三版序

這幾年在資策會、電腦公會與大專院校教授物聯網與 OpenCV 課程，深刻覺得現今這些技術的發展與當年唸書時不可同日而語，一塊小小的主機板就可以執行完整 UNIX 系統，並且可以做好多事情。各種硬體也變成一個一個的元件或是模組，插到板子上再寫點程式碼就可以驅動與整合他們。要自己動手做點小東西，已經不太需要多麼高深的電子電路背景，任誰都可以買點硬體元件回來，然後找份好的參考文件照著操作就會動了。

也發現，如果在物聯網的課程中加進了攝影機主題的話，會有更多有創意的物聯網系統出現，畢竟讓電腦看的懂人看的懂的東西，是更有挑戰與更有成就感的領域。試想，當電腦看得懂的時候，就可以自動發出訊號去控制硬體元件做事情，例如人臉辨識開門、偵測停車位是否有空位、用車牌辨識繳費，或是產品線上用來偵測瑕疵品並記錄分析改善良率等一堆的應用。

物聯網涵蓋的領域太廣，好的參考書籍又很多，想要挑一本書來涵蓋授課範圍實在很困難，不太可能全部囊括在一門課裡面，所以就乾脆自己動手，把會用到的部分整理出來自己出本書，讓願意花錢花時間來聽我課的學員有一本非常貼近上課內容的參考用書，即使課程結束也很容易透過書中內容回想起上課所學，達到事半功倍效果。即使沒上過我課的讀者，相信這本書也可以幫助您在面對相關問題時，可以照著書中的範例快速解決，不會卡在某一個關鍵地方太久，這也是我寫這本書所期待的。

謝謝碁峰資訊協助本書付梓，也希望讀者們學得愉快，不要有東西燒掉 :p

想要進入物聯網與機器視覺領域，您手上自然需要先有一些不算昂貴的設備，以下整理了一份書中所有用到的硬體元件清單供您參考。

樹莓派主板以及攝影機單價較高，建議您可以上網搜尋，找到合適的銷售商家。其他零散的硬體元件（如表列第 3~ 第 25 項），除了可以到電子材料行或網站上一一找到外，我也幫讀者準備了一份完整的零件包，如果您需要的話可以在研蘋果官網 https://www.chainhao.com.tw 找到購買連結。

https://www.chainhao.com.tw

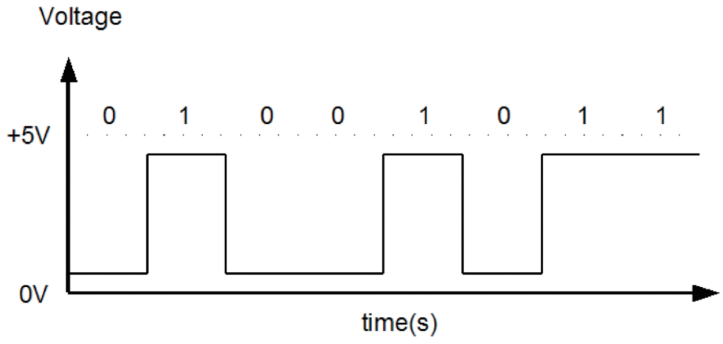
編號	品項
1.	樹莓派 4B
2.	樹莓派官方攝影機 Camera V2
3.	麵包板
4.	杜邦線（公公、公母、母母）
5.	220Ω 與 10KΩ 電阻
6.	LED
7.	微動開關
8.	SG-90 舵機
9.	無源蜂鳴器模組
10.	HC-SR04 超音波感測模組
11.	HC-SR501 紅外線移動感測模組
12.	DHT11 溫濕度感測模組
13.	繼電器模組

編號	品項
14.	七段顯示器
15.	ADXL345 三軸加速儀模組
16.	MAX7219 LED 矩陣模組
17.	2x16 LCD 螢幕
18.	全彩 LED 燈條
19.	光敏電阻
20.	火焰感測
21.	MQ-2 氣體感測模組
22.	雨滴感測模組
23.	電容式土壤濕度感測模組
24.	74HC595 晶片
25.	MCP3008 晶片

樹莓派 2

2-7 數位感測器

數位感測器傳出的訊號只有高電位與低電位兩種訊號，例如按鈕按下去後得到高電位訊號，鬆開的時候得到低電位訊號。若高電位訊號以 1 代表，低電位訊號以 0 代表，感測器可以傳出一系列的 0/1 組合，例如 01000001，再把這一串 0/1 組合的數字以每 8 個為一組後轉成 10 進位，就可以得到 65 這個數字。也許 65 代表了當時的相對濕度是 65%，或是經由 ASCII 表得到英文字母 A。不論是哪一種，0/1 的組合可以讓感測器產出任何我們想要得到的資訊。



