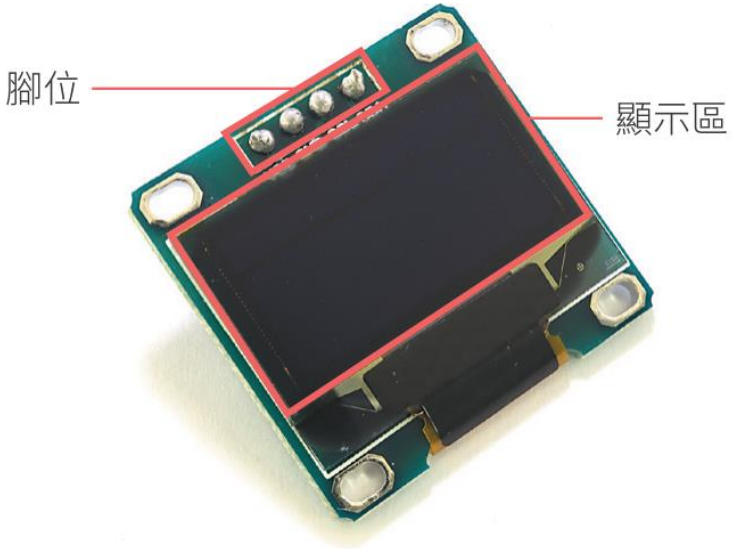
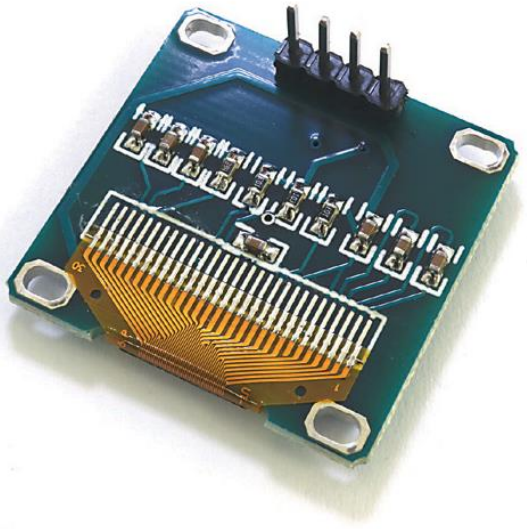
Ch02 顯示文字與圖形 – OLED 模組

