

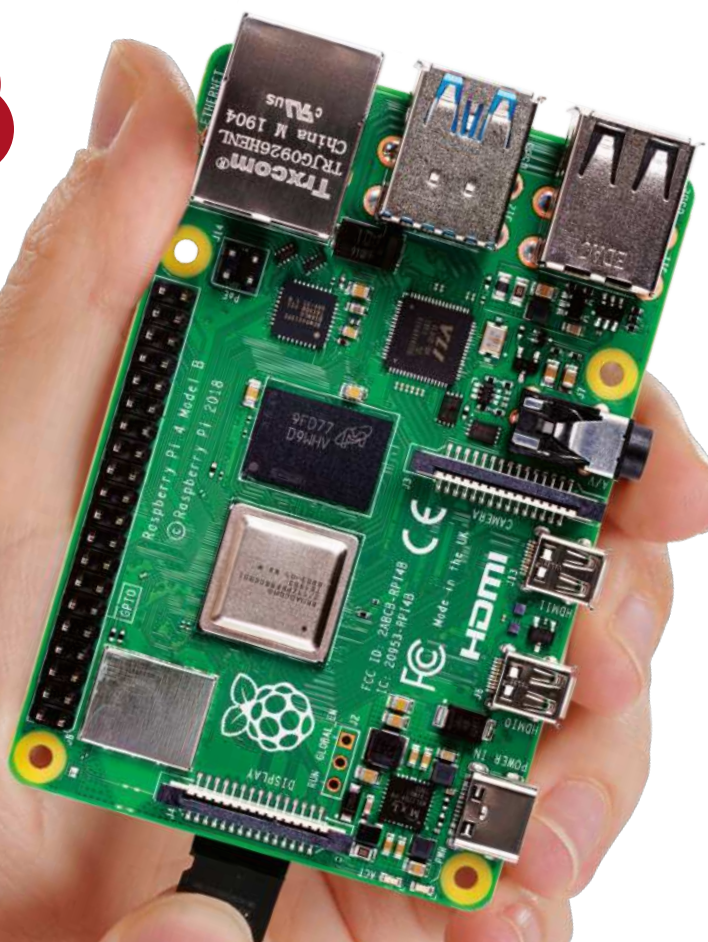
NEW

Raspberry Pi

The Complete Manual

The independent handbook for all Raspberry Pi users

25+
projects
inside



Digital
Edition



TWENTY-
THIRD
EDITION

100% UNOFFICIAL

Welcome to **Raspberry Pi** The Complete Manual

The Raspberry Pi is one of the most exciting things to happen to computer technology in recent years.

As an educational tool, this tiny PC has reignited interest in bare-metal computing in schools and homes all over the world. As a platform for open-source software, it has also inspired millions of people to try Linux – many for the first time. Most exciting of all is the potential to incorporate the device into practical projects, as demonstrated by the tutorials in this new edition of Raspberry Pi The Complete Manual. Grab your Pi and start creating!



「 FUTURE 」

Raspberry Pi

The Complete Manual

Future PLC Quay House, The Ambury, Bath, BA1 1UA

Editorial

Compiled by **April Madden & Adam Markiewicz**

Features Editor **Aiden Dalby**

Senior Art Editor **Andy Downes**

Head of Art & Design **Greg Whitaker**

Editorial Director **Jon White**

Photography

James Sheppard

All copyrights and trademarks are recognised and respected

Advertising

Media packs are available on request

Commercial Director **Clare Dove**

International

Head of Print Licensing **Rachel Shaw**

licensing@futurenet.com

www.futurecontenthub.com

Circulation

Head of Newstrade **Tim Mathers**

Production

Head of Production **Mark Constance**

Production Project Manager **Matthew Eglinton**

Advertising Production Manager **Joanne Crosby**

Digital Editions Controller **Jason Hudson**

Production Managers **Keely Miller, Nola Cokely,**

Vivienne Calvert, Fran Twentyman

Printed by William Gibbons, 26 Planetary Road,

Willenhall, West Midlands, WV13 3XT

Distributed by Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU

www.marketforce.co.uk Tel: 0203 787 9001

Raspberry Pi is a trademark of the Raspberry Pi Foundation

Raspberry Pi The Complete Manual Twenty Third Edition (CMB4200)

© 2022 Future Publishing Limited

We are committed to only using magazine paper which is derived from responsibly managed, certified forestry and chlorine-free manufacture. The paper in this bookazine was sourced and produced from sustainable managed forests, conforming to strict environmental and socioeconomic standards. The paper holds full FSC or PEFC certification and accreditation.

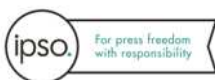
All contents © 2022 Future Publishing Limited or published under licence. All rights reserved. No part of this magazine may be used, stored, transmitted or reproduced in any way without the prior written permission of the publisher. Future Publishing Limited (company number 2008885) is registered in England and Wales. Registered office: Quay House, The Ambury, Bath BA1 1UA. All information contained in this publication is for information only and is, as far as we are aware, correct at the time of going to press. Future cannot accept any responsibility for errors or inaccuracies in such information. You are advised to contact manufacturers and retailers directly with regard to the price of products/services referred to in this publication. Apps and websites mentioned in this publication are not under our control. We are not responsible for their contents or any other changes or updates to them. This magazine is fully independent and not affiliated in any way with the companies mentioned herein.



Future plc is a public
company quoted on the
London Stock Exchange
(symbol: FUTR)
www.futureplc.com

Chief executive: **Zillah Byng-Thorne**
Non-executive chairman: **Richard Huntingford**
Chief financial officer: **Penny Ladkin-Brand**

Tel +44 (0)1225 442 244



Contents

What you can find inside the bookazine

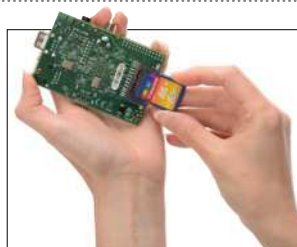
Getting started



- | | |
|-------------------------------------------------------------|-----------------------------------------------------------------------|
| 8 Raspberry Pi models
Meet the Pi 4 Model B | 26 Get Pixel desktop
The desktop environment |
| 12 The starter kit
What you need for your Pi | 30 Master the Config tool
How to tweak your settings |
| 14 Set up your Pi
Configure your new PC | 32 Get online
Access a world of apps |
| 16 Set up your Pi Zero
Start up your tiny Pi | 34 Install & use packages
How to use apt-get |
| 20 Install a distro
Get your new OS running | 36 Use graphical installations
Install & remove graphically |
| 22 Command line basics
Learn essential new skills | 38 GPIO explained
Get to grips with GPIO pins |

The projects

- | |
|------------------------------------------------------------------------------|
| 40 Get to grips with the Enviro pHAT
Take weather-related readings |
| 42 Back up your Pi
Never lose a file again |
| 44 Beginner's guide to nano
Edit text from the CLI |
| 46 Remote desktop access
Use Raspberry Pi OS anywhere |
| 48 Raspberry Pi plus Arduinos
How to use your Arduino |
| 52 Program with Scratch
Drag and drop coding |
| 56 Create a Snake clone with Scratch
Make your first game |
| 60 Check your mail
Let Raspberry Pi act as a mail checker |
| 64 Boost your Pi's performance
Improve performance |
| 68 Monitor your network
Analyse your local network |
| 70 Tether to Android
Access the Internet anywhere using a hotspot |





Raspberry Pi 4 Model B

A super-charged Raspberry Pi that finally does everything you'd want it to, for the exact same price as the previous models

While the Raspberry Pi has enjoyed years of success, there's always been a couple of things a lot of users wanted. A slightly more powerful CPU that could handle day-to-day computing, plenty of USB ports, and maybe wireless to make connecting to the network easier.

The Raspberry Pi 3 solved these problems, but the Pi 4 Model B has improved the overall performance and added a couple of new features to make it the most impressive model yet. It uses a similar board design as the previous Pi models, but comes with two micro-HDMI ports, as opposed to the one standard HDMI that was on previous models. More importantly, these micro-HDMI ports are capable of outputting 4K video. The Pi 4 Model B also has added built-in wireless capabilities for connecting to Wi-Fi and Bluetooth.

The new BCM2711 chip is the heart of the Raspberry Pi 4 Model B. The quad-core, 1.5 GHz

processor helps to make the Pi 4 a much more functional board. Whereas before you might have had problems surfing the internet or writing a document, now the Pi 4 Model B breezes through these tasks with ease and plenty of processor power to spare.

At heart though, the Raspberry Pi 4 Model B is still the same board as the Raspberry Pi B+. As well as two USB 3.0 and two USB 2.0 ports, there's the Ethernet port for wired internet, a good-quality 3.5mm headphone jack for sound, a USB-C power input and a 40-pin GPIO port. This expanded GPIO port is fantastic for making your physical projects even more involved and complicated, letting you do far cooler things.

For those worried about compatibility, all your old files and projects and such work just fine on the Raspberry Pi 4 Model B, and all you need to do is transfer them over like any normal files.

Raspberry Pi 4 Model B

Getting started

GPIO port

The 40 pins in the GPIO port give you a range of power and function slots to control a project or read more data from your surroundings. This makes the Raspberry Pi 4 the perfect core for an Internet of Things or Maker project

Ethernet port

The Pi 4 retains the improved Gigabit Ethernet port for wired network and internet connection that was on the Pi 3 Model B+. It can run up to three times faster than older Pi models

USB ports

The four USB ports give you much more flexibility with the Raspberry Pi 4, allowing you to easily add a keyboard, mouse, wireless dongle and external storage without needing to constantly switch out or get a powered hub

Integrated wireless

The Raspberry Pi 4 Model B has built-in 802.11n wireless LAN and Bluetooth 5.0. Connecting to the internet and other devices has never been easier

MicroSD

Underneath the board is where the boot medium lives – the microSD card. Much smaller than the SD card of the original, it still holds the full operating system and allows the Pi 4 to be much smaller

Micro-HDMI ports

New to the Raspberry Pi 4 Model B is its special ability to decode 4K video on the fly with very little problem. Even better, the device has two of these ports which allows for dual-display support

Headphone jack

Need to listen to your Raspberry Pi privately? Do you want to connect it to a pair of portable speakers? Well if you do, the 3.5mm jack is still on the Pi 4



“The Pi 400 is a Raspberry Pi 4 computer integrated into a keyboard. It has the same specs and outputs as the Pi 4, but with some added convenience. It retails for £67/\$70”

Raspberry Pi 3 Model A+

The A+ took what worked with the Pi 3 and condensed it

Despite the Model A+ skipping Raspberry Pi 2 series of devices, Raspberry Pi's engineers took a look at everything that worked with the Pi 3, listened to feedback from the community, then tweaked the specs in a few places and put everything on the model A+ board.

While it may look pretty much identical to the previous A+, it is faster in terms of connectivity

both wired and wirelessly. The Raspberry Pi 3 Model A+ also kept a lot of the features introduced in the model B+, such as the 64-bit quad-core processor.

If you want a slightly more expensive alternative there is the Model B+, which is similar to the A+ but has more connectivity options and memory built onto a larger board.

MicroSD Storage

No more worries about the SD card being snapped off or lost. The Raspberry Pi 3 Model A+ features a push-push slot for a microSD card

USB input

There is a single USB input on the Pi 3 Model A+. If you require more inputs then you will want to purchase the model B+ instead

CSI camera port

This input is compatible with the Raspberry Pi Camera Module; the accessory can be purchased for around £24 and is compatible with a range of Raspberry Pi devices

Its size

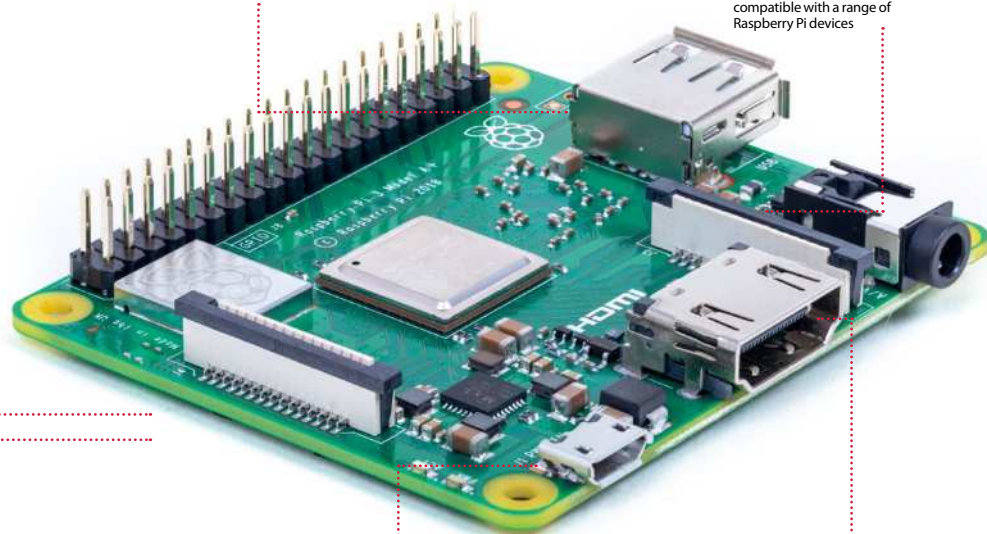
The Pi 3 model A+ has kept the same dimensions as the original Raspberry model A+ so any accessories you have for that should be compatible

Power socket

With the micro USB power socket positioned beside the HDMI socket, it's easier to arrange and manage cables

HDMI output

The Raspberry Pi 3 Model A+ has one full-size HDMI input that supports video up to full 1080p



Raspberry Pi Zero W

The tiny £4 computer has taken the world by storm, but what's changed?

While identical to the original Raspberry Pi Zero in shape and form, the Pi Zero W was the first of its kind to offer full wireless connectivity. Thanks to an on-board 802.11 b/g/n wireless chip, the Pi Zero W can act as a central hub for any wireless projects you may own.

Aside from its wireless integration, it's also one of the few Pi variants to offer built-in Bluetooth,

and includes further connectivity options through mini-HDMI and USB OTG ports. Plus, thanks to the HAT-compatible 40-pin header, the Zero W is a great way for connecting other single-boards to this mini-marvel. It's low price makes it the perfect starting point for children to get into coding, and schools are starting to introduce them into the classroom, too.

Smaller than a credit card

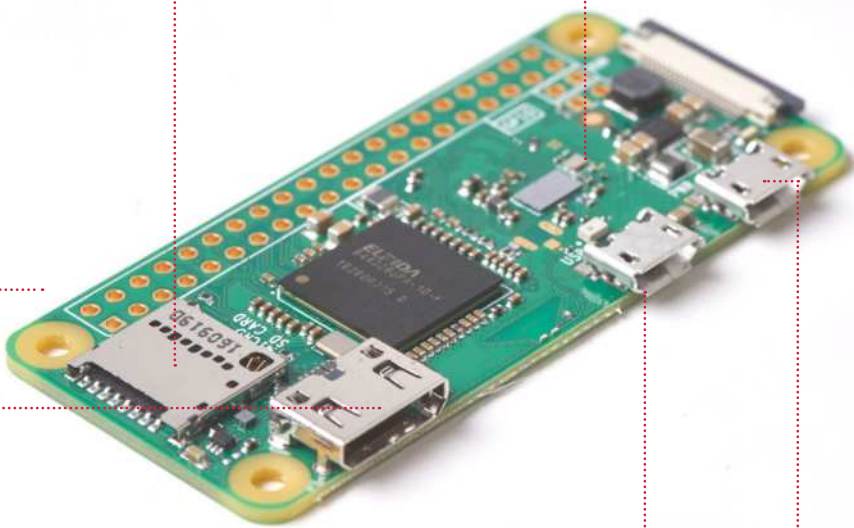
Measuring in at 65mm x 30mm x 5mm the Zero comes in at a tenth the model B's overall footprint

MicroSD card

This is the same as the newer B models, meaning you can swap in an existing card to be up and running in seconds

Unpopulated GPIO header

With an identical pinout to Model A+/B+/2B/3, headers can optionally be soldered on yourself, making the Zero HAT compatible



New mini HDMI port

You will need an adapter to make use of this port. Or better still, go headless and install yourself a lite OS image

Minimal connectors

No Ethernet or audio port and no camera or display connector. However unpopulated composite video and reset headers are still available

Micro USB port

The shortage of ports means a USB hub and an adapter will be more important than ever to connect things such as a keyboard or Wi-Fi adapter

The starter kit

There's more to your Pi than first meets the eye. Here are some vital peripherals to get you started

In order to get the very best experience from your Raspberry Pi, you're going to have to get hold of a few extras on top of the actual Raspberry Pi board itself. For example, you're going to need a keyboard and mouse with which to enter commands and navigate. While it's possible to do projects without a keyboard and mouse attached, you'll need them for the initial setup. An SD card is also an important purchase – it's where the operating system lives.

Perhaps you'll need a Wi-Fi adapter, or maybe just a length of network cable. Then there's the basic electronics side of the Raspberry Pi, what would you need to start some of the beginner electronics and control experiments? Clearly, there's more to the Raspberry Pi than some might think.

By 'peripherals', we mean other hardware that can be attached and utilised by the Raspberry Pi. They could be something as simple as a decent HDMI or they could be the latest, greatest bespoke gadgets that enhance your project capabilities.

There is an entire world of possibilities available for the Raspberry Pi; from robot arms to remote-controlled helicopters... The only limits are the hardware available and your imagination.

Did you know...

Most online retailers sell packages complete with all the accessories you might need – even pre-installed SD cards.



Keyboard and mouse

Let's start with the most basic of components, the keyboard and mouse. Generally speaking, virtually any USB keyboard and three-button scroll mouse will work with the Raspberry Pi, and although for some projects you won't even need a keyboard and mouse, you'll need them for initial setup.



SD card

Early Raspberry Pi computers required an SD card, whereas later models such as the Raspberry Pi 2 and 3 use microSD cards for storage. This is where your chosen operating system is installed, and these can be bought in various sizes, pre-installed with the OS, or blank.



Power cable

The Raspberry Pi uses a standard micro USB connector for its power input, running at 5V. In most cases a micro USB to USB cable will suffice, of which one end can be plugged into your desktop computer's USB port. An Android phone charger should also work perfectly (5.25V 1500mA).



Video output

There are two video output ports, a HDMI port and an RCA Socket. HDMI is the primary video-output connector for most users, but the RCA video-out port can also be used to connect TVs, monitors or to a SCART cable. Remote access to your Raspberry Pi is also possible via SSH or VNC.



Raspberry Pi High Quality Camera

This is a custom designed add-on board that attaches to one of the Raspberry Pi's on-board sockets via a flexible cable. It has a remarkably powerful Sony 12.3 megapixel sensor. It supports C and CS-mount lenses and it has a quarter-inch mount so you can attach it to a tripod.



Case

Securing your Raspberry Pi in a case will protect it and prevent the delicate GPIO pins from accidental damage. A case can also make your Raspberry Pi a more attractive or striking unit, perhaps as a media centre. Ensure you choose the right case for your Raspberry Pi model.



Powered USB hub

Extra USB ports are worth considering as an early purchase with your Pi. Once you've connected a keyboard and mouse you'll realise why! Using a powered USB hub is important, to stop any power being drained from the Pi, and allow you to attach the likes of an external hard drive, for example.



USB Wi-Fi adaptor

Using a USB Wi-Fi adaptor will bring flexibility to where you position your Raspberry Pi. Without a restrictive Ethernet cable, it could be used for more advanced projects where running a wired internet connection isn't a valid option. Just make sure you buy a Raspberry Pi-friendly Wi-Fi adaptor.

Set up your Raspberry Pi

Learn what goes where in your brand new Raspberry Pi with our easy-to-follow guide

While it looks daunting, setting up the Raspberry Pi for day-to-day use is actually very simple. Like a TV or a normal computer, only certain cables will fit into the specific slots, and the main job really is making sure you've got plugged in what you need at any one time. The Raspberry Pi itself doesn't label much of the board. However, most good cases will do that for you anyway – if you decide to invest in one.

Power adapter

The Raspberry Pi is powered using a micro USB cable, much like a lot of modern Android phones. It can be powered off a laptop or computer. But to make the most out of it, a proper mains adapter – like this one – is ideal

Monitor

The Raspberry Pi is capable of displaying a 1920 x 1080 output – otherwise known as 1080p. Some modern monitors allow you to plug HDMI straight into them, just like TVs do. However you may need an adapter in some cases

USB hub

There are only a limited number of USB ports on a Raspberry Pi (just one, if you have Model A). To get around this you will need a USB hub. It's important to get a powered one, as the Pi cannot supply enough juice on its own

Case and accessories

A case is not necessary to use the Pi correctly, but a decent one can keep it well protected from dust, and make it easier to move while in operation. You will need an SD card, however, of at least 4GB

Keyboard and mouse

Like any computer, you'll need a keyboard and mouse for any standard PC-style operations you do with the Raspberry Pi. The more basic the keyboard, the better; same with the mouse, as some special ones need additional software



Set up your Raspberry Pi

Getting started

Analogue output

For setups that don't use HDMI, the yellow video out port is available. To use this with sound, you'll need to use the small black port next to it, with headphones, or an auxiliary cable to pipe out the audio

USB

All the peripherals you want to connect via USB – USB hubs, keyboard, mouse, USB storage etc – is plugged in here. Ensure you have external power to the USB Hub if you have to use one though

SD card

The SD card goes in underneath the Raspberry Pi board. This will hold your operating system that runs the Raspberry Pi. The Pi OS needs to be set up from another computer before using it though

Digital output

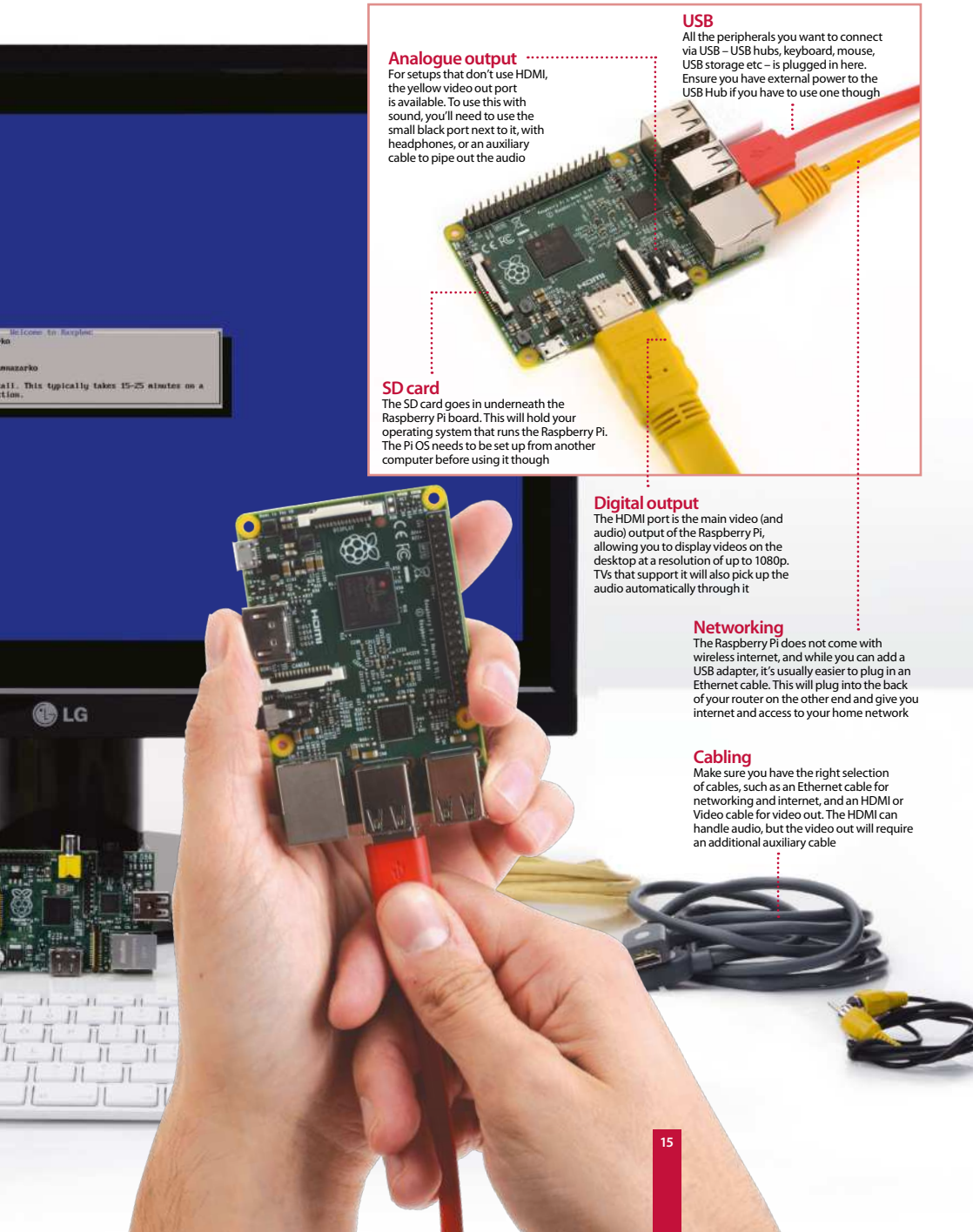
The HDMI port is the main video (and audio) output of the Raspberry Pi, allowing you to display videos on the desktop at a resolution of up to 1080p. TVs that support it will also pick up the audio automatically through it

Networking

The Raspberry Pi does not come with wireless internet, and while you can add a USB adapter, it's usually easier to plug in an Ethernet cable. This will plug into the back of your router on the other end and give you internet and access to your home network

Cabling

Make sure you have the right selection of cables, such as an Ethernet cable for networking and internet, and an HDMI or Video cable for video out. The HDMI can handle audio, but the video out will require an additional auxiliary cable



What you'll need...

Raspberry Pi Zero

Micro USB power supply

Soldering iron and solder

Pi Zero adaptor bundle

Monitor, mouse and keyboard
Optional

USB Wi-Fi or USB
Ethernet adaptor
Optional

USB hub
Optional

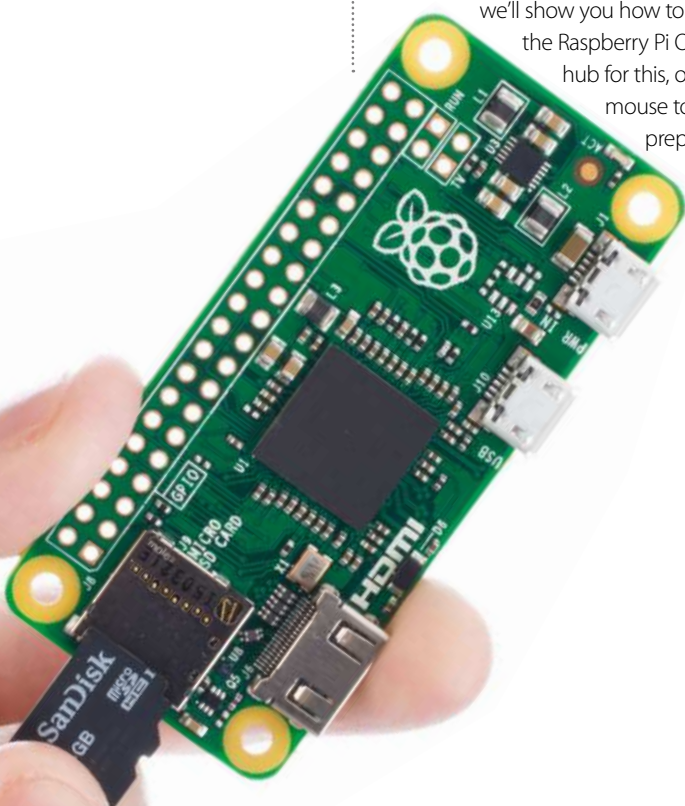
Set up your Pi Zero

Get to grips with your Raspberry Pi Zero, either as a headless device or for use together with a screen and keyboard

So you've picked up one of the tiny yet powerful Zeros, but before the coding fun can begin you need to get more familiar with it. Don't worry; we'll walk you through the Raspberry Pi Zero, the required cables, how to prepare a NOOBS SD card, and how to solder the GPIO header onto the Pi. Once the Pi is working and booted we'll show you how to get it working on Wi-Fi through the Raspberry Pi OS user interface. You'll need a USB hub for this, or even just to use a keyboard and mouse together. We'll also show you how to prepare a Raspberry Pi OS SD card for headless use (either VNC or SSH) with only a Wi-Fi adapter or USB-to-Ethernet adaptor.

Raspberry Pi Zero Cable Overview

01 The Raspberry Pi Zero is very small, and as such cannot fit normal-sized USB and HDMI connectors on. To use it, you therefore need adaptors that break out micro USB into full-size USB and mini HDMI to full-size HDMI. You also need to be very careful when connecting the micro USB cables as the micro USB power cable will fit into the connector meant for USB data. It's easy to tell them apart though, as they're clearly labelled, and the USB data connector can be found between the HDMI and power connectors.



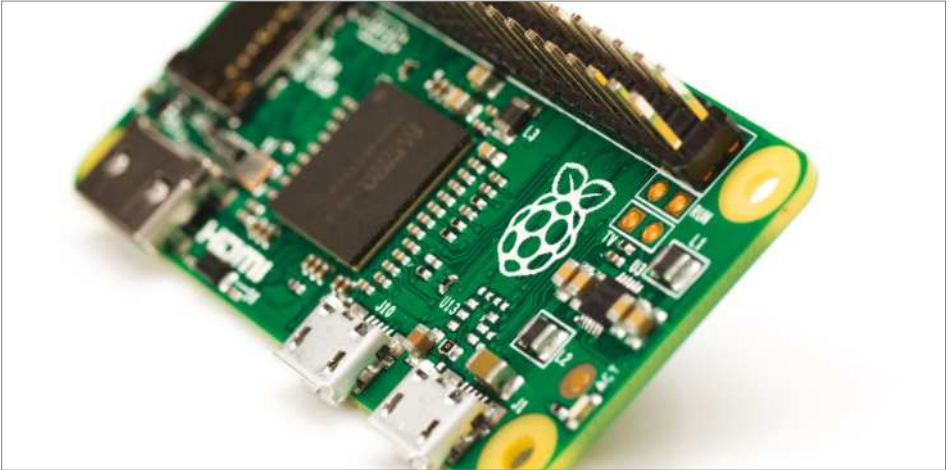


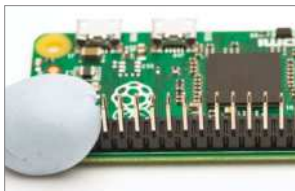
Fig 1: Once you've soldered the header into place, your Pi Zero should resemble any other Raspberry Pi

GPIO header

02 Soldering your brand new Raspberry Pi Zero might seem like a scary prospect at first, but it's not that difficult! What is difficult, however, is snapping off the correct number of GPIO header pins (40), as the kit supplies more than 40. It's also well worth noting at this point that it doesn't matter too much if you mess up and end up missing a couple of the bottom pins.

Soldering kits

03 Soldering irons are very cheap these days. If you are going to be doing a lot of soldering then it's probably worth getting a temperature-controlled one where you can change the tip. However,



the kit we used with a soldering iron, stand, solder sucker and some lead-free solder was £8 on Amazon. We managed to solder the GPIO pins using this kit no problem.

Holding the GPIO headers in place

04 Before you can solder the GPIO headers, you need to be able to hold them in place. We recommend putting some blu-tack on either side of the pins for this. This also has the advantage that you can flip the Pi over and then use the blu-tack to keep it in place on a table while you are soldering. The blu-tack should just easily peel off once you are done.

Solder the GPIO headers

05 Here comes the bit you might have been dreading, but don't worry! Make sure you have wet the sponge in the soldering iron holder, as you will need to wipe the iron on the sponge to keep the tip clean. If this is the first time your iron has been used, the heating element will probably give off a lot of smoke for

the first few minutes, so don't worry if that happens. Still, be mindful of your safety and make sure that you are soldering in a well-ventilated area – try not to breathe in any fumes. Once the iron is hot, apply some solder to the tip and wipe any excess solder on the sponge. Then start to solder the pins. For each pin, touch the tip of the iron on the bottom of the GPIO header and the metal contact on the Pi, then apply a very small amount of solder. Once the solder has flowed onto the pin and the metal contact, then you can remove the iron. If there is too much solder then you can reheat the solder and use the solder sucker to remove it. Take breaks when soldering the GPIO headers for a couple of reasons: 1) you don't want to overheat any components on the Pi, and 2) you can melt the plastic of the GPIO headers and that will allow the pin to fall through. Keep wiping the tip of the iron on the sponge to keep it clean throughout the soldering process. Make sure you unplug the iron and put it somewhere safe to cool down when you are finished.

Getting started

Set up your Pi Zero

GPIO

Once you've soldered on a 2x20 male header, your GPIOs will work as usual. To the right, you can see the four unpopulated pins for video output and a reset switch

Video

You'll need a mini-HDMI-to-HDMI adaptor to use this audio/video port, although you can also use the RCA composite video output via the unpopulated pin

Data

The power port, on the right, is micro USB as usual. The data port beside it is now micro USB as well, however, so you will likely need a micro USB-to-USB adaptor

Prepare NOOBS SD Card

06 See www.raspberrypi.org/help/noobs-setup for more details. NOOBS requires an SD card formatted as FAT32. You then need to download the latest NOOBS image from https://downloads.raspberrypi.org/NOOBS_latest and then unzip it to the SD card. On Linux, the steps are as follows:

```
sudo parted /dev/mmcblk0
(parted) mktable msdos
(parted) mkpart primary
fat32 0% 100%
(parted) quit
sudo mkfs.vfat /dev/
mmcblk0p1
cd /mnt
sudo mkdir pi
sudo mount /dev/mmcblk0p1 pi
cd pi
sudo unzip ~/Downloads/
NOOBS_v1_5_0.zip
sync
cd ..
sudo umount pi
```

Boot NOOBS and install Raspberry Pi OS

07 Connect your Pi Zero up as shown in the first step. The minimum you need connected for a NOOBS install is a monitor and a keyboard. However, a mouse and either an Ethernet adaptor or Wi-Fi adaptor are also very useful. Press Enter to select Raspberry Pi OS and then press I to install. Then press Enter to agree. Once it is finished it will say 'OS installed successfully'. Press OK and your Pi will reboot. Alternatively, if you don't want to use NOOBS, you can flash Raspberry Pi OS to an SD card in the usual manner. It'll boot into a desktop environment by default.

Configure Wi-Fi

08 If you are using a USB-to-Ethernet adaptor then the Pi should already be connected to the internet. If you are using a Wi-Fi adaptor then you will need to configure it to connect to your wireless network. We are using an Edimax EW-7811UN, which works



perfectly with the Pi out of the box. Once at the Raspberry Pi OS desktop, you can click on the network icon in order to see the available wireless networks. Once you click on one it will ask you for the password. After that it should be associated; you can hover your mouse over the icon and see the networks that you are connected to.

Configure Wi-Fi from another machine

09 If you want to use the Pi Zero as a headless device with Wi-Fi then you can prepare an SD card using another Linux machine that will already be configured to connect to the correct Wi-Fi network. You have to mount the SD card and edit /etc/wpa_supplicant/wpa_supplicant.conf, which is the same file that is configured by the

Raspberry Pi OS user interface from the previous step. Insert the SD card into your Linux machine and work out what the device is called.

```
dmesg | tail -n 3
```

[320516.612984] mmc0: new high speed SDHC card at address 0001

```
[320516.613437] mmcblk0: mmc0:0001 SD8GB 7.35 GiB
```

So the device is /dev/mmcblk0 – now we need to work out which partition number the root partition is (this will be different on a Raspberry Pi OS image; we are using a NOOBS image here).

```
sudo parted /dev/mmcblk0 print
```

This will give you a list of the partitions. The largest partition will be the root partition. In this case it's partition 7, so the root filesystem is at /dev/mmcblk0p7. To mount the SD card and edit the wpa_supplicant.conf file do the following

```
cd /mnt
sudo mkdir pi
sudo mount /dev/mmcblk0p7 pi/
cd pi/
sudo nano etc/wpa_supplicant/wpa_supplicant.conf
```

Then fill in your Wi-Fi details:

```
network={
    ssid="your_wifi_network"
    psk="your_wifi_password"
    key_mgmt=WPA-PSK
}
```

Then finally:

```
cd ..
sudo umount pi/
```

Remotely access your Pi

10 You can use nmap to scan the local network to find a Raspberry Pi. You need to know the address range of your local network (common networks are 192.168.1.0/24, and 192.168.2.0/24). You can find it

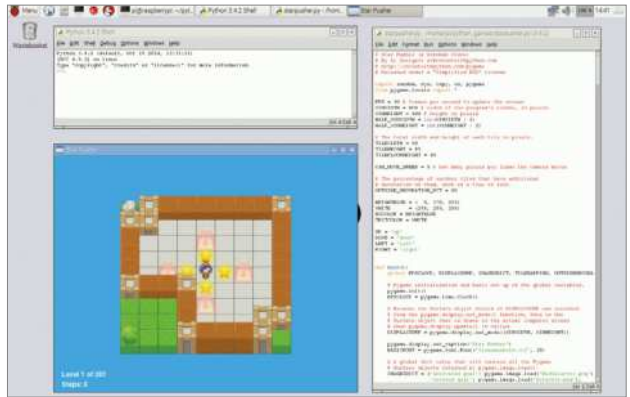


Fig 2: The Zero may be tiny but it is just as good for programming

with the ip addr command. nmap -p22 -sV 192.168.157.0/24 will scan for a list of devices with SSH open. Example output:

```
Nmap scan report for 192.168.157.29
Host is up (0.070s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh
(protocol 2.0)
```

Then you can SSH in with:

```
ssh pi@192.168.157.29
```

The password is 'raspberrypi'. If you are using the Pi headless, you'll want to disable the user interface that is started on boot by default:

```
sudo systemctl set-default multi-user.target
```

Setup a VNC server

11 VNC stands for Virtual Network Computing. Using VNC you can access the Raspberry Pi OS desktop over the network (meaning you only need power and Ethernet/Wi-Fi

connected). There is no audio support, but for any other tasks (including the use of pygame) VNC should provide an acceptable level of performance. You can install a VNC server with the following commands

```
sudo apt-get update
sudo apt-get install tightvncserver
```

There are several free VNC clients available so a search engine will help you find a suitable one. To start a VNC session on your Pi, log in over SSH and then run tightvncserver. You will be prompted to enter a password the first time you run it. You can specify a screen resolution with the -geometry option; for example, -geometry 1024x768. You can kill an existing vnc session with tightvncserver -kill :1, where 1 is the session number.

