用 Python 學物聯網

旗標科技公司
行銷工程師
黃昕暐

今日議程

時間	主題
09:00—09:30	00：課程簡介與軟體環境設定 D1 mini 與 Thonny 開發環境
09:30—10:00	01： Python 與硬體基礎 物件、資料型別、變數、匯入模組
10:00—10:30	02：控制 LED 亮暗 - 數位輸出 Python 流程控制 (while 迴圈) 與區塊縮排
10:30—10:40	休息
10:40—11:20	03：防盜即時警報器 手機簡訊、LINE即時通知
11:20—12:00	04：PM 2.5 空污警報燈 Open Data網路爬蟲

今日議程

時間	主題
13:00—13:40	05：RFID刷卡紀錄 雲端資料庫
13:40—14:20	06：雨量即時統計圖 雲端數位儀表板
14:20—14:30	休息
14:30—15:10	07：手機 APP 遙控感測 Blynk自訂介面手機 APP
15:10—15:50	08：自製雲端平台 用網頁遙控家電
15:50—16:00	Q & A

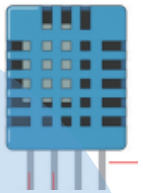
3 小時議程

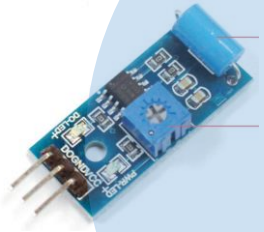
時間	主題
09:00—09:20	00：課程簡介與軟體環境設定 D1 mini 與 Thonny 開發環境
09:20—09:40	01： Python 與硬體基礎 物件、資料型別、變數、匯入模組
09:40—10:20	02：控制 LED 亮暗 - 數位輸出 Python 流程控制 (while 迴圈) 與區塊縮排
10:20—10:30	休息
10:30—11:00	03：PM 2.5 空污警報燈 Open Data網路爬蟲
11:00—11:20	04：溫濕度感測器 環境感測
11:20—12:00	05：手機 APP 遙控感測 Blynk自訂介面手機 APP

00 課程簡介與軟體環境設定

D1 mini 與 Thonny 開發環境

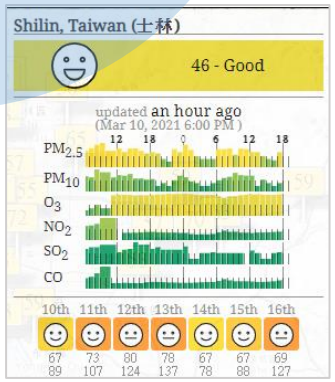
實驗架構





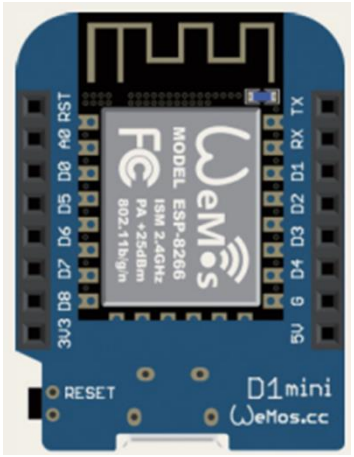
感測器

輸入



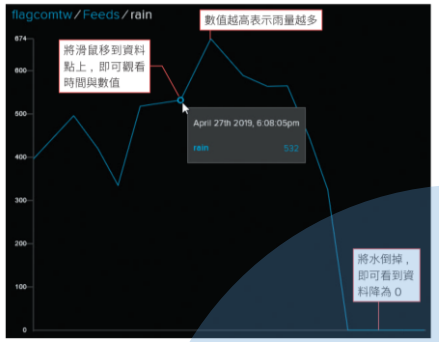
控制板

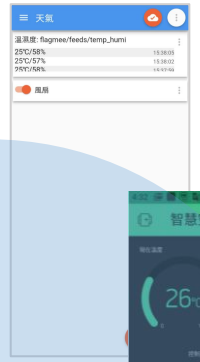
程式邏輯

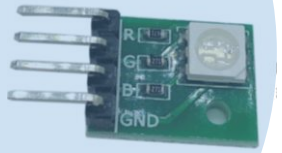


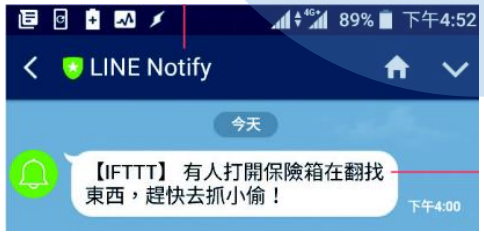
聯網

呈現、通知、儲存









	A	B	C
1	March 12, 2021	home	1931a16e

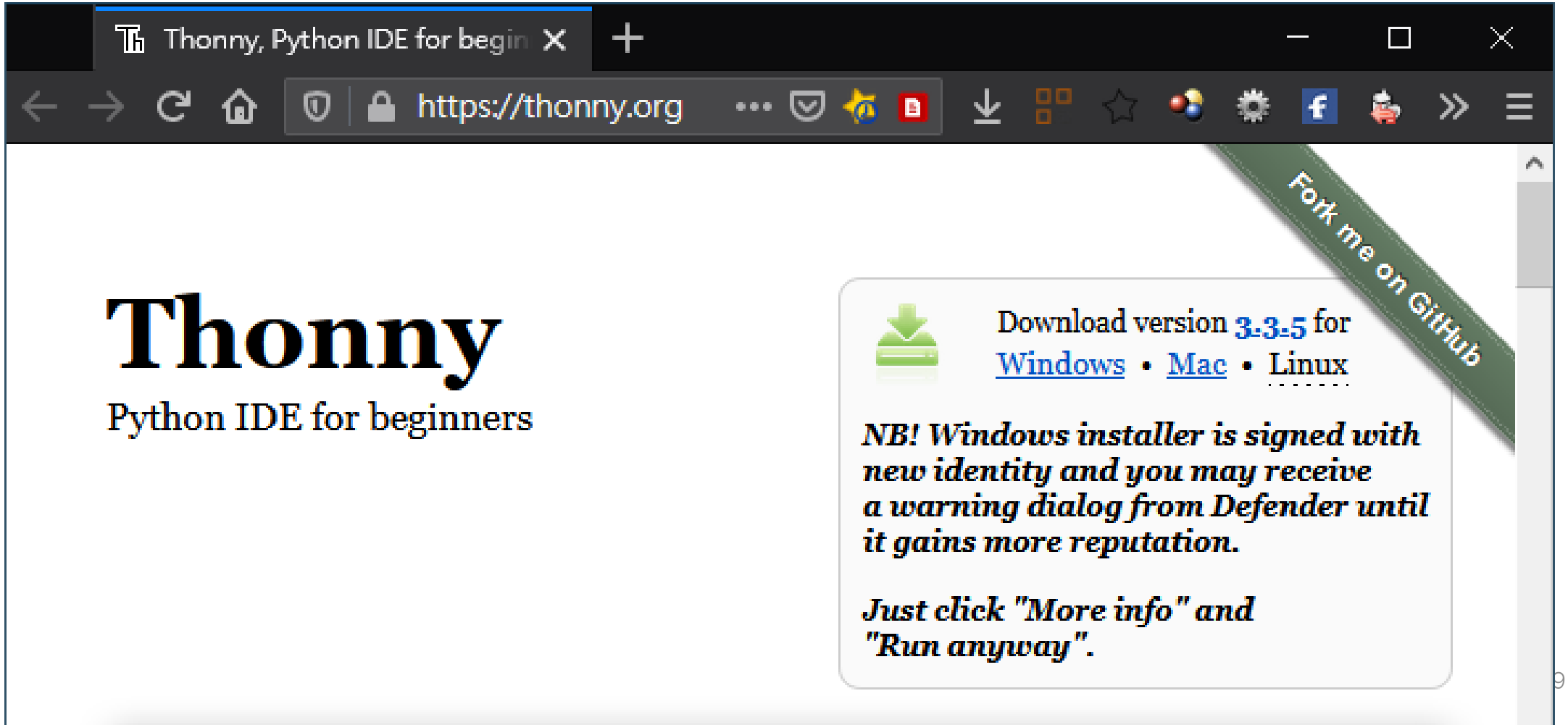
實驗流程

前置作業

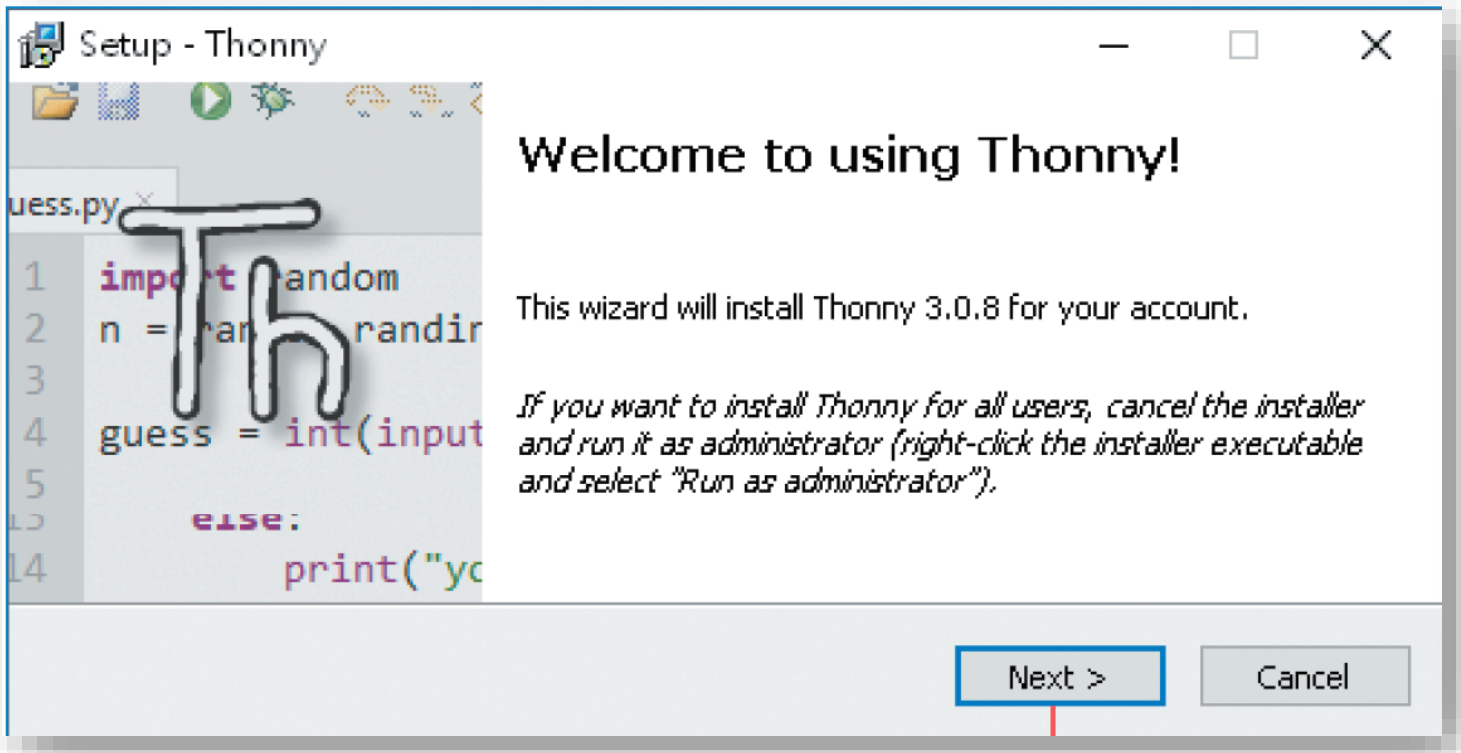
Thonny 程式開發環境

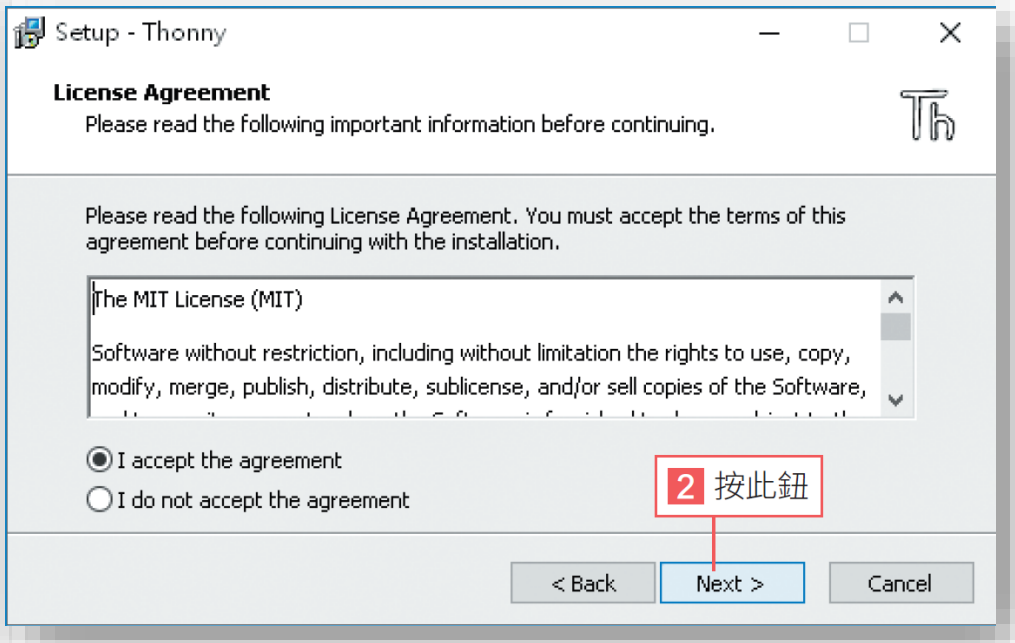
thonny.org

在各位桌面上的 FM617A 資料夾下有安裝檔案
課程開始前現場工作人員應該也幫大家裝好了

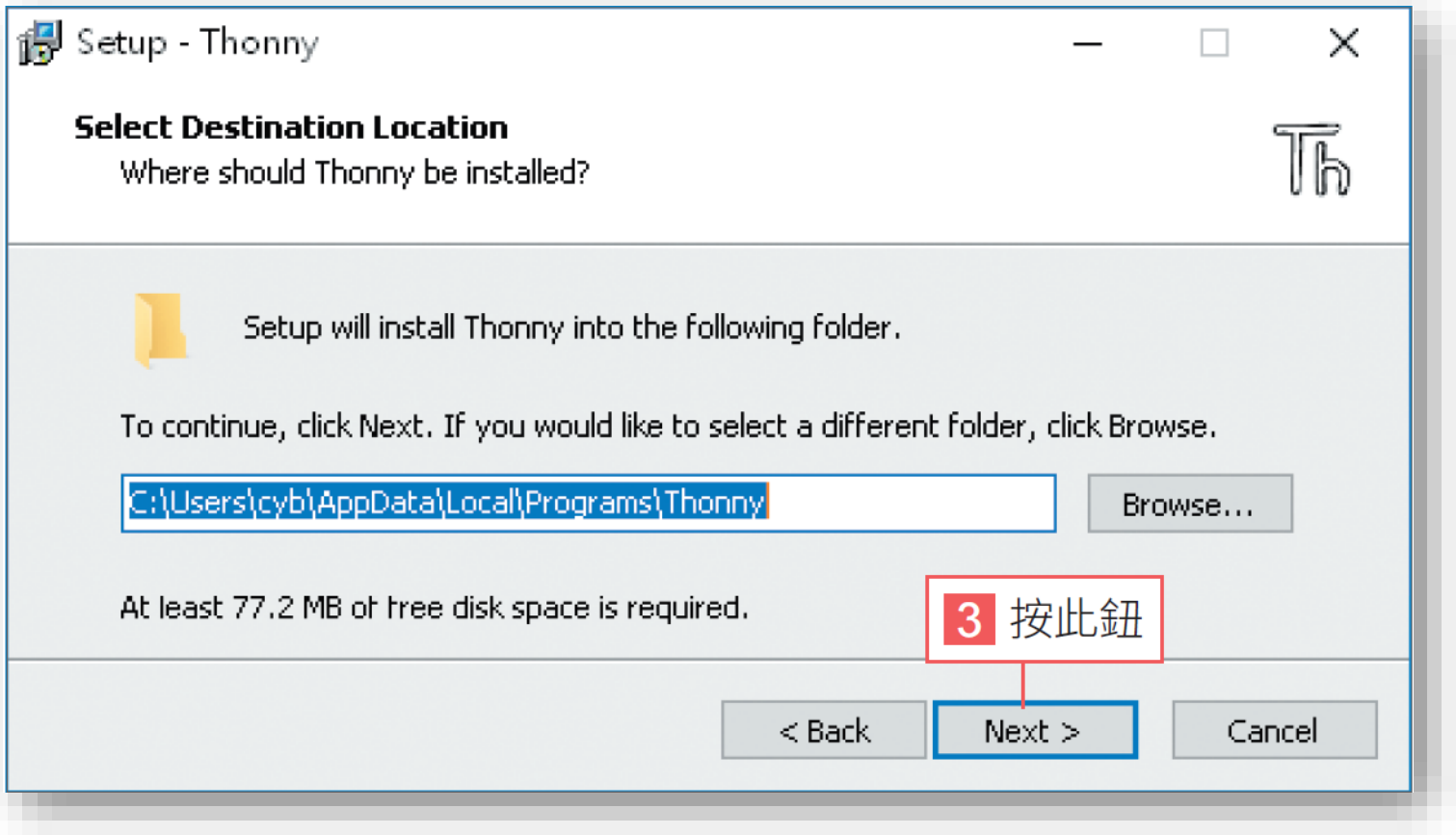


Thonny 程式開發環境

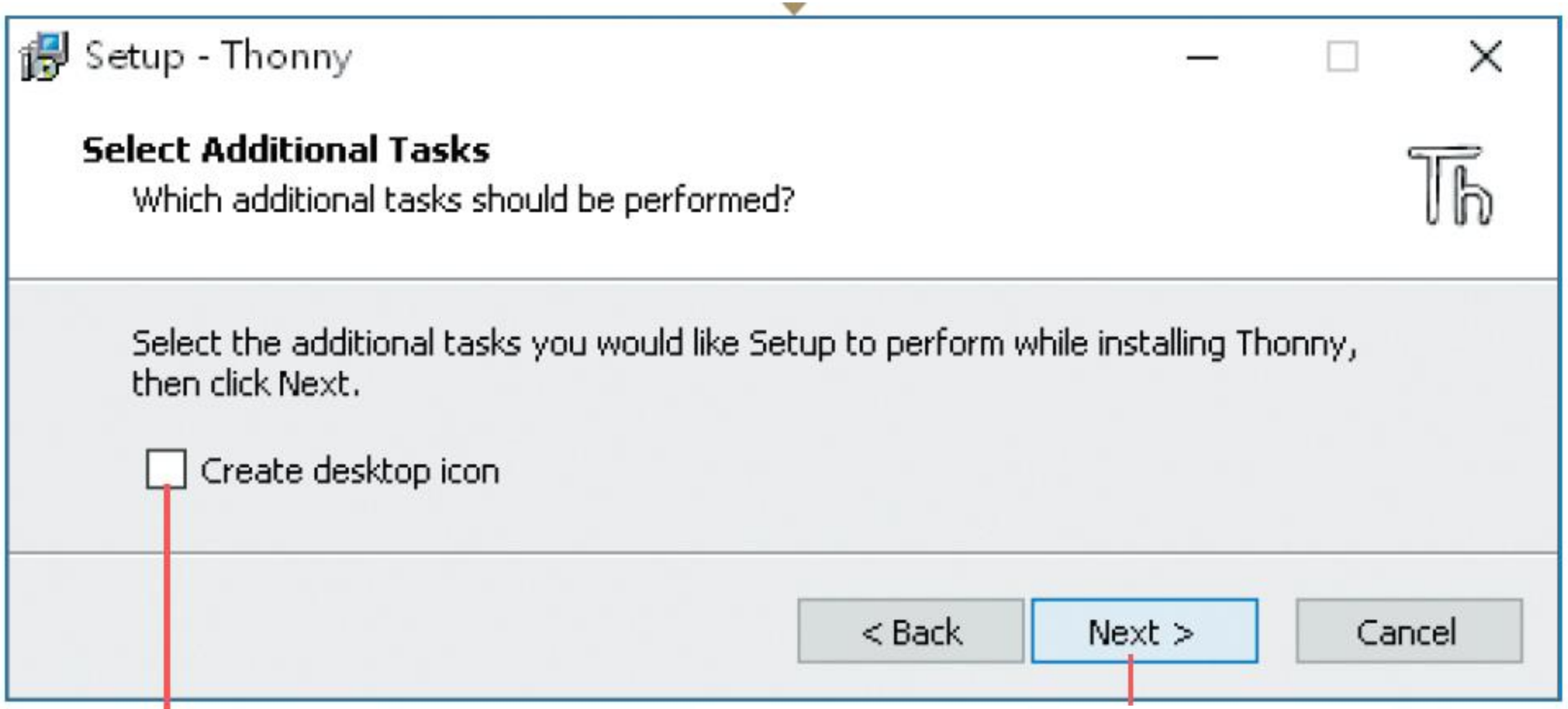




Thonny 程式開發環境

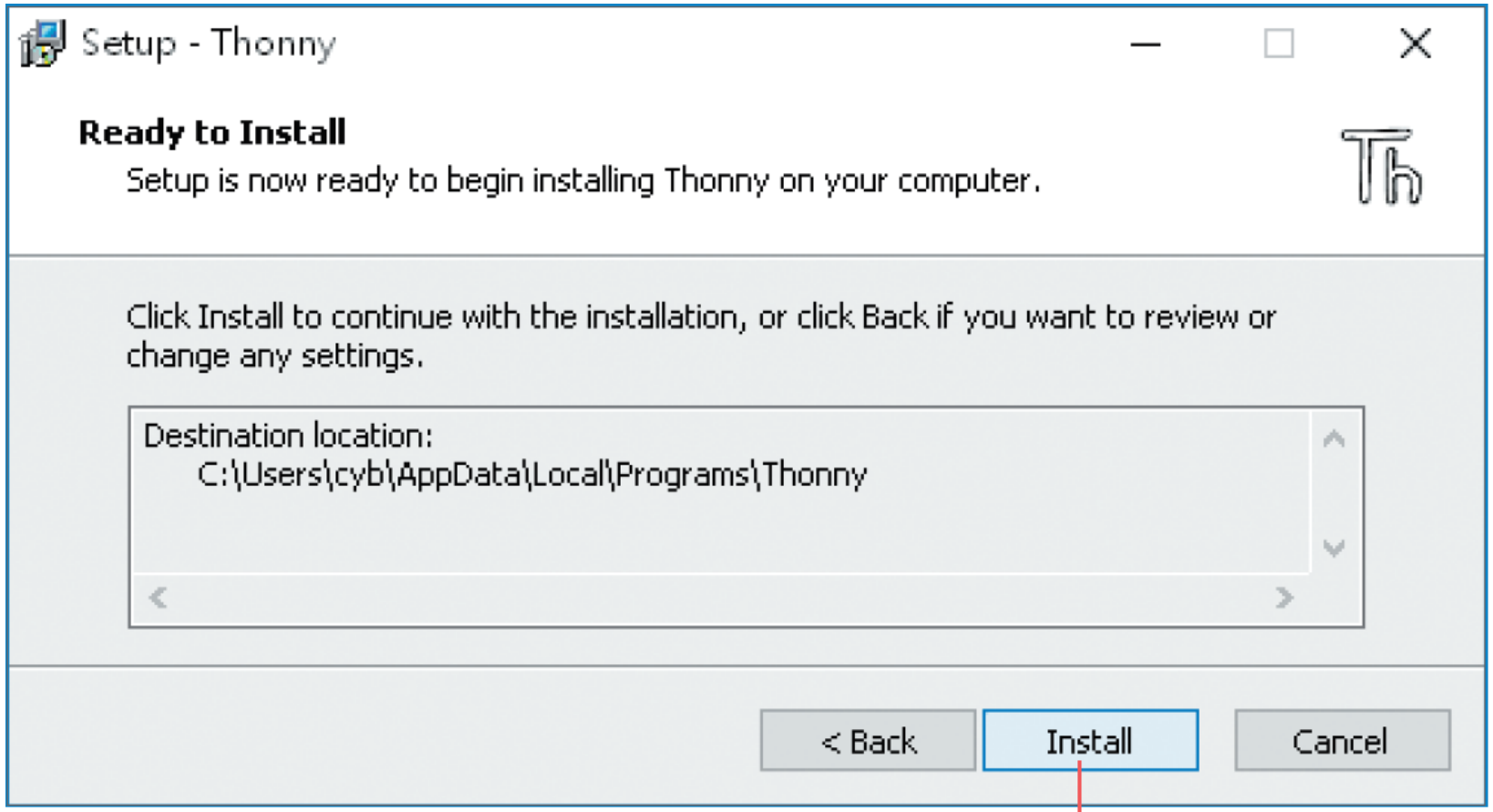


Thonny 程式開發環境

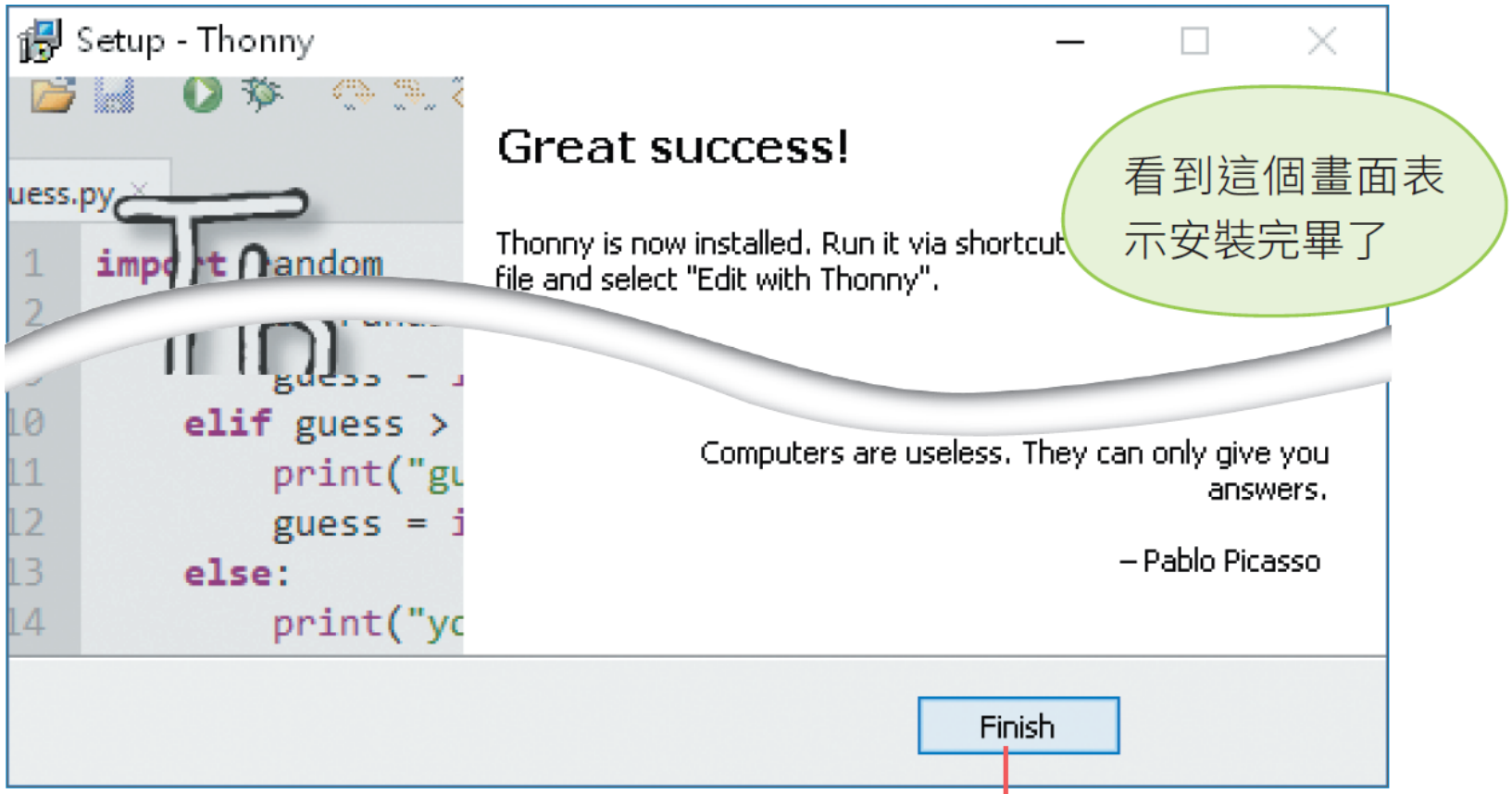


勾選這個項目在桌面建立捷徑

Thonny 程式開發環境

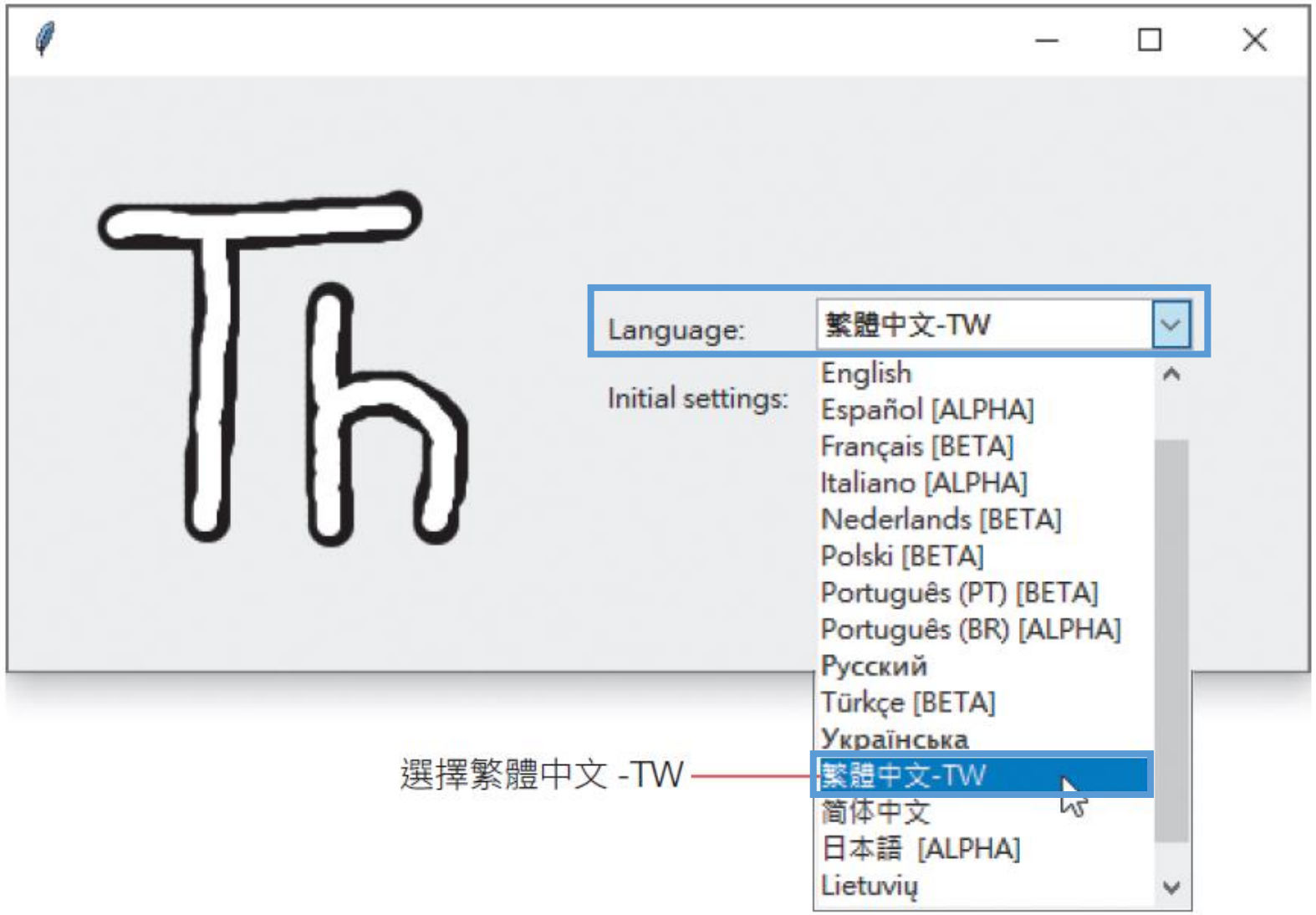


Thonny 程式開發環境



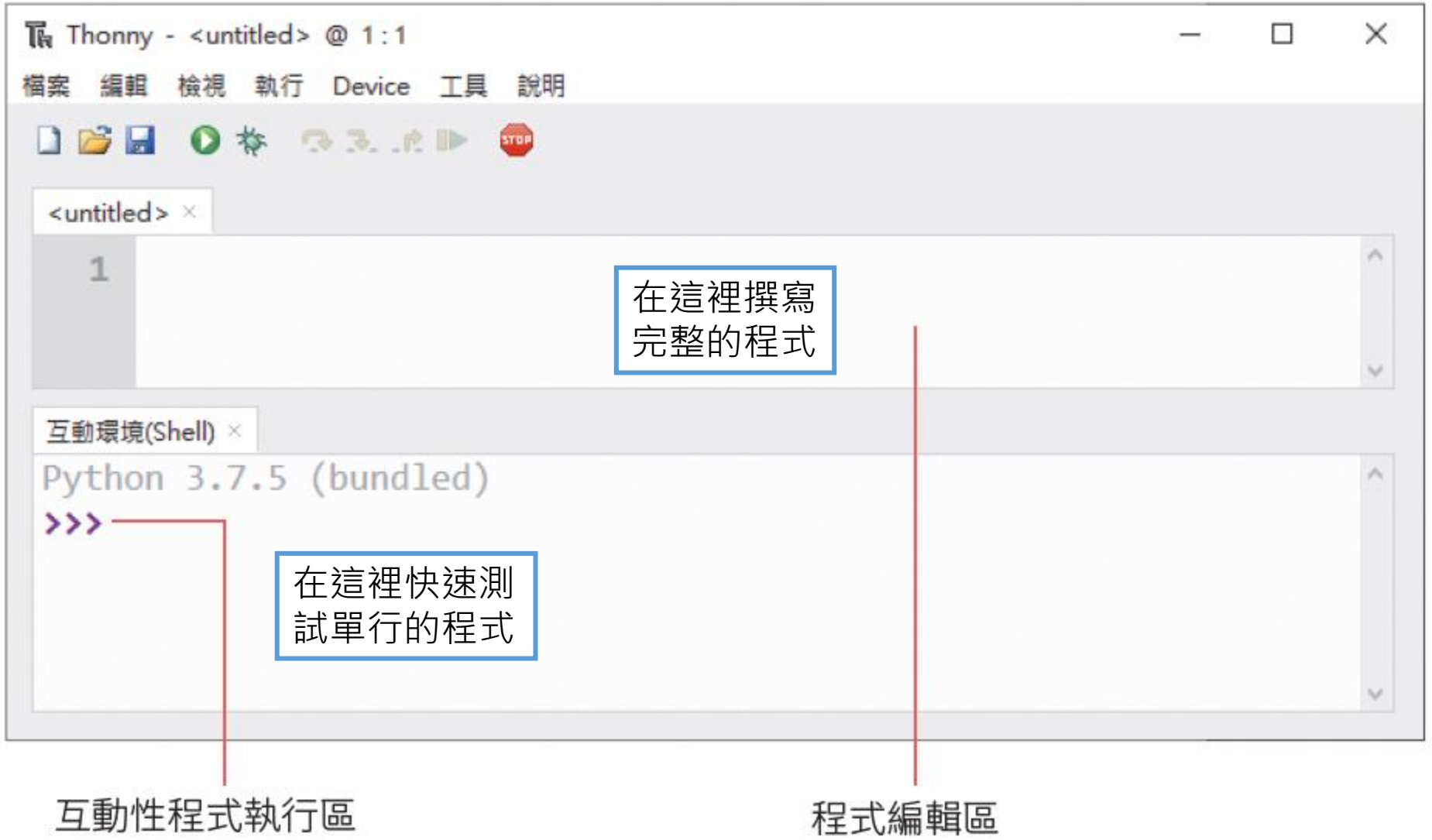
Thonny 程式開發環境

點擊桌面捷徑

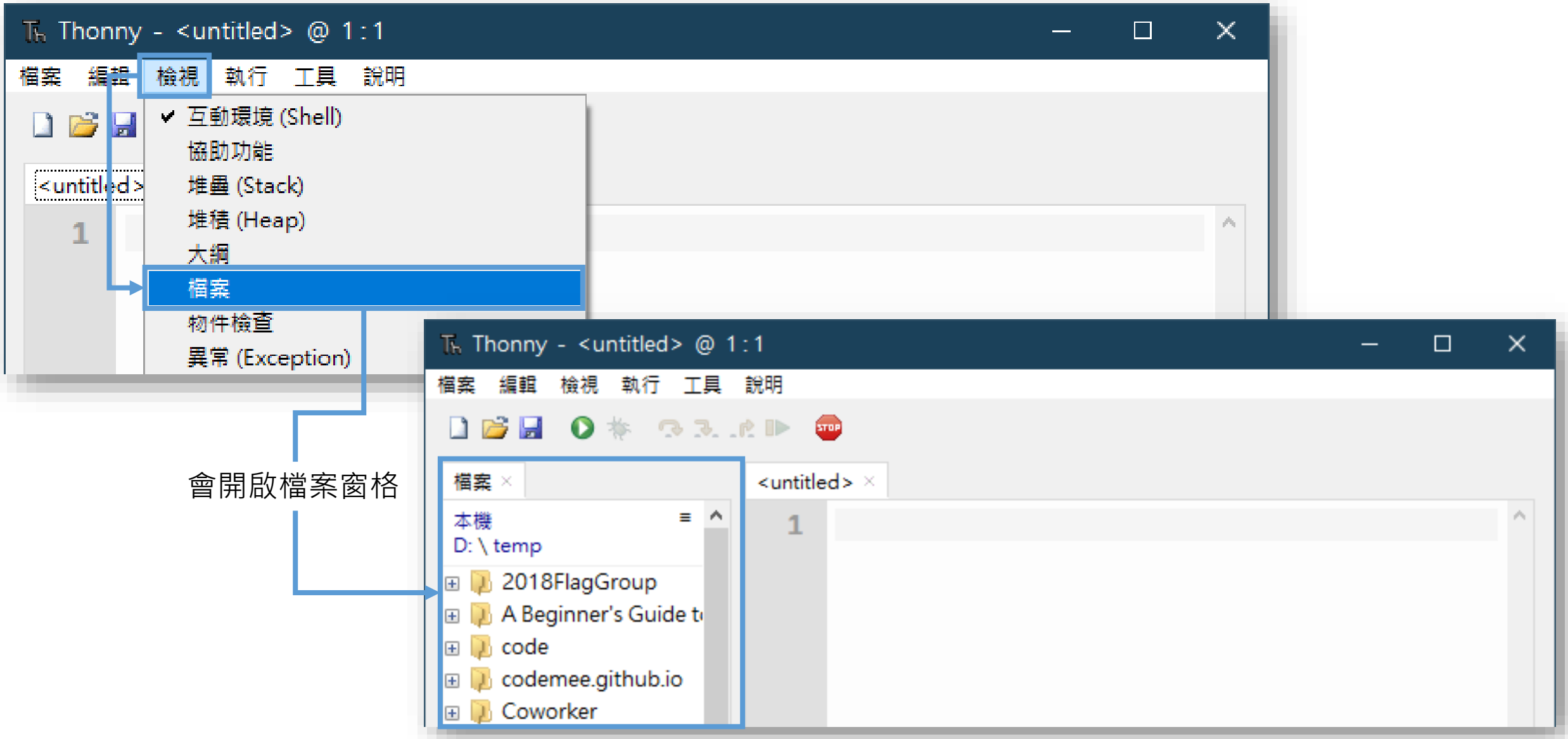


Thonny 程式開發環境

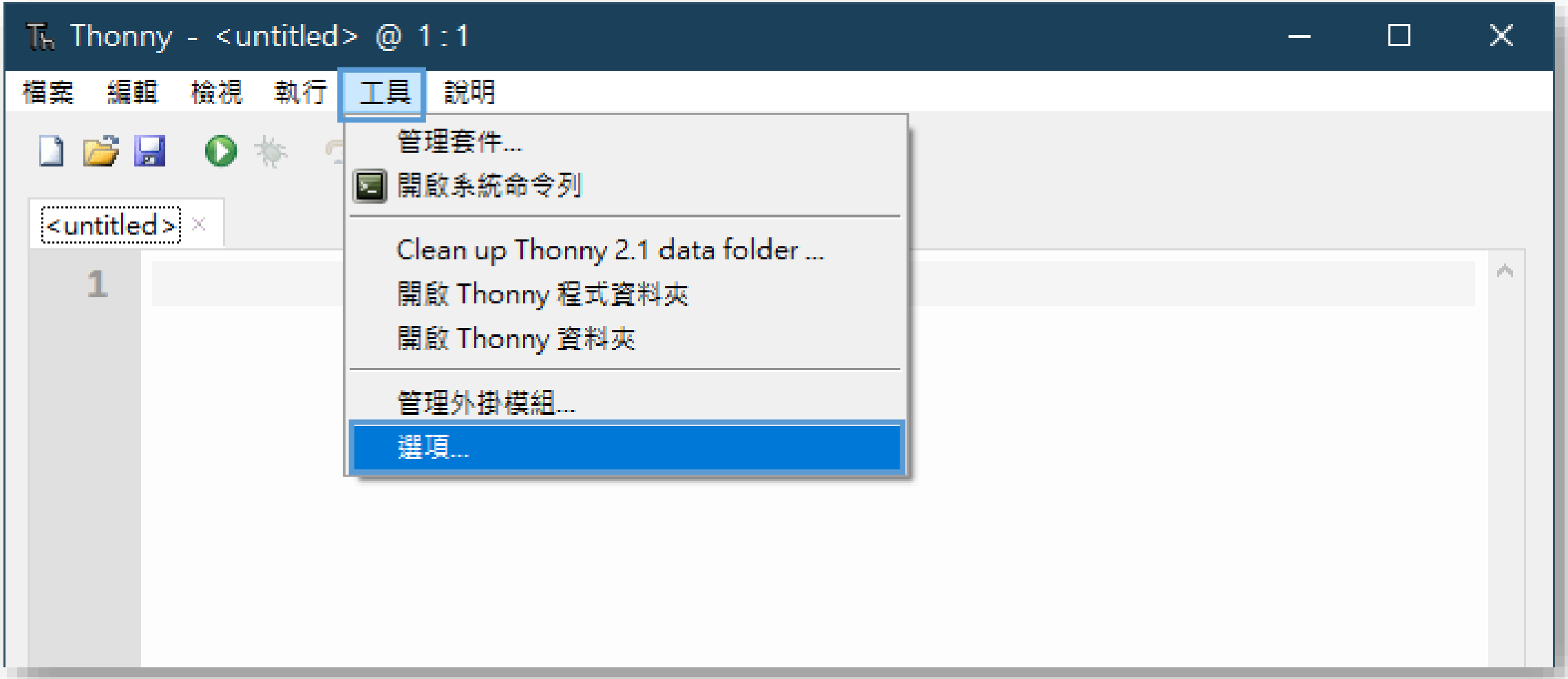
Thonny 程式開發環境



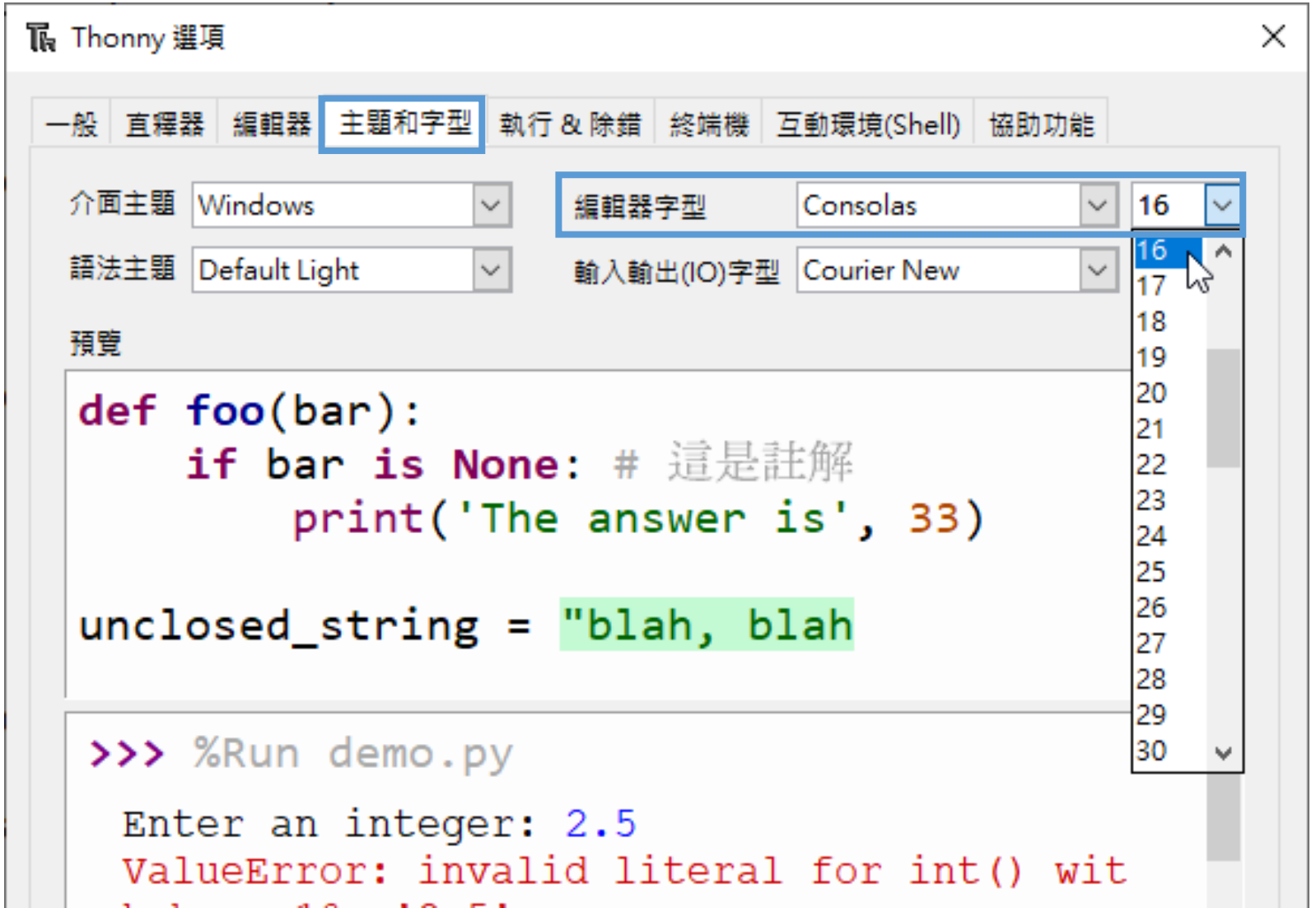
Thonny 程式開發環境



設定開發環境

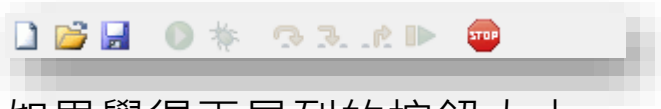

Thonny - <untitled> @ 1 : 1
檔案
編輯
檢視
執行
工具
說明
管理套件...
開啟系統命令列
Clean up Thonny 2.1 data folder ...
開啟 Thonny 程式資料夾
開啟 Thonny 資料夾
管理外掛模組...
選項...
<untitled>
1

調整字型大小

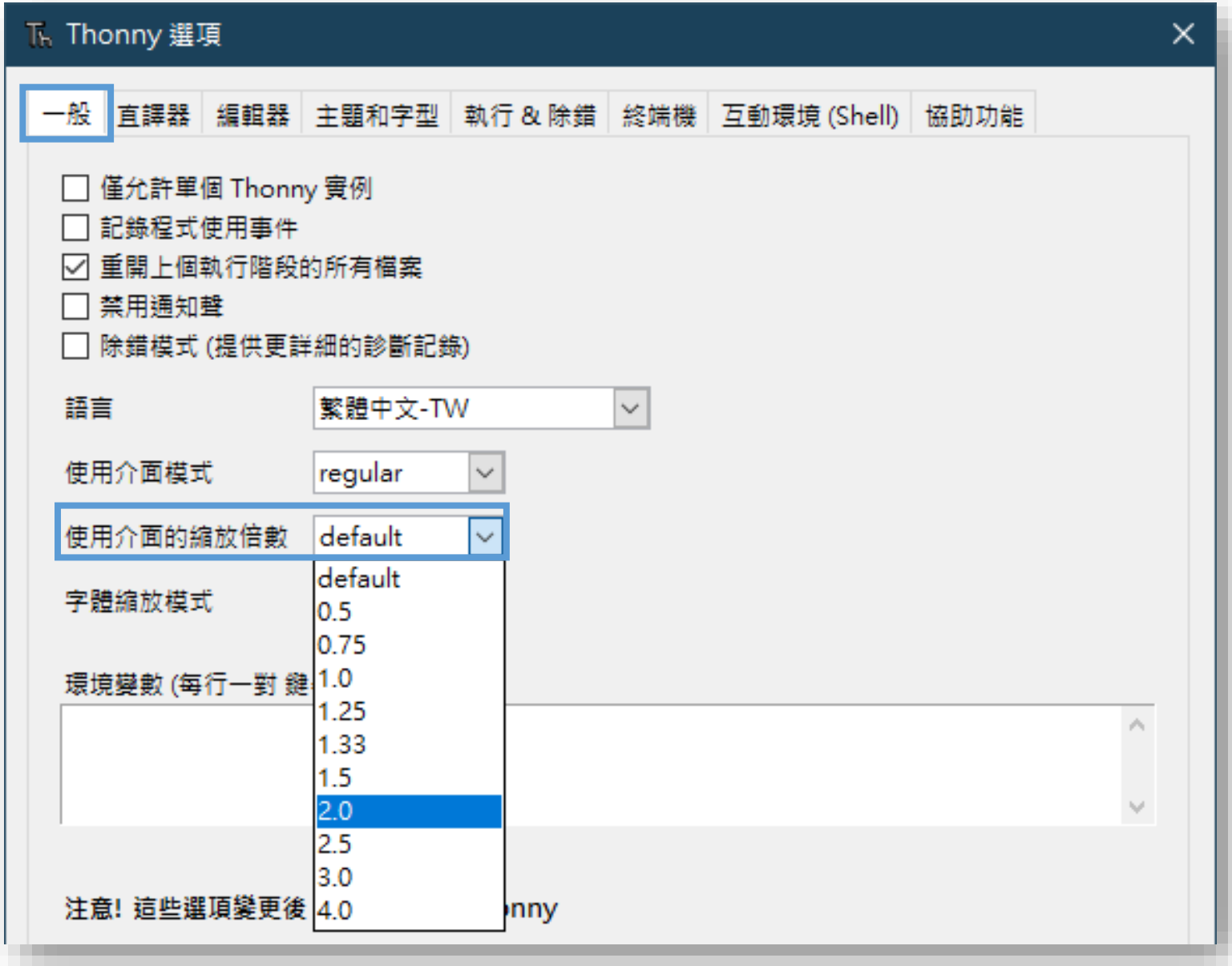


或是按住Ctrl + 滑鼠滾輪
調整字體大小

調整使用介面按鈕大小



如果覺得工具列的按鈕太小



這個設定需要重新執行 Thonny 才會生效

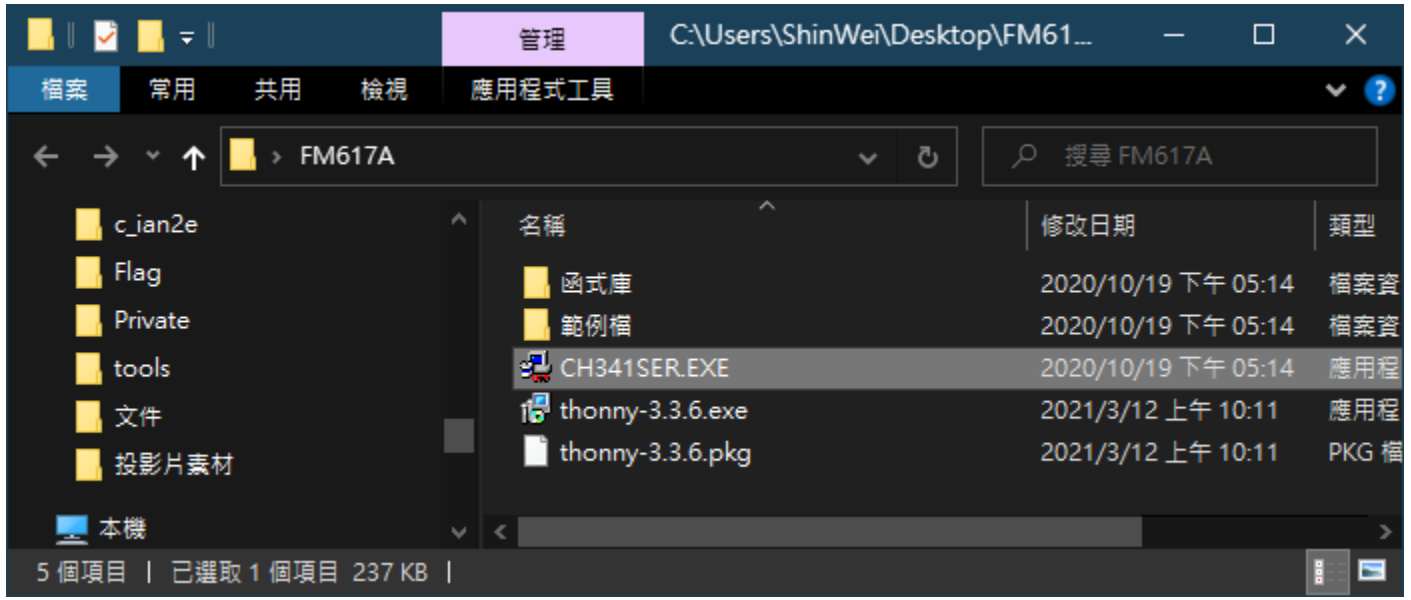
前置作業

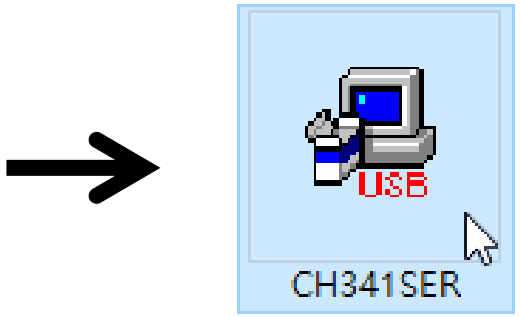
安裝CH340驅動程式

Mac 電腦不用安裝驅動程式！！

本套件範例程式檔下載網址：
https://www.flag.com.tw/download.asp?FM617A

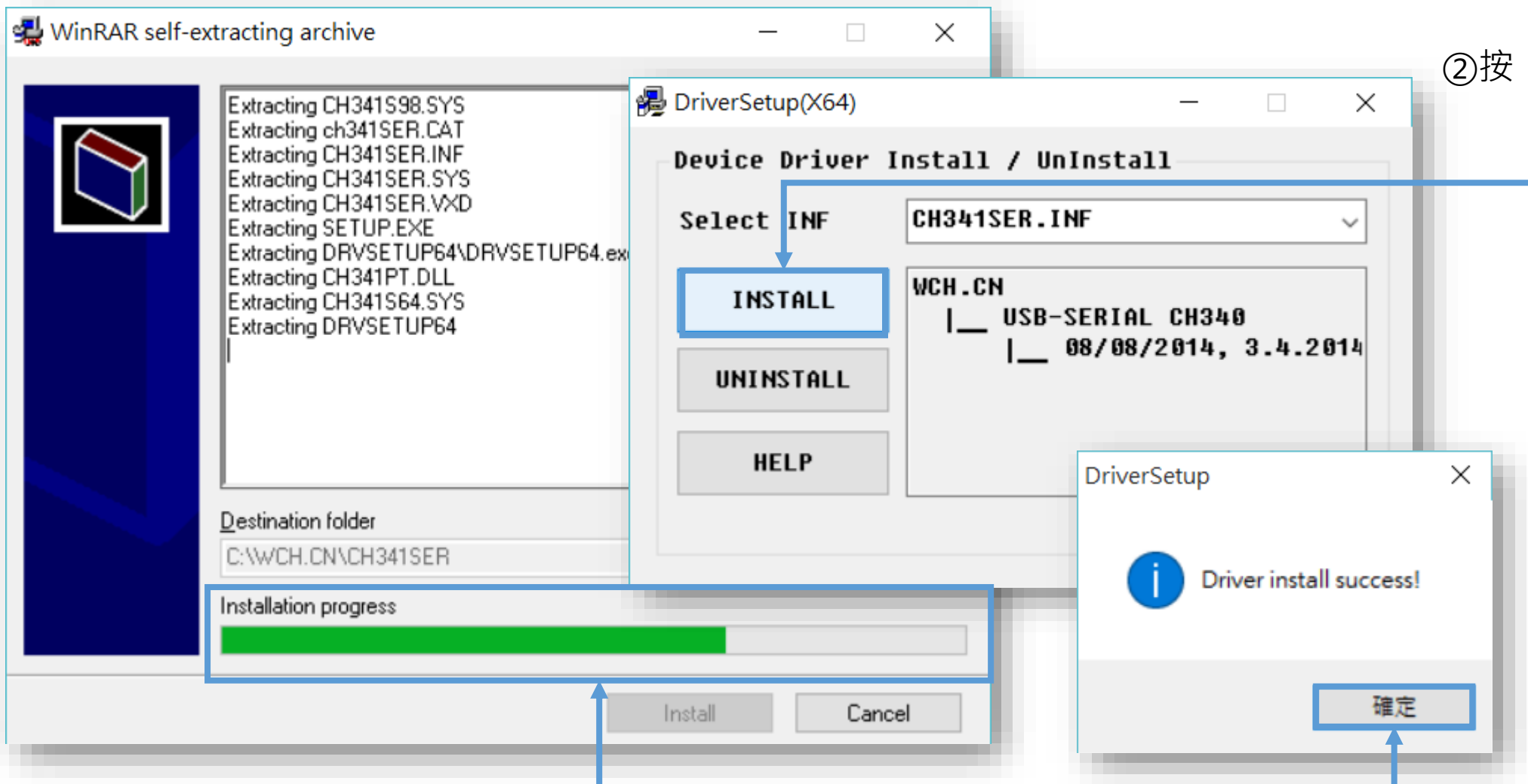
在各位桌面上的 FM617A 資料夾下有安裝檔案
課程開始前現場工作人員應該也幫大家裝好了





