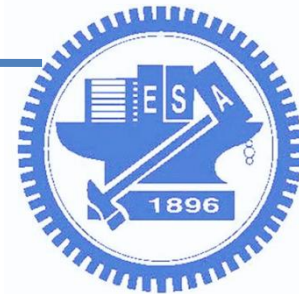




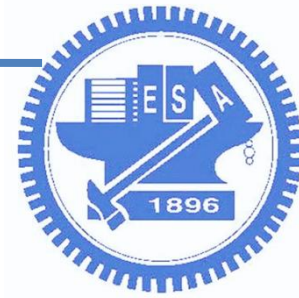
5. PI camera

National Chiao Tung University



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 μm
- ❑ 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

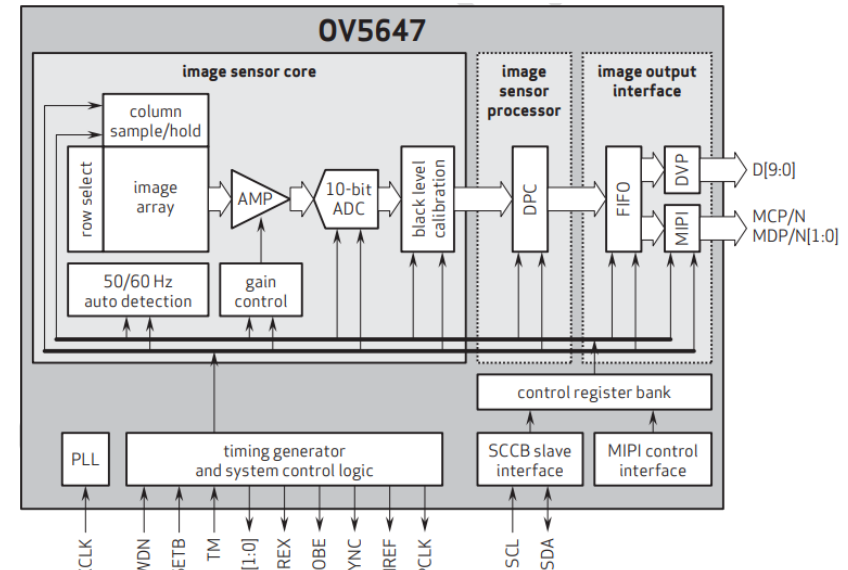
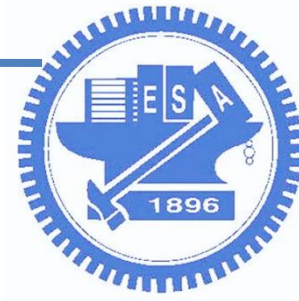


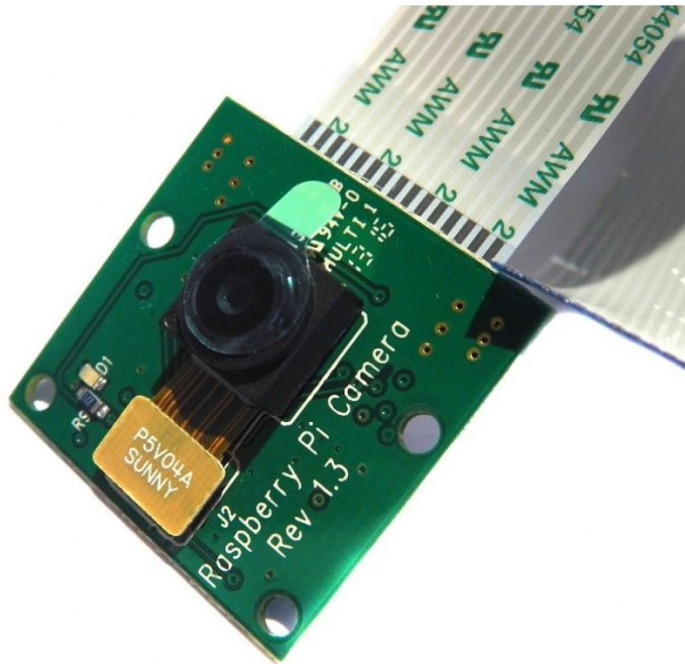
table 2-1 format and frame rate

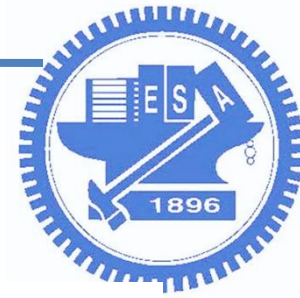
format	resolution	frame rate	scaling method	pixel clock
5 Mpixel	2592x1944	15 fps	full resolution	80 MHz
1080p	1920x1080	30 fps	cropping	68 MHz
960p	1280x960	45 fps	cropping, subsampling/ binning	91.2 MHz
720p	1280x720	60 fps	cropping, subsampling/ binning	92 MHz
VGA	640x480	90 fps	cropping, subsampling/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



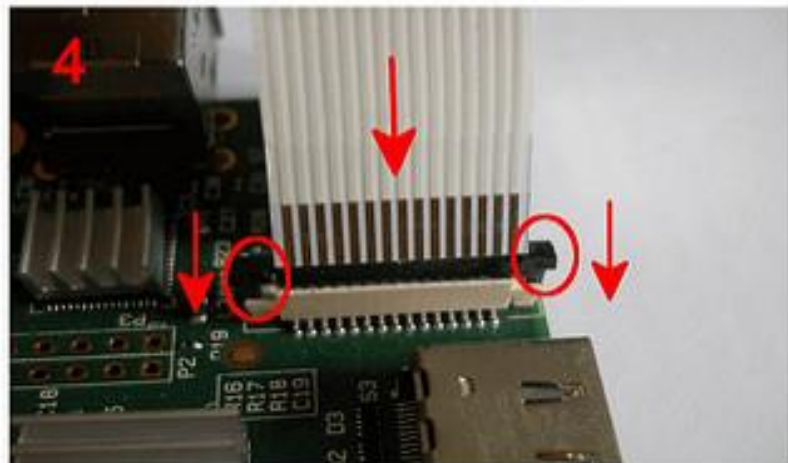
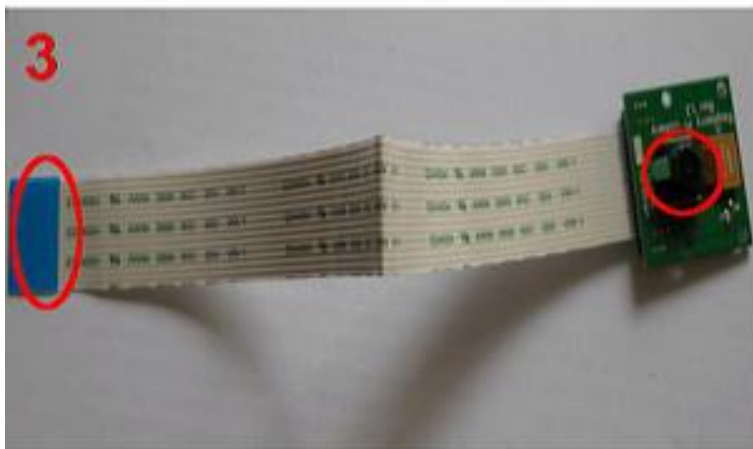
Install PI camera

