前往下載教學資源
www.geniopy.com.tw

Pomas

MicroPython AIOT

名威智慧通訊

專案講師 林羿君 Lynn Lin

為什麼要選Python程式語言呢?

簡單好學

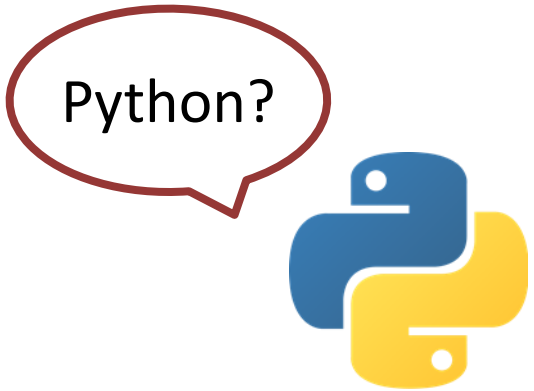
Python 的**基本語法規則很簡單**，學習曲線短，而且語法設計上已經預防你寫錯，可以減少初學時的時間浪費！

現成功能多

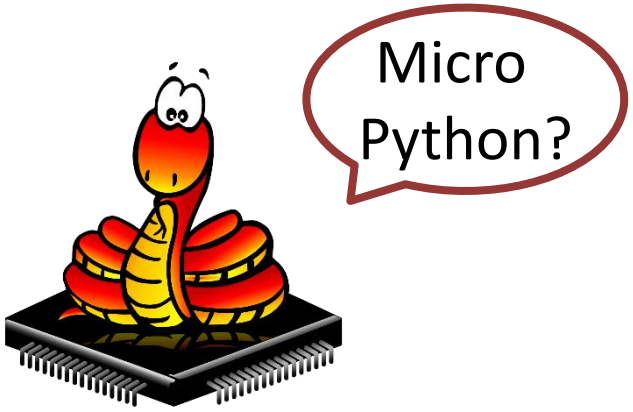
有全世界眾多善心人士寫好的功能（稱為**模組**）可以套用，要做任何事都可以省去自己從頭打造的時間！

大家都在用

現在最火熱的資料科學、人工智慧、機器學習等都是以 Python 程式語言為主流，學了到哪裡都會用得到！

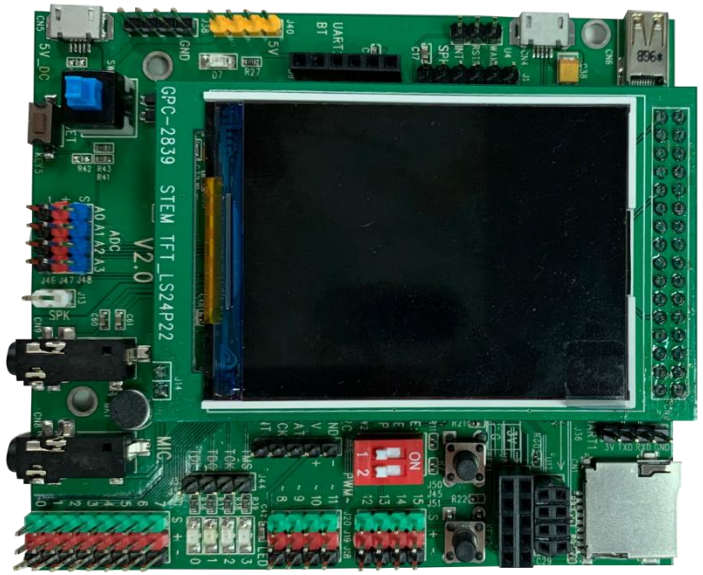


Python 可以唸拍嬸 (英國) 也可以唸拍賞 (美國)
這個單字是蟒蛇, 所以 Python 的 logo 就是蟒蛇
來源是英國喜劇 Monty Python's Flying Circus



架構

STRUCTURE



1 **Python 基礎入門**

自己動手寫出最簡單的程式
操控各種感測器在控制板上執行

2 **語音辨識**

簡易智慧管家

3 **人臉辨識**

刷臉時代的到來

4 **智慧家庭Demo**

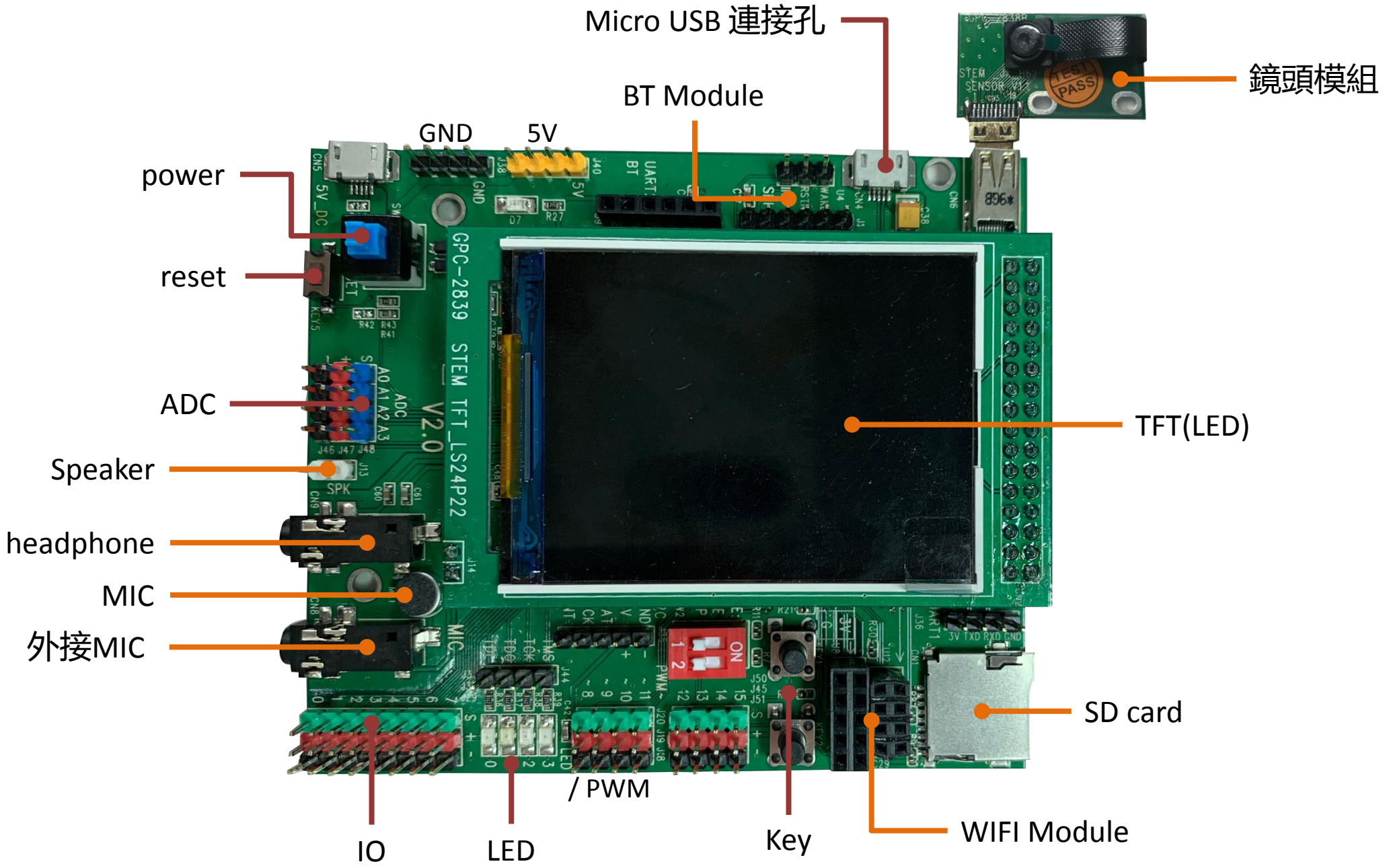
親身感受AI

0

PART

Pomas

硬體介紹


Micro USB 連接孔
BT Module
鏡頭模組
GND
5V
power
reset
ADC
Speaker
headphone
MIC
外接MIC
TFT(LED)
SD card
/ PWM
IO
LED
Key
WIFI Module

MIT？

Made In Taiwan!!

行政院自去年開始依《資通安
關及三級機關的資通安全實地
核重點對象，檢視其設備汰除

該官員進一步說，已通令全國包括中央
使用或採購的中國廠牌資通產品，各機
情況，回報期限在本月底，不少機關因
設備及中國品牌投影機，也將面臨全面

資安處指出，已責請各機關資
（含軟體、硬體及服務）相關
有經費需求應循程序爭取資源
進行汰換。

凌通科技股份有限公司(“凌通”)，是一家台灣企業位於台灣 300 新竹市新竹科學工業園區工業東四路 19 號，在此聲明 **GP329_STEM-KIT-20** 晶片由凌通設計，委由 SMT 廠:宸宥科技股份有限公司製造，是一家台灣企業位於台灣 300 新竹市香山區埔前路 240 號，及 PCB 廠:百為實業有限公司製造，是一家台灣企業位於台灣 238 新北市樹林區三俊街 81 巷 42 號。

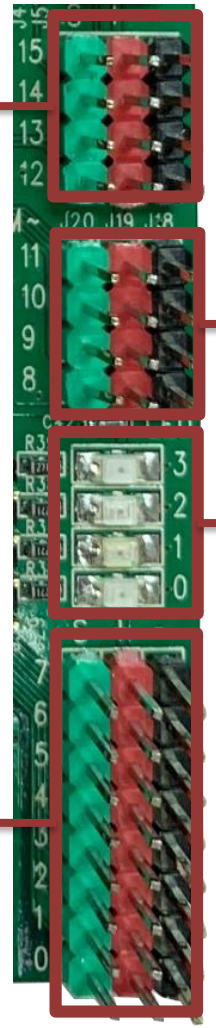
Generalplus Technology Inc.
凌通科技股份有限公司

簽署代表人：賈懿行 Kevin Chia
職稱：總經理 President

輸出/入腳位對照表

IO15	E7	3.3V	GND
IO14	E6		
IO13	E5		
IO12	E4		

IO7	F3	3.3V	GND
IO6	F2		
IO5	F1		
IO4	F0		
IO3	C3		
IO2	C2		
IO1	C1		
IO0	C0		



PWM11	E3	3.3V	GND
PWM10	E2		
PWM9	E1		
PWM8	E0		

LED3	C15	3.3V	GND
LED2	C14		
LED1	C13		
LED0	C12		

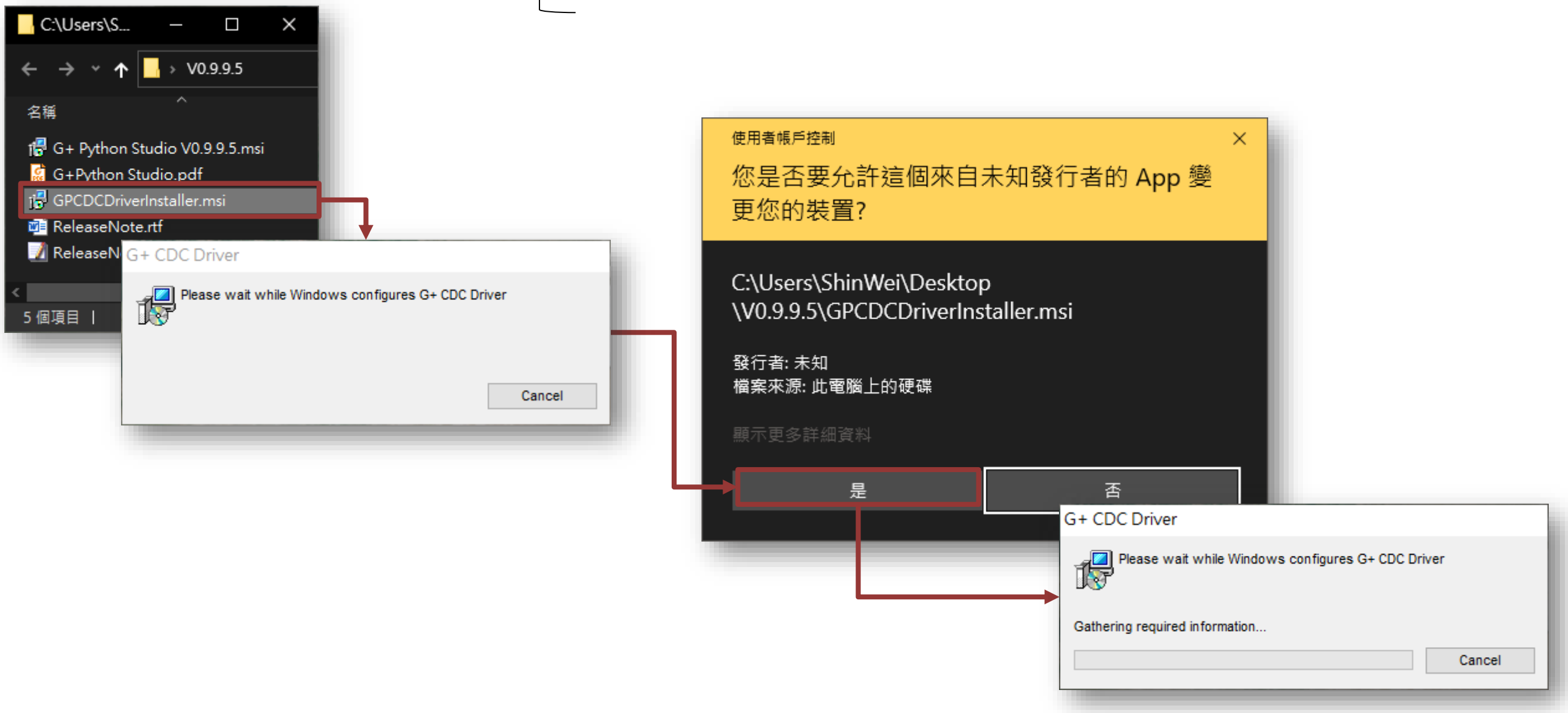
1

PART

安裝與測試環境

動手做: 安裝環境

安裝驅動程式

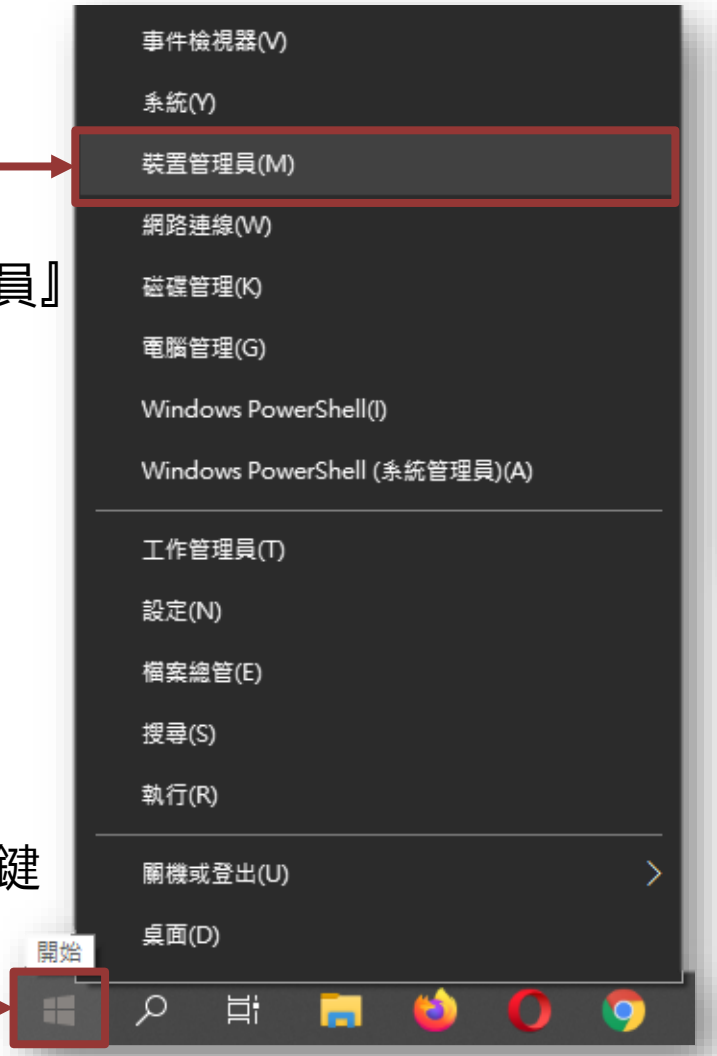

C:\Users\S...
V0.9.9.5
名稱
G+ Python Studio V0.9.9.5.msi
G+Python Studio.pdf
GPCDCDriverInstaller.msi
ReleaseNote.rtf
ReleaseN
5 個項目
G+ CDC Driver
Please wait while Windows configures G+ CDC Driver
Cancel
使用者帳戶控制
您是否要允許這個來自未知發行者的 App 變更您的裝置?
C:\Users\ShinWei\Desktop\V0.9.9.5\GPCDCDriverInstaller.msi
發行者: 未知
檔案來源: 此電腦上的硬碟
顯示更多詳細資料
是
否
G+ CDC Driver
Please wait while Windows configures G+ CDC Driver
Gathering required information...
Cancel

連接 USB 傳輸線並檢查連接埠編號

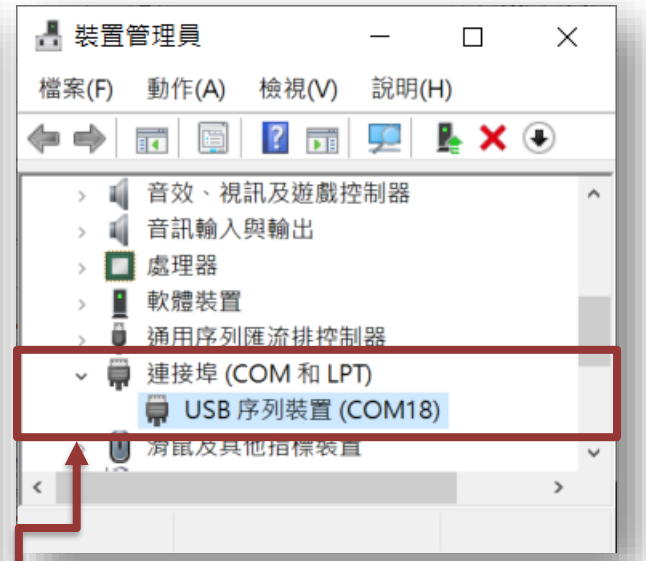
① Micro-USB 請接這裡

② 請把開關下押開啟電源

④ 選『裝置管理員』



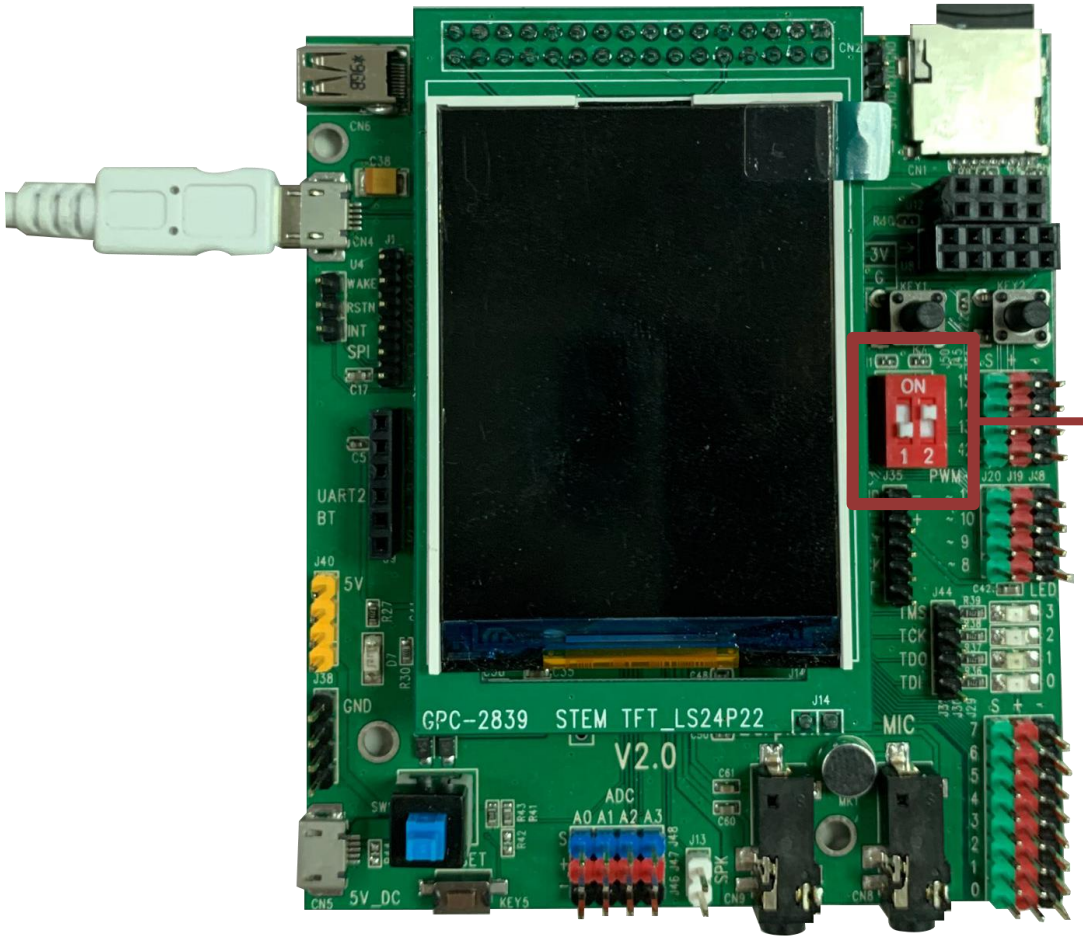
③ 按滑鼠右鍵



⑤ 點開『連接埠/USB 序列裝置』確認編號 (你看到的和我不一定會一樣)

如果看到多個序列埠一定不會是 COM1

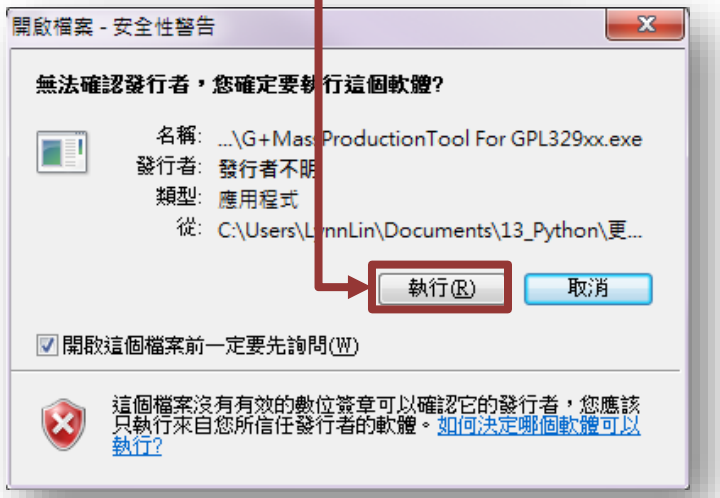
燒錄韌體

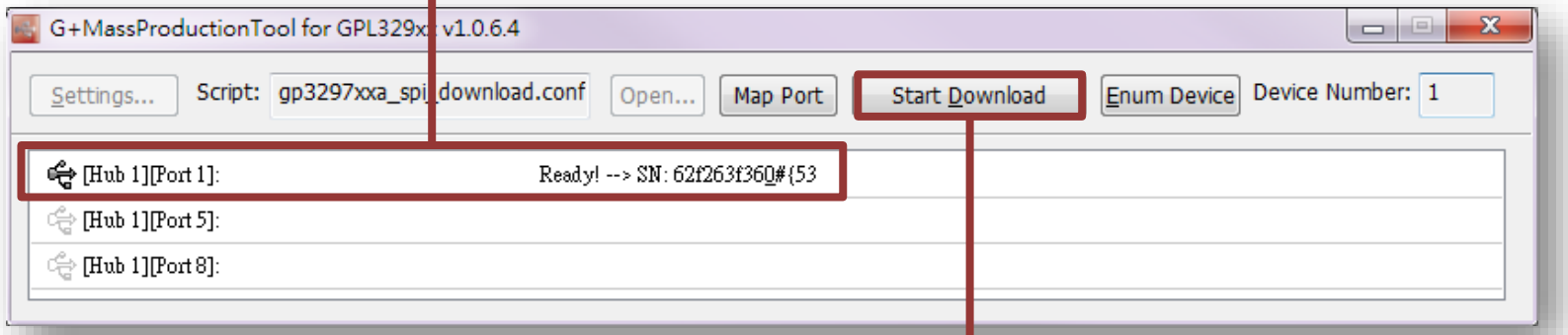


將**2號**指撥開關
向上撥進入燒錄模式

燒錄韌體

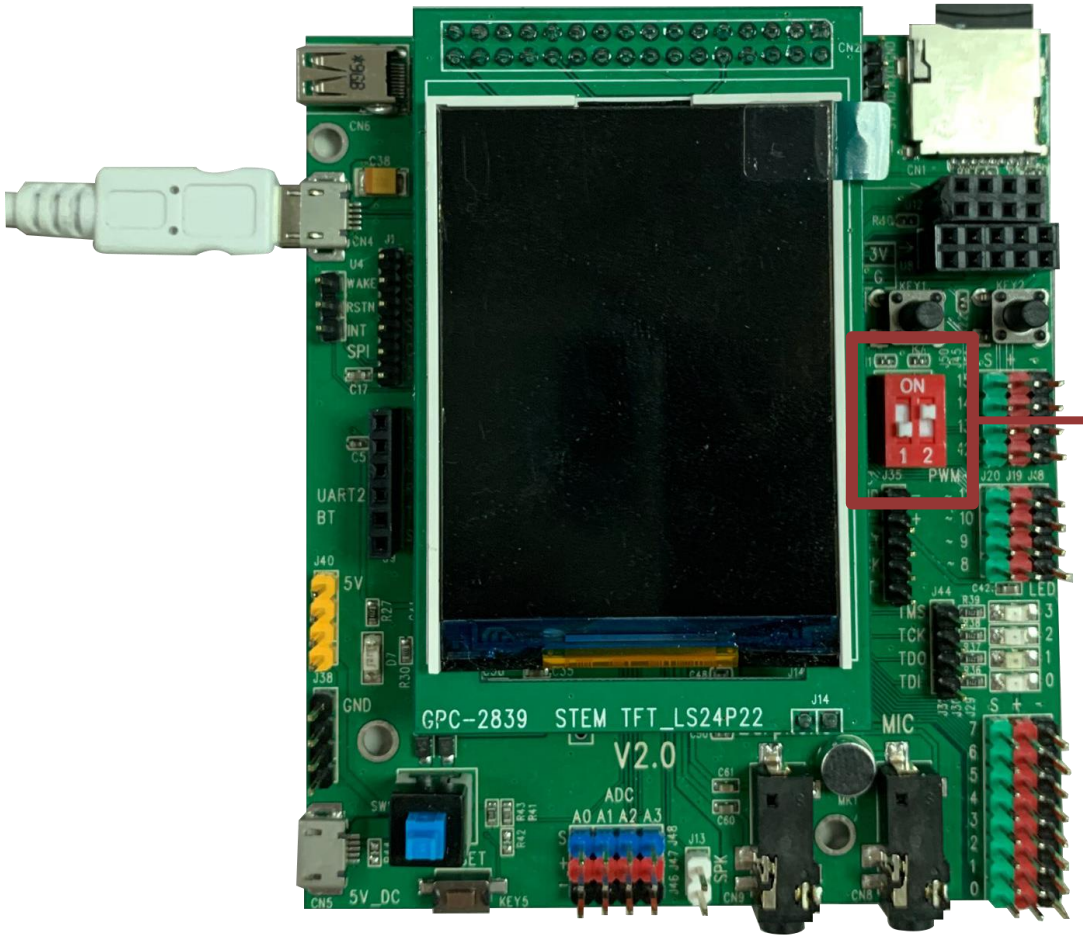

更新工具(Code_Ver12154)_V2.0
檔案資料夾
USBMassProductionTool_For_GPL329XX
檔案資料夾
G+MassProductionTool For GPL329xx
G+MassProductionTool


開啟檔案 - 安全性警告
無法確認發行者，您確定要執行這個軟體?
名稱: ...\G+MassProductionTool For GPL329xx.exe
發行者: 發行者不明
類型: 應用程式
從: C:\Users\LynnLin\Documents\13_Python\更...
執行(R)
取消
開啟這個檔案前一定要先詢問(W)
這個檔案沒有有效的數位簽章可以確認它的發行者，您應該只執行來自您所信任發行者的軟體。如何決定哪個軟體可以執行?


系統自動偵測代入USB連接埠
G+MassProductionTool for GPL329xx v1.0.6.4
Settings...
Script: gp3297xxa_spi_download.conf
Open...
Map Port
Start Download
Enum Device
Device Number: 1
[Hub 1][Port 1]: Ready! --> SN: 62f263f360#{53
[Hub 1][Port 5]:
[Hub 1][Port 8]:
按下開始燒錄


G+MassProductionTool for GPL329xx v1.0.6.4
Settings...
Script: gp3297xxa_spi_download.conf
Open...
Map Port
Stop Download
Enum Device
Device Number: 1
[Hub 1][Port 1]: 100% Download Completed!
[Hub 1][Port 5]:
[Hub 1][Port 8]:
燒錄完成，按下停止即可關閉程式

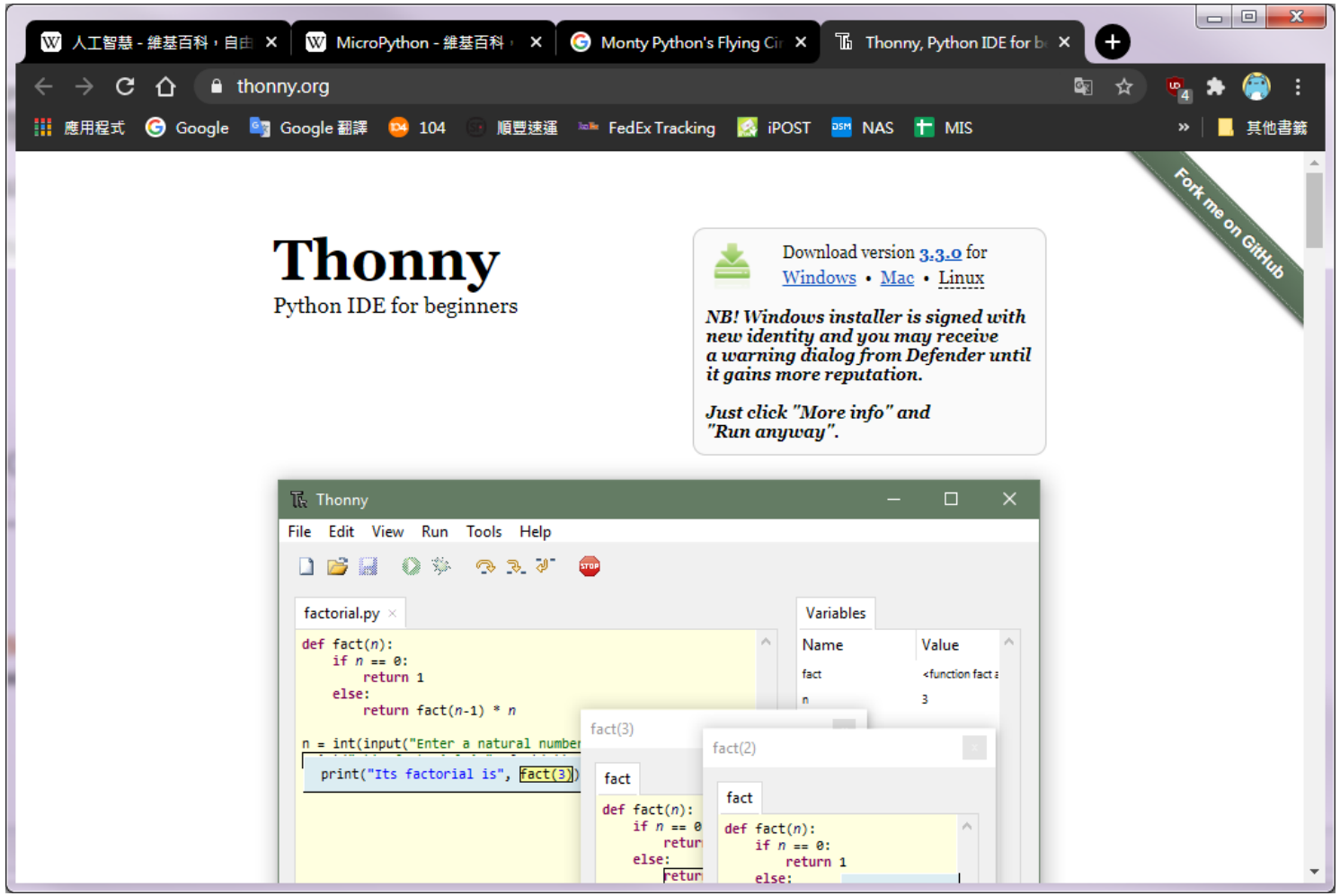
完成燒錄



開發板韌體不管是**升級**還是**降級**
都是以同一種方式燒錄哦!