執行 CH341SER

完成驅動程式安裝步驟



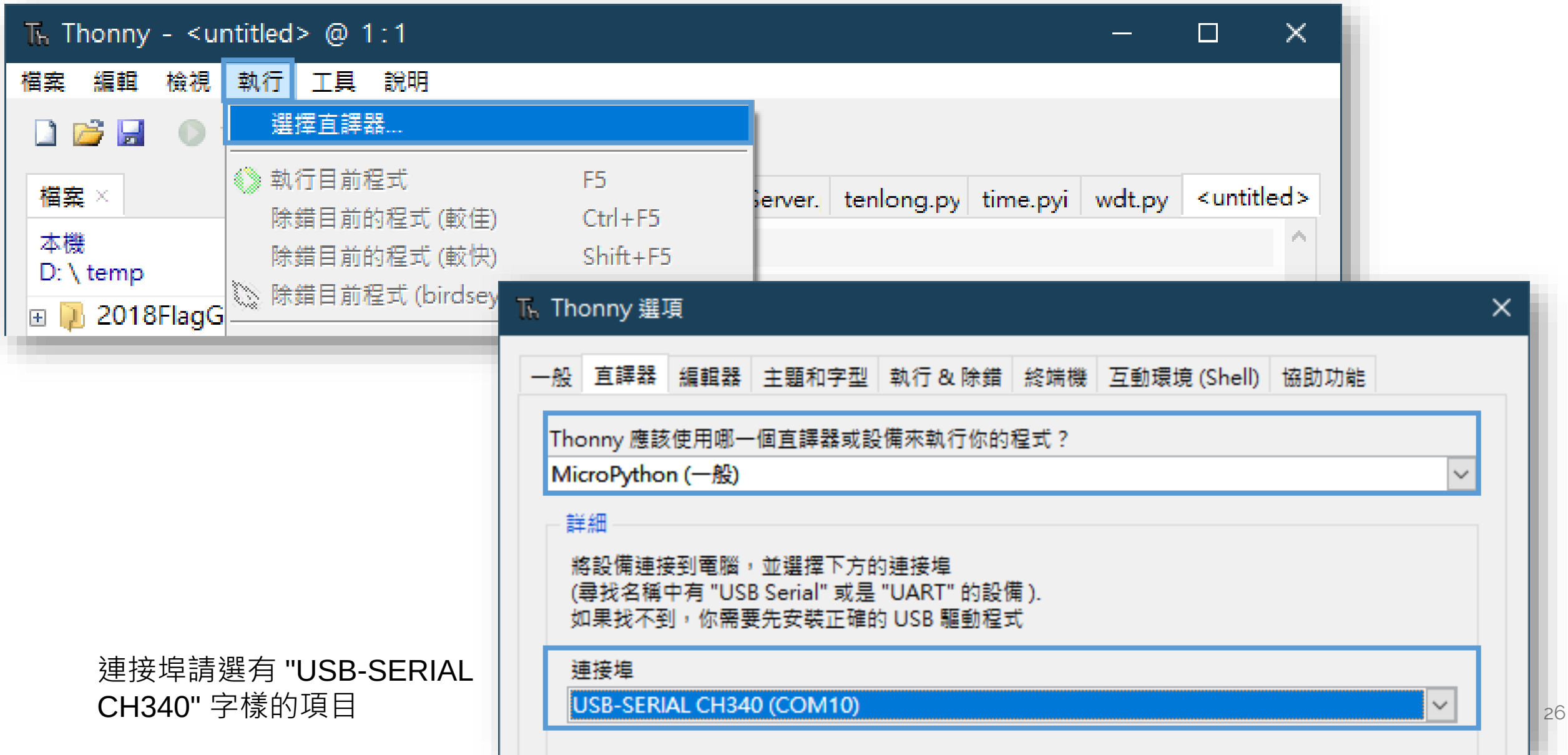
②按『INSTALL』開始安裝

①會先解開驅動程式安裝檔案

③安裝完成

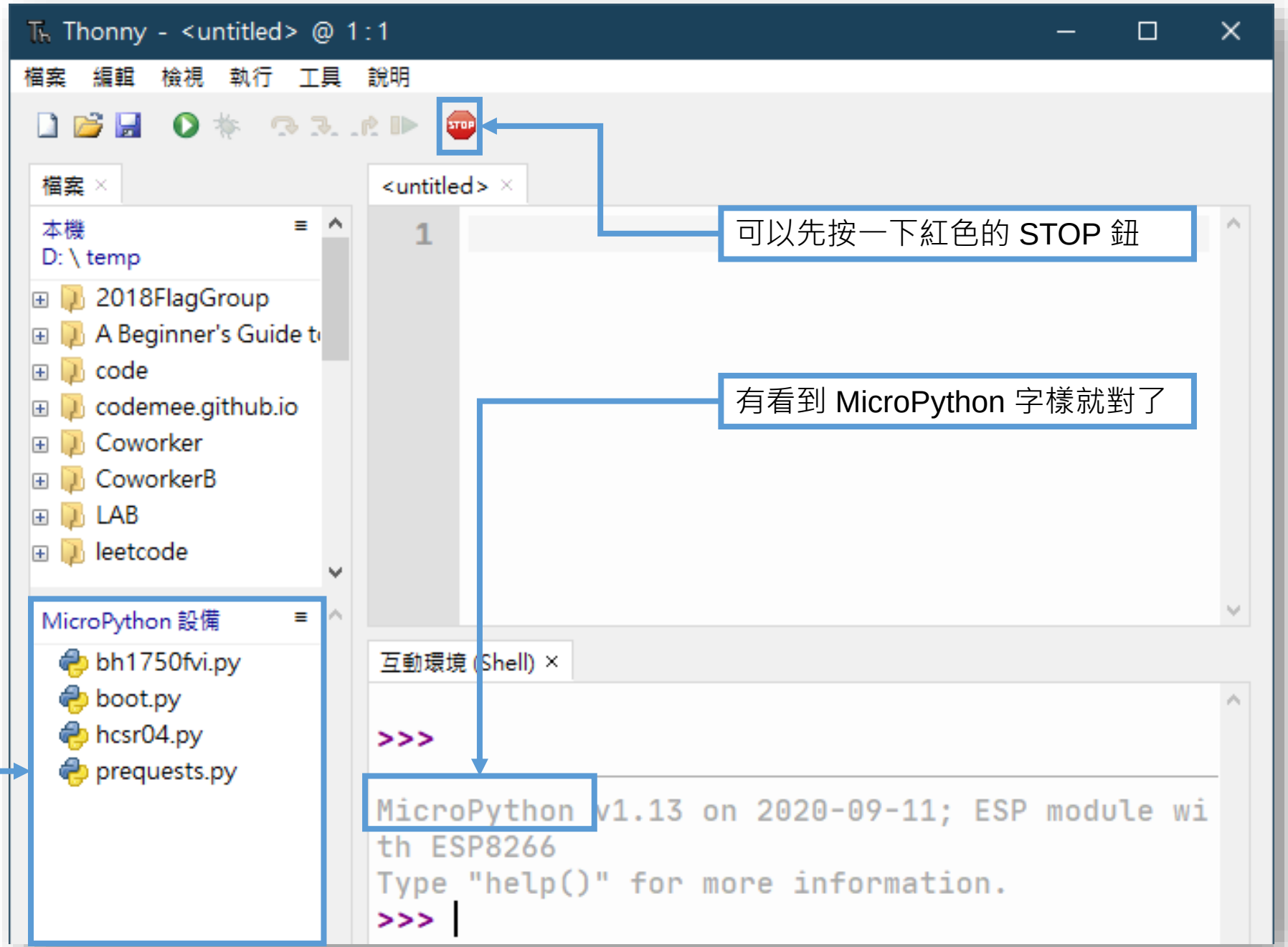
連接 D1 mini 開發板

動手做 - 連接 D1 mini 開發板



連接埠請選有 "USB-SERIAL CH340" 字樣的項目

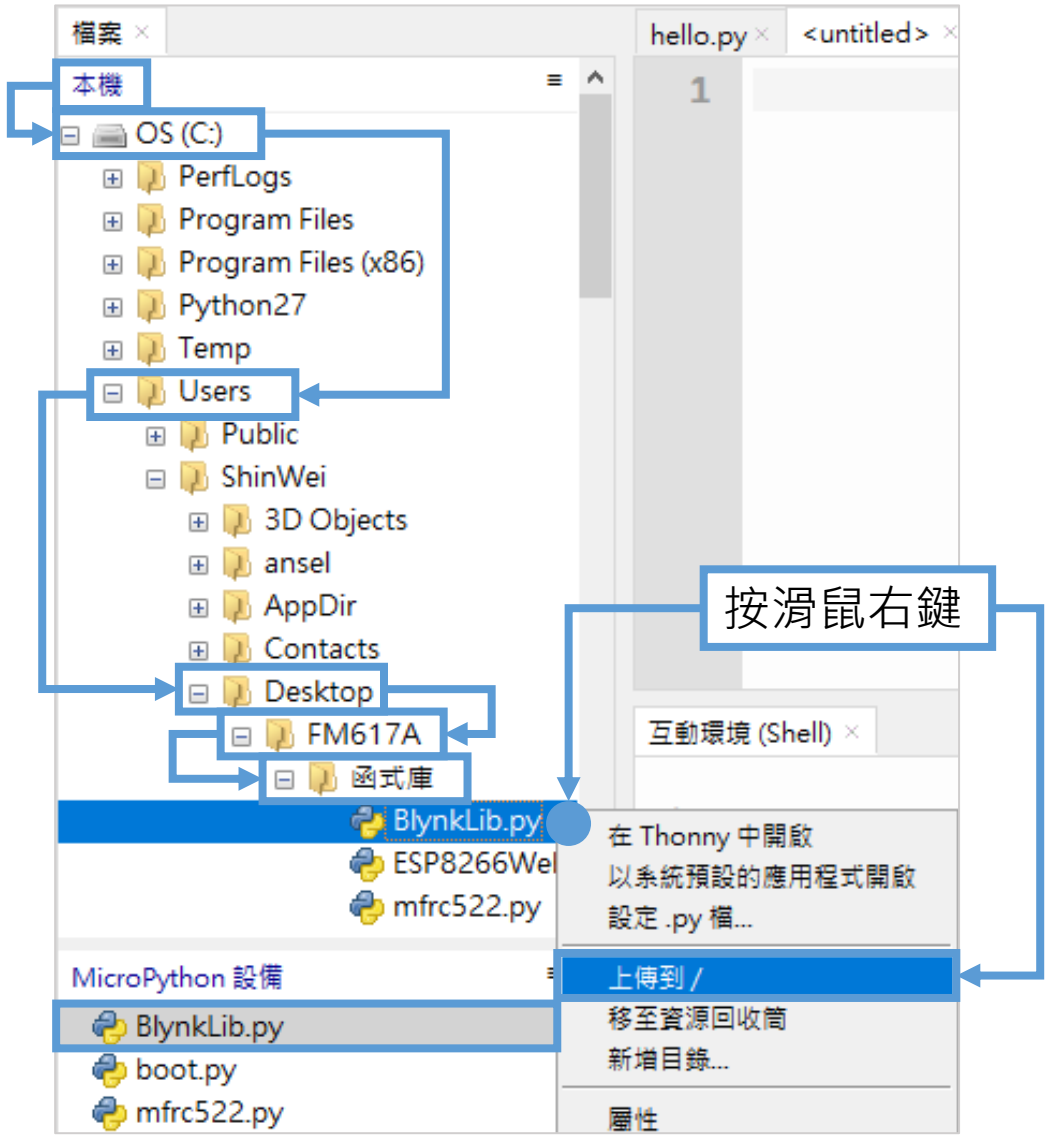
也會出現『MicroPython 設備窗格』

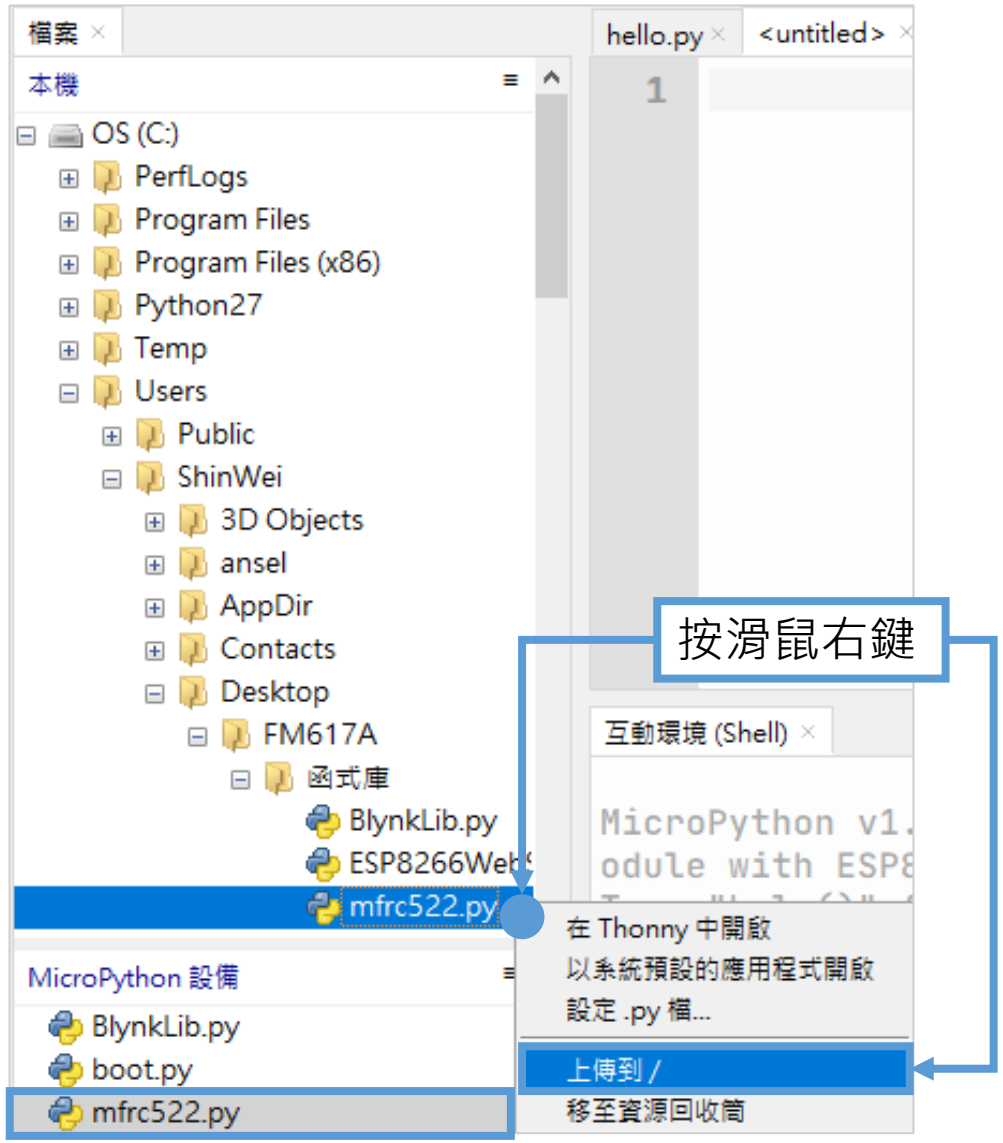


前置作業

安裝程式庫

如果使用 Thonny 3.1.2
請參考手冊上操作步驟





01 Python 程式語言簡介

物件、資料型別、變數、匯入模組

Python 物件

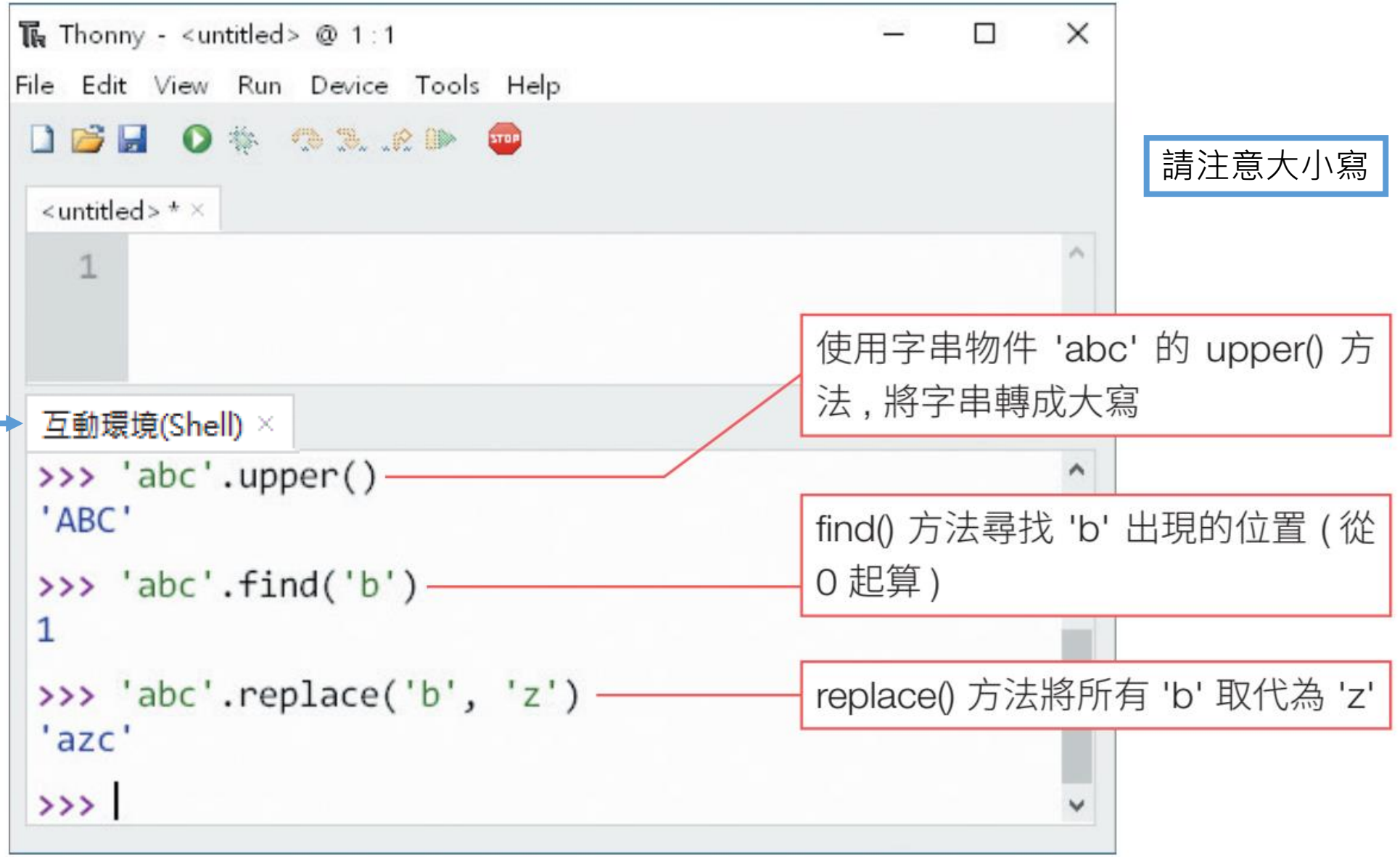
前面提到 Python 的語法簡潔且口語化，近似用英文寫作，一般我們寫句子的時候，會以主詞搭配動詞來成句。用 Python 寫程式的時候也是一樣，Python 程式是以『**物件**』(Object) 為主導，而物件會有『**方法**』(method)，這邊的物件就像是句子的主詞，方法類似動詞，請參見下面的比較表格：

寫作文章	寫 Python 程式	
車子	car	← car 物件
車子向前進	car.go()	← car 物件的 go 方法

Python 物件

物件的方法都是用點號 . 來連接，您可以將 . 想成『的』，所以 car.go() 便是 car **的** go() 方法。

方法的後面會加上括號 ()，有些方法可能會需要額外的資訊參數，假設車子向前進需要指定速度，此時速度會放在方法的括號內，例如 car.go(100)，這種額外資訊就稱為 **『參數』**。若有多個參數，參數間以英文逗號 " , " 來分隔。



請在互動環境測試

請注意大小寫

使用字串物件 'abc' 的 upper() 方法，將字串轉成大寫

find() 方法尋找 'b' 出現的位置（從 0 起算）

replace() 方法將所有 'b' 取代為 'z'

⚠ 不同的物件會有不同的方法，本書稍後介紹各種物件時，會說明該物件可以使用的方法。

資料型別

上面我們使用了字串物件來練習方法，Python 中只要用成對的 " 或 ' 引號括起來的就會自動成為字串物件，例如 "abc"、'abc'。

除了字串物件以外，我們寫程式常用的還有整數與浮點數（小數）物件，例如 111 與 11.1。所以數字如果沒有用引號括起來，便會自動成為整數與浮點數物件，若是有括起來，則是字串物件：

```
>>> 111 + 111
222

>>> '111' + '111'
'111111'
```

我們可以看到雖然都是 111，但是整數與字串物件用 + 號相加的動作會不一樣，這是因為其資料的種類不相同。這些資料的種類，在程式語言中我們稱之為**『資料型別』**(Data Type)。

寫程式的時候務必要分清楚資料型別，兩個資料若型別不同，便可能會導致程式無法運作：

```
>>> 111 + '111'      ◄—— 不同型別的資料相加發生錯誤
  Traceback (most recent call last):
    File "<pyshell>", line 1, in <module>
  TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

變數

在 Python 中，**變數**就像是掛在物件上面的名牌，幫物件取名之後，即可方便我們識別物件，其語法為：

```
變數名稱 = 物件
```

例如：

```
>>> n1 = 123456789 ←—— 將整數物件 123456789 取名為 n1
>>> n2 = 987654321 ←—— 將整數物件 987654321 取名為 n2
>>> n1 + n2        ←—— n1 + n2 實際上便是 123456789 + 987654321
1111111110
```

變數命名時只用**英**、**數字**及**底線**來命名，而且第一個字不能是數字。

內建函式

函式 (function) 是一段預先寫好的程式，可以方便重複使用，而程式語言裡面會預先將經常需要的功能以函式的形式先寫好，這些便稱為**內建函式**，您可以將其視為程式語言預先幫我們做好的常用功能。

前面第一章用到的 print() 就是內建函式，其用途就是將物件或是某段程式執行結果顯示到螢幕上：

```
>>> print('abc')   <---- 顯示物件
  abc

>>> print('abc'.upper())   <---- 顯示物件方法的執行結果
  ABC

>>> print(111 + 111)   <---- 顯示物件運算的結果
  222
```

匯入模組

既然內建函式是程式語言預先幫我們做好的功能，那豈不是越多越好？理論上內建函式越多，我們寫程式自然會越輕鬆，但實際上若內建函式無限制的增加後，就會造成程式語言越來越肥大，導致啟動速度越來越慢，執行時佔用的記憶體越來越多。

為了取其便利去其缺陷，Python 特別設計了**模組** (module) 的架構，將同一類的函式打包成模組，預設不會啟用這些模組，只有當需要的時候，再用**匯入 (import)** 的方式來啟用。

模組匯入的語法有兩種，請參考以下範例練習：

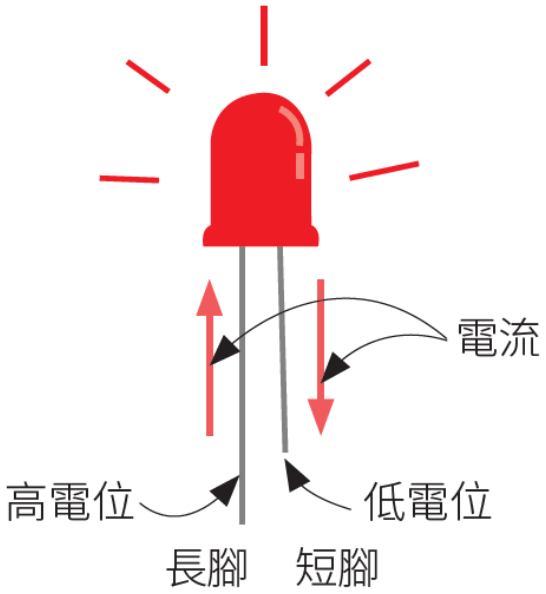
```
>>> import time   ◀—— 匯入時間相關的 time 模組
>>> time.sleep(3)◀—— 執行 time 模組的 sleep() 函式, 暫停 3 秒

>>> from time import sleep ◀—— 從 time 模組裡面匯入 sleep() 函式
>>> sleep(5)   ◀—— 執行 sleep() 函式, 暫停 5 秒
```

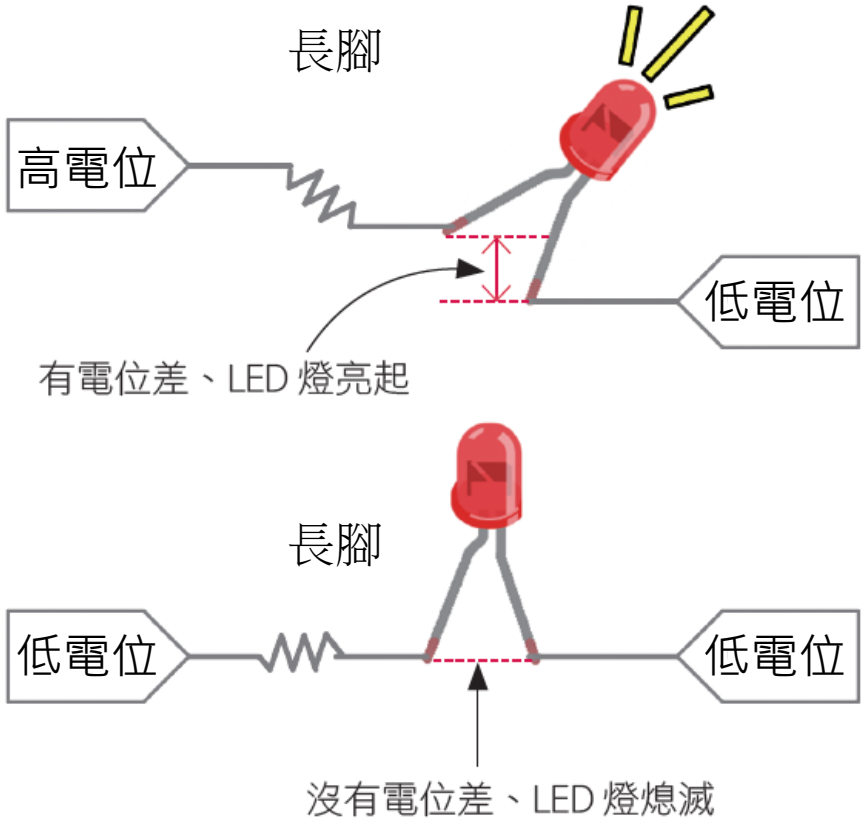
01 控制 LED 亮暗 - 數位輸出

電子電路基礎

LED 燈

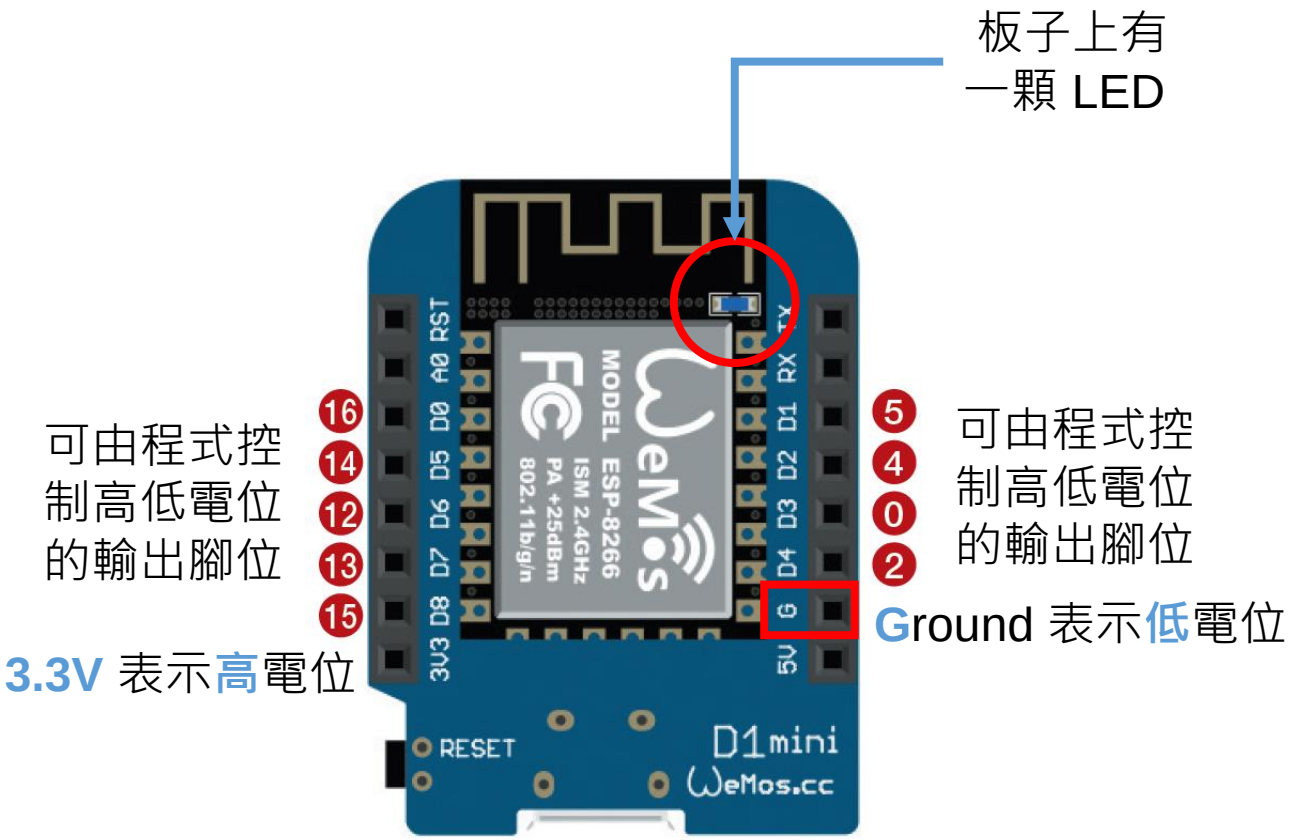
電跟水一樣
由高往低流

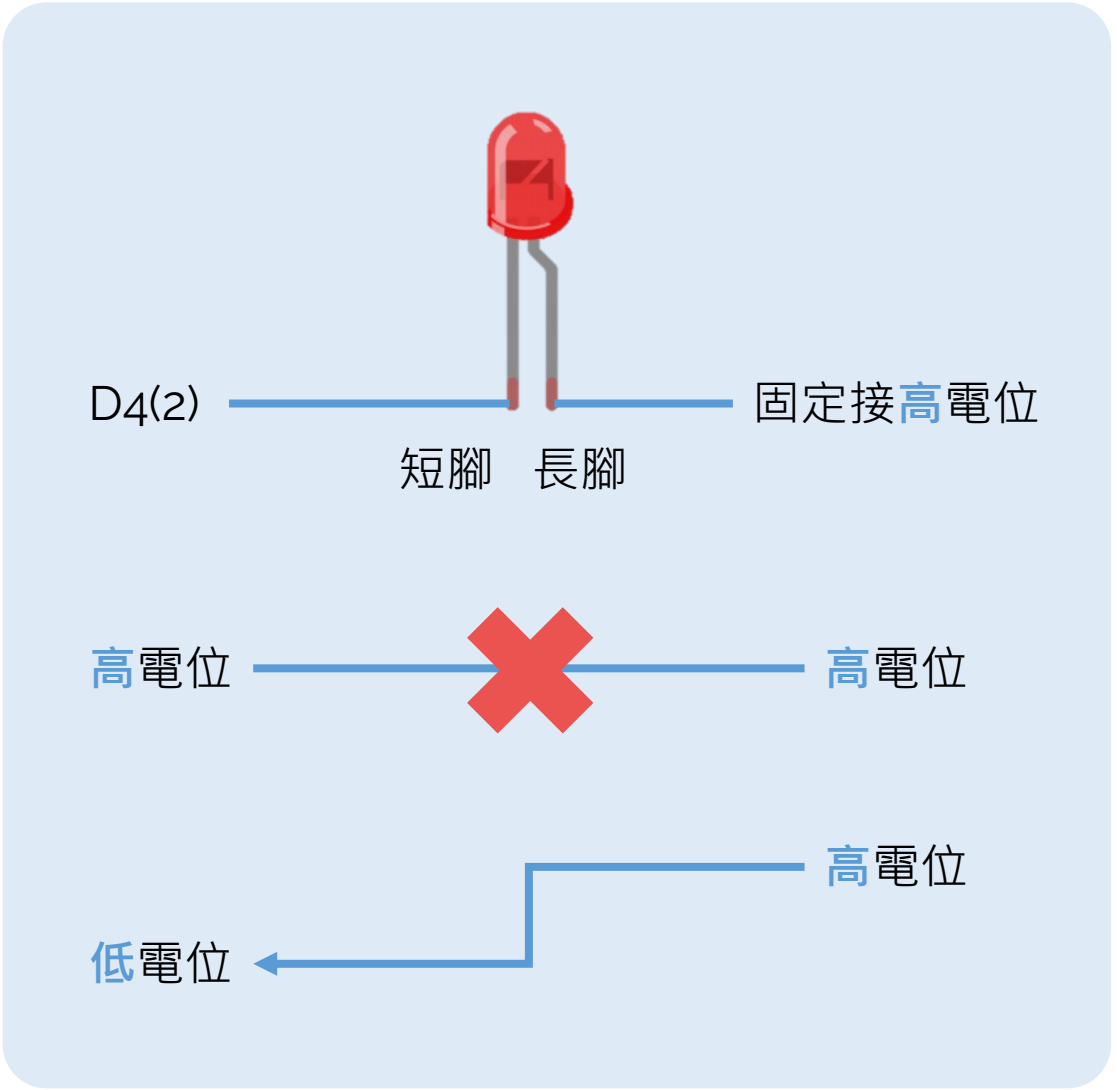




圖中各元件的高低關係為彼此的電位關係

閃爍 LED 燈


板子上有
一顆 LED
可由程式控
制高低電位
的輸出腳位
16
14
12
13
15
3.3V 表示高電位
5
4
0
2
可由程式控
制高低電位
的輸出腳位
Ground 表示低電位
RESET
D1mini
WeMos.cc


D4(2)
固定接高電位
短腳
長腳
高電位
高電位
高電位
低電位

Lab01

點亮/熄滅 LED

實驗目的	用 Python 程式控制 D1 mini 腳位, 藉此點亮或熄滅該腳位連接的 LED 燈。
材料	• D1 mini

線路圖

無需接線。

控制 D4 腳位的電位高低

請在**互動窗格**內測試

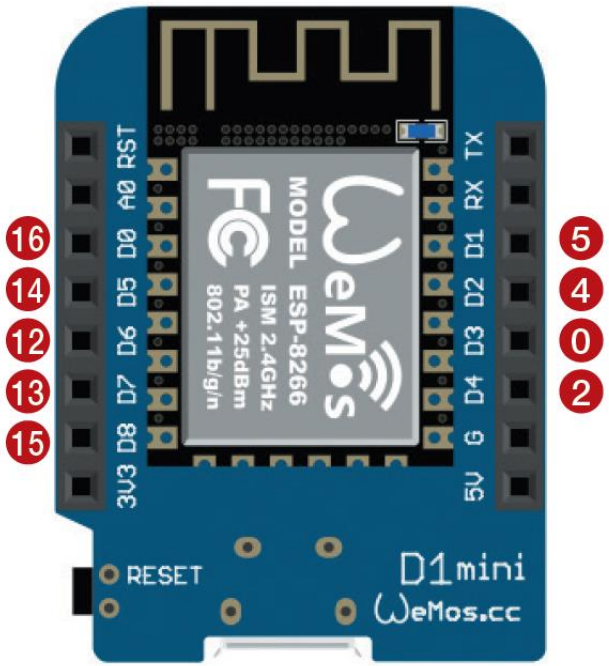
用 machine 模組的 Pin 功能

```
互動環境 (Shell)
>>> from machine import Pin
>>>
>>> led = Pin(2, Pin.OUT)
>>>
>>> led.value(0)
>>>
```

表示要控制的腳位編號

表示要控制高低電位

1：高電位
0：低電位



請注意大小寫

到這裡燈應該就亮起來了
現在考一考自己讓燈熄滅

02 閃爍 LED

Python 流程控制與區塊縮排

Lab02

閃爍 LED

實驗目的	用 Python 的 while 迴圈重複執行 LED 的控制程式, 使其每 0.5 秒閃爍一次。
材料	• D1 mini

線路圖

無需接線。

動手做：閃爍 LED 燈

Thonny - C:\Users\ShinWei\Desktop\tets00.py @ 11 : 20

檔案 編輯 檢視 執行 工具 說明

打完後請按執行鈕或 F5 鍵

是寫在上面的編輯區喔！！

tets00.py

```
1  from machine import Pin
2  import time
3
4  # 建立物件 (# 開始是註解)
5  led = Pin(2, Pin.OUT)
6
7  while True:
8      led.value(0)
9      time.sleep(0.5)
10     led.value(1)
11     time.sleep(0.5)
```

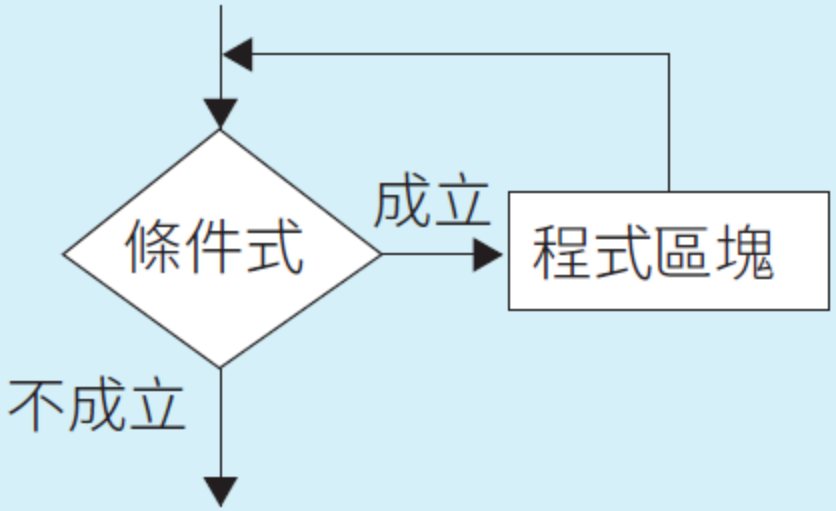
表示無條件成立

一定要加上冒號

這幾行一定要空相同格數 (Thonny 會自動幫你空**官方推薦的 4 格**)

while 條件式：

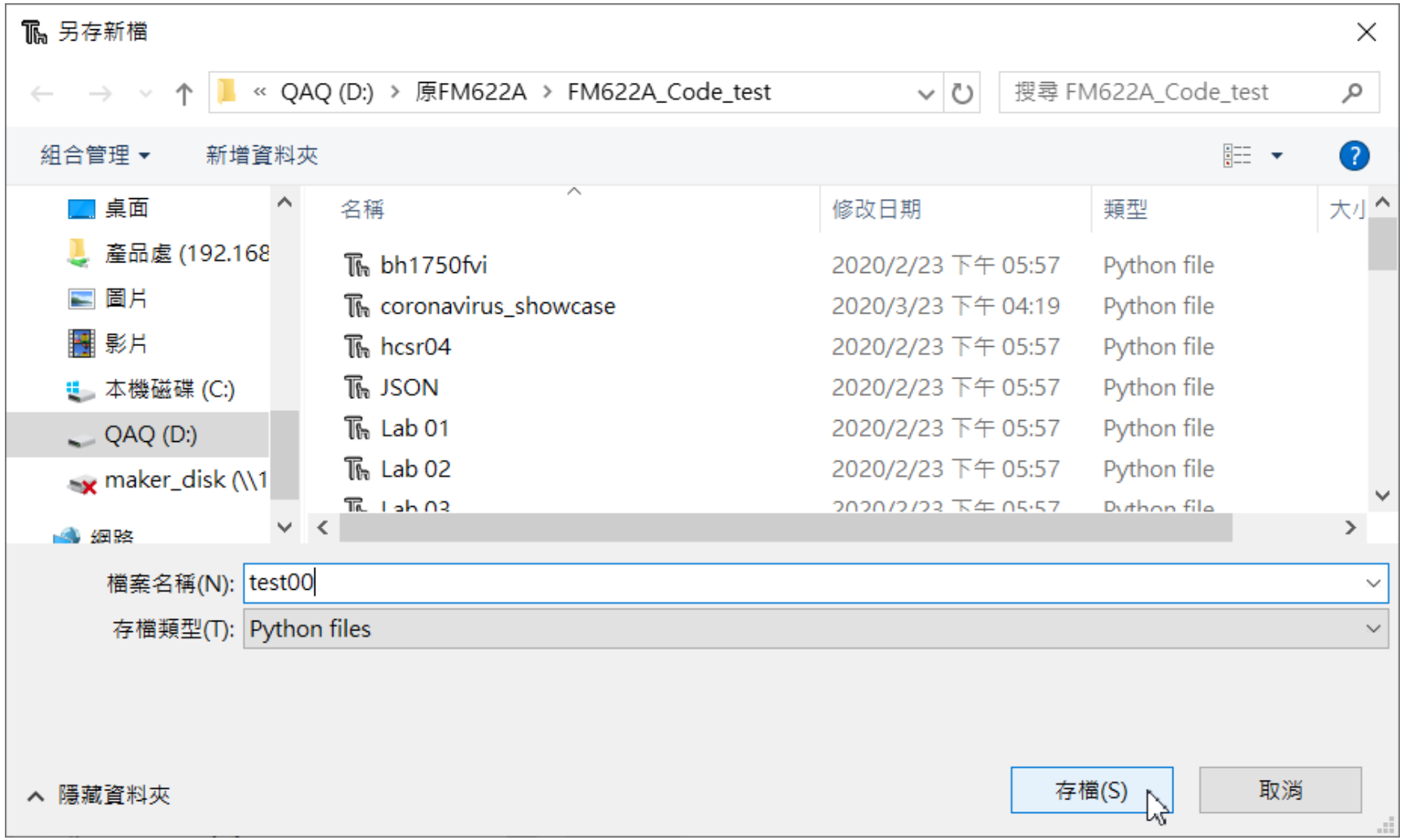
程式區塊



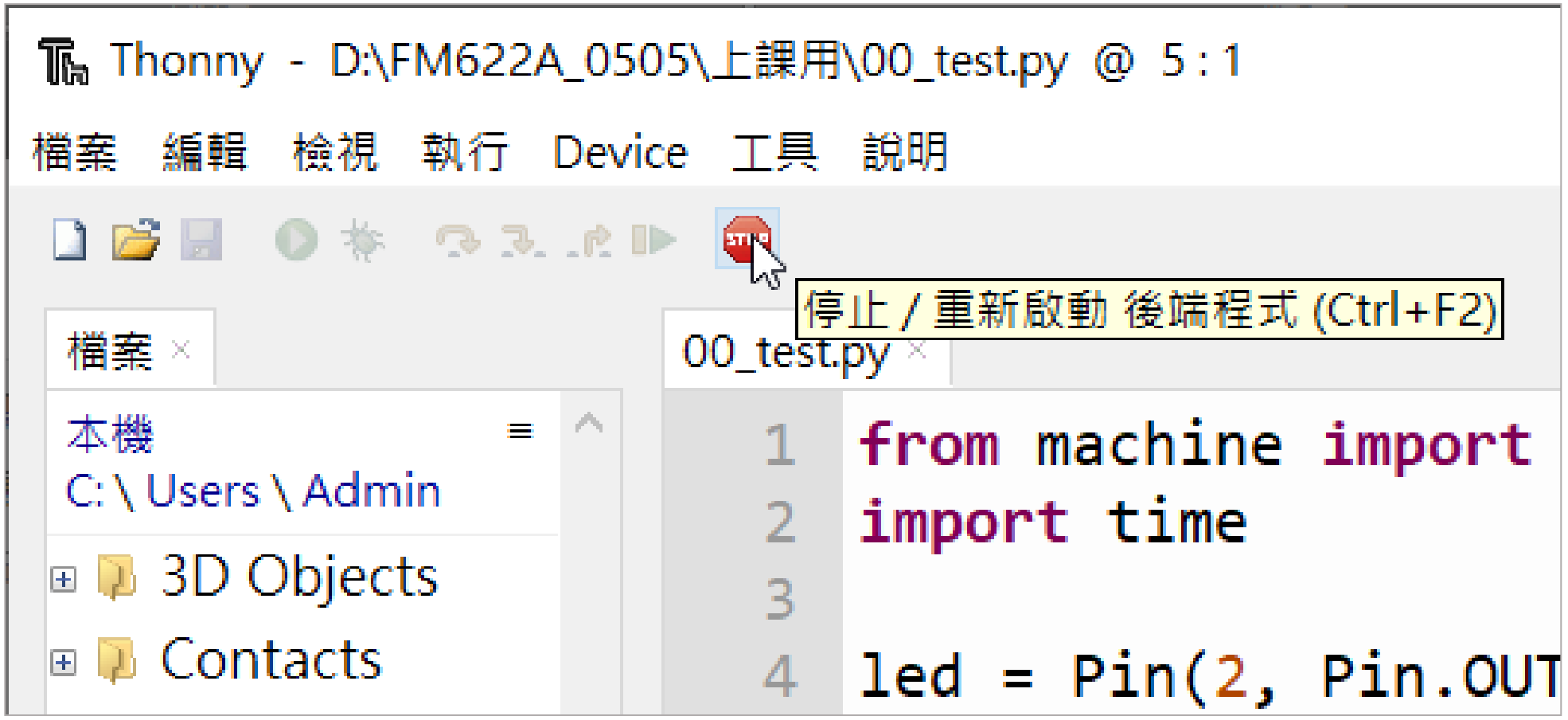
它會強迫存檔才能執行

本機就是你的電腦

MicroPython 設備
就是指你的控制板



停止執行程式

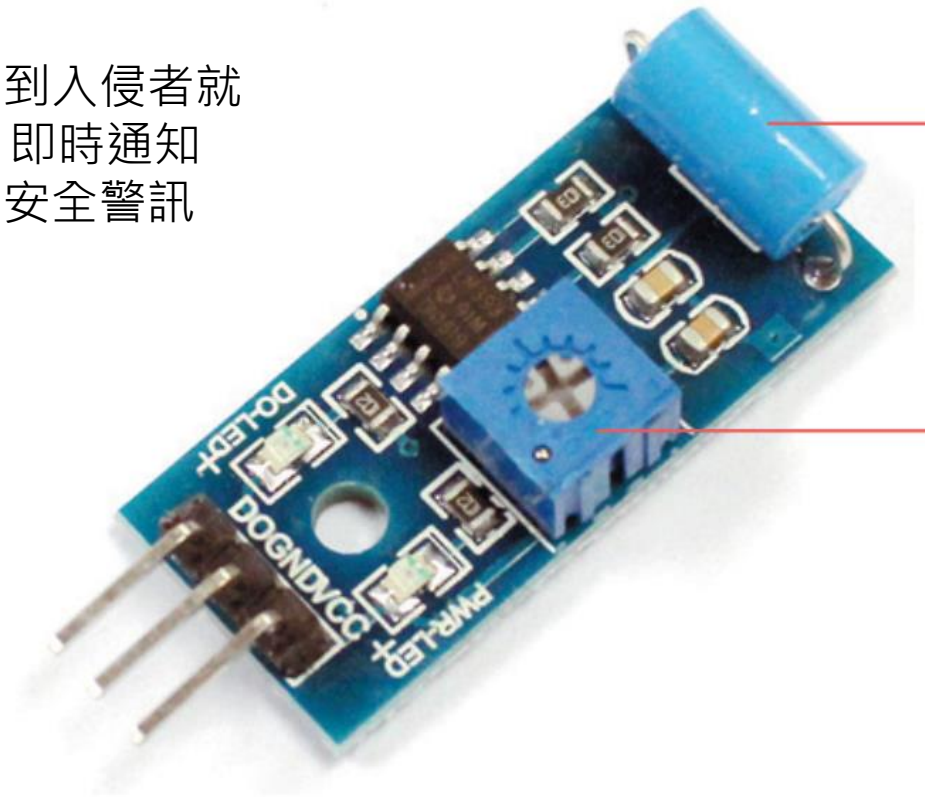

Thonny - D:\FM622A_0505\上課用\00_test.py @ 5 : 1
檔案 編輯 檢視 執行 Device 工具 說明
停止 / 重新啟動 後端程式 (Ctrl+F2)
檔案
本機
C: \ Users \ Admin
3D Objects
Contacts
00_test.py
1 from machine import
2 import time
3
4 led = Pin(2, Pin.OUT

03 防盜即時警報器

振動模組、手機簡訊、LINE即時通知

認識振動感測模組

一旦振動感測模組偵測到入侵者就
透過手機簡訊或 LINE 即時通知
出門在外也可隨時收到安全警訊

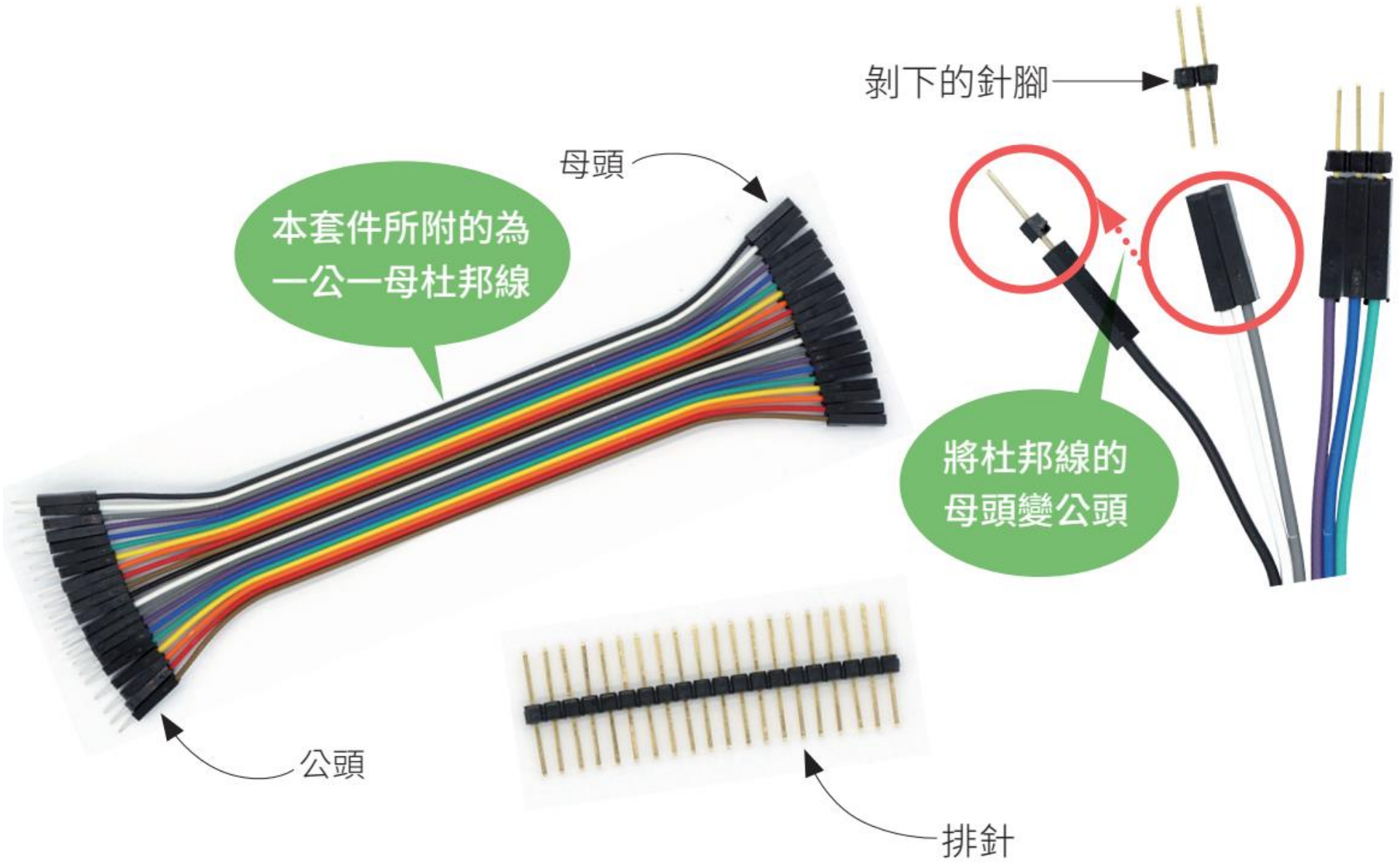


裡面有滾珠，模組會利用滾珠來偵測是否有震動

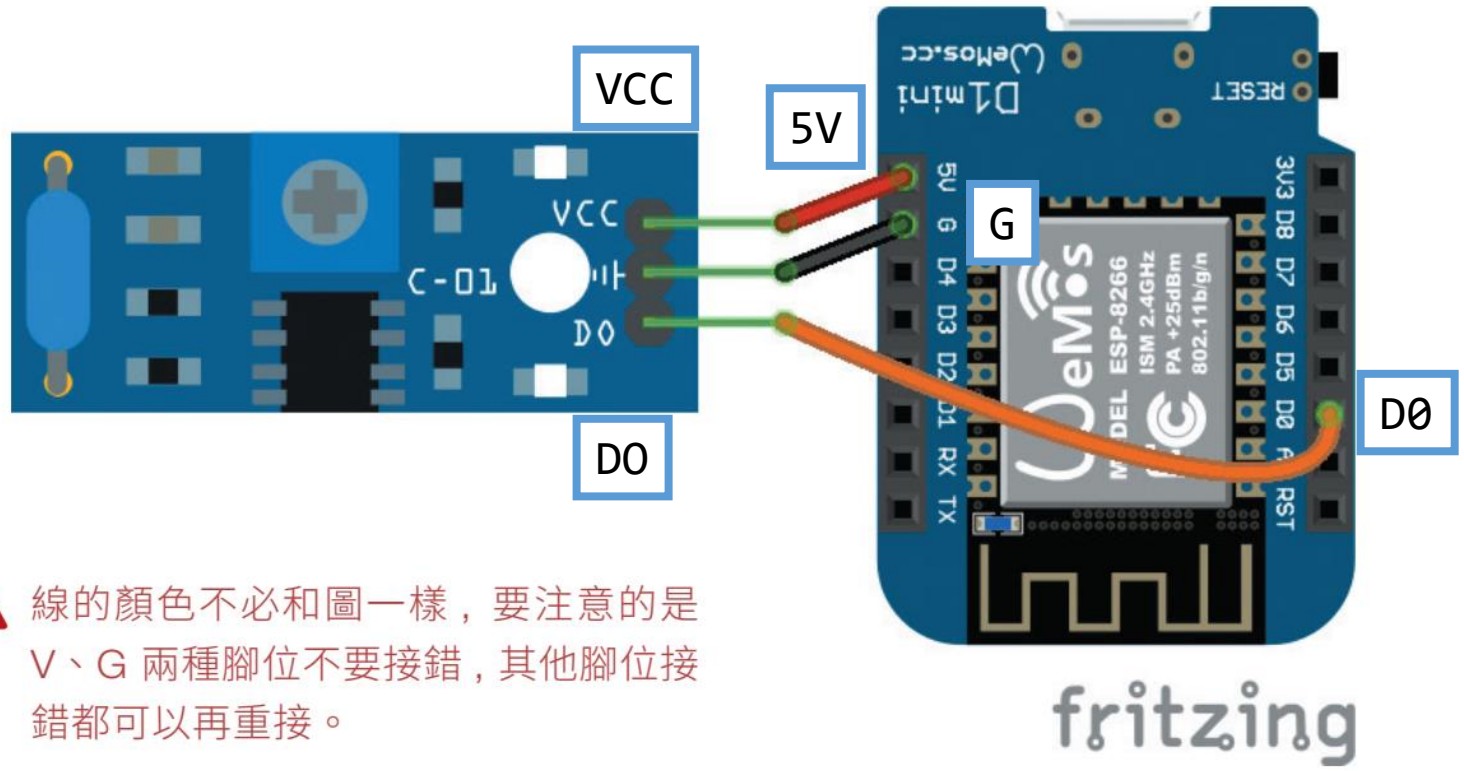
此旋鈕可以修改偵測靈敏度，順時針旋轉可增加靈敏度，反之則降低靈敏度

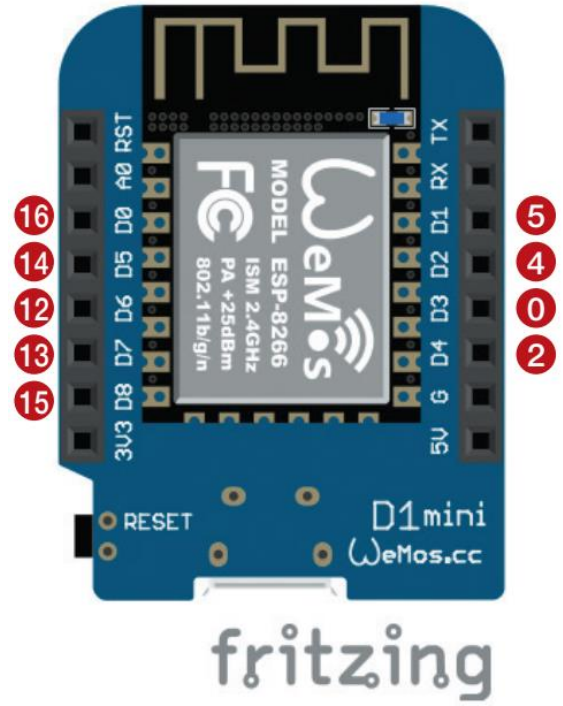
當模組感應到振動時，DO 腳位會輸出高電位。只要將這個振動感測模組放在保險箱、抽屜內，當有人翻找物品的話，模組就會感測振動輸出高電位，此時就立刻送出警報通知到我們手機。

杜邦線與排針



請依線路圖接線





⚠ 線的顏色不必和圖一樣，要注意的是 V、G 兩種腳位不要接錯，其他腳位接錯都可以再重接。

Lab03

讀取振動感測模組的輸入值

實驗目的	用程式讀取振動感測模組的輸入值, 藉以判斷是否正在振動。
材料	• D1 mini • 震動感測模組

設計原理

當我們建立腳位的 Pin 物件時，可用 "**Pin.IN**" 作為參數，設定這個腳位為輸入腳位：

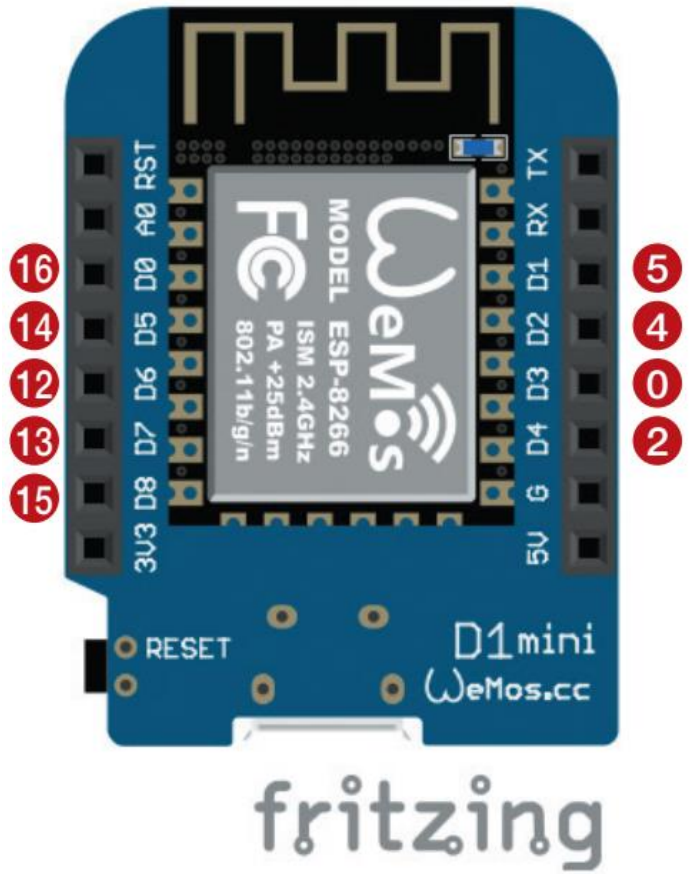
```
>>> from machine import Pin
>>> shock = Pin(16, Pin.IN)
```

上面我們建立了 16 號 (D0) 腳位的 Pin 物件，並且將其命名為 shock, 因為建立物件時使用了 "**Pin.IN**" 參數，所以 16 號腳位就會被設定為輸入腳位。

建立好輸入腳位的 Pin 物件後，便可以使用 value() 方法來讀取外部裝置輸出的電位高低：

```
>>> shock.value()
0                    ←—— 讀到 0 表示外部裝置輸出低電位
>>> shock.value()
1                    ←—— 讀到 1 表示外部裝置輸出高電位
```

振動感測模組偵測到振動會輸出高電位，所以若讀到 1 便代表有振動了。

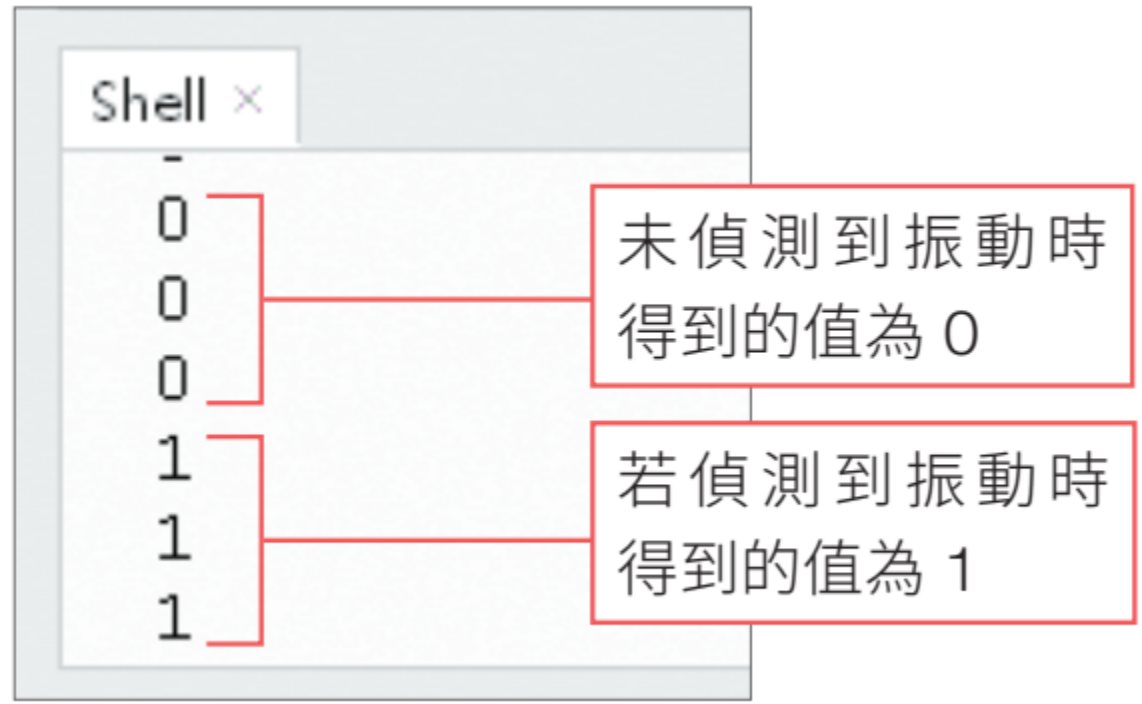


程式設計

```
01 from machine import Pin
02 import time
03
04 # 建立 16 號腳位的 Pin 物件, 設定為輸入腳位, 並命名為 shock
05 shock = Pin(16, Pin.IN)
06
07 while True:
08     # 用 value() 方法從 16 號腳位讀取按鈕輸出的高低電位
09     # 然後將讀到的值用 print() 輸出
10     print(shock.value())
11
12     # 暫停 0.05 秒
13     time.sleep(0.05)
```

實測

請按 F5 執行程式，然後用手搖動振動感測模組，在 Thonny 的 Shell 窗格觀察程式輸出的值：



04 發送手機簡訊

連接網路使用 Web API

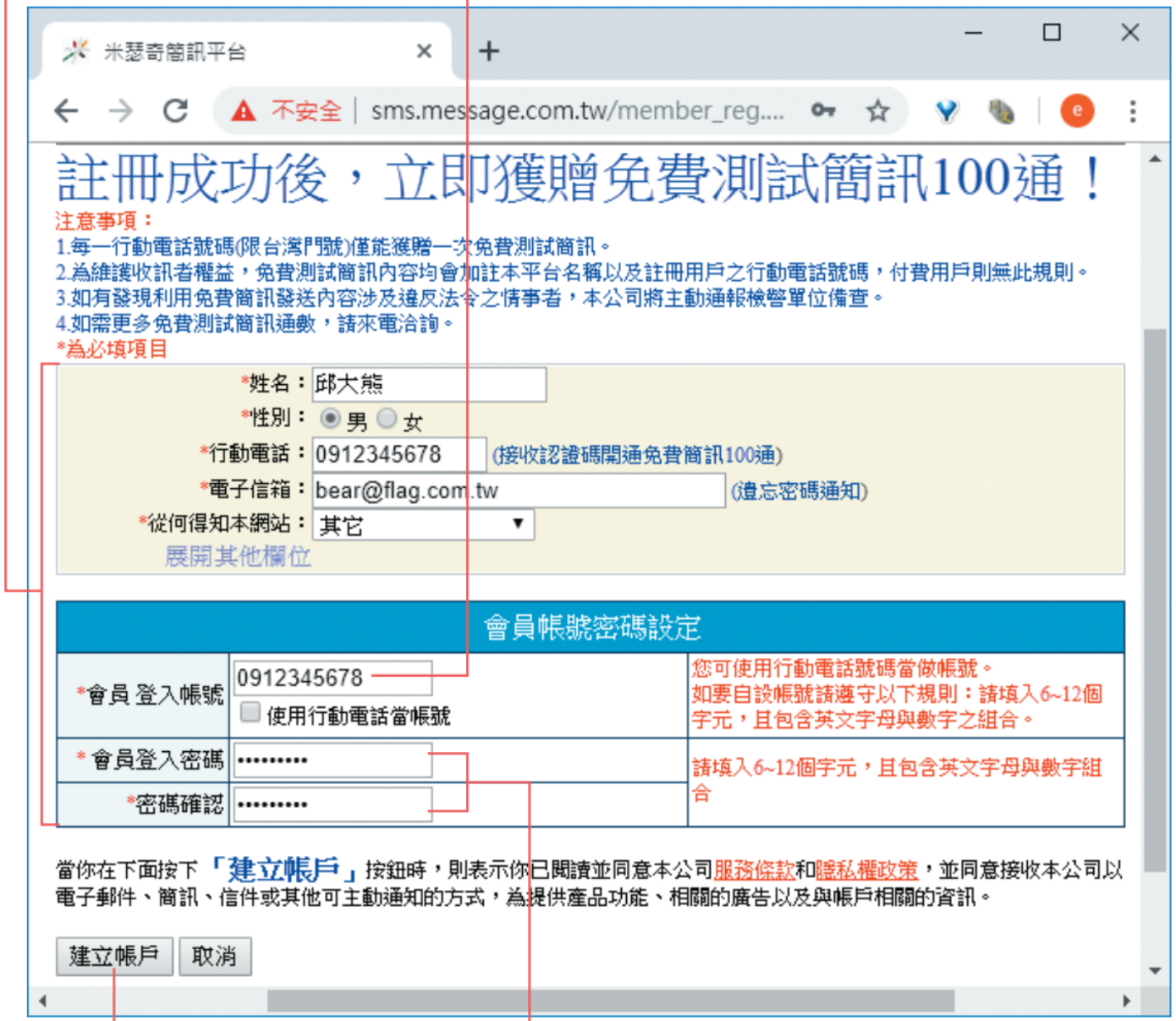
為了透過手機簡訊傳送感測器的資訊，我們將使用簡訊服務廠商的 API 來發送簡訊。

請連線 http://www.message.com.tw 如下操作加入會員：

1 按此連結

2 依照網頁說明輸入資料

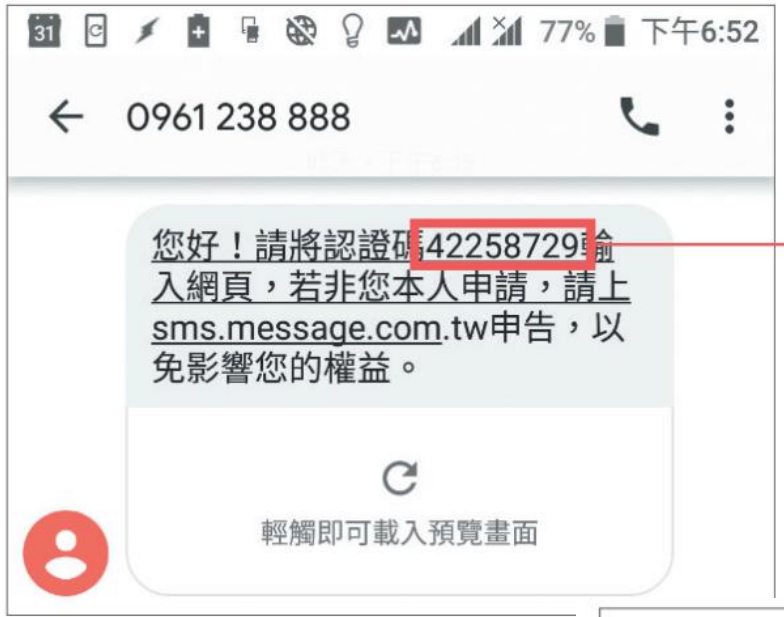
建議直接使用手機門號作為會員帳號



3 按此鈕建立帳戶

請注意密碼必須包含英文與數字

稍待片刻後手機簡訊會收到認證碼：



1 打開手機上的簡訊
取得驗證碼

2 回到電腦瀏覽器上
輸入驗證碼

3 按此鈕即可收到 100 通測試用的簡訊，若超過時需要付費購買額度

用 HTTP 連線即可發送簡訊

⚠ 不同廠商 API 連線的 HTTP 網址格式皆不相同，若您使用其他廠商的話，請自行參閱該廠商網站提供的說明文件。

Lab04

防盜即時簡訊通知

實驗目的	利用振動感測模組感應是否有振動, 若有則發送簡訊到手機。
材料	• D1 mini • 振動感測模組

線路圖

同 Lab03

WiFi 連線

要使用網路，首先必須匯入 **network 模組**，利用其中的 **WLAN 類別**建立控制無線網路的物件：

