**HOWTO - Create a Raspberry Pi IP Camera with RTSP server.**

At a certain day I wanted to do some video surveillance, see if my kids are safe playing in the street.  After a little while searching on the net for viable camera systems I quickly realised that professional HD IP camera's don't come cheap these days.   Since I'm on a tight budget, I started searching for alternatives for this expensive equipment, and since I was already playing around with a Raspberry Pi decided to go further in that direction.   So I ordered a CSI camera module on some vendor outlet website, and started tinkering with it.  Luckily for me there are tons of articles on the net with information about the Raspberry Pi and it's camera module.  
[](https://3.bp.blogspot.com/-OrDGaEgV49E/VU8D5YWThCI/AAAAAAAAIxs/_SiFar6yj5E/s1600/raspberrypi-2.jpg)Since I already owned a Synology nas with a surveillance station software on it, the choice to build a compatible IP camera for this purpose was logical, since surveillance station already had a website and mobile applications where I could view my video feed.  
If I wanted to build my IP camera to be compatible with my Synology nas, it had to support either MJPEG or H.264 RTSP streaming.  I knew there is a GPU in the Raspberry PI capable of doing H.264 encoding, and a utility called raspivid to drive all this and get this to the stdout of our Rasbian OS, so it seemed logical to take benefit of that knowledge and go the H.264 way for streaming.  
Now that we had a way to get our video stream, the only thing that remained was pipe this raw video feed into some RTSP server for dispersion to my Synology nas.  
After searching the net (again :-)) for a while found that there are source-code libraries for standards-based RTP/RTCP/RTSP/SIP multimedia streaming, suitable for embedded and/or low-cost streaming applications provided by Live Networks, you can find their website at <http://www.live555.com/>  
Now that I had the whole plot, the only thing that remained was to get busy and put da thing together:  
  
To obtain this camera, you have two options, you can build it yourself according to the below instructions, or you can download a ready-built copy from: <http://ronnyvdbr.github.io/RaspberryIPCamera/> featuring a easily configurable PHP/Web interface.  
  
Download and install Miniban from <http://sourceforge.net/projects/minibian/files/latest/download?source=files> and put on SD card with win32diskimager.  
This is a minimal Rasbian installation which is compatible with the official Foundation Raspbian distribution.  (it's totally not a requirement, you can just as easily start off with the official foundation distribution)  
  
  
After booting up, log in with root and raspberry as password, then perform below steps to expand the root parition, default this is only 512 Mb, we want this bigger for the rest of our software:

* fdisk /dev/mmcblk0
* Then delete partitions with d and create a new with n. You can view the existing table with p.
* p to see the current start of the main partition
* d, 2 to delete the main partition
* n p 2 to create a new primary partition, next you need to enter the start of the old main partition (just press enter), and enter +2G to expand the partition to 2 Gb, Check the p output!
* w write the new partition table Now you need to reboot:
* shutdown -r now After the reboot you need to resize the filesystem on the partition. The resize2fs command will resize your filesystem to the new size from the changed partition table.
* reboot
* resize2fs /dev/mmcblk0p2

Let's regenerate our SSH keys, to privatise our SSH server:

* rm /etc/ssh/ssh\_host\_\* && dpkg-reconfigure openssh-server

Set the correct time zone for our Raspberry Pi:

* echo "Europe/Brussels" > /etc/timezone
* dpkg-reconfigure -f noninteractive tzdata

Generate locale data for our Raspberry Pi:

* sed -i "s/# en\_US.UTF-8 UTF-8/en\_US.UTF-8 UTF-8/g" /etc/locale.gen
* /usr/sbin/locale-gen

Update our Raspberry Pi to the latest version:

* apt-get update
* apt-get -y upgrade
* apt-get -y install rpi-update
* rpi-update
* reboot

Install raspi-config and enable our camera:

* apt-get -y install raspi-config
* execute raspi-config and select 5 to enable camera
* now reboot

Since the Miniban installation does not have a pi user set up, let create one and configure it:

* apt-get -y install sudo
* useradd pi
* echo 'pi:raspberry'|chpasswd
* usermod -a -G sudo pi
* mkdir /home/pi
* cp /root/.profile /home/pi/.profile
* cp /root/.bashrc /home/pi/.bashrc
* chown -R pi /home/pi
* chgrp -R pi /home/pi
* chmod -R 755 /home/pi
* sed -i 's/pi:x:1000:1000::\/home\/pi:\/bin\/sh/pi:x:1000:1000::\/home\/pi:\/bin\/bash/g' /etc/passwd
* sed -i 's/# export LS\_OPTIONS=/export LS\_OPTIONS=/g' /home/pi/.bashrc
* sed -i 's/# eval/eval/g' /home/pi/.bashrc
* sed -i "s/# alias ls=/alias ls=/g" /home/pi/.bashrc
* sed -i "s/# alias ll=/alias ll=/g" /home/pi/.bashrc
* sed -i "s/# alias l=/alias l=/g" /home/pi/.bashrc

Install our development environment for building software:

* apt-get -y install build-essential

Install Live555, Source-code libraries for standards-based RTP/RTCP/RTSP/SIP multimedia streaming, suitable for embedded and/or low-cost streaming applications.

* cd /home/pi/
* wget http://www.live555.com/liveMedia/public/live555-latest.tar.gz
* tar -zxvf live555-latest.tar.gz
* mv live live555
* cd live555
* sed -i 's/-D\_FILE\_OFFSET\_BITS=64/-D\_FILE\_OFFSET\_BITS=64 -DALLOW\_RTSP\_SERVER\_PORT\_REUSE=1/g' ./config.linux
* ./genMakefiles linux
* make

Download a small script from the Raspberry Pi foundation forum:

* wget https://www.raspberrypi.org/forums/download/file.php?id=4285
* tar -zxvf testRaspi.tar.gz
* cd raspi
* (edit the file testRaspi.cpp and change the OutPacketBuffer::maxSize to 400000)
* make

You can now enable a RTSP stream server, which can be viewed by a variaty of software players by launching the below command:

raspivid -t 0 -fps 30 -g 1 -b 2000000 -h 1920 -w 1080 -o - | ./testRaspi

With VLC player I can connect to the RTSP stream, i'm using 170 ms buffering in VLC to keep the image movement fluid.

I tried connecting this with my Surveillance application in my Synology diskstation (ds411j), this seems to work quite nicely.

### HOWTO - Raspberry Pi HD IP Camera (based on UV4L - Video4Linux drivercollection)

## [https://3.bp.blogspot.com/-8BtuOkpp5uY/Vc8wf1YpsqI/AAAAAAAAI_4/CPtilGwcrvc/s320/Pi2ModB1GB_-comp.jpeg](https://3.bp.blogspot.com/-8BtuOkpp5uY/Vc8wf1YpsqI/AAAAAAAAI_4/CPtilGwcrvc/s1600/Pi2ModB1GB_-comp.jpeg)[https://4.bp.blogspot.com/-vPXeVqaFoXM/Vc8z6eQallI/AAAAAAAAJAY/0XA-cT_v2jg/s200/034718_BB_00_FB.EPS_1000.jpg](https://4.bp.blogspot.com/-vPXeVqaFoXM/Vc8z6eQallI/AAAAAAAAJAY/0XA-cT_v2jg/s1600/034718_BB_00_FB.EPS_1000.jpg)

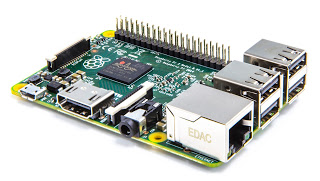
For anyone searching for high quality video surveillance on a tinkering budget, will admit that the Raspberry Pi provides a reasonable and affordable option.

When you search the internet for IP Cameras capable of doing higher resolutions than 640x480 pixels, you will notice a remarkable difference in price range.  There is a price gap between the standard home consumer IP Camera and the more professional higher resolution IP Camera, so if you want high quality home surveillance video feed but don't want to bear the price, the Raspberry Pi IP Camera is for you.

All Raspberry Pi Camera modules (currently as of writing there are three different models available) are capable of doing high definition resolution (and i'm talking about 2592x1944px).  For people interested in the full specifications of the Camera, they can be found on the Raspberry Foundation web site at: <https://www.raspberrypi.org/documentation/hardware/camera.md>

A bill of material for this guide looks as follows (prices are from adafruit's site):

A Raspberry Pi 2 board with power supply: 39,95$

[](https://3.bp.blogspot.com/-8BtuOkpp5uY/Vc8wf1YpsqI/AAAAAAAAI_4/CPtilGwcrvc/s1600/Pi2ModB1GB_-comp.jpeg)

A power supply for the Raspberry: 5,95$

[](https://4.bp.blogspot.com/-Wb6V1UU2jFg/Vc8yFZRPe_I/AAAAAAAAJAE/ZmSuEgCYSaI/s1600/31Iu6ulWyPL.jpg)

A Raspberry Pi Spy Camera module: 39,95$

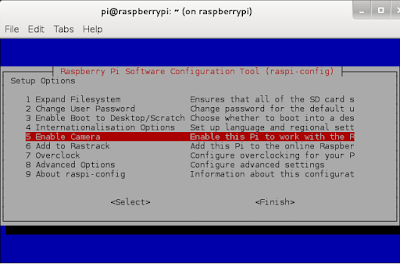
[](https://3.bp.blogspot.com/-oCqqRIHeKoo/Vc8y6_9Td4I/AAAAAAAAJAM/LNtVhzS_cGM/s1600/SpyPiCAM5.jpg)

A dummy IP Camera Housing: 5$

[](https://4.bp.blogspot.com/-vPXeVqaFoXM/Vc8z6eQallI/AAAAAAAAJAY/0XA-cT_v2jg/s1600/034718_BB_00_FB.EPS_1000.jpg)

This brings the total cost of our bill of material at 90,85$, not too bad for a capable high definition IP Camera.  
  
First job would obviously be, mounting all the parts together, but that's a bit beyond the scope of this tutorial.  If you're a bit of a diy handyman, this shouldn't be an issue.  
Once you have all parts mounted together, let's continue on the software side.

First step is downloading the [Raspbian image](http://downloads.raspberrypi.org/raspbian_latest) from the [Raspberry Foundation web site](https://www.raspberrypi.org/downloads/).  
Once downloaded, unzip your image (if you downloaded the zip archive) and write it onto an sdcard with [win32diskimager](http://sourceforge.net/projects/win32diskimager/).  
  
Next, let's continue booting up the image and doing some preparation work.  
Once you boot the Foundation's Raspbian image for the first time, you'll end up in the initial configuration menu, make sure to enable the Camera module here, or your Camera module won't be able to start.

[](https://4.bp.blogspot.com/-TZlVfNJxqp4/VdAzF_-_KxI/AAAAAAAAJAs/V8cx1zQ1KtA/s1600/raspi-config.png)

Now let's make sure that our Raspberry Pi is up to date with all the latest hotfixes, to update firmware and software, punch in below one-liner:

***sudo apt-get update && sudo apt-get -y upgrade && echo y | sudo rpi-update && sudo reboot***

Reboot your Raspberry Pi after this operation to properly activate the firmware update, if any.  
  
Next thing is to load the UV4L2 suite of drivers onto our Raspberry Pi.  
A clear set of [instructions](http://www.linux-projects.org/modules/sections/index.php?op=viewarticle&artid=14) for this can be found on the [linux projects](http://www.linux-projects.org/) website, but if you're lazy just like I am, you can just punch in the instructions below:  
  
To install UV4L open a terminal and type the following command:

***sudo curl http://www.linux-projects.org/listing/uv4l\_repo/lrkey.asc | sudo apt-key add -***

Add the following line to the file /etc/apt/sources.list :

***echo "deb http://www.linux-projects.org/listing/uv4l\_repo/raspbian/ wheezy main" | sudo tee -a /etc/apt/sources.list***

Update your apt cache:

***sudo apt-get update***

Now let's install our UV4L2 driver suite requirements:

***sudo apt-get install uv4l uv4l-raspicam uv4l-raspicam-extras uv4l-server uv4l-webrtc***

After installing the UV4L2 suite of drivers, these will be started automatically after reboot or u can start or restart them manually with below command:

***sudo service uv4l\_raspicam restart***

Upon start the drivers will load their configuration from below configuration file, you can edit it by pasting the same

***sudo nano /etc/uv4l/uv4l-raspicam.conf***

To get an overview of available parameters which you can list in the configuration file:

***uv4l --help --driver raspicam --driver-help***

Or to kill a running driver:

***pkill uv4l***

Once you have started the driver, a web gui will be available which u can use to configure parameters on the fly.