15-Pins, CSI interface





Install PI camera





(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 hardware

6 Overclock Configure overclocking for your processor

7 Advanced Options Configure advanced settings

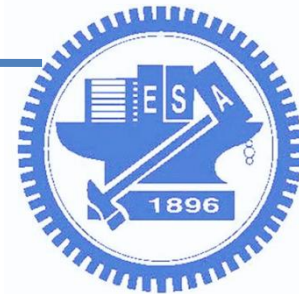
8 Update Update this tool to the latest version

9 About raspi-config Information about this configuration tool

<Select> <Finish>



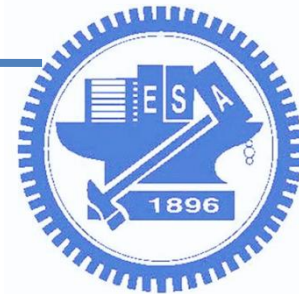
-
- The screenshot shows a terminal window titled "(COM8) [80x24]" with standard Windows window controls. The menu bar includes "連線(C)", "編輯(E)", "檢視(V)", "視窗(W)", "選項(O)", and "說明(H)". The main content area displays the "Raspberry Pi Software Configuration Tool (raspi-config)" interface. It lists eight options: P1 Camera, P2 SSH, P3 VNC, P4 SPI, P5 I2C, P6 Serial, P7 1-Wire, and P8 Remote GPIO. Each option has a description of its function. The "P1 Camera" option is currently selected and highlighted. At the bottom of the screen, there are navigation prompts "<Select>" and "<Back>". The entire interface is rendered in a monospaced font on a black background.



Raspbian configuration

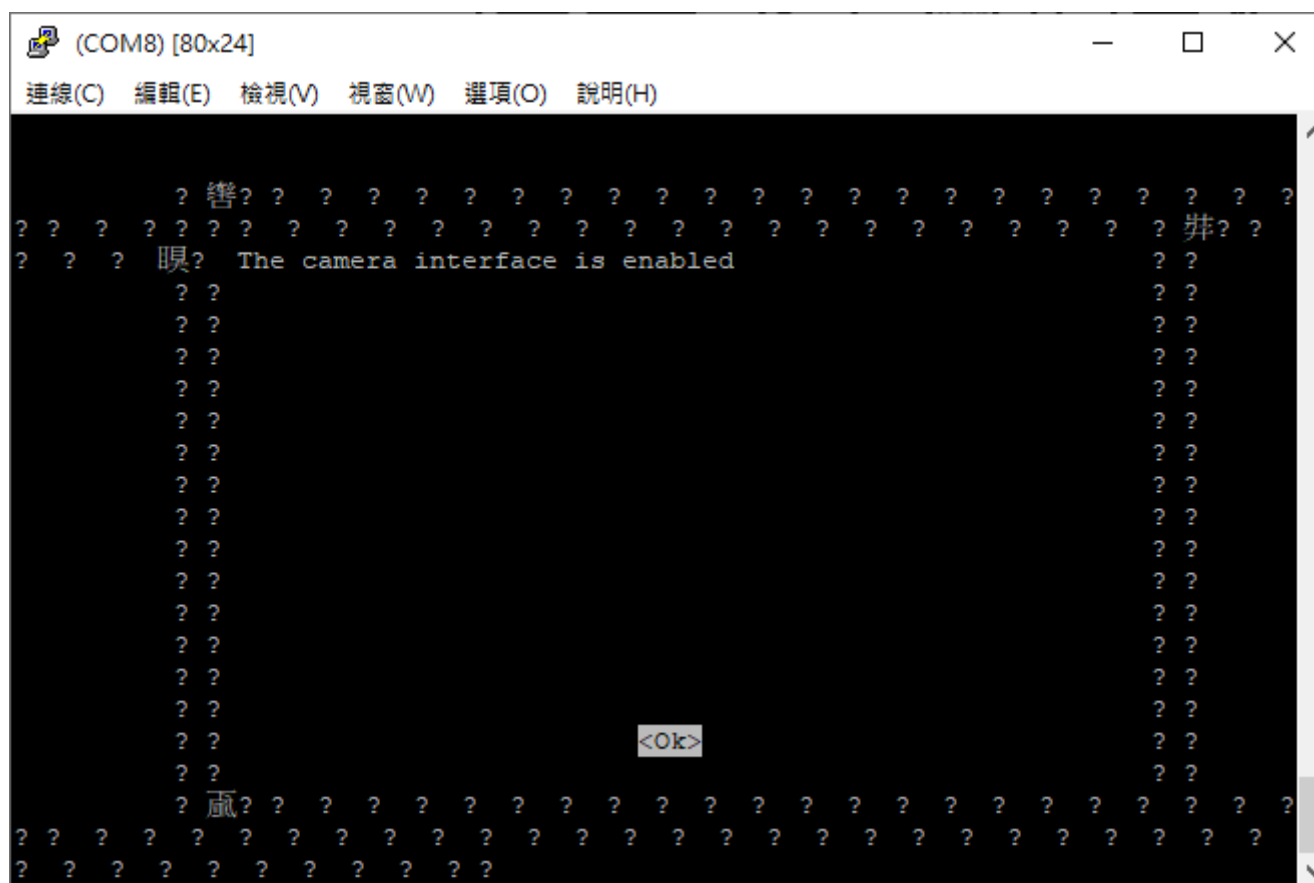
- sudo raspi-config -> P1 Camera -> Enable



