Computer based Salmon Surveillance

Bjorn Hugsted

August 16, 2022

Contents

1 Abstract **2**

2 Introduction **2**

3 More on the HQ camera **3**

3.1 The HQ camera sensor 3

3.2 Lenses for the HQ camera 4

3.3 Calculating the image size on the sensor 4

4 Practical information **5**

4.1 Initial configuration of the operating system 5

4.2 Using wpa_cli for connecting to WiFi 6

4.3 Focussing the HQ camera 9

4.3.1 Using Raspbistill for focussing 9

4.3.2 Using libcamera for focussing 9

5 Example images **9**

5.1 16 mm lens 10

5.2 55 mm lens 10

5.3 1300 mm lens 10

6 Install and use of mjpeg_streamer **11**

6.1 Transfering the source files to user pi 13

6.2 Building the mjpeg_streamer 13

6.3 Starting and stopping the streamer 13

6.4 Allowing for the streamer to start at boot time 14

6.4.1 Adding service file to /etc/systemd/system 14

6.4.2 Adding commands to /etc/rc.local 15

6.5 Browser address lines for the mjpeg_streamer 15

6.6 Commends from the mscontrol program 16

1 Abstract

We have put together an assembly of lens, camera, and Single Board Computer (SBC) as gift for Jan Gravdal (DY) at his birthday. We also supply a CS-mount (or C-mount) to Canon EOS lens adapter assuming that Canon lenses may be used with the Raspberry Pi HQ camera. In addition we supply an adapter that may allow the HQ camera to be mounted to a telescop. The latter works with telescopes with 1.25 inch eye-piece-tube diameter. For use on a telescope the eye-piece should be removed and the camera mounted in its place.

2 Introduction

We have for a long time worked on video and still image transfer from remotely placed camera servers. Our main focus has been to provide reasonably priced observation stations that send video streams through the Internet to suitable viewstations. These cameras has fixed lenses and the image quality is sufficient for surveillance, but not for capturing video intended for presentation. This Salmon Surveillance project is a part of the more general Video Surveillance. Observe that with the recent development of the Salmon size we cound just as well say Smolt (or even parr) Surveillance.

Around 2020 Raspberry Pi presented a new, somewhat more expensive, camera - the High Quality (HQ) camera. The HQ camera promised a much better image quality and we immediately started trying it out for our projects. At may 2022 the price asked by PIMORONI was about 55 £. In addition to the camera one will also need some sort of lens and a computer. Making a the cost of a total assembly getting close to 200 £. The cost makes the

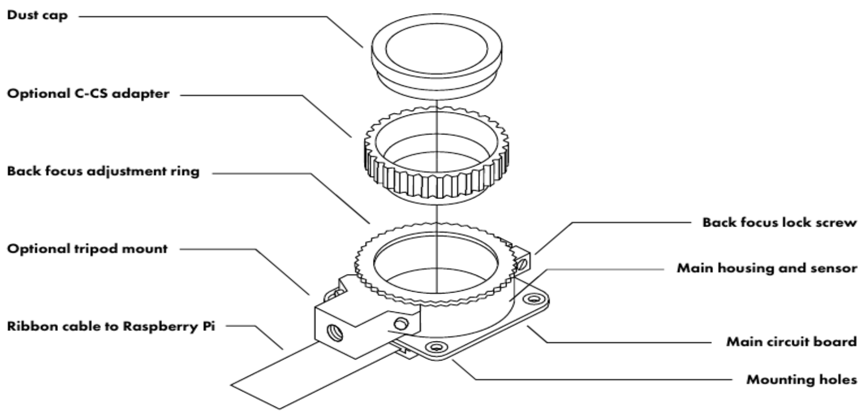

Figure 1: Drawing of the HQ camera lens mount

HQ out of reach for simple surveillance, but when a good image quality is needed the cost may be justified.

Much more information is avaliable in the file **Projects/VideoSurveillance/Admin_Guide/admin_guide.pdf**, but observe that this guide is written for the more general video surveillance project and filenames may differ.

3 More on the HQ camera

This camera is delivered without lens, but has CS-mount for a number of avaliable lenses. Also a CS to C mount adapter is delivered for using C-mount lenses with the HQ camera. A drawing of the lens mount of the HQ camera is shown in Figure 1. Also the innermost ring on the mount may be turned for finely adjusting the lens to sensor distance. It should at least be checked that one may focus to infinity. The ring is fixed by a small screw and a suitable screwdriver is enclosed in the camera box. We have acquired several of these and are in the process of evaluating wether they may be used for more serious work.