To connect to that session on a Linux machine, you could use the command: vncviewer 192.168.157.29:1, substituting for the IP address of your Raspberry Pi.

"The minimum that you need connected for a NOOBS install is a monitor and a keyboard"

What you'll need...

Raspberry Pi downloads
www.raspberrypi.org/downloads

Did you know...

The official distro for the Pi is called Raspberry Pi OS. That's what we recommend, but there are other options.

Install a distro

We take a look at some of the key aspects involved in installing a pre-built OS

With its small size and cheap price, many people might be fooled into thinking that the Raspberry Pi is only usable for basic tasks, and learning to program on. While one of the primary goals of the Pi was to increase computer literacy at a lower level rather than just learning how to create Excel spreadsheets, the Pi has many other great uses. As the Raspberry Pi is essentially a mini PC, with an HDMI and analog TV output rather than a traditional monitor connection, it can perform many common tasks that a laptop or desktop is often used for. While it doesn't really have the processing power or RAM to run the latest version of Windows, there are other options.

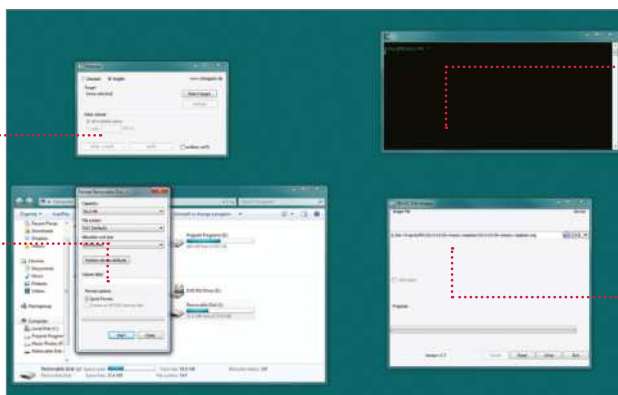
There are a wealth of fully fledged operating systems, many forked from their desktop big brothers that have been optimised specifically for the Pi. One of the most popular of these is Raspberry Pi OS, which is a port of Debian. Debian is a key part of the Linux ecosystem, and many other popular open source distributions are forked from the Debian source code. The original Debian was released in 1993, and it's come a long way since. Raspberry Pi OS needed work to get performance levels up to standard, as the Pi uses the older ARMv6 architecture. It's now a great everyday desktop.

Card speed

It's a good idea to get a reasonably fast SD card to keep your system running smoothly. Class 4 or above is best

Card format

Before you copy your OS image, you'll need to make sure the SD card is formatted into the FAT32 file system

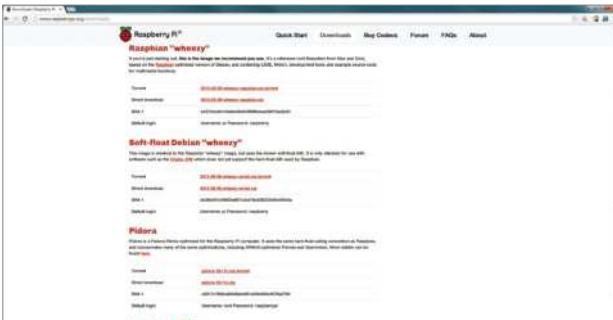


Command line

If you are using OS X or Linux, then it's likely you will use the command line to install your prebuilt operating systems

Automated tools

There are a couple of graphical tools available which make installing an image onto an SD card easy

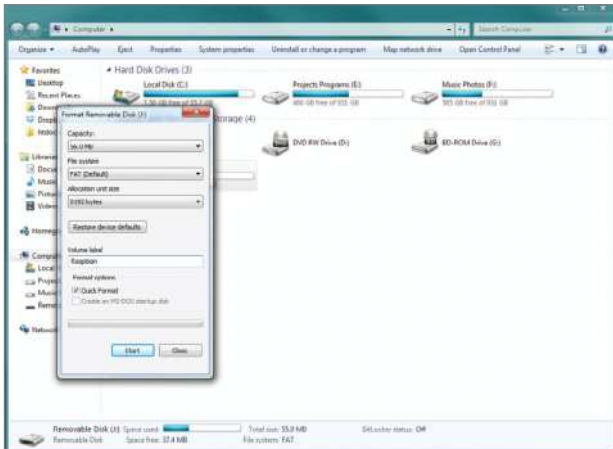


Obtaining OSs

01 One of your first questions may be "where can I find some operating systems to download?". Most of the common images can be found on the main Raspberry Pi site: www.raspberrypi.org/downloads. These are stable and well tested systems worth investigating.

Unzipping

02 When you've downloaded your image, the first thing you'll most likely need to do is unzip it. This can be done in Windows by right clicking and choosing 'extract'. In OS X, just double click to extract the files.



OS Format

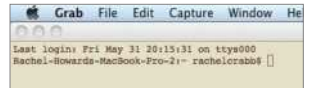
03 Within the zip you'll find a file with a .img or .iso extension. These are the equivalent of a 'snapshot' of an installation CD or DVD. Simply copying the file to the SD card won't do anything; you'll need to use a program to extract it.

SD card format

04 The SD card that you'll boot from needs to be blank, so make sure there is nothing important on it first. You'll also need to format it to use the FAT32 file system. This is a common system, used by most USB sticks and cameras.

Formatting the card

05 In Windows, to format the card simply insert and wait for it to mount. Then click on 'My Computer' and then right click on the cards icon. After that choose format and then 'FAT32' from the drop-down menu.



Using the terminal

06 If you are using OS X or Linux, then you'll have to use the terminal to copy the image. In OS X, the Terminal app comes installed by default, and most Linux versions come with one in some form or other. It may be referred to as the 'console' or 'command line'.

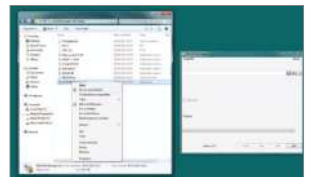
DD command

07 The command you need to use is called 'dd'. This is entered in the format of 'sudo dd bs=1m if=[img] of=/dev/[sdcard]'. Eg:

```
sudo dd bs=32m if=/Users/rachelcrabb/Desktop/ArchLinux/archlinux-hf-2013-02-11.img of=/dev/disk1
```

Win 32 Disk Imager

08 Windows users can use Win32 Disk Imager. Once you've downloaded the tool, simply right click on the .exe, and choose 'run as administrator' and follow the prompts. When the installation is complete you can put the SD card in your Pi. Easy!



What you'll need...

Raspberry Pi OS
www.raspberrypi.org

Did you know...

The command line remembers your last commands. Simply use the Up and Down arrows to use them.

Command line basics

Command line basics

Learn an alternative way to control your Raspberry Pi by using the command line and your keyboard

We've probably all been there with the Raspberry Pi. You've installed Raspberry Pi OS to your SD card and you've rushed through the setup script or not quite done your research. You start the operating system and... you end up at a command line. The first step here is to not panic: this is perfectly normal. It may just be a bit of a foreign concept to you, only seen in films with streetwise hackers who want to bring down 'the system'.

The second step, at least in this case, is simply to type:

```
$ start x
```

That's it. Raspberry Pi OS will load up the desktop and you can start using the mouse again. Quick and painless in this case, and in that of many other operating systems as well. What you've done is use a command, specifically in this case to start the X server. The X server handles the graphical interface and can be turned off by default on some Pi systems.



Fig 1: The terminal emulator allows you to access the command line while still being in the desktop environment

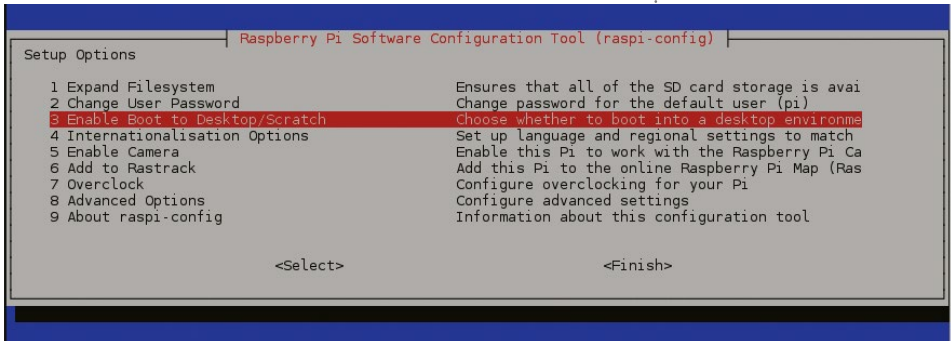


Fig 2: Access raspi-config to change settings such as boot to desktop or adding a camera module

A new world

Getting your Raspberry Pi into the desktop isn't the only thing you can do on the command line, though. There's a whole world of functionality built into the command line; in fact, most of the graphical programs you're using are just executing these commands in such a way. You don't have to leave the comfort of the desktop environment to perform these commands either, as all Raspberry Pi operating systems will come with an application known as a terminal emulator.

This creates a window where a command can be written in the same way that we launched the desktop, and use the exact same commands (Fig 1). On Raspberry Pi OS, look for the app LX Terminal in the Accessories section of the menu and click on it. If you've had to use **start x** to get into the desktop, then we can now fix that before continuing. In the terminal, enter:

```
$ raspi-config
```

Here's the initial setup screen (Fig 2). From here you can enable the desktop on boot (Fig 3), and even update the firmware and add support for the official Raspberry Pi camera module. This allows you to modify Raspberry Pi OS without having to reinstall again.

You won't be using those two commands very often, though, so is there practical use for delving into the command line? Very much so. For starters, Raspberry Pi OS doesn't have an official package manager. This is a program that allows you to browse the available software for the operating system, similar to the Pi Store. However, there's other software available to Raspberry Pi OS that you can't get through the store. You also can't specifically update the Pi software either, and all of this can be fixed using the command line.

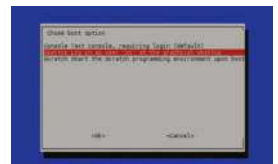


Fig 3: Change the default selection to desktop for the next time you use the Raspberry Pi

```

pi@raspberrypi: ~
File Edit Tabs Help
Do you want to continue [Y/n]? y
Get:1 http://mirrordirector.raspbian.org/raspbian/ wheezy/main dpkg armhf 1.16.12+rp11 [2,583 kB]
Get:2 http://mirrordirector.raspbian.org/raspbian/ wheezy/main curl armhf 7.26.0-1+wheezy8 [267 kB]
Get:3 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libcurl3 armhf 7.26.0-1+wheezy8 [315 kB]
Get:4 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libcurl3-gnutls armhf 7.26.0-1+wheezy8 [306 kB]
Get:5 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libyaml-0-2 armhf 0.1.4-2+deb7u2 [49.3 kB]
Get:6 http://mirrordirector.raspbian.org/raspbian/ wheezy/main dpkg-dev all 1.16.12+rp11 [1,349 kB]
Get:7 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libdpkg-perl all 1.16.12+rp11 [953 kB]
Get:8 http://mirrordirector.raspbian.org/raspbian/ wheezy/main libxfont1 armhf 1:1.4.5-3 [145 kB]
Fetched 5,968 kB in 7s (787 kB/s)
(Reading database ... 68759 files and directories currently installed.)
Preparing to replace dpkg 1.16.12 (using ../dpkg_1.16.12+rp11_armhf.deb) ...
Unpacking replacement dpkg ...
Processing triggers for man-db ...
Setting up dpkg (1.16.12+rp11) ...
(Reading database ... 68759 files and directories currently installed.)
Preparing to replace curl 7.26.0-1+wheezy7 (using ../curl_7.26.0-1+wheezy8_armhf.deb) ...
Unpacking replacement curl ...
Preparing to replace libcurl3:armhf 7.26.0-1+wheezy7 (using ../libcurl3_7.26.0-1+wheezy8_armhf.deb) ...
Unpacking replacement libcurl3:armhf ...
Preparing to replace libcurl3-gnutls:armhf 7.26.0-1+wheezy7 (using ../libcurl3-gnutls_7.26.0-1+wheezy8_armhf.deb) ...
Unpacking replacement libcurl3-gnutls:armhf ...
Preparing to replace libyaml-0-2:armhf 0.1.4-2 (using ../libyaml-0-2_0.1.4-2+deb7u2_armhf.deb) ...
Unpacking replacement libyaml-0-2:armhf ...
Preparing to replace dpkg-dev 1.16.12 (using ../dpkg-dev_1.16.12+rp11_all.deb) ...
Unpacking replacement dpkg-dev ...
Preparing to replace libdpkg-perl 1.16.12 (using ../libdpkg-perl_1.16.12+rp11_all.deb) ...
Unpacking replacement libdpkg-perl ...
Preparing to replace libxfont1 1:1.4.5-2 (using ../libxfont1_1:1.4.5-3_armhf.deb) ...
Unpacking replacement libxfont1 ...
Processing triggers for man-db ...

```

Fig 4: Upgrade your system and files, and have the latest updates and bug fixes in the process

Software for all

The first thing you'll want to do is let Raspberry Pi OS know exactly what's available online. It's a very simple task: all you need to do is:

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

This will run down a list of online repositories (or repos) that contain the software that Raspberry Pi OS uses. Once it's finished, the command-line prompt will pop up again waiting for your next command. As this is the first time you've done it, you'll likely need to update the current software on your Raspberry Pi. You can do that with the command:

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

It may ask you to confirm the upgrade, in which case type 'y' and then press Enter. What we're doing both times is using the command-line package manager Aptitude (apt-get) to first check the repos, and then upgrade packages according to that (Fig 4). The first command, sudo, allows it to run the apt-get task as an administrator, and is used in a lot of other command-line operations. To install software you use **install** instead of update or upgrade, followed by the name of the package. For example, with the mathematical programming language, you can install it with:

```
$ sudo apt-get install wolfram-engine
```

Did you know...

You can use the Tab to complete commands. Just start typing and hit tab. If you like what you see hit Enter to finish!

Move and create

Installing and updating are just a couple of the many things you can do in the command line. You can also browse the entire file system, move files, create folders and delete items. All of these are very simple operations.

When you first open the terminal, it will open up in your home folder. While you can't specifically tell that it is, you can display exactly what kind of files are in the directory with (Fig 5):

```
$ ls
```

The tilde sign (~) is used to denote the home folder and can be used for navigating around the file system. To navigate, we'll be using the **cd** command, followed by the location you want to move to. This can be done like so:

```
$ cd /home/pi/Downloads
```

This will move you to the Downloads directory. As we were starting off in the home folder to begin with, we actually only needed to do this:

```
$ cd Downloads
```

It's context sensitive and knows to look in the directory it's already in. There's another trick you can use so you don't have to remember the exact name of the path – the command line or terminal will try to auto-complete the phrase if you press the Tab key while something is only partially typed. Try the **cd** command again, but try pressing Tab when you've only written 'Down'.

Finally, there are some quick commands you can use to manipulate files. Individual files can be copied using the command **cp**, followed by the filename and the new location like so:

```
$ cp file.txt ~/Documents/file.txt
```

You can also use this to rename files by doing:

```
$ cp file.txt otherfile.txt
```

The original file can then be deleted by using the **rm** command:

```
$ rm file.txt
```

Want to create a new folder? Use **cd** to move to the directory you need to add a folder to, and then use **mkdir** followed by the name you want to give the folder:

```
$ mkdir NewFolder
```

There's a lot more you can do with the command line, but these are the very basics. As you use Linux more and more, you'll be confronted with tasks that need the command line, and through this process you'll learn just how much can be accomplished when you work like a street-wise movie hacker.

Did you know...

You can always return to your Home folder in the command line by typing `cd ~` and pressing Enter.

"The command line or terminal will try to auto-complete the phrase if you press the Tab key while something is partially typed"



Fig 5: There are many simple command-line tools that can help you browse and use your system

What you'll need...

Raspberry Pi:

(Ideally a Pi 2 or 3 if you want to use Chromium for streaming video)

8GB Micro SD card:

(If installing from Raspberry Pi OS from scratch)

Access to a computer:

(For connecting via SSH and/or to install Raspberry Pi OS with Pixel)

Get the new Pixel desktop

Get the Pixel desktop

Explore Pixel, the official desktop environment for the Raspberry Pi

Pixel (Pi Improved Xwindows Environment, Lightweight), is one of the latest iterations of the Raspberry Pi OS desktop. The major changes to prior editions are apparent the first time you boot up – the multitude of boot messages has been replaced with a simple splash screen with the release number.

There are now 16 stunning desktop background images to choose from thanks to Pi Foundation developer Greg Annandale. The icons on the file manager, task bar and menu now have a crisp, professional appearance. Menus are also cleaner and more readable as application icons no longer appear by default.

The rather clunky windows we formerly knew in Raspbian have now been replaced with rounded corners and a modified title bar. The infinity patchset actually also makes for much cleaner font rendering.

Beneath the hood, RealVNC Server is now bundled to allow you to easily select VNC from the interfaces menu, then connect via a viewer. There's also a provisional release of Chromium for the Pi, which in combination with the h264ify plugin makes use of the Pi's hardware to stream video.

Did you know...

Some of the newly designed icons aren't immediately visible as none of the default applications belong in the Engineering or Education categories. If you're curious about this, head over to `/usr/share/icons/PIX` to take a peek at the icons. Alternatively click on Menu>Preferences>Add/Remove software to see the categories and their respective sleek icons. It is possible to modify the icons for individual apps. Check Step 15 for more information.



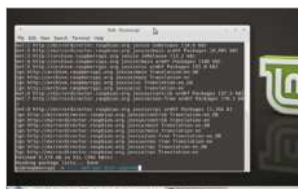
Get the Pixel desktop

Getting started



Choose your method

01 There are several ways to install Raspberry Pi OS with Pixel. If you have Raspberry Pi OS with NOOBS, you can restart your Pi by holding Shift and choose to reinstall. This will upgrade you to the latest version using Pixel but will also wipe your existing installation. Alternatively you can download the latest Raspberry Pi OS Image from <https://www.raspberrypi.org/downloads> and follow the easy installation guide at <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>. If you use either of these methods, skip ahead to Step 5.



Manual upgrade to Raspberry Pi OS with Pixel

02 If your Raspberry Pi has an existing installation of Raspberry Pi OS, you can upgrade to the latest version of Raspberry Pi OS with Pixel by opening Terminal or connecting via SSH and running the commands:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

The upgrade process will take some time. You'll see a message about the

plymouth I/O multiplexing framework; press Q to dismiss this and proceed.

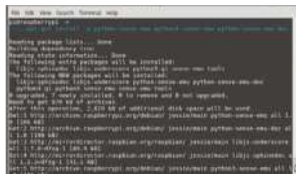
Install Chromium

03 The latest version of Raspberry Pi OS includes an initial release of Chromium. As we're performing a manual upgrade, you can install this with the command:

```
sudo apt-get install -y rpi-chromium-mods
```



Chromium comes bundled with the previously mentioned h264ify extension, which forces YouTube to use the Pi's hardware acceleration when streaming videos. The awesome adblocker uBlock Origin is also bundled to strip out resource-hungry adverts.



Install SenseHAT Emulator

04 This step is optional, but is included as the SenseHAT emulator is also included with the latest version of Raspberry Pi OS. The SenseHAT is an add-on board for the Raspberry Pi with a range of sensors from a thermometer to a gyroscope. The emulator allows developers to test code for devices without actually owning a SenseHAT itself. For more information see www.raspberrypi.org/blog/desktop-sense-hat-emulator/. Run this command to install all necessary files:

```
sudo apt-get install -y
python3-sense-emu python3-sense-
```

emu python-sense-emu-doc



Load Pixel Desktop

05 Reboot the Pi to see your new, shiny Pixel Desktop. You may see a message stating that your previous configuration files have been overwritten. Click OK to dismiss this. At this stage you might also want to change the default wallpaper. Right click anywhere on the desktop and click Desktop Preferences. Click the Wallpaper menu to choose from any of the stunning options. Our current favourite is Mountain.

Examine your interfaces

06 Disable both Bluetooth and Wi-Fi from within the desktop environment with the click of a button. Simply click on the relevant icon and turn it off. Head over to Menu>Preferences>Raspberry Pi Configuration>Interfaces and click Enabled under VNC. The VNC icon will appear at the top-right of the desktop. Click this to view your Pi's private IP address for VNC clients.



Import old bookmarks into Chromium

07 You can open Chromium by clicking the web browser icon

Getting started

at the top-left of the screen. If you want to import your bookmarks from Epiphany, open Terminal on the Pi and run:

```
epiphany-browser.
```



Click the Settings gear icon on the top-right and choose Edit Bookmarks. On the window that opens you'll have the option to export your bookmarks to HTML format. Close Epiphany and choose the blue link Import Bookmarks in Chromium to add them there..

Change the Appearance Settings

08 While the default options make for a stunning desktop, you may wish to perform some small tweaks at this stage, particularly if this is a new install of Raspberry Pi OS. In the main menu, head over to

Get the new Pixel desktop

Preferences>Appearance Settings. If you're used to a menu bar at the bottom of the screen, change the Position setting to Bottom. You can also change the size of the bar to medium or small.



launch it from the Programming menu in Applications. The Emulator comes bundled with around a dozen example scripts to get you started. Simply click the File menu at the top-left of the emulator window, then Open Example. There are beginner, intermediate and advanced projects. If your interest has been piqued, SenseHATs are currently available online from the Pi Hut for £30. See thepihut.com/products/raspberry-sense-hat-astro-pi.

Enable login screen

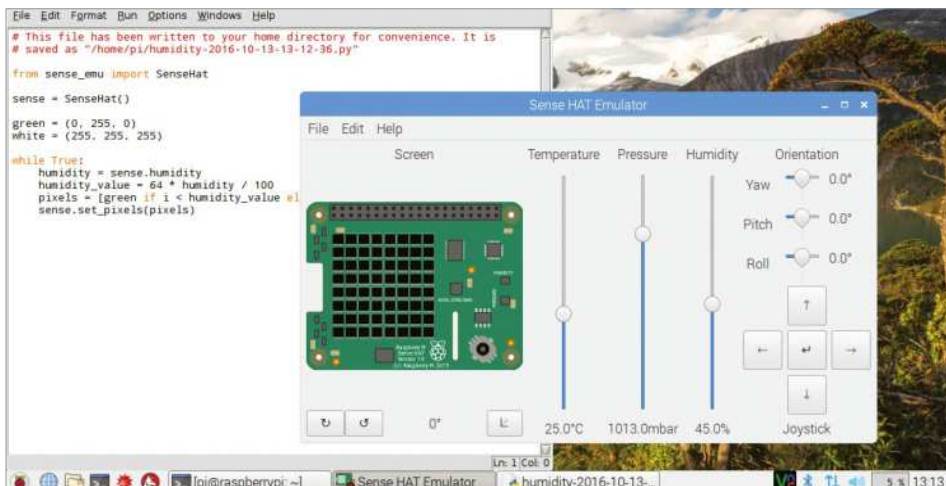
10 Now the Raspberry Pi has a rich desktop environment and modern web browser, you may wish to use it for a work or home computer. To make the Pi require a password on startup, open Terminal on or connect via SSH and run the command:

```
sudo nano /etc/lightdm/lightdm.conf
```

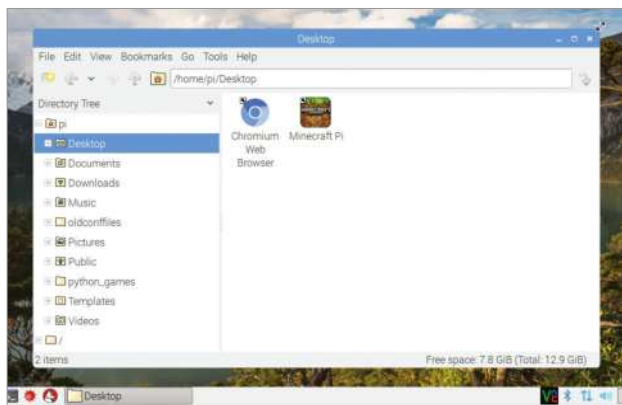
Scroll down to the line `autologin-user=pi` and put a hash (#) at the start. Press Ctrl+X, then Y, then return to save and exit. Remember the default password is 'raspberrypi'.

Test SenseHAT Emulator

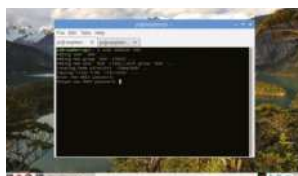
09 If you installed the SenseHAT Emulator previously, you can



Get the Pixel desktop



Getting started



Now enter the password for the new user twice when prompted to create the account.

Practise resizing windows

12 One blessing of the old Raspbian desktop is that window frames were quite thick, whereas Pixel includes much slicker, thinner windows so it can be tricky at first to resize them. Fortunately in the latest version of Raspberry Pi OS the grab handles now extend outside the window, so even if the mouse is just outside the frame, you'll see the cursor change.

Use the right arrow to find the text 'quiet splash' and delete it. Use Ctrl+X, then Y, then return to save and exit.

Add temperature/voltage monitors

14 In previous versions of Raspbian, if the Pi was overheating or underpowered, you may have noticed crude yellow and red squares appearing in the corner of the screen. These have now been replaced with pictures of a thermometer and lightning bolt respectively. Right-click the task bar and Add Panel Items to display the system temperature and voltage if you wish.

Add new users

11 If you followed the previous step, you may want to add extra user accounts for your family or colleagues. First open Terminal on the Pi or connect via SSH and then run the following command:

■ `passwd.`

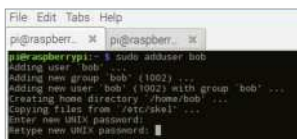
This will allow you to change the default password 'raspberrypi' to something more meaningful. Once you have done this add more users with the command:

■ `sudo adduser`
(eg `sudo adduser bob`)

Configure splash screen

13 Pixel's sedate splash screen on boot may not sit well with some people's principles; the previous text boot is also very useful for detecting errors. If you wish to go back to the old style boot screen, open Terminal or connect via SSH and run the command:

■ `sudo nano /boot/cmdline.txt`



Explore your icons

15 The icons have been painstakingly redrawn for Pixel by Sam Alder and Alex Carter, who also illustrate the graphics of the official Raspberry Pi website. You can find all the system icons in `/usr/share/icons/PIX`. The icons are very professional and easy on the eye, however if you wish to replace any of them, make sure to try and use a .PNG image and to give it the exact same name as the image you're replacing, for example `launch.png`.

What you'll need...

Raspberry Pi OS

Did you know...

You can access the Config tool anytime by typing 'sudo raspi-config' from a terminal window or the command line.

Master the Config tool

Tell your Raspberry Pi how to behave using the powerful built-in Config tool

The 'RasPi Config' tool allows configuration of your system that would otherwise be trickier in the Linux environment and it's the first thing you'll see when you install Raspberry Pi OS. Why? Tasks such as setting the date and time or regional settings for your keyboard are often done in a command-line interface with no dialogs, no additional help – for a new user, this is a nightmare.

There are some further specifics for the Pi and Raspberry Pi OS itself, such as: the ability to easily enable overscan for your TV; change the split of memory to the computer/graphics card or even overclock your system to make it a little faster; enable remote SSH access to the system; stop the system booting into the desktop environment among other things. The RasPi Config tool takes the pain out of the process and puts real power at your fingertips.

Change password

Change the password for your default 'pi' username to make it something more personal or easier to remember for you

Expand roots

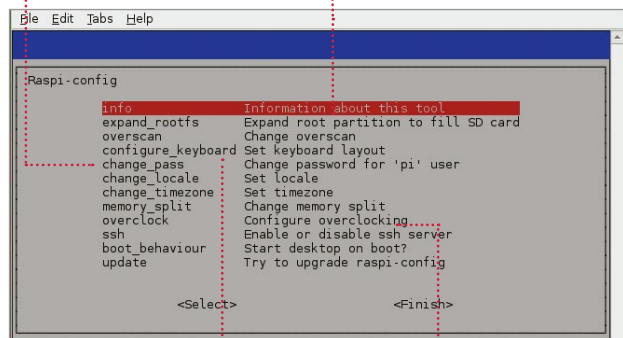
Allows you to very quickly and easily change the partition of the roots to fill the SD card completely

Open the raspi-config tool

01 Start by double-clicking the LXTerminal icon on your desktop. This will start the command prompt, where you'll be able to run the config tool. To do this you'll need to run a command.

■ `sudo raspi-config`
When asked for your password, you won't actually see it being typed.

When you've typed the password and pressed Enter to submit it, the config screen will be shown to you. There are a few settings of particular interest that we'll cover in this section, although they all have their uses in the running of your Pi. Some of the settings in this menu are important and some are irreversible, so use them with caution.



Configure your keyboard

Set the correct keyboard up – there are many different layouts. Using the wrong one can be annoying

Overclocking

Allows you to quickly and easily overclock your Pi to give you some extra speed and power with little risk

Expand the root file system

02 By default, the Raspberry Pi OS root file system will be 2GB – this is done so that the image provided for it can fit on as many different SD cards as possible.

If your card is larger the 'expand_roots' option will make the OS use the entire space. Upon using this option, the command will be executed immediately. The operating can take some time. Reboot your system to see the changes.



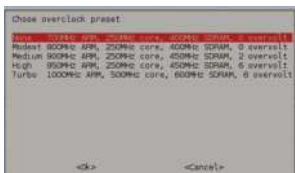
Configure your location

03 Locale is the language and regional settings that your Pi is using – while this generally has little impact on what you'll see, it is also responsible for any default currency settings, etc, so could prove to be an irritant at a later time if wrong.

Upon selecting the option, you'll be taken through a wizard. Use the arrow keys to check the built locale before building more (it takes a while). Timezone will take you to a tzdata screen where you can adjust it.

Overclock your Pi

04 You can set the clock speed and voltage of your Pi to several different presets. Setting the clock speed and voltage at higher rates than the specification may cause instability, so do so in small increments



and ensure good airflow around your Pi.

If you see any noticeable instability, run this wizard again and set the clock speed back down to something slightly lower – repeat until your system is completely stable. For the scope of this tutorial, a Modest/Medium overclock is recommended – it seems to give a little extra performance with no noticeable side effects. It is also recommended to reboot your system after making this change. Hold the Shift key to temporarily disable overclocking.

Change the memory split

05 Changing the memory split of the Pi allows you to give either the system or the graphics processor a larger amount of memory.

The value you give to it must be either 16/32/64/128/256. Here are our recommendations:

32MB GPU memory for basic distro usage where video and 3D rendering aren't required.

64MB GPU memory for desktop use that requires video playback or have 3D effects enabled.

128MB GPU memory for graphical applications and games that do extensive multimedia or play 3D rendered games.

For most people, a 64MB split for graphics will suffice.

Change boot behaviour

06 By default, the Raspberry Pi OS distro will boot into a command-line interface, whereby you have to first log in as 'pi'.

If you then want to run a window manager (in this case, it's called 'X'), you have to give the system a command to let it know that's what you want to do.

For a lot of people this isn't really ideal since command lines scare them. Because of this, there's an option to

start X automatically, on boot. Set this option to 'Yes' to enable this behaviour by default.

You can obviously revert this at any time to return to a text-based login where you have to start manually:

startx

Turn overscan on and off

07 You may have noticed one of two behaviours if you're using your Pi with a modern HDTV.

There is a black border the whole way around the image output by the Pi – it just doesn't fit correctly. This is caused by underscan.

If you can't see the edges of your screen to get to them you're suffering from overscan.

If you have the former issue, you may need to either turn on overscan, or enable a 'zoom' mode or similar on your TV.

If you have the latter issue, you need to turn overscan off so that you can see the edges.

Update raspi-config

08 The raspi-config tool receives updates from time to time. This is generally to either add more features or fix small bugs, or both!

It's not a bad idea to run the updater when you use the tool – before you start changing any system settings. While it's much more likely that it'll be updated to look better or do more things, it's not impossible there could be miscellaneous bug fixes hidden within that would otherwise cause you some grief.

Remember, though, when you're trying to update your copy of the raspi-config tool, you'll need an active internet connection, either through an LAN cable or wireless dongle.

Without them, it's never going to get any newer. Always try to make sure you're on the latest version.

What you'll need...

Any Raspberry Pi distro

www.raspberrypi.org/downloads

Wi-Fi dongle or Ethernet

Did you know...

If your Raspberry Pi is going to be placed near your Internet router, all you need to do is plug in an Ethernet cable.

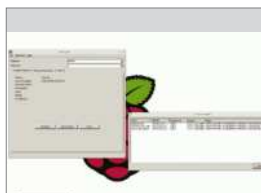


Fig 1: With a Wi-Fi dongle attached, run the utility and scan for available networks



Fig 2: Enter the pre-shared key in order to connect to your home router

Get online

Get online

To access a world of utilities, apps and resources you need to get online. This is how to do it...

The easiest way to get online is to buy a Raspberry Pi Model B+, as it comes with an Ethernet socket. The Model A not only lacks the Ethernet port, but is handicapped by only having one USB port. That means you will have to buy a power USB hub in order to get online. Back to the Model B+ though and to get online, simply plug an Ethernet cable into the socket on the Pi and connect it to a similar port on the back of your internet modem/router. Turn your Pi on and launch the desktop, then double-click on Empathy and you should see the internet appear (main image). To check that it's working, look at the lights on the Pi itself. The red power light should be on. Above this is the green light that flickers when accessing the SD card. Below the power light are the three Ethernet-related lights. Note that the Model A does not have these LEDs because it doesn't have the Ethernet socket. The middle light is green and comes on when it detects a Full Duplex LAN connection. This means it is able to send and receive data to the internet. The next light is green and flashes when actually accessing the internet by sending or receiving data. The last light is yellow and will come on and stay on when a 100Mb LAN connection is detected.

The Wi-Fi option

If you aren't close enough to the modem/router to be able to plug in the Ethernet connection, or you simply have a Model A, then a powered USB hub is required. This plugs into a USB port on the Pi. You can then plug a Wi-Fi dongle into this. Boot up the Pi and launch the desktop. Then double-click on the Wi-Fi Config icon. You should see a name for the dongle in the Adapter section. Click on Scan to look for networks and a list of those found should appear (Fig 1). Double-click on the one you want to connect to and the details for it will be listed. Almost all home networks use a network key, which is usually written on the modem itself. Click on PSK, which stands for pre-shared key, and type it in (Fig 2). Then click on Add. It will process this, then associate the connection and

then finally, a new IP address for the Wi-Fi connection will appear. If you click on the Manage Networks tab, the network will now be listed and have an Enabled radio button active. To get on the internet, simply launch Empathy and you'll be connected. The Wi-Fi utility will remain running on the bottom right of the panel. If you right-click on the Wi-Fi icon you will see options to Disconnect or Reconnect, event history and the results of the most recent scan. Click on Status to see how it's performing.

Checking the connection

To check that the Pi has a valid internet connection, double-click on LXTerminal. Enter this command:

```
ip addr
```

You should see a list of numbers, with the bottom line starting 'inet' and then the IP address of the Pi connection (Fig 3). Typically this is something like 192.168.1.11 and this shows that the connection is working because the Pi has been assigned an IP address based on the one used by your internet modem/router. If this doesn't come up then there may be a problem at the router end. The modem/router should be running a DHCP server and when the Pi connects to it, it will be given the IP address. If it isn't running then nothing else connected to it will be able to access the internet either.

Use the web interface with another device to log onto 192.168.1.0 or whatever is your modem's actual IP address in order to check that the DHCP server service is turned on. Finally, in the terminal, type:

```
ping google.com
```

Sharing a connection

If you don't have a Wi-Fi dongle, a powered USB hub or a long enough Ethernet cable, but do have another computer connected to the internet, there's another way of getting access. On a Mac, connect it to the Pi via a USB or Ethernet cable. Launch System Preferences; under Internet & Wireless, click on Sharing. Click on Internet Sharing, then select Wi-Fi (or AirPort) as the connection type to share, and select how the Pi is connected to your Mac (Fig 4).

On a Windows PC, go to Windows Explorer>Networking>Networking and Sharing Center>Change Adapter Settings.

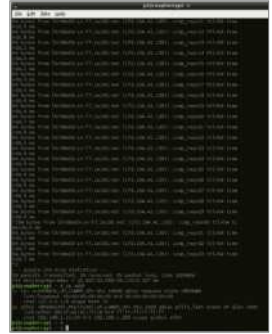
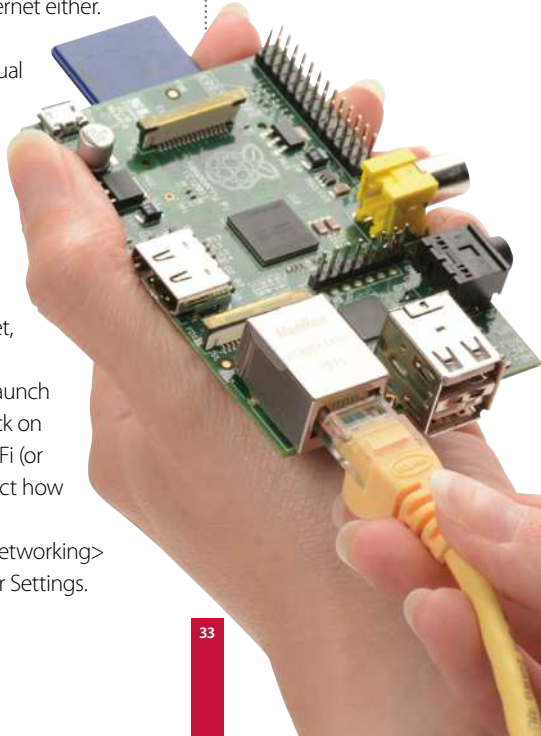


Fig 3: If it doesn't look like the connection is working, there are some easy ways of checking what's going on



Fig 4: Both Windows and Mac computers can share their internet connections with a directly-connected Pi



What you'll need...

