Raspberry Pi

互動科技藝術與感測設計

劉士達

課程大綱

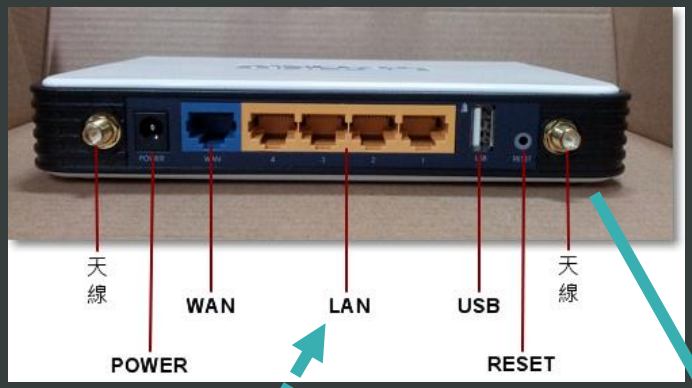
- **1　電子材料清單介紹、認識Raspberry Pi 3 硬體、安裝Raspbian系統、GPIO控制(LED元件、按鈕元件)**
- 2　感測器模組與Raspberry Pi連接(繼電器、溫度、超音波距離感測器)
- 3　攝影機模組、SimpleCV影像處理
- 4　網路連線、伺服器架設、遠距馬達控制、距離感測器
- 5　物聯網、文字轉語音、Arduino應用開發
- 6　音樂版蘑菇總動員(1) 範例製作
- 7　語彙版蘑菇總動員(2) 範例製作

課前環境設定

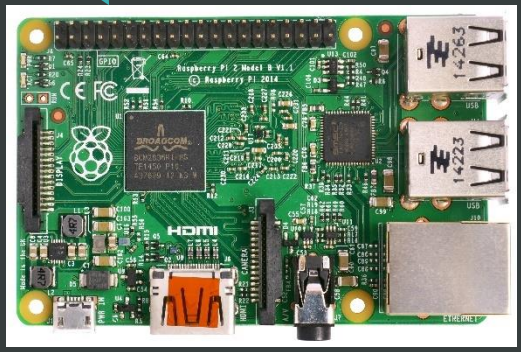
- 無線AP帳號/密碼：
  - SSID = RaspberryPi_AP
  - 密碼 = raspberry
- 預設的樹梅派帳號/密碼：
  - 帳號：pi
  - 密碼：raspberry
- IP位置查找：
  - Advanced IP Scanner
- 有線網路連接：
  - 拿出2M長的網路線連接至Hub(用此方法比較穩定)

連線到Rpi的方式-1

- 有線網路/無線網路
- 透過VNC/SSH連線
- 透過Serial Port方式



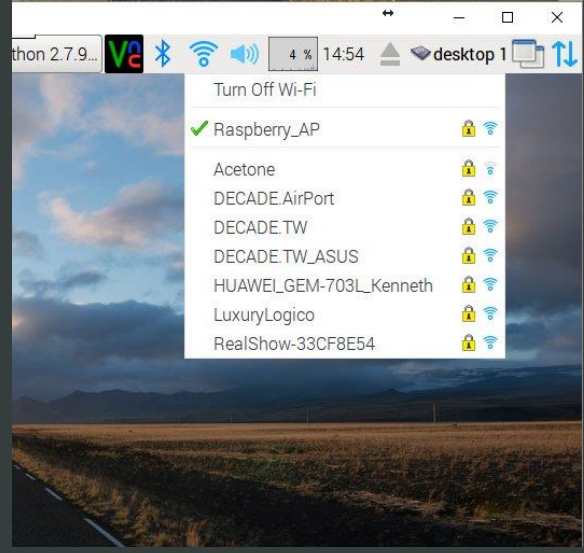
無線連接 到無線AP



有線連接後得到 IP:192.168.3.x

連線到Rpi的方式-2

- 打開手機分享熱點功能
- iOS預設的ip位置：
  - 172.20.10.0~15 [共16個]
- Android預設的ip位置：
  - 各家廠牌不同



