

Raspberry Pi Camera + Python

台灣樹莓派 <sosorry@raspberrypi.com.tw>
2019/05/04 @NUTC

CC (Creative Commons)

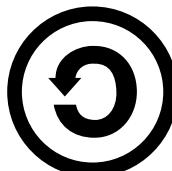
姓名標示 — 非商業性 — 相同方式分享



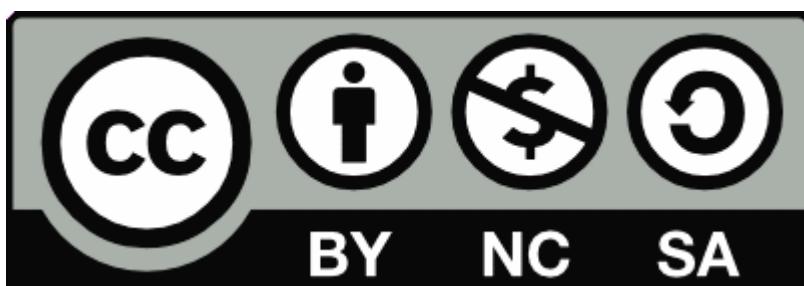
姓名標示 — 你必須給予 適當表彰、提供指向本授權條款的連結，以及 指出（本作品的原始版本）是否已被變更。你可以任何合理方式為前述表彰，但不得以任何方式暗示授權人為你或你的使用方式背書。



非商業性 — 你不得將本素材進行商業目的之使用。



相同方式分享 — 若你重混、轉換本素材，或依本素材建立新素材，你必須依本素材的授權條款來散布你的貢獻物。



關於我們

- Raspberry Pi 官方經銷商



about 台灣樹莓派

- 專注於 Raspberry Pi 應用與推廣
- 舉辦社群聚會 / 工作坊 / 讀書會 / 黑客松
- Website:
 - <https://www.raspberrypi.com.tw/>
- Facebook:
 - 搜尋 RaspberryPi.Taiwan
 - <https://www.facebook.com/RaspberryPi.Taiwan>



分享 x 教學

- COSCUP, MakerConf, PyCon 講者
- 投影片
 - <http://www.slideshare.net/raspberrypi-tw/presentations>
- 程式碼
 - <https://github.com/raspberrypi-tw>



學習路徑

 Pi選購指南

 Pi設定安裝

 Linux系統管理

I/O硬體控制

GPIO學習套件 初

感測器學習套件
(基礎/進階) 中

空氣盒子套件
(PiM25) 初

Win10開發套件 初

智慧開關套件 初

Linux Driver
學習套件 進

無線/IoT

RFID/NFC
門禁系統 初

LoRa IoT
閘道器套件 初

生理資訊
監控IoT(藍牙) 初

毫米波人流/熱點監控
(mmWave) 初

相機/影像處理

特色相機改裝套件 初

寵物小車套件 初

自控機器手臂套件 中

小鴨車套件
(Duckietown) 進

語音/訊號處理

智慧音箱套件 初

AIY Voice Kit 初

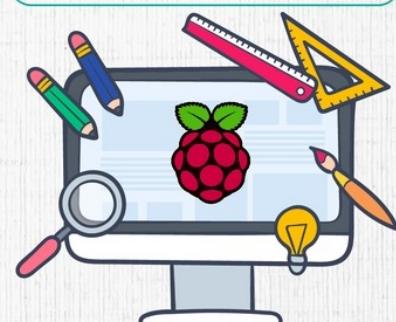
人工智慧

驢車套件
(DonkeyCar) 初

AIY Vision Kit 中

Intel神經運算棒 中

Google Coral
USB加速器 中



初 初階課程

中 中階課程

進 進階課程

大綱

- 相機原理與應用
- 控制 Raspberry Pi Camera
 - 使用指令列
 - 使用 Python
 - 串接 imagga 網路服務
- Camera 和 Webcam
- 影像串流
 - 使用 RTSP + H.264
 - 使用 HTTP + MJPG

今日環境

- 硬體 : Raspberry Pi 3B/3B+
- 作業系統 : 2018-11-13-raspbian-stretch.img
- 為了可以使用USB轉TTL傳輸線
 - 修改 /boot/config.txt , 新增三行
 - dtoverlay=pi3-miniuart-bt
 - core_freq=250
 - enable_uart=1
- 修改 /boot/cmdline.txt , 將 quiet splash 的 quiet 移除

```
55 # Enable audio (loads snd_bcm2835)
56 dtparam=audio=on
57 dtoverlay=pi3-miniuart-bt
58 core_freq=250
59 enable_uart=1
```

新增三行

```
1 dwc_otg.lpm_enable=0 console=serial0,115200
    console=tty1 root=/dev/mmcblk0p2 rootfstype=
    ext4 elevator=deadline fsck.repair=yes rootw
    ait quiet splash plymouth.ignore-serial-con
    sole quiet init=/usr/lib/raspi-config/init_r
    esize.sh
```

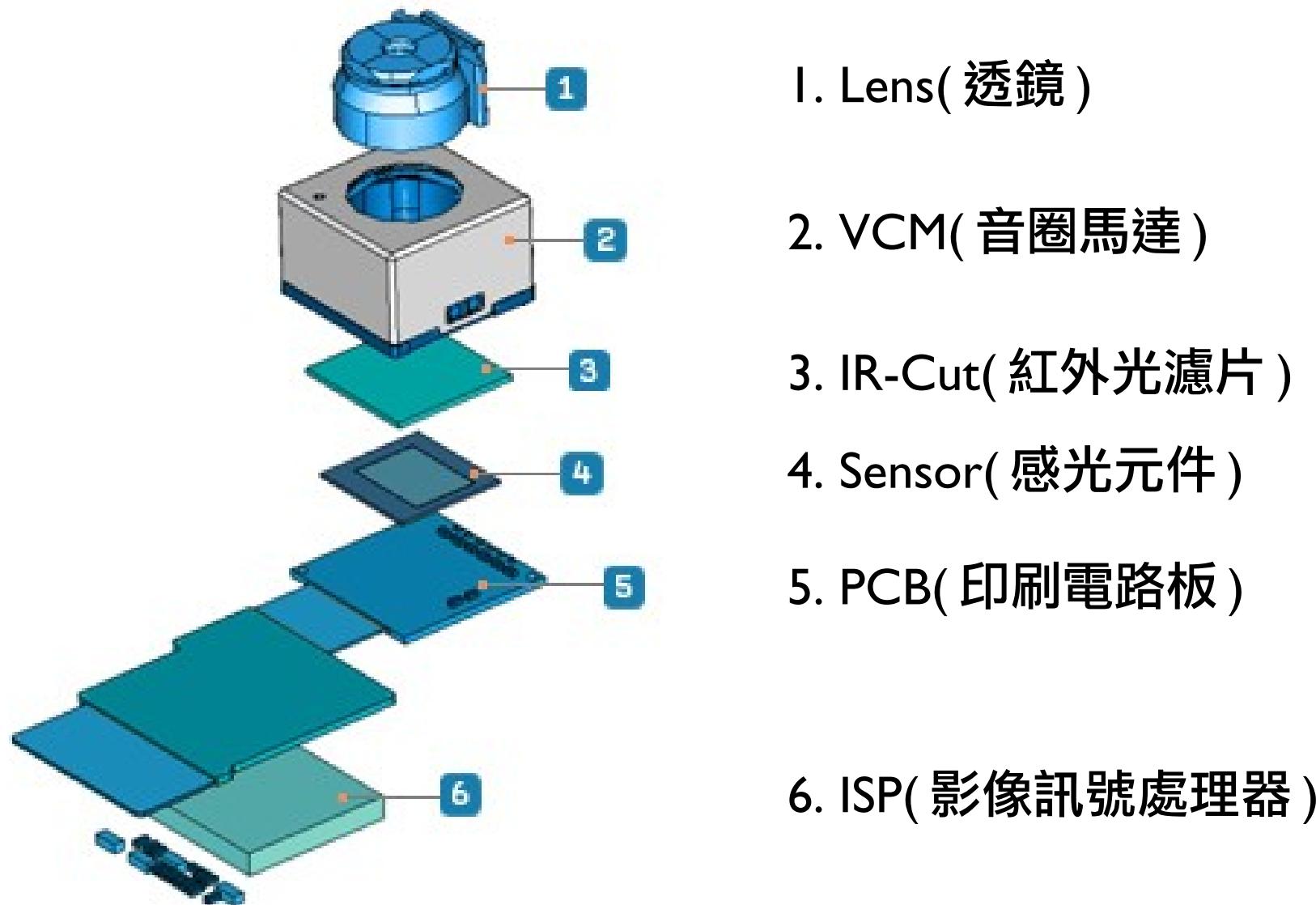
刪除 quiet

安裝今日所需軟體

- \$ sudo apt-get update
- \$ sudo apt-get install -y festival
python-dev python-opencv python-pip
x11vnc liblivemedia-dev libv4l-dev
cmake libpthread-stubs0-dev vlc
python-matplotlib
- \$ sudo pip2 install requests flask
numpy

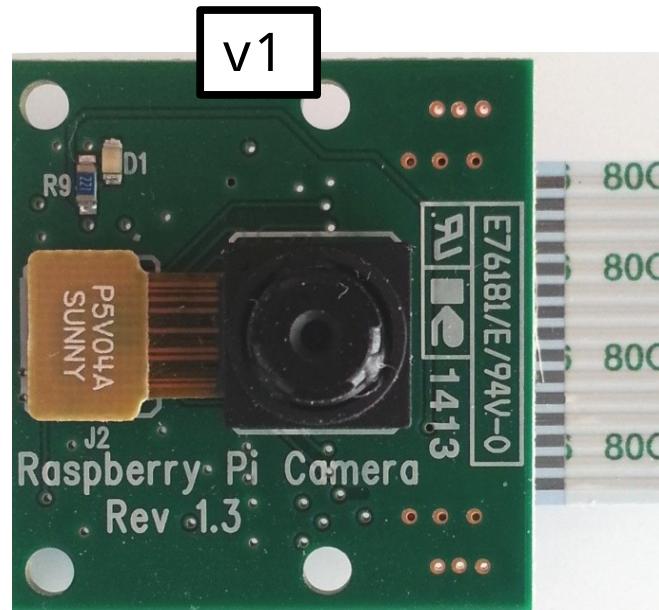
Raspberry Pi Camera 簡介

從手機相機模組講起

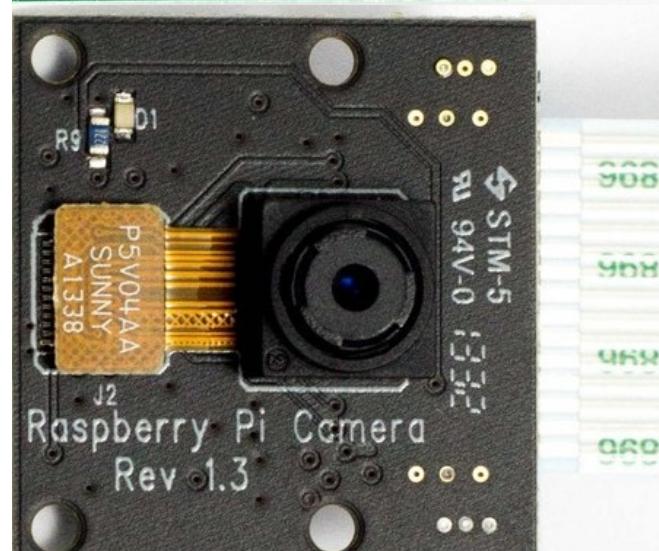


Type of Raspberry Pi Camera

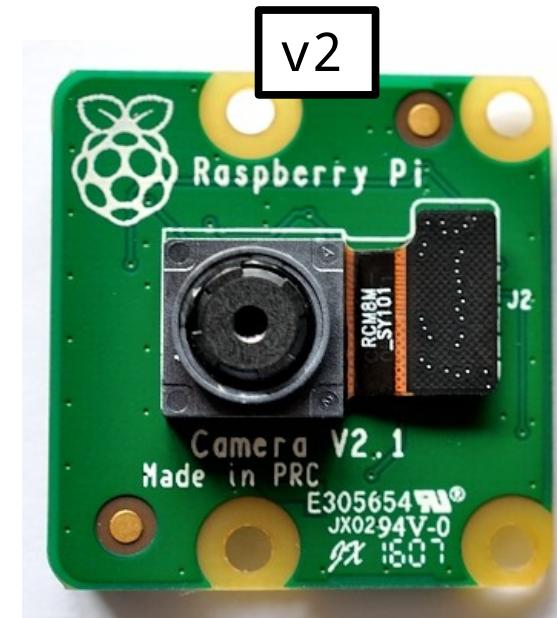
Raspberry Pi
Camera Module



NoIR Camera Module



v2



Raspberry Pi Camera 技術規格 (v1)

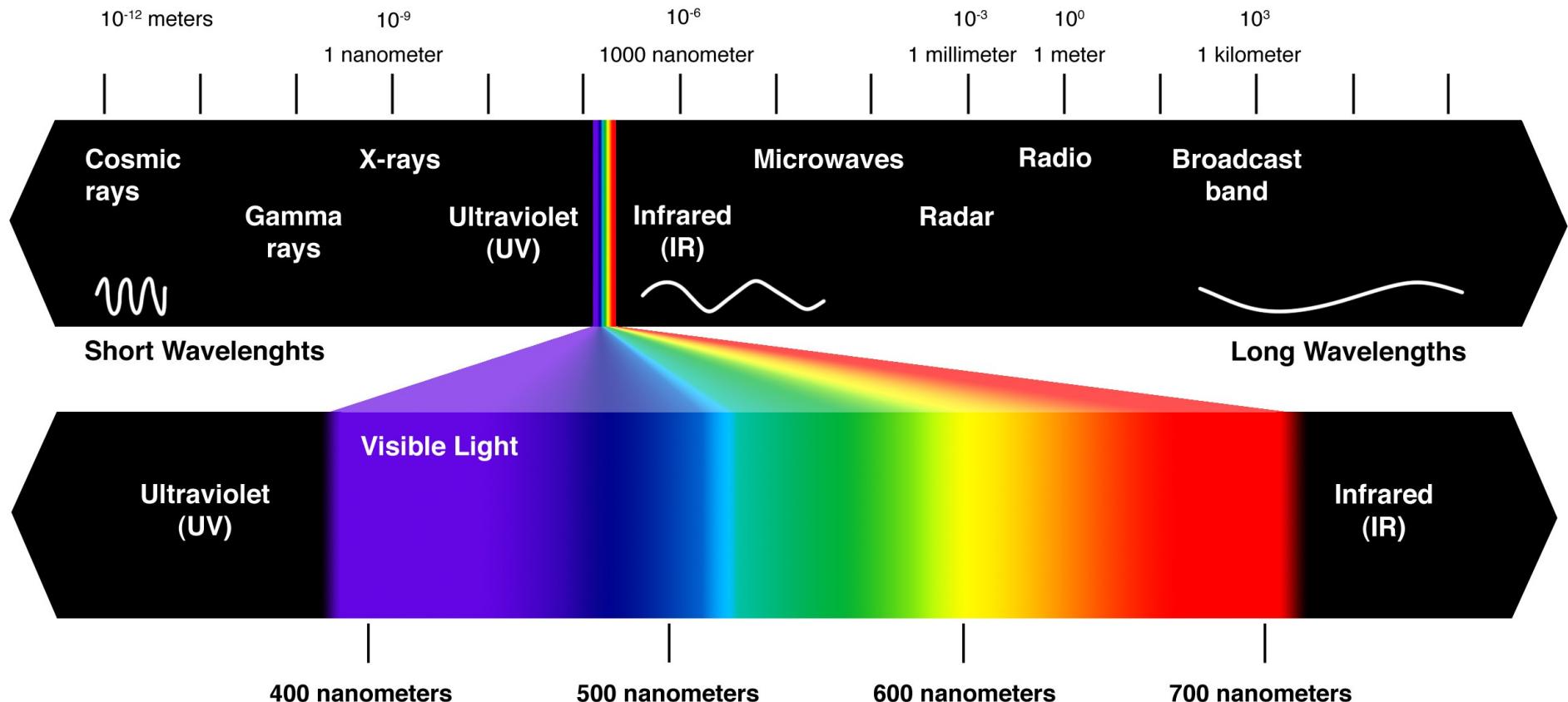
- Sensor: OmniVision OV5647 (5MP)
- 靜態拍照最高解析度 : 2592 x 1944 pixel
- Pixel Size: 1.4 x 1.4 μm
- Lens: f=3.6 mm, f/2.9
- Angle of View: 54 x 41 degrees
- Field of View: 2.0 x 1.33 m at 2 m
- Fixed Focus: 1m to infinity
- 動態攝影最高解析度 : 1080p@30 FPS with H.264/AVC

Raspberry Pi Camera 技術規格 (v2)

- Sensor: Sony IMX219 (8MP)
- 靜態拍照最高解析度 : 3280 x 2464 pixel
- Pixel Size: 1.4 x 1.4 μm
- Lens: f=3.6 mm, f/2.9
- Fixed Focus: 1m to infinity
- 動態攝影最高解析度 : 1080p@30 FPS with H.264/AVC

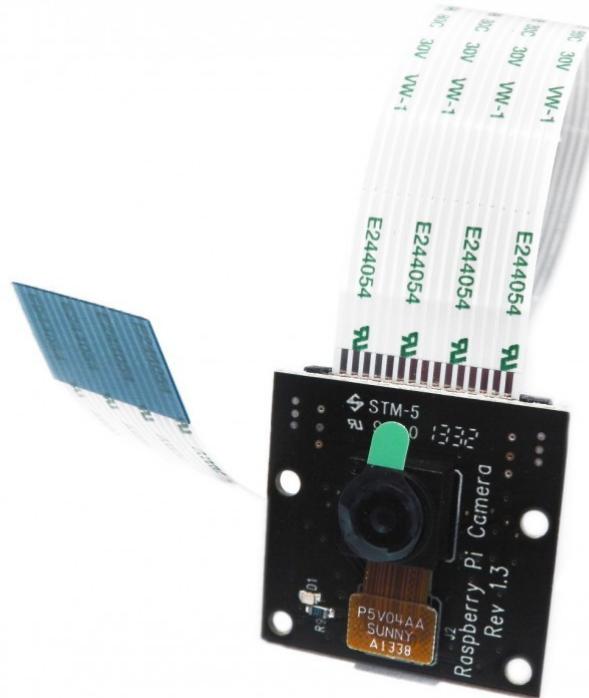
基礎光學原理

- 問：樹葉為什麼看起來是綠色的？
- 答：因為樹葉吸收了大部分可見光，只反射綠色光



No IR Camera

- No IR = No 'IR cut filter' installed
 - 因此 CMOS 可吸收到不可見光 (Infrared)
 - No IR 相機 + 紅外線發光源 = 夜視相機



黑色 PCB 板



外接式紅外光



可控式紅外光

<https://www.buyapi.ca/product/raspberry-pi-noir-camera-module-v2-8mp/>

兩種相機效果比較



1. 非 NoIR 相機



2. NoIR 相機

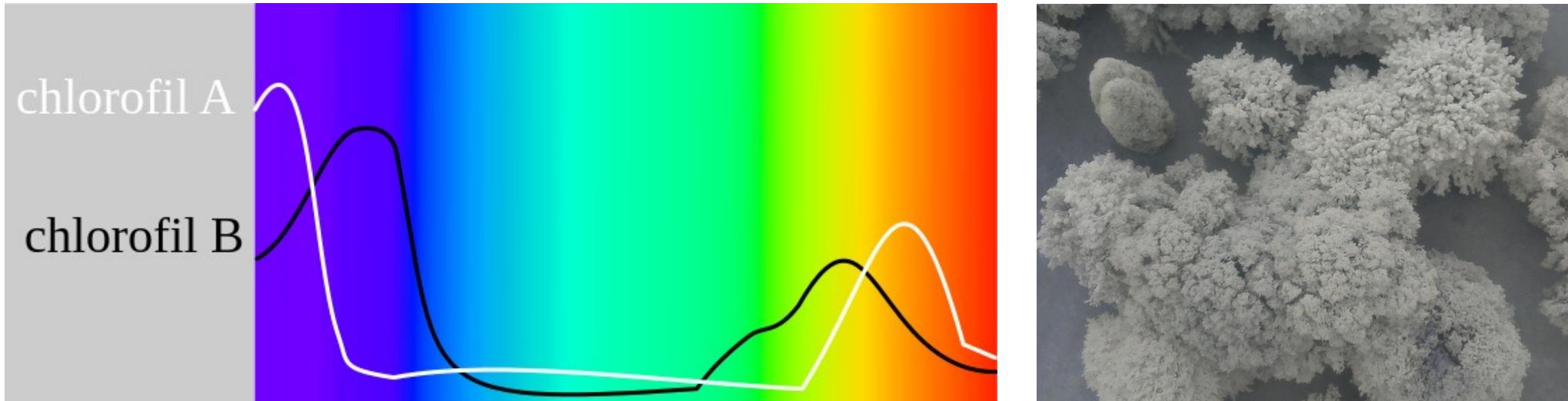


3. NoIR 相機



4. NoIR 相機 + 藍色濾光片

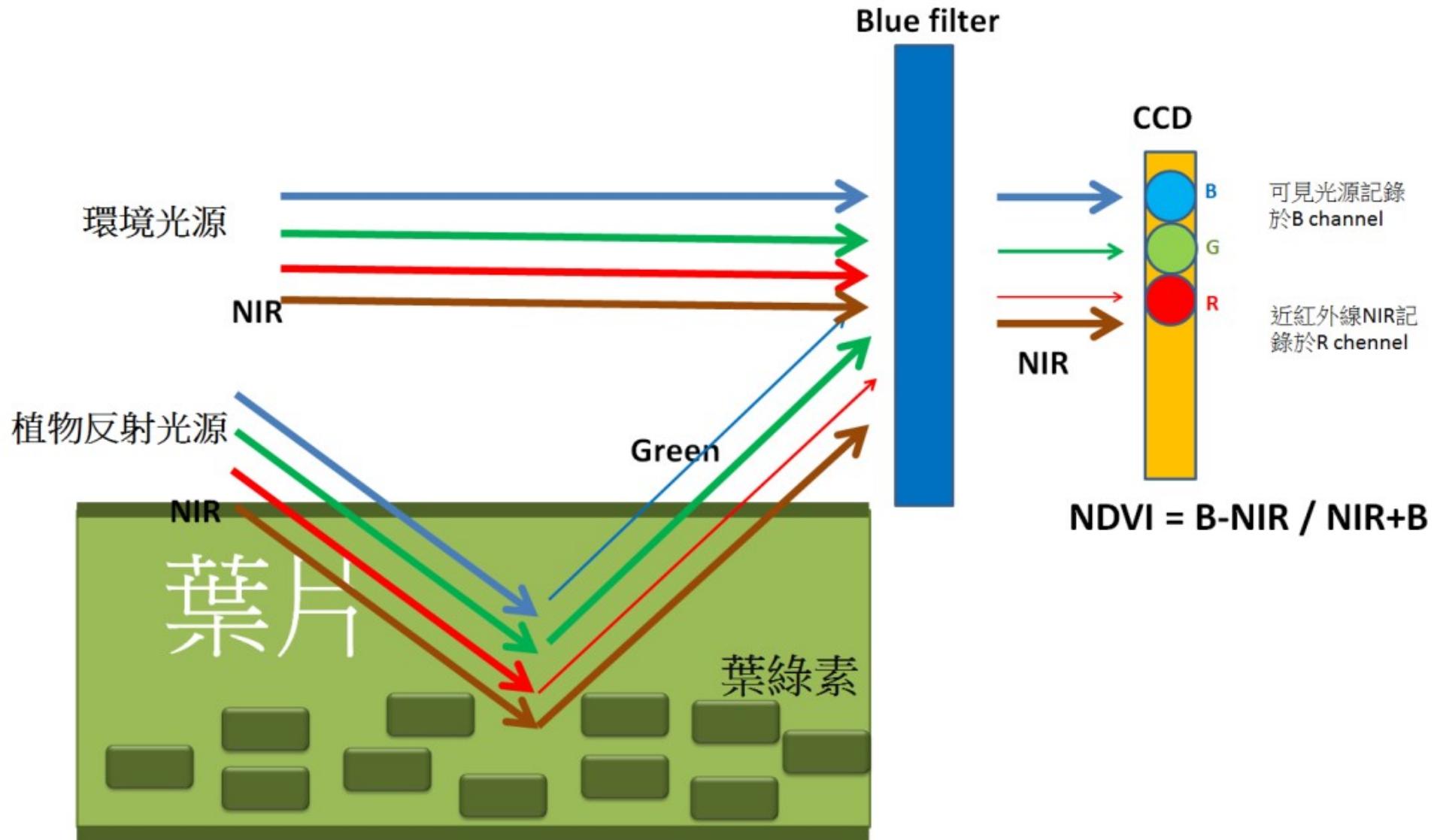
用 NoIR 相機量測植物生長指標 NDVI



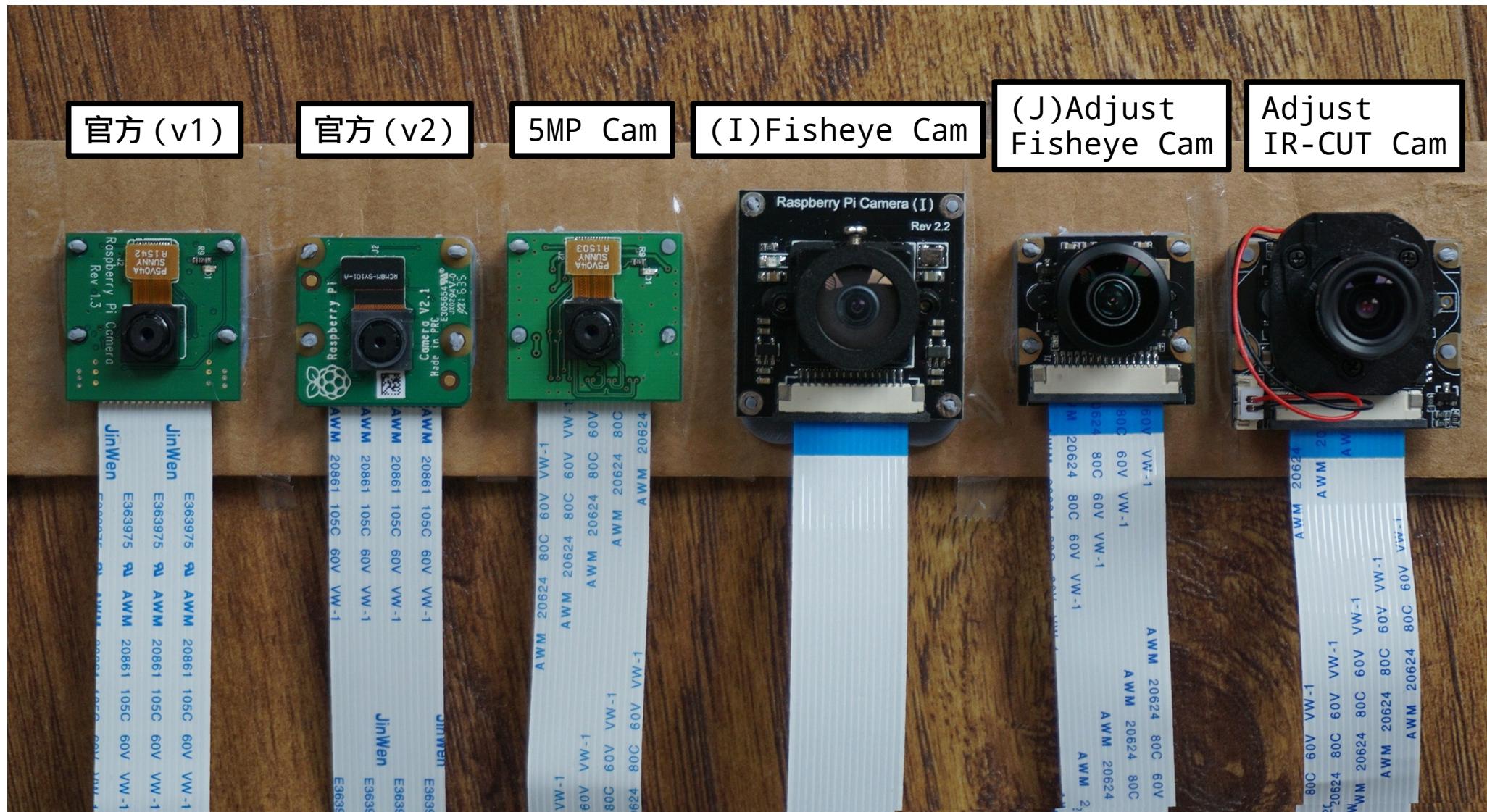
- NDVI 可顯示植物生長，生態系活力與生產力資訊
- 測量光合作用的進行量 = 尋找綠色或是找紅色
- 計算公式為 $NDVI = \frac{(NIR - Red)}{(NIR + Red)}$

<https://www.raspberrypi.org/blog/whats-that-blue-thing-doing-here/>

藍色濾光片應用在計算 NDVI



更多 Camera



<http://www.semifluid.com/2017/01/23/raspberry-pi-camera-comparison/>

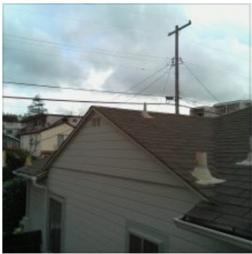
Raspberry Pi Camera Comparison

semifluid.com

Raspberry Pi
Camera



Arducam 5MP
RPi Camera



Waveshare RPi
Camera IR-CUT (on)



Waveshare RPi
Camera IR-CUT (off)



Waveshare
RPi Camera (I)



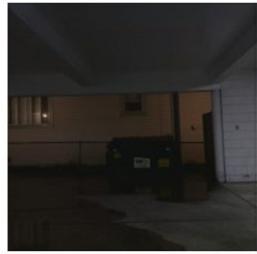
Waveshare
RPi Camera (J)



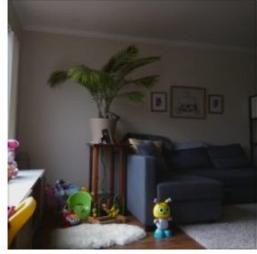
Raspberry Pi
v2 Camera



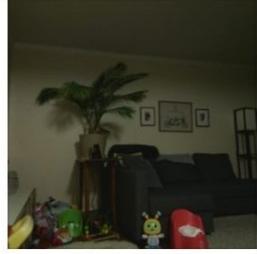
Outdoor
(day)



Indoor
(day)

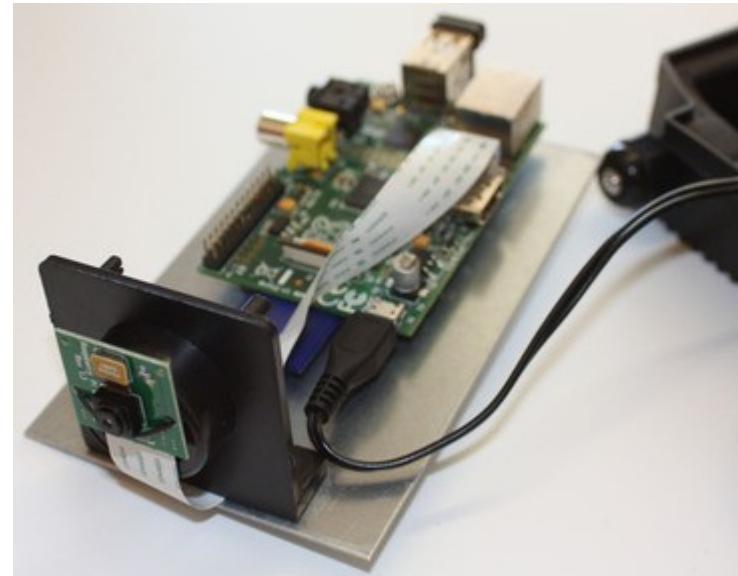


Indoor
(night)



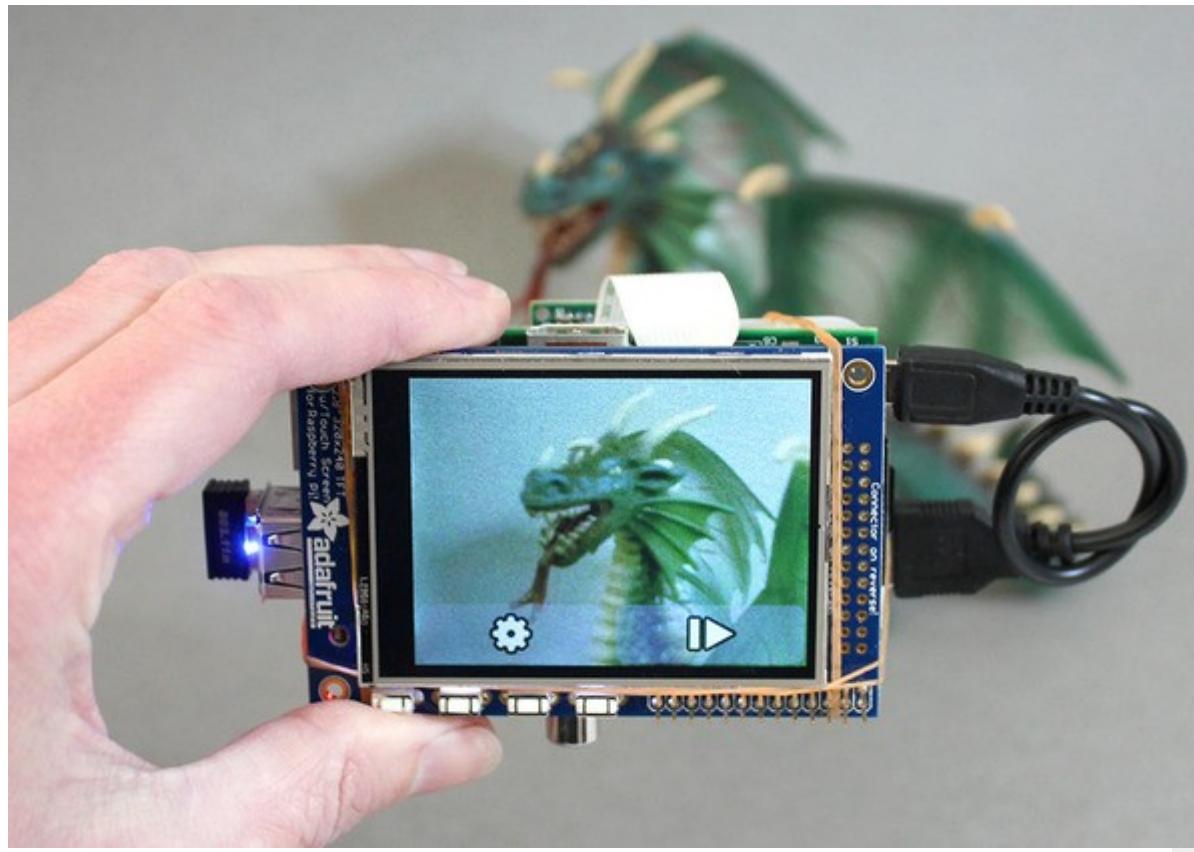
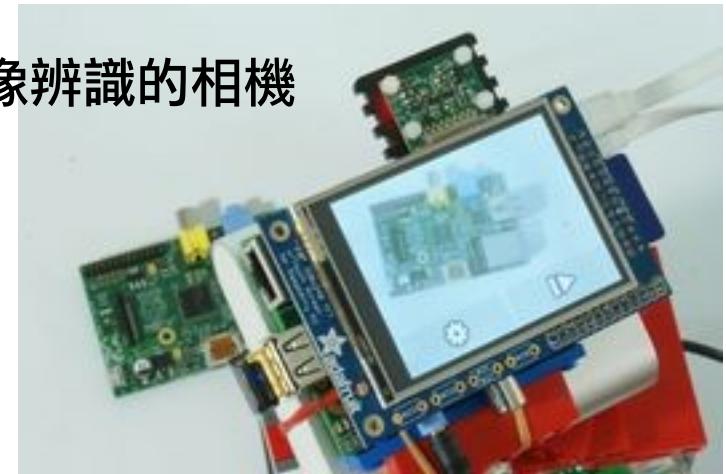
Raspberry Pi Camera 應用介紹

IP Camera



雲端相機

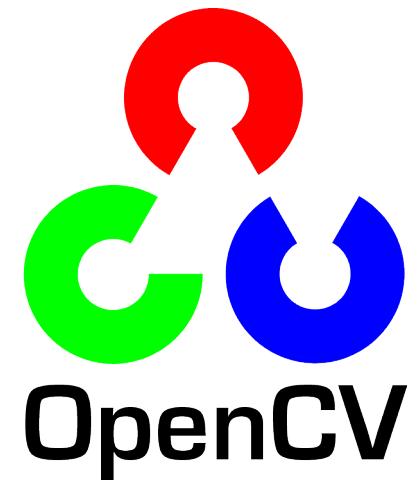
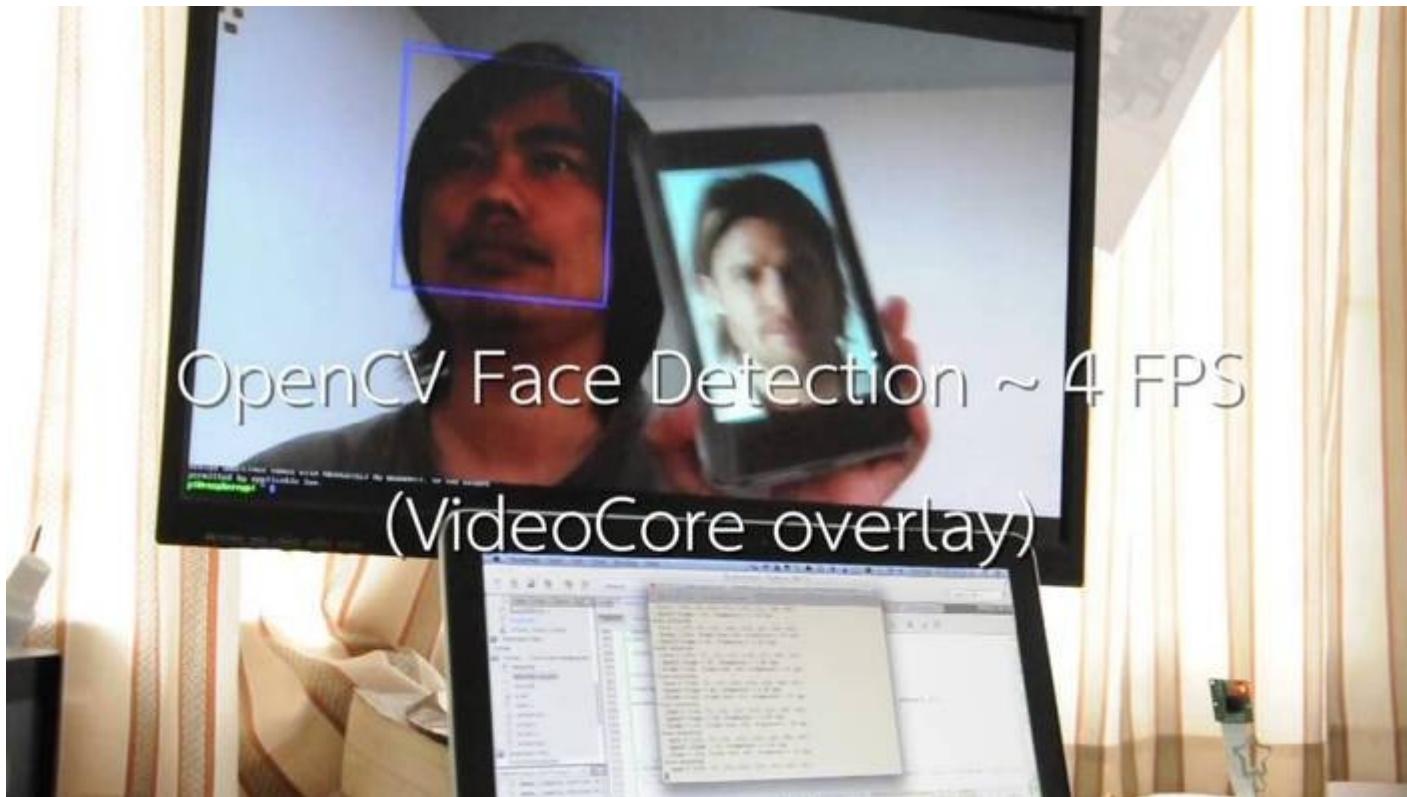
可做影像辨識的相機



