嵌入式系統設計概論與實作

曾煜棋、吳昆儒

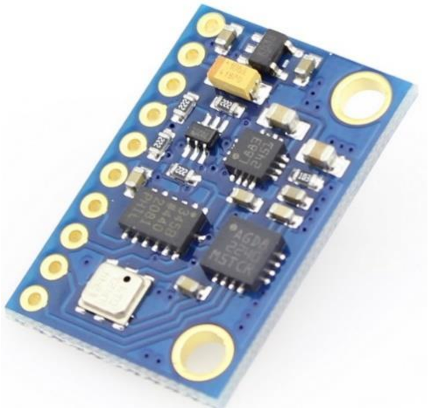
National Yang Ming Chiao Tung University

補充: 嵌入式系統設計概論與實作.0312update

周次	日期	主題
1	2/26	0.嵌入式課程介紹
2	3/5	1.嵌入式開發板 - 樹莓派介紹與設定
3	3/12	2.感測器應用(溫溼度、超音波)
4	3/19	3.人體活動偵測
5	3/26	4.人體活動偵測
~~6~~	~~4/2~~	~~兒童節及民族掃墓節調整放假(2日-5日)~~
7	4/9	5.網路攝影機 IP cam
8	4/16	6. 網路攝影機 + 機器學習影像辨識 (NYCU:期中考試(12日-16日)) By 台灣樹莓派的講師!!
9	4/23	7. 網路攝影機 + 影像辨識
10	4/30	Midterm, Project分組
11	5/7	8.推播廣告(beacon)應用
12	5/14	9.語音助理
13	5/21	Final Project – Proposal
14	5/28	10.樹莓派核心編譯 (Cross compile, Kernel)
15	6/4	Final Project prepare, Q&A, 補demo
16	6/11	Final Project demonstration (NYCU:學期考試(7日-11日))
17	6/18	(暫定)Final Project demonstration part 2

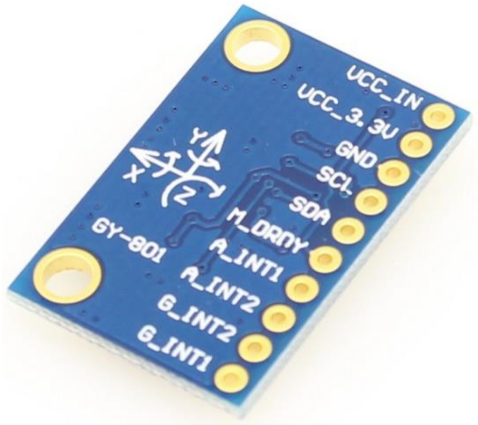
Last week

- 嵌入式應用: 人體活動偵測
  - 加速度、陀螺儀...等



- GY801 (I2C sensor)
  - 3-axis Accelerometer, Gyroscope, magnetometer and pressure

1. ADXL345：Accelerometer
2. L3G4200：Gyroscope
3. HMC5883：Magnetometer
4. BMP085：Pressure



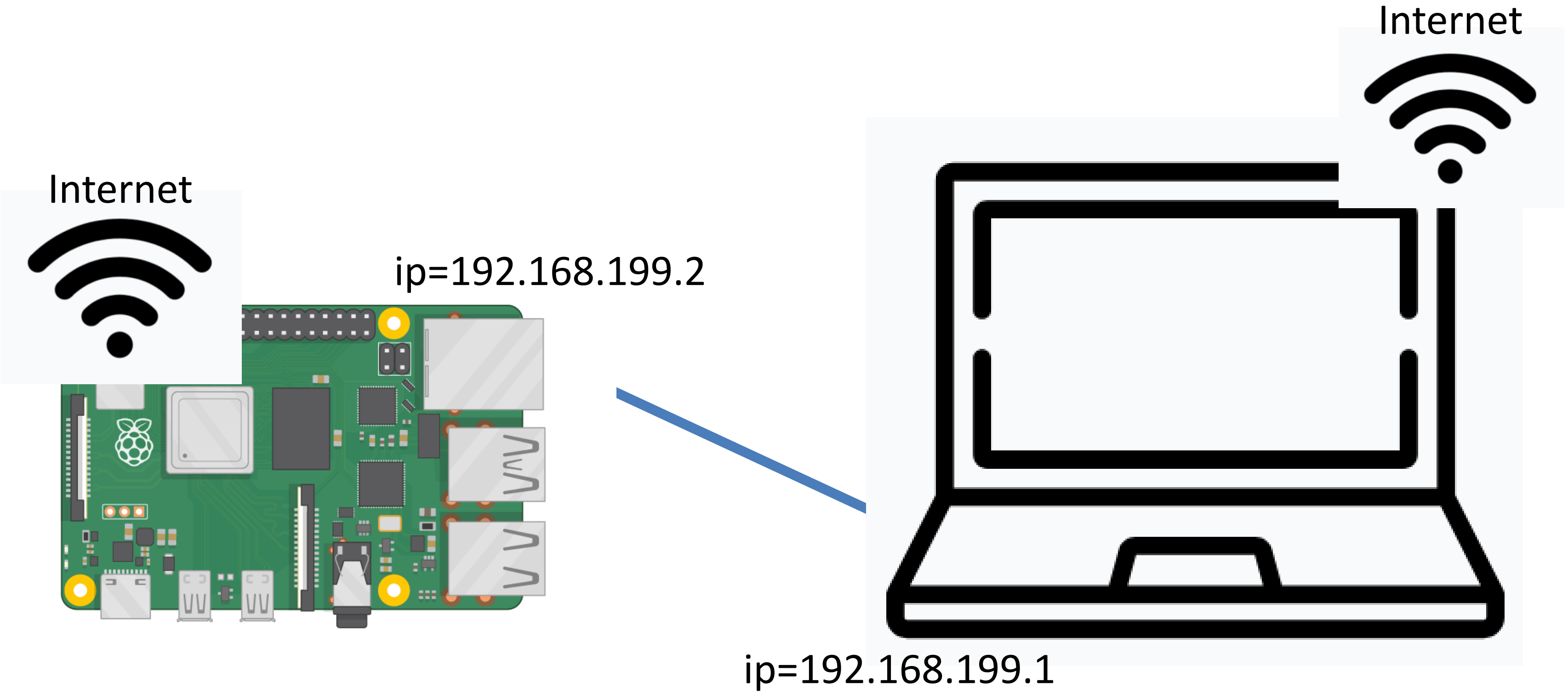
This week

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS

  - 建立網路串流
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264
    - 使用 RTMP

強烈建議: 使用有線網路遠端桌面

- 設定static IP address (PI與電腦互聯)



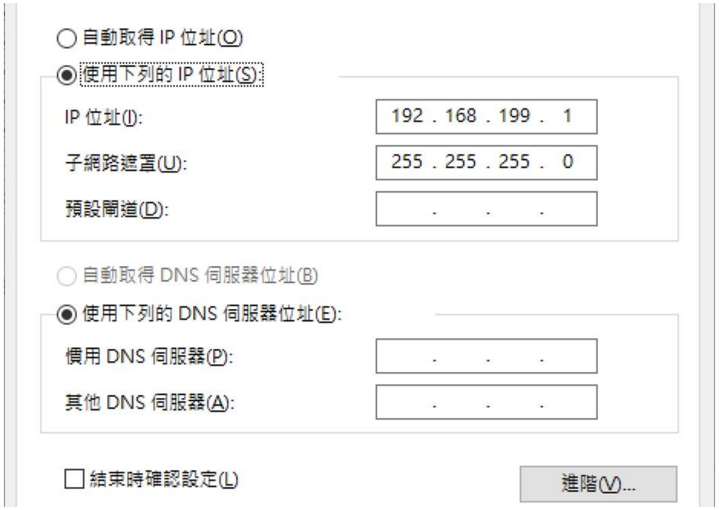
設定有線網路對接

- PI端:
  - sudo nano /boot/cmdline.txt
  - 最後加上 ip=192.168.199.2

設定這個後, 在開機時需要連接網路線

```
console=serial0,115200 console=tty1 root=PARTUUID=fba96bfa-02
rootfstype=ext4 elevator=deadline fsck.repair=yes rootwait
plymouth.ignore-serial-consoles ip=192.168.199.2
```

- 電腦端:
  - 有線網路設定: 192.168.199.1/255.255.255.0



○ 自動取得 IP 位址(O)

● 使用下列的 IP 位址(S):

IP 位址(I): 192 . 168 . 199 . 1

子網路遮罩(U): 255 . 255 . 255 . 0

預設閘道(D): . . .

○ 自動取得 DNS 伺服器位址(B)

● 使用下列的 DNS 伺服器位址(E):

慣用 DNS 伺服器(P): . . .

其他 DNS 伺服器(A): . . .

☐ 結束時確認設定(L)

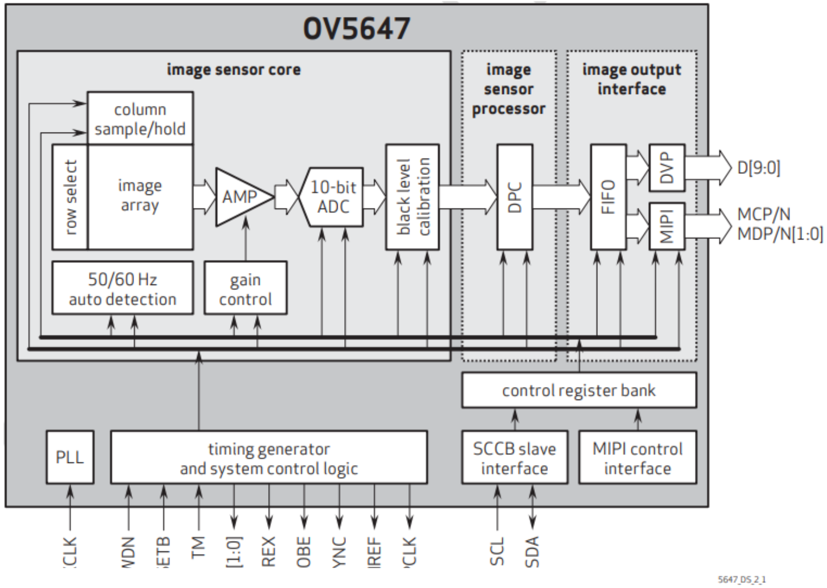
Outline

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS

  - 建立網路串流 (IP cam, Video streaming)
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264
    - 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



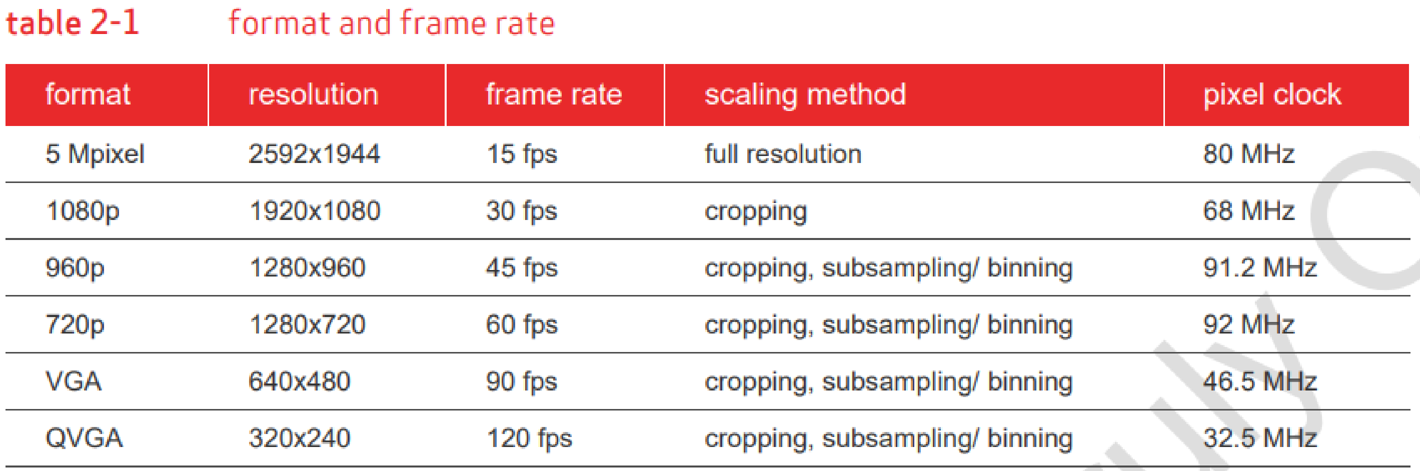
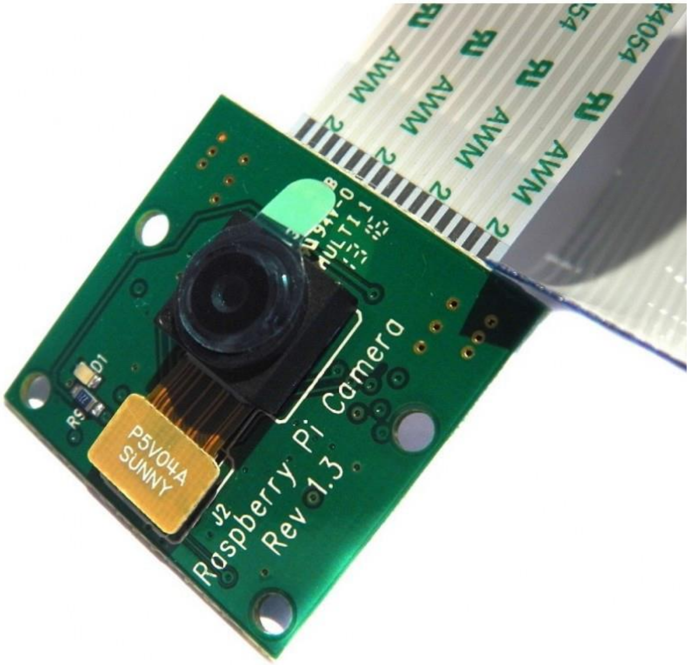