2-7-1 舵機（Servo）

舵機是一種透過 PWM 訊號控制旋轉角度的馬達。一般來說舵機有兩種規格，一種是可以 360 度旋轉的，另一種則是有最大轉動角度。拿到一個舵機後必須先看規格書，有幾個數據必須先查到，以 SG90 這個在許多電子玩具上常見的舵機而言，規格書中必須要知道的數據請見下表：

黑線或棕線	GND
紅線	5V
橘線	GPIO
最大轉動角度	180 度
頻率	50Hz
脈衝寬度	500us ~ 2400us
轉動速度	60 度 / 0.1s

根據最大轉動角度與脈衝寬度資料，我們必須先算出 PWM 在頻率 50Hz 的情況下要舵機轉到 0 度時的佔空比以及 180 度的佔空比，公式如下：

$$0\text{ 度：}\frac{500\times10^{-6}}{50^{-1}}\times100=2.5\%$$

$$180\text{ 度：}\frac{2400\times10^{-6}}{50^{-1}}\times100=12\%$$

根據以上計算的結果，佔空比與角度之間的轉換公式為：

$$f(x)=\frac{(12-2.5)}{180}\times x+2.5$$

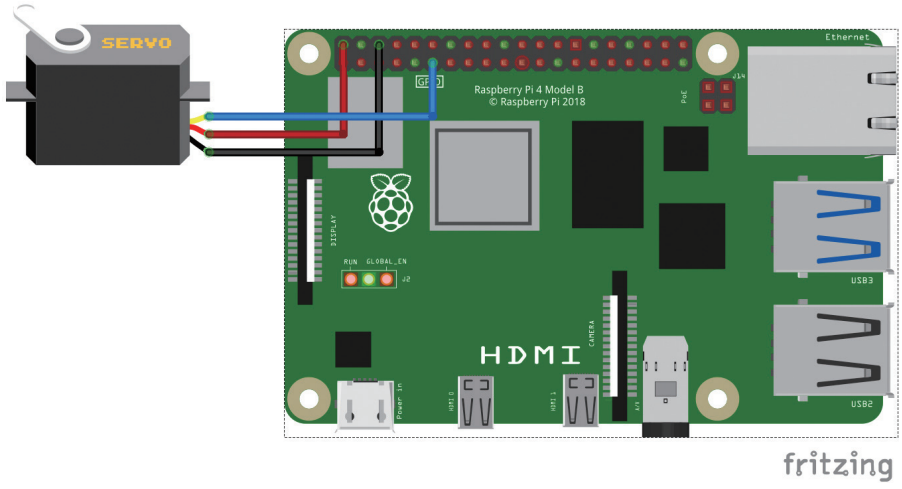
例如我們想要讓舵機轉到 60 度，計算 $f(60)$ 的結果為：

$$f(60)=\frac{(12-2.5)}{180}\times 60+2.5=5.67$$

數字 5.67 就是我們要讓舵機轉到 60 度的佔空比。

步驟與說明

1. 將舵機接到樹莓派上，並且將一片塑膠葉片裝到舵機上，這樣待會舵機轉動時才看的到轉到的位置。

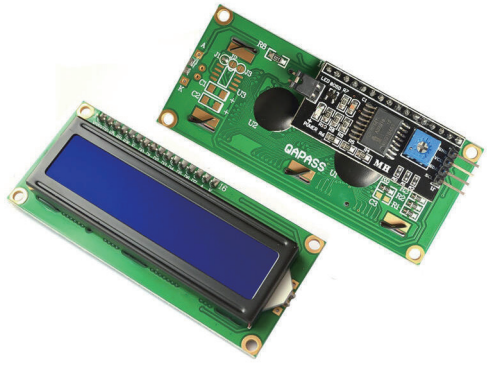


2 撰寫程式碼。一開始讓舵機轉到 0 度，然後轉到 180 度，最後停在 90 度。

```
import RPi.GPIO as GPIO
import time

pin = 17
GPIO.setmode(GPIO.BCM)
GPIO.setup(pin, GPIO.OUT)
p = GPIO.PWM(pin, 50)

# degree 0
p.start(2.5)
time.sleep(0.4)
# degree 180
p.ChangeDutyCycle(12)
time.sleep(0.4)
# degree 90
p.ChangeDutyCycle(7.25)
time.sleep(0.4)

p.stop()
GPIO.cleanup()
```

