THE OFFICIAL RASPBERRY PI MAGAZINE

# The MagPi

The official Raspberry Pi magazine

Issue 35 July 2015

raspberrypi.org/magpi

**Win! £200 of robotics equipment** FROM DAWN ROBOTICS

## INSIDE VIDEOCORE

The Raspberry Pi engineering team reveal all

## MEET THE SPACE MAN

We speak to David Akerman about sending Pis to the edge of space

# AMAZING PI PROJECTS

10 imaginative creations that innovate and inspire

## BUILD AN INTERNET RADIO

It's easier than you might think

## MOVE SERVOS WITH A FLICK OF THE WRIST

How the UltraBorg can put you in control of your projects

## Also inside:

- THE ARCADE COCKTAIL CABINET
- MAKE A SOUNDBOARD WITH PYGAME
- MASTER THE COMMAND LINE (PART 5)
- BUILD A PI-POWERED SPRINTING GAME

## HANDS-ON WITH ASTRO PI

How to take scientific measurements with this amazing Raspberry Pi HAT

**Plus** PYGAME ZERO: MAKING GAMES WITH RASPBERRY PI HAS NEVER BEEN EASIER

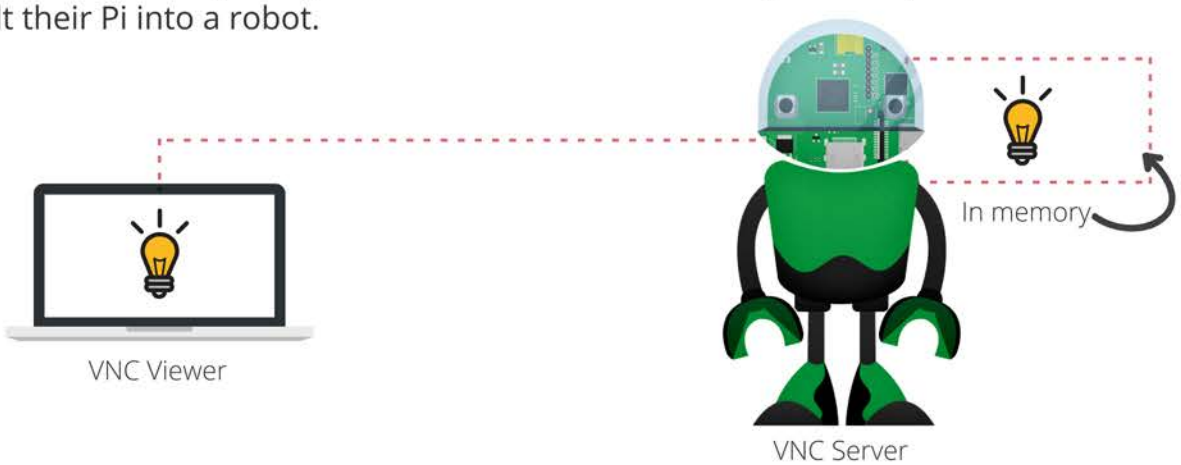
# VNC® now available for RASPBERRY PI!

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!

## VNC can be used on your Pi in two ways:

① If the Raspberry Pi is connected to a monitor or TV and is running a graphical desktop, you can see exactly what a user sitting in front of the Pi would see; perfect for generic remote access or sharing a screen with a friend!


② If your Pi is headless or not running a graphical desktop, then graphical remote access can still be achieved by using VNC to create a virtual desktop instead; ideal for users who have built their Pi into a robot.




Download VNC: www.realvnc.com/download/
Getting connected: www.realvnc.com/products/vnc/raspberrypi/

REAL VNC For more information contact sales@realvnc.com

in YouTube

# WELCOME TO ISSUE 35!

Last month we brought you the exciting news that *The MagPi* is becoming a fully fledged print magazine from issue 36. UK readers will be able to walk into their local WHSmith and pick up issue 36 on 30 July, while US readers will be able to buy it from Barnes & Noble or (and this is hot off the press) their local Micro Center a few short weeks later. If you're not in the UK or USA, or even if you don't relish the thought of leaving the house when there's perfectly good WiFi reception on your couch, you'll also be able to order a copy online and have it delivered to your door.

If that wasn't exciting enough, a quick jaunt over to pages 28–29 will reveal the next step in our bid for total world domination: the ability to subscribe to the first print issues TODAY.

No matter where you are in the world, we'll deliver a piping-hot edition of Official Raspberry Pi goodness to your door before it even hits store shelves. There are lots of subscription options available (including six issues, a full year, and Direct Debit for UK residents) and you can save up to 25% on the usual cover price. It's first come, first served, so call our subscriptions hotline, visit the subscriptions webpage, or put pen to paper today. Get all the details from pages 28–29.

Enjoy the issue!

**Russell Barnes**


## THIS MONTH:



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

The MagPi

Available on the App Store

Get it on Google play

**EDITORIAL**
Managing Editor: **Russell Barnes**
**russell@raspberrypi.org +44 (0)7904 766523**
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**

Contributors: **Tim Anderson, Arron Churchill, Mike Cook, David Crookes, Lucy Hattersley, Richard Hayler, Mike Horne, Phil King, Simon Long, Sean McManus, Simon Monk, Les Pounder, Matt Richardson, Tim Richardson, Richard Smedley, Sean M Tracey** & **Robin Withers.**


# Contents

Issue 35 July 2015 raspberrypi.org/magpi

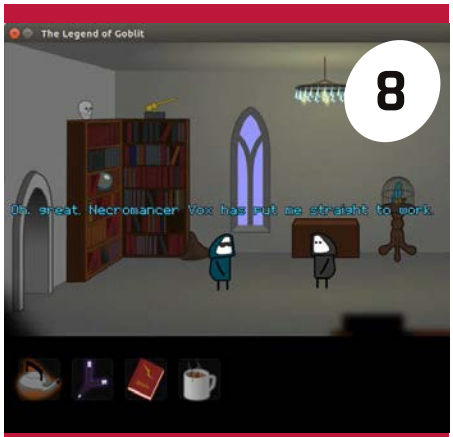
## TUTORIALS



## COVER FEATURE


### YOUR AMAZING RASPBERRY PI PROJECTS

We love to hear what you do with the Raspberry Pi. Here are some of our favourites

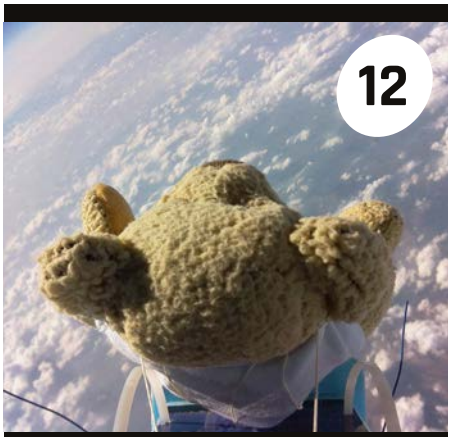



### PYGAME ZERO: GAMES MADE EASY

We speak to the creator of a new Pygame wrapper designed to make coding games with Python child's play

### KICKSTARTING THE WILDLIFE CAM KIT

Naturebytes tells us more about the Raspberry Pi camera kit that's just hit Kickstarter


### SPACED OUT WITH DAVE AKERMAN

We catch up with high-altitude balloonist Dave Akerman, to learn about his many jaunts to the edge of space


## PINET: A CLASS NETWORK

Join us as we speak to 18-year-old Andrew Mulholland about PiNet – an ingenious creation that can make a classroom full of Pis easy to manage

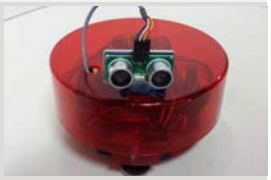
## YOUR PROJECTS

### LifeBox 22

We learn more about this fabulous creation of artificial intelligence using the Raspberry Pi

### PiPlateBot 24

In a fabulous display of tinkering, Robert Doerr has built an entire Pi robot out of his Raspberry Pi case



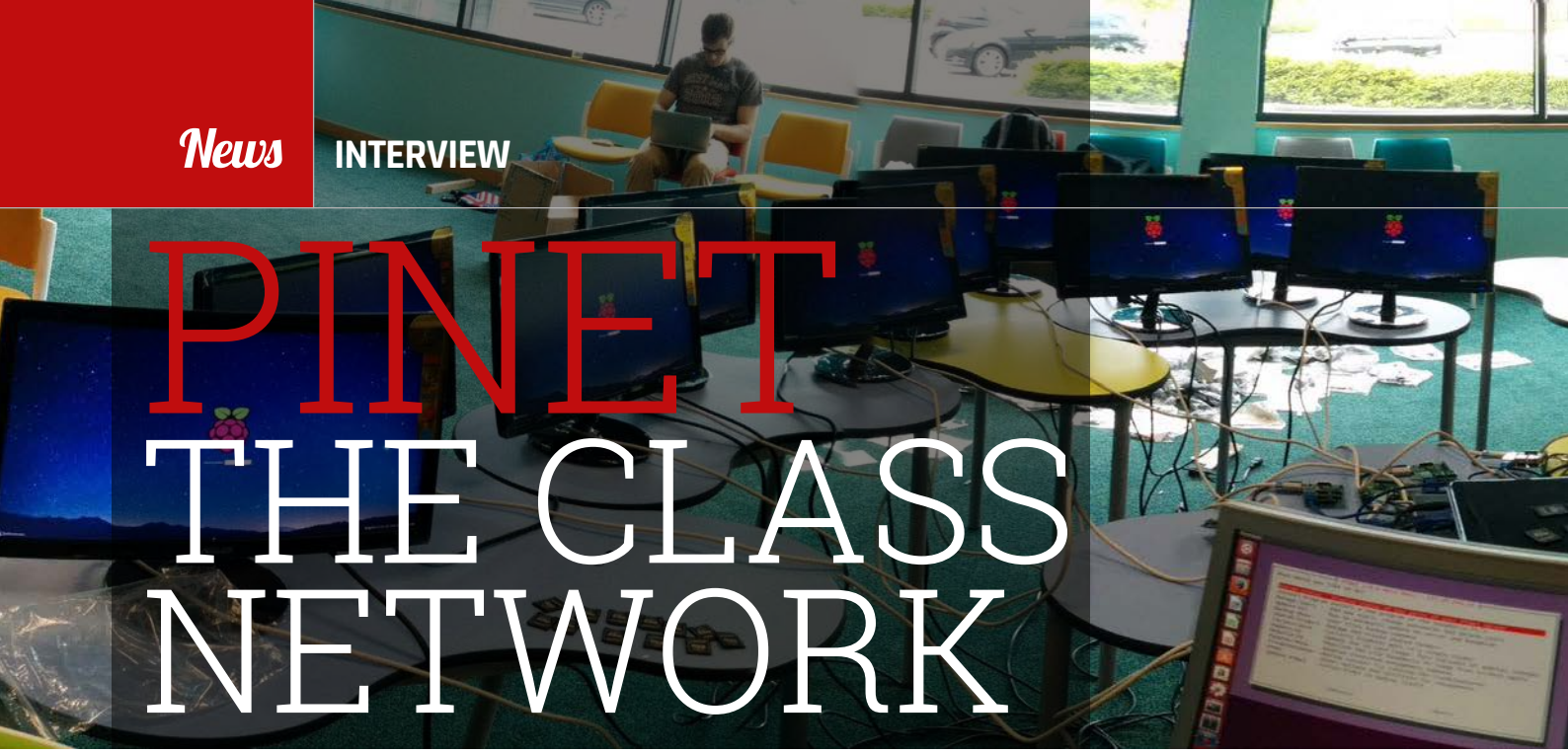
### Coffee Table Pi 26

Graham Gelding's fully fledged cocktail arcade cabinet is a sight to behold. We salute you!

## REGULARS



## REVIEWS



# PiNET THE CLASS NETWORK

Designed for the classroom, PiNet makes it far easier to teach computing with the Raspberry Pi. We catch up with its 18-year-old creator, Andrew Mulholland...

## ITS YOUNG CREATOR

Now a first-year computer science student at Queen's University, Belfast, PiNet creator Andrew Mulholland is still massively involved in the Raspberry Pi community. The prize money from winning the overall TalkTalk Digital Hero award for 2014 has enabled him to focus on continuing to promote computer science to youngsters: over the past seven months, he's led activities including Pi workshops, mentoring a beginner First Lego League Robotics team, and running the monthly Northern Ireland Raspberry Jams. "I have been able to show kids how awesome programming and making stuff is!"

In conjunction with the Farset Labs hackspace, he is also in the process of preparing a Pi-based introduction to programming which will be heading out to schools in Northern Ireland starting in June and continuing in September and October.

With all this and PiNet keeping him busy, he hasn't much time for new Pi projects, although a recent exception involved creating a full-size traffic light for a talk he was giving. He was recently also accepted onto the Raspberry Pi Creative Technologists programme, for which he's been working on a few ideas, "but those are still hush-hush for now."

What's better than a Raspberry Pi? A whole classroom of them! The only downside is that it can be laborious to install new software onto each Pi, not to mention keeping track of SD cards and their users. PiNet (**pinet.org.uk**) is an ingenious solution to the problem, enabling the creation of a network of Pis linked to a central server PC, which handles user accounts and network-boots the Pis using a master version of the Raspbian OS.

Formerly known as RaspberryPi-LTSP, PiNet is the brainchild of Andrew Mulholland, who was still a 16-year-old schoolboy when he came up with the idea while delivering a two-day Pi workshop at a local primary school. "I had eight Raspberry Pis and discovered on the evening of the first day that I had forgotten to install a piece of software onto the card," he recalls. "So I had to boot up every SD card and install the package. It got me thinking, there must be a better way... I ended up discovering LTSP (Linux Terminal Server Project), but found it stupidly complicated to get it working with the Raspberry Pi. But with some help from Vagrant Cascadian and Alkis Georgopoulos from the LTSP development team, we threw [together] a rough prototype and I went [on to] develop it from there."

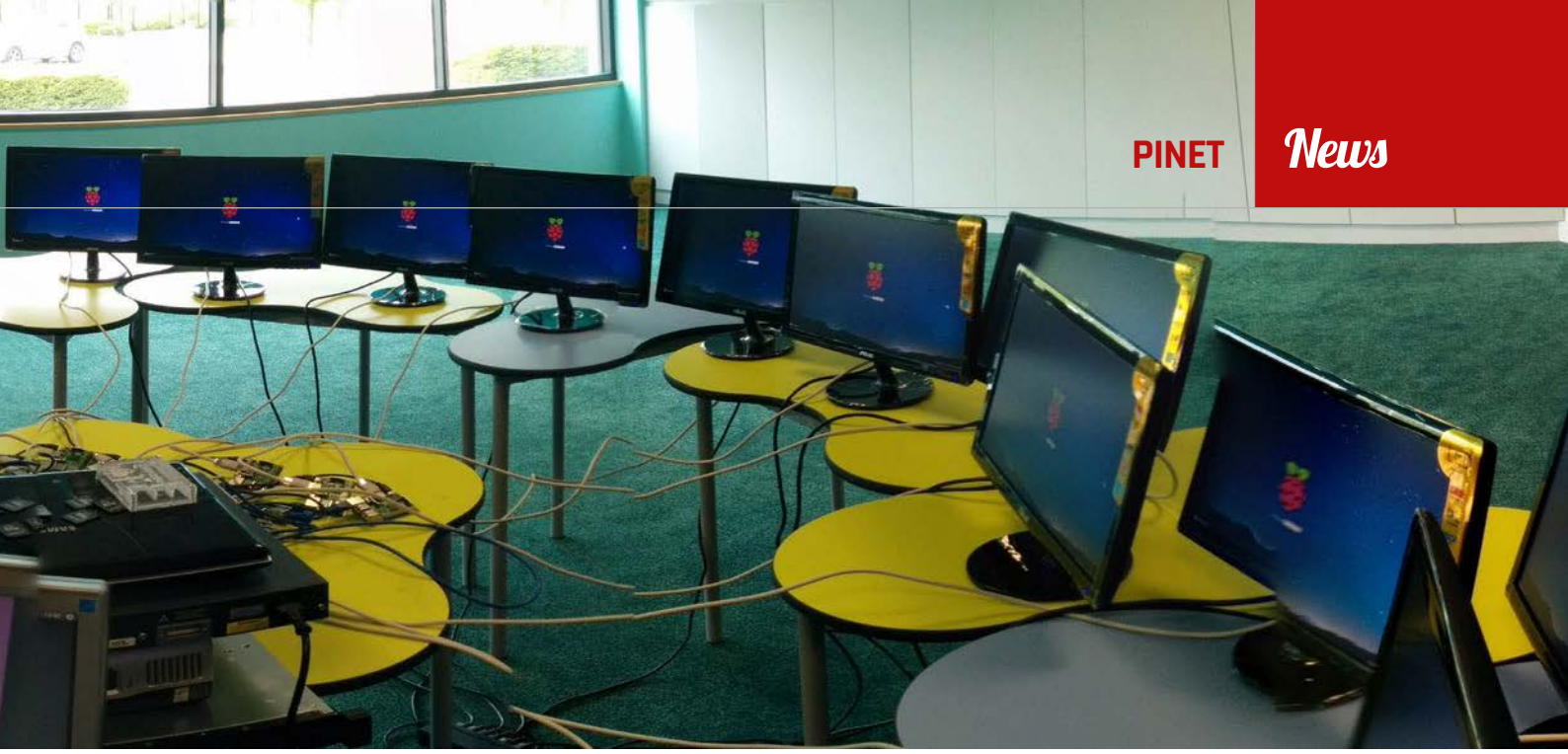
Andrew says he got a lot of great feedback from educators early on in the development of the project and, since PiNet is still very much in active development, he continues to receive plenty of emails and tweets from educators across the world suggesting new ideas. "Most of the features in PiNet have come directly from requests from educators."

### System relaunch

Some two years later, the free and open-source project has recently been relaunched as PiNet: "much easier to remember and a much friendlier name in general." As well as a rebrand, it comes with "a huge number of bug fixes, a lot more validation, and some new back-end stuff. We also relaunched the documentation site, which now is a lot easier to use."

As before, the project enables multiple Pis to be linked via a network switch to the central server, a PC running Ubuntu Linux 14.04 and the PiNet server software. Once installed, the latter creates a folder of files ready to be copied onto each SD card, so that each Pi can communicate with the server.

Due to high bandwidth demands, PiNet requires a wired Ethernet network: "A gigabit wired network usually has anywhere from 10-20 times the bandwidth of wireless N Wi-Fi," explains Andrew. "If you



> it enables multiple Pis to be linked via a network switch to the central server

think about it, you are loading an entire operating system over the network, or, as is the case for most classes, 30 entire operating systems over the network at the same time!" Despite this, boot-up takes only around 30 seconds longer than usual on most networks and each Pi then runs normally, suffering no slowdown.

According to Andrew, a gigabit network and medium-spec server PC can easily run 30 clients and in theory should be able to manage 60+. "Above that, you start hitting the limitations of a single Gigabit Ethernet connection, which is why I am also currently playing around with Ethernet bonding (trunking) – the process of using two gigabit network cards – and managed network switches to bump the bandwidth up to 2 gigabit."

## So many advantages

The benefits of PiNet are manifold. First of all, every Raspberry Pi in the classroom network-boots off a single server computer. "This means you as the teacher maintain a single master perfect operating system and this is loaded on every Pi. This has the massive advantage of giving you the option that if you discover, say, you need a certain piece of software for a class in 10 minutes, [there's] no problem as you can install it on the server copy and reboot all the Pis. That is it..."

Another key feature is the centralised login system: "The pupils' user accounts are stored on the server, not the Pis. This means any student can sit down at any Raspberry Pi in the school, log on and get stuck into their work."

Combined, these two features mean you no longer need an SD card for each pupil in the school to allow them to continue to work on the same project over multiple classes. On top of that, PiNet offers shared folders, automated backups, one-click installation of software, Epoptes classroom management software integration, and a whole series of other small features.

Simple to set up, it's no wonder it's already being used by hundreds of schools and organisations in over 30 countries. Educator Rob Jones, assistant headteacher at Cowley International College, says he would "definitely recommend PiNet for any schools moving from using Pis in a lunchtime or after-school club to part of a fully-fledged curriculum offering."

## PROJECT PI SETUP

**One of the main attractions of PiNet is how easy it is to set up, something the guys at Project Pi (projectpi.org.uk) sought to test recently by creating a network of 21 Pis. "We used an old Samsung laptop (R519) with 2.16GHz, 10/100 Ethernet [and] 2GB RAM," reveals Tim Greenwood. "We were concerned mostly with its ability to serve all 21 [Pis] a Raspbian image within a decent time frame. We chose this laptop to simulate the kind of PC that might be made available to schools or projects for free or at low cost. The Samsung R519 succeeded!"**

**Fellow Pi Project member Tyson Dye was equally impressed: "I've worked in corporate technology for more than ten years and nothing works first time, but I can't say that any more as PiNet worked first time!"**

**The team hope to use PiNet within schools and community projects around the world to make the Raspberry Pi even more well suited to the classroom. "PiNet makes administration of the Pi a cinch and that just opens the door to teachers and administrators [who] need to be able to maintain a working and up-to-date classroom full of Pis," says Tim.**

# PYGAME ZERO
## GAME CODING MADE EASY

It's designed to make it easier for teachers and students to start making games with Python. **Sean McManus** speaks to its creator, Daniel Pope...

If you've written a Python game, you've probably wrestled with Pygame. It's one of the go-to libraries for games programming, providing code for displaying images, drawing shapes, playing sounds, and writing text. The Python games in Raspbian also use it, but if you look at any Pygame code, you'll see that it can be a bit convoluted.

At the 2012 UK Python Conference, PyCon UK, professional programmer Daniel Pope participated in the education track. "I dashed off the most simple Pygame program I could, a rabbit sliding across the screen – about 15 lines of Python," he tells us. "The teachers told me it was too complicated to teach – that in the context of a 30-minute lesson, just tasked with reproducing my code, some of the students would still be struggling with non-runnable code at the same time that others were done and waiting for the next challenge."

He's now released Pygame Zero as a solution to that problem. "The simplest programs are just a couple of lines. And Pygame Zero lets teachers easily break lessons into bite-size steps: it feels like you can keep changing a couple of lines at a time and keep making meaningful progress."

Outside the classroom, it'll also help hobbyist programmers to get results more quickly.

Daniel is a two-times winner and nine-times entrant in PyWeek (**pyweek.org**), a week-long games programming competition. *Art Attack* from PyWeek 12 and *Legend Of Goblit* from PyWeek 19 were made using Pygame. His other games were made with Pyglet (**pyglet.org**), another library which had some influence on parts of Pygame Zero's design.

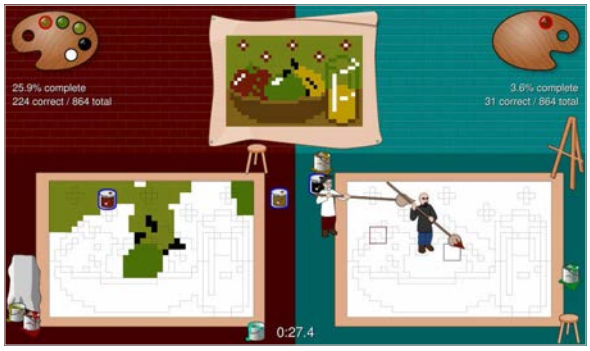
As an experienced Pygame coder, he could see the opportunity to improve upon it. "Pygame is a library that allows you to access input, load and draw graphics, play sounds – but you have to decide how to make the library do those things and write a little bit of code to do it," he says. "Mostly, people make the same decisions, so that code ends up looking the same. Pygame Zero makes some of the decisions for you, so your program doesn't need to include that code."

Pygame Zero also adds a few features on top of Pygame. "Pygame is just a library for input, graphics, sound, and so on," Daniel says. "We have provided some extra ability to animate objects, draw text in a number of styles, and schedule in-game events – for example, to create a power-up every 60 seconds."

### The main event

This is one way that Pygame Zero enables more events-driven programming. Instead of having to loop endlessly to check for a mouse click, for example, you can write a function that is triggered when the mouse button is pressed. Pygame Zero does the checking for you in the background.

"Event-driven code is a very natural way to write programs that respond to input from the user, a hardware device, or a network," says Daniel. "Most Pygame programs use an event loop but require you to write it yourself, and call your own code when certain types of events occur. In Pygame Zero, we keep the event loop inside the framework so you don't have to write it, and it will call any handlers you define. It's really just a simpler way of working




**Below** *Art Attack*, one of Daniel's PyWeek entries made with Pygame

### INSTALLING PYGAME ZERO

**Pygame Zero will be included in a future Raspbian release, but for now you can install using the Python package manager, pip. From the Raspbian command line, enter these commands to install pip (if necessary) and Pygame Zero (which utilises Python 3):**

```
sudo apt-get update
sudo apt-get install python3-setuptools python3-pip
sudo pip-3.2 install pgzero
```

**Read the docs at: bit.ly/PGZero**

with the same programming model, but it does feel different to work with, more organised."

The new library also simplifies image handling by introducing the concept of actors, which provide an easy way to load and manage picture files. These are ideal for game characters, because you can easily move them around by just changing their x and/or y position. There's support for positioning by one of the actor's corners too, so you can avoid the messy calculations that might otherwise be associated with that.

Pygame Zero can best help with 2D graphical games. "Retro games would be a very good fit," Daniel reckons, "but some more modern 2D smartphone games can be easily written too, like *Flappy Bird*."

For those who want to write more advanced games, Pygame Zero can be a useful stepping stone. "Pygame Zero is likely to be slow if you have a lot of things to draw," advises Daniel. "Lots of the ways of improving performance will require some control of Surfaces and when to draw or invalidate them. Support for that kind of thing is mostly in Pygame itself. Also, there are APIs in Pygame to rotate sprites, access webcams, joysticks, and more. So there's lots in the Pygame toolbox that you can experiment with as you get more experienced."


Photo by Richard Cooper

**Above** Daniel Pope: "We write games so that others can dip into our imaginations"

**Left** *The Legend of Goblit*, Daniel's winning PyWeek entry last year, based on Pygame

Pygame Zero improves compatibility by enforcing some simple rules. It requires images to be stored in a folder called 'images' and sounds to be stored in a folder called 'sounds'. There are rules on valid filenames too, to avoid cross-platform compatibility issues.

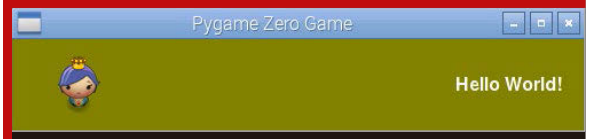
As for next steps, Daniel says the short-term focus is on ensuring that Pygame Zero meets its educational goals. "If something seems confusing to use, or crashes with an error message that doesn't give good information about what went wrong, that's a bug – let us know," he says. "Longer term, I'm keen to explore better ways of sharing and distributing games; I'd love to see a website dedicated to sharing Pygame Zero games. But we could also try to make Pygame Zero games work even without Python pre-installed. And maybe we will make progress on the lack of hardware acceleration in Pygame. I've spoken to people in the Python community who have part of the solution to these problems. Pygame Zero, Pygame, and Python are all open-source projects and benefit from the collaboration of lots of people solving little parts of the problem."

> "Pygame Zero enables more events-driven programming – a more natural way to code"

## Sharing is caring

"The reason we write games is to share them," says Daniel. "We write games so that others can dip into our imaginations and have fun with our ideas. It's always very disappointing when you want to play someone else's game but it doesn't work, and sadly this is often the situation when sharing games written on a computer with different hardware and software."

## HELLO_WORLD.PY




Try our example Pygame Zero scipt. Create a subfolder in the same directory called `images` and copy the file `princess.png` from Raspbian's `python_games` folder into it. To run the code, use `pgzrun helloworld.py` from the command line. The `update()` function is called once per frame, moving the princess across the window. Click the mouse button to make her jump.

Read the docs at: **bit.ly/PGZero**

```
# Pygame Zero demo from The MagPi
player = Actor('princess')
player.pos = 50, 50

WIDTH = 500 # window size
HEIGHT = 80

def draw():
    screen.fill((128,128,0)) # green stage
    player.draw() # show princess.png
    screen.draw.text(
    "Hello World!", topright=(480,30))

def update():
    player.x += 2 # move princess.png
    if player.topleft[0] > WIDTH:
        player.x = -50

def on_mouse_down():
    player.y -= 10 # jump up
    clock.schedule_unique(
    player_reset, 0.5)

def player_reset():
    player.y += 10 # fall down again
```

# KICKSTARTING THE WILDLIFE CAM KIT

Capture stunning wildlife photos with Naturebytes' Raspberry Pi-powered camera trap...

Since we first reported on the Wildlife Cam Kit in issue 32, Naturebytes' fledgling Pi-powered project has had its design finalised and is now the subject of a Kickstarter crowdfunding campaign. Co-founder Jon Fidler tell us the kit design has come on in leaps and bounds: "We've added a number of new features, such as a real-time clock and a special Poly IR PIR cover to ensure that the PIR's detection range wasn't reduced, yet the case is fully sealed. The bird feeder arm that featured in our early renders is also now available."