```
>>> import network
>>> sta_if = network.WLAN(network.STA_IF)
```

網路介面	說明
network.STA_IF	工作站 (station) 介面，專供連上現有的 Wi-Fi 無線網路基地台，以便連上網際網路
network.AP_IF	熱點 (access point) 介面，可以讓 D1 mini 變成無線基地台，建立區域網路

啟用並連接 Wi-Fi 無線網路

由於我們需要讓 D1 mini 連上網際網路擷取資訊，所以必須使用**工作站介面**。取得無線網路物件後，要先啟用網路介面：

```
>>> sta_if.active(True)
```

參數 True 表示要啟用網路介面；如果傳入 False 則會停用此介面。接著，就可以嘗試連上無線網路：

```
>>> sta_if.connect('無線網路名稱', '無線網路密碼')
```

確認是不是有連上網路：

```
# 檢查連線狀態，還沒連上就繼續跑迴圈
while not wifi.isconnected():
    pass
```

什麼都不做的動作
單純為了合乎語法

取代瀏覽器連接網址的 urequests 模組

在 Python 中有個 requests 模組可以讓我們的程式扮演瀏覽器的角色，連線網站使用各式各樣的網路服務，不過因為 D1 mini 控制板的記憶體比較少，所以在 MicroPython 中提供的是精簡版的 urequests 模組，名稱開頭的 "u" 是 "micro" 的意思。只要匯入此模組，即可使用該模組提供的 get() 連線網路服務：

```
>>> import urequests ◄── 匯入 urequests 模組
>>> urequests.get("https://flagtech.github.io/flag.txt")
                                                  連線網址
```

程式設計


```
01 from machine import Pin
02 import time, network, urequests
03
04 # 連線 Wifi 網路
05 sta_if = network.WLAN(network.STA_IF)
06 sta_if.active(True)
07 sta_if.connect("Wifi基地台", "Wifi密碼")
08 while not sta_if.isconnected():
09     pass
10 print("Wifi已連上")
11
12 username = "簡訊服務帳號"
13 passwd = "簡訊服務密碼"
14 phone = "接收簡訊的手機號碼"
15 message = "有人打開保險箱在翻找東西,趕快去抓小偷!" #請勿輸入空格
16
```

```
17 # 建立 16 號腳位的 Pin 物件, 設定為輸入腳位, 並命名為 shock
18 shock = Pin(16, Pin.IN)
19
20 while True:
21     if shock.value() == 1:
22         print("感應到振動!")
23
24         # 連線簡訊服務發送簡訊通知
25         urequests.get(
26             "http://api.message.net.tw/send.php?mtype=G&id="
27             + username + "&encoding=utf8&password=" + passwd
28             + "&tel=" + phone + "&msg=" + message)
29
30         # 暫停 60 秒, 避免短時間內一直收到重複的警報
31         time.sleep(60)
```

- 請依照您的環境修改程式第 7 行中的『Wifi 基地台』、『Wifi 密碼』等設定
- 請依照 2-2 節申請簡訊服務帳戶時輸入的資料，修改程式第 12、13 行的『簡訊服務帳號』、『簡訊服務密碼』等設定
- 請修改程式第 14 行的『接收簡訊的手機號碼』，輸入您的手機號碼
- 程式第 15 行的簡訊內容可以任意修改，但請注意不要輸入空格

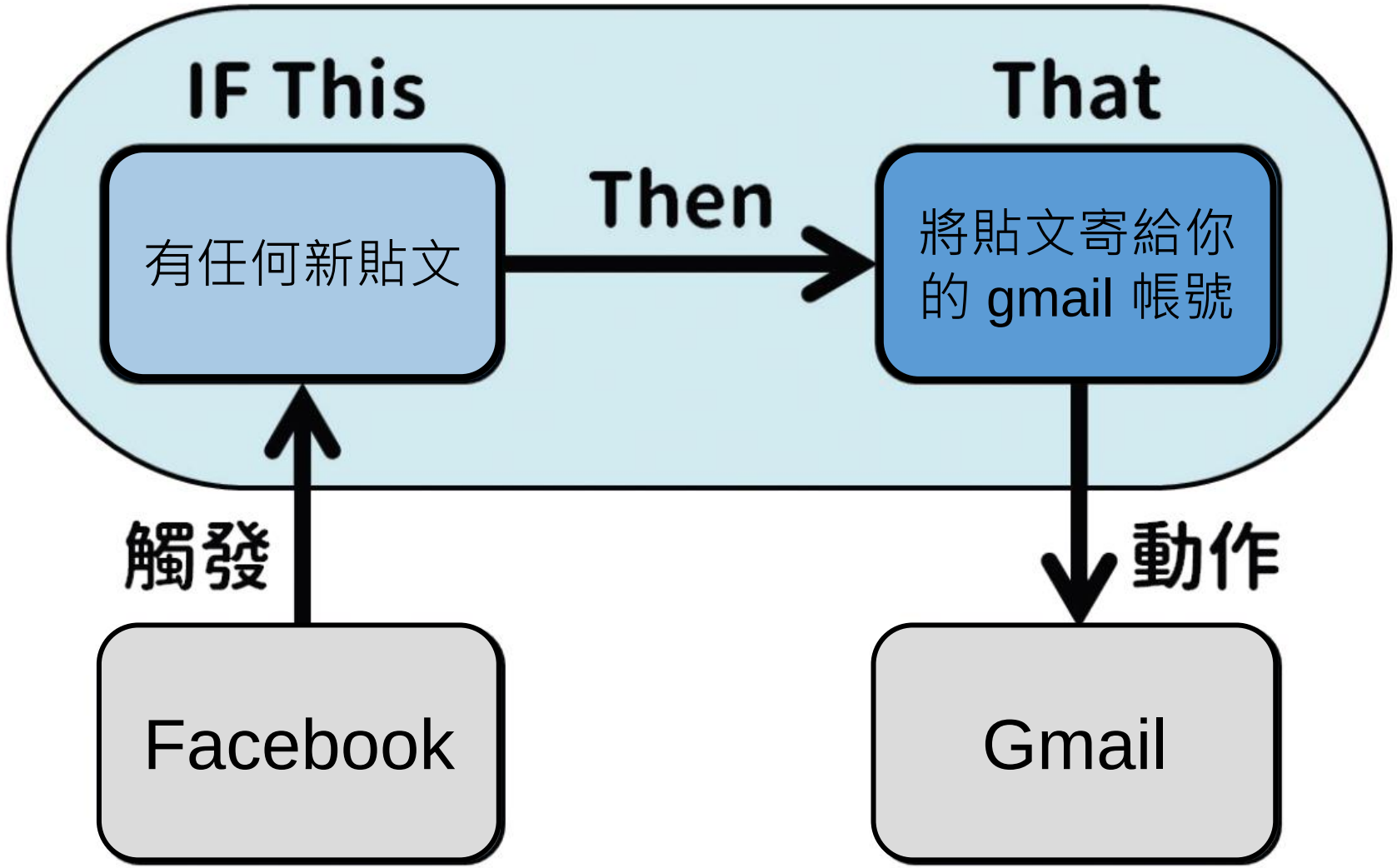
實測

請按 F5 執行程式，然後用手搖動振動感測模組，可以在 Thonny 的 Shell 窗格看到程式輸出 " 感應到振動 !"，接著稍等片刻您的手機應該就會收到簡訊通知。

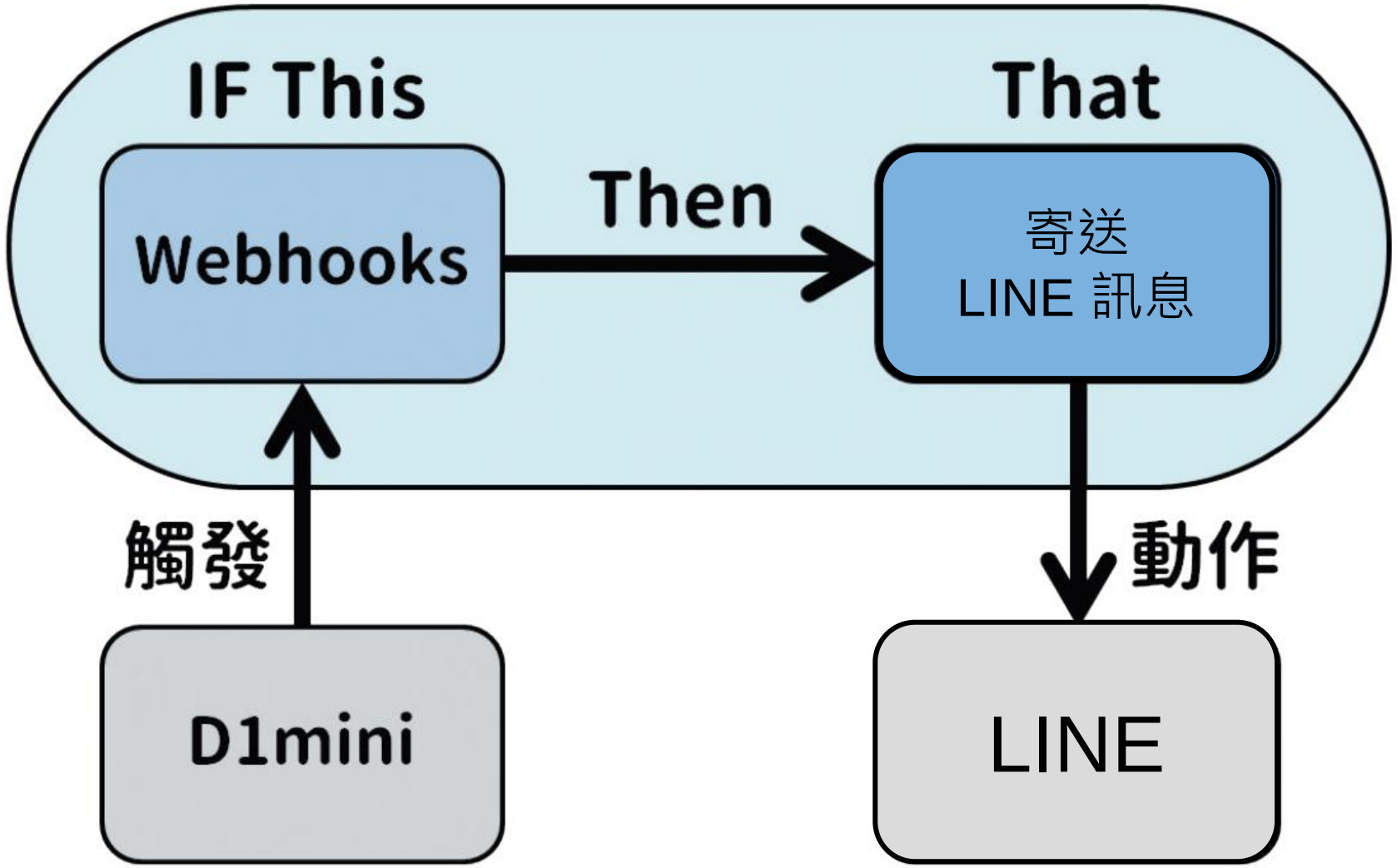
05 使用 IFTTT 發送 LINE 通知

串聯網路服務

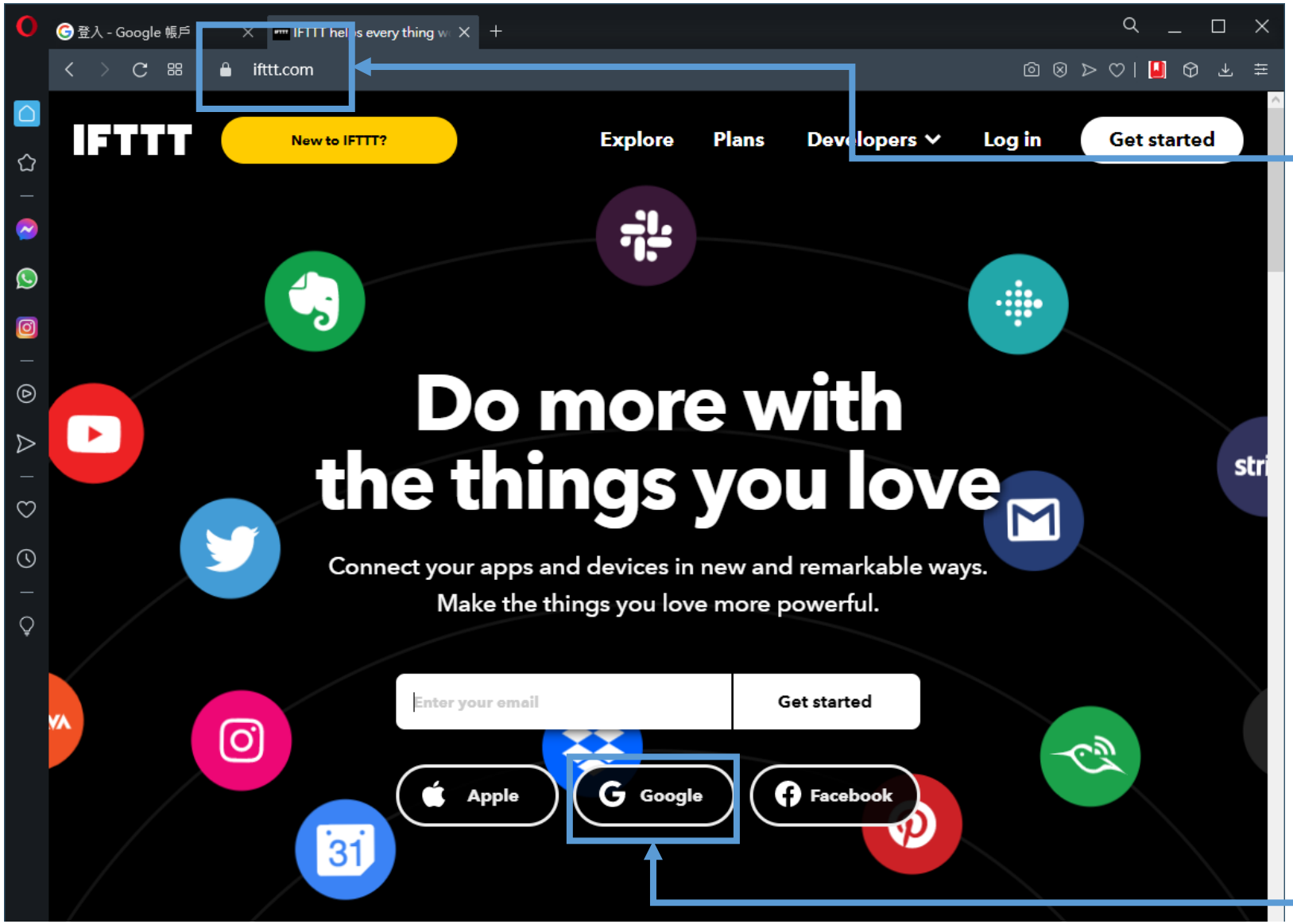
IFTTT 服務簡介


IF This
That
有任何新貼文
Then
將貼文寄給你的 gmail 帳號
觸發
動作
Facebook
Gmail

改由 D1 mini 從程式中觸發事件



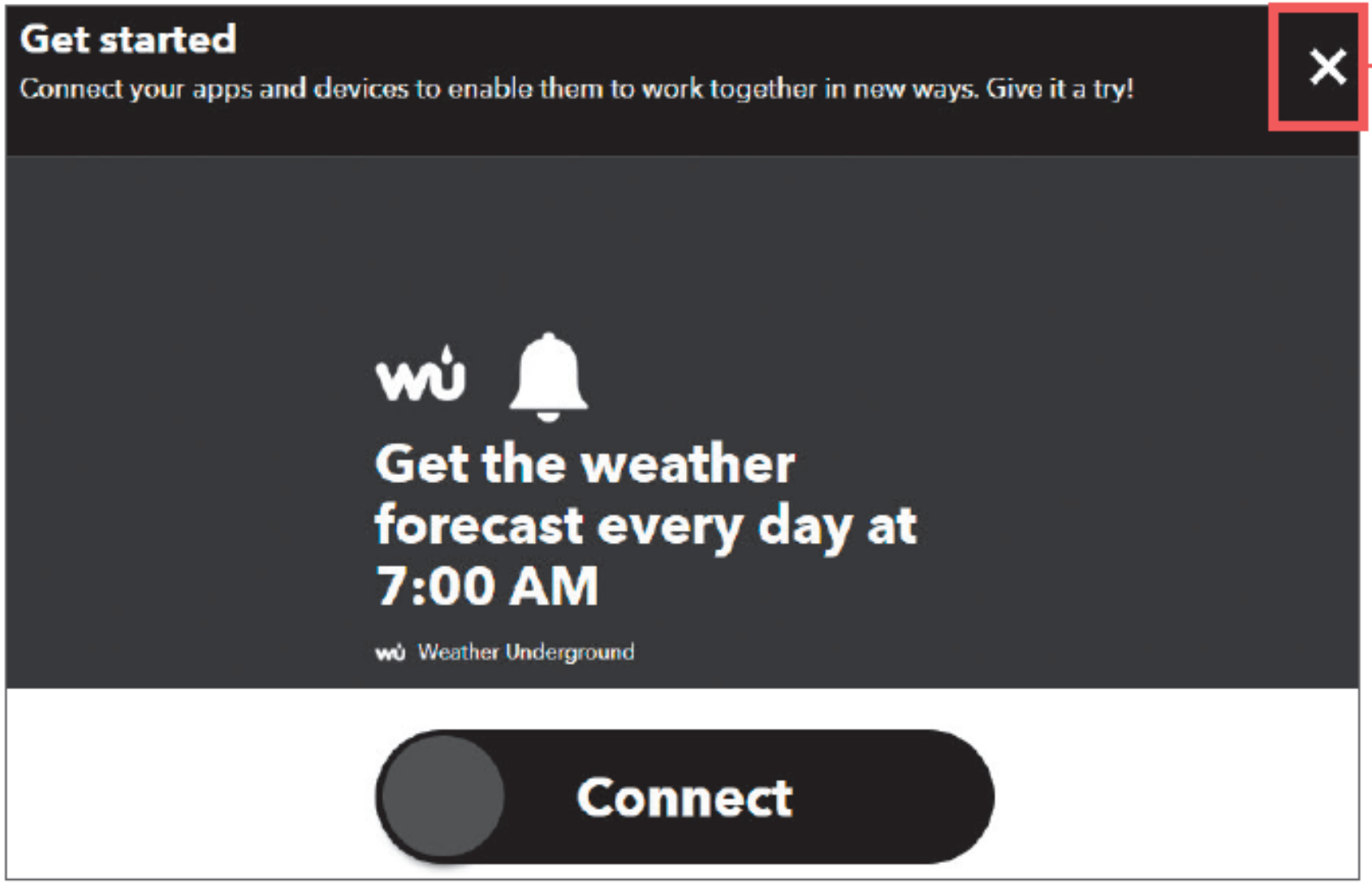
註冊 IFTTT 帳號 (ifttt.com)



請在網址列輸入
ifttt.com

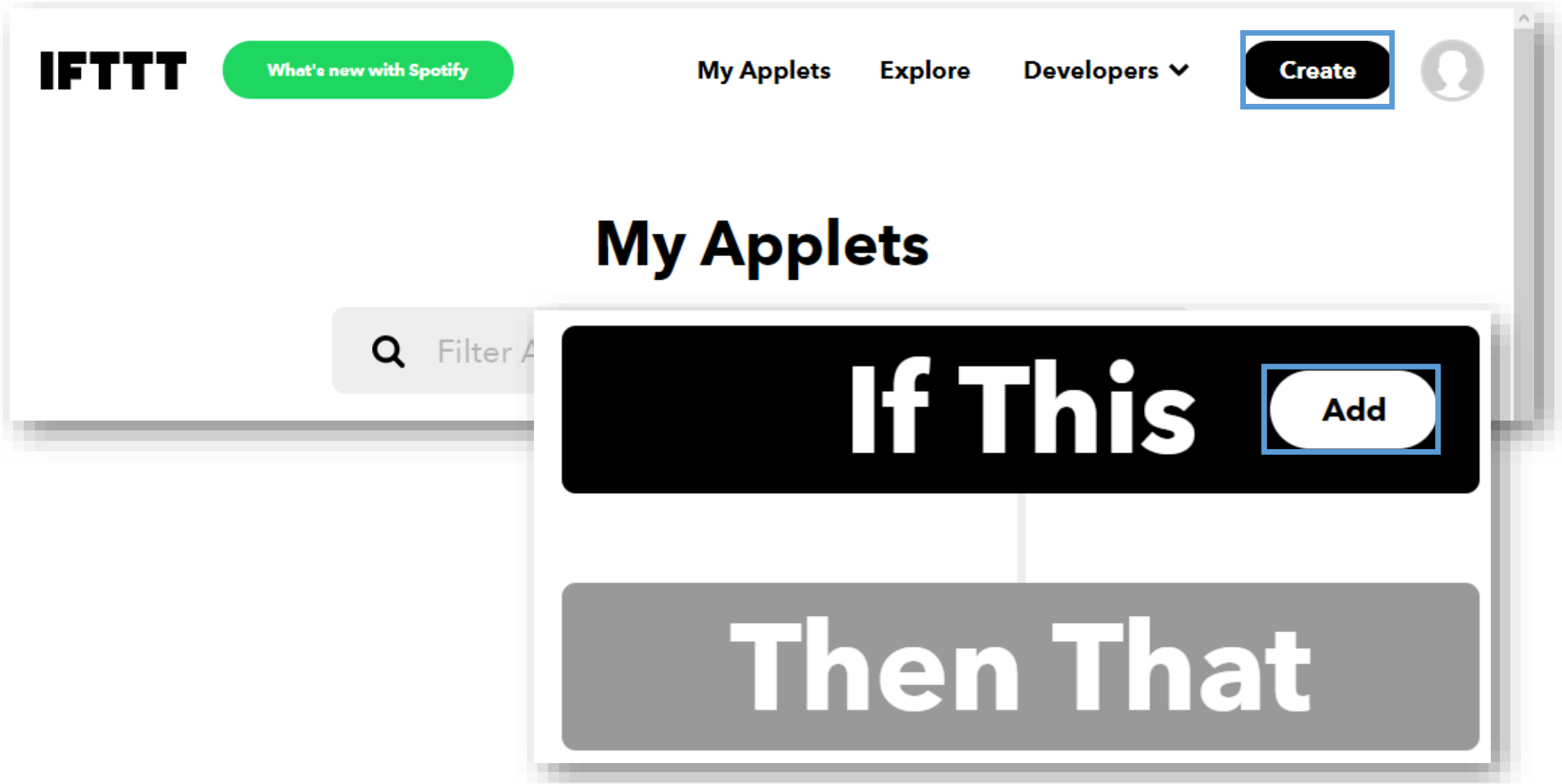
等下會需要用到 Google 服務
直接用 Google 帳號比較方便

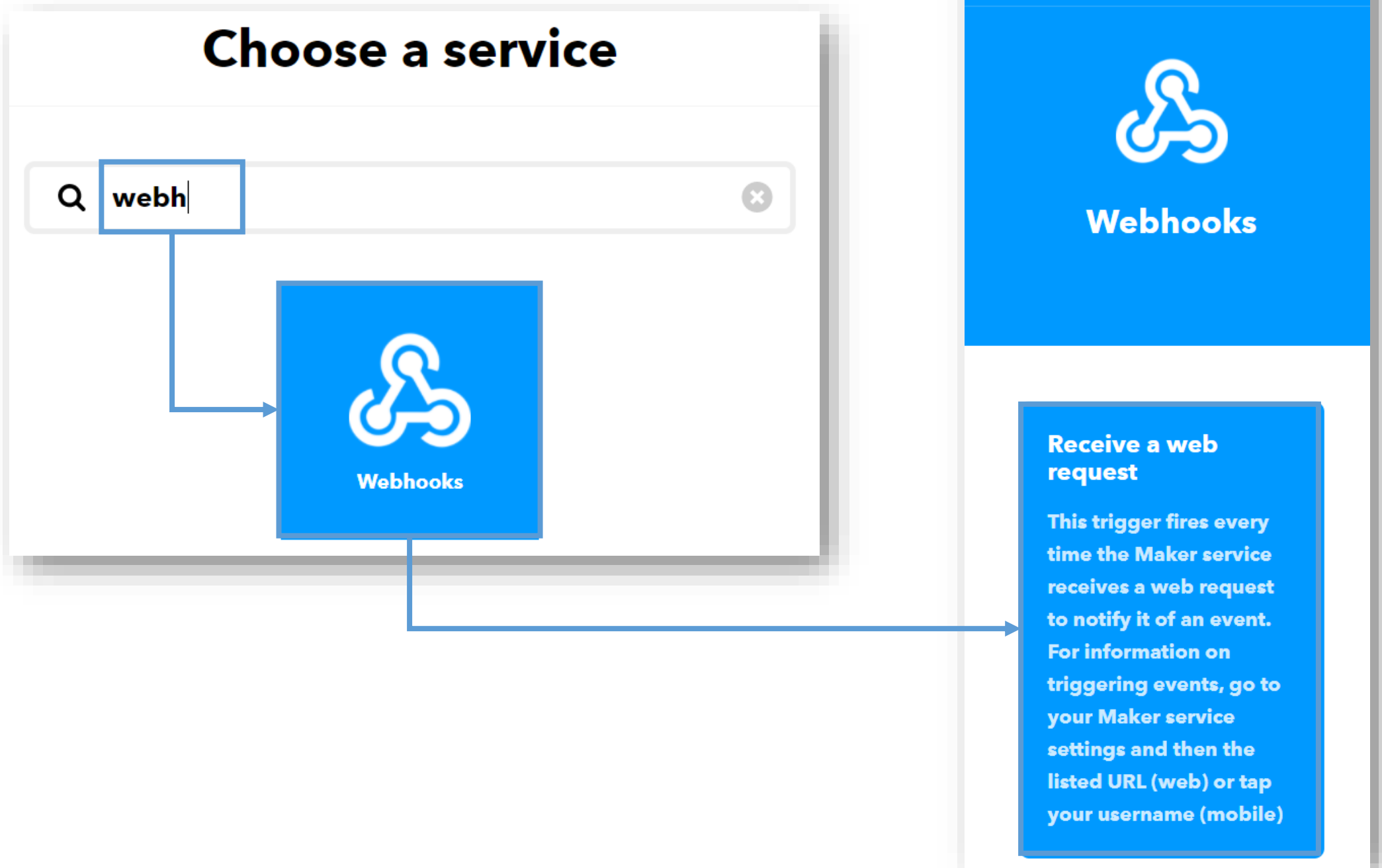
註冊 IFTTT 帳號



6 申請完成後，若出現推薦服務畫面，點右上角叉叉關閉。

建立新的 Applet


IFTTT
What's new with Spotify
My Applets
Explore
Developers
Create
My Applets
Filter A
If This
Add
Then That


Choose a service
webh
Webhooks
Choose a trigger
Webhooks
Receive a web request
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Back

Complete trigger fields

Receive a web request

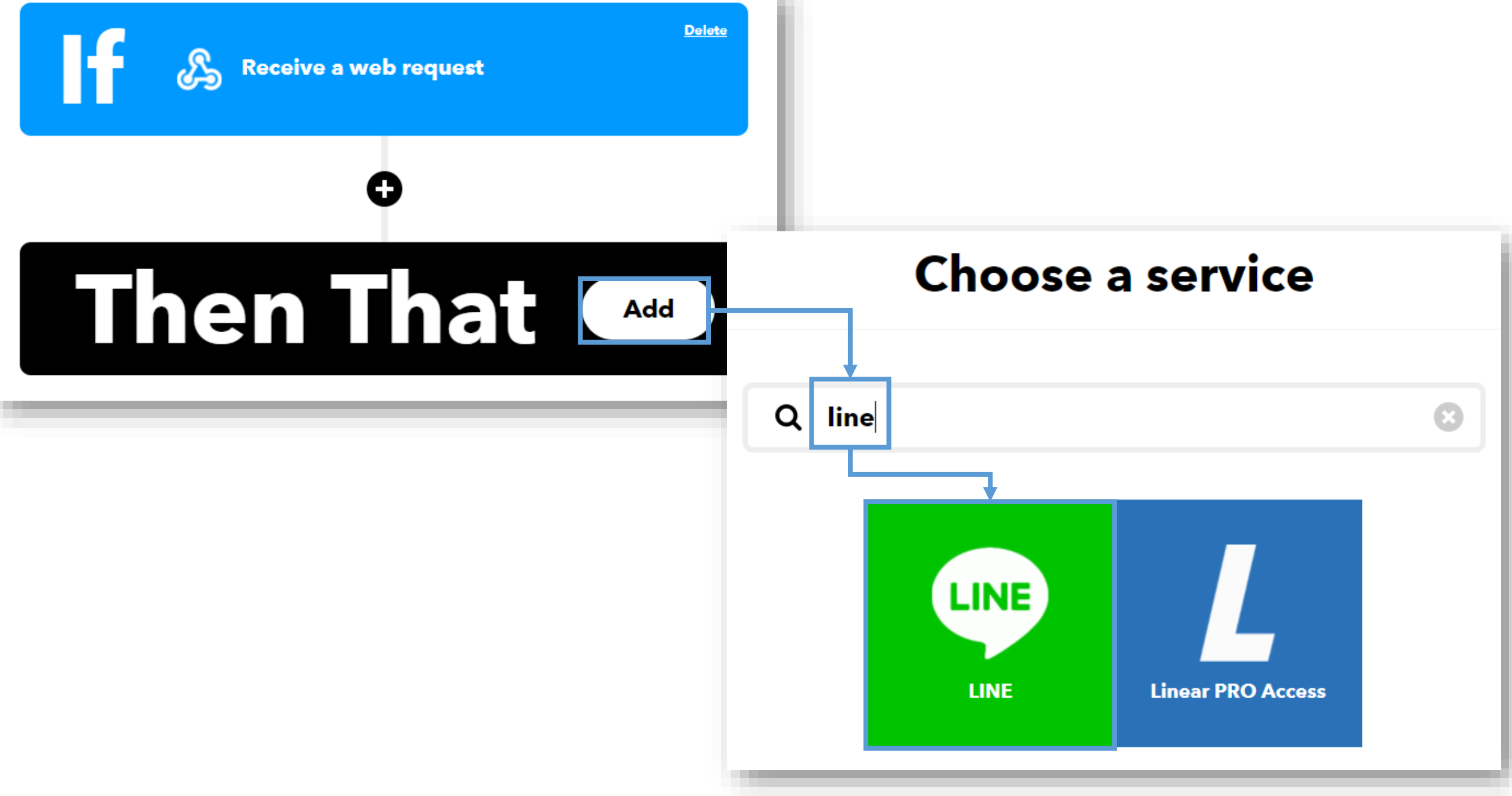
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)

Event Name

theft

The name of the event, like "button_pressed" or "front_door_opened"

Create trigger


If
Receive a web request
Delete
Then That
Add
Choose a service
line
LINE
LINE
Linear PRO Access

Choose an action

LINE

LINE

Send message

This Action will post a message to LINE.

Connect service

LINE

LINE

LINE is a global messaging app used in over 230 countries and regions. LINE offers fun and free voice, video, and chat communication across multiple platforms. Receive event notifications from LINE Notify official account.

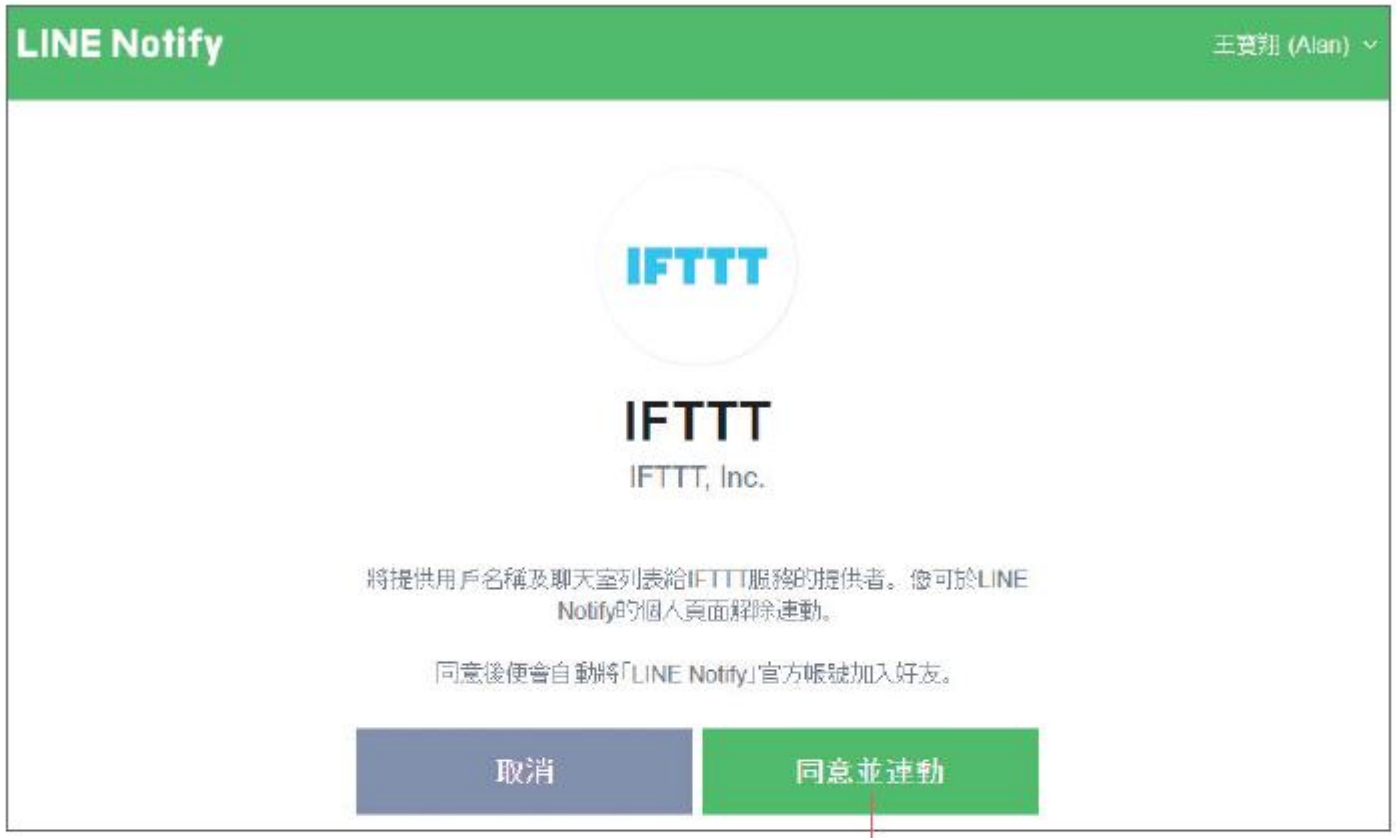
Connect

若看到此畫面, 請至
手機上用 LINE 確認

5 輸入 Line 帳號與密碼

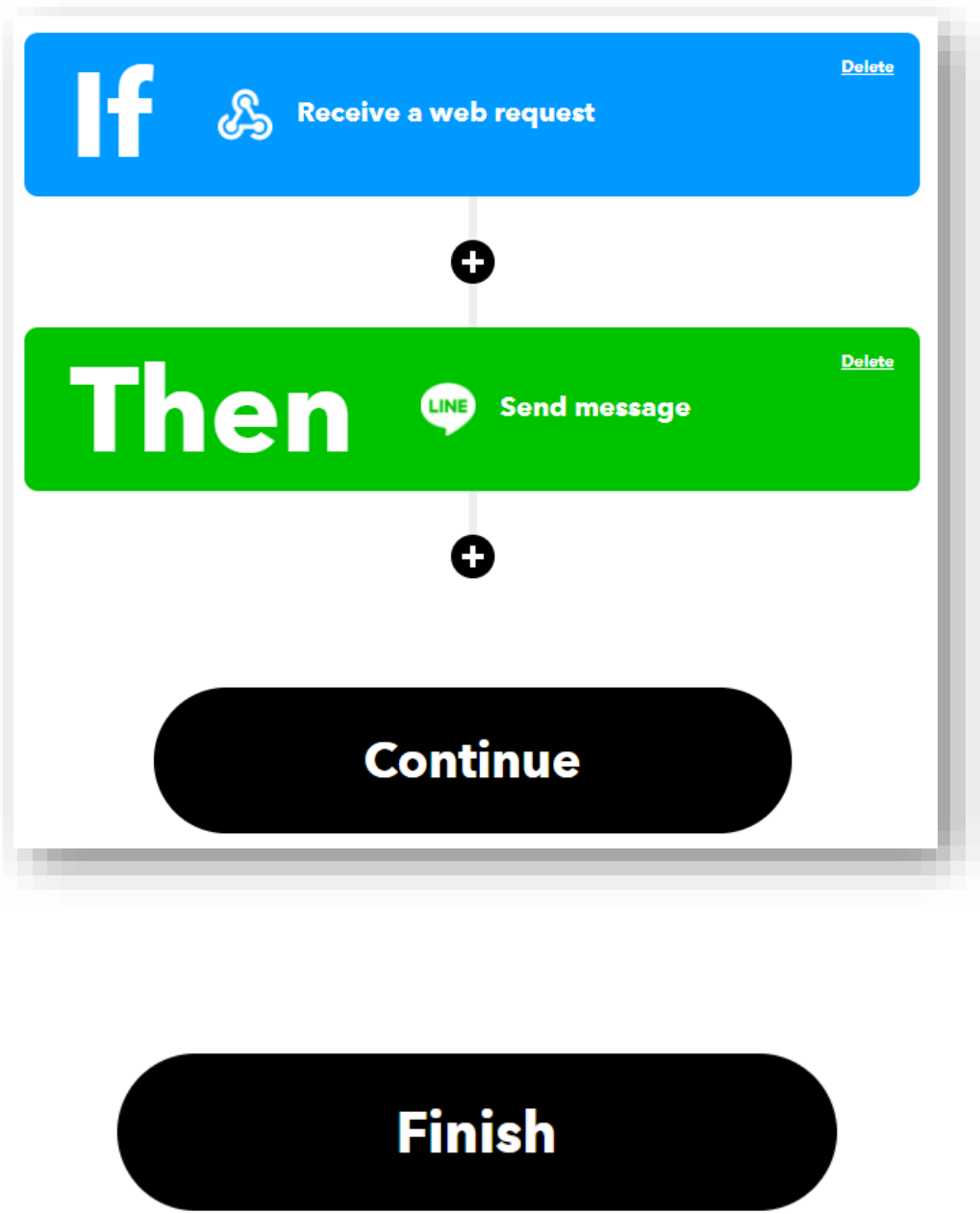
6 登入 Line

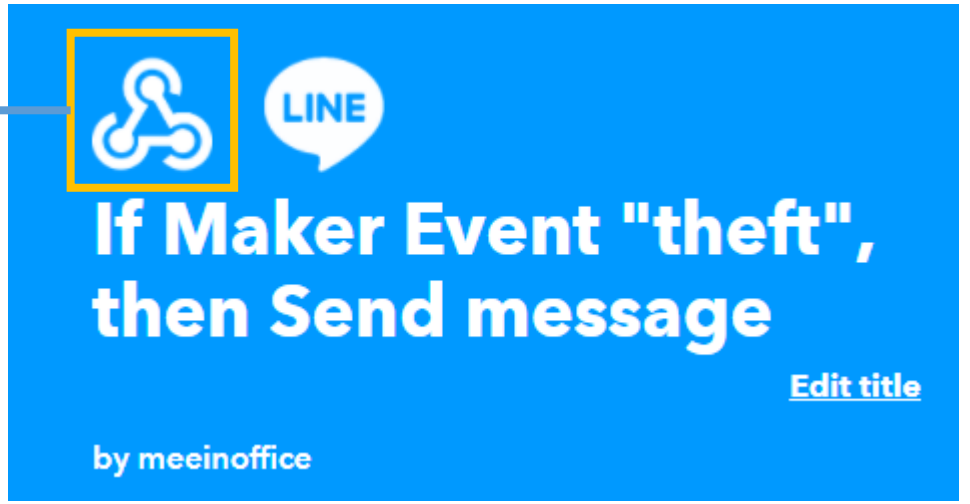
若忘記 LINE 密碼可至 LINE APP
『設定/我的帳號』重新設定

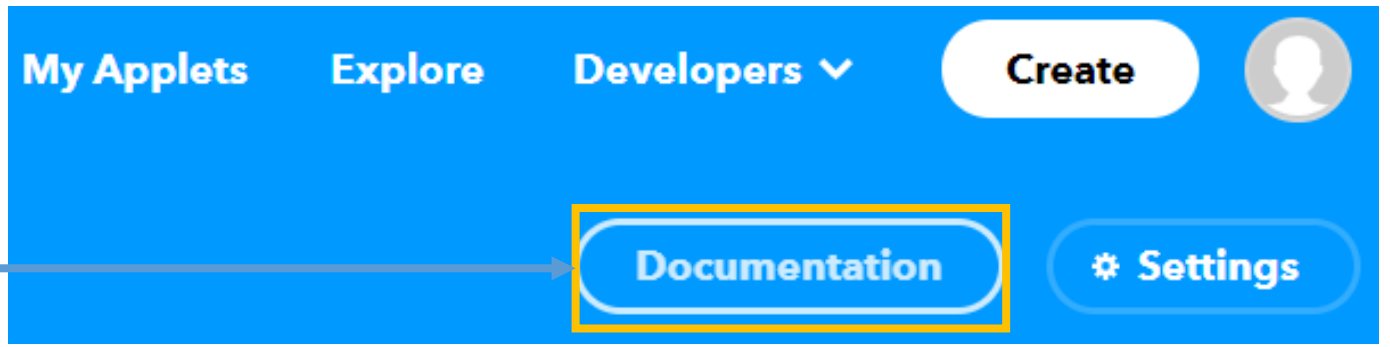


7 點同意並連動
來授權 IFTTT 存取你的 Line 帳號

8 你的 Line 會收到連動通知


If
Receive a web request
Delete
Then
Send message
Delete
Continue
Finish


LINE
If Maker Event "theft", then Send message
Edit title
by meeinoffice


My Applets
Explore
Developers
Create
Documentation
Settings

測試 LINE 通知功能

Documentation 頁面中,可以看到我們的 **key** 與 **HTTP 請求網址**


curl -X POST https://maker.ifttt.com/trigger/theft/with/key/bOXBrL9i9aT6c98ET13E0y
Test It
2 按此鈕測試
完整的 HTTP 請求網址，請複製下來
3 打開手機的 LINE
選擇 LINE Notify
89% 下午4:52
LINE Notify
今天
【IFTTT】 有人打開保險箱在翻找東西，趕快去抓小偷！
下午4:00
4 收到 IFTTT 傳來的通知

Lab05

防盜即時 LINE 通知

實驗目的	利用振動感測模組感應是否有振動, 若有則發送通知到手機的 LINE App。
材料	• D1 mini • 振動感測模組

程式設計

Lab05

```
01 from machine import Pin
02 import time, network, urequests
03
04 # 連線 Wifi 網路
05 sta_if = network.WLAN(network.STA_IF)
06 sta_if.active(True)
07 sta_if.connect("Wifi基地台", "Wifi密碼")
08 while not sta_if.isconnected():
09     pass
10 print("Wifi已連上")
11
12 # 建立 16 號腳位的 Pin 物件, 設定為輸入腳位, 並命名為 shock
13 shock = Pin(16, Pin.IN)
```

```
14
15 while True:
16     if shock.value() == 1:
17         print("感應到振動!")
18
19         # 連線 IFTTT 服務發送簡訊通知
20         urequests.get("IFTTT的HTTP請求網址")
21
22         # 暫停 60 秒, 避免短時間內一直收到重複的警報
23         time.sleep(60)
```

⚠ 請依照您的環境修改程式中的『Wifi 基地台』、『Wifi 密碼』等設定，另外『IFTTT 的 HTTP 請求網址』請輸入 2-4 節最後取得的 HTTP 請求路徑，請務必記得將開頭的 "https" 更改為 "http"。

06 PM2.5 空污警報燈

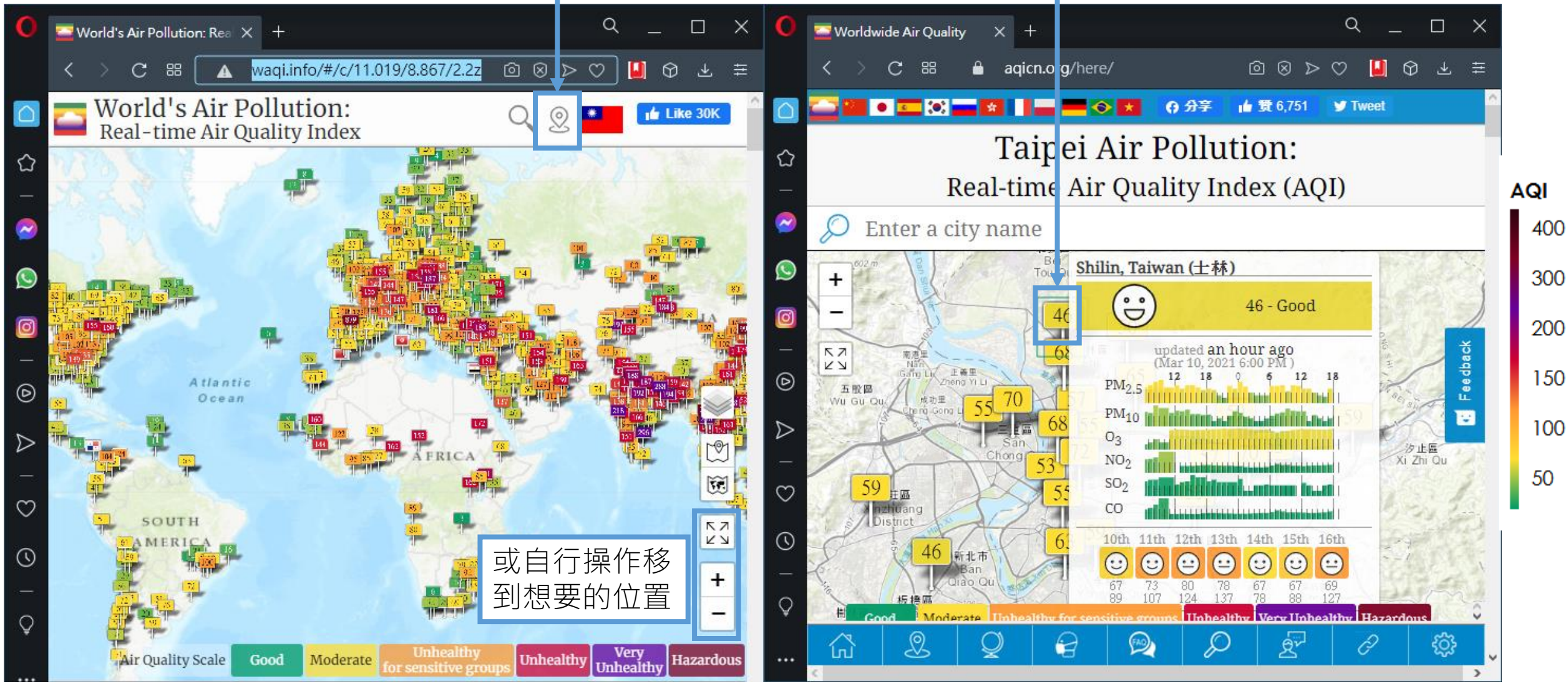
Open Data 網路爬蟲

AQI 空氣品質指標

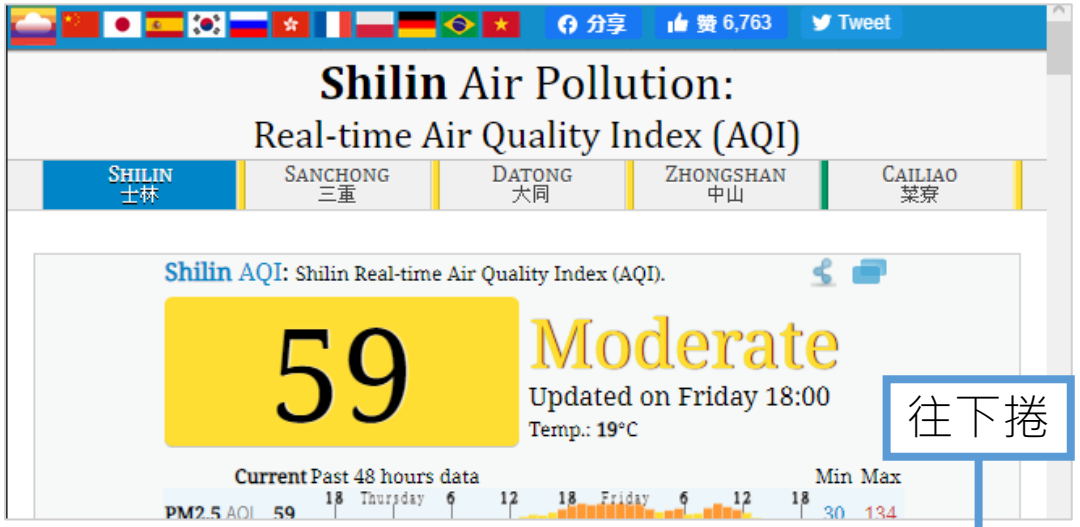
waqi.info

會快速移到目前位置

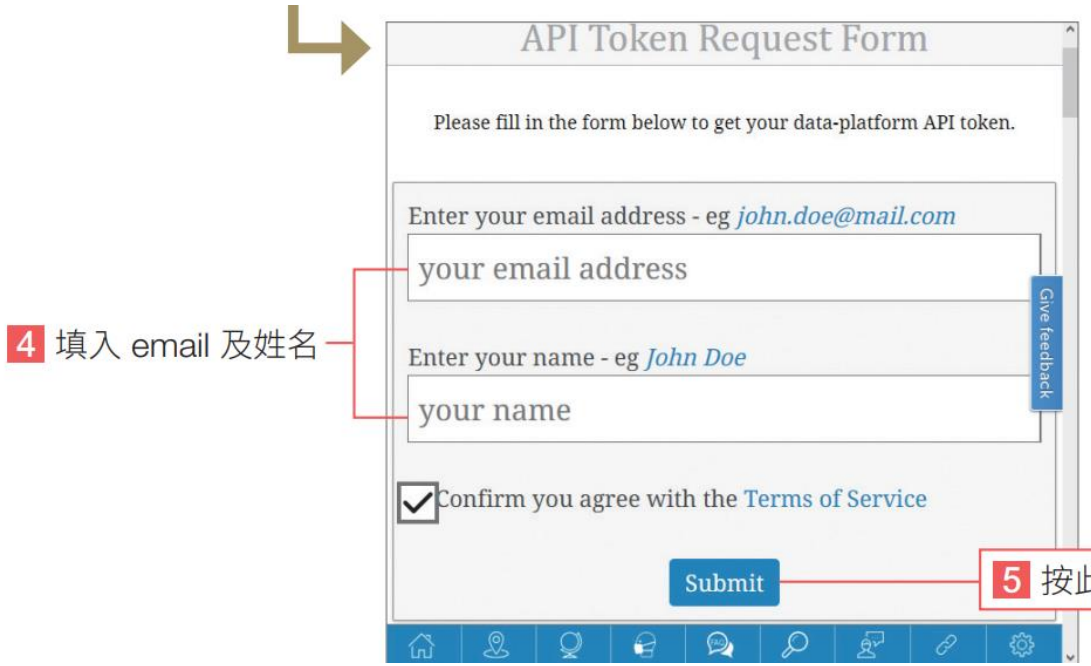
顯示詳細頁面



申請網路 API 使用金鑰

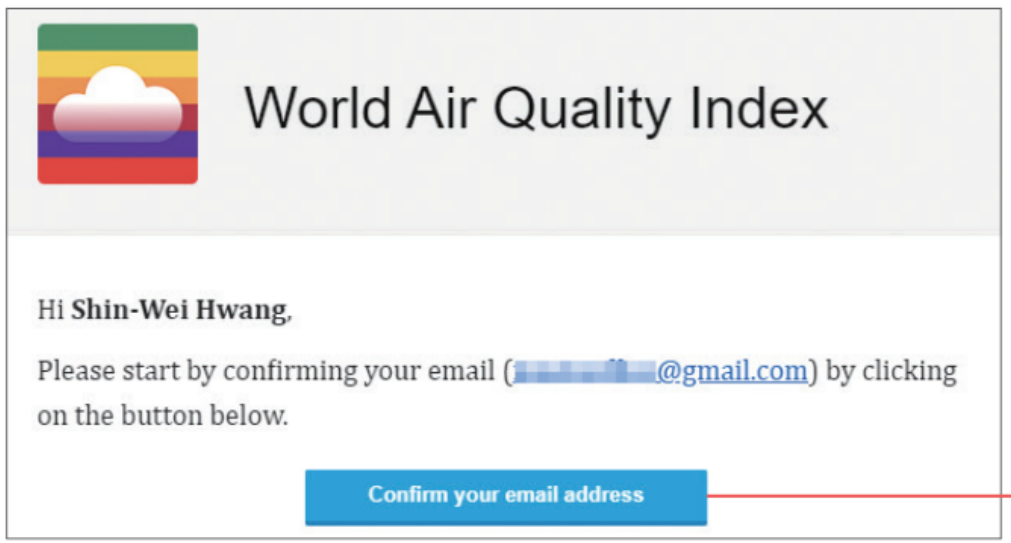


往下捲

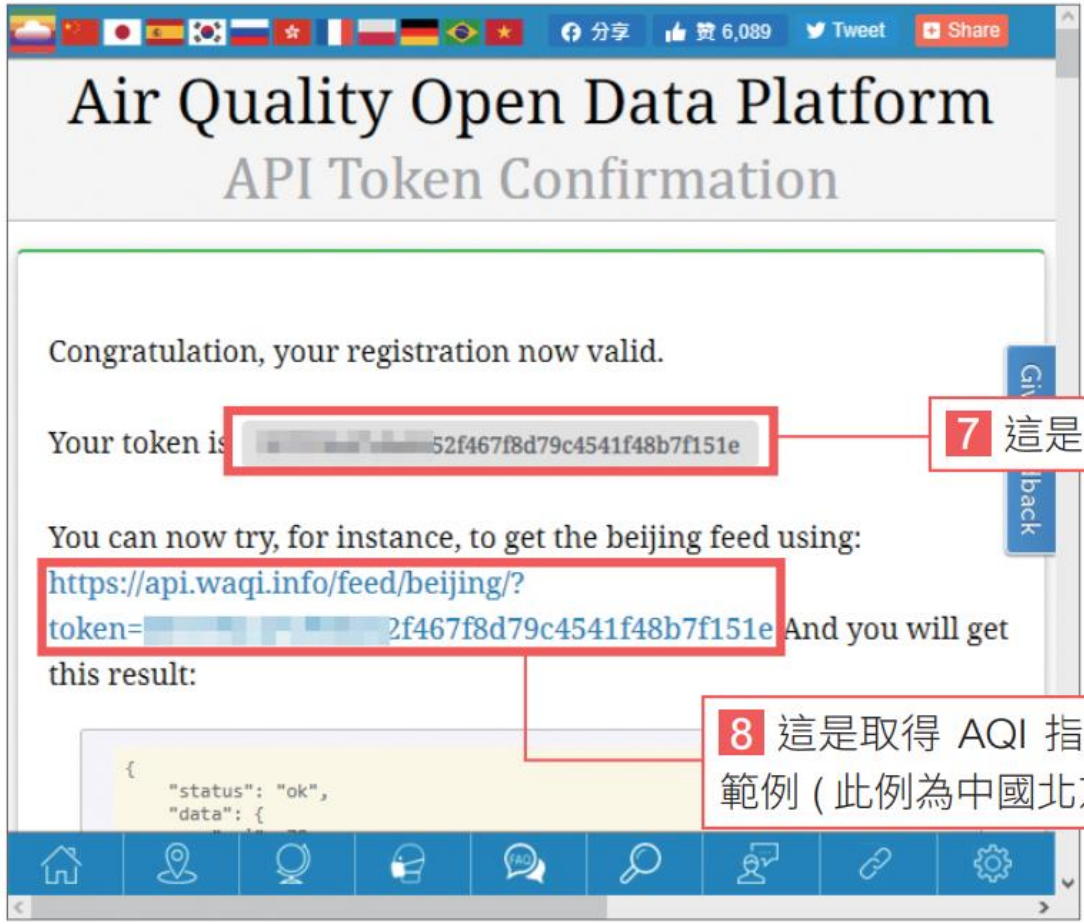


4 填入 email 及姓名

5 按此鈕送出



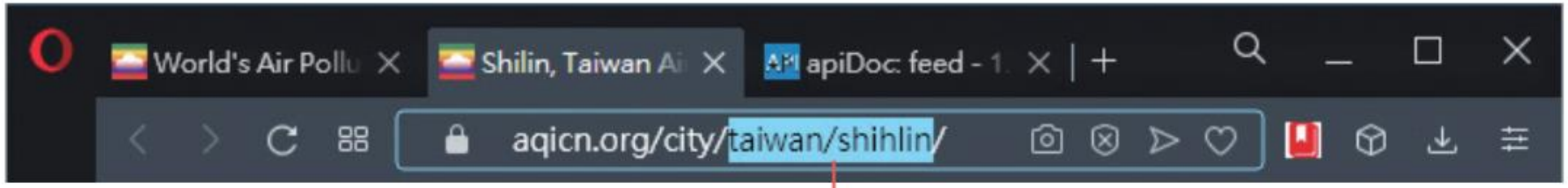
6 按一下確認後會轉至 WAQI 網站



7 這是你的權杖

8 這是取得 AQI 指數的網址範例（此例為中國北京站）

找到觀測站的名稱



先進到觀測站的詳細頁面

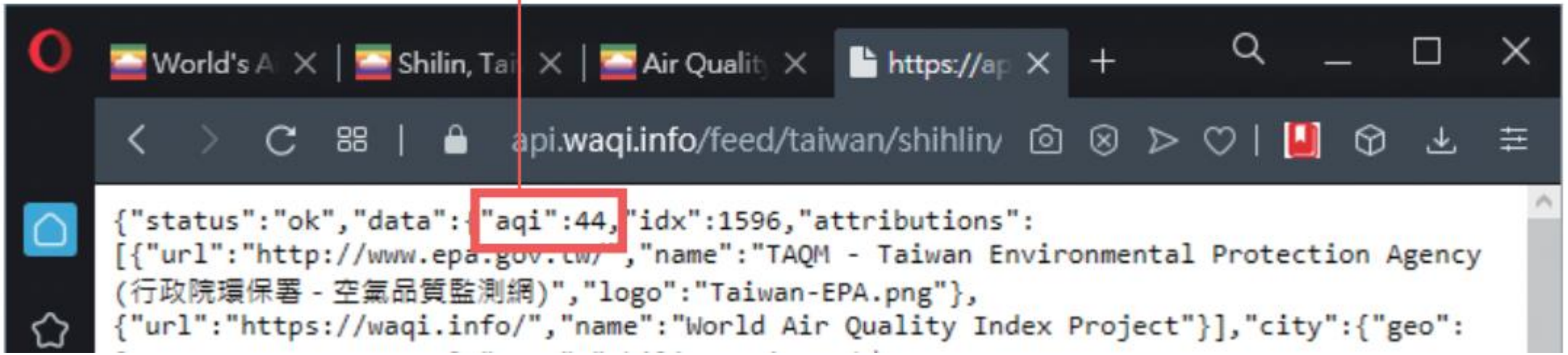
網址中 city/ 之後就是觀測站的名稱，此例為士林站

`https://api.waqi.info/feed/`**`taiwan/shihlin`**`/?token=你的權杖`

AQI
400
300
200
150
100
50

這裡可以看到 AQI 指數

以後萬一忘記權杖
可再回郵件按確認



透過程式取得 AQI 的第一步：WiFi 連線

要使用網路，首先必須匯入 **network 模組**，利用其中的 **WLAN 類別**建立控制無線網路的物件：

```
>>> import network
>>> sta_if = network.WLAN(network.STA_IF)
```

網路介面	說明
network.STA_IF	工作站 (station) 介面，專供連上現有的 Wi-Fi 無線網路基地台，以便連上網際網路
network.AP_IF	熱點 (access point) 介面，可以讓 D1 mini 變成無線基地台，建立區域網路

啟用並連接 Wi-Fi 無線網路

由於我們需要讓 D1 mini 連上網際網路擷取資訊，所以必須使用**工作站介面**。取得無線網路物件後，要先啟用網路介面：