3.1 The HQ camera sensor

The sensor used in the HQ camera is the Sony IMX477R stacked, back-illuminated sensor with pixel numbers of 4056 times 3040. The sensor itself is of 6.287 mm times 4.712 mm size and each pixel is a square with 1.55 μm sides.

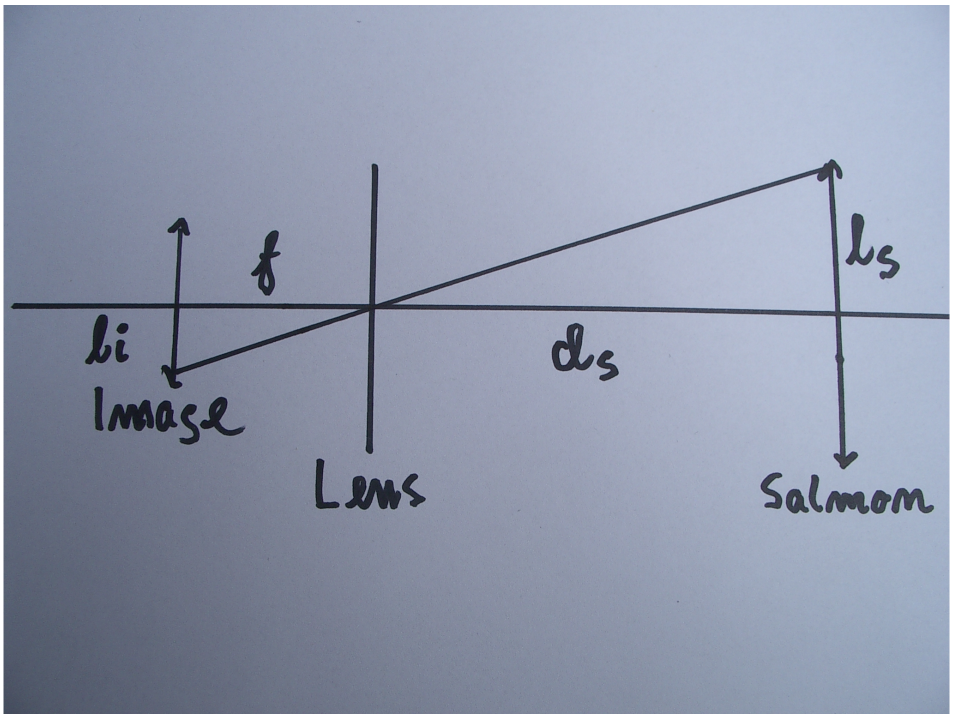

Figure 2: Principles of calculation image size

3.2 Lenses for the HQ camera

We have tried out a number of different lenses from the 6 mm wide angle lens to the 1300 mm Sky-Watcher Maksutov 102. In between these we have tried out a couple of very old Minolta lenses. Lenses produced for use with Single Lens Reflex cameras and system cameras may be used with the HQ camera if a suitable adapter can be found. We have used an adapter for the Minolta lenses, but the results are varying. The placement of the lenses never end up so that the focussing distance engraved on the focus ring actually agree with the distance to the object. Nevertheless we have obtained an adapter that should allow for mounting Canon EOS lenses to the Raspberry Pi HQ camera. If one should select one single lens it seems to us that the 16 mm lens is the best choice but the Minolta 55 mm lens also provides nice images.

3.3 Calculating the image size on the sensor

With a the assumption that the sceene is far from the camera we may use the approximation that the sensor is placed at the focal distance from the lens. Figure 2 show priciples of imaging a salmon standing upright and of height l_s at a distance of d_s. With this simple geometry the relation between sizes and distances is simply $l_s/d_s = l_i/f$. where f is the focal length of the lens. As an example say we want to get a good image of a salmon of length about 1.6 m that is splashing in a pond. We must be on a safe distance of say 10 m, but still wants the salmon to fill the image sensor for the best resolution. With the HQ camera having a sensor width of 6.3 mm we obtain $1.6/10 = 0.0063/f$ where the lens focal length calculates to $f = 0.0063 \cdot 10/1.6$ that again calculates to a lens of focal length of about 39 mm.