用螢幕看到想要連線的手機熱點分享名稱點選之後連線

查詢目前Rpi的IP

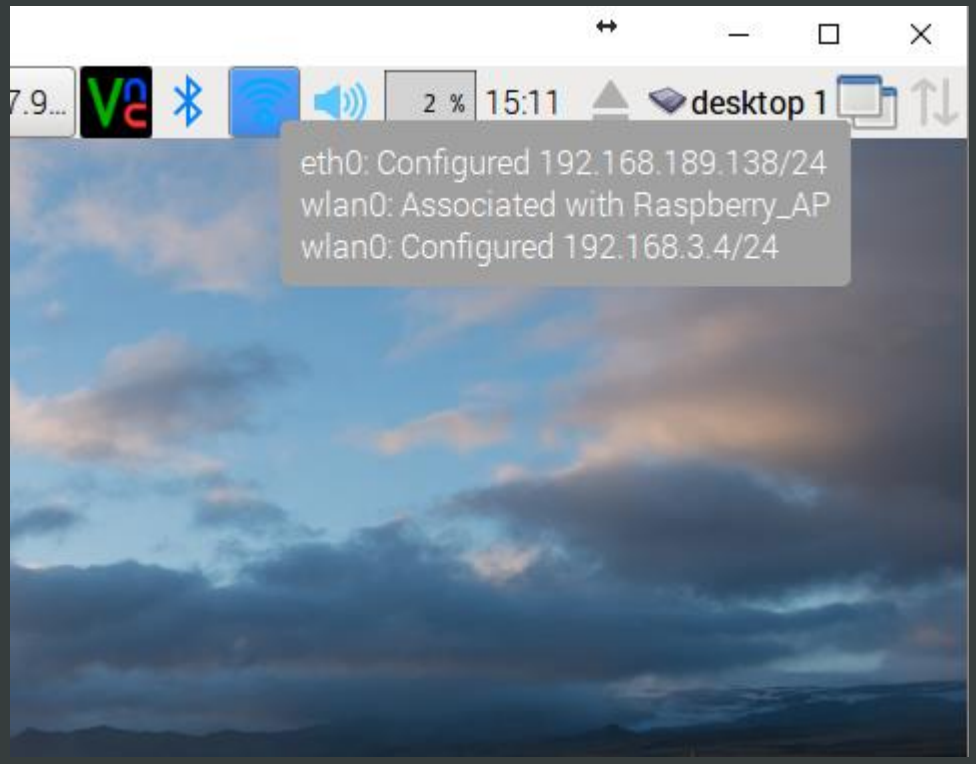
- 將滑鼠移到右上角的無線網路符號

上停留一秒左右，會跳出兩個訊息

eth0 192.168.189.138 這是有線網路得到IP

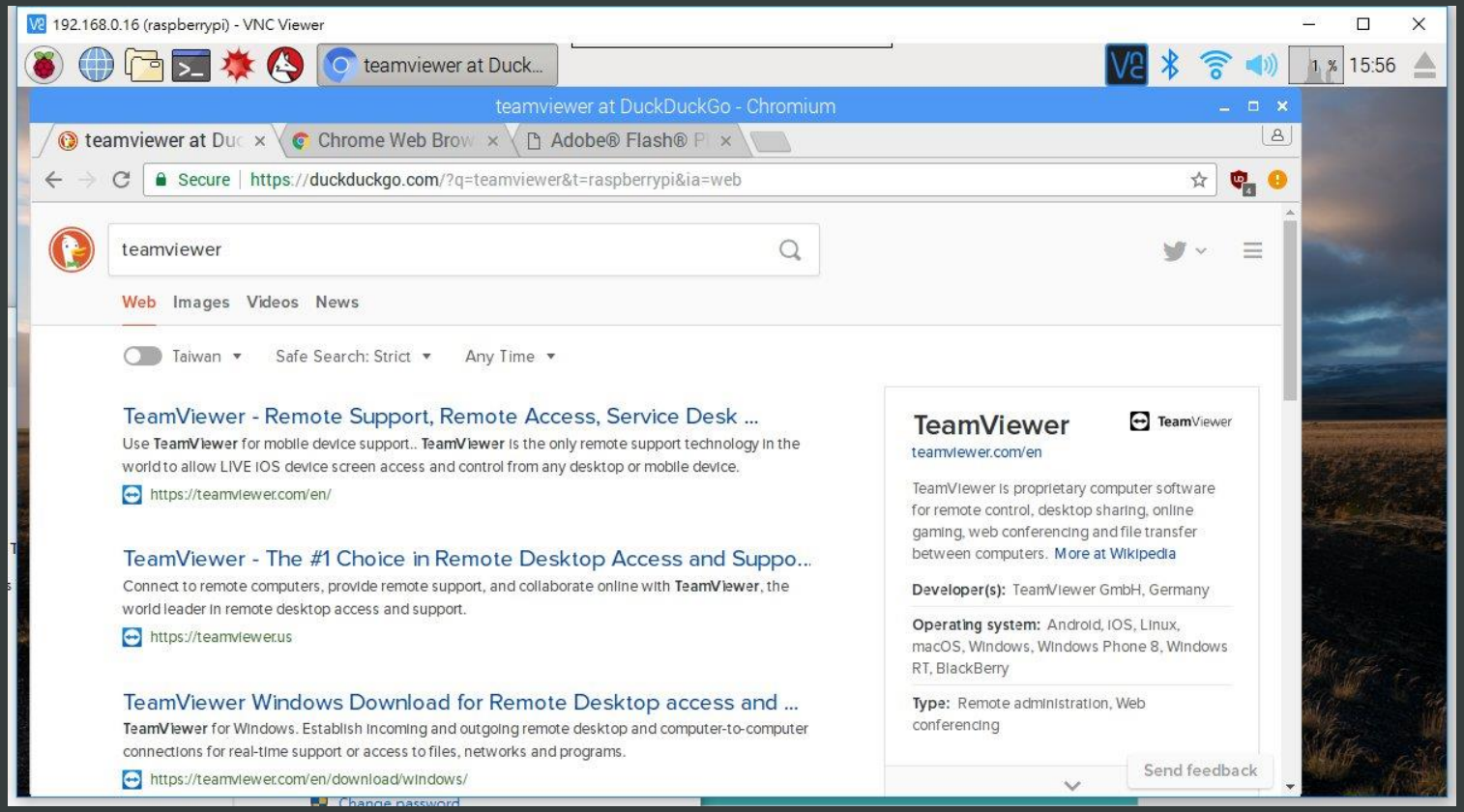
wlan0 192.168.3.4 這是無線網路得到IP

- 以上兩個都可以透過VNC進行連線



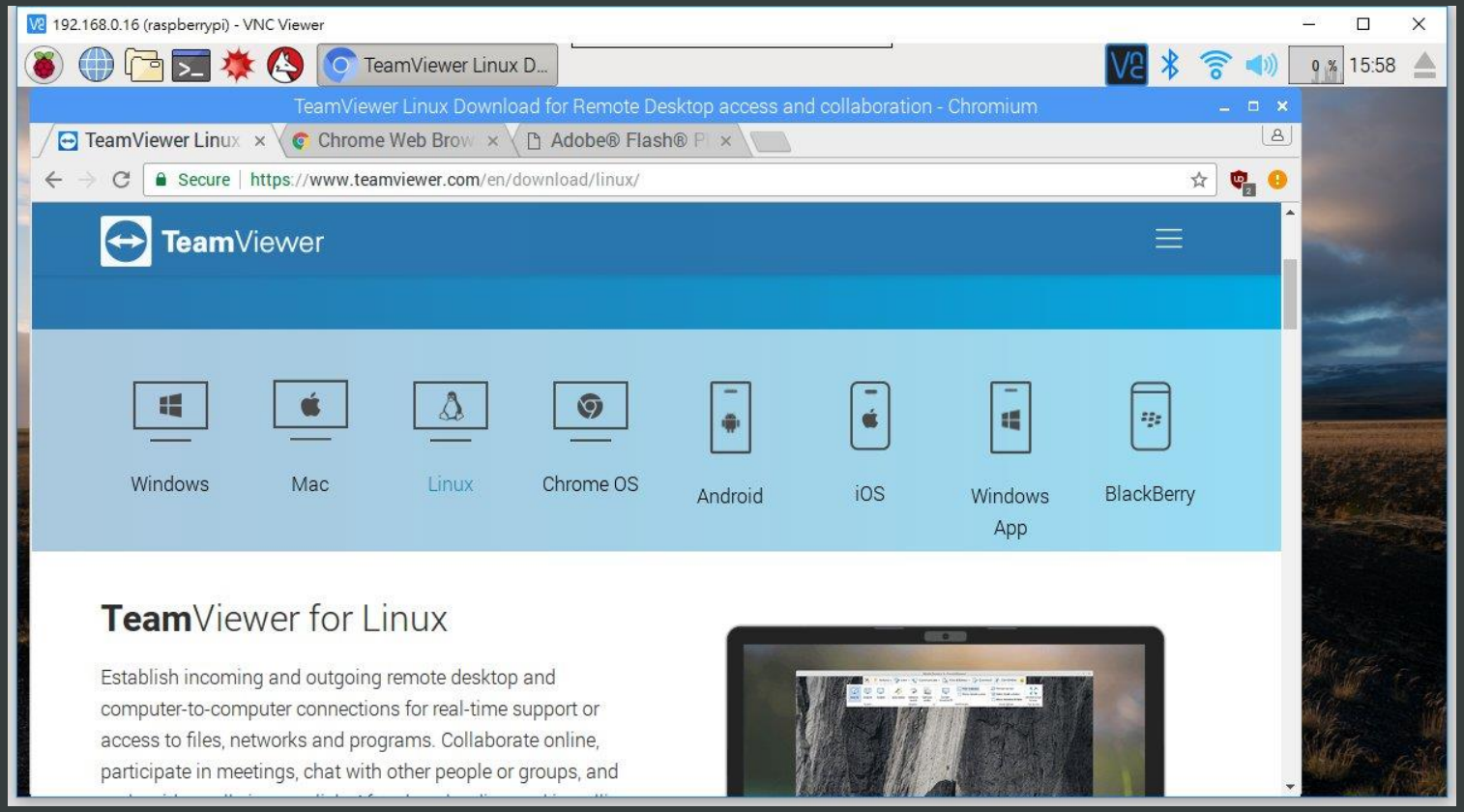
安裝TeamViewer進行遠端連線

在Rpi內打開瀏覽器連上Teamviewer網站

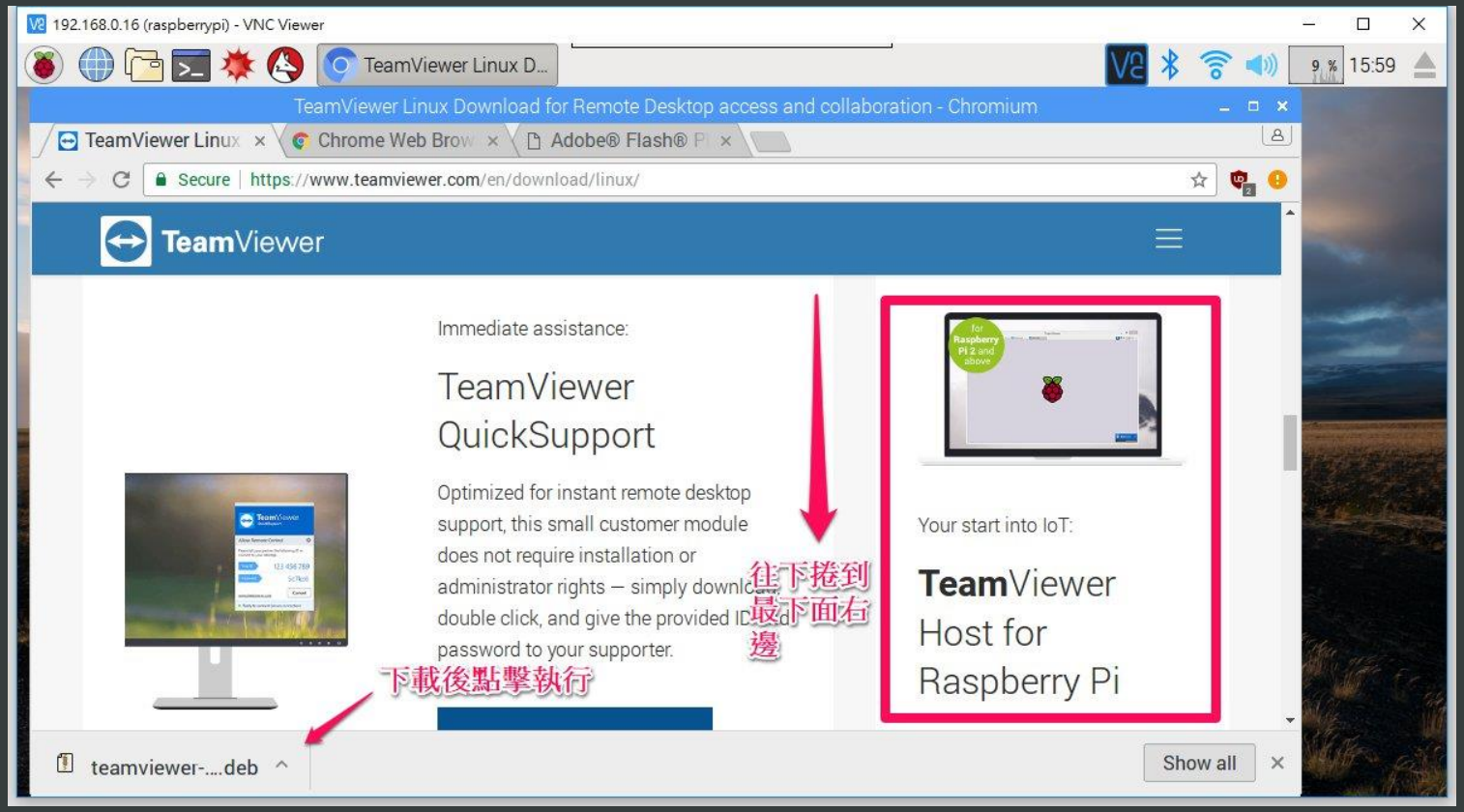


http://teamviewer.com/en/

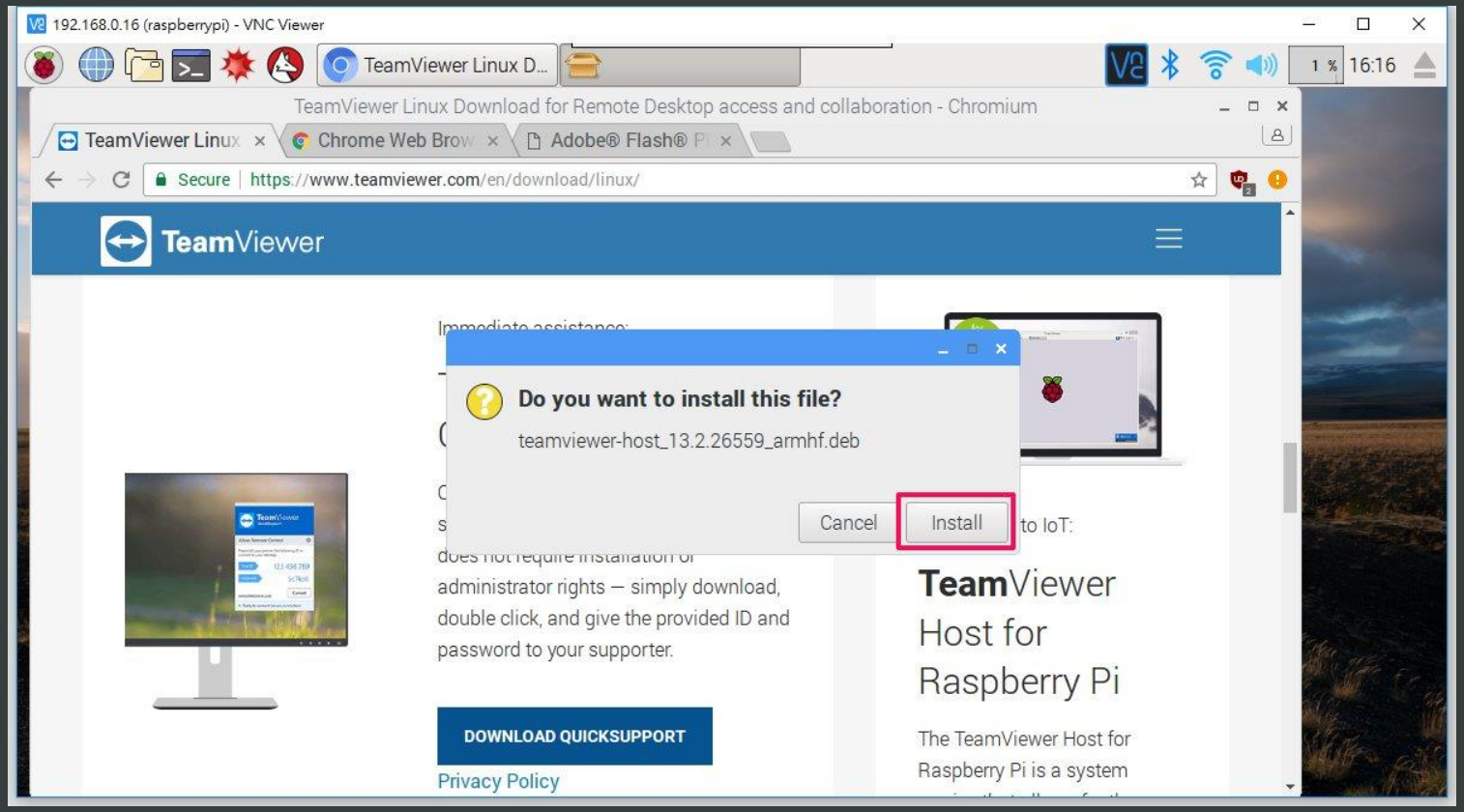
選擇Linux系統版本的Teamviewer



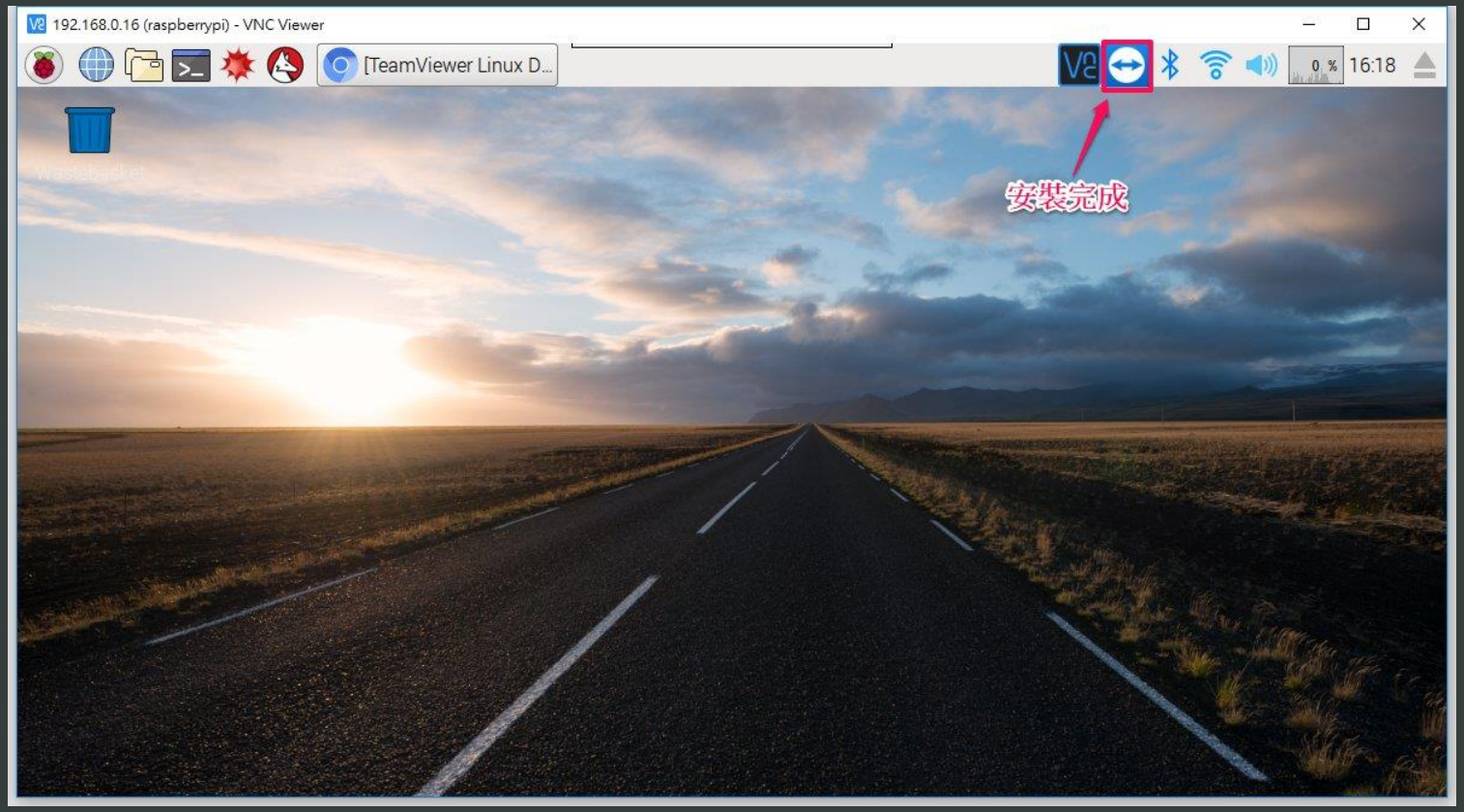
下載TeamViewer Host for Raspberry Pi



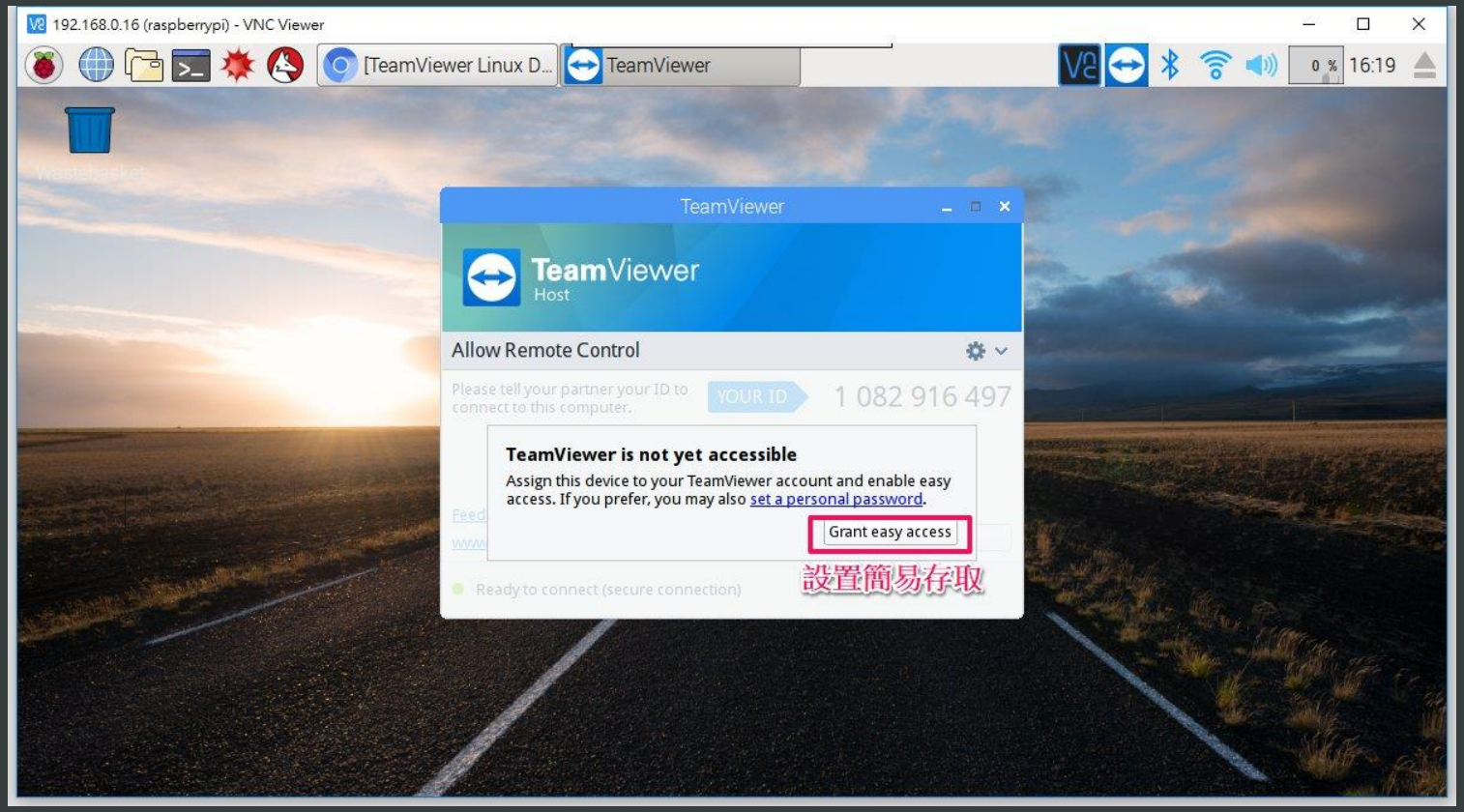
安裝deb


192.168.0.16 (raspberrypi) - VNC Viewer
TeamViewer Linux D...
16:16
TeamViewer Linux Download for Remote Desktop access and collaboration - Chromium
TeamViewer Linux
Chrome Web Brow
Adobe® Flash® P
Secure | https://www.teamviewer.com/en/download/linux/
TeamViewer
Do you want to install this file?
teamviewer-host_13.2.26559_armhf.deb
Cancel
Install
administrator rights — simply download, double click, and give the provided ID and password to your supporter.
DOWNLOAD QUICKSUPPORT
Privacy Policy
to IoT:
TeamViewer Host for Raspberry Pi
The TeamViewer Host for Raspberry Pi is a system

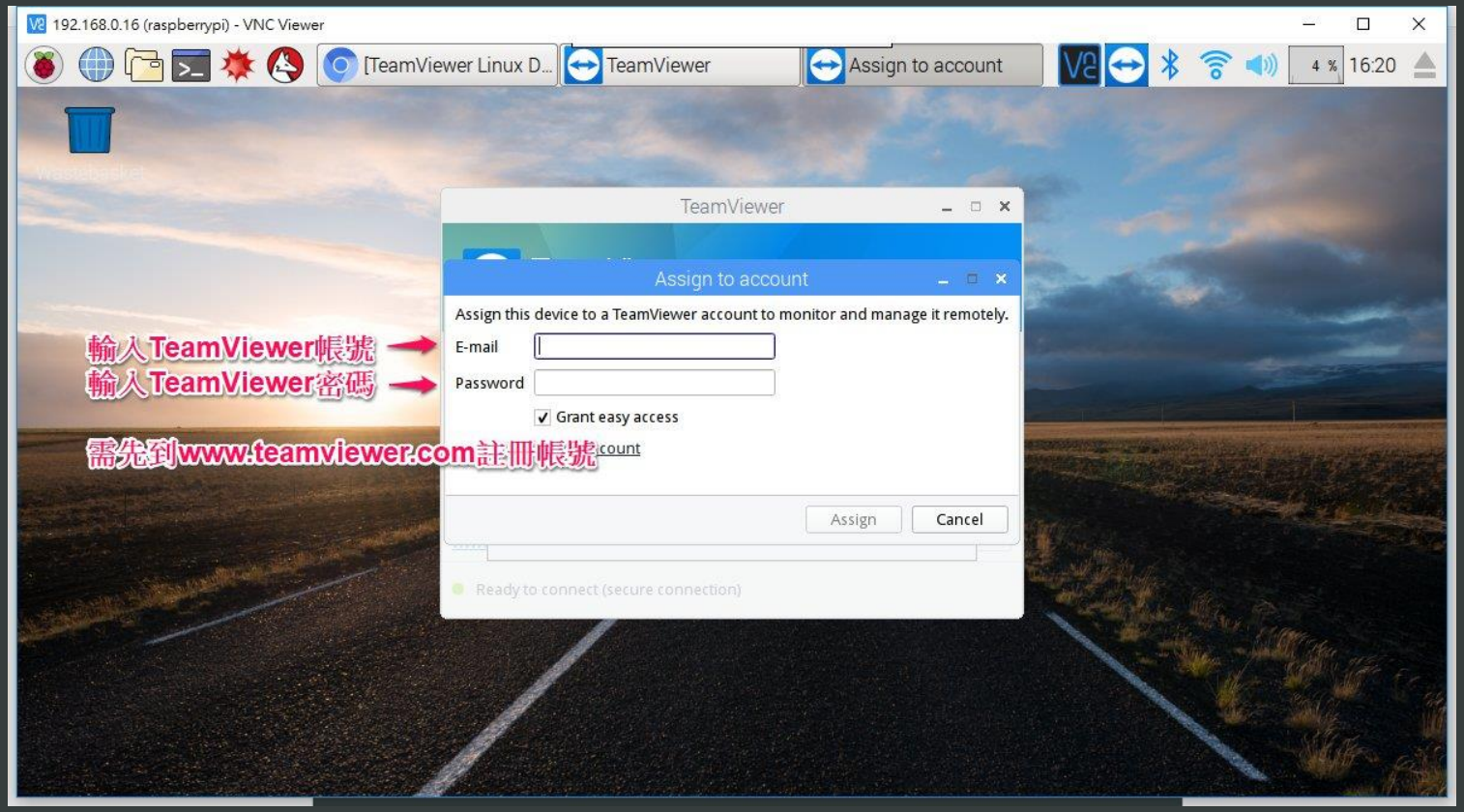
安裝完畢會出現遠端介面



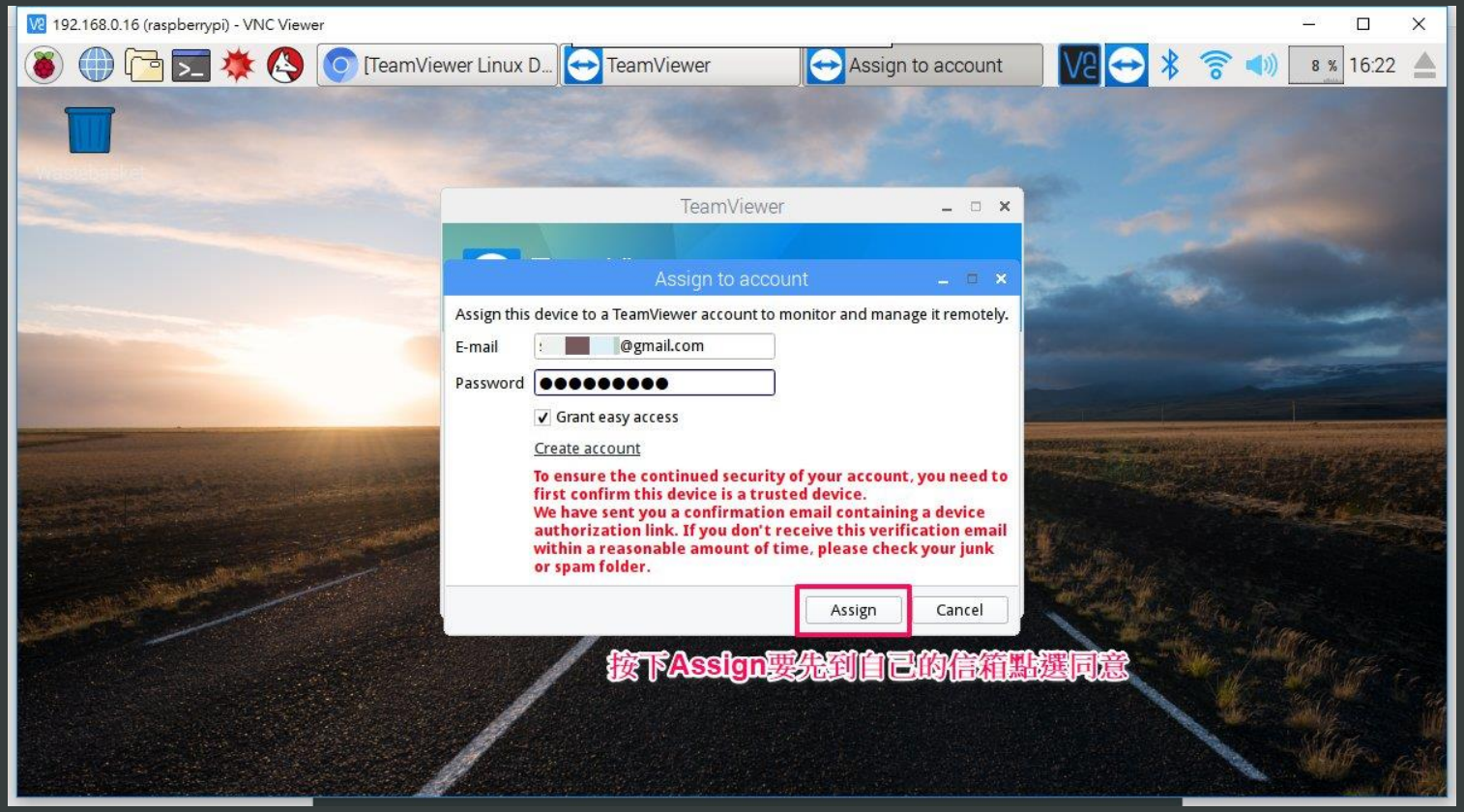
點選右上角的圖標設定簡易存取



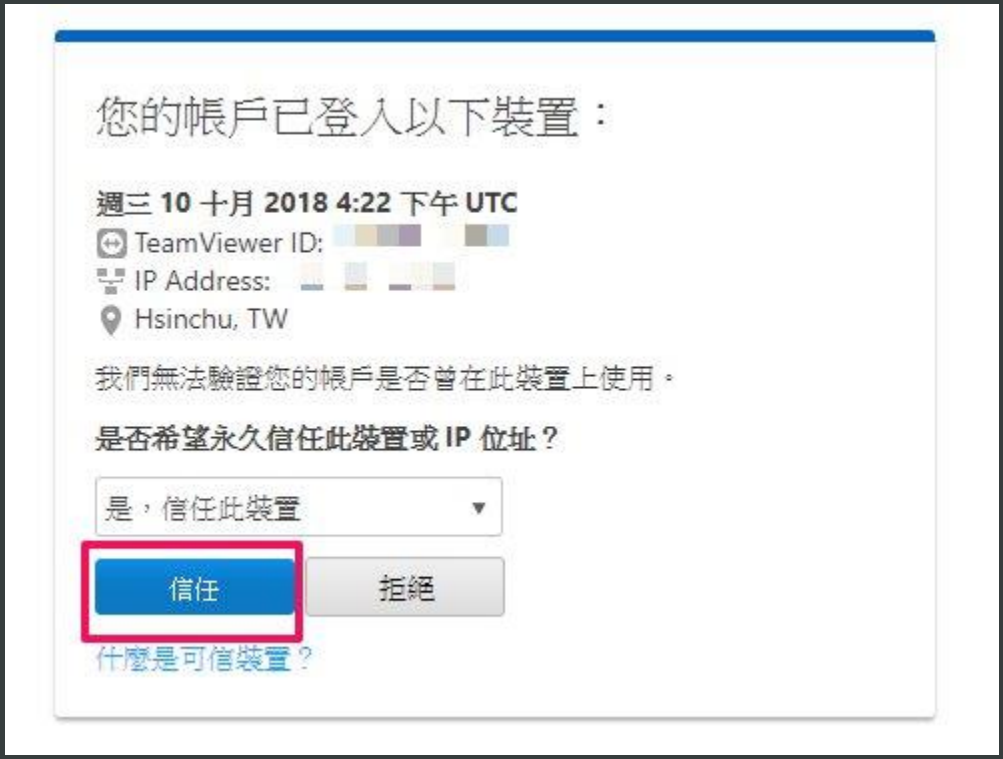
註冊teamviewer.com帳號/登入



設置簡易存取


192.168.0.16 (raspberrypi) - VNC Viewer
[TeamViewer Linux D...
TeamViewer
Assign to account
8 %
16:22
TeamViewer
Assign to account
Assign this device to a TeamViewer account to monitor and manage it remotely.
E-mail
@gmail.com
Password
Grant easy access
Create account
To ensure the continued security of your account, you need to first confirm this device is a trusted device.
We have sent you a confirmation email containing a device authorization link. If you don't receive this verification email within a reasonable amount of time, please check your junk or spam folder.
Assign
Cancel
按下Assign要先到自己的信箱點選同意

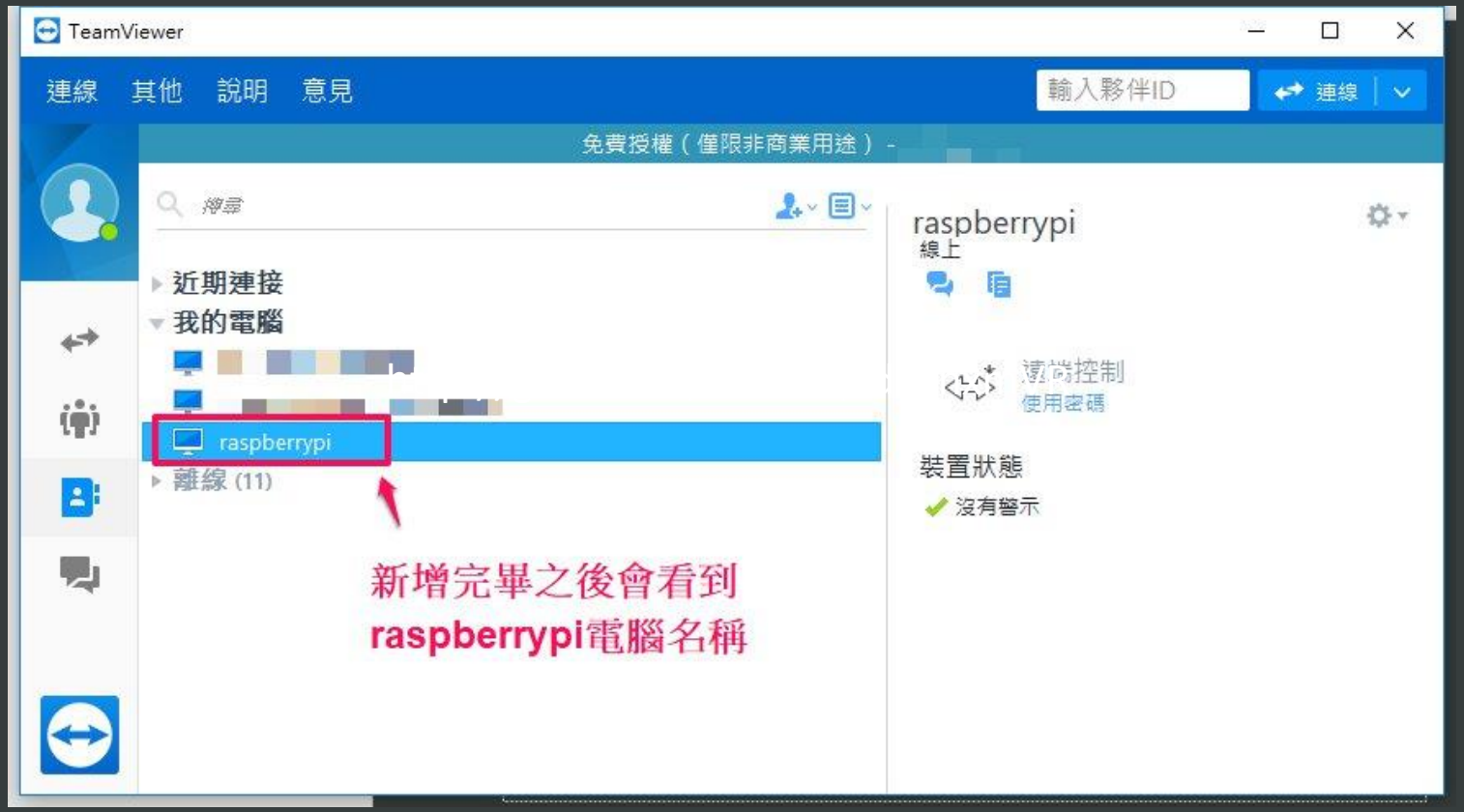
設置信任該裝置


您的帳戶已登入以下裝置：
週三 10 十月 2018 4:22 下午 UTC
TeamViewer ID:
IP Address:
Hsinchu, TW
我們無法驗證您的帳戶是否曾在此裝置上使用。
是否希望永久信任此裝置或 IP 位址？
是，信任此裝置
信任
拒絕
什麼是可信裝置？

