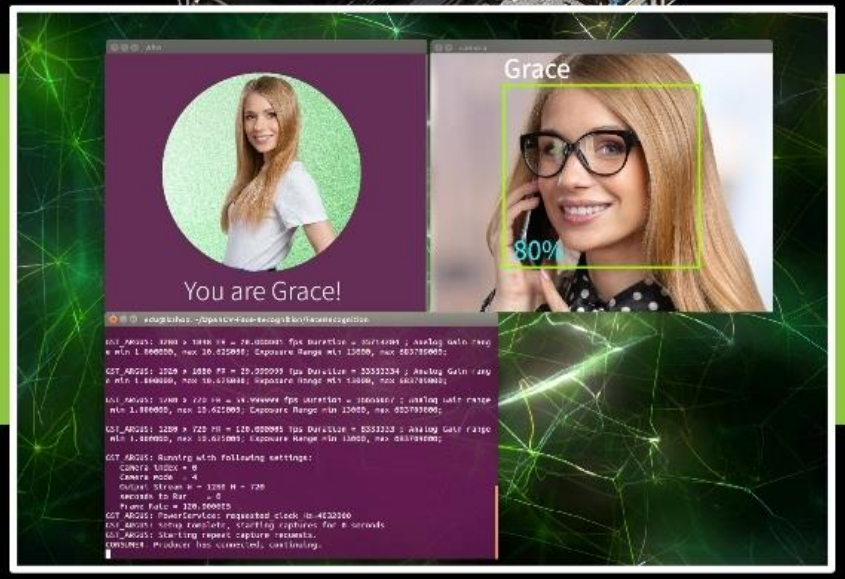
創客萊吧
Maker Lab

# NVIDIA Jeston Nano
## 影像辨識工作坊
### 探索強大影像辨識功能




## 1.專業網購

- IC電子零件
- 工具儀錶
- 微控制器開發板
- 感測器模組

## 2.教學設備

- ROS 1.0/2.0 機器人應用平台
- 智慧機器人移動平台
- DOBOT視覺辨識機器人
- 樹莓派應用平台
- 技能/技藝競賽平台
- 兩足人型機器人
- 物聯網、智慧家庭
- 智慧農場

## 3.Maker軍火庫

- STEAM教育
- micro：bit 多元應用
- 國中生活科技
- 自造教育及科技中心規劃

## 4.創客萊吧 MakerLab

- 創客空間推廣營運
- 社群共學、共筆
- 空間導覽參訪
- 機具設備體驗實作

## 5.教學服務

- 共編教材
- 業界協同
- 研習培訓
- 產學合作
- 師徒共學

## 6.客製化專案

- AI機器人應用開發
- 樹莓派應用開發
- 教學設備應用開發

## 7.專業研發設計

- 量測設備研發
- IC電路設計
- 設備預警系統
- 資料雲端化
- 專案設計

## 8.海外代詢代購

- 專業技術規格諮詢
- 快、好、便宜 服務

# 講者介紹

喜歡學習與分享新科技的 **工程師 \ 設計師 \ 程式與機器人教育者**


## 林威志

專案工程師

github.com/kjoelovelife

IcShopping 淩耀電子

07 – 5564686 # 18
0931-912-173

joe@icshop.com.tw


## 工作經歷

**2015-2016 圓創力科技教育中心**
**機器人授課人員**
帶領學生參與 2015、2016 WRO世界機器人競賽

**2016-2017 Mzone 大港自造特區**
**廠長**
參與 第零屆大港自造節、設展 2016台北Maker Faire
培訓 20位以上實習生 、管理 Maker space 各項機台
研發 Maker 教案與商品、講授20場以上課程

**2017-2018 標準桿實業有限公司**
**技術工程師**

| | |
|---|---|
| 正修科技大學電子工程系新興科技研習 | 講師 |
| 東方設計大學遊戲與玩具設計科 | 講師 |
| ROS - 機器人操作系統 | 研究人員 |
| 人工智慧 Tensorflow與其應用 | 研究人員 |

