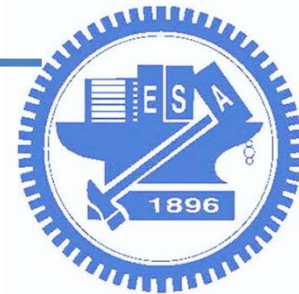


# Outline

- 嵌入式應用: 網路攝影機
  - 控制Raspberry Pi Camera
  - 建立網路串流
    - 使用 RTSP + H.264
    - 使用 HTTP + MJPG
    - 使用 RTMP



# PI Camera Spec.

- ❑ Sensor: OmniVision OV5647 (5MP)
- ❑ 靜態拍照最高解析度:2592 x 1944 pixel
- ❑ Pixel Size:1.4 x 1.4  $\mu\text{m}$
- ❑ Lens: f=3.6 mm, f/2.9
- ❑ Angle of View:54 x 41 degrees
- ❑ Field of View:2.0 x 1.33 m at 2 m
- ❑ Fixed Focus:1m to infinity
- ❑ 動態攝影最高解析度:1080p@30 FPS with H.264/AVC

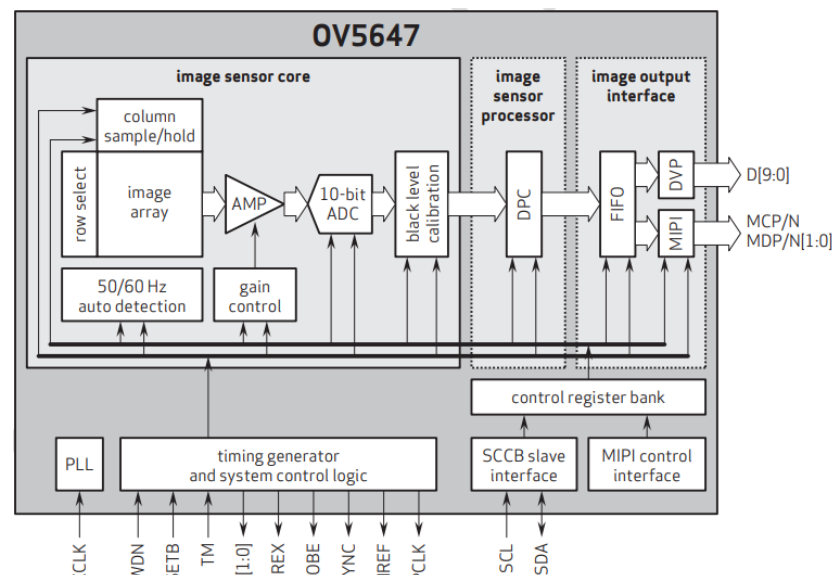
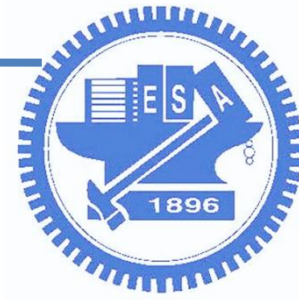


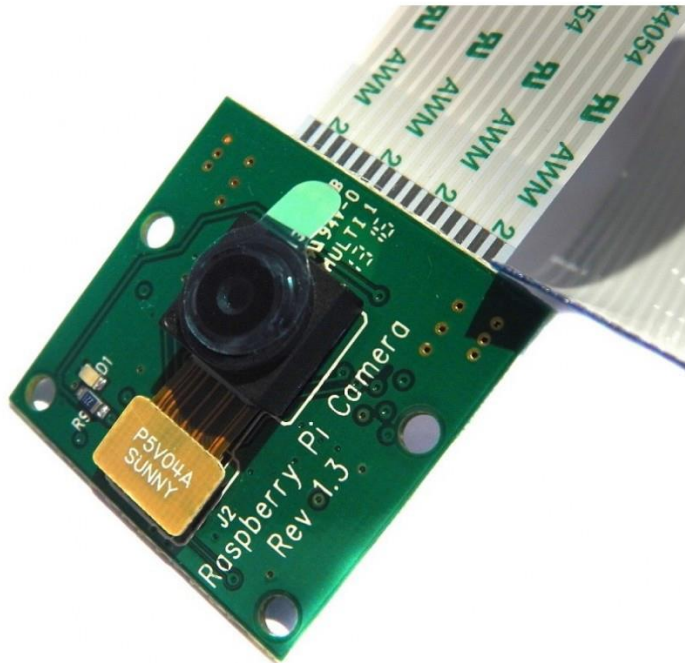
table 2-1 format and frame rate

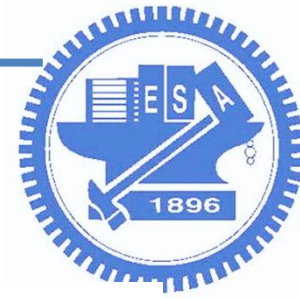
format	resolution	frame rate	scaling method	pixel clock
5 Mpixel	2592x1944	15 fps	full resolution	80 MHz
1080p	1920x1080	30 fps	cropping	68 MHz
960p	1280x960	45 fps	cropping, subsampling/ binning	91.2 MHz
720p	1280x720	60 fps	cropping, subsampling/ binning	92 MHz
VGA	640x480	90 fps	cropping, subsampling/ binning	46.5 MHz
QVGA	320x240	120 fps	cropping, subsampling/ binning	32.5 MHz