3 執行看看。

2-7-9 LCD 螢幕



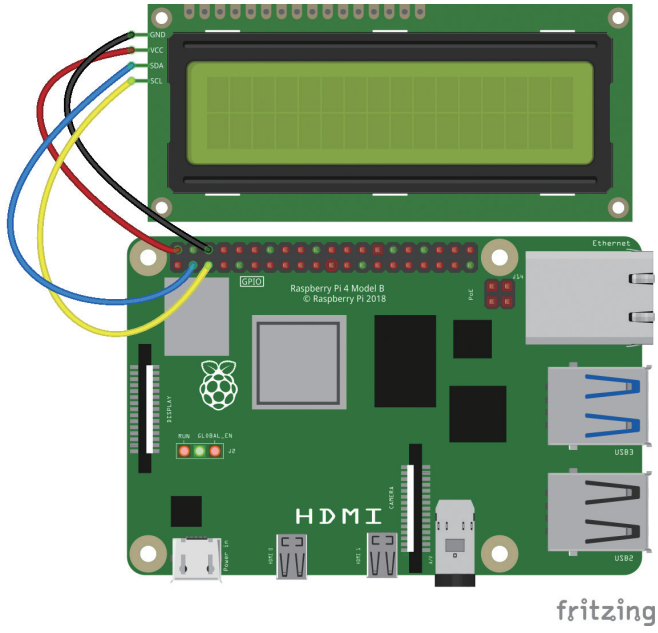
LCD 螢幕規格有很多種，這裡要介紹的一般電子材料行很常見，網路上也很容易買到的規格，如上圖。特徵是單色，可顯示 2 行每行 16 個字並且有背光，模組背後加上了一片 LCM1602 的控制板，可以讓樹莓派透過 I2C 協定來操作。

將資料顯示到螢幕上是很有趣的一件事情，讓樹莓派不需要接螢幕就可以顯示一些簡單的資料，例如溫濕度資料，所以這個模組我們就不要太辛苦的自己寫驅動程式，安裝一個第三方函數庫即可。

步驟與說明

1. 接線。這個 LCD 模組使用 I2C 協定，因此接線方式是固定的。

ADXL345	RPi
VCC	5V
GND	GND
SDA	GPIO2
SDL	GPIO3



2 安裝第三方函數庫。

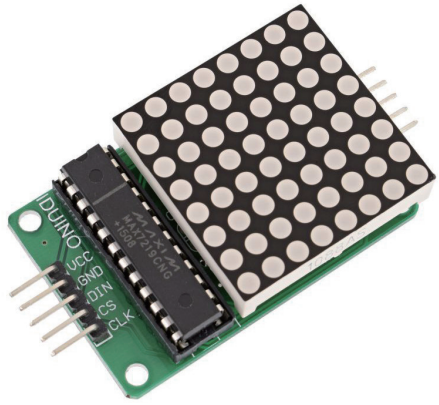
```
$ pip3 install rpi-lcd
```

3 程式如下。其中 lcd.text() 函數中的第二個參數代表字串要顯示在第幾行。

```
from rpi_lcd import LCD
import time

lcd = LCD()
lcd.text('Hello World!', 1)
lcd.text('Raspberry Pi', 2)

time.sleep(5)
lcd.clear()
```