**2018 -- 淩耀電子有限公司**
**教育事業部 - 專案工程師**

| | |
|---|---|
| ROS - 機器人操作系統 | 研究人員 |
| 人工智慧與其應用 | 研究人員 |
| 國立高雄科技大學 - 南科 Ai計畫 | 研究人員 |
| 創客萊吧、萊吧研究所 | 講師 |
| 高雄科技大學 ROS研習營 | 講師 |

## 技能

Python

ROS

Illustrator

fusion360

# 今日人臉辨識完成所需條件

1. 自行攜帶 Jetson-nano 開發套件組與相關配備 ，詳細規格請查看，購買連結。

2. MicroSD卡需事先燒錄完畢 ，詳細燒入步驟請查閱，連結點擊。

3. 請自備鍵盤、滑鼠組。
4. 請攜帶個人筆電，並已裝備好 Ubuntu 16.04 或 18.04 版本。
詳細安裝步驟請參考 ，連結點擊。

# 今日課程規劃

預計花費 : **3 小時**

| 時　間 | 內容 |
| --- | --- |
| 13:50 – 14:00 | 報到 |
| 14:00 – 15:00 | 如何建立 Jetson Nano 操作環境 |
| 15:00 – 16:00 | NVIDIA Jetson Nano使用心得分享 |
| 16:00 – 17:00 | 實作人臉辨識 |

** 時程將會依照課程進度而有所變更*

# 今日課程重點

為了方便地使用 Jetson –nano ： **系統環境的建立為主要目的**

## 設備完整性

良好的設備將節省開發時間，例如

- 電源的選擇
- 記憶體的配置
- 溫度的可控性
- 鏡頭
- 輪胎
- 馬達控制器
- 雷達

## 系統環境建置

不同的硬體架構，需配上不同的軟體、韌體與系統，才能夠正常地開發相關應用程式。

系統環境的完成度與否，將會影響到應用程式開發的時間。

## 應用程式開發

根據功能需求，決定所需開發的程式語言。

目前主流的程式語言有

- C
- C++
- Python
- Java
- Swift
- Julia

## 實際應用

根據前三項敘述方法，打造出自己心目中的功能應用。

# 如何建立 Jetson –nano 操作環境

# Jetson-nano developer kit

Nvidia 官方映像檔所搭載系統為 **Ubuntu 18.04 LTS 系統**


## 人工智慧的開源

人工製智慧的爆炸性發展，
始於 2012年的 ImageNet 競賽。
其後相關的學者一致認為，人工智慧的技術發展應具有開源的特性，以確保人工智慧相關技術能夠持續爆炸性地發展。

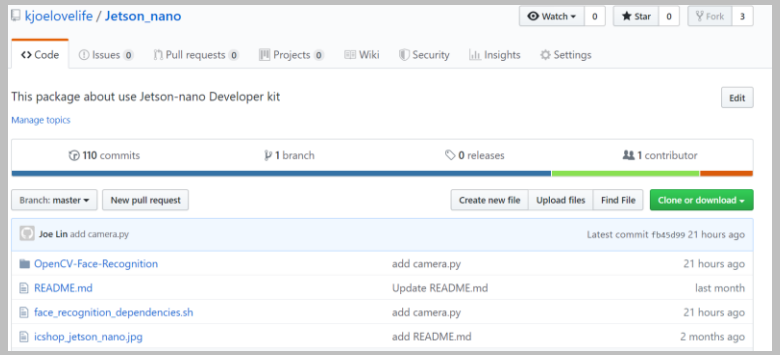
Linux 系統一開始便以開源的特性，開源給全世界使用者自行修改使用。
讓使用者能夠依據個人的喜好、講求的功能性等因素，打造獨特的系統。

結合上述兩個關鍵因素，「開源的人工智慧」與「開源的系統」，便使得 Nvidia 以 Ubuntu 18.04 LTS 系統為基底，製作出 Jetson-nano Developer kit 所搭載的系統。

