

# 快速測試 Donkey Car

台灣樹莓派 <sosorry@raspberrypi.com.tw>  
2020/05/01 @VNU

# CC (Creative Commons)

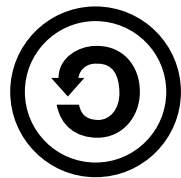
## 姓名標示 — 非商業性 — 相同方式分享



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



非商業性 — 你不得將本素材進行商業目的之使用。



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



# 關於我們

- Raspberry Pi 官方經銷商



# about 台灣樹莓派

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



# 分享 x 教學

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





# 學習路徑



Pi選購指南



Pi設定安裝



Linux系統管理



Python程式設計

## I/O硬體控制

GPIO學習套件

初

感測器學習套件  
(基礎/進階)

中

空氣盒子套件  
(PiM25)

初

Win10開發套件

初

智慧開關套件

初

Linux Driver  
學習套件

進

## 無線/IoT

RFID/NFC  
門禁系統

初

LoRa IoT  
閘道器套件

初

生理資訊  
監控IoT(藍牙)

初

毫米波人流/熱點監控  
(mmWave)

初

## 相機/影像處理

特色相機改裝套件

初

寵物小車套件

初

自控機器手臂套件

中

小鴨車套件  
(Duckietown)

進

## 人工智慧

驢車套件  
(DonkeyCar)

初

AIY Vision Kit

中

Intel神經運算棒

中

Google Coral  
USB加速器

中

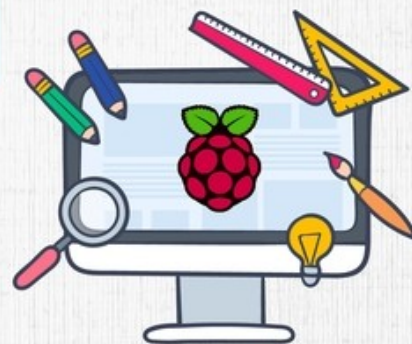
## 語音/訊號處理

智慧音箱套件

初

AIY Voice Kit

初



初 初階課程

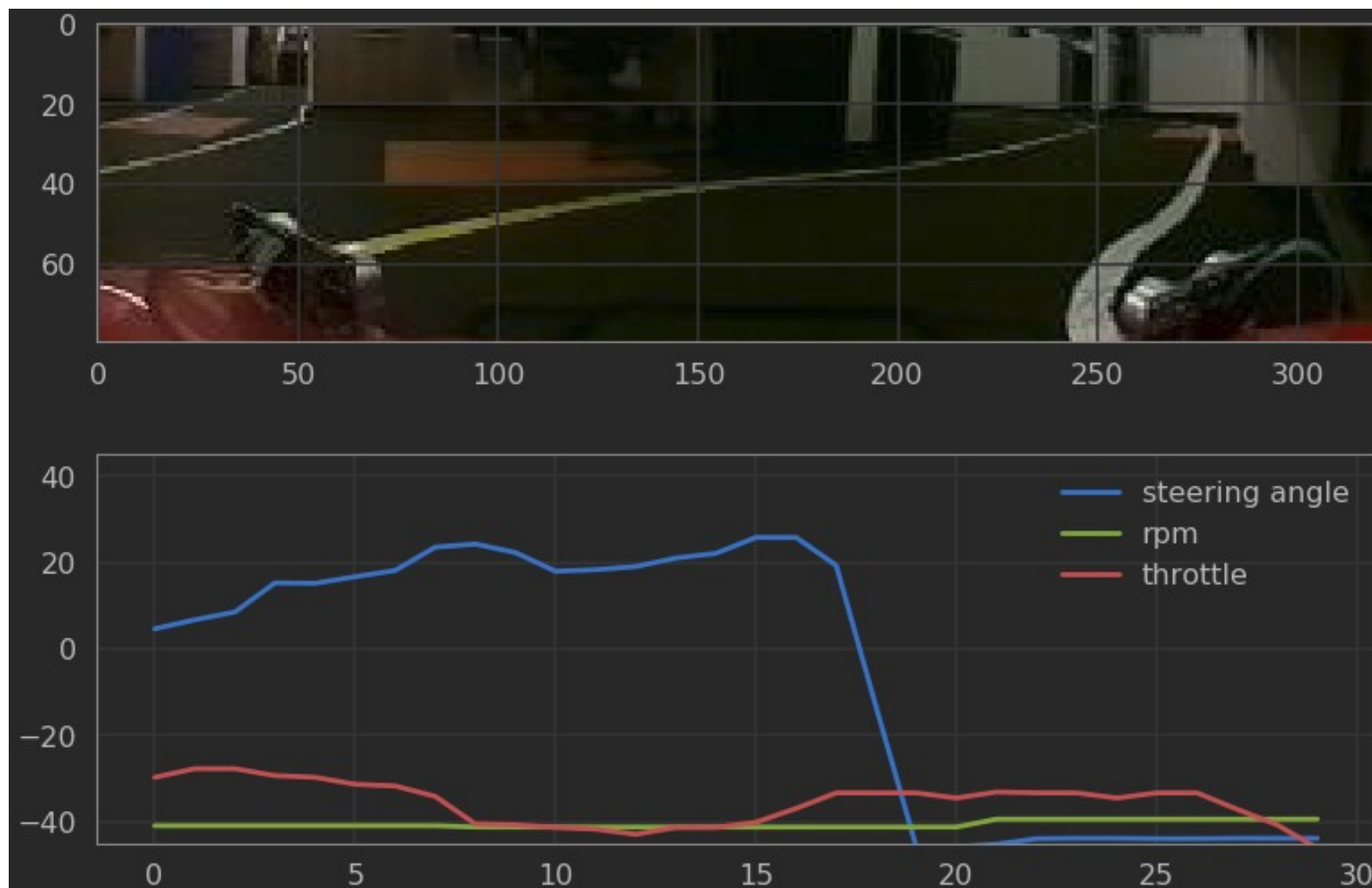
中 中階課程

進 進階課程

# 大綱

- 實驗 1：建立 donkeycar 專案
- 實驗 2：相機測試
- 實驗 3：輪子校正
- 實驗 4：測試搖桿

# 目標：根據影像預測油門和轉向



<https://www.donkeycar.com/updates>



# 安裝必要軟體 ( 已安裝 /hackmd)

- `$ sudo apt-get update`
- `$ sudo apt-get install -y build-essential joystick python3-dev python3-pip python3-virtualenv python3-numpy python3-picamera python3-pandas python3-rpi.gpio git i2c-tools avahi-utils gfortran libopenjp2-7-dev libtiff5-dev libatlas-base-dev libopenblas-dev libhdf5-serial-dev`

# ~/donkeycar 專案架構

- ~/donkeycar /
  - ├─ Dockerfile
  - ├─ donkeycar /
  - ├─ donkeycar.egg-info
  - ├─ install
  - ├─ Makefile
  - ├─ mkdocs.yml
  - ├─ setup.cfg
  - └─ setup.py

- ~/donkeycar/**donkeycar**

- ├─ management
  - ├─ **base.py**
  - ├─ joystick\_creator.py
  - ├─ makemovie.py
  - ├─ tub.py
  - └─ tub\_web
- ├─ parts
  - ├─ **actuator.py**
  - ├─ camera.py
  - ├─ controller.py
  - ├─ cv.py
  - ├─ datastore.py
  - ├─ image.py
  - ├─ imu.py
  - ├─ keras.py
  - ├─ lidar.py
  - ├─ ros.py
  - ├─ simulation.py
  - ├─ transform.py
  - └─ web\_controller/
- ├─ **templates**
- └─ tests