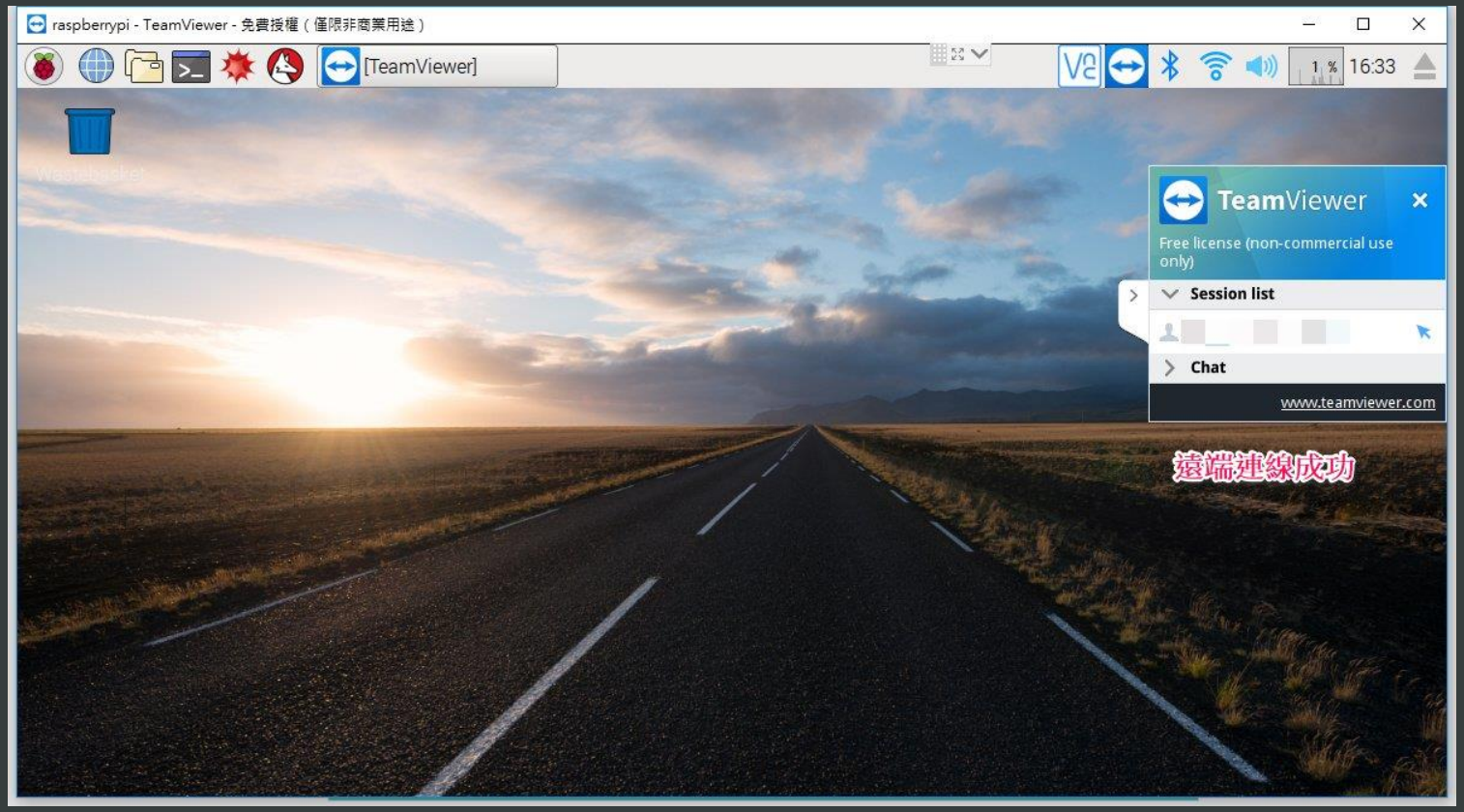
確認已經連上網際網路


TeamViewer
TeamViewer
Host
Allow Remote Control
Please tell your partner your ID to connect to this computer.
YOUR ID
1 08 497
Feedback
www.teamviewer.com
OK
Ready to connect (secure connection)

再自己的電腦上安裝TeamViewer Client端



遠端連入成功!


raspberrypi - TeamViewer - 免費授權（僅限非商業用途）
[TeamViewer]
16:33
TeamViewer
Free license (non-commercial use only)
Session list
Chat
www.teamviewer.com
遠端連線成功

Python語言

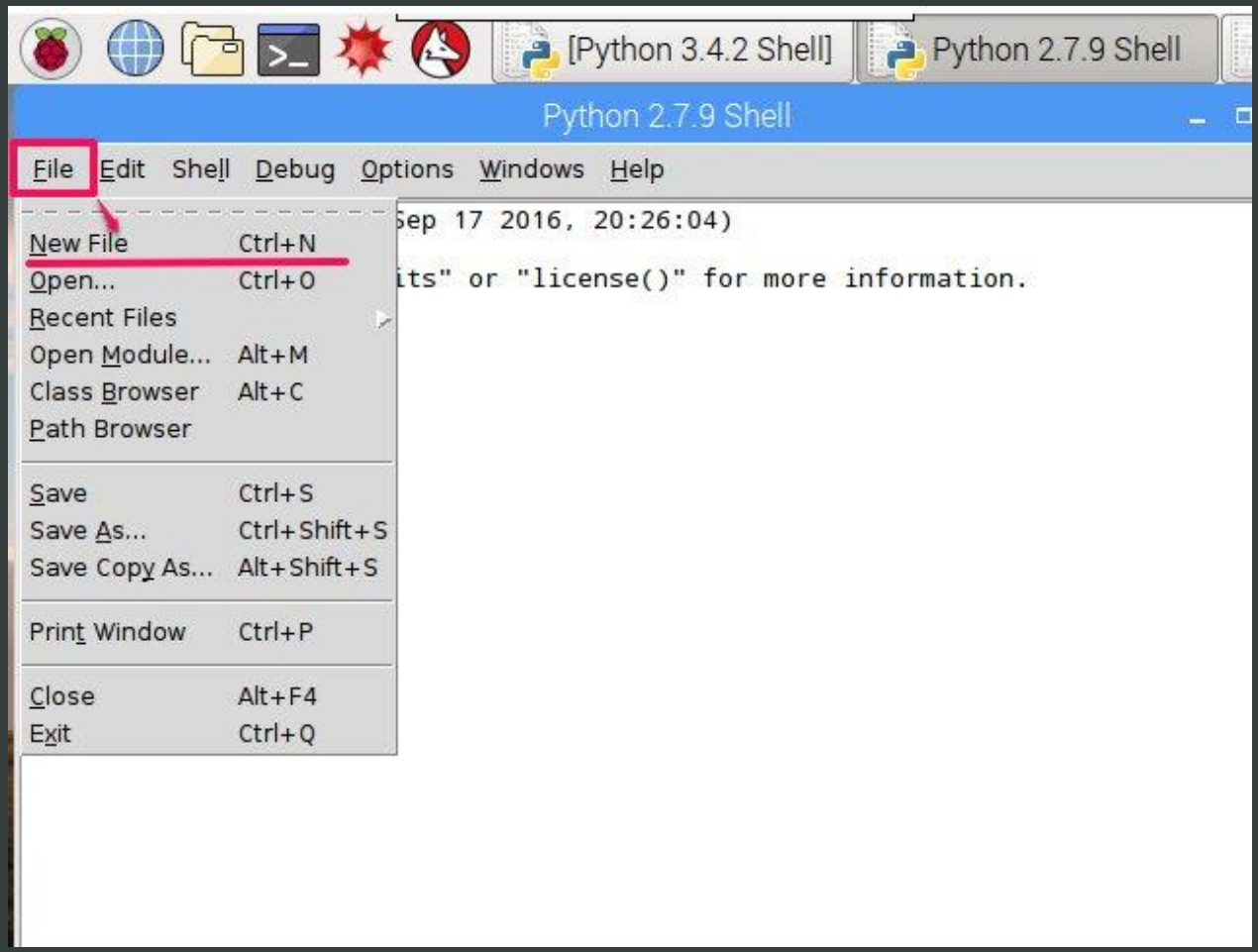
- 1991年發展至今
- 物件導向，指令式編程，函數語言程式設計
- 語法不使用 { } 做程式語言區塊
- 撰寫Python時使用縮進(Tab)方式
- Google, NASA, … 等公司政府單位大量使用Python語言
- 變數不須宣告定義型別，結尾不用 ; 不使用小括弧
- 多平台支援，MAC內建，Windows，Linux都支援
- 大量的教學與網站資源

Arduino VS Python 語言

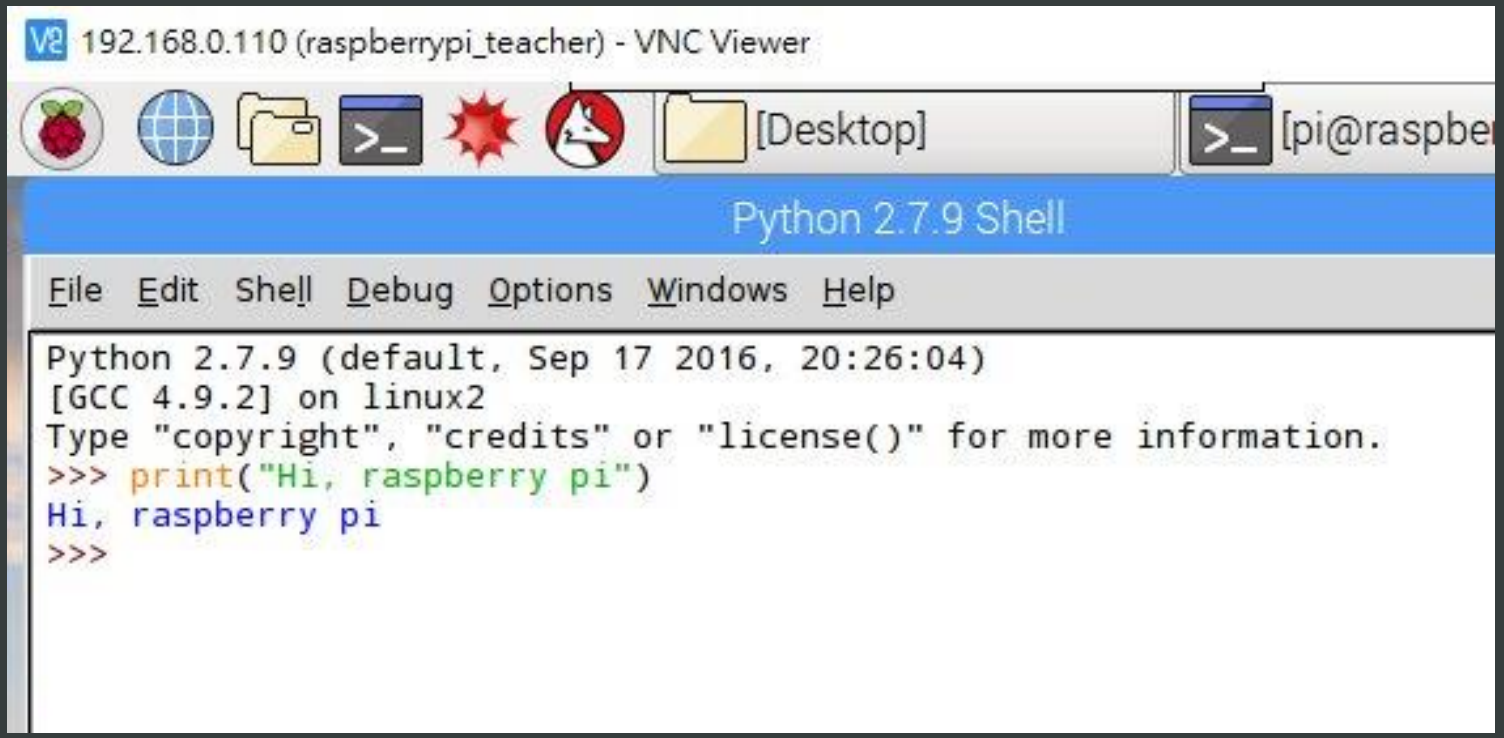
```
int pin = 13;

void setup() {
  pinMode(pin,OUTPUT);
}

void loop() {
  digitalWrite(pin,HIGH);
  delay(1000);
  digitalWrite(pin,LOW);
  delay(1000);
}
```

```
import RPi.GPIO as GPIO
import time
led=13
GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)

while True:
    GPIO.output(led, True)
    time.sleep(1)
    GPIO.output(led, False)
    time.sleep(1)
```

Python Shell

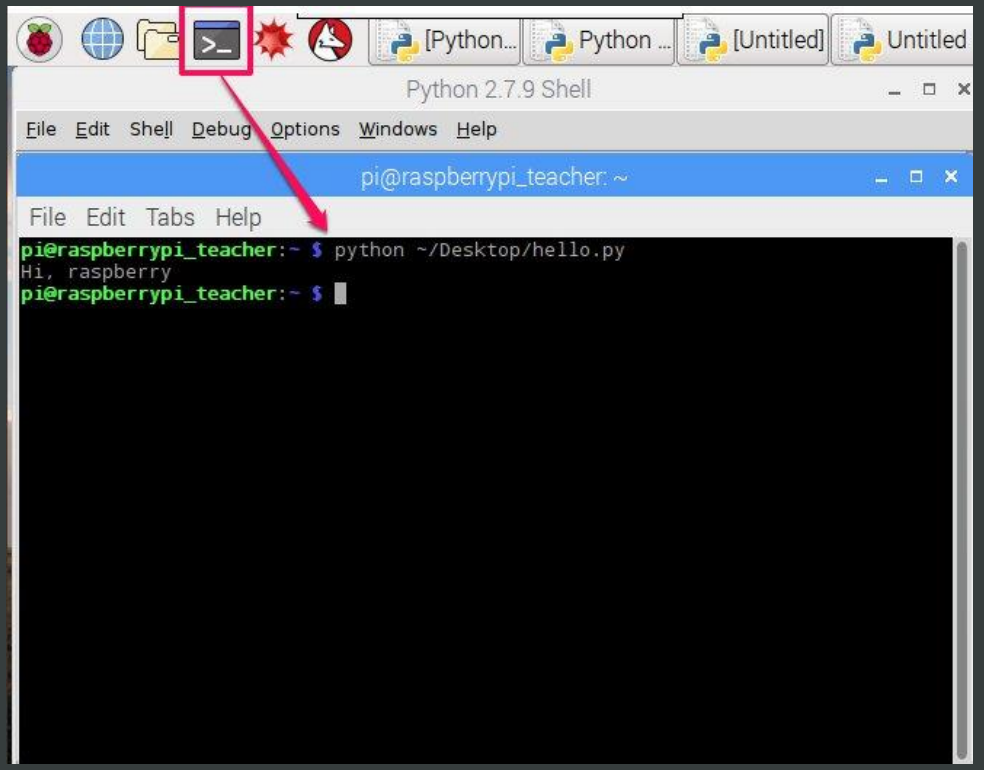


在Shell內直接輸入程式碼

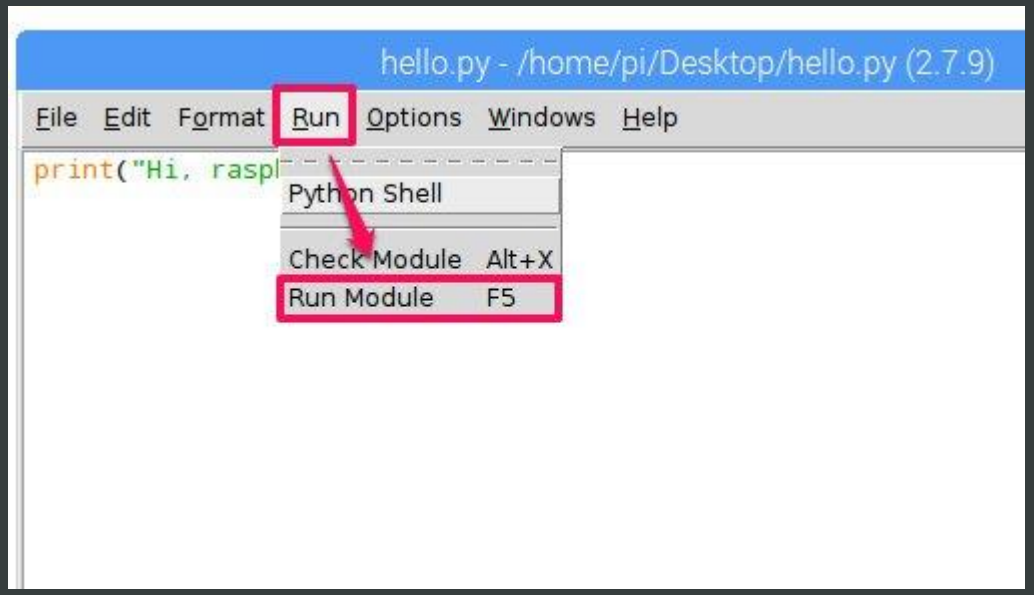


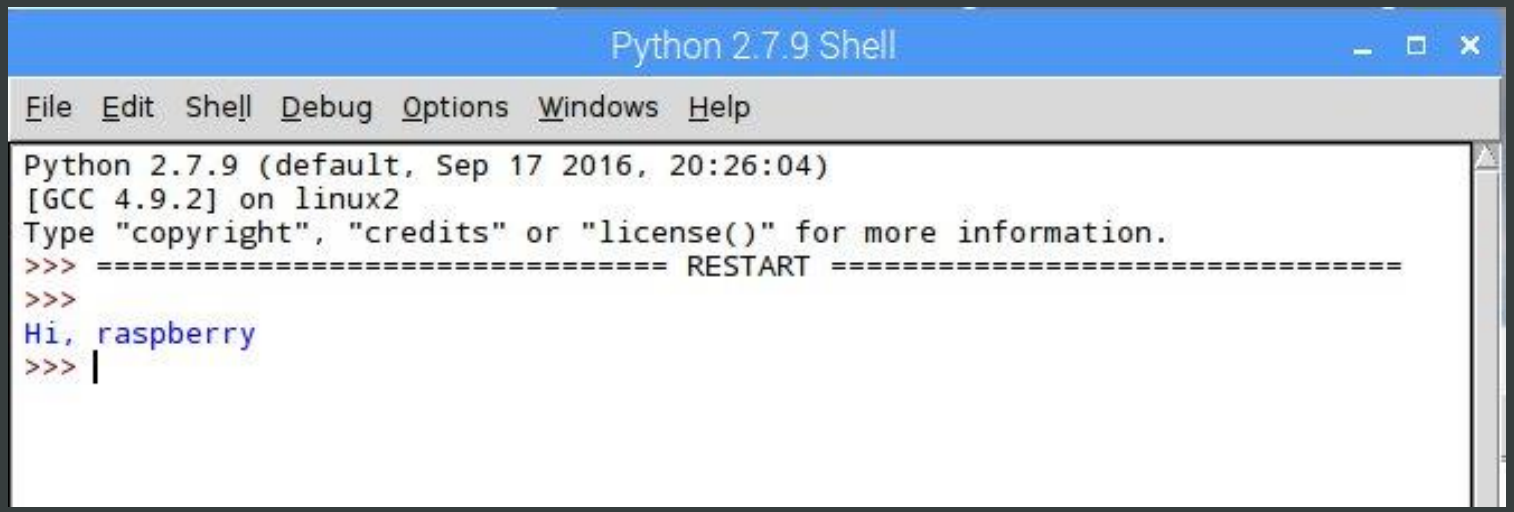
執行程式碼

- 執行撰寫好的 *.py 檔的方法有兩種
  - 使用編輯器內的Run -> Run Module 進行編譯
  - 打開終端機 畫面輸入
    `python ~/Desktop/hello.py`
  - 輸入之後立即會看到輸出的結果
    - `Hi, raspberry`



