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

https://cdn.sparkfun.com/datasheets/Dev/RaspberryPi/ov5647_full.pdf

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

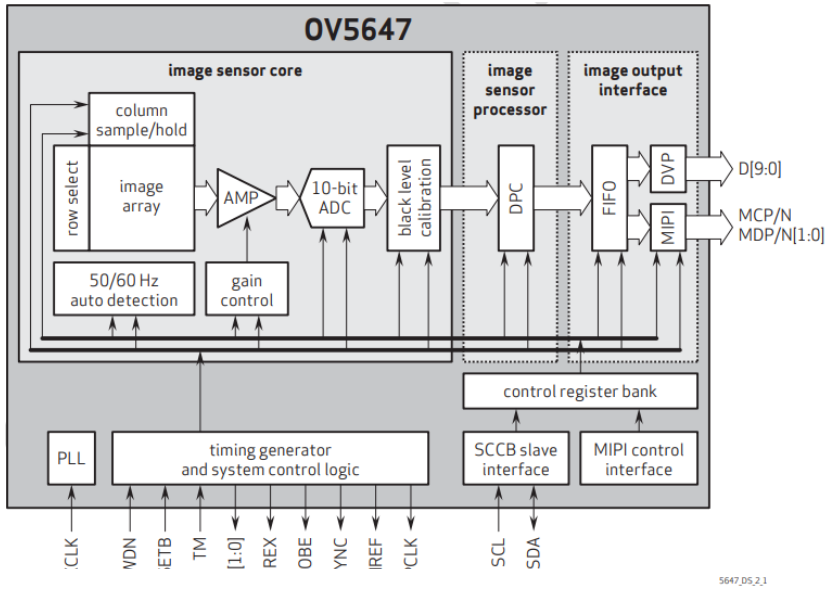
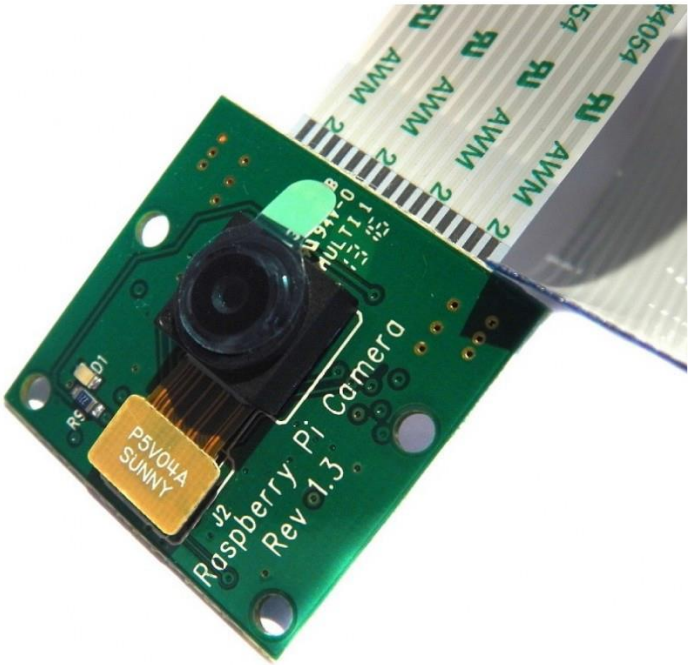