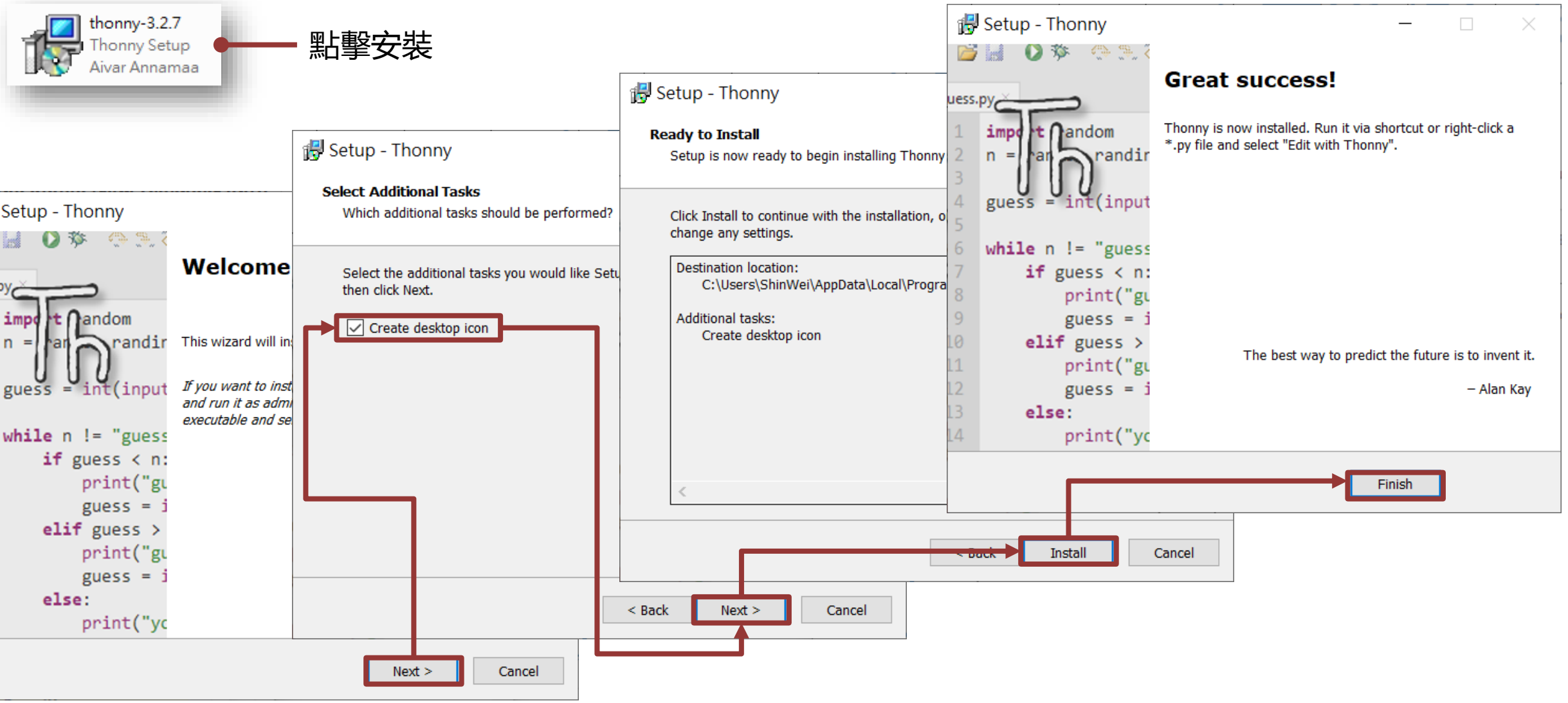
燒錄完成後記得將開關撥回原位
不然開發板會一直在燒錄模式，
無法進行其他操作!

安裝Thonny 開發環境



到 https://thonny.org 下載

安裝Thonny開發環境

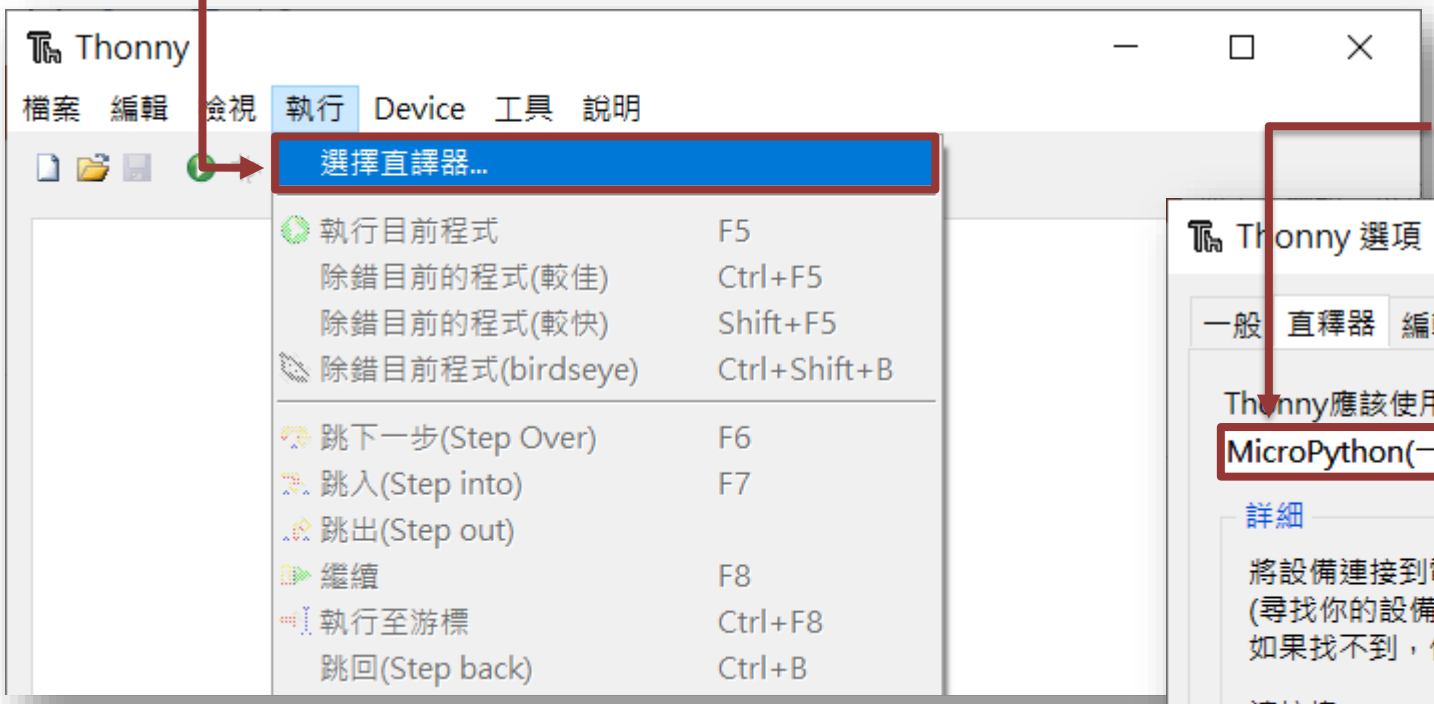


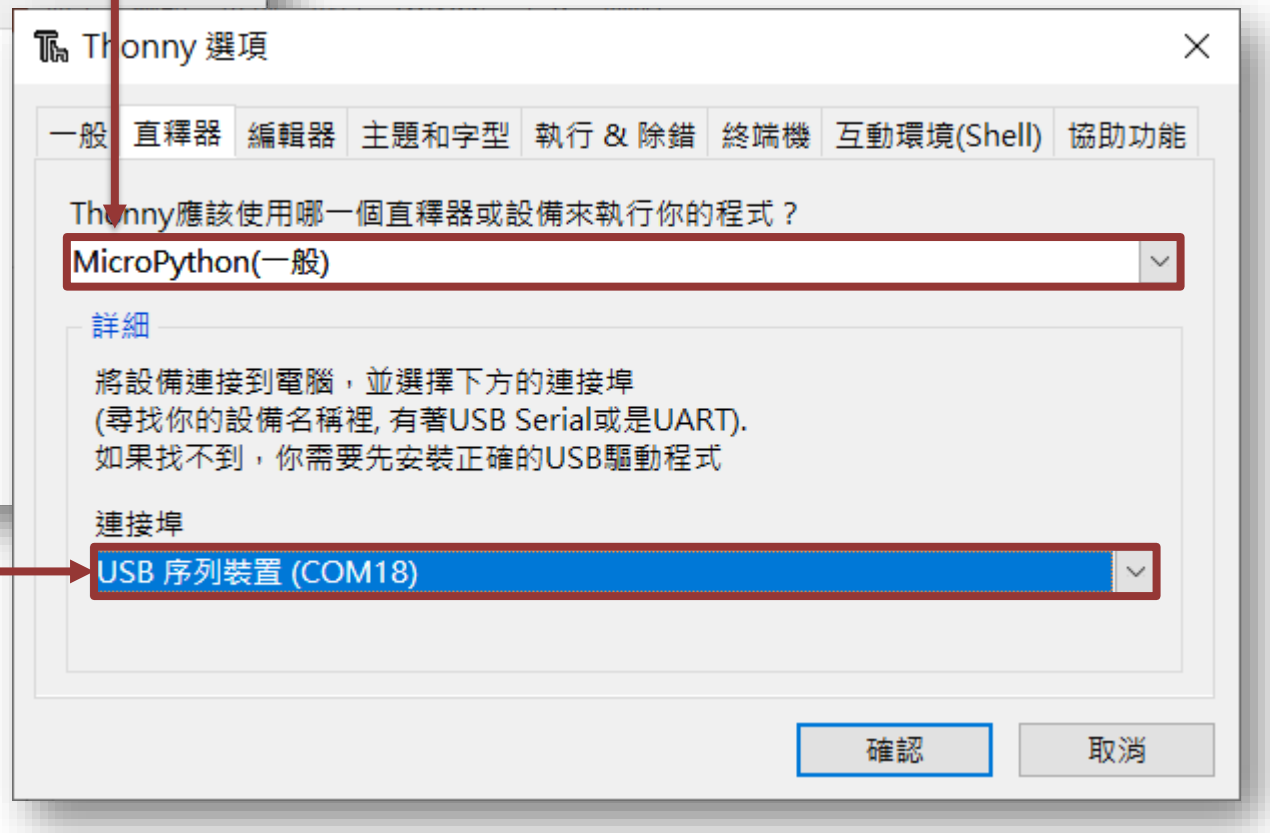
設定 Thonny

Thonny 是通用的程式開發環境
可開發**不同平台**的 Python 程式

① 執行『執行/選擇直譯器』

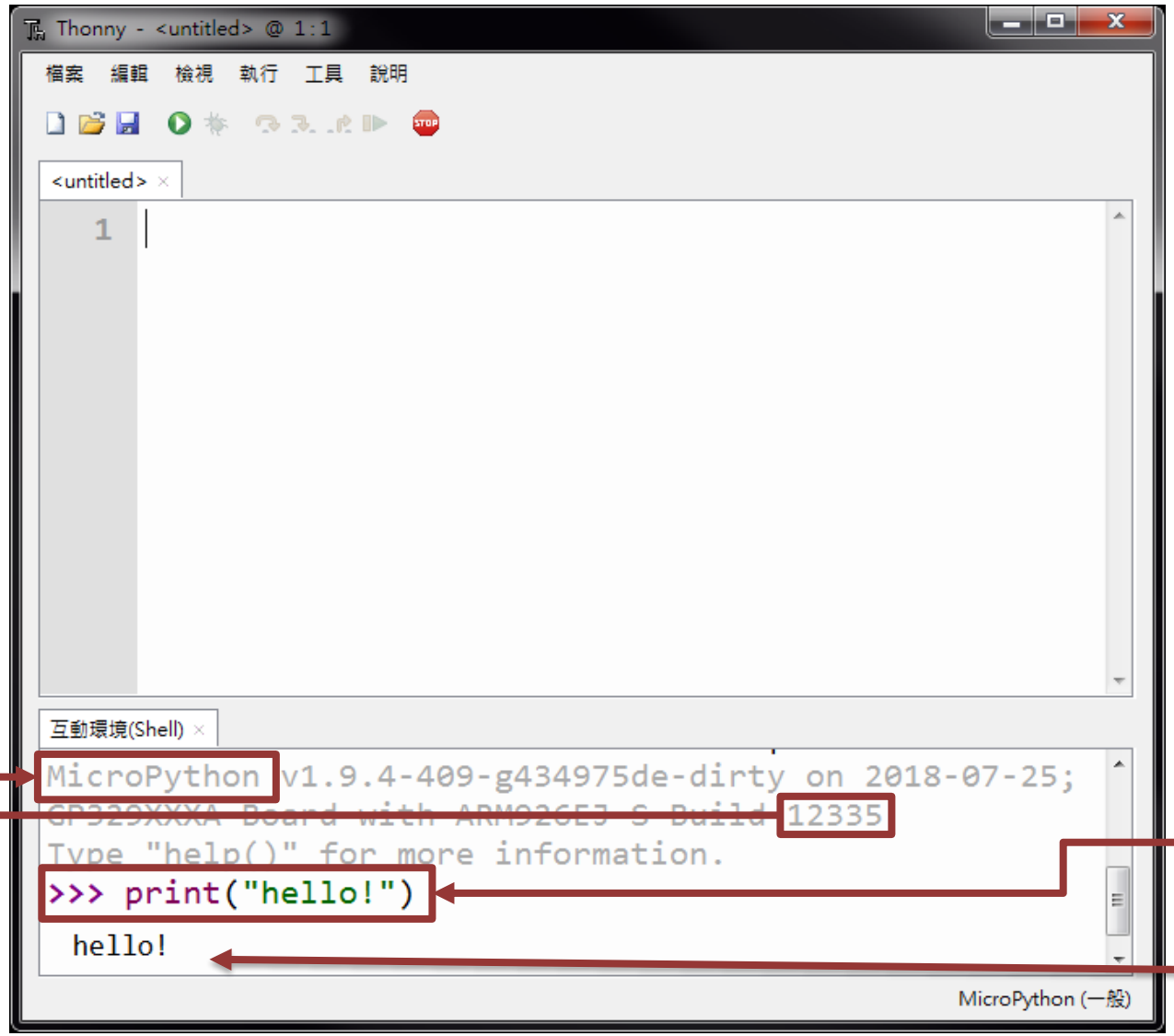
② 選擇『MicroPython(一般)』





③ 依據剛剛看到的連接埠編號

快速測試



① 看到這個字樣就
表示安裝設定成功

② 韌體版本編號

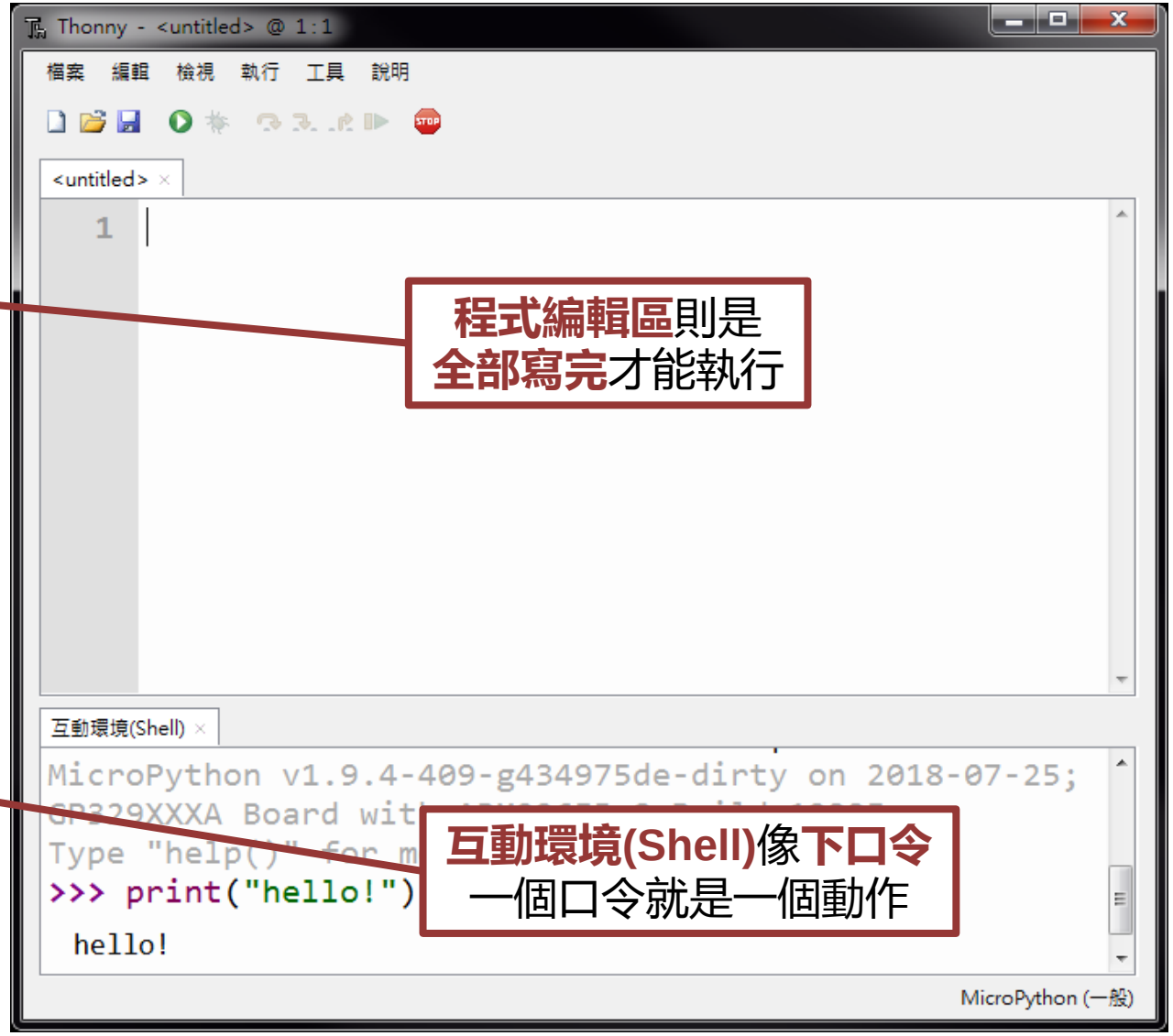
Python 程式有**區分大小寫**
括號、引號都要正確配對

文字會隨你完整度而變色
方便你檢視有沒有打正確

③ 請輸入 print(“hello!”)

④ 看到顯示就
表示正確運作

Thonny 環境簡介



向前走一步
向右轉
向前走一步
機器人動作

程式編輯區則是
全部寫完才能執行

向前走一步
機器人動作
向右轉
機器人動作
向前走一步
機器人動作

互動環境(Shell)像**下口令**
一個口令就是一個動作

拖拉這條線可
調整窗格空間

2

PART

Python 基礎入門

動手做:寫出最簡單的程式

Python 的核心 ★ 物件

注意**大小寫是不一樣的**
括號、引號也都要**配對**

① 請先試看看

互動環境(Shell)

```
>>> 'abc'.upper()
'ABC'
>>>
```

② 會得到大寫的版本

```
'abc'.upper()
```

物件.方法()

執行 'abc'的 **變大寫** 方法

執行**汽車**的**往前跑**方法

同一個物件有很多不同的方法

互動環境(Shell)

```
>>> 'abc'.find('b')
1
>>>
```

執行 'abc' 的 **找出** 'b' **位置** 方法
括號內填入的**參考資料** 'b' 稱為**參數**
Python 中順序編號都是**從 0 開始算**

互動環境(Shell)

```
>>> 'abc'.replace('b', 'z')
'azc'
>>>
```

執行 'abc' 的 **把** 'b' **換成** 'z' 方法

互動環境(Shell)

```
>>> "abc".upper()
'ABC'
>>> "it's a book".upper()
"IT'S A BOOK"
>>>
```

'abc' 這種叫做**字串**物件
可以用 'abc' 或是 "abc"
但**頭尾引號**一定要**一致**

物件有各種不同的類別

可以用 type() 檢查物件的類別

互動環境(Shell)

```
>>> type('abc')
<class 'str'>
>>> type(20)
<class 'int'>
>>> type(20.3)
<class 'float'>
>>>
```

string (字串) 類別

20 是 integer 類別 (整數)

20.3 是 floating point 類別 (就是帶小數的數值)

互動環境(Shell)

```
>>> 'abc' + 'ABC'
'abcABC'
>>> 20 + 10
30
>>> 20.3 + 20
40.3
>>>
```

物件可以進行**運算**
同一符號意義不同

整數物件

字串物件

互動環境(Shell)

```
>>> 20 + 'abc'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported types for __add__: 'int', 'str'
>>>
```

整數和字串沒辦法相加

變數

將字串'abc' 指定為變數**name**

互動環境(Shell)

```
>>> name = 'abc'
>>> name.upper()
'ABC'
>>> name.find('b')
1
>>> name.replace('b', 'z')
'azc'
>>> name1 = name
>>> name1
'abc'
>>>
```

可以用name這個變數，
執行物件的各種方法

變數通常會以有意義的名稱定義

計算結果會產生一個新的物件
並且用 = 號幫新物件取好名字

```
>>> price=100
>>> discount=0.2
>>> real_price=price*(1-discount)
>>> real_price
80.0
```

Python內建函式

函式可以提供特定的功能
用法跟物件的方法很相似
但是不需要『**物件.**』開頭

Python 提供隨時可用的稱為**內建 (built-in) 函式**

互動環境(Shell)

```
>>> print("Hello, world")
 Hello, world
>>> len("abc")
3
>>> type("abc")
<class 'str'>
>>> dir("abc")
['count', 'endswith', 'find', 'format', 'index'
, 'isalpha', 'isdigit', 'islower', 'isspace', '
isupper', 'join', 'lower', 'lstrip', 'replace',
'rfind', 'rindex', 'rsplit', 'rstrip', 'split',
'startswith', 'strip', 'upper']
```

取得字串內的文字長度

取得物件的類別

取得物件可用方法清單

另外一種自己寫程式建立的稱為**自訂函式**

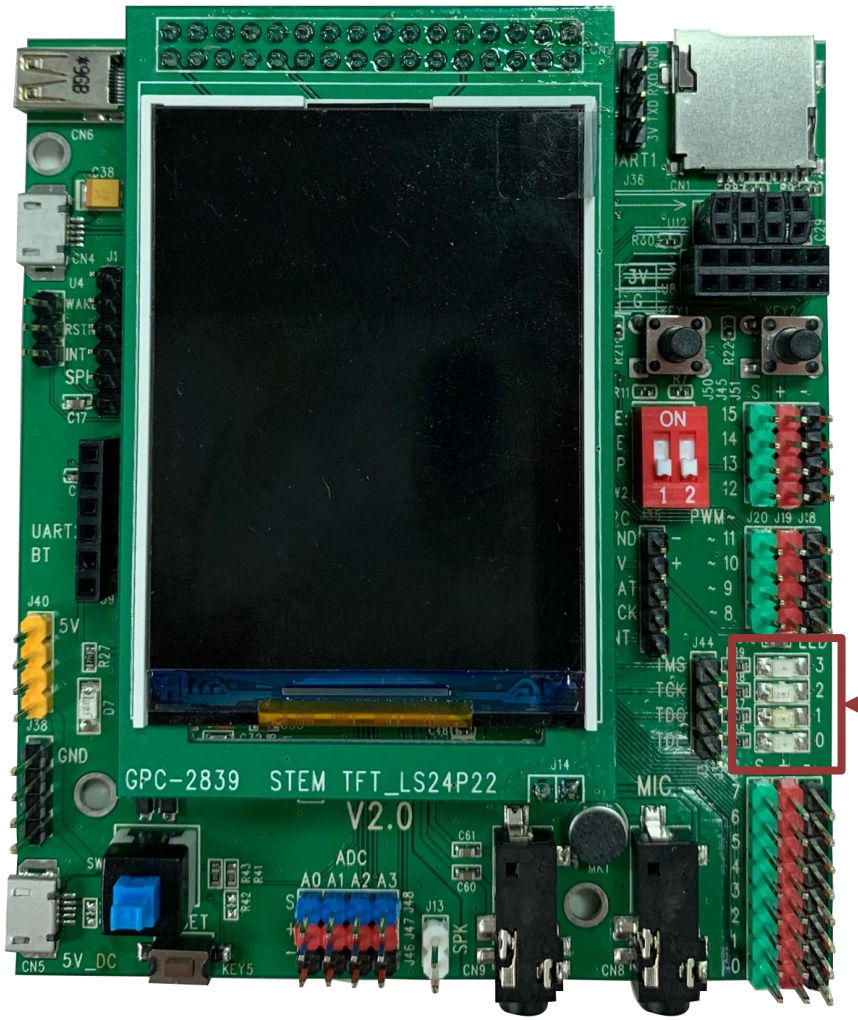
3

PART

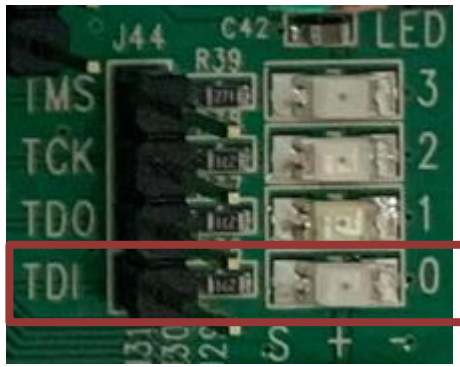
點亮人生的光明燈

數位輸出 (Digital Output)

動手做：點亮 LED 燈



這裡有 4 顆 LED



LED 3
LED 2
LED 1
LED 0

本實驗要點這一顆燈

原理

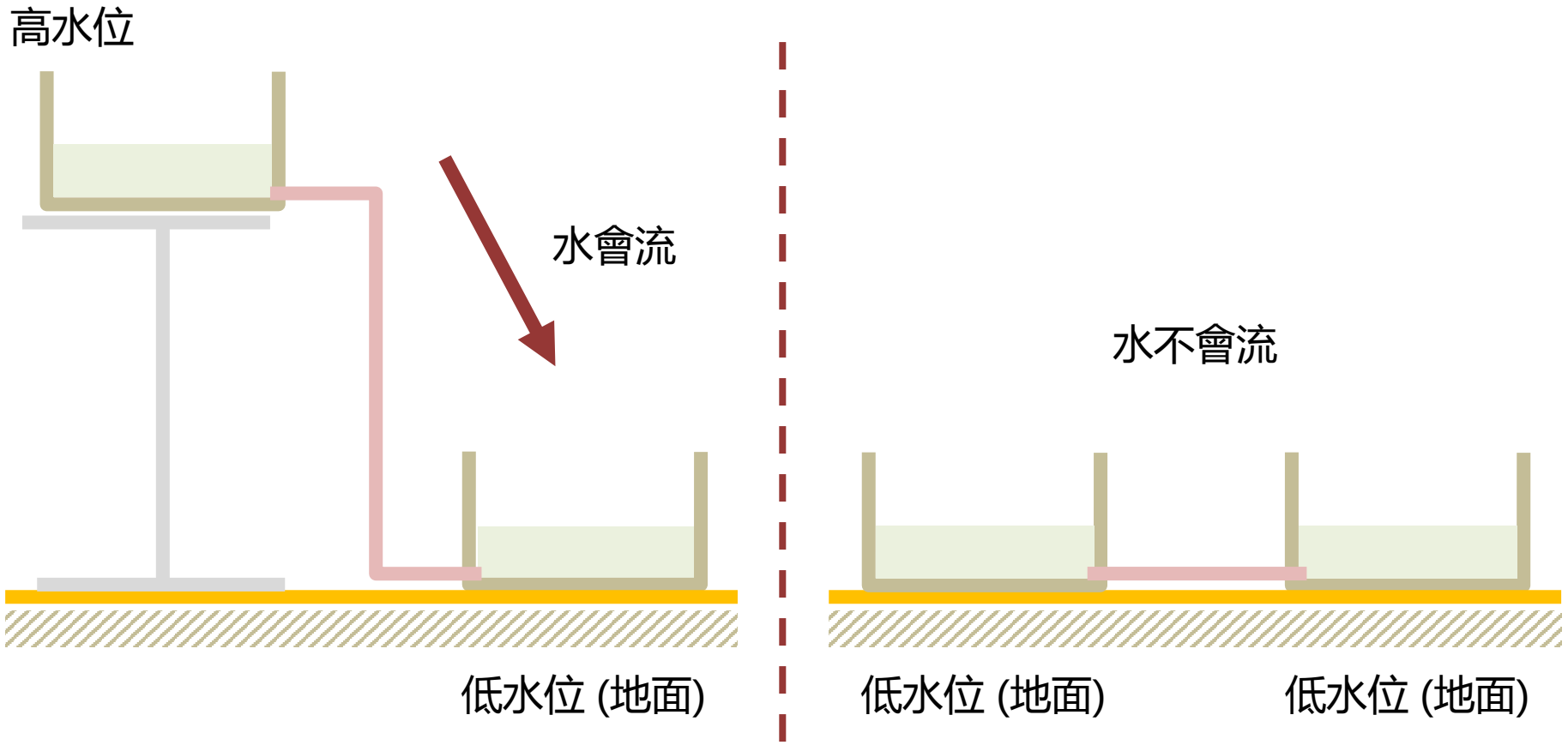
這個水車要怎樣才會轉？

原理 水 (電) 流動，水車 (燈) 才會轉 (亮)

原理

控制水位就可控制水的流動

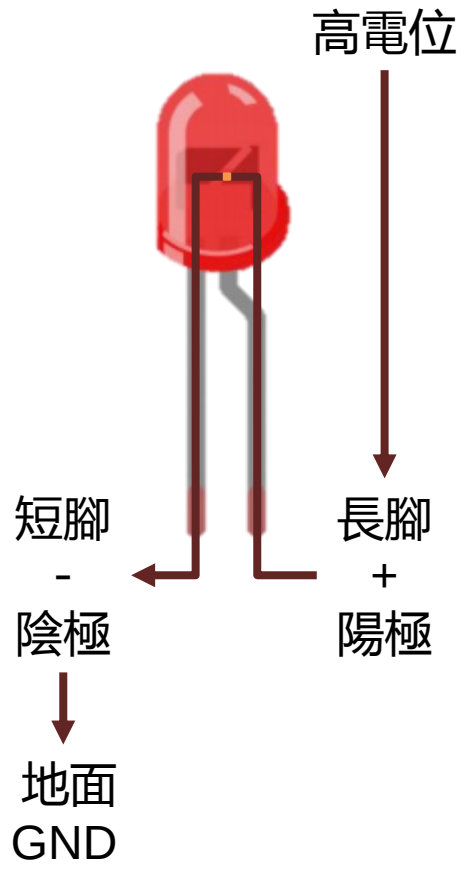
如果能夠控制升降左邊水池
就可以自由控制水流動與否



原理

發光二極體 - LED

LED 具有**方向性**
跟推門/拉門一樣
方向對了才會亮



原理

水往低處流

水 (電) 會從高水位 (高電位) 流向低水位 (低電位)

原理

沒有防洪就會變成這樣!