Closely related is the Field Of View (FOV) obtained with one particular focal length. I.e. $\alpha = 0.0063/f$ where α is the angular FOV in radians. We approximate $\sin \alpha = \alpha$. This is actually only true when α is small. For a normal objective the field FOV is about same perspective as with the eye, i.e. 45 degree that again is 0.8 rad. For the HQ camera an objective of about 8 mm focal lenght may be considered normal.

4 Practical information

The chosen machine for this project is the Raspberry Pi 4 model B. This Single Board Computer (SBC) has both Ethernet and WiFi communication, but initial configuration of the operating system is most conveniently done through an ordinary screen and keyboard combination. Details of the configuration is outlined below.

4.1 Initial configuration of the operating system

Usually we select the **lite** version of the operating system, but as the pi 4 is quite capable and for the convenience of a Graphical User Interface (GUI) we choosed to try the ordinary distribution, i.e. the **2022-04-04-raspios-buster-armhf**. We have chosen the buster instead of the new bulleye as the bullseye impements a new driver fopr the camer and this driver does not provide MJPEG compression. Buster includes the raspistill family of camera control commands while the bullseye changes over to libcamera.

At first start of the system (keyboard, mouse and screen connected) we are guided through some simple menues for setup. We were asked to supply a password for user **pi** and set the **SimplePiPass**. We went on to be located in Norway with timezone Oslo. Then went on to select english language and we did not tick off the US keyboard. The install then went on to downloading and installing updates and then a reboot.

After the reboot we are automatically logged in as user **pi**. We immediately changed the /etc/hostname from raspberrypi to **salmohq** and also did the same change in /etc/hosts. Using the **sudo passwd root** we set the password for root to **SimpleRootPass**.

Also the camera interface has been enabled from the **raspi-config** program. Select the **3 Interface options** and further the **P1 Camera**. Select **yes** to enable the camera and on exit do a reboot to make the changes effective. Enabling the camera does need a reboot, but we will do that later.

When this machine is started it obtains an IP-address through the Distributed Host Configuration Protocol (DHCP) and if the DHCP server also provides name to address translation one may use the name **salmohq** or **salmohq.local** with programs like the netbrowser, ping, and ssh.

The final placement of the salmohq will usually be at a location where the computer is not easily reached so it is of interest to have the server for Secure Shell (SSH) enabled. When user **root** shall be able to log in through **ssh** one will also have to modify the file **/etc/ssh/sshd_config** and replace the line **#PermitRootLogin prohibit-password** with **PermitRootLogin yes**. Enabling the sshd to start at boot may be done with the command: **systemctl enable ssh.service**. To start immediately use: **systemctl start ssh.service**. The ssh allow for remote login, but is also used by the Seure CoPy **scp** command.

After all these modifications one may do a reboot to assert that the system actually starts as expected and that the camera interface is enabled. Futher configuration may be done at the connected keyboard or through the secure shell.

4.2 Using wpa_cli for connecting to WiFi

While other distributions use the NetworkManager for maintaining network interfaces the Raspios seems to adher to the Wireless Supplicant that is controlled by the wpa_cli program. Started without parameters the wpa_cli enters interactive mode and you should select the interface, add a network, configure it, connect, and if satisfied save the configuration. Remember: We must also check with "rfkill list" that the WiFi is not blocked. These steps are shown below.

Before anything can be done with the WiFi one must make sure the device is not blocked by **rfkill**. Use the commend **list** to see the status of radio devices:

```
root@raspberrypi:~# rfkill list
0: phy0: Wireless LAN
        Soft blocked: yes
        Hard blocked: no
```

```
1: hci0: Bluetooth
        Soft blocked: no
        Hard blocked: no
root@raspberrypi:~#
```

