AIoT

用Python學智慧聯網

Chapter 06：二元分類 – 雲端步頻紀錄儀

旗標創客

踏入 AIoT 的世界

1 何謂 AIoT

2 用Python 玩轉 AI

3 AI 的小大腦 – 微控制器

4 迴歸問題 – 體溫監測站

5 IoT 應用 – 體溫通報器

6 二元分類 – 雲端步頻紀錄儀

7 多元分類 – 無線體感鍵盤

8 CNN – 智慧聲控燈

旗標創客

前言

- 睡眠品質、步數記錄...等數據都是很重要的身體資訊。本章要自製１台『計步器』來蒐集『步頻』，並將其上傳到雲端 。

前言

- 6-1　計步器簡介
- 6-2　六軸感測器
  - LAB08 顯示六軸感測器資訊
- 6-3　六軸感測器數據分析
- 6-4　按鈕開關
  - LAB09 按鈕開關測試
- 6-5　二元分類
- 6-6　實作：步頻記錄儀
  - LAB10 實作：步頻記錄儀 – 蒐集資料
  - LAB11 步頻記錄儀
- 6-7　IoT 應用 – 雲端步頻記錄儀
  - LAB12 雲端步頻記錄儀
  - LAB13 手機步頻監測

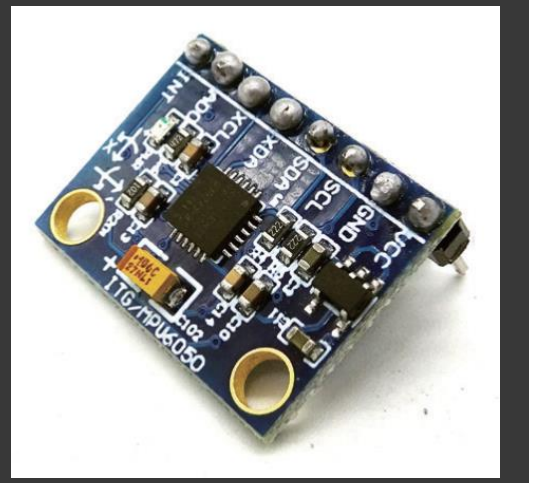
旗標創客

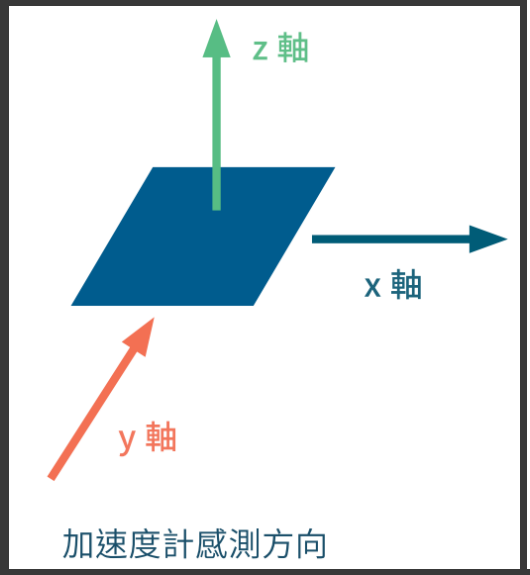
6-1 計步器簡介

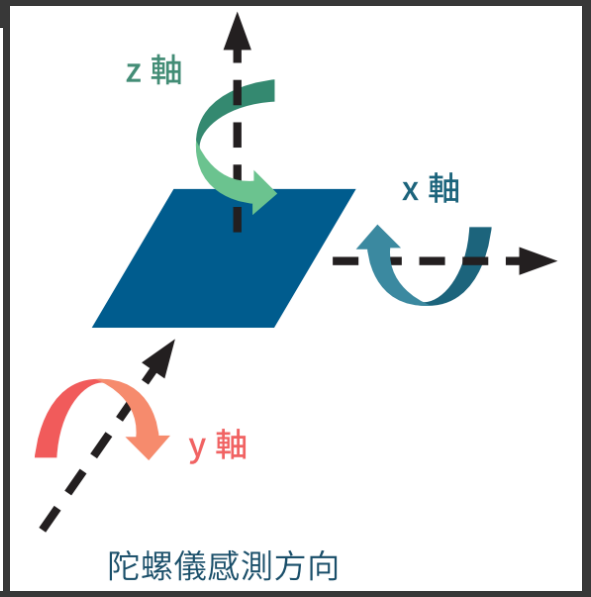
- 計步器藉由偵測震動判斷步數
- 智慧手環不僅偵測震動，內部感測器可偵測體感資訊並結合演算法，判斷此次震動是否跟走路時的震動相似。
- 取得體感資訊是首要條件。

6-2 六軸感測器

- 偵測人體動作的感測器統稱 **體感偵測器**，
  使用型號為 MPU6050 的『六軸感測器』，可感測到 6 軸：

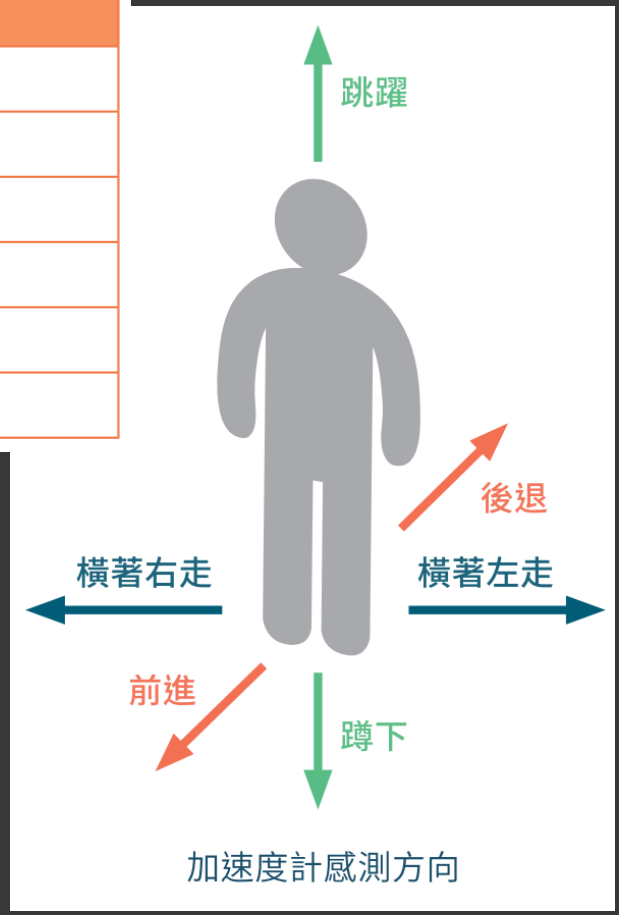


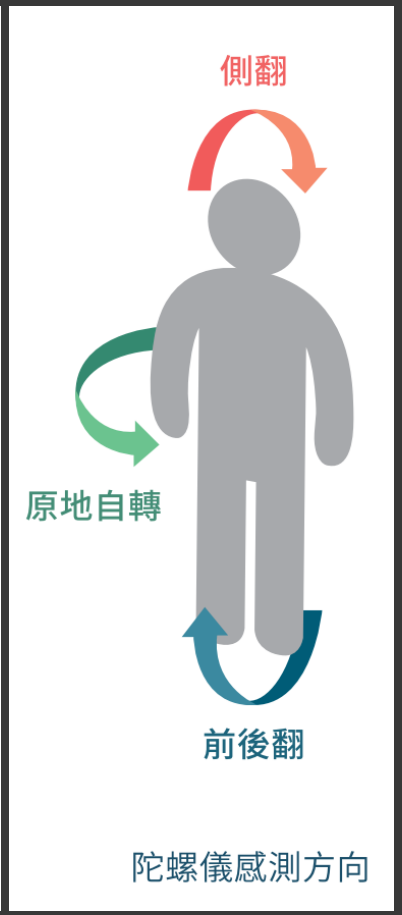




六軸感測器

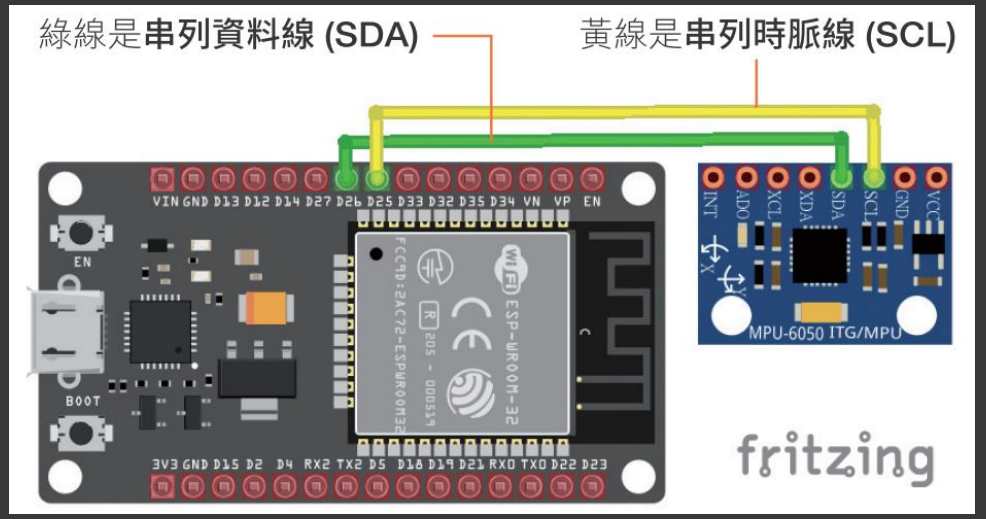
六軸感測器軸度	人體動作
加速度計 x 軸	橫著左右走
加速度計 y 軸	前進或後退
加速度計 z 軸	跳躍或蹲下
陀螺儀 x 軸	正面翻跟斗
陀螺儀 y 軸	側面翻跟斗
陀螺儀 z 軸	原地自轉





通訊方式

- 六軸感測器需透過 I²C 的通訊協定與 ESP32 開發板溝通。I²C 是控制周邊電子元件的通訊協定，通過串列時脈線 (SCL) 與串列資料線(SDA) 2 條線即可控制多個外部裝置：

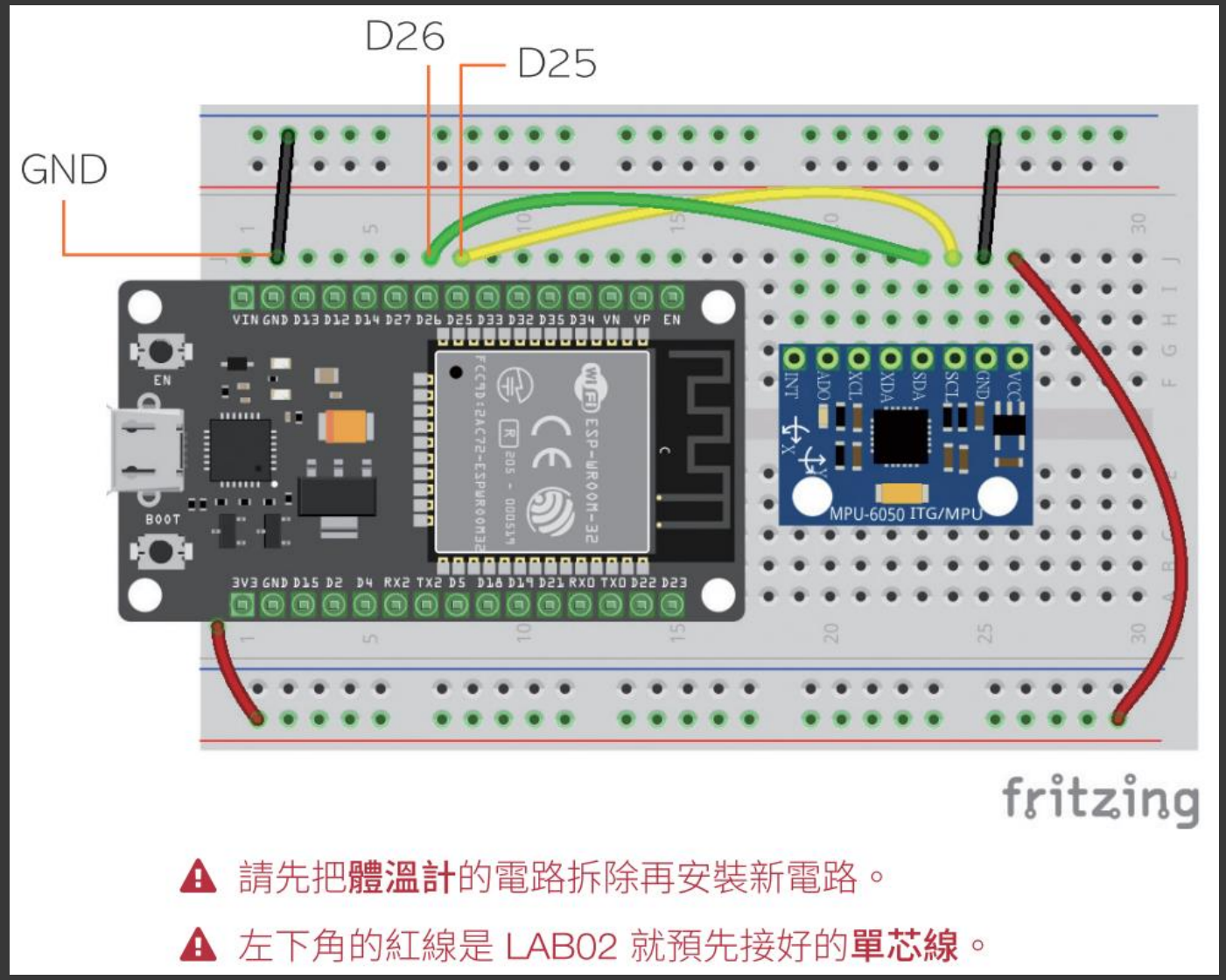


LAB08 顯示六軸感測器資訊

- 實驗目的：
  顯示六軸感測器的加速度值和陀螺儀值
- 材料： ESP32 、六軸感測器、單芯線 1 條、杜邦線若干條、麵包板
- 開發環境： Thonny

旗標創客

線路圖



⚠ 請先把**體溫計**的電路拆除再安裝新電路。

⚠ 左下角的紅線是 LAB02 就預先接好的**單芯線**。

旗標創客

實驗原理

ex5-1

Thonny

```
from machine import I2C, Pin
import mpu6050
```

Thonny

```
i2c = I2C(scl=Pin(25), sda=Pin(26))
accelerometer = mpu6050.accel(i2c)
```

