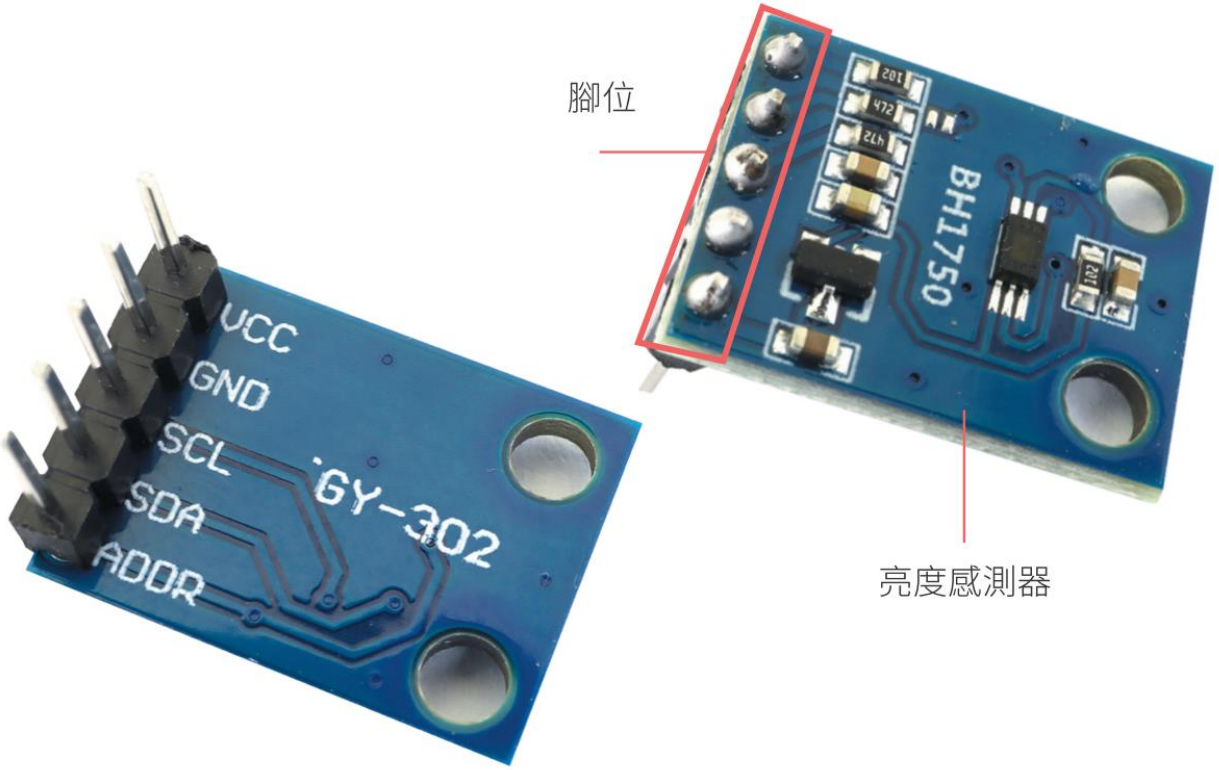
Ch04 偵測亮度 – 亮度感測模組

4-1 認識亮度感測模組

BH1750 亮度感測器能測量環境的照度；
照度是物體每單位面積受到的光線照射程度。



4-1 認識亮度感測模組

照度的衡量單位是勒克斯 (lux) 。

BH1750 模組能傳回的照度值最大為 65535 lux 。

照明條件	照度 (lux)
直射陽光	32000~100000
大白天 (非直接陽光)	10000~25000
陰天 / 電視台攝影棚	1000
晴朗日落	400
辦公室	320~500
大陰天	100
晴朗夜晚的滿月	0.05~0.3