In this case the Wireless LAN was soft blocked and need to be unblocked with: **rfkill unblock 0**. Now the WiFi may be connected using wpa_cli:

```
# wpa_cli
> interface wlan0
Connected to interface 'wlan0.
> scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
> scan_results
bssid /  / ssid
c8:be:19:5d:2a:19       Maltwhisky2.4
08:62:66:8c:a6:58       Kiara
28:28:5d:05:9a:e7       Telenor2754lag
fa:8f:ca:34:e8:15
40:a5:ef:96:be:42       ASUS
> add_network
0
> set_network 0 ssid "Maltwhisky2.4"
OK
> set_network 0 psk "VerySecretPassword"
OK
>
> enable_network 0
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
<3>Trying to associate with c8:be:19:5d:2a:19 \
    (SSID='Maltwhisky2.4' freq=2412 MHz)
<3>Associated with c8:be:19:5d:2a:19
<3>WPA: Key negotiation completed with \
    c8:be:19:5d:2a:19 [PTK=CCMP GTK=TKIP]
<3>CTRL-EVENT-CONNECTED - Connection to \
    c8:be:19:5d:2a:19 completed [id=0 id_str=]
>
> status
```

```
bssid=c8:be:19:5d:2a:19
freq=2412
ssid=Maltwhisky2.4
id=0
mode=station
pairwise_cipher=CCMP
group_cipher=TKIP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
ip_address=192.168.1.190
p2p_device_address=ba:27:eb:85:68:fe
address=b8:27:eb:85:68:fe
uuid=d520f2c3-1708-5254-9fdf-b4e30f2f2f1b
>
> save_config
> quit
# ifconfig
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
inet 192.168.1.190  netmask 255.255.255.0  \
    broadcast 192.168.1.255
inet6 fe80::b932:dcce:bd82:e94d  prefixlen 64  \
    scopeid 0x20<link>
ether b8:27:eb:85:68:fe  txqueuelen 1000  (Ethernet)
RX packets 28  bytes 3498 (3.4 KiB)
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 27  bytes 4253 (4.1 KiB)
TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
#
# ping storage
PING storage (192.168.1.3) 56(84) bytes of data.
64 bytes from storage (192.168.1.3): \
    icmp_seq=1 ttl=64 time=16.0 ms
```

Observe that in the set_network 0 ssid and psk the "" are mandatory, we tried without and always got the reply FAIL. The save_config command makes the settings persistent and the WiFi will reconnect to same network at next boot. If you are just exploring the WiFi you may choose to avoid this command. In the scan_results listing we have removed frequency, signal level, and flags to fit the page. Other lines have been broken with a \ for the same reason.

To disconnect from a network one may use the **disable_network ID** where ID is the network number obtained from the **add_network** and listed in the **status**. The command **remove_network ID** will remove the network from the list of known networks.

4.3 Focussing the HQ camera

Focussing the camera is a real pain when the image is displayed on a remote desktop. Another method is to arrange for the image to be displayed on a connected screen. We have been using a 1024 times 768 DVI screen that works nicely. This may also be a good time to adjust the "back focus adjustment ring" of the camera.

When turning the focus ring on an ordinary lens the setting for infinity is when the lens is closes to the detector plane. A possible procedure for adjusting the "back focus adjustment ring" would to first adjust the lens for infinity. Thet would be fully retracted. Then loosen the "back focus lock screw" and rotated the ring it until an object far away is in focus. Remember to tighten the lock screw afterwards. Be warned that this procedure may not succeed with every lens.

4.3.1 Using Raspbistill for focussing

A keybord for the Pi is actually not reqired, but may be convenient. The following command may be given at the console or a **ssh** terminal and the display is still on the Pi-connected screen. The command **raspbistill -t 0** will display images from the camera until terminated by CTL-C.

4.3.2 Using libcamera for focussing

As of the bullseye version of Raspios the new libcamera driver system is replacing elderly code and the raspistill is no more available. This also had some implications for our own version of the image streamer so we are still using the buster, but at some time in the future we will need to change to the new driver system.

5 Example images

To get an impression of what images to expect we have captured two images using the raspistill program. We use the command "raspistill -t 0" to focus the camera. When satisfied we hit CNTRL-C and then use the command: "raspistill -o 16mm.jpg" to capture an image in JPEG format to the file 16mm.jpg. We did this with the supplied 16 mm lens and also with a 55 mm lens. The distance from the object to the sensor was about 1.60 m. Later inspection of the images show the dimensions to be 4056x3040 pixels indicating that without further options the raspistill capture full resolution images.

