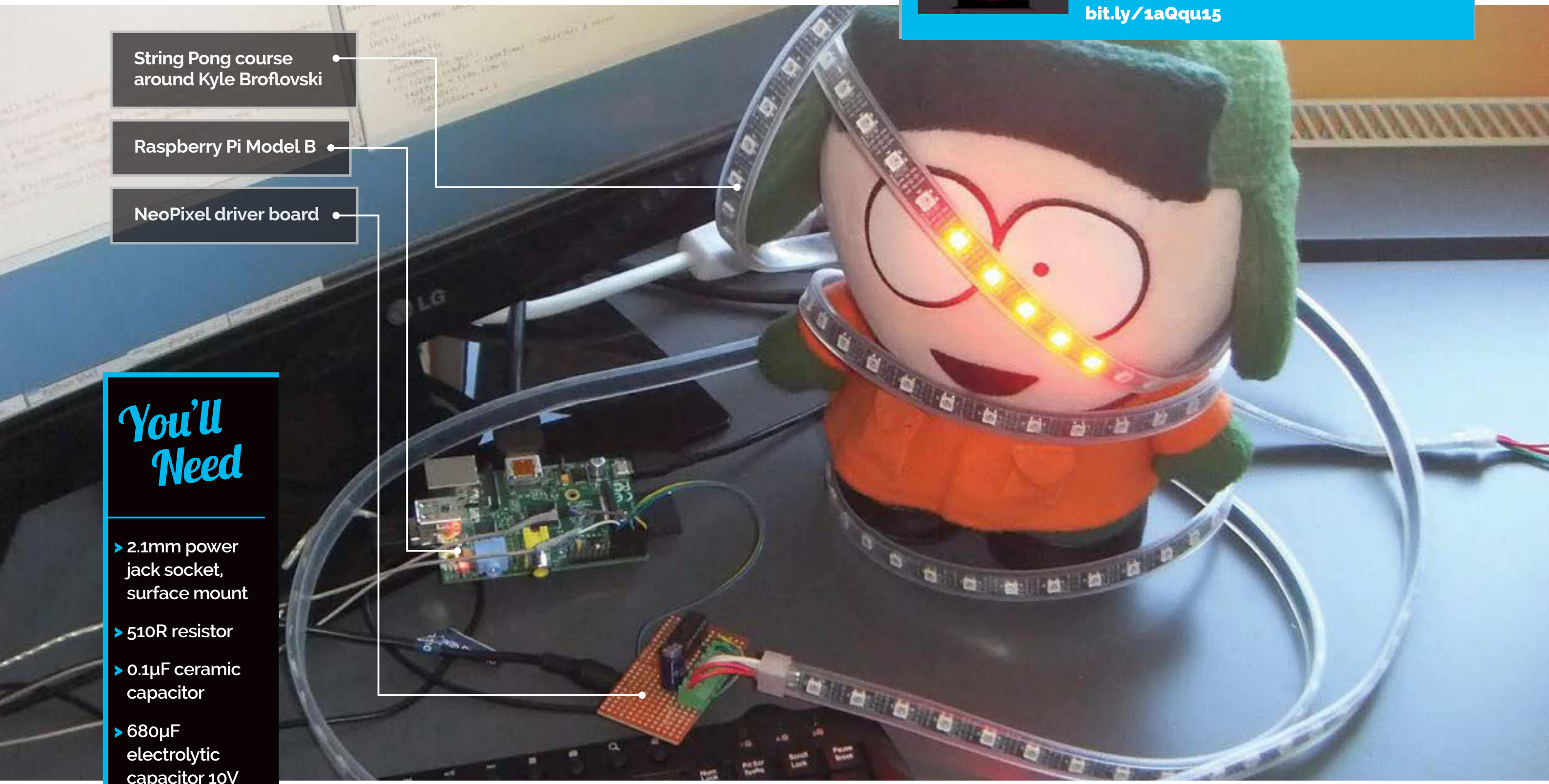


```
when I receive [play game v]
show

when I receive [title screen v]
hide
```

## MIKE'S PI BAKERY

**MIKE COOK**

Veteran magazine author from the old days and writer of the Body Build series. Co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.
**bit.ly/1aQqu15**



**You'll Need**

- 2.1mm power jack socket, surface mount
- 510R resistor
- 0.1µF ceramic capacitor
- 680µF electrolytic capacitor 10V working (or more)
- 6-way 0.2″ (5mm) pitch screw connector
- 14-pin DIL IC socket
- 74LS14 hex buffer inverter Schmitt input
- Stripboard, 24 by 13 holes
- 4 metres WS2812B RGB LED strip in waterproof tube, (60 LEDs per metre)
- 5V power supply, 2A minimum but up to 15A
- 2× foot switches (non-latching) **www.maplin.co.uk/p/heavy-duty-non-latching-footswitch-r61hq** (or last month's foot switches)
- Raspberry Pi 1 Model B/B+ This project will **not** work with a Raspberry Pi 2

# STRING PONG

String Pong is a new twist on the game of ping-pong, where players metaphorically bat a string of lit LEDs between each other. Here's how to make it yourself...

String Pong is a digital table-tennis game with a unique angle. As it is played over a flexible LED strip, you can change the shape of the course each time you play! It is a simple project, made possible by the new WS2812B combined controller and RGB LED chip. This is marketed under the brand name NeoPixel by Adafruit, but is also available elsewhere. NeoPixels can be bought as pre-assembled flexible LED strips at densities of 30, 60, and 120 LEDs per metre. For this project, we chose the version with 60 LEDs.

### The game

A white LED indicates it's your turn to serve. Trigger your foot switch and a slug of five lit LEDs travels towards the opposite end of the strip. When it reaches the end, the opponent has to bat it back by pressing their foot switch. If they bat too early, this counts as a miss and the LED turns blue. Otherwise, the LED turns green and the slug is sent in the other direction. Both the speed and colour of the returned slug will change depending on when it was hit. The longer you leave it before pressing the switch, the faster it'll be returned.

### The hardware

Individual LEDs typically consume 20mA maximum, but these are RGB LEDs: each one has three LED components, so you end up consuming 60mA. While this does not seem much, a 4-metre strip with 60 LEDs per metre will need a current of about 15A at 5V to drive it to full brightness. Fortunately, this game does not require so much current, as the LEDs are never all going to show white simultaneously. Nevertheless, they do require more current than the Raspberry Pi can supply, so you'll need an external 5V power supply. We used a 4A supply, but you could get away with a 2A one.

Beside the two power inputs, these strips have only one data input: the data consists of a rapid stream of pulses with quite tight timing requirements. This would normally rule out a Linux-based system like the Pi, but the people at Adafruit have come up with a clever way to use the direct memory access (DMA) hardware in conjunction with the PWM shift register to set up a precisely timed pulse waveform to drive these LEDs. Sadly, this has not yet been ported to work on the Pi 2, so you'll need a first-generation Pi.

The waveform needs to be a 5V logic signal, but the Pi only produces a 3V3 signal, so we have to boost it before it can drive the LED strip. Normally you could use a transistor for this, but these signals are going at just under 1MHz, and making a transistor work at those frequencies is tricky. The solution is to use a logic buffer. We used two Schmitt input inverting buffers, which need a voltage lower than 0.5V to register a logic 0, and in excess of 1.9V to register as a logic 1 when powered from 5V. The hardware schematic is shown in **Fig 1**: note that two buffers are chained so that the resulting output is not inverted. The remaining buffers in the package are chained together at a fixed logic level to prevent instability in the chip. The output signal passes through a 510R resistor before being connected to the LED strip, to absorb any reflections on the line out to the first LED, and to limit the current if you connect up to an unpowered LED strip. Finally, a large bulk decoupling capacitor is placed across the external 5V supply; the value is not critical, so you can use something larger than shown. The schematic also shows two foot switches connected to the Raspberry Pi; you can use momentary-type foot switches here, not latching ones, although they can be expensive.

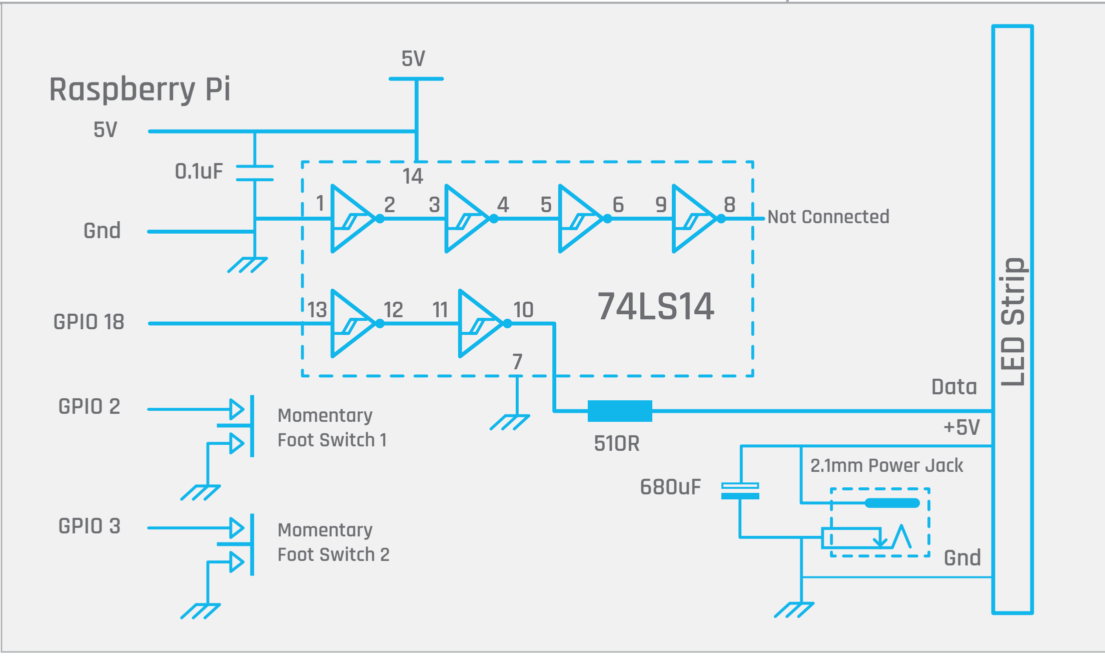


**Fig 1** The schematic of String Pong

## The software

The Python callback function is used to read the switch under the RPi.GPIO library. You'll need to install the Adafruit NeoPixel driver library (**learn.adafruit.com/neopixels-on-raspberry-pi/software**), too.

The NeoPixel library generates a buffer where each entry is the colour you want each LED to have. There are calls that can set a specific pixel number to a specific colour: any code should set the buffers up with the colours you want; then, by calling the show method, it will transfer this buffer to the LEDs.

The game works by having a variable define an insertion point in the buffer where the slug of lit LEDs will be placed. This is complicated by the fact that as the slug is disappearing down one end, you see fewer of the LEDs until you see none, so there is code that prevents you writing into non-existent buffer locations. When a 'bat' is detected, the value of the current insertion point is used to determine whether you have hit the LEDs or you have gone too early.

## The code

The program (listed on page 57) runs in Python. You can use any length of LED strip: the number of LEDs is set up in the **`NUM_PIXELS`** constant, at the start of the program. The **`ballDir`** variable determines which direction the 'ball' is going, and the **`interval`** variable controls the speed of travel. The list **`ballSpeed`** is used to set this variable on each successful bat, and controls

"In String Pong, you have to play your bat at just the right moment"

the pace of the game. The **`red`**, **`green`**, and **`blue`** lists control the colours of the returned ball, linked to the speed. The **`main`** function controls the game: each bat is continuously checked and, on the expiration of the set time interval, the LED display is advanced. Checks are made to see if the ball has overrun either end, or a ball has been hit. Finally, if the ball is in play, signalled by the **`serve`** variable, the LED strip is updated. The GPIO **`callback`** function is set up to be **`buttonPress`**, which looks at the number returned and clears the **`bat1`** or

**Left** The top view of the LED string driver

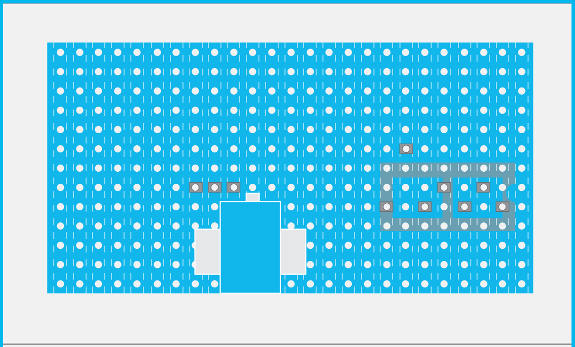
**bat2** variable accordingly. Determining whether this signal represents a serve, hit, or miss is done by the two **checkBat** functions; these only operate when the **bat** variable has been cleared by the **callback** function.

The **place** function is where the buffer is updated: first it's cleared using the **wipe** function, then the slug of LEDs is placed in the buffer at the insertion point, if this is within the bounds of the buffer. The end LEDs indicating 'serve', 'hit', or 'too soon' are updated, and the **pixels.show** method writes out the buffer to the LEDs. The **endRally** function tidies up things at the end, when one player has failed to return the ball.

## Customisation

As String Pong is based on a flexible LED strip, you can make the track straight or curved. The more twisted the path, the harder the game! You can choose a one- or two-player game; for the one-player version, you simply alternate the foot switch you hit. There are lots of ways in which you can customise this code, apart from tinkering with colours and speed. You could add sound effects, like an 'out' call or the sound of a racket striking a ball. You could also adopt tennis scoring and use the Python console window for feedback, score-keeping, or even a championship record.

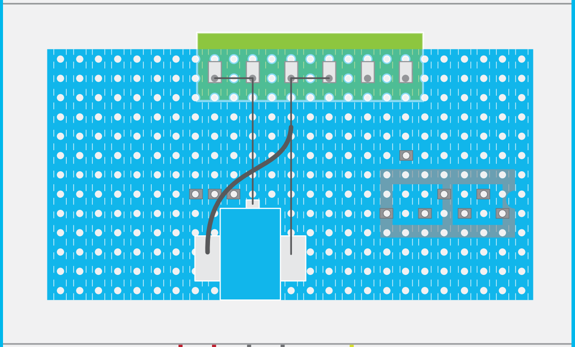
## BUILDING THE WS2812B DRIVER BOARD



### >STEP-01

**Track cuts and sockets**

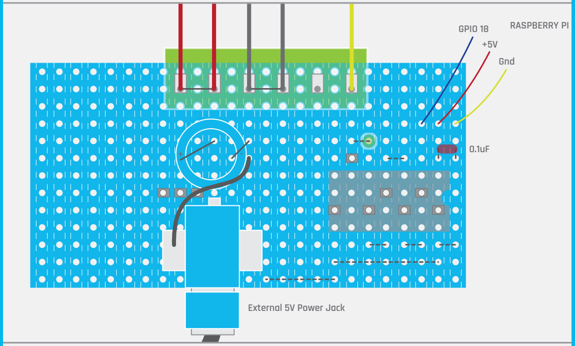
This view shows the underside or track side of the stripboard. Cut a piece of stripboard, 24 holes wide by 13 holes deep. Make sure that the copper strips run vertically. Cut the breaks in the track on the underside, marked by the grey squares. Then position the surface-mount connector so the central connector is exactly 11 holes from the right and solder it down. Check that it is straight – at this stage, it is easy to adjust by melting just this one joint. Now solder the two side pieces to make a solid connection. Add the IC socket on the non-copper side of the board and solder it in.



### >STEP-02

**Power and connector**

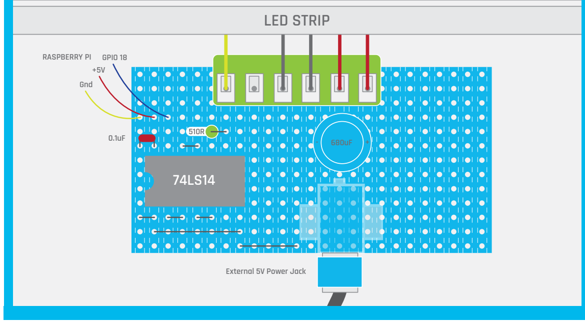
Next, add the connector strip and solder up the two left-hand pins to secure it. The current drawn by the LED strip is too much for the copper strip to take, so reinforce the strip by soldering some uninsulated wire over the strip for both the positive and negative connections of the LED strip. Continue this around the adjacent pin so that there are two connections carrying each power wire. Finally, add a heavy insulated wire from the left side of the power jack over to the negative track from the other side of the jack. This needs to be insulated so that it does not short out the tracks it passes over.



### >STEP-03

**Links and components**

Now add the wire links around the IC socket; these should be placed on the component side of the board. Solder in the 0.1μF decoupling capacitor and the resistor. Note that the resistor is stood up on its end to allow mounting on adjacent holes. Add the 680μF capacitor, making sure the polarity is the right way round, and connect it up across the supply. Plug in the IC, making sure the registration mark is nearest the left-hand side of the board. Finally, wire up the connections to the Raspberry Pi and the LED strip. You will have to cut off the plug on the end of the LED strip to give you the bare wires.



### >STEP-04

**Top view with connections**

Double-check all the connections and then connect the appropriate three wires to the Raspberry Pi's GPIO header with the Pi unpowered. Attach the foot switches. Boot up the Raspberry Pi and connect the external power supply just before you run any code. When the code runs, you should see the LED at the start of the strip light up white. If you do not see this LED light up immediately, remove the external 5V power, shut down the Raspberry Pi, and check the circuit again.