Thonny

```
accelerometer.get_values()
```

旗標創客

實驗原理

ex5-1

Thonny

```
>>> accelerometer.get_values()
{'GyZ': 52, 'GyY': 180, 'GyX': -200, 'Tmp': 24.85941, 'AcZ':
18856, 'AcY': -712, 'AcX': 1180}
>>> accelerometer.get_values()['AcX']
1180
```

Thonny

```
>>> accelerometer.get_values()
{'GyZ': 0, 'GyY': 0, 'GyX': 0, 'Tmp': 36.53, 'AcZ': 0,
'AcY':0, 'AcX':0}
```

旗標創客

實驗原理

Thonny

```
while(accelerometer.get_values()['AcX']==0 and
      accelerometer.get_values()['AcY']==0 and
      accelerometer.get_values()['AcZ']==0 ):
      pass
```

旗標創客

程式設計

LAB08.py 顯示六軸感測器資訊 Thonny

```
1   from machine import I2C, Pin
2   import mpu6050
3
4   i2c = I2C(scl=Pin(25), sda=Pin(26))
5   accelerometer = mpu6050.accel(i2c)
6
7   while(accelerometer.get_values()['AcX']==0 and
8         accelerometer.get_values()['AcY']==0 and
9         accelerometer.get_values()['AcZ']==0 ):
10      pass
11
12  six_data = accelerometer.get_values()
13
14  print(six_data)
15  print(six_data['AcX'])
```

旗標創客

測試程式

六軸感測器的資訊

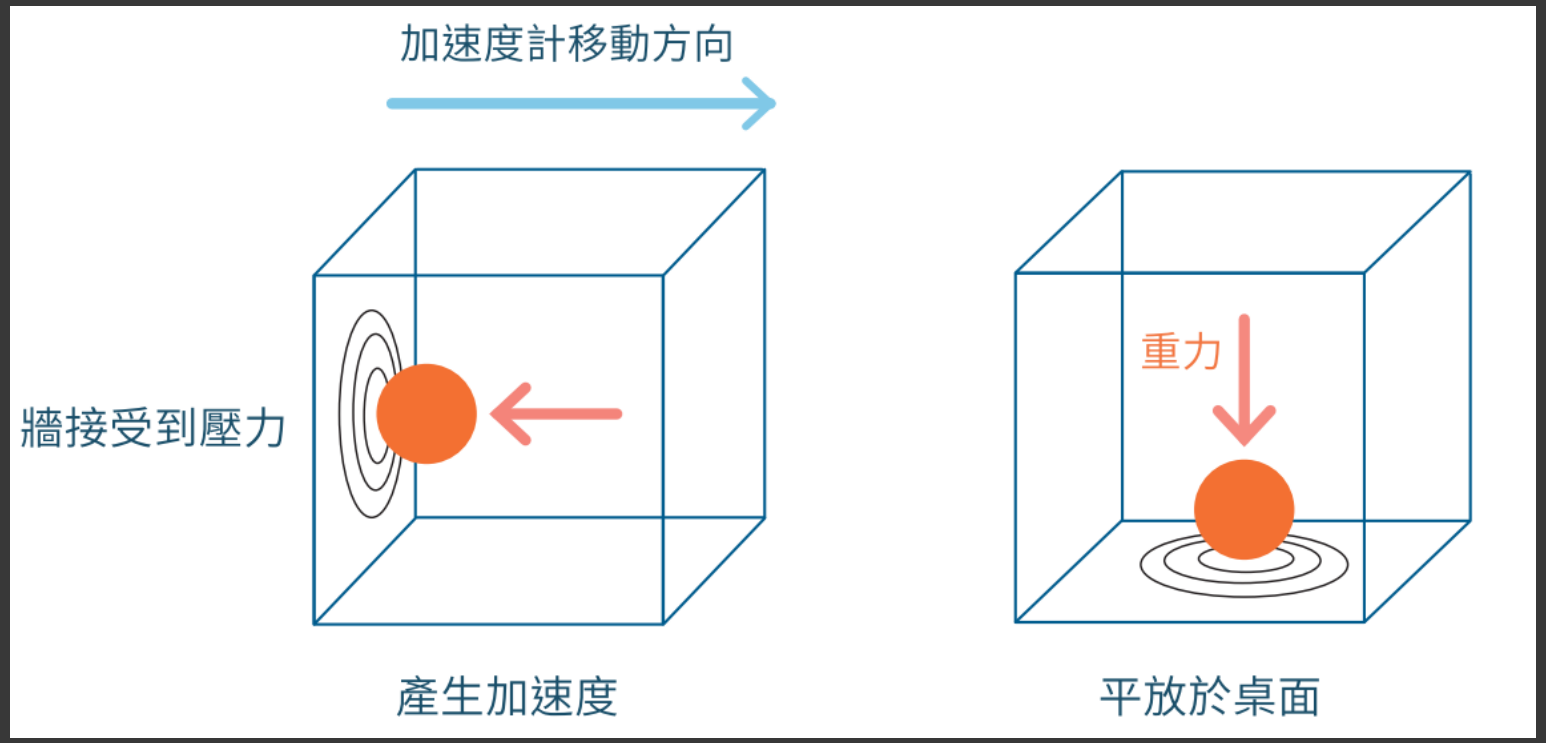
```
>>> %Run -c $EDITOR_CONTENT
{'GyZ': -92, 'GyY': -26, 'GyX': -178, 'Tmp': 26.88294,
'AcZ': 12852, 'AcY': -80, 'AcX': 1588}
1588
```