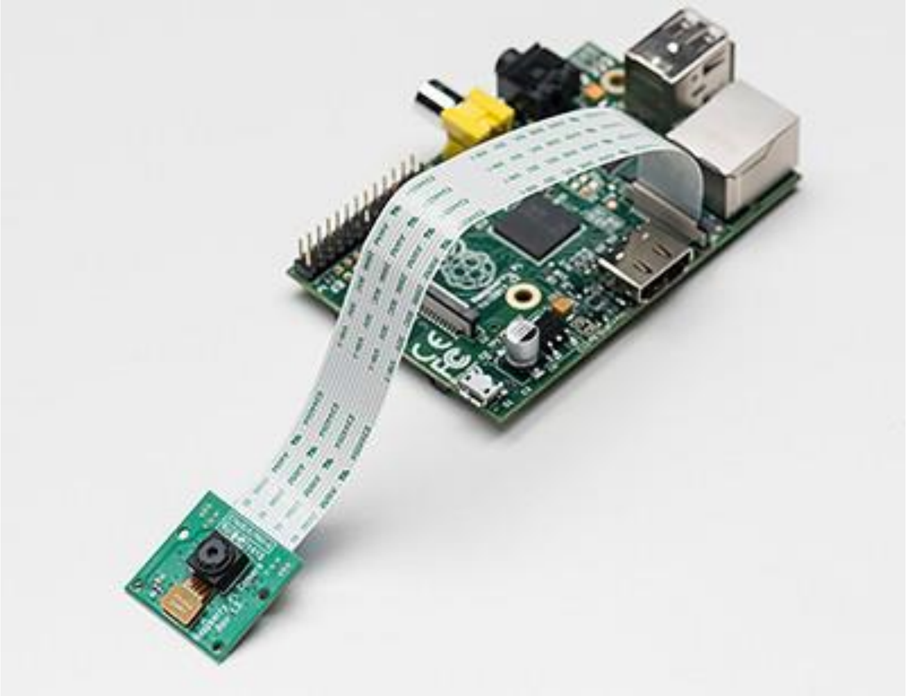
table 2-1 format and frame rate

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

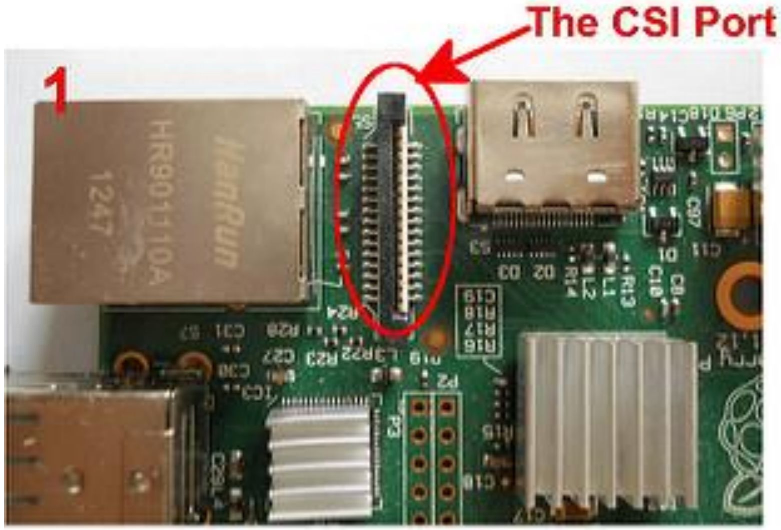
15-Pins, CSI interface

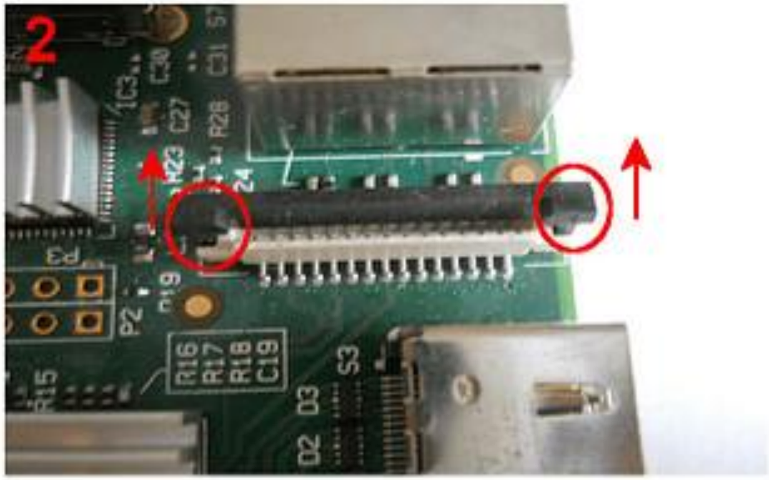


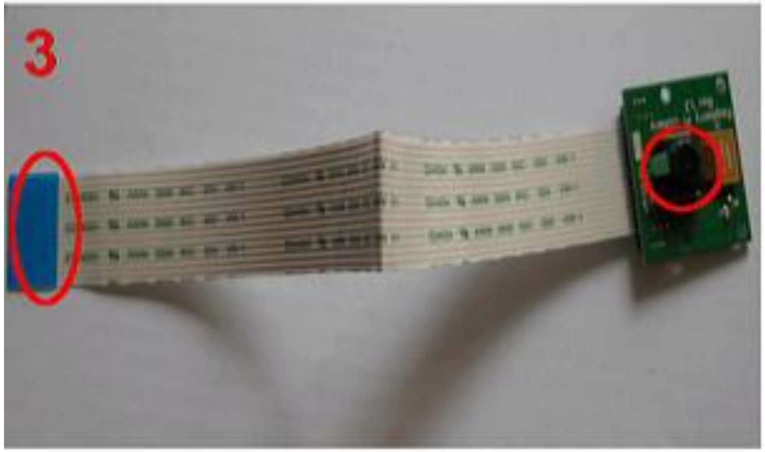


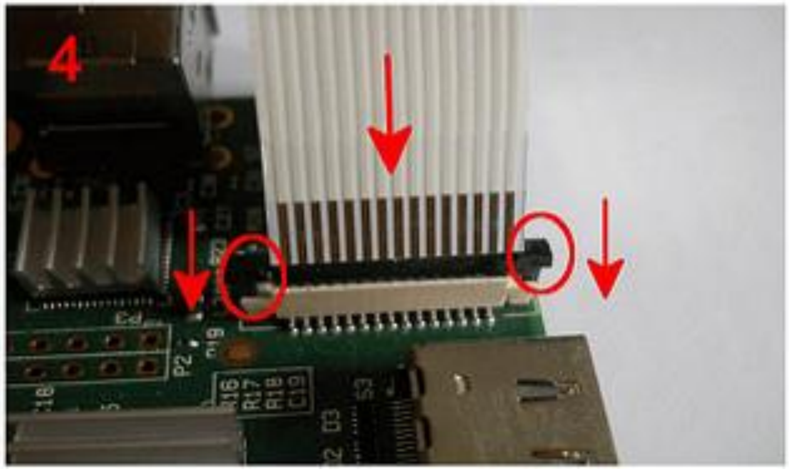
http://xmodulo.com/install-raspberry-pi-camera-board.html

Install PI camera









Camera commands

- Take a picture: raspistill
  - 3秒後拍照, 並編碼成png格式,長640x寬480, 無預覽
  - raspistill -n -t 3000 -o test.png -e png -w 640 -h 480
    - 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>

https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspivid.md

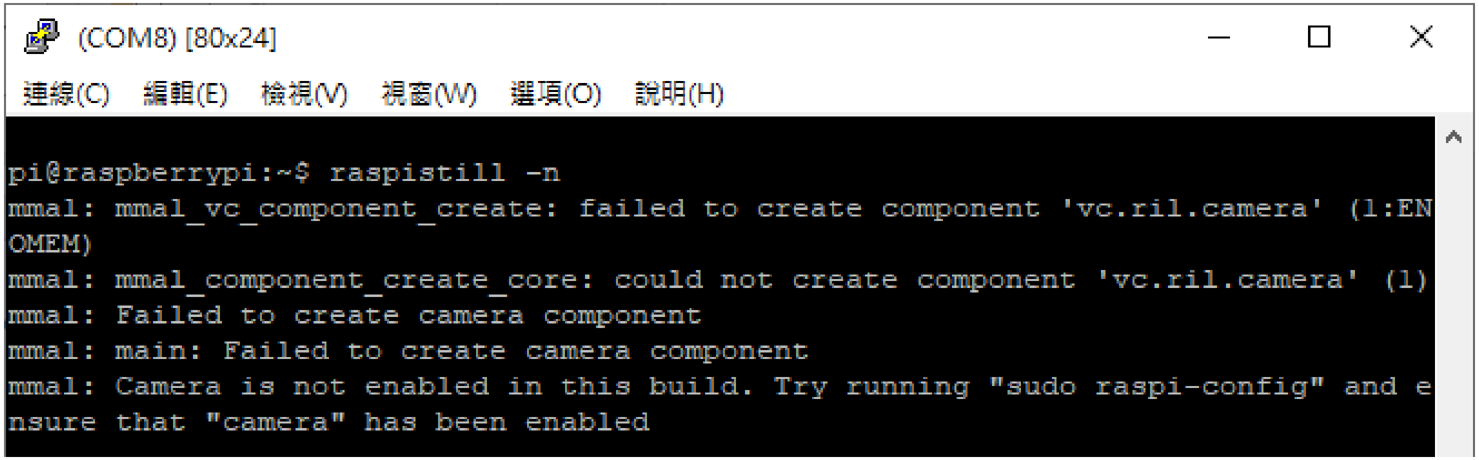
Camera commands

- Record a video: raspivid
  - 錄5秒的1080p30影片, 長640x寬480, 無預覽
  - Raspivid -n -t 5000 -w 640 -h 480 -o video.h264
    - 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

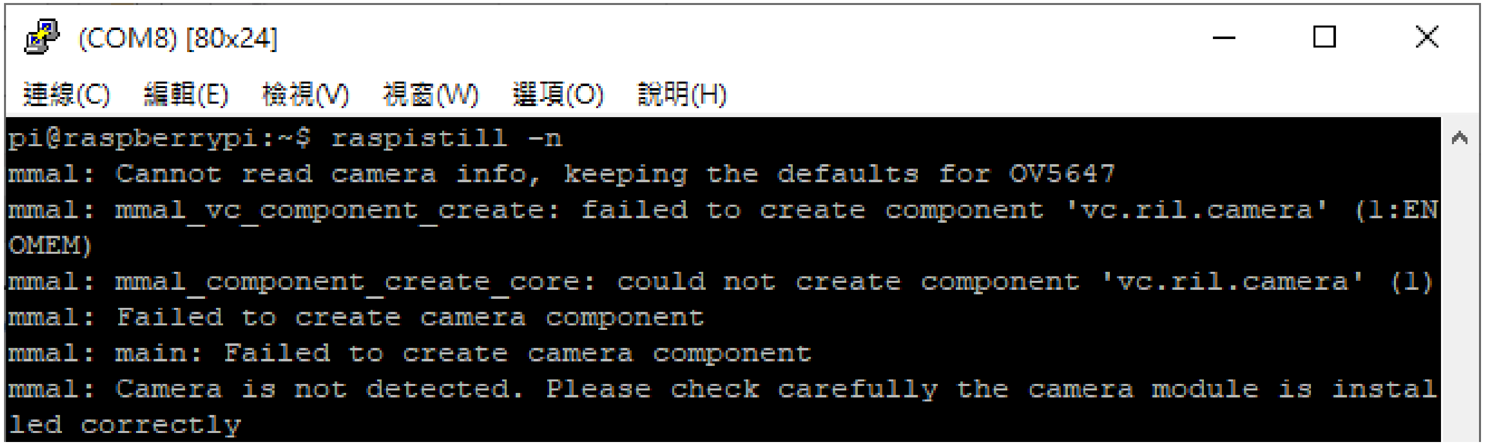


```
pi@raspberrypi:~$ raspistill -n
mmal: mmal_vc_component_create: failed to create component 'vc.ril.camera' (1:EN
OMEM)
mmal: mmal_component_create_core: could not create component 'vc.ril.camera' (1)
mmal: Failed to create camera component
mmal: main: Failed to create camera component
mmal: Camera is not 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 https://www.modmypi.com/blog/how-to-replace-the-raspberry-pi-camera-cable

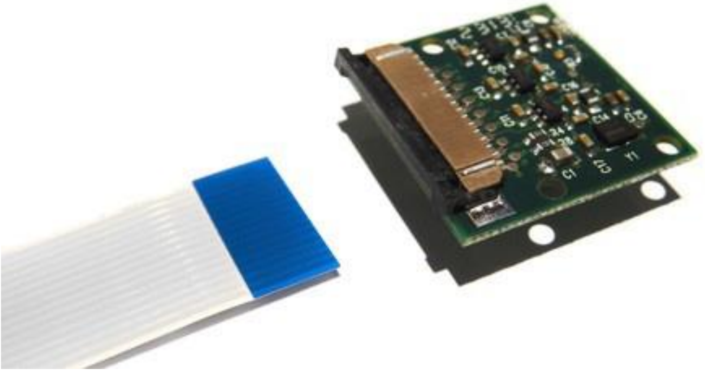
Error message?

- Msg: Camera is not detected



```
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
```

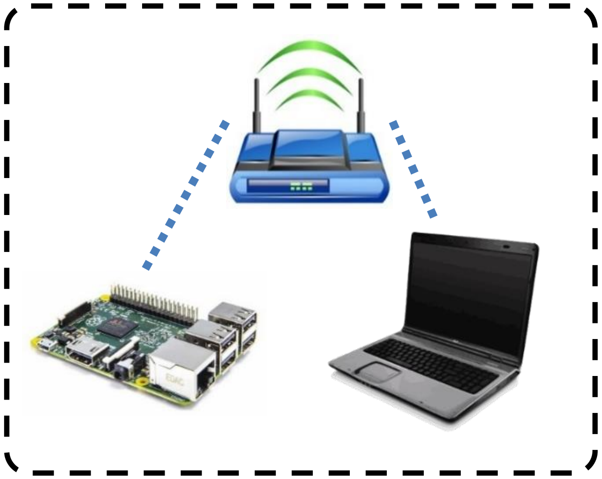
- Sol:
  - 重新安裝camera,或是更換排線
  - 或是檢查camera module是否鬆脫



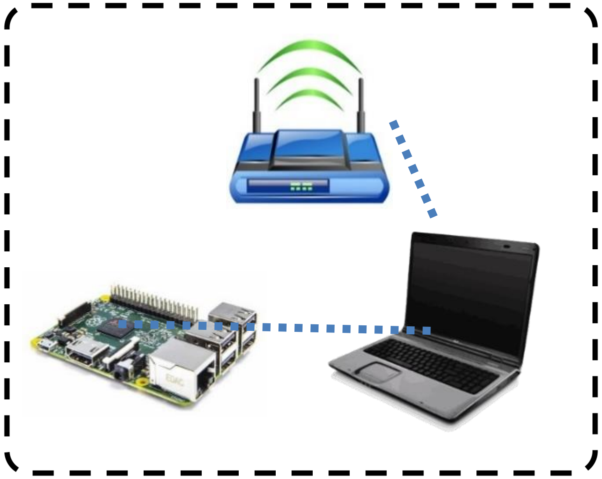
How to view image/video?

□ Methods:

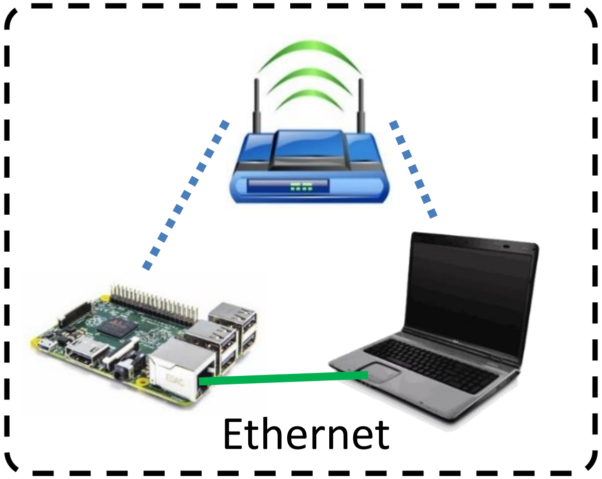
1. VNC
2. HDMI
3. winscp
4. (… etc)



Wi-Fi AP



Wi-Fi hotspot



Wired + Wireless

Outline

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS

  - 建立網路串流 (IP cam, Video streaming)
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264
    - 使用 RTMP

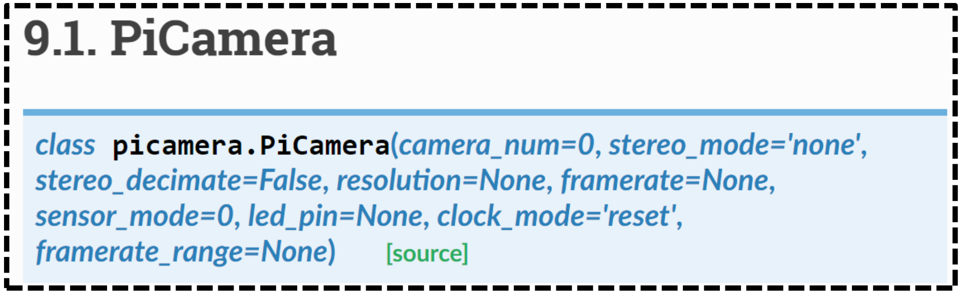
https://picamera.readthedocs.io/en/release-1.13/

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

Python code

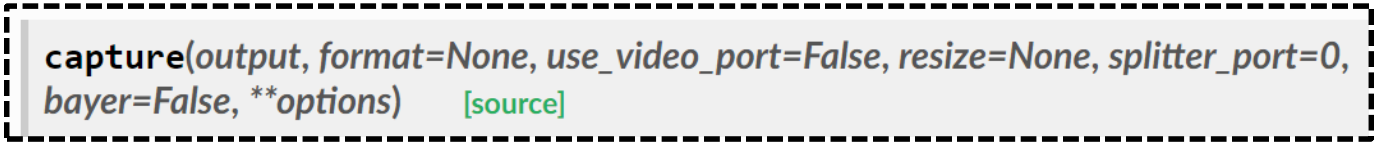
- Sample code for taking a picture