Python IDLE 內執行Run module





Python的縮進分層

Python語言 – 用冒號代表條件式的開始

```
if (a == true){
   print (“ok”);
 }
```

C語言 – 用3行表達

```
if a == True:
    print (“ok”)
```

Python語言 – 原本大括弧內的語法用四個空白鍵或Tab取代，代碼簡潔

```
if (a == true){print (“ok”); }
```

C語言 – 不做換行，難以辨識

Python快速入門

- 變數
  - 字串
  - 數值
  - 型別轉換
- 陣列
  - list
  - dict
- 邏輯
  - If … elif …else
  - 邏輯比較
  - 運算子
- 迴圈
  - for
  - range()
  - while
- 自訂函式
  - def
- 輸入/輸出
  - print
  - import

Python的字串

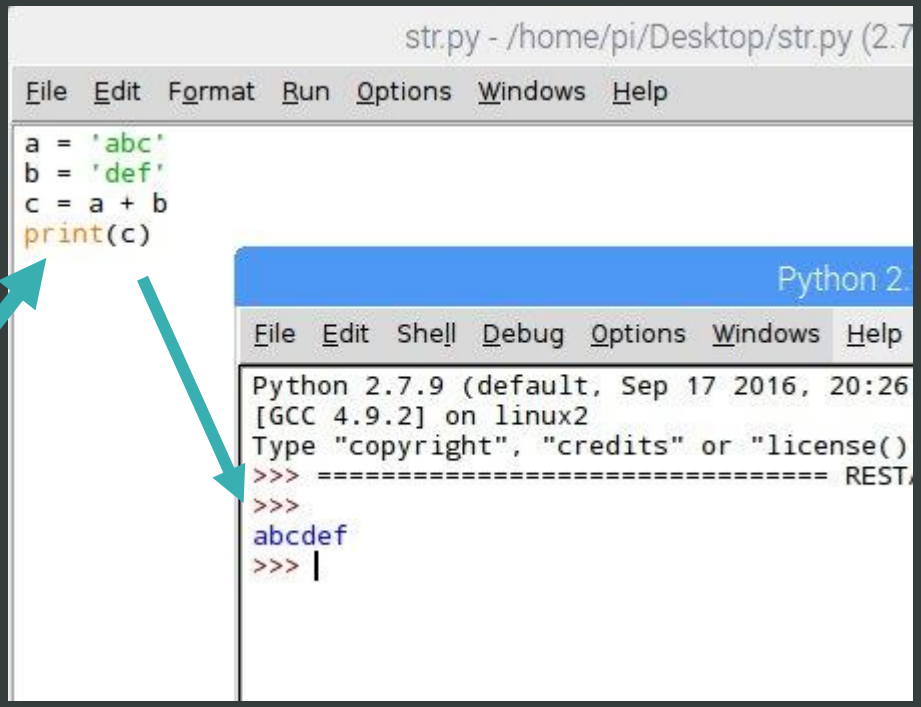
- 字串型別

```
A = “abc”
B = ‘abc’
C = “樹莓派”
D = ‘’’樹莓派 ‘’ ’
E = “””樹莓派 “” ”
```

- 字串相加

```
A = ‘abc’
B = ‘def’
C = A + B
print(C)
```

str.py



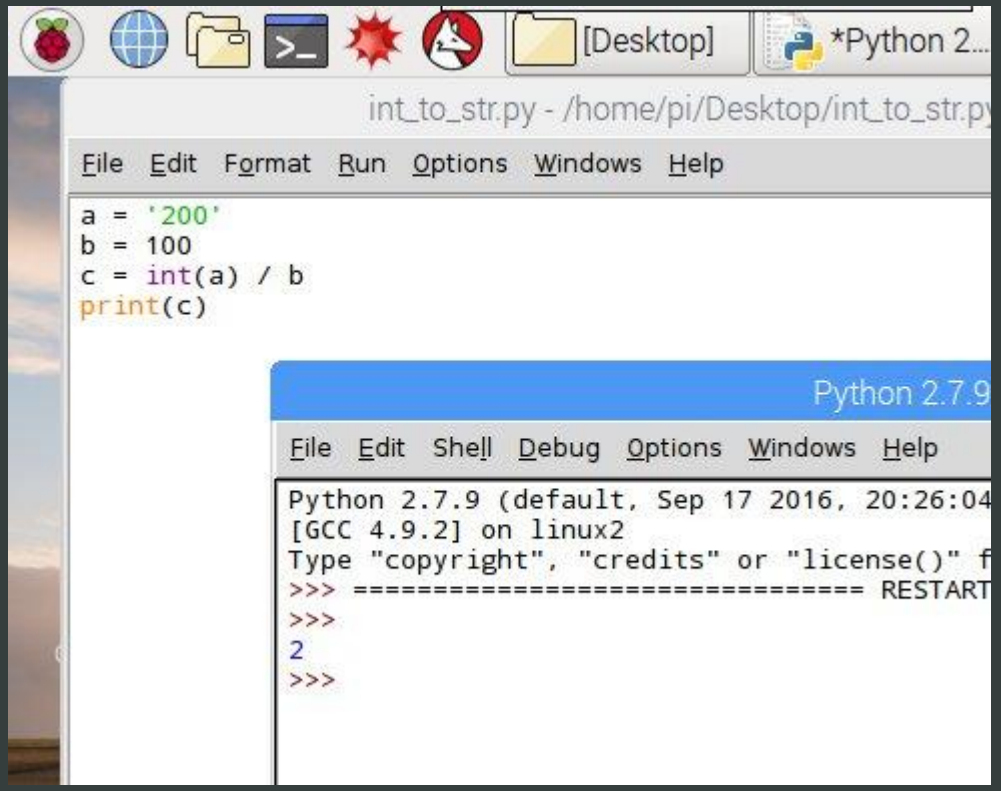
Python的數值

- Int 整數
  - 正負範圍64bit數值，$-10^{64} \sim 10^{64}$
- Float 浮點數
  - 類似C語言的double雙精度浮點數，最大為64bit，0.0000000000001 [64位元長]
- 布林變數
  - True 或 False / 1 或 0 ，True 跟 False要大寫開頭
- 長整數
  - 無限大，超過64bit數值會自動模擬，998293829138219384985912L，L代表長整數

Python的型別轉換

```
a = “200”
b = 100
c = int(a) / b
print(c )
```

Int_to_str.py



常用的Python型別轉換

- `int( )` 轉成整數
- `str( )` 轉成字串
- `float( )` 轉成浮點數
- `ord( )` 轉成ascii碼 `ord('a') => 97`
- `chr( )` 轉成字元 `chr(97) => a`
- ex：

  `int('100')` => 100 [數值的100]

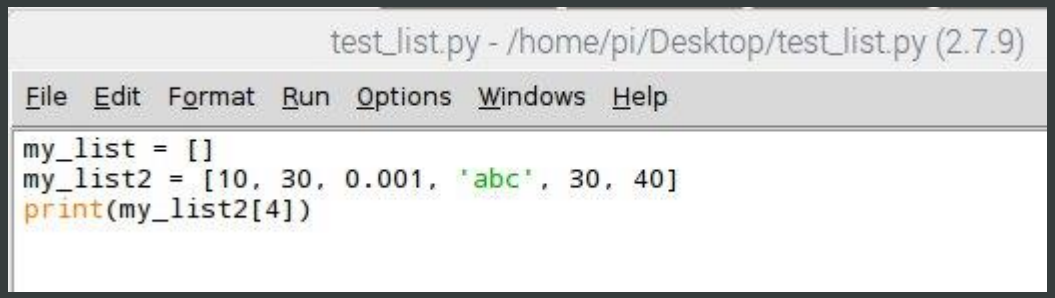
  `str('500')` => 500 [字串的500]

```
>>> print(ord('a'))
97
>>> print(chr(97))
a
>>>
```

Python的list (列表)，不須定義，可以混和數據

```
my_list = []      #宣告一個空的list
my_list2 = [10, 30, 0.001, ‘abc’, 30, 40]
print(my_list2[4])
```

Test_list_empty.py

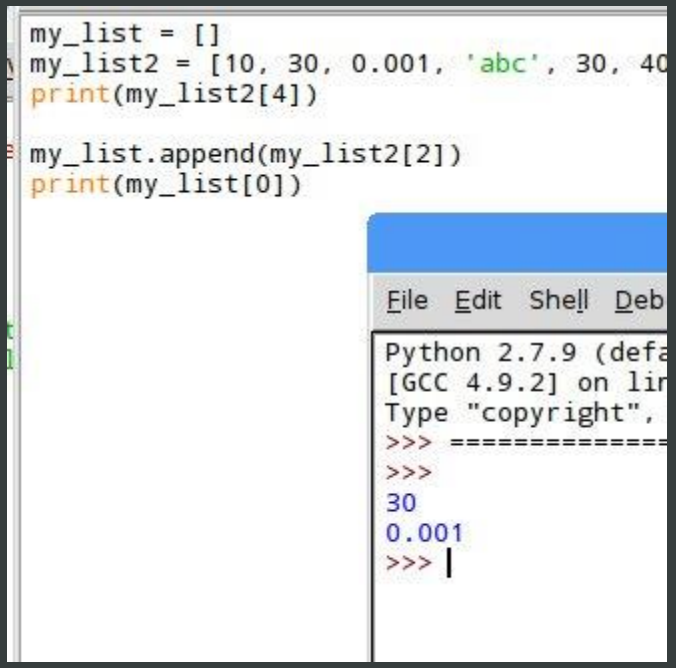


List的一些操作函數

- `List.append(x)` #列表內追加一個x元素
- `List.count(x)` #列表內的x元素出現次數
- `List.index(x)` #列表內x元素的序號
- `List.remove(x)` #列表內x元素刪除
- `List.insert(index, x)` #列表插入x元素

```
my_list = []
my_list2 = [10, 30, 0.001, ‘abc’, 30, 40]
print(my_list2[4])
my_list.append(my_list2[2])
Print(my_list[0])
```

Test_list.py

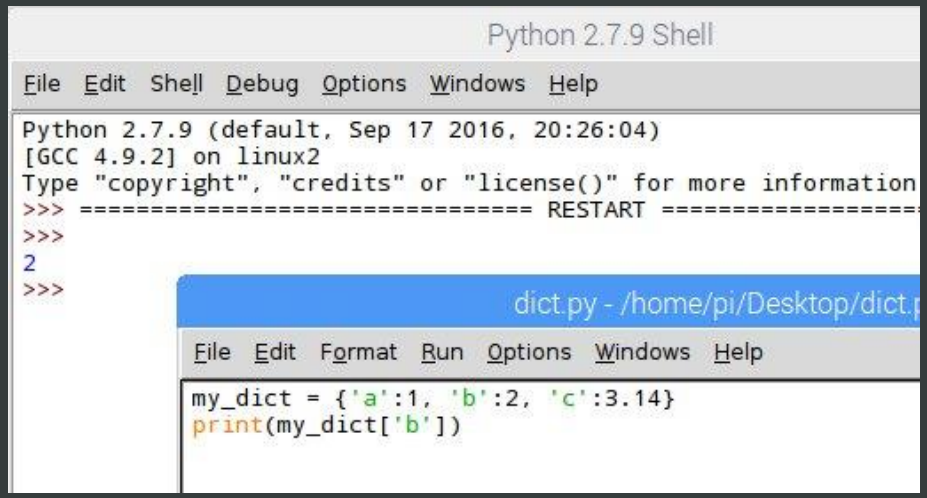


Python的dict (字典) , 用索引方式操作

- 字典是Python的特殊數值類型，每個字典中的成員用 鍵 : 值 成對表示
- 字典的數據表示用大括弧 { } 包住每個 鍵 : 值 內容的數值集合

```
my_dict = {‘a’:1, ‘b’:2, ‘c’:3.14}      #宣告一個字典
Print(my_dict[‘b’])                      #調用字典內的’b’鍵的值
```

dict.py

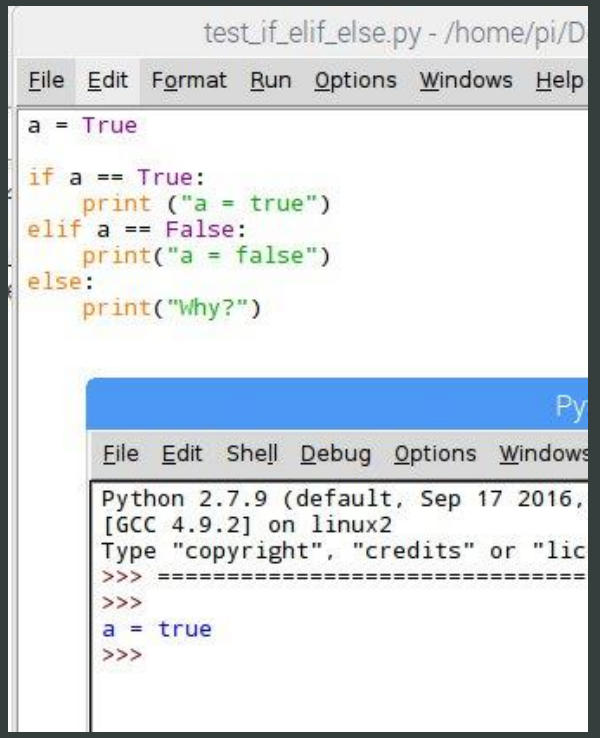


Python的邏輯判斷 if

- Python的程式碼使用Tab或4個空白進行程式碼分區，因此if邏輯判斷的使用需按照此規格進行程式碼撰寫。

```
a = True

if a == True:
    print (“a = true”)
elif a == False:
    print(“a = false”)
else:
    print(“Why?”)
```

Test_if_elif_else.py



巢狀式if … elif … else

```
if 條件1 :
    if 條件3 :
        <程式碼>
    elif 條件4 :
        <程式碼>
    else:
        <程式碼>
elif 條件2:
    <程式碼>
else:
    <程式碼>
```

Test_if_if.py

```
a = 80
b = 50

if a < 100:
    if b > 30:
        print (“b>30”)
    elif b < 30:
        print (“b<30”)
    else:
        print (“b=30”)
elif a > 100 :
    print(“a>100”)
else:
    print(“a=100”)
```

Python的邏輯比較

- Python的邏輯比較有 and, or , not三種
- 邏輯的變數中，False, None, 0, 空字串, 空陣列(列表、字典)都是False，其餘都為True
- Ex: not False => True
- Ex: not () => True
- Ex: not 3 => False

true_false.py

```
a = 0
b = 1
c = a and b
Print (c)
```

for迴圈與range()函數

- Python的for迴圈與其他程式語言用法差異很大但卻更加好用。
- C語言的for迴圈用下列方法表示

```
for ( i =0; i < 10; i++){  <語句> };
```

- Python的for迴圈用下列方式表示

```
for i in x:
    <語句>
for i in range(0, 10, 2):
    <語句>
```

for <變數> in <變數內容>:

    <語句>

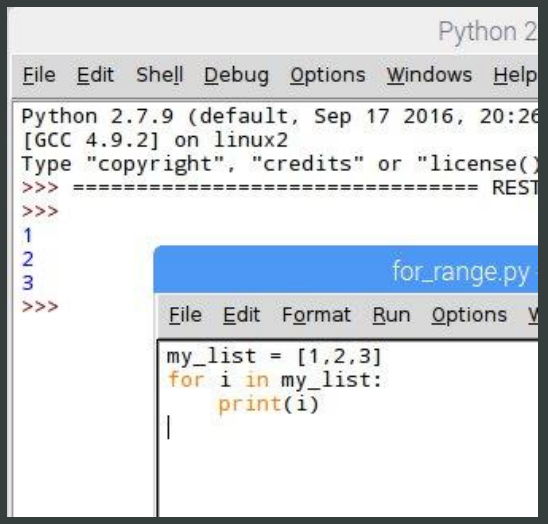
for <變數> in range(起始值, 終止值, Step值):

    <語句>

For迴圈範例

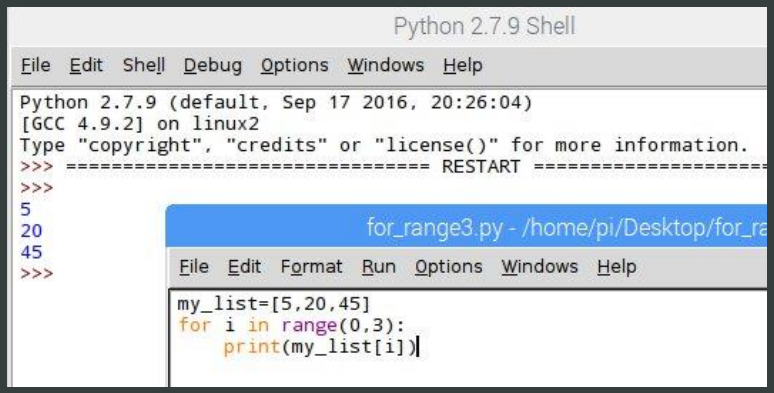
for_range.py

```
my_list = [1,2,3]
for i in my_list:
    print(i)
```



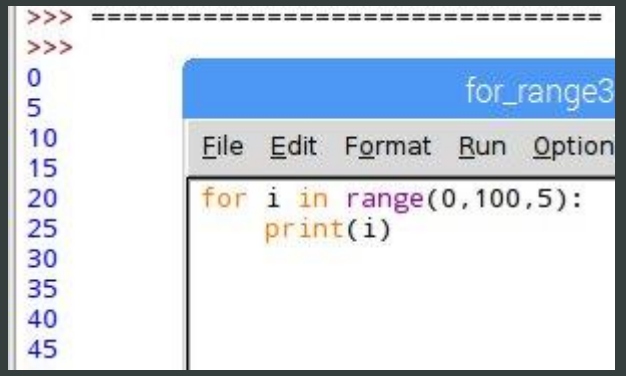
for_range2.py

```
my_list=[5,20,45]
for i in range(0,3):
    print(my_list[i])
```



for_range3.py

```
for i in range(0,100,5):
    print(i)
```



While … else …

- While 用法跟一般的C語言差不多，當條件成立時持續執行直到條件滿足時跳出。
- Python的While語法多了else判斷式，當While條件都不滿足時可跳到else判斷式執行語句。

```
while 條件 :
    <程式碼>
else 條件:
    <程式碼>
```

```
my_list = [10,20,30]
i = 0
listTotal = len(my_list)
while i < listTotal:
    print("list:",i," = ",my_list[i])
    i += 1
else:
    print("end")
```

while_else.py

Python的自訂函式

- Python的自訂函式需先定義聲明才能執行，調用函式就是函式(參數)的方式執行
- 參數沒有一定要先宣告參數的資料型別，也沒有一定要有參數，但可先給預設值
- Return值可以省略，回傳的值是None

```
def <函式名> (參數1,參數2,…):
    <函式內語句>
    return <返回值>
```

```
def <函式名> (參數1=預設值,參數2=預設值):
    <函式內語句>
    return <返回值>
```

```
函式名稱(參數1,參數2,…)
```

```
函式名稱(參數1,…)
```

自訂參數範例

```
def sumAll(x,y):
    z = x + y
    return z

print(sumAll(4,8))
```

def_Funtion.py

```
def name (a='John', b='Kate'):
    if a == "May":
        print(a)
    else:
        print(b)

name("X")
```

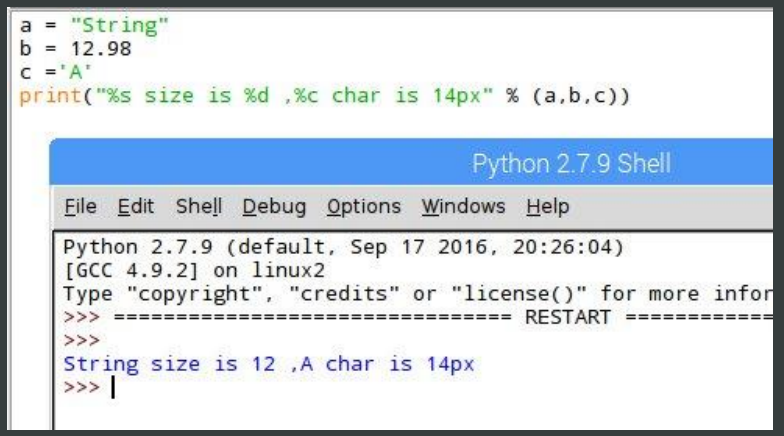
def_Funtion2.py

Python 的Print

- Print 輸出可用格式化字串，要替代的

字串或數值用%表示，後面

  - %d 十進制
  - %s 字串
  - %c 字元



Print_format.py

```
a=“String”
b=12.98
c=‘a’
print(“%s size is %d, %d char is 14px” % (a,b,c))
```

Python 的import

- Python的模組套件相當廣泛，從硬體控制到網路伺服器操控都有，引用正確的模組套件就可快速開發出需要的功能
- Python所撰寫的 *.py 檔案屬於一個模組(module)的概念，每個模組內含許多函式與運算，當許多的模組包在一起就是套件(package)
- 引用一個模組或套件的語法如下

