Raspberry Pi 寵物小車

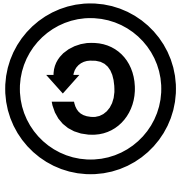
台灣樹莓派 <sosorry@raspberrypi.com.tw>

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。

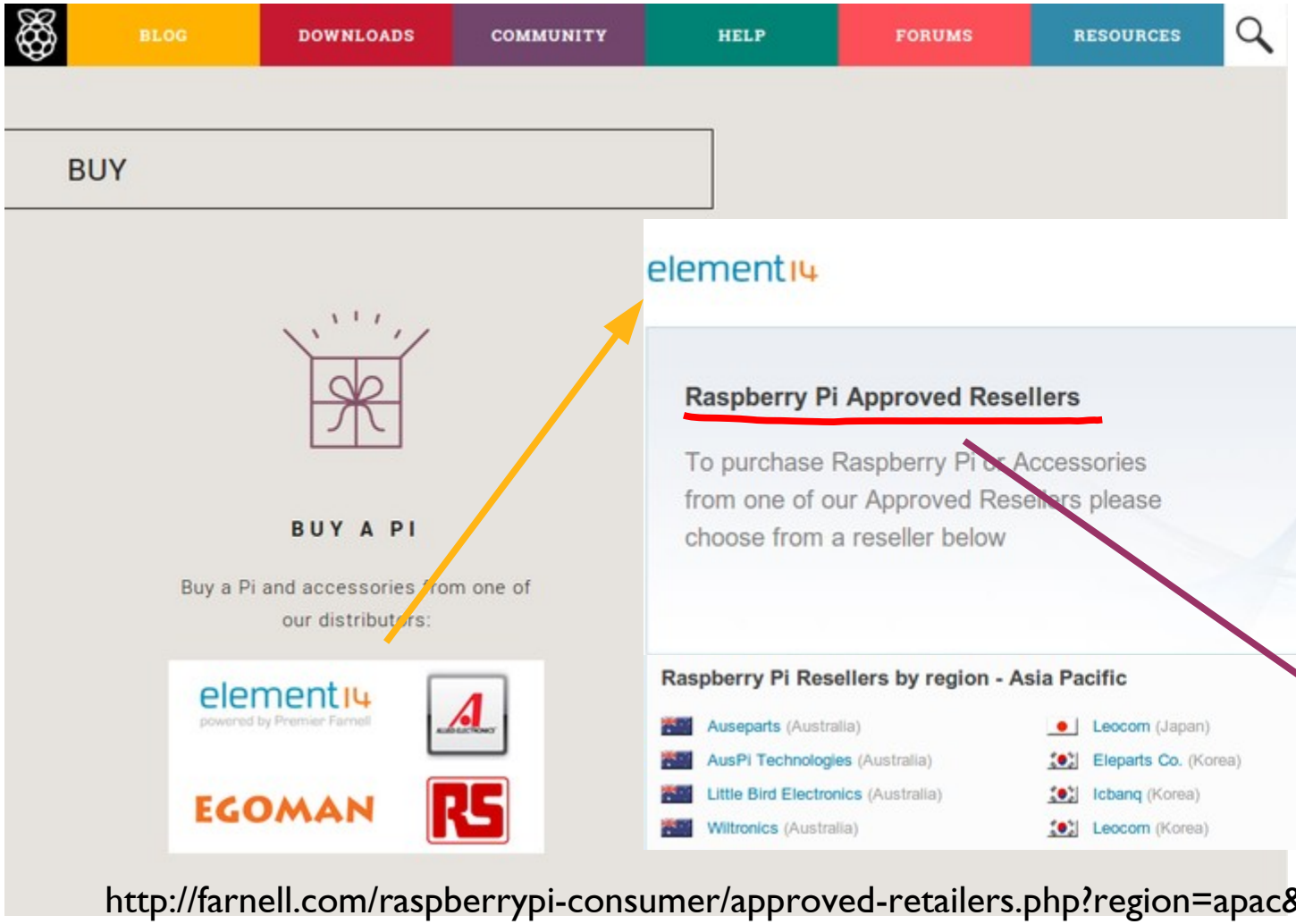
非商業性 — 你不得將本素材進行商業目的之使用。



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。

about 台灣樹莓派

- Raspberry Pi 官方經銷商



http://farnell.com/raspberrypi-consumer/approved-retailers.php?region=apac&MER=MER-LM-OB-RPICC-76315

社群 x 活動

- 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

1 Pi選購指南

2 Pi設定安裝

3 Linux系統管理

4 GPIO
遊戲機學習套件
入門

- 感測器學習套件
(基礎)+(進階)
- 空氣盒子套件
- 智慧開關
- 全端學習套件(進階)

4 無線
模組套件
藍牙+ESP8266

- IoT閘道器套件
(LoRa)
- 生理資訊監控IoT
(藍牙)

4 影像
辨識相機
Camera+Python+OpenCV

- 寵物小車
- 小鴨車

4 語音
套件(自然語言處理)

今日主題

- 馬達與小車介紹
  - GPIO 介紹
  - 馬達控制
  - 小車組裝與控制
- Raspberry Pi Camera 介紹
  - 數位影像處理與 OpenCV
  - 自走車設計與優化

今日環境

- 硬體：Raspberry Pi 3B/3B+
- 作業系統：2018-06-27-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

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

新增三行

  - 修改/boot/cmdline.txt，將quiet splash的quiet移除

```
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-cons
  oles quiet init=/usr/lib/raspi-config/init_r
  esize.sh
```

刪除 quiet

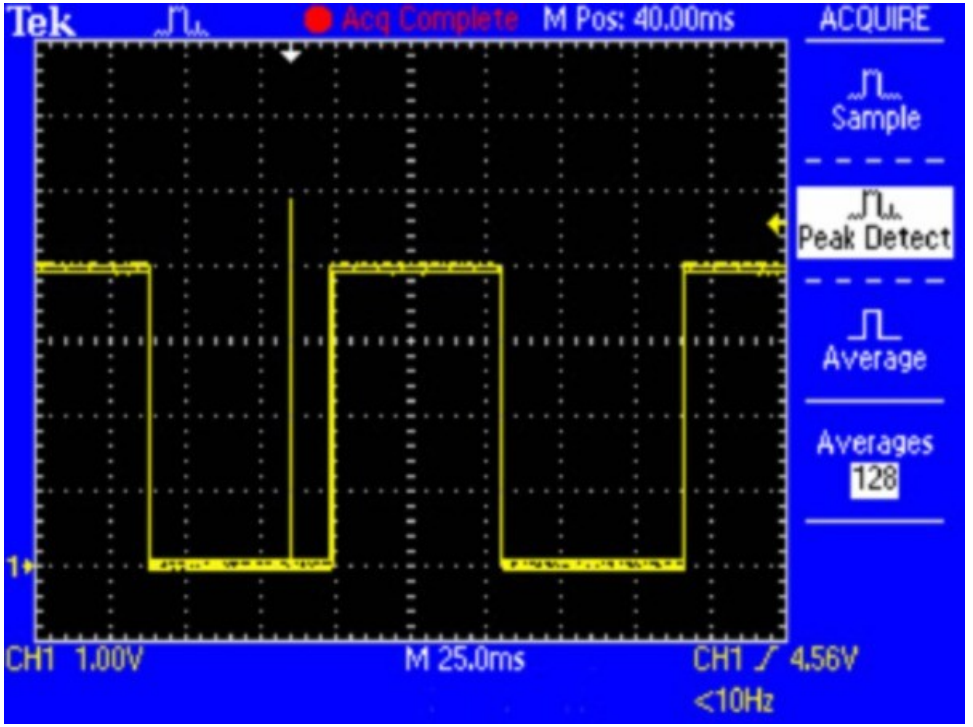
安裝今日所需軟體

- `$ sudo apt-get update`
- `$ sudo apt-get install -y python-pip python-opencv python-dev x11vnc`
- `$ sudo pip install readchar flask`

Raspberry Pi GPIO 介紹

什麼是 GPIO ？

- General Purpose Input Output
- GPIO 是一種可用軟體控制的數位訊號



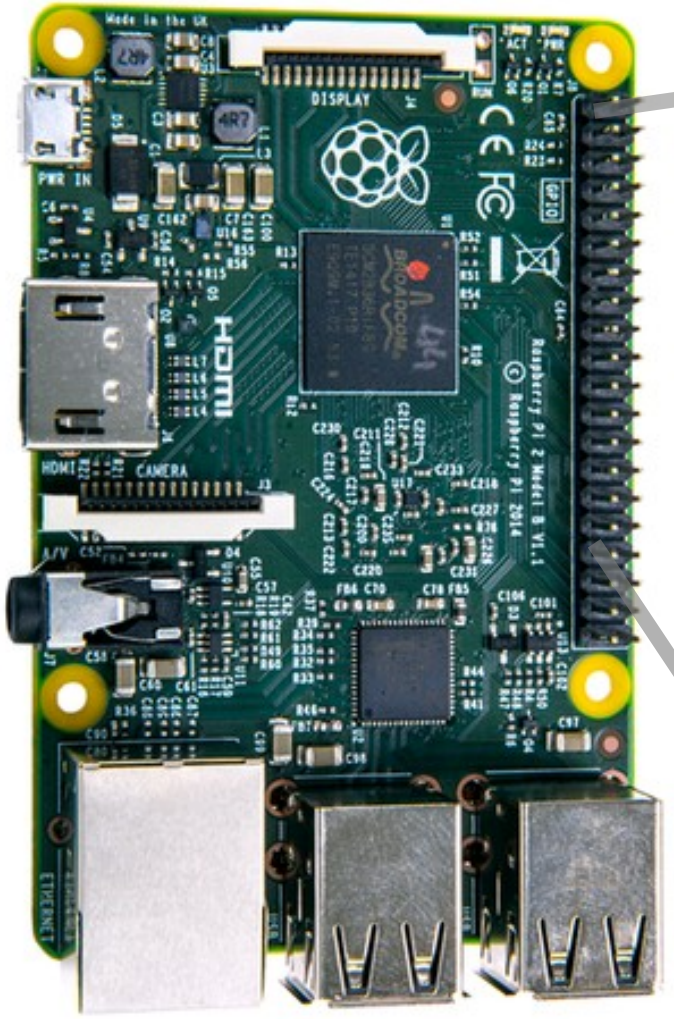
http://www.tek.com/datasheet/tps2000b-series-digital-storage-oscilloscopes-datasheet

用軟體控制什麼？

- 決定是輸入還是輸出
  - 輸出：寫值到某根腳位
  - 輸入：從某根腳位讀值
- 等待中斷（interrupt）的發生
- 決定是正緣觸發還是負緣觸發

GPIO 腳位在哪裡？

Z 字型的腳位編號



Pin 1 Pin2

Pin3 Pin4

Pin25 Pin26

3V3 Power	1	2	5V Power	
GPIO2 SDA1 I2C	3	4	5V Power	
GPIO3 SCL1 I2C	5	6	Ground	
GPIO4	7	8	GPIO14 UART0_TXD	
Ground	9	10	GPIO15 UART0_RXD	
GPIO17	11	12	GPIO18 PCM_CLK	
GPIO27	13	14	Ground	
GPIO22	15	16	GPIO23	
3V3 Power	17	18	GPIO24	
GPIO10 SPI0_MOSI	19	20	Ground	
GPIO9 SPI0_MISO	21	22	GPIO25	
GPIO11 SPI0_SCLK	23	24	GPIO8 SPI0_CE0_N	
Ground	25	26	GPIO7 SPI0_CE1_N	
ID_SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM	
GPIO5	29	30	Ground	
GPIO6	31	32	GPIO12	

http://www.raspberrypi-spy.co.uk/

幾個 GPIO 的數字

- GPIO 高電位輸出為 3.3V
- GPIO 容忍輸入電位為 3.3V
- 單一 Pin 輸出電流為 3mA-16mA
- 全部 Pin 輸出總和小於 50mA
- GPIO 輸入低電位為小於 0.8V, 高電位為大於 1.3V

https://www.scribd.com/doc/101830961/GPIO-Pads-Control2

如何控制 Raspberry Pi 的 GPIO？

- C
- C + wiringPi
- C#
- Ruby
- Perl
- Python
- Scratch
- Java Pi4J Library
- Shell script

數位輸出

實驗 1-1：硬體的 Hello World

目的：從硬體到軟體的思維

LED

- 發光二極體
- 單向導通
- 省電

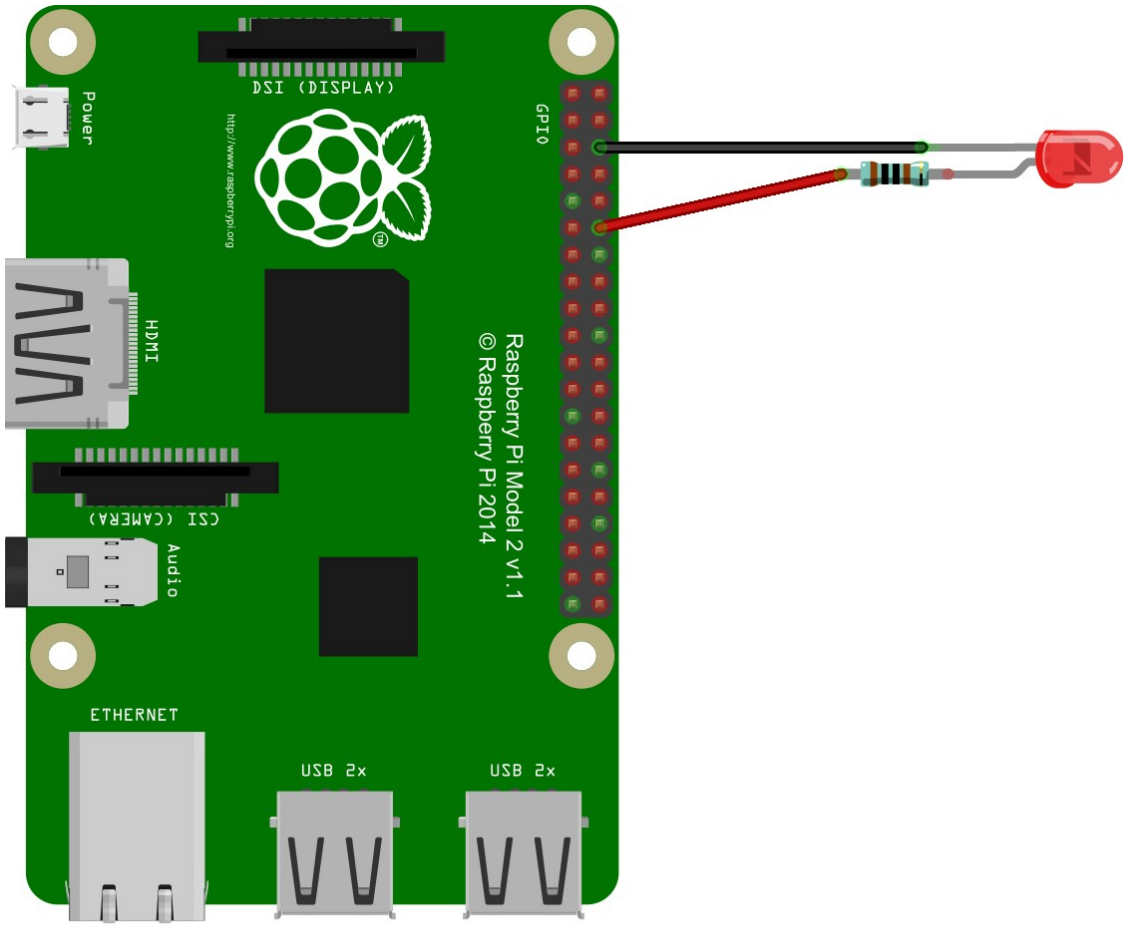
短腳接負極 GND

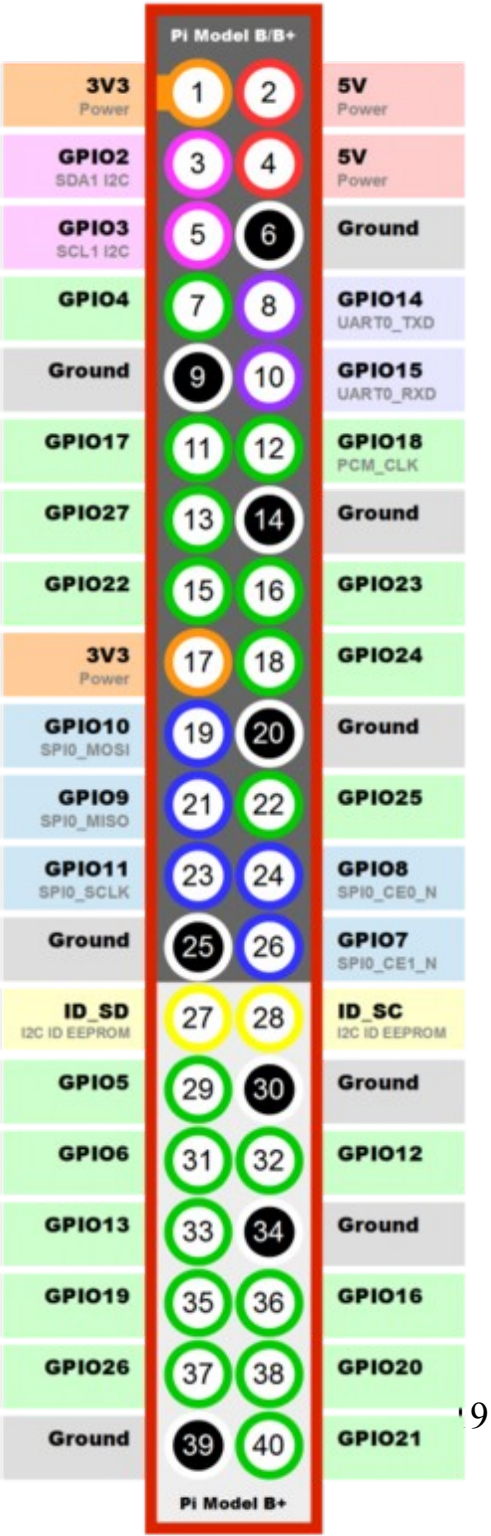
長腳接正極 Vcc

http://upload.wikimedia.org/wikipedia/commons/c/c8/5mm_Red_LED.jpg

接線圖

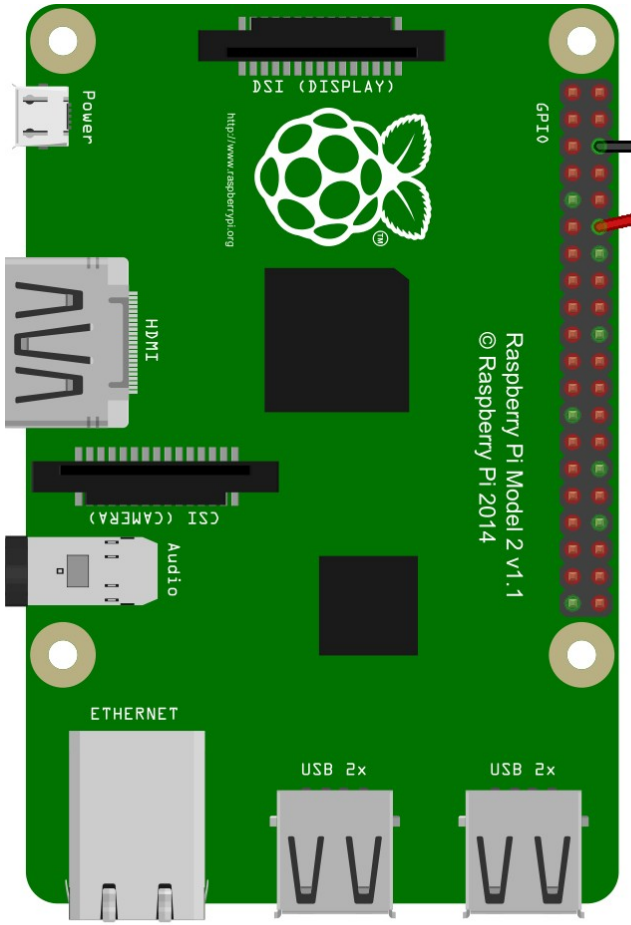
LED	RPi
長腳 (RED)	Pin12
短腳 (BLACK)	Pin6 (Ground)

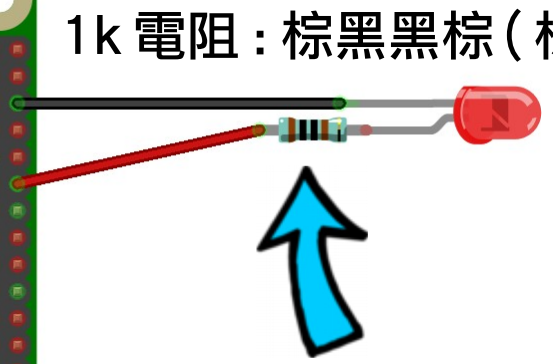




接線圖

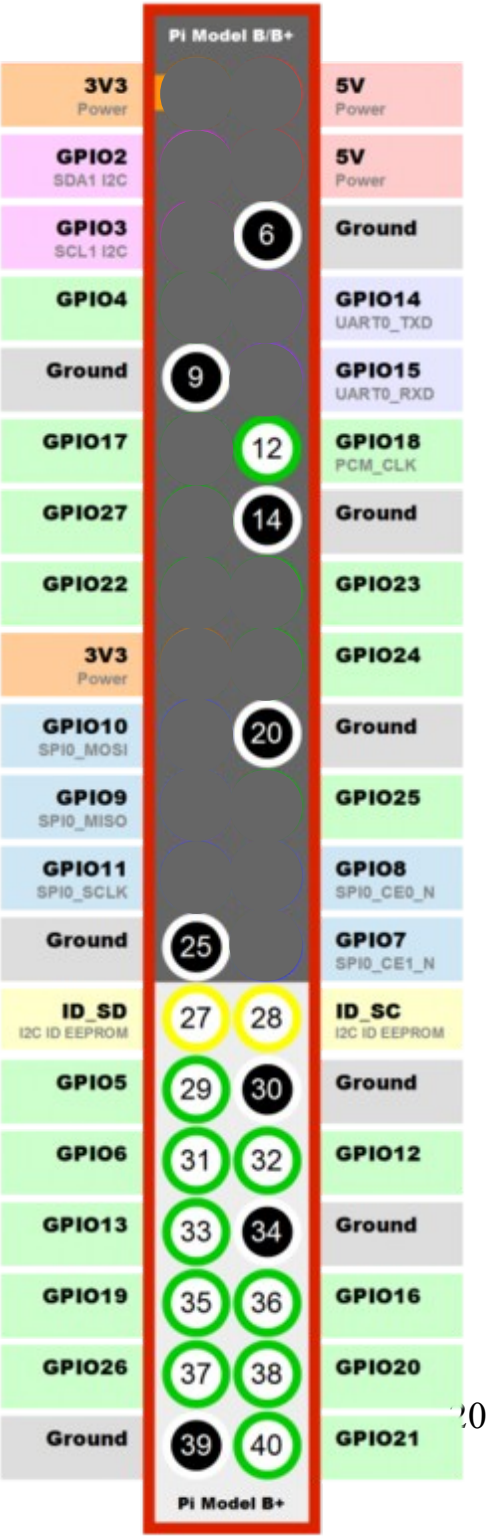
LED	RPi
長腳 (RED)	Pin12
短腳 (BLACK)	Pin6 (Ground)



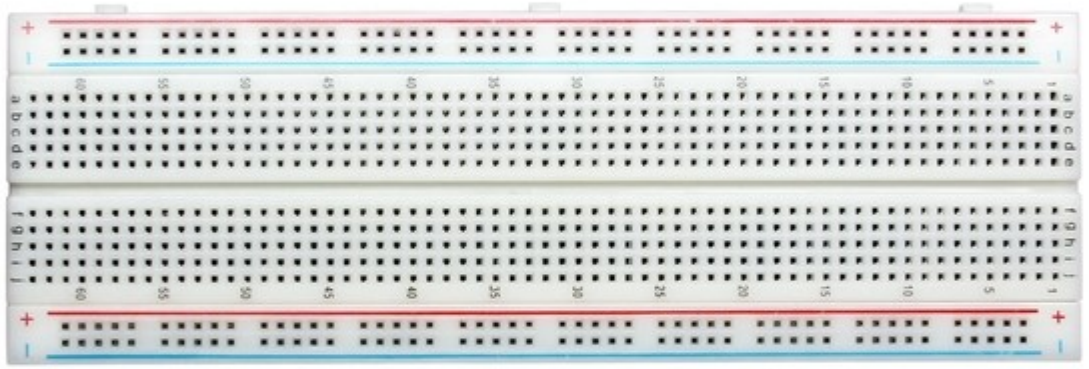


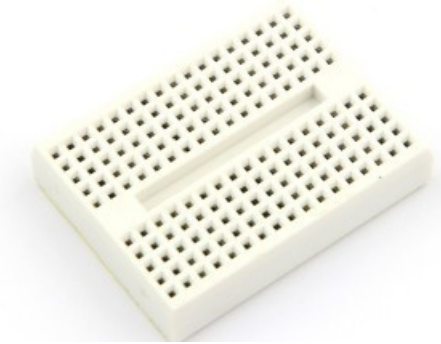
1k 電阻：棕黑黑棕（棕）

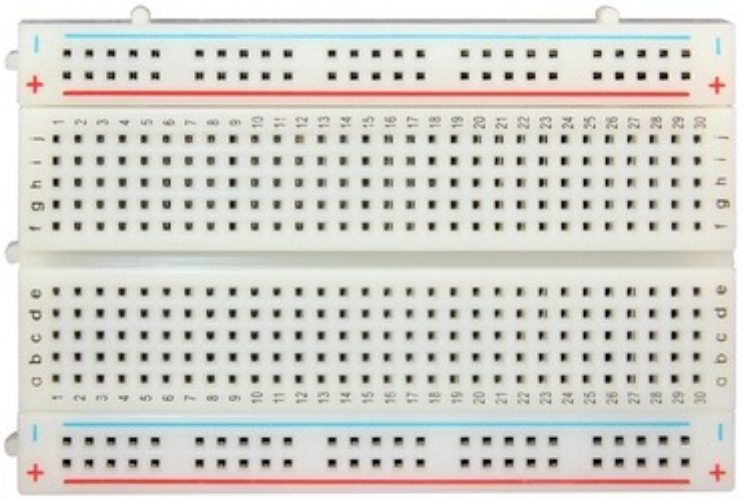
接電阻的目的是
限制電流大小，
防止 LED 燒毀

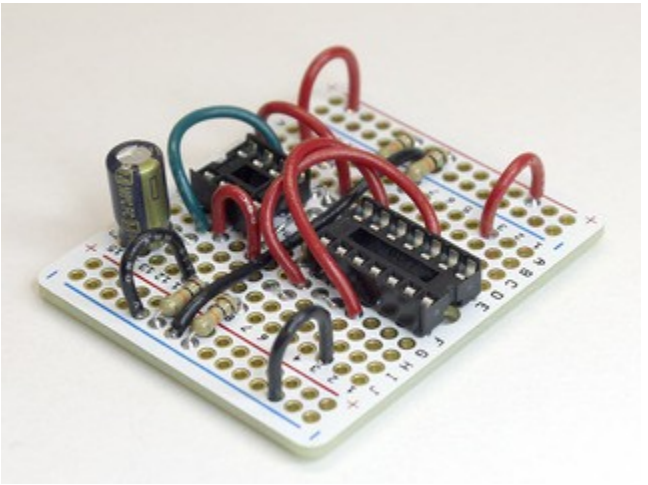


麵包板的種類





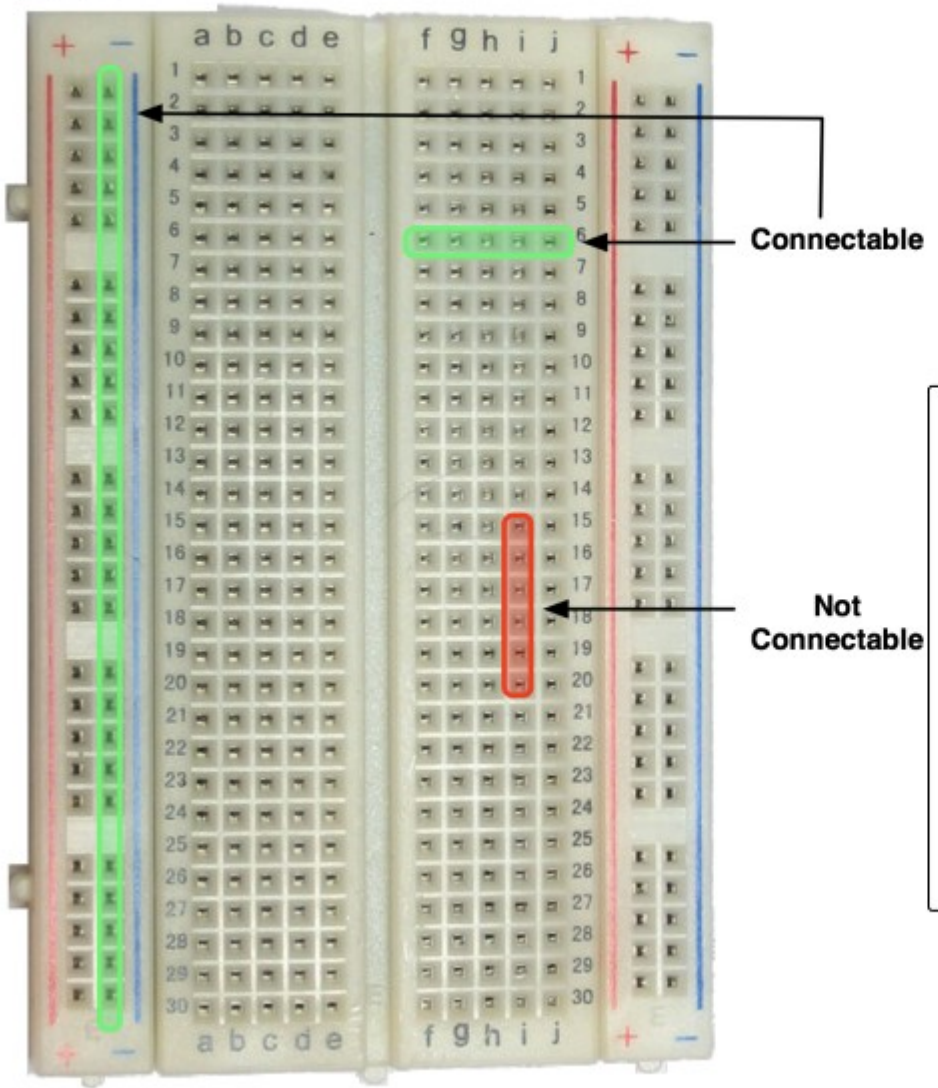




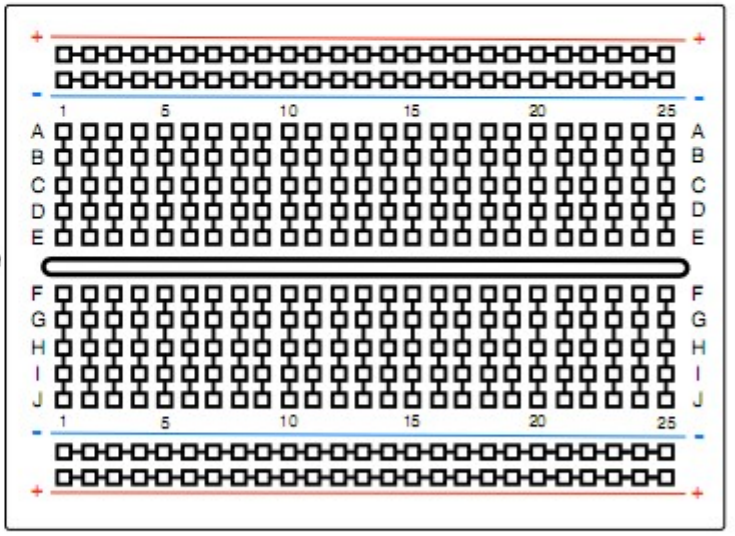
https://goo.gl/JipVgH

麵包板的使用

interaction design centre



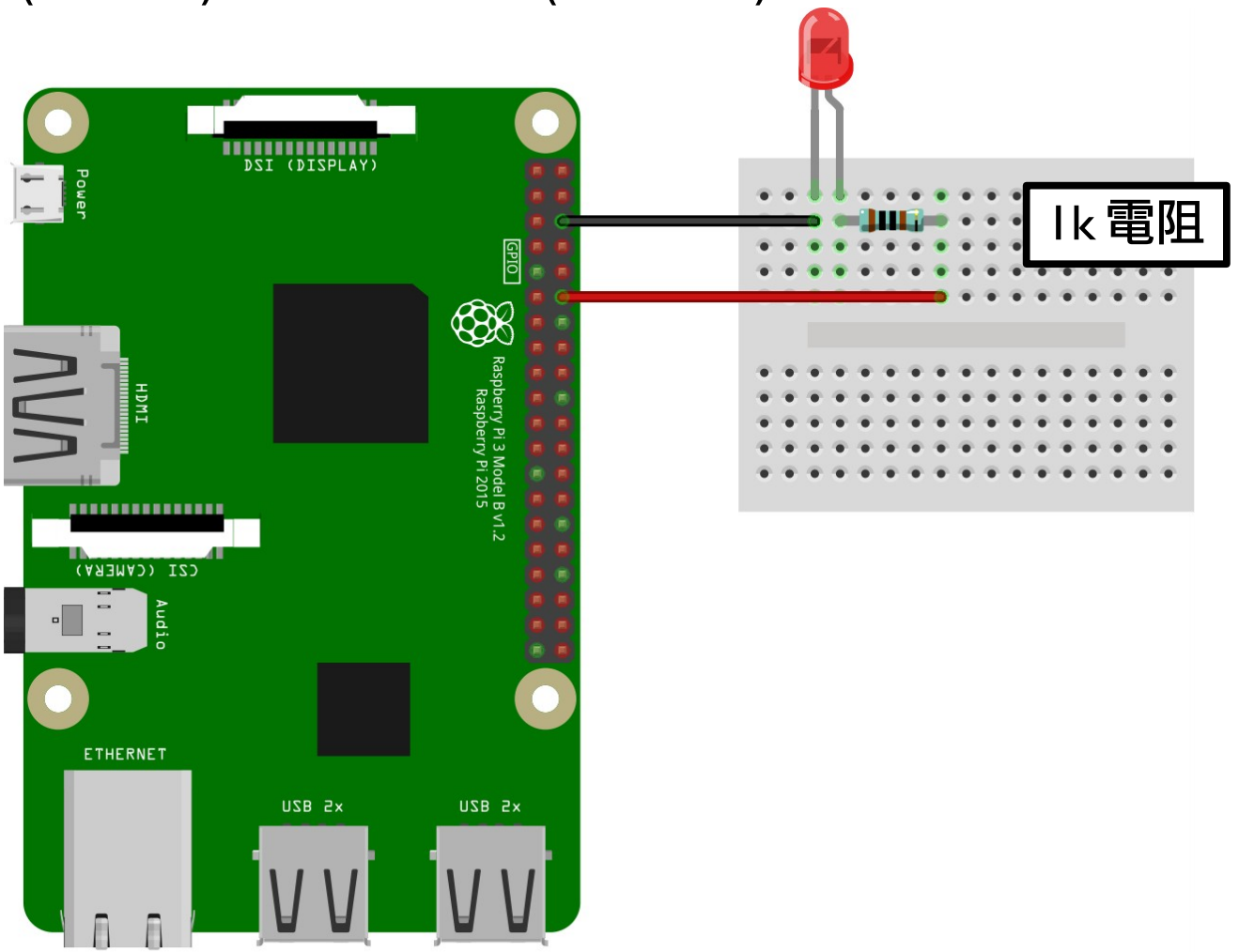
Solderless Breadboard

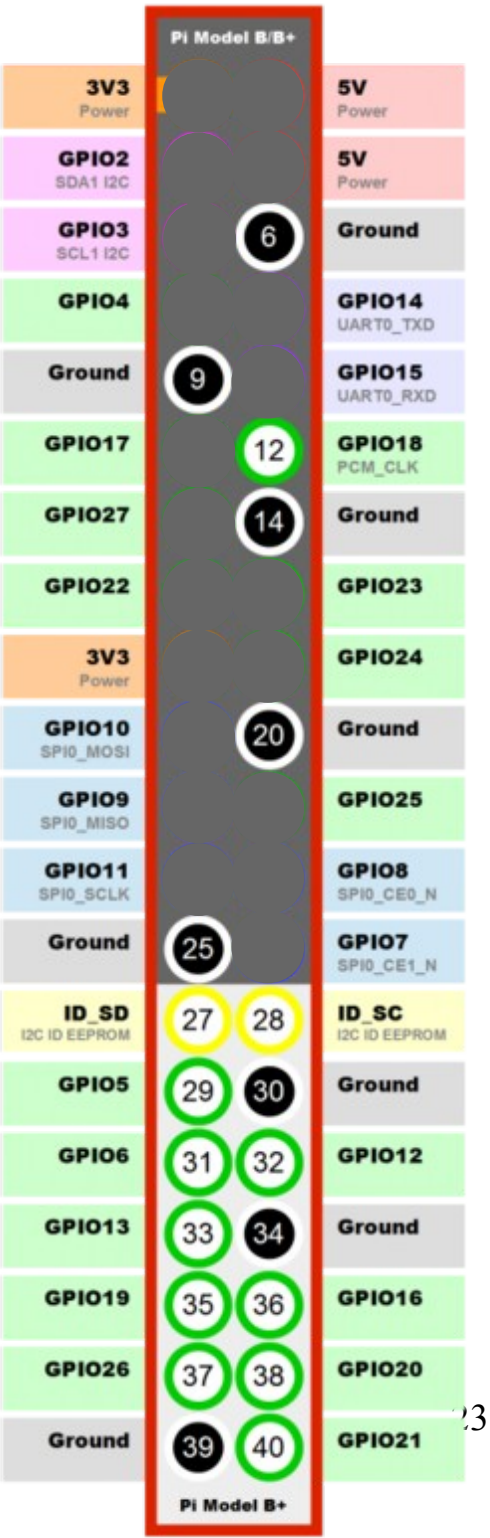


https://www.slideshare.net/eoinbrazil/imediaarduino08

使用麵包板

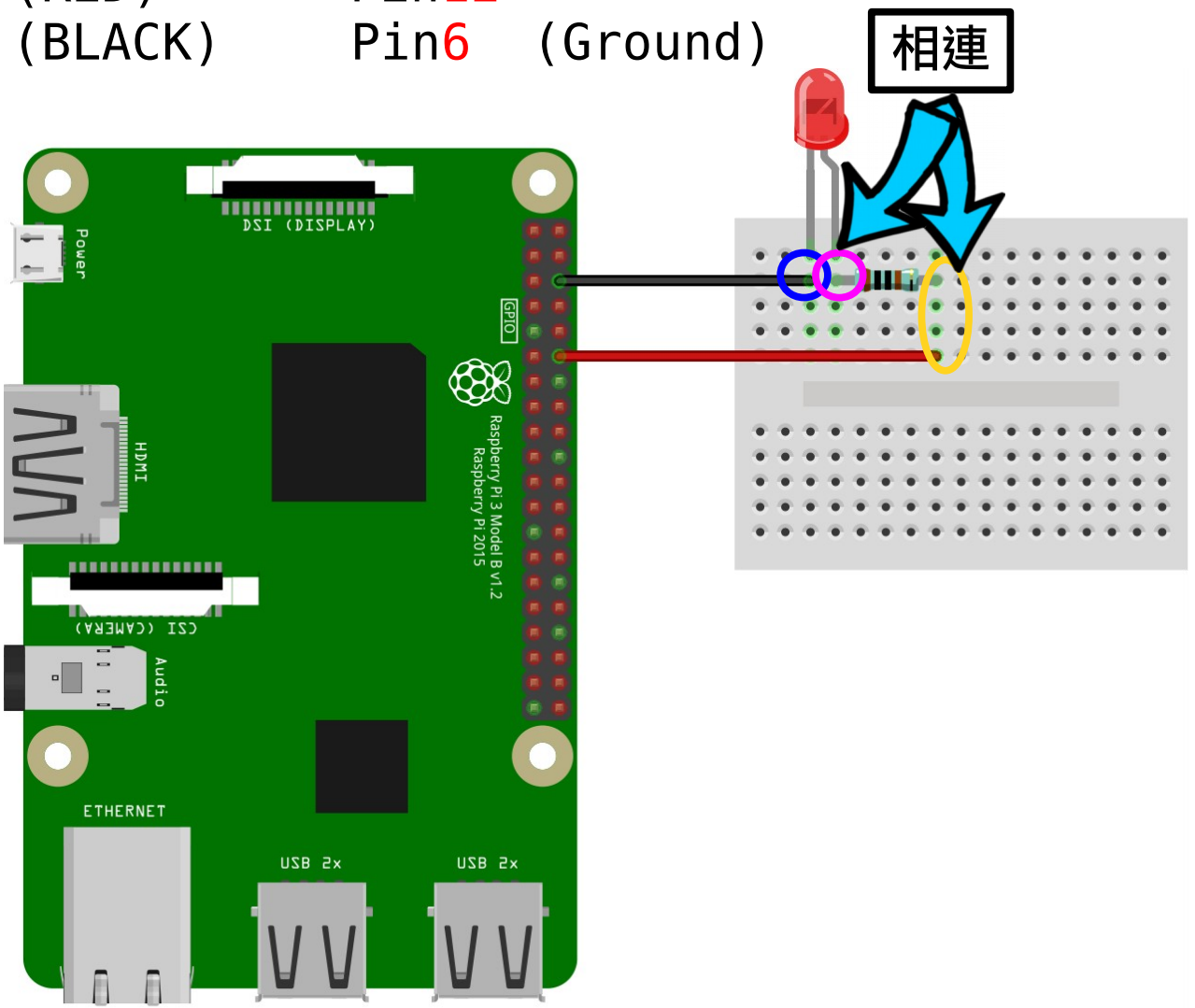
LED	RPi
長腳 (RED)	Pin12
短腳 (BLACK)	Pin6 (Ground)

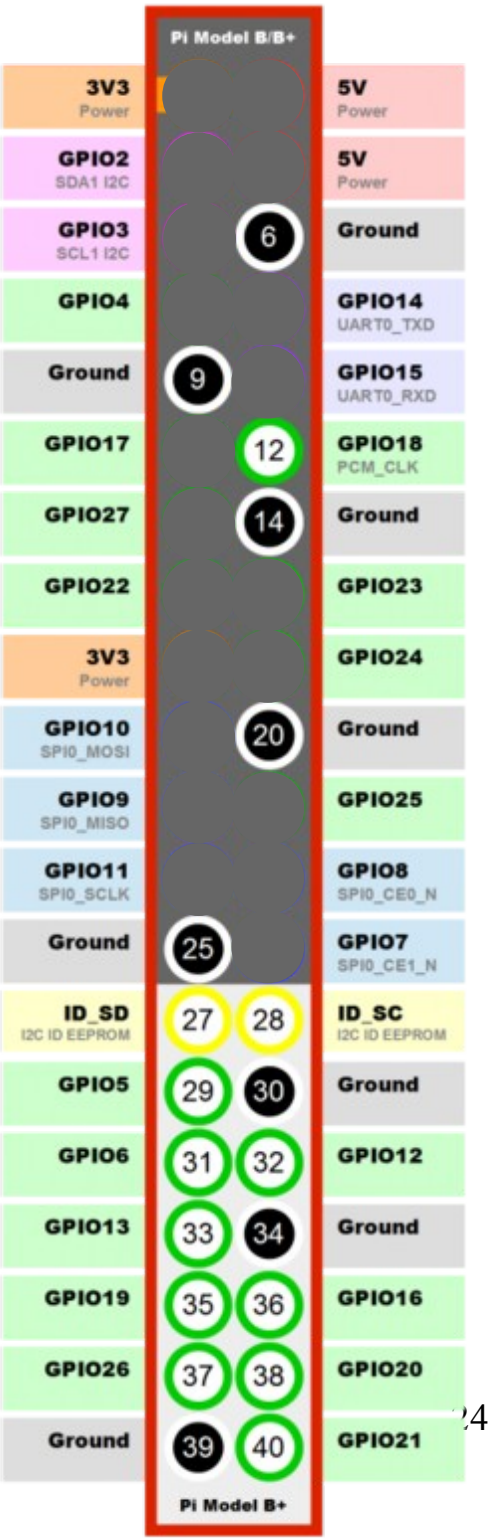




使用麵包板

LED	RPi
長腳 (RED)	Pin12
短腳 (BLACK)	Pin6 (Ground)





開始用 Python 控制 GPIO 吧

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 RPi.GPIO
# import MODULE as ALIAS
import RPi.GPIO as GPIO
```

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 condition1:
    process1
elif condition2:
    process2
else:
    process3
process4
```

函式

```
def function_name():
    process

def function_name( param_name ):
    process

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

Python Code 基本流程

- 載入模組 (Import module)
- 選擇編號系統 (Define pin numbering)
- 定義腳位 (Setup up a channel)
- 讀取輸入 / 寫入輸出 (Input/Output)
- 清理 (Cleanup)

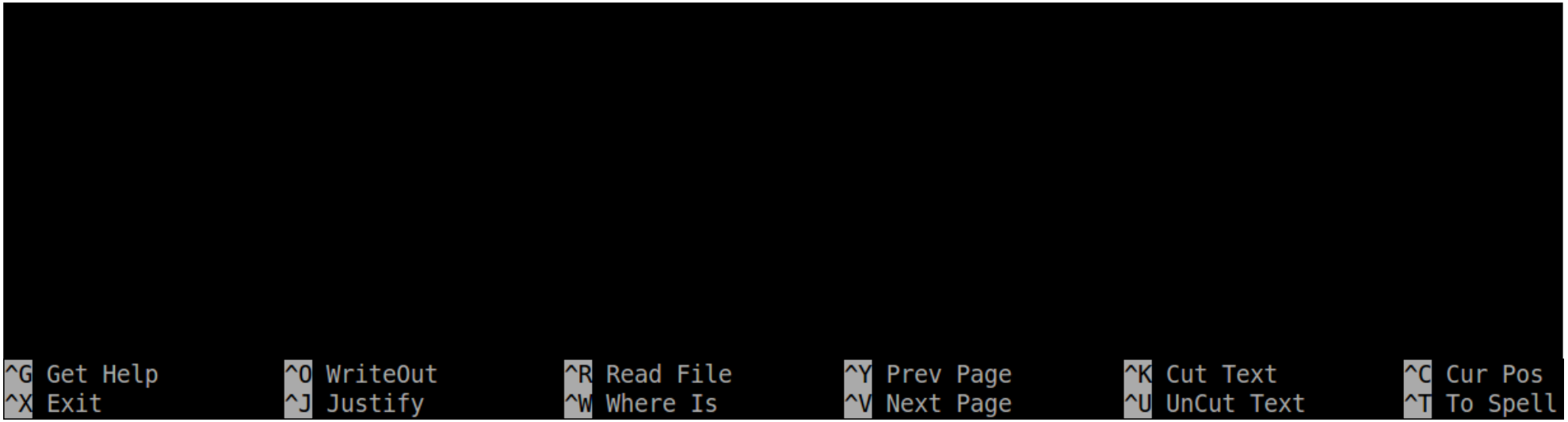
http://code.google.com/p/raspberry-gpio-python/wiki/BasicUsage

一個實際的範例

```
#!/usr/bin/python

import RPi.GPIO as GPIO      # Import module
import time

GPIO.setmode(GPIO.BOARD)     # Define pin numbering
LED_PIN = 12
GPIO.setup(LED_PIN, GPIO.OUT)      # Setup up a channel

print "LED is on"
GPIO.output(LED_PIN, GPIO.HIGH)    # Input/Output status
time.sleep(3)

GPIO.cleanup()     # Cleanup
```

以下都是使用別名(alias)

- 建立新檔案
  - $ nano <檔名>, 例如 $ nano led_on.py



- 離開:Ctrl + x
  - > 令存新檔:y
  - > 不存離開:n
  - > 離開:Ctrl + c

更多 RPi.GPIO 的使用方法

- Wiki
  - http://sourceforge.net/p/raspberry-gpio-python/wiki/Home/

- 用 Raspberry Pi 學 GPIO - 自己做遊戲機
  - http://www.slideshare.net/raspberrypi-tw/gpio-gameconsolestarterkit

DEMO
led_on.py

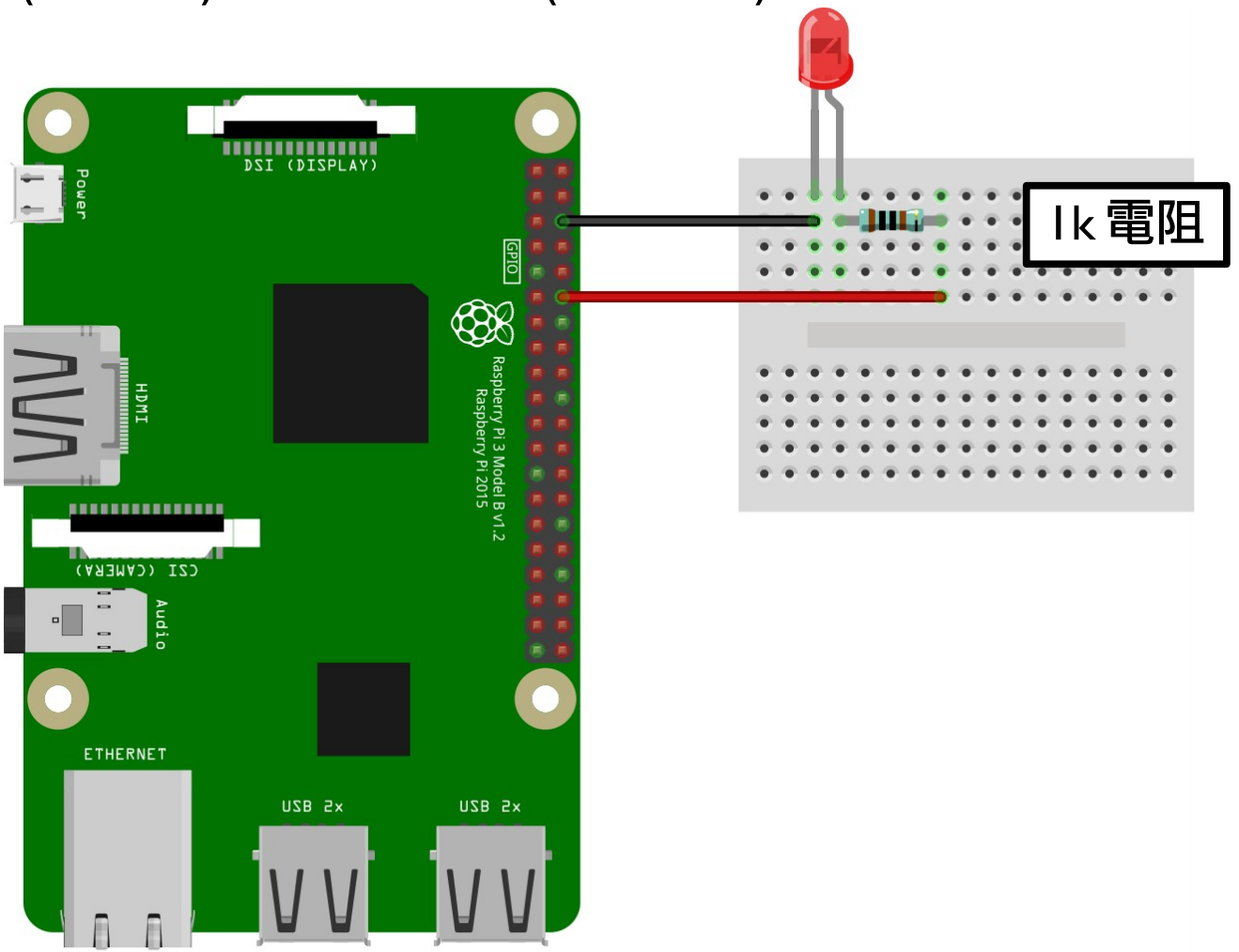
```
$ cd ~/pi-follower-car/01-gpio
$ python led_on.py
```

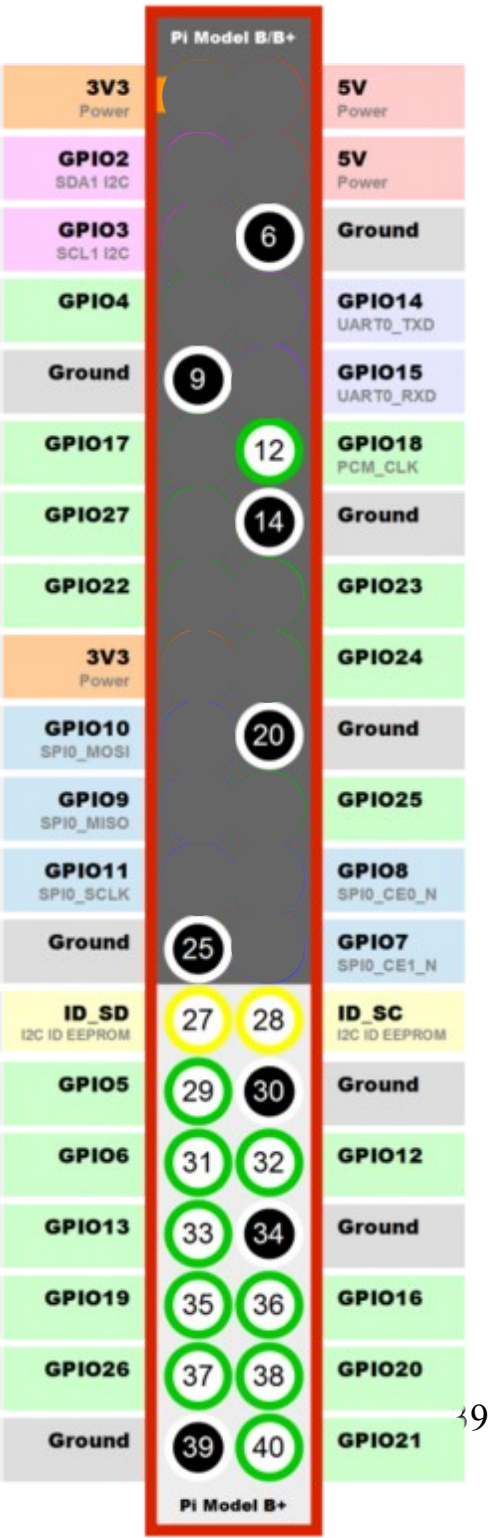
實驗 1-2：控制 LED 閃爍

目的：熟悉 Python 語法

一樣的接線圖

LED	RPi
長腳 (RED)	Pin12
短腳 (BLACK)	Pin6 (Ground)





fritzing

接住例外

按 Ctrl+c 跳出迴圈

```
try:
    while True:
        Print "LED is on"
        GPIO.output(LED_PIN, GPIO.HIGH)
        time.sleep(1)
        Print "LED is off"
        GPIO.output(LED_PIN, GPIO.LOW)
        time.sleep(1)
except KeyboardInterrupt:
    print "Exception: KeyboardInterrupt"
finally:
    GPIO.cleanup()
```

DEMO
led_blink.py