***http://ip-of-your-raspberry:8080***

This concludes the initial set-up of the IP Camera with drivers and control panel.    
The camera will only be useful this way to be viewed via the control panel and a web browser.  
If you want to connect it via VLC or a Video Recording Solution, like for example Surveillance Station on a Synology Nas, we need to install an extra server to run the RTSP protocol to serve the video feed.  
  
There's a nice project called [h264\_v4l2\_rtspserver](https://github.com/mpromonet/h264_v4l2_rtspserver) over on [Github](https://github.com/) written by a user named [Michel Promonet](https://github.com/mpromonet) which provide us with such a server, the only drawback is that you need to compile it yourself, but since you're a DIY handyman, follow below guide:

* sudo apt-get -y install cmake libv4l-dev liblivemedia-dev liblog4 cpp5-dev
* git clone https://github.com/mpromonet/h264\_v4l2\_rtspserver.git
* cd h264\_v4l2\_rtspserver/
* cmake . && make
* cpack .
* sudo dpkg -i h264\_v4l2\_rtspserver\*.deb

The RTSP server is now compiled and installed, the only thing what remains is to make it start at boot, for this purpose I wrote a little service script to start it via the normal debian services system.    
  
Copy and paste below script in a file, run this: ***sudo nano /etc/init.d/rtspserver*** then copy below into the file and save it.

#!/bin/sh### BEGIN INIT INFO

# Provides: RTSPSERVER

# Required-Start:    $local\_fs $network $named $time $syslog

# Required-Stop:     $local\_fs $network $named $time $syslog

# Default-Start:     2 3 4

5# Default-Stop:      0 1 6

# Description:       Provides the Mpromoneth rtsp server

### END INIT INFO  
SCRIPT="/home/pi/h264\_v4l2\_rtspserver/h264\_v4l2\_rtspserver -P 8554 -Q 10 -r -s -W 2592 -H 1944 -F 15"

RUNAS=root

NAME=h264\_v4l2\_rtspserver  
PIDFILE=/var/run/$NAME.pid

LOGFILE=/var/log/$NAME.log

start() {  if [ -f $PIDFILE ] && kill -0 $(cat $PIDFILE); then    echo 'Service already running' >&2    return 1  fi  echo 'Starting service…' >&2  local CMD="$SCRIPT &> \"$LOGFILE\" & echo \$!"  su -c "$CMD" $RUNAS > "$PIDFILE"  echo 'Service started' >&2}

stop() {  if [ ! -f "$PIDFILE" ] || ! kill -0 $(cat "$PIDFILE"); then    echo 'Service not running' >&2    return 1  fi  echo 'Stopping service…' >&2  kill -15 $(cat "$PIDFILE") && rm -f "$PIDFILE"  echo 'Service stopped' >&2}

uninstall() {  echo -n "Are you really sure you want to uninstall this service? That cannot be undone. [yes|No] "  local SURE  read SURE  if [ "$SURE" = "yes" ]; then    stop    rm -f "$PIDFILE"    echo "Notice: log file was not removed: '$LOGFILE'" >&2    update-rc.d -f <NAME> remove    rm -fv "$0"  fi}

status() {        printf "%-50s" "Checking $NAME..."    if [ -f $PIDFILE ]; then        PID=$(cat $PIDFILE)            if [ -z "$(ps axf | grep ${PID} | grep -v grep)" ]; then                printf "%s\n" "The process appears to be dead but pidfile still exists"            else                echo "Running, the PID is $PID"            fi    else        printf "%s\n" "Service not running"    fi}  
  
case "$1" in  start)    start    ;;  stop)    stop    ;;  status)    status    ;;  uninstall)    uninstall    ;;  restart)    stop    start    ;;  \*)    echo "Usage: $0 {start|stop|status|restart|uninstall}"esac

Now change the permissions on this script so it can be executed:

***sudo chmod 755 /etc/init.d/rtspserver***

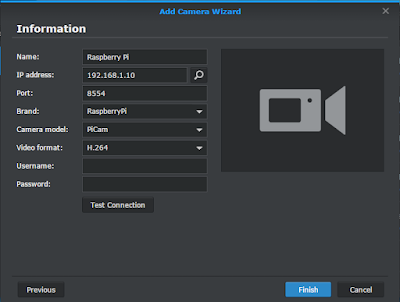
And schedule the script so it will start as a service on boot time:

***sudo update-rc.d rtspserver defaults***

After this reboot your Raspberry Pi to activate the RTSP server.  
After the reboot the RTSP server will be available, and you will be able to connect to it with VLC via the following mrl:  rtsp://ip-address-of-your-Raspberry-Pi:8554/unicast  
  
  
For the people who want to connect this to their Synology Nas, follow below instructions:  
  
First make sure you have installed Surveillance Station on your nas via the webgui.  
ssh to your nas and log in  
  
We need to create a camera definition for the Raspberry Pi so we can select this in our Surveillance Station.  
To do this create a configuration file for it (just copy and paste the whole damn thing):

***cd /volume1/@appstore/SurveillanceStation/device\_pack/camera\_support/  
echo [RaspberryPi\*PiCam] | tee RaspberryPi.conf  
echo api = rasbpicam-h264 | tee -a RaspberryPi.conf  
echo channel\_list = 1  | tee -a RaspberryPi.conf  
echo default\_channel = 1  | tee -a RaspberryPi.conf  
echo resolutions\_h264 = 2592x1944  | tee -a RaspberryPi.conf  
echo default\_resolution\_h264 = 2592x1944  | tee -a RaspberryPi.conf  
echo fps\_h264\_2592x1944 = 15  | tee -a RaspberryPi.conf  
echo default\_fps\_h264\_2592x1944 = 15  | tee -a RaspberryPi.conf  
echo default\_image\_quality = 5  | tee -a RaspberryPi.conf  
echo h264 = rtsp  | tee -a RaspberryPi.conf***

After creating this configuration file, you will need to restart your Surveillance Station software on your nas to let it pick this configuration file up, do this on your web gui, in the package manager, and restart your Surveillance Station package (stop start).  
  
Now you can start defining your camera in Surveillance Station.  
Open Surveillance Station, go over to IP Camera, add Camera, do quick setup and define your Camera as per below screenshot:

[](https://2.bp.blogspot.com/-4N5RLLk6C80/VdBbzEyeINI/AAAAAAAAJA4/8I3v7lUBwgk/s1600/surveillance+station+ip+camer+add.PNG)

Don't worry about the username and password, confirm to finish.  
  
you will notice after adding the camera that it's disconnected, this is because the Surveillance Station software is looking at the root mrl (without the unicast string behind it).

[https://3.bp.blogspot.com/-fiGM4RRJDow/VdBfVPmMFqI/AAAAAAAAJBE/PIzaXHaOsQw/s320/disconnected.PNG](https://3.bp.blogspot.com/-fiGM4RRJDow/VdBfVPmMFqI/AAAAAAAAJBE/PIzaXHaOsQw/s1600/disconnected.PNG)

It looks at:  
rtsp://ip-address-of-your-raspberry:8554  
and not at:  
rtsp://ip-address-of-your-raspberry:8554/unicast  
where it needs to look, hence the disconnection.  
  
Currently I only have a work around for this, if someone has another option, please post a message on my blog below.  
  
In Surveillance Station, select your Camera, then click Configuration, Export, give your export a name, and select a destination (in my case web folder) add your Raspberry Pi Camera and finish the export, now delete your camera from Surveillance Station.  
  
Now edit a configuration file in the Camera Export folder which you have just exported:

***vi /volume1/web/SSCamExport\_RaspberryPi/.ExpCam***

Change the follow 2 parameters from:  
  
path = '/'  
live\_path = '/'  
to   
path = 'unicast'  
live\_path = 'unicast'  
  
Save and re-import your Camera, it should be working now.  
Note that, if you make configuration changes to your Camera in Surveillance Station, that u need to redo this procedure.  
  
I hope this write up will be useful for people wanting to experiment with the Raspberry Pi Camera and Surveillance Station, if u like the write up, have any comments, suggestions, or improvements, leave me a message at the bottom of this blog.

### [HOWTO - Create a Raspberry Pi Infrared remote control](https://random-notes-of-a-sysadmin.blogspot.tw/2015/12/howto-create-raspberry-pi-infrared.html)

If you have the need to control all your home equipment which has a remote available, look no further, the Raspberry Pi is your ultimate low-cost universal remote control tool available.

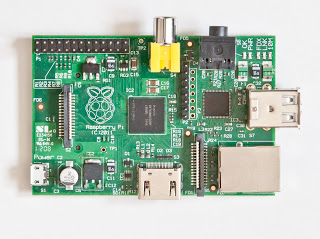
|  |
| --- |
| [https://2.bp.blogspot.com/-2KdXELhM1X0/Vn1EH7aTtaI/AAAAAAAAJys/3OagzqR2wzo/s640/20151225_141243.jpg](https://2.bp.blogspot.com/-2KdXELhM1X0/Vn1EH7aTtaI/AAAAAAAAJys/3OagzqR2wzo/s1600/20151225_141243.jpg) |
| My Raspberry Pi Infrared Remote Control Appliance |

When I have some time to spare, you can find me at home messing around with Linux, Raspberry Pi's and some home automation domotics.  
Recently I was looking to extend my home domotics system to be able to switch every device which has a remote control in my house.    
When I started searching for a solution, I was suprised how easily it was to build a solution consisting of a Raspberry Pi and some locally acquired electronic components.  
  
Before I start explaining how to build the solution, let me first give u an overview of all the components which I used to build my remote.

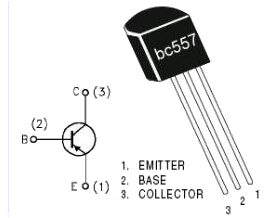
## Putting the Hardware together.

### Component List:

1x Raspberry Pi, model B+ (look on second hand market, or internet, can be acquired for as low as 20€)

[](https://2.bp.blogspot.com/-GT5lVqxTDGo/Vn1MuvxlT3I/AAAAAAAAJy4/emcNiBSf3SU/s1600/Front_of_Raspberry_Pi.jpg)

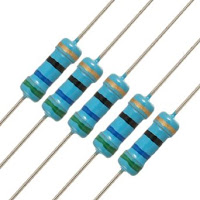
1 x Transistor BC557C TO92 KEC  
Conrad article: 155959  
Price: 0,1 € / piece

[](https://4.bp.blogspot.com/-cQ_A4mP7PsY/Vn1NKEaESzI/AAAAAAAAJzA/TlWFtBJh0a8/s1600/bc557.gif)

4 x IR-LED 5mm HE3-290AC  
Conrad article: 181713  
Price: 0,35 € / piece  
[](https://2.bp.blogspot.com/-QSqDiONTzVM/Vn1OC2EDOcI/AAAAAAAAJzI/LDA2iNfyH_8/s1600/Harvatek-HE3-290AC.jpg)  
1 x IR Receiver 2,7-5,5v OS-0038G  
Conrad article: 184296  
Price: 0,67 € / piece

[](https://2.bp.blogspot.com/-I_OzHQcbZbA/Vn1OZD_e-SI/AAAAAAAAJzQ/ip5k1Y_yXu0/s1600/184281_LB_00_FB.EPS_1000.jpg)

100 x Resistor 56 Ohm 1/4w (you only need 4)  
Conrad article: 408018 - 89  
Price: 3 € / 100 pcs

[](https://3.bp.blogspot.com/-q9_i5uIBJVs/Vn1OyE8xTMI/AAAAAAAAJzY/EOJL_VGLFk0/s1600/5133AefitML._SX342_.jpg)

### The Schematics:

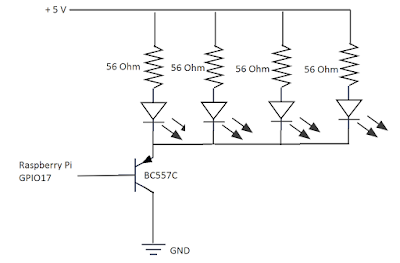
To make it able to send and receive with our home built remote control, we  have to connect two circuits to our Raspberry Pi, one send circuit and one receive circuit.  
The send circuit consists of a few parts while the receiver circuit (the ir receiver) can be soldered directly onto the raspberry.