***微軟也決定將 Linux 核心內納入 Windows系統***

# 講者個人 github

目前測試之相關程式碼皆放在 https://github.com/kjoelovelife/Jetson_nano




## 豐富的開源資源

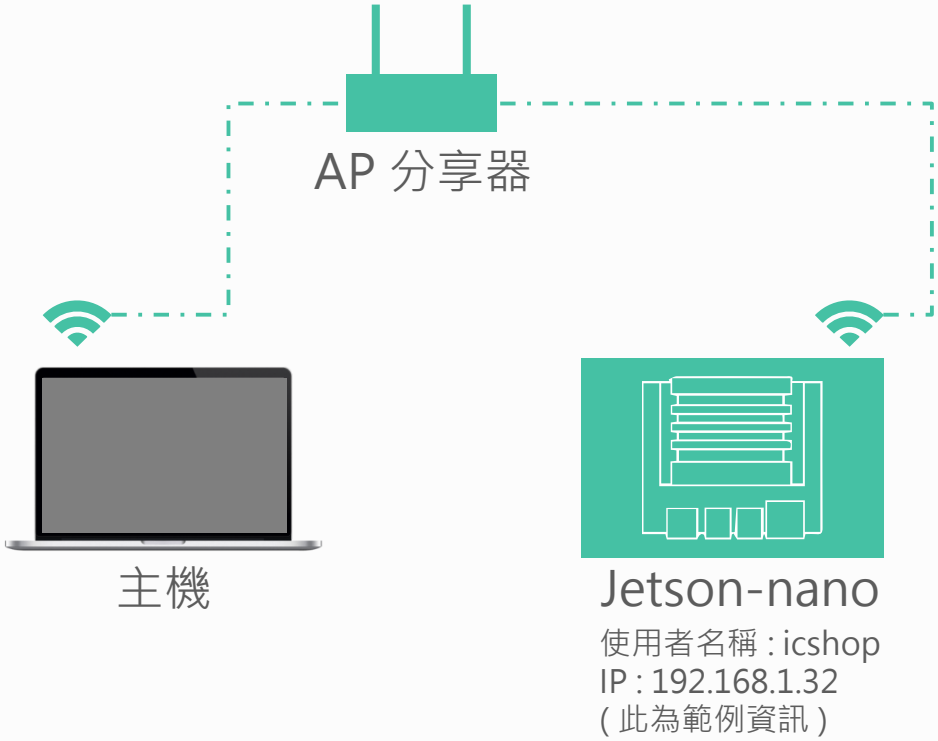
目前我已將測試 Jetson –nano Developer kit 的相關程式碼，放至 github上。

在此 github上，有各種關於 Jetson-nano 使用的心得與部落格連結；也有關於如何建立 Jetson_nano的使用環境腳本檔案。

透過這個 github ，將可以快速上手 Jetson - nano Developer Kit ，並學習使用 Jetson - nano 完成人工智慧技術的應用。

# 簡易遠端連線方式

主機可透過 **SSH ( Secure Shell )與 wifi , 遠端連線至 Jetson – nano Developer kit**




1. 將主機與 Jetson – nano 連線至同一個 AP分享器
2. 記錄 Jetson –nano 的 IP 位置、使用者名稱
3. 開啟終端機視窗
4. 輸入 「 ssh icshop@192.168.1.32 」(此為範例)
5. 輸入 Jetson –nano 的 使用者密碼

- *注意 : windows 系統主機須為 win 10以上系統且須使用系統管理員身分開啟終端機*

# 更新 pyton3-pip 工具

若要下載 Tensorflow-gpu套件 , **則必須更新 python3-pip 工具**

## Python3 依賴套件的下載工具 – pip3

若是想使用 Python3 的各種程式庫、依賴套件，例如 Tensorflow – gpu , numpy , matplotlib 等，可透過 pip3 工具下載