```
>>> sta_if.active(True)
```

參數 True 表示要啟用網路介面；如果傳入 False 則會停用此介面。接著，就可以嘗試連上無線網路：

```
>>> sta_if.connect('無線網路名稱', '無線網路密碼')
```

確認是不是有連上網路：

```
# 檢查連線狀態，還沒連上就繼續跑迴圈
while not wifi.isconnected():
    pass
```

什麼都不做的動作
單純為了合乎語法

扮演瀏覽器腳色的 urequests 模組

在 Python 中有個 requests 模組可以讓我們的程式扮演瀏覽器的角色，連線網站使用各式各樣的網路服務，不過因為 D1 mini 控制板的記憶體比較少，所以在 MicroPython 中提供的是精簡版的 urequests 模組，名稱開頭的 "u" 是 "micro" 的意思。只要匯入此模組，即可使用該模組提供的 get() 連線網路服務：

```
>>> import urequests ◄── 匯入 urequests 模組
>>> urequests.get("https://flagtech.github.io/flag.txt")
                                                  連線網址
```

用程式取得網路資料

用瀏覽器取得資料

get() 在連線網站時, 也會同時取回網站傳回的資料

```
>>> res = urequests.get('https://flagtech.github.io/flag.txt')
>>> print(res.text)
FLAG
```

用程式取得 AQI 空氣指數

只要使用 urequests.get() 來連線剛剛取得的網址，就可以得知我們所在地當天的 AQI 空污指數：

```
>>> import urequests
>>> res = urequests.get("http://api.waqi.info/feed/taiwan/shi
hlin/?token=d08147c688ce823984d8a42281bdc01f7e3c3a53")
>>> print(res.text)
{"status":"ok", "data":{"aqi":44, "idx":1596,
"attributions":[{"url":"http://www.epa.ov.tw/", "name":"TAQM
- Taiwan Environmental Protection Agency (行政院環保署一空氣品質
監測網)", "logo":
...
```

JSON 格式資料解析


3 按此鈕
5 按 Tree Viewer 鈕
6 按此鈕展開
7 這是空污資料的資料結構
JSON Viewer
Save & Share
Result mode:
tree
Load Url
Browse
Tree Viewer
2 Tab Space
Beautify
View
Select a node...
object {2}
有 2 個欄位的物件
status : ok
data {9}
data 是內含 9 個欄位的物件
aqi : 44
idx : 1596
attributions [2]
0 {3}
url : http://www.epa.gov.tw
name : TAQM - Taiwan Environ Protection Agency (行品質監測網)
logo : Taiwan-EPA.png
1 {2}
url : https://waqi.info/
4 原本的文字會重新編排，清楚展現資料層級結構的樣貌
此欄位是目前的 AQI 空污指數

用程式讀取 JSON 資料

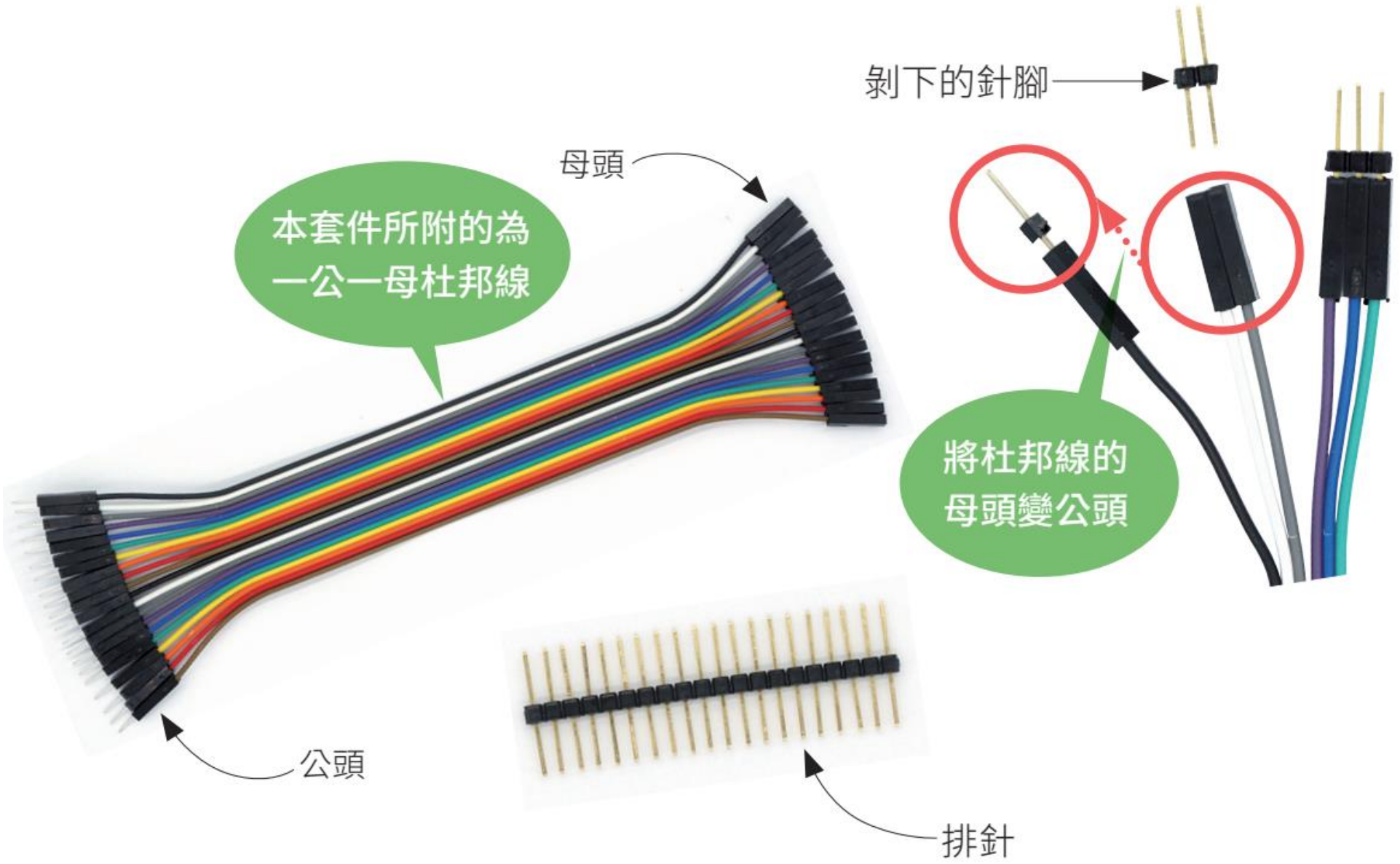
使用程式解讀 JSON 資料

為了解讀 JSON 格式的資料，urequests 模組提供了 json() 方法可以解析 JSON 格式，從文字形式轉換成 Python 內部使用的資料結構，使用方法非常簡單，以下假設 res 是使用 urequests.get() 取回的 JSON 格式 AQI 資料：

```
>>> j = res.json()                    ←載入並解析 JSON 格式資料
>>> print(j['data']['aqi'])           ←從 data 物件取得 aqi 欄位的資料
44
>>> print(j['data']['city']['name'])  ←從 data 物件取得 city
Shilin, Taiwan (士林)                    欄位物件的 name 欄位資料
```

json() 會將 JSON 資料中的陣列轉換為 Python 的串列 (list), 而 JSON 資料中的物件則會轉換為 Python 的字典 (dictionary), 所以我們只要用串列與字典的存取語法，即可將特定欄位的資料取出使用。

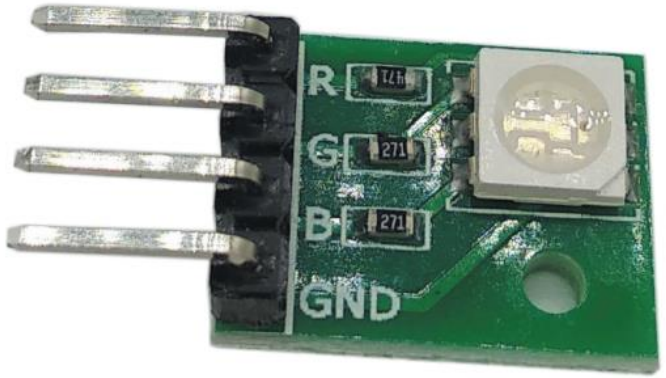
杜邦線與排針



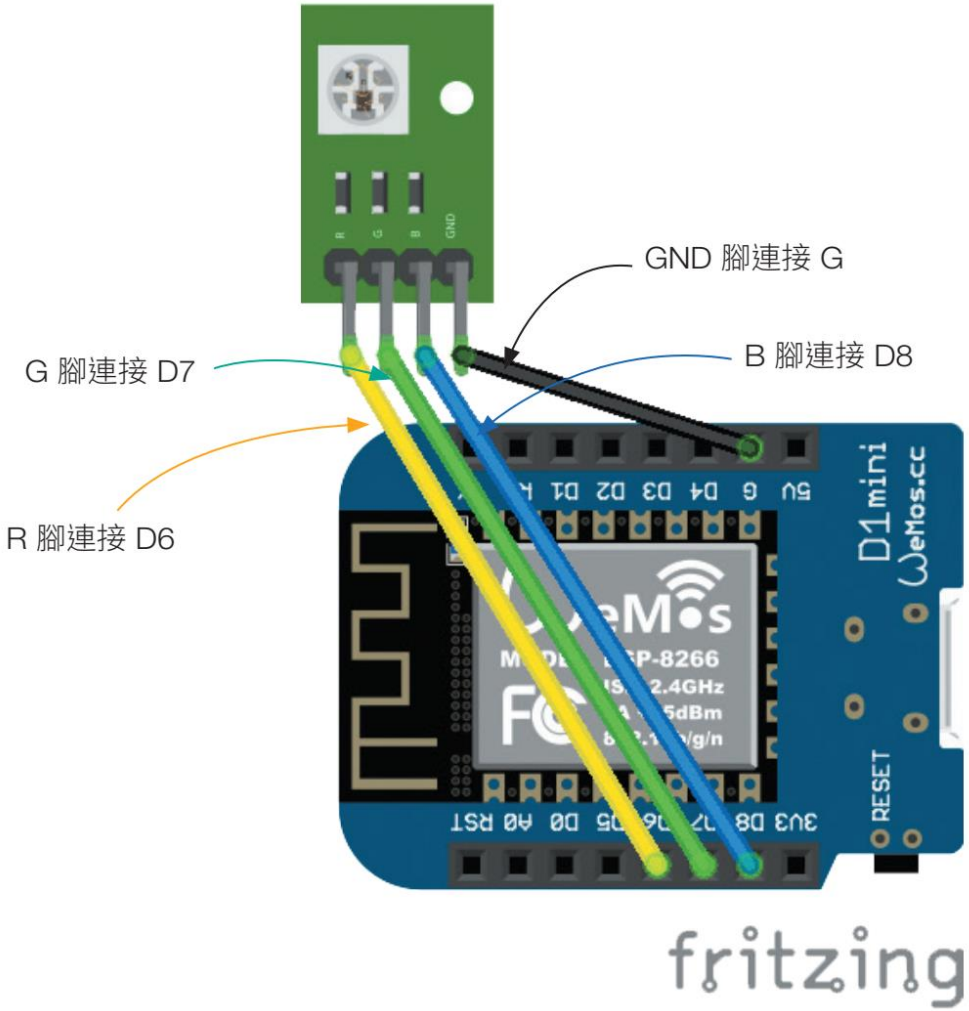
三色 LED 燈簡介

RGB 三色 LED 是把紅、藍、綠三個 LED 包裝成一顆 LED, 我們可以個別控制其發光，因此三色 LED 除了能夠單獨發出紅、綠、藍三種色光外，還可混搭出各種顏色的光。如果同時發出等亮的紅綠藍三種色光，則可產生白光。

第 1 章我們曾經說明 LED 的發光方式是長腳接高電位，短腳接低電位，像水往低處流一樣產生高低電位差，讓電流流過 LED 即可發光。若是長腳連接的電壓越高，LED 發出的光就會越亮。



RGB 三個腳位分別控制
紅、綠、藍 3 個色光



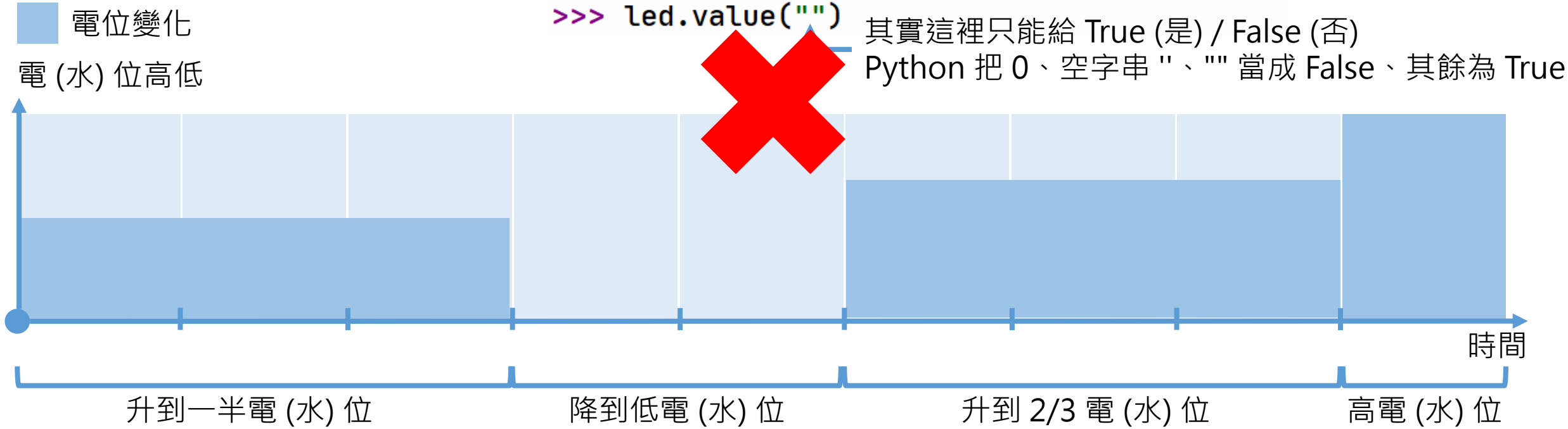
數位輸出不能控制大小

數位輸出只有高電位與低電位
沒有高、低電位間的中間大小

```
>>> from machine import Pin
>>> led = Pin(2, Pin.OUT)
>>> led.value(1)
>>> led.value(0)
>>> led.value(0.2)
>>> led.value(30)
>>> led.value("hello")
>>> led.value("")
```

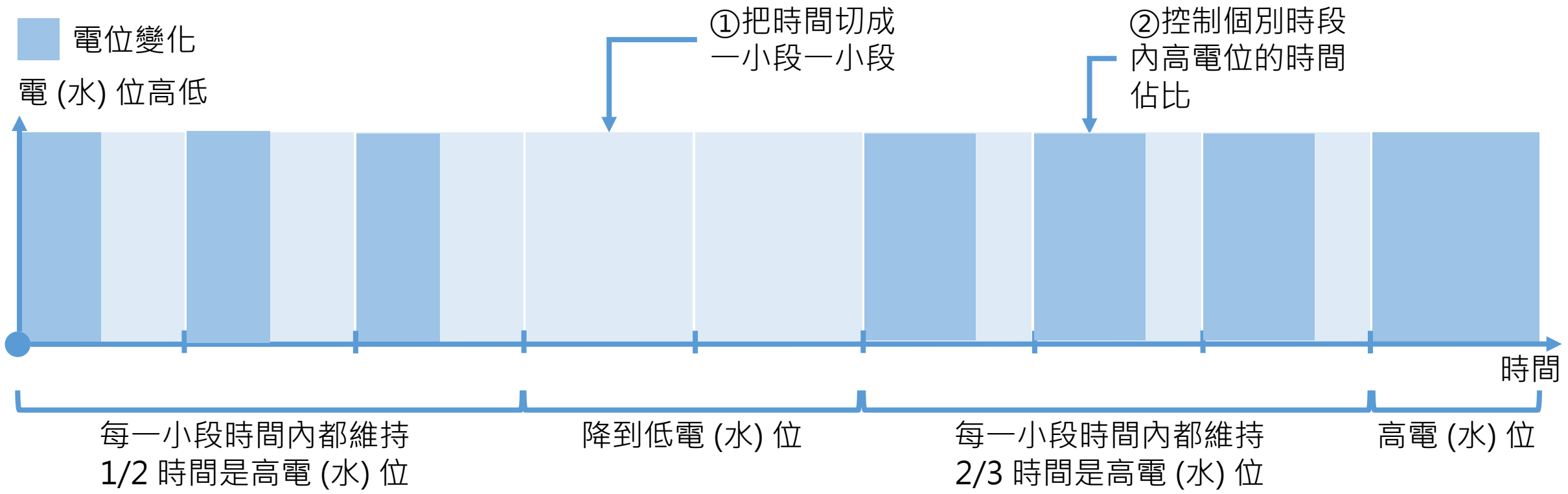
猜猜會怎樣？

其實這裡只能給 True (是) / False (否)
Python 把 0、空字串 ''、"" 當成 False、其餘為 True



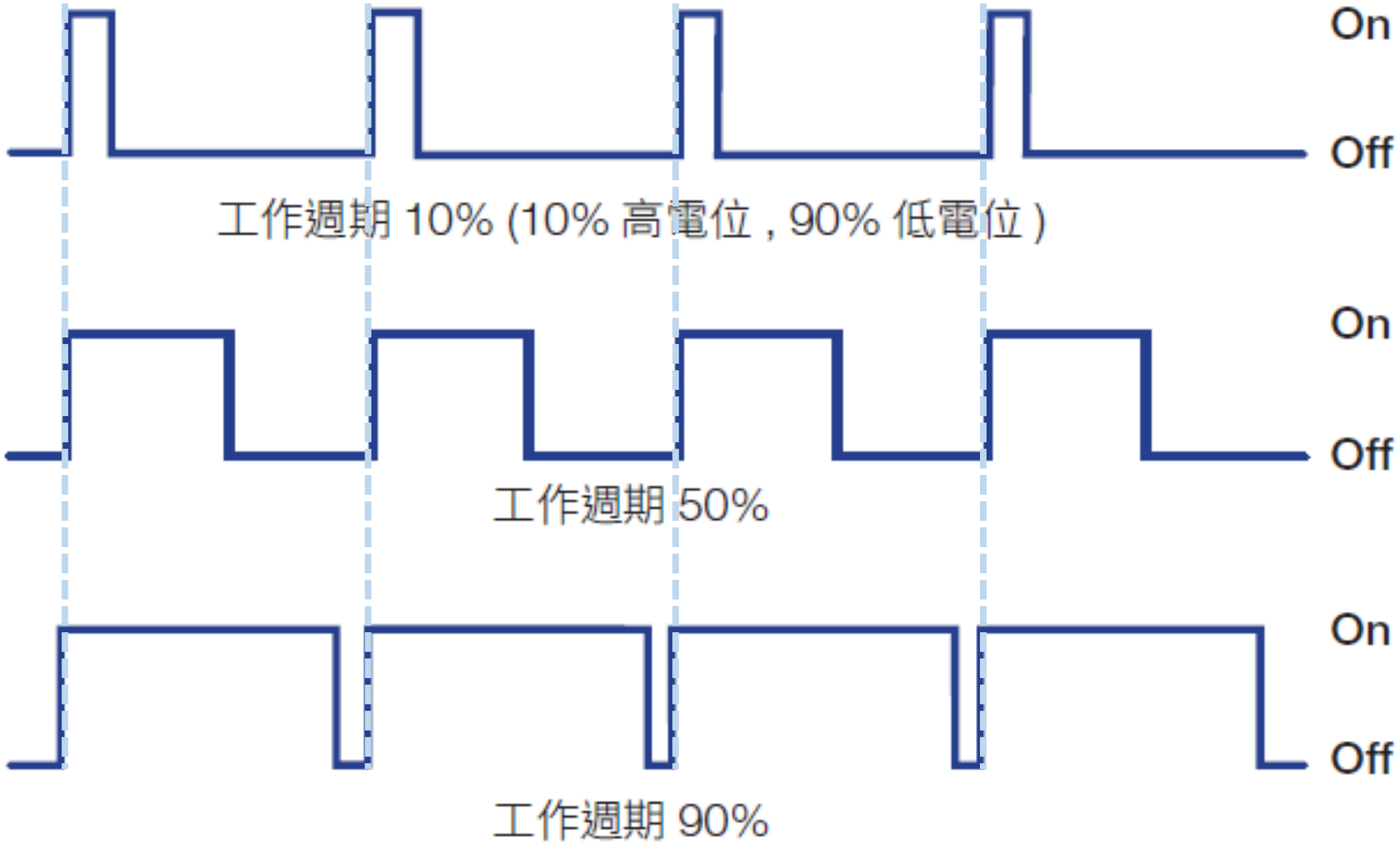
用時間控制大小變化的 PWM

原本→想直接控制高低電位間變化
改成→控制高、低電位的持續時間
就像→用踩腳踏車板時間控制速度
稱 **PWM (Pulse Width Modulation)**



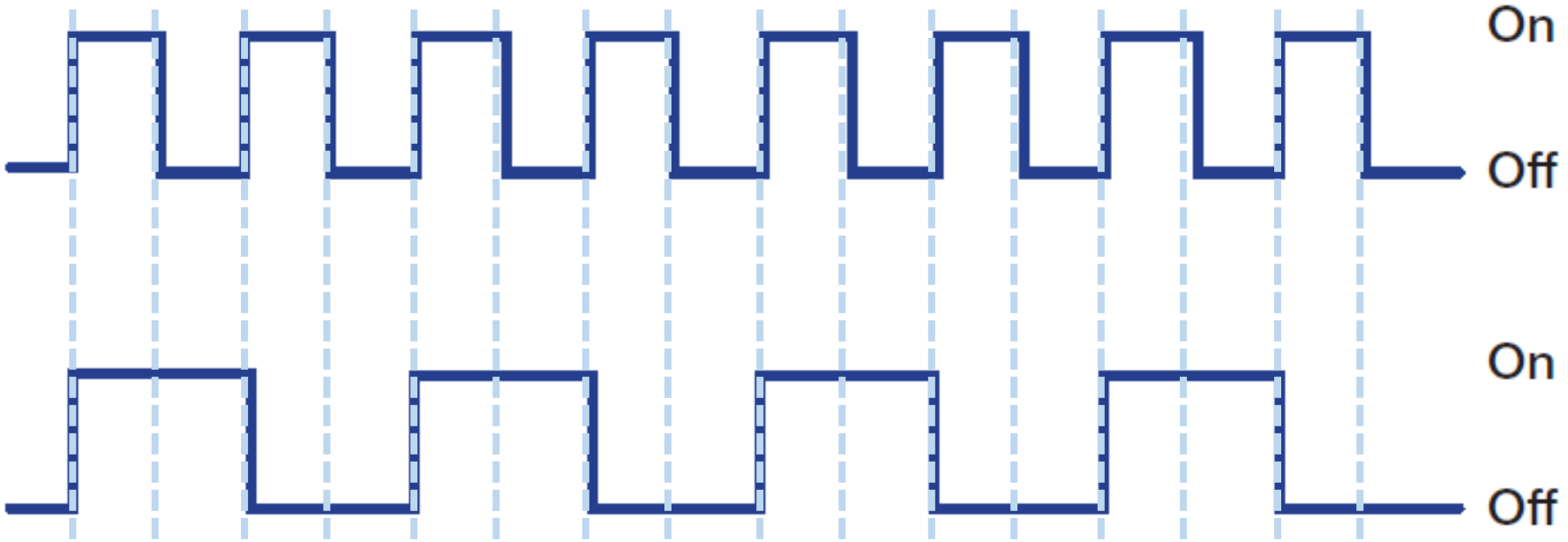
duty cycle 控制 PWM 產生的大小變化

工作週期 (duty cycle) 是單一時間切片內**高電位**佔的時間比例
工作週期越高模擬出來的電壓也越高



frequency (頻率) 控制 PWM 的變換速度

頻率 (duty cycle) 控制要將時間切成多細
要切多細主要是看要控制的裝置



兩者工作週期皆為 50%，輸出電壓相同，
但上面的輸出頻率是下面的 2 倍

MicroPython 中的 PWM

```
from machine import Pin, PWM # 匯入 Pin 和 PWM 子函式庫
```

```
buzzer = PWM(Pin(15, Pin.OUT))
buzzer.freq(0)
buzzer.duty(512)
```

頻率 1~1000Hz

工作週期 0~1023 -> 0%~100%

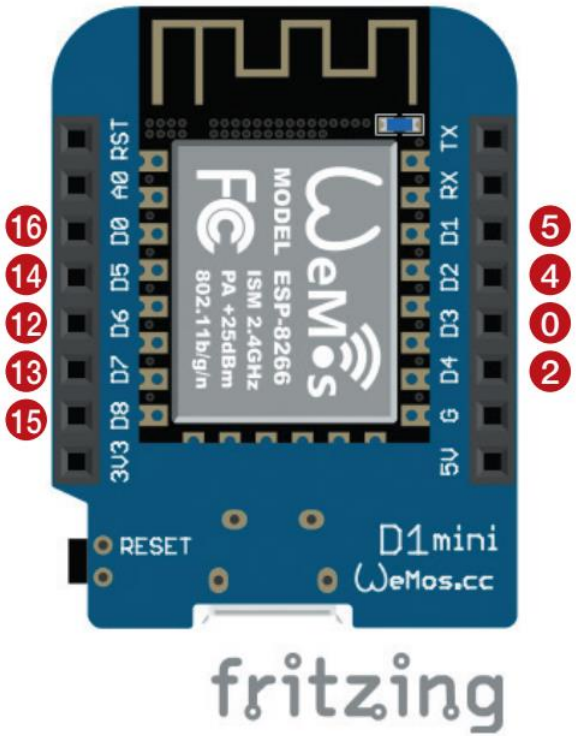
上面這三行程式也能合併成一行：

```
buzzer = PWM(Pin(15, Pin.OUT), freq=0, duty=512)
```

設計原理

當我們建立腳位的 Pin 物件後，將這個 Pin 物件作為參數再建立 PWM 物件，便可以設定這個腳位為 PWM 輸出腳位：

```
>>> from machine import Pin, PWM
>>> led = PWM(Pin(15))
```



然後即可使用 duty() 方法來指定 PWM 的輸出值：

```
>>> led.duty(1023) ←—— 設定 PWM 輸出值為 1023 (最亮)
>>> led.duty(0)    ←—— 設定 PWM 輸出值為 0 (最暗)
```

本章的 LED 是短腳連接低電位，長腳連接 D6～D8 腳位，與第 1 章的 LED 不同，所以寫程式時應該設定 D6～D8 腳位為高電位，才能點亮 LED。而且用 PWM 輸出數值越高，代表電位越高，亮度就會越亮。

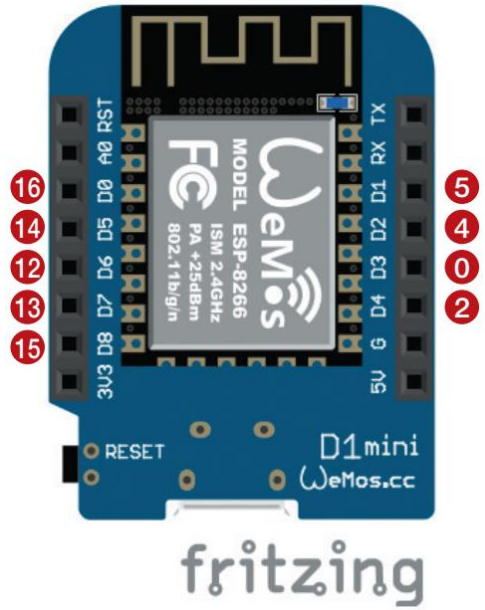
Lab06

PM2.5 空氣品質警報站

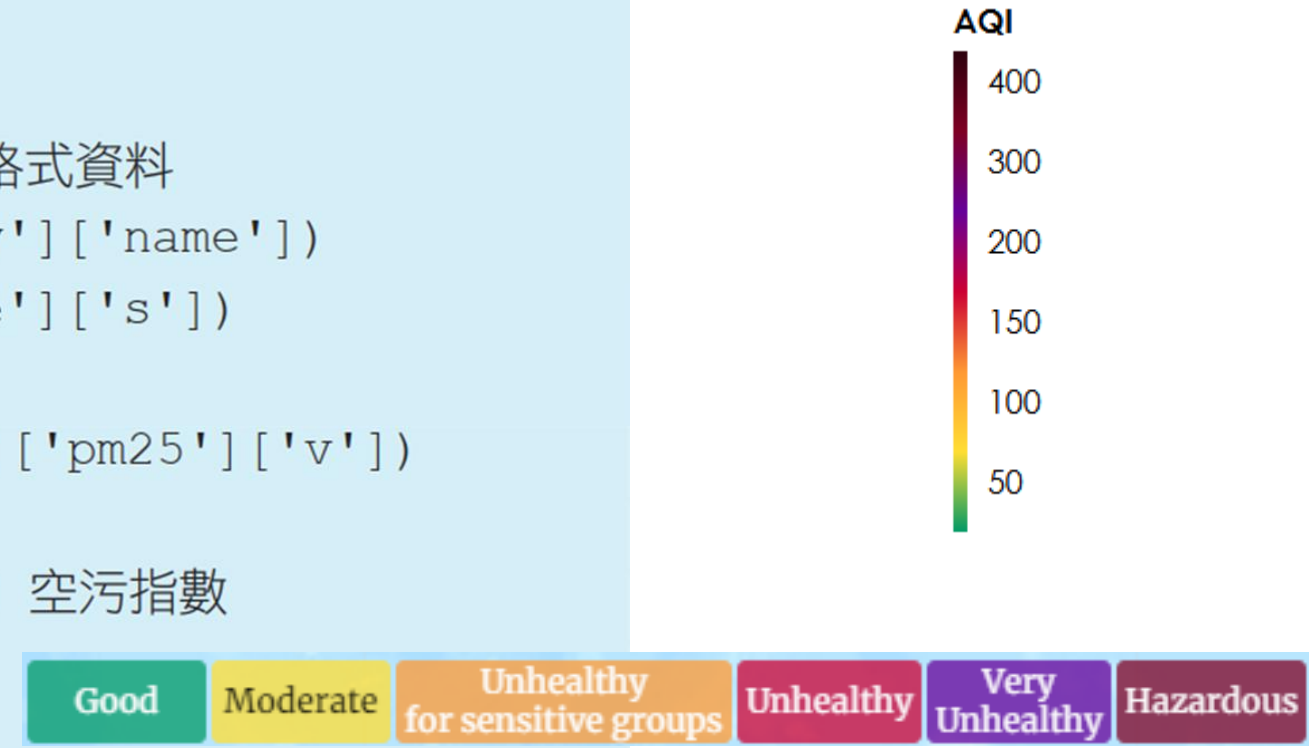
實驗目的	利用從政府資料開放平台取得的空污資料, 依據 AQI 指數顯示燈號。
材料	• D1 mini　　• RGB 三色 LED

程式設計

```
01 from machine import Pin, PWM
02 import time, network, urequests
03
04 # 連線 Wifi 網路
05 sta_if = network.WLAN(network.STA_IF)
06 sta_if.active(True)
07 sta_if.connect("Wi-Fi 基地台", "Wi-Fi 密碼")
08 while not sta_if.isconnected():
09     pass
10 print("Wifi 已連上")
11
12 rLED = PWM(Pin(12))   # 控制紅燈
13 gLED = PWM(Pin(13))   # 控制綠燈
14 bLED = PWM(Pin(15))   # 控制藍燈
```



```
15
16 while True:
17     # 取得 AQI 空污指數
18     res = urequests.get("AQI 網址")
19     j = res.json()  # 載入並解析 JSON 格式資料
20     print("測站名稱：", j['data']['city']['name'])
21     print("發布時間：", j['data']['time']['s'])
22     print("AQI：", j['data']['aqi'])
23     print("PM2.5：", j['data']['iaqi']['pm25']['v'])
24
25     AQI = j['data']['aqi']  # 取得 AQI 空污指數
26
27     if AQI <= 50:
28         gLED.duty(1023); rLED.duty(0)   # 空氣品質良好顯示綠燈
29     elif 51 < AQI <= 100:
30         gLED.duty(300); rLED.duty(1023) # 空氣品質普通顯示黃燈
31     else:
32         gLED.duty(0); rLED.duty(1023)   # 空氣品質不佳顯示紅燈
33
34     time.sleep(1800)  # 每半小時更新一次燈號
```



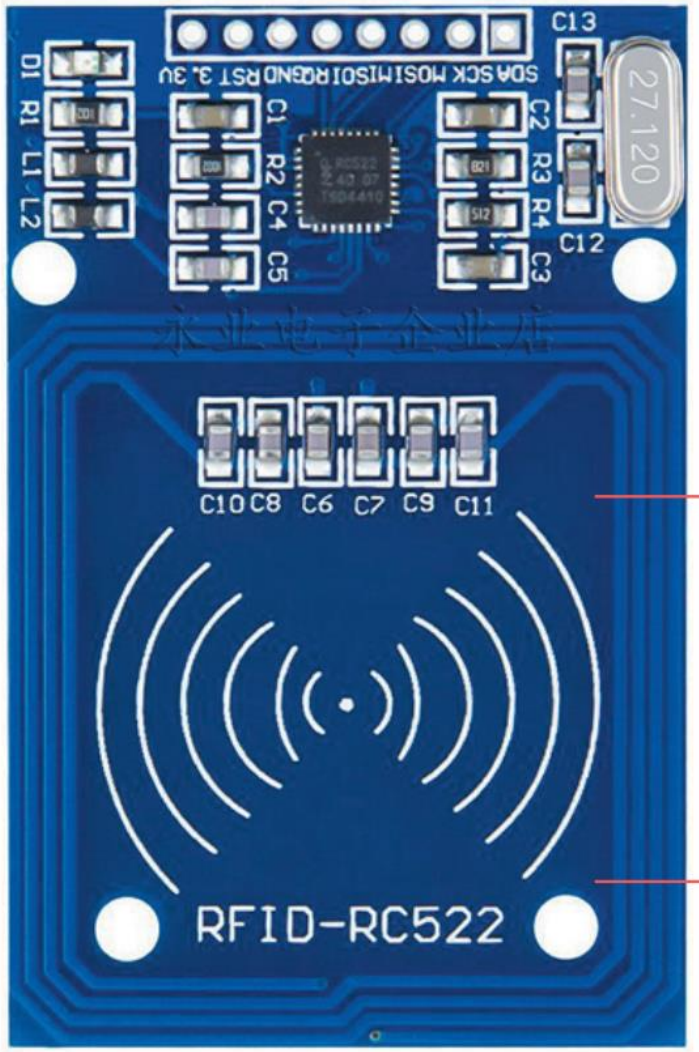
實測

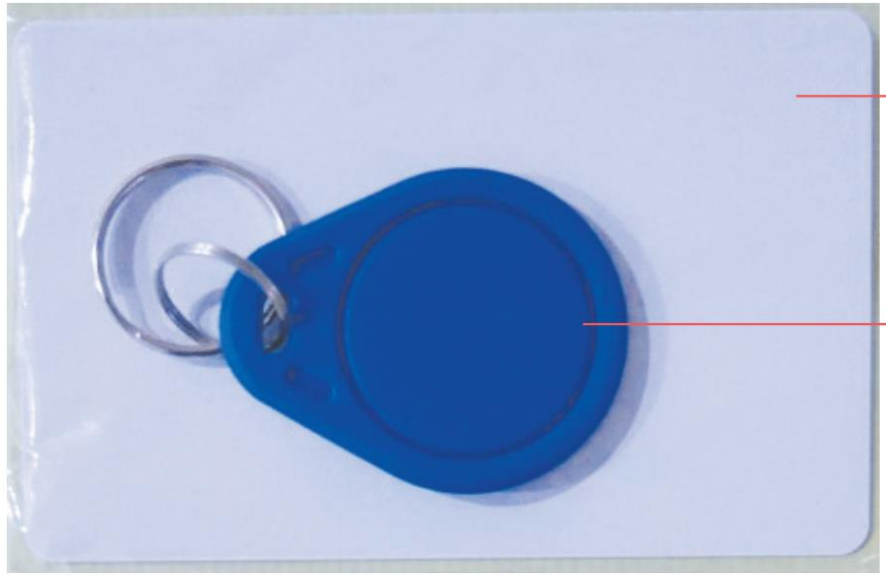
請按 F5 執行程式，可以在 Thonny 的 Shell 窗格看到程式輸出當日空氣品質的狀況，LED 也會依照 AQI 指數顯示相對應的燈號。

07 RFID 刷卡紀錄

RFID 模組

認識 RFID 感測模組





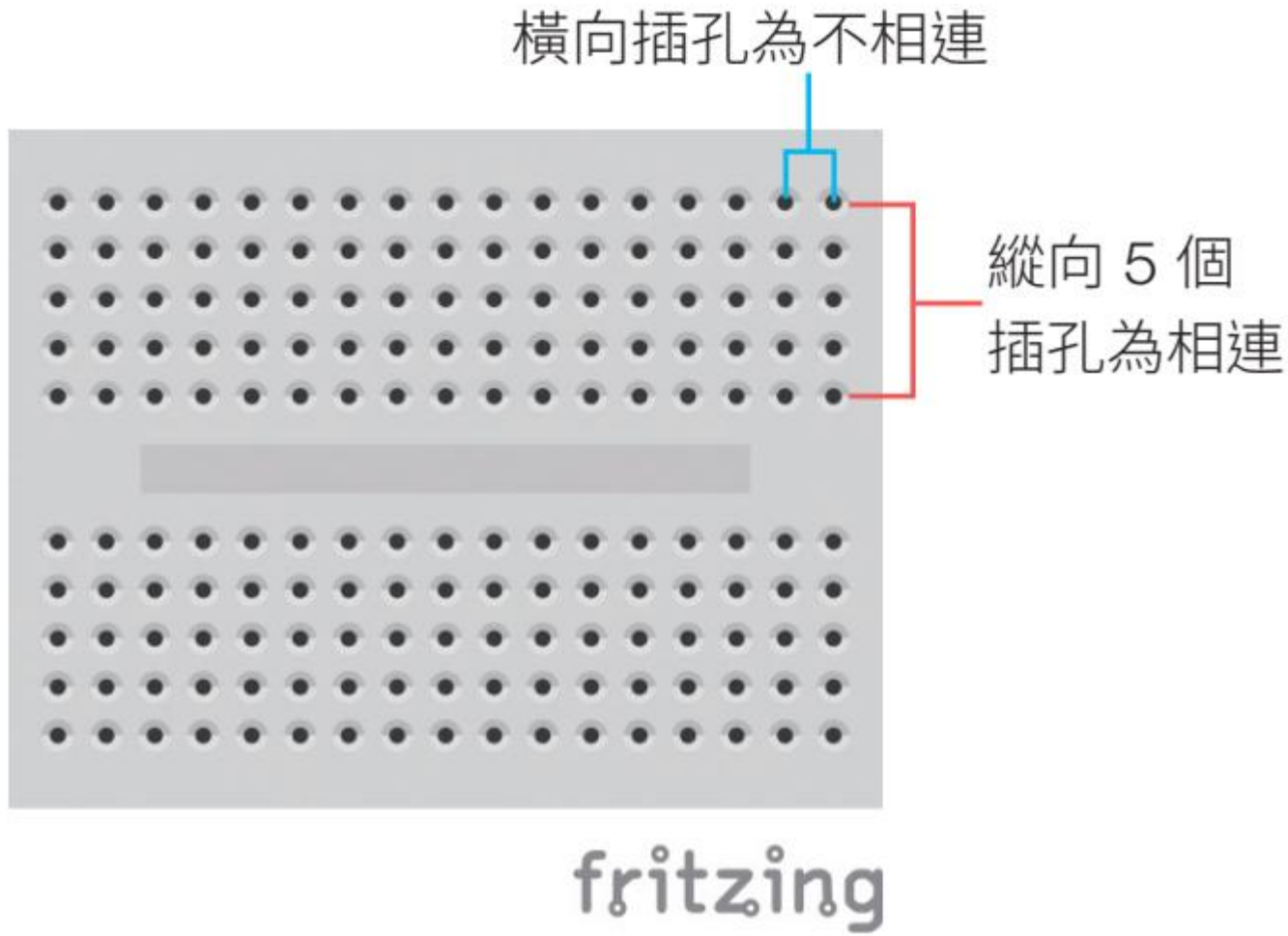
認識 RFID 感測模組

市面上常見的 RFID 感測模組有兩種頻率，分別是 125khz 與 13.56MHZ, 我們使用的是 13.56MHZ, 這個頻率與悠遊卡相同，所以我們的模組可以用來讀取悠遊卡的卡號。

本套件隨附了一個白色的 RFID 卡片以及藍色的 RFID 鑰匙圈，您可以使用本套件的 RFID 卡片或鑰匙圈來刷卡，也可以使用您自己的悠遊卡來刷卡。刷卡後 RFID 感測模組會將卡號傳給 D1 mini 控制板，D1 mini 控制板隨之將卡號送到雲端資料庫儲存，即可完成一個雲端的刷卡系統。

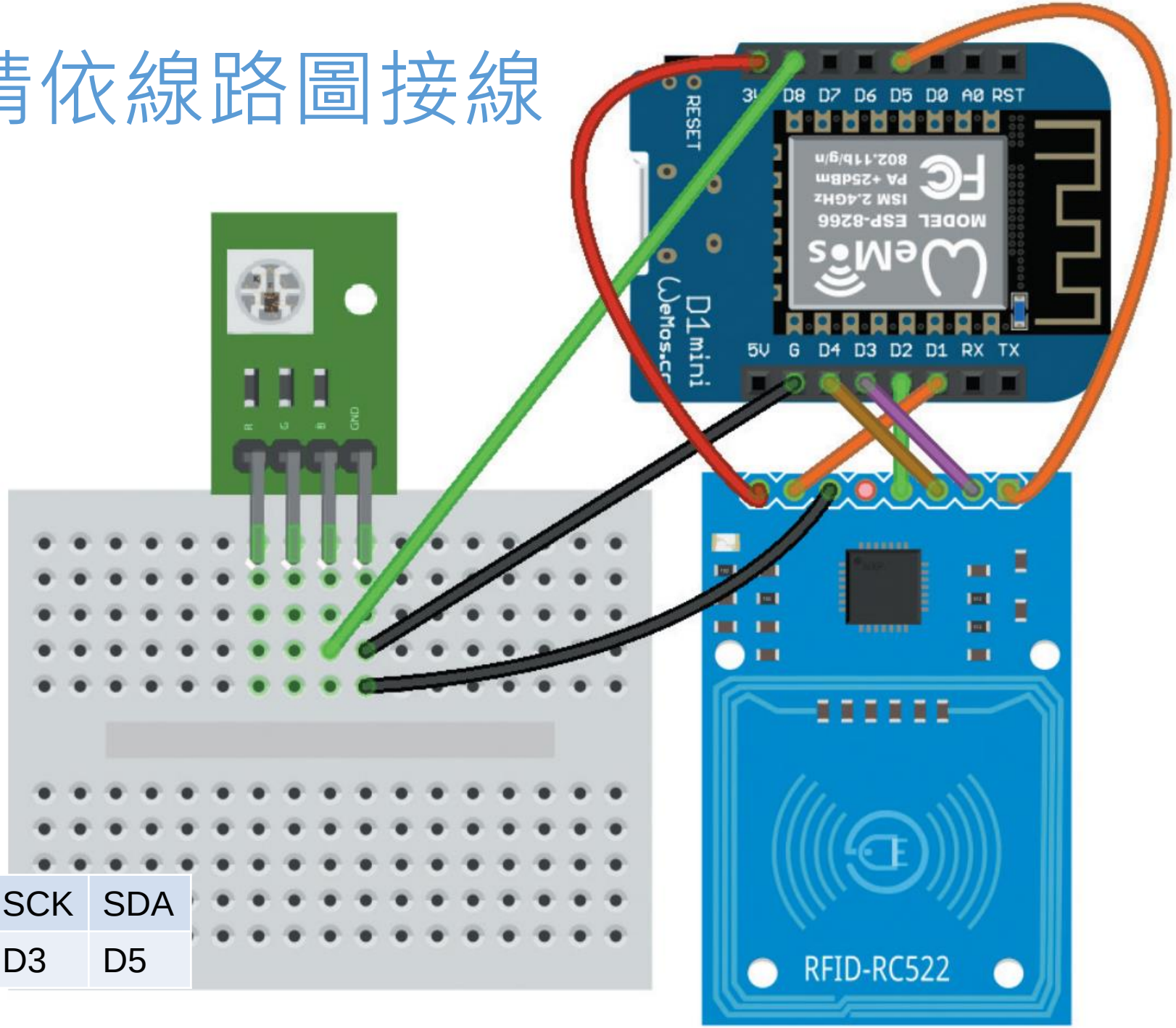
麵包板

麵包板的表面有很多的插孔。插孔下方有相連的金屬夾，當零件的接腳插入麵包板時，實際上是插入金屬夾，進而和同一條金屬夾上的其他插孔上的零件接通，在本套件實驗中我們就需要麵包板來連接 D1 Mini 與感測器模組。



杜邦線與排針

請依線路圖接線



線的顏色不必和圖一樣

3.3V	RST	GND	IRQ	MISO	MOSI	SCK	SDA
3V3	D1	G		D2	D4	D3	D5

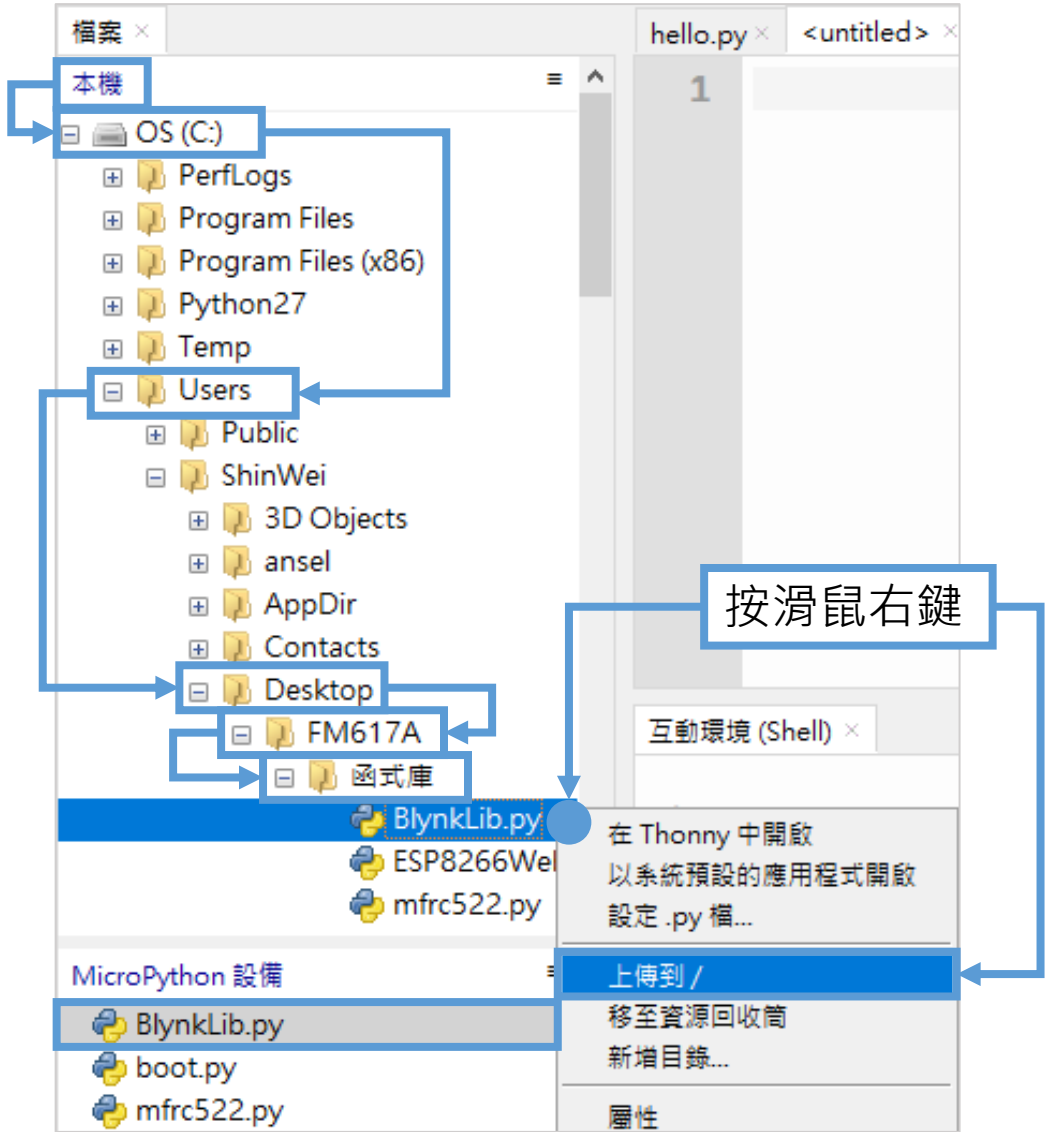
Lab07

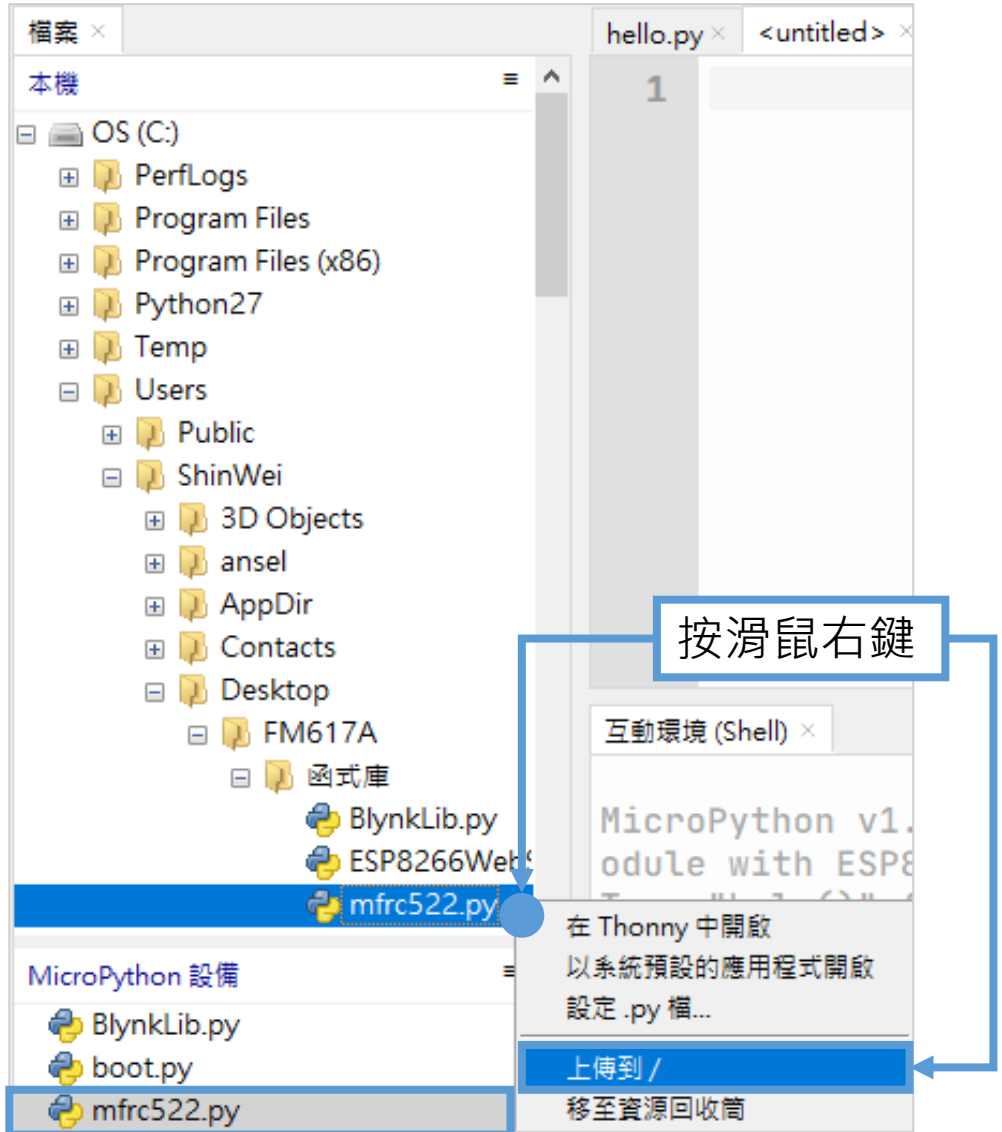
讀取悠遊卡與 RFID 卡的卡號

實驗目的	利用 RFID 感測模組讀取悠遊卡與 RFID 卡的卡號。
材料	• D1 mini　• RFID 感測模組　• RGB 三色 LED

安裝程式庫

如果使用 Thonny 3.1.2
請參考手冊上操作步驟





使用 RFID 程式庫

函式庫安裝後，在程式中如下匯入函式庫，並且建立 rfid 物件：

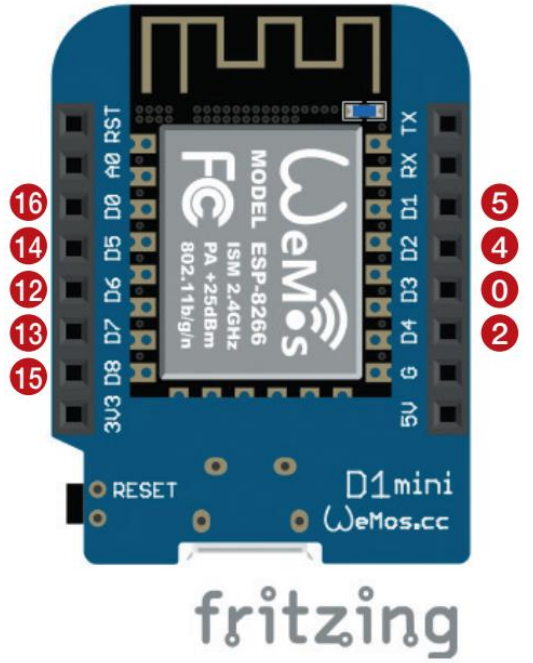
```
import mfrc522
rfid = mfrc522.MFRC522(0, 2, 4, 5, 14)
```

MOSI RST

SCK MISO SDA

然後便可以用 request() 方法來搜尋卡片：

```
stat, tag_type = rfid.request(rfid.REQIDL)
```



request() 方法有兩個回傳值，其中第一個回傳值如果是 0 就表示成功，代表有卡片靠近刷卡。第二個回傳值是 RFID 標籤 (tag) 的類型，此回傳值我們不會用到。

3.3V	RST	GND	IRQ	MISO	MOSI	SCK	SDA
3V3	D1	G		D2	D4	D3	D5
	5			4	2	0	14

成功找到卡片後，請立刻使用 anticoll() 方法來讀取卡號：

```
stat, raw_uid = rfid.anticoll()
```

rfid.OK

anticoll() 方法也有兩個回傳值，同樣地第一個回傳值如果是 0 就表示成功，若不是 0 代表使用者可能中途抽走卡片導致讀取失敗。若成功的話，第二個回傳值便是二進位格式的卡號。

RFID 的卡號總共有 4 碼，因為二進位格式的數值無法正常顯示在螢幕上，所以我們可以透過 Python 的格式化字串功能，將二進位格式的卡號轉為 16 進位的字串：

```
id = "%02x%02x%02x%02x" % (
                raw_uid[0], raw_uid[1], raw_uid[2], raw_uid[3])
```

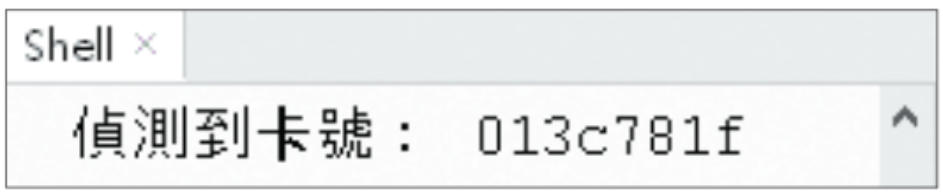
在字串中以 % 開頭的 %02x 代表格式化參數，會依序被中間 % 算符之後的元素取代，本例 %02x 代表要將對應的整數轉為兩位十六進位數字來表示。

程式設計