## StringPong.py

**Language**
**>PYTHON 2**

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

```
import time, random, os
import RPi.GPIO as io
from neopixel import *

DATA_PIN  = 18 # pin connected to the NeoPixels
NUM_PIXELS = 240 # number of LEDs 240 LEDs in 4 meters
try :
   io.setmode(io.BCM)
   io.setwarnings(False)
except :
   print"start IDLE with 'gksudo idle' from command line"
   os._exit(1)

pixels = Adafruit_NeoPixel(NUM_PIXELS,DATA_PIN,800000,5,False)

bat1pin = 3
bat2pin = 2
bat1 = bat2 = 1
serve = False
ballDir = True
toServe = 0 # player to serve next
slugLength = 5
hitPoint1 = hitPoint2 = -1
lastTime = time.time()
interval = 0.1 # delay between frames
insertPlace = 0
slugCol = 5

red =   [ 204,  0, 255, 255, 255, 255]
green = [ 102, 255,  0,  0,  153,  225]
blue =  [ 0,   255, 255, 0,  153,  0]
ballSpeed  = [ 0.005, 0.01, 0.015, 0.020, 0.025, 0.03 ]

def main() :
   global lastTime, interval, ballDir, insertPlace, serve
   global slugLength, hitPoint1,hitPoint2,slugCol
   init()
   while(True):
      checkBat1()
      checkBat2()
     # advance the ball
      if ((time.time() - lastTime) > interval) & serve:
        lastTime = time.time()
        if(ballDir) :
           insertPlace += 1
        else :
           insertPlace -= 1

        if (insertPlace + slugLength) < 0 : # hit the player 1 end
            insertPlace = -slugLength
            ballDir = not ballDir
            hitPoint2 = -1 # clear the other players hit position
            if hitPoint1 == -1 or hitPoint1 > slugLength :  # failed or too early
                 endRally(0) # tidy up an wait for serve
            else :
                 slugCol = hitPoint1 # change colour for next rally
                 interval = ballSpeed[slugCol]

        if insertPlace > NUM_PIXELS : # hit the player 2 end
           insertPlace = NUM_PIXELS-1
           ballDir = not ballDir
           hitPoint1 = -1 # clear other players hit position
           if hitPoint2 == -1 or (NUM_PIXELS -1 - hitPoint2) > slugLength :
                # failed to hit or too early
                 endRally(1) # tidy up an wait for serve
           else :
              slugCol = NUM_PIXELS -1 - hitPoint2 # change colour for next rally
               interval = ballSpeed[slugCol]
      if serve :
         place(insertPlace) # don't update if we have just missed the ball

def init() :
   global insertPlace, interval
   print"String Pong - by Mike Cook"
   print"Blue LED - you swung too soon"
   print"Green LED - you swung in time"
   print"No LED - you swung too late"
   print"White LED - serve when you are ready"
   pixels.begin() # This initializes the NeoPixel library.
   setupStart()
   print"Please ignore this stupid warning:-"
   io.setup(bat1pin,io.IN, pull_up_down = io.PUD_UP)
   io.add_event_detect(
bat1pin,io.FALLING, callback=buttonPress, bouncetime=30)
   io.setup(bat2pin,io.IN, pull_up_down = io.PUD_UP)
   io.add_event_detect(
bat2pin,io.FALLING, callback=buttonPress, bouncetime=30)
   insertPlace = -slugLength
   interval = ballSpeed[slugCol]
   print"Player",toServe+1,"to serve"

def buttonPress(number): #call back function
   global bat1,bat2
   if number == 2:
      bat2 = 0
   else :
      bat1 = 0

def endRally(player):
     global serve,toServe,bat1,bat2
     global interval,slugCol
     print"out"
     bat1 = bat2 = 1
     serve = False
     time.sleep(1.0) # leave it
     wipe()
     pixels.show() # clear display
     time.sleep(0.8)   # leave blank for a time
     toServe = player
     print"Player",player+1,"to serve"
     setupStart()
     bat1 = bat2 = 1
     if ballSpeed[slugCol] == ballSpeed[0] :
        slugCol = random.randint(2,5)
        interval = ballSpeed[slugCol]

def checkBat1():
  global serve, hitPoint1,hitPoint2, bat1
  if bat1 == 0 :
    bat1 = 1
    if (serve==False) & (toServe == 0) :
       serve=True # put ball in service
       print"serve 1"
       hitPoint1 = hitPoint2 = -1
    else :   # trying to hit the ball
       if hitPoint1 == -1 : # only record the first hit
           hitPoint1 = slugLength + insertPlace


def checkBat2():
  global serve, hitPoint1,hitPoint2, bat2
  if bat2 == 0 :
    bat2 = 1
    if (serve==False) & (toServe == 1) :
       serve=True # put ball in service
       print"serve 2"
       hitPoint1 = hitPoint2 = -1
    else :   # trying to hit the ball
       if hitPoint2 == -1 : # only record the first hit
         if(insertPlace != NUM_PIXELS) :
            hitPoint2 = insertPlace

def place(point):
  global hitPoint1,hitPoint2
  wipe()
  for i in range(0, slugLength) : # put the slug in the buffer
    if (i+ point) >= 0 & ((i+ point) < NUM_PIXELS) :
       pixels.setPixelColor(
i+ point,Color(red[slugCol],green[slugCol],blue[slugCol]))
  # set the end LEDs acording to the hit
  if(hitPoint1 != -1):
       if(hitPoint1 > slugLength) :
         pixels.setPixelColor(0, Color(0,0,128))
       else :
         pixels.setPixelColor(0, Color(0,128,0))

  if hitPoint2 != -1 :
       if hitPoint2 < ( NUM_PIXELS- 1 - slugLength) : # bat too soon
          pixels.setPixelColor(NUM_PIXELS -1, Color(0,0,128))
          else :
          pixels.setPixelColor(NUM_PIXELS -1, Color(0,128,0))
  pixels.show()

def setupStart(): # set display waiting to serve
  wipe()
  if toServe == 0:
     pixels.setPixelColor(0, Color(128,128,128))
  else :
     pixels.setPixelColor(NUM_PIXELS-1, Color(128,128,128))
  pixels.show()

def wipe():
    for i in range(0,pixels.numPixels()):
       pixels.setPixelColor(i, Color(0,0,0))

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

PART 6

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**

# PHYSICS & FORCES

In the sixth part of this series, we give our game objects mass and let gravity come into play…

**You'll Need**

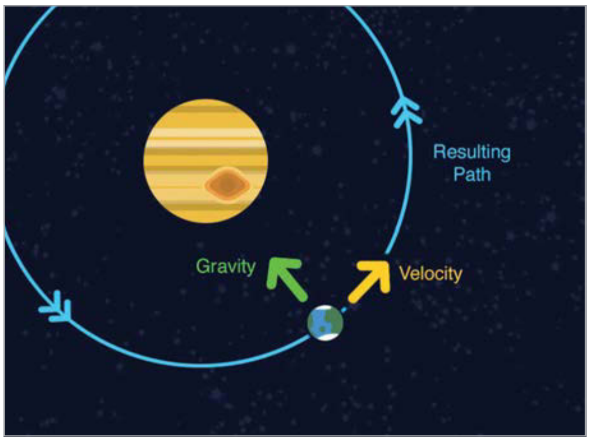
- Raspberry Pi running Raspbian
- Internet access to download a script from GitHub

In the previous parts of this series, we've put together code that let us take control of elements in our program whenever we interact with them, be it by clicking, dragging or typing. The thing is, there's only so much we can do with these interactions; no matter what, everything we do will be determined by ourselves in some way, and that can get a little bit boring after a while. This being the case, in this issue we're going to give certain elements of our program the ability to interact with things around them without us having to do anything – we're going to add gravity (well, motion that really closely resembles gravity) to some planets that we're going to make. Yes, we're going to put together a solar system simulator – it's going to have its own logo and everything!

A hat tip goes to Daniel Shiffman for this tutorial. His book *The Nature of Code* explains the concepts found here and more in far greater detail. All of his code is written in Java (Processing), but you should be able to convert it to Python with a little bit of work.

## Understanding gravity

"Wait, we've done gravity before!" Yes, we have, but only 'kind of'. In the past, we've added a force we've called gravity to certain objects to make them fall to the bottom of the window. However, the force that we called gravity wasn't very dynamic; no matter the object's size or velocity, that force would just add to the Y value of an object until it hit the bottom– not very interesting. For this new kind of gravity, we're going to be using something called vectors. A vector is a value which describes two things: direction and magnitude. With these we can calculate the effect of one object with mass on the velocity (the speed and direction) of another object with mass. This program is bigger than anything we've made before, but we're not going to look at a great deal of it. Most of the things, such as drawing images or handling key presses, we've done before, so we're not going to revisit them; instead, we're going to focus on describing gravity and implementing that in our code. This is going to take some serious thinking, so if you don't understand everything the very first time, don't worry: we promise almost everyone else who reads this will probably feel like they might go through it again.



**Above** An illustration demonstrating the gravitational attraction of two bodies, one orbiting the other

## CAUTION: MATHS AHEAD

### So, what is this 'gravity' business, anyway?

In the real world, gravity follows a rule called the inverse square law, which is this:

"The gravitational attraction between two point masses is directly proportional to the product of their masses and inversely proportional to the square of the separation distance. The force is always attractive and acts along the line joining them."

Yeah, we know, our heads hurt when we first read that too, so what does it mean? It's actually really simple: it means the force acting on something reduces as the distance increases. So however strong the pull of gravity on something is – like, for example, the Earth pulling on a football 1 foot in the air – if we were to move the same object so that it was 3 feet away from the gravity source, the force will be 1/9th as strong, that is 1/3 squared or 1 / distance$^2$. The same law applies to more than just gravity - it affects light and sound too - but that's not important here. What is important is why we've made you read that long science-y statement. Now, we get a feeling for how gravity should work – far away = less force, closer = more force – but the last bit of that statement is equally important…

"The force is always attractive and acts along the line joining them."

Gravity always pulls, it never repels, and it always pulls in the direction of the objects it's pulling. It's because of this truth that we're going to use vectors to simulate gravity. Using vectors, we can calculate the direction of each object in relation to another and adjust it to the force of gravitational attraction accordingly. The result is… gravity happens.

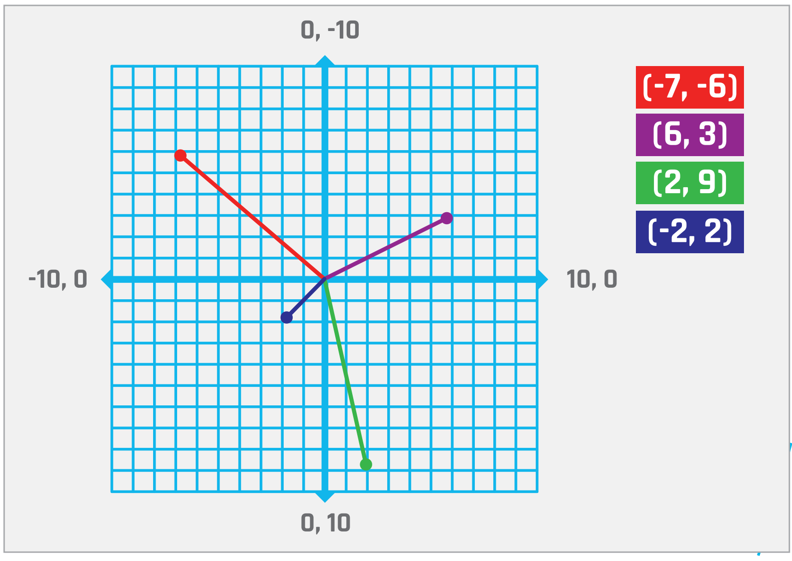
### V is for vector

So now we've got an understanding of how gravity works, it's time to take a little look at what a vector is. You can think of a vector like an arrow: it has two values, an X and a Y, and together these point in a direction. For example, if we were to draw a line from (0,0) along a vector of (8, 4) on a grid, it would point down and to the right; for every unit travelled along the X axis (pixels, centimetres, inches, fathoms, the unit type doesn't matter), 0.5 units would be travelled along the Y axis. If we were to draw another line from (0,0) along a vector of (−1, −2), the line would travel to the left and up; for each unit travelled along the X axis, two would be traversed along the Y axis.

So with vectors we can describe direction, but we can also express something called magnitude. The magnitude of the vector is the length of the line drawn between (0,0) and the vector on a grid, but we can also think of the magnitude as an amount or size of something; for example, we could use it as speed.

When we use vectors to describe direction, it often helps to normalise them. This means we take a vector, such as (1, 3), and turn each value into a value somewhere between −1 and 1 by dividing it by the magnitude. For instance, the vector (1,3) would be normalised to (0.316, 0.948), while (−8, 2.4) would normalise to (−0.957, 0.287). We can normalise our values by dividing them by our magnitude. Normalising our values makes it much easier to affect things with force and direction. By having a value between −1 and 1, we have only an indication of direction. When we have that, we're free to adjust it by any value to suit our needs; for instance, we could multiply the values by a speed value to give something motion.

**Below** A diagram showing vectors on a grid



## A speedy overview

Right, so that was a lot to take in. A quick recap: gravity always attracts in the direction of something with a mass; vectors describe a direction and a magnitude which is an amount of something, such as speed; and vectors can be shrunk down to a value between -1 and 1 to describe only a direction, through a process called normalisation. Now that we have these key concepts bouncing around inside our heads, it's finally time to start looking at some Pygame stuff.

Like we said earlier, we're going to skip over explaining a lot of the code for this tutorial – it's all stuff we've done before – but for the sake of clarity we'll do a quick pass over the functions, what they do, and the order they're called in. With the knowledge we've gained above, we're going to have a solar system simulator that moves planets around with gravity, based on their mass and velocity.

On lines 1-30 of **simulator.py** we have all of the variables we need to run our program. The **`import`** statements at the top of our script are almost identical to our previous programs, with one exception - **`import solarsystem`** on line 5. This is not a module like the other **`import`** statements but a custom script written for this tutorial, and you can grab it from GitHub. Just drop it in the same folder as simulator.py; it simply creates new planets for our simulator and doesn't need to be in the main simulator.py code, as our games are going to start to get messy if everything is in one script!

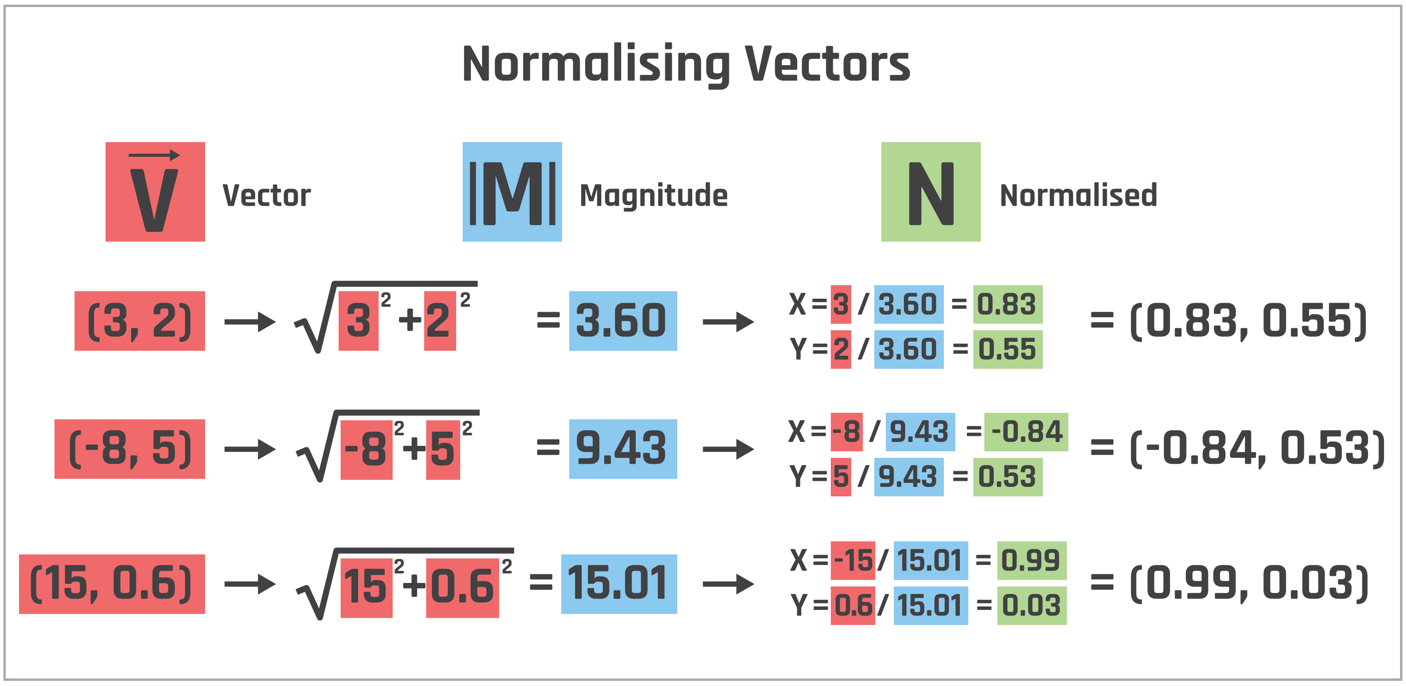
Lines 31-54 contain the functions **`drawUI()`**, **`drawPlanets()`** and **`drawCurrentBody()`**. These are responsible for drawing the elements of our program to our window. All of these are called once every time the main loop runs, in the order **`drawUI()`**, **`drawPlanets()`**, and then **`drawCurrentBody()`**. The **`currentBody`** function is responsible for drawing the planet that the user is currently dragging around the window, before letting it go to affect other planets with its gravity.

Lines 56-82 involve the **`calculateMovement()`** function. It's here that we make all of the gravity magic happen. It gets called in the main loop, just before **`drawPlanets()`**. This is the clever bit of our program and we'll work through every single line in a moment.

Lines 84-106 handle the mouse and system events. When our player clicks somewhere in our window, **`handleMouseDown()`** is run and checks whether or not our user clicked in one of the planet tabs at the bottom of our window with **`checkUIForClick()`**. If they have, **`checkUIForClick()`** will return the name of that planet and it will be created with **`solarsystem.makeNewPlanet()`**, the only function that we imported with **`import solarsystem`** at the start of our script.

Finally we have lines 109-165, our now-fabled 'main' loop. Just like in our previous programs, it's from here that we call functions to handle user interactions and update our surface. The function calls on lines 146-157 are where we update and draw our planets.

**Below** The maths of calculating the magnitude of a vector and normalising it


Normalising Vectors
V Vector
|M| Magnitude
N Normalised
(3, 2) → √(3² + 2²) = 3.60 → X = 3 / 3.60 = 0.83, Y = 2 / 3.60 = 0.55 = (0.83, 0.55)
(-8, 5) → √(-8² + 5²) = 9.43 → X = -8 / 9.43 = -0.84, Y = 5 / 9.43 = 0.53 = (-0.84, 0.53)
(15, 0.6) → √(15² + 0.6²) = 15.01 → X = -15 / 15.01 = 0.99, Y = 0.6 / 15.01 = 0.03 = (0.99, 0.03)

## The movement of the Spheres

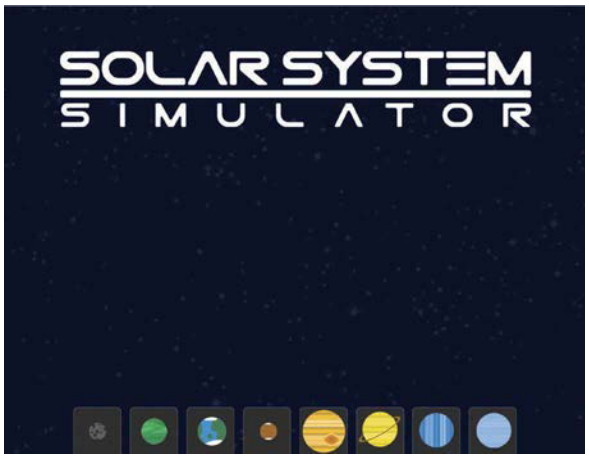
So, let’s get down to it. If you fire up the simulator.py script, you’ll see our Solar System Simulator. After four seconds, the logo will disappear and you’ll then be able to drag one of the eight planets at the bottom to somewhere on the screen. Each planet has characteristics which loosely reflect that of its real-world counterpart. Jupiter has the greatest mass, Mercury has the least, Venus is only slightly smaller than Earth, Mars is a third of Earth’s size and so on. By clicking on a planet, we create a new planet which is stored in the **`currentBody`** variable. The latter lets us create and place a planet without affecting any other planets on the screen. It’s only when we let go of the planet that gravity is allowed to take effect on the new body in the solar system. All of this neat gravitational magic happens on lines 56-82.

Fun fact: no matter how far apart two objects are in the universe (the real universe, that is), they still have a gravitational pull on one another, even if it is infinitesimally small. If you were to place two dice a metre apart in space and leave them to their own devices, eventually the pull of the dice on one another would bring them both together without any help from another force. This effect is duplicated in our simulator. For each planet in our solar system, it has a gravitational effect on every other body in our system. To do this, we create a **`for`** loop that works through every planet in our **`celestialBodies`** list on line 58.

For each planet we have, we want to calculate its effect on every other planet in our system, so the first thing our **`for`** loop on line 58 does is create another loop to work through the rest of the planets. We don’t want to calculate the effect of a planet on itself, so before we start crunching numbers we check that **`otherPlanet`** is not the same as the planet we’re using for the gravitational pull calculations. Once we have a valid planet to affect, we can start crunching some numbers and figuring out some vectors.

The first thing we need to find is the vector between the planet and otherPlanet. We do this on line 64 with the variable **`direction`**. The variable is named ‘direction’ because it points from the coordinates of our planet to the coordinates of the otherPlanet that we’re trying to affect. Once we have the direction, we can work out the magnitude (in this case, the distance) between the two planets.

To help us work out the magnitude of our direction vector, we can use Python’s built-in maths library, specifically the **`hypot`** method which we use on line 65. If you’re curious about the actual formula for figuring out the magnitude, it’s the square root of the X squared and Y squared coordinates added to each other. Once we have our magnitude, we can use it to normalise our direction vector. Normalising our vector means we’ll have vector X and Y values that are proportional to one another but fall between -1 and 1. This is super-handy for us, because that lets us multiply our vector by any value we wish to affect our force. To normalise our vector, all we have to do is divide our direction vector by the magnitude – simple! – and we do that on line 66 with the variable **`nDirection`**.



**Left** Our Solar System Simulator on its first run

We have almost everything we need to start applying gravity, but before we do, we should limit magnitude. Strange things happen when forces are very big or very small, even in simulators, so we set a maximum for the number that magnitude can be on lines 68-72.

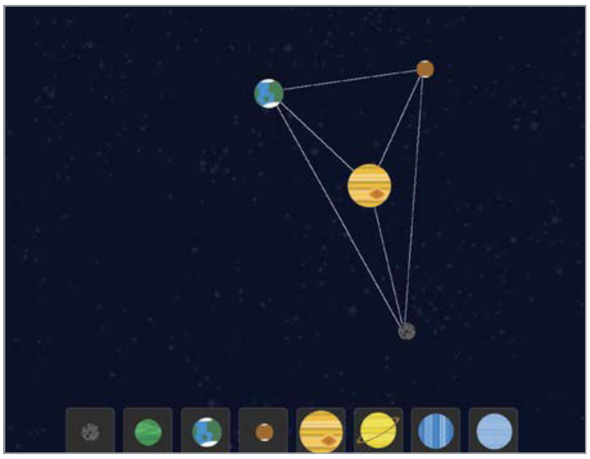
> “For each planet we have, we want to calculate its effect on every other planet”

We now have all we need to apply gravity to our planet. Cracking! But at this point, we’d be applying an arbitrary value that had nothing to do with the properties of our planets. What we want to do now is take into consideration the mass of our objects, because gravity only affects things with mass.

On line 74 we have the **`strength`** variable. Here, we calculate how much force we need to apply to each planet to generate gravity. First, we multiply the planet’s mass by the otherPlanet’s mass and multiply that by our **`gravity`** variable on line 29. The gravity value is arbitrary and we can tweak it to generate stronger or weaker gravitational effects – remember, we’re creating the illusion of gravity, not actually modelling the universe. Next, we divide that value by

**magnitude** squared – this lets our objects accelerate as they approach one another. Finally, we divide all of that by the mass of the planet we're affecting, otherPlanet. This lets our objects move slower if they are dense, and faster if they are less dense. By making it harder to move the big planets, we avoid small planets towing much larger ones around.
That's it! We now have the values we need to apply gravity to our planets. On line 75, we create a new vector. By multiplying our normalised direction vector (**nDirection**) by the **strength** value, we now have a vector with both direction and magnitude determined by the gravitational attraction of our objects! That was rather painless really, wasn't it? On lines 77 and 78 we apply this new vector to the velocities of our otherPlanet; the next time our planet is drawn, its position will have been adjusted by gravity.

The last little bit of **calculateMovement()** on lines 80 to 82 doesn't have anything to do with moving the planets – it simply draws a line between our planet and every otherPlanet that it's having an effect on. It's the line of attraction we looked at earlier, and it illustrates the directions that gravity is pulling our planets in. You can toggle this on and off with the 'A' key.


**Above** The lines of attraction drawn between planets

## Rounding up

Well, we've covered a lot this time! We've learned all about vectors and how we can use them to determine both speed and direction, a lot like velocity. We've also learned how to normalise values so they can be made to do our bidding through multiplication. We've learned how gravity works in the real world, and how we can emulate that in our simulated one. We also have some pretty neat code for handling our mouse and keyboard events. These things are never easy, but if you stick at it, you can learn anything.

## Simulator.py

```
01.  import pygame, sys, random, math
02.  import pygame.locals as GAME_GLOBALS
03.  import pygame.event as GAME_EVENTS
04.  import pygame.time as GAME_TIME
05.  import solarsystem
06.
07.  windowWidth = 1024
08.  windowHeight = 768
09.
10.  pygame.init()
11.  surface = pygame.display.set_mode((windowWidth,
     windowHeight), pygame.FULLSCREEN)
12.
13.  pygame.display.set_caption('Solar System Simulator')
14.
15.  previousMousePosition = [0,0]
16.  mousePosition = None
17.  mouseDown = False
18.
19.  background = pygame.image.load("assets/background.jpg")
20.  logo = pygame.image.load("assets/logo.png")
21.  UITab = pygame.image.load("assets/tabs.png")
22.  UICoordinates = [{"name" : "mercury", "coordinates" :
     (132,687)}, {"name" : "venus", "coordinates" : (229,687)},
     {"name" : "earth", "coordinates" : (326,687)},
     {"name" : "mars", "coordinates" : (423,687)},
     {"name" : "jupiter", "coordinates" : (520,687)},
     {"name" : "saturn", "coordinates" : (617,687)},
     {"name" : "neptune", "coordinates" : (713,687)},
     {"name" : "uranus", "coordinates" : (810,687)}]
23.
24.  celestialBodies = []
25.  currentBody = None
26.
27.  drawAttractions = True
28.
29.  gravity = 10.0
30.
31.  def drawUI():
32.       surface.blit(UITab, (131,687))
33.       surface.blit(solarsystem.images["mercury"], (158,714))
34.       surface.blit(solarsystem.images["venus"], (247,706))
35.       surface.blit(solarsystem.images["earth"], (344,704))
36.       surface.blit(solarsystem.images["mars"], (451,714))
37.       surface.blit(solarsystem.images["jupiter"], (524,692))
38.       surface.blit(solarsystem.images["saturn"], (620,695))
39.       surface.blit(solarsystem.images["neptune"], (724,697))
40.       surface.blit(solarsystem.images["uranus"], (822,697))
41.
42.  def drawPlanets():
43.
44.       for planet in celestialBodies:
45.            planet["position"][0] += planet["velocity"][0]
46.            planet["position"][1] += planet["velocity"][1]
47.            surface.blit(solarsystem.images[planet["name"]],
     (planet["position"][0] - planet["radius"],
     planet["position"][1] - planet["radius"]))
48.
49.  def drawCurrentBody():
50.
51.       currentBody["position"][0] = mousePosition[0]
52.       currentBody["position"][1] = mousePosition[1]
53.
54.       surface.blit(solarsystem.images[currentBody["name"]],
     (currentBody["position"][0] - currentBody["radius"],
     currentBody["position"][1] - currentBody["radius"]))
55.
```

**Language**

**>PYTHON**

**DOWNLOAD:**
**Github.com/seanmtracey/Games-with-Pygame**