1. **開起 Jetson-nanom終端機，輸入更新指令**
   「pip3 install –upgrade pip」

2. **修改 「 /usr/bin/pip3 」 的部分程式碼，必要時可再更改回來**

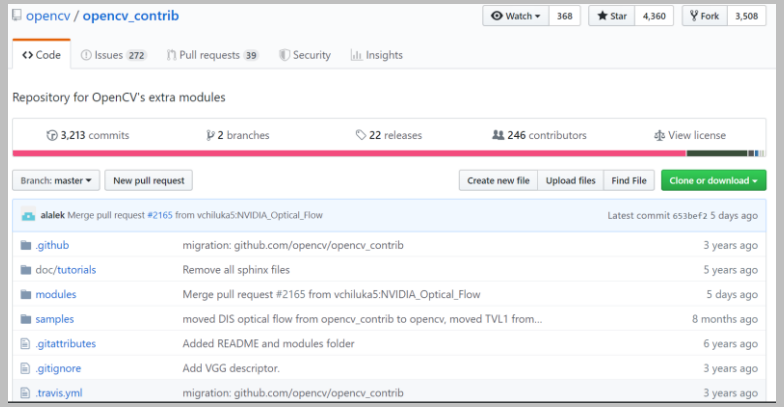
```
from pip import main
if __name__ == '__main__':
    sys.exit(main())
```

→

```
from pip import __main__
if __name__ == '__main__':
    sys.exit(__main__.main())
```

# 安裝 opencv-contrib

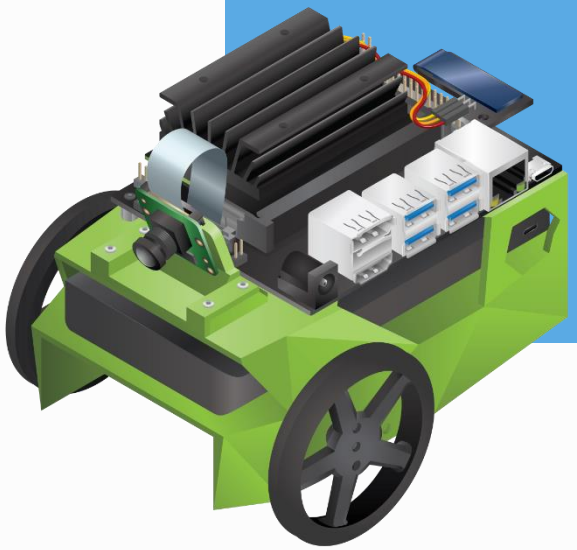
Opencv的擴充套件之一, **可快速使用神經網路模型進行訓練與推論**




目前僅能從源碼安裝

源碼網址：

https://github.com/opencv/opencv_contrib



# 安裝 Jetbot  Services

- 可使用 O’ led 查詢 Jetson – nano developer kit 部份當前資訊
- 可快速使用 Jupyter 進行遠端連線
- 可執行 Jetbot 的範例程式