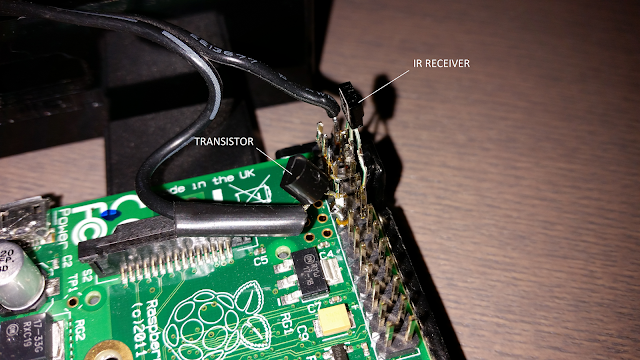
#### The send circuit:

Since there are not too many components in the send circuit, no separate print plate is necessary to solder the components on.

I took my Raspberry Pi housing, drilled 4 holes in it to fit my IR leds, glued these to the casing with epoxy, soldered a resistor on each led's anode, then lead two wires to my Raspberry Pi board where the leds cathode wire was soldered to the emitter of my transistor and the anode wire directly to pin 2 on the Raspberry Pi board (+5v).

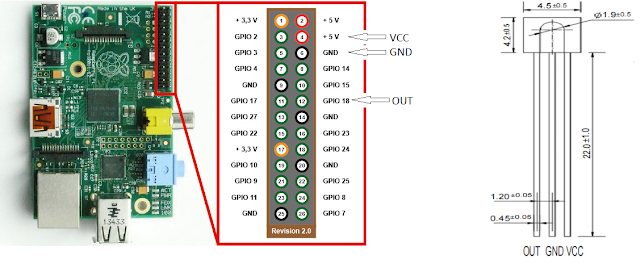
I had soldered the collector of my transistor directly to pin 9 (GND) and the Base of my transistor to pin 11 (GPIO17)

[](https://1.bp.blogspot.com/-3t1XbDkMCBM/Vn1TnKPO3RI/AAAAAAAAJzo/bPQX9CJlHeA/s1600/ir+blaster+schematic.png)

[](https://2.bp.blogspot.com/-dl9T5wLETUw/Vn1cPuwkz7I/AAAAAAAAJ0E/E8cxKF8KhTY/s1600/circuit+picture.png)

#### The receive circuit:

The IR receiver can be soldered directly onto the Raspberry Pi's pins 4 (+5v) - 5 (GND) - and 12 (GPIO18) respectively.

[](https://2.bp.blogspot.com/-KjOkBNvMs7w/Vn1WoLF_DcI/AAAAAAAAJz0/WDRX6-UTHaY/s1600/ir+receiver+schematic.png)

## Configuring the Software.

#### [https://1.bp.blogspot.com/-wVXGFALCqiY/Vn2hIK3w2iI/AAAAAAAAJ0U/dzhea8QhSes/s200/raspbian.png](https://1.bp.blogspot.com/-wVXGFALCqiY/Vn2hIK3w2iI/AAAAAAAAJ0U/dzhea8QhSes/s1600/raspbian.png)The Raspberry Pi operating system.

Take the latest Raspbian version, as of writing, this is Raspbian Jessie from November 2015, download it from the Raspberry foundation website, put it to SD card and start off.

#### The Infrared software driver.

The linux platform (Raspbian in this case) already has packaged support for lirc - infrared control software for linux.  
To interface this software with the hardware on the Raspberry Pi, a driver was written called lirc\_rpi.    
This driver is provided as an overlay kernel snapshot file which needs to be enabled in the configuration of the Raspberry Pi.  Once enabled it loads when the Raspberry Pi boots up, no further loading via modules or modprobe is necessary.  
  
To enable the lirc\_rpi driver, we need to make a modification in the /boot/config.txt file  
  
In the config file there is an option called: #dtoverlay=lirc-rpi  
  
It needs to be uncommented and a small addition needs to be made to invert the IO logic to make this work properly.  The config file is on a read only partition, so we first need to remount it read/write to edit the config file.

* sudo mount -o remount rw /boot
* sudo nano /boot/config.txt
* remove: #dtoverlay=lirc-rpi
* add: dtoverlay=lirc-rpi,invert=on
* Ctrl-x to save

Reboot your Raspberry Pi, it will load the lirc-rpi driver on boot now.

* sudo reboot

#### The infrared control software.

As stated previously, the linux platform (Raspbian in this case) already has packaged support for lirc - infrared control software for linux, setting this up in Raspbian is just installing a package.

* sudo apt-get -y install lirc

This will set up the required binaries and services for our remote control.  
For a detailed description and manuals about this software, you can visit the lirc project website <http://www.lirc.org/>  
  
configure your lirc hardware config file to reflect below configuration:  
  
# /etc/lirc/hardware.conf  
#  
# Arguments which will be used when launching lircd  
LIRCD\_ARGS="--uinput"  
  
#Don't start lircmd even if there seems to be a good config file  
#START\_LIRCMD=false  
  
#Don't start irexec, even if a good config file seems to exist.  
#START\_IREXEC=false  
  
#Try to load appropriate kernel modules  
LOAD\_MODULES=true  
  
# Run "lircd --driver=help" for a list of supported drivers.  
DRIVER="DEFAULT"  
# usually /dev/lirc0 is the correct setting for systems using udev  
DEVICE="/dev/lirc0"  
MODULES="lirc\_rpi"  
  
# Default configuration files for your hardware if any  
LIRCD\_CONF=""  
LIRCMD\_CONF=""  
  
  
The binaries that we will be using for our remote control are:

* mode2 -> it can visualise ir reception on our ir sensor
* irrecord -> this binary can sample keys from existing remote controls and record them to a config file
* irsend -> send out ir control signal

Once the software is installed, a daemon called lircd will be running, it reads it's configuration from /etc/lirc/lircd.conf - this file contains our remote control defenitions that we can send out amongst some other stuff.

To work manually with the above mentioned binaries we first need to stop the lircd daemon because it will interfere with our work otherwise:

* service lircd stop

To make sure our ir receiver is working properly, we will perform a small test, enter below command and point a remote control to your ir sensor and press a button, it should display remote control codes:

* mode2 -d /dev/lirc0

Once verified that our ir sensor is working, we can now start recording existing remote control buttons, launch below command and follow instructions, it will record a configuration file which we can feed to our ir daemon afterwards.

* irrecord -d /dev/lirc0 ~/lircd.conf

Once recorded, copy your remote control file to /etc/lirc/lircd.conf and start your lircd service:

* service lircd start

Once that the daemon is running with the recorded remote control configuration, you should be able to send out IR remote control codes, U can do a test by using below example.

* irsend -# 2 -d /var/run/lirc/lircd SEND\_ONCE telenet KEY\_OK (read irsend manpage)

Now that we have configured the Infrared hardware and software, it's time to make it a bit more user friendly by installing a website for all this so we can easily use the remote control from any device which has a web browser.  
There is a nice project on Github written by Alex Bain which just does all this:  
<https://github.com/alexbain/lirc_web>  
  
As a prerequisite you will have to install node.js webserver on your Raspberry Pi  
  
Install Node.js  
Download Node.js source  
  
Raspberry Pi Model A, B, B+ and Compute Module  
wget https://nodejs.org/dist/v4.0.0/node-v4.0.0-linux-armv6l.tar.gz  
tar -xvf node-v4.0.0-linux-armv6l.tar.gz  
cd node-v4.0.0-linux-armv6l  
  
Raspberry Pi 2 Model B  
wget https://nodejs.org/dist/v4.0.0/node-v4.0.0-linux-armv7l.tar.gz  
tar -xvf node-v4.0.0-linux-armv7l.tar.gz  
cd node-v4.0.0-linux-armv7l  
Copy to /usr/local  
  
sudo cp -R \* /usr/local/  
  
Verify the version of your node installation with:  
node -v  
  
  
cd ~/  
sudo npm install -g lirc\_web  
  
Test the website by running it manually:  
lirc\_web  
  
sudo apt-get install -y nginx  
curl https://raw.githubusercontent.com/alexbain/lirc\_web/master/example\_configs/nginx/nginx.conf | sudo tee /etc/nginx/sites-available/default  
  
sudo service nginx restart

 [Tutorial] RTSP Raspberry Pi B+ Viewer [6-Cam] [4-Cam]

**Video Tutorial**

**UPDATED (12/4/2016)**

**RTSP Screen Calculator Attached Below**

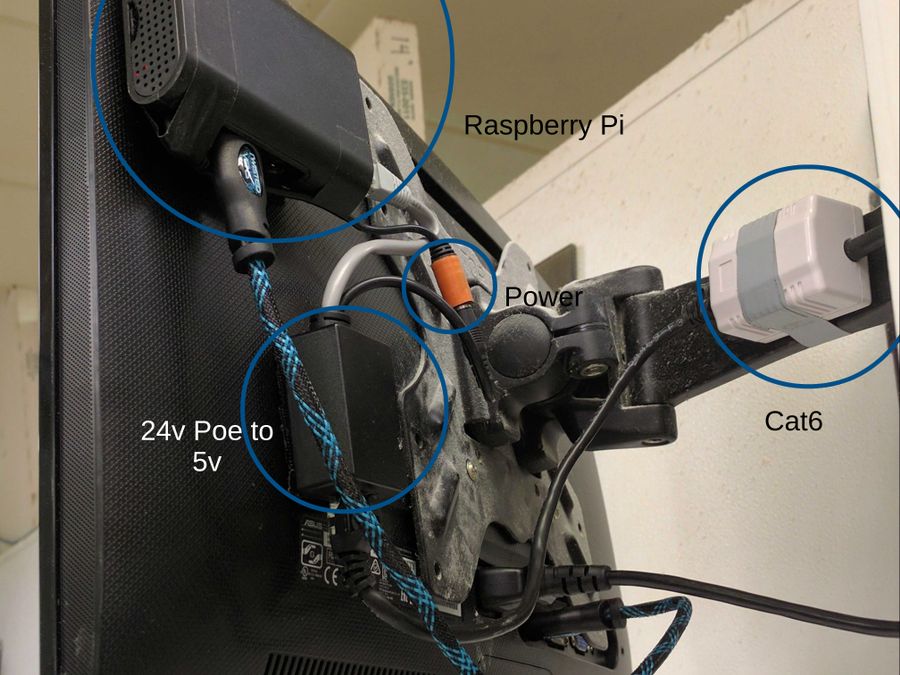
*I had to enclose it into a zip file in order to upload it.*

**Why?**

This Tutorial will hopefully help anyone looking to create a cheap RTSP viewer for themselves or clients.  I had to create this for a resturant client that wanted to see certain points of his resturant while he was rolling out pizza dough.