```
def calculateMovement():

    for planet in celestialBodies:

        for otherPlanet in celestialBodies:

            if otherPlanet is not planet:

                direction = (
otherPlanet["position"][0] - planet["position"][0],
otherPlanet["position"][1] - planet["position"][1])
# The difference in the X, Y coordinates of the objects
                magnitude = math.hypot(
otherPlanet["position"][0] - planet["position"][0],
otherPlanet["position"][1] - planet["position"][1])
# The distance between the two objects
                nDirection = (
direction[0] / magnitude, direction[1] / magnitude)
# Normalised vector pointing in the direction of force

                # We need to limit the gravity
                if magnitude < 5:
                    magnitude = 5
                elif magnitude > 30:
                    magnitude = 30

                strength = ((
gravity * planet["mass"] * otherPlanet["mass"]) /
(magnitude * magnitude)) / otherPlanet["mass"]
# How strong should the attraction be?

                appliedForce = (
nDirection[0] * strength, nDirection[1] * strength)

                otherPlanet["velocity"][0] -=
appliedForce[0]
                otherPlanet["velocity"][1] -=
appliedForce[1]

                if drawAttractions is True:
                    pygame.draw.line(
surface, (255,255,255),
(planet["position"][0],planet["position"][1]),(
otherPlanet["position"][0],otherPlanet["position"][1]),
1)

def checkUIForClick(coordinates):

    for tab in UICoordinates:
        tabX = tab["coordinates"][0]

        if coordinates[0] > tabX and coordinates[0] <
tabX + 82:
            return tab["name"]

    return False

def handleMouseDown():
    global mousePosition, currentBody

    if(mousePosition[1] >= 687):
        newPlanet = checkUIForClick(mousePosition)

        if newPlanet is not False:
            currentBody = solarsystem.
makeNewPlanet(newPlanet)


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

# 'main' loop
while True:

  mousePosition = pygame.mouse.
get_pos()
  surface.blit(background, (0,0))

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

    if event.type == pygame.KEYDOWN:

      if event.key == pygame.K_ESCAPE:
        quitGame()

    if event.type == pygame.KEYUP:

      if event.key == pygame.K_r:
           celestialBodies = []
      if event.key == pygame.K_a:
           if drawAttractions is True:
                drawAttractions = False
           elif drawAttractions is False:
                drawAttractions = True

    if event.type == pygame.MOUSEBUTTONDOWN:
      mouseDown = True
      handleMouseDown()

    if event.type == pygame.MOUSEBUTTONUP:
      mouseDown = False

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

  # Draw the UI, update the movement of the planets,
  # draw the planets in their new positions.
  drawUI()
  calculateMovement()
  drawPlanets()

  # If our user has created a new planet,
  # draw it where the mouse is
  if currentBody is not None:
      drawCurrentBody()

      # If our user has released the mouse, add the new
      # planet to the celestialBodies list
      if mouseDown is False:
           currentBody["velocity"][0] = (
mousePosition[0] - previousMousePosition[0]) / 4
           currentBody["velocity"][1] = (
mousePosition[1] - previousMousePosition[1]) / 4
           celestialBodies.append(currentBody)
           currentBody = None

      # Draw the logo for the first four seconds
  if GAME_TIME.get_ticks() < 4000:
    surface.blit(logo, (108,77))

  # Store the previous mouse coordinates to create a
  # vector when we release a new planet
  previousMousePosition = mousePosition

  pygame.display.update()
```

**RICHARD HAYLER**

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

**richardhayler.blogspot.co.uk**
**coderdojoham.org**

**You'll Need**

- Red, blue, yellow, and green LEDs
- Four 470-ohm resistors
- A bunch of patch wires
- Pimoroni Explorer HAT **shop.pimoroni.com**
- The Explorer Hat library **github.com/pimoroni/explorer-hat**

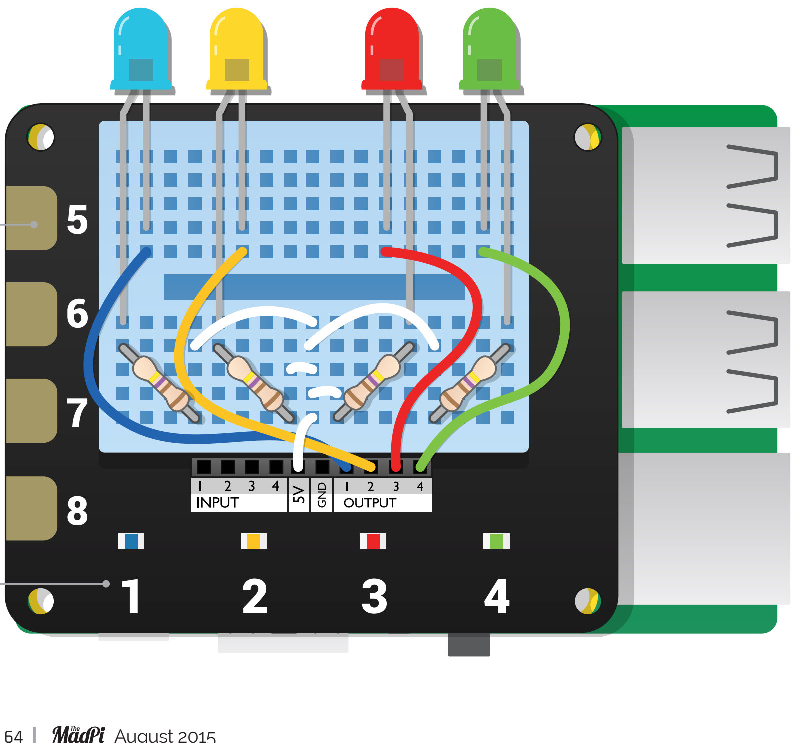
# SIMON SAYS... EXPLORE!

Simon was one of the iconic electronic games of the 1970s. Now you can create your own version using a few simple components and the marvellous Pimoroni Explorer HAT

Much of the assembly language used to create the original Simon game was developed by Dr Charles Kapps, author of one of the first books on computer programming. So perhaps it is no coincidence that creating a Simon clone is a really good way to get your head around some core coding concepts like lists and loops. With its built-in breadboard, buttons and LEDs, the Explorer HAT is the perfect choice for this sort of prototyping. There are two versions of the Explorer HAT, the basic and the Pro (which includes analogue inputs and motor drivers), but this project will work with either.

Four capacitive touchpads can be used as buttons, and each has a corresponding LED

The crocodile clip contact pads can also be used as buttons, or to attach other components



## >STEP-01

### Getting started with Explorer HAT

Like most HATs, this one is dead easy to use. Simply plug it carefully onto the GPIO pins of your Pi, then install the Explorer HAT library. This needs the I$^2$C bus on the Pi to be enabled, and there are plenty of instructions for this online. Or you could use the handy Pimoroni script:

```
curl get.pimoroni.com/i2c | bash
```

SMBUS is also required, so install that with:

```
sudo apt-get install python-smbus
```

If you don't already have it, install the Python package manager, PIP:

```
sudo apt-get install python-pip
```

And then, finally, the Explorer HAT library:

```
sudo pip install explorerhat
```

## >STEP-02

### Light 'em up

Let's test the built-in LEDs, which sit just above the buttons. Open a Python shell with **sudo** (either using the command line or IDLE) and import the Explorer HAT library:

```
>>> import explorerhat as eh
```

This enables us to access the API functions that control the Explorer HAT. To turn on all LEDs, type:

```
>>> eh.light.on()
```

You've probably guessed how to turn them all off, but let's just pick on a single LED: the red one.

```
>>> eh.light.red.off()
```

We can also address the LEDs by number (blue=0, yellow=1, etc.), which will be useful in our Simon game. Let's switch off the green LED:

```
>>> eh.light[3].off()
```

## >STEP-03
### Button it

The four capacitive touchpads below the LEDs are equally easy to control. The following line of code will return True if the first one is pressed:

```
>>>eh.touch.one.is_pressed()
```

Try running it with your finger away from the button, then again while you're touching it. One neat feature of the API is the ability to create a handler function that runs when a button is pressed or released. The code in the **button_led.py** file (downloadable from **github.com/topshed/ExplorerHat-Simon**) uses one to light up an LED when the button below it is pressed. Note that the four capacitive crocodile clip pads can also be used as buttons (5-8), but, as they don't have any corresponding LEDs, we ignore them if they've been pressed. Run the code with:

```
sudo python button_led.py
```

## >STEP-04
### Build the circuit

The Explorer HAT has four 5V outputs which we can use to control additional circuit elements. For our Simon game, we'll add four more LEDs to the breadboard which will display the pattern the player has to repeat using the buttons. Assemble the circuit shown on page 64, making sure you insert the long leg of the LED – the kinked leg in the picture. As usual, we connect each LED in series with a resistor to prevent damage. Test your wiring by opening a Python shell and typing:

```
>>> import explorerhat as eh
>>> eh.output.on()
```

All four of the circuit LEDs should light up.

## >STEP-05
### Blinkin' lights

Now let's generate a simple sequence using a useful way of controlling the LEDs and outputs. The code in **led-pattern.py** (download from GitHub) uses Python's random library to pick which of the four output LEDs is turned on then off (toggled) as we iterate through the loop ten times. Run the code a few times to test it out.

## >STEP-06
### Putting it all together

We've now explored all the functionality needed for our Simon clone. Type the code from the listing into a file called **simon_explorer.py** and run it with **`sudo python`**. One output LED should flash; press the corresponding button and you'll see all eight LEDs light up in celebration. Then you're on to the next level. Not only does the sequence get longer, but the speed of the pattern gets faster. A countdown is displayed in the Python console: you only have 20 seconds for each round before the game times you out!

To make the game even more like the original Simon, how about adding a piezoelectric buzzer and some code to play a different sound for each output LED?

## Simon_Explore.py

Language
>PYTHON 2.7

**DOWNLOAD:**
**github.com/topshed/ExplorerHat-Simon**

```
import explorerhat as eh
import time, random

target_seq = [] # List to store the sequence
global user_seq
user_seq = [] # List to hold player sequence
leds_list= [1,2,3,4] # The output LEDs

def wait_for_press(c,e): # Handler to run when button pressed
        if c > 4: # Crocodile pads don't have LEDs...
                led = c - 5 # ...so map to button LEDs
        else:
                led = c - 1 # Button LEDs are 0-3, outputs 1-4
        if e == 'press': # Turn on the appropriate button LED
                eh.light[led].on()
        else:
                eh.light[led].off()
                user_seq.append(led + 1) # Add to player sequence

print('Starting. Replay the sequence before time runs out.')
level = 0
GameOn = True # If True then a game is active
gap = 0.8 # Sets the speed of flashing
while GameOn:

        user_seq = [] # Clear sequence
        level+=1
        count = 1 # Number of steps in sequence
        print('Starting Level ' + str(level))
        for i in  range(count): # Create a target sequence
                led = random.choice(leds_list) # Pick random LED
                target_seq.append(led) # Add to sequence
        for seq_n in target_seq: # Display the sequence
                for t in range(2):
                        eh.output[seq_n-1].toggle()
                        time.sleep(gap)
        eh.touch.pressed(wait_for_press) # Wait for player to play
        eh.touch.released(wait_for_press)
        countdown = 20 # Amount of time player has for each level
        waiting = True
        while waiting:

                if (len(user_seq) == len(target_seq)) or
(countdown == 0):

                        # Did the player get it right?
                        if user_seq == target_seq:
                                waiting = False
                                time.sleep(0.5)
                                print('Correct')
                                eh.light.on() # Flash all LEDs
                        eh.output.on()
                                time.sleep(0.4)
                                eh.light.off()
                                eh.output.off()
                                gap = gap* 0.8 # Go faster
                        else:
                                waiting = False
                                GameOn = False # Game Over
                                print(
'Game Over: You reached level: ' + str(level))
                                eh.pause() # Cleanup

                time.sleep(1) # Brief pause between levels
                countdown-=1 # Decrement countdown
                print countdown # Display countdown to console
```

**MARTIN O'HANLON**

Martin 'Minecraft' O'Hanlon is an active member of the Raspberry Pi community, co-author of *Adventures in Minecraft*, and keeps an excellent account of his projects on his blog.
**stuffaboutcode.com**

**Left** Nintendo's *Splatoon* is a fun multiplayer game where each team has to paint the play area in their team colours

**Below** *Minecraft Splat* uses *Splatoon*'s brilliant game mechanic to create a fun game for you and a friend

# MINECRAFT SPLAT

Create an exciting two-player game in Minecraft: Pi, inspired by Nintendo's recent hit game Splatoon...

You can play *Minecraft: Pi Edition* in multiplayer mode when two or more Raspberry Pis on the same network join the same world. In this guide, we use this technique to create a simple versus game that works along similar lines to Nintendo's *Splatoon*, which sees two teams trying to paint the game area in their team colours.

The objective of our game is very similar: to splat (turn to your team colour) as many blocks as possible for your side, while the opposing team will also be splatting blocks for themselves and claiming your splats for themselves. You will earn points for each block that is still your colour at the end of the game, and the player with the most splats wins!

## MINECRAFT SPLAT IS SPLIT INTO 5 PARTS:

| | |
|---|---|
| 01 | Create the framework for the program and make sure your code runs. |
| 02 | Build the pitch that will appear when the game starts and be the splat battleground. |
| 03 | Splat blocks by hitting them with a sword. |
| 04 | Game over and displaying the winner. |
| 05 | Making a better game. |

## CREATE THE PROGRAM

Open Python 2 from the Programming menu. The Python Shell will appear; when it does, create a new program using **File>New Window** – it's also a good idea to save your program now, using **File>Save**.

Import the Python modules you will need:

```
from mcpi.minecraft import Minecraft
from mcpi import block
from time import sleep, time
from random import getrandbits
```

You'll need a constant to hold the colour each team will use; it's the colour of the wool block that will be used when a player splats a block. Create a list which holds two values: 13 for green and 14 for red.

```
TEAMCOLS = [13,14]
```

Create the definition for two functions, which you will complete later in this tutorial.

```
def buildPitch(mc, pos):
    pass
def splatBlock(mc, x, y, z, team):
    pass
```

You will need a list to hold the points each team has scored. The first element will be team 1's score; the second, team 2's – they should both be set to 0.

```
points = [0,0]
```

Create the connection to *Minecraft* and post a message to the screen.

```
mc = Minecraft.create()
mc.postToChat("Minecraft Splat")
```

At this point, you can run your program and if everything is set up, you should see the 'Minecraft Splat' message posted to the screen.

Now start up *Minecraft: Pi Edition*. Create a new game and then run your program by clicking **Run>Run Module**.

## BUILD THE PITCH

The game needs a pitch where the action can take place; it's a glass 'room' with two glass walls running down the middle.

Find the **buildPitch** function in your program:

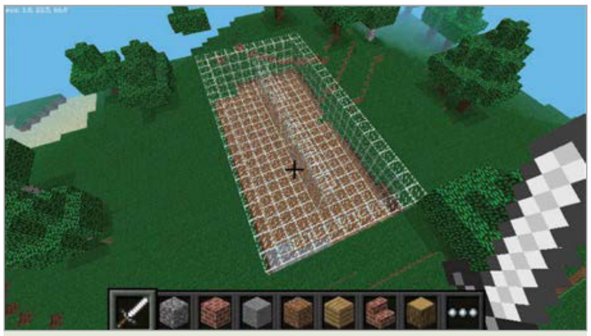
```
def buildPitch(mc, pos):
    pass
```

The *Minecraft* connection, **mc**, and a position, **pos**, where the pitch should be built, should be passed to the function.

Delete the **pass** statement and replace it with the following code, which will create a cube of glass blocks. Then create a cube of air inside it before building the central walls of glass.

```
def buildPitch(mc, pos):
    # glass cube
    mc.setBlocks(pos.x - 5, pos.y - 1, pos.z
- 10,
            pos.x + 5, pos.y + 3, pos.z + 10,
            block.GLASS.id)
    # hollow it out
    mc.setBlocks(pos.x - 4, pos.y, pos.z - 9,
            pos.x + 4, pos.y + 3, pos.z + 9,
            block.AIR.id)

    # add 2 walls down the middle
    mc.setBlocks(pos.x, pos.y, pos.z - 7,
            pos.x, pos.y + 3, pos.z - 1,
            block.GLASS.id)
    mc.setBlocks(pos.x, pos.y, pos.z + 1,
            pos.x, pos.y + 3, pos.z + 7,
            block.GLASS.id)
```

The **buildPitch** function now needs to be called from your program. Add the following code to the end of the program to get the player's position and call the function.

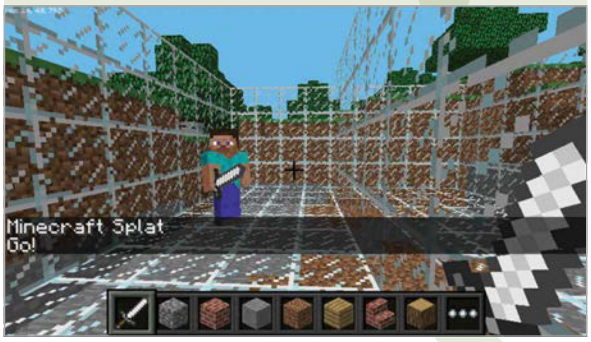


```
pos = mc.player.getTilePos()
buildPitch(mc, pos)
```

Before the game starts, you should also include a delay, to let the players get ready, and a message to let them know the game has started.

```
sleep(3)
mc.postToChat("Go!")
```

Run the program. You should see the pitch appear around your player and the message to 'Go!'.


Minecraft Splat
Go!

## SPLATTING BLOCKS

The blocks of the pitch's walls and floor can be splatted by hitting them (right-click) with a sword – when you splat a glass block, it'll turn it into a wool block of your team's colour; splatting a block belonging to the opposition will turn it back to glass.

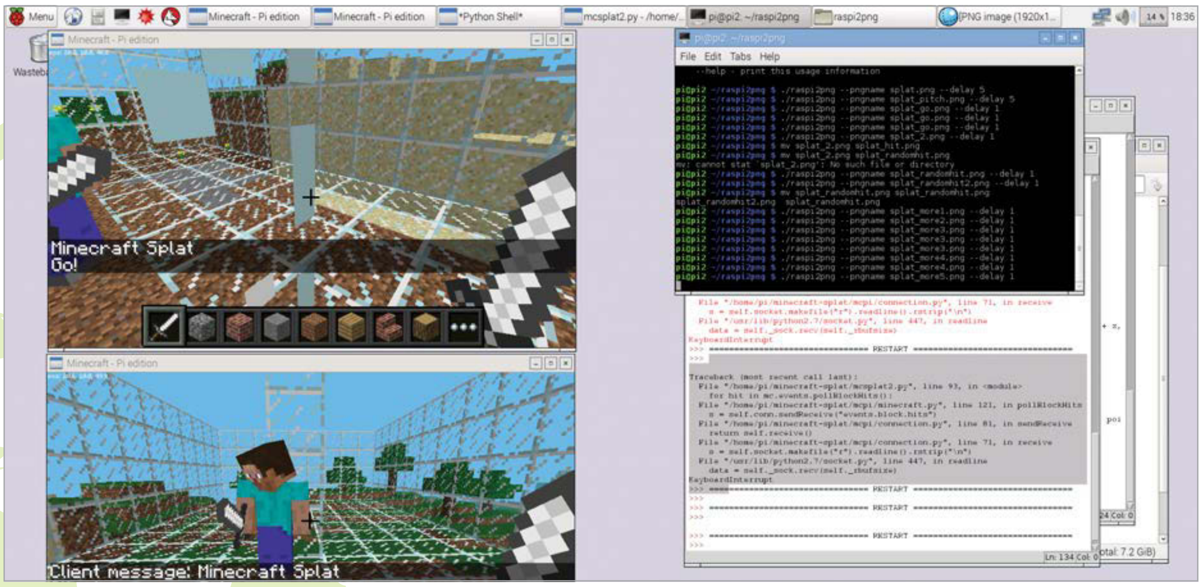
You earn points for each block splatted with your team's colour, and the opposition will lose a point for each block you turn back to glass.

Find the **splatBlock** function in your program:

```
def splatBlock(mc, x, y, z, team):
    pass
```

Change the function so that it splats the block at the position **x**, **y**, **z** for **team**, which are variables passed to the function. When executed, the function will return the number of points scored for each team.

Delete the **pass** statement and create a list which will hold the points scored for each team:



```
  --help - print this usage information

pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat.png --delay 5
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_pitch.png --delay 5
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_go.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_go.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_go.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_2.png --delay 1
pi@pi2 ~/raspi2png $ mv splat_2.png splat_hit.png
pi@pi2 ~/raspi2png $ mv splat_2.png splat_randomhit.png
mv: cannot stat 'splat_2.png': No such file or directory
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_randomhit.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_randomhit2.png --delay 1
pi@pi2 ~/raspi2png $ mv splat_randomhit.png splat_randomhit.png
splat_randomhit2.png  splat_randomhit.png
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more1.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more2.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more3.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more3.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more3.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more4.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more4.png --delay 1
pi@pi2 ~/raspi2png $ ./raspi2png --pngname splat_more5.png --delay 1
```

```
  File "/home/pi/minecraft-splat/mcpi/connection.py", line 71, in receive
    s = self.socket.makefile("r").readline().rstrip("\n")
  File "/usr/lib/python2.7/socket.py", line 447, in readline
    data = self._sock.recv(self._rbufsize)
KeyboardInterrupt
>>> ================================ RESTART ================================
>>>

Traceback (most recent call last):
  File "/home/pi/minecraft-splat/mcsplat2.py", line 93, in <module>
    for hit in mc.events.pollBlockHits():
  File "/home/pi/minecraft-splat/mcpi/minecraft.py", line 121, in pollBlockHits
    s = self.conn.sendReceive("events.block.hits")
  File "/home/pi/minecraft-splat/mcpi/connection.py", line 81, in sendReceive
    return self.receive()
  File "/home/pi/minecraft-splat/mcpi/connection.py", line 71, in receive
    s = self.socket.makefile("r").readline().rstrip("\n")
  File "/usr/lib/python2.7/socket.py", line 447, in readline
    data = self._sock.recv(self._rbufsize)
KeyboardInterrupt
>>> ================================ RESTART ================================
>>>
>>> ================================ RESTART ================================
>>>

>>> ================================ RESTART ================================
>>>
```

> You earn points for each block splatted with your team's colour, and the opposition will lose a point for each block you turn back to glass

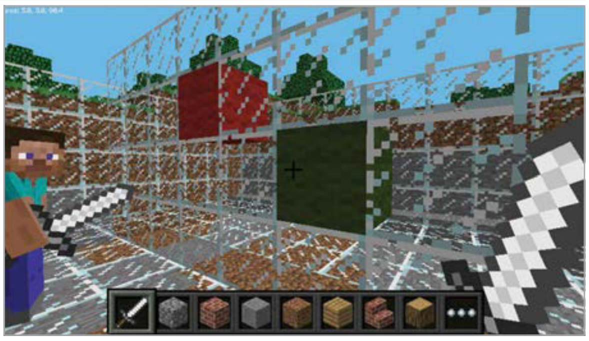
```
def splatBlock(mc, x, y, z, team):
    pointsScored = [0,0]
```

The variable **team**, which is passed into **splatBlock**, will hold either a 0 or 1 depending on which team splatted the block. Use this value to create a variable to hold the other team:

```
        otherTeam = 1 - team
```

Check to see if the block that was hit was a glass block; if it was, turn it into a wool block of the team's colour, and increase the team's score by 1:

```
  blockHit = mc.getBlockWithData(x, y, z)
      if blockHit.id == block.GLASS.id:
          mc.setBlock(x, y, z, block.WOOL.
id, TEAMCOLS[team])
          pointsScored[team] += 1
```



If the block isn't glass, check to see if it's a wool block of the other team's colour before turning it back to glass and decreasing the other team's score:

```
  elif blockHit.id == block.WOOL.id:
      if blockHit.data == TEAMCOLS[otherTeam]:
          mc.setBlock(x, y, z, block.GLASS.id)
              pointsScored[otherTeam] -= 1
```

The last step in the **splatBlock** function is to return the number of points scored:

```
      return pointsScored
```

Now that the **splatBlock** function is complete, you need to add to the code at the bottom of your program which will start the game.

You will find out how many players are in the game, create a loop which will continue until the end of the game, and call **splatBlock** each time a block is hit.

Get a list of players currently in the game, and the time the game started, and store them in variables:

```
  players = mc.getPlayerEntityIds()
  start = time()
```

Set the variable **gameOver** to False before creating a **while** loop which will continue until **gameOver** is set to True when the game finishes.

**Left** *Splatoon* on the Nintendo Wii U console is colourful and fun

```
gameOver = False
while not gameOver:
```

Use **pollBlockHits()** to find out if any blocks have been hit, before looping through each 'hit' with a **for** loop:

```
blockHits = mc.events.pollBlockHits()
    for hit in blockHits:
```

Every player in *Minecraft* has an entity ID and these are held in the **players** list you created earlier. The player's position in the list will determine what team they are on: even = team 1, odd = team 2. Use the **players** list and the entity ID of the player who hit the block to work out what team they are on.