table 2-1 format and frame rate

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

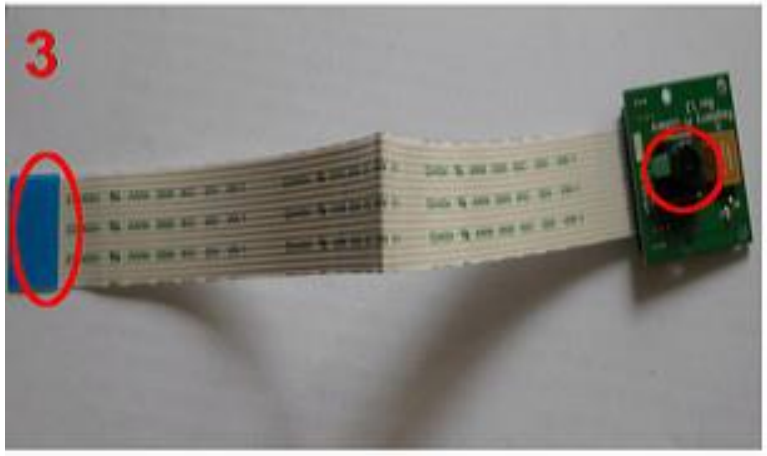
Install PI camera

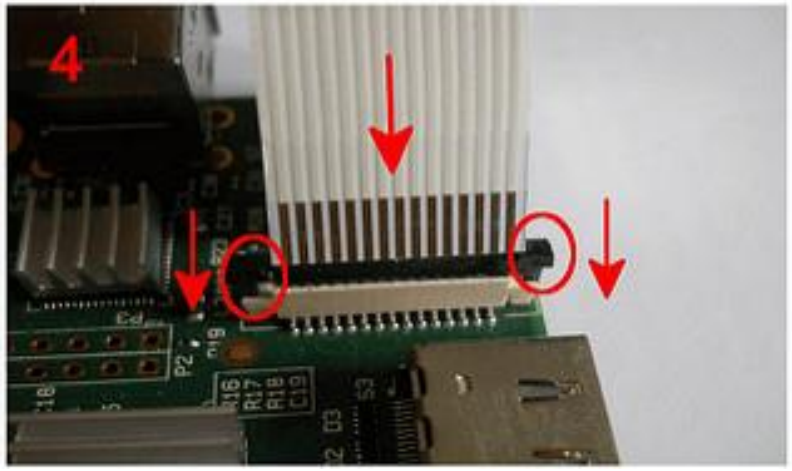
15-Pins, CSI interface



http://xmodulo.com/install-raspberry-pi-camera-board.html

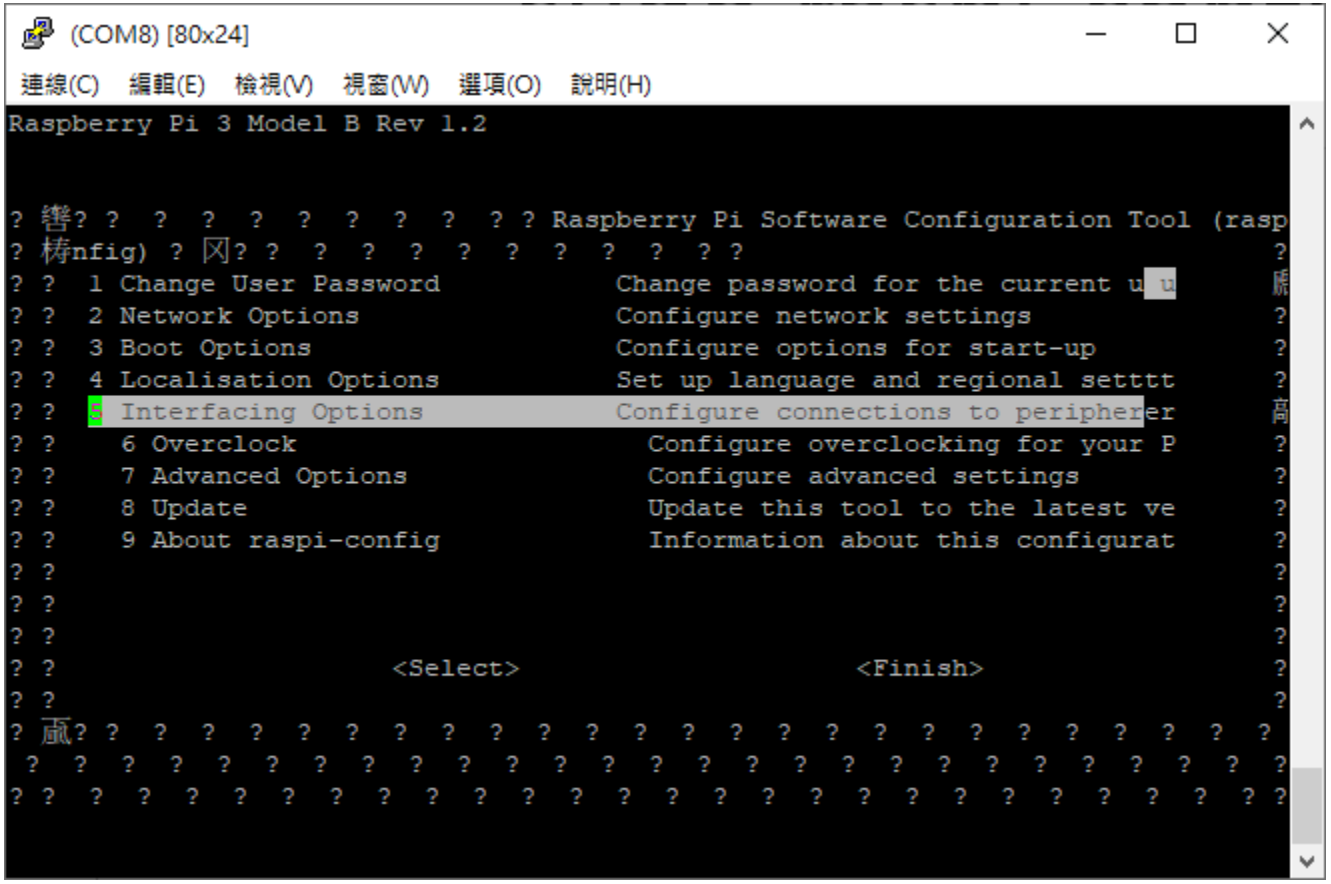
Install PI camera





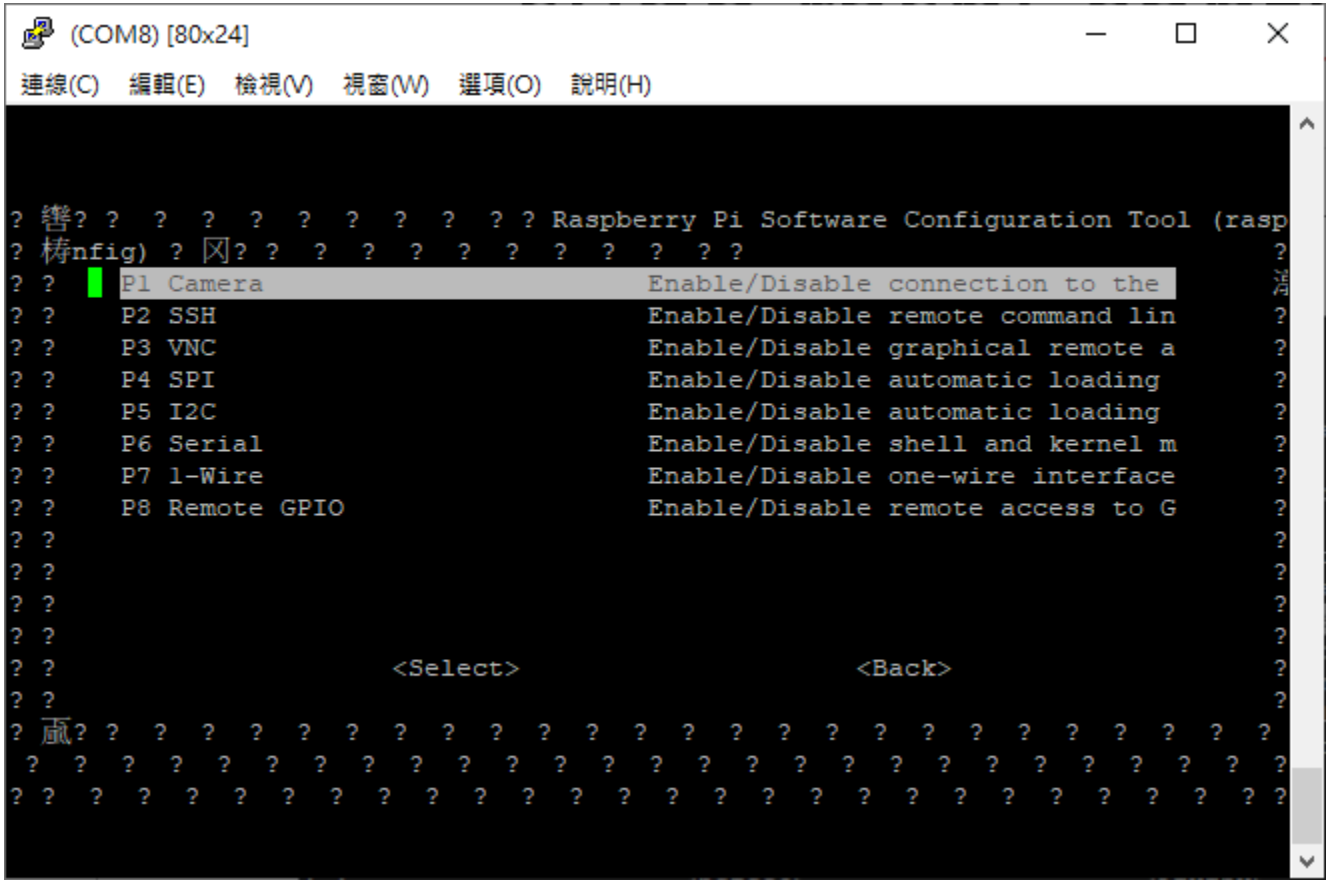
Raspbian configuration

- sudo raspi-config



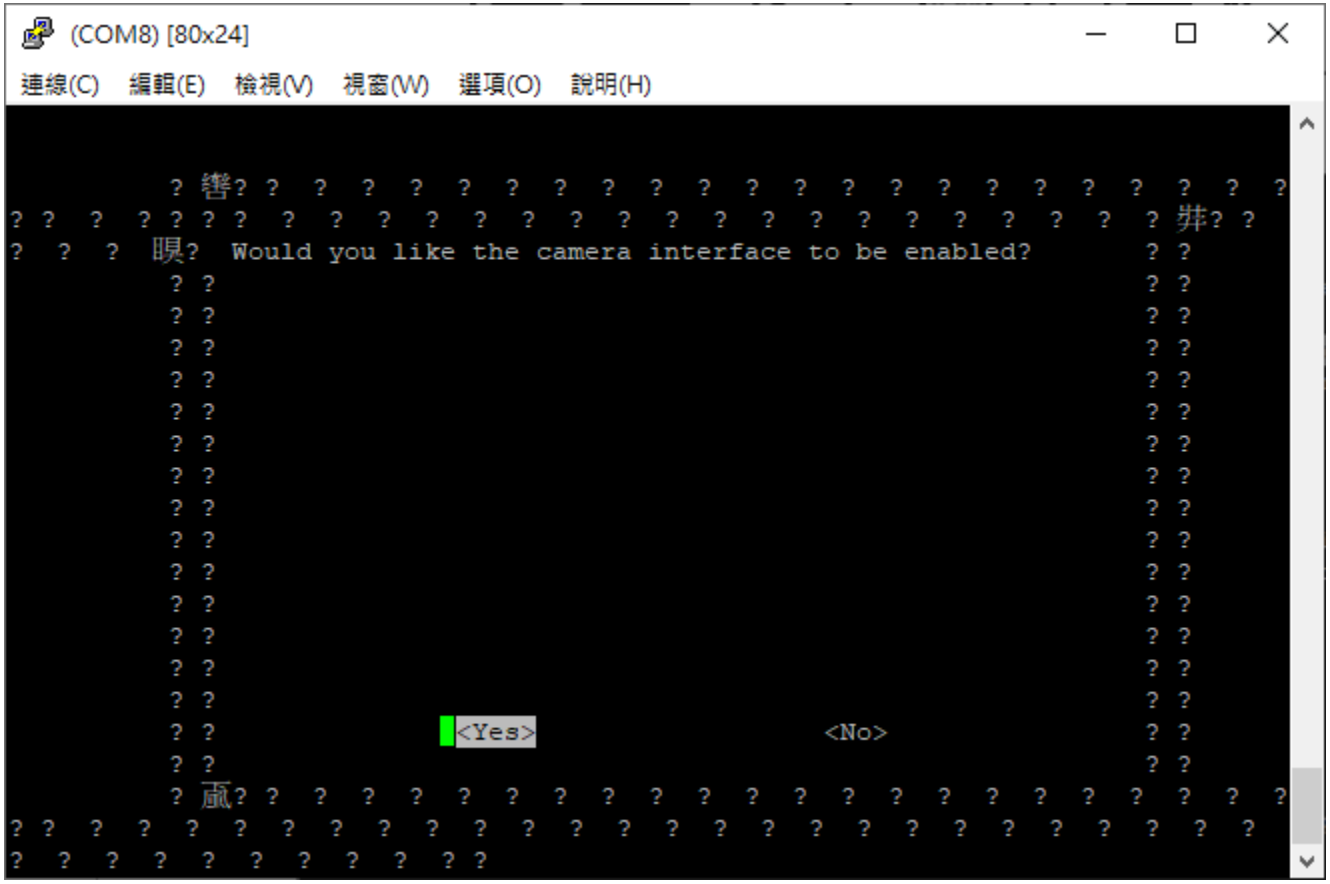
Raspbian configuration

- sudo raspi-config -> P1 Camera



Raspbian configuration

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

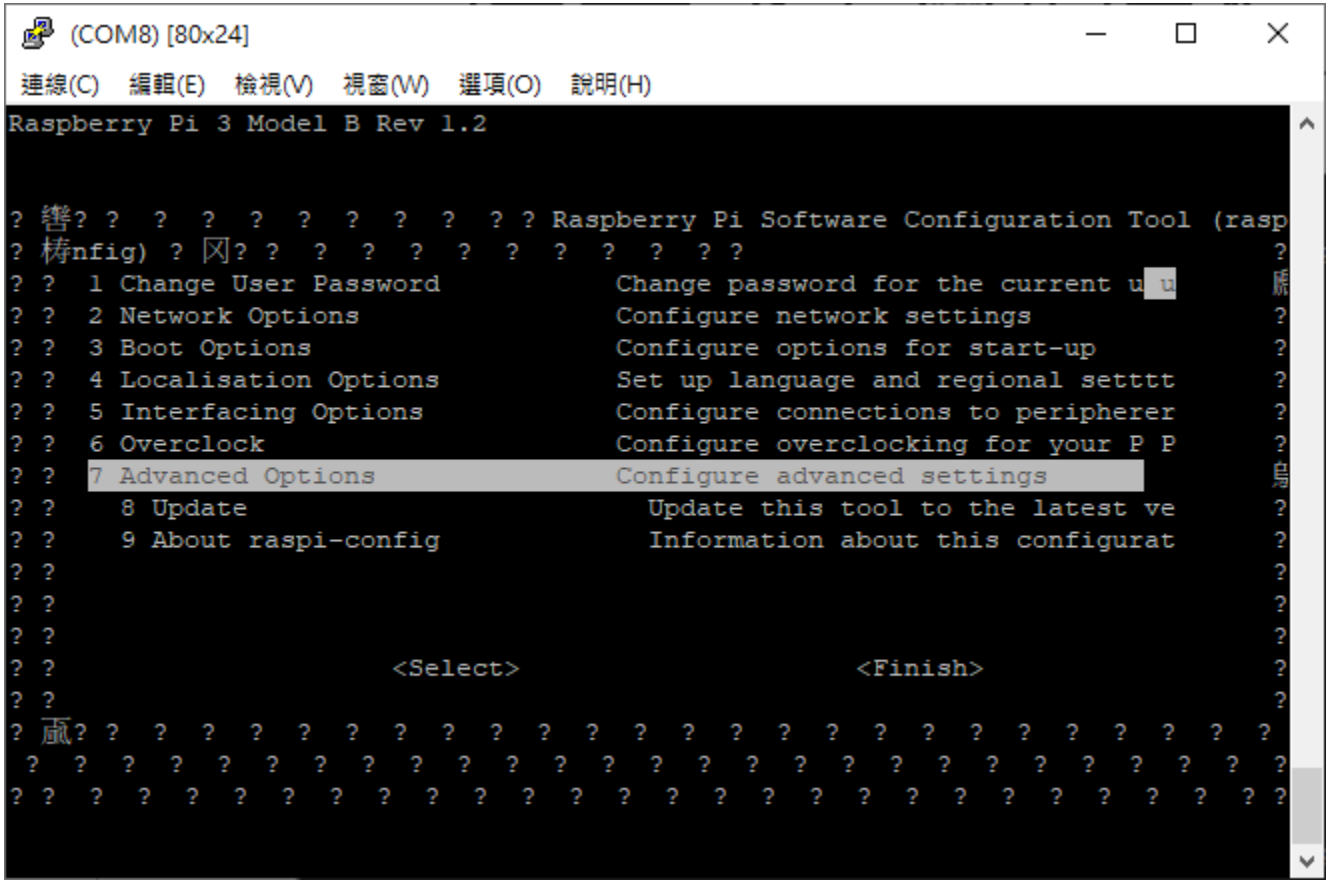


Raspbian configuration

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

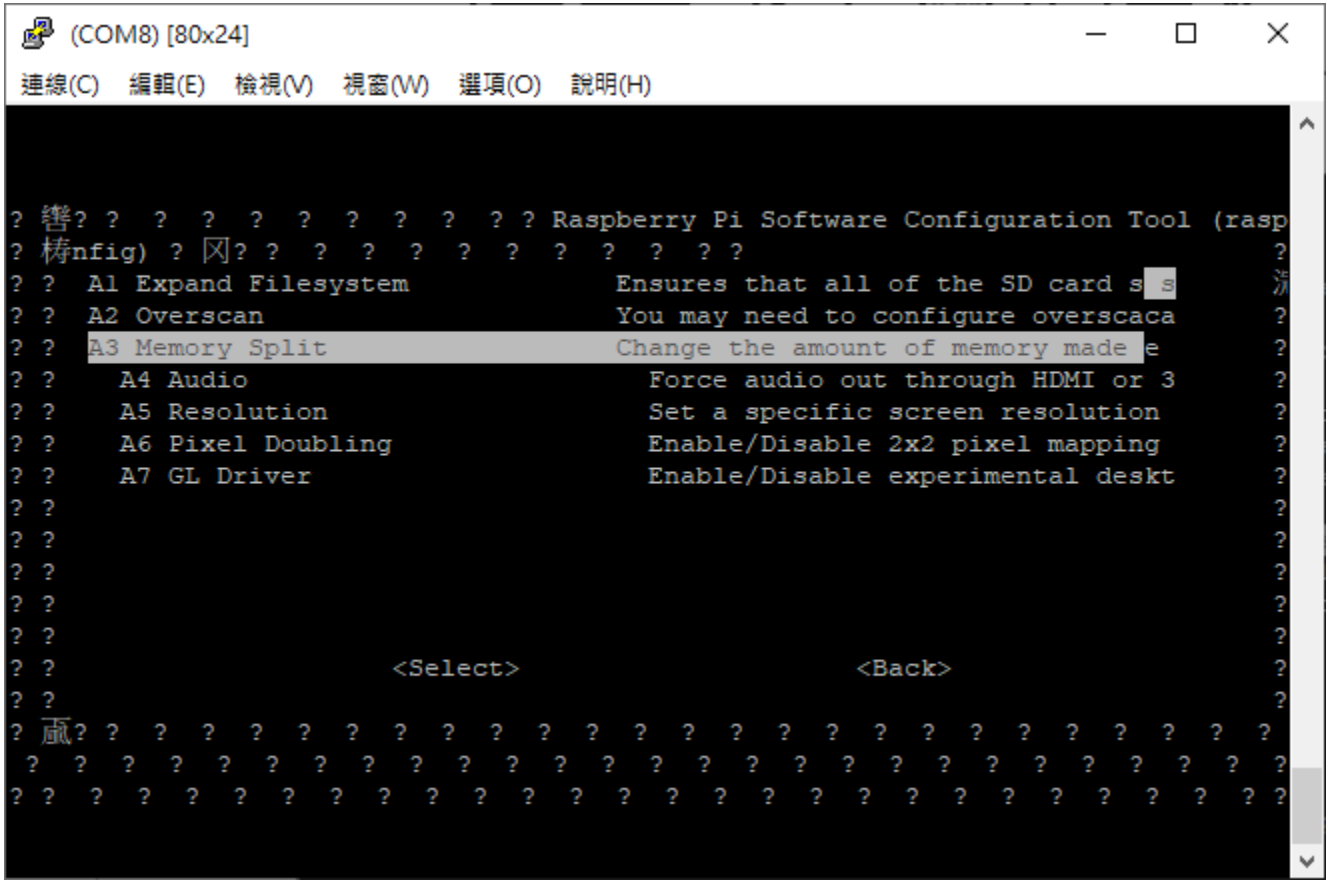