**Main Items**

* Raspberry Pi 3 [Yummy Pi](https://www.amazon.com/Raspberry-Pi-RASP-PI-3-Model-Motherboard/dp/B01CD5VC92/ref=sr_1_3?s=pc&ie=UTF8&qid=1474148564&sr=1-3&keywords=raspberry+pi+3)
* Wifi Texas 24v to 5v Dongle (To Power Raspberry Pi over network) [24v Poe to 5v & Network](https://www.amazon.com/WS-POE-12v-10w-Extender-Ethernet-Wanscam-Injectors/dp/B00EBCQ5FM/ref=sr_1_2?ie=UTF8&qid=1474148283&sr=8-2&keywords=wifi+texas+5v)
* Wifi Texas 5v Micro USB Adapters <https://find-a-poe.com/product/1-35mm-ra-4x/>
* 1' HDMI Cable Pack [Pack of Cables](https://www.amazon.com/Twisted-Veins-Adapter-Supports-Ethernet/dp/B00FX6KO8Y/ref=sr_1_1?ie=UTF8&qid=1474148471&sr=8-1&keywords=1ft+hdmi)



----------------------------------------------------------------------------------------------------------------------------

**1. Download the latest Raspbian img**

[**https://www.raspberrypi.org/downloads/raspbian/**](https://www.raspberrypi.org/downloads/raspbian/)

2. **Download win32diskimager**

[**https://sourceforge.net/projects/win32diskimager/**](https://sourceforge.net/projects/win32diskimager/)

3. **After writing the img file to the sd card and booting up the pi, go to terminal window.**

4. **Once in a terminal window type**

**sudo raspi-config**

**5. Select these options:**

(8) (Advanced Options)

- Then select (A3) “split memory” and type *256*(Then go back to main menu)

(4) (Internationalism Options)

- Then change / check date and time to be accurate (Then go back to main menu)

(3) (Enable Boot……)

- Then select “Console Text Console”  pick auto login (Then go back…)

(1) (Expand Filesystem)

**-  Confirm to expand file system, then reboot your pi**

**6. Now that you have booted into console, update your pi (This will take awhile)**

**sudo apt-get update && sudo apt-get upgrade -y && sudo reboot**

**7. After your pi reboots, omxplayer and screen need to be installed.  Omxplayer is the program that will display the feeds.  However, omxplayer can only display one stream per console session.  To overcome this we use screen to run omxplayer in “windowed mode”**

**sudo apt-get install omxplayer -y && sudo apt-get install screen && sudo reboot**

**8. After your pi reboots, we want to change its ip address.  Open the network configuration file with this command**

sudo nano /etc/network/interfaces

**9. By default the pi is configured to operate in DHCP mode.  In order for this script to work correctly, I have found setting up the pi in Static mode works better.**

**The pi is configured by default as:**

*auto lo*

*iface lo inet loopback*

*iface etho inet manual*

Delete “ *iface etho inet manual “*

Now set we are going to set the address to 192.168.1.77 (You may need to change this depending on your network)

Blue text stays the same.  Red is modified text.\*

*auto lo*

*iface lo inet loopback*

*auto eth0*

*iface eth0 inet static*

*address 192.168.1.77*

*netmask 255.255.255.0*

*gateway 192.168.1.1*

*dns-nameservers 192.168.1.1*

*network 192.168.1.0*

*broadcast 192.168.1.255*

**10. Now if you reboot your pi, you will have a problem.  You will have not only a static address but also a DHCP address.  We need to disable DHCP all together.  To accomplish this we need to edit a file.  The command below will open that file.**

sudo nano /etc/dhcpcd.conf

**11. Add the line of code below to the bottom of the file we just opened.  This will now disable DHCP for good and you will have one Static IP address.**

denyinterfaces eth0

**12. To exit the file press [ctrl + x] then press [y] followed by [enter] to confirm.  Now reboot**

**Your pi.**

sudo reboot

**13. Next go to your desktop and download putty.  This program will allow us to remotely**

**manage our pi and allow us to copy and paste, saving lots of time.**

[**http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html**](http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

**14. Now open putty.  Type the ip address you gave your pi in the ip address bar.  Now**

**click “open”.  If everything went correctly a shell window should open asking for your**

**Username.**

**Raspberry Pi Default Login;**

**Username = pi**

**Password = raspberry**

**15. Now we need to create the file that the script will sit in (UPDATED).**

sudo nano /etc/init.d/displaycameras

**16. The script below is for a 2x2 camera matrix. At the bottom of the document there are**

**More scripts for more cameras and different configurations.  I recommend experimenting**

**with this script first, then moving into more complex screen mapping. Before copying this**

**Make sure your pi is on a wired connection, NOT WIFI. Next go to unifi video an enable**

**“Medium RTSP Stream” for your camera.  Set the FPS slider to 30 and the Bitrate under**

**700 kbps to start.  Now that those are set you are ready to continue.**

**(To copy this script Into putty, highlight the text the press [ctrl + c] on your keyboard to copy to your clipboard. Then right click in putty and it will paste)**

**(Replace Red text with your RTSP url)**

#!/bin/bash

### BEGIN INIT INFO

# Provides: omxplayer

# Required-Start:

# Required-Stop:

# Default-Start: 2 3 4 5

# Default-Stop: 0 1 6

# Short-Description: Displays camera feeds for monitoring

# Description:

### END INIT INFO

# Camera Feeds & Positions

top\_left="screen -dmS top\_left sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"0 0 960 540\" rtsp:// <...cam1\_url...> --live -n -1'";

top\_right="screen -dmS top\_right sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"960 0 1920 540\" rtsp:// <...cam2\_url...> --live -n -1'";

bottom\_left="screen -dmS bottom\_left sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"0 540 960 1080\" rtsp:// <...cam3\_url...> --live -n -1'";

bottom\_right="screen -dmS bottom\_right sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"960 540 1920 1080\" rtsp:// <...cam4\_url...> --live -n -1'";

# Camera Feed Names

# (variable names from above, separated by a space)

camera\_feeds=(top\_left top\_right bottom\_left bottom\_right)

#---- There should be no need to edit anything below this line ----

# Start displaying camera feeds

case "$1" in

start)

for i in "${camera\_feeds[@]}"

do

eval eval '$'$i

done

echo "Camera Display Started"

;;

# Stop displaying camera feeds

stop)

sudo killall omxplayer.bin

echo "Camera Display Ended"

;;

# Restart any camera feeds that have died

repair)

for i in "${camera\_feeds[@]}"

do

if !(sudo screen -list | grep -q $i)

then

eval eval '$'$i

echo "$i is now running"

fi

done

;;

\*)

echo "Usage: /etc/init.d/displaycameras {start|stop|repair}"

exit 1

;;

esac

**17. Before this script can be run at startup, the pi needs permissions to access the file and**

**Run it.**

**sudo chmod 755 /etc/init.d/displaycameras**

**18. Now the script can be added to startup**

sudo update-rc.d displaycameras defaults

**19.  Now we have a script setup to pull all of our RTSP streams up on boot.  There is a slight**

**problem, omxplayer will quite sometimes and or unifi video will drop the streams. To solve this issue we add a cron job to run the repair portion of the script to restart camera feeds that has dropped.**

sudo crontab -e

**20. If it asked to choose from 3 different ways to open crontab, select option 2.  Then insert**

**this at the bottom of the crontab. This will run the script “/etc/init.d/displaycameras repair” every 5 min.  To have it check every 1 min, simply change the 5 to a 1.**

**\*/5 \* \* \* \* /etc/init.d/displaycameras repair**

**21. Now you should be able to reboot you pi and have four camer feeds working! The great thing is this setup without a monitor should cast $50 - $85.  And the system is professional not bunched together with a ton of parts that were never ment to work together in that way like a chromecast or something of that nature.  It is also cheaper and more reliable than having a windows machine mounting on the back of a monitor.**

**The next steps are for creating a 6 Cam matrix viewer and setting up a series of cron jobs to turn the monitor off to prevent burn in.**

**---------------------------------------------6 Cam Matrix--------------------------------------------------**

**22. Start off by opening the camera display file.**

**sudo nano /etc/init.d/displaycameras**

**23. Now delete everything in the file and replace with this 6 Cam Matrix Code.  Replace the red text with your address.**

**#!/bin/bash**

**### BEGIN INIT INFO**

**# Provides: omxplayer**

**# Required-Start:**

**# Required-Stop:**

**# Default-Start: 2 3 4 5**

**# Default-Stop: 0 1 6**

**# Short-Description: Displays camera feeds for monitoring**

**# Description:**

**### END INIT INFO**

**# Camera Feeds & Positions**

**#First Colmn**

**large\_left="screen -dmS large\_left sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"0 0 1280 720\" rtsp://192.168.1.168:7447/56fc759be4b01f1769f35f7f\_1 --live -n -1'";**

**mid\_lcenter="screen -dmS mid\_lcenter sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"640 720 1280 1080\" rtsp://192.168.1.168:7447/56fc791fe4b01f1769f35fad\_1 --live -n -1'";**

**bottom\_left="screen -dmS bottom\_left sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"0 720 640 1080\" rtsp://192.168.1.168:7447/56fc791fe4b01f1769f35fae\_1 --live -n -1'";**

**#Second Colmn**

**top\_right="screen -dmS top\_right sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"1280 0 1920 360\" rtsp://192.168.1.168:7447/56fc795be4b01f1769f35faf\_1 --live -n -1'";**

**mid\_rcenter="screen -dmS mid\_rcenter sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"1280 360 1920 720\" rtsp://192.168.1.168:7447/56de2e1888117cf574015d71\_1 --live -n -1'";**

**bottom\_right="screen -dmS bottom\_right sh -c 'omxplayer --avdict rtsp\_transport:tcp --win \"1280 720 1920 1080\" rtsp://192.168.1.168:7447/56de2e1888117cf574015d71\_1 --live -n -1'";**

**# Camera Feed Names**

**# (variable names from above, separated by a space)**

**camera\_feeds=(large\_left mid\_lcenter bottom\_left top\_right mid\_rcenter bottom\_right)**

**#---- There should be no need to edit anything below this line ----**

**# Start displaying camera feeds**

**case "$1" in**

**start)**

**for i in "${camera\_feeds[@]}"**

**do**

**eval eval '$'$i**

**done**

**echo "Camera Display Started"**

**;;**

**# Stop displaying camera feeds**

**stop)**

**sudo killall omxplayer.bin**

**echo "Camera Display Ended"**

**;;**

**# Restart any camera feeds that have died**

**repair)**

**for i in "${camera\_feeds[@]}"**

**do**

**if !(sudo screen -list | grep -q $i)**

**then**

**eval eval '$'$i**

**echo "$i is now running"**

**fi**

**done**

**;;**

**\*)**

**echo "Usage: /etc/init.d/displaycameras {start|stop|repair}"**

**exit 1**

**;;**

**esac**

**24. Save and exit the display camera file and now lets go back into the cron tab file.**

**sudo crontab -e**

**25. Delete the one line of code at the end and replace with this.**

**Line One : stops the camera streams (12:00 am)**

**Line Two : Turns off the video output to the monitor (12:02 am)**

**Line Three : Turns the output to the monitor back on (6:00 am)**

**Line four : Starts the camera script again (6:02 am)**

**Line five : Reboots the Pi once a week at 6:00 am**

**Line six : Runs the repair script ever minute between 6 am and 11 pm**

**0 0 \* \* \* /etc/init.d/displaycameras stop**

**2 0 \* \* \* /opt/vc/bin/tvservice -o**

**0 6 \* \* \* /opt/vc/bin/tvservice -p**

**2 6 \* \* \* /etc/init.d/displaycameras start**

**0 6 \* \* 1 sudo reboot**

**\*/1 6-23 \* \* \* /etc/init.d/displaycameras repair**

**----------------------------------------------------------------------------------------------------------------------------**

**Resources:**

[https://community.spiceworks.com/how\_to/123787-raspberry-pi-powered-surveillance-camera-monitoring-d...](https://community.spiceworks.com/how_to/123787-raspberry-pi-powered-surveillance-camera-monitoring-display)