```
team = players.index(hit.entityId) % 2
```

Call the **splatBlock** function, passing the position of the block which was hit and the team who hit it, and add the points scored to the total points for the team.

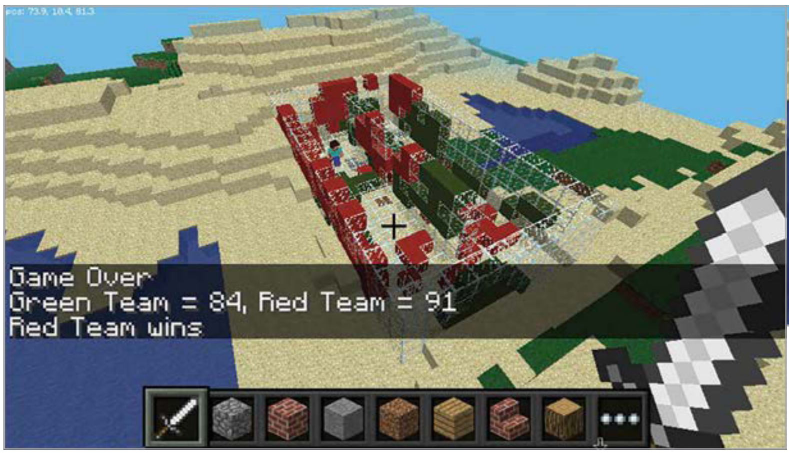
```
pointsScored = splatBlock(mc,
   hit.pos.x, hit.pos.y, hit.pos.z, team)
   points[0] += pointsScored[0]
   points[1] += pointsScored[1]
```

Run your program and, as before, the pitch should appear around your player. Now, however, hitting blocks (right-clicking while holding a sword) should turn the blocks to coloured wool. You could even get a friend to join your game and test turning your opponent's blocks back to glass.

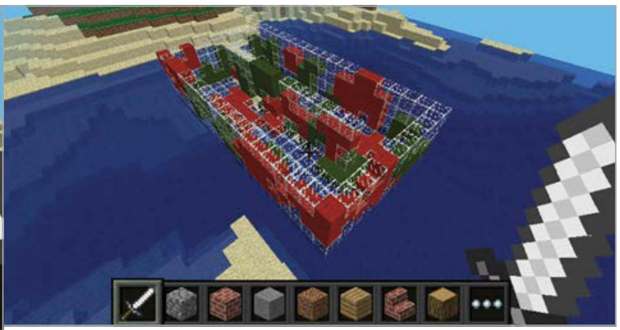
As you haven't created the code to end the game, the program will continue for ever. You can use **CTRL+C** or click **Shell>Restart Shell** in the Python Shell to end the program.

## SPLATOON

Splatoon is a refreshing take on team-based combat games produced by Nintendo for the Wii U console. While other popular multiplayer combat games like Battlefield or Call of Duty centre around war and destruction, Splatoon takes a more colourful approach to the formula by tasking its players to paint the playing field in their team colours. The team with the most paint at the end of the round wins. The 'Inklings' you control can use splatter guns and rollers in their quest to conquer their opponents. It's a brilliant family-friendly game and a great concept Martin O'Hanlon has kindly transformed into the fun Minecraft: Pi game for you to play.



Game Over
Green Team = 84, Red Team = 91
Red Team wins



"Find a friend with a Raspberry Pi, challenge them to a game"

## GAME OVER

Each match is 30 seconds long and the game is over when the time runs out. Under the **while** loop, you need to check whether the time now minus the time the game started is greater than 30 seconds. Once the game is over, you should post the team's points to the chat window, along with the winner.

```
    if time() - start > 30:
        gameOver = True
        mc.postToChat("Game Over")
        mc.postToChat(
"Green Team = " + str(points[0]))
        mc.postToChat(
"Red Team = " + str(points[1]))
        if points[0] > points[1]:
            mc.postToChat("Green Team wins")
        else:
            mc.postToChat("Red Team wins")
```

Find a friend with a Raspberry Pi, challenge them to a game of *Minecraft Splat*, and run your program.

## MAKING A BETTER SPLAT

The splat made at the moment is less of a splat and more of a blob. If you want to take the program further, in the next section you will use randomisation to splatter the blocks around the block that was hit as well.

After your code to splat the block, loop through each of the blocks around the one which was hit:

```
    for hit in blockHits:
        team = players.index(hit.entityId) % 2

            pointsScored = splatBlock(
                mc, hit.pos.x, hit.pos.y,
hit.pos.z, team)

            points[0] += pointsScored[0]
            points[1] += pointsScored[1]

            for x in [-1, 0, 1]:
                for y in [-1, 0, 1]:
                    for z in [-1, 0, 1]:
```

Using the code **getrandbits(1)**, you can randomly generate a 1 or 0, giving a 50/50 chance of it being 1 – if it is, splat the block for the team and add the points to the total.

```
        if getrandbits(1) == 1:
            pointsScored = splatBlock(mc,
                hit.pos.x + x,
                hit.pos.y + y,
                hit.pos.z + z,
                team)
            points[0] += pointsScored[0]
            points[1] += pointsScored[1]
```

Run your program again. Now, each time you splat a block, it should randomly splatter the blocks around it, too.

This is just one improvement you can make to the game; the only limit is your imagination. How will you take it forward and make it your own?

The code for *Minecraft Splat* can be found on GitHub at **github.com/martinohanlon/minecraft-splat**

Language
>PYTHON

## MCSplat.py

```python
# import modules
from mcpi.minecraft import Minecraft
from mcpi import block
from time import sleep, time
from random import getrandbits

TEAMCOLS = [13,14]

def buildPitch(mc, pos):
    # create the glass cube playing area
    mc.setBlocks(pos.x - 5, pos.y - 1, pos.z - 10,
                 pos.x + 5, pos.y + 3, pos.z + 10,
                 block.GLASS.id)

    # hollow it out
    mc.setBlocks(pos.x - 4, pos.y, pos.z - 9,
                 pos.x + 4, pos.y + 3, pos.z + 9,
                 block.AIR.id)

    # add 2 walls down the middle
    mc.setBlocks(pos.x, pos.y, pos.z - 7,
                 pos.x, pos.y + 3, pos.z - 1,
                 block.GLASS.id)

    # add 2 walls down the middle
    mc.setBlocks(pos.x, pos.y, pos.z + 1,
                 pos.x, pos.y + 3, pos.z + 7,
                 block.GLASS.id)

def splatBlock(mc, x, y, z, team):

    pointsScored = [0,0]

    # who is the other team?
    otherTeam = 1 - team

    # what type of block has been hit?
    blockHit = mc.getBlockWithData(x, y, z)
    # has a glass block been hit?
    if blockHit.id == block.GLASS.id:
        # claim it for the team
        mc.setBlock(
x, y, z, block.WOOL.id, TEAMCOLS[team])
        # increase the team's score
        pointsScored[team] += 1

    # was it a wool block?
    elif blockHit.id == block.WOOL.id:
        # if other team's colour turn it back to GLASS
        if blockHit.data == TEAMCOLS[otherTeam]:
            mc.setBlock(x, y, z, block.GLASS.id)
            # reduce the other team's score
            pointsScored[otherTeam] -= 1

    return pointsScored

# set up points
points = [0,0]

# create connection to Minecraft
mc = Minecraft.create()

# post the message to the screen
mc.postToChat("Minecraft Splat")

# find out the host player's position
pos = mc.player.getTilePos()

# build the pitch
buildPitch(mc, pos)

sleep(3)

mc.postToChat("Go!")

# get a list of the players
players = mc.getPlayerEntityIds()

start = time()

gameOver = False
# continue till the end of the game
while not gameOver:

    # has a block been hit?
    blockHits = mc.events.pollBlockHits()
    for hit in blockHits:

        # which team was it?
        team = players.index(hit.entityId) % 2

        pointsScored = splatBlock(
            mc, hit.pos.x, hit.pos.y, hit.pos.z, team)

        # update the points
        points[0] += pointsScored[0]
        points[1] += pointsScored[1]

        # splat blocks around it
        for x in [-1, 0, 1]:
            for y in [-1, 0, 1]:
                for z in [-1, 0, 1]:
                    if getrandbits(1) == 1:
                        pointsScored = splatBlock(mc,
                                hit.pos.x + x,
                                hit.pos.y + y,
                                hit.pos.z + z,
                                team)

                        # update the points
                        points[0] += pointsScored[0]
                        points[1] += pointsScored[1]

    # if the time has run out, set game over
    if time() - start > 30:
        gameOver = True
        mc.postToChat("Game Over")
        mc.postToChat("Green Team = " + str(points[0]))
        mc.postToChat("Red Team = " + str(points[1]))
        if points[0] > points[1]:
            mc.postToChat("Green Team wins")
        else:
            mc.postToChat("Red Team wins")
```

π))) Sonic Pi PART 1

**SAM AARON**

Sam 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**

# LIVE CODING

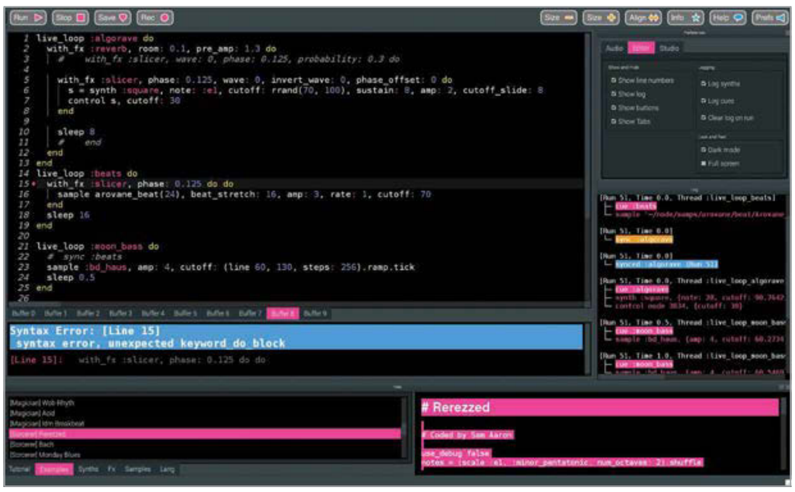
**You'll Need**

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

Digital musician and Cambridge Computer Lab researcher **Sam Aaron** starts a new Sonic Pi tutorial series by introducing the art of live coding

The laser beams sliced through the wafts of smoke as the subwoofer pumped bass deep into the bodies of the crowd. The atmosphere was rife with a heady mix of synths and dancing. However, something wasn't quite right in this nightclub. Projected in bright colours above the DJ booth was futuristic text, moving, dancing, flashing. This wasn't fancy visuals; it was merely a projection of Sonic Pi running on a Raspberry Pi. The occupant of the DJ booth wasn't spinning discs; she was writing, editing, and evaluating code. Live. This is Live Coding.

This may sound like a far-fetched story from the future, but coding music like this is a growing trend and is often described as live coding (**toplap.org**). One of the recent directions this approach to music-making has taken is the Algorave (**algorave.com**) – events where artists like myself code music for people to dance to. However, you don't need to be in a nightclub to live-code; with Sonic Pi v2.6+, you can do it anywhere you can take your Raspberry Pi and a pair of headphones or some speakers. Once you reach the end of this article, you'll be programming your own beats and modifying them live. Where you go afterwards will only be constrained by your imagination.


**Below** The new Dark theme for v2.6 is lovely!

```
live_loop :algorave do
  with_fx :reverb, room: 0.1, pre_amp: 1.3 do
  | #    with_fx :slicer, wave: 0, phase: 0.125, probability: 0.3 do

    with_fx :slicer, phase: 0.125, wave: 0, invert_wave: 0, phase_offset: 0 do
      s = synth :square, note: :e1, cutoff: rrand(70, 100), sustain: 8, amp: 2, cutoff_slide: 8
      control s, cutoff: 30
    end

    sleep 8
    #    end
  end
end
live_loop :beats do
  with_fx :slicer, phase: 0.125 do do
    sample arovane_beat(24), beat_stretch: 16, amp: 3, rate: 1, cutoff: 70
  end
  sleep 16
end

live_loop :moon_bass do
  # sync :beats
  sample :bd_haus, amp: 4, cutoff: (line 60, 130, steps: 256).ramp.tick
  sleep 0.5
end
```

```
Syntax Error: [Line 15]
 syntax error, unexpected keyword_do_block
[Line 15]:   with_fx :slicer, phase: 0.125 do do
```

```
# Rerezzed

# Coded by Sam Aaron

use_debug false
notes = (scale :e1, :minor_pentatonic, num_octaves: 2).shuffle
```

## Live loop

The key to live coding with Sonic Pi is mastering the **live_loop**. Let's look at one:

```
live_loop :beats do
  sample :bd_haus
  sleep 0.5
end
```

There are four core ingredients to a **live_loop**. The first is its name. Our **live_loop** above is called **:beats**. You're free to call yours anything you want. Go crazy. Be creative. I often use names that communicate something about the music they're making to the audience. The second ingredient is the **do** word, which marks where the **live_loop** starts. The third is the **end** word, which marks where the **live_loop** finishes. Finally, there is the body of the **live_loop**, which describes what the loop is going to repeat – that's the bit between the **do** and **end**. In this case, we're repeatedly playing a bass drum sample and waiting for half a beat. This produces a nice regular bass beat. Go ahead: copy it into an empty Sonic Pi buffer and hit Run. Boom, boom, boom!

## Redefining on-the-fly

OK, so what's so special about the **live_loop**? So far it just seems like a glorified 'loop'! Well, the beauty of **live_loop**s is that you can redefine them on-the-fly. This means that while they're still running, you can change what they do. This is the secret to live coding with Sonic Pi. Let's have a play:

```
live_loop :choral_drone do
  sample :ambi_choir, rate: 0.4
  sleep 1
end
```

Now hit the Run button or press **ALT+R**. You're now listening to some gorgeous choir sounds. Now, while

*Language*
>RUBY

it's still playing, change the rate from **0.4** to **0.38**. Hit Run again. Whoa! Did you hear the choir change note? Change it back up to **0.4** to return to how it was. Now, drop it to **0.2**, down to **0.19**, and then back up to **0.4**. See how changing just one parameter on-the-fly can give you real control of the music? Now play around with the rate yourself - choose your own values. Try negative numbers, really small numbers, and large numbers. Have fun!

## Sleeping is important

One of the most important lessons about **live_loop**s is that they need rest. Consider the following **live_loop**:

```
live_loop :infinite_impossibilities do
  sample :ambi_choir
end
```

If you try running this code, you'll immediately see Sonic Pi complaining that the **live_loop** did not sleep. This is a safety system kicking in! Take a moment to think about what this code is asking the computer to do. That's right, it's asking the computer to play an infinite amount of choir samples in zero time. Without the safety system, the poor computer will try to do this and crash and burn in the process. So remember- your **live_loop**s must contain a **sleep**.

## Combining sounds

Music is full of things happening at the same time. Drums at the same time as bass at the same time as vocals at the same time as guitars... In computing we call this concurrency, and Sonic Pi provides us with an amazingly simple way of playing things at the same time. Simply use more than one **live_loop**!

```
    live_loop :beats do
      sample :bd_tek
      with_fx :echo, phase: 0.125, mix: 0.4 do
        sample  :drum_cymbal_soft, sustain: 
0, release: 0.1
        sleep 0.5
      end
    end

    live_loop :bass do
      use_synth :tb303
      synth :tb303, note: :e1, release: 4, 
cutoff: 120, cutoff_attack: 1
      sleep 4
    end
```

Here, we have two **live_loop**s: one looping quickly, making beats; another looping slowly, making a crazy bass sound.

One of the interesting things about using multiple **live_loop**s is that they each manage their own time. This means it's really easy to create interesting polyrhythmical structures and even play with phasing, Steve Reich style. Check this out:

## Steve Reich's piano phase

```
    notes = (ring :E4, :Fs4, :B4, :Cs5, 
:D5, :Fs4, :E4, :Cs5, :B4, :Fs4, :D5, :Cs5)

    live_loop :slow do
      play notes.tick, release: 0.1
      sleep 0.3
    end

    live_loop :faster do
      play notes.tick, release: 0.1
      sleep 0.295
    end
```

## Bringing it all together

In each of these tutorials, we'll end with a final example in the form of a new piece of music which draws from all of the ideas introduced. Read this code and see if you can imagine what it's doing. Then, copy it into a fresh Sonic Pi buffer and hit Run and actually hear what it sounds like. Finally, change one of the numbers, or comment and uncomment things out. See if you can use this as a starting point for a new performance – and most of all, have fun! See you next time...