Having settled on a garden green colour for the acrylic case, which will be injection-moulded in the final kit, the team have been ensuring it's fully weatherproof. "Our kit has now been tested in all weathers out in the great British countryside," says Jon. "All open areas had to be fully sealed, so we also improved the lens cover. It wasn't really an issue as we just used clear 2mm laser-cut acrylic along with a laser-cut gasket."

The PIR sensor cover required a little more research and development, however, since it must provide protection from the elements but also allow the IR wavelength to pass through to detect wildlife. The team eventually managed to source a custom material from the US, "essentially a very thin plastic film to cover the lens and provide optimal IR detection for animals, both big and small."

## In any case

One key aspect of the kit is its versatility, with a customisable laser-cut insert (see 'Custom components' box nearby) enabling users to insert a choice of components, including any model of Raspberry Pi. "When you order your kit, you'll be able to select the insert you need for your own Pi, or you'll get an A+ insert with the standard full kit," says Jon.

There's also a choice of power source: while the standard kit features an 8800mAh Li-Po battery, Naturebytes' Alasdair

## CUSTOM COMPONENTS

The camera case's customisable component holders are laser-cut from acrylic. "We've designed the inserts so that they perfectly fit all of the Raspberry Pi models currently available," explains Jon Fidler. "The design is very simple, using spacers, screws and nuts, making it accessible to anyone with a small screwdriver."

A template for the insert will also be available for download via Naturebytes' community website, along with tutorials on how to edit them using open-source 3D design software. "The inserts can not only be laser-cut, but 3D-printed too," reveals Jon, "and if anyone makes a cool design, we will look to make that openly available to our community."

**Above** Some young makers try out the camera kit during a workshop

## FANTASTIC FOX

The first version of the kit's Raspbian-based OS, 'Fantastic Fox', will feature an easy-to-use GUI and be configured to run all of the Naturebytes activities out of the box. In addition, it will include simple guides for beginners wanting to try out Python, enter terminal commands, and explore the Linux environment that may be new to some users. "We're also focusing on a fun experience for schools," adds Alasdair Davies, "and an OS that's teacher-friendly will be high on our agenda." Another major consideration is enabling users to submit important data to active conservation projects and assist with their research: "An OS that makes it easy to share data and bind the community together will be really exciting."


Davies tells us it will also work with Power-over-Ethernet (PoE) via a cable access grommet in the bottom of the case. "Right from the start we knew that a lot of people would want to hook up their kits to a permanent power supply… PoE is exciting as you could power the kit from your garden shed, for example, by burying the cable."

Solar power is another potential option, for which Naturebytes is planning to release an add-on kit in the future. Other planned add-on 'pods' include night-vision (using a servo to switch between standard and Pi Noir cameras), a mic for sound capture, Wi-Fi adaptor for wireless connectivity, and weather capture (via communication with a Pi weather station).

For now, however, the first official add-on is a bird-feeder arm. "It's extremely versatile as it can be attached to a wall or hung from a tree, or even a balcony if you don't have a garden," says Alasdair. Made from sustainably sourced plywood and easily assembled, the arm also allows for the camera to be easily detached when the user wants to take it inside.

The kit's Raspberry Pi can also be easily removed from the hinged case when required, and will run a custom Raspbian-based operating system that is designed to be inviting for beginners, yet flexible for experts (see 'Fantastic Fox' box above).

### Kickstarter campaign

Naturebytes is currently raising funds through Kickstarter to undertake the injection mould tooling for the camera case, having 3D-printed the prototypes. It's aiming to supply the full Wildlife Cam Kit, including a Pi Model A+, for £95, which will be delivered approximately three months after a successful Kickstarter campaign. Many other options are available, however: '"We've created a range of rewards to make sure that anyone who would like to support us can get their hands on something that's just right for them, regardless of ability or budget," insists Alasdair. Options include a stripped-down, 'print your own' Developer Kit, which comprises all of the internal components and CAD files necessary for experienced makers to modify and hack the kit. At the top end of the range is the Computer Kit, adding various peripherals and coming with an optional bird-feeder attachment. In addition, there are bundles of ten kits plus workshop teaching materials, while for £3,000, organisations will get a specialist weekend workshop from the Naturebytes co-founders. Education is a key aim of the not-for-profit project. "One of our key goals is to encourage young people to become digital makers," explains Naturebytes co-founder Stephen Mowat. "We're doing this by providing educational resources mapped to the syllabus for teachers and educators to download and experience Naturebytes' exciting wildlife-based activities in the classroom, [which] will focus on developing key skills, supporting creativity, and reconnecting young people to nature through digital making."

> "It's extremely versatile as it can be attached to a wall or hung from a tree, or even a balcony…"

You can find out more about the kit on Kickstarter (**kck.st/1Ndsg99**) or **naturebytes.org**.

**Below** The bird-feeder arm can be used to attach the camera kit to a bird table

**Right** The Pi mascot, Babbage Bear, beats Felix Baumgartner by reaching 39km

# SPACED OUT

The MagPi looks at **Dave Akerman**'s frequent and eye-catching forays to the edge of space as the Pi's foremost near-spaceman...

It may have proven to be one small step for man, but space travel has certainly been one giant leap for the tiny Raspberry Pi. When the makers of this bare-bones computer came up with the device, no one in their wildest dreams would have thought it would boldly go where few other machines have travelled before: to the stratosphere and back. But thanks to the Pi community's very own 'space man' Dave Akerman, that is exactly where it has been, and the results have been nothing less than stunning.

**Below** Dave Akerman and Heston Blumenthal during filming for the celebrity chef's UK TV series

Dave is a high-altitude ballooning enthusiast who has been tethering Raspberry Pis to helium balloons and sending them to the edge of space since 2012. His hobby and choice of computer have been attracting much attention since, leading to a rather hectic life for the software programmer, and helping to show further evidence of the adaptability of the Pi.

In the Pi's lifetime, Dave has appeared on television, become something of a YouTube sensation and worked with a celebrity chef, making him one of the most well-known users in the ever-growing Pi community. Not that he's complaining about becoming a Pi celeb. "The Raspberry Pi had two big effects [on my high-altitude ballooning]: the addition of live images and all of the media attention," Dave tell us. "I expected the former, but not the latter. It's all been good, though."

## High-profile performances

One of his most high-profile recent performances came on the BBC 2 show *Stargazing Live* on 20 March. Dave was invited to rub shoulders with the likes of the European Space Agency astronaut Paolo Nespoli and Paul Franklin, the visual effects supervisor of the Hollywood movie *Interstellar*.

Stationed at Leicester racecourse, Dave was asked to launch a 434MHz balloon equipped with 'Pi in the Sky' telemetry boards to capture stills and video from above the clouds of the solar eclipse taking place that day. The Raspberry Pi soared to a maximum altitude of 30km, taking in images, before popping and landing in a field just south of Leighton Buzzard - a huge success which delighted viewers and the BBC. "It was just a shame that there was so little time to explain the flight," says Dave. "The images shown weren't near our best ones either, but these are

the limitations of live TV."

Even so, the flight proved to be a thoroughly enjoyable experience for the radio amateur. "There was so much rehearsal that by the time the cameras were live, I was pretty relaxed," Dave tells us about his long, busy and exhausting day. Unfortunately, he did not get to meet the main presenters, Prof Brian Cox and Dara O'Briain, since they were based at Jodrell Bank, just a few miles from Manchester Airport – "I'd have been unlikely to get permission from the CAA to launch – and yes, I did ask!", he says. Nevertheless, Dave is fired up about the total solar eclipse that will take place on 21 August 2017 in the USA. "The whole thing was good practice and I can't wait for that one."

Indeed, while thinking up and taking part in a good number of headline-grabbing events, Dave has been able to refine his methods, regularly coming up with new and more efficient ways of 'exploring' life beyond our planet. "The first Pi I sent into the air used a webcam, and that's been replaced by a Pi camera," he tells us.

"Back then, I also handmade the radio boards, but now I use PCBs, which makes things simpler and more reliable. The very first Pi flight used linear regulators with more batteries than was strictly necessary, meaning that the inside of the payload box got very, very warm. Nowadays the tracker board and Pi all use switching regulators, so fewer batteries are needed."

## The LoRa module

To make the Pi more useful for his hobby in the future, Dave has been working on refining a long-range radio (LoRa) module for the Pi. It was on board the balloon used in *Stargazing Live*, but the media activities surrounding the launch meant Dave did not get a chance to set up the tracking. Had he done so, he would have been able to eliminate the need for a PC, since LoRa devices are transceivers – they receive as well as transmit. The hard work of demodulating the signal is done

> "The Pi is an ideal device, allowing the telemetry to be automatically uploaded"

## GET INVOLVED

David Akerman discusses what it takes to be a Pi space man or woman...

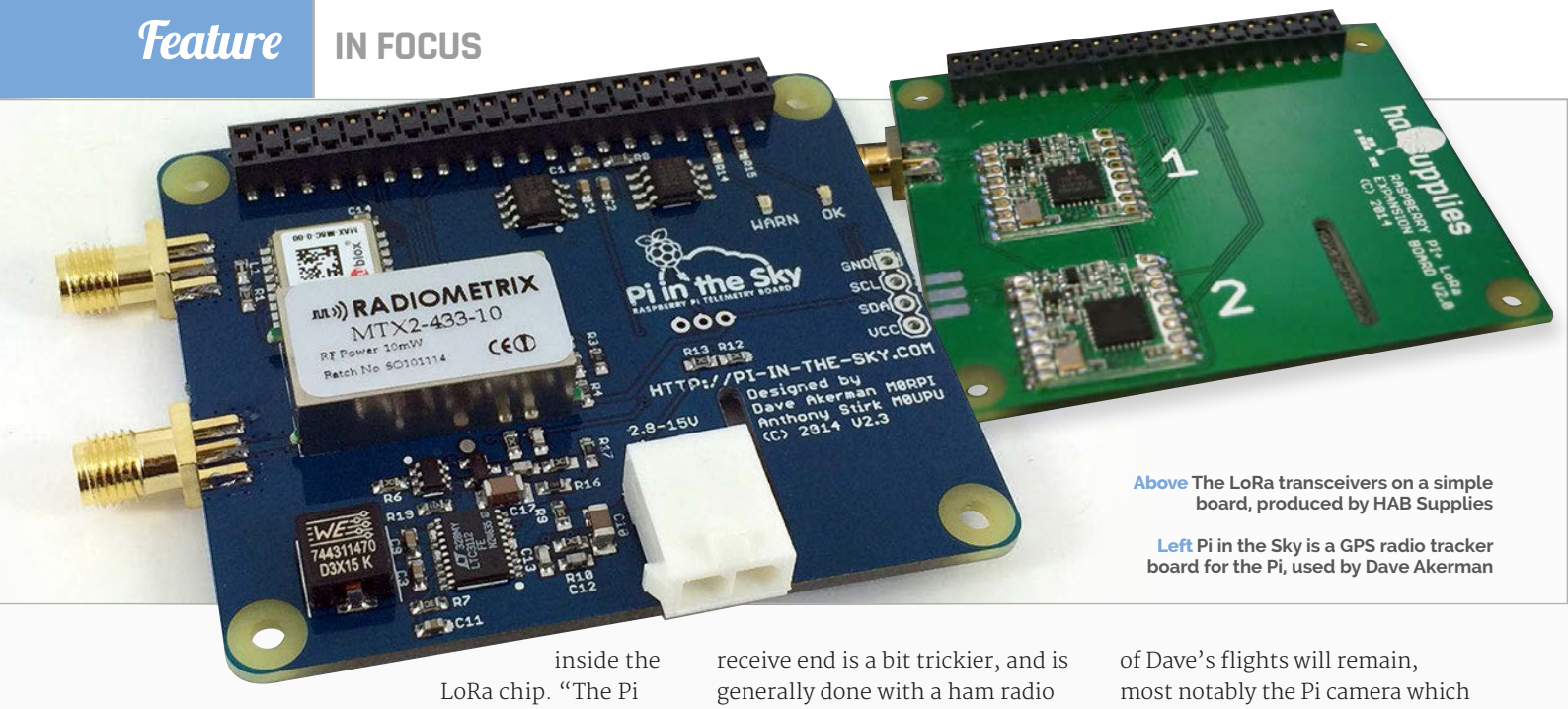
**MagPi: What should amateur Pi fans be doing if they want to get involved?**
**Dave Akerman:** Research! There's a lot of information on the web, with the UKHAS wiki (**ukhas.org.uk**) being the best single resource. Also, I encourage everyone to design, build and code their own tracker, as it's much more rewarding to fly something of your own design than to just fly an 'off the shelf' tracker. That said, if you are short of time or skills, then you can buy a Pi tracker board rather than make one, and/or you can use open-source software rather than write your own. But, if you can DIY, please do.

**MagPi: What will readers learn along the way?**
**Dave:** Lots. Whilst high-altitude ballooning isn't complicated (pause as I wheel out an old joke) – it's not rocket science – it does encompass many fields, such as physics, weather prediction, how the atmosphere works, electronics, radio, and software.

**MagPi: Which Pi model would you recommend?**
**Dave:** These days the A+ is the obvious choice. What you need from a tracker computer is light weight and low power consumption; you don't need higher power consumption, extra USB sockets, and a network interface!


**Behind** A view over Cornwall taken from a high-altitude balloon carrying a Raspberry Pi

**Below** Dave Akerman being interviewed on TV by Dr Lucie Green




**Above** The LoRa transceivers on a simple board, produced by HAB Supplies

**Left** Pi in the Sky is a GPS radio tracker board for the Pi, used by Dave Akerman

inside the LoRa chip. "The Pi is an ideal device, allowing the telemetry to be automatically uploaded to the internet for display on a map," says Dave. "It also becomes possible to upload messages to the balloon tracker."

LoRa will replace the traditional RTTY (radioteletype) balloon tracker for Dave, even though the older system is very easy to program and has been used for around 95 per cent of amateur high-altitude balloon launches in the UK. "It's essentially the same as RS232 serial communications, with the ones and zeroes being denoted by two slightly different radio frequencies," explains Dave. "On the Pi, all that's needed is to connect the serial port to a small radio transmitter via a few resistors, and then send the data out of the serial port. The receive end is a bit trickier, and is generally done with a ham radio receiver and a PC."

Dave has carried out four LoRa test flights so far, the first of which landed on a golf course during a competition. "The payload was collected by one of the golfers, who wrapped it round his trolley, which explained the fact that the landing position kept moving!" Another test flight had two LoRa trackers, one receiving data from the first and then relaying to the ground over an RTTY link. "Most impressive, though, was a trial of high-data-rate images, where the incoming packets managed to saturate the uplink on my admittedly slow ADSL internet," Dave adds. "More work is needed, but this does look promising."

Yet even though LoRa modules are being introduced, some aspects of Dave's flights will remain, most notably the Pi camera which replaced a webcam on the balloon's payload as soon as it was released. "The Pi camera quality was a big improvement on the webcam," he tells us. "It was lighter too, which helps. Previously, I used Canon compact cameras for stills, or a Kodak camcorder for video."

## Video capture

Such video-taking capabilities have proven to be very useful, especially during some of the publicity-generating stunts that Dave has pulled. His favourite involved taking a 20cm-tall teddy bear of the Raspberry Pi mascot, Babbage, to an altitude of 39 kilometres and dropping it from the sky. He wanted the toy to rival human daredevil Felix Baumgartner, who had set the world record by skydiving from a height of 38,969 metres in October 2012.

For the 2013 Babbage flight, the teddy dropped at speeds of up to 200 miles per hour as a fitted camera filmed what the toy 'saw'. Landing four hours later in a field near Shaftesbury, the teddy – which contained a Pi and a tracking device in its stomach – was intact, and its endeavours eventually came to be watched by more than 160,000 YouTube users. "I wanted that downward video of the [Baumgartner] jump, but with a slightly lower budget and rather less seriousness,"

**Below** The potato in space, as filmed from a GoPro

Stills of Earth as the Pi is sent into space, as part of filming for the solar eclipse

says Dave. "I'd bought a Babbage soon after he went on sale, with the hope that I'd be able to fit a Pi Model A inside him, and was pleased to see that it just fitted. The hardest part was replacing Babbage's eye with a Pi camera – those eyes are very, very well fixed in." Babbage beat the world record by 31 metres.

This 'mission' showed Dave's eye for detail, even though there were problems. At one stage he was going to build a replica of Baumgartner's capsule, until it appeared it would be too heavy. Tests had also shown a reluctance for Babbage to jump. Even when it transpired all had gone to plan, Dave's attempts to track the toy were hampered by the SIM card in his MiFi (mobile Wi-Fi hotspot) running out of credit. "I also had a call from the BBC asking about the progress, just as I was trying to find Babbage in a field. I managed to walk straight past him."

> "For the 2013 Babbage flight, the teddy dropped at speeds of up to 200 miles per hour"

However, as with Dave's other exploits, publicity followed. Two journalists from Austria even accused him of faking the flight as a publicity stunt for the Raspberry Pi Foundation. "Well, like the moon landings, it was easier to do it than fake it," laughs Dave. Not that the launches are actually simple. He says anyone wanting to replicate Pi in the Sky flights would need to spend a good few months researching first, and even he has had his fair share of mishaps. "I did manage to lose a couple of payloads to the sea: one early on, when I was unaware of the effect of wind on gauging the amount of gas in a balloon, and one later on, when the predicted flight path was completely wrong because the Met Office didn't fly its own balloons the day before."

But with the top four flight records for live images at around 40 kilometres altitude, Dave's hobby is, he adds, a thrill that never leaves him. "On the very first flight, the thrill was to hold something in my hand that had, to all intents and purposes, been to space," he says. "Prior to that flight, that was all I wanted, plus images that I had taken from near space. For many people this is enough, but like a few others in this hobby, I like to find new things that I've not done before."

## CHEF ENJOYS A SPUD LAUNCH

In October 1995, the potato made history when top scientists working for NASA developed a special kind of spud: one that could be grown in space to feed hungry astronauts. This amazing fact was not lost on English celebrity chef Heston Blumenthal when he was creating his Channel 4 series, *Heston's Great British Food*.

For the 'Pie' episode, Heston decided it would be a good idea to use Dave Akerman's high-altitude Pi in the Sky balloon to launch a potato. "My first question was 'are you expecting anything physical to happen to the potato?', thinking that Heston probably wanted it freeze-dried or something," says Dave. "But no, it was just to put it into near space and get it back again. No problem then."

To achieve this, Dave built a payload with a Raspberry Pi doing the tracking and live imaging, plus three GoPro cameras – one up, one down, one sideways – for video. There was a second AVR tracker in a separate payload as a backup. Dave then had to find a day when the wind predictions were suitable and Heston was available. "That was not an easy task," he adds.

But how receptive was Mr Blumenthal? "Heston, as you might imagine, is basically a ten-year-old adult, so he and I got on just great," Dave replies. "We started chatting and the producer stopped us, explaining that he wanted cameras to record Heston's reaction when he learned things for the first time. We then had to drive up and down the road for some footage, during which Heston kept starting sentences like 'So, when the balloon bursts...' and I'd reply with 'Sorry, can't tell you yet!' It was obvious that he'd been looking forward to the day for some time."

The flight was straightforward, with the crew following the live images in the chase car. Quite fittingly, the landing was in a muddy field and lots of coverage was gathered for the TV programme. "It was great to see that when it eventually got broadcast, and it was a lot of fun to then see the flight ridiculed a few days later by [English comedian] Alan Carr."

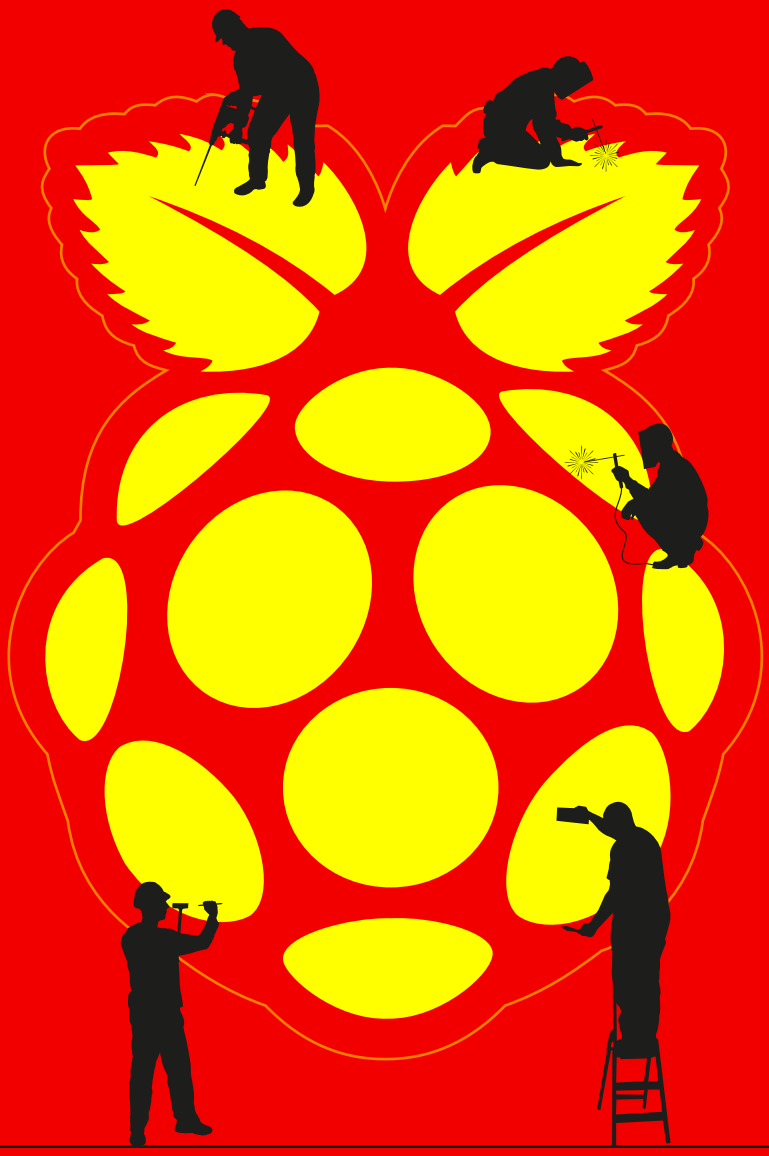
Even so, it piqued a lot of interest. Following the show, Dave had a noticeable increase in the number of emails received from people who wanted to follow in his footsteps. "Flights like these do encourage others to get involved," he says.

## SKYCADEMY

**Are you a teacher or your youth leader? Would you like to learn more about high altitude ballooning? Learn more and sign up: raspberrypi.org/picademy/skycademy/apply**

# AMAZING PI PROJECTS

10 imaginative creations that *innovate* and *inspire*



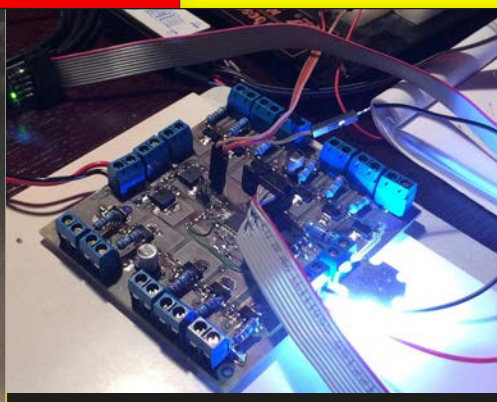
While the Raspberry Pi is to all intents and purposes a simple Linux-powered PC, it's sometimes difficult to convince people on the street that they merely need to connect a standard TV or monitor and keyboard and mouse to use it like any regular computer or laptop. They struggle to comprehend that something so small and strange in appearance is anything other than alien in origin. The difficulty is usually intensified when you talk to them about all the incredible things people do with the Pi. Sending Raspberry Pis to the edge of space to take pictures of the curvature of the Earth, configuring them to adjust the temperature or control the lighting in their home… how can they hope to even turn it on, let alone take control of their home?

Amazingly, turning on a light is usually the first step for anyone wanting to make use of the Raspberry Pi's extensive physical computing repertoire. All it takes is a few lines of code and a basic grasp of elementary electronics (if that). Once they've cracked it, though, everything else is common sense and trial and error. One minute a tiny LED light is blinking; the next, your robot is deciding whether to turn left or right.

Once that lightbulb moment occurs and the initial culture shock has subsided, the final blocker is usually indecision. What do you do with a credit-card-sized PC that can go anywhere and do anything? That's the focus of this feature – 12 pages of ideas, inspiration and advice from some of the community's best hackers and makers.

# Project Aquarius

Two brothers recreated the Amazon rainforest in their home, including indigenous flora and fauna. Weather, sound and lighting effects produce accurate day and night cycles…



**Above** It uses a custom PCB to aid control

**Above** It simulates light, sound, and weather

## Simulating THE AMAZONIAN RAINFOREST

Poopi and Piter's paludarium is controlled by Raspberry Pi and four ATmega 168Ps. Here is its extensive feature list:

- 6× independent sections of halogen lights
- 27× independently controlled 1W LEDs for various effects
- 3× independent 3W RGB LEDs for ambient colour effects
- 3× independent 3W LEDs for thunder and moon simulation
- 3× independent 10W LEDs for paludarium lighting
- 2× independent fans for wind simulation
- 3× fog generators
- 2× independent solenoids for rain control
- Temperature monitoring

All lights are fully dimmable, and Poopi and Piter also have full control over fan speed.

Wojciech 'Poopi' Lazarski is a technology evangelist from Poland. Poopi was once a developer in the demo scene, way back when the ZX Spectrum and Amiga were the best show in town. These days Poopi and his brother-in-law Piter are more concerned with recreating the Peruvian rainforest – specifically, a region called Rio Tahuayo – in their home with their custom-made paludarium.

"One day my brother-in-law showed me a video on YouTube and said it would be cool to have something like this. He said that making the tank wouldn't be a problem, but creating [the project] would be impossible for him. I asked him to give me time to digest it and a week later I said yes, but that I'd like to do it my way. I didn't just want to add lighting effects so I started thinking about a more sophisticated solution."

At the start, it controlled lights to create the illusion of night-time with the phases of the moon, and sunrises and sunsets with all the associated colour effects. Everything is accurately calculated based on the target location. Later, wind simulation was added using two fans, and then mist and rain effects using solenoids. Since delicate flora are present, careful temperature monitoring is also of utmost importance.

You can see Project Aquarius in action on YouTube (**youtu.be/FeS5zqL8frk**), but stay tuned – we'll be bringing more in-depth coverage of Poopi and Piter's project in the coming months.

# One Controller to Control Them All

Retro gaming is a popular avenue for the Raspberry Pi, but few have done it with as much flair and precision as Brian Corteil…


All images courtesy of Alex Eames www.raspi.tv

Brian Corteil is a Cambridge Raspberry Jam regular and founding member of the Cambridge Makespace (**makespace.org**). Brian is well known for the remarkable quality of his projects and their stunning attention to detail.

"I wanted to make something my two boys could set up and play with, without me being around. It's also a great honeypot to draw people in to my table when I show off at Maker Faires and Raspberry Jams [see sidebar]."

Brian's retro gaming console-cum-arcade controller is certainly that, and it should be of little surprise that it's just as neatly presented on the inside as it is on the outside.

"I used one of the laser cutters at the Cambridge Makespace to cut [and] engrave the box and back-plate. I also used Inkscape (**inkscape.org**) to design the back-plate and modify the box design from Steve Upton's box-generating script (**officeoffairetrade.com**). The RetroPie SD card image takes care of the software side of things (**bit.ly/1lYZkDg**)." digitalLumberjac's arcade joystick driver (**bit.ly/1xHKQCz**) also proved to be the best way for Brian to interface the joystick and buttons to the Raspberry Pi's GPIO pins.

Brian keeps an excellent log of his many Raspberry Pi projects on his website (**bit.ly/1LyTQjR**), though he's a little behind in cataloguing his adventures. "I'm not sure when I'm going to have time to write this up. Maybe over the next two or three weeks."

## Make it HAPPEN

Brian is one of the founding members of Cambridge Makespace (**makespace.org**), which gives him full access to the tools and equipment (like 3D printers and laser cutters) he uses in his projects, not to mention the expertise of fellow members.

While trailing its US equivalent, the UK maker scene is growing fast, with new locations popping up all the time. Try searching online for 'maker spaces' or visit **hackspace.org.uk**.

There are also numerous online services that will do the 3D printing or laser cutting for you and send the results by post.


# Digital Zoetrope

Brian's second project is a deliciously modern take on a pre-film animation device that produces the illusion of movement with still images...

"I started the Digital Zoetrope over Christmas 2014 and I'm still refining it. I was inspired from the work of Eadweard Muybridge, an early pioneer of high-speed photography [who] was the first person to show that a horse takes all of its feet off the ground when it's running. Researching his work led on to Zoetrope and I thought it would be a wonderful project."

As with his retro-gaming console, Brian utilised the laser cutter at Makespace, and Inkscape for the computer-aided design work. Despite bundling a wealth of technology, his Digital Zoetrope is moved by hand. Like the original designs, you spin the device and look through the slats to see movement in the still images as they rotate.