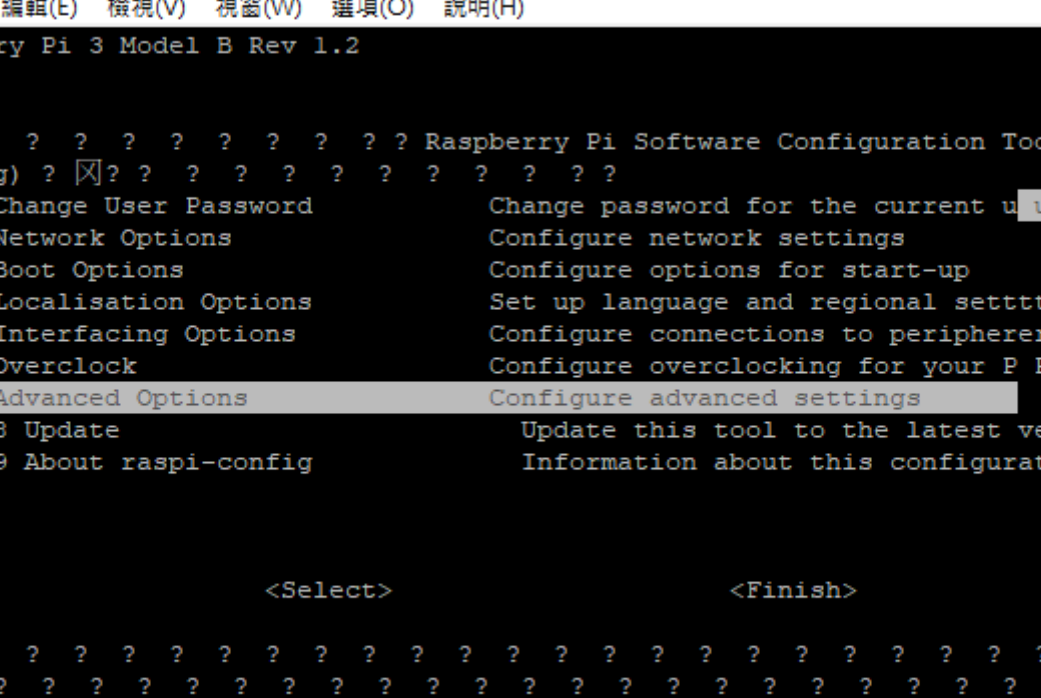


Raspbian configuration

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





- 
- ```

(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 peripherals
6 Overclock Configure overclocking for your Pi
7 Advanced Options Configure advanced settings
8 Update Update this tool to the latest version
9 About raspi-config Information about this configuration tool

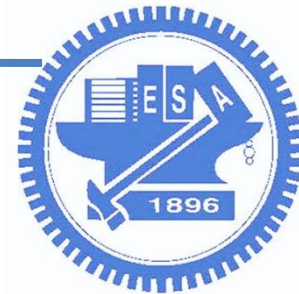
<Select> <Finish>

```

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 two question marks "? ?":

- A1 Expand Filesystem: Ensures that all of the SD card space is available.
- A2 Overscan: You may need to configure overscanning if you have a monitor.
- A3 Memory Split: Change the amount of memory made available to the GPU.
- A4 Audio: Force audio out through HDMI or 3.5mm jack.
- A5 Resolution: Set a specific screen resolution.
- A6 Pixel Doubling: Enable/Disable 2x2 pixel mapping.
- A7 GL Driver: Enable/Disable experimental desktop graphics driver.

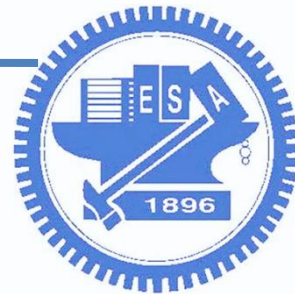
At the bottom, there are navigation options "<Select>" and "<Back>". The terminal output is displayed in a monospaced font on a black background.



# 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

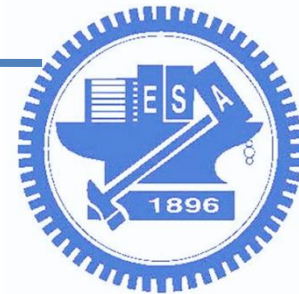
## □ Record a video: raspivid

□ 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:EN
OMEM)
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 e
nsure that "camera" has been enabled
```

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





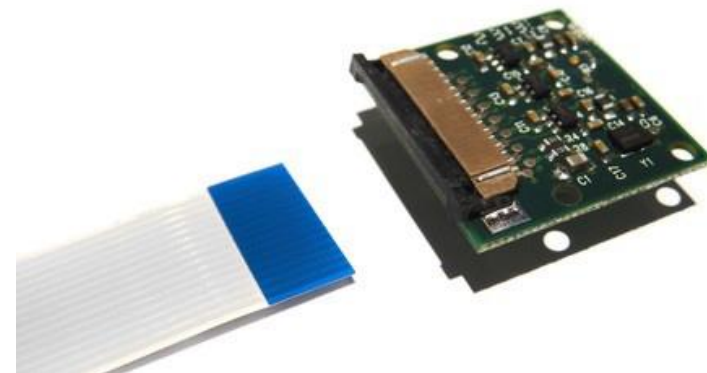
# Error message?

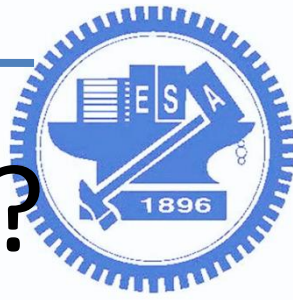
## □ Msg: Camera is not detected