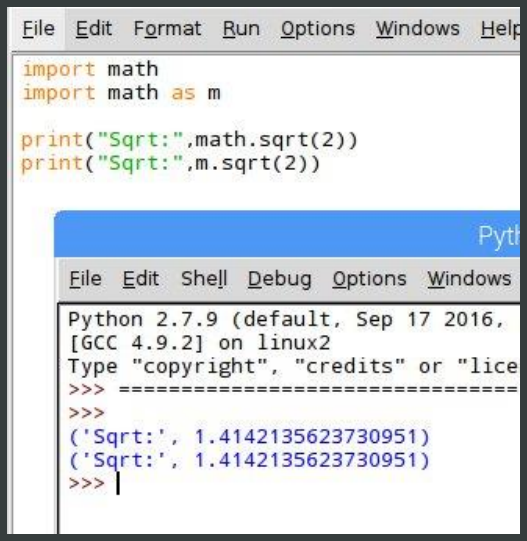
```
import <模組名>
import <模組名> as <自訂名稱>
from <模組名> import <函式名稱>
```

範例

Import_module.py

```
import math

Print(“Sqrt:”,math.sqrt(2))
```

#調用math模組內的sprt函式輸出2的平方根



Import_module.py

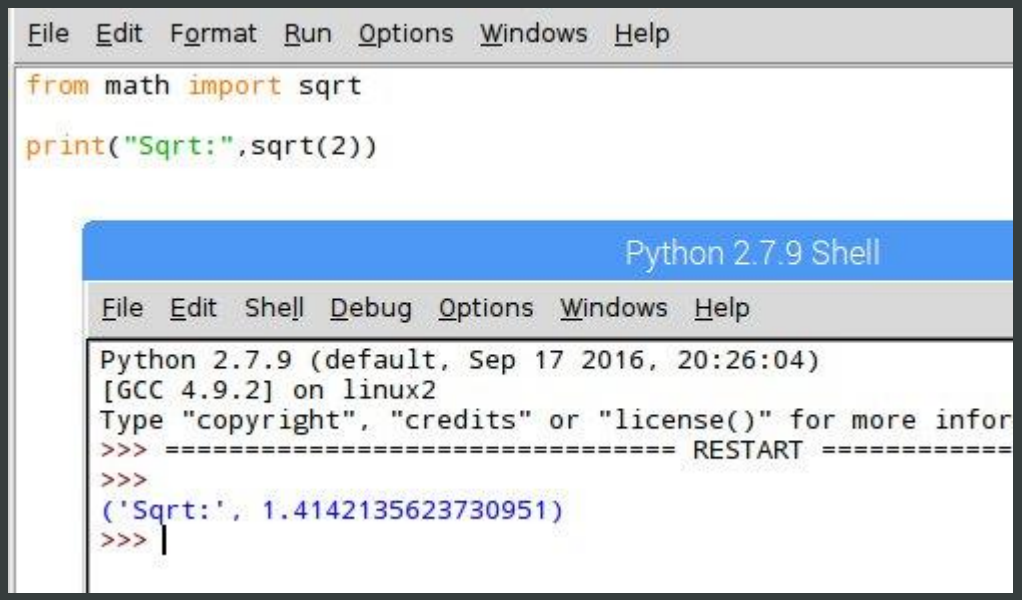
```
import math as m

Print(“Sqrt:”, m.sqrt(2))
```

#調用math模組，並重新定義m的新名稱再調用math模組內的sprt函式輸出2的平方根

Import_module2.py

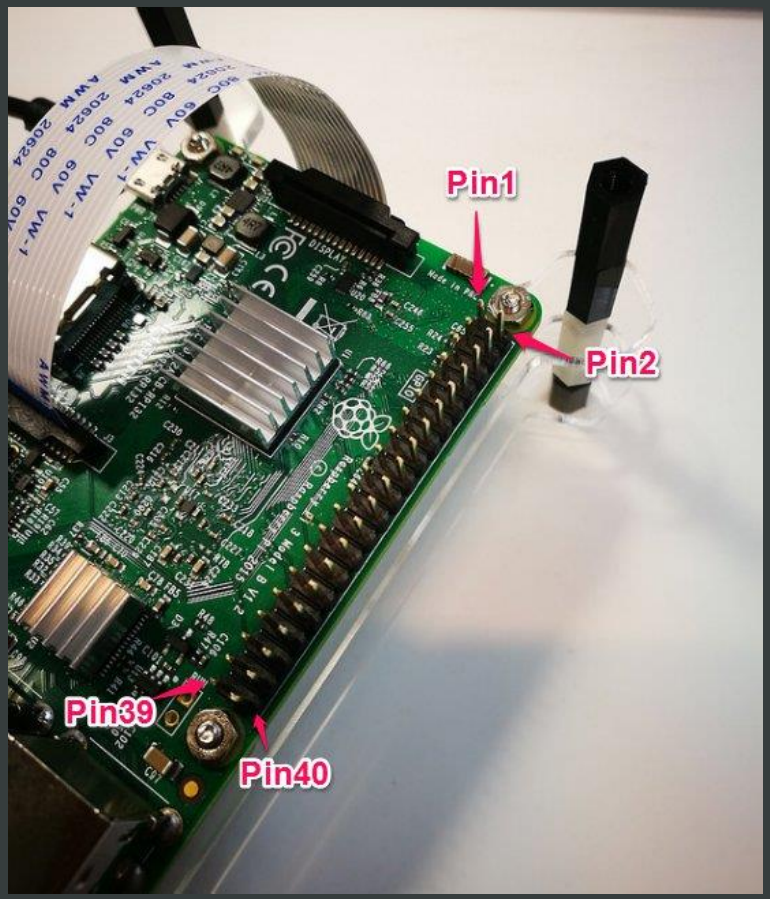
```
from math import sqrt

Print(“Sqrt:”,sqrt(2))
```

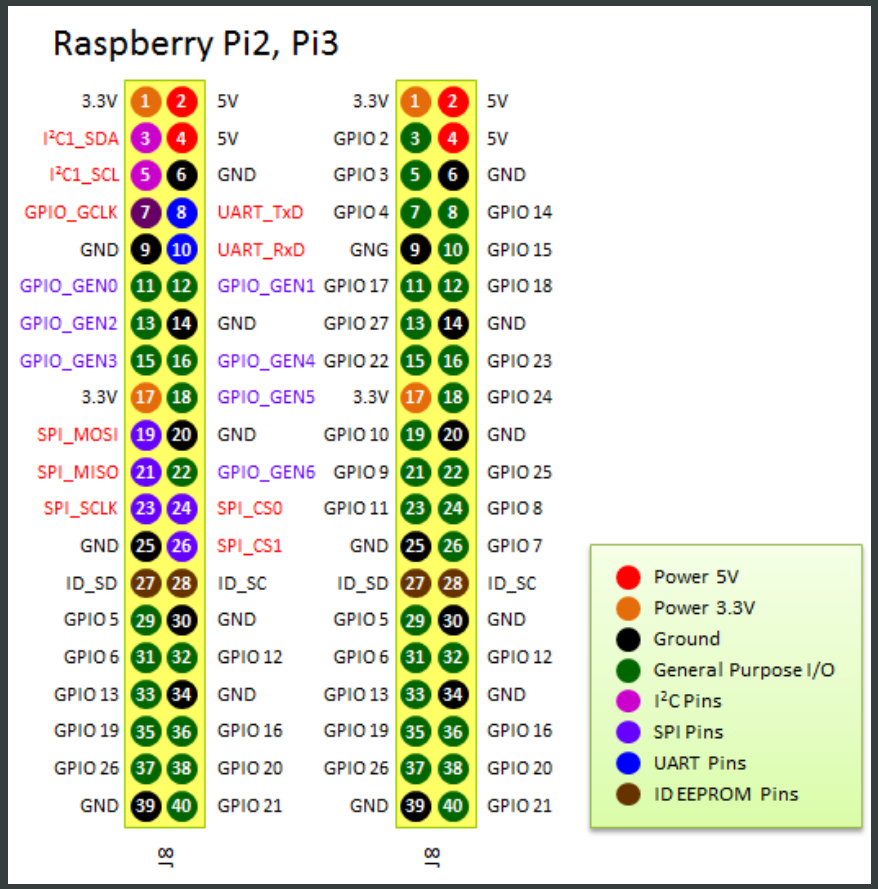
#直接調用math模組內的sprt函式並輸出2的平方根



GPIO (PWM)

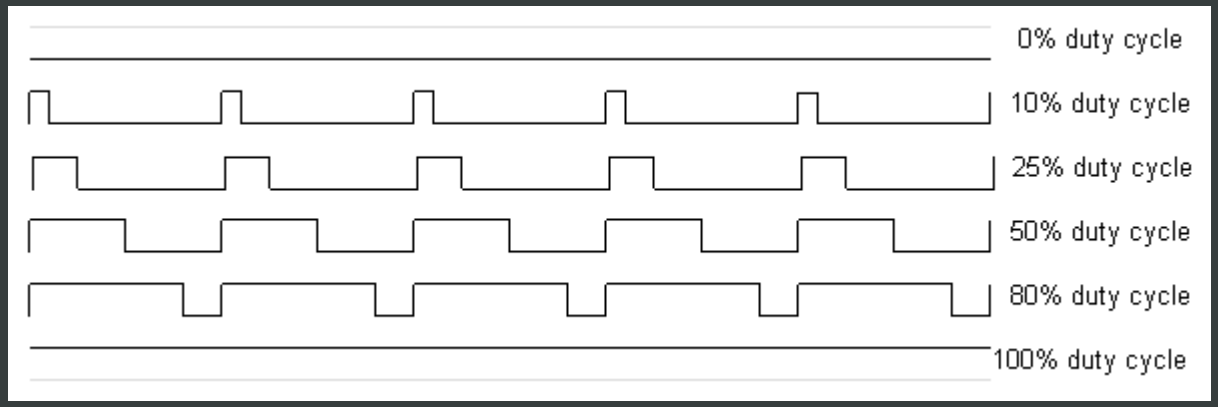
GPIO(General Purpose Input & Output)



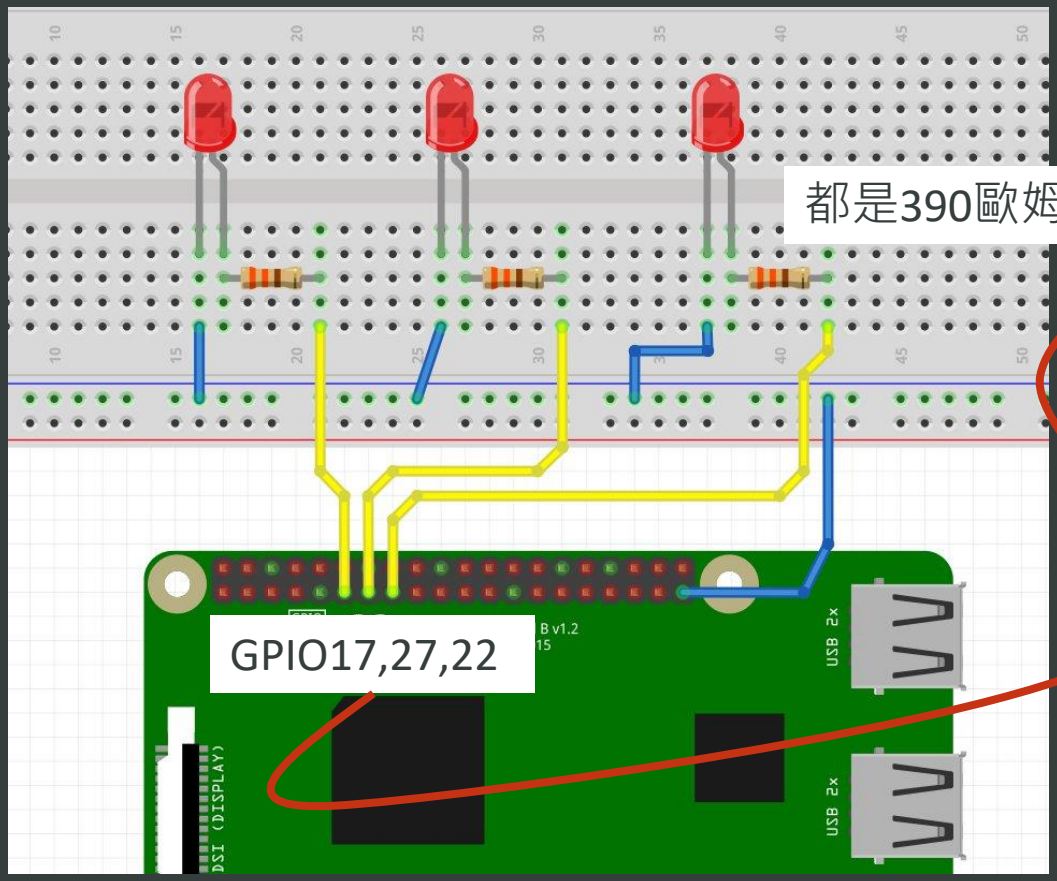


PWM

- PWM(Pulse Width Modulation) 脈衝寬度調變
- Raspberry Pi 3的PWM可產生出 1Hz ~ 19.2MHz
- Arduino Uno的PWM可產生出 1Hz ~ 36.35KHz
- 最常使用PWM的裝置如LED, Servo, 某些LED(WS2811,WS2812)晶片會需要
- Pi的調變PWM數值範圍跟Arduino不同，Arduino頻率被固定在490Hz或980Hz(Pin5, 6)，Pi的頻率可以透過GPIO.PWM()設定需要的頻率



實驗1-PWM控制LED*3



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

```
import RPi.GPIO as GPIO
import time

led1=17

delayTime = 0.01

GPIO.setmode(GPIO.BCM)
GPIO.setup(led1, GPIO.OUT)

pwm1 = GPIO.PWM(led1, 50)
pwm1.start(0)

while True:
    for i in range(0,101):
        pwm1.ChangeDutyCycle(i)
        time.sleep(delayTime)

    for i in range(100,-1,-1):
        pwm1.ChangeDutyCycle(i)
        time.sleep(delayTime)
```

```
#匯入gpio套件，並重新自訂名稱為GPIO
#匯入time套件，後面使用sleep函數

#定義要點亮LED的腳位，PWM只支援
#17, 18, 27, 22, 23, 24, 25
#每次更新的延遲時間

#設定模式, BCM模式為Broadcom GPIO模組
#設定GPIO腳位為輸出狀態

#設定pwm1為50Hz速度 = 20ms為一個脈波週期
#設定pwm1初始值為0，最大到100

#當程式開始執行時
#用for…Range產生0~100的循環數值i，默認遞增1
#改變PWM週期，隨著變數i從0~100
#delay 0.05秒

#用for…Range產生100~0的循環數值，遞減1
#改變PWM週期，隨著變數i從0~100
#delay 0.05秒
```

Blink_1Leds_PWM.py

```
import RPi.GPIO as GPIO
import time

led1= [17,27,22]
delayTime = 0.01
GPIO.setmode(GPIO.BCM)
for i in range(0,3):
    GPIO.setup(ledPins[i], GPIO.OUT)

pwmR = GPIO.PWM(ledPins[0], 50)
pwmG = GPIO.PWM(ledPins[1], 50)
pwmB = GPIO.PWM(ledPins[2], 50)
pwmR.start(0)
pwmG.start(0)
pwmB.start(0)
while True:
    for i in range(0,101):
        pwmR.ChangeDutyCycle(i)
        #pwmG.ChangeDutyCycle(i)
        #pwmB.ChangeDutyCycle(i)
        time.sleep(delayTime)

    for i in range(100,-1,-1):
        pwmR.ChangeDutyCycle(i)
        #pwmG.ChangeDutyCycle(i)
        #pwmB.ChangeDutyCycle(i)
        time.sleep(delayTime)
```

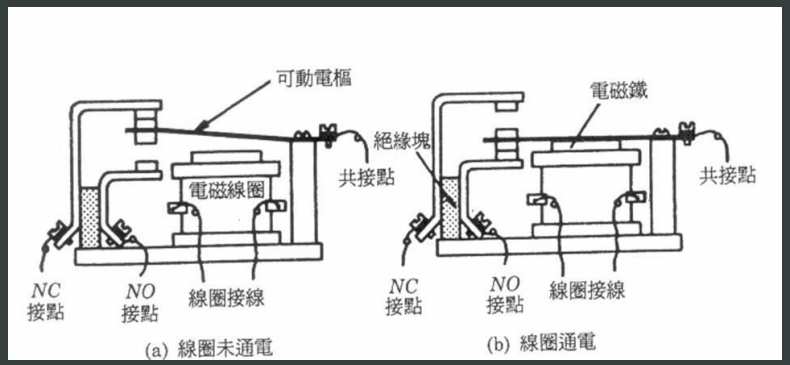
多個IO可以用list來表示

索引list內的元素

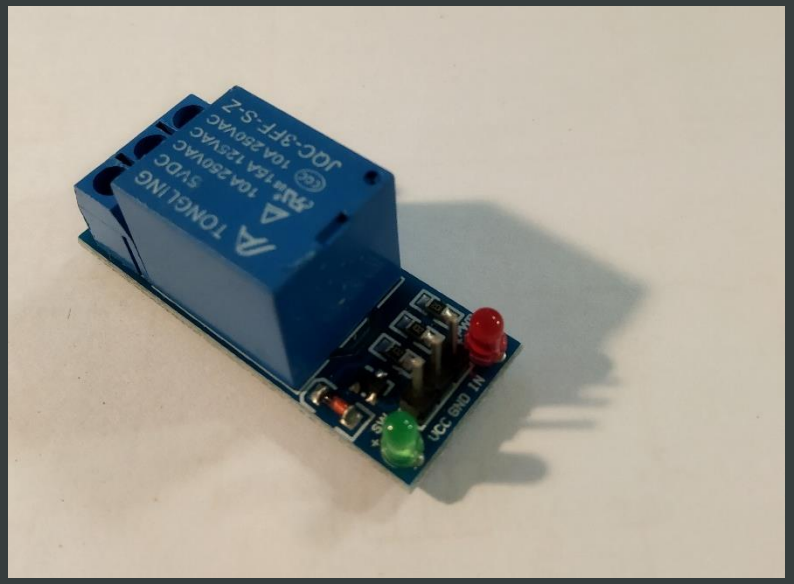
補充說明

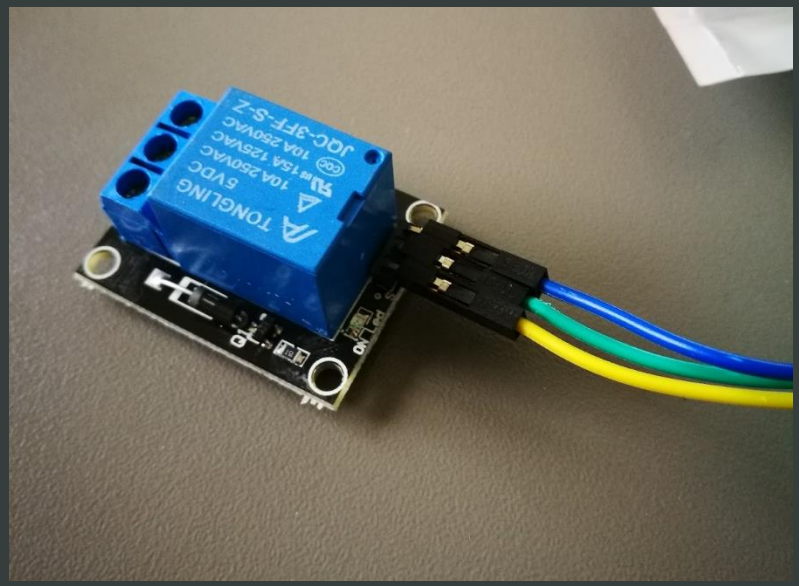
繼電器模組

5V驅動的繼電器模塊



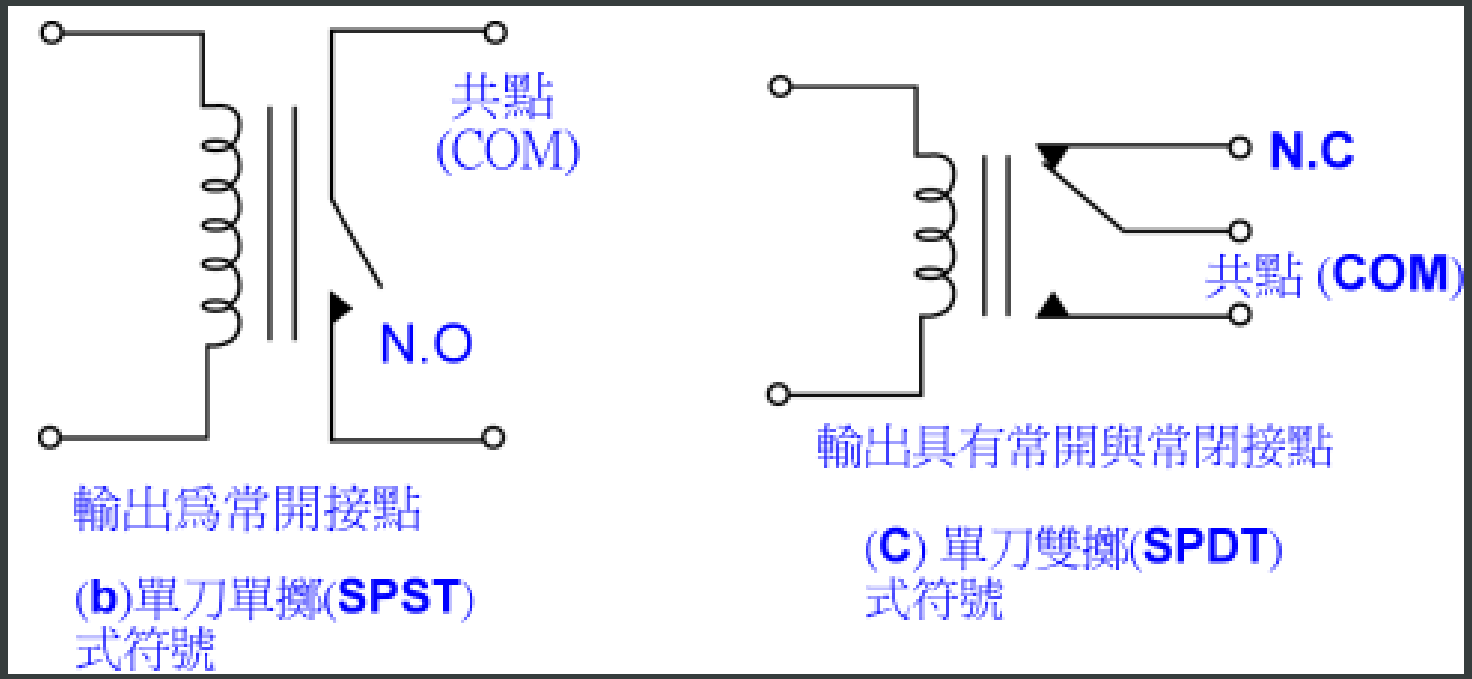
- 5VDC
- 10A250VAC
- 15A125VAC
- 7A250VDC