Pi 立得

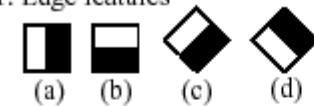


人臉偵測與追蹤

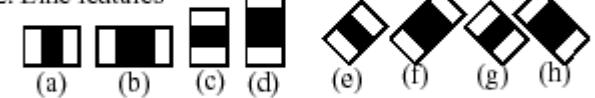


Haar	Daubechies-4 (D4)
$H = \{1 / \sqrt{2}, 1 / \sqrt{2}\}$ $G = \{1 / \sqrt{2}, -1 / \sqrt{2}\}$	$H = \{(1 + \sqrt{3}) / (4 * \sqrt{2}), (3 + \sqrt{3}) / (4 * \sqrt{2}), (3 - \sqrt{3}) / (4 * \sqrt{2}), (1 - \sqrt{3}) / (4 * \sqrt{2})\}$ $G[0] = H[3], G[1] = -H[2], G[2] = H[1], G[3] = -H[0]$

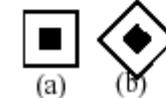
1. Edge features



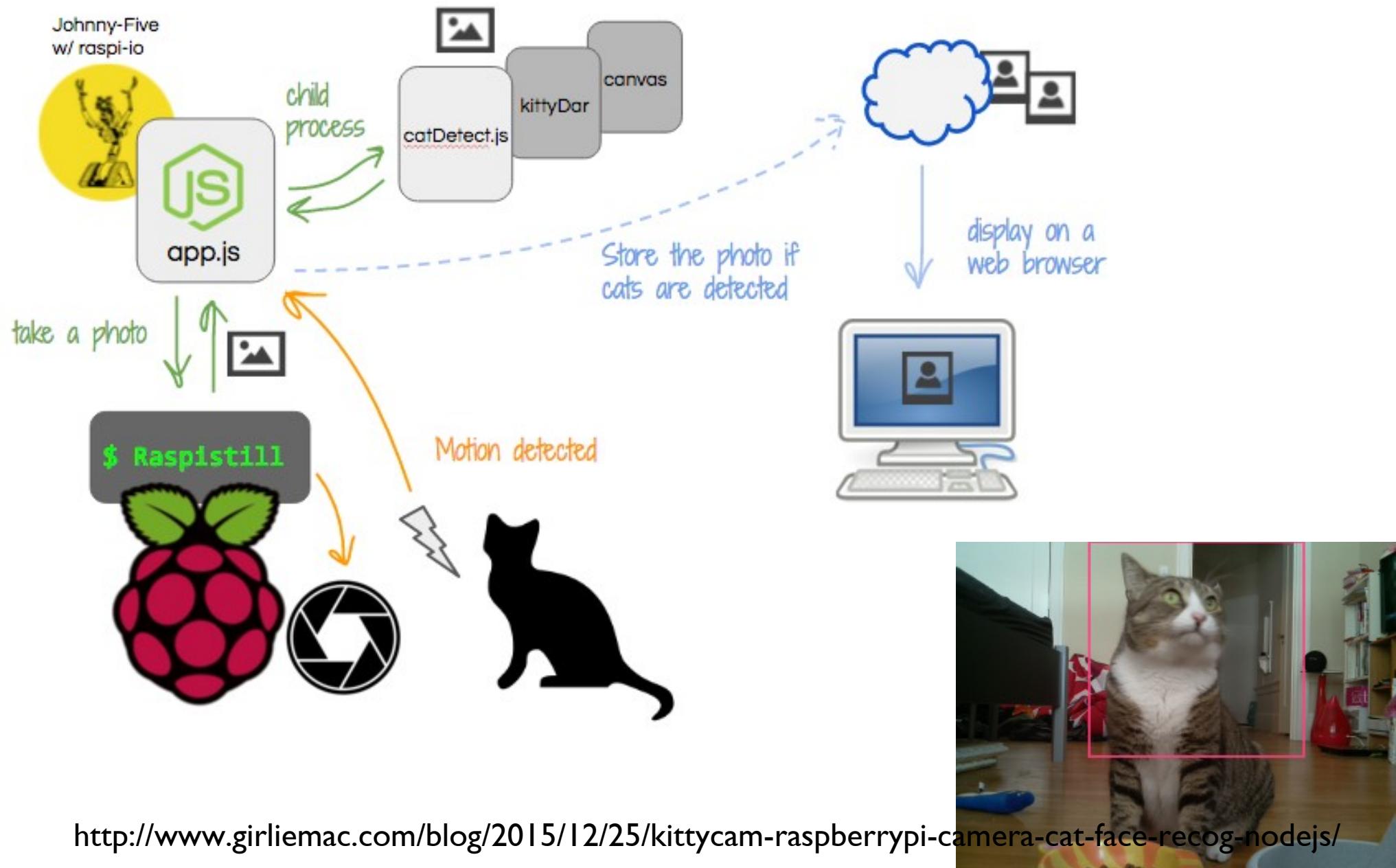
2. Line features



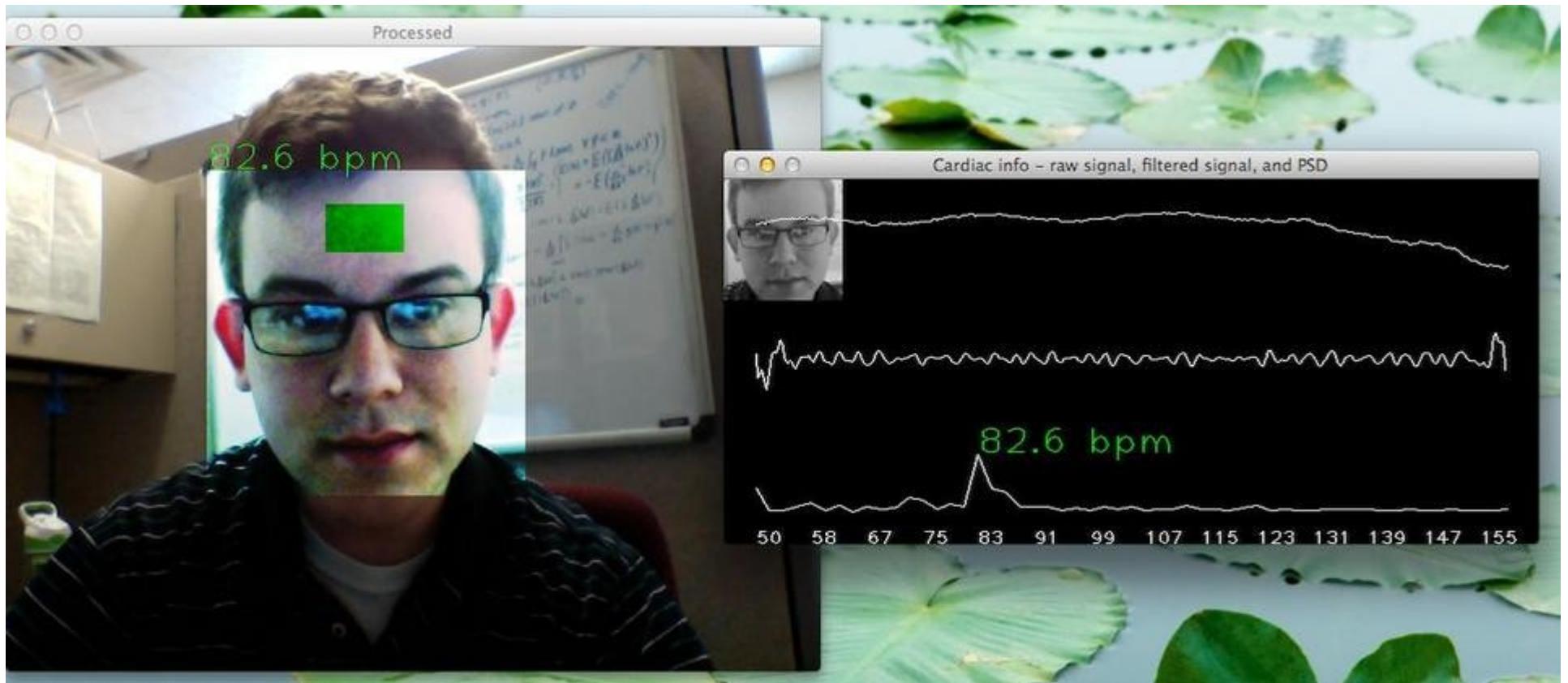
3. Center-surround features



貓臉偵測

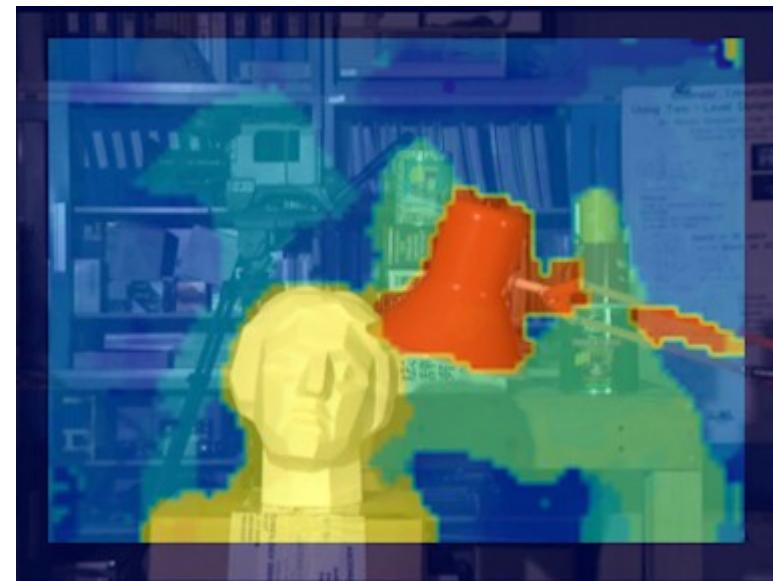
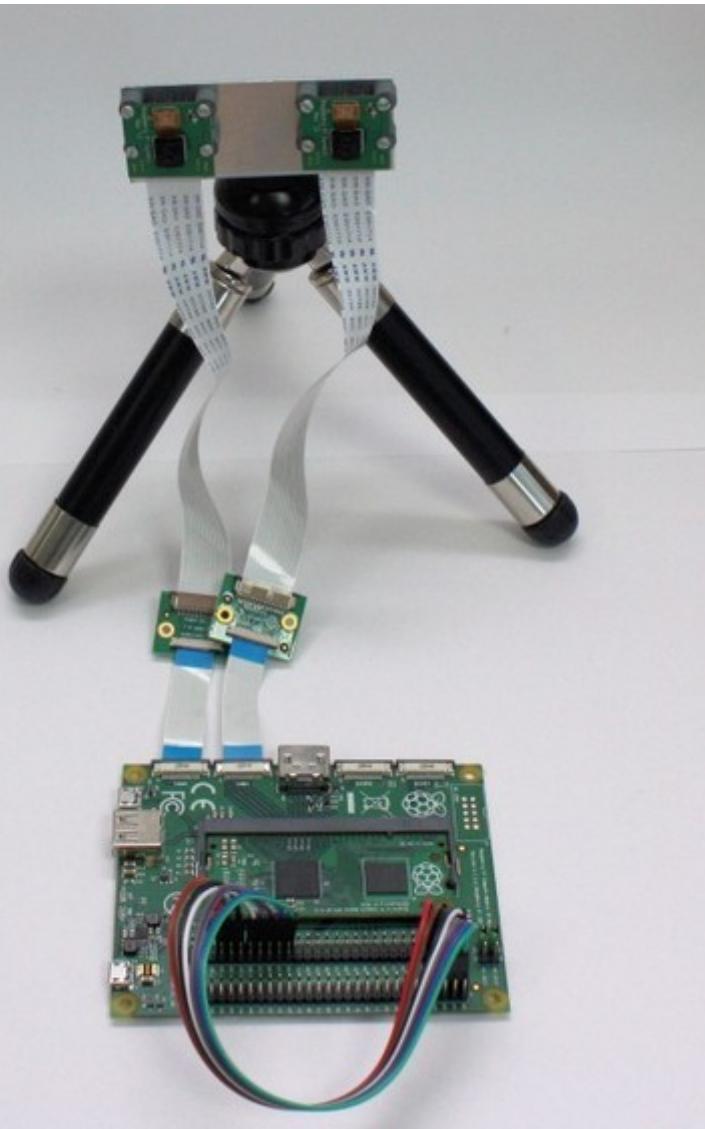


脈博辨識



<https://github.com/theearn/webcam-pulse-detector>

3D 建模計算深度 /2 Cameras



<http://www.raspberrypi.org/real-time-depth-perception-with-the-compute-module/>

3D 掃描 /50 Cameras



<http://www.pi3dscan.com/>

效果 + Autodesk Recap



WWW.Pi3DScan.Com

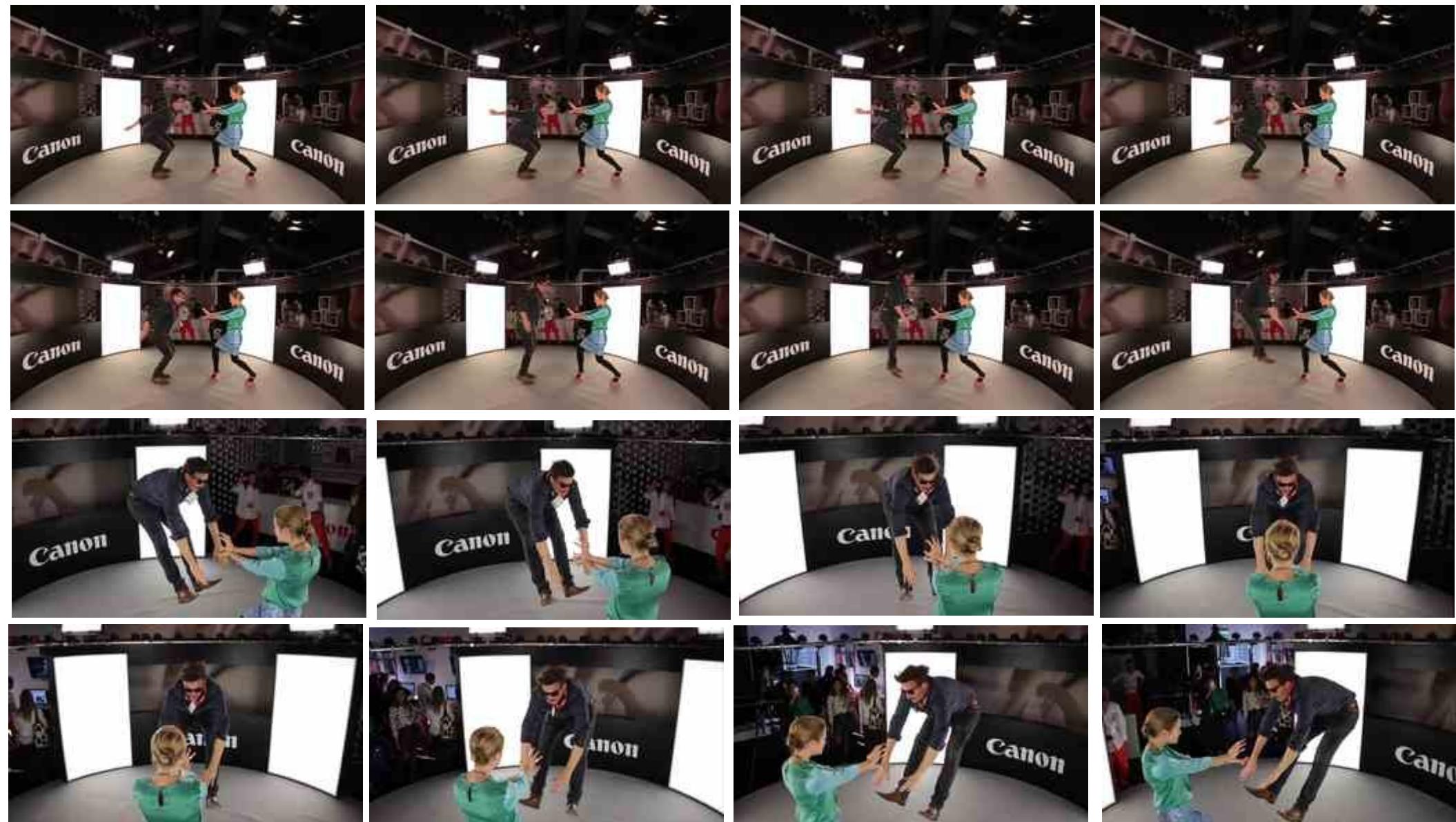
<http://www.pi3dscan.com/>

360 度照片

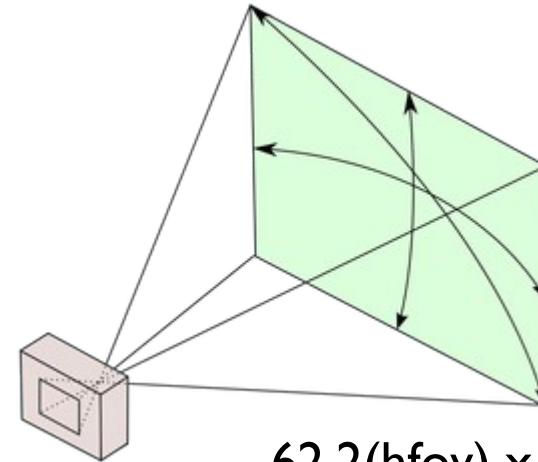


<https://vimeo.com/77218985>

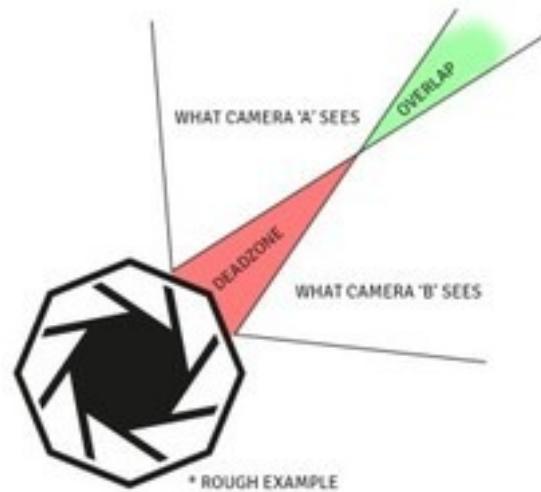
效果



ZERO360



62.2(hfov) x 48.8(vfov)



<http://raspberryjamberlin.de/introducing-zero360/>

八台相機拍出另一種全景照片



<http://raspberryjamberlin.de/introducing-zero360/>

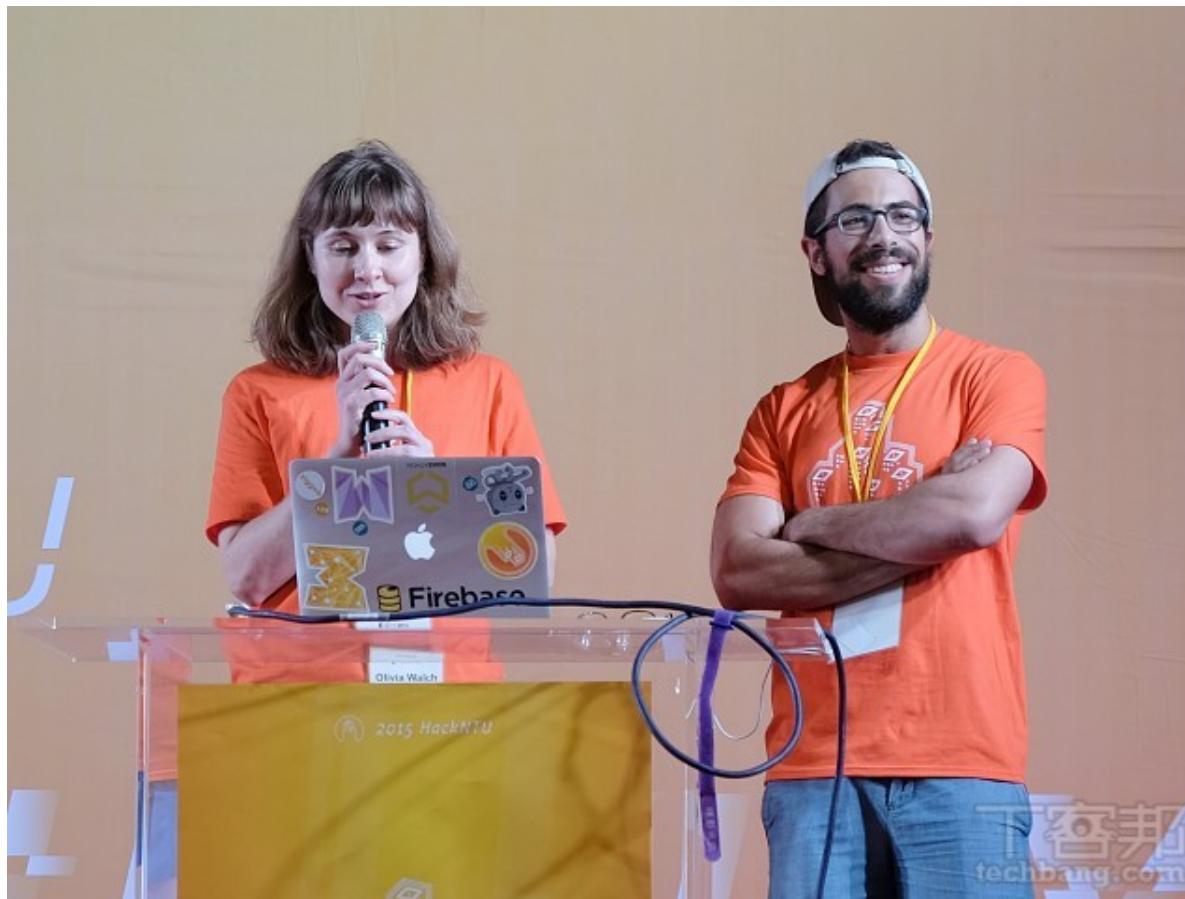
Camera 技術與比賽範例

備註：範例比賽所使用的技術不一定是和舉例相同

第三屆 2015 台大黑客松

- 冠軍: Twig

- 從語言學習出發，透過 App 拍照來辨認物品，找出相對應的單字後可翻譯成多國語言，讓使用者輕鬆快樂學習外文



<http://www.techbang.com/posts/25444>

2016 通訊大賽「智慧城市應用服務設計競賽」

- 作品名稱：醫巴

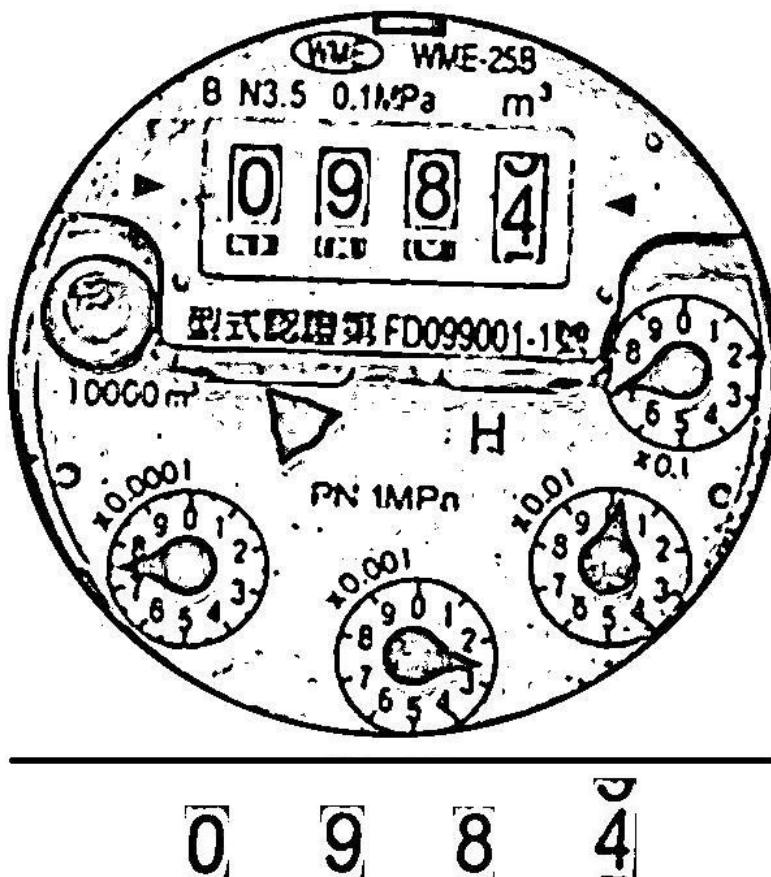
- 持續監控司機的生理狀態，提升大眾交通工具安全



2016 通訊大賽「智慧城市應用服務設計競賽」

- 作品名稱：外掛式水表偵測器 - 物聯網應用

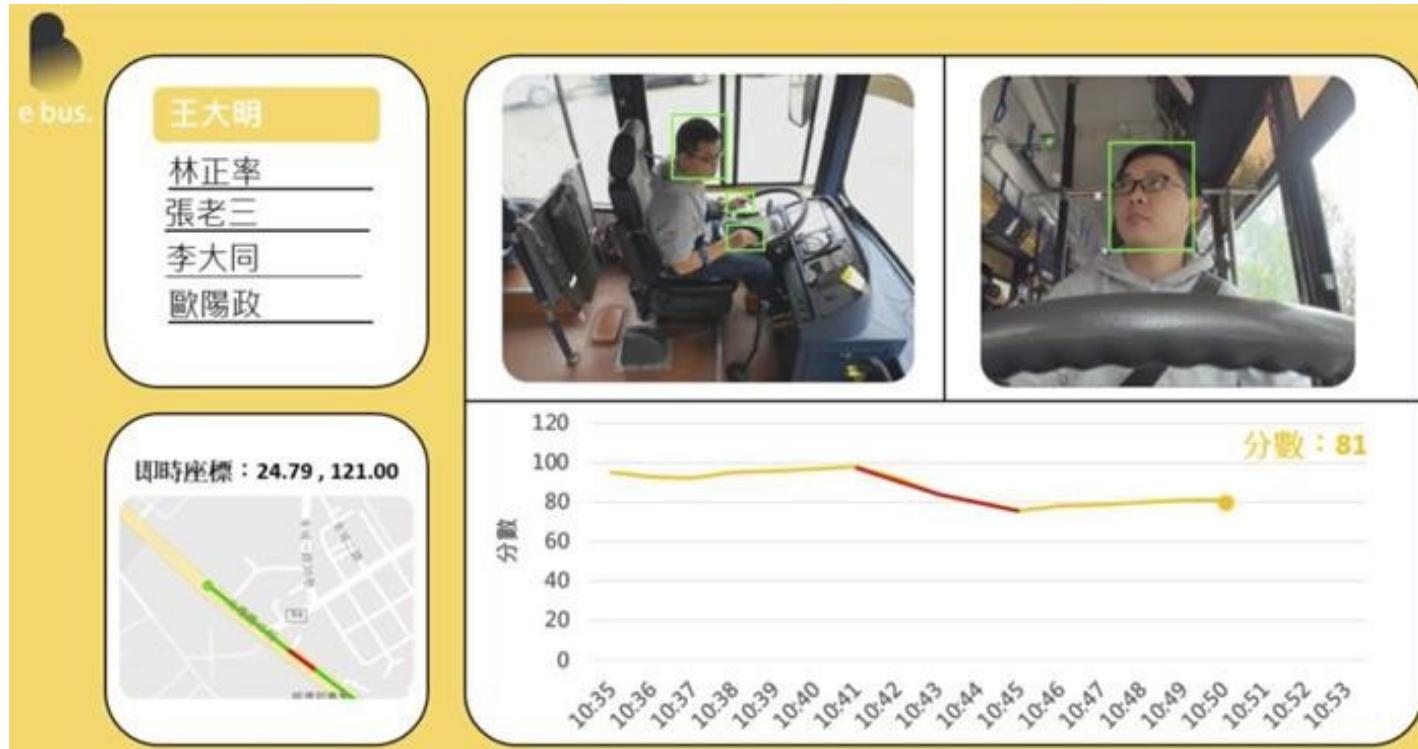
- 樹莓派 + 攝影機，讓傳統水表變身智慧水表



2017 通訊大賽「智慧城市應用服務設計競賽」

- 作品名稱：醫巴 eBus

- 用深度學習技術聰明辨識疲勞駕駛與分心駕駛



技術核心

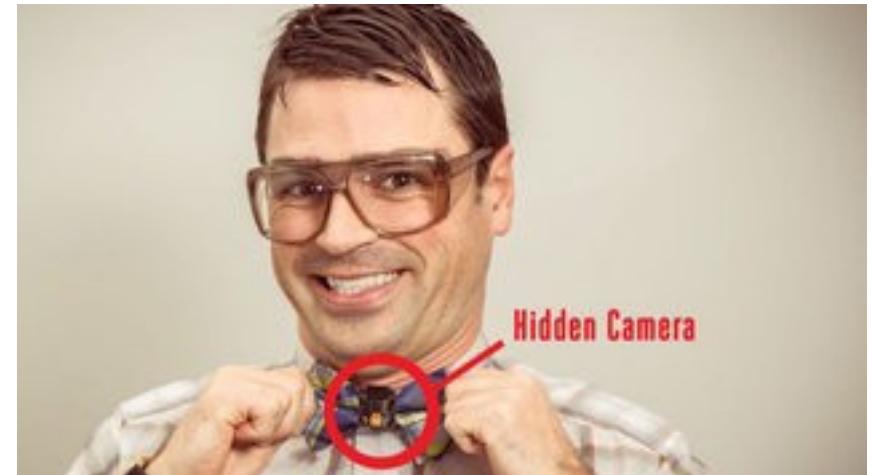
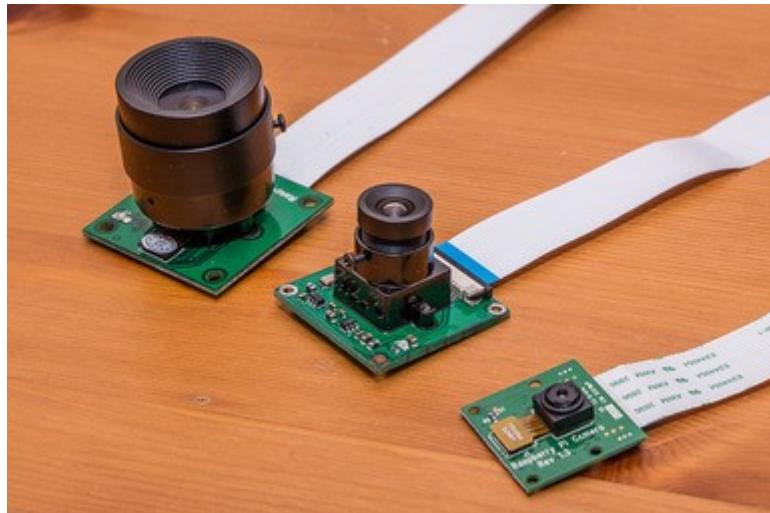