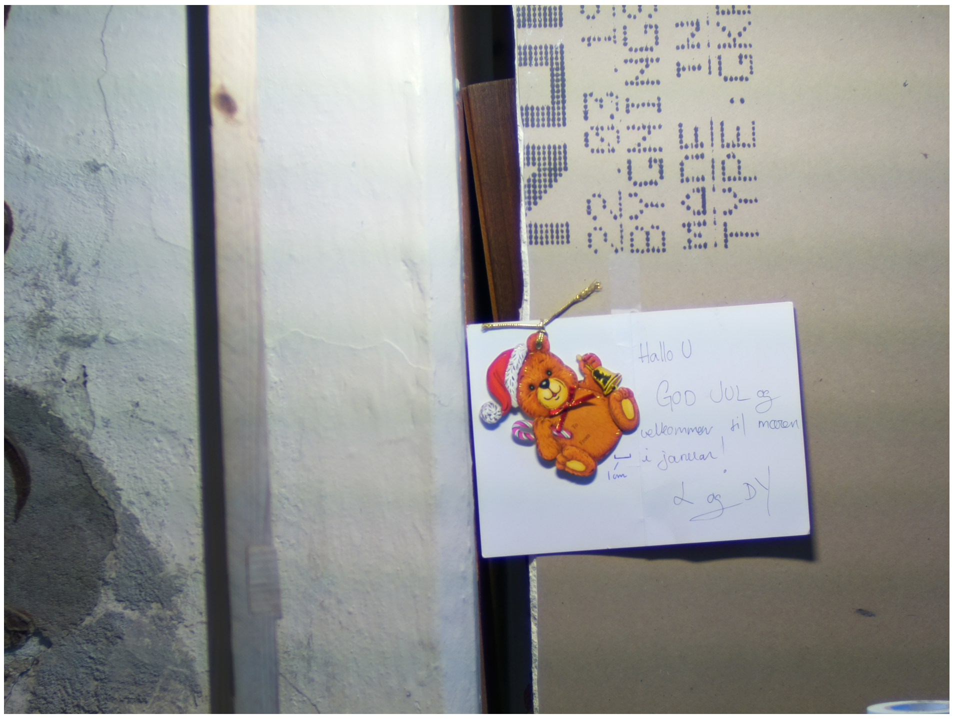

Figure 3: HQ camera image with 16 mm lens

5.1 16 mm lens

Some information is given at the PIMORONI product page, viz. Resolution 10MP, Image format 1”, Aperture F1.4-16, Mount C, Back focal length 17.53mm The lens is a C-mount device, so it has a longer back focus than the 6 mm lens and therefore requires the adapter. Figure 3 show an image of a familiar object at 1.6 m distance.

5.2 55 mm lens

This lens is an about 50 year old lens from a Minolta Single Lens Reflex (SLR) camera. We use a C-mount to Minolta adapter for mounting the lens to the camera. We also have another Minolta lens, a 200 mm, but with this lens we could not obtain a focussed image.

5.3 1300 mm lens

We also did some testing with the 1300 mm focal length Maksutov 102 telescope. Image 5 shows the telescop and HQ camera box. In the distance we may also see the telephone pole that is shown in Image 6. The image