Raspbian configuration

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



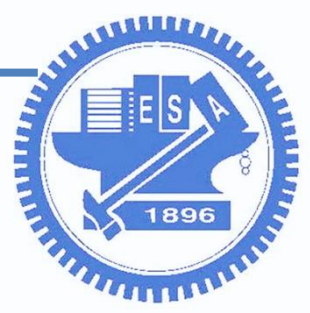
Raspbian configuration

- Optional



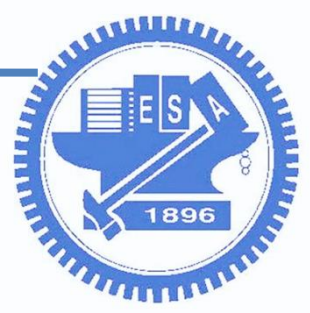
Raspbian configuration

- 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?

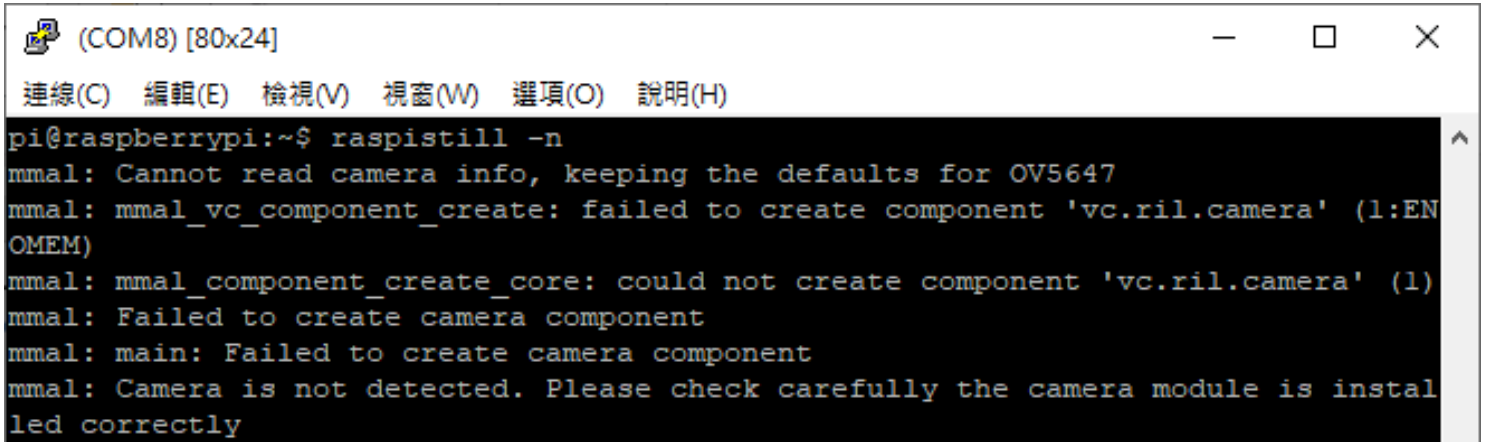
- Msg: Camera is not enabled in this build

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

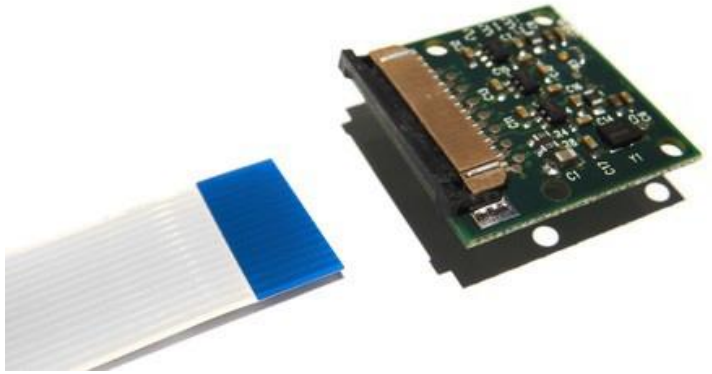
https://www.modmypi.com/blog/how-to-replace-the-raspberry-pi-camera-cable

Error message?

- Msg: Camera is not detected

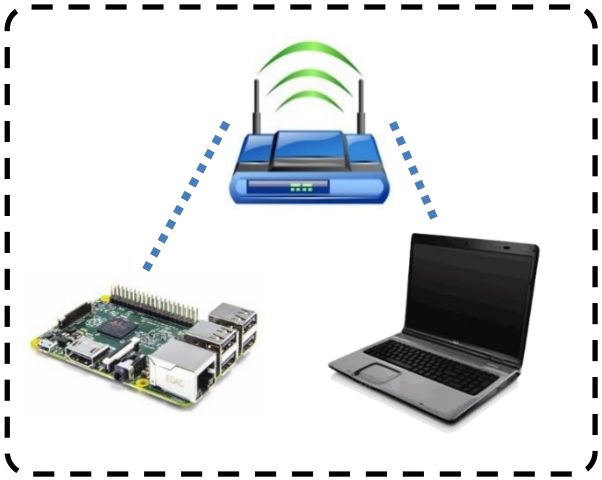


- Sol:
  - 重新安裝camera,或是更換排線
  - 或是檢查camera module是否鬆脫

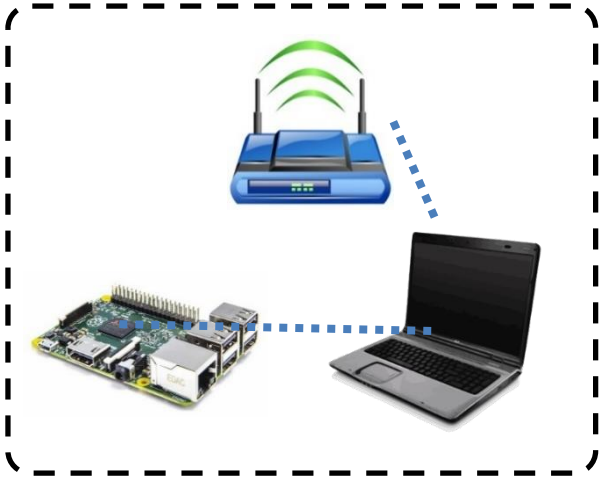


How to view image/video?