Camera 改裝套件

固定的機構



相機模組改裝



<https://www.adafruit.com/products/1937>

<https://www.flickr.com/people/100320847@N06/>

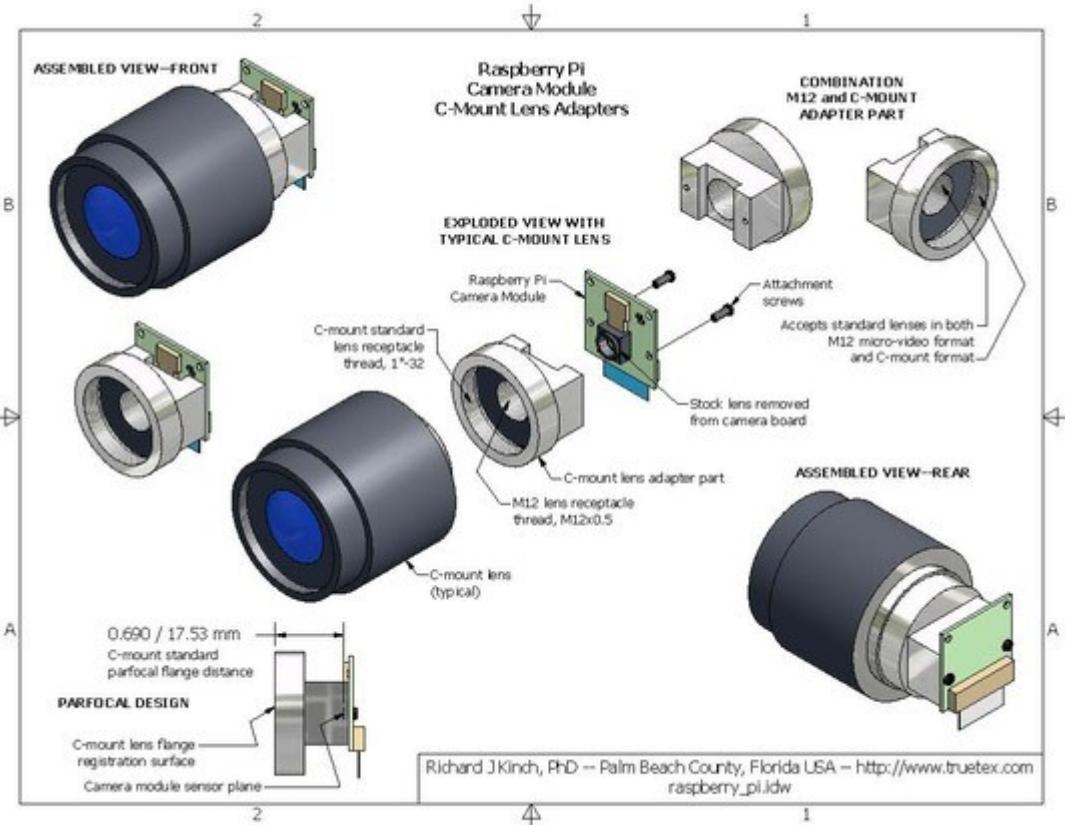
360 相機



鏡頭改裝



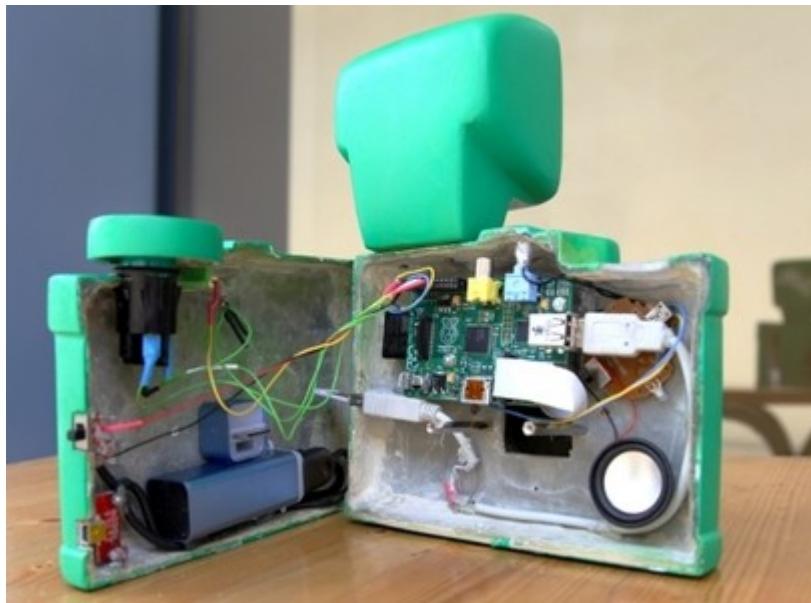
惡改鏡頭





<http://www.truetex.com/raspberrypi>

外殼改裝



<http://www.modmypi.com/>

<http://blog.pi3g.com/2013/11/coming-soon-raspberry-with-case-mounted-camera/>

Camera 安裝

在關機的狀態下安裝 Camera

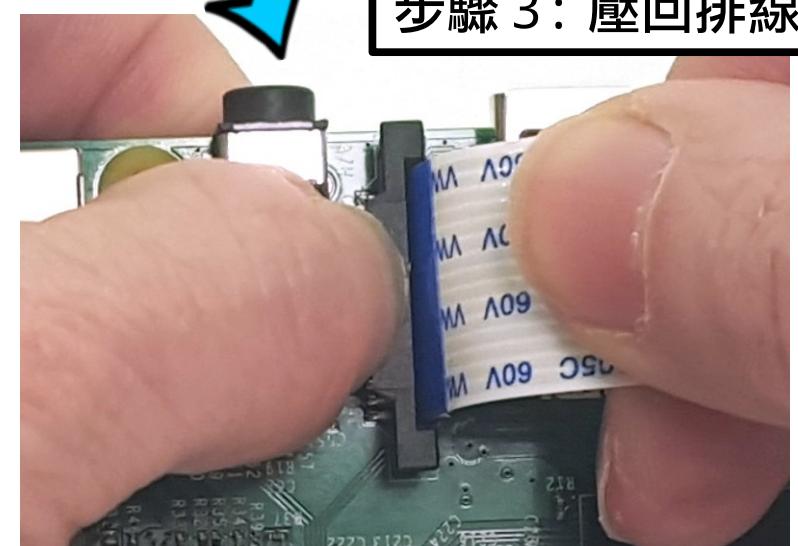
關機指令：\$ sudo poweroff

步驟 2：插入排線

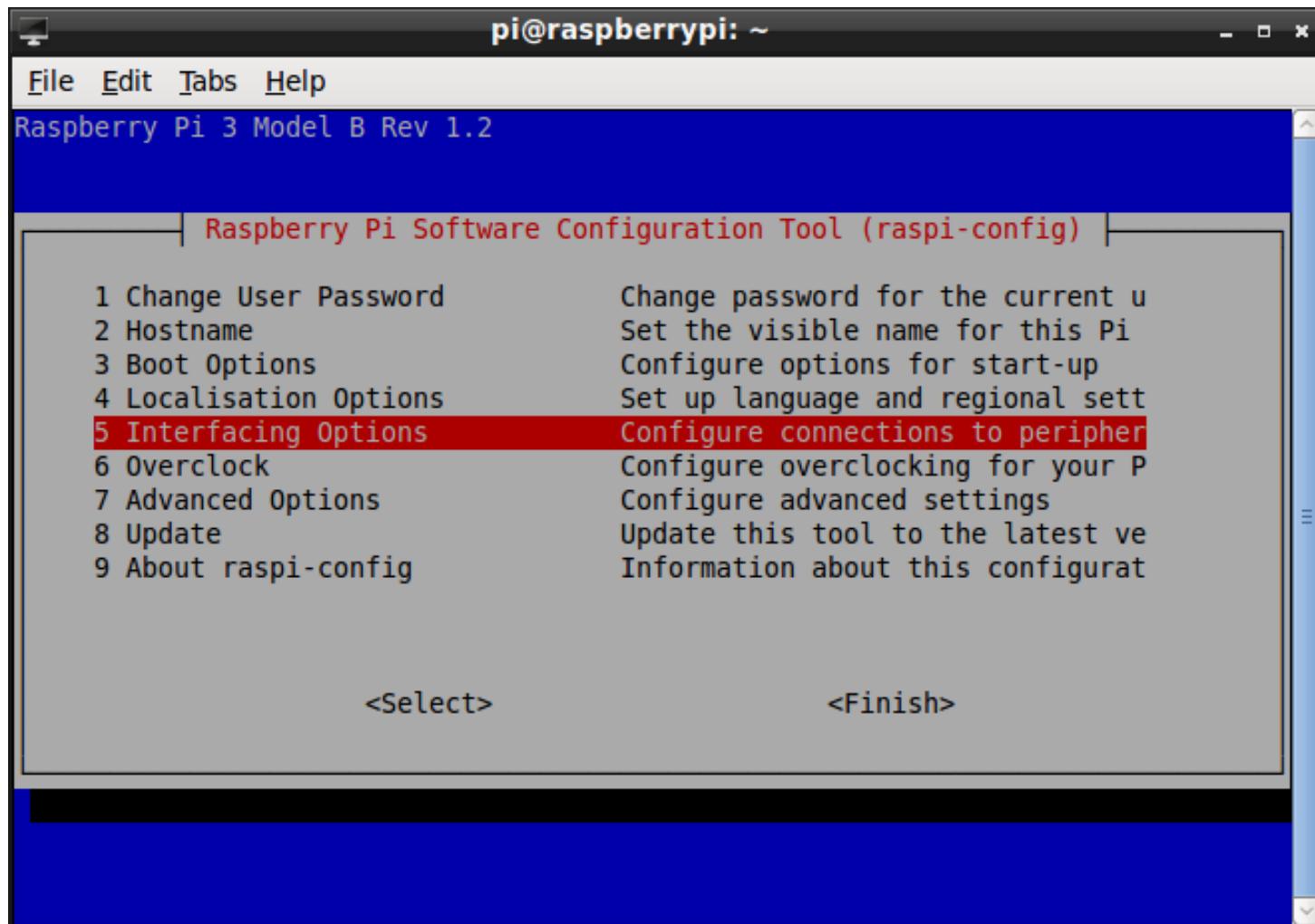
步驟 1：拉起卡榫



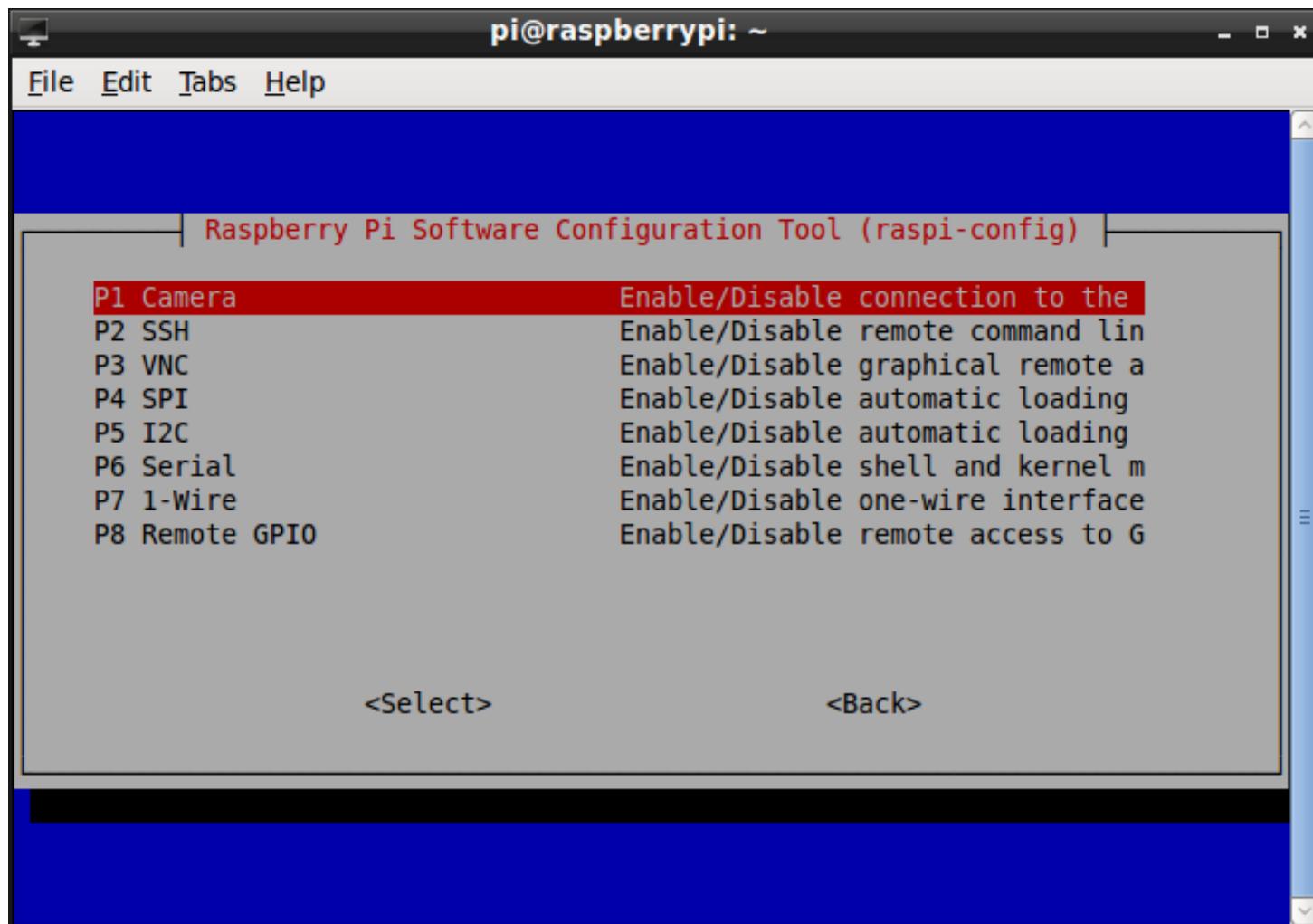
步驟 3：壓回排線



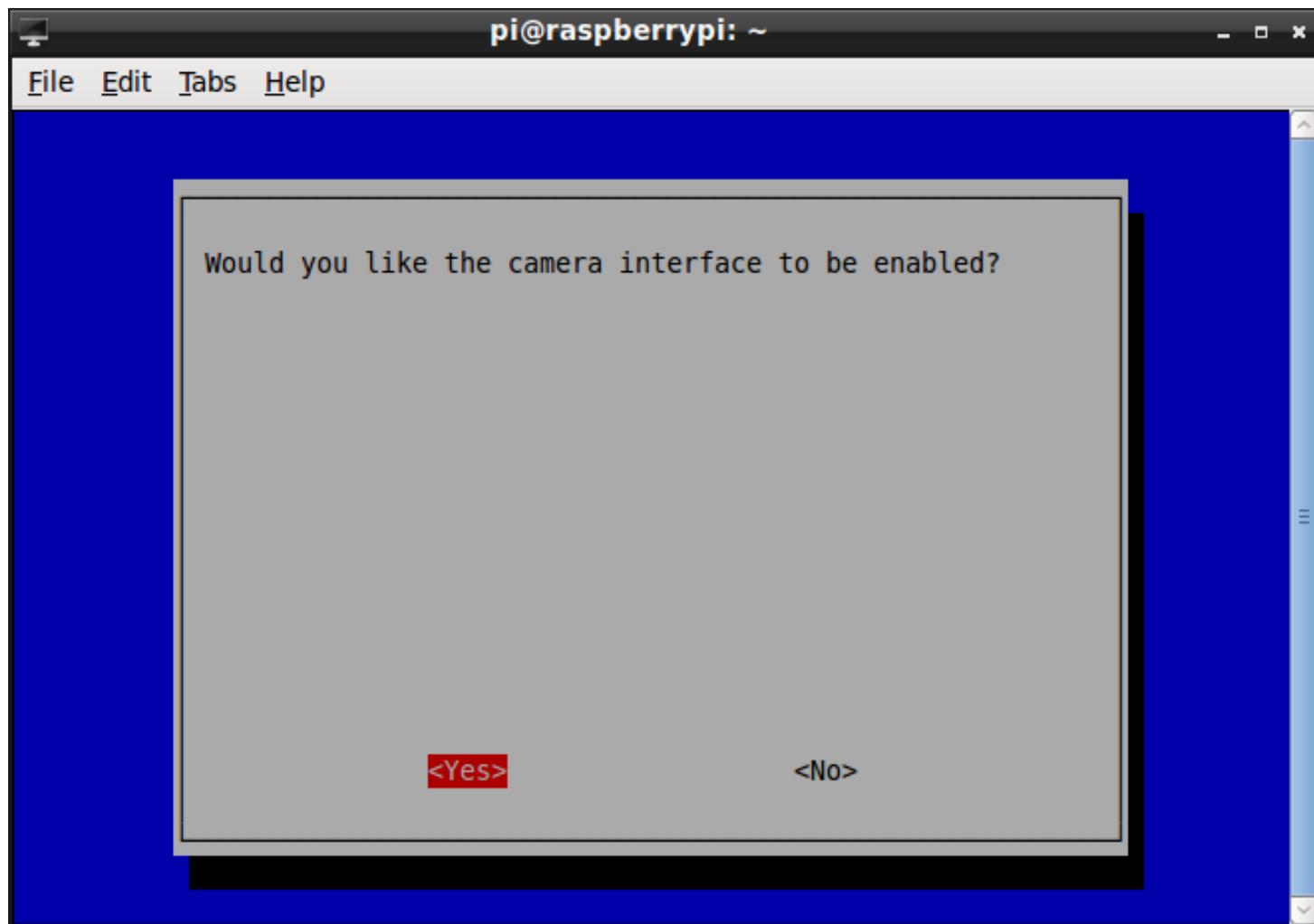
\$ sudo raspi-config



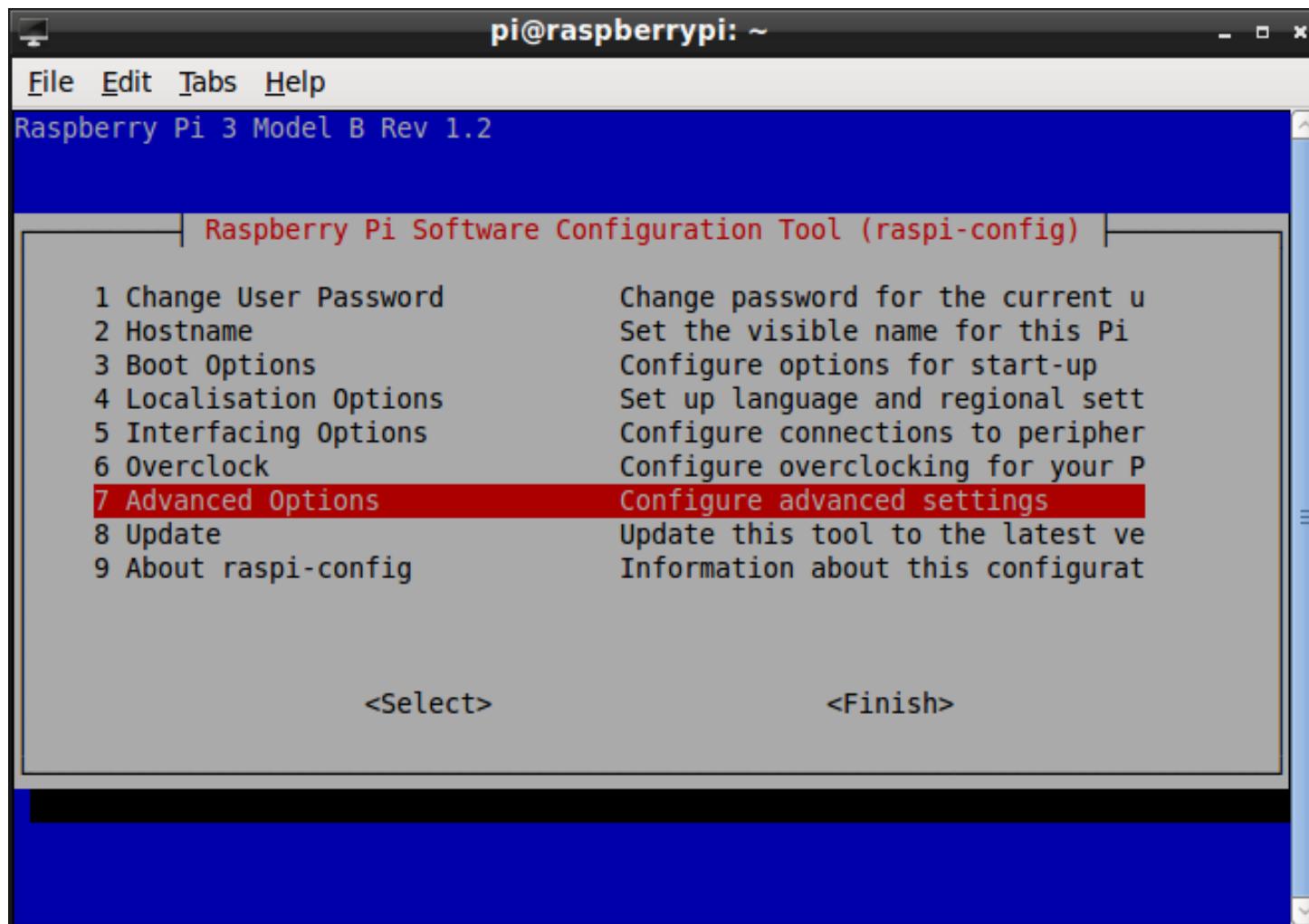
選擇 Camera



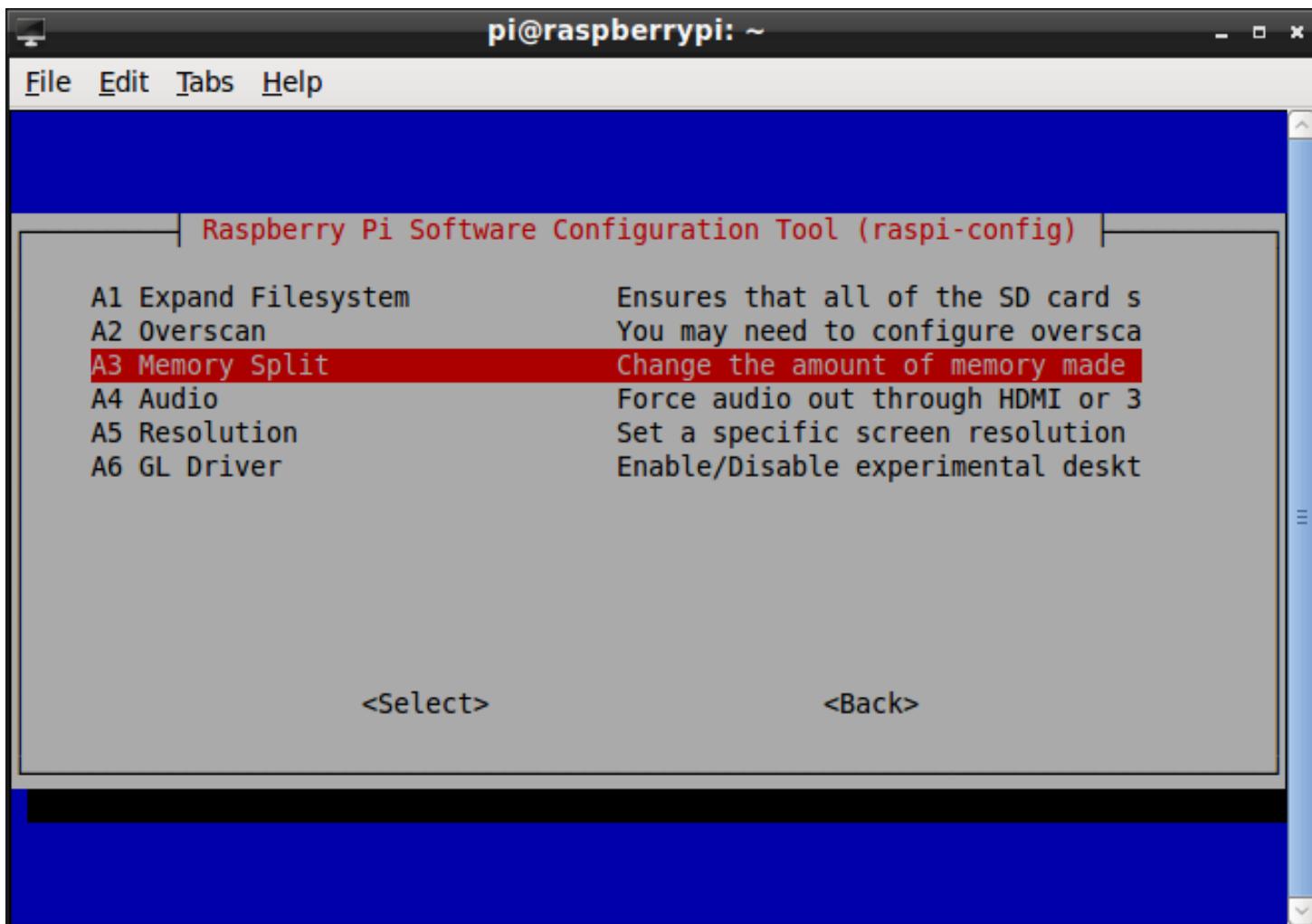
啟用 Camera



進階選項



設定記憶體分配



要大於等於 128



實戰 Camera 使用

使用 Camera 前先消除靜電吧



實驗 1 : Hello Camera

目的：練習照相和攝影的指令

拍照指令 RaspiStill

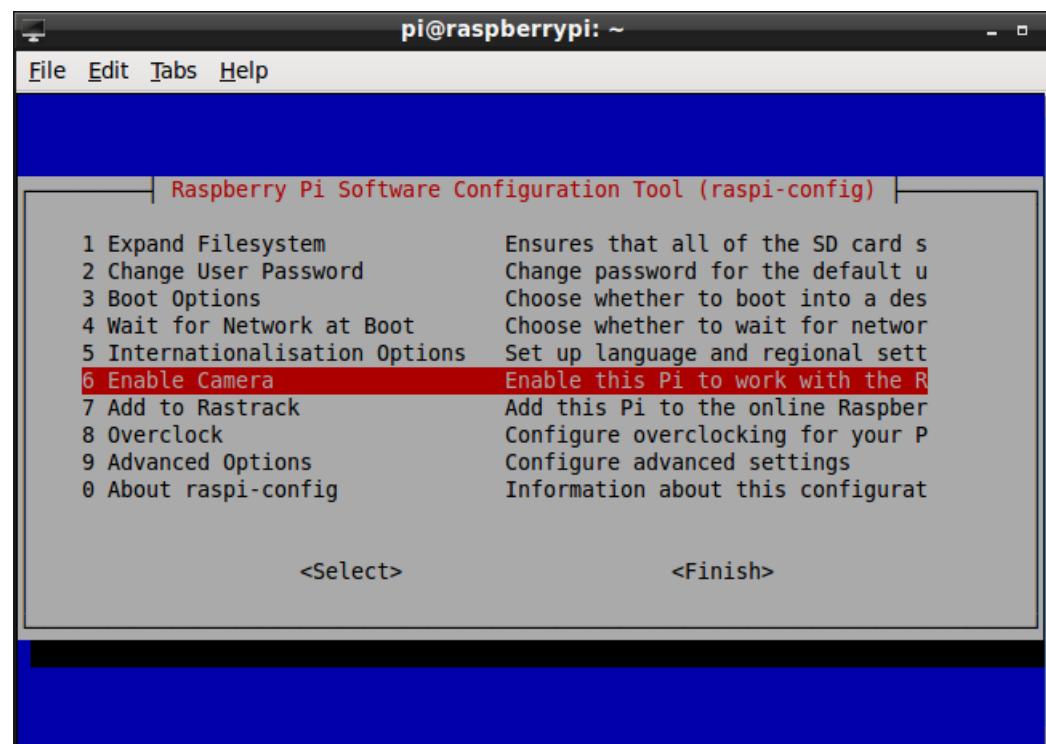
- 只預覽 2 秒 (-t)，不存檔
 - \$ raspistill -t 2000
- 5 秒後拍照（預設），檔案 test.jpg(-o)
 - \$ raspistill -o test.jpg
- 3 秒後拍照，並編碼成 png 格式 (-e)，長 640x 寬 480
 - \$ raspistill -t 3000 -o test.png -e png -w 640 -h 480

常見 Camera 問題？

- 訊息 : Camera is not enabled in this build

```
mmal: mmal_vc_component_create: failed to create component 'vc.ril.camera' (1:ENOMEM)
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 ensure that "camera" has been enabled
```

- 解法：進 raspi-config 重新 enable camera
 - \$ sudo raspi-config



- 訊息 : Camera is not detected

```
mmal: mmal_vc_component_create: failed to create component 'vc.ril.camera' (1:ENOMEM)
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 installed correctly
```

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



是否有鬆脫 ?

錄影指令 RasPiVid

- 錄 5 秒 1080p30 影片 (預設 w/h = 1920/1080)
 - \$ raspivid -t 5000 -o video.h264
- 錄 5 秒的 1080p30 影片，長 640x 寬 480
 - \$ raspivid -t 5000 -w 640 -h 480 -o video.h264

更多參數或用法請看文件
<http://goo.gl/V4k1cZ>

如何看照片和影片？

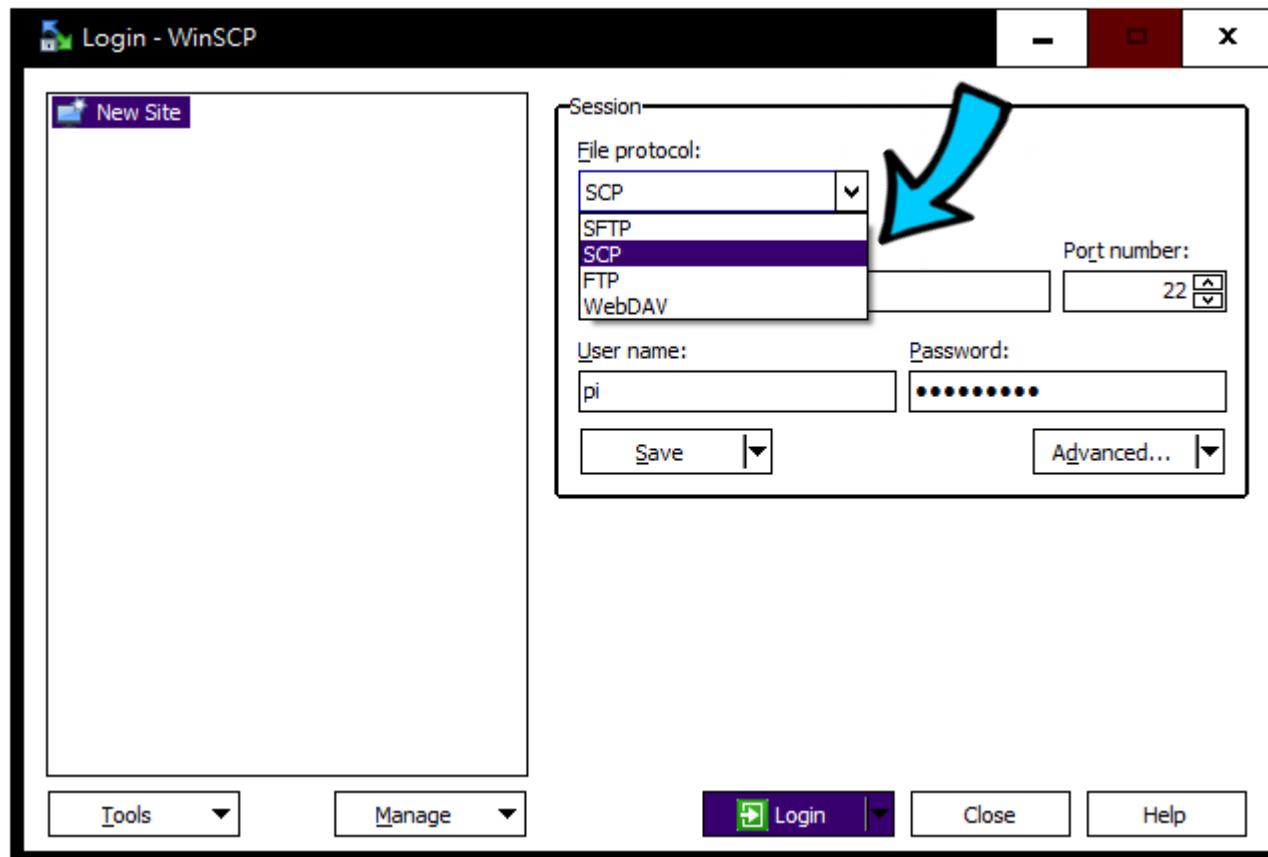
方法一：將 Pi 的檔案傳回本機端

如何將電腦和 Pi 的檔案互傳？

- 使用 SCP
 - Pi 當作 SCP Server，啟動 SSH Server 即可
 - Windows 當作 SCP Client，需安裝 WinSCP

在 Windows 上安裝 WinSCP

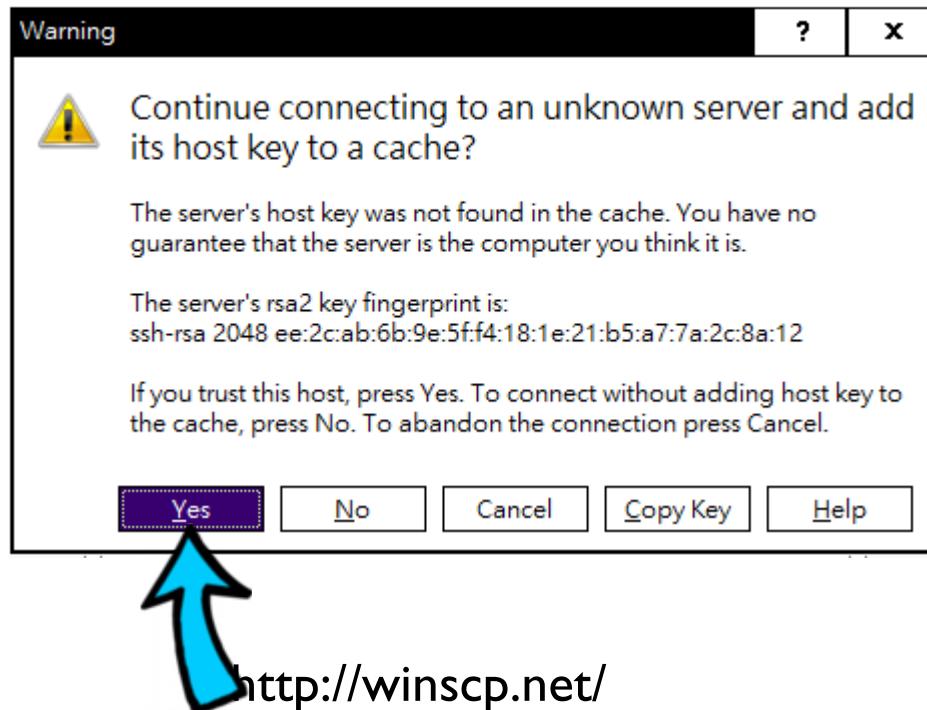
- <http://winscp.net/eng/download.php>
- 連線設定選擇 SCP



<http://winscp.net/>

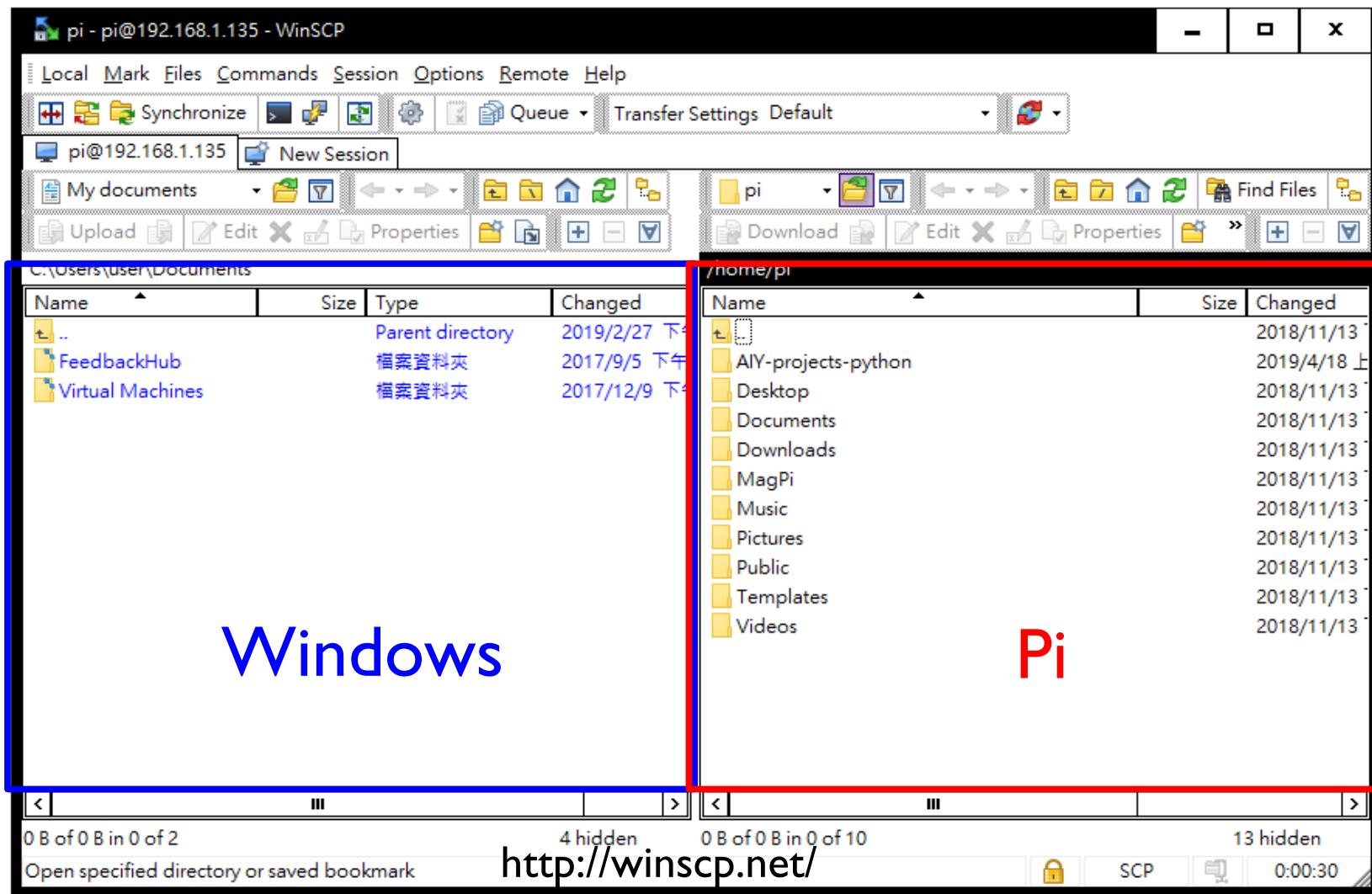
在 Windows 上安裝 WinSCP

- <http://winscp.net/eng/download.php>
- 連線設定選擇 SCP
- 接受交換金鑰



在 Windows 上安裝 WinSCP

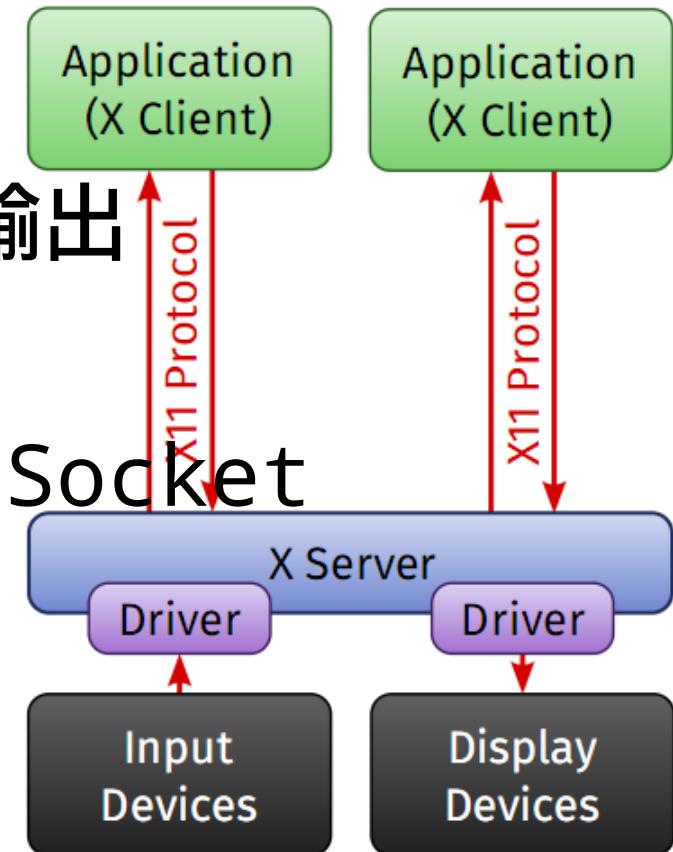
- 左右兩邊都可以做檔案傳輸



方法二：使用 X11 Forwarding

X Window System

- 是一種圖形應用標準
- Client/Server 架構
 - X Client：應用程式
 - X Server：管理硬體輸入 / 輸出
- 可透過網路傳輸
 - TCP/IP 或是 Unix Domain Socket
- X11 是通訊協定名稱



在Windows 安裝 X Server

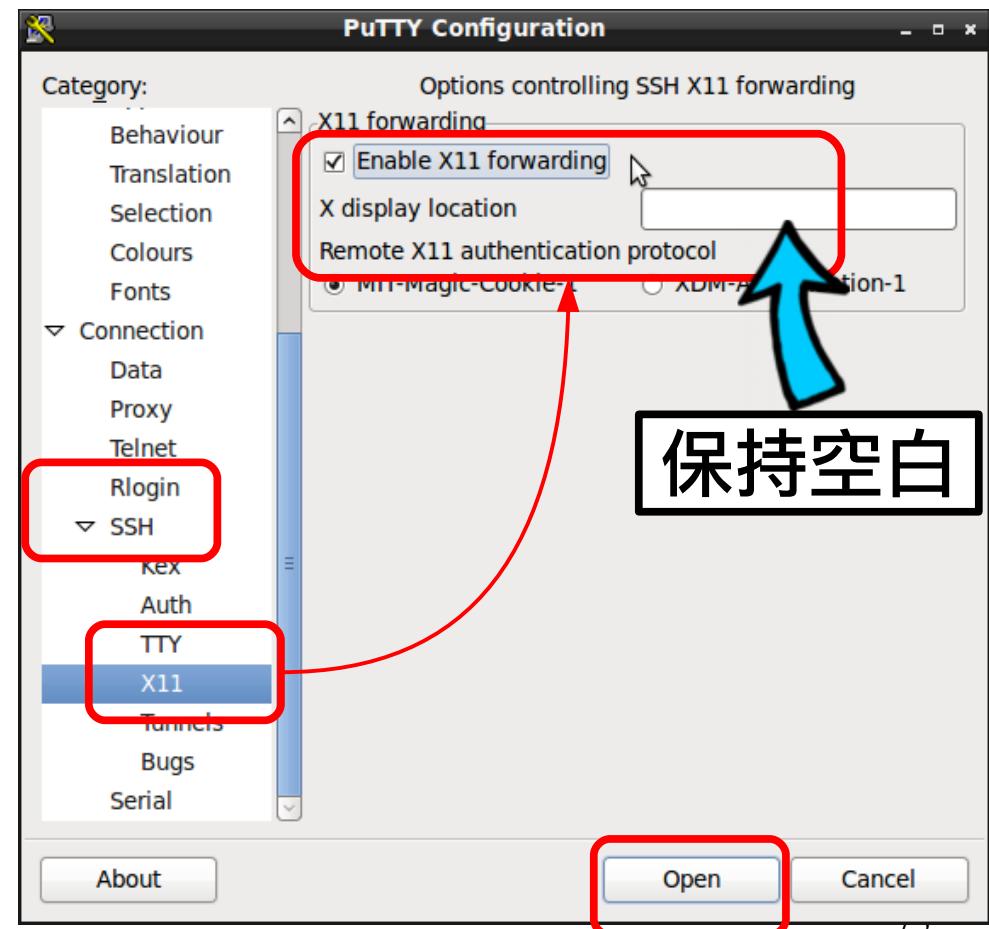
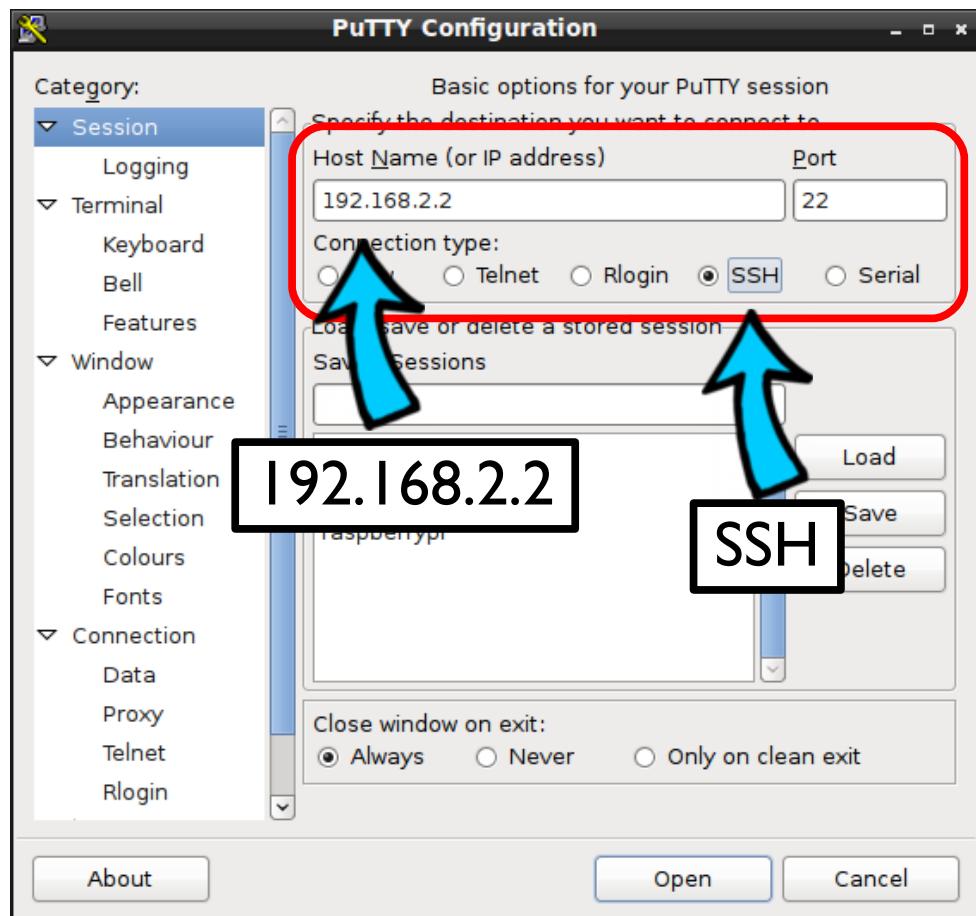
- 安裝 Xming , 下一步到底