```
$ cd ~/pi-follower-car/01-gpio
$ python led_blink.py
```

實驗 1-3：改變 LED 亮度

目的：調變脈衝寬度輸出脈波

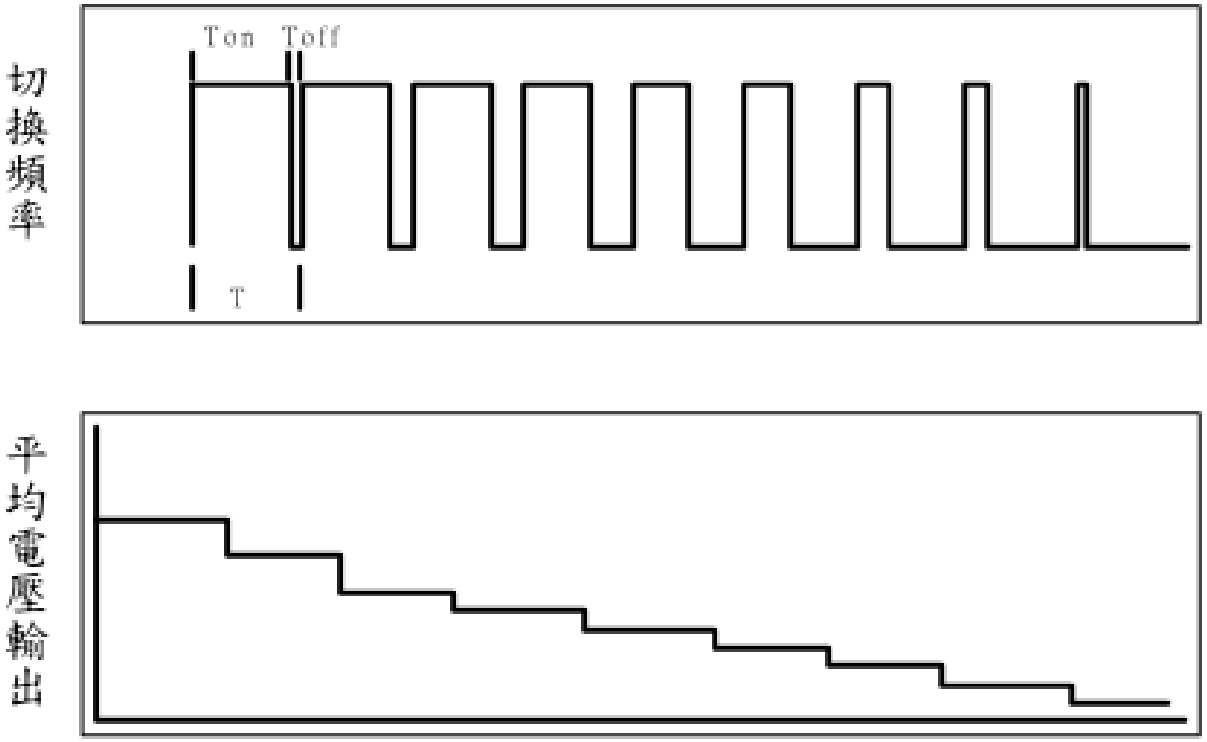
數位與類比

- 數位：0 與 1 的訊號
- 類比：連續的訊號
- 可是 GPIO 腳位輸出都是固定值，怎麼辦？

http://www.bitscope.com/software/blog/DJ/?p=DJ19A

脈寬調變 (Pulse-Width Modulation)

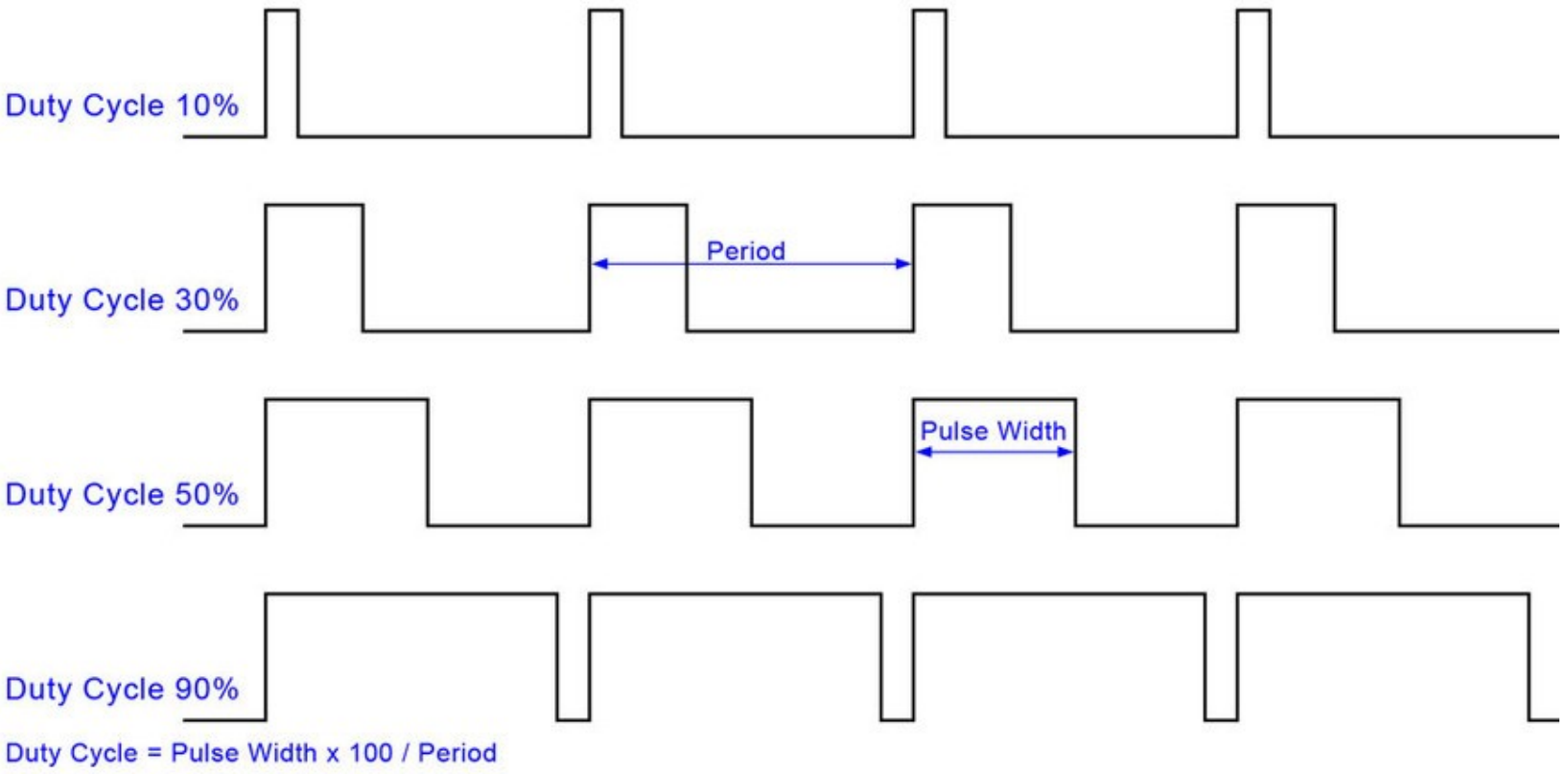
- 是將類比信號轉為脈波的一種技術
- 頻率不變 + 改變工作週期，使整體平均電壓值改變
- 改變工作週期 (duty cycle)= 改變平均電壓



http://wiki.csie.ncku.edu.tw/embedded/PWM

公式計算

- 輸出總功率 = 脈衝寬度（時間）x 高電位值



http://www.protostack.com/blog/2011/06/atmega168a-pulse-width-modulation-pwm/

GPIO.PWM()

- To create a PWM instance:
  - p = GPIO.PWM(channel, frequency)
- To start PWM:
  - p.start(dc) # dc is the duty cycle
- To change the duty cycle:
  - p.ChangeDutyCycle(dc) # where 0.0 <= dc <= 100.0
- To stop PWM:
  - p.stop()

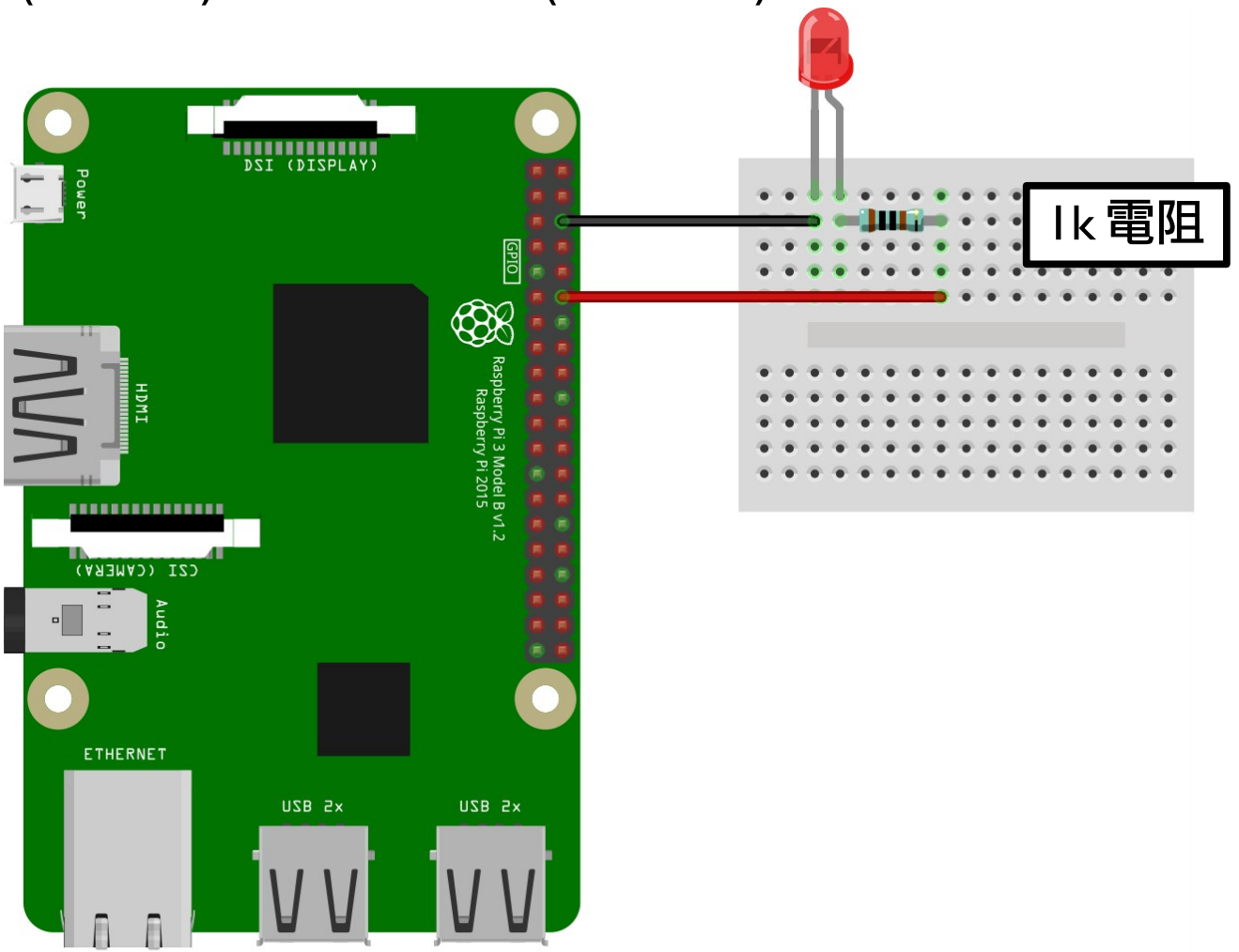
http://sourceforge.net/p/raspberry-gpio-python/wiki/PWM/

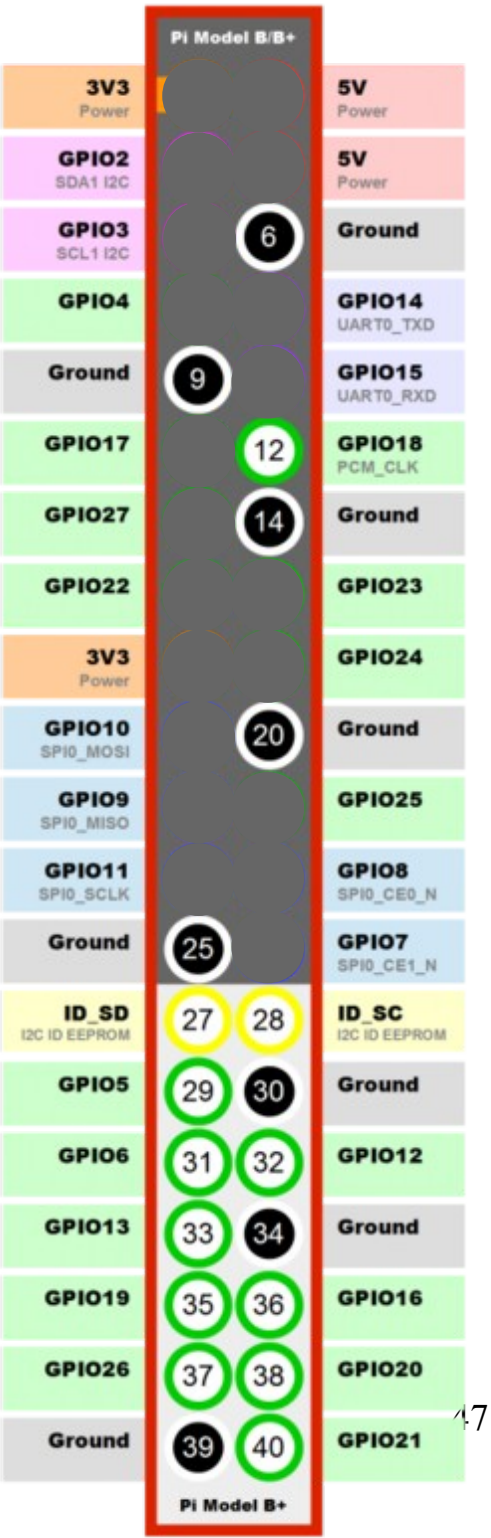
接線圖

LED **RPi**

長腳 (RED) Pin12

短腳 (BLACK) Pin6 (Ground)





fritzing

互動式的調光

```
PWM_PIN = 12
GPIO.setup(PWM_PIN, GPIO.OUT)
pwm = GPIO.PWM(PWM_PIN, 100)
pwm.start(0)

try:
    while True:
        duty_s = raw_input("Enter Brightness (0 to 100):")
        duty = int(duty_s)

        if duty >= 0 and duty <=100 :
            pwm.ChangeDutyCycle(duty)

except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
```

DEMO
pwm_led.py

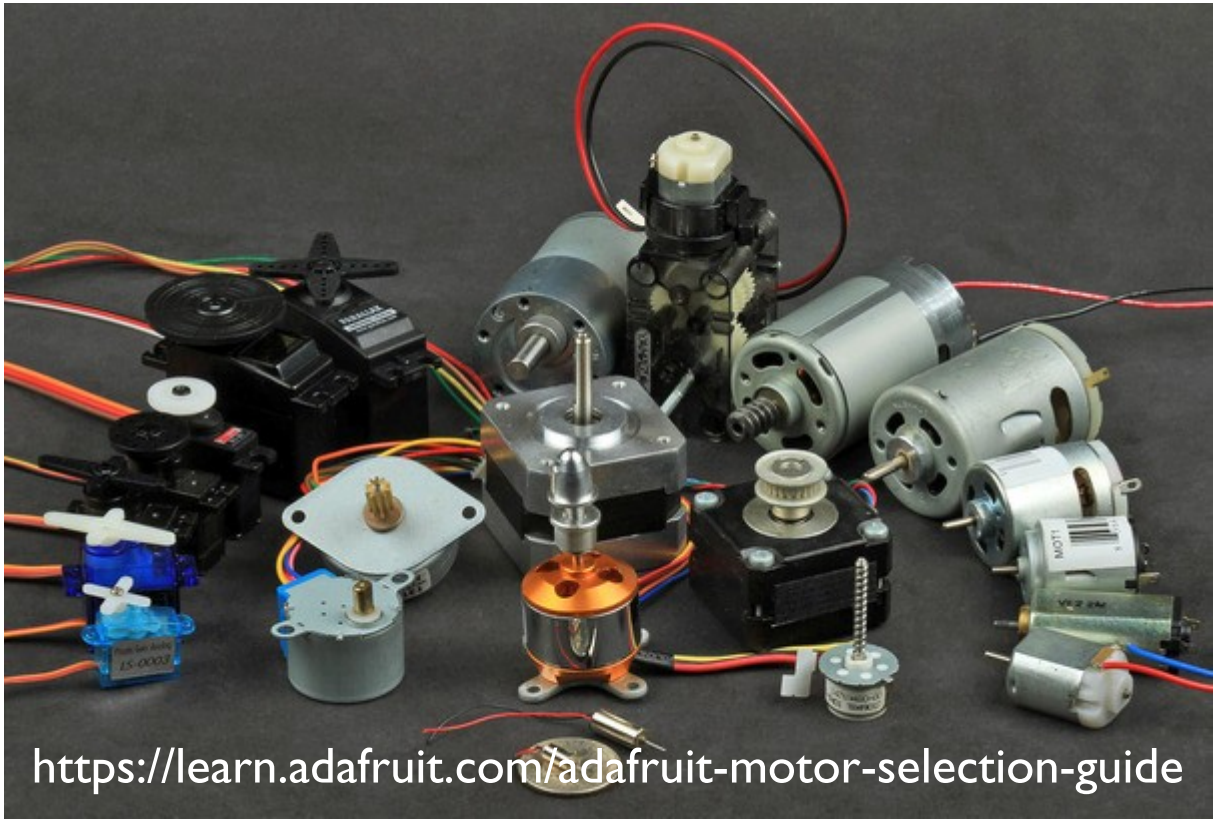
```
$ cd ~/pi-follower-car/01-gpio
$ python pwm_led.py
```

實驗 2-1：馬達驅動

目的：用電晶體電路驅動馬達

馬達種類

- 直流馬達(DC motors)
- 伺服馬達(Servo motors)
- 步進馬達(Stepper motors)

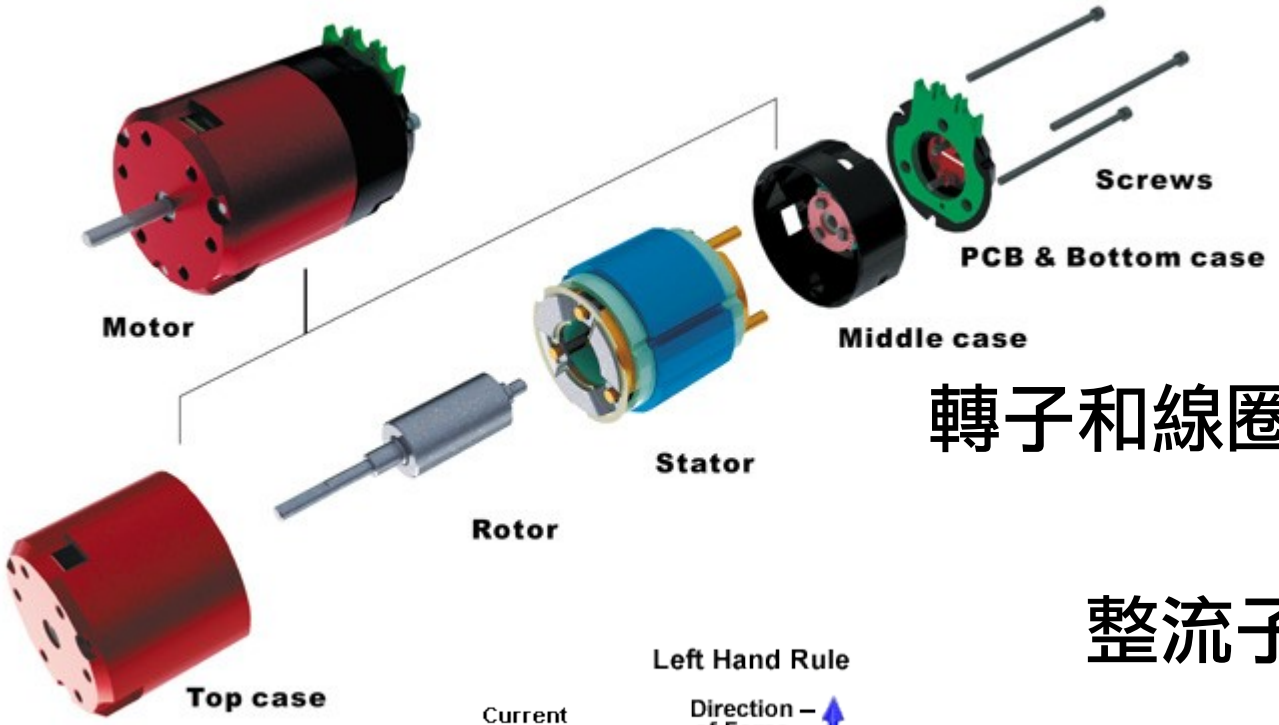


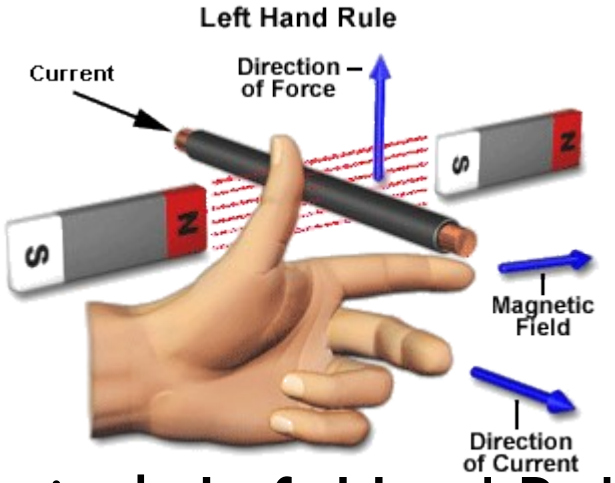
https://learn.adafruit.com/adafruit-motor-selection-guide

	Brushed and Brushles Gearmotors	Stepper Gearmotors	RC Servos	Continuous Rotation Servos
				
Continuous Rotation	☑	☑	☒	☑
Speed Control	★★☆☆☆	★★★★★	★★☆☆☆	★★☆☆☆
Speed Control (w/encoder)	★★★★☆	★★★★★	★★☆☆☆	★★★★☆
Position Control	☒	★★★★★	★★★★☆	☒
Position Control (w/encoder)	★★★★☆	★★★★★	★★★★☆	★★★★☆

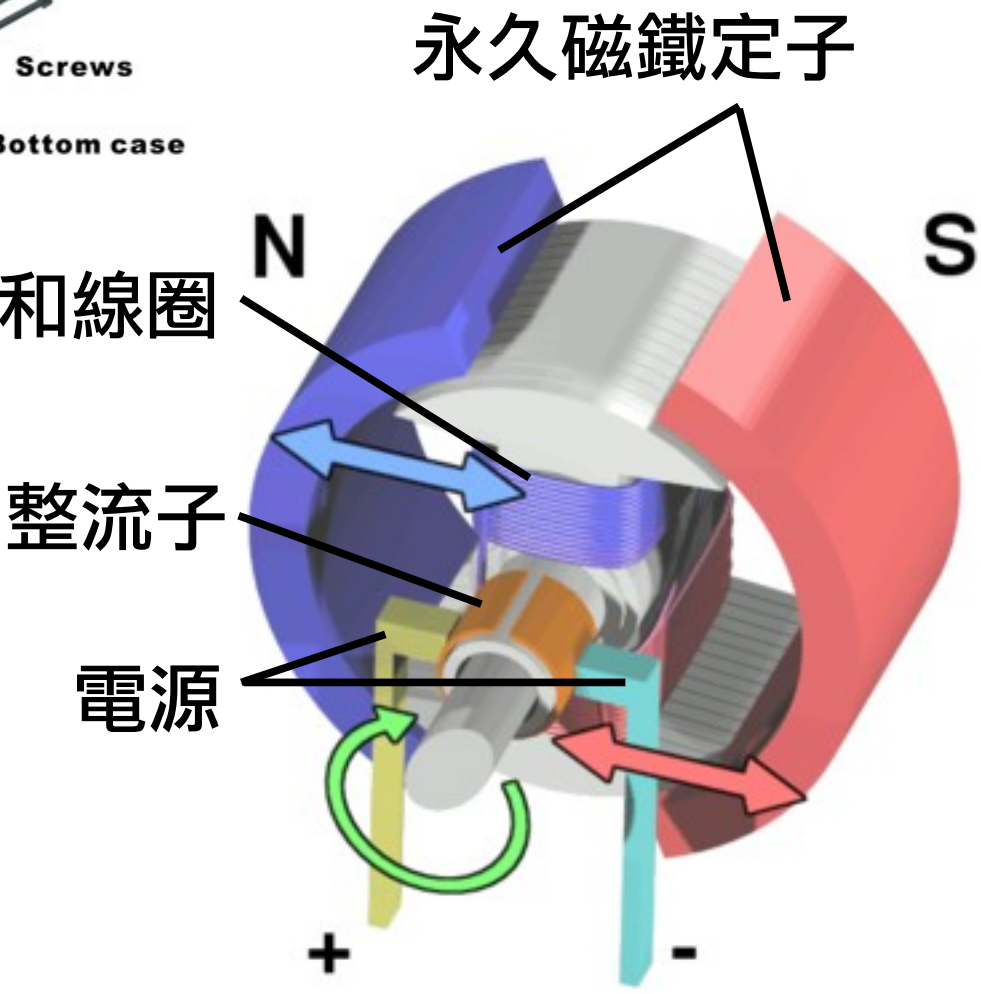
https://learn.adafruit.com/adafruit-motor-selection-guide

直流 (碳刷) 馬達





Fleming's Left Hand Rule



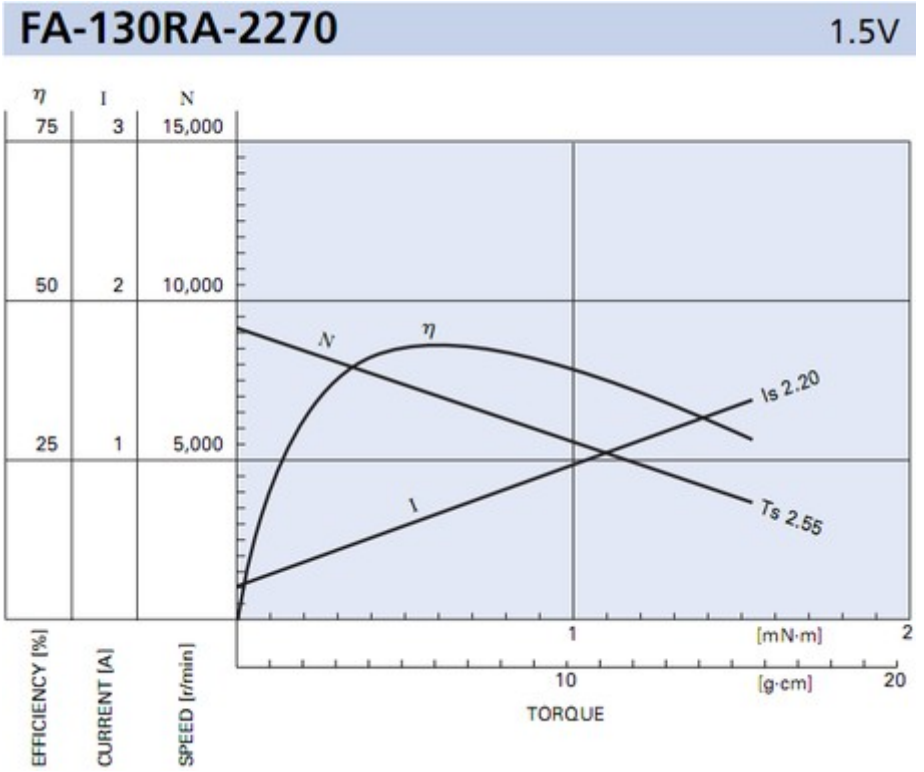
https://learn.adafruit.com/adafruit-motor-selection-guide

FA-130RA 直流馬達規格

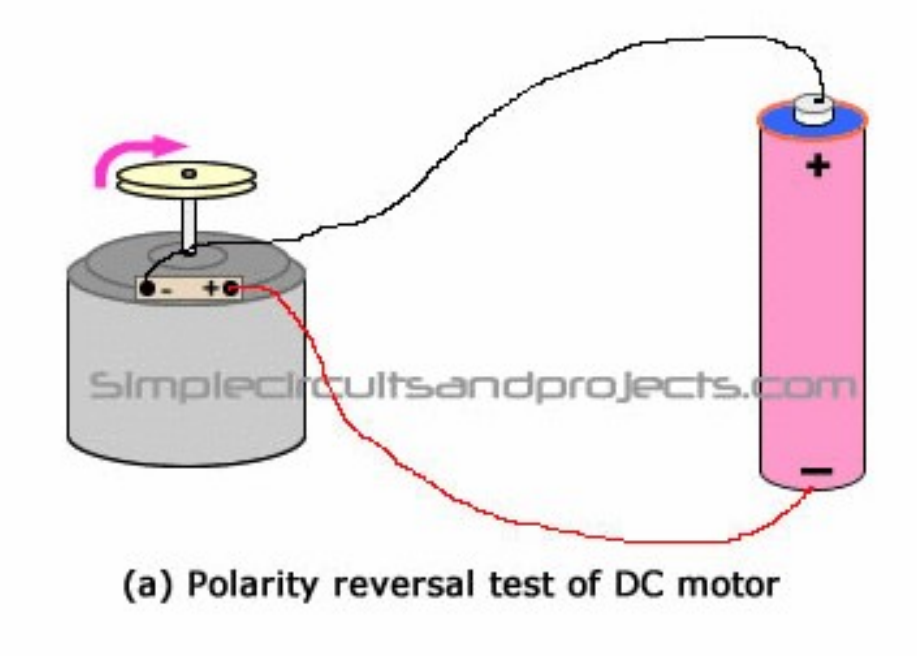
工作電壓　　空載　　負載　　堵轉

MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
FA-130RA-2270	1.5~3.0	1.5V CONSTANT	9100	0.20	6990	0.66	0.59	6.0	0.43	2.55	26	2.20

- 速度和負重的需求（負載）
  - 轉速
  - 扭力（轉矩）
- 電源和控制器的配置
  - 工作電壓
  - 消耗電流
  - 堵轉電流

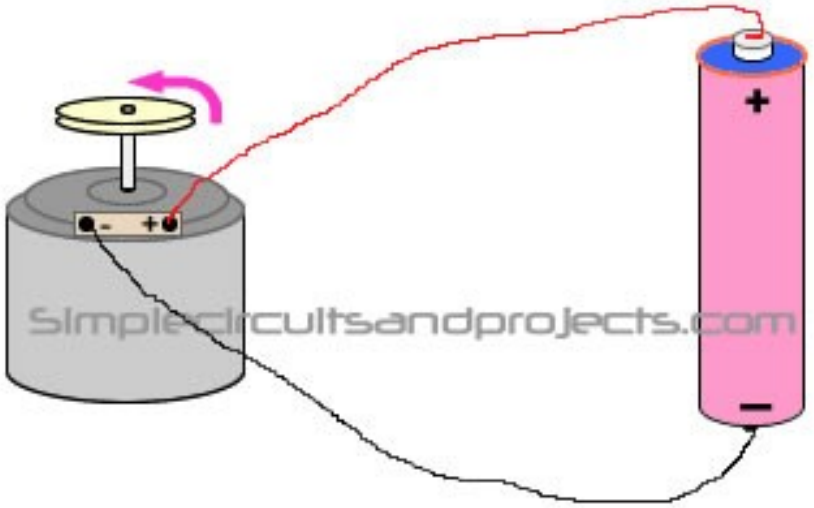


馬達和直流電源直接相接



(a) Polarity reversal test of DC motor

順時鐘旋轉



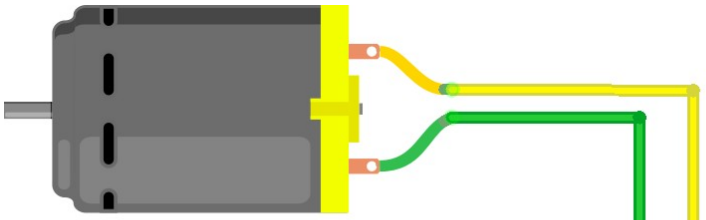
(b) Polarity reversal test of DC motor

逆時鐘旋轉

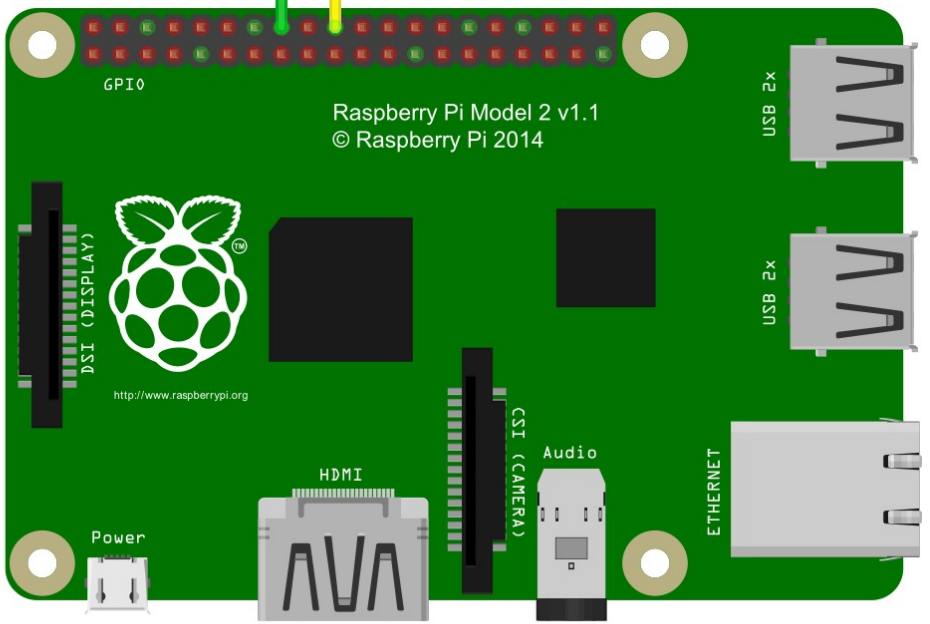
http://www.simplecircuitsandprojects.com/circuits/dc-motor.html

接線圖

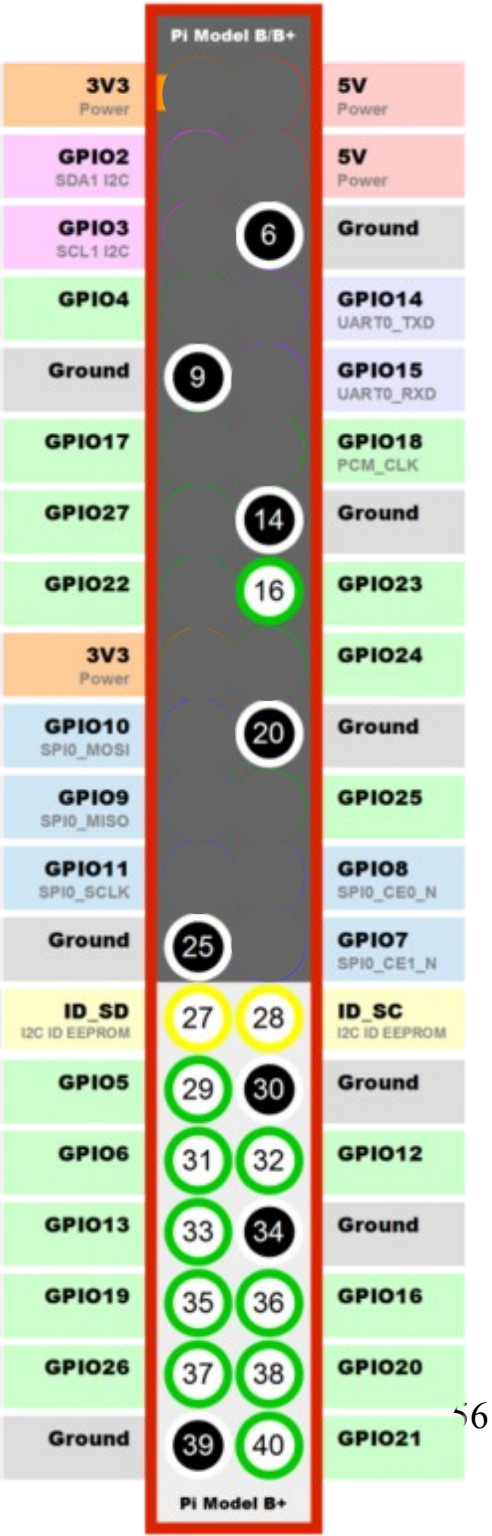
Motor	RPi
馬達 A(Green)	Pin16
馬達 A(Yellow)	Pin20(Ground)



馬達 A



fritzing



轉 - 停 - 轉 - 停 -

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(MOTOR_PIN, GPIO.OUT)
try:
    while True:
        print "Motor start ..."
        GPIO.output(MOTOR_PIN, GPIO.HIGH)
        time.sleep(2)
        print "Motor stop !!!"
        GPIO.output(MOTOR_PIN, GPIO.LOW)
        time.sleep(2)
finally:
    GPIO.cleanup()
```

DEMO
dc_motor.py

```
$ cd ~/pi-follower-car/02-motor
$ python dc_motor.py
```

為什麼不會動？

重新檢視直流馬達規格

工作電壓　　空載　　負載　　堵轉

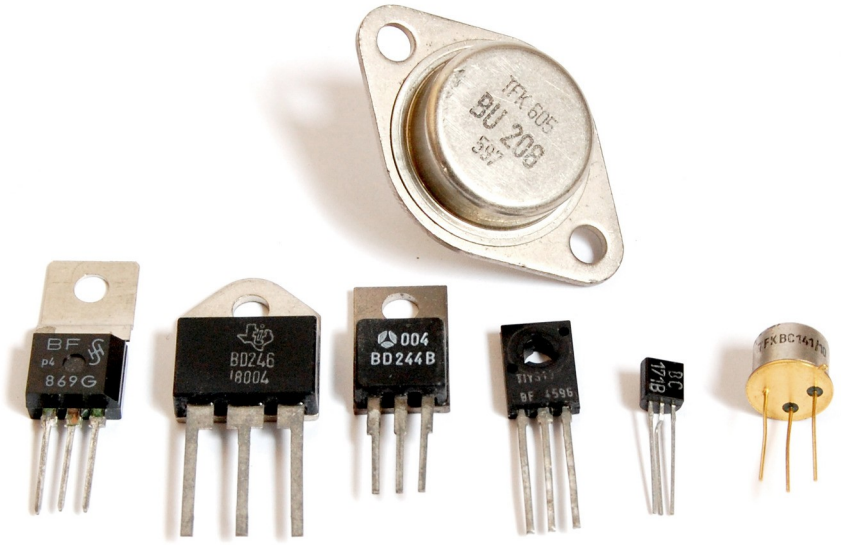
MODEL	VOLTAGE		NO LOAD		AT MAXIMUM EFFICIENCY					STALL		
	OPERATING RANGE	NOMINAL	SPEED	CURRENT	SPEED	CURRENT	TORQUE		OUTPUT	TORQUE		CURRENT
			r/min	A	r/min	A	mN·m	g·cm	W	mN·m	g·cm	A
FA-130RA-2270	1.5~3.0	1.5V CONSTANT	9100	0.20	6990	0.66	0.59	6.0	0.43	2.55	26	2.20

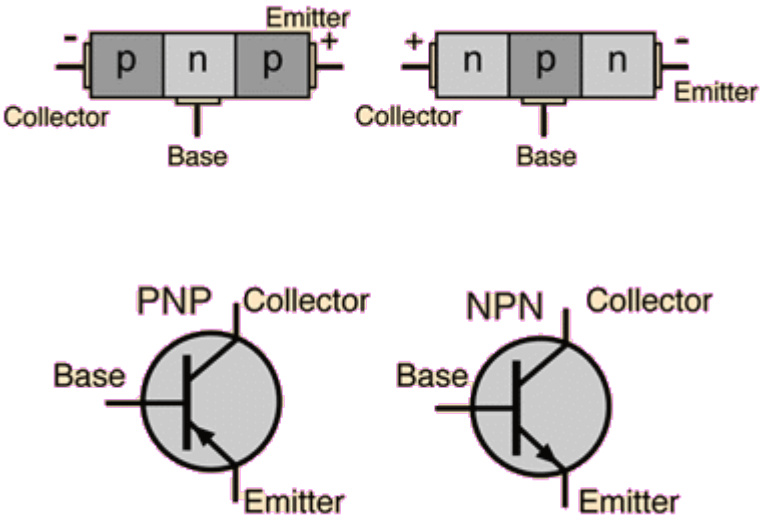
- 電源和控制器的配置
  - 工作電壓
  - 消耗電流
  - 堵轉電流

- Raspberry Pi GPIO 的電流輸出為 2mA 到 16mA

透過電晶體驅動馬達

- 一種可放大，開關，穩壓，信號調節的元件
  - 由於微處理器的輸出微弱，無法驅動大型負載
  - 但可透過電晶體放大訊號，控制外部裝置





常用符號

http://de.wikipedia.org/wiki/Transistor

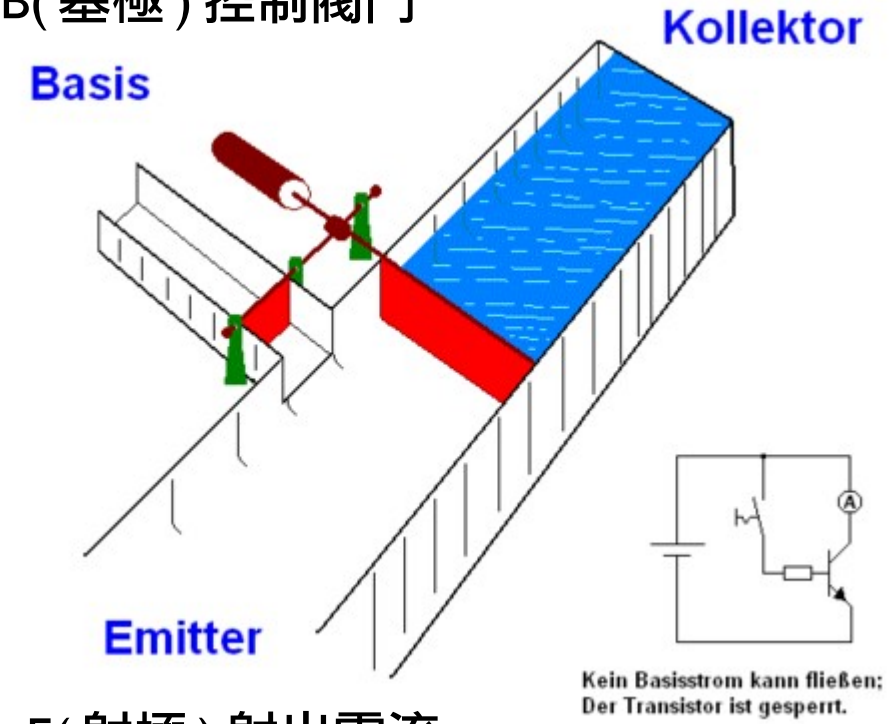
水管閥門功能

腳位意義 (依功能分)

B(基極) 控制閥門

C(集極) 收集電流

E(射極) 射出電流

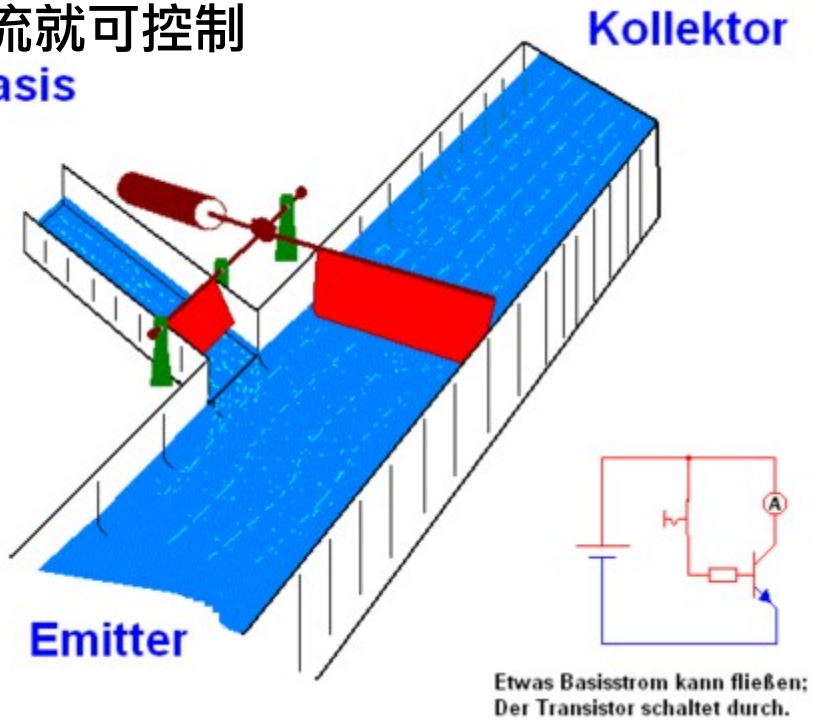


實際運作情形

微小電流就可控制

允許大電流通過

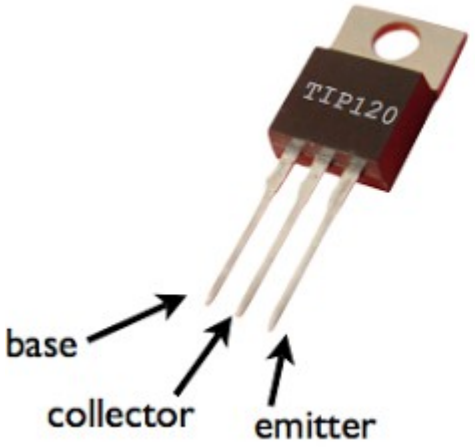
推動大型負載

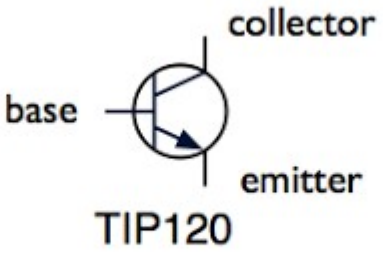


http://commons.wikimedia.org/wiki/File:Transistor_animation.gif

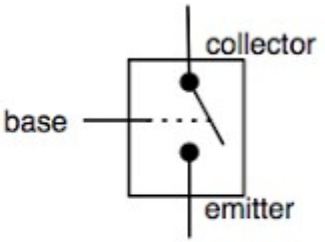
使用 TIP120 做開關

- NPN 型電晶體
- 由基極的訊號控制開關，LOW 斷路，HIGH 開通





schematic symbol

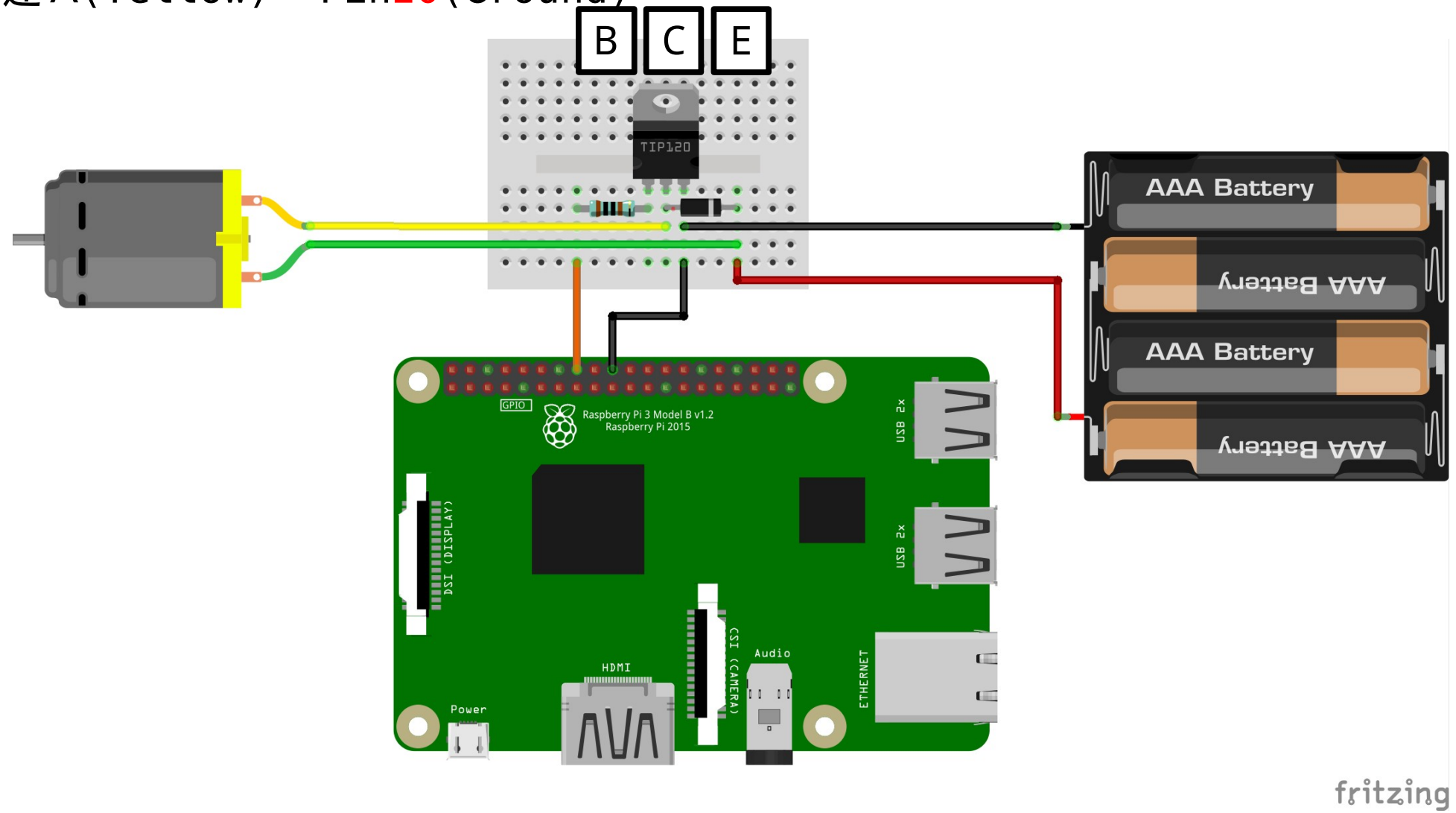


how it kind of works

http://interactive2go.blogspot.tw/2009/05/blog-post.html

接線圖

Motor	RPi
馬達A(Green)	Pin16
馬達A(Yellow)	Pin20(Ground)



dc_motor.py 控制的是電晶體

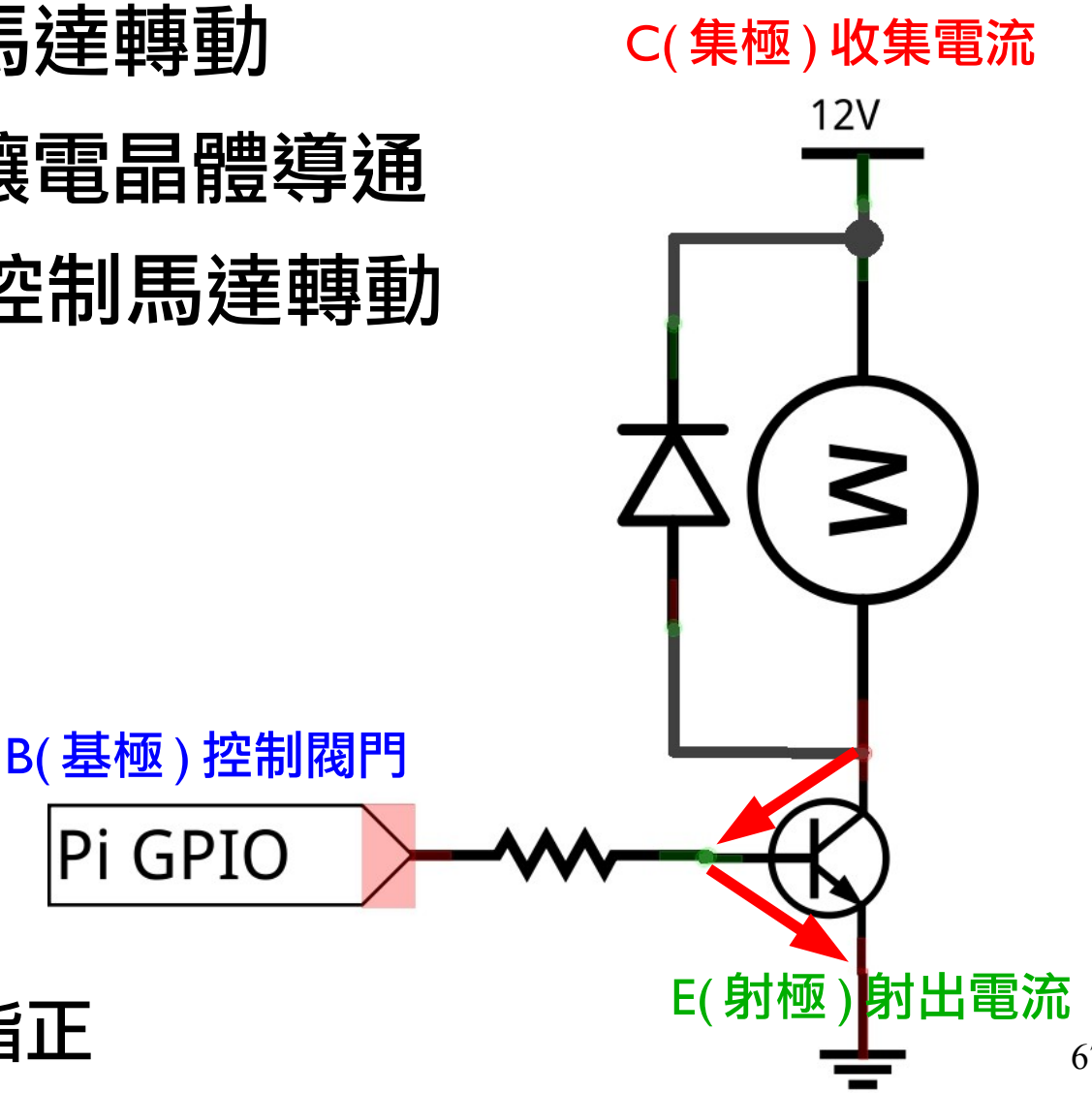
```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(MOTOR_PIN, GPIO.OUT)
try:
    while True:
        print "Motor start ..."
        GPIO.output(MOTOR_PIN, GPIO.HIGH)
        time.sleep(2)
        print "Motor stop !!!"
        GPIO.output(MOTOR_PIN, GPIO.LOW)
        time.sleep(2)
finally:
    GPIO.cleanup()
```