Apt command help page:

<http://linux.die.net/man/8/apt>

Apt-get help page:

<http://manpages.ubuntu.com/manpages/lucid/man8/apt-get.8.html>

Did you know...

If you press the Tab key the command line will attempt to auto-complete your command for you – just press Enter to finish.

Install and use packages

The Raspberry Pi is great, but it's made better with the software you install onto it

On its own, the Raspberry Pi is a near-perfect mini computer. It already contains a wealth of educational software, a few games, some programming utilities and a number of system tools. But, as with most computers, this is only the tip of the proverbial iceberg. By installing more programs, you can do much more.

These programs, known as packages, are as wide and as varied as the developers who originally designed them. In Linux, if there's a need for a particular program, then someone develops one. They then put it out to the world and make the source code freely available, hence 'open source'. Once the program has been tested, it will eventually make its way onto one of the many remote servers for that particular Linux distro.

These remote servers, called repositories, or repos, contain all the elements of the package in order for it to be downloaded and installed onto your system. The process is very quick and easy once you know how it's done...

```
pi@raspberrypi / $ sudo apt-get
```

```
(__)  
(oo)  
  /-----\  
 /   |   |   \  
*  /\---/\  
   ~~~~
```

```
...."Have you mooed today?"...
```


Update and upgrade

01 Getting hold of a package on the Raspberry Pi involves dropping into the command-line terminal, via the LXTerminal icon on the desktop, and entering a few commands. But before we do that, we need to make sure the system is up to date. Enter the following into the terminal:

```

■ sudo apt-get update
■ sudo apt-get upgrade
Or...
■ sudo apt-get update && sudo apt-get upgrade

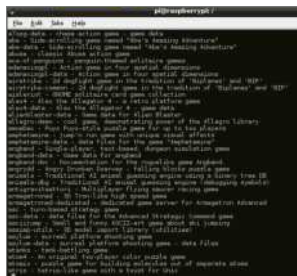
```

Search for a package

02 The apt-get command

(Advanced Package Tool) is the key to downloading and installing packages on the Raspberry Pi. In the previous instance, we updated the existing packages and system, upgraded any that needed it, and updated the current package list. Now, let's search the list of server packages for available games.

```
apt-cache search game | less
```



Apt searching

03 The current list you find yourself in is the name of all the packages labelled as 'games' from the available server. In the list, the part before the hyphen tells you the name of the package, which is what

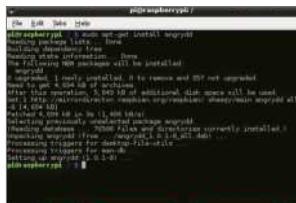
you will need to know to be able to install it. Use the arrow keys up/down to navigate; press 'Q' to exit.

Installing a package

04 Using the up and down arrow keys, navigate the list. If you find something you like the look of, say Angry Drunken Dwarves, remember the name of the package, in this case 'angrydd', and press 'Q' to exit the list.

To install the package, enter the following in the terminal:

```
sudo apt-get install angrydd
```



Executing the package

05 The result of the previous command should be the successful download and installation of the game, Angry Drunken Dwarves. To execute the newly installed package, you can either run it from the LXDE Menu under Games>Angry Drunken Dwarves, or by typing in the following into the terminal:

angrydd

Remove a package

06 This installing of packages is perfectly fine, and you can see just how powerful a command Apt really is. But, what if you want to remove a package?

Using the `Apt` command again, let's say we want to completely remove all trace of Angry Drunken Dwarves from the Raspberry Pi.

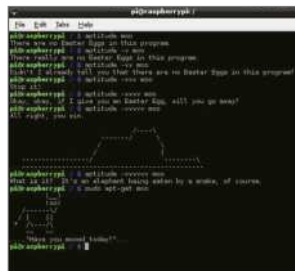
```
sudo apt-get --purge remove  
angrydd
```

Enter 'Y' to accept the removal.

Apt Easter eggs

07 The `Apt` command is a shorter, non-menu-driven variant of the `Aptitude` command. This command has a long history in Linux, and as a result has some rather special 'features', also known as Easter eggs. Purely for a little bit of fun, type in the following commands and see the results:

```
aptitude moo
aptitude -v moo
aptitude -vv moo
aptitude -vvv moo
aptitude -vvvv moo
aptitude -vvvvv moo
aptitude -vvvvvv moo
sudo apt-get moo
```



Man the Apt command

08 As you can see, there is more to the simple Apt command than what first meets the eye. There are many different sub-commands that you can run, and many different variations in which to run them.

If you want to see what else the Apt command can do, enter the following:

man apt



Getting started

What you'll need...

Synaptic:
www.nongnyu.org/synaptic/

Did you know...

Synaptic has access to the same repo as via the command line as demonstrated on the previous two pages.

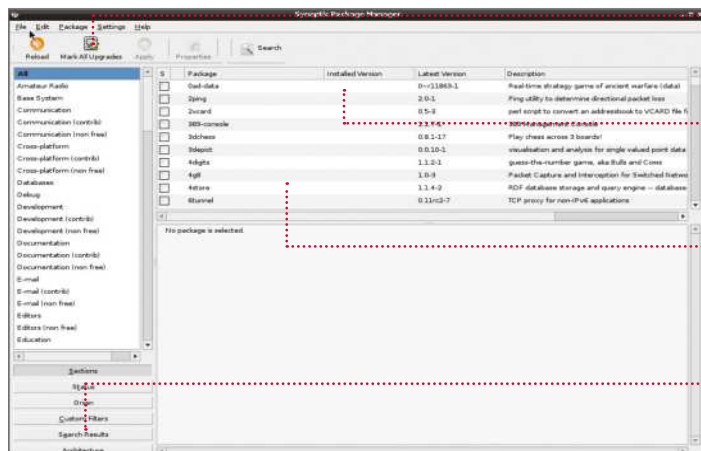
Use graphical installations

Use graphical installations

Would you prefer a graphical interface to install new programs? If so, then read on...

If you're new to Linux, you may find using its built-in Apt package management tool a bit intimidating and confusing. The apt-get command is used for installing applications through the internet, connecting to the remote servers – called repositories – which house the programs as packages. But it is used through the terminal command prompt, which can be daunting, so we need an alternative: a desktop environment interface method of getting hold of packages.

This is where Synaptic comes in. Synaptic is a friendly-looking graphical interface to the apt-get terminal command which allows you to manage your application installations, and removals, through the already familiar desktop environment. Think of it as a kind of online shop where you can pick and choose the programs you want and have them downloaded and installed onto your Raspberry Pi without you having to drop into the terminal.



Upgrade entire systems

Synaptic has the ability to update and upgrade every program or package, and it can upgrade your entire system to the latest version

Install and more

Synaptic is a very powerful tool. With it you can install, remove, upgrade and downgrade single or multiple packages and programs

[Browse all documentation](#)

From within Synaptic, you are able to browse and read all available online documentation related to a package or program

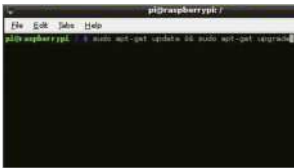
Easily find programs

Synaptic enables you to easily locate packages and programs by name, description, version and even by who developed the program

Update the system

01 Unfortunately, if you have an aversion to dropping into the command-line terminal, then you're going to be stuck at the first step. Before we install anything, we need to make sure that the Raspberry Pi is fully updated and any existing packages are upgraded. Simply enter the following into the LXTerminal:

```
■ sudo apt-get update
■ sudo apt-get upgrade
```



Installing Synaptic

02 To install Synaptic, you'll first need to enter the LXTerminal and run the command below – don't forget to type Y to any prompts asking you to accept the installation:

```
■ sudo apt-get install synaptic
```



Running Synaptic – Part 1

03 In essence, that's all you need to do. Synaptic is now installed and ready to use. However, due to its complexity, there may be some bugs that need ironing out first, so it's best to follow these steps. To test if Synaptic is working okay, first enter the following command into the terminal:

```
■ gksudo synaptic
```



Running Synaptic – Part 2

04 You should be now looking at the Synaptic program window, where you can scroll through the list of available programs and click on each to download and install. Now we need to test whether it will run from the LXDE menu. Click on the icon in the bottom left, then go to Preferences>Synaptic Package Manager.

Running Synaptic – Part 3

05 Running Synaptic from the LXDE menu results in an error. Don't panic, however: all it's doing is asking for a password. Enter the following password into the box:

```
■ raspberry
This is the default Pi password so we're assuming you haven't changed it.
```

Fixing Synaptic – Option 1

06 This will temporarily fix the issue, but to permanently resolve it, do one of the following. First, right-click the Synaptic icon in

the menu and left-click Properties. In the Command text box, change the text to add the gksudo command. So instead of 'synaptic-pkexec', it will read:

```
■ gksudo synaptic-pkexec
```

Fixing Synaptic – Option 2

07 The second, and best, option is to drop back into the terminal and alter the way in which the program is executed from the menu. All that's needed is to change one line to another, so that the gksudo command is again used instead of the plain synaptic-pkexec. From the terminal, type:

```
■ sudo nano /usr/share/applications/
synaptic.desktop
Change 'Exec=synaptic-pkexec' to...
■ Exec=gksudo synaptic-pkexec
```

Synaptic fully working

08 After you've entered those changes, exit nano via Ctrl+X, followed by Y to accept the changes, and then press Enter a couple of times to get back to the command prompt. You can now launch Synaptic from the menu, or by entering the following command when you're in the terminal:

```
■ gksudo synaptic
```

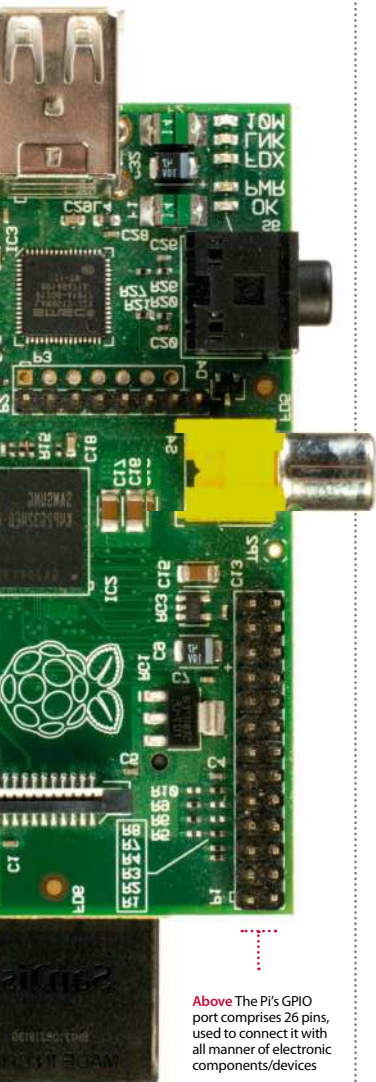


What you'll need...

Prototyping equipment

RPi.GPIO

<https://pypi.python.org/pypi/RPi.GPIO>



Above The Pi's GPIO port comprises 26 pins, used to connect it with all manner of electronic components/devices

GPIO port explained

GPIO port explained

Learn how to harness the power of the GPIO port on your Raspberry Pi. It's easier than you think...

The general-purpose input/output (GPIO) pins on your Raspberry Pi are often central to the success of the projects you'll find in this book. Without them you have no way of interfacing with the real world, be it to trigger lights, buttons or buzzers or read sensors.

GPIO pins aren't special to the Pi; they're actually a standard designed to help control input and output behaviour with all kinds of integrated circuits. Usually you'll find that any one GPIO pin has no particular use pre-defined and they tend to be turned off by default.

Raspberry Pi GPIO

The GPIO pins on the Raspberry Pi can be controlled and triggered in many ways. You can use them from the terminal directly and through Bash scripts, or you can control them using specially designed modules for popular programming languages. Since Python is the official language of the Raspberry Pi, you'll find the GPIO module for Python gives you among the best control for inputs and outputs available. The library is called RPi.GPIO and is installed by default on all Raspberry Pi, but can otherwise be installed in exactly the same way you'd install any useful Python library. The project is hosted on SourceForge and can be found at sourceforge.net/projects/raspberry-gpio-python. You'll also find useful links, information and examples of how to use and control the GPIO pins from within simple Python scripts. It helps to have a basic understanding of Python if you plan to use RPi.GPIO, so we'd recommend a basic introductory course like the one found at www.codecademy.com or by reading the official Python documentation at www.python.org/doc.

There are 26 GPIO pins on the Raspberry Pi and you can use the vast majority of them in any way you want. There are a few pins that

have special purposes, though, so we recommend you familiarise yourself with their layout. For example, the very top row of pins are designed to offer power to external devices like buttons and lights. Since an earth line (often called 'ground') is needed to safely create a circuit, you'll also find several ground pins located in the GPIO port.

How to use GPIO pins

To exploit the power of the GPIO port you'll need a few essential components, the most important of which are jumper leads. Since the pins on the port are 'male', you'll need to purchase either 'female to male' or 'female to female' cables, depending on what hardware you intend to connect to your Pi. Assuming the device you're connecting to also has male connectors, 'female to female' jumper leads will do nicely, but often you'll be using a breadboard to prototype your circuits, in which case 'female to male' connectors are preferred. Cables and breadboards can be bought very cheaply from just about any online store that sells Raspberry Pi accessories and can usually be found in the 'prototyping' section of the store.

Naming conventions

Once you're ready to connect your device, the next task is to find the right pin for the job. While it's true that all GPIO ports are multipurpose, some are more multipurpose than others! As we've already discovered, some pins are reserved for 5V, 3.3V and ground. Others also have special capabilities, but what's worse is that they can also be called different things. For example, GPIO 18 is also known as pin 12 and PCM_CLK. This particular pin (around halfway down the right side of the GPIO port) is capable of hardware pulse-width modulation (PWM), and is useful for controlling LED lights and motors among other things.

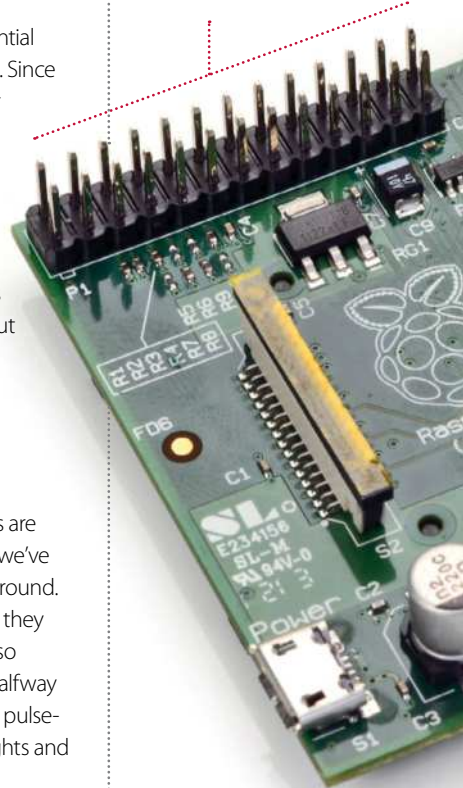
The pin-naming convention you use in your Python scripts can be set manually. This can either be set as BCM (the Broadcom pin name) or the physical pin locations (BOARD).

You'll see in any of the projects where we're using the GPIO port, the following line with either BCM or BOARD in the brackets:

```
GPIO.setmode(GPIO.BCM)
```

The easiest way to deal with the GPIO pin-naming issues is to pick a convention and stick with it.

Below Become familiar with the layout of the GPIO pins and what they do – some have special purposes, such as ground pins



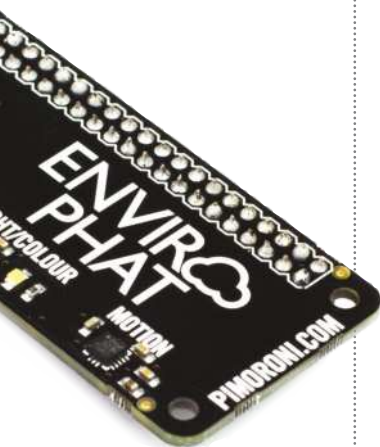
Did you know...

The RPi.GPIO library has some really useful documentation. All you need to do is click on the 'wiki' on Sourceforge.

What you'll need...

Raspberry Pi

Pimoroni Enviro pHAT



Did you know...

The RGB colour model is an additive colour model in which red, green and blue light is combined in various amounts to create a wide range of colours, over sixteen million different variations. The lowest value for a colour is zero which usually denotes black, as black is not a colour but an absence of colour! The top value is 255, for example, red 255 means that the maximum amount of red is being detected by the sensor.

Get to grips with the Enviro pHAT

Take weather-related readings such as temperature and pressure, track movement, measure light and even identify the colour of an object

Pimoroni makes a wide range of HAT (Hardware Attached on Top) boards for the Raspberry Pi. These boards are designed to be fully compatible with all Raspberry Pi models. Enter the Pi Zero, a super-slim, super-cheap, \$4 version of the Pi. In response, Pimoroni created a range of HATs that fit the Pi Zero, called pHATs. This tutorial focuses on the Enviro pHAT, a nifty little board that packs in four different sensors that let you measure temperature, pressure, light levels, colour, acceleration, take a compass heading and retrieve readings from analog inputs. This creates up to ten different variables, making it ideal for monitoring conditions in your house, garage or garden.

Installation

01 Attach your Enviro pHAT onto the GPIO pins and push down to secure in place. Then start up your Raspberry Pi and install the required libraries. Pimoroni makes this very easy, open the LX Terminal and type:

```
sudo apt-get update
sudo apt-get upgrade
curl -sS https://get.pimoroni.com/envirophat | bash
```

This downloads and installs the required software. It also includes a folder that contains example programs, which you can use to get started with the Enviro pHAT.

Your first program, LED on

02 Open the LX Terminal, and type:

```
sudo idle3
```

...on line 3. Add a short pause, line 4, before turning the LEDs off, line 5. Save and run your program; the LEDs will blink once.

```
import time
from envirophat import led
leds.on()
time.sleep(5)
leds.off()
```

Take a temperature reading

03 Start a new Python window and import the weather module from the Python library; type:

```
from envirophat import weather
...on line 1. Next, create a variable called temperature to store the temperature reading. Use code:
```

```
weather.temperature()
...on line 2. This takes the current temperature and then saves this value
```

into the variable. Finally, print out the reading using a print statement, line 3. Save the program and run it; the temperature will be on your screen.

```
from envirophat import
weather,
temperature = weather.
temperature()
print("{} degrees celsius".
format(temperature))
```

Read the pressure

04 The EnviroPHAT comes equipped with a digital barometer to take pressure readings. This is easily achieved with a few lines of code. Import the weather module, which includes the lines of code required for reading the pressure, line 1. On line 2, create a new variable called pressure. This will hold the pressure reading from the sensor:

```
weather.pressure
to take the reading, line 2. Finally, on
line 3, print out the reading:
from envirophat import
weather,
pressure = weather.pressure()
print("{} Pascal".
format(pressure))
```

Sense movement: part one

05 The accelerometer can be used to detect motion including across three axes and plot a compass reading. Start a new program by importing the time and motion modules. Next, create a threshold value that holds the amount of motion that will trigger the detection. This means that you can detect small wobbles or larger movements! Create a list called readings to store the motion readings. Create a variable called last_z which holds the position of the z axis. Set the z axis to an initial value of zero to denote that no movement has yet been detected.

```
import time
```

```
from envirophat import motion
threshold = 0.2
readings = []
last_z = 0
```

Sense movement: part two

06 On the next line, create a while loop to keep checking for movement, then take a reading of the position of the z axis and add it to the 'readings list' using the code readings.append, on line 2. On line 4, take an average of the readings, and then assign this value back the variable 'z'; this is used to compare the new value with the previous value. Note that lines 3, 4 and 5 are indented.

```
while True:
    readings.append(motion.
accelerometer().z)
    readings = readings[-4:]
    z = sum(readings) /
len(readings)
```

Sense movement: part three

07 Create a condition using an if statement to compare if 'the last value of z is greater than zero' and that the 'absolute value of the new value of z minus the previous value is greater than the threshold value'. If the value meets these criteria, then the Enviro pHAT has moved to a new position. Inform the user with a simple print statement, line 2. On line 3, update the variable last_z with the current z reading. Take a small pause before checking again for movement. The program then checks again to see if the z value has changed again and is within the threshold tolerance.

```
if last_z > 0 and abs(z-
last_z) > threshold:
    print("Motion Detected!!!")
    last_z = z
    time.sleep(0.01)
```

Take a light reading and sense colour: part one

08 This final program takes a reading of the amount of light that is in the room or surrounding environment. Start a new window and import the time and the light module for the program, lines 1 and 2. Create a variable called 'amount_of_light'; this stores the light reading. To take the reading use this code:

```
light.light()
Then store this in the variable. Then,
on the next line down, create three
variables to store the amount of red,
green and blue light that is sensed. To
take a colour reading of the light use:
light.rgb()
...on line 4.
import time
from envirophat import light
amount_of_light = light.light()
r, g, b = light.rgb()
```

Take a light reading and sense colour: part two

09 The last step is to print out the values of light and colour that have been read and stored in the four variables. Display these readings using

```
print("Amount of light",
amount_of_light)
...which prints out a message
followed by the reading. The values
shown are the amount of white light
and the amount of red, green and
blue light (RGB) between a value of
zero and 255. Try running the program
while moving your hand over the
sensors. Also, try placing a coloured
object over the sensors to change the
colour reading. This is similar to the
paint-matching software that is used
in hardware stores.
```

```
print("Amount of light",
amount_of_light)
print ("Colour", r, g, b)
```


What you'll need...

A second computer

External storage

Did you know...

SD cards don't last forever so always make sure you've got a backup of your files and your entire OS.

Back up your Pi

Take the initiative and back up your Raspberry Pi to make sure you never lose files again

While the Raspberry Pi is a very solid piece of kit, failure can happen, so it's best to be well prepared and keep your files safe.

The good news is that the Pi's files are all kept externally on the SD card. If your Pi breaks, everything will still be available on the SD card and accessible from elsewhere. The SD card is still susceptible to problems, though. There are a number of ways to back up a Pi.

The methods can be broken down into two main categories: saving the important files and creating an exact copy of the state of the SD card. The former involves having copies of files elsewhere, while the latter has you create the same kind of image that you'd normally write to the SD card when installing Raspberry Pi OS or other Pi-operating systems.

Important files

To save important files, we need to create a copy on an external source, such as external hard drive or another PC. One of the best methods to do this doesn't even involve a Pi; all you need is a PC or laptop with a card reader and you're good to go. Turn off your Pi, unplug it and remove the SD card from the slot. Find the SD card reader on your PC and slot it in. The main file system of the SD card can be read

by Linux PCs by default, and a Windows or Mac computer once you've installed a program that lets it read the ext file system, such as Ext2Fsd. On Windows, the SD card will be listed with the rest of the drives under My Computer (Fig 1). On Linux and Mac, it will be listed wherever storage is shown on the menus and file managers.

Once you've found the SD card on here, open it up and navigate to **home** and then **pi**. This is where the Documents, Downloads, Desktop and other directories can be found. All you need to do is select the files you want to copy and move them to a secure directory on your PC or a connected external hard drive.



Back up your Pi

If you want to keep the files on another computer, that's fine, but it will be prone to the exact same problems as the Pi in the long run. Keeping them on an external hard drive is a good idea, and putting them on a cloud storage service is better yet, enabling you to access them from anywhere with an internet connection (Fig 2).

Cloning

Creating a clone depends on what operating system you're using on your main computer. For Macs and Linux, you can use a simple command-line tool called **dd** to create an exact copy of the SD card (Fig 3). This is done in the terminal emulator or command line, so bring that up first. Make sure the SD card is plugged in and enter:

```
$ fdisk -l
```

This lists all connected storage devices. The SD card will have 2, 4 or 8GB of space, depending on its size. It'll likely be listed as something like `/dev/sd[x]`, where `x` is the letter the computer attaches to the SD card. To copy it using **dd**, enter the following into the terminal:

```
$ dd if=/dev/sd[x] of=backup.img bs=1M
```

You can also add a path to the image you're creating to put it in a specific folder. The process will take some time, and will produce a multi-gigabyte file which you can then write onto the SD by reversing the previous command:

```
$ dd if= backup.img of=/dev/sd[x] bs=1M
```

While this is useful as a backup, you can also use the image to mass-produce SD cards to give to friends or keep in the various places where you use your Raspberry Pi.

Windows

To create a clone on Windows, we can use the Win32 Disk Imager (Fig 4). Download it from here to install it: bit.ly/L8JdYG.

Once installed, insert the SD card and launch the software. Choose a name for the backup file and select the SD card from the list of devices. Now press Read and it will create the backup file. Again, this might take a while; however, this time you are at least shown a progress bar.

Storing your cloned image is a little more difficult than your important files – the size of the image being in the gigabytes means it will fill up a lot of cloud-storage services. If you have the space, definitely keep it on there; however, you may need to put it on an external hard drive.

Projects



Fig 1: Accessing the SD card from another PC is an easy alternative to transferring files between machines via a USB stick

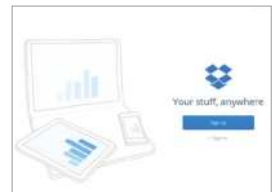


Fig 2: Cloud storage services make backing up files easy and secure, as they're a lot less prone to problems



Fig 3: The **dd** tool is what you'll use on a Linux and Mac computer, and it's available by default in the terminal emulators



Fig 4: Win32 Disk Imager makes backing up the entire OS easy, and you can even use it to write the image back to the SD card

What you'll need...

Full nano manual

www.nano-editor.org/dist/v1.2/nano.1.html

Did you know...

Nano is one of the best command line editors for beginners, but there are lots of options like Kate and Vim.

Beginner's guide to nano

Learn how to edit text on the command line and in a terminal with one of Linux's best tools

If you have the itch to do more with your Pi, one of the skills you'll need to learn to pull off many projects is the ability to edit system files. The command line text editor nano is definitely one of the best tools for the job. Text editors are very basic tools; the clue is in the name in this case. There's no formatting or colouring or anything of the sort you would get in a word processor, but that's the point. The kind of files you'll be creating or editing will generally contain code – code which doesn't require to be made bold or bulleted. nano removes all of these distractions, but still has a few of the more handy features you'd find in a graphical text editor. We'll teach you how to make the most out of nano to make your projects run quickly and efficiently.

Writing

Create plain text files in the command line, even write some code for a program

Editing

Edit system files to suit your needs and projects without digging through a file manager

Advanced functions

Search, copy, paste and insert text from another file using some of the built-in nano functions

```

# Imported fast libtcl to 20 Kallman's library class/
# By Al Nisgort alnisgort@python.com
# http://www.python.org
# Released under a "Modified BSD" license

from pygame.locals import *

FPS = 30 # frames per second to update the screen
WINDOWWIDTH = 640 # width of the program's window, in pixels
WINDOWHEIGHT = 480 # height in pixels
HALF_WINDOWWIDTH = int(WINDOWWIDTH / 2)
HALF_WINDOWHEIGHT = int(WINDOWHEIGHT / 2)

FRAMESCOLOR = (24, 255, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)

CAMERASPACE = 90 # how far from the center the squirrel moves before moving the camera
MOVESPEED = 8 # how fast the player moves
BOUNCESPEED = 8 # how fast the player bounces (large is slower)
BOUNCEHEIGHT = 30 # how high the player bounces
STAYTIME = 20 # how long the player stays off
WINNERS = 1000 # how big the player needs to be to win
DANGERTIME = 2 # how long the player is invulnerable after being hit in seconds
CAMERATIME = 4 # how long the "game area" will stay on the screen in seconds
MOVESPEED = 3 # how much health the player starts with

# number of grass objects in the active area
# number of squirrels in the active area
# slowest squirrel speed
# fastest squirrel speed
# chance of direction change per frame

LEFT = 'left'
RIGHT = 'right'

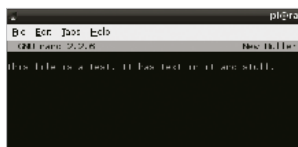
"""
This program has three data structures to represent the player, enemy squirrels, and grass background objects. The data structures are dictionaries with
three used by all three data structures:
    'x' - the left edge coordinate of the object in the game world (not a pixel coordinate on the screen)
    'y' - the top edge coordinate of the object in the game world (not a pixel coordinate on the screen)
    'rect' - the pygame.Rect object representing where on the screen the object is located.
Player data structure keys:
    'surface' - the pygame.Surface object that stores the image of the squirrel which will be drawn to the screen.
    'facing' - either set to LEFT or RIGHT, stores which direction the player is facing.
    'size' - the width and height of the player in pixels. (The width & height are always the same.)
    'bounce' - represents of what point in a bounce the player is in. 0 means standing (no bounce), up to BOUNCESPEED (the completion of the bounce)
    'health' - an integer showing how many more times the player can be hit by a larger squirrel before dying.
Enemy Squirrel data structure keys:
    'surface' - the pygame.Surface object that stores the image of the squirrel which will be drawn to the screen.
    'move' - how many pixels per frame the squirrel moves horizontally. A negative integer is moving to the left, a positive to the right.
    'move' - how many pixels per frame the squirrel moves vertically. A negative integer is moving up, a positive moving down.
    'width' - the width of the squirrel's image, in pixels.
    'height' - the height of the squirrel's image, in pixels.
"""

```



Open nano

01 Open the terminal, or enter the command line, and simply type **nano**. This will open a blank new file. From here you can create a simple text file such as a list, or create a system file, script or piece of code. How the system interprets your file depends on what you write and how you save it.



Save and continue

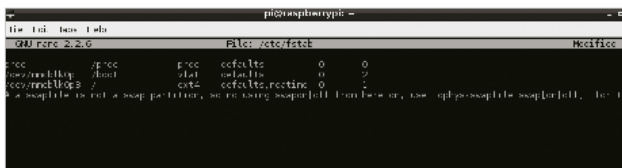
02 Once you've finished with nano, you can save/write out using **Ctrl+O**. All shortcuts and functions such as this are done using **Ctrl** and a letter key. It will ask you what name to save the file under. Whatever you name it, it will be saved in the directory you opened nano from unless you specify a path.



Opening files

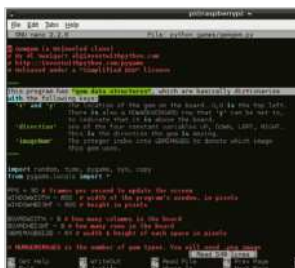
03 To edit already existing files, you'll first need to know their location and name. To open them in nano, type the following:

```
$ nano /path/filename
For example, to edit fstab you would type:
$ nano /etc/fstab
```



Save and exit

04 Once you've finished modifying the file and need to get on with the next task, you can press **Ctrl+X** to exit nano. It will ask if you want to save any modifications, which just requires a **Y** to confirm. If you want to exit without saving changes, you can just use **N**. To cancel before making a decision, use **Ctrl+C**.



Copy and paste

05 This is more of a terminal-specific command, but it can be used just as well in nano. While you cannot use your mouse to navigate around the file in nano, you can highlight text in the same way you would in a graphical text editor. Once highlighted, pressing **Ctrl+Shift+C** will copy any text. **Ctrl+Shift+V** will paste.

Insert from file

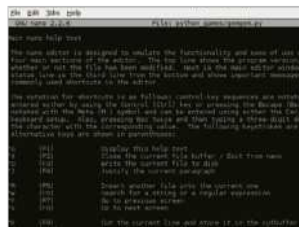
06 If you need to directly insert the contents of another file, there's a quick way to do it without needing to use copy and paste. Typing **Ctrl+R** and entering the path to the file will insert it into the spot your cursor is at.

Advanced navigation

07 You've probably been using the arrow keys to move one space at a time left, right, up or down. There are other ways to move around the file, though. Using **Ctrl+A** will act the same way as the Home key does in a graphical editor, moving the cursor to the start of the line. Same with **Ctrl+E** moving you to the end like the End key. **Ctrl+V** is page down, and **Ctrl+Y** is page up.

Searching a file

08 Sometimes you'll be looking for a specific line or phrase in a large text file. Instead of using the arrow keys and tirelessly reading every line, you can use the search function via **Ctrl+W**. Entering the search term will begin looking through the document, and once you're finished you can press **Ctrl+C** to exit the search.



Extra help

09 There are many more functions available in nano. For a full list of commands, you can use **Ctrl+G**, which lists all the shortcuts and what they do. The caret symbol (^) in this list denotes the use of **Ctrl** on your keyboard.

What you'll need...

TightVNC

www.tightvnc.com/release-2.7.php

Did you know...

TightVNC Viewer is a free resource for accessing VNC servers, like the one we'll install in this tutorial.

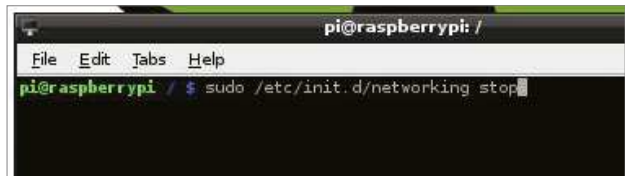
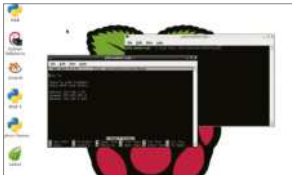
Gain remote desktop access

Learn how to get access to your Raspberry Pi desktop without it being in front of you...

VNC (Virtual Network Computing) is a graphical desktop access and sharing system that allows a user to remotely control and use the desktop of another computer from their own system.

It's handy in many ways: to give you control over a remote system; to help out another user elsewhere; to allow a computer to remain powered on, but without the need for a keyboard, mouse or even a monitor.

In our case, we aim to allow access to the Raspberry Pi desktop without it being hooked up to the aforementioned peripherals. This way, you can do everything you would normally do, but with just the power supply and network attached, thereby freeing up space and saving extra expense. In real-world terms, this means you could potentially access the Pi desktop via an Android tablet or phone.



Static IP address

01 The first step for us is to make sure that the Raspberry Pi has a static IP address. Basically, an IP address is a group of numbers that your network assigns to devices in order to tell them apart. To set up a static IP address, simply double-click LXTerminal and type the following and then press Enter:

```
sudo nano /etc/network/
interfaces
```

Static IP part 2

02 This file controls the IP addressing for the Pi. You need to scroll down to the 'iface eth0' line and remove DHCP and replace it with static. Now, on the line directly below, enter the IP address that you want force your Pi to have, along with the subnet mask and the gateway:

```
address 192.168.1.93
netmask 255.255.255.0
gateway 192.168.1.254
```

Static IP part 3

03 After you've entered those details, exit nano by pressing Ctrl+X, followed by Y to accept the changes, and then press Enter a couple of times to get back to the command prompt in the terminal. You can now either reboot your Pi, or type the following into the terminal:

```
sudo /etc/init.d/networking stop
sudo /etc/init.d/networking
start
```

Installing VNC part 1

04 Now we'll install VNC and ensure that it starts automatically whenever the Pi is booted up. This used to be a bit annoying under older Raspbian versions, as configuring services often had a nasty habit of breaking the system. But no longer. Enter the following commands, pressing Enter after each one:

```
sudo apt-get update
sudo apt-get install tightvncserver
tightvncserver
```

```
pi@raspberrypi: /
File Edit Tabs Help
pi@raspberrypi / $ sudo apt-get update
Hit http://archive.raspberrypi.org wheezy InRelease
Get:1 http://mirrordirector.raspbian.org wheezy InRelease [14.9 kB]
Hit http://archive.raspberrypi.org wheezy/main armhf Packages
Get:2 http://mirrordirector.raspbian.org wheezy/main armhf Packages [7,414 kB]
Hit http://archive.raspberrypi.org wheezy/main Translation-en GB
```

Installing VNC part 2

05 When the packages have downloaded and installed, follow the instructions on screen (see below) to set up a password and confirm it, but answer 'n' to the view only option. This really is just a security feature and since you are only accessing the Pi at home, it's not absolutely necessary – it's still good practice, though.

```
You will require a password
to access your
desktops
Password:
Verify:
Would you like to enter a
view-only
password (y/n)? n
New 'X' desktop is raspberrypi:1
```

Configuring VNC server

06 That's the VNC server installed, up and running. Now we need to make sure it loads up as a service every time the Raspberry Pi reboots, so you can access it even if the Pi undergoes a power cycle. To configure the Pi to do this, type the following in the terminal and press Enter.

```
sudo nano /etc/init.d/
tightvncserver
```

Configuring boot service

07 We're back in nano, and we'll need to enter some lines of commands in order to allow the Raspberry Pi to activate the VNC server when it boots. In the editor, type:

```
#!/bin/sh
# /etc/init.d/tightvncserver
# Set the VNCUSER variable to
the name of the user to start
tightvncserver under
VNCUSER='pi'
case "$1" in
start)
su $VNCUSER -c '/usr/bin/
tightvncserver :1'
echo "Starting TightVNC
server for $VNCUSER"
;;
stop)
kill Xtightvnc
echo "Tightvncserver stopped"
;;
*)
echo "Usage: /etc/init.d/
tightvncserver
{start|stop}"
exit 1
;;
esac
exit 0
```

Reboot and ready to go

08 Now press Ctrl+X, then Y to save, followed by Enter a couple of times to get you back into the Terminal. What we need to do now is edit the permissions of the script we've just created so that it's executable and active. Do this by typing the following commands into the terminal, ensuring that you press Enter after each one otherwise it will not be registered:

```
sudo chmod 755 /etc/init.d/
tightvncserver
update-rc.d tightvncserver
defaults
sudo reboot
```

Once you have completed these steps, the final thing that you will need to do is to unplug the Raspberry Pi and locate it somewhere that has easy access to a network cable. If you install the likes of TightVNC Viewer, or any other remote access software (as long as it uses the Tight protocol) then you should be able to point the client to the IP address 192.168.1.93:1 (or whatever the static IP address is of the network that you wish to connect the device to) and have full access to the Raspberry Pi.

