Webcams: Big

Ian Barton shares the delicious feeling of being able to keep an eye on your property without being there. He may be watching you right now...

Before the phrase was appropriated by the freaks and ghouls of reality television, the phrase "Big Brother" served as a handy word-association stick with which to beat anyone who wanted to take away our civil liberties. If information is power, we should try to stop our elected representatives from collecting too much information about us, so that the people remain in control of the government and not the other way round. So what is *Linux Format* doing running a webcam tutorial? Surely we all believe in freedom with a capital F round here?

Well I'm not interested in suppressing the proletariat; I just want to look after my cows. I live on a farm and need to keep an eye what's going on outside at all times of the day and night. Maybe someone has driven into the yard and is trying to steal something, or perhaps the animals have decided to go for a midnight ramble. This article will show you how to build a relatively inexpensive security system, using webcams and motion detection software. If you have broadband, you can monitor your cameras from any location that is connected to the internet and you can be alerted via email if the system detects movement. Of course, you could go all Stasi with your new-found power, but we hope that you won't.

Our expert

Ian Barton is no ordinary farmer. While others are busy chewing hay or stroking their sheep, Ian is busy putting webcams into every barn, tractor and goat within reach.

Part 1 Configure the cameras

We shall be using a Debian-based system, but the instructions provided should work with almost any Linux distro. The only major problem you may encounter is getting your distro to recognise a particular model of USB webcam.

Webcams can be connected via USB, Ethernet or wireless (these are generally described as 'network cameras'). Linux support for USB webcams is patchy. The problem is made worse because manufacturers can use different chipsets in the same brand of USB webcam, so even if you find one that works, another one which is supposedly the same type may not. Network cameras usually contain an internal webserver and may contain a wireless transmitter, so they can be connected via Wi-Fi. We shall be using a Linksys WVC54GCA in this tutorial, because it's not too expensive, is readily available and can be made to work with Linux.

The first step to installing a USB camera is to work out exactly what chipset it is using. Plug it into your computer and run **lsusb**:

```
ian@scamper:~$ lsusb
Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 002: ID 046d:08af Logitech, Inc. QuickCam Easy/Cool
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 003: ID 04f2:b013 Chicony Electronics Co., Ltd
```

Here we can see that it's a Logitech QuickCam, which can be picked up for under £10 pounds. A quick spin on Google reveals that this is supported by the *gspca* driver. Many distros, including Ubuntu Hardy, have support for this built into the kernel, but the version of the driver supplied with Hardy doesn't work with that particular Logitech Webcam.

We can check on the support for a particular webcam by Googling for its device ID, in this case 08af. A bit of research showed that support for this webcam is available in the latest *gspca* driver, available from **http://mxhaard.free.fr/download.html**. The first step is to download the latest version for the 2.6 kernel from **http://mxhaard.free.fr/spca50x/Download/gspcav1-20071224.tar.gz**.

Once we have downloaded the driver we need to unpack it:

```
tar -xvzf qc-usb-0.6.6.tar.gz
```

Installing it is fairly straightforward. First, we must install the Linux headers and the **build-essential** package:

» **Last month** We souped up Bash to make it 10% more hardcore.

Brother is you

› **Knife-wielding ASBO hoodies could be anywhere!**

```
sudo apt-get install build-essential linux-headers-$(uname -r)
```

To build the *gspca* driver, run this as root:

```
./gspca_build
```

If all goes well the kernel module will be built and loaded. Next we plug in the webcam and *Cheese*, but any other application that uses a webcam would do. If you can see a picture of your ugly mug on the screen, you know the webcam is working.

To set the camera up we shall plug it into a network hub, rather than try to fiddle with a wireless connection. The first step is to change our computer's IP address so that it's on the same subnet as the camera. We shall set our computer to use 192.168.1.10. Now open *Firefox* and browse to the camera's default address of **http://192.168.1.115/** . You should see the camera's default welcome page. Click on Setup and enter the username and password (the default is admin admin). Adjust the IP address and netmask so that the camera is part of your own subnet. The network in this tutorial is part of the 192.168.0.0 subnet, so enter 192.168.0.26 as the camera's IP address and 255.255.255.0 as the netmask, and enter your time zone. It's also a good idea to enable LED operation, so you can tell by looking at it that the webcam is working. Don't forget to set your computer's IP address back to something in the 192.168.0.0 range when you have completed the setup.