# Install PI camera

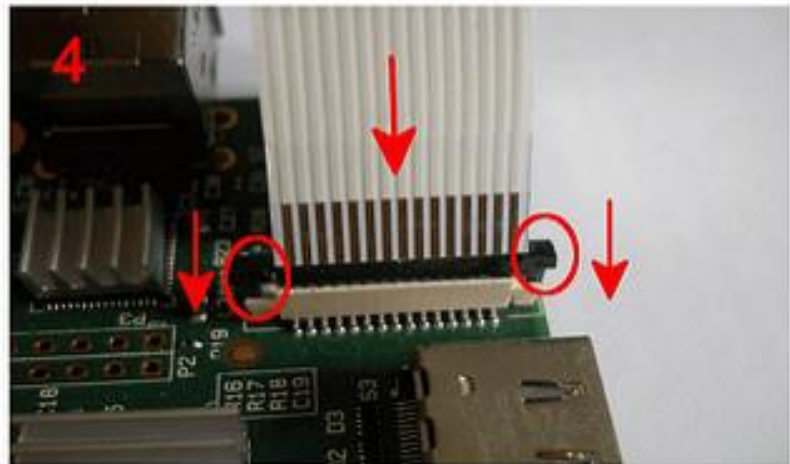
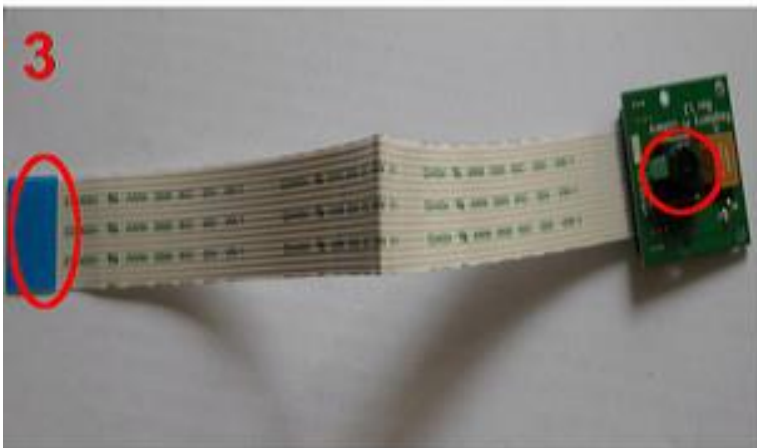
15-Pins, CSI interface





# Install PI camera

The CSI Port



[illegible]



- 
- The screenshot shows the Raspberry Pi Software Configuration Tool (raspi-config) running in a terminal window titled "(COM8) [80x24]". The menu options at the top are: 連線(C), 編輯(E), 檢視(V), 視窗(W), 選項(O), and 說明(H). The main menu lists several configuration options, each preceded by "? ?":
- P1 Camera
  - P2 SSH
  - P3 VNC
  - P4 SPI
  - P5 I2C
  - P6 Serial
  - P7 l-Wire
  - P8 Remote GPIO
- The "P1 Camera" option is currently selected, highlighted with a light blue background. To the right of each option is a description: "Enable/Disable connection to the camera module" for P1, "Enable/Disable remote command line access" for P2, "Enable/Disable graphical remote access" for P3, "Enable/Disable automatic loading of kernel modules" for P4, "Enable/Disable automatic loading of firmware" for P5, "Enable/Disable shell and kernel modules" for P6, "Enable/Disable one-wire interface" for P7, and "Enable/Disable remote access to GPIO" for P8. At the bottom of the screen, there are two navigation options: "<Select>" and "<Back>". The terminal output is displayed in a monospaced font on a black background.