<https://www.bitpi.co/2015/03/18/raspberry-pi-surveillance-monitor/>

Attachment

RTSP Screen Calculator.zip

[https://ubnt.i.lithium.com/skins/images/7165FD62CB6DE87AD06FB0D297DB6D6F/castlerockui/images/icon_download.png](https://community.ubnt.com/ubnt/attachments/ubnt/airVision_board/54835/2/RTSP%20Screen%20Calculator.zip)

**樹莓派, 手機, 電腦與 Raspberry Pi Camera Module 的串流視訊建立**

*本篇網頁用到的 Camera 模組套件可至露天賣場訂購：*

* [*CameraPi - Raspberry Pi + Camera Board (樹莓派 + 官方攝像機)*](http://goods.ruten.com.tw/item/show?21307201184919)

|  |
| --- |
| <http://4.bp.blogspot.com/-hcpse8eSTvU/UvwKPxgie2I/AAAAAAAADeY/9xJtr2TowwI/s1600/RasPi-wi-CAM.png> |
| 正確組裝完成圖 |

*NOTE：攝像機非常不喜歡靜電！要從防靜電袋中拿出 Camera Board 組裝前，一定要記得先將自己放電先 ( 例如：摸一下水龍頭或身體與地表接觸 ) !*  
*( Please note that the camera can be damaged by static electricity. Before removing the camera from its grey anti-static bag, please make sure you have discharged yourself by touching an earthed object (e.g. a radiator or water tap). )*  
  
這篇網頁不說明 Raspberry Pi Camera Module ( 網頁中簡稱 (官方) 攝像機 ) 軟體安裝和裝設的方法，如果還未完成這部分的話，請先到官方網站去看攝像機安裝說明影片影像和啟用說明。  
  
  
  
下面列出官方攝像機網站與使用說明書的連結網址：

* [Raspberry Pi Camera](http://www.raspberrypi.org/camera) ( Web, 英文 )

官方網站關於攝像機模組的硬體安裝與軟體設置的影片與文字說明，主要主題有：

* + 如何安裝攝像機

裡面有影片可以看。

* + 如何在 Raspbian 作業系統啟用攝像機的支援  
    最近 ( wheezy-raspbian-2013-02-09 之後 ) 的作業系統裡面已經有選項可以選用，因此只要進入 ***raspi-config*** 選單中 "**Enable Camera**" 就可以了，網頁中有選單結圖可以做參考。  
      
    若您的作業系統是舊版的，就請先更新與升級作業系統，之後再進入選單就會有啟用攝像機的選項出現了。
  + 如何使用樹莓派的攝像機軟體

***raspivid***、***raspistill*** 這兩個攝像機命令列指令 (前者是攝影指令；後者是照相紙令) 的使用範例，但沒有詳細的指令參數說明，這兩個指令的參數說明請尤其他提供的連結下載資料。

* + 如何串流傳送攝像機影片到網路上

官網寫的一大堆指令範例要做串流，又要安裝軟體的，如果各位看得懂，就看官網中的，不然就看這網頁中的視訊串流建立的介紹說明。

* [RaspiCam Documentation (odt](https://github.com/raspberrypi/userland/blob/master/host_applications/linux/apps/raspicam/RaspiCamDocs.odt), [PDF](http://goo.gl/xyKt7Z), December 7th 2013)

官方攝像機最新說明文件 ( odt 是原始檔案；PDF 是 odt 的轉檔 )  
但這份文件主要著重在軟體指令的說明，重點是在第五頁之後的 **raspistill**, **raspivid**, **raspiyuv** 三個命令列指令的相關參數的說明，還有第十六頁的指令使用範例，都比上、下兩個文件清楚。

* [Unofficial guide to getting up and running with the Raspberry Pi Camera (PDF, 英文)](http://www.farnell.com/datasheets/1722614.pdf)

這份是針對舊版的 wheezy-raspbian 作業系統設置官方攝像機的軟體應設定說明，雖然新版的用不到所有的東西，但是也是有一些東西可供參考

* [OV5647 5-megapixel product brief - OmniVision ( PDF, 英文 )](http://www.electronicsdatasheets.com/download/5195f320e34e247728000001.pdf?format=pdf)

官方攝像機 Image Sensor 的簡介

如果您想學影像處理，或是對硬體控制有興趣，使用樹莓派以及專用攝像機的組合是一個可作為起步的套件組合。接著，我們就進入此篇網頁的重點：樹莓派官方攝像機怎麼串流視訊資料到：

1. 樹莓派主機  -- 使用 PiVision

PiVision 是一個使用 GUI 來調整官方攝像機在攝影、照相時的參數的圖型介面程式。

1. Android 手機、平板 兩個可觀看樹莓派攝像機的 APP，必須要先在手機做安裝。
   * 使用 RaspiCAM Remote

**強力推薦安裝**這個樹莓派官方攝像機  APP，可直接拍照或是錄影存到手機上，重點是：不需要在樹莓派主機安裝任何程式就可使用手機操作，讚吧！！

* + 使用 MX Player

MX Player 是個撥放影片的 Android APP，但是它可以用來撥放串流檔案，也就是要用它來接收由樹莓派傳送出來的攝像機串流影片。

1. Windows(8) 作業系統

Windows作業系統 ( 我使用 Win8 ) 是作為串流影片的接收端，但是不只一個地方可以做串流影片的接收，在這部分我會展示一次使用兩個軟體來做。

新版的 wheezy-raspbian 裡已經預先安裝了官方攝像機的相關驅動與軟體，所以如果你(妳)使用的作業系統是舊的，先檢查一下 **/opt/vc/bin** 下有無 ***raspivid*** 和 ***raspistill*** 兩個執行檔，確認是否之前已經對系統做過更新與升級到支援官方攝像機；沒有的話就參照上方所提供的檔案一步一步操作吧。沒完成之前不要往下看！

**\*\* 測試環境說明 \*\***  
  
在進行之前，說明一下樹莓派的周邊與系統簡單設定。在網頁中所出現的資料，都是在這條件上完成的，若是你(妳)的設置不一樣有可能會造成不一樣的結果。

* 樹莓派作業系統：**2014-01-07-Wheezy-Rraspbian** ( 全新安裝 )  
  無超頻與變更記憶體分割，延伸磁區、開啟 SSH、設置時區與鍵盤系統，相關基本作業系統環境設置可參考 "[[PiFace Digital 首部曲] 樹莓派作業系統 (2013-02-09-wheezy-raspbian) 之環境設定](http://ruten-proteus.blogspot.com/2013/02/piface-digital-2012-12-16-wheezy.html)"
* 攝像機：Raspberry Pi Camera Module ( 樹莓派攝像機 )
* 記憶卡：8G MicroSD ( + 轉卡 ), Class 4 ( 賣場套件使用的卡 )
* 鍵盤滑鼠：[ione SCORPIUS P6](http://www.ione-usa.com/ione-scorpius-p6-slim-panel-mount-touchpad-keyboard.html) ( USB 介面, 鍵盤滑鼠一體整合 )
* 螢幕：ViewSonic VA2037m-LED  
  樹莓派使用 HDMI 轉 VGA 轉接頭轉接到螢幕
* 電源：[樹莓派電源組合](http://goods.ruten.com.tw/item/show?21211167147160) ( USB (5V, 1000 mA) + MicroUSB 轉接線 )

電流輸出至少 1000mA，並依照外接設備增加電流輸出量。

* 網路：
  + 路由器：[Baffalo WHR-G300N-V2](http://www.buffalo-tech.com.tw/product.php?T_Id=1&P_Id=155#content)  
    無線網路運作在 300Mbps 模式  
    有線網路 100Mbps 模式
  + 筆電無線網路卡：[Intel® Centrino® Wireless-N 2230](http://www.intel.com/content/www/us/en/wireless-products/centrino-wireless-n-2230.html)

最大速度 300Mbps

* + 樹莓派使用有線網路連接到路由器
* 其他
  + PC 作業系統： Windows 8
  + 手機：Android, Sony Xperia P, LT22i

**軟體安裝：**  
在這篇網頁中，需要用到幾個軟體，包括樹莓派、Android 手機或平板和 Windows 作業系統都各有軟體需要下載或安裝，裝不裝可以在看過全文之後再決定也可以。  
  
  
**\*\*樹莓派官方攝像機啟用與 Pi Vision 安裝：**   
  
*樹莓派的官方攝像機在使用時有預覽的功能 ( Preview )，但一定要接機子上面的視訊輸出才會正常動作，不然是看不到預覽的影像的 (使用 VNC 連過去也是看不到的)。所以在繼續之前****請****接上 HDMI線，或 HDMI 轉 VGA 轉接器，或接 AV Out 到螢幕上。*  
  
作業系統：2014-01-07-Wheezy-Raspbian 全新安裝 ( 預設狀態，不修改任何 ***raspi-config*** 下任何系統參數)，但必須 "**Enable Camera**" 和啟用 "**8 Advanced Option / A4 SSH**" 這兩個選項，其餘的預設就好 ( 或自己再去設定，不過建議完成所有測試之後再做，避免其他因素影響測試結果 )。  
  
*ps. "****8 Advanced Option / A3 Memory Split****" 已預設 GPU 分割為 128MB，除非你(妳)知道自己在做什麼，不然先不要改。*

|  |
| --- |
| <http://3.bp.blogspot.com/-R7MIBcnVTjU/UvxWALNofJI/AAAAAAAADfA/e5saUgNEsTQ/s1600/2014-01-07-raspi-config-enable-camera.jpg> |
| 2014-01-07-wheezy-raspbian, raspi-config, "Enable Camera" |

|  |
| --- |
| <http://3.bp.blogspot.com/-WEprb2hEYLY/UvxWfl3byoI/AAAAAAAADfI/PtiRbSYDiMA/s1600/2014-01-07-raspi-config-SSH.jpg> |
| 2014-01-07-wheezy-raspbian, raspi-config, 啟用 "A4 SSH" |

在樹莓派的桌面上打開瀏覽器輸入 Pi Vision 網址，在網站中點擊 **Downalod**下載 Pi Vision 的壓縮檔 ( **PiVision\_RPi.tar.gz** ) 直接存到 ***/home/pi*** 目錄下  
  
[Pi Vision](http://heywhatsthebigidea.net/projects/pi-vision-a-raspberry-pi-camera-controller/) 網站：[http://wp.me/P2IUMh-gc](http://heywhatsthebigidea.net/projects/pi-vision-a-raspberry-pi-camera-controller/)

|  |
| --- |
| <http://2.bp.blogspot.com/-IZuk8ojNAYM/UvxiJ9uIanI/AAAAAAAADfc/FxU72axmMHg/s1600/Pi-Vision-download.jpg> |
| 到 Pi Vision 官方網站下載檔案 |

接著繼續在**桌面**打開 **LXTerminal** ( 終端機 ) 輸入下面指令解開下載的壓縮檔。解開的壓縮檔是一個執行檔，但權限不對需要修改為可執行的檔案

pi@raspberrypi ~ $ tar -zxvf PiVision\_RPi.tar.gz

PiVision\_RPi/rpiCC

PiVision\_RPi/

pi@raspberrypi ~ $ cd PiVision\_RPi/

pi@raspberrypi ~/PiVision\_RPi $ ls -l

total 17084

-rw-r--r-- 1 pi pi 17490013 May 30 2013 rpiCC

pi@raspberrypi ~/PiVision\_RPi $ file rpiCC

rpiCC: ELF 32-bit LSB executable, ARM, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.0.0, not stripped

pi@raspberrypi ~/PiVision\_RPi $ ls -l

total 17084

-rw-r--r-- 1 pi pi 17490013 May 30 2013 rpiCC

pi@raspberrypi ~/PiVision\_RPi $ chmod 755 rpiCC

pi@raspberrypi ~/PiVision\_RPi $ ls -l

total 17084

-rwxr-xr-x 1 pi pi 17490013 May 30 2013 rpiCC

pi@raspberrypi ~/PiVision\_RPi $

完成上面的指令輸入之後，在目錄 ***~/PiVision/*** 就可得到 **rpiCC** 可執行檔。想研究原始碼的話，就再繼續輸入下面指令下載原始碼

pi@raspberrypi ~/PiVision\_RPi $ cd

pi@raspberrypi ~ $ git clone git://github.com/local-vision/Pi-Vision

Cloning into 'Pi-Vision'...

remote: Reusing existing pack: 213, done.

remote: Total 213 (delta 0), reused 0 (delta 0)

Receiving objects: 100% (213/213), 30.52 MiB | 2.91 MiB/s, done.