Since his project uses 12 OLED displays with the Raspberry Pi, it's actually possible to update the frames in real-time, so you could watch an entire film if you wanted. Moreover, using the technological trickery, it's possible for two people to view entirely independent animations when looking into the Digital Zoetrope from different angles.

Since Brian doesn't anticipate blogging about his Zoetrope any time soon, we'll be covering his project in more detail in the coming months.

## Organise
## YOUR OWN JAM EVENT

Raspberry Jams are a great way to get together with like-minded hackers and makers. They're also the best place to find inspiration and help to build amazing Raspberry Pi projects. You can find a Raspberry Jam near you by visiting our events pages at the back of the magazine, or by visiting **raspberrypi.org/jam**. Can't find an event near you? Don't worry – it's easy to set up your own! Here are a few tips from Cambridge Raspberry Jam (**camjam.me**) organisers Mike Horne and Tim Richardson:

1. **Find a partner and team:** Don't try to organise everything on your own. Find a core team of two or three people who can meet regularly and gather helpers on the day.
2. **Get a venue:** It must be accessible with good parking and transport links, and have the space you need to do what you want. It could be as small as your kitchen table or as large as a local community or sports hall.
3. **Get talking:** Everyone likes 'show and tells' and talks, so start with those. You can add things like workshops, sponsorship and vendors later.
4. **Walkthroughs:** In the run-up to every Jam, put yourself in an attendee's shoes. Imagine walking in through the front door for the first time. What do you see? What do you need to do?
5. **Questionnaires:** Just after the end of the event, send out a questionnaire to all attendees to find out what was good and what could be improved. Use it to make your next Jam even better!

# Internet of Things Chessboard

16-year-old Ben James spent two months designing and building his IoT chessboard, and three months developing the software. The results speak for themselves...

Ben James is a home-educated Raspberry Pi enthusiast who plans on becoming a robotics engineer when he's older.

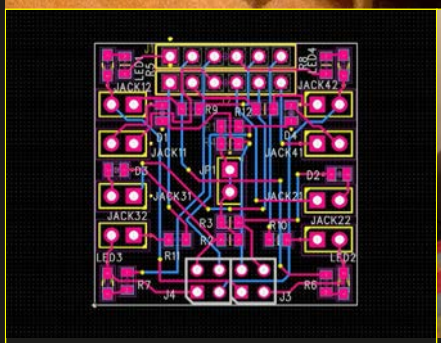
"The inspiration for this project came about one afternoon when I felt like playing a game of chess, but there was no one to play with. I would have gone online and played, but that's a 2D diagram view that feels really unintuitive, so I decided to make a real physical chessboard that could play online."

To play, your opponent requests a new game on the chessboard's own website. "When they make a move by dragging a piece on the web interface, the relevant piece and square will flash on the IoT Chessboard. Whoever's sitting in front of it will move that piece to update the board, then will make their own move. The board detects this, then sends that move back to the opponent's web UI, where the piece is moved on the screen."

According to Ben, the biggest challenge of the build was desiging the circuitry. "Each of the 64 squares on the chessboard has to have an individually addressable LED on its surface, an LED on the piece sitting on top of it, and a detection circuit to determine whether there is a piece on it."

Instead of controlling an unwieldy 192 input/outputs, the project utilises multiplexing. "Each group of four squares is handled by a PCB, then the 16 PCBs are connected horizontally and vertically in groups of four." While it makes I/O much more managable, it required over 2,000 solder joints and made troubleshooting issues a gargantuan task. "It was more efficient in the end, though, cutting the I/Os down to under 60, and it meant the software was a lot easier to write."

See it in action at **youtu.be/bWeObKths-I** and learn more at **engineercheer.wordpress.com**.




**Above** The circuitry is impressive

**Above** Ben had to solder 2,000+ joints

# Flappy Brain

Control a Flappy Bird clone with your brainwaves? It's mind-blowing stuff...

Images courtesy of Alex Eames www.raspi.tv

Albert Hickey is a coordinator of the Egham Raspberry Jam (**bit.ly/1HjrkBS**). His latest project uses a MindFlex band (a Mattel game from 2009) to read your brainwaves and convert the data to allow you to play a version of *Flappy Bird* with nothing but the power of your mind.

In a recent interview with Alex Eames on RasPi.TV (**bit.ly/1FEvmNK**), Albert explained more: "The headband features a little piece of metal that you put against your temple and it reads your brainwaves." It's connected to an Arduino that converts this information into usable data that's then passed over to the Raspberry Pi and used as controller input for the game. "Just search online for 'MindFlex Arduino Hack' and you'll come up with a page that explains how do it."

Albert used Pygame to create *Flappy Brain* on the Raspberry Pi and, much like the game on which it's based, you need to move the protagonist (in this case a tiny brain) past a series of obstacles.

Unlike *Flappy Bird*, though, the brain automatically moves up the screen – to move it down (and past the obstacles), the user needs to concentrate; Albert demonstrates by doing mental arithmetic to move it down. To allow it to move up again, Albert blinks repeatedly to break his concentration. Amazing! Learn more at **winkleink.blogspot.co.uk**.

## PiWars COMPETITION

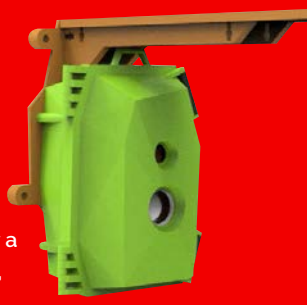
If you're looking to find amazing Raspberry Pi robots, look no further than **piwars.org** – it's a UK-based competition organised by the makers of Cambridge Raspberry Jam (**CamJam.me**) to bring home-made Raspberry Pi robots together to compete in a number of events and categories. We'll be sharing news of this year's event, scheduled to be held in December, in the next issue of the magazine. You don't need to be an expert to attend, but check out this excellent video by RaspberryPiIVBeginners from last year's event to get your creative juices flowing: **youtu.be/PQyoDZzQJIY**

**Above** One of last year's smallest bots

## Crowdsourcing SUCCESS

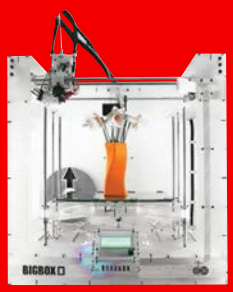
As we're seeing in this feature, Raspberry Pi projects come in all shapes and sizes. If you've got a really big idea and you'd like to take it to the next level, you might want to try crowdfunding it on a platform like Kickstarter. Here are two examples we've got our eyes on...

### NATUREBYTES WILDLIFE CAM KIT



The Naturebytes Wildlife Cam Kit incorporates a Raspberry Pi and a Raspberry Pi Camera Module that's triggered by a PIR motion sensor, to help just about anyone take candid shots of birds and animals in their natural habitat. The team's Kickstarter campaign has just started – learn more on pages 10 and 11 or visit the funding page at **kck.st/1Ndsg99**.

### BIGBOX-3D



BigBox-3D incorporates two of our favourite things in the world – Raspberry Pis and 3D printers – so we're naturally rather excited at the prospect of this forthcoming crowdfunding campaign.

It's the brainchild of E3D, a British engineering company with extensive 3D printing experience, and LittleBox, which has previously found success on Kickstarter with the MicroSlice mini laser cutter.

The idea of the BigBox-3D is to make it easy and affordable to achieve high-quality 3D prints. The project uses OctoPi (a cloud-based print solution) and the Raspberry Pi Camera Module for a remote live view of your prints as they happen. Learn more at **bigbox-3D.com**.

**FERRAN FÀBREGAS**

A Spanish computer scientist who works at a consultancy for urban ecology by day, by night he is an active member of the Barcelona maker community.
**lifebox.ferranfabregas.info**

# LIFEBOX

## Quick Facts

- The project took two months to complete
- Ferran learnt how to laser-cut wood for the project
- It is apparently not inspired by Conway's Game of Life
- The code is available on GitHub: **bit.ly/1T8VKtC**
- There's a mini version in development

Creating artificial intelligence on the Pi may sound like the start of the robot uprising, but the LifeBox isn't taking any chances, and has imprisoned them in lights...

There are so many different type of light-display projects on Raspberry Pi that you could be forgiven for thinking that the LifeBox was just another neat little programmed series of LEDs. This would be a huge mistake to make because the little lights are a lot cleverer than you could imagine: they're alive. Well, sort of. At the very least, they have been programmed with behaviour. Instead of launching into a thesis on when artificial intelligence can be classed as alive, we're going to concentrate on the LifeBox itself. Here's what it says on the side of the box:

"In this box live two pixelic entities, the blue and yellow species. These two species compete to survive and reproduce, feeding with the white mana that grows under [their] feet.

"Each species has eight configurable variables that can change their behaviour. The white mana also has five parameters that determine their behaviour and also rule the future of the two species that feeds.

"Learn the basic concepts of programming and biology being the god of these entities, varying all the parameters and seeing the consequences of your actions in the LifeBox!"

Its creator, Ferran Fàbregas, explains it to us in a less poetic manner, but one that makes more technical sense.

"LifeBox, in short, is a virtual ecosystem simulator on a 32×32 RGB led panel," Ferran tells us. "It's composed of two species that compete for the resources (mana) to get energy, survive, reproduce, and grow.

"Both species and the mana (which actually acts as a different species itself) have a user-defined parameterization that allows [the user] to change their behaviour and see the consequences on the panel, acting as a god of the virtual ecosystem."

> "Learn the basic concepts of programming and biology being the god of these entities"

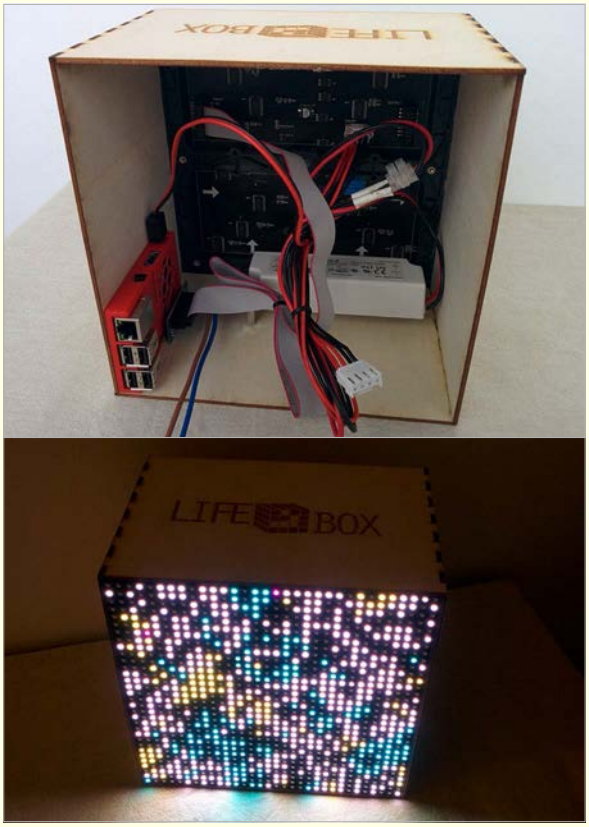
Ferran's god complex has been with him a while, as well as the interest in simulating these kind of ecosystems:

"Since I was a child, I was attracted to robotics and the possibilities of simple life simulations. I programmed some basic life simulators software before, but one day I found a shiny, beautiful, and cheap 32×32 RGB LED panel at a fair and I decided to create the LifeBox. I was captivated by the idea of seeing the evolution of a programmable ecosystem on a beautiful box on my dining room [table], like other people can enjoy with a fishbowl."

Inside the box is just a Pi, the LED panel, and a driver to connect the two. Because of this, the code is split up into two sections: one part to control the driver, and one part that controls the simulation.

"The driver is based on the great work of the original C++ driver by Henner Zeller and a reimplementation in C by Peter Onion with slight modifications," Ferran explains.

"The simulator is build in C and the main goal is to maintain it as simple as possible, so anybody can change not only the species parameterization but also the simulation algorithm itself (although this is not the objective, because it can result in an unusable LifeBox situation)."




**Top** It looks messy inside, but it's generally quite a simple setup once you break it all down

**Above** The most interesting night light you've ever used: maybe it will be this decade's lava lamp?

The whole thing is configurable via specific files, and you don't need to recompile it each time either. The code is still being improved, and you can find it on GitHub (**bit.ly/1T8VKtC**), or, if you really like the project, you can look at getting one of the LifeBox kits by checking out the crowdfunding page at: **lifebox.ferranfabregas.info**.

## BRING YOUR RASPBERRY PI TO LIFE

### >STEP-01
**Set the parameters**

Delve into the config files and define how your life forms and mana work: life expectancy, energy requirements, reproductive rates, and so on.

### >STEP-02
**Start the experiment**

Power on the LifeBox and the simulation begins. The simulation will run for as long as you supply it with power, and any resets will start it from the beginning.

### >STEP-03
**A new beginning**

Once you've completed the experiment, it's time to start again. Go back into the parameters and reprogram the life forms and mana.

**ROBERT DOERR**

Robert Doerr runs Robot Workshop and has built robots that compete in *BattleBots* (the US version of *Robot Wars*).
**robotworkshop.com/robotweb**




## Quick Facts

- PiPlateBot took two weeks to design and build
- Turtle robots were first used in the 1940s
- They were developed because few computers had monitors
- They're often controlled using a language called Logo
- They're named after *Alice in Wonderland*'s Mock Turtle

# PIPLATEBOT

Drawing inspiration from the turtle robots of old, Robert Doerr created a Raspberry Pi robot with the components hidden inside a Bud Pi Plate case. Say hello to PiPlateBot...

Robert Doeer is no stranger to building robots. He's the owner of Robot Workshop, an organisation dedicated to restoring classic robots, and has entered two robots into *BattleBots* (the US version of *Robot Wars*): Crash Test Dummy and Crash Test Junior.

But this is no battle droid. When Robert saw a Bud Pi Plate case (**budind.com**), he was struck by how similar it looked to the turtle-style robots used to train computer science students. It was "definitely [like] the early turtle robots like the Terrapin Turtle robots and the early Tasman Turtle robots," says Robert.

The Pi Plate's circular design enables easy access (you twist off the top) and there is space inside for additional components. Robert immediately decided to see if he could place a Raspberry Pi, along with all the parts required to build a moving robot, inside the case. "It is the only Raspberry Pi-based robot that I know of built using an off-the-shelf Raspberry Pi case," he claims. "I tried to use as many Raspberry Pi-type products in the construction as I could."

"Getting everything to fit was the biggest hurdle," says Robert. He cut two rectangular holes in the base of the Pi Plate enclosure, and glued servos to the bottom of the case. An EZO power bank sits on top of the Raspberry Pi and RoboPi boards, and works as a battery. Finally, a USB Wi-Fi adaptor enables wireless communication via SSH. "Those that have seen it really like the robot and ask where I bought it," Robert says with pride. "The completed robot really looks like a finished project, so they assume it may be sold in stores."

The powerful RoboPi board (**mikronauts.com**) is an important component. "The Raspberry Pi is great at the high-level thinking, while the Parallax Propeller chip on the RoboPi board is a great I/O controller for offloading all the real-time tasks."

**Above** The finished product is a friendly-looking turtle-style robot that can sense objects in front of it

> If you have an idea for a project, just go for it! You'll never get it done just thinking about it

"I have been programming the robot in C," Robert tells us. "The RoboPi controller has libraries available for both C and Python. Eventually it would be fun to write a Logo interpreter so it could also use Logo and emulate the early turtle robots."

Finally, Robert added an HC-SR04 sonar sensor (ultrasonic transducer) so the PiPlateBot could measure objects directly in front of it. "[It] was built over the course of a couple weeks during my spare time in the evenings," he says. "It could have been built in a few evenings, but I had to wait for some parts to come in." There is still a bit of room left on the robot for additional sensors like contact bump, IR line following, or even a I²C compass.

Having created, and rescued, countless robots, Robert has good advice for budding robot builders: "If you have an idea for a project, just go for it! You'll never get it done just thinking about it. Even if the first iteration doesn't work out, you can always change it along the way and you'll learn a lot as you go."


**Above** A standard smartphone battery pack is squeezed on top of the boards and inside the case

## BUILDING THE PIPLATEBOT

### >STEP-01
**Cutting and mounting**

Holes cut in the base allow the wheels to fit, and glue sets the servos in place. Furniture gliders attached to the front and rear stop PiPlateBot from wobbling.

### >STEP-02
**Assembling the components**

The Raspberry Pi is fitted and a USB Wi-Fi adaptor is connected. The RoboPi board is stacked on top of the Raspberry Pi board. A smartphone charger provides power.

### >STEP-03
**Power and motion**

The RoboPi comes with libraries for both C and Python that are used to control the servos. An HC-SR04 sonar sensor is fitted: this enables PiPlateBot to sense its surroundings.

**GRAHAM GELDING**

Graham is a software developer, working on Linux to make graphical interfaces with Qt. In his spare time he has a number of hobbies, including learning to be a blacksmith.
**instructables.com/id/Coffee-Table-Pi**




# COFFEE TABLE PI

## Quick Facts

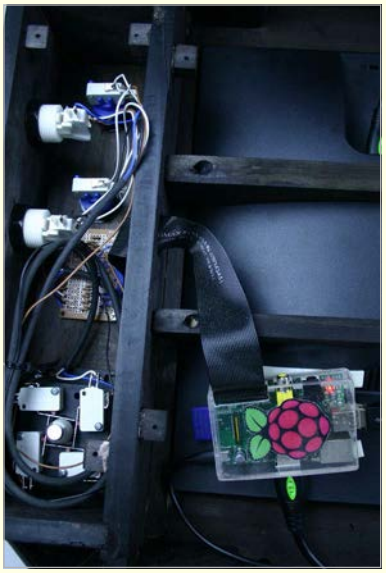
- The project took a few weeks to complete
- The Perspex over the screen can get scratched easily by kids
- *Donkey Kong* is Graham's favourite game to play on it
- It does actually get used for resting cups of coffee on
- There will be no kit, so use the Instructables guide

This fully fledged cocktail arcade cabinet, apparently masquerading as a coffee table, is one of those Pi projects everyone wants to do...

As you probably know, the Raspberry Pi, while an excellent educational tool, is big among the maker community. The kind of folks who like to create machines are always on the lookout for a tiny computer controller, one that can power their project with the smallest footprint. Along with the makers, any tiny computer released to the market also tends to attract a great deal of attention from the arcade gaming community, looking for the next thing to power their work-in-progress MAME cabinet. With this cross-section of interests for one device, it's a wonder why we haven't seen full-size arcade cabinets powered by a Raspberry Pi in every single issue of the magazine. Graham Gelding is one of the few to take on this task. Not only that, he's gone a step further and created the ultimate in classy, grown-up arcade gaming apparatus: a cocktail arcade cabinet (although it has a slight twist, as he likes to refer to it as his 'coffee table').

"It was an attempt to recreate the classic arcade cocktail cabinet," Graham tells us, "but in a way that can fit into a lounge room. It's also a way of introducing my kids to the games that I had growing up.

"I also wanted to try some woodworking and needed a project for my Raspberry Pi. The idea of

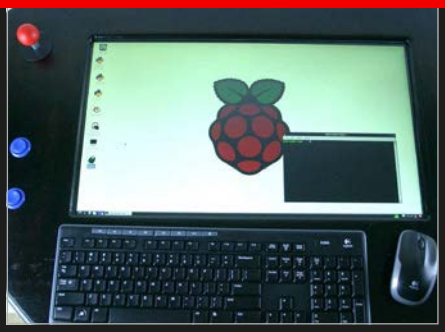


**Above** The insides are neatly arranged and very chunky, just like any good arcade machine should be

## USING GRAHAM'S COCKTAIL CABINET

### >STEP-01
**Table or arcade machine**

When the coffee table is off… it's still a table. To turn it into its true arcade machine form, you need to switch it on.



### >STEP-02
**Select your game**

Like a lot of Pi emulation software, you're met with a selection of pre-loaded games that you've added yourself. Selecting a game will launch it.


### >STEP-03
**King of Kong**

Play your game and have fun! Try not to put your coffee cup directly on top of the screen, and try not to knock it off while screaming at Mario.

mixing the two projects seemed perfect. I could make use of my experience with Linux, but also learn about woodworking."

He's not joking about the woodwork either. While you might think it was just a table that had been modified, or an existing cocktail cabinet that had been gutted, Graham made the whole table from scratch using pine from an old bookshelf, as well as installing the screen, arcade controls, and the Raspberry Pi itself that powers it. This made the most expensive parts of the project the LCD screen and some of the controls.

As well as those main components, Graham gutted some old PC speakers for their drivers and transformers, installed a sheet of clear Perspex over the screen to protect it, and did a lot of custom wiring and powering.

As for the future of the project, Graham is just happy with what he's made. However, he has plans for his next Raspberry Pi thing, telling us: "I would like to do something combining the Pi with the Oculus Rift virtual-reality headset."

The headset is finally coming to the consumer market, so marrying the two, if possible, would be a big leap in Raspberry Pi projects.

> "It was an attempt to recreate the classic arcade cocktail cabinet, but in a way that can fit into a lounge room…"

**Above** The table is set up for games that use a vertical screen, so most arcade games during and before the 1980s

# SUBSCRIBE TODAY!

Subscribe to the print edition to be among the first to receive the magazine. Issue 36 will be available to buy in-store & online 30 July, priced £5.99

## 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 PI**
AVAILABLE FROM **30TH JULY!**

**SAVE UP TO 25%**

# 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
- Print the form on this page

**Or subscribe on your tablet or phone:**

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#35

**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**

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)

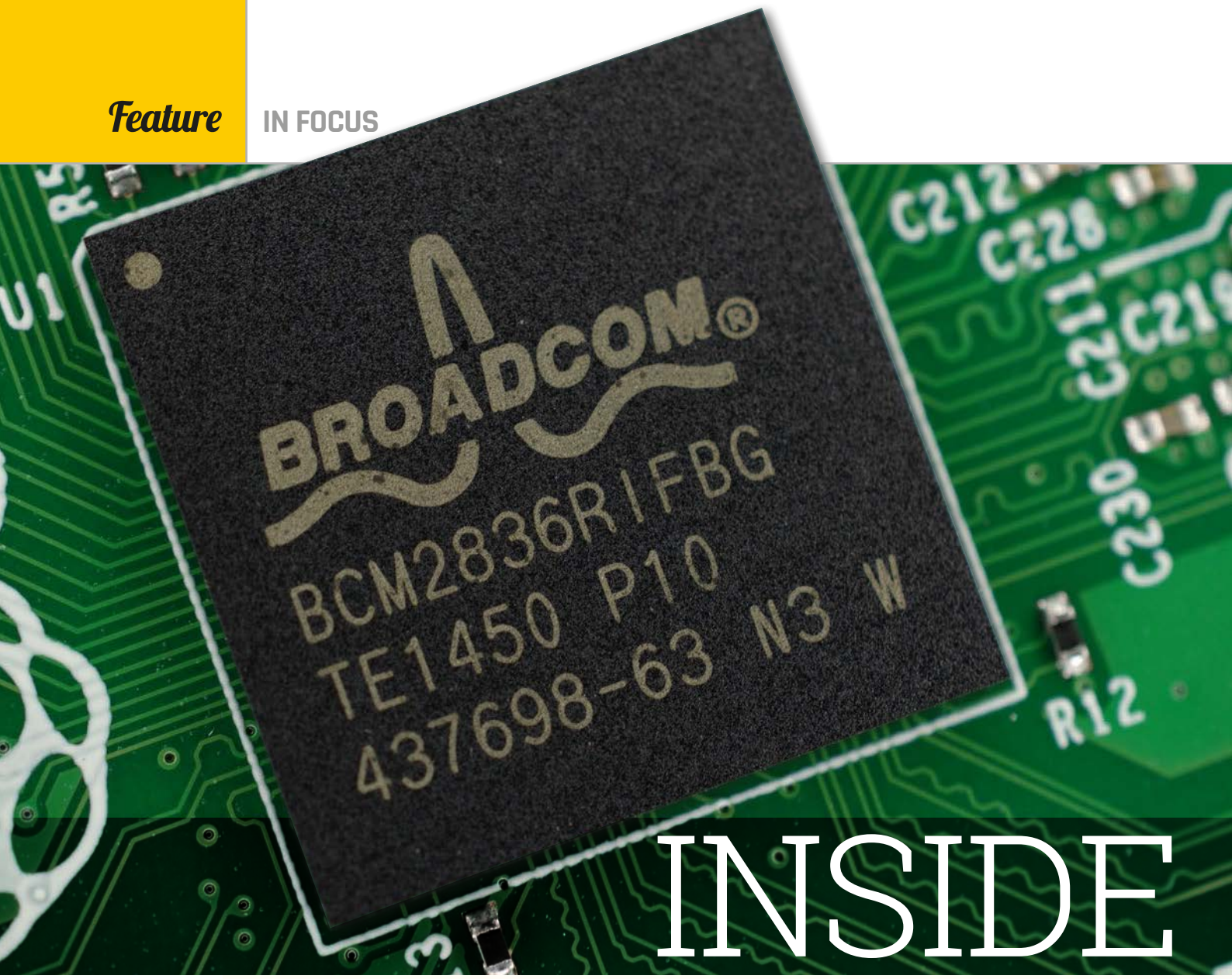
**RETURN THIS FORM TO:**

*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.




# INSIDE VIDEOCORE

Above The VideoCore IV has always been the star of the Raspberry Pi, providing the credit-card-sized PC with its excellent video and graphical capabilities

## The GPU in the Raspberry Pi is remarkably powerful and energy-efficient. **Tim Anderson** talks to its creators...

"This is for me probably the finest bit of engineering I've ever been involved in," says Eben Upton, founder of the Raspberry Pi Foundation. He is talking about VideoCore IV, the Broadcom GPU (graphics processing unit) in both the original Raspberry Pi and the newer Raspberry Pi 2.

We chatted to Eben, along with director of software Gordon Hollingworth and director of hardware engineering James Adams, about the history of VideoCore, what makes it so impressive, and how you can take advantage of it in your own projects.

The origins of VideoCore go back to a Cambridge-based firm called Alphamosaic. "The company was spun out of Cambridge Consultants in 2000. The guys who did the very first Orange videophone figured there was a market for a dedicated multimedia chip to do the video compression. VideoCore I was basically a two-dimensional DSP [digital signal processing] engine plus SRAM plus peripherals," recalls James Adams, who worked on the project at Alphamosaic.

SRAM (static RAM) is memory that keeps data without having to be constantly refreshed, and is faster than dynamic RAM, though it also uses more power and takes more space. VideoCore I was used in a number of mobile phones.

The firm went on to develop VideoCore II. "It was a refinement with more power, a bit more memory, fixed all the issues, and dual issue on the scalar side," says James. 'Dual issue' is the ability to execute two instructions per cycle. "It won the slot in a popular video media player of the time."

Whereas VideoCore I targeted CIF video (352×288 pixels), VideoCore II could extend to VGA (640×480).

Eben Upton did not work on VideoCore until later, but says that even these early versions had a cool architecture. "You have a 64×64 register file and a fairly conventional scalar CPU that's got its own register file. But there is also this vector ALU (arithmetic logic unit) that is 16-way and can access strips of pixels anywhere in the 64×64 register file. It turns out this is incredibly useful for doing the kinds of operations you need to do for video, such as lots of the transform operations.

"On the front end you have the instruction fetch that then issues instructions for one of those two paths (scalar ALU or vector ALU). That turns out to be fantastically powerful." Parts of this design still live on in the current VideoCore.

Alphamosaic was acquired by Broadcom in 2004, when VideoCore II was beginning production and the team was getting started on VideoCore III, which was to support 720p resolution (1280×720).

"It was funny because at the time, I'm not convinced that most of the team believed in 720p on a phone. Everyone goes, I know we're doing it but why? Why would you want to plug this into a TV? It's a phone," recalls Gordon Hollingworth.

The design of VideoCore III was influenced by what people were attempting to do with VideoCore I and II. "People were trying to do composited display output, stuff on the camera, decode and encode video, and even try and do some 3D graphics," Eben tells us.

VideoCore III also saw the move to dedicated video hardware. "For higher resolutions you need a lot more RAM, so the SRAM was chucked. It's a more conventional architecture with a lot of dedicated hardware blocks for 3D, video, camera processing, and scale display," explains James.

On the table is a prototype board for VideoCore III, dating from 2005, and saved from the skip because it bears the number 3.14.

"I saw this on the day I came for interview [at Broadcom]", remembers Eben. "This was then a fixture of my life for over a year until we got the silicon back."

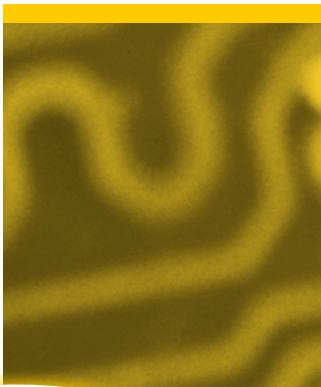
James led the 3D graphics team. "It's a nice plug-in block, it's got an interface, and it's just a black box to the rest of the chip," he says, though it does use the 2D vector processor to do shader processing, "which actually worked surprisingly well."