# Raspbian configuration

- sudo raspi-config -> P1 Camera -> Enable





# Raspbian configuration

- sudo raspi-config -> P1 Camera -> Enable





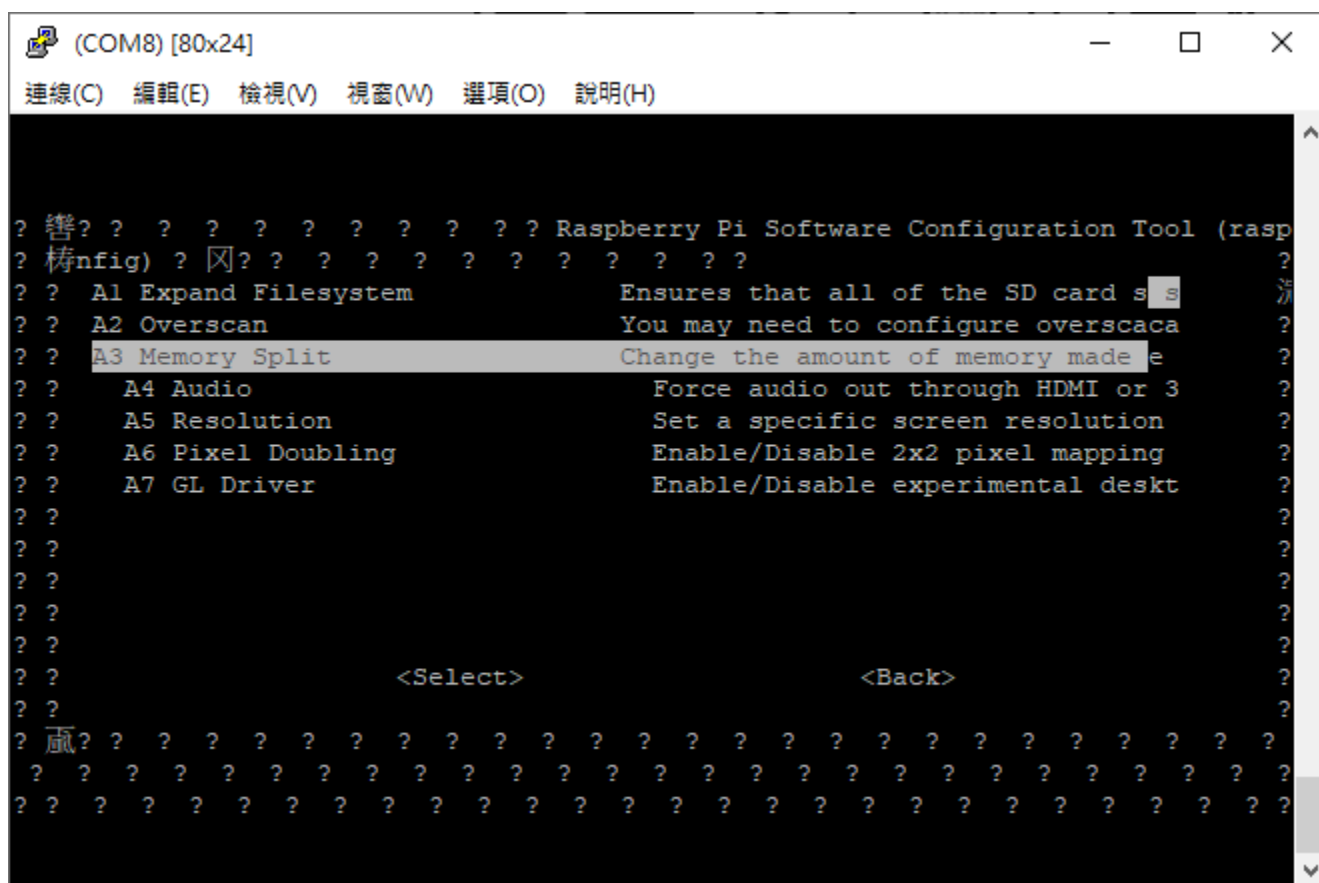


- 
- (COM8) [80x24]
- 連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
- Raspberry Pi 3 Model B Rev 1.2
- Raspberry Pi Software Configuration Tool (raspi-config)
- 1 Change User Password Change password for the current user
- 2 Network Options Configure network settings
- 3 Boot Options Configure options for start-up
- 4 Localisation Options Set up language and regional settings
- 5 Interfacing Options Configure connections to peripheral devices
- 6 Overclock Configure overclocking for your processor
- 7 Advanced Options Configure advanced settings
- 8 Update Update this tool to the latest version
- 9 About raspi-config Information about this configuration tool
- <Select> <Finish>



# Raspbian configuration

## □ Optional

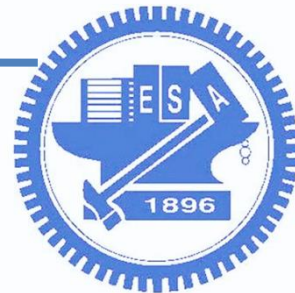




# Raspbian configuration

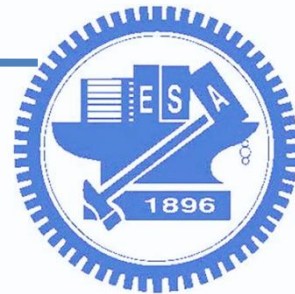
## □ Optional





# Camera commands

- Take a picture: `raspistill`
  - `raspistill -n -t 3000 -o test.png -e png -w 640 -h 480`
  - 3秒後拍照, 並編碼成png格式, 長640x寬480, 無預覽
    - n: Do not display a preview window
    - t: timeout, Time before the camera takes picture and shuts down
    - o: output filename
    - e: Encoding to use for output file (jpg, bmp, gif, and png)
    - w: Set image width <size>
    - h: Set image height <size>



# Camera commands

## □ Record a video: raspivid

□ `Raspivid -n -t 5000 -w 640 -h 480 -o video.h264`

■ 錄5秒的1080p30影片, 長640x寬480, 無預覽

■ t: Time (in ms) to capture for. Default = 5 sec.

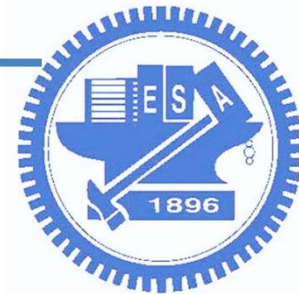
■ o: output filename

■ w: Set image width <size>

■ h: Set image height <size>

## □ Official document

□ <https://github.com/raspberrypi/documentation/blob/master/raspbian/applications/camera.md>



# Error message?

- ❑ Msg: Camera is not enabled in this build



```
(COM8) [80x24]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

pi@raspberrypi:~$ raspistill -n
mmal: mmal_vc_component_create: failed to create component 'vc.ril.camera' (l:ENOMEM)
mmal: mmal_component_create_core: could not create component 'vc.ril.camera' (l)
mmal: Failed to create camera component
mmal: main: Failed to create camera component
mmal: Camera is not enabled in this build. Try running "sudo raspi-config" and ensure that "camera" has been enabled
```

- ❑ Sol: go to “sudo raspi-config”, then enable camera



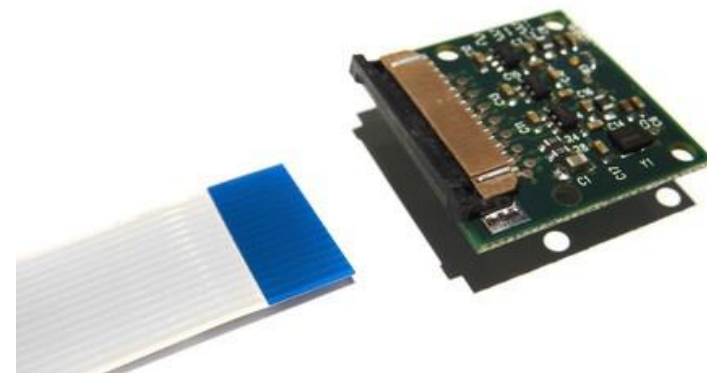
# Error message?

## □ Msg: Camera is not detected

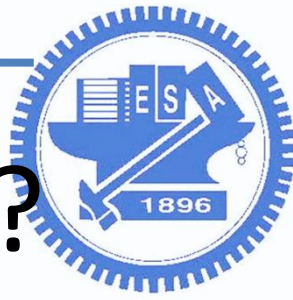
```
(COM8) [80x24]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
pi@raspberrypi:~$ raspistill -n
mmal: Cannot read camera info, keeping the defaults for OV5647
mmal: mmal_vc_component_create: failed to create component 'vc.ril.camera' (1:EN
OMEM)
mmal: mmal_component_create_core: could not create component 'vc.ril.camera' (1)
mmal: Failed to create camera component
mmal: main: Failed to create camera component
mmal: Camera is not detected. Please check carefully the camera module is instal
led correctly
```

## □ Sol:

- 重新安裝camera,或是更換排線
- 或是檢查camera module是否鬆脫







# How to view image/video?

## □ Methods:

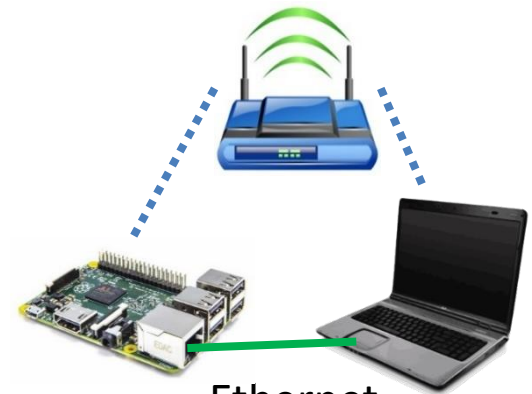
1. `python -m SimpleHTTPServer 8000`
2. `winscp`
3. `vnc`
4. HDMI