加速度計 x 軸的數值

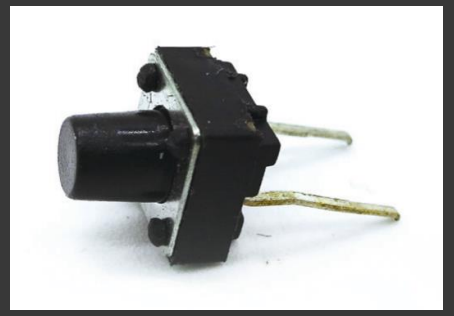
旗標創客

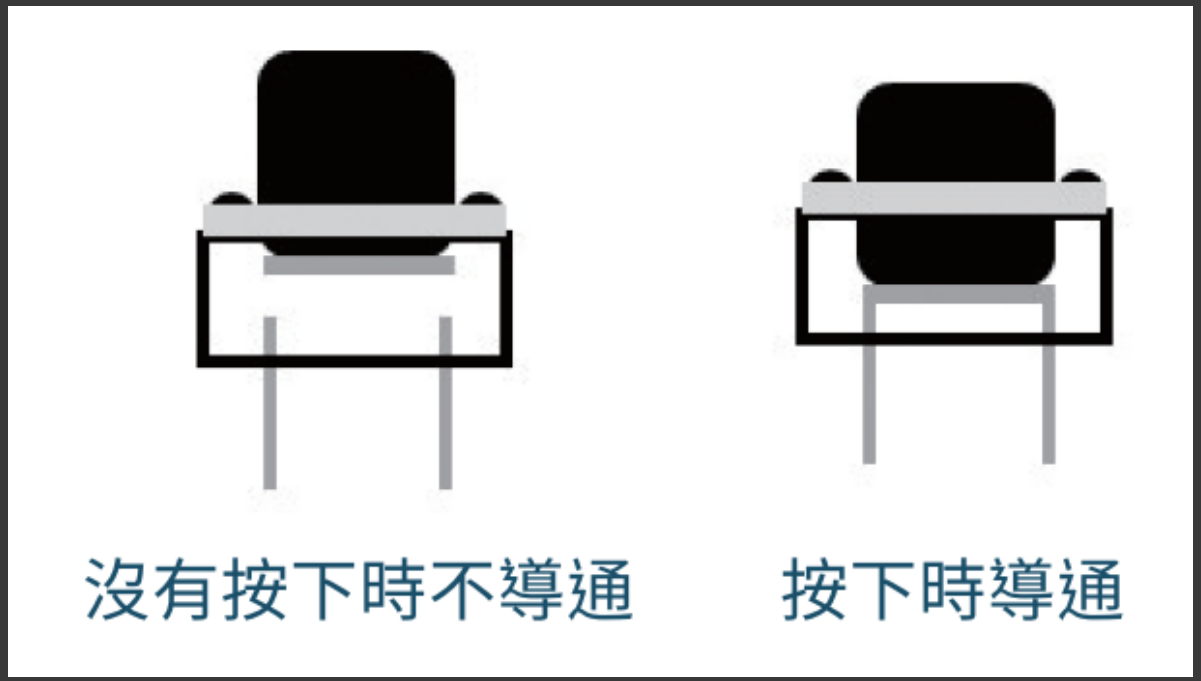
6-3 六軸感測器數據分析

- 將六軸感測器平放於桌上靜止不動, 六軸資訊卻都不是 0, 為什麼？



6-4 按鈕開關





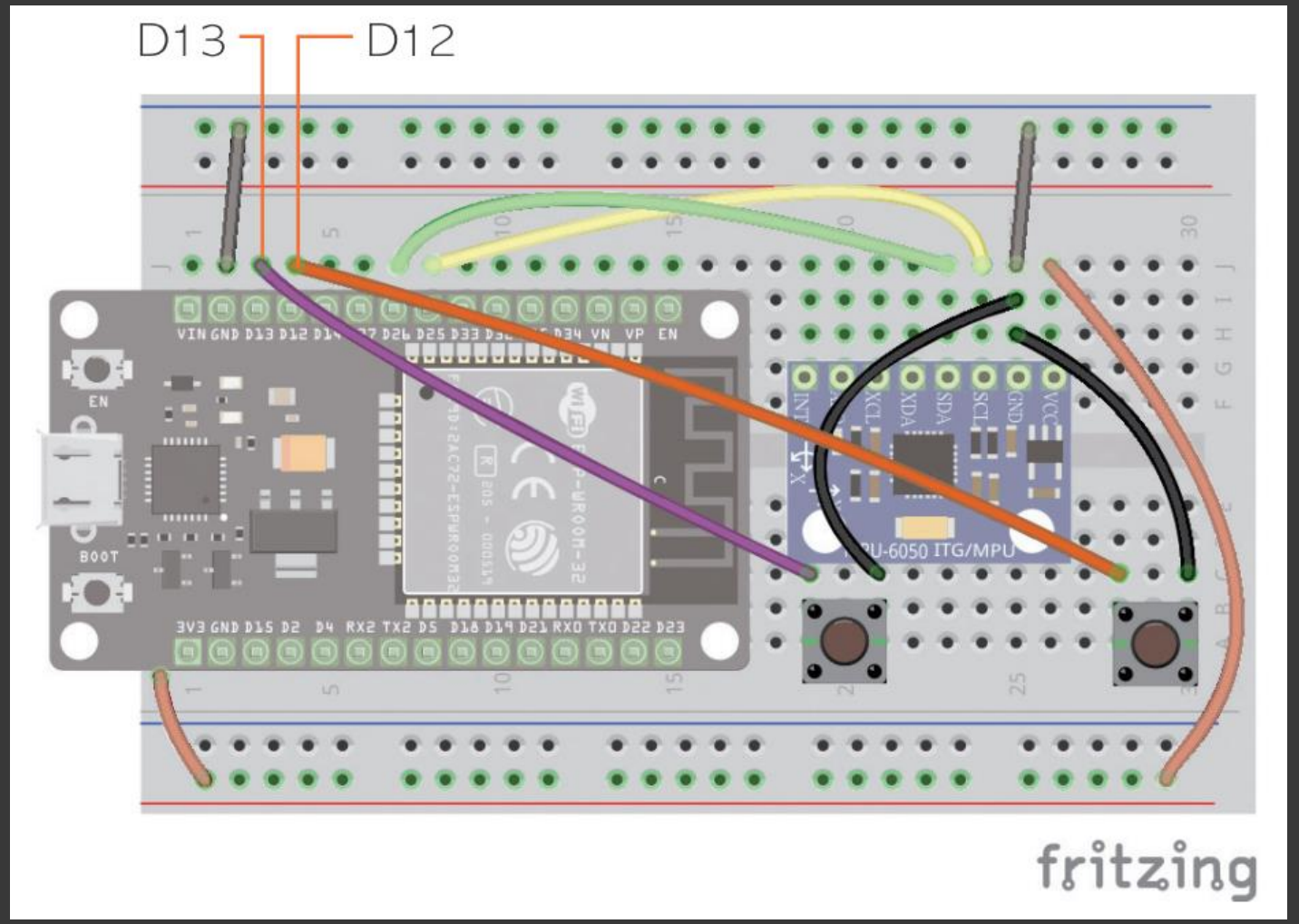
旗標創客

LAB09 按鈕開關測試

- 實驗目的：
  使用 ESP32 的輸入腳位讀取按鍵開關
- 材料：ESP32、按鈕 ×2、麵包板、杜邦線
- 開發環境： Thonny

旗標創客

線路圖



旗標創客

實驗原理

Thonny

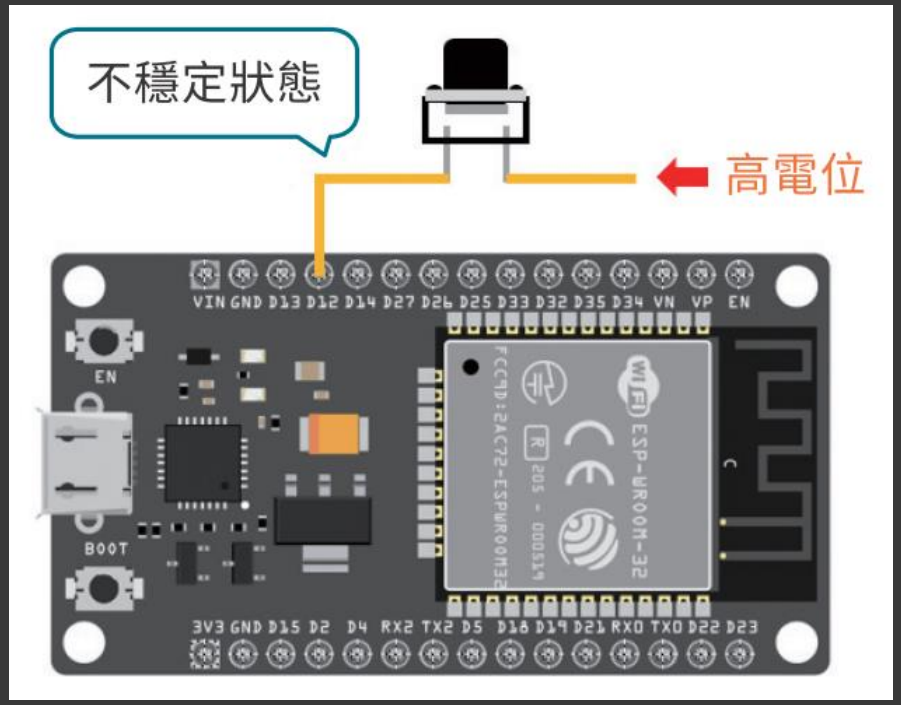
```
button=Pin(12, Pin.IN)
```

Thonny

```
button.value()
```

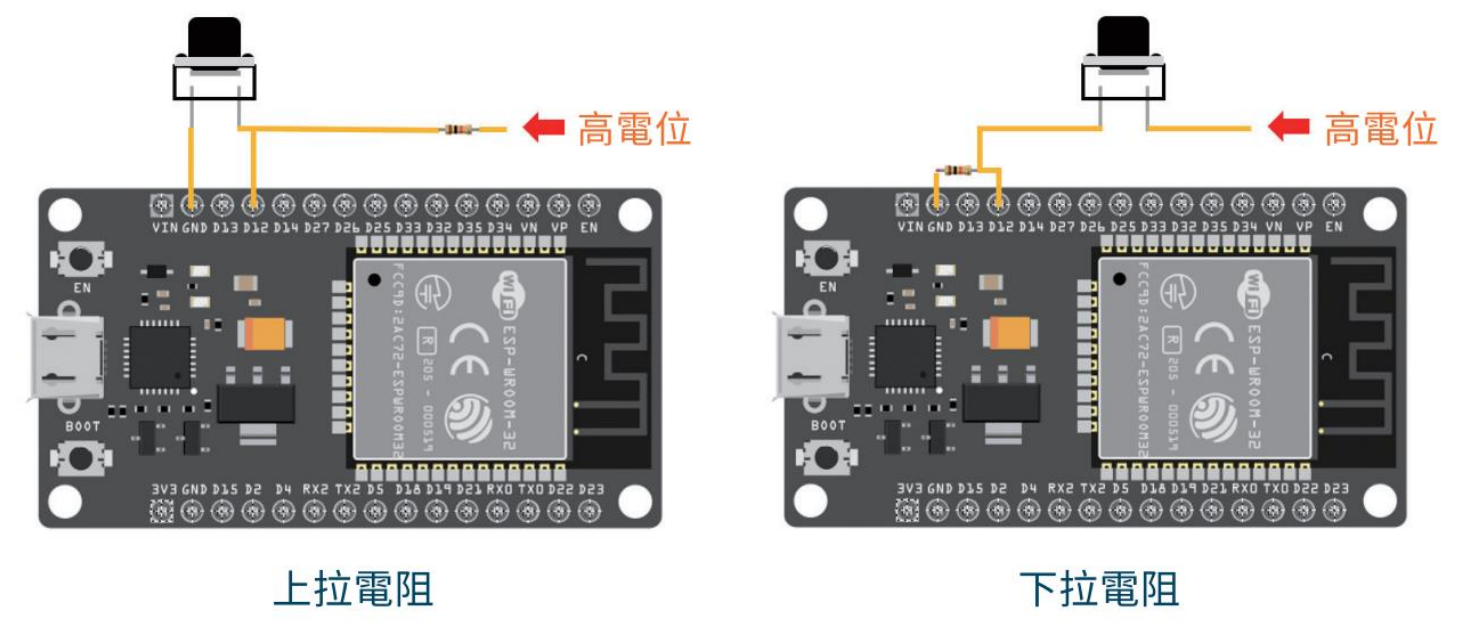
旗標創客

實驗原理



旗標創客

實驗原理



Thonny

```
button=Pin(12, Pin.IN, Pin.PULL_UP)
```

旗標創客

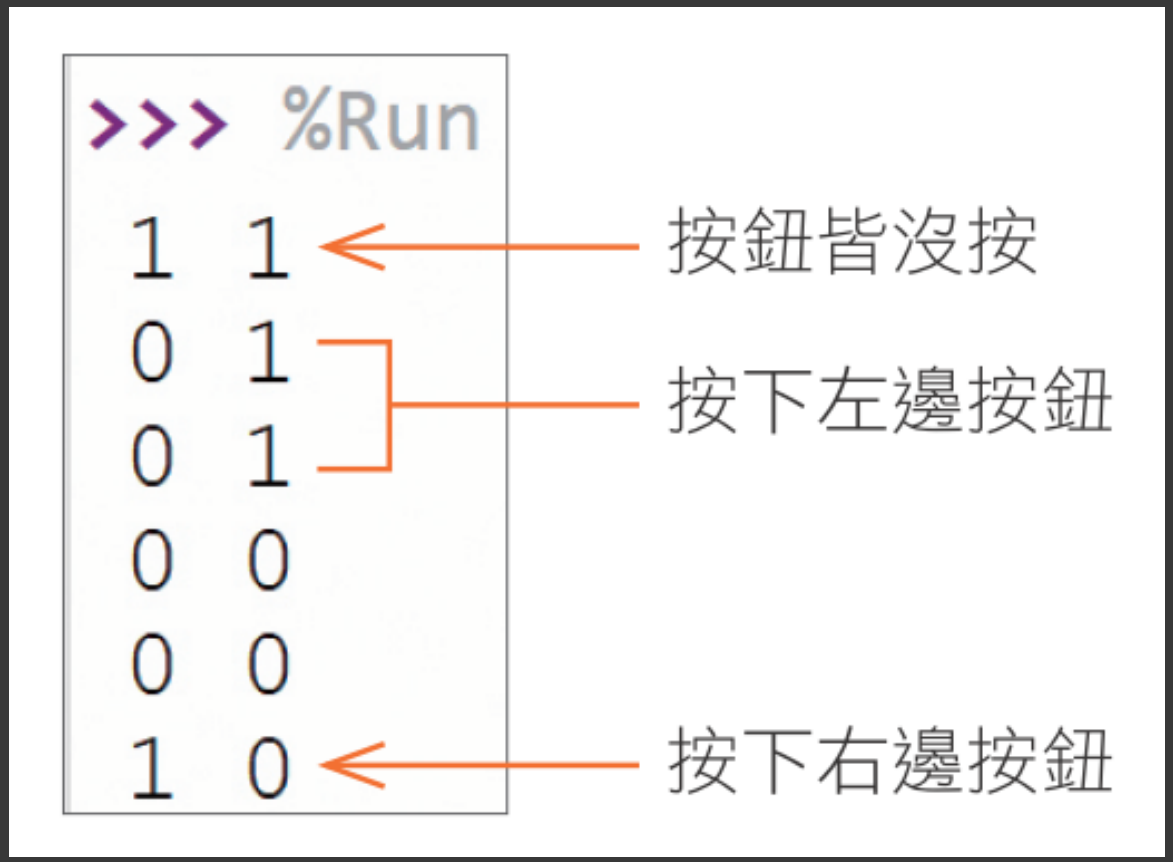
程式設計

LAB09.py 按鈕開關測試 Thonny

```
1   from machine import Pin
2   import time
3
4   button_yes=Pin(12, Pin.IN, Pin.PULL_UP)
5   button_no=Pin(13, Pin.IN, Pin.PULL_UP)
6
7   while True:
8       print(button_no.value(), button_yes.value())
9       time.sleep(0.1)
```