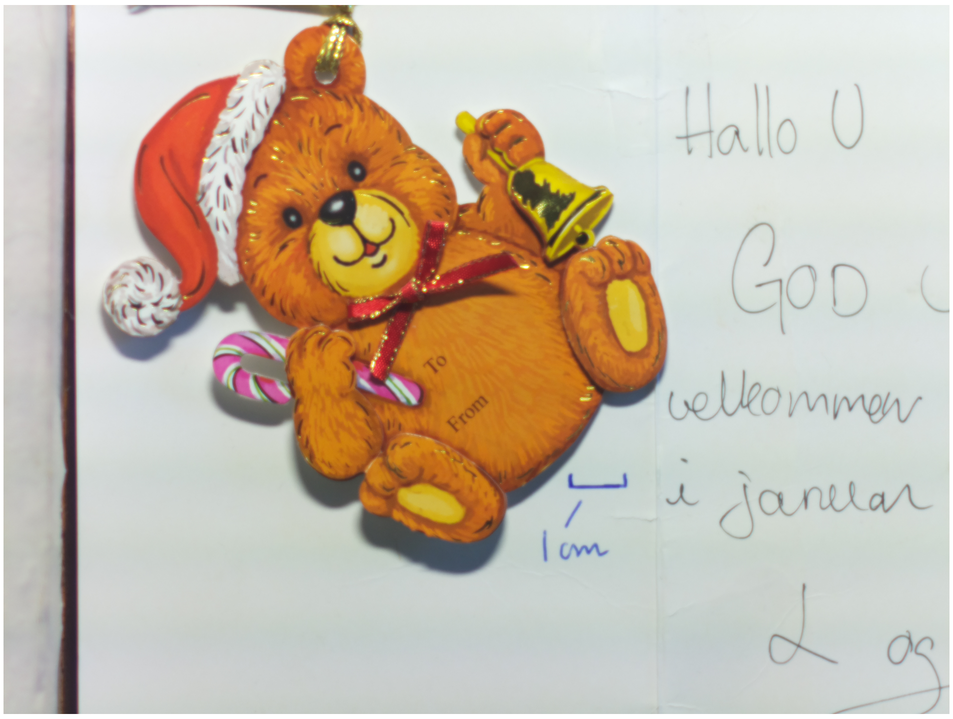

Figure 4: HQ camera image with 55 mm Minolta lens

through the telescope is within the red rectangle shown in Image 5. Imaging though the telecope is challenging. Focussing is difficult as every motion makes the image unstable. The focus does not seem to be very good and the contrast seems to be low. In part this may be because the image has been taken through an ordinary glass window. Theoretically the resolution of the telecope should be much better.

6 Install and use of mjpeg_streamer

While working with streaming images for the Video Surveillance projects we found the time to write our own video server. This server is built with a close look at Tom Stoevecken's mjpg-streamer. No code has been directly copied, but we do think that to start with coding on Video for Linux 2 (V4L2) is so involved that it would be very difficult when the V4L2 documentation is the only source of information. Our attempt is called mjpeg_streamer out of respect to Tom's much larger and more flexible mjpg-streamer.

Observe that the mjpeg_streamer is a work in progress. We have not been able to utilize the H.264 compression and even the MJPEG compression seems to be lower quality that what is obtained with software compression