# Jetson – nano developer kit

使用心得

# Jetson – nano ?
# Jetson – nano developer kit?

# Jetson – nano

單一嵌入式晶片

預計上市時間

2019/6

價格

129 美元

# Jetson-nano developer kit

開發套件組

上市時間

2019/4

價格

99美元

# Jetson – nano developer kit

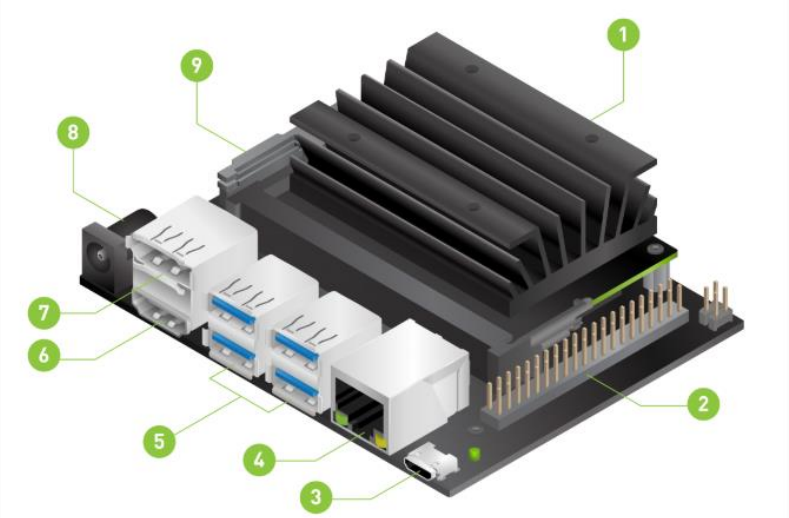
簡單介紹

# JETSON NANO SPECIFICATIONS

| | |
|---|---|
| GPU | 128 Core Maxwell<br>472 GFLOPs (FP16) |
| CPU | 4 core ARM A57 @ 1.43 GHz |
| Memory | 4 GB 64 bit LPDDR4 25.6 GB/s |
| Storage | 16 GB eMMC |
| Video Encode | 4K @ 30 \| 4x 1080p @ 30 \| 8x 720p @ 30<br>(H.264/H.265) |
| Video Decode | 4K @ 60 \| 2x 4K @ 30 \| 8x 1080p @ 30 \| 16x 720p @ 30 \| (H.264/H.265) |
| Camera | 12 (3x4 or 4x2) MIPI CSI-2 DPHY 1.1 lanes<br>(1.5 Gbps) |
| Display | HDMI 2.0 or DP1.2 \| eDP 1.4 \| DSI (1 x2)<br>2 simultaneous |
| UPHY | 1 x1/2/4 PCIE<br>1 USB 3.0 |
| SDIO/SPI/SysIOs/GPIOs/I2C | 1x SDIO / 2x SPI / 5x SysIO / 13x GPIOs / 6x I2C |

# Maker 的小型 AI 單板電腦套件

Maker 的小型 **AI 單板電腦套件**




1. microSD card slot for main storage
2. 40-pin expansion header
3. Micro-USB port for 5V power input or for data
4. Gigabit Ethernet port
5. USB 3.0 ports (x4)
6. HDMI output port
7. DisplayPort connector
8. DC Barrel jack for 5V power input
9. MIPI CSI camera connector

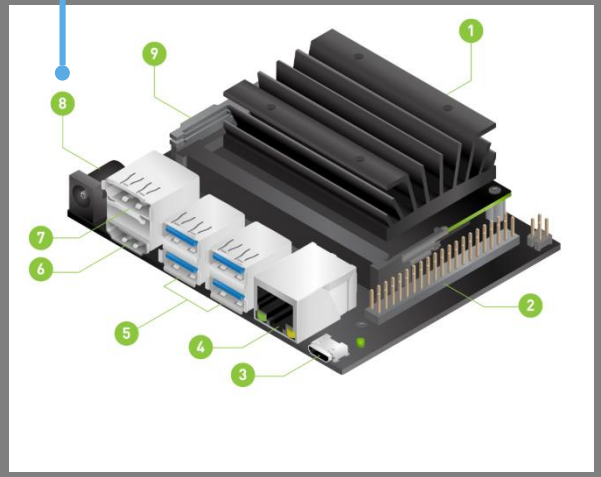
## 其實是個小型吃電怪獸

為了能夠執行神經網路的運算，Jetson –nano 預設所使用的功率為 10W。雖有 5W 功率模式可供選擇，但若是使用一般的行動電源卻也難以負荷過大的瞬間電流。