```
01 from machine import Pin
02 import mfrc522, time
03
04 rfid = mfrc522.MFRC522(0, 2, 4, 5, 14)
05 led = Pin(15, Pin.OUT)
06
07 while True:
08
09     led.value(0)  # 搜尋卡片之前先關閉 LED
10     stat, tag_type = rfid.request(rfid.REQIDL) # 搜尋 RFID 卡片
11
12     if stat == rfid.OK:  # 找到卡片
13         stat, raw_uid = rfid.anticoll()  # 讀取 RFID 卡號
14         if stat == rfid.OK:
15             led.value(1)  # 讀到卡號後點亮 LED
16
17             # 將卡號由 2 進位格式轉換為 16 進位的字串
18             id = "%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1],
19                                        raw_uid[2], raw_uid[3])
20             print("偵測到卡號：", id)
21
22             time.sleep(0.5)  # 暫停一下, 避免 LED 太快熄滅看不到
```

實測

請按 F5 執行程式，然後使用悠遊卡或本套件附的 RFID 卡片靠近 RFID 感測模組，看到三色 LED 亮藍燈後，即可在 Thonny 的 Shell 窗格看到卡號：

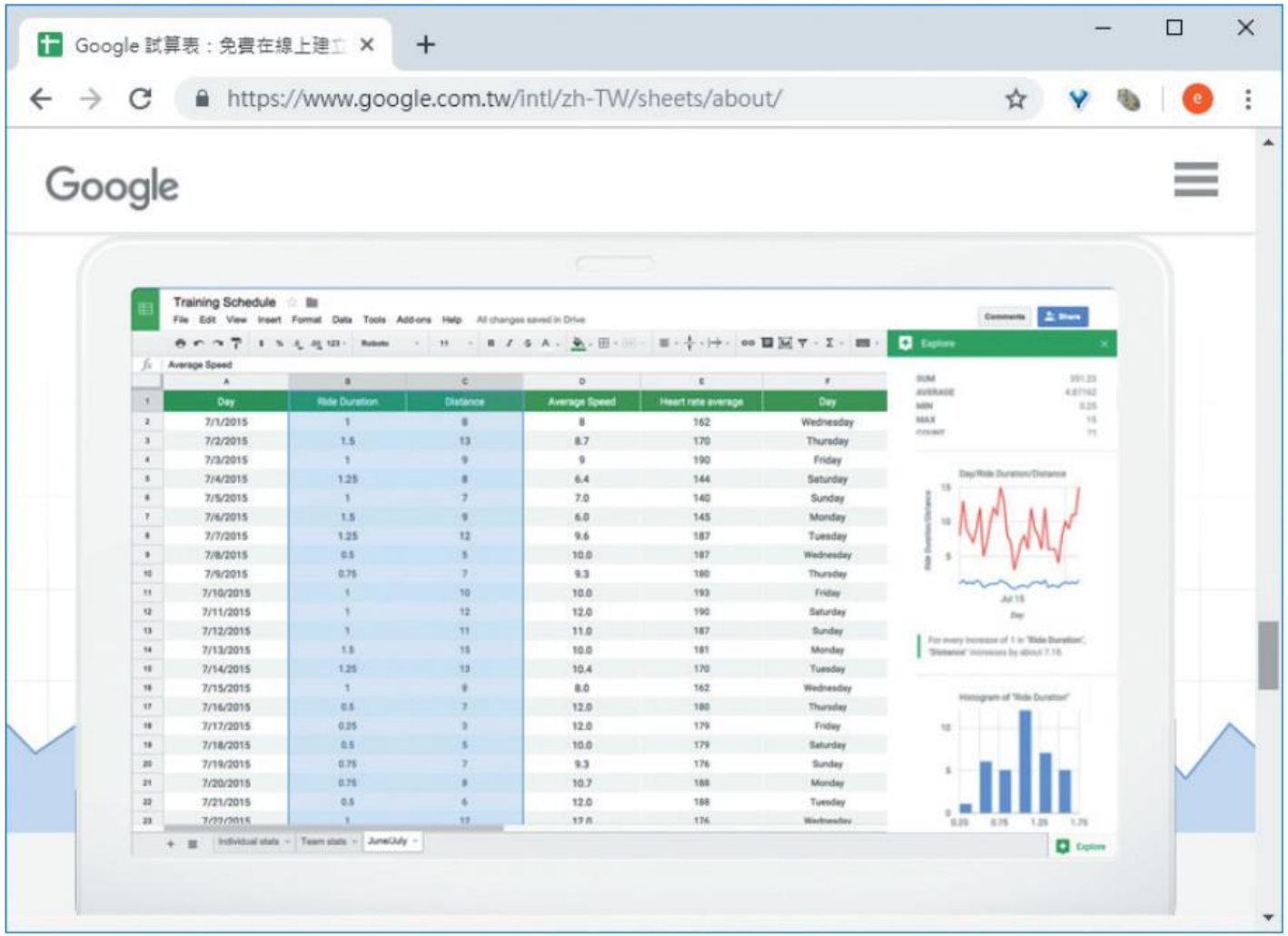


您可以將 LED 改換成蜂鳴器 (本套件未附), 這樣刷卡後就會嗶一聲。

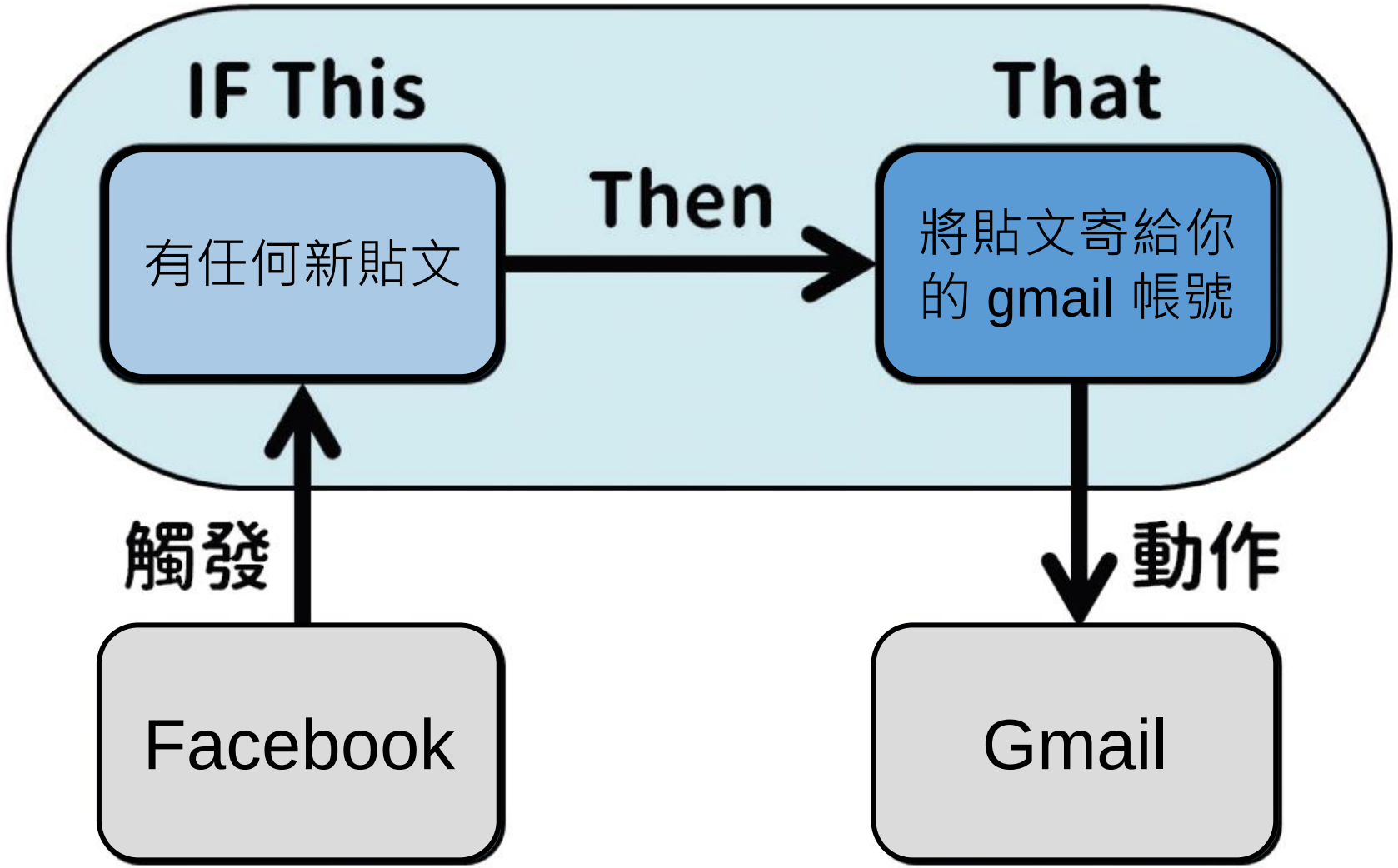
08 使用 Google 試算表儲存資料

雲端資料庫

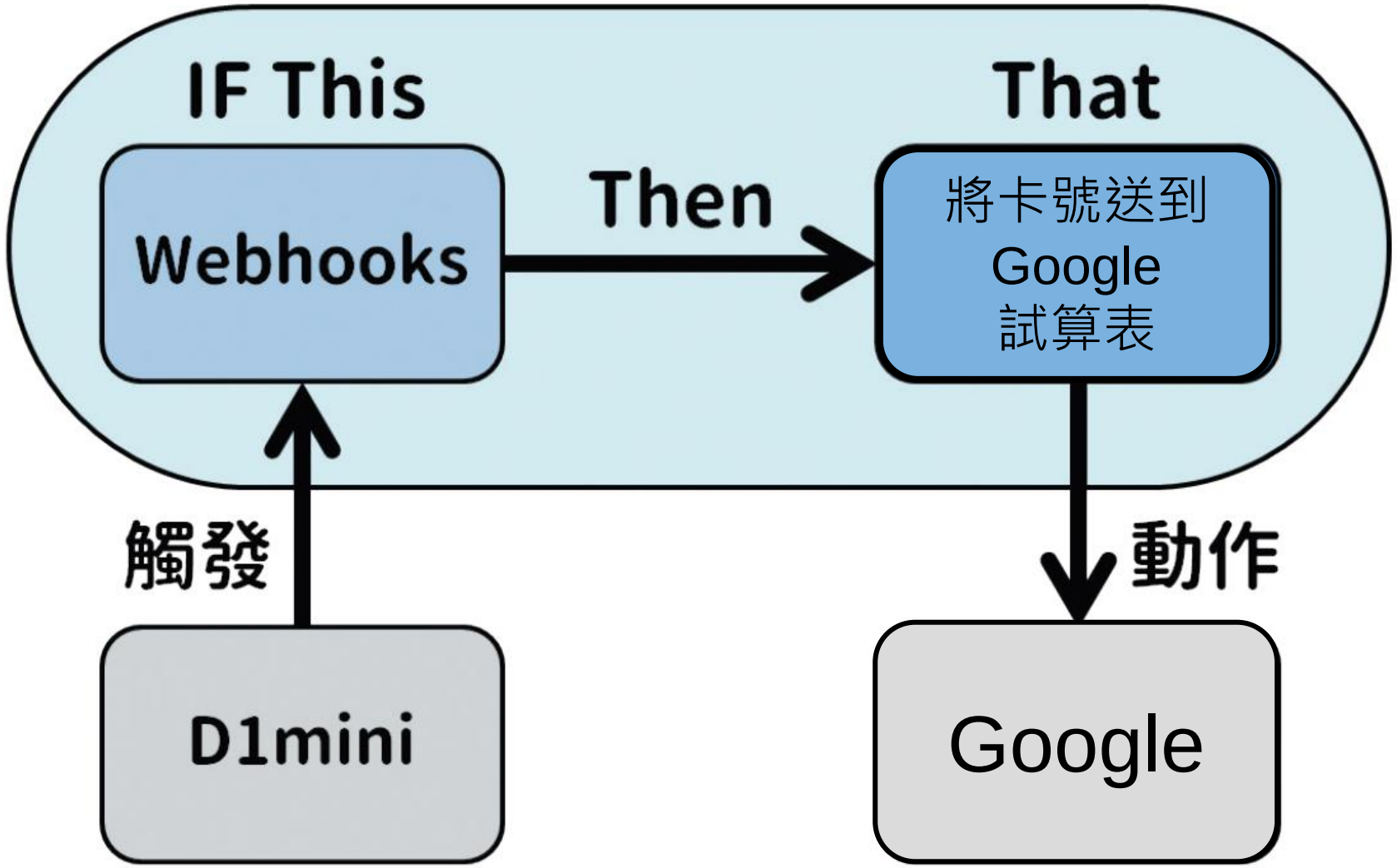
我們將使用免費的 Google 試算表作為我們儲存資料的雲端資料庫，Google 試算表類似我們常用的微軟 Excel 試算表，除了可以儲存資料以外，還可以利用這些資料來進行分析與產生圖表。



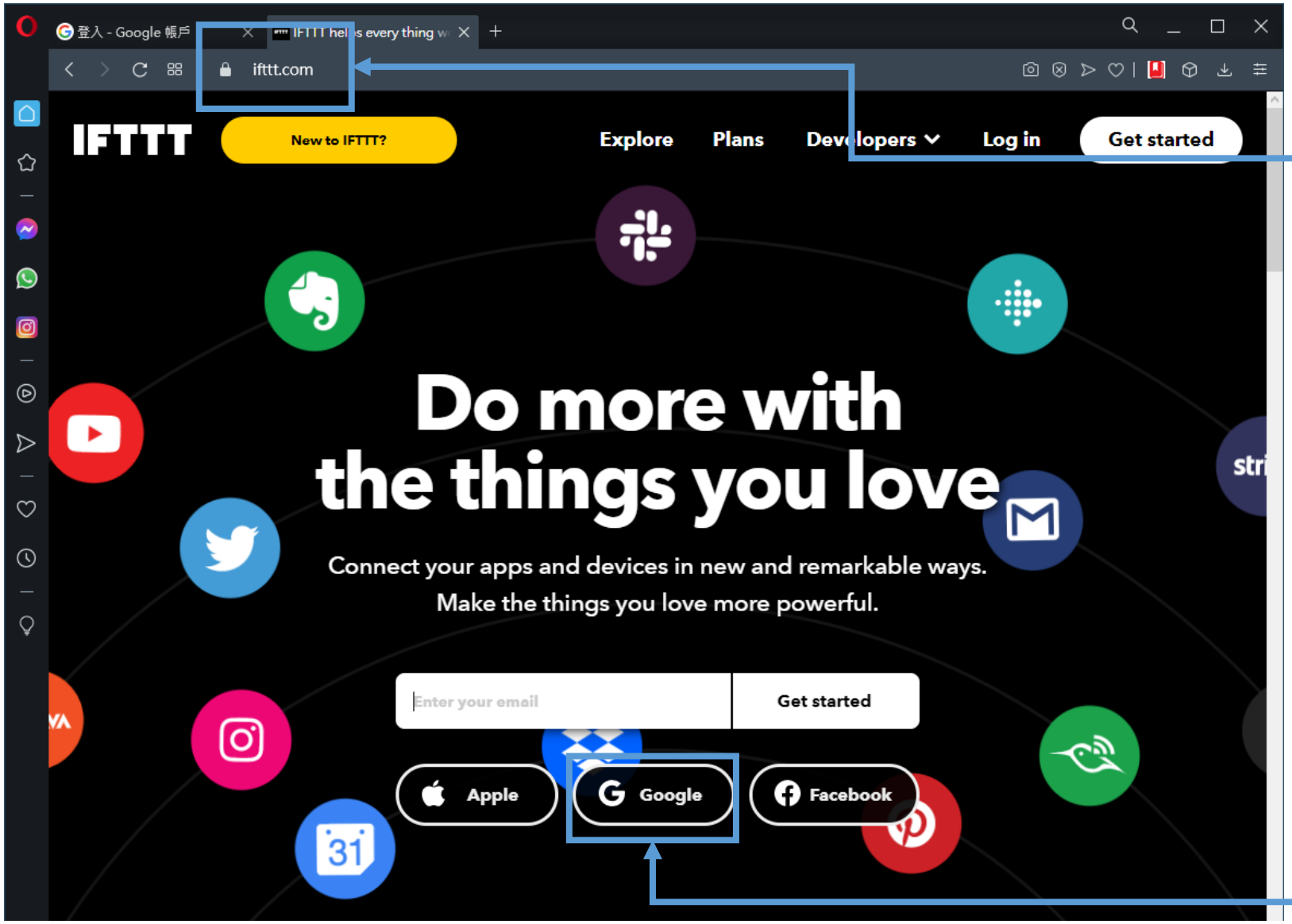
IFTTT 服務簡介


IF This
That
有任何新貼文
Then
將貼文寄給你的 gmail 帳號
觸發
動作
Facebook
Gmail

改由 D1 mini 從程式中觸發事件



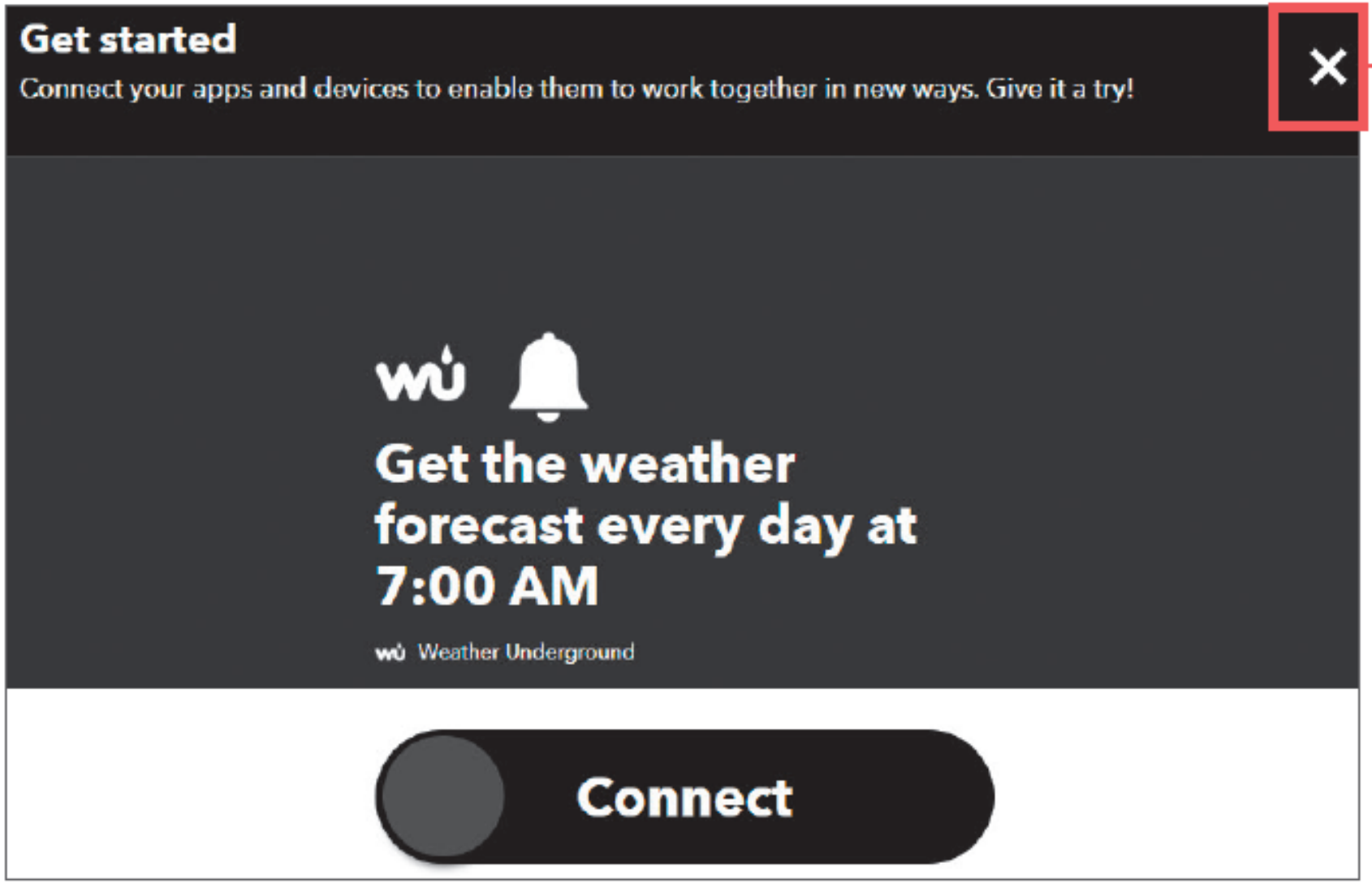
註冊 IFTTT 帳號 (ifttt.com)



請在網址列輸入
ifttt.com

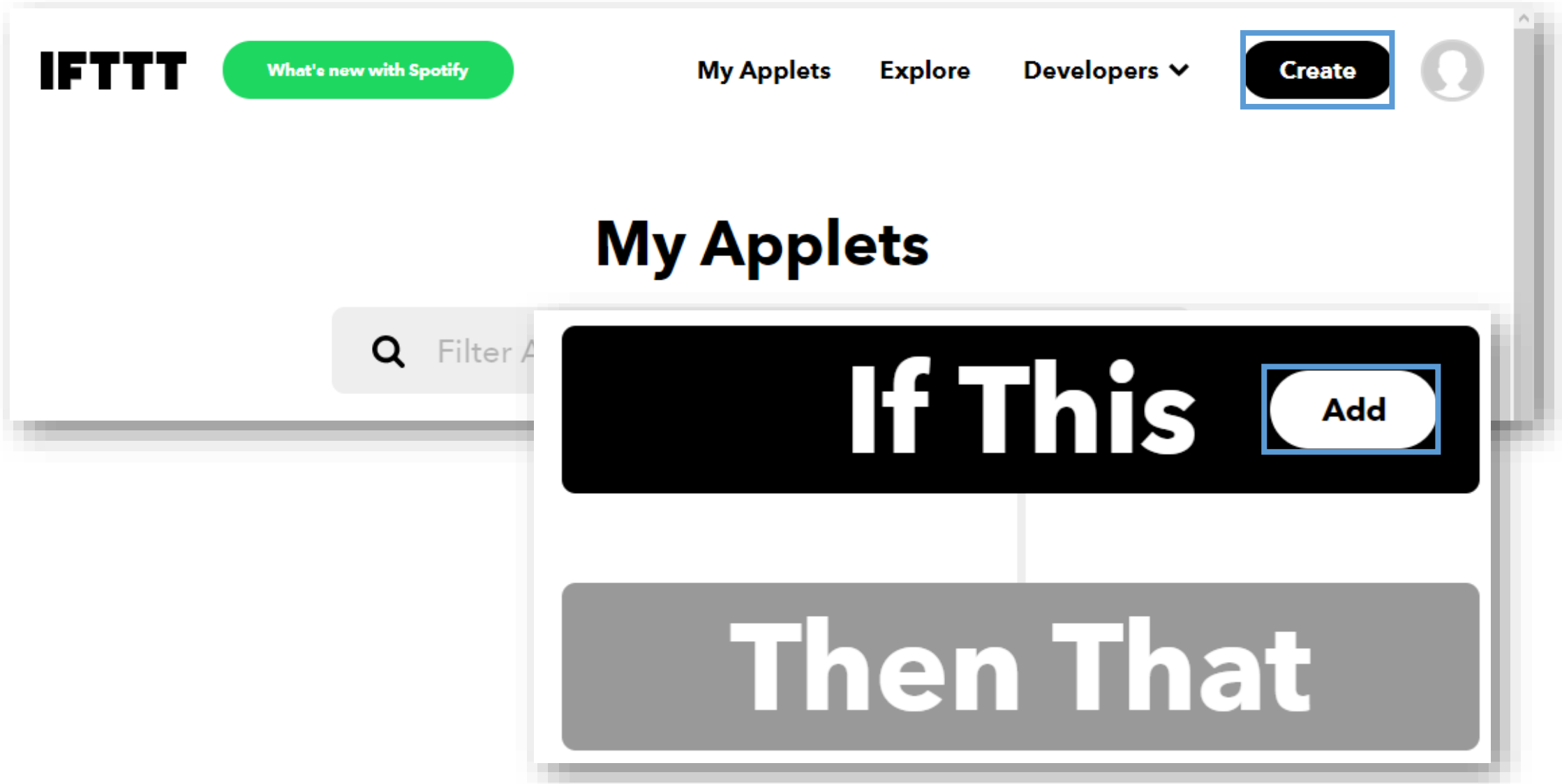
等下會需要用到 Google 服務
直接用 Google 帳號比較方便

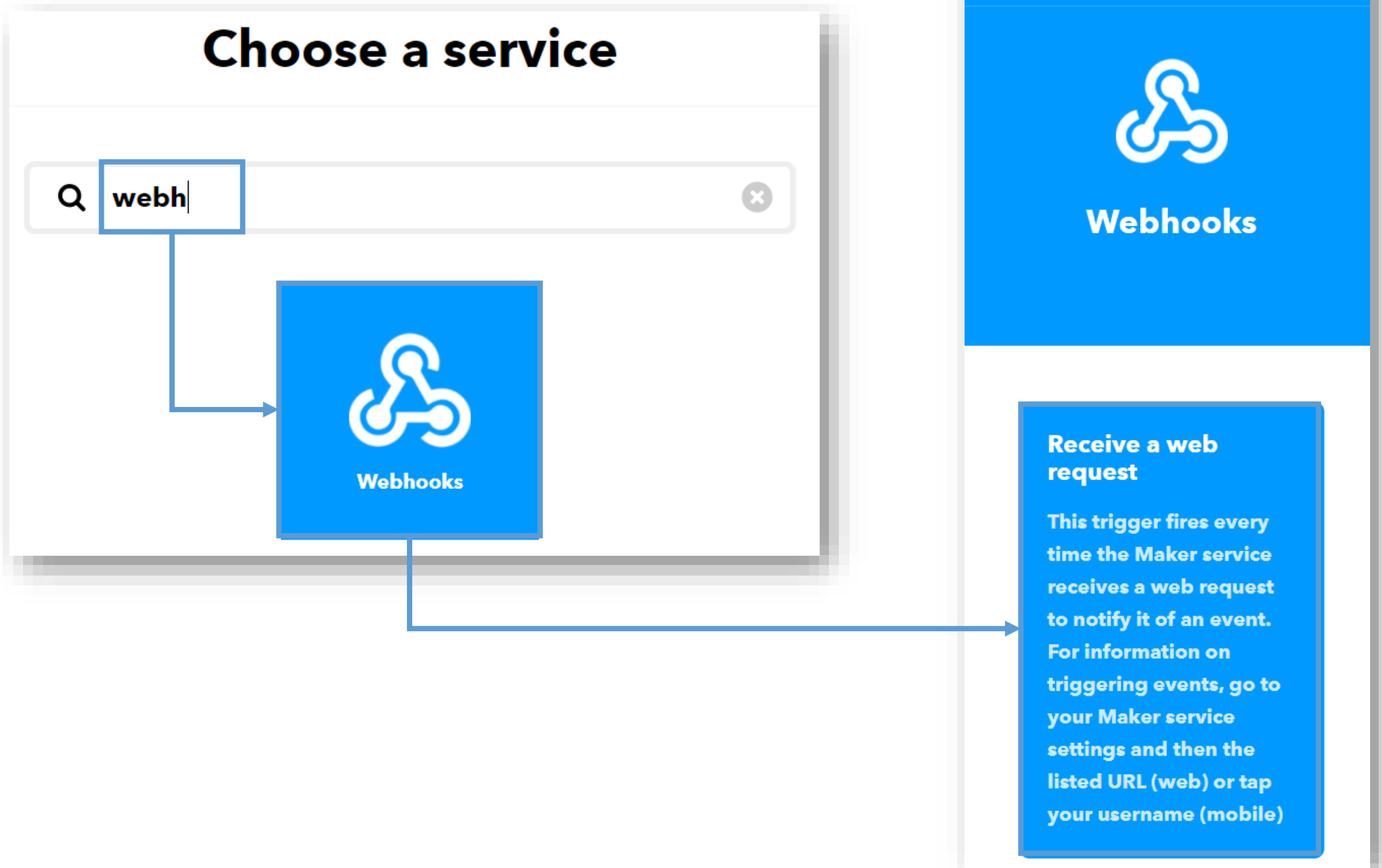
註冊 IFTTT 帳號

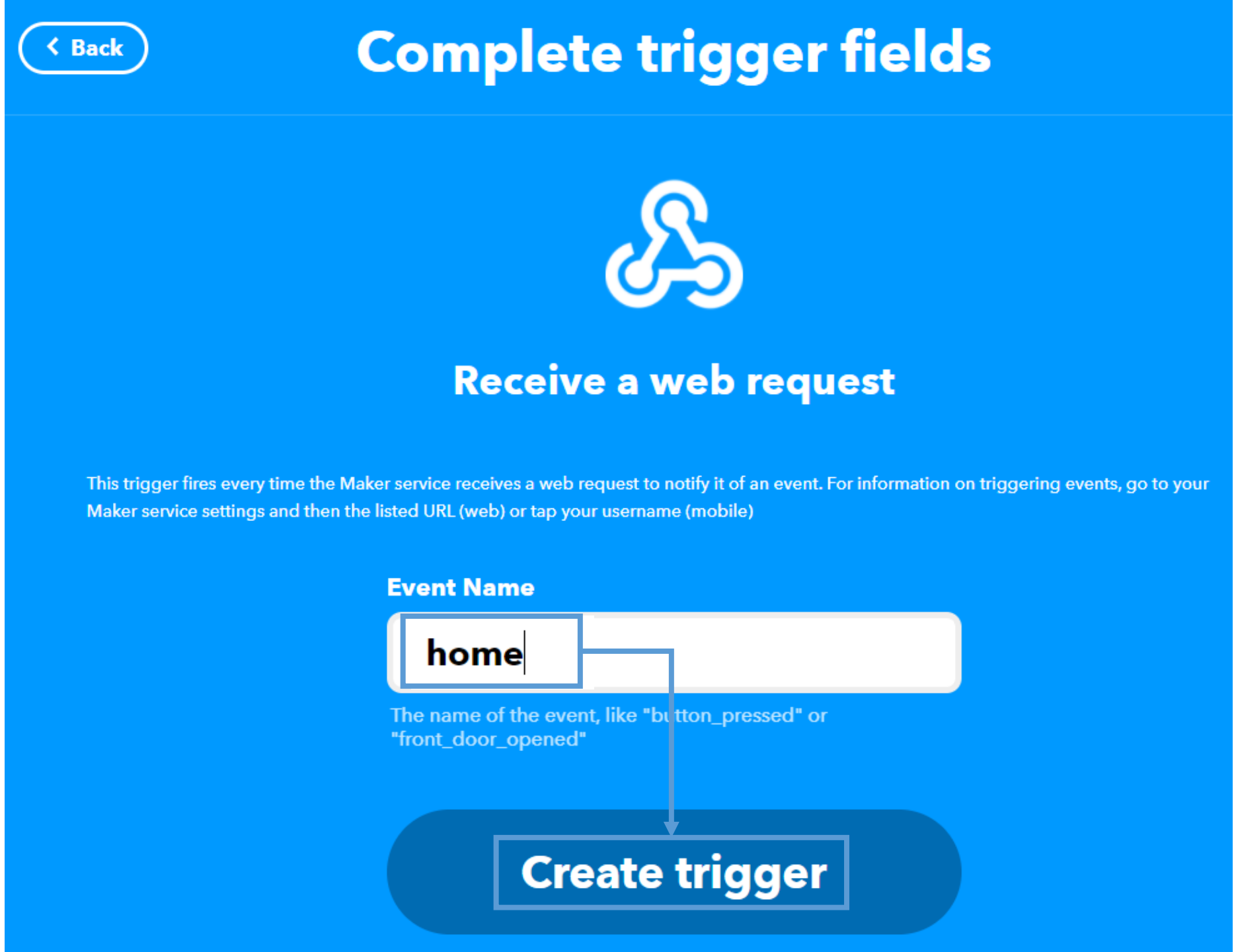


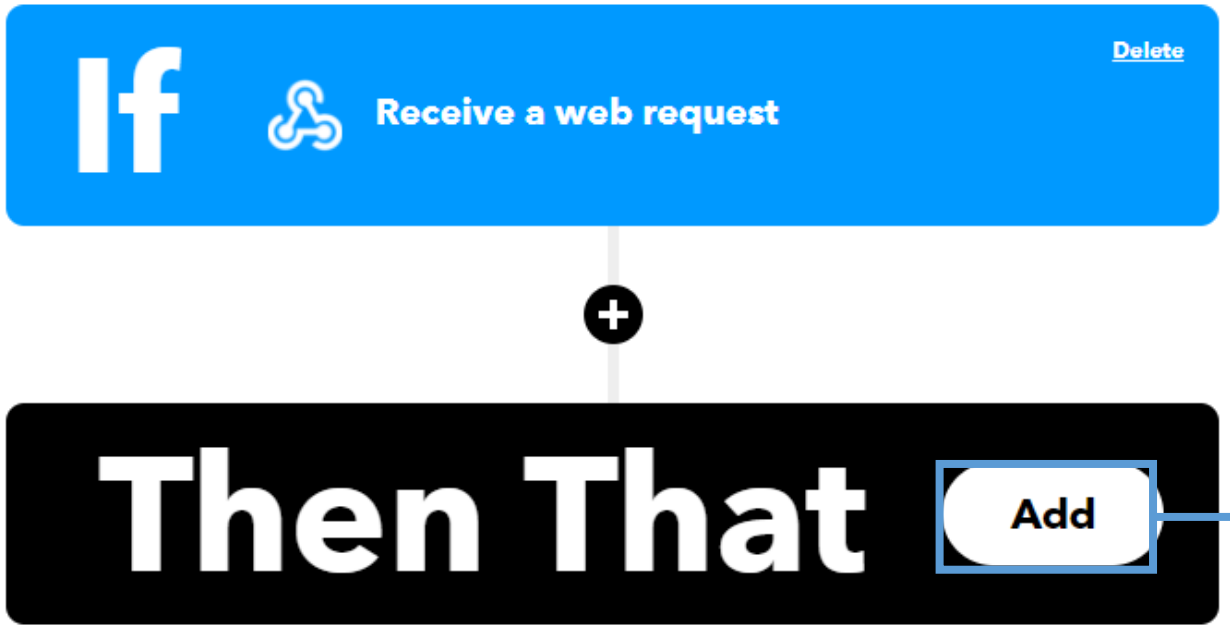
6 申請完成後，若出現推薦服務畫面，點右上角叉叉關閉。

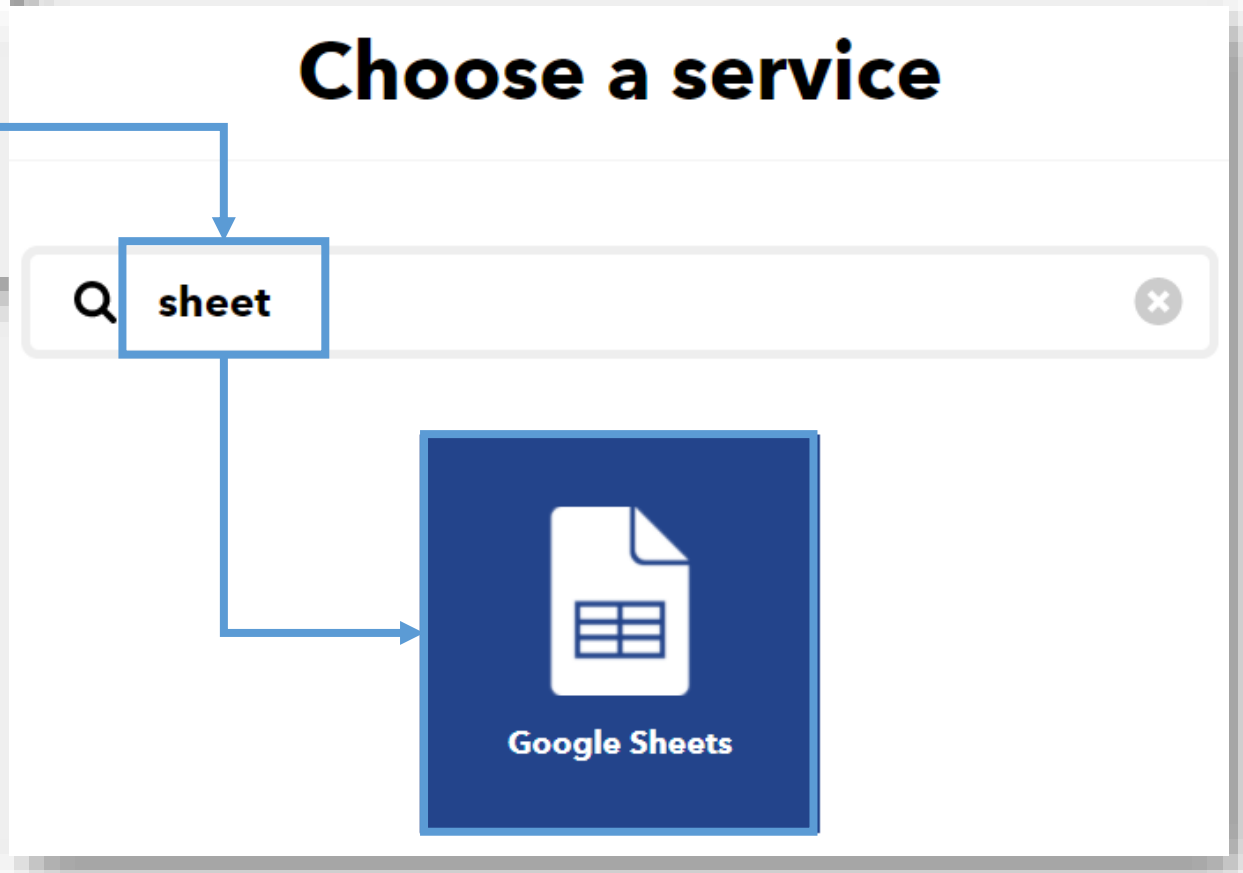
建立新的 Applet


IFTTT
What's new with Spotify
My Applets
Explore
Developers
Create
My Applets
Filter A
If This
Add
Then That


Choose a service
webh
Webhooks
Choose a trigger
Webhooks
Receive a web request
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)


Back
Complete trigger fields
Receive a web request
This trigger fires every time the Maker service receives a web request to notify it of an event. For information on triggering events, go to your Maker service settings and then the listed URL (web) or tap your username (mobile)
Event Name
home
The name of the event, like "button_pressed" or "front_door_opened"
Create trigger


If
Receive a web request
Delete
Then That
Add


Choose a service
sheet
Google Sheets

Choose an action

Google Sheets

Add row to spreadsheet

This action will add a single row to the bottom of the first worksheet of a spreadsheet you specify. Note: a new spreadsheet is created after 2000 rows.

Update cell in spreadsheet

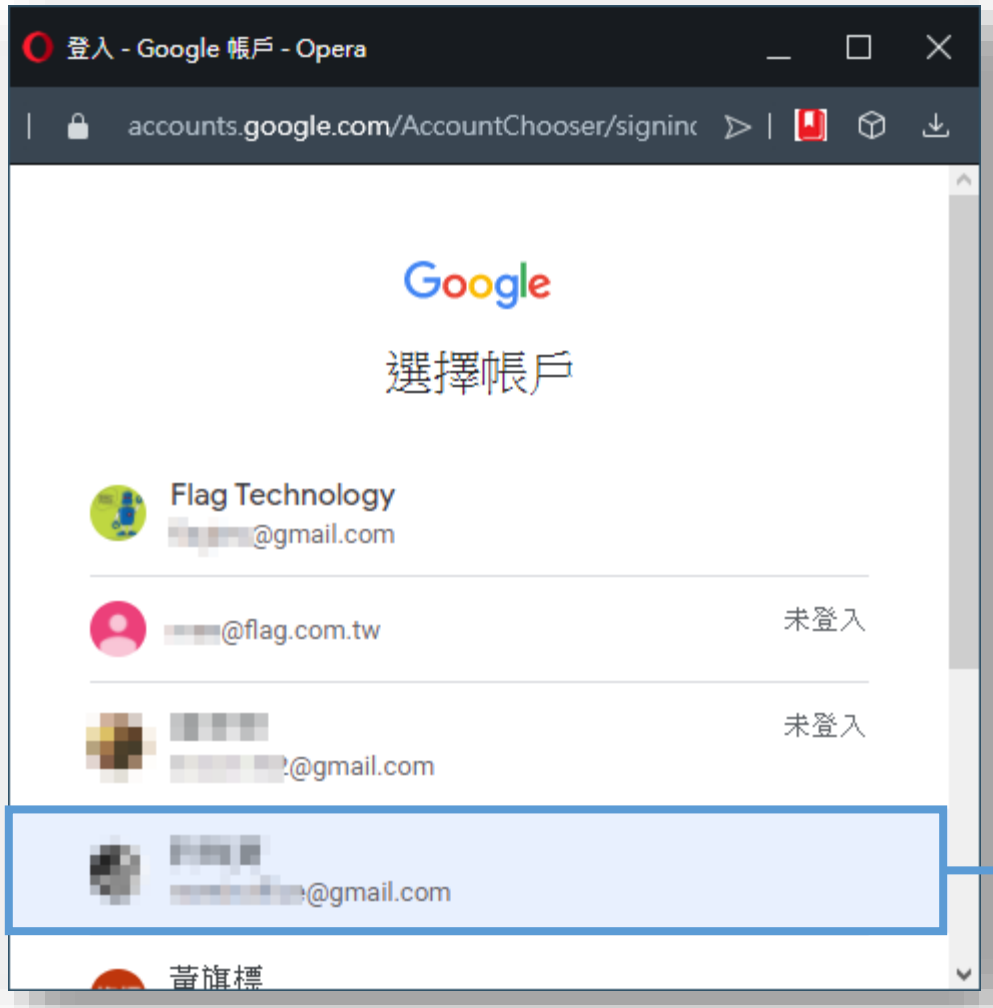
This action will update a single cell in the first worksheet of a spreadsheet you specify. Note: a new spreadsheet is created if the file doesn't exist.

Connect service

Google Sheets

Google Sheets lets you create and edit spreadsheets stored on your Google Drive. Turn on Applets to monitor specific cells in your spreadsheets as well create news docs, rows, and cell updates.

Connect


登入 - Google 帳戶 - Opera
accounts.google.com/AccountChooser/signinc
Google
選擇帳戶
Flag Technology
@gmail.com
@flag.com.tw
未登入
未登入
@gmail.com
@gmail.com
黃旗標


Google
「IFTTT」想要存取您的 Google 帳戶
@gmail.com
這麼做將允許「IFTTT」進行以下操作：
查看、編輯、建立及刪除您的所有 Google 雲端硬碟檔案
查看、編輯、建立及刪除 Google 雲端硬碟中的試算表
連結您在 Google 上的個人資料
查看您的個人資訊，包括您已公開的任何個人資訊
檢視電子郵件地址
點選 [允許] 即表示您允許這個應用程式和 Google 依據各自的《服務條款》和《隱私權政策》使用您的資訊。 您隨時可以變更這項權限與其他帳戶權限。
拒絕
允許

Back

Complete action fields

Add row to spreadsheet

This action will add a single row to the bottom of the first worksheet of a spreadsheet you specify. Note: a new spreadsheet is created after 2000 rows.

Spreadsheet name

回家刷卡紀錄

Will create a new spreadsheet if one with this title doesn't exist — Add ingredient

Formatted row

OccurredAt ||| EventName ||| Value1

Use "|||" to separate cells — Add ingredient

Drive folder path

IFTTT

Format: some/folder/path (defaults to "IFTTT") — Add ingredient

Create action

If Receive a web request — Delete

Then Add row to spreadsheet — Delete

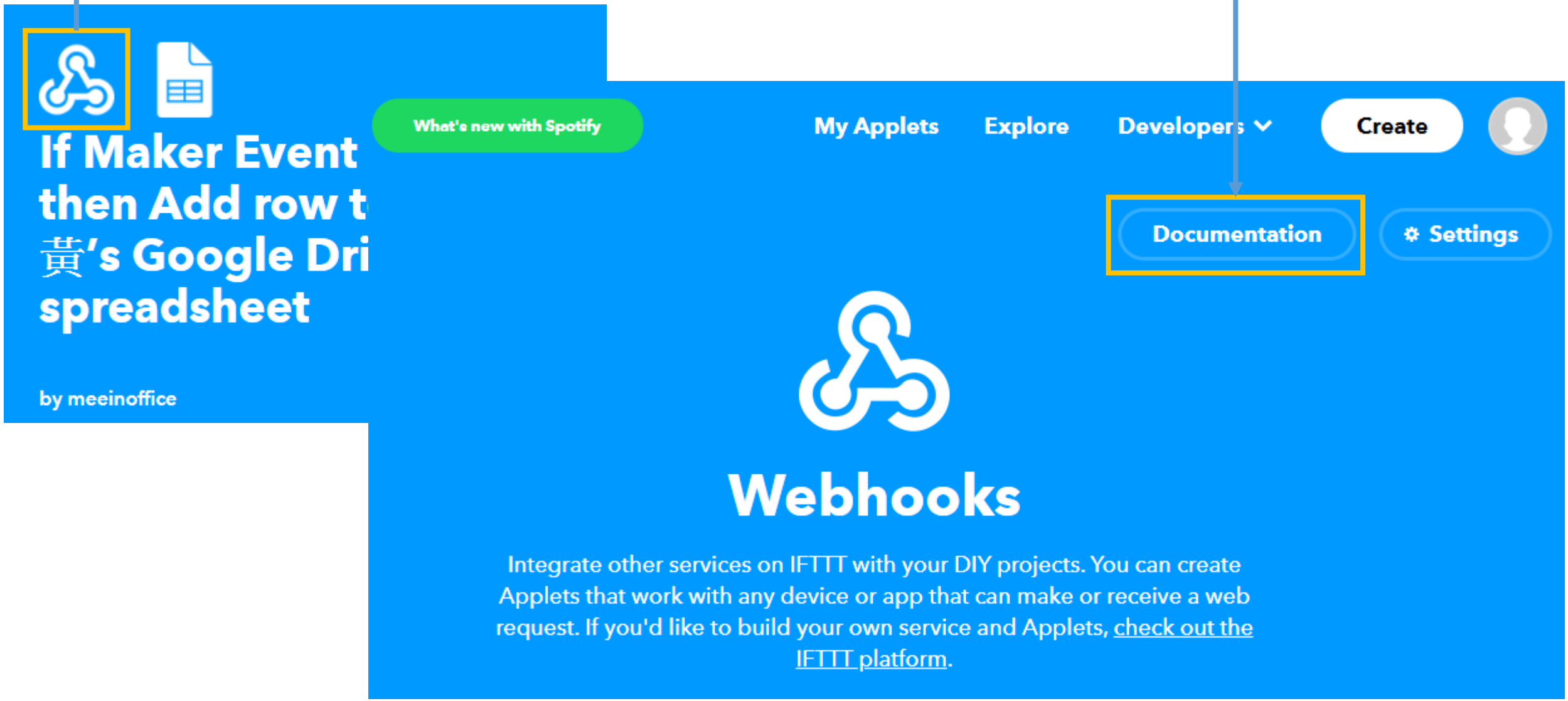
Continue

OccurredAt EventName Value1

	A	B	C
1	March 12, 2021	home	1931a16e

Finish

測試 Applet


If Maker Event
then Add row t
黃's Google Dri
spreadsheet
by meeinoffice
What's new with Spotify
My Applets
Explore
Developers
Create
Documentation
Settings
Webhooks
Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that can make or receive a web request. If you'd like to build your own service and Applets, check out the IFTTT platform.

Your key is: dT[redacted]G

Back to service

這是替代帳密的**金鑰**
等下在程式中會用到

To trigger an Event

Make a POST or GET web request to:

```
https://maker.ifttt.com/trigger/home/with/key/dl[redacted]aG
```

輸入剛剛的**事件名稱**

With an optional JSON body of:

```
{ "value1" : "[redacted]", "value2" : "[redacted]", "value3" : "[redacted]" }
```

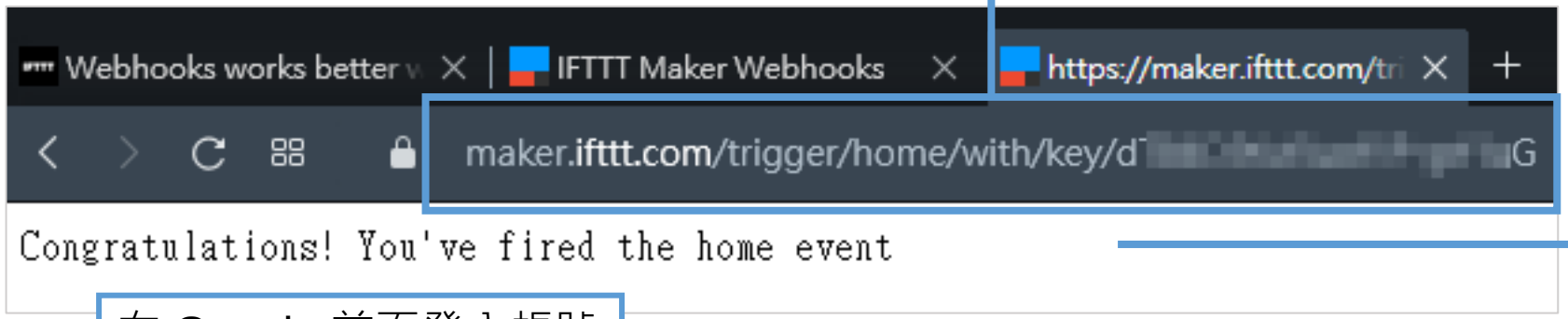
The data is completely optional, and you can also pass `value1`, `value2`, and `value3` as query parameters or form variables. This content will be passed on to the action in your Applet.

You can also try it with `curl` from a command line.

```
curl -X POST https://maker.ifttt.com/trigger/home/with/key/dT[redacted]iG
```

複製這段**觸發事件的網址**
等下可以用瀏覽器做測試

Please read our FAQ on using Webhooks for more info.



貼入剛剛複製的網址

成功觸發事件的回應

在 Google 首頁登入帳號

按一下 Google 應用程式鈕

IFTTT 幫你建立的試算表

Lab08

小孩到家刷卡紀錄系統

實驗目的	讀取悠遊卡與 RFID 卡的卡號, 然後將卡號送到雲端資料庫儲存, 完成一個雲端的刷卡系統。
材料	• D1 mini　• RFID 感測模組　• RGB 三色 LED

設計原理

在 IFTTT 的 HTTP 請求路徑後面可以加入參數名稱與內容，夾帶資訊給 IFTTT, 總共可以夾帶 3 個參數，參數名稱固定為 value1, value2, value3, 之間用 "&" 隔開：

```
https://maker.ifttt.com/trigger/home/with/key/您的
                                        key?value1=A&value2=B&value3=C
```

請求路徑加上 "?" 再加上參數

夾帶參數內容為 value1=A、value2=B、value2=C

這個實驗中，我們將透過 value1 參數夾帶卡號資訊給 IFTTT, IFTTT 收到後，就會將事件發生時間、事件名稱以及卡號新增到試算表。

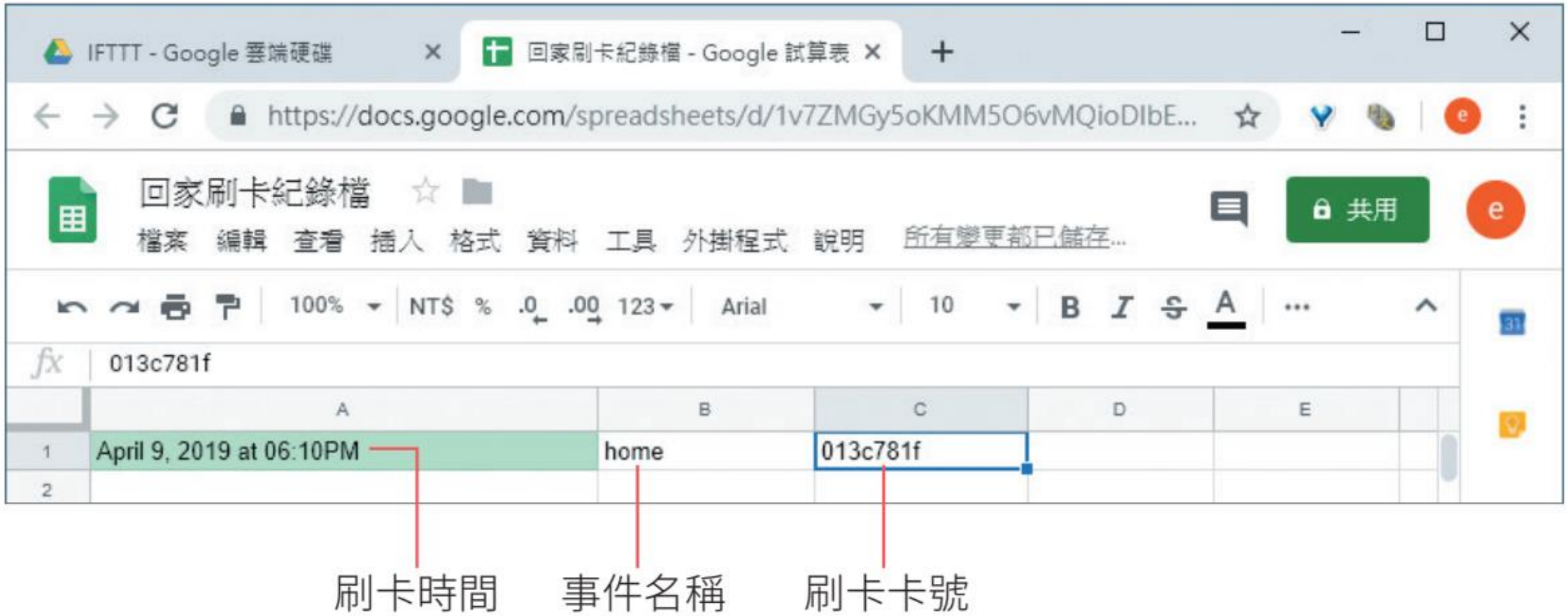
程式設計

```
01 from machine import Pin
02 import mfrc522, network, urequests, time
03
04 # 連線 Wifi 網路
05 sta_if = network.WLAN(network.STA_IF)
06 sta_if.active(True)
07 sta_if.connect("Wifi基地台", "Wifi密碼")
08 while not sta_if.isconnected():
09     pass
10 print("Wifi已連上")
11
12 rfid = mfrc522.MFRC522(0, 2, 4, 5, 14)
13 led = Pin(15, Pin.OUT)
```

```
14
15 while True:
16
17     led.value(0)  # 搜尋卡片之前先關閉 LED
18     stat, tag_type = rfid.request(rfid.REQIDL)  # 搜尋 RFID 卡片
19
20     if stat == rfid.OK:  # 找到卡片
21         stat, raw_uid = rfid.anticoll()  # 讀取 RFID 卡號
22         if stat == rfid.OK:
23             led.value(1)  # 讀到卡號後點亮 LED
24
25             id = "%02x%02x%02x%02x" % (raw_uid[0], raw_uid[1],
26                                        raw_uid[2], raw_uid[3])
27             print("偵測到卡號：", id)
28
29             # 連線 IFTTT 服務以便將卡號傳送到 Google 試算表
30             ifttt_url = "IFTTT的HTTP請求網址"
31             urequests.get(ifttt_url + "?value1=" + id)
32
33             time.sleep(0.5)  # 暫停一下, 避免 LED 太快熄滅看不到
```

實測

請按 F5 執行程式，然後使用悠遊卡或本套件附的 RFID 卡片靠近 RFID 感測模組，當您看到三色 LED 亮藍燈然後熄滅，便表示卡號已經成功上傳，請登入 Google 雲端硬碟 (https://drive.google.com/)，如下開啟刷卡紀錄試算表：


IFTTT - Google 雲端硬碟
回家刷卡紀錄檔 - Google 試算表
https://docs.google.com/spreadsheets/d/1v7ZMGy5oKMM5O6vMQioDIbE...
回家刷卡紀錄檔
檔案 編輯 查看 插入 格式 資料 工具 外掛程式 說明
所有變更都已儲存...
共用
100% NT$ % .0 .00 123 Arial 10
013c781f
A B C D E
1 April 9, 2019 at 06:10PM home 013c781f
2
刷卡時間
事件名稱
刷卡卡號

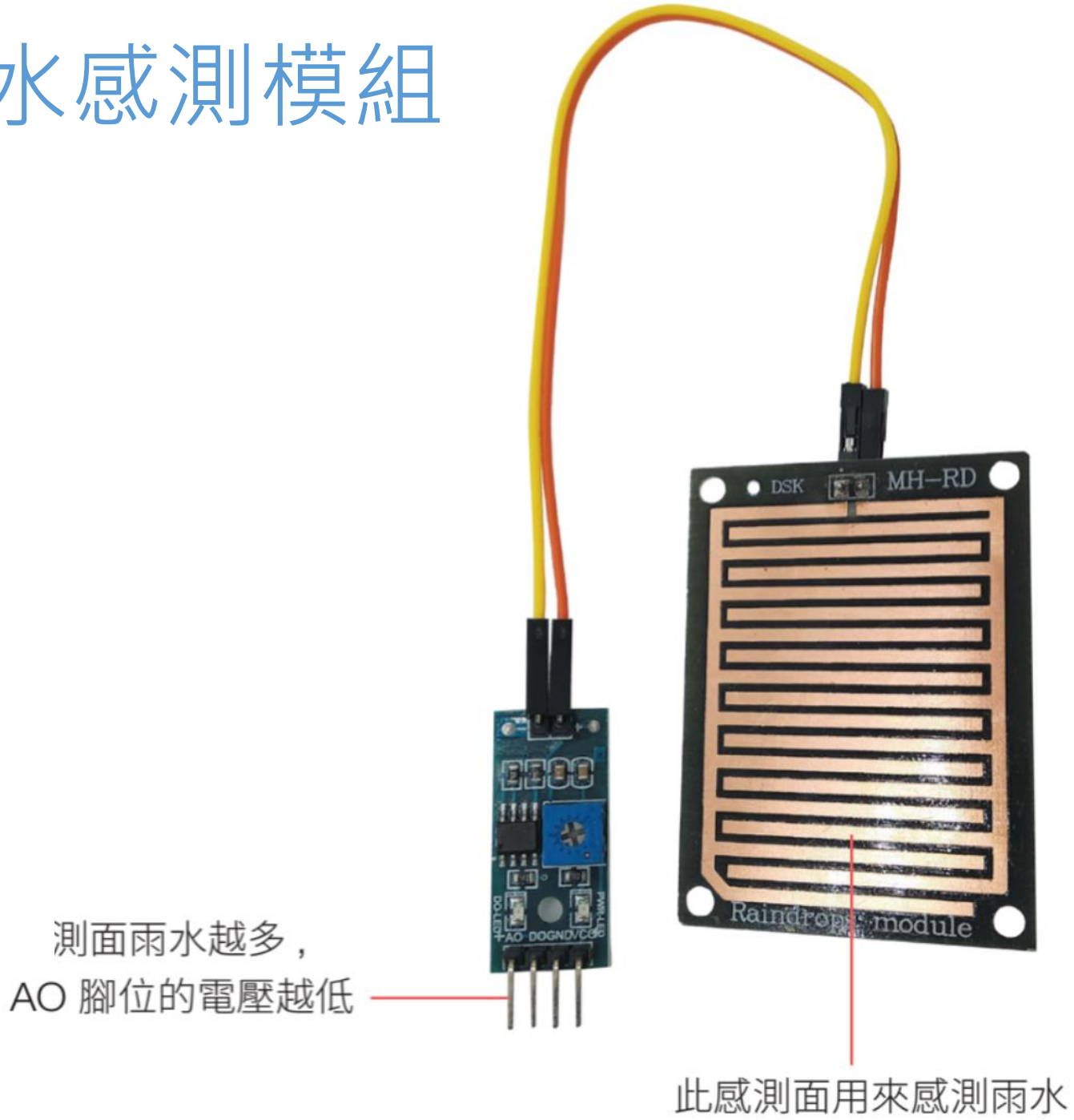
09 雨量感測

類比輸入

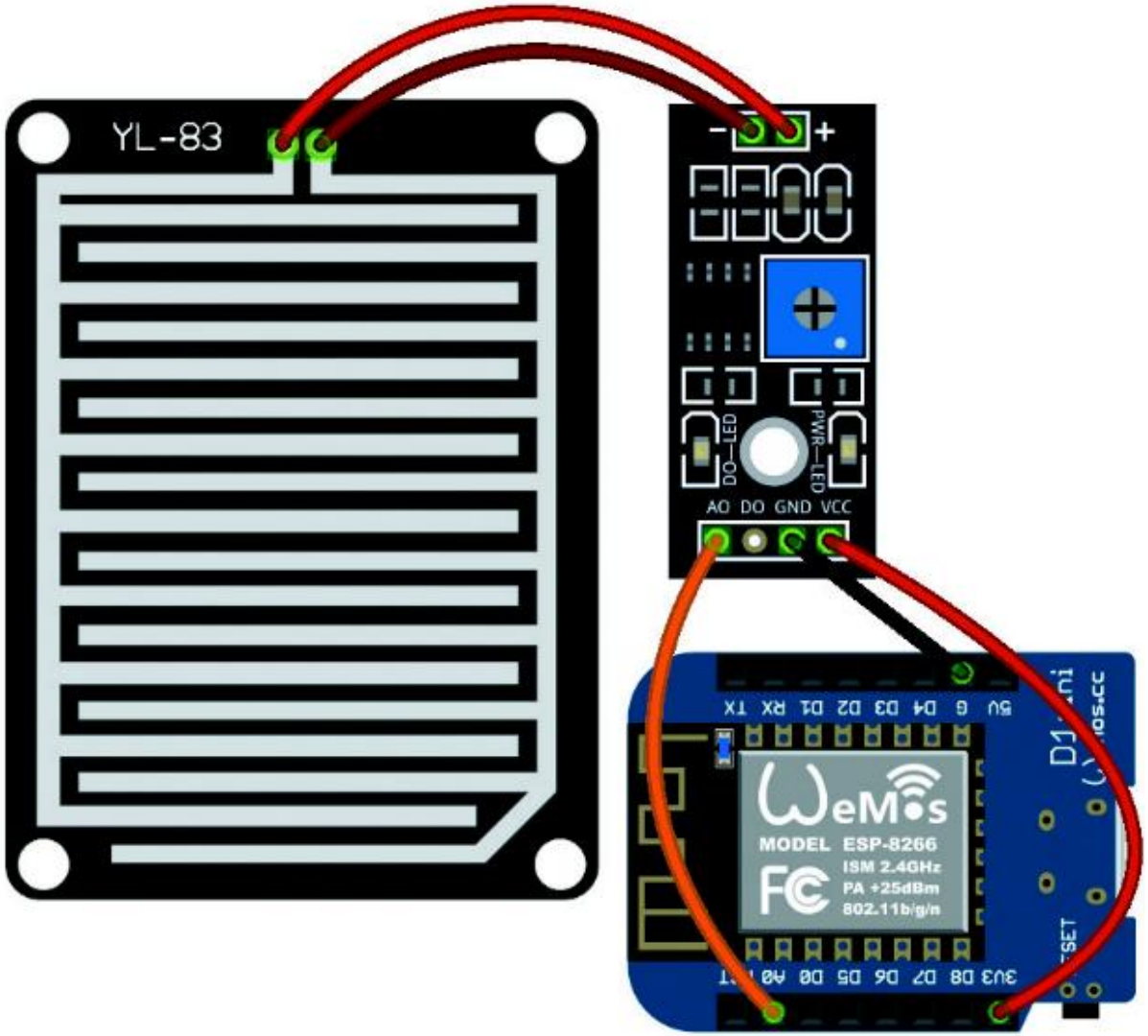
認識雨水感測模組

我們準備使用以下雨水感測模組來製作一個雨量即時統計圖，用來紀錄每天的雨量並且繪製折線圖。

只要偵測雨水感測模組 AO 腳位的電壓變化，便可以獲得目前雨水量的多寡。



請依線路圖接線


YL-83
DO LED
PWR LED
AO DO GND VCC
WeMos
MODEL ESP-8266
ISM 2.4GHz
PA +25dBm
802.11b/g/n

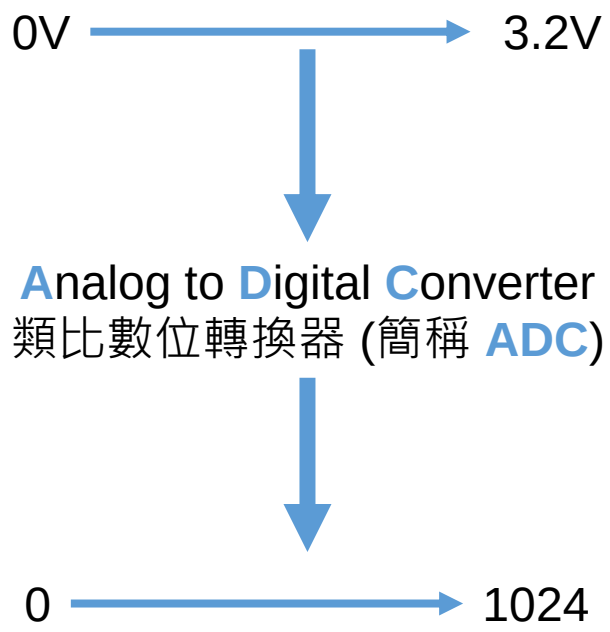
用 ADC 偵測電壓變化

在電子的世界中，訊號只分為高電位跟低電位兩個值，這個稱之為**數位訊號** (0/1、High/Low、或 On/Off), 所以前面章節我們使用 D1 mini 輸出或輸入時，只能輸出 / 輸入高、低電位兩個值。

但電壓變化不是這樣的二分值，而是連續的變化，例如 1V、2.1V 等都是可能的值，這種訊號稱為**類比值**。

為了偵測雨水感測模組 AO 腳位的電壓變化，必須透過 **ADC (Analog-to-Digital Conversion, 類比數位轉換器)**, 將電壓值轉換為電腦可以讀取的數位值。

D1 mini 控制板具備 ADC 的是 A0 腳位，當 A0 腳位有電壓輸入時，ADC 會將 0 ～ 3.2V 電壓範圍轉成 0～1024 再傳給 D1 mini。所以傳回值 1024 就是 3.2V 電壓輸入，341 表示大約 1.1V 電壓輸入。也就是說，將傳回值先除以 1024 再乘上 3.2 就可以換算成電壓。



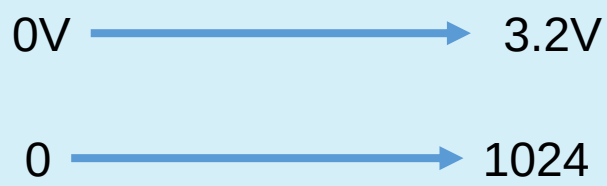
設計原理

請使用以下語法建立 A0 腳位的 ADC 物件：

```
>>> from machine import ADC
>>> adc = ADC(0)
```

然後使用 read() 方法即可讀取 ADC 轉換後的數值，數值越大表示電壓越大：

```
>>> adc.read()
168
>>> adc.read()
666
```



Lab09

讀取雨水感測模組的值

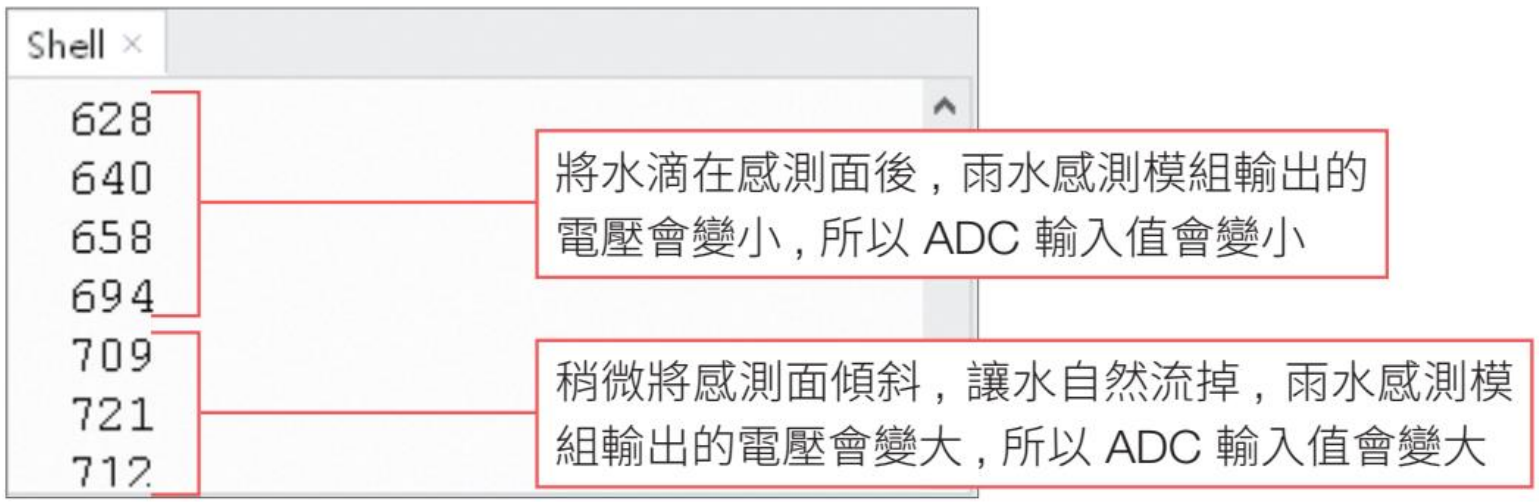
實驗目的	用程式讀取雨水感測器的電壓。
材料	• D1 mini　　• 雨水感測模組

程式設計