```
pi@raspberrypi: /
File Edit Tabs Help
GNU nano 2.2.6  File: /etc/init.d/tightvnc
#!/bin/sh
# /etc/init.d/tightvncserver
# Set the VNCUSER variable to the name of the user to start
VNCUSER='pi'
case "$1" in
start)
su $VNCUSER -c '/usr/bin/tightvncserver :1'
echo "Starting TightVNC server for $VNCUSER"
;;
stop)
kill Xtightvnc
echo "Tightvncserver stopped"
;;
*)
echo "Usage: /etc/init.d/tightvncserver {start|stop}"
exit 1
;;
esac
exit 0
```

Raspberry Pi plus Arduinos

Find out just how to get your Raspberry Pi to talk to and use your Arduino

The Raspberry Pi is a truly amazing single-board computer that gets used in lots of DIY projects. That has been the basis for this whole column and the previous several articles. While the Raspberry Pi has a GPIO and can communicate with sensors and actuators, you may have cases where you want to use your Raspberry Pi as the brains of your project and offload the interactions with the physical world to another system.

This is usually handled by one of the many microcontroller boards that are available. In this issue, we will specifically use the Arduino board and see how to connect it to a Raspberry Pi and how to handle the communications between the two. As always, we will be using Python as the language of choice to handle all of the actual coding in the examples here.

The Arduino is an open source prototyping platform defined as a specification. This means that you can get Arduino implementations from several different manufacturers, but they should all behave in a similar fashion. For this article, the assumption will be that whatever implementation you wish to use will behave properly. The first step is to connect the two boards together. You

will probably want to use a powered USB hub to connect them since the Raspberry Pi can't provide huge amounts of power through its USB port. While they are connected over USB, the Arduino will appear as a serial port to the Raspberry Pi.

This means that you can communicate with the Arduino directly over the serial connection. To be completely sure you have all of the relevant libraries installed, you can simply install the Arduino IDE with the command:

```
■ sudo apt-get install arduino
```

This will make sure that you are starting with all of the core software that you might need. When you plug in your Arduino, you need to know over which port communications will happen. The specific port name will vary based on the exact version of Raspberry Pi and Arduino that you are using. However, it should be something like `/dev/ttyUSB0` or `/dev/ttyACM0`. In the example code below, we will be assuming that the Arduino is visible on the port `/dev/ttyUSB0`.

Once you have the two devices connected, you can start writing code to have them talk to

“For this article, the assumption will be that whatever implementation you wish to use will behave properly”

each other. We will start with the most low-level protocols and build upwards from there. The first step is to open a serial connection to the Arduino.

In order to do this, you will need to make sure that the Python serial module is installed. If you are using Raspberry Pi OS, you can do this with:

```
sudo apt-get install python-serial
```

You then need to create a new Serial object connected to a given serial port, along with the speed you need to use.

```
import serial
ser = serial.Serial('/dev/ttyUSB0', 9600)
```

In the above example, the speed is 9600 baud (bits/sec). With this Serial object, you can read and write data to and from the Arduino. But you need code on the Arduino to handle its part of this communication. The Arduino has its own programming language, based on C, which you use to write the code that will run on the board. The way Arduinos work is that at bootup it will load a program that will run as long as it is powered up. As a simple example, the following code will listen on an input pin to see if it goes high. If so, it will then fire off a message on the serial port.

```
int pirPin = 7;
void setup() {
  pinMode(pirPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  if (digitalRead(pirPin) == HIGH) {
    Serial.println("High");
  }
  delay(500);
}
```

To load this program to your Arduino board, you will need to use the Arduino IDE that was installed at the beginning of this article. This is a graphical program, so you will need to connect your Raspberry Pi to a monitor if you want to do this step using it. Otherwise, you can do this programming of your Arduino using your regular desktop. If you are using the standard bootloader on most Arduinos, it will start up whatever program was last uploaded to it. This way you can use your desktop to upload your code and then connect it to your Raspberry Pi later on. Moving back to the Raspberry Pi, how can you read this message from the Arduino? You can simply do a read from the Serial object that you created earlier.

```
import time
while True:
  message = ser.readline()
  print(message)
  if (message[0] == 'H')
    do_something_useful()
    time.sleep(.5)
```

As you can see, we imported the time module in order to be able to sleep in the loop between attempts to read from the serial port. What about sending instructions out to the Arduino? This is also requires Arduino code to be uploaded ahead of time. For example, the following code will take an input number and flash an LED that number of times

```
int ledPin = 13;
void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}
void loop() {
  if (Serial.available()) {
    flash(Serial.read() - '0');
  }
}
```

```

delay(1000);
}
void flash(int n) {
  for (int i = 0; i < n; i++) {
    digitalWrite(ledPin, HIGH);
    delay(100);
    digitalWrite(ledPin, LOW);
    delay(100);
  }
}

```

Then, you can send a count from your Python code with something like:

```
ser.write('5')
```

This will flash the LED 5 connected to pin 13 on your Arduino five times. One missing element on the Raspberry Pi is an analogue-to-digital (ADC) converter to take a given voltage and turn it into a number that can be used in some piece of control software. This is where attaching an Arduino can be extremely helpful, as it has a 10-bit ADC converter included. The following code will read the voltage on pin 3 and then send it out over the serial connection to the Raspberry Pi.

```

int analogPin = 3;
int val = 0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  val = analogRead(analogPin);
  Serial.println(val);
}

```

This maps the measured voltage to an integer between 0 and 1,023. The minimum voltage is zero, while the maximum voltage can be set with the function `analogReference()`. By default, the maximum is the power supplied to the board (5

volts for 5V boards, or 3.3 volts for 3.3V boards).

You can also use two internally supplied reference voltages, one at 1.1 volts and a second at 2.56 volts. For special cases, you can supply an external reference voltage to the AREF pin. You need to be sure that it is only between 0 volts and 5 volts.

Going in the opposite direction, you can use the Arduino to supply an output voltage. This is done by using a PWM (pulse width modulation) output signal. The idea is to actually send out a number of pulses with some duty cycle that is on for some percentage of the time and off for the remainder of the time. For example, if you have an LED connected to one of the pins, you can light it at half brightness with the following code.

```

int ledPin = 9;
void setup() {
  pinMode(ledPin, OUTPUT);
}
void loop() {
  analogWrite(ledPin, 127);
}

```

The second parameter to the `analogWrite()` function is a value between 0 and 255, which defines the duty cycle between 0% (or fully off) to 100% (or fully on). This analog output signal stays on at the given duty cycle until a new call to the `analogWrite()` function. By having your Raspberry Pi write out values over the serial connection, it can then control the output duty cycle by sending a simple integer. This short article will spark some ideas on how you can start combining multiple computing platforms to expand the capabilities of your own projects.

There is no reason to try to find the one silver-bullet platform for your project when you can pick the sub-modules that actually do their own individual jobs best and build up the complex behaviour you need from these simpler parts.

PyFirmata can help even more

Discover ways to make interacting easier

While you can write your own code to run on the Arduino, there are several projects that can be uploaded to it to make interacting a bit easier. One of these is the Firmata project, which includes a Python module to help you talk to the Arduino. The first step will be downloading the Firmata Arduino code and uploading it to your Arduino, most easily done with a desktop computer. The code is available at github.com/firmata/arduino. There are a few different versions available, but for these examples you should upload the StandardFirmata sketch with the Arduino IDE. There are client libraries available for many different programming languages, including several for Python. The one we will look at using is pyFirmata. You can install it on your Raspberry Pi with:

```
sudo pip install pyFirmata
```

You can now use Firmata to act as a sort of remote control to the Arduino port, where your Python code can get almost direct access to all of the functionality available. To get started, import the pyFirmata module and create a new Arduino object connected to the relevant serial port:

```
import pyfirmata
board = pyfirmata.Arduino('/dev/ttyUSB0')
```

You can now access digital I/O pins directly. For example, the following code would write a 1 to pin 10.

```
board.digital[10].write(1)
```

When you want to read from a pin, you have the possibility of overflowing input buffers. To deal with this issue, you can create an iterator object and start it before doing any reads, using code like that below.

```
it = pyfirmata.util.Iterator(board)
it.start()
```

You can now get selected pins for either input or output. The following code will get pin 4 for digital input and pin 12 for analogue PWM output.

```
pin4 = board.get_pin('d:4:i')
pin12 = board.get_pin('a:12:p')
```

You can then read and write with these new pin objects with the related methods:

```
val = pin4.read()
pin12.write(100)
```

When you are done, don't forget to reset any output pins back to 0 volts, and then you can close down the connection with:

```
board.exit()
```

"There are a few different versions available, but for these examples you should upload StandardFirmata sketch with the Arduino IDE"

What you'll need...

Scratch

Internet connection

Did you know...

The website for Scratch, www.scratch.mit.edu, contains lots of programs and games other users have made.

Program with Scratch

An interactive guide to coding with the Pi's graphical programming language

Would you like to delve into the world of animation and game creation? Do you want to bring your creative ideas to life without learning a software-development language? With Scratch you can do all this, and much more.

Scratch 1.4 is already installed on the official 'wheezy' Raspbian operating system image. If your Raspberry Pi doesn't already have Scratch installed, don't worry, just hop on over to the official MIT Scratch website to find the download and install instructions.

To begin, all we need to do is open the Scratch Studio. Click on Scratch's cat icon on the desktop, or find Scratch in the LXDE desktop menu.

The Scratch Studio is a complete development environment. It's divided up into a number of separate panels. Each panel has a specific role in the app-construction process and its own specific set of features and tools.

Scratch studio projects

01 Located at the top of the Studio are three quick-access icons and the main menu (Fig 2).

The first globe-style icon sets the language for the Studio. The other two buttons provide rapid access to the project save and share features.

Under the 'File' menu there's a typical set of file-management features to open, save and import Scratch projects. There is also a 'Project Notes' option where we are able to enter feature descriptions and comments.

The 'Edit' menu contains a mixed bag of animation, image and audio-editing tools. While the 'Help' section

provides access to the browser-hosted help pages.

Rather interestingly, the 'Share' menu allows us to share our projects with the whole world. Any Scratch project can be posted onto the Scratch community website via the 'Share This Project Online...' option

and the 'Upload To Scratch Server' form (Fig 3).

Let's load the 'Aquarium' example project. Select the 'Open...' option in the 'File' menu to display the Open Project dialogue. From the list of large buttons on the left, click on the one called 'Examples'. Next, on the right, select the 'Animation' folder with a double click. Then select the '6 Aquarium' item. The open dialogue window contents should look like the one in Fig 4.

With the Aquarium project loaded our Scratch Studio should look similar to Fig 1. We'll begin our Studio tour with the staging area.





Fig 2: Scratch Studio Menu – Scratch Studio's main menu and shortcut icons in action

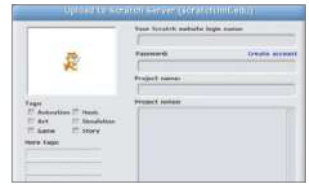


Fig 3: Project Share Dialogue – We can share our projects using the Studio's Share menu



Fig 4: File Open Dialogue – Use the Studio's File menu and 'Open...' to load the Aquarium project

Scratch studio stage

02 The stage is where all the action takes place and is located at the upper right of the Scratch Studio.

The stage is constructed from graphical elements called sprites. Here we have plants, bubbles, fish and other creatures. You can also add and create your own assets with scratch, but we'll come to that later.

At the top of the 'Staging Area' there's a green flag and a red circle. Click on the green flag to bring the aquarium to life. Now spend a little time studying the Aquarium animation. Note the creature's movements and rising bubbles. The red circle icon stops the action.

We can set the view mode with the three buttons located just above the green flag.

The two left-hand buttons increase or decrease the size of the Staging Area panel. A smaller Staging Area means the central area of the Studio increases in relative size compared.

The right-hand button is the Presentation Mode which displays the stage in full-screen mode (see Fig 5). Exit presentation mode with the curly arrow button at the top left, or press the 'esc' key.

Scratch studio sprites

03 Beneath the Staging Area is the collection of sprites for this project. The 'Stage' sprite is separated from the rest. It's a little different to the others and acts as the background image for the stage.

The three buttons across the top of this area offer various ways to create a new sprite. The first button opens up a blank canvas in the Paint Editor. The second button creates a new sprite based in an image file, as selected by the popup file section dialogue window. The third will select a random image from the pre-installed image collection.

We can manage sprites directly from the stage using the four buttons to the right of the main menu. Here we click on a particular button and then a sprite on the stage.

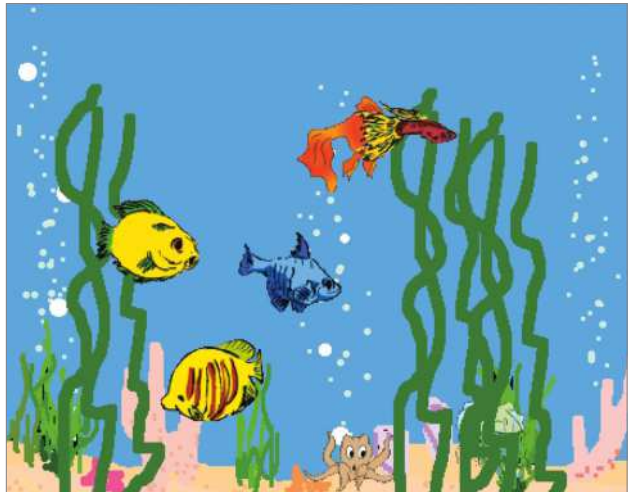


Fig 5: Stage Presentation Mode – The Studio's stage presentation mode is the best way to see the project

Use Scratch blocks and tools

Getting to grips with the Scratch Studio toolbox

Situated in the centre of the Studio is the Edit Panel. The panel contents relate to the currently selected sprite.

Let's start by selecting the jelly fish sprite, called Creature1, from the Sprite Collection area.

At the top we have the sprite's image and name, plus an indication of its current stage coordinates and direction. On the left are three animation control buttons. The top button will rotate the sprite, the second switches between left- and right-facing states, and the third turns animation off.

Below are three Edit Panel tabs. The script tab is where block scripts are created. Here's where we'll drag and drop our blocks, snapping them together in various combinations.

To change a sprite's visual appearance we'll use the 'Costumes' tab. Each sprite can have one or more costumes. For example, the jellyfish has two costumes (Fig 1). Each costume has buttons to edit, copy and delete. New costumes can be painted, imported or captured using the three 'New Costumes' buttons. The sound tab allows us to add audio to our project.



Fig 1: Jellyfish Sprite Costumes – The jellyfish sprite has two different costumes

Scratch block styles

01 The 'Blocks Palette' contains the complete collection of scripting blocks. Blocks come in three basic styles, namely hats, stacks and reporters (see Fig 3 on the opposite page).

A hat-style block will start block script execution based on a specific event. The classic hat block is the 'green flag' click event. However, there are numerous other hat blocks, including hat blocks that start script execution after a specific key press, a mouse click and even following sensor event from the some GPIO connected hardware. As you can probably tell, this offers a lot of options to Scratch programmers.

Reporter blocks allow us to specify textual, numeric and boolean values. They fit into specific shaped 'holes' in other blocks. A rectangular reporter will contain a text string. While the rounded end reporters are associated with numeric values, angle-ended reporters contain boolean true and false values.

Stack blocks are the core script building elements. They interconnect with other blocks via their top-edge notches and bottom-edge bumps. Many stack blocks contain 'holes' for reporter style blocks, which will modify their operation depending on the specified reporter block values.

The Scratch block collection is divided into groups. We select a block group using the eight buttons located at the top of the Block Palette panel, namely 'Motion', 'Control', 'Looks' and so on. These groups are colour coded. Apart from aiding block selection this colour coding provides a visual clue to a block's type when reading a block script in the Edit Panel.



Fig 2: Sound Recorder Tool – Scratch Studio includes a tool to record our own sounds

Scratch block help

02 As we've seen, there are many blocks, each with their own specific functionality and capabilities.

In one way this is great news. A large block collection means Scratch can be used in a vast range of software projects, such as games, animation, music, graphics, math, science, robotics, electronics and much more.

However, the wide selection of blocks can be quite a challenge for the novice Scratch coder. To help with this problem the Scratch Studio designers have included an informative set of block-centric help pages.

A simple right click on any block will display a pop-up help page option. The help page contains context-specific descriptions, graphical images and, where appropriate, a script example of how to use this particular block (Fig 4).

It's a terrific feature which greatly simplifies the process of deciding which blocks to use. More importantly, studying these help pages is a highly effective way to enhance our scripting skills and discover the potential contained within Scratch's feature-rich block collection.

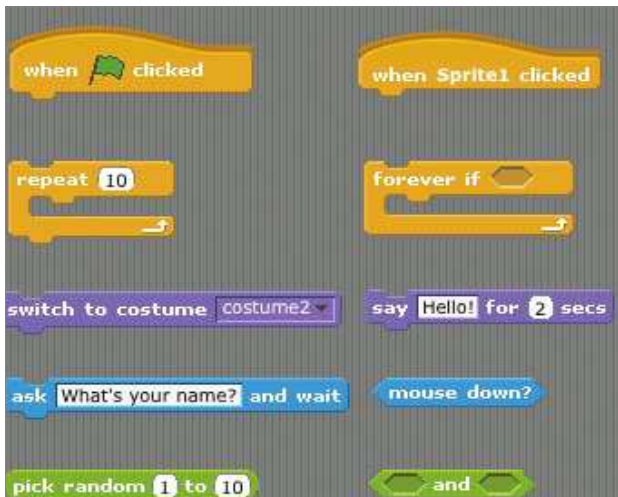


Fig 3: Block Style Examples – the Scratch blocks come in a number of different styles

Did you know...

There is more than one option available to start your scripts beyond the standard Green Flag block.

Block script walkthrough

03 Let's dig deeper into how a block script works in practice. For this, we will use a simple Aquarium project block script. From the 'Sprite Collection' panel select the Stage sprite. Then go back to the central 'Edit Panel' and select the 'Scripts' tab.

There's just a single block script. Starting at the top there's a 'green flag' hat-style block to kick off the activity. Next there's a 'forever loop'. The blocks inside this loop are actioned until the stop button is pressed. This forever loop block contains two other blocks.

The first inner script block selects the next background image. Click on the 'Backgrounds' tab to view all the stage images. The second inner block simply pauses execution for a number of seconds. Setting the value to '1' means that this script will pause for a second before then performing the action specified by the next block.

It's important that you remember these two blocks are enclosed in the forever loop block. So, the stage background images will be displayed in sequence for one second each.



Fig 4: Block Help Window – Example of the help window associated with an 'ask and wait' block

What you'll need...

Scratch project archives
www.scratch.mit.edu/explore/?date=ever

Did you know...

Snake is a very popular beginner game that started life in the arcades in the Seventies and is still popular today.

Create a Snake clone in Scratch

Design your own version of Snake to test your new programming skills!

Here, we will create a version of the classic Snake game where you move the snake around the Scratch stage using the arrow keys. You control the head of the snake and must avoid a collision with either the body of the snake or the edge of the stage.

The snake body grows longer each time you eat an egg. You get points added to your score for eating good yellow eggs and lose points for eating bad black eggs. There are also bonus sprites to eat for extra points.

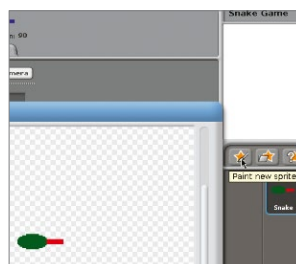
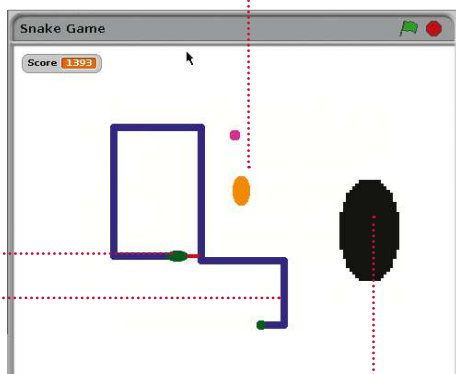
By following this tutorial you will learn to create your own simple sprite graphics, send and receive broadcast events, use a list variable to store data, play sound effects, generate random numbers and use sensing commands to detect when a sprite is touching something.

Egg sprite

The Egg sprite appears randomly on the screen and lets other sprites know when it has been eaten (touched by the snake tongue)

Snake sprite

The Snake sprite moves the head around the stage and draws the body behind it. It also detects collisions with the body or edge



Paint the Snake sprite

01 Click the New Sprite: Paintbrush icon to paint the Snake sprite. In the Paint Editor, draw a small green ellipse for the snake head and add a red rectangle for the tongue. It's important that the tongue is a different colour to the head. Name your sprite Snake.

Tail sprite

The Tail sprite follows the head, erasing the end of the body so the snake moves, and pausing when it needs to grow

Bad Egg sprite

The Bad Egg sprite also appears randomly but decreases the score when eaten. It also grows in size, getting harder to avoid

Add a Snake sound

02 When the Snake tongue touches the snake body or the edge of the stage, we are going to play a Game Over sound. We need to add this sound to the Snake sprite. With the Snake sprite selected, click the Sound tab and choose Import. Select the Electronic>Screch sound.



Respond to arrow keys

03 Drag four when key pressed commands from the Control palette, and four point in direction commands from the Motion palette. Configure them as shown so that the up arrow changes the direction to 0 degrees (up) and so on. Click the green flag above the stage to test this.



Make Snake variables

04 Click on the Variables palette. Make two variables, Score and Speed, that are visible to all sprites. Make a list called Next Direction which is visible to all sprites; it will store the sequence of directions that the head takes. Only have the Score variable checked so it appears on the stage.



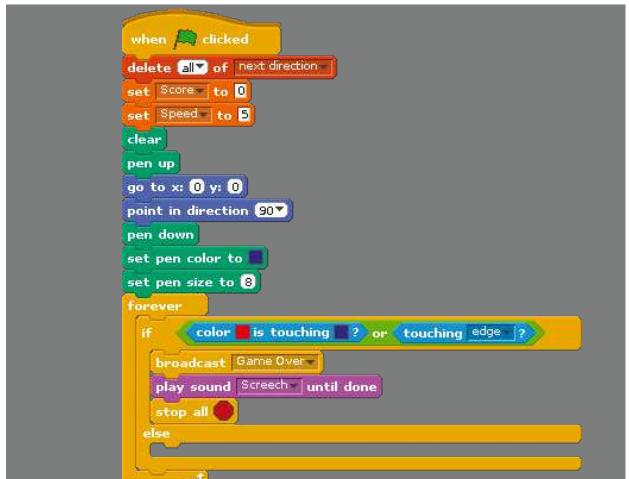
Initialise Snake variables

05 Use a 'when green flag clicked' Control command and initialise the Snake variables as shown, using commands from the Variables palette. We want to start each game with an empty Next Direction list, so delete all of its entries. The Score must start at zero. The Speed sets difficulty.



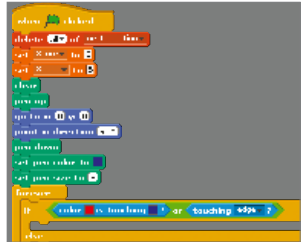
Draw Snake body

06 Use commands from the Pen palette to control the drawing of the snake body. You should also use commands from the Motion palette to move the snake to the centre of the stage and point left at the beginning of each game. The pen is up until the snake is in the starting position. In the next steps we'll use colours to check to see if the head is touching anything it shouldn't.



Add main action loop

07 Use a 'forever' command with an 'if-else' command nested inside. We have a collision if the red tongue is touching the blue body (the head is always touching the body) or the Snake sprite is touching the edge. Use the Eyedropper tool to select colours within Scratch.

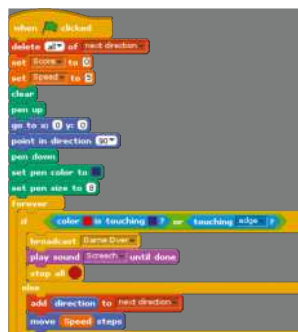


Handle Game Over

08 When a collision has been detected, broadcast a Game Over event (you'll need to create a new event) to the other sprites so they can also react. Also, play the Screch sound effect and stop all scripts so that the snake freezes in its current position at the end of the game.

Handle movement

09 Now handle the typical case where there is no collision and the snake must move in its current direction. The pen is down so it will draw the body. The Speed variable determines how many steps to move. Add the current direction to the Next Direction list for the tail to read.



Try out the snake

10 You can now try out your Snake sprite. It will move around the screen in response to pressing the arrow keys. It will draw its body, which will just get longer and longer because we need the tail to erase it. And it will screech and end the game on detecting a collision.



Paint the tail

11 Click the New Sprite: Paintbrush icon to paint the Tail sprite. Draw a small green circle to represent the end of the tail. Name this sprite Tail. The Tail sprite will follow the Snake and erase the end of its tail so that the snake body doesn't grow indefinitely.

Make a Grow variable

12 Make a Grow variable which is for this sprite only – no other sprites need access to it. The Grow variable is used to determine when the snake body needs to grow and the tail therefore needs to pause before following to allow the head to get further ahead.



Handle events

13 The Tail needs to listen for two new events which you create as you need them. When it receives an Egg Eaten event from one of the Egg sprites, it sets Grow=1 so that the tail can pause. And when it receives a Game Over event from the Snake, it must freeze.

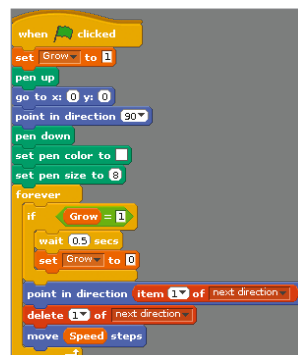


Initialise the tail

14 When the green flag is clicked to start the game, Grow is set to 1 so the snake gets a short body. Move to the centre of the stage with the pen up and configure the pen to draw a trail the same colour and size as the stage background so that it erases the body.

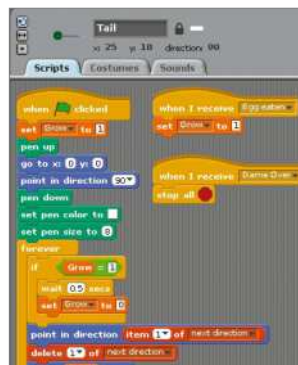
Grow and move

15 Use a 'forever' command to keep the tail moving. If Grow is 1 it should pause and reset Grow to 0 – this makes the body grow longer. Use the first value from Next Direction to set the direction and remove it so you get a new value next time. Move Speed steps.



Try out the tail

16 Now you can try out the Tail sprite. The snake won't keep growing yet because it won't receive any Egg Eaten events. But the tail will follow the snake head around the stage, erasing the snake body as it goes by drawing over it with a white pen (which is the same colour as the background).



Paint the Egg sprite

17 Click the New Sprite: Paintbrush icon to paint a new sprite. In the Paint Editor, draw a small yellow ellipse. Name the sprite Egg. The Egg will appear randomly on the stage and cause the snake to grow and increase its score.



Add Egg sound

18 Go to the Sounds tab for the Egg sprite and import the Percussion>Cymbal Crash sound. Or you can choose a different sound if you like. This sound will play when the snake eats an egg.

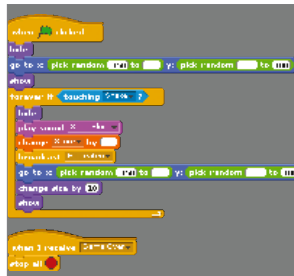


Add Egg scripts

19 Copy the Egg script so that the Egg appears randomly at the start of the game. When the Egg senses that it has been eaten, it must hide, play the Cymbal sound (or whatever you chose in Step 18), update the score, broadcast the Egg Eaten event and then randomly appear again. When the Game Over event is received, it must stop.

Make Bad Egg

20 Create the black Bad Egg in the same way as the Egg but using a different graphic and the Instruments>StringPuck sound. Drag the Egg's green flag scripts onto the Bad Egg to copy them – just change the sound that's played and reduce the score instead of increasing it.



Grow Bad Egg

21 Add another 'when green flag clicked' script to the Bad Egg so it sets its size to the default 100% when a new game is started and then increases its size by ten every ten seconds. The Bad Egg will get bigger and bigger and harder to avoid.



Create Bonus sprite

22 Create the Bonus sprite in a similar way. You can choose the shape and sound for the Bonus. Its scripts are similar to the Egg ones so you could drag one of those to the Bonus sprite and work from that. Make sure you change the sound and increase the score by a random bonus.



Check your mail

With Python, you can have your Raspberry Pi act as mail checker, giving you a running list on incoming email

Since the Raspberry Pi is such a small computer, it gets used in a lot of projects where you want to monitor a source of data. One such monitor you might want to create is a mail-checker that can display your current unread emails. This issue, we'll look at how to use Python to create your own mail-checking monitor to run on your Pi. We'll focus on the communications between the Pi and the mail server and not worry too much about how it might be displayed. That will be left as a further exercise.

To start with, most email servers use one of two different communication protocols. The older, simpler one was called POP (Post Office Protocol), and the newer one is called IMAP (Internet Message Access Protocol). We will cover both protocols to cover all of the situations that you might run into. We'll start with the older POP communications protocol. Luckily, there is support for this protocol as part of the standard library. In order to start using this protocol, you will need to import the `poplib` module, and then create a new POP3 object. For example, the following will create a connection to the POP server available through Gmail.

```
import poplib
my_pop = poplib.POP3_SSL(host='pop.gmail.com')
```

You need to use the `POP3_SSL` class when connecting to Gmail because Google uses SSL for its connections. If connecting to a different email server, you can use POP3 to make an unencrypted connection. The POP communication protocol involves the client sending a series of commands to the server to interact with it. For example, you can get the welcome message from the server with the `getwelcome()` method:

```
my_pop.getwelcome()
```

The first things that you will want to communicate to the server are the username and password for an email account that you are interested in. Having the username in your code is not too much of a security issue, but the password is another matter. Unless you have a good reason to have it written out in your code, you should probably ask the end-user for it. Included within the standard library is the `getpass` module, which you can use to ask the end-user for their password in a safer fashion. You could use the following code, for example.

```
import getpass
my_pop.user('my_name@gmail.com')
my_pop.pass_(getpass.getpass())
```

“The first things that you will want to communicate to the server are the username and password for an email account”

You should now be fully logged in to your email account. Under POP, your account will be locked until you choose to execute the `quit()` method of the connection. If you need a quick summary of what is on the open server you can execute the `stat()` method:

```
my_pop.stat()
```

This method returns a tuple consisting of the message count and the mailbox size. You can get an explicit list of messages with the `list()` method. You have two options for looking at the actual contents of these emails, depending on whether you want to leave the messages untouched or not. If you want to simply look at the first chunk of the messages, you can use the `top()` method. The following code will grab the headers and the first five lines of the first message in the list.

```
email_top = my_pop.top(1, 5)
```

This method will return a tuple consisting of the response text from the email server, a list of the headers and the number of requested lines, and the octet count for the message. The one problem with the `top()` method is that it is not always well implemented on every email server. In those cases, you can use the `retr()` method. It will return the entire requested message in the same form as that returned from `top()`.

Once you have your message contents, you need to decide what you actually want to display. As an example, you might want to simply print out the subject lines for each message. You could do that with the following code.

```
for line in email_top[1]:
    if 'Subject' in i:
```

```
        print(i)
```

You need to explicitly do the search because the number of lines included in the headers varies between each message. Once you're done, don't forget to execute the `quit()` method to close down your connection to the email server. One last thing to consider is how long the email server will keep the connection alive. While running test code for this article, it would frequently time out. If you need to, you can use the `noop()` method as a keep-alive for the connection.

As mentioned previously, the second, newer, protocol for talking to email servers is IMAP. Luckily, there is a module included in the standard library that you can use, similar to the `poplib` module we looked at above, called `imaplib`. Also, as above, it contains two main classes to encapsulate the connection details. If you need an SSL connection, you can use `IMAP4_SSL`. Otherwise, you can use `IMAP4` for unencrypted connections. Using Gmail as an example, you can create an SSL connection with the following code.

```
import imaplib
import getpass
my_imap = imaplib.IMAP4_SSL('imap.gmail.com')
```

As opposed to `poplib`, `imaplib` has a single method to handle authentication. You can use the `getpass` module to ask for the password.

```
my_imap.login('my_username@gmail.com', getpass.getpass())
```

IMAP contains the concept of a tree of mailboxes where all of your emails are organised. Before you can start to look at the emails, you need to select which mailbox you want to work with. If you don't give a mailbox name, the default is the inbox.

This is fine since we only want to display the newest emails which have come in. Most of the interaction methods return a tuple that contains a status flag (either 'OK' or 'NO') and a list containing the actual data. The first thing we need to do after selecting the inbox is to search for all of the messages available, as in the following example.

```
my_imap.select()
typ, email_list = my_imap.search(None, 'ALL')
```

The email_list variable contains a list of binary strings that you can use to fetch individual messages. You should check the value stored in the variable typ to be sure that it contains 'OK'. To loop through the list and select a given email, you can use the following code:

```
for num in email_list[0].split():
    typ, email_raw = my_imap.fetch(num, '(RFC822)')
```

The variable email_raw contains the entire email body as a single escaped string. While you could parse it to pull out the pieces that you want to display in your email monitor, that kind of defeats the power of Python.

Again, available in the standard library is a module called email that can handle all of those parsing issues. You will need to import the module in order to use it, as in the example here.

```
import email
email_mesg = email.message_from_bytes(email_raw[0][1])
```

All of the sections of your email are now broken down into sections that you can pull out much more easily. Again, to pull out the subject line for a quick display, you can use the code:

```
subject_line = email_mesg.get('Subject')
```

There are many different potential items that you could select out. To get the full list of available header items, you can use the keys method, as shown below:

```
email_mesg.keys()
```

Many times, the emails you get will come as multi-part messages. In these cases, you will need to use the get_payload() method to extract any attached parts. It will come back as a list of further email objects. You then need to use the get_payload() method on those returned email objects to get the main body. The code might look like:

```
payload1 = email_mesg.get_payload()[0]
body1 = payload1.get_payload()
```

As with POP email connections, you may need to do something to keep the connection from timing out. If you do, you can use the noop() method of the IMAP connection object. This method acts as a keep-alive function.

When you are all done, you need to be sure to clean up after yourself before shutting down. The correct way to do this is to close the mailbox you have been using first, and then log out from the server. An example is given here:

```
my_imap.logout()
my_imap.close()
```

You now should have enough information to be able to connect to an email server, get a list of messages, and then pull out the sections that

you might want to display as part of your email monitor. For example, if you are displaying the information on an LCD, you might just want to have the subject lines scrolling past. If you are using a larger screen display, you might want to grab a section of the body, or the date and time, to include as part of the information.

“When you are done, you need to be sure to clean up after yourself before shutting down”

What about sending emails?

Find out how to send as well as receive

In the main body of the article, we have only looked at how to connect to an email server and how to read from it. But what if you need to be able to also send emails off using some code? Similar to poplib and imaplib, the Python standard library includes a module called smtplib. Again, similar to poplib and imaplib, you need to create an SMTP object for the connection, and then log in to the server. If you are using the GMail SMTP server, you could use the code

```
import smtplib
import getpass
my_smtp = smtplib.SMTP_SSL('smtp.gmail.com')
my_smtp.login('my_email@gmail.com', getpass.
getpass())
```

This code asks the end user for their password, but if you aren't concerned about security, you could have it hard-coded into the code. Also, you only need to use the login() method for those servers that require it. If you are running your own SMTP server, you may have it set up to accept unauthenticated connections. Once you are connected and authenticated, you can now send emails out. The main method to do this is called

sendmail(). As an example, the following code sends a 'Hello World' email to a couple of people.