```
(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

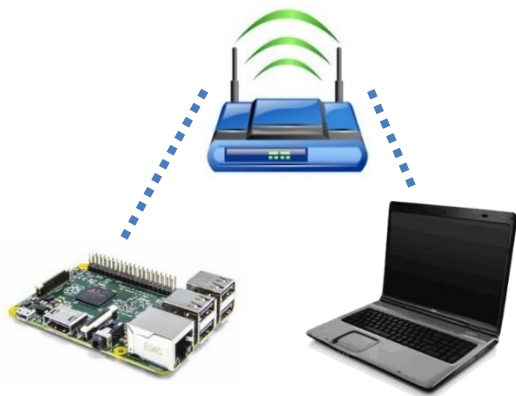




# How to view image/video?

## □ Methods:

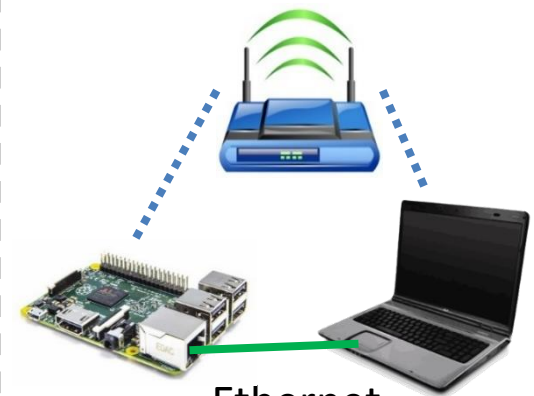
1. `python -m SimpleHTTPServer 8000`
2. `winscp`
3. `vnc`
4. HDMI



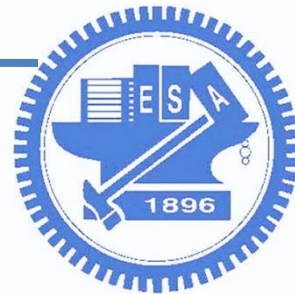
Wi-Fi AP



Wi-Fi hotspot



Wired + Wireless



# Python code

- Sample code for taking a picture

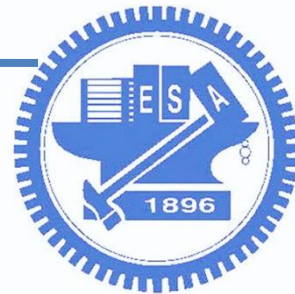
```
import picamera
import time

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
```

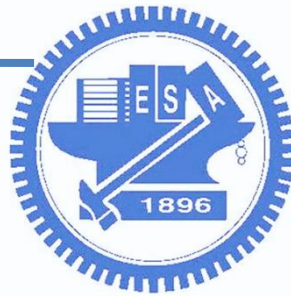
```
camera = picamera.PiCamera()
camera.start_recording('video.h264')
camera.wait_recording(3)
camera.stop_recording()
```

```
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.

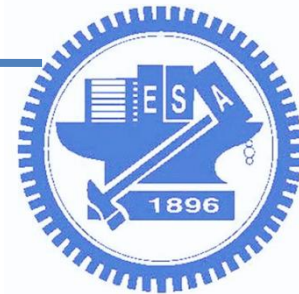


# Python code

- 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



# Discussion

- Read the online document. If we want to set the output **file name as data and time**, how do we set filename in the code?

- Ex: image20190403\_1720.jpg

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

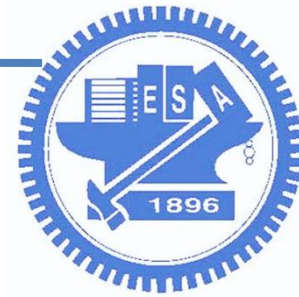
Capture images continuously from the camera as an infinite iterator.

This method returns an infinite iterator of images captured continuously from the camera. If *output* is a string, each captured image is stored in a file named after *output* after substitution of two values with the `format()` method. Those two values are:

- `{counter}` - a simple incrementor that starts at 1 and increases by 1 for each image taken
- `{timestamp}` - a `datetime` instance

- Original: `camera.capture_continuous('image{counter:02d}.jpg')):`
- New: ????????????????

Hint: <https://docs.python.org/2/library/datetime.html>



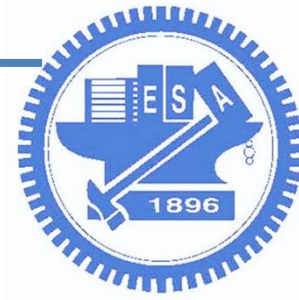
# Discussion

- Read the online document. If we want to set the output **file name as data and time**, how do we set filename in the code?

|    |                                                                   |
|----|-------------------------------------------------------------------|
| %w | Weekday as a decimal number, where 0 is Sunday and 6 is Saturday. |
| %d | Day of the month as a zero-padded decimal number.                 |
| %b | Month as locale's abbreviated name.                               |
| %B | Month as locale's full name.                                      |
| %m | Month as a zero-padded decimal number.                            |
| %y | Year without century as a zero-padded decimal number.             |
| %Y | Year with century as a decimal number.                            |
| %H | Hour (24-hour clock) as a zero-padded decimal number.             |
| %I | Hour (12-hour clock) as a zero-padded decimal number.             |

**Hint: timestamp**



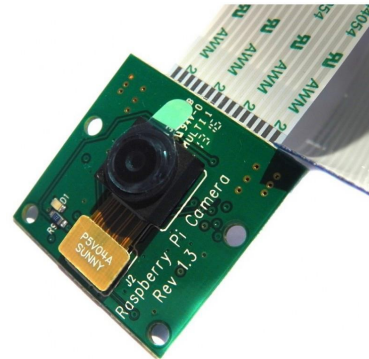


# Quiz 1

- Automatically sunrise timelapse pictures
  - Execute the code, then take a series pictures at a specific time.
  - You might need “schedule” module.



AM 08:00, start capturing...

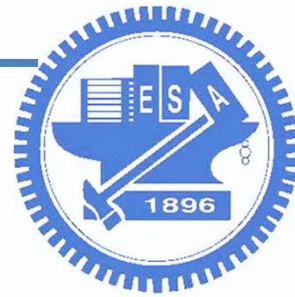


001.jpg

002.jpg

.

010.jpg



# 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)
```

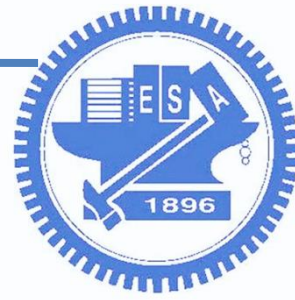
**at**(*time\_str*)

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]