## parts 目錄



馬達控制程式

# ~/mycar 專案架構

- ~/mycar 由 base.py 根據 templates 建立出來
- ~/mycar

└─ config.py

└─ data/

儲存控制和影像路徑

└─ logs/

└─ manage.py

程式進入點

└─ models/

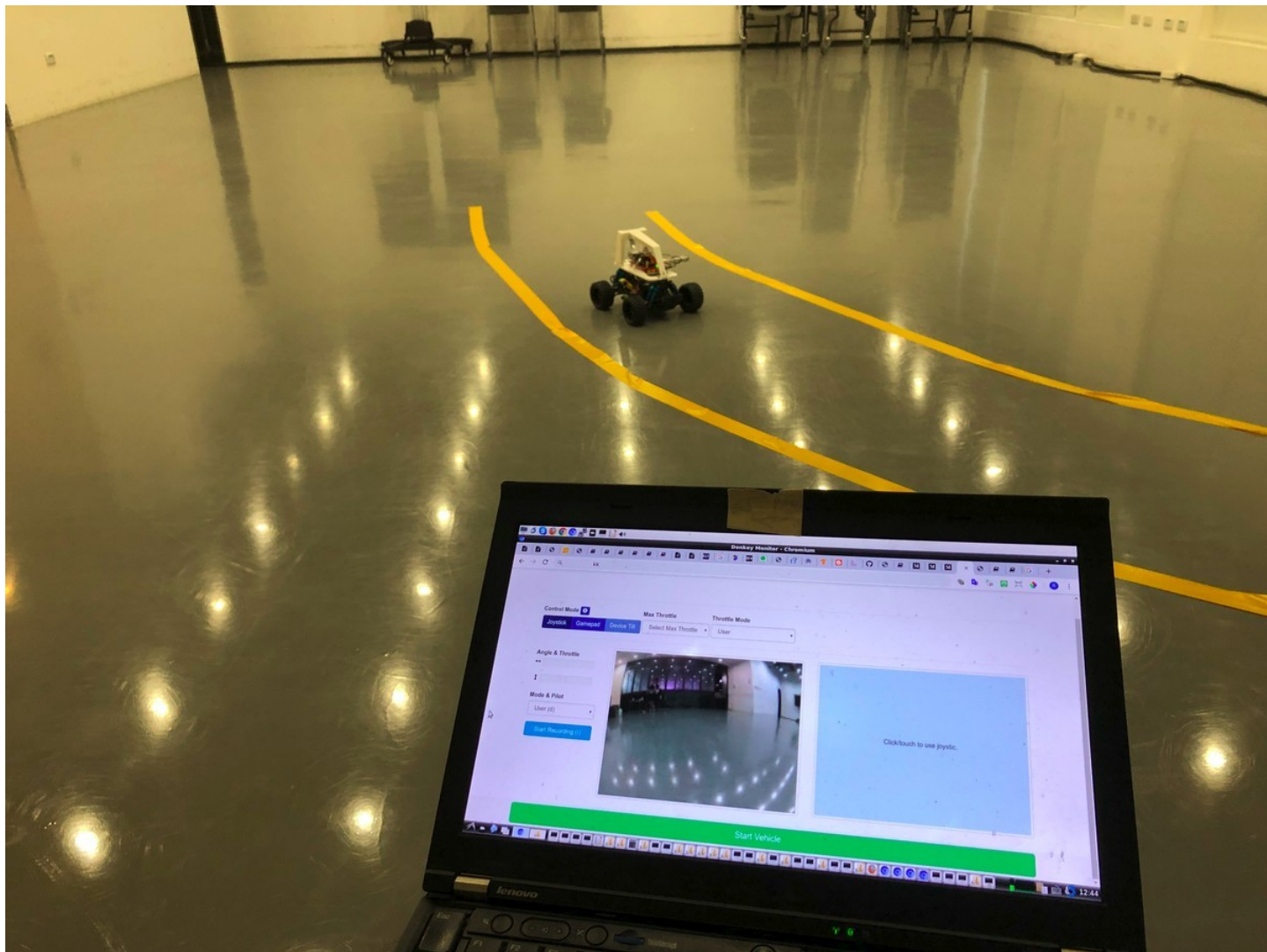
儲存模型路徑

└─ myconfig.py

└─ train.py

# 讓你的 Donkey Car 跑起來

- 使用瀏覽器或是搖桿控制





# 實驗 1：建立 donkeycar 專案

目的：初始化環境

# 安裝 donkeycar( 已安裝 /hackmd)

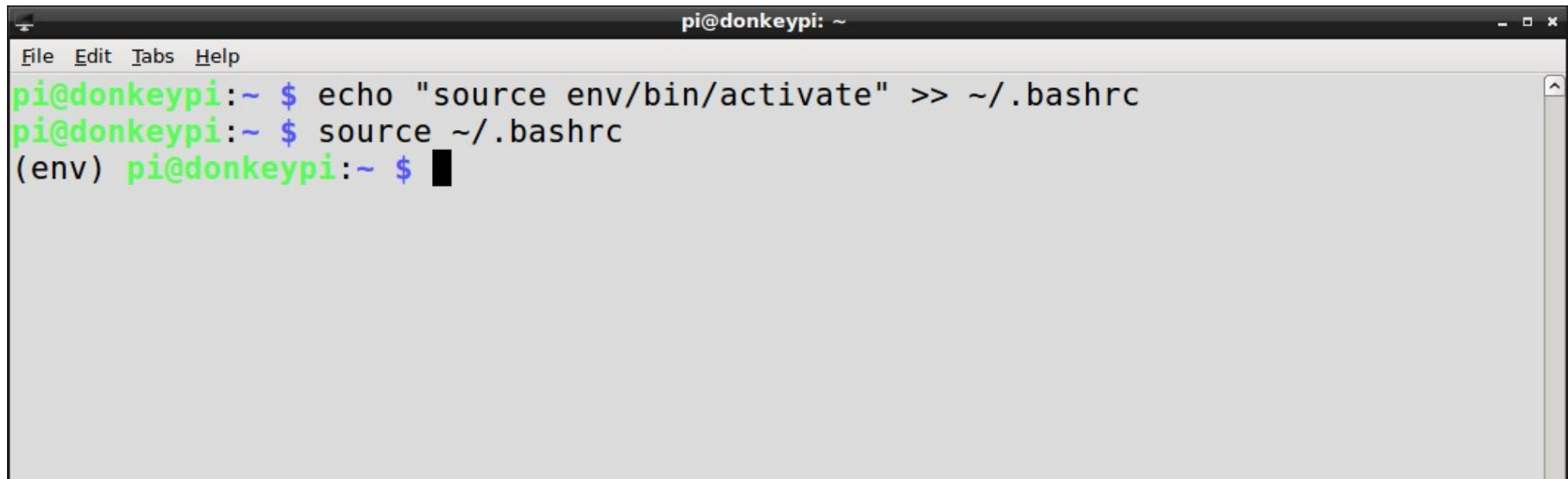
- `$ cd ~`
- `$ wget`  
`https://www.piwheels.org/simple/tensorflow/tensorflow-1.13.1-cp35-none-linux\_armv7l.whl`
- `$ sudo pip3 install tensorflow-1.13.1-cp35-none-linux_armv7l.whl`
- `$ git clone`  
`https://github.com/autorope/donkeycar`
- `$ cd donkeycar`
- `$ git checkout ec7ea7a9d`
- `$ time pip3 install -e .[pi]`

# 虛擬環境 (Virtualenv)

- Virtualenv 可以隔離函數庫需求不同的專案，讓它們不會互相影響，達到
  - 在沒有權限的情況下安裝新套件
  - 不同專案可以使用不同版本的相同套件
  - 套件版本升級時不會影響其他專案

# 加入 Python3 環境變數 (已安裝 /hackmd)

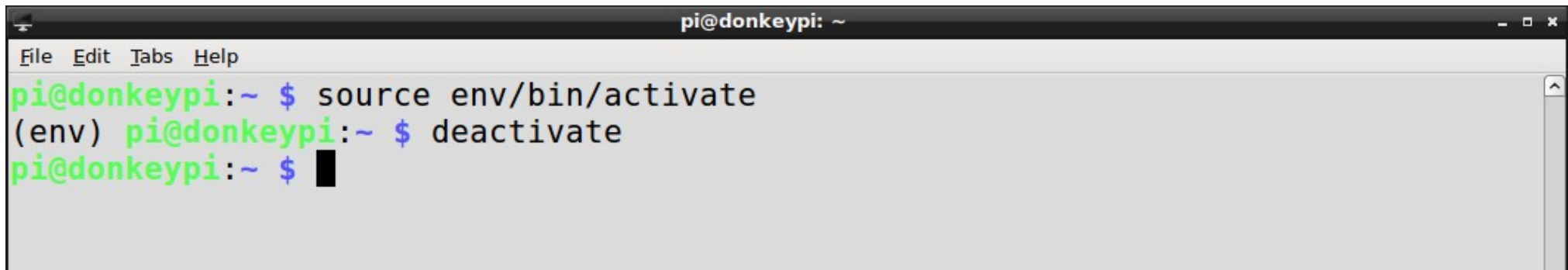
- `$ python3 -m virtualenv -p python3 env --system-site-packages`
- `$ echo "source env/bin/activate" >> ~/.bashrc`
- `$ source ~/.bashrc`
- `(env)$`

A terminal window titled 'pi@donkeypi: ~' with a menu bar containing 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the following commands and output:

```
pi@donkeypi:~ $ echo "source env/bin/activate" >> ~/.bashrc
pi@donkeypi:~ $ source ~/.bashrc
(env) pi@donkeypi:~ $
```

# 進入與退出虛擬環境

- 進入名稱為 env 的虛擬環境
- `$ source env/bin/activate`
- `(env)$` 在 env 虛擬環境下執行指令
- 退出虛擬環境
- `(env)$ deactivate`



```
pi@donkeypi: ~  
File Edit Tabs Help  
pi@donkeypi:~ $ source env/bin/activate  
(env) pi@donkeypi:~ $ deactivate  
pi@donkeypi:~ $
```



# 建立第一個 donkeycar 專案 (hackmd)

- `(env)$ cd ~`
- `(env)$ donkey createcar --path ~/mycar`
- `(env)$ cd ~/mycar`
- `(env)$ ls`

# 建立第一個 donkeycar 專案 (hackmd)

```
pi@donkeypi: ~/mycar
File Edit Tabs Help
(env) pi@donkeypi:~ $ cd ~
(env) pi@donkeypi:~ $ donkey createcar --path ~/mycar
using donkey v3.0.2 ...
Creating car folder: /home/pi/mycar
making dir /home/pi/mycar
Creating data & model folders.
making dir /home/pi/mycar/models
making dir /home/pi/mycar/data
making dir /home/pi/mycar/logs
Copying car application template: complete
Copying car config defaults. Adjust these before starting your car.
Copying train script. Adjust these before starting your car.
Copying my car config overrides
Donkey setup complete.
(env) pi@donkeypi:~ $ cd ~/mycar
(env) pi@donkeypi:~/mycar $ ls
config.py  data  logs  manage.py  models  myconfig.py  train.py
(env) pi@donkeypi:~/mycar $
```

## 實驗 2: 相機測試

目的：測試相機模組和 X-Forwarding

# 測試相機 > 拍照指令 RaspiStill

- 5 秒後拍照 ( 預設 ), 檔案 test.jpg(-o)
  - \$ cd ~/mycar
  - \$ raspistill -v -o test.jpg

**如何看照片和影片？**



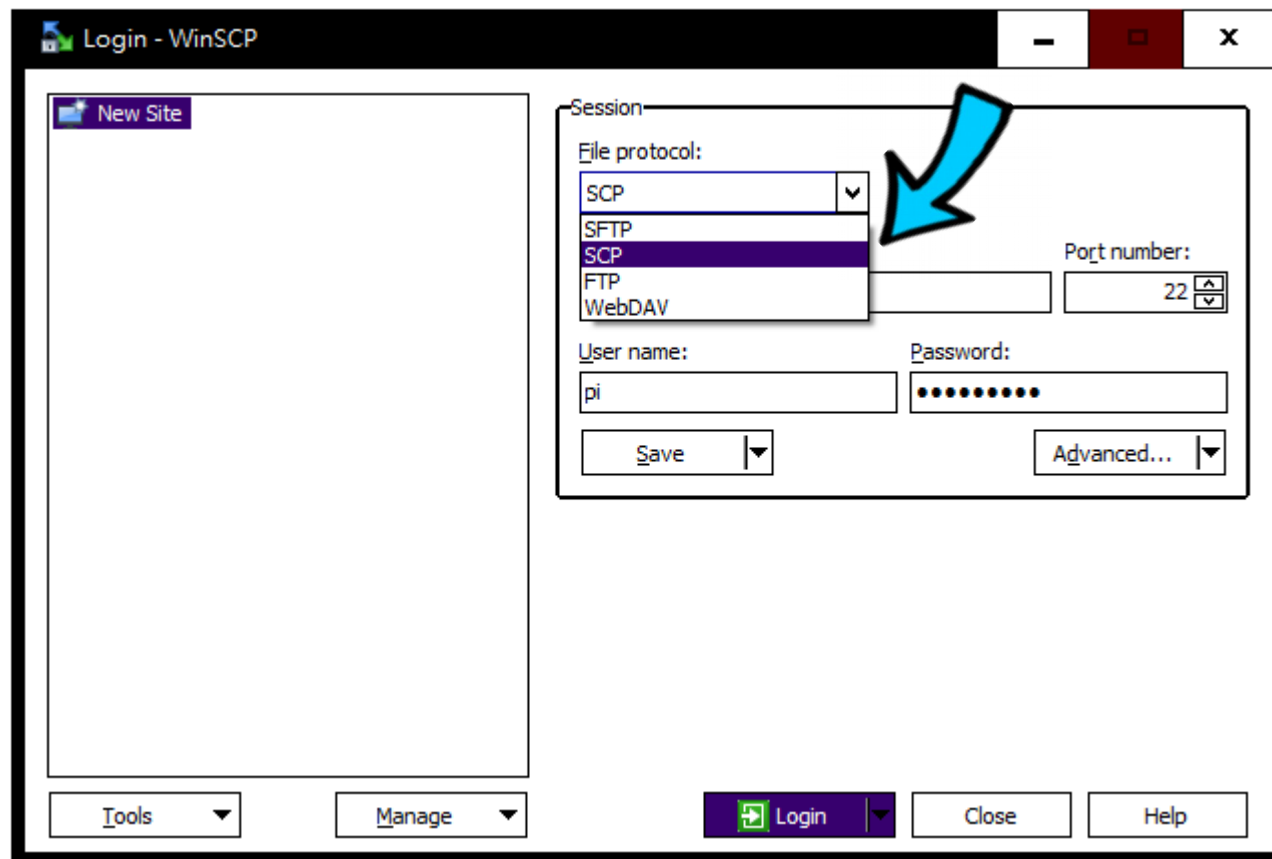
**方法一：將 Pi 的檔案傳回本機端**

# 使用 SCP

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

# 在 Windows 上安裝 WinSCP

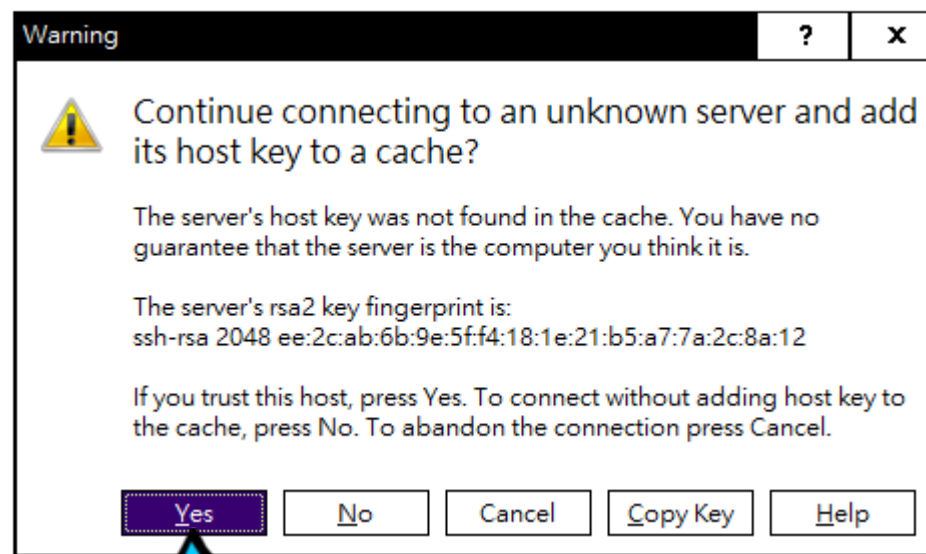
- <http://winscp.net/eng/download.php>
- 連線設定選擇 SCP