Resolving deltas: 100% (101/101), done.

Checking out files: 100% (53/53), done.

pi@raspberrypi ~ $ ls -l

drwxr-xr-x 2 pi pi 4096 Jan 8 06:09 Desktop

-rw-r--r-- 1 pi pi 5781 Feb 3 2013 ocr\_pi.png

drwxr-xr-x 6 pi pi 4096 Feb 12 13:16 Pi-Vision

drwxr-xr-x 2 pi pi 4096 Feb 12 14:22 PiVision\_RPi

drwxrwxr-x 2 pi pi 4096 Mar 10 2013 python\_games

drwxr-xr-x 2 pi pi 4096 Feb 12 12:38 temp

pi@raspberrypi ~ $

下載完成之後，就會在家目錄中產生一個 ***Pi-Vision*** 的資料夾，**rpiCC** 的原始碼就在裡面。  
  
好了！進入到 ***PiVision\_RPi*** 目錄中執行 ***rpiCC***，就會在桌面出現 Pi Vision 的圖形介面程式

pi@raspberrypi ~ $ cd PiVision\_RPi/

pi@raspberrypi ~/PiVision\_RPi $ ./rpiCC &

pi@raspberrypi ~/PiVision\_RPi $

Pi Vision GUI 在桌面的中間靠右下的地方，直接按下 "**Test Camera Preview**" 按鈕就會在桌面的左上方看到攝像機現在看到的影像。

|  |
| --- |
| [2014-01-07-Wheezy-Raspbian, Pi Vision GUI](http://2.bp.blogspot.com/-ODggHXHmsno/UvwH_Zo2EiI/AAAAAAAADds/7ejZpoQ_Yzo/s1600/Desktop-PiVision-blog.jpg) |
| 2014-01-07-Wheezy-Raspbian, Pi Vision GUI |

Pi Vision GUI 提供了拍照 ( Photo )、攝影 ( Video ) 和攝像機拍照攝影時的可調參數設定 ( Settings ) 介面選項，在開發或是撰寫影像處理程式時，無非提供了可馬上得知參數單獨或是混合使用時的即時反饋。

|  |
| --- |
| [Pi Vision GUI](http://1.bp.blogspot.com/-qPMbmhKPShE/Uv2eYmiDfAI/AAAAAAAADfs/3McOUomE_8s/s1600/Desktop-PiVision-crop-blog.jpg) |
| Pi Vision GUI |

***Note：****在測試環境中的說明，可以知道樹莓派外接了 USB 整合式鍵盤滑鼠和 HDMI 轉 VGA 轉接頭，因此在預覽很大的影像時，螢幕會突然熄滅再開啟，但是設定小一點時就沒有這問題，因此若是有這問題的話，有三個解決的方法：*

1. *不使用外接螢幕 ( HDMI 和 AV Out 都不要接東西 ) 。*
2. *設定小一點的預覽影像。*
3. *更換能輸出大一點輸出電流的電源供應器。*

*依我的情況，若是要以最大的影像做預覽輸出，****1200 mA*** *以上的電源供應器就可以了。*  
  
  
**\*\* 樹莓派中安裝 VLC Media PLayer：**  
  
VLC Medial Player 除了用來撥放影片之外，可用來做為將樹莓派攝像機的影像資料傳送到網路的軟體，所以輸入下面指令安裝 VLC：

pi@raspberrypi ~ $

pi@raspberrypi ~ $ pi@raspberrypi ~ $ sudo apt-get update && sudo apt-get install vlc

pi@raspberrypi ~ $

**\*\* Android 手機、平板 APP 安裝：**  
  
在 PC 或是 Android 裝置上開啟瀏覽器或是進入到 [Google Play](https://play.google.com/store?hl=zh-TW&tab=w8) 安奘兩個建議的 APP：

* [MX Player](https://play.google.com/store/apps/details?id=com.mxtech.videoplayer.ad)
* [RaspiCam Remote](https://play.google.com/store/apps/details?id=com.pibits.raspberrypiremotecam)

|  |
| --- |
| <http://1.bp.blogspot.com/-UX8HbbMKKl0/Uv2-xatfxoI/AAAAAAAADf8/BPv1SHtVQBs/s1600/Play-MX_Player.jpg> |
| Google Play - MX Player 下載頁面, from: [Google play](https://play.google.com/store) |

|  |
| --- |
| <http://1.bp.blogspot.com/-19Nkh2v-rwc/Uv2-yeuSoNI/AAAAAAAADgE/Bcb2iRye5Bg/s1600/Play-RaspiCam+Remote.jpg> |
| Google Play - RaspiCam Remote 下載頁面, from: [Google play](https://play.google.com/store) |

***Note:*** *如果你(妳)是使用蘋果裝置，RaspiCam Remote 就不用裝了！另外的 MX Player 可以找支援 rtsp 的撥放器下載使用即可，但就不保證一定可以用！*  
  
  
**\*\* Windows 下載可攜板 mplayer 與 VLC ：**  
  
下載網址如下：

* **VLC Media Player** ( [下載頁](http://www.videolan.org/vlc/download-windows.html)，[執行檔](http://get.videolan.org/vlc/2.1.3/win32/vlc-2.1.3-win32.exe)、[7zip可攜板](http://get.videolan.org/vlc/2.1.3/win32/vlc-2.1.3-win32.7z)、[zip可攜板](http://get.videolan.org/vlc/2.1.3/win32/vlc-2.1.3-win32.zip) )  
  上面提供的檔案是 win32 版本的，如果你(妳)的作業系統是 win95/98/me 或是 win64 的，請點及下載頁連結點選適當的檔案下載。  
  *網頁撰寫時最新的版本是：2.1.3 ( 2014/02/14 )*
* **mplayer** ( [下載頁](https://code.google.com/p/mplayer-for-windows/downloads/list) )  
  請直接上下載頁下載最新版本的 mplayer，檔案名稱應該會是：  
  ***mplayer-svn-xxxxx.7z***，xxxxx 代表版本號碼，網頁撰寫時版本為：r36350 (2014/02/14)

下載之後請先解壓縮並記住解壓縮的地方。  
  
  
**軟體測試與攝像機視訊串流檔建立：**  
  
現在你(妳)如果跟我一樣不太喜歡接一大堆東西在樹莓派上面，留下電源、記憶卡和網路線就好，其他的都可以拔掉了！因為我們接下來都會使用 SSH 連線到樹莓派，只要下指令就可以了。  
  
*ps. 因為接下來我們會使用到 SSH 連線，如果電腦中沒有這軟體可以用的話，利用一下網頁右上角的搜尋功能，在部落格中搜尋與 SSH 相關的文章，裡面有軟體可以用！*  
  
現在，手機、樹莓派和 Windows 系統都已經裝好要用的軟體與套件，可以開始動手玩了！  
  
*ps. Pi Vision 已經在上面安裝說明過了，下面就不再贅述!*  
  
  
\*\* **樹莓派攝像機影像至手機 -- 使用 RaspiCam Remote：**  
  
在手機桌面或是軟體裡面尋找 **RaspiCam Remote** 並點擊打開

|  |
| --- |
| [RaspiCam Remote 圖示](http://1.bp.blogspot.com/-vqquX8pA9JE/Uv3TQmrvk2I/AAAAAAAADgU/COVDiKW9-BE/s1600/Screenshot_2014-02-14-16-22-38-crop-blog.png) |
| RaspiCam Remote 圖示 |

每次打開軟體視窗會出現上一次與樹莓派連線的資料，如果要每次打開 **RaspiCam Remote** 自動連線上樹莓派，在 **Auto connect on startup** 前面的方框點擊一下，變成打勾圖示就可以了。  
  
按一下 **Connect** 按鈕，開始與樹莓派連線

|  |
| --- |
| [RaspiCam Remote 設定畫面](http://2.bp.blogspot.com/-E-ZEsZtNOak/Uv3TRDtm5YI/AAAAAAAADgc/AWtpqc8yjh8/s1600/Screenshot_2014-02-14-16-19-33-blog.png) |
| RaspiCam Remote 設定畫面 |

我事先找了一張紙並寫上 ruten.protes 在上面，因為沒有特別作調整以及拍攝環境不是很亮，所以拍出來的照片暗暗的。但這不影響我們的目的，因為重點不是在拍攝影片或是照片，是要能在遠端即時看到攝像機的影像。

|  |
| --- |
| [RaspCam Remote 主畫面](http://4.bp.blogspot.com/-u0TgeU4TdJU/UvwH-FOQpJI/AAAAAAAADdY/lQ-Q4X1K_70/s1600/Screenshot_2014-02-13-07-03-23.png) |
| RaspCam Remote 主畫面 |

主畫面下方有四個按鈕，由左至右為"設定"、"連續模式"、"拍照"、"存檔"。  
  
開啟"連續模式"之後，就會看到影像。按下"設定"按鈕，就會出現很多影像處理的參數和設定選單，隨便點幾個看看，可看到相對應的影像變化。  
  
按下"拍照"，就會出現一張靜止的影像，若要儲存這張影像就按下"存檔"，檔案會儲存到手機拍照目錄下的 ***Camera*** 中，以我的手機為例就是儲存到 ***/storage/sdcard0/DCIM/Camera*** 中。

|  |
| --- |
| [RaspiCam Remote 設定畫面](http://3.bp.blogspot.com/-IgT3VtjHBOs/Uv3XF7utdII/AAAAAAAADgo/WsXp_GllP0c/s1600/Screenshot_2014-02-14-16-39-47-blog.png) |
| RaspiCam Remote 設定畫面 |

RaspiCam Remote 操作上非常簡單與方便，而且不需要額外在樹莓派中安裝任何軟體，非常適合樹莓派官方攝像機的操作。  
  
*ps. RaspiCam Remote 現在不只支援樹莓派官方攝像機，也支援一般 USB Camera 。*  
  
  
**\*\* 接收由樹莓派攝像機傳送出來的串流影片：**  
  
要接收由樹莓派攝像機傳送出來的串流影片，必須先在樹莓派裡建立影片資料的串流再由網路傳送出來，只要接收端能夠接收與解讀這串流格式即可 ( 但是還是有很難開啟這串流的軟體 )。如果在\使用的時候遇到問題，建議換個軟體試試會比較快！  
  
由樹莓派建立的串流格式是 rtsp，所以只要接收端能夠解讀這格式就能夠讀取，不一定要跟我用一樣，只要能夠正確執行就好。  
  
使用 SSH 連線至樹莓派 ( 或直接在樹莓派 ) 終端機模式下輸入下面指令：

pi@raspberrypi ~ $

pi@raspberrypi ~ $ raspivid -o - -t 0 -w 300 -h 200 -n | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/}' :demux=h264

... << 以下省略 >> ...