旗標創客

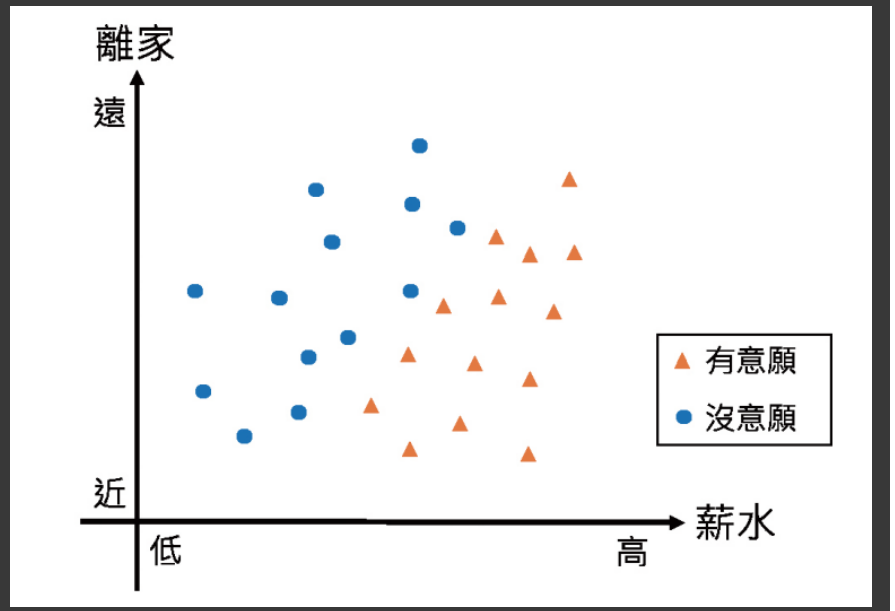
測試程式

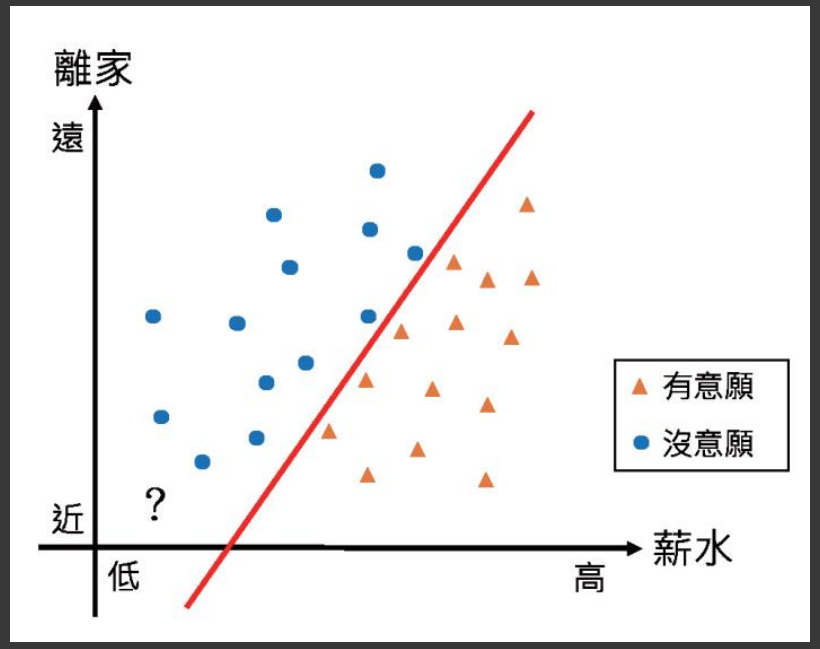


旗標創客

二元分類

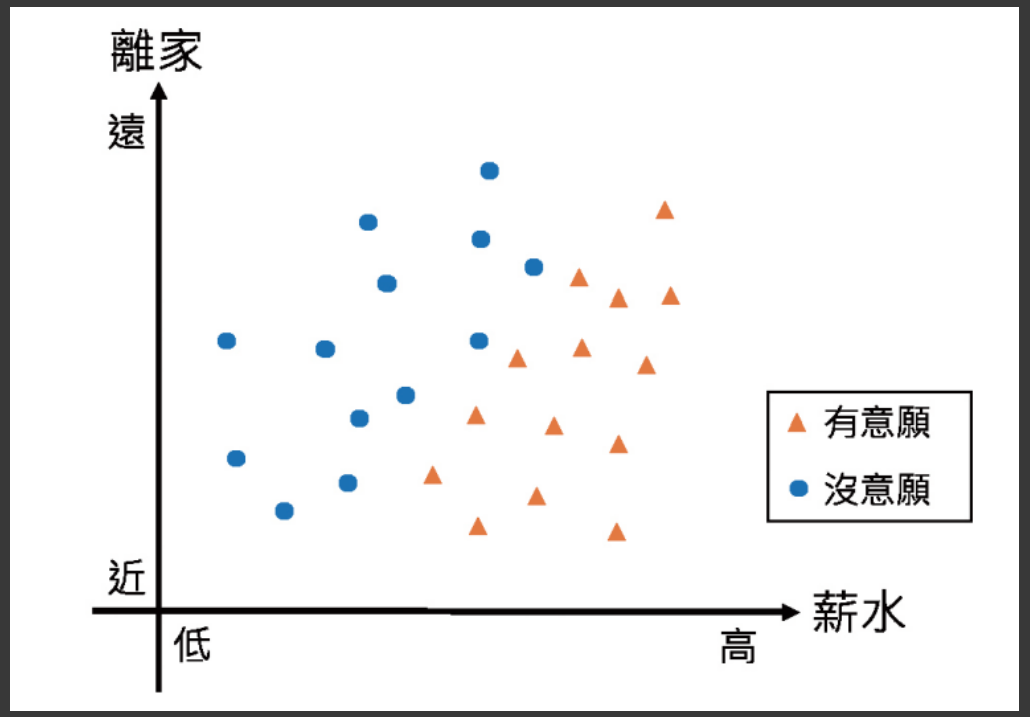
- 分類問題，依據選項的數量，可分為『二元分類』和『多元分類』。

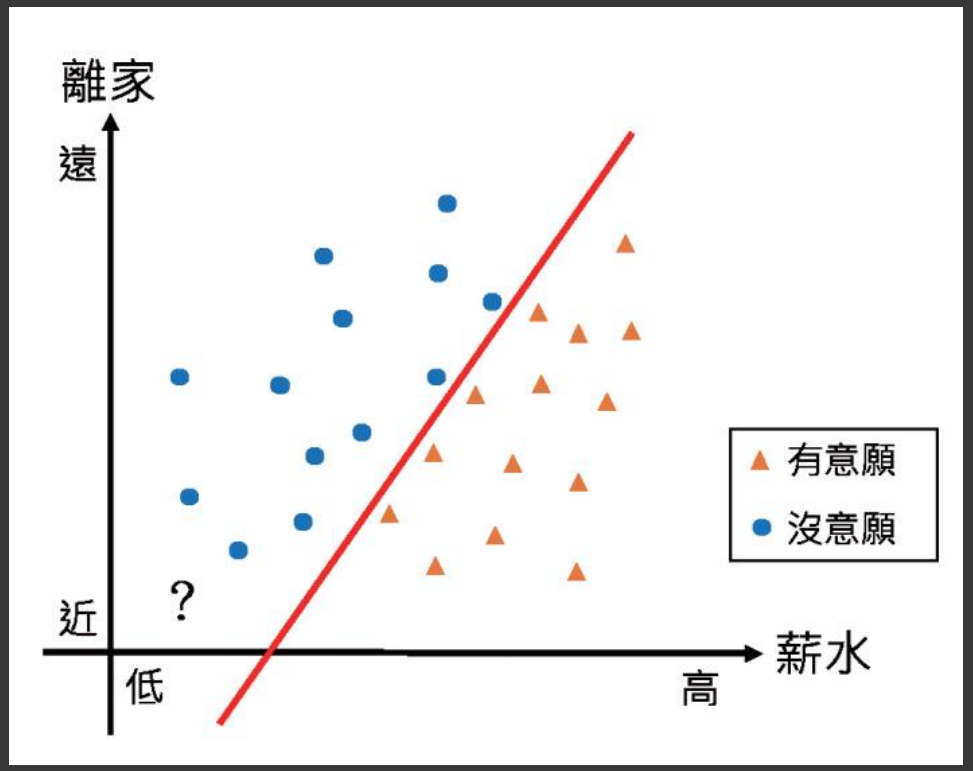




二元分類

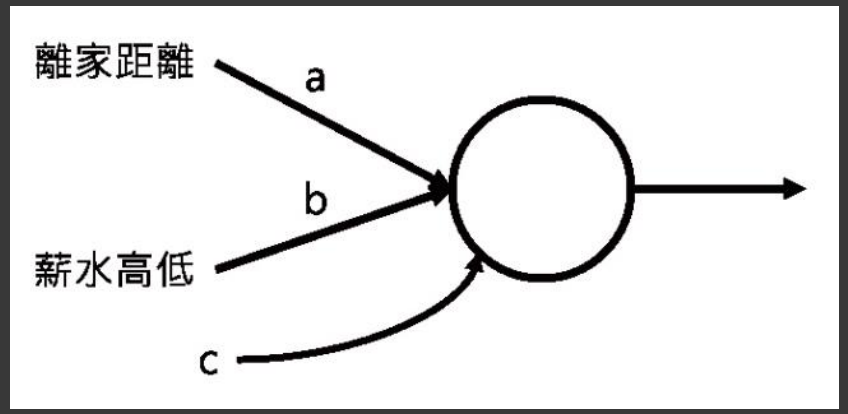
- 分類問題，依據選項的數量，可分為『二元分類』和『多元分類』。

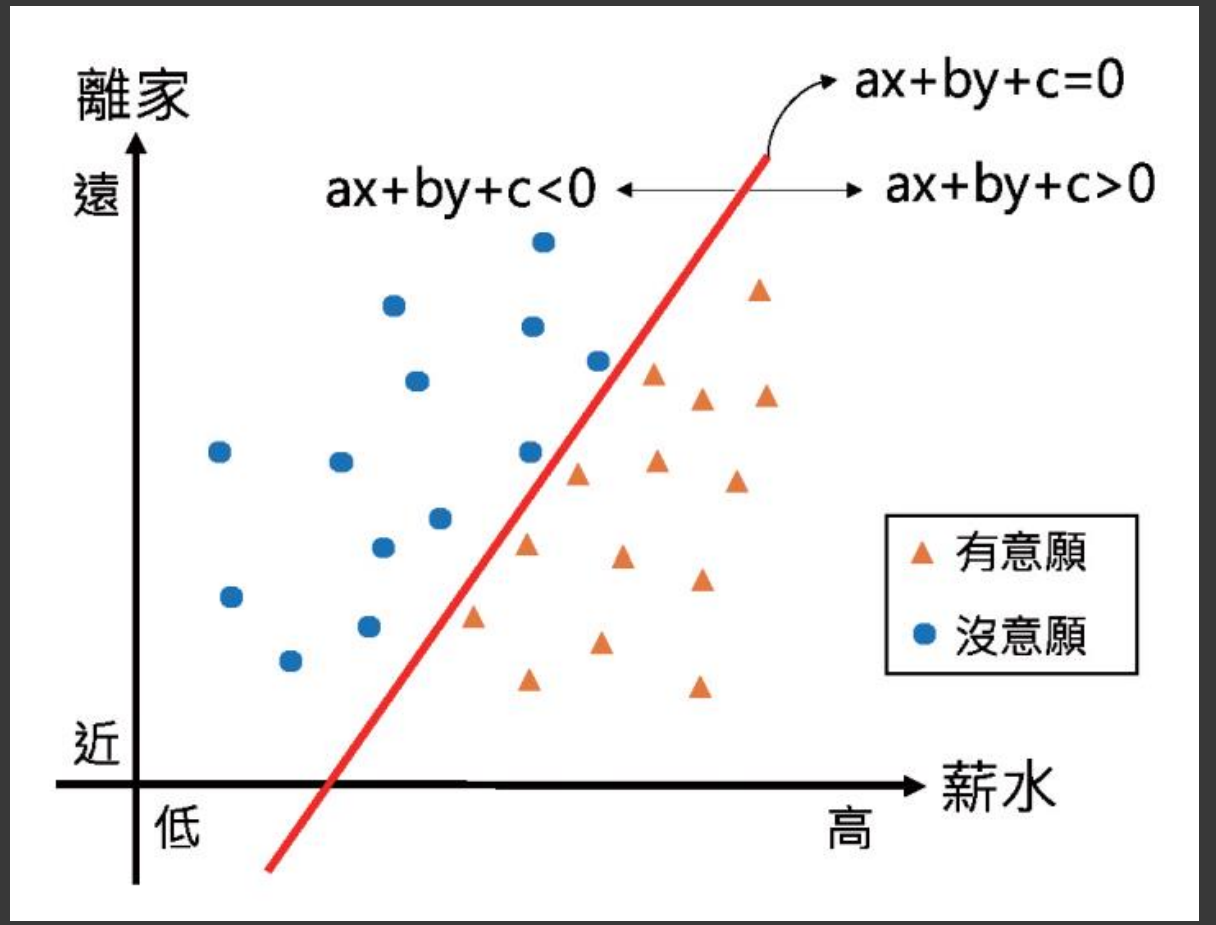




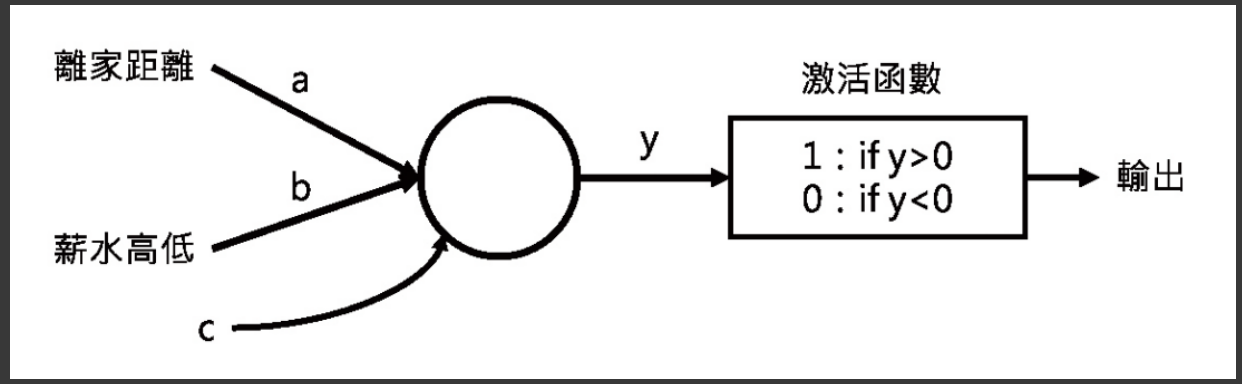
二元分類

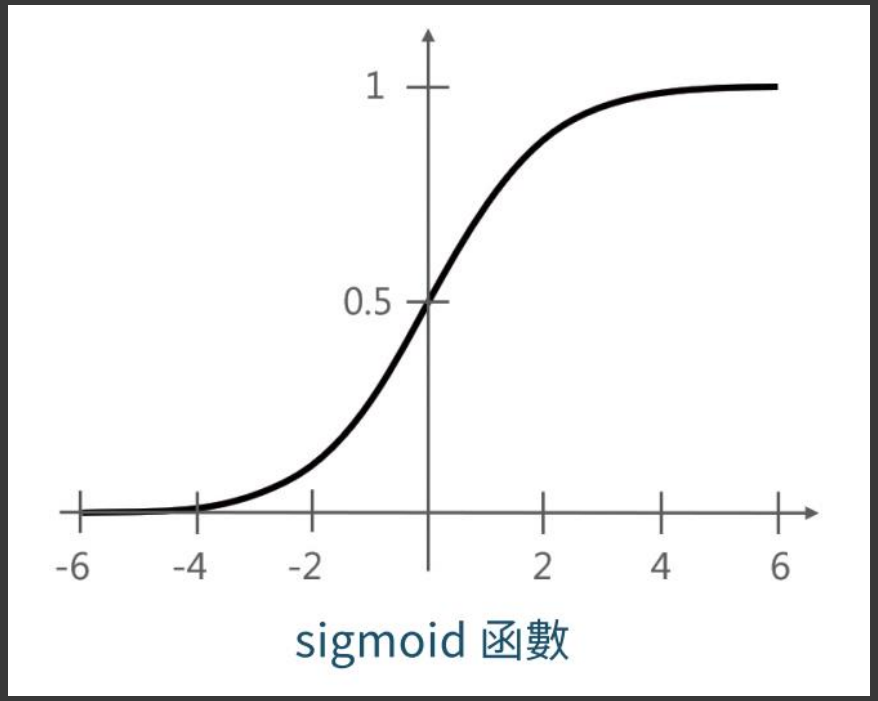
決策函數 =ax+by+c





輸出的激活函數





旗標創客

6-6 實作：步頻記錄儀

- 自製計步器主要有 2 個功能：

❶ 判斷為 " 走路 "、" 非走路 "

❷ 記錄 1 分鐘的步數

旗標創客

❶ 蒐集資料：量測及記錄『走路』、『非走路』的六軸資訊

- 蒐集走路和非走路的六軸資訊作為兩者特徵
- 經過測試發現在神經網路訓練過程中，每個類別各 100 筆資料不會花太長時間蒐集和訓練，更可訓練出分類準確率極高的模型。

LAB10 實作：步頻記錄儀 – 蒐集資料

- 實驗目的：
  蒐集 " 走路 " 和 " 非走路 " 各 100 筆六軸資訊供訓練模型使用
- 材料、線路圖：同 LAB09
- 開發環境： Thonny

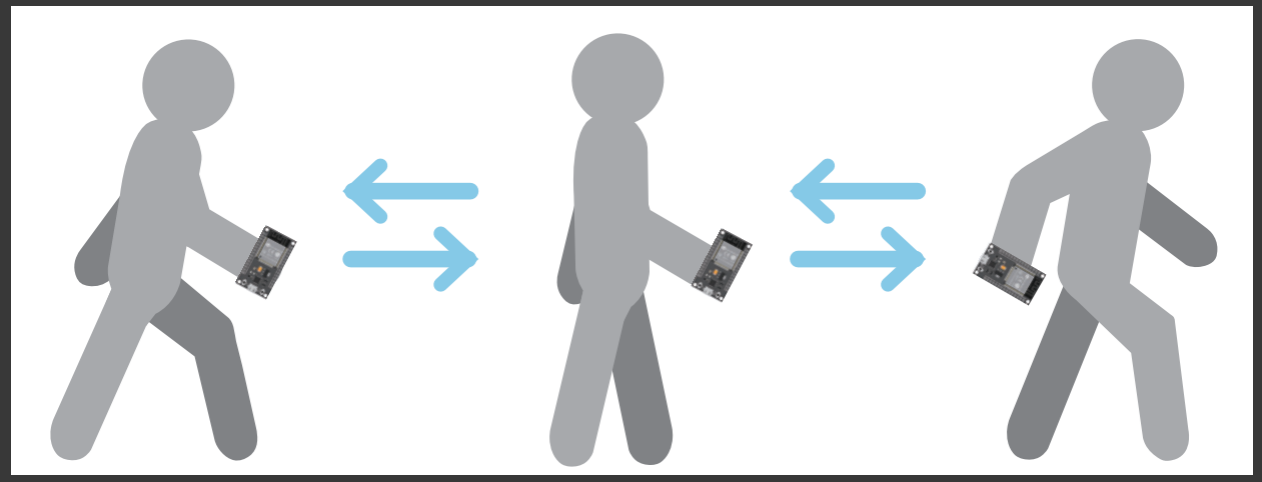
旗標創客

實驗原理

- 本實驗預設使用者以特定走路方式蒐集資料：

❶ 右手拿計步器

❷ 走路時手要快速的擺動



對動作有疑問者請掃描

實驗原理

- 為使用六軸數據表現甩手的連續動作，需要每 0.01 秒蒐集 1 次六軸數據，並在蒐集完 17 次後，自動形成一筆資料。
- 為避免蒐集『走路』資料時，不小心採集到『非走路』資料，使用 2 個按鈕開關決定是否儲存資料，一個代表儲存，一個代表捨棄。