> "The origins of VideoCore go back to a Cambridge-based firm called Alphamosaic"

Eben describes the 3D processing in VideoCore III as like a train track. "We have a buffer here, and a vector register file, and then four shadow vector register files which were 4K blocks, and we have a 3D pipeline. The 3D pipeline would prepare some pixels or some vertices and would deposit them, then push the shadow VRFs off round the train track. Then the VPU (vector processing unit) would sit there waiting to be signalled that something had arrived, it would read the vertex data, perform the vertex or pixel shading operation, and it would then write the shaded vertices or the colour back. The shadow VRFs would then amble off around the train track a bit more and then end up back in the 3D pipeline, so the 3D pipeline is both a front-end and a back-end." In its back-end role, the 3D pipeline pushes the pixels into the frame buffer. "That was a really nice architecture," says Eben. "You were able to reuse quite a lot of logic over here which you could then use for other stuff, when you weren't doing 3D."

Another challenge in VideoCore III was to adapt it for general-purpose use. In earlier devices like media players, the chip would be used in one of several modes, such as to play audio or video, and you could optimise accordingly. When you have to support a general-purpose operating system, the usage is unpredictable and contention, where the chip is asked to do several different things simultaneously, is more likely. "What you end up with is a much more conventional architecture," says Eben.

The work on VideoCore III resulted in support for H.264,MPEG2, MPEG4, and VC1 at 720p, plus the ability to encode video in MPEG4 or H.264.



**Below** The operation of the vector ALUs

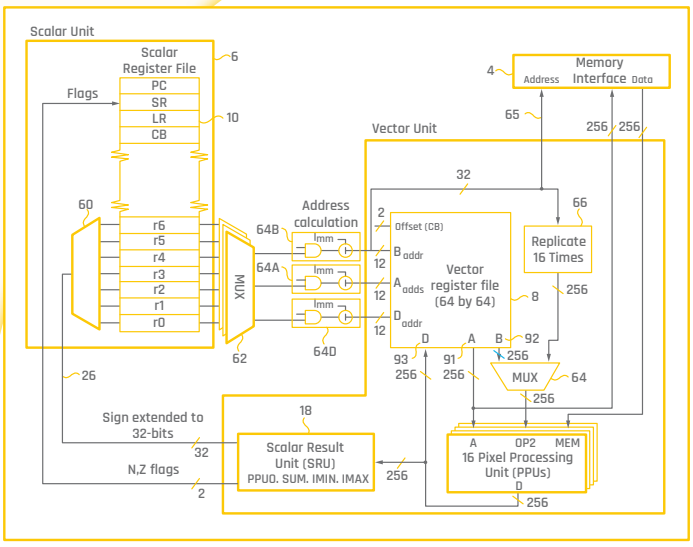



FIG. 4

## Power optimisation – a special ingredient

There was also a huge focus on power optimisation. "It's one of the really special ingredients," says James.

What goes into power optimisation? The high-level architecture needs to avoid energy-expensive operations, while at the micro-architectural level, the design has to be amenable to things like clock-gating, where parts of the circuit are disabled to save power. There is also work at the tools level, the tools being the software programmed into the ASIC (application-specific integrated circuit) that drives the system. "If you've wrapped up some opportunities in your micro-architecture itself to do all sorts of clever stuff, you need to then drive the tools properly to make sure they actually exploit those opportunities," explains Eben.

> "What's next? 'There is a VideoCore V,' says Eben. 'It's freaking awesome'"

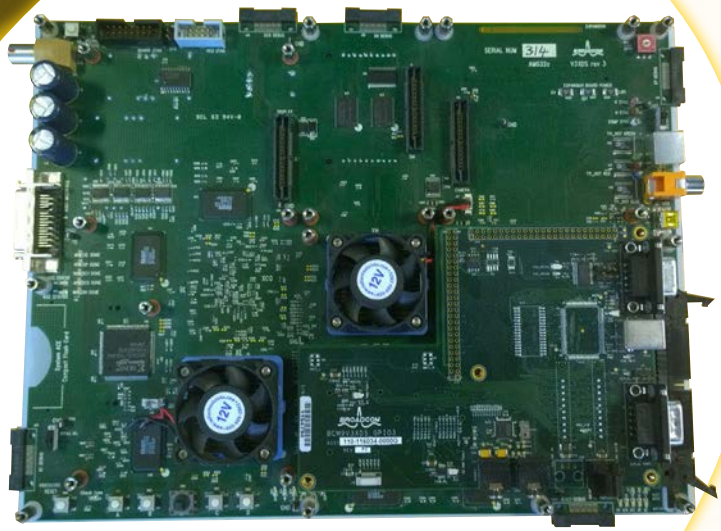
Imagine the chip gets an easy video clip to decode. Should you turn the voltage down to run the core slower, or run it fast and then turn it off so it is only running half the time? "The balance between those two is a non-obvious calculation," advises Gordon Hollingworth.

You might think that power optimisation is more important in a battery-powered device like a smartphone than in something like the Pi, which is more often mains-powered, but in fact it is important in all scenarios. "Power dissipation is about more than the actual power usage," says Gordon. "It has all got to go somewhere and results in the same thing, which is heat."

VideoCore III went into production in 2007. "The first tapeout was June 4th," recalls Eben. "It was going to tapeout in the first half of the year, even if we had to add more days."

'Tapeout' is the term for when the final circuit of a chip is sent for manufacture.

A little-known fact is that Eben built a prototype Raspberry Pi based on VideoCore III. "I did build a thing, but it never went into a shipping product," he says. "I had a port of Python to VideoCore III and it was the first thing I showed to people. It had a PS2 keyboard interface on a bit of Veroboard. So Raspberry Pi was actually founded as an organisation with a view to building something on VideoCore III. But because it's a proprietary closed architecture, you end up building everything yourself. It could have made it into a product, but it wasn't the thing."



**Below** A prototype board for VideoCore III, appropriately numbered 3.14

## Introducing VideoCore IV

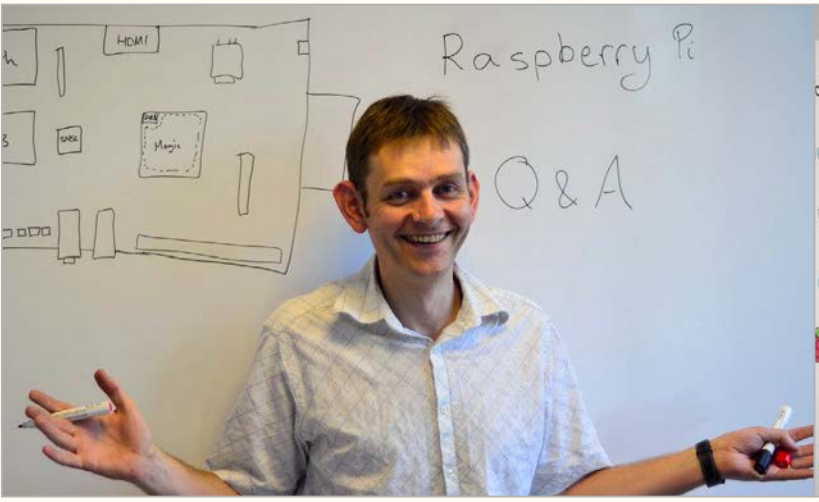
The thing, rather, was VideoCore IV, which is when the chip moved from an immediate mode to a tiled mode architecture, where the image is rendered in a series of small tiles rather than as a single large image. "James had been kicking round this idea of building a tile-mode architecture for fun. And then we just proposed it," recalls Eben. "I was massively surprised when we were allowed to do [it]; we were just cut an enormous amount of slack."

"What happened with III to IV was that the 3D team effectively went off on their own, got more resources, were allowed to rip up most of what we'd done for the current 3D and restart," says James.

The advantage of tile mode is less memory traffic, which translates to less energy. "You trade up-front work in figuring out which bits of the screen a triangle will cover for less overall energy," says Eben.

Another part of this change was the introduction of a processor called the QPU (quad processor unit), a term which is unique to Broadcom. It is called 'quad' for two reasons. "One, because the hardware is four-way parallel, and two, because we have this notion of a quad, four adjacent pixels," explains Eben. "You can't really make a 3D core that pushes single pixels through because you can't get derivative information to select mipmaps."

The QPU then is a lightweight processor for doing shading, though the team gave it unusual flexibility. "Your conventional shader processor is a very siloed architecture, so you've got say 16 SIMD [single instruction, multiple data] channels and they proceed along in stately isolation from each other. A lot of what made VideoCore was the ability to get data between channels. You don't need that for 3D, but we wanted to keep some element of that. For example, there is a rotate capability so you can take one of your 16 vectors which you've read out of the register file and you can rotate it by a certain amount;





Above left Gordon Hollingworth is the director of software for Raspberry Pi

Above Eben Upton speaking at the launch of the Raspberry Pi 2

that turns out to be totally useless for 3D and super-useful for pretty much everything else.

"We also ended up with a thing which we called the VPM, the Vertex and Primitive Memory, which all of the QPUs can access, so when you need it you have some transposition capability."

Eben wrote the QPU proposal at the end of 2007. The Broadcom team spent 2008 writing VideoCore IV, with the first samples arriving in 2009. VideoCore IV supports 1080p (1920×1080) encode and decode along with OpenGL ES 2.0, the embedded system version of the popular 3D graphics API.

Work is also under way to support H.265, the next-generation video compression standard, also known as High Efficiency Video Coding. "We've got somebody working on H.265 at the moment and he is in the process of trying to get 1080p 24 H.265 support up and running," discloses Gordon.

"If you look at Raspberry Pi, the performance is pretty damn good. That's a chip we taped down in 2009. It's crushingly good performance," says Eben.

"It's also the first unified shader architecture in mobile. In fact, it beat some of the PC guys," adds James. "It's a well-understood architecture, it's power-efficient, and it's now very well supported. One of the great things about Pi is how stable the software is. We like our architecture and we also like the form factor of the Pi."

## Opening up VideoCore

Another key feature of VideoCore IV is how open it is. "One of the wonderful things is that Broadcom released the docs," says Eben. "That is completely unheard of. Mali [ARM], you can't get the docs. Imagination Technology, you can't get the docs. Adreno [Qualcomm], you can't get the docs. Nvidia has a nod in this direction. In the mobile world, nobody releases the docs and nobody releases the driver sources. In February 2014, Broadcom released the full architecture docs and a full driver implementation for one of the other chips that uses VideoCore."

The extent of the Pi's openness is increasing. In June 2014, as a result of Broadcom's publication of VideoCore documentation, a Linux graphics driver developer called Eric Anholt joined the team, coming from Intel, where he had been working on Intel's support for the Mesa open-source graphics stack. He is now working on an MIT-licensed (a permissive free software licence) Mesa and kernel DRM (direct rendering manager) driver for the Raspberry Pi. This will bypass most of the code which currently runs in VideoCore firmware as a closed-source 'blob' which is called by software drivers.

"This will be the first driver for a mobile-side modern PC graphics architecture which has been developed by open specification," reveals Eben. "I expect we will ship at least a technical preview this year."

What's next? "There is a VideoCore V," says Eben, who will not be pressed on further details. "It's freaking awesome."

## BEYOND VIDEO

What can you do with VideoCore IV? It is key to the appeal of the Raspberry Pi, especially in Raspberry Pi 2 with its faster CPU and increased RAM, bringing more balance to the capabilities of the GPU and CPU sides. Aside from the ability to use the Pi as a PC with surprisingly capable graphics, VideoCore enables media-centre and camera applications.

What about if you are using the Raspberry Pi as a headless board, or want to exploit the GPU to accelerate non-graphical applications? Eben Upton is not an enthusiast for OpenCL, an open-source API for using the GPU for general-purpose computing, rather than just to drive a display.

"I'm a massive OpenCL skeptic," reveals Eben Upton. He points out that you can do image processing, a common use case for OpenCL, with OpenGL instead. While there are examples of other uses, they are niche, he argues. "There are people using it for finance. But where's the consumer killer app for this stuff? It's image processing, right? You could do image processing in a much less expressive language."

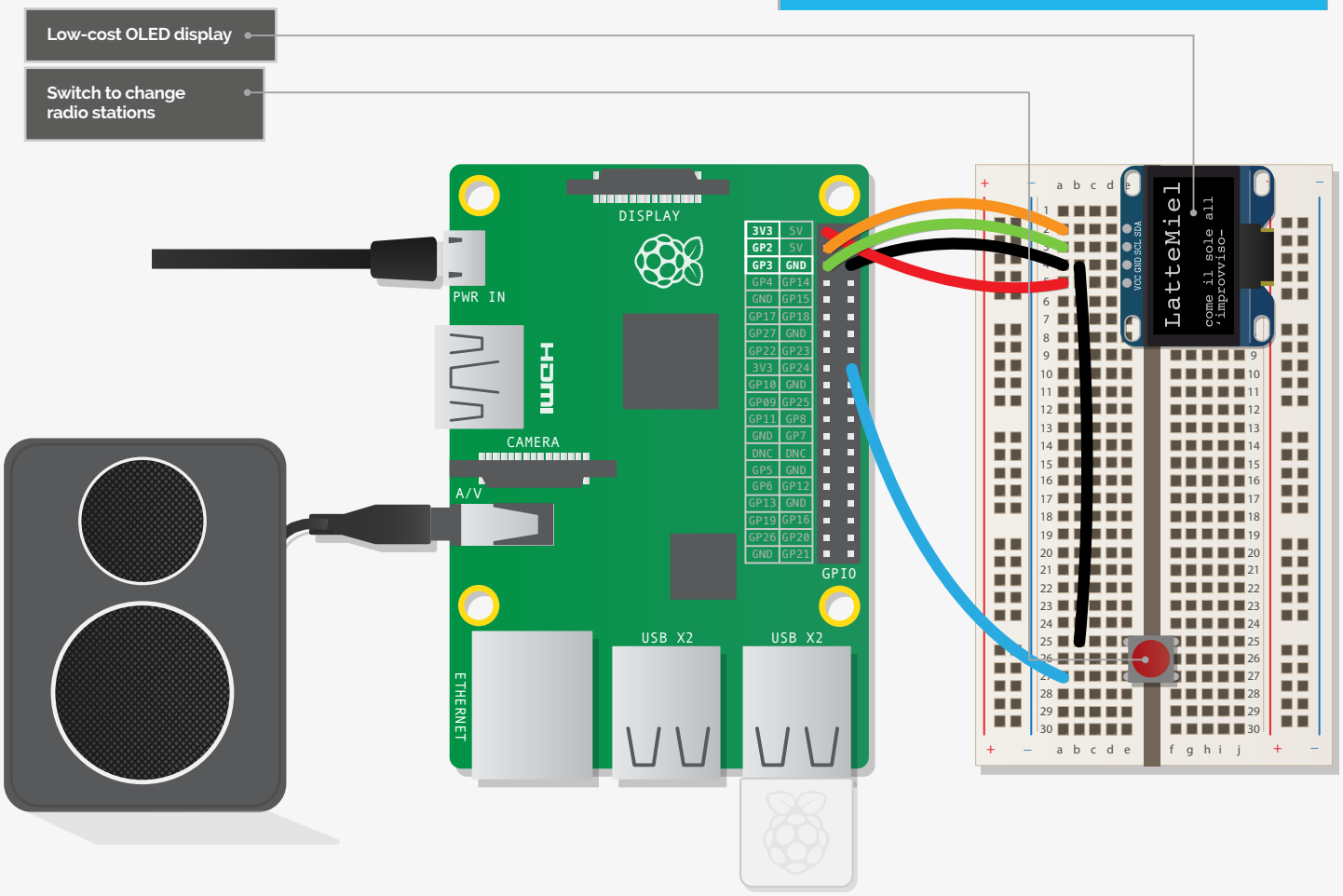
That said, it is possible to use VideoCore for general-purpose computing. It is ideal for fast Fourier transforms (FFT), for example, but you have to code in assembly language as well as understanding VideoCore at a low level. "We've helped out people who email us and say how do I do this, or ask about a strange feature. We're keen to help people with that. Is it a mainstream thing? Not unless we come up with some sort of API," says James Adams.

# EVERYDAY ENGINEERING PART 5

**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**




## You'll Need

- Half-size breadboard
- 128×64 OLED 0.9-inch (eBay)
- Tactile push switch
- 5 male-to-female jumper wires
- 1 male-to-male jumper wire
- Powered speakers or headphones
- Short audio lead to suit speakers
- USB Wi-Fi adaptor (optional)

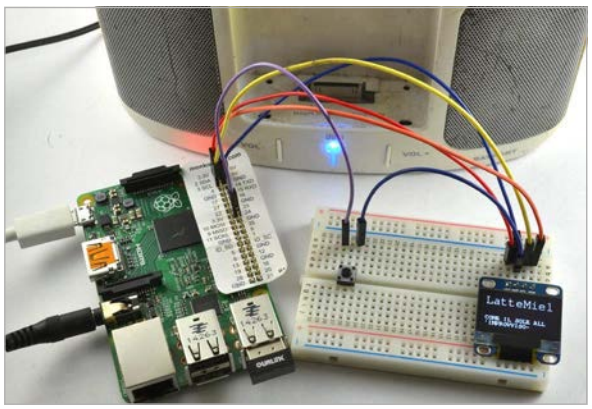
# INTERNET RADIO

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

Gone are the days when if you wanted to listen to the radio, you had a choice of maybe the dozen or so radio stations that were being broadcast from an FM transmitter on a hill near you. These days, you can listen to almost every radio station in the world over the internet.

In this project, we will show you how to turn your Raspberry Pi into a simple no-frills internet radio using a small OLED display to tell you the station you are 'tuned to' and, if the station provides it, the track that is currently playing.

A pair of powered speakers (in our case, a redundant iPhone dock with audio-in jack) or headphones provide the sound.



Above Your very own no-frills world radio

As you'll see from the list of required components on the previous page, this project uses a breadboard to provide a solid base for the display and the push switch that is used to change channels.

The OLED display we used came from eBay. Look for one that has just four pins, with the same pin names and order as shown on the breadboard diagram. Before wiring up the breadboard, do check that the pin names on the OLED are the same as shown here – we've seen some OLED displays where the VCC and GND pins are swapped over, which would be disastrous unless you change the breadboard wiring to match.

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 Pi will include the breadboard, jumper wires, and a switch.

## OLED displays

OLEDs (organic light-emitting diodes) are fantastic little bits of technology that offer high-resolution graphical displays in a small package. The display used in this project has 128×64 pixels in a screen that measures less than an inch diagonally.

These kinds of display are often coupled with a driver board containing an SSD1306 driver chip. This takes care of controlling the pixels on the display and allows you to connect it to a Raspberry Pi using a type of serial interface called I²C. This uses just two pins on the Raspberry Pi: SDA and SCL, which double as GPIO pins 2 and 3. You also need to supply the display with 3.3V.

## Making your internet radio

Using the type of speakers that take their power from USB, but have a separate audio input, would allow you to power the project from a single power supply to the Raspberry Pi. Having a USB WiFi dongle for your Pi will allow you to make the radio portable, rather than needing to plug it into your home router.

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 whole load of Python libraries, as well as the mpd (music player daemon) Linux software that runs a background process which does the actual music streaming. The display also requires the Pi to be set up to use I²C and various other pieces of software to be installed.

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

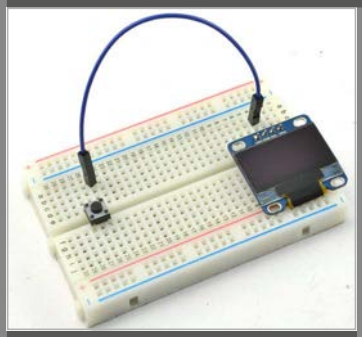
```
sudo apt-get update
```

The display uses the I²C interface on the Raspberry Pi. To enable this, run the raspi-config tool using:

```
sudo raspi-config
```

# 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

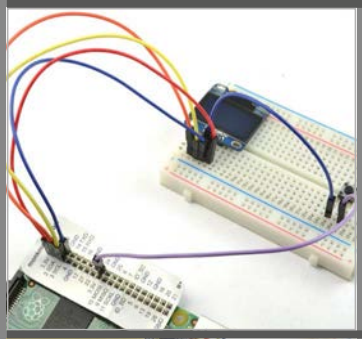


### >STEP-01

**Fit the display and switch onto the breadboard**

Fit the display and switch onto the breadboard in the positions shown. The switch will only fit the correct way around across the two banks of breadboard holes.

Link GND from the display 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 display pins and bottom switch terminal to the Raspberry Pi GPIO pins.

### >STEP-03

**Attach the speakers and WiFi adaptor**

Use a short audio lead to attach the speaker to the audio socket on the Raspberry Pi, or use headphones. Using a USB WiFi adaptor will keep the project more mobile. After plugging in the WiFi adaptor, you will need to configure your Raspberry Pi to use it. This guide will show you how: **raspberrypi.org/documentation/configuration/wireless/**

…and then choose 'Advanced' and then 'I2C' from the menu that appears. You will then be prompted with 'Would you like the ARM I2C interface to be enabled?' Say 'yes'. You will then be asked 'Would you like the I2C kernel module to be loaded by default?' Again, this is an option that you would like to use, so say 'yes' again. Select 'Finish' to exit raspi-config.

If these instructions don't tally with your system, we strongly recommend that you update to the latest version of Raspbian and then try again. This guide tells you how: **raspberrypi.org/documentation/raspbian/updating.md**.

> The Python code for this program is fairly short, but a little complex in places

Continuing with the display and I²C setup, there is some software to install, so run the following commands from your home directory:

```
sudo apt-get install python-smbus i2c-tools python-pip
git clone https://github.com/rm-hull/ssd1306.git
cd ssd1306
sudo python setup.py install
sudo pip install pillow
```

Before going any further, you can check that your Raspberry Pi can communicate with the display by issuing the following command:

```
sudo i2cdetect -y 1
```

The '3c' in the middle of the results is your display. If there are no entries in the table except dashes, then the display isn't connected correctly, or I²C is not set up.

Now that the Raspberry Pi is ready for the display, it's time to turn your attention to the software that you need for the radio, so run the commands:

```
sudo apt-get install mpd mpc
sudo apt-get install python-mpd
```

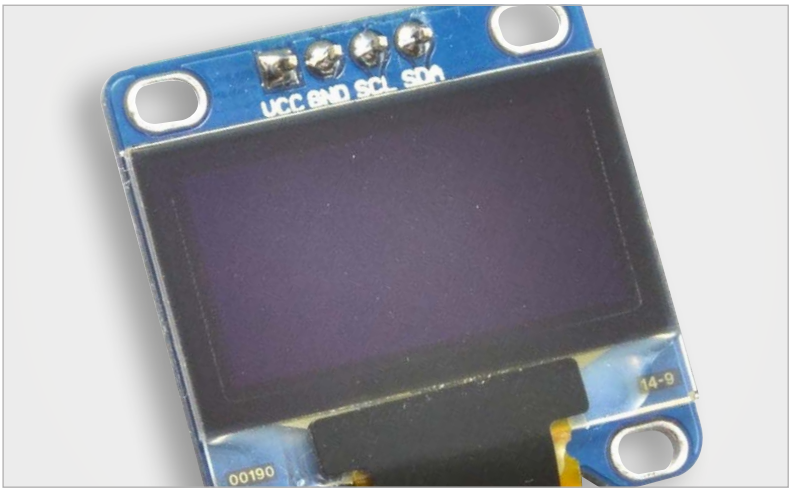
Finally, you can download the Python program for this project using the command:

```
git clone 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 our earlier articles, change directory to **pi_magazine** and run the following command to update your directory with this project (05_internet_radio).

```
git pull
```

Below OLED displays are cool




## How the code works

The Python code for this program is fairly short, but a little complex in places. Fortunately, most of the hard work of controlling the display and the music-playing software is taken care of in the libraries that you installed earlier.

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. The **oled** and **PIL** (Python Imaging Library) libraries are needed for the display. The **mpd** library provides a Python interface to the mpd music-playing software, and the **RPi.GPIO** library is needed so that we can tell when the switch is pressed. The **time** library is used for adding delays to the code, and the **textwrap** library helps format the text for the display.

The next section configures the GPIO pin 24 that is connected to the switch, and after this comes the code to set up the display. This first establishes a connection with the display using:

```
device = ssd1306(port=1, address=0x3C)
```

If your display showed a different address than '3c' when you ran `i2cdetect` earlier, then you will need to change the address here. Two fonts are also defined: one large font for the top line of text, and a smaller one for the second and third lines on the display.

The next section sets up mpc, connecting to the locally running web service that mpc uses to communicate. The `num_stations` global variable will be used to hold the number of stations stored in the playlist; the `current_station` global variable contains the index of the current station.

The `display_info` function displays whatever station information is available. This information is not always present for a station and can also change as the station starts playing the next track. Ideally, it will provide both the station name and the title of the track that is currently playing. This uncertainty about what will be available means that the parts of the code that get the station name and track title are both surrounded by `try: except:` blocks that trap any errors when the information is retrieved and formatted onto three lines for the display.

`display_info` uses the function `display_message` to actually write text onto the display using the two different font sizes.

The main program loop is contained in a `try: finally:` block, so that the GPIO pins and connection to the mpc will be tidied up when the program exits. Inside the main loop, the number of stations is found. Checking the number of stations inside the loop means that if you add new stations, they will be added to

the list automatically, without having to restart the program. If there are no stations, then an appropriate error message is displayed.

If the switch button is pressed, then 1 is added to **current_station** and this variable is then checked to see if it's got to the end of the station list. If it has, **current_station** is set back to 0. Finally, mpc is then instructed to play **current_station**.

The **finally** clause is used to set the GPIO pins back to inputs when the program is closed using **CTRL+C**.

## Using your internet radio

Before running the program, you will need to add some radio stations to listen to. Finding correct URLs for internet radio stations that work is a lot more time-consuming than it should be. The BBC, for instance, is notorious for changing its URLs.

You add stations from the Linux command line using the mpc software. To get you started with a few stations, run the following commands:

```
mpc add http://stream-sd.radioparadise.com:8056
mpc add http://mp3.live.tv-radio.com/fip/all/fip-32k.mp3
mpc add http://bbcwssc.ic.llnwd.net/stream/bbcwssc_mp1_ws-einws
```

We found this site quite useful for finding station URLs: **dronkert.net/rpi/radio.html**.

There are lots of other listing websites out there, but be prepared for many of the stations to have the wrong URL. If you end up with a URL that won't play, then you will see a message like '5 ?' on the top line of your internet radio's screen. The '?' means there is a problem, while the digit is the number of the station with an issue.

To remove the station, first press the button on the internet radio to select a good channel and then send the following command from a terminal window:

```
mpc del 5
```

Start the internet radio program with the command:

```
sudo python internet_radio.py
```

The display should tell you the station and current track title (if the station supports that).

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

```
sudo chmod +x internet_radio.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/05_internet_radio/internet_radio.py &
```

Restart your Raspberry Pi and this time the internet radio should start up automatically.

# internet_radio.py

**Language**
>PYTHON

**DOWNLOAD:**
github.com/simonmonk/pi_magazine/tree/master/05_internet_radio

```
from oled.device import ssd1306, sh1106
from oled.render import canvas
from PIL import ImageFont
from mpd import MPDClient
import RPi.GPIO as GPIO
import time, textwrap

# Configure the GPIO pins
BUTTON_PIN = 24
GPIO.setmode(GPIO.BCM)
GPIO.setup(BUTTON_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Set up display
device = ssd1306(port=1, address=0x3C)
small_font = ImageFont.truetype('FreeMono.ttf', 12)
large_font = ImageFont.truetype('FreeMono.ttf', 24)

# Set up MPC
mpc = MPDClient()
mpc.connect("localhost", 6600)
num_stations = 0

current_station = 0 # index of the current station

# Display whatever information is available- with luck, the
# station name and the song title
def display_info():
    station = mpc.currentsong()
    try:
        station_name = station['name'][0:9]
    except:
        station_name = str(current_station + 1) + ' ?'
    try:
        titles = textwrap.wrap(station['title'], 19)
        titles += [''] # so there are always 2
    except:
        titles = ['', '']
    display_message(station_name, titles[0], titles[1])