<http://sourceforge.net/projects/xming>



在 Windows 設定 X11 Forwarding

- SSH > X11 > Enable X11 forwarding



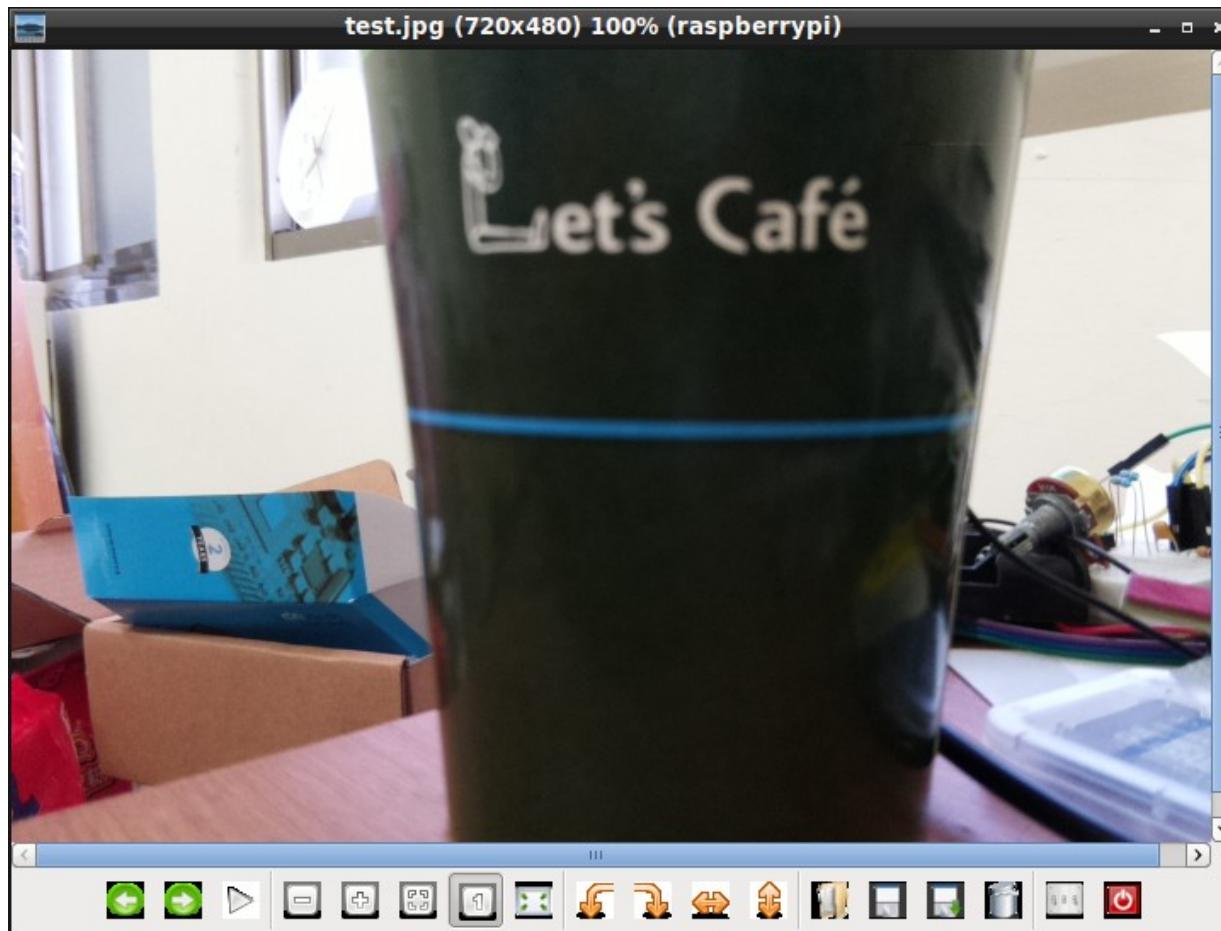
如果是 Linux 或是 Mac OS
開啟終端機 , ssh -X pi@PI 的 IP

“Can not open display” on Mac

- 第一步：在 Mac 編輯 /etc/sshd_config
(或是 /etc/ssh/sshd_config)
修改這行 # X11Forwarding no
把 no 改成 yes 並且把註解拿掉
- 第二步：下載安裝 XQuartz 並重開機
<http://xquartz.macosforge.org/landing/>
- 感謝 Dami 和 YUN-TAO CHEN 的貢獻

XII Forwarding 連線成功後

- 看照片
 - \$ gpicview test.jpg

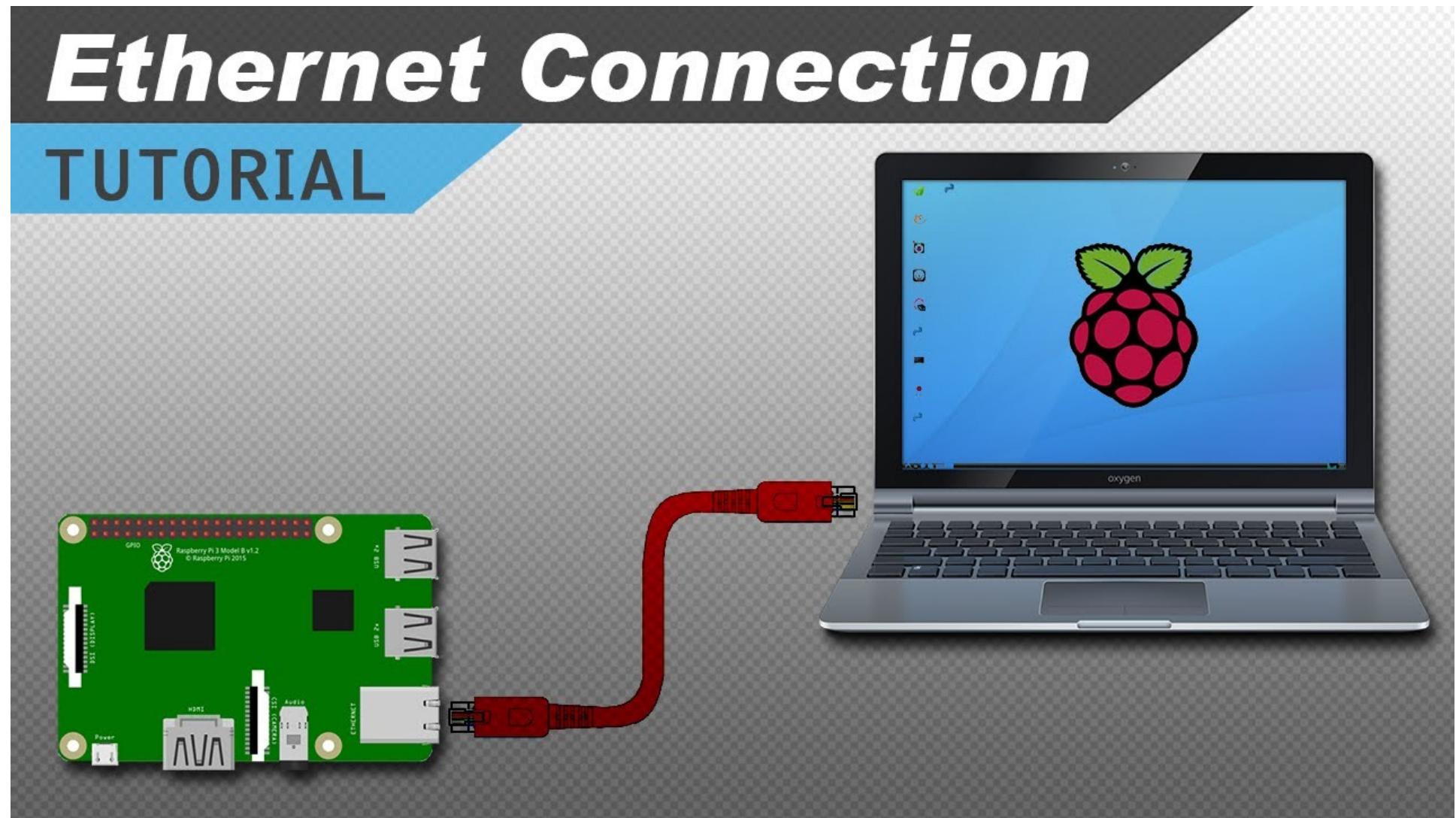


如果無線網路很慢的話 . . .

直接用網路線對接

Ethernet Connection

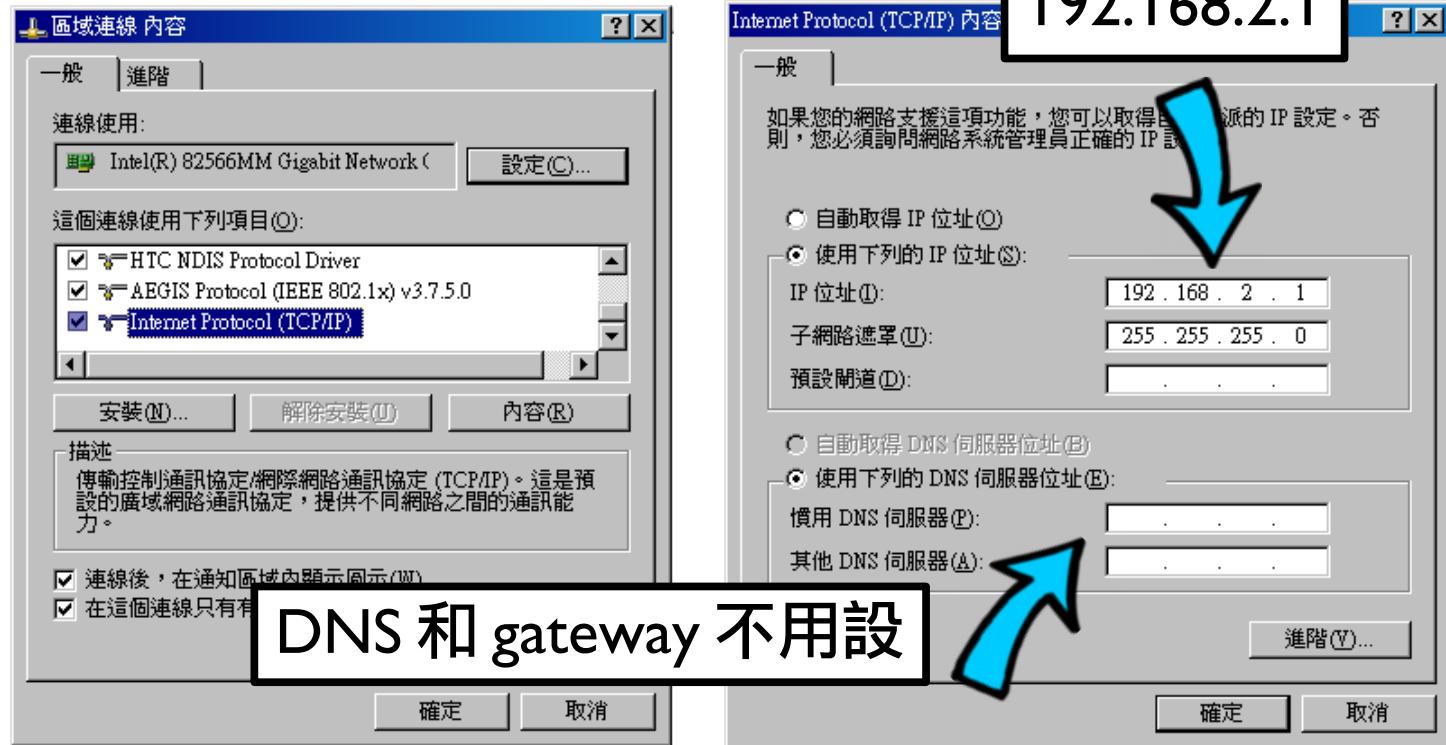
TUTORIAL



<https://www.youtube.com/watch?v=5DCPDQnRXm8>

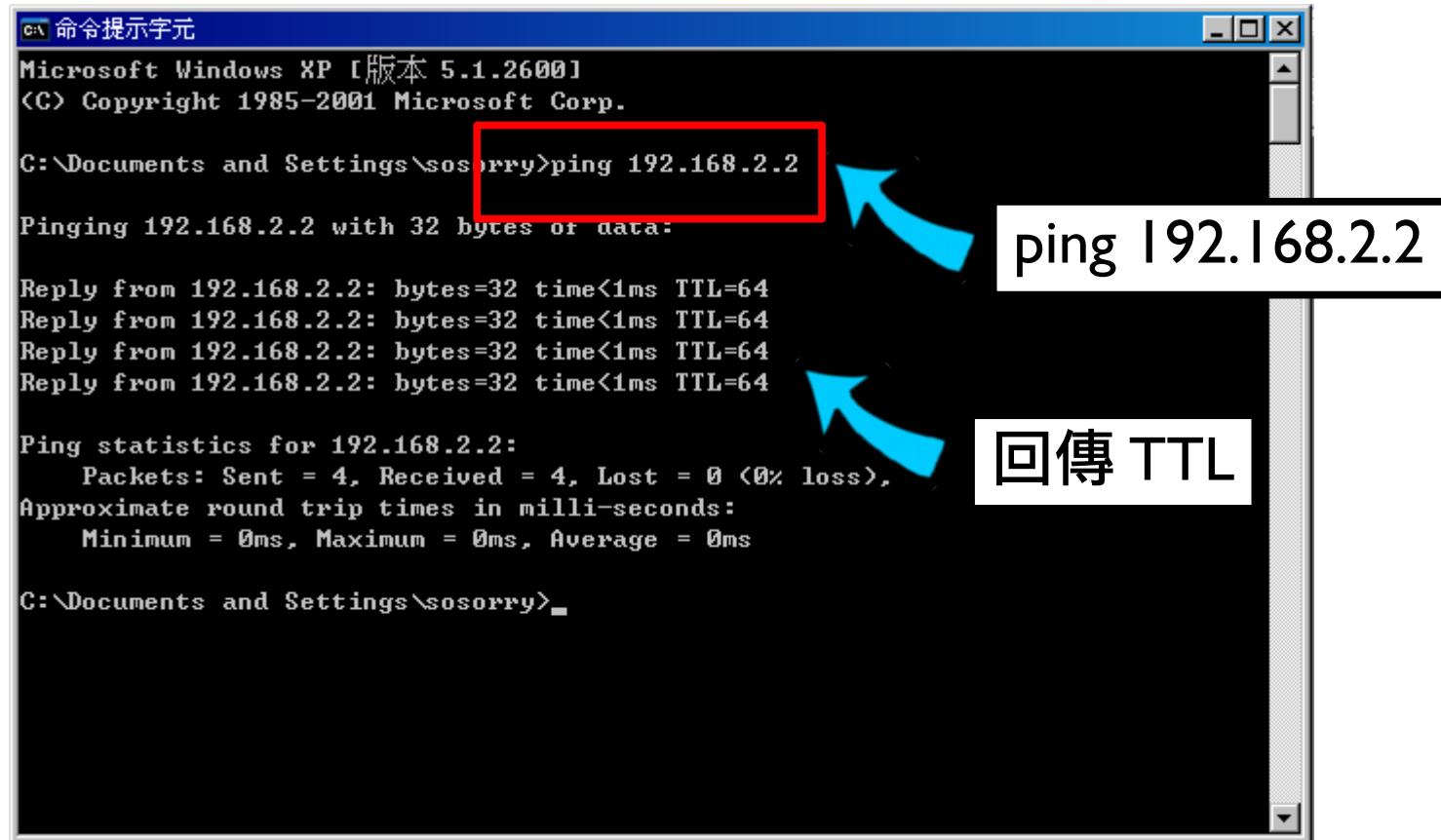
網路設定

- 把 Pi 設成 .2.2
 - \$ sudo ifconfig eth0 192.168.2.2 netmask 255.255.255.0
- 把 Windows 設成 .2.1



確認網路是否有通？

- 在 Windows 執行 cmd
- 輸入 ping 192.168.2.2 看是否有回應？



XII Forwarding FAQ

- Q: 什麼時候可以使用 X11 Forwarding ?
- A: 當 GUI toolkit 或是 library 是架在 X 的時候
 - 例如 Qt ,Gtk ,Tkinter ,SDL 等等適合
 - Framebuffer ,GPU 等直接輸出不能使用
- Q: 什麼時候要使用 sudo 什麼時候不用 ?
- A: X11 Forwarding 需要認證 (authorization) , 所以如果要畫面回傳時都不能使用 sudo 執行

練習

- 請查詢網頁文件，將拍照後旋轉 90 度

<http://goo.gl/V4k1cZ>

- 一般拍照：

- ```
$ raspistill -t 2000 -o normal.jpg
```

- 旋轉拍照：

- ```
$ raspistill -t 2000 ? -o rotation.jpg
```

- 負片效果：

- ```
$ raspistill -t 2000 -ifx ? -o neg.jpg
```



請找出適合的參數



請找出適合的參數

# 小結

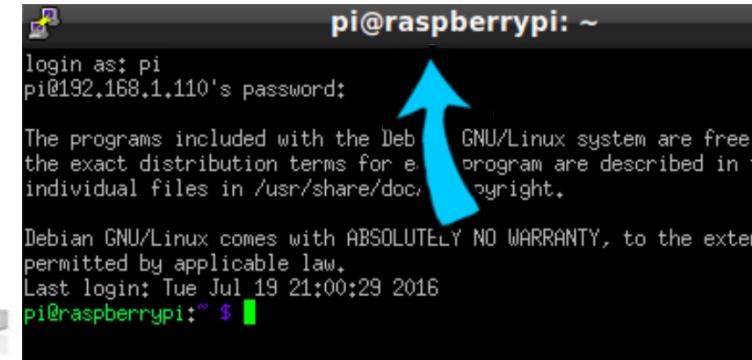
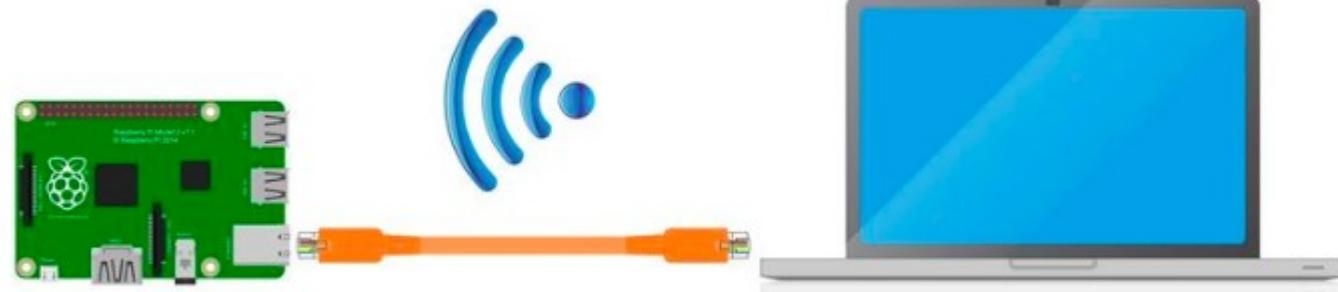
- Raspberry Pi Camera 指令
  - 拍照 raspistill
  - 錄影 raspivid
- 看照片或影片
  1. 使用 scp
  2. 使用 X11 Forwarding

# Serial 連線和 SSH 連線有什麼不同？

- Serial 以實體線路相連，純文字，是獨占式的連線



- SSH 是 TCP/IP 通訊協定，透過 Ethernet 或 WiFi 連線



# 實驗 2：寫程式控制 Camera

目的：自己的 Camera 自己做

使用 picamera (Python library)

# Python2 五分鐘速成

- 變數，物件，型別，註解
- 模組
- 縮排
- 迴圈
- 條件判斷
- 函式

# 變數，物件，型別，註解

- 動態型別 (dynamic typing)

```
這是註解
i = 3 # 變數 i 指到數字物件 3

i = [1, 2, 3, 4, 5] # 變數 i 指到串列物件
print i[2] # 印出串列中第三個元素

i = "abcde" # 變數 i 指到字串物件
print i[2] # 印出字串中第三個元素
```

# 模組

```
import MODULE
import picamera

from Module import function
from time import sleep
```

# 縮排

- 用縮排取代大括號
- 程式碼的區塊是用縮排分隔
- 不使用 tab，使用空白鍵
- 常見縮排為 4 個空白鍵

# 迴圈

- 自動迭代 (iterator)

```
for i in xrange(start, stop[, step]) :
 process
```

```
for i in xrange(0, 11, 5) :
 print i
```

# 條件判斷

```
if condition_1 :
 process_1
elif condition_2 :
 process_2
else :
 process_3
process_4
```

# 函数

```
def function_name() :
 process
```

```
def function_name(param_name) :
 process
```

```
def function_name(param_name = 3) :
 process
```

# 兩種執行 Python 的方法

- 1. 存成檔案以後，用 python 執行
    - \$ nano test.py
    - \$ python test.py
  - 2. 進到互動模式，可直接看輸出結果
    - \$ python
    - Python 2.7.13 (default, Jan 19 2017, 14:48:08)  
[GCC 6.3.0 20170124] on linux2
    - Type "help", "copyright", "credits" or  
"license" for more information.
- >>>

# 照相

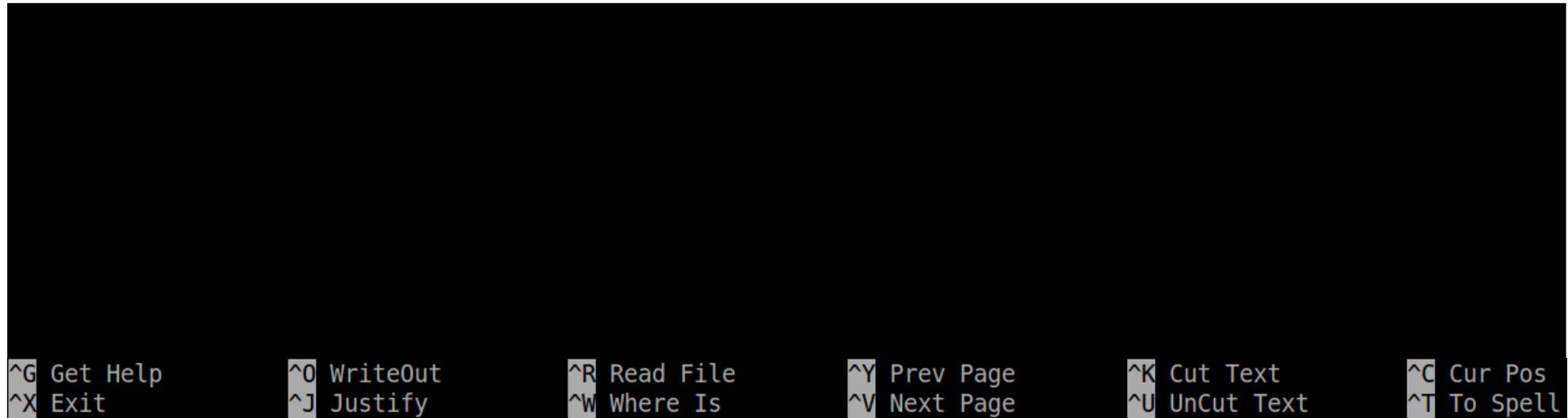
```
#!/usr/bin/python
import picamera
import time

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

- 預設相片解析度為 720x480

- 建立新檔案

- \$ nano <檔名，例如 take\_photo.py>



- 離開: Ctrl + X

> 令存新檔: y

> 不存離開: n

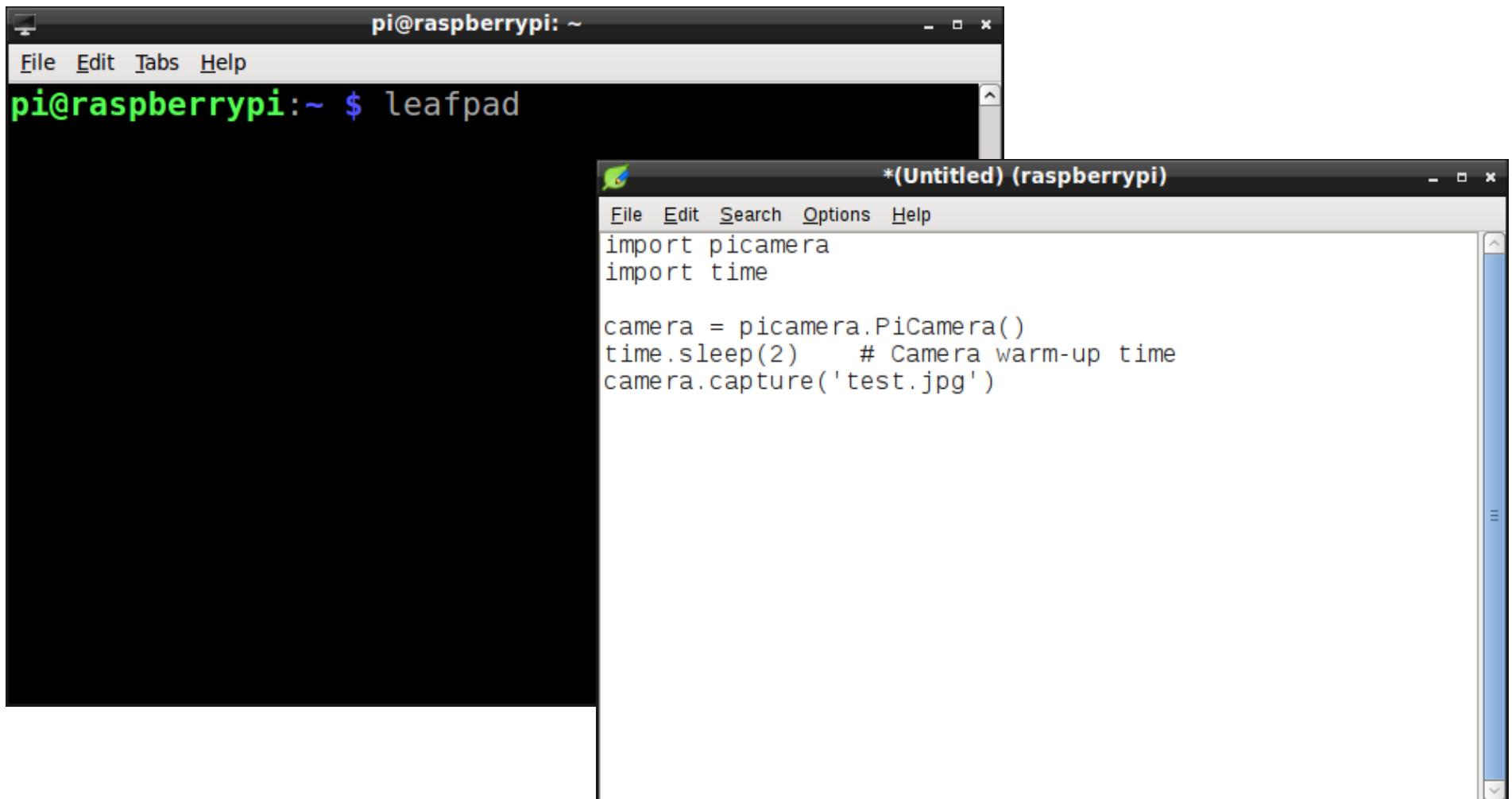
> 離開: Ctrl + C

- 修改已經存在的檔案

- \$ nano <檔名，例如 ~/.bashrc>

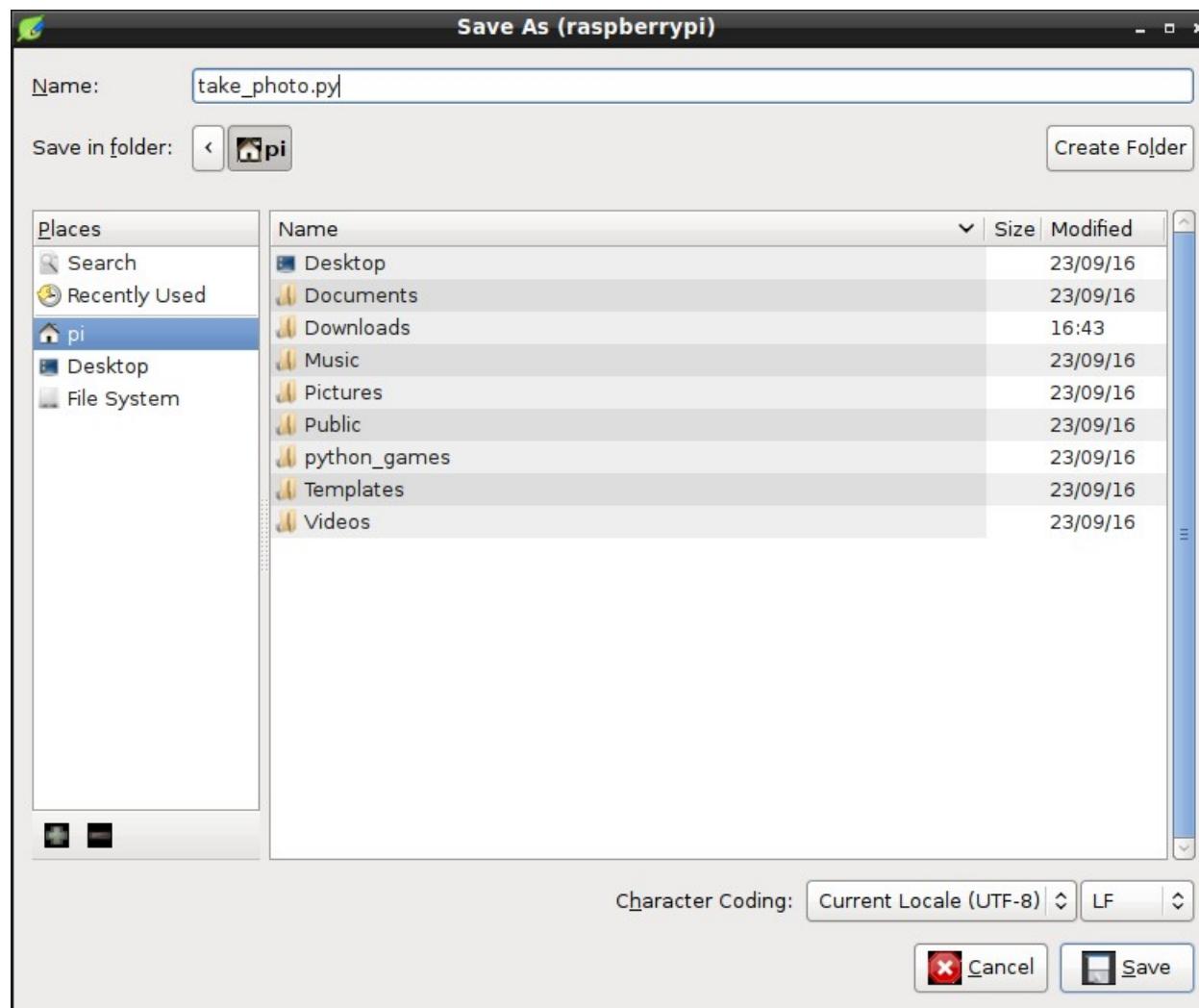
# leafpad 編輯器使用

- 在 X11 forwarding 連線成功下執行 leafpad



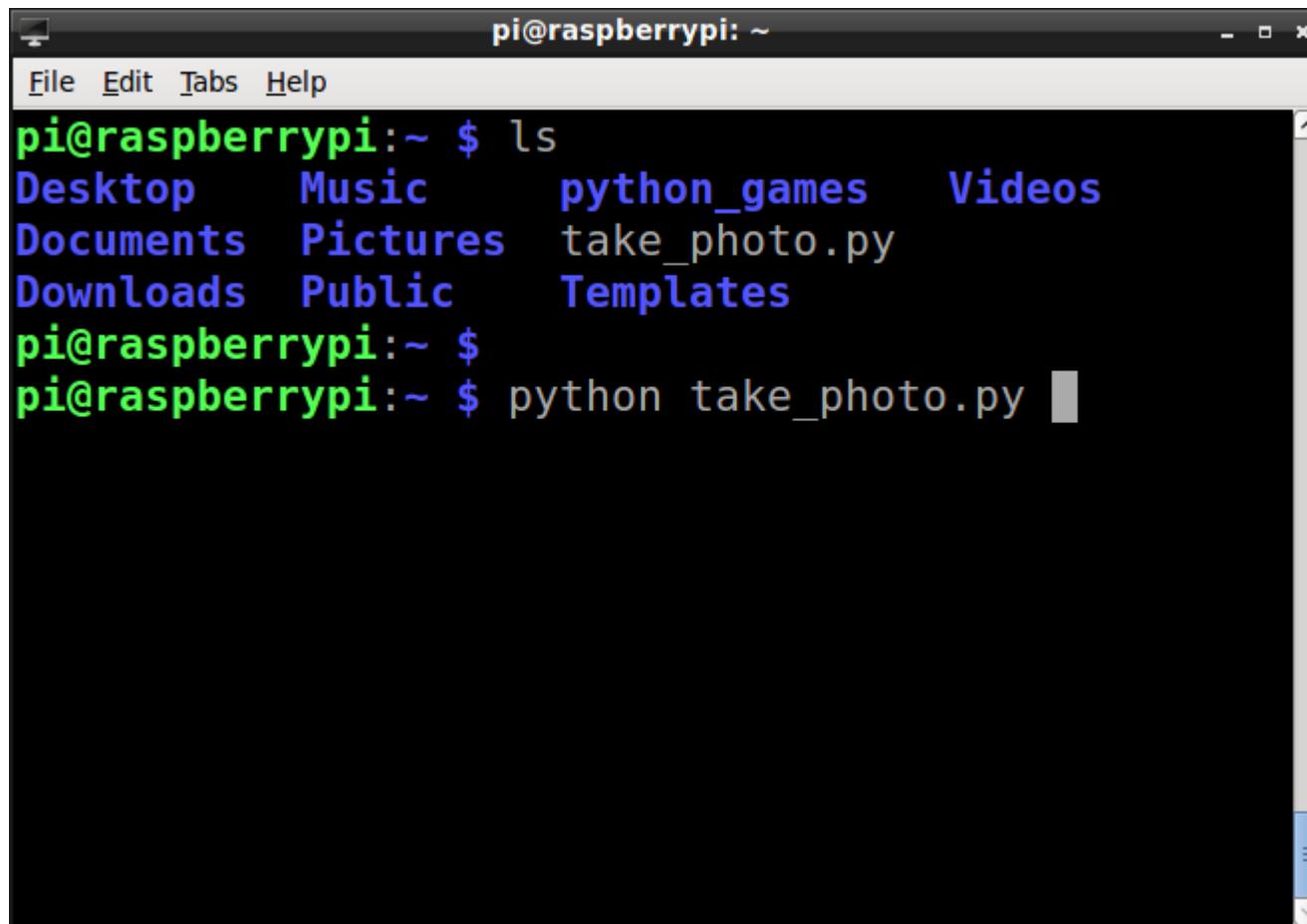
# 存檔

- 存檔 take\_photo.py



# 執行

- \$ python take\_photo.py



A screenshot of a terminal window titled "pi@raspberrypi: ~". The window has a menu bar with "File", "Edit", "Tabs", and "Help". The terminal displays the following command-line session:

```
pi@raspberrypi:~ $ ls
Desktop Music python_games Videos
Documents Pictures take_photo.py
Downloads Public Templates
pi@raspberrypi:~ $
pi@raspberrypi:~ $ python take_photo.py
```

# DEMO

## take\_photo.py

```
$ cd ~/cam-py-cv/day1/02-picamera
$ python take_photo.py
```

# 錄影

```
#!/usr/bin/python
import picamera

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

- 錄 3 秒鐘影像，儲存到檔案 video.h264
- 預設錄影格式為 H.264/AVC 壓縮，解析度 1280x800

# DEMO

## record\_video.py

```
$ cd ~/cam-py-cv/day1/02-picamera
$ python record_video.py
```

# 低光源拍照

```
import picamera
import time
from fractions import Fraction

camera = picamera.PiCamera()
camera.resolution = (640, 480)
camera.framerate = Fraction(1, 6)
camera.shutter_speed = 6000000 單位 :microsecond
camera.iso = 800
time.sleep(30)
camera.exposure_mode = 'off'
camera.capture('dark.jpg')
```