程式設計

LAB10.py 實作：步頻記錄儀-蒐集資料 Thonny

```
01  from machine import I2C, Pin
02  import mpu6050
03  import time
04
05  button_yes=Pin(12,Pin.IN,Pin.PULL_UP)    # 儲存按鈕
06  button_no=Pin(13,Pin.IN,Pin.PULL_UP)     # 捨棄按鈕
07  i2c = I2C(scl=Pin(25),sda=Pin(26))
08  accelerometer = mpu6050.accel(i2c)
09  f=open('walk.txt','w')                   # 開啟txt檔
10
11  data=[]                 # 儲存資料的list
12  reset=False             # 是否開始偵測新的一筆資料
13
```

旗標創客

程式設計


```
14  # 等待值恢復正常
15  while(accelerometer.get_values()['AcX']==0 and
16        accelerometer.get_values()['AcY']==0 and
17        accelerometer.get_values()['AcZ']==0):
18      pass
19
20  # 儲存100筆
21  for j in range(100):
22
23      while True:
24          # 如果第一筆或上一筆資料剛結束時, 初始化六軸的值
25          if(reset==False):
26              time.sleep(0.3)                 # 暫停一下
27              print('')
28              print("第"+str(j+1)+"筆, 可以開始走動:")
29              data=[]
30              accelerometer = mpu6050.accel(i2c)
31              six_data = accelerometer.get_values()
```

旗標創客

程式設計


```
32
33              # 將x軸加速度的值存到『上次x軸加速度』
34              last_AcX=six_data['AcX']
35              reset=True
36
37          # 開始偵測值
38          six_data = accelerometer.get_values()
39          AcX=six_data['AcX']                 # x軸加速度
40
41          # 上次x軸加速度與這次x軸加速度相差大於3000
42          if abs(AcX-last_AcX)>3000:
43
44              reset=False
45              print('action')
46
47              for i in range(17):    # 收集17次6軸數值
48                  six_data = accelerometer.get_values()
49
```

旗標創客

程式設計

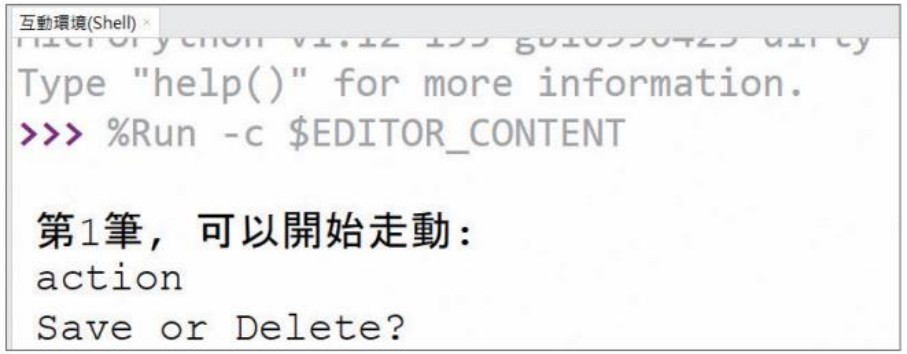

```
50                data.append(six_data['AcX']) # 加速度計 x 軸
51                data.append(six_data['AcY']) # 加速度計 y 軸
52                data.append(six_data['AcZ']) # 加速度計 z 軸
53                data.append(six_data['GyX']) # 陀螺儀 x 軸
54                data.append(six_data['GyY']) # 陀螺儀 y 軸
55                data.append(six_data['GyZ']) # 陀螺儀 z 軸
56                time.sleep(0.01)
57            print('Save or Delete?',end=' ')
58            # 2顆按鈕皆沒按
59            while(button_yes.value()==1 and
60                  button_no.value()==1):
61                time.sleep(0.01)
```

程式設計


```
62
63              if button_yes.value()==0:
64                  f.write(str(data)[1:-1])   # data存到檔案中
65                  f.write("\n")              # 換行字元
66                  print('Save')
67                  break                      # 跳脫while迴圈
68
69              elif button_no.value()==0:
70                  print('Delete')
71
72          # 將『這次的值』存到『前一次值』
73          last_AcX=AcX
74
75          time.sleep(0.01)    # 每次資料間隔0.01秒
76
77  f.close()                   # 關閉txt檔
```

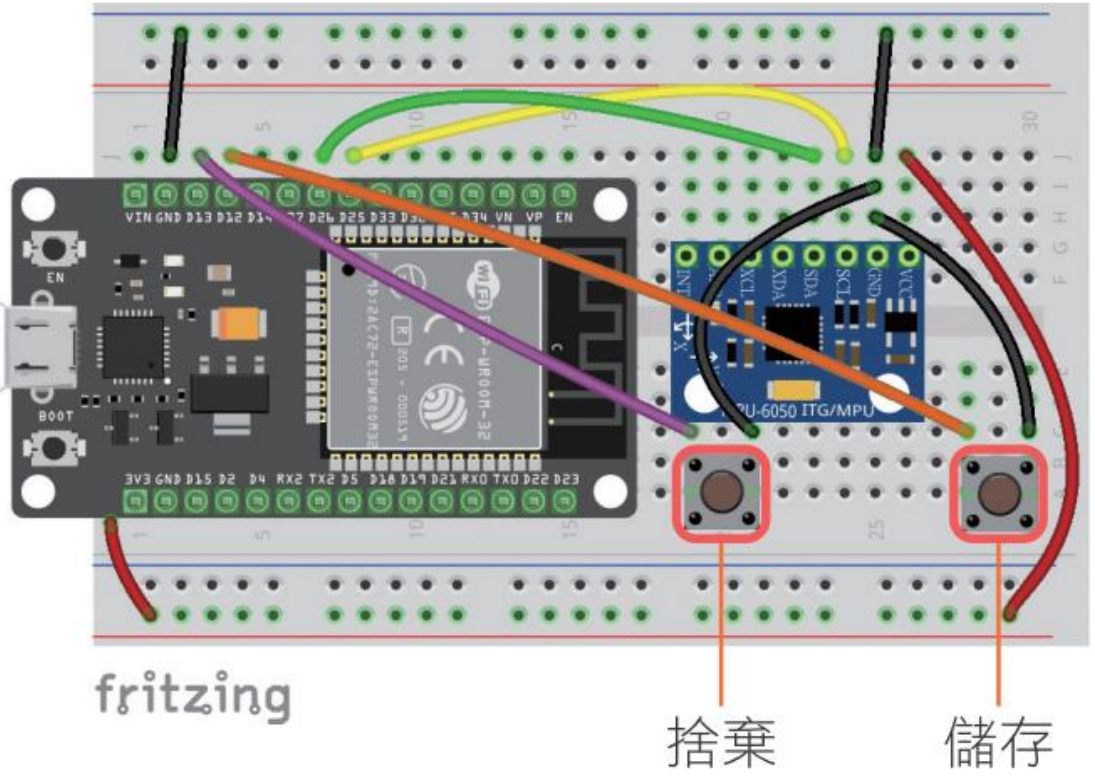
旗標創客

測試程式



旗標創客

測試程式

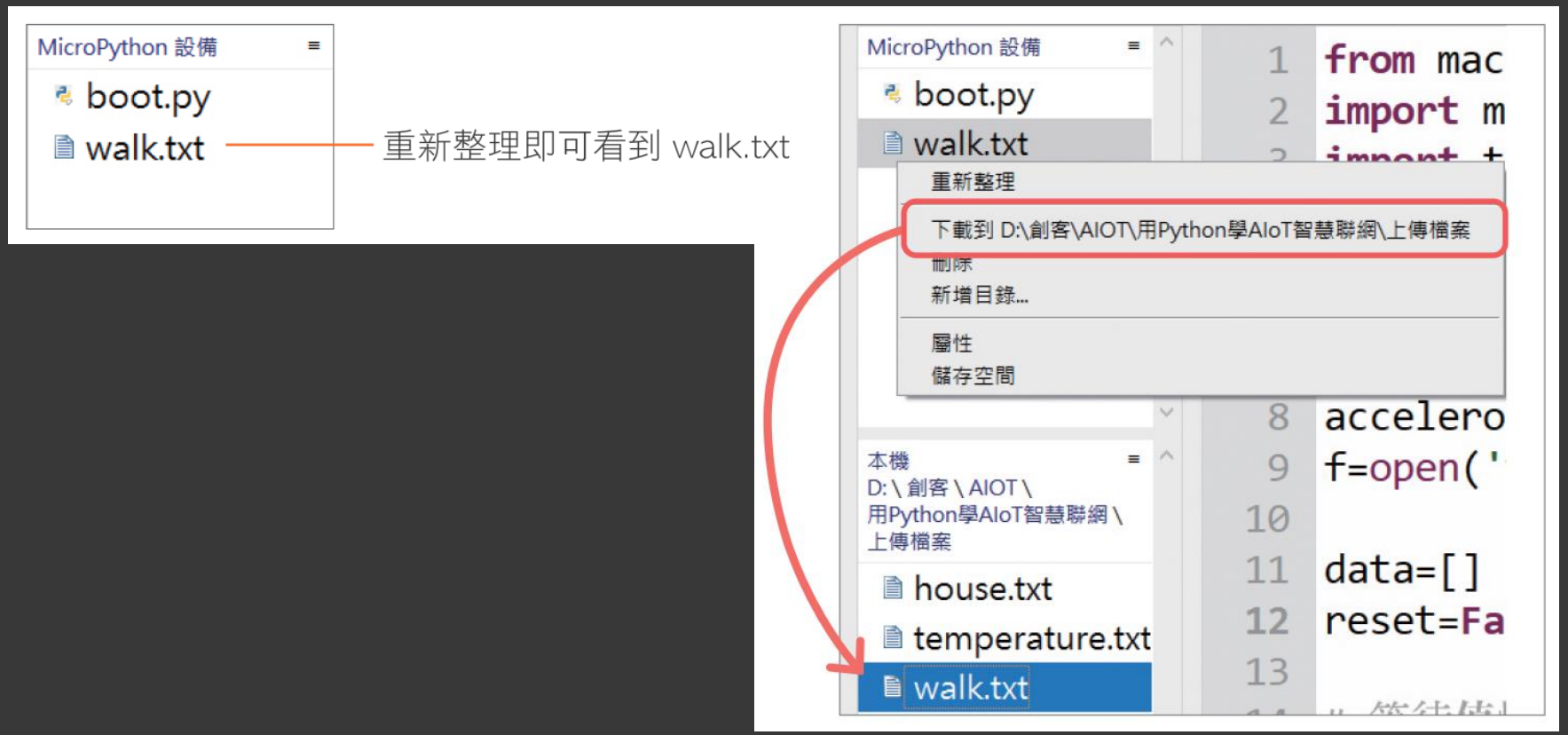


互動環境(Shell)

```
Type "help()" for more informati
>>> %Run -c $EDITOR_CONTENT

第1筆, 可以開始走動:
action
Save or Delete? Save ——儲存

第2筆, 可以開始走動:
action
Save or Delete? Delete —捨棄

第2筆, 可以開始走動:
```

旗標創客

測試程式



旗標創客

測試程式

- 完成 " 走路 " 樣本後 , 將程式碼第 9 行改成 others.txt, 並重新執行一次程式碼。

```
9 f=open('others.txt','w')
```