# Quiz 2

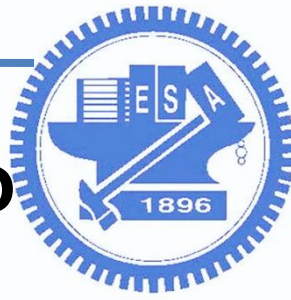
- Take a picture with UNIX timestamp. (Ex: 1554282107)
  - Hint: read the online document and understand the definition of parameters

```
import picamera
import time

with picamera.PiCamera() as camera:
 camera.resolution = (640, 480)
 camera.framerate = 24
 camera.start_preview()
 camera.annotate_text = 'Hello world!'
 time.sleep(2)
 # Take a picture including the annotation
 camera.capture('foo.jpg')
```

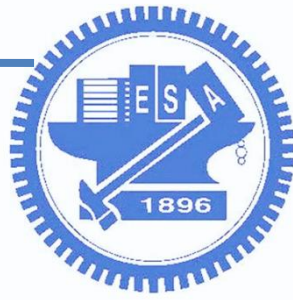
Put message on picture





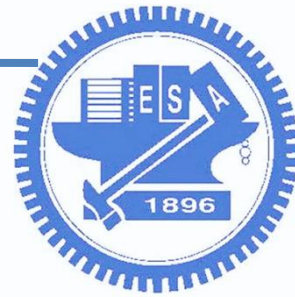
# What is UNIX timestamp?

*The unix time stamp is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. Therefore, the unix time stamp is merely the number of seconds between a particular date and the Unix Epoch. It should also be pointed out (thanks to the comments from visitors to this site) that this point in time technically does not change no matter where you are located on the globe. This is very useful to computer systems for tracking and sorting dated information in dynamic and distributed applications both online and client side.*



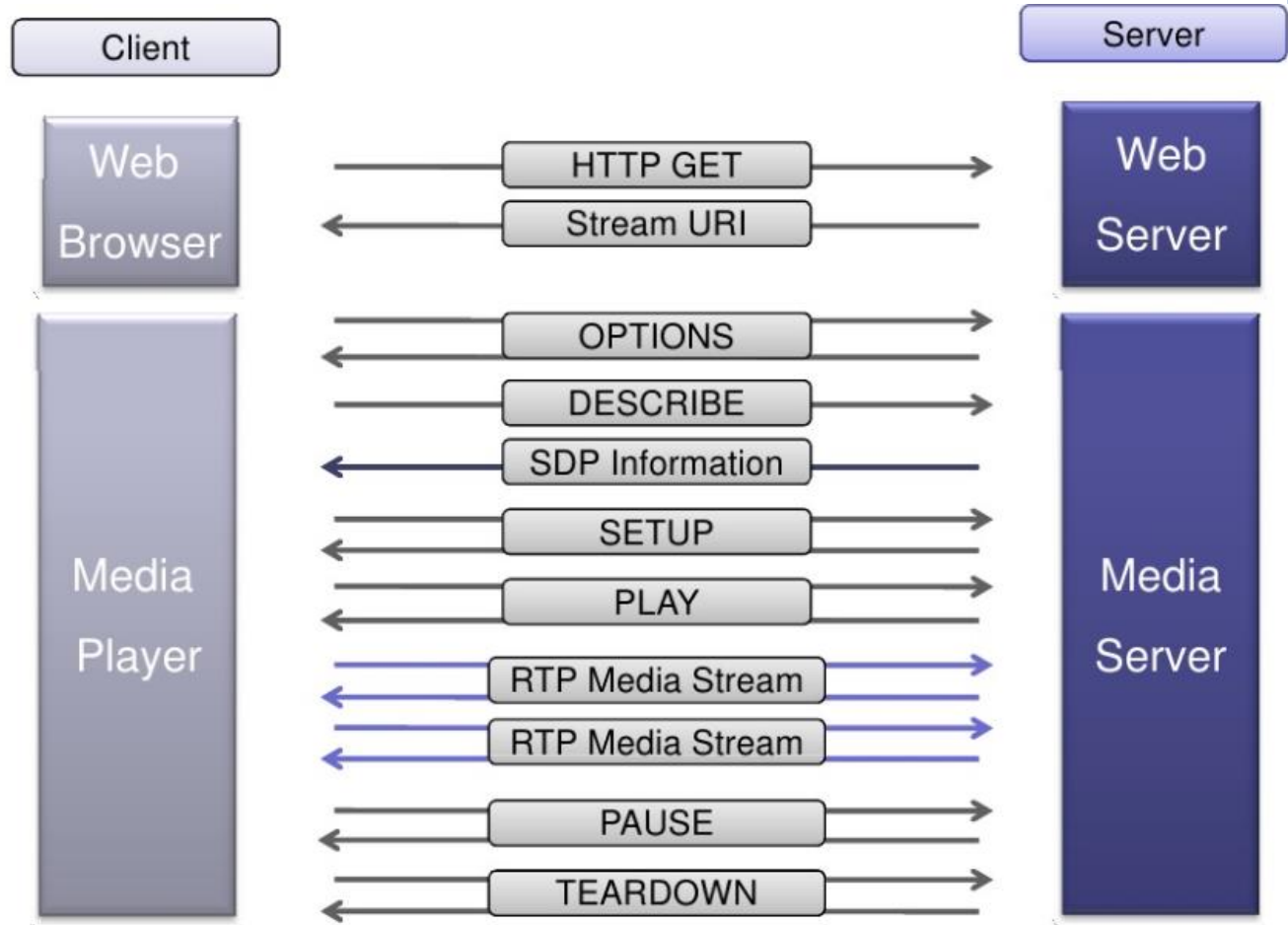
# IP cam

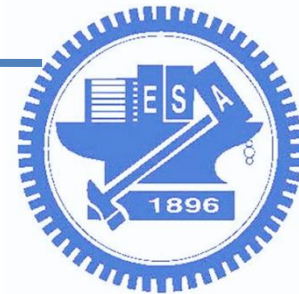
- Video Streaming
  - 使用 RTSP + H.264
  - 使用 HTTP + MJPG
  - 使用 RTMP