# DEMO

## low\_light.py

```
$ cd ~/cam-py-cv/day1/02-picamera
$ python low_light.py
```

# 練習

- 請查詢網頁文件，在拍照後的圖片疊上文字  
<http://goo.gl/2ShIrQ>
- 文字包含年，月，日，分，時，秒，週  
例如 Wed Apr 19 18:37:08 2017
- 提示：疊表示 overlay

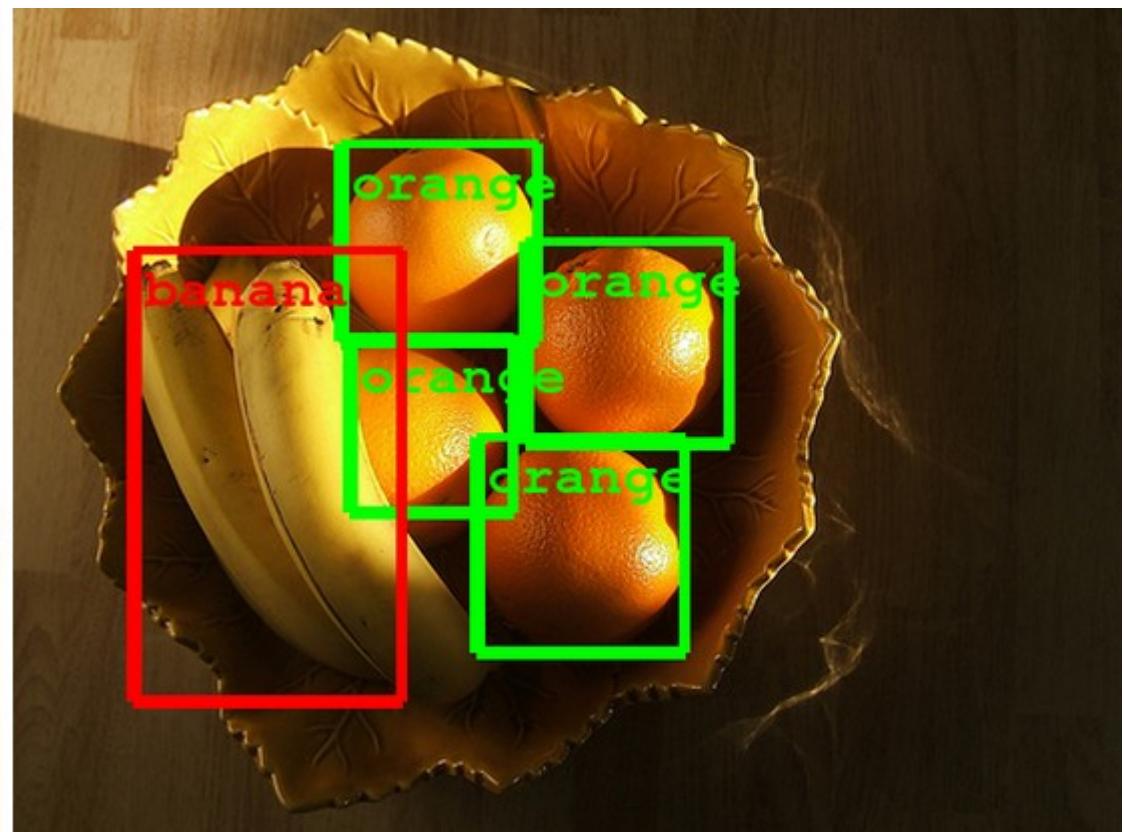
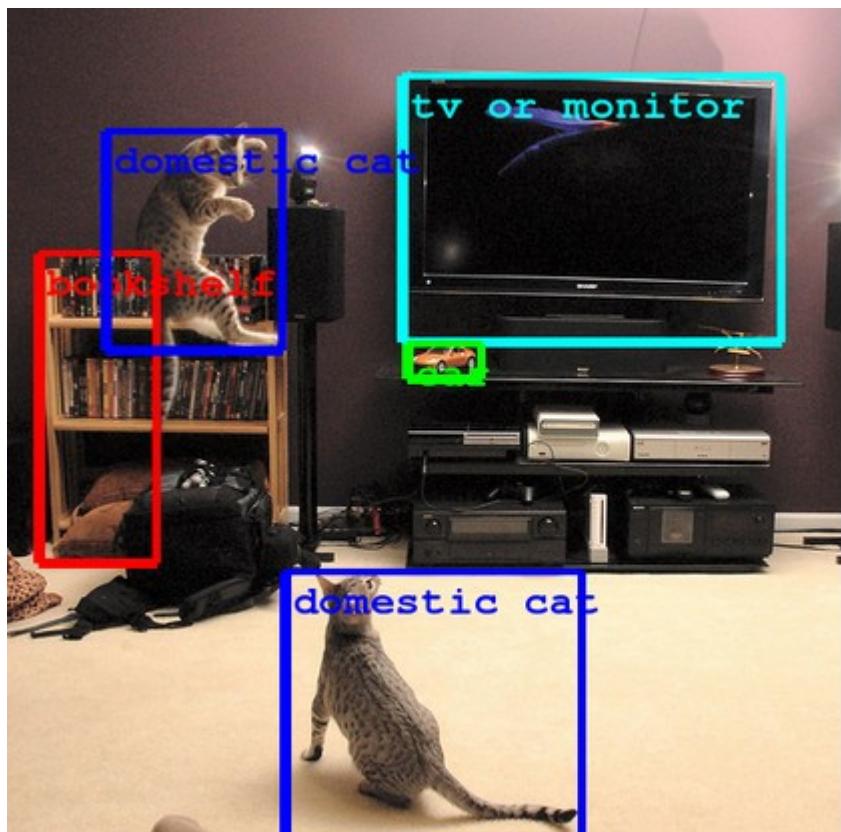
# 小結

- picamera 是 Pi Camera 的 Python 套件
- 查詢網頁文件 <http://goo.gl/2ShIrQ>

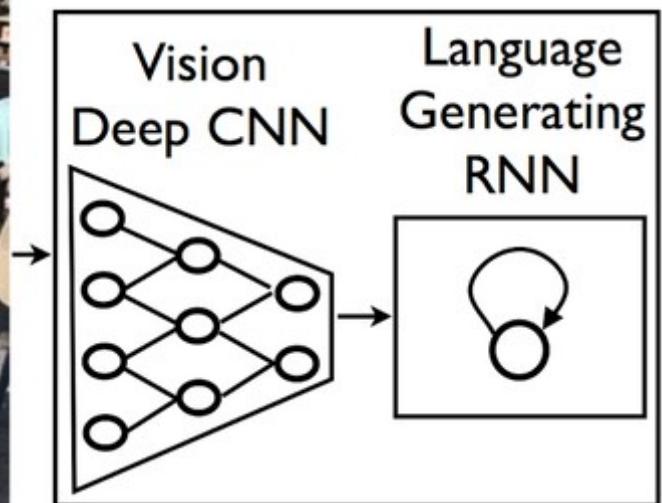
# 實驗 3：會認東西的 Camera

目的：串接網路服務

# 影像辨識



# 更強大的看圖說故事



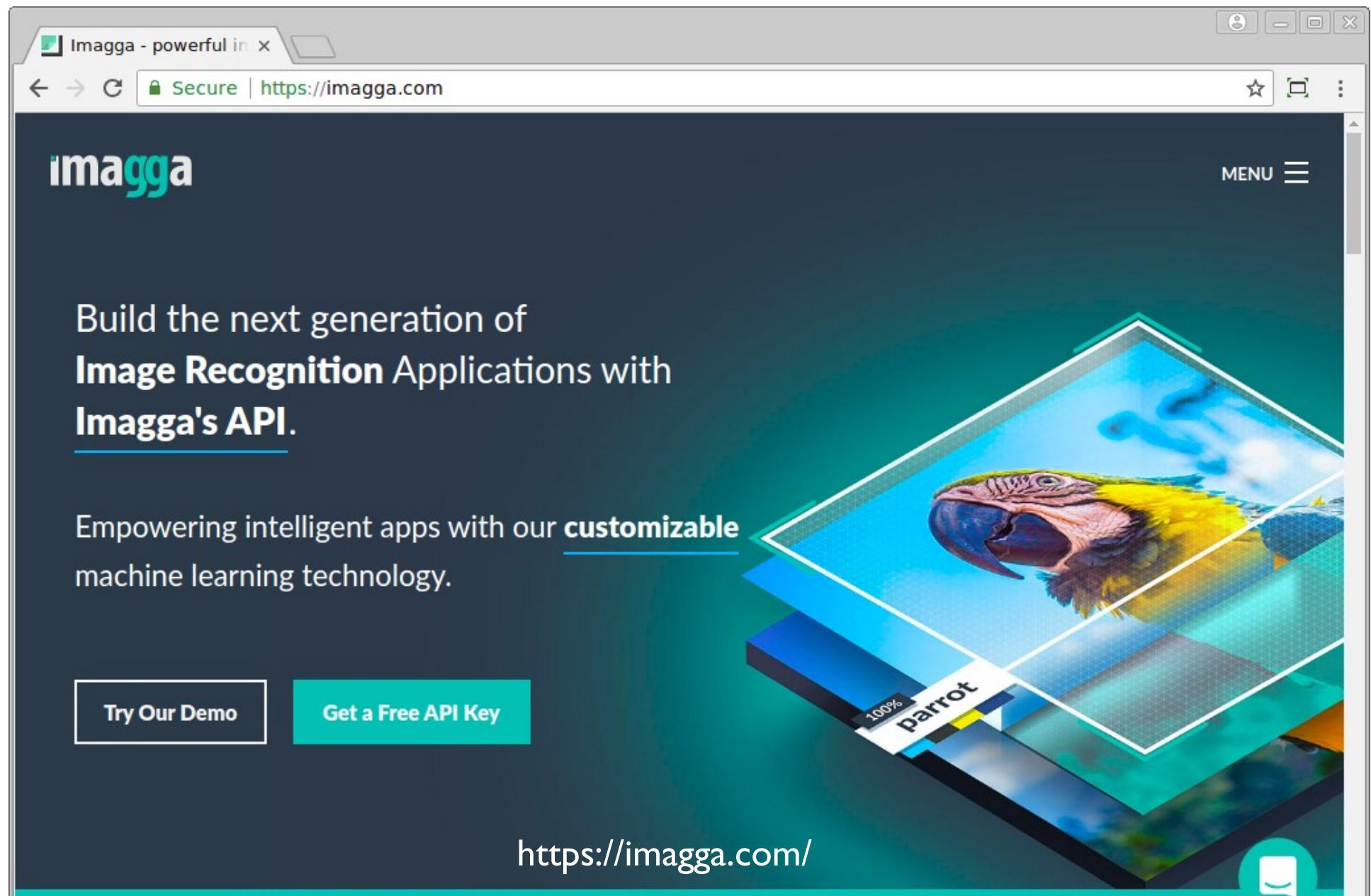
**A group of people shopping at an outdoor market.**

**There are many vegetables at the fruit stand.**

CNN : Convolutional Neural Network( 捲積式類神經網路 )

RNN : Recurrent Neural Network( 遞迴式類神經網路 )

# 影像分類服務



The screenshot shows a web browser displaying the Imagga website at <https://imagga.com>. The page features a dark teal background with the Imagga logo in white and green. A large, stylized image of a parrot's head is shown within a 3D perspective grid. The text "Build the next generation of **Image Recognition** Applications with **Imagga's API.**" is displayed above the grid. Below the grid, a call-to-action button says "Try Our Demo". Another button below it says "Get a Free API Key". The URL <https://imagga.com/> is visible at the bottom of the page.

Imagga - powerful in x

Secure | <https://imagga.com>

imaga MENU

Build the next generation of **Image Recognition** Applications with **Imagga's API.**

Empowering intelligent apps with our **customizable** machine learning technology.

Try Our Demo

Get a Free API Key

<https://imagga.com/>

# 看 DEMO

<https://imagga.com/auto-tagging-demo>

# Auto-Tagging

<https://imagga.com/auto-tagging-demo>

# 如何開始使用服務？

## 1. 註冊與認證

- <https://imagga.com/auth/signup>

## 2. 取得 Authorization

- <https://imagga.com/profile/dashboard>

## 3. 串接

- input：程式指定圖片的 URL 或是上傳圖檔
- output：JSON 字串

# Dashboard

- Basic authentication 使用 authorization value

The screenshot shows the Imagga User Dashboard. At the top right, it displays "Active subscription: DEVELOPER \$14/m | Change plan". On the left, a sidebar menu includes "Dashboard" (selected), "Billing details", "Account details", "Invoices", and "Logout". The main content area shows "API Usage: 07.16.2016 - 08.15.2016 [MM.DD.YYYY]" with monthly and daily usage counts (0 / 12,000 and 0). Below this, the "API Details:" section lists "API Key", "API Secret", and "Authorization". The "Authorization" field is highlighted with a red box. To the right, there's an "API Endpoint" section with a sample curl request:

```
curl
-u "acc_f4526858204b9dd:ca5c7ee8c1defb9218ff340ae3f8f512"
http://api.imagga.com/v1/tagging?url=http://imagga.com/sta
```

**Authorization 的值會用到**

<https://imagga.com/profile/dashboard>

# HTTP 兩三件事

# GET & POST Method

- RFC 2616 / Hypertext Transfer Protocol - HTTP/1.1
- HTTP method
  - OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT
- GET 像明信片
  - 資料 (query string) 在 URL 傳送
- POST 像信紙 + 信封
  - 資料可以在 message-header
  - 也可以在 message-body



# 把設定檔寫成獨立的 conf 檔案

- \$ nano web\_service.conf

```
[imagga]
authorization = Basic
YWNgXzJkYzdkNzNjMmYwODliMToxYzQ3Yzg2ZDg0YjdM
jdjYjZjNzQ1NTQ1MmYwNTgzMQ==
```



將黃色部份換成自己的 authorization

# 用 import ConfigParser 讀取 conf 檔案

```
import ConfigParser

config = ConfigParser.ConfigParser()
config.read('web_service.conf')
authorization = config.get('imagga', 'authorization')
```



這是從 conf 讀出的變數

# Python 程式串接 (File URL)

```
import requests

authorization = config.get('imagga', 'authorization')
url = "http://api.imagga.com/v1/tagging"
querystring = {"url": "網路上圖檔的URL 網址"}
headers = {
 'accept': "application/json",
 'authorization': authorization
}
response = requests.request("GET", url, headers=headers,
 params=querystring)
```

tagging API 的 URL

這是從 conf 讀出的變數

# 回傳結果 (JSON)

```
{
 "results": [
 {
 "image": "http://playground.imagga.com/static/img/example_photo.jpg",
 "tags": [
 {
 "confidence": 100,
 "tag": "shore"
 },
 {
 "confidence": 52.19003265126093,
 "tag": "sea"
 },
 {
 }
]
 }
]
}
```

- JSON(JavaScript Object Notation) 是一種資料結構
  - 物件 (object) 以 `{ }` 表示
  - 鍵 / 值 (collection) 以 `:` 表示
  - 陣列 (array) 以 `[ ]` 表示

# 解析 JSON

```
import json

authorization = config.get('imagga', 'authorization')
url = "http://api.imagga.com/v1/tagging"
querystring = {"url": "網路上圖檔的URL 網址"}
headers = {
 'accept': "application/json",
 'authorization': authorization
}
response = requests.request("GET", url, headers=headers,
params=querystring)

data = json.loads(response.text.encode("ascii"))
print data["results"][0]["tags"][0]["tag"].encode("ascii")
```

這是從 conf 讀出的變數

# 解析 JSON

```
{
 "results": [
 {
 "image": "http://playground.imagga.com/static/img/example_photo.jpg",
 "tags": [
 {
 "confidence": 100,
 "tag": "shore"
 },
 {
 "confidence": 52.19003265126093,
 "tag": "sea"
 },
 {

```

```
data = json.loads(response.text.encode("ascii"))
print data["results"][0]["tags"][0]["tag"].encode("ascii")
```

# 解析 JSON

```
{
 "results": [
 {
 "image": "http://playground.imagga.com/static/img/example_phc",
 "tags": [
 {
 "confidence": 100,
 "tag": "shore"
 },
 {
 "confidence": 52.19003265126093,
 "tag": "sea"
 },
 {
 "confidence": 33.33333333333333,
 "tag": "water"
 }
]
 }
]
}
```

```
data = json.loads(response.text.encode("ascii"))
print data["results"][0]
```

# 解析 JSON

```
{
 "results": [
 {
 "image": "http://playground.imagga.com/static/img/example_phot
 "tags": [
 {
 "confidence": 100, tags[0]
 "tag": "shore"
 },
 {
 "confidence": 52.19003265126093,
 "tag": "sea"
 },
 {
 "confidence": 33.33333333333333, tags[2]
 "tag": "water"
 }
]
 }
]
}
```

```
data = json.loads(response.text.encode("ascii"))
print data["results"][0]["tags"][0]
```

# 解析 JSON

```
{
 "results": [
 {
 "image": "http://playground.imagga.com/static/img/example_phot
 "tags": [
 {
 "confidence": 100, "tag": "shore"
 },
 {
 "confidence": 52.19003265126093, "tag": "sea"
 },
 {
 "confidence": 33.33333333333333, "tag": "water"
 }
]
 }
]
}
```

result[0]

tags[0]

tag

```
data = json.loads(response.text.encode("ascii"))
print data["results"][0]["tags"][0]["tag"].encode("ascii")
```

# DEMO

## imagga\_tag\_file\_url.py

```
$ cd ~/cam-py-cv/day1/03-imagga_web_service
$ python imagga_tag_file_url.py
```

# Python 程式串接 (Upload File)

- 先拍張照片
- 根據文件得知查詢上傳的檔案需要兩個步驟
  - 上傳檔案後取得檔案 uid
  - 將 uid 以參數方式送出查詢

# 上傳檔案取得檔案 uid

```
import requests
import json

url = "http://api.imagga.com/v1/content"
files = {"file": open("/home/pi/test.jpg","rb")}
headers = {
 'accept': "application/json",
 'authorization': authorization
}

response = requests.post(url, files=files, headers=headers)
print(response.text)
data = json.loads(response.text.encode("ascii"))
print data["uploaded"][0]["id"]
```

不同的 URL

# 將 uid 以參數方式送出查詢

```
... 接前頁
url = "http://api.imagga.com/v1/tagging"
querystring = {"content":data["uploaded"][0]["id"]}
response = requests.request("GET", url, headers=headers,
params=querystring)
data = json.loads(response.text.encode("ascii"))
print data["results"][0]["tags"][0]["tag"].encode("ascii")
```



uid

# DEMO

## imagga\_tag\_upload\_file.py

```
$ cd ~/cam-py-cv/day1/03-imagga_web_service
$ python imagga_tag_upload_file.py
```

# 實做雲端相機

# 拆解功能

- 執行 python 程式後的步驟
  1. 拍照 & 存檔
  2. 將檔案上傳後取得 uid
  3. 再將 uid 以參數方式送出查詢
  4. 將查詢結果用 TTS(Text To Speech) 發聲

```
$ echo tag | festival --tts
```



由 imagga 回傳的 tag

# 拍照後的連續動作

```
camera = picamera.PiCamera()
camera.capture("test.jpg")
files = {"file": open("test.jpg", "rb")}
```

```
url = "http://api.imagga.com/v1/content"
response = ...
```



上傳圖檔

```
url = "http://api.imagga.com/v1/tagging"
querystring = ...
data = json.loads(...)
```



查詢圖檔 tag

```
obj = data[...]["tag"]
print "<< " + obj + " >>"
cmd = "echo " + obj + " | festival --tts"
os.system(cmd)
```



喇叭發聲 (TTS)

# 練習

- 實做能自動辨識物體的相機，可在拍照後的圖片疊上第一個 tag( 文字 )+ 第一個信心水準 (confidence)  
<http://goo.gl/2ShIrQ>
- 拆解功能
  1. 拍照 & 存檔
  2. 將檔案上傳後取得 uid
  3. 再將 uid 以參數方式送出查詢
  4. 將文字疊在照片上，信心水準型態為 float 需轉型

# 小結

- Python 的 request 套件可提供 get 和 post 方法
- 網路 RESTful API 回傳結果通常為 JSON 格式

# 實驗 4 : Video4Linux 2nd(V4L2)

目的：從 OpenCV 看 Camera 和 Webcam



# OpenCV

- Open Source Computer Vision Library

- 跨平台的計算機函式庫，主要由 C/C++ 撰寫

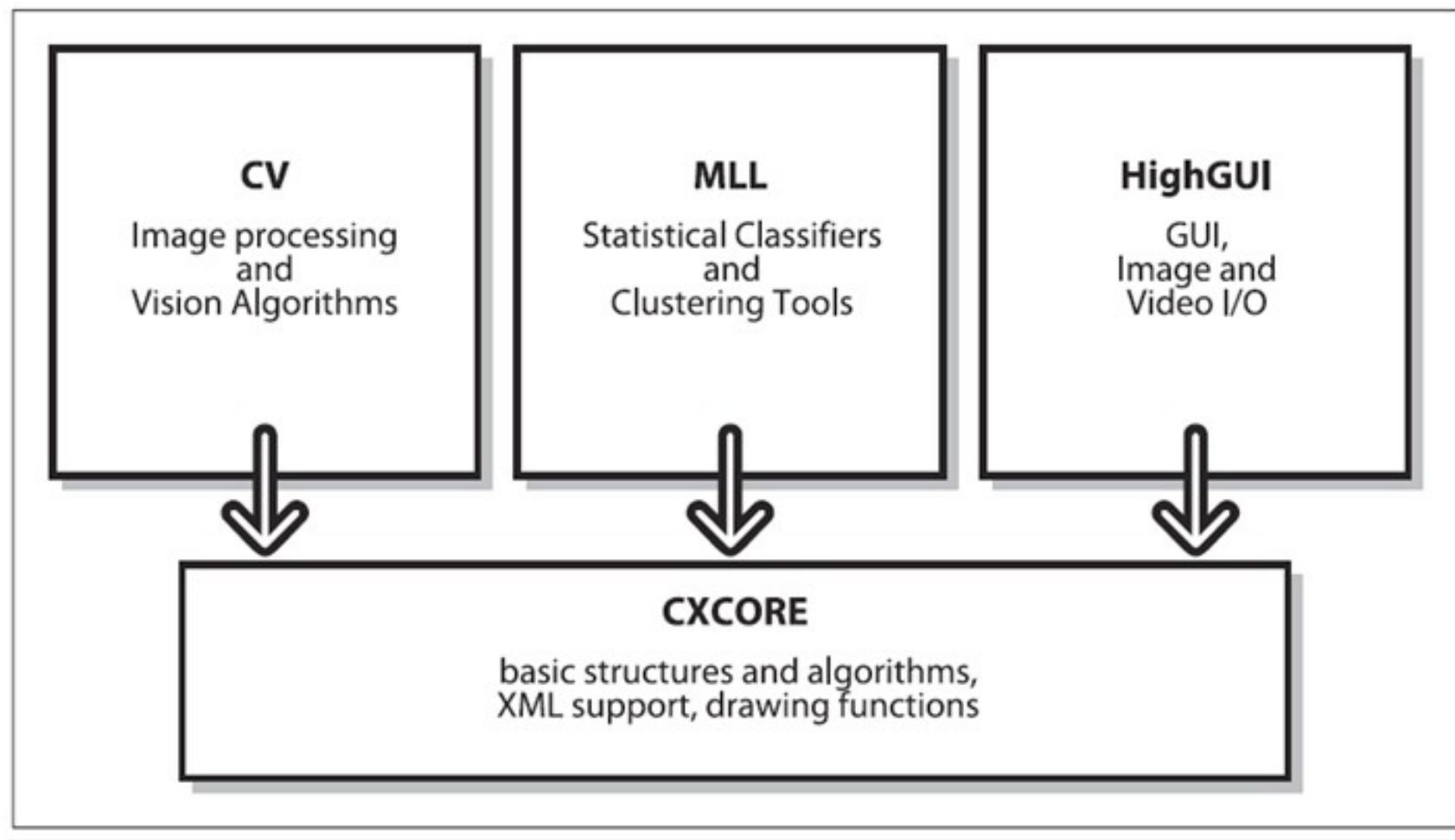
**OpenCV Overview:** > 500 functions

[opencv.willowgarage.com](http://opencv.willowgarage.com)

The collage illustrates the wide range of applications and algorithms supported by OpenCV, including:

- General Image Processing Functions:** Includes image loading, saving, and basic processing.
- Image Pyramids:** Shows coarse-to-fine optical flow estimation.
- Robot support:** A robot arm interacting with objects.
- Segmentation:** Techniques like background subtraction and depth maps.
- Geometric descriptors:** Illustrations of SIFT and SURF descriptors.
- Features:** Extraction of features from images.
- Tracking:** Real-time tracking of objects.
- Transforms:** Affine and perspective transformations.
- Machine Learning:** Detection and recognition.
- Camera calibration, Stereo, 3D:** Reconstruction of 3D scenes.
- Utilities and Data Structures:** A diagram of the OpenCV architecture showing its layered structure.
- Fitting:** Examples of curve fitting.
- Matrix Math:** Basic linear algebra operations.

# OpenCV 的架構



# 載入圖檔並顯示

```
import cv2
import sys

imagePath = sys.argv[1]
image = cv2.imread(imagePath)

cv2.imshow("preview", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

# DEMO

## image\_load.py

```
$ cd ~/cam-py-cv/day1/04-webcam
$ python image_load.py /home/pi/test.jpg 142
```

# OpenCV 還提供了許多好用的函式

- 在圖上疊字

```
cv2.putText(img, TEXT, (x,y), font, scale,
color)
```

- 範例：

```
cv2.putText(image, "HELLO", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255))
```

BGR

# 除了開檔與顯示以外

- 將影像存成檔案

```
cv2.imwrite(filename, image)
```

- 範例：

```
cv2.imwrite("puttext.png", image)
```

# DEMO

## puttext.py

```
$ cd ~/cam-py-cv/day1/04-webcam
$ python puttext.py abba.png
```

除了拍照存檔後開圖以外，  
可以即時預覽 Camera 的結果嗎？

# 讀取 Camera 並顯示

```
import cv2
import imutils

cap = cv2.VideoCapture(0)

while True:
 ret, frame = cap.read()
 frame = imutils.resize(frame, 320)
 cv2.imshow("preview", frame)
 if cv2.waitKey(1) & 0xFF == ord("q"):
 break