2-1 認識 OLED 模組



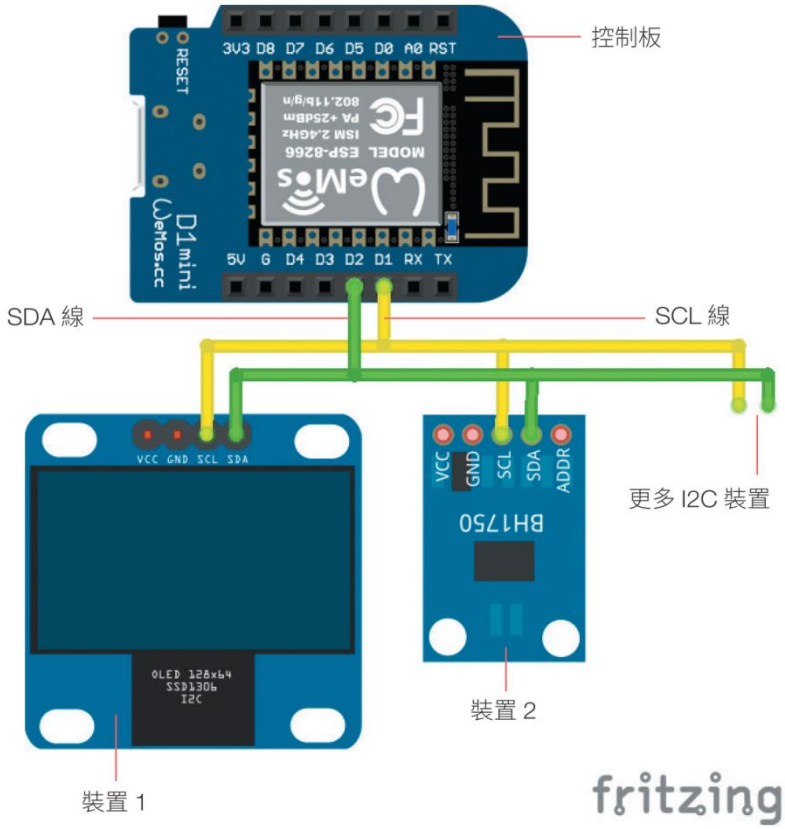


OLED (Organic LED, 有機 LED)，目前普遍用於手機和電視螢幕。

這裡使用 0.96 吋 OLED模組，驅動晶片 為 SSD1306，解析度 128x64 像素。

2-1 認識 OLED 模組

D1 mini 必需透過 I2C 通訊協定來控制這種 OLED 模組。



I2C 通訊協定

- 兩個裝置之間需要用兩條線來當作通訊的橋梁，這兩條橋的名字分別是：
  - SDA (Serial Data Line, holds Data or address signal)，簡單來說就是用來傳輸資料的線路，雙向傳輸，收發都靠這條。
  - SCL (Serial Clock Line, holds Clock signal)，用來同步通訊時序的線，這條線的震盪速度就代表傳輸速度。

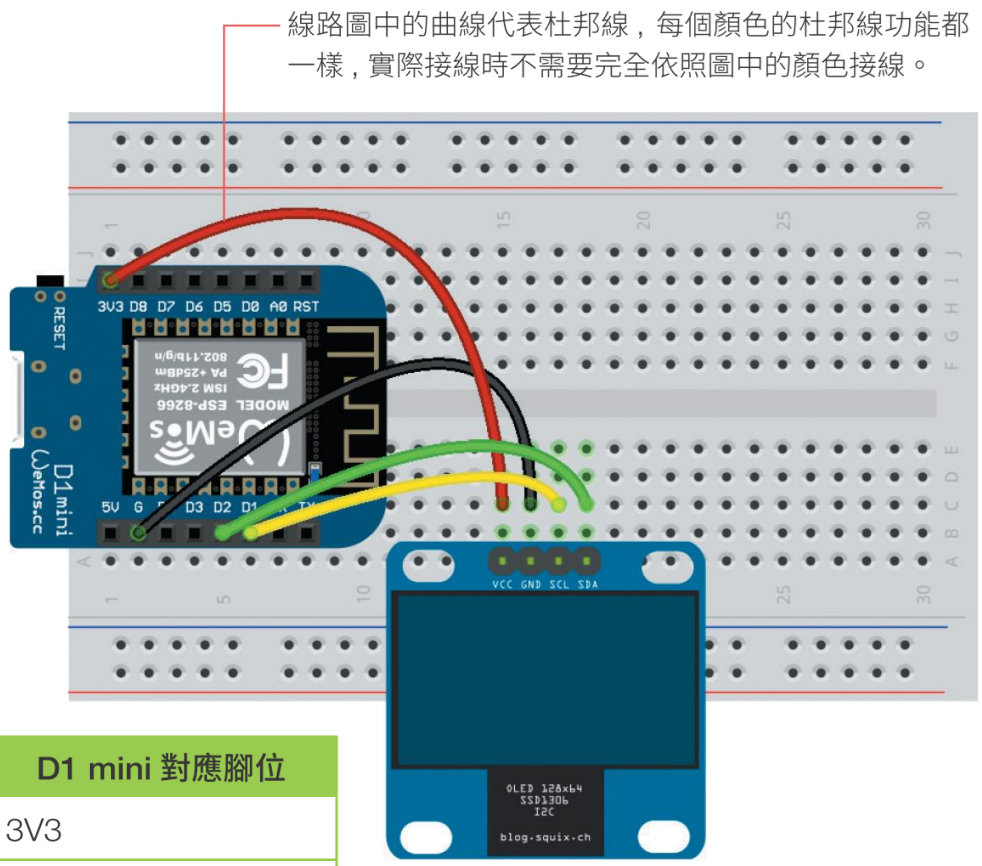
2-2 在 OLED 顯示文字

- **Lab03**

使用 OLED 模組顯示文字	
實驗目的	讓 OLED 顯示一行文字
材料	• D1 mini　　• OLED 模組

2-2 在 OLED 顯示文字

- 線路圖



OLED 腳位	意義	D1 mini 對應腳位
VCC	電源	3V3
GND	接地	G
SCL	串列時脈線	D1 (5 號腳位)
SDA	串列資料線	D2 (4 號腳位)