By default the Linksys WVC54GCA streams video in MPEG4 ASF format. The video stream can be viewed directly using *MPlayer* or *VLC*:

```
mplayer http://cam.era.ip.address/img/video.asf
```

However, we need it to stream in MJPEG format. Enter the the camera's URL in your browser in the following format:

```
http://your.webcam.ip.address/adm/file.cgi?h_
videotype=mjpeg&todo=save
```

If required the camera be reset back to ASF format using the following URL format:

```
http://your.webcam.ip.address/adm/file.cgi?h_
videotype=mpeg4&todo=save
```

Part 2 Install the monitoring software

Now that we have a couple of working webcams we need to set up some software to monitor them. *Motion* (**www.lavrsen.dk/twiki/bin/view/Motion/WebHome**) strikes the right balance between functionality and ease of setup. It's worth studying the wiki on the *Motion* website, as *Motion* has many options that can be tweaked. *Motion* is available as a Debian/Ubuntu package, so installing on a Debian based system is just a matter of:

```
sudo apt-get install motion
```

› **Your webcam's should have a configuration web page.**

We are using Ubuntu, so *Motion* is installed as a daemon. This is great once we've configured everything. However, initially it's easiest to run it in standalone mode as a normal user, which allows us to easily try out various configuration options. The first thing to do is to stop the daemon:

```
sudo /etc/init.d/motion stop
```

Motion looks for its configuration files in several different places, and you can specify the location of the configuration file on the command line when you start motion. Ubuntu places the configuration files in **/etc/motion**. We are going to copy these files into a directory called **.motion** in our home folder, where we can edit them and try out the various options. Create the **.motion** file: »

Sending an alert

Here is an example **email_alert.sh** script:

```
#!/bin/sh
#Destination e-mail address
TO="ian@firewall.banter.local"
#Subject of the e-mail
SUBJECT="Motion detected"
```

» `mkdir ~/.motion`

Copy the default distribution files to the **.motion** directory, then change the ownership of the files, so we can edit them:

`sudo chown -R ian.users /.motion`

The main configuration file is **motion.conf**. To monitor more than one webcam, each monitoring process is started in its own thread and has its own configuration file (**thread1.conf**, **thread2.conf**, etc). Examples from the default install are included in the **~/.motion** directory that we have just created.

Oracular options

We shall start with a single webcam and set its options in **thread1.conf**. The **motion.conf** file is very well commented, with many possible options. Note that as we are using the thread files we must include them in **motion.conf** (see the bottom of the example **motion.conf** file for how to do this).

```
# Remember: If you have more than one camera you must
have one
# thread file for each camera. E.g. 2 cameras requires 3 files:
# This motion.conf file AND thread1.conf and thread2.conf.
# Only put the options that are unique to each camera in the
# thread config files.
thread /home/ian/.motion/thread1.conf
thread /home/ian/.motion/thread2.conf
```

We shall leave most of the options in **thread1.conf** set to the default. The ones we really need to edit are shown below. The **videodevice** setting assumes a USB webcam is being used. To find out what device our webcam is using, plug it in and type:

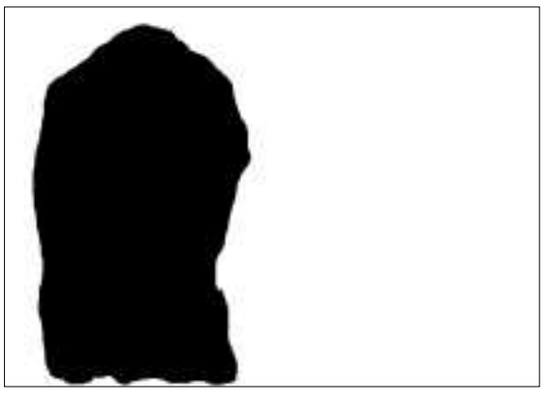
```
ian@scamper:~$ ls /dev/video*
/dev/video0
```

This will list all the connected video devices. Since we have only one USB camera it is **/dev/video0**.

Now we shall make the following changes in **thread1.conf**:

```
# Video device
videodevice /dev/video0
# Image width (pixels). Valid range: Camera dependent,
default: 352
width 640
# Image height (pixels). Valid range: Camera dependent,
default: 288
height 480
# Make automated snapshot every N seconds (default: 0 =
disabled)
snapshot_interval 0
# Target base directory for pictures and videos
```

Function	Option	Argument appended
Start of event (first motion)	on_event_start	None
End of event (no motion for gap seconds)	on_event_end	None
Still picture saved	on_picture_save	Filename of picture
Movie starts	on_movie_start	Filename of movie
Movie ends	on_movie_end	Filename of movie
Motion detected (each single frame with Motion detected)	on_motion_detected	None



› **Use a mask file to blank out areas of the camera's view that you don't want motion-detection to work on.**

```
# Recommended to use absolute patch. (Default: current
working
# directory)
# Must be writeable by Motion.
target_dir /home/ian/public_html/webcam/usb_cam
```

You can adjust the image height and width settings to suit your webcam's default. It's possible to make these smaller than your webcam's maximum settings if you wish. If you want motion to work like a conventional webcam app, taking snapshots at predetermined intervals, set the **snapshot_interval** in your **thread.conf** file to something other than zero. Most importantly you must set the **target_dir** option to somewhere that *Motion* can write to.

Is this thing on?

Now to test if it's all working. Start *Motion*:

`motion -c ~/.motion/motion.conf`

The **-c** option tells motion to use the configuration file **~/.motion/motion.conf**, not the one in **/etc/motion**. it's a good idea to always use the **-c** option and specify the location of the configuration file, as *Motion* looks in several different places for its config files and the last location found wins.

Try jumping up and down in front of your webcam a few times and take a look in the directory you specified in **target_dir**. If it's working you should see some AVI files. If you've set the **snap_shot_interval** to anything other than zero, you should also see some JPEG files. Note that motion always creates a file called **lastsnap.jpg**, which is a symbolic link to the most recent snapshot.