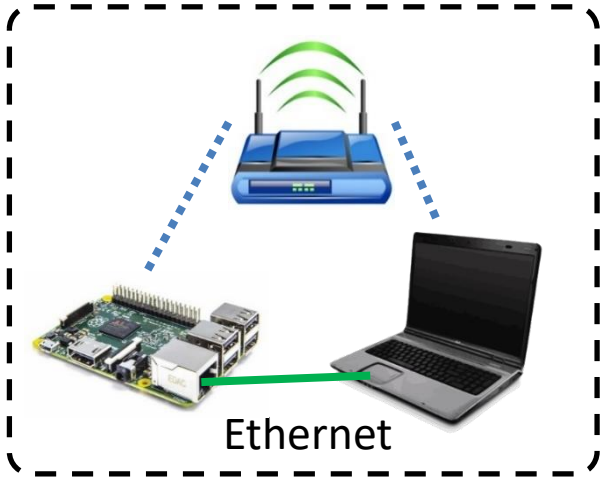
- Methods:
  1. python -m SimpleHTTPServer 8000
  2. winscp
  3. vnc
  4. HDMI



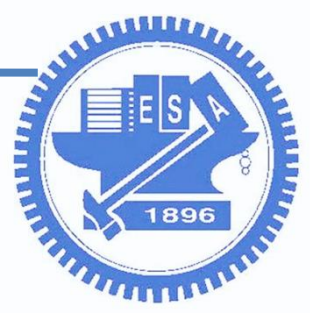
Wi-Fi AP



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]