```
01 from machine import ADC
02 import time
03
04 # 建立 A0 腳位的 ADC 物件, 並命名為 adc
05 adc = ADC(0)
06
07 while True:
08     # 用 read() 方法從 A0 號腳位讀取 ADC 轉換後的數值
09     # 然後將讀到的值用 print() 輸出
10     print(adc.read())
11
12     # 暫停 0.05 秒
13     time.sleep(0.05)
```

實測

請按 F5 執行程式，然後以少許水滴在雨水感測模組的金屬感測面（請小心！不要讓水接觸到感測面以外的電子零件），在 Thonny 的 Shell 窗格觀察程式輸出的值：



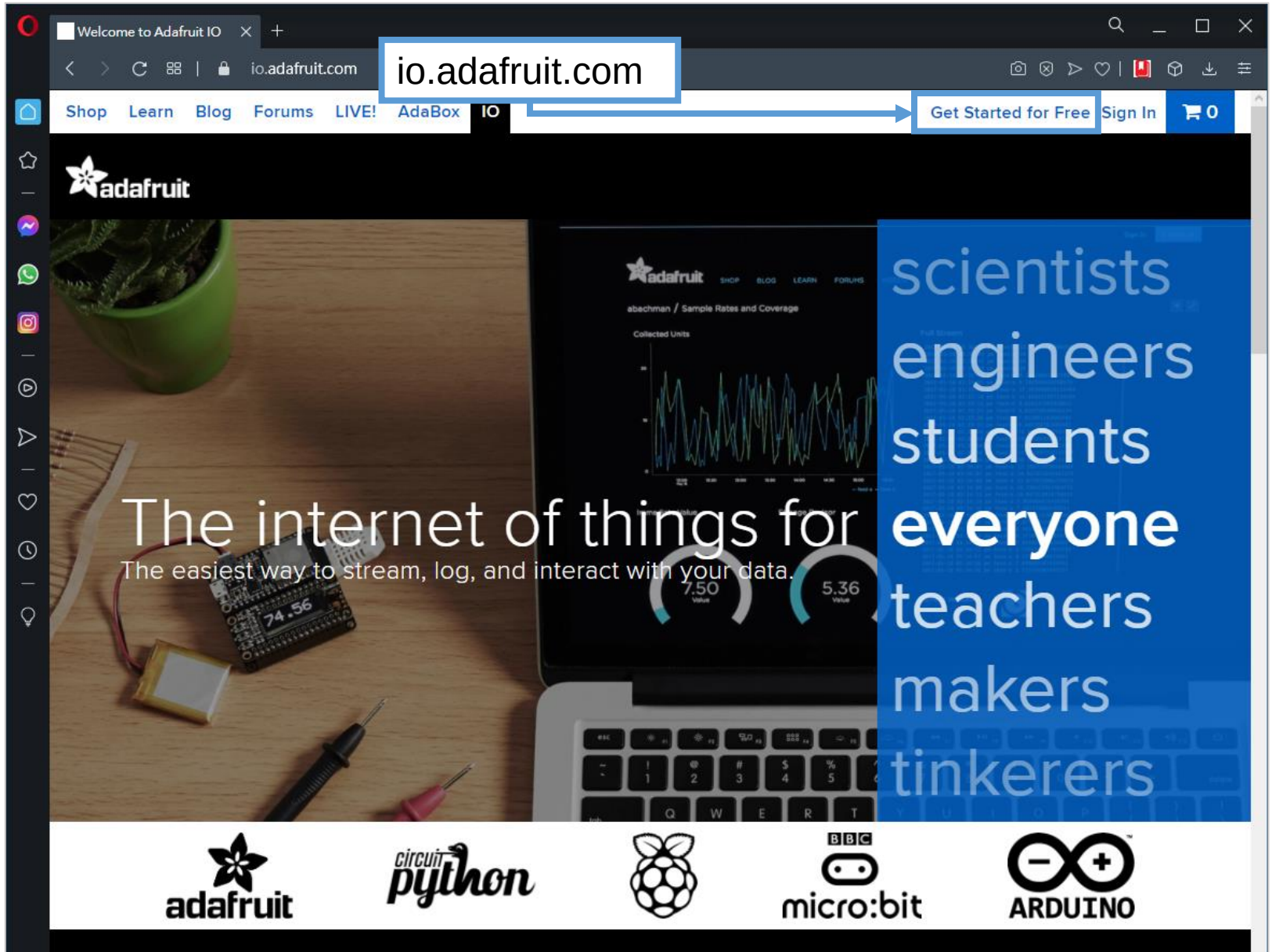
經過實測後，我們發現雨水感測模組的感測面有水時，ADC 輸入值會小於 700, 若水自然流掉後 ADC 輸入值會大於 700, 所以接下來我們會用 700 這個數值來判斷是否有下雨。

⚠ 您可以依照自己實測的結果來挑選適當的數值。

10 繪製即時雨量圖

使用 Adafruit IO 服務

註冊 Adafruit IO 帳號



FIRST NAME

LAST NAME

EMAIL

USERNAME

Username is viewable to the public on the forums, Adafruit IO, and elsewhere.

PASSWORD

CREATE ACCOUNT

HAVE AN ADAFRUIT ACCOUNT?

SIGN IN

We think you are in the Taipei time zone, if this isn't correct, please choose the correct time zone below.

FIRST NAME

大偉

LAST NAME

黃

EMAIL

meein.office@gmail.com

USERNAME

flagmee

GRAVATAR EMAIL

FAVORITE COLOR

TIME ZONE

(GMT+08:00) Taipei

SAVE SETTINGS

Confirm password to continue

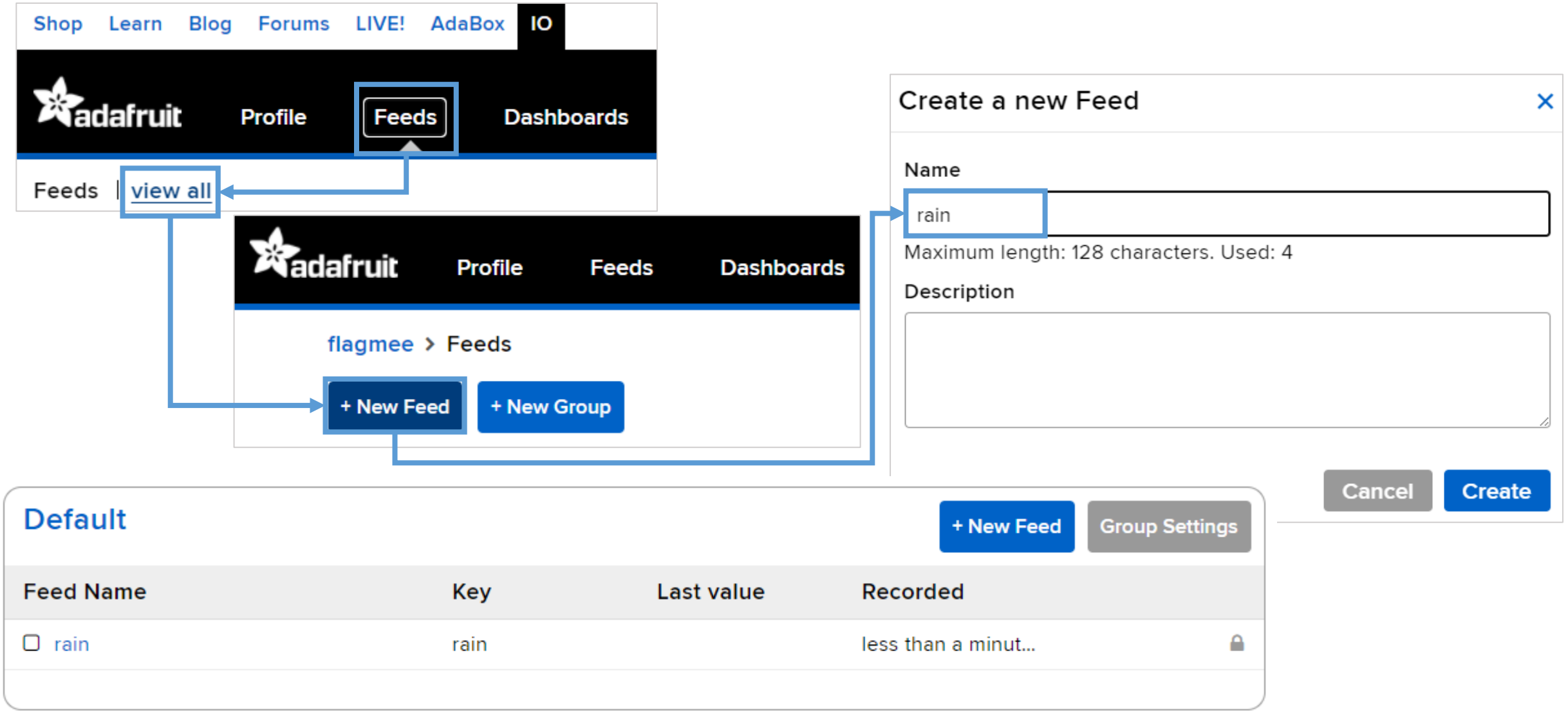
In order to update this account, please confirm your password.

PASSWORD Forget your password?

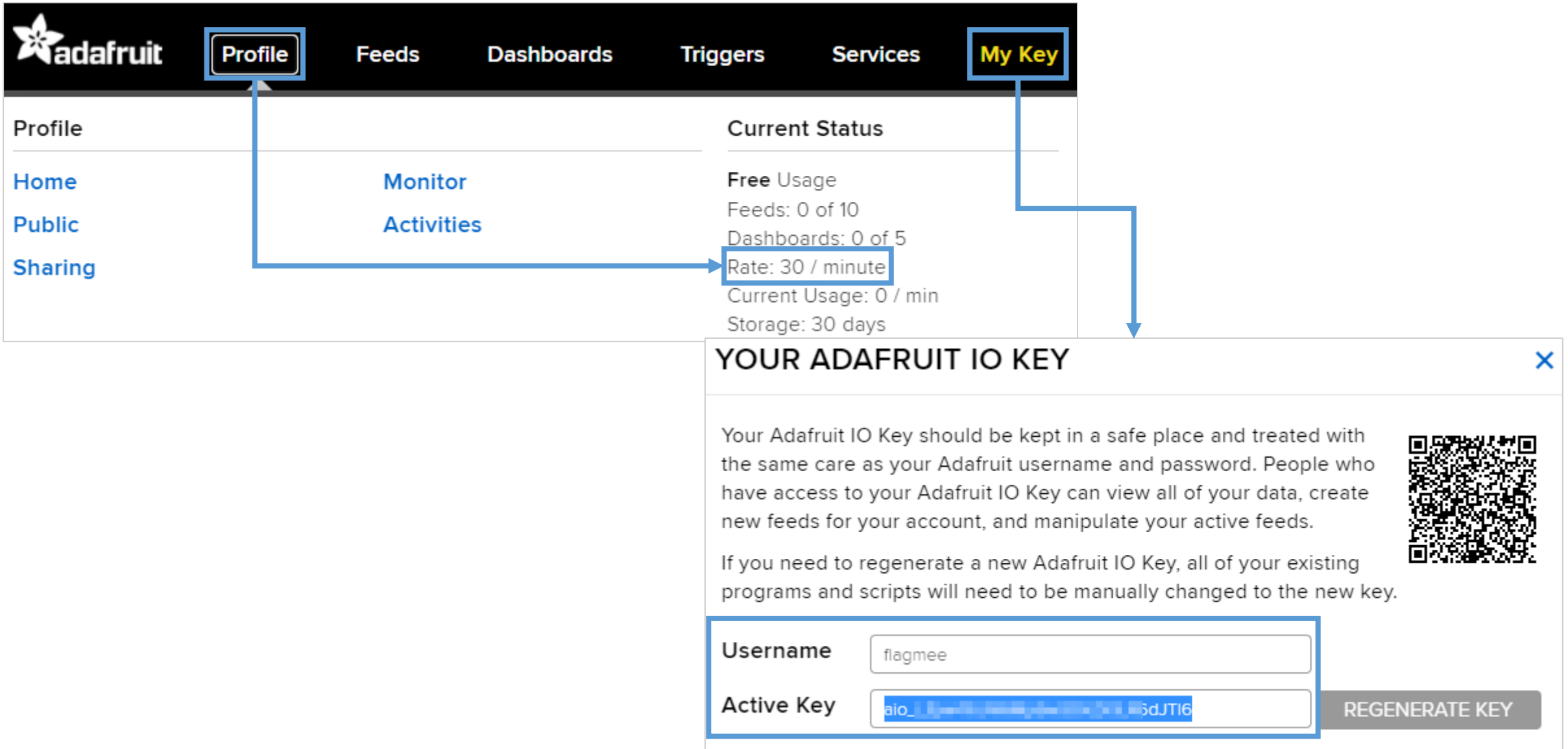
••••••••••

CONFIRM PASSWORD

建立 Feed(資料來源)


Shop
Learn
Blog
Forums
LIVE!
AdaBox
IO
adafruit
Profile
Feeds
Dashboards
Feeds
view all
adafruit
Profile
Feeds
Dashboards
flagmee › Feeds
+ New Feed
+ New Group
Create a new Feed
Name
rain
Maximum length: 128 characters. Used: 4
Description
Cancel
Create
Default
+ New Feed
Group Settings
Feed Name
Key
Last value
Recorded
rain
rain
less than a minut...

查看使用限制以及帳號、金鑰



Lab10

雨量即時統計圖

實驗目的	讀取雨水感測模組的感測資料, 將雨量的大小數值上傳到 Adafruit IO, 產生雨量即時統計圖。
材料	• D1 mini　　• 雨水感測模組

設計原理

第 2 章介紹過使用 urequests 模組的 get() 來連線 HTTP 服務，get() 使用的是 HTTP 協定中的 GET 方法。不過 Adafruit IO 規定上傳資料要使用 HTTP 協定的 POST 方法，所以我們將改用 urequests 模組的 post() 來上傳資料。

⚠ 因篇幅所限，本書無法詳述 HTTP 與 HTML，請您自行參考相關書籍。

用 POST 方法上傳的資料目前主要有兩種格式：FORM 與 JSON，我們將採用第 3 章介紹過的 JSON 格式來上傳資料，其語法如下：

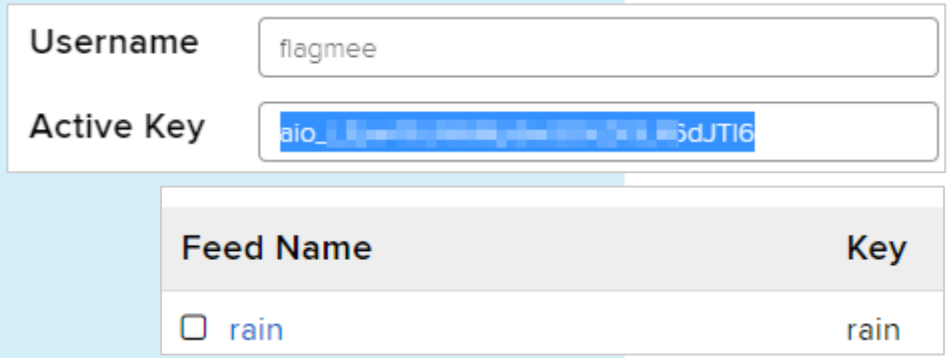
```
>>> import urequests
>>> data = {"value": 100}
>>> urequests.post("http://xxx.xxx", json=data)
```

以 Python 字典來設定要上傳的資料名稱與資料值

只要將 Python 字典帶入 urequests.post() 的 json 參數，urequests 就會自動將字典內的資料轉換為 JSON 格式來上傳。

程式設計

```
01 from machine import ADC
02 import time, network, urequests
03
04 # 連線 Wifi 網路
05 sta_if = network.WLAN(network.STA_IF)
06 sta_if.active(True)
07 sta_if.connect("Wifi基地台", "Wifi密碼")
08 while not sta_if.isconnected():
09     pass
10 print("Wifi已連上")
11
12 aio_username = "您的 Adafruit IO 帳號"
13 aio_key = "您的 Adafruit IO 金鑰"
14 aio_feed = "您的 Adafruit IO feed 名稱"
15
16 # 建立 A0 腳位的 ADC 物件, 並命名為 adc
17 adc = ADC(0)
```

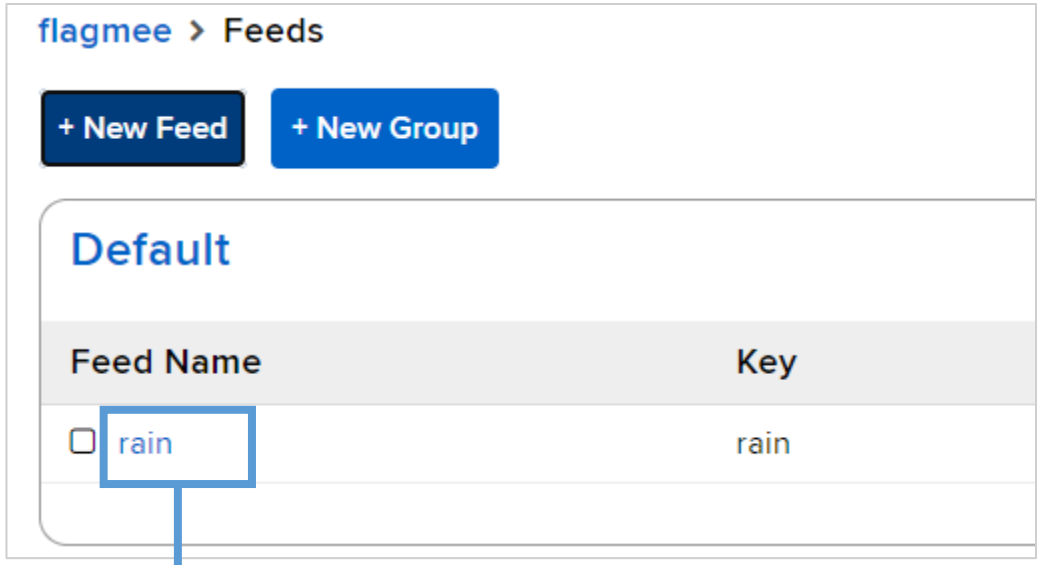


```
18
19 while True:
20     # 讀取雨水感測器經過 ADC 轉換後的數值
21     value = adc.read()
22
23     if value < 700: # 依照 Lab09 的測試, 低於 700 表示有下雨
24         # 雨水越多, ADC 值越低, 所以用最大值 1024 減 ADC 值,
25         # 以便將資料反轉為雨水越多, 數值越高
26         data = {"value": 1024-value}
27     else:
28         # 沒下雨的話就送出 0
29         data = {"value": 0}
30
31     # 設定 Adafruit IO 上傳資料的 API 網址
32     url = ("https://io.adafruit.com/api/v2/" + aio_username +
33            "/feeds/" + aio_feed + "/data?X-AIO-Key=" + aio_key)
34
35     # 用 POST 上傳 JSON 資料
36     urequests.post(url, json=data)
```

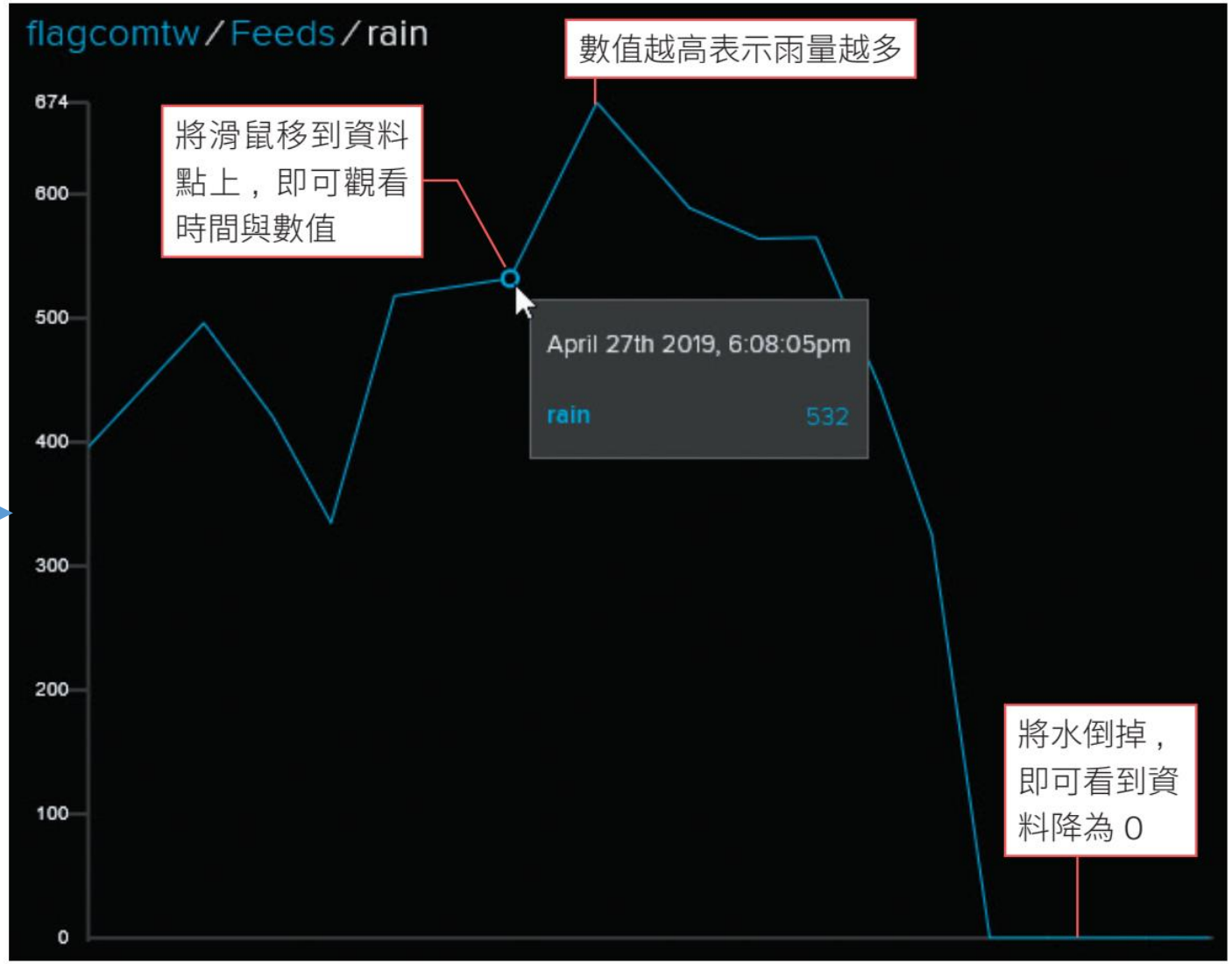
```
37
38      # 暫停 2 秒, 避免送出太多資料超過 Adafruit IO 免費額度
39      time.sleep(2)
```

- 請依照您的環境修改程式中的『Wifi 基地台』、『Wifi 密碼』等設定
- aio_username、aio_key、aio_feed 等三個變數，請改成您在 5-3 節取得的 Adafruit IO 帳號、金鑰，以及新建立的 Feed 名稱。
- 第 23 行的 700 請依照 Lab09 的實測結果來挑選適當的數值，此值若設定過低可能會導致無法偵測毛毛雨，如果設定過高則無法偵測雨停。

實測



請按 F5 執行程式，然後以少許水滴在雨水感測模組的金屬感測面（請小心！不要讓水接觸到感測面以外的電子零件），然後到 Adafruit IO 觀看 Feed 的資料：



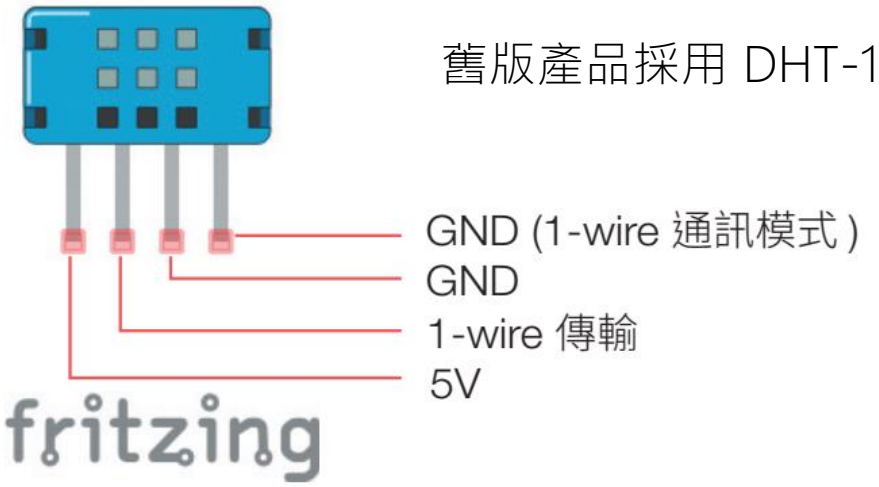
11 溫度感測

DHT11 感測模組

認識溫濕度感測器

每種溫濕度感測器可偵測的溫濕度範圍皆不相同，本套件採用的 DHT12 感測器可以偵測 -20 ° C ~ 60 ° C 的溫度，以及 20~95% 相對濕度。

DHT12 溫濕度感測器支援 1-wire 與 I2C 兩種通訊，因為 MicroPython 已經內建 1-wire 通訊的 dht 模組，所以為了方便撰寫程式，我們將直接使用 MicroPython dht 模組以 1-wire 通訊來取得溫濕度的值。

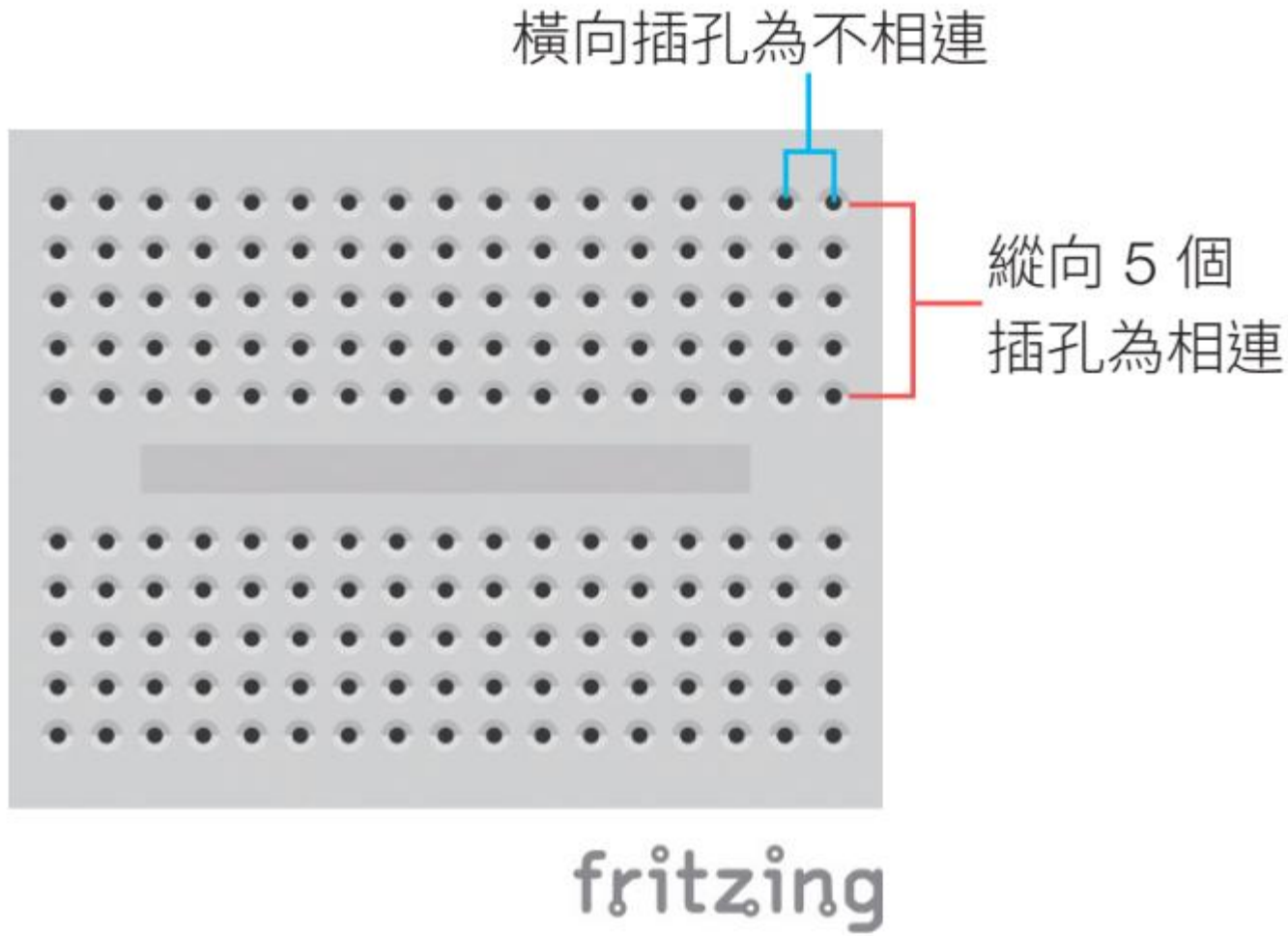


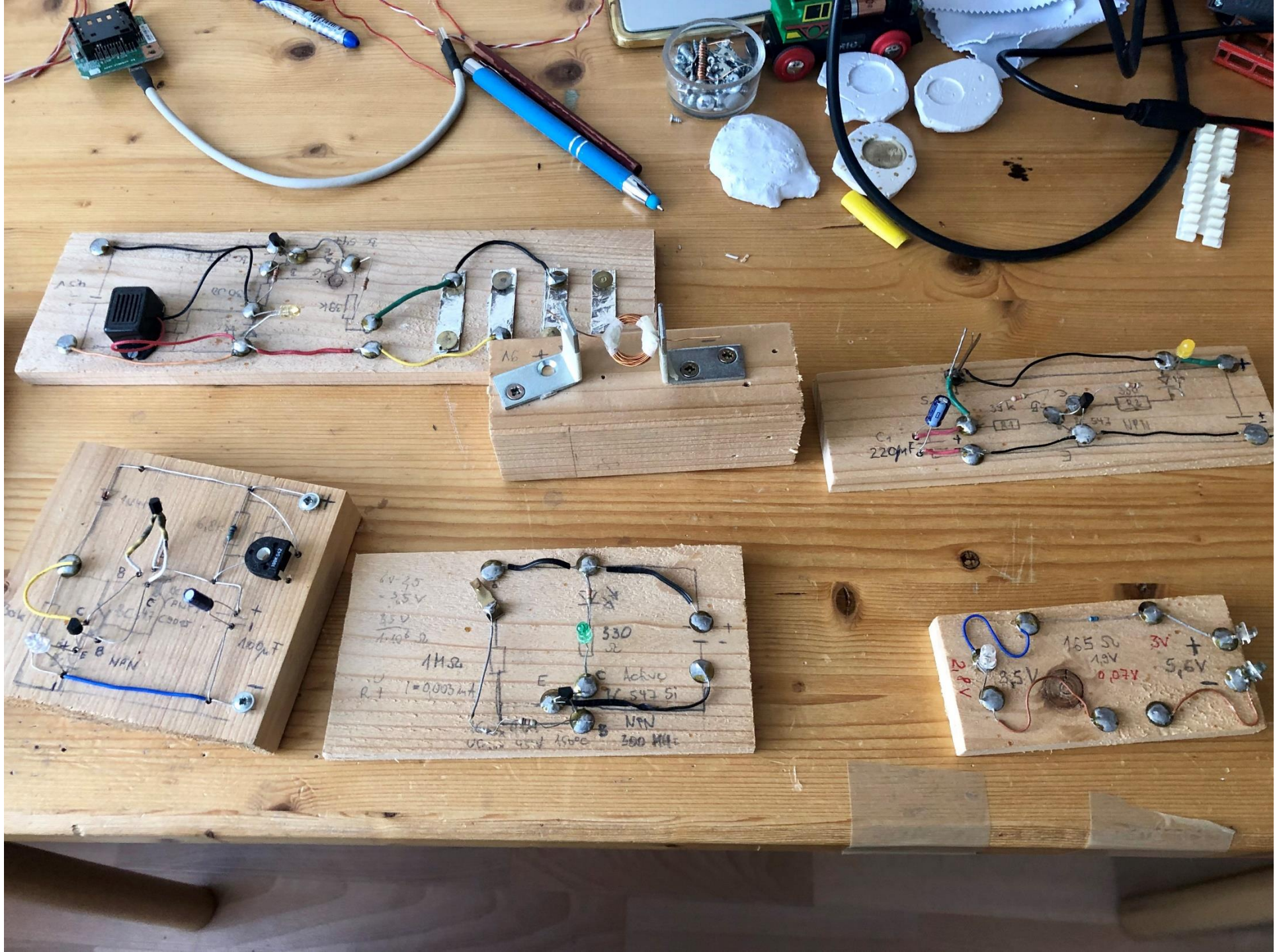
舊版產品採用 DHT-12

新版產品採用 DHT-11

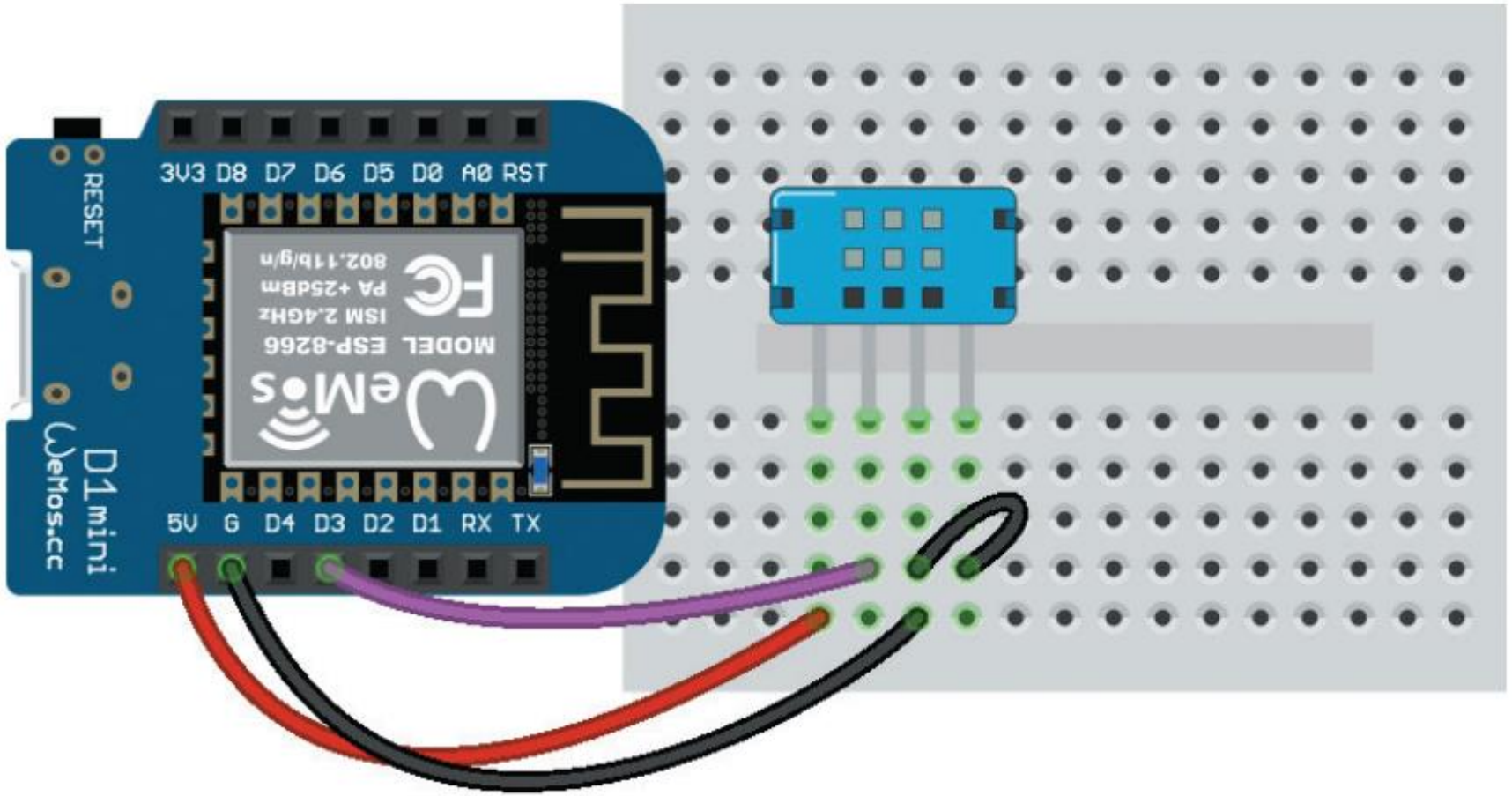
麵包板

麵包板的表面有很多的插孔。插孔下方有相連的金屬夾，當零件的接腳插入麵包板時，實際上是插入金屬夾，進而和同一條金屬夾上的其他插孔上的零件接通，在本套件實驗中我們就需要麵包板來連接 D1 Mini 與感測器模組。



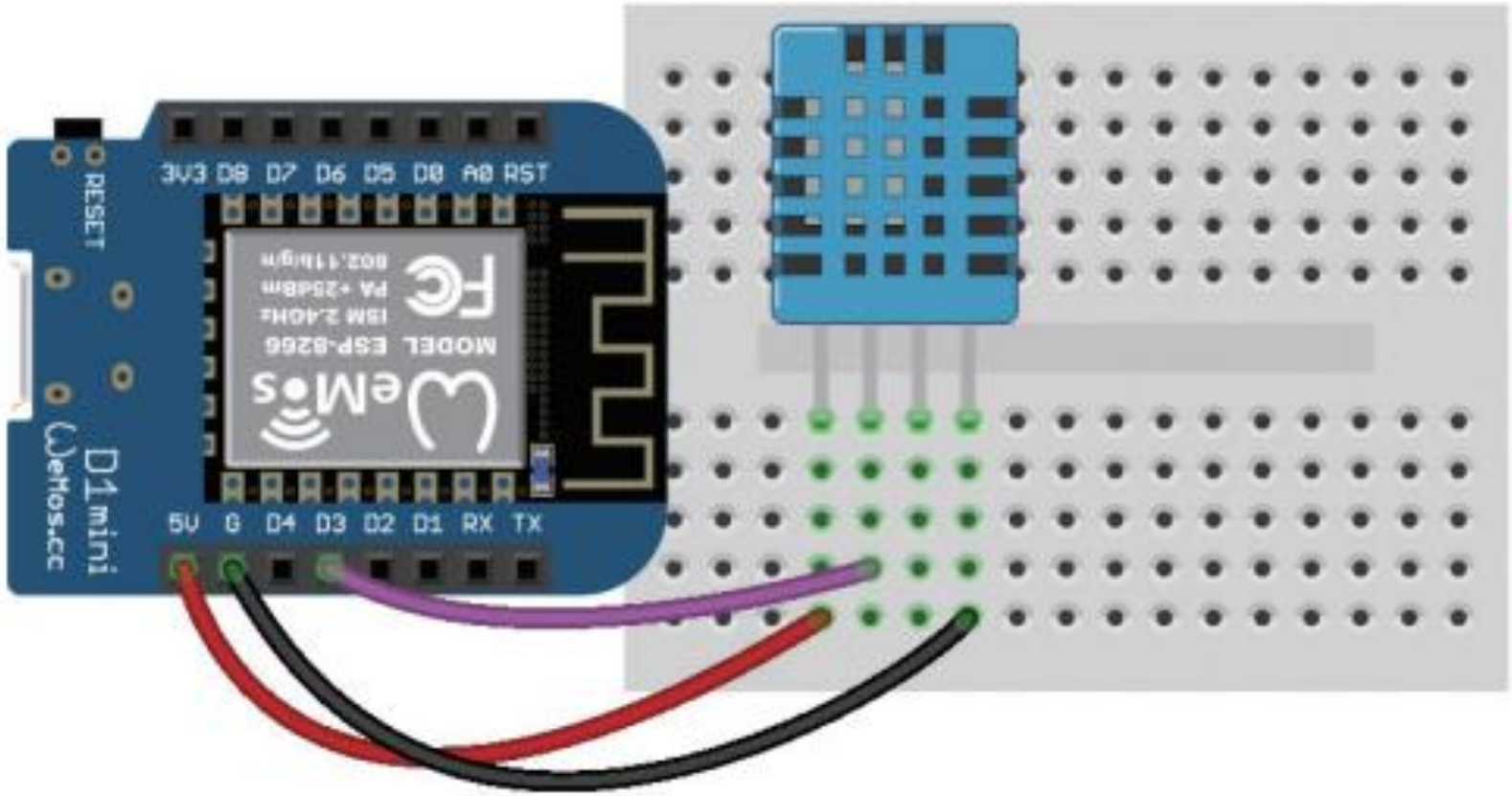

330
1MΩ
NPN
165 Ω
3V
5,6V
3,5V
220µF

請依線路圖接線



⚠ DHT12 感測器兩面不同，一面有格洞、另一面為文字，請勿接錯面，否則**感測器會燒毀**。

請依線路圖接線



⚠ DHT11 感測器兩面不同，一面有格洞、另一面為文字，請勿接錯面，否則**感測器會燒毀**。

Lab11

讀取溫濕度感測值

實驗目的	讀取 DHT12 溫濕度感測器的值, 熟悉感測器的用法。
材料	• D1 mini • DHT12 溫濕度模組 • 杜邦線及排針若干

dht 模組

MicroPython 已經內建 dht 模組，只要直接匯入模組就可以使用：

```
from machine import Pin
import dht
```

由於 dht 模組需要指定實際接線的腳位，通常也會同時匯入 machine 模組使用其中的 Pin 類別。接著必須建立與 dht 模組溝通的物件：

```
sensor = dht.DHT11(Pin(0))
```

建立時唯一需要的參數就是腳位，1-wire 通訊會使用同一個腳位輸出 / 輸入資料，dht 模組會自行切換模式，建立物件時並不需要指定輸出入模式。

建立物件後，只要呼叫它的 measure 方法，就可以讓 DHT12 感測器送出溫濕度資料，我們可以透過 temperature 及 humidity 方法取得溫度與濕度。

要注意的是，由於 DHT12 感測器本身的運作方式限制，至少要**相隔 2 秒**才能再次呼叫 measure, 否則 DHT12 感測器可能無法回應，造成 dht 模組等待資料逾時而發生錯誤。

程式設計

```
01 from machine import Pin
02 import time
03 import dht
04
05 sensor = dht.DHT11(Pin(0))          # 使用 D3 腳位取得溫溼度物件
06 while True:
07     sensor.measure()                # 讀取溫濕度值
08     temp_humi = "%2d ℃/%2d%%" % (# 格式化字串
09         sensor.temperature(),       # 置入溫度值
10         sensor.humidity())          # 置入濕度值
11     print(temp_humi)                # 顯示溫濕度值
12     time.sleep(3)                   # 暫停 3 秒
```

程式執行後，每隔 3 秒就會看到新的溫濕度值：

```
>>> %Run Lab11.py
28 ℃/55%
28 ℃/55%
28 ℃/60%
28 ℃/60%
28 ℃/60%
28 ℃/60%
28 ℃/61%
28 ℃/68%
29 ℃/72%
```

雲端溫濕度計

MQTT 通訊協定

物聯網裝置傳輸資料的難題

除了讀取感測值顯示在螢幕上以外，我們希望可以讓溫濕度感測值透過網路傳送到你的手機上顯示，甚至可以從手機上送出指令遙控控制板，這需要控制板與手機之間能夠**雙向傳輸**。要做到以上所述，最大的難題是要**突破網路的安全限制**，舉例來說，D1 mini 控制板可以藉由 Wi-Fi 網路連上外部的伺服器，像是前面的章節就連接到 IFTTT 等服務，但如果要從外部的裝置主動連線到位於 Wi-Fi 網路內部的 D1 mini 控制板，就會被 Wi-Fi 寬頻分享器或路由器隔絕而無法連線。

使用 MQTT 突破網路限制

為了在這樣的限制上仍然能夠提供雙向傳輸，就有了像是 MQTT 這樣的中介服務，它由 3 個元件組成：分別是負責轉送資料的 **MQTT 中介伺服器 (broker)**，提供資料的**發佈端 (publisher)**，以及接收資料的**訂閱端 (subscriber)**。運作方式如下：

發佈端會將資料送到 MQTT 中介伺服器上，MQTT 中介伺服器就會轉送給訂閱端。這裡的關鍵就是不管是發佈端或是訂閱端，都是主動連線到位於公開網路上的中介伺服器，因此只要能夠連網，雙方就可以透過中介伺服器傳輸資料。由於這樣的架構，所以不論是發佈端或是訂閱端，都通稱為『**MQTT 用戶端 (client)**』。

個別的裝置可以同時是發佈端與訂閱端，既能發送資料給遠端的裝置，也能接收遠端裝置送出的資料。在 MQTT 中，資料還必須分門別類，區分為不同的『**頻道 (channel)**』，發佈資料時必須指定頻道，訂閱端也必須先**訂閱**頻道，才能收到發佈到該頻道上的資料。

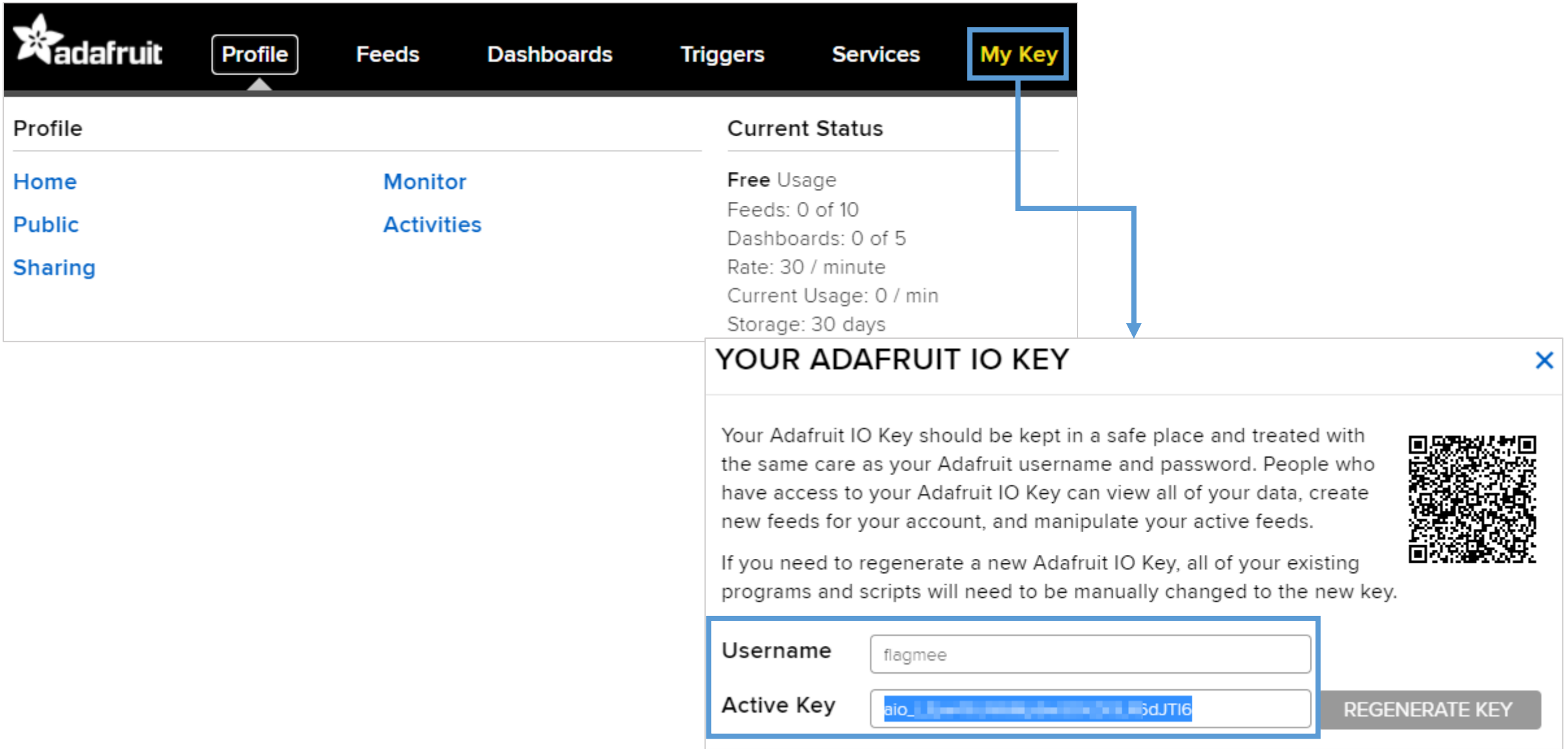
Adafruit IO 中介伺服器

在第 5 章中使用過的 Adafruit IO 也提供有 MQTT 中介伺服器的功能，相關資料如下：

Adafruit IO MQTT 中介伺服器

主機網址	io.adafruit.com
連接埠編號	1883
使用者帳號	你註冊的 Adafruit IO 帳號名稱
密碼	在 Adafruit IO 儀表板頁面按 View AIO Key 取得的金鑰

請特別注意 MQTT 連線的密碼並不是你登入的密碼，必須經由儀表板上的連結取得：



umqtt 模組

有了 MQTT 中介伺服器後，D1 mini 和手機就可以扮演 MQTT 用戶端的角色相互傳送訊息了。在 MicroPython 中內建有 umqtt 模組，提供 MQTT 用戶端的功能，使用時必須先匯入其中的 MQTTClient 類別：

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

接著建立物件：

```
client = MQTTClient(
    client_id="weather",            # 用戶端識別名稱
    server="io.adafruit.com",       # 中介伺服器網址
    user="帳戶名稱",                 # 帳戶名稱
    password="密碼",                 # 剛剛查詢到的金鑰
    ssl=False)                      # 是否
```

其中 client_id 是用戶端識別名稱，使用同一帳號連上伺服器的個別裝置都要指定不一樣的名稱。

建立物件後就可以呼叫 connect 方法連上 MQTT 中介伺服器，接著再呼叫 publish 就可以送出資料到指定的頻道：

```
client.connect() # 連上伺服器
...
temp_humi = "23 ℃/23%"
client.publish(
        b"帳戶名稱/feeds/temp_humi",    # 頻道名稱
        temp_humi.encode())             # 資料內容
```

第 1 個參數是頻道名稱，可以使用 "/" 切割層級，設計成樹狀結構的頻道架構，例如上例中就是 3 層的頻道結構。不過 Adafruit IO 對於頻道名稱有特別的規定，第 1 層一定是帳號名稱，第 2 層固定是 "feeds"，第 3 層是自訂的名稱。第 2 個參數是要傳送到該頻道的內容。

上述 2 個參數的資料型別都是 **bytes 物件**，你可以在字串的引號前面加上 **'b'** 或是呼叫字串的 **encode 方法**將字串轉換成 bytes 物件。

Lab12

雲端溫濕度監測

實驗目的	利用 MQTT 將 DHT12 溫濕度感測器傳送到手機上顯示。
材料	• D1 mini • DHT12 溫濕度模組 • 杜邦線及排針若干 • Android 手機

程式設計

```
01 from machine import Pin
02 import time
03 import network
04 from umqtt.robust import MQTTClient
05 import dht
06
07 sensor = dht.DHT11(Pin(0)) # 使用 D3 腳位取得溫溼度物件
08
09 client = MQTTClient(                  # 建立物件
10     client_id="weather",               # 用戶端識別名稱
11     server="io.adafruit.com",          # 主機網址
12     user="帳戶名稱",                    # 請填入帳戶名稱
13     password="填入你的金鑰",            # 請填入你的金鑰
14     ssl=False)
15
16 sta_if = network.WLAN(network.STA_IF)     # 取得無線網路介面
17 sta_if.active(True)                       # 啟用無線網路
18 sta_if.connect('無線網路名稱', '密碼')      # 連結無線網路
19 while not sta_if.isconnected():           # 等待無線網路連上
20     pass
```

