Camera controller

Back up your photos using your Pi.

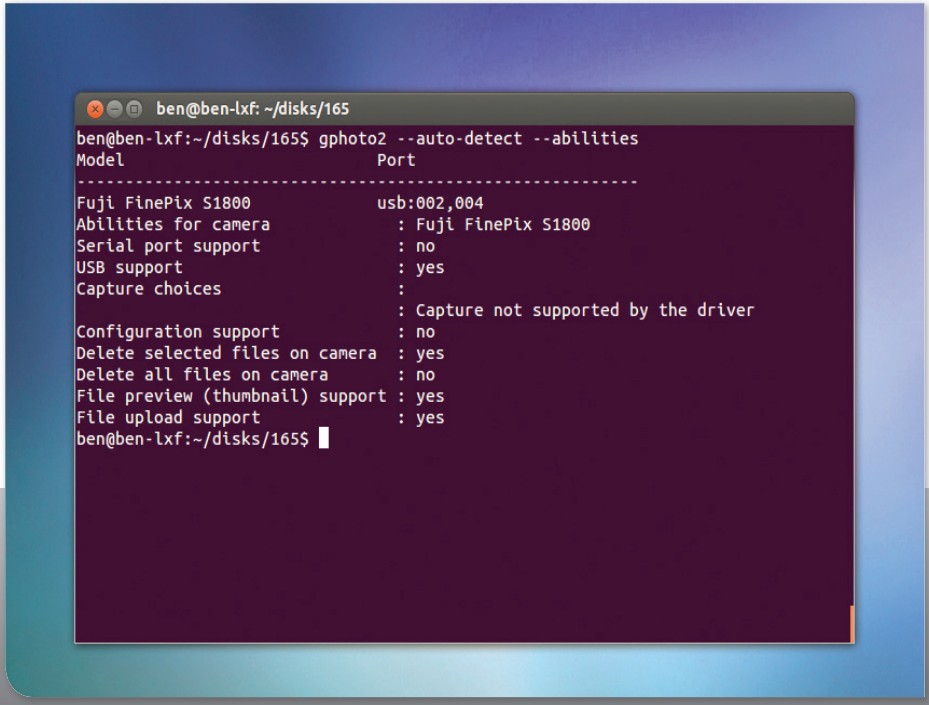
The size of the Raspberry Pi means we can use it to take control of other embedded devices.

This may seem a little redundant – the embedded devices obviously have some form of controller already – but it means we can script and extend them in ways that aren't possible (or are, at least, very difficult) without the extra device. Almost anything that you can plug in to a normal desktop can be scripted by a Pi, but we're going to look at cameras for a couple of reasons.

Firstly, there's support for most in Linux, and secondly there's a range of useful projects you can do once you've grasped the basics.

The best command-line tool for manipulating cameras in Linux is *Gphoto2*. Get it with:

```
apt-get install gphoto2
```



■ Gphoto2 has far more capabilities than we use here, including bindings for Java and Python. For full details, check out the project website: www.gphoto.org.

Before getting stuck in to the project, we'll take a look at this useful tool to see what it can do.

The desktop environment can try to mount the camera, and this can cause *Gphoto2* a few problems, so the easiest thing to do is run without it. Open a terminal and run **sudo raspi-conf**, and under Boot Behaviour, select 'No' to not start the windowing system, then reboot.

In the new text-only environment, plug in your camera and run:

```
gphoto2 --auto-detect
```

This will try to find any cameras attached to the Pi. Hopefully, it will pick up yours. While it does support an impressive array, there are a few cameras that won't work. If yours is one of the unlucky few, you'll need to beg, steal or borrow one from a friend before continuing.

Not all supported cameras are equal, so the next step is to see what the camera can do. To list the available actions, run:

```
gphoto2 --auto-detect --abilities
```

There are, broadly speaking, two main classes of abilities: capture, and upload/download. The former let you take photos with your scripts, and are present mostly on higher-quality cameras. The latter let you deal with photos stored on the memory card, and are present on most supported cameras. In this project, we'll deal only with the second set of abilities.

The simplest command we can send to the camera is to get all the photos stored on it. This is:

```
gphoto2 --auto-detect
--get-all-files
```

Running this will download all the files from the camera into the current directory. This would be fine on a normal computer, but you may not want to do it on a Pi, as you run the risk of filling up your memory card pretty quickly. Instead, we'll copy them on to a USB stick. To do this in an interactive session, you could simply use a GUI tool to mount the stick then run **df -h** to see where the USB stick is mounted, and **cd** to the directory. However, since this will run automatically, we need to know where the device will be. There are a few ways of doing this, but we'll keep it simple. We'll mount the first partition of the first serial disk, and store the photos there.

Powering your Pi

The Raspberry Pi gets power from its micro USB port. This supplies 5V, and the Raspberry Pi foundation recommends an available current of at least 1A. This can easily be delivered via a mains adaptor, or a USB cable from a computer. If you want your Raspberry Pi to be portable, however, there are other options. Four AA batteries, for instance, should provide enough power, provided you have the appropriate housing and cables to get the power into the micro USB port. However, we found the best solution was to get a backup power supply for a mobile phone that plugs directly into the Pi.

Here, we're assuming that you're using the default user pi. If you're not, you'll need to adjust the script.