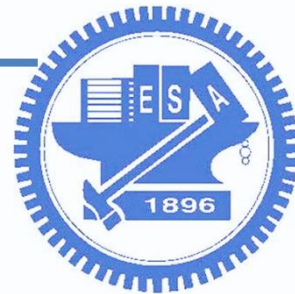
Wi-Fi AP



Wi-Fi hotspot



Wired + Wireless



# Python code

- Sample code for taking a picture

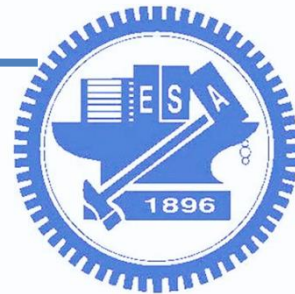
```
import picamera
import time

camera = picamera.PiCamera()
time.sleep(2) # Camera warm-up time
camera.capture('test.jpg')
```

## 9.1. PiCamera

```
class picamera.PiCamera(camera_num=0, stereo_mode='none',
stereo_decimate=False, resolution=None, framerate=None,
sensor_mode=0, led_pin=None, clock_mode='reset',
framerate_range=None) [source]
```

```
capture(output, format=None, use_video_port=False, resize=None, splitter_port=0,
bayer=False, **options) [source]
```



# Python code

## □ Sample code for record a video

```
start_recording(output, format=None, resize=None, splitter_port=1, **options) [source]
```

Start recording video from the camera, storing it in *output*.

```
import picamera
```

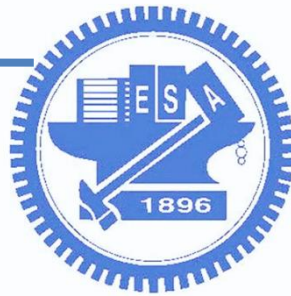
```
camera = picamera.PiCamera()
camera.start_recording('video.h264')
camera.wait_recording(3)
camera.stop_recording()
```

```
wait_recording(timeout=0, splitter_port=1) [source]
```

Wait on the video encoder for timeout seconds.

```
stop_recording(splitter_port=1) [source]
```

Stop recording video from the camera.

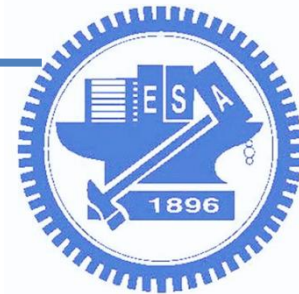


# Python code

- Sample code for taking many pictures

```
import time
import picamera
with picamera.PiCamera() as camera:
    camera.start_preview()
    try:
        for i, filename in enumerate(camera.capture_continuous('image{counter:02d}.jpg')):
            print(filename)
            time.sleep(1)
            if i == 59:
                break
    finally:
        camera.stop_preview()
```

File name



# Python schedule

## □ Usage: pip install schedule

```
import schedule
import time
```

```
def job():
    print("I'm working...")
```

```
schedule.every(10).minutes.do(job)
schedule.every().hour.do(job)
schedule.every().day.at("10:30").do(job)
schedule.every().monday.do(job)
schedule.every().wednesday.at("13:15").do(job)
schedule.every().minute.at(":17").do(job)
```

```
while True:
    schedule.run_pending()
    time.sleep(1)
```

**at**(*time\_str*)

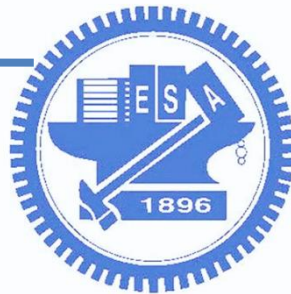
Specify a particular time that the job should be run at.

**Parameters:** **time\_str** – A string in one of the following formats:

*HH:MM:SS*, *HH:MM*, *`:MM`*, *:SS*. The format must make sense given how often the job is repeating; for example, a job that repeats every minute should not be given a string in the form *HH:MM:SS*. The difference between *:MM* and *:SS* is inferred from the selected time-unit (e.g. *every().hour.at(':30')* vs. *every().minute.at(':30')*).

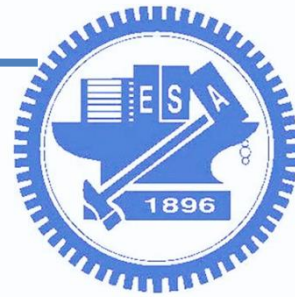
**Returns:** The invoked job instance

[source]