# Display a message on 3 lines, first line in a large font
def display_message(top_line, line_2, line_3=''):
    global device
    with canvas(device) as draw:
        draw.text((0, 0),  top_line, font=large_font, fill=255)
        draw.text((0, 40),  line_2, font=small_font, fill=255)
        draw.text((0, 50),  line_3, font=small_font, fill=255)

try:
    mpc.play(current_station)
    while True:
        num_stations = int(mpc.status()['playlistlength'])
        if num_stations == 0:
            display_message('Error', 'add stations',
'mpc add {url}')
        elif GPIO.input(BUTTON_PIN) == False:
            current_station += 1
            if current_station == num_stations:
                current_station = 0
            mpc.play(current_station)
            time.sleep(0.2) # key debounce
        else:
            display_info()
        time.sleep(0.1)
finally:
    GPIO.cleanup()
    mpc.close()
    mpc.disconnect()
```

**NEXT MONTH**

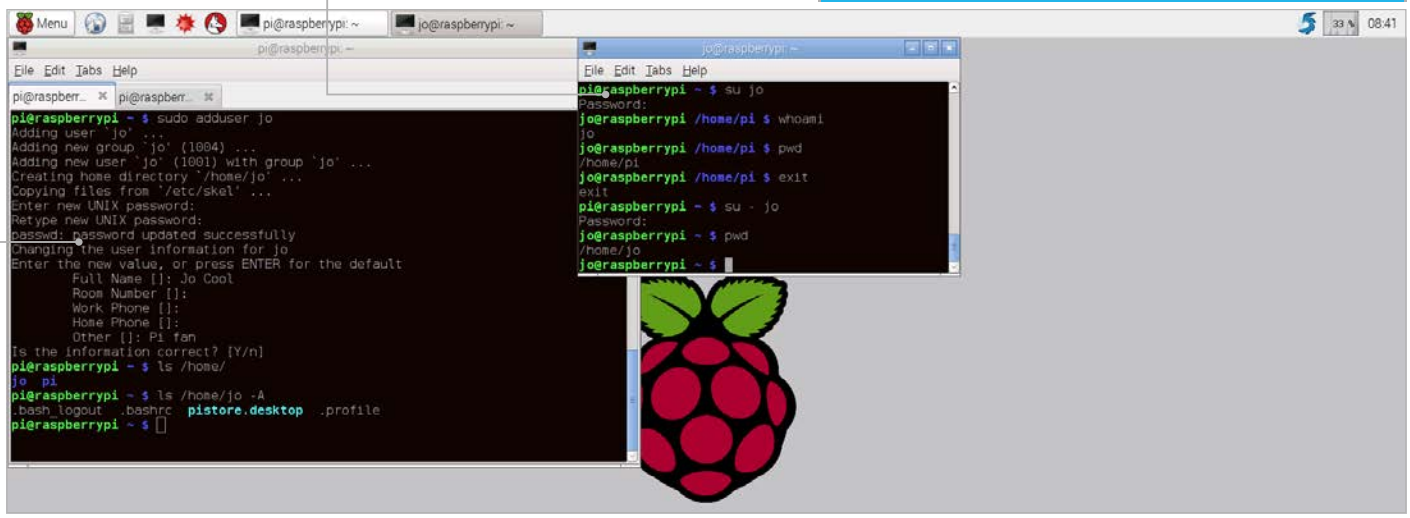
In the next project in this series, we'll create a mood light using a length of RGB LED strip.

**RICHARD SMEDLEY**

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

Share your Pi: make new user accounts and others can log in or switch users from a command-line session

The command-line 'environment' is personal to each user. You can change your identity with or without a change of environment, depending upon what you need to do in another role




## 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.

# COMMAND LINE PI PART 5: CUSTOMISE THE COMMAND LINE

**Richard Smedley** presents your cut-out-and-keep guide to using the command line on the Raspberry Pi. In part 5, we make Raspbian a little more personal as we get it to behave and look just the way you want it

Take a look at that blinking cursor on your terminal, and what's behind it:

```
pi@raspberrypi ~ $
```

The $ is known as the 'dollar prompt', awaiting your command; before it you see the **~** (tilde), shorthand for 'home' – which is **/home/pi** in this case. Before that is [user name]@[computer name], in the form **pi@raspberrypi**. Not only is this informative (at least if you've forgotten who and where you are), but it's something you can change and personalise.

## BASIC ACCOUNT

**adduser** creates a new user, then takes care of all of the extra details like making a home directory. If all you want is a user created with no extra frills, then the command you want is **useradd**.

## New user

Let's start with that user name – pi. If more than one person in your family uses the Pi, you may want to keep the pi user for shared projects, but set up individual login accounts for family members, including yourself. Creating a new user in Raspbian is easy:

```
sudo adduser jo
```

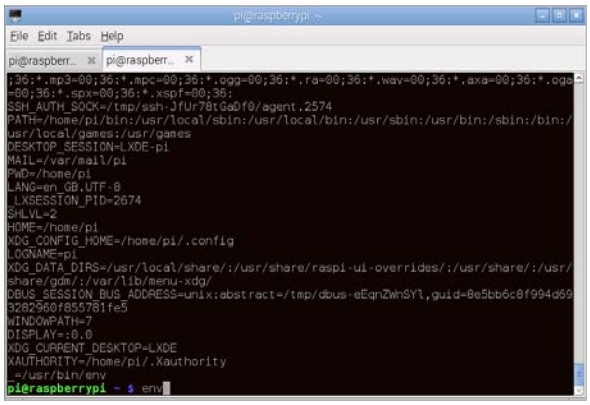
…will create a new user account named **jo**. You will be prompted for a password (pick a good one) and lots of irrelevant info (dating back to shared university computers of the 1970s) that you can safely ignore by just pressing **ENTER** at each prompt. Now we have a user account for jo, have a look at **/home/jo**. Looks empty? Use **`ls -A`**. Jo has never logged into the computer, so you will see the absence of most of the contents of /home/pi for now, such as **~/.gconf**, but there is a **.bashrc** and a couple of other config files.

Not every user has a home directory and logs in:

```
cat /etc/passwd
```

…and you'll see a lot of users listed that aren't people. This is because files and programs running on a Unix-type system have to belong to a user (and a group – take a look at **/etc/group**), as we saw back in part 1 when we did **`ls -l`**. The user passwords are fortunately not listed in the **/etc/passwd** file in plain text, so if you want to change a password you'll need to use the **passwd** command: **`sudo passwd jo`** will change the password for user jo. If you're logged in as user pi, then simply calling **passwd** will prompt you to change pi's password.

Transformations in the virtual world are always easier than those in nature, and this is the case with switching from being 'pi' to 'jo' – we use the change




**Above** Bash stores information, from your previous 'present working directory' to who you are, in environmental variables like OLDPWD and USER. See individual variables with e.g. `echo $USER`, or view them all with `env`

(or substitute) user command, **su**, like so: **su jo**. After typing this, you should see the prompt change to **jo@raspberry** – you can also confirm whom you are logged in as with **whoami**.

## Changing identity

**su - jo** (note the dash) is usually preferred, as you'll gain all of jo's specific environment settings, including placing you in **/home/jo.** Note that on many other Linux systems, **su** on its own will enable you to become the root or superuser, with absolute powers (permissions to run, edit, or delete anything). Raspbian (and some other popular GNU/Linux systems like Ubuntu) prefer **sudo** to run individual programs with root permissions. Root's godlike powers may be temporarily attained with **sudo -s** – try it and note how the prompt changes – but it's generally a bad idea to run with more permissions than you need, for the same reason it's a bad idea to run with scissors!

For any user, you can customise elements of their command-line use most simply by editing **~/.bashrc**. Take a look through that configuration file now: **more ~/.bashrc**. Note a number of variables in all capital letters, such as **PATH**, **HISTSIZE**, and **PS1**. The last of these controls the prompt you see – currently **jo@raspberry ~ $**. To change it (for the duration of your current terminal session), try something like:

```
export PS1="tutorial@magpi > "
```

This is a temporary change: type **exit** and you've left the **su** value of **jo**, so you'll see **pi@raspberry ~ $** once more. If you **su** back to jo, the magpi prompt will still be gone. To make your change permanent, you need to put the **PS1** value you want into **~/.bashrc**. A search around the web will bring up many fancy options for better customising the Bash prompt.

The **~/.bashrc** file is read upon each login to a Bash session – in other words, every time you log into a console or open a terminal. That's unless you change Raspbian's default shell away from Bash, something you may have reason to do in the future– there are interesting alternatives available for extra features or for smaller memory footprint– but let's not worry about that for now. You can put all sorts of commands in there to personalise your environment: command aliases are great for regularly used combinations.

## Alias

See what's already there with:

```
grep alias ~/.bashrc
```

There are a few already in there, particularly for the **ls** command. One entry is:

```
# alias ll='ls -l'
```

…which sounds quite useful, although the # indicates that it is 'commented out' – in other words, it will not be read by Bash. Open **.bashrc** in your text editor; the simple text editor from the Accessories menu will do for now, as although we've touched on using **nano** for editing text from the command line, we haven't time to go into it in detail until the next part of this series. Removing the # will mean that now when you type **ll**, you'll get the action of running **ls -l**. Handy, but we could make it better. Change it to:

```
alias ll='ls -lAhF'
```

…and you'll get an output in KB or MB, rather than bytes, along with trailing slashes on directory names and the omission of the ever present **.** and **..** (current and parent) directories. Changes take effect after you next start a Bash session, but you can just run that alias as a command (**Fig 1**). To disable an alias for a session, use:

```
unalias ll
```

## Key point

We'll end with the very first thing many users need to change – the keyboard map. The system-wide setting is in **/etc/default/keyboard**, but often you need to change it just for individual users. If £ signs and letters without accents are not enough for them, log in as the user who wants a different keyboard, or add **sudo** and the correct path to the commands below. For example, for a Greek keyboard:

```
touch ~/.xsessionrc
echo "setxkbmap el" > ~/.xsessionrc
```

Replace **el** with **pt**, **us**, or whatever language you desire. Note that config file we created – **.xsessionrc** – it holds settings that are read when we start the GUI with **startx**, so the keyboard setting will cover not just the terminal, but every app used in the session.

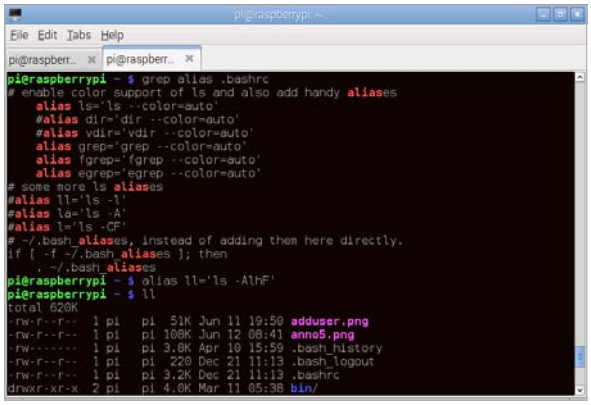



**Fig 1** Those terse, two- or three-letter commands are not set in stone – make your own shortcuts to keep, or just for use over a specific session

### HOME RUN

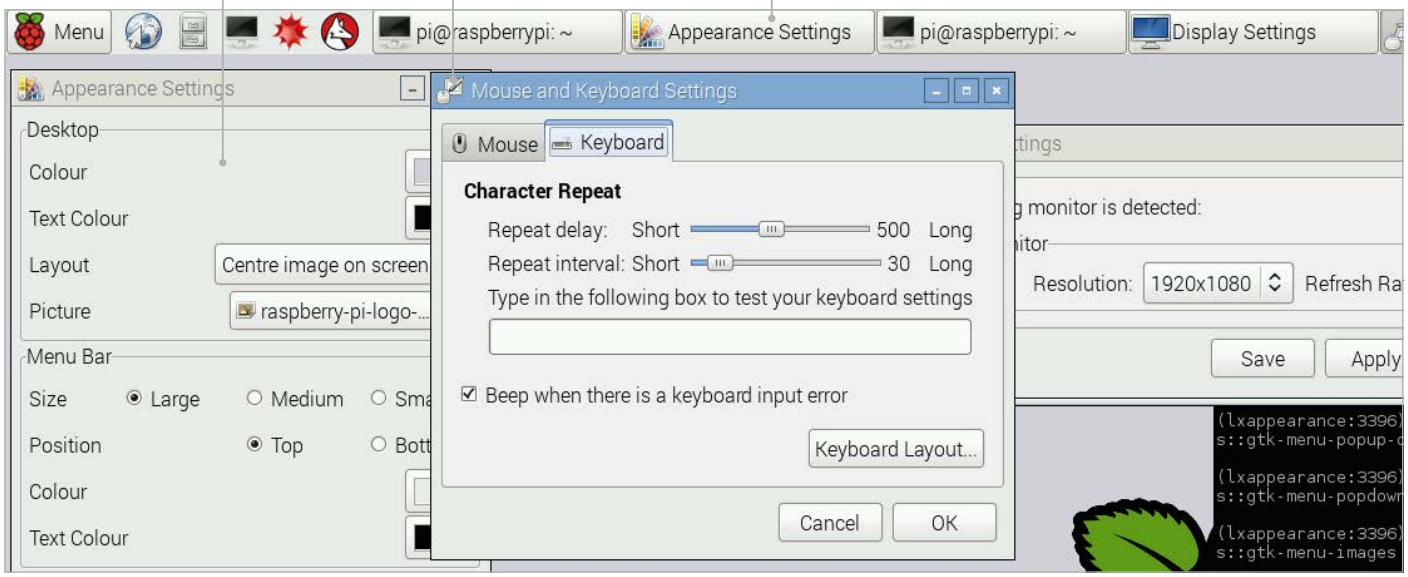
If you're logged in as user **pi**, then **~** is a shortcut to **/home/pi** – but **ls ~jo** can be used as a shortcut to list **/home/jo**, substituting any other user name as desired, with tab completion working after **~j** is typed.

### WHO AM I?

From a virtual console (before typing **startx**), **su** and that's who you're logged in as. From an xterm, you can change to someone else, but start another app from the menu and you'll be back to your original login.

**SIMON LONG**

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




# HACKING RASPBIAN'S DESKTOP PART 4: CUSTOMISING APPLICATIONS

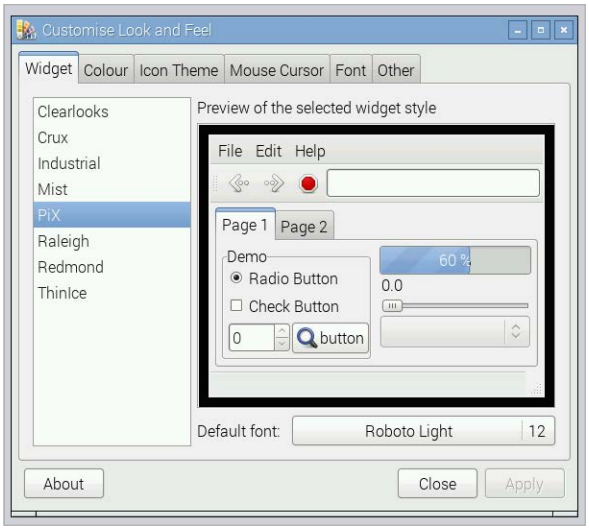
In the fourth and final part of his series, **Simon Long** talks us through how to customise the appearance of applications running on the Raspbian desktop...

Applications written to run under Raspbian's desktop environment, LXDE, make use of one of a number of user interface toolkits.

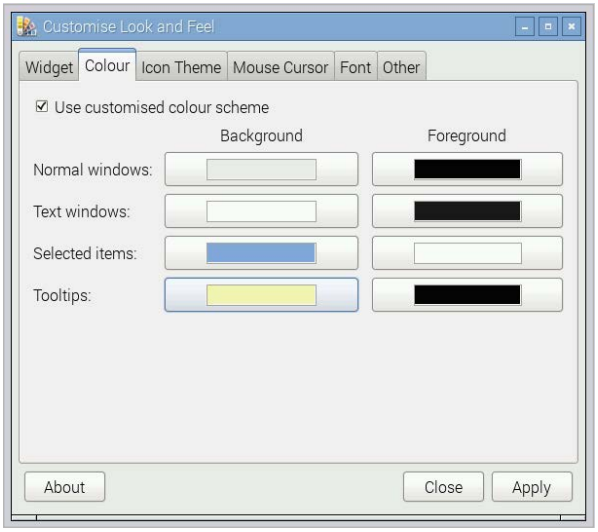
A UI toolkit provides reusable code for standard UI elements, such as windows, menus, and buttons. This serves two main purposes. First, it makes creating new applications a lot faster for the developer, as it isn't necessary to create the code for a button from scratch every time one is needed in an application. Second, it ensures that applications look and behave in a consistent fashion – a button in one application will look exactly the same as in every other application, and will display the same behaviour (such as changing appearance when it is clicked) everywhere in the system.

That's in an ideal world anyway. Unfortunately, things aren't quite that ideal, because there are a number of different toolkits out there, and it is up to the application developer to choose which one to use. LXDE itself is written using a toolkit called GTK+ (GIMP Toolkit), and most of the applications included in the Raspbian image also use GTK+. However, there are others – you may see applications written with a toolkit called Qt – and even with GTK+, there is extra complexity due to there being multiple versions in use: you'll find applications which use version 2.x and version 3.x, and they can look significantly different.

All very interesting, but why should a user care? One of the advantages of using a UI toolkit is that it can be themed. In other words, while the toolkit ensures that




**Above** The first page of the lxappearance application lists the themes installed on the system and shows what each widget looks like in them




**Above** Other pages of lxappearance allow colour schemes, icons, fonts, and cursors to be customised

every button looks the same as every other button, the user can change the theme to make buttons look the way they want them to, while keeping them consistent across all applications. By changing the GTK+ theme, you can make everything on your Pi's desktop – including the menu bar – change its appearance at the same time.

## Choosing a theme

There are a number of alternative themes already installed in Raspbian – they can be found in the directory **/usr/share/themes** – or you can download and install new themes from various places online. Themes can also be installed for individual users in the **.themes** subdirectory of their home directory. A theme directory contains subdirectories with theming information for different toolkits – most of LXDE is written using version 2 of GTK+, so the **gtk-2.0** subdirectory contains the relevant theme file, which is called **gtkrc**.

The easiest way to change the appearance of your applications is just to change the theme to another pre-installed theme, and there is an application installed in LXDE to do this for you. From a terminal window, run **lxappearance**.

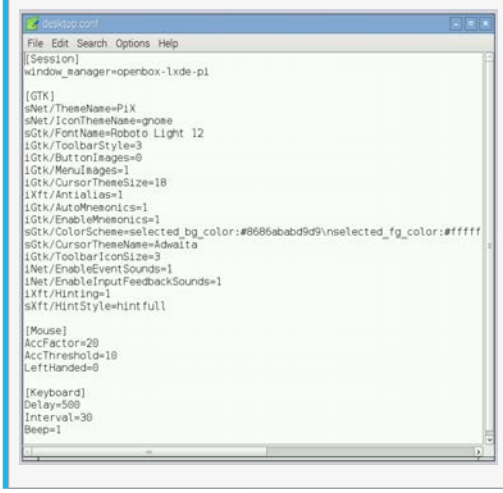
This shows a list of all GTK+ themes installed. Clicking on any of them shows how various UI elements will look under that theme on the example to the right of the screen. Choose one, click Apply, and it will change the appearance of the whole system to match.

The **lxappearance** application also allows you to customise various other aspects of the appearance and behaviour of the desktop, so feel free to explore the other tabs and options. For instance, the Widget tab also allows you to change the system font, which is used for all text displayed inside a window. (Note that to change the font used for the title bar of a window, you need to change the font used by Openbox – see the previous article for details of how to do this.)

## Customising themes

If you really want to change every detail of the appearance of your desktop, you can modify the theme file itself, download one of the many custom themes online, or even create one from scratch. The **gtkrc** files mentioned above are plain text and can be modified with your favourite editor – however, this is not for the faint-hearted, and you can end up with some truly bizarre effects if you don't understand what you are doing. A full tutorial on modifying GTK+ theme files is outside the scope of this article, but information can be found at the GTK+ website, **gtk.org**.

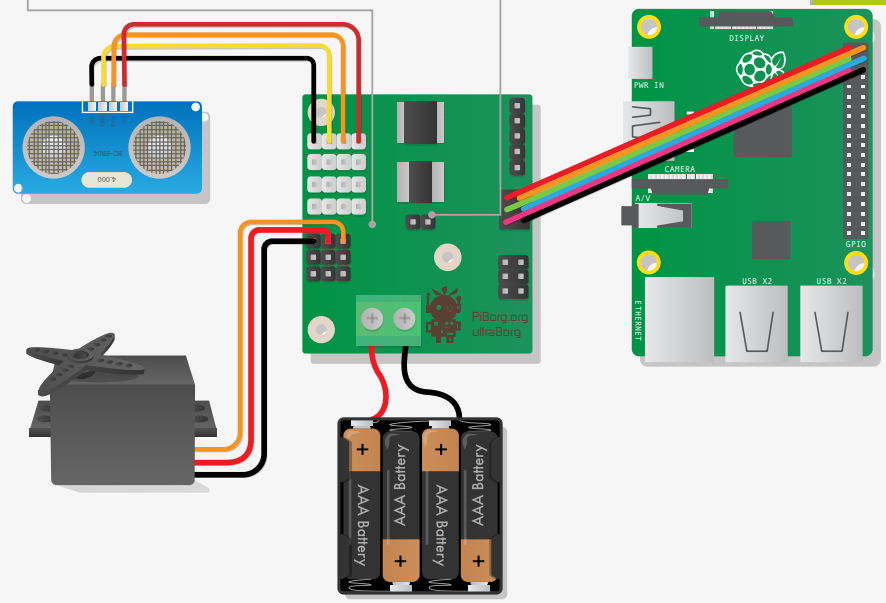
## THE SESSION CONFIGURATION FILE



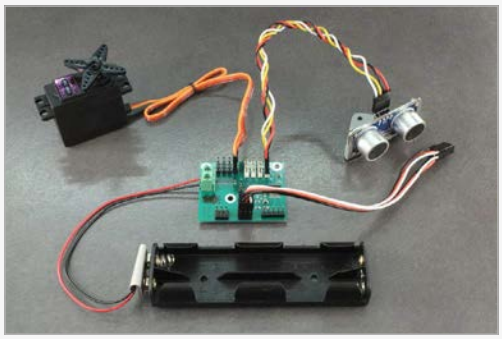

All the changes made in the **lxappearance** application are stored in a file in the hidden **.config** subdirectory of your home directory. Look inside **.config/lxsession** for a directory with the name of your current lxsession profile – this is called **LXDE-pi** on a default installation of Raspbian – and inside this is a file called **desktop.conf**. Each line in this file corresponds to a setting in **lxappearance**, so you can use a text editor to modify this file instead of using **lxappearance** if you prefer. If you edit the file, you will need to run **lxsession -r** from a terminal window to force the new settings to take effect, or just reboot your Pi.

**ARRON CHURCHILL**

Arron writes the software for the robots at **PiBorg.org**, giving them the ability to chase balls and obey commands. We know what you're thinking, but robots are cheaper and tidier than dogs.
**piborg.org**




**Below** The various parts connected to UltraBorg, just missing the Raspberry Pi. The sensor is attached to one of PiBorg's optional mounting kits



## You'll Need

- UltraBorg **piborg.org/ultraborg**
- HC-SR04 sensor
- 4-pin sensor cable
- Servo motor
- 4x AA or AAA battery holder

**All items are available from piborg.org or third-party vendors**

# MOVE A SERVO WITH A FLICK OF THE WRIST

The UltraBorg helps you use servos and sensors with your projects. Here you'll make a servo move by waving your hands like a lunatic...

When building robots, we tend to think of walking or driving robots, but they can take many different forms. Servos are an inexpensive form of robot motor and they're used in many robots, such as planes (UAVs), robotic arms, and even remote-controlled (RC) cars. Ultrasonic sensors are another useful bit of robot-building kit. You can use them to determine how far away things are from the robot, which is ideal for collision avoidance. Both are cheap and easy to obtain, so in this project we're going to demonstrate how you can use both of them together to build a very simple robot that responds to motion.

## >STEP-01

### Connect the UltraBorg

Start by removing the jumper that is fitted on the two-pin connector in the middle of the board. With the Raspberry Pi powered off, connect the pair of three-pin cables between the UltraBorg and the GPIO header. Make sure the cables are fitted so that the pin marked '1' on the UltraBorg is connected to pin 1 on the GPIO header – that's the pin that is closest to the SD card slot.

## >STEP-02

### Connect the servo

Next, we connect the servo to one of the three-pin connectors. The servo connections are numbered 1 to 4, with #1 towards the middle of the board. You may use any combination of the four available connectors; for this project, we want to connect the servo up to the #1 connector. Make sure the servo is connected the correct way around: the ground (also called GND, -VE, or 0V) wire should be connected to the pin closest to the edge of the board. Servo cables may vary in colour, but usually the ground wire is black or brown.

## >STEP-03

### Connect the HC-SR04

The four-pin ultrasonic connections are also numbered 1 to 4; 1 is at the top of the board. Since the sensor has pins instead of a cable, we will need to fit a four-pin female-to-female cable. Connect the cable to the HC-SR04; make a note of which colour wire is connected to the ground pin, marked 'Gnd' on the sensor. Now connect the other end of the cable to the #1 connection on the UltraBorg; the ground wire should be attached to the pin closest to the edge of the board.

## >STEP-04

### Power the servo and sensor

The UltraBorg uses an external power connection for the servos and ultrasonic sensors. Take the positive wire from the battery holder (usually red) and place the exposed metal into the left screw terminal. Screw the left terminal down until the wire is held firmly. Take the negative wire from the battery holder (usually black) and place the exposed metal into the right screw terminal. Screw the right terminal down until the wire is held firmly.

Now fit four rechargeable batteries into the holder. The four batteries will supply approximately 5V of power, which is fine for the servos and sensors.

## >STEP-05

### Install the software

Now we need to get our Raspberry Pi running. Connect the power supply and wait for the Raspberry Pi to boot. Open a terminal and use this command to install the software:

```
bash <(curl https://www.piborg.org/install-ultraborg.txt)
```

After the software has installed, restart the Pi. You can check that the servo and HC-SR04 are connected and working by using the demo GUI icon on the desktop. The distance reading should change as you move things in front of the sensor. The leftmost slider should move the servo around. All we need now is the example script to control the servo for us.

## >STEP-06

### Run the example

First, download the example script from GitHub:

```
git clone https://github.com/piborg/MoveMyServo.git
```

Now, move to the new folder and run the script:

```
cd MoveMyServo
./MoveMyServo.py
```

It works, but we can improve it. Lines 8 to 11 allow us to change how the movement works. By changing the values of **distanceMin** and **distanceMax**, you can change how far you have to move your hand or object. By changing **fastMode** to **False**, you can make the servo slower. Also, changing **updateInterval** slows down the servo response. See what else you can do.

## MoveMyServo.py

**Language**
>PYTHON 2.7

**DOWNLOAD:**
github.com/piborg/MoveMyServo

```
# Import the libraries we need
import UltraBorg
import time

# Settings
# Minimum distance in mm, corresponds to servo at -100%
distanceMin = 100.0
# Maximum distance in mm, corresponds to servo at +100%
distanceMax = 300.0
# True moves faster, False gives a more stable position
fastMode = True
# Time between updates, smaller is faster
updateInterval = 0.1
# Start the UltraBorg
UB = UltraBorg.UltraBorg() # Create a new UltraBorg object
UB.Init() # Set the board up (checks the board is connected)

# Calculate our divisor
distanceDiv = (distanceMax - distanceMin) / 2.0

# Loop over the sequence until the user presses CTRL+C
print 'Press CTRL+C to finish'
try:
    # Set our initial position
    lastServoPosition = 0.0
    newServoPosition = 0.0
    UB.SetServoPosition1(lastServoPosition)
    # This is the loop which reads the sensor and sets the servo
    while True:
        # Read the ultrasonic values
        if fastMode:
            # We use the raw values so we respond quickly
            distanceMeasured = UB.GetRawDistance1()

        else:
            # We use the filtered values so we get stable readings
            distanceMeasured = UB.GetDistance1()
        # Convert to the nearest millimeter
        distanceMeasured = int(distanceMeasured)

        # Generate the servo positions based on distance readings
        if distanceMeasured != 0:
            newServoPosition = ((distanceMeasured - distanceMin)
/ distanceDiv) - 1.0
            if newServoPosition > 1.0:
                newServoPosition = 1.0
            elif newServoPosition < -1.0:
                newServoPosition = -1.0
        # Display our readings
        print '%4d mm -> %.1f %%' % (distanceMeasured,
newServoPosition * 100.0)