```
21
22 print("connected")
23
24 client.connect()                          # 連上中介伺服器
25 while True:
26     sensor.measure()
27     temp_humi = "%2d ℃/%2d%%" % (
28         sensor.temperature(),
29         sensor.humidity())
30     client.publish(                        # 發佈溫濕度資料
31         b"帳戶名稱/feeds/temp_humi",
32         temp_humi.encode())
33     time.sleep(3)
```

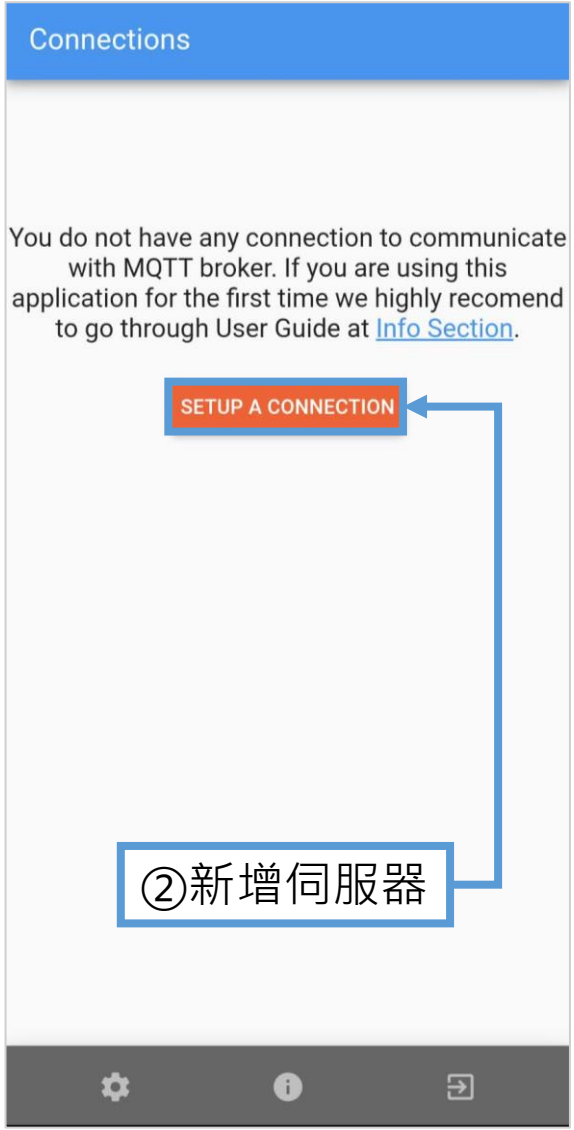
請記得將第 12、31 列中的『帳戶名稱』更改為您自己的帳戶名稱，並且在第 13 列填入您的金鑰，並將第 18 列的無線網路名稱及密碼改寫成您自己的無線網路。

⚠ Adafruit IO 有限制發送資料的頻率，免費帳號最多每 2 秒只能送出一筆資料，實測上最好間隔久一點，像是本例就每隔 3 秒才傳送一次。

手機端 App-Android

AIO 金鑰請自行用 gmail 或 line 等工具寄到你的手機上

①安裝此 App



②新增伺服器

③自訂名稱

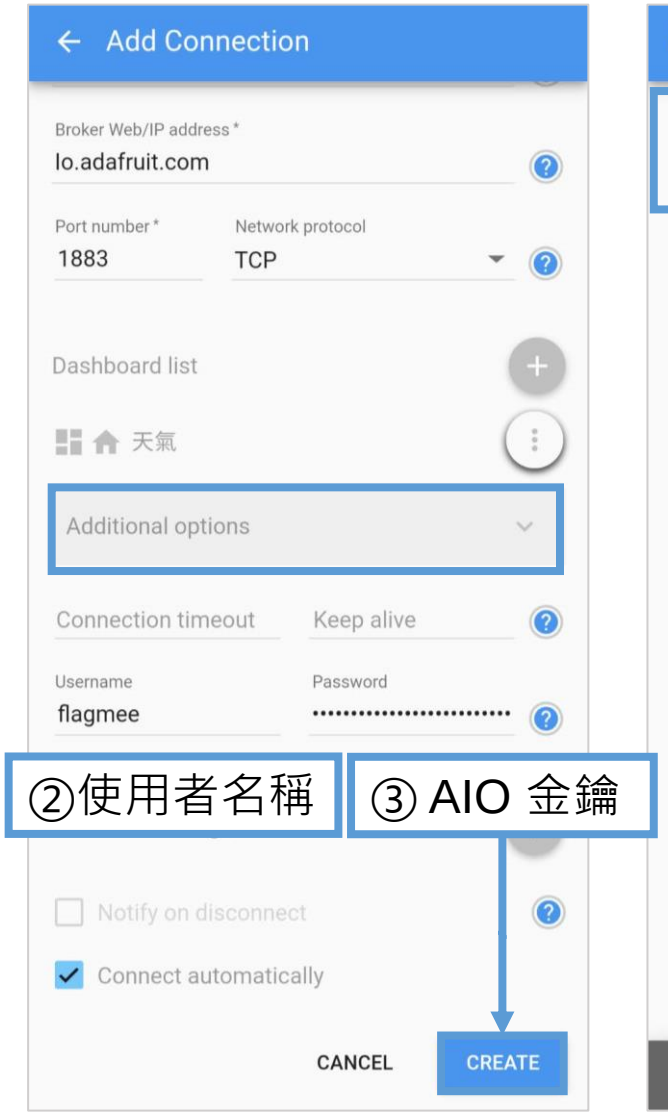
④伺服器位址

⑤選用 TCP

⑥新增資訊儀表板

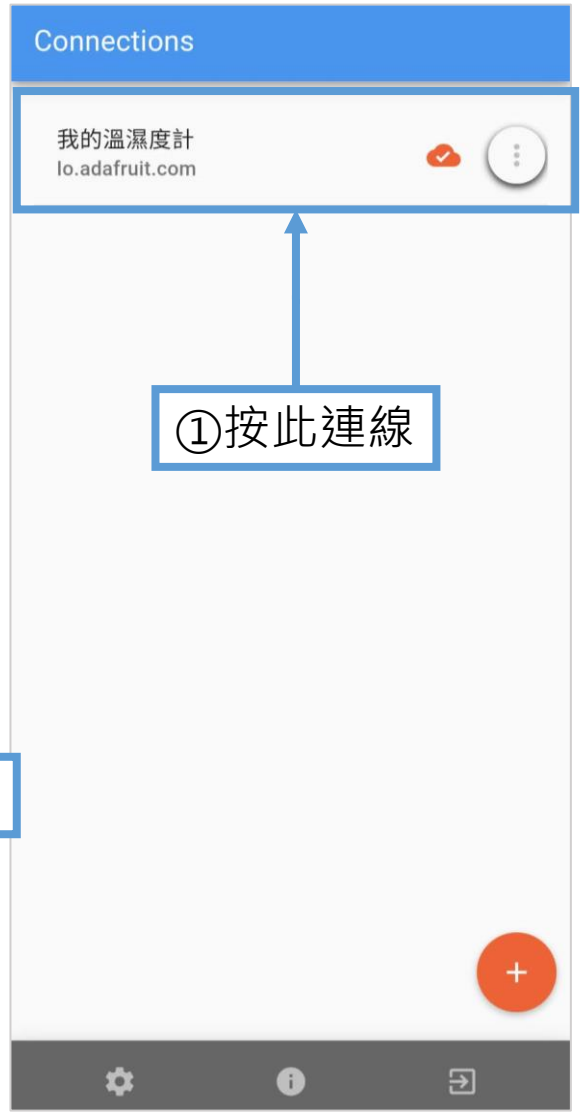
①儀表板名稱

AIO 金鑰請自行用 gmail 或 line 等工具寄到你的手機上

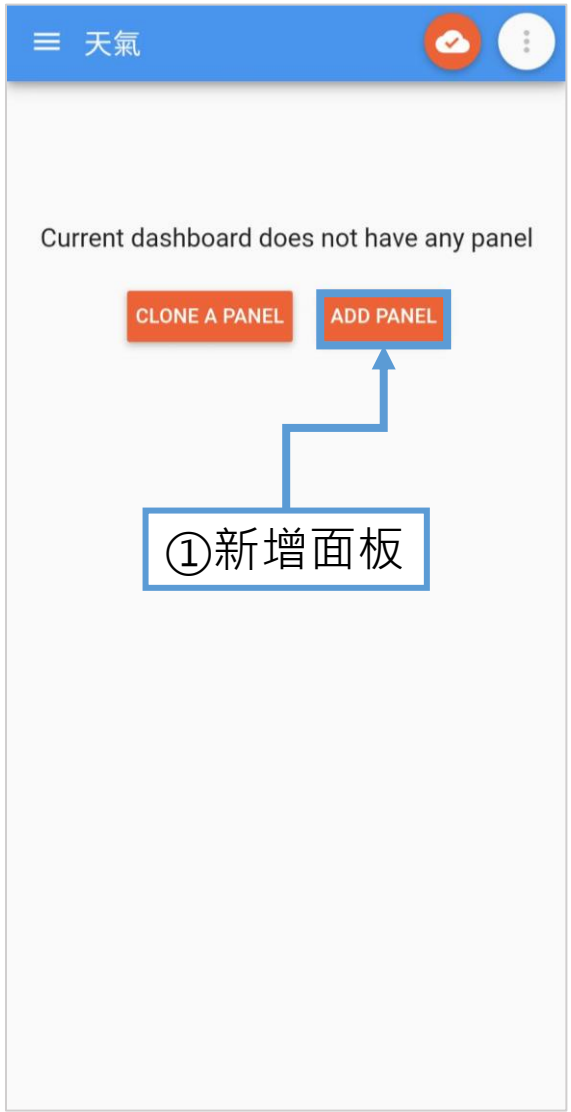


②使用者名稱

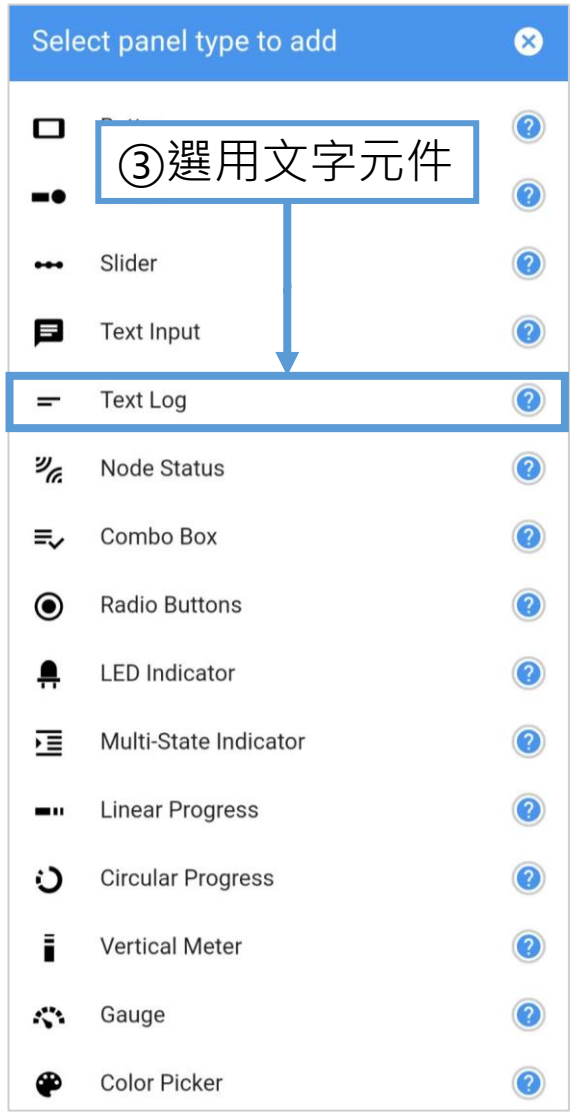
③ AIO 金鑰



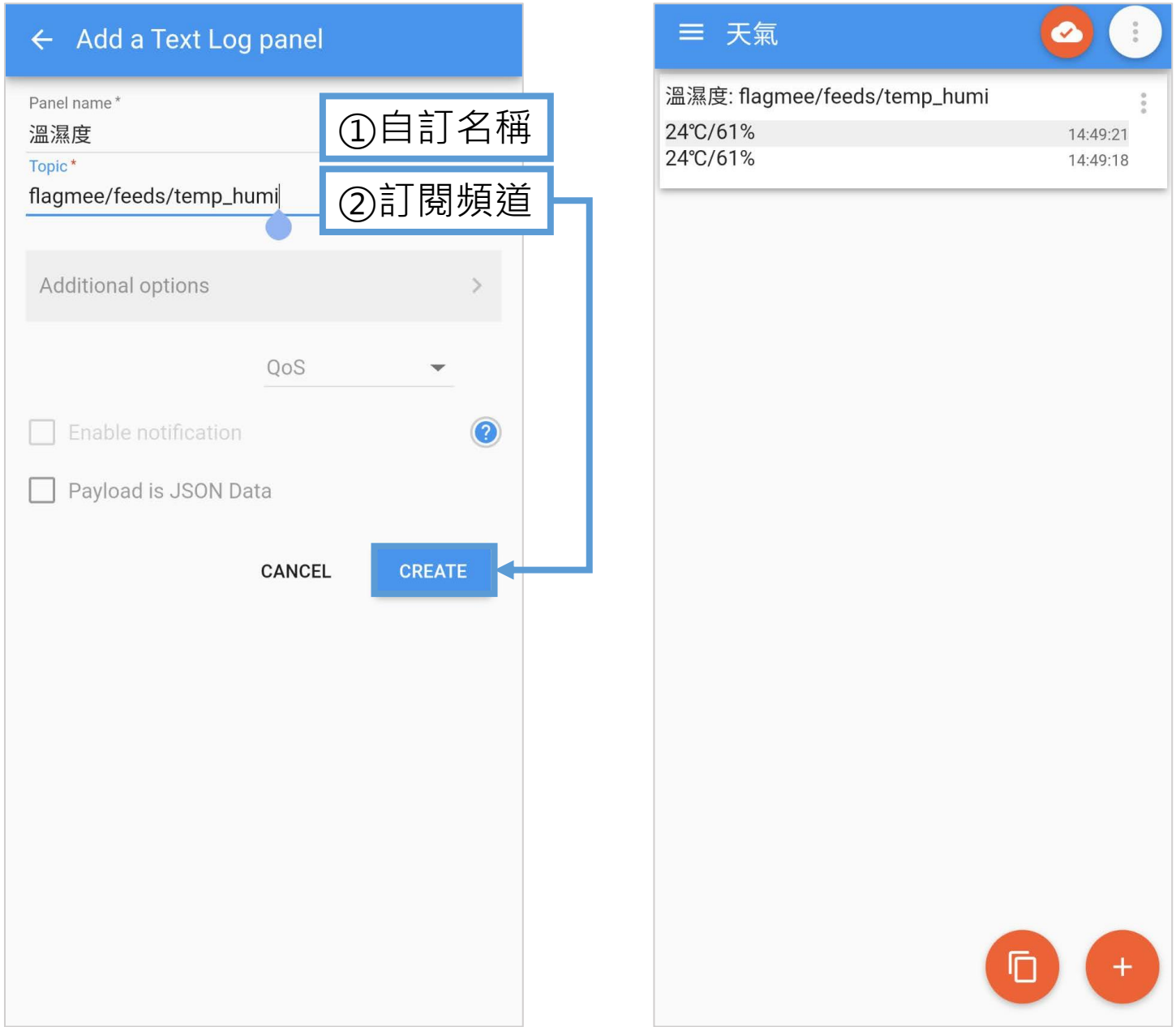
①按此連線



①新增面板

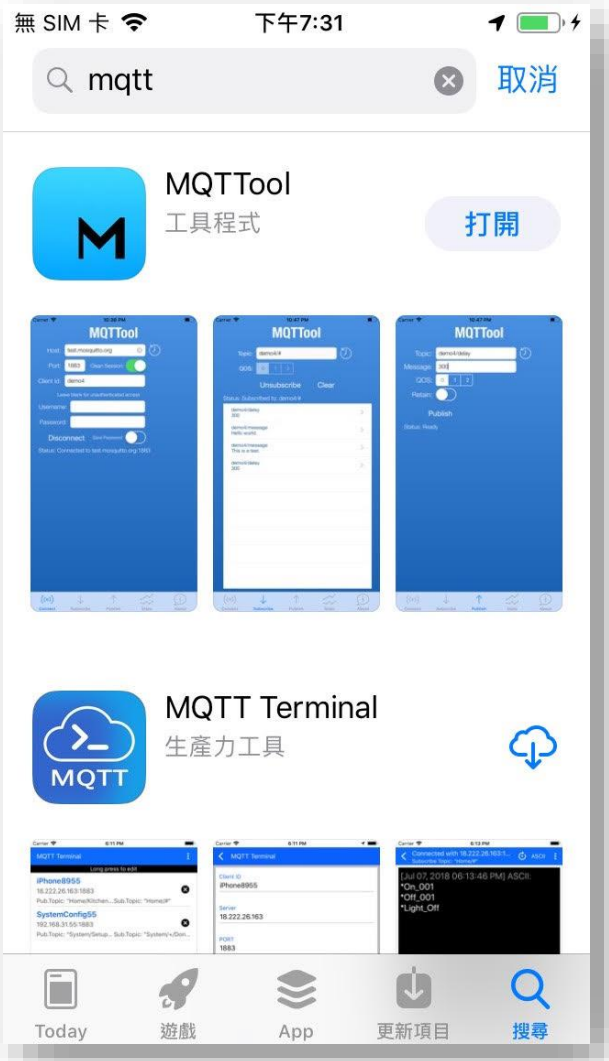


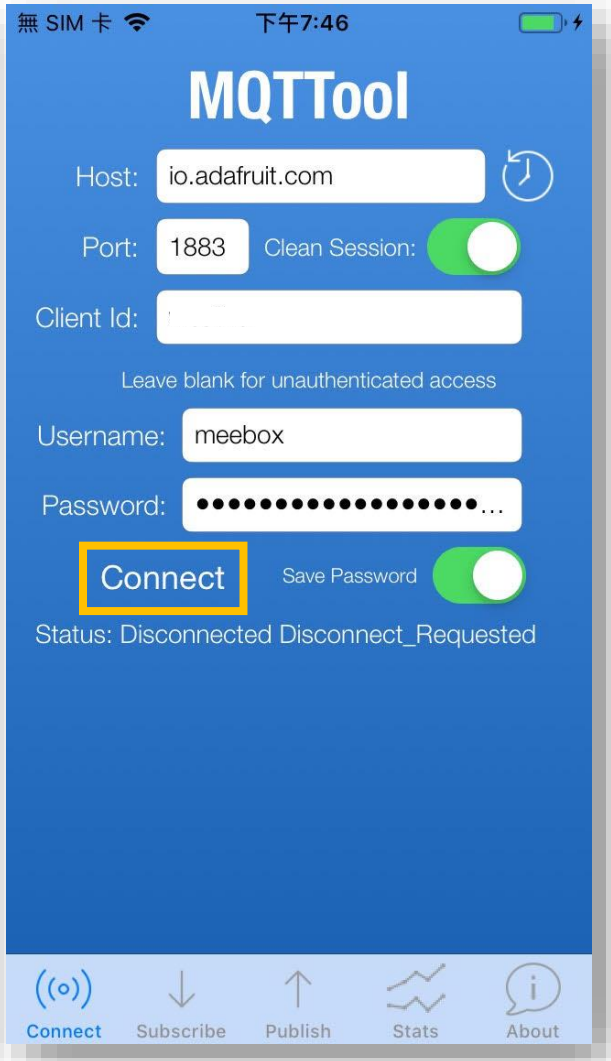
③選用文字元件

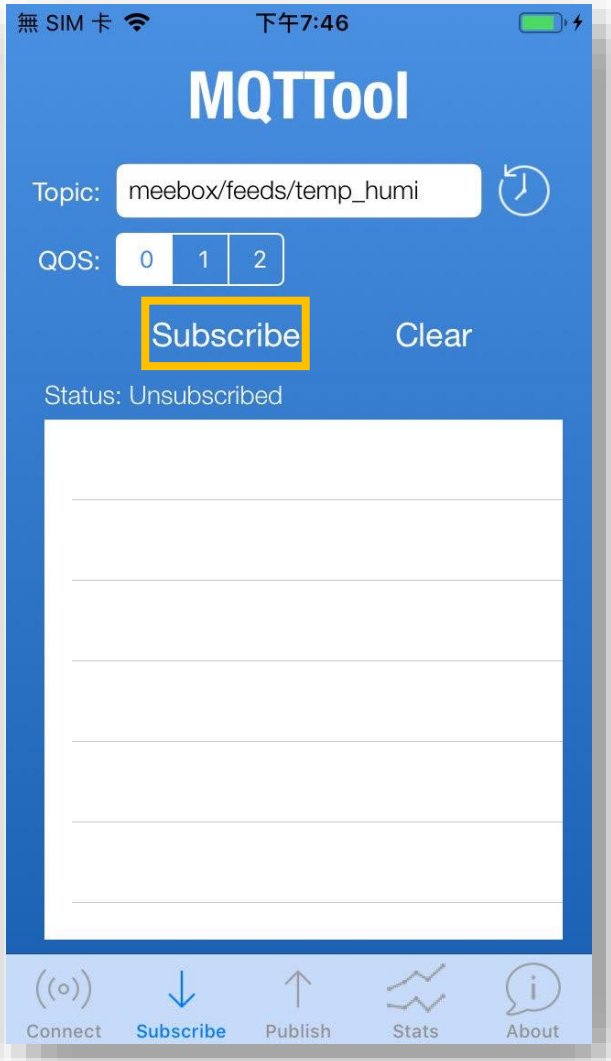

Add a Text Log panel
Panel name *
溫濕度
Topic *
flagmee/feeds/temp_humi
①自訂名稱
②訂閱頻道
Additional options
QoS
Enable notification
Payload is JSON Data
CANCEL
CREATE

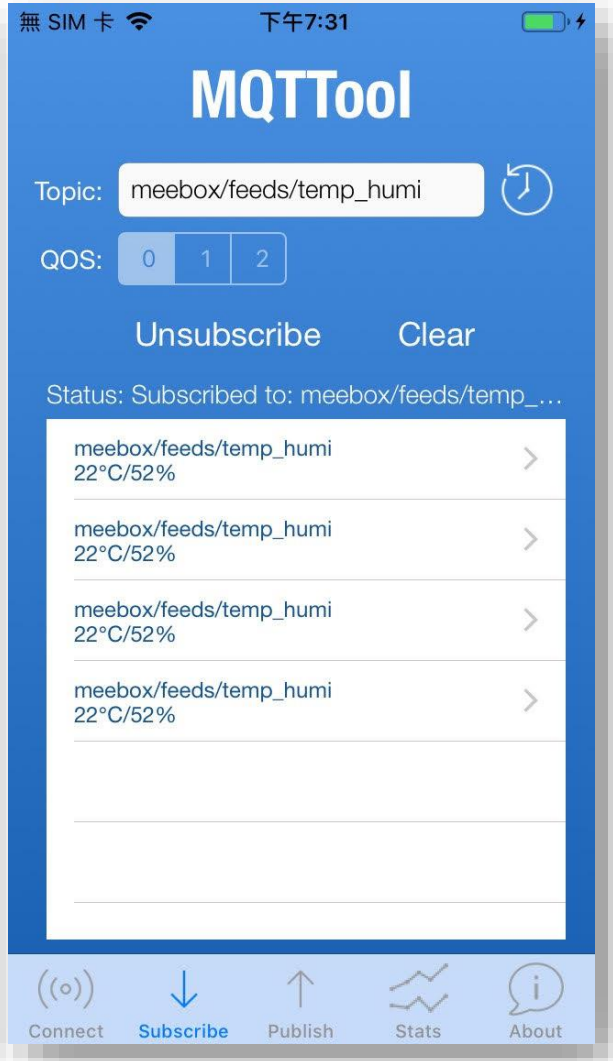
天氣
溫濕度: flagmee/feeds/temp_humi
24℃/61%
14:49:21
24℃/61%
14:49:18

如果使用 iPhone


無 SIM 卡
下午7:31
mqtt
取消
MQTTool
工具程式
打開
MQTT Terminal
生產力工具
Today
遊戲
App
更新項目
搜尋


無 SIM 卡
下午7:46
MQTTool
Host:
io.adafruit.com
Port:
1883
Clean Session:
Client Id:
Leave blank for unauthenticated access
Username:
meebox
Password:
Connect
Save Password
Status: Disconnected Disconnect_Requested
Connect
Subscribe
Publish
Stats
About


無 SIM 卡
下午7:46
MQTTool
Topic:
meebox/feeds/temp_humi
QOS:
0
1
2
Subscribe
Clear
Status: Unsubscribed
Connect
Subscribe
Publish
Stats
About


無 SIM 卡
下午7:31
MQTTool
Topic:
meebox/feeds/temp_humi
QOS:
0
1
2
Unsubscribe
Clear
Status: Subscribed to: meebox/feeds/temp_...
meebox/feeds/temp_humi
22°C/52%
meebox/feeds/temp_humi
22°C/52%
meebox/feeds/temp_humi
22°C/52%
meebox/feeds/temp_humi
22°C/52%
Connect
Subscribe
Publish
Stats
About

13 遠端遙控家電

利用 MQTT 雙向溝通

認識繼電器

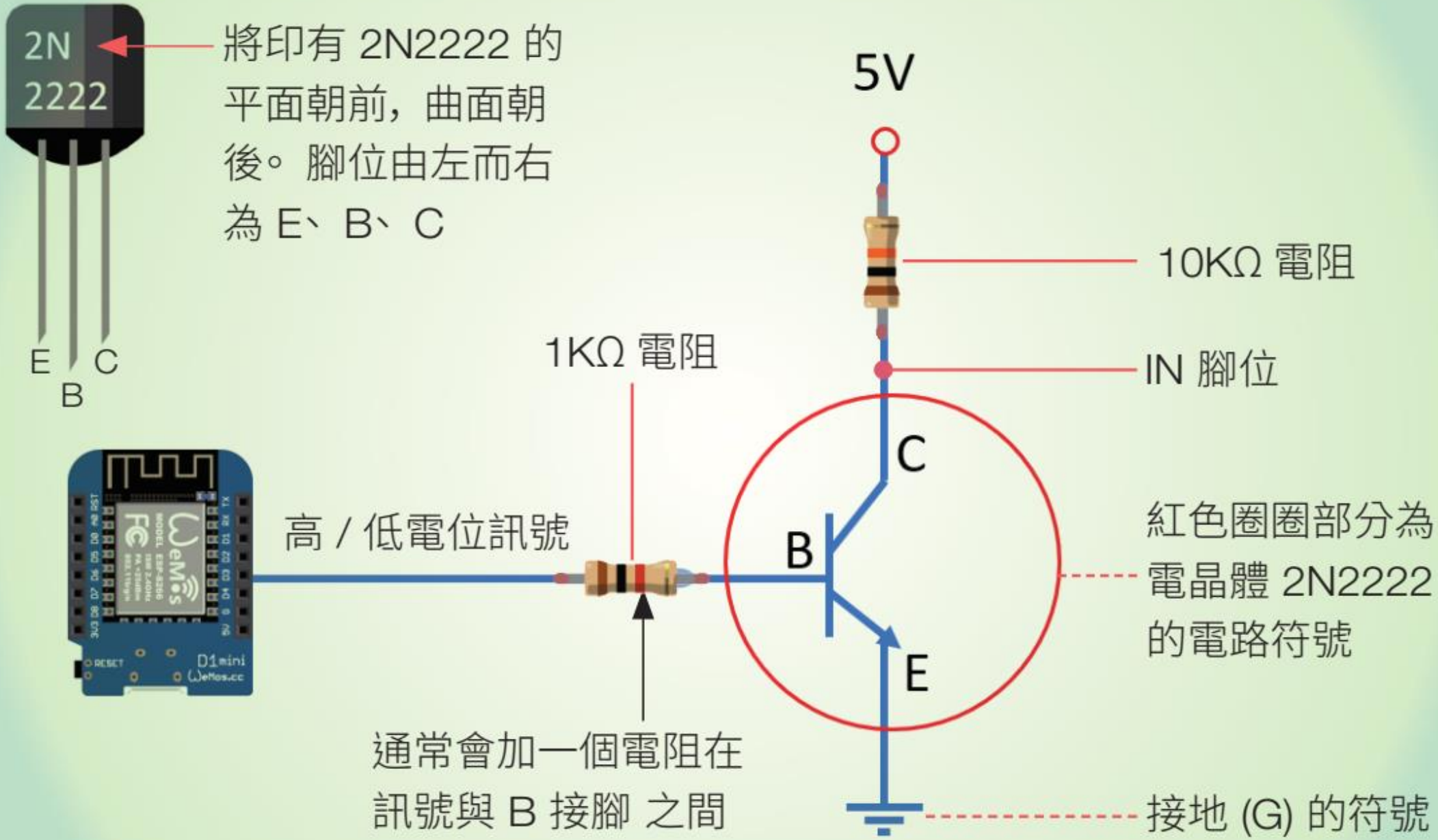
平常我們用來實作創客應用的 D1 mini 或 Arduino 都是以直流供電，如果想要控制使用交流電的家電裝置，必須透過**繼電器 (Relay)** 這個電子元件來控制。繼電器可以用小電流來控制大電流是否通電，並且具備保護電路，能夠避免大電流回流衝擊小電流端：

當我們用 D1 mini 連接繼電器，若對 IN 腳位輸出高電位，則大電流端會斷電，如果對 IN 腳位輸出低電位，那麼大電流端就會通電。

套件中的繼電器需要使用 5V 供電，接到 IN 腳位的也必須是以 5V 為高電位的訊號，而 D1 mini 上的數位輸出腳位都是 3.3V, 因此我們必須藉助電晶體當開關，從 5V 腳位控制變換高低電位訊號到繼電器的 IN 腳位。

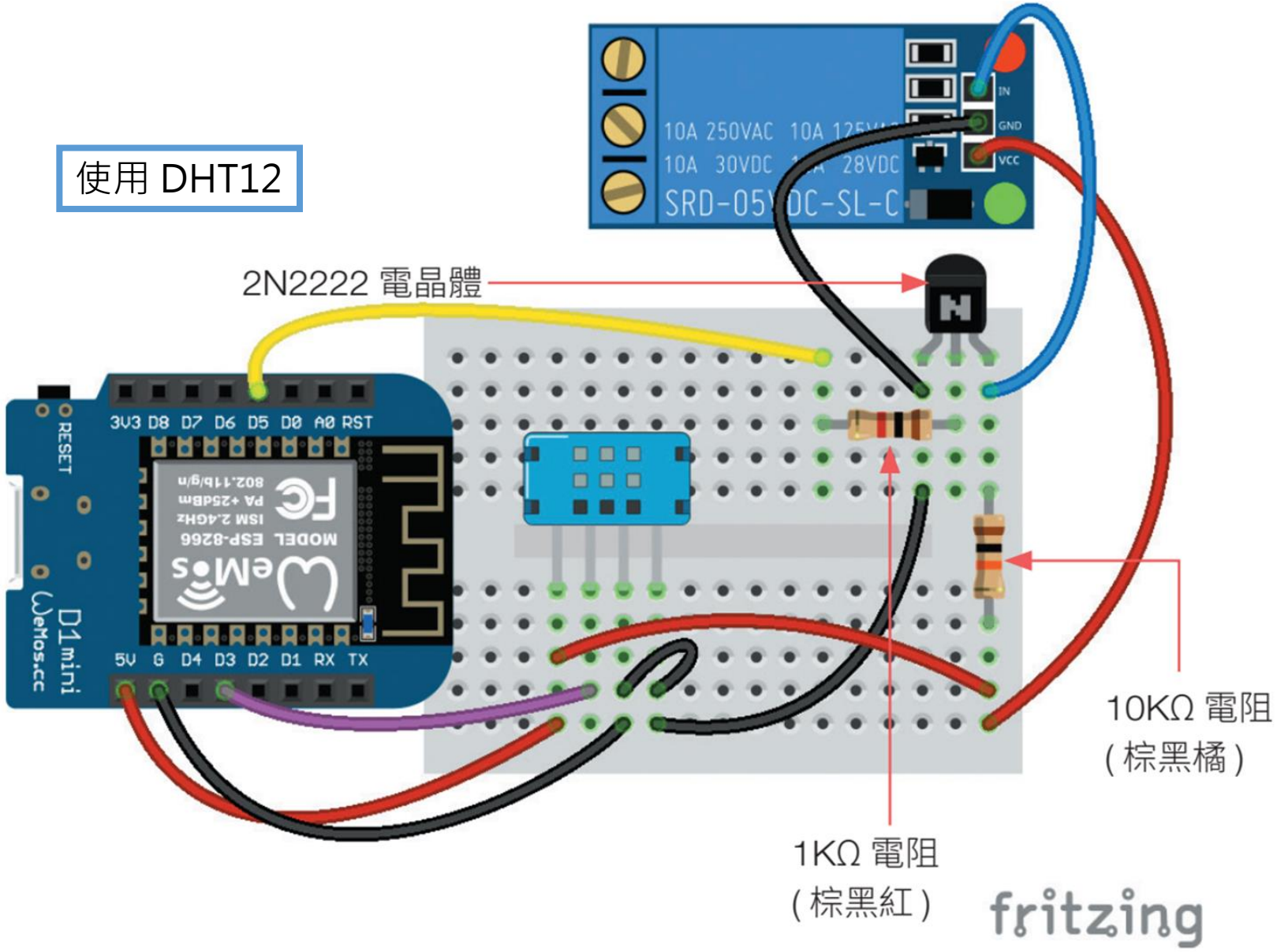
電晶體元件

本套件使用的電晶體型號為 2N2222, 共有 3 隻接腳，分別為 B (基極)、C (集極)、E (射極)。藉由 D1 mini 輸出高電位到 B 接腳，可導通電晶體，讓 C、E 連通，即可送出 E 端與 GND 連通的低電位訊號到 IN 腳位；若給予 B 接腳低電位，則電晶體的 C、E 不連通，即會送出 C 端與 5V 連通的訊號到 IN 腳位。電晶體就像是一個透過電子訊號控制的開關：

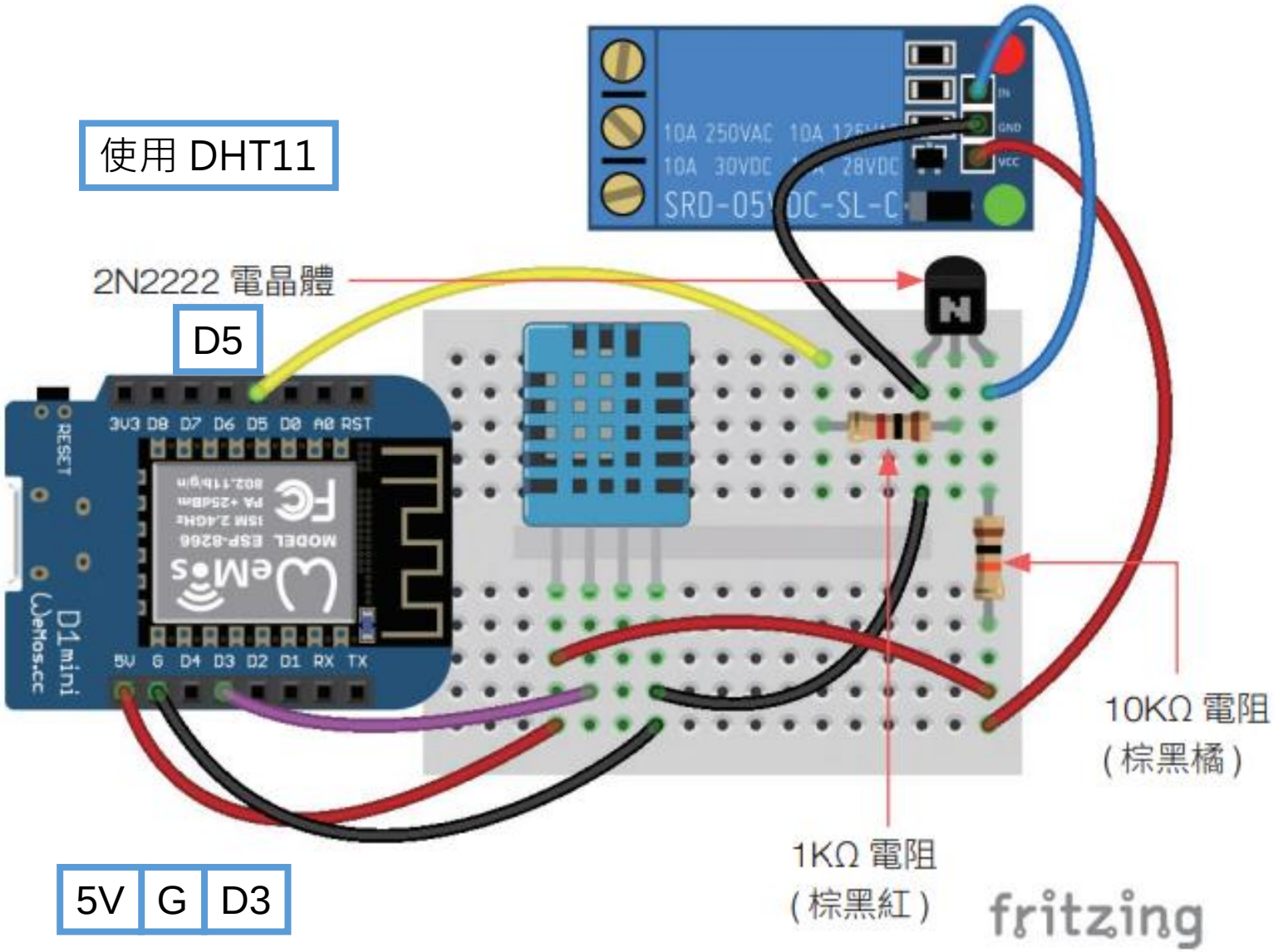

2N
2222
E
B
C
將印有 2N2222 的
平面朝前，曲面朝
後。腳位由左而右
為 E、B、C
5V
10KΩ 電阻
IN 腳位
1KΩ 電阻
C
B
E
高 / 低電位訊號
紅色圈圈部分為
電晶體 2N2222
的電路符號
通常會加一個電阻在
訊號與 B 接腳 之間
接地 (G) 的符號

請依線路圖接線

使用 DHT12



請依線路圖接線



使用 umqtt 模組訂閱頻道

umqtt 模組也提供用戶端向 MQTT 中介伺服器訂閱頻道的功能，首先要準備一個收到訂閱資料時會被自動呼叫的函式，例如：

```
def get_cmd(topic, msg):          # 頻道名稱, 資料
    if msg == b"on":              # bytes 物件的比較
        print("收到打開命令")
    elif msg == b"off":
        print("收到關閉命令")
```

函式名稱可以隨意取，但一定要有 2 個參數，第 1 個參數是頻道名稱、第 2 個參數是收到的資料。由於所有訂閱的頻道有新資料時都是由此函式處理，因此就必須要以頻道名稱來判斷此次資料屬於哪一個頻道？要注意的是這 2 個參數都是 bytes 物件，若要與字串進行比較，必須轉換資料型別，像是上例中就把要比較的 "on" 字串加上 b 前綴字轉換成 bytes 物件。

定義好函式後，還要將該函式註冊為收到訂閱資料時的處理函式：

```
client.set_callback(get_cmd)
```

接著就可以向伺服器訂閱頻道：

```
client.subscribe(b"帳戶名稱/feeds/fan");
```

最後還有一個關鍵的步驟，就是要不斷檢查是否有新的資料：

```
while True:
    client.check_msg()
```

這樣只要發佈端發送了新資料到訂閱的頻道，就會自動呼叫剛剛註冊的函式，收取新的資料了。

Lab13

雲端電源開關

實驗目的	利用 MQTT 從手機端發送命令遙控 D1 mini 開關電器。
材料	• D1 mini • DHT12 溫濕度模組 • 繼電器模組 • 杜邦線及排針若干 • Android 手機 • 10KΩ電阻 • 1KΩ電阻 • 2N2222電晶體

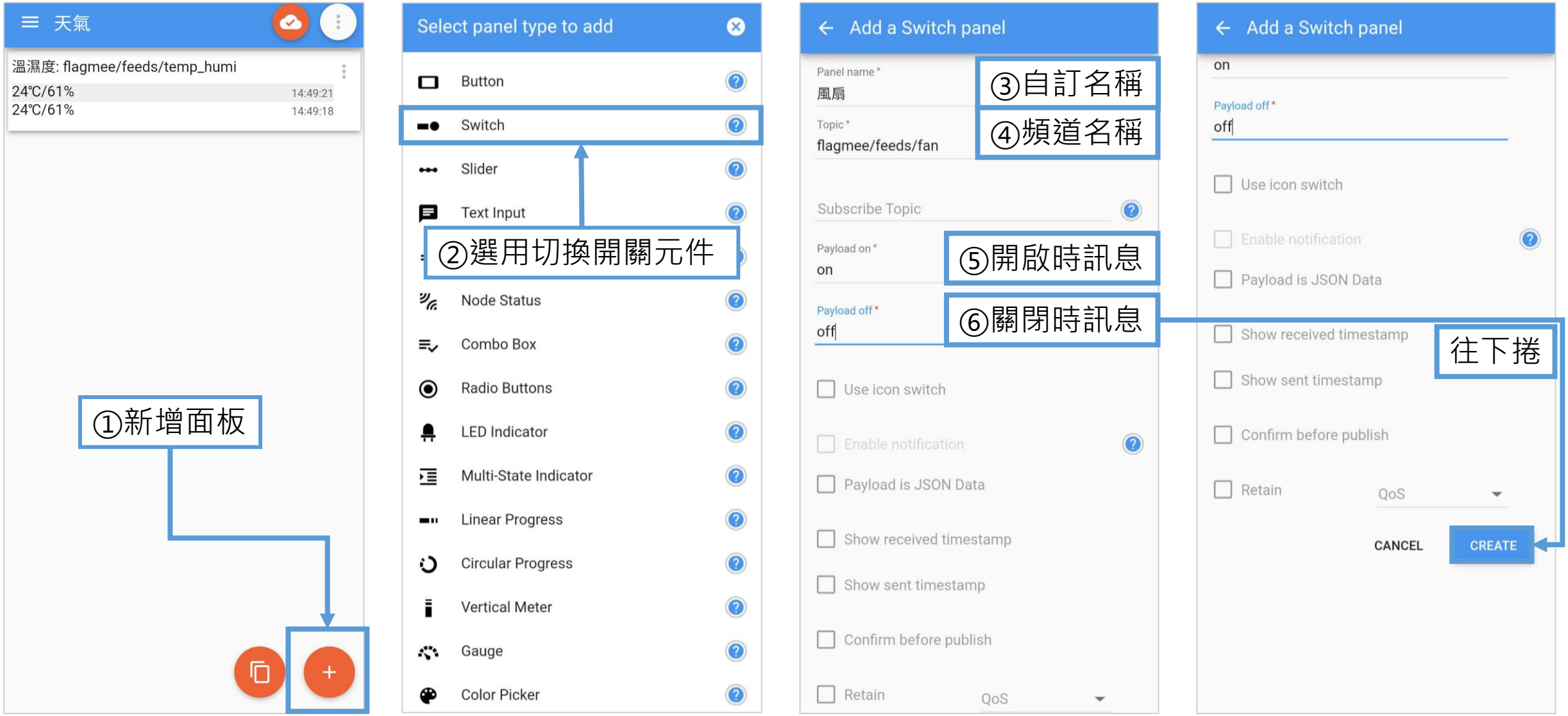
程式設計

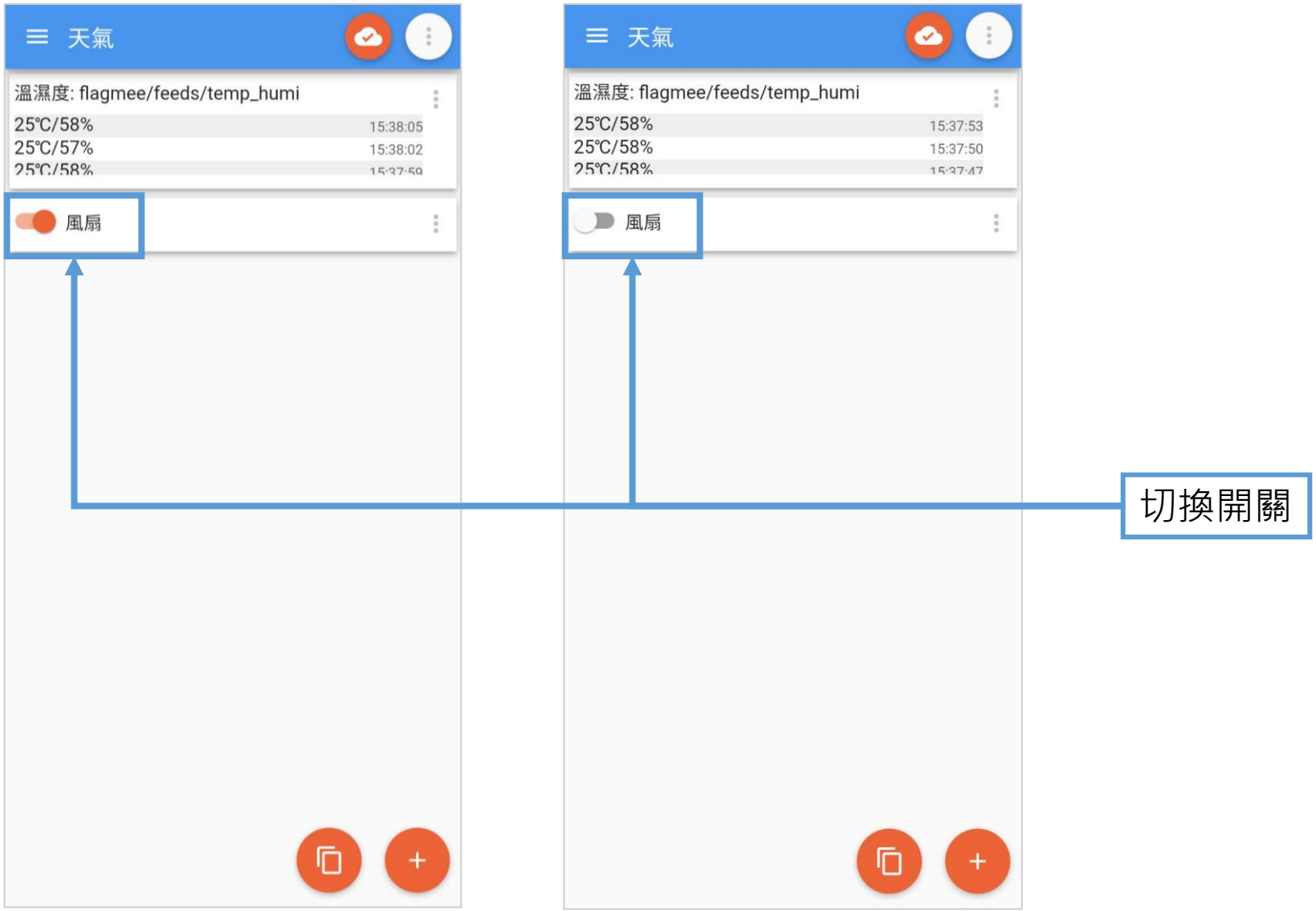
```
01 from machine import Pin
02 import time
03 import network
04 from umqtt.robust import MQTTClient
05 import dht
06
07 sensor = dht.DHT11(Pin(0))  # 使用 D3 腳位建立温溼度物件
08 relay = Pin(14, Pin.OUT, value = 0)  # 使用 D5 腳位控制繼電器
09
10 client = MQTTClient(
11     client_id="raindrop",
12     server="io.adafruit.com",
13     user="帳戶名稱",
14     password="你的金鑰",
15     ssl=False)
16
```

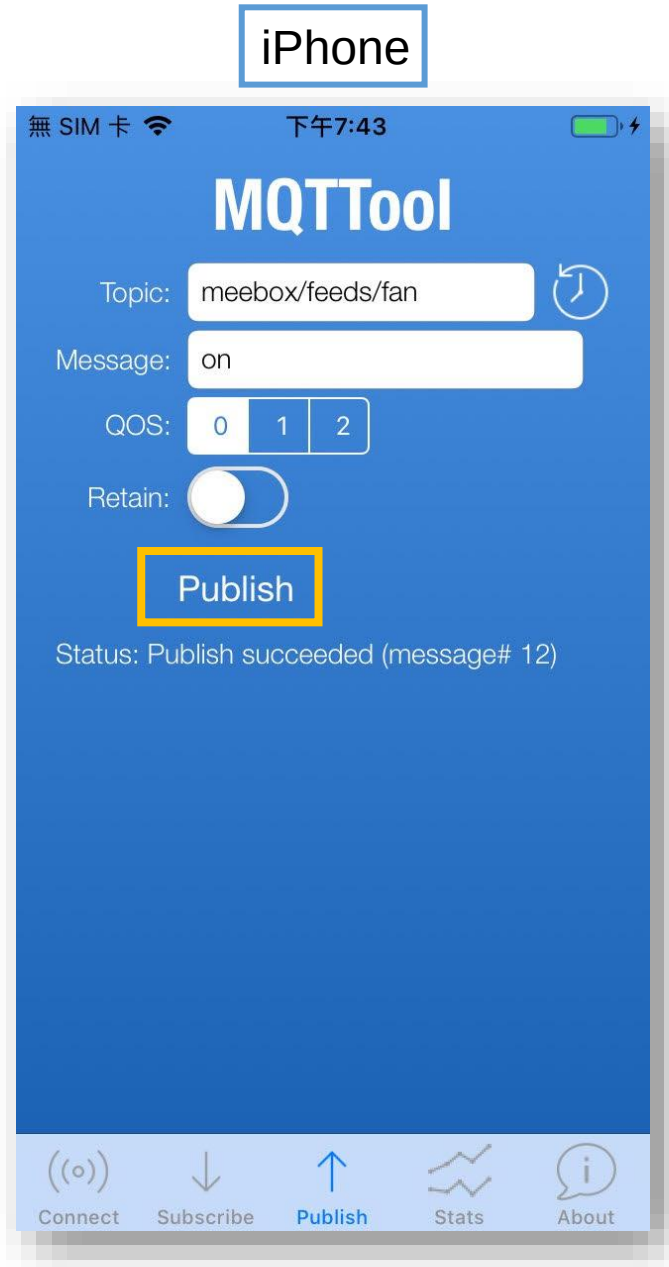
```
17 sta_if = network.WLAN(network.STA_IF)   # 取得無線網路介面
18 sta_if.active(True)                     # 啟用無線網路
19 sta_if.connect('無線網路名稱', '密碼')    # 連結無線網路
20 while not sta_if.isconnected():         # 等待無線網路連上
21     pass
22
23 print("connected")
24
25 def get_cmd(topic, msg):                # 處理新資料的函式
26     if msg == b"on":                    # 收到 on 指令
27         relay.value(1)                  # 打開繼電器
28     elif msg == b"off":                 # 收到 off 指令
29         relay.value(0)                  # 關閉繼電器
30     print(msg)
31
```

```
client.connect()
client.set_callback(get_cmd)                # 註冊處理函式
client.subscribe(b"帳戶名稱/feeds/fan");    # 訂閱頻道

last_time = 0                  # 記錄前次發送資料的時間點
while True:
    if time.time() - last_time >= 3: # 若上次發送已超過 3 秒
        sensor.measure()
        temp_humi = "%2d ℃/%2d%%" % (
            sensor.temperature(),
            sensor.humidity())
        client.publish(
            b"帳戶名稱/feeds/temp_humi",
            temp_humi.encode())
        last_time = time.time()             # 記錄本次發送時間點
    client.check_msg()                      # 檢查新訊息
```

從手機端 App 發布指令-Android




天氣
溫濕度: flagmee/feeds/temp_humi
25℃/58% 15:38:05
25℃/57% 15:38:02
25℃/58%
風扇
天氣
溫濕度: flagmee/feeds/temp_humi
25℃/58% 15:37:53
25℃/58% 15:37:50
25℃/58%
風扇
切換開關

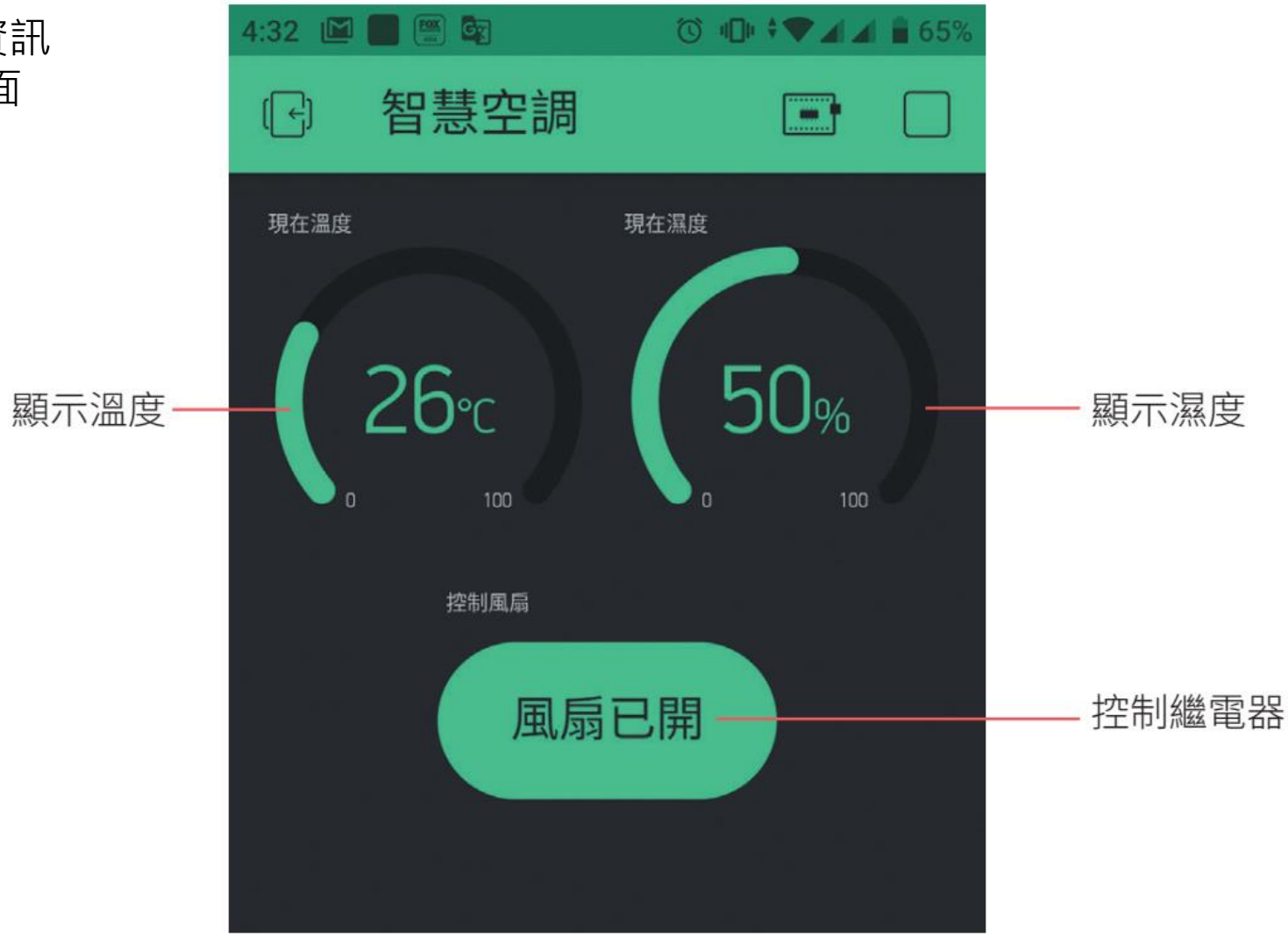

iPhone
無 SIM 卡
下午7:43
MQTTool
Topic:
meebox/feeds/fan
Message:
on
QOS:
0
1
2
Retain:
Publish
Status: Publish succeeded (message# 12)
Connect
Subscribe
Publish
Stats
About

14 手機 APP 感測遙控

Blynk 自訂介面手機 APP

實驗目標

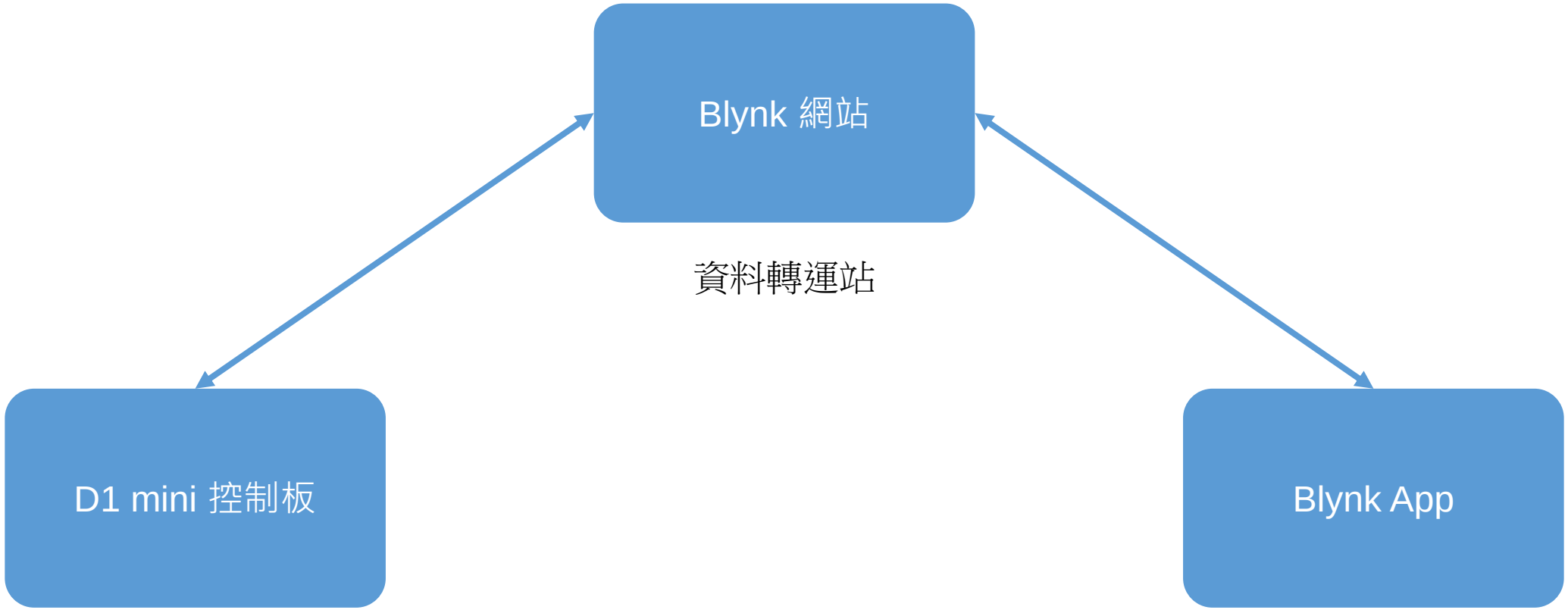
透過 Blynk 服務在手機與 D1 mini 間交換資訊
並藉由 Blynk 手機 App 設計漂亮好用的介面



物聯網裝置傳輸資料的難題

除了讀取感測值顯示在螢幕上以外，我們希望可以讓溫濕度感測值透過網路傳送到你的手機上顯示，甚至可以從手機上送出指令遙控控制板，這需要控制板與手機之間能夠**雙向傳輸**。要做到以上所述，最大的難題是要**突破網路的安全限制**，舉例來說，D1 mini 控制板可以藉由 Wi-Fi 網路連上外部的伺服器，像是前面的章節就連接到 IFTTT 等服務，但如果要從外部的裝置主動連線到位於 Wi-Fi 網路內部的 D1 mini 控制板，就會被 Wi-Fi 寬頻分享器或路由器隔絕而無法連線。

突破網路屏障的 Blynk 中介服務簡介



Blynk 的虛擬腳位

透過虛擬腳位與 Blynk App 交換訊息

Blynk 的運作方式是把手機**模擬成一塊控制板**，擁有 256 個**輸出入腳位**，依序稱為 V0、V1、...V255, 可以和外部互相傳輸資料。D1 mini 這一端可以將資料送到特定的虛擬腳位，而手機端的 App 就可以從該腳位讀取資料，進而取得 D1 mini 端的資訊。反之，如果手機端的 App 想要送資料給 D1 mini, 就可以將資料由特定的虛擬腳位送出，D1 mini 這一端便能夠從該虛擬腳位讀取資料。整體運作就像是 D1 mini 與手機端真的有用傳輸線相互連接起來一樣。

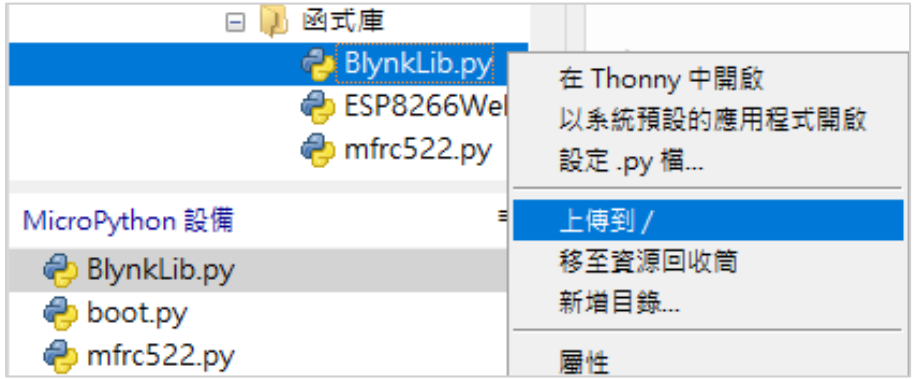
Blynk 程式庫

啟用 Blynk 服務

要使用 Blynk 模組，必須先匯入該模組，接著再建立 Blynk 物件：

```
blynk = BlynkLib.Blynk(token) # 取得 Blynk 物件
```

這裡傳入的 token 是一個字串，稱為**權杖 (token)**。當你在 Blynk 的手機端 App 建立專案時，Blynk 就會幫你產生該專案的權杖，D1 minin 端必須要提供此權杖才能與手機端傳輸資料。在手機端的 App 可以依據需要建立多個專案，每個專案都會有自己的使用介面以及專屬的權杖。



搭配 Blynk 運作的程式庫
在課程一開始就安裝好了

Blynk 向裝置索取資料

送出資料給手機

建立 Blynk 物件後，還要決定要透過哪一個虛擬腳位與手機溝通。我們先以傳送資料到手機為例，首先要先撰寫一個送出資料到特定虛擬腳位的函式，例如（這裡假設 sensor 是我們在第 6 章就介紹過的 DHT11 物件）：

```
def v1_handler():
    sensor.measure()
    blynk.virtual_write(1, sensor.temperature())
```

我們可以使用 Blynk 物件的 virtual_write 方法，即可將資料送出到指定的虛擬腳位，其中第 1 個參數就是虛擬腳位的編號，第 2 個參數則是要送出的資料，這裡就是把感測到的溫度送到 V1 虛擬腳位。

註冊處理函式

有了剛剛的函式後，我們還需要將它註冊為手機端要讀取對應虛擬腳位資料時的專屬函式：

```
blynk.on("readV1", v1_handler)
```

這表示如果手機端要**讀取 V1 虛擬腳位**時，就自動呼叫 v1_handler 函式將資料**從 V1 腳位送出去**。

檢查新收到的 Blynk 需求

為了讓註冊的函式生效，我們還需要在主程式中加入無窮迴圈，持續檢查是否有收到新的 Blynk 需求，以便執行對應的處理函式：

```
while True:
    blynk.run()
```

blynk.run() 會檢查是否有收到新的手機端需求，並且依照需求的內容找出要使用的虛擬腳位，再找出並呼叫已註冊的對應函式。

接受手機送來的資料

讀取從手機端送來的資料

為了讀取從手機端送來的資料，一樣必須撰寫專屬的函式，並且註冊到對應的虛擬腳位 (以下假設 relay 是控制繼電器的 Pin 物件)：

```
def v3_handler(value):
    relay.value(int(value[0]))