4 執行看看。

2-7-10 LED 矩陣（MAX7219）

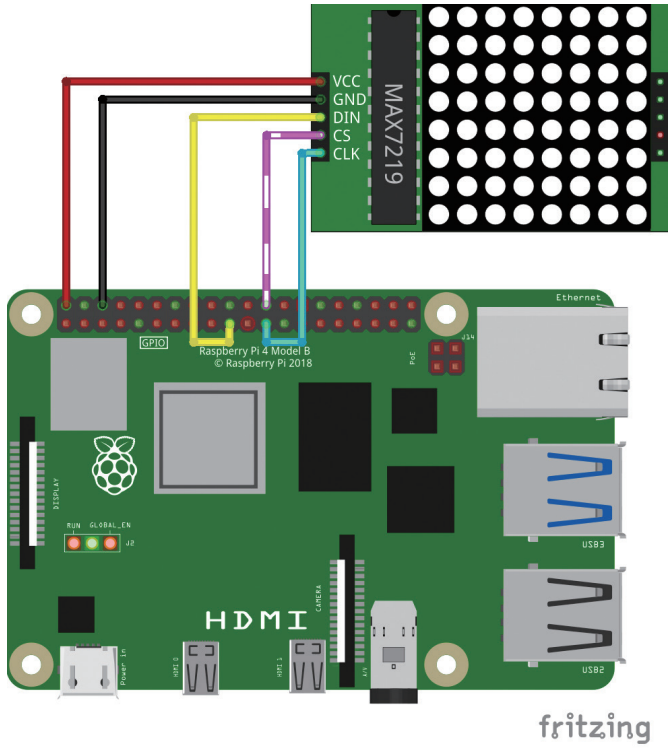


之前我們看過如何使用 74HC595 晶片控制一顆七段顯示器，這個單元我們要使用 MAX7219 晶片控制 64 顆 LED 矩陣，也相當於 8 顆七段顯示器。MAX7219 是透過 SPI 協定與樹莓派溝通，因此必須先確認樹莓派是否載入了 SPI 驅動程式。執行 sudo raspi-config 後選 Interfacing Options 就可以確認了。建議讀者買這種晶片與 LED 矩陣已經組裝好的模組，使用上方便許多。跟 I2C 協定一樣，我們自己 DIY 程式碼來控制這一顆晶片。

步驟與說明

1. 接線。由於是 SPI 協定，因此 GPIO 必須固定不可任意選取，這裡我們使用樹莓派的 SPI0。

ADXL345	RPi
VCC	5V
GND	GND
DIN	GPIO10（MOSI）
CS	GPIO8（CE0）
CLK	GPIO11（SCLK）



2 匯入函數庫、開啟 SPI 通道必且設定 MAX7219 晶片的暫存器位址。

```
import spidev

spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 10000000

NOOP        = 0x0
DECODEMODE  = 0x9
INTENSITY   = 0xA
SCANLIMIT   = 0xB
SHUTDOWN    = 0xC
DISPLAYTEST = 0xF
```

補充說明

暫存器位址	用途	說明
0x0	No-OP	多個 MAX7219 串連時，設定哪個晶片不作用。
0x1～0x8	資料 0～7	
0x9	解碼模式	參考規格書。0x0 代表可輸出任意圖案。
0xA	顯示亮度	0x0～0xF。0x0 最暗，0xF 最亮。
0xB	掃描限制	參考規格書
0xC	是否關機	0: 關機 , 1: 開機
0xF	顯示測試	0: 正常 , 1: 全亮

3 編碼要顯示的圖案，此圖案為一顆愛心。

```
love = (
    0b01000010,
    0b11100111,
    0b11111111,
    0b11111111,
    0b01111110,
    0b00111100,
    0b00011000,
    0b00000000
)
```

4 實作用來初始化 MAX7219 的函數

```
def init():
    send(DISPLAYTEST, 0)
    send(SCANLIMIT, 7)
    send(INTENSITY, 8)
    send(DECODEMODE, 0)
    send(SHUTDOWN, 1)
```

5 實作透過 SPI 送資料到 MAX7219 的函數。

```
def send(reg, data):
    spi.writebytes([reg, data])
```

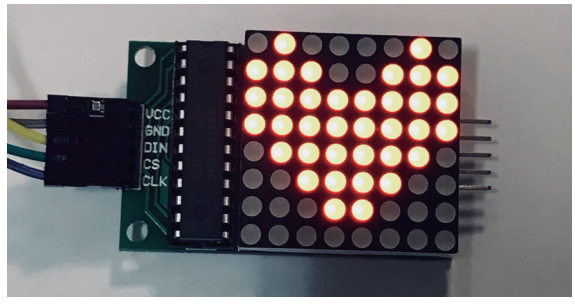
6 實作在 LED 矩陣上顯示圖案的函數。

```
def show(graph):
    for i in range(8):
        send(i + 1, graph[i])
```

7 顯示資料最後關閉 LED 矩陣。

```
def main():
    init()
    show(love)
    input('enter to stop')
    send(SHUTDOWN, 0)

main()
```