2-2 在 OLED 顯示文字

- **設計原理**

先建立 I2C 物件，用 I2C 物件建立 OLED 物件。

第一步 匯入相關函式庫 :

```
# 匯入 machine 的 Pin 和 I2C 子函式庫
from machine import Pin, I2C
# 匯入 OLED 函式庫
from ssd1306 import SSD1306_I2C
```

2-2 在 OLED 顯示文字

建立 I2C 物件 :

```
# 指定 SCL 在 5 號腳位(D1), SDA 在 4 號腳位(D2)
I2c = I2C(scl=Pin(5), sda=Pin(4))
```

建立 OLED 物件 :

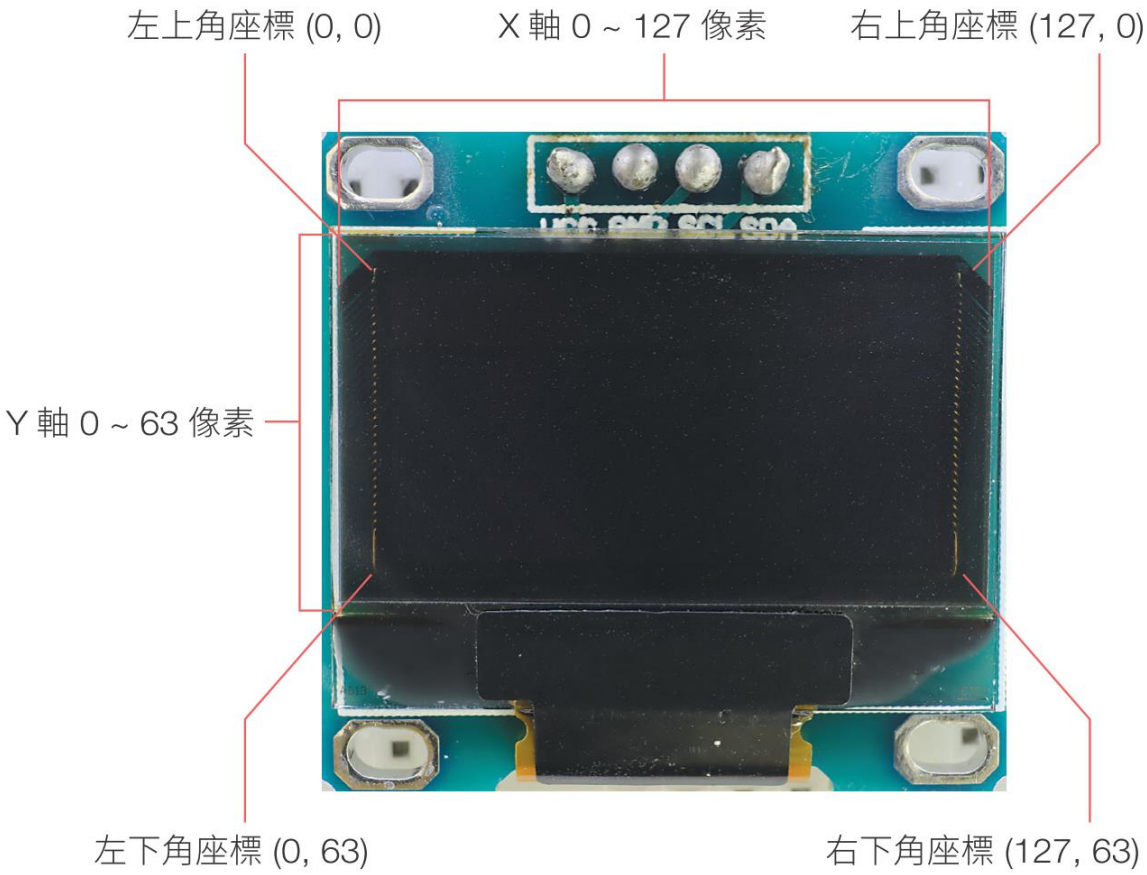
```
# 指定寬 128 像素, 高 64 像素, 以及要使用的 I2C 物件
oled = SSD1306_I2C(128, 64, i2c)
```