# Set our new servo position if it has changed
        if newServoPosition != lastServoPosition:
            UB.SetServoPosition1(newServoPosition)
            lastServoPosition = newServoPosition
        # Wait between readings
        time.sleep(updateInterval)

except KeyboardInterrupt:
    # User has pressed CTRL+C
    print 'Done'
```

**RICHARD HAYLER**

Richard is a mentor at CoderDojo Ham, and his school CodeClub was one of the winning teams in the primary category of the Astro Pi competition.
**richardhayler.blogspot.co.uk**
**coderdojoham.org**

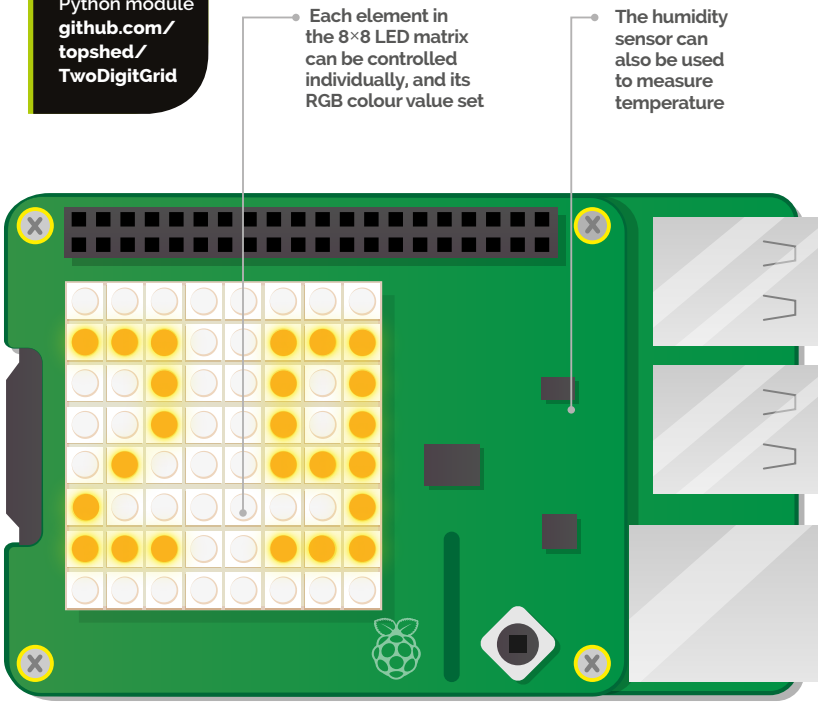
# SENSING YOUR ENVIRONMENT WITH ASTRO PI

The Astro Pi is packed with sensors and has a dazzlingly colourful LED screen. Here's how to use it to do real scientific measurements...

## You'll Need

- An Astro Pi HAT **raspberrypi.org/astro-pi-tech-specs/**
- AstroPi Python module **pythonhosted.org/astro-pi/#installation**
- TwoDigitGrid Python module **github.com/topshed/TwoDigitGrid**

The Astro Pi is one of the most sought-after Raspberry Pi HATs. If you've been lucky enough to get hold of one, this is a great first project to introduce you to some of the amazing functions available. We'll be using the accelerometer, humidity sensor, and the wonderful multicoloured LED matrix to create an interactive sensor that displays the temperature or humidity, depending on which way you hold it. The colour of the digits will indicate whether the latest reading is higher or lower than the previous one. Perfect for use in zero gravity!




## >STEP-01

### Get the Astro Pi Python API

If you have a complete Astro Pi kit, then you should have a pre-installed SD card with everything you need. However, if you just received a HAT on its own, you'll need to set up your Pi before you can start using it. First, install and boot the latest vanilla version of Raspbian. Then connect your Pi to the internet and run the following command from a terminal window to download and launch the Astro Pi install script:

```
wget -O - http://www.raspberrypi.org/files/astro-pi/astro-pi-install.sh --no-check-certificate | bash
```

## >STEP-02

### Connect the Astro Pi HAT

The Astro Pi has only two components: the circuit board and a narrow connector which simply snaps on to the corresponding header underneath. Once this is assembled, power off the Pi and attach the Astro Pi. The connector will fit snugly onto the GPIO pins, just like other HATs. There should also be four hexagonal spacing columns that keep the board firmly attached and stop it from flexing. Be careful not to overtighten the screws.

## >STEP-03

### Testing, testing

Once the Astro Pi is attached, power the Pi back up and start a Python shell. You can either run it directly from the command line or use IDLE (our code will work with Python 3.3 or 2.7). Now type in the test code from listing 1. When finished, you should see a message scrolling across the LED matrix. As you've probably guessed, you can adjust the RGB values passed into the function as `text_colour` to alter the colour of the text.

# Code listings

### Listing 1
```
from astro_pi import AstroPi
ap = AstroPi()
ap.show_message("Hello, Space!",
text_colour=[255,255,255])
```

### Listing 2
```
from astro_pi import AstroPi
ap = AstroPi()
while True:
        ap.get_accelerometer()
```

### Listing 3
```
from astro_pi import AstroPi
import time
while True:
        ap.get_temperature()
        ap.get_humidity()
        time.sleep(0.5)
```

## >STEP-04
### Try the accelerometer
Some of the sensors on the Astro Pi require superuser privileges to run, so now you need to run a Python shell or IDLE with the **sudo** command. Type in the Python from listing 2 (don't forget to indent the last line). The output should be values that correspond to the pitch, yaw, and roll axes of the Pi. Change the orientation of the Pi and you should see the values change. When you've finished playing, press **CTRL+C** to stop the code.

## >STEP-05
### Get hot and sweaty!
Keep the same shell or IDLE window running and add the lines from listing 3. This is similar to what we did in step 4 but with a **sleep** included to slow the output from the two sensors down. Typical values in the UK should be around 30–40% for the humidity and 30 degrees for temperature (the Pi itself generates heat, so this reading will be above room temperature). Gently put your finger on the humidity sensor chip, which is labelled on the top of the Astro Pi, and you should see the temperature rise. Now breathe out over the Pi and you should see the humidity rise too. How hot and steamy can you make it?

## >STEP-06
### Putting it all together
Now let's use these functions to make an interactive data logger. To display measurements on the LEDs, we could just use the scrolling text method from step 3. This can be tricky to read at a glance, so we've written a simple module (TwoDigitGrid) that takes a two-digit number and creates a 64-item list, suitable for display on the LEDs using the **set_pixels()** function. Install TwoDigitGrid from GitHub and then create a file called **HTLogger.py** in the **TwoDigitGrid** directory. Add the code from listing 4 and then run it with **sudo**.

In addition to being shown on the LEDs, the temperature and humidity readings will be recorded in a file called **readings-<date>-<time>.log.**

Language
>PYTHON 2.7

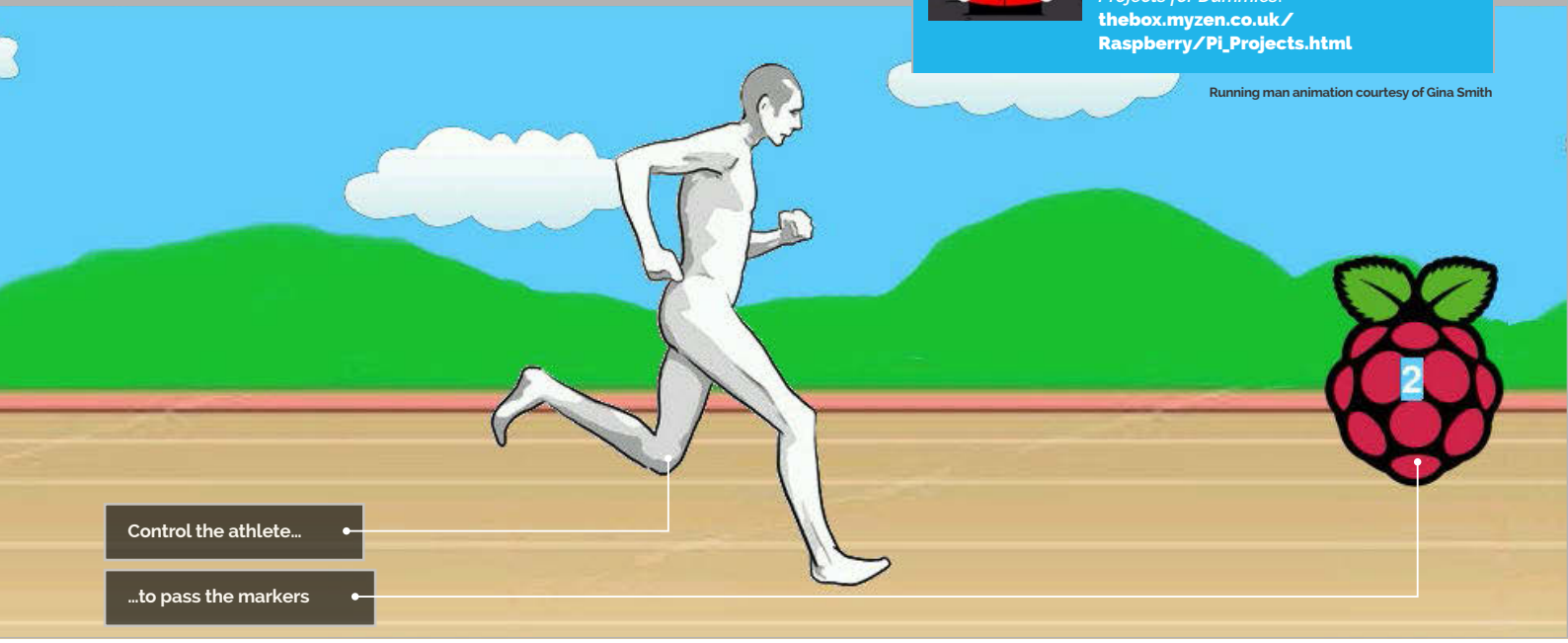
**DOWNLOAD:**
github.com/topshed/HTLogger

### Listing 4
```
from astro_pi import AstroPi
import time
import logging
from datetime import datetime
import TwoDigitGrid as TDG

ap = AstroPi()

dataWriteInterval = 120 # how often data is
saved
dataDisplayInterval = 0.5 # how often the displayed value is updated

tmstmp = time.strftime("%Y%m%d-%H%M%S") # Set timestamp format

logging.basicConfig(
format='%(asctime)s %(message)s',filename='readings'+str(tmstmp)
+'.log',level=logging.DEBUG) # set up logging

hum_prev = 0 # set previous humidity and temp values to zero
temp_prev = 0
sec_count = 0

while True: # Main program loop
        # Get raw accelerometer values and round them
        x, y, z = ap.get_accelerometer_raw().values()

        x = round(x, 0)
        y = round(y, 0)
        temp_f = ap.get_temperature() # Get temperature from astro-pi
        hum_f = ap.get_humidity() # Get humidity from astro-pi
        hum_int = int(hum_f) # convert to integers
        temp_int = int(temp_f)

        if (sec_count >= dataWriteInterval/dataDisplayInterval) or \
(sec_count == 0):
                logging.info('humidity: ' + str(hum_f) +
' temperature: ' + str(temp_f))
                sec_count = 0

        if x == -1 and y != -1: # humidity display if HDMI port is up

                ap.set_rotation(270) # rotate the image on the LED
                if hum_int > hum_prev: # Is the latest reading higher?
                        r = [0,255,0] # green if higher
                elif hum_int == hum_prev:
                        r = [0,0,255] # blue if the same
                else:
                        r = [0,255,255] # light blue if lower
                hum_prev = hum_int
                image=TDG.numToMatrix(hum_int,
back_colour=[0,0,0],text_colour=r)
                ap.set_pixels(image)

        elif y == -1 and x != -1: # temp display if USB ports point up
                ap.set_rotation(180) # rotate the image on the LED
                if temp_int > temp_prev: # Is latest reading higher?
                        r = [255,0,0] # red if higher
                elif temp_int == temp_prev:
                        r = [255,128,0] # orange if the same
                else:
                        r = [255,215,0] # yellow if lower
                temp_prev = temp_int
                # use numToMatrix to turn number into a 64-item list
                image=TDG.numToMatrix(temp_int,
back_colour=[0,0,0],text_colour=r)
                ap.set_pixels(image)

        elif x == 0 and y == 0: # if the Pi is flat on its back
                ap.show_message("Recording",
text_colour=[150,150,150],scroll_speed=0.03)

        else:
                # display a '?' if at another orientation
                ap.show_letter("?")

        time.sleep(dataDisplayInterval)
        sec_count+=1
```

# 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*.
**thebox.myzen.co.uk/Raspberry/Pi_Projects.html**




Running man animation courtesy of Gina Smith

# PI SPRINTER

Relive the glory years with 1984's phenomenally successful Micro Olympics with new graphics and home-made foot controllers...

**You'll Need**

- 2× piezoelectric sensors
- 2× 2N7000 FETs
- 2× small pieces of stripboard
- 2× 1M resistors
- 6mm thick MFD & paint
- 4m thin microphone wire
- 8× M3 hex spacers 5mm long
- 16× M3 countersunk screws 10mm long
- Connectors for GPIO

Back in the early days of the BBC Micro, there was a phenomenally successful game called *Micro Olympics*, designed as a tie-in for the 1984 Olympic games in Los Angeles. One of the highlights of the game was the sprint, where players hammered alternately on the keyboard to get blocky little men to run. This month we would like to make an updated Pi Bakery version that not only spares your keyboard, but has infinitely better graphics too.

As this is a running race, it is only appropriate that it should be foot operated; however, in keeping with the modern computer age, it is a running race you can do sitting down. So first of all, we are going to make some foot sensors and then look at the software to bring the race to life.

## The hardware

While you can use a mechanical foot switch, it is not very good as it is hard to get the switches to respond rapidly; instead we are going to use piezoelectric sensors, one for each foot. These are readily available and quite cheap, and can generate a surprisingly high voltage, as shown by the signal you can get from just flicking the back of the sensor with your finger. That peak voltage is about 45 volts and only lasts about 6 milliseconds; also, notice the aftershock, a secondary peak as the sensor rebounds. That is way too high to feed into the Raspberry Pi, so it needs some simple conditioning. You might see some projects that use a 1M resistor in parallel with the sensor and fed directly into a processor. Even with a resistor, the voltage is twice as high as the maximum voltage you should feed into the GPIO pins. To fix this, we feed into a FET so the sensor voltage causes the FET to turn on and pull down a GPIO pin which has a pull-up resistor fitted. The circuit diagram is shown in **Fig 1**, along with how to lay it out on stripboard. You only need a tiny piece of stripboard, 9 holes by 5. You will need two of these circuits, each mounted in a separate box – see the 'Building the foot switches' boxout overleaf for construction details. There are five or six different types of signal you get when you tap them with your foot, depending on how you do it. Notice the rapid pulses followed by a large secondary pulse when the sensor bounces; we can arrange the software to ignore this by specifying a 'debounce' time in it.

## The software - principal

As you can see from an oscilloscope trace, the signal from the foot sensor lasts only 5ms or so,

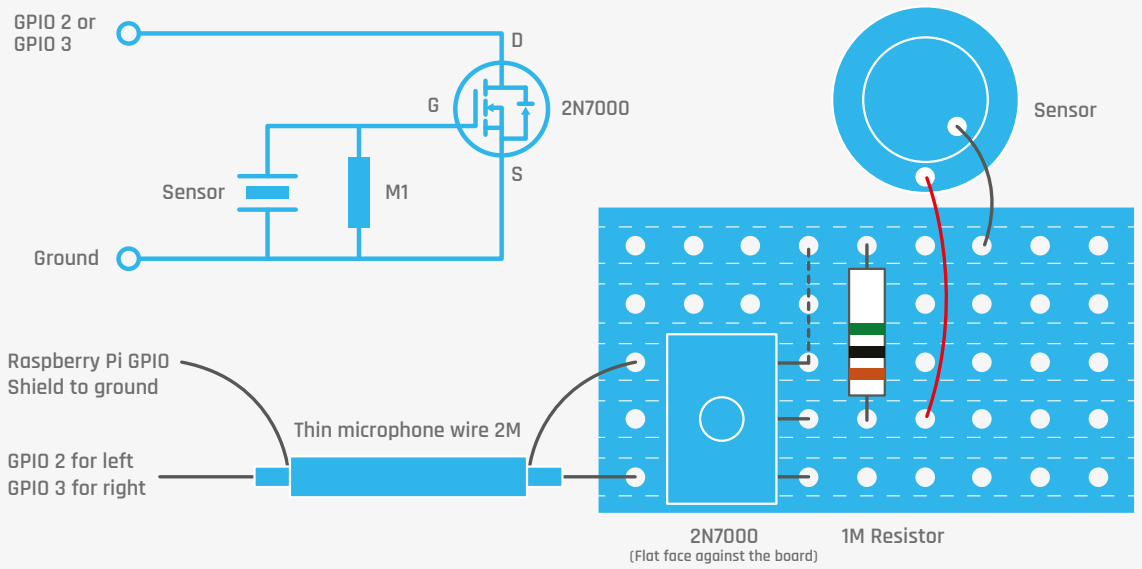



Fig 1 Schematic and physical layout of the sensors

therefore you need to have your code looking at these switches at intervals of 5ms or less. That is a big ask for an interpreted language like Python working under a time-stealing operating system like Linux. Fortunately, there is a very neat way around this. Buried in the processor is a GPIO pin register that detects not logic levels but edges. Once a transition occurs, this can cause an interrupt which causes a Python function to be called asynchronously with the code running. There is a degree of buffering so that these interrupts can stack up to some limited extent. All that we want to do here is to have a variable incremented; this indicates a step has been recorded.

In order to use interrupts, we have had to switch from our usual WiringPi library and go with the pre-installed **RPi.GPIO**. This was a bit of a problem because we found it would not update: the software kept telling us it was at the latest version when it was not. Therefore we had to uninstall the previous version of **RPi.GPIO** by hand before installing the new one, using:

```
$ sudo apt-get update
$ sudo apt-get remove python-rpi.gpio
$ sudo pip uninstall RPi.GPIO
$ sudo apt-get install python-rpi.gpio
```

## Graphics

The idea is that on each step, a different image of a runner is plotted on the screen; when done in quick succession, this makes it look like he is running on the spot. Our art skills being somewhat lacking, we looked around for some clip art and came across this video (**vimeo.com/35916928**) from Gina Smith. We downloaded it and converted it to a sequence of individual frames. Then we cut down the sequence by removing duplicate frames and isolated the runner by making the background transparent. This required a bit of touching up of each image in a pixel editor to prevent the flood fill leaking inside the figure. Each frame was saved as a PNG-type image to preserve the transparent layer.

Now, to get the illusion of movement, you have to plot the figure in a new place on each step, by increasing the X coordinate of where it is drawn. However, if this is made large enough to look realistic, then the runner races across the screen in only a few strides. The solution to this is to have a small amount of movement made for the runner and a much larger amount of movement made to the background.

To that end, we created a background image that was twice as wide as the screen window. At first, it is plotted on the screen window at a position of X = 0. Then, to get the background to scroll, it is plotted at a successively larger value of a negative offset position in the screen window. This has the effect of missing out the first part of the image. As this offset gets larger and larger, a point is reached where the negative offset is so large that the background image is not wide enough to fill the screen window. At this point, the background image is plotted again, but this time with a positive offset so the first part of the background image is joined onto the end of the image. It is said to 'wrap round'.

In drawing the background, the trick is to make the end of the image match up seamlessly with the start of the image. We drew an image that did this and populated the background with white fluffy clouds and rolling hills; if you want to make it more interesting, you can easily add other details in the background, such as clip-art buildings. At the end of the background image, we placed a raspberry; this was to act as a distance marker and the finish line. When it is on the screen, a number is placed in the middle of it to show how far you have to race. We found that passing four raspberries made the game the 'right' length.

So, there are two components to the movement: the background going to the left and the running figure

going to the right. When the figure reaches the end of the screen, the race is over. By carefully tuning the amount of movement per step of each component, we could get the end position of both components to coincide. A total of 180 steps finishes the race.

## The code

The program runs under the Pygame system and has a window of 1,000 pixels wide by 400 pixels high; you could make this larger if your display can accommodate it, but it just about covers most of a standard screen. Sound effects are used to enhance the race experience and the samples are in a **sounds** directory, alongside the code and the **images** directory. The 12 images of the runner are loaded into a `runningFrames` list and the sounds into the `gameSounds` list, the Pygame system is initialised and the GPIO pins set up. Note here that the **RPi.GPIO** library has what we consider to be a fault, in that it throws up an unnecessary warning when GPIO 2 and 3 are initialised – this should be ignored.

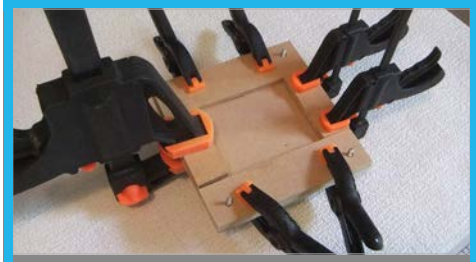
The main function then starts the race with a fixed 'on your marks, get set, go' sequence, with the go being a starting-gun sound effect. Note here that there is a random delay between the 'get set' and the start gun in order to add a bit of real-life tension.

The interrupt function `buttonPress` is invoked when either of the foot switches is sensed, incrementing the global variable `moved`. It is this variable that is used to see if the screen and background scrolling need to be updated with the next step. The `moved` variable is immediately decremented and then the screen display is updated with the `showPicture` function. This draws the background image and if it will not fill the window, it draws the start portion of the background image as well. Then, if the raspberry is visible, it draws the countdown number on it. Then the two position variables are adjusted for the next step. The `manDistance` variable is where to draw the runner, and the `distance` variable is where to draw the background. The `posts` variable is how many raspberry posts are needed to finish the run, and the `frame` variable is what image of the runner to draw next. Finally, if there are no more posts left to pass and the distance is greater than 1160, the race is over, the crowd cheers, and the time for the race is displayed. Pressing the **RETURN** key starts the race again.

## Performance

By removing the `moved -= 1` line, you can get an idea of how much processing is involved for the graphics. Using the Raspberry Pi 2, the race will be over in about 4.5 seconds, whereas on the original Pi it takes around 15 seconds. This does not affect the game, however, as you won't be able to 'run' fast enough to hit these limits, whatever Pi you use.

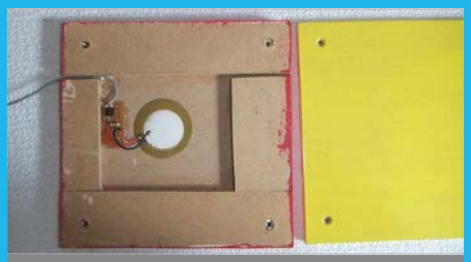
# BUILDING THE FOOT SWITCHES



### >STEP-01

**Making the box**

Cut two pieces of 6mm-thick MDF, 130mm square. Clamp together so they line up and drill a 3mm hole in each corner. Now take four scrap pieces of the same material and arrange them around the outside, leaving a small well in the middle. Use the top holes as a template and drill through the pieces so the holes line up. Then enlarge the holes in the pieces to 6mm with a drill and push a M3 5mm hexagonal pillar into each hole. Make sure there is enough space to get a wire between one of the pieces. Glue with white woodworking glue and clamp them together while the glue dries. You will need two of these.



### >STEP-02

**Finishing the boxes**

Countersink the holes in the top and the base and paint them. Paint the single-piece base for both units the same colour (we used yellow) and paint the top units another colour (we used blue and red). Make sure the top and the bottom can be screwed together to the pillars. We used 10mm M3 countersunk screws for one side and had to file off a few mm of the screws for the other side to allow them to tighten up without the screws meeting in the pillar. Make two of the circuits in Fig 1. Glue the large side of the sensor to the top plate using a very thin smear of contact adhesive. Fix the circuit and the wire with hot melt glue.

### >STEP-03

**Using the sensors**

Fasten the base and top together and connect to the Raspberry Pi's GPIO port. Place the two units on the floor and arrange them to be under each foot when you are seated. When the starting gun goes, tap on each box alternately as fast as possible. Be sure to tell anyone else in the vicinity what you are doing to avoid concerned enquiries. You might consider mounting these two sensors in one large box so that you can actually do running on the spot without being seated. Make sure the side with the sensor glued to it is uppermost, for the best sensitivity. We will be using these foot switches in future projects in this series, so you will get your money's worth from them.

# racer.py

Language
>PYTHON 2.7

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

```
import pygame, time, os, random
import RPi.GPIO as io

pygame.init()              # initialise graphics interface
pygame.mixer.quit()
pygame.mixer.init(frequency=22050, size=-16, channels=2,
buffer=512)

os.environ['SDL_VIDEO_WINDOW_POS'] = 'center'
pygame.display.set_caption("The Pi Sprinter")
pygame.event.set_allowed(None)
pygame.event.set_allowed([pygame.KEYDOWN,pygame.QUIT])
screen = pygame.display.set_mode([1000,400],0,32)
textHeight = 36
font = pygame.font.Font(None, textHeight)

try :
   io.setmode(io.BCM)
   io.setwarnings(False)
except :
   print"start IDLE with 'gksudo idle' from command line"
   os._exit(1)

pinList = [2,3] # GPIO pins for input switches
random.seed()
moved = 0
restart = True
soundEffects = ["marks","set","go","end"]

runningFrames = [ pygame.image.load(
"images/Man"+str(frame)+".png").convert_alpha()
                  for frame in range(0,12)]
background =  pygame.image.load(
"images/BackgroundPi.png").convert_alpha()
gameSound = [ pygame.mixer.Sound(
"sounds/"+soundEffects[sound]+".wav")
                  for sound in range(0,4)]

def main():
   global moved, restart
   initGPIO()
   print"Pi Racer"
   while True:
     checkForEvent()
     if restart:
        frame = 0
        distance = 0
        manDistance = -85
        posts = 3
        moved = 1
        restart = False
        print"On your marks"
        gameSound[0].play()
        showPicture(frame,distance,manDistance,posts)
        time.sleep(2.0)
        print"Get set"
        gameSound[1].play()
        time.sleep(random.randint(2,5))
        print"Go"
        startTime = time.time()
        gameSound[2].play()
     if moved > 0 :
        moved -= 1
        showPicture(frame,distance,manDistance,posts)
        manDistance += 5
        distance += 40
        if distance > 3000:
           distance -= 2000
           posts -=1
        frame = frame + 1
        if frame > 11:
           frame = 0
     if posts == 0 and distance >=1160 :
        raceTime = int(100*(time.time()
- startTime))
        gameSound[3].play()
        drawWords("Finished "+str(raceTime / 100.0)+
" Seconds",400,220)
        pygame.display.update()
        print"Type return for another race"
        while not restart:
           checkForEvent()

def initGPIO():
   print"Please ignore this stupid warning:-"
   for pin in range (0,2):
      io.setup(pinList[pin],io.IN, pull_up_down = io.PUD_UP)
      io.add_event_detect(pinList[pin],io.FALLING,
callback=buttonPress, bouncetime=30)

def showPicture(frame,distance,manDistance,post):
   screen.blit(background,[-distance,0])
   if distance > 1000 :
      screen.blit(background,[1999-distance,0])
   drawWords(str(post),-distance+1932,220)
   screen.blit(runningFrames[frame],[manDistance,0])
   pygame.display.update()

def drawWords(words,x,y) :
        textSurface = pygame.Surface((14,textHeight))
        textRect = textSurface.get_rect()
        textRect.left = x
        textRect.top = y
        pygame.draw.rect(screen,(102,204,255),
(x,y,14,textHeight-10), 0)
        textSurface = font.render(words, True,
(255,255,255), (102,204,255))
        screen.blit(textSurface, textRect)

def buttonPress(number): # call back function
   global moved
   moved += 1

def terminate(): # close down the program
    print "Closing down please wait"
    io.cleanup()
    pygame.mixer.quit()
    pygame.quit() # close pygame
    os._exit(1)

def checkForEvent(): # see if we need to quit
    global restart
    event = pygame.event.poll()
    if event.type == pygame.QUIT :
         terminate()
    if event.type == pygame.KEYDOWN :
       if event.key == pygame.K_ESCAPE :
          terminate()
       if event.key == pygame.K_RETURN :
          restart = True
          print"Start the race again"

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

PART 5

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**

# PYGAME SOUNDBOARD

Learn about loading and playing sounds in your Pygame projects with this fun farmyard soundboard...