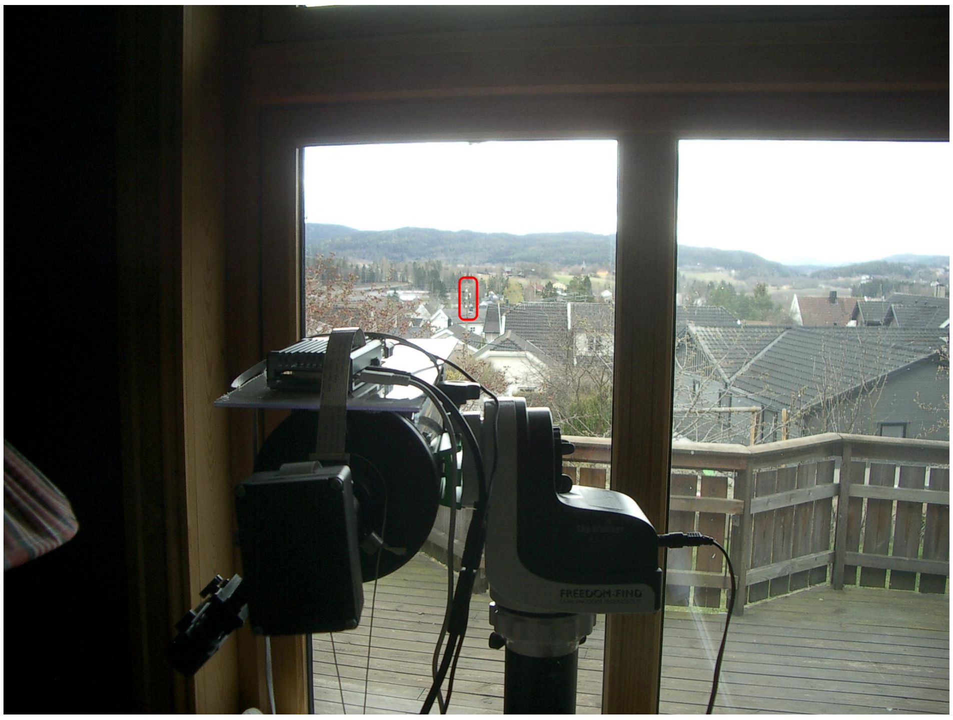

Figure 5: HQ camera mounted on the telescope

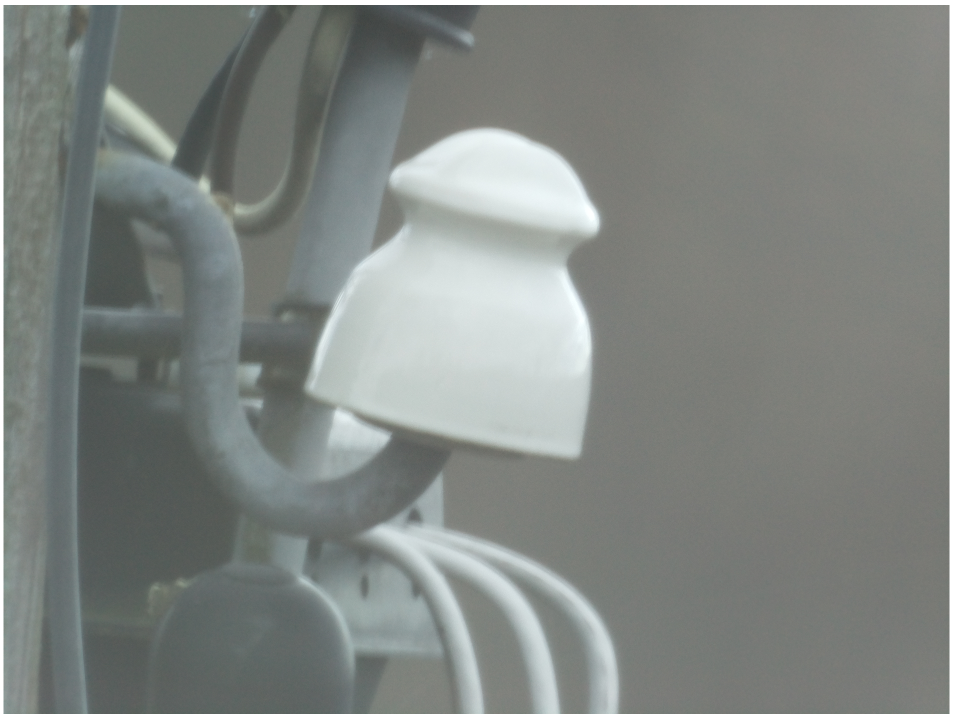

Figure 6: HQ camera mounted on the 1300 mm telescope

algorithms. The V4L2 has several settings for the quality of compression so setting other values may result in better quality.

6.1 Transfering the source files to user pi

As a start we need to copy the necessary files for building the **mjpeg_streamer** from the development machine to the **salmohq**. We have created a Bash shell script for gathering all the source parts that are needed to build the **mjpeg_streamer**. This script must be run when positioned above the **Projects**, the **src.dev**, and the **Own_Papers**. Pieces of code will be fetched from all of these. When correctly positioned the script is run as: **$ ./Projects/Video_Surveillance/Admin_Files/tgz_salmohq_-complete_parts.sh** this will produce the file **salmohq_complete_parts.tgz** that should be transferred to the machine salmohq. The **tar** file will be about 40 MB, but would be smaller if we had taken the bother to clean first.

When the ssh daemon is running at salmohq we may transfer the file with: **$ scp salmohq_complete_parts.tgz pi@salmohq:salmohq_complete-_parts.tgz**. The file will end up at user pi's home directory. Logged in as user pi the files are unpacked with **tar -xzvf salmohq_complete_-parts.tgz**.

6.2 Building the mjpeg_streamer

One should now log in to salmohq as user pi. Just after login one is situated in pi's home directory where the file **salmohq_complete_parts.tgz** has been transferred. This file should be untarred with the command **$ tar -xzvf salmohq_complete_parts.tgz**. Enter the directory to build the **mjpeg_streamer** with **$ cd Projects -Video_Surveillance/V4L2_Streamers** and do a **$ make clean** followed by a **$ make**. This will build the **mjpeg_streamer** for the correct architecture.

6.3 Starting and stopping the streamer

For development and testing the **mjpeg_streamer** can be started from the terminal session. Either without paraneters or with one single parameter that will be taken as the name of the configuration file. When there is no parameters the configuration file is assumed to have the name **"DEFAULT.txt"**. This configuration file must contain all value needed for running. When the file has been read the program initializes and starts the streaming. We believe that the configuration file is self-documented by commebts in the file so no further explanation of the contents of this file are given. If any problem is detected during the startup the program will terminate itself. Please have a look at the default configuration file for relavant values. **DEFAULT.txt** should be useful as a starting point for creating more specialized versions.

If started from the terminal without beeing put into background excution the program may be stopped by hitting CTRL-C. This will send SIGINT signal to the program. The **mjpeg_streamer** picks up this signal and terminates. During startup the **mjpeg_streamer** will write a Process ID (**PID**) file that contains the **PID** of the running program. The file will have the name of the running instance of the streamer. This name is set in the configuration file and we usually use the same name as the file. The configuration file DEFAULT.txt will create a **PID** file with name DEFAULT.pid. We have written a script that reads the **PID** file and enters the PID to the **$ kill -INT xxx** command where xxx is the actual PID number. The script is called **mskill** and must take one parameter, viz. the first part of the name of the **PID** file. When the **PID** file has the name **DEFAULT.pid** we should use **mskill** as: **$ ./mskill DEFAULT**.