為了解決因為瞬間電流，而產生的死機問題，我進行了多項嘗試。最後發展出了「桌上開發組」、「行動開發組」兩組套件。

# 電源接孔的選擇

DC jack 接孔較容易找尋 **5V / 4A 的變壓器**




輕巧的 MicroUSB 與
耐用的 DC Jack 中，
我選擇使用 DC Jack

# Jetson-nano 桌上型

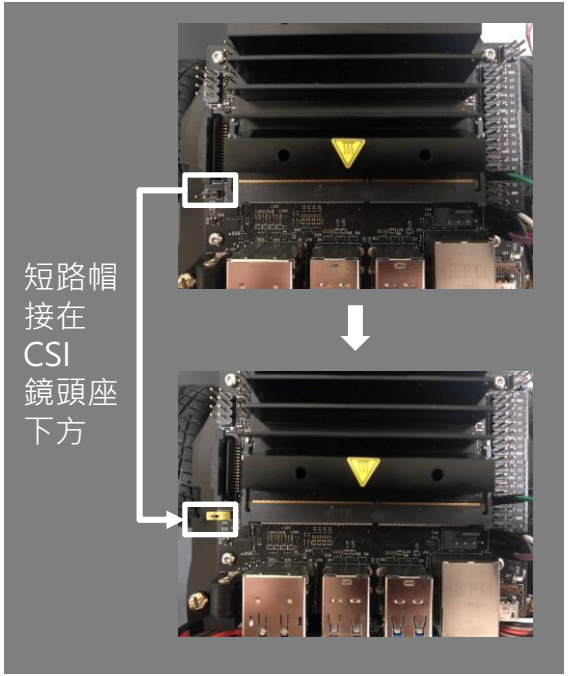
穩定電源輸出的 **桌上型周邊設備(連結)**


全配(含 Jetson-nano)


標配(不含 Jetson-nano)

# 使用 DC Jack 的注意事項

請記得在加上 **短路帽、電壓一樣須為 5V**




## 使用 DC Jack 時須注意

DC Jack 的規格為 5.5x2.1 (mm)。
電壓須注意仍須為 5V 。
電流建議 4A 上下，避免因為瞬間電流過大而造成的死機。

連接短路帽後， MicroUSB 電源供應端將會失去作用。

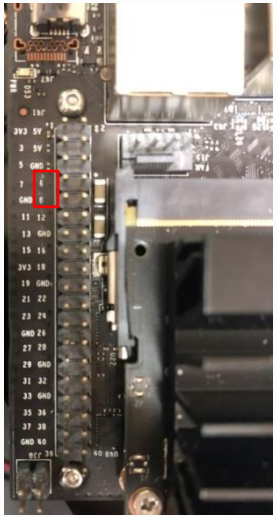
為何要使用 DC Jack接頭?
更新 Jetpack會使用到。

# GPIO腳位

與樹莓派**一模一樣**

樹莓派

Jetson-nano
Developer kit



## 但要注意的是標示正確與否

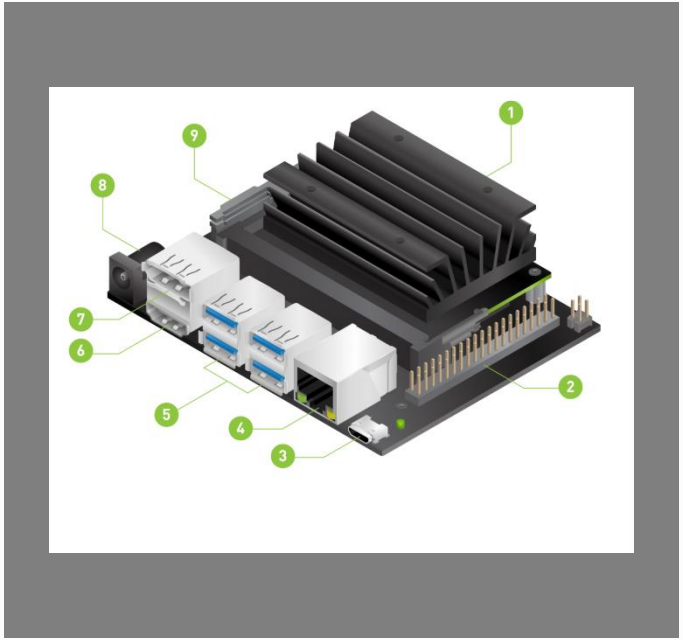
Jetson-nano developer kit
很貼心的把每個腳位的編號
與功能都列在旁邊。
但或許是為了趕上發布時間，
有小部分腳位標示錯誤。
例如本次測試的板子標示為
6、8號腳位，但實際上是
8、10號腳位。

# Jetson-nano V.S. Raspberry pi

| | Jetson Nano | Raspberry Pi |
|---|---|---|
| CPU | 64-bit Quad-core ARM A57 (1.43 GHz) | 1.4 GHz 64-bit quad-core ARM Cortex-A53 |
| GPU | 128-Core Nvidia Maxwell | Broadcom VideoCore IV |
| RAM | 4GB DDR4 | 1GB DDR2 |
| WIFI | X | 802.11ac |
| Bluetooth | X | LE 4.2 |
| Ethernet | Gigbit | Gigbit (300Mbps max) |
| GPIO | 40 pin | 40 pin |
| USB | USB 2.0 x 3<br>USB 3.0 x 1 | USB 2.0 x 4 |
| Audio | X | Audio jack |
| Power | 5~10W | 400 mA (2.0W) |
| Price | $99 | $35 |
| 算力 | 472 Gflops | 24 Gflop |