實驗 2-2：控制馬達轉向

目的：瞭解車子移動的原理

回顧剛剛的接線圖

- C 極外部電源提供馬達轉動
- B 極發送控制訊號讓電晶體導通
- 一個電晶體電路可控制馬達轉動

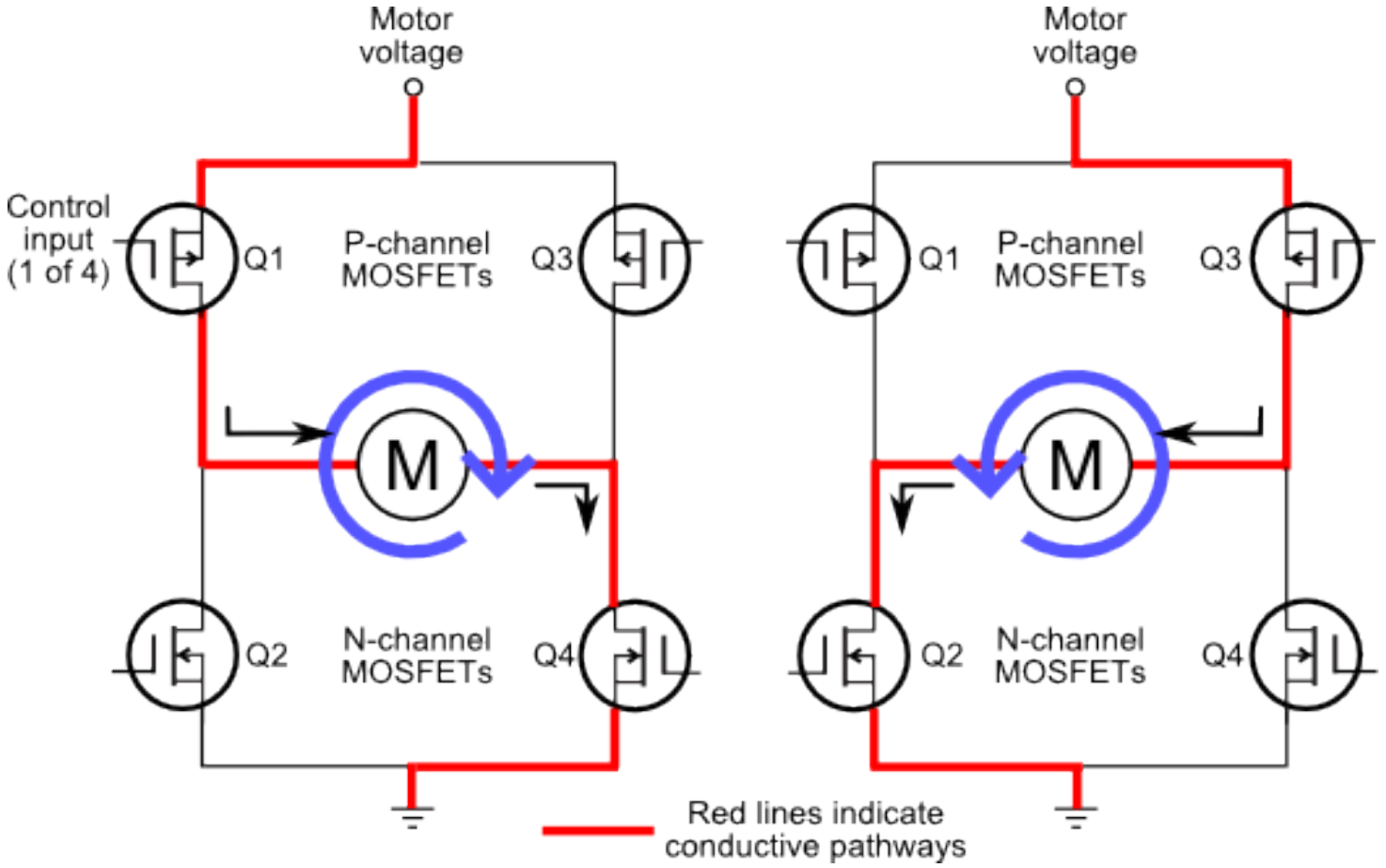


- 感謝 daniel chen 指正

https://www.raspberrypi.org/forums/viewtopic.php?t=34830

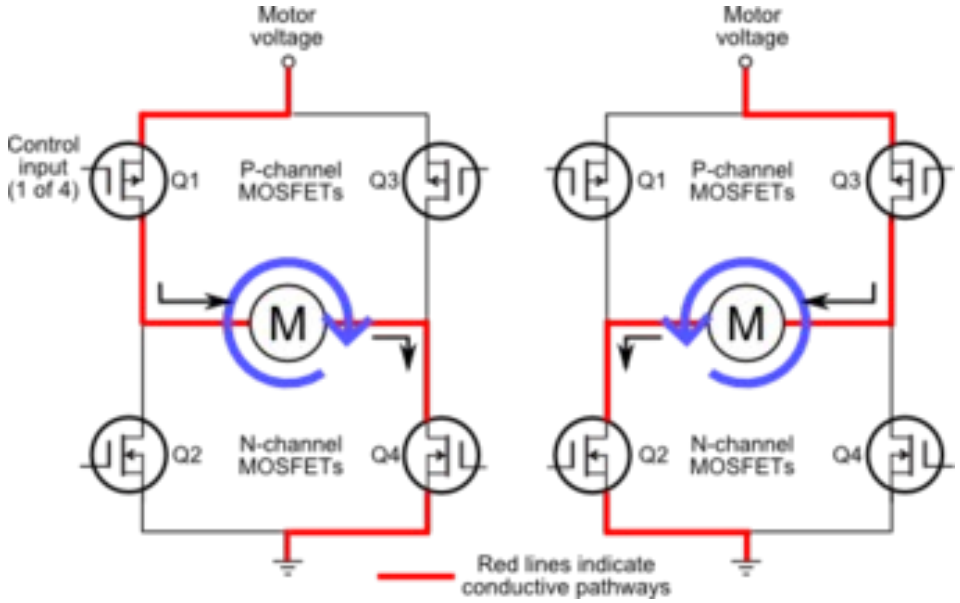
控制馬達轉向 - H 橋式電路

- 每一個節點都是電晶體控制電路



http://www.robotoid.com/my-first-robot/rbb-bot-phase2-part1.html

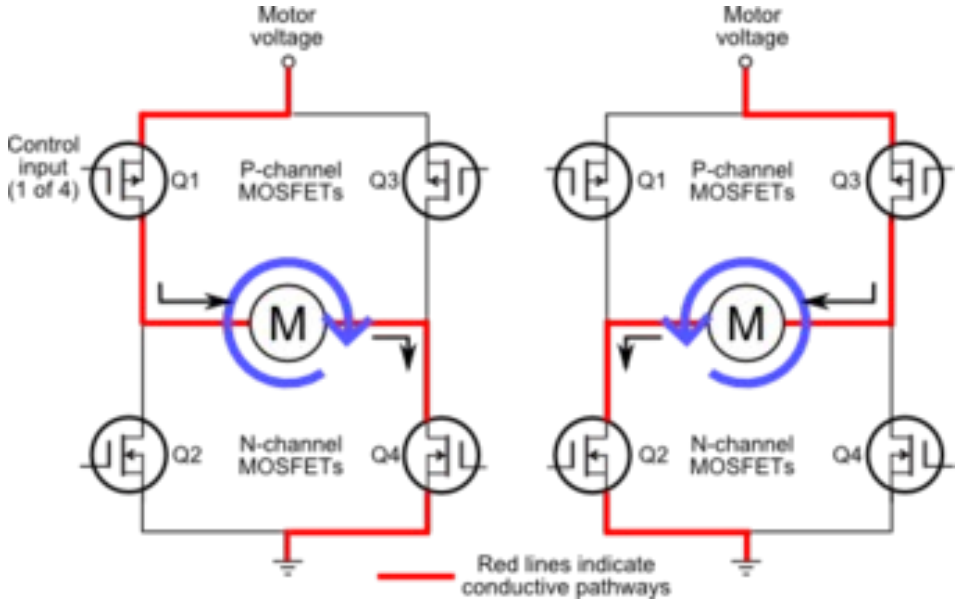
如何用程式控制？



連接	作用
Q1 和 Q4	順時鐘轉動
Q3 和 Q2	逆時鐘轉動
Q1 和 Q3	煞車
無	慣性自由轉動
Q1 和 Q2	短路！
Q3 和 Q4	短路！

http://www.robotoid.com/my-first-robot/rbb-bot-phase2-part1.html

如何用程式控制？



連接	作用
Q1 和 Q4	順時鐘轉動
Q3 和 Q2	逆時鐘轉動
Q1 和 Q3	煞車
無	慣性自由轉動
Q1 和 Q2	短路！
Q3 和 Q4	短路！

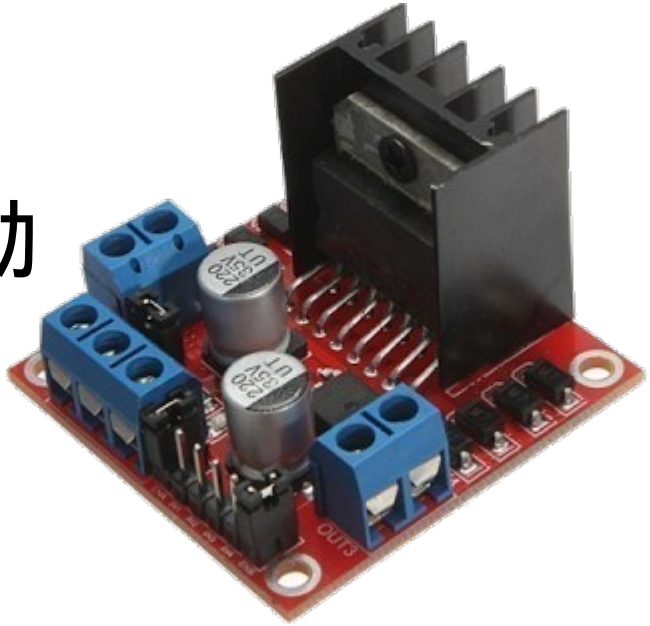
可是…

1. 不小心給錯電就會把電晶體燒掉
2. 一個馬達要搭配四個電晶體電路
3. 我想要一次控制兩個馬達的轉動

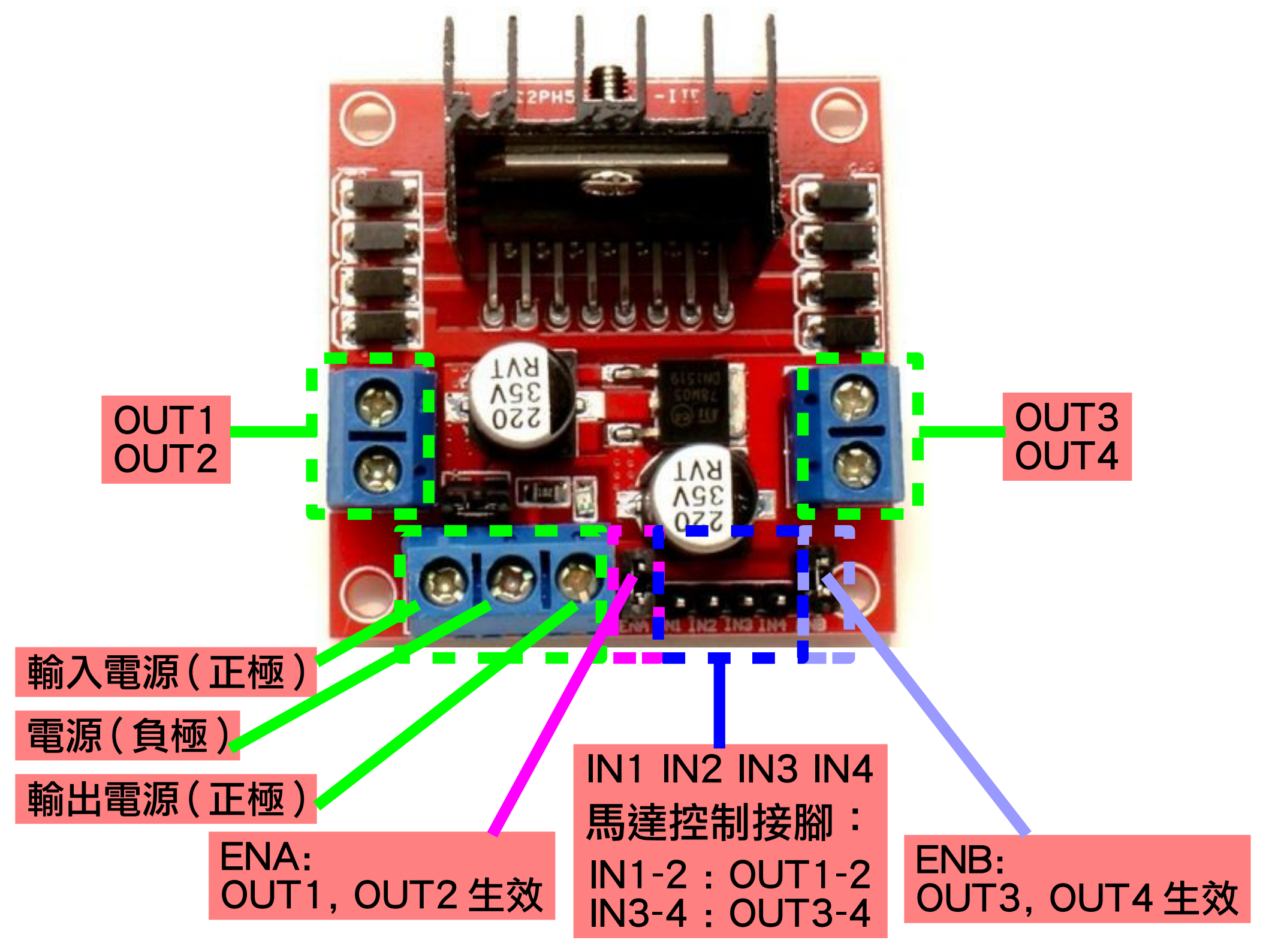
http://www.robotoid.com/my-first-robot/rbb-bot-phase2-part1.html

L298N 馬達控制板

- 馬達驅動晶片：L298N
  - 雙 H 橋式直流（步進）馬達驅動
  - 最大馬達驅動電壓：DC 50V



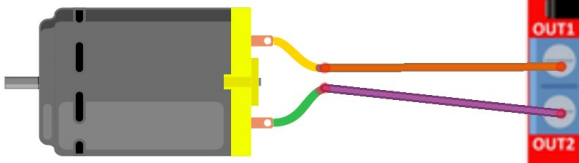
- 降壓晶片：78M05
  - 將外部輸入電源降壓為 DC 5V 給電路板
  - 外部輸入電源電壓限制範圍：DC 7V – DC 35V

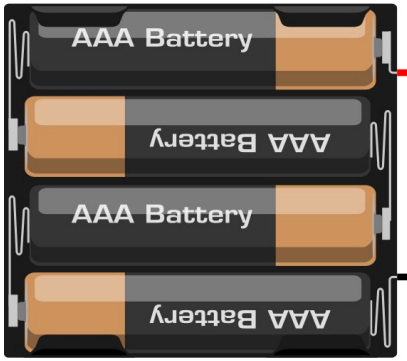

OUT1
OUT2
OUT3
OUT4
輸入電源(正極)
電源(負極)
輸出電源(正極)
ENA:
OUT1, OUT2 生效
IN1 IN2 IN3 IN4
馬達控制接腳：
IN1-2：OUT1-2
IN3-4：OUT3-4
ENB:
OUT3, OUT4 生效

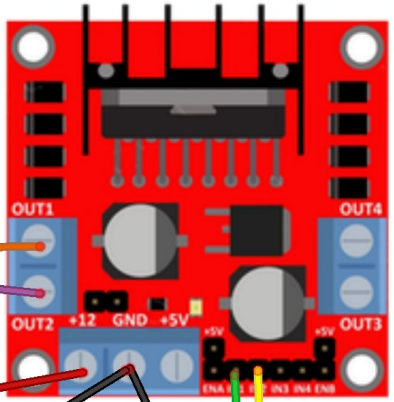
單輪接線圖
(L298N 要和 Pi 共接地)

Motor	L298N
馬達 A(Yellow)	OUT1(Yellow)
馬達 A(Green)	OUT2(Green)

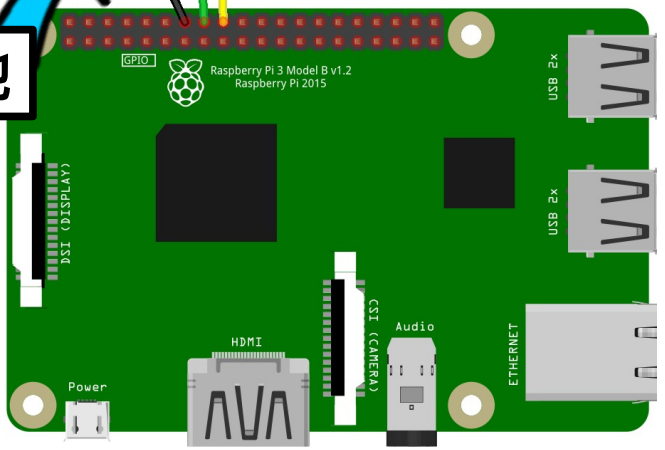
馬達 A



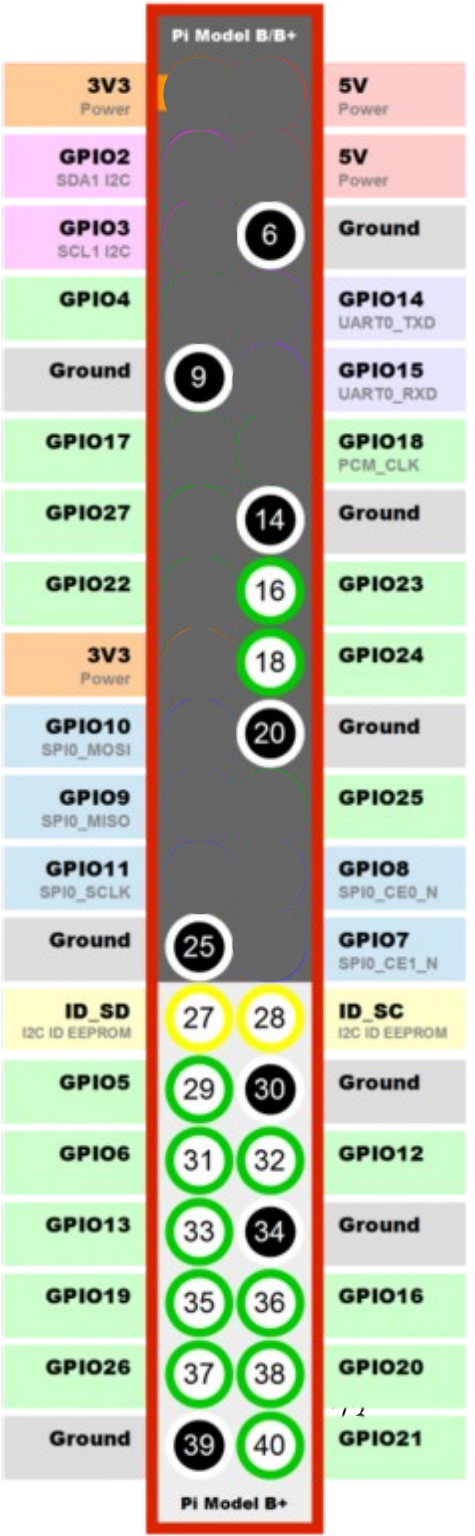




共接地



L298N	RPi
IN1(Green)	Pin16
IN2(Yellow)	Pin18



控制單輪正反轉

```
Motor_R1_Pin = 16
Motor_R2_Pin = 18
GPIO.setmode(GPIO.BOARD)
GPIO.setup(Motor_R1_Pin, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(Motor_R2_Pin, GPIO.OUT, initial=GPIO.LOW)
try:
    GPIO.output(Motor_R1_Pin, GPIO.HIGH)     # clockwise
    time.sleep(1)
    GPIO.output(Motor_R1_Pin, GPIO.LOW)
    time.sleep(1)                            # protect motor
    GPIO.output(Motor_R2_Pin, GPIO.HIGH)     # counterclockwise
    time.sleep(1)
    GPIO.output(Motor_R2_Pin, GPIO.LOW)
finally:
    GPIO.cleanup()
```

DEMO
l298n_motor.py

```
$ cd ~/pi-follower-car/02-motor
$ python l298n_motor.py
```

實驗 2-3：控制馬達轉速

目的：應用 PWM 技術調整馬達轉速

直流馬達轉速由輸入電壓決定

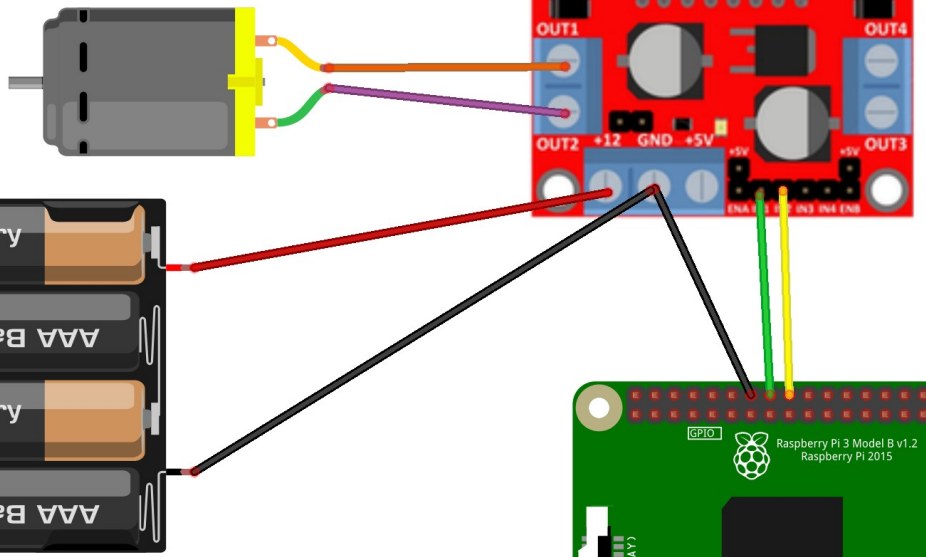
- 改變電壓的方式
  - 增加（減少）電池
  - 增加負載（利用可變電阻）
  - 脈寬調變（PWM）技術

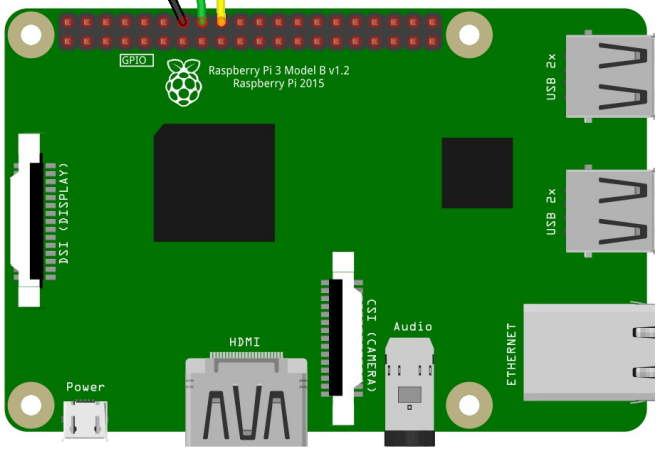
一樣的接線圖

Motor
馬達 A(Yellow)
馬達 A(Green)

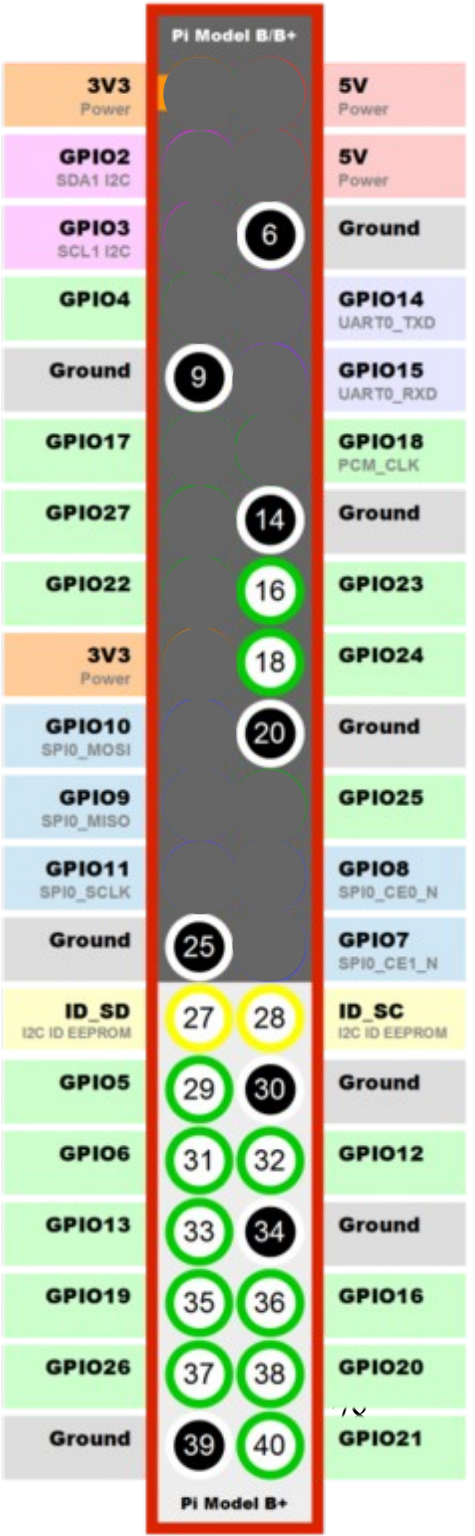
L298N
OUT1(Yellow)
OUT2(Green)

馬達 A





L298N **RPi**
IN1(Green) Pin16
IN2(Yellow) Pin18

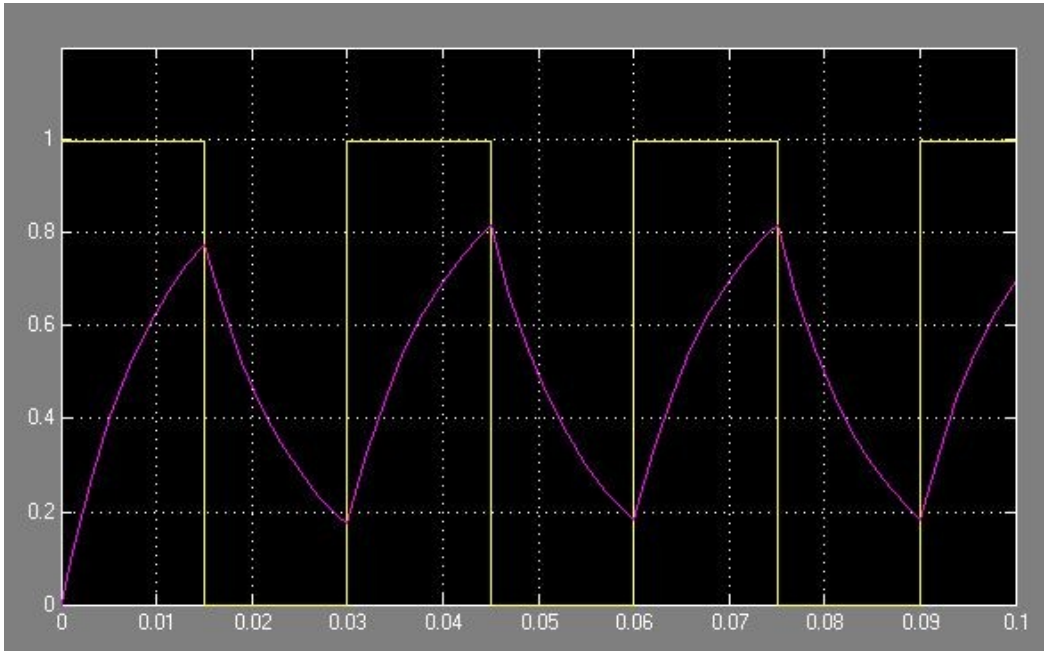


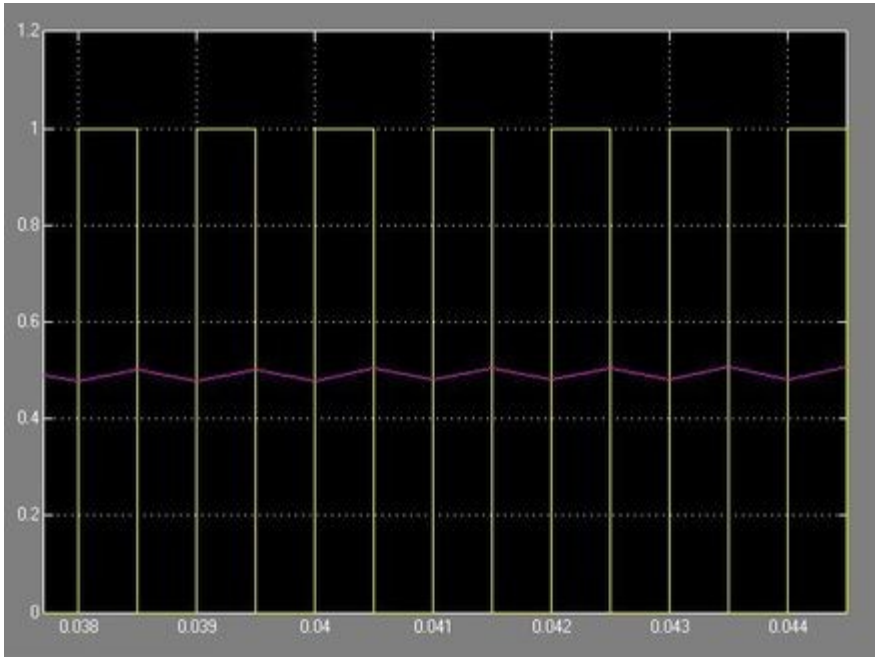
使用 PWM 調整輸出電壓

```
PWM_PIN = 16
GPIO.setup(PWM_PIN, GPIO.OUT)
pwm = GPIO.PWM(PWM_PIN, 600)
pwm.start(0)

try:
    while True:
        duty_s = raw_input("Enter Duty Cycle (0 to 100):")
        duty = int(duty_s)

        if duty >= 0 and duty <=100 :
            pwm.ChangeDutyCycle(duty)

except KeyboardInterrupt:
    pwm.stop()
    GPIO.cleanup()
```

PWM 頻率對馬達的影響

- 不能太高 (>20KHz)，容易產生共振
- 不能太低 (<100Hz)，馬達無法正確響應





https://electronics.stackexchange.com/questions/242293/

DEMO
pwm_l298n.py

```
$ cd ~/pi-follower-car/02-motor
$ python pwm_l298n.py
```

實驗 2-4：遙控車

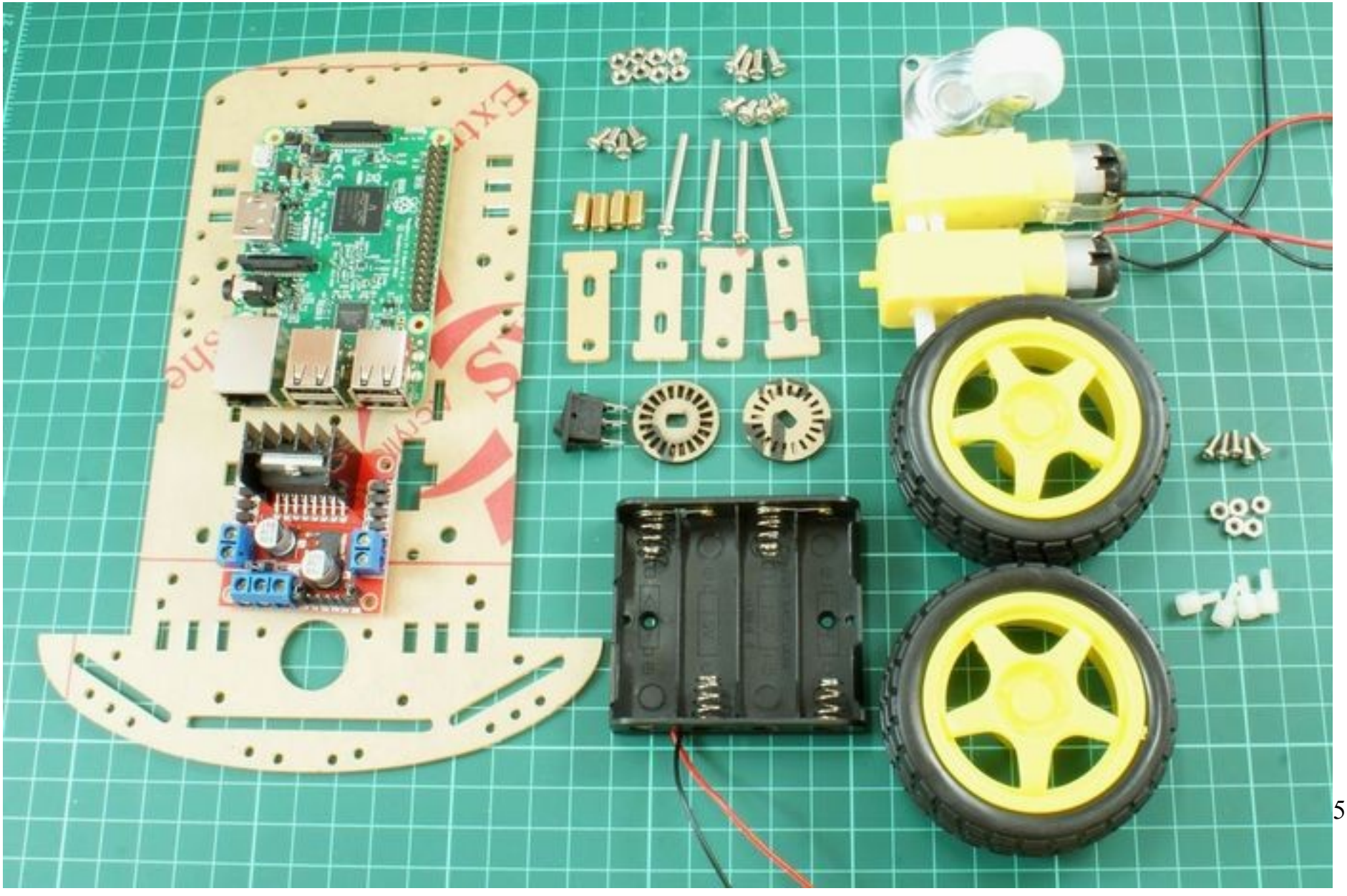
目的：瞭解小車的各項機構

遙控車的組成

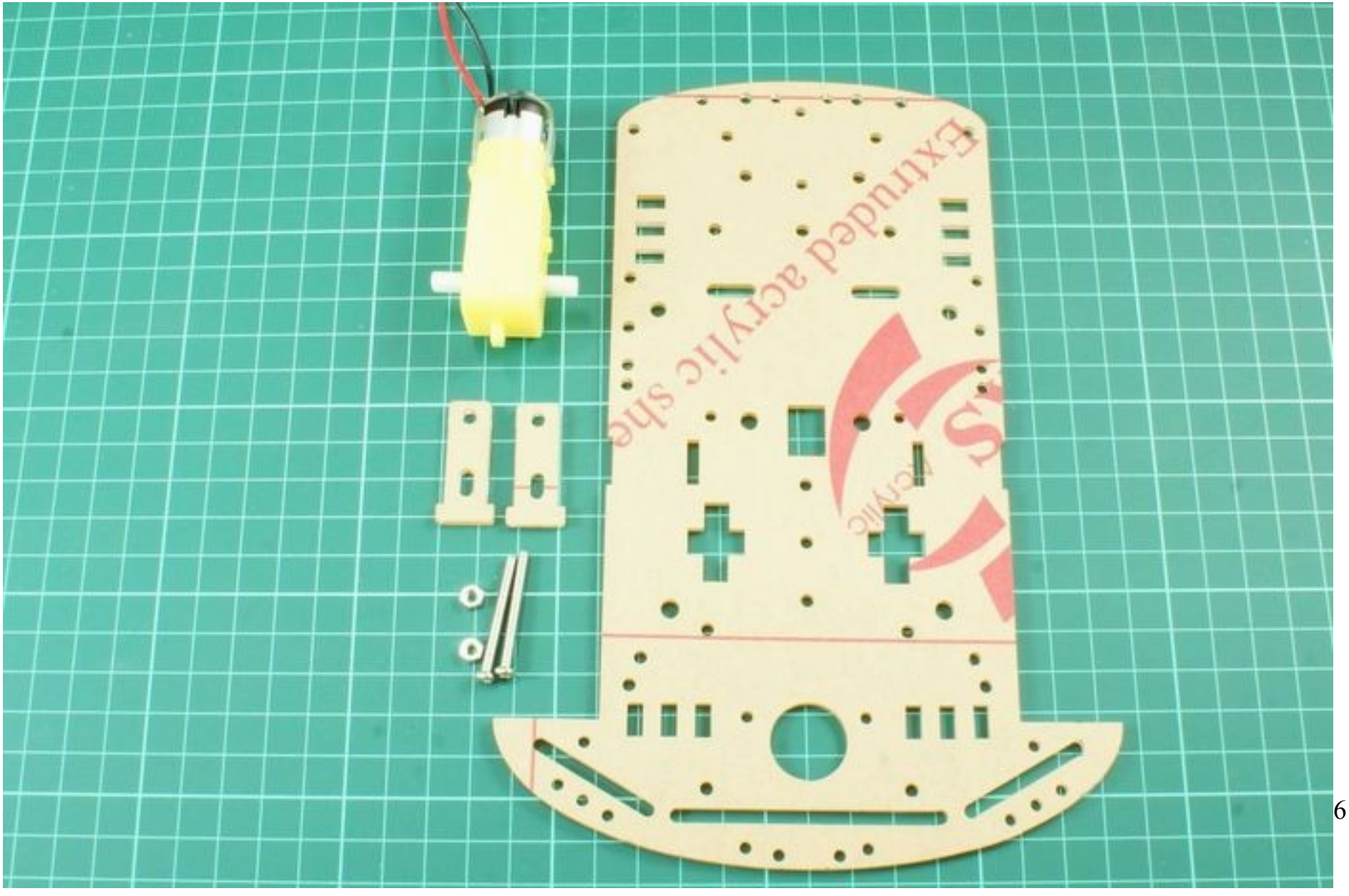
- 車體（機構）
- 馬達 / 車輪（動力）
- 控制板 / 微處理器（邏輯）
- 無線網路 / 紅外線 / 藍牙（傳輸控制）

先動手組裝車體吧！

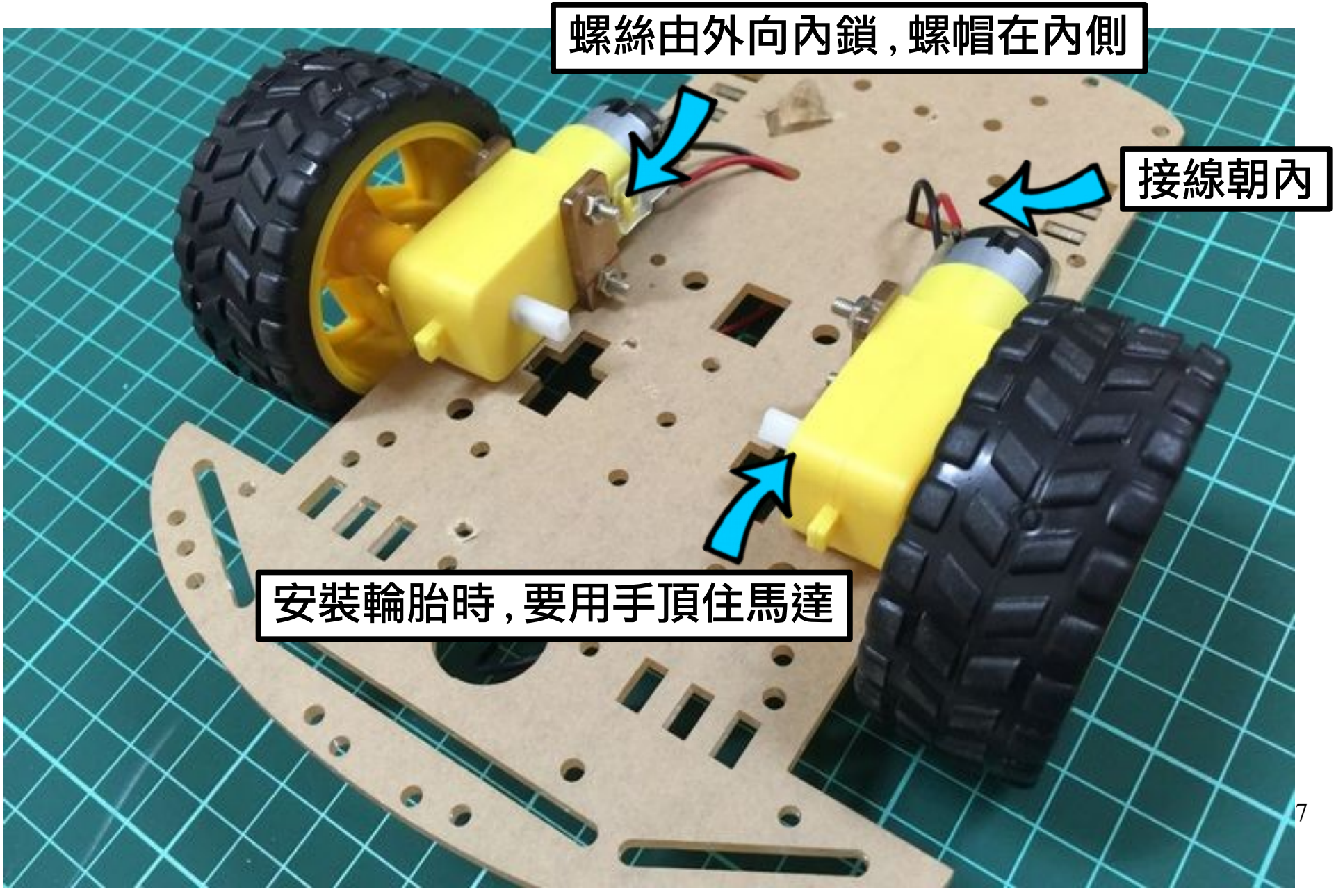
所有零件 (含 Pi 和 L298N)