# 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

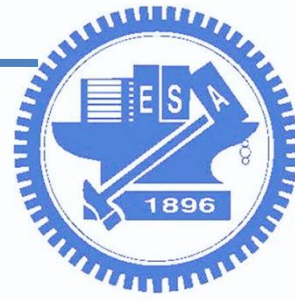
- Execute the command

```
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=;

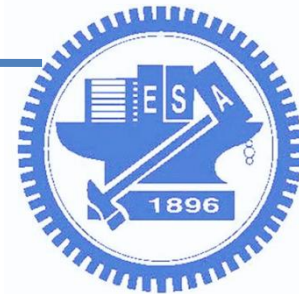
[75400520] main input debug: Buffering 66%
[75400520] main input debug: Buffering 73%
[75400520] main input debug: Buffering 80%
[75400520] main input debug: Buffering 86%
[75400520] main input debug: Buffering 93%
[75400520] main input debug: Buffering 100%
[75400520] main input debug: Stream buffering done (320 ms in 335 ms)
[75400520] main input debug: Decoder wait done in 0 ms
```





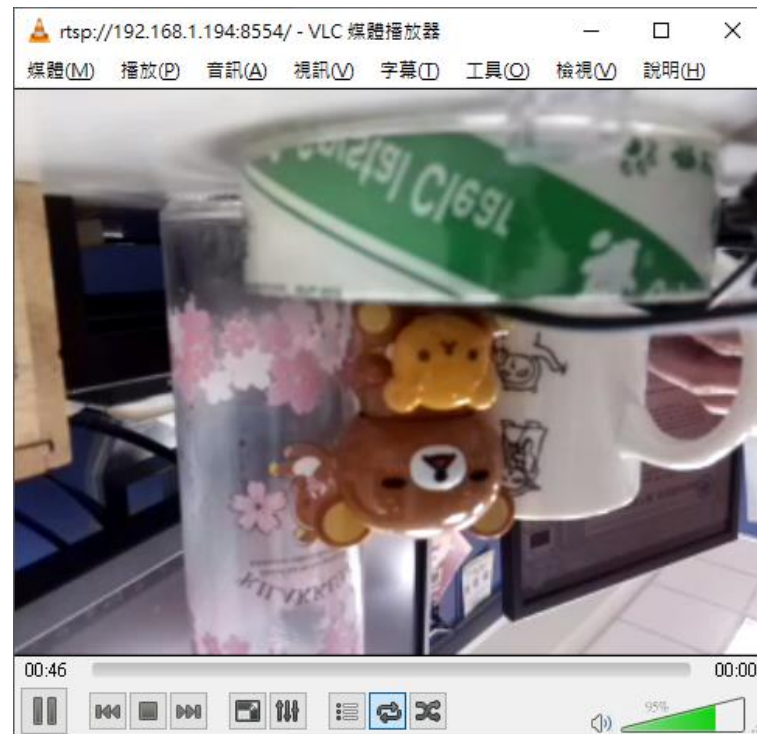
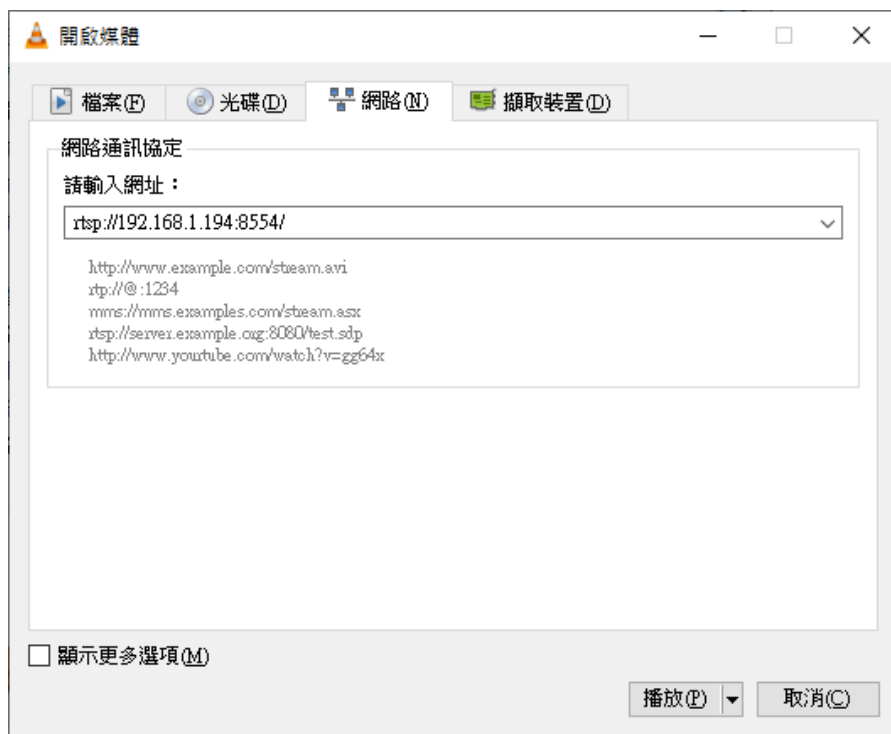
# 1. RTSP on Raspberry Pi

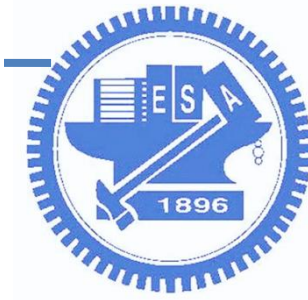
- `cvlc -vvv stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554}' :demux=h264`
  - ▣ `stream:` Stream MRL syntax: `[[access][[/demux]://]URL[#[title][:chapter][-title][:chapter]]] [:option=value ...]`
  - ▣ `/dev/stdin:` Standard input. The source of input data for command line programs. Here, the input 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