<http://winscp.net/>

# 在 Windows 上安裝 WinSCP

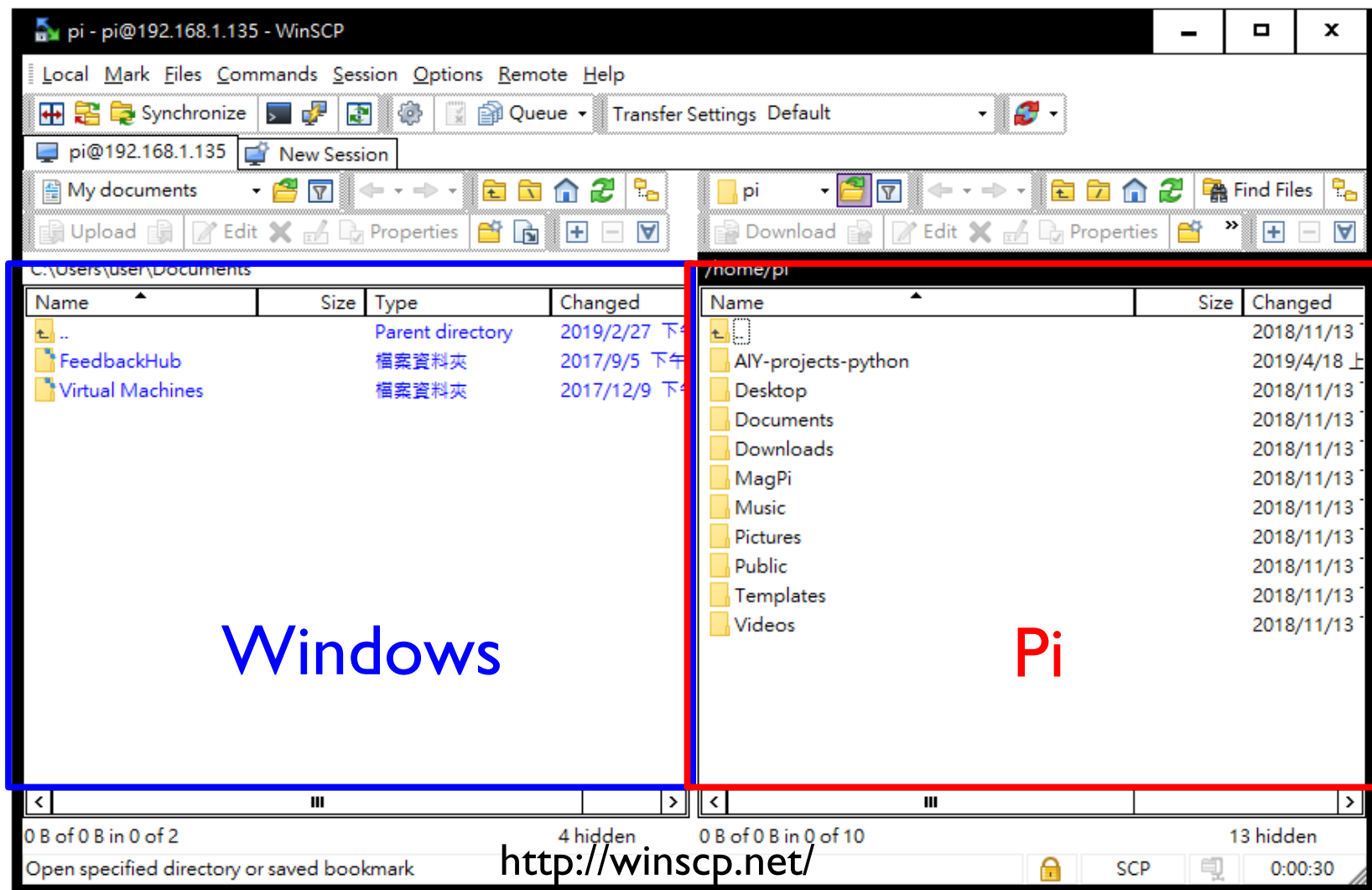
- <http://winscp.net/eng/download.php>
- 連線設定選擇 SCP
- 接受交換金鑰



<http://winscp.net/>

# 在 Windows 上安裝 WinSCP

- 左右兩邊都可以做檔案傳輸





如果是 Linux 或是 Mac OS

```
$ scp pi@192.168.2.2:~/test.jpg .
```



換成自己的 pi 的 IP

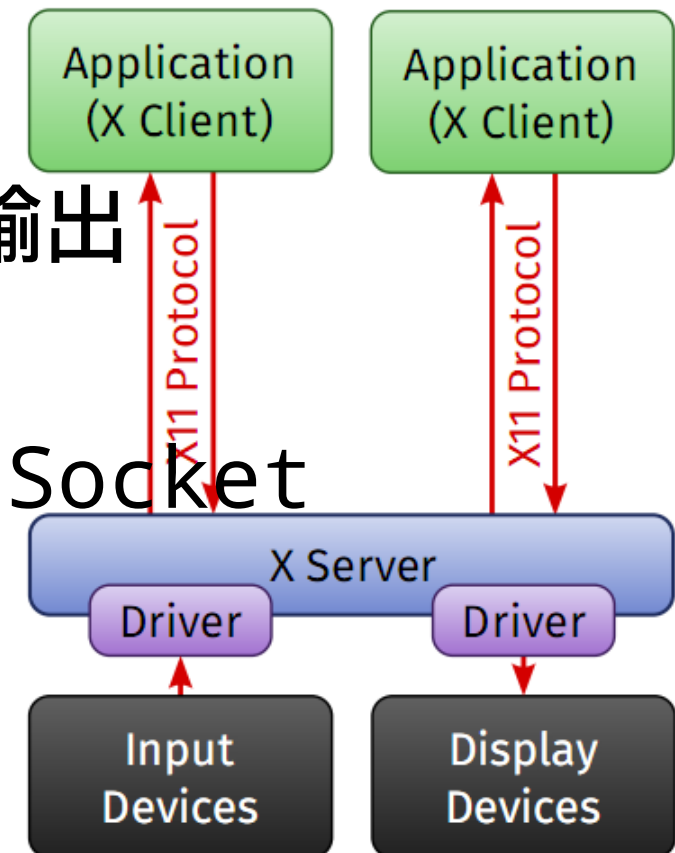


點

方法二：使用 X11 Forwarding

# X Window System

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



# 在 Windows 安裝 X Server

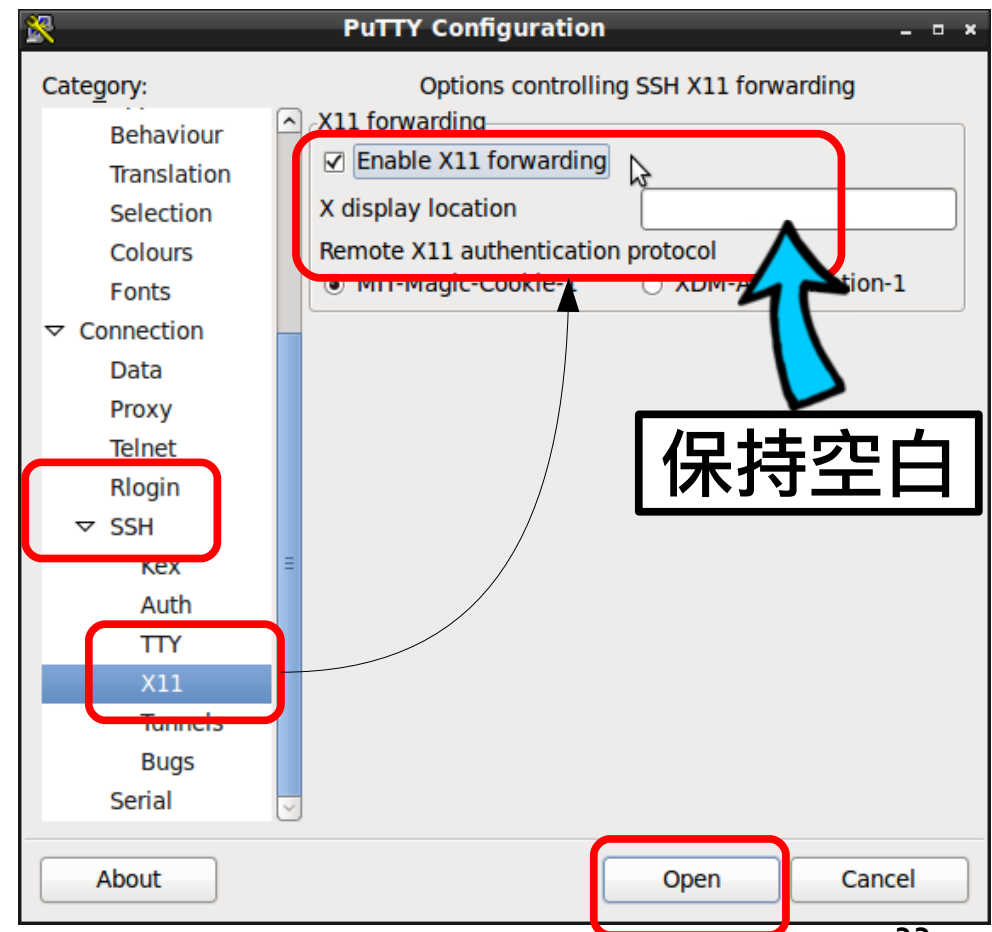
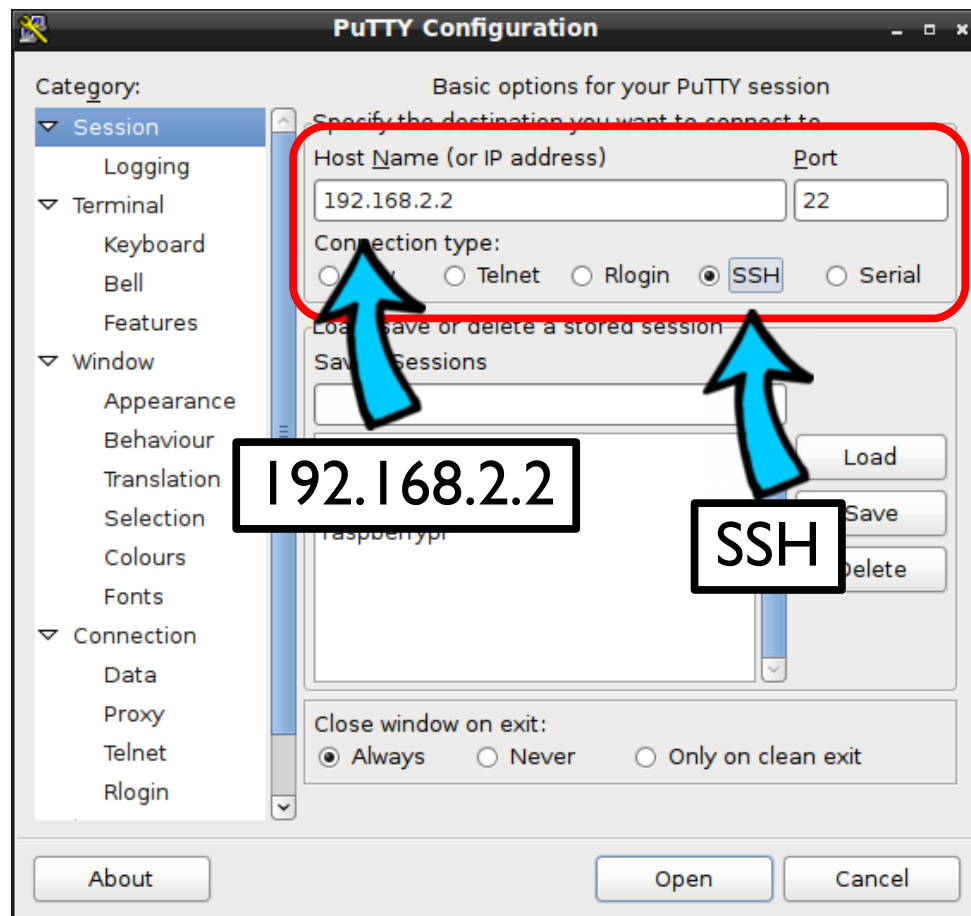
- 安裝 Xming, 下一步到底