馬達與固定螺絲

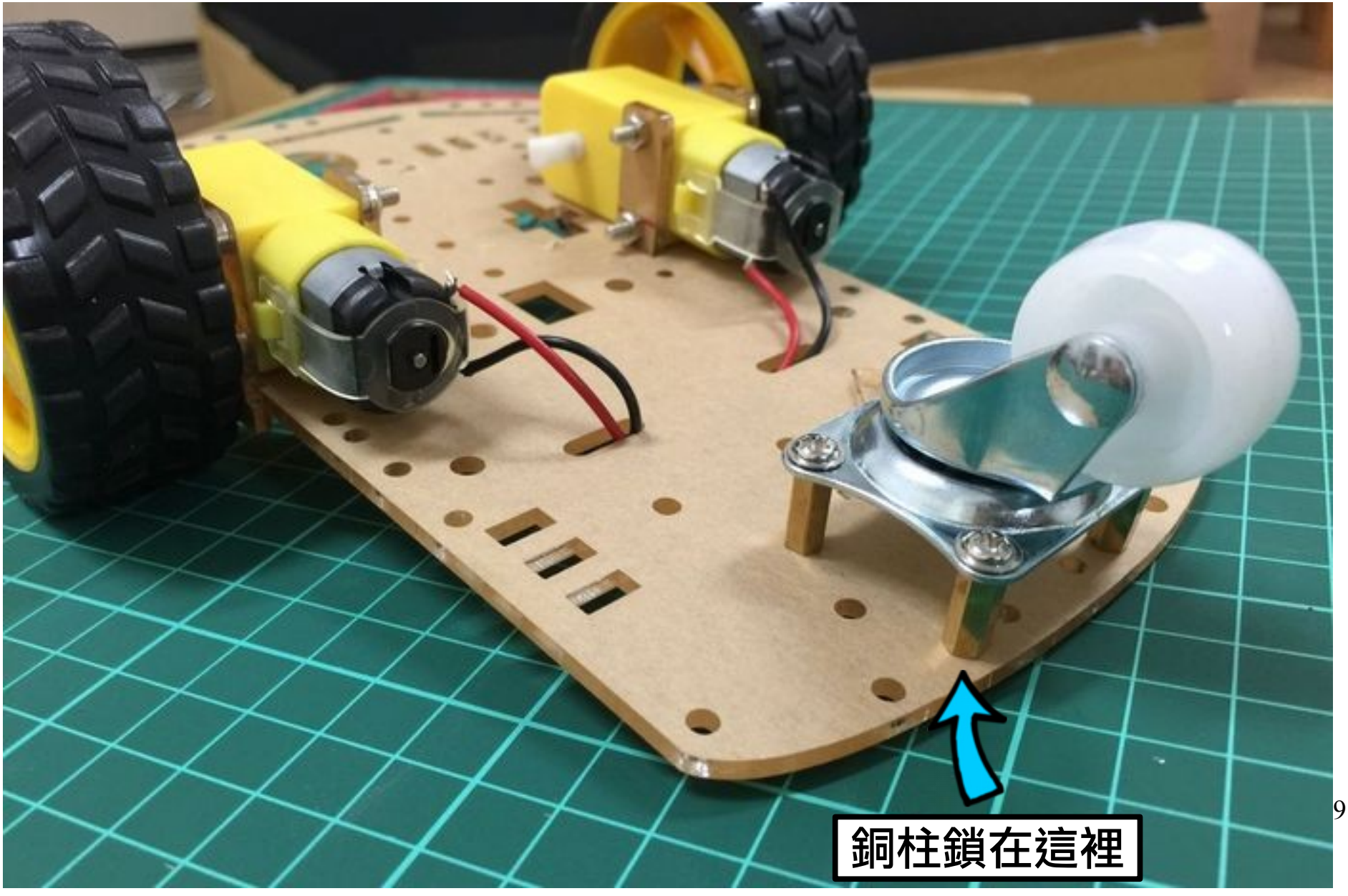


將馬達固定在車體上

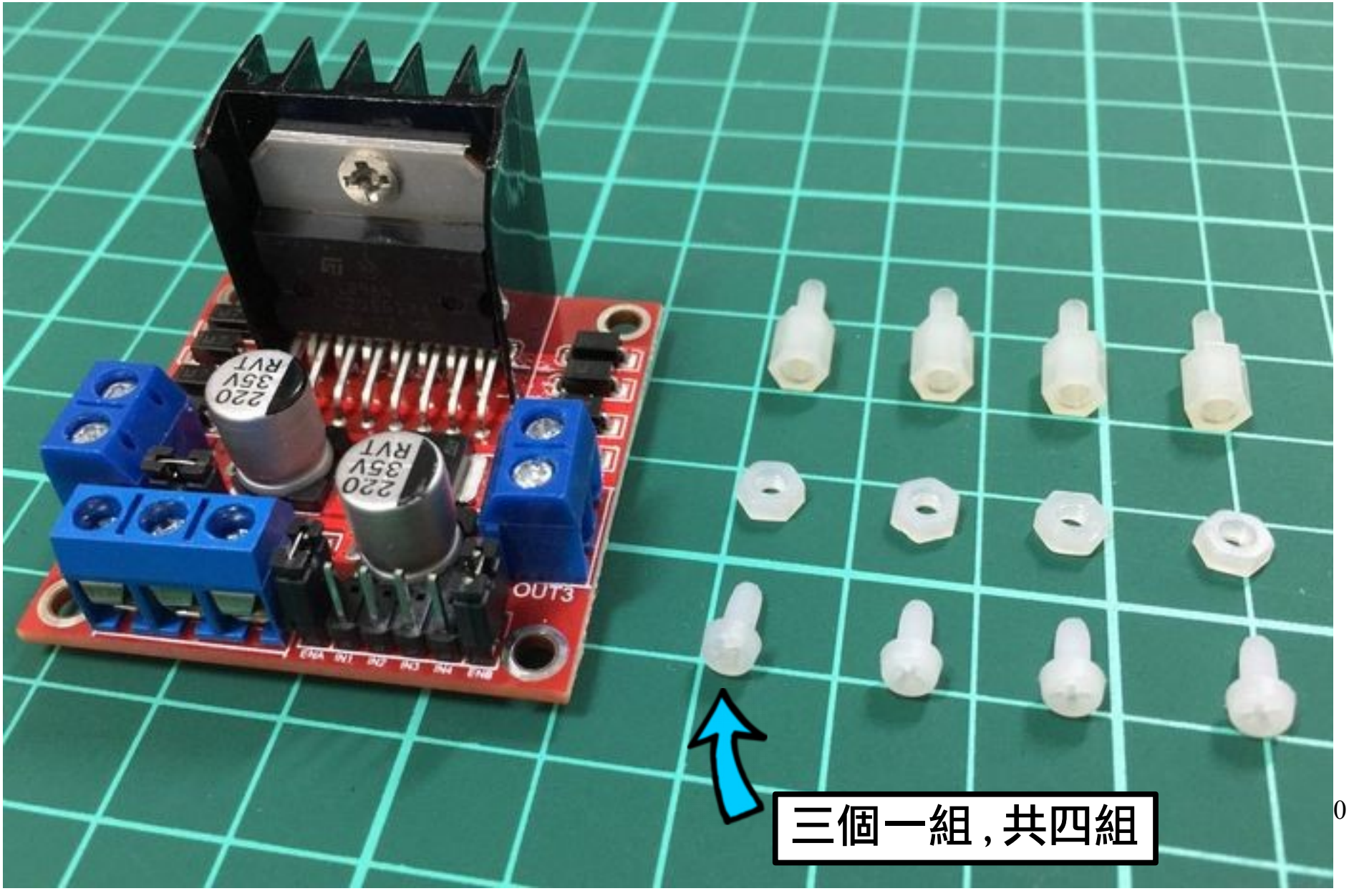


萬向輪和固定螺絲

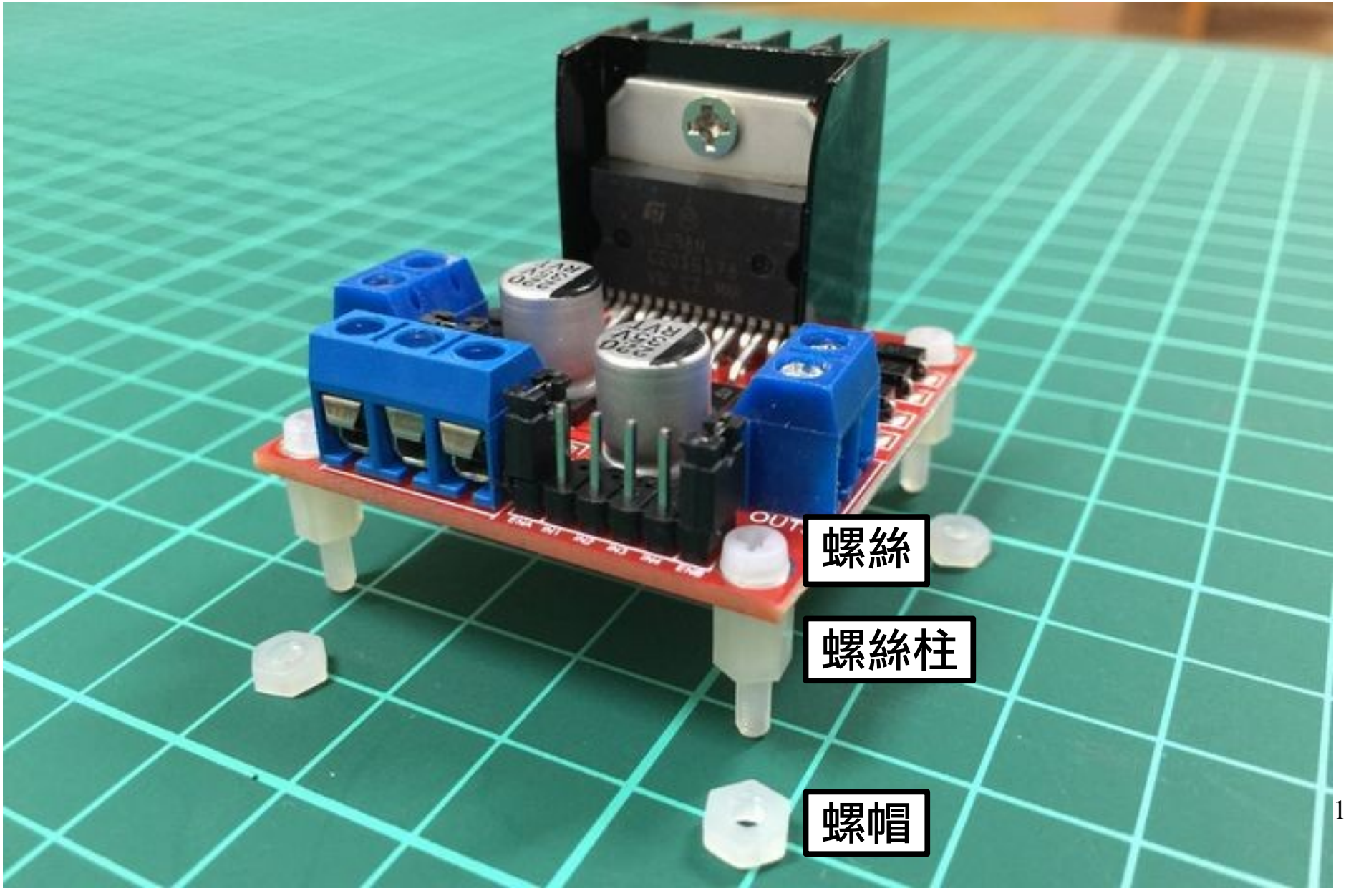
固定在車體上



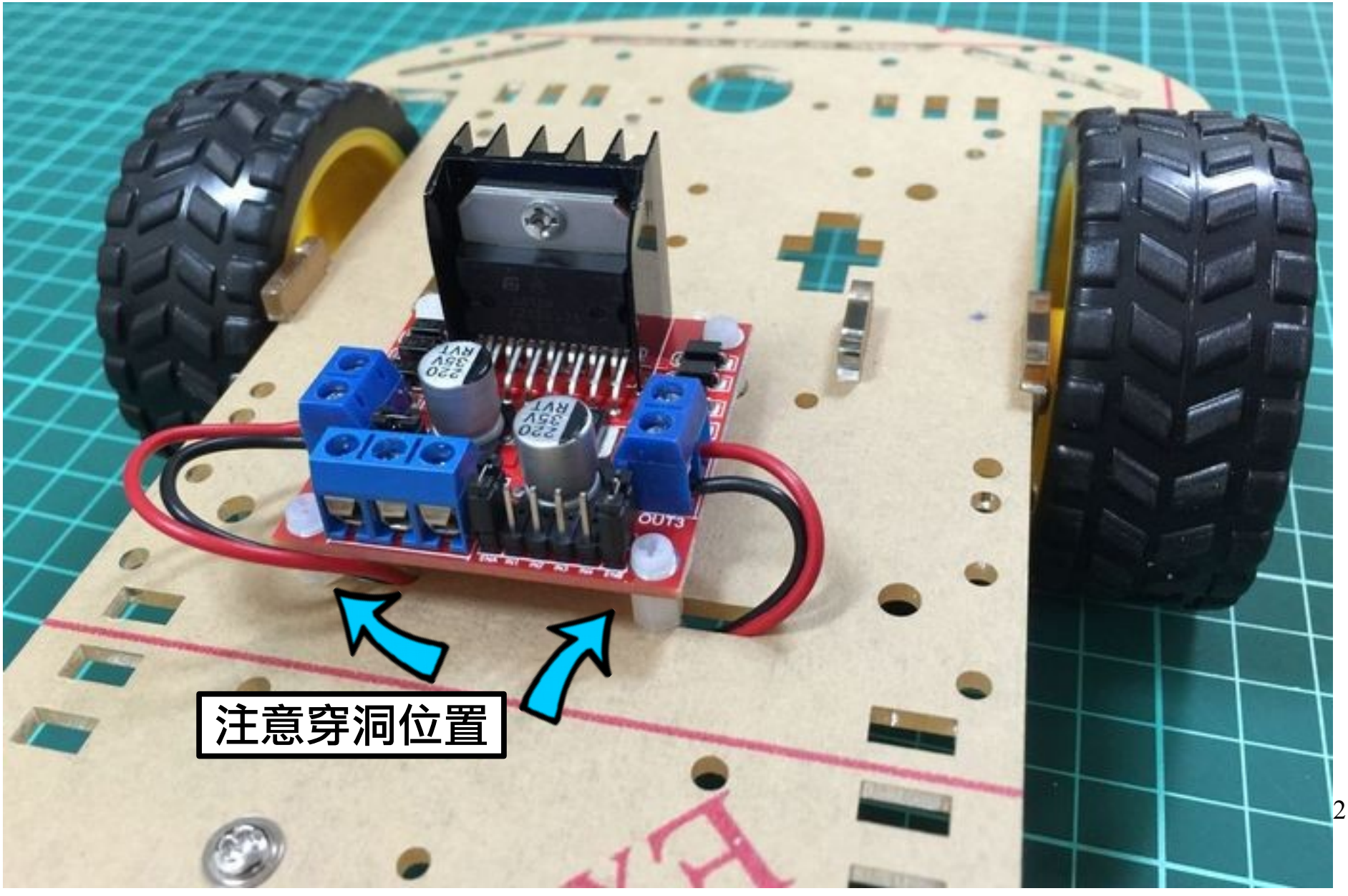
L298N 和固定螺絲



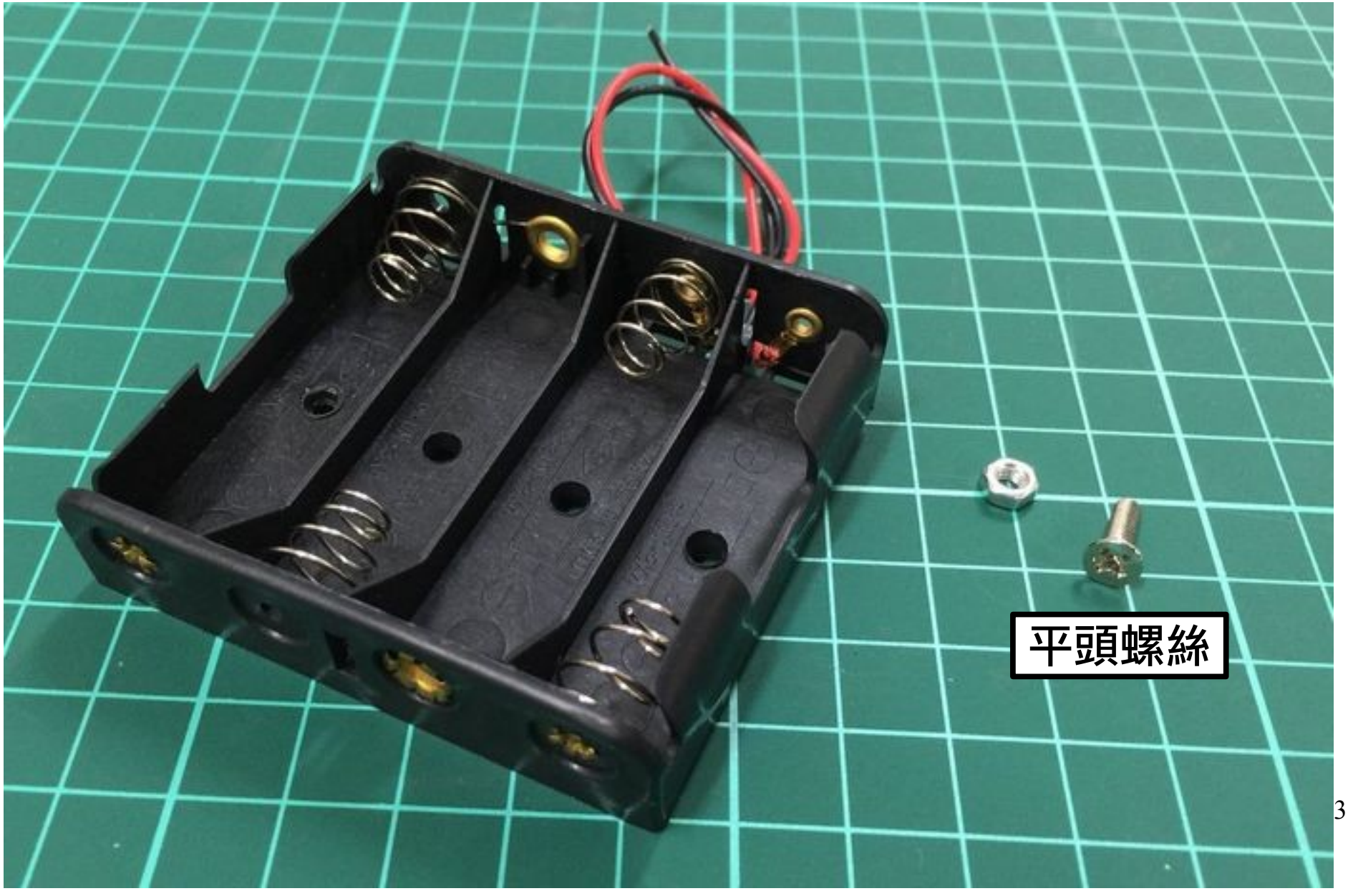
將螺絲和螺絲柱鎖在 L298N

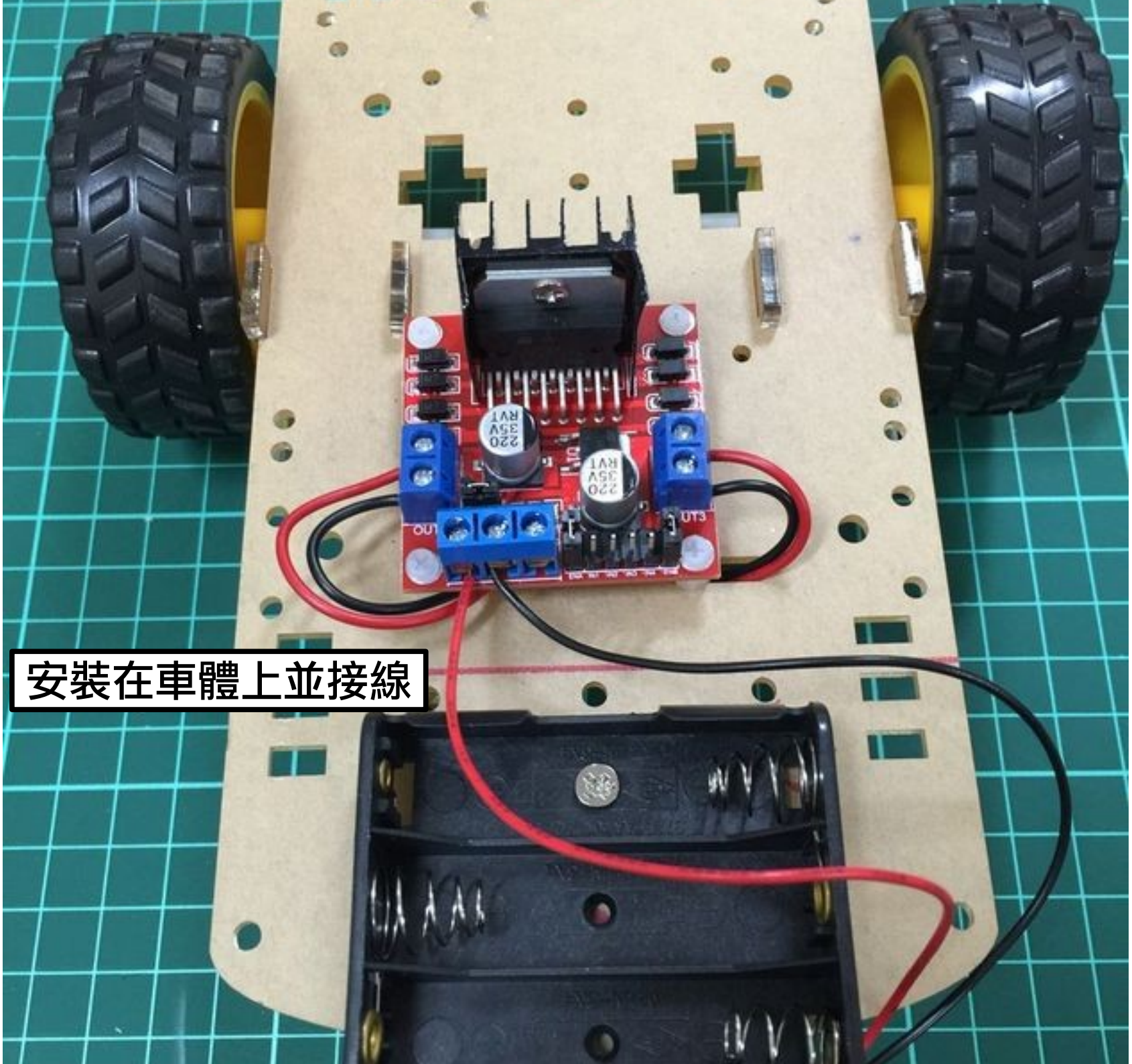


固定在車體上 + 馬達接線

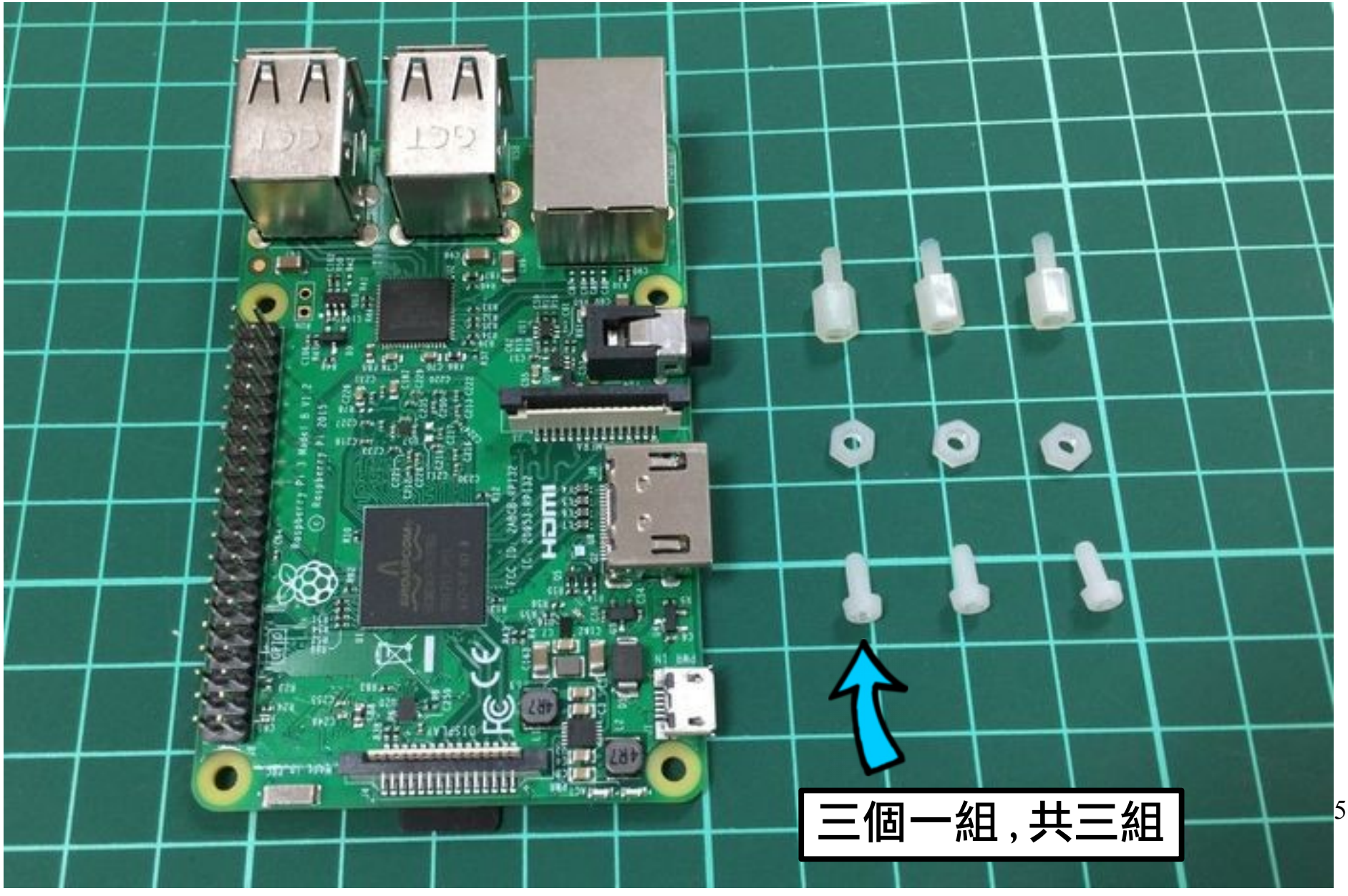


電池盒和固定螺絲

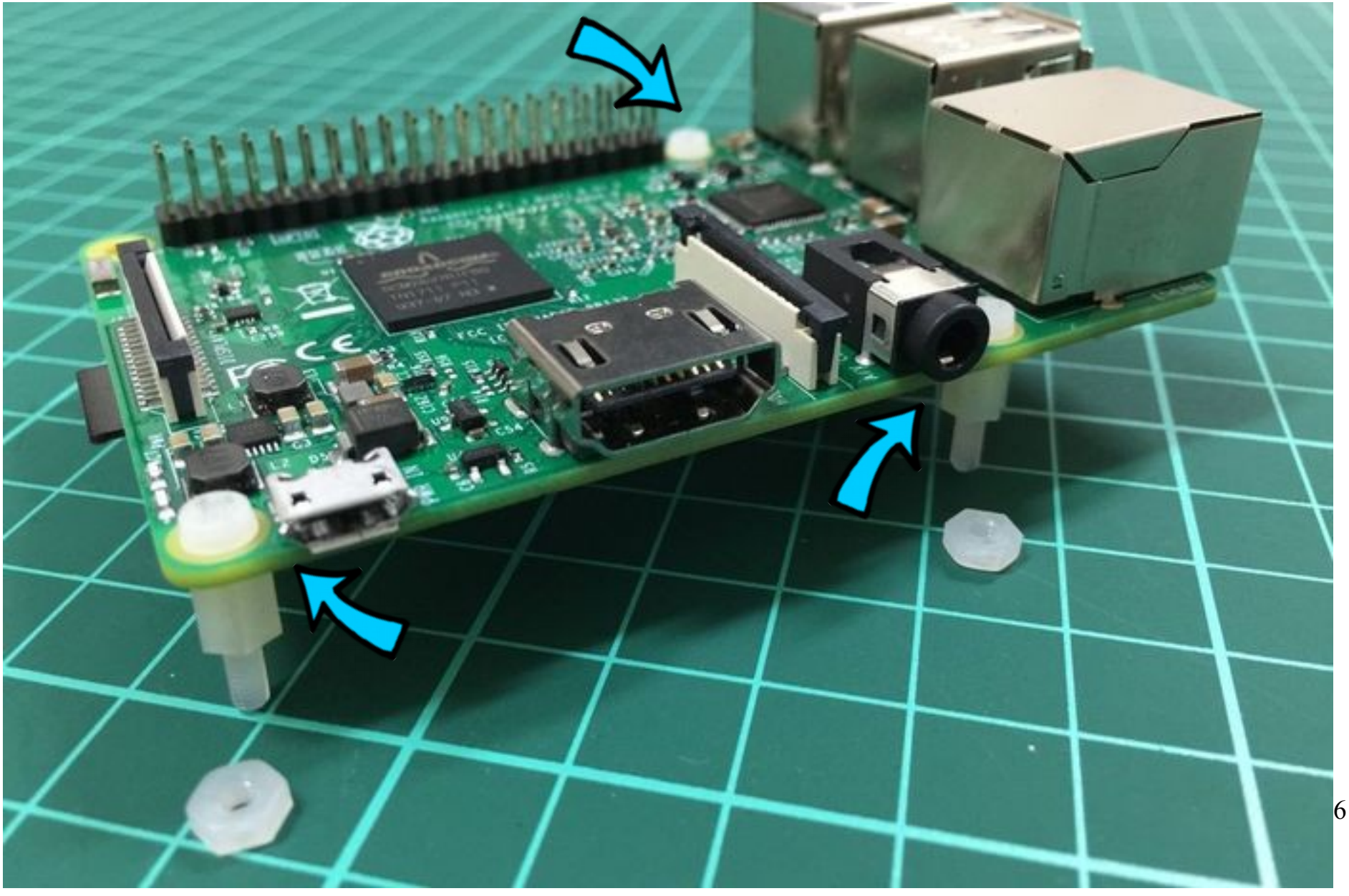



安裝在車體上並接線

Pi 和固定螺絲

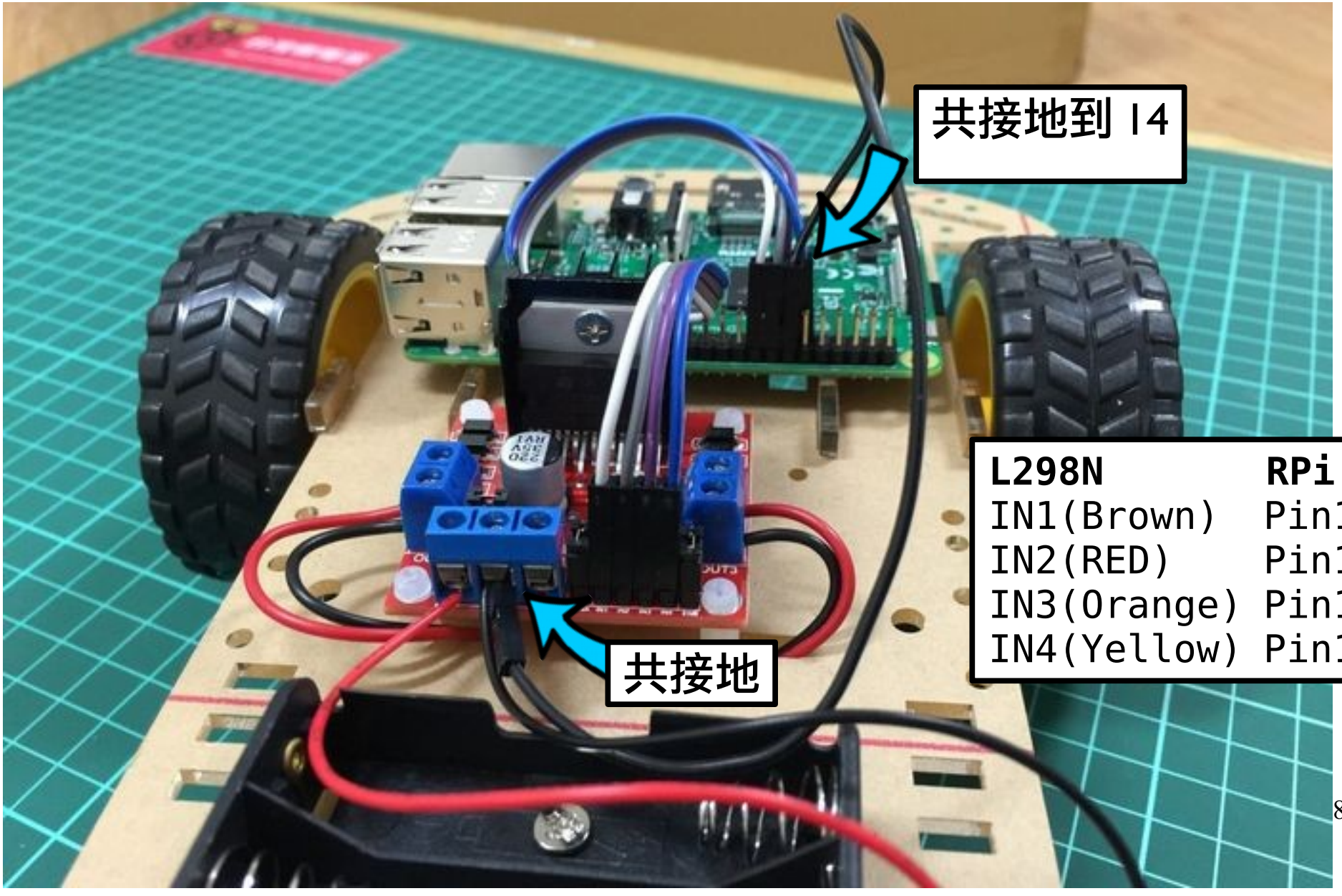


固定螺絲鎖三處

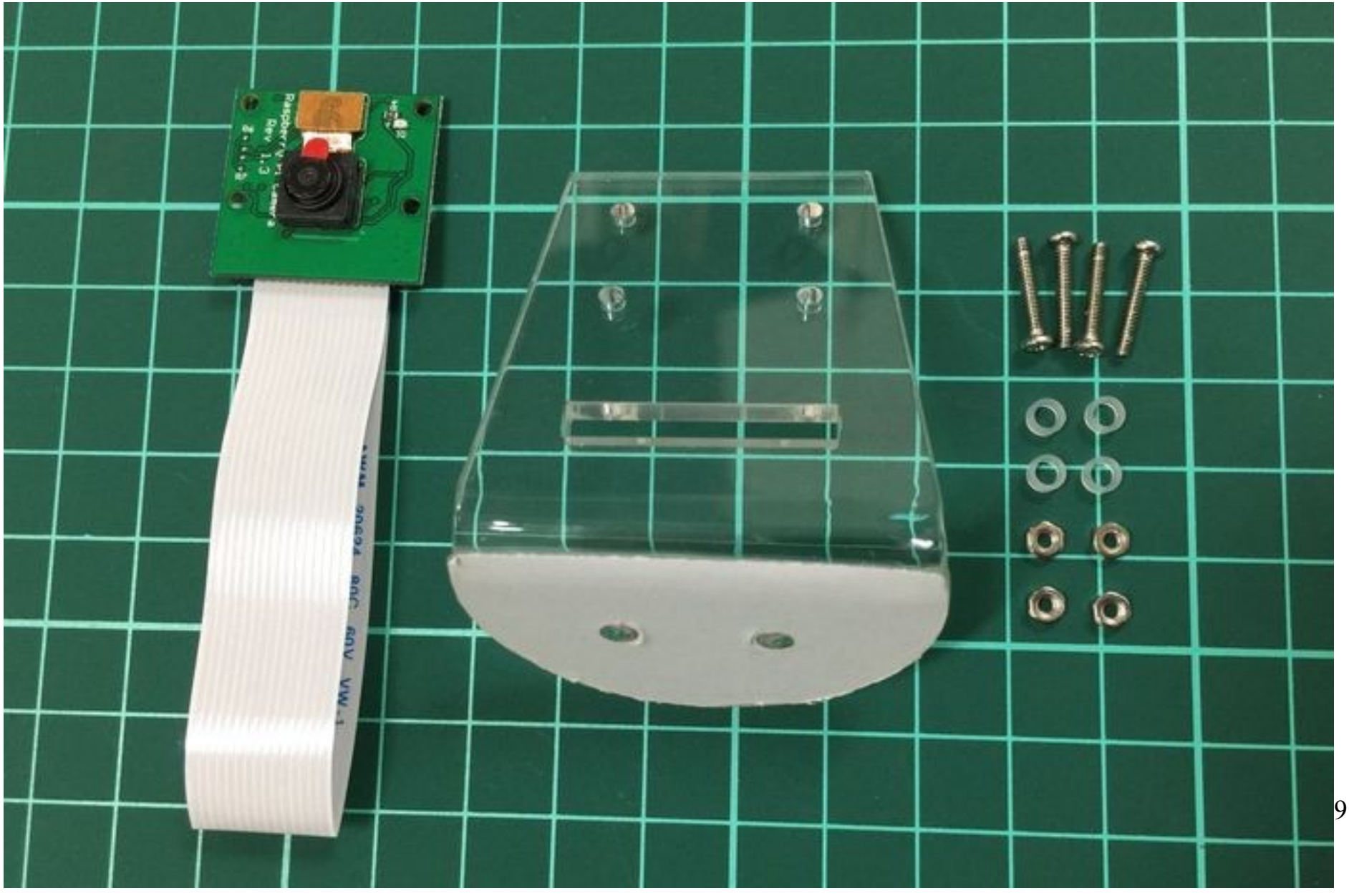


固定在車體上 (只有三處能固定)

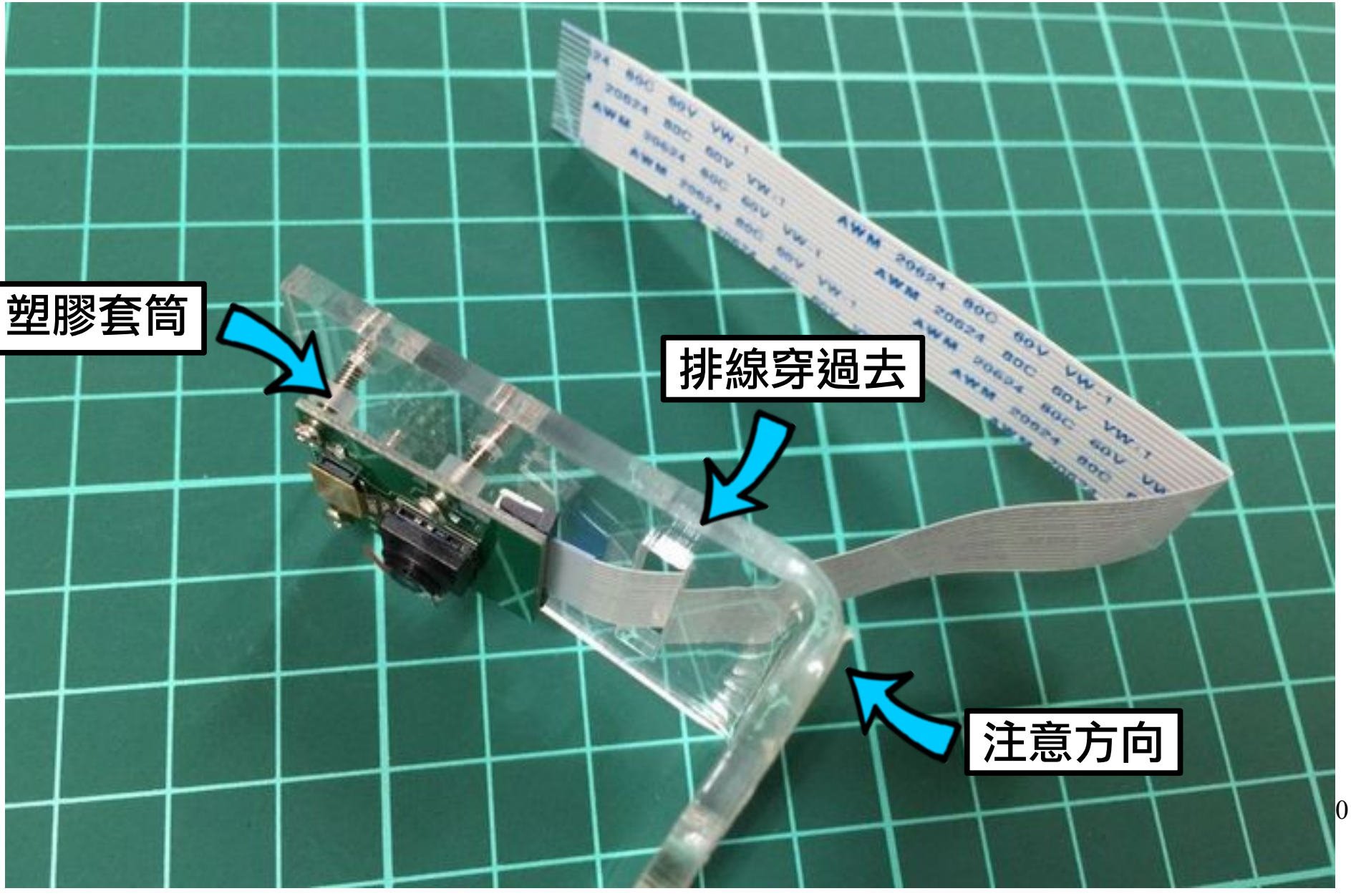
和 L298N 相接


共接地到 I4
共接地
L298N RPi
IN1(Brown) Pin16
IN2(RED) Pin18
IN3(Orange) Pin11
IN4(Yellow) Pin13

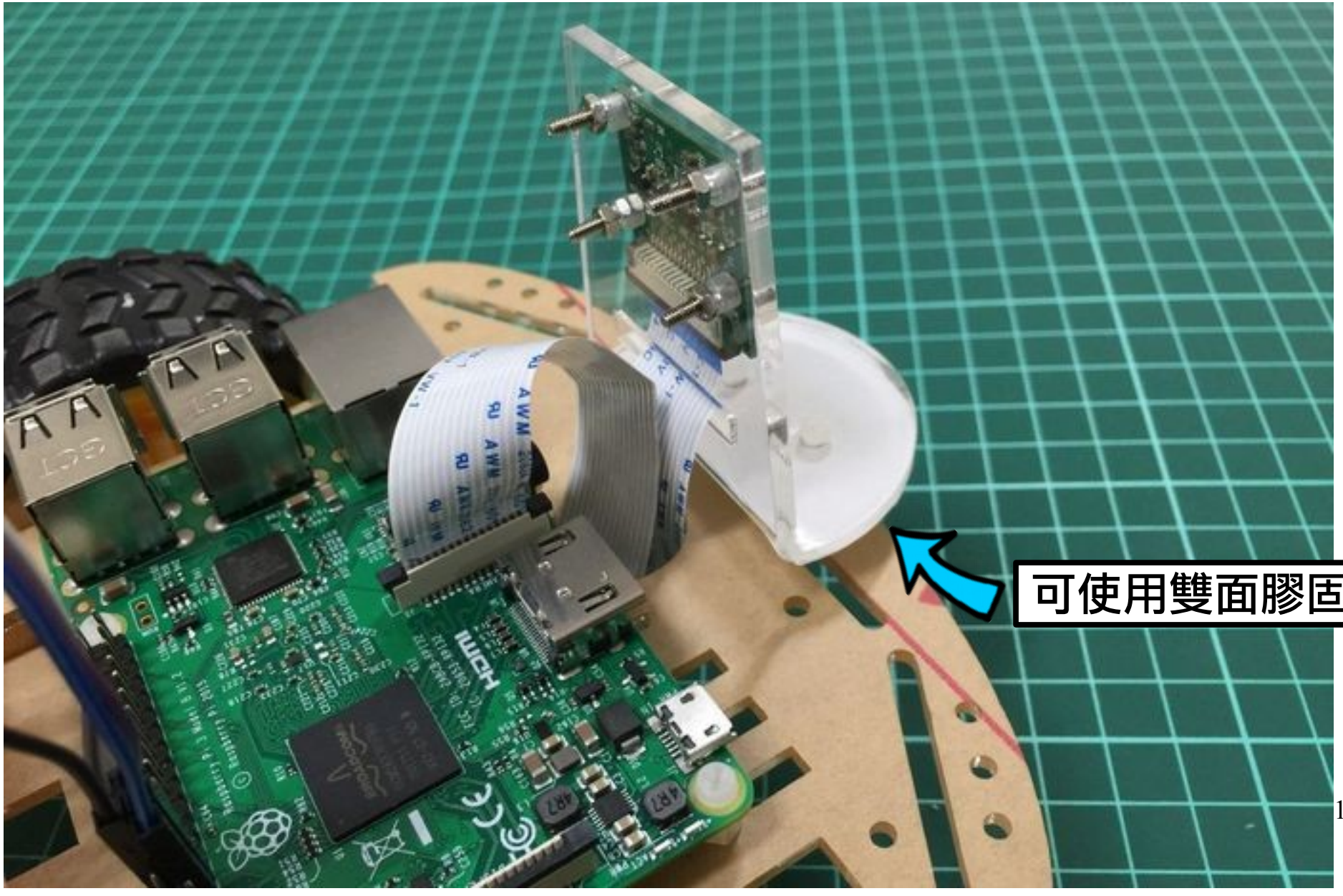
相機模組與固定架

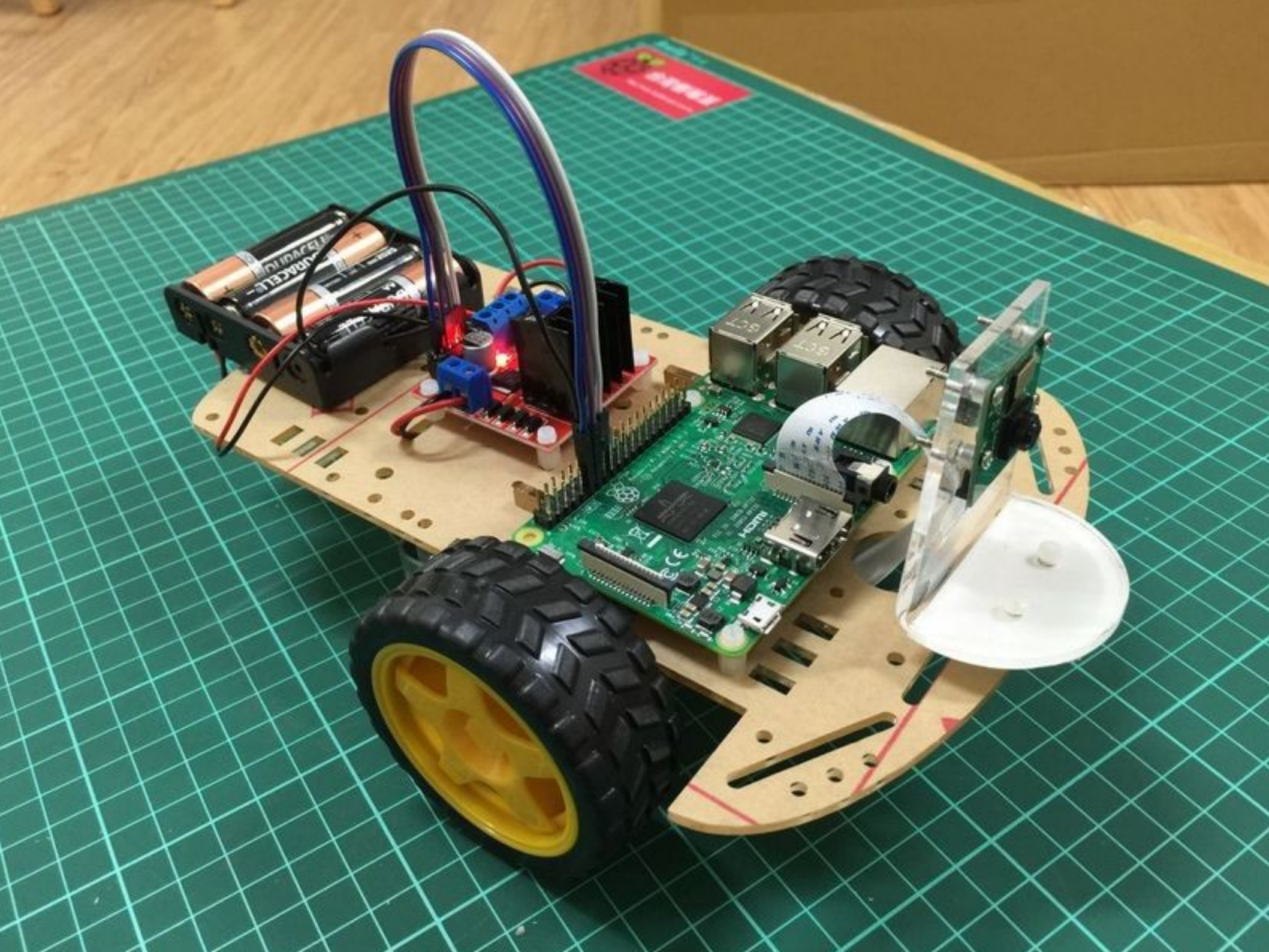


安裝相機模組

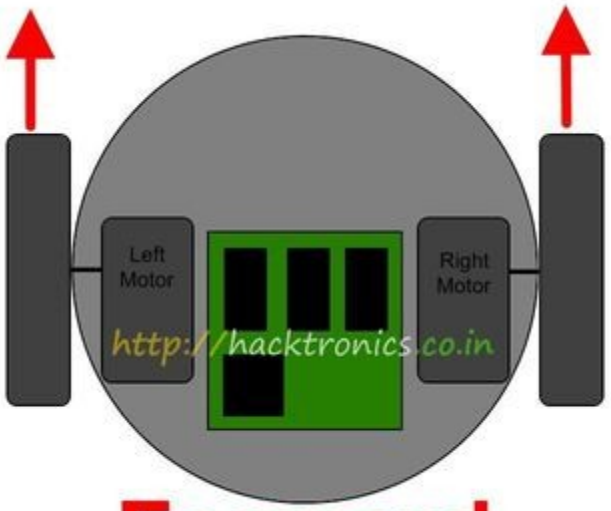


將相機模組與支架固定在車體上

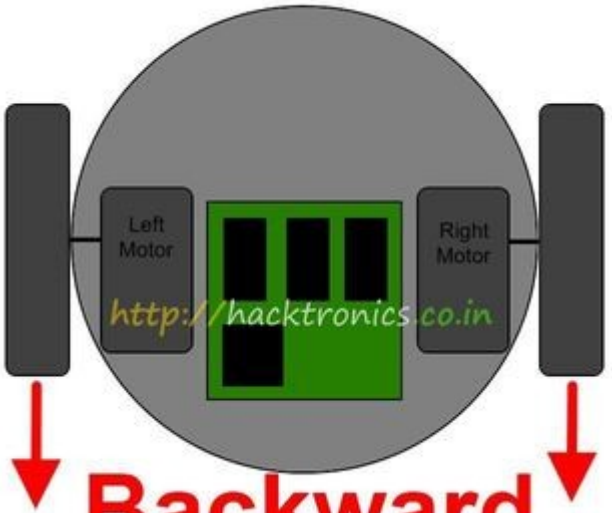




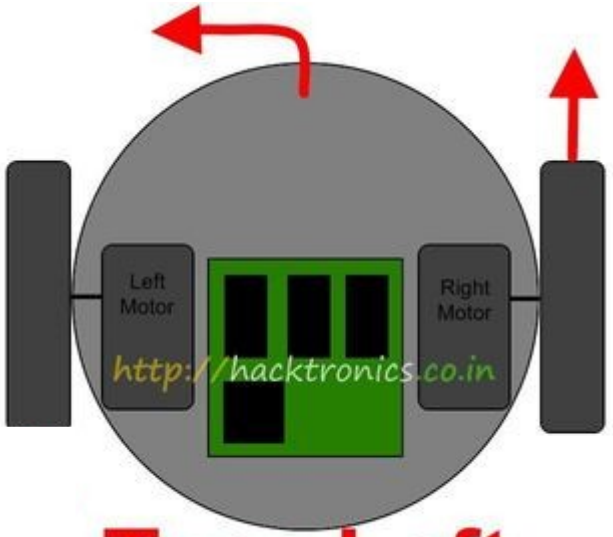
車子如何移動？



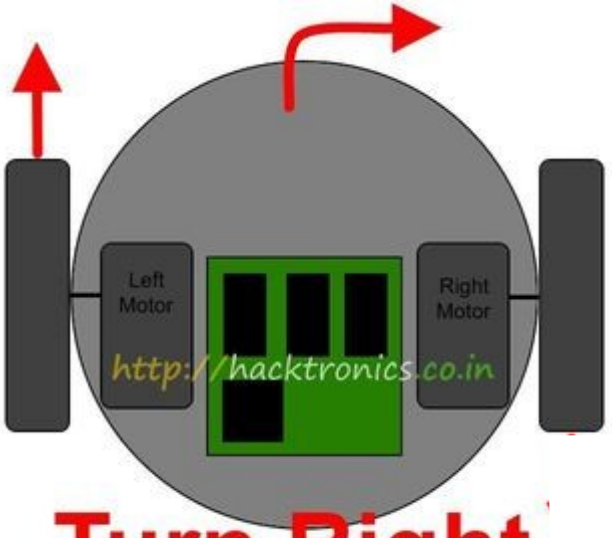
Forward



Backward



Turn Left

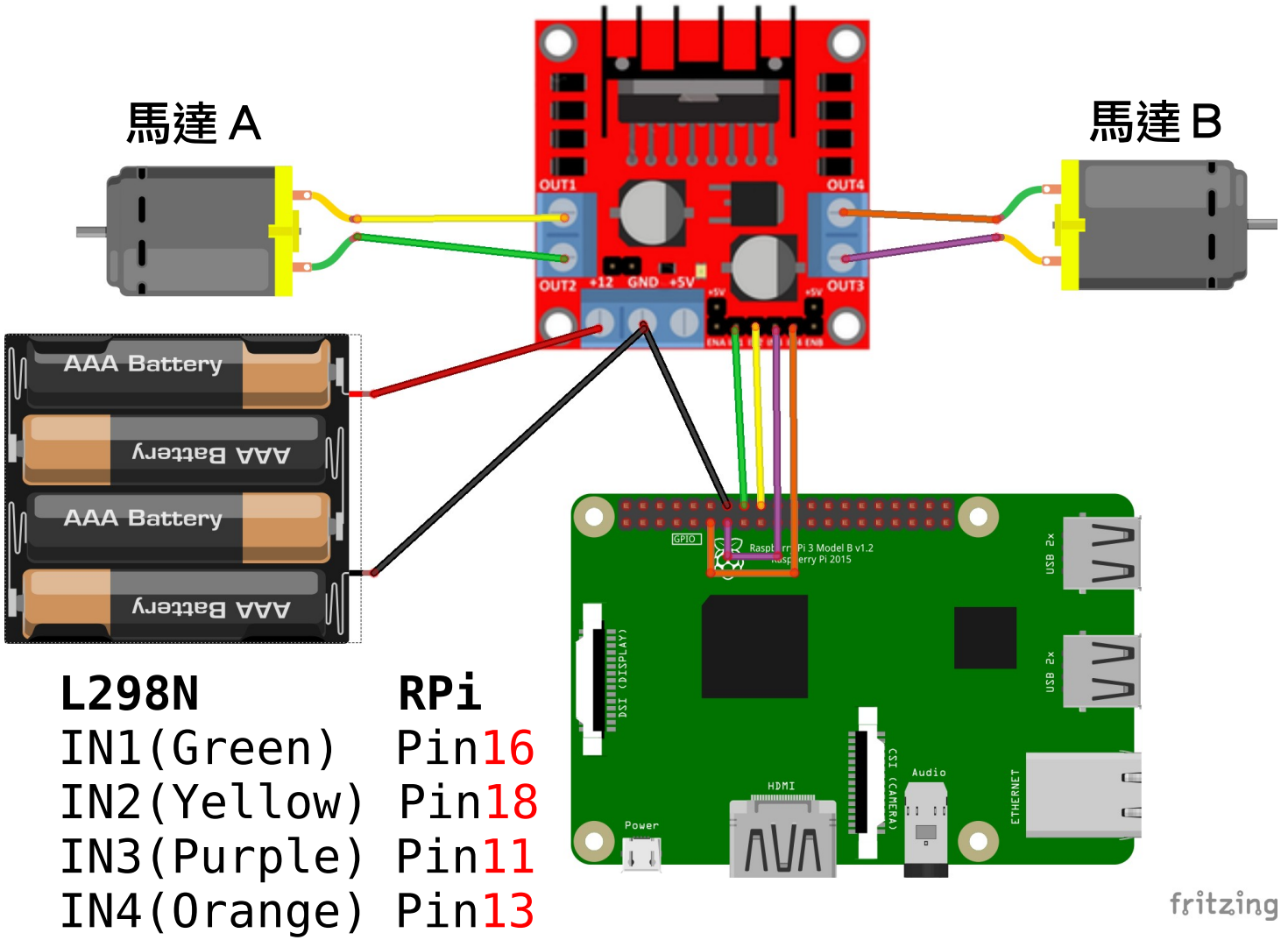


Turn Right

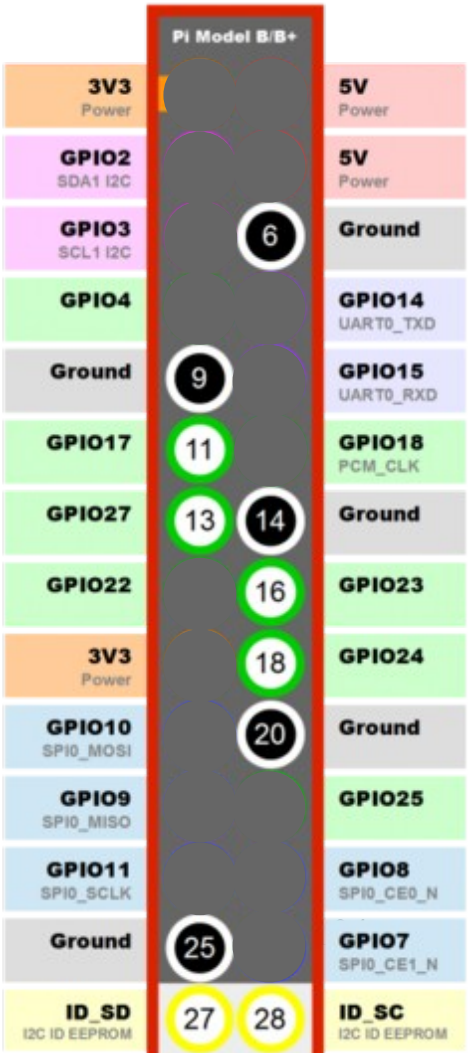
https://hacktronics.co.in/wtblog/4_using-the-l298-motor-driver-to-control-the-ro.html

雙輪接線圖

Motor	L298N	L298N	Motor
馬達 A(Yellow)	OUT1(Yellow)	OUT3(Orange)	馬達 B(Green)
馬達 A(Green)	OUT2(Green)	OUT4(Purple)	馬達 B(Yellow)



L298N	RPi
IN1(Green)	Pin16
IN2(Yellow)	Pin18
IN3(Purple)	Pin11
IN4(Orange)	Pin13



控制小車移動

```
def stop():
    # stop all wheels
def forward():
    GPIO.output(16, GPIO.HIGH)
    GPIO.output(18, GPIO.LOW)
    GPIO.output(11, GPIO.HIGH)
    GPIO.output(13, GPIO.LOW)
    time.sleep(1)
    stop()
try:
    while True:
        ch = readchar.readkey()

        if ch == 'w':
            forward()
```

順時鐘轉

逆時鐘轉

讀取鍵盤輸入

執行 python 程式後，按 w/a/s/d 鍵移動

DEMO

move_car.py

```
$ cd ~/pi-follower-car/02-motor
$ python move_car.py
```

練習

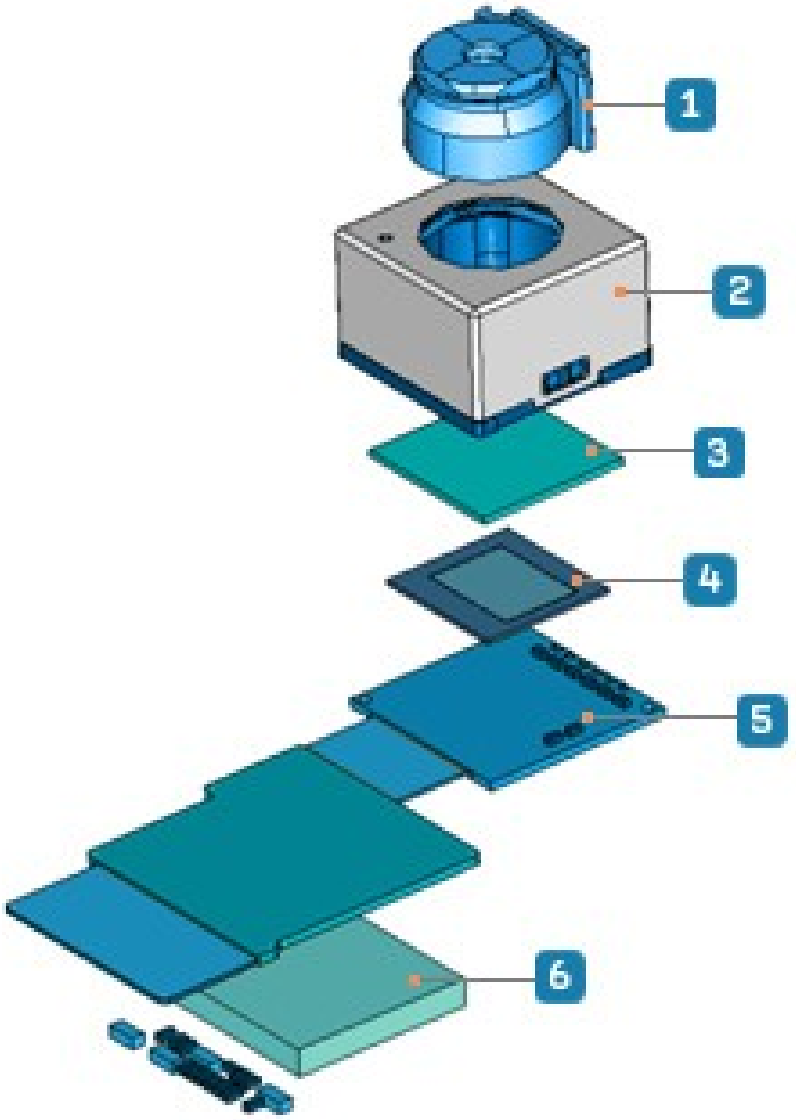
- 如何讓小車原地打轉？
- 如何讓小車自動走一個半徑 30cm 的圓形？

實驗 3:Camera 即時預覽

目的：瞭解 Camera 和 Webcam 的差異

Raspberry Pi Camera 簡介

從手機相機模組講起



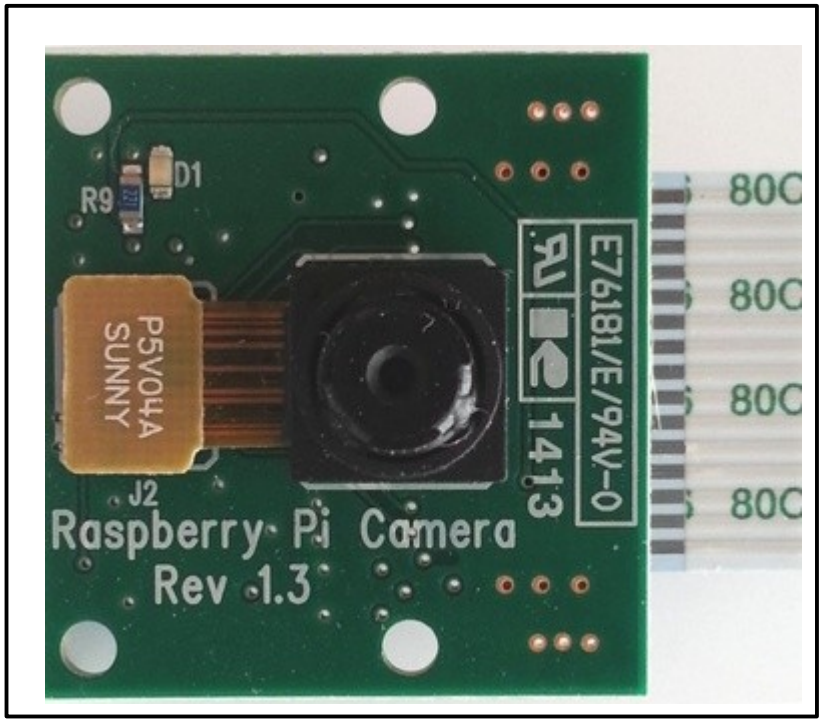
1. Lens(透鏡)
2. VCM(音圈馬達)
3. IR-Cut(紅外光濾片)
4. Sensor(感光元件)
5. PCB(印刷電路板)
6. ISP(影像訊號處理器)

http://www.phonearena.com/news/13MP-camera-tipped-for-Samsung-Galaxy-S-IV_id35168

技術規格

- Sensor:OmniVision OV5647(5M)
- 靜態拍照最高解析度:2592 x 1944 pixel
- Pixel Size:1.4 x 1.4 um
- 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

Type of Raspberry Pi Camera

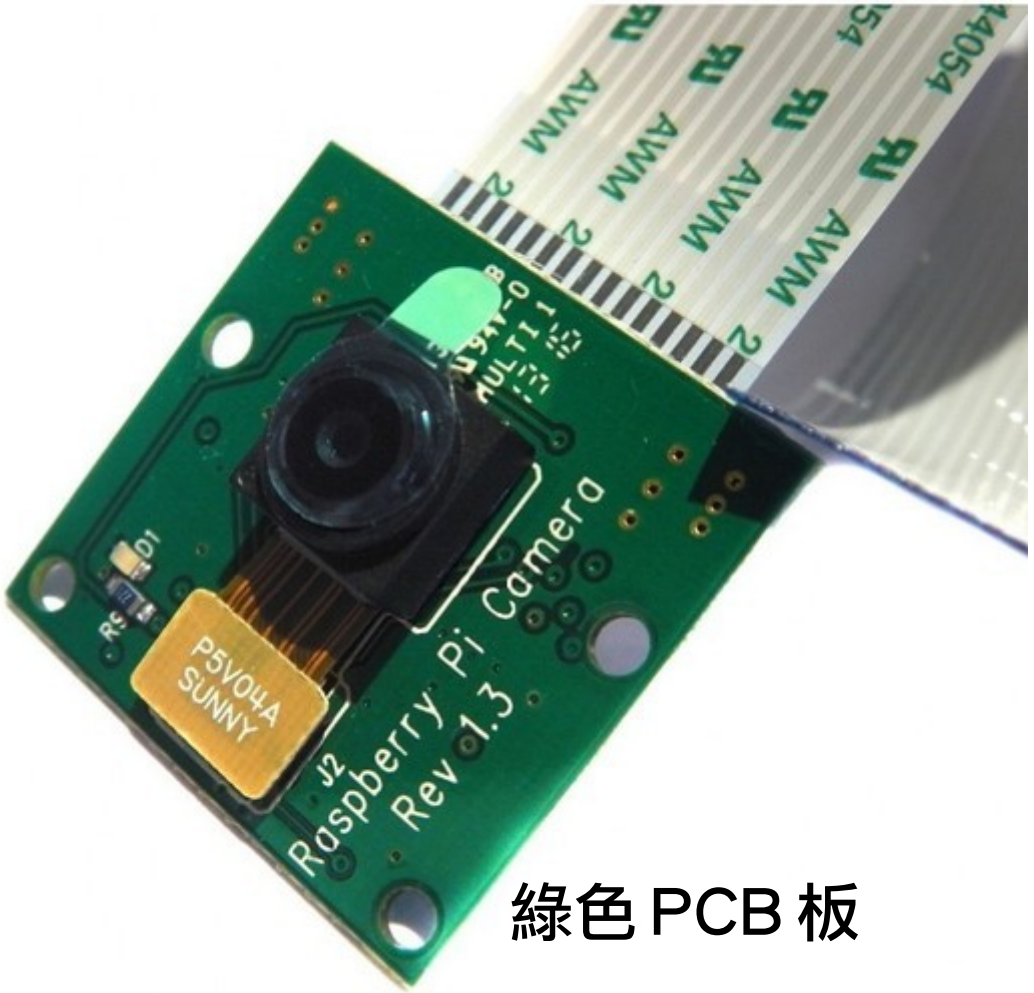


Raspberry Pi Camera Module

NoIR Camera Module

http://elinux.org/Rpi_Camera_Module

Raspberry Pi Camera Module(v1)