```
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]



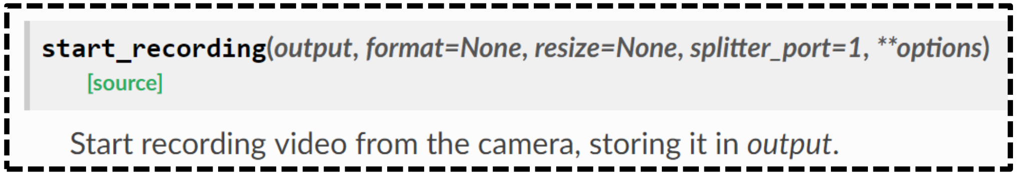
capture(*output, format=None, use_video_port=False, resize=None, splitter_port=0, bayer=False, **options*) [source] https://picamera.readthedocs.io/en/release-1.13/

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

Python code

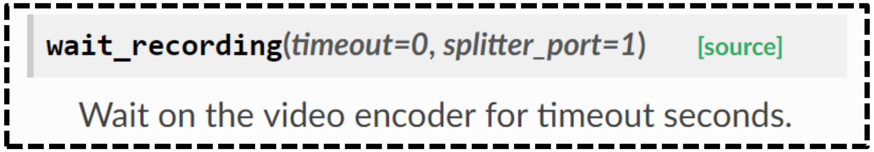
□ Sample code for record a video

```
import picamera

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



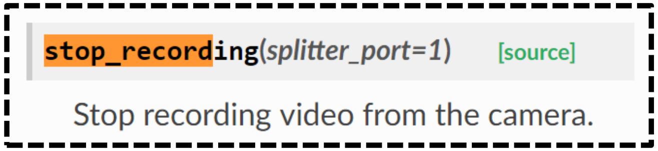
start_recording(*output, format=None, resize=None, splitter_port=1, **options*)
[source]

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

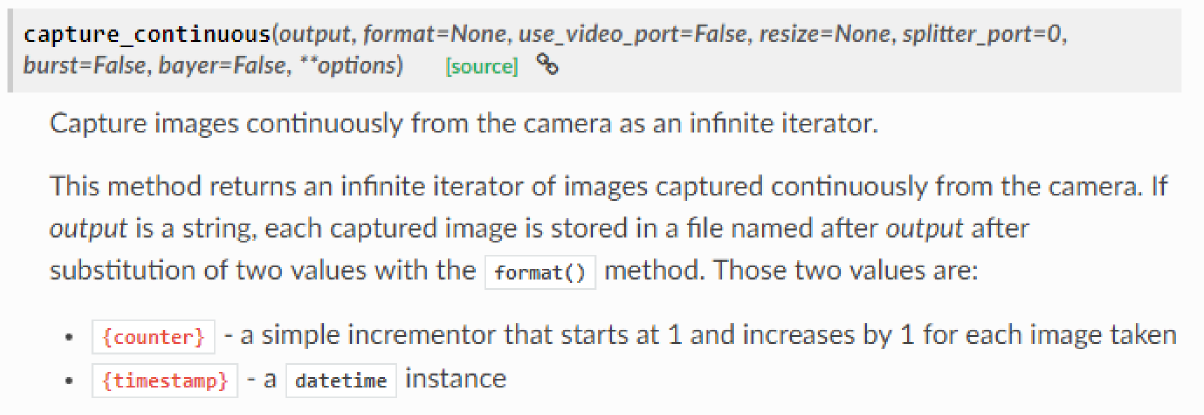
- Sample code for taking many pictures

```
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

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: image20200403_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:

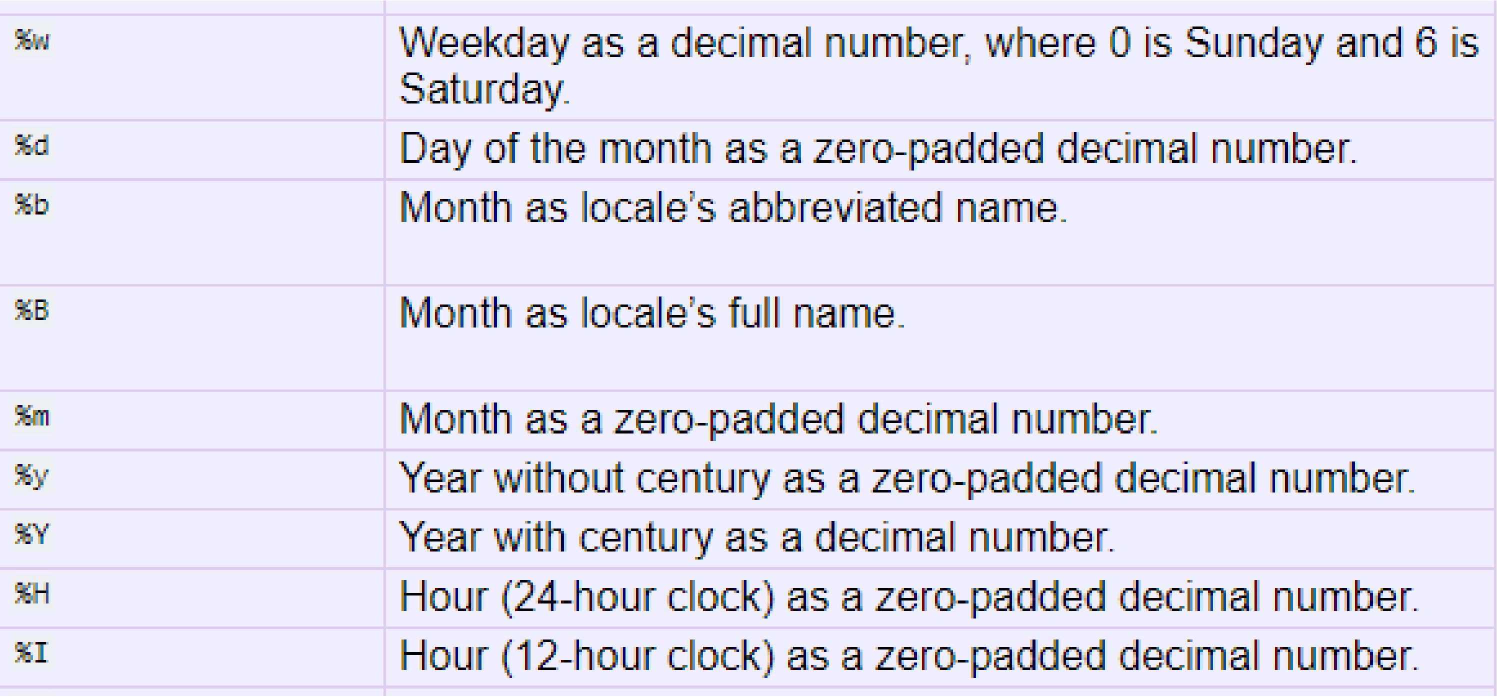
- `{counter}` - a simple incrementor that starts at 1 and increases by 1 for each image taken
- `{timestamp}` - a `datetime` instance

  - 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?



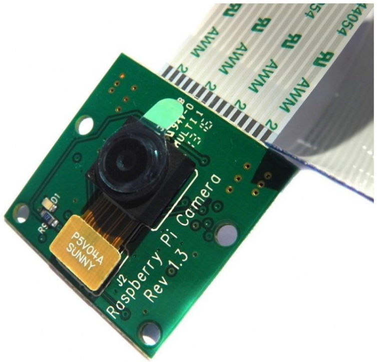
%w	Weekday as a decimal number, where 0 is Sunday and 6 is Saturday.
%d	Day of the month as a zero-padded decimal number.
%b	Month as locale's abbreviated name.
%B	Month as locale's full name.
%m	Month as a zero-padded decimal number.
%y	Year without century as a zero-padded decimal number.
%Y	Year with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

At HH:MM, start capturing…



001.jpg

002.jpg

.
.
.
.

010.jpg

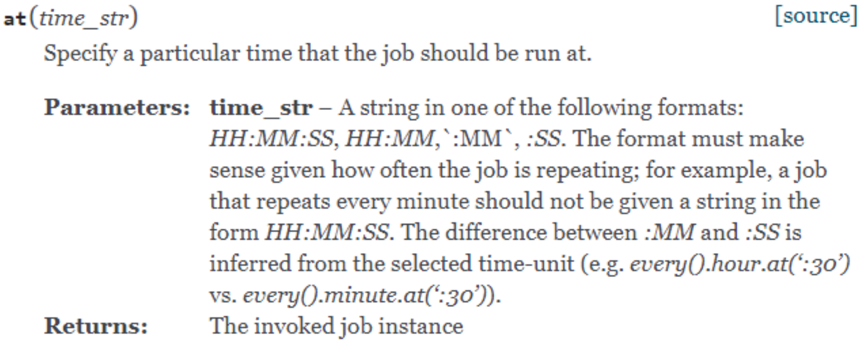
Check your current time first!
You might need to change time.

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.

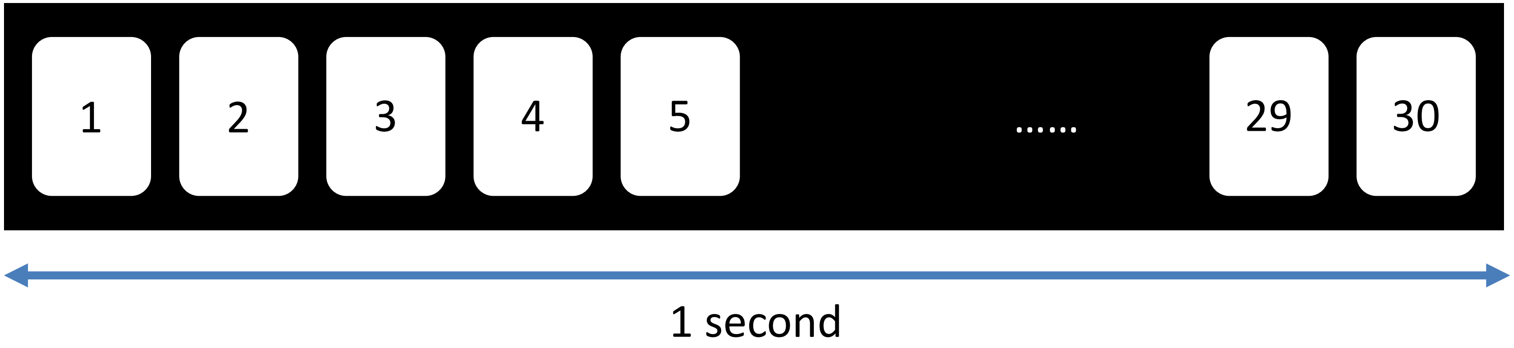
Parameters: **time_str** – A string in one of the following formats: *HH:MM:SS*, *HH:MM*,`:MM`, *:SS*. The format must make sense given how often the job is repeating; for example, a job that repeats every minute should not be given a string in the form *HH:MM:SS*. The difference between *:MM* and *:SS*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

Returns: The invoked job instance

Outline

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS
  - 建立網路串流 (IP cam, Video streaming)
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264
    - 使用 RTMP

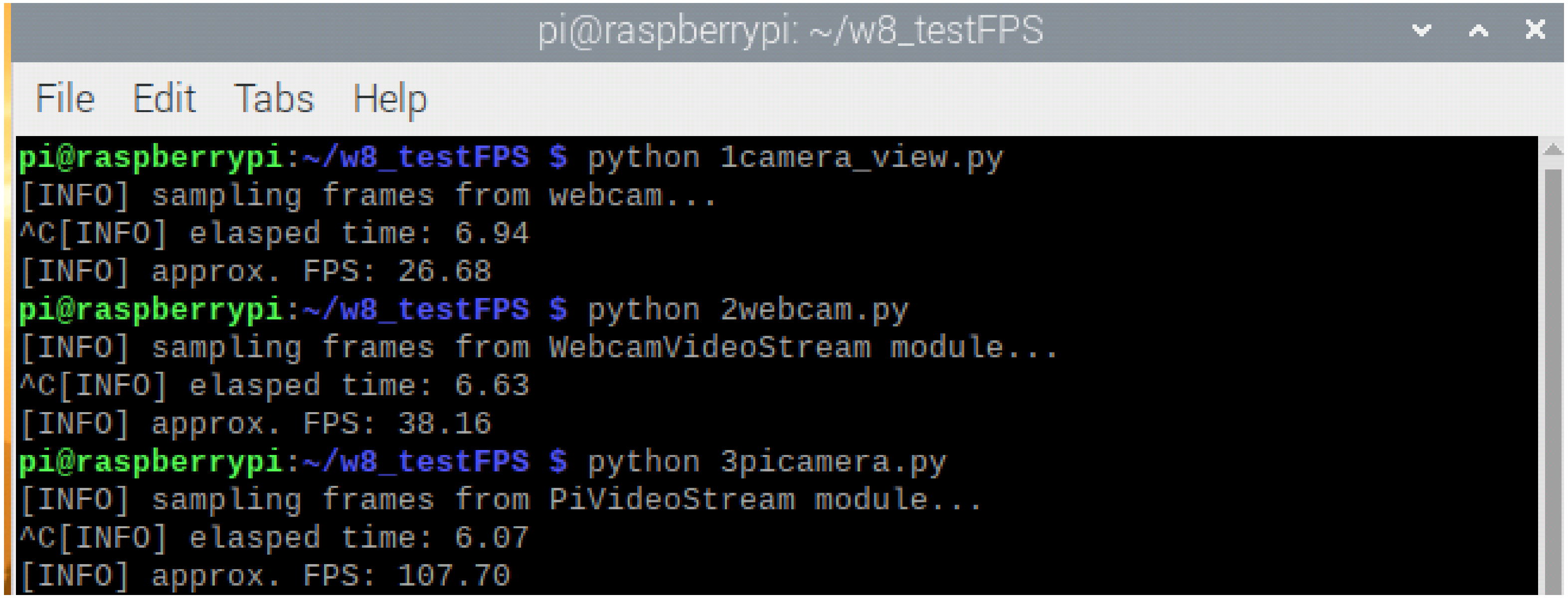
FPS (Frame per Second)