- 這時把麵包板平放並左右平移, 記錄左右擺動的資料 , 使 walk 和 others 差異性大一點。

測試程式

旗標創客

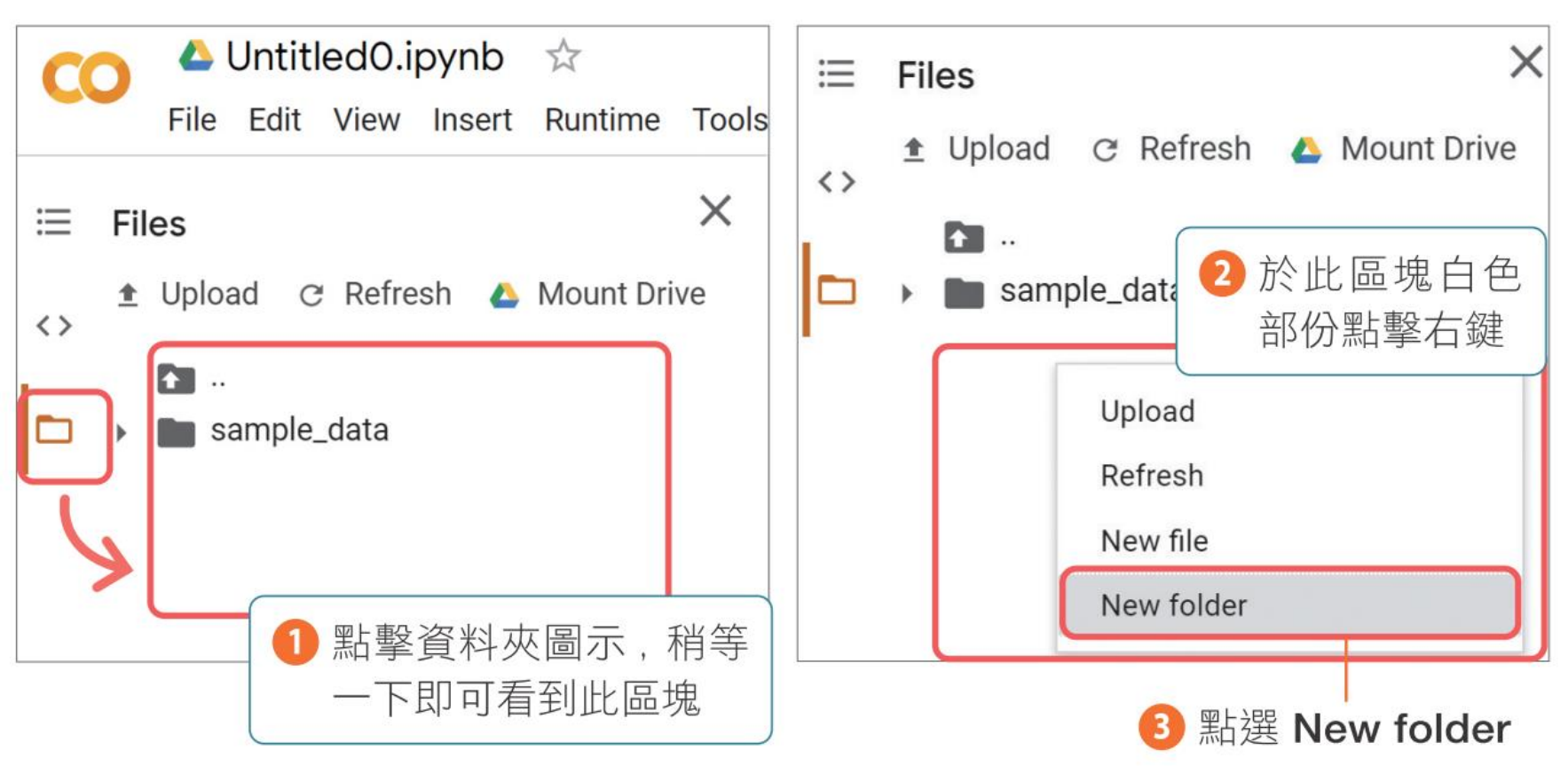
❷ 建立神經網路：二元分類

- 利用六軸資訊當作特徵；類別『走路』、『非走路』當作標籤，以此訓練出二元分類的模型。
- 開發環境：
  Colab, 需神經網路函式庫及大量運算能力。

❷ 建立神經網路：二元分類

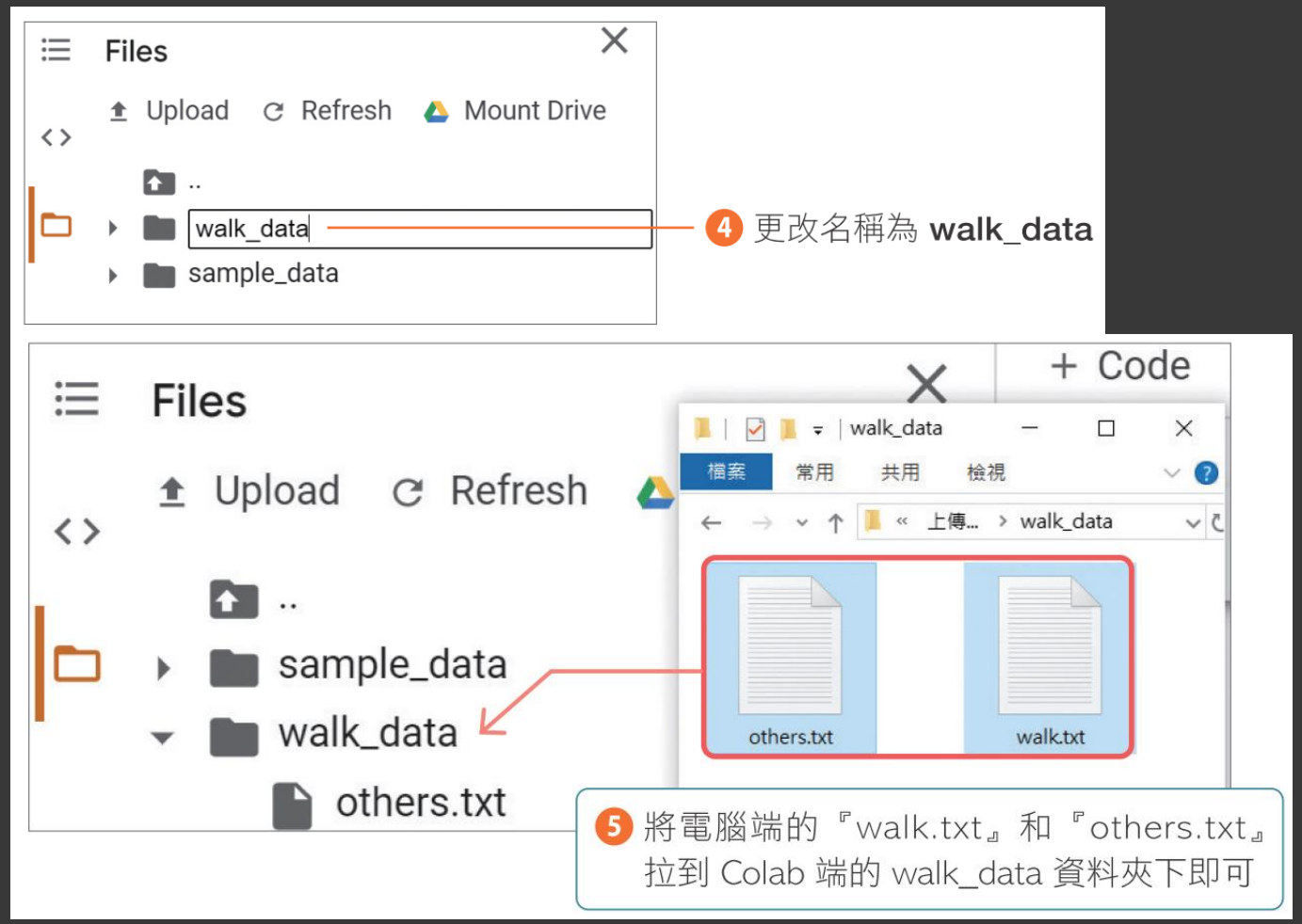
- 開發環境：
  Colab, 需神經網路函式庫及大量運算能力。

讀取檔案



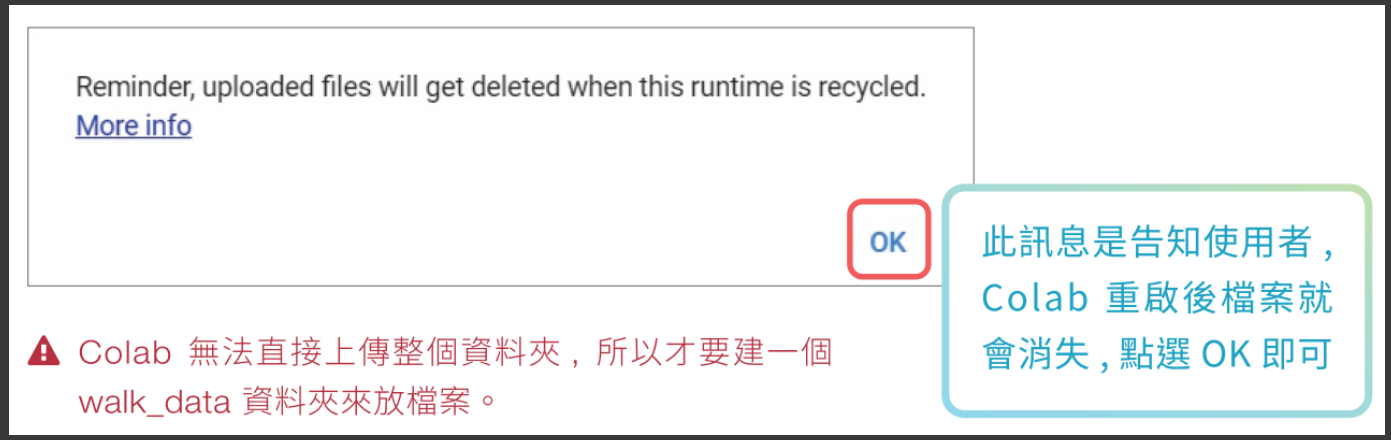
旗標創客

讀取檔案



旗標創客

讀取檔案



walk_model.ipynb 讀取 walk_data 資料夾內的檔案 **Colab**

```
import keras_lite_convertor as kc
path_name = 'walk_data'
Data_reader = kc.Data_reader(path_name,
                             mode='binary',
                             label_name = ['others', 'walk'])
data, label = Data_reader.read()
```

旗標創客

資料預處理

walk_model.ipynb (續) 資料預處理 Colab

```
# 取資料中的 90% 當作訓練集
split_num = int(len(data)*0.9)
train_data=data[:split_num]
train_label=label[:split_num]

# 正規化
mean = train_data.mean() # 平均數
data -= mean
std = train_data.std()    # 標準差
data /= std
```

walk_model.ipynb (續) 資料分割-驗證集、測試集 Colab

```
# 驗證集
validation_data=data[split_num:-5]
validation_label=label[split_num:-5]

# 測試集
test_data=data[-5:]
test_label=label[-5:]
```

旗標創客

建立神經網路架構

walk_model.ipynb (續) 建立神經網路架構 Colab

```
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
model = Sequential()
model.add(layers.Dense(10,activation = 'relu',
                       input_shape=(102,)))
model.add(layers.Dense(1, activation = 'sigmoid'))
model.summary()
```

旗標創客

建立神經網路架構


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 10)                1030
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 1, 041
Trainable params: 1, 041
Non-trainable params: 0
_________________________________________________________________
```

旗標創客

編譯及訓練模型

walk_model.ipynb (續) 編譯及訓練模型 Colab

```
model.compile(optimizer='adam',loss='binary_crossentropy',
              metrics=['acc'])
train_history = model.fit(train_data,train_label,
          validation_data=(validation_data,validation_label),
          epochs=300)
Epoch 1/300
6/6 ... - loss: 0.6488 - acc: 0.5444 ...
Epoch 2/300
6/6 ... - loss: 0.5885 - acc: 0.6278 ...
Epoch 3/300
6/6 ... - loss: 0.5436 - acc: 0.6556 ...
...
Epoch 298/300
6/6 ... - loss: 9.7673e-04 - acc: 1.0000 ...
Epoch 299/300
6/6 ... - loss: 9.6867e-04 - acc: 1.0000 ...
Epoch 300/300
6/6 ... - loss: 9.6220e-04 - acc: 1.0000 ...
```

旗標創客

測試模型

walk_model.ipynb (續) 測試模型 Colab

```
print('predict:')
print(model.predict(test_data))
print()
print('real:')
print(test_label)
```

```
predict:
[[9.9988079e-01]
 [2.5223169e-04]
 [9.9951291e-01]
 [9.9997926e-01]
 [9.9995196e-01]]

real:
[1 0 1 1 1]
```

第 1 筆測試集的預測值

最後 1 筆測試集的預測值

第 1 筆測試集的實際值

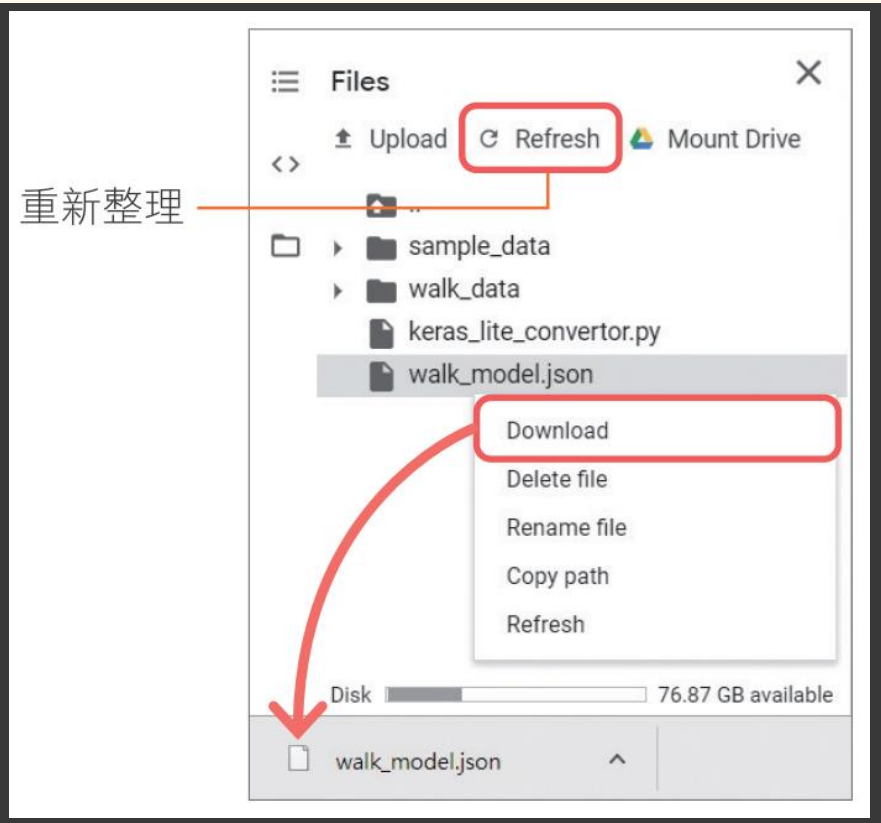
最後 1 筆測試集的實際值

旗標創客

儲存模型

walk_model.ipynb (續) 儲存模型 Colab

```
kc.save(model, 'walk_model.json')
```



旗標創客

顯示正規化相關資訊

walk_model.ipynb (續) 顯示資訊 Colab

```
print('mean=', mean)
print('std=', std)
```

```
mean= 849.1830065359477
std= 16766.660464036682
```

旗標創客

❸ 使用訓練好的模型進行分類，並顯示每分鐘的步數

- 預測分類和蒐集資料都是用 **加速度計 x 軸的變化**當作震動，但此時是代入模型計算
- 程式碼部分只需增加『神經網路參數』、『預測分類』和『顯示步頻』3 大部分即可

旗標創客

LAB11 步頻記錄儀

- 實驗目的：
  將六軸數據帶到模型中進行分類，並顯示每分鐘的步數
- 材料、線路圖：同 LAB09
- 開發環境： Thonny

旗標創客

實驗原理

Thonny