實際指令執行如下圖所示，大約幾秒鐘的時間！

|  |
| --- |
| [建立樹莓派攝像機影像串流](http://4.bp.blogspot.com/-Xyx0GLNMwhM/UvwH_m9ql1I/AAAAAAAADd4/Gv0U-KFFR8k/s1600/cam-vcl-stream-blog.jpg) |
| 建立樹莓派攝像機影像串流 |

指令順利執行之後，串流輸出端已經建立完成，現在只要開啟支援 rtsp ( 上面指令建立 rtsp 串流 ) 的接收端軟體，就可以觀看影片了。  
  
指令我們在網頁最後再做說明。下面我們要在 Android 手機以及 Windows 作業系統接收這串流的影片，也就是看網路影片！  
  
**-- Android 手機**  
  
在手機桌面或是軟體裡面尋找 **MX Player** 並點擊打開

|  |
| --- |
| <http://2.bp.blogspot.com/-MshUYF17ow0/Uv3axTrT9yI/AAAAAAAADg0/9wJJanZq1BM/s1600/Screenshot_2014-02-14-16-54-50-crop-blog.png> |
| MX Player 圖示 |

按下手機的選單圖示開啟 MX Player 的選單，選擇"網路串流"，並在其中輸入：  
  
***rtsp://192.168.11.12:8554***  
  
其中，**192.168.11.12** 是樹莓派主機的 IP 位址；**8554** 是 Port 號碼，設定在剛剛建立串流的指令中。

|  |
| --- |
| [MX Player - "網路串流"設定視窗](http://4.bp.blogspot.com/-IaSIvRS5ZLI/Uv45ZeNi6wI/AAAAAAAADhE/zxIh7KsUAHU/s1600/Screenshot_2014-02-14-23-35-47-blog.png) |
| MX Player - "網路串流"設定視窗 |

輸入完成之後，按下"確定"，就能看到影像了！

|  |
| --- |
| <http://1.bp.blogspot.com/-6Dbw3EcoAFA/UvxKGcxwkaI/AAAAAAAADes/MTar30evTpw/s1600/Screenshot_2014-02-13-12-23-21-blog.png> |
| MX Player 播放由樹莓派傳過來的攝像機影像 |

影像看到了！現在可以換到 Windows OS 了嗎 ?  
  
**-- Windows (8) 作業系統**  
  
在桌面開啟一個 "**命令提示字元**" 視窗。還記得我們下載了兩個可攜板的軟體嗎？其中 mplayer 是一個要在命令列輸入指令的軟體，而 VLC Media Player 可以直接點擊開啟。  
  
在 ***VLC*** 目錄裡點擊 ***vcl.exe***開啟 VCL Media Play ( 媒體撥放器 )，點選 "**媒體 / 開啟網路串流...**" 開啟輸入視窗

|  |
| --- |
| [VCL Medial Player - 開啟網路串流...](http://2.bp.blogspot.com/-Z8htwO1-1hY/Uv5EbgcZ-wI/AAAAAAAADhk/E9fP0tGVkVU/s1600/VLC_Open-Stream-blog.jpg) |
| VCL Medial Player - 開啟網路串流... |

在"**網路**"頁面中，輸入網址 ***rtsp://192.168.11.12:8554*** 再按下"**播放**"按鈕，開啟網路串流

|  |
| --- |
| <http://1.bp.blogspot.com/-z-V2sekfNco/Uv5Ebu6MAcI/AAAAAAAADho/Mz36gd931s4/s1600/VLC_Open-Stream2-blog.png> |
| "網路串流"輸入畫面 |

成功在 VLC Medial Player 開啟網路串流之後，在命令提示字元視窗裡輸入下面指令格式開啟 mplayer  
  
***[Path to mplayer.exe]\mplayer.exe rtsp://192.168.11.12:8554/***  
  
例如：

c:\portables\mplayer-svn-36251\mplayer.exe rtsp://192.168.11.12:8554/

成功之後就會出現 mplayer 的視窗。  
  
如下圖就是同時執行 VLC Media Player 和 mplayer 的擷取畫面。若是手機上的 MX Player 還未關閉的話就是同時兩個地方三個軟體觀看樹莓派傳送的串流影片 ^\_^

|  |
| --- |
| [Windows 系統, mplayer 和 VLC media player 同時開啟同一串流影像](http://1.bp.blogspot.com/-rksoQ5qZNo4/Uv47obD8IdI/AAAAAAAADhQ/6gzbmjipE4g/s1600/RasPi-MultiStreaming-to-win8-blog.jpg) |
| Windows 系統, mplayer 和 VLC media player 同時開啟同一串流影像 |

所以一旦在樹莓派中建立串流輸出之後，只要接收端軟體可以正確解碼，就可以同時在不同地方開啟這串流影片，而且幾乎是即時的 ( 這跟串流輸出設定、網路品質... 等因素有關 ) !  
*Note ：rtsp 很適合用在區域網路上，但要經由路由器出去到外面網路有一點難度；要克服這問題，可以使用 VLC 將串流組態成 HTTP 格式，但這種方式比較耗 CPU 資源，有時也會丟失影像，但這方式很容易在路由器做 port forward。*  
  
  
**總結：**  
  
總體來說，使用上因為有官方與其他第三方的軟體配合，文件與資源很多，遇到問題時也可以馬上找到問題點，入門與自學上非常方便！  
  
希望上面的介紹能夠讓處在猶豫邊緣的你(妳)，提供一個做為要不要入手樹莓派攝像機的選擇參考！

**\*\* 指令 ( raspivid, cvlc, mplayer ) 簡單說明**  
  
*這裡只簡單說明在本網頁所使用的指令格式，至於完整的指令說明請閱讀下面提供的超連結：*

* ***raspivid****( or****raspistill****)：如網頁開頭所提供的*[*文件*](http://goo.gl/xyKt7Z)*資料*
* ***cvlc****：參照這*[*WiKi*](https://wiki.videolan.org/Documentation:Documentation)*，資料很多、*[*Comand-line help*](https://wiki.videolan.org/VLC_command-line_help/)*、*[*Comand-lne examples*](https://wiki.videolan.org/Documentation:Streaming_HowTo/Command_Line_Examples/)、[*VLC user guide*](http://www.videolan.org/doc/vlc-user-guide/en/index.html)*、C*[*hapter 4. The command line interface*](http://www.videolan.org/doc/vlc-user-guide/en/ch04.html)*尋找****rtp***
* ***mplayer****：官網的*[*命令列參數說明*](http://www.mplayerhq.hu/DOCS/HTML/zh_CN/commandline.html)

在上節中，樹莓派開啟串流檔的指令，同一個指令時為兩個指令的輸入：  
  
***raspivid -o - -t 0 -w 300 -h 200 -n | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/}' :demux=h264***  
  
其中，  
***raspivid*** 指令用來抓取影像

* **-o -**：將影像資料輸出到 **stdout**
* **-t 0**：-t 後面接的是開啟後多久抓取影像的時間 (單位 ms)；沒有設定就是預設時間 5 秒；設為 0 是為連續模式直到使用者按下 "Ctrl + C"
* **-w 300 -h 200**：設定影像大小 ( w: 寬度 ( 64 - 1920 )；h：高度 ( 64 - 1028 )。這邊的設定會影響網路傳輸速度，建議先設小再慢慢放大避免室這邊造成其他問題，另外也要考慮 -n ( nopreview, 預覽功能 )。
* **-n**：不預覽。***建議建立串流時使用這個選項，不加上有可能會影響樹莓派處理速度以及造成串流失敗。***

**cvlc** ( Console  VLC Player) 指令用來處理攝像機的串流

* **-vvv**：從哪裡取得串流資料，這邊是指 ***/dev/stdin***
* **--sout**：指定參數要輸出到哪裡去。這參數後面接的格式很多樣，要看上面所提供的資料並消化一下！

# Raspberry Pi 使用Python Flask達成串流視訊方法分享

by [skynet](http://hophd.com/author/admin/) | 四月 11, 2016 | [Raspberry Pi](http://hophd.com/category/raspberry-pi/) | [11 comments](http://hophd.com/raspberry-pi-python-flask-video-streaming/#respond)

之前介紹了幾篇Raspberry Pi Camera的視訊應用，再來一篇更簡單，而且效果更好。這是一個使用Python Flask Web Framework做的開源專案，作者是Miguel Grinberg，他是「Building Web APIs with Flask」一書的作者，在他的部落格文章<http://blog.miguelgrinberg.com/post/video-streaming-with-flask>中有詳細的教學。本文簡單說明一下如何實作在你的樹莓派中。先來看看效果：

基本上就是在自己的Raspberry Pi上使用pip安裝Flask模組（當然，picamera也要確定有安裝）：

|  |  |
| --- | --- |
| 1  2  3 | $ sudo apt-get udpate  $ sudo pip install picamera  $ sudo pip install Flask |

接下來，在自己的家目錄下把作者的開源專案複製一份下來：

|  |  |
| --- | --- |
| 1 | $ git clone https://github.com/miguelgrinberg/flask-video-streaming |

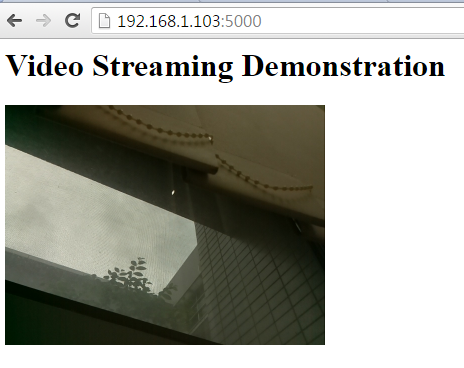
然後切換到flask-video-streaming目錄下，編輯app.py，看到前面幾行如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | 1: #!/usr/bin/env python  2: from flask import Flask, render\_template, Response  3:  4: # emulated camera  5: # from camera import Camera  6:  7: # Raspberry Pi camera module (requires picamera package)  8: from camera\_pi import Camera  ...以下省略 |

把如上所示的第5行註解，然後取消第8行的註解，存檔之後，執行python app.py就可以了，訊息如下：

|  |  |
| --- | --- |
| 1  2  3  4  5 | $ python app.py   \* Running on <http://0.0.0.0:5000/> (Press CTRL+C to quit)   \* Restarting with stat   \* Debugger is active!   \* Debugger pin code: 304-482-366 |

看到上述的訊息，就算是完成了。接下來，請開啟瀏覽器，輸入網址再加上埠號5000即可，如下圖所示：

[](http://hophd.com/wp-content/uploads/2016/04/2016-04-11_17-21-55.png)

寫到這邊也許有人會問，這些工作在一般的電腦中都可以使用，而且效果更好，為什麼要用Raspberry Pi呢？答案很簡單，因為Raspberry便宜、省電、體積小。所以，它可以一直開機而不用擔心浪費電，而且，還可以放到任何你想要放的地方，使用彈性比電腦來得大多了。

**rtsp\_streaming.sh**

|  |
| --- |
| #!/bin/bash |
|  |

|  |
| --- |
|  |
|  |

raspivid -o - -t 0 -hf -w 320 -h 240 -fps 15 | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554}' :demux=h264

3) RTSP server: <https://github.com/mpromonet/h264_v4l2_rtspserver>  
  
Get the code and build it.  
• sudo git clone <https://github.com/mpromonet/h264_v4l2_rtspserver.git>   
• sudo apt-get install liblivemedia-dev libv4l-dev cmake   
• cd h264\_v4l2\_rtspserver   
• sudo cmake .   
• sudo make

Run your server like this:

• cd ~/ sudo h264\_v4l2\_rtspserver/h264\_v4l2\_rtspserver -F 25 -W 1280 -H 720 -P 8555 /dev/video0

# Raspberry PI RTSP Guide

This is a quick guide to run an RTSP service on the raspberry pi so that you can view the pi camera using suitable clients such are vlc or gstreamer from a remote machine. Or even from another Raspberry PI. For this I am starting of with a completly fresh minimum raspbian image. I have used 2017-03-02-raspbian-jessie-lite

# Some obvious first steps.

* Change the password for user "pi"
* sudo raspi-config
  + Enable ssh
  + Enable camera
  + Increase memory split to 256MB
* sudo apt-get update
* sudo apt-get dist-upgrade
* reboot

Note: To enable ssh / camera they are under the interface options in the raspi-config program

# Install Packages

We are going to need to compile things later on so we need gstreamer and a build environment that will work for the camera plugin. These can take some time to run as the gstreamer packages will also pull down quite a large list of dependencies. This can take some time to run so you may want to combine the commands or all of the packages onto a single apt-get command.