30 FPS = 30 frames in 1 second

Measure FPS

- Download and unzip “testFPS.zip” file
- Three example:
  - 1camera_view.py
  - 2webcam.py
  - 3picamera.py



```
pi@raspberrypi:~/w8_testFPS $ python 1camera_view.py
[INFO] sampling frames from webcam...
^C[INFO] elasped time: 6.94
[INFO] approx. FPS: 26.68
pi@raspberrypi:~/w8_testFPS $ python 2webcam.py
[INFO] sampling frames from WebcamVideoStream module...
^C[INFO] elasped time: 6.63
[INFO] approx. FPS: 38.16
pi@raspberrypi:~/w8_testFPS $ python 3picamera.py
[INFO] sampling frames from PiVideoStream module...
^C[INFO] elasped time: 6.07
[INFO] approx. FPS: 107.70
```

Test FPS

- □ Install opencv: **sudo apt-get install python3-opencv**

- □ [1camera_view]

  import cv2
  vs = **cv2.VideoCapture(0)**

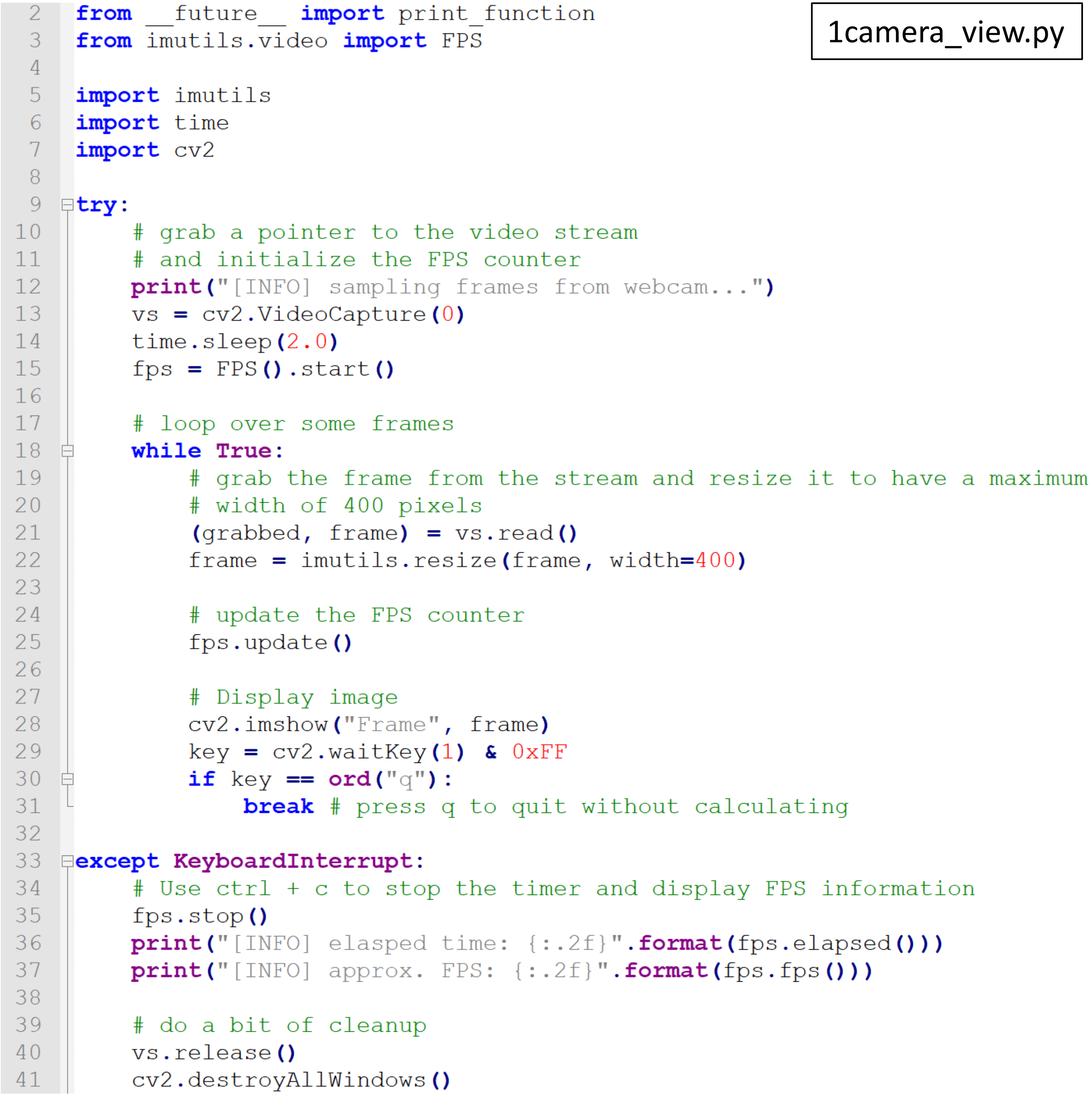
- □ [2webcam]

  from imutils.video import WebcamVideoStream
  vs = **WebcamVideoStream(src=0).start()**

- □ [3picamera]

  from imutils.video.pivideostream import PiVideoStream
  vs = **PiVideoStream().start()**

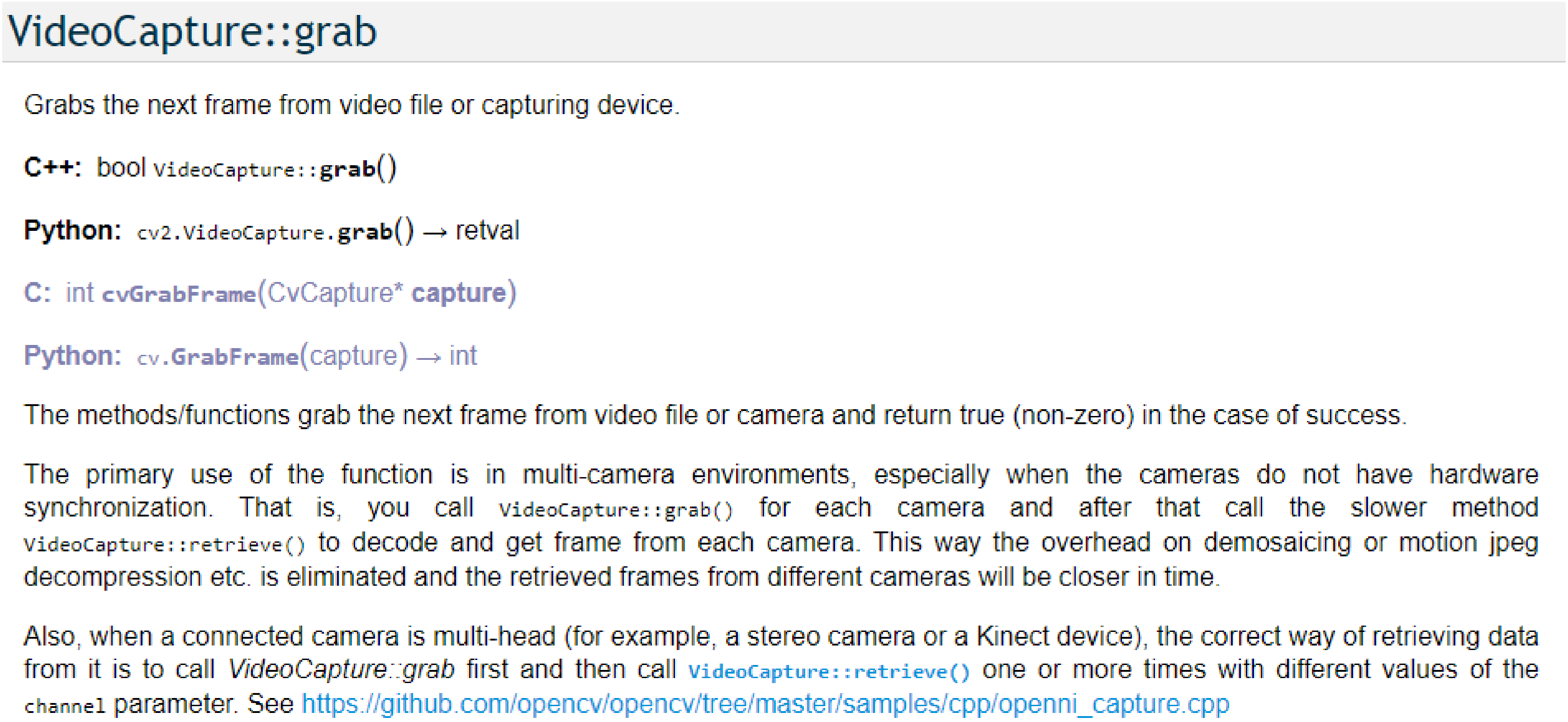
1camera_view.py



```python
from __future__ import print_function
from imutils.video import FPS

import imutils
import time
import cv2

try:
    # grab a pointer to the video stream
    # and initialize the FPS counter
    print("[INFO] sampling frames from webcam...")
    vs = cv2.VideoCapture(0)
    time.sleep(2.0)
    fps = FPS().start()

    # loop over some frames
    while True:
        # grab the frame from the stream and resize it to have a maximum
        # width of 400 pixels
        (grabbed, frame) = vs.read()
        frame = imutils.resize(frame, width=400)

        # update the FPS counter
        fps.update()

        # Display image
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break # press q to quit without calculating

except KeyboardInterrupt:
    # Use ctrl + c to stop the timer and display FPS information
    fps.stop()
    print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
    print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

    # do a bit of cleanup
    vs.release()
    cv2.destroyAllWindows()
```

https://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html#videocapture-grab

(1) VideoCapture::grab



VideoCapture::grab

Grabs the next frame from video file or capturing device.

C++: bool `VideoCapture::`**`grab`**()

Python: `cv2.VideoCapture.`**`grab`**() → retval

C: int **`cvGrabFrame`**(CvCapture* **capture**)

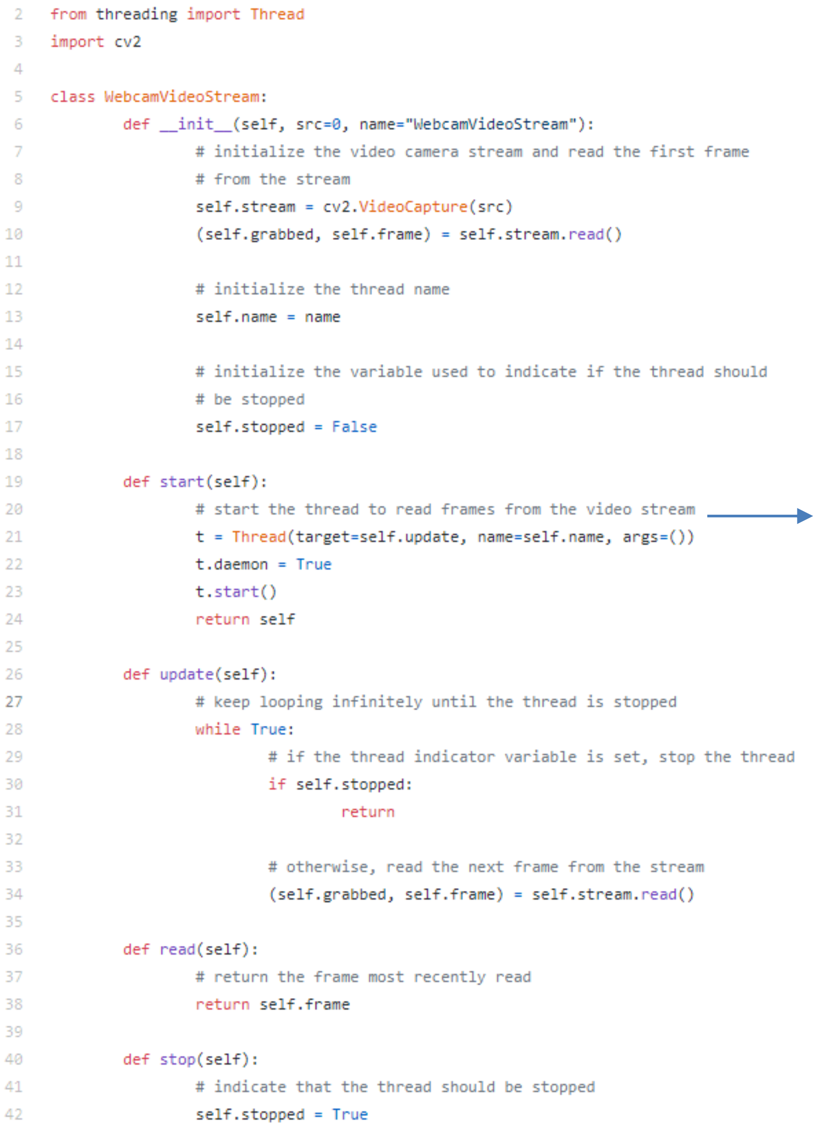
Python: `cv.`**`GrabFrame`**(capture) → int

The methods/functions grab the next frame from video file or camera and return true (non-zero) in the case of success.

The primary use of the function is in multi-camera environments, especially when the cameras do not have hardware synchronization. That is, you call `VideoCapture::grab()` for each camera and after that call the slower method `VideoCapture::retrieve()` to decode and get frame from each camera. This way the overhead on demosaicing or motion jpeg decompression etc. is eliminated and the retrieved frames from different cameras will be closer in time.

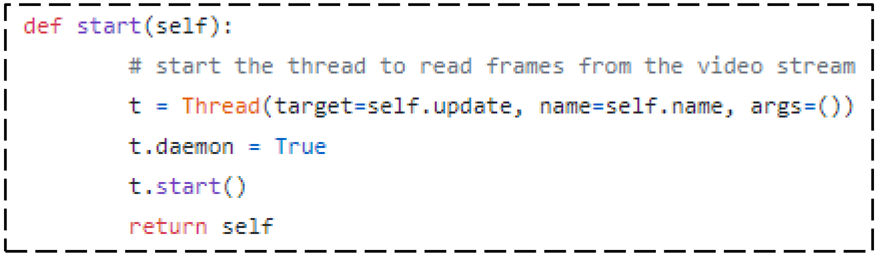
Also, when a connected camera is multi-head (for example, a stereo camera or a Kinect device), the correct way of retrieving data from it is to call *VideoCapture::grab* first and then call `VideoCapture::retrieve()` one or more times with different values of the `channel` parameter. See https://github.com/opencv/opencv/tree/master/samples/cpp/openni_capture.cpp https://github.com/jrosebr1/imutils/blob/224d591b4c3a9efc855e2e8eabc9c55199c696c3/imutils/video/webcamvideostream.py

(2) WebcamVideoStream



```python
from threading import Thread
import cv2

class WebcamVideoStream:
    def __init__(self, src=0, name="WebcamVideoStream"):
        # initialize the video camera stream and read the first frame
        # from the stream
        self.stream = cv2.VideoCapture(src)
        (self.grabbed, self.frame) = self.stream.read()

        # initialize the thread name
        self.name = name

        # initialize the variable used to indicate if the thread should
        # be stopped
        self.stopped = False

    def start(self):
        # start the thread to read frames from the video stream
        t = Thread(target=self.update, name=self.name, args=())
        t.daemon = True
        t.start()
        return self

    def update(self):
        # keep looping infinitely until the thread is stopped
        while True:
            # if the thread indicator variable is set, stop the thread
            if self.stopped:
                return

            # otherwise, read the next frame from the stream
            (self.grabbed, self.frame) = self.stream.read()

    def read(self):
        # return the frame most recently read
        return self.frame

    def stop(self):
        # indicate that the thread should be stopped
        self.stopped = True
```