```
status_label = model.predict_classes(data)
```

Thonny

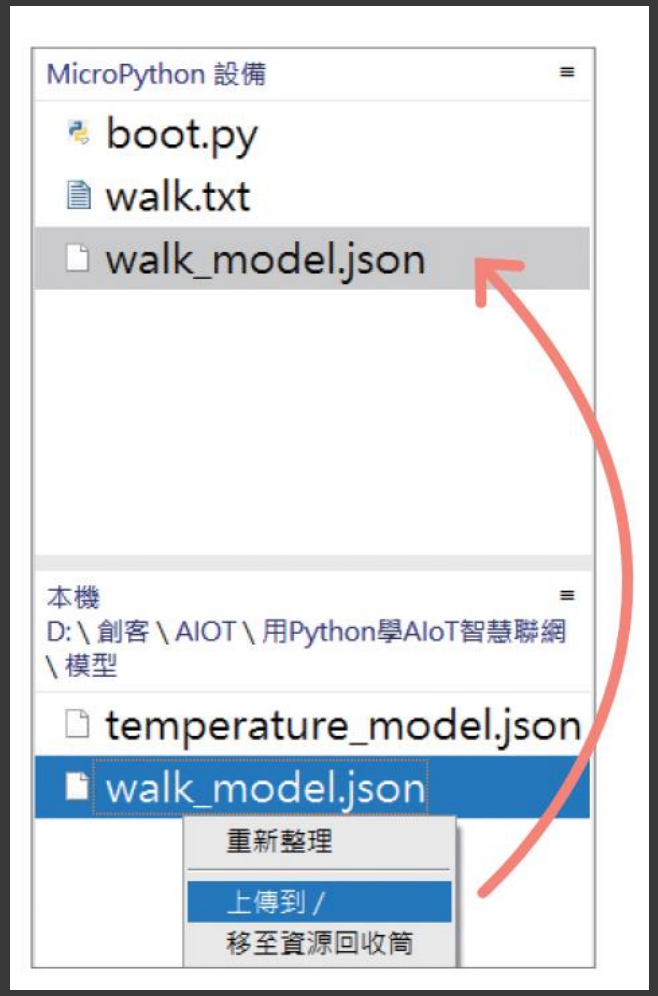
```
label_name = ['others', 'walk']   # label 名稱
label_name[status_label[0]]       # 回傳值為"others"或"walk"
```

旗標創客

實驗流程

1 上傳 walk_model.json 到 ESP32

2 複製 Colab 中的平均值和標準差 。



旗標創客

程式設計

LAB11.py 步頻記錄儀 Thonny

```
01  from machine import I2C, Pin
02  import mpu6050
03  import time
04  from keras_lite import Model
05  import ulab as np
06
07  #增加神經網路的參數
08  mean = 849.1830065359477
09  std = 16766.660464036682
10  model = Model('walk_model.json')
11  label_name = ['others', 'walk']
12
```

旗標創客

程式設計


```
13  i2c = I2C(scl=Pin(25), sda=Pin(26))
14  accelerometer = mpu6050.accel(i2c)
15  data=[]
16  reset=False
17
18  while(accelerometer.get_values()['AcX']==0 and
19        accelerometer.get_values()['AcY']==0 and
20        accelerometer.get_values()['AcZ']==0):
21      pass
22
23  step_count=0              # 總步數
24  last_time=time.time()     # 記錄上次時間
25
26  while True:
27
```

旗標創客

程式設計


```
28      if(reset==False):
29          time.sleep(0.3)
30          data=[]
31          accelerometer = mpu6050.accel(i2c)
32          six_data = accelerometer.get_values()
33          print('start:')
34          last_AcX=six_data['AcX']
35          reset=True
36
37      six_data = accelerometer.get_values()
38      AcX=six_data['AcX']
39
40      if(abs(AcX-last_AcX)>3000):
41
42          reset=False  # 重置 reset
43
44          for i in range(17):
45              six_data = accelerometer.get_values()
```

旗標創客

程式設計


```
46
47              data.append(six_data['AcX'])
48              data.append(six_data['AcY'])
49              data.append(six_data['AcZ'])
50              data.append(six_data['GyX'])
51              data.append(six_data['GyY'])
52              data.append(six_data['GyZ'])
53              time.sleep(0.01)
54          data = np.array([data])
55          data = data-mean
56          data = data/std
57          status_label = model.predict_classes(data)
58          status = label_name[status_label[0]]
59
60          print('status:', status)
61          print('')
```

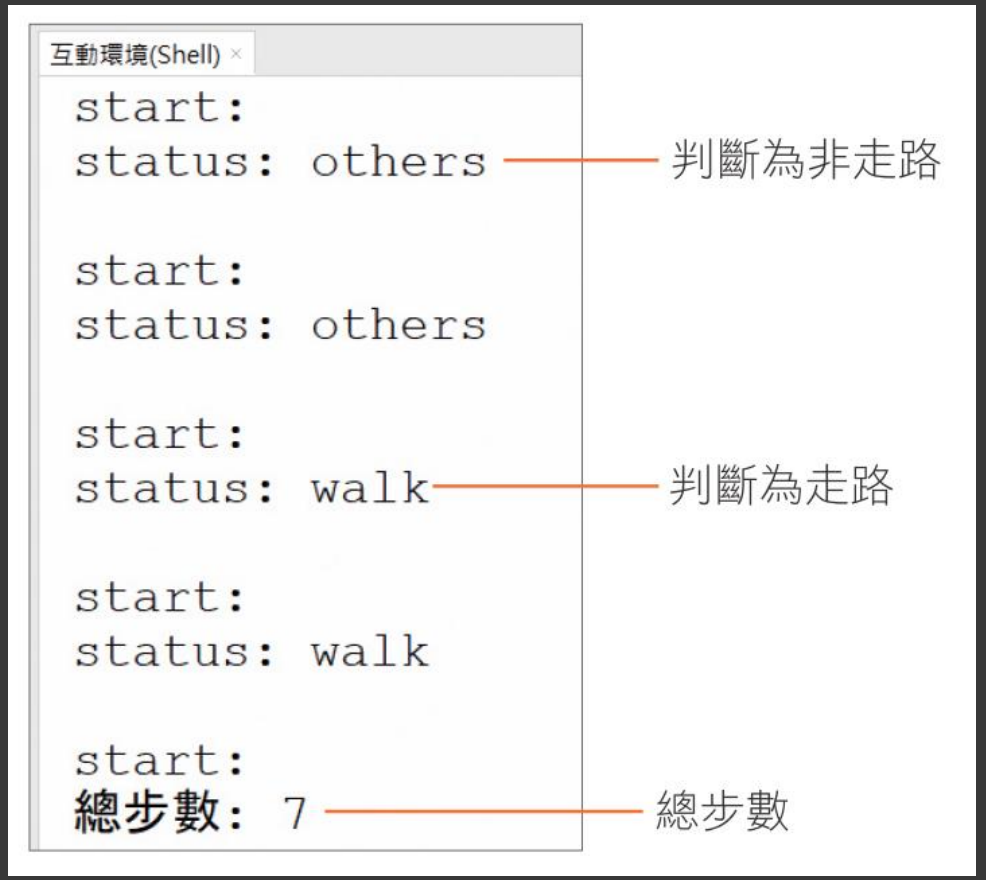
旗標創客

程式設計


```
62
63          if(status=='walk'):
64              step_count += 1      # 步數加 1
65
66      last_AcX=AcX
67
68      time.sleep(0.01)
69
70      if(time.time()-last_time>=60):      # 每 1 分鐘顯示 1 次
71          print("每分鐘步數:", step_count) # 顯示步數
72          print('')
73          step_count=0
74          last_time=time.time()
```

旗標創客

測試程式



旗標創客

6-7 IoT 應用 – 雲端步頻記錄儀

- 把計步器的資料再次上傳到 中華電信 IoT 平台，不只方便查看，更可以長期記錄自己的步數來觀察自己的運動狀態。
- 中華電信平台除了 POST 方法，還有提供一種上傳方式 MQTT 通訊。

MQTT 通訊協定簡介

- MQTT 由 3 個元件所組成：

❶ MQTT 中介伺服器 (broker)

❷ 發佈端 (publisher)

❸ 訂閱端 (subscriber)

旗標創客

MQTT 通訊協定簡介

- 發佈端和訂閱端都要連接至 MQTT 中介伺服器，兩者統稱為 MQTT 用戶端(client)。

MQTT 中介伺服器

主機網址	iot.cht.com.tw
連接埠編號	1883
使用者帳號	金鑰 (CK)
密碼	金鑰 (CK)

旗標創客

umqtt 模組

Thonny

```
from umqtt.robust import MQTTClient
```

Thonny

```
client = MQTTClient(client_id="step",       # 用戶端識別名稱
                    server=iot.cht.com.tw,  # 中介伺服器網址
                    user=金鑰,              # 帳戶名稱
                    password=金鑰)          # 金鑰
```

Thonny

```
client.connect()          # 連上伺服器
payload=[{"id":"step", "value":str(step_count)}]
client.publish(b" /v1/device/設備編號/rawdata ",
str(payload).encode())   # 資料內容
```

旗標創客

LAB12 雲端步頻記錄儀

- 實驗目的：
  利用 MQTT 將步頻上傳至雲端平台
- 材料、線路圖：同 LAB09
- 開發環境： Thonny
- 實驗原理：延續 LAB11, 將每一分鐘顯示的步頻利用 MQTT 上傳至中華電信 IoT 平台。

建立步頻記錄儀

- 再次進入中華電信 IoT 平台的『專案管理』，並點選 + 增加專案 ：

建立步頻記錄儀

旗標創客

建立步頻記錄儀

旗標創客

建立步頻記錄儀

旗標創客

建立步頻記錄儀

旗標創客

建立步頻記錄儀

旗標創客

建立步頻記錄儀

- 查看設備編號 (device_id)、金鑰 (CK)和感測器編號 (id)：

旗標創客

程式設計

LAB12.py 雲端步頻記錄儀 Thonny

```
06  import network
07  from umqtt.robust import MQTTClient
08
09  # 連線至無線網路
10  LED=Pin(2,Pin.OUT,value=0)     # 連上WiFi前關閉內鍵led燈
11  sta=network.WLAN(network.STA_IF)
12  sta.active(True)
13  sta.connect('無線網路名稱','無線網路密碼')
14  while not sta.isconnected() :
15      pass
16  LED.value(1)                   # 連上WiFi時亮燈
17
```

旗標創客

程式設計


```
18  # mqtt參數
19  mqtt_client_id = 'step'                    # 用戶端識別名稱
20  CHT_URL = 'iot.cht.com.tw'                 # 主機網址
21  CHT_USERNAME = '中華電信IoT平台的金鑰'      # 帳戶名稱
22  CHT_IO_KEY = '中華電信IoT平台的金鑰'        # 金鑰
23
24  client = MQTTClient(client_id=mqtt_client_id,
25                      server=CHT_URL,
26                      user=CHT_USERNAME,
27                      password=CHT_IO_KEY)
28
29  client.connect()                           # 連線至mqtt伺服器
...
96      if(time.time()-last_time>=60):         # 每1分鐘顯示1次
97          print("總步數:",step_count)        # 顯示步數
```

程式設計