```
my_smtp.sendmail('my_email@gmail.com',
['friend1@email.com', 'friend2@email.com'], 'This
email\r\nsays\r\nHello World')
```

The first parameter is the 'from' email address. The second parameter is a list of 'to' email addresses. If you have only a single 'to' address, you can put it as a single string rather than a list. The last parameter is a string containing the body of the email you are trying to send. One thing to be aware of is that you will only get an exception if the email can't be sent to any of the 'to' email addresses specified.

As long as the message can be sent to at least one of the given addresses, it will return as completed. Once you have finished sending your emails, you can clean up with the code:

```
my_smtp.quit()
```

This cleans everything up and shuts down all active connections. So now your project can reply to incoming emails, too.

Projects

What you'll need...

Raspberry Pi OS
www.raspberrypi.org/downloads

SD card

Supercharge your Pi

Supercharge your Pi

Get the most out of your Raspberry Pi with these performance-enhancing tips and tricks

Your Raspberry Pi is plugged in. Raspberry Pi OS is installed on the SD card and you are right in the middle of setting up a wireless print server or building a robot to collect your mail from your doormat. But are you truly getting the most from your little computer? Perhaps you haven't explored the full set of options in Raspberry Pi OS, or you're running the entire OS from SD card, something that can reduce SD card lifespan. Various tools and techniques can be employed to improve performance, from choosing the right hardware to overclocking the CPU. You might even maximise storage space on the Pi's SD card or all but replace it with a secondary device. Use these tips and tricks to reconfigure your Pi setup and optimise software and hardware to ensure you get the best performance.





Use better storage hardware

01 Your choice of storage media can have an impact on your Raspberry Pi's performance, regardless of the operating system. A low capacity SD card with poor error correction, is going to be slower than a larger card with greater resilience, so you need to find the right balance for your project and shop wisely.

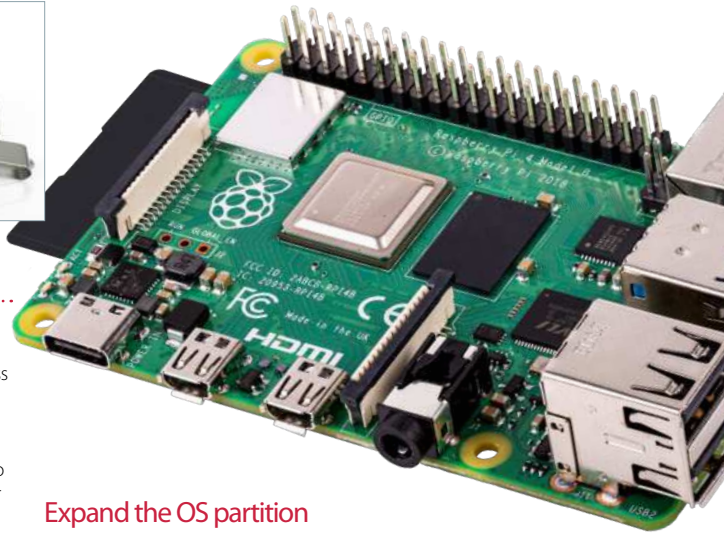
Choosing the best SD card

02 Various standards of SD card are available, with the more expensive designed for better error correction. For the best performance on your Raspberry Pi, choose an SDHC card with a high rating. The same advice applies to MicroSD cards, which you can use on your Raspberry Pi with an SD card adaptor or directly insert into a Raspberry Pi B+.

Make the most of your storage

03 You'll typically need 1-2GB of storage for your chosen Raspberry Pi distro, so any remaining storage on your SD card will be used for updates and data you create or save. In Raspberry Pi OS you can open a command line and run the configuration utility to gain more space (only if your SD card's greater than 2GB):

```
■ sudo raspi-config
```



Expand the OS partition

04 Maximising the partition affords the full capacity of your SD card, which will increase the media's lifespan (there is more space to write too, so the same sectors aren't being overwritten as often). With `raspi-config` running, use the arrow keys to select `expand_rootfs` in the menu. Then wait briefly while the partition is resized.

Write data to RAM

05 Rather than reading and writing data to your SD card – something that will eventually result in a deterioration of reliability and performance in the card – you can configure Raspberry Pi OS to write to the system RAM, which will speed things up slightly and improve the SD card's overall performance. This is achieved using `fstab` (file systems table), a system configuration available in most Linux distros.

Enable fstab in the OS

06 This is much like creating a RAM disk in Windows. In the command line, enter:

```
■ sudo nano /etc/fstab
Add the following line to mount a virtual file system:
■ tmpfs /var/log tmpfs
  defaults,noatime,nosuid,mode=
    0755,size=100m 0 0
Follow this by saving and exiting nano
(Ctrl+X), then safely restarting the Pi:
■ sudo shutdown -r now
```



"You'll typically need 1-2GB of storage for your chosen Raspberry Pi distro"

Projects

Picking an external USB drive

Speeding up your Raspberry Pi by migrating the root filesystem to an external USB drive is a start, but what sort of device should you use for the best performance? With a USB thumb drive you can add flash storage up to 16GB without running into any significant problems (the larger the drive, the greater the current is required to read/write). Anything larger is expensive and unnecessary. If you're planning to use an external HDD, there are no power issues as it will have its own power supply. As ever, your choice should suit your project.

Below Having your filesystem on a USB stick is great for backups as well as performance boosts

Supercharge your Pi

Configure fstab for fast performance

07 Upon restarting, the virtual filesystem will be mounted and /var/log on the RAM disk. Other directories that can be moved to RAM include:

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,size=100m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid,size=30m 0 0
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=100m 0 0
tmpfs /var/run tmpfs defaults,noatime,nosuid,mode=0755,size=2m 0 0
tmpfs /var/spool/mqueue tmpfs defaults,noatime,nosuid,mode=0700,gid=12,size=30m 0 0
```

Add each to /etc/fstab in nano.

Move your OS to a HDD

08 If you're concerned about the lifespan of the SD card, why not reduce your Raspberry Pi's reliance on it? Instead of using the SD card as a sort of budget SSD, change its role and add a HDD or USB stick to run the operating system, leaving the SD card for bootstrapping. This can give a marked performance boost to the SD card.

Back up the SD card

09 Create a copy of your Pi's SD card. Shut down, remove the card and insert it into your desktop computer. In the command line, run:

```
sudo dd bs=4M if=/dev/sdb of=~/.backup.img
```

The path /dev/sdb represents the SD card. Copying takes 5-10 minutes. When complete, remove the SD card and connect your USB device.

Copy Raspberry Pi OS to USB

10 Using a blank Ext4-formatted USB thumb drive (or external HDD) as the destination drive, enter:

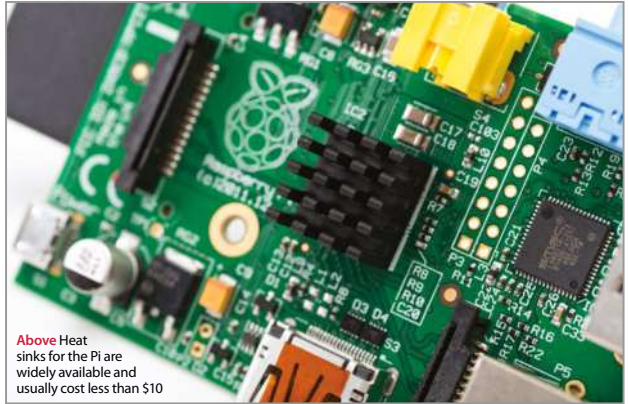
```
sudo dd bs=4M if=~/.backup.img of=/dev/sdc
```

Leave the backup on your computer, just in case something goes wrong. With an SD card and USB storage device sharing an identical disk image, it's time to consider what you're going to do next – create a faster Raspberry Pi.



Split the OS partitions

11 Ideally, the boot partition should remain on the SD card while the root filesystem is run from the external HDD or USB thumb drive. Using your preferred partition manager (Disk Utility is in most distros), unmount and delete the root filesystem from the SD card, ensuring you have retained the boot partition. After removing the SD card, connect your USB device and delete the boot partition, taking care to leave the root filesystem intact. Then resize the root filesystem on the USB device, making sure that 10MB remains.



Identify the root filesystem

12 You're going to have the SD card and the external USB storage connected, so you need to tell the Pi where the root filesystem is. On the desktop Linux computer with your SD card inserted, run:

```
■ sudo nano /boot/cmdline.txt
```

Find `root=/dev/mmcblk0p2` (or similar) and change that to read `root=/dev/sda2` which is your external USB storage. Save and exit.

Add other USB devices

13 You can now restart your Pi with the storage devices attached, but as soon as you connect further USB media you'll suffer problems. Avoid by installing `gdisk`:

```
■ sudo apt-get update
■ sudo apt-get install gdisk
```

Then run `gdisk`:

```
■ sudo gdisk /dev/sdb
```

Enter `?` to display the options and select Recovery and Transformation options (experts only), followed by Load MBR and Build Fresh GPT. Tap `?` one last time and select 'Write Table to Disk' and exit. Remove and replace the USB device and run `gdisk` again. This time enter `l` and then `1` to display the Partition Unique GUID.

Make your Pi fast and reliable

14 Make a note of the GUID and then switch to the SD card. Reopen `cmdline.txt` and change `root=/dev/mmcblk0p2` to `root=PARTUUID=XXXXXX`, where the numerical string from the partition unique GUID should replace the `XXXXXX`. When you're done, save and exit. You can then start your Raspberry Pi. Congratulations, your Raspberry Pi is now faster and more reliable to use.

Boost performance with overclocking

15 Need more from your Pi? It is possible to overclock the computer, although you should be aware of the risks inherent with this activity. You should also ensure that your Raspberry Pi's processor is suitably cooled – heatsinks for the CPU, Ethernet controller and power regulator can be purchased online.

Overclock your Pi

16 Overclocking is available through `raspi-config`. Launch from the command line and arrow down to the overclock option. Four further options are available: Modest, Medium, High and Turbo. With your ideal clock speed selected, exit `raspi-config` and restart your Raspberry Pi to apply:

```
■ sudo shutdown -r now
```

Now you will need to perform tests to see how stable it is overclocked. Raspberry Pi founder, Eben Upton, suggests running Quake 3 as a good stress test. Should the Pi fail to boot, hold Shift to boot without overclocking, run `raspi-config` and select a more modest overclock.

Run Raspberry Pi OS without the GUI

17 Despite these changes, you may find that the GUI remains slow. If you find yourself running a lot of commands in bash, the best thing to do is disable launching into X. In `raspi-config`, choose `boot_behaviour` and select the first (default) option to ensure your Pi boots to the command line. Should you need the GUI, enter 'startx' in Terminal.

What you'll need...

NagiosPi

piimagehub.com/project/nagiospi

Win32 Disk Imager

bit.ly/L8JdYG

Disk Utility

bit.ly/1Lec9r5

Internet connection**4 GB (or larger) SD card**

Monitor your local network with NagiosPi

Embrace the power of Nagios to keep an eye on servers, switches and applications on your network

Is your PC offline? Has your Linux box stopped serving Minecraft or Counter-Strike? If you're out of the house, or even the country, there is no real way of knowing without trying to log in – something you probably won't be able to do without being on the premises (unless you're using remote desktop software).

A far better way would be to simply receive notifications when your network devices are knocked offline, and this is why we turn to NagiosPi, a Raspberry Pi-built version of the popular open source network monitoring tool.

NagiosPi is available as a full image ready to be written to SD card, with the real configuration taking place once it's up and running. Let's get started.

Download NagiosPi

01 Windows users should write the extracted contents of the NagiosPi_v2.0.zip file to a formatted SD card using Win32 Disk Imager. Linux desktop users can use Disk Utility or the command line (bit.ly/1z36sp8). With the image written to SD, safely eject the card and insert it into your Pi before booting.

Log in to NagiosPi

02 As with most Pi projects, you'll probably want to operate via SSH, so check your router's list of connected devices to find the IP address and connect. You can also use a keyboard and monitor connected to your Raspberry Pi. The default username and password for NagiosPi is as follows: pi/raspbery.

Expand the filesystem

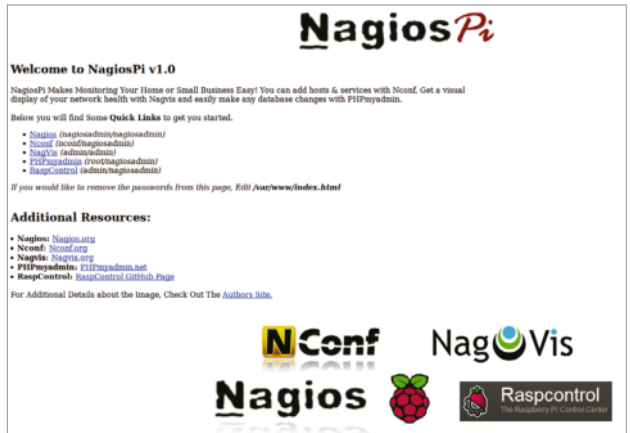
03 Before proceeding, run `sudo raspi-config`. You'll need to select the first option, Expand Filesystem, and wait a moment as the filesystem is expanded to the full size of the SD card.

Once done, select Change User Password to add some security to your NagiosPi, then select Finish and reboot.

Open in your browser

04 With the Pi rebooted, you'll be able to open the NagiosPi web console in your browser. Visit [http://\[your.IPaddress.here\]](http://[your.IPaddress.here]) to see the available options.

Here you'll spot a menu of links in the top-left corner of the page, each accompanied with the username and password to sign in. Start with RaspControl.



Monitor your NagiosPi box

05 In the RaspControl screen you'll get a flavour of just what Nagios can do. On the home screen you'll see general hardware information such as connectivity and system status, and as you flick through Details, Services and Disks you'll see what level of monitoring is possible.



View host status

06 Next, go to Nagios and pick Hosts. Here you will see the current status for the configured hosts, which is a combination of items detected on your local network and preset entities. Look for Current Network Status in the upper-left area of the console, just below this you will find alternate views.

Add a host to monitor

07 Open NConf to add the server you wish to monitor, using the 'Hosts - Add' button to input the device hostname, IP address and alias. Click Submit when done, then switch to 'Services - Add', where you can assign a name and check command (such as `check_ping`) to monitor.

Create configuration file

08 Each check must be set up individually. Some require the installation of NRPE (Nagios Remote Plugin Executor) on remote devices to interrogate and present full system details, but this isn't necessary for basic things like ping.

When you're done, click Submit, then Generate Nagios Config. Following this, select Deploy.

Monitor your server

09 In the NagiosPi window, select Services for a view of currently monitored servers and devices. For each listed device, there will be additional information that you drill down into by clicking Actions. We've only shown you the basics of NagiosPi - investigation will demonstrate just how powerful it really is!

What you'll need...

Android device

USB cable

Tether to an Android device

Need the internet on your Pi? Try out a physical tether to your Android device for online access

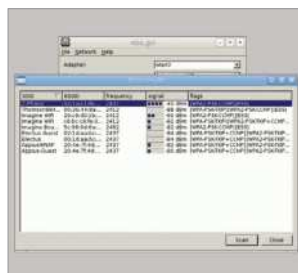
The portability of the Raspberry Pi is one of its most lauded features. Mini screens, mini wireless keyboard and mouse combos, portable batteries and more can get you out and about, but the internet is a stumbling block that you can't easily fix with an accessory. What you do also usually have with you is an Internet-connected magic pocket box called a smartphone that, with a bit of know-how, you can connect the Pi to and steal some internet from. Over the next two pages we will impart this know-how to get you using your Raspberry Pi on the Internet when you're on the go.

The easy way

01 Many smartphones have a Wi-Fi hotspot feature, which your Pi can easily attach to. First of all, turn the hotspot on and then boot into the Pi. Connect a wireless dongle and open up the `wpa_gui` in `Preferences>Wi-Fi Configuration`.

Scan for device

02 Click Scan to open up the scan window and then select Scan again from inside there. It should pick up your device – connect it as you would to any Wi-Fi network and the Pi will remember it for when it needs it next.



Set up tether

03 First connect your phone to your Raspberry Pi via a USB cable – depending on the amount of power your Pi has, it might have trouble charging your phone but it will still let you tether. In the tethering menu you can now activate USB tethering.



```

pi@raspberrypi ~ $ ifconfig
eth0:
Link encap:Ethernet HWaddr b8:27:db:15:01:5b
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 transmit:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 transmit:0
RX bytes:1104 (1.0 KiB) TX bytes:1104 (1.0 KiB)

pi@raspberrypi ~ $ ifconfig
eth0:
Link encap:Ethernet HWaddr b8:27:db:15:01:5b
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 transmit:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 transmit:0
RX bytes:1104 (1.0 KiB) TX bytes:1104 (1.0 KiB)

usb0:
Link encap:Ethernet HWaddr 02:57:07:01:10:39
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 transmit:1000

```

Check connection

04 Your Android device will create an interface known as eth0 on the Raspberry Pi. You can check to make sure this is happening, and that it will let you tether, by opening up a terminal and typing the following:

```
$ ifconfig
```

Test connection

06 There's a few ways to test your connection. We'd usually stay in the terminal and ping www.google.com, which you can do, or you can click on the browser and see if it loads the page.



Interface settings

08 Here you'll find all the current network settings – yours might look different from ours depending on if you have added any fixed wireless settings or passthroughs. Using the same syntax as the eth0 line, add:

```
iface usb0 inet dhcp
```

“What you do also usually have with you is an internet-connected magic pocket box called a smartphone that you can connect the Pi”

Quick connect

05 You can connect from the terminal right now to access the Internet. You should be able to do this by typing the following into the terminal:

```
$ sudo dhclient usb0
```

This will automatically grab any available IP address that your phone will give to it.

Save the settings

07 Once you reboot your Raspberry Pi, it won't remember to automatically connect to the phone's tether. However, we can add an entry to its config so that it will try and do this in the future. From the terminal use:

```
$ sudo nano /etc/network/interfaces
```

Tether on the go

09 After a save and reboot, your Pi should now automatically connect to your phone, whether it's via Wi-Fi hotspot or a physical connection. It may draw a little more charge than usual while tethering, so be sure to keep an eye on your battery level.

What you'll need...

AA battery box
bit.ly/1FDIJGa

3-Amp UBEC
bit.ly/1HLKih7

3-Amp terminal strip

6x AA rechargeable batteries

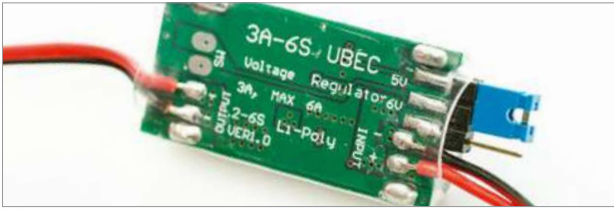


Add a battery pack to your Raspberry Pi

Don't leave your Raspberry Pi behind – incorporate it into mobile projects by powering it up using humble AA batteries

Your Raspberry Pi's mobility is usually restricted by the length of the power lead. Rather than limiting it to your desk or living room, however, you can use it for mobile projects as diverse as launching it into near-Earth orbit or monitoring and automating your garden.

Of course, to do this you will need batteries, but adding battery power to your Raspberry Pi is simpler than you might have imagined. All that is required are six rechargeable AA batteries (or single-charge alkaline), a battery box with space for the batteries and a UBEC. The latter is a Universal Battery Elimination Circuit, a voltage regulator that will regulate the power supply and prevent damage to the Raspberry Pi, and can be bought for under £10.



“You can use your device for mobile projects as diverse as launching it into near-Earth orbit”

Make your order

01 If you're buying your components online, you should be able to get them all within five days. However, if you're ordering offline (specifically the UBEC), you should avoid traditional electronics stores and instead visit a model enthusiast store, as these circuits are regularly used in RC devices.

Check your UBEC

02 Two types of UBEC are available to choose from. If you used the store that we suggest in the resources box to the left, you'll receive one with a micro USB power connector for easy connection to your Raspberry Pi. However, if you bought one from eBay then there is a strong chance that you will receive one with a 3-pin connector.



Move connector pins

03 In order to use the UBEC with a 3-pin connector, you'll need to alter the position of the pins so that they occupy the two outer slots. Just use a small jeweller's screwdriver to lever up the small plastic catch and remove the red wire from the central slot, before sliding into the unoccupied outer slot.

Connect to battery box

04 With five batteries in the battery box, connect it to the UBEC (red-to-red, black-to-black) by twisting the wires, soldering or employing a 3-amp terminal strip, cut down to two pairs. It can be cut to size using a modelling knife.

Add a battery to boot

05 With your Pi ready to use and your Wi-Fi dongle plugged in, connect the UBEC to the micro USB port and insert the sixth battery into the battery box. The Pi's power and status lights should indicate that the computer is booting up, which gives you a fully portable computer.

Connect the 3-pin UBEC

06 If you purchased the UBEC with the now-modified 3-pin connector, you'll need to connect this to the Raspberry Pi's GPIO header. Connect the positive +5V (red) connector to Pin 2 and the negative 0V connector to Pin 6.

Measure uptime

07 You should have already set up your Pi for SSH use, so connect to the device via Putty after giving it time to boot fully (at least 60 seconds). In the terminal, enter:

```
sudo dd bs=32m if=/Users/rachelcrabb/Desktop/ArchLinux/archlinux-hf-2013-02-11.img of=/dev/disk1
```

This command will display the system uptime and also keep the Wi-Fi connection active.

Judge your uptime results

08 Uptime results depend upon the type of battery you use and the Raspberry Pi model. Single-charge batteries will last a little bit longer, but this is a more expensive option. Meanwhile, newer models have greater power requirements but run for less time. For more power, add more batteries!

Power extreme!

09 More batteries added in parallel should result in almost double the uptime (at least 16 hours on a 256MB Raspberry Pi Model A), but instead of alkaline or rechargeable batteries you might consider a modern lithium-based AA cell, which will last considerably longer than alkaline batteries.

What you'll need...

Bare Conductive paint
(pen or tub)

Male to female jumper wires

An assortment of LEDs,
switches and resistors
(optional)

Draw circuits with paint

Assembling circuits has never been so easy with the joys of conductive paint, enabling you to bring together art and electronics in a whole new way

Playing with electronics and physical computing is a very rewarding task. For a beginner though, the mess of wires and components can become very confusing quite quickly and things like soldering can be a safety concern when children are involved. Bare Conductive has taken the joy of electronics and made it far safer, easier and more versatile with their conductive paint. You can literally draw wires on paper with a paintbrush, use it for cold-soldering or a conductive adhesive and much, much more. There are not a great deal of boundaries to what you can do. Pair this paint with a microcontroller board and you could be creating interactive art, clothing and projects in no time.

Get your tools

01 Paint and a paintbrush aren't the first items that come to mind when you think about electronics, so you may be wondering where to get them from. Bare Conductive stock the paint and a selection of components in their shop (bareconductive.com/shop) but you will need to go somewhere else for art supplies. We would recommend trying a high street craft shop such as Hobbycraft (hobbycraft.co.uk) or a local independent.



BARE
CONDUCTIVE

ELECTRIC PAINT

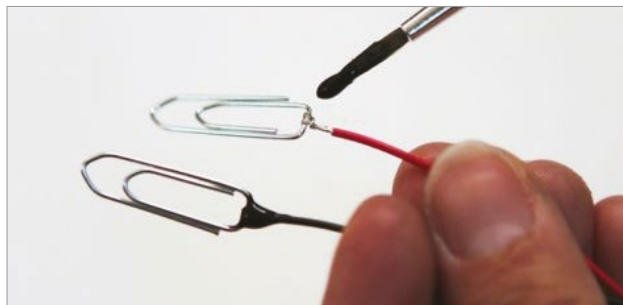


Pick your platform

02 The great thing about Bare Conductive paint is that, when dry, it works just like normal wiring! That means you can use it with any of your favourite microcontrollers like the Bare Conductive Touch Board, a Raspberry Pi or Adafruit's wearable FLORA platform. Or you can just use some small pin batteries and flashing LEDs for a standalone system.

Start to paint

03 You can paint Bare Conductive paint onto pretty much any surface – paper, fabric, walls, clothing, wood, plastic and much more. For really accurate shapes and results, the best idea is to create or purchase a stencil (paper stencils are easiest to make at home but use vinyl for the best edge finish).



Connect it up

04 There are plenty of ways to connect to the conductive paint (from battery packs for example) no matter what surface it's on, because once it is dry it acts just like an uninsulated wire. Therefore you can use wires glued on with the paint, paper clips, bulldog clips, alligator clips or even sewn-in conductive snaps for wearable projects.

Make repairs

05 The conductive paint is thick and when it's dry it becomes quite strong. These means you can use it to cold solder things together and repair any breakages. In other words, you could glue components into a circuit board or glue wires together and they would still function electrically.

“The mess of wires and components can become very confusing”

Clean up

06 A lot of you are probably thinking that something as cool as conductive paint is going to be nasty stuff. Actually Bare Conductive paint is non-toxic, water-based and water-soluble, and can therefore be cleaned easily with soap and water.

Make it waterproof

07 This paint only comes in black and is not waterproof. However, the great thing is that you can use it underneath or alongside any regular paints, varnishes and waterproofing sprays in order to act as insulation – or just to add some colour into your designs.



Touch and sound

08 Bare Conductive paint can also be used as a capacitive surface, meaning you can use it for touch, gesture or proximity controls when it is paired with a suitable control board. Bare Conductive make their own called the Touch Board which has everything you need to start experimenting with touch and sound. It can even act as a MIDI controller, an interface or an instrument.

Send an SMS from your Raspberry Pi

What you'll need...

Raspberry Pi
Twilio account

Create a program that combines Twilio and simple Python code to enable you to send an SMS from your Pi to a mobile phone

Text messaging, or SMS (Short Message Service), has become a staple of everyday communication. What began life as a 40 pence message service is now offered by most tariff providers as an unlimited service used worldwide. Twilio, a cloud communications company, enables you to send SMS messages for free from your Raspberry Pi to a mobile phone using just six lines of code.

“Create other communication programs, such as making phone calls, recording a call, and retrieving data”



Set up your Twilio account

01 The first step of this project is to register for a Twilio account and Twilio number. This is free and will enable you to send an SMS to a registered, verified phone. Once signed up, you will receive a verification code via SMS to the registered phone. When prompted, enter this onto the Twilio site to authenticate your account and phone. Go to [twilio.com/try-twilio](https://www.twilio.com/try-twilio) and create your account now.

Register numbers

02 Your Twilio account is just a trial account unless you pay the upgrade fee, which means you can only send and receive communications from a validated phone number. Enter the phone number of the contact who you want to verify, ensuring that you select the correct country code. Twilio will text you a verification code and you will need to enter it into the website form and press submit.

The dashboard

03 Once registered and logged in on Twilio, visit the dashboard page, which will display your AccountSid and your Auth Token. These are both required to use the Twilio REST. Keep these secure and private, but be sure to make a note of them as you will need them for your Python program later.



Install the software

04 Now boot up your Raspberry Pi and connect it to the internet. Before you install the Twilio software, it is worth updating and upgrading your Pi. In the LX Terminal, type `sudo apt-get update`, then `sudo apt-get upgrade`. Once complete, type `sudo easy_install twilio` or `sudo pip install twilio` to install the software. (If you need to install pip, type `sudo apt-get install python-pip python-dev`, press Enter, then type `sudo pip install -U pip`.)

Twilio authentication

05 Now you are ready to create the SMS program that will send the text message to your mobile phone. Open your Python editor and import the Twilio REST libraries (line one, below). Next, add your AccountSid and Auth Token, replacing the X with yours, as you will find on your dashboard:

```
from twilio.rest import
TwilioRestClient
account_sid = "XXXXXXXXXXXXX
XXXXXXXXXXXXX"
# Enter
```

```
Yours
auth_token =
"XXXXXXXXXXXXXXXXXXXXX"
# Enter
```

```
Yours
client =
TwilioRestClient(account_sid,
auth_token)
```

Create your message

06 You will probably want to be able to change your text messages rather than send the same one. Create a new variable in your program called message. This will prompt you to enter the phrase that you want to send to the mobile phone. When the program runs, this is

the message that will be sent:

```
message = raw_input("Please
enter your message")
```

Add your numbers

07 To send the message, you need to add the code line below and your two phone numbers. The first number is your mobile phone number, which is registered and validated with Twilio (Step 2). The second number is your Twilio account number, which can be retrieved from your dashboard page under 'Call the Sandbox number'. Change the Sandbox number to your country location and remember to add the international country code.

```
message =
client.messages.
create(to="+44YOURMOBNUMBER",
from_="+44YOURTWILIONUMBER",
body=message)
```

Send the message

08 Now send your message. The code below is not required, but is useful to indicate your message has been sent. Add the lines and save your program. Ensure your Raspberry Pi is connected to the internet and that your mobile is on, then run your program. You have just texted from your Raspberry Pi.

```
print message.sid
print "Your message is being
sent"
print "Check your phone!"
```

Other API and codes

09 You can also create other communication programs, such as making phone calls, recording a call, and retrieving data including caller IDs and call duration. The API here also complements a wide range of programming languages, including Ruby, PHP, Java and Node.js ([twilio.com/api](https://www.twilio.com/api)) to name a few.

What you'll need...

Internet connectivity

Web browser

Google Coder

googlecreativelab.github.io/coder/raspberry/sonicpi/teaching.html

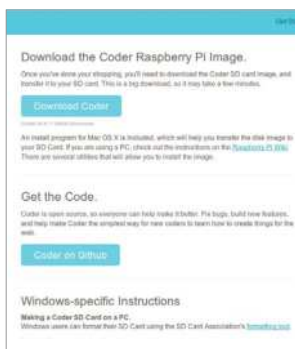


Build your first web server

Use Google Coder to turn your Raspberry Pi into a tiny, low-powered web server and web host!

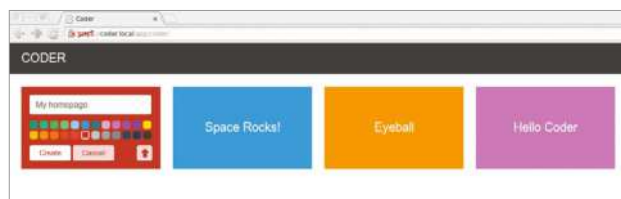
We're teaching you how to code in many different ways on the Raspberry Pi in this book, so it only seems fitting that we look at web development too.

There's a new way to use the web on the Raspberry Pi as well: internet giant Google has recently released Coder specifically for the tiny computer. It's a Raspberry Pi OS-based image that turns your Pi into a web server and web development kit. Accessible easily over a local network and with support for jQuery out of the box, it's an easy and great way to further your web development skills.



Plug in your Pi

02 For this tutorial, you'll only need to connect a network cable into the Pi. Pop in your newly written SD card, plug in the power and wait a few moments. If you've got a display plugged in anyway, you'll notice a Raspberry Pi OS startup sequence leading to the command-line login screen.



Get Google Coder

01 Head to the Google Coder website, and download the compressed version of the image. Unpack it wherever you wish, and install it using dd, like any other Raspberry Pi image:

```
$ dd if=[path to]/rasi.img of=/dev/[path to SD card] bs=1M
```

Connect to Coder

03 Open up the browser on your main system, and go to <http://coder.local>. You may have to manually accept the licence. It will ask you to set up your password, and then you'll be in and ready to code.

Language of the web

04 Now it's time to create your own app or website. Click on the '+' box next to the examples, give your app a name and then click Create. You'll be taken to the HTML section of the app. Change the Hello World lines to:

```
<h1>This is a HTML header</h1>
```

```
<p>This is a new block of default text</p>
```



Styled to impress

05 Click on the CSS tab. This changes the look and style of the webpage without having to make the changes each time in the main code. You can change the background colour and font with:

```
body {
    background-color:
#000000;
    color: #ffffff;
}
```

Querying your Java

06 The third tab allows you to edit the jQuery, making the site more interactive. We can make it create a message on click with:

```
$(document).click(function()
{
    alert('You clicked
the website!');
});
```

Full code listing

Some simple HTML code that can point us to some important websites. The h2 tag is used to display the time thanks to Java

HTML

```
<h1>Welcome to the internet...</h1>
<h2></h2>
<p><a href="http://www.linuxuser.co.uk">Linux User & Developer</p>
<p><a href="http://www.reddit.com/">Reddit</p>
<p><a href="http://www.linuxfoundation.org/">The Linux Foundation</p>
<p><a href="http://www.fsf.org/">Free Software Foundation</p>
```

Java

We're calling the current time using jQuery in the JS tab so that we can ultimately display it on the webpage

We're going to display the time as a 12-hour clock in the first if statement, and use AM and PM to differentiate the time

We make the minutes readable by adding a 0 if it's below 10, then concatenate all the variables and assign to the tag h2

```
var d = new Date;
var hours = d.getHours();
var mins = d.getMinutes();
if (hours > 12) {
    var hour = (hours - 12);
    var ampm = "PM";
} else {
    var hour = hours;
    var ampm = "AM";
}
if (hours == 12) {
    var ampm = "PM";
}
if (mins > 9){
    var min = mins;
} else {
    var min = "0" + mins;
}
var time = "The time is " + hour +
":" + min + " " + ampm;
$("#h2").html(time);
```

"Coder is a Raspberry Pi OS-based image that turns your Raspberry Pi into a web server and web development kit. It's an easy and great way to further your skills"

What you'll need...

Latest Raspberry Pi OS image

USB printer

USB wireless card

Print wirelessly with your Pi

Breathe new life into an old printer by using your Raspberry Pi as a wireless print server

Wireless printing has made it possible to print to devices stored in cupboards, sheds and remote rooms. You don't have to own a shiny new printer for this to work; old printers without native wireless support don't have to end up in the bin, thanks to the Raspberry Pi.

The setup is simple. With your Pi set up with a wireless USB dongle, you connect your printer to a spare USB port on the computer. With Samba and CUPS (Common Unix Printing System) installed on the Raspberry Pi, all that is left to do is connect to the wireless printer from your desktop computer, install the appropriate driver and start printing.

CUPS gives the Raspberry Pi a browser-based admin screen that can be viewed from any device on your network, enabling complete control over your wireless network printer.

Below Setting your Raspberry Pi to print wirelessly is a great way to get rid of annoying cables at your workstation



Check your printer works

01 Before starting, check that the printer you're planning to use for the project still works and has enough ink. The easiest way to do this is to check the documentation (online if you can't find the manual) and run a test print.



Detect your printer

02 With your Raspberry Pi set up as usual and the printer connected to a spare USB port, enter:

```
l sudo
```

This will confirm that the printer has been detected by your Raspberry Pi. In most cases you should see the manufacturer and model displayed.

Set up print admin

04 Set up the CUPS print admin tool. Boot into the GUI (startx) and launch the browser, entering 127.0.0.1:631. Switch to Administration, before ensuring that the 'Share printers' and 'Allow remote administration' boxes are selected. Select Add Printer and proceed to enter your Raspberry Pi OS username and password.

Configure Samba for network printing

06 Using a Windows computer for printing? Samba will need some configuration. Open /etc/samba/smb.conf in nano, search (Ctrl+W) for '[printers]' and find 'guest ok' which you should change as follows:

```
l guest ok = yes
```

Next, search for "[print\$]". Then change the path as follows:

```
l path = /usr/share/cups/drivers
```

Install Samba and CUPS

03 Install Samba on your Pi to enable file and print sharing across the entire network:

```
l sudo apt-get install samba
```

Next, install CUPS:

```
l sudo apt-get install cups
```

With a print server created, begin configuration by adding default user 'pi' to the printer admin group:

```
l sudo usermod -a -G lpadmin pi
```

Add your printer

05 A list of printers will be displayed, so select yours to proceed to the next screen where you can confirm the details, add a name and check the Share This Printer box. Click Continue to load the list of printer drivers and select the appropriate one from the list.

Join a Windows workgroup

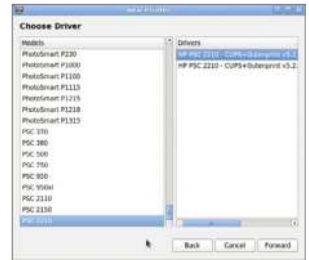
07 With these additions made, search for "workgroup" in the configuration file and then add your workgroup:

```
l workgroup = your_
workgroup_name
```

```
l wins support = yes
```

Make sure you uncomment the second setting so that the print server can be seen from Windows. Next, save your changes and then restart Samba:

```
l sudo /etc/init.d/samba
restart
```



Accessing your printer

08 Meanwhile, it's a lot easier to access your wireless printer from a Linux, Mac OS X or other Unix-like system, thanks to CUPS. All you need to do is add a network printer in the usual way and the device will be displayed.

Add AirPrint compatibility

09 It's also possible to print wirelessly from your Apple iPad using Apple's AirPrint system. To do this, you need to add the Avahi Discover software:

```
l sudo apt-get install avahi-
discover
```

Your wireless printer will now be discoverable from your iPad or iPhone and will be ready to print.

What you'll need...

Latest Raspberry Pi OS Image
www.raspberrypi.org/downloads

Breadboard

Connectors

Jumper wire

DSLR camera

Compatible shutter cable

Time-lapse camera trigger

Make shooting time-lapse video with your DSLR camera a cinch with our expert advice

You'd be forgiven for thinking that creating mesmerising time-lapse videos like those of Vincent Laforet (www.laforetvisuals.com) or John Eklund (www.theartoftimelapse.com) might be out of reach of the Average Joe. With the help of the Raspberry Pi and a sprinkling of Python code, though, that's no longer the case. In this guide we're going to trigger our DSLR camera to create pixel-perfect time-lapse imagery...

Set up the Raspberry Pi

01 For this tutorial we're assuming you're using a recent build of Raspberry Pi OS. With the Raspberry Pi set up with a keyboard, mouse and monitor, open the terminal and type:

```
sudo apt-get update
```

Install the RPi.GPIO library

02 Next we want to make sure your development environment is set up. Follow these steps to make sure you're all set. In the terminal, type:

```
sudo apt-get install python-dev
sudo apt-get install python-rpi.gpio
```

Set up the Pi Cobbler

03 For this tutorial we've used a cheap prototyping breadboard and an Adafruit Pi Cobbler to give us easy access to the Raspberry Pi's GPIO pins. As you can see from the picture, the Cobbler straddles the centre-point of the breadboard and a ribbon cable connects the two.

Full code listing

```
import RPi.GPIO as GPIO
import time

print '\nWelcome to the Complete Manual Time-lapse Tool.'
print "Just tell us how many shots you'd like to take and ↵
the interval between them.\n"
print "Try googling 'time-lapse interval calc' if you need ↵
help deciding.\n"

def main():
    shots = raw_input('How many shots would you like to ↵
take?\n ->')
    interval = raw_input('How frequently do you want to ↵
take them (in seconds)?\n ->')

    if shots.isdigit() and interval.isdigit():
        shots = int(shots)
        interval = int(interval)

    print "You'll be shooting for %d minutes.\n" % ↵
(shots * interval / 60)
    answer = raw_input('Are you ready to proceed?(yes/ ↵
no):')
    confirm = answer.lower() in ['yes', 'y']

    if confirm:
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(16, GPIO.OUT)
        taken = 1
        print
        print 'Starting a run of %d shots' % (shots)
```

Manual focus

We won't be controlling the autofocus with our Python app, so set the focus to manual and select your camera settings in advance of the shoot

2.5mm to 3.5mm

We're using a cheap Canon EOS DSLR, so to trigger the shutter with the Raspberry Pi, all we need is a simple 2.5mm to 3.5mm cable

Pi Cobbler

We're using the Pi Cobbler as a breakout for the Pi's GPIO pins, making the build process easier (though it's not required)

```
for i in range(0, shots):
    print
    print 'Shot %d of %d' % (taken, shots)
    taken +=1
    GPIO.output(16, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(16, GPIO.LOW)
    time.sleep(interval)
    GPIO.cleanup()
else:
    print "Let's try again (or press Ctrl + C to ↵
quit):\n"
    main()
else:
    print "Oops - You can only enter numbers. Let's try ↵
again:\n"
    main()

print
print 'Thanks for using the Complete Manual Time- ↵
lapse Tool!\n'
again = raw_input('Would you like to do another time- ↵
lapse? (yes/no):\n -> ')
proceed = again.lower() in ['yes', 'y']

if proceed:
    main()
else:
    print '\nSee you next time!\n'
    quit()

if __name__ == '__main__':
    main()
```

Configure the breadboard

04 For the Raspberry Pi's GPIO to control the camera, we need to create a circuit between a pin on the GPIO (in this case pin 23 on the Cobbler – but it's actually physical pin 16) and the pin that connects to the 'head' or 'tip' of the camera cable that activates the shutter when connected. The base of the connector cable is always ground, so make sure you ground the 'GND' pin on the Cobbler and the middle pin on the audio jack. With the circuit complete, we can focus on the code.