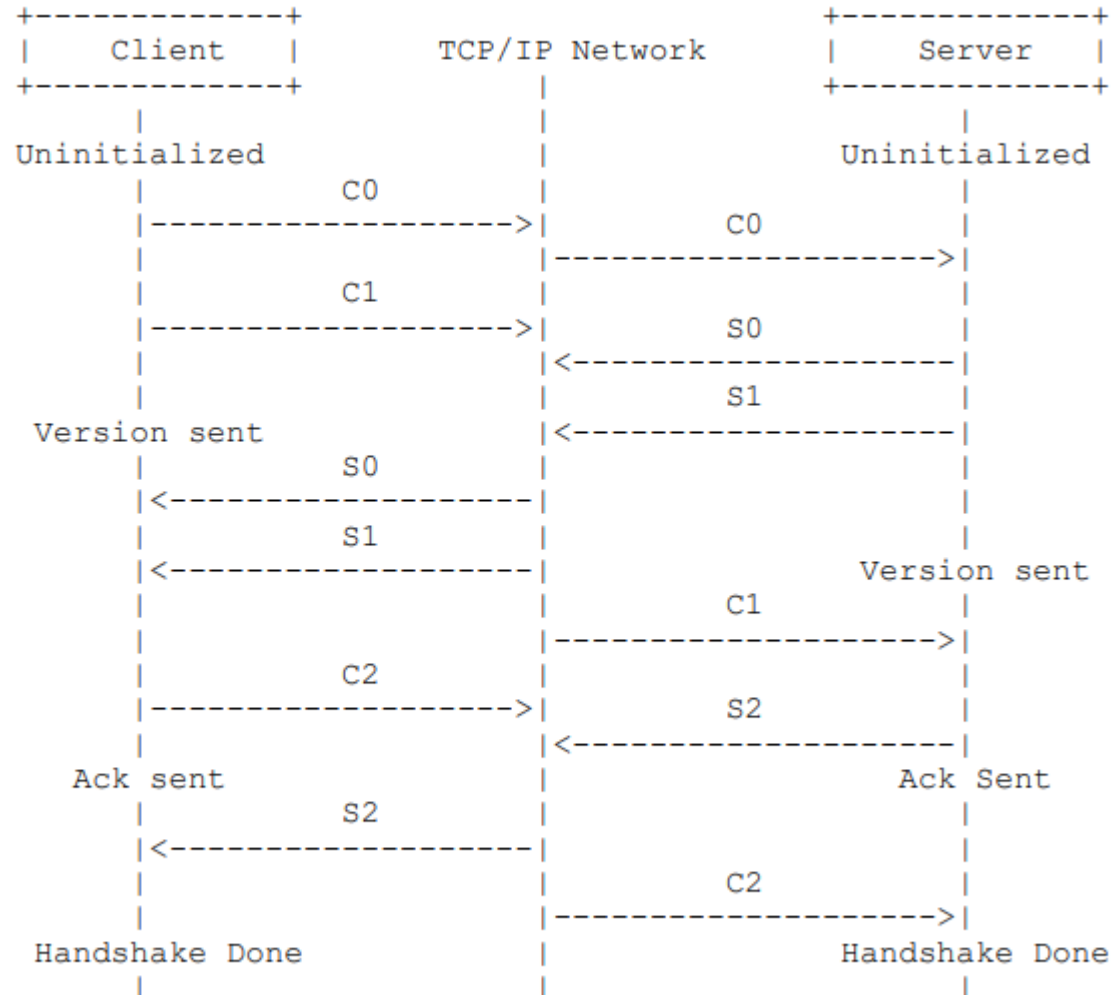
## □ Use VLC to watch video



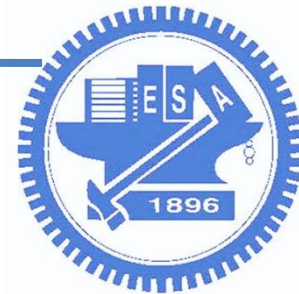


## 2. RTMP

### 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

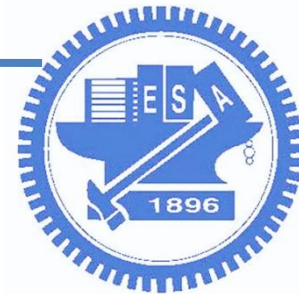
rtmp://a.rtmp.youtube.com/live2

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)

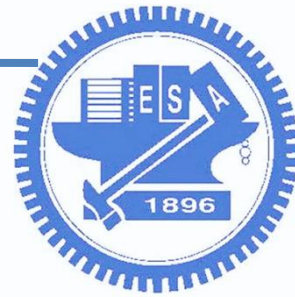
Stream #1:0: Video: h264 (High), yuv420p(progressive), 1920x1080, 25 fps, 25 tbr, 1200k tbn, 50 tbc
Output #0, flv, to 'rtmp://a.rtmp.youtube.com/live2/':
Metadata:
 encoder : Lavf57.56.101
Stream #0:0: Video: h264 (High) ([7][0][0][0] / 0x0007), yuv420p(progressive), 1920x1080, q=2-31, 25 fps, 25 tbr, 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

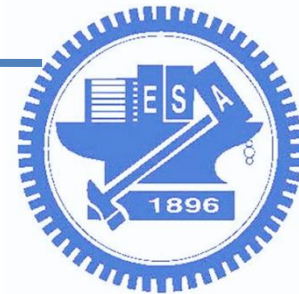
串流名稱/金鑰

.....



## 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...

The screenshot shows a video player interface with a live stream status bar at the top. The status bar includes a red dot icon, the text "直播中" (Live), and a question mark icon. Below this, it says "串流狀況" (Stream Status). The main video area is black with a white loading spinner. At the bottom of the video area, there are two buttons: "建立焦點片段" (Create Focus Segment) and "變更縮圖" (Change Thumbnail). To the right of the video area, there is a terminal window titled "(COM8) [80x24]". The terminal window displays a list of streaming statistics for each frame, including frame number, fps, q, size, time, and bitrate. The statistics show a consistent stream with a bitrate around 1260-1295 kbps and a frame size around 1600-2200 KB. The terminal window also shows a menu bar with options: 連線(C), 編輯(E), 檢視(V), 視窗(W), 選項(O), and 說明(H). At the bottom of the terminal window, it shows "speed=0.632x".