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



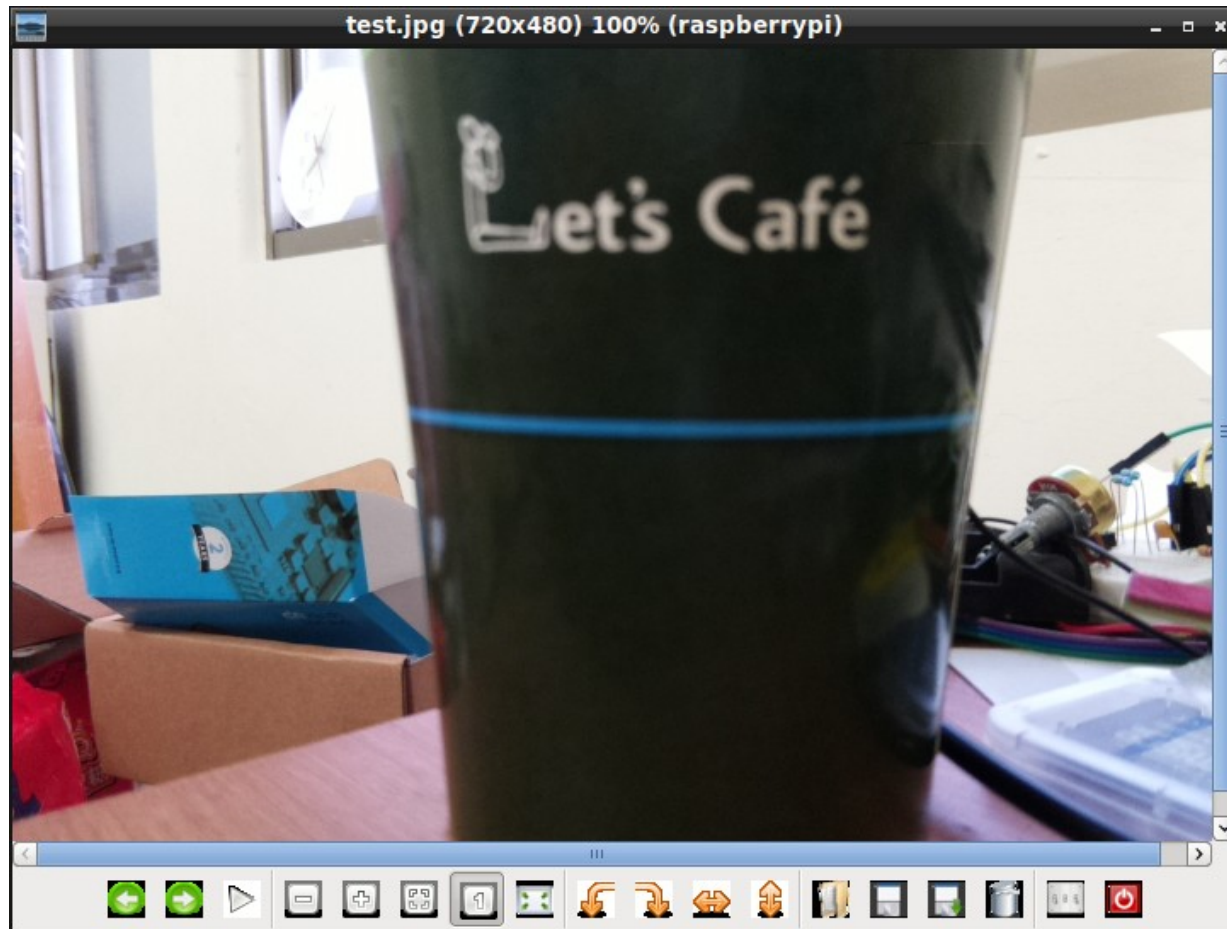
# 在 Windows 設定 X11 Forwarding

- SSH > X11 > Enable X11 forwarding



# X11 Forwarding 連線成功後

- 看照片
  - `$ gpicview test.jpg`



如果是 Linux 或是 Mac OS

```
$ ssh -X pi@192.168.2.2
```



換成自己的 pi 的 IP

# “Can not open display” on Mac OS X

- 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



# 實驗 3：輪子校正

目的：校正油門和轉向

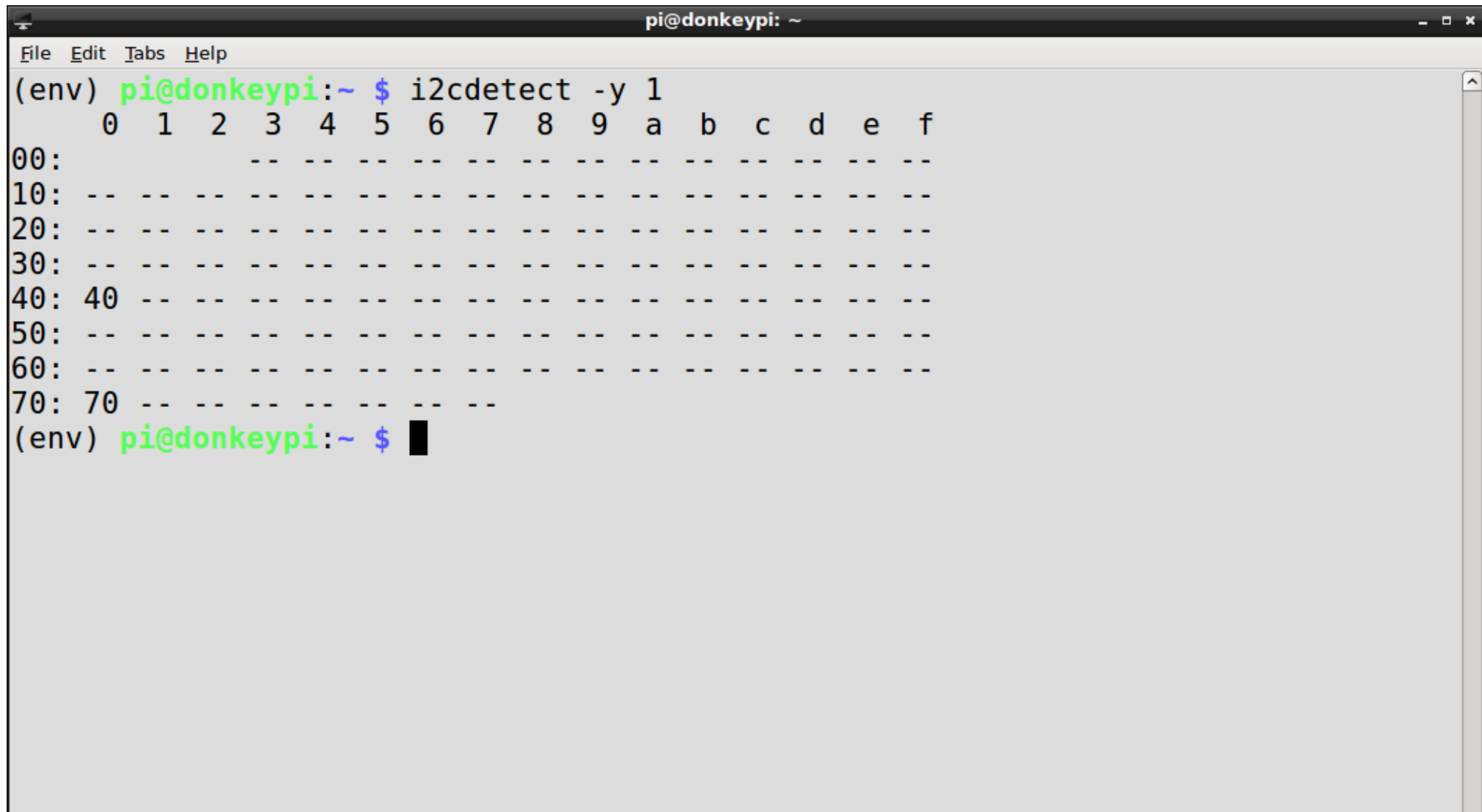
# 先把車架高！



<http://bit.ly/2Z0GeYH>

# 確認 PCA9685 馬達控制板接線正確

- (env)\$ i2cdetect -y 1

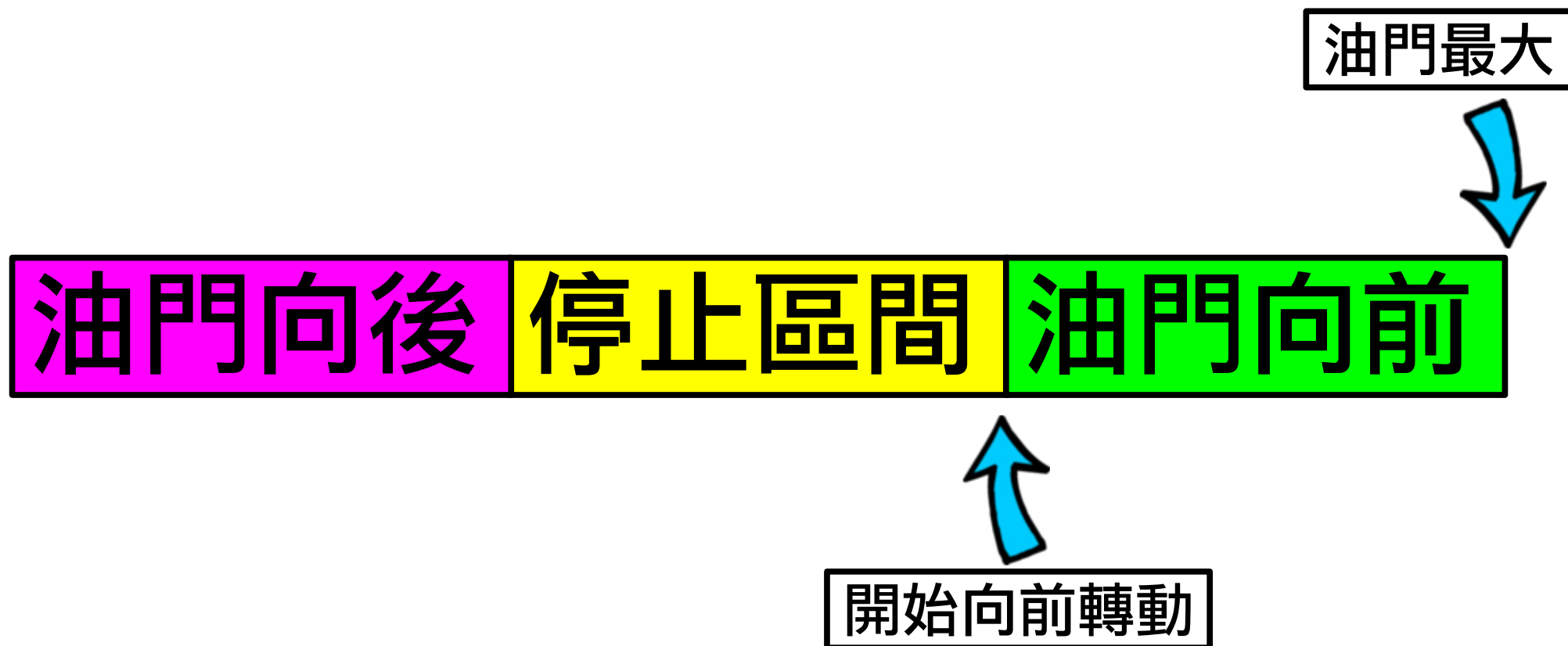


The screenshot shows a terminal window titled "pi@donkeypi: ~". The command "(env) pi@donkeypi:~ \$ i2cdetect -y 1" has been executed. The output is a grid of hexadecimal addresses (00 to 70) and their corresponding I2C device IDs. Most addresses show "--", indicating no device is connected. Address 40 shows "40" and address 70 shows "70", indicating that devices are connected at these addresses.

```
(env) pi@donkeypi:~ $ i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- -- -- -- -- -- -- -- --
(env) pi@donkeypi:~ $
```