```
with_fx :reverb, room: 1 do
    live_loop :time do
      synth :prophet, release: 8, note: :e1, cutoff: 90, amp: 3
      sleep 8
    end
  end

  live_loop :machine do
    sample :loop_garzul, rate: 0.5, finish: 0.25
    sample :loop_industrial, beat_stretch: 4, amp: 1
    sleep 4
  end

  live_loop :kik do
    sample :bd_haus, amp: 2
    sleep 0.5
  end

  with_fx :echo do
    live_loop :vortex do
      # use_random_seed 800
      notes = (scale :e3, :minor_pentatonic, num_octaves: 3)
      16.times do
        play notes.choose, release: 0.1, amp: 1.5
        sleep 0.125
      end
    end
  end
```

# FREQUENTLY ASKED QUESTIONS

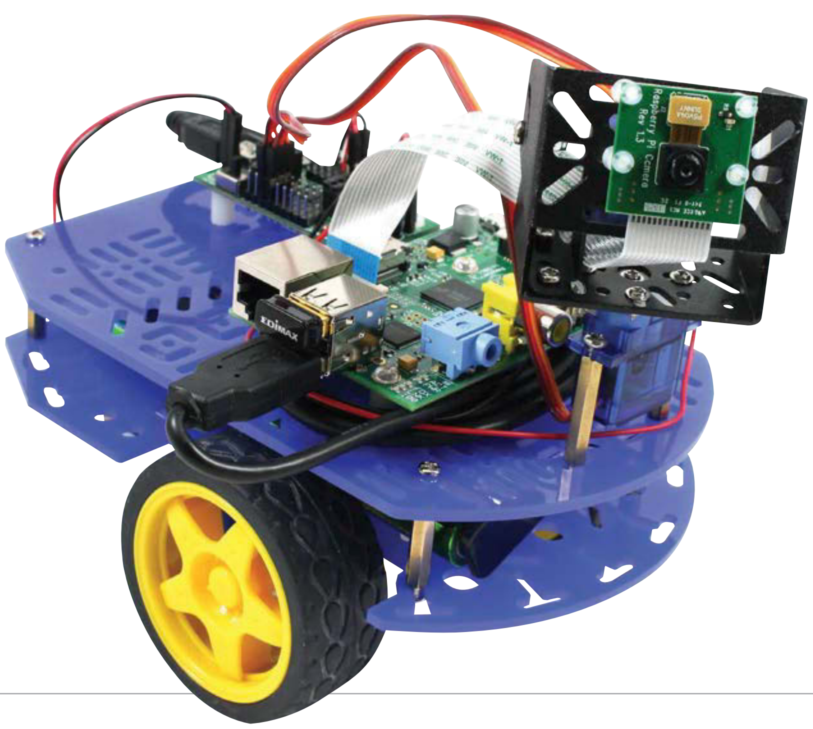
Your technical hardware and software problems solved…

**NEED A PROBLEM SOLVED?**

Email **magpi@raspberrypi.org** or find us on **raspberrypi.org/forums** to feature in a future issue.

## BUILDING ROBOTS

**I've already built a few things using the Raspberry Pi, including a media centre for my lounge TV, and a camera on my front door I can load in a browser when the doorbell rings. Next, though, I'd love to build a Raspberry Pi robot. While I've seen lots of great ready-to-go solutions in the various Raspberry Pi stores, I'd rather learn how it all works for myself than buy something ready off the shelf. Any help or assistance would be appreciated.**
**Mike Holland**



It's great to hear you're not immediately set on getting an off-the-shelf robot solution for use with your Raspberry Pi. While they're perfect in certain situations (after-school clubs or experienced enthusiasts more interested in coding than building, for example), ready-made robots don't help you understand the core principles of robotics in quite the same way as some good old elbow grease and soldering can. The Raspberry Pi Guy's robotics series is a great place to start. It's a gentle introduction that covers the basics and takes you right through basic motor control, to obstacle avoidance and line following. You can find his series on YouTube via **bit.ly/1IfZB3w**. You should also look at **piwars.org**. The organisers of the second Pi Wars competition (happening in Cambridge this September) have written a getting started guide for budding robotics enthusiasts. This year's event is set to improve on last year's by quite some margin, allowing contenders to pit their robots against each other, just like the good old days of *Robot Wars*! Learn more about the event at piwars.org or via the official Raspberry Pi blog at **bit.ly/1RMEnjZ**.

Raspberry Pi Robots - Introduction
The Raspberry Pi Guy
Raspberry Pi Robots Series
Raspberry Pi Robots - Introduction
Raspberry Pi Robots - Basic Motor Control - Episode 1
Raspberry Pi Robots - Obstacle Avoidance - Episode 2
Raspberry Pi Robots - Line Following - Episode 3

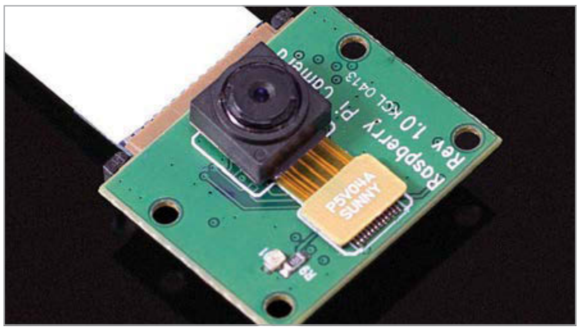
## FROM THE RASPBERRY PI FAQ
## RASPBERRYPI.ORG/HELP

**What is the user name and password for the Raspberry Pi?**
The default user name for Raspbian is 'pi' (without any quotation marks) and the default password is 'raspberry' (again, do not include the quotation marks). If this does not work, check the information about your specific distro on the downloads page (**raspberrypi.org/downloads**).

**Can I use an old VGA monitor with my Raspberry Pi?**
The chip we use supports HDMI and composite outputs, but does not support VGA. VGA is considered to be an end-of-life technology, so supporting it doesn't fit with our plans at the moment. However, if you really want to use a VGA monitor with a Raspberry Pi, then it is possible to use an HDMI-to-VGA adaptor or Gert Van Loo's VGA666 adaptor (**bit.ly/1KhZUgk**).

**What are the power requirements of my Raspberry Pi?**
The Raspberry Pi is powered by 5V micro-USB. Exactly how much current (mA) the Raspberry Pi requires is dependent on what you hook up to it and do with it. Purchasing a 1.2A (1200mA) power supply from a reputable retailer will provide you with ample power to run your Raspberry Pi for most applications. However, you may want to get a 2.5A (2500mA) one if you want to use all four USB ports on the Models B+/2B without using an external powered USB hub. The table outlines the power requirements of each model...

| Product | Recommended PSU current capacity | Maximum total USB peripheral current draw | Typical bare-board active current consumption |
|---|---|---|---|
| Pi Model A | 700mA | 500mA | 200mA |
| Pi Model B | 1.2A | 500mA | 500mA |
| Pi Model A+ | 700mA | 500mA | 180mA |
| Pi Model B+ | 1.8A | 600mA/1.2A (switchable) | 330mA |
| Pi 2 Model B | 1.8A | 600mA/1.2A (switchable) | |



### INSTALLING THE CAMERA MODULE

**Which way round does the Camera Module cable fit into the Raspberry Pi? I'm petrified of connecting it the wrong way round and breaking it.**
**David Abbot**

The Camera Module is a great (and affordable) way of adding 'eyes' to your Raspberry Pi projects. The Camera Module flex cable fits into the port nearest the top of your Raspberry Pi (the Ethernet port on a B+ or similar, or the USB port on an A+). The silver side of the cable – the bit with the actual connectors on it – should be facing towards the HDMI port, while the blue side of the cable should be facing towards the top of the board. Check out the official Camera Module video guide for further assistance: **bit.ly/1GyLctR**.

### MAGPI APP ISSUES?

**Having trouble with *The MagPi* on the App Store or Google Play? Here are your most common questions answered:**

**How do I find *The MagPi* on Google Play or the App Store?**
All you have to do is go to the search bar and type 'The MagPi' or 'Raspberry Pi' to find us.

**I've subscribed to the digital edition and I can't sign in to restore my purchases. Please help!**
Since your *The MagPi* purchases are linked to your Google or Apple accounts, there's no need to sign in at all. If you'd like to re-download your purchases on your current device, or make your purchases available on other devices, all you need to do is hit 'Subscribe' on the home screen, then 'Restore Purchases' on the next screen.

**How can I search the digital magazine for keywords?**
Finding direct references is really easy with *The MagPi* app – all you have to do is tap the screen to get the app's GUI to show, and then press the small magnifying glass icon in the top-right corner of the screen. Now, just type in your search term to find the relevant results.

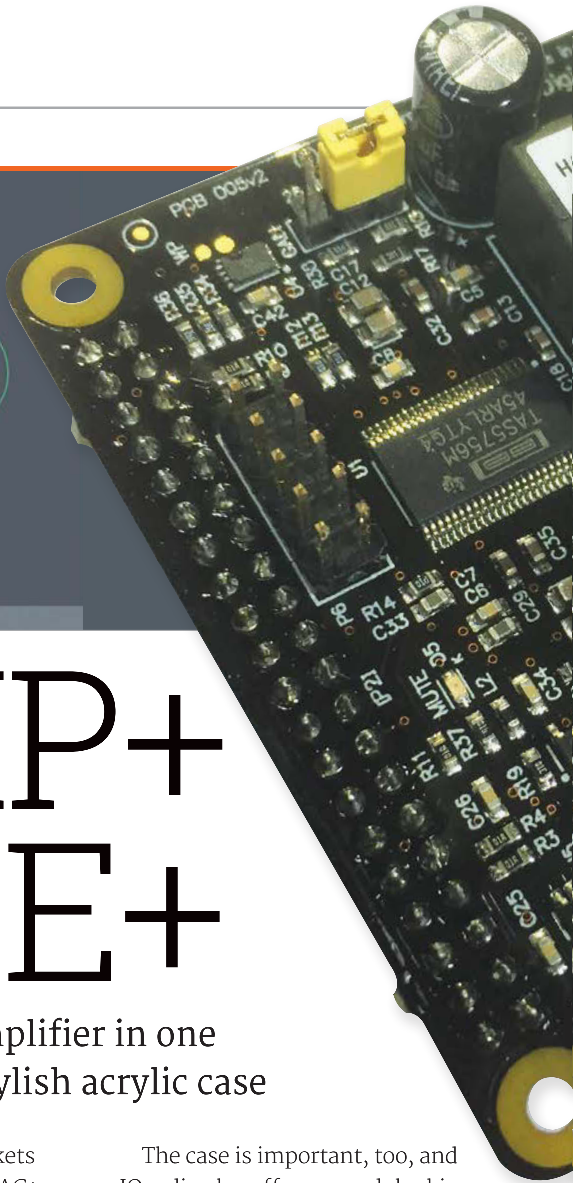
Available on the App Store

Get it on Google play

Richard Thompson
1952 Vincent Black Lightning
Acoustic Classics
Playlist position 1/1

192.168.255.54 - PuTTY

```
──────────────── AlsaMixer v1.0.27.2 ────────────────
Card: IQaudIODAC                          F1:  Help
Chip:                                     F2:  System information
View: F3:[Playback] F4: Capture  F5: All  F6:  Select sound card
Item: PCM                                 Esc: Exit