cap.release()
cv2.destroyAllWindows()
```

0 是 V4L2 的裝置節點

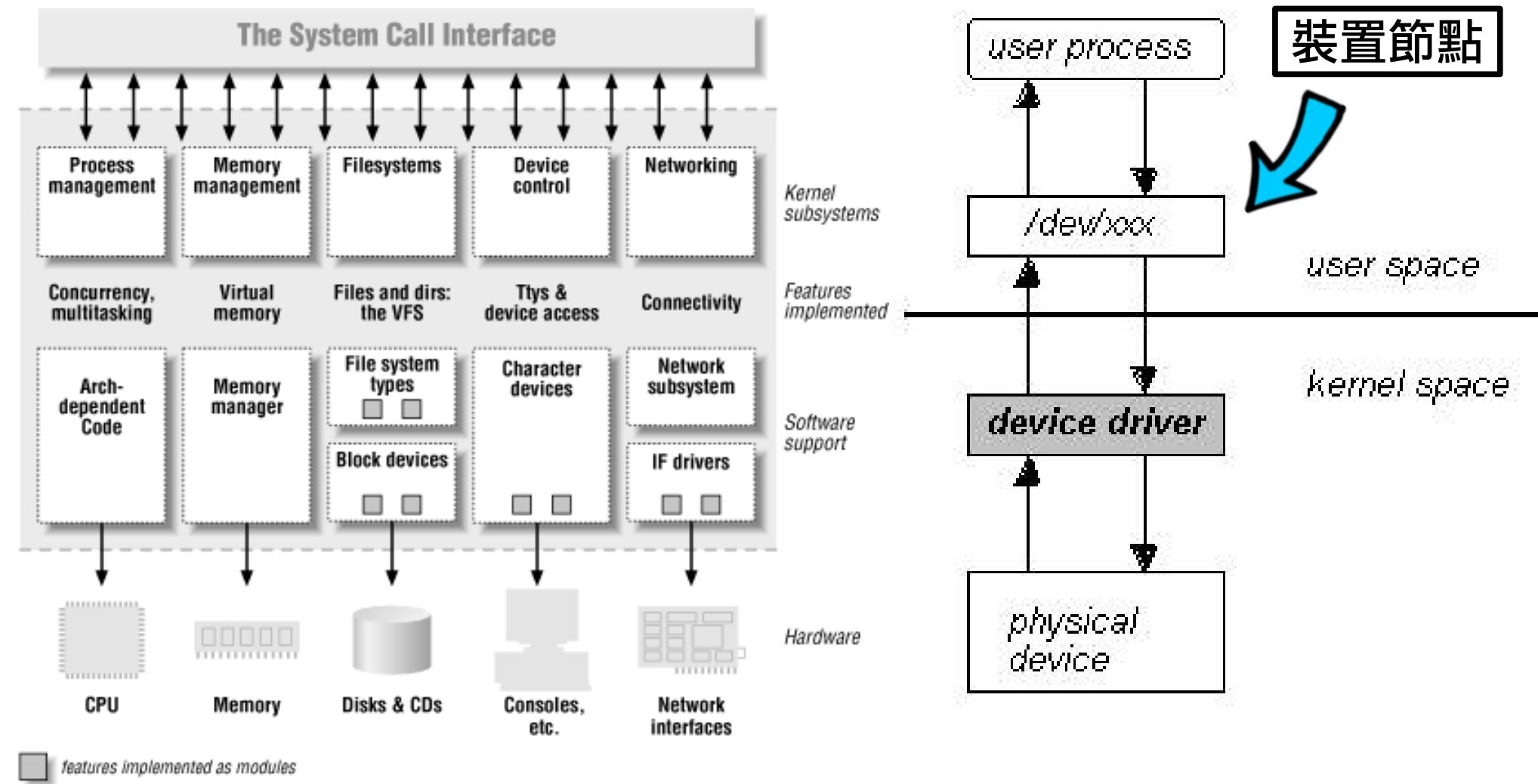


回來看 Camera  
你有用過 Webcam 嗎？

# Webcam

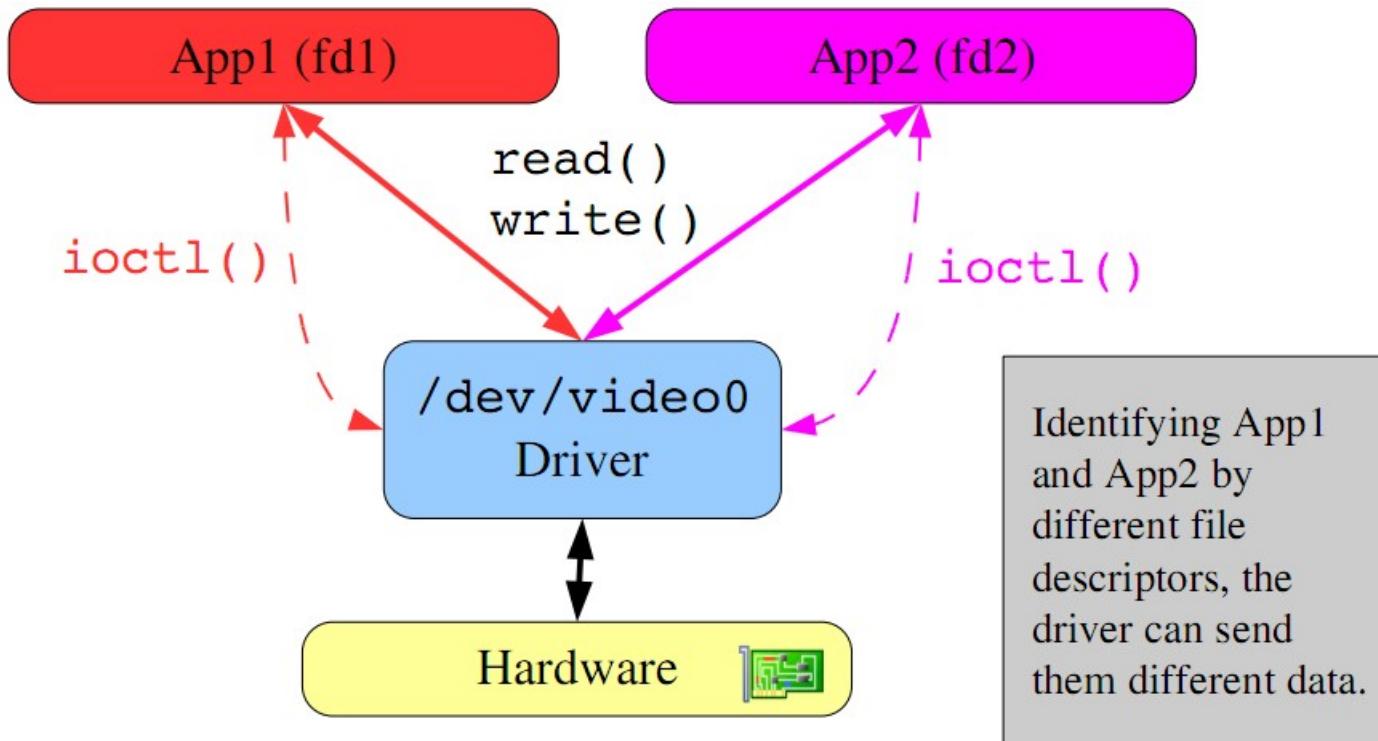


# Linux 如何存取硬體？



# Video For Linux 2nd(V4L2)

- 是 Linux 對視訊設備（如 Webcam）的 Userspace API

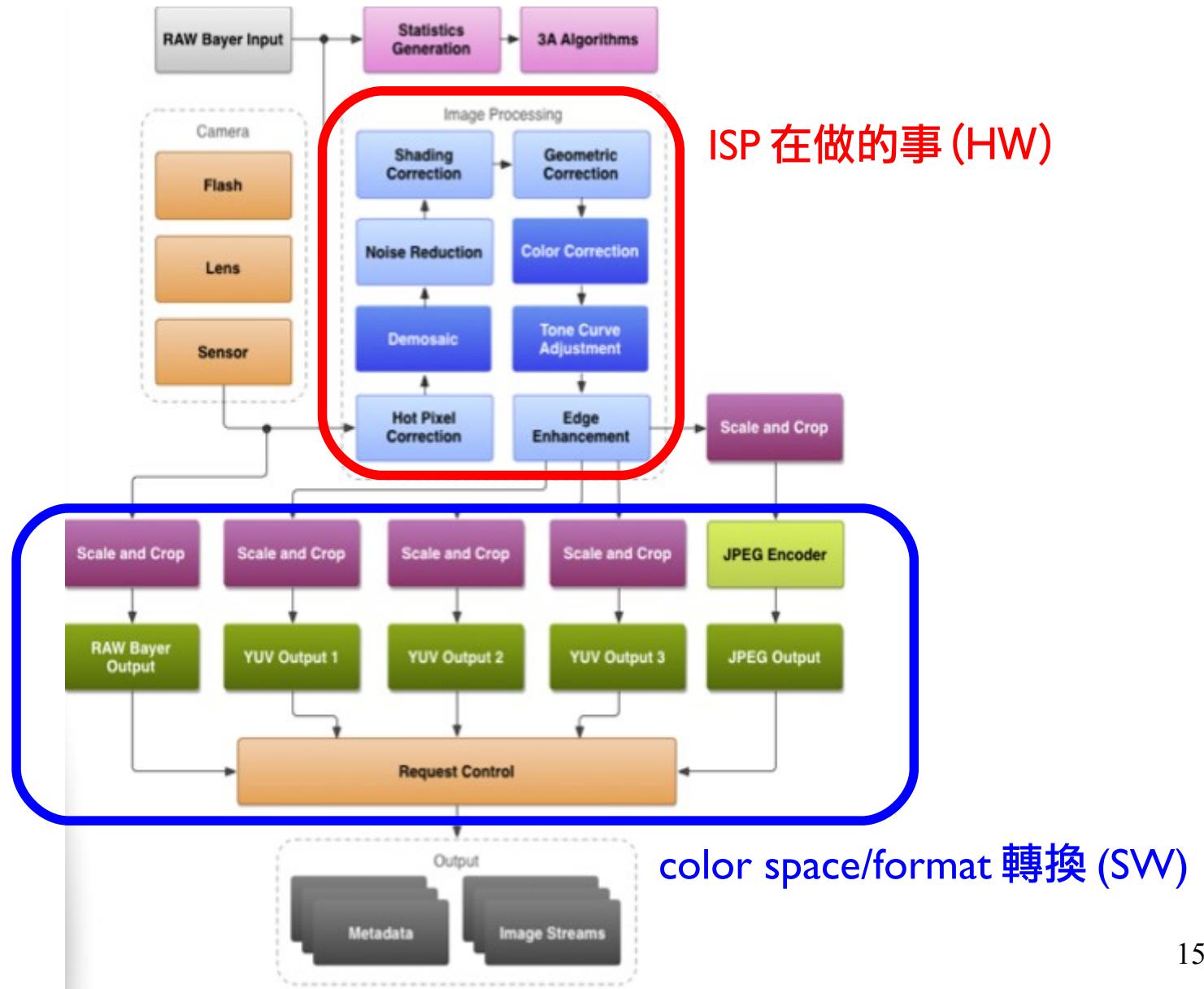


[http://free-electrons.com/doc/embedded\\_linux\\_multimedia.pdf](http://free-electrons.com/doc/embedded_linux_multimedia.pdf)

[https://www.linuxtv.org/downloads/legacy/video4linux/API/V4L2\\_API/spec-single/v4l2.html](https://www.linuxtv.org/downloads/legacy/video4linux/API/V4L2_API/spec-single/v4l2.html)

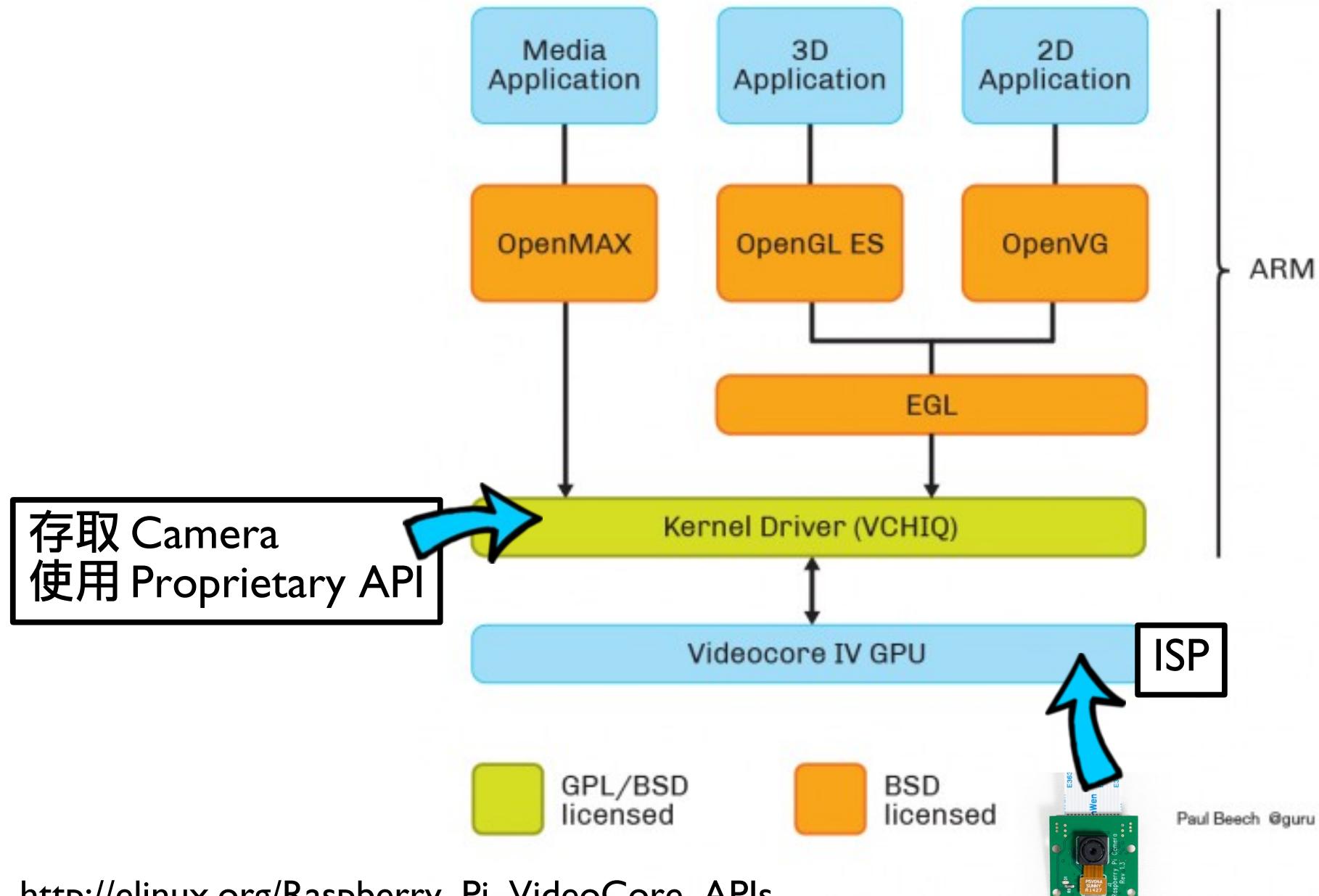
**Camera ≠ Webcam**

# Image Processing Pipeline



# Raspberry Pi Software Architecture

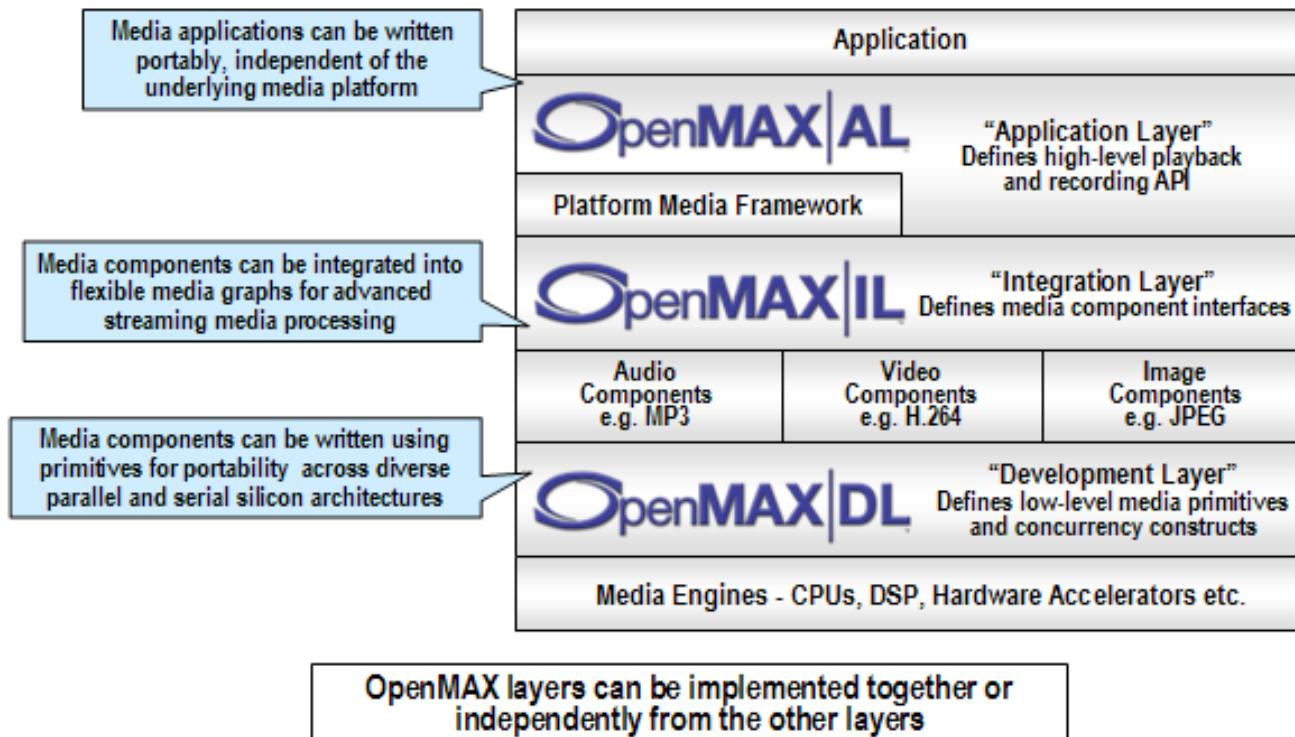
Broadcom BCM2835 SoC



Paul Beech @guru

# OpenMAX

- 開放多媒體加速層 (Open Media Acceleration)
  - 由 Khronos Group 提出的標準
  - 統一的介面，加速大量多媒體資料的處理



# 官方 V4L2 驅動程式

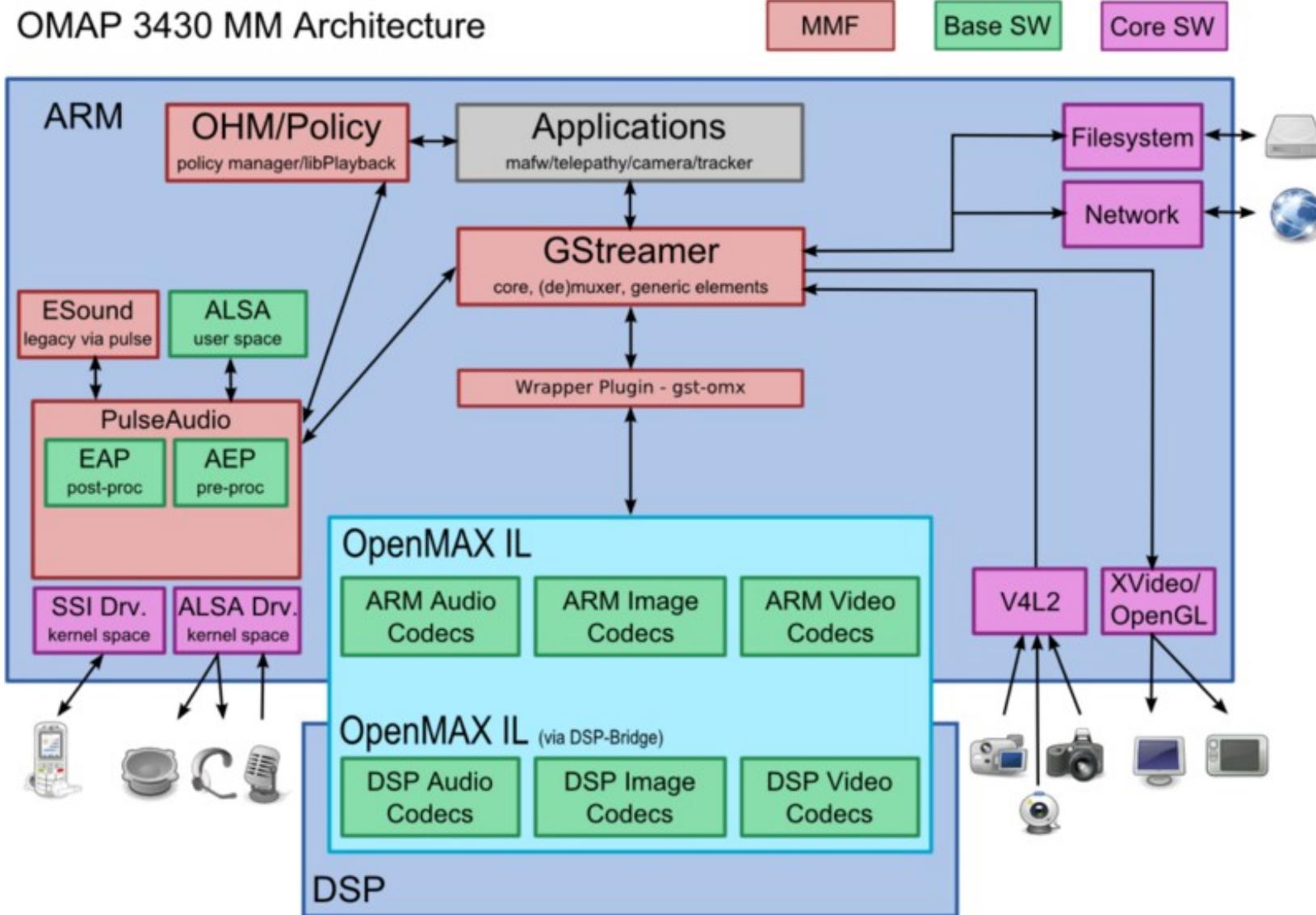
- Kernel driver
- 使用 camera 像是 webcam 一樣
  - \$ sudo modprobe bcm2835-v4l2
- 可直接存取 /dev/videoX
  - \$ v4l2-ctl --list-devices
  - \$ v4l2-ctl --list-formats
  - \$ v4l2-ctl -L

不是數字 1，是小寫 L

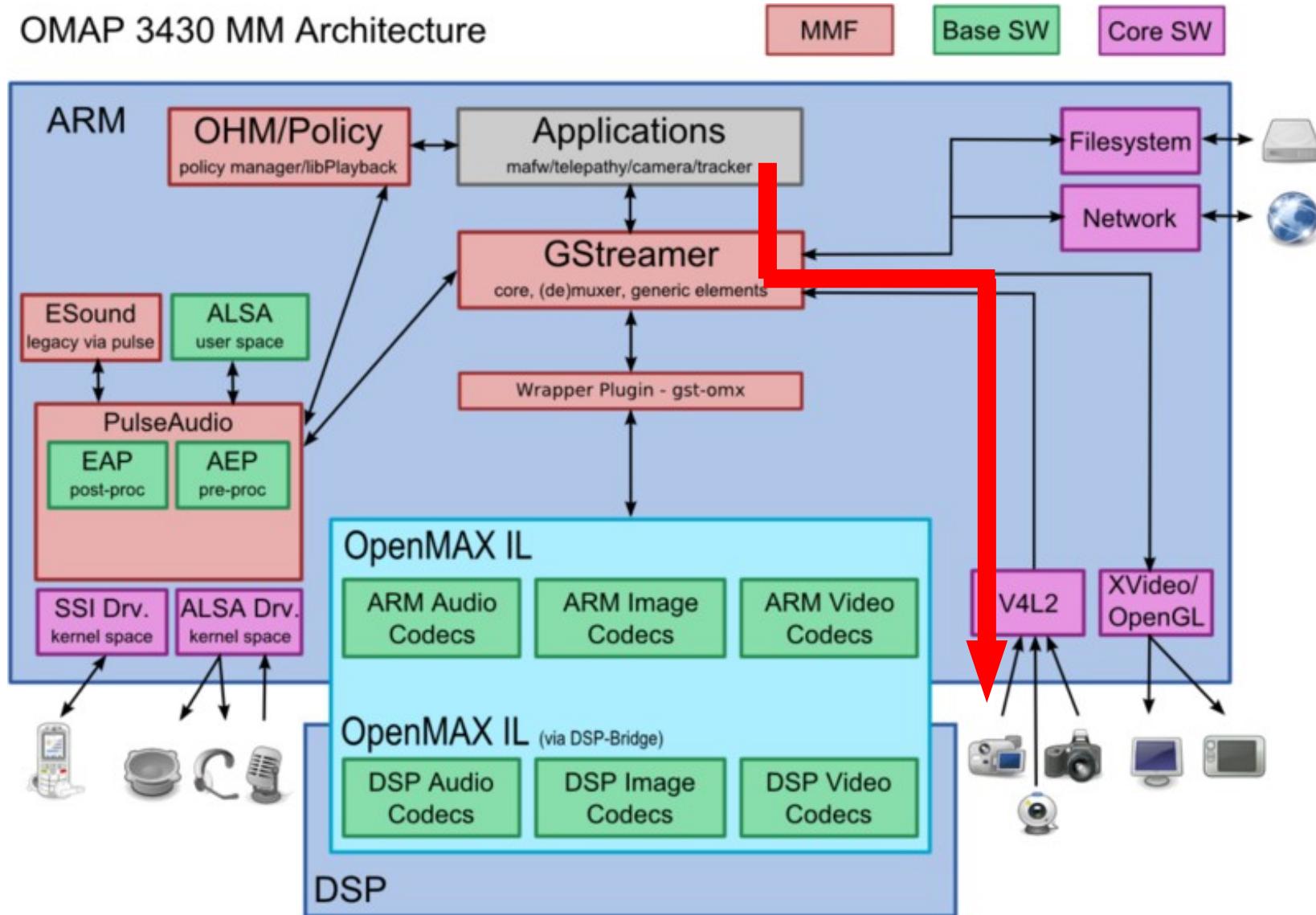


# Multimedia Stack Example

## - TI OMAP 3430



# 走 V4L2



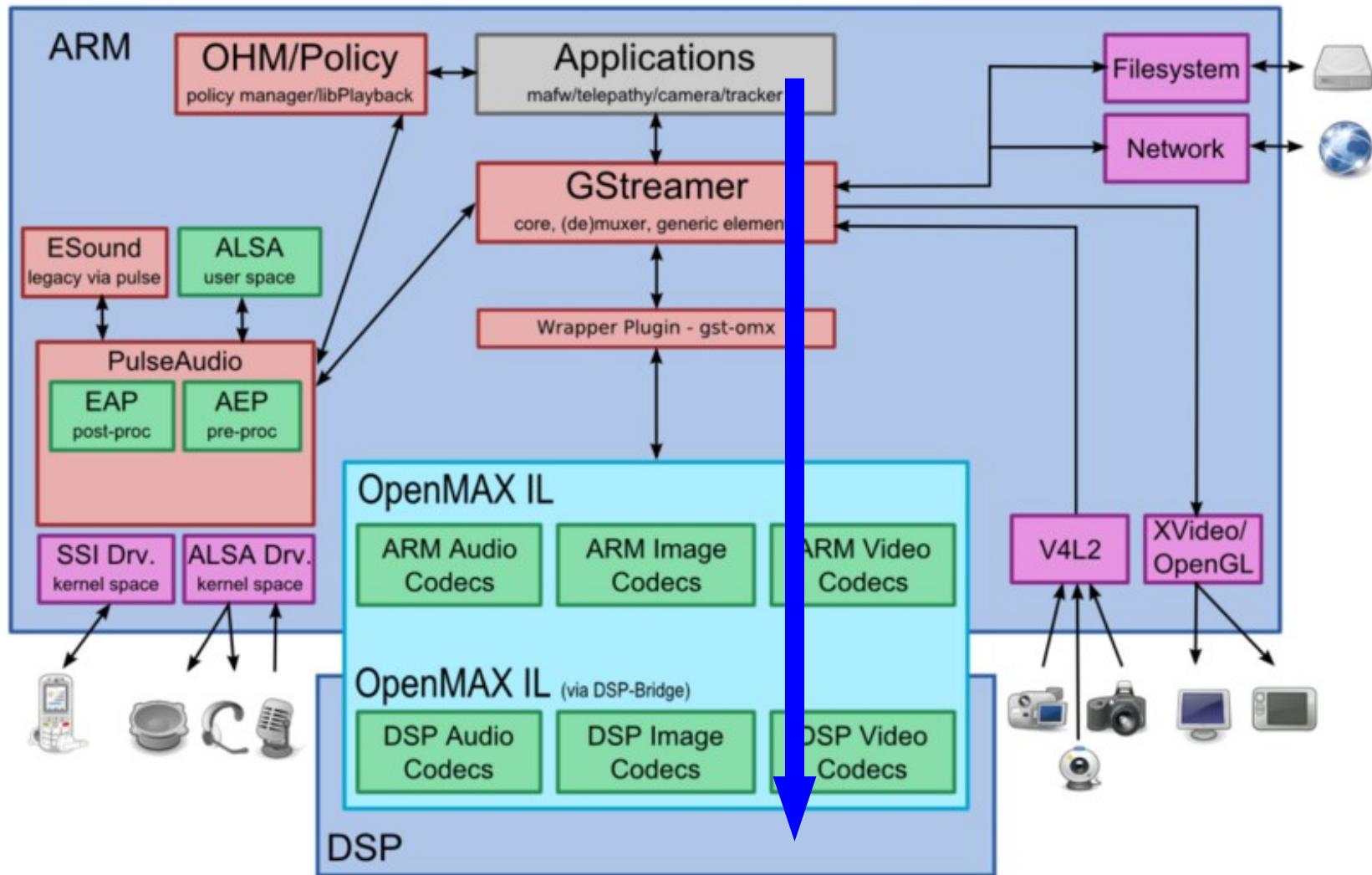
# 走入 OpenMAX

OMAP 3430 MM Architecture

MMF

Base SW

Core SW



# 比較兩種路徑的分別

- OpenMAX

優點：

- 接到 GPU，可硬解

缺點：

- 範例程式少

- MMAL 只有標頭檔

- 不易接 OpenCV

- V4L2

優點：

- 標準的 Linux API

- 多數應用程式可支援

- 可直接接到 OpenCV

缺點：

- CPU 運算，軟解，慢

# 但我想使用 C 語言接到 OpenMAX

- 參考這幾隻程式吧
- raspicam
  - [https://github.com/raspberrypi/userland/tree/master/host\\_applications/linux/apps/raspicam](https://github.com/raspberrypi/userland/tree/master/host_applications/linux/apps/raspicam)
- rpi-omx-tutorial
  - <https://github.com/SonienTaegi/rpi-omx-tutorial>
- omxcam
  - <https://github.com/gagle/raspberry-pi-omxcam>
- rpi-mmal-demo
  - <https://github.com/tasanakorn/rpi-mmal-demo/tree/develop>

使用前記得要先載入模組  
\$ sudo modprobe bcm2835-v4l2



不是數字 1，是小寫 L

## DEMO camera\_preview.py

```
$ cd ~/cam-py-cv/day1/04-webcam
$ python camera_preview.py
```

# 小結

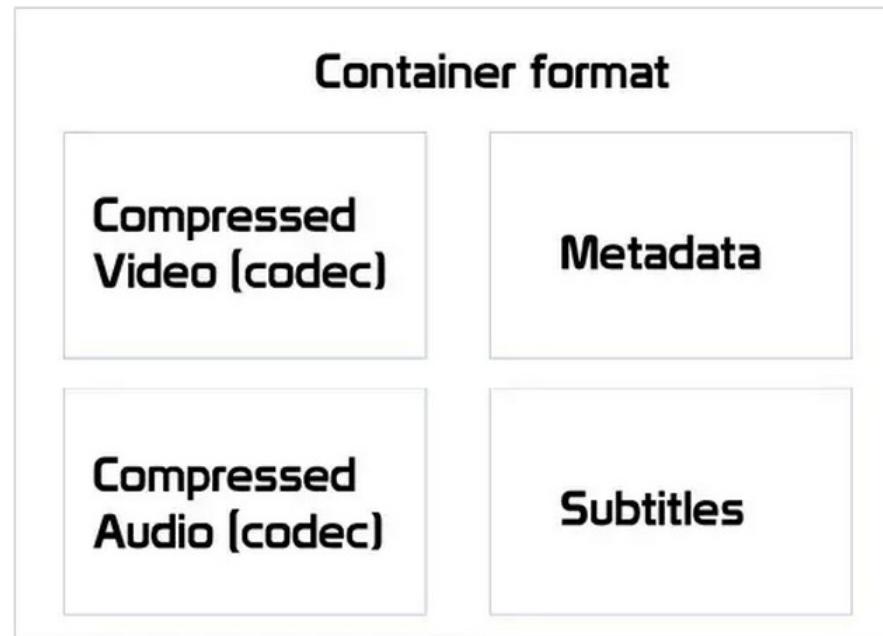
- 指令 `sudo modprobe bcm2835-v4l2` 可讓 camera 使用 V4L2 的 API
- 模組可以放在 `/etc/rc.local` 開機自動載入
  - `sudo modprobe bcm2835-v4l2`
- 也可以放在 `/etc/modules` 開機自動載入
  - `bcm2835-v4l2`

# 實驗 5 : Video Streaming

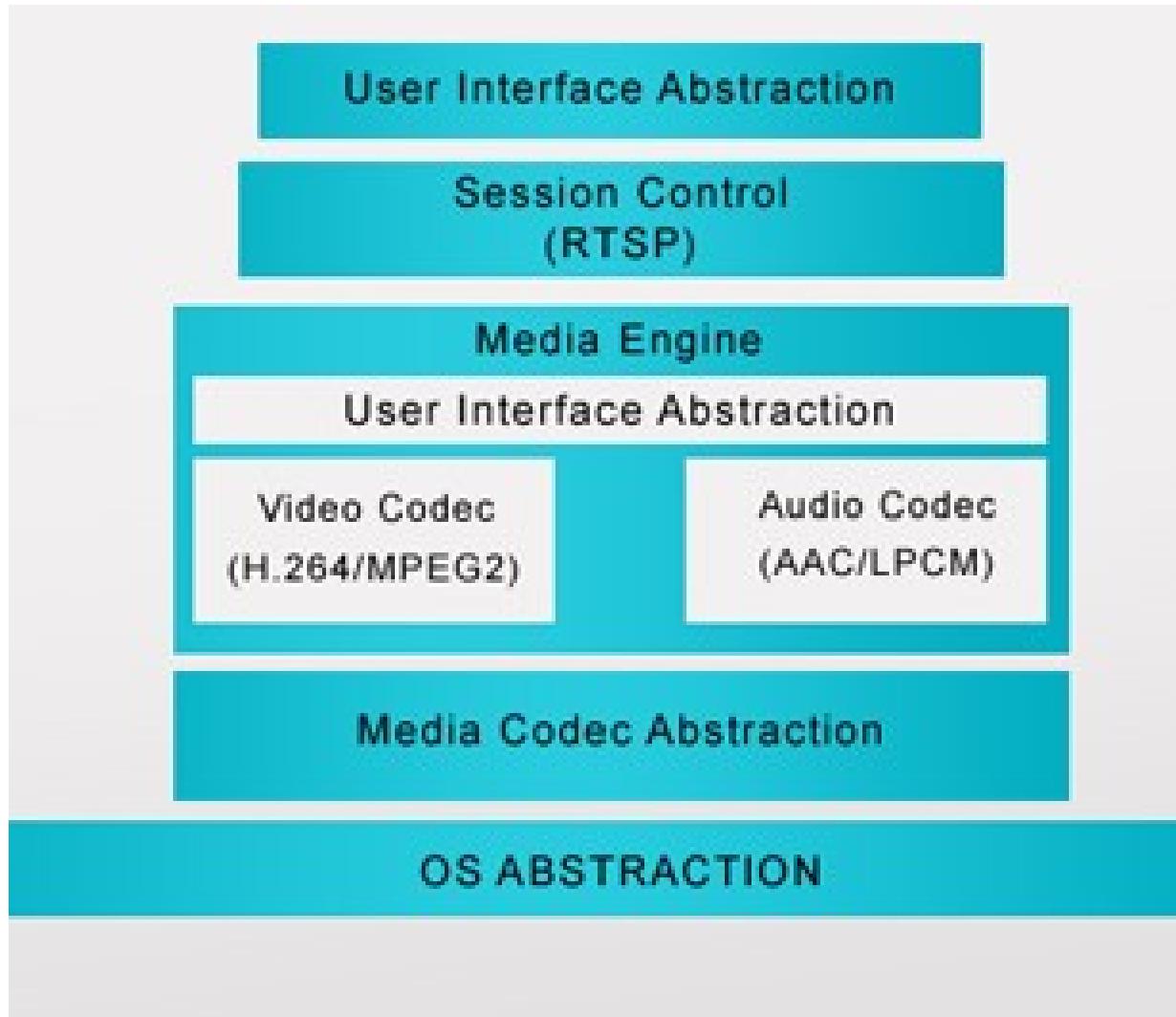
目的：瞭解如何做影像串流

# Codec and Container

- Codec 是 CODer+DECoder
- Codec 是資料壓縮 / 解壓縮的演算法
- Container 是裝載 Video 和 Audio 的容器

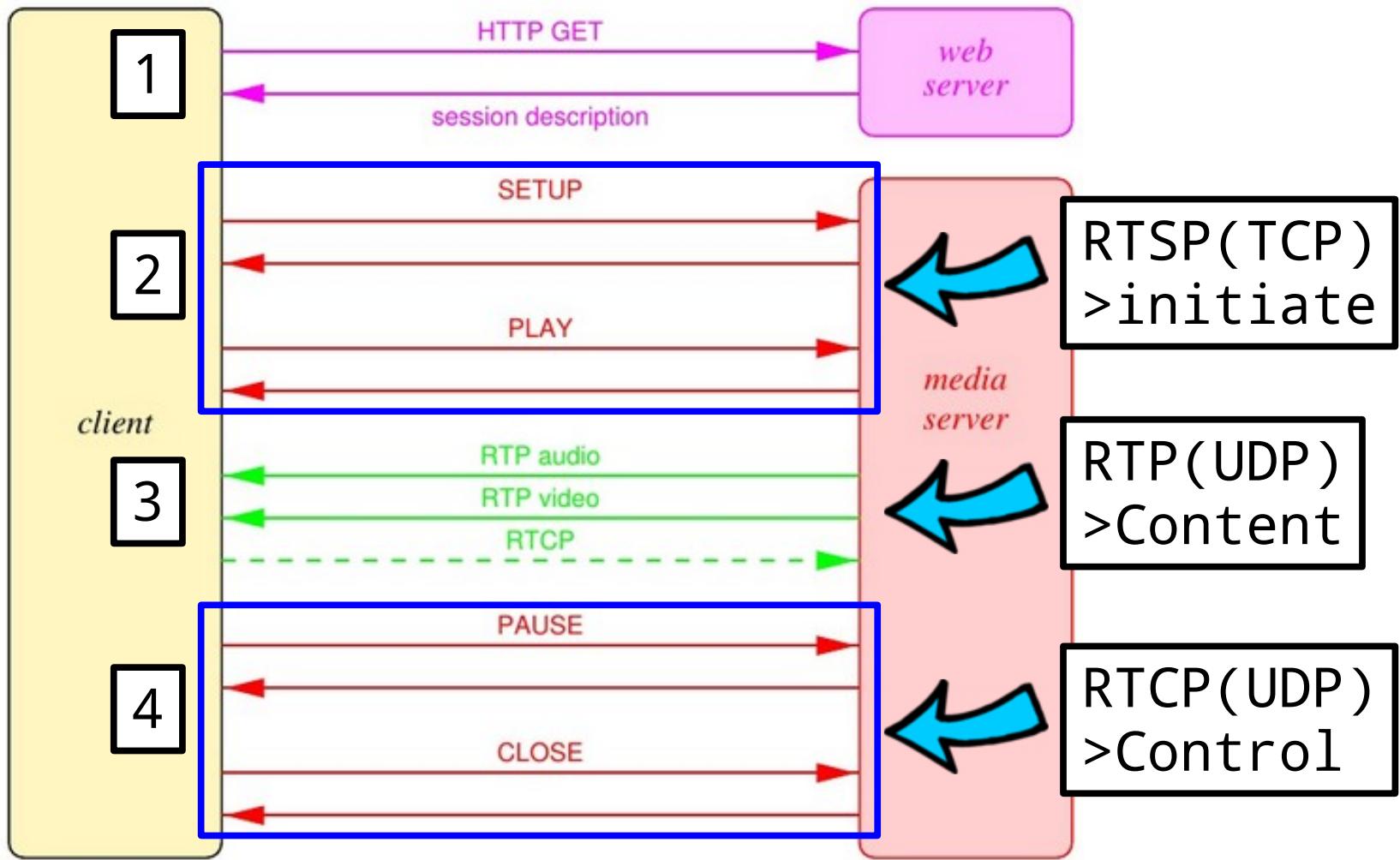


# Streaming Media 技術架構



# 使用 RTSP + H.264

# RTSP 如何運作 ?



# RTSP Server

- 在 Raspberry Pi

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

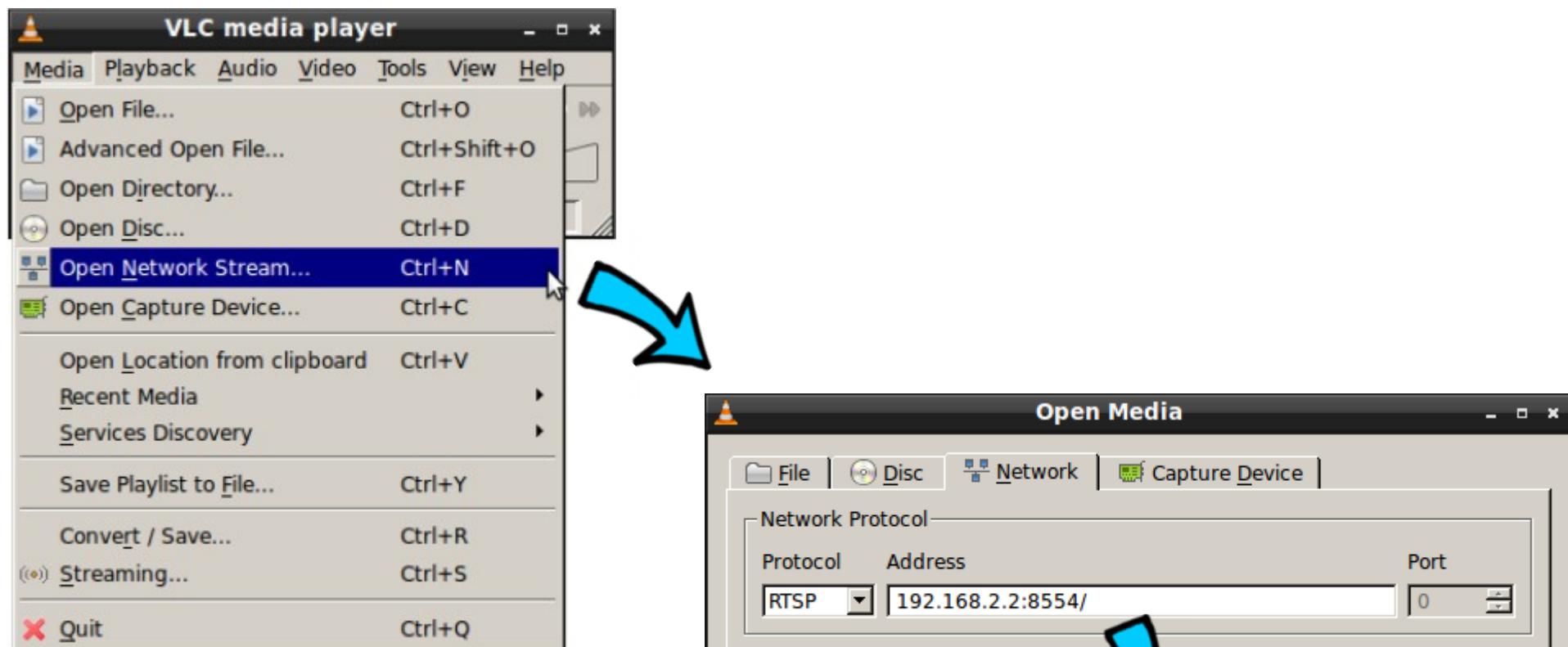
# DEMO

## rtsp\_streaming.sh