https://picamera.readthedocs.io/en/release-1.13/api_camera.html

Python code

- Sample code for record a video

```
import picamera

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

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

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

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.

https://picamera.readthedocs.io/en/release-1.13/api_camera.html

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

https://schedule.readthedocs.io/en/stable/

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*) [source]

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

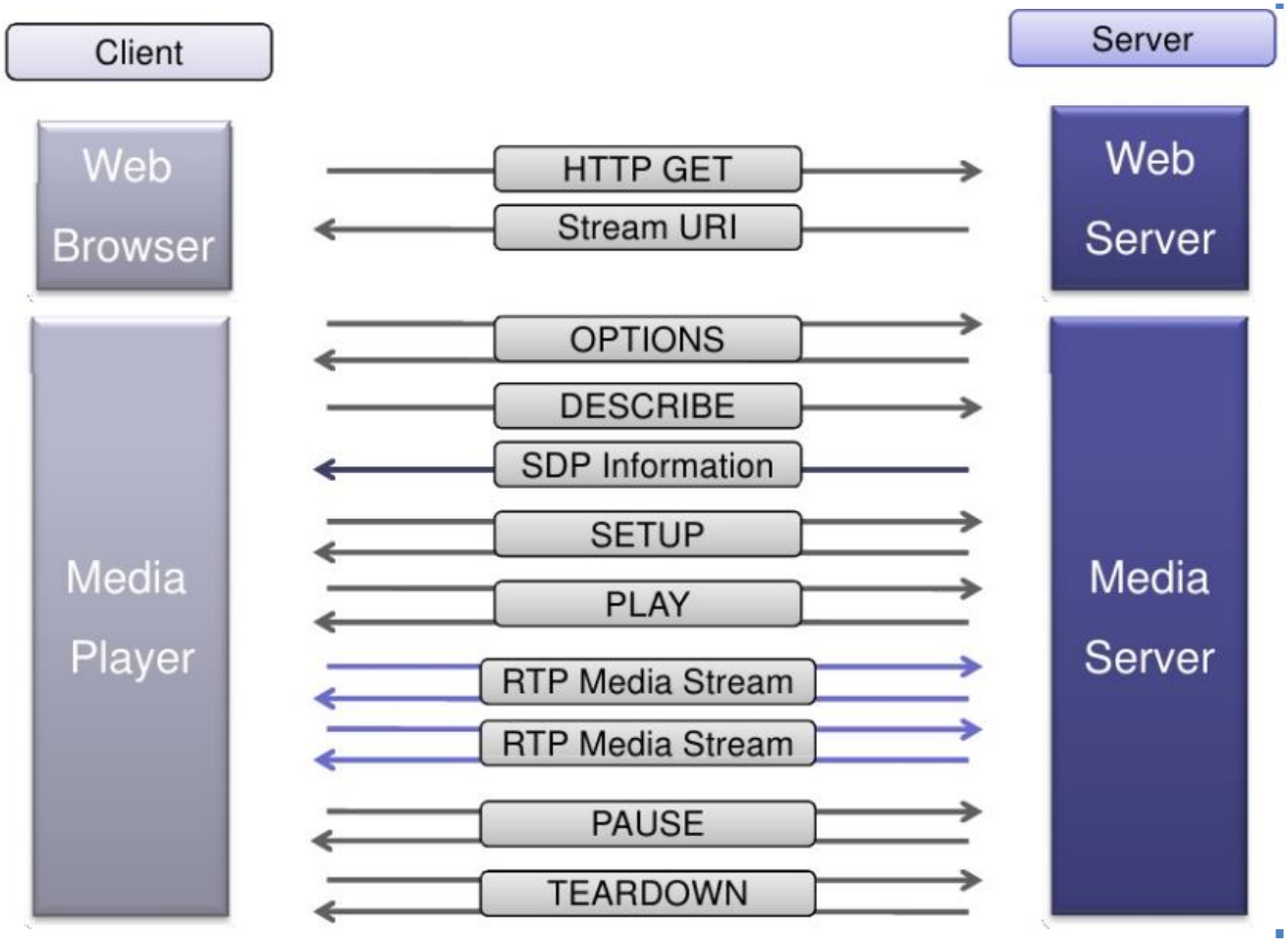
IP cam

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

https://www.slideshare.net/DSPIP/rtsp-analysis-wireshark
https://tools.ietf.org/html/rfc2326

1. RTSP

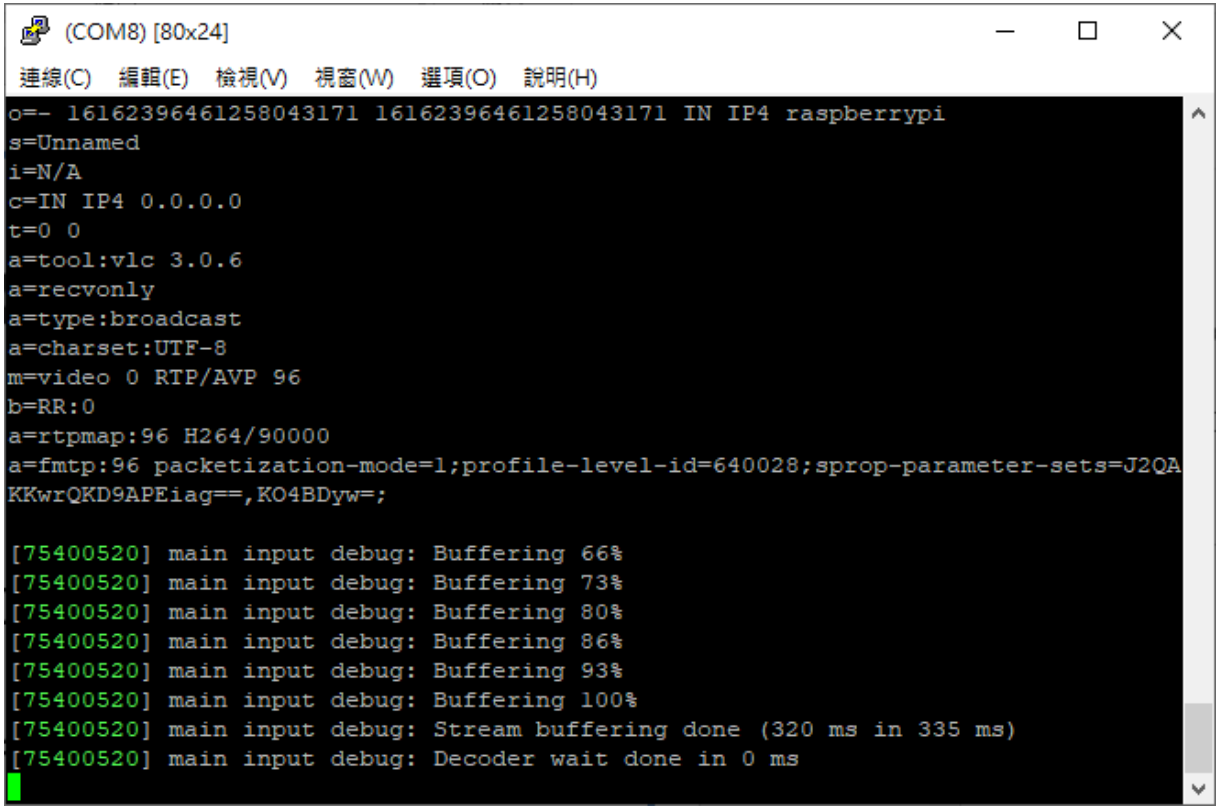