如此便能用 oled.text() 方法在 OLED 上顯示文字 :

```
oled.text("I love PYTHON!", 0, 0) # 在座標(0, 0)顯示文字
oled.show() # 套用改變
```

2-2 在 OLED 顯示文字

OLED 的座標軸如下：



2-2 在 OLED 顯示文字

- 程式設計

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c = I2C(scl=Pin(5), sda=Pin(4))
oled = SSD1306_I2C(128, 64, i2c)

oled.text("I love PYTHON!", 0, 0)
oled.show()
```

2-2 在 OLED 顯示文字

- 實測

2-3 顯示不斷更新的資料

設計感測器應用時，通常需要顯示持續變化的資料值。

- **Lab04**

在 OLED 不斷顯示和更新資料	
實驗目的	讓 OLED 顯示 D1 mini 開機後經過的時間
材料	• D1 mini • OLED 模組

- **線路圖：**

同 Lab 03

2-3 顯示不斷更新的資料

- **設計原理**

使用 utime 函式庫底下的ticks_ms() 函式，
取得 D1 mini 開機後系統經過的時間：

```
import utime # 匯入時間計算函式庫
system_time = utime.ticks_ms()     # 取得系統開機後經過毫秒數
oled.text(str(system_time), 0, 0) # 在 OLED 印出資料
oled.show() # 讓 OLED 顯示資料
```

變數 system_time 得到數值，但 oled.text() 只能顯示文字資料，
須用 str() 將數值轉為文字型態。

2-3 顯示不斷更新的資料

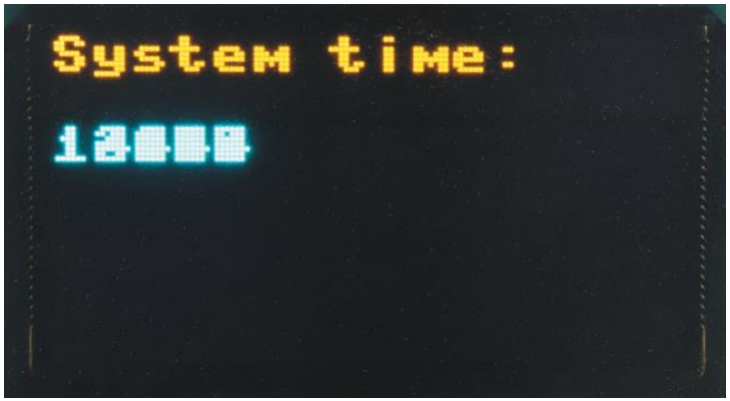
用 while 迴圈來重複執行程式，連續顯示慢慢增加的毫秒數：

```
while True: # 無窮迴圈
    system_time = utime.ticks_ms()
    oled.text(str(system_time), 0, 0)
    oled.show()
    utime.sleep_ms(100) # 每次停頓 100 毫秒
```

控制板和 OLED 通訊需要時間，
故在迴圈內加入 100 毫秒的時間延遲。

2-3 顯示不斷更新的資料

但執行後會發現字全部疊在一起：



因此每次顯示新資料前必須先清除畫面：

```
oled.fill(0)
```

oled.fill() 的功能是把整個螢幕的像素填滿或清除

2-3 顯示不斷更新的資料

加上說明文字：

```
oled.text("System time: " + str(system_time), 0, 0)
```

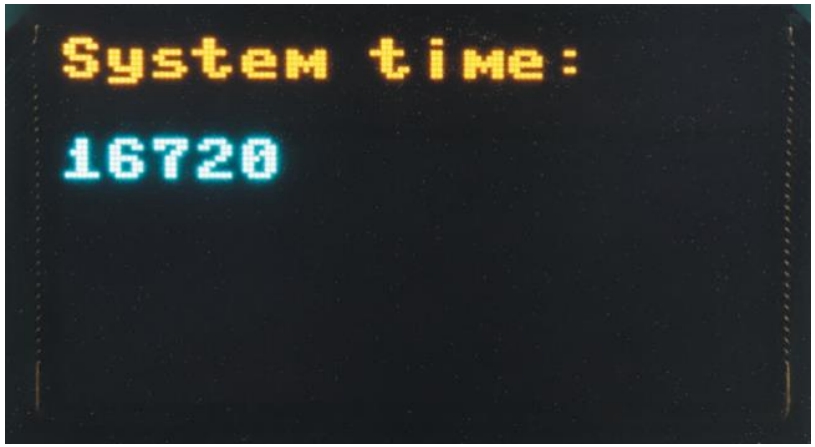
分成兩行，避免文字太長，超出 OLED 顯示範圍：