```
$ cd ~/cam-py-cv/day1/05-streaming
$./rtsp_streaming.sh
```

# RTSP Client

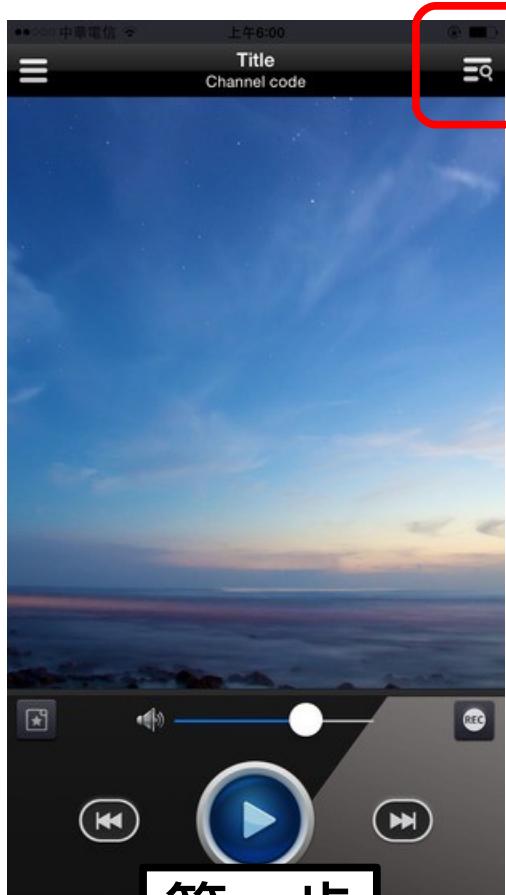
- 在 Windows 使用 VLC



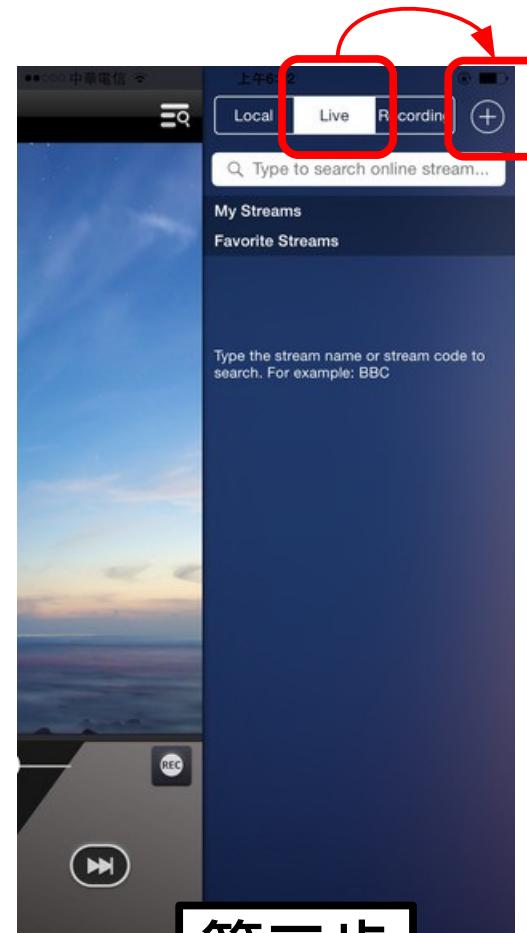
Pi 的 IP:8554/ 斜線很重要

# RTSP Client

- 在 iPhone/iPad 使用 Live Media Player



第一步



第二步

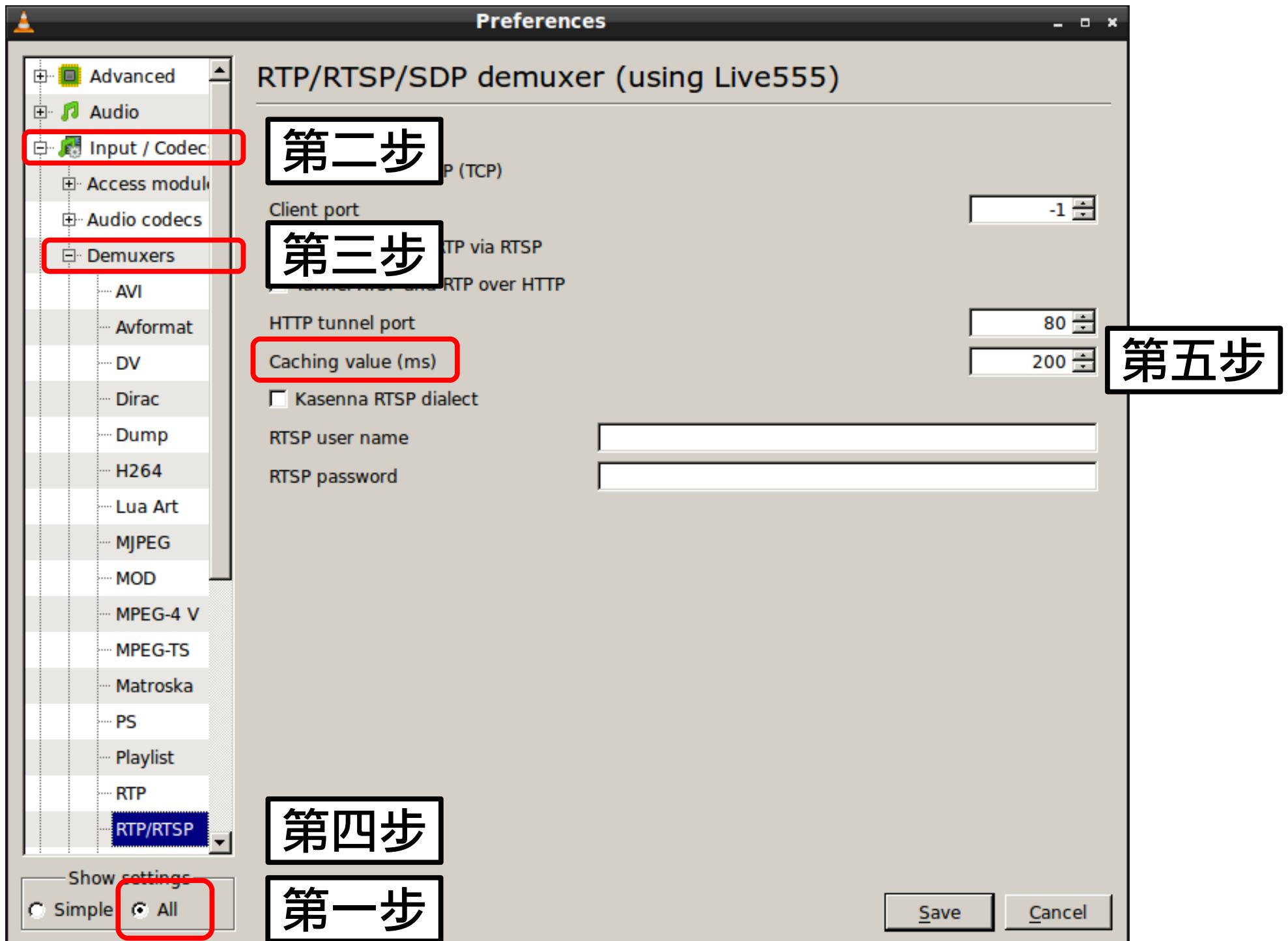


第三步

rtsp://Pi 的 IP:8554/

# 如何降低延遲？

- 方法一：調整 VLC 的參數
  - tools->preferences->all->input/codecs->demuxers->RTP/RTSP->caching value
  - tools->preferences->all->input/codecs->demuxers->RTP->RTP de-jitter buffer length



## Preferences

- + Advanced
- + Audio
- Input / Codec
  - + Access modules
  - + Audio codecs
  - Demuxers
    - AVI
    - Avformat
    - DV
    - Dirac
    - Dump
    - H264
    - Lua Art
    - MJPEG
    - MOD
    - MPEG-4 V
    - MPEG-TS
    - Matroska
    - PS
    - Playlist
    - RTP
    - RTP/RTSP

### Real-Time Protocol (RTP) input

RTP de-jitter buffer length (msec)

100

RTCP (local) port

0

SRTP key (hexadecimal)

[ ]

SRTP salt (hexadecimal)

[ ]

Maximum RTP sources

1

RTP source timeout (sec)

5

Maximum RTP sequence number dropout

3000

Maximum RTP sequence number misordering

100

Show settings

Simple  All

第一步

Save

Cancel

# 如何降低延遲？

- 方法二：安裝輕量級的RTSP Server(live555)

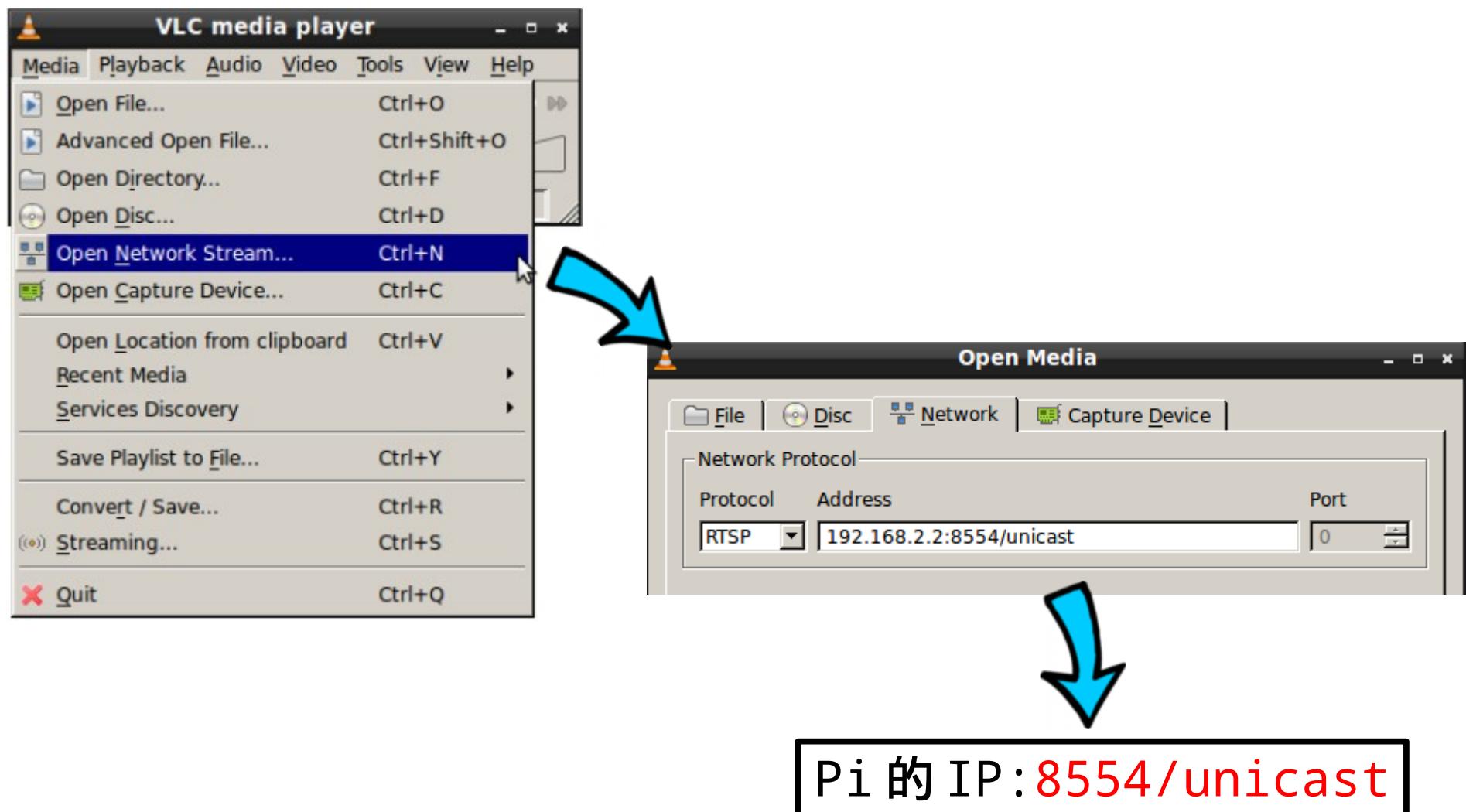
- \$ cd ~
- \$ git clone [https://github.com/mpromonet/h264\\_v4l2\\_rtspserver.git](https://github.com/mpromonet/h264_v4l2_rtspserver.git)
- \$ sudo apt-get install liblivemedia-dev libv4l-dev cmake libpthread-stubs0-dev
- \$ cd h264\_v4l2\_rtspserver
- \$ cmake .
- \$ make -j4
- \$ sudo ./h264\_v4l2\_rtspserver -F 15 -W 320 -H 240 -P 8554 /dev/video0

```
Play this stream using the URL "rtsp://192.168.1.130:8554/unicast"
2017-08-17 04:55:31,592 [NOTICE] - /home/pi/h264_v4l2_rtspserver/src/H264_V4l2De
viceSource.cpp:63
 profile-level-id=640028;sprop-parameter-sets=J2QAKKwrQGQJvywDxImo,K04fLA
==
```

<https://hpcc.uk/discussion/30/making-a-raspberry-pi-hd-camera>

# RTSP Client

- 在 Windows 使用 VLC

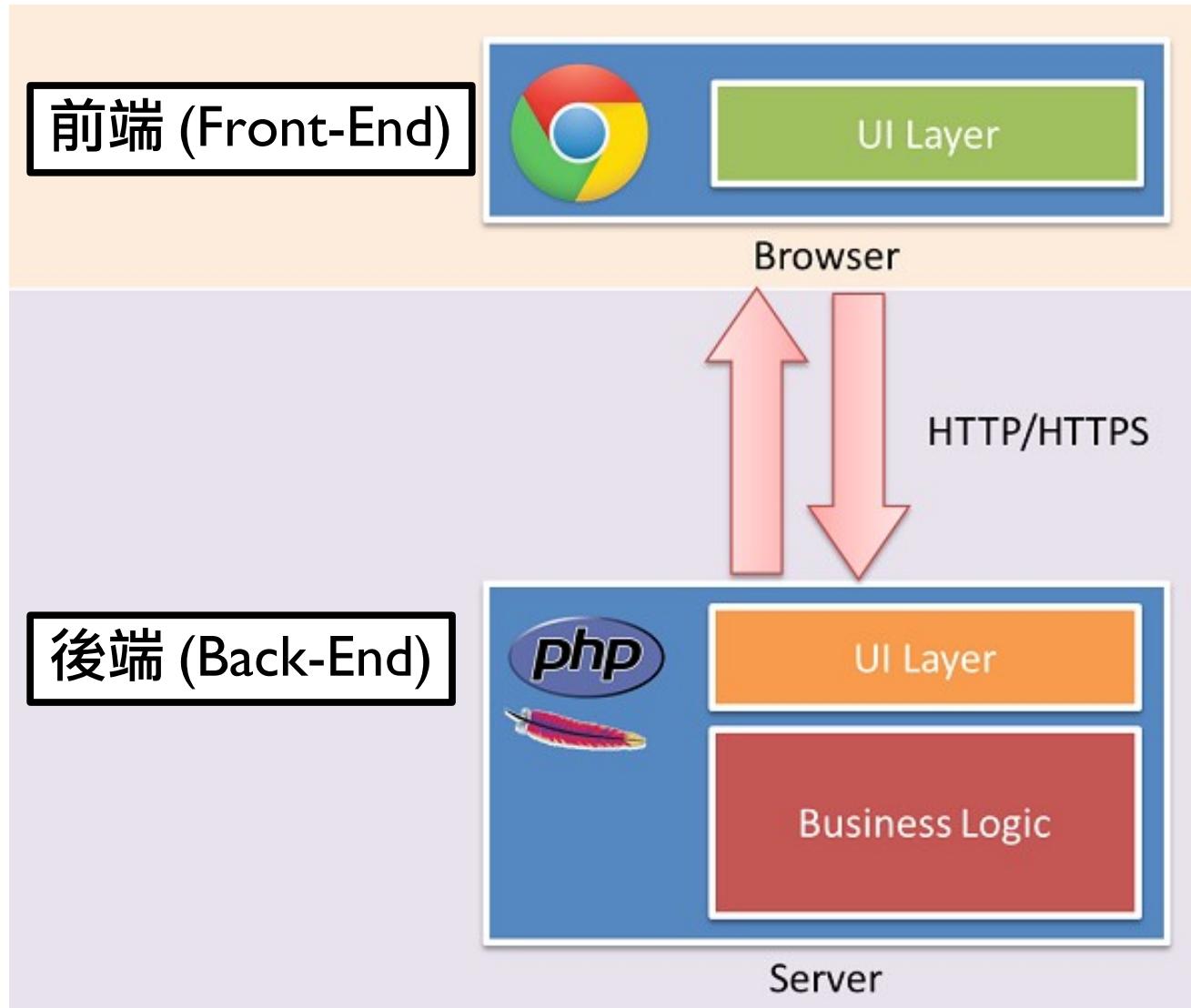


# 使用 HTTP + MJPG

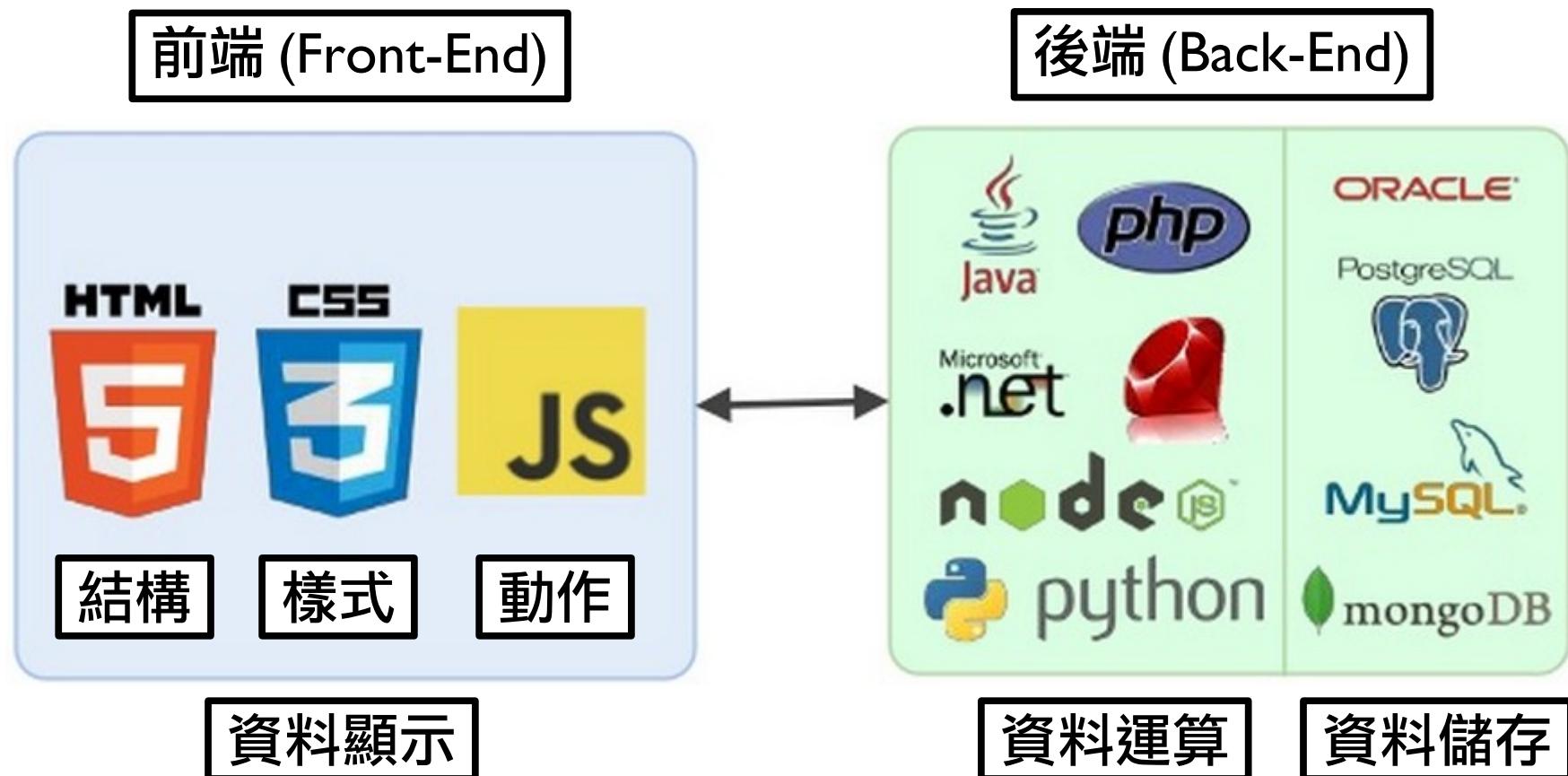
# 使用 HTTP 做 Video Streaming

- 原理
  - 向 Web Server 請求一個很大的檔案
  - 該檔案是一個即時的資料

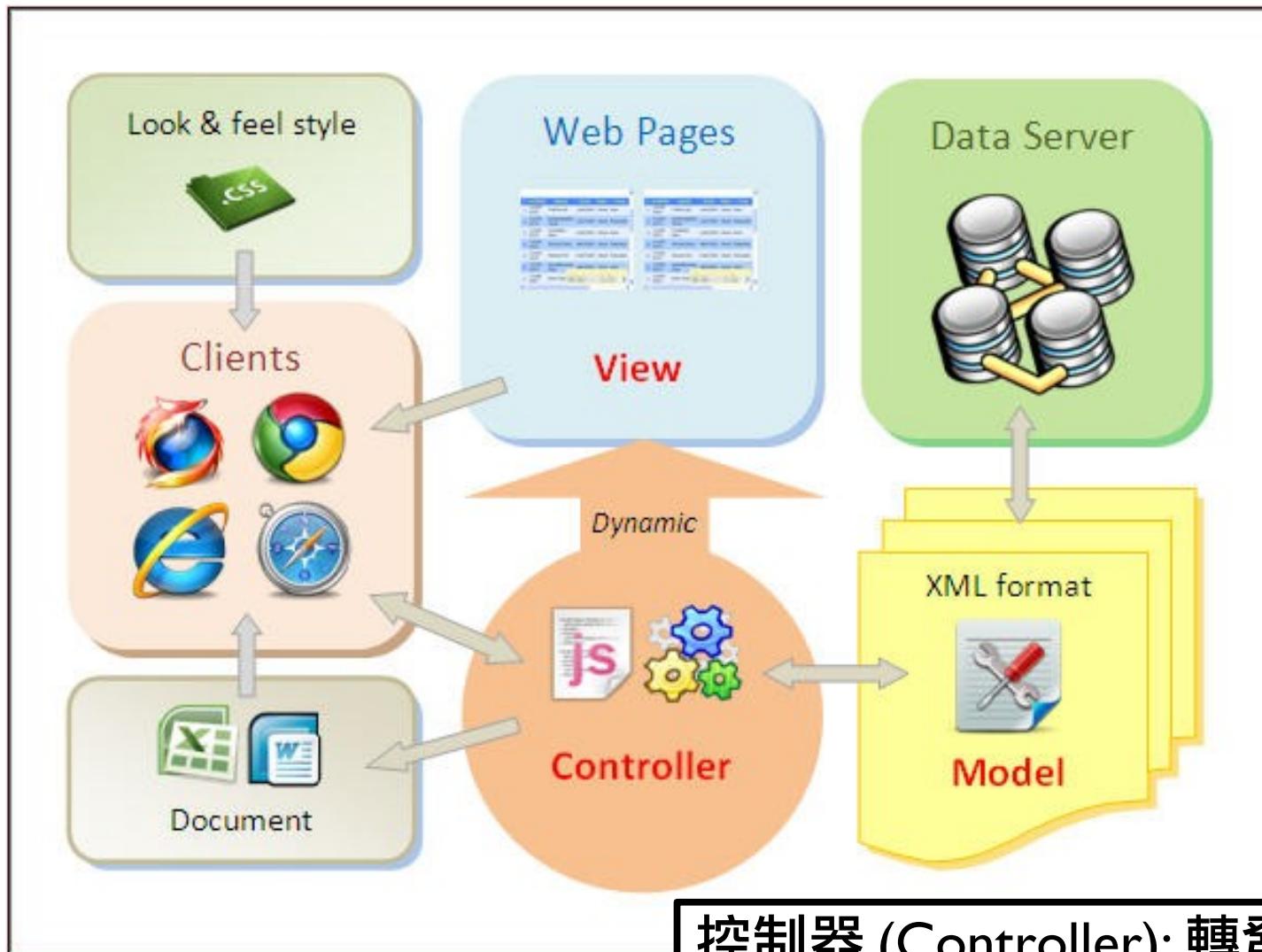
# 網頁的運作原理



# 可拆解成不同的實做



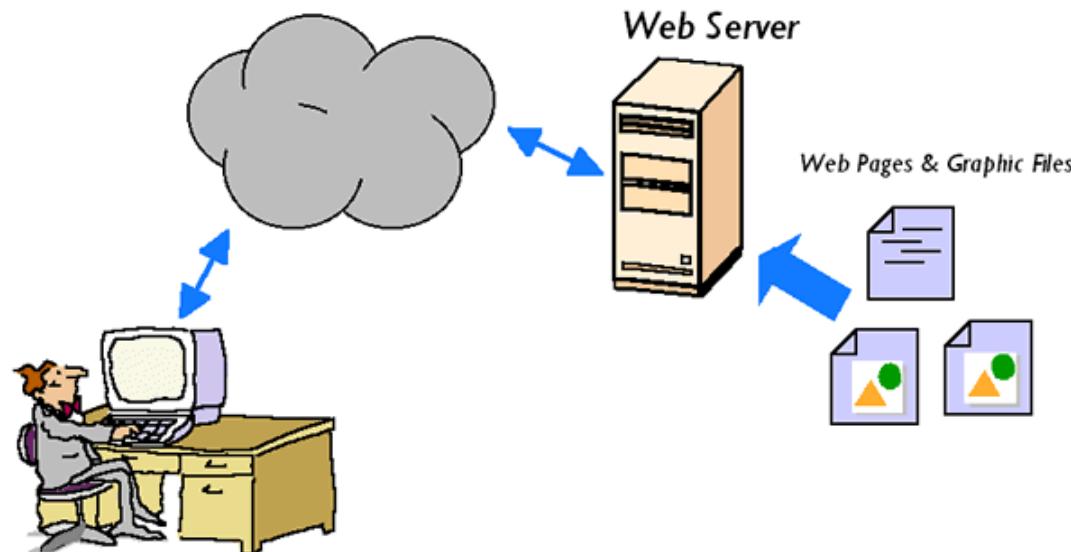
# Model-View-Controller 設計模式



控制器 (Controller): 轉發請求  
視圖 (View): 圖形界面顯示  
模型 (Model): 實現邏輯與資料儲存

# 網頁伺服器 (Web Server)

- 是一個軟體
- 回應從 80/8080 port 進來的 HTTP 要求
- 可透過 CGI 或 module 方式擴充
- 如 Apache, Nginx, Boa





- A Python Microframework

# app-hello.py

```
from flask import Flask
app = Flask(__name__)

@app.route("/") ← Controller
def index():
 return "Hello Flask" ← View

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

# 執行

```
$ sudo python app-hello.py
```

```
* Running on http://0.0.0.0:80/
* Restarting with reloader
192.168.2.1 - - [07/May/2017 15:04:39] "GET / HTTP/1.1" 200 -
192.168.2.1 - - [07/May/2017 15:04:39] "GET /favicon.ico
HTTP/1.1" 404 -
```



# DEMO

## app-hello.py

```
$ cd ~/cam-py-cv/day1/05-streaming
$ sudo python app-hello.py
```

# hello 使用到的檔案

- app-hello.py

# 樣板引擎 (Template Engine)

# 動態網頁

Yahoo!奇摩股市 <https://tw.stock.yahoo.com/q/q?s=2454>

是新用戶嗎？請註冊 | 登入 | 服務說明 設 Yahoo 奇摩為首頁 信箱 | 拍賣 | Yahoo 奇摩

**YAHOO! 股市 奇摩**

搜尋 網頁搜尋

新品首發 小米盒子 NT\$ 1999 1元瘋狂秒殺 小米手環2 NT\$ 865

首頁 投資組合 當日行情 大盤 類股 期權 港滙深股 美股 新聞 Y 選股

股票代號/名稱 2454 --當日個股股價-- 檢查 2016/11/08 00:22 距離美股收盤還有4小時38分鐘 更新

凱基客戶專區：委託成交 庫存報價 資料日期: 105/11/07

| 股票代號              | 時間    | 成交           | 買進    | 賣出    | 漲跌   | 張數    | 昨收    | 開盤    | 最高    | 最低    | 個股資料                           |
|-------------------|-------|--------------|-------|-------|------|-------|-------|-------|-------|-------|--------------------------------|
| 2454聯發科<br>加到投資組合 | 14:30 | <b>231.0</b> | 231.0 | 231.5 | △5.0 | 3,873 | 226.0 | 228.0 | 232.0 | 227.5 | 成交明細<br>技術 新聞<br>基本 等碼<br>個股健診 |

凱基證券下單  買  賣  張  零股交易

個股新聞與研究報告 » 更多

- 台股漲121.69點 收9189.84點 (中央社 2016/11/07 19:46)
- 聯發科10月營收著涼 月減1成創半年來新低 (網)

<https://tw.stock.yahoo.com>

爭鮮 SUSHI EXPRESS 美味蟹逅 濃香【鮮】台勘壽司

# 動態網頁

Yahoo!奇摩股市 <https://tw.stock.yahoo.com/q/q?s=2498>

是新用戶嗎？請註冊 | 登入 | 服務說明 Download the new Yahoo Mail app 信箱 | 拍賣 | Yahoo 奇摩

**YAHOO! 股市** 搜尋 網頁搜尋

**DELIVERING CONNECTED COMMERCE WITH EVERY STROKE** LEARN MORE DIEBOLD NIXDORF DieboldNixdorf.com

首頁 投資組合 當日行情 大盤 類股 期權 港滬深股 美股 新聞 Y 選股

股票代號/名稱 2498 --當日個股股價-- 檢查 2016/11/08 00:23 距離美股收盤還有4小時37分鐘 更新

凱基客戶專區：委託成交 庫存報價 資料日期: 105/11/07

| 股票代號              | 時間    | 成交   | 買進   | 賣出   | 漲跌   | 張數    | 昨收   | 開盤   | 最高   | 最低   | 個股資料                                                                                                                          |
|-------------------|-------|------|------|------|------|-------|------|------|------|------|-------------------------------------------------------------------------------------------------------------------------------|
| 2498宏達電<br>加到投資組合 | 14:30 | 84.4 | 84.4 | 84.5 | △1.1 | 9,475 | 85.5 | 84.4 | 84.9 | 83.2 | <a href="#">成明細</a><br><a href="#">技術</a> <a href="#">新聞</a><br><a href="#">基本</a> <a href="#">籌碼</a><br><a href="#">個股健診</a> |

凱基證券下單  買  賣 張 送出 零股交易

個股新聞與研究報告 » 更多

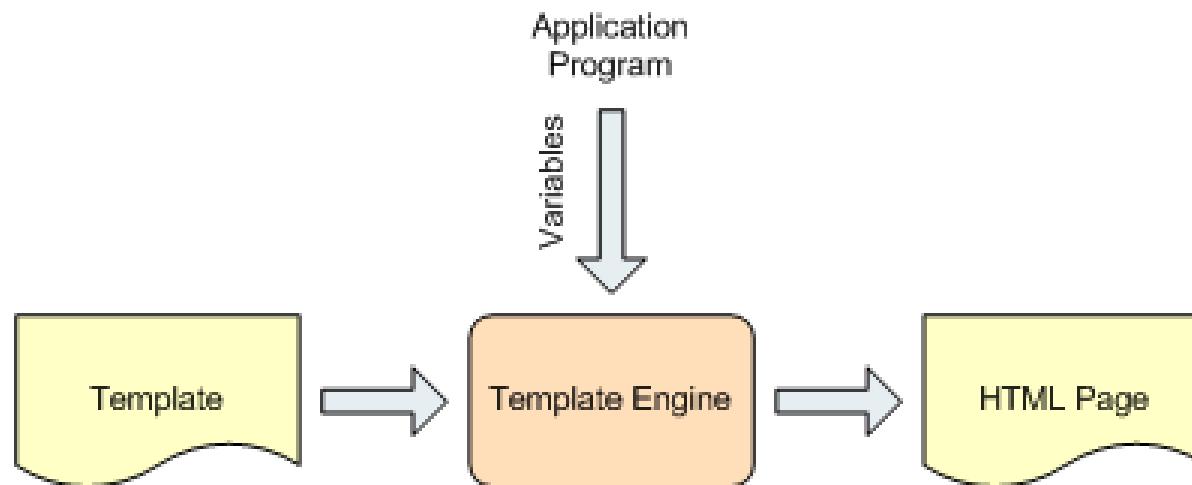
- 台股漲121.69點 收9189.84點 (中央社 2016/11/07 19:46)
- 亞洲投顧 股期龍哥 盤後評述 (東網 2016/11/07 19:46)

New AS6200 temperature sensor  
delivers accuracy and low power consumption in a small package.  
Available now!

<https://tw.stock.yahoo.com>

# 動態網頁

- 是由靜態 HTML 加上動態文字產生
- 樣板引擎將程式碼與使用者介面分離



- Flask 預設使用 Jinja 做為樣板引擎

# 新增一個 route & template

```
from flask import Flask, render_template

app = Flask(__name__)

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

@app.route("/foo")
def foo():
 extns = ['Flask', 'Jinja2', 'Awesome']
 return render_template('bar.html', extns=extns)

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

View 可以由樣板產生

樣板可以塞變數

# 建立 template

```
$ mkdir templates
$ nano templates/link.html
<h1>Hello Template</h1>
foo
```

```
$ nano templates/bar.html
```

```

 {% for ext in exts %}
 {{ ext }}
 {% endfor %}

```

# 執行

```
$ sudo python app-route.py
```



# DEMO

## app-route.py

```
$ cd ~/cam-py-cv/day1/05-streaming
$ sudo python app-route.py
```

# route( 樣板 ) 使用到的檔案

- app-route.py
- templates
  - └ bar.html (/foo)
  - └ link.html (/)

# Streaming 圖片 (app-stream.py)

Camera 類別

```
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')
```

回傳內容

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

回傳型態

# 建立 Camera 類別 (stream\_pi.py)

```
from time import time

class Camera(object):
 def __init__(self):
 self.frames = [open(f + '.jpg', 'rb').read()
for f in ['1', '2', '3']]

 def get_frame(self):
 return self.frames[int(time()) % 3]
```

# 修改 template

```
$ nano templates/stream.html
```

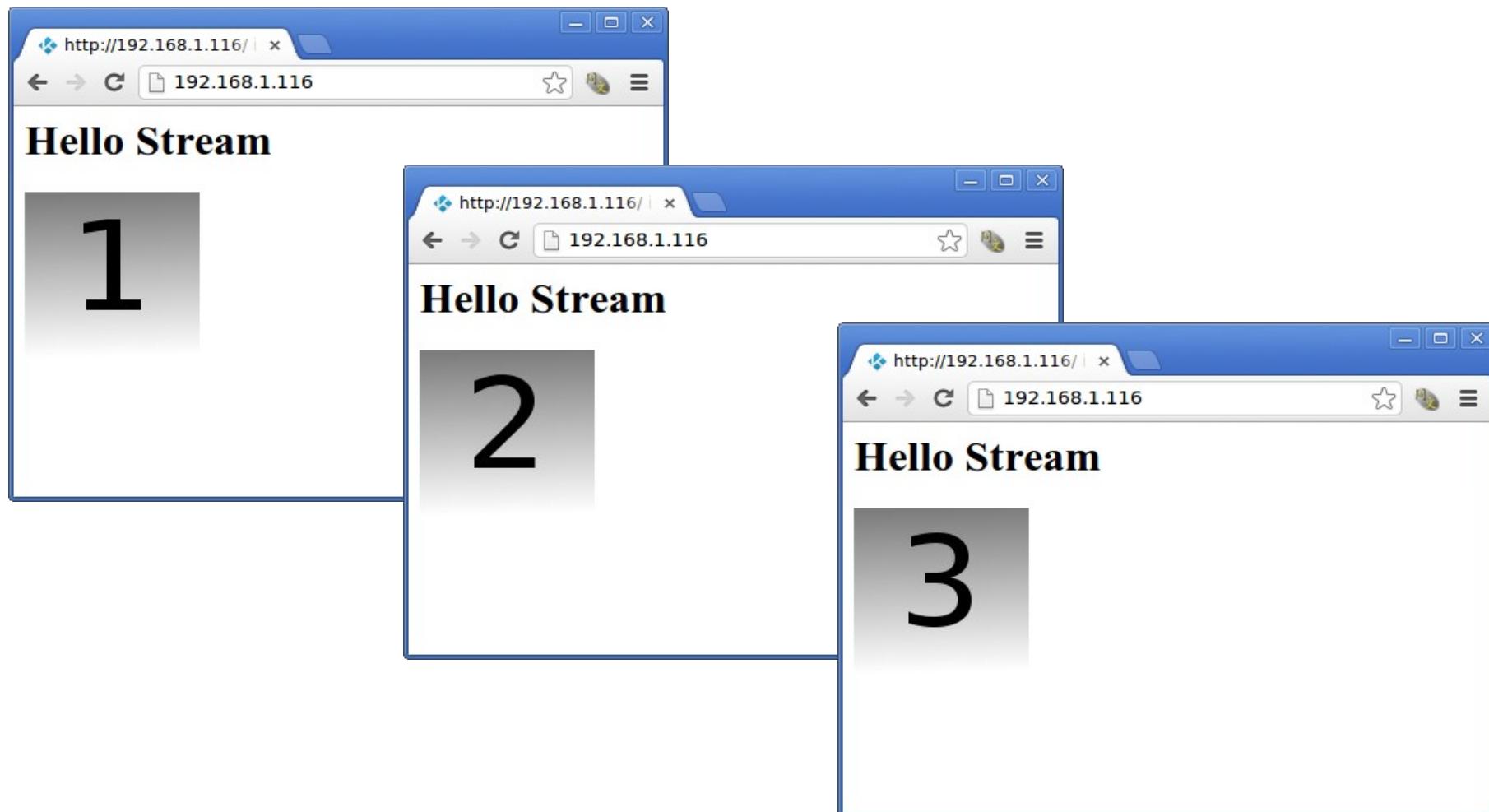
```
<h1>Hello Stream</h1>

```

- 補個圖 1.jpg, 2.jpg, 3.jpg

# 執行

```
$ sudo python app-stream.py
```



# DEMO

## app-stream.py

```
$ cd ~/cam-py-cv/day1/05-streaming
$ sudo python app-stream.py
```

# stream 使用到的檔案

- app-stream.py
- stream\_pi.py (class Camera)
- templates
  - └ stream.html (/video\_feed)
  - └ link.html (/)
- static
  - └ 1.jpg
  - └ 2.jpg
  - └ 3.jpg

# HTTP + MJPEG

- 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

# 從 Camera 讀取影像 (camera\_pi.py)

```
import cv2

class Camera(object):
 def __init__(self):
 self.video = cv2.VideoCapture(0)

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

 def get_frame(self):
 success, image = self.video.read()
 image = imutils.resize(image, 640)
 ret, jpeg = cv2.imencode('.jpg', image)
 return jpeg.tostring()
```

開啟 /dev/videoX

V4L2 API

使用前記得要先載入模組  
\$ sudo modprobe bcm2835-v4l2



不是數字 1，是小寫 L

## DEMO

### app-camera.py

```
$ cd ~/cam-py-cv/day1/05-streaming
$ sudo python app-camera.py
```

# camera 使用到的檔案

- app-camera.py
- camera\_pi.py
- templates
  - └ stream.html
  - └ link.html

# 如何降低延遲？(camera\_pi.py)

```
import cv2

class Camera(object):
 def __init__(self):
 self.video = cv2.VideoCapture(0)