The Real Time Streaming Protocol, or RTSP, is an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live 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



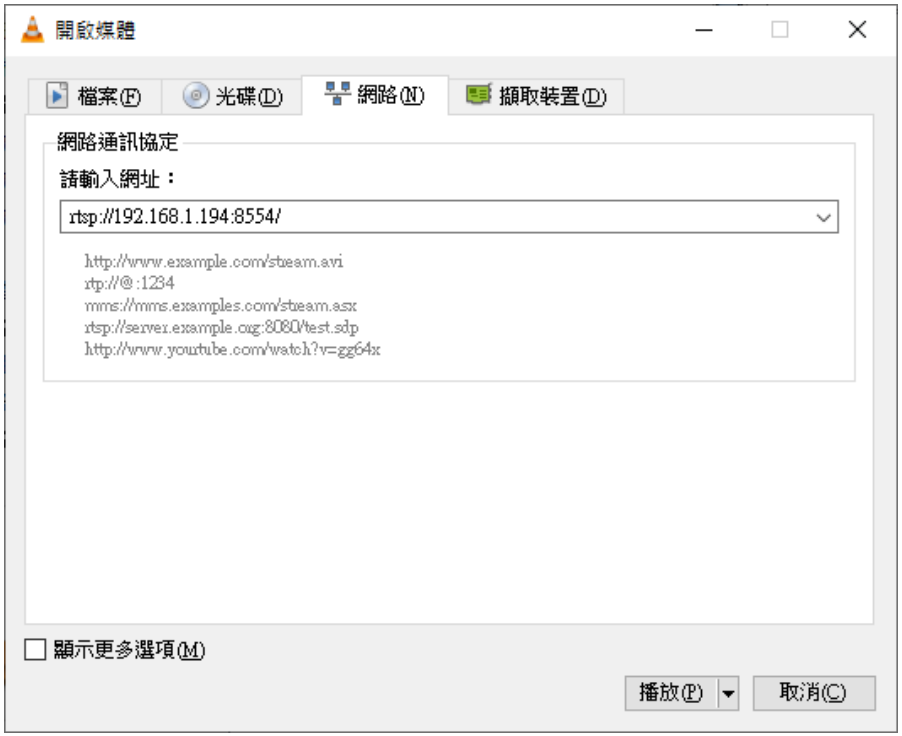
https://wiki.videolan.org/Documentation:Streaming_HowTo/Command_Line_Examples/
https://wiki.videolan.org/VLC_command-line_help/
https://helpmanual.io/help/cvlc/

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

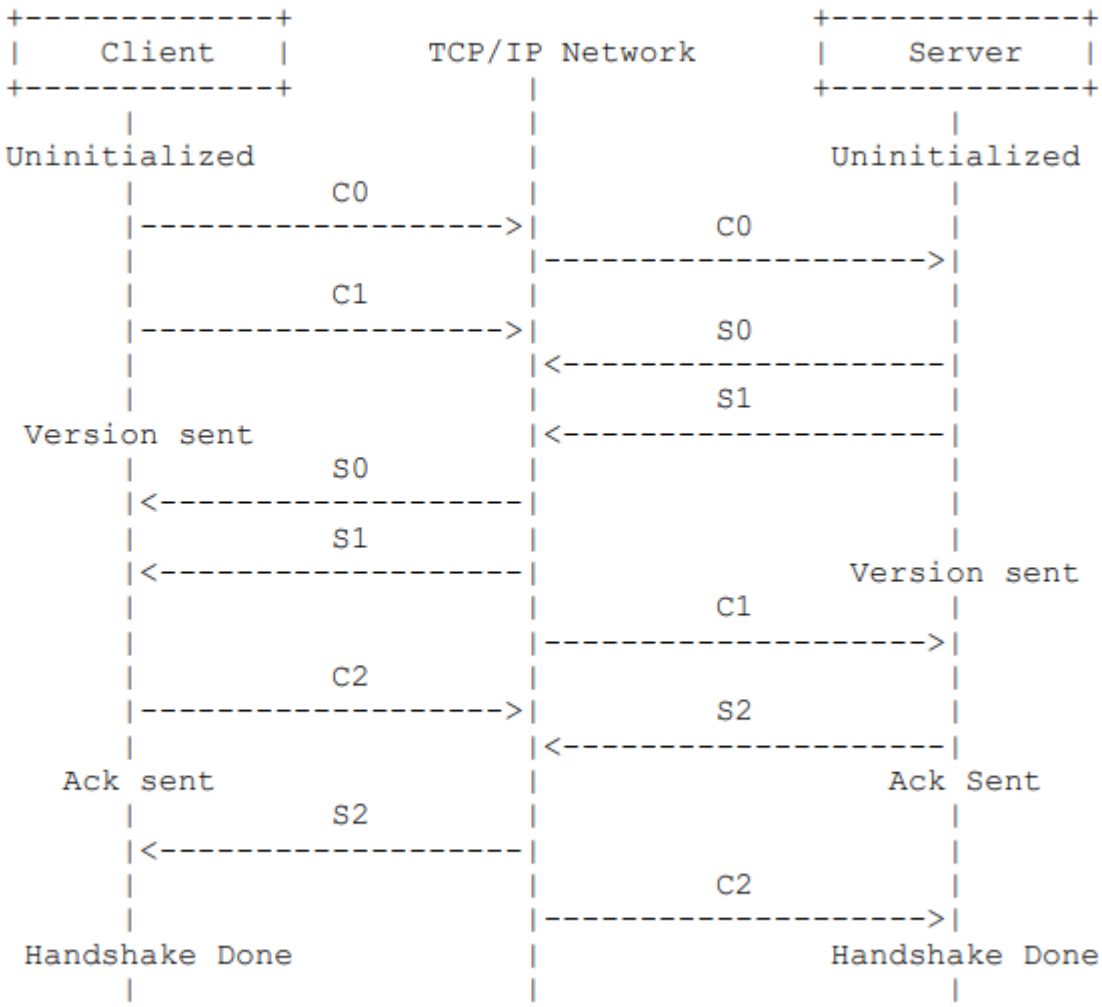
- Use VLC to watch video



https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol
https://www.adobe.com/content/dam/acom/en/devnet/rtmp/pdf/rtmp_specification_1.0.pdf

2. RTMP

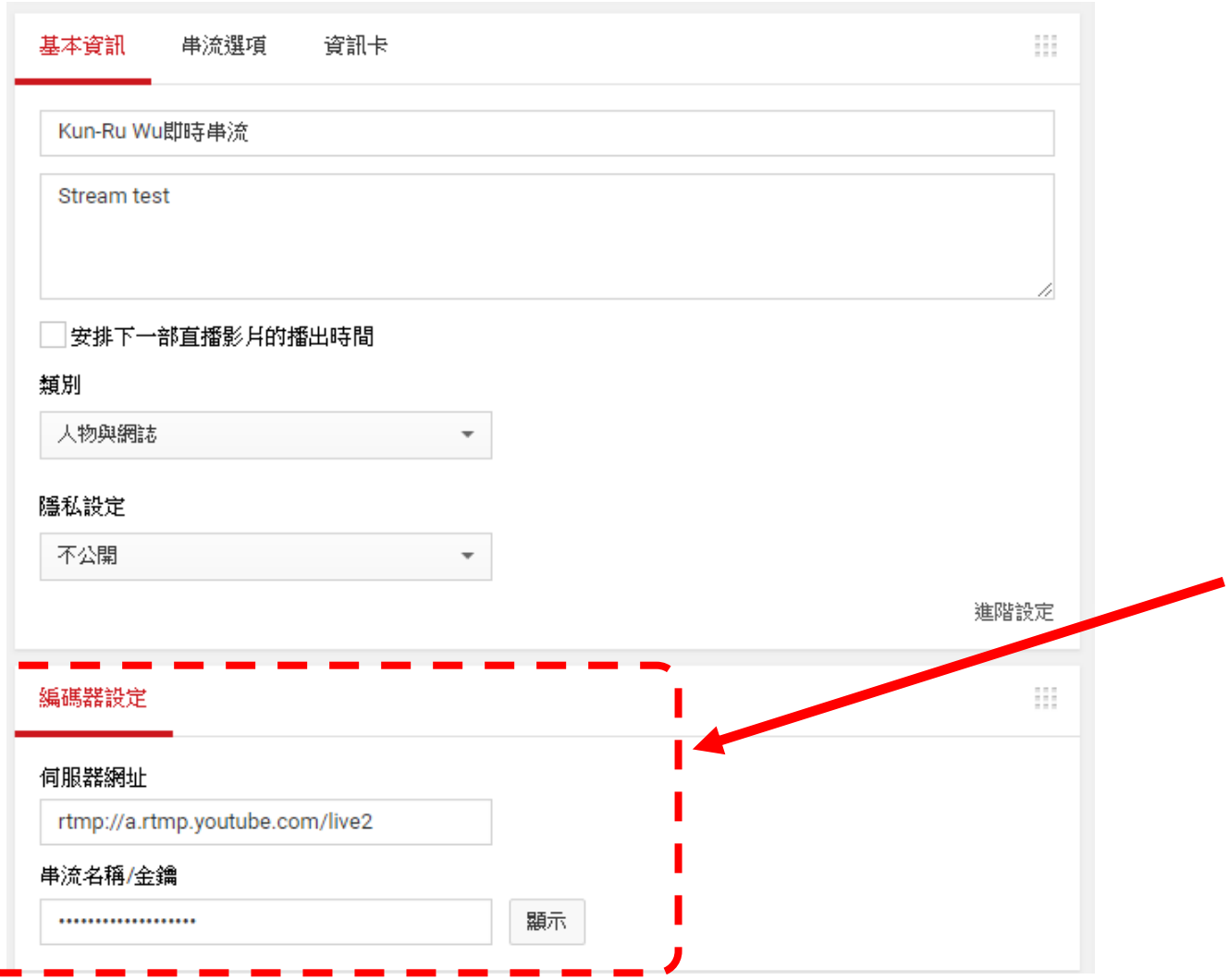
5.2.5. Handshake Diagram



Pictorial Representation of Handshake

2. RTMP to Youtube

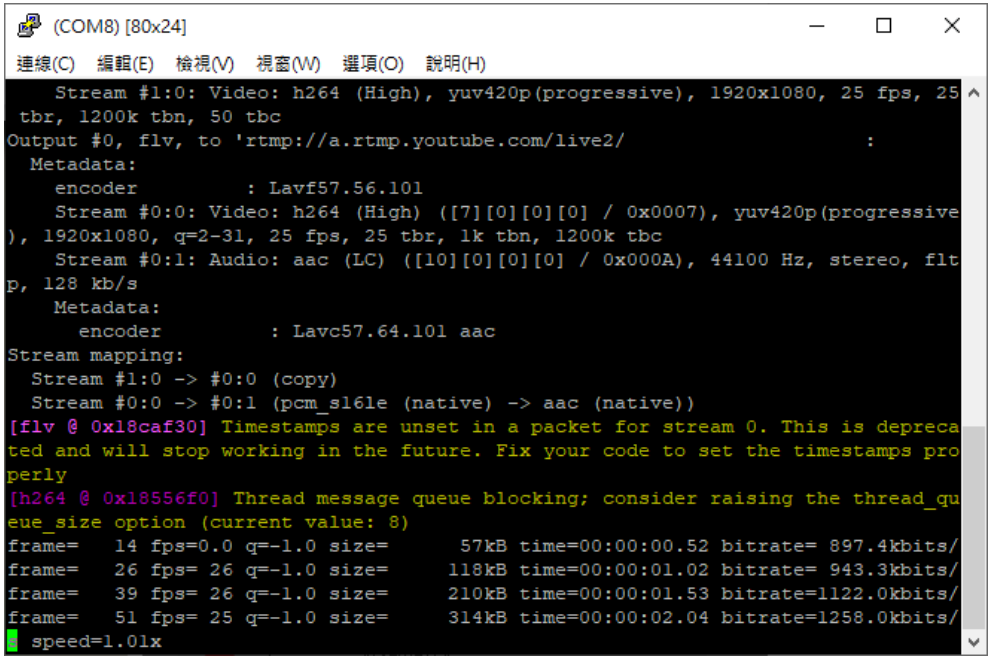
https://www.youtube.com/live_dashboard

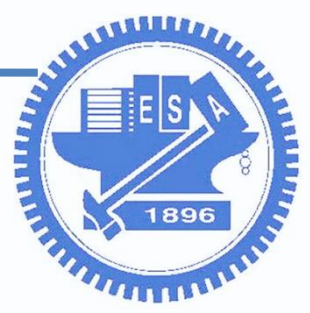


2. RTMP on PI

- 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**