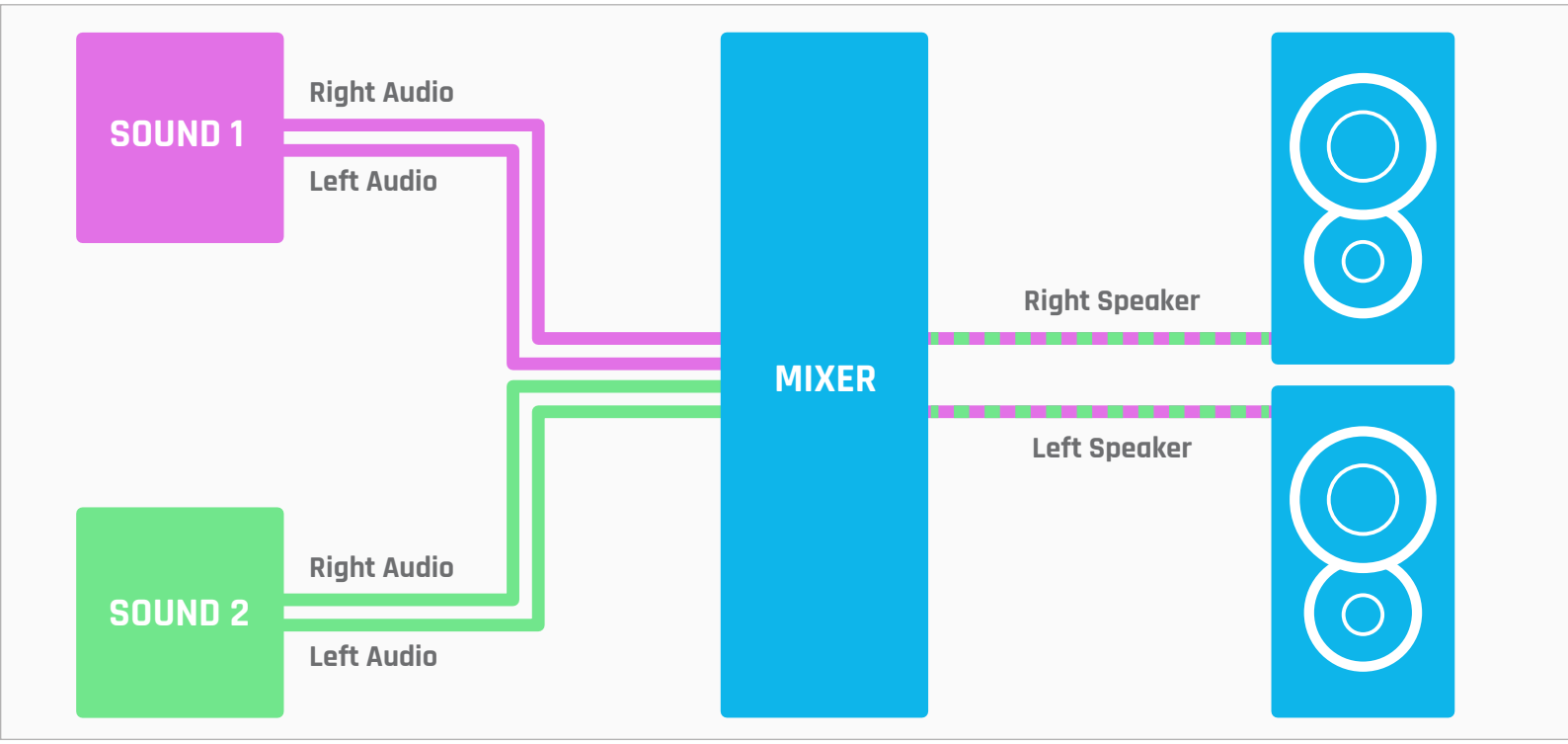
In our last tutorial, we put together a simple video game in which we tried to avoid the dreadful fate of being crushed by a ceiling by dropping through platforms into the space below. It didn't have the fanciest graphics, but fancy graphics aren't everything. Games with just two colours can be great! One simple thing that we can do to enhance our players' experience is to add sounds, and that's what we're going to be doing this month. We're going to learn how sounds work with Pygame by putting together a soundboard with some simple controls. We'll learn about loading sounds, playing them, adjusting the sound controls, and using the mixer to stop everything. We'll also put together some code to create the soundboard buttons; this will draw from our knowledge of lists and dictionaries, and mouse events, that we've covered in past tutorials.

While MP3 is a really popular format for playing music and sounds (no doubt you have thousands of files sitting on a hard drive somewhere), the downside is that it's a proprietary technology. As such, Pygame and other popular libraries don't support MP3 out of the box, perhaps because they can't afford to pay for a licence. Don't worry, though, because we have OGG, an open sound format that your Pi and Pygame will play without any problems at all. All of the sounds for this project are available on GitHub, in OGG and MP3 format, for you to play with. Download the code & sounds: **bit.ly/1J7Ds6m**.

## First off

Just like any Pygame project, there are a couple of things we need to sort out before we can get our hands dirty writing some real code. Lines 1–14 should look really familiar to you by now: first we have our import statements on lines 1–5 , then we set the properties of our windows on lines 6–11, and finally we create a couple of variables for use in our Pygame program a little later on lines 13–17. If you look at line 13, you'll see the **`buttons`** variable; when we're ready to start creating our buttons, we'll append some dictionaries to this list so we can easily keep track of all of the soundboard buttons we create. On the next line, we have our **`stopButton`** dictionary; when we create our stop button, it'll behave much like the rest of the buttons on our soundboard except that it will stop all current sounds playing. Since it's unique, our stop button gets its own variable.

On lines 83–106 we have our familiar old 'main' loop. It's looking a lot smaller than last time: that's because we've broken out all of the code that we could put in main into separate functions. If we didn't, things would




**Above** A basic diagram of how the Pygame audio mixer works

start to get quite messy and hard to follow. By having functions that handle one thing very well, we can write a program that runs well and looks great, too. Just as before, our main loop is responsible for wiping the screen (line 84); handling mouse, keyboard, and system events (lines 88–100); and calling functions to draw in our window.

## Let's mix it up with Pygame mixer

If you're going to use sounds in Pygame, you're more than likely going to be using Pygame's built-in mixer. What is the mixer? Well, you can think of it like its real-world equivalent: the mixer has all sounds across the system (or in our case, game) pass through it. When a sound is in the mixer, it can be adjusted in a variety of ways, volume being one. When our mixer is finished, it passes the sound through to an output, which, in this case, is our speakers. So, before we start loading or playing any sounds, we need to initialise the mixer, exactly the same as when we initialise Pygame before we draw things, and we do that on line 19.

## Our first sound

You can play sounds a couple of different ways in Pygame: you can either play a stream of sound, which you can think of as sound being played as it's being loaded, or you can create and play a sound object, which loads the sound, stores it in our Raspberry Pi's memory, and then plays it. Each way of playing sound is good for different instances. The streaming of sound is better, for example, when we want to create background music that plays as we do other things, whereas the sound object is better for when we want to play short sounds quickly and often.

The sound object fits the bill for our soundboard better than the sound stream, so we'll use those for our buttons a little later on. First we're going to add some ambience to our soundboard with a little background audio from a farm. Background audio usually loops without any sort of user interaction, and streaming audio can be set to loop without too much trouble, so that's what we're going to do. Before we can play any music, we need to load it: on line 20 we point Pygame to our background audio **farm.ogg**; this loads the audio into our mixer, but it won't play straight away. On line 21 we call **`pygame.mixer.music.play(-1)`**, which starts playing our sound file. The number we pass is the number of times we want to our sound to repeat before it stops playing. We've passed -1, which means that it

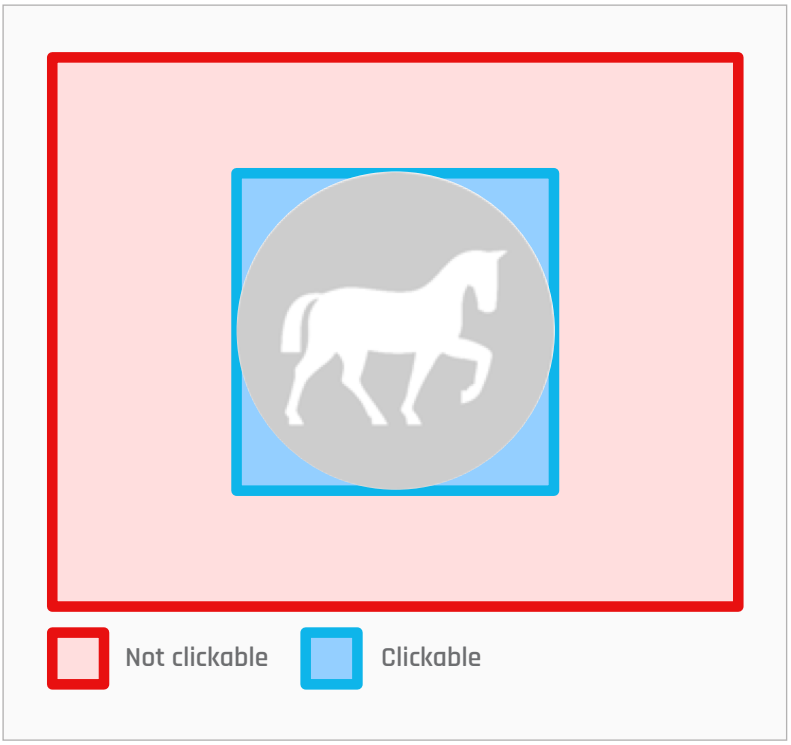
> "The sound object fits the bill for our soundboard better than the sound stream"

will loop forever – until we stop it, that is. If we ran our soundboard at this point, we'd have a lovely big blank window with some calming farm sounds playing, but that's a bit bare. It's time to make some buttons!

> The last property, sound, is just like our image property, except that it loads a sound

## A button here, a button there, buttons EVERYWHERE!

So, how are we going to make these buttons? We could do what we've done in our previous tutorials and draw some shapes and add text to them; that would certainly do the job, but it won't look great. Instead, we're going to make our buttons out of some images your expert has put together for each different animal sound. If you want to take a peek at the buttons before we load them, they're included in the folder **`code/assets/images`**, which you can grab from the GitHub repo. Each button has a silhouette of the animal for which it will make a sound when we click it, but how do we make an image make a sound? Well, we're going to be using lists and dictionaries again. Remember the **`buttons`** variable we looked at right at the start of this tutorial? You can see that it's currently empty, but now it's time to add some dictionaries describing our buttons to it.




**Below** A diagram visualising the imaginary bounding box that surrounds the buttons in our Pygame program

If you look at lines 71–80, you'll see that each line creates a new dictionary for each animal. Each dictionary has three keys (or properties: you can use either term). The first one is **`image`**, which will load the image for that button for us. In previous dictionaries, we've stored strings in dictionaries and then used those strings to load images when we've needed them; this time, however, we've actually loaded each image into our dictionary with **`pygame.image.load()`**. This saves time when we have to draw something many times, and seeing as the image never changes, it makes sense to have it there. Our next key is **`position`**; this is just a simple tuple that contains the x and y coordinates for where our buttons will be drawn. The last property, **`sound`**, is just like our image property, except that it loads a sound instead of an image. Ten points if you saw that coming! Here we're loading the sounds as objects, which means that they're kind of self-contained in how they work. With the background audio we loaded earlier, we passed the data straight through into the mixer and played it through the latter. A sound object has functions that let us control the audio by itself, however. For example, we could call **`sound.play()`** and the sound would play, or we could call **`sound.stop()`**, but it would only apply to the sound we were calling those functions on: if we had two sounds playing at the same time and we stopped only one, the other would keep playing.

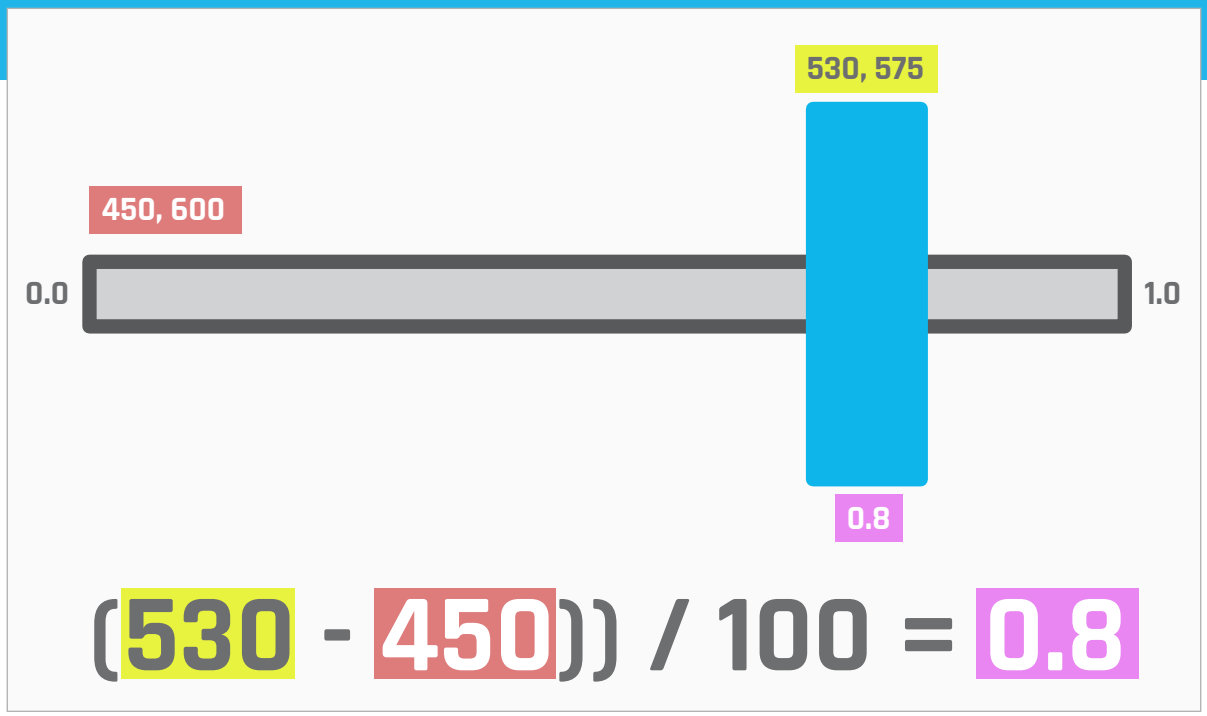
## Drawing our buttons

On lines 71–80, we've added nine different buttons, but if we were to run our program without finishing it, we would still only see a blank white window. This is because we haven't drawn the buttons yet. We've only loaded the necessary bits and bobs to make them work. In our main loop on line 101, we call the function **`drawButtons()`**, which lives on lines 23–28; this will draw the buttons to our surface. Only five lines to draw nine buttons? Yeah, cool isn't it? Because we've done all of the hard work of finding and loading our images and sounds before our main loop could even run, it leaves us with very little to do when we actually have to draw the buttons to our surface. On line 25 we have a **`for`** loop which works through the buttons list we looked right at the start on line 13; for every dictionary it finds in the list, it will draw a button onto our surface, using the properties it finds and a process called blitting. This happens on line 26. To blit or not to blit… what was the question?

Language
>PYTHON

# Sounds.py

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

windowWidth = 600
windowHeight = 650

pygame.init()
surface = pygame.display.set_mode((windowWidth, windowHeight))
pygame.display.set_caption('Soundboard')

buttons = []
stopButton = { "image" : pygame.image.load("assets/images/stop.png"), "position" : (275, 585)}

mousePosition = None
volume = 1.0

pygame.mixer.init()
pygame.mixer.music.load('assets/sounds/OGG/farm.ogg')
pygame.mixer.music.play(-1)

def drawButtons():

  for button in buttons:
    surface.blit(button["image"], button["position"])

  surface.blit(stopButton["image"], stopButton['position'])

def drawVolume():

  pygame.draw.rect(surface, (229, 229, 229), (450, 610, 100, 5))

  volumePosition = (100 / 100) * (volume * 100)

  pygame.draw.rect(surface, (204, 204, 204), (450 + volumePosition, 600, 10, 25))

def handleClick():

  global mousePosition, volume

  for button in buttons:

    buttonSize = button['image'].get_rect().size
    buttonPosition = button['position']

    if mousePosition[0] > buttonPosition[0] and mousePosition[0] < buttonPosition[0] + buttonSize[0]:

      if mousePosition[1] > buttonPosition[1] and mousePosition[1] < buttonPosition[1] + buttonSize[1]:
        button['sound'].set_volume(volume)
        button['sound'].play()

    if mousePosition[0] > stopButton['position'][0] and mousePosition[0] < stopButton['position'][0] + stopButton['image'].get_rect().size[0]:
      if mousePosition[1] > stopButton['position'][1] and mousePosition[1] < stopButton['position'][1] + stopButton['image'].get_rect().size[1]:
        pygame.mixer.stop()

def checkVolume():

  global mousePosition, volume

  if pygame.mouse.get_pressed()[0] == True:

    if mousePosition[1] > 600 and mousePosition[1] < 625:
      if mousePosition[0] > 450 and mousePosition[0] < 550:
        volume = float((mousePosition[0] - 450)) / 100

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

# Create Buttons
buttons.append({ "image" : pygame.image.load("assets/images/sheep.png"), "position" : (25, 25), "sound" : pygame.mixer.Sound('assets/sounds/OGG/sheep.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/rooster.png"), "position" : (225, 25), "sound" : pygame.mixer.Sound('assets/sounds/OGG/rooster.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/pig.png"), "position" : (425, 25), "sound" : pygame.mixer.Sound('assets/sounds/OGG/pig.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/mouse.png"), "position" : (25, 225), "sound" : pygame.mixer.Sound('assets/sounds/OGG/mouse.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/horse.png"), "position" : (225, 225), "sound" : pygame.mixer.Sound('assets/sounds/OGG/horse.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/dog.png"), "position" : (425, 225), "sound" : pygame.mixer.Sound('assets/sounds/OGG/dog.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/cow.png"), "position" : (25, 425), "sound" : pygame.mixer.Sound('assets/sounds/OGG/cow.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/chicken.png"), "position" : (225, 425), "sound" : pygame.mixer.Sound('assets/sounds/OGG/chicken.ogg')})
buttons.append({ "image" : pygame.image.load("assets/images/cat.png"), "position" : (425, 425), "sound" : pygame.mixer.Sound('assets/sounds/OGG/cat.ogg')})

# 'main' loop
while True:

  surface.fill((255,255,255))

  mousePosition = pygame.mouse.get_pos()

  for event in GAME_EVENTS.get():

    if event.type == pygame.KEYDOWN:

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

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

    if event.type == pygame.MOUSEBUTTONUP:
      handleClick()

  drawButtons()
  checkVolume()
  drawVolume()