Now that we've set up a USB webcam, we'll move on to configure motion to work with our Linksys network camera. Create a **thread2.conf** (or edit the default one) and make sure it's included in **motion.conf**. Also make sure that you have used the hack above to set the Linksys to stream MJPEG, rather than ASF files. Comment out any existing **videodevice** line and insert the following:

```
#videodevice /dev/video1
netcam_url http://your.webcam.ip.address/img/mjpeg.cgi
```

Set the target directory so the network camera saves its files in a different place to the USB camera:

`target_dir /home/ian/public_html/webcam/usb_cam`

Press Ctrl+C to stop *Motion* running and restart it:

`motion -c ~/.motion/motion.conf`

You should now be able to see stills and video captured from your network camera.

» **Never miss another issue** Subscribe to the #1 source for Linux on p102.

Part 3 Refine your results

The default motion detection settings work well in most circumstances, but it's possible that you might be monitoring an area that overlooks a road, or has some trees. Movement in these areas would trigger a *Motion* event and we would end up being spammed with spurious warnings. To avoid this happening we can use a **mask_file**, which specifies an area of the picture where motion will be ignored. The mask file is a PGM (portable graymap) file which must be the same height and width as the frames being captured. Any area in which you do not want motion detection must be black, and any area where you do want motion detected must be white.

We can easily create a mask file using a captured frame from the webcam and editing it in *Gimp*. In the image below there is a sapling and a bush in the foreground, which trigger a movement event when they are blown by the wind. I fired up *Gimp* and edited this image to produce the mask file shown above left. You can see that we have excluded the area containing the sapling from motion detection by filling it with black. The rest of the image, where we want motion to be detected is filled with white. Make sure that you save the image as a PGM file in *Gimp*.

Now we need to tell *Motion* to use the mask file. In the **thread.conf** corresponding to the webcam edit the **mask_file** option:

```
mask_file /home/ian/webcam/my_mask_file.pgm
```

It's also possible to specify how many pictures frames in a row must contain movement before *Motion* detects it as true movement. This option is specified using **minimum_motion_frames**, the default being 1. The movement sensitivity can also be adjusted via the threshold setting in the configuration file. This specifies the number of pixels that must change between pictures before movement is detected. The default is 1,500.

Email notifications

Motion can run external commands on motion detection and related events. The events available are shown in the table over the page. You can specify a shell script in the event handler in either **motion.conf** or the appropriate **thread.conf** file. When *Motion* detects movement it creates an MPEG file of the event. In addition it also creates a JPEG file from one of the video frames. The point during the movement event that the JPEG file is created is configured using the **output_normal** setting in the **thread.conf** file. Setting this to "best" makes *Motion* save the image with the most changed pixels during the event.

› If someone's gone joyriding in my tractor, I want to know!

The various events can be used to send email notifications with either an attached MPEG or JPEG file. Since the MPEG files can potentially be quite large it's probably best to send the JPEG file showing the picture with the most changed pixels (around 50k), otherwise you may end up being spammed by videos of your cat running around in front of your webcam.

We are going to use *Mutt* to send an email alert with an attached JPEG. First we need to install *Mutt*:

```
sudo apt-get install mutt
```

Now we need to create a shell script to send the JPEG file. The *Motion* wiki indicates that you can use the **%f** placeholder in the event handler and it will be replaced by the filename. However, that didn't work for the version of Motion shipped with Ubuntu Hardy. We shall use the **on_event_end** event to send the JPEG.

Note that we have defined **jpeg_filename** (the name of the file to send) in our **thread.conf**. The filename format we use in our **on_end_event** handler must correspond to the one defined in **jpeg_filename**. Refer to the *Motion* wiki for details of the placeholders that you can use in filenames.

Edit the **thread.conf** corresponding to your camera:

```
jpeg_filename %v-%Y%m%d%H%M%S-%q
on_event_end send_mail ~/scripts/email_alert.sh /
  "/home/ian/public_html/webcam/linksys/%v-
%Y%m%d%H%M%S-%q"
```

Now we need to create the **email_alert.sh** script in our favourite editor. Don't forget to make the script executable:

```
#!/bin/sh
#Destination e-mail address
TO="ian@firewall.banter.local"
#Subject of the e-mail
SUBJECT="Motion detected"
echo 'Email body text goes here' | mutt -a $1 -s $SUBJECT
$TO
```

In the script **$1** is replaced with the name of the JPEG file, which is supplied the the *Motion* event handler. LXF

Resources

» **Linux gspca webcam driver:**
http://mxhaard.free.fr/download.html
» **Motion software motion detector:**
www.lavrsen.dk/twiki/bin/view/Motion/WebHome

And there's more…

You're not limited to receiving updates of intruders by email: Motion can keep you informed of what's going on via a mixture of its own embedded web server, PHP and FTP. To find out how, see the extension to this tutorial at **www.linuxformat.co.uk/mag/webcam**.

» **Next month** Open source version control the Torvalds way with Git.