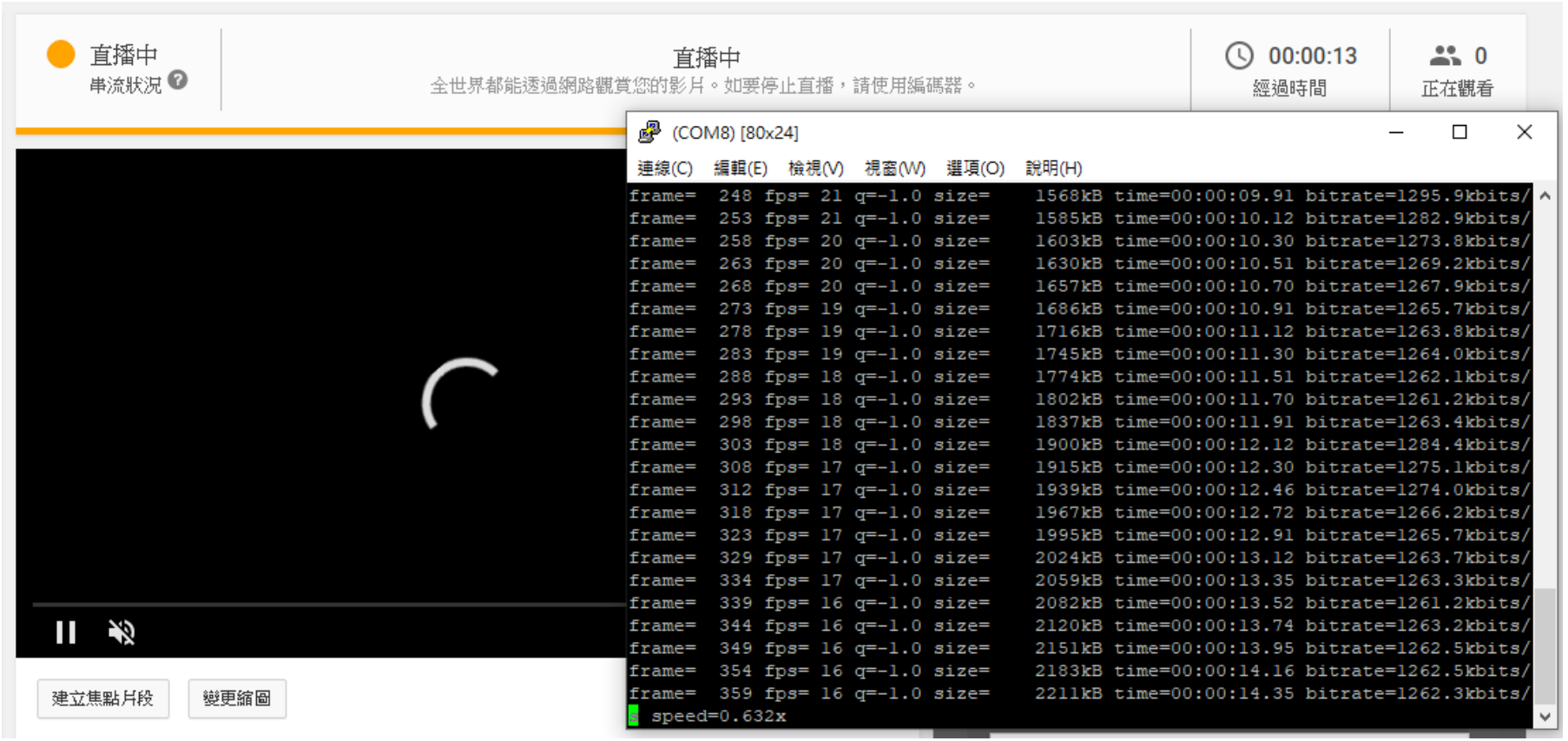


2. RTMP on PI

- 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…



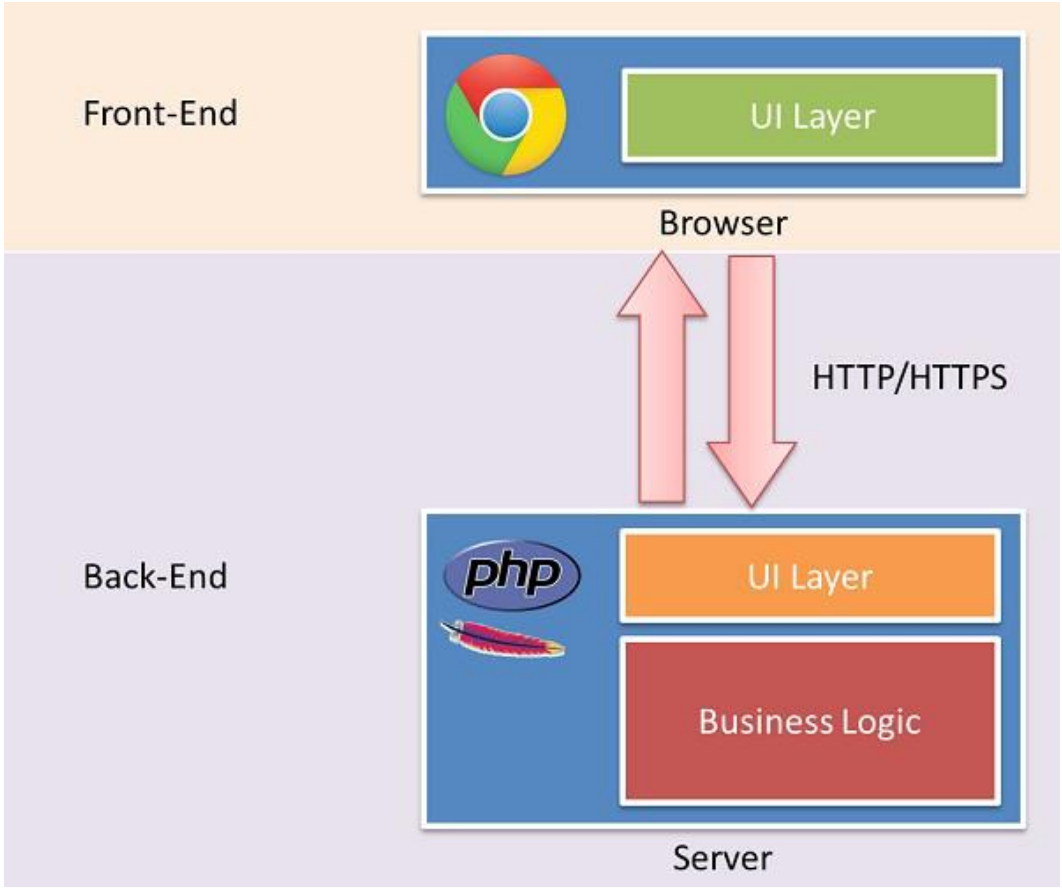
2. RTMP on PI

- Watch video

https://www.nczonline.net/blog/2013/10/07/node-js-and-the-new-web-front-end/
https://docs.microsoft.com/zh-tw/visualstudio/python/learn-flask-visual-studio-step-01-project-solution

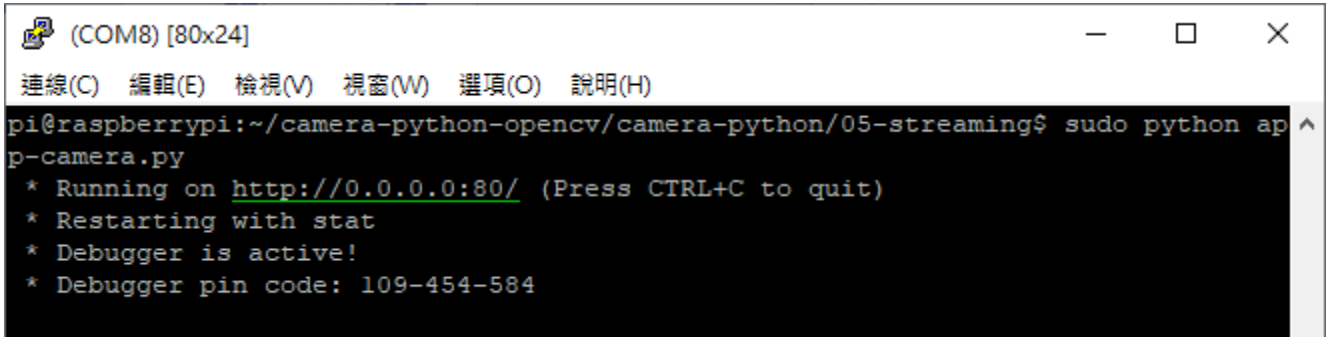
3. HTTP + MJPG

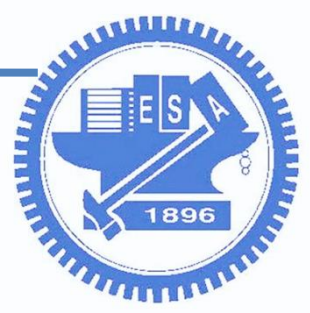
- MJPEG = Motion JPEG
  - 一種視訊壓縮格式
  - 每一個frame都使用JPEG編碼
  - 對運算能力與記憶體的需求較低
  - 許多網頁瀏覽器原生支援M-JPEG
- Flask 是一個輕量型的 Python Web 應用程式架構，可提供 URL 路由和頁面轉譯的基本要素。



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



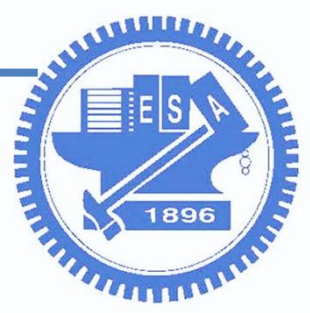


3. MJPG on PI

- 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')

@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>
<img id="bg" src="{{ url_for('video_feed') }}">
```