《看見台灣》

空中攝影/齊柏林 圖片版權/台灣阿布電影股份有限公司

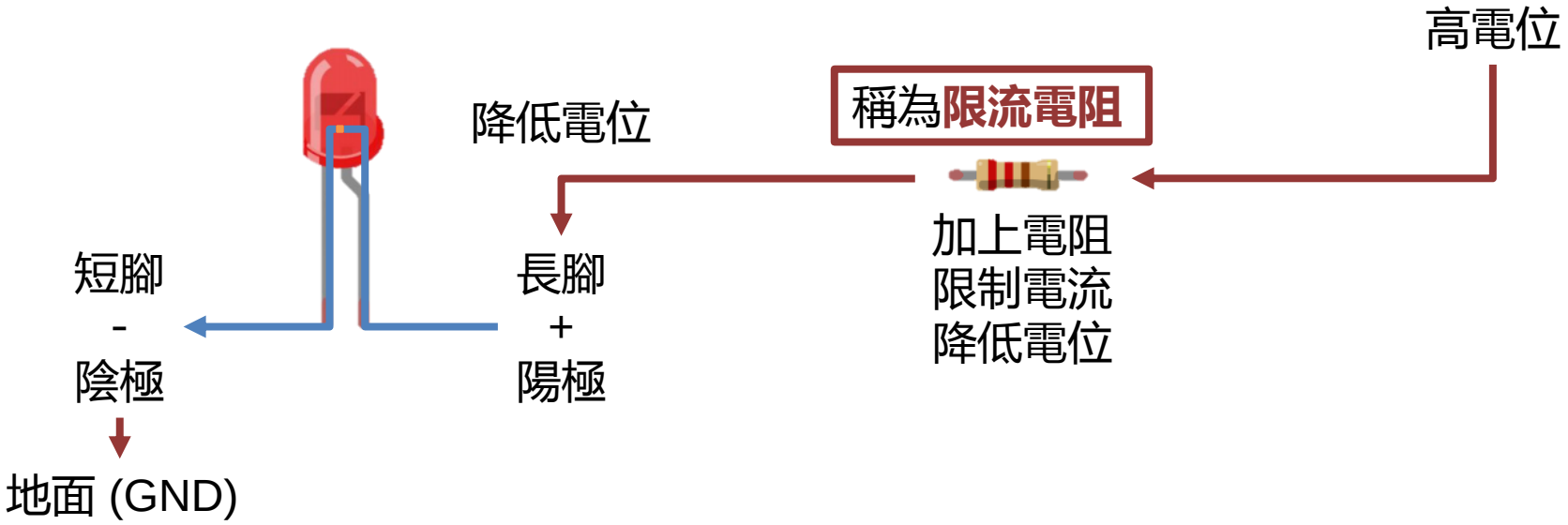
原理

用攔砂壩限流降壓

不同攔阻能力的攔砂壩，限流降壓的能力會不同

原理

用電阻限流降壓



電阻可比擬成漏斗
水流經過後就變小

10KΩ (歐姆)
K=1000

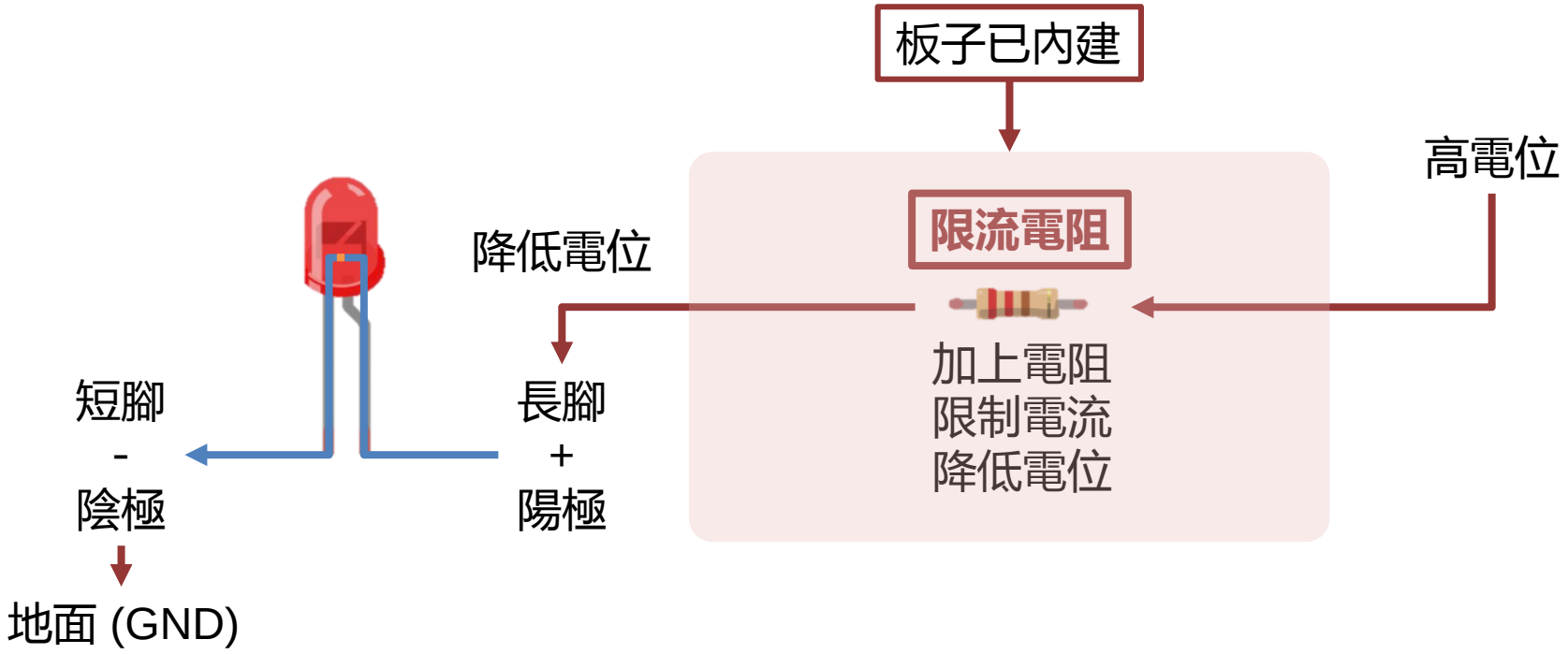
220Ω (歐姆)

不同漏斗阻擋能耐不同
限流降壓的程度也不同

漏斗圖片引用自 http://big5.made-in-china.com/

原理

內建限流電阻



原理

GPIO 腳位就是魔術水池

General Purpose Input/Output
中文稱為**通用目的的輸入/輸出**

這些 GPIO 腳位/插槽可透過程式控制瞬間變化**輸出高/低電位**

LED燈	腳位		
LED3	C15		
LED2	C14	3.3V	GND
LED1	C13		
LED0	C12		

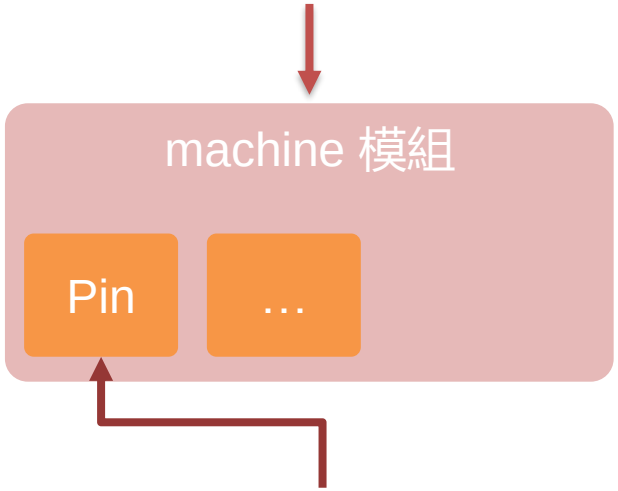
板子上有一條電線從 C12 經過 LED0 到 GND

動手做

呼叫外部支援-控制腳位

Python 本身沒有控制腳位的功能
因此必須倚靠外部功能才做得到

Python 是以**模組**提供外部功能
你可以把模組看成**人力派遣公司**
machine 是專精**硬體控制**的模組



模組內再以**類別或函式**的形式提供細部功能
你可以把類別看成是人力派遣公司內的**部門**
可以根據你的需求派出**派遣人員** (建立物件)
Pin 類別可產生幫你**控制 GPIO 腳位**的物件

動手做 寫程式控制 IO 腳位

① 使用外部模組功能的動作叫做**匯入 (import)**：

```
>>> from machine import Pin
```

模組名稱 (人力派遣公司)

類別 (部門) 名稱

② 請**類別**建立**物件 (請該部門依據需求派出人力)**：

```
>>> led = Pin('C12', Pin.OUT)
```

變數 (派遣人員) 的名字

腳位編號

部門中事先定義好的**功能代號**
Pin.OUT是指用來**輸出**高低電位

③ 利用物件**控制腳位 (請該派遣人員操作 LED)**：

```
>>> led.value(1)
```

value 方法可控制高低電位

1 表示**高**電位
0 表示**低**電位

動手做

使用模組提供的函式

Python 本身沒有控制時間的功能

我們提供的 gpb 模組
(gpb -> genio py board)
內含有與**時間**相關的功能

模組內也有以**函式**形式提供的功能
(不歸屬部門獨立作業的專業人員)
delay 函式可讓程式**等候一段時間**

動手做 寫程式控制時間狀態

① 從**模組**匯入**函式**：

```
>>> from gpb import delay
```

模組名稱 ↑　　↑ 函式名稱

② 使用**函式**提供的功能： 不需要先建立變數, 直接就可以使用的函式

```
>>> delay(3000)
```

函式名稱 ↑　　↑ 這個參數是暫停的毫秒數 (1/1000 秒)

匯入模組也可以用這樣的寫法：

```
>>> import gpb
>>> gpb.delay(3000)
```

只匯入模組名稱 ←

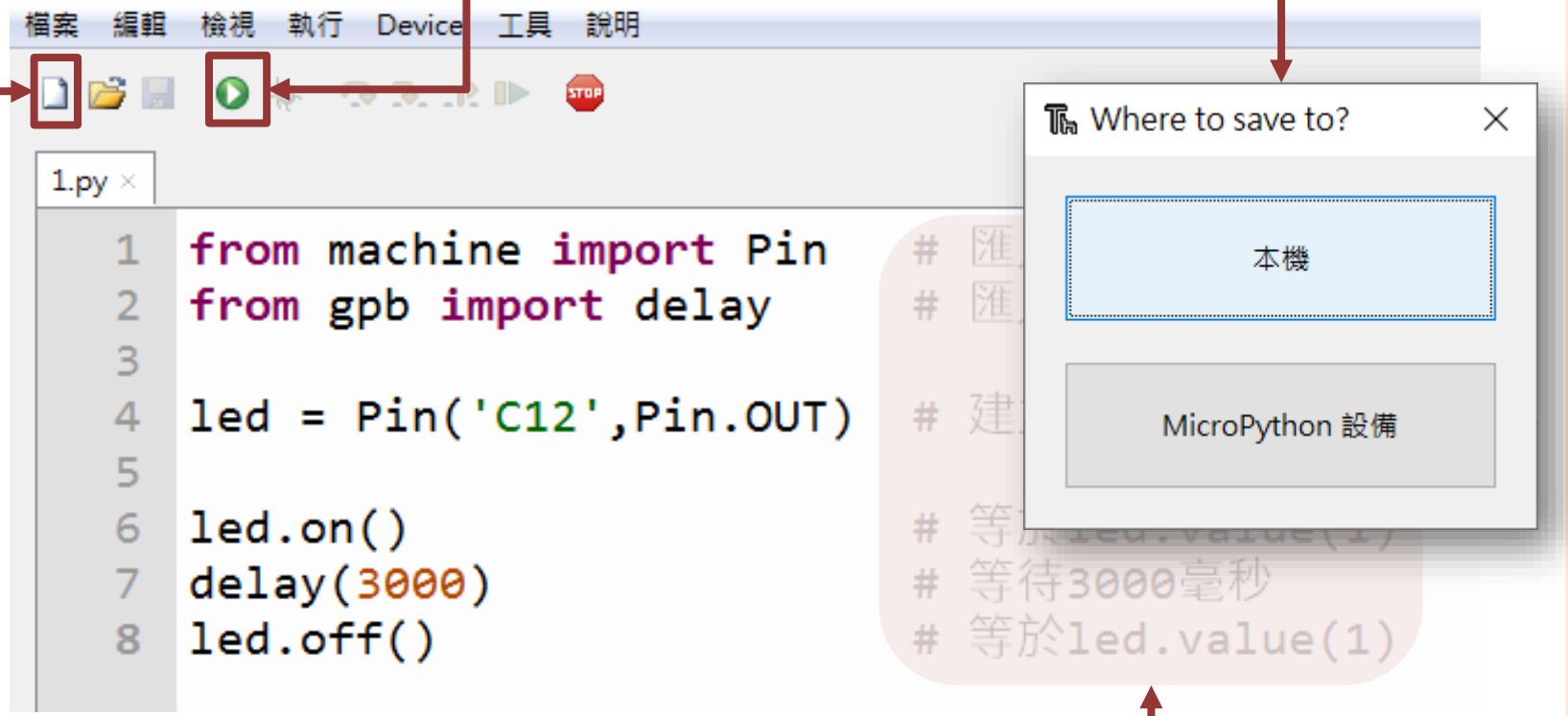
但就必須額外冠上模組名稱 ↑

動手做

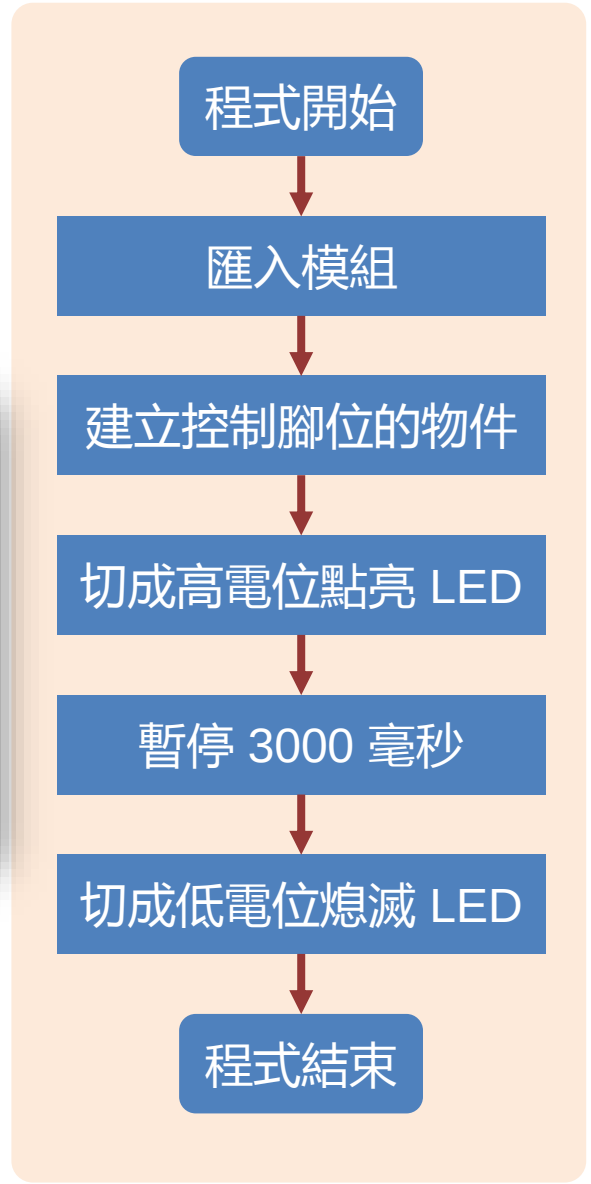
撰寫程式

① 按此建立新檔案

③ 按下執行會請您先存檔



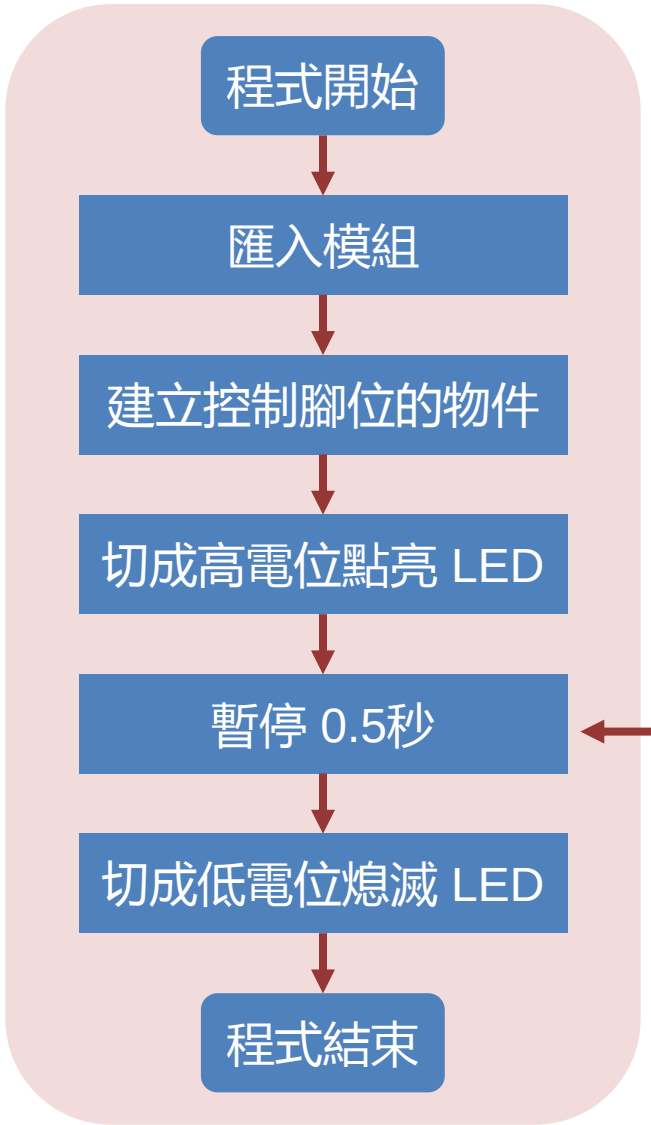
② “ # ” 開始後面是註解, 不一定要打



執行後請注意看板子上的 LED，若沒看到可以重新再執行一次

練習題

點亮後0.5秒熄滅



在程式中要改成delay(???)

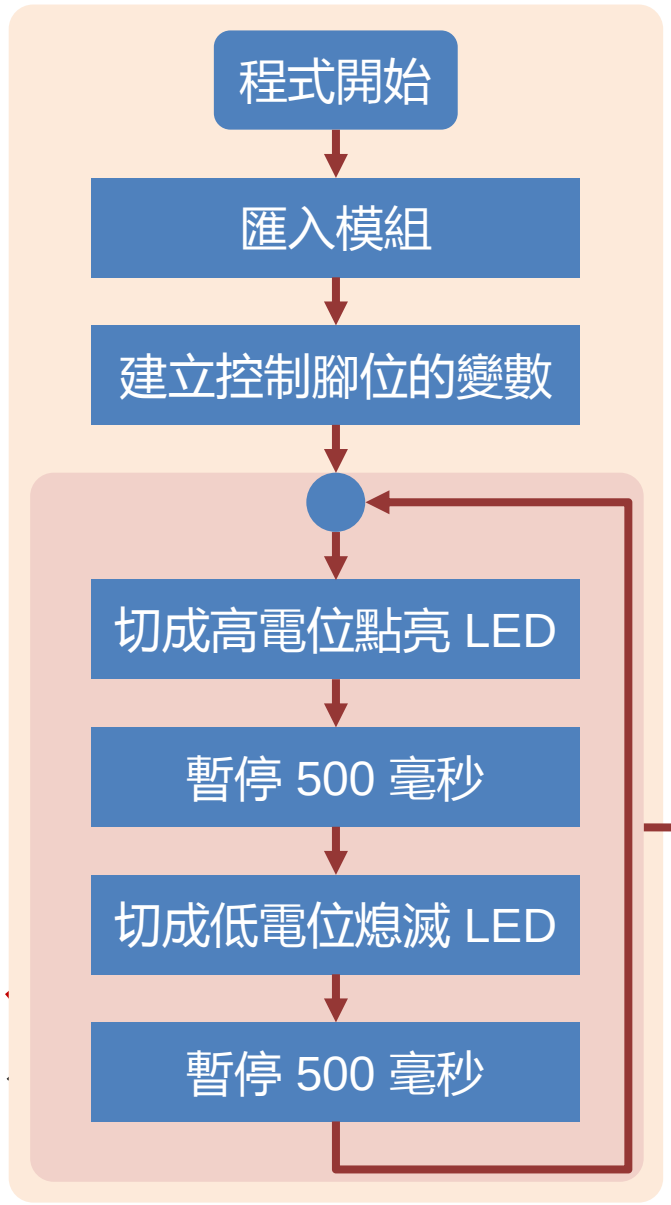
4

PART

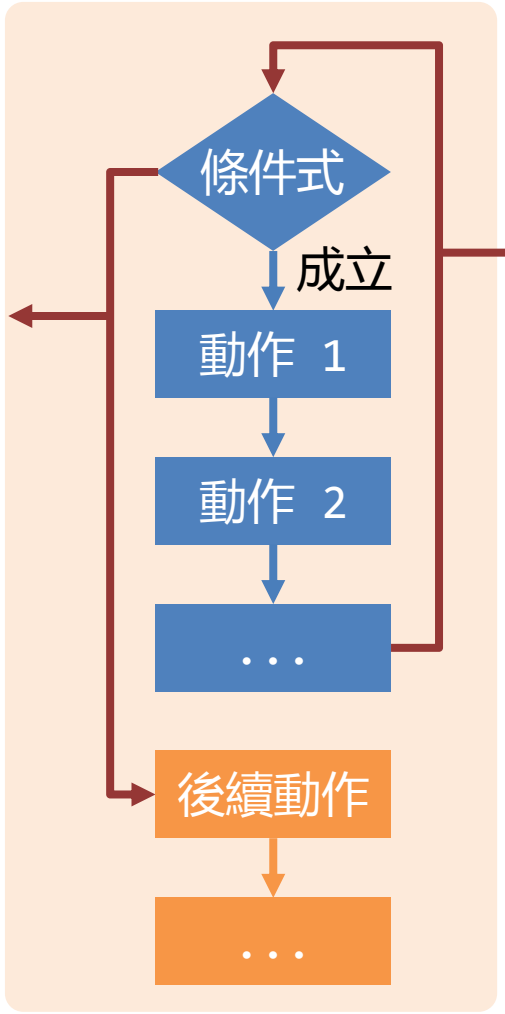
閃爍 LED

Python 的 while 迴圈

把前個範例改成閃爍 LED

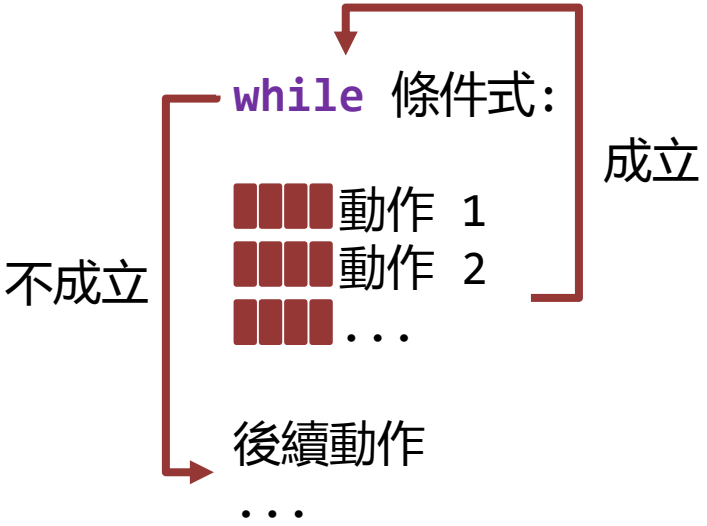


我們需要可**重複相同動作**的機制在程式中稱**迴圈**



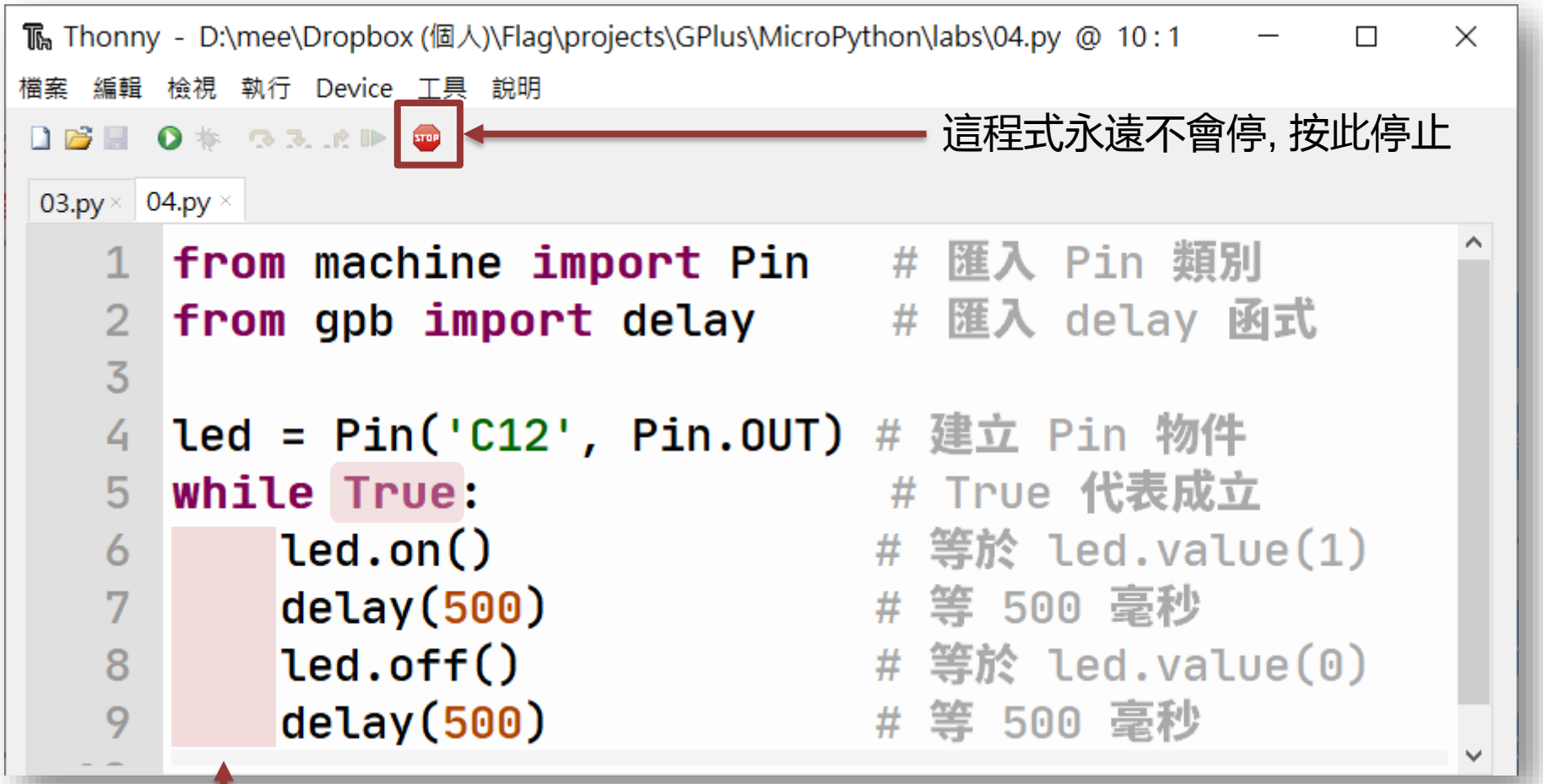
條件式運算不成立(**False**)，就直接執行後續動作

條件式運算結果如果是 **True** 表示條件成立，則執行迴圈內部的敘述

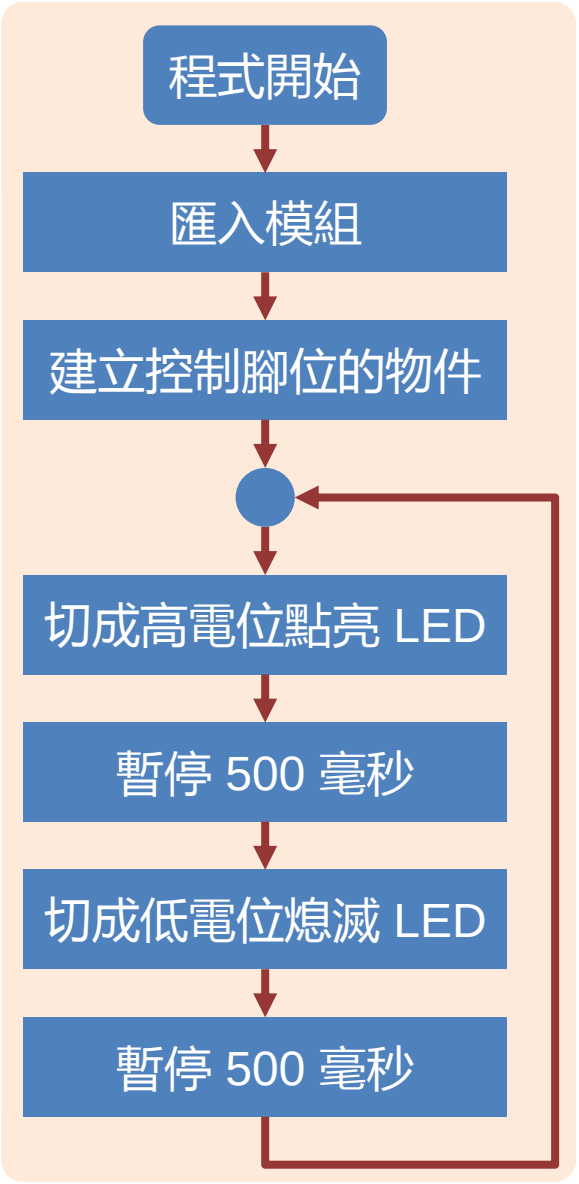


動手做 閃爍LED燈

再次提醒**大小寫不一樣**，while 那列**結尾有冒號**



上一列按 Enter 會自動幫你空

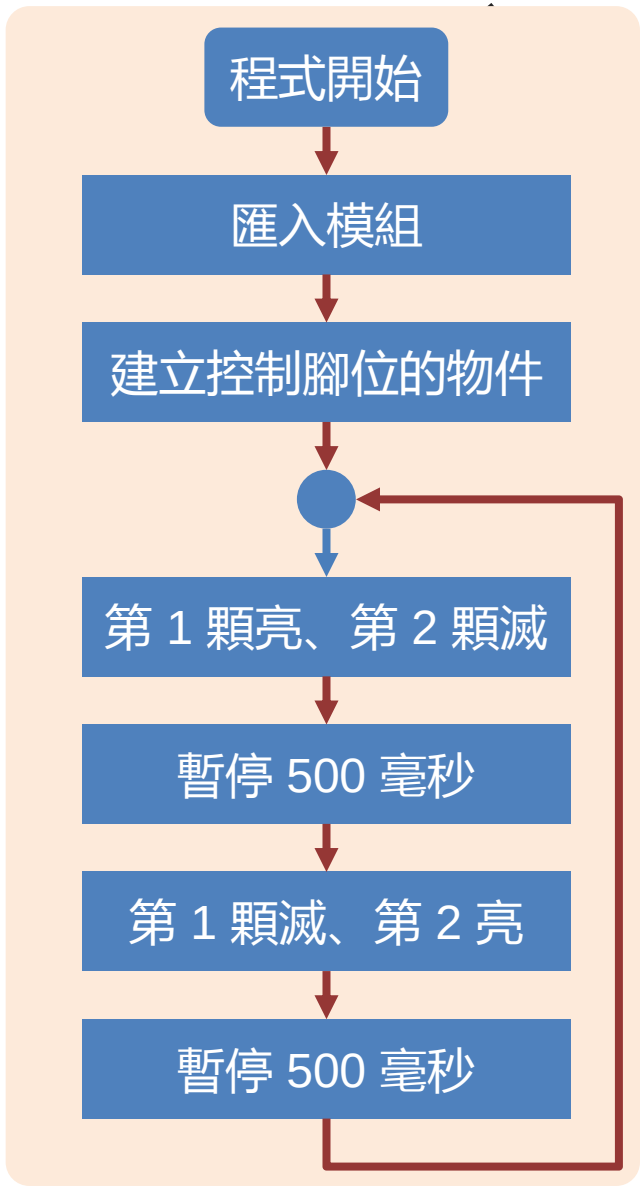


動手做

兩顆 LED 交互閃

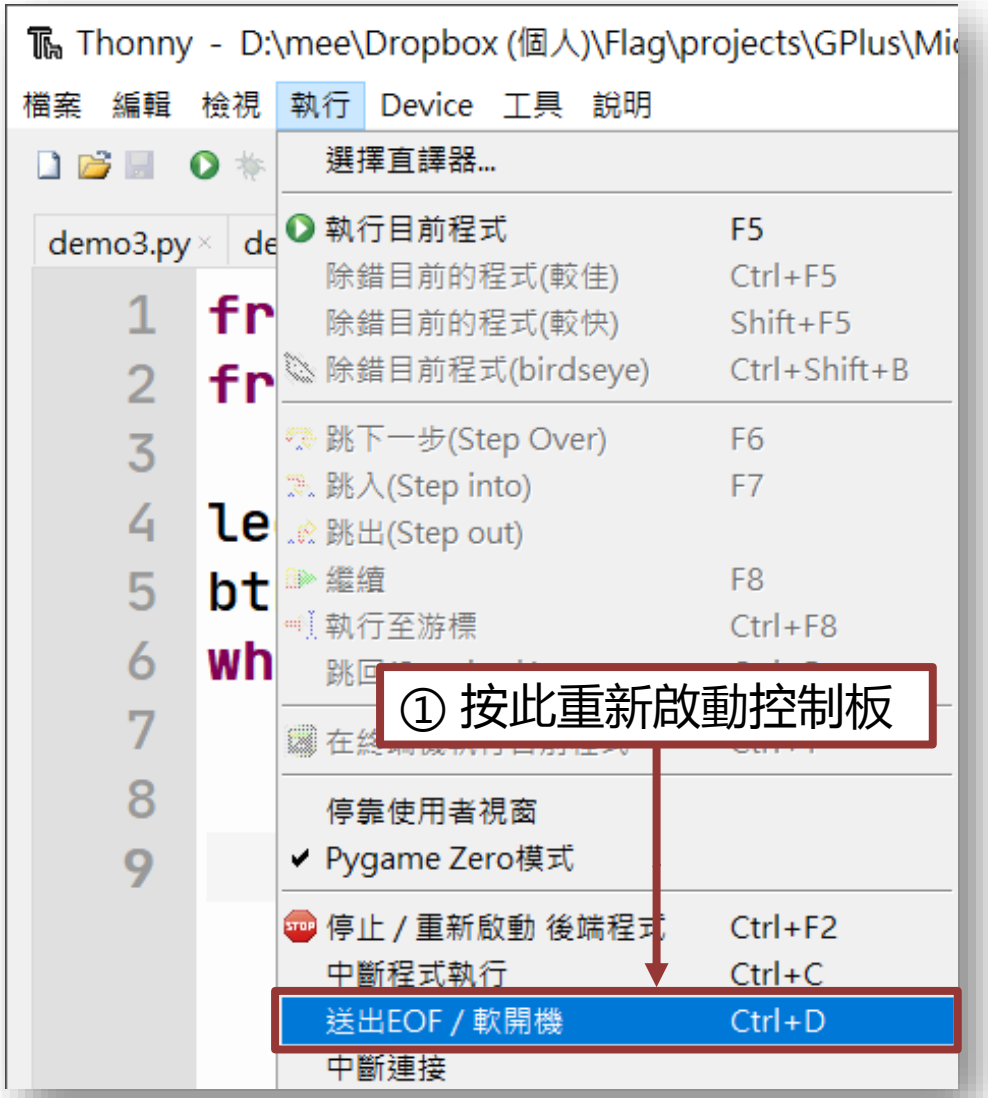
可參考sample code_led1.py

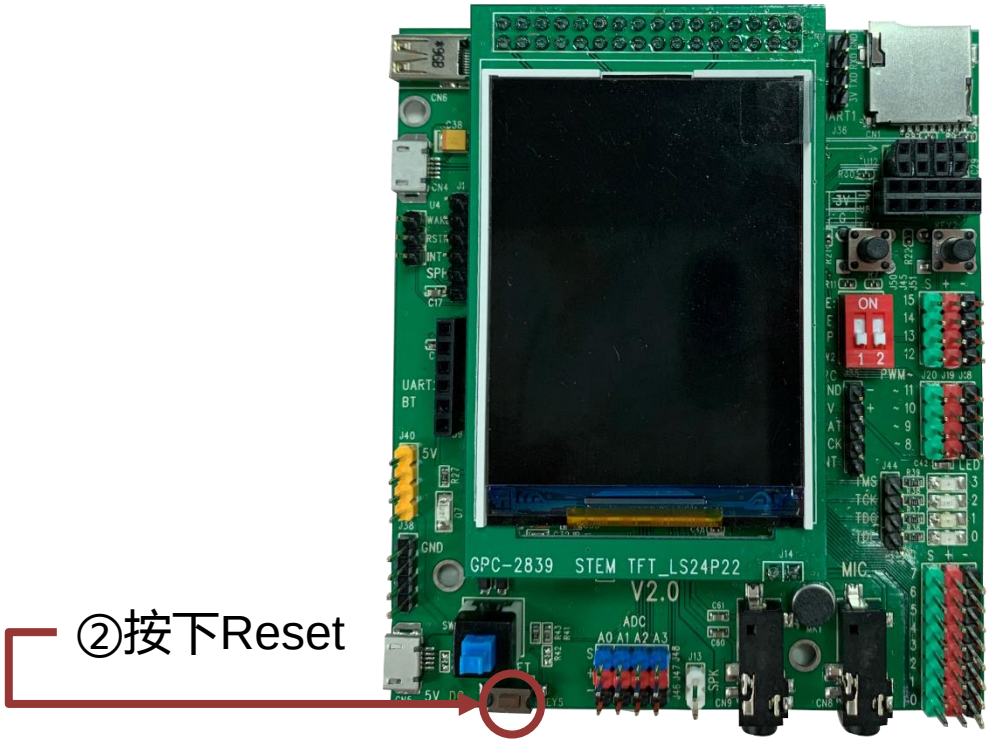
```
1  from machine import Pin     # 匯入Pin類別
2  from gpb import delay       # 匯入delay函式
3
4  led1 = Pin('C12',Pin.OUT) # 建立Pin物件
5  led2 = Pin('C13',Pin.OUT) # 建立Pin物件
6
7  while True:                 # True代表成立
8      led1.on()               # 點亮led1
9      led2.off()              # 熄滅led2
10     delay(500)              # 等500毫秒
11     led1.off()              # 熄滅led1
12     led2.on()               # 點亮led2
13     delay(500)              # 等500毫秒
```