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

 def get_frame(self):
 success, image = self.video.read()
 image = imutils.resize(image, 320)
 ret, jpeg = cv2.imencode('.jpg', image, [1, 50])
 return jpeg.tostring()
```

縮小長寬

調整 JPG 壓縮品質

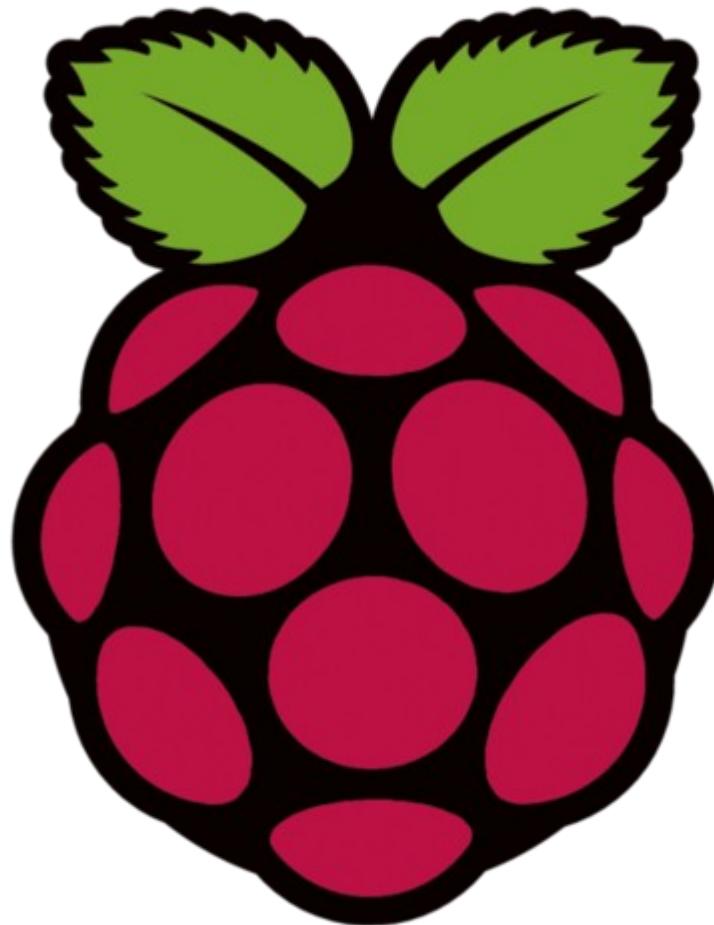
# 練習

- 使用 OpenCV 套件所提供的函式，讓 Camera 的 Streaming 顯示日期

# 小結

- 串流服務可使用 RTSP+H.264 或是 HTTP+MJPG 等多種組合
- 使用自己寫的 HTTP 串流，可以在 JPG 丟出去前再進行處理
- 如果要多人連線，可以實做 videocopy 或是在服務前面新增一個 host 儲存目前影像結果，新連線連到 host 後直接回傳儲存結果

# Raspberry Pi Rocks the World



Thanks

補充

# 可見光影像可以進行各種辨識



<https://www.youtube.com/watch?v=aElkA0Jy0Xg>

# 但會受到天候或是光線影響



(a)



(b)



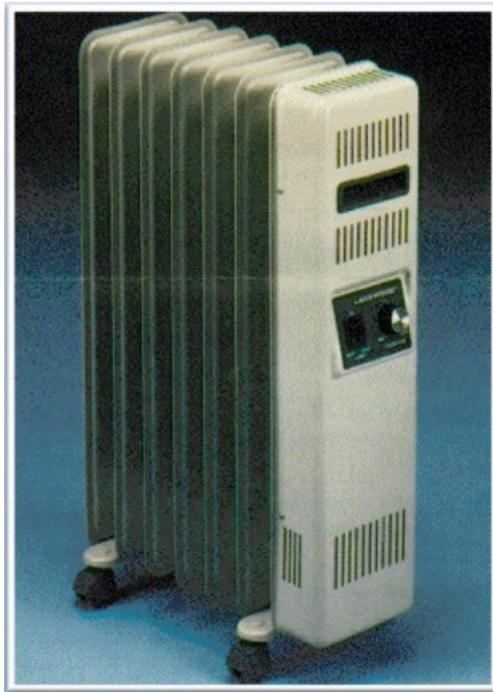
(c)



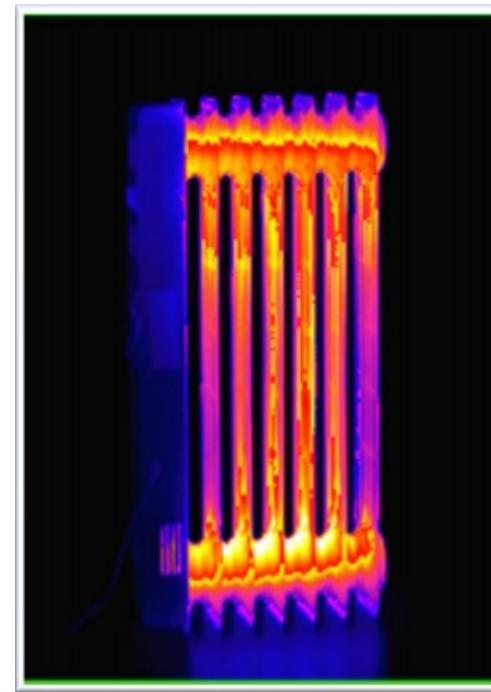
(d)

# 為什麼需要紅外線影像？

- 我們需要外部的光源來讓我們”看到”
- 紅外線影像讓我們對溫度有”第六感”



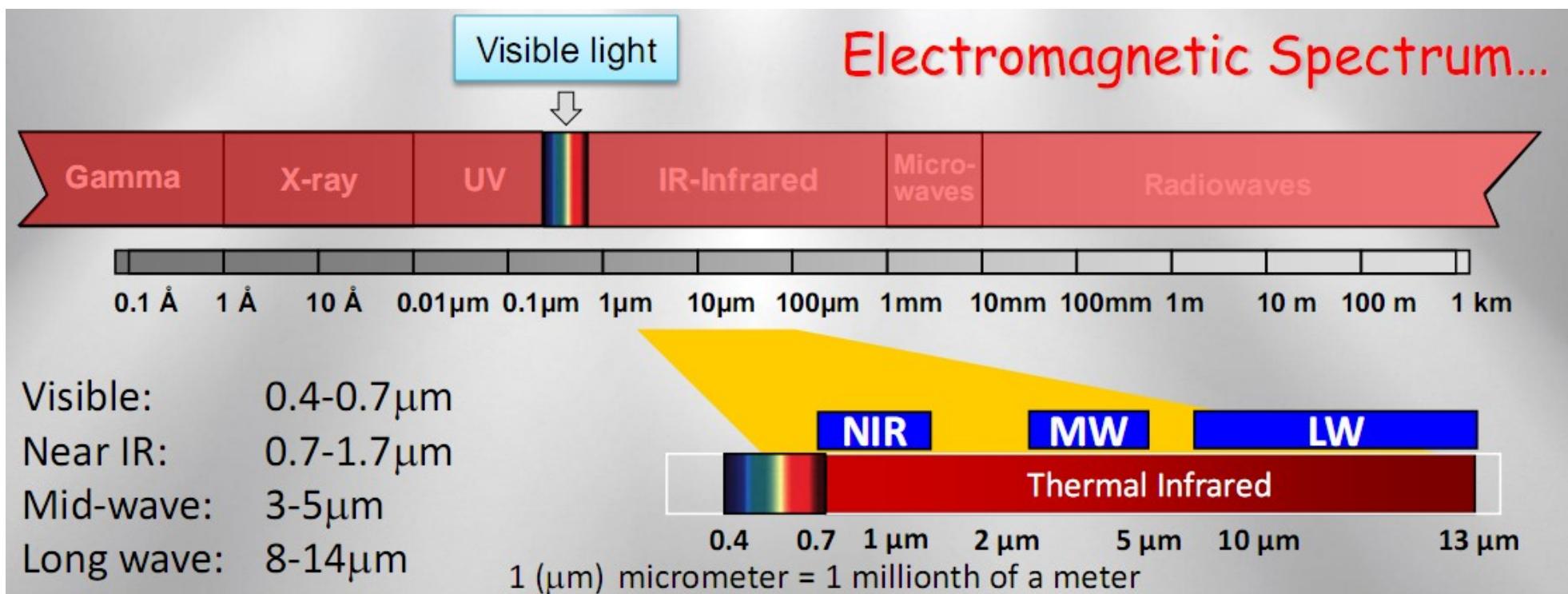
可見光影像



紅外線影像

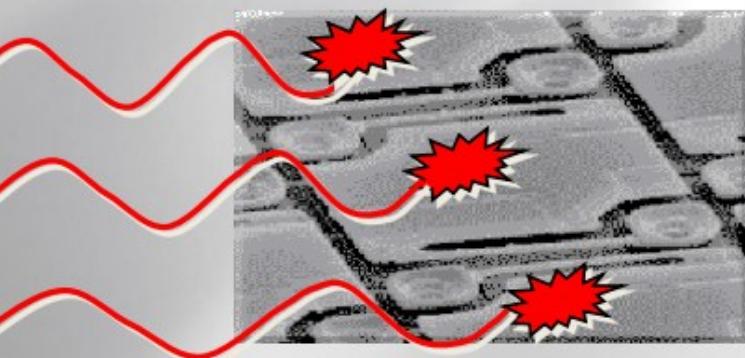
# 熱輻射 (Infrared Radiation)

- 热的傳播是靠傳導、對流和輻射（不需介質）
- 不受白天、晚上、煙霧、下雨等天候影響
- 热輻射是電磁波的一種（波長從 700nm 到 1mm）



# 熱紅外線成像原理

## How IR Cameras Work

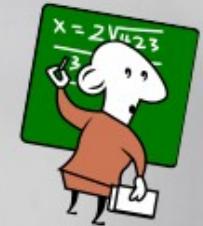


Photons Impact and  
HEAT the Detector Element.



Stefan - Boltzmann Law

$$E = \varepsilon\sigma T^4$$



Advanced Calculations

Temperature Values are calculated



22	39	44	96	98	77
23	13	35	55	78	75
77	65	45	44	34	25
56	76	90	12	34	67
85	45	77	89	44	23
25	48	13	46	75	89

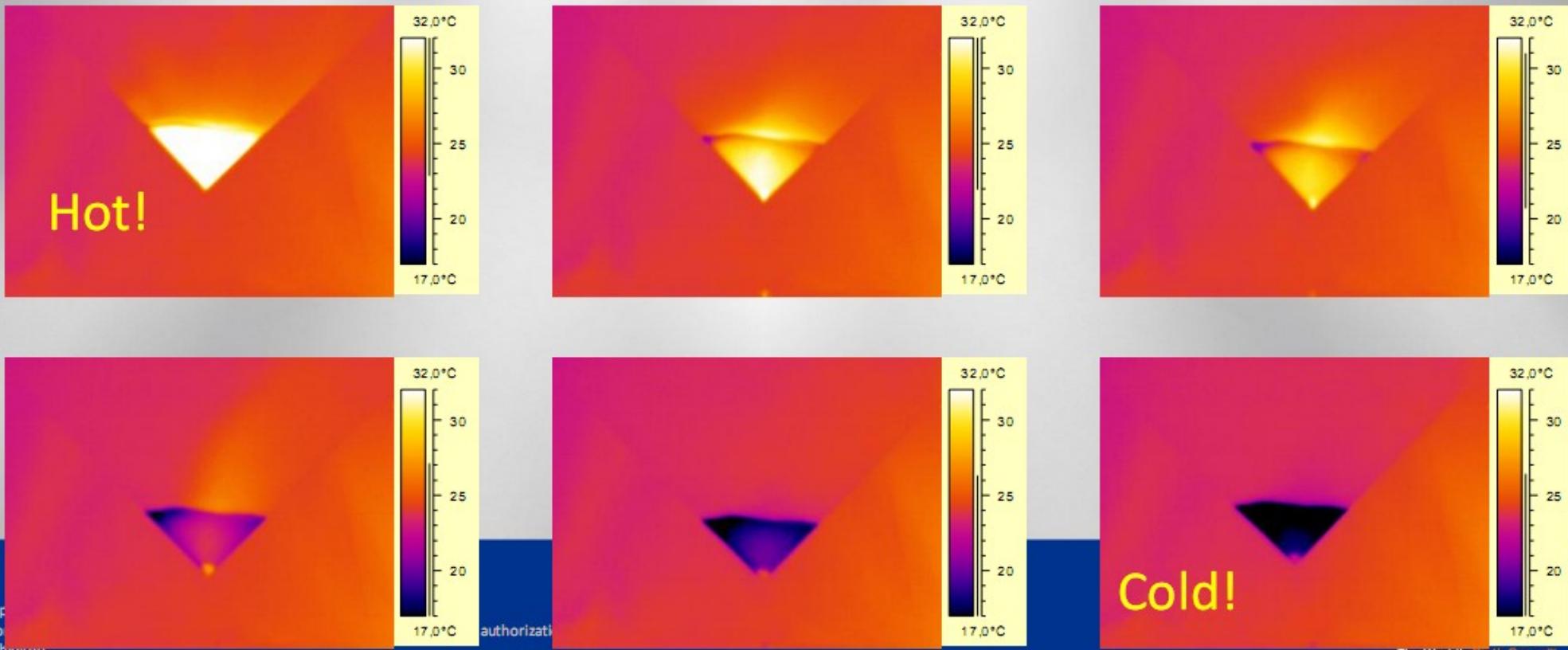
Colorized based on Temp Value



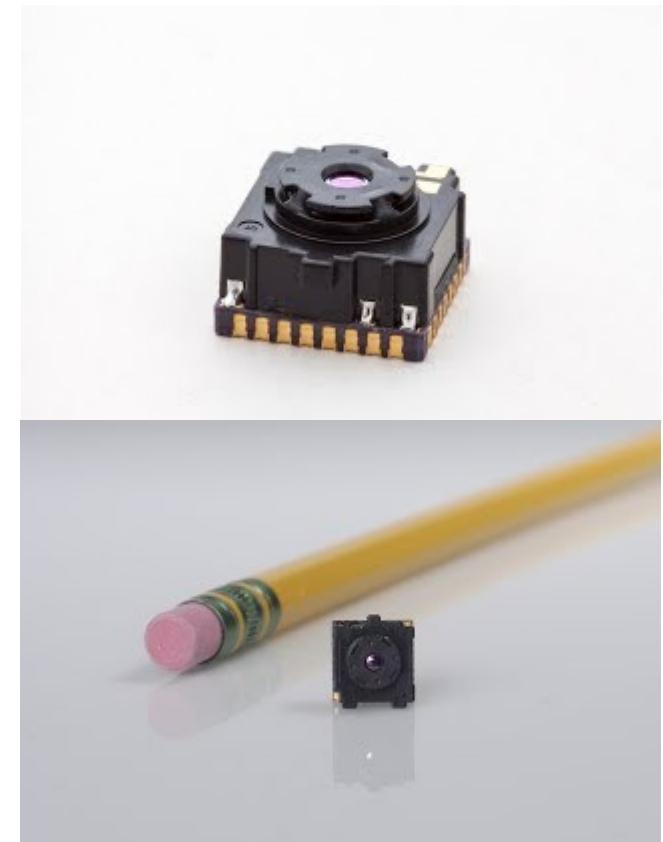
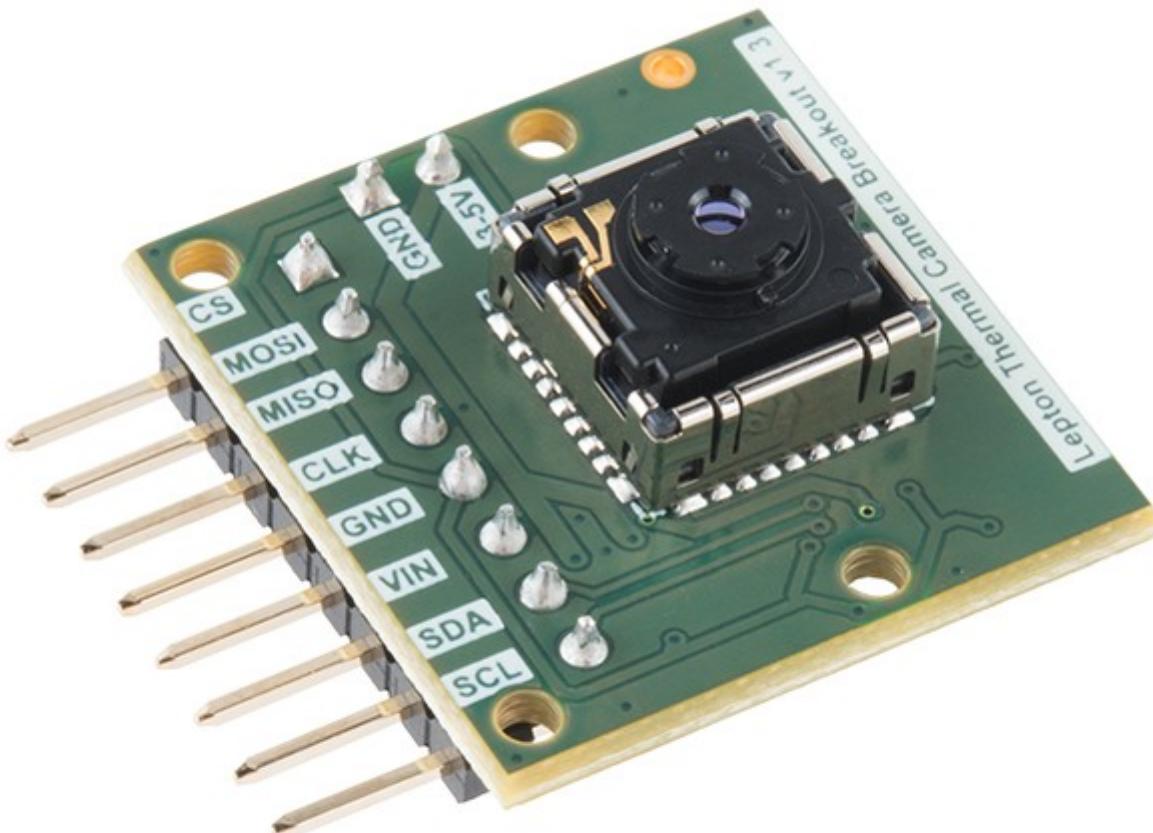
# 溫度可視化

## IR Theory - Evaporation

*Example: Wet a piece of paper with warm water and watch the changes*

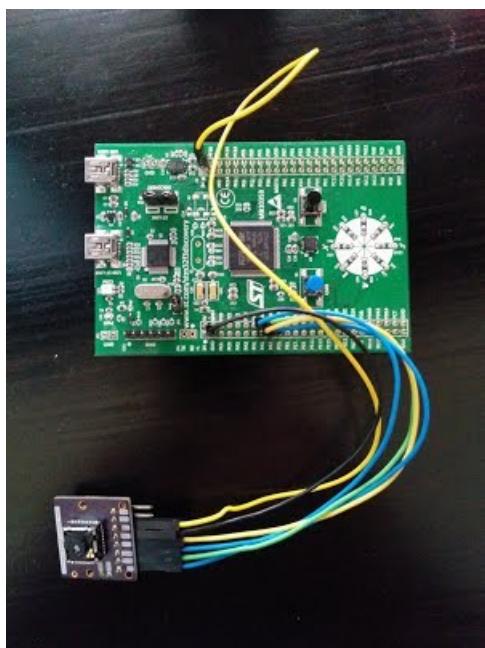
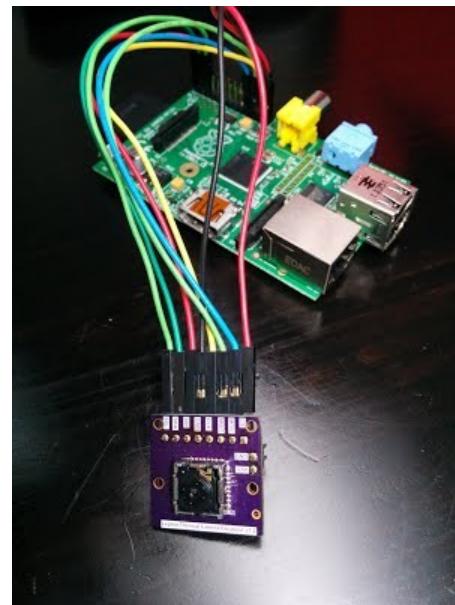
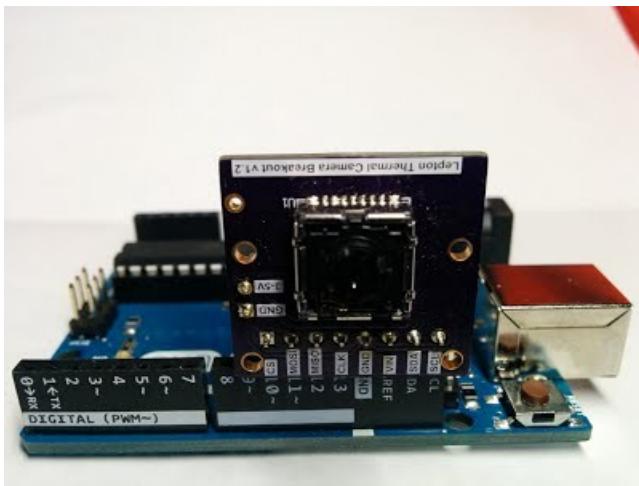


# FLIR Lepton LWIR Camera



<https://www.sparkfun.com/products/13233>  
<http://www.pureengineering.com/projects/lepton>

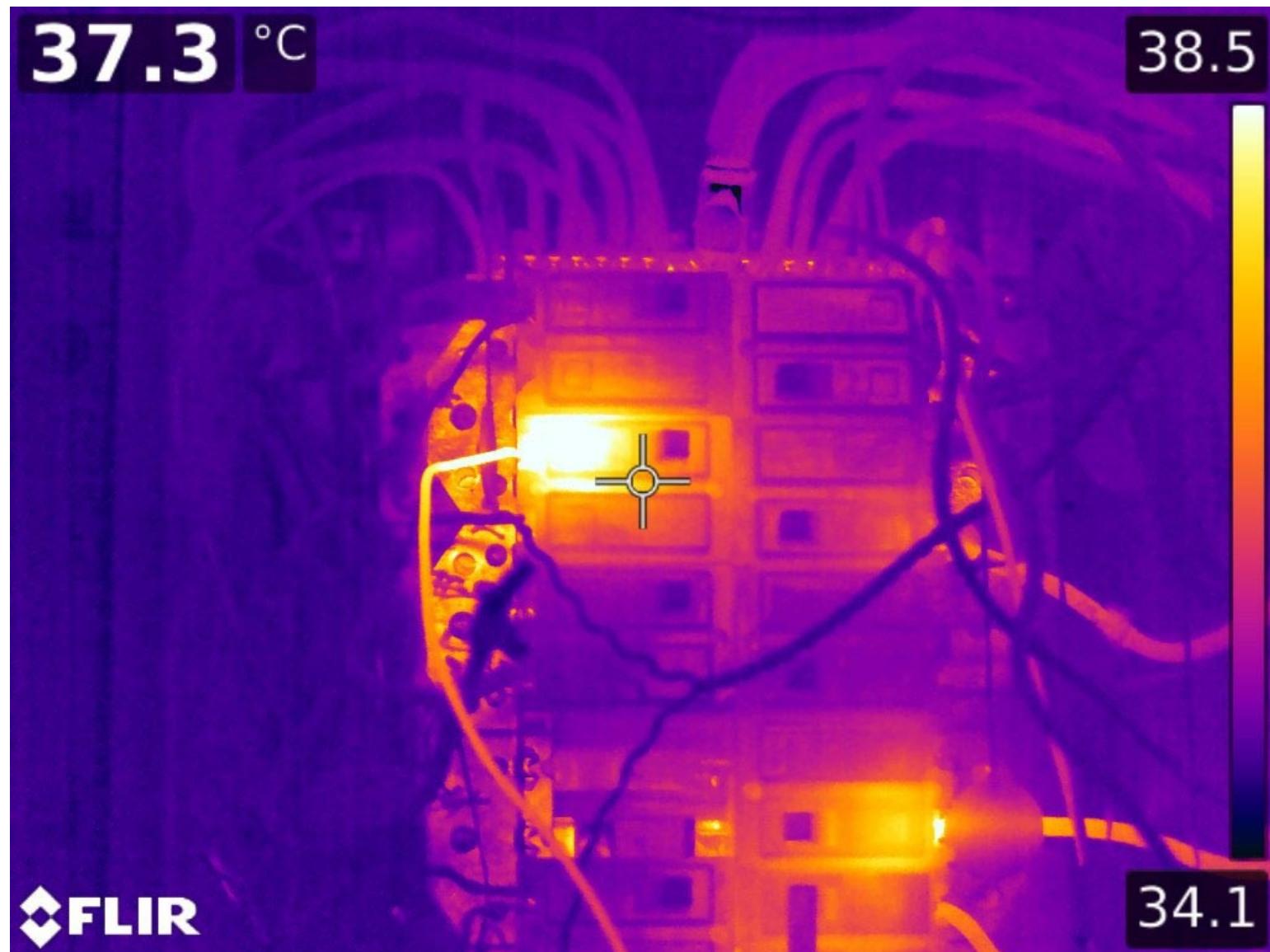
# 適用多種平台



# 規格

- LWIR sensor, wavelength 8 to 14  $\mu\text{m}$
- 51-deg HFOV, 63.5-deg diagonal
- 80(h)  $\times$  60(v) active pixels
- Thermal sensitivity <50 mK
- MIPI and SPI video interfaces
- Two-wire I2C-like serial-control interface
- Fast time to image (< 0.5 sec)
- Low operating power, nominally 150 mW

# 應用：故障分析

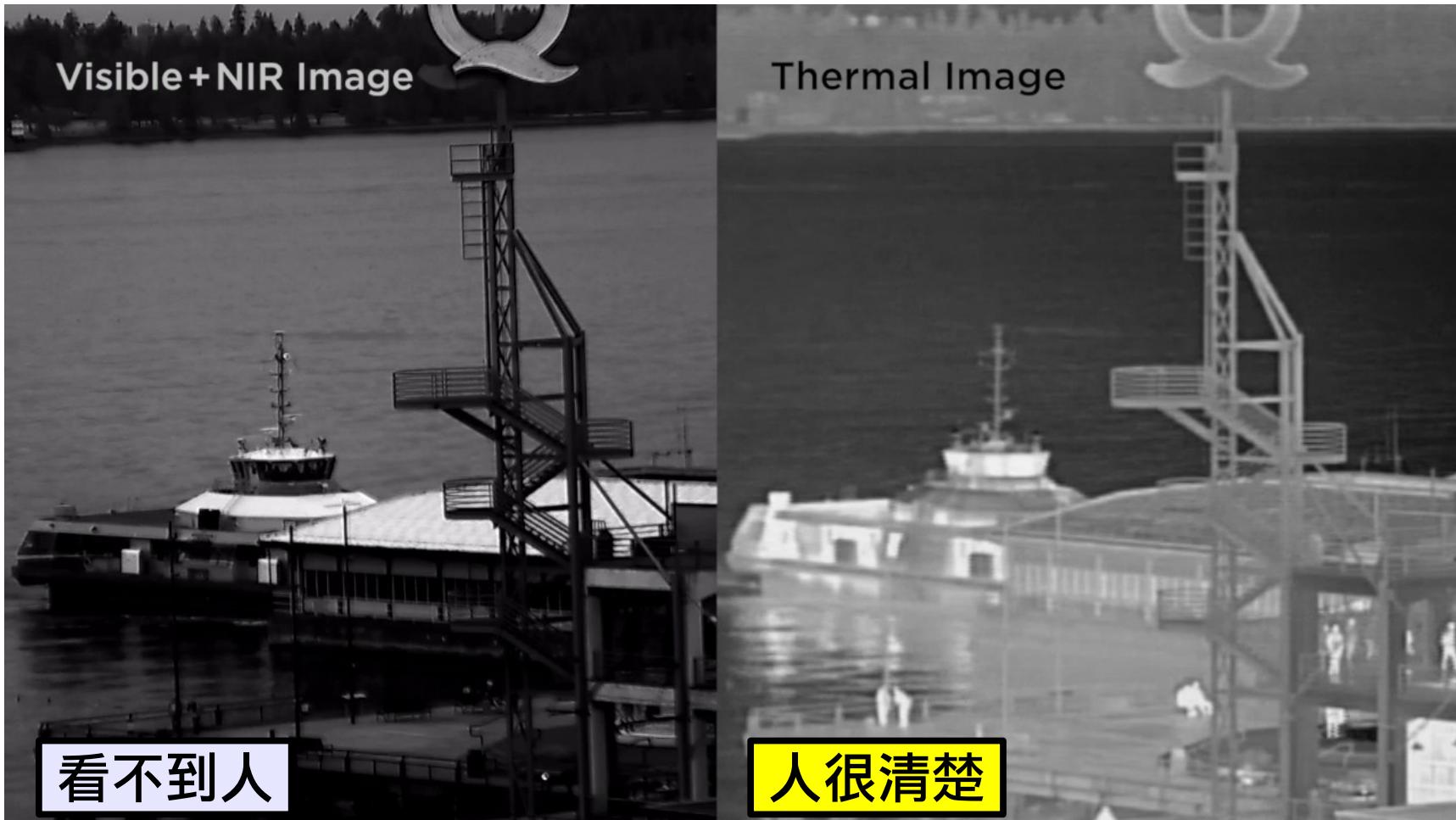


FLIR

<http://www.flir.com/instruments/content/?id=70502>

# 應用：安防監控

- 在完全無光的場景也能清楚顯示物體



但我們需要大量的影像資料做訓練  
才能做自動化辨識

# 光學影像已經有大資料庫

## ImageNet Dataset



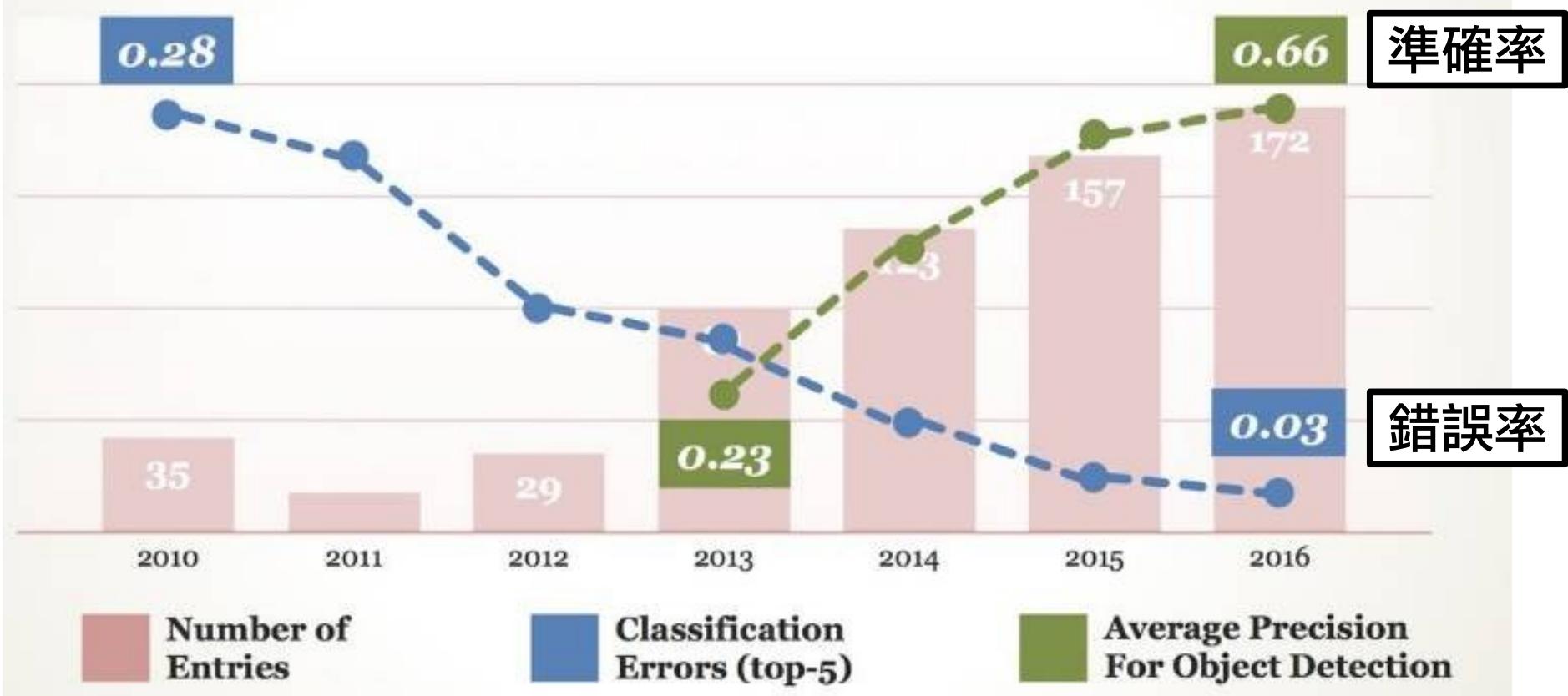
IMAGENET

超過 1,500 萬張

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). [Imagenet large scale visual recognition challenge](#). *arXiv preprint arXiv:1409.0575*. [\[web\]](#)

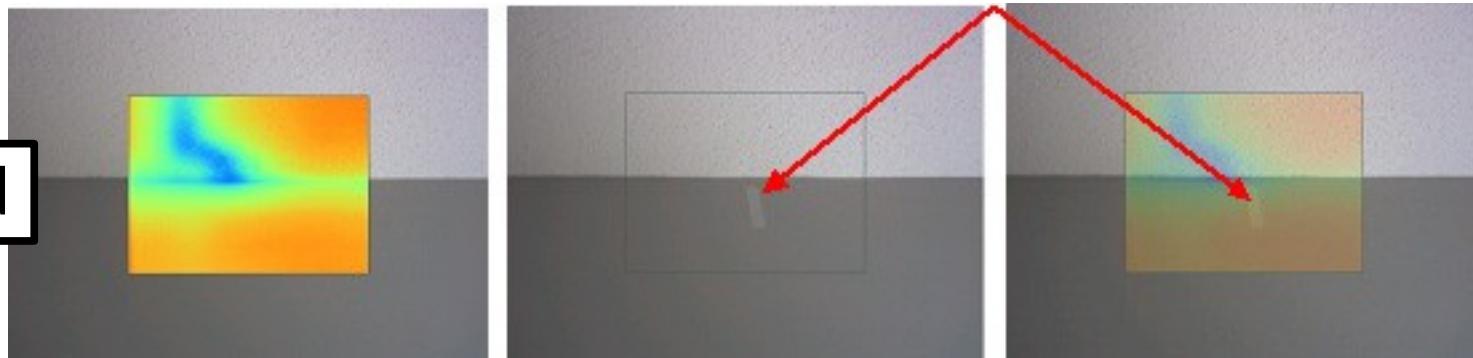
# 數據驅動機器學習的準確率提高

## Participation and Performance

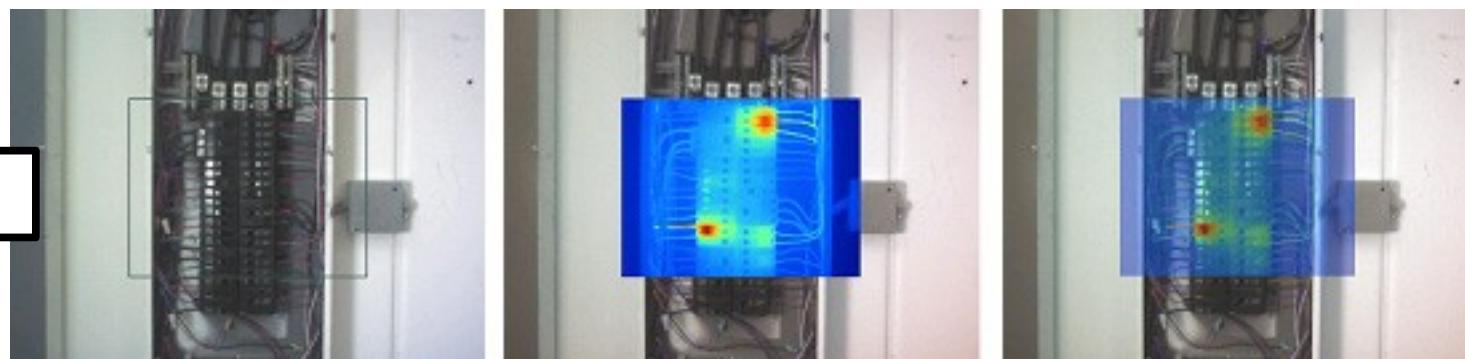


# 結合可見光影像與熱紅外線影像

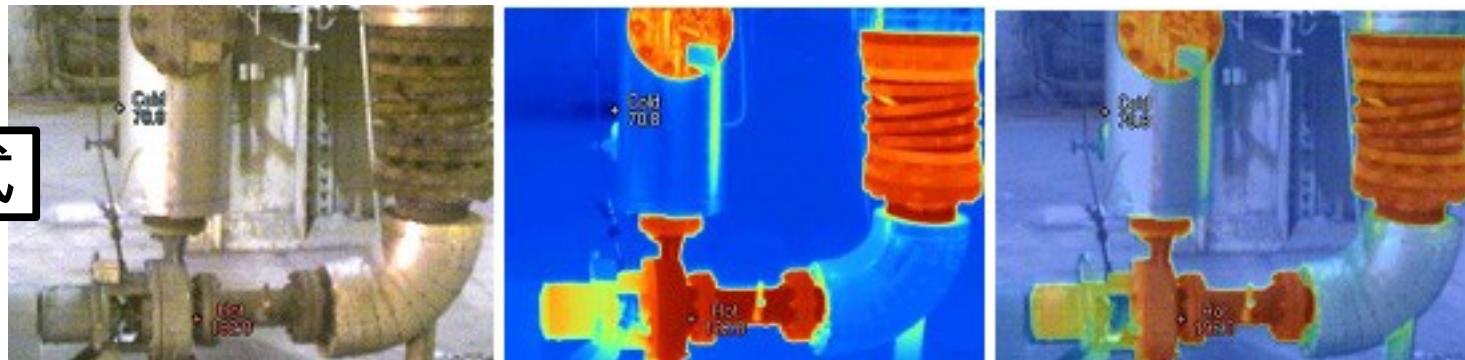
blended



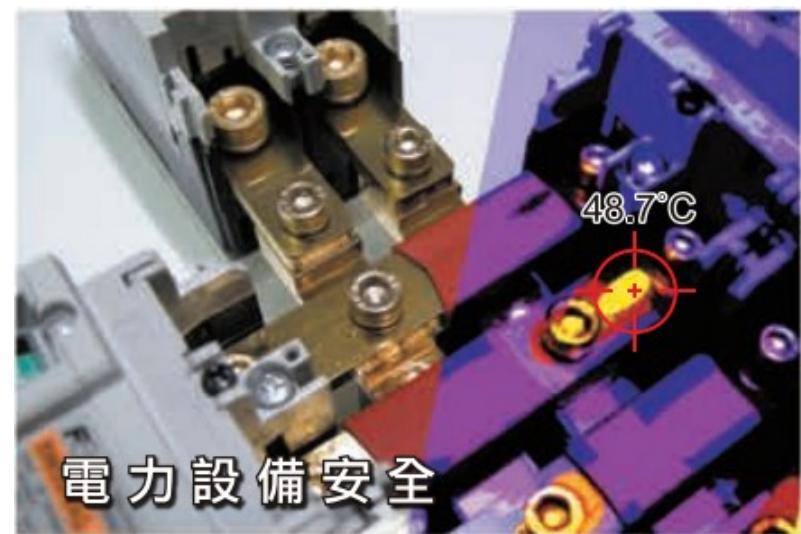
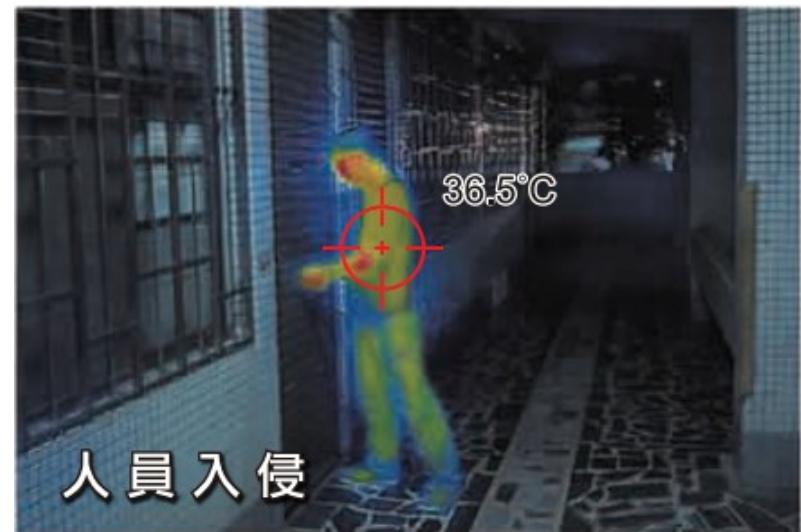
子母畫面



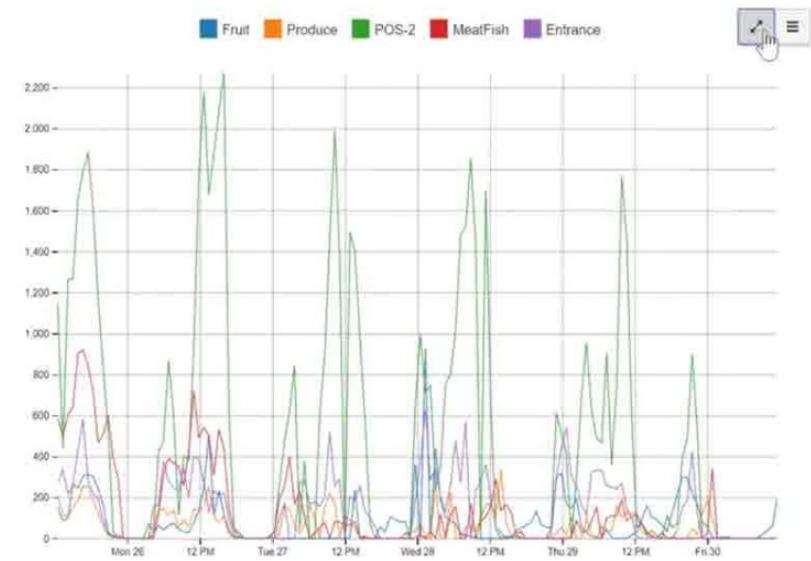
全螢幕模式



# 樹莓派 x 热物聯網



# 熱區分佈 + 客流分析



# 更多：串接 picamera 到 OpenCV

```
import picamera
from picamera.array import PiRGBArray
import time
import cv2

camera = picamera.PiCamera()
camera.resolution = (320, 240)
rawCapture = PiRGBArray(camera)
stream = camera.capture_continuous(rawCapture,
format="bgr",use_video_port=True)

for f in stream:
 frame = f.array
 rawCapture.truncate(0)
 cv2.imshow("preview", frame)
 if cv2.waitKey(1) & 0xFF == ord("q"):
 break
```