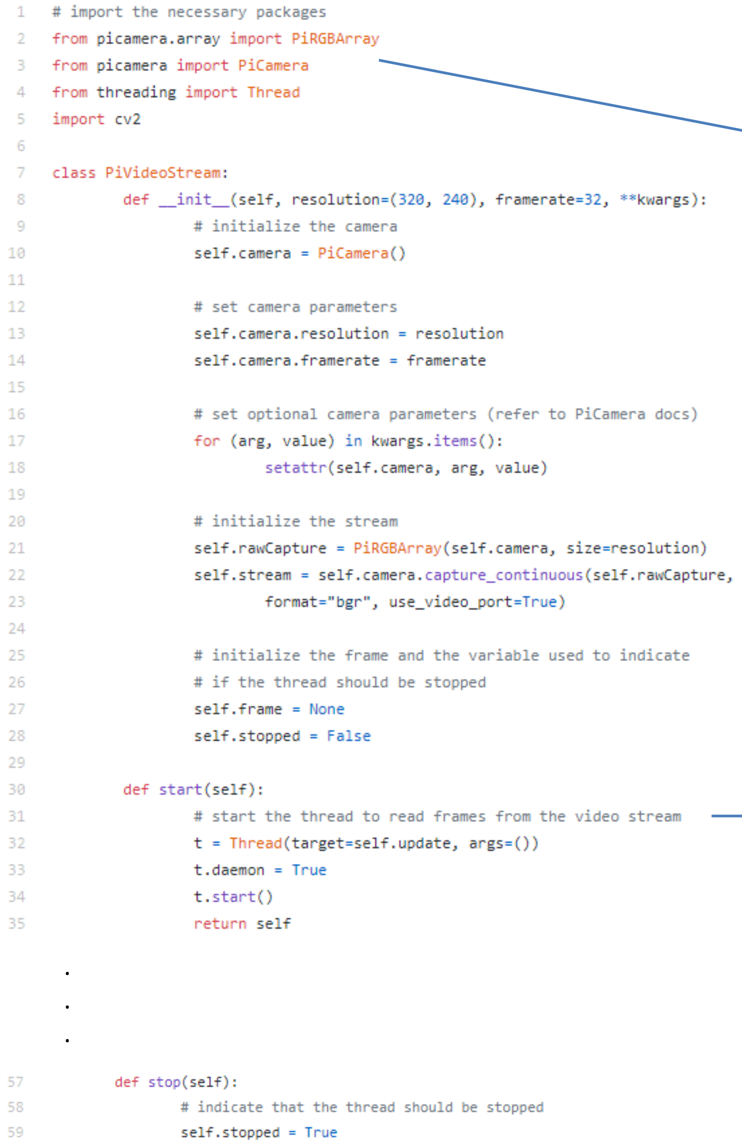
Use thread to read frames



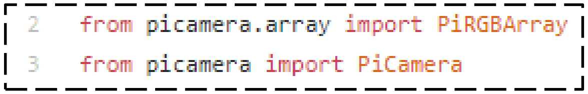
```python
def start(self):
    # start the thread to read frames from the video stream
    t = Thread(target=self.update, name=self.name, args=())
    t.daemon = True
    t.start()
    return self
```

https://github.com/jrosebr1/imutils/blob/65e7ea624757bc3f527e6316d1cdf6d0e86360cc/imutils/video/pivideostream.py

(3) PiVideoStream

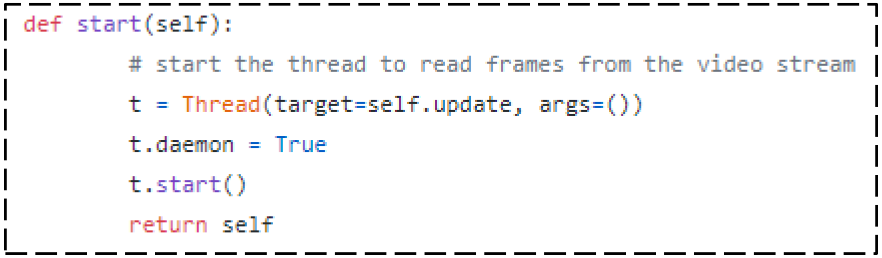


```python
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
from threading import Thread
import cv2

class PiVideoStream:
    def __init__(self, resolution=(320, 240), framerate=32, **kwargs):
        # initialize the camera
        self.camera = PiCamera()

        # set camera parameters
        self.camera.resolution = resolution
        self.camera.framerate = framerate

        # set optional camera parameters (refer to PiCamera docs)
        for (arg, value) in kwargs.items():
            setattr(self.camera, arg, value)

        # initialize the stream
        self.rawCapture = PiRGBArray(self.camera, size=resolution)
        self.stream = self.camera.capture_continuous(self.rawCapture,
            format="bgr", use_video_port=True)

        # initialize the frame and the variable used to indicate
        # if the thread should be stopped
        self.frame = None
        self.stopped = False

    def start(self):
        # start the thread to read frames from the video stream
        t = Thread(target=self.update, args=())
        t.daemon = True
        t.start()
        return self
.
.
.
    def stop(self):
        # indicate that the thread should be stopped
        self.stopped = True
```



```python
from picamera.array import PiRGBArray
from picamera import PiCamera
```

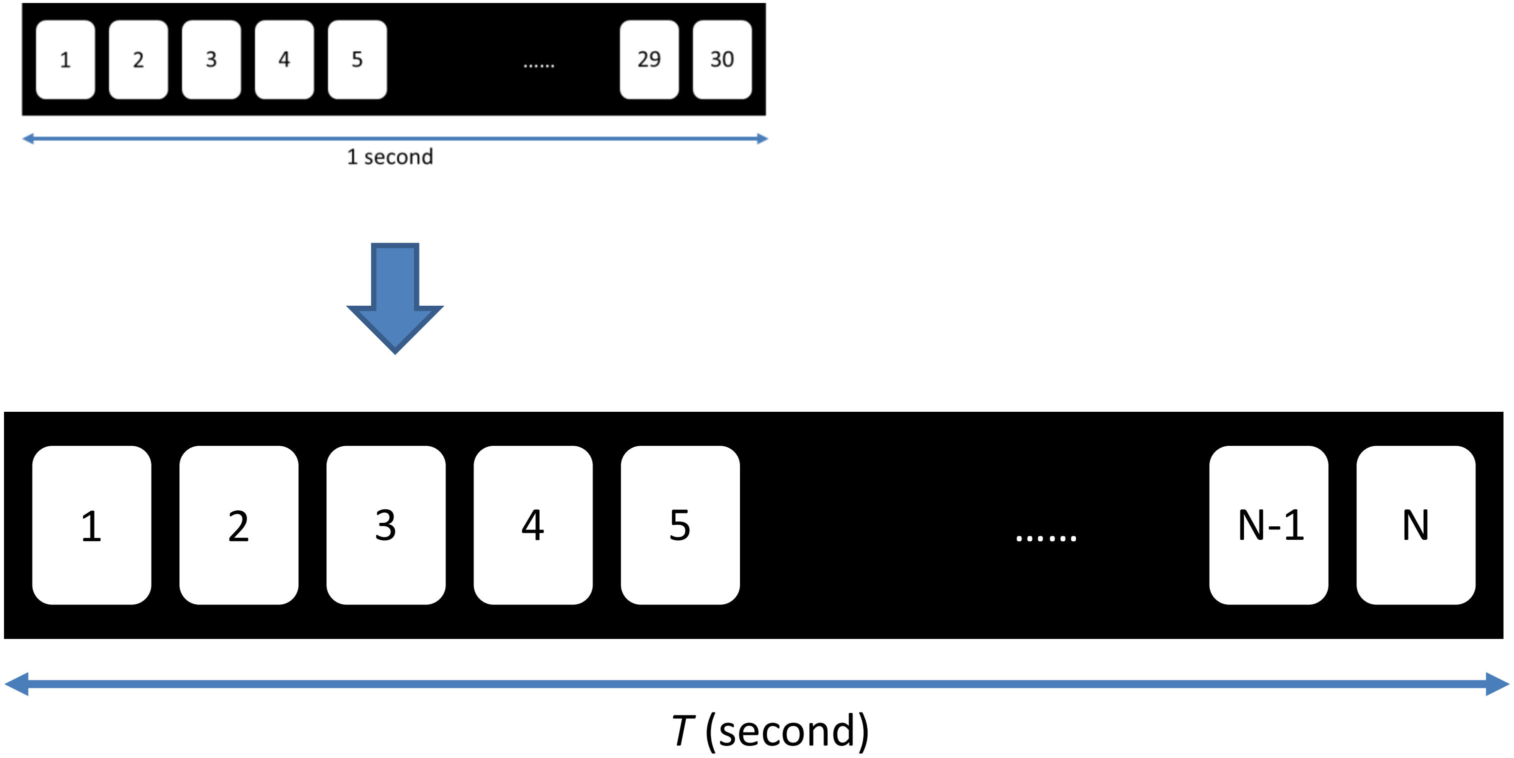
Use PiCamera module to read frames



```python
def start(self):
    # start the thread to read frames from the video stream
    t = Thread(target=self.update, args=())
    t.daemon = True
    t.start()
    return self
```

https://github.com/jrosebr1/imutils/blob/master/imutils/video/fps.py

How to calculate FPS?



https://github.com/jrosebr1/imutils/blob/master/imutils/video/fps.py

How to calculate FPS?

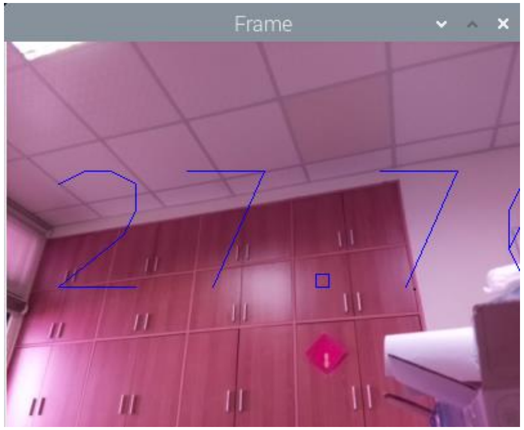
from imutils.video import FPS

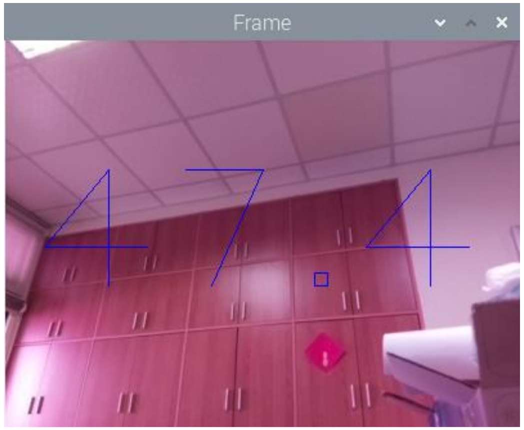
```
import datetime

class FPS:
def __init__(self):
        # store the start time, end time, and total number of frames
        # that were examined between the start and end intervals
        self._start = None
        self._end = None
        self._numFrames = 0

    def start(self):
        # start the timer
        self._start = datetime.datetime.now()
        return self

    def stop(self):
        # stop the timer
        self._end = datetime.datetime.now()

    def update(self):
        # increment the total number of frames examined during the
        # start and end intervals
        self._numFrames += 1

    def elapsed(self):
        # return the total number of seconds between the start and
        # end interval
        return (self._end - self._start).total_seconds()

    def fps(self):
        # compute the (approximate) frames per second
        return self._numFrames / self.elapsed()
```

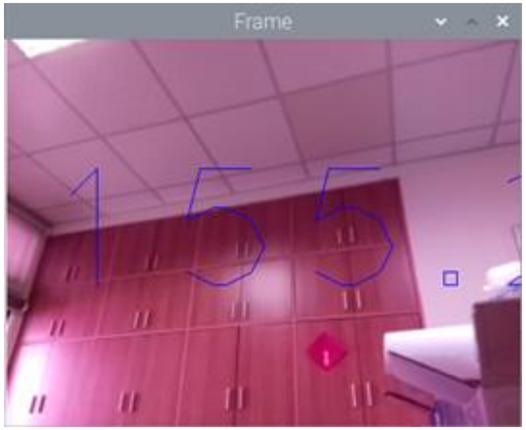
$$FPS = \frac{\text{the number of frames}}{\text{time duration between first and last frame}}$$

Quiz 2

- Modify the sample code of Test FPS, **show current FPS on the frame for each capture mode**.
  - cv2.VideoCapture(0); WebcamVideoStream(src=0); PiVideoStream()
  - Calculate the time duration between frame and frame.
  - FPS = 1 / (time duration)
- Hint: read the formula in imutils/video/fps.py
- Hint2: cv2.putText(img, text, org, fontScale, color)







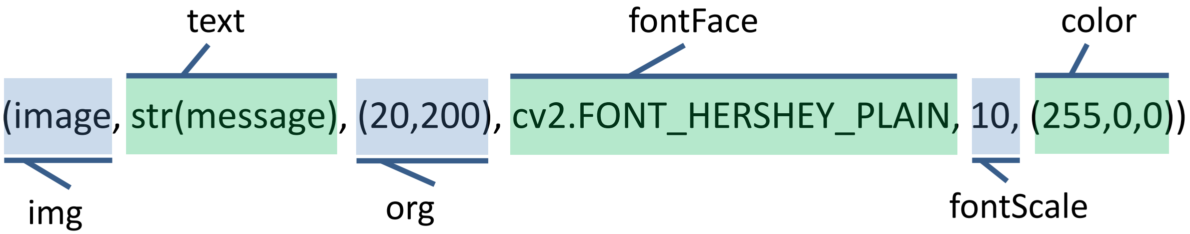
https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html

putText

- Python: cv2.putText(img, text, org, fontScale, color)

- Parameters:
  - **img** – Image.
  - **text** – Text string to be drawn.
  - **org** – Bottom-left corner of the text string in the image.
  - **fontFace** – Font type. One of FONT_HERSHEY_SIMPLEX, FONT_HERSHEY_PLAIN, FONT_HERSHEY_DUPLEX, FONT_HERSHEY_COMPLEX, FONT_HERSHEY_TRIPLEX, FONT_HERSHEY_COMPLEX_SMALL, FONT_HERSHEY_SCRIPT_SIMPLEX, or FONT_HERSHEY_SCRIPT_COMPLEX, where each of the font ID's can be combined with FONT_ITALIC to get the slanted letters.
  - **fontScale** – Font scale factor that is multiplied by the font-specific base size.
  - **color** – Text color.