6.4 Allowing for the streamer to start at boot time

To separate production and development versions we place the currently useful vesion of the **mjpeg_streamer** in the directory **ssstreamer** in the home directory of user **pi**. The executable and the configuration file should be copied into this directory. Possibly also the **mskill.sh** for an easy way to stop the streamer. Of the two methods outlined below the use of the systemd seems to be more in correspondemce with the other services running on a debian machine. Previously we always used the method of adding commands to the **/etc/rc.local**, but now we have changed to using the **systemd** and controlling the start/stop/enable/disable of **mjpeg_streamer** through the **systemctl** command.

6.4.1 Adding service file to /etc/systemd/system

The **mjpeg_streamer** and **DEFAUL.txt** is placed in the **/home/pi/ssstreamer** directory. In **/etc/systemd/system** we create the file **ssstreamer.service** and fill in:

```
[Unit]
Description=Salmon Surveillance MJPEG streamer

[Service]
User=pi
```

```
WorkingDirectory=/home/pi/ssstreamer
ExecStart=/home/pi/ssstreamer/mjpeg_streamer
Restart=always

[Install]
WantedBy=multi-user.target
```

The source directory contains the file **vsstreamer.example** that is an starting point for creating the .service file. When a change is made to the .service file we have to **systemctl daemon-reload** and then the service may be started with **systemctl start ssstreamer.sevice**. To make the service start at a boot it should be enabled with **systemctl enable ssstreamer.sevice**. Furthermore, status of a service may be obtained by **systemctl status ssstreamer.sevice** and disabling the service is done with **systemctl disable ssstreamer.sevice**.

6.4.2 Adding commands to /etc/rc.local

We recommand the systemd method outlined above, but another way to arrange for the **mjpeg_streamer** to be started at boot time is to insert the command in the **/etc/rc.local**. The lines to start the streamer will then be:

```
cd /home/pi/ssstreamer
./mjpeg_streamer > /dev/null 2>&1 &
```

We will normally insert this as the last commands in **rc.local** and remember that the script should always end with **exit 0**.

6.5 Browser address lines for the mjpeg_streamer

The Univesal Resource Locator (URL) entered into the address line of the browser (Chrome, Edge, Firefox, Safari, etc...) is of the form protocol:// host :port / request and parameters. Here the protocol will normally be http or https and the host is the name or IP-address of the host running the server program to connect to. An IPv4 address shall be entered in its dotted form while an IPv6 address with its colon separated numbers must be enclosed in square brackets. Connection will be attempted to port number port that is normally 80 for http. To avoid this often used port number the mjpeg_streamer listens to port 8080. The request and parameters are the end of the URL and directs the server program to what information the client requests.

When the server receives the request it will be of the form **GET / HTTP** for a request witout any specific request while the **GET /info HTTP** makes the **mjpeg_streamer** return information about itself and the environment. The **mjpeg_streamer** handles the following targets:

/ Just the empty target - will give a list of legal targets

/info Display information about the running mjpeg_streamer

/stream Start streaming to the browser

/exslow Set extra slow speed. Possibly 1 image each second

/slow Set slow speed i.e. 3 images/second

/med Medium speed. Maybe 10 images each second

/fast This should be up to 30 images each second. If that is possible.

/single Display one single image

6.6 Commends from the mscontrol program

The **mjpeg_streamer** will also respond to a number of commands that may not be sent from from the browser. These are listed below, but are intended for obtaining information about and programmatic control of the **mjpeg_streamer**. We have written a program that sends these commands and then wait for a reply. This is the **mscontrol** that are included in the textbfV4L2_Streamers directory. The commands all start with SYS followed by a command. The available commands are:

PING The mjpeg_streamer will return the line SYS ECHO

ADDRESS Will return the words SYS ADDRESS followed by IPv4 and IPv6 addresses.

TERMINATE Will arrange for mjpeg_streamer to terminate gracefully. This may not agree with the systemd when that mechanism is used.

POWEROFF This is a command that shall shutdown the machine, but is not implemented yet. For the time beeing we do not know how to do this cleanly

This set of commands may be sent from the mscontrol program.