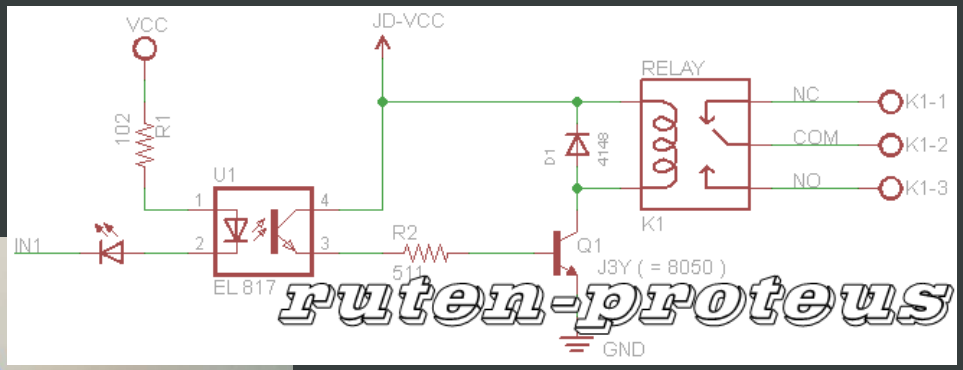
繼電器原理圖

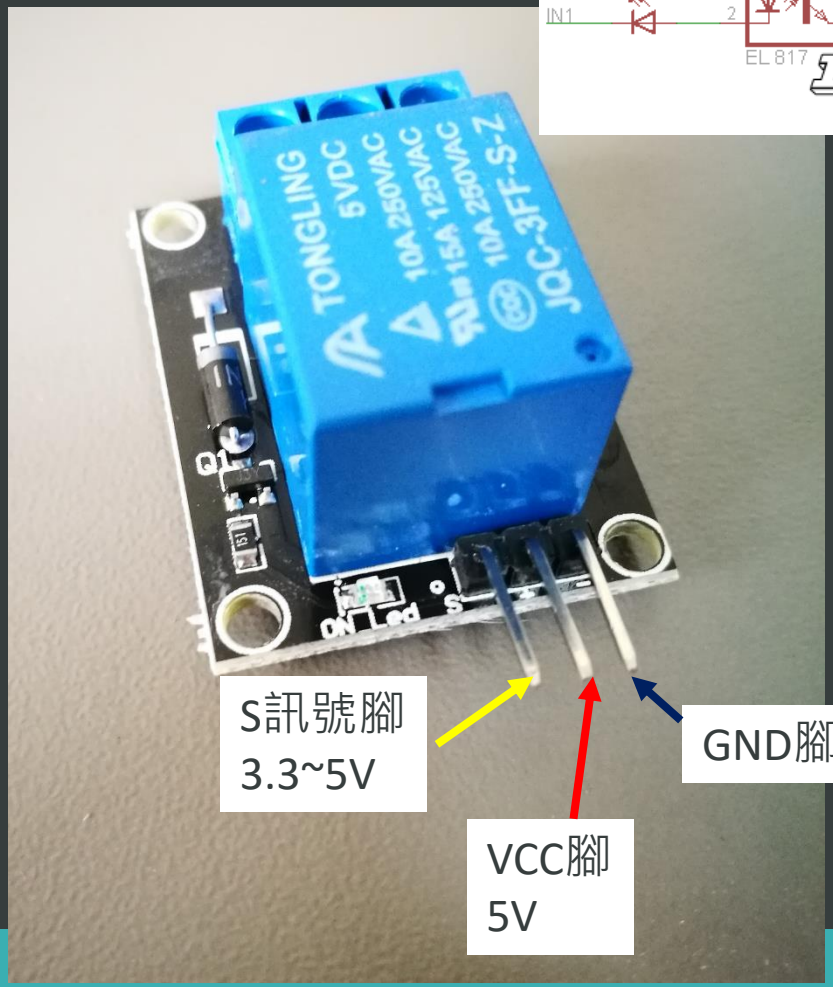
- 常開接點(N.O Normal Open)
- 常關接點 (N.C Normal Close)
- 共通點 (COM)



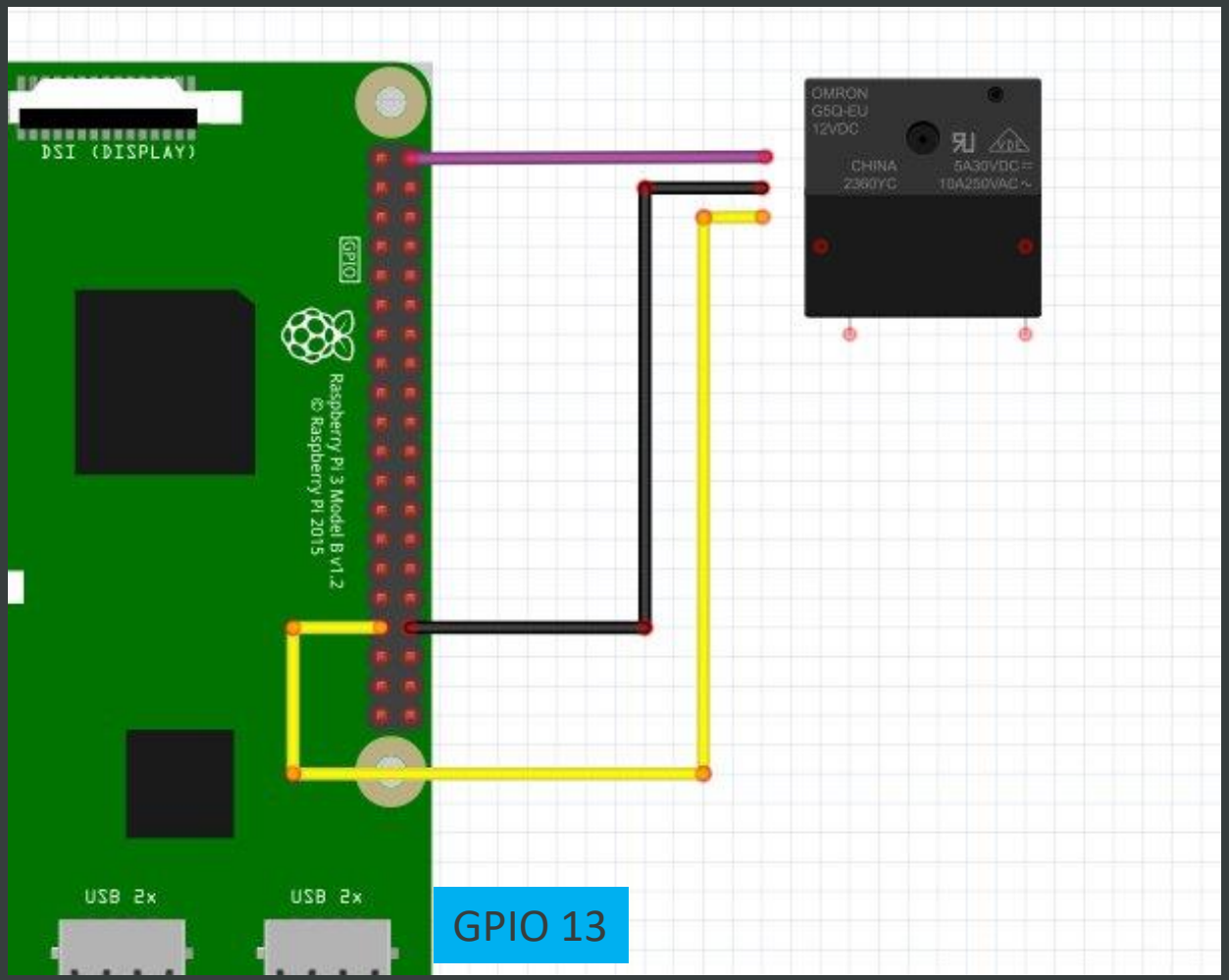
http://content.saihs.edu.tw/contentbook/bep1/U1/15/b/b5.htm

確認腳位定義


VCC
JD-VCC
RELAY
NC
COM
NO
K1-1
K1-2
K1-3
K1
102
R1
U1
1
2
3
4
EL 817
IN1
R2
511
Q1
J3Y (= 8050)
D1
4148
GND
ruten-proteus

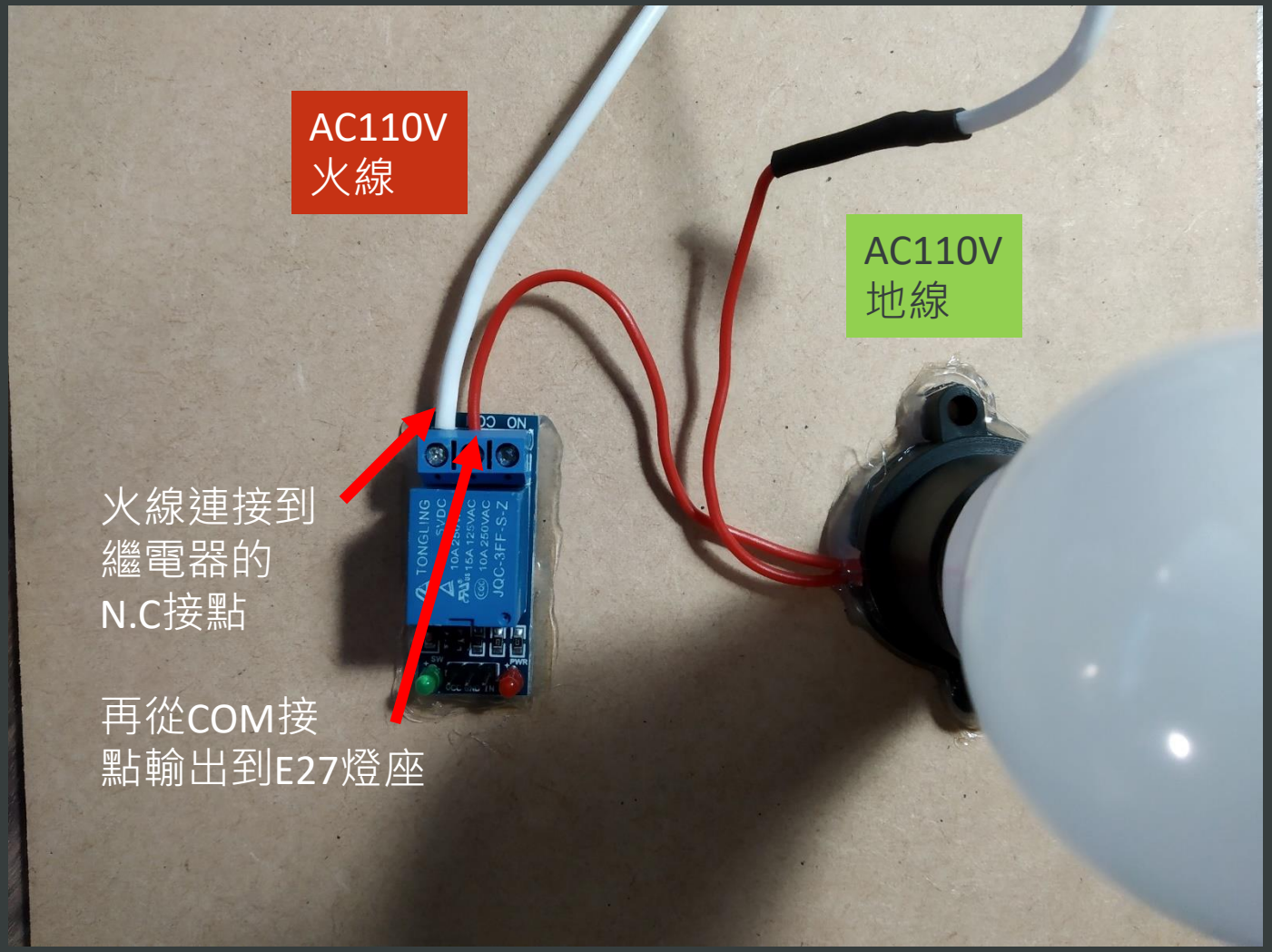

TONGLING
5VDC
10A 250VAC
15A 125VAC
10A 250VAC
JQC-3FF-S-Z
S訊號腳
3.3~5V
GND腳
VCC腳
5V

實驗2-繼電器接線圖



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

連接交流電與E27燈泡座



!!!注意確定繼電器的接點有鎖緊才可以接交流電110V!!!

繼電器模塊控制程式碼

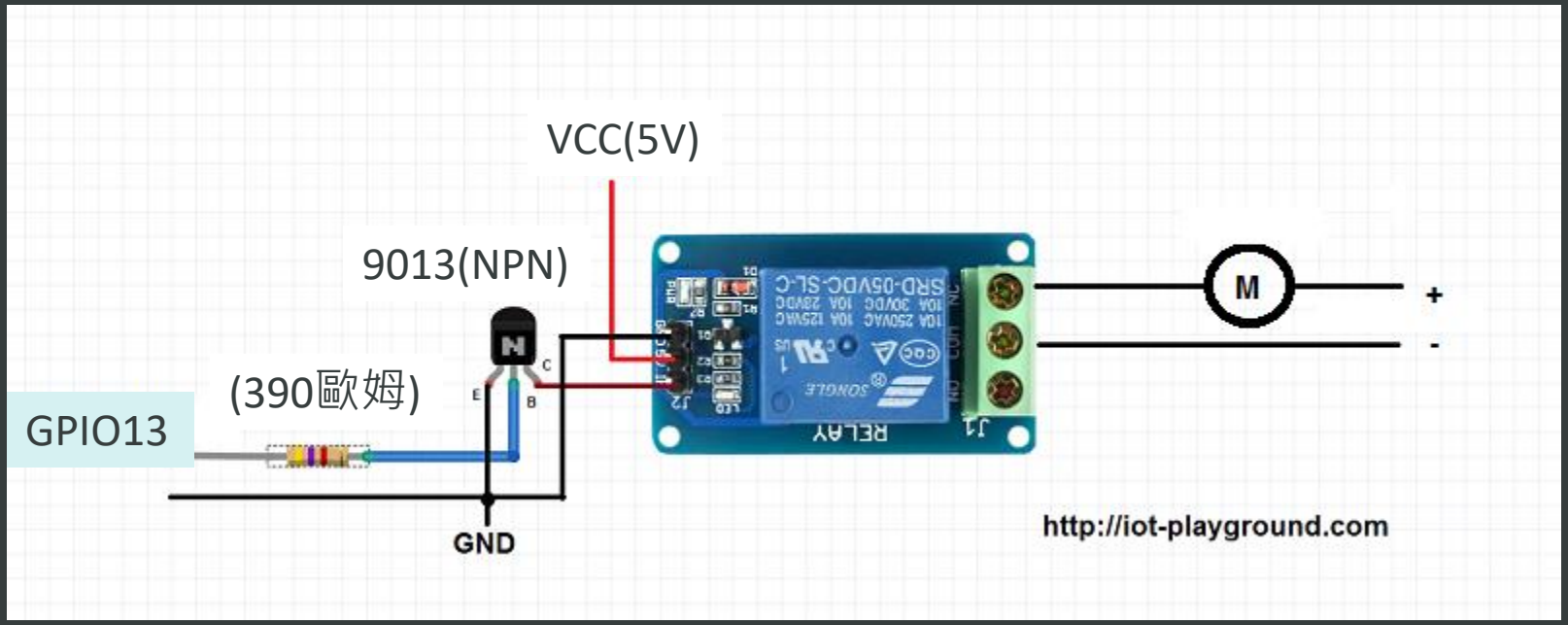
```
import RPi.GPIO as GPIO
import time

relay1=13
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay1, GPIO.OUT)

while True:
    GPIO.output(relay1,True)
    time.sleep(1)
    print("ON")
    GPIO.output(relay1,False)
    time.sleep(1)
    print("OFF")
```

```
#匯入gpio套件，並重新自訂名稱為GPIO
#匯入time套件

#設定Relay pin腳為GPIO 13
#設定GPIO模式為BCM
#設定GPIO 13為輸出模式 GPIO.OUT

#程式開始執行
#Relay Pin 13輸出高電位
#delay 1秒

#Relay Pin 13輸出低電位
#delay 1秒
```

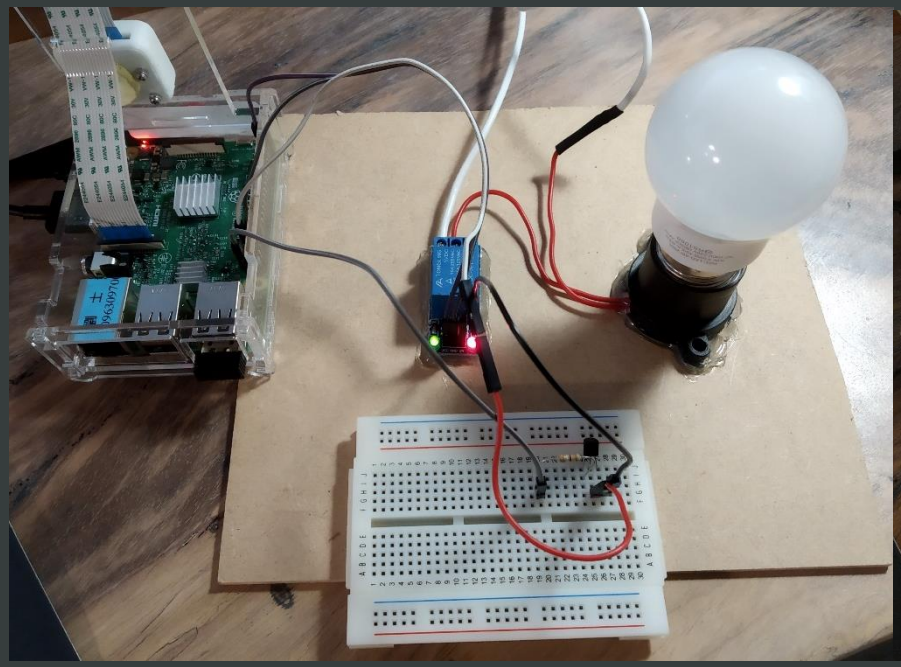
Relay_1.py

如果沒有驅動(因驅動電流太弱)