疑難排解

執行後好像沒有反應？





5

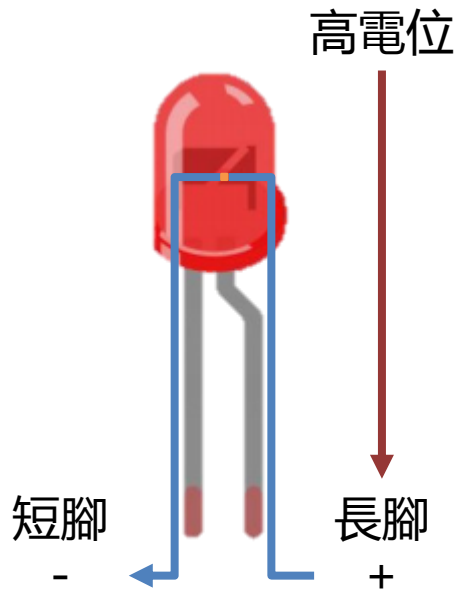
PART

點亮自己的 LED

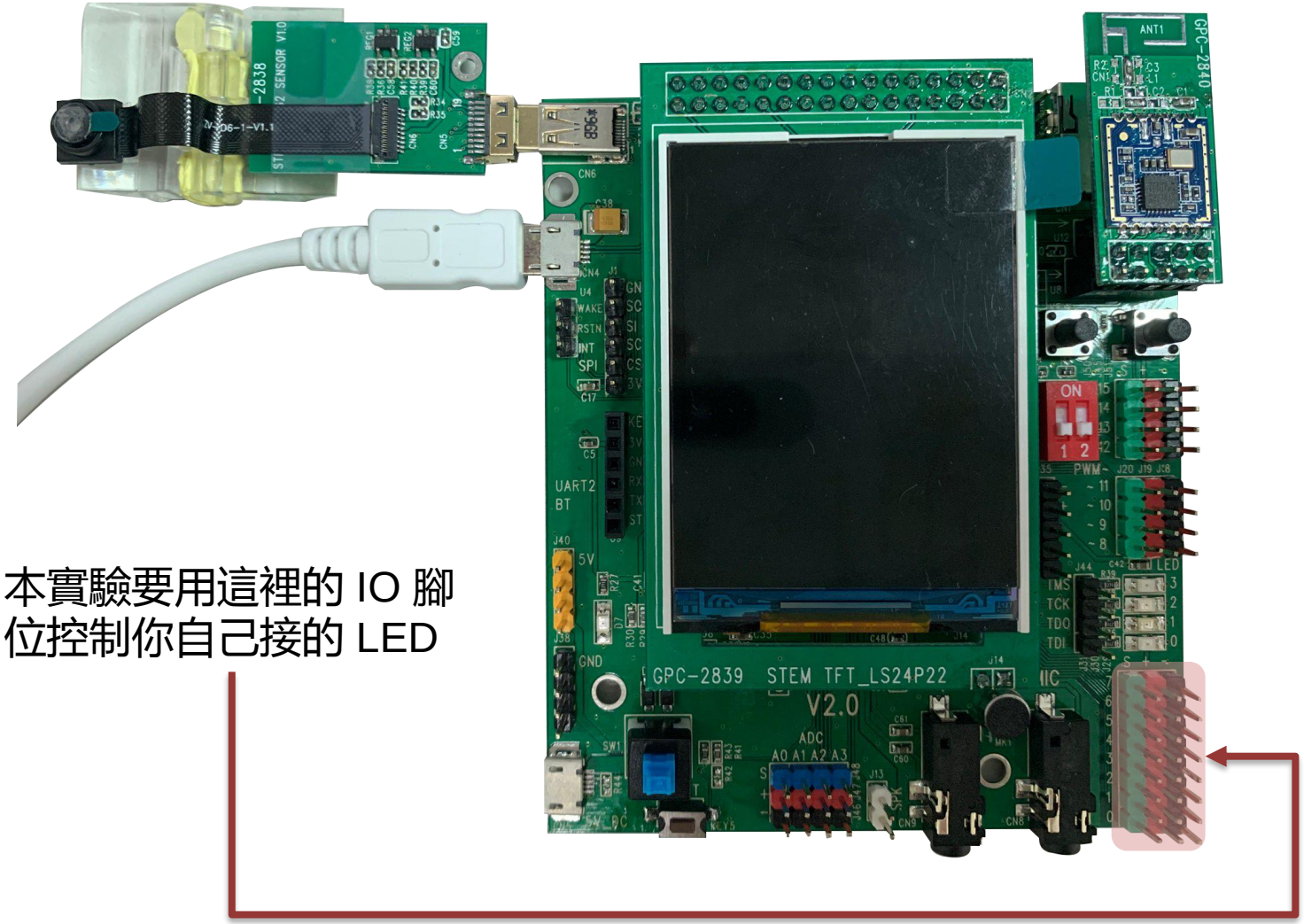
動手做：認識 LED、麵包板、杜邦線

原理

發光二極體 - LED



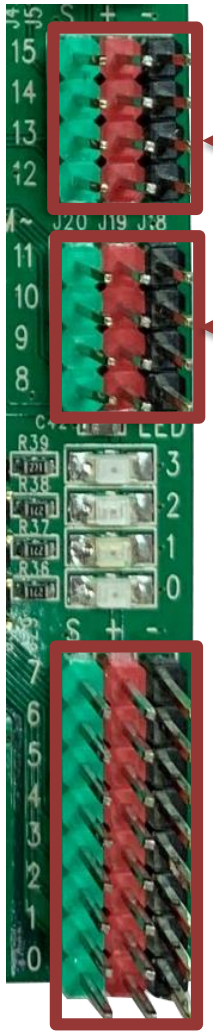
LED 具有**方向性**
跟推門/拉門一樣
方向對了才會亮



本實驗要用這裡的 IO 腳
位控制你自己接的 LED

原理

輸出入(I / O)腳位



IO15	E7	3.3V/ 5V	GND
IO14	E6		
IO13	E5		
IO12	E4		

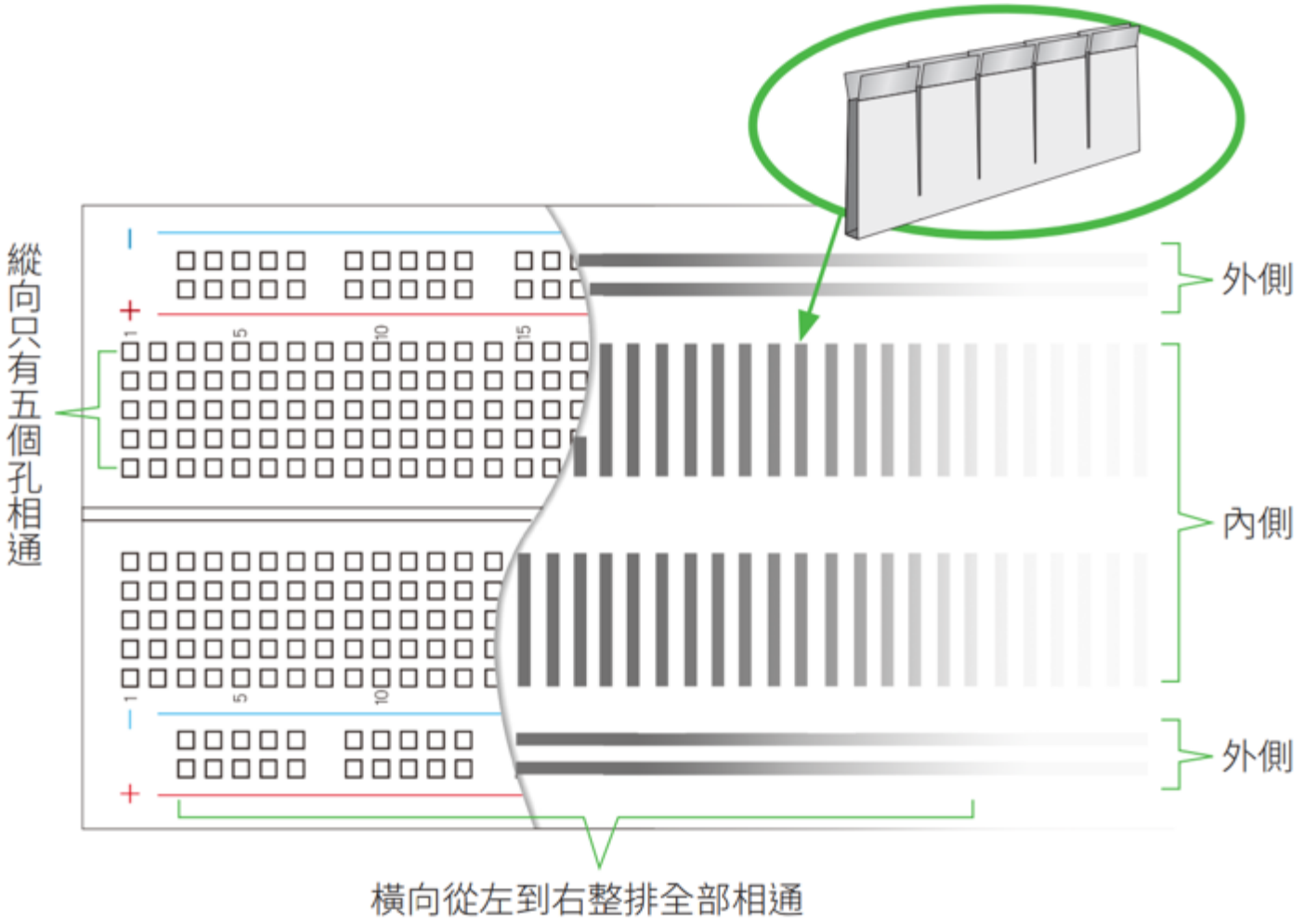
PWM11	E3	3.3V/ 5V	GND
PWM10	E2		
PWM9	E1		
PWM8	E0		

本實驗會使用 F3 控制外接的 LED

IO7	F3		
IO6	F2	3.3V/ 5V	GND
IO5	F1		
IO4	F0		
IO3	C3		
IO2	C2		
IO1	C1		
IO0	C0		

原理

麵包板與杜邦線



杜邦線每條**撕開**使用
顏色僅供識別**無差異**

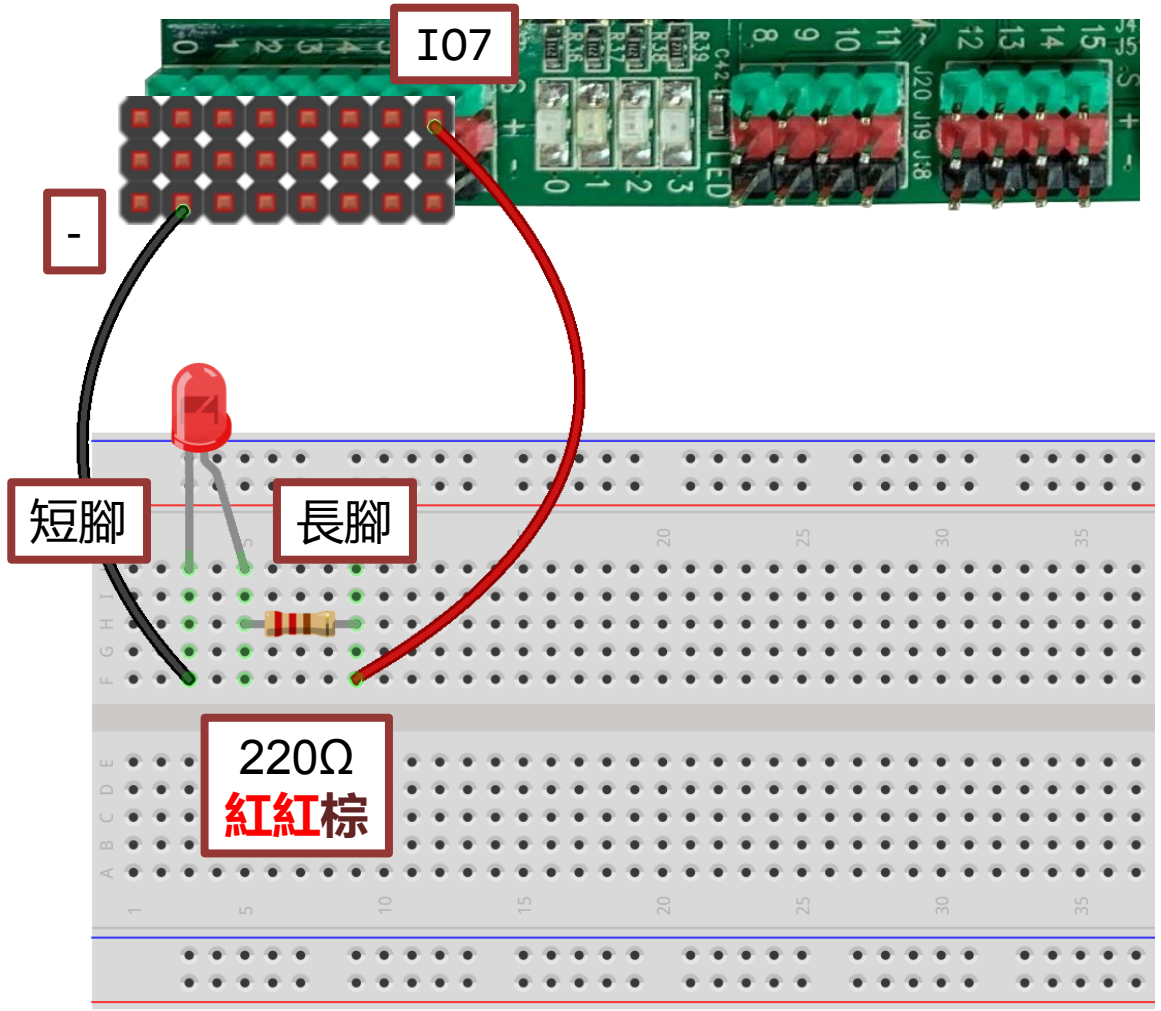
原理

這個水車要怎樣才會轉？

左邊小水車放到右邊水庫下可能就解體了，所以需要幫小水車加上防護措施避免毀損

動手做

接線



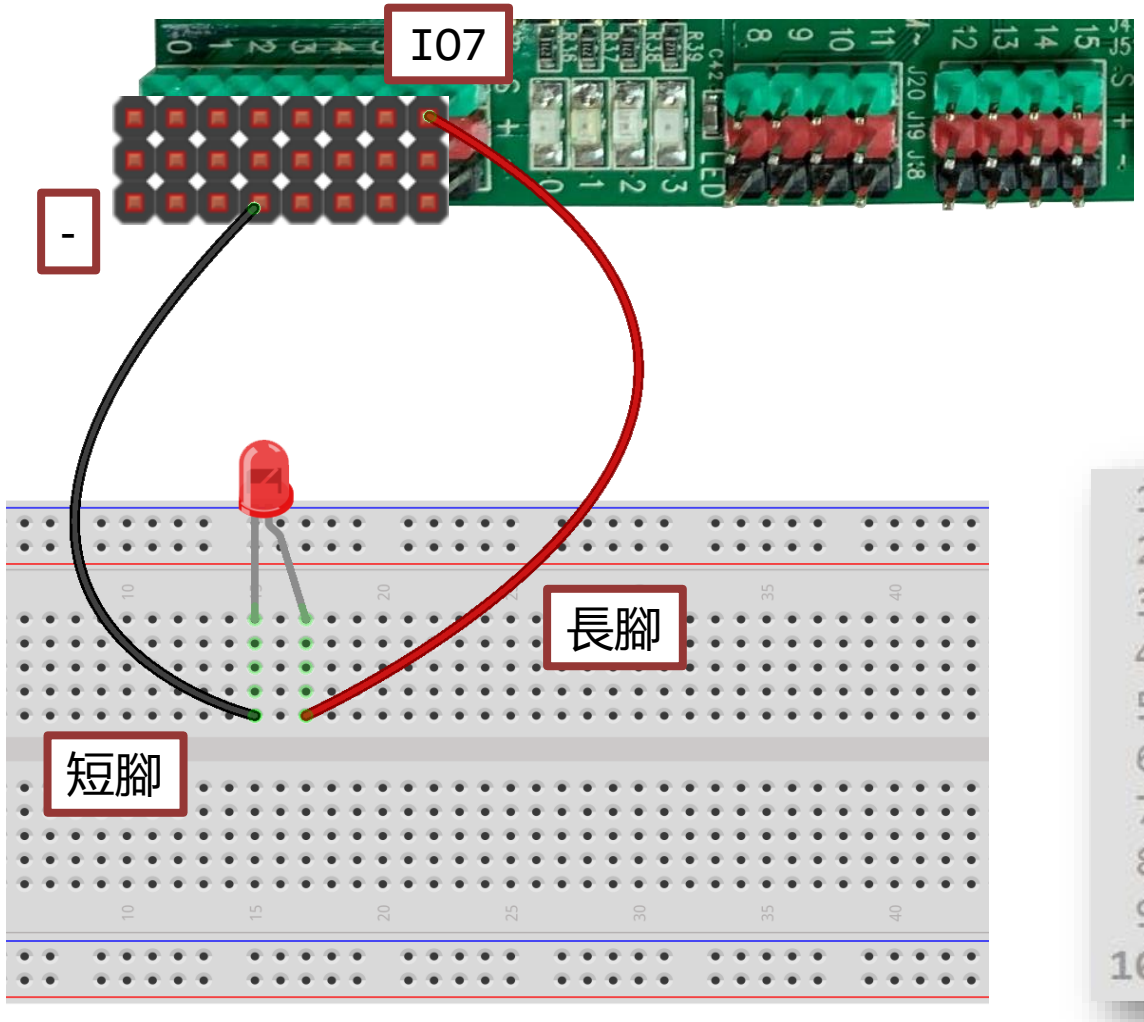
接線時請都先把**有針的那一頭**插好
避免**針腳亂晃**意外接觸到其他針腳

```
>>> from machine import Pin
>>> led = Pin('F3',Pin.OUT)
>>> led.on()
>>> led.off()
```

動手做

更改腳位名稱

可參考sample code_led.py

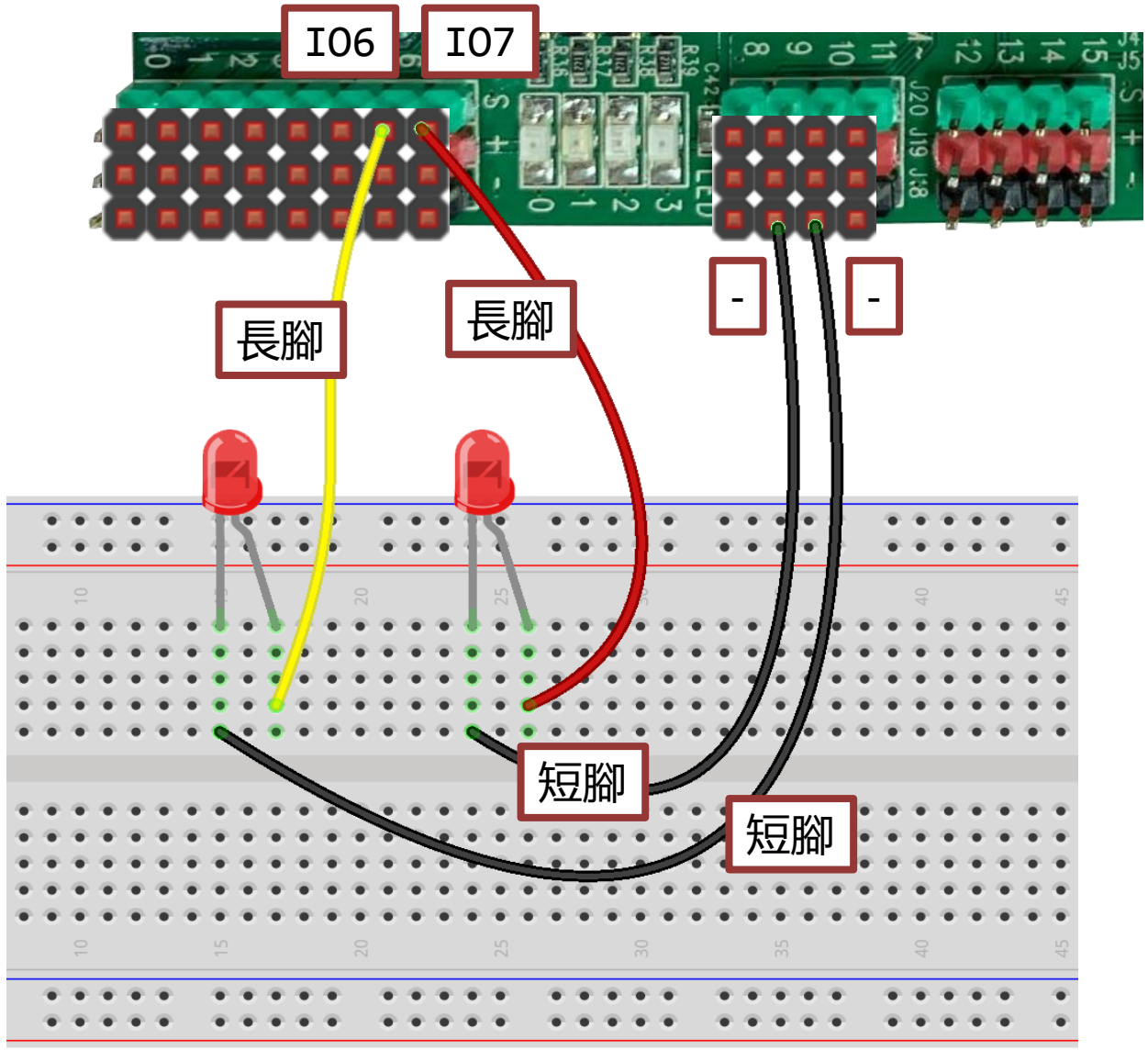


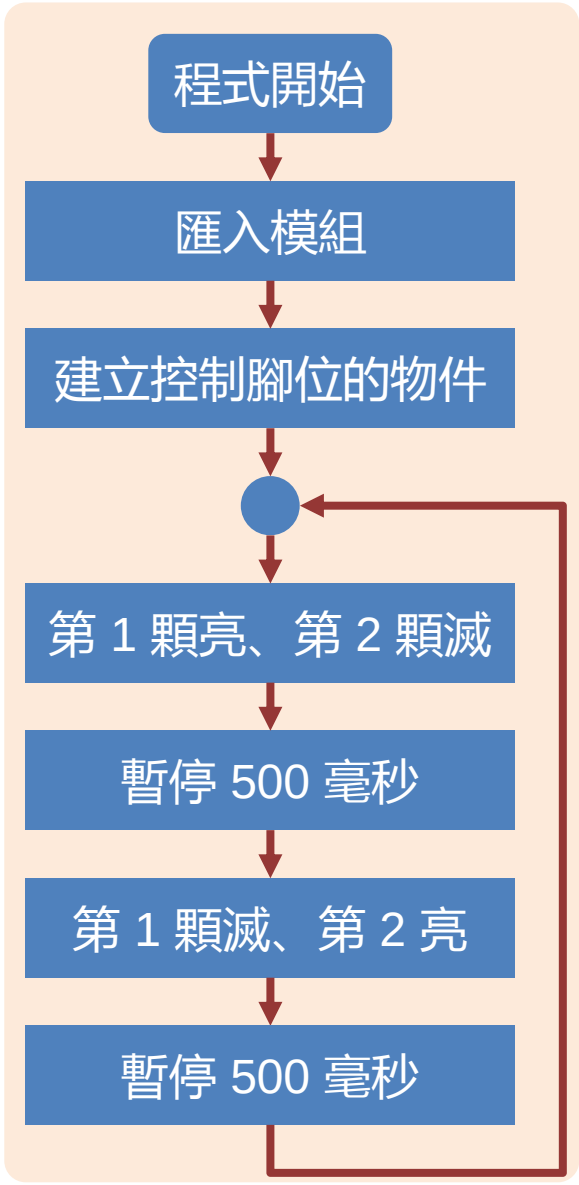
IO0	IO1	IO2	IO3	IO4	IO5	IO6	IO7
C0	C1	C2	C3	F0	F1	F2	F3
3.3V / 5V							
GND							

```
1  from machine import Pin      # 匯入 Pin 類別
2  from gpb import delay        # 匯入 delay 函式
3
4  led = Pin('F3',Pin.OUT)      # 建立 Pin 物件
5
6  while True:                  # True 代表成立
7      led.on()                 # 等於 led.value(1)
8      delay(500)               # 等 500 毫秒
9      led.off()
10     delay(500)
```

動手做

兩顆 LED 交互閃



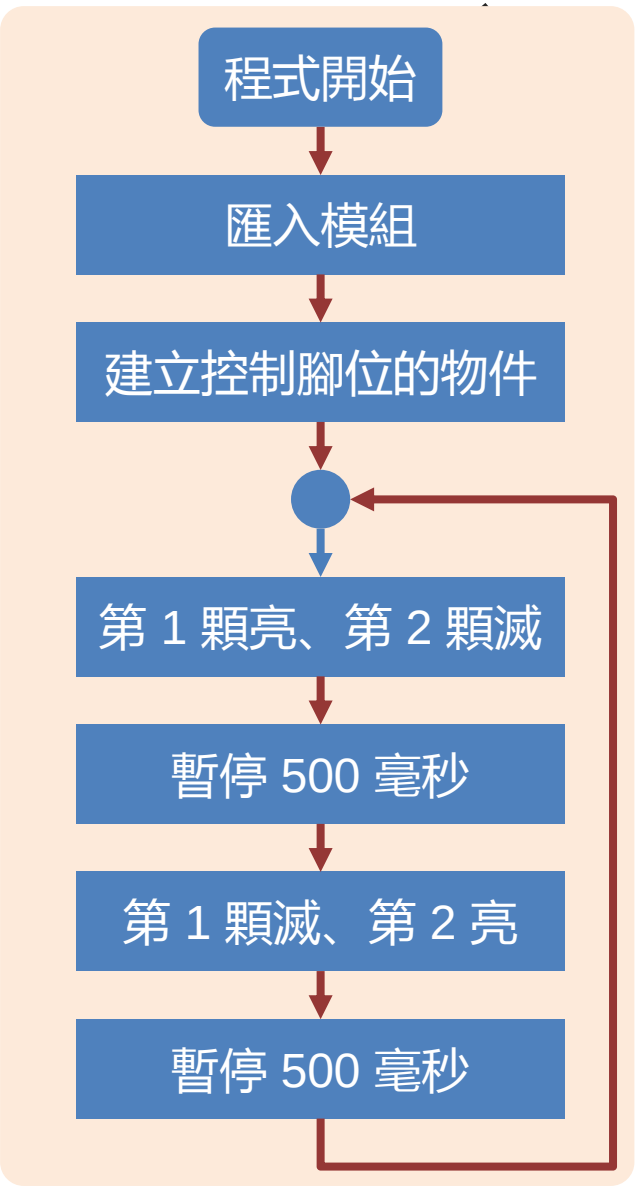


動手做

兩顆 LED 交互閃

可參考sample code_led1.py

```
1  from machine import Pin     # 匯入 Pin 類別
2  from gpb import delay       # 匯入 delay 函式
3
4  led1 = Pin('F3',Pin.OUT)    # 建立 Pin 物件
5  led2 = Pin('F2',Pin.OUT)    # 建立 Pin 物件
6
7  while True:                 # True 代表成立
8      led1.on()               # 等於 led1.value(1)
9      led2.off()              # 等於 led2.value(0)
10     delay(500)              # 等 500 毫秒
11     led1.off()
12     led2.on()
13     delay(500)
```