blynk.on("V3", v3_handler)
```

這裡要注意的是讀取虛擬腳位的函式必須要有 1 個參數，實際上就會傳入手機端寫入對應虛擬腳位的資料。收到的資料是**串列**，我們可以透過索引取出想要的資料。

另外，在註冊專屬的讀取函式時，第 1 個參數只有虛擬腳位的名稱 "V3"，而不像是註冊輸出資料函式時還要加上 read 字首。

認識繼電器

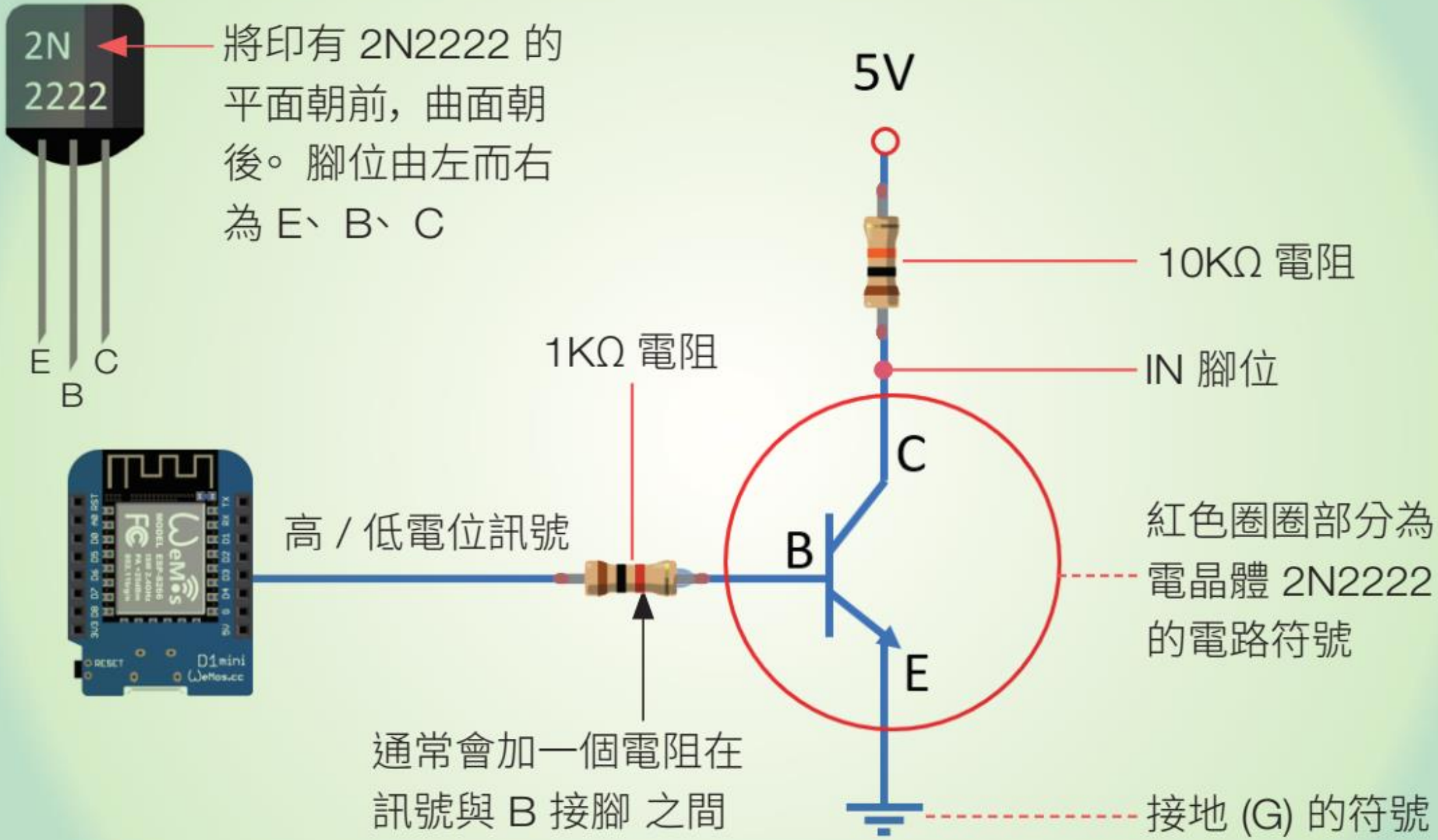
平常我們用來實作創客應用的 D1 mini 或 Arduino 都是以直流供電，如果想要控制使用交流電的家電裝置，必須透過**繼電器 (Relay)** 這個電子元件來控制。繼電器可以用小電流來控制大電流是否通電，並且具備保護電路，能夠避免大電流回流衝擊小電流端：

當我們用 D1 mini 連接繼電器，若對 IN 腳位輸出高電位，則大電流端會斷電，如果對 IN 腳位輸出低電位，那麼大電流端就會通電。

套件中的繼電器需要使用 5V 供電，接到 IN 腳位的也必須是以 5V 為高電位的訊號，而 D1 mini 上的數位輸出腳位都是 3.3V, 因此我們必須藉助電晶體當開關，從 5V 腳位控制變換高低電位訊號到繼電器的 IN 腳位。

電晶體元件

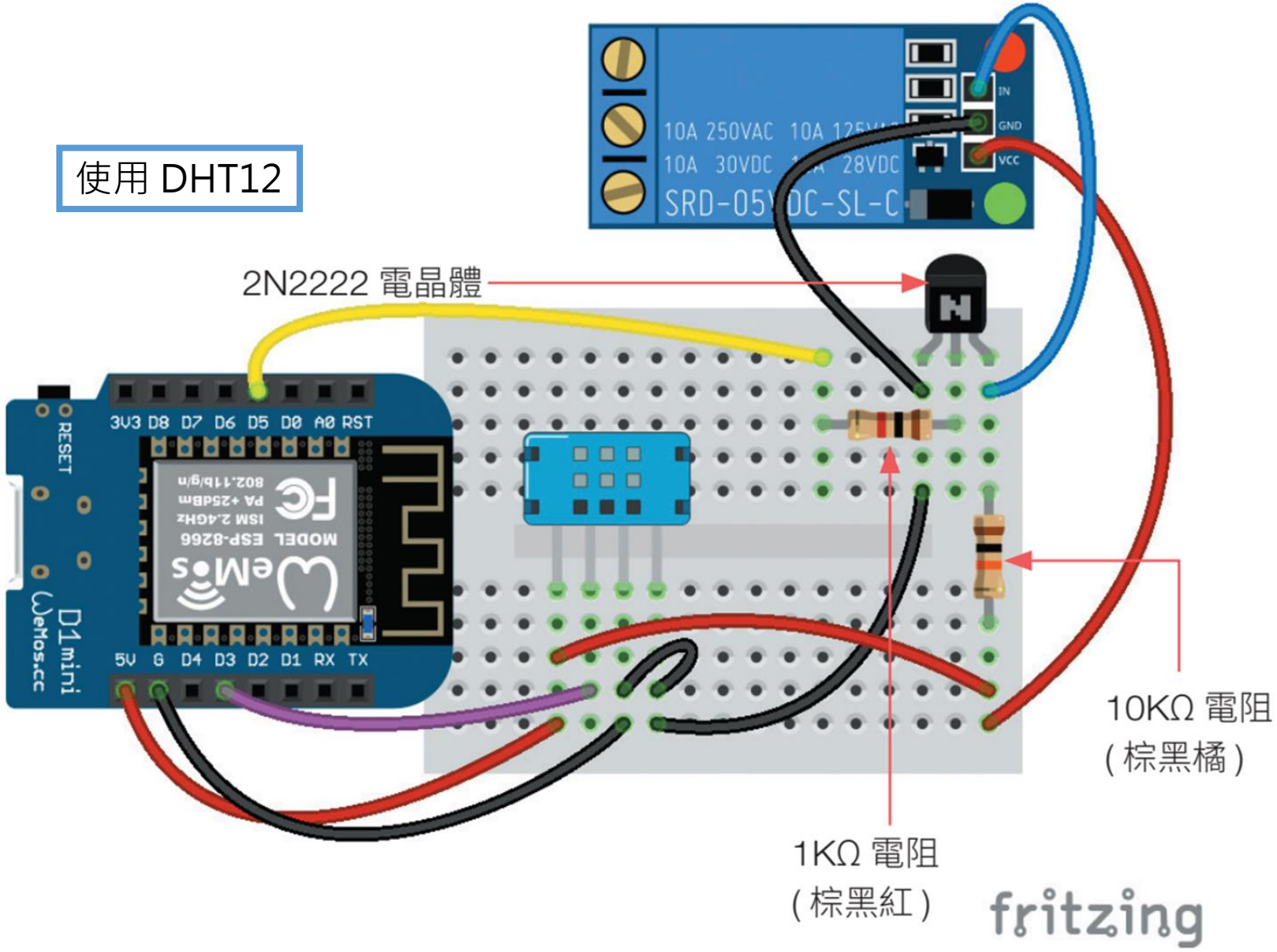
本套件使用的電晶體型號為 2N2222, 共有 3 隻接腳，分別為 B (基極)、C (集極)、E (射極)。藉由 D1 mini 輸出高電位到 B 接腳，可導通電晶體，讓 C、E 連通，即可送出 E 端與 GND 連通的低電位訊號到 IN 腳位；若給予 B 接腳低電位，則電晶體的 C、E 不連通，即會送出 C 端與 5V 連通的訊號到 IN 腳位。電晶體就像是一個透過電子訊號控制的開關：


2N
2222
E
B
C
將印有 2N2222 的
平面朝前，曲面朝
後。腳位由左而右
為 E、B、C
5V
10KΩ 電阻
IN 腳位
1KΩ 電阻
C
B
E
高 / 低電位訊號
紅色圈圈部分為
電晶體 2N2222
的電路符號
通常會加一個電阻在
訊號與 B 接腳 之間
接地 (G) 的符號

請依線路圖接線

線路與LAB13相同

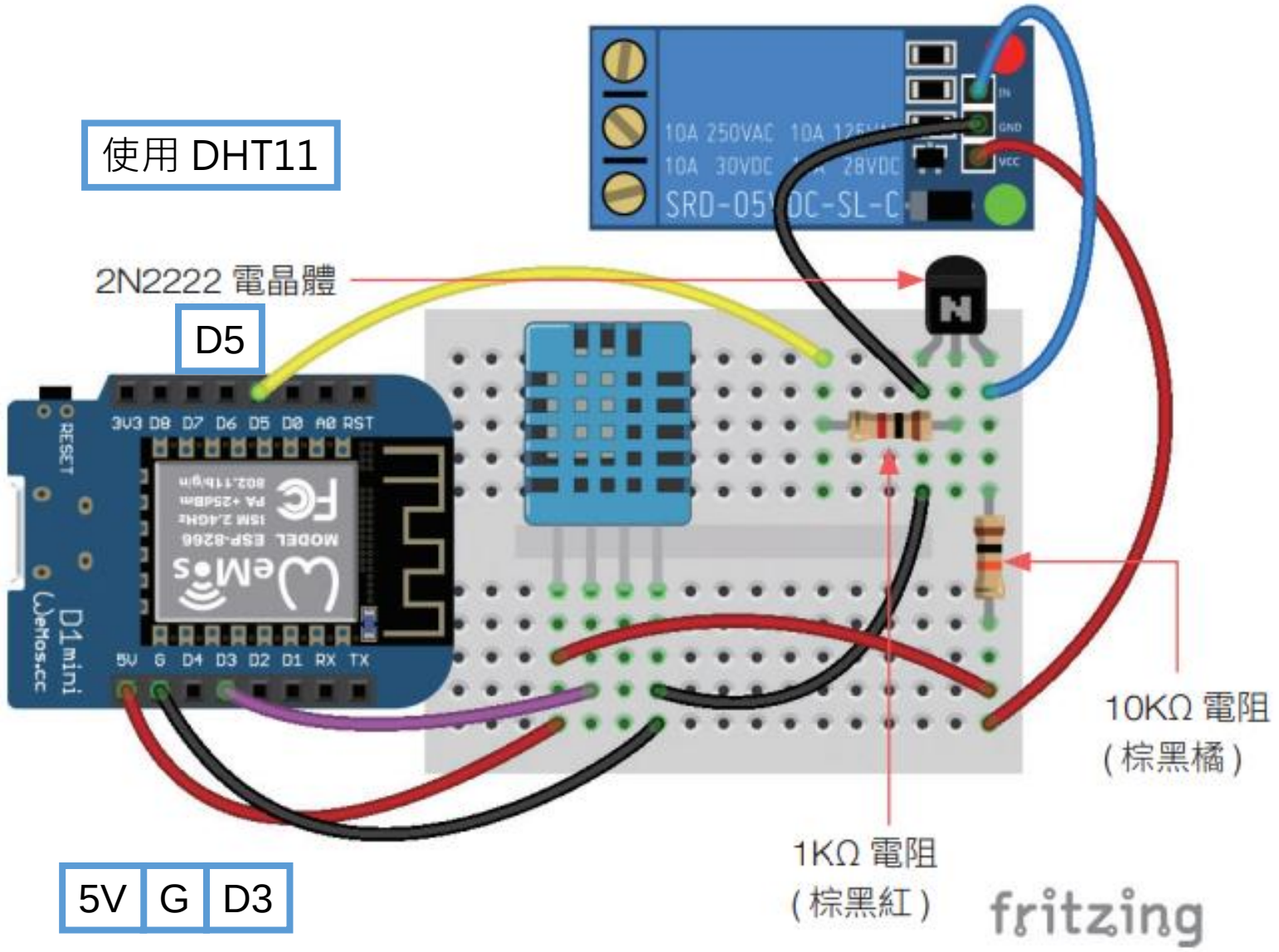
使用 DHT12



請依線路圖接線

線路與LAB13相同

使用 DHT11



5V	G	D3

Lab14

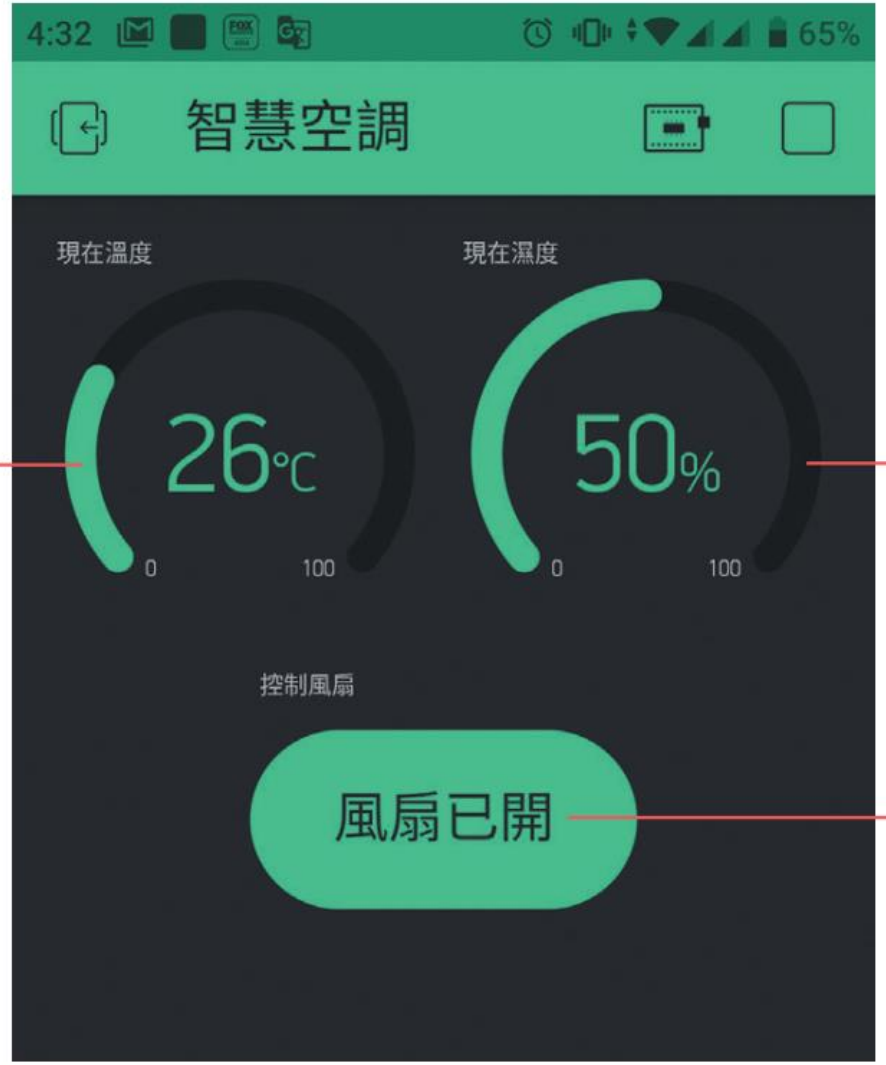
智慧空調手機 APP 雲端感測遙控

實驗目的	利用手機 App 遠端監測溫濕度, 並可下達指令遙控家電開關。
材料	• Di mini • 繼電器 × 1 • DHT12 溫濕度感測器 • 低功率家電 (如檯燈) × 1 • 杜邦線及排針若 • 10KΩ電阻 • 1KΩ電阻 • 2N2222電晶體

實驗結果

溫度與濕度分別透過虛擬腳位V1、 V2 讀取

按鈕會切換 0/1 透過虛擬腳位 V3 送給 D1 mini

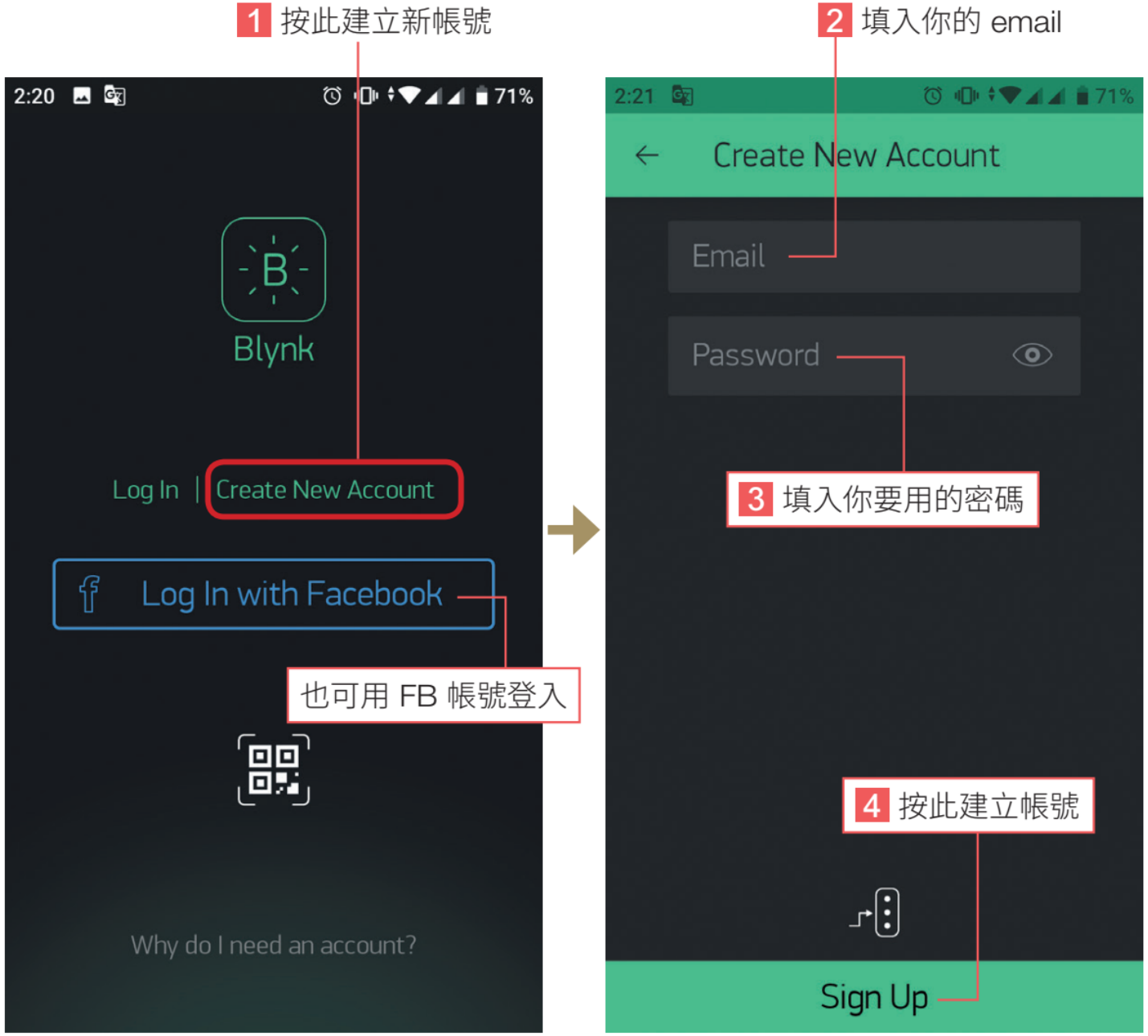


安裝 Blynk App

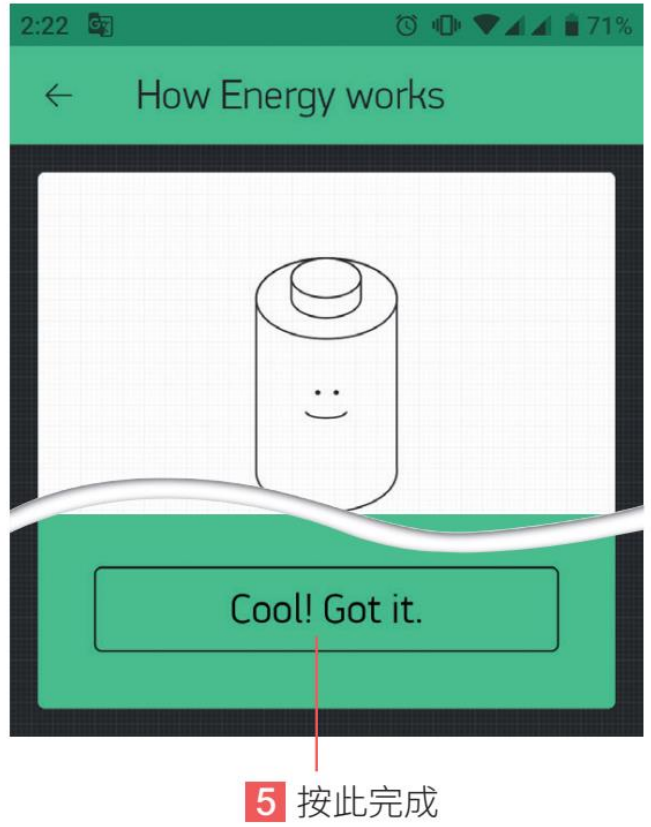
1 使用 Google Play 商店安裝 Blynk App：

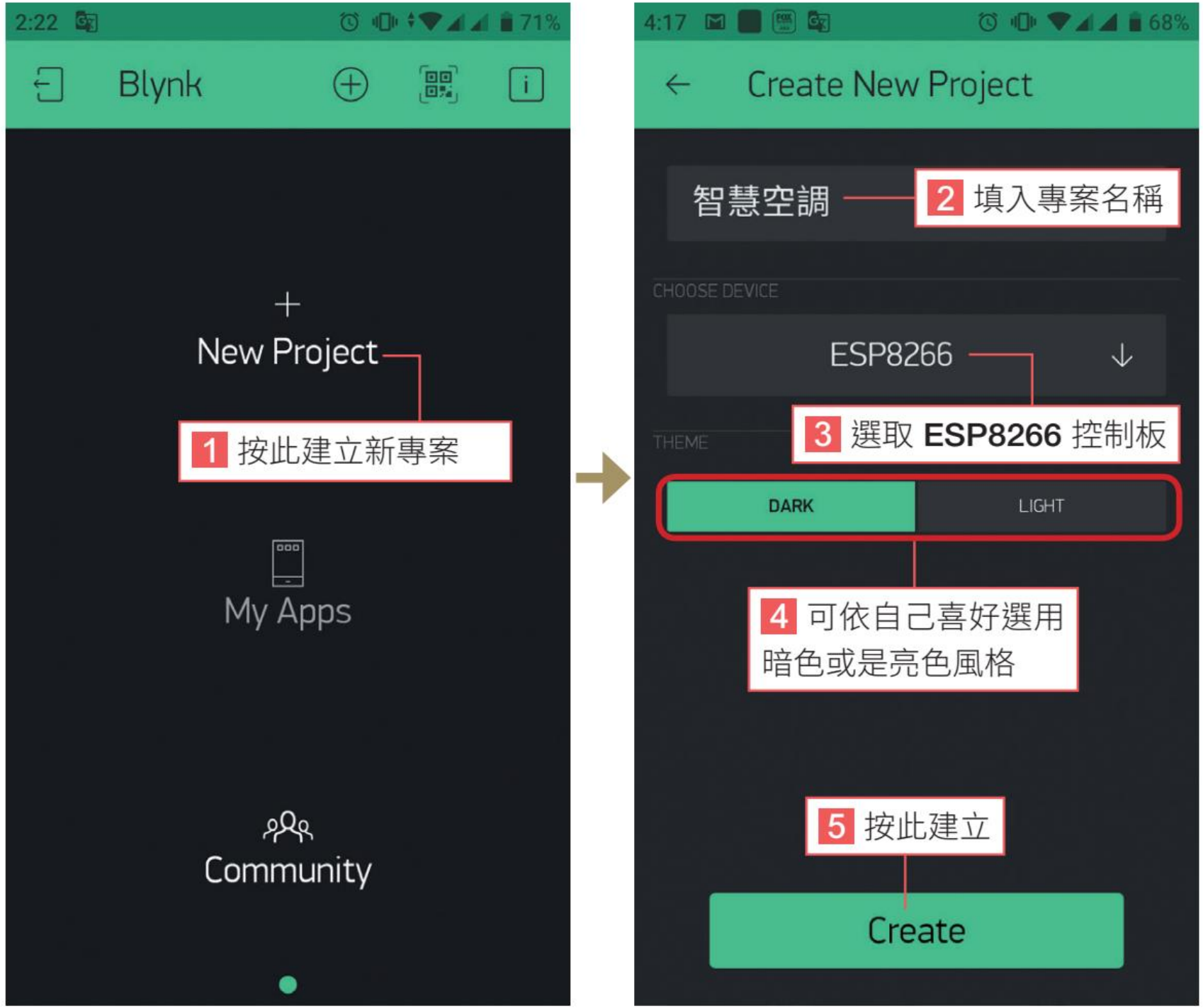
1 搜尋 "Blynk"

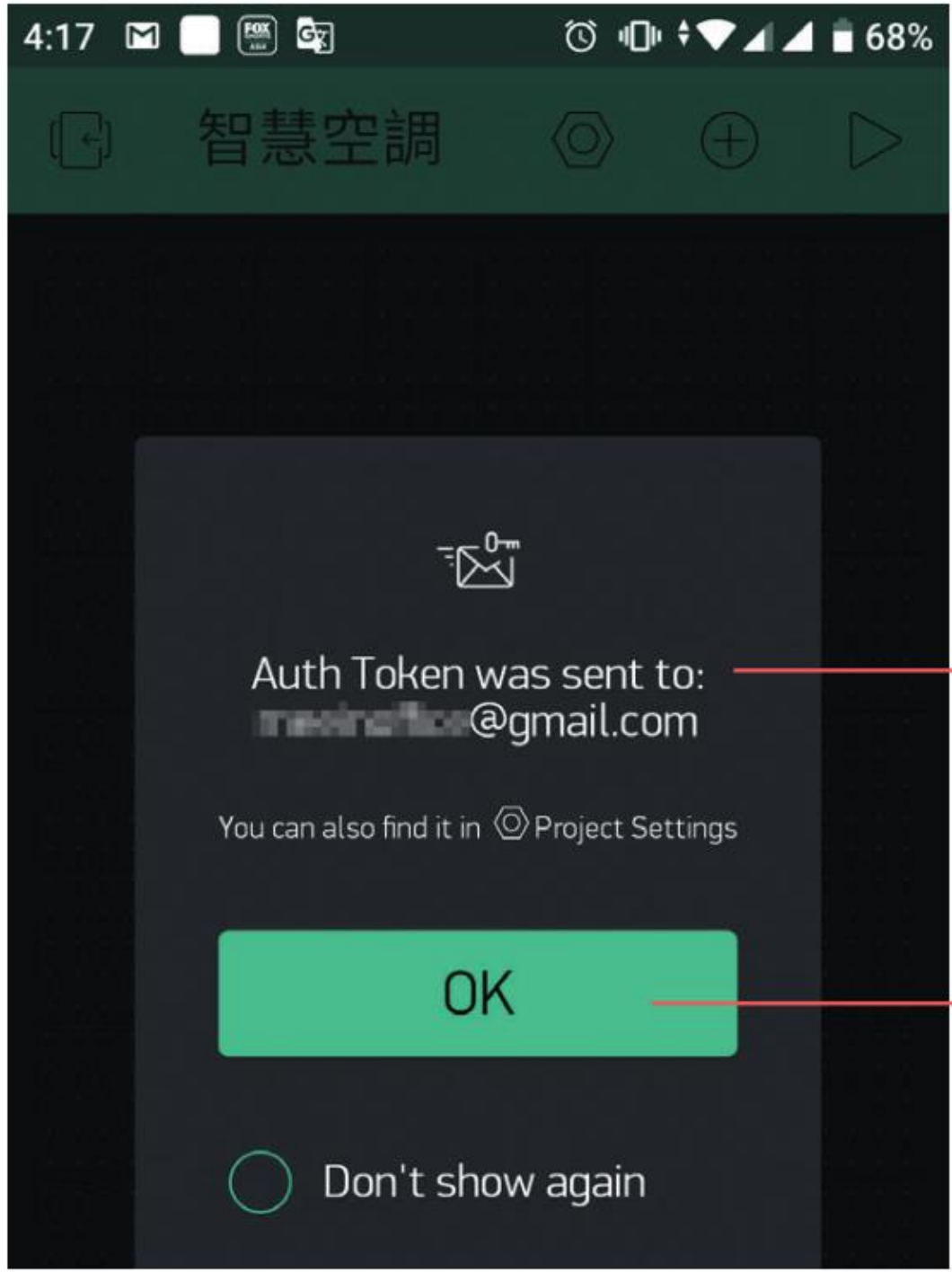
2 點選安裝



這裡請填入方便在待會兒撰寫程式的電腦上收信的信箱，因為 Blynk 會將稍後建立專案時產生的權杖寄到此信箱，你會需要將權杖內容複製到 Python 程式中，如果無法在電腦上收信，複製權杖就會比較麻煩。



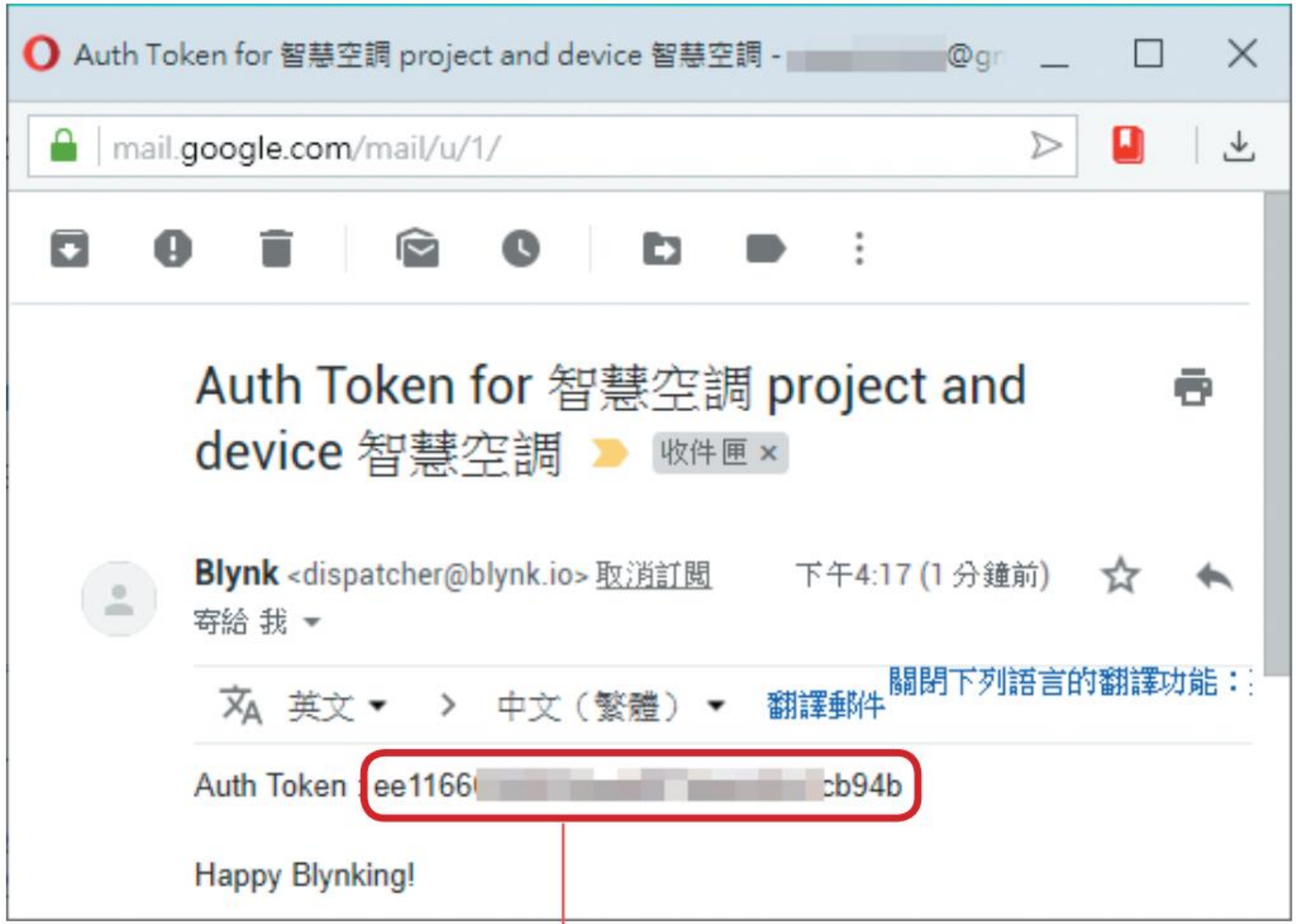

2:22
71%
Blynk
New Project
1 按此建立新專案
My Apps
Community
4:17
68%
Create New Project
智慧空調
2 填入專案名稱
CHOOSE DEVICE
ESP8266
3 選取 ESP8266 控制板
THEME
DARK
LIGHT
4 可依自己喜好選用
暗色或是亮色風格
5 按此建立
Create



6 已經將專案的權杖寄到你指定的信箱

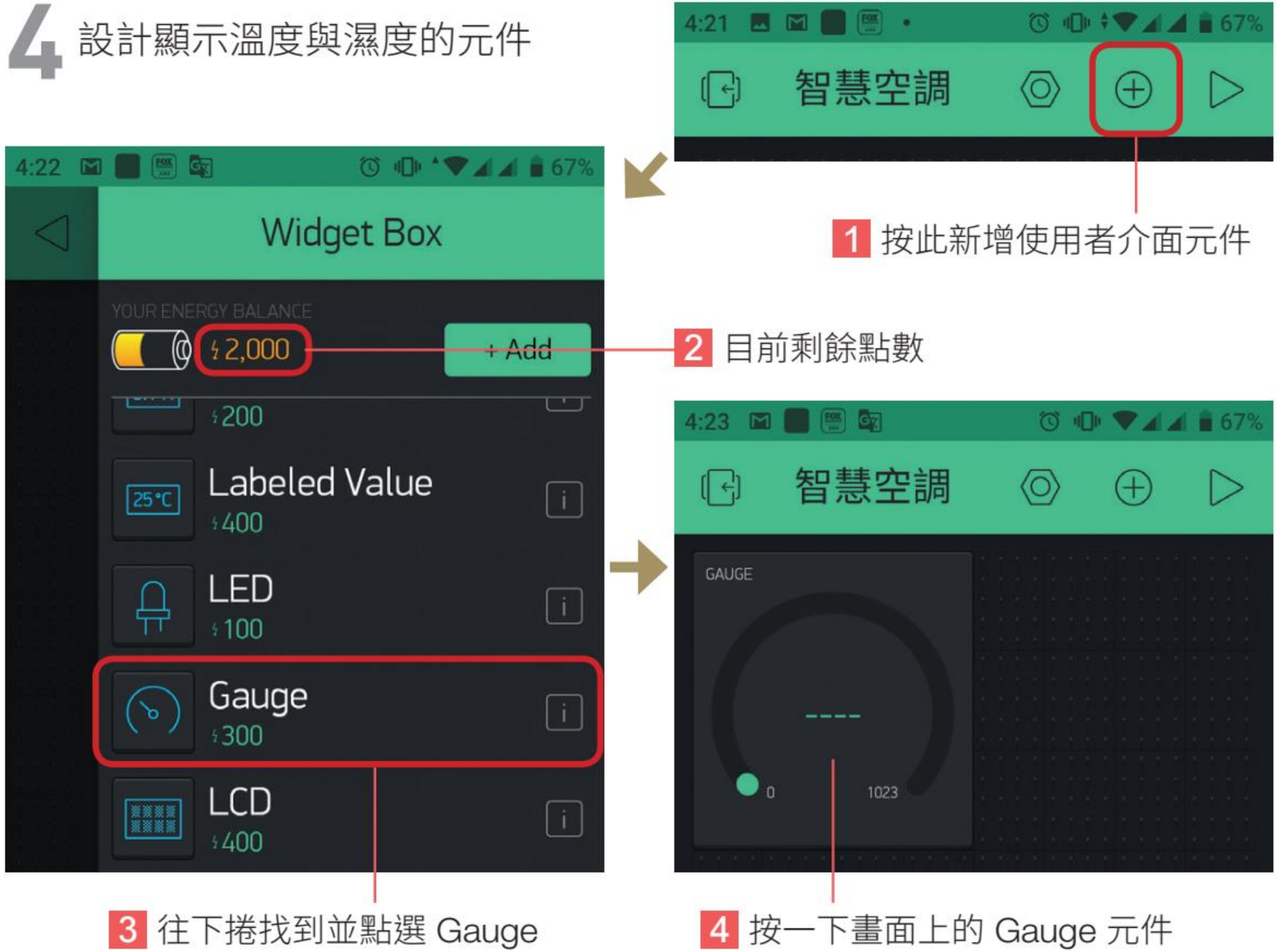
7 按此完成建立程序

請記得到你指定的信箱收信，你會收到像是這樣的一封信：



這就是等一下撰寫程式時要用到的權杖

4 設計顯示溫度與濕度的元件



Blynk 的基本服務是可以免費試用的，但是設計使用者介面的這些元件都要耗費點數，新建立的帳戶內有 2000 點，如果你建了許多專案，或是你的介面用了太多元件，就可能會花完點數，這時就必須付費加值。

14 按此回到設計畫面

10 更改最大值為 100

11 填入單位為 "/pin/°C"，這裡的 "/pin/" 運作時會代入從 V1 腳位讀到的溫度值

12 選用大字體便於閱讀

13 選取 2 **sec** 每秒讀取 2 秒讀取一次數值

21 按此回到設計畫面

15 依據相同步驟再加入一個 Gauge 元件，設定標題為『現在濕度』

16 腳位選用 V2

17 最大值一樣設為 100

18 單位填入 "/pin/%"

19 改用大字體

20 選用 2 **sec**

5 加入控制繼電器的按鈕：

1 按此新增元件

2 往下捲找到並點選 **Button**

3 按一下畫面上的按鈕元件

9 設定好按此回到設計介面

4 按鈕標題設為『控制風扇』

5 選用 V3 腳位

7 更改關閉時文字為『風扇已關』、開啟時為『風扇已開』

8 改用大字體

SWITCH 模式表示按一下以 0->1/1->0 切換值，若是 PUSH 模式，則是按住的時候是開啟、放開按鈕則關閉。

10 按住按鈕不放

11 將按鈕拖曳至此

12 拖曳此控制點將按鈕橫向拉大

最後完成圖

這樣我們就完成了使用者介面的設計，接著就要撰寫程式了。

程式設計

```
01 from machine import Pin
02 import dht, BlynkLib, network                  # 匯入 Blynk 模組
03
04 sta_if = network.WLAN(network.STA_IF)          # 取得無線網路介面
05 sta_if.active(True)                            # 啟用無線網路
06 sta_if.connect('無線網路名稱', '密碼')          # 連接無線網路
07 while not sta_if.isconnected():                # 等待連上無線網路
08     pass
09 print("Wifi 已連上")                           # 顯示連上網路的訊息
10
11 token = '這裡要填入剛剛收到的權杖'
12 blynk = BlynkLib.Blynk(token)                  # 取得 Blynk 物件
13
14 sensor = dht.DHT11(Pin(0))                     # 使用 D3 腳位取得溫溼度
15 relay = Pin(14, Pin.OUT, value = 0)            # 使用 D5 腳位控制繼電器
```

```
16
17 def v1_handler():          # 提供溫度到 V1 虛擬腳位的函式
18     sensor.measure()
19     blynk.virtual_write(1, sensor.temperature())
20
21 def v2_handler():          # 提供濕度到 V2 虛擬腳位的函式
22     # 使用剛剛讀取的感測值
23     blynk.virtual_write(2, sensor.humidity())
24
25 def v3_handler(value):  # 從 V3 虛擬腳位讀取手機按鈕狀態的函式
26     relay.value(int(value[0]))
27
28 blynk.on("readV1", v1_handler) # 註冊 v1_handler 處理 V1 腳位
29 blynk.on("readV2", v2_handler) # 註冊 v2_handler 處理 V2 腳位
30 blynk.on("V3", v3_handler)      # 註冊 v3_handler 處理 V3 腳位
31
32 while True:
33     blynk.run()              # 持續檢查是否有收到 Blynk 送來的指令
```

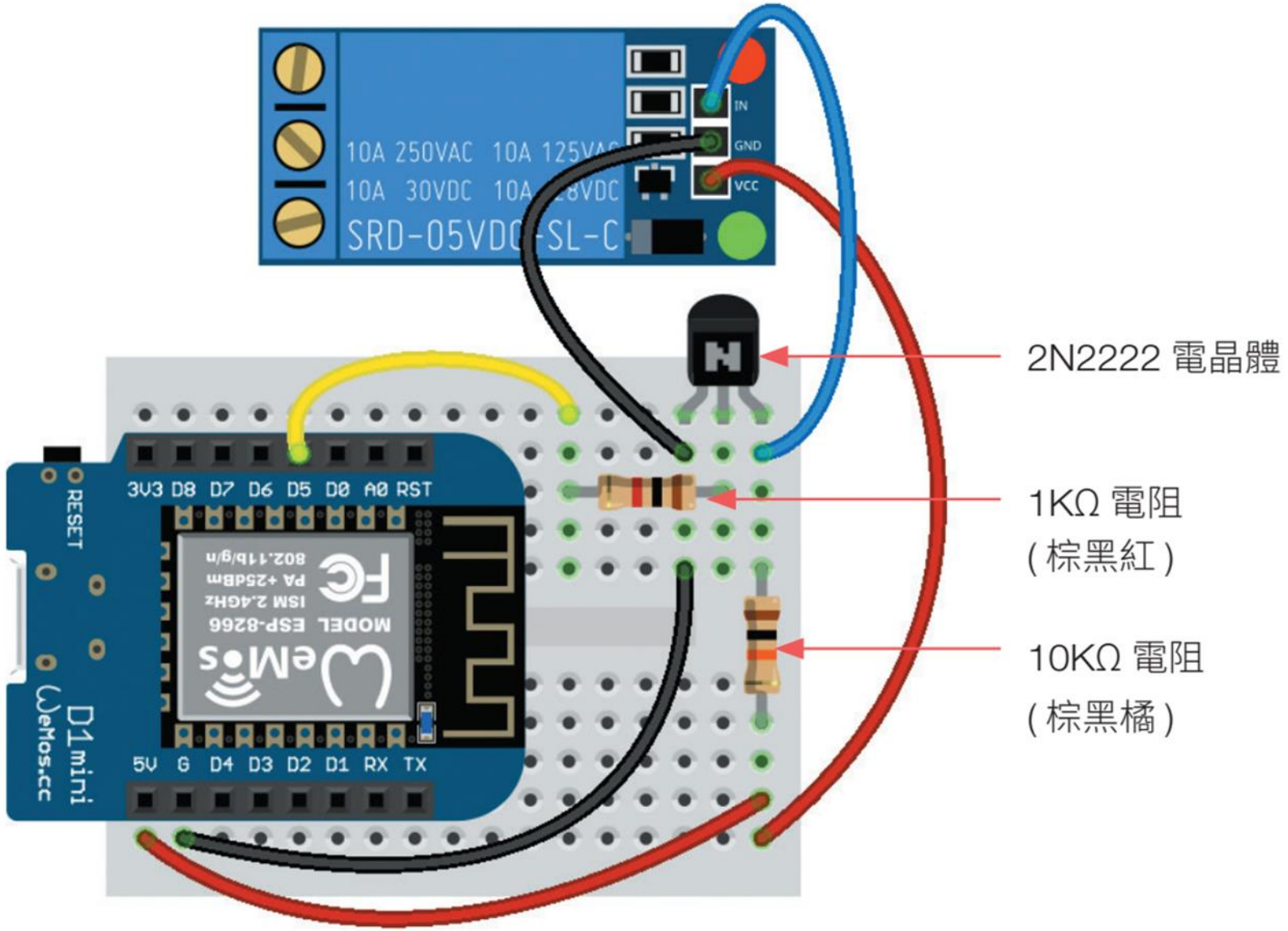
程式執行後，看到『Wifi 已連上』字樣，就可以執行我們剛剛建立好的 Blynk 專案：

15 自製雲端平台

用瀏覽器遙控家電

請依線路圖接線

線路與LAB13相同



把 D1 mini 控制板變網站

為了讓手機或是筆電等裝置都能遙控家電，我們採用最簡單的方式，就是讓 D1 mini 控制板變成網站，接收手機或筆電等裝置送來的指令，這樣只要具備瀏覽器的裝置，就可以用來遙控家電，而不需要為個別裝置設計專屬的 App 或應用程式。

ESP8266WebServer 模組

要讓 D1 mini 變成網站，可以使用 ESP8266WebServer 模組，透過簡單的 Python 程式提供網站的功能。在範例檔中已經提供有該模組，只要開啟 ESP8266WebServer.py 檔案，再執行『**Device/Upload current script with current name**』將模組檔案上傳到 D1 mini 控制板即可使用。

啟用網站

使用 ESP8266WebServer 模組，必須先匯入該模組，接著再啟用網站功能：

```
import ESP8266WebServer      # 匯入模組
ESP8266WebServer.begin(80)   # 啟用網站
```

這裡傳入的 80 稱為**連接埠編號**，就像是公司內的分機號碼一樣，其中 80 號連接埠是網站預設使用的編號，就像總機人員分機號碼通常是 0 一樣。如果更改了這裡的編號，稍後在瀏覽器鍵入網址時，就必須在位址後面加上 ": 編號 "。例如，若網站的 IP 位址為 "192.168.100.38"，啟用網站時將編號改為 **5555**，那麼在瀏覽器的網址列中就要輸入 "192.168.100.38**:5555**"，若**保留 80** 不變，網址就只要寫 "192.168.100.38"，瀏覽器就知道你指的是 "192.168.100.38**:80**"。

處理指令

啟用網站後，還要決定如何處理接收到的指令（也稱為『**請求(Request)**』), 這可以透過以下程式完成：

```
ESP8266WebServer.onPath("/cmd", handleCmd)
```

第 1 個參數是路徑，也就是指令名稱，開頭的 "/" 表示根路徑，需要的話還可以再用 "/" 分隔名稱做成多階層的指令架構。個別指令可透過第 2 個參數指定專門處理該指令的對應函式。在瀏覽器的網址中指定路徑的方式就像這樣：

```
http://192.168.100.38/cmd
```

尾端的 "/cmd" 就是路徑。指令還可以像是函式一樣傳入參數附加額外的資訊，附加參數的方式如下：

```
http://192.168.100.38/cmd?relay=on
```

指令名稱後由問號隔開的部分就是參數，由『參數名稱 = 參數內容』格式指定。本節的範例就會使用名稱為 relay 的參數來切換繼電器開關，參數內容為 "on" 時通電，"off" 時斷電。若需要多個參數，參數之間要用 "&" 串接，例如：

```
http://192.168.100.38/cmd?relay=on&time=50
```

上例中就有 relay 和 time 兩個參數。

對應路徑（指令）的處理工作則是交給指定的函式來處理，在前面的例子中就指定由 handleCmd 來處理 "/cmd" 路徑的請求。處理網站指令的函式必須符合以下規格：

```
def handleCmd(socket, args):
    .....
```

第 1 個參數是用來進行網路傳輸用的物件，要傳送回應資料給瀏覽器時，就必須用到它。第 2 個參數是一個字典物件，內含就是隨指令附加的參數，你可以透過 **in 運算**判斷字典中是否包含有指定名稱的元素，並進而取得元素值，即可得到參數內容。例如：

```
def handleCmd(socket, args):
    if 'relay' in args:                  # 判斷是否有名稱為 relay 的參數
        if args['relay'] == 'on': # 判斷 relay 參數內容是否為 on
            ...
        elif args['relay'] == 'off':
            ...
```

如此即可依據參數內容進行對應的處理。

回應資料給瀏覽器

瀏覽器送出指令後會等待網站回應資料，程式在處理完指令後，可以使用以下程式傳送資料回去給瀏覽器：

```
# 指令正確執行
ESP8266WebServer.ok(socket, "200", "OK")
# 若指令執行發生錯誤, 例如參數不正確
ESP8266WebServer.err(socket, "400", "ERR")
```

第 1 個參數就是處理指令的函式收到的傳輸用物件，第 2 個參數為狀態碼，200 表示指令執行成功、400 則表示錯誤。最後一個參數就是實際要傳送回瀏覽器的資料，這可以是純文字或是 HTML 內容。

檢查新收到的請求指令

為了讓剛剛建立的網站運作，我們還需要在主程式中加入無窮迴圈，持續檢查是否有收到新的指令，執行對應的指令處理函式：

```
while True:
    ESP8266WebServer.handleClient()
```

取得 D1 mini 的 IP

若要檢查連上網路後的相關設定，可以呼叫網路介面物件的 ifconfig()：

```
>>> sta_if.ifconfig()
('192.168.100.39', '255.255.255.0', '192.168.100.254',
'168.95.192.1')
```

ifconfig() 傳回的是稱為『**元組 (tuple)**』的資料，元組是以小括號 () 表示，使用上和串列非常相似。在 ifconfig() 傳回的元組中，共有 4 個元素，依序為**網路位址 (Internet Protocol address, 簡稱 IP 位址)**、**子網路遮罩 (subnet mask)**、**閘道器 (gateway) 位址**、**網域名稱伺服器 (Domain Name Server, 簡稱 DNS 伺服器) 位址**。如果只想顯示其中單項資料，可以使用中括號 [] 標註從 0 起算的索引值 (index), 例如以下即可顯示 IP 位址：

```
print("伺服器位址：" + sta_if.ifconfig()[0])
```

在瀏覽器中就可以依據 IP 位址鍵入控制繼電器的網址了。

Lab15

遠端遙控開關

實驗目的	利用手機或是筆電上的瀏覽器, 直接下達指令遙控家電開關。	
材料	• Di mini • 繼電器 × 1 • 低功率家電 (如檯燈) × 1 • 杜邦線及排針若干	• 10KΩ電阻 • 1KΩ電阻 • 2N2222電晶體

設計原理

本實驗會讓 D1 mini 變成網站，並接受如下的網址 (假設 D1 mini 網站的 IP 位址為 192.168.100.38) 當成指令控制繼電器通電打開電器：

```
http://192.168.100.38/cmd?relay=on
```

以下的指令則會讓繼電器斷電：

```
http://192.168.100.38/cmd?relay=off
```

因此，我們的程式就要能夠處理 "/cmd" 指令，並取出伴隨指令的 "relay" 參數，再依據參數內容是 "on" 還是 "off" 切換繼電器的通電或斷電。

程式設計

```
01 import network
02 import ESP8266WebServer                  # 匯入網站模組
03 from machine import Pin
04
05 def handleCmd(socket, args):            # 處理 /cmd 指令的函式
06     if 'relay' in args:                 # 檢查是否有 relay 參數
07         if args['relay'] == 'on':       # 若 relay 參數值為 'on'
08             relay.value(1)              # 讓繼電器通電
09         elif args['relay'] == 'off': # 若 relay 參數值為 'off'
10             relay.value(0)               # 讓繼電器斷電
11         ESP8266WebServer.ok(socket, "200", "OK")    # 回應 OK
12     else:
13         ESP8266WebServer.err(socket, "400", "ERR") # 回應 ERR
```

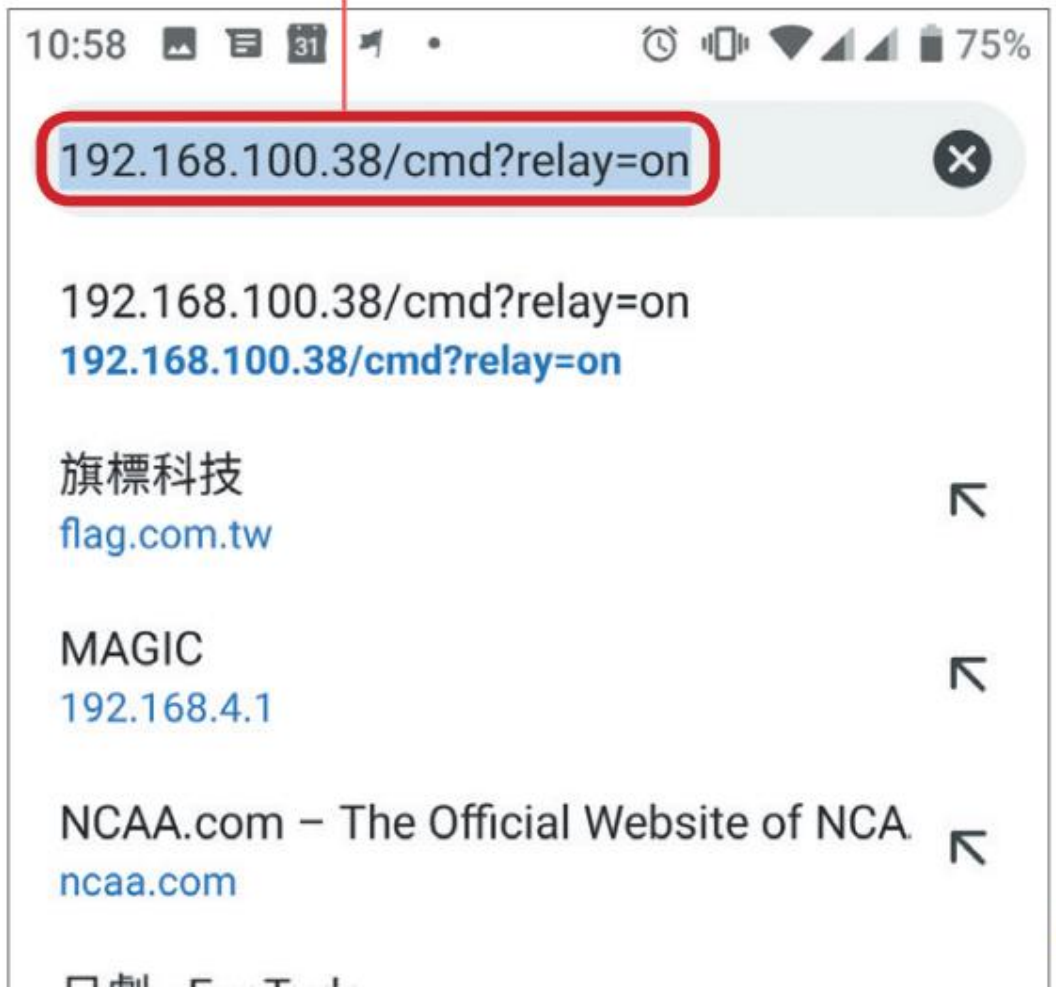
```
14
15 print("啟動中...")
16 sta_if = network.WLAN(network.STA_IF)   # 取得無線網路介面
17 sta_if.active(True)                    # 啟用無線網路
18 sta_if.connect('無線網路名稱', '密碼')    # 連結無線網路
19 relay = Pin(14, Pin.OUT, value=0)       # 控制 D5 腳位並預設斷電
20 while not sta_if.isconnected():         # 等待無線網路連上
21     pass
22
23 ESP8266WebServer.begin(80)                         # 啟用網站
24 ESP8266WebServer.onPath("/cmd", handleCmd) # 指定處理指令的函式
25 print("伺服器位址:" + sta_if.ifconfig()[0])   # 顯示 IP 位址
26
27 while True:
28     ESP8266WebServer.handleClient()   # 持續檢查是否收到新指令
```

請將第 18 列的無線網路名稱及密碼改寫成您自己的無線網路，程式執行後會先看到 D1 mini 的 IP 位址 (本例為 192.168.100.38)：

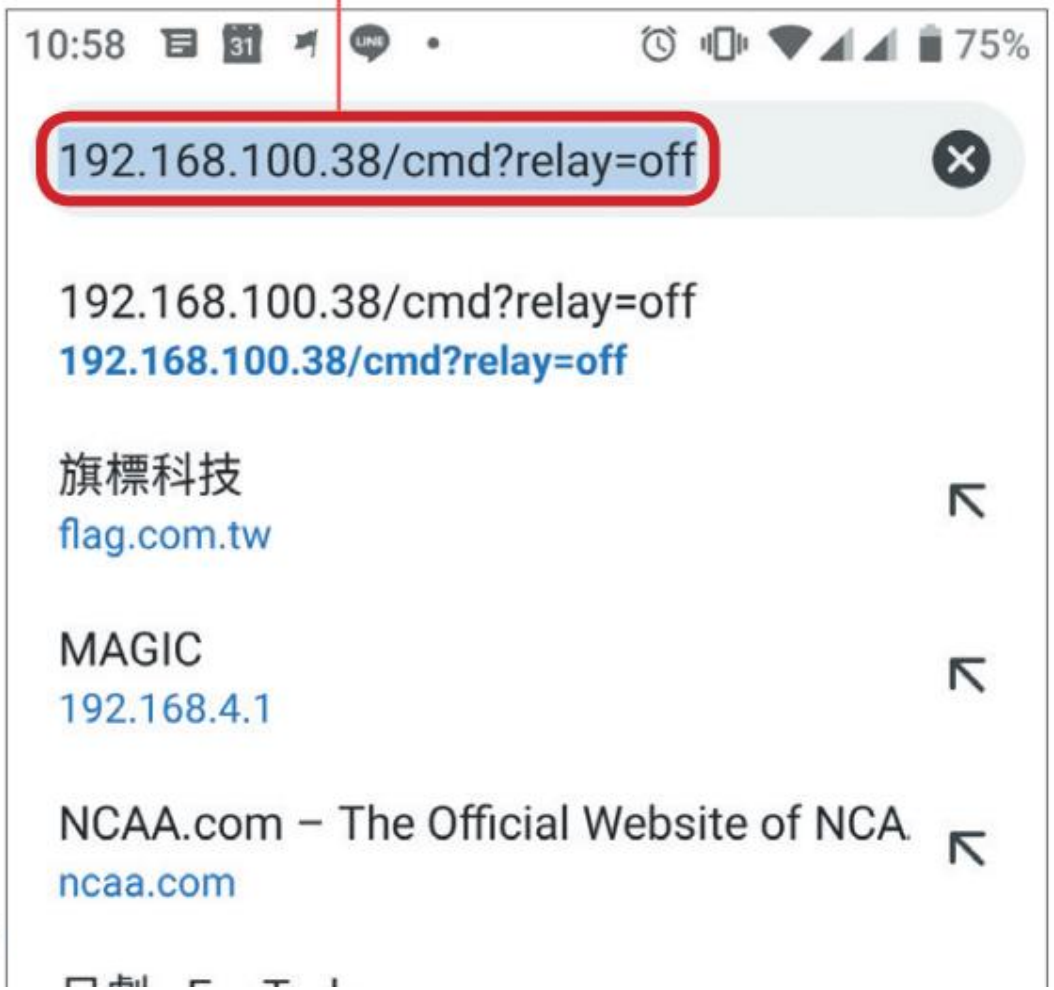
```
>>> %Run RemoteLED_01.py
啟動中...

伺服器位址：192.168.100.38
```

看到 IP 位址後就可以使用手機或筆電，連接相同的無線網路，開啟瀏覽器下達指令了：

讓繼電器通電

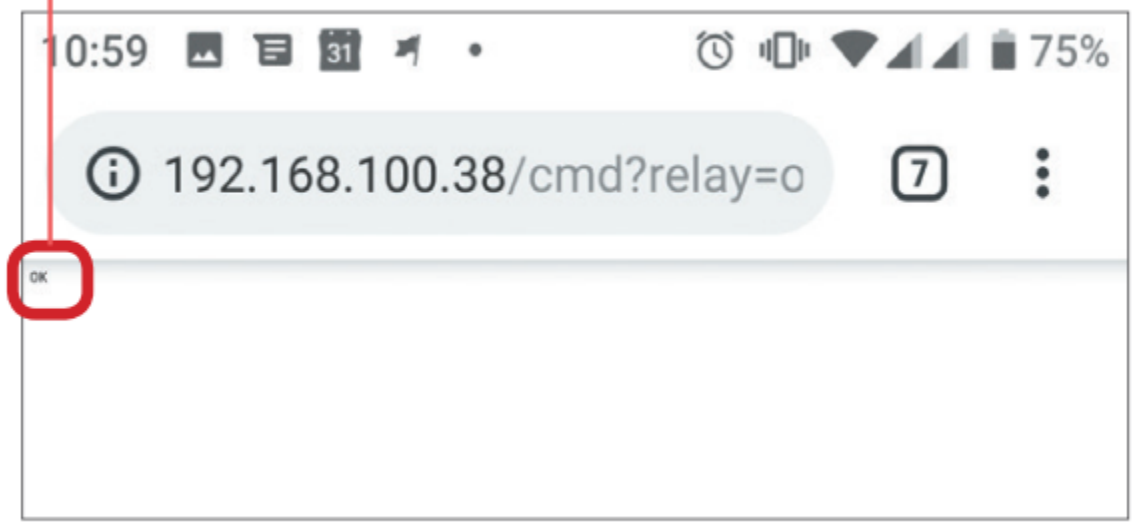


讓繼電器斷電

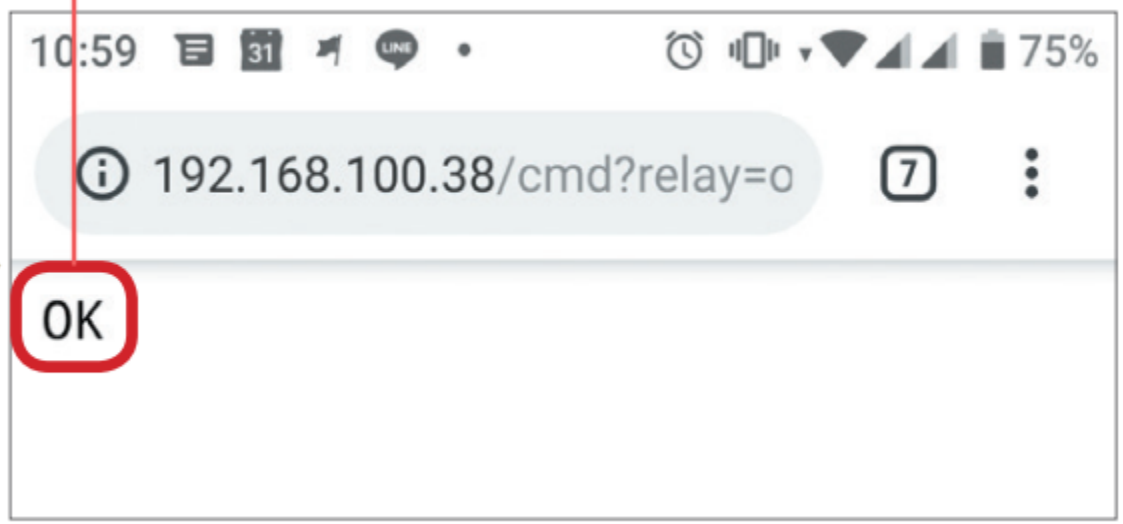


要注意的是，瀏覽器送出指令後，會收到 D1 mini 的回應，瀏覽器會將回應內容當成網頁內容顯示，在手機上顯示的字很小，可以用捏放方式放大即可看到：

回應的 OK 在這裡



放大顯示就可以看到



16 使用 HTML 網頁簡化操作

建立易用的使用介面

前一節的範例雖然可以正確運作，不過下指令還要打一長串的網址，如果能夠提供 HTML 網頁讓使用者直接**點選連結**，就會更容易操作了。

指定回應網頁

在 ESP8266WebServer 模組中，也提供有回傳 HTML 網頁的功能，只要使用以下函式：

```
ESP8266WebServer.setDocPath("/relay")
```

就會把 "/relay" 開頭的指令當成是檔案名稱，將 D1 mini 模組上同名的檔案傳回給瀏覽器。例如，如果輸入以下網址：

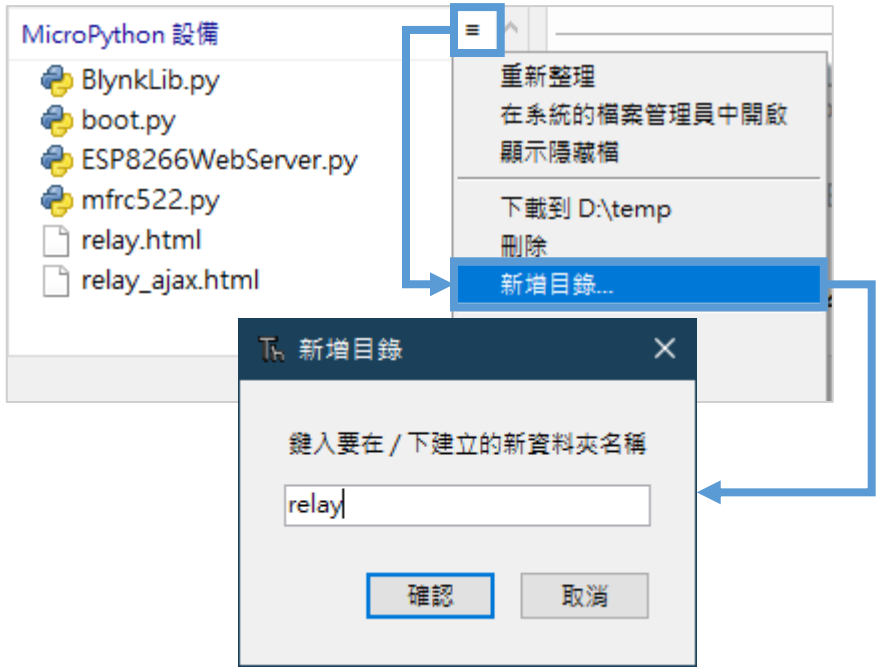
```
http://192.168.100.38/relay.html
```

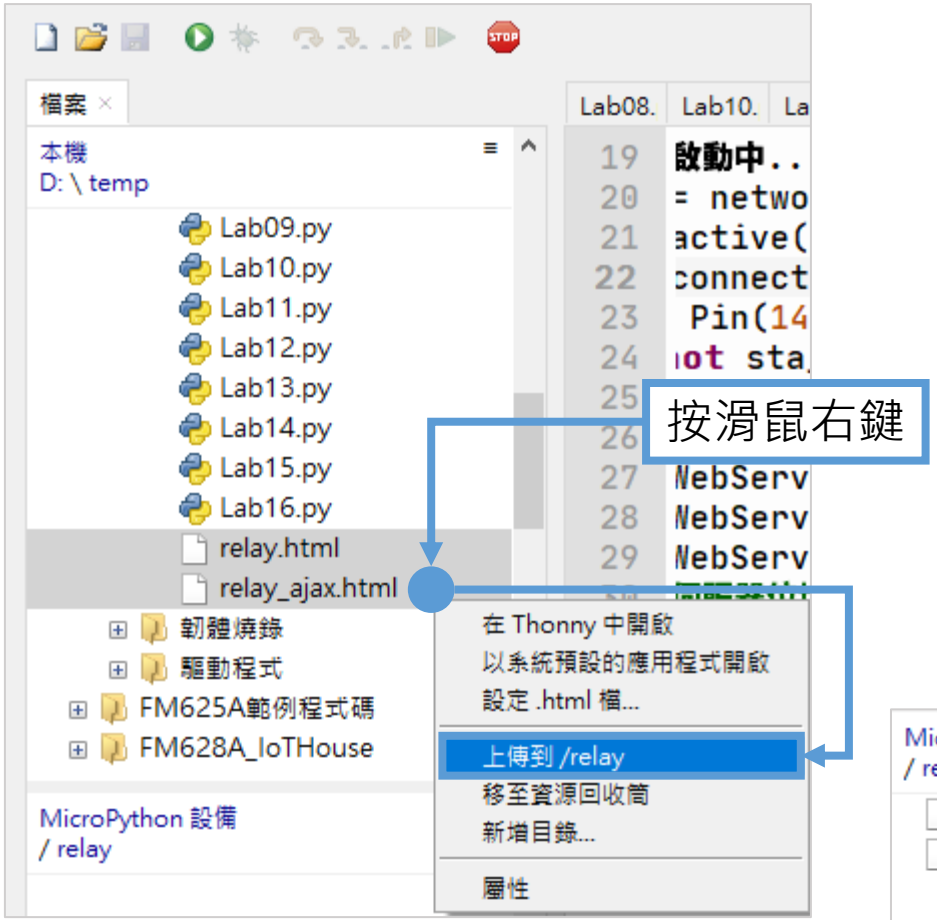
由於指令為 "relay.html"，開頭部分與 setDocPath 中指定的 "/relay" 相同，因此就會把 "/relay.html" 當成是檔案名稱，直接傳回 D1 mini 上現有的 /relay.html 檔案內容給瀏覽器。

上傳任意檔案到 D1 mini

要搭配傳回檔案的功能，我們還必須將檔案上傳到 D1 mini 中，這一樣可以使用 **Device/Upload current script with current name** 功能表指令，不過這個指令只能在沒有程式執行時運作，如果上一個實驗的程式還在執行，請先按 Ctrl + F2 結束程式，然後開啟本書範例資料夾內的 HTML 檔：

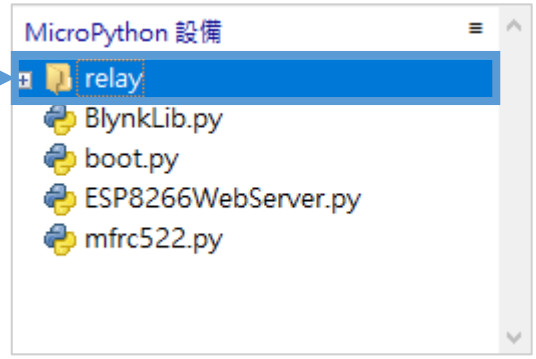
上船網頁檔案到 D1 mini



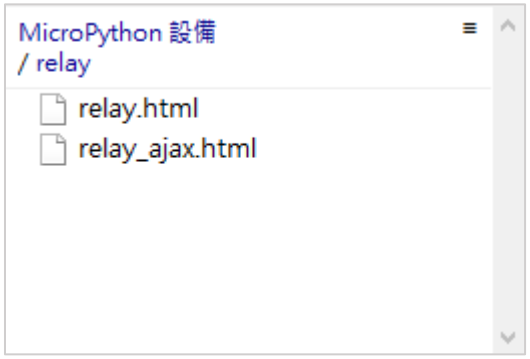


按滑鼠右鍵

雙按切換到
relay 資料夾



確認 relay 資
料夾內有檔案



Lab16

遠端遙控網頁版

實驗目的	利用手機或是筆電上的瀏覽器, 直接**點選網頁上的連結**下達指令遙控家電開關。
材料	• Di mini • 繼電器 × 1 • 低功率家電 (如檯燈) × 1 • 杜邦線及排針若干

設計原理

本實驗會設定網頁路徑，讓 D1 mini 接到以下指令時：

```
http://192.168.100.38/relay.html
```

直接傳回 relay.html 檔給瀏覽器顯示，方便使用者操作。我們已經提供有這個 HTML 檔，內容如下：

```
01 <!DOCTYPE html>
02 <html>
03 <head>
04   <meta charset='UTF-8'>
05   <meta name='viewport'
06     content='width=device-width, initial-scale=1.0'>
07   <title>家電遙控器</title>
08 </head>
09 <body>
10   <h1>
11     <a href='/cmd?relay=on'>通電</a> 或
12     <a href='/cmd?relay=off'>斷電</a></h1>
13 </body>
14 </html>
```

程式設計

```
01 import network
02 import ESP8266WebServer                    # 匯入網站模組
03 from machine import Pin
04
05 def handleCmd(socket, args):               # 處理 /cmd 指令的函式
06     if 'relay' in args:                    # 檢查是否有 relay 參數
07         if args['relay'] == 'on':          # 若 relay 參數值為 'on'
08             relay.value(1)                 # 讓繼電器通電
09         elif args['relay'] == 'off': # 若 relay 參數值為 'off'
10             relay.value(0)                 # 讓繼電器斷電
11         ESP8266WebServer.ok(socket, "200", "OK") # 回應 OK
12     else:
13         ESP8266WebServer.err(socket, "400", "ERR")# 回應 ERR
```

```
14
15 print("啟動中...")
16 sta_if = network.WLAN(network.STA_IF)      # 取得無線網路介面
17 sta_if.active(True)                        # 啟用無線網路
18 sta_if.connect('無線網路名稱', '密碼')      # 連結無線網路
19 relay = Pin(14, Pin.OUT, value=0)          # 控制 D5 腳位
20 while not sta_if.isconnected():            # 等待無線網路連上
21     pass
22
23 ESP8266WebServer.begin(80)                  # 啟用網站
24 ESP8266WebServer.onPath("/cmd", handleCmd)# 指定處理函式
25 ESP8266WebServer.setDocPath("/relay")      # 指定 HTML 檔路徑
26 print("伺服器位址：" + sta_if.ifconfig()[0])# 顯示網站的 IP 位址
27
28 while True:
29     ESP8266WebServer.handleClient()   # 持續檢查是否收到新指令
```

要測試程式之前，請參見 8-2 節先將 relay.html 檔案上傳到 D1 mini。執行程式看到網站的 IP 位址 (以下仍假設為 192.168.100.38) 後，即可開啟瀏覽器如下操作：