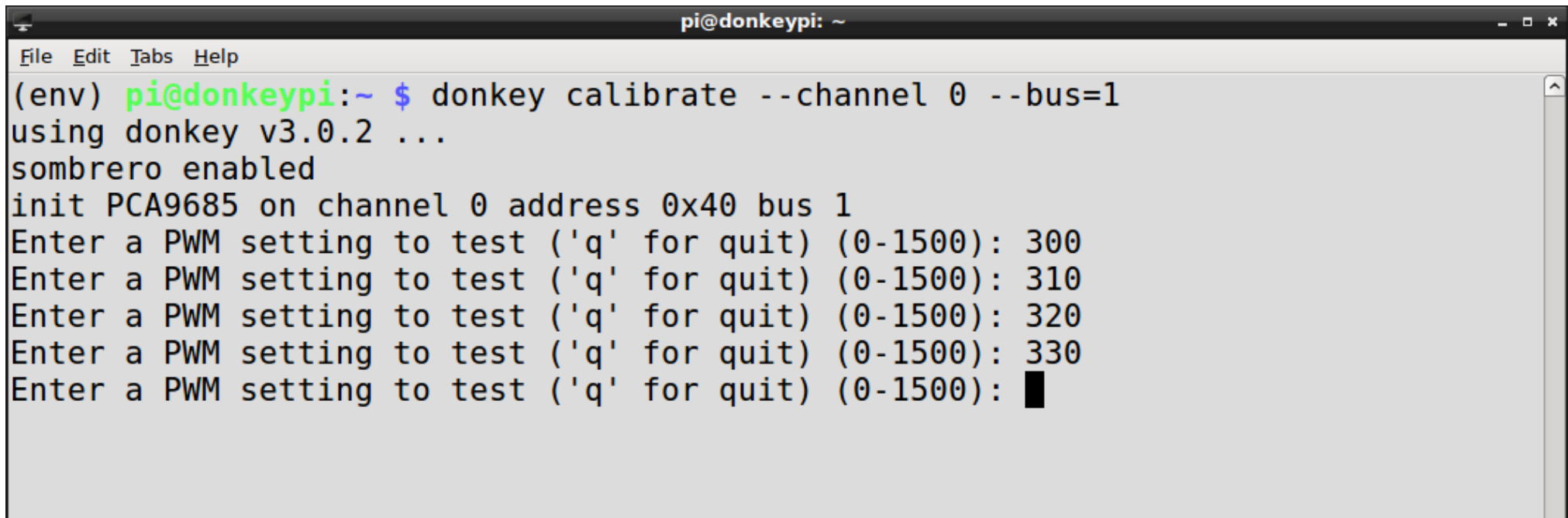
# 測試油門前進 (Throttle)

- 目標：找出油門開始 / 停止區間



# 測試油門前進 (Throttle) (hackmd)

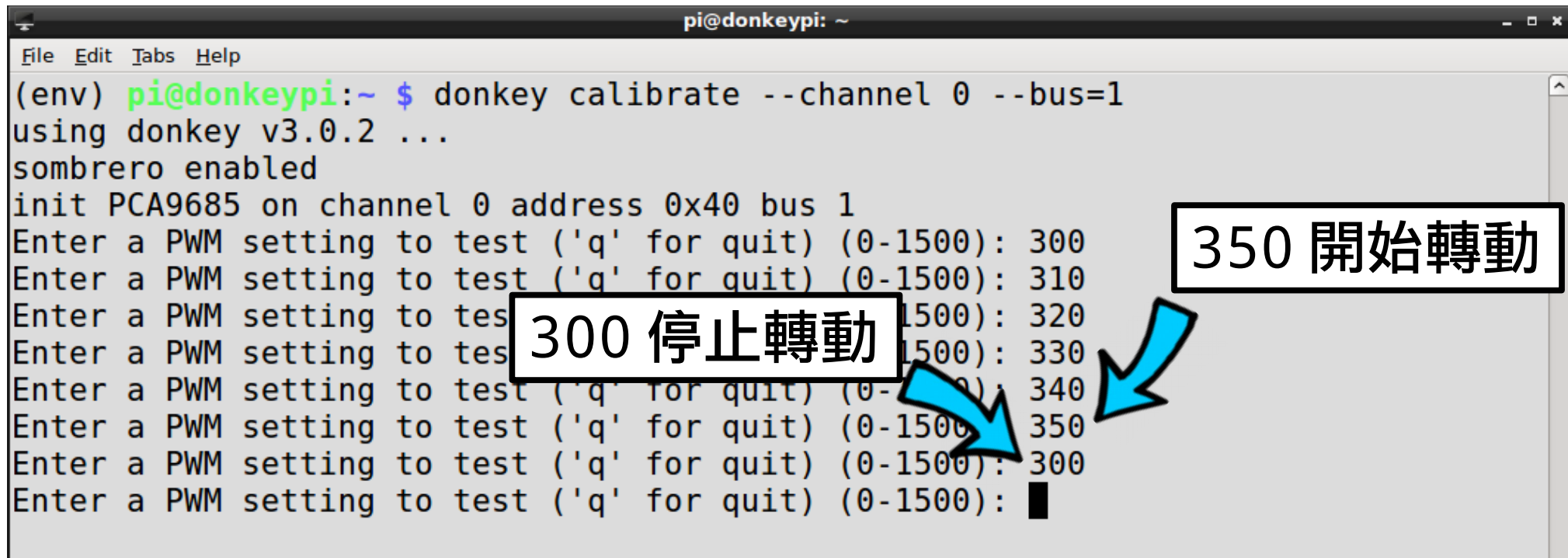
- 目標：找出油門開始轉動 PWM 值（每台車都會不同）
- 步驟：從 300 開始，每次加 10，直到輪胎開始轉動
- `(env)$ donkey calibrate --channel 0 --bus=1`

A screenshot of a terminal window titled 'pi@donkeypi: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal shows the command '(env) pi@donkeypi:~ \$ donkey calibrate --channel 0 --bus=1' being executed. The output includes 'using donkey v3.0.2 ...', 'sombbrero enabled', and 'init PCA9685 on channel 0 address 0x40 bus 1'. It then prompts the user to 'Enter a PWM setting to test ('q' for quit) (0-1500):' with values 300, 310, 320, and 330 entered sequentially. The last prompt is followed by a black square cursor.

```
pi@donkeypi: ~
File Edit Tabs Help
(env) pi@donkeypi:~ $ donkey calibrate --channel 0 --bus=1
using donkey v3.0.2 ...
sombbrero enabled
init PCA9685 on channel 0 address 0x40 bus 1
Enter a PWM setting to test ('q' for quit) (0-1500): 300
Enter a PWM setting to test ('q' for quit) (0-1500): 310
Enter a PWM setting to test ('q' for quit) (0-1500): 320
Enter a PWM setting to test ('q' for quit) (0-1500): 330
Enter a PWM setting to test ('q' for quit) (0-1500): █
```

# 測試油門前進 (Throttle) (hackmd)

- 目標：找出油門停止區間（每台車都會不同）
- 步驟：如果是停止區間，則開始轉動後再輸入區間值將停止轉動
  - > 例如 350 是開始轉動，300 是停止區間
- (env)\$ donkey calibrate --channel 0 --bus=1



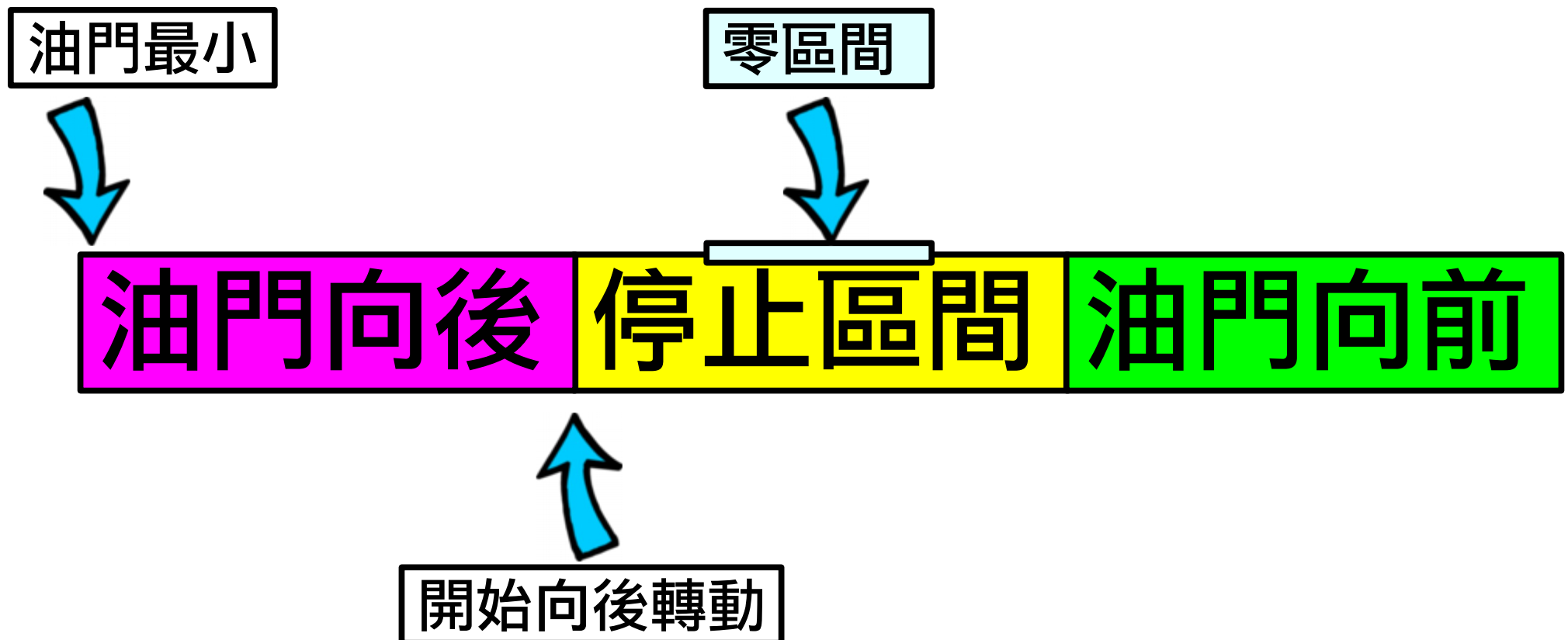
```
pi@donkeypi: ~  
File Edit Tabs Help  
(env) pi@donkeypi:~ $ donkey calibrate --channel 0 --bus=1  
using donkey v3.0.2 ...  
sombbrero enabled  
init PCA9685 on channel 0 address 0x40 bus 1  
Enter a PWM setting to test ('q' for quit) (0-1500): 300  
Enter a PWM setting to test ('q' for quit) (0-1500): 310  
Enter a PWM setting to test ('q' for quit) (0-1500): 320  
Enter a PWM setting to test ('q' for quit) (0-1500): 330  
Enter a PWM setting to test ('q' for quit) (0-1500): 340  
Enter a PWM setting to test ('q' for quit) (0-1500): 350  
Enter a PWM setting to test ('q' for quit) (0-1500): 300  
Enter a PWM setting to test ('q' for quit) (0-1500): █
```