6

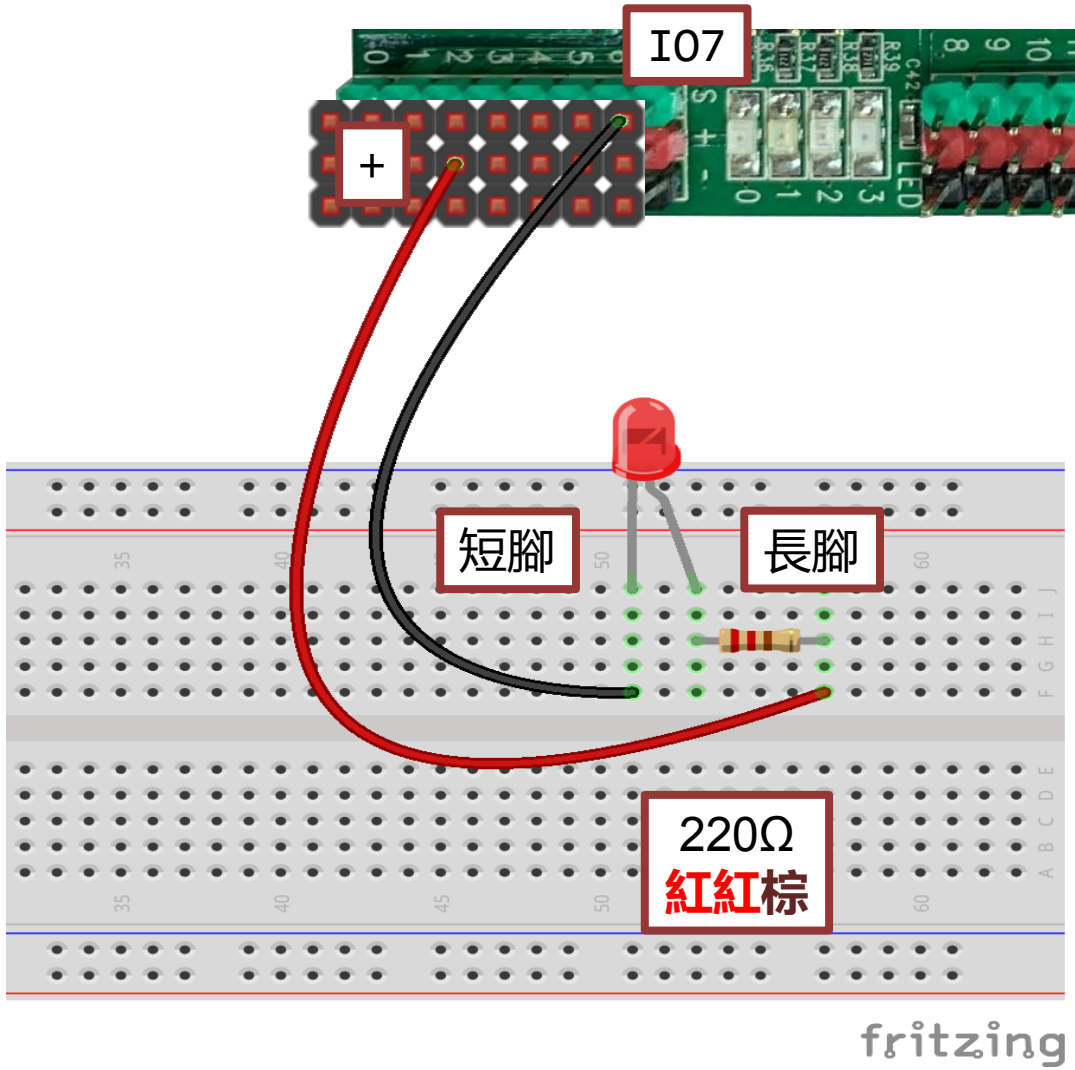
PART

讓LED變得更亮

直接從電源汲取夠大的電流

動手做

接線



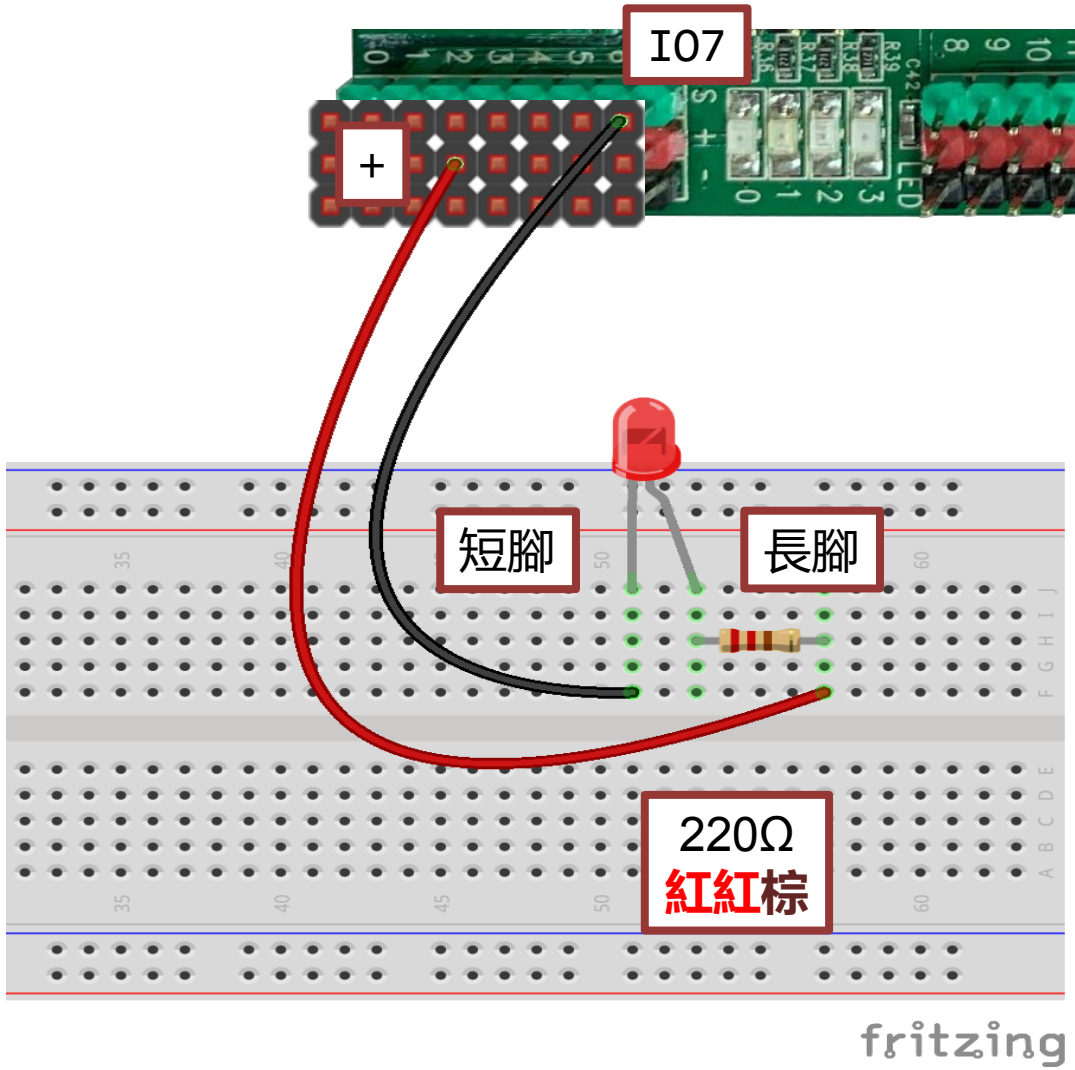
由於 IO 腳輸出電流有限制不接限流電阻
也不會有關係若**不夠亮可反接從電源供電**

IO0	IO1	IO2	IO3	IO4	IO5	IO6	IO7
C0	C1	C2	C3	F0	F1	F2	F3
3.3V / 5V							
GND							

本實驗會使用 F3 控制外接的 LED

動手做

修改程式



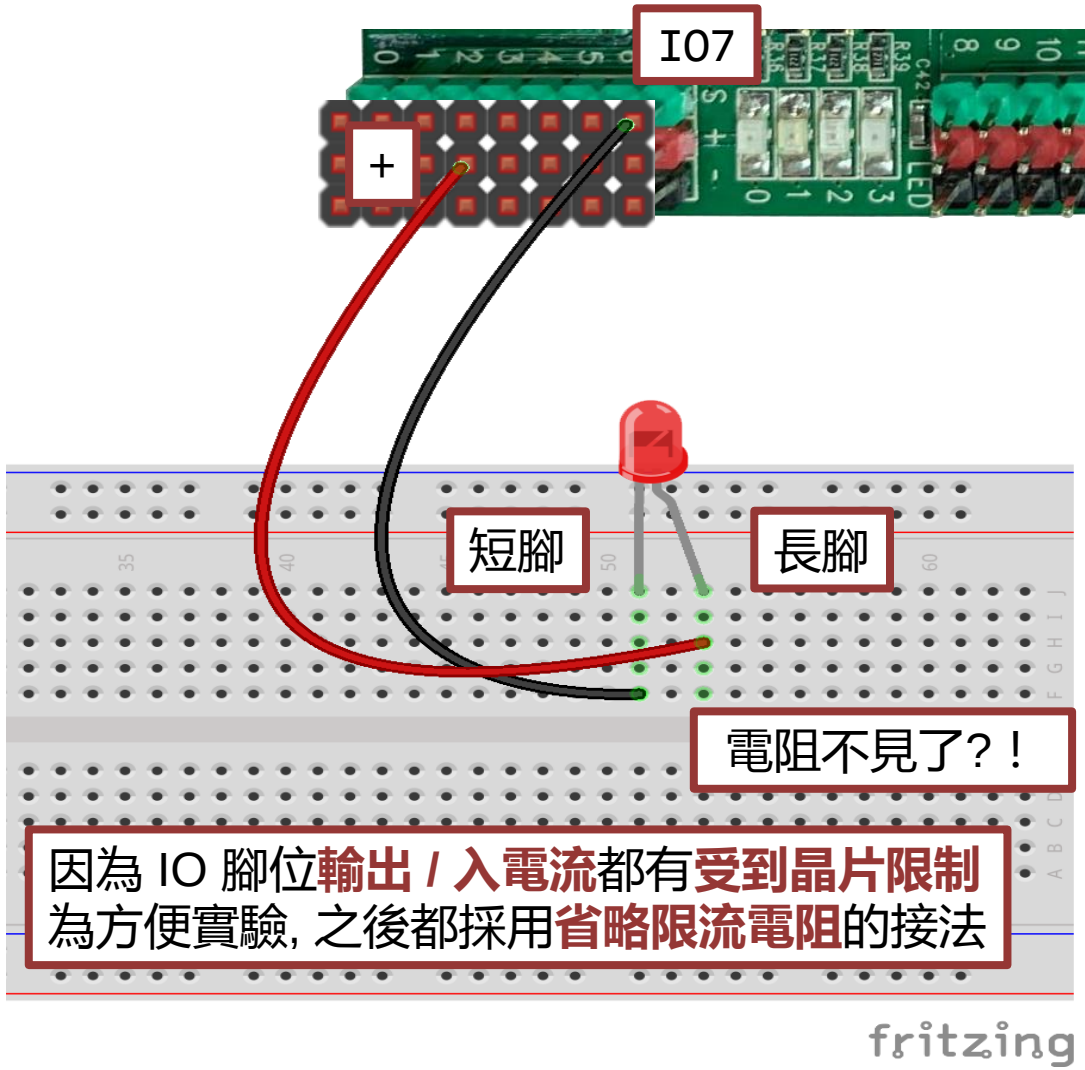
由於現在控制的是 LED **短腳**所以變成**低電位**會**點亮** LED但**高電位**反而會讓 LED **熄滅**

```
01: from machine import Pin   # 匯入 Pin 類別
02: from gpb import delay     # 匯入 delay 函式
03:
04: led = Pin('F3', Pin.OUT) # 建立 Pin 物件
05: while True:               # True 代表成立
06:     led.off()             # 低電位亮燈
07:     delay(500)
08:     led.on()              # 高電位熄燈
09:     delay(500)            # 等 500 毫秒
```

off() 亮燈會讓你很不自在嗎？

動手做

修改程式



Pin.OUT_INV (INVserse) 模式可以讓 IO 腳位真正輸出時**反向**

```
01: from machine import Pin   # 匯入 Pin 類別
02: from gpb import delay     # 匯入 delay 函式
03:
04: led = Pin('F3', Pin.OUT_INV) # 建立 Pin 物件
05: while True:                  # True 代表成立
06:     led.on()                 # 低電位亮燈
07:     delay(500)               # 等 500 毫秒
08:     led.off()                # 高電位熄燈
09:     delay(500)               # 等 500 毫秒
```

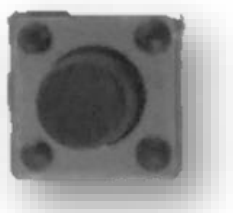
7

PART

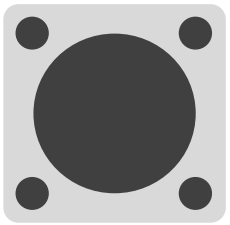
用按鈕控制LED

數位輸入 (Digital Input)

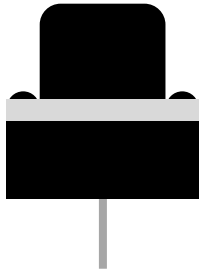
認識按壓開關



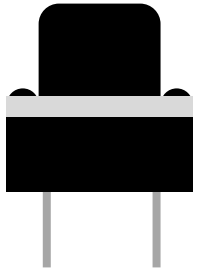
鳥瞰圖

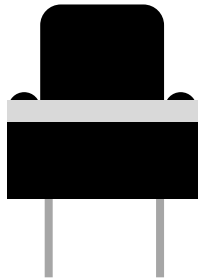


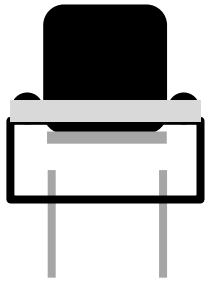
側視圖



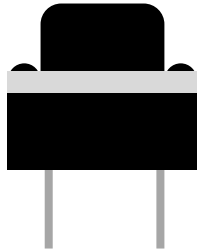
側視圖

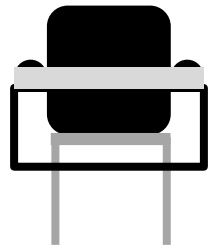






沒有按下時不導通

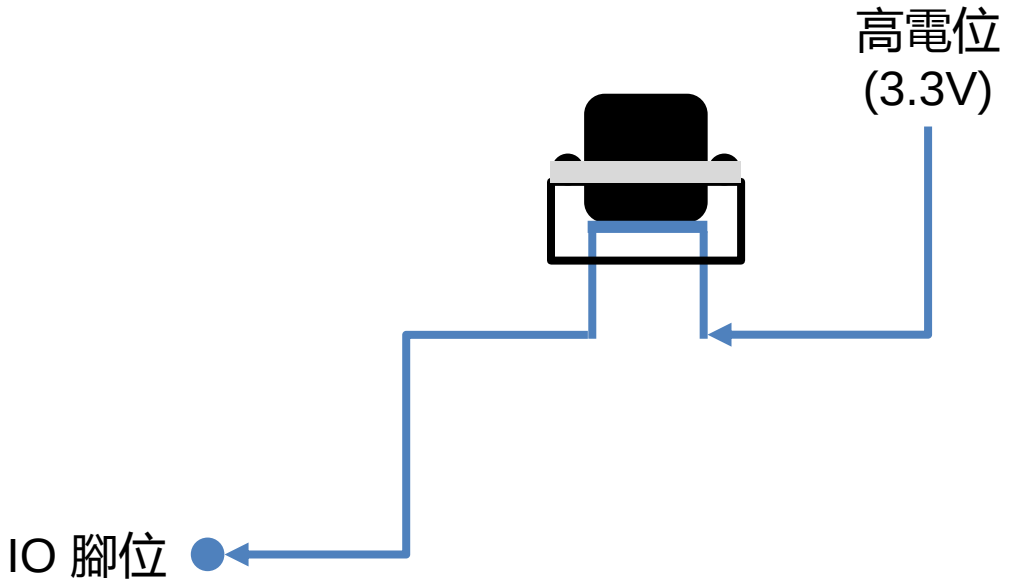




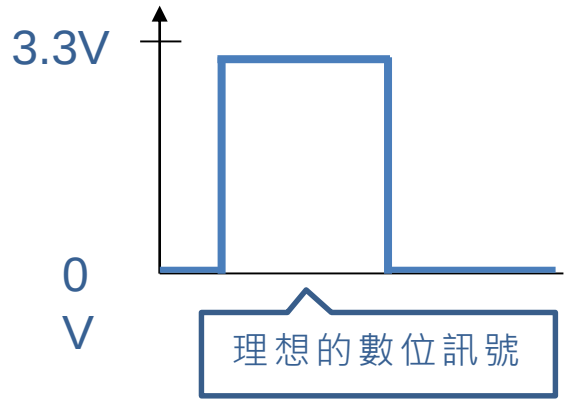
按下時導通

原理

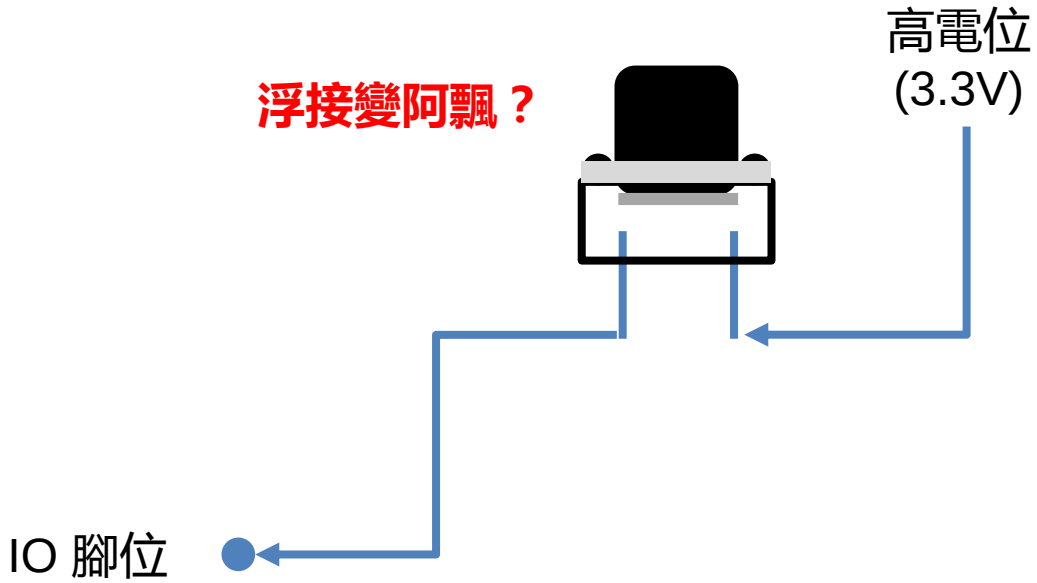
開關這樣接的話...？



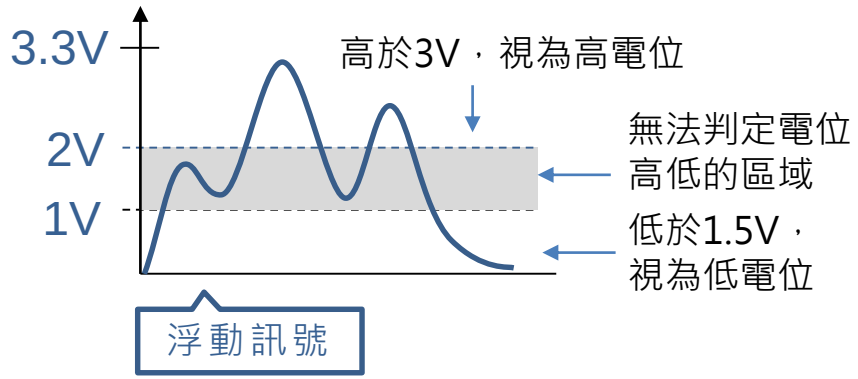
按下按鈕時會偵測到 3.3V



浮接變阿飄？



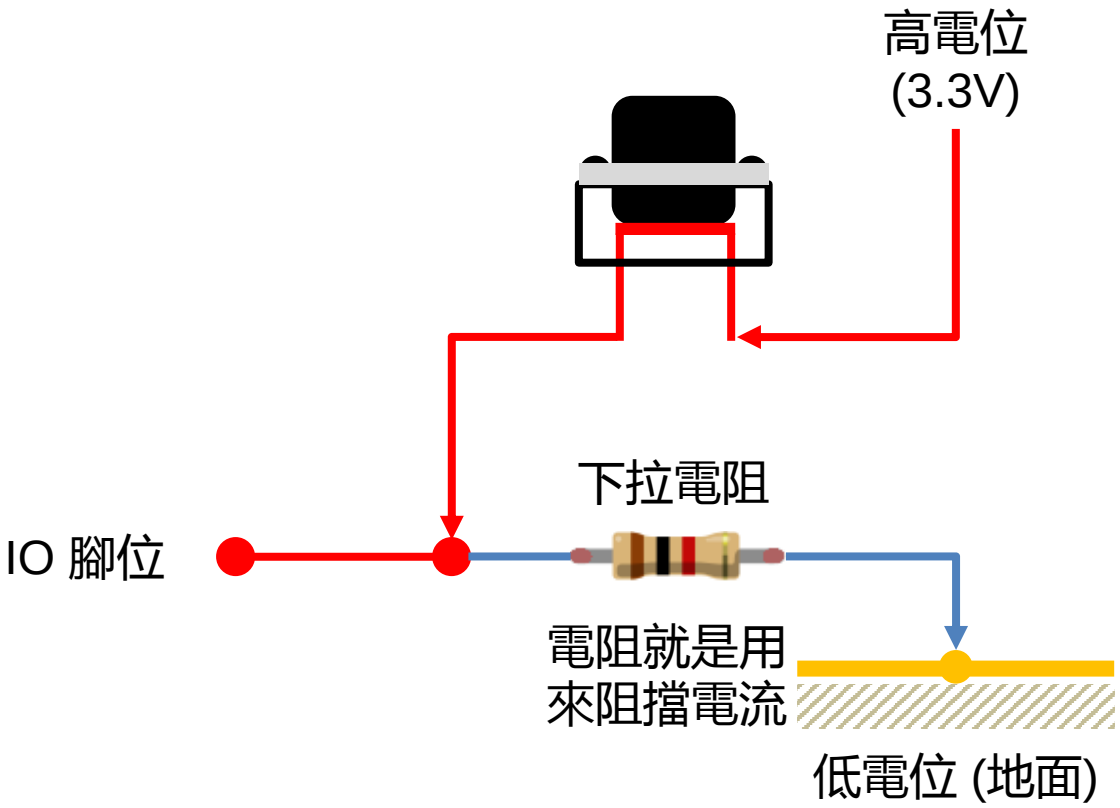
沒按下按鈕時右邊是空接會受雜訊影響電位不明確



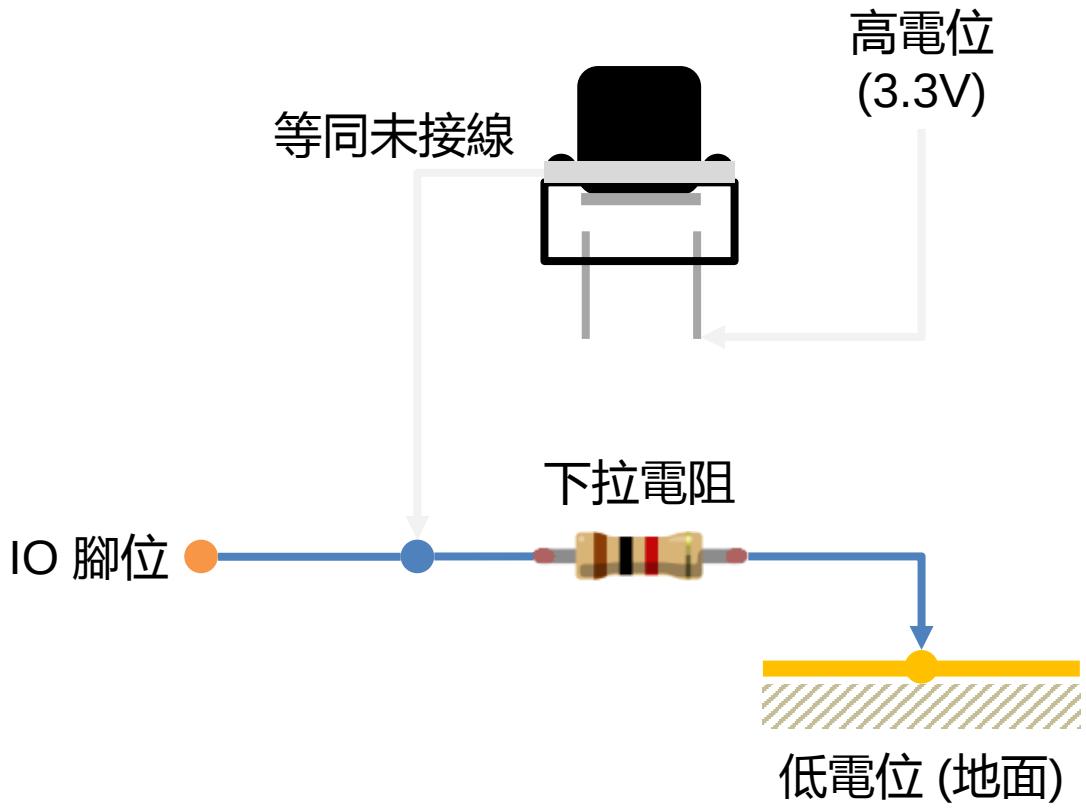
原理

正確接法 (一)

按下開關時會偵測到高電位 (1)

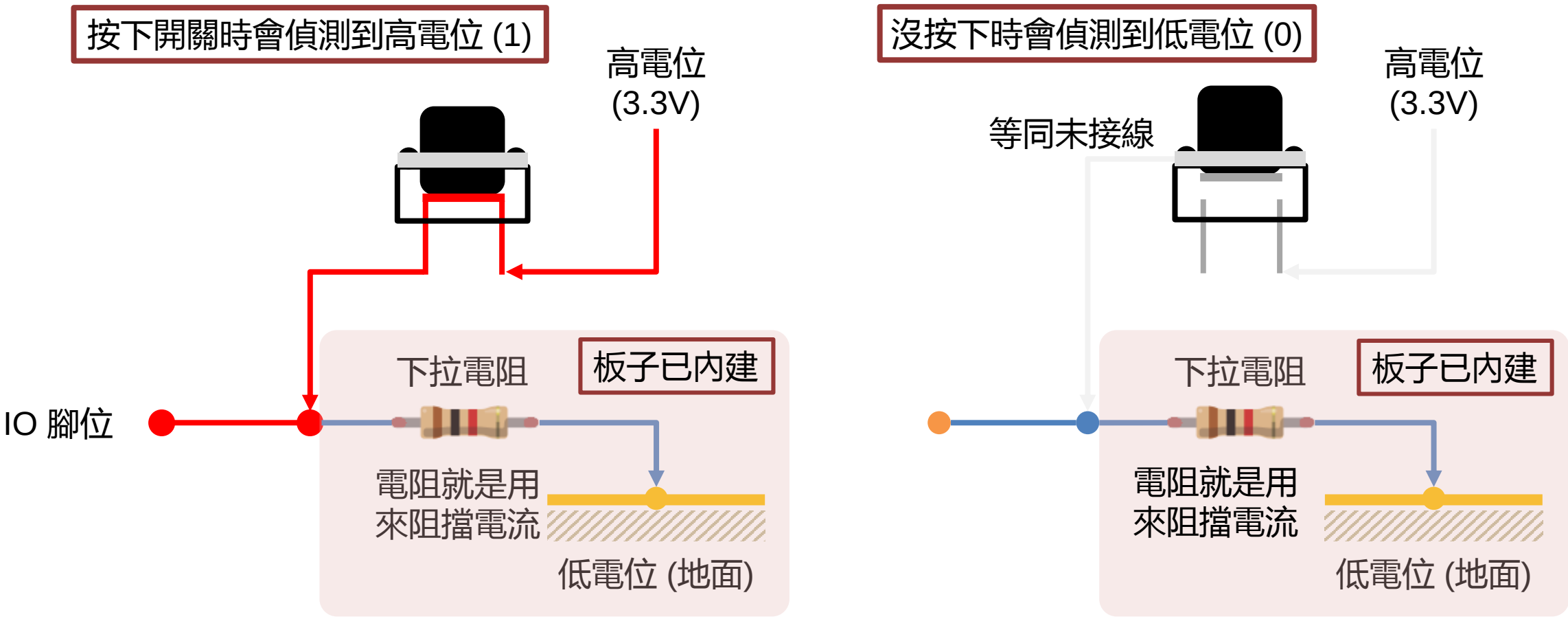


沒按下時會偵測到低電位 (0)



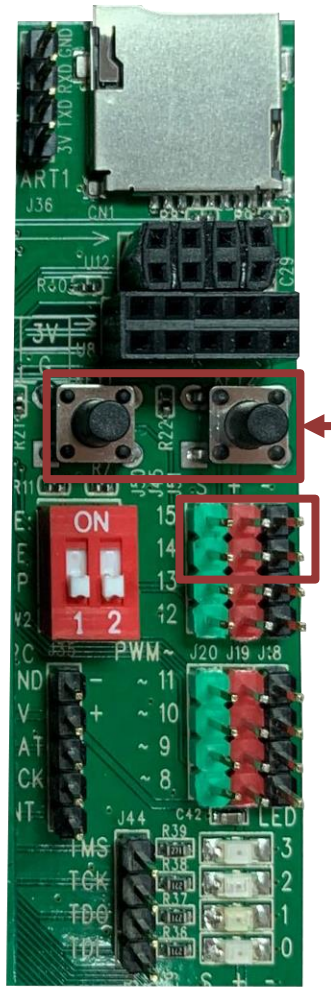
原理

正確接法 (二)


按下開關時會偵測到高電位 (1)
高電位
(3.3V)
下拉電阻
板子已內建
IO 腳位
電阻就是用
來阻擋電流
低電位 (地面)
沒按下時會偵測到低電位 (0)
等同未接線
高電位
(3.3V)
下拉電阻
板子已內建
電阻就是用
來阻擋電流
低電位 (地面)

動手做

Genio Py.上的按鍵



KEY2	E7	3.3V	GND
KEY1	E6		

```
>>> from machine import Pin
>>> btn = Pin('E6',Pin.IN)
>>> btn.value()
0
>>> btn.value()
1
>>> btn = Pin('E7',Pin.IN)
>>> btn.value()
0
>>> btn.value()
1
```