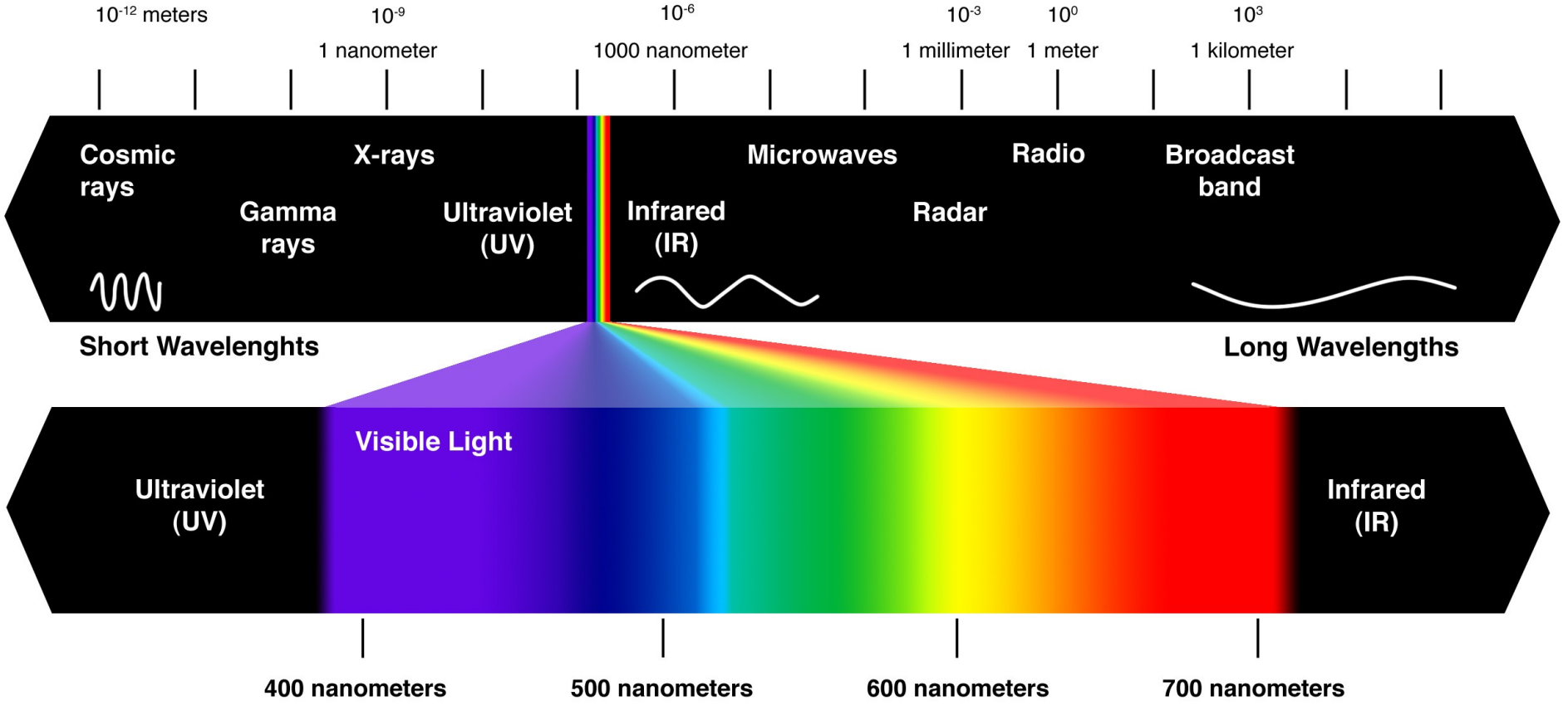
15-Pins, CSI 介面

綠色 PCB 板

https://www.modmypi.com

基礎光學原理

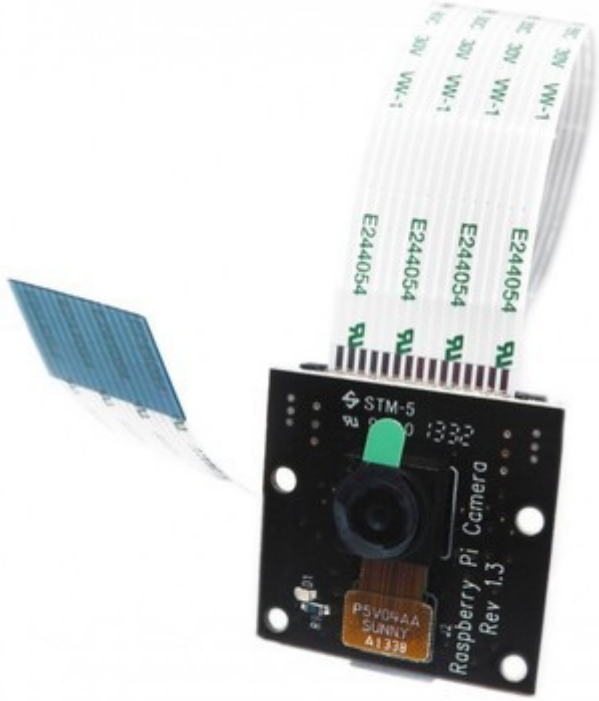
- 問：樹葉為什麼看起來是綠色的？
- 答：因為樹葉吸收了大部分可見光，只反射綠色光



https://www.raspberrypi.org/learning/infrared-bird-box/worksheet/

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

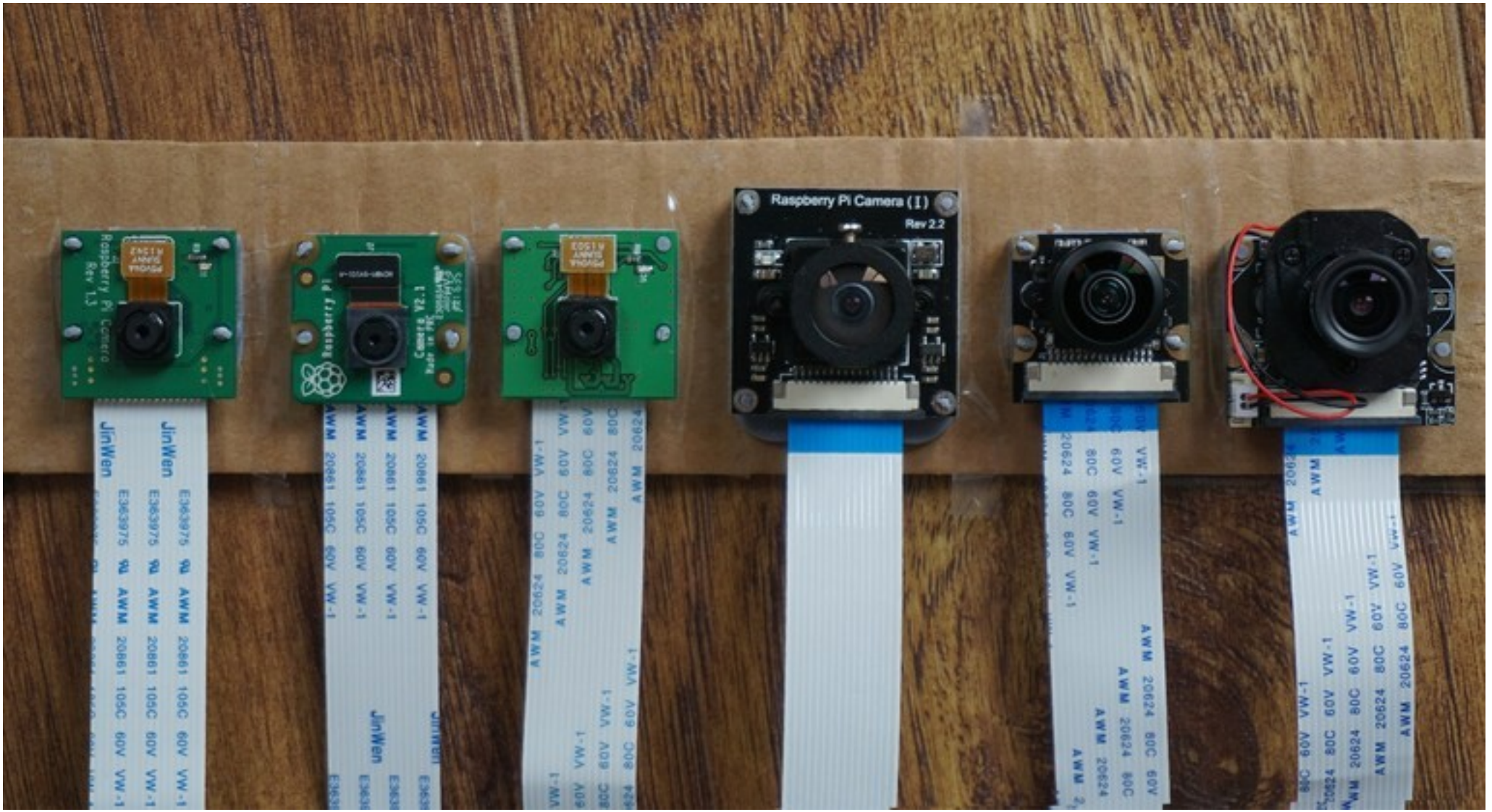
2. NoIR 相機

3. NoIR 相機

4. NoIR 相機 + 藍色濾光片

https://www.raspberrypi.org/magpi/issues/18/

更多相機超級比一比



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

Raspberry Pi Camera Comparison

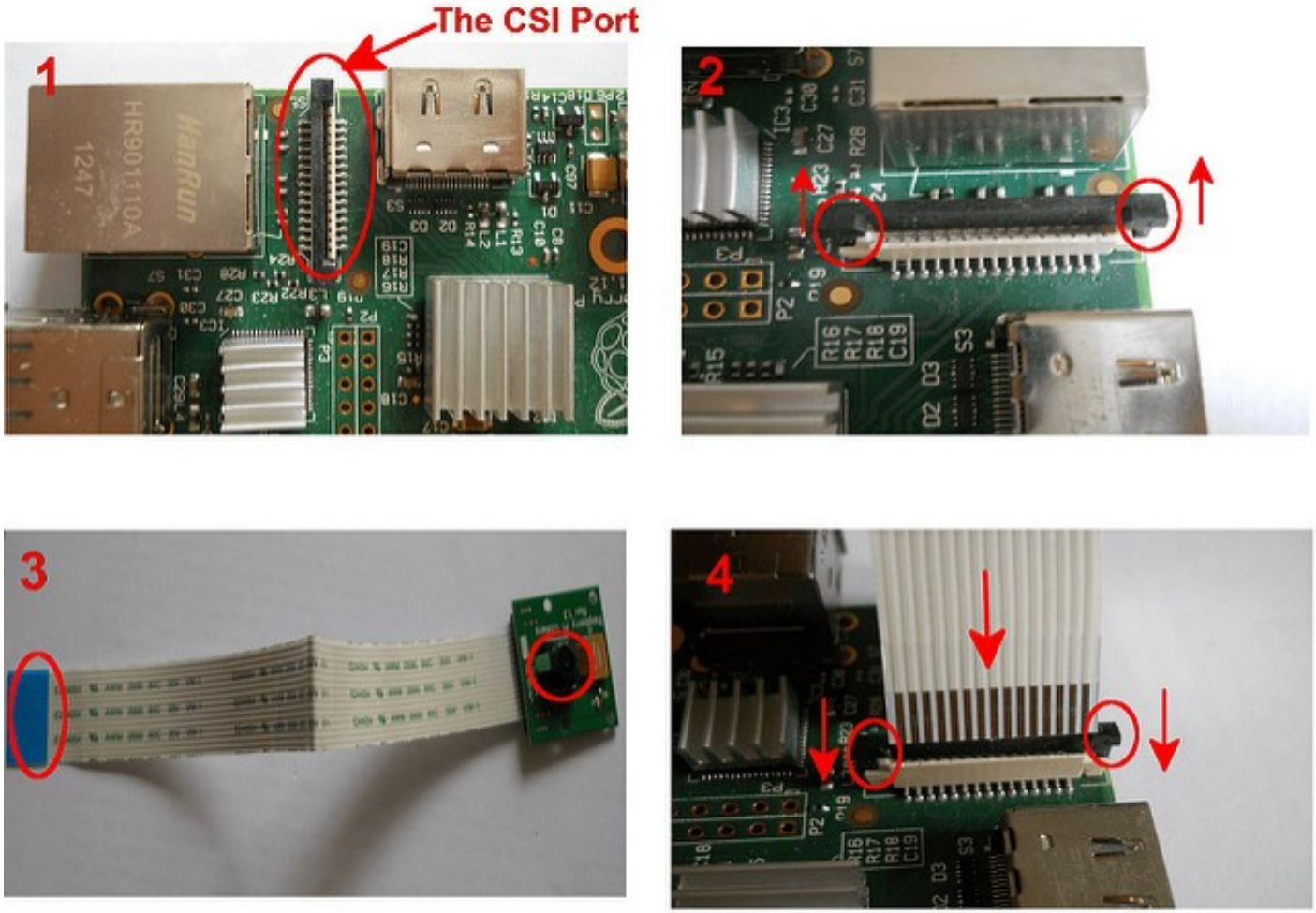
semifluid.com

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

Camera 安装

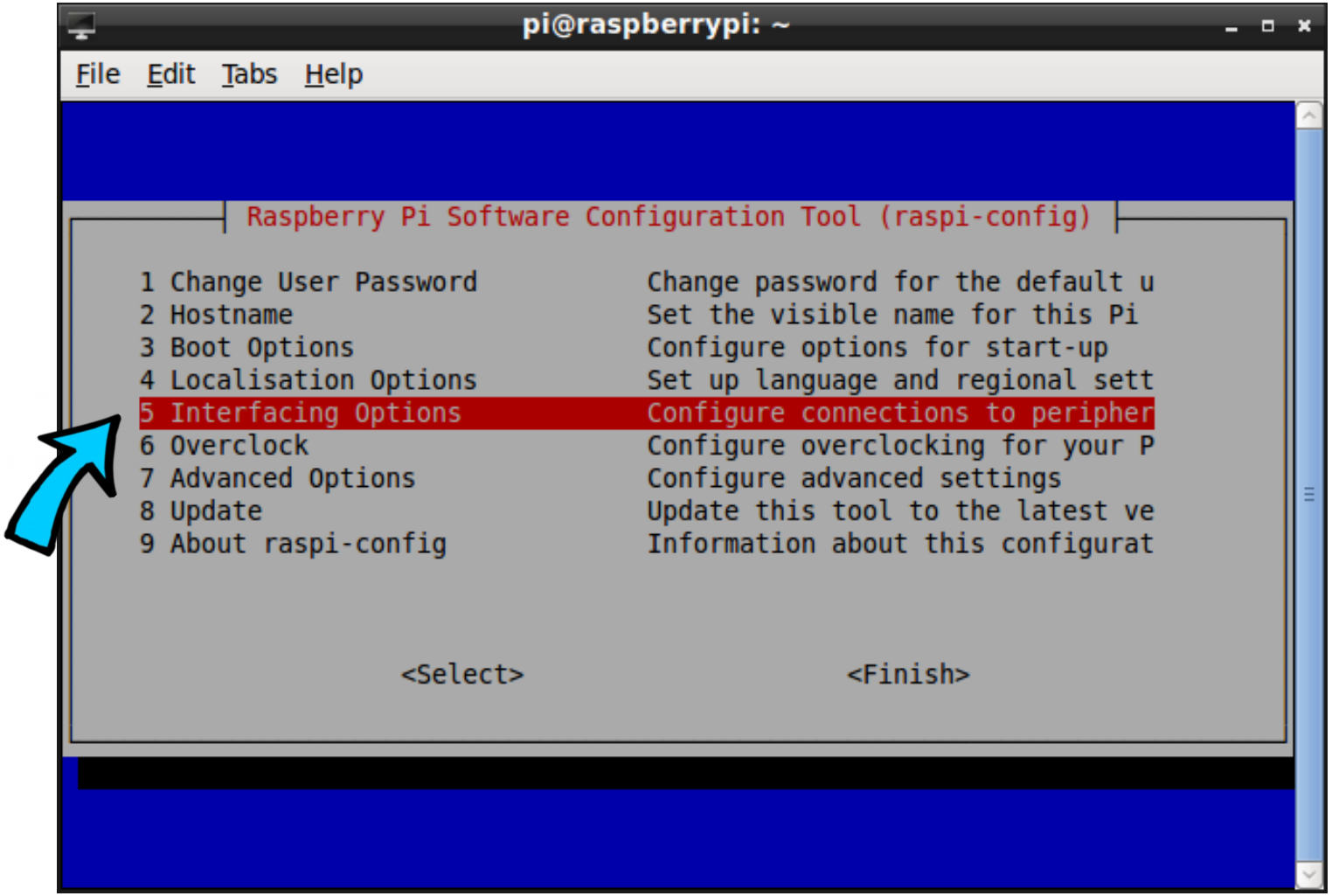
在關機的狀態下安裝 Camera

關機指令： $ sudo poweroff

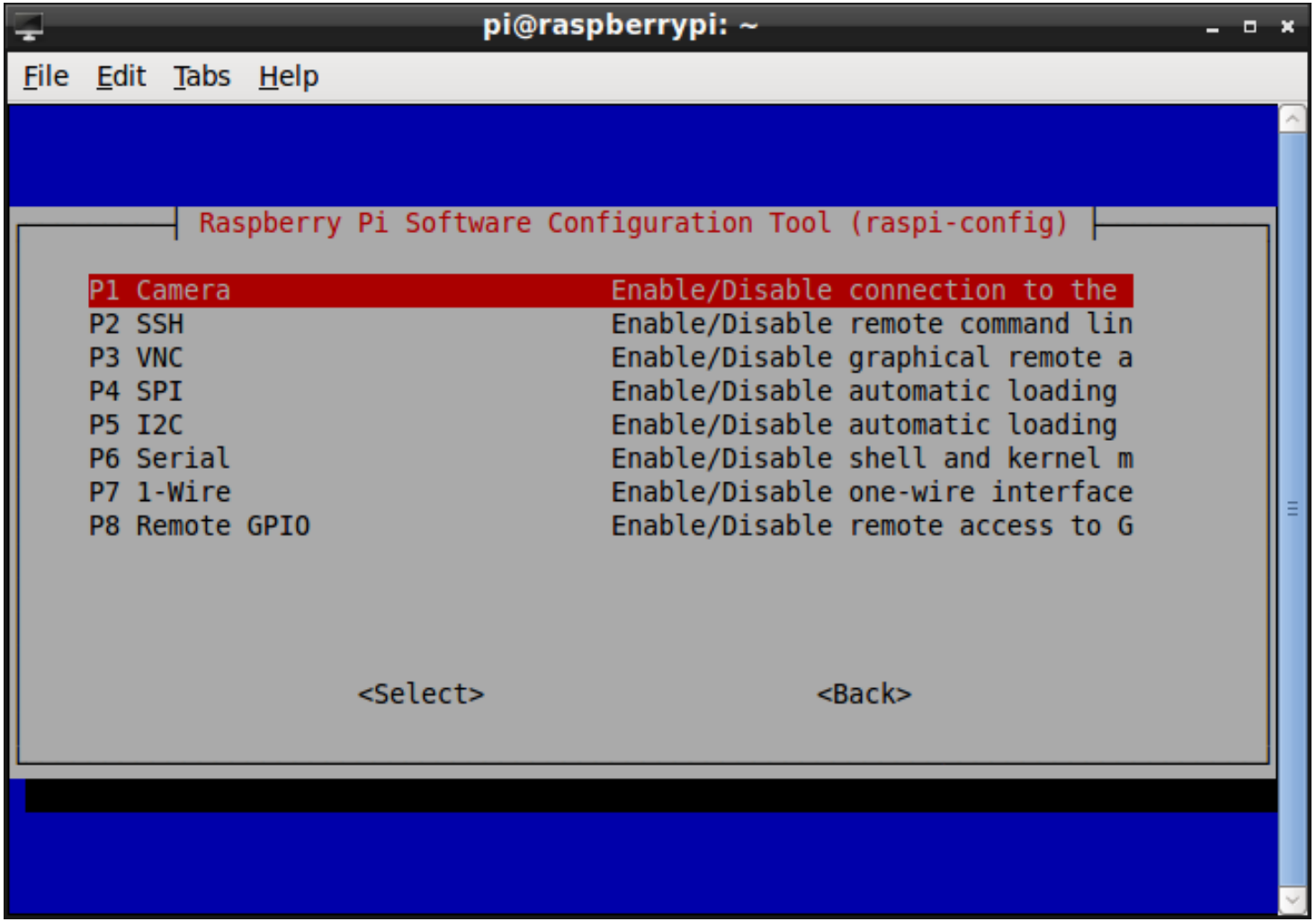


http://xmodulo.com/install-raspberry-pi-camera-board.html

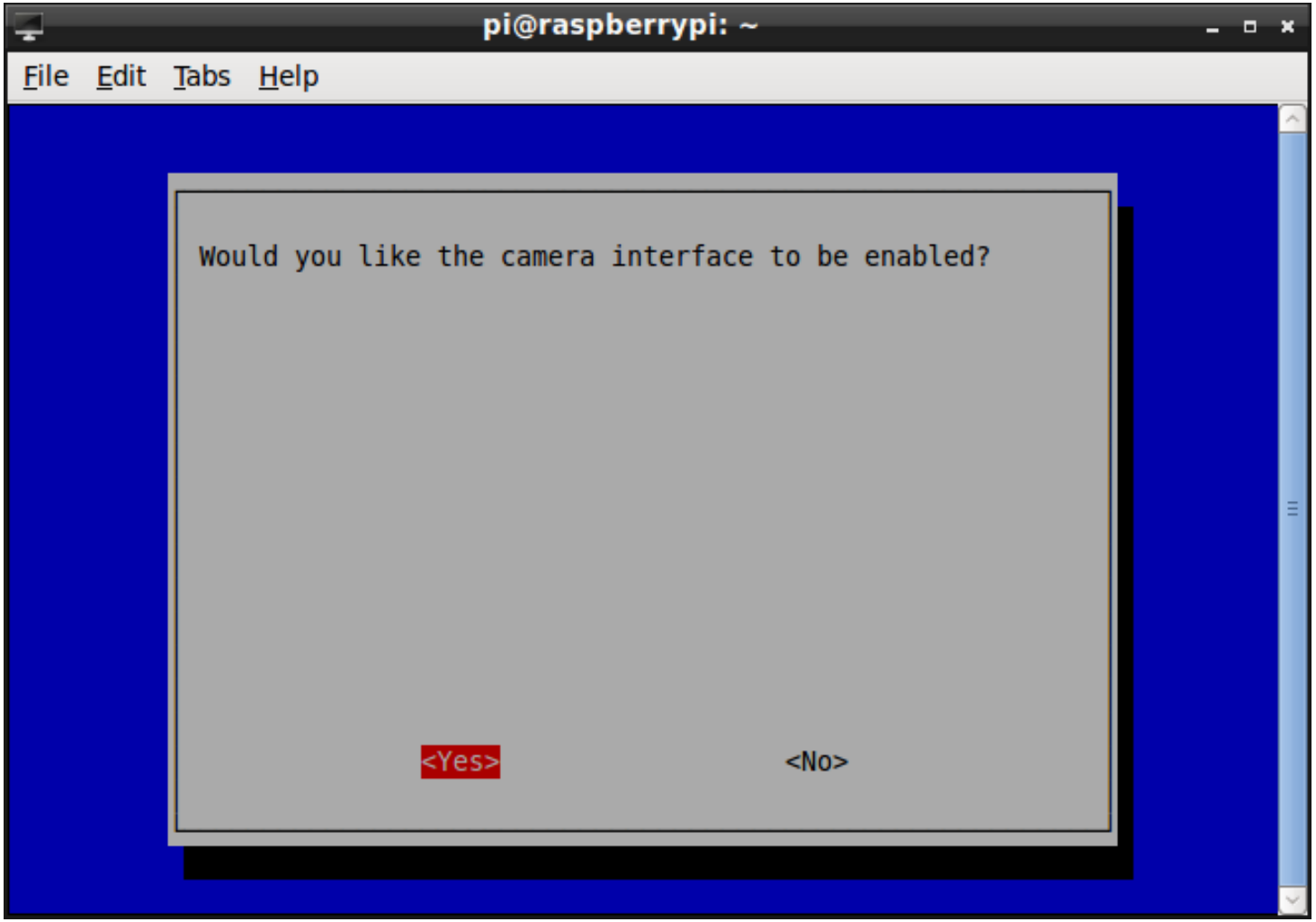
$ sudo raspi-config



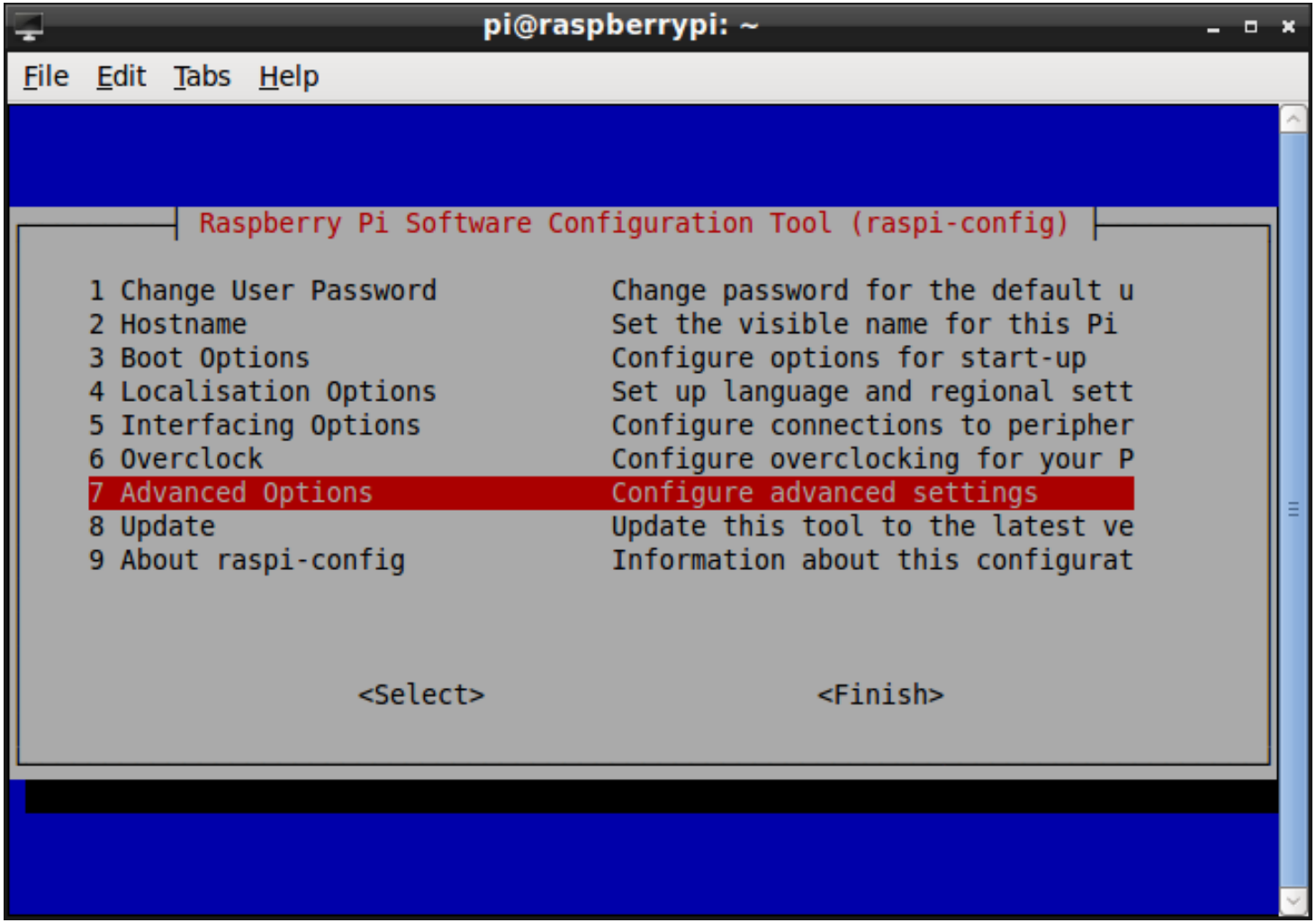
選擇 Camera


pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
P1 Camera Enable/Disable connection to the
P2 SSH Enable/Disable remote command lin
P3 VNC Enable/Disable graphical remote a
P4 SPI Enable/Disable automatic loading
P5 I2C Enable/Disable automatic loading
P6 Serial Enable/Disable shell and kernel m
P7 1-Wire Enable/Disable one-wire interface
P8 Remote GPIO Enable/Disable remote access to G
<Select>
<Back>

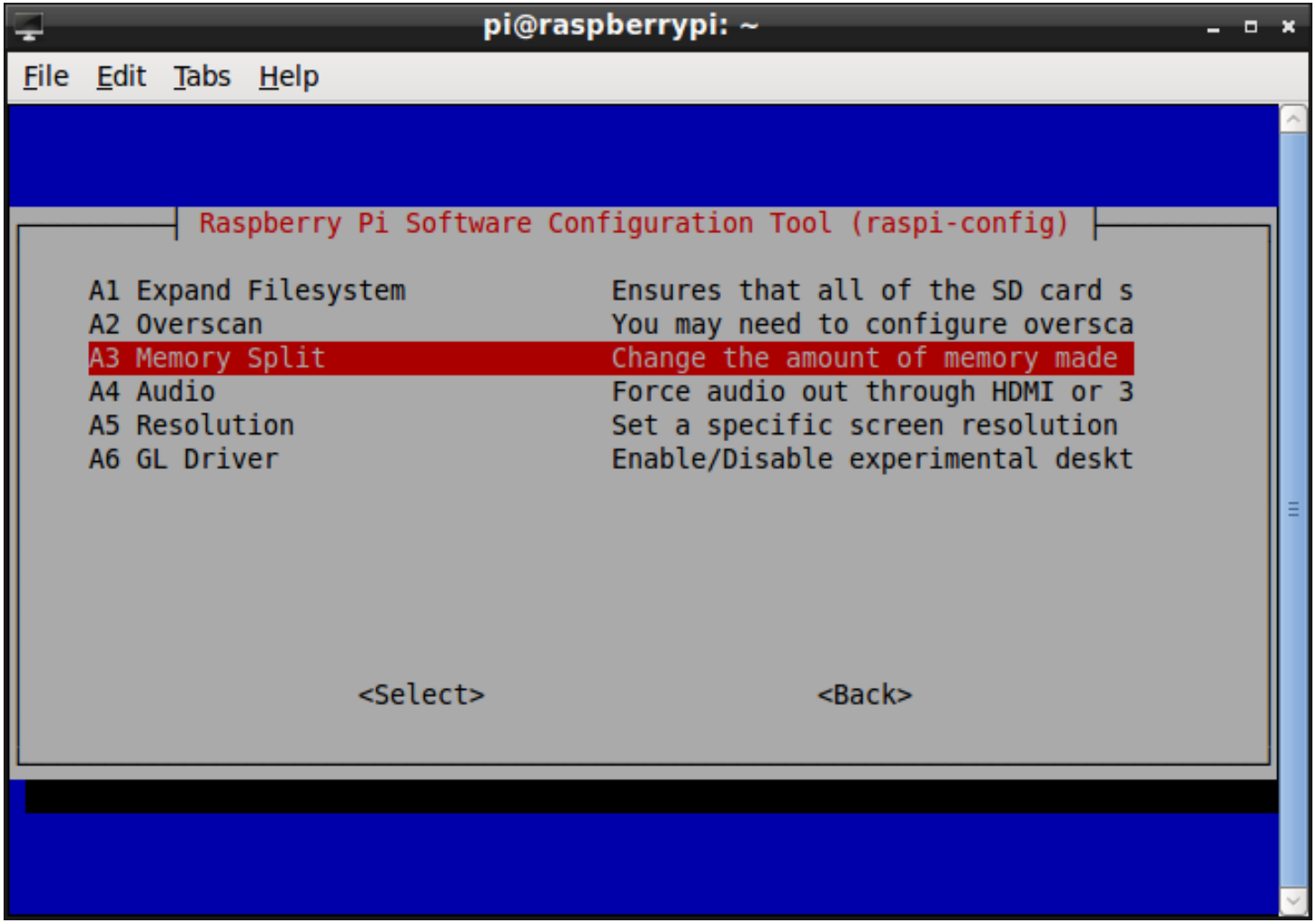
啟用 Raspberry Pi Camera



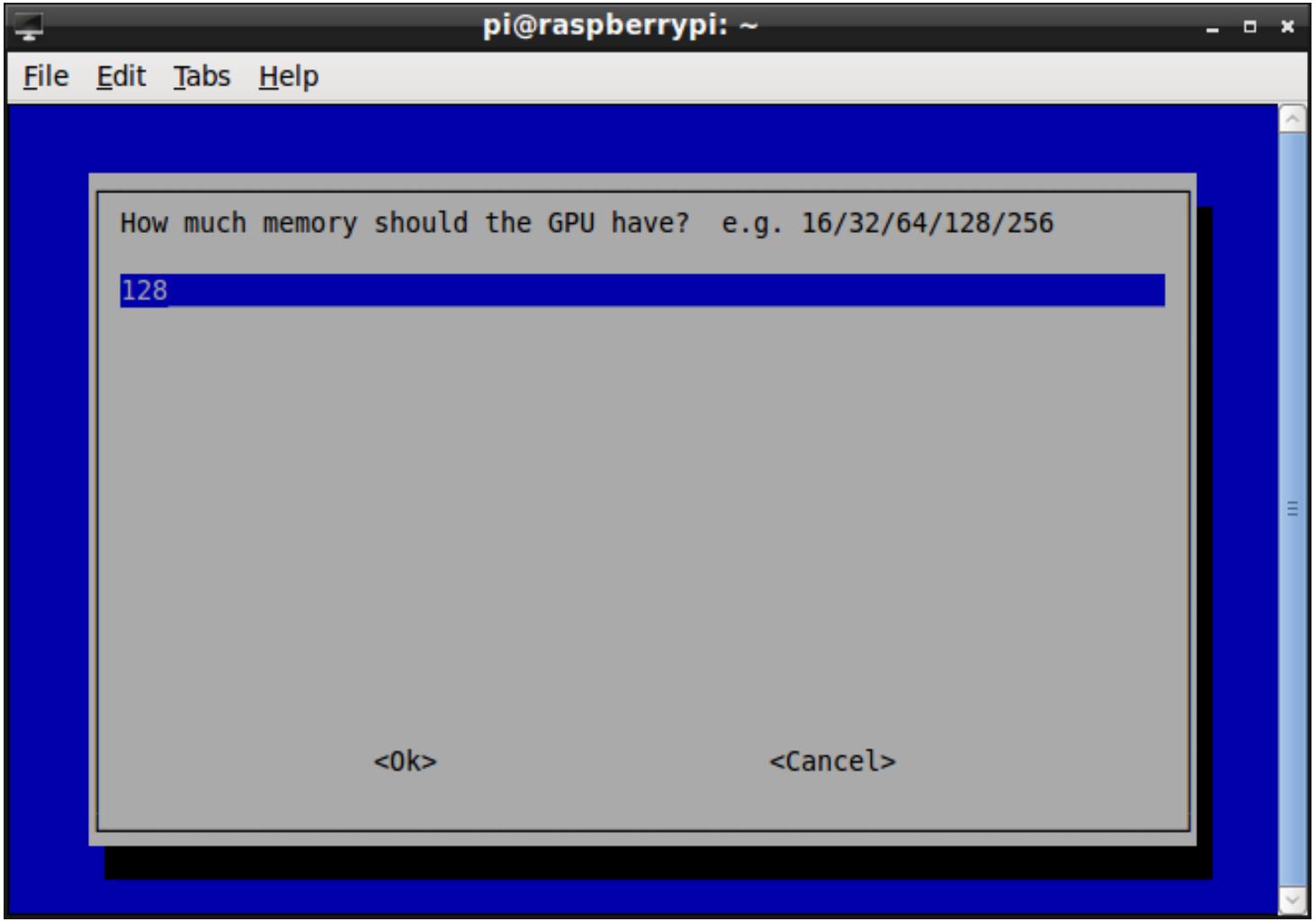
進階選項


pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
1 Change User Password Change password for the default u
2 Hostname Set the visible name for this Pi
3 Boot Options Configure options for start-up
4 Localisation Options Set up language and regional sett
5 Interfacing Options Configure connections to peripher
6 Overclock Configure overclocking for your P
7 Advanced Options Configure advanced settings
8 Update Update this tool to the latest ve
9 About raspi-config Information about this configurat
<Select>
<Finish>

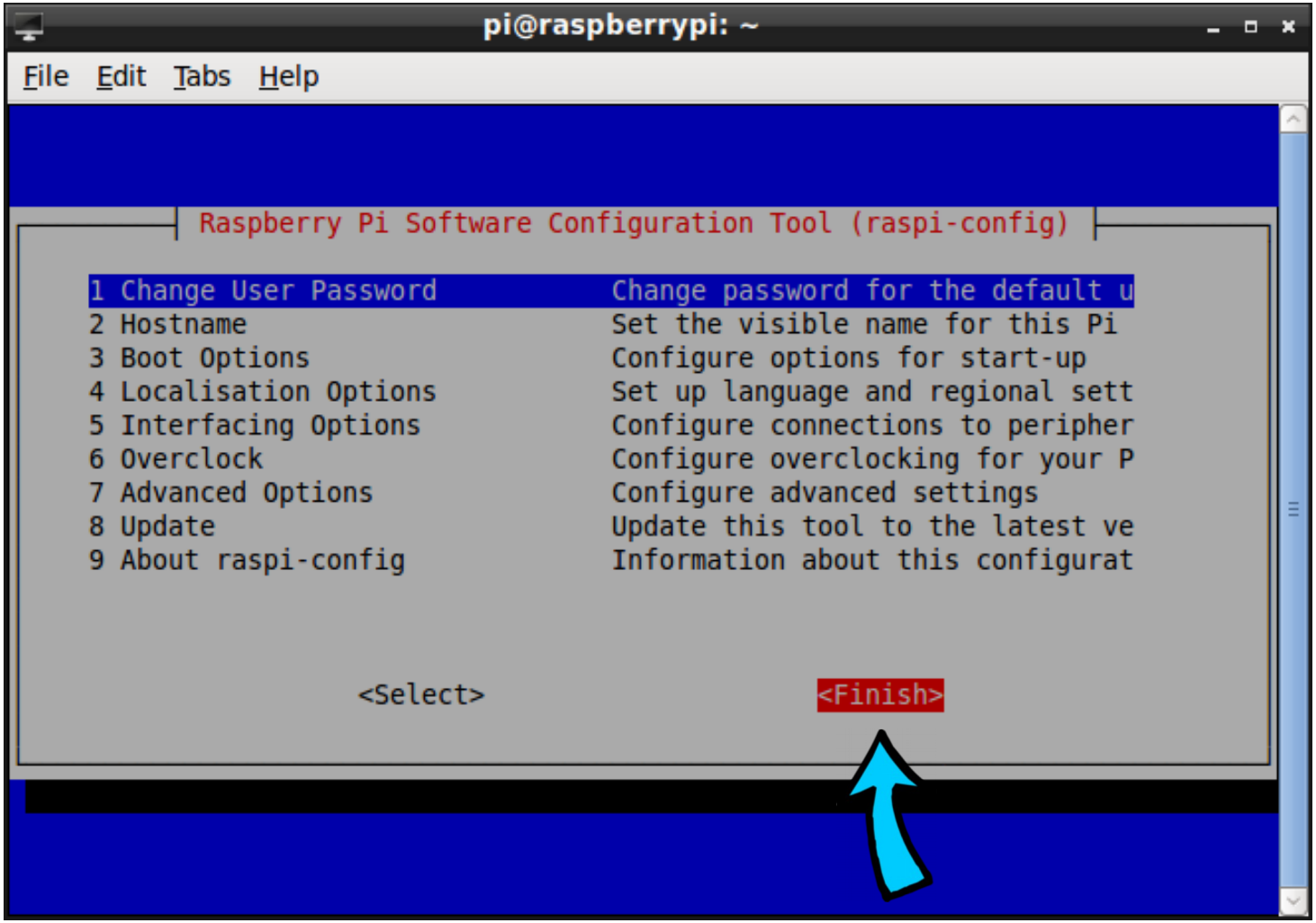
設定記憶體分配


pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
A1 Expand Filesystem
Ensures that all of the SD card s
A2 Overscan
You may need to configure oversca
A3 Memory Split
Change the amount of memory made
A4 Audio
Force audio out through HDMI or 3
A5 Resolution
Set a specific screen resolution
A6 GL Driver
Enable/Disable experimental deskt
<Select>
<Back>

>128M


pi@raspberrypi: ~
File Edit Tabs Help
How much memory should the GPU have? e.g. 16/32/64/128/256
128
<Ok>
<Cancel>

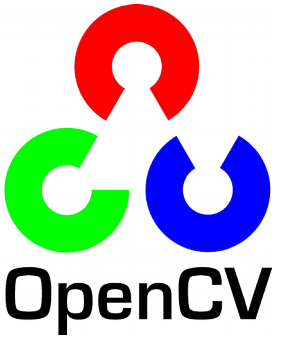
回主選單後選擇 <Finish>



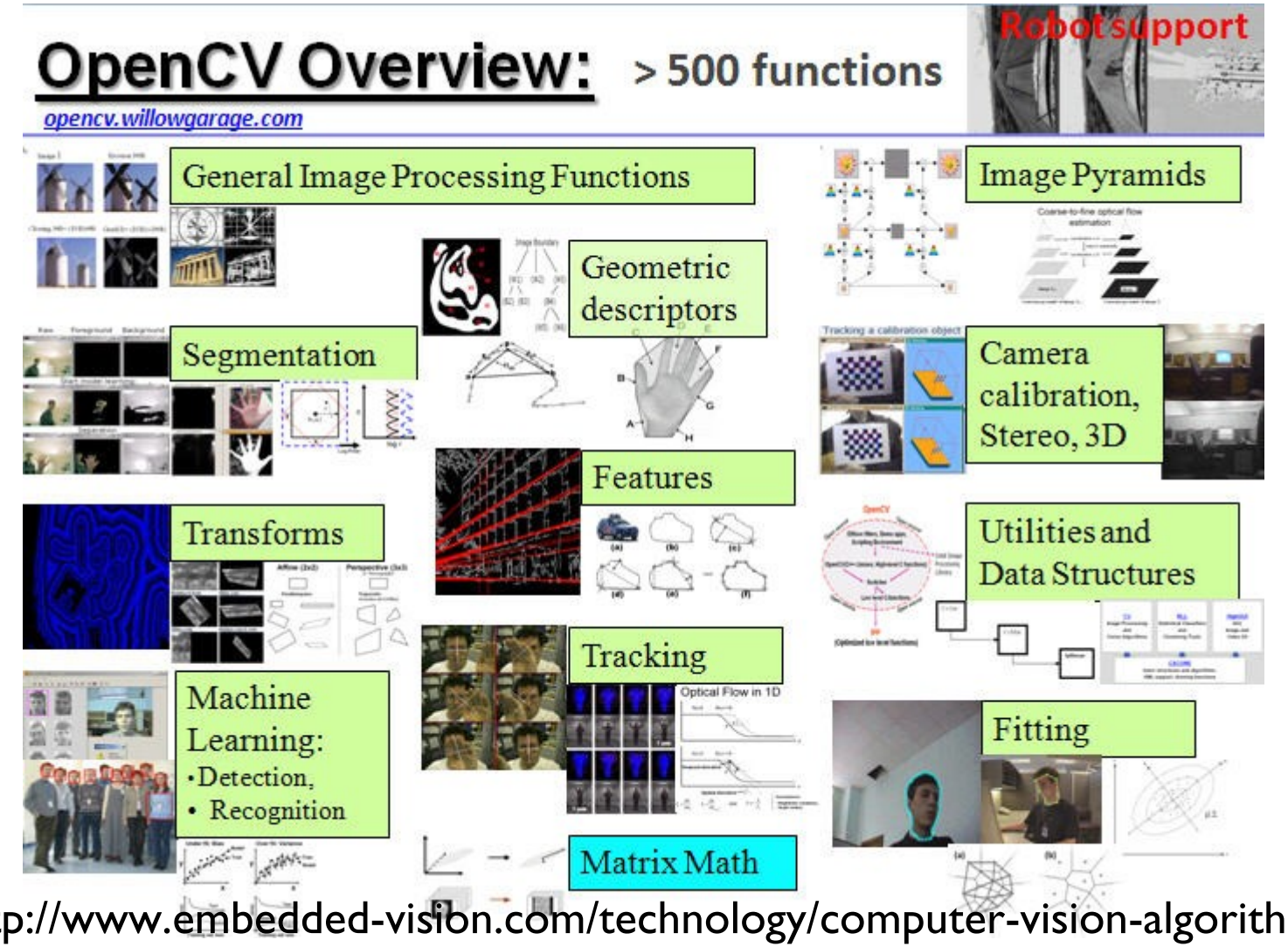
實戰 Camera 使用

OpenCV

- Open Source Computer Vision Library



- 跨平台的計算機函式庫，主要由 C/C++ 撰寫



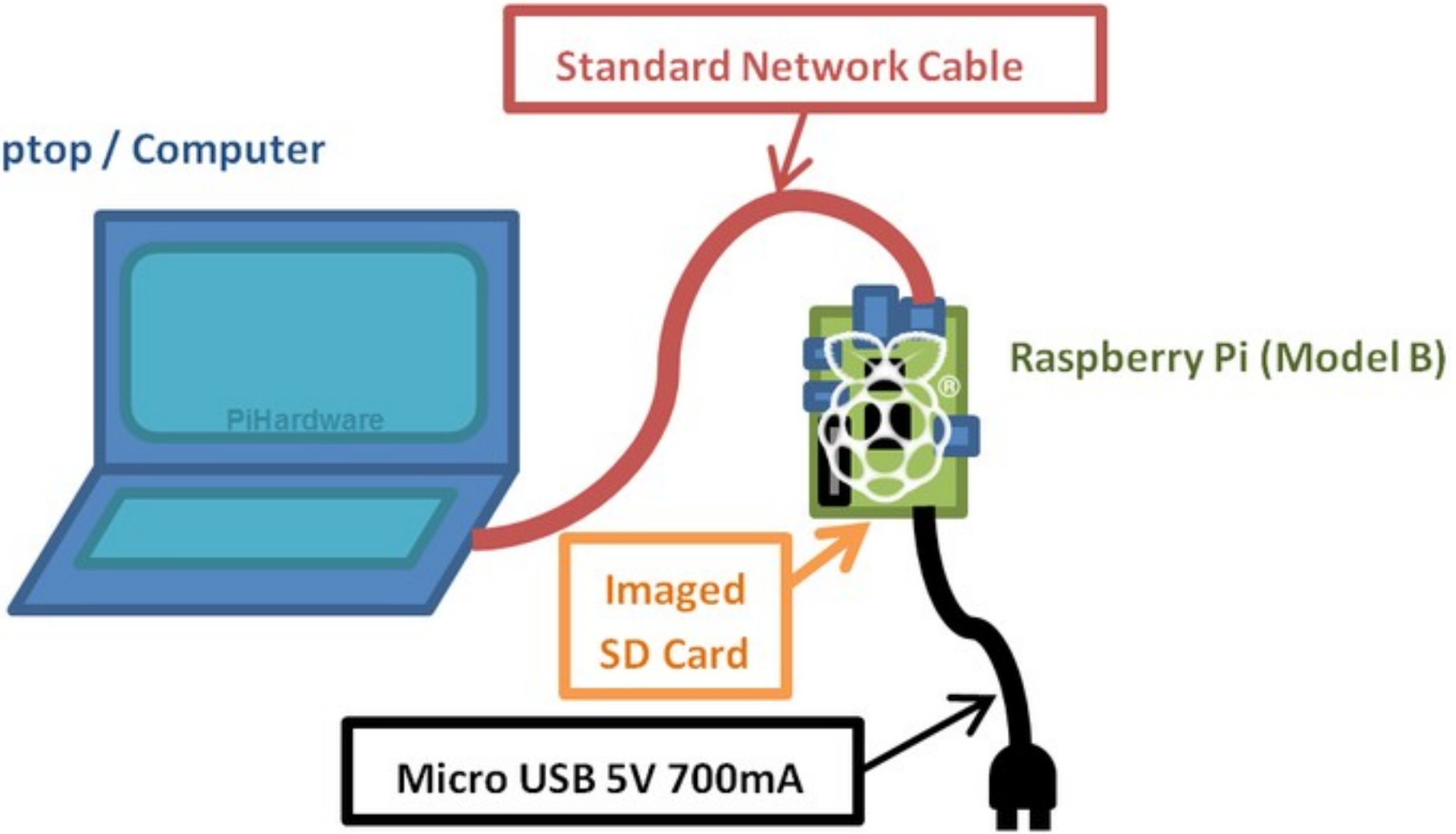
http://www.embedded-vision.com/technology/computer-vision-algorithms