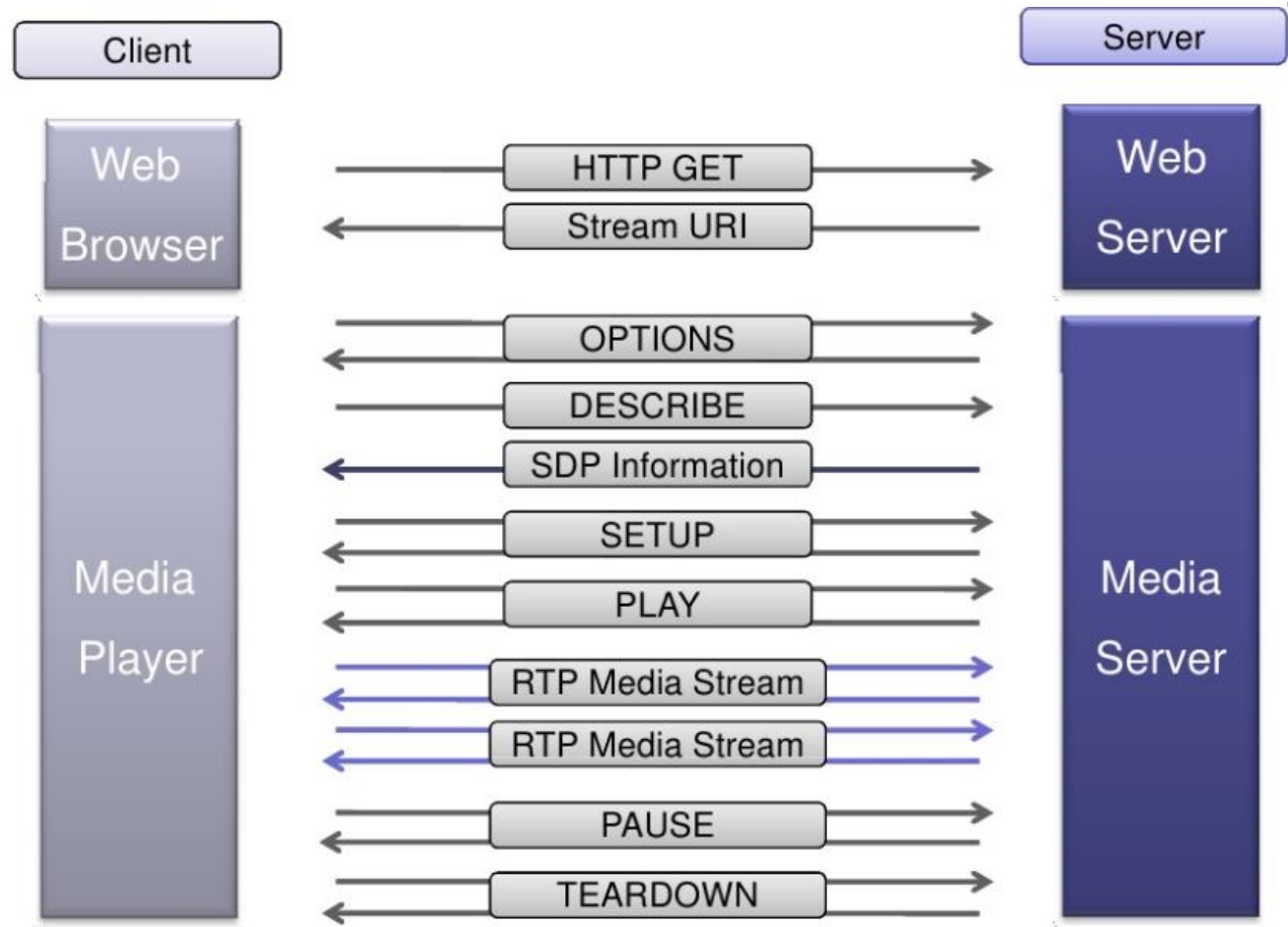
# IP cam

- Video Streaming
  - 使用 RTSP + H.264
  - 使用 HTTP + MJPG
  - 使用 RTMP



# 1. RTSP

The Real Time Streaming Protocol, or RTSP, is an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP (RFC 1889).





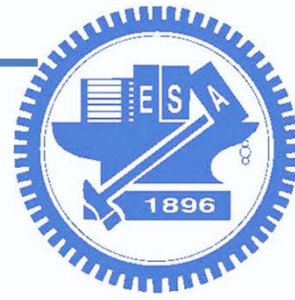


# 1. RTSP on Raspberry Pi

- Execute the command

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

```
(COM8) [80x24]  
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)  
o=- 16162396461258043171 16162396461258043171 IN IP4 raspberrypi  
s=Unnamed  
i=N/A  
c=IN IP4 0.0.0.0  
t=0 0  
a=tool:vlc 3.0.6  
a=recvonly  
a=type:broadcast  
a=charset:UTF-8  
m=video 0 RTP/AVP 96  
b=RR:0  
a=rtpmap:96 H264/90000  
a=fmtp:96 packetization-mode=1;profile-level-id=640028;sprop-parameter-sets=J2QA  
KKWrQKD9APEiag==,KO4BDyw=;  
  
[75400520] main input debug: Buffering 66%  
[75400520] main input debug: Buffering 73%  
[75400520] main input debug: Buffering 80%  
[75400520] main input debug: Buffering 86%  
[75400520] main input debug: Buffering 93%  
[75400520] main input debug: Buffering 100%  
[75400520] main input debug: Stream buffering done (320 ms in 335 ms)  
[75400520] main input debug: Decoder wait done in 0 ms
```



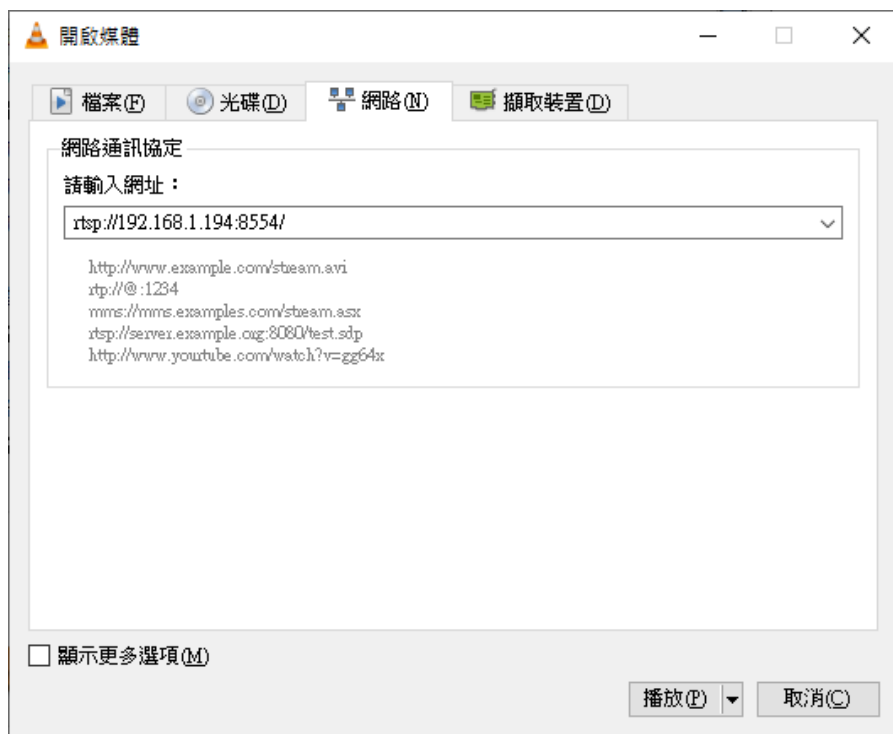
# 1. RTSP on Raspberry Pi

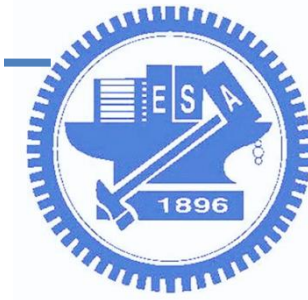
- `cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554}' :demux=h264`
  - ▣ `stream:` Stream MRL syntax: `[[access][[/demux]://]URL[#[title][:chapter]][:title][:chapter]] [:option=value ...]`
  - ▣ `/dev/stdin:` Standard input. The source of input data for command line programs. Here, the input is from raspivid.
  - ▣ `sout:` stream output
  - ▣ `rtp:` A Transport Protocol for Real-Time Applications
  - ▣ `sdp:` RTSP Session Descriptions
  - ▣ `rtsp:` an application-level protocol
  - ▣ `demux:` handle the different formats