            FIR inte                           21ms    21ms
                                  00     00
 80<>80             100<>100 100<>100
<  PCM  >DSP Prog  Playback Playback Auto Mut Auto Mut Auto Mut Auto Mut
```



**Maker Says**

"A full-HD audio quality DAC and amplifier for the Raspberry Pi"

**IQaudIO**

# PI-DIGIAMP+ WITH PI-CASE+

Just add speakers: a complete hi-res audio DAC and amplifier in one small board, no soldering required, with an optional stylish acrylic case

A Raspberry Pi makes a great audio streaming device, and with the right add-ons is capable of high-end performance as well as being cheap and convenient. The Pi has its own basic on-board audio, but for the best sound you need either an external USB DAC (digital to analogue converter), or an add-on board. An add-on is preferable, since it will use the Pi's I²S interface, a dedicated digital connection that avoids USB and reduces CPU load. IQaudio already offers a DAC (the Pi-DAC+), as well as a second board (Pi-AMP+) which provides a 2×20W Class D amplifier on a board that mounts on the Pi-DAC+.

Now the firm has combined the two products into the Pi-Digi-AMP+, a single-board solution which also saves around 30% compared to buying the two previous products. The DigiAMP+ is based on the Texas Instruments TAS5756M chip and supports up to 24-bit/192kHz PCM audio. It is not all gain: you lose the line out and headphone sockets which you get with the Pi-DAC+. The result is still spectacular, though: a complete audio streamer to which you only need add speakers and your preferred music source, such as Logitech Media Server, Apple AirPlay, Spotify, or simply an attached hard drive full of music.

The Digi-AMP+ can also be used in other projects where you need audio, such as in-car entertainment, custom digital jukeboxes, robotics, and more. The board is HAT (Hardware Attached on Top) compliant, which means it complies with the official Pi recommendation for size and auto-configuration.

The Pi does not supply enough power for an audio amplifier, so the DigiAMP+ requires an additional 15V power supply, such as the XP Power VEH60US15 available from IQaudio. This also powers the Pi itself, and it is important NOT to connect USB power as well, once the DigiAMP+ has been fitted.

The case is important, too, and IQaudio also offers a good-looking acrylic case which has cut-outs for the speaker and power connections.

## Getting started

Assembling the DigiAMP+ is a matter of screwing four spacers to the Pi, mounting the board, and securing it on the spacers with screws. Note that if you are using the IQaudio case, you also need four small spacers which fit on the underside of the Pi. We were using the case, so the next thing to think about is fitting the assembled unit into the case and attaching the speaker cables.

This is a slightly tricky operation, the reason being that the speaker cables are secured with small screws that are not accessible once the top of the case is fitted. Just to make this more fun, the case is a jigsaw-like construction that falls apart until the top is fitted, which is why the guide suggests

**Related**

**HIFIBERRY AMP+**

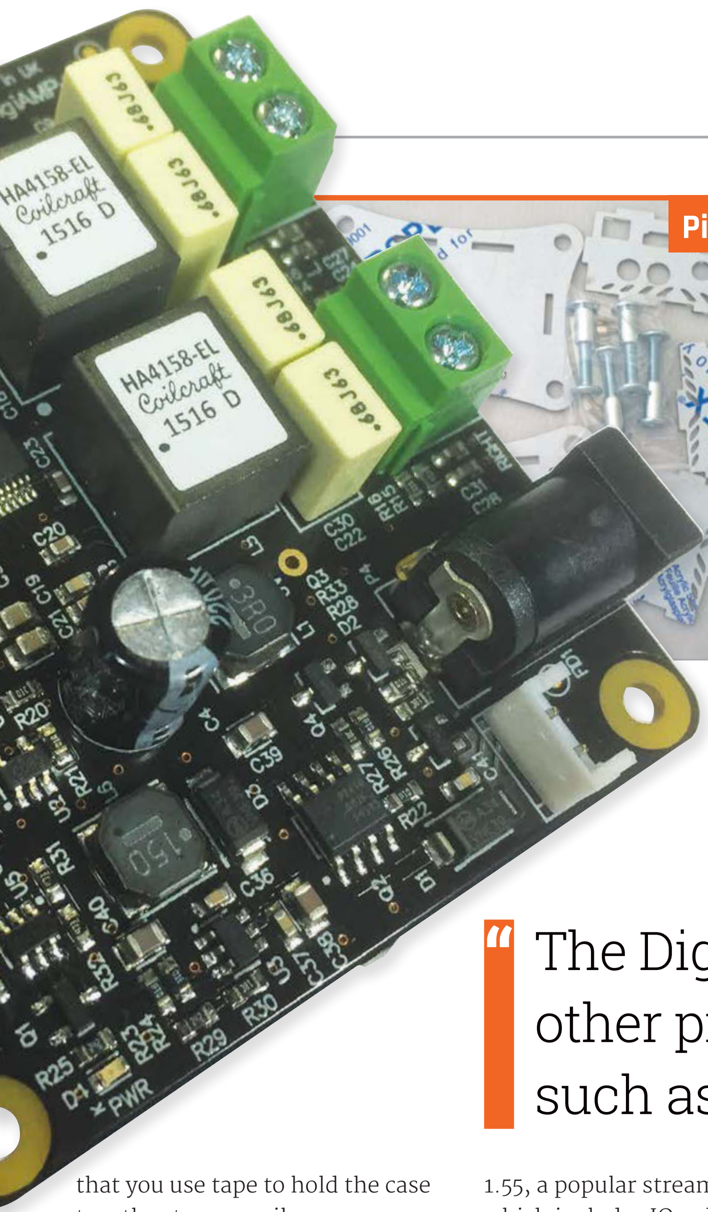
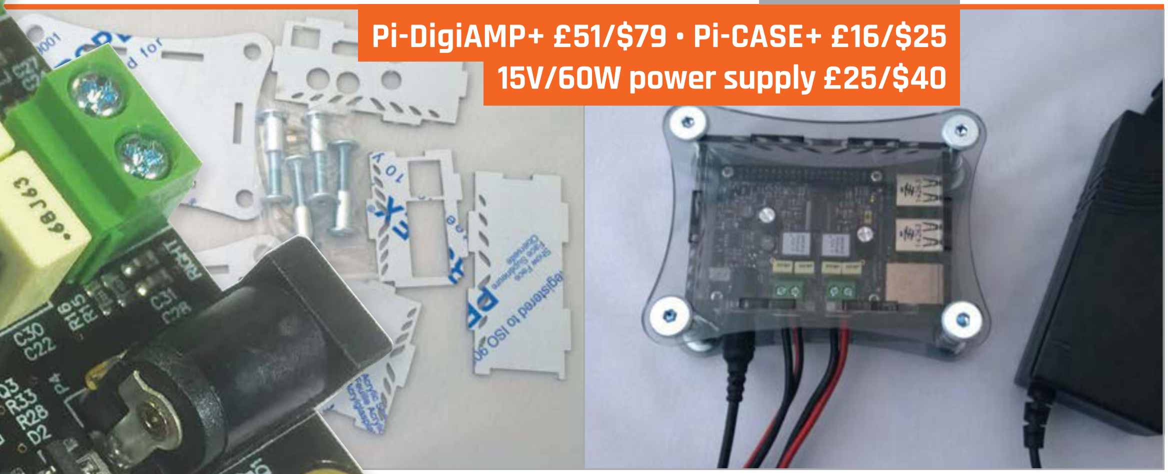
HifiBerry offers the Amp+, which includes a standard resolution DAC, as well as the DAC+ (hi-res DAC) and Digi+ (dedicated S/PDIF digital output).

£45 / $65
18V/60W power supply £18 / $26

hifiberry.com

iqaudio.com

Pi-DigiAMP+ £51/$79 · Pi-CASE+ £16/$25
15V/60W power supply £25/$40

**Far left** Volumio running with the DigiAMP+, also showing the AlsaMixer control panel running over SSH

**Above** The DigiAMP+ assembled with the Pi-Case, showing the attached speaker cables and power supply

> "The Digi-AMP+ can also be used in other projects where you need audio, such as in-car entertainment"

that you use tape to hold the case together temporarily.

It is not too difficult, but we don't really like the way the speaker cables attach in combination with this particular case. The terminals are on the small side (you can forget your chunky audiophile speaker cables), but more importantly, you have to remove the top of the case if you want to change or remove the cables, whereupon it falls apart. A better solution would be binding posts on the outside of the case, though this would add to the cost. The Pi's microSD card is also hard to fit once the case is assembled, so it's best to get this all in place first.

## Playing music

On the software side, IQaudio offers suitable Pi OS images on its site; there is also documentation to configure your existing installation. In our case, we were already using Volumio 1.55, a popular streaming client, which includes IQaudio drivers. Configuration was a matter of booting the Pi, connecting the Volumio's browser-based user interface, and enabling I²S support with the IQaudio DAC+ driver.

There was one other thing, which was a slight annoyance. The DigiAMP+ starts up muted, so you do not get any music until you have made an SSH connection to the Pi as root and run a script. The problem is that without this feature, you may get a loud start-up thump to your speakers. You can add the script (which is on the IQaudio site) to /etc/rc.local if you want it to run automatically. While you are there, you should also run AlsaMixer and set the two Playback volumes to 100%.

With all that in place, we plugged in the speakers (a pair of classic Quad bookshelf models), browsed back to Volumio, and started playing music. The Volumio volume control worked fine with the DigiAMP+, using the optimal "hardware" setting.

And how is the sound? In a word, great. This is real hi-fi, not just a cheap and cheerful streamer. We compared it to a Squeezebox playing through a traditional integrated amplifier and felt that, if anything, the DigiAMP+ beat it on clarity, with the Squeezebox sounding slightly soft in comparison. The DigiAMP+ goes loud, too: not enough for a wild party or room-shaking bass, but plenty for day-to-day listening.

## Last word

**This is true hi-fi in a compact and good-value package. The results are superb, though with a few small annoyances and no headphone socket.**

★★★★☆

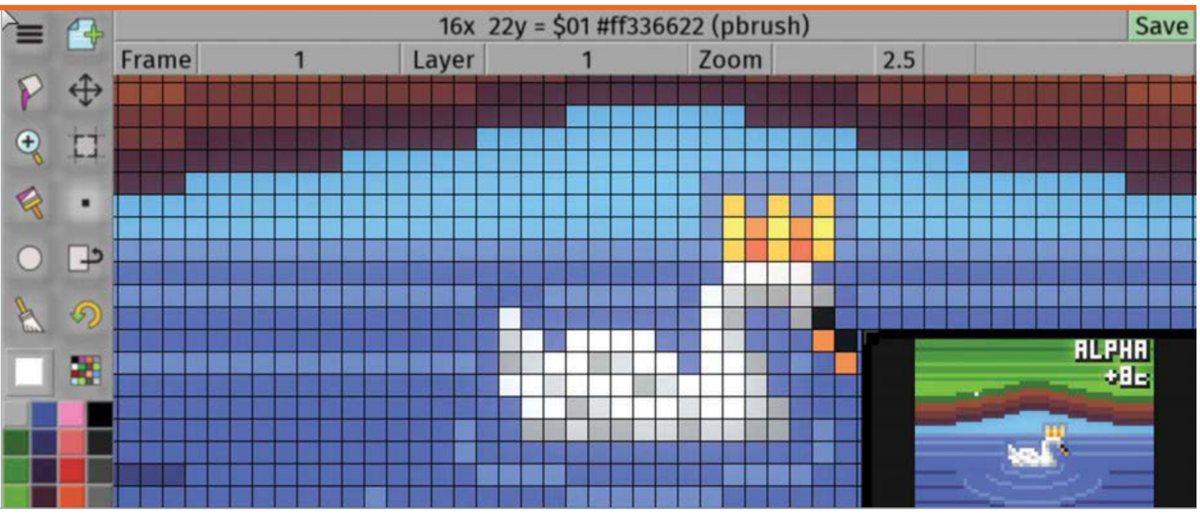
dime.lo4d.net

**$0.10 per download**

**Maker Says**

A perfect tool for creating pixel art and animation

**WetGenes**



# SWANKY PAINT

Designed by and for pixel artists, does Swanky Paint deliver on its promise of a modern Deluxe Paint? **Lucy Hattersley** finds out…

Digital artists of a certain age will remember fondly the Commodore Amiga and, in particular, Deluxe Paint. More than two decades after it was discontinued, 'DPaint' still has fans who will be pleased to hear that there's a spiritual successor. Developed by Bradford-based WetGenes, Swanky Paint will be immediately familiar to anyone who worked in digital art back in the 1980s and 1990s.

Currently available as a cross-platform alpha build, as well as a Raspberry Pi-compatible executable, the download includes versions for Linux on 32-bit and 64-bit PCs, Windows, and OS X, and there's even an APK file for Android devices. Swanky Paint is certainly flexible. Sadly, the download process isn't straightforward: available exclusively through WetGenes' Dimeload platform, users must register an account, then pay a minimum of $1 for 10 'Dimes.' These Dimes act as download tokens, each of which can be redeemed for a single download; when a new version is released, another Dime must be spent to download it.

Novel distribution method aside, Swanky Paint is straightforward to install and run. A ZIP archive, weighing in at just shy of 10MB, contains all the versions bar the Android build. Copying the files to a Raspberry Pi acts as the installation; executing the 'gamecake.raspi' application at the terminal loads the software itself. Interestingly, an X desktop session is not required: Swanky Paint can be loaded directly at the console, or the Pi configured to boot directly into Swanky Paint if required.

When it loads, Swanky Paint is attractive and simple. DPaint-inspired keyboard shortcuts make it quick to use, and there's an amazing amount of flexibility for pixel artists, including colour palettes based on classic computers like the Commodore 64 and ZX Spectrum, and a selection of rendering filters which simulate cathode-ray tube displays. Even animations are handled smoothly, and attractive images are easily created using the various brushes and tools.

The software, in alpha at present, is undeniably a work-in-progress, but the team behind it are releasing improved versions all the time. A pair of bugs encountered during testing – preventing the mouse from being detected, and spamming text to the console – were quickly fixed and a new release issued accordingly.

**Related**

**MTPAINT**

Created by Mark Tyler and maintained by Dmitry Groshev, mtPaint is a very lightweight image editor with a pixel-art bent.

FREE

mtpaint.sourceforge.net

**Last word**

**Although in the early alpha stage of release, Swanky Paint shows real promise. For fans of Deluxe Paint, it's certainly worth trying out for its low cost and wide range of features.**

★★★★☆

fuze.co.uk

**From £69 - £129**

**Maker Says**

"Nominated in 2014 and 2015 as a BETT Finalist in ICT tools for teaching and learning

**FUZE**

# THE FUZE

An attractive, robust workstation for the Raspberry Pi, but **Ian McAlpine** wonders if it is more than just a keyboard and case?

The FUZE may appear to be just a case and keyboard, but closer inspection reveals this to be a well-thought-out "workstation"solution that encourages experimentation. It is excellent for home use and ideal for schools.

The FUZE is available in three main versions: the T2-A, T2-B, and T2-C. There is also the T2-R, which includes the OWI/Maplin Robot Arm kit, and the T2-SE-R, a special edition with the same colour scheme as the original BBC Micro, complete with red function keys. All other FUZE versions have a very pleasing red and black case.

The T2-C is the entry-level workstation and comprises a sturdy, sheet aluminium case, keyboard, powered 4-port USB hub, 2A power supply, and the FUZE I/O board. More on this later.

The T2-B includes everything in the T2-C, but adds an 8GB SD card, mouse and mouse mat, 840-hole breadboard, plus an electronic components kit with jumper wires, LEDs, resistors, switches, light sensor, and a 7-segment display. There are also numerous breadboard wires of different lengths, which help to avoid the "bird's nest" appearance of projects. Additionally, the T2-B includes three superb spiral-bound manuals: a 90-page FUZE BASIC tutorial and workbook, a 168-page FUZE BASIC manual, and a "pocket-sized" 126-page FUZE BASIC Programmer's Reference Guide.

Finally, the T2-A is the same as the T2-B, but also includes a Raspberry Pi 2 Model B.

## Thoughtful design

All connections to the FUZE are on the back of the case; they include analogue audio, HDMI, SD card, Ethernet, four USB ports, and power, plus a very convenient on/off switch. The case also contains holes, which just happen to be the perfect size for popular building blocks!

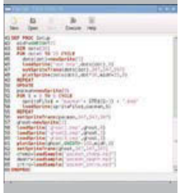
The keyboard is high quality, with a size and feel similar to an Apple Mac keyboard. Our only complaint is that the space bar is not long enough for our liking.

The FUZE I/O board is unique to FUZE. It exposes a clearly labelled Raspberry Pi 26-pin header so other add-ons can be installed, but it also breaks out the more common connections: Ground, 5V, 3.3V, PWM, plus eight buffered GPIO ports. As a bonus, the FUZE I/O board also offers four analogue in ports and one analogue out port: these are not standard on the Raspberry Pi. The FUZE I/O board and the 840-hole breadboard fit inside a trough in the top of the FUZE case, making for a very convenient and tidy work environment.

**Related**

**FUZE BASIC**

Read a review of FUZE BASIC in Issue 35 of *The MagPi*. There is also a FUZE BASIC game-programming tutorial in issues 25 to 29.



FREE

fuze.co.uk

**Last word**

**The FUZE is a holistic product, where the whole is greater than the sum of the individual parts. It is superb quality, an excellent learning platform, and significantly enhances the Raspberry Pi experience.**

★★★★★

Maker Says

It allows you to build 3D structures and shapes

Velleman

# VELLEMAN 3D PRINTING PEN

**Gareth Halfacree** asks if this can be an affordable alternative to a 3D printer…

3D printing is an undeniably hot topic in the maker community, and for good reason: a 3D printer lets you create physical objects quickly and easily, from replacement parts through to entirely novel objects. Sadly, there's a drawback: much like early 2D printers, 3D printers are currently expensive and bulky, and while both issues are being addressed with the launch of ever-cheaper and more compact 3D printers, the entry point is still in the hundreds of pounds.

That's where Velleman's 3D Printing Pen comes in. While it's not the first to hit the market, it's the first to come from a big name like Velleman rather than a crowdfunded startup. Not surprisingly, it follows the thinking of its predecessors, taking the technology of a 3D printer and making it more affordable.

The pen does this by doing away with the bulk of what makes a 3D printer. There's no printing bed, no stepper motors driving X, Y, and Z axis, nor any on-board intelligence or means to interface the device with a computer. Instead, the owner provides all these: your brain is the intelligence, and your muscles the motors.

The pen itself, then, is nothing more than the extrusion head of a 3D printer, modified to allow it to be hand-held. Bulky yet surprisingly light, the hand-feel is immediately familiar for anyone who has used similar soldering pens, but with a plastic overtone that can't help but make the unit feel cheap.

Controls are located on the side of the pen. A button allows the plastic filament to be fed through the pen, while a reverse button is used to unload unused filament when you've finished; on the other side is a sliding control that adjusts the speed at which the filament is pushed through the pen. The top hides a small adjuster for the temperature of the extruder, allowing adjustment to accommodate different qualities of filament, while the rear contains a hole for the filament to enter and a DC jack for the bundled 12V power supply.

Getting started with the pen is easy, with none of the setup required of a fully fledged 3D printer. The pen is connected to its power supply and set to preheat;

Related

**3DOODLER**

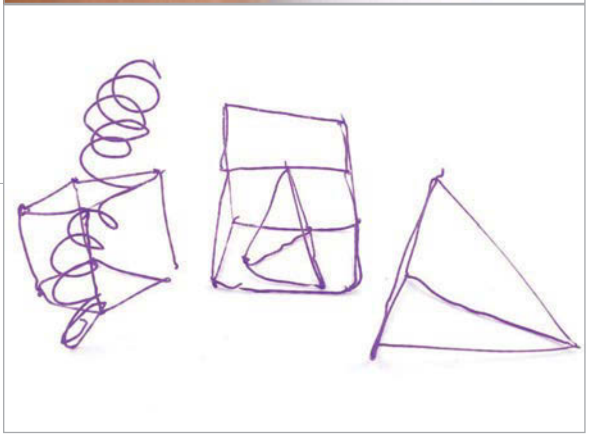
The crowd-funded 3Doodler was one of the first 3D printing pens, and offers the same features and drawbacks of the Velleman model at a slightly lower cost.

£54.99 / $99

the3doodler.com

cpc.farnell.com

£65 / $99



**Above** Even drawing the simplest of shapes is an exercise in frustration; vertical struts are particularly tricky to create

once heated, indicated by an LED, the filament is inserted into the rear and the feed button depressed until it appears from the nozzle at the front.

It's here where things start to feel a little clumsy. The box shows someone using the pen to 'draw' a three-dimensional representation of the Eiffel Tower; in practice, drawing something as simple as a wireframe cube is an exercise in frustration. Without the heated bed of a 3D printer, the filament often curls or warps as you're working; once you leave the support of a 2D surface to create the vertical struts, the filament proves too soft for too long, although it's possible to get somewhat better results by blowing across the filament as you're drawing, to cool and harden it faster.

The instructions supplied with the pen, along with three coils of PLA filament, warn that the device is for educational use only; that certainly seems to be the case in practice. Any attempts to recreate the Parisian landmark shown on the box will end in frustration, and using the pen for practical purposes seems unlikely.

Treating the pen more like a true 3D printer and constantly moving backwards and forwards to create a solid surface almost works, but the inaccuracy of a human operator means that results are unattractive. Worse, the speed control is near-unusable: a few millimetres into the slider's movement, the speed goes from 'glacial' to 'usable,' but one millimetre past that – and easily knocked into during use, thanks to the slider's position and lack of a locking system – the filament pours out too quickly to be of any use.

The thin nature of the filament extruded by the pen is another issue: when a 3D printer spends six hours carefully placing thin lines of plastic down it's no real problem, but after a few minutes – nowhere near long enough to create anything exciting – your arm will soon tire.

## Last word

**Treated like the toy it is, the Velleman 3D Printing Pen is an interesting device, but it's not something that can take the place of even the cheapest of true 3D printers.**

★★☆☆☆

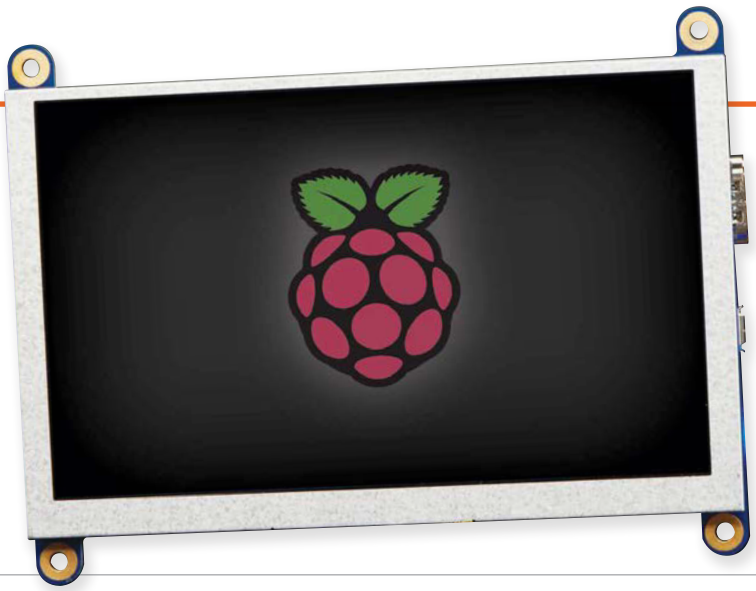
pi-supply.com

£69 / $75

Maker Says

"So small and simple, you can use this display with any computer that has HDMI output"

Adafruit



# ADAFRUIT HDMI BACKPACK

**Les Pounder** looks at a portable 5-inch touchscreen that has a clear picture, on-board hardware, and comes ready to go...

Portable screens for your Raspberry Pi are becoming more commonplace, but there has yet to be a neat and cost-effective portable display. Adafruit has stepped into the breach and built the HDMI Backpack. The screen comes in 5″ (12.7cm) and 7″ (17.8cm) sizes, both with an 800×480 resolution, which is ample for Raspberry Pi projects.

## Common connections

Connection is made via an HDMI interface which connects to the built-in TFP401 HDMI/DVI decoder. Power is supplied via an on-board micro-USB, and this – along with the HDMI – results in a very neat board. We tested the 5″ touchscreen version which comes with an AR1100 touch controller, again built into the board. The AR1100 simulates a mouse and enables control of the mouse via the touch interface. One issue that we encountered was calibrating the AR1100. The calibration software is currently Windows only and is provided by the chip manufacturer, not Adafruit. With the touchscreen configured, we followed the guidance on Adafruit's website on how to edit our config.txt file to ensure the correct screen resolution.

## Powered by micro-USB

We then powered up the screen from an external power supply – you can power the unit from the Raspberry Pi itself and there are instructions on how it's done.

At full power, the screen takes only 500mA for the display and the backlight; the latter can be controlled via PWM (pulse-width modulation), which can see the current draw reduced to 370mA. We found that when powering the unit from a Pi, it became glitchy once we unplugged the mouse and keyboard, the most likely cause being the AR1100 reacting to the event.

The screen is bright and easy to read, providing enough space to work, even for applications such as Scratch and Sonic Pi. The screen does not come with a speaker, though, thus requiring the use of an external speaker attached to the 3.5mm headphone jack.

This is a great screen that merges portability with great design and is another great Adafruit product.

Related

**HDMIPI**

Funded via a successful Kickstarter, this 1280×720 HD screen inside a custom case is a rugged platform that's suitable for children.

£75 / $117

hdmipi.com

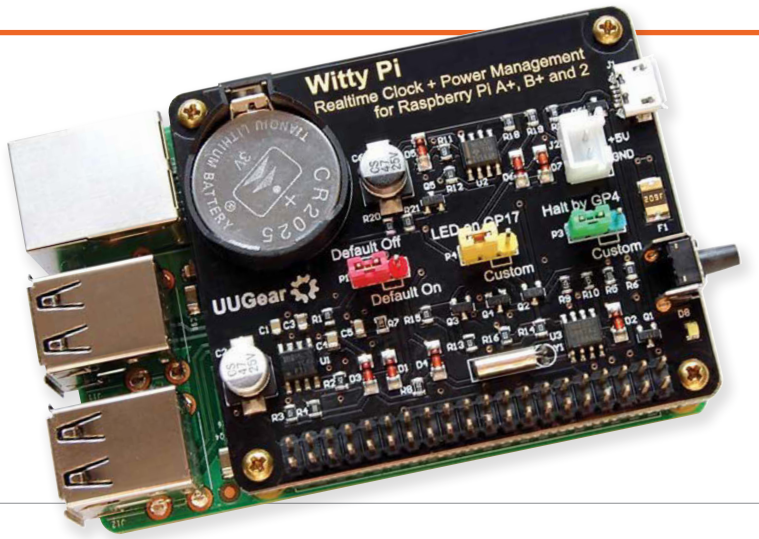
Last word

**Adafruit has always produced quality components, and this screen is no different. The portability of the board, thanks to its decoder and power, makes the HDMI Backpack great for any type of project.**

★★★★★

uugear.com

£10 / $16



**Maker Says**

"Real-time clock and power management for Raspberry Pi"

**UUGear**

# WITTY PI

Set your Raspberry Pi to routinely switch itself on and off again with this handy power management board