4-2 環境亮度監測

- **Lab10**

環境照度計			
實驗目的	從亮度感測模組讀取環境亮度		
材料	• D1 mini	• OLED 模組	• 亮度感測模組

4-2 環境亮度監測

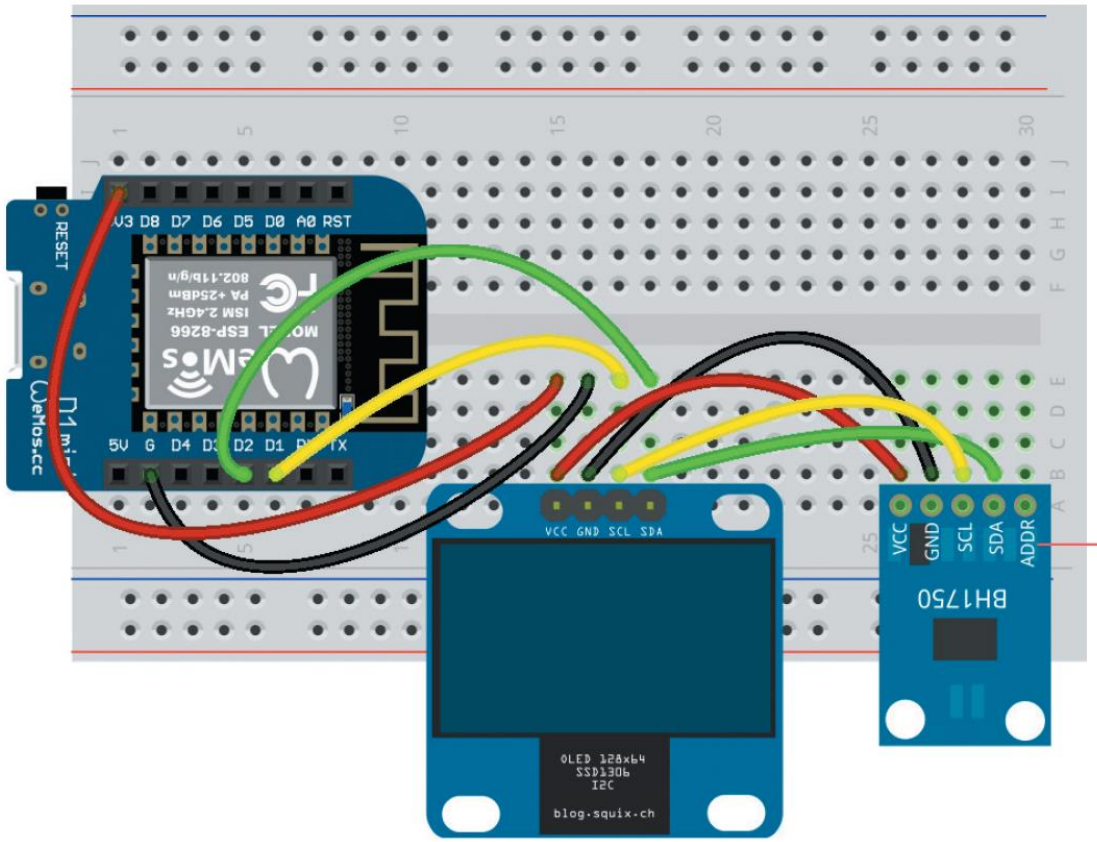
- 接線圖

亮度模組腳位	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)
ADDR	位址切換	不接線

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)

4-2 環境亮度監測

- 接線圖



圖中亮度感測模組上的字樣為方便接線所做的標示，實體模組插在麵包板上並不會有這些字樣。

4-2 環境亮度監測

- **設計原理**

先在 D1 mini 中安裝專用的函式庫後，才能匯入BH1750 模組:

```
import bh1750fvi
```

因 OLED 和 BH1750 需要透過 I2C 控制，得建立共用 I2C 物件 :

```
from machine import Pin, I2C
i2c = I2C(scl=Pin(5), sda=Pin(4))
```

完成後 便能讀取 BH1750 偵測到的環境亮度 :

```
light_level = bh1750fvi.sample(i2c, mode=0x23)
```

4-2 環境亮度監測

第一個參數是 i2c 物件，
第二個 mode 參數是用來選擇 BH1750 的讀取模式：

模式代碼	功能	資料精確度	讀取時間
0x20	高精確模式 1	1 lux	120 毫秒
0x21	高精確模式 2	0.5 lux	120 毫秒
0x23	低精確模式	4 lux	16 毫秒

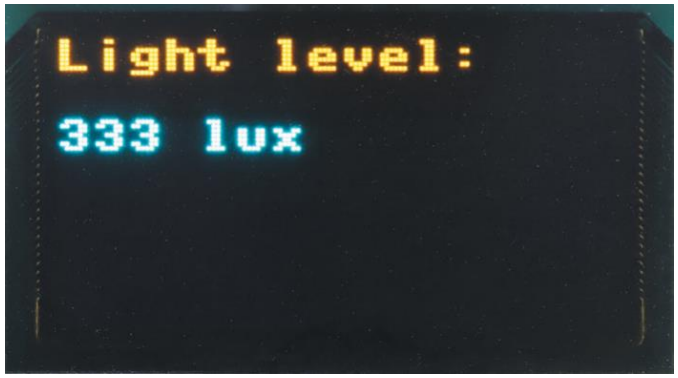
4-2 環境亮度監測

- 程式設計

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import bh1750fvi, utime

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

while True:

    light_level = bh1750fvi.sample(i2c, mode=0x23)

    oled.fill(0)
    oled.text("Light level:", 0, 0)
    oled.text(str(light_level) + " lux", 0, 16)
    oled.show()
    print("偵測亮度: " + str(light_level) + " lux")
```

4-2 環境亮度監測

- 實測



偵測亮度: 386 lux ←—— 測試環境正常亮度
偵測亮度: 493 lux
偵測亮度: 2470 lux
偵測亮度: 36073 lux ←—— 手機燈貼近亮度
偵測亮度: 3843 lux
偵測亮度: 17233 lux
偵測亮度: 2860 lux
偵測亮度: 363 lux
偵測亮度: 0 lux ←—— 用手指蓋住感測器
偵測亮度: 0 lux

4-3 照明警示