The Time-lapse Photography Tool

05 We've created a small 55-line Python utility called The Linux User Time-lapse Photography Tool, which asks the user to input how many shots they'd like to take and the frequency they'd like them taken. It then takes that information and uses it in a For loop to activate the shutter using GPIO pin 16. If you'd like to use the project 'in the field' we'd recommend using the Android app ConnectBot to SSH into your Raspberry Pi for input and feedback. Don't forget to start your script with `sudo python time_lapse_camera.py`

Creating a video

06 With your camera packed with images, we need to collect and output them as a video file. While it's possible on the Pi, copy them to an easily accessible folder on a separate Linux PC to make it much faster. We're going to use Ffmpeg. With the terminal open in the folder where your images are stored, type: `ffmpeg -f image2 -i image%04d.jpg -vcodec libx264 -b 800k video.avi`. This assumes you have libx264 installed on your machine and the 'image%04d.jpg' assumes the file format and the number of digits it's dealing with (in this case: 'picture0001.jpg').

What you'll need...

Github repository: <https://github.com/alexellis/motephatalexa>

Pimoroni mote-phat and accessories (pimoroni.com)

Soldering iron, flux and solder



Control lighting with the Pi and Amazon Echo

Control Pimoroni's home lighting kit for the Raspberry Pi through the Alexa Skills kit

Natural Language Parsing (NLP)

The Alexa service sends recordings from your device to the Amazon cloud where NLP (Natural Language Parsing) breaks the words down into intents and slots. A sample phrase such as "Alexa, what is the weather for London tomorrow?" would be parsed as an intent of "FindWeather" with two slots of: Date=tomorrow and Location=London, UK. Sample utterances help cover the many different ways we can say the same thing in a language.

In 2016, Amazon brought the Echo and Echo Dot devices to the UK market, providing a voice assistant for your home. The Alexa service provides lots of built-in features such as radio and music streaming, creating and editing your shopping lists, weather updates and many other custom skills provided by third parties. Skill builders such as Uber and Capital One publish their skills for the general public, which means they go through a vetting process similar to that of the Apple App Store. For this tutorial, we will be creating an unpublished skill for our own Echo or Echo Dot to control the Pimoroni home lighting kit called Mote.

Our skill will provide utterances for changing the colour of our LEDs like "Alexa, ask Mote change to blue" and will let us turn them off by saying "Alexa, tell mote turn off."

Prepare your hardware

01 Prepare your mote-phat add-on board by attaching and soldering its 40-pin female header, which will be included in the packaging. If you're using a Pi Zero you will also need to solder a 40-pin male header before continuing.

Set up the base system

02 Flash a new SD card with Raspberry Pi OS, making sure to create a file in the boot partition called 'ssh'; this will let us connect over SSH remotely and copy/paste commands without needing UI packages or a screen. Once plugged in, your Pi will be accessible via `ssh pi@raspberrypi.local`.

Responding to Alexa – Lambda or HTTPS

03 Alexa can either invoke code over a HTTPS endpoint (web-service) or via a Lambda function, which is a piece of code uploaded to Amazon's AWS service and invoked on demand. For our tutorial, we'll set up our own HTTPS endpoint from our Pi to the public internet with the ngrok tool. Download and unzip ngrok for Linux ARM into /usr/bin from <https://ngrok.com/download>.

Install Docker

04 Docker is a packaging and runtime system that allows us to build, ship and run software easily. Run these two commands then reconnect over SSH.

```
'''
# curl -sSL get.docker.com | sh
# sudo usermod pi -aG docker
'''
```

Now clone the Github repository and build the Docker image (this will take some time):

```
'''
# apt-get update && apt-get -qy install git jq
# git clone https://github.com/alexellis/motephath-alexa
# docker build -t alexamote .
'''
```

The resulting Docker image contains everything needed for our application in an isolated package.

"Our skill will change the colour of the LEDs"

This repository: Search

Pull requests Issues Gist

alexellis / motephath-alexa

Unwatch 2 Star 4 Fork 0

<> Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master motephath-alexa / Dockerfile Find file Copy path

alexellis Merge removal of endpoint b3214b2 13 days ago

1 contributor

21 lines (17 sloc) 495 Bytes Raw Blame History

```
1 FROM resin/rpi-raspbian
2 MAINTAINER alexellis2@gmail.com
3
4 RUN apt-get update \
5     && apt-get install git python-dev python-pip gcc \
6     && git clone https://github.com/pinoroni/mote-phat \
7     && pip install flask \
8     && cd mote-phat/library && python setup.py install \
9     && apt-get -qy remove python-dev gcc \
10    && rm -rf /var/lib/apt/lists/*
11
12 WORKDIR /root/
13 RUN mkdir /root/alexa/
14 WORKDIR /root/alexa/
```

“The flag -p tells Docker to expose the port for our web server code that talks to Alexa. The -d flag tells the service to run in the background”

Start the code with Docker

05 Our project's code is packaged with all its dependencies into a single container. We can now run that in the background and open the ngrok HTTPS tunnel to the internet. The flag -p tells Docker to expose the port for our web server code that talks to Alexa. The -d flag tells the service to run in the background.

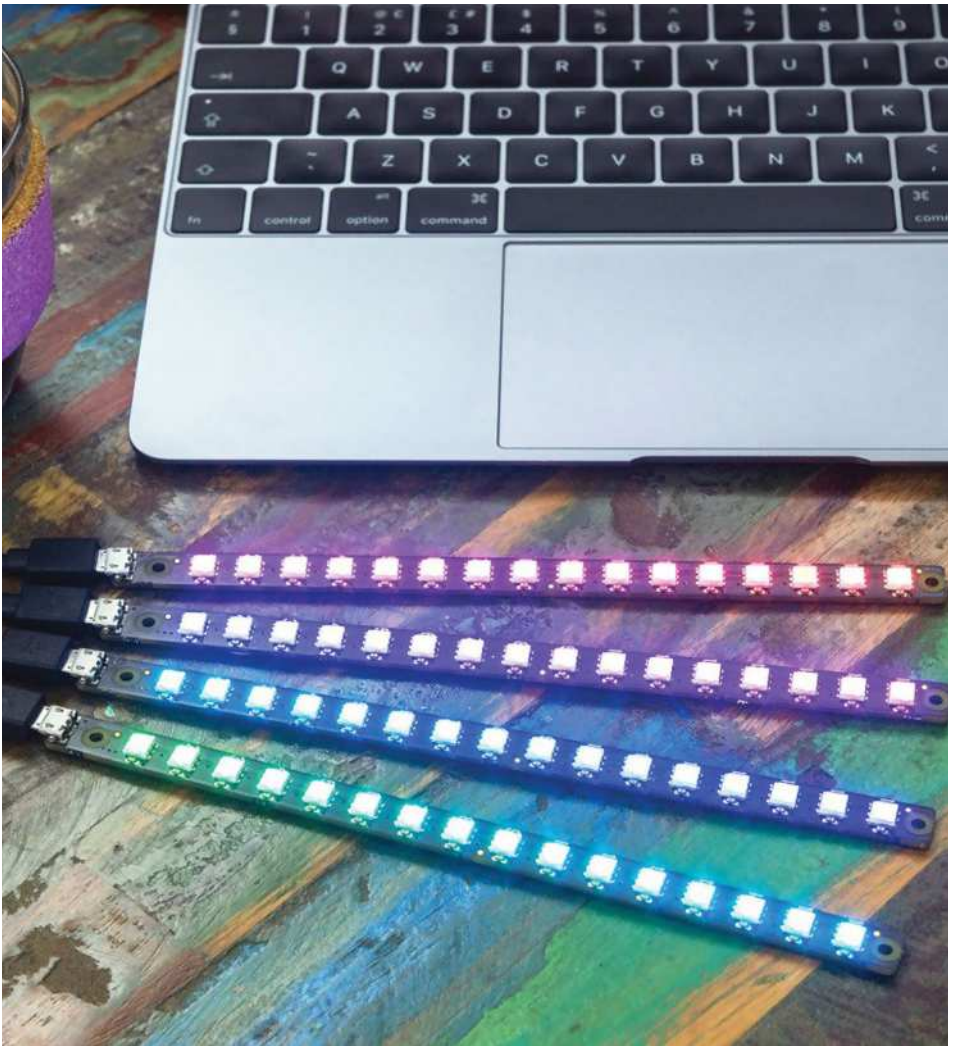
```
# docker run --name mote --privileged -d -p 5000:5000 alexamote
# ngrok http 5000 > /dev/null &
# curl localhost:4040/api/tunnels | jq -r ".tunnels[1].public_url "
```

Test the endpoint

06 Now you can test everything out by sending in a request just like the one the Alexa SDK creates. We have captured two samples and saved them in the Git repository. Test going red:

```
'''
# curl -X POST -H "Content-type: application/json" -d @coloursample.json https://c00738f6.ngrok.io
'''
Test turning the lights off:
'''
# curl -X POST -H "Content-type: application/json" -d @coloursample.json https://c00738f6.ngrok.io
'''
```

```
pi@mote: ~/dev/motephath-alexa $ docker run --name mote --privileged -d -p 5000:5000 alexamote
3c1325c7b0d39b0d537c37801ff74618ffa835644abd1b0f48df081e5bebc82c
pi@mote: ~/dev/motephath-alexa $ ngrok http 5000 > /dev/null &
[1] 2428
pi@mote: ~/dev/motephath-alexa $ curl localhost:4040/api/tunnels | jq -r ".tunnels[1].public_url "
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left   Speed
100  755  100  755    0     0  17730      0 --:--:-- --:--:-- --:--:-- 18414
pi@mote: ~/dev/motephath-alexa $ curl -X POST -H "Content-type: application/json" -d @coloursample.json https://9438e1a8.ngrok.io
{"version": "1.0", "response": {"outputSpeech": {"text": "OK, changing to red", "type": "PlainText"}, "shouldEndSession": true, "card": {"content": "OK, changing to red", "type": "Simple", "title": "Colour change"}}, "sessionAttributes": {}}
```



“Now you can test everything out by sending in a request just like the one the Alexa SDK creates. We have captured two samples and saved them in the Git repository”

The screenshot shows the Amazon Developer Console for a skill named 'mote' in the English (U.K.) locale. On the left, a sidebar contains links to Skill Information, Interaction Model, Configuration, SSL Certificate, Test, Publishing Information, and Privacy & Compliance, each with a status icon. The main area is titled 'Intent Schema' and contains a JSON editor for the schema. Below the editor is a section for 'Custom Slot Types' with a table for 'MoteColour' and a 'Sample Utterances' section.

Intent Schema

The schema of user intents in JSON format. For more information, see [Intent Schema](#). Also see [built-in slots](#) and [built-in intents](#).

```

1 {
2   "intents": {
3     {
4       "intent": "ChangeColourIntent",
5       "slots": {
6         {
7           "name": "Colour",
8           "type": "MoteColour"
9         }
10      }
11    }
12  }
13 }

```

Custom Slot Types (Optional)

Custom slot types to be referenced by the intent schema and sample utterances. For general information about custom slots, see [Custom Slot Types](#). Example: TOPPING <chocolate> onions | ham (note: newlines displayed as | for brevity)

Type	Values
MoteColour	red green blue warm white

Sample Utterances

These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. [Learn more](#). Up to 3 of these will be used as Example Phrases, which are hints to users.

```

1 ChangeColourIntent change to {Colour}
2 TurnOffIntent turn off

```

Above Test your codes on Amazon's developer website

Docker


Docker is a game-changer for packaging, deploying and running software. Each time you build software, Docker creates an image with its own root filesystem and network addresses. Inside a Docker container it feels exactly like a full virtual machine, except faster, because a container is regular process with some advanced syscalls applied for security and isolation. The Docker CLI is intuitive for Linux users with commands like 'docker ps', 'docker run', and 'docker kill'. The 'docker build' command uses a Dockerfile, which is similar to a Makefile.

Create an Alexa skill

07 Head over to <https://developer.amazon.com/myapps.html> and click Alexa>Alexa Skills Kit>Get Started. You may need to register for this step and provide billing information for any purchases you want to make.

Click Add a New Skill>English UK and type in 'mote' for the name and invocation name fields. For the intent schema, copy/paste 'speechAssets/intentSchema.json' and for sample utterances 'speechAssets/sampleUtterances.txt'. You must also add a custom slot called colour with the values red/green/blue on separate lines. The custom slot helps Alexa by providing a list of all the things you could say to her – it's like a parameter in coding.


"The custom slot helps Alexa by providing a list"

English (U.K.)  Add New Language

Global Fields

These fields apply to all languages supported by the skill.


Endpoint

Service Endpoint Type: ☐ AWS Lambda ARN (Amazon Resource Name)  ☒ HTTPS

Recommended

AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

[More info about AWS Lambda](#)
[How to integrate AWS Lambda with Alexa](#)

Pick a geographical region that is closest to your target customers: 

☐ North America ☒ Europe

Europe

Account Linking

Do you allow users to create an account or link to an existing account with you? ☐ Yes ☒ No

[Learn more](#)

Point Alexa to your HTTPS endpoint

08 Under the configuration tab of your Alexa Skill, click service endpoint type: HTTPS. Then select your nearest region (this will be Europe in the UK) and paste in the ngrok URL from earlier.

Now click 'My development endpoint has a certificate from a trusted certificate authority' on the SSL Certificate tab.

On the Test tab you can type in sample utterances such as "change to blue" or "turn off" – when you click 'Ask mote' a message will be transmitted to your Pi from Alexa's online service bypassing the Echo/Dot.

Point Alexa to your HTTPS endpoint

09 If everything worked you will be able to talk to your Echo/Dot. Simply say "Alexa, ask mote change to red" or "Alexa, ask mote to turn off". For dimming the brightness level you can take inspiration from Alex's Christmas Tree hack's source-code at: <http://blog.alexellis.io/christmas-iot-tree/>

```

10 @app.route('/', methods=['POST'])
11 def post Alexa_request():
12     post_data = request.json if data is not None else {}
13     response = None
14
15     if post_data["request"]["intent"]["name"] == "turnofflight":
16         response = get_response("Turning off", "OK")
17
18     elif:
19         response = get_response("OK setting desired colour", "OK")
20         slot_colour = post_data["request"]["intent"]["slots"]["Colour"]["value"]
21         if not slot_colour in ["red", "green", "blue"]:
22             response = get_response("Can only set red, green or blue", "error")
23
24     slot:
25         red = 0
26         green = 0
27         blue = 0
28
29         if slot_colour == "red":
30             red = 100
31         elif slot_colour == "green":
32             green = 100
33         elif slot_colour == "blue":
34             blue = 100
35         mote.set_colour(red, green, blue)
36
37     return Response(json.dumps(response), mimetype="application/json")

```

Next steps

10 Now that you have created your first skill, maybe you can think of some ways to extend it or to apply it to other hardware projects? We think dimming the light could be useful and it should be easy to add other colours. If you want to know more about Docker, check out the boxout, the Dockerfile on the articles GitHub and Alex's beginner tutorials at: <http://blog.alexellis.io/tag/raspberry-pi/>

What you'll need...

SenseHAT

Take a different reading

Humidity measures the amount of water vapour there is in the air. You can replace humidity with 'sense.temperature' to create a simple digital thermometer. Put the SenseHAT in the fridge or near a warm heat source to take various readings.

Create a real-time LED humidity display with Python

Use the Astro Pi (aka SenseHAT) to take humidity readings and immediately display the results

Humidity is a measure of how much moisture or water there is in the air. In this tutorial you will use Python to build a real-time humidity display. The program takes and stores a humidity reading using the SenseHAT's on-board sensor. Then it calculates a simple ratio to determine how many LEDs to turn on. In essence, the higher the humidity the more LEDs are turned on. Then the number of LEDs that need to be turned 'off' is calculated and these are added to the list. Finally, you'll set the program to display and 'turn on' the required number of LEDs.

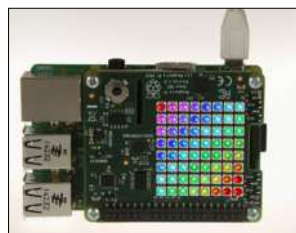
Attach the SenseHAT

01 Ensuring that your Raspberry Pi is off, take the SenseHAT and attach it to the GPIO pins. Slot it firmly into place with the SenseHAT covering the main body of the Pi. Add the power supply and boot up the Pi. Open the LXTerminal Window and type `sudo idle3` to open the Python 3 code editor. From the File menu, select New File to create a new program.

A test program

02 Next create a simple program to take a temperature reading and test that your SenseHAT is working correctly. The SenseHAT has a built-in heat sensor that can be used to read and return the current temperature. The sensor is close to the CPU and will pick up some of the residual heat. However, on the whole the reading is sound. Import the SenseHAT module and set the sense variable (lines 1 and 2). Take a temperature reading and store it in a variable called temp (line 3). Finally, print out the current reading (line 4). Save your program and run it; by pressing F5 on the keyboard.

```
from sense_hat import
SenseHat
sense = SenseHat()
temp = sense.get_
temperature()
print("Temperature: %s C" %
temp)
```



Create the live humidity reader

03 Now that you have tested the SenseHAT, you can begin to create the main program. First, import the SenseHat library (line 1). Then use the clear code to turn off any LEDs that have previously been on (line 3). This ensures that the LED matrix is reset each time the program starts. Create two variables, one for the colour of the LEDs when they are on and another for when they are off. In this example the 'on' colour is set to red but you can change the values to change the colours to your own preference.

```
1 from sense_hat import
SenseHat
2 sense = SenseHat()
3 sense.clear()
4 on_pix = [255,0,0]
5 off_pix = [0,0,0]
```

Take a humidity reading

04 Now it's time to take a humidity reading and round it to one decimal place. Begin by creating a while True loop (line 1), which means the program will continually take a reading and display the results. This ensures that the LED display is always updated and the readings are 'live'. On line 2 create a variable to store the humidity reading. The reading will be very accurate and contain several values after the decimal point. These are not required, so round up the reading to one decimal place (line 3).

```
6 while True:
7     hum = sense.humidity
8     hum = round(hum,1)
```

```
9 from sense_hat import SenseHat
10 sense = SenseHat()
11 sense.clear()
12 on_pix = [255,0,0]
13 off_pix = [0,0,0]
14 while True:
15     hum = sense.humidity
16     hum = round(hum,1)
```

What about larger readings?

05 Depending on the time of year or where you are located, the humidity reading may be very high. However, above a certain value the readings become insignificant. Set a conditional to check if the reading is greater than 100. If it is then change the reading value to 100 (line one and two). This means that the top humidity reading is set to a maximum of 100 and makes the calculation of which LEDs to turn on easier. On the next line create a list called leds which will store the number of LEDs to turn on or off. The SenseHAT has 64 LEDs so calculate the value of humidity which each LED represents (line four). (Each LED represents a value of 0.64 humidity.) Note that the first two lines are indented in line with the previous step.

```
17 if hum > 100:
18     hum = 100
19 leds = []
20 ratio = 64 / 100.0
```

LEDs On or Off

06 You now have the components to work out the number of LEDs to turn on to represent the humidity. The formula `ratio*hum` takes the humidity reading and multiplies it by the ratio from Step 5. For example, if the humidity reading is 50, then 50 multiplied by the ratio equals 32, half the LEDs. Convert this to an integer using `int` and store the value in a variable called `on_count` (line 1). To calculate the number of 'off' LEDs, subtract the number of 'on' LEDs from 64, since there are 64 LEDs in total (line 2).

```
21 on_count = int(ratio*hum)
22 off_count = 64 - on_count
```

LEDs On or Off

07 Now you have the total number of LEDs that need turning on or off. Write these values back to the list that you created in (Step 5), combining the colour and the total number of LEDs. First, take the colour of the 'on' pixels set in Step 1 and multiply this by the number of LEDs to turn on. Then use the extend list function to write these values into the list created in Step 2. For example, if the humidity is 50, then the total 'on LEDs' will be 32 and this step will add 32 red-coloured LEDs to the led list.

```
23 leds.extend([on_pix]*on_count)
```

Add the 'off' pixels to the list and turn on the LEDs

08 In Step 6 you calculated the number of LEDs that needed to be turned off using the code: `off_count = 64 - on_count`. Use this value and multiply it by the colour that you set for the LEDs when they are off, in Step 1. (In this tutorial, (0, 0, 0).) Use the code, 'extend' to create a list which holds the 64 LEDs and record how many of these are 'on' and how many are 'off'. Now read the list and plot it onto the SenseHAT LED matrix using the code: `sense.set_pixels(leds)` line 2.

```
24 leds.extend([off_pix]*off_count)
25 sense.set_pixels(leds)
```

Run the program

09 That completes the program. Press F5 to save and run the program. Watch the LED display, which will display the current humidity as a ratio of the 64 LEDs. Try huffing onto the humidity sensor to see what happens. The number of LEDs displayed should increase. Challenge family and friends to see who can light up the most LEDs.

What you'll need...

Raspberry Pi:

Tor:
<https://www.torproject.org/>

Turn your Pi into a Tor proxy with Onion Pi

Turn your Pi into a Tor proxy with Onion Pi

Turn your Raspberry Pi into a wireless access point to access the anonymous Tor network

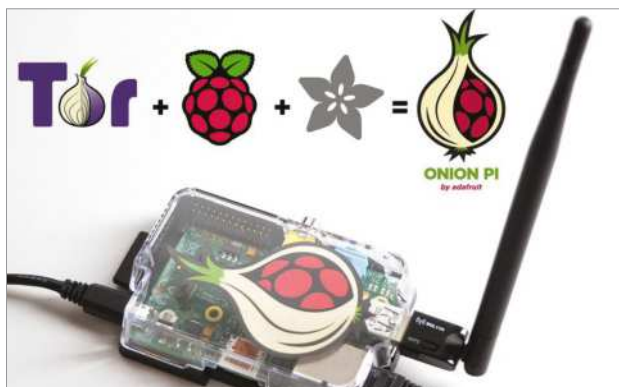
In this age of ubiquitous surveillance it's harder than ever to stop hackers, advertisers and shadowy government organisations from snooping on your browsing habits.

However if you choose to connect through Tor, your connection is encrypted and passed through a number of proxies through a process known as 'onion routing'. While this does slow down your connection, it also increases your privacy, making it extremely difficult to trace your actual current location.

Follow the steps in this tutorial to turn your Pi into a wireless AP (Access Point) named Onion_Pi. Any devices connecting to Onion_Pi will do so over the Tor network. When you're done, use a service like www.whatismyip.com to see that your location has changed. For more information about Tor visit <https://www.torproject.org/about/overview.html.en>.

Did you know...

The Tor network is a group of servers or 'relays' operated by volunteers. When you start tor on the Pi, it will build a circuit of encrypted connections through relays on the network. Each relay only knows the last relay a data packet came from and where it's going, meaning it's extremely difficult to trace the data back to you. Tor also changes the circuits it uses every few minutes to make it even harder to find your machine.



Turn your Pi into a Tor proxy with Onion Pi

Projects

Connect to Pi and check wireless is detected

01 Attach your Pi to your router via the Ethernet cable, then either open Terminal on the Pi or connect to it via SSH. Run the command `sudo ifconfig -a`. You should see the text `wlan0` which shows that the wireless module is up and running.

Install essential software

02 Run the command `sudo apt-get update` then `sudo apt-get install hostapd isc-dhcp-server tor iptables-persistent` to install the software. When you install `iptables-persistent` you'll be asked if you want to save the rules for your current configuration. Select 'Yes' both times.

Configure the DHCP server

03 Run `sudo nano /etc/dhcp/dhcpd.conf`. Find the two lines beginning 'option domain-name' and put a '#' at the start of each. Remove the '#' from the line '#authoritative'. Scroll to the end and type:

```
# subnet 192.168.42.0 netmask
255.255.255.0 {
#   range 192.168.42.10
192.168.42.50;
#   option broadcast-address
192.168.42.255;
#   option routers 192.168.42.1;
default-lease-time 600;
max-lease-time 7200;
#   option domain-name "local";
#   option domain-name-servers
8.8.8.8, 8.8.4.4;
}
```

Edit interface

04 Run `sudo nano /etc/default/isc-dhcp-server`. Scroll to the word `INTERFACES=""` and insert

'wlan0'. Press Ctrl + X, Y, return to save and close. Run the commands `sudo update-rc.d hostapd enable` and `sudo update-rc.d isc-dhcp-server enable` to make sure your changes start.

Set static IP

05 Run `sudo nano /etc/network/interfaces`. If you see the text 'auto wlan0' add a '#' at the start to comment it out. Find the line `allow-hotplug wlan0` and delete the two lines below it. Replace them with:

```
iface wlan0 inet static
    address 192.168.42.1
    netmask 255.255.255.0
Run sudo ifconfig wlan0 192.168.42.1
to set your IP.
```

Configure the Access Point

06 Run `sudo nano /etc/hostapd/hostapd.conf` to create a blank file. Paste in the following:

```
interface=wlan0
driver=nl80211
ssid=Onion_Pi
country_code=US
hw_mode=g
channel=6
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=Raspberry
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP
wpa_group_rekey=86400
ieee80211n=1
wme_enabled=1
```

Apply Access Point configuration

07 Run `sudo nano /etc/default/hostapd`. Find the line: and run:

```
#DAEMON_CONF=""
Edit it so it says:
```

```
#DAEMON_CONF="/etc/hostapd/
hostapd.conf"
Don't forget to remove the # in front
to activate it. Repeat these same
steps for hostapd with the command
sudo nano /etc/init.d/hostapd again
modifying the line #DAEMON_
CONF="" so that it reads DAEMON_
CONF="/etc/hostapd/hostapd.conf"
```

Configure Tor

08 Run `sudo nano /etc/tor/torrc` to configure Tor. Find `## https://www.torproject.org/docs/faq#torrc` and after it paste:

```
# Log notice file /var/log/tor/
notices.log
# VirtualAddrNetwork
10.192.0.0/10
# AutomapHostsSuffixes .onion,.
exit
# AutomapHostsOnResolve 1
# TransPort 9040
# TransListenAddress
192.168.42.1
# DNSPort 53
# DNSListenAddress 192.168.42.1
Save and exit. Run sudo update-rc.d
tor enable to make Tor start on boot.
```

Configure IP Tables

09 Run `sudo nano /etc/tor/torrc` to configure Tor. Find `## https://www.torproject.org/docs/faq#torrc` and after it paste:

```
# sudo iptables -t nat -A
PREROUTING -i wlan0 -p tcp --dport
22 -j REDIRECT --to-ports 22
# sudo iptables -t nat -A
PREROUTING -i wlan0 -p udp --dport
53 -j REDIRECT --to-ports 53
# sudo iptables -t nat -A
PREROUTING -i wlan0 -p tcp
--syn -j REDIRECT --to-ports
9040
Next, make your changes permanent:
# sudo sh -c "iptables-save >
/etc/iptables/rules.v4"
Reboot your Pi when done.
```

What you'll need...

Suitable for all models of Raspberry Pi

Make a doomsday switch

Keep your data safe with a handy 'nuke' password to erase your home folder in case of emergency

If you're worried about the somewhat Orwellian notion of forced disclosure of passwords, this project posits a rather radical solution to the dilemma by creating a second password for your user account, which, instead of logging you in, will nuke your home folder using special tools.

While this is simple to set up, do make sure to back up your personal data to a safe place before going ahead. Also bear in mind that anyone with physical access to your machine may seize it before you have a chance to flip this kill switch.

Hidden extras

In addition to the `srn` command in the `nuke` script, which securely erases the home folder, you may have noticed two other commands, `sfil` and `smem`. Remove the `#` at the start of these lines to overwrite the free space in the `/home` folder and clean any files in the Pi's virtual memory (RAM) respectively. This will take much longer than the `srn` command on its own but is a good deal safer.

```
File Edit Tabs Help
pi@raspberrypi ~$ sudo adduser alice
Adding user 'alice' ...
Adding new group 'alice' (1001) ...
Adding new user 'alice' (1001) with group 'alice' ...
Creating home directory '/home/alice' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
Full Name []: Alice Smith
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] Y
pi@raspberrypi ~$ sudo adduser alice sudo
Adding user 'alice' to group 'sudo' ...
Adding user alice to group sudo
Done.
pi@raspberrypi ~$
```

Create your new user account

01 Create a new Pi account for this project even if you already have one by opening Terminal on your Pi or connecting via SSH and running the command:

```
■ sudo adduser name
Add your new user as an Admin with:
■ sudo adduser name sudo
Substitute 'name' with your chosen username.
```

Make a doomsday switch

Projects

Create your Nuke script

02 You should stay logged into the 'Pi' user for now and run the following command:

```
sudo nano /etc/security/
security.sh
In the new window, paste
the following:
#!/bin/bash
read password
# If the username and
password match...
if [ "$PAM_USER" =
"name" ] && [ "$password" =
"nukepassword" ]
then
#Begin Nuke Process
echo "Nuke is starting."
#Securely erase the home
folder
srm -rvvv /home/name/
echo "Home folder has been
erased."
#Overwrite the /home folder
with random data
#sfill /home
echo "Home folder has been
overwritten"
#Clean RAM memory
#smem
echo "RAM is clean"
echo "User data has been
nuked."
fi
exit 0
```



Modify the script

03 In Line 5, substitute 'name' and 'nukepassword' for the username of your new account and the desired nuke password. Make sure this is different to your current one. Change 'srm -rvvv /home/name/' to the path of your real home folder.



Modify the script

04 Run nuke script on login. Make your nuke script executable with the command...

```
sudo chmod a+x security.sh
Next run...
sudo nano /etc/pam.d/common-
auth
...to open the Pluggable
Authentication Modules (PAM).
Find the line starting 'auth
[success=1...]' and change this to 'auth
[success=2...'. Immediately below this
line, paste the following:
```



Install secure delete tools

05 Run the command...
`sudo apt-get install secure-delete`
...to install the tools necessary to erase your home folder securely. Substitute 'name' with your chosen username.

Migrate your data (Optional)

06 If you previously had personal data in another user account, you should take this chance to move data across from that account to another from your backup drive. If you wish to delete the originals, do so using the new secure-delete tools, for instance:

```
'srm -r /home/bob/Pictures'
```



Test your new account

07 Reboot your Pi and log into your new user account using the normal login password. Check that your files are where you need them.



Test your nuke switch

08 If your data is backed up, there's no harm checking your nuke password works. Reboot the Pi once again. Select your new username and enter the nuke password. The system will hang while it removes your files.



Check nuke logs

09 You can still connect to the Pi via SSH while the nuke script is running. Use the command...

```
cat /tmp/pam.log
...to check the progress of the nuke.
Any further attempts to log in will just
take the user back to the login screen.
```



What you'll need...

Pimoroni Blinkt!

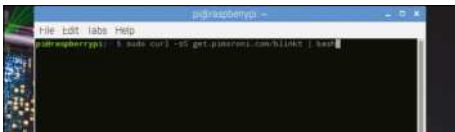
Who is using the Internet?

Do you live in a household or work in an office where several people use the Internet? Are there numerous devices always connected to the network that suck the bandwidth up and slow down the speed for all other users? Combine the Blinkt with Python and nmap to identify users, track them on the network and create a colourful LED visual for each user as they join and leave the network. Watch the video here: youtu.be/r_JDw65FTMA.

Get hands on with the Pimoroni Blinkt!

Learn the basics and create sparkling lights with your Blinkt!

Pimoroni has created an awesome 'super-bright RGB LED indicator' that is ideal for adding visual notifications to your Raspberry Pi without breaking the bank! The Blinkt! offers eight APA102 pixels in the smallest (and cheapest) form factor to plug straight into your Raspberry Pi 3/2/B+/A+/Zero. This tutorial walks you through how to install the Blinkt! and the required Python libraries and modules. Next, create and try out some simple code to control the lights and change the colours and the brightness of the LEDs. Finally, combine these skills together to code an LED colour generator that selects a random LED and a random RGB colour value for the range of eight lights. This creates a psychedelic disco-light setup.



Install the Blinkt!

01 The guys at Pimoroni have made it very easy to install the software for your Blinkt! (which also includes a wide selection of cool examples to show off its features). To get started, ensure that the power is off, attach the Blinkt! to the Raspberry Pi's GPIO pins and push down firmly. One side of it is straight and one is curved. The curved edges align with the curved corners of your Raspberry Pi. This ensures that you attach it the correct way round. Now attach your power supply and boot up your Pi. Open the Terminal and type:

```
curl -sS get.pimoroni.com/blinkt | bash
```

This will install all the required code.

Turn on an LED

02 To turn on an LED, open your Python editor and import the required Python module, line 1. Use the code `set_pixel()`, line 2, to set which pixels are turned on. The first number in the brackets corresponds to each LED. As with the common computing method of numbering, the first LED is referred to by the number zero, the second LED is number one and so on up to the eighth LED, number seven. Copy the code below and run the program to turn the first LED on.

```
from blinkt import set_pixel, show
set_pixel(0,255,0,0) show()
```



Change the colour of an LED

03 The next three values in the brackets (x, 255, 210, 150), are used to control the colour of the LED with the standard RGB colour palette. You can control and set the amount of Red, Green and Blue between the values of 0 and 255; the higher the number, the more of that colour is displayed on the LED. Combining these values gives you over 16 million possible combinations and colours. Change the values in your code, save and then run.

```
■ set_pixel(0,255,255,0) show()
```

Adjust the brightness of the LEDs

04 You may find that the LEDs are too bright to look at. It is not advisable to look at them for long periods as they may damage your eyes. Instead, turn down the brightness so that they can be viewed safely. First, import the brightness module, line 1. You can import multiple modules by including the 'module name' on the same line—for example, from blinkt import set_pixel, show, set_brightness. Now set the brightness to a suitable level, line 2. The values range between zero and one, where one is full brightness and zero renders all the LEDs off.

```
■ from blinkt import set_brightness
  set_brightness(0.5)
```

Turn on multiple lights

05 Turn multiple LEDs to on using the set_pixel code and then set the RGB colour values. Remember that for the LED number, 'zero' is the first LED, as you count from zero upwards. The last LED is number seven. Try turning on two or more LEDs; for example, the last LED set to red and the fourth LED set to blue. Save and run the program.

```
■ set_pixel(7,255,0,0) show()
■ set_pixel(3,0,0,255) show()
```



Create a random blinking light

06 Combine the previous code steps and create a random set of sparkling disco lights! Begin by selecting a new blank Python file and importing the random module, line 1. Finally, import the codes to clear the LEDs when the program exits, set a particular pixel to on and reduce the brightness, line 3. Line 4 enables the LEDs to clear on exit and line 5 sets your required brightness value.

```
■ import random
■ import time
■ from blinkt import set_clear_on_exit,
  set_pixel, show, set_brightness
■ set_clear_on_exit()
■ set_brightness(0.1)
```

Create a while loop and set random values

07 Now create a while loop to ensure that the program continually repeats and loops over the code, line 1. Use a for loop to iterate the code over each of the LEDs. This has the effect of applying the next instruction to each of the LEDs, working from the first LED to the last, line 2. Turn the LED on using the set_pixel code, line 3.

```
■ while True:
■     for i in range(8):
■         set_pixel(i,
```

Select a random colour

08 When coding the LED, state the RGB colour values of the LED as covered in step 3; use the code random.randint(0,255). Add two more incidences of the code to select values for the Green and Blue colours. Set the LEDs to display the colour using the code show(); note that this line doesn't have an indent. Finally, add a short time delay so that you can observe the lights before they change.

```
■         set_pixel(i, random.randint(0,255), random.
  randint(0,255), random.randint(0,255))
■     show()
■     time.sleep(0.05)
```

Run the program

09 Save your program file and run it by pressing F5; call the file a suitable name. Press Enter and the program will run, displaying a variation of random colours on each of the LEDs. Experiment with the colour values to create variations that you like and also to speed up or slow down the delay between changes.

What you'll need...

Raspberry Pi 3
8GB SD card
Xbox 360 controller
Oculus Rift Developer Kit
(optional)

Create a Pi-powered virtual reality setup

Combine the Raspberry Pi, Python-VRZero and 3D graphics module Pi3D to edit or make virtual reality environments

Virtual Reality is huge now and has come a long way since the concepts and CGI visuals of Stephen King's *Lawnmower Man*. It is one of the fastest growing areas of technology and you can now design models, explore new places and play games all within a virtual environment.

A professional VR hardware package is expensive and will set you back several hundred pounds. However, it's possible to emulate the VR setup up using a Raspberry Pi, Python-VRZero and a 3D graphics module, Pi3D. Now, this is purely for fun and learning, so don't expect huge gaming PC frame rates, although some of the demos

do peak at around 25-30 FPS on a Raspberry Pi 3. This tutorial shows you how you create a VR setup using the Raspberry Pi 3, a Xbox 360 controller and some Python code. Our first steps will walk you through how to install the required software and modules. We'll then cover configuring the hardware and drivers to enable you to control movement within the VR environment. The final steps look at the program code structure, where you can develop your own versions of the VR demo or design and build your own virtual worlds.





Python-VRZero

01 Using Python-VRZero is a frictionless way to get started creating your own virtual reality worlds in Python on a Raspberry Pi and combine an Oculus Rift. The program adds a number of features on top of Pi3D and solves the headaches of configuring a Pi 3 for VR development in Python. It includes default input event handling for keyboard, mouse, controller and the Rift for moving and altering the view of the player. It also supports Xbox 360 controller configuration and uses OpenHMD to read the rotational sensor reading from the Rift.

Fresh SD card install

02 Before you get started, it is recommended that you use a new SD card with a fresh installed image of the Raspberry Pi's official operating system, Raspberry Pi OS. You can download the operating system directly from the Raspberry Pi website at <https://www.raspberrypi.org/downloads>. Install using your normal preferred method. This project setup was tested using the Pixel OS image.