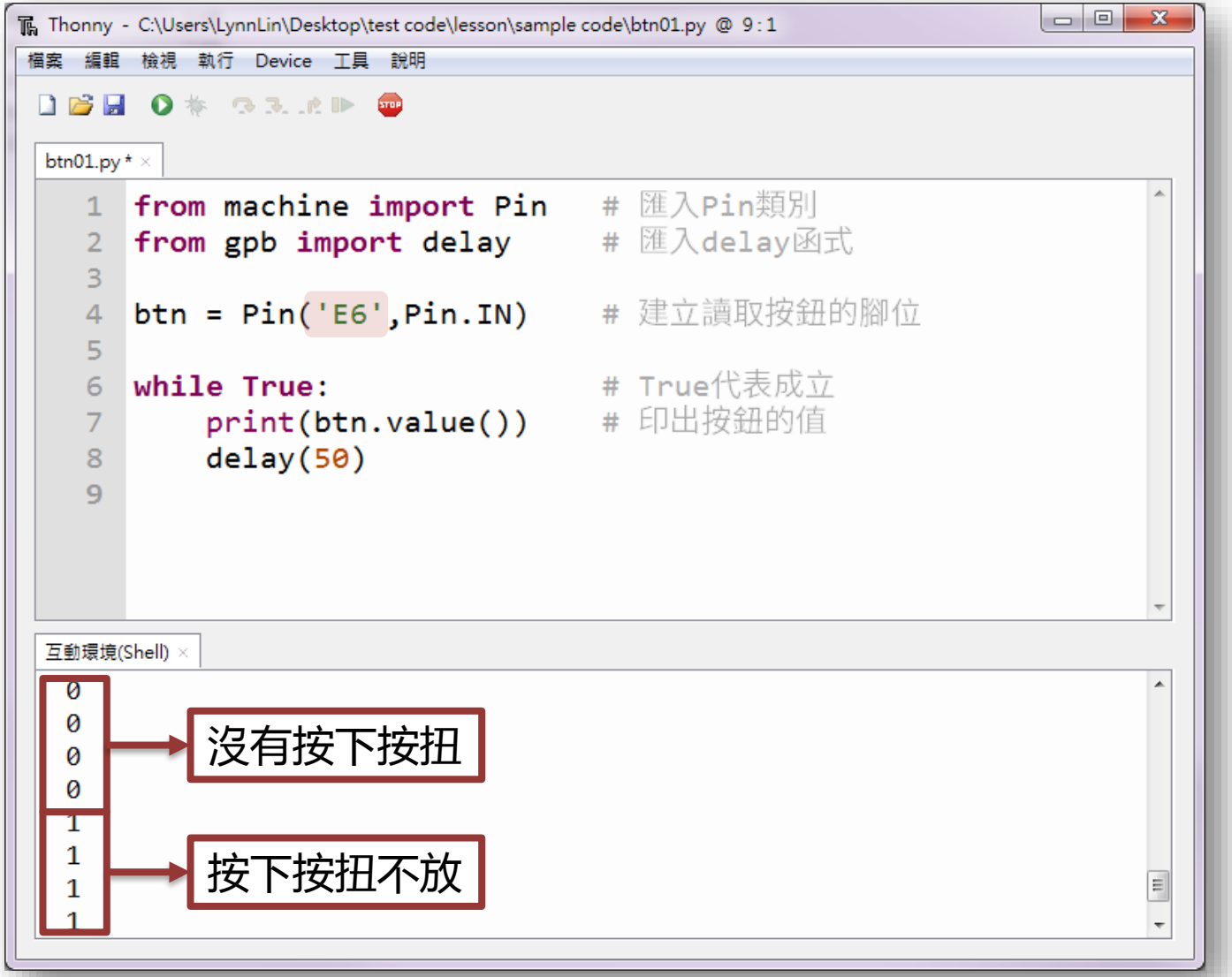
按下按鍵值為1

按下按鍵值為1

動手做

寫程式觀察按鈕狀態

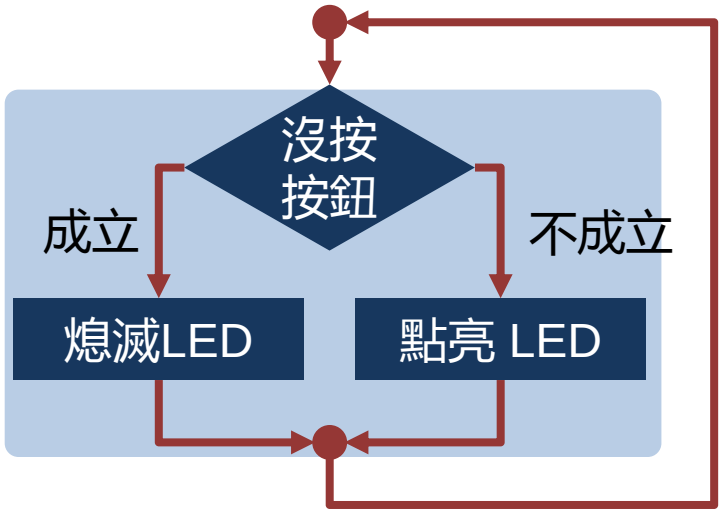
可參考sample code: 04_btn.py



動手做 用按鈕控制LED

可參考sample code : 05_btn.py

```
1  from machine import Pin      # 匯入Pin類別
2  from gpb import delay        # 匯入delay函式
3
4  led=Pin('C13',Pin.OUT)        # 建立控制 LED 的腳位
5  btn=Pin('E7',Pin.IN)         # 建立讀取按鈕的腳位
6
7  while True:                  # True 代表成立
8      if btn.value() == 0:     # 如果沒按下按鈕
9          led.value(0)         # 熄滅LED
10     else:                    # 否則
11         led.value(1)         # 點亮LED
```




```
if 條件判斷式:
    成立時要做的動作
    ...
else:
    不成立時要做的動作
    ...
```

注意 if else 尾端的**冒號**以及**縮排**

== 會判斷左邊的值是否**等於**右邊
判斷式依此得到 **True** 或是 **False**

動手做

將程式簡化如下

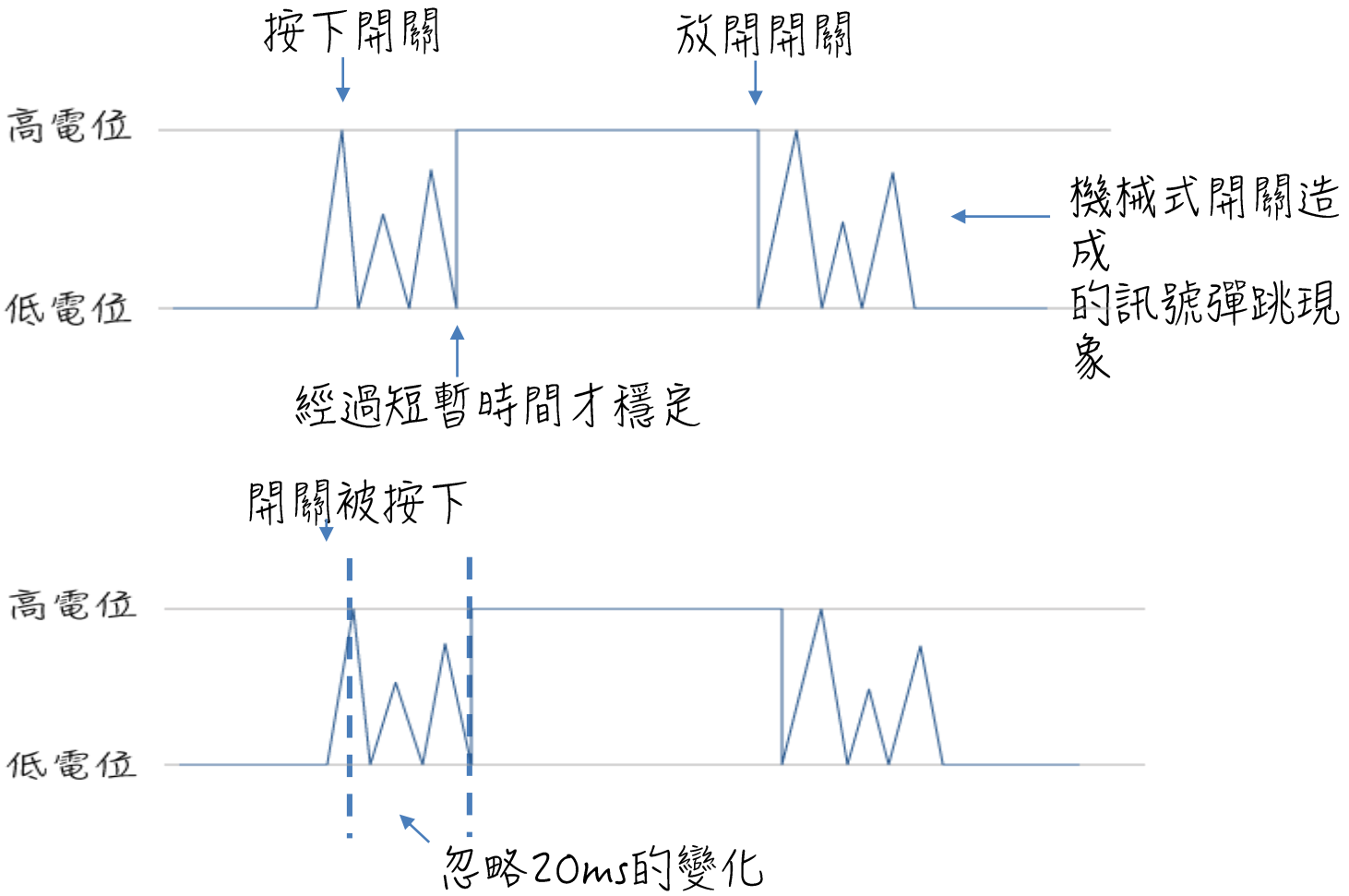
可參考sample code : 06_btn.py

由於按鈕的狀態值與 LED 的設定值一致
沒按->0　　0->熄滅 LED
按下->1　　1->點亮 LED
所以可直接用按鈕狀態值設定 LED 亮滅

```
1  from machine import Pin          # 匯入Pin類別
2  from gpb import delay            # 匯入delay函式
3
4  led=Pin('C13',Pin.OUT)           # 建立控制 LED 的腳位
5  btn=Pin('E7',Pin.IN)             # 建立讀取按鈕的腳位
6
7  while True:                      # True 代表成立
8      led.value(btn.value())       # 依照按鈕亮滅
```

原理

訊號彈跳現象



按下或放開按鈕時會有**彈跳狀況**

最簡單解法就是等**彈跳結束重新驗證**

圖片來源：旗標科技 超圖解 Python 物聯網

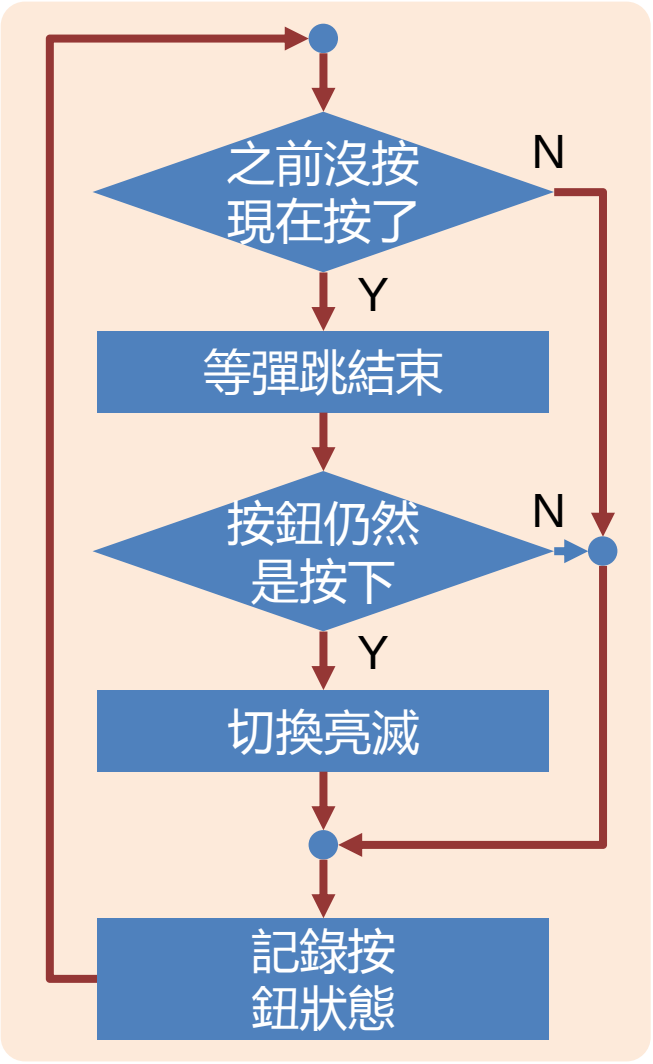
動手做

按一下亮，按一下滅

可參考sample code: 07_btn.py

```
1  from machine import Pin                  # 匯入Pin類別
2  from gpb import delay                    # 匯入delay函式
3
4  led=Pin('C13',Pin.OUT)                   # 建立控制 LED 的腳位
5  led.value(0)                             # 熄滅 LED
6  btn=Pin('E7',Pin.IN)                     # 建立讀取按鈕的腳位
7  prev = 0                                 # 上一次的按鈕狀態
8  led_state = 0                            # 預設熄燈
9
10 while True:                              # True 代表成立
11     curr = btn.value()                   # 取得按鈕狀態
12     if prev == 0 and curr == 1:          # 如果前次沒按且現在按下
13         delay(20)                        # 等 20 毫秒彈跳結束
14         curr = btn.value()               # debounce連續讀值兩次
15         if curr == 1:                    # 仍然是按下的狀態
16             led_state=1-led_state        # 1 -> 0 ; 0 -> 1
17             led.value(led_state)         # 將led_state的值寫入LED
18     prev = curr                          # 記錄本次狀態
```

如果原本是 0 -> 1 - 0 -> 1
如果原本是 1 -> 1 - 1 -> 0



自訂函式

定義一個函式為BMI

```
>>> def BMI(h,w):
        bmi=w/(h**2)
        print(bmi)

>>> BMI(1.8,75)
 23.14815
```

將身高和體重代入函式便會自動計算BMI

函式中沒有的變數會往外找

```
>>> cm2m=100
>>> def BMI(h,w):
        h=h/cm2m
        bmi=w/(h**2)
        print(bmi)

>>> BMI(180,75)
 23.14815
```

原理

使用外部名稱

```
>>> y=4
>>> def func(x):
        y=x+y
        print(y)
>>> func(2)
```

但在這裡就要取得 y 名牌對應物件的內容

這會導致在函式內部建立一個新名稱但還沒掛到物件上

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in func
NameError: local variable referenced before assignment
```

區域變數在命名前已經先被引用

要使用全域變數的y，就必須加入global

```
>>> y = 4
>>> def func(x):
        global y
        y = x + y
        print(y)

>>> func(2)
 6
>>> y
6
```

y 代表的物件變了

動手做

暗中默默工作的計時器

先定義一個每次執行就會
遞增 x 並列印數值的函式

```
>>> x = 1
>>> def counter(t):
        global x
        x = x + 1
        print(x)

>>> from gpb import delay,timer
>>> counter_timer = timer(1)
>>> counter_timer.init(freq=10,callback=counter)
>>> 2
3
4
5
6
7
8
9
10
```

匯入 timer 模組

建立 1 號計時器
共有 0~7 號 8 個

會立刻出現 >>> 符號
並依指定頻率列印序號

每秒10次

每 1 次中斷程式
就喚醒 counter

注意這裡沒有要執行函式
只是註冊函式的名稱所以
不用加上小括號

計時器可讓你做其他事
並依指定頻率中斷程式
達成好像**同時**作兩件事

動手做 用計時器避免按鈕彈跳

可參考sample code_btn05.py

```
from machine import Pin
from gpb import delay,timer                    # 匯入計時器類別

led = Pin('F3',Pin.OUT)                        # 建立控制LED的腳位
led.value(0)                                   # 熄滅LED
btn = Pin('C0',Pin.IN)                         # 建立讀取按鈕的腳位
prev = 0                                       # 上一次的按鈕狀態

def switch_led(t):                             # 計時喚醒時要執行的動作
    global prev                                # 使用外面的 prev 名稱
    curr = btn.value()                         # 讀取目前按鈕狀態
    if prev == 0 and curr == 1:                # 之前沒按現在按了
        led.value(1 - led.value())             # 切換 LED 亮按
    prev=curr                                  # 記錄按鈕狀態

led_timer = timer(1)                           # 建立 1 號計時器
led_timer.init(freq=50,callback=switch_led)    # 指定每秒喚醒 50 次, 也就是每 20 毫秒 1 次

while True:
    delay(1)                                   # 要有迴圈計時器才能運作
```

```
while True:
    curr = btn.value()
    if prev == 0 and curr == 1:
        delay(20)
        curr = btn.value()
        if curr == 1:
            if led_state == 1:
                led_state = 0
            else:
                led_state = 1
            led.value(led_state)
    prev = curr
```

8

PART

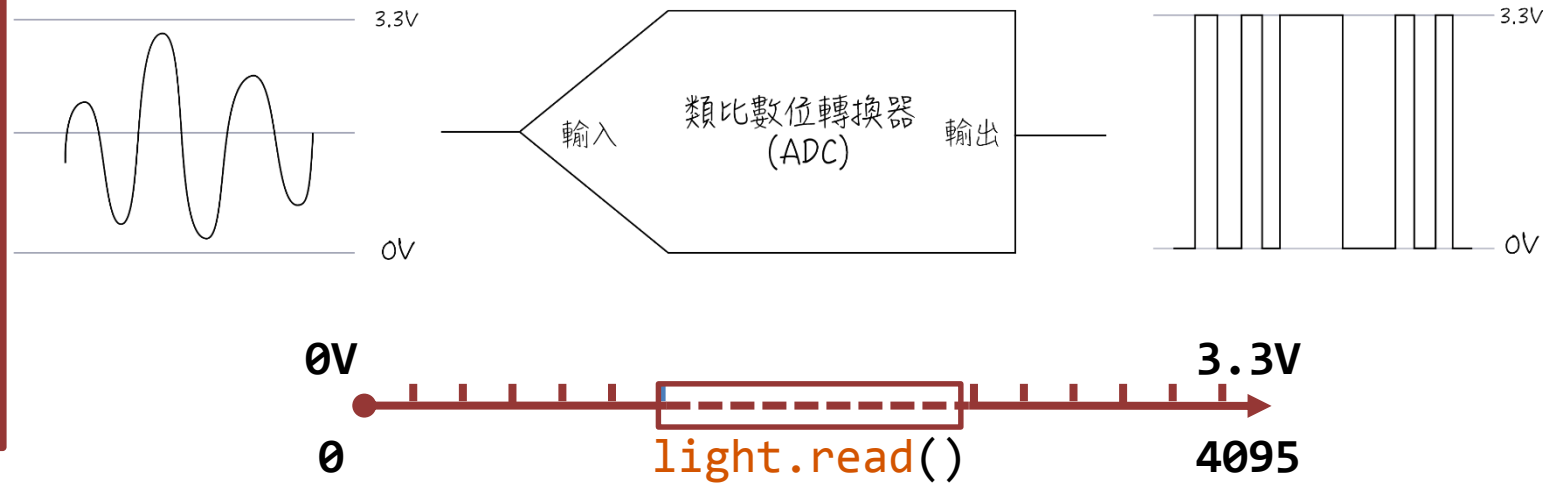
光感應自動燈

類比轉數位輸入 (Analog-to-Digital)

原理

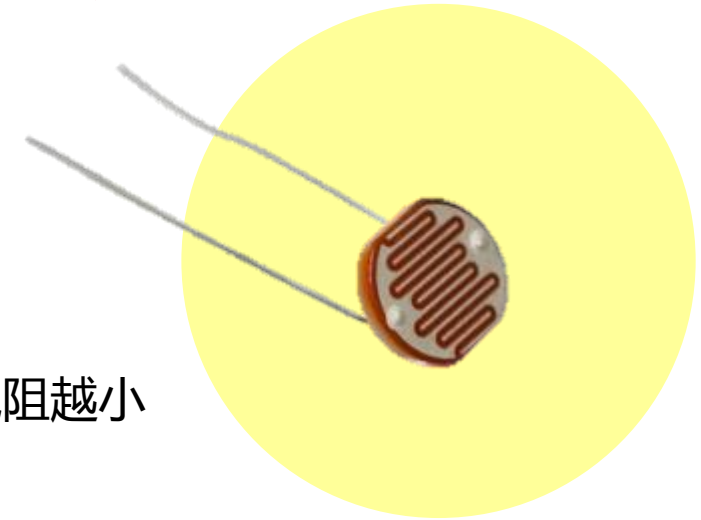
ADC腳位
Analog-to-Digital Converter

A0	A1	A2	A3
ADC(0)	ADC(1)	ADC(2)	ADC(3)
3.3V			
GND			

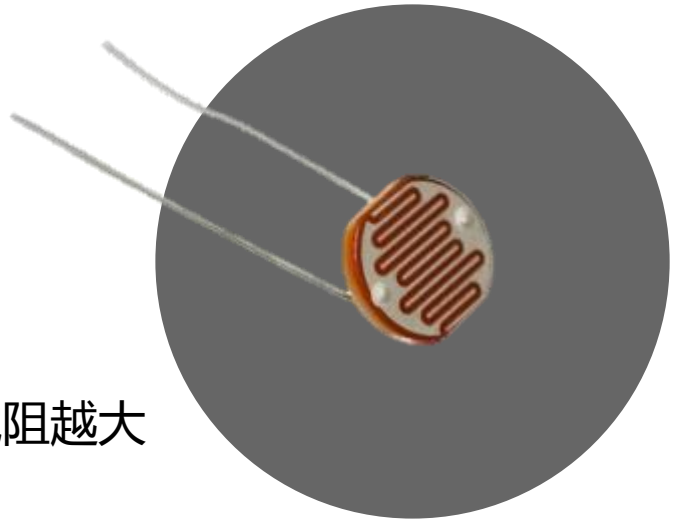


圖片來源：旗標科技 超圖解 Python 物聯網

原理



光越亮電阻越小

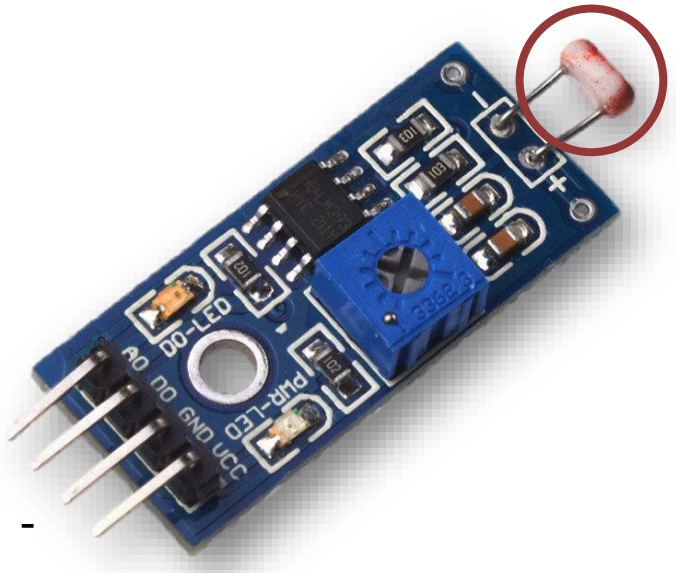


光越暗電阻越大

暗 ◄——————► 亮

電壓高　　電壓低

光敏電阻

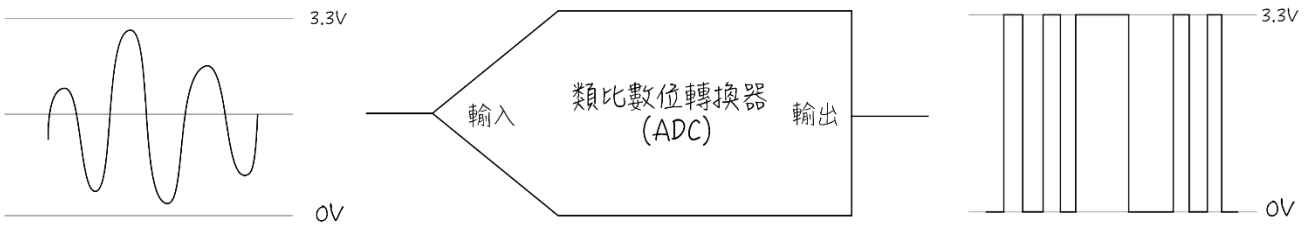


AO：電壓變化

GND：-

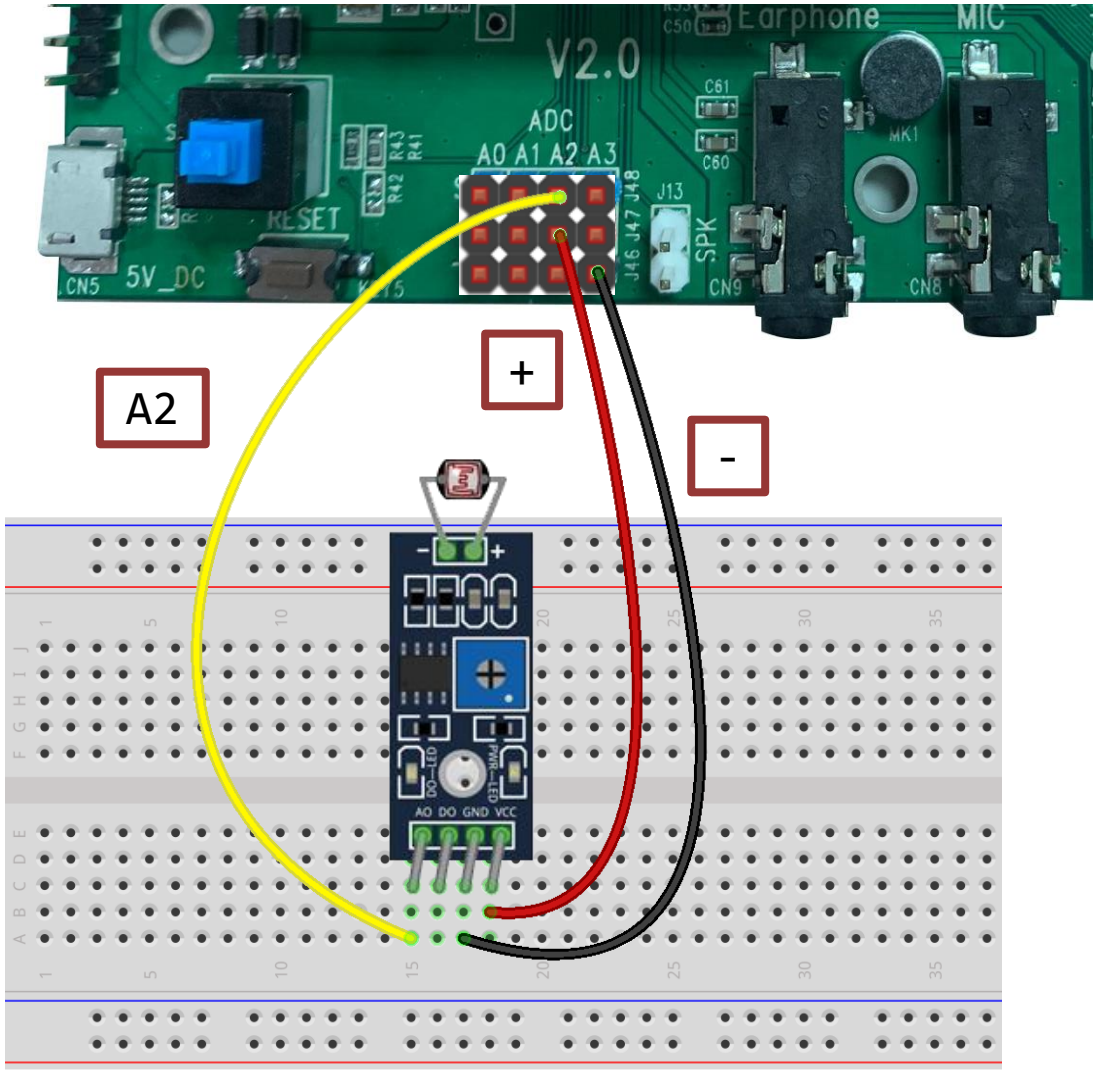
VCC：3.3V

控制板無法直接**偵測光敏電阻的變化，**
不過我們使用的是現成的**模組**
內建電路把**電阻轉成電壓**變化



動手做

接線



本實驗會使用 AD2
讀取光敏感測數據

```
>>> from machine import ADC
>>> light = ADC(2)
>>> light.read()
180
>>> light.read()
841
>>> light.read()
2413
```

越亮數值越小

越暗數值越大

動手做 寫程式製作光感應燈

可參考sample code: 09_light.py

```
1  from machine import Pin,ADC          # 匯入Pin,ADC類別
2  from gpb import delay                # 匯入delay函式
3
4  led = Pin('C13',Pin.OUT)             # 建立控制LED的腳位
5  light = ADC(2)                       # 建立讀取光線變化的物件
6
7  while True:                          # True代表成立
8      if light.read()>1000:            # 根據實驗值調整值
9          led.value(1)                 # 太暗亮燈
10     else:
11         led.value(0)                 # 夠亮熄燈
12     print(light.read())
13     delay(50)                        # 等50毫秒
```

這裡有兩層縮排
第一層是 while
第二層是 if 結構

```
if 條件判斷式:
    成立時要做的動作
    成立時要做的動作
    ...
else:
    不成立時要做的動作
    不成立時要做的動作
    ...
```

注意 if else 尾端的**冒號**以及**縮排**

> 會判斷左邊的值**是否大於**右邊
判斷式會依此得 **True** 或是 **False**

9

PART

超音波感測器

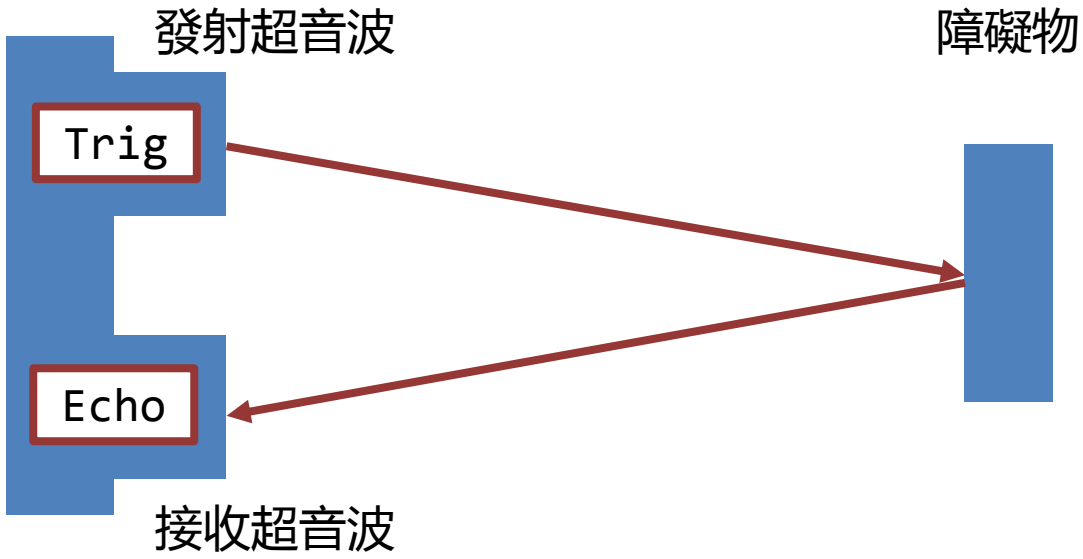
監測距離

原理

超音波

送出**高電位 10μs**
就會**發出超音波**

送出超音波之後
會先切到**高電位**
等**收到**超音波就
會變回**低電位**



可由超音波來回時間以及音速 340.29m/s 推算距離

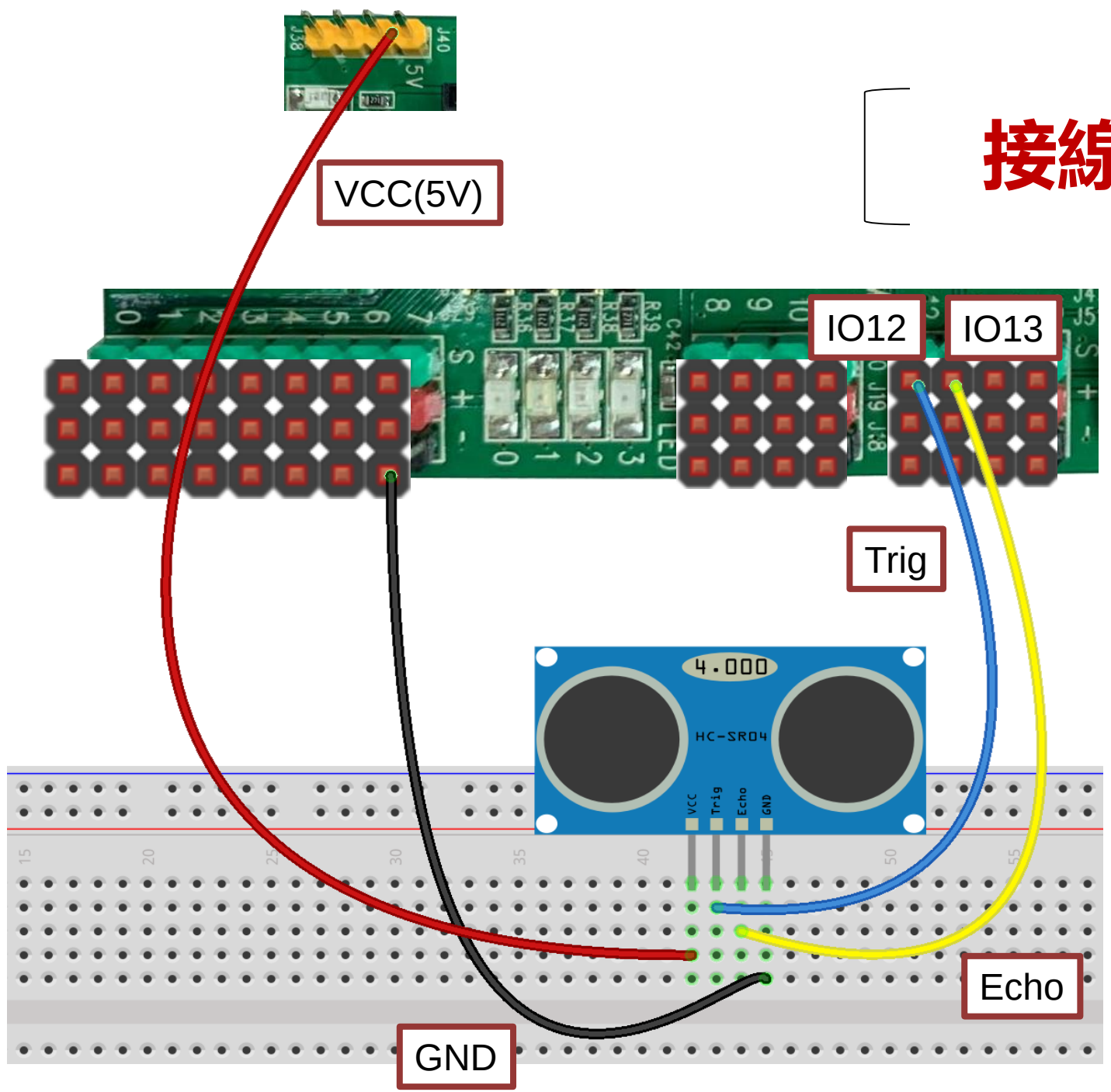
動手做 使用模組提供的函式

IO0	IO1	IO2	IO3	IO4	IO5	IO6	IO7
C0	C1	C2	C3	F0	F1	F2	F3
3.3V							
GND							

PWM8	PWM9	PWM10	PWM11
E0	E1	E2	E3
3.3V			
GND			

IO12	IO13	IO14	IO15
E4	E5	E6	E7
3.3V			
GND			

本實驗會使用 E4(trig)、E5(echo) 控制超音波測距感測器



接線