載入圖檔並顯示

```
import cv2

image = cv2.imread("lena256rgb.jpg")

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

如何看執行結果？

直接用網路線對接



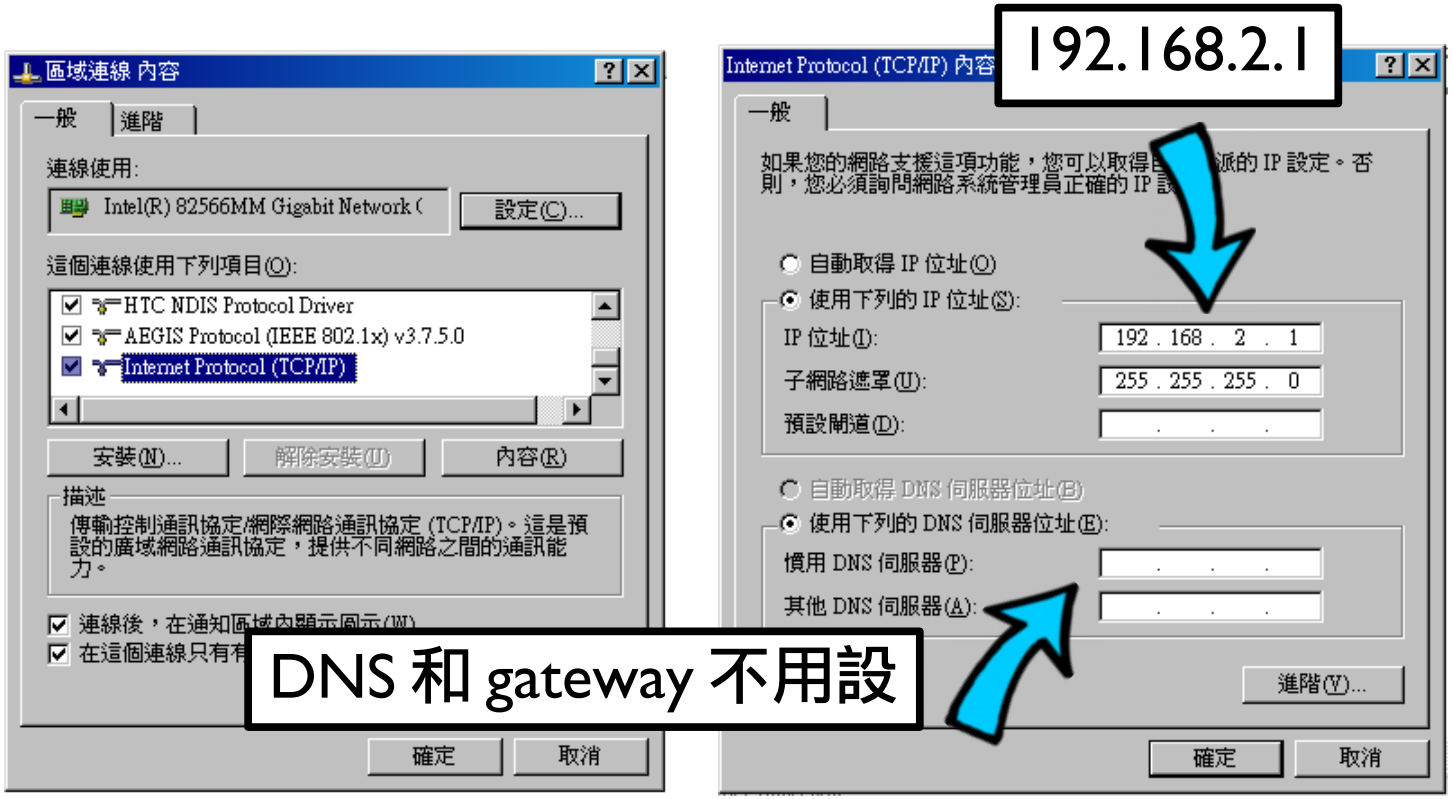
https://pihw.wordpress.com/guides/direct-network-connection/

網路設定

- 把Pi設成.2.2

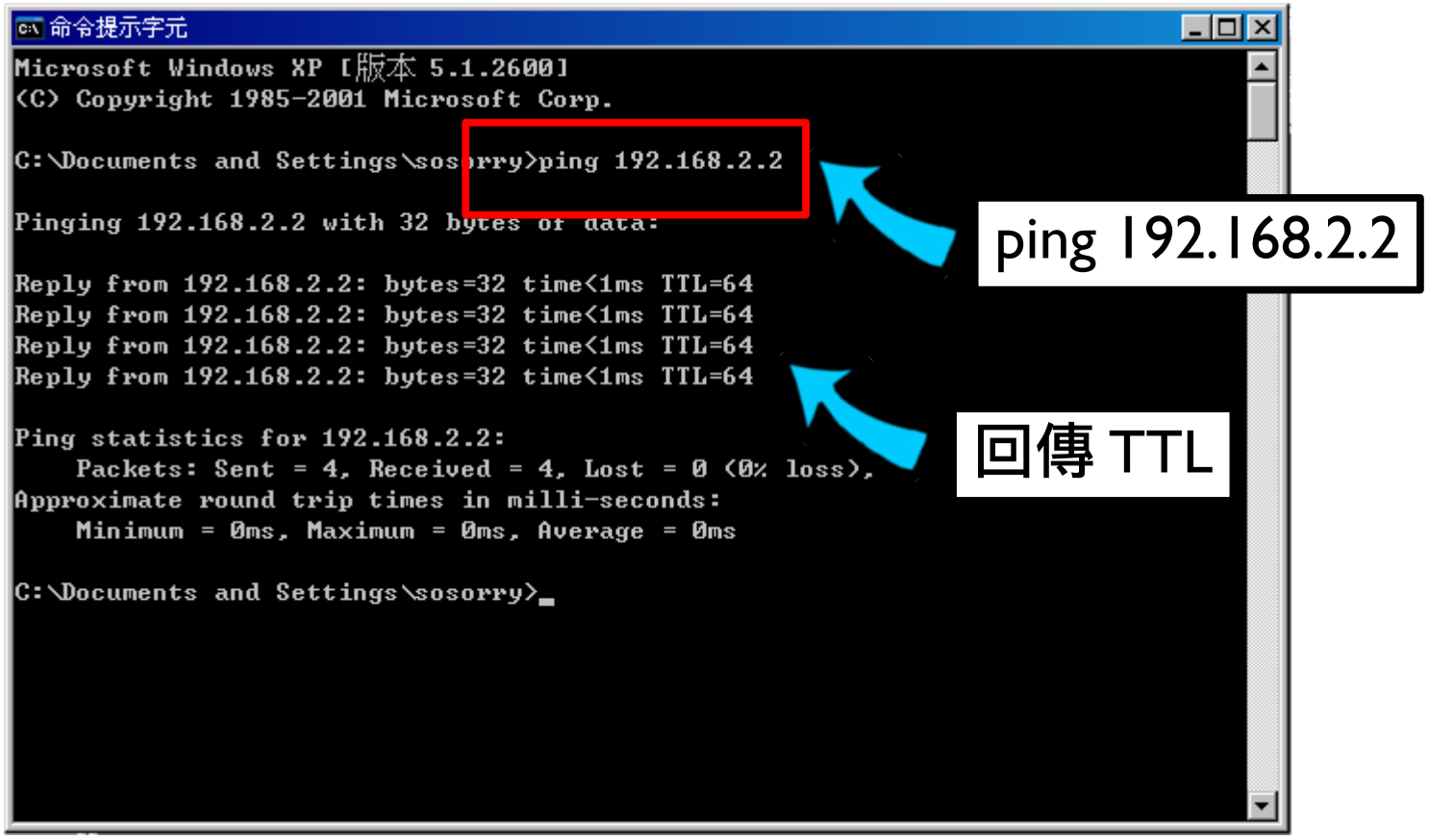
```
$ sudo ifconfig eth0 192.168.2.2 netmask 255.255.255.0
```

- 把Windows設成.2.1



確認網路是否有通？

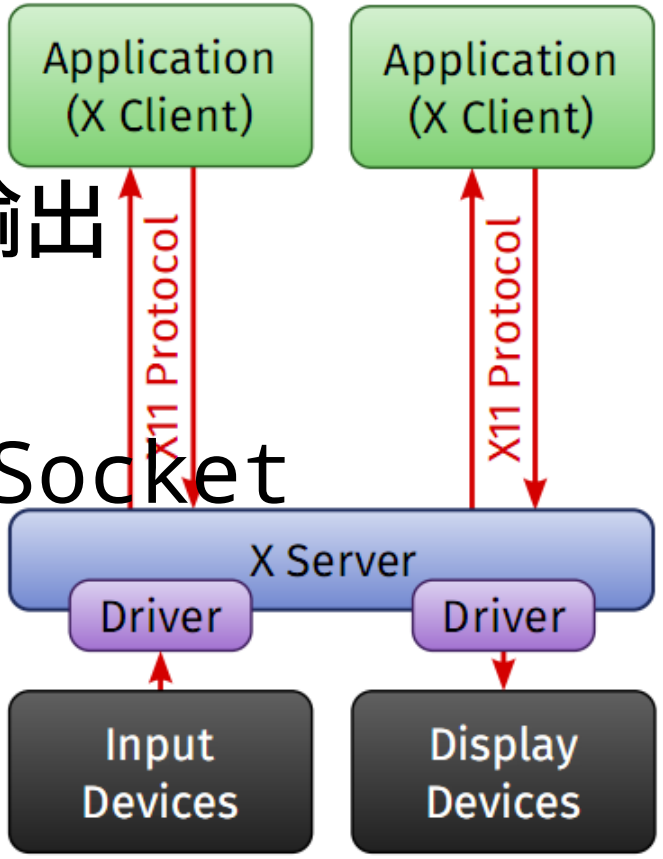
- 在 Windows 執行 cmd
- 輸入 ping 192.168.2.2 看是否有回應？



使用 X11 Forwarding 看圖片

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



http://keyj.emphy.de/files/linuxgraphics_en.pdf

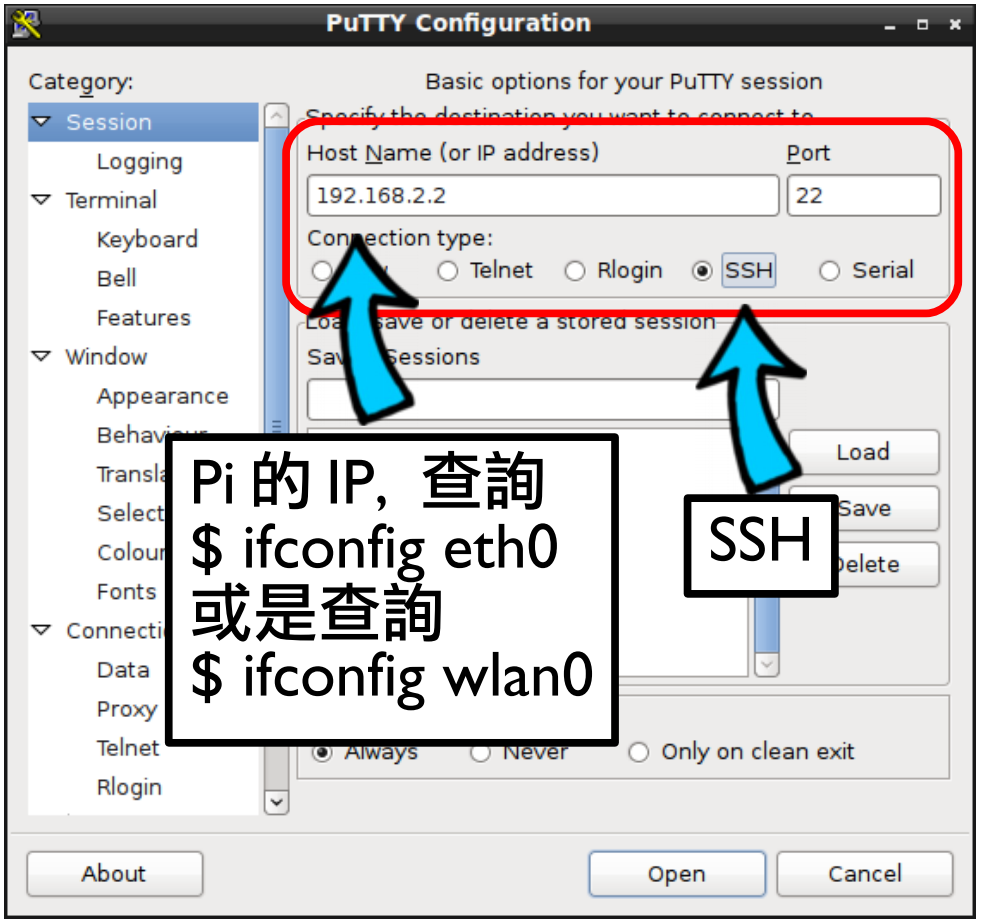
在 Windows 安裝 X Server

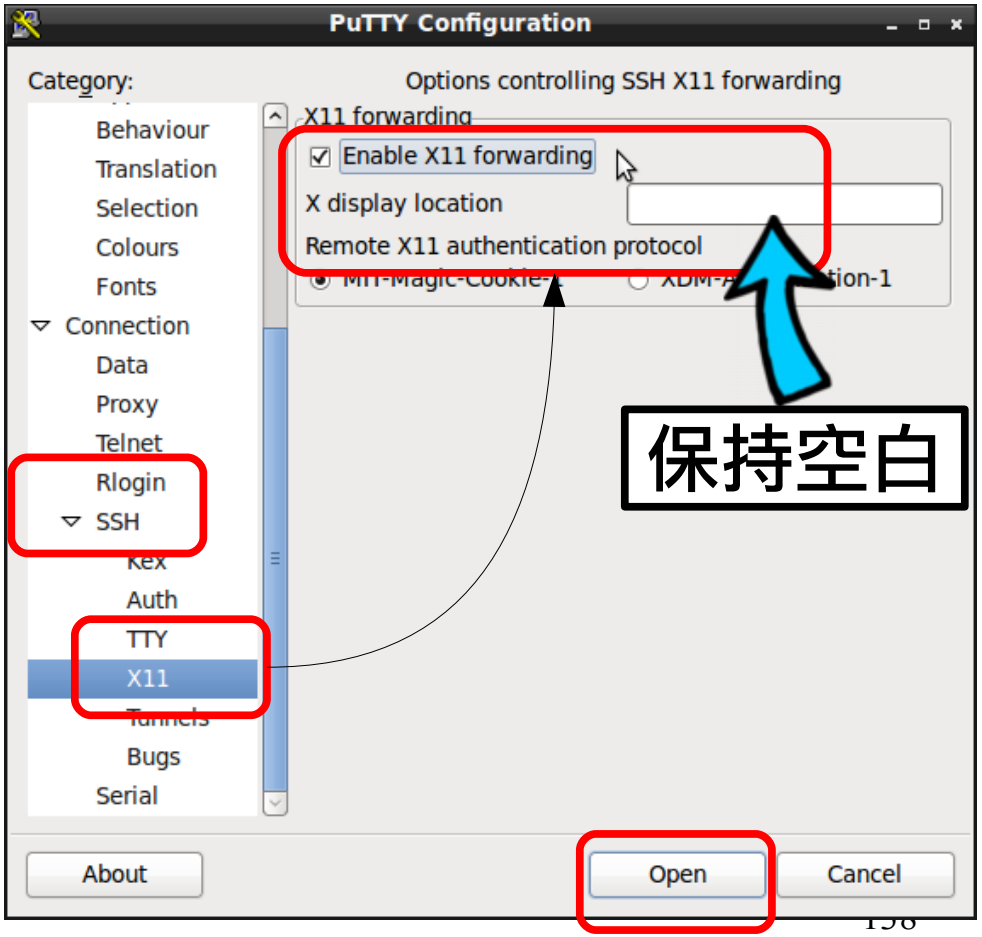
- 安裝 Xming，下一步到底

  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

https://hackpad.com/X11-Forwarding-FcyKHioKxmW

X11 Forwarding 連線成功後

DEMO
image_load.py

```
$ cd ~/pi-follower-car/03-camera
$ python image_load.py lena256rgb.jpg
```

可以即時預覽 Camera 的結果嗎？

讀取 Camera 並顯示

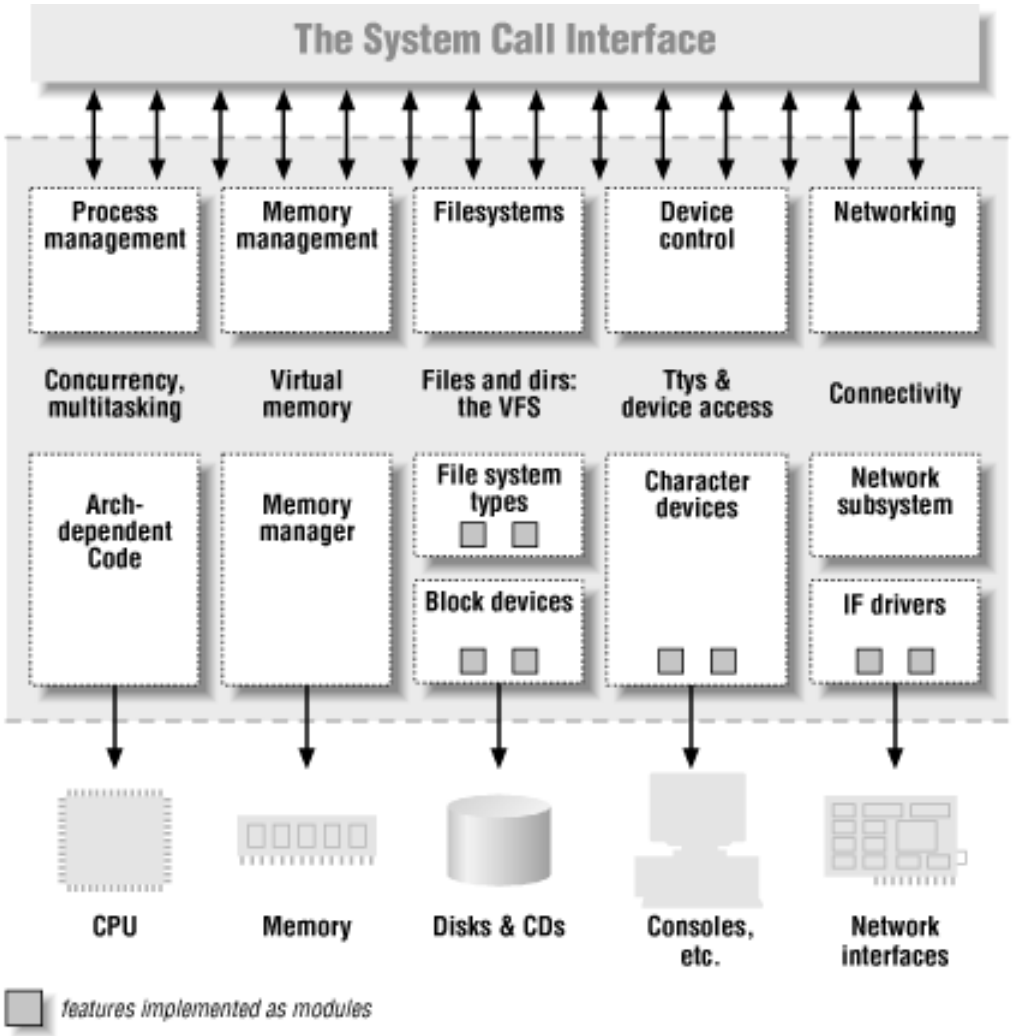
0 是 V4L2 的裝置節點

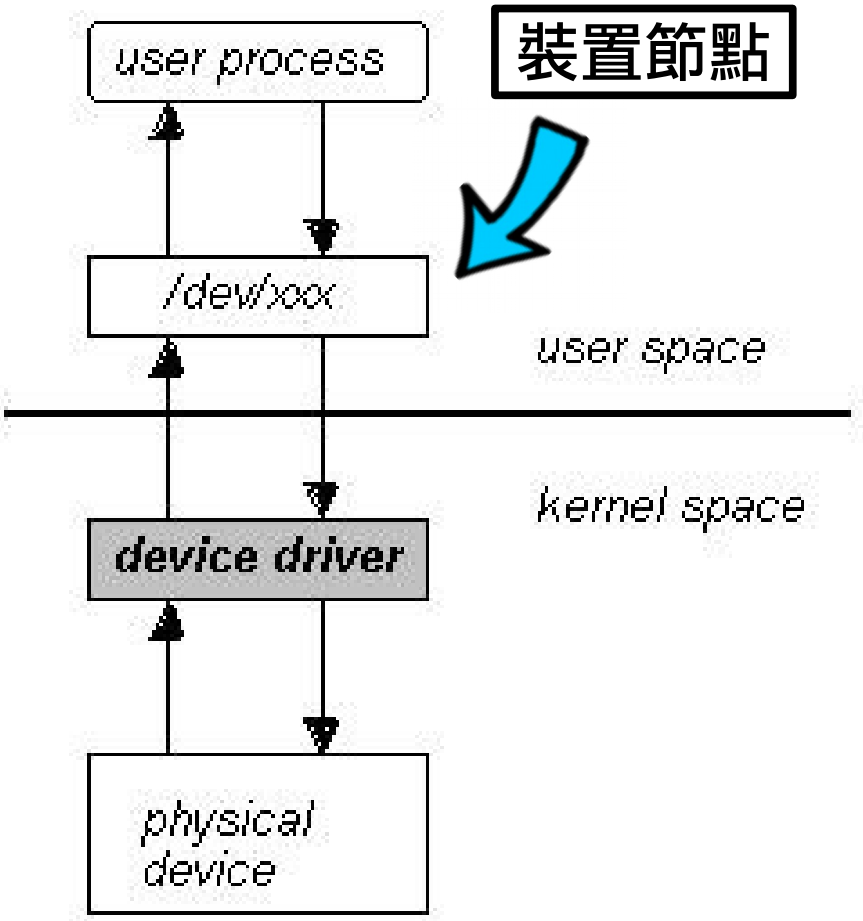
```
import cv2

cap = cv2.VideoCapture(0)
cap.set(cv2.cv.CV_CAP_PROP_FRAME_WIDTH,  320)
cap.set(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT, 240)

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

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

Webcam

http://www.slideshare.net/gxben/abs-2014-android-kit-kat-internals

Linux 如何存取硬體？

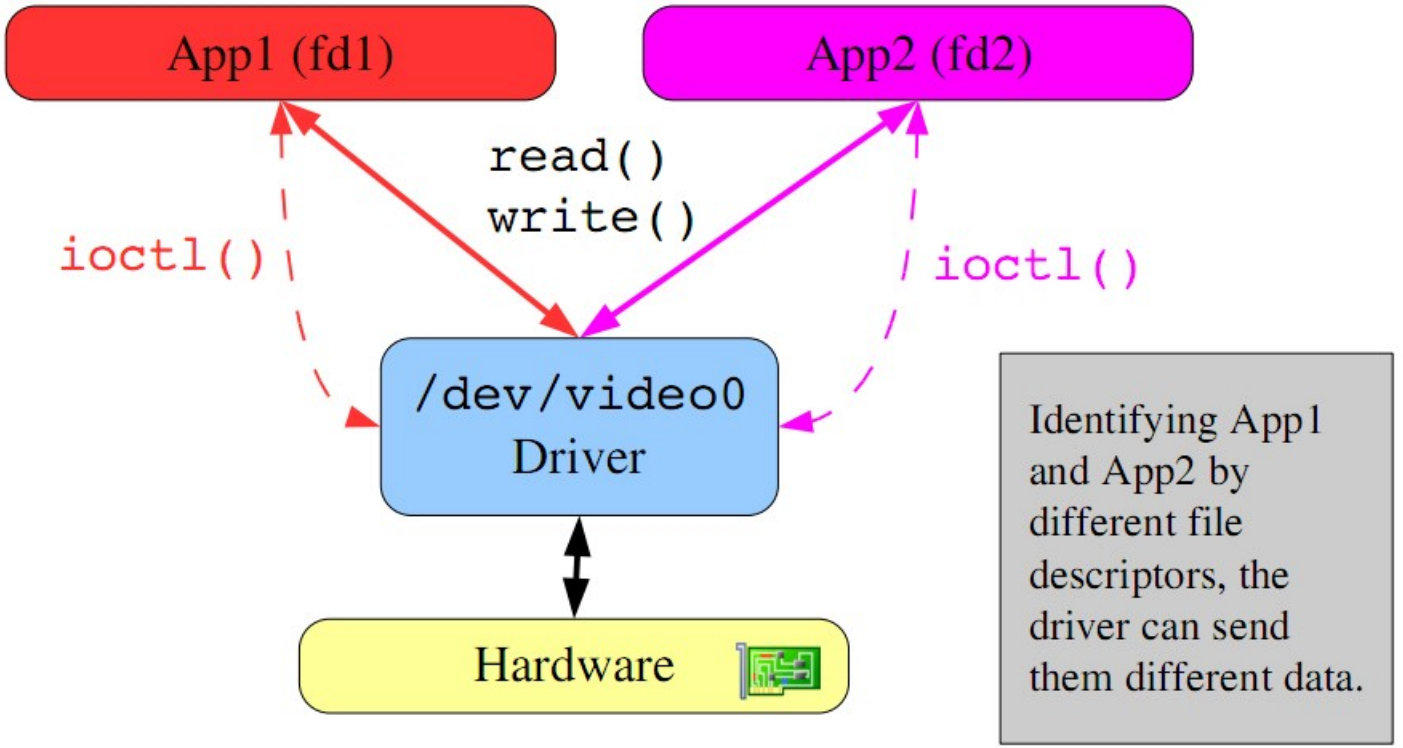




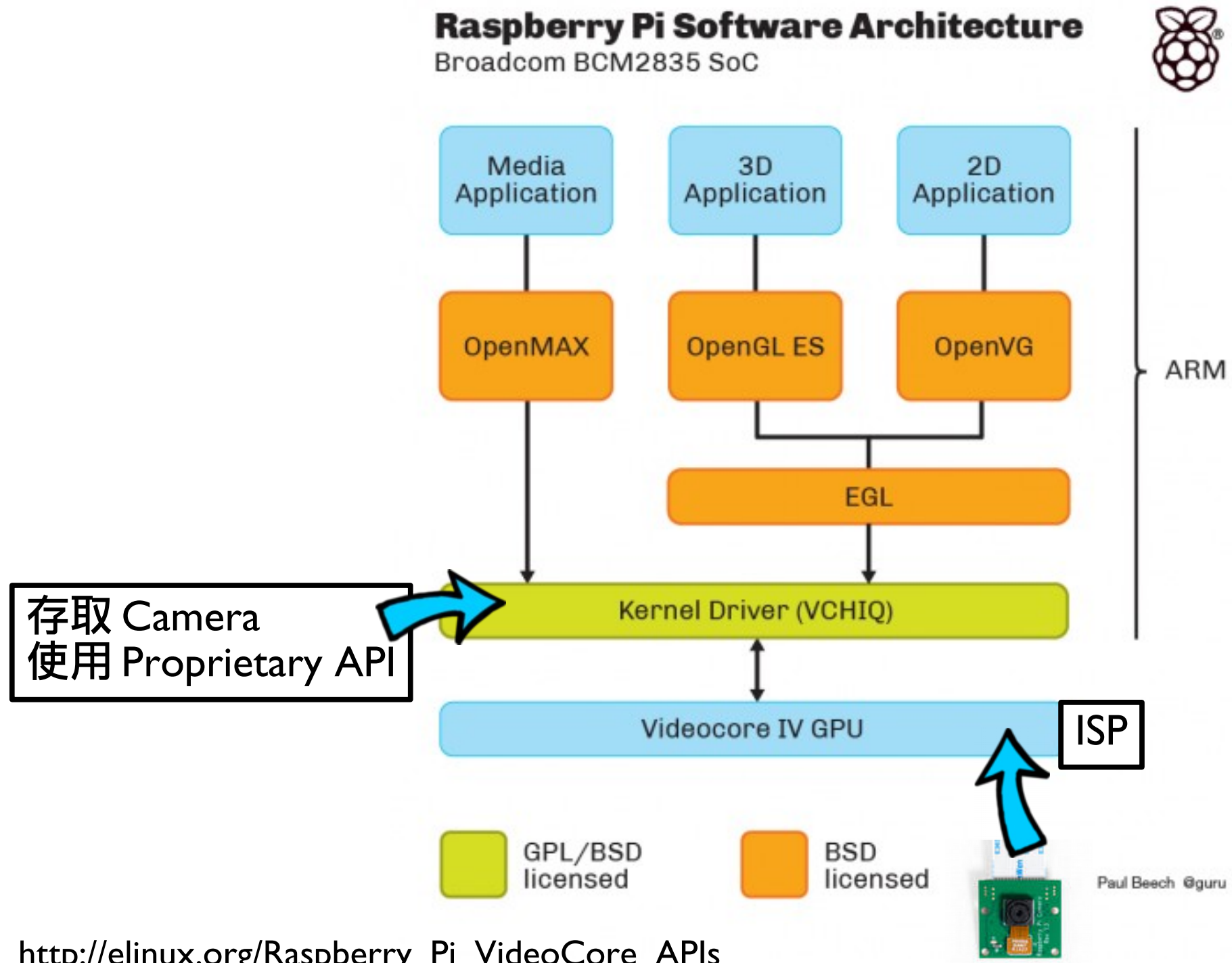
http://www.xml.com/ldd/chapter/book/ch01.html

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
https://www.linuxtv.org/downloads/legacy/video4linux/API/V4L2_API/spec-single/v4l2.html



http://elinux.org/Raspberry_Pi_VideoCore_APIs

官方 V4L2 驅動程式

- Kernel driver
- 使用 camera 像是 webcam 一樣
  - $ sudo modprobe bcm2835-v4l2

不是數字 1，是小寫 L

- 可直接存取 /dev/videoX
  - $ v4l2-ctl --list-devices
  - $ v4l2-ctl --list-formats
  - $ v4l2-ctl -L

https://github.com/raspberrypi/linux/blob/rpi-3.10.y/Documentation/video4linux/bcm2835-v4l2.txt

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 ~/pi-follower-car/03-camera
$ python camera_preview.py
```

小提醒

- 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
```

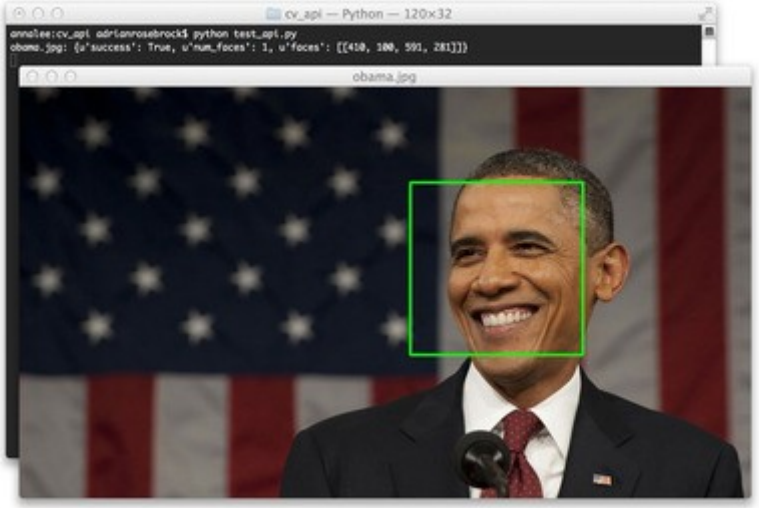
實驗 4-1：色彩空間轉換

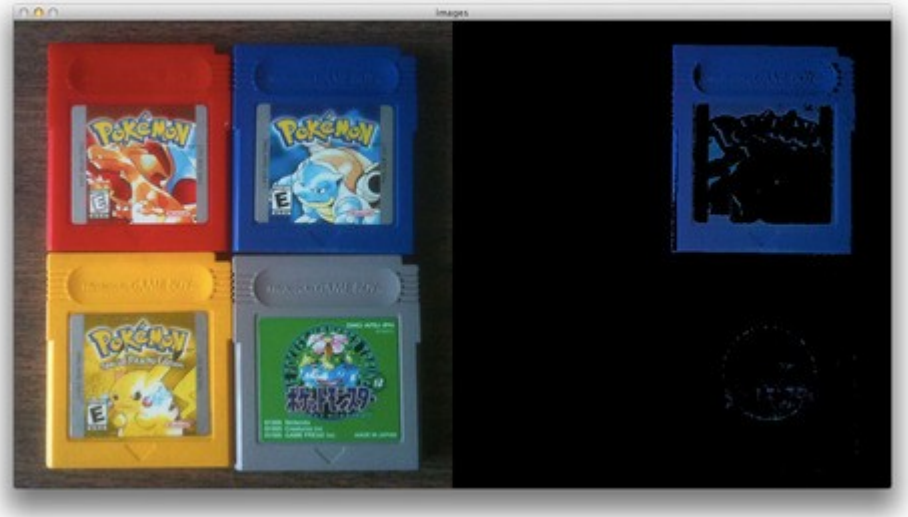
目的：數位影像入門

電腦如何分辨東西？

- 顏色
- 形狀
- 特徵

. . .

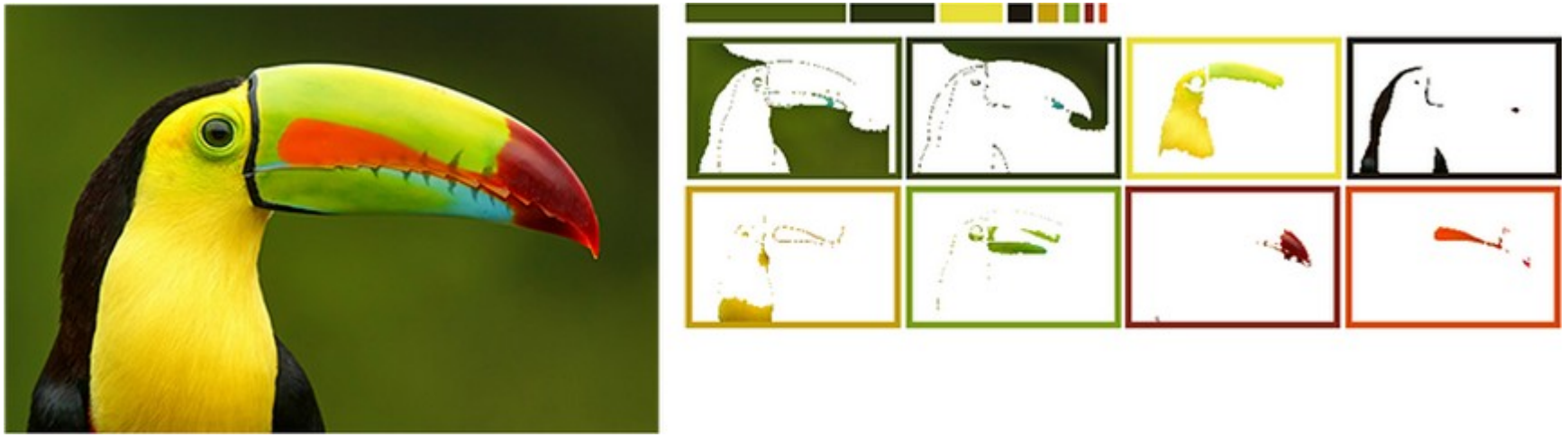




http://www.pyimagesearch.com/

影像處理的意義

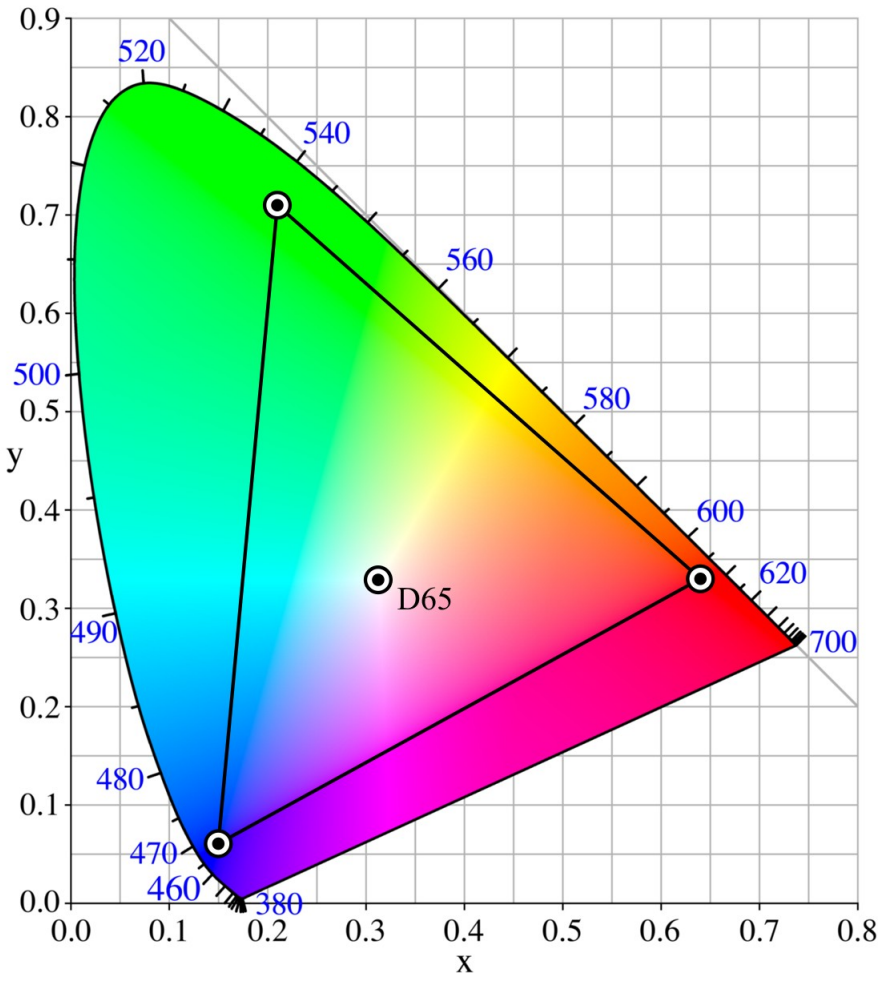
- 分離與萃取資訊
- 增強或平滑訊號
- 作為分群、特徵識別等應用的前處理



http://mkweb.bcgsc.ca/color-summarizer/

色彩空間 (Color Space)

- RGB
- CMY(K)
- YUV(YCbCr)
- HSV

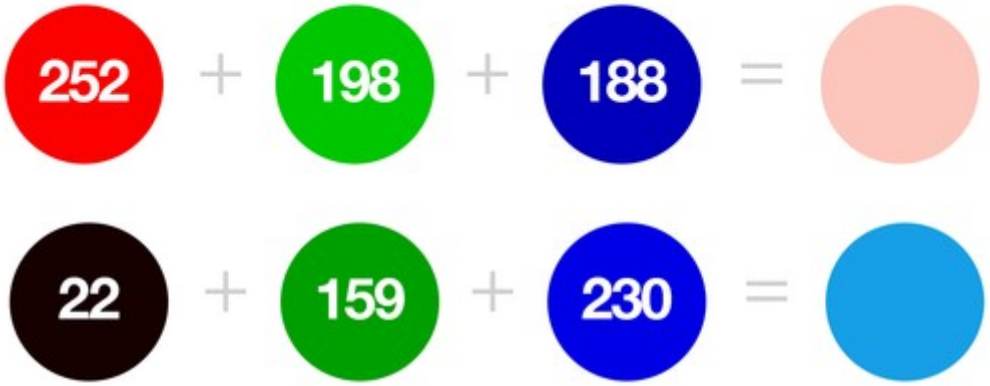


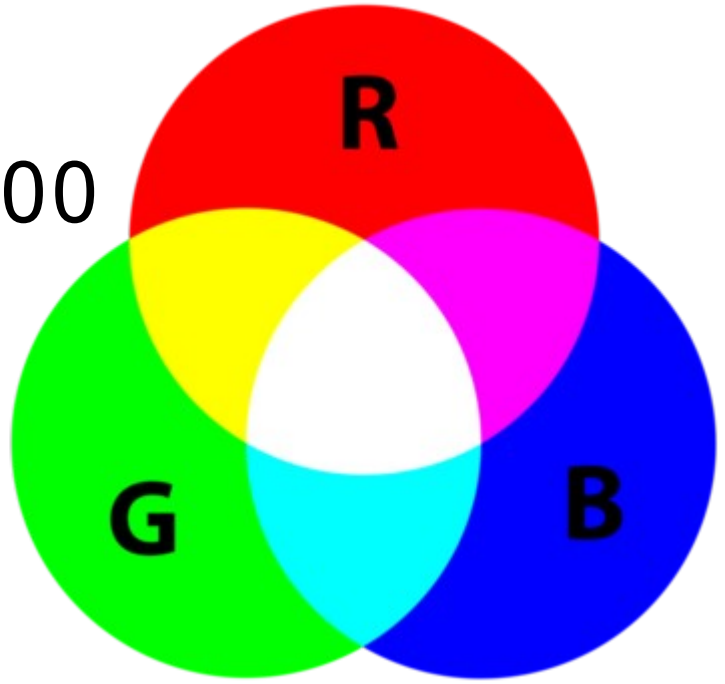
https://en.wikipedia.org/wiki/Adobe_RGB_color_space

RGB

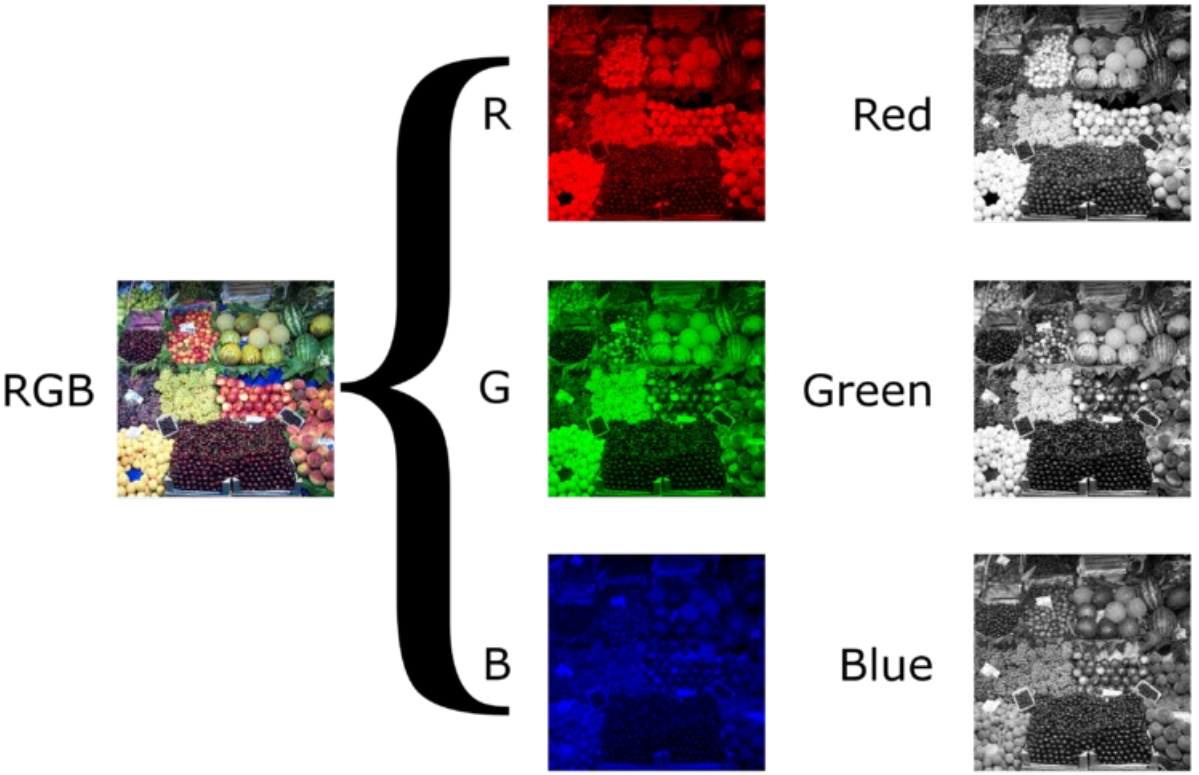
- R(Red), G(Green), B(Blue)

- R(紅),G(綠),B(藍)
- 光的三原色
- 疊加型混色的色彩模型
- 常用在電腦上表現色彩
  - 8-bit:(255,0,0),#FF0000





https://en.wikipedia.org/wiki/RGB_color_model



彩色，灰階，黑白

NUMBERS		
R 255 G 0 B 0	R 102 G 102 B 255	R 51 G 204 B 153
R 255 G 255 B 102	R 255 G 0 B 204	R 51 G 204 B 255
R 51 G 51 B 0	R 51 G 51 B 153	R 255 G 153 B 153

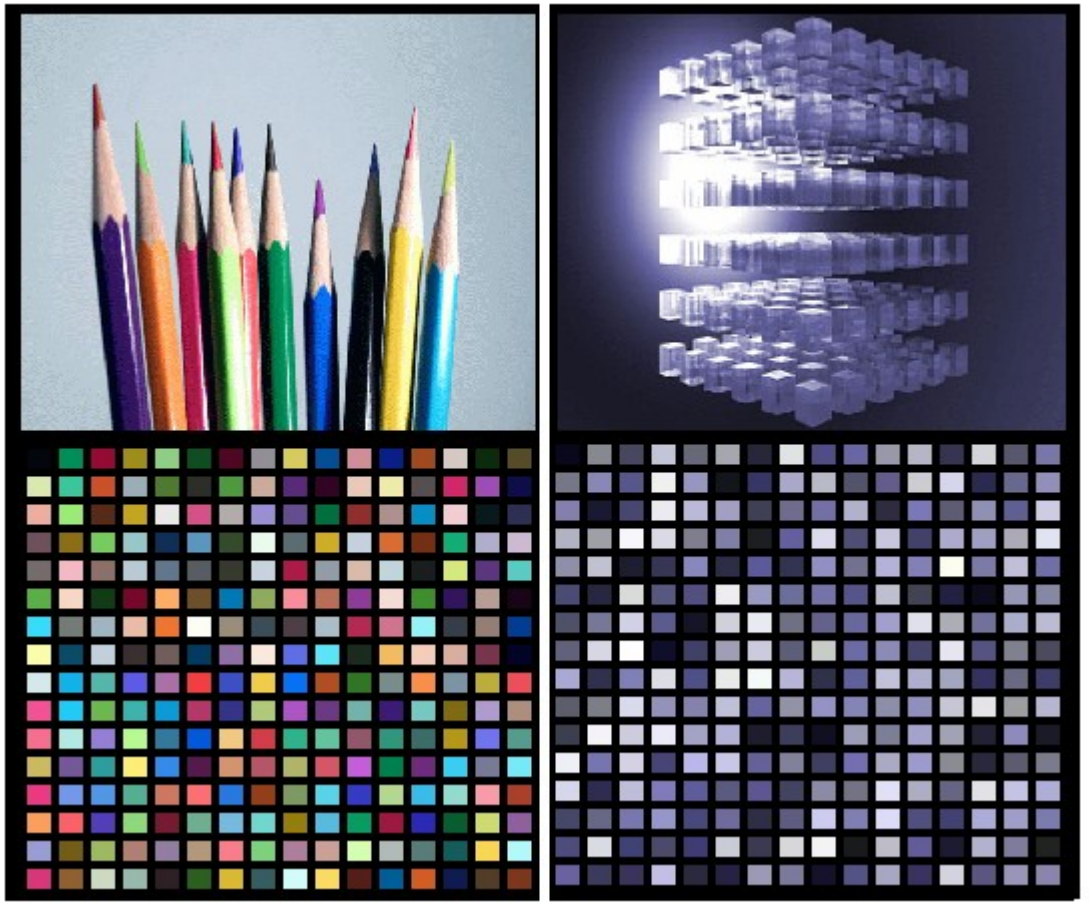
GRAY = 1 SET OF DIGITS		
11111111	11100110	11001101
10110100	10011011	01110011
01010000	00101000	00000000

BLACK & WHITE		
0	1	0
1	0	1
0	1	0

https://en.wikipedia.org/wiki/Grayscale

Indexed Color

- 從全彩 (True Color) 中挑選 256 個顏色

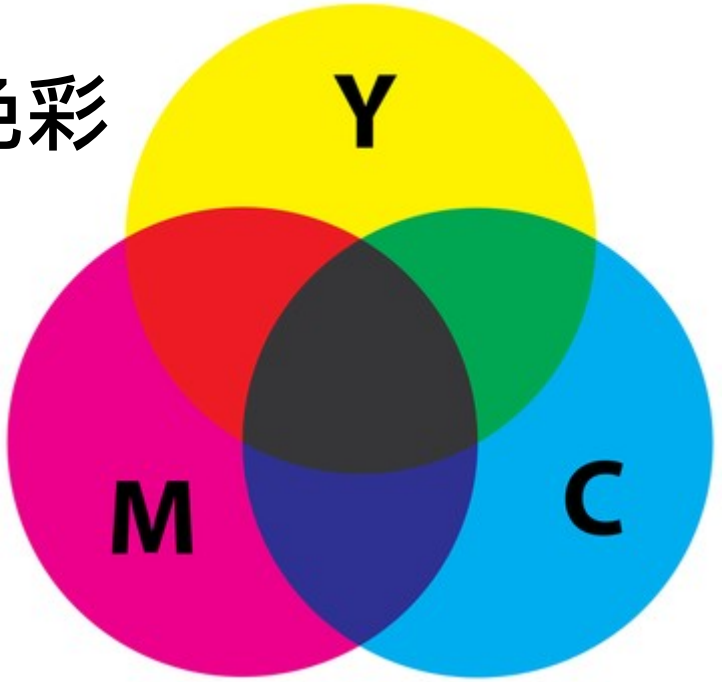


http://goo.gl/xijFPu

CMY(K)

- C(Cyan), M(Magenta), Y(Yellow), K(Black)

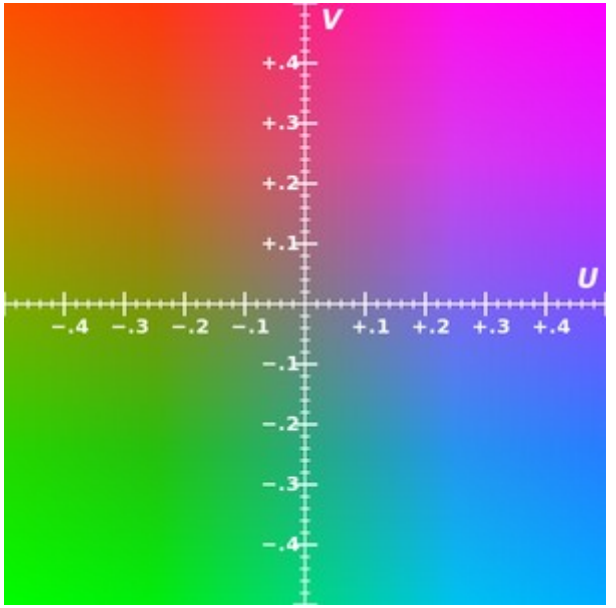
- C(青),M(紫),Y(黃),K(黑)
- 彩色墨水三原色
- 削減型混色的色彩模型
- 常用在印表機 / 影印機表現色彩
- 黑色 ,R=G=B=1



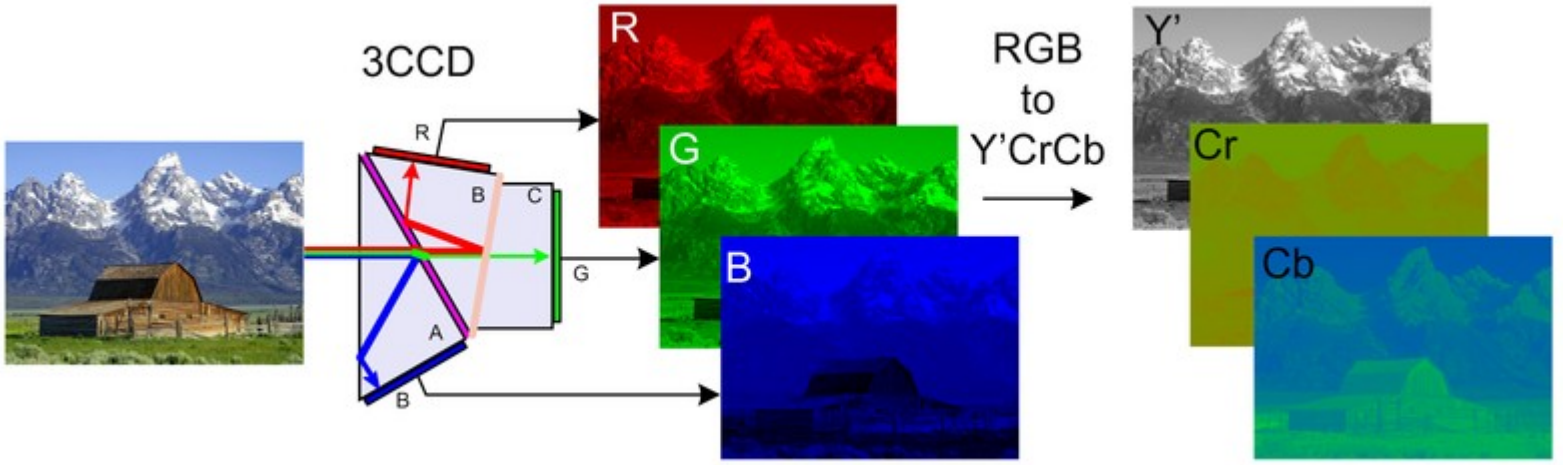
https://en.wikipedia.org/wiki/CMYK_color_model

YUV(YCbCr)

- Y(Luma), UV(Chroma)



- Y(亮度 / 輝度) , UV(色度)
- 常用在電視系統表現色彩
- 源自於 RGB 模型，表示色差訊號
  - YCbCr 是數位色差訊號
  - YPbPr 是類比色差訊號



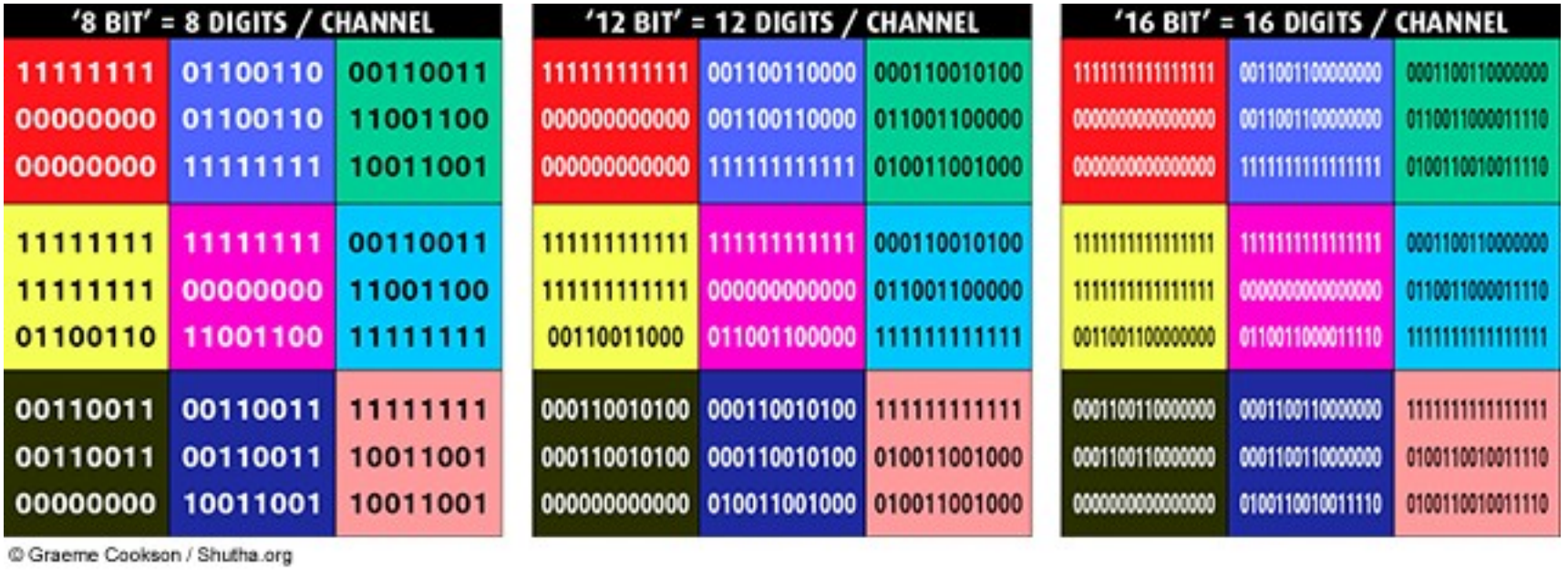
https://en.wikipedia.org/wiki/YUV

不同色彩空間儲存的資料量不相同

© Graeme Cookson / Shutha.org

http://www.shutha.org/node/789

同一色彩空間也有不同的色彩深度



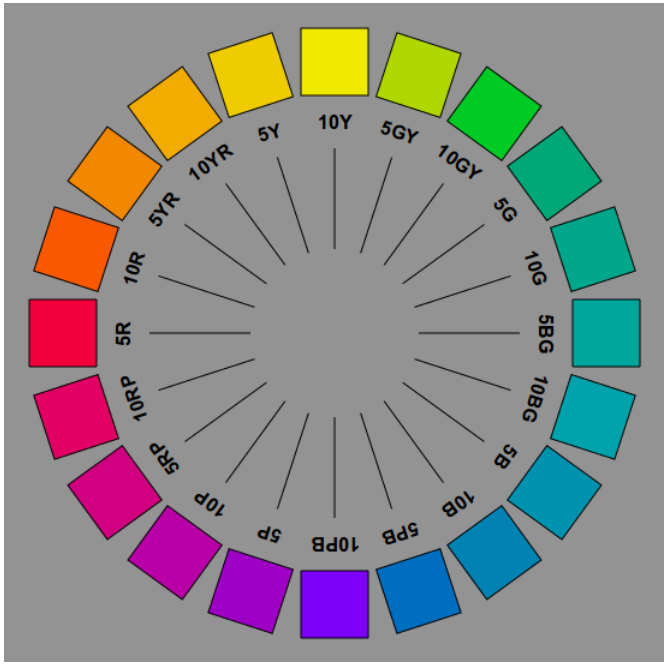
© Graeme Cookson / Shutha.org

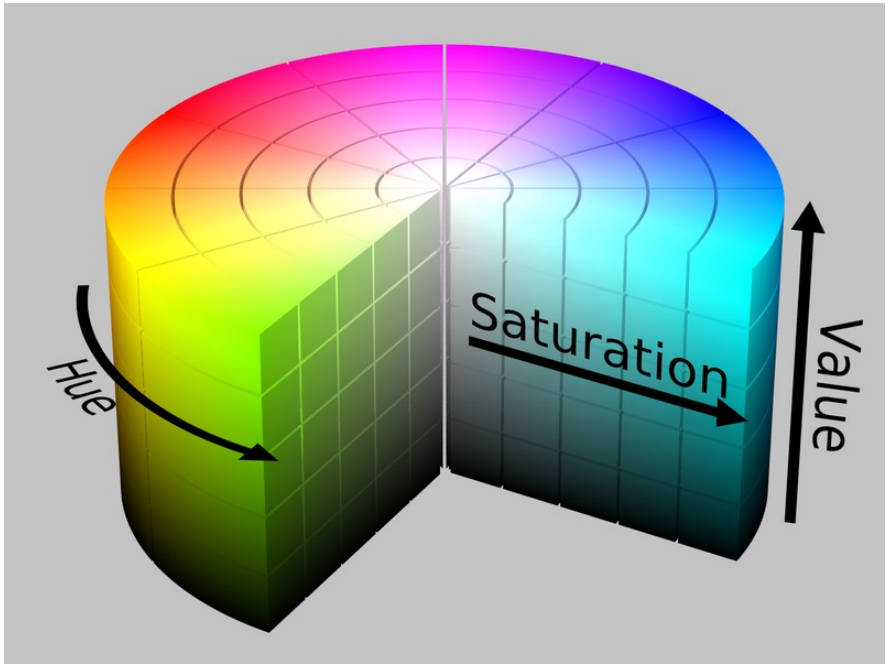
http://www.shutha.org/node/789

HSV

- H(Hue), S(Saturation), V(Value)

- H(彩度 ,0-179)
- S(飽和度 ,0-255)
- V(明度 ,0-255)
- 符合人對顏色的感知，常用在**數位影像處理**





https://en.wikipedia.org/wiki/HSL_and_HSV

色彩空間的轉換

- 以 RGB 為中心
  - RGB to CMY

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1-R \\ 1-G \\ 1-B \end{pmatrix}$$

  - CMY to RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1-C \\ 1-M \\ 1-Y \end{pmatrix}$$

  - RGB to YUV

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

http://isis-data.science.uva.nl/koelma/horus/dox/horus2.0/refcpp/html/HxColConvert_8h.html

色彩空間的轉換

- 以 RGB 為中心
  - HSV to RGB

$$V \leftarrow \max(R, G, B)$$

$$S \leftarrow \begin{cases} \frac{V - \min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H \leftarrow \begin{cases} 60(G - B)/(V - \min(R, G, B)) & \text{if } V = R \\ 120 + 60(B - R)/(V - \min(R, G, B)) & \text{if } V = G \\ 240 + 60(R - G)/(V - \min(R, G, B)) & \text{if } V = B \end{cases}$$