8 執行看看。



2-7-11 多個 MAX7219 模組串接

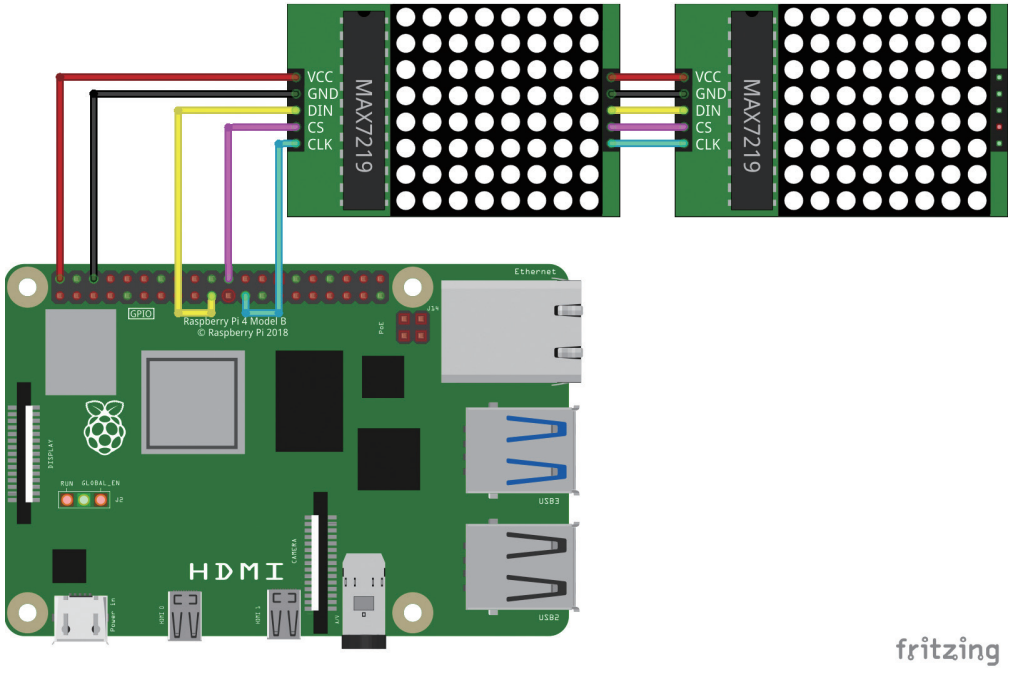
MAX7219 模組可以透過頭尾相連的方式串接多個，形成更大塊的 LED 矩陣。程式寫法跟單一一個模組不同，主要是模組上的 Chip Select（CS）接腳不可以使用樹莓派上的 CE0 接腳，必須要接到另一個 GPIO 上，然後需要在程式中自行控制 CS 接腳的電位訊號。

多個 MAX7219 模組串接時，每筆資料輸出必須連續輸出多次，第一次輸出的資料會流到離樹莓派最遠那一個模組，最後一次輸出的資料會流到離樹莓派最近的那一個模組。

決定哪一個模組要顯示資料，是透過 NOOP 暫存器，這個暫存器被設定的 LED 矩陣模組就不會更新資料。NOOP 可視為「垃圾桶」，將資料丟到垃圾桶去，畫面自然就不會更新了。

步驟與說明

1. 接線，注意 MAX7219 模組上的 CS 腳已經換到 GPIO25 了，任何一個 GPIO 都可以除了 GPIO8 外。



2 匯入函數庫、設定 CS 接腳、開啟 SPI 通道並且設定 MAX7219 晶片的暫存器位址。

```
import RPi.GPIO as GPIO
import spidev

pinCS = 25
GPIO.setmode(GPIO.BCM)
GPIO.setup(pinCS, GPIO.OUT)

spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 10000000

NOOP        = 0x0
DECODEMODE  = 0x9
INTENSITY   = 0xA
SCANLIMIT   = 0xB
SHUTDOWN    = 0xC
DISPLAYTEST = 0xF
```

3 設定兩個圖形的編碼矩陣。

```
love = (
    0b01000010,
    0b11100111,
    0b11111111,
    0b11111111,
    0b01111110,
    0b00111100,
    0b00011000,
    0b00000000
)

smile = (
    0b00111100,
    0b01000010,
    0b10100101,
    0b10000001,
    0b10100101,
    0b10011001,
```

```
    0b01000010,
    0b00111100
)
```

4 實作初始化函數。

```
def init():
    send(DISPLAYTEST, 0)
    send(SCANLIMIT, 7)
    send(INTENSITY, 8)
    send(DECODEMODE, 0)
    send(SHUTDOWN, 1)
```

5 實作透過 SPI 送資料到 MAX7219 的函數。which=[1, 1] 表示有兩個 MAX7219 模組串接，如果有三個串接的話，設定 which=[1, 1, 1]，以此類堆。

```
def send(reg, data, which=[1, 1]):
    GPIO.output(pinCS, 0)

    for p in which:
        if p == 1:
            spi.writebytes([reg, data])
        else:
            spi.writebytes([NOOP, data])

    GPIO.output(pinCS, 1)
```

6 實作在 LED 矩陣上顯示圖案的函數。

```
def show(graph, which):
    for i in range(8):
        send(i + 1, graph[i], which)
```

7 讓兩個 LED 矩陣分別顯示笑臉與愛心，並且每一秒鐘交換一次。

```
def main():
    import time
    init()
    try:
        while True:
            show(love, [0, 1])
            time.sleep(1)
            show(smile, [1, 0])
            time.sleep(1)
    except KeyboardInterrupt:
        pass
    send(SHUTDOWN, 0)

main()
```

8 執行看看。