```
>>> from sensor import HC_SR04
>>> sr04=HC_SR04('E4','E5')
>>> sr04.Ultrasound()
41

>>> sr04.Ultrasound()
16

>>> sr04.Ultrasound()
22

>>> sr04.Ultrasound()
30
```

動手做 寫程式監測距離

可參考sample code : 10_sr04.py

```
1  from sensor import HC_SR04                  # 匯入HC_SR04類別
2  from gpb import delay                       # 匯入delay函式
3
4  sr04 = HC_SR04('E4','E5')                   # Trig腳位,Echo腳位
5
6  while True:                                 # True 代表成立
7      print(str(sr04.Ultrasound())+"cm")# 用數字建立字串物件再串接字串
8      delay(1000)                             # 每秒監測一次
```

```
互動環境(Shell) ×
42cm
14cm
19cm
30cm
42cm
42cm
43cm
```

超出可測範圍（太近或太遠）傳回 0，要自己過濾掉

10

PART

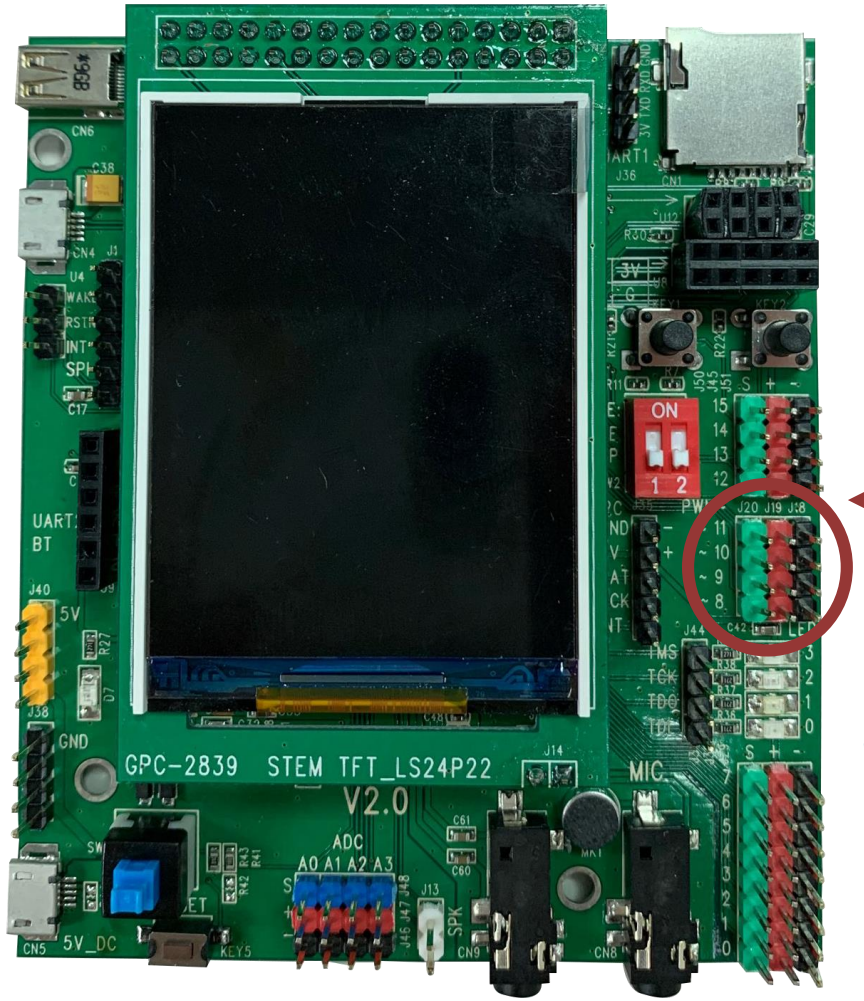
伺服馬達

柵欄、機器手臂的根本

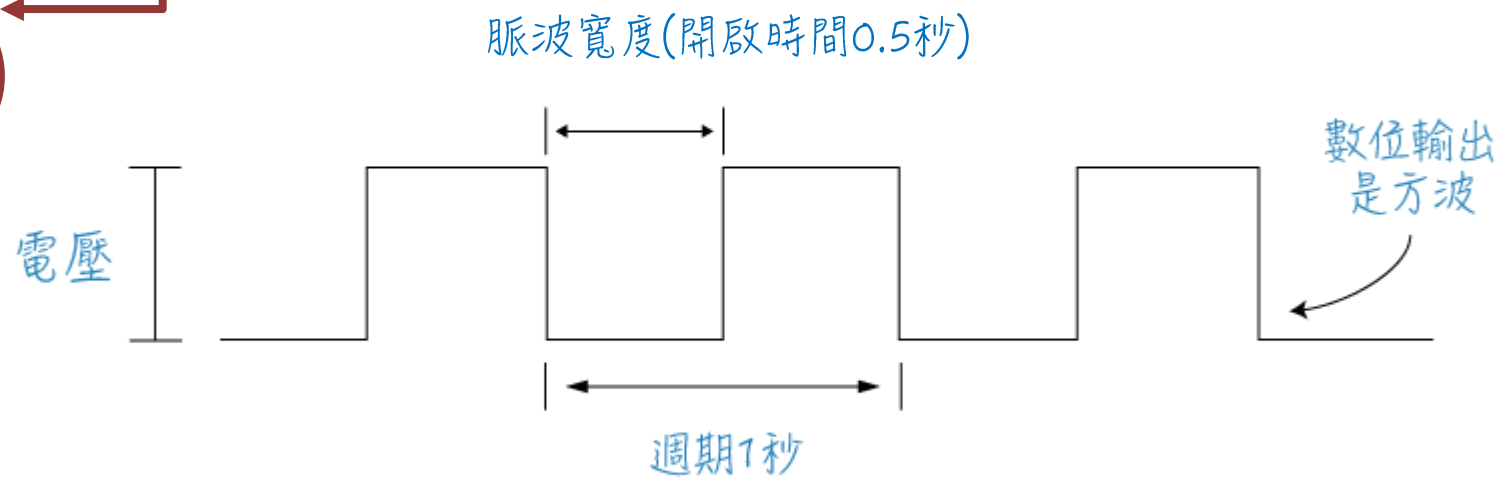
原理

PWM腳位

Pulse-width modulation



PWM8	PWM9	PWM10	PWM11
E0	E1	E2	E3
Servo(1)	Servo(2)	Servo(3)	Servo(4)

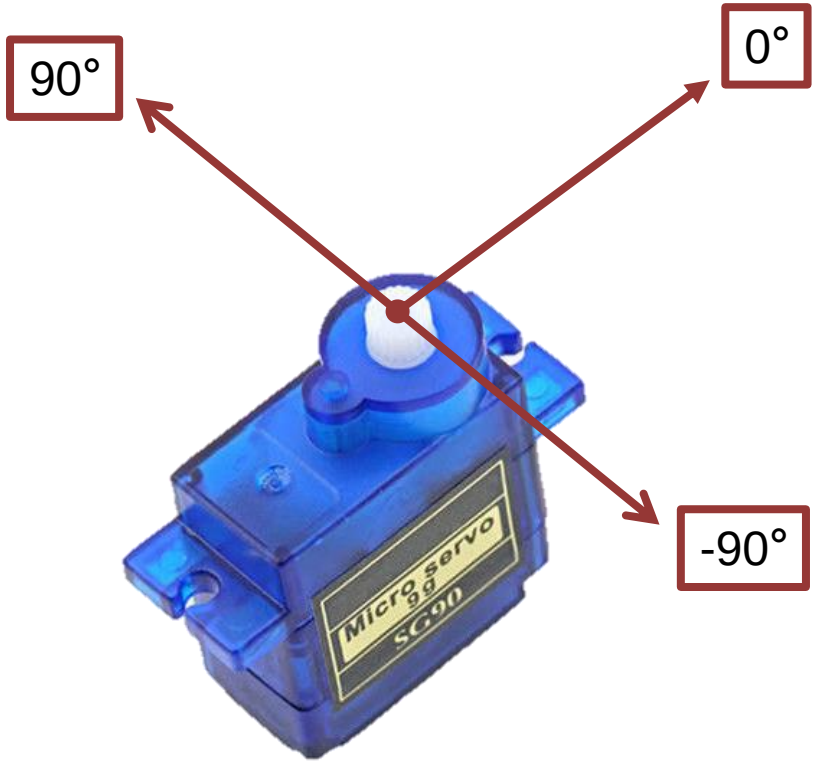


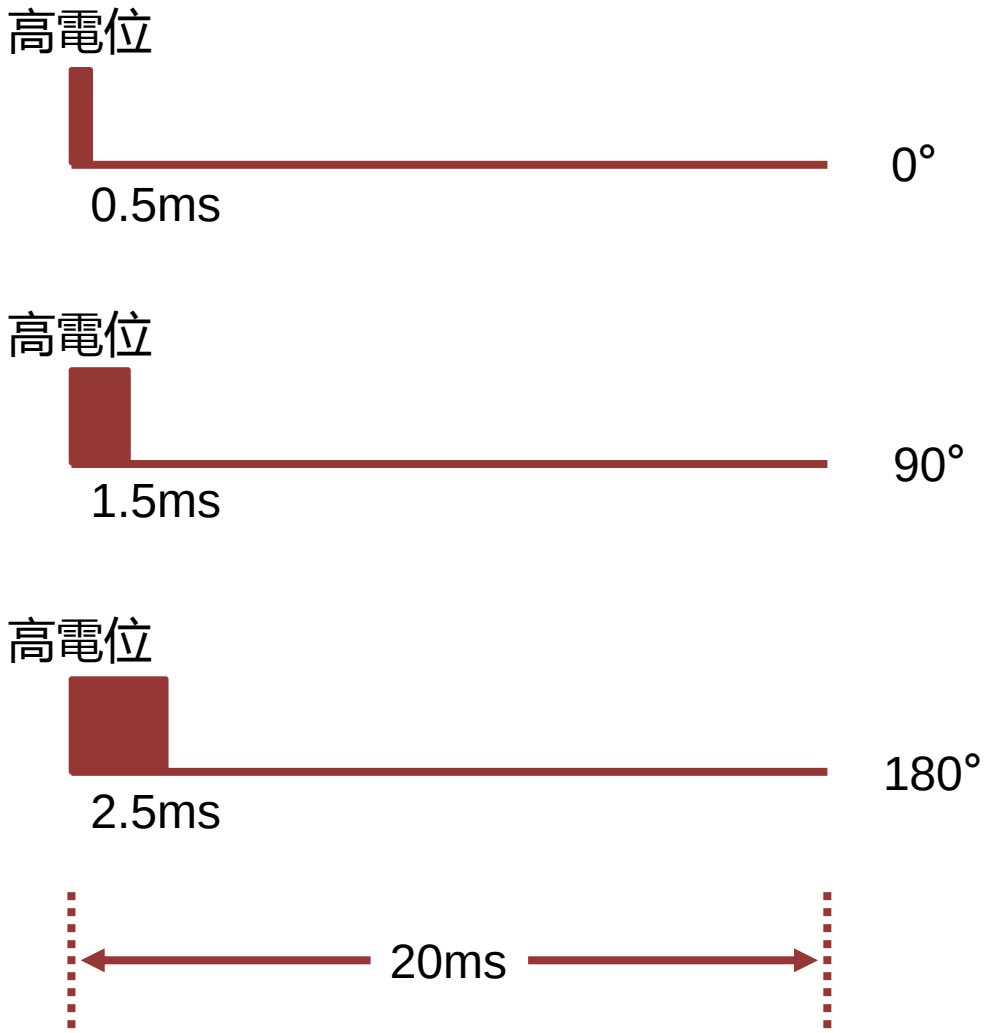
圖片來源：旗標科技 超圖解 Python 物聯網

原理

可控制角度的伺服馬達

我們使用的是**90-180度伺服馬達**，
可以控制**角度**

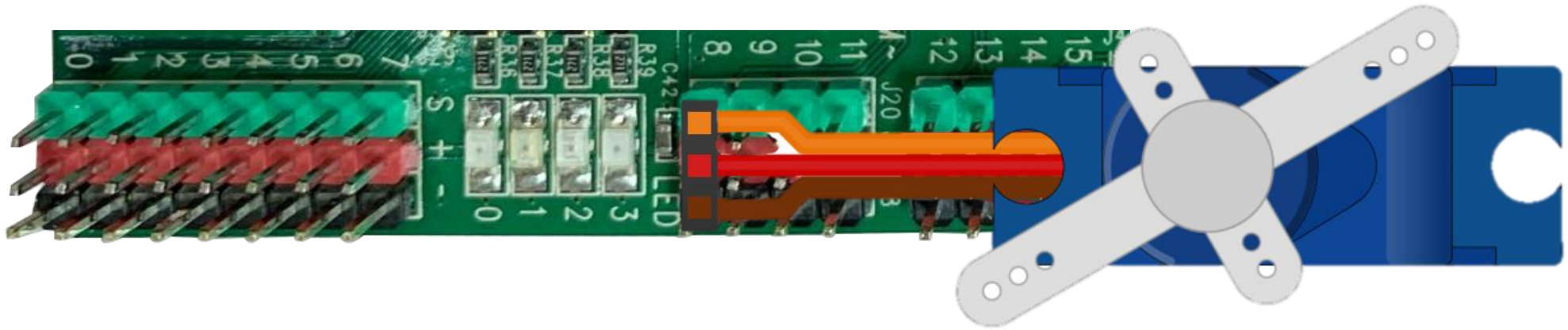




動手做

接線

直接將接頭接在 PWM8 上



PWM8	PWM9	PWM10	PWM11
E0	E1	E2	E3
Servo(1)	Servo(2)	Servo(3)	Servo(4)

PWM腳位總共可控制 4 組馬達，
本實驗將使用PWM8控制Servo1

使用**現成的模組**

互動環境(Shell)

```
>>> from gpb import Servo
>>> door = Servo(1)
>>> door.angle(0)
>>> door.angle(-90)
>>> door.angle(90)
>>> door.angle(0)
>>>
```

動手做

寫程式轉動柵欄

可參考sample code:11_servo.py

```
1  from gpb import Servo      # 匯入 Servo 類別
2  from gpb import delay      # 匯入 delay 函式
3
4  servo1 = Servo(1)          # 使用 PWM8 腳位操控 Servo1
5
6  while True:                # True 代表成立
7      servo1.angle(0)        # 停止
8      delay(500)             # 等 0.5 秒
9      servo1.angle(-50)      # 順時針轉 50度
10     delay(500)
11     servo1.angle(0)
12     delay(500)
13     servo1.angle(50)       # 逆時針轉 50度
14     delay(500)
```

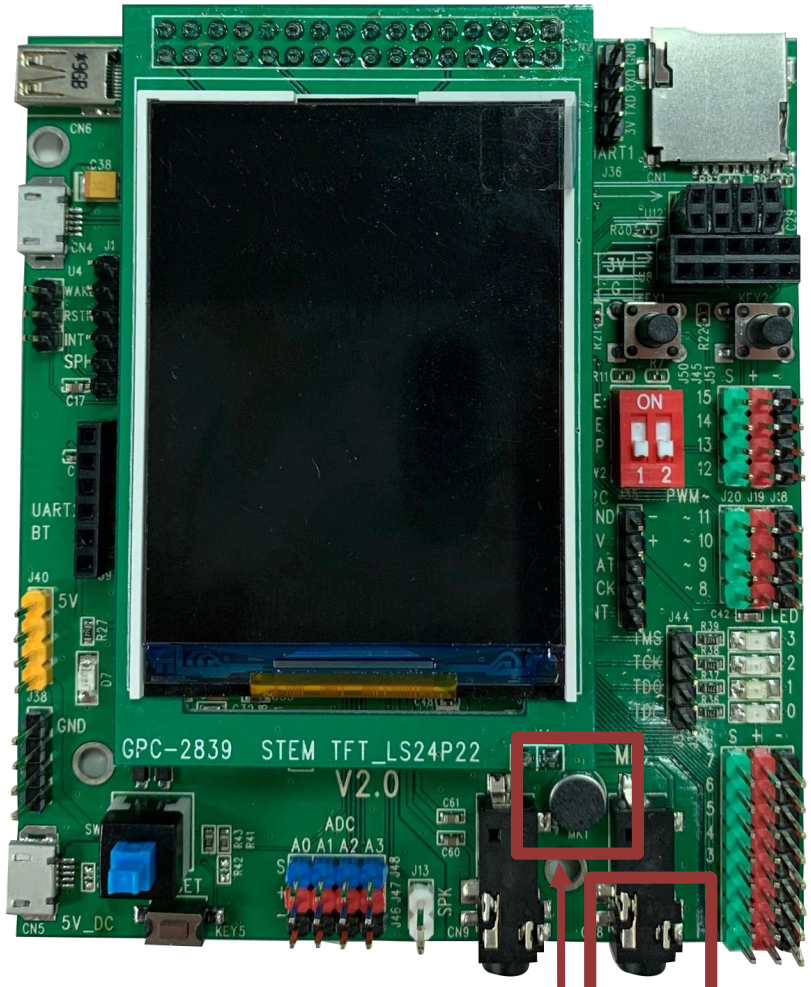
11

PART

語音辨識

簡易智慧管家

原理

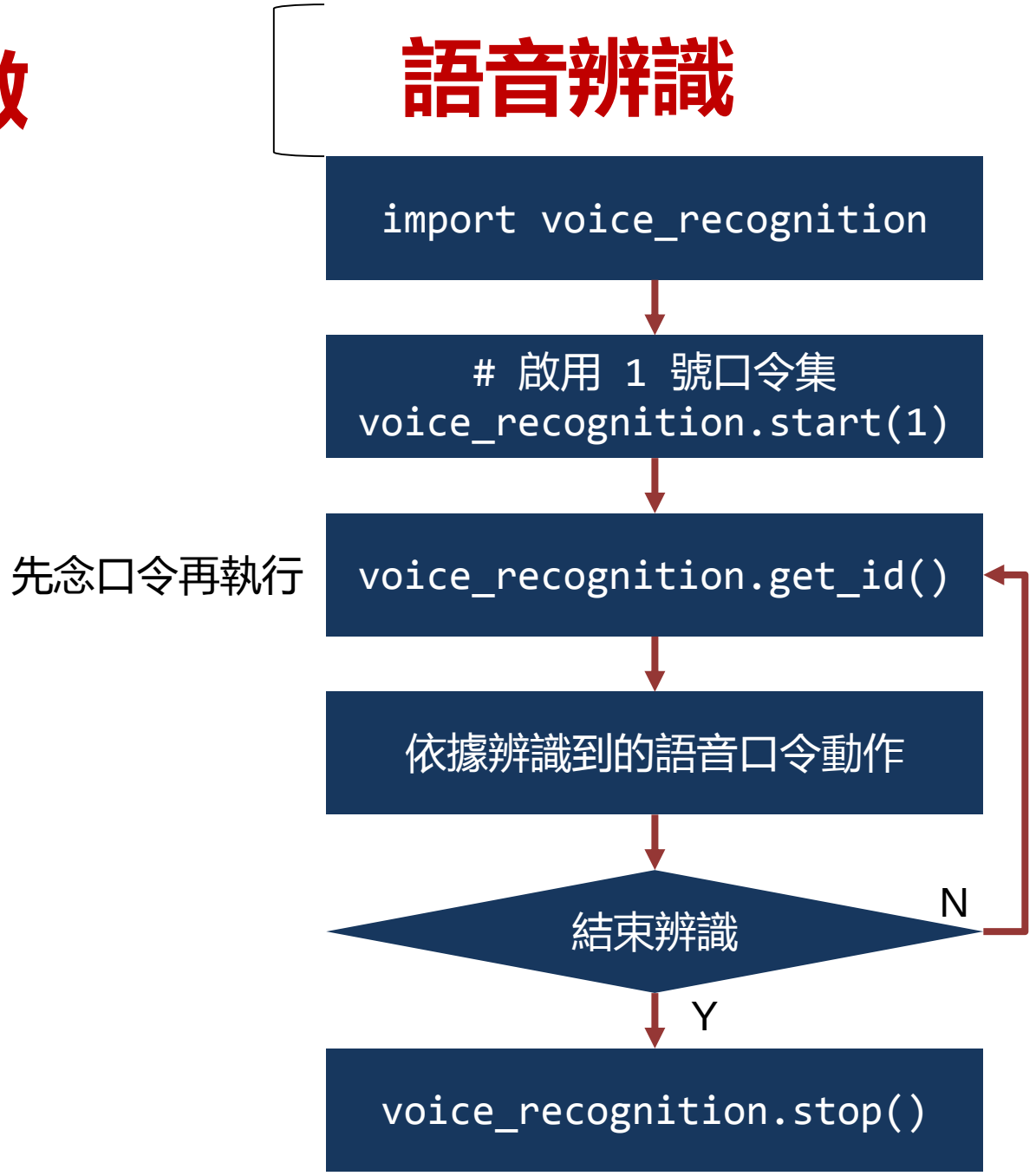


內建語音口令辨識

編號id	口令集 1	編號id	口令集 1
1	小愛管家	11	停止音樂
2	打開電燈	12	現在溫濕度
3	關閉電燈	13	大門上鎖
4	打開風扇	14	大門解鎖
5	提高風速	15	凌威管家
6	降低風速	16	拍森管家
7	關閉風扇	17	小珍同學
8	播放音樂	18	志玲姐姐
9	增加音量	19	柔伊妹妹
10	降低音量	20	美玉阿姨

動手做

語音辨識

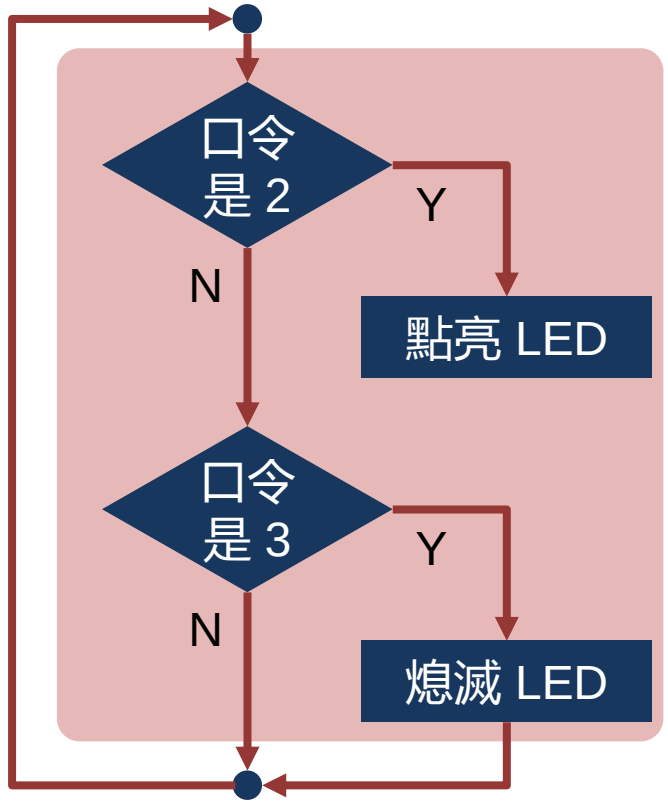


先念口令再執行

動手做 寫程式用語音控制LED

可參考sample code:
12_voice_recognition_led1.py

```
1  import voice_recognition                    # 匯入語音辨識模組
2  from machine import Pin
3
4  led = Pin('C13',Pin.OUT)
5  led.value(0)                                # 預設熄燈
6
7  voice_recognition.start(1)                  # 啟用1號口令集
8
9  while True:
10     cmd_id = voice_recognition.get_id()     # 取得辨識結果
11     if cmd_id == 2:                         # 口令:打開電燈
12         led.value(1)                        # 亮燈
13     elif cmd_id == 3:                       # 口令:關閉電燈
14         led.value(0)                        # 滅燈
```



利用 elif 建立多個條件判斷的流程

原理

擴充語音口令辨識

編號id	口令集 10	編號id	口令集 10
1	小林管家	11	停止音樂
2	打開電燈	12	關閉音樂
3	關閉電燈	13	大門上鎖
4	打開風扇	14	大門解鎖
5	芝麻開門	15	鎖上大門
6	芝麻關門	16	大家平安
7	關閉風扇		
8	播放音樂		
9	增加音量		
10	降低音量		

語音建模說明書請參照PDF檔or教學資源下載區下載

動手做

利用擴充語音控制LED

可參考sample code:
13_voice_recognition_led2.py

```
1  import voice_recognition
2  from machine import Pin
3  from gpb import delay
4
5  led=Pin('C13',Pin.OUT)
6  led.value(0)
7
8  voice_recognition.load_database('VRnew.bin')
9  delay(500)
10 voice_recognition.start(10)    #啟用10號口令集
11
12 while True:
13     cmd = voice_recognition.get_id()
14     if cmd == 1:
15         led.value(1)
16     elif cmd == 3:
17         led.value(0)
```

對於用戶數據庫，如果未加載數據庫，則使用默認數據庫，
請把要擴充的語音口令集(bin檔)放入SD卡根目錄。

12

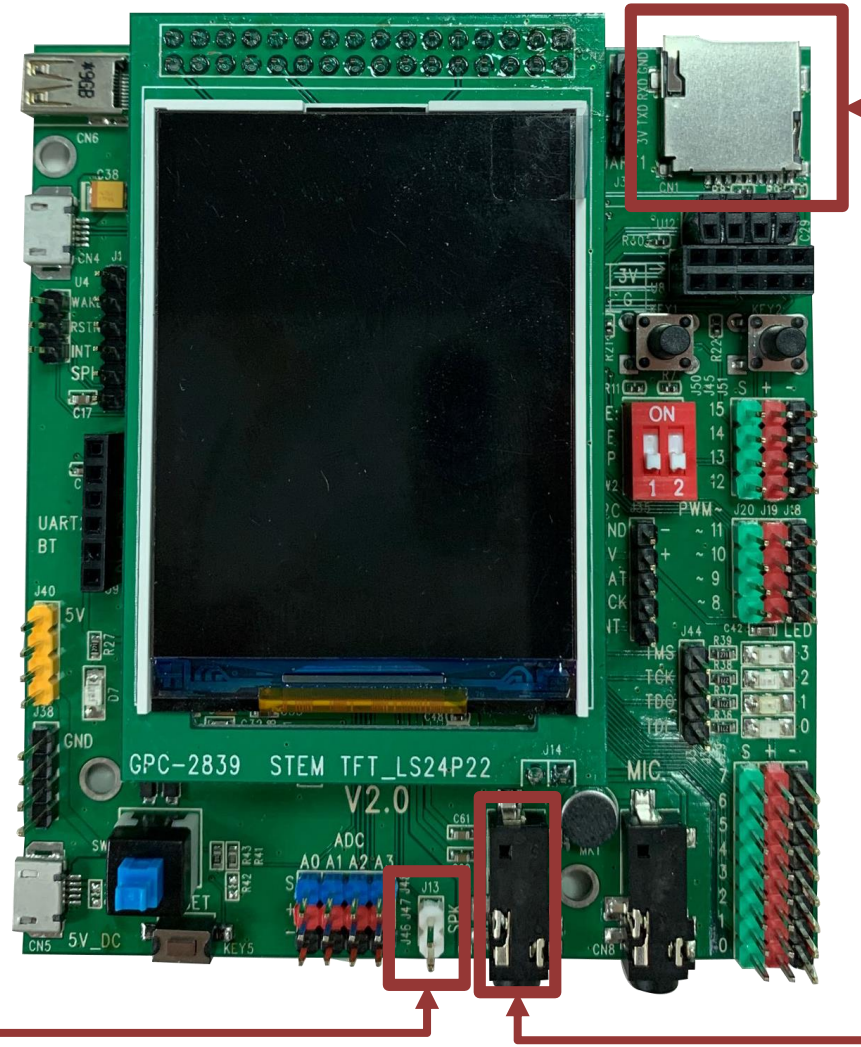
PART

播放聲音

播放 SD 卡上的 mp3/wav 檔

原理

內建音效處理晶片



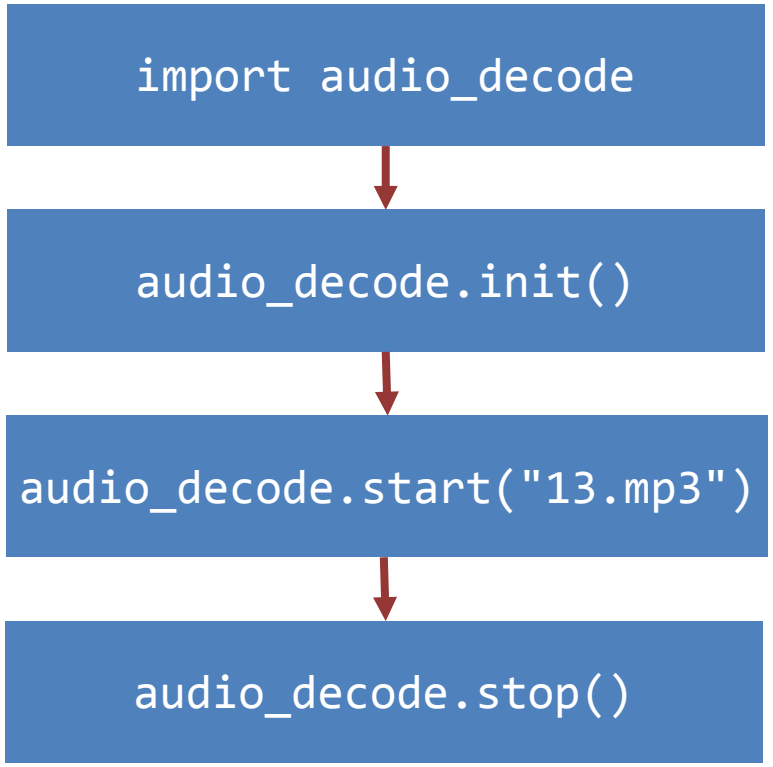
MicroSD 卡會變成隨身碟

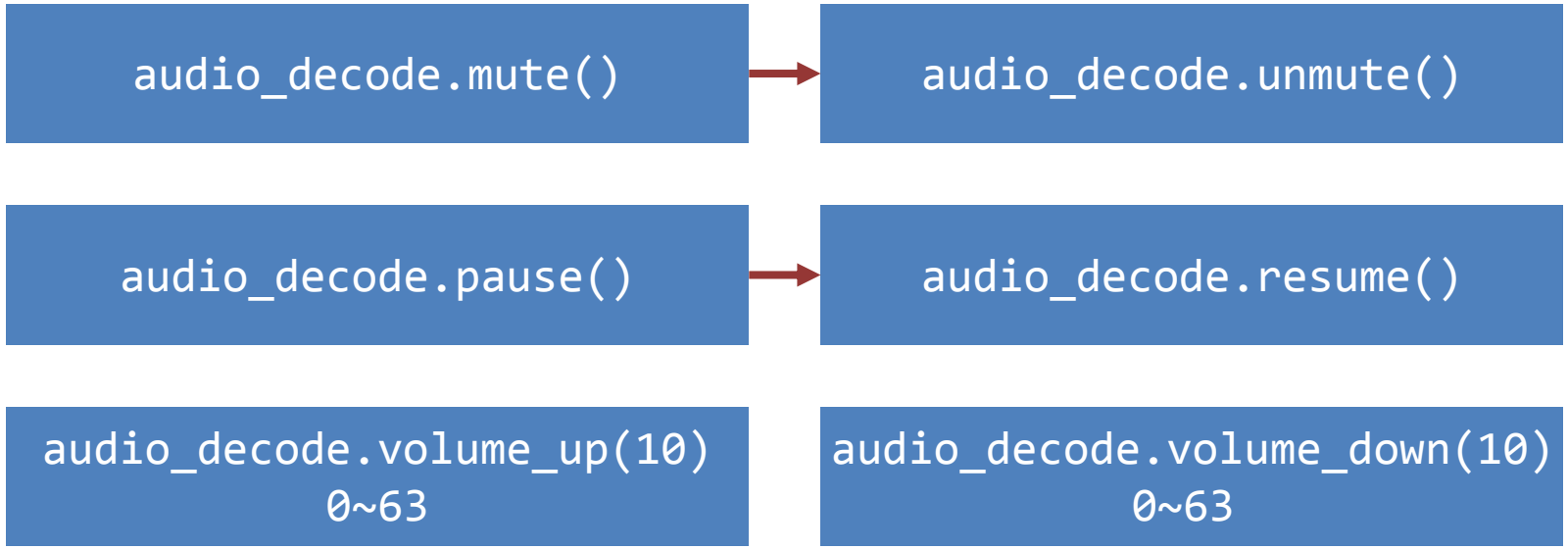
可播放 SD 卡上的 mp3 / wav 檔

內建喇叭接腳

3.5mm 接頭
可接耳機、喇叭

音效播放流程