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]

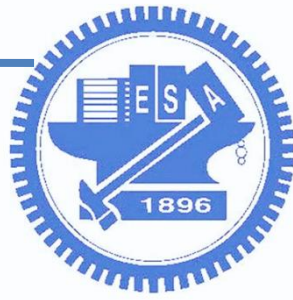
連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

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

speed=0.632x

建立焦點片段 變更縮圖

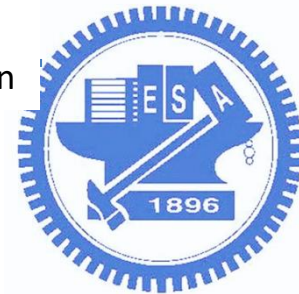




## 2. RTMP on PI

- Watch video



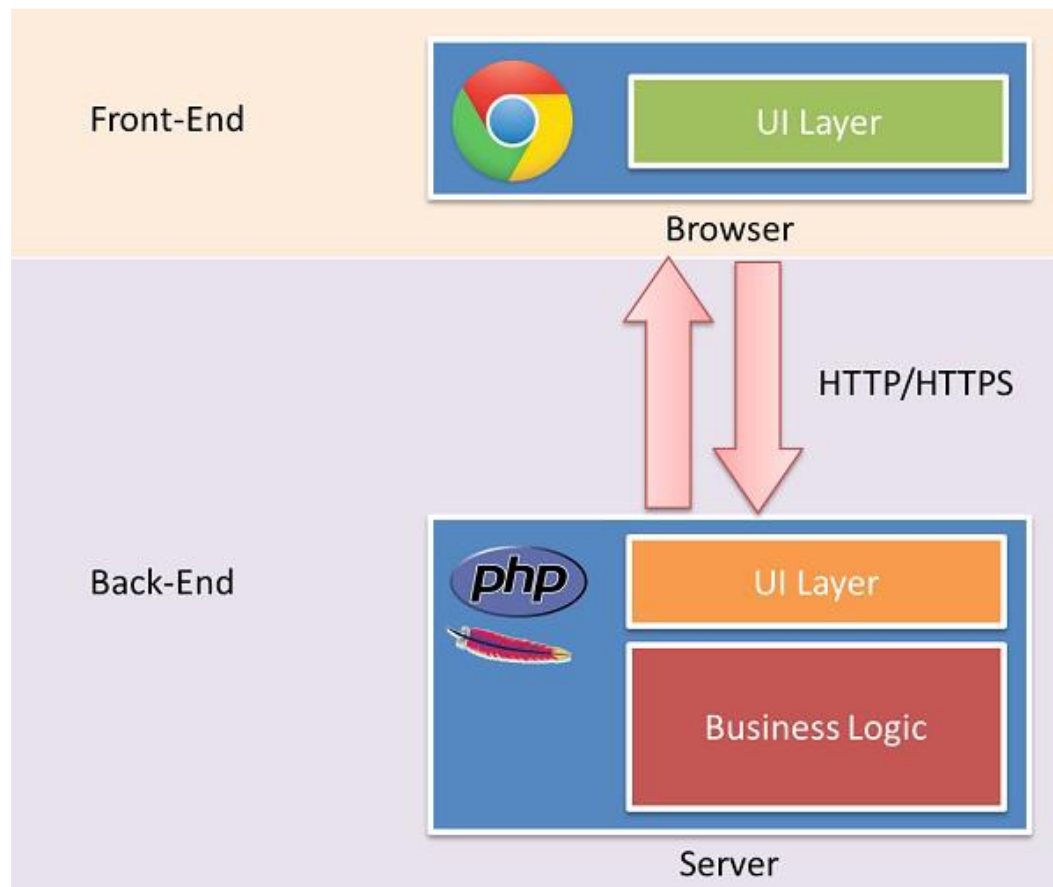


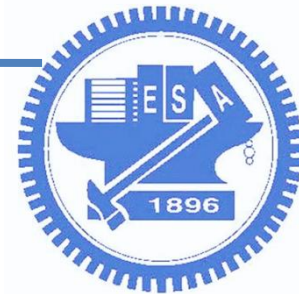
## 3. HTTP + MJPG

### □ 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。

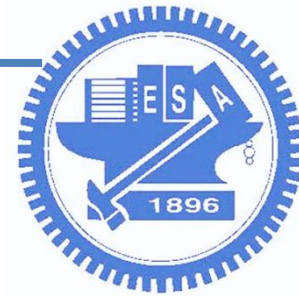




# 3. HTTP + MJPG on PI

- Install tools:
  - `sudo pip install request flask numpy`
  - `sudo modprobe bcm2835-v4l2`
  - Download and unzip “5\_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__)

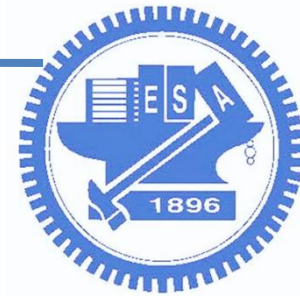
@app.route('/')
def index():
 return render_template('stream.html')

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

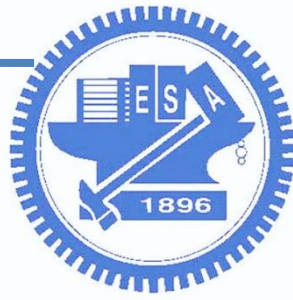
```
import cv2

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

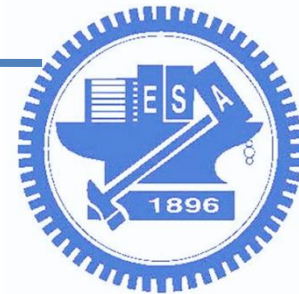
- Watch video



**Hello Stream**

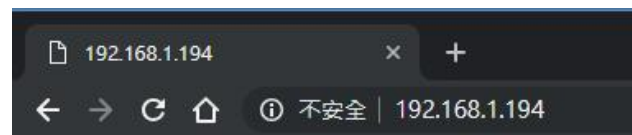


No stream? You might need: **sudo modprobe bcm2835-v4l2**



# Discussion

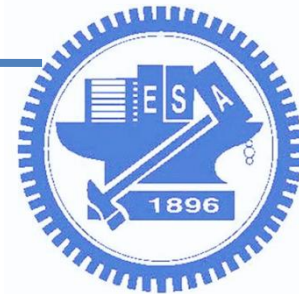
- Try to use RTSP, RTMP and MJPG to stream your camera.
  - ▣ Put screenshot to show your result.



**Hello Stream**







# Summary

- Practice Lab (PI camera, RTSP, MJPG)
- Write down the answer for discussion
  - Discussion:
    - 1. Read the online document. How do we set filename in the code?
    - 2. Show your RTSP and MJPG results
- Write code for **Quiz 1 - 2**, then **demonstrate it to TAs**
  - Quiz1: Timeslape
  - Quiz2: Camera overlay (put timestamp)