The Raspberry Pi is intentionally light on features, and one of the things not included is a built-in battery and real-time clock. Witty Pi is a small extension board that adds a clock battery and real-time clock functionality to the Raspberry Pi. More importantly, Witty Pi provides power management functionality, enabling it to start up and shut down a Raspberry Pi.

## Setting up the Witty Pi

Setting up the Witty Pi is easy. The expansion board connects to the 40-pin GPIO header on the Raspberry Pi, and a set of copper stand-offs can be used to mount the board securely. The power cable is connected to the Witty Pi (instead of the Raspberry Pi), and pressing the On/Off button on the Witty Pi automatically starts up (or shuts down) the Raspberry Pi.

Having an On/Off button is neat, but more important is the wittyPi.sh script used to automate power functionality. The script is used to set the date, hour, minute and second to start up and shut down (although the second function is absent from shutdown). Entering 15 07:30:00 ensures that the Raspberry Pi starts up on the 15th day of the month, at 7:30 in the morning. You use '??' as a wildcard: ?? 23:30:00 starts the Raspberry Pi at 11:30 every night, and ?? ??:30:00 starts it up at half past every hour.

There are limitations: you can't set Witty Pi to come on twice a day or every Monday, for example.

## Testing the Witty Pi

We set up Witty Pi to come on at five minutes past every hour and switch off at 15 minutes past every hour, and it performed admirably. We used a script in /etc/rc.local to run each time we started up the Raspberry Pi and log the time to a file.

Witty Pi has three jumpers. One is used to determine if the Raspberry Pi auto-starts when you connect the power. The other two can be used to connect specific GPIO pins to control startup and shutdown, offering some interesting electronic integration options. Witty Pi is also beneficial for battery-powered projects: these can run for weeks if you turn the Raspberry Pi on for brief durations rather than keeping it switched on.

**Related**

**MINI RTC MODULE**

The Mini RTC module adds a real-time clock to the Raspberry Pi so it can remember the time when it starts up. It doesn't provide power management functionality, though.

£5

thepihut.com

**Last word**

Witty Pi is a simple extension that enables you to schedule startup and shutdown for your Raspberry Pi. We would like to see more complex scheduling options, but it worked perfectly.

★★★★☆

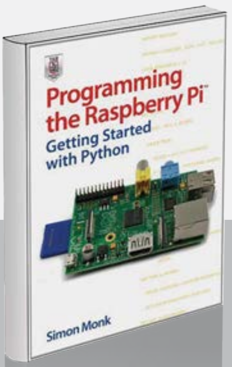
## RASPBERRY PI BESTSELLERS

## TAB ELECTRONICS

**McGraw Hill's little maker-oriented offshoot has turned out several popular titles for hobbyists...**

### PROGRAMMING THE RASPBERRY PI

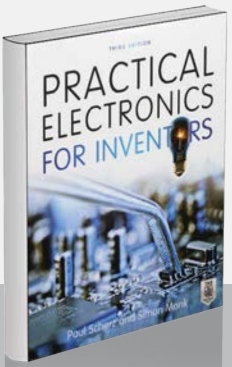
**Author:** Simon Monk
**Publisher:** Tab Electronics
**Price:** £9.99
**ISBN:** 978-0071807838
simonmonk.org/?page_id=63



A brilliant Python introduction which plays to the Pi's strengths. Buy this one now (but read the online errata), then pass on to a friend when you buy the updated edition in November.

### PRACTICAL ELECTRONICS FOR INVENTORS

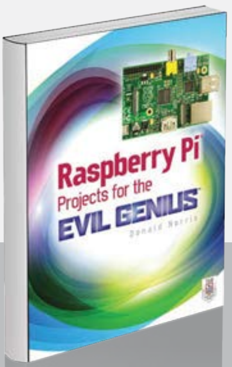
**Authors:** Paul Scherz & Simon Monk
**Publisher:** Tab Electronics
**Price:** £24.99
**ISBN:** 978-0071771337
bit.ly/1H7TaAS



How to use and understand electronic components, with updated chapters on sensors, microcontrollers, modular electronics, and the latest software tools. Worth the price for the theory chapter alone.

### RASPBERRY PI PROJECTS FOR THE EVIL GENIUS

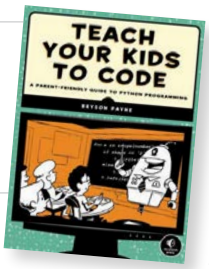
**Authors:** Donald Norris
**Publisher:** Tab Electronics
**Price:** £15.99
**ISBN:** 978-0071821582
bit.ly/1GBAHJx



Quirkily written beginner's introduction to building hardware projects with the Pi (based on the older B model, but little change needed on most projects for current models), from a Bluetooth-controlled robot to a weather station.

## TEACH YOUR KIDS TO CODE

**Author:** Bryson Payne
**Publisher:** No Starch
**Price:** £19.99
**ISBN:** 978-1593276140
nostarch.com/teachkids



Dr Payne promises "programming so easy a parent can do it!" Starting with turtle graphics, the reader is drawn in, and Python seems natural, easy, yet still a thing of wonder. Learners are encouraged to experiment, rather than overloaded with details of how and why – but where details are necessary, such as number types and operators, they are introduced.

As concepts are introduced, we feed them into turtle graphics – so conditionals lead to fractal spirals, and user input selects the shape drawn. Programming challenges at the end of each chapter – turning the High Card game into War, or adding sound effects to the Pong game – steer further learning.

Aimed at children aged nine and up, there's enough here for everyone – parents can take younger children through some of the projects, and teens shouldn't feel talked down to. Payne has the balance right beween giving enough in plentiful, fun projects to keep interest, and introducing programming concepts to build real understanding almost by osmosis. Python and Pygame installation are banished to an appendix, where you'll also find instructions on creating your own modules! And once more, delightful illustrations by Miran Lipovača lift an already excellent No Starch book to another level. Strongly recommended for learners of all ages.

**Score** ★★★★★

## JAVASCRIPT ROBOTICS

**Author:** Edited by Rick Waldron
**Publisher:** Maker Media
**Price:** £19.99
**ISBN:** 978-1457186950
oreil.ly/1Gv23Rv

The Raspberry Pi is a great little board, but it doesn't move around very much – unless you put it inside a robot, of course. The Johnny-Five robotic library offers the chance to easily program robots on various platforms with JavaScript – in this book's case, "robots that rove, swim, type, walk, dance, send alerts, make music" and more.
Of the projects, only the PiDuino5 Mobile Robot Platform demands a Raspberry Pi (combining the Arduino's extensive peripheral range with the Pi's high performance software) – in an excellent project by Jonathan Beri that includes useful tips for hardware newbies, such as connecting to the Pi without hooking your robot up to a network, as well as offering challenges to build into the project, such as computer vision.

The other chapters all feature different boards, and vastly different costs. Some have ready-made parts available and some involve a lot of great DIY – particularly the chapter on Delta Bots, the kind of industrial robot that built your car (unless you're driving a Morgan). This chapter also offers a lot of flexibility for dropping in another Pi and extending the interface. All feature clear colour pictures to help construction. Plenty of fun, plenty of learning, and a great introduction to practical robotics.

**Score** ★★★★☆

## MACHINE LEARNING WITH SPARK

**Author:** Nick Pentreath
**Publisher:** Packt
**Price:** £30.99
**ISBN:** 978-1783288519
**bit.ly/1GnnQdE**

With its four cores and 1GB of RAM, the Raspberry Pi 2 is just the thing for a small cheap cluster to run Apache Spark – the cluster computing framework that excels with machine learning algorithms. Enter Pentreath with an example-led book on machine learning with Apache Spark.

Clustering framework MapReduce has too high an overhead for the Pi 2; however, Spark's easy-to-understand API, and design optimised for keeping intermediate tasks and data in memory – yet still fitting in the Hadoop ecosystem – is a great platform for studying and developing machine learning algorithms. This work starts with setting up on Amazon EC2, but there's plenty of online help for getting a Spark cluster running on the Pi, then you're ready for the remaining chapters.

These chapters cover real-life use cases for machine learning which will take anyone with basic or no knowledge of the subject a long way, and help those with some machine learning experience gain a strong understanding of using Spark, from text mining to dimensionality reduction. Examples are nearly all in Scala and Python – the latter a welcome plus point for the book from the Pi user's perspective. Recommended for all aspiring data scientists.

**Score** ★★★★☆

## DATA SCIENCE FROM SCRATCH

**Author:** Joel Grus
**Publisher:** O'Reilly
**Price:** £26.50
**ISBN:** 978-1491901427
**oreil.ly/1awOcy7**

There are some great titles that steer you through frameworks and libraries to build data science projects, but Grus takes a different approach to educate the data curious: building tools and implementing algorithms from scratch, to illuminate particular points that will enable you to take better advantage of the frameworks when you finish the book.

After introducing data science, we start with a fast-paced intro or refresher in Python programming that does the job well, but highlights this book's one flaw – data science libraries still depend, for the most part, upon Python 2.7, and thus there's no Python 3 here. However, the Python 2 is very good and worth sticking with for the other early lessons in maths.

The chapters on statistics, probability, and linear algebra are an excellent refresher for those of us who are rusty, and form a strong basis for what comes next.

Having covered the maths and the programming – "the raw materials to do data science", as Grus puts it – the author turns to both the science and the art of working with data, showing how to explore what you've got before getting to modelling and machine learning. Grus looks into details where necessary, but trusts the reader to follow up references herself in other cases. In all cases, this book should make better data scientists.

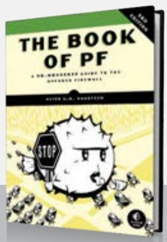
**Score** ★★★★☆

## ESSENTIAL READING: SECURITY

**Security is not a checkbox, but a continual process. These five recent titles should keep you thinking**

### The Book of PF

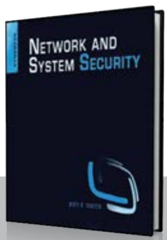
**Author:** Peter N.M. Hansteen
**Publisher:** No Starch
**Price:** £23.50
**ISBN:** 978-1593275891
**nostarch.com/pf3**



OpenBSD's stateful packet filter, pf, helps you build flexible and powerful firewalls – from traffic shaping to blocking policy.

### Network and System Security

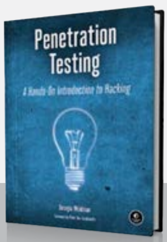
**Author:** John Vacca
**Publisher:** Syngress
**Price:** £37.99
**ISBN:** 978-0124166899
**johnvacca.com**



Useful overview for the challenges of organisations' complex networks – from LANs to RFID, via cloud security and intrusion detection.

### Penetration Testing

**Author:** Georgia Weidman
**Publisher:** No Starch
**Price:** £33.50
**ISBN:** 978-1593275648
**nostarch.com/pentesting**



Great introduction to finding vulnerabilities in your system – penetration testing made accessible, and well illustrated too.

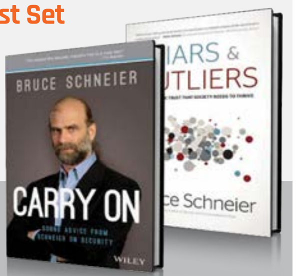
### Blockchain: Blueprint for a New Economy

**Author:** Melanie Swan
**Publisher:** O'Reilly
**Price:** £16.50
**ISBN:** 978-1491920497
**oreil.ly/1L2AFMq**

Blockchain is more than just bitcoin, and information security and digital liberties will become more dependent upon this technology.

### Bruce Schneier on Trust Set

**Author:** Bruce Schneier
**Publisher:** Wiley
**Price:** £26.99
**ISBN:** 978-1118906835
**oreil.ly/1JfnR7G**



Bargain pairing of Liars & Outliers (how trust works – and fails – and what we need to rethink) and Carry On (155 "thought-provoking" essays).

# WANT TO GET NOTICED?

**ADVERTISE IN THE MAGPI MAGAZINE**

REACH THE RIGHT AUDIENCE FAST

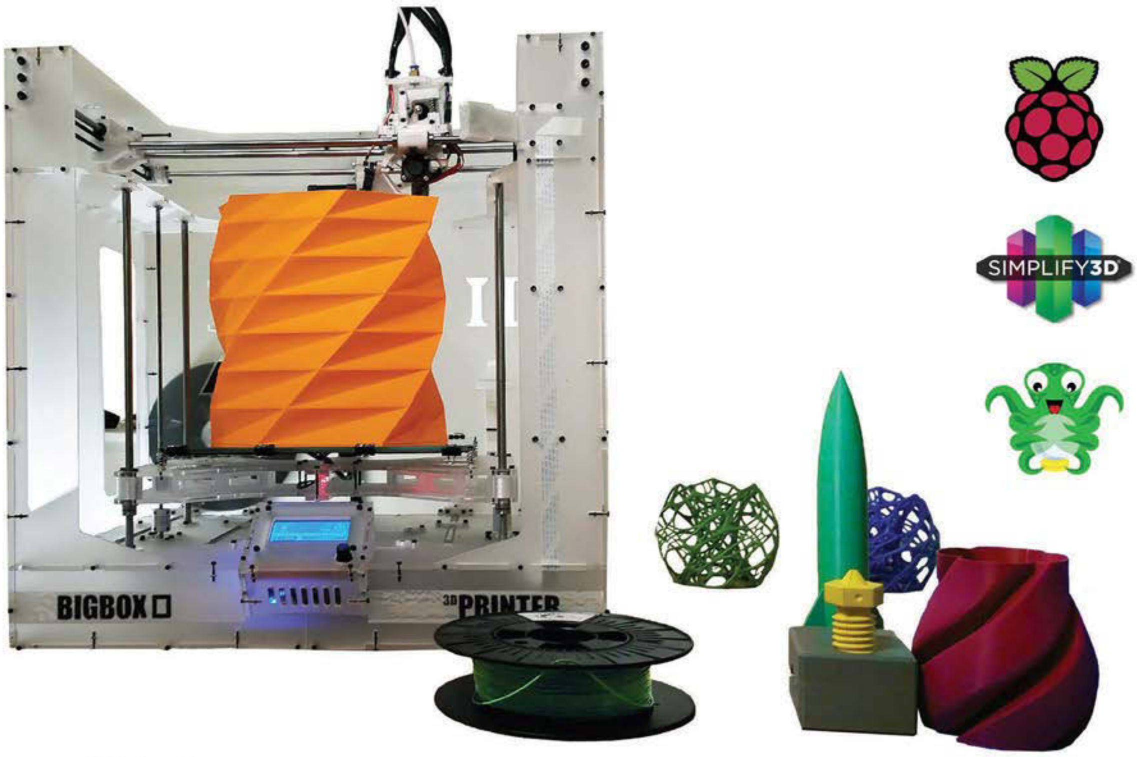
- World's #1 Pi magazine
- Block booking discounts
- Free to download & share
- Live links on iOS & Android

The MagPi is the most exciting mag in tech today, boasting one of the biggest & most engaged audiences in the industry.

The MagPi
70 PAGES OF ESSENTIAL RASPBERRY PI TIPS AND TUTORIALS
RASPBERRY PI DESKTOP SUPER-TEST
Find out which desktop distro deserves a place on your Pi
Plus MORE AMAZING RASPBERRY PI PROJECTS MADE BY YOU

FOR MORE INFO EMAIL **RUSSELL@RASPBERRYPI.ORG**

Print with Pi. Print with **BIGBOX.** Kickstarter ends August 16th.



BigBox-3D.com

# 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. I²C address selection allows you to add up to 32 analogue channels to your Raspberry Pi.

## IO Pi Plus

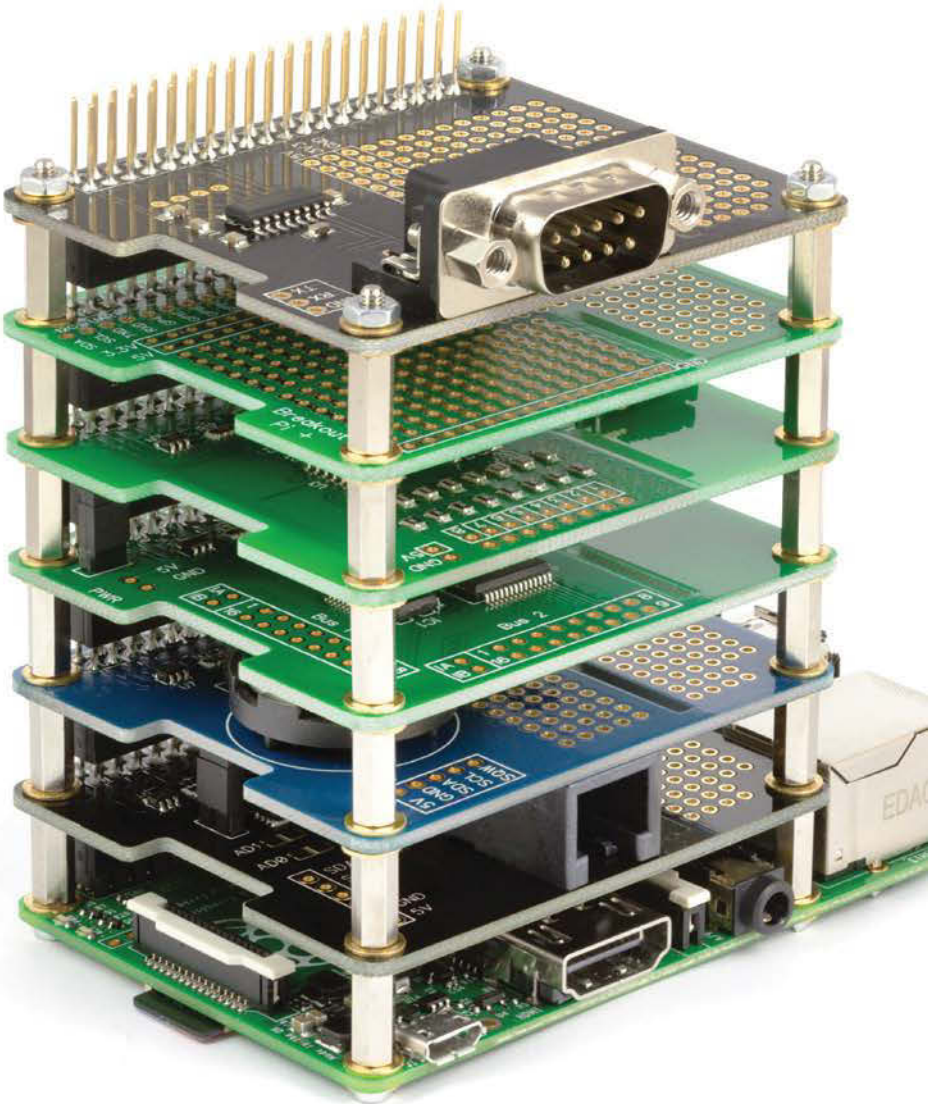
32 digital 5V inputs or outputs. I²C address selection allows you to stack up to 4 IO Pi Plus boards on your Raspberry Pi.

## RTC Pi Plus

Real-time clock with battery backup and 5V I²C level converter for adding external 5V I²C devices to your Raspberry Pi.

## 1 Wire Pi Plus

1-Wire® to I²C host interface with ESD protection diode and I²C 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

8 FAMILY HACK JAM
Hamilton Grammar School, Hamilton
7 COVENTRY MAKE JAM
Koco Community Building, Coventry
4 TORBAY TECH JAM
Paignton Library and Information Centre, Paignton
2 RHÔNE VALLEY JAM #4
13 Avenue Jean Jaurès, 84350 Courthézon

# RASPBERRY JAM EVENT CALENDAR

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

## PUT YOUR EVENT ON THE MAP

Want to add your get-together? List it here:
**raspberrypi.org/jam/add**

### 1 EXETER RASPBERRY JAM

**When:** Saturday 1 August
**Where:** Exeter Library, Exeter, Devon, UK
**bit.ly/1edwEsH**
Exeter Jam takes place between 10am and 12pm and – as always – features plenty of projects, help, and advice!

### 2 RHÔNE VALLEY JAM #4

**When:** Sunday 2 August
**Where:** 13 Avenue Jean Jaurès, 84350 Courthézon, France
**bit.ly/1dfy71k**
Alan McCullagh and Amaury Doucet are taking another slice of Pi in the Rhône valley.

### 3 PRESTON RASPBERRY JAM

**When:** Monday 3 August
**Where:** Media Innovation Studio, Media Factory Building, Preston, UK
**bit.ly/1DrBaun**
Join in for a day of fun lightning talks, demonstrations, and hands-on time with the Pi.

### 4 TORBAY TECH JAM

**When:** Saturday 8 August
**Where:** Paignton Library and Information Centre, Paignton, UK
**torbaytechjam.org.uk**
Come along for a fun and informal day of hacking with your own kit, or use the equipment provided.

### 5 HUDDERSFIELD RASPBERRY JAM

**When:** Saturday 8 August
**Where:** Sound and Vision Room, Huddersfield, UK
**huddersfieldraspberryjam.co.uk**
Join in the fun at Huddersfield Library. No prior experience is necessary to come along!

### 6 IPSWICH RASPBERRY JAM

**When:** Saturday 8 August
**Where:** Enterprise + Innovation Hub at Ipswich County Library, Ipswich, UK
**ipswichraspberryjam.co.uk**
Show your support for the first Ipswich Jam. Come along to the morning or afternoon sessions.

3 PRESTON RASPBERRY JAM
Media Factory Building, Preston
5 HUDDERSFIELD RASPBERRY JAM
Sound and Vision Room, Huddersfield
6 IPSWICH RASPBERRY JAM
Ipswich County Library, Ipswich
1 EXETER RASPBERRY JAM
Exeter Library, Exeter

7

## COVENTRY MAKE JAM

**When:** Saturday 22 August
**Where:** Coventry Makerspace, Koco Community Building, Coventry, UK
**bit.ly/1FECioE**
Come along and meet fellow Pi users, look at some cool projects, and build your own!

8

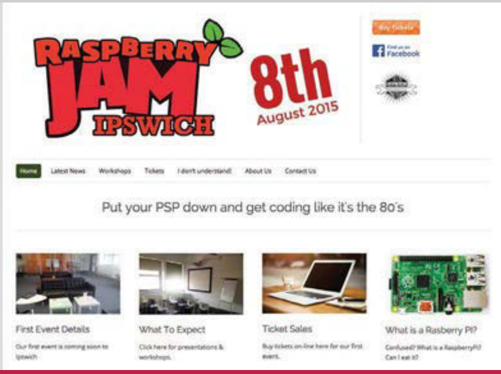
## FAMILY HACK JAM

**When:** Friday 28 August
**Where:** Hamilton Grammar School, Hamilton, South Lanarkshire, UK
**bit.ly/1edySIA**
This Family Hack Jam event brings families together to discover the fun, excitement and power of computer science.

## DON'T MISS: IPSWICH RASPBERRY JAM

**When:** Saturday 8 August
**Where:** Enterprise + Innovation Hub at Ipswich County Library, Ipswich, UK

Andy Proctor, of issue 31's iData Truck fame, invites you to put your PSP down and get coding like it's the 1980s at the inaugural Ipswich Raspberry Jam. Due to a limited capacity of 50 people at the Enterprise + Innovation Hub in Ipswich County Library, Andy and friends will be running two sessions – one from 10am to 1pm, and another at 1pm until 4pm. Find out more via **ipswichraspberryjam.co.uk**



Put your PSP down and get coding like it's the 80's

First Event Details
Our first event is coming soon to Ipswich

What To Expect
Click here for presentations & workshops.

Ticket Sales
Buy tickets on-line here for our first event.

What is a Rasberry Pi?
Confused? What is a RaspberryPi? Can I eat it?

# TAKE US ANYWHERE

THE OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
Win! £200 of robotics equipment
AMAZING PI PROJECTS
10 imaginative creations that innovate and inspire
WINDOWS 10 ON RASPBERRY PI

**SAVE 45%** with a Newsstand subscription (limited time offer)

**The MagPi Magazine**

Available now for smartphones & tablets

Available on the App Store

Get it on Google play

*From just*

**£2.29** rolling subscription or **£19.99** full year subscription

**Why buy digitally?**

- Get all copies back to issue 31
- Instant downloads every month
- Fast rendering performance
- Live links & interactivity

In association with:

element14

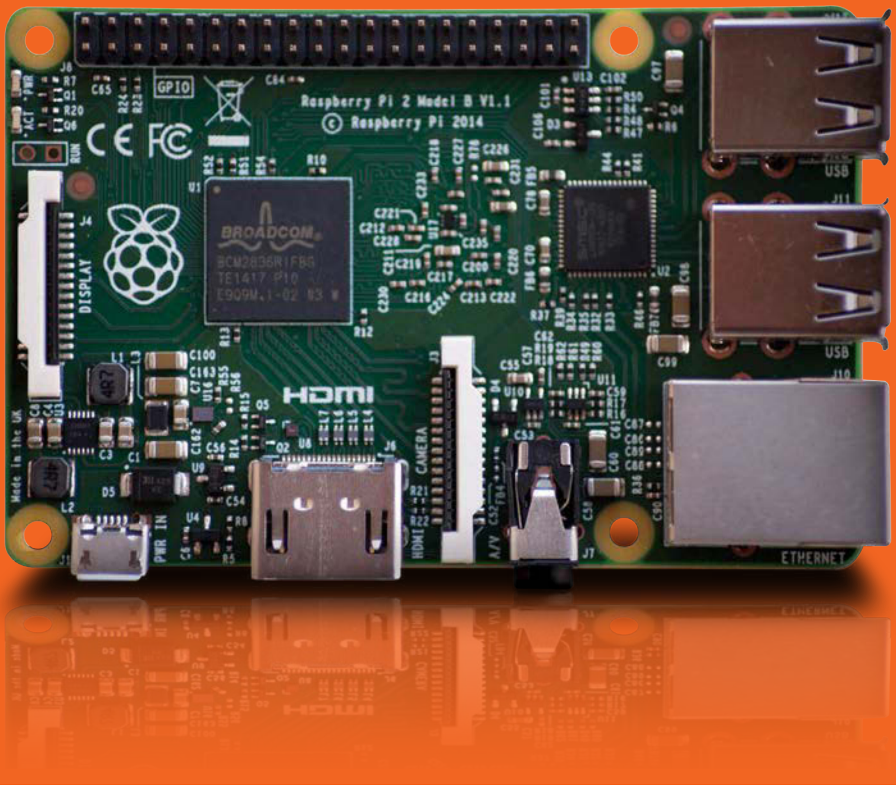
element14.com/raspberrypiprojects

# 10 Raspberry Pi 2s MUST BE WON!

**GET INVOLVED** in the exciting world of Raspberry Pi with engineers, innovators and experts from around the world on the element14 **COMMUNITY!**

# WHERE DO YOU BUY YOUR **RASPBERRY PI EQUIPMENT?**

Tell us by 26 August for your chance to win!



## How to enter:

All you have to do is email **competition@raspberrypi.org** letting us know where you like to buy your Raspberry Pi equipment and components. It really is that easy!

## Terms & Conditions

Competition closes 26 August 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.

# IS THERE A PLACE FOR THE RASPBERRY PI IN PRIMARY EDUCATION?

**Graham Hastings** of St John's College School in Cambridge takes on our first ever big question...

This is a question I first asked about three years ago, and revisited while a member of the working group tasked with writing the new Computing programmes of study. I have been teaching computing to primary school children for more than 30 years, starting with the BBC Micro in the 1980s. I was an early adopter of Scratch and Python as suitable languages for teaching programming to the under-12s. On discovering that the Pi would run both of these, my interest was further whetted; however, I was already running them on my school's Windows PCs and could see little reason to change. It was only when I began to explore the GPIO pins on the Pi and recalled some of the fantastic control projects that I used to interface with the BBC Micro all those years ago that I saw a possible niche for the Raspberry Pi in my school.

It has been well documented by Ofsted that the teaching of control technology has been deficient in many UK schools in recent years. This was not a fault of the old ICT programmes of study, which included attainment targets for control technology. There is in fact a cocktail of factors which mitigate against the teaching of physical computing. I would identify the following:

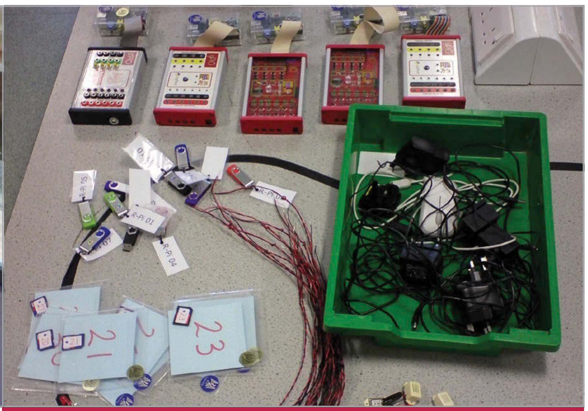
- A perception that the subject is conceptually difficult.
- Logistical issues involved in setting up and maintaining the hardware to teach computer control.
- The cost of specialist equipment – hardware and software.
- The cyclical need to upgrade and the additional costs this brings in terms of new equipment, time, and teacher training.
- More recently, particularly in primary schools, we are seeing tablets, with no means of connecting control interfaces, replacing Windows PCs.

A Prezi presentation, titled Miss Smith's Tail, summarises the situation in a typical primary school (**bit.ly/1Jb2xPL**).