- Example:
  - cv2.putText(image, str(message), (20,200), cv2.FONT_HERSHEY_PLAIN, 10, (255,0,0))



Outline

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS

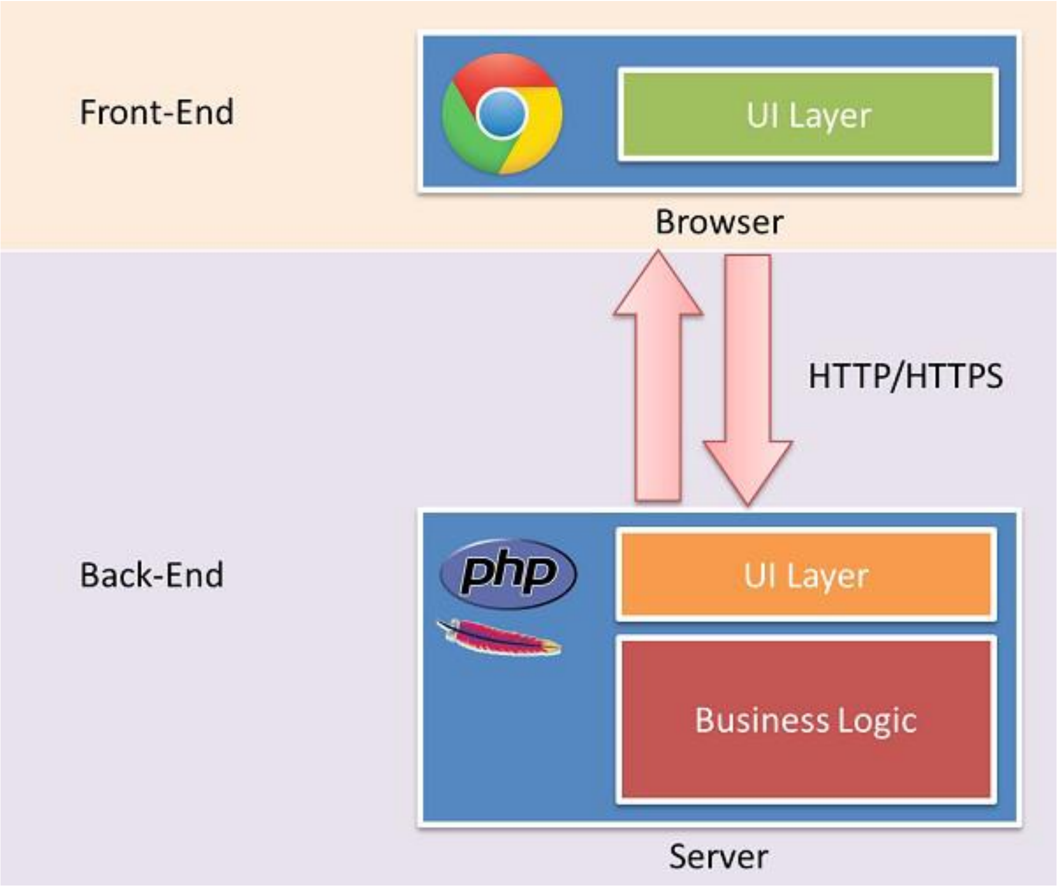
  - 建立網路串流 (IP cam, Video streaming)
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264 (參考用)
    - 使用 RTMP (參考用)

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

1. HTTP + MJPG

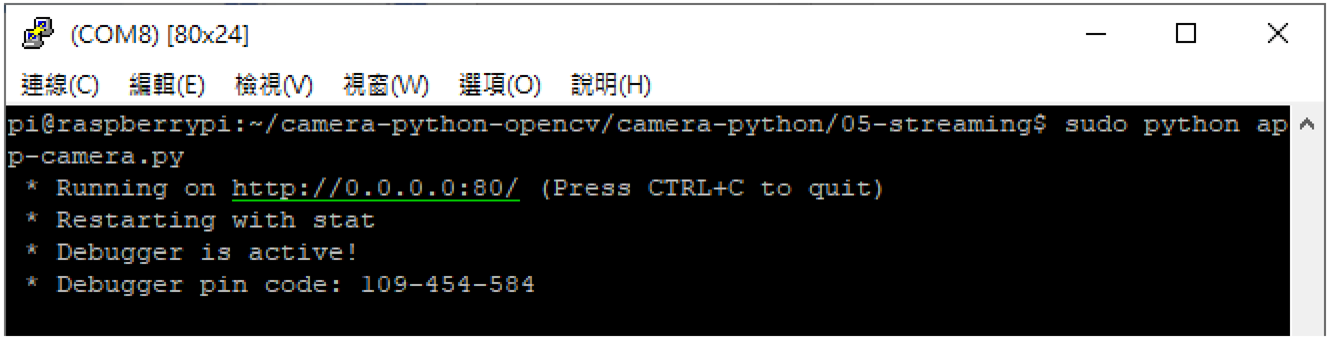
- MJPEG = Motion JPEG
  - 一種視訊壓縮格式
  - 每一個frame都使用JPEG編碼
  - 對運算能力與記憶體的需求較低
  - 許多網頁瀏覽器原生支援M-JPEG

- 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。



1. HTTP + MJPG on PI

- Install tools:
  - sudo apt-get install python-opencv
  - sudo pip install request flask numpy
  - sudo modprobe bcm2835-v4l2
  - Download and unzip “mjpg_sample.zip” file
  - sudo python app-camera.py

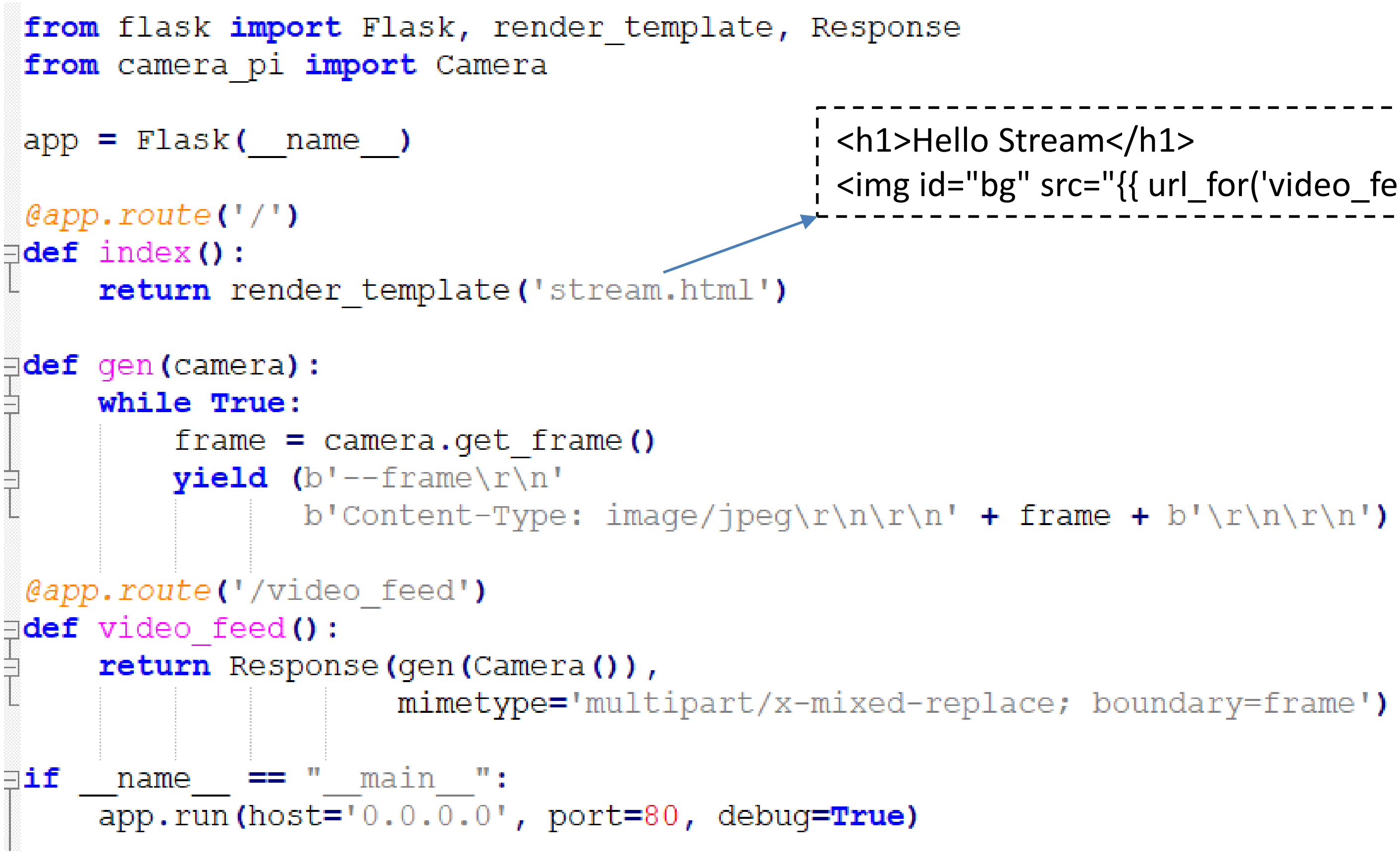


```
pi@raspberrypi:~/camera-python-opencv/camera-python/05-streaming$ sudo python ap
p-camera.py
 * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger pin code: 109-454-584
```

http://flask.pocoo.org/docs/1.0/tutorial/layout/

1. MJPG on PI

- Sample code (app-camera.py)



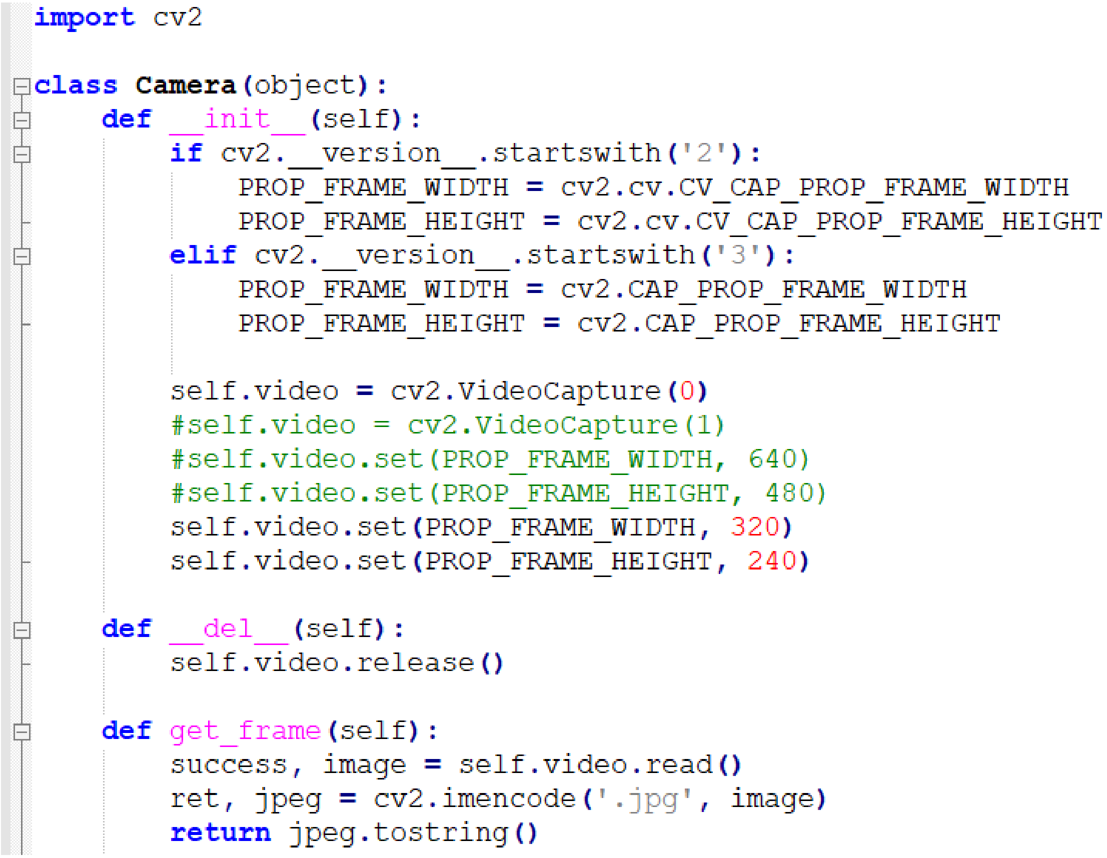
```
from flask import Flask, render_template, Response
from camera_pi import Camera

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') }}">
```

https://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html

1. MJPG on PI

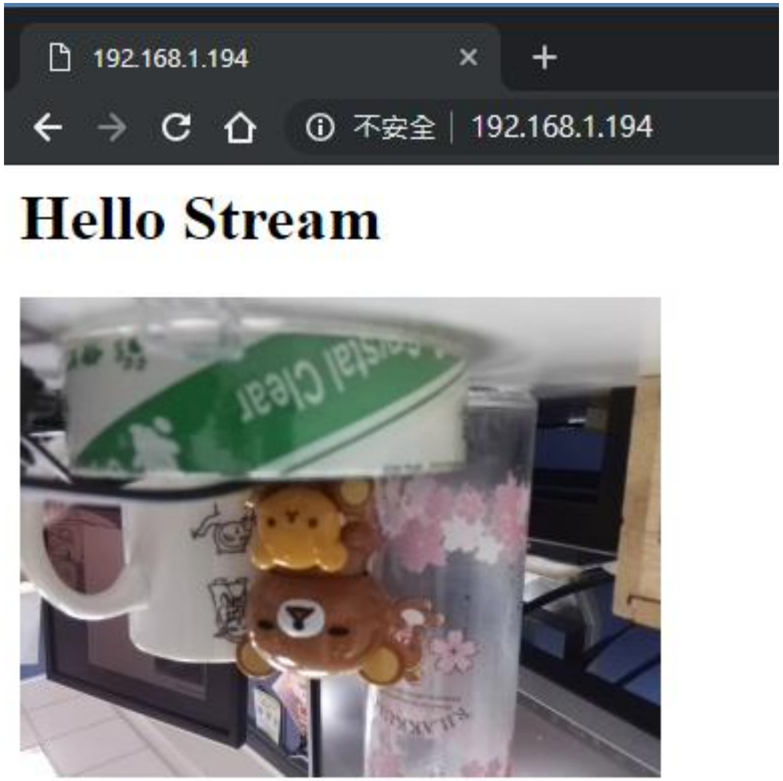
- camera_pi.py



```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()
```

1. MJPG on PI

□ Watch video



Hello Stream

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

Outline

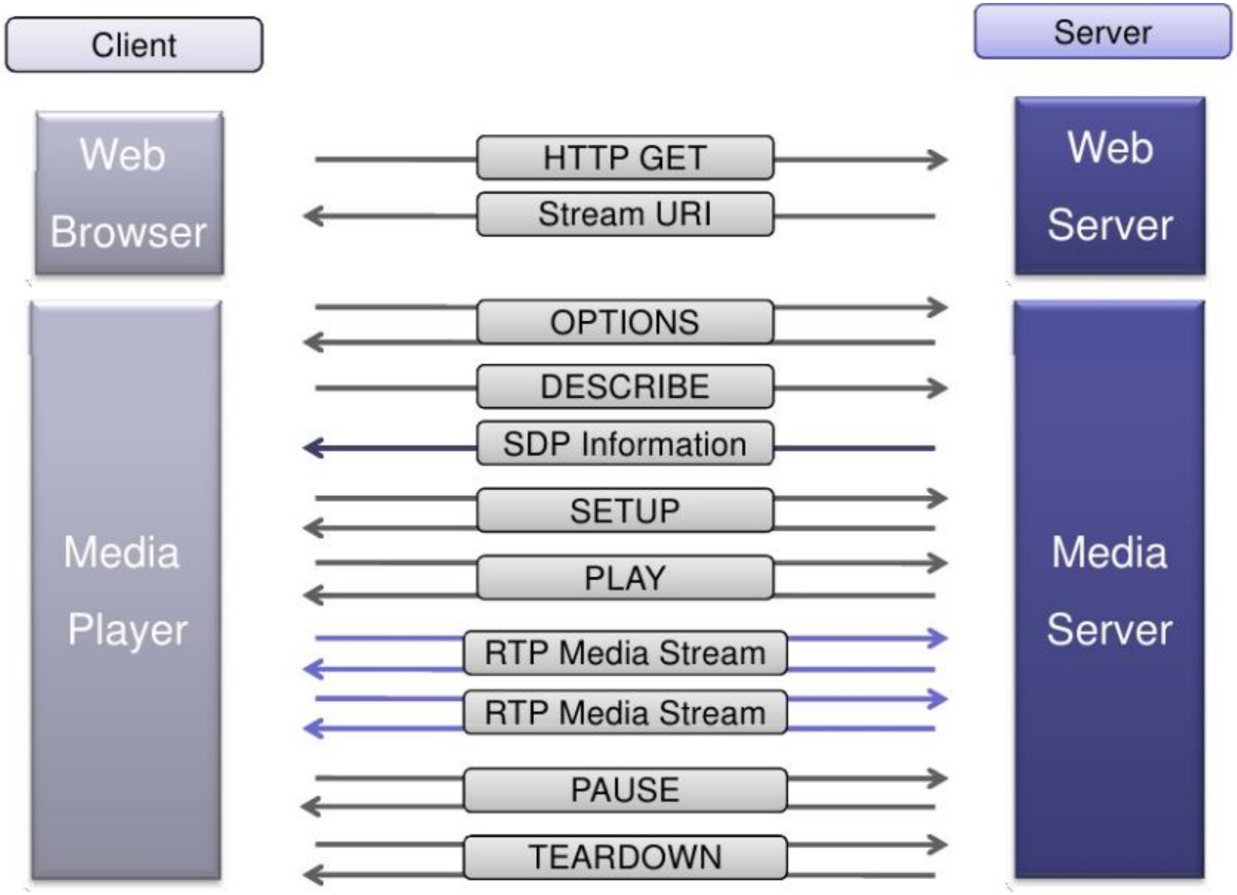
- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS

  - 建立網路串流 (IP cam, Video streaming)
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264
    - 使用 RTMP

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

2. RTSP

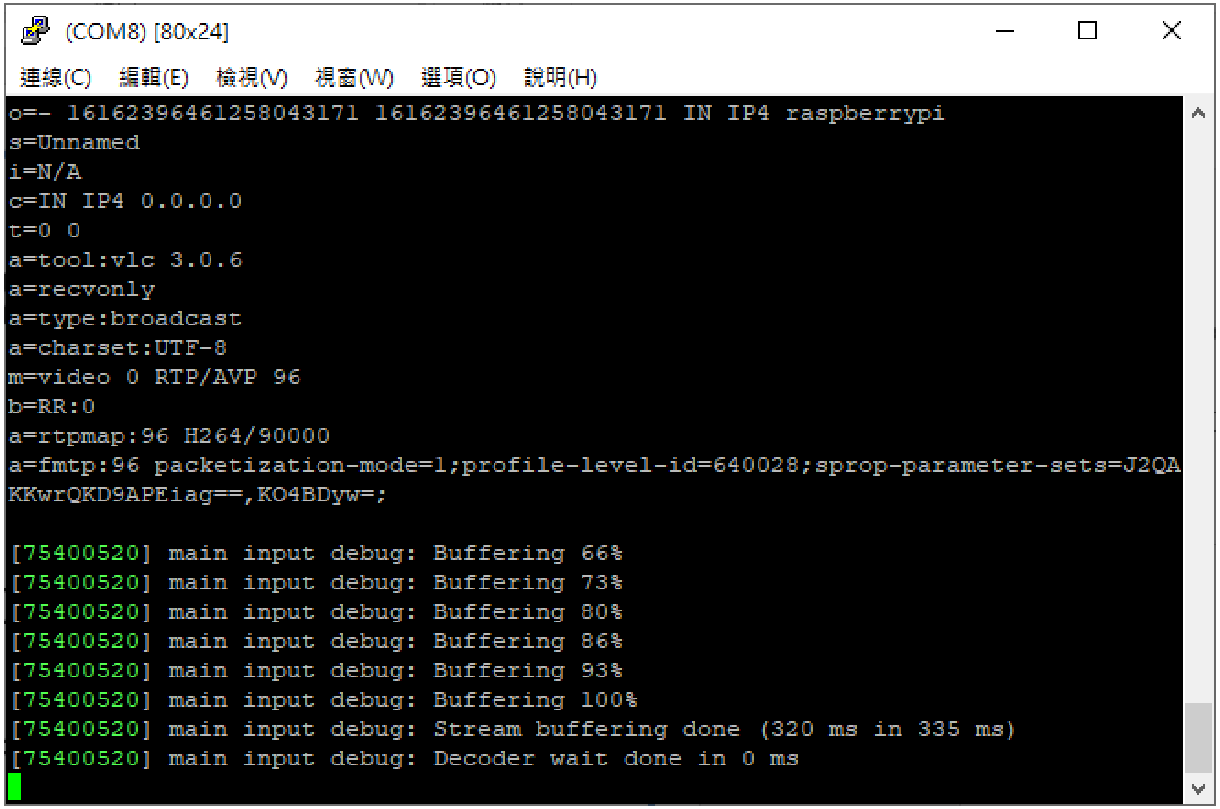
The Real Time Streaming Protocol, or RTSP, is an **application-level protocol** for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both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



2. RTSP on Raspberry PI

□ Execute the command (one line)

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



```
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
```

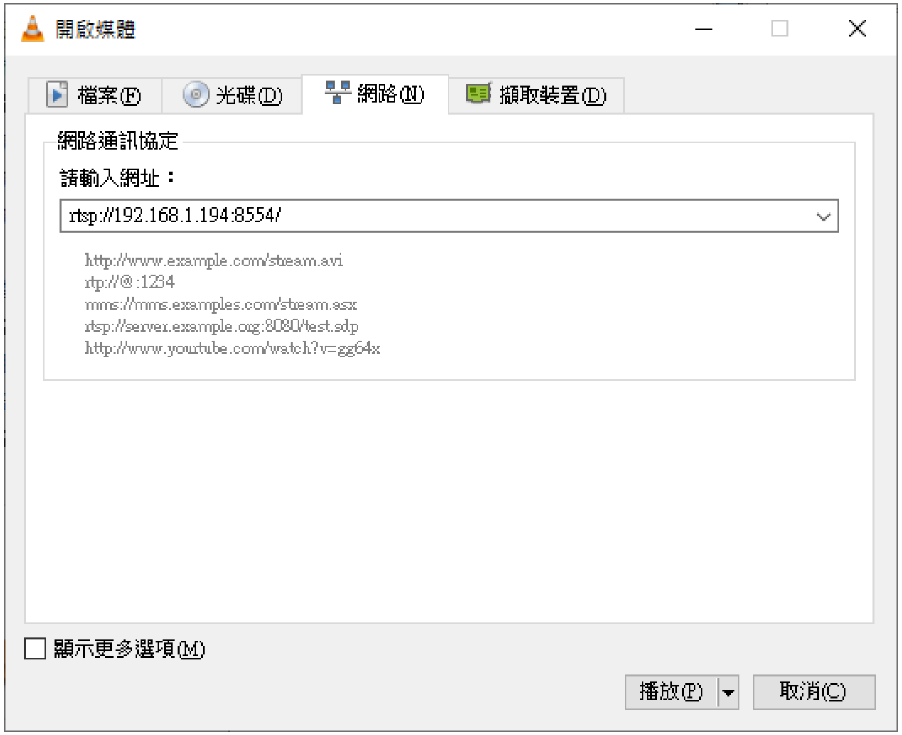
https://wiki.videolan.org/Documentation:Streaming_HowTo/Command_Line_Examples/
https://wiki.videolan.org/VLC_command-line_help/
https://helpmanual.io/help/cvlc/

2. 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

2. RTSP on Raspberry PI

- Use VLC to watch video


開啟媒體

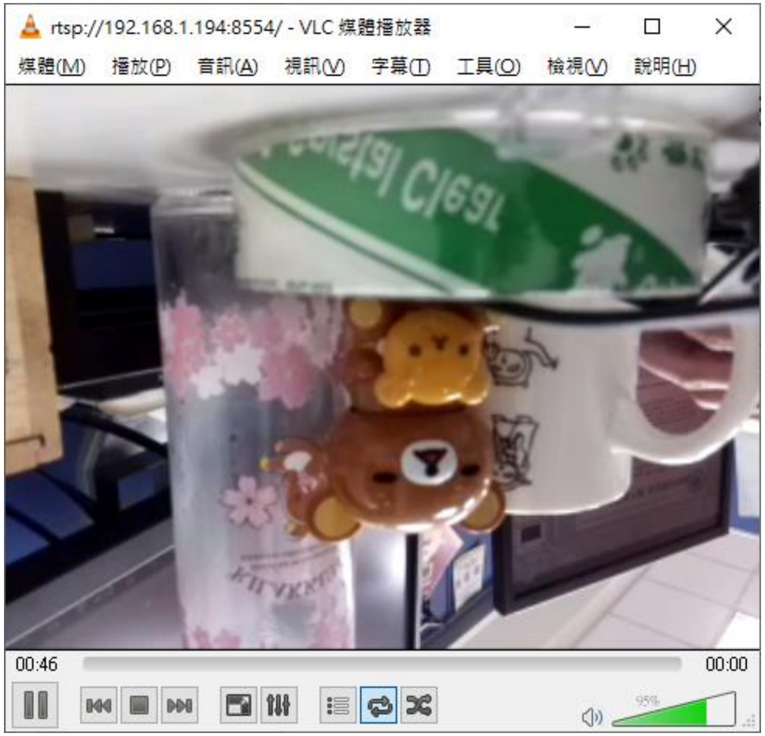
網路通訊協定

請輸入網址：

rtsp://192.168.1.194:8554/

http://www.example.com/steam.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.youtube.com/watch?v=gg64x

□ 顯示更多選項(M)


rtsp://192.168.1.194:8554/ - VLC 媒體播放器

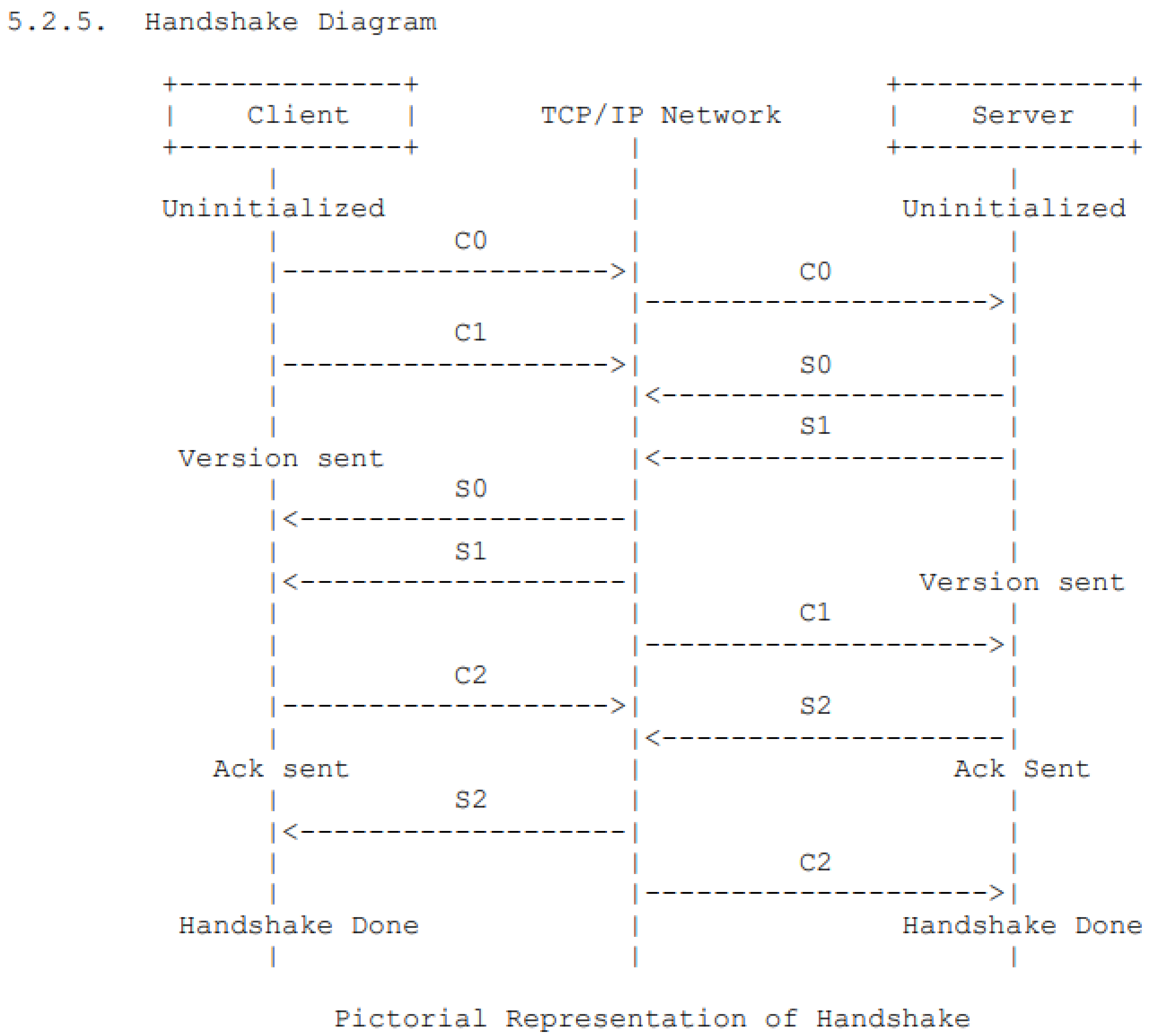
Outline

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS

  - 建立網路串流 (IP cam, Video streaming)
    - 使用 HTTP + MJPG
    - 使用 RTSP + H.264 (參考用)
    - 使用 RTMP (參考用)