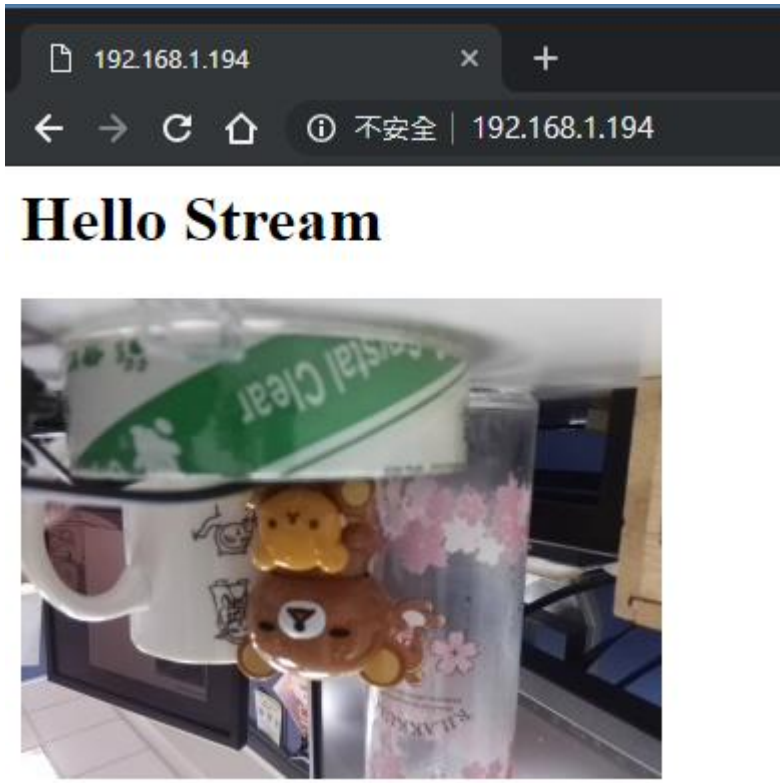
3. MJPG on PI

- camera_pi.py

```python
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



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