**Below** Here's the gPiO kit, ready to go

## Affordable and sustainable

So, the niche I imagined for the Raspberry Pi in primary schools was as an affordable, sustainable, and fully programmable workstation for teaching physical computing. My early experiments with breadboard, jumper wires and batteries were not promising. All bar the most gifted young children lack the manual dexterity, resilience and conceptual understanding of electronics to work directly with the GPIO pins on the Pi. In fact, I lost a couple of Pis to short circuits during these experiments. It was back to the drawing board. Children can build control systems with little difficulty using interfaces designed for primary schools. Large 4mm plugs and clear colour coding make it easy for children to connect their components correctly and there is no danger that they will accidentally destroy the computer. A solution finally arrived in the form of the gPiO (**gpio.co.uk**), a child-friendly interface designed for the Pi. I purchased a class set of gPiOs and recycled keyboards and mice from old Windows PCs. Fortunately, I already had some digital screens which now double up as monitors for Windows PCs and Raspberry Pi control workstations. The cost, including cables, was less than £150 per workstation: little more than the price of just a control


**Above** Pupils learn about physical computing using a Pi and gPiO box

interface for a PC. It only takes the children three minutes to switch from PC workstation to Pi control workstation.

I envision an important role for the Raspberry Pi in primary schools. It is timely that the Pi and gPiO combination is an option at a time when the Windows PCs that have been the mainstay for teaching physical computing for the past 20 years are rapidly disappearing from schools. It also enables an escape from the tyranny of the upgrade cycle. On more than one occasion I have been forced to replace hardware and upgrade software, despite the fact that they work perfectly, for no better reason than that the for-profit companies which supplied them will no longer support them.

## From Scratch

There are sound pedagogical reasons for using the Raspberry Pi for physical computing. Scratch is a language the majority of primary schools are already teaching. To be able to use it for control simplifies the learning for the children and training for the teachers. The fact that Python can also be used creates a progression route to extend the most able children, and control is an excellent context for the introduction of this textual programming language. Using the Pi offers another significant advantage over the traditional way of teaching control through the use of languages such as Flowol. Many of the traditional control languages make it very difficult for children to work on projects that do little more than respond to sensors by switching a number of outputs. It is not possible within these programs to properly consider human factors and the needs of the user, security features, access to data stored on other files, or to write data to memory for storage. By using Scratch or Python via the Pi for control programs, it is possible for children to create a sophisticated user interface with complex graphics. Children can be encouraged to think about the 'whole system' and not just the simple input-process-output routines.

The Raspberry Pi was conceived with a mission to recreate the conditions in the 1980s so that young people might once again discover a fascination for computer programming. But times are very different: children now have the internet, Xbox, smartphones, and social networking. Primary schools have the advantage that they can engage and enthuse children before they develop fixed ideas about what a computer is and what it should be used for. It was when the BBC Micro appeared in primary schools that the UK computing revolution really got underway. If the adoption of the Pi for control can be encouraged, this will go a long way towards ensuring that the Raspberry Pi Foundation succeeds in this mission.

## Camera Trap Project case study

### Task

Create a camera trap as part of a security system or for wildlife surveillance. The trap must be triggered by movement and the image taken must be stored on a memory stick, for ease of downloading. There must be an indicator light to show when the trap is armed. If the trap is used as part of a security system, it must also sound an alarm. Here's a video of an outline of the project: **youtu.be/MW7Yhjr_5-U**.

### Aim

To provide a 'real life' context for teaching physical computing using Python in Year 6.

### Prior learning

The children know how to connect up a Pi and they have completed some lessons on Python. They can work confidently with variables, while loops, functions, importing modules, and using GPIO for simple control work via the gPiO interface. The code for Pi Camera Module operation is given to the children.

### Class management

16 children working in pairs. The gPiO interfaces are connected to the Raspberry Pis in advance of the lesson. The children use a standard SD card prepared for them with the resources they will need. The components are laid out in trays. Each pair is issued with a USB memory stick on which to save their work.

### Extension

The children are asked to write a security feature that will allow the user to set a code with a four-digit PIN so that the alarm can only be reset if the correct PIN is entered. A GUI keypad should be created to make it easier for the user to enter the code.

70 PAGES OF EXPERT ADVICE IN THE OFFICIAL RASPBERRY PI MAGAZINE

The MagPi

The official Raspberry Pi magazine Issue 34 June 2015 raspberrypi.org/magpi

Win! £300 of PaPiRus ePaper HATs FROM PI-SUPPLY.COM

BUILD A MULTIPLE-CHOICE QUIZ USING SCRATCH
The local pub quiz will never be the same

USE YOUR RASPBERRY PI REMOTELY WITH VNC
Access your Raspberry Pi desktop on your tablet or smartphone

PUT YOUR CODE IN SPACE!
The latest on the Astro Pi project. There's still time to enter

WINDOWS 10 ON RASPBERRY PI
How Microsoft plans to help you build Internet of Things devices

MAKE A PLATFORM GAME WITH PYTHON
Put all your game-making skills to work on your first full game

Also inside:
- HACK A CLOCK WITH YOUR RASPBERRY PI
- GET CREATIVE WITH CONDUCTIVE PAINT
- MORE AMAZING PROJECTS REVEALED
- 4TRONIX AGOBO ROBOT REVIEWED

SCROLL TEXT OVER THE UNICORN HAT
Put your text up in lights with our expert step-by-step guide

## Why Windows 10?

**While I very much enjoyed issue 34 of *The MagPi*, I have no idea why anyone would want to use an open-source device like the Raspberry Pi to run Windows 10. It flies in the face of everything the Raspberry Pi stands for.**
**Greg Pearce**

Linux and open-source software are at the core of the Raspberry Pi, but it's important to realise that the Raspberry Pi itself is essentially just a computer and, as such, developers and users are free to install any software they like. There's the argument of choice (why not run Windows on it if you can?), but there's also the argument of reach. There are millions of Microsoft developers out there that would like to hack, make and develop with Raspberry Pi, but don't have the time and resources to learn entirely new environments and languages. Being able to develop Windows 10 apps using the Raspberry Pi 2 simply opens the doors to more amazing creations and even more people learning to code with affordable, accessible hardware, and that's one of the core goals of the Raspberry Pi Foundation.

## Back issues in print?

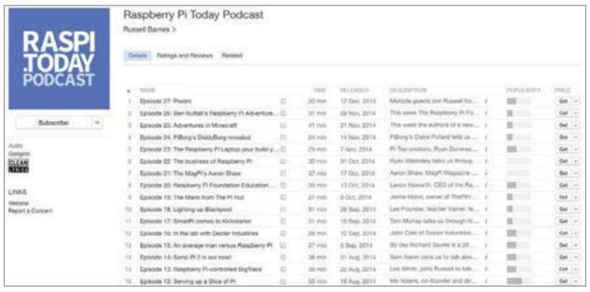
**I was really pleased to learn that *The MagPi* was going into print with issue 36 and I've already bought a one-year subscription online. Since I'd really like to complete my collection (having already got the first three volumes of the original *MagPi* fanzine), I wonder if you plan to produce print copies of your first five issues as the Official Raspberry Pi magazine?**
**Janet Bates**

You're not the only person to ask this, Janet, and we've been carefully weighing up our ability to produce a limited run of our first five issues. Sadly, it's simply not possible to produce them at a reasonable price. That said, we do have a plan up our sleeves to come up with what we think would be a brilliant compromise, but you'll need to sit tight for a couple of months while we get everything in place. If any other readers would like to subscribe to the print edition, you can learn more on pages 24 and 25. If you'd rather browse our back catalogue digitally, you can have them delivered to your Apple or Android device – see page 90 for more details.

## More podcasts?

**I just wanted to get in touch to say how much I used to love your weekly podcast recordings for Raspberry Pi Today and would love to see them come back. It's been really missed since you stopped at Christmas 2014 and I would love to know if you plan to do something similar for *The MagPi* magazine?**
**Dillon Hooper**

As some of you may know, before moving across to run *The MagPi*, managing editor Russell Barnes was a blogger and weekly podcaster with **RasPi.Today**. Over the last six months, Russell has had hundreds of emails and requests from regular and new listeners alike, asking if there might ever be new recordings of the show. Due to work commitments, though, Russell tells us that particular podcast won't be returning. That said, if there's sufficient demand for a podcast from *The MagPi* team, we'll definitely give it a go!



Raspberry Pi Today Podcast
Russell Barnes >

| | NAME | TIME | RELEASED | DESCRIPTION | POPULARITY | PRICE |
|---|---|---|---|---|---|---|
| 1 | Episode 27: [illegible] | 30 min | 17 Dec, 2014 | Multiple guests join Russell fro... | | Get |
| 2 | Episode 26: Ben Nuttall's Raspberry Pi Adventure... | 31 min | 28 Nov, 2014 | This week The Raspberry Pi Fo... | | Get |
| 3 | Episode 25: Adventures in Minecraft | 41 min | 21 Nov, 2014 | This week the authors of a new... | | Get |
| 4 | Episode 24: PiBorg's DiddyBorg revealed | 24 min | 14 Nov, 2014 | PiBorg's Claire Pollard tells us ... | | Get |
| 5 | Episode 23: The Raspberry Pi Laptop your build y... | 29 min | 7 Nov, 2014 | Pi-Top creators, Ryan Dunwoo... | | Get |
| 6 | Episode 22: The business of Raspberry Pi | 30 min | 31 Oct, 2014 | Ryan Walmsley talks us throug... | | Get |
| 7 | Episode 21: The MagPi's Aaron Shaw | 37 min | 17 Oct, 2014 | Aaron Shaw, MagPi Magazine ... | | Get |
| 8 | Episode 20: Raspberry Pi Foundation Education... | 39 min | 13 Oct, 2014 | Lance Howarth, CEO of the Ra... | | Get |
| 9 | Episode 19: The Mann from The Pi Hut | 27 min | 8 Oct, 2014 | Jamie Mann, owner of ThePiH... | | Get |
| 10 | Episode 18: Lighting up Blackpool | 51 min | 26 Sep, 2014 | Les Pounder, teacher trainer, te... | | Get |
| 11 | Episode 17: SmartPi comes to Kickstarter | 31 min | 19 Sep, 2014 | Tom Murray talks us through hi... | | Get |
| 12 | Episode 16: In the lab with Dexter Industries | 28 min | 12 Sep, 2014 | John Cole of Dexter Industries... | | Get |
| 13 | Episode 15: An average man versus Raspberry Pi | 27 min | 5 Sep, 2014 | By day Richard Saville is a 28 ... | | Get |
| 14 | Episode 14: Sonic Pi 2 is out now! | 38 min | 31 Aug, 2014 | Sam Aaron joins us to talk abo... | | Get |
| 15 | Episode 13: Raspberry Pi-controlled BigTrak | 36 min | 22 Aug, 2014 | Lee White, joins Russell to talk... | | Get |
| 16 | Episode 12: Serving up a Slice of Pi | 32 min | 18 Aug, 2014 | Mo Volans, co-founder and dir... | | Get |

Audio Gadgets
CLEAN LYRICS
LINKS
Website
Report a Concern

# FROM THE FORUM: BLAST FROM THE PAST

The Raspberry Pi Forum is a hotbed of conversation and problem solving for the community – join in via **raspberrypi.org/forums**

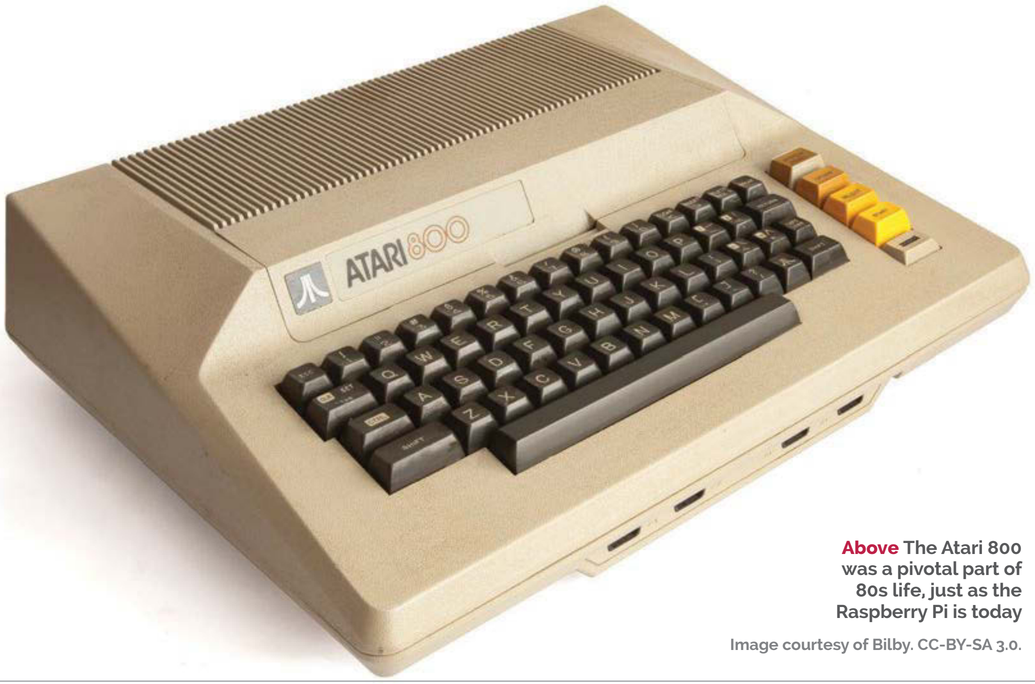
I have to say how much I enjoyed reading your magazine. I had a realisation and a sort of flashback the other night. Back when I was a teenager in the early 1980s, I got started with an Atari 800. We didn't have internet back then but we did have magazines. I had to order my Atari and wait weeks for it to come in. I had read every magazine I could find at the time from cover to cover, just waiting for it to arrive. When it arrived, the magic began and I found myself typing in program listings for games and utilities. It was hands-on learning – and no cut and paste either!

For years I have lamented how my children will never experience this feeling, especially when something you typed in comes to life. Well, imagine my surprise when I started reading *MagPi*. There are program listings and explanations of projects, just like when I was a teenager. Like the topic states, this was a real blast from the past. I picked up my new Pi this weekend and have been in love. I have Linux laptops and desktops, and have been programming since back when I picked up my Atari, and this seems to have captured the same magic as my old Atari. It really hit home when I realised this as I was sitting at my TV with a computer hooked up to it. It hasn't been since I was a teenager that I have sat in front of a TV to type programs into a computer. Now, if I could just get a cassette tape drive to store the programs...

Thanks to everyone at *MagPi* and Raspberry Pi. I am loving this.

**Fig Newton**

**We're blushing. The Raspberry Pi is all about a renaissance of those early years of bedroom coding. We've often longed for a return to the days of code listings, pokes (cheats in today's parlance) and innovative projects, so we're really pleased we've managed to recapture happy memories from the 80s for you. Thanks for reading.**



**Above** The Atari 800 was a pivotal part of 80s life, just as the Raspberry Pi is today

Image courtesy of Bilby. CC-BY-SA 3.0.

## COMPETITION WINNERS

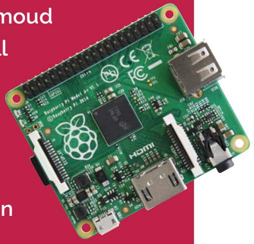
Congratulations to all the winners of our first four competitions! If you'd like to win a Raspberry Pi 2 courtesy of **element14.com**, turn to page 91 now to find out how...

### Issue 31
### 5 Raspberry Pi 2s

Aris Tsitras
Guido Erlinger
Richard Mitchell
Peter Davidse
Andy Jackson

### Issue 32
### 10 Pi Model A+s

Max Filley
Mamdouh Mahmoud
Andrew Banwell
Istvan Nagy
Grace Turner
Jim Toews
David Kieffer
David Harrington
Jacob Payne
Seth Bearden

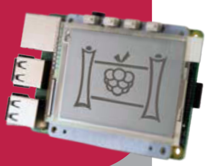


### Issue 33
### Diddyborg Robot

Joshua Lowe
Three runners-up:
Darcy Pach
Andrew Goodney
Dave Rogers

### Issue 34
### 5 PaPiRus HAT Packs

Ewald Beekman
Mark Brown
Patrick Goodleaf
Rhys Thomas
Daniel Schlapa



## WRITE TO US

**Have you got something you'd like to say?**

Get in touch via **magpi@raspberrypi.org** or on The MagPi section of the forum at **raspberrypi.org/forums**

**MATT RICHARDSON**

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

# THE INTERSECTION OF ART AND TECHNOLOGY

After a trip to Disneyland, **Matt Richardson** considers creative uses of Raspberry Pi

I'm a very lucky guy. As part of my job as product evangelist at Raspberry Pi, I recently had the opportunity to visit Disneyland and get a tour of some of their backstage areas. I've been to Disney parks as a kid, but even seeing it as an adult, it feels like a magical place (and Space Mountain is just as much fun as I remember). I especially enjoyed this visit; from the moment I entered the park, I observed technology being used to make each guest's visit feel magical.

The people behind that magic, Disney Imagineers, do amazingly creative work with some of the newest technology. Even though I was learning about how they use technology creatively within the park, it didn't ruin the fun. It's a lot like figuring out or learning how an illusionist plies his trade. Even if you do figure it out, it usually doesn't make the illusion any less enjoyable. In fact, sometimes knowing how an effect is achieved makes you appreciate it even more.

My visit to Disneyland got me thinking about how Raspberry Pi can be used creatively. Often people think that the Raspberry Pi is all about learning to program, but it's important to understand that programming is simply a means to an end. That end could be a creative endeavour: for example, to create visual art, to compose a piece of music, to tell a story, to make a game, to control the effects in a theatrical performance, or to make an interactive experience like Pirates of the Caribbean.

While I certainly appreciate all the strictly practical and educational uses of Raspberry Pi, there's something especially delightful when a Raspberry Pi is used creatively. For instance, interactive developer Michael Newman used Raspberry Pi to create Thirty-five Pixels (**bit.ly/1O9bicZ**), an installation for the lobby of UCLA Extension's Westwood Center in Los Angeles. The piece looks like an array of 35 circular displays, even though it's actually a single screen covered by a masking with 35 holes in it. The viewer can interact with each of the "pixels" over the web, on their phone, to set the colour of the displays or post photos to them individually. Thirty-five Pixels is powered by a Raspberry Pi Model B+ and uses the Camera Module for motion detection in order to respond to passersby.

Works like Thirty-five Pixels require both technical mastery and also a strong creative vision. For Michael, not only does he have the know-how to program the Raspberry Pi and fabricate the installation, he also has the visual design sense to make it look beautiful. Michael's talents in the realms of technology and art make a unique combination.

I think you'll agree that there's something special that exists at the intersection between technology and art. At Raspberry Pi, we recognise this and have inducted a group of students into a year-long Creative Technologists programme run by Rachel Rayns and Ben Nuttall. The students get access to mentoring, industry networking, and technical support. The program will culminate in a gallery show of their work next year. I've met most of these students and I'm excited to see what they create.

I encourage all of you to try your hand at creative technology. Whether you're a technical person that wants to be creative, or a creative person that needs technical chops, Raspberry Pi makes a great platform for experimenting with creative work. It's quite possible that many Imagineers and artists of tomorrow are getting their feet wet with Raspberry Pi today.

Raspberry Pi SWAG STORE

swag.raspberrypi.org

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

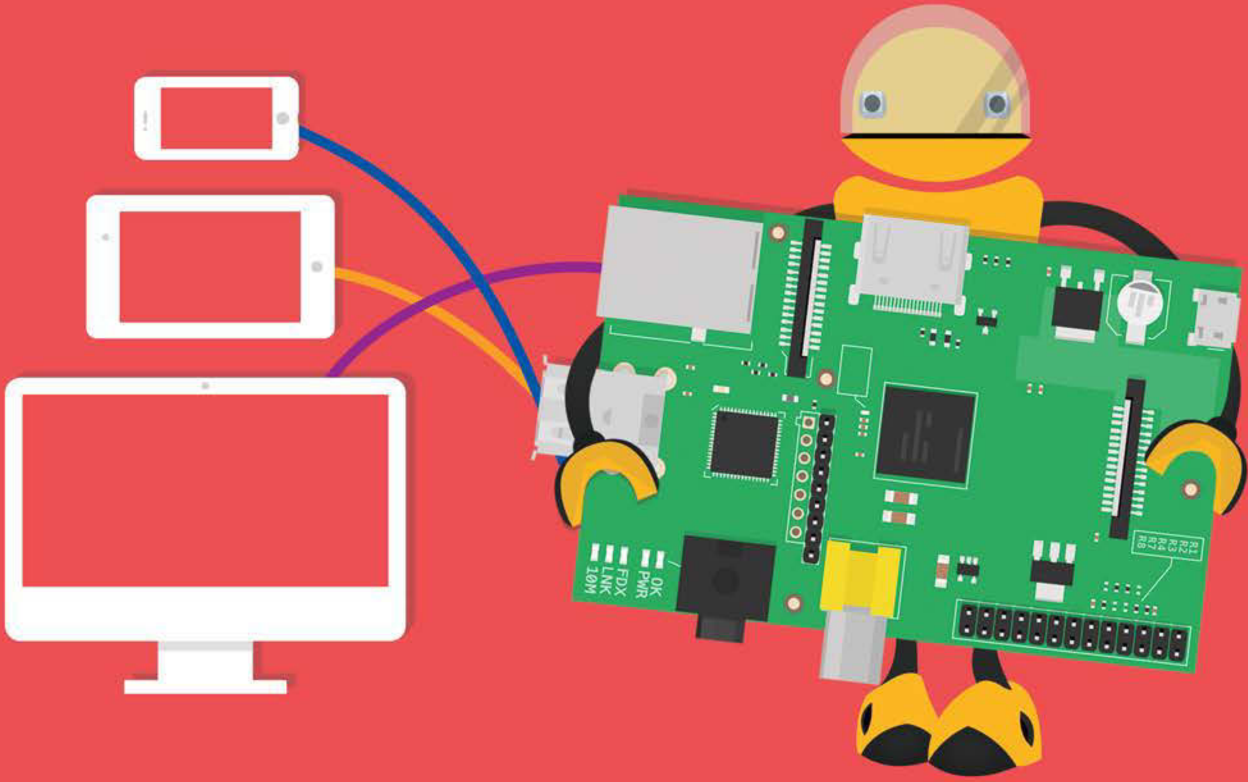
Just drag & drop blocks to create your applications, using Visual Programming

www.wyliodrin.com

python C++ node Perl Java ARDUINO php

# VNC® now available for RASPBERRY PI!

RealVNC has released VNC for Raspberry Pi, which means you can now connect to your Pi from any Windows, Mac or Linux computer, or iPhone, iPad or Android device!



Check out our video tutorial to see how you can set up VNC Server on your Raspberry Pi.

Watch the video here: **https://www.youtube.com/user/RealVNCLtd**

REAL VNC

Download VNC: www.realvnc.com/download/
Getting connected: www.realvnc.com/products/vnc/raspberrypi/
For more information contact sales@realvnc.com

Made in the UK

# IQaudIO

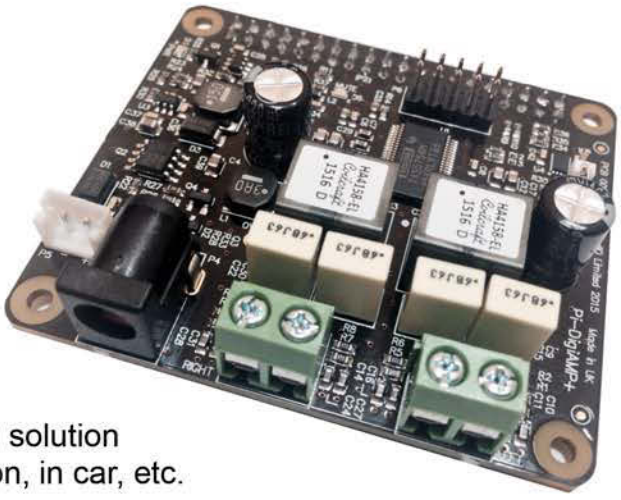
Audiophile accessories for the Raspberry Pi

## Pi-DAC+

- Raspberry Pi HAT, no soldering required
- Full-HD Audio (up to 24bit/192MHz)
- Texas Instruments PCM5122
- Industry standard Phono (RCA) sockets
- Headphone Amplifier / 3.5mm socket
- Out-of-the-box Raspbian support
- Integrated hardware volume control
- Access to Raspberry Pi GPIO
- Connect direct to your own HiFI's line-in/aux

## Pi-DigiAMP+

- Raspberry Pi HAT, no soldering required
- Up to 2x30w of crystal clear stereo amplification
- Full-HD Audio (up to 24bit/192MHz)
- Out-of-the-box Raspbian support
- Integrated hardware volume control
- Provides power to the Raspberry Pi
- Software controlled AMP mute on GPIO22
- I/O available (i2c, 3v, 5v, 0v, GPIO22/23/24/25)
- Just add speakers for a complete HiFi streaming solution
- Designed for custom install, retro radio restoration, in car, etc.



## Buy direct

PIMORONI
tindie
Franken schaum
MODMYPI
The PiHut
LIMETRACE

## Supported applications

rune audio
moOde audio player
PiMusicBox
piCorePlayer SQUEEZE AUDIO DEVICE
Archphile
openelec
VOLUMIO
Mike Brady's Shairport-Sync
Pete Manchester's MediaPlayer

AND MANY MORE....

Email: info@iqaudio.com
Twitter: @IQ_audio

WWW.IQAUDIO.COM

IQaudio Limited,
14 Ulysses Road
Swindon SN25 2JR
Company No.: 9461908