Update the Pi

03 Boot up your Raspberry Pi. At this stage you do not need to connect the Xbox 360 controller or Oculus Rift. When loaded, open the LXTerminal and update and upgrade the OS typing the two lines below. This may take some time.

```
sudo apt-get update
sudo apt-get upgrade
```

Install the Xbox 360 controller drivers

04 Now install the package dependencies for the Xbox 360 controller. You can keep the library up

to date by adding the code --upgrade to the end of the first line. Then install Pi3D software which will render the images. Type the two lines below as shown and press Enter.

```
sudo apt-get install -y
libhidapi-libusb0 xboxdrv
sudo pip3 install pi3d==2.14
```

Install the VR software – part 1

05 The Python-VRzero is available from the GitHub website and is easy downloaded using the git clone command, line one. Type this into your LXTerminal. Then move to the python-vrzero folder (line two) and install the program (line three). This deploys the software to interact between the hardware, 3D software and VR environment.

```
sudo git clone https://
github.com/WayneKeenan/python-
vrzero
cd python-vrzero
sudo python3 setup.py
install
```

Install the VR software – part 2

06 Once the installation completes, select and install the OpenHMD (line one) which enables the data from the Oculus Rift sensors to be read and worked with. Type line one into the LXTerminal and press Enter, then line two to install the required module. Enter line three and press Enter to link together all the required libraries:

```
sudo dpkg -i install/
openhmd_0.0.1-1_armhf.deb
sudo apt-get install -f
sudo ldconfig
```

Try some other demos

You may be interested in trying out some other demonstrations which are available at https://github.com/pi3d/pi3d_demos. Perhaps try riding a Roller Coaster or driving a tank! This resource also provides a guide how to create your own models using the Pi3D python library and code.

Copy over the configuration files – part 1

07 To interact with the Oculus Rift's rotation sensor and the Xbox 360 controller, you'll need to copy over the configuration files. This enables you to orientate and look around the environments. As you turn your head to the left the VR environment will adjust as if you are looking to the left. Type both of the lines below into the LXTerminal and press Enter after each line:

```
■ sudo cp config/83-hmd.rules
/etc/udev/rules.d/
■ sudo cp config/xboxdrv.init /
etc/init.d/xboxdrv
```

```
pi@raspberrypi ~ % sudo cp config/83-hmd.rules
/etc/udev/rules.d/
pi@raspberrypi ~ % sudo cp config/xboxdrv.init
/etc/init.d/xboxdrv
pi@raspberrypi ~ %
```

Copy over the configuration files – part 2

08 The Xbox 360 controller setup requires an additional command line to copy the default configuration file to the folder which contains the Xbox Drivers. This file contains all the mapping for the buttons, paddles and joysticks. Type in the line as shown below and press Enter:

```
■ sudo cp config/xboxdrv.
defaults /etc/default/xboxdrv
```

Rift Development Kit 1

09 If you do not have an Oculus Rift kit you can still use a HDMI monitor to display the output. Move to Step 11. If you own or have access to the Oculus Rift Development

kit, you will need to copy over the configuration file into the config.txt file. This file contains the configuration settings for the operating system which are loaded when you Raspberry Pi boots up. Type the line below into the LXTerminal and press Enter.

```
■ sudo cp config/config_DK1.txt
/boot/config.txt
```

Rift Development Kit 2

10 If you have access the Oculus Rift development kit version 2, then you will again be required to copy over a configuration file. Except this time select the **config_DK2.txt** file and copy the contents to the boot/config file. In the LXTerminal type the line below and press enter. Ensure that you select the correct configuration for the kit version which you have.

```
■ sudo cp config/config_DK2.
txt /boot/config.txt
```

Complete the set up

11 Finally run two commands. The first command (line one) enables the root-less USB udev config setup which was set up earlier in Step 7. The second command (line two) disables BluetoothLE. This is required as it stops the OpenGL ES, from hanging. Ensure that each line is typed in as printed, and press Enter after each line to enable the command to run:

```
■ sudo udevadm control
--reload-rules
■ sudo systemctl disable
hciuart
```

```
pi@raspberrypi ~ % sudo udevadm control
--reload-rules
pi@raspberrypi ~ % sudo systemctl disable
hciuart
pi@raspberrypi ~ %
```

Restart and plug in

12 This completes the installation and project setup. From the LXTerminal, shutdown the Raspberry Pi (line one). Attach the Xbox 360 controller and if you have one, the Oculus Rift. Turn the Rift on first before starting the Pi to ensure that it registers the hardware when the Pi boots up:

```
■ sudo shutdown
```

Hardware controls and setup

13 Python-VRZero sets up sensible defaults for handling input events from attached devices. The keyboard controls movement and the default mappings are: WASD, SPACE for jump and ENTER for action. The mouse controls looking (and direction of travel when moving). Mouse button 1 is action and mouse button 2 is jump. An Xbox 360 controller controls movement and view using the left and right stick respectively. The 'A' button is 'action' and the 'B' button is jump. The OpenHMD library is used to read the HMD sensor data. VR Zero automatically rotates the Pi3D Steroscopic camera in response to HMD readings.

Running a demo

14 Now for the fun part, which is to run a demo program and immerse yourself in a VR world. There are several to choose from, and each one demonstrates the features of the Python-VRZero program. The demos need to be run using Python 3 and executed as a script from the demos folder. (If you are using a Oculus Rift you will need to navigate to the folder via the display inside the Rift headset.) Open the LX Terminal, and move to the **python-vrzero/demos** folder, line one. To list the available

Create a Pi-powered virtual reality setup

Projects

demo type ls, this will list the file names of all the demos. To run a demo type ./ followed by the name of the demo, for example to run the abbey demo type, ./abbey.py (line 2). You will be presented with a VR render of Buckfast Abbey, to end the environment simply press Escape on the keyboard.

```
cd python-vrzero/demos
./abbey.py
```

Editing the textures

15 If you have used Pi3D before you can access the program template to set up your own models. If not, then change the textures in the Shape demo program. First open the LXTerminal and type sudo idle 3 to load Python 3. Select File and open, navigate to the following folder /home/pi/python-vrzero/demos and select the shapes.py program. Locate line 14 which begins pating = pi3d (pictured, top of the page). This is the first line of code which tells the program which textures to load for each shape. There are several after which can also be edited.

Change a texture

16 Using the folder explorer, click the file icon to navigate to the textures in the texture folder /home/pi/python-vrzero/demos/textures. You will see the files that are used for the shapes demo. Replace the image file with one of your own and then on line 14 of the program change the file name to match your selected image file choice.

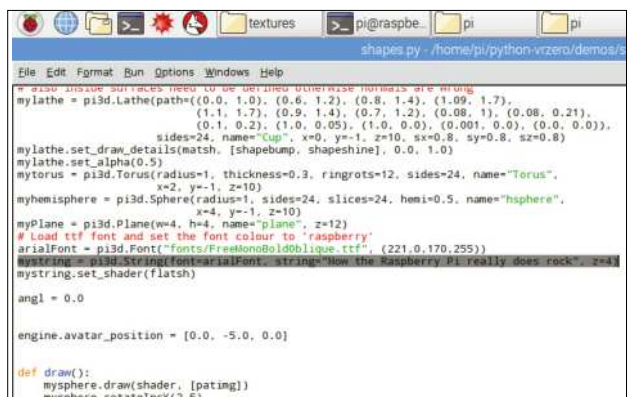
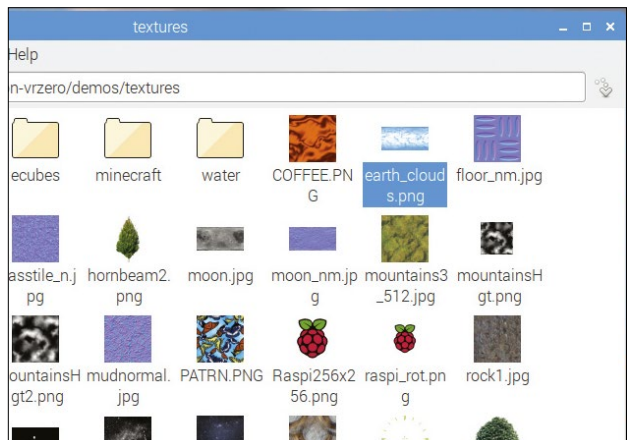
If you don't want to use your own texture file then you can change the file name to one of the other image files listed in the folder. Press F5 to save and run the demo. You will notice that the shape textures have changed.

Alter the message

17 Return to your Python editor and locate line 71. This holds the message which is displayed in the shapes VR demo. Change the text to a sentence of your choice. Save and run the program. Congratulations you have now begun to modify your own VR demos. Experiment with the program files for each of the demos editing the textures. For example, how about creating a church made out of chocolate? If you want to try other demos, go to https://github.com/pi3d/pi3d_demos.

Keep up to date

Python-VRzero is an ongoing development, and updates and improvement are always being added. Refer to the GitHub repository for the latest updates: <https://github.com/WayneKeenan/python-vrzero>



Projects

Build a networked Hi-Fi with a Pi Zero

What you'll need...

Github repository

<http://github.com/alexellis/pyPlaylist>

pimoroni pHAT DAC

Soldering iron, flux & solder

Build your own networked Hi-Fi with a Pi Zero

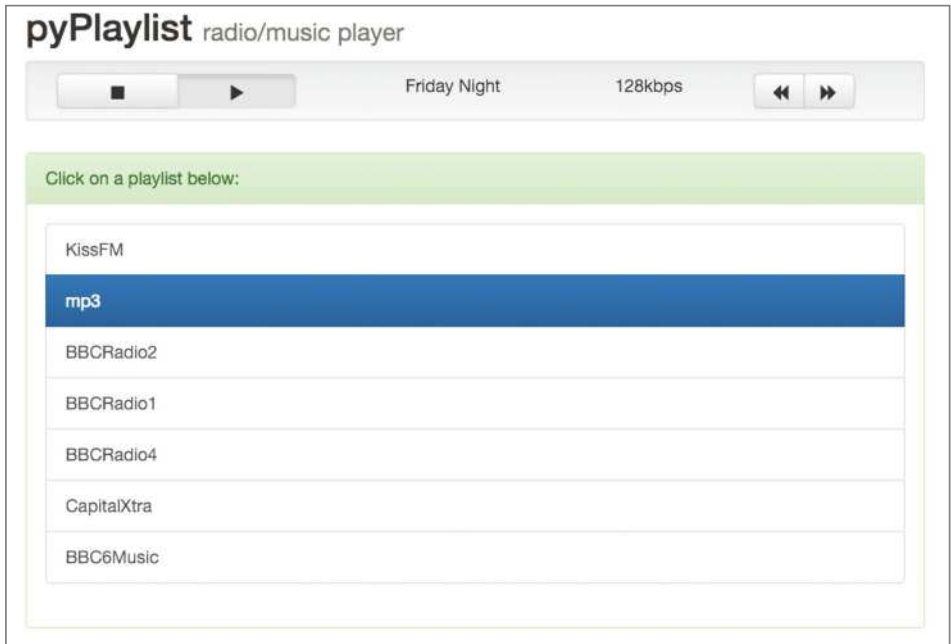
Put the Pimoroni pHAT DAC together with a Pi Zero to create a networked Hi-Fi

Take advantage of the UK's online radio stations, Linux's popular Music Player Daemon, and utilise a responsive web-server to control it all. The full-sized Raspberry Pis have two built-in audio outputs: audio over HDMI cable and a 3.5mm headphone jack that can suffer interference and noise. The Pi Zero itself has no audio jacks but Pimoroni has come to the rescue and built a high-quality DAC (digital audio converter) using the same chip as the Hi-Fi berry (PCM5102A).

Did you know...

We wrote pyPlaylist with the Python flask framework, which is an ideal starting point for simple RESTful websites. The front-end code saves the screen from completely reloading by using jQuery to update the song or radio information. Bootstrap has been employed to make the pages responsive (compatible with your PC, phone and tablet). The code has been released under GPL, so why not fork the code and tweak it to your own needs?





Solder the headers

01 The pHAT DAC comes with a 40-pin header, which you will need to solder. We consider a flux pen, work-lamp and thin gauge 60/40 solder essential for this. An optional RCA jack can also be bought to give a phono-lead output for older stereos.

Install drivers

02 The DAC relies on I2C, so we have to load some additional kernel modules. If you are running Raspberry Pi OS then you can type in the following for a one-script installation over secure HTTP:

```
curl -sS https://get.pimoroni.com/phatdac | bash
```

While HTTPS provides a secure download, curious types may want to review the script before running it.

Install Music Player Daemon (MPD)

03 Now go on to install the MPD package and enable it to start on boot. MPD will be the backbone of the project, providing playback of MP3s and internet radio stations. The MPC (client) software is also installed for debugging and setting up your initial playlists.

```
sudo apt-get install mpd mpc
sudo systemctl enable mpd
```

Clone and install pyPlaylist web-server

04 NpyPlaylist is a responsive (mobile-ready) web-server written with Python & Flask web framework. Once configured it will

give us a way of controlling our Hi-Fi through a web-browser. The following will install pyPlaylist on Raspberry Pi OS:

```
sudo pip install flask
python-mpd2
cd ~
git clone https://github.com/
```

“An optional RCA jack can also be bought to give a phono-lead output for older stereos”

Did you know...

In Raspbian Jessie the controversial systemd software was added, giving a highly modular way of managing start-up scripts among other things. While systemd configuration files are now considered best practice, they can take time to fully understand. For that reason we would suggest using cron to start the script on reboot as a temporary measure.

```
crontab -e
@reboot /usr/bin/python
/home/pi/pyPlaylist/
app.py
```

Choose the radio stations

05 We have put together a list of some popular radio stations in the UK which can be run into MPD with the `add_stations.sh` file. You can edit this file or find your own from this site: <http://radiofeeds.co.uk>.

```
cd ~/pyPlaylist
./add_stations.sh
```

of the playlists are available, as shown in the list below:

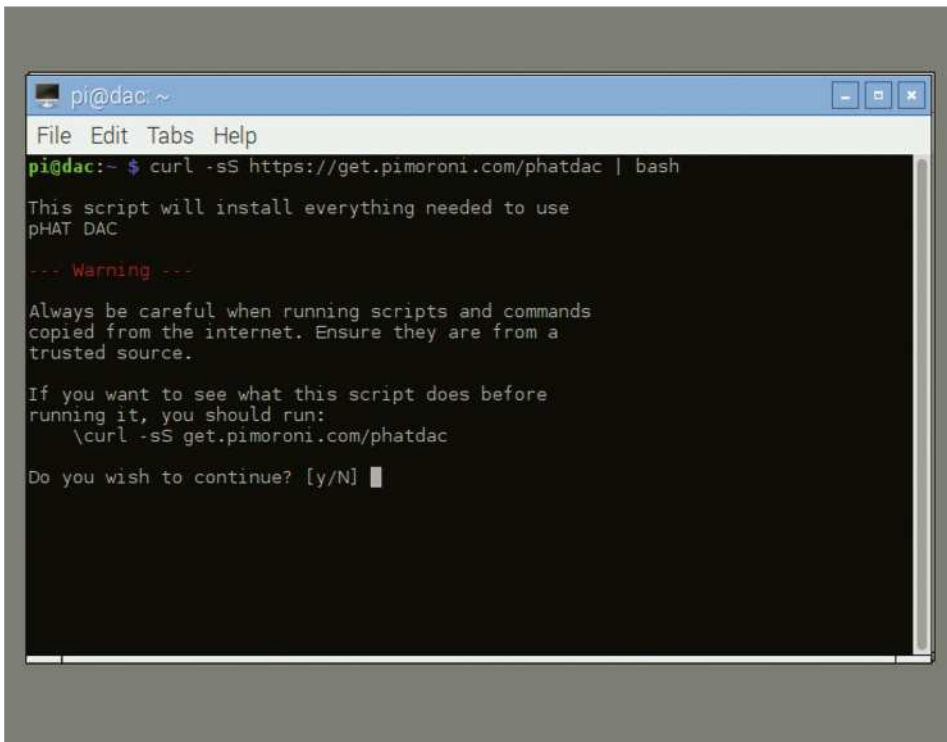
```
$ mpc ls
BBC6Music
BBCRadio1
BBCRadio2
BBCRadio4
CapitalXtra
KissFM
```

If you decide that you want to remove one of the stations then just type in the following:

```
mpc rm BBC6Music
```

Review the stations

06 Each of the radio station are added into their own playlists – the `mpc ls` command shows which



Start the web-server

07 Now that we have some stations, we can run the web-server from the pyPlaylist directory. Then open up a web browser to start playing a radio station. The following command reveals your IP address on Raspberry Pi OS:

```
$ ./Raspberry Pi OS_get_ip.sh
192.168.0.20
```

Once you know the IP address, connect to the URL in a web-browser on port

Add a custom music playlist

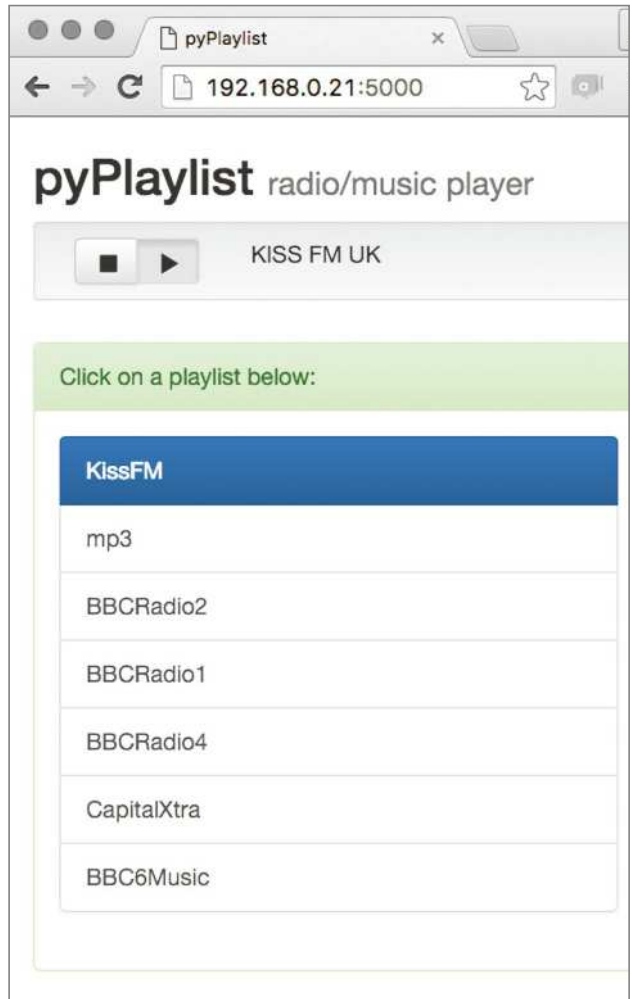
08 Now put together a sub-directory with your music files under `/var/lib/mpd/music/` and ensure that `mpd` has access to read it. Then we update `mpd`'s database, clear out the current playlist and add in all the tracks from the new directory (`ambient`), finally saving it as a new playlist.

```
mpc update
mpc clear
```

```
mpc ls ambient | mpc add
mpc save ambient
```

Finishing up

09 Now your music player is functioning, all that's left to do is to add some speakers, obviously! Almost anything with a RCA or 3.5mm input source will work for this purpose. That part we will leave up to you. To take a look at the code here in full, go to <http://bit.ly/290mailH>. Go ahead and enjoy the tunes!



“Now that we have some stations, we can run the web-server from the pyPlaylist directory. Then open up a web browser to start playing a radio station”

What you'll need...

Wi-Fi enabled Raspberry Pi
Parrot Drone 2.0 (or other)

Use Raspberry Pi to take control of your drone

Create code which enables you to customise and automate drone flights, retrieve flight data and respond to the drone tag



Drones are becoming ever more popular and mainstream. You may already be aware that Amazon is currently working on using drones to deliver packages directly to your house within hours of you placing an order, creating a new requirement for them to be autonomous and programmable.

This tutorial offers a little taster of a Python module which enables you to write and deploy programs to your drone. The API used is now a few years old (2014) and supports operation with Python 2.7. However, it offers a number of simple methods for beginners to create their own programs and spark your curiosity.

The tutorial begins with a quick walkthrough on installing the required software and libraries. Then jump straight in and create a simple but inspiring program which, when run, automatically launches the drone and then lands it. The tutorial covers the instructions to connect to the drone via the Raspberry Pi's Wi-Fi and then deploy your programs. You will then learn how to automate the drone and program a predetermined set of flight actions; for example, fly forward, turn left, turn right and then land. You can experiment with the various movements and create your own versions and flight plans. The final section of the tutorial introduces the use of the 'tag' symbol which can be recognised by the drone's front-facing camera. Once recognised, flight data is sent back to the drone and you can use these values to trigger an action or response such as flying forward towards the tag or landing the drone.

Update Pi and install drone API

01 This project uses a Python API which enables you to access and control the drone directly with Python code. To get started, update your Raspberry Pi using the standard update and upgrade commands (`sudo apt-get update`, `sudo apt-get upgrade`). Then, using the web browser on your Pi or other computer, head over to www.playsheep.de/drone/downloads.html, right-click on the PS-Drone API program and save it.

Test program – part 1

02 The PS-Drone API file requires no installation, but it must be placed into the same folder as the Python files that you create to control your drone. Open the LXTerminal and type `sudo idle` to open the Python 2 programming environment. Begin by creating a simple test program which will launch the drone and then land it. Start a new file and import the time and PS-Drone module.

```
import time
import ps_drone
```

Test Program part 2

03 The next step is to initialise the drone, setting up and opening a communication connection between it and the Raspberry Pi. Begin by initialising the API, then connect to the drone via the software, starting the subprocesses. Next, add the command to launch the drone; this uses the code `drone.takeoff()`.

```
drone = ps_drone.Drone()
drone.startup()
drone.takeoff()
```

Test program – part 3

04 The last stage is to use the time library to add a short delay between the drone taking off and then landing. This can be used to check the connections are working by setting the delay to two seconds, which is enough time to start the rotors and stop them without the drone leaving the ground. To land the drone, use `drone.land()`. Add the following two lines to your program and save it, ensuring that the file is in the same folder as the PS-Drone API file.

```
time.sleep(2)
drone.land()
```

Connect to the drone via Wi-Fi

05 Before the program will interact with the drone, you need to connect to it via the wireless network that it creates. Each drone has an on-board router which creates and broadcasts an open network. Power up the drone and wait for it to run the preflight checks. Success is usually indicated by four green lights. Then use the network finder on your Pi to search for the drone. Double-click to connect. Note that it requires no key or password to connect.

Run the program

06 To run the program, you'll need to execute it from the LXTerminal window. Open the Terminal and use `cd` to navigate to the folder where your test program is saved. To run the program, type `python name_of_the_file.py` and press Enter. (In this tutorial, the file is named `take_off_land`). Ensure you have enough space to safely launch the drone. If not, then set the time value to one.

```
sudo python name_of_the_file.py
```

Hints and tips

When connecting to the drone via the Raspberry Pi, there are possible errors. Many have been resolved in the PS-Drone API version two.

1) Sometimes, running a program after a previous program has finished can result in the error 'address in use'. This is because the drone/Pi has not disconnected from the network socket. Simply reset your Pi.

2) Always ensure that the drone has completed its startup checks and is online, indicated by four green LEDs, before you attempt to connect to it.

3) When using the 'tag', it is easier to recognise if the background is in contrast; for example, white paper against a white shirt does not aid identification. Check out the tag hack in action at <https://youtu.be/gXKdUhz7zAw>.

Automate the drone

07 Now that you have a working connection and program, adapt it to move or fly the drone. Start a new Python file and import the `ps_drone`. Then, as before, initialise the API and connect to the drone. Set the drone to take off, but this time set the time delay to 7.5 seconds. This provides sufficient time for the drone to take off, stabilise itself and wait for the next commands.

```
import time
import ps_drone
drone = ps_drone.Drone()
drone.startup()
drone.takeoff()
time.sleep(7.5)
```

Fly forward

08 Add the code to fly the drone forward; this is simply `drone.moveForward()`. Then add a pause which represents the duration that you want the drone to fly forward for. In this example, the drone flies forward for two seconds. Next, stop the drone. Like a car, this requires a stopping time, so add a short time delay; you can match the delay value used when flying forward. You can test this program before moving onto the next steps.

```
drone.moveForward()
time.sleep(2)
drone.stop()
time.sleep(2)
```

“Ensure your Pi is connected to the drone’s network”

Fly backwards

09 Once the drone has stopped, it will hover until it receives another command. On the next line down, add the code to fly the drone backwards. Again, you will need to add a short time delay, then stop the drone. Finally, set the code to land the drone.

```
drone.moveBackward()
time.sleep(1.5)
drone.stop()
time.sleep(2)
drone.land()
```

Run the program

10 Ensure that your Pi is connected to the drone’s network, as shown in Step 5. Remember that the program needs to be executed from the Terminal window. As before, open it and use `cd` to navigate to the folder where the program is saved. To run the program, type `sudo python name_of_the_file.py` and press Enter.

Turning left or right

11 Modify the same program to alter the flight direction of the drone. This uses the code lines, `drone.turnLeft()` and `drone.turnRight()`. After each call, remember to state the duration or time that the action runs for. For example, using `time.sleep(2)` will turn the drone left or right for two seconds. After the required time, stop the turning action using the code `drone.stop()`. This returns the drone to the hovering state. Add the relevant code to your program and then save it. Connect and run as previously demonstrated in Steps 5 and 6.

```
drone.turnLeft()
time.sleep(2)
drone.stop()
time.sleep(2)
```

Tag detection

12 The drone software has the capability to use the forward-facing camera to identify and read a ‘tag’. Once detected, this can then be used to trigger an event such as landing the drone or flying it towards the tag. Begin the program by starting a new Python file; import the `time` and `sys` modules. Next, import `ps_drone`. As before, add the startup and connect to the subprocesses.

```
import time, sys
import ps_drone
drone = ps_drone.Drone()
drone.startup()
```

Tag settings – part 1

13 There are four configuration settings to set. First, reset the drone (line one) so that the status is set to ‘good’, as indicated by four green LEDs. Use the code `drone.useDemoMode(True)` to use a dataset of 15 when transferring the data from the camera. Now set which packets will be decoded. Finally, set a small time delay to enable the drone to fully awake after the reset.

```
drone.reset()
drone.useDemoMode(True)
drone.
getNDpackage(["demo","vision_
detect"])
time.sleep(0.5)
```

Tag settings – part 2

14 In this second set of, start by enabling the universal detection by setting the detect type to a value of 3. This triggers the drone to look for the specific tag. Disable detection from the ground camera. Then set the drone

configuration count.

```

drone.
setConfig("detect:detect_type",
"3")
drone.
setConfig("detect:detections_
select_h", "128")
drone.
setConfig("detect:detections_
select_v", "0")
CDC = drone.ConfigDataCount
while CDC == drone.
ConfigDataCount:
time.sleep(0.01)

```

Takeoff and taking a reading

15 The drone is now configured to recognise and read the tag; add a small time delay, line one. This is useful if you need to move the drone outside before it launches. Add lines two and three to launch the drone; once deployed, it will continue to hover. Next, create a loop to continually check for the tag and take the readings. On line nine, tagNum returns the number of tags found; tagX returns the horizontal position of Drone in relation to the tag. The vertical position of the drone is collected with the code on line 11 and stored in a variable called tagY. The distance of the drone from the tag, tagZ, is on the penultimate line and orientation of the drone is stored in tagRot. Remember to include the indentations when adding the lines of code.

```

time.sleep(10)
###take off###
drone.takeoff()
time.sleep(7.5)
# Get detections
stop = False
while not stop:
NDC = drone.NavDataCount
while NDC == drone.
NavDataCount:

```

```

time.sleep(0.01)
if drone.getKey():
stop = True
# Loop ends when key was
pressed
tagNum = drone.
NavData["vision_detect"][0]
tagX = drone.
NavData["vision_detect"][2]
tagY = drone.
NavData["vision_detect"][3]
tagZ = drone.
NavData["vision_detect"][6]
tagRot = drone.
NavData["vision_detect"][7]

```

Responding to the data

16 Now set up a conditional, an if statement to display the data and take action. Line three prints out the collected data; note that it is converted into a string as the original data format is returned as a float, ie a decimal. Create a new variable called distance to store the physical measurement of the distance of the drone from the tag. This is stored as an integer, line four. Check if this distance is greater than 300, line six; if it is then trigger an event. For example, set the drone to fly towards the tag for two seconds, lines seven and eight.

```

if tagNum:
for i in range (0,tagNum):
print "Tag no "+str(i)+"

```

```

: X= "+str(tagX[i])+" Y=
"+str(tagY[i])+" Dist=
"+str(tagZ[i])+" Orientation=
"+str(tagRot[i])
distance = int(tagZ[i])
print (distance)
if distance > 300:
print ("Moving Forward")
drone.moveForward()
time.sleep(2)

```

Close enough

17 Once the drone is close enough to the tag, (in this program less than 300), then it stops flying forward, line one. Add a short time delay to allow it to stop, line two. Add the two other conditions; if the drone is already less than a distance of 300 from the tag then print 'close enough'. Finally, respond if the tag is not detected, lines five and six. Save your program code and connect to the drone. (Ensure that the indentation levels are correct; if not, this will cause errors when you run the code.) Execute the program as before, following the method described in Steps 5 and 6.

```

drone.stop()
#time.sleep(2)
else:
print ("close enough")
else:
print "No tag detected"
#drone.stop()

```

```

def takeoff():
time.sleep(7.5)
# Get detections
stop = False
while not stop:
NDC = drone.NavDataCount
while NDC == drone.
NavDataCount:
time.sleep(0.01)
if drone.getKey():
stop = True
tagNum = drone.
NavData["vision_detect"][0]
tagX = drone.
NavData["vision_detect"][2]
tagY = drone.
NavData["vision_detect"][3]
tagZ = drone.
NavData["vision_detect"][6]
tagRot = drone.
NavData["vision_detect"][7]
distance = int(tagZ)
print (distance)
if distance > 300:
print ("Moving Forward")
drone.moveForward()
time.sleep(2)
drone.stop()
else:
print ("close enough")
drone.stop()
time.sleep(0.01)

```

What you'll need...

A toy RC car with two channels (steering and drive)

Adafruit PWM I2C servo driver

Female-to-female jumper cables

5V battery power bank

Estimated cost: £60 / \$100

Components from
www.modmypi.com

Build a Raspberry Pi-powered car

Make use of cutting-edge web technologies to take control of a remote controlled car with a smartphone or tablet...





Did you know...

You can make this project work with just about any remote controlled car. Follow the guide for more details.

Web technologies are moving forward at a huge pace, cloud technologies are bringing mass computing to individuals, and hardware has reached a perfect moment in time where sensors, displays and wireless technology have all evolved into efficient and affordable devices. We truly are at a point where nearly anyone can take an idea from nothing to a working product in a week and at very little cost. Just like this project, which is fun, quick and easy to build on and a fantastic way to learn. We're going to grab an old remote-control car, rip off its radio receiver and replace it with the Raspberry Pi, hook it up on the network, fire up a bleeding-edge web server and then get your smartphone or tablet to control it by tilting the device. By the end of this, not only will you have a fun toy, you will have learnt about the basic technologies that are starting to power the world's newest and biggest economy for the foreseeable future.

Raspberry Pi-controlled car build process

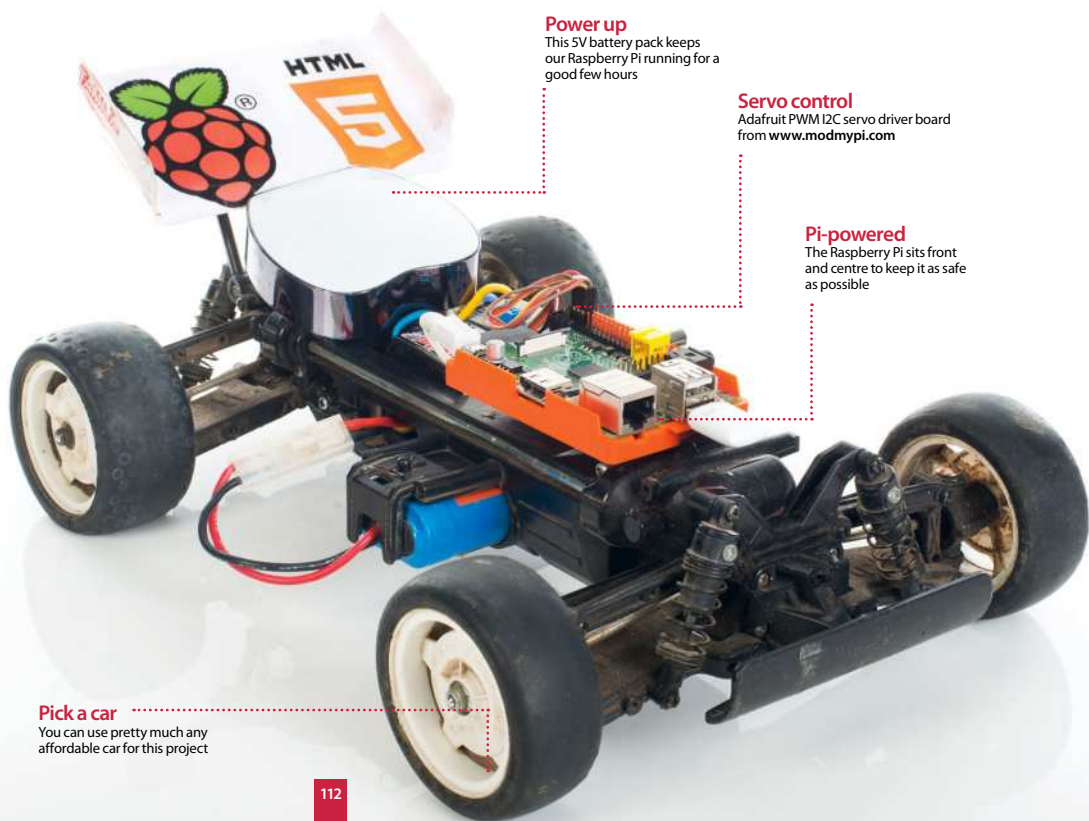
To help our toy car come to life using the latest web technologies and our credit card-sized computer, we're going to need to make some pretty significant changes to its workings. Fortunately, the most complex aspects of the build can be accomplished with a couple of affordable purchases, namely a servo controller board to take care of the steering and throttle, and a 5V battery pack to keep the Raspberry Pi running smoothly.

Identify and remove old radio

01 This project is effectively replacing the car's normal transmitter and receiver. Notice the three sockets on the original receiver: one goes to the motor controller and one to the steering servo. Some remote-control cars also have separate

battery for the electronics, but those (especially with an electronic speed controller with BEC) get their 5V power supply directly from the speed controller, saving on components. If you don't have a speed controller with 5V BEC, you'll need to get a 5V

supply elsewhere. Many shops sell 5V battery power supplies – often as mobile phone emergency top-ups. www.modmypi.com sells a suitable 5V battery power bank for under £20 and you should get a couple of hours of use from your Raspberry Pi.



Power up

This 5V battery pack keeps our Raspberry Pi running for a good few hours

Servo control

Adafruit PWM I2C servo driver board from www.modmypi.com

Pi-powered

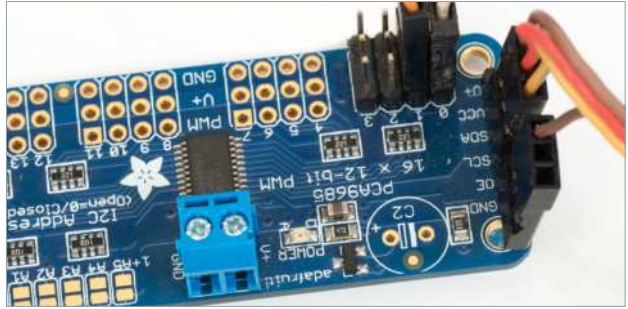
The Raspberry Pi sits front and centre to keep it as safe as possible

Pick a car

You can use pretty much any affordable car for this project

Attach the servo cables to the new controller

02 We soldered our 16-channel I2C servo controller board from www.modmypi.com as per its instructions and simply plugged channel 0 (steering) and channel 1 (motor) headers onto it. There are six cables in total: the bottom two are ground, the middle two are the power and the top two are the PWM (pulse-width modulation) signals. This is a good time to think of places to mount the extra components and the best fixing method seems to be sticky-back Velcro.



Connect the I2C bus to the Raspberry Pi



03 We're using the Raspberry Pi's I2C bus to control the servo interface board, which only needs four cables – they all go between the Raspberry Pi and the servo controller board as pictured. Visit <http://bit.ly/N3nq4J> for a tutorial on how to set up I2C on the Raspberry Pi.

From top to bottom we need to use the 1. GND, 2. SCL, 3. SDA and 4. VCC, which map directly to the same ports on the Raspberry Pi. Essentially this is power, ground and two communication channels.

Hook it up to the Raspberry Pi

04 On a Rev 1 Raspberry Pi, the cables look the same. Though the Rev boards have different labelling, the physical pins are in the same place. Bottom left (closest to the RasPi power connector) is the 3.3V power; next to that is the SDA header, which is the data channel. Next to that in the bottom right is the SCL channel, which controls the clock of the I2C devices. And finally – on the top-right port – is the Ground. We recommend printing a labelled image of the GPIO pins.

Overview of the main components

05 You should now have the servo board in the middle with the steering servo and speed controller on one side and the Raspberry Pi on the other. The motor is connected to the other end of the speed controller (that end should have much thicker wires); the speed controller also has two thick wires going to the main car's battery – in this case a 7.2V NiCad. We now have two very separate power systems with the high current motors on one side and the low current electronics on the other. Let's make sure it stays that way.

Find everything a home

06 We can now put it together. Use plenty of sticky-back Velcro, tie wraps or elastic bands to keep everything secure and find spaces in the car's body to hide the wires where possible. While it is possible to stick or screw the Raspberry Pi directly to the car, we recommend to use at least the bottom half of a case for added protection and ease of access. Insert your SD card, network cable or Wi-Fi dongle and power supply. Sit back and admire your hacking skills.



What you'll need...

A RasPi car, ready to go

An internet connection

A reasonably modern smartphone/tablet

Pi car source code
github.com/shaunuk/picar

Did you know...

Our code will send instructions to our car over twenty times per second. It will be very responsive to drive!

Control your Raspberry Pi-powered car

Control a toy car with a smartphone and the latest web technologies

Now that we have our fantastic Raspberry Pi-powered car all wired, charged and ready to go, it's time to make it come alive. We're using the best web technologies that the JavaScript programming language offers, to harness the natural movement of your hand and wirelessly drive the vehicle.