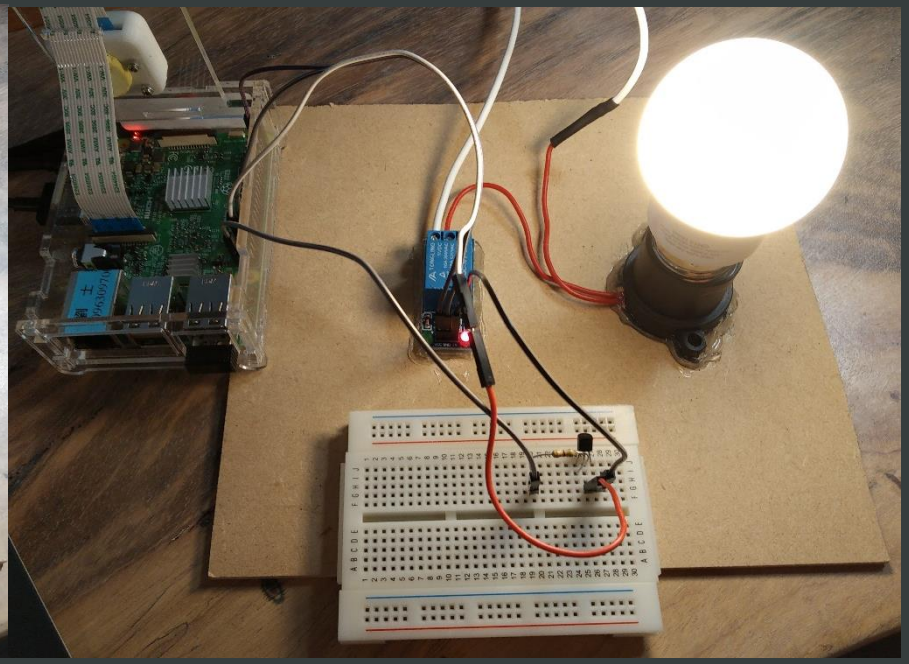
- 增加一顆9013 NPN電晶體放大驅動電流即可



執行效果



OFF

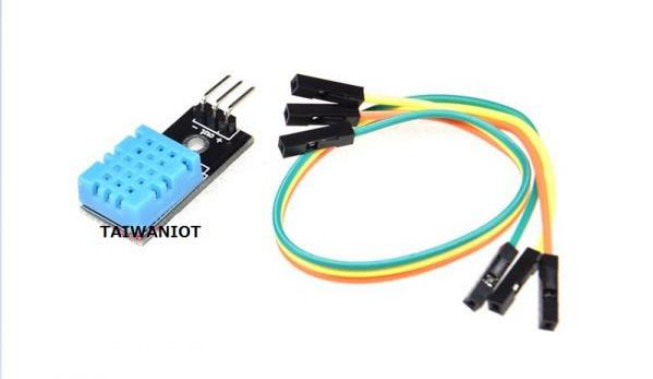


ON

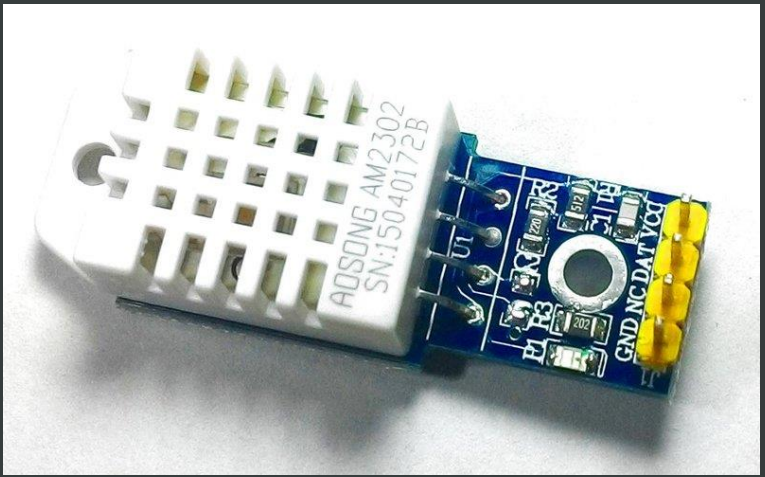
DHT11 溫度感測器

溫溼度模組

- DHT11
- DHT22
- LM35



DHT11



DHT22

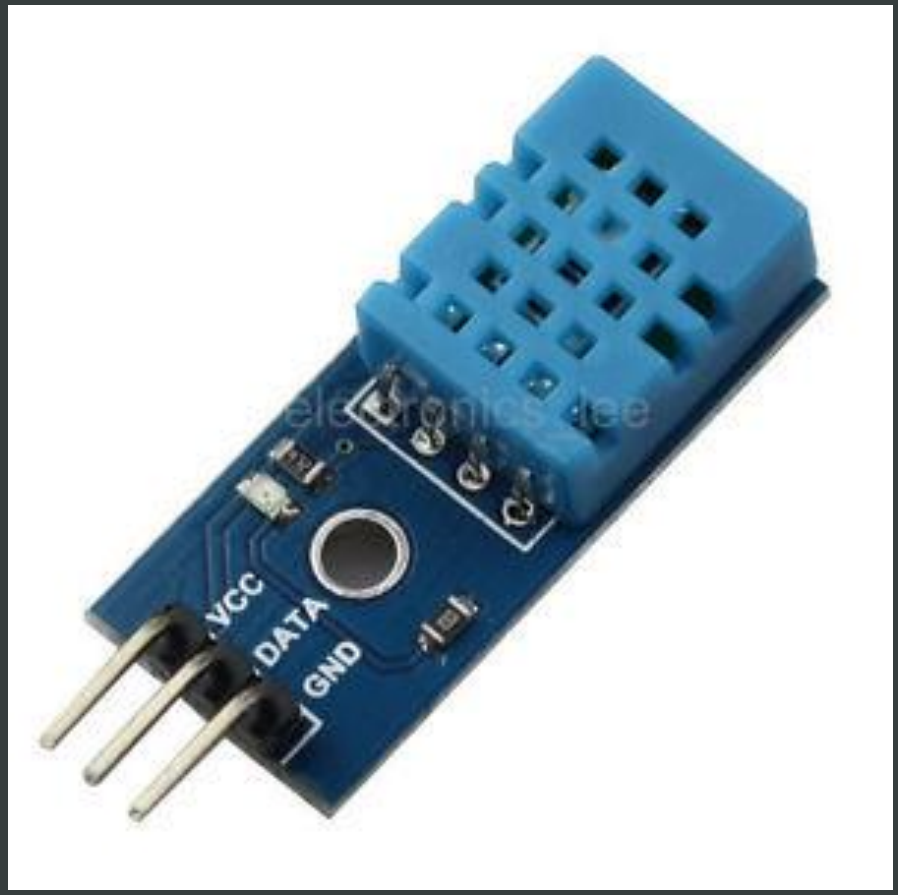
LM35

常見三款溫溼度感測器差異

項目	DHT11	DHT22	LM35
輸出訊號方式	數位式	數位式	類比式
感測溫度範圍	0 ~ +50℃	-40℃–80℃	−55°C to 150°C
感測濕度範圍	20% ~ 90%RH	0%–100%RH	無
測量精確度	±2.0℃ ±5.0%RH	±0.5℃ ±2%RH	±¼°C
回應時間	<5s	<2s	<1s

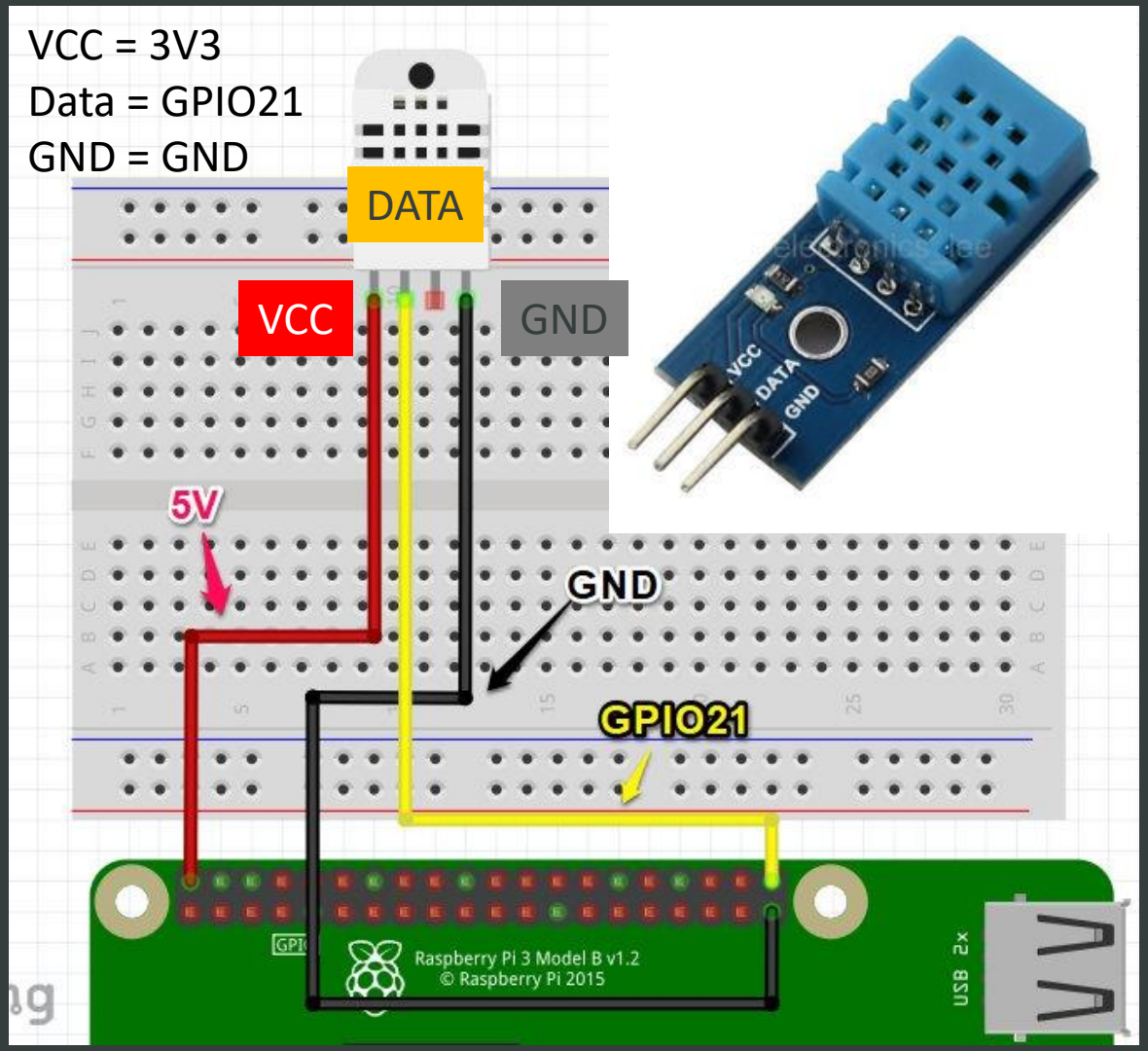
DHT11 溫溼度感測器

- 測量範圍
  - 溫度0~50度
  - 濕度20~90%RH
- 精確度
  - 濕度 正負5%RH
  - 溫度 正負2%度
  - 解析度單位：1
  - 分辨率：8bit
- 輸入電壓
  - 3 ~ 5.5V



http://www.raspberrypi.com.tw/upload/spec/sensor/DHT11-chinese.pdf

實驗3-DHT11溫溼度感測器接線圖



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

安裝Adafruit_Python_DHT模組

Step1 – 下載Adafruit_Python_DHT程式庫

```
$ cd ~
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

Step2 – 安裝

```
$ cd Adafruit_Python_DHT
$ sudo python3 setup.py install
```

程式碼

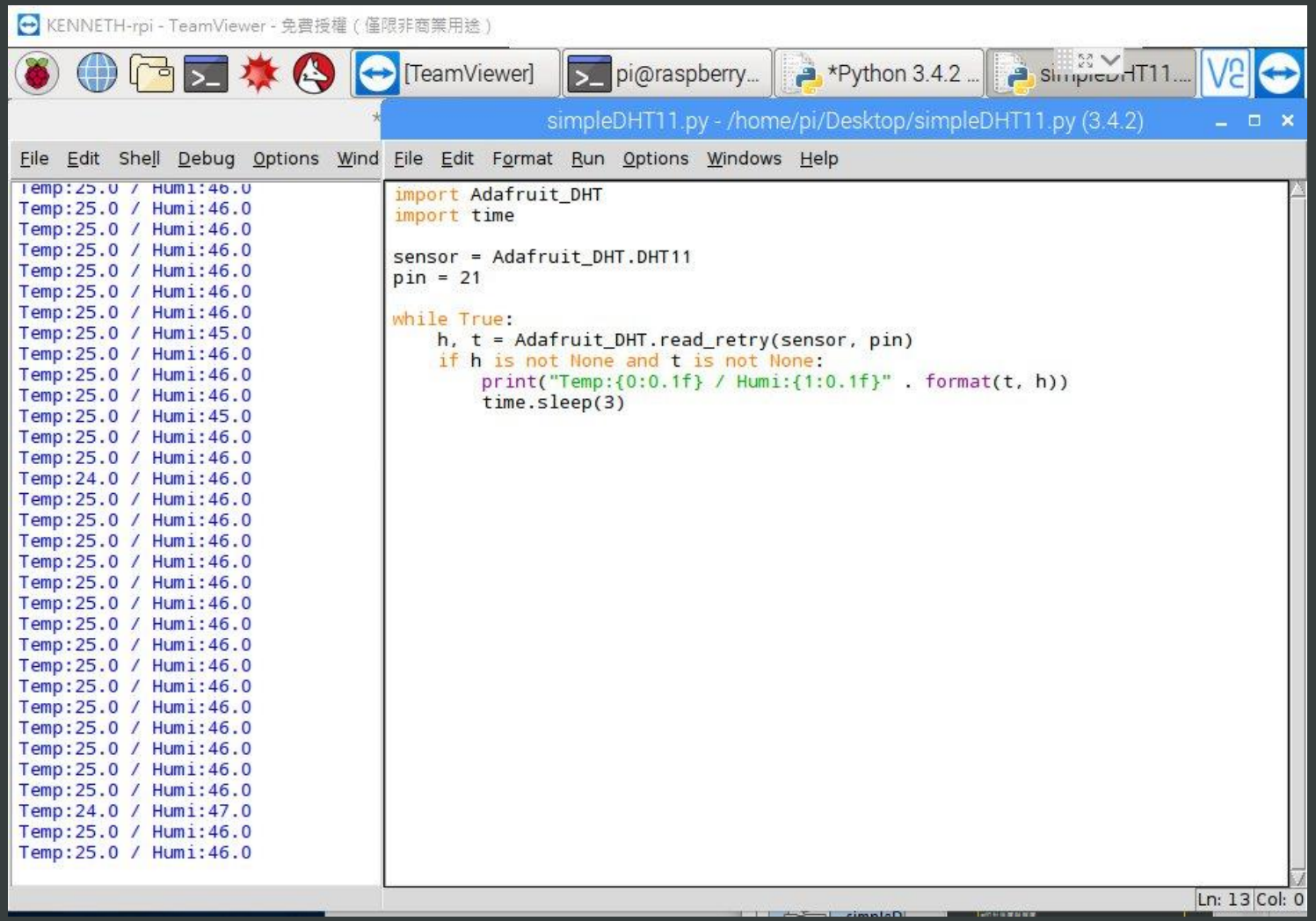
```
import Adafruit_DHT
import time

sensor = Adafruit_DHT.DHT11
pin = 21

while True:
    h, t = Adafruit_DHT.read_retry(sensor, pin)
    if h is not None and t is not None:
        print("Temp:{0:0.1f} / Humi:{1:0.1f}".format(t, h))
        time.sleep(3)
```

simpleDHT11.py

程式碼說明

```
#匯入 Adafruit_DHT
#匯入 time

#建立sensor物件
#宣告要感測的腳位=21


#每3秒讀取溫度與濕度
#判別溫度與濕度是否有
#印出溫度與濕度
#每3秒抓取一次
```

輸出結果


KENNETH-rpi - TeamViewer - 免費授權（僅限非商業用途）
[TeamViewer]
pi@raspberry...
*Python 3.4.2 ...
simpleDHT11....
simpleDHT11.py - /home/pi/Desktop/simpleDHT11.py (3.4.2)
File Edit Shell Debug Options Wind
File Edit Format Run Options Windows Help
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:45.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:45.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:24.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
Temp:24.0 / Humi:47.0
Temp:25.0 / Humi:46.0
Temp:25.0 / Humi:46.0
import Adafruit_DHT
import time
sensor = Adafruit_DHT.DHT11
pin = 21
while True:
h, t = Adafruit_DHT.read_retry(sensor, pin)
if h is not None and t is not None:
print("Temp:{0:0.1f} / Humi:{1:0.1f}" . format(t, h))
time.sleep(3)
Ln: 13 Col: 0

Format( )

- 增強型格式化字串函式
- 用法1

  “{0},{1}”.format(‘raspberry’,’pi’)

  輸出:raspberry,pi

```
>>> "{0},{1}".format('raspberry','pi')
'raspberry,pi'
>>>
```

- 用法2

  “Hi {0} {1} / {2}”.format(‘raspberry’,’pi’,3)

  輸出:Hi raspberry pi / 3

```
>>> "Hi {0} {1} / {2}".format('raspberry','pi',3)
'Hi raspberry pi / 3'
```

- 用法3

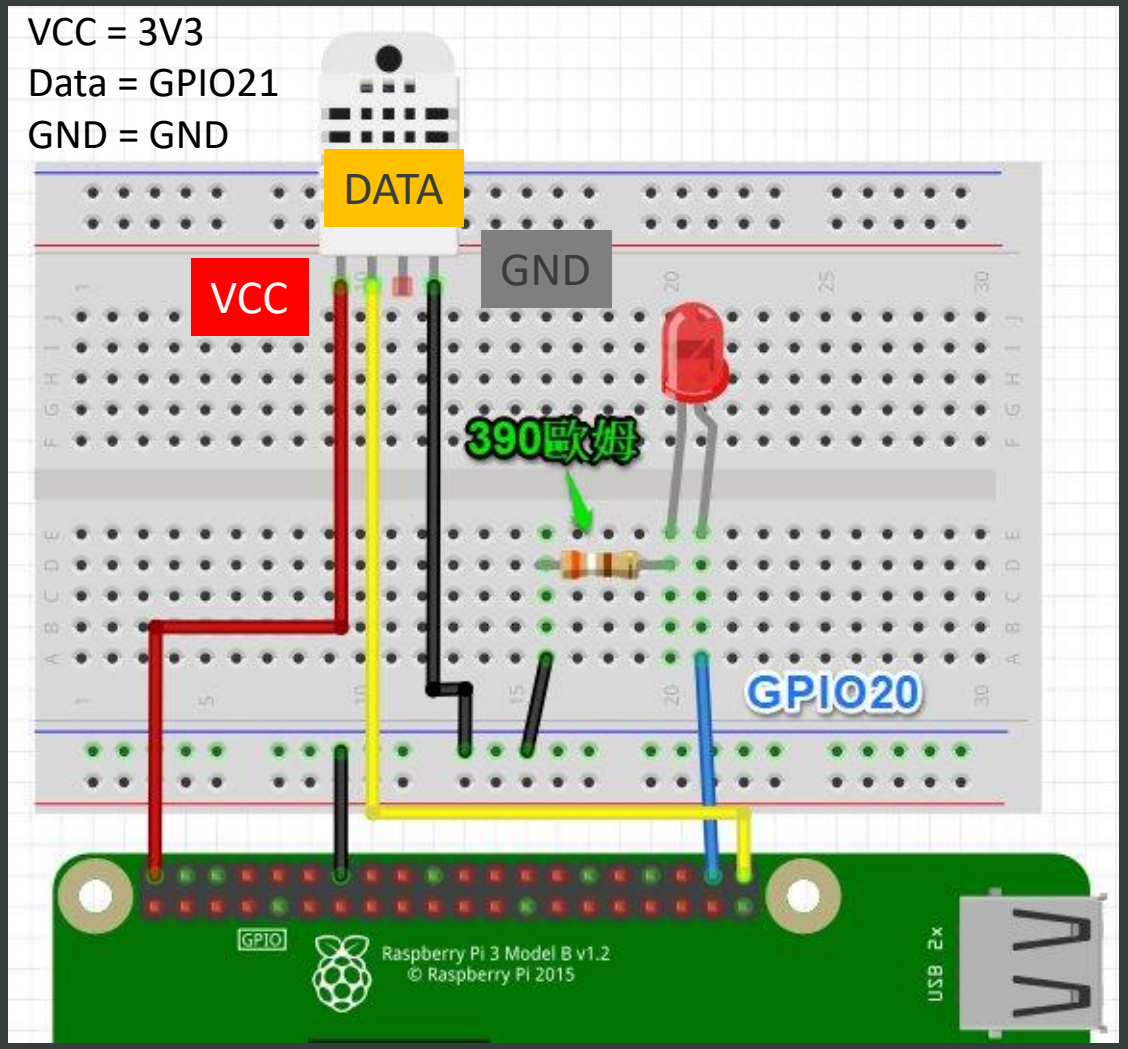
  “Hi {0}:{1} / {2} = {3:0.1f}”.format(‘raspberry’,’pi’,3, 25.22456)

  輸出:Hi raspberry:pi / 3 = 25.2

```
>>> "Hi {0}:{1} / {2} = {3:0.1f}".format('raspberry','pi',3, 25.22456)
'Hi raspberry:pi / 3 = 25.2'
```

實驗4-溫度感測+LED控制

VCC = 3V3
Data = GPIO21
GND = GND



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

程式碼