# 僅有 gigabit ethernet port

需自行安裝**藍芽、wifi等無線模組**



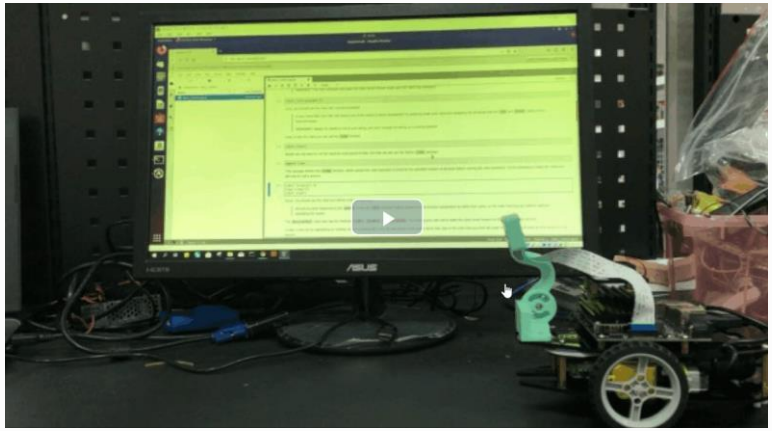

## 不太方便的物聯網開發套件

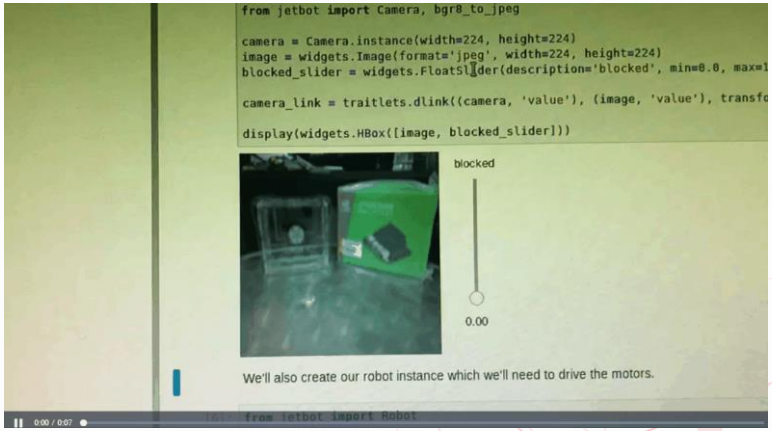
官方售出的 Jetson-nano developer kit ，其中並無包含藍芽、wifi 等無線模組，因此需自行安裝。

本次測試時使用 **Intel8265AC無線網卡**網路卡與 EDIMAX 7811-Un USB網路卡

# Jet – falcon 千年鷹

能夠使用 **Jetbot的移動平台**





# 人臉辨識實作

# 今日使用之模型

Opencv_contrib套件中的 "Haar Cascade classifier"

**Haar-cascade Detection in OpenCV**

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: Cascade Classifier Training.

Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/ folder. Let's create face and eye detector with OpenCV.

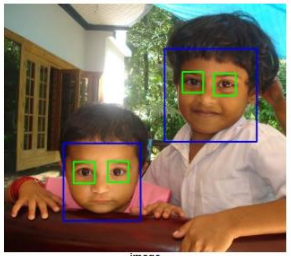
First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode.

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Now we find the faces in the image. If faces are found, it returns the positions of detected faces as Rect(x,y,w,h). Once we get these locations, we can create a ROI for the face and apply eye detection on this ROI (since eyes are always on the face !!! ).

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result looks like below:



image

# 人臉辨識主要步驟

蒐集資料、 **訓練模型** 、開始偵測

開始偵測

訓練模型

收集資料

# Thanks