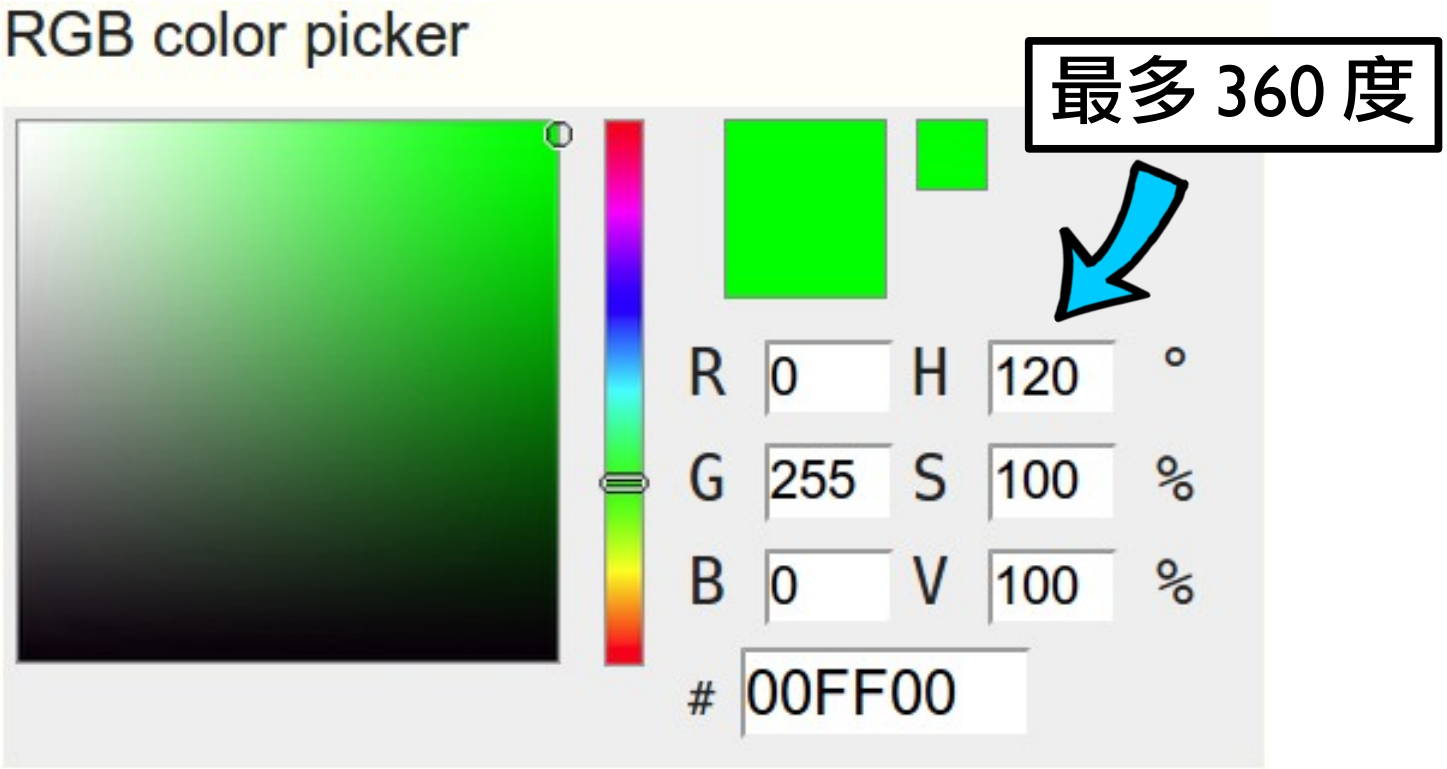
Color	Color name	(H,S,V)	Hex	(R,G,B)
	Black	(0°,0%,0%)	#000000	(0,0,0)
	White	(0°,0%,100%)	#FFFFFF	(255,255,255)
	Red	(0°,100%,100%)	#FF0000	(255,0,0)
	Lime	(120°,100%,100%)	#00FF00	(0,255,0)
	Blue	(240°,100%,100%)	#0000FF	(0,0,255)
	Yellow	(60°,100%,100%)	#FFFF00	(255,255,0)
	Cyan	(180°,100%,100%)	#00FFFF	(0,255,255)
	Magenta	(300°,100%,100%)	#FF00FF	(255,0,255)
	Silver	(0°,0%,75%)	#C0C0C0	(192,192,192)
	Gray	(0°,0%,50%)	#808080	(128,128,128)
	Maroon	(0°,100%,50%)	#800000	(128,0,0)
	Olive	(60°,100%,50%)	#808000	(128,128,0)
	Green	(120°,100%,50%)	#008000	(0,128,0)
	Purple	(300°,100%,50%)	#800080	(128,0,128)
	Teal	(180°,100%,50%)	#008080	(0,128,128)
	Navy	(240°,100%,50%)	#000080	(0,0,128)

http://www.rapidtables.com/convert/color/hsv-to-rgb.htm

線上轉換工具

- http://goo.gl/EV4l0z

- 將 RGB 的綠色轉成 HSV



除了看圖也看值

- 在 Python 中如何看 RGB 的綠色轉成 HSV 是多少？
- $ python

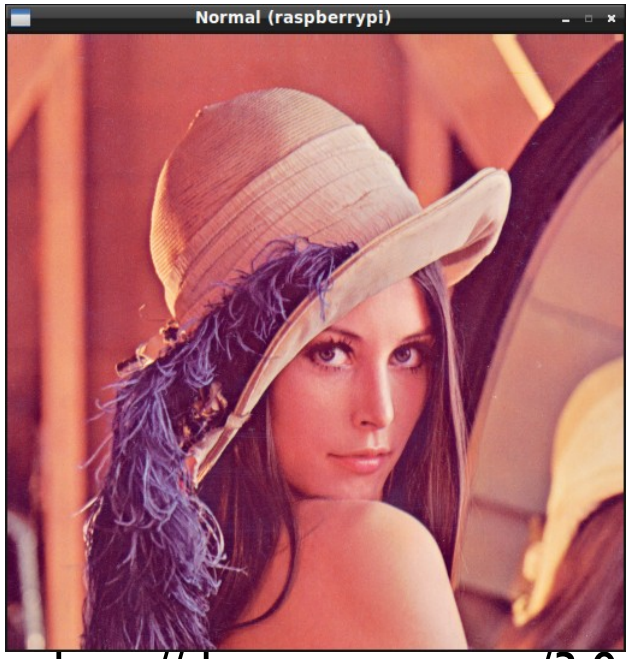
```
>>> import cv2
>>> import numpy as np
>>> green = np.array([[[0,255,0]]], dtype='uint8')
>>> hsv_green = cv2.cvtColor(green, cv2.COLOR_BGR2HSV)
>>> print hsv_green
[[[ 60 255 255]]]
```

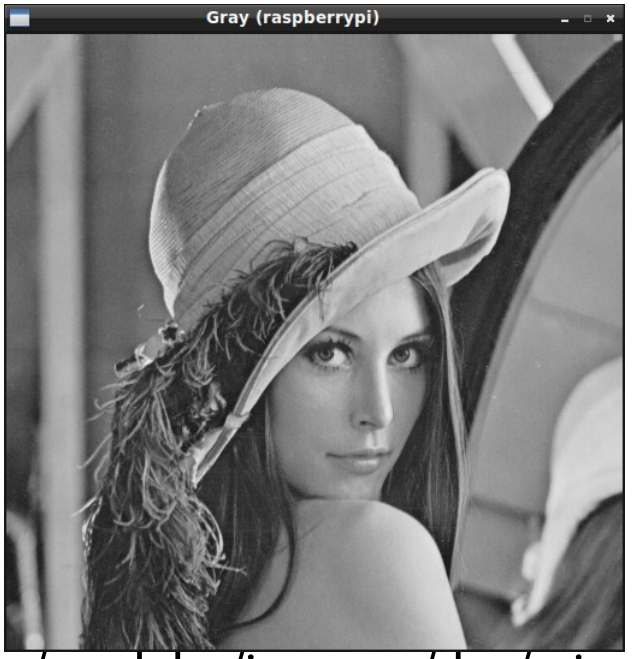
0-180

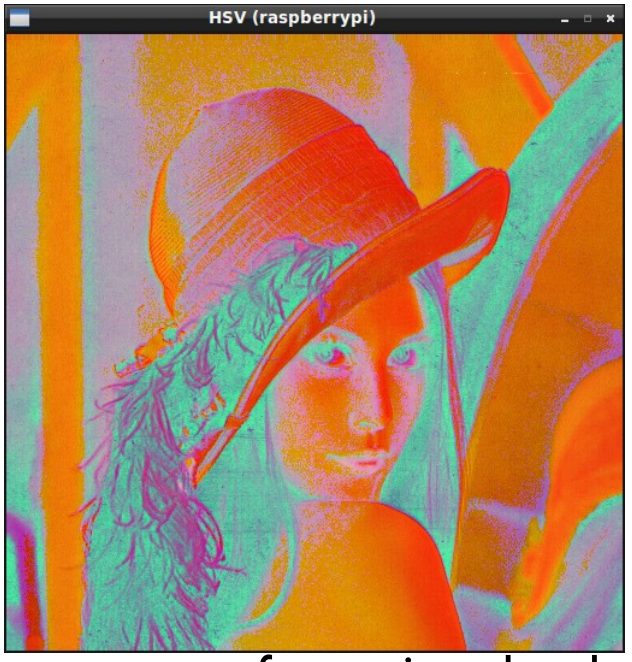
http://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html

色彩空間的轉換

- `cv2.cvtColor(img,flag)`
  - RGB 轉灰階 ,`flag = cv2.COLOR_BGR2GRAY`
  - RGB 轉 HSV,`flag = cv2.COLOR_BGR2HSV`
  - HSV 轉 RGB,`flag = cv2.COLOR_HSV2BGR`







http://docs.opencv.org/3.0-beta/modules/imgproc/doc/miscellaneous_transformations.html

色彩空間轉換

```
image = cv2.imread(“lena256rgb.jpg”)
cv2.imshow("Normal", image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)
```

BGR 轉灰階

```
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV", hsv)
```

BGR 轉 HSV

```
bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
cv2.imshow("BGR", bgr)
```

HSV 轉 BGR

影像出現後，按任意鍵會顯示下個影像

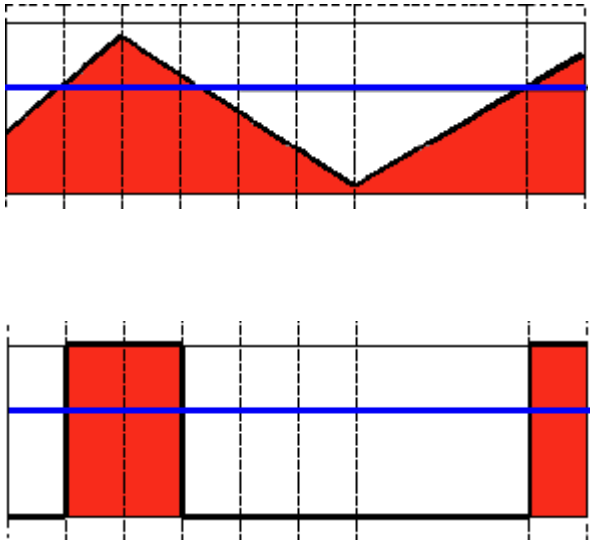
DEMO
color_space.py

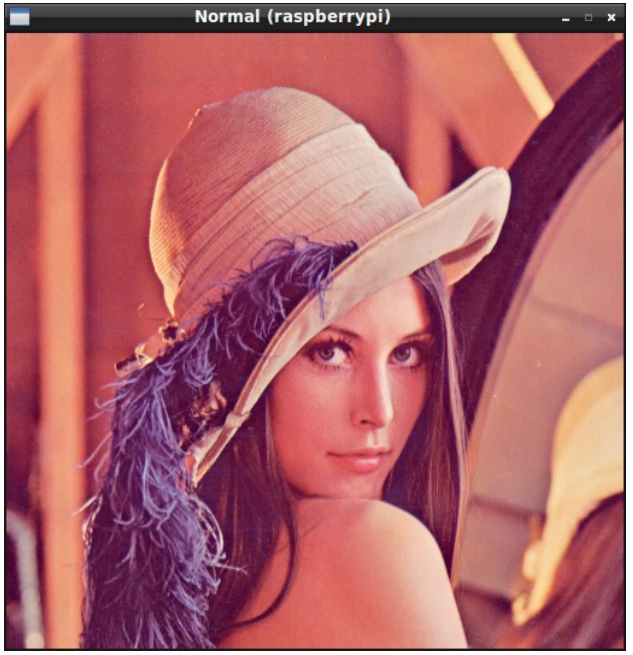
```
$ cd ~/camera-opencv/04-color_space
$ python color_space.py
```

二值化

- 將影像以強度 (threshold) 區分

- `cv2.threshold(img,thresh,maxval,type)`
- 從灰階轉成黑白，會有兩個回傳值
  - `cv2.threshold(gray,127,255,cv2.THRESH_BINARY)`





http://docs.opencv.org/3.0-beta/modules/imgproc/doc/miscellaneous_transformations.html

練習

- 修改 color_space.py 將灰階影像轉為二值化影像

```
# Convert BGR to Gray
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)
print "Gray image..."
cv2.waitKey(0)

# Threshold the gray image to binary image
# ret, binary = cv2.threshold(...)
```

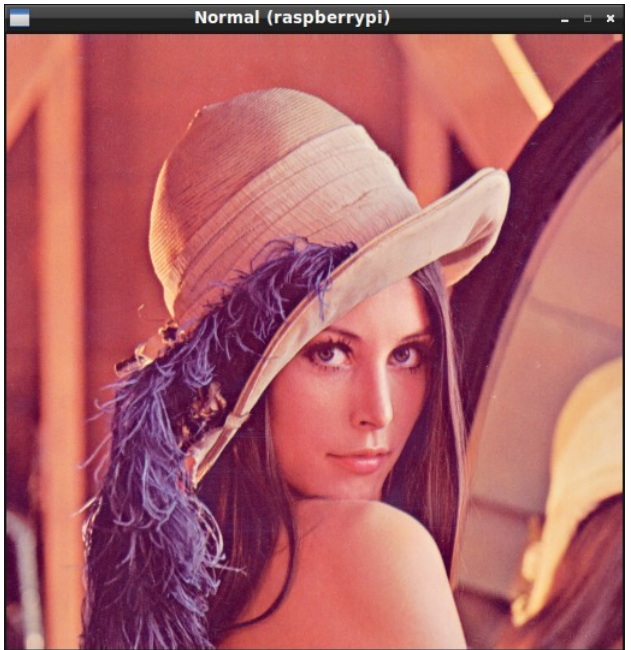
兩個回傳值

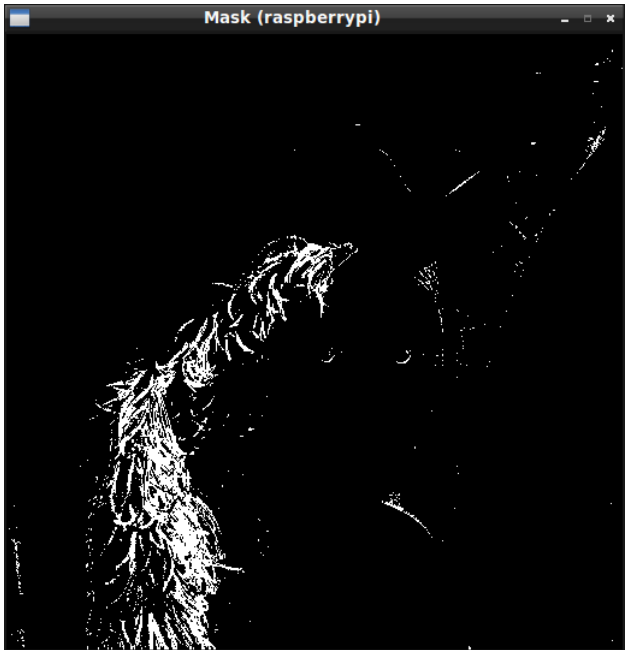
請完成這邊

另一種二值化

- 在 HSV 空間裡，根據區間 (lower/upper) 分割

- `cv2.inRange(img,lowerval,upperval)`
- 範例：只取出紫色部份
  - `lower=np.array([141,0,0])`
  - `upper=np.array([164,145,197])`
  - `binary=cv2.inRange(hsv,lower,upper)`





色彩空間轉換與二值化處理

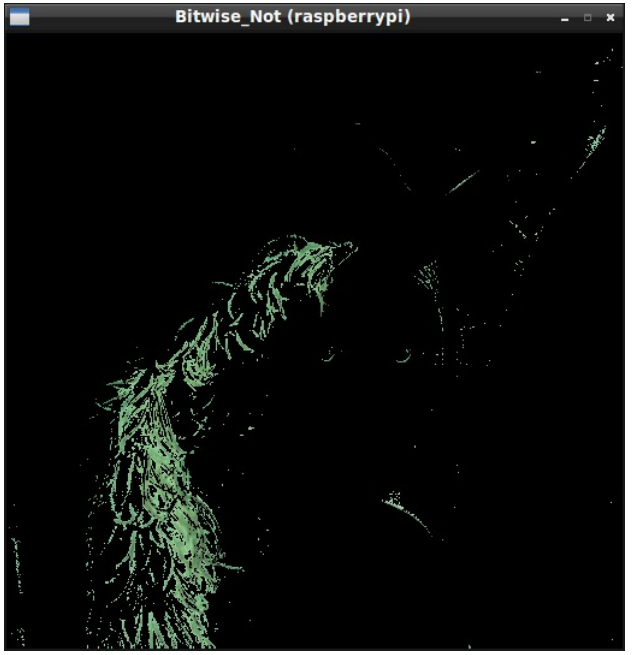
```
image = cv2.imread("lena256rgb.jpg")

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
lower = np.array( [141, 0, 0] )
upper = np.array( [164, 145, 197] )
binary = cv2.inRange(hsv, lower, upper)

cv2.imshow("Binary", binary)
print "HSV Binary image..."
cv2.waitKey(0)
```

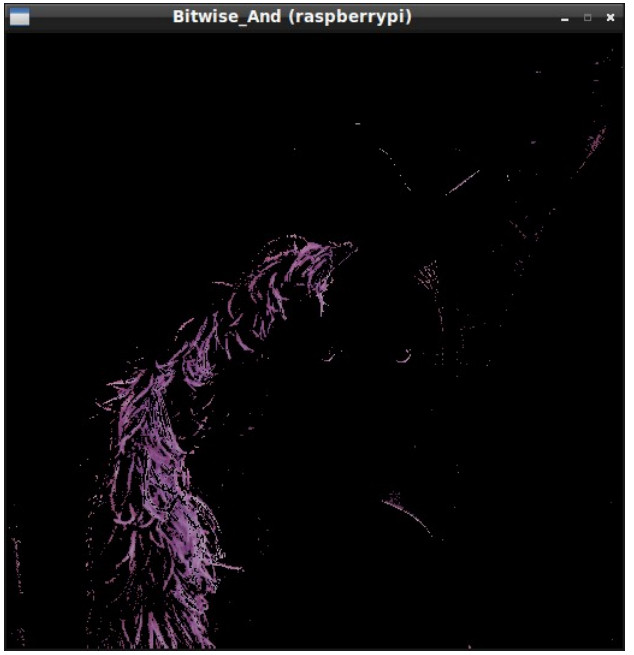
DEMO
hsv_binary.py

```
$ cd ~/camera-opencv/04-color_space
$ python hsv_binary.py
```

位元運算處理 (Bitwise)

- `cv2.bitwise_not(mask)  # 反向 mask 結果`
- `cv2.bitwise_not(src, mask)`
- `cv2.bitwise_and(src, dst, mask)`





http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_image_arithmetics/py_image_arithmetics.html

原圖與遮罩相疊

```
image = cv2.imread("lena256rgb.jpg")

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
lower = np.array([141, 0, 0])
upper = np.array([164, 145, 197])
binary = cv2.inRange(hsv, lower, upper)

bitwise_not = cv2.bitwise_not(binary)
cv2.imshow("Bitwise_not", bitwise_not)

bitwise_not = cv2.bitwise_not(image, mask=binary)
cv2.imshow("Bitwise_not", bitwise_not)
```

src dst mask

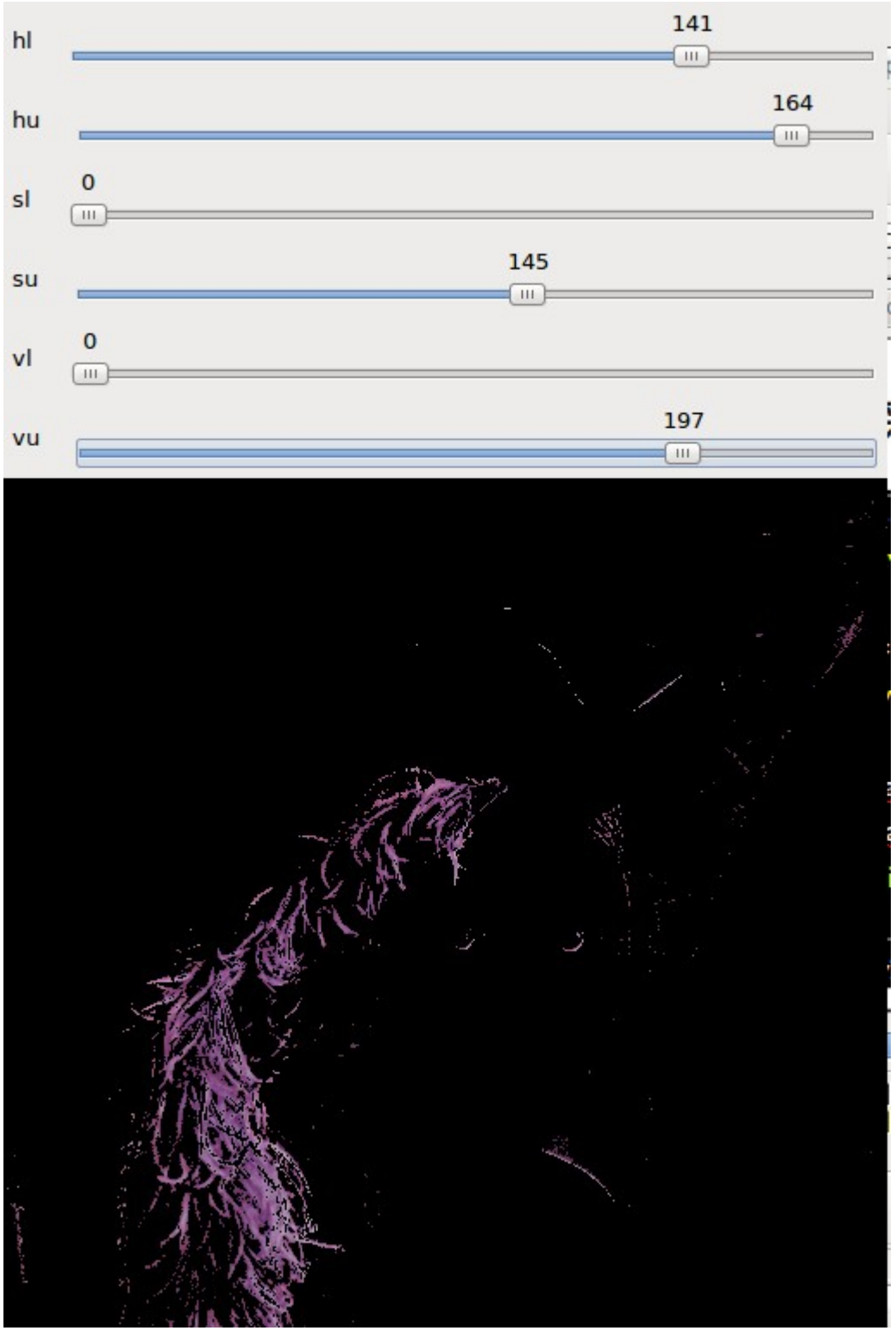
```
bitwise_and = cv2.bitwise_and(image, image, mask=binary)
cv2.imshow("Bitwise_and", bitwise_and)
```

DEMO
hsv_mask.py

```
$ cd ~/camera-opencv/04-color_space
$ python hsv_mask.py
```

HSV 的值？

- 即時調整，按 q 離開
- 受光線影響大



建立拉桿與顯示即時結果

```
cv2.namedWindow('hsv_demo')
h, s, v = 100, 100, 100
cv2.createTrackbar('hl', 'hsv_demo', 0,   179, nothing)
cv2.createTrackbar('hu', 'hsv_demo', 179, 179, nothing)

while True:
    hl = cv2.getTrackbarPos('hl', 'hsv_demo')
    hu = cv2.getTrackbarPos('hu', 'hsv_demo')


    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower = np.array([hl, sl, vl])
    upper = np.array([hu, su, vu])
    mask = cv2.inRange(hsv, lower, upper)
    result = cv2.bitwise_and(image, image, mask=mask)

    cv2.imshow("hsv_demo", result)
```

建立拉桿

讀取拉桿數值

DEMO
hsv_value.py

```
$ cd ~/pi-follower-car/04-opencv
$ python hsv_value.py lena256rgb.jpg
```

練習

- 執行以下指令，找出黃色，並紀錄 HSV 的 upper 和 lower
- `$ python hsv_value.py balloon.jpg`

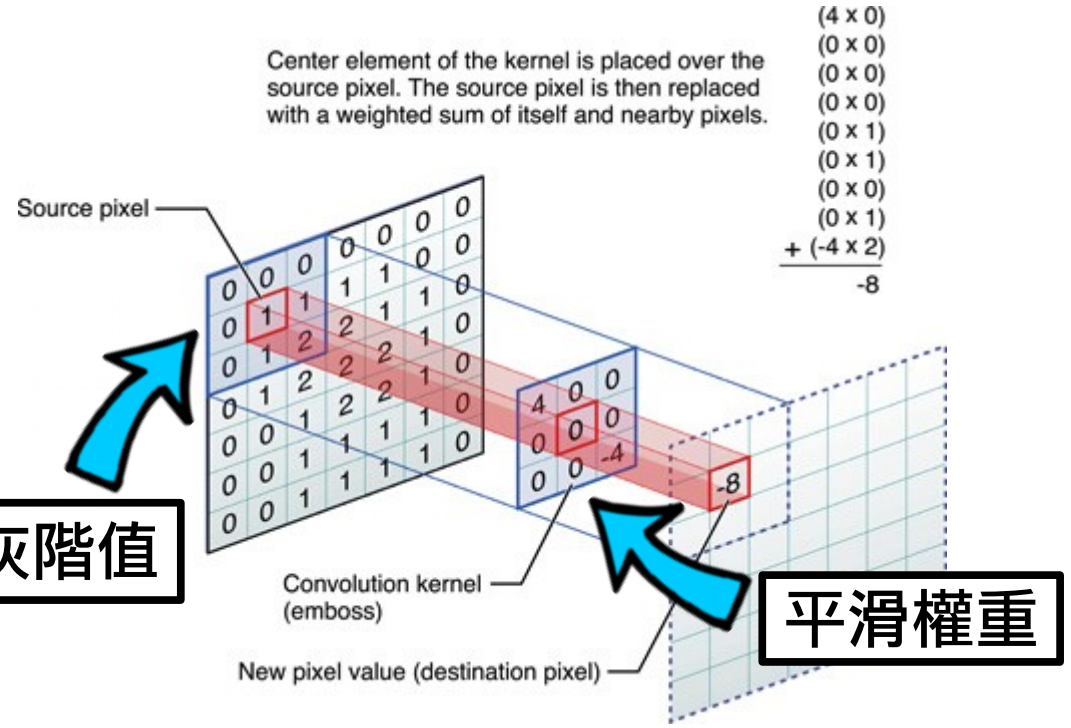
https://pixabay.com/en/balloon-year-market-bloat-2216318/

實驗 4-2：常用影像處理

目的：數位影像處理方法

影像平滑

- 目的是為了去除雜訊，但對比度可能下降
- 每次要處理的像素區域稱為 Kernel



kernel，數字為灰階值

平滑權重

http://goo.gl/dck6tD

影像平滑方法

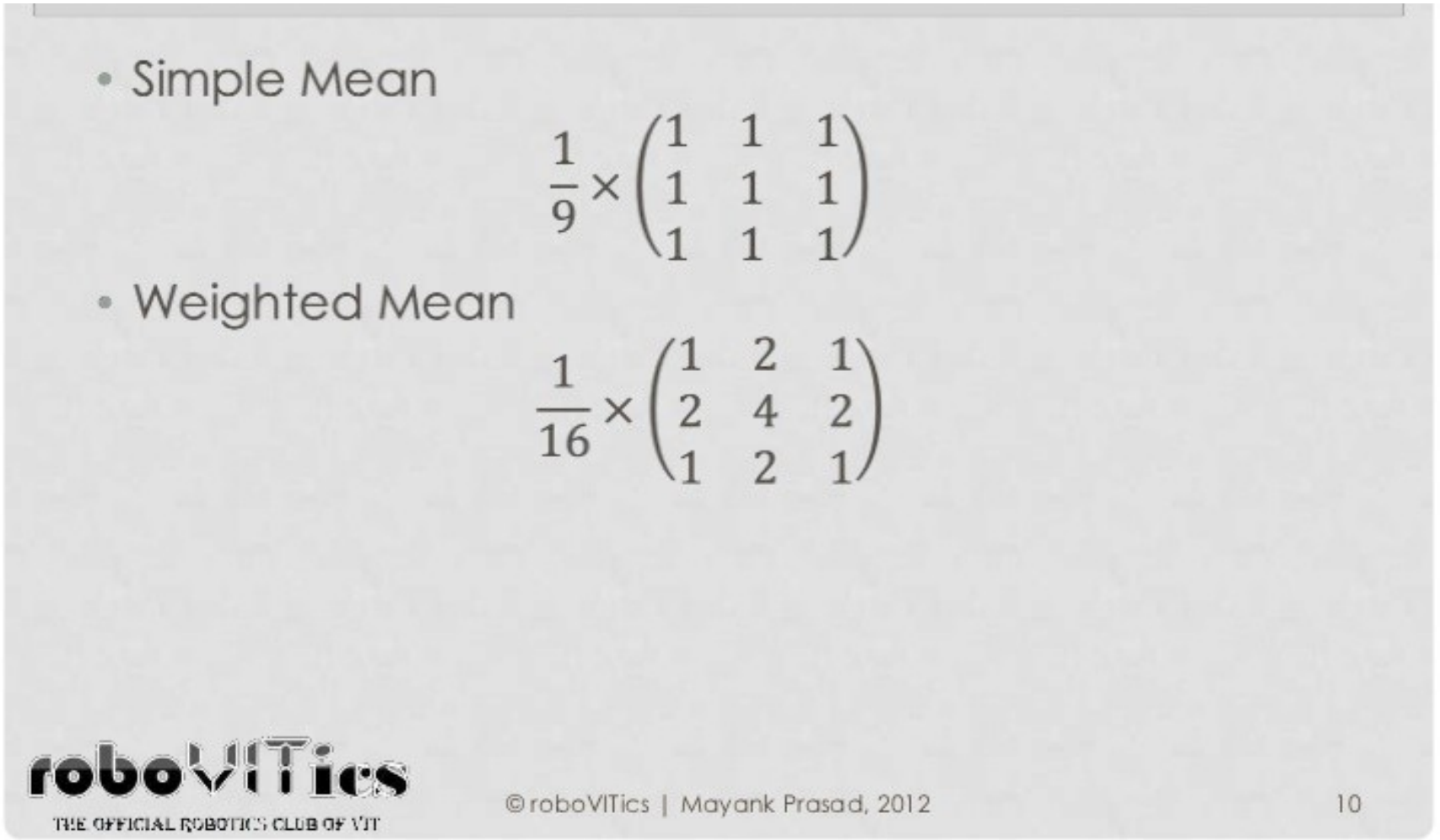
- 濾波器 (平滑化)
- 侵蝕 (Erode)
- 膨脹 (Dilate)

常見濾波器

- 線性濾波
  - 平均平滑 (blur)
  - 高斯平滑 (GaussianBlur)
- 非線性濾波
  - 中值濾波 (medianBlur)
  - 雙邊濾波 (bilateralFilter)

平均平滑 (Blur)

- Kernel 裡的權重都是 1



https://www.slideshare.net/roboVITics/robocv-module-4-image-processing-techniques-using-opencv

Original Image

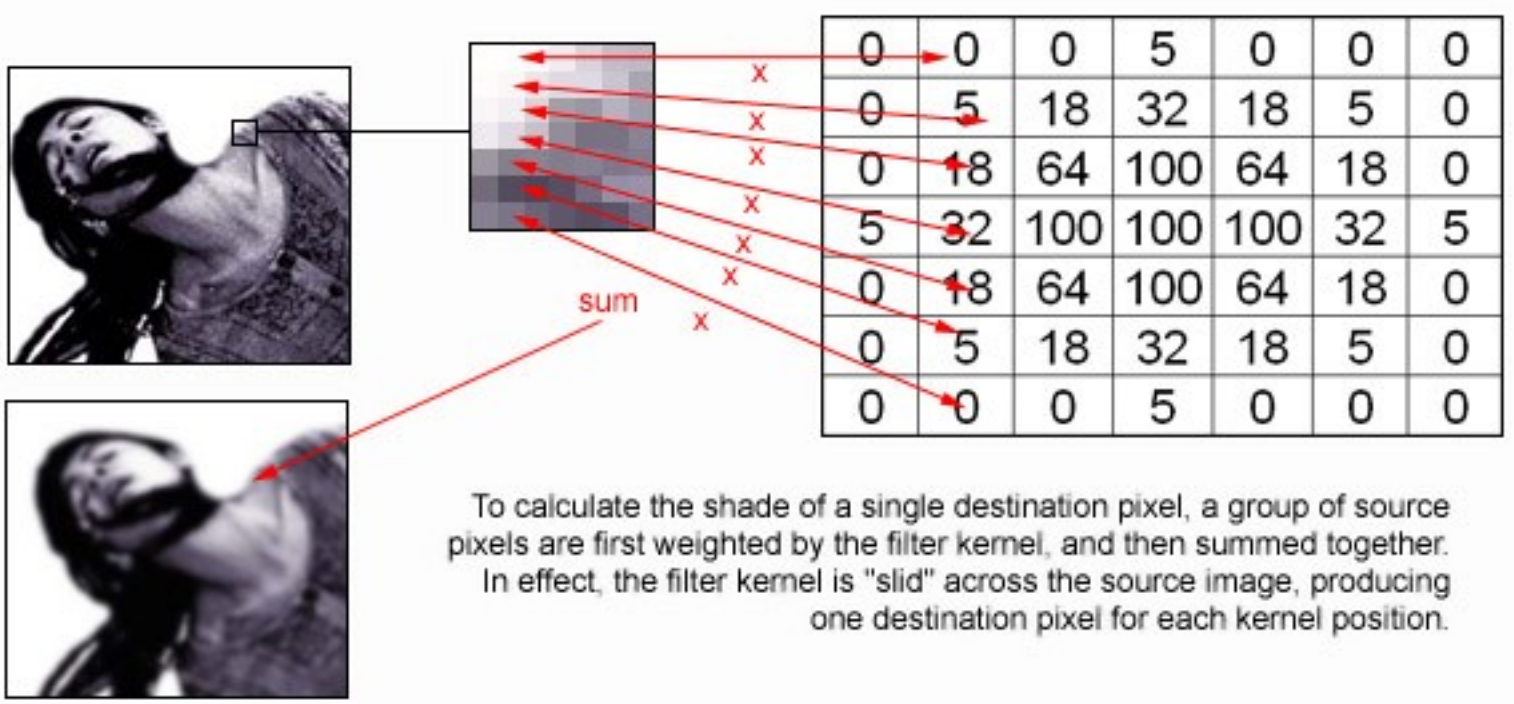
Box-filtered image

Gaussian-filtered image

Kernels used for blurring:
(note that the values shown have been scaled up to integers for clarity)

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0



0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0

To calculate the shade of a single destination pixel, a group of source pixels are first weighted by the filter kernel, and then summed together. In effect, the filter kernel is "slid" across the source image, producing one destination pixel for each kernel position.

http://www.gamasutra.com/view/feature/131511/four_tricks_for_fast_blurring_in_.php

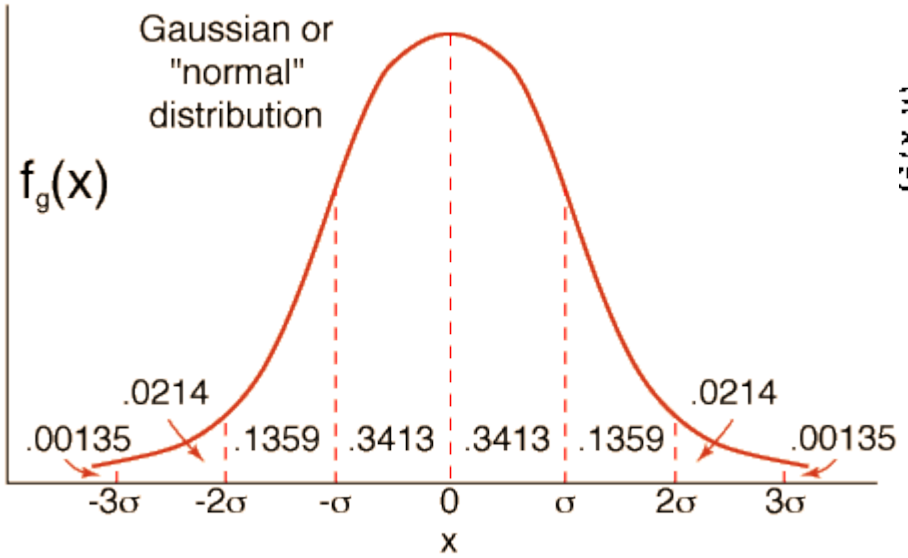
高斯平滑 (GaussianBlur)

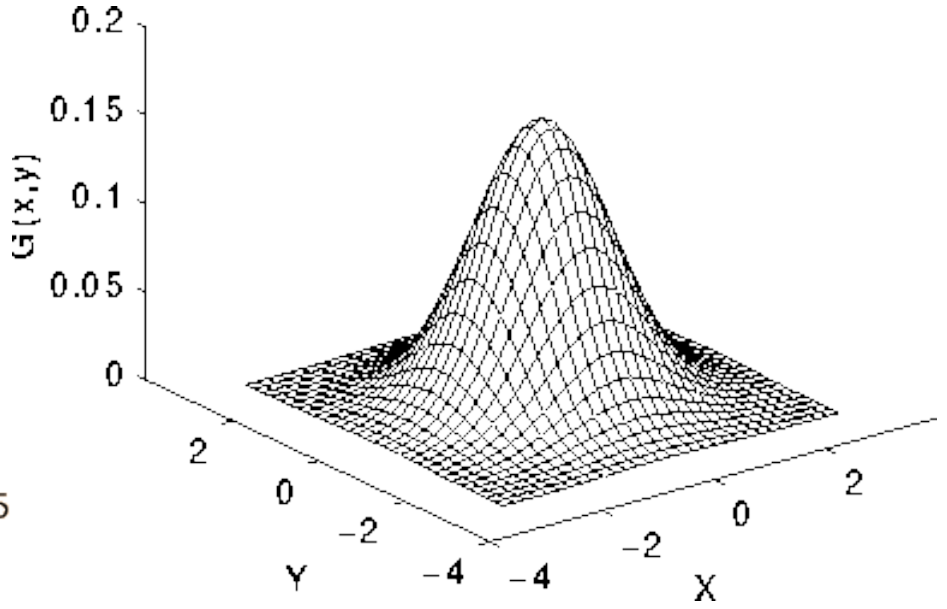
- `cv2.GaussianBlur(src, ksize, sigmaX)`

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

一維計算公式

標準差





http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

高斯平滑效果

```
cv2.namedWindow('Gaussian_Blur')
cv2.createTrackbar('ksize', 'Gaussian_Blur', 0, 10, nothing)

image = cv2.imread("lena512rgb.png")

while True:
    ksize  = cv2.getTrackbarPos('ksize', 'Gaussian_Blur')
    image = cv2.imread(imagePath)
    blur = cv2.GaussianBlur(image, (2*ksize+1, 2*ksize+1), 0)
    cv2.imshow('Gaussian_Blur', blur)
```

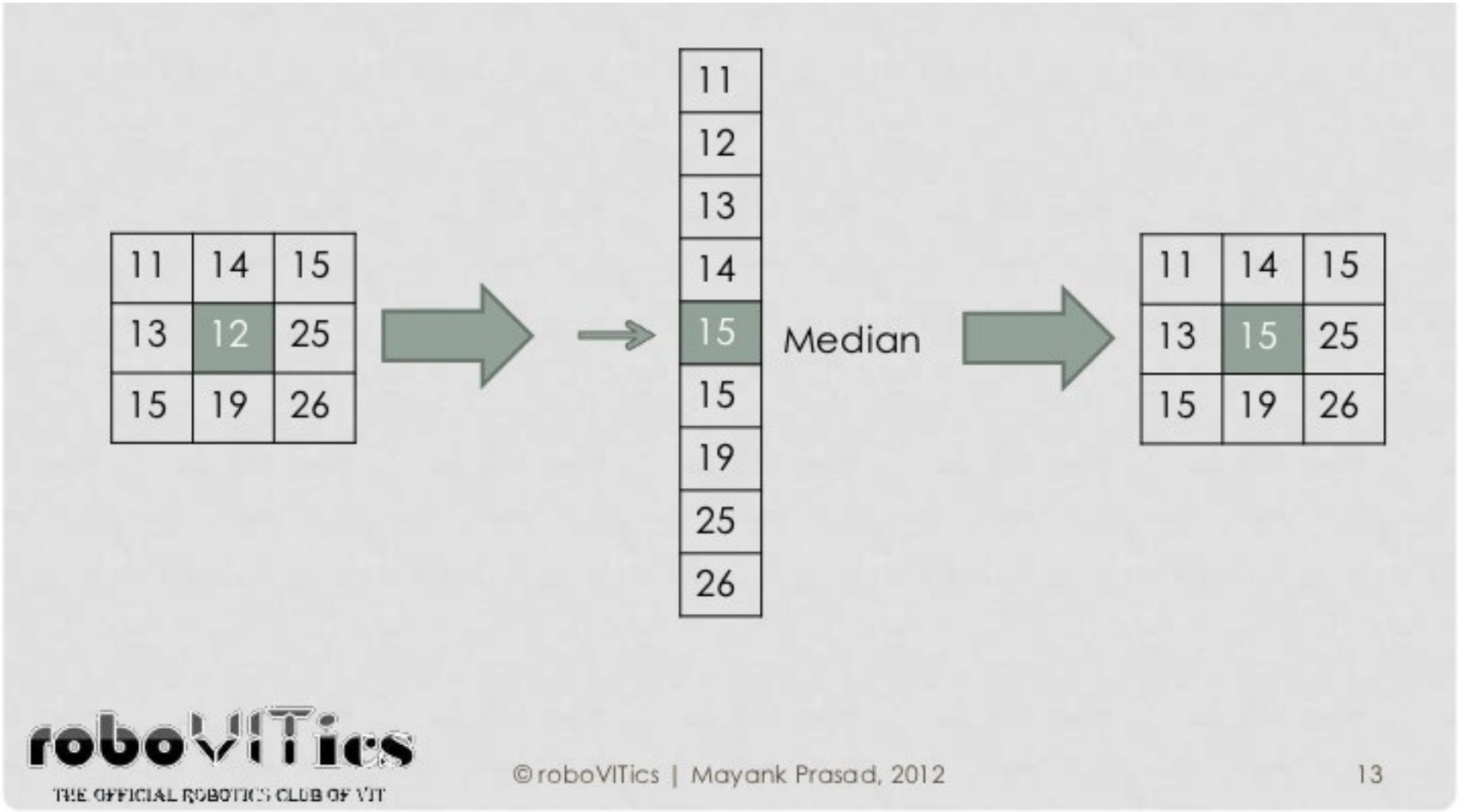
kernel size

DEMO
gaussian_blur.py

```
$ cd ~/pi-follower-car/04-opencv
$ python gaussian_blur.py
```

中值濾波 (MedianBlur)

- 取中位數的值作為目標像素

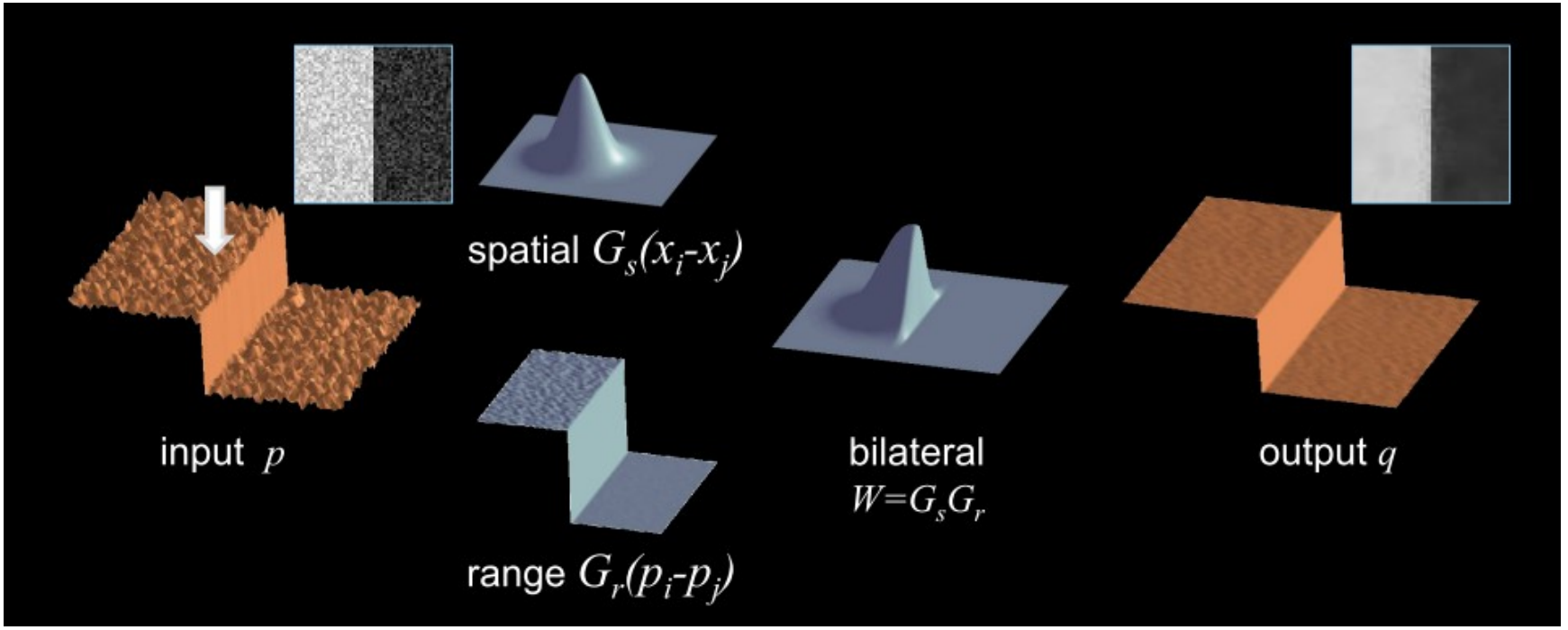


https://www.slideshare.net/roboVITics/robocv-module-4-image-processing-techniques-using-opencv

雙邊濾波 (BilateralFilter)

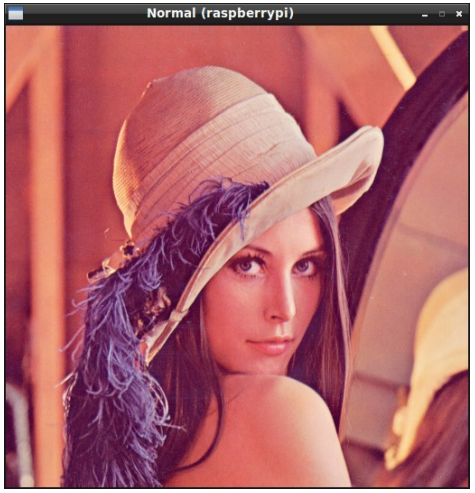
- 根據幾何空間距離和像素色差決定濾波係數
- 特色在具有平滑效果 + 保留邊緣

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|)$$



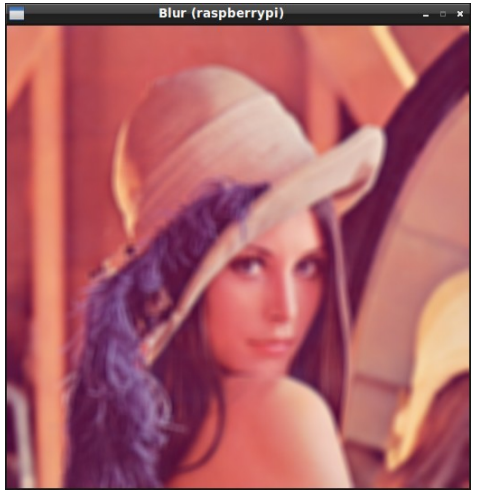
https://www.phototalks.idv.tw/academic/?p=581

不同濾波器有不同的平滑效果

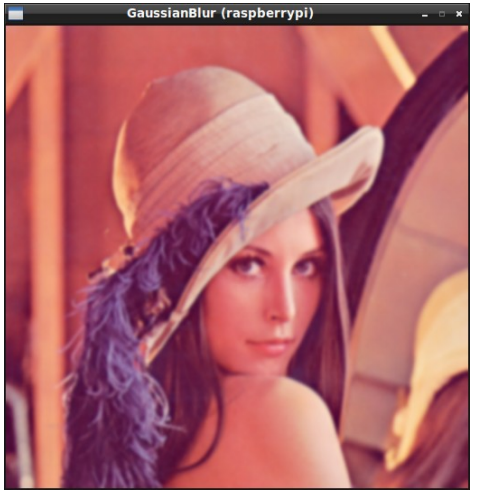


Normal

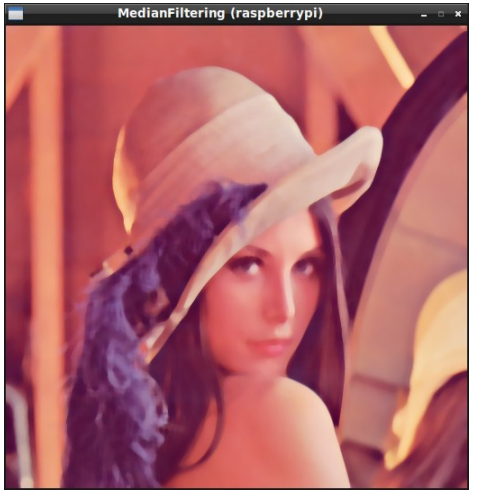
- 平均平滑最簡單，速度快
- 高斯平滑模糊化效果比平均平滑明顯且自然
- 中值濾波適合除噪（椒鹽噪聲）
- 雙邊濾波有模糊效果，但又能保留邊緣



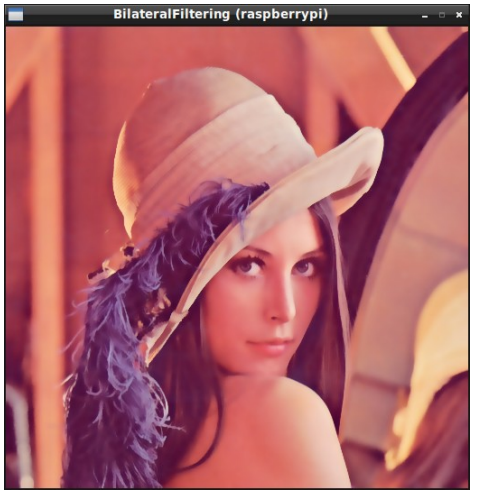
Box Blur



Gaussian Blur



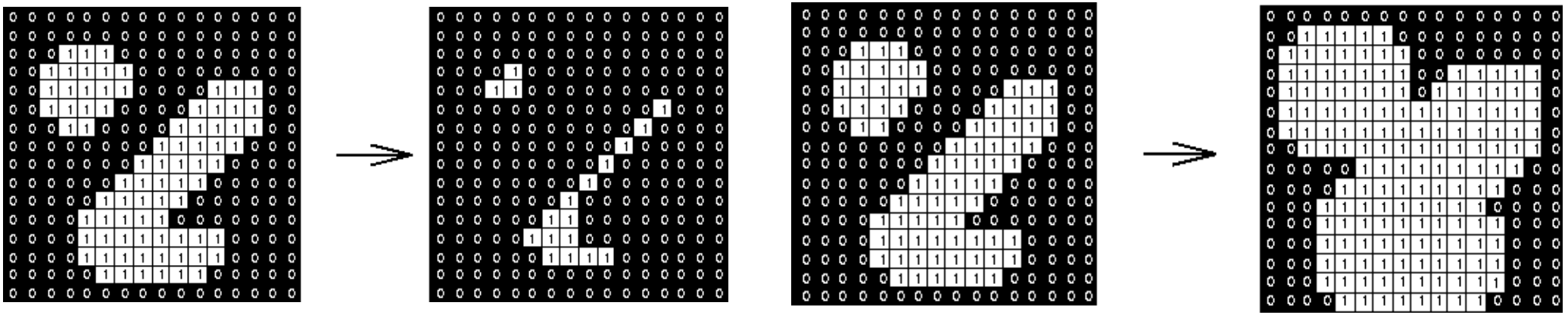
Median Blur



Bilateral Blur

從形態學的觀點

- 侵蝕 (Erode)
  - 消融物體的邊界
- 膨脹 (Dilate)
  - 擴大物體的邊界

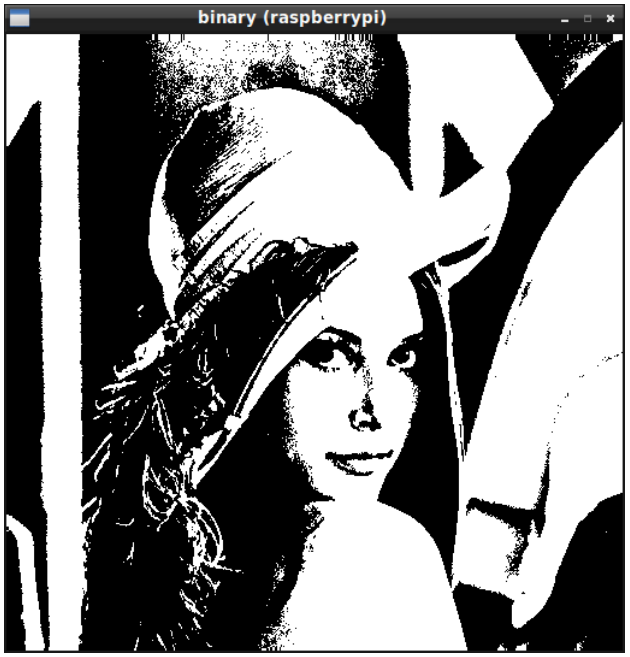


通常以奇數矩形為結構元素 (3x3, 5x5, 7x7...)，預設為 3x3

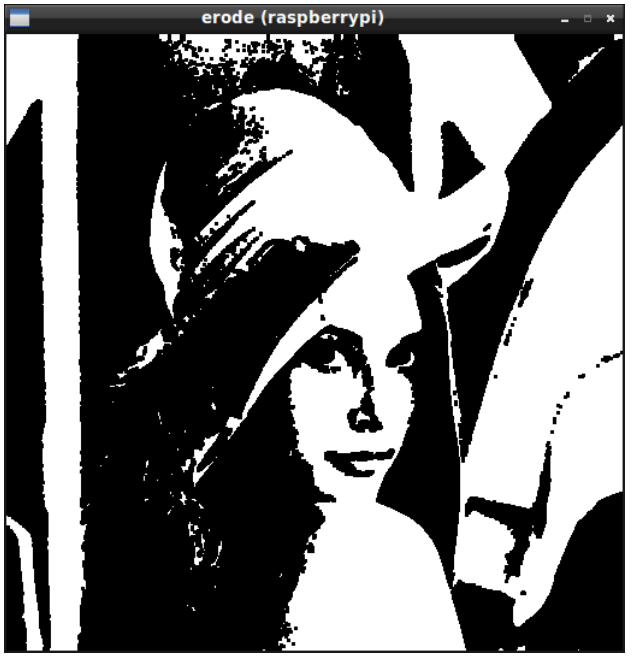
https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

侵蝕 (Erode)

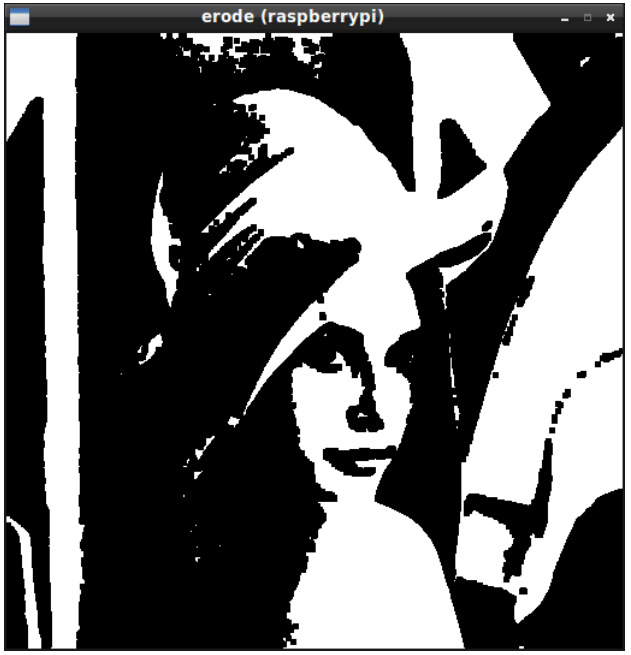
- `cv2.erode(src, kernel[, iterations])`
  - `iterations - number of times erosion is applied.`