```
import Adafruit_DHT
import time
import RPi.GPIO as GPIO

sensor = Adafruit_DHT.DHT11
pin = 21
led = 20

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(led, GPIO.OUT)

tempData = "0.0"
humiData = "0.0"
```

```
#匯入 Adafruit_DHT
#匯入 time
#匯入 GPIO

#建立sensor物件
#宣告要感測的腳位=21
#宣告LED腳位=20


#關閉IO錯誤警告
#設置LED為輸出模式

#宣告讀取的溫度字串
#宣告讀取的濕度字串
```

1/2

程式碼

```
while True:
    h, t = Adafruit_DHT.read_retry(sensor, pin)
    if h is not None and t is not None:
        print("Temp:{0:0.1f} / Humidity:{1:0.1f}" . format(t, h))

        tempData = "{0:0.1f}" . format(t)
        humiData = "{0:0.1f}" . format(h)

        if float(tempData) > 25.0:
            GPIO.output(led, True)
        else:
            GPIO.output(led, False)

        time.sleep(3)
```

simpleDHT11_withGPIO.py

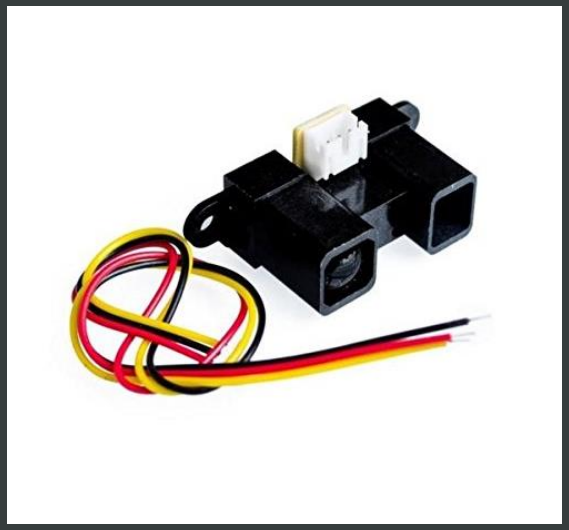
2/2

距離感測器

距離感測器種類

- 利用超音波方式經由音速量測目標物體距離
- 利用紅外線反射光的方式量測目標物距離
- 利用雷達微波感應方式量測
- 利用環境光線方式感測
- 利用移動偵測方式感測

SR-04 超音波感測器



紅外線距離感測器
GP2Y0A02TK0F

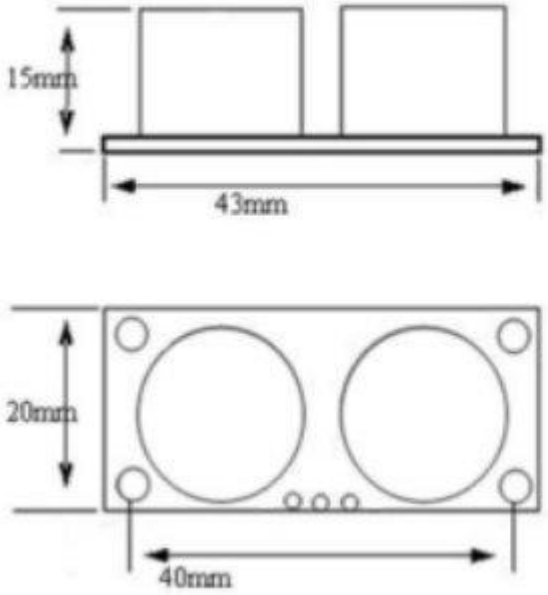
雷達微波感測器
HB100

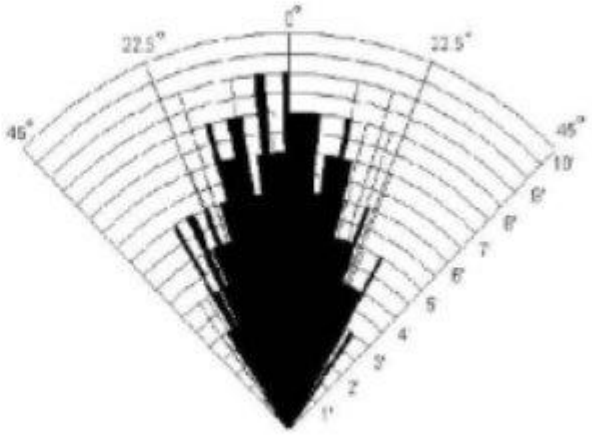
SR-04 超音波感測器

VCC = 5V
Trig = 控制信號輸入
Echo = 信號輸出
GND = 接地

工作電壓=5V (只能5V，給3.3V會工作不正常)
工作電流=15mA
感應角度=30度角
測量距離=2cm ~ 400cm
測量解析度=0.3cm
響應時間=10uS

測量角度

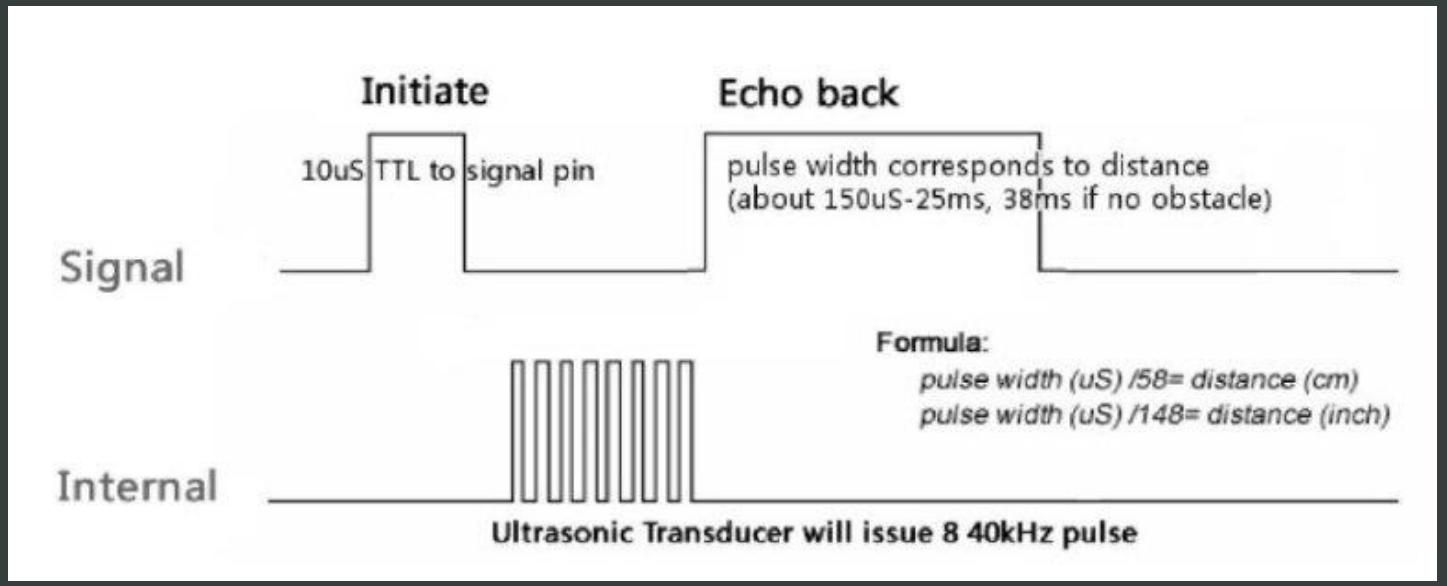




Practical test of performance,
Best in 30 degree angle

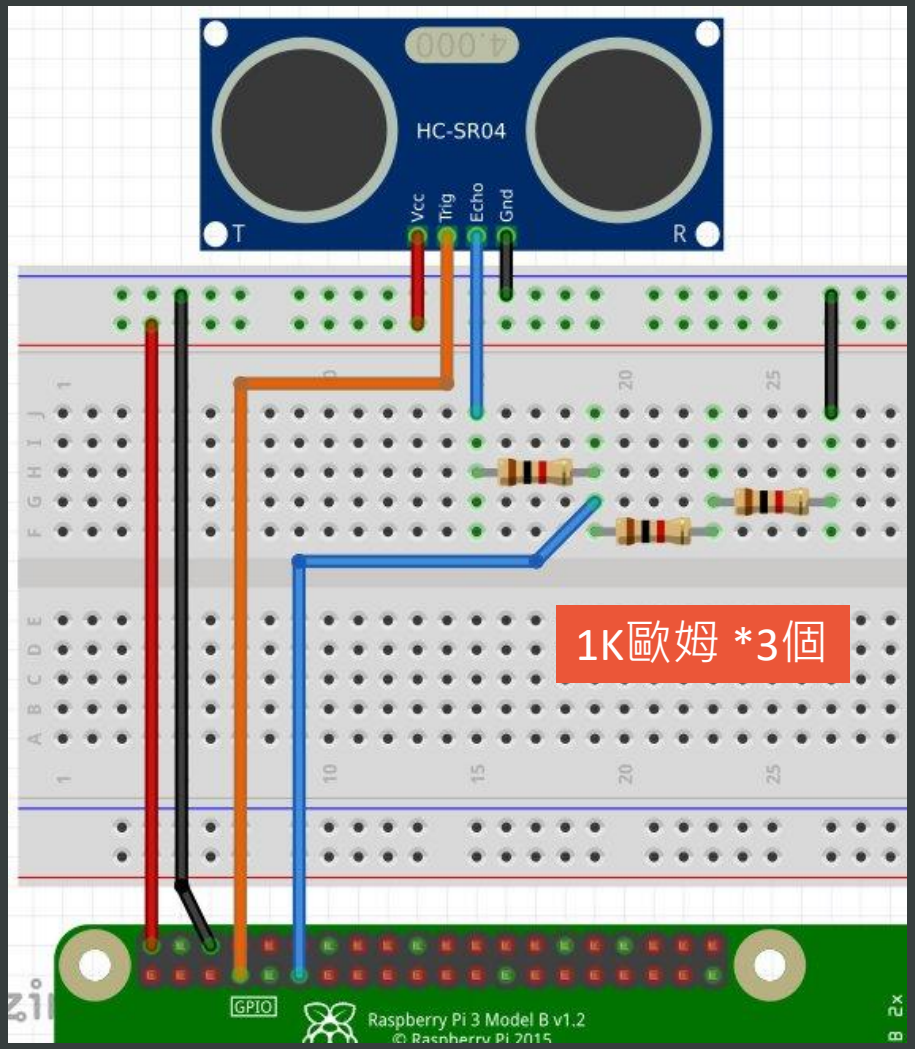
驅動方式

- Time = Width of Echo pulse, in uS (micro second)
  - Distance in centimeters = Time / 58
  - Distance in inches = Time / 148
  - Or you can utilize the speed of sound, which is 343m/s



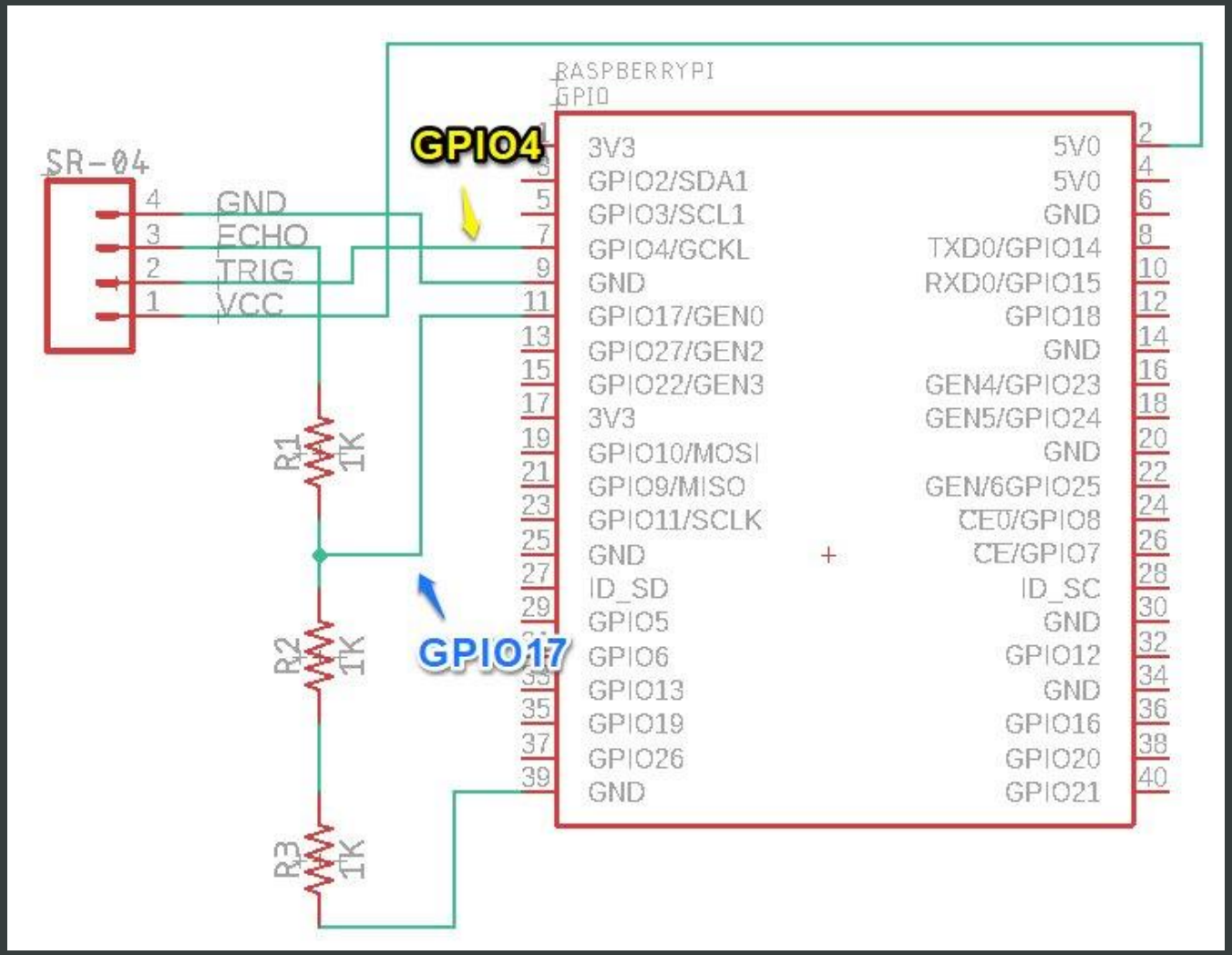
1.發送10uS的TTL訊號觸發模塊
2.發送8次40KHz超音波
3.等待回傳的超音波時間響應(uS, 微秒)，距離(cm) = (響應時間/2) / 29.41微秒
距離(cm) = 響應時間 * 17150(cm/秒)

實驗5-SR-04超音波距離感測器



	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

電路原理圖


SR-04
4 GND
3 ECHO
2 TRIG
1 VCC
R1 1K
R2 1K
R3 1K
GPIO4
GPIO17
RASPBERRYPI GPIO
3V3
GPIO2/SDA1
GPIO3/SCL1
GPIO4/GCKL
GND
GPIO17/GEN0
GPIO27/GEN2
GPIO22/GEN3
3V3
GPIO10/MOSI
GPIO9/MISO
GPIO11/SCLK
GND
ID_SD
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
GND
5V0
5V0
GND
TXD0/GPIO14
RXD0/GPIO15
GPIO18
GND
GEN4/GPIO23
GEN5/GPIO24
GND
GEN/6GPIO25
CE0/GPIO8
CE/GPIO7
ID_SC
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21

程式碼

```
import RPi.GPIO as GPIO
import time

TRIG = 4
ECHO = 17

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

GPIO.output(TRIG, False)
print("Waiting…")
time.sleep(1)
```

```
#匯入 GPIO
#匯入 time

#觸發感測的腳位=4
#接收pulse腳位=17


#關閉IO錯誤警告
#設置TRIG為輸出模式
#設置ECHO為輸入模式

#使TRIG腳位拉低電位
#等待Sensor
#停止1秒
```

1/2

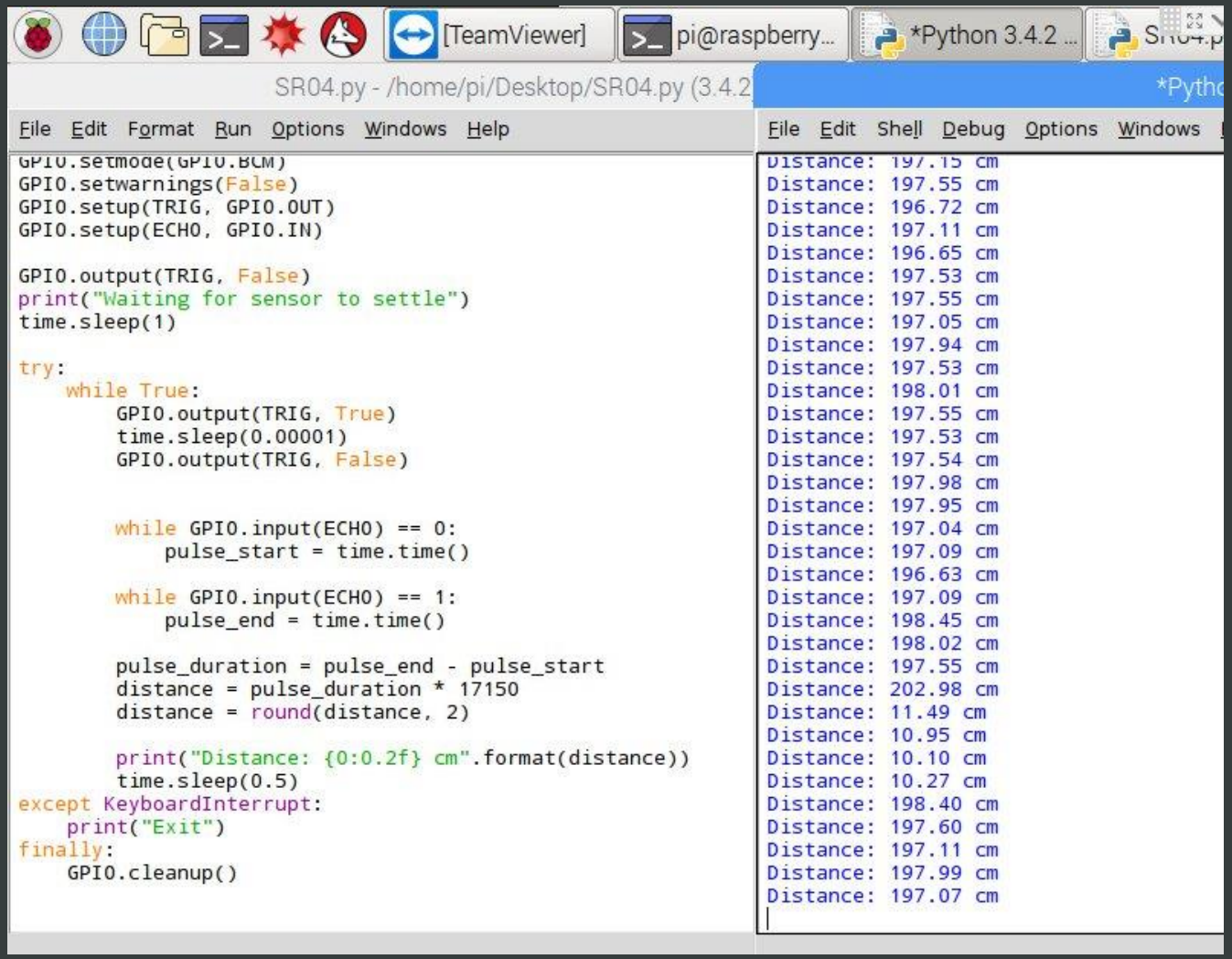
SR04.py

```
try:
    while True:
        GPIO.output(TRIG, True)
        time.sleep(0.00001)
        GPIO.output(TRIG, False)


        while GPIO.input(ECHO) == 0:
            pulse_start = time.time()

        while GPIO.input(ECHO) == 1:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 17150
        distance = round(distance, 2)

        print("Distance: {0:0.2f} cm".format(distance))
        time.sleep(0.1)

except KeyboardInterrupt:
    print("Exit")
finally:
    GPIO.cleanup()
```

SR04.py

2/2

輸出結果


[TeamViewer]
pi@raspberry...
*Python 3.4.2 ...
SR04.py - /home/pi/Desktop/SR04.py (3.4.2)
File Edit Format Run Options Windows Help
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.output(TRIG, False)
print("Waiting for sensor to settle")
time.sleep(1)
try:
while True:
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
while GPIO.input(ECHO) == 0:
pulse_start = time.time()
while GPIO.input(ECHO) == 1:
pulse_end = time.time()
pulse_duration = pulse_end - pulse_start
distance = pulse_duration * 17150
distance = round(distance, 2)
print("Distance: {0:0.2f} cm".format(distance))
time.sleep(0.5)
except KeyboardInterrupt:
print("Exit")
finally:
GPIO.cleanup()
File Edit Shell Debug Options Windows
Distance: 197.55 cm
Distance: 196.72 cm
Distance: 197.11 cm
Distance: 196.65 cm
Distance: 197.53 cm
Distance: 197.55 cm
Distance: 197.05 cm
Distance: 197.94 cm
Distance: 197.53 cm
Distance: 198.01 cm
Distance: 197.55 cm
Distance: 197.53 cm
Distance: 197.54 cm
Distance: 197.98 cm
Distance: 197.95 cm
Distance: 197.04 cm
Distance: 197.09 cm
Distance: 196.63 cm
Distance: 197.09 cm
Distance: 198.45 cm
Distance: 198.02 cm
Distance: 197.55 cm
Distance: 202.98 cm
Distance: 11.49 cm
Distance: 10.95 cm
Distance: 10.10 cm
Distance: 10.27 cm
Distance: 198.40 cm
Distance: 197.60 cm
Distance: 197.11 cm
Distance: 197.99 cm
Distance: 197.07 cm

回家作業

- 試著設計利用溫度感測的數值，與LED的PWM做結合，至少1~10顆LED顯示出漸層變化用來表示三個不同層次的溫度，例如： 大於30度 顯示顏色比較亮，小於20度，顯示比較暗，若用越多顆LED顯示溫度越細越好，顯示的方式越有創意越好。
- 試著設計用距離感測器偵測距離數值，與LED的PWM做結合，至少1~10顆LED顯示出距離值不同，當距離越近時，只有漸變1顆LED，距離越遠漸變越多顆LED，與距離感測器即時互動時，需考慮LED顯示的順暢度。
- 控制繼電器與距離感測器做多段控制，例如：小於10公分開關一次，大於150公分開關10次，中間的開關次數隨感測物體的距離而改變，繼電器開關次數時間間格也隨距離遠近而改變，繼電器的負載可用LED燈泡或其他元件(馬達、喇叭、電磁閥開關、等)
- 以上3項作業，請任選 1 項，請上傳到iLMS課程網站，以小組為單位上傳以下內容：
  - 程式碼
  - 作業的DEMO影片(或youtube連接，點選後須可以直接撥放)
  - 作業的電路板連線圖的照片數張(需清楚拍照)