  pygame.display.update()
```




**Right** An illustration of the equation used to control the volume

Blitting is something you might have come across in the past, but don't worry if you haven't: it's basically a fancy way of saying 'paste', and we used it in our last tutorial to draw the start and finish screens for our drop game. When we blit something, we take the pixels of our surface and then we change the pixels so that they're the same as the image we're adding. This means that anything that was beneath the area being blitted is lost. It's a lot like cutting letters out of a newspaper and supergluing them onto a piece of paper: whatever was beneath the newspaper clipping is gone forever, but we can see what we cut out of the newspaper just fine.

> "Buttons are one of those things that can be really simple or really tricky"

## Clicking our buttons

Now that we've got a soundboard that has buttons, we need to make those buttons do something. Buttons are one of those things that can be really simple or really tricky: some systems do a lot of the work for you, whereas others don't do so much. Unfortunately, Pygame is one of the latter systems, but that doesn't matter: we can write the code for our buttons ourselves. On lines 89–100 we have some code which handles some of the events that happen in Pygame. You'll recognise the code on lines 89–97: it's the same code that we've used to quit Pygame for the last four tutorials, but on lines 99–101 we're looking for a `MOUSEBUTTONUP` event. Why 'MOUSEBUTTONUP' and not something like 'CLICK'? Well, for a mouse button to go up, it has to have gone down; that means it's been clicked, so, same difference. If the mouse has been clicked, we call the `handleClick()` function, which is on lines 38–55. Just like when we drew our buttons, we're going to work through the buttons list to find out where they are on our surface. If our mouse clicked where a button is, we'll play that sound, otherwise we'll do nothing.

In order to check whether or not a button was clicked, we need to know three things: 1) the position of each button, 2) the size of that button, and 3) where the mouse was when it was clicked. If our mouse coordinates are greater than the x and y coordinates of the button image, but less than the x and y coordinates plus the width and height of the image, then we can be sure that the button we're checking against was clicked and we can therefore play the sound for that button; otherwise, the mouse was outside the button. Checking this way is a little bit of a cheat: our buttons are circles, but we check whether or not a click has happened within a square that surrounds the button. We do this because the result is almost exactly the same and the code to check for a click in a square is much quicker than that for checking a circle or irregular shape. This square is often referred to as a bounding box, and it's often used to check for clicks.

The checks happen on lines 47 and 49. If either statement is found to be False, then nothing will happen, but if both are correct, then we play the

sound with line 50. Remember, this is a sound object, not a sound stream, so when we play this sound, it gets played through the mixer, as that's where all sounds pass through to play. Note that the mixer has no control over that specific sound, however, because it plays in its own separate channel.

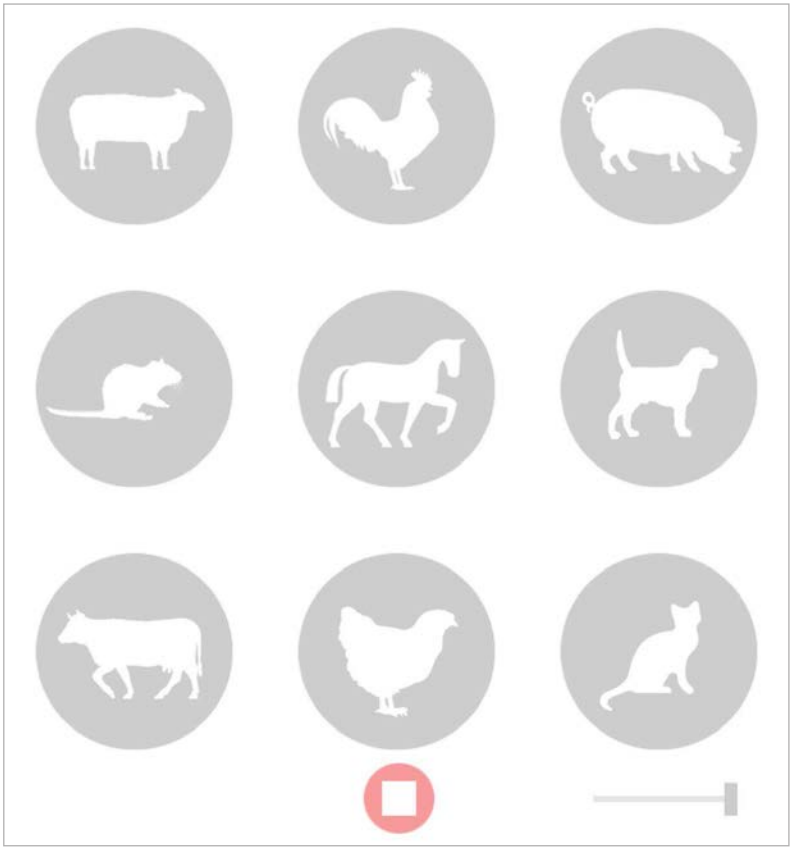
Having said that, we can control certain aspects of the sounds with our mixer. For example, we can pause the playback of all sound, or stop it altogether, which leads us nicely onto...

## Stopping all of our sounds

Now we have sounds coming at us left, right, and centre! It's like a farmyard has found its way into your Raspberry Pi, and that's what we're going for, but good heavens! Who would have thought that these creatures would be so noisy? If only we could make them stop. Well, we've got a nice friendly button to do that for us. On line 14 we have a dictionary called **`stopButton`**; unlike our soundboard buttons, it doesn't have a sound, just an image and a position element. That makes it special. Beneath all of the code used to handle our sound buttons, we have some code that only deals with the stop button: on line 28 we draw the stop button, just after we've drawn all of the sound ones, and on lines 53–55 we specifically check whether or not it's the stop button that has been clicked. Why do we give the stop button special treatment? Because it's unique, and for every button that doesn't do the same thing as all of the other buttons, we need to write custom code. Sure, we could use dictionaries and lists like we've done for our sound buttons, but that's far more complicated than is needed for right now: a lesson for another tutorial, perhaps!

## IT'S LOUD! Oh... it's quiet now...

So, we've loaded sounds, played them, and stopped them dead, but what if we just wanted to make them a little quieter? Well, that's simple enough. Each of our sound objects has a **`set_volume()`** function which can be passed a value between 0.0 and 1.0. 0.0 is mute, whereas 1.0 is full volume. If you pass a value larger than 1.0, it will become 1.0, and if you pass a value less than 0.0, it will become 0.0. First things first, let's make a volume slider. On lines 30–36 we draw two rectangles. The first rectangle represents the range of 0.0 to 1.0, and the second rectangle is an indicator of the current volume. When we first start our soundboard, the volume (set on line 17) is set at 1.0, so our indicator is all the way to the right, but that doesn't do us much good if we want to turn things down. Just before we call **`drawVolume()`** on line 104, we call **`checkVolume()`** on line 103. Here we look at the current position of the mouse and whether or not the left mouse button is held down. If it is, our user is likely trying to drag our indicator to the level they want the sound to be at. So we work out where the mouse is on between 0.0 and 1.0 on our indicator and set the volume to the new level. Then, when our **`drawVolume`** function is called, the indicator will be drawn at the correct position.



**Above** A screenshot of our finished soundboard

Now, when we next click a sound, it will be set to the level we've chosen with the **`set_volume()`** function on our sound object, which you can see on line 50.

## That's all, folks

And that's it. We've learned everything we need to know about making sounds play in our game. We've covered background audio, the sound mixer, streaming sound, and sound objects. We've also used lists and dictionaries to create and manipulate buttons. Aren't we lucky that we learned those last time? Now, we have a fully functioning soundboard that you can use to drive your friends quackers... wait, we don't have a duck. Oh, never mind.

If you want a challenge, see if you can write code using what you've learned in the past to trigger the animal sounds with the keys 1–9.

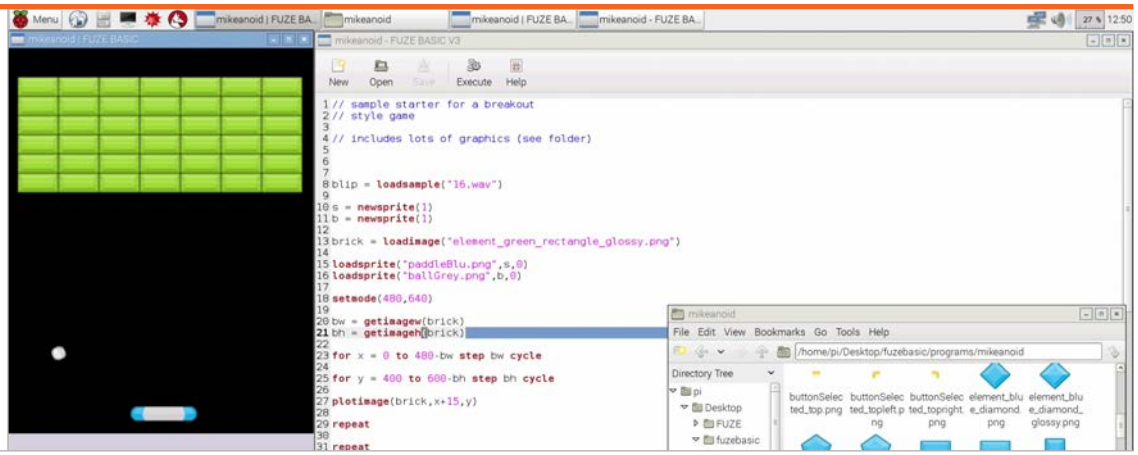
fuze.co.uk

FREE

**Maker Says**

"Ideal platform for learning and teaching computational thinking"

**Fuze**



# FUZE BASIC V3

Just like **Lucy Hattersley,** a whole generation of coders cut their teeth on BASIC; follow in their footsteps with FUZE BASIC V3...

Few things divide the programming world as much as BASIC (Beginner's All-purpose Symbolic Instruction Code). Once a standard inclusion with all home computers, BASIC was the first language an entire generation of programmers discovered.

FUZE BASIC has quietly earned its reputation as the best version of BASIC for the Raspberry Pi. Part of this success can be put down to sales of the FUZE Workstation to schools (you don't need one to run FUZE BASIC). This provides a huge range of high-quality support materials: Project Workbooks, Reference Guides and Project Cards, all available as free downloads.

## Installing FUZE BASIC V3

In previous versions, FUZE BASIC was installed using a preconfigured boot image (based on Raspbian). Now it is installed as a separate download. We did have to dive into the Advanced Options using `sudo raspi-config` and enable I²C support to get it to work, though, but a preconfigured boot image for newcomers is said to be on the way.

There's a lot to discover in the latest version. It includes new sprite handling tools, enabling rotation, size, and transparency. You can also import, rotate, and scale images, and new audio tools enable music playback and up to four channels of sound effects. These join a stack of comprehensive functions that make programming more fun. FUZE BASIC can control a Maplin USB Robot, draw on-screen graphics, and manage GPIO.

## Is FUZE good for you?

FUZE BASIC V3 comes with a text editor and we found it ran programs windowed by default. So it feels more up-to-date than many versions of BASIC. Mind you, it still starts with a command line (known as Direct Mode) where you can use line numbers and good old-fashioned commands like LIST and RUN. Nostalgia aside, this throwback is faintly ridiculous in the modern world. The mere presence of line numbers and, God forbid, the GOTO function is enough to make most programmers shudder.

While you can define procedures, this isn't an object-orientated programming (OOP) language. The argument that children should move from Scratch directly to another OOP language carries some water. But it could be argued the leap between Scratch and Python is too big for many newcomers, and FUZE BASIC is a great intermediate step that's fun to use.

**Related**

**ARM BBC BASIC V**

ARM BBC BASIC V is included with RISC OS (an option in NOOBS). While it's not as feature-rich as FUZE BASIC V3, plenty of manuals and guides are available.


FREE

riscosopen.org

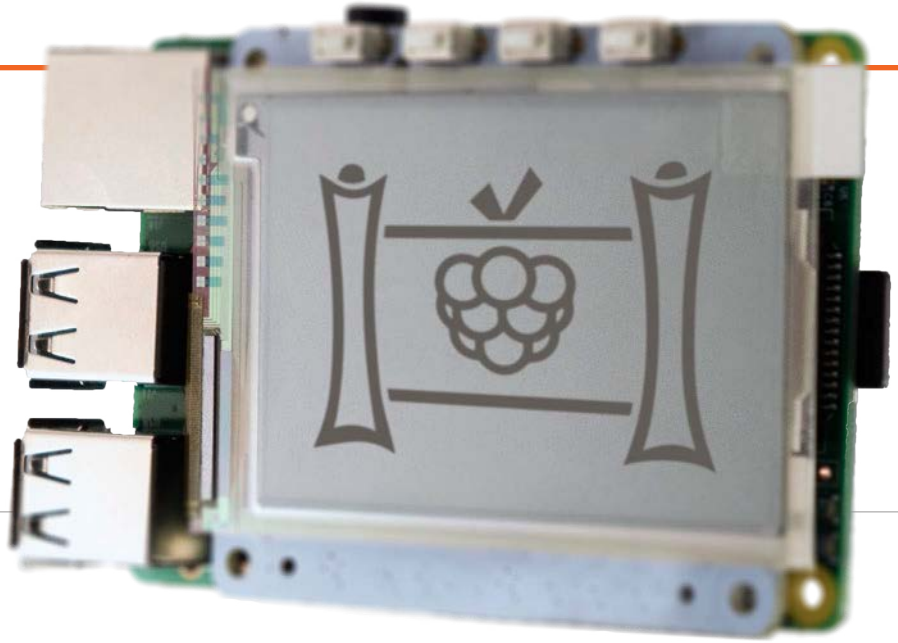
**Last word**

**The new sprite and sound functions enhance an already creative learning platform, but it's the wealth of support materials that really make it special. Shame you have to sign up to download it, though.**

★★★★☆

papirus.ws

**From £30 / $45**



## Maker Says

"Add low-power display tech to your project today!"
**PiSupply**

# PI SUPPLY PAPIRUS

**Les Pounder** is keen to build his own eReader with this low-power E Ink display HAT, but can he?

E Ink screens are an attractive proposition for single-board computers, but the biggest issue generally faced by users is the rather cumbersome array of breakout boards required. To solve this, Pi Supply has unleashed its latest E Ink add-on board, which uses the HAT (Hardware Attached on Top) standard.

PaPiRus is an E Ink display and controller board that has been designed to fit seamlessly onto the 40-pin GPIO found on the Model A+, B+ and Pi 2. It offers a standard connector, to which a range of different-sized E Ink displays may be fitted. For our review, we opted for the largest version, which has a diagonal size of 2.7 inches. Attaching the screen to the board is simple, and uses a latch mechanism very similar to that of the official Raspberry Pi Camera Module. On the reverse you'll find that a battery is fitted, to enable real-time clock functionality.

## Software situation

Currently, the software installation is not for the faint-hearted, and there were times when the method wasn't easy to discern. However, with some assistance, we were able to install the necessary software and run the demos to test functionality.

The demos supplied reveal how to use the various functions of the PaPiRus and we were pleased with the inclusion of a temperature sensor and real-time clock, enabling our Pi to retain the correct time without network connectivity. One of the most powerful demos is `ImageDemo.py`, which uses PIL, a Python image-processing library, to convert images to display on the screen.

The hardware itself is very well developed and built, demonstrating the great care and attention to detail paid by the development team. The software is another matter, but is still a work in progress. We have been assured that the installation and demos will be ready for everyone to hack easily when the final versions are released.

## Last word

**PaPiRus is a great hardware platform, fitting neatly upon the A+ and creating a small footprint for projects. The software is still in development and will hopefully improve with the help of the community.**

★★★☆☆

## Related

### EMBEDDED ARTISTS 2.7-INCH DISPLAY

An inexpensive development board for those who wish to tinker with E Ink displays. It's not as neat as PaPiRus.

£21 / $ 32
bit.ly/1eXGaRN

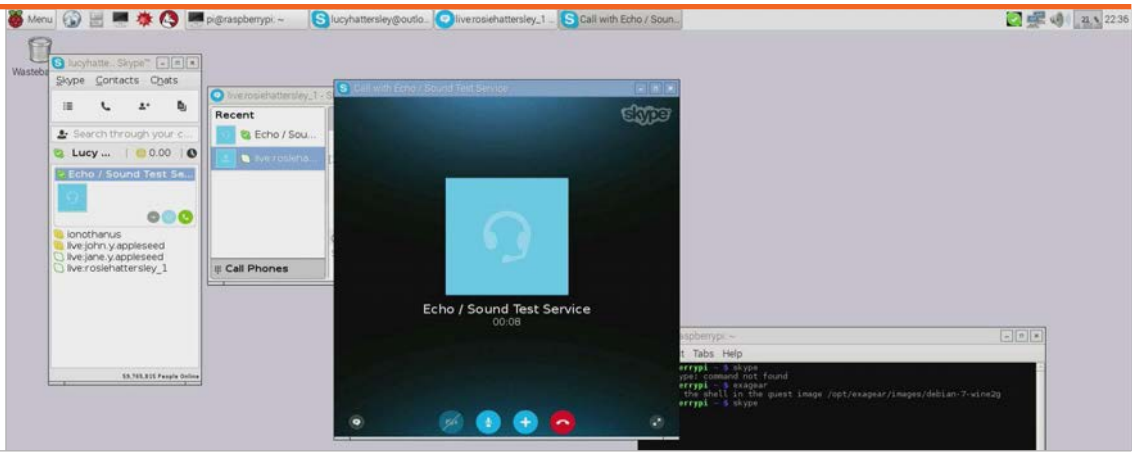
eltechs.com

**£16 / $25**

## Maker Says

"Like QEMU, but five times faster"

**eltechs.com**



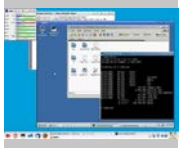

# EXAGEAR DESKTOP
## FOR RASPBERRY PI 2

Access a huge range of additional Linux (and even Windows) software using the Intel x86 virtualisation program

## Related

### QEMU

QEMU is a generic open-source virtualiser that runs software from one board on another. It can be used to install x86 software on the ARM-based Raspberry Pi board, but it's usually used the other way around (to emulate a Raspberry Pi on a larger Linux machine).



FREE

qemu.org

While there's no shortage of great software for the ARM-based Raspberry Pi, most Linux programs are compiled to run on Intel x86 processors.

This is where virtualisation software steps in. It sits between an x86 application and the ARM version of Linux and translates the x86 code to run on your Raspberry Pi.

ExaGear Desktop creates a second system known as the 'guest' system. Once installed, you can switch between the guest and your regular ('host') system using the exagear and exit commands. Inside the guest system, apt-get and dpkg are used to install Intel x86 software. Tantalisingly, one software option is Wine, another layer application that is used to run Windows apps inside the Linux environment.

## Setting up ExaGear

ExaGear includes a handy script for automatic installation, but we had to read through the manual instructions to discover which of the Debian packages to install. The choice varies depending on whether you have a 2G/2G or 3G/1G split between your user and kernel memory.

We followed the instructions to install nginx server software from our Raspberry Pi. After that, we downloaded and installed Skype inside the LXDE desktop. We then got Eclipse 3.8 up and running (albeit in a sluggish manner).

## Performance

Performance takes a hit when moving from the host to the guest system, but it's not as bad as you'd imagine. Even the more resource-intensive apps we tested, like Skype, ran perfectly well.

We installed LibreOffice in the host environment. LibreOffice took 20 seconds to launch in the host environment, and 52 seconds in the guest environment.

Where things started to go awry was in attempting to run 32-bit Windows software via Wine. There were a lot of complications. You have to set up Raspbian with a 3G/1G memory split, then reinstall ExaGear with the right package. Then you need to install Wine (which required refreshing all the dependencies). After all this effort, it was surprisingly tricky to find compatible software; Wine compatibility documentation gets patchier the further back in time you go, and 32-bit support at the Raspberry Pi level (through two layers of middleware) means you can only run very old Windows software.

In all, we found ExaGear Desktop provided no practical use for Windows software, but was fantastic for running Linux x86 programs.

## Last word

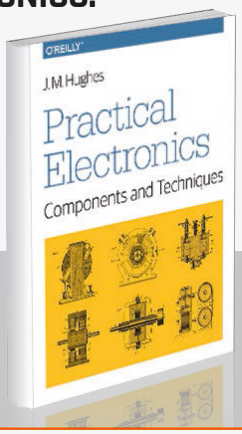
**Once ExaGear Desktop is up and running, it becomes a neat solution for running a wider range of Linux software on your Raspberry Pi. The Windows aspect is interesting, but not practical.**

★★★★☆

# RASPBERRY PI BESTSELLERS ELECTRONICS

**Go beyond following instructions for attaching components to your GPIO pins, and learn the nitty-gritty of electronic components...**

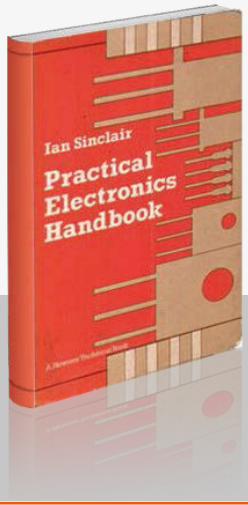
## PRACTICAL ELECTRONICS: COMPONENTS & TECHNIQUES




**Author:** John M Hughes
**Publisher:** O'Reilly
**Price:** £26.50
**ISBN:** 978-1449373078
**oreil.ly/1ICmbEM**

Practically oriented book for those who want to get making electronic projects. Comprehensive, but aimed squarely within the reach of the beginner.

## PRACTICAL ELECTRONICS HANDBOOK




**Authors:** Ian Sinclair
**Publisher:** Newnes
**Price:** Free Download
**ISBN:** n/a
**bit.ly/1QJ3BKf**

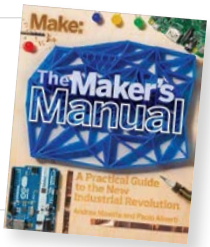
The 1980 first edition, freely available to download from the Internet Archive, is a great intro to electronics – but the latest edition (with John Dunton) covers newer components and circuits, and is a strong recommendation.

## ELECTRONIC CIRCUITS: 4TH EDITION


**Authors:** Mike Tooley
**Publisher:** Routledge
**Price:** £26.99
**ISBN:** 978-1138828926
**key2electronics.com**

Welcome update to Tooley's well-regarded further education text: very good on the theories of circuits, with a companion website that helps you design them. Now with a section on Raspberry Pi.

# MAKE: THE MAKER'S MANUAL




**Author:** Paolo Aliverti & Andrea Maietta
**Publisher:** Maker Media
**Price:** £16.50
**ISBN:** 978-1457185922
**themakersmanual.com**

"A revolution is happening: the manufacture of objects is shifting from big companies... to individuals, producing a previously unseen variety in things we make." That revolution may be at an early stage, but with the easy availability of low-cost boards like the Raspberry Pi, and manufacturing kit like 3D printers (often available as a shared resource at fab labs and makerspaces), all you need to get involved is an idea and a little knowledge.

*The Maker's Manual* doesn't just set out to provide that knowledge – which it does both with a concise history of the maker movement, and a roundup of practical tools from laser cutters to GitHub – but also how to nurture and grow an idea. The second section of the book looks at the mental skills involved in creativity and how they can be developed; the practical side of managing a project and running a business; and the soft skills of collaboration.

Chapters on basic electronics, Arduino and the Pi provide useful, concise introductions, while a look at the Processing language gets straight down to the artistic possibilities released by its random function. With even the short Internet of Things chapter containing a practical Pi example, there's little fluff and a lot of value packed into this manual's 200 pages.

**Score** ★★★★★

# RASPBERRY PI PROJECTS FOR KIDS


**Author:** Daniel Bates
**Publisher:** Packt
**Price:** £16.99
**ISBN:** 978-1785281525
**bit.ly/1MFA5EI**

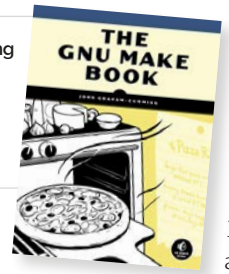
"The Raspberry Pi exposes programming software to make it as easy as possible to get started," says Daniel Bates in his preface. After a chapter on getting set up with the Pi, the book launches into trying out programming through six mini projects, starting with the excellent Scratch visual programming language: first to do some animation, with programming concepts – and the power and freedom of programming – introduced as a natural part of learning the animation procedure; then an *Angry Birds* clone, which includes physics (gravity and bouncing), scoring, and a chance to add in your own extensions.

Python is next, with a random insults generator – always a popular bit of fun – which introduces list handling and functions. Then a connection to the physical world: first through an excellent DIY game controller for a speed test game, then with Google maps and Tkinter to look at your neighbourhood. Lastly, there's an introduction to music making with software, using Sonic Pi.

All in all, this software-focused introduction gives a quicker starting path than many of the more maker-oriented Pi books, which themselves would make an ideal follow-on. After working through Bates's readable and friendly guide, any child will know a surprisingly large amount of core programming skills, as well as gain a thirst for more. Recommended.

**Score** ★★★★★

# THE GNU MAKE BOOK




**Author:** John Graham-Cumming
**Publisher:** No Starch
**Price:** £23.50
**ISBN:** 978-1593276492
**nostarch.com/gnumake**

Not the O'Reilly imprint, the *other* Make! Yes, GNU Make is something most of us only come across when we download a program in source form, and have to untar then run the `./configure && make && make install` incantation to compile and install the software. GNU Make is the build automation tool heroically doing the hard work in the background.

Readers may know of John Graham-Cumming from his successful petition to the UK government asking for an apology for its persecution of Alan Turing. Here is a success on a smaller stage – but no less vital, as many people have started out using Make, and writing makefiles, for their own projects, then run up against some hurdle. This is not a beginner's book, but for anyone already using Make in their projects, it's a really well-written guide to getting more from it.

The author gained his Make knowledge writing a complete clone of the software in C++ – not a unique project, as some readers may know the Python Snakemake, designed for complicated workflows in bioinformatics – and he has a comprehensive knowledge of GNU Make. This is most usefully reflected in the excellent chapter on 'Pitfalls and Problems' – essential reading for anyone whose project, and consequently makefile, has grown large enough to run into difficulties.

**Score** ★★★★☆

# AUTOMATE THE BORING STUFF WITH PYTHON


**Author:** Al Sweigart
**Publisher:** No Starch
**Price:** £19.99
**ISBN:** 978-1593275990
**automatetheboringstuff.com**

If you've ever found yourself repetitively carrying out the same task over and over, wishing you had a little more command-line experience so that you know the correct Bash magic to get it all done in one go from the terminal, this is your chance to leapfrog past that and learn how to use Python to quickly write programs to take on the tedious tasks.

The first section introduces Python and programming basics, such as flow control, functions, lists, and string manipulation, with practice questions and useful projects rounding off each chapter. The main part of the book is a dozen chapters dedicated to problem areas where a little Python code will smooth away your problems: organising files, getting data from websites, working with text from inside PDF and Word documents, batch-editing images, and even automating when and how programs run on your PC.

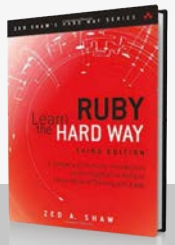
Sweigart is an able teacher with a number of well-regarded Python books to his name, and it shows through his clear teaching style and well-paced lessons. Not only are these skills which would benefit any regular computer user, but after working through Sweigart's book, non-coders will find themselves competent and practical beginner programmers.

**Score** ★★★★★

# ESSENTIAL READING: RUBY

**There are many ways to learn and use Ruby's expressive power: here are five of the best recent tutorials...**

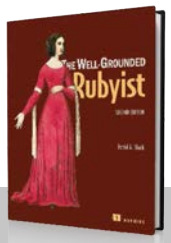
## Learn Ruby the Hard Way




**Author:** Zed A Shaw
**Publisher:** Addison Wesley
**Price:** £24.99
**ISBN:** 978-0321884992
**learnrubythehardway.org**

Great intro for the new programmer: a little 'Pythonic', but strongly practical, preparing you for real-world coding.

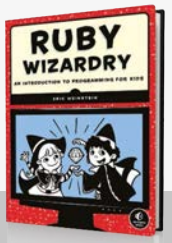
## The Well-Grounded Rubyist Second Edition




**Author:** David A Black
**Publisher:** Manning
**Price:** £27.99
**ISBN:** 978-1617291692
**manning.com/black3**

Introduction and reference with a strongly object-oriented approach to the language; ideal for Pythonistas new to Ruby.

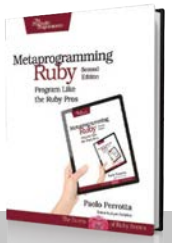
## Ruby Wizardry: An Introduction to Programming for Kids




**Author:** Eric Weinstein
**Publisher:** No Starch
**Price:** £19.99
**ISBN:** 978-1593275662
**nostarch.com/rubywizardry**

While the story pulls children in, serious programming concepts are painlessly transmitted by Codecademy author Eric Weinstein: wizardry indeed!

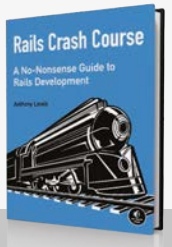
## Metaprogramming Ruby 2: Program Like the Ruby Pros




**Author:** Paolo Perrotta
**Publisher:** Pragmatic
**Price:** £25.50
**ISBN:** 978-1941222126
**oreil.ly/1e0lhnN**

The next step: gain real insight into Ruby through writing code that writes code. Practical and thought-provoking.

## Rails Crash Course A No-Nonsense Guide to Rails Development




**Author:** Anthony Lewis
**Publisher:** No Starch
**Price:** £23.50
**ISBN:** 978-1593275723
**nostarch.com/railscrashcourse**

We couldn't leave out Rails, and Lewis's new guide is concise, yet lacking nothing you need to get going.

# 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 RECURSION 2015

**When:** Saturday 4 July
**Where:** King Edward VI School, Stratford-upon-Avon, UK
**bit.ly/1Jif2Kh**
A packed day dedicated to computing and computer science. Admission to Recursion is free and doors open at 11am.

### 2 PRESTON RASPBERRY JAM

**When:** Monday 6 July
**Where:** Media Innovation Studio, Media Factory Building, Preston, UK
**bit.ly/1J7KfOu**
Join in for a day of fun lightning talks, demonstrations, and hands-on time with Raspberry Pi.

### 3 TORBAY TECH JAM

**When:** Saturday 11 July
**Where:** Paignton Library and Information Centre, Paignton, UK
**torbaytechjam.org.uk**
A fun, informal day of hacking: bring your own kit or use some of the equipment provided.

### 4 RASPBERRY JAM SILICON VALLEY

**When:** Saturday 18 July
**Where:** Computer History Museum, Mountain View, USA
**bit.ly/1IKPPZ1**
This Jam takes place on the third Saturday of every month. Follow on Twitter: @RaspberryJamSV

### 5 RASPBERRY JAM BERLIN

**When:** Saturday 25 July
**Where:** Prenzlauer Allee 242, Berlin, Germany
**raspberryjamberlin.de**
Germany's premier Raspberry Jam event, which is held in excellent facilities provided by Fab Lab Berlin.

### 6 CARMARTHEN CODER DOJO

**When:** Saturday 25 July
**Where:** Carmarthen Library, Carmarthen, Wales
**bit.ly/1e366L4**
Come and see technology and coding in action! There will be a range of talks and stands, and plenty of demos.


## 7 LEEDS RASPBERRY JAM

**When:** Saturday 25 July
**Where:** Digital Garage from Google, Leeds, UK
**bit.ly/1HdD8pc**

Come along to Digital Garage in Leeds to join in with hands-on activities in partnership with JamPacked.

## 8 RHÔNE VALLEY RASPBERRY JAM

**When:** Sunday 2 August
**Where:** Foyer Laique, Courthézon, France
**bit.ly/1dfy71k**

Bring your Raspberry Pi along to learn about exciting projects and get the inspiration to build your own.

# DON'T MISS: RECURSION 2015

**When:** Saturday 4 July **Where:** King Edward VI School, Stratford-upon-Avon, UK

King Edward VI School in Stratford-upon-Avon plans host to another Recursion, a packed day of computing and computer science. Recursion is free to exhibitors and visitors, and is made possible with the support of the Pound-A-Day-Scheme. Visit the website for a full rundown of exhibitors and participating schools, and don't forget to book in for a workshop. For more information about the event, visit **recursioncomputerfair.co.uk.**

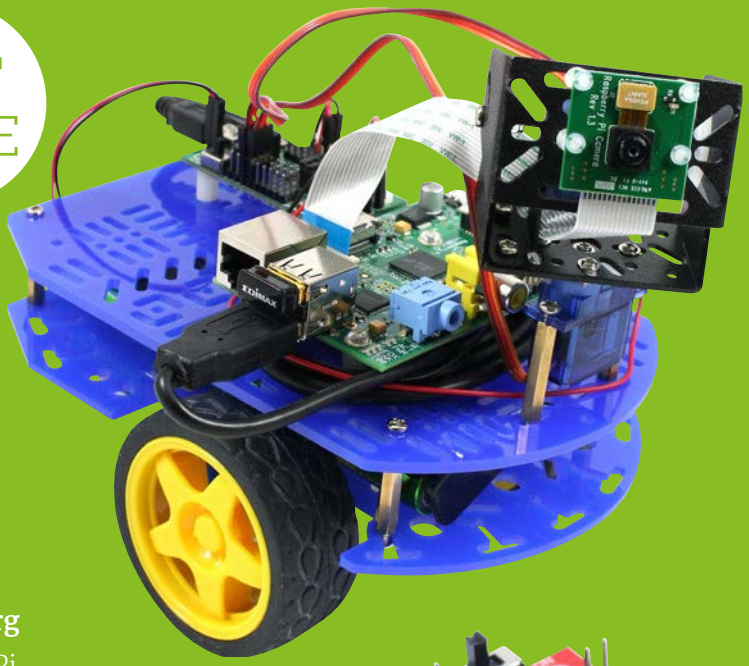
In association with:
DAWN ROBOTICS
www.dawnrobotics.co.uk

# A RASPBERRY PI ROBOT & 4 MOTOR DRIVERS MUST BE WON!

# WHY DO YOU HACK AND MAKE WITH RASPBERRY PI?

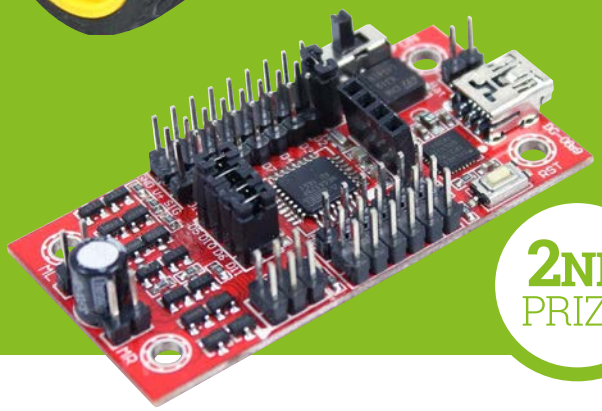
**Tell us by 27 July 2015 for a chance to win!**

1ST PRIZE



## How to enter:

Simply email **competition@raspberrypi.org** and let us know why you use the Raspberry Pi in your hacking and making projects.
One randomly selected entry will win a £150 Raspberry Pi Camera robot, and four runners-up will win a Dagu Mini Driver MkII board worth £14.



2ND PRIZE

### Terms & Conditions

Competition closes 27 July 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.

**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.

# MAKER MOTIVATION

While at Maker Faire, **Matt Richardson** learns about top-secret plans one boy has for his Raspberry Pi...

It's quite likely you're already aware of the Maker Movement. Thanks to inexpensive accessible technology and the culture of sharing knowledge on the internet, people are making the things they want rather than waiting for a company to manufacture those things. Although that's not the only possible motivation for a maker to create things, it's certainly important one. In a time where people line up for the release of a new cell phone – something I'm guilty of doing – it's nice to see that individuals are now more than ever empowered to innovate.

The spirit of the Maker Movement is almost tangible at Maker Faires all over the world, where people get together to 'show what they make and share what they learn', to paraphrase the inscription on wristbands that all makers and crew wore at a Maker Faire a few years ago. It's a great place to better understand all the possible motivations for making.

## Hobby electronics & microcontrollers

Maker Faire is very familiar territory for me. The first one I attended was in New York City in 2010 and it was a big moment for me. Around that time, I was just starting to explore the realm of hobby electronics, microcontrollers, and fabrication. I was participating in online maker communities within Twitter and YouTube, but being at Maker Faire was much different. I was able to meet makers face to face. I was able to see projects, products, and presentations. There's an interesting blend of creativity and technology among makers and it's especially evident at Maker Faire. Within a few hours at my first Maker Faire, I knew I had found my tribe.

Since then, I've attended many Maker Faires, including the big one in the Bay Area. My Maker Faire strategy usually centres around the goal of seeing as much of the Faire as I possibly can. This year's Maker Faire Bay Area, however, was different. With my new position at Raspberry Pi, I had a singular focus at Maker Faire: to spread the word about our favourite single-board computer.

## Keep up the good work

Nine of us took responsibility for a 10 foot by 20 foot piece of floor space in the enormous Expo Hall at the San Mateo County Event Center. We laid out literature and stickers, and offered a hands-on activity for kids. Even before the gates opened, our booth was busy. During peak hours, we were absolutely swamped. Many visitors stopped by just to say "keep up the good work" and other visitors learned about Raspberry Pi for the first time.

One particularly amusing interaction still sticks out in my mind. On one table, we spread out project 'recipe cards', which visitors could take as a reminder of a project that they could find online and make themselves with Raspberry Pi. A father and his young son were at the table and while I explained some of the details of Raspberry Pi to the father, the son selected a card to take home. As they were getting ready to go, I asked the son about the project he wanted to work on.

The son said to me in a false whisper, "the P. D." followed by an exaggerated wink. At first I was confused; I had no clue what he was talking about. It was clear he didn't want his dad to know about this project. Then it hit me: one of the recipe cards is for a Parent Detector project. It uses a motion sensor to trigger the camera to record video, capturing intruders' actions. I gave the boy a thumbs-up. The father smiled and shrugged, presumably glad that his son had found a top-secret motivation to be a maker.

# 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

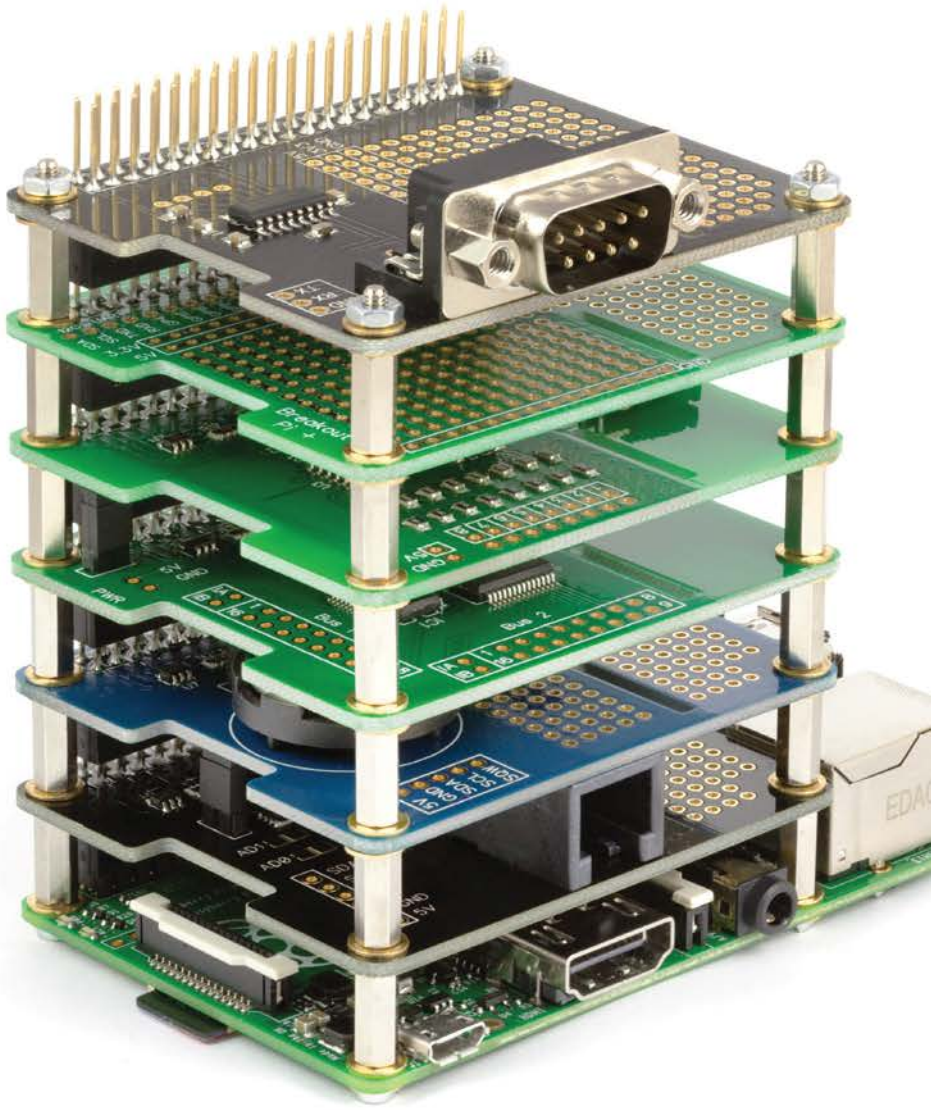
32 digital 5V inputs or outputs. I²C address selection allows you to stack up to 4 IO Pi Plus boards on your Raspberry Pi.

### 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