300 停止轉動

350 開始轉動

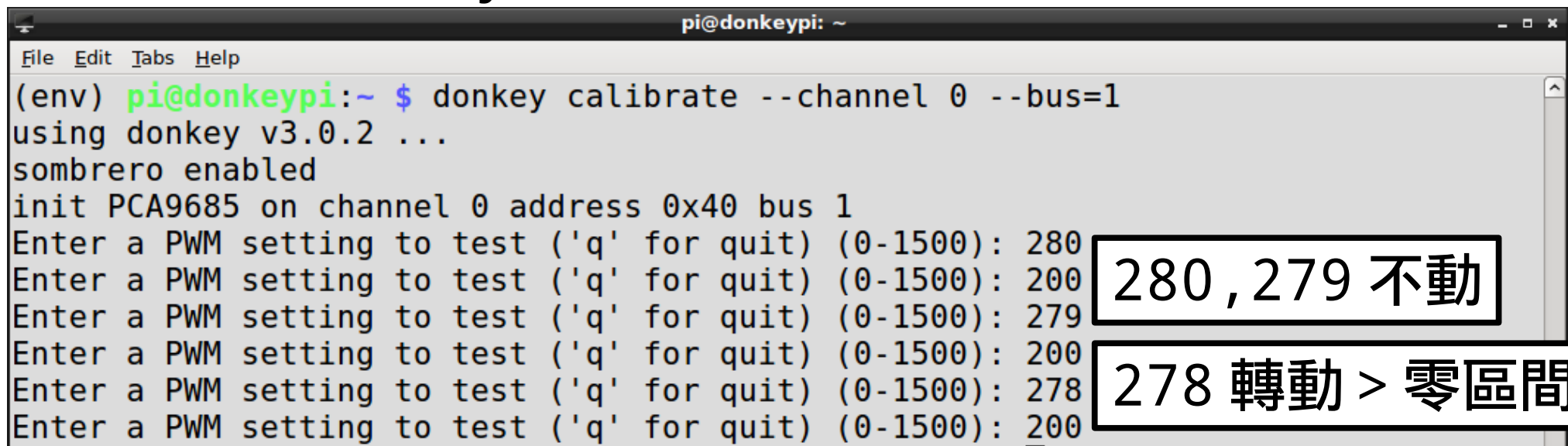
# 測試油門後退 (Throttle)

- 目標：找出零點 / 開始向後
- 注意：油門要向後轉動的必要條件是找到零區間



# 測試油門後退 (Throttle)

- 目標：找出油門零區間（每台車都會不同）
- 步驟：假設 200 會向後轉，並且先假設零區間為 280
  - > 先輸入 200 觀察是否會向後轉，如果否零區間持續遞減
  - > 直到輪胎開始向後轉，剛剛找到為零區間
  - > 如果遞減找不到零點，就需要遞增
- (env)\$ donkey calibrate --channel 0 --bus=1



```
pi@donkeypi: ~
File Edit Tabs Help
(env) pi@donkeypi:~ $ donkey calibrate --channel 0 --bus=1
using donkey v3.0.2 ...
sombbrero enabled
init PCA9685 on channel 0 address 0x40 bus 1
Enter a PWM setting to test ('q' for quit) (0-1500): 280
Enter a PWM setting to test ('q' for quit) (0-1500): 200
Enter a PWM setting to test ('q' for quit) (0-1500): 279
Enter a PWM setting to test ('q' for quit) (0-1500): 200
Enter a PWM setting to test ('q' for quit) (0-1500): 278
Enter a PWM setting to test ('q' for quit) (0-1500): 200
```

280, 279 不動

278 轉動 > 零區間



# 測試油門後退 (Throttle)

- 目標：找出油門最小 PWM 值（每台車都會不同）
- 步驟：輸入任意值，如果不會轉動表示不是最小值
  - > 先輸入任意值後觀察轉動行為，再輸入零區間
- (env)\$ donkey calibrate --channel 0 --bus=1

```
pi@donkeypi: ~
File Edit Tabs Help
(env) pi@donkeypi:~ $ donkey calibrate --channel 0 --bus=1
using donkey v3.0.2 ...
sombrero enabled
init PCA9685 on channel 0 address 0x40 bus 1
Enter a PWM setting to test ('q' for quit) (0-1500): 278
Enter a PWM setting to test ('q' for quit) (0-1500): 200
Enter a PWM setting to test ('q' for quit) (0-1500): 278
Enter a PWM setting to test ('q' for quit) (0-1500): 190
Enter a PWM setting to test ('q' for quit) (0-1500): 278
Enter a PWM setting to test ('q' for quit) (0-1500): 191
Enter a PWM setting to test ('q' for quit) (0-1500):
```

200 轉動

190 不動

191 轉動 > 最小值

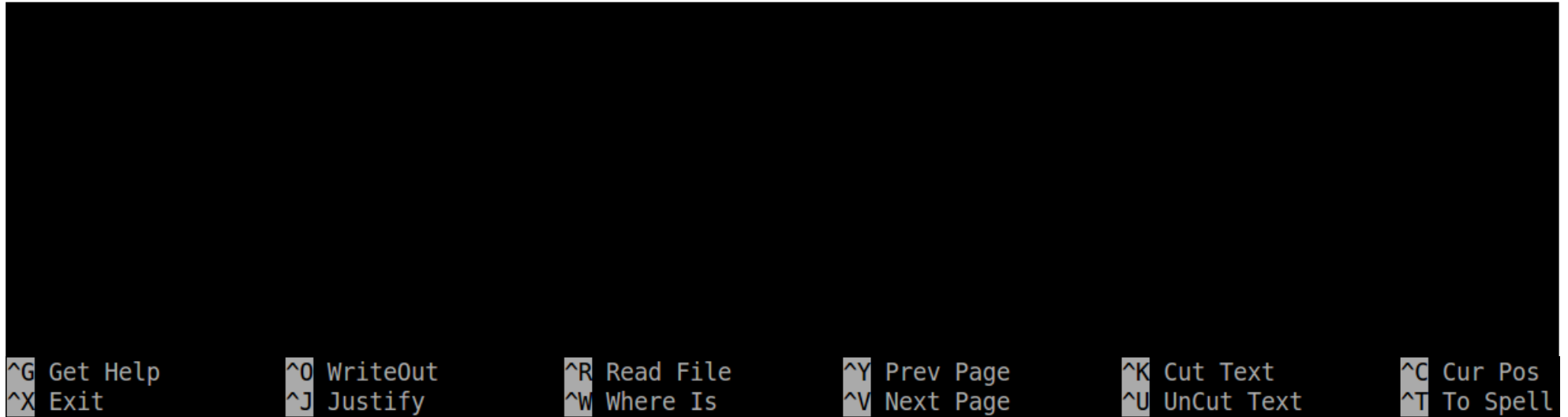
# 修改 myconfig.py

- 將油門 PWM 向前 / 零區間 / 向後填入 myconfig.py

去除 # 並且靠左對齊

```
pi@donkeypi: ~/mycar
File Edit Tabs Help
38 # DRIVE_TRAIN_TYPE = "SERVO_ESC" # SERVO_ESC|DC_STEER_THROTTLE|DC_TWO_WHEEL|
SERVO_HBRIDGE_PWM
39 #
40 #
41 # 1 #channel on the 9685 pwm board 0-15
42 # STEERING_LEFT_PWM = 460 #pwm value for full left steering
43 # STEERING_RIGHT_PWM = 290 #pwm value for full right steering
44 #
45 # #THROTTLE
46 # THROTTLE_CHANNEL = 0 #channel on the 9685 pwm board 0-15
47 THROTTLE_FORWARD_PWM = 595 #pwm value for max forward throttle
48 THROTTLE_STOPPED_PWM = 308 #pwm value for no movement
49 THROTTLE_REVERSE_PWM = 190 #pwm value for max reverse throttle
50 #
51 # #DC_STEER_THROTTLE with one motor as steering, one as drive
52 # #these GPIO pinouts are only used for the DRIVE_TRAIN_TYPE=DC_STEER_THROTTLE
53 # HBRIDGE_PIN_LEFT = 18
54 # HBRIDGE_PIN_RIGHT = 16
55 # HBRIDGE_PIN_FWD = 15
56 # HBRIDGE_PIN_BWD = 13
57 #
58 # #DC_TWO_WHEEL - with two wheels as drive, left and right.
-- VISUAL LINE -- 49,1 18%
```

# nano 編輯器使用



使用 : nano 檔名 ( 例如 nano myconfig.py )

離開 : Ctrl + x

> 令存新檔 : y

> 不存離開 : n

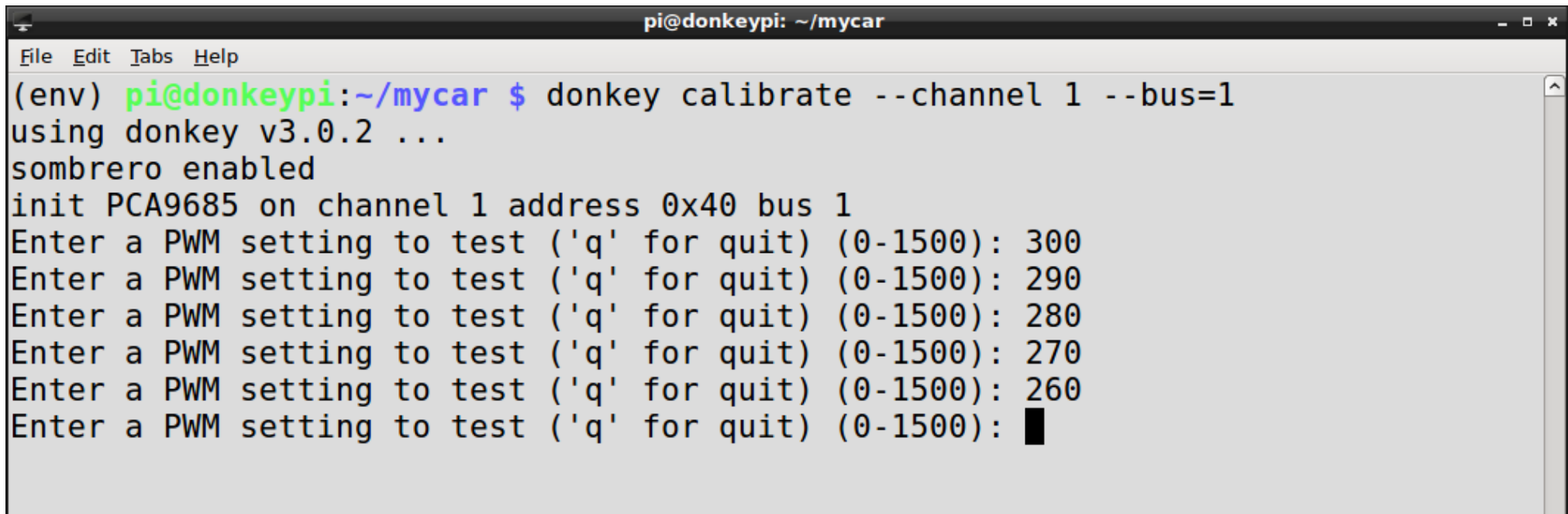
> 離開 : Ctrl + c

# 有刷馬達參考油門 (Throttle) 數值

- 油門 (Throttle) 原始數值
  - 向前最大 590
  - 中間 310
  - 向後最大 190
- myconfig.py 參數 ( 參考值 , 需依實際修改 )
  - THROTTLE\_FORWARD\_PWM = 350
  - THROTTLE\_STOPPED\_PWM = 310
  - THROTTLE\_REVERSE\_PWM = 280