First, we need to create a mount point for the drive. This is just a folder, and can be put anywhere – we're going to spurn convention and put it in our home folder. So before running the script, run:

```
mkdir /home/pi/pic_mount
```

With this done, we're ready to go. The script to mount the drive and get the photos is:

```
#!/bin/bash

if mount /dev/sda1
/home/pi/pic_mount ; then
        echo "Partition mounted"
        cd /home/pi/pic_mount
        yes 'n' | gphoto2 --
auto-detect --get-all-files
        umount /dev/sda1
else
        echo "/dev/sda1 could not
be mounted"
fi
```

yes 'n' is a command that simply emits a stream of n characters. This means that when *Gphoto2* prompts to overwrite any previously downloaded files, it will decline. The **umount** is essential, because it ensures that the drive is properly synced and can be removed.

We've called the script **get-pics.sh**, and saved it in Pi's home directory. To make it executable, run:

```
chmod +x /home/pi/get-pics.sh
```

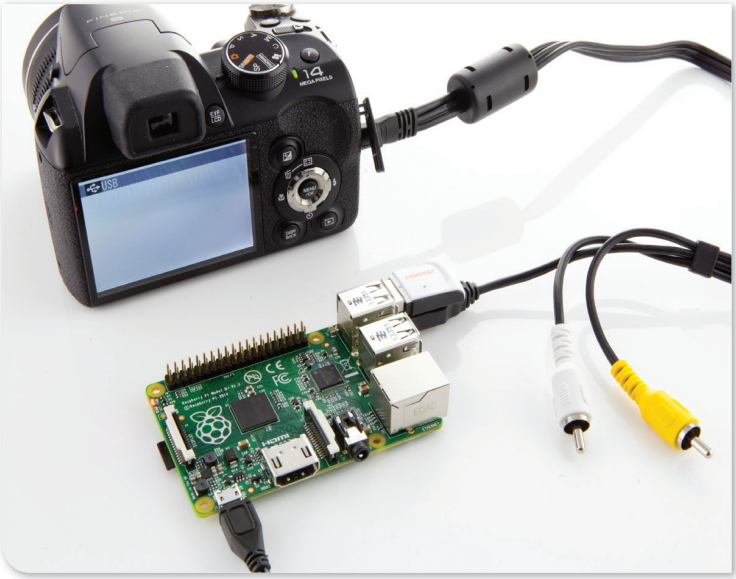
You should now be able to run it manually. You'll need to use **sudo** because it needs to mount the drive.

The final piece of the puzzle is to get the script to run automatically. To do this, we add it to the file **/etc/rc.local**. This script runs when you boot up, and it executes as root, so there's no need to worry about permissions. Just open the file with a text editor as root. For example, with **sudo nano /etc/rc.local**, and add the line:

```
/home/pi/get-pics.sh
```

just before the line exit 0.

Now all you have to do is plug in your camera (making sure it's turned on, of course) and your USB stick, and it'll back up your photographs when you boot up your Raspberry Pi.



In this form, it's not very portable, but with a little bit of judicious DIY, you should be able to package it more conveniently.

New directions

If you want to run the device headless, as will most likely be the case, you could attach LEDs to the GPIO pins, as shown over the page, and then use these to indicate statuses.

As well as saving them to the USB stick, you could upload them to an online service, such as Flickr. See the box on networking below for information on how to connect your Pi to your phone.

You could include some sort of switch to tell your Pi which photos to upload, and which to store on the USB stick – for example, upload low-resolution images, and store high-res ones. Or you could create low-resolution versions of the images, and upload those. Of course, you don't have to stop there. If you have a wireless dongle in your Pi, you could use it to run an HTTP server. With some PHP (or other web language) scripting, you should be able to create an interface to *GPhoto2* that will allow you to connect from your mobile phone.

Taking it in a different direction, if your camera supports capture options, you could use your Pi to take photos as well as copy them.

Networking

The Raspberry Pi comes with a wired Ethernet connection, which is fine for most occasions, but sometimes the cable just won't reach. You could use a USB wireless dongle – however, if you've got an Android phone, and your carrier hasn't disabled the feature, you can use this as your networking device. This has an extra advantage of not drawing as much power from the Pi, and so makes it easier when running from batteries.

You should be able to share your phone's connection to Wi-Fi as well as 3G, so it won't necessarily eat into your data allowance. Of course, it's best to check the connection type before downloading large files.

To do this, connect your phone to your Pi, and enable tethering in *'Settings > Wireless and Networks > Tethering and Portable Hotspots'* (on the phone). Back on the Pi, if you type **`sudo ifconfig`**, you should then see the interface **`usb0`** listed, but it won't have an IP address.

Networking interfaces are controlled by the **`/etc/network/interfaces`** file. By default, there isn't an entry in here for USB networking, so we need to set one up. Open the file with your favorite text editor as sudo. For example, with **`sudo nano /etc/network/interfaces`** and add the lines:

```
iface usb0 inet dhcp
nameserver 208.67.220.220
nameserver 208.67.222.222
```

This used the OpenDNS nameservers, but you could use others if you wish.

You can now either restart the interfaces or restart your Raspberry Pi to pick up the changes. You should have an internet connection up and running.