```
98
99          payload=[{"id":"step","value":[step_count]}]
100         # 傳送資料到IoT平台
101         client.publish(
102                     b'/v1/device/請填入設備編號/rawdata',
103                     str(payload).encode()
104                     )
105         print('publish finish')
106         step_count=0                    # 重置步數
107         last_time=time.time()           # 重置時間
```

旗標創客

測試程式

- 開始執行後，一樣會開始判斷是否正在走路，等待 1 分鐘後，下方執行區塊會出現『publish finsh』：

旗標創客

測試程式

- 查看中華電信 IoT 平台是否有接收到值：

旗標創客

測試程式

旗標創客

測試程式

旗標創客

測試程式

- MQTT 的優點：

❶ 沒有複雜的表頭，擁有更輕量的傳輸內容

❷ 發佈端只要將資料上傳至伺服器，多個訂閱該主題的訂閱端都可以得到資料

❸ 網路有很多現成的 MQTT APP

❹ 不同的 MQTT 伺服器需要參數相同

旗標創客

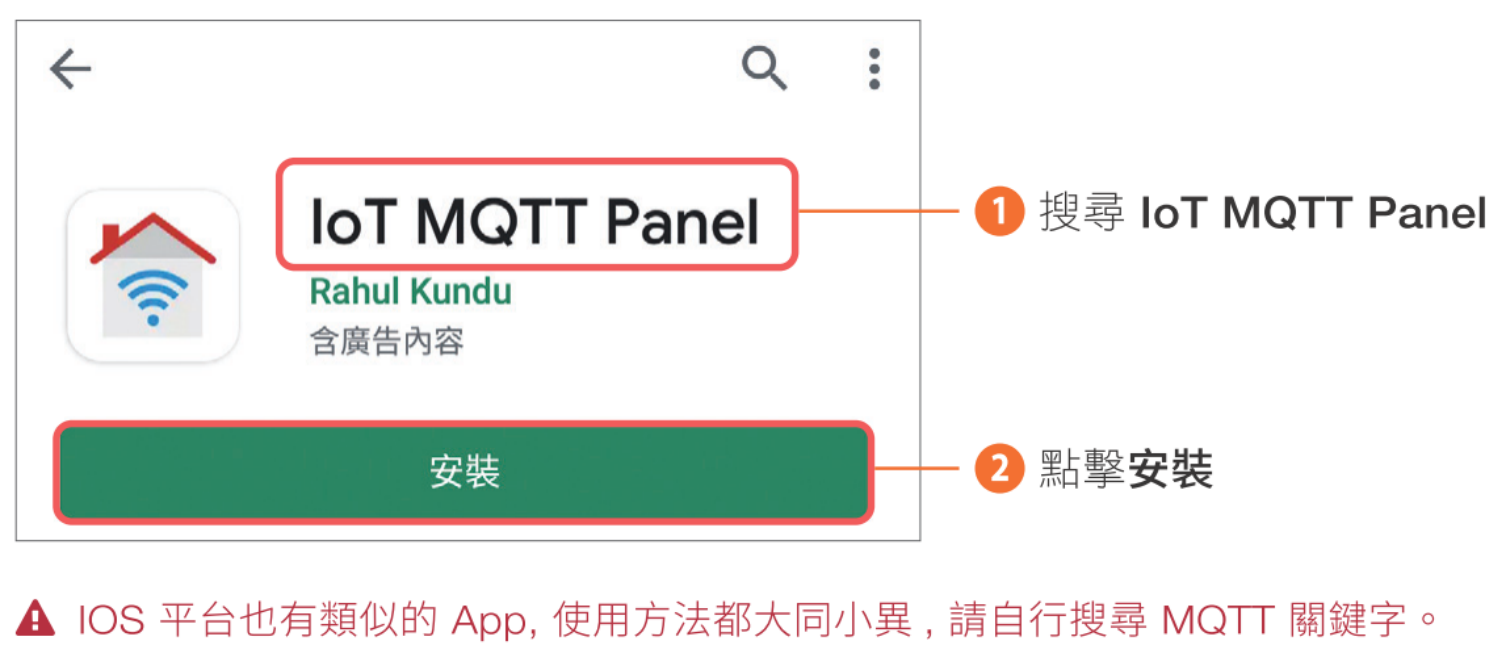
LAB13 手機步頻監測

- 實驗目的：
  利用 MQTT 將步頻傳送到手機上顯示
- 材料、線路圖：同 LAB09
- 開發環境： Thonny
- 實驗原理： MQTT APP 做為訂閱端 ；ESP32 為發佈端 。當 ESP32 使用 LAB12 將步頻資料上傳時 APP 就會同步得到資料。

旗標創客

安裝並設定手機　APP

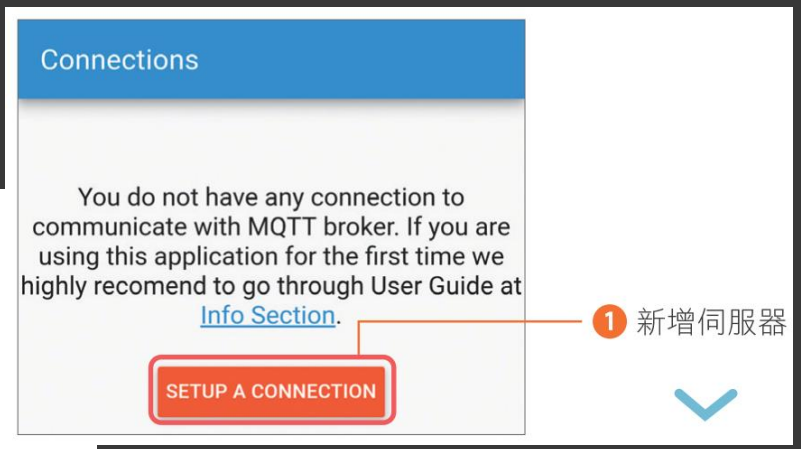
1　請先開啟 Google Play 並下載 APP：



旗標創客

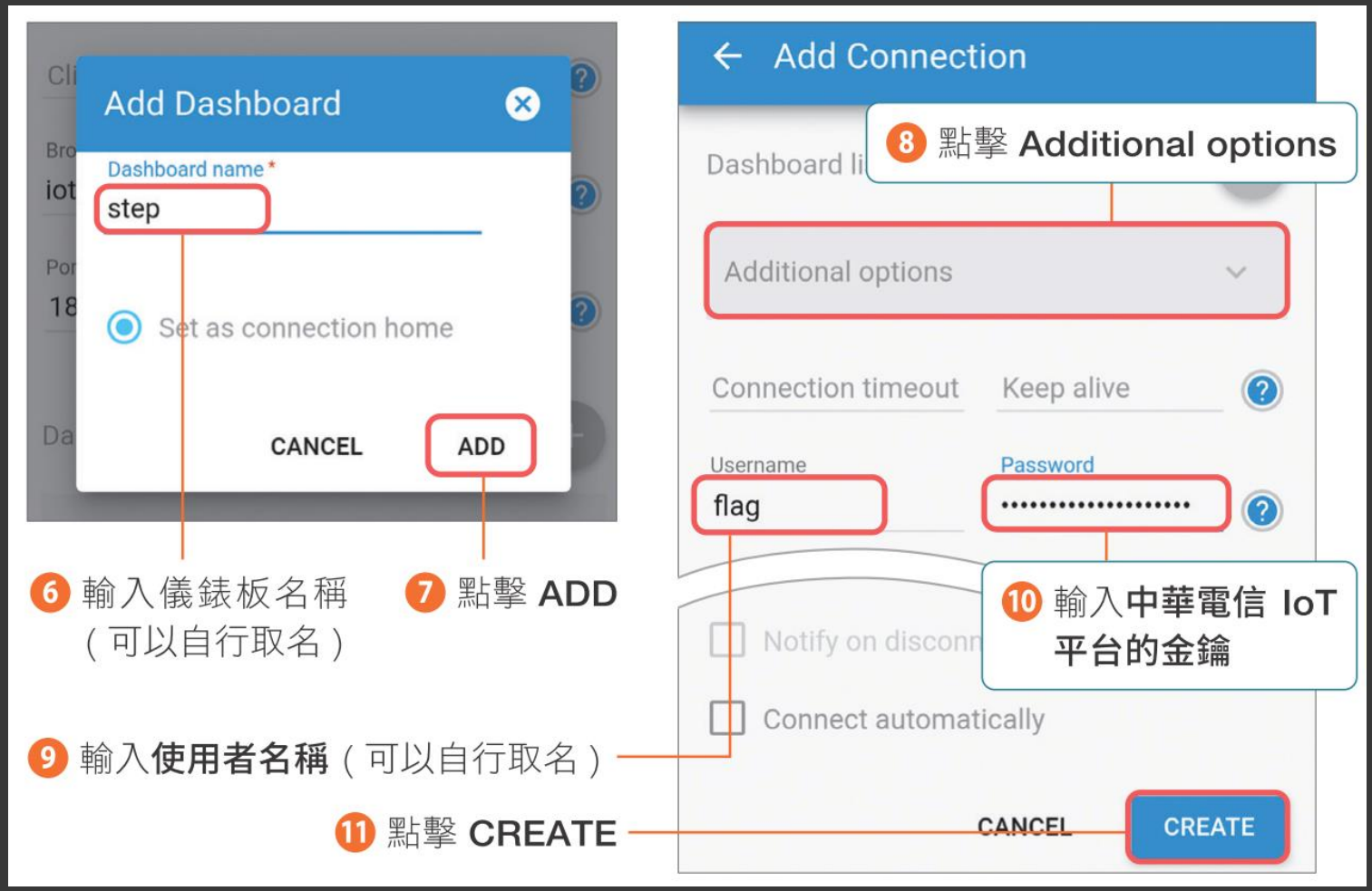
安裝並設定手機　APP

2　建立中介伺服器資訊：



旗標創客

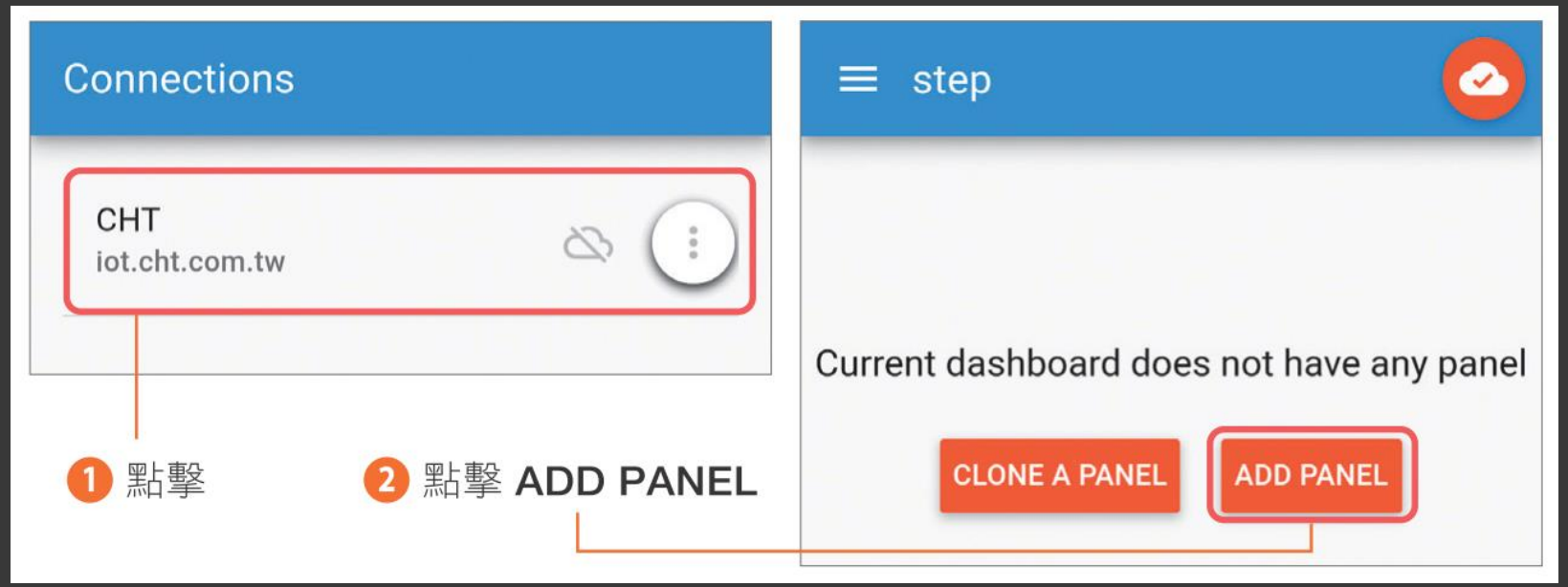
安裝並設定手機 APP



旗標創客

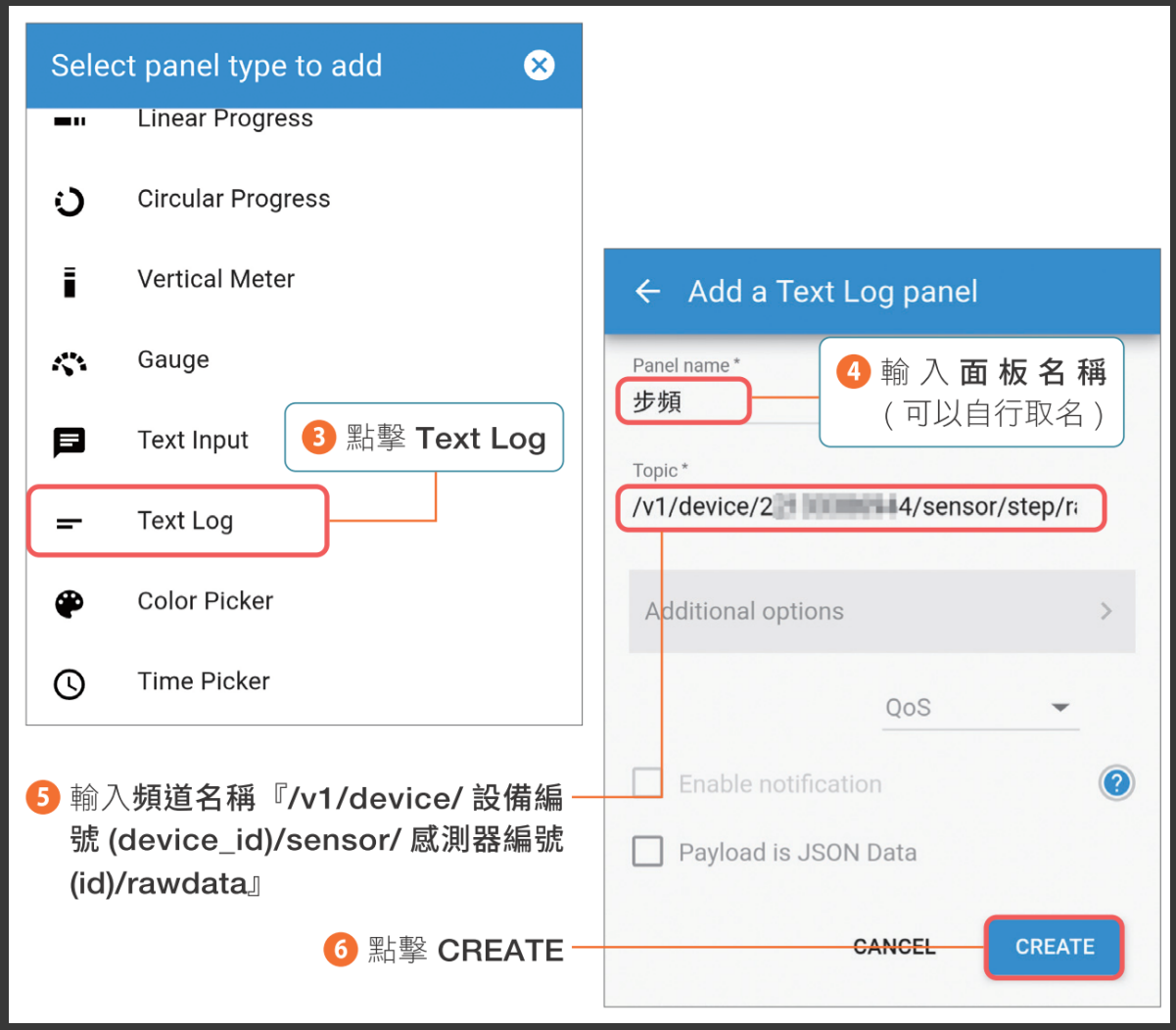
安裝並設定手機　APP

3　設定訂閱頻道與內容格



旗標創客

安裝並設定手機 APP



旗標創客

安裝並設定手機 APP

7 看到相關資訊，**value** 值代表『步數』

旗標創客