```
oled.text("System time: ", 0, 0)
oled.text(str(system_time) + " ms", 0, 8)
```

OLED 內建字體高度 8 像素，建議第二行 Y 座標從 8 開始 。

2-3 顯示不斷更新的資料

以下程式把時間資料顯示在 Y 座標 16。顯示結果如下圖：



2-3 顯示不斷更新的資料

- 程式設計

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import utime

i2c = I2C(scl=Pin(5), sda=Pin(4))
oled = SSD1306_I2C(128, 64, i2c)

while True:

    system_time = utime.ticks_ms()

    oled.fill(0)
    oled.text("System time: ", 0, 0)    # 顯示於第一行
    oled.text(str(system_time), 0, 16) # 顯示於第三行
    oled.show()

    utime.sleep_ms(100)
```

- 實測

執行程式，OLED 上出現不斷更新的系統開機後毫秒數。

2-4 畫個愛心

- **Lab05**

在 OLED 上畫個愛心	
實驗目的	讓 OLED 顯示自訂的 8x8 愛心圖案
材料	• D1 mini • OLED 模組

- **線路圖**

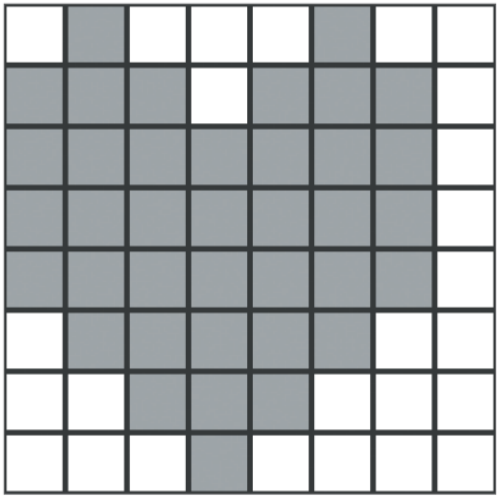
同 Lab 03 。

2-4 畫個愛心

- **設計原理**

要在 OLED 模組上畫自訂圖案，必須先定義圖案的像素資料，並轉換成 OLED 可用來繪圖的影格緩衝區 (framebuffer) 物件。

例如：



將有像素的格子視為 1，沒像素的視為 0，可轉成以下表格：

(LSB)

	行 0 最高位元	行 1	行 2	行 3	行 4	行 5	行 6	行 7 最低位元		2 進位值 最高位元 ← 最低位元	16 進位值
列 0 最低位元	0	1	0	0	0	1	0	0	→	01000100	0x44
列 1	1	1	1	0	1	1	1	0	→	11101110	0xee
列 2	1	1	1	1	1	1	1	0	→	11111110	0xfe
列 3	1	1	1	1	1	1	1	0	→	11111110	0xfe
列 4	1	1	1	1	1	1	1	0	→	11111110	0xfe
列 5	0	1	1	1	1	1	0	0	→	01111100	0x7c
列 6	0	0	1	1	1	0	0	0	→	00111000	0x38
列 7 最高位元	0	0	0	1	0	0	0	0	→	00010000	0x10
	↓	↓	↓	↓	↓	↓	↓	↓			
(LSB) 2 進位值 最低位元 ↓ 最高位元	00011110	00111111	01111110	11111100	01111110	00111111	00011110	00000000			
16 進位值	0x1e	0x3f	0x7e	0xfc	0x7e	0x3f	0x1e	0x00			