# 測試轉向 (Steering) 向右 (hackmd)

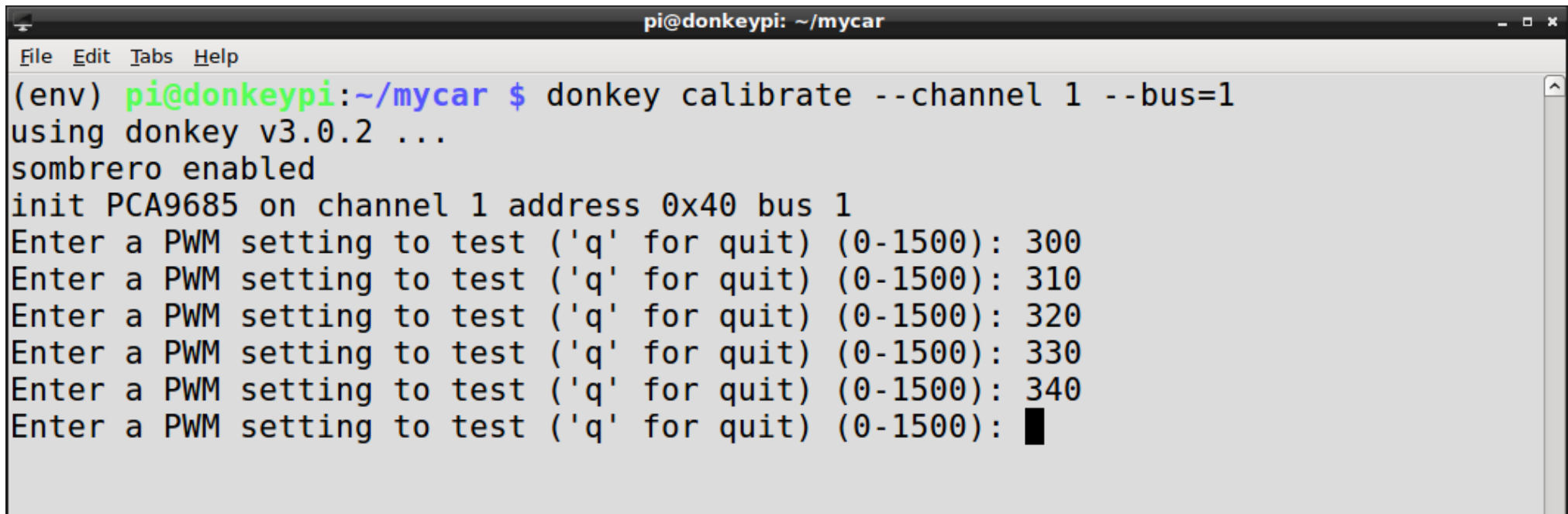
- 目標：找出轉向 PWM 最右
- 步驟：從 300 開始，每次減 10，直到輪胎不再轉動（最右）
- `(env)$ donkey calibrate --channel 1 --bus=1`

A terminal window titled 'pi@donkeypi: ~/mycar' with a menu bar (File, Edit, Tabs, Help). The terminal shows the command '(env) pi@donkeypi:~/mycar \$ donkey calibrate --channel 1 --bus=1' being executed. The output includes 'using donkey v3.0.2 ...', 'sombbrero enabled', and 'init PCA9685 on channel 1 address 0x40 bus 1'. It then prompts for PWM settings, showing a sequence of values from 300 down to 260, with the last prompt ending in a black square cursor.

```
pi@donkeypi: ~/mycar
File Edit Tabs Help
(env) pi@donkeypi:~/mycar $ donkey calibrate --channel 1 --bus=1
using donkey v3.0.2 ...
sombbrero enabled
init PCA9685 on channel 1 address 0x40 bus 1
Enter a PWM setting to test ('q' for quit) (0-1500): 300
Enter a PWM setting to test ('q' for quit) (0-1500): 290
Enter a PWM setting to test ('q' for quit) (0-1500): 280
Enter a PWM setting to test ('q' for quit) (0-1500): 270
Enter a PWM setting to test ('q' for quit) (0-1500): 260
Enter a PWM setting to test ('q' for quit) (0-1500): █
```

# 測試轉向 (Steering) 向左

- 目標：找出轉向 PWM 最左
- 步驟：從 300 開始，每次加 10，直到輪胎不再轉動（最左）
- `(env)$ donkey calibrate --channel 1 --bus=1`



```
pi@donkeypi: ~/mycar
File Edit Tabs Help
(env) pi@donkeypi:~/mycar $ donkey calibrate --channel 1 --bus=1
using donkey v3.0.2 ...
sombbrero enabled
init PCA9685 on channel 1 address 0x40 bus 1
Enter a PWM setting to test ('q' for quit) (0-1500): 300
Enter a PWM setting to test ('q' for quit) (0-1500): 310
Enter a PWM setting to test ('q' for quit) (0-1500): 320
Enter a PWM setting to test ('q' for quit) (0-1500): 330
Enter a PWM setting to test ('q' for quit) (0-1500): 340
Enter a PWM setting to test ('q' for quit) (0-1500): █
```

# 驗證轉向 (Steering) 數值

- 中間值應該是 ( 最左 + 最右 ) / 2 , 如果該值不在中間則錯
- 也可以先找到中間值以後加減各 100 為最左和最右值
- 可實際向前走一公尺 , 確定是否為中間值

# 修改 myconfig.py

- 將轉向 PWM 最右 / 最左值填入 myconfig.py

去除 # 並且靠左對齊

```
pi@donkeypi: ~/mycar
File Edit Tabs Help
32 #
33 # #DRIVETRAIN
34 # #DC_TWO_WHEEL uses HBridge pwm to control two drive motors, one on the left, and one on the right
35 #
36 # #SERVO_HBRIDGE_PWM use ServoBlaster to output pwm control from the PiZero directly to control steering, and HBridge for a drive motor.
37 # #DRIVE_TRAIN_TYPE = "SERVO_ESC" # SERVO_ESC|DC_STEER_THROTTLE|DC_TWO_WHEEL|SERVO_HBRIDGE_PWM
38 #
39 # #STEERING
40 #
41 STEERING_CHANNEL = 1 #channel on the 9685 pwm board 0-15
42 STEERING_LEFT_PWM = 460 #pwm value for full left steering
43 STEERING_RIGHT_PWM = 260 #pwm value for full right steering
44 #
45 # #THROTTLE
46 # THROTTLE_CHANNEL = 0 #channel on the 9685 pwm board 0-15
47 # THROTTLE_FORWARD_PWM = 500 #pwm value for max forward throttle
48 # THROTTLE_STOPPED_PWM = 370 #pwm value for no movement
49 # THROTTLE_REVERSE_PWM = 220 #pwm value for max reverse throttle
50 #
51 # #DC_STEER_THROTTLE with one motor as steering, one as drive
52 # #these GPIO pinouts are only used for the DRIVE_TRAIN_TYPE=DC_STEER_THROTTLE
53 # HBRIDGE_PIN_LEFT = 18
54 # HBRIDGE_PIN_RIGHT = 16
-- VISUAL LINE --
```

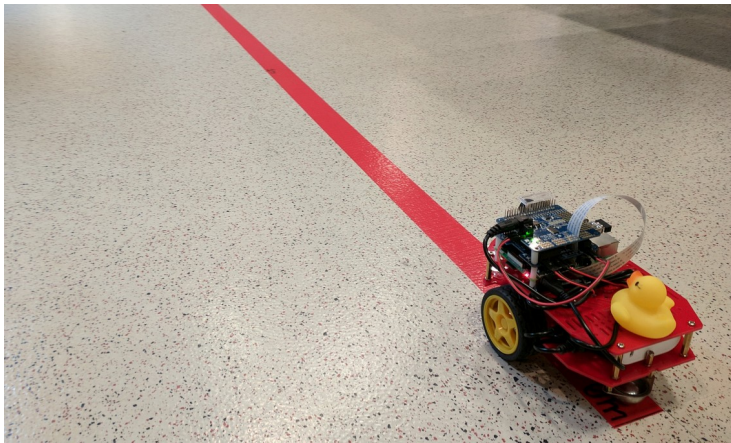


# E600I 參考轉向 (Steering) 數值

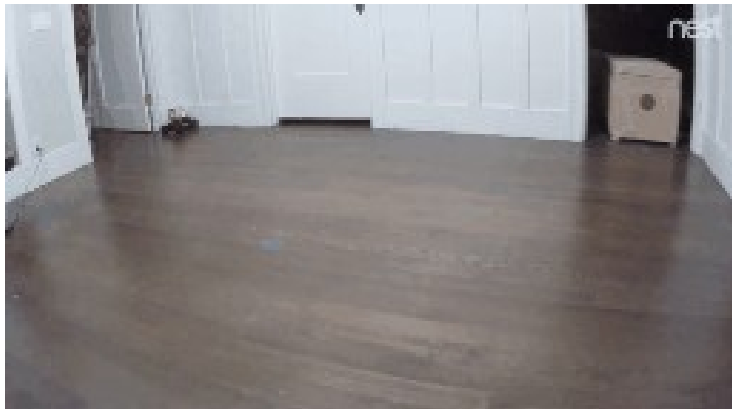
- 轉向 (Steering) 數值
  - 最右 460
  - 中間 360
  - 最小 260
- myconfig.py 參數
  - STEERING\_LEFT\_PWM = 460
  - STEERING\_RIGHT\_PWM = 260

# 轉向 (Steering) 校正重點

- 1. 直行一公尺會偏移嗎？

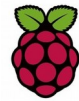


- 2. 往左和往右的 PWM 數值迴轉半徑相同嗎？



# 使用瀏覽器控制

- (env)\$ cd ~/mycar
- (env)\$ python3 manage.py drive



<ipaddress:8887>

Control Mode 

Joystick

Gamepad

Device Tilt

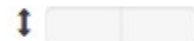
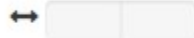
Max Throttle