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

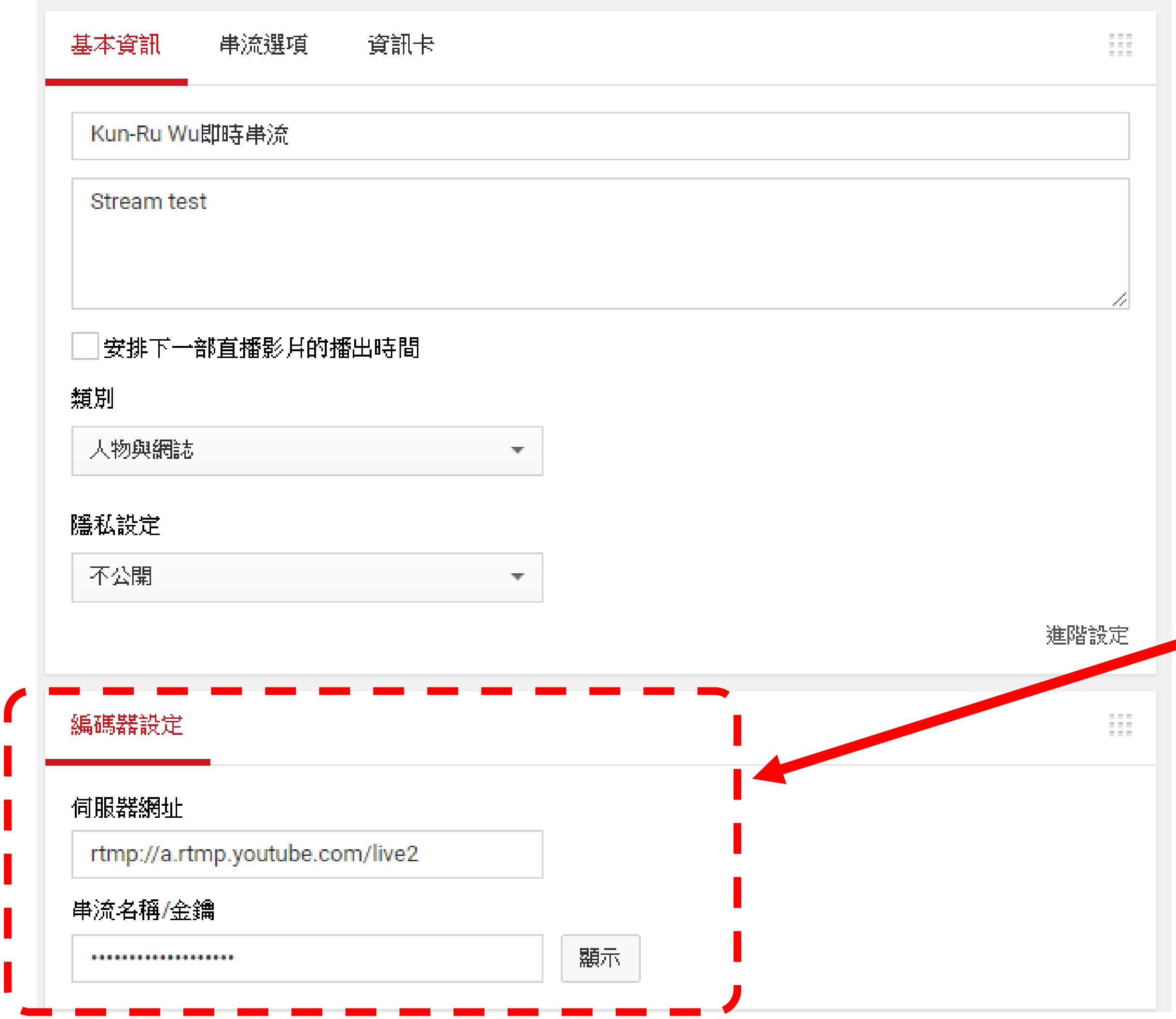
3. RTMP



```
5.2.5.   Handshake Diagram

          +-------------+                            +-------------+
          |   Client    |       TCP/IP Network       |    Server   |
          +-------------+            |               +-------------+
                |                    |                     |
          Uninitialized              |               Uninitialized
                |          C0        |                     |
                |------------------->|         C0          |
                |                    |-------------------->|
                |          C1        |                     |
                |------------------->|         S0          |
                |                    |<--------------------|
                |                    |         S1          |
           Version sent              |<--------------------|
                |          S0        |                     |
                |<-------------------|                     |
                |          S1        |                     |
                |<-------------------|                Version sent
                |                    |         C1          |
                |                    |-------------------->|
                |          C2        |                     |
                |------------------->|         S2          |
                |                    |<--------------------|
             Ack sent                |                  Ack Sent
                |          S2        |                     |
                |<-------------------|                     |
                |                    |         C2          |
                |                    |-------------------->|
           Handshake Done            |              Handshake Done
                |                    |                     |

                   Pictorial Representation of Handshake
```

3. RTMP to Youtube

https://www.youtube.com/live_dashboard



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

編碼器設定

伺服器網址

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

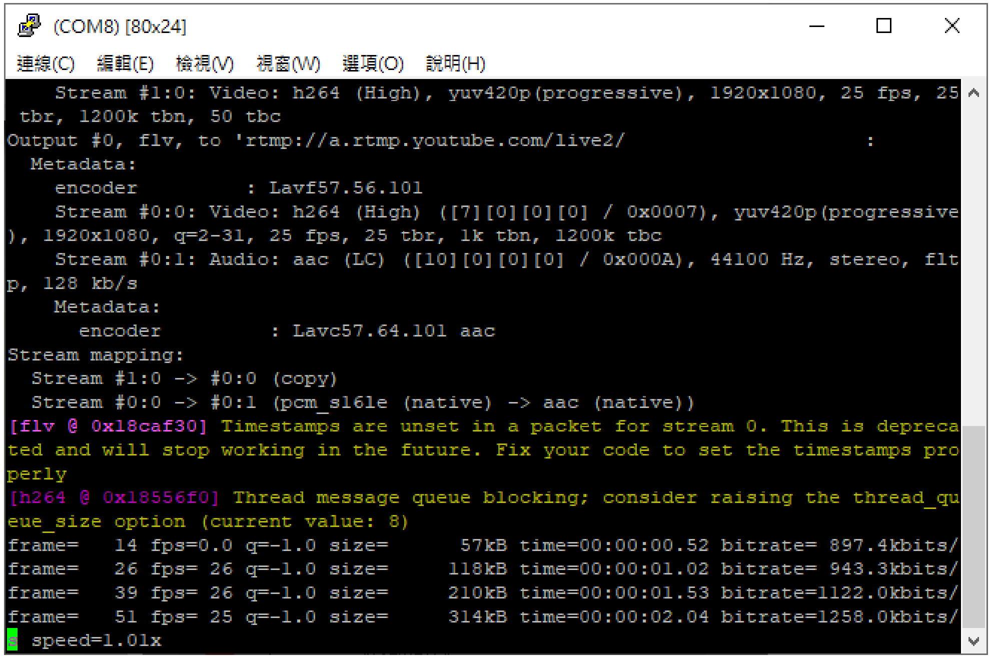
串流名稱/金鑰

••••••••••••••••••••

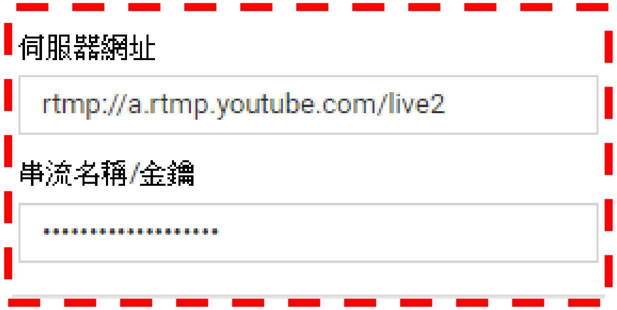
3. RTMP on PI

- □ 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
```



```
    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/[redacted]':
  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, flt
p, 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 depreca
ted and will stop working in the future. Fix your code to set the timestamps pro
perly
[h264 @ 0x18556f0] Thread message queue blocking; consider raising the thread_qu
eue_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/
s speed=1.01x
```



伺服器網址

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

串流名稱/金鑰

••••••••••••••••••••

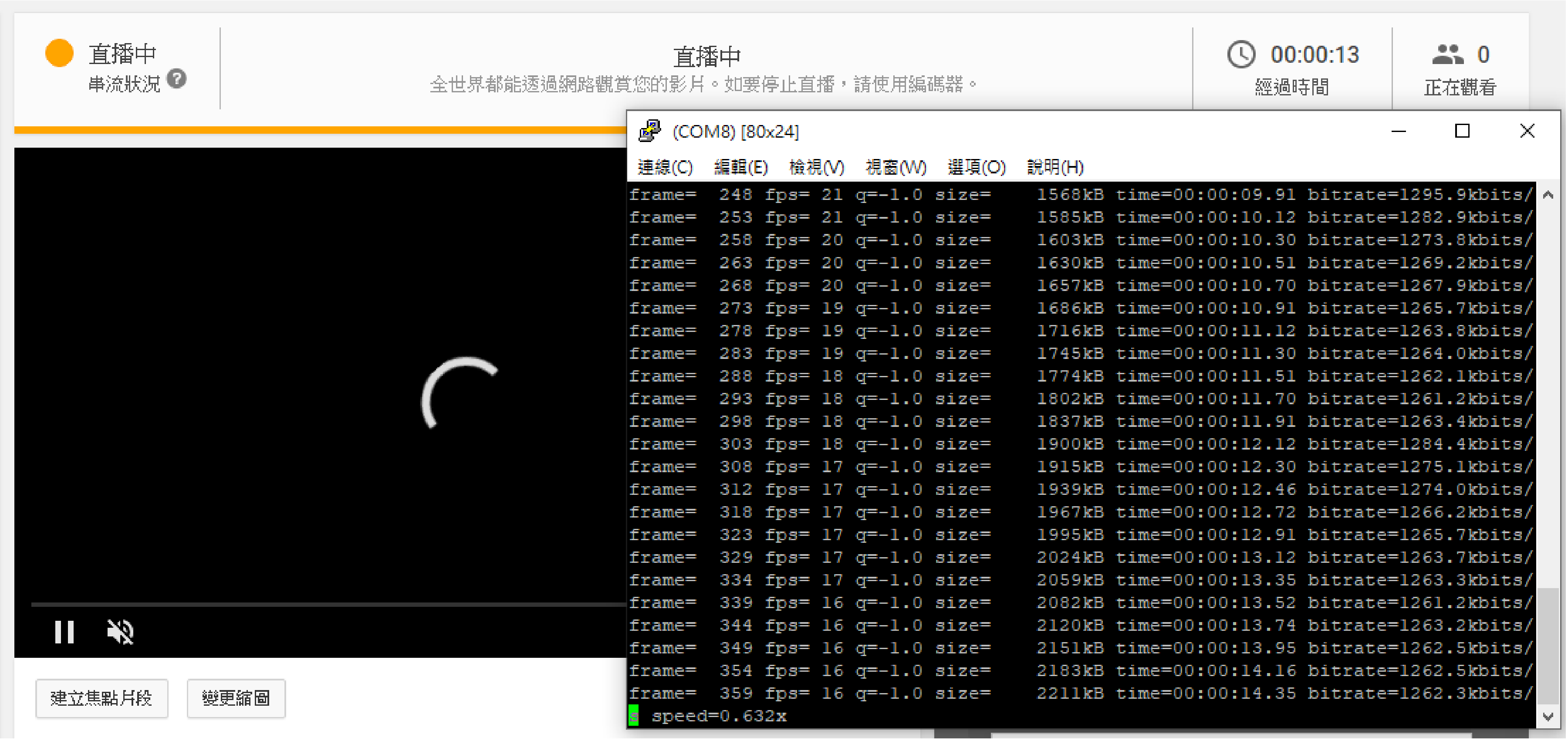
https://ffmpeg.org/ffmpeg.html

3. 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.

3. RTMP on PI

- Start streaming…



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。

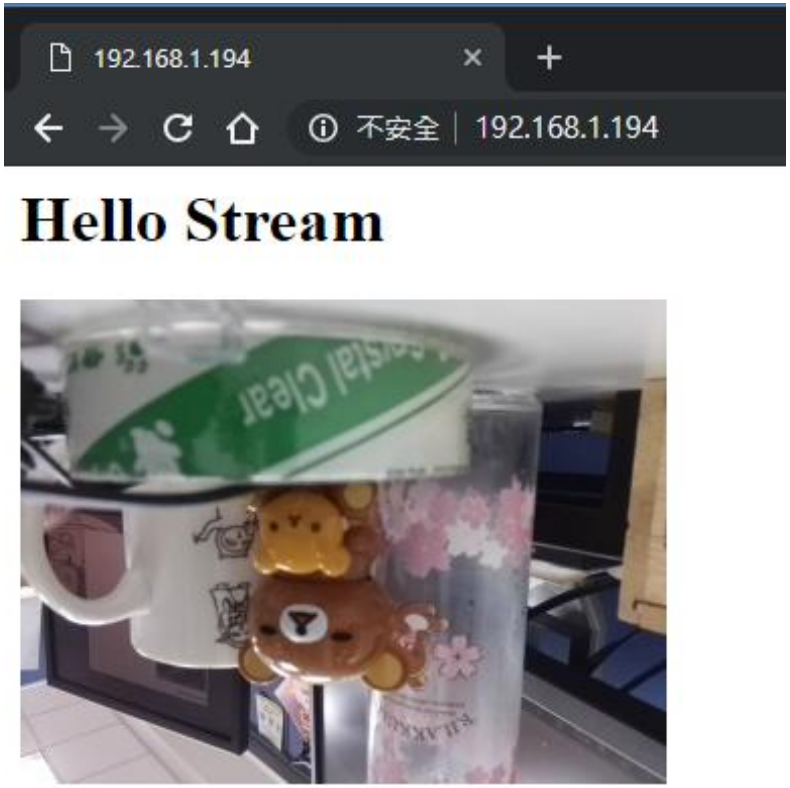
```
frame=  248 fps= 21 q=-1.0 size=    1568kB time=00:00:09.91 bitrate=1295.9kbits/
frame=  253 fps= 21 q=-1.0 size=    1585kB time=00:00:10.12 bitrate=1282.9kbits/
frame=  258 fps= 20 q=-1.0 size=    1603kB time=00:00:10.30 bitrate=1273.8kbits/
frame=  263 fps= 20 q=-1.0 size=    1630kB time=00:00:10.51 bitrate=1269.2kbits/
frame=  268 fps= 20 q=-1.0 size=    1657kB time=00:00:10.70 bitrate=1267.9kbits/
frame=  273 fps= 19 q=-1.0 size=    1686kB time=00:00:10.91 bitrate=1265.7kbits/
frame=  278 fps= 19 q=-1.0 size=    1716kB time=00:00:11.12 bitrate=1263.8kbits/
frame=  283 fps= 19 q=-1.0 size=    1745kB time=00:00:11.30 bitrate=1264.0kbits/
frame=  288 fps= 18 q=-1.0 size=    1774kB time=00:00:11.51 bitrate=1262.1kbits/
frame=  293 fps= 18 q=-1.0 size=    1802kB time=00:00:11.70 bitrate=1261.2kbits/
frame=  298 fps= 18 q=-1.0 size=    1837kB time=00:00:11.91 bitrate=1263.4kbits/
frame=  303 fps= 18 q=-1.0 size=    1900kB time=00:00:12.12 bitrate=1284.4kbits/
frame=  308 fps= 17 q=-1.0 size=    1915kB time=00:00:12.30 bitrate=1275.1kbits/
frame=  312 fps= 17 q=-1.0 size=    1939kB time=00:00:12.46 bitrate=1274.0kbits/
frame=  318 fps= 17 q=-1.0 size=    1967kB time=00:00:12.72 bitrate=1266.2kbits/
frame=  323 fps= 17 q=-1.0 size=    1995kB time=00:00:12.91 bitrate=1265.7kbits/
frame=  329 fps= 17 q=-1.0 size=    2024kB time=00:00:13.12 bitrate=1263.7kbits/
frame=  334 fps= 17 q=-1.0 size=    2059kB time=00:00:13.35 bitrate=1263.3kbits/
frame=  339 fps= 16 q=-1.0 size=    2082kB time=00:00:13.52 bitrate=1261.2kbits/
frame=  344 fps= 16 q=-1.0 size=    2120kB time=00:00:13.74 bitrate=1263.2kbits/
frame=  349 fps= 16 q=-1.0 size=    2151kB time=00:00:13.95 bitrate=1262.5kbits/
frame=  354 fps= 16 q=-1.0 size=    2183kB time=00:00:14.16 bitrate=1262.5kbits/
frame=  359 fps= 16 q=-1.0 size=    2211kB time=00:00:14.35 bitrate=1262.3kbits/
s speed=0.632x
```

3. RTMP on PI

- □ Watch video

Quiz 3

- Use “PiVideoStream” to speed up MJPG stream
  - Modify “camera_pi.py”
  - Refer to 3picamera.py
  - Show your code to TA



192.168.1.194

不安全 | 192.168.1.194

Hello Stream

Summary

- Practice Lab (PI camera)
- Write down the answer for discussion
  - **Discussion:**
    - **1. Read the online document. How do we set filename in the code?**
  - Deadline: Before 4/16, 12:00 (before next class)

- Write code for Quiz 1 - 3, then demonstrate it to TAs
  - **Quiz1: Timeslape**
  - **Quiz2: show current FPS on the frame (for each capture method)**
  - **Quiz3: Use "PiVideoStream" to speed up MJPG stream**
    - Deadline: Before 4/9, 15:10
    - Late Demo: Before 15:10, 4/16 (before next class)