# 1. RTSP on Raspberry Pi

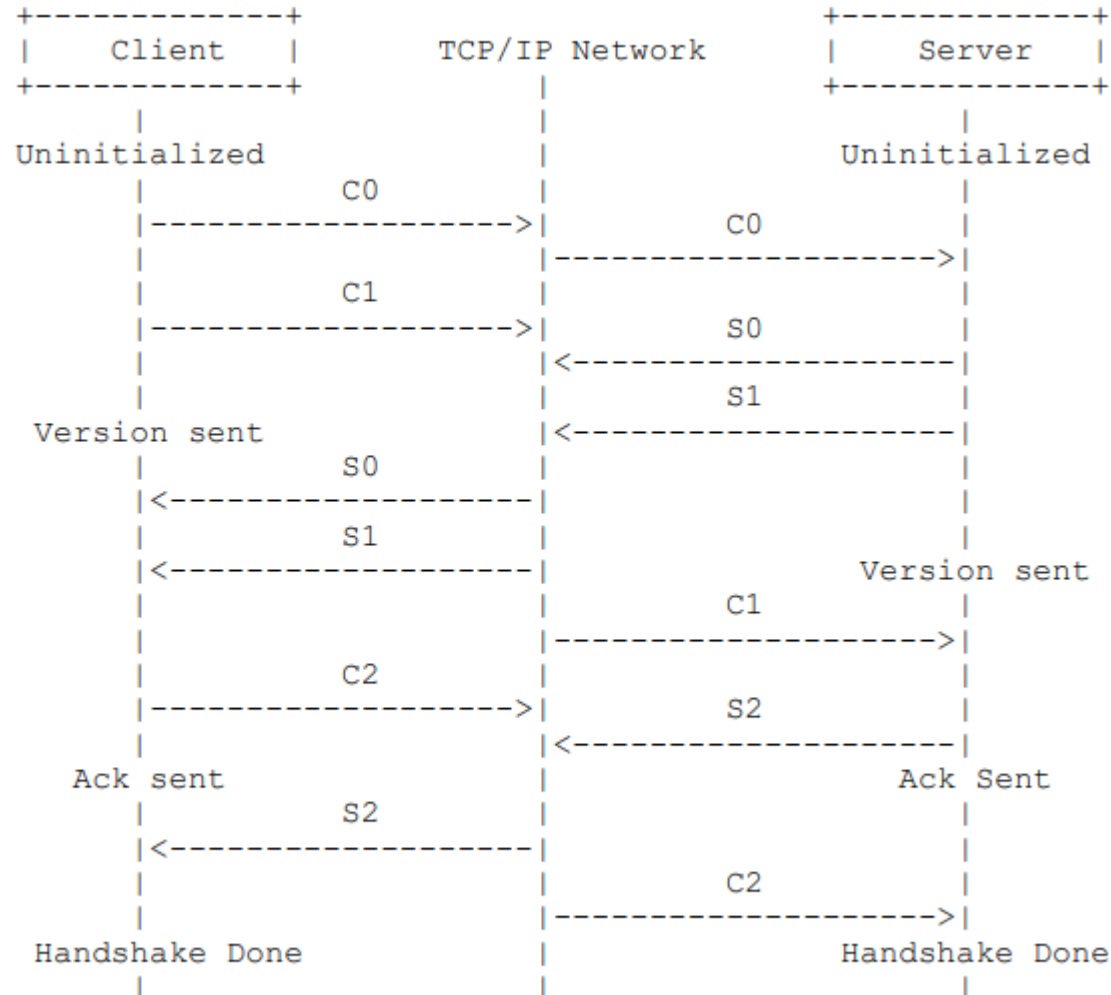
- Use VLC to watch video





## 2. RTMP

### 5.2.5. Handshake Diagram



Pictorial Representation of Handshake



## 2. RTMP to Youtube

[https://www.youtube.com/live\\_dashboard](https://www.youtube.com/live_dashboard)

基本資訊

串流選項

資訊卡

Kun-Ru Wu即時串流

Stream test

☐ 安排下一部直播影片的播出時間

類別

人物與網誌

隱私設定

不公開

進階設定

編碼器設定

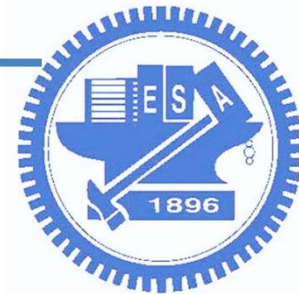
伺服器網址

rtmp://a.rtmp.youtube.com/live2

串流名稱/金鑰

.....