```
>>> import audio_decode
>>> audio_decode.init()
>>> audio_decode.start("tank.mp3")
>>> audio_decode.volume_down(10)
>>> audio_decode.volume_up(10)
>>> audio_decode.pause()
>>> audio_decode.resume()
>>> audio_decode.stop()
```

動手做 用口令播放音樂

可參考sample code:
14_audio_decode.py

```
1  import voice_recognition
2  import audio_decode
3  from gpb import delay
4
5  voice_recognition.load_database('VRnew.bin')# 載入擴充數據庫
6  delay(500)
7  voice_recognition.start(10)                  # 啟用10號口令集
8  audio_decode.init()                          # 啟用音效模組
9
10 while True:
11     cmd_id=voice_recognition.get_id()        # 取得辨識ID
12     if cmd_id == 8:                          # 口令:播放音樂
13         audio_decode.start('tank.mp3')
14     elif cmd_id == 11:                       # 口令:停止音樂
15         audio_decode.stop()                  # 關閉音效模組
```

播放音樂時需要執行其他口令，可能需要把喇叭遮住才能辨識

13

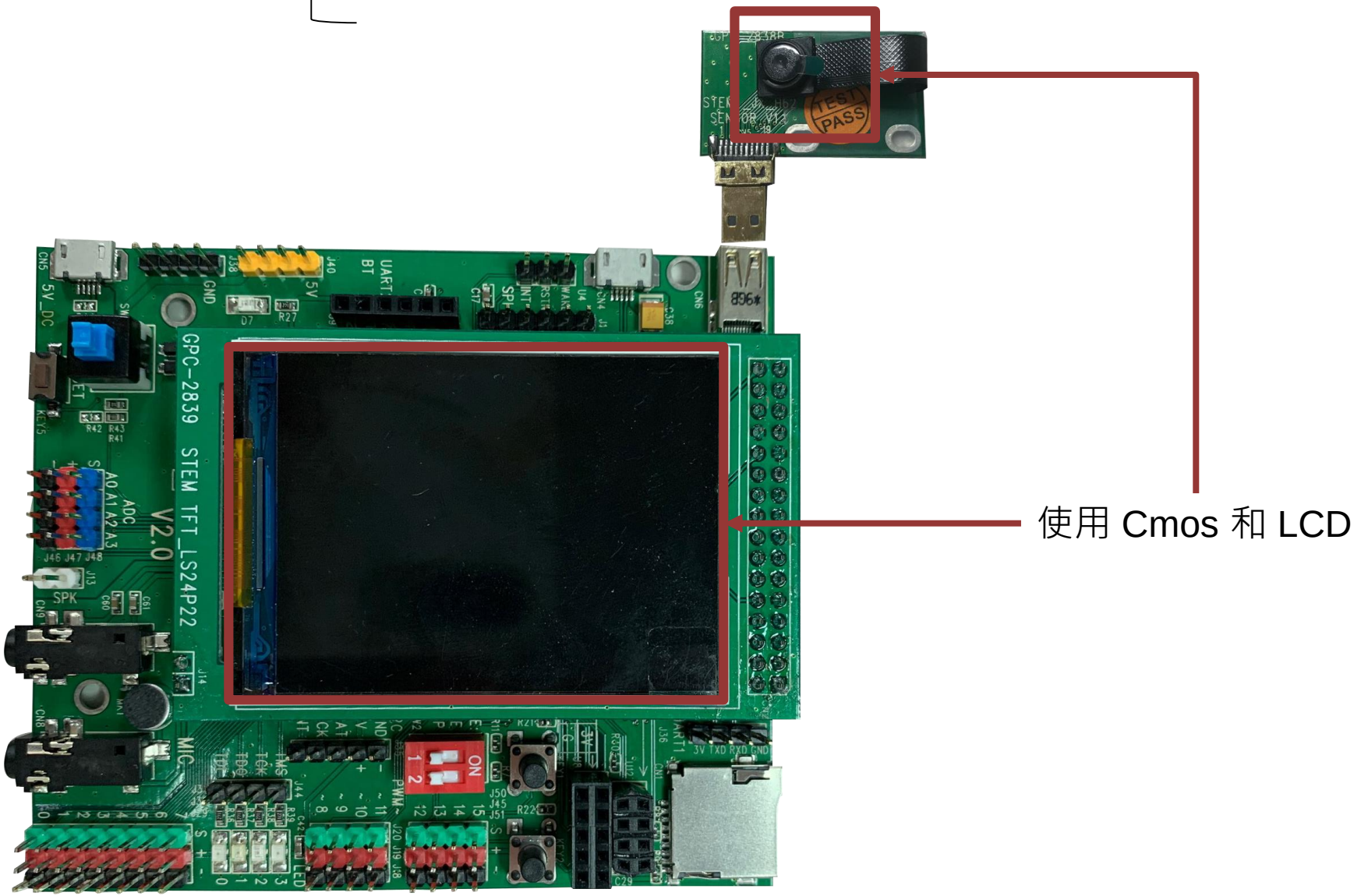
PART

人臉辨識

刷臉時代來臨了

原理

內建人臉偵測晶片

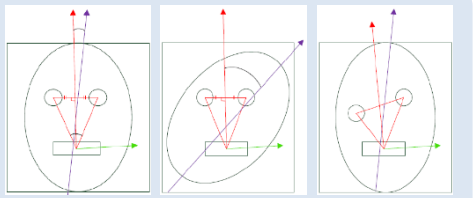


原理

人臉偵測

是 ✖ 不是人臉

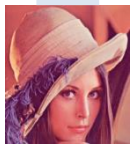
輸出



規則 1　規則 2　規則 3

人工建立的規則

輸入



測試樣本

規則式系統
(工人智慧)

是 ✖ 不是人臉

輸出

抽取特徵

人工定義的特徵

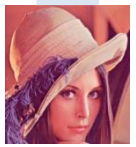
訓練模型

模型

輸入

大量樣本

輸入



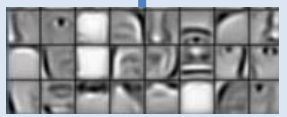
測試樣本

傳統機器學習
(半人工智慧)

是 ✖ 不是人臉

輸出

組合臉部特徵



組合五官特徵

產生線條特徵

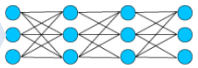
訓練模型

輸入

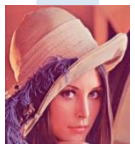
大量樣本

模型

輸出



輸入



測試樣本

深度機器學習
(人工智慧)

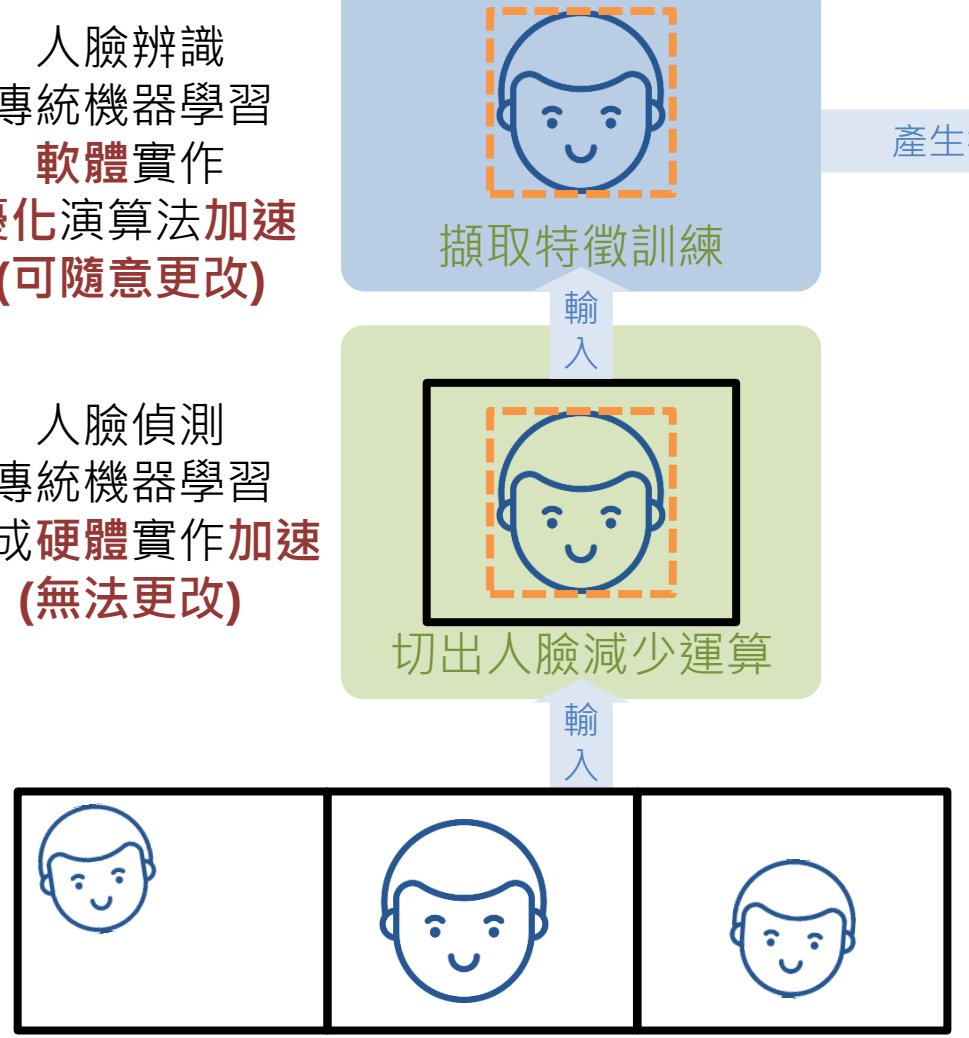
本研習利用Genio Py.
人臉辨識解決方案

人臉辨識硬體加速解決方案

訓練模式

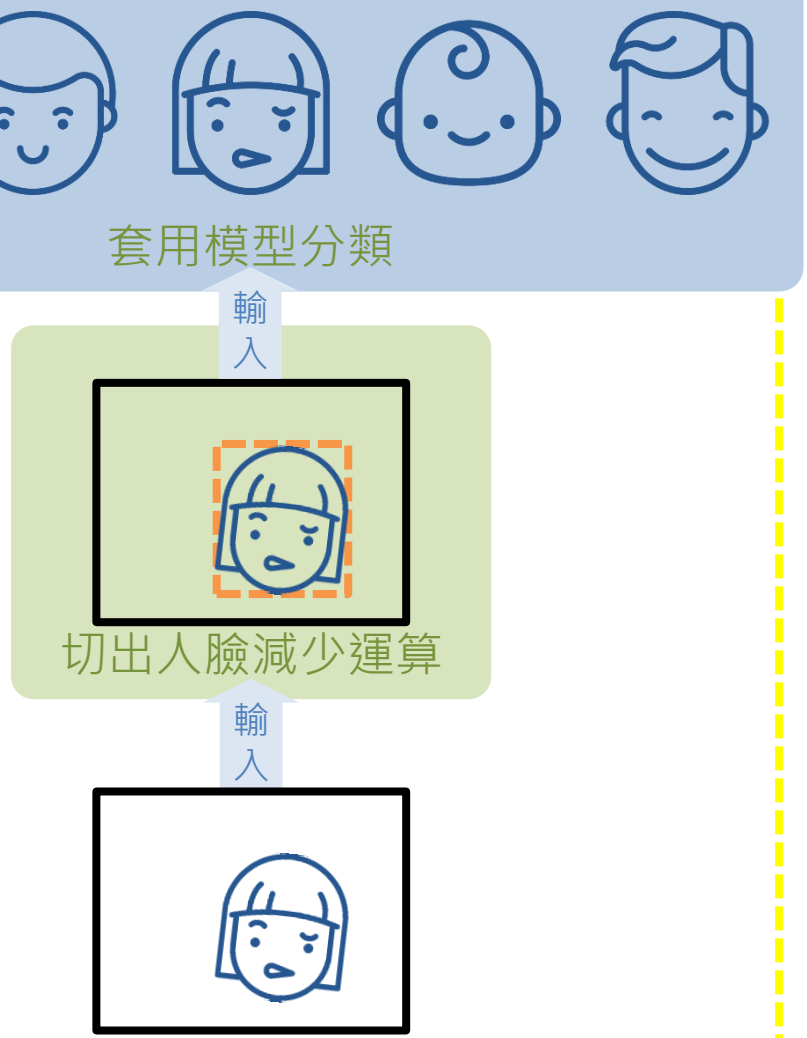
人臉辨識
傳統機器學習
軟體實作
優化演算法**加速**
(可隨意更改)

人臉偵測
傳統機器學習
完成**硬體**實作**加速**
(無法更改)



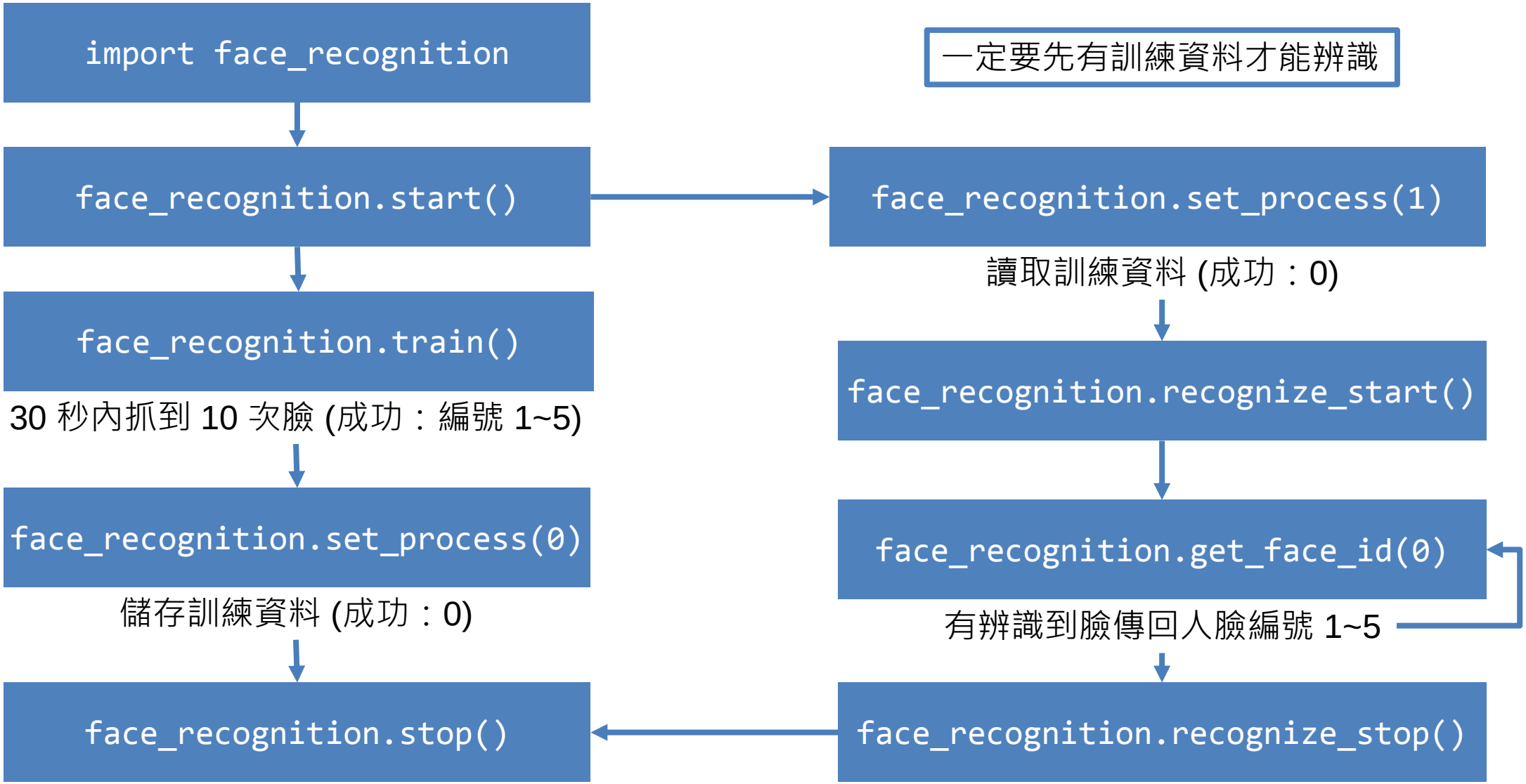
產生模型

辨識模式

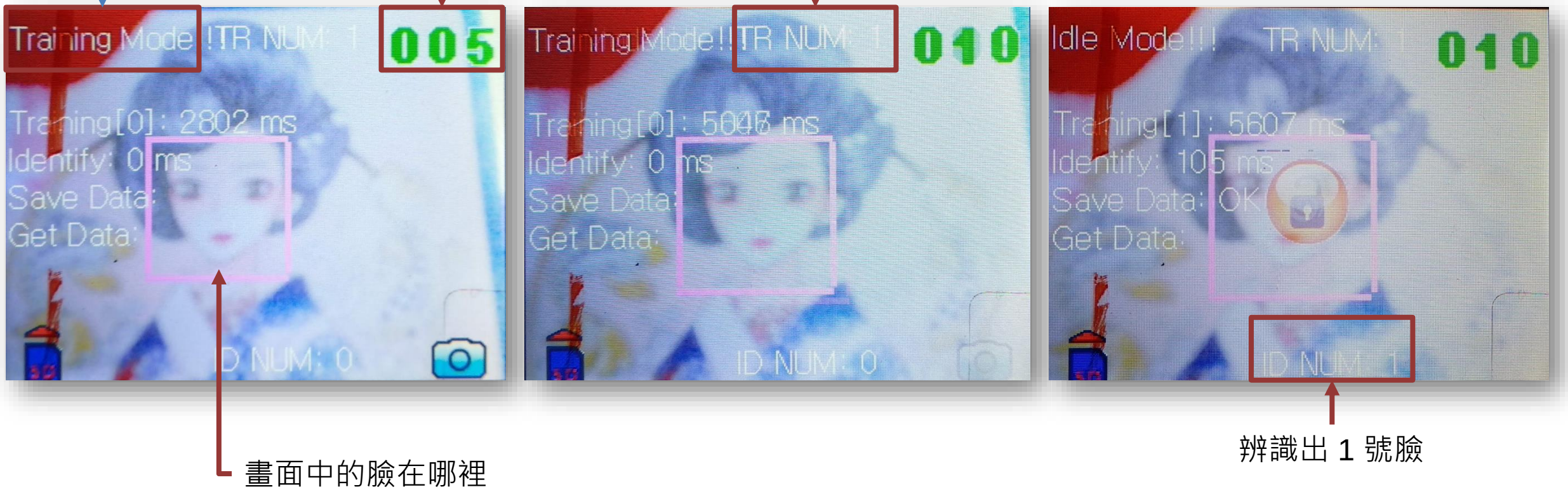


實驗原理

人臉訓練/辨識流程



動手做 人臉訓練與辨識


訓練模式
已經看到幾次臉
當前訓練的人臉編號
Training Mode !TR NUM: 1
005
Training[0]: 2802 ms
Identify: 0 ms
Save Data:
Get Data:
ID NUM: 0
Training Mode!! TR NUM: 1
010
Training[0]: 5048 ms
Identify: 0 ms
Save Data:
Get Data:
ID NUM: 0
Idle Mode!!! TR NUM:
010
Training[1]: 5607 ms
Identify: 105 ms
Save Data: OK
Get Data:
ID NUM: 1
畫面中的臉在哪裡
辨識出 1 號臉

動手做 人臉訓練與辨識

互動環境(Shell)

```
Type "help()" for more information.
>>> import face_recognition
>>> face_recognition.start()
0

>>> face_recognition.train()
1
>>> face_recognition.set_process(0)
0

>>> face_recognition.recognize_start()
0
>>> face_recognition.get_face_id(0)
1
```

訓練完成的人臉編號

進入辨識模式後就返回, 不像訓練時會等待

辨識出來的人臉編號

讀取存取資料

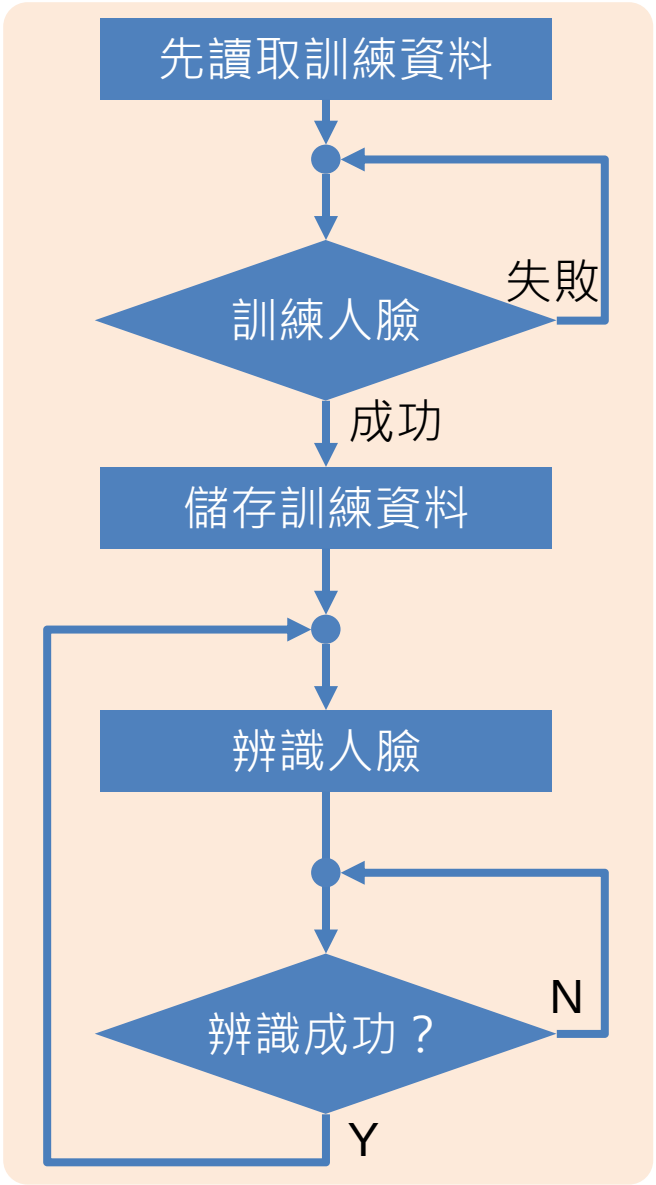
```
>>> import face_recognition
>>> face_recognition.start()
0

>>> face_recognition.set_process(1)
0

>>> face_recognition.recognize_start()
0

>>> face_recognition.get_face_id(0)
1
```

動手做

訓練/辨識配套程式

可參考sample code:
15_face_recognition.py

```
1  import face_recognition                          # 匯入人臉辨識模組
2  import audio_decode
3  from gpb import delay
4
5  audio_decode.init()
6  face_recognition.start()                         # 啟用人臉辨識模組
7
8  while face_recognition.train()== 0:              # 訓練失敗
9      pass                                         # 不做任何事
10 audio_decode.start('train_ok.mp3')               # 播放通知
11 face_recognition.set_process(0)                  # 儲存訓練資料
12 delay(2000)                                      # 等待通知播完
13
14 face_recognition.recognize_start()               # 開始辨識
15
16 while True:                                      # True代表成立
17     if face_recognition.get_face_id(0) != 0:     # 若有辨識出人臉
18         audio_decode.start('get_face.mp3')       # 播放通知
19         delay(2000)                              # 等待通知播完
20         face_recognition.recognize_start()       # 重新辨識
```



14

PART

通訊模組 藍芽傳輸

手機遠端遙控

實驗

藍芽傳輸：安裝App

iOS : App Store
Android : Google Play
中搜尋 FeasyBlue!

透過這顆藍芽模組
來傳輸控制。

動手做

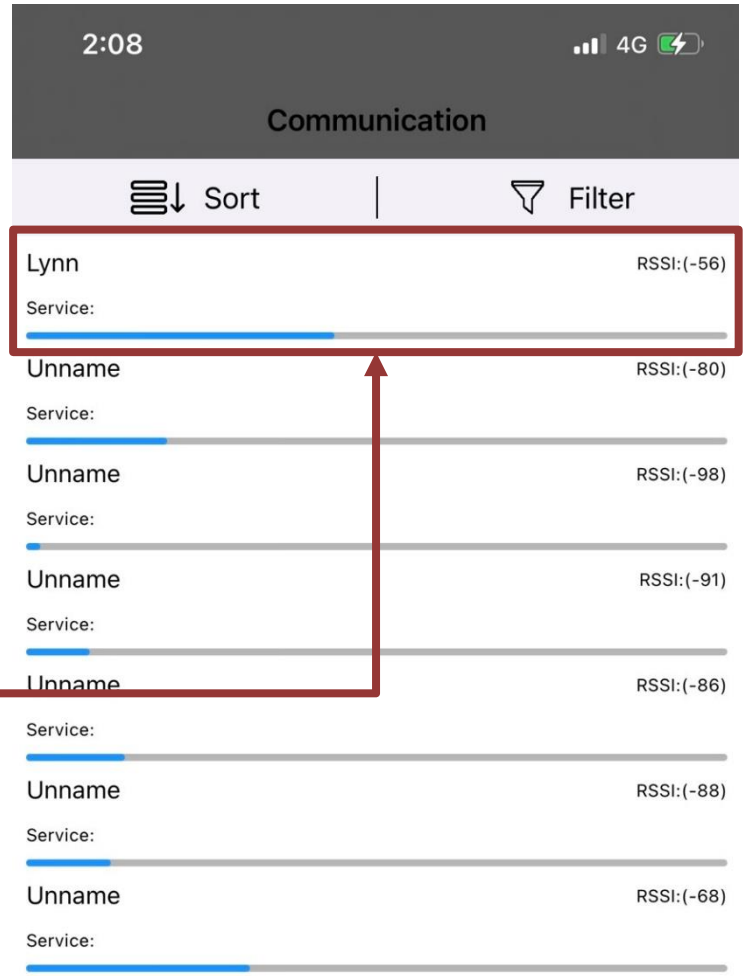
藍芽廣播流程

```
# 匯入 >>> import bluetooth
# 啟動 >>> bluetooth.active(1)
# 廣播 >>> bluetooth.advertise(100,'Lynn')
```

廣播時間間隔 (ms)

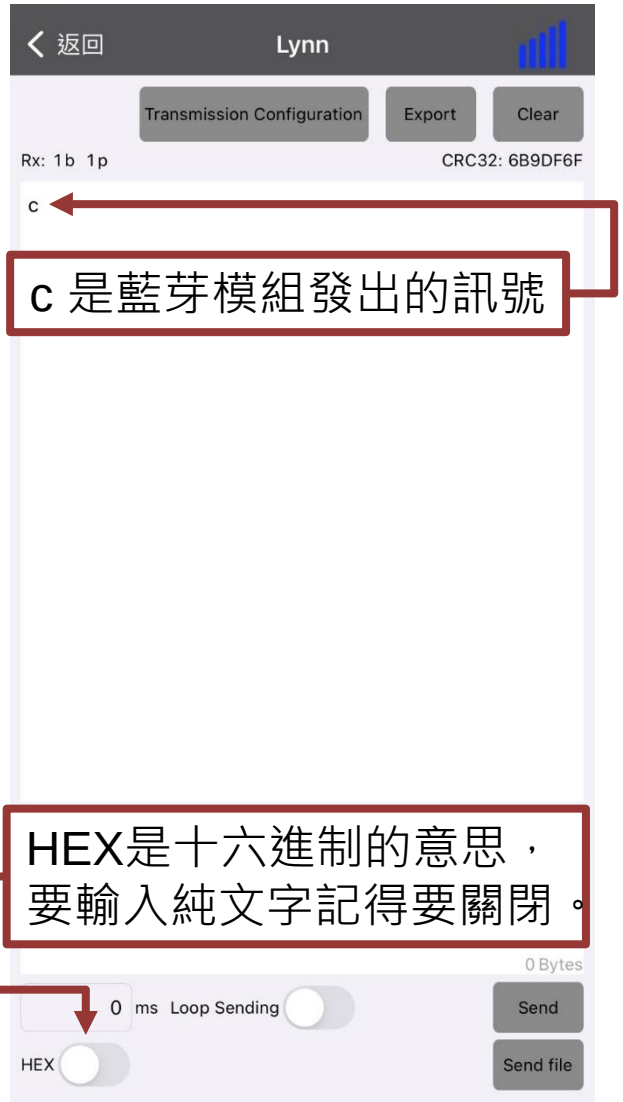
自訂裝置名稱 (最多 8 個字)

在App中點選剛才設定好啟動廣播的模組



動手做

傳送與接收資料



c 是藍芽模組發出的訊號

HEX是十六進制的意思，
要輸入純文字記得要關閉。

```
>>> bluetooth.read()          # 開發板接收
b'hello lynn'
>>> bluetooth.write('HI PC')  # 開發板傳送
```

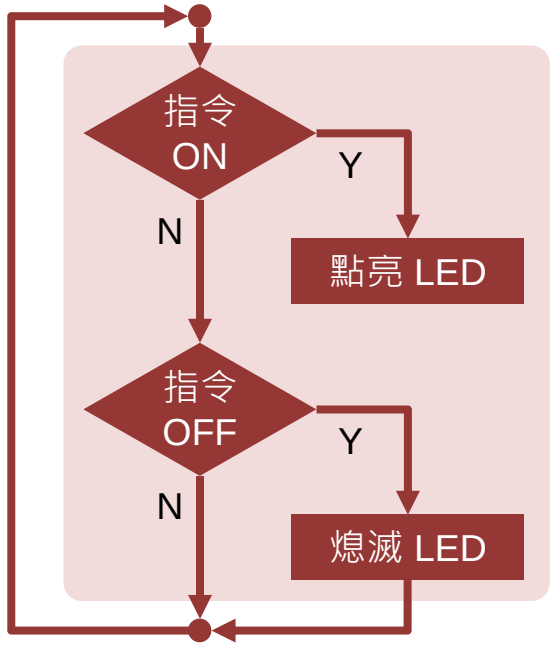
b 表示這是位元組資料

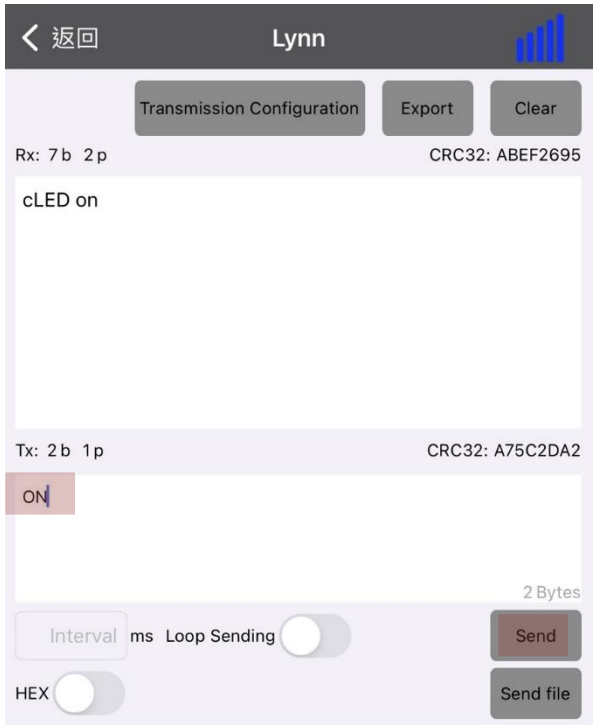
可參考sample code :
Bluetooth_LED.py

動手做 手機遙控 LED

```
1  import bluetooth                        # 匯入藍牙模組
2  from machine import Pin
3
4  led = Pin('C13', Pin.OUT)
5  led.value(0)                            # 預設燈熄
6
7  bluetooth.active(1)                     # 啟用並廣播名稱
8  bluetooth.advertise(100, 'Lynn')
9
10 while True:
11     cmd = bluetooth.read()              # 取得手機傳送過來的指令
12     if cmd != b'':
13         print(cmd)
14     if cmd.startswith(b'ON'):           # 指令：ON
15         led.value(1)                    # 亮燈
16         bluetooth.write("LED on")       # 回饋資訊給手機
17     elif cmd.startswith(b'OFF'):        # 指令：OFF
18         led.value(0)                    # 熄燈
19         bluetooth.write("LED off")      # 回饋資訊給手機
```

裝置名稱 (最多 8 個字)





THANKS

Q&A 時間