Select Max Throttle ▼

Throttle Mode

User ▼

Angle &  
Throttle



Mode & Pilot

User (d) ▼

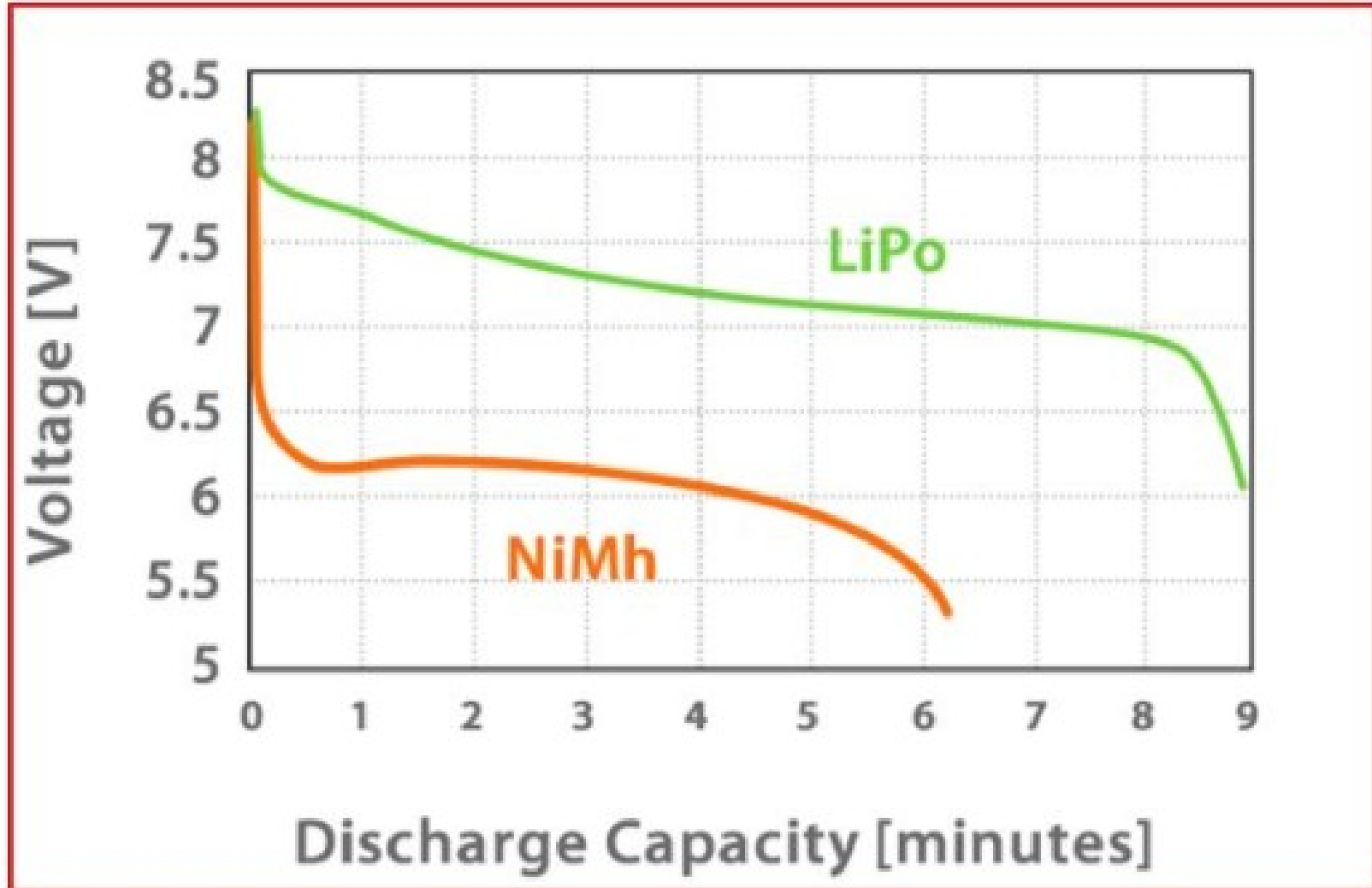
Start Recording



Click/touch to use joystick.

鍵盤 i / k ，表示馬達向前轉 / 馬達向後轉  
鍵盤 j / l ，表示車向左 / 車向右  
按空白鍵停止

# 校正要在鎳氫電池充飽的情況下做



## **實驗 4：測試搖桿**

**目的：使用 2.4GHz 搖桿控制車子**

## 2.4G 搖桿

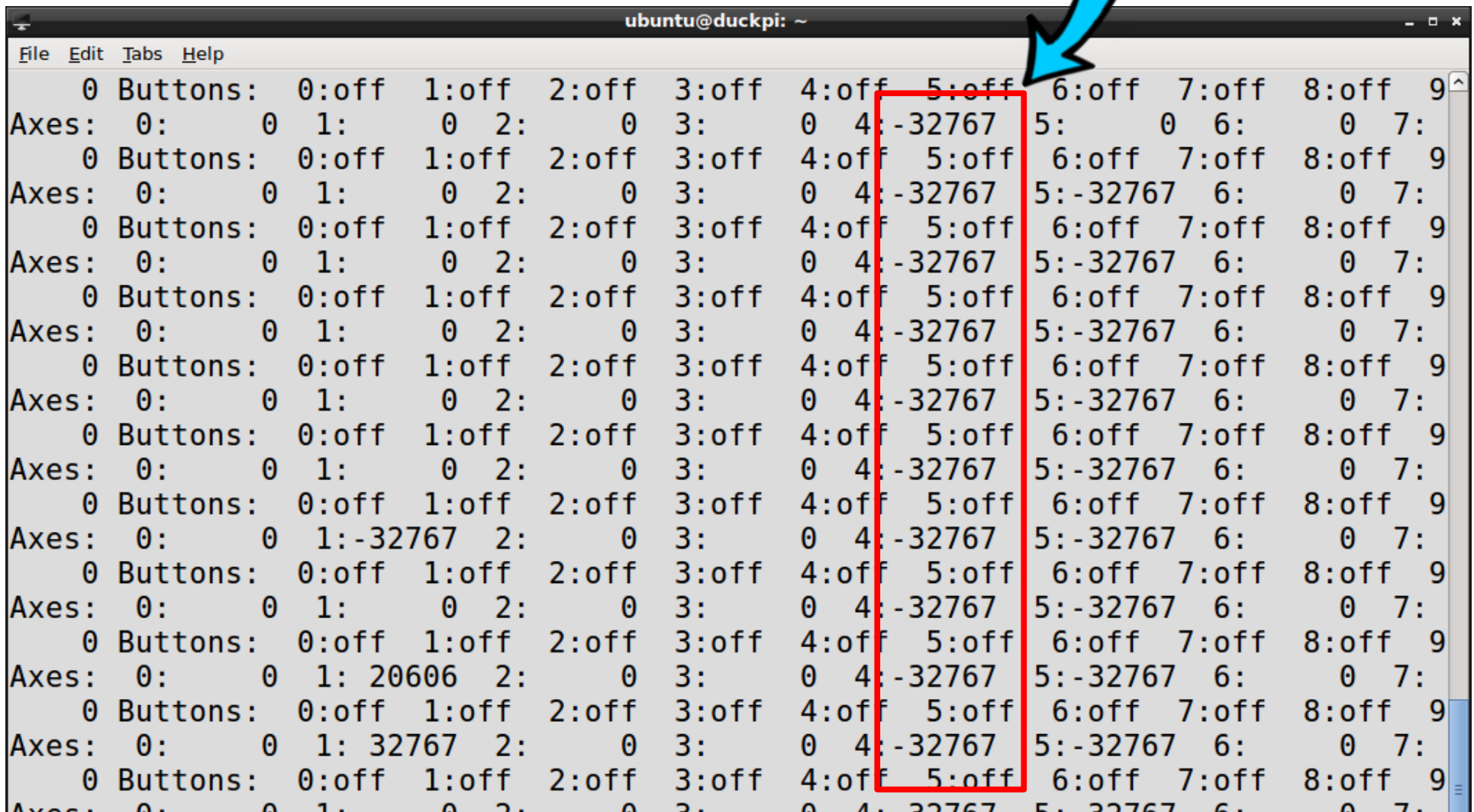


電源開關

# 測試搖桿 (hackmd)

- `$ jstest /dev/input/js0`

如果有數值改變



```
ubuntu@duckpi: ~  
File Edit Tabs Help  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: 0 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: -32767 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 20606 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 32767 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:  
 0 Buttons: 0:off 1:off 2:off 3:off 4:off 5:off 6:off 7:off 8:off 9  
Axes: 0: 0 1: 0 2: 0 3: 0 4: -32767 5: -32767 6: 0 7:
```



# 測試搖桿 ( 除錯 )

- 如果搖桿沒反應，重新插拔 USB 接收器
- 將 off 切到 on 並快速按搖桿上的 start
- 再執行 jstest 確認是否配對成功

# 先把車架高！



<http://bit.ly/2Z0GeYH>

# 使用搖桿控制 Donkey Car 步驟

- 1. 安裝搖桿控制程式
- 2. 使用搖桿控制

# I. 安裝搖桿控制程式 ( 未安裝 /hackmd)

- (env)\$ cd ~
- (env)\$ git clone [https://github.com/raspberrypi-tw/donkeypart\\_ps3\\_controller](https://github.com/raspberrypi-tw/donkeypart_ps3_controller)
- (env)\$ cd ~/donkeypart\_ps3\_controller
- (env)\$ python3 setup.py install

## 2. 使用搖桿控制 (hackmd)

- 用範例執行 ( `sample_manage.py` )
- (env)\$ `cd ~/mycar`
- (env)\$ `cp ~/donkeypart_ps3_controller/sample_manage.py .`
- (env)\$ `python3 sample_manage.py drive --js`



有一個點

# 注意事項

- 先確定搖桿是否配對成功？
- 左右控制和前後都會拍照（收資料）
- 搖桿控制是數位資料，比例從 `myconfig.py` 修改

# 測試搖桿和 Donkey Car 的移動行為

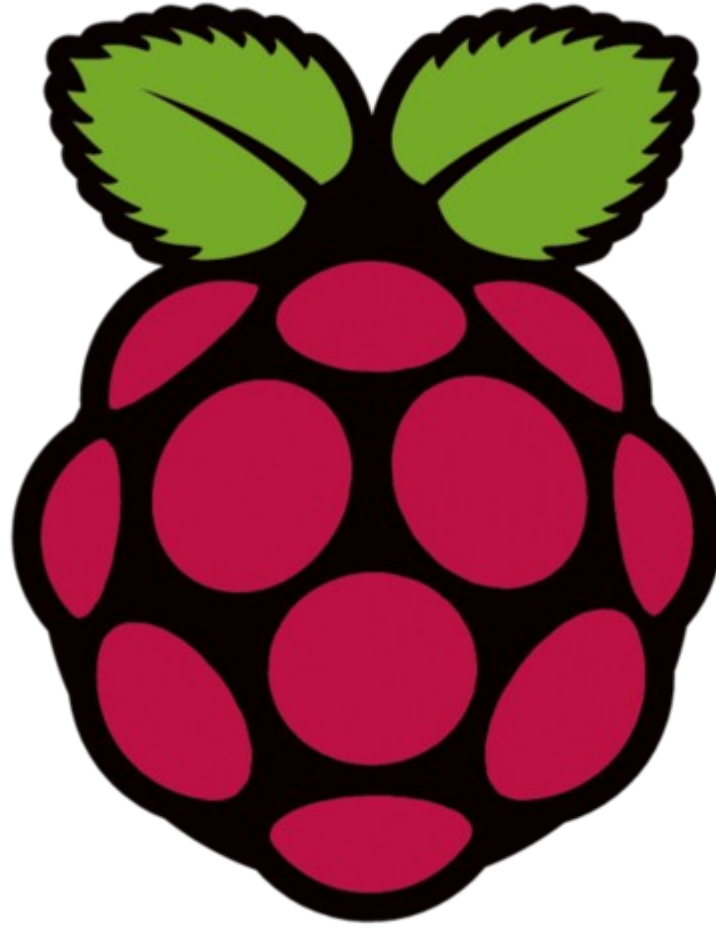


### 3. 檢查蒐集資料結果

- 如果使用搖桿控制，當控制前後或是左右後將會自動照相（每秒 30 張）並紀錄搖桿控制結果（方向和油門）
- `(env)$ cd ~/mycar`
- `(env)$ ls data`
- 如果出現類似 `tub_1_19-09-18` 的目錄，檢查該目錄裡面是否有一堆 `XXX_cam-image_array_.jpg` 和 `record_XXX.json` 檔案（XXX 表示數字）
- 後面將會使用 `.jpg` 和 `.json` 做訓練



# Raspberry Pi Rocks the World



Thanks