黒白



iteration=1

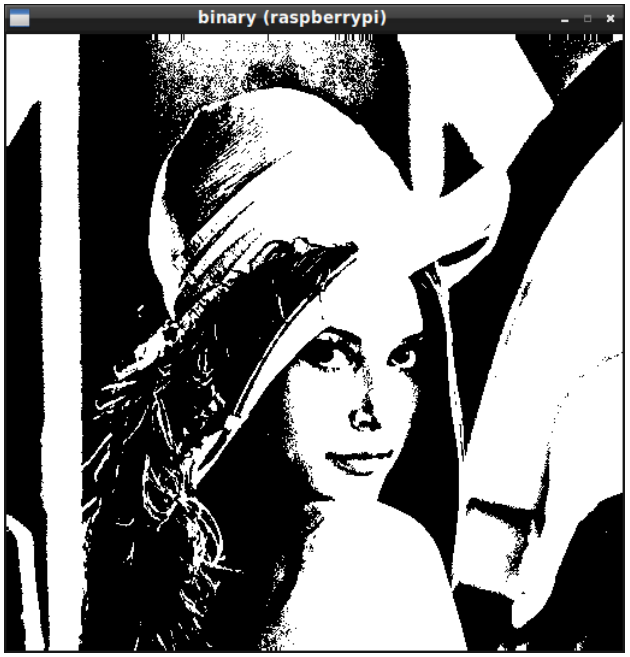


iteration=2

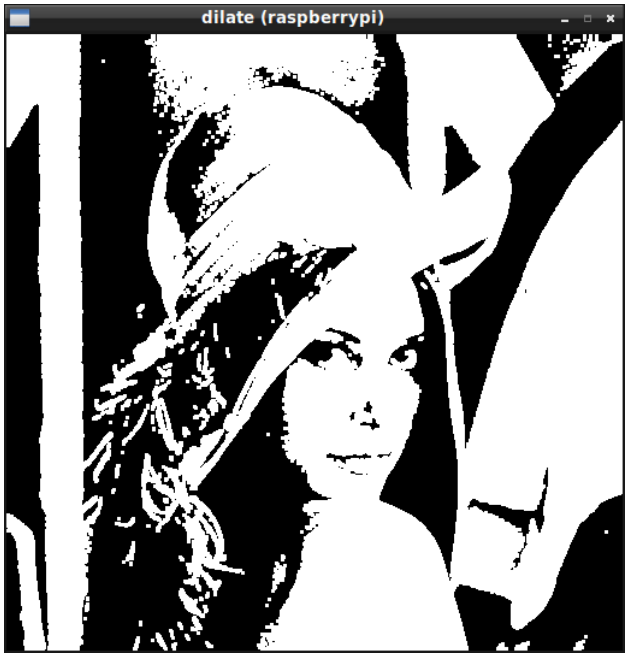
http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html

膨脹 (Dilate)

- `cv2.dilate(src, kernel[, iterations])`
  - `iterations - number of times dilation is applied.`



黑白



iteration=1

iteration=2

http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html

侵蝕效果

```
image = cv2.imread("lena256rgb.png")
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(_, binary) = cv2.threshold(gray, 127, 255,
cv2.THRESH_BINARY)

kernel = np.ones((3,3),np.uint8)
erode  = cv2.erode(binary, kernel, iterations=1)
cv2.imshow("erode", erode)

dilate = cv2.dilate(binary, kernel, iterations=1)
cv2.imshow("dilate", dilate)
```

DEMO
erode_dilate.py

```
$ cd ~/pi-follower-car/04-opencv
$ python erode_dilate.py
```

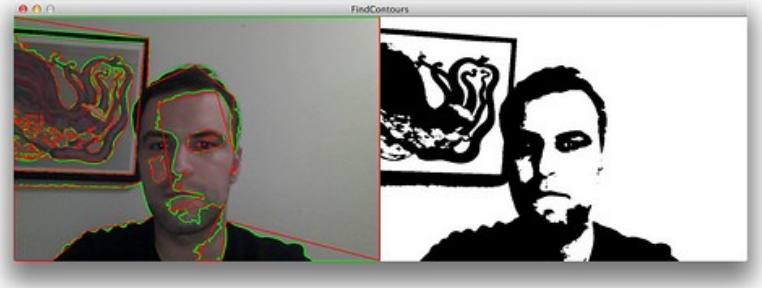
實驗 4-3：輪廓與中心點

目的：找尋物體中心

輪廓與邊緣

- 邊緣（edge）是亮度明顯改變的點（灰階劇烈變化）
- 輪廓為（contour）連接具有相同顏色或強度點的曲線
- 輪廓可能是邊緣的一部分
- 輪廓具是一群向量組，具有層次結構

邊緣

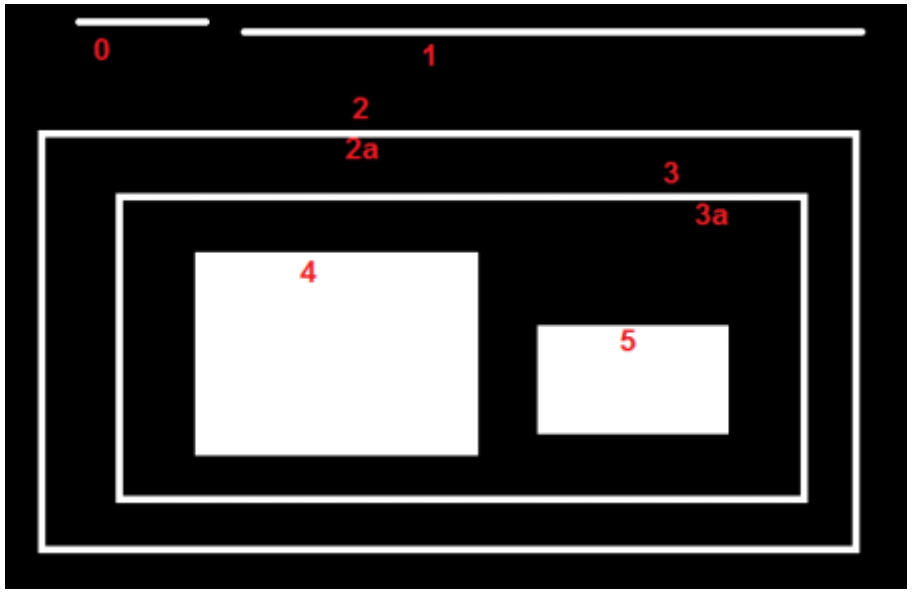


輪廓

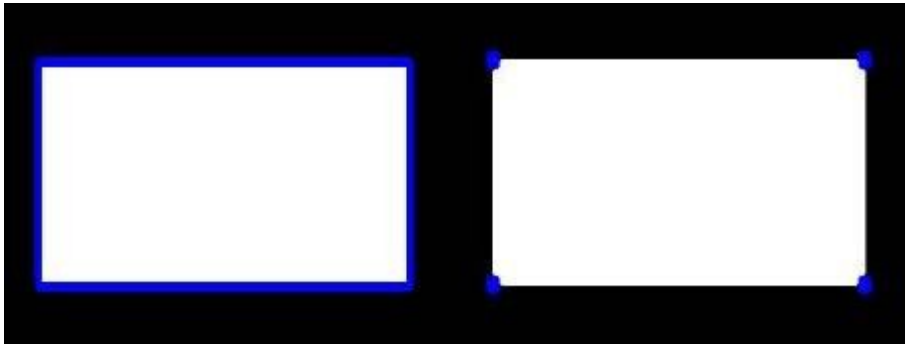
https://github.com/balrampandey19/opencv-webcam-processing

尋找輪廓 (findContours)

- `cv2.findContours(image, mode, method)`
  - `mode(` 輪廓檢索模式 `)`
  - `method(` 輪廓近似方法 `)`



輪廓檢索模式



輪廓近似方法

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html

參數說明

- mode(輪廓檢索模式)
  - CV_RETR_EXTERNAL: 只取最外層的輪廓
  - CV_RETR_LIST: 取得所有輪廓，不建立階層 (hierarchy)
  - CV_RETR_CCOMP: 取得所有輪廓，但只儲存成兩層的階層
  - CV_RETR_TREE: 取得所有輪廓，以全階層的方式儲存

- method(輪廓近似方法)
  - CV_CHAIN_APPROX_NONE: 儲存所有輪廓點
  - CV_CHAIN_APPROX_SIMPLE: 只留下頭尾點或對角頂點

http://monkeycoding.com/?p=615
http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html

畫輪廓 (drawContours)

- `cv2.drawContours(image,contours,contourIdx,color)`
  - `contourIdx = -1` 表示畫出所有輪廓點
  - 先轉為二值化影像能降低雜訊

二值化後找到的輪廓點

在原圖畫上輪廓點

http://docs.opencv.org/3.0-beta/modules/imgproc/doc/drawing_functions.html

找出輪廓並且全部畫出

```
image = cv2.imread("lena256rgb.jpg")


gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(_, binary) = cv2.threshold(gray, 127, 255,
cv2.THRESH_BINARY)


(contours, _) = cv2.findContours(binary,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(image, contours, -1, (0,255,0), 3)
cv2.imshow("Contours", image)
```

影像出現後，按任意鍵會顯示下個影像

DEMO
draw_contour.py

```
$ cd ~/pi-follower-car/04-opencv
$ python draw_contour.py lena256rgb.jpg
```

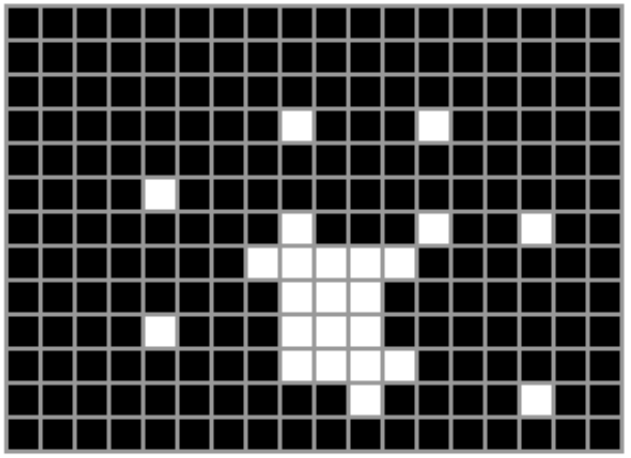
找中心點

- 從矩(moment)的觀點來看
  - 物理學：表示作用力促使物體繞支點旋轉的趨向
  - 數學：用來描述資料分佈特徵

- 以二值化（黑白）的圖形找中心點

http://docs.opencv.org/3.1.0/d8/d23/classcv_1_1Moments.html

常見矩



- 空間矩 (spatial moments)

$$\mathbf{m}_{ji} = \sum_{x,y} \left(\mathbf{array}(x,y) \cdot x^j \cdot y^i \right)$$

- 中心矩 (central moments)

$$\mathbf{mu}_{ji} = \sum_{x,y} \left(\mathbf{array}(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i \right)$$

$$\bar{x} = \frac{\mathbf{m}_{10}}{\mathbf{m}_{00}},\ \bar{y} = \frac{\mathbf{m}_{01}}{\mathbf{m}_{00}}$$

中心點 (質心 / 圖心)

http://aishack.in/tutorials/image-moments/

找中心點與外框

```
image = cv2.imread("moment.jpg")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(_, binary) = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
(contours, _) = cv2.findContours(binary, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

cnt = contours[0]

((x, y), radius) = cv2.minEnclosingCircle(cnt)
M = cv2.moments(cnt)

center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
cv2.circle(image, (int(x), int(y)), int(radius), (0, 255, 255), 2)
cv2.circle(image, center, 5, (0, 0, 255), -1)
```

取得輪廓點

第一組輪廓點

計算外框半徑

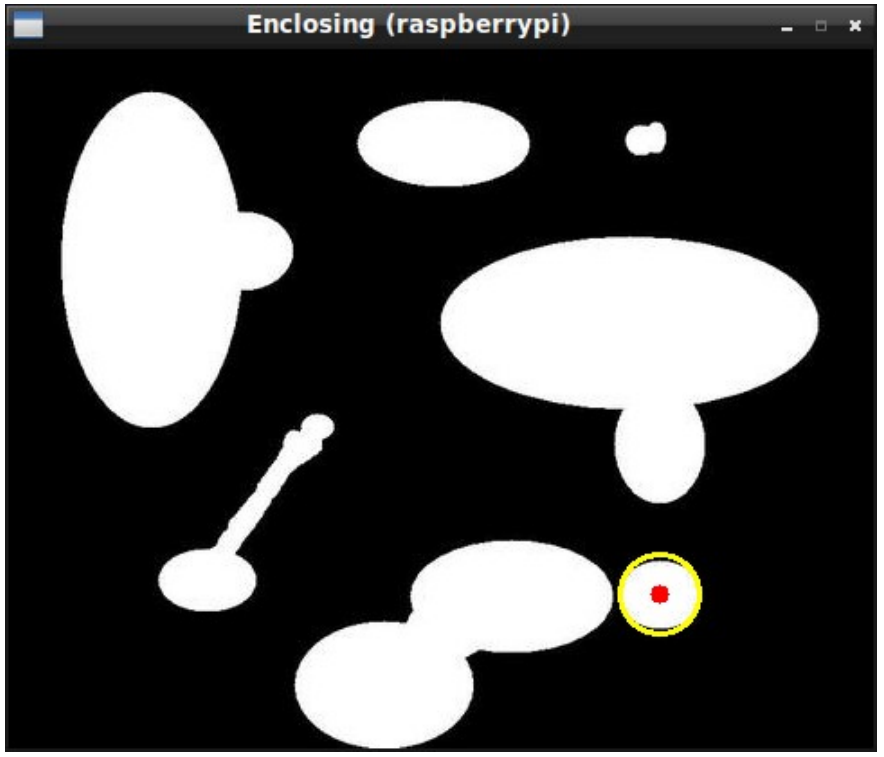
第一組輪廓點的矩

中心點

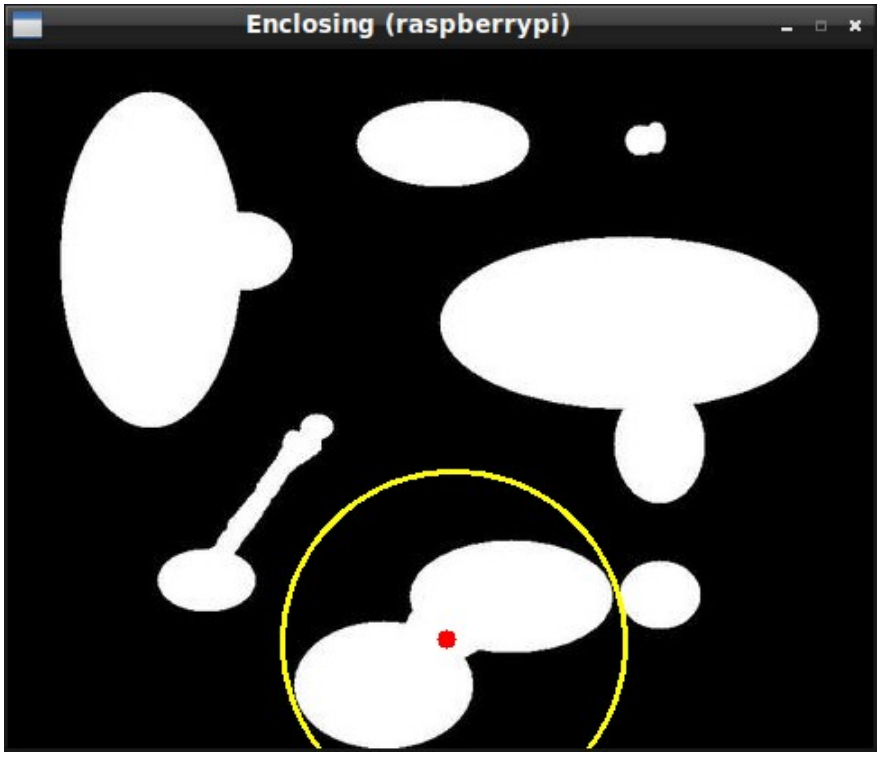
DEMO
draw_moment.py

```
$ cd ~/pi-follower-car/04-opencv
$ python draw_moment.py moment.jpg
```

每組輪廓都可以找到找中心點



第一組輪廓點
cnt = contours[0]



第二組輪廓點
cnt = contours[1]

實驗 5-1：色彩追蹤

目的：找出 HSV 與即時色彩追蹤

追蹤流程

- 定義要追蹤的顏色（例如綠色）
- 將影像轉到 HSV 空間後二值化
  - 綠色參考數值
    - `Color_Lower = (36, 130,46)`
    - `Color_Upper = (113, 255, 255)`
- 找出該顏色的輪廓
- 找出質心與半徑
- 畫出質心與半徑

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

DEMO
color_tracking.py

調整好 HSV 數值後 , 按 'q' 離開就會儲存

```
$ cd ~/pi-follower-car/05-car
$ python color_tracking.py
```

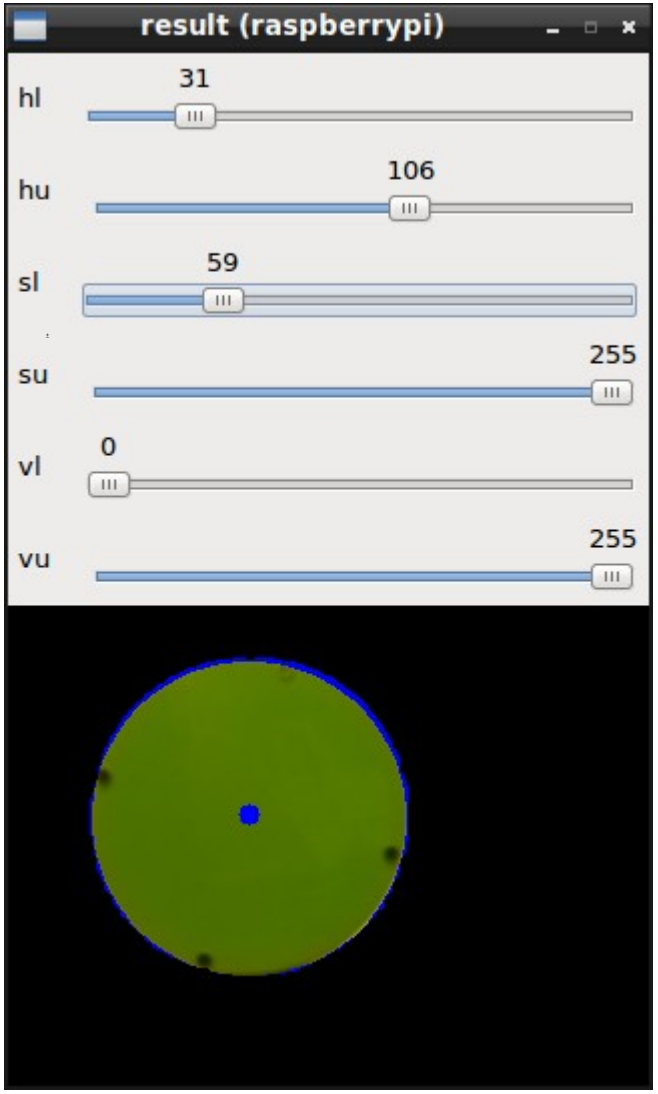
實驗 5-2：寵物小車

目的：根據顏色控制小車移動

根據移動規則控制小車移動

- 找出顏色質心,定義規則(超過中心線 ±10 就動作)

```
if radius < 100:
    motor.forward()

if center[0] > width/2 + 10:
    motor.turnRight()
elif center[0] < width/2 - 10:
    motor.turnLeft()
```



DEMO
follower_car.py

```
$ cd ~/pi-follower-car/05-car
$ python follower_car.py
```

優化 (硬體 , 軟體 , 網路)

- 小車直線走得不直
- 小車轉彎和直線速度不一樣
- 小車轉彎角度太大，導致相機會看不到
- 顯示結果或處理速度太慢
- 顏色判斷不準，容易受干擾
- 移動規則不好

優化 (硬體 , 軟體 , 網路)

- 小車直線走得不直 (硬體 + 軟體)

  => 更換車體 & 移動需要先做校正

- 小車轉彎和直線速度不一樣 (軟體)

  => 轉彎改為原地轉

- 小車轉彎角度太大，導致相機會看不到 (硬體 + 軟體)

  => 用 PWM 控制轉速，或是改用魚眼 (廣角) 相機

- 顯示結果或處理速度太慢 (軟體 + 網路)

  => 超頻 & 使用原生 OpenCV & 關掉顯示視窗

- 顏色判斷不準，容易受干擾 (硬體)

  => 使用封閉空間，背景單純，只用日光燈源

- 移動規則不好 (軟體)

  => 需根據小車與目標的相對位置計算移動速度與角度

顯示結果或處理速度太慢

- 關掉顯示視窗(follower_car.py)

```
Display_Frame = True
...
...
if Display_Frame == True:
    cv2.imshow("Frame", frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break
```

改成 False 就會很順了

小車轉彎角度太大，導致相機會看不到

- PWM 控制轉速 (pwm_motor.py)

```
dc = 85
pwm_r1 = GPIO.PWM(Motor_R1_Pin, 600)
pwm_r2 = GPIO.PWM(Motor_R2_Pin, 600)
pwm_l1 = GPIO.PWM(Motor_L1_Pin, 600)
pwm_l2 = GPIO.PWM(Motor_L2_Pin, 600)

def turnLeft():
    pwm_r1.ChangeDutyCycle(dc)
    pwm_r2.ChangeDutyCycle(0)
    pwm_l1.ChangeDutyCycle(0)
    pwm_l2.ChangeDutyCycle(0)
    time.sleep(t)
```

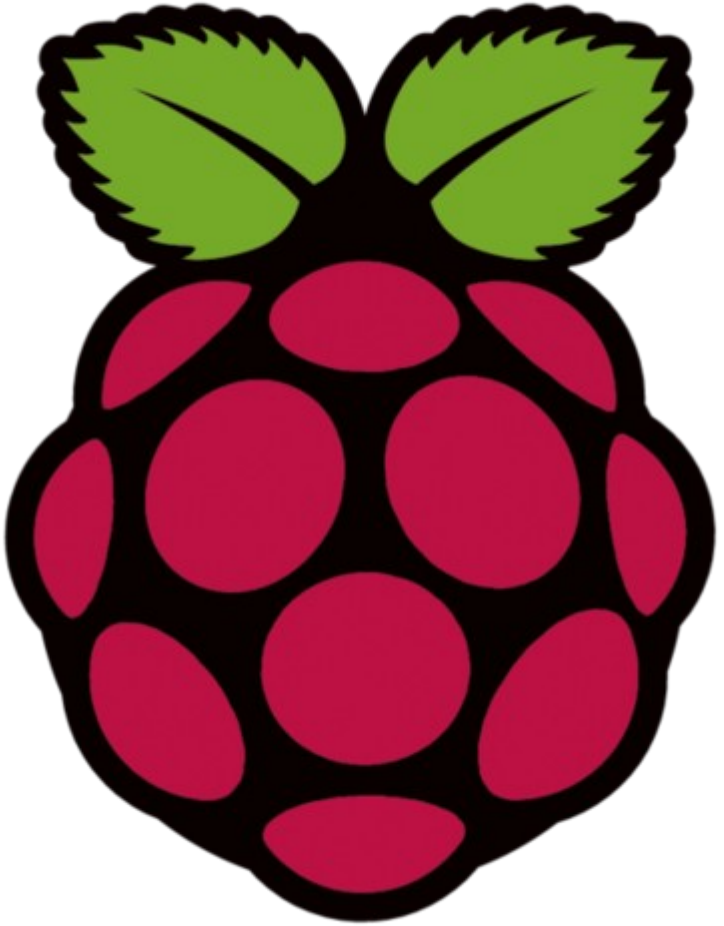
小車轉彎角度太大，導致相機會看不到

- 改用PWM(follower_car.py)

```
import cv2
import numpy as np
import ConfigParser
import pwm_motor as motor
#import dc_motor as motor
```

- 執行follower_car.py前要先刪除pwm_motor.pyc
- 刪除指令：$ rm pwm_motor.pyc

Raspberry Pi Rocks the World



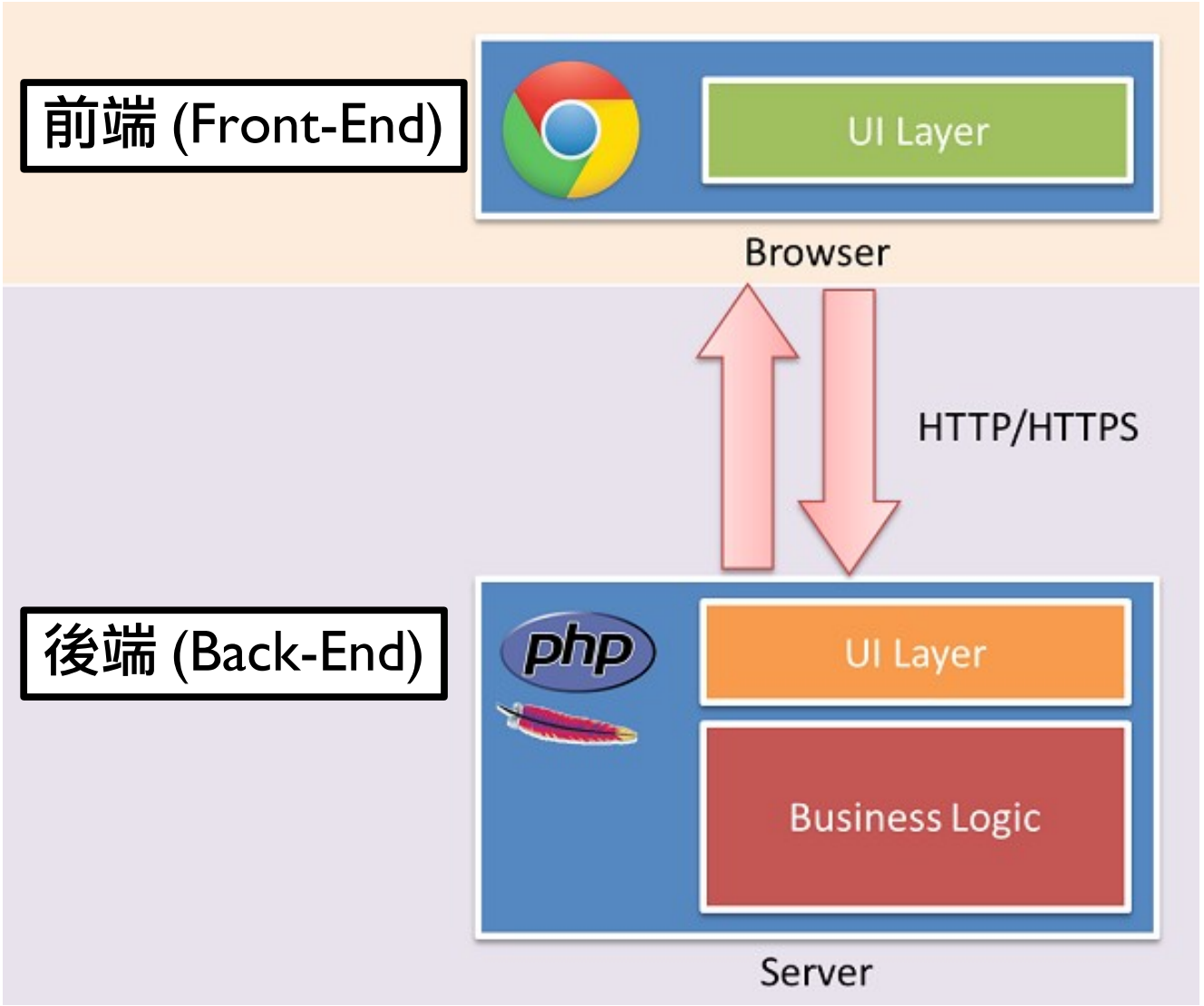
Thanks

補充資訊

實驗 2-5:Web Control

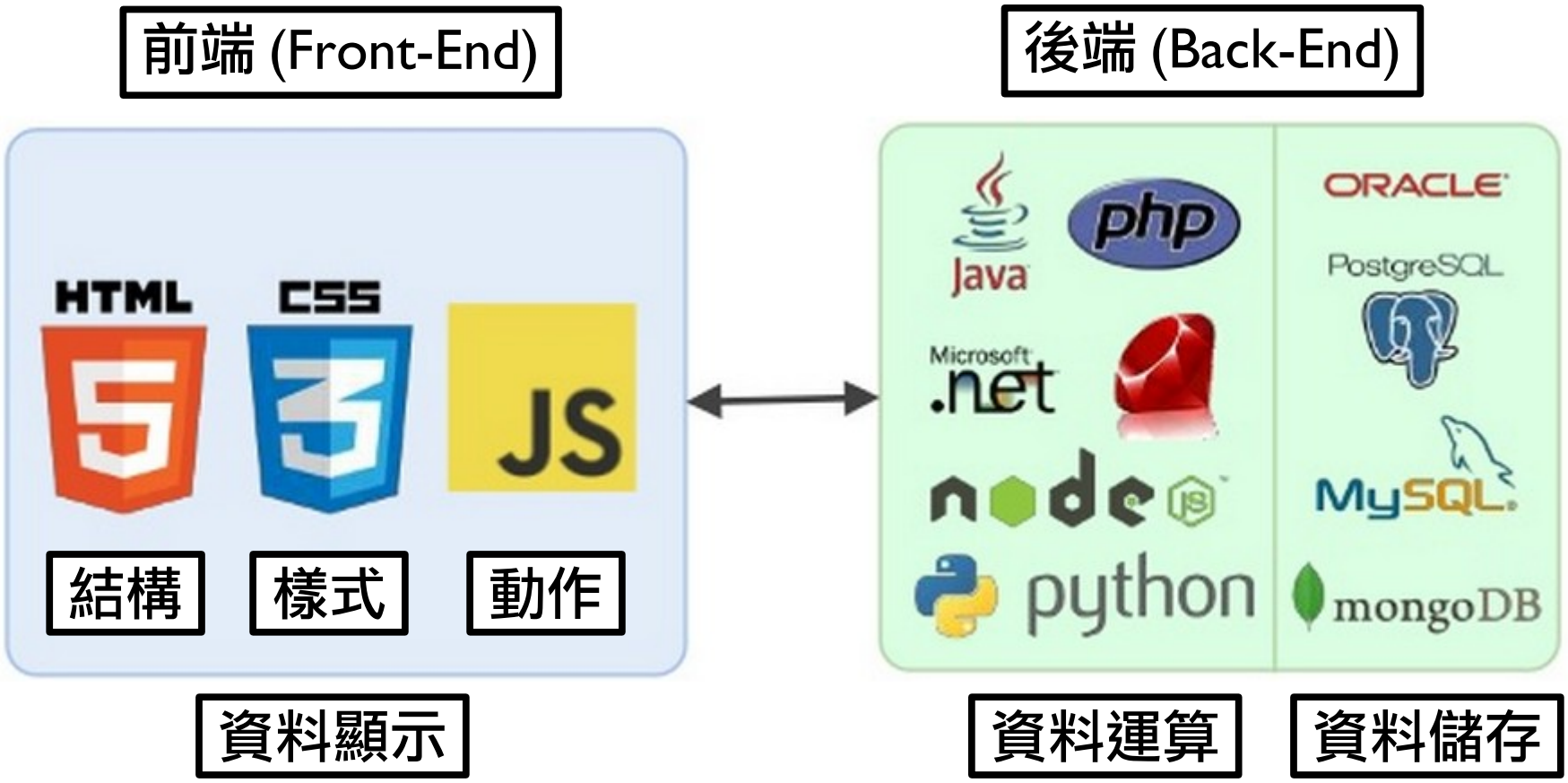
目的：學習如何從網頁控制硬體

網頁的運作原理



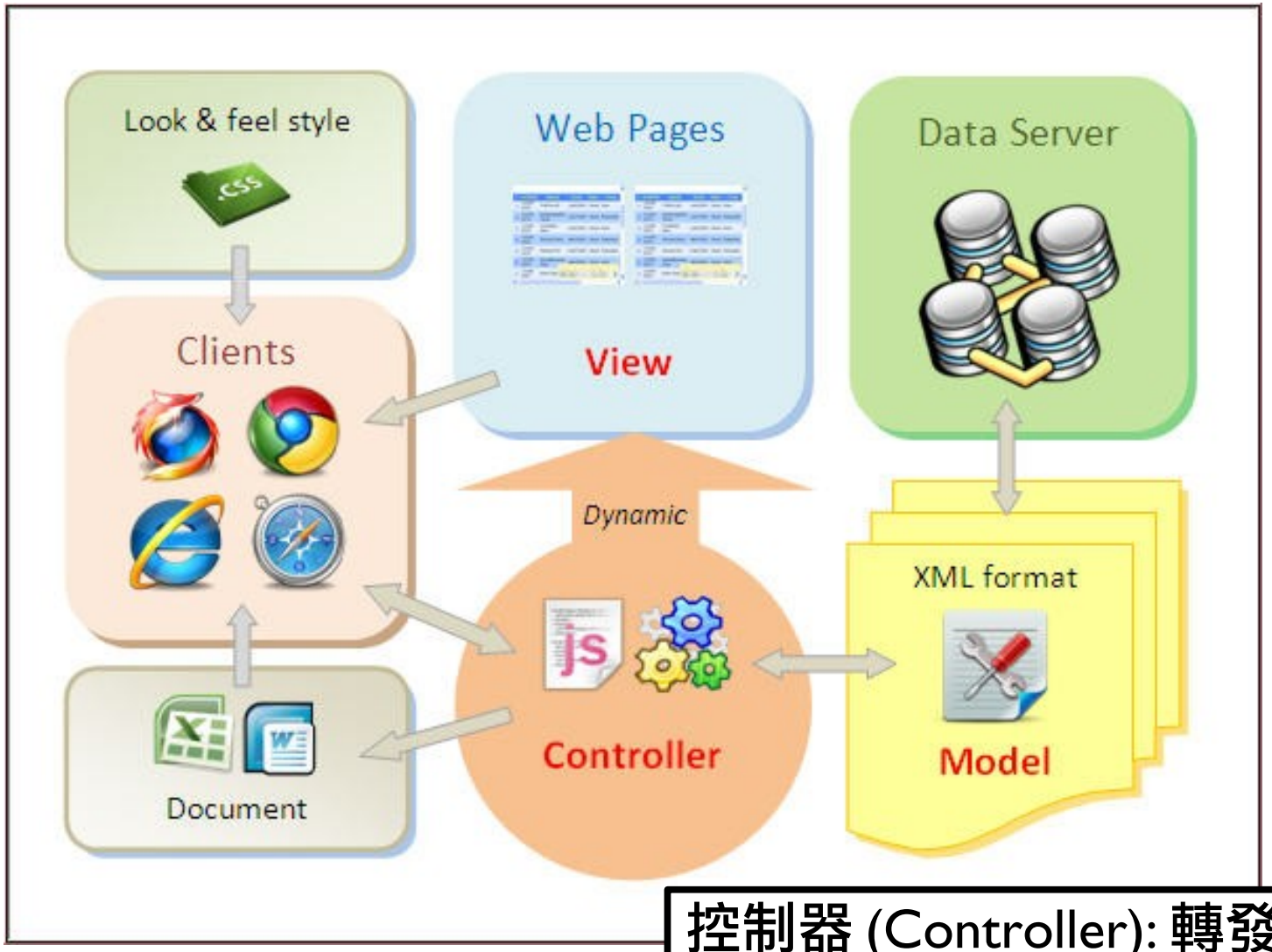
https://www.nczonline.net/blog/2013/10/07/node-js-and-the-new-web-front-end/

可拆解成不同的實做



http://goo.gl/yZj6mH

Model-View-Controller 設計模式

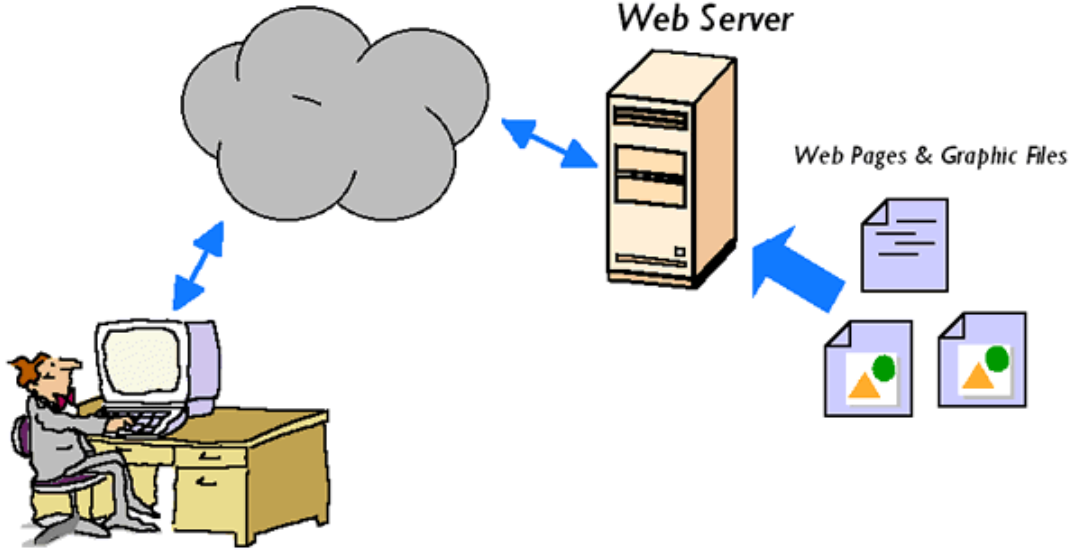


控制器 (Controller): 轉發請求
視圖 (View): 圖形界面顯示
模型 (Model): 實現邏輯與資料儲存

https://webformdeveloping.wordpress.com/

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



http://www.resultantsys.com/index.php/general/what-is-a-web-application-server/

Flask

web development,
one drop at a time

- A Python Microframework

app-hello.py

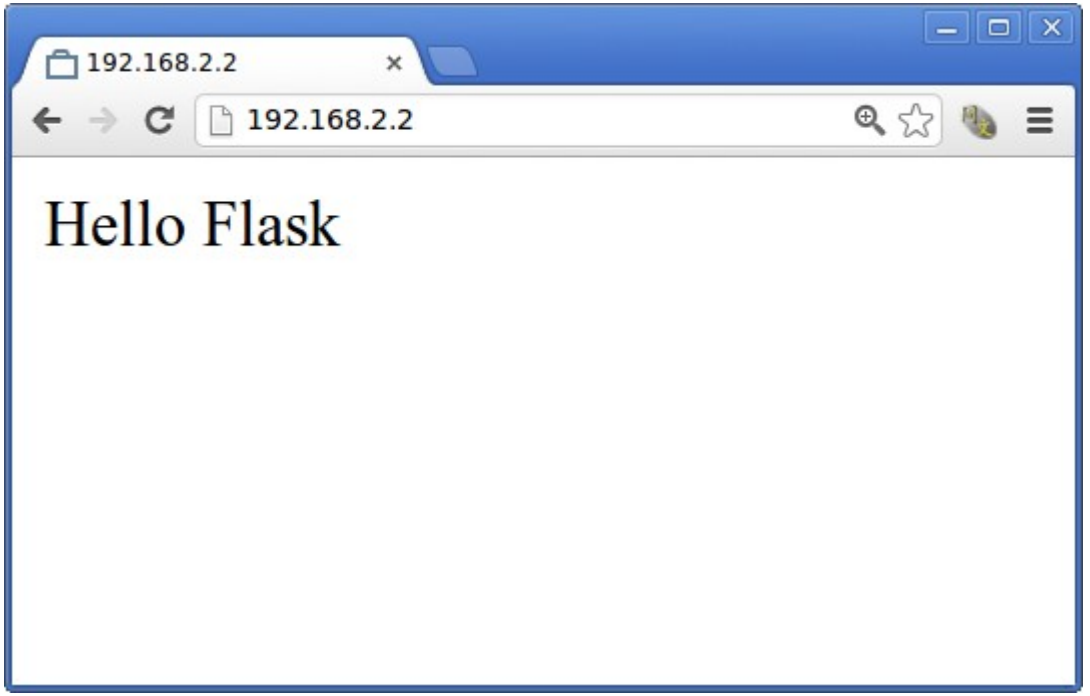
```
from flask import Flask
app = Flask(__name__)

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

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

Controller

View

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 ~/pi-follower-car/02-motor/02_5-http
$ sudo python app-hello.py
```

hello 使用到的檔案

- `app-hello.py`

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)
```

建立 template

```
$ mkdir templates
$ nano templates/link.html
```

```
<h1>Hello Template</h1>
<a href="{{ url_for('foo') }}">foo</a>
```

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

```
<ul>
{% for ext in extns %}
<li>{{ ext }}</li>
{% endfor %}
</ul>
```

執行

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

DEMO
app-route.py

```
$ cd ~/pi-follower-car/02-motor/02_5-http
$ sudo python app-route.py
```

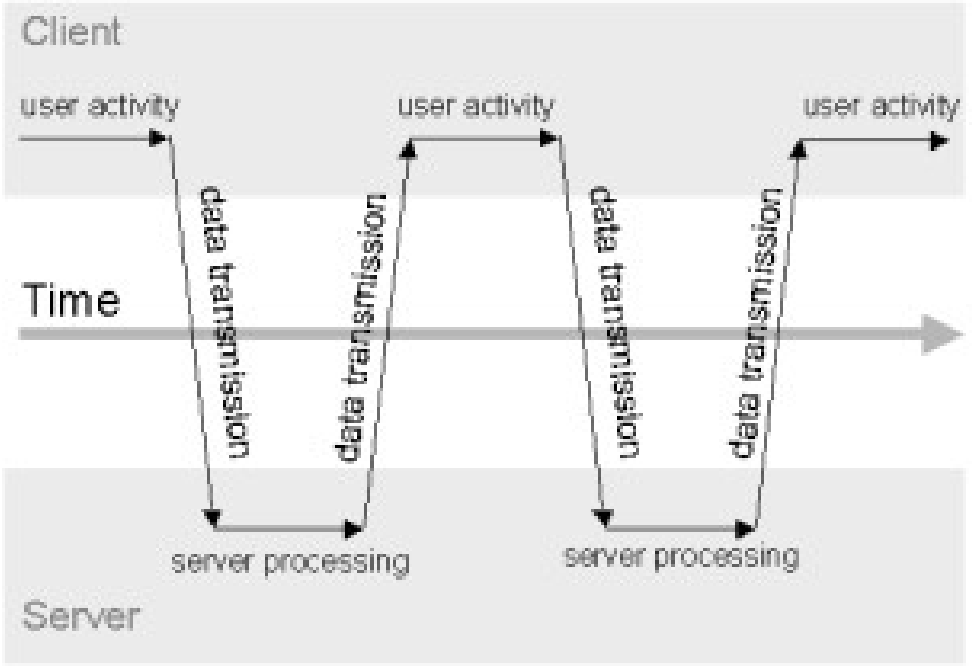
route(樣板) 使用到的檔案

- app-route.py
- templates

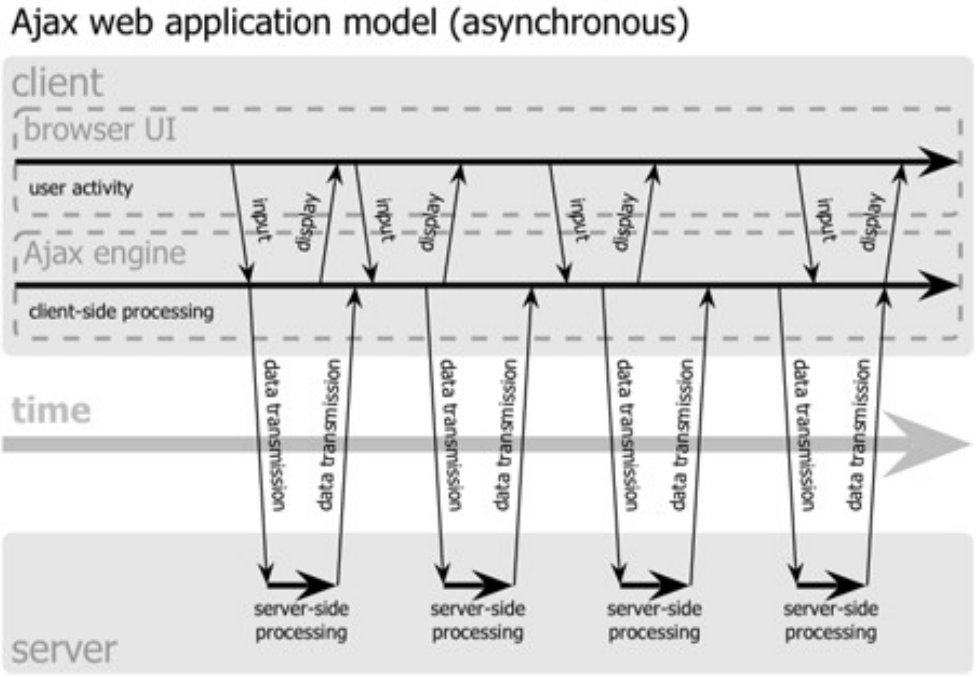
```
├── bar.html (/foo)
└── link.html (/)
```

AJAX

- Synchronous

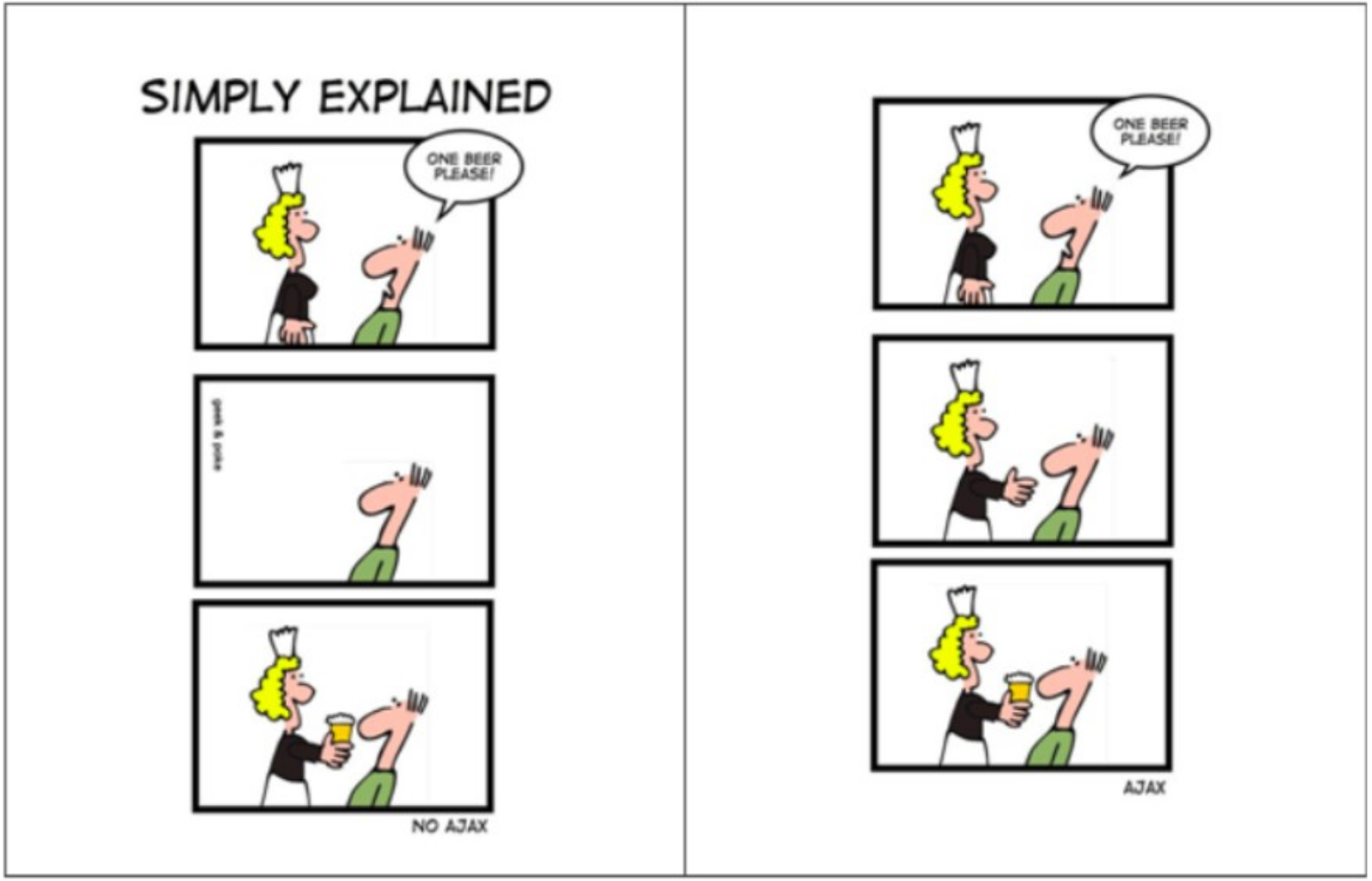


- Asynchronous



http://edndoc.esri.com/arcobjects/9.2/NET_Server_Doc/developer/ADF/ajax_callback.htm

概念是這樣的



http://slideplayer.com/slide/6422767/

AJAX 實做

- 目標：按下按鍵以後換圖，但不重新刷新頁面
- 準備項目：
  1. 回傳圖檔路徑 (python)
  2. 宣告網頁 DOM 元素 (html)
  3. JS 發送 request 後取回資源再更新畫面 (js)

app-ajax.py

```
from flask import Flask, request, render_template
import time

app = Flask(__name__)

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

@app.route('/<path:path>')
def static_file(path):
    return app.send_static_file(path)

@app.route('/ajax', methods=['GET'])
def ajax():
    return str(int(time.time()%3)+1) + ".jpg"

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

回傳圖片路徑

DOM 元素

```
<html>
<body>
  <h1>AJAX Demonstration</h1>
  <img id="img_ajax" src="{{ img }}" /> <br />
  <input type="button" value="click me"
onclick="send_ajax();" />
</body>
</html>
```

JavaScript 送 request

```
<script language="Javascript">
  function send_ajax() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        document.getElementById("img_ajax").src =
xhttp.responseText;
      }
    };
    xhttp.open("GET", "/ajax", true);
    xhttp.send();
  }
</script>
```

DEMO
app-ajax.py

```
$ cd ~/pi-follower-car/02-motor/02_5-http
$ sudo python app-ajax.py
```

網頁控制小車

- 目標：按下畫面的控制項以後驅動小車
- 準備項目：

  1. 根據傳進來的參數驅動小車(python)
  2. JS 發送 request 傳送參數(js)

根據傳進來的參數驅動小車

```
GPIO.setmode(GPIO.BOARD)
def stop():
def forward():
def backward():
def turnRight():
def turnLeft():
```

GPIO 控制小車前後左右

```
app = Flask(__name__)

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

@app.route('/ajax', methods=['POST'])
def ajax():
    arrow = request.form['arrow']

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

從網頁讀取傳進來的參數

DEMO
app-motor.py

```
$ cd ~/pi-follower-car/02-motor/02_5-http
$ sudo python app-motor.py
```

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

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

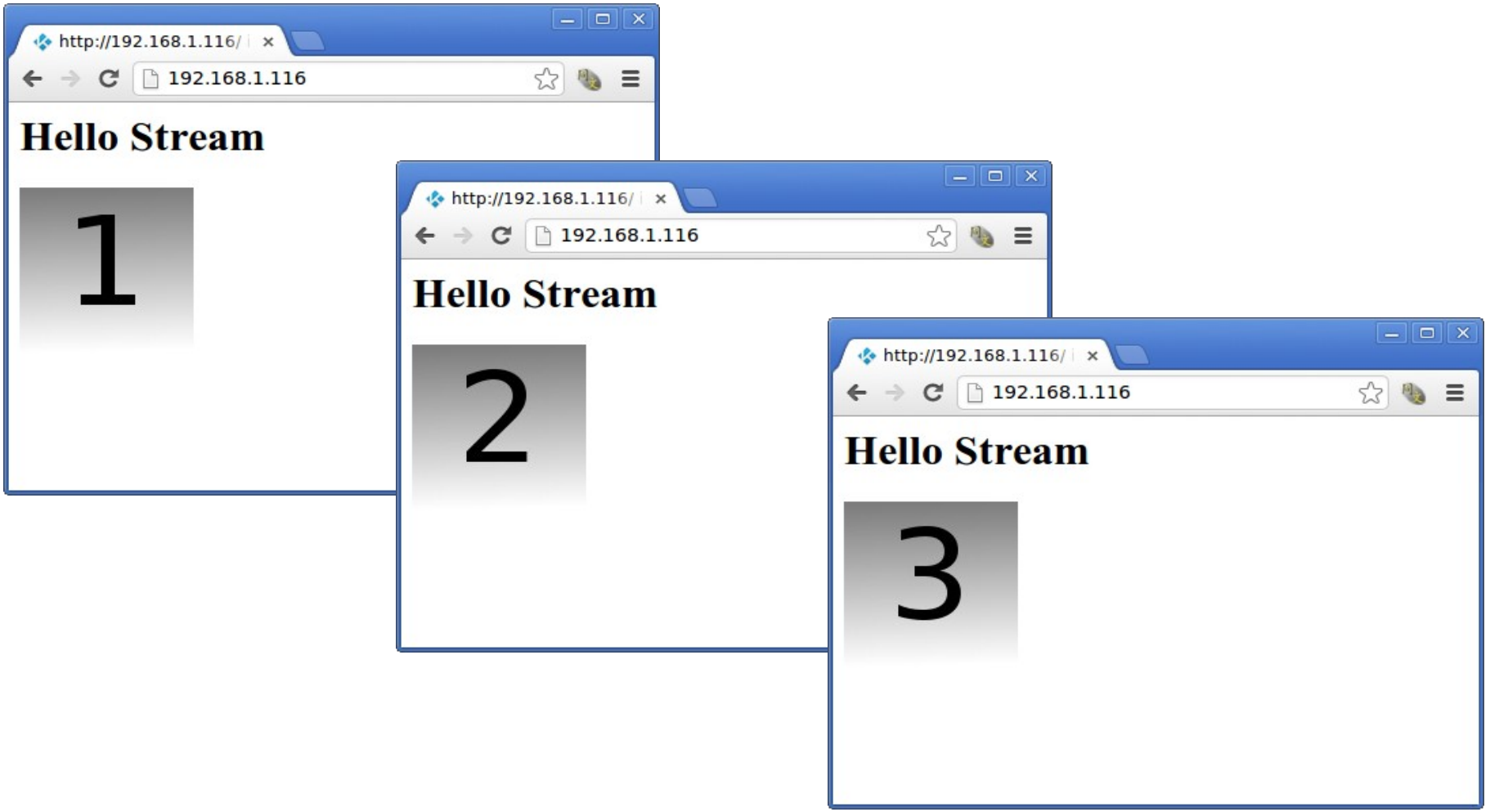
```
$ nano templates/stream.html
```

```
<h1>Hello Stream</h1>
<img id="bg" src="{{ url_for('video_feed') }}">
```

- 補個圖 1.jpg, 2.jpg, 3.jpg

執行

```
$ 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()
        ret, jpeg = cv2.imencode('.jpg', image)
        return jpeg.tostring()
```

開啟 /dev/videoX

V4L2 API

網頁控制小車 + 視訊串流

DEMO

app-car.py

```
$ cd ~/pi-follower-car/02-motor/02_5-http
$ sudo python app-car.py
```