顯示



## 2. RTMP on PI

### □ Execute command:

```
raspivid -o - -t 0 -vf -hf -fps 10 -b 500000 | ffmpeg -re -ar 44100 -ac 2 -acodec pcm_s16le -f s16le -ac 2 -i /dev/zero -f h264 -i - -vcodec copy -acodec aac -ab 128k -g 50 -strict experimental -f flv rtmp://a.rtmp.youtube.com/live2/keyxxxx
```

```
(COM8) [80x24]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

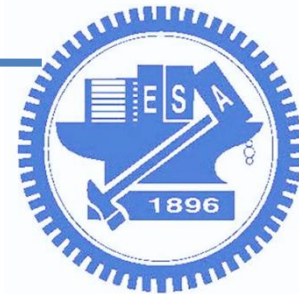
Stream #1:0: Video: h264 (High), yuv420p(progressive), 1920x1080, 25 fps, 25 tbr, 1200k tbn, 50 tbc
Output #0, flv, to 'rtmp://a.rtmp.youtube.com/live2/':
Metadata:
  encoder      : Lavf57.56.101
Stream #0:0: Video: h264 (High) ([7][0][0][0] / 0x0007), yuv420p(progressive), 1920x1080, q=2-31, 25 fps, 25 tbr, 1k tbn, 1200k tbc
Stream #0:1: Audio: aac (LC) ([10][0][0][0] / 0x000A), 44100 Hz, stereo, fltp, 128 kb/s
Metadata:
  encoder      : Lavc57.64.101 aac
Stream mapping:
  Stream #1:0 -> #0:0 (copy)
  Stream #0:0 -> #0:1 (pcm_s16le (native) -> aac (native))
[flv @ 0x18caf30] Timestamps are unset in a packet for stream 0. This is deprecated and will stop working in the future. Fix your code to set the timestamps properly
[h264 @ 0x18556f0] Thread message queue blocking; consider raising the thread_queue_size option (current value: 8)
frame= 14 fps=0.0 q=-1.0 size=      57kB time=00:00:00.52 bitrate= 897.4kbits/
frame= 26 fps= 26 q=-1.0 size=     118kB time=00:00:01.02 bitrate= 943.3kbits/
frame= 39 fps= 26 q=-1.0 size=     210kB time=00:00:01.53 bitrate=1122.0kbits/
frame= 51 fps= 25 q=-1.0 size=     314kB time=00:00:02.04 bitrate=1258.0kbits/
speed=1.01x
```

伺服器網址

rtmp://a.rtmp.youtube.com/live2

串流名稱/金鑰

.....



## 2. RTMP on PI

- ❑ `ffmpeg -re -ar 44100 -ac 2 -acodec pcm_s16le -f s16le -ac 2 -i /dev/zero -f h264 -i - -vcodec copy -acodec aac -ab 128k -g 50 -strict experimental -f flv rtmp://a.rtmp.youtube.com/live2/keyxxxx`
  - ❑ `re`: Read input at native frame rate.
  - ❑ `ar`: Set the audio sampling frequency.
  - ❑ `ac`: audio channels.
  - ❑ `acodec`: Set the audio codec.
  - ❑ `f`: Force input or output file format. (S16LE: 16-bit signed PCM audio)
  - ❑ `vcodec`: set the video codec. Use “copy” to indicate that the stream is not to be re-encoded.





## 2. RTMP on PI

### □ Start streaming...

● 直播中  
串流狀況 ?

直播中  
全世界都能透過網路觀賞您的影片。如要停止直播，請使用編碼器。

⌚ 00:00:13  
經過時間

👤 0  
正在觀看

COM8 [80x24]

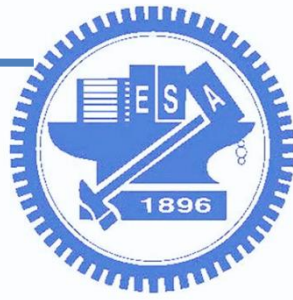
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

frame=	248	fps=	21	q=-1.0	size=	1568kB	time=00:00:09.91	bitrate=1295.9kbps/
frame=	253	fps=	21	q=-1.0	size=	1585kB	time=00:00:10.12	bitrate=1282.9kbps/
frame=	258	fps=	20	q=-1.0	size=	1603kB	time=00:00:10.30	bitrate=1273.8kbps/
frame=	263	fps=	20	q=-1.0	size=	1630kB	time=00:00:10.51	bitrate=1269.2kbps/
frame=	268	fps=	20	q=-1.0	size=	1657kB	time=00:00:10.70	bitrate=1267.9kbps/
frame=	273	fps=	19	q=-1.0	size=	1686kB	time=00:00:10.91	bitrate=1265.7kbps/
frame=	278	fps=	19	q=-1.0	size=	1716kB	time=00:00:11.12	bitrate=1263.8kbps/
frame=	283	fps=	19	q=-1.0	size=	1745kB	time=00:00:11.30	bitrate=1264.0kbps/
frame=	288	fps=	18	q=-1.0	size=	1774kB	time=00:00:11.51	bitrate=1262.1kbps/
frame=	293	fps=	18	q=-1.0	size=	1802kB	time=00:00:11.70	bitrate=1261.2kbps/
frame=	298	fps=	18	q=-1.0	size=	1837kB	time=00:00:11.91	bitrate=1263.4kbps/
frame=	303	fps=	18	q=-1.0	size=	1900kB	time=00:00:12.12	bitrate=1284.4kbps/
frame=	308	fps=	17	q=-1.0	size=	1915kB	time=00:00:12.30	bitrate=1275.1kbps/
frame=	312	fps=	17	q=-1.0	size=	1939kB	time=00:00:12.46	bitrate=1274.0kbps/
frame=	318	fps=	17	q=-1.0	size=	1967kB	time=00:00:12.72	bitrate=1266.2kbps/
frame=	323	fps=	17	q=-1.0	size=	1995kB	time=00:00:12.91	bitrate=1265.7kbps/
frame=	329	fps=	17	q=-1.0	size=	2024kB	time=00:00:13.12	bitrate=1263.7kbps/
frame=	334	fps=	17	q=-1.0	size=	2059kB	time=00:00:13.35	bitrate=1263.3kbps/
frame=	339	fps=	16	q=-1.0	size=	2082kB	time=00:00:13.52	bitrate=1261.2kbps/
frame=	344	fps=	16	q=-1.0	size=	2120kB	time=00:00:13.74	bitrate=1263.2kbps/
frame=	349	fps=	16	q=-1.0	size=	2151kB	time=00:00:13.95	bitrate=1262.5kbps/
frame=	354	fps=	16	q=-1.0	size=	2183kB	time=00:00:14.16	bitrate=1262.5kbps/
frame=	359	fps=	16	q=-1.0	size=	2211kB	time=00:00:14.35	bitrate=1262.3kbps/