2-4 畫個愛心

每一行或每一列的 0 或 1可組合成 2 進位數

方向有兩種，最低位元優先排列 (Least Significant Bit, LSB) 以及最高位元優先排列 (Most Significant Bit, MSB) 。

若是水平方向，

LSB 排法會和圖案本身一樣，使第 0 列的 2 進位值是 01000100，換算成 16 進位值即為 0x44 。

2-4 畫個愛心

這裡使用水平方向 LSB 排法。
換算出的 8 個 16 進位值，就是 8x8 愛心圖案的像素資料 :

```
pic_list= [0x44, 0xee, 0xfe, 0xfe, 0xfe, 0x7c, 0x38, 0x10]
```

pic_list 是個串列，用 [] 中括號將內含值包起來。
0x44 是陣列裡的第 0 項，直到第 7 項為 0x10，一共 8 項。

2-4 畫個愛心

將 pic_list 轉換成位元組陣列 (bytearray)，再轉為影格緩衝區 (framebuffer) 物件 :

```
import framebuf # 匯入 framebuf 函式庫
# 前面的 pic_list 轉成 bytearray
pic_buffer = bytearray(pic_list)
# 再轉成 framebuffer
pic = framebuf.FrameBuffer(pic_buffer, 8, 8,
                           framebuf.MONO_HLSB)
```

2-4 畫個愛心

在 framebuf.FrameBuffer()方法中，
第 1 個參數是包含像素資料的位元組陣列，
第 2、3 個參數是圖案寬和高。
第 4 參數則是圖案像素資料是單色(MONO)、水平(H) 並以最低位元優先排列 (LSB)。

MONO_HLSB	單色，水平，最低位元優先排列
MONO_HMSB	單色，水平，最高位元優先排列
MONO_VLSB	單色，垂直，最低位元優先排列
MVLSB	同 MONO_VLSB

2-4 畫個愛心

若以不同方向或順序產生圖像資料，要搭配正確的參數才能畫出正確結果。例如若改用前面的垂直 LSB 排列方式:

```
pic_list = [0x1e, 0x3f, 0x7e, 0xfc, 0x7e, 0x3f, 0x1e, 0x00]
pic_buffer = bytearray(pic_array)
```

而呼叫 framebuf.FrameBuffer() 方法時就變成

```
pic = framebuf.FrameBuffer(pic_buffer, 8, 8,
                           framebuf. MONO_VLSB)
```

最後轉換完成的 framebuffer 物件便可用 oled.blit() 方法送到 OLED 顯示出來 :

```
oled.blit(pic, 0, 0) # 在座標 (0, 0) 印出圖案
oled.show()
```

2-4 畫個愛心

- 程式設計

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import framebuf

oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))

pic_list = [0x1e, 0x3f, 0x7e, 0xfc, 0x7e, 0x3f, 0x1e, 0x00]

pic_buffer = bytearray(pic_list)
pic = framebuf.FrameBuffer(pic_buffer, 8, 8,
                           framebuf.MONO_VLSB)

oled.text("I", 8, 8)
oled.blit(pic, 20, 8)
oled.text("PYTHON", 32, 8)
oled.rect(4, 4, 80, 16, 1) # 畫方框
oled.show()
```

2-4 畫個愛心

2-5 其它繪圖功能

OLED 還有許多內建繪圖功能：

```
oled.fill(1)          # 點亮所有畫素
oled.invert(1)        # 螢幕反相
oled. contrast(255) # 設定對比度 (0~255)
oled.pixel(x, y, 1) # 點亮座標 x, y 的像素
oled.hline(x, y, len, 1) # 畫水平線, 起點座標 x, y 長度 len
oled.vline(x, y, len, 1) # 畫垂直線, 起點座標 x, y 長度 len
oled.line(x1, y1, x2, y2, 1) # 畫直線, 從座標 x1, y1 到 x2, y2
oled.rect(x1, y1, x2, y2, 1) # 畫方框, 兩角座標 x1, y1 到 x2, y2
oled.fill_rect(x1, y1, x2, y2, 1) # 畫實心方塊, 兩角座標 x1, y1
                                    到 x2, y2
oled.scroll(x, y)               # 捲動畫面 x, y 像素
```

許多方法後面的 1 代表畫出顏色，若設為 0 就等於清除顏色 。