Each little movement of your hand will trigger an event that calculates what the car should do and then sends it over a socket connection. If all goes to plan you should have complete control over your vehicle.

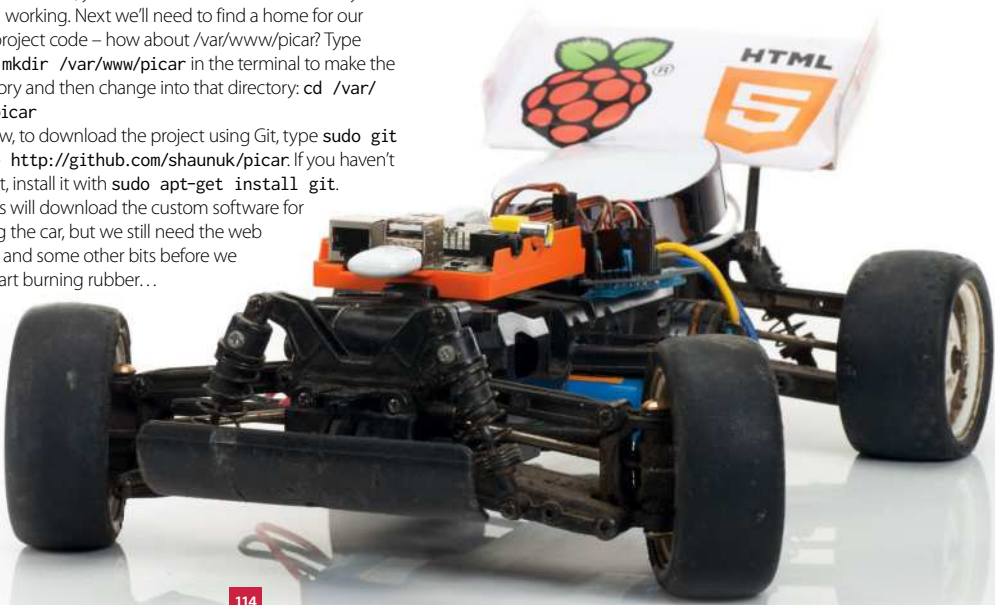
Download and install the software

Below All you need to finish off your project is access to a smartphone or tablet

01 First of all, you will need to the I2C connectivity working. Next we'll need to find a home for our new project code – how about `/var/www/picar`? Type `sudo mkdir /var/www/picar` in the terminal to make the directory and then change into that directory: `cd /var/www/picar`

Now, to download the project using Git, type `sudo git clone http://github.com/shaunuk/picar`. If you haven't got Git, install it with `sudo apt-get install git`.

This will download the custom software for driving the car, but we still need the web server and some other bits before we can start burning rubber...



Download and install Node.js

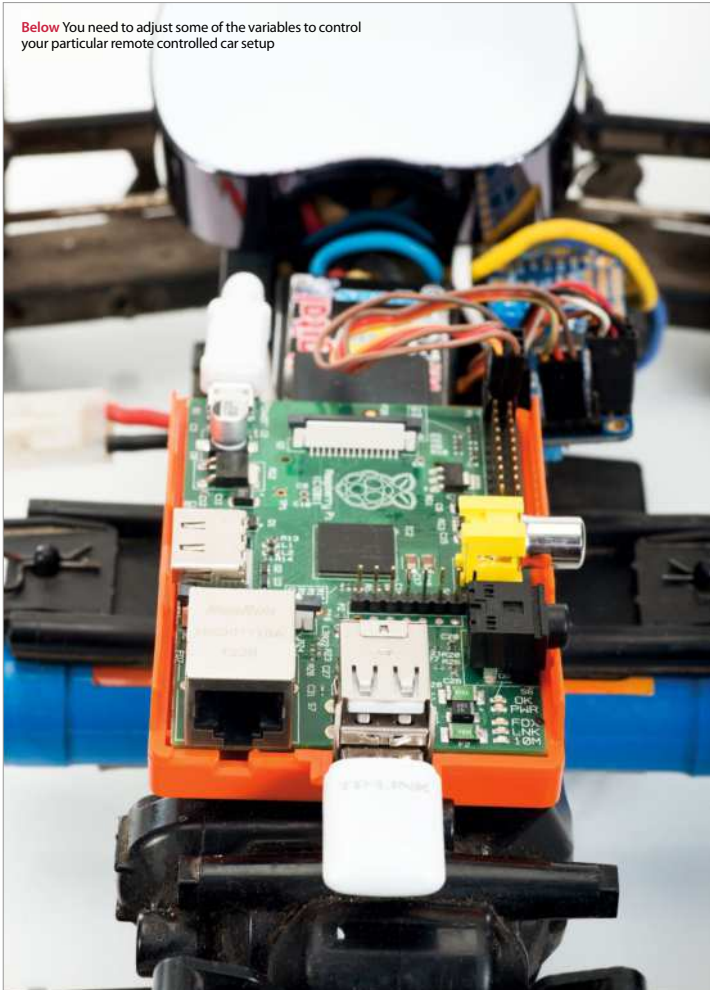
02 Next we're using Node.js and its package tool, the Node package manager (npm). Type `sudo wget http://nodejs.org/dist/v0.10.21/node-v0.10.21-linux-arm-pi.tar.gz`. This will download a fairly recent version of Node.js – the version Raspberry Pi OS has in its repositories is way too old and just doesn't work with the new technologies we're about to use. Extract the node package by typing:

```
■ sudo tar -xvzf node-v0.10.21-linux-arm-pi.tar.gz
```

Configure Node.js

03 To make it easy to run from everywhere, we will create symbolic links for Node and npm binaries. Type `sudo ln -s /var/www/node-v0.10.21-linux-arm-pi/bin/node /bin/node` and then `sudo ln -s /var/www/node-v0.10.21-linux-arm-pi/bin/npm /bin/npm`. Then, to get the extra modules, type `npm install socket.io node-static socket.io adafruit-i2c-pwm-driver sleep optimist`

Below You need to adjust some of the variables to control your particular remote controlled car setup



Get to know the project

04 Now we have everything, you should see three files: the server (app.js), the client (socket.html) and the jQuery JavaScript library for the client. The server not only drives the servos, but it is a web server and sends the socket.html file and jQuery to the browser when requested – it's a really neat and simple setup and just right for what we're trying to achieve.

Test the servos

05 Our handy little program (app.js) has a special mode just for testing. We use two keywords here: beta for servo 0 (steering) and gamma for servo 1 (motor control). Type `node app.js beta=300`. You should see the front wheels turn. Now the numbers need experimenting with. On our example, 340 was left, 400 was centre and 470 was right. Do the same for the motor by typing `node app.js gamma=400` and take note of the various limits of your car.



“We’re using the best web technologies that the JavaScript programming language has to offer”

Configure sensible defaults

06 Now you know what your car is capable of, we can set the defaults in `app.js` and `socket.html`. Edit `app.js` and find the section that says ‘function emergencyStop’. Adjust the two numbers to your car’s rest values. Then open `socket.html` and adjust the predefined values under ‘Define your variables here’.

Going for a spin

07 We’re almost ready to try it out, but you need to know the IP address of your Pi car, so type `ifconfig` at the terminal. Then fire up the app by typing `node app.js`. Now grab the nearest smartphone or tablet, making sure it’s on the same network as your Pi. Open the web browser and go to `http://[your IP address]:8080/socket.html`. You should get an alert message saying ‘ready’ and as soon as you hit OK, the gyro data from your phone will be sent to the car and you’re off.

Full code listing

socket.html

```
<html>
<head>
<script src="jquery-2.0.3.min.js"
language="javascript"></script>
<script src="/socket.io/socket.io.js"></
script>
<meta name="viewport" content="user-
scalable=no, initial-scale=1.0, maximum-
scale=1.0;" />
<script>
//----- Define your variables here
var socket = io.connect(window.location.
hostname+'8080');
var centerbeta = 400; //where's the middle?
var minbeta = '340'; //right limit
var maxbeta = '470'; //left limit
var multbeta = 3; //factor to multiply the
// raw gyro figure
var centergamma = 330;
```

```
var ajustmentgamma = 70; //what do we do to
the angle to get to 0?
var mingamma = 250; //backwards limit
var maxgamma = 400; //forward limit
var multgamma = 1; //factor to multiply the
//raw gyro figure by to get the desired
//rate of acceleration
window.lastbeta='0';
window.lastgamma='0';
$(function(){
  window.gyro = 'ready';
  alert('Ready -- Lets race !');
});
window.ondeviceorientation = function(event)
{
  beta = centerbeta+(Math.round(↵
event.beta*-1)*multbeta);
  if (beta >= maxbeta) {
    beta=maxbeta;
  }
  if (beta <= minbeta) {
    beta=minbeta;
```

```

    }
    gamma = event.gamma;
    gamma = ((Math.round(event.
gamma)+ajustmentgamma)* multgamma)+
centergamma;
//stop sending the same command more than
once
send = 'N';
if (window.lastbeta != beta) { send = 'Y' }
if (window.lastgamma != gamma) { send = 'Y'
}
window.lastbeta=beta;
window.lastgamma=gamma;
if (window.gyro == 'ready' && send=='Y') {
//don't send another command until ready...
    window.gyro = 'notready';
    socket.emit('fromclient', { beta: beta
gamma: gamma } );
    window.gyro = 'ready'; }}

```

app.js

```

//declare required modules
var app = require('http').
createServer(handler)
, io = require('socket.io').listen(app)
, fs = require('fs')
, static = require('node-static')
, sys = require('sys')
, PwmDriver = require('adafruit-i2c-pwm-
driver')
, sleep = require('sleep')
, argv = require('optimist').argv;
app.listen(8080);
//set the address and device name of the
breakout board
pwm = new PwmDriver(0x40,'dev/i2c-0');
//set pulse widths
setServoPulse = function(channel, pulse) {
    var pulseLength;
    pulseLength = 1000000;
    pulseLength /= 60;
    print("%d us per period" % pulseLength);
    pulseLength /= 4096;
    print("%d us per bit" % pulseLength);
    pulse *= 1000;
    pulse /= pulseLength;
    return pwm.setPWM(channel, 0, pulse);
};
//set pulse frequency
pwm.setPWMFreq(60);
//Make a web server on port 8080
var file = new(static.Server)();
function handler(request, response) {
    console.log('serving file',request.url)
    file.serve(request, response);
};

```

```

console.log('Pi Car we server listening
on port 8080 visit http://ipaddress:8080/
socket.html');

```

```

lastAction = "";
function emergencyStop(){
    //center front wheels
    pwm.setPWM(0, 0, 400);
    //stop motor
    pwm.setPWM(1, 0, 330);
    console.log("###EMERGENCY STOP - signal
lost or shutting down");
}

if (argv.beta) {
    console.log("\nPerforming one off servo
position move to: "+argv.beta);
    pwm.setPWM(0, 0, argv.beta);
    //using direct i2c pwm module
    pwm.stop();
    return process.exit();
}

if (argv.gamma) {
    console.log("\nPerforming one off servo
position move to: "+argv.gamma);
    pwm.setPWM(1, 0, argv.gamma); //using
direct i2c pwm module
    pwm.stop();
    return process.exit();
}

//fire up a web socket server
io.sockets.on('connection', function (socket)
{
    socket.on('fromclient', function (data) {
        console.log("Beta: "+data.beta+" Gamma:
"+data.gamma);
        //exec("echo 'sa "+data+"' > /dev/
// ttyAMA0", puts);
        //using http://electronics.chroma.se/rpisb.php
        //exec("picar.py 0 "+data.beta, puts);
        //using python adafruit module
        pwm.setPWM(0, 0, data.beta);
        //using direct i2c pwm module
        pwm.setPWM(1, 0, data.gamma);
        //using direct i2c pwm module
        clearInterval(lastAction);
        //stop emergency stop timer
        lastAction = setInterval(emergencySt
op,1000);
        //set emergency stop timer
    });
});
process.on('SIGINT', function() {
    emergencyStop();
    console.log("\nGracefully shutting down
from SIGINT (Ctrl-C)");
    pwm.stop();
    return process.exit();
});

```



Xbox Zero arcade

Let's make a self-contained arcade machine out of old bits of kit, a spare Xbox pad and a Pi Zero!

What you'll need...

- Raspberry Pi Zero
- Original Xbox controller
- Wire cutters
- Craft knife
- Isopropyl alcohol swabs
- Micro SD card
- BluTak
- Micro USB OTG cable
- Cross-head screwdriver
- Electrical tape
- 2A micro USB power supply
- Mini HDMI cable/adaptor

The Raspberry Pi Zero is tiny, ridiculously tiny. It's also small enough to be hidden in a variety of household objects in order to enhance their capabilities. Whatever you can find to fit it in, you can turn into some kind of smart machine.

Take old game controllers. If you're anything like us you've probably got a couple of boxes full of old computer equipment you just can't bear to throw away – an Atari Jaguar that hasn't been touched since the 90s, a Sega Dreamcast which you're sure you'll plug in again one day, an old Xbox that lies languishing since you picked up something bigger and better. Turns out it actually was useful to keep them around – it's time to bring these old systems back to life.

We're going to show you how to gut an old videogames controller, replace its innards with a Raspberry Pi Zero, and then load it up with a treasure trove of retro games. From start to finish, this project should take you under an hour to complete – and then you'll be able to load up the ROMs you legally own on your new console and enjoy them from the comfort of your sofa.

Gather your equipment

01 While the Zero doesn't take up much space, videogame controllers are often stuffed full of delicate electronics. The trick here is to find a games controller which has enough space inside for the Zero. We're going to be using the original Xbox controller, nicknamed The Duke. If you don't have one to hand, they can be picked up for a couple of quid from most second-hand electronics shops, and they're easily found online too.

If you can't find one, you can use newer USB game pads that are designed to look like controllers for classic systems like the SNES and Mega Drive. Make sure you choose a controller that has enough buttons for the games you want to play – some classic fighting games, for example, really can't be played on a two-button NES controller.

Warning!

02 Working with electrical items and sharp objects can be dangerous. You risk damaging yourself or, worse, breaking your toys. Please ensure everything is unplugged from electrical supplies before attempting this project. As with any electronics projects, you should also take care to fully ground yourself before playing around with sensitive components –the static electricity from your body can ruin them. Anti-static wrist straps or a few taps on a radiator should do the trick.

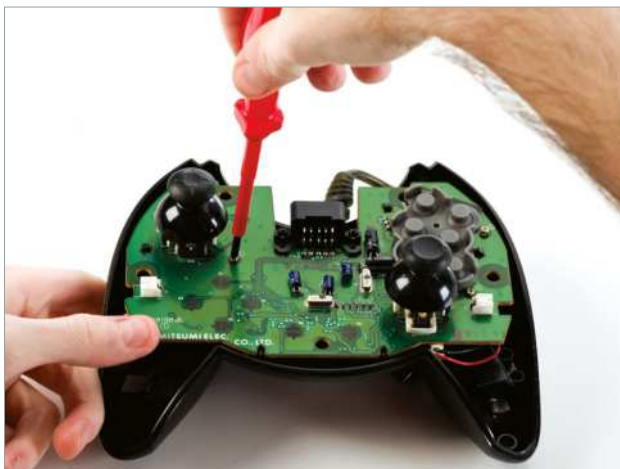


Above You can mod your controller with just a few simple tools

Lots of little bits

When taking apart electronics, keep a few small bowls or containers nearby. Put each type of screw in its own separate container so you don't accidentally mix them up. With the Xbox controllers, you'll find that the buttons especially have a habit of rolling away from you, so stash them somewhere safe as well. Keep track of any random bits of plastic or rubber which may be useful in re-assembling.

Right Be careful that you don't lose any small parts when opening up the controller



The build

03 You should be now have a reasonably good idea of the controller that we'll be working with. 'The Duke' has dual joysticks, six buttons, a D-Pad and two triggers – and it's compatible with most retro games systems.

Fitting

04 If you're using a different controller, double-check that the Pi is likely to fit inside before you crack it open. As you'll see here, the Pi nestles neatly between the triggers of this controller – the original Xbox controller is one of the largest.

Unscrewing

05 The controller is held together by half a dozen cross-head screws. Be careful when opening the case as the buttons and rubber contacts are loose within the controller – they will spill everywhere!

Opening

06 With the shell removed, you should be able to undo the screws holding the main circuit board in place. There are also a couple of connectors which power the vibration motors – gently unclip them in order to completely remove the board. You might find it easier to use a pair of pliers for this – just be very gentle as you pull!

Gently does it

07 You can see for yourself just how well the Pi fits here; it can be squeezed under the memory card slot. If you want to hold it firmly in place, use some BluTak as a temporary solution. Also, if you're using an older controller, it's worth giving it a bit of a clean. Remove the rubber contacts and gently swab under them using the isopropyl alcohol swabs.

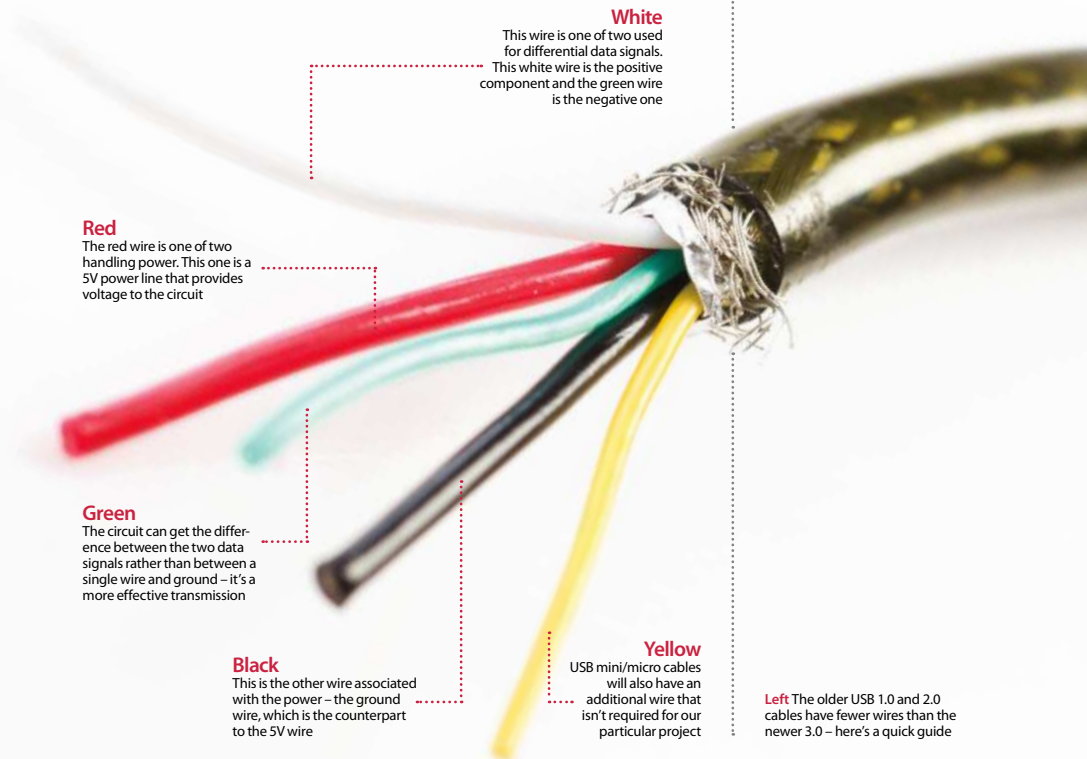
Cut to fit

08 Depending on the model of controller, you may find that the Pi blocks one of the internal plastic struts. The plastic is soft enough that a craft knife will easily cut it down to size, though. Start with small strokes, shaving off a tiny bit at a time until you have enough room. Make sure the plastic dust is cleaned out before you reassemble the controller. If you have a can of compressed air, you can use it to easily blow away the shavings.

Connecting it up

09 If you're using a controller that has a regular USB port on it, you can just plug it into the Pi via a USB OTG converter. If you're using the original Xbox Controller, it's slightly tricky. Microsoft, in its infinite wisdom, has decided that the original Xbox should use USB – but with an incompatible plug design. This means, in order to connect the controller to the Pi, we need to do some wire stripping. Fun!

The wiring inside the Xbox controller's cable uses bog-standard USB wiring colours, so once you've chopped the plugs off the controller and the OTG cable, it's pretty straightforward to connect them together.



Left The older USB 1.0 and 2.0 cables have fewer wires than the newer 3.0 – here's a quick guide



Above You can solder the OTG cable and controller together, but sticky-tape will also do the trick

The right controller

Second-hand stores like CEX or GAME often have some older, obsolete consoles and accessories out of public view, as they aren't particularly high-selling these days. It's worth asking the staff what they have if you can't see what you need on display. Some charity shops also have old consoles for sale. Failing that, local car boot sales or simply asking your gamer friends are both excellent ways to grab inexpensive controllers for all sorts of consoles.

Wiring

10 Strip the wires by a couple of centimetres and then connect them together. You should have Red, Green, White, and Black. The Xbox cable also has a Yellow wire which you can ignore. It is worth noting at this point that you need to be sure that you have a USB data transfer cable and not just a plain old power cable – the former will look like the photo above, but power cables will be missing the two data wires.

With the wires stripped, we temporarily used regular sticky-tape to make the connections between the OTG cable and the controller – for a more permanent installation, you can use electrical tape or simply solder the wires together.

Insulation

11 One thing to note: you'll need to insulate the bottom of the Pi against all the contacts on the controller. For this quick hack, we've used some of the cardboard packaging – but any non-conductive material will do.

From there, it's as simple as screwing the case back together. Make sure that the controller's buttons and joysticks don't slip out of alignment. Keep track of which coloured buttons go where and you should be fine.

Wiring up

12 The Pi will need three wires connected to it in order to work. The controller cable needs to be connected to the USB OTG port. An HDMI cable goes from your TV to the mini HDMI port on the Pi. Finally, a 2A micro USB power supply needs to be plugged into the Pi's power socket. We've used a standard mobile phone charger, but you can use a USB battery pack if you want to reduce the number of wires trailing around your room.

A word about power

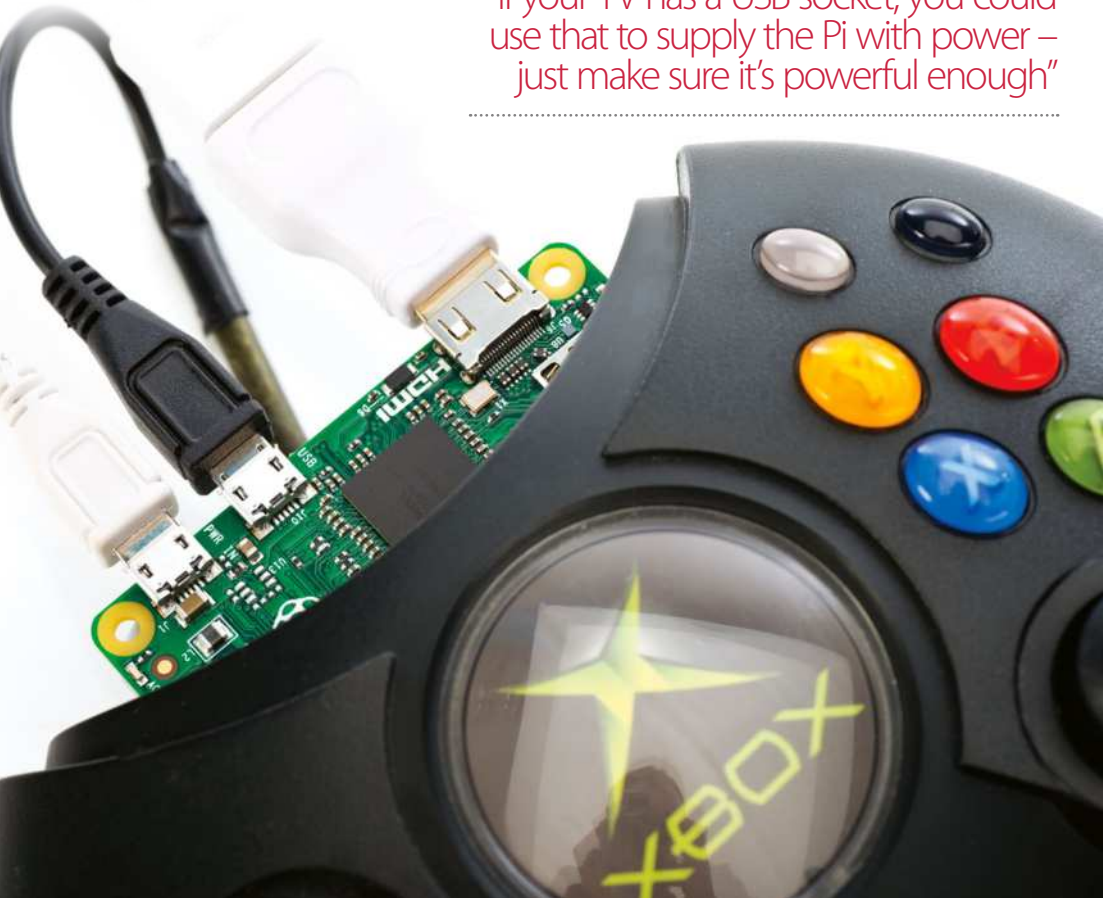
13 You might be wondering whether it's possible to get the HDMI cable to supply power from the TV to the controller. Sadly, the HDMI specification doesn't permit power to flow in that direction. If your TV has a USB socket on it, you could use that to supply the Pi with power – just make sure the socket itself is powerful enough. The Pi needs at least 1 Amp, and ideally 2 Amps. Many TVs will only output 500mA which isn't enough to run the Pi.



Let's play!

14 Okay, It's looking good – you're nearly ready to play. The next step is to get some emulation software on this thing.

"If your TV has a USB socket, you could use that to supply the Pi with power – just make sure it's powerful enough"



What's an emulator?

An emulator is software which lets your computer pretend to be a different sort of computer. It will allow a Raspberry Pi Zero to run software originally designed for the Sega Mega Drive, or Nintendo N64, old DOS-based PCs, etc. Emulators aren't without their problems, though – it's nearly impossible to perfectly recreate a games console in software. Keep in mind that older games may have bugs ranging from minor sound and graphical glitches to full-blown crashes.

Installing the RetroPie emulator

It's not as difficult as you might think to run retro software through an emulator

Right, so you've managed to get your Pi safely ensconced in a controller and all wired up – all you need now are some videogames to play.

For this section of the tutorial we're going to be using the RetroPie emulator. By the end of this tutorial, you'll be able to play a number of games directly from your Raspberry Pi, provided that you legally own the ROM files, of course.

The whole process is as easy as installing the software onto your SD card and then copying across any games that you want to play. If you've already got Raspberry Pi OS installed on your Pi, you can install RetroPie alongside it – or you can dedicate the whole disk to the software if you'd rather.

Install RetroPie inside Raspberry Pi OS

01 If you've already started using your Pi and want to add RetroPie to it, you'll need to install the software from GitHub. The latest instructions can be found at github.com/RetroPie/RetroPie-Setup.

Open up a terminal on your Pi (for example, by SSHing into it from another machine, or by logging in directly to the Pi). Update your repositories and make sure the latest version of the Git software is installed:

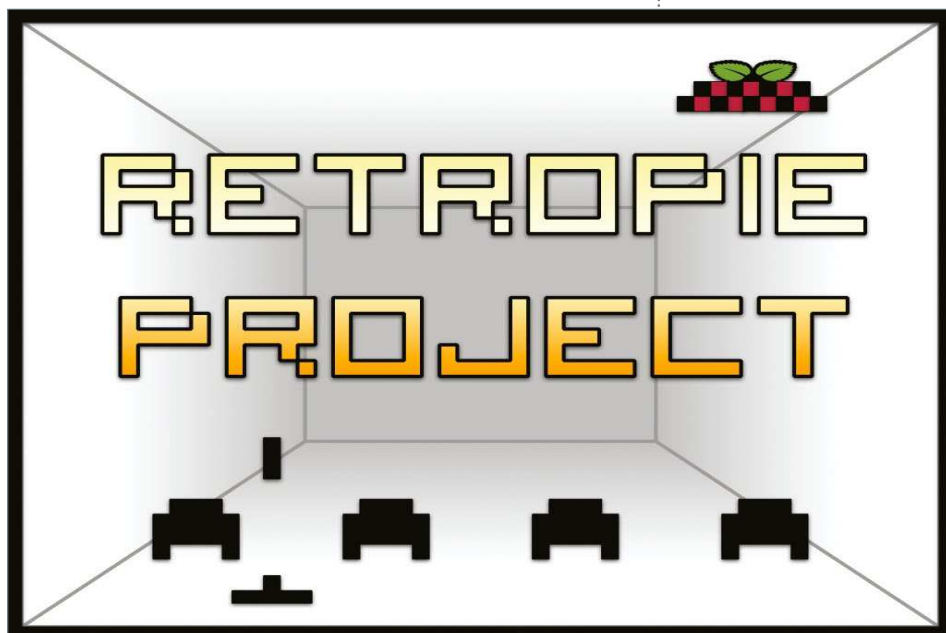
```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git
```

Download the latest version of the RetroPie setup script:

```
git clone --depth=1 https://github.com/RetroPie/RetroPie-Setup.git
```

If you're security-conscious, it's a good idea to check what the script does before running it. Once you're ready, you can install it by changing into the correct directory and executing the script:

```
cd RetroPie-Setup
sudo ./retropie_setup.sh
```



The script will take several minutes to run, depending on the speed of your internet connection. It may also ask you for permission to install extra software that is needed – you should allow this. Once fully installed, you will need to reboot your Pi:

```
sudo reboot
```

RetroPie can now be run by typing `emulationstation`. We'll come on to configuring your setup in just a moment.

Install RetroPie onto a blank SD card

02 If you want your Raspberry Pi Zero to be used solely as a RetroPie machine, this is the choice for you. Be warned: it will completely wipe a micro SD card, so if you're using one you've used before, make sure you back up any important data before starting.

Download the latest version of the software from <http://blog.petrockblock.com/retropie/retropie-downloads>. Make sure you download the correct SD card image for your machine – the image for the Raspberry Pi 2 is not compatible with the Raspberry Pi Zero. Download the Standard version (not the BerryBoot version). The download is an 800MB .gz file. Unzip it and extract the .img file, which will be around 2.6GB.

You'll now need to write this image file onto your micro SD card. This is done in the same way that you would install a normal Raspberry Pi image onto a card. There are slightly different instructions for Linux, Mac and Windows.

Above If you see a splash screen like this when you power on again, the installation worked!

Where to get ROMs

Many older games have, effectively, been abandoned. The original publishers are defunct and it's not clear legally who owns the rights. There are several sites which claim to have permission from the original creators to distribute their games – but it's not always easy to tell how legitimate they are. You should ensure that you either buy legitimate copies or download from organisations with the legal right to distribute them.

Below RetroPie can be restored straight to SD if you don't need Raspberry Pi OS as well

Installing the RetroPie emulator

Linux

03 Use the Disk Manager to select the image file and the micro SD card. Follow the on-screen instructions until the image has been fully written to the card.

Mac

04 Download the ApplePi Baker from www.tweaking4all.com/hardware/raspberry-pi/mac-osx-apple-pi-baker. Once you have it installed, you can select the image file and the micro SD card. Follow the on-screen instructions.

Windows

05 Download the Win32 DiskImager from <http://sourceforge.net/projects/win32diskimager>. Once installed, select the image file and the micro SD card. Follow the instructions until the image has been written to the card.

Configuring

06 Right – you're almost ready to play. Put the micro SD card into the Raspberry Pi Zero, hook up the controller USB cable and the HDMI cable. Finally, plug the Pi into the power. It should boot up automatically and, after a few seconds, you'll be greeted with a configuration screen.

RetroPie should automatically detect any connected USB game pads and step you through setting up the buttons. Once you've finished, you'll be presented with a screen showing all the choices you made.



Installing the RetroPie emulator

Set up the disk

07 Before we get to playing any games, we need to make sure that RetroPie is able to use all the space on the micro SD card. This will allow you to store ROMs and save your games. Select 'RetroPie' from the menu. You'll be presented with several configuration options. Select 'Raspberry Pi Configuration Tool RASPI-CONFIG'

You can change the default username and password at a later date; for now just use the controller to select 'Expand Filesystem'. Next, highlight the 'Select' button and click on it. After a short delay, you will see a success screen – press OK and you'll be taken to the configuration screen. Press right until 'Finish' is highlighted, then click on it. You should now reboot your Raspberry Pi.

Adding ROMs

08 The final step is adding new ROMs. Once you've legally purchased and downloaded ROMs from the internet, you'll need to copy them onto the micro SD card. ROMs are stored in a separate folder for each system. So, for example, you need to place your Sega Master System ROMs in `~/RetroPie/roms/mastersystem/`. Once you've installed ROMs, you're ready to play.

Projects

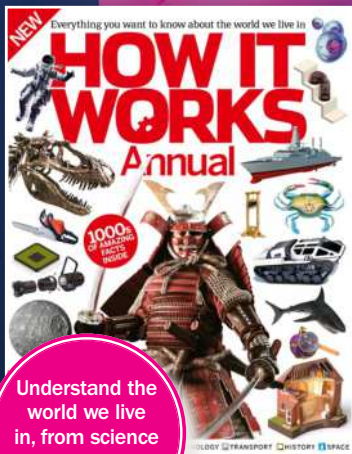
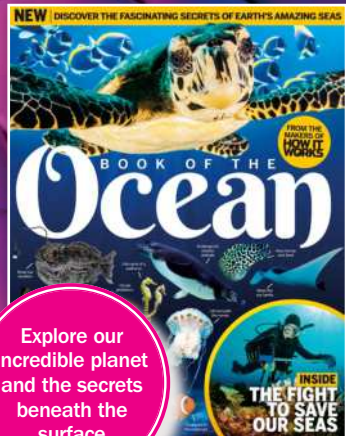


Playing

09 Once booted, you'll see a menu with all the available games systems on it. Some emulators will only show up once game ROMs for that system are installed. Scroll until you find the game you want to play – then let rip!

You can always return back to RetroPie if you want to change any of the configuration options, or update the software. And that's all there is to it! Time to sit back and play some games. If you want to find out more about the RetroPie software, visit <http://blog.petrockblock.com/retropie>.





Find out everything you've ever wanted to know about outer space

Explore our incredible planet and the secrets beneath the surface

Understand the world we live in, from science and tech to the environment



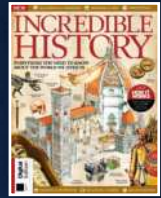
Get great savings when you buy direct from us



1000s of great titles, many not available anywhere else

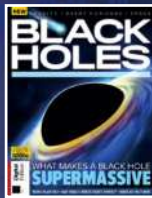


World-wide delivery and super-safe ordering



FEED YOUR MIND WITH OUR BOOKAZINES

Explore the secrets of the universe, from the days of the dinosaurs to the miracles of modern science!



Discover answers to the most fascinating questions

Follow us on Instagram  @futurebookazines

www.magazinesdirect.com

Magazines, back issues & bookazines.



SUBSCRIBE & SAVE UP TO 61%

Delivered direct to your door
or straight to your device



Choose from over 80 magazines and make great savings off the store price!

Binders, books and back issues also available

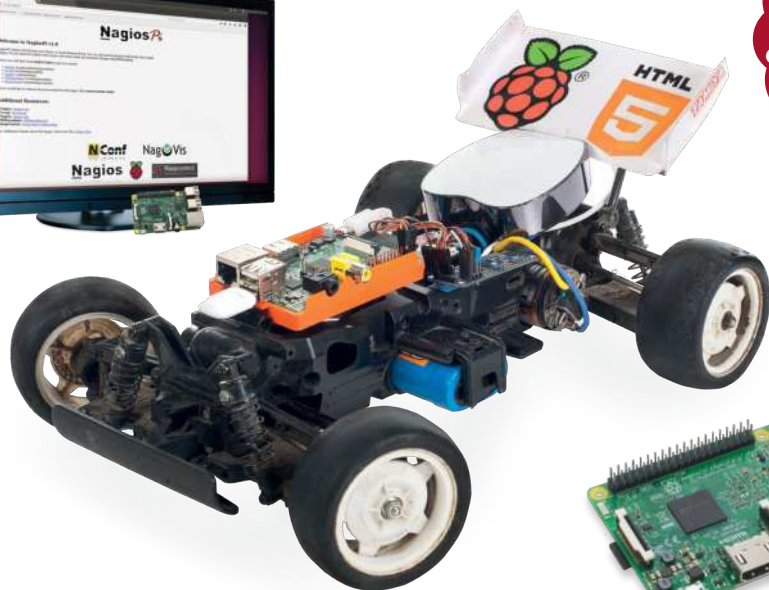
Simply visit www.magazinesdirect.com

✓ No hidden costs 🚚 Shipping included in all prices 🌐 We deliver to over 100 countries 🔒 Secure online payment



magazinesdirect.com
Official Magazine Subscription Store

**Pi 4
& Pi Zero
guides**



Raspberry Pi

The Complete Manual

- ✓ **Explore the Pi 4**
Find your way around some of the latest and greatest additions to the Raspberry Pi family
- ✓ **Set up your Pi**
Make sure you have the essential kit and the know-how to get started straight out of the box
- ✓ **Master Raspberry Pi OS**
Get comfortable using the official Raspberry Pi operating system by following simple guides
- ✓ **Understand applications**
Everything you need to know about the best applications included with the Raspberry Pi
- ✓ **Get to grips with Linux**
Learn how to use common Linux applications and master the command line basics
- ✓ **Get started with programming**
Navigate the world of Scratch and Python programming through easy-to-follow tutorials
- ✓ **Take control with your Pi**
Use the GPIO pins on your Raspberry Pi to control lights, switches and more
- ✓ **Create a wireless hotspot**
Access the Internet anywhere by tethering your Raspberry Pi to your Android device
- ✓ **Build & code with the web**
Create your own wearable tech, print wirelessly and tether a Pi to your smartphone
- ✓ **Creative projects made easy**
Play retro games, send SMS messages, build a Raspberry Pi car and much more!