* sudo apt-get install vim git
* sudo apt-get install gstreamer1.0-plugins-bad gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-tools libgstreamer1.0-dev libgstreamer1.0-0-dbg libgstreamer1.0-0 gstreamer1.0-omx gstreamer1.0-omx-dbg libgstreamer-plugins-base1.0-dev gtk-doc-tools

# Gstreamer camera source

We need to build the gstreamer rpi camera source so we can access the camera from a gstreamer pipeline. I ran these command from /home/pi/src in order to make this happen. This does not get included yet with gstreamer and isn't available to install from apt-get.

* git clone <https://github.com/thaytan/gst-rpicamsrc.git>
* cd gst-rpicamsrc
* ./autogen.sh
* make
* sudo make install

Lets run some tests for the new gstreamer plugin.

The rpicamsrc should be visible in the gst-inspect list. Example: "gst-inspect |grep rpicamsrc"

You should also be able to see the list of extensive properties that the element can support for controlling the camera by running "gst-inspect rpicamsrc"

For a really quick video test at this point to make sure the camera is working we can run a gstreamer pipeline that will display the camera image on the streamer and just "dump" the h274 data that is generated. I used this command. "GST\_DEBUG=3 gst-launch-1.0 -v rpicamsrc keyframe-interval=30 ! fakesink silent=false". If it works you should see some lines of text flying up the screen and a preview window with the camera image on the raspberry pi screen.

Note: I really suggest trying the above. I had a loose camera cable to the camera module and had the following error as an example.

0:00:02.662390832 2301 0x71201200 ERROR

rpicamsrc RaspiCapture.c:757:camera\_control\_callback: Received unexpected camera control callback event, 0x4f525245

# Gstreamer RTSP server library

Since the version of gstreamer on raspbian is so old. It doesn't ship with the new gstreamer rtsp server. So for this we need to clone it switch to the correct branch and compile and install it. This can be found at <https://cgit.freedesktop.org/gstreamer/gst-rtsp-server/> and take note of the output from gst-launch --version which on this system is 1.4.4 so we will need to switch to the 1.4 branch in order to compile and install the rtsp server.

* git clone git://anongit.freedesktop.org/gstreamer/gst-rtsp-server
* cd gst-rtsp-server
* git checkout 1.4
* ./autogen.sh
* make
* sudo make install

# RTSP Sever

So in order to get this completely working we can use an example program that comes with the gst-rtsp-server source code we downloaded. We will be specifically interested in the example code from gst-rtsp-server/examples/test-launch.c This will have been compiled but will not have been install with the library. All we need is a valid pipeline and the example already gives us an h264 pipeline "( videotestsrc ! x264enc ! rtph264pay name=pay0 pt=96 )". The PI is of course not powerful enough to run with the x264enc element. But the rpicamsrc produces h264 data directly as shown from "gst-inspect-1.0 rpicamsrc"

Pad Templates:

SRC template: 'src'

Availability: Always

Capabilities:

video/x-h264

width: [ 1, 2147483647 ]

height: [ 1, 2147483647 ]

framerate: [ 0/1, 2147483647/1 ]

stream-format: byte-stream

alignment: nal

profile: { baseline, main, high }

image/jpeg

width: [ 1, 2147483647 ]

height: [ 1, 2147483647 ]

framerate: [ 0/1, 2147483647/1 ]

video/x-raw

format: { I420, RGB, BGR, RGBA }

width: [ 1, 2147483647 ]

height: [ 1, 2147483647 ]

framerate: [ 0/1, 2147483647/1 ]

So for a suitable pipeline I trying the following first from gst-launch to see if it runs ok. This should produce a 2mbit stream with key frames produced every 15 frames (1 key frame per second) and and frame rate of 15 per second.

GST\_DEBUG=3 gst-launch-1.0 -v rpicamsrc preview=false bitrate=2000000 keyframe-interval=15 \

! video/x-h264, framerate=15/1 ! h264parse ! fakesink silent=false

Now for a final test. From the gst-rtsp-server/examples directory I run the following. Though it prints the URL as 127.0.0.1. This can be replaced with the ip address of the pi.

./test-launch "( rpicamsrc preview=false bitrate=2000000 keyframe-interval=15 ! video/x-h264, framerate=15/1 ! h264parse ! rtph264pay name=pay0 pt=96 )"

From a remote machine I can then access this using vlc. From ubuntu I ran "vlc rtsp://piaddress:8544/" and get a slightly lagged video stream. The reason for the lag is mostly due to the buffering inside vlc that is hard to control. If you want something really low latency try the following gstreamer pipeline from the remote machine or change the setting in vlc so it doesn't buffer.

gst-launch-1.0 rtspsrc location="rtsp://piaddress:8554/test" latency=0 \

! rtph264depay ! decodebin ! videoconvert ! ximagesink

# Some proof

# Some further ideas

If you look at the output of the "gst-inspect rpicamsrc" you will find options to be able to flip / rotate the image as well as control specific capture / encoding settings available on the camera.

This will currently only support single client being connected. Making it support multiple tcp clients requires a much more complex application as it need to split the pipeline into multiple components

Authentication is probably also an important feature. There is more example code with gst-rtsp-server which will demonstrate this.

# RTSP Server

###### Set up a Real-time Streaming Server (RTSP)

Install the required packages:

raspberrypi ~ $ sudo apt-get install vlc

Run UV4L with the raspicam driver:

raspberrypi ~ $ uv4l --driver raspicam --auto-video\_nr --framerate 25 --encoding=h264 --extension-presence=0

[core] Device detected!

[core] Registering device node /dev/video0

Optionally give a real-time scheduling policy to the driver process for better performance (which is the same as running uv4l with sudo uv4l –sched-rr):

raspberrypi ~ $ sudo chrt -a -r -p 99 `pgrep uv4l`

Run the real-time VideoLan streaming server on your Raspberry Pi with the preferred resolution:

raspberrypi ~ $ cvlc -vvv v4l2c:///dev/video0:width=640:height=480:chroma=H264 --sout '#rtp{sdp=rtsp://:8554/}' --demux h264

although more appropriate in theory, the above command has been reported to leak memory because of some buggy versions of cvlc. If this is the case, try the following alternative command to run the server:

raspberrypi ~ $ dd if=/dev/video0 bs=1M | cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/}' --demux=h264

Now you can connect to your Raspberry Pi from the client, for example (don’t forget the final slash):

mypc ~ $ vlc rtsp://raspberrypi:8554/

where raspberrypi is the host name or IP of your RaspberryPi.

You can set any image property on the fly (while the camera is streaming). For example, to list all the available controls and change the brightness:

raspberrypi ~ $ sudo apt-get install v4l-utils

raspberrypi ~ $ v4l2-ctl --list-ctrls --device /dev/video0

brightness (int) : min=0 max=100 step=1 default=50 value=50

contrast (int) : min=-100 max=100 step=1 default=0 value=0

saturation (int) : min=-100 max=100 step=1 default=0 value=0

iso (int) : min=0 max=1200 step=1 default=400 value=400

horizontal\_mirror (bool) : default=0 value=0

vertical\_mirror (bool) : default=0 value=0

sharpness (int) : min=-100 max=100 step=1 default=0 value=0

raspberrypi ~ $ v4l2-ctl --set-ctrl=brightness=50 --device=/dev/video0

A more user-friendly interface for changing the image properties while streaming with this/any method is the Conrol Panel web page coming along with the HTTP Streaming Server plug-in for UV4L.

## How to stream from RaspberryPi PiCamera

May 18, 2017

## Pre-requisites

OS: Raspbian >= 8

Do sudo raspi-config and enable PiCamera, in ‘Memory Split’, give GPU more than 128M of memory. Connect PiCamera, reboot.

Test PiCamera works by raspistill -o photo.jpg.

Load v4l2 module:

sudo modprobe bcm2835-v412

Optionally add bcm2835-v412 to /etc/modules file.

v4l2 should already present in the OS, if not:

sudo apt-get install v4l-utils

There’re multiple ways to stream PiCamera to remote clients. Here I present 4.

## Method 1: vlc

Make sure vlc is installed:

sudo apt-get install vlc

shell script:

raspivid -o - -n -fps 15 -w 1280 -h 720 -t 0 \

|cvlc -vvv stream:///dev/stdin \

--sout '#rtp{sdp=rtsp://:8554/}' \

:demux=h264 :h264-fps=15

When you run the script, it will stream, the stream can be viewed from rtsp://raspberrypi\_ip:8554/ from VLC on a remote computer.

## Method 2: v4l2rtspserver

Details are in [v4l2rtspserver github](https://github.com/mpromonet/v4l2rtspserver). But basically:

sudo apt-get install cmake liblog4cpp5-dev libv4l-dev

wget www.live555.com/liveMedia/public/live555-latest.tar.gz

tar xfz live555-latest.tar.gz

cd live/

./genMakefiles linux

make CPPFLAGS=-DALLOW\_RTSP\_SERVER\_PORT\_REUSE=1

sudo make install

cd

git clone https://github.com/mpromonet/v4l2rtspserver.git

cd v4l2rtspserver/

cmake .

make

sudo make install

Then you can run:

v4l2rtspserver -F15 -H 720 -W1280 -P 8555 /dev/video0

The stream can now be viewed from rtsp://raspberrypi\_ip:8555/unicast from VLC on another computer.

## Method 3: ffmpeg/avconv

Install avconv by sudo apt-get install libav-tools.

Don’t bother trying to get ffmpeg from third-part apt source, it mostly likely would not work. Also don’t bother compile (it takes a long time) or cross-compile (hassle) ffmpeg. avconv works for our purpose.

shell script:

avconv -f video4linux2 -framerate 15 -video\_size 960x540 \

-i /dev/video0 -an -f flv -y \

rtmp://yourstreamserver/live/pi

Run it and view the stream on another computer with vlc: rtmp://yourstreamserver/live/pi.

yourstreamserver is the name or IP of your RTMP stream server, it can be set up with Nginx-rtmp, Wowza, or srs etc.

Using this method, the stream has longest delay, cpu load is pretty high. I will try to reduce it when I get time.

## Method 4: gstreamer

Install gstreamer 1.0:

sudo apt-get install gstreamer1.0-tools

Test we can stream with a MP4 video file:

gst-launch-1.0 -v filesrc location="video.mp4" \

! qtdemux ! video/x-h264 ! h264parse \

! flvmux ! rtmpsink \

location="rtmp://yourstreamserver/live/pi"

After this works, we try to make it work with camera. Unfortunately I couldn’t get v4l2src to work with gstreamer to stream RTMP.

I turned to [gst-rpicamsrc](https://github.com/thaytan/gst-rpicamsrc).

Install dependencies:

sudo apt-get install autoconf automake libtool

sudo apt-get install libgstreamer1.0-dev

sudo apt-get install libgstreamer-plugins-base1.0-dev

sudo apt-get install libraspberrypi-dev

Build gst-rpicamsrc:

git clone https://github.com/thaytan/gst-rpicamsrc.git

cd gst-rpicamsrc/

./autogen.sh --prefix=/usr --libdir=/usr/lib/arm-linux-gnueabihf/

make

sudo make install

sudo reboot

After reboot, create a script:

gst-launch-1.0 -v rpicamsrc \

!'video/x-h264,width=1280,height=720,framerate=15/1' \

! h264parse ! flvmux \

! rtmpsink location="rtmp://yourstreamserver/live/pi"

The stream from this method has much lower delay compared with method 3. It’s still much slower than method 1 and 2. This is because it send stream to a RTMP server, you request stream from the server, In method 1/2, the stream is sent directly to you from RPi. The streams from method 3/4 can be viewed in a browser (using for example JWPlayer), while the streams from method 1/2 can’t.