speed=0.632x

⏮ ⏪ ⏩ ⏭

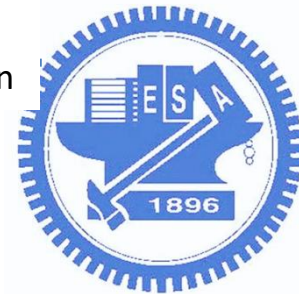
建立焦點片段 變更縮圖



## 2. RTMP on PI

- Watch video



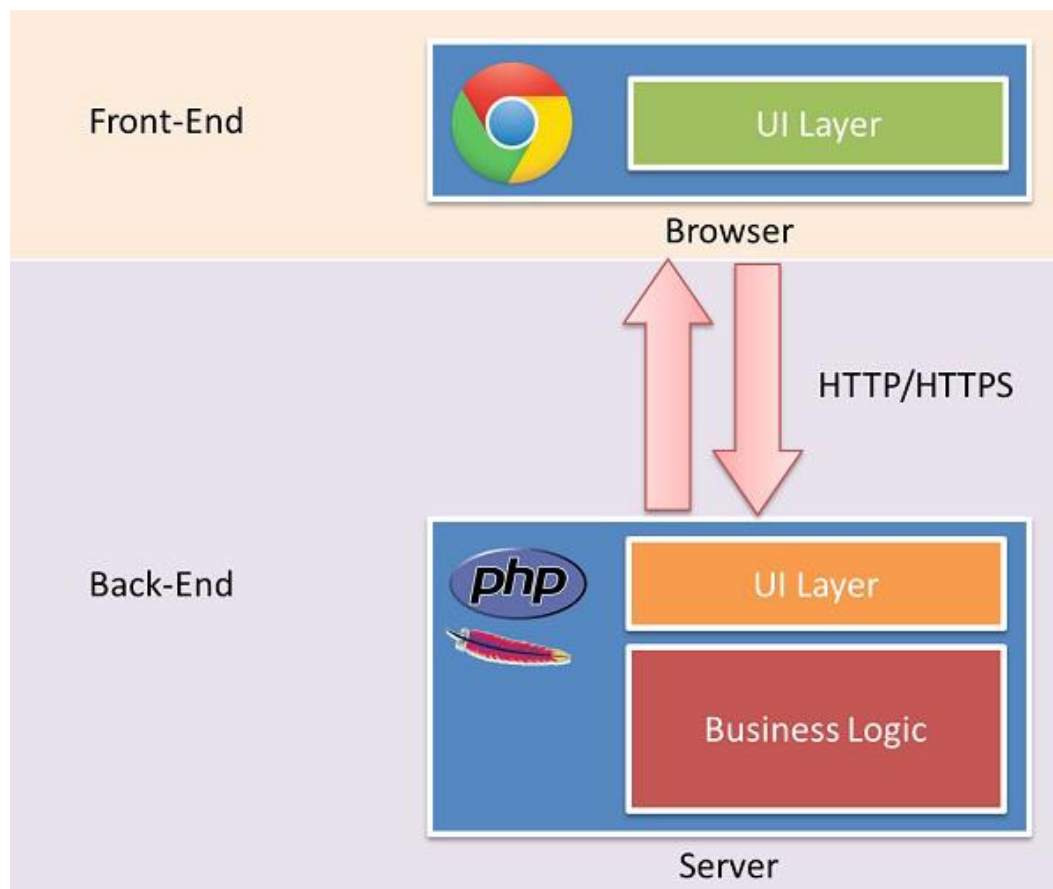


## 3. HTTP + MJPG

### □ MJPEG = Motion JPEG

- 一種視訊壓縮格式
- 每一個frame都使用 JPEG編碼
- 對運算能力與記憶體的需求較低
- 許多網頁瀏覽器原生支援M-JPEG

- Flask 是一個輕量型的 Python Web 應用程式架構，可提供 URL 路由和頁面轉譯的基本要素。

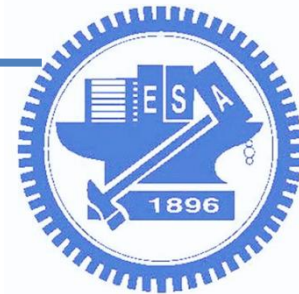




# 3. HTTP + MJPG on PI

- Install tools:
  - `sudo pip install request flask numpy`
  - `sudo modprobe bcm2835-v4l2`
  - Download and unzip “w7\_mjpg\_sample.zip” file
  - `sudo python app-camera.py`

```
(COM8) [80x24]
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)
pi@raspberrypi:~/camera-python-opencv/camera-python/05-streaming$ sudo python ap
p-camera.py
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger pin code: 109-454-584
```



# 3. MJPG on PI

## □ Sample code (app-camera.py)

```
from flask import Flask, render_template, Response
from camera_pi import Camera

app = Flask(__name__)

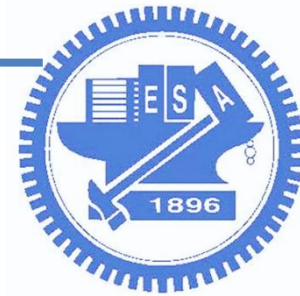
@app.route('/')
def index():
    return render_template('stream.html')

def gen(camera):
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(Camera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80, debug=True)
```

<h1>Hello Stream</h1>  

# 3. MJPG on PI

## □ camera\_pi.py

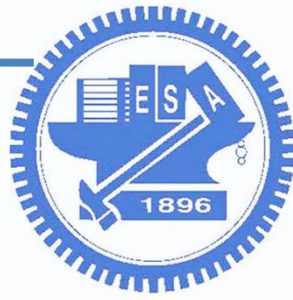
```
import cv2

class Camera(object):
    def __init__(self):
        if cv2.__version__.startswith('2'):
            PROP_FRAME_WIDTH = cv2.cv.CV_CAP_PROP_FRAME_WIDTH
            PROP_FRAME_HEIGHT = cv2.cv.CV_CAP_PROP_FRAME_HEIGHT
        elif cv2.__version__.startswith('3'):
            PROP_FRAME_WIDTH = cv2.CAP_PROP_FRAME_WIDTH
            PROP_FRAME_HEIGHT = cv2.CAP_PROP_FRAME_HEIGHT

        self.video = cv2.VideoCapture(0)
        #self.video = cv2.VideoCapture(1)
        #self.video.set(PROP_FRAME_WIDTH, 640)
        #self.video.set(PROP_FRAME_HEIGHT, 480)
        self.video.set(PROP_FRAME_WIDTH, 320)
        self.video.set(PROP_FRAME_HEIGHT, 240)

    def __del__(self):
        self.video.release()

    def get_frame(self):
        success, image = self.video.read()
        ret, jpeg = cv2.imencode('.jpg', image)
        return jpeg.tostring()
```



# 3. MJPG on PI

- Watch video



**Hello Stream**



No stream? You might need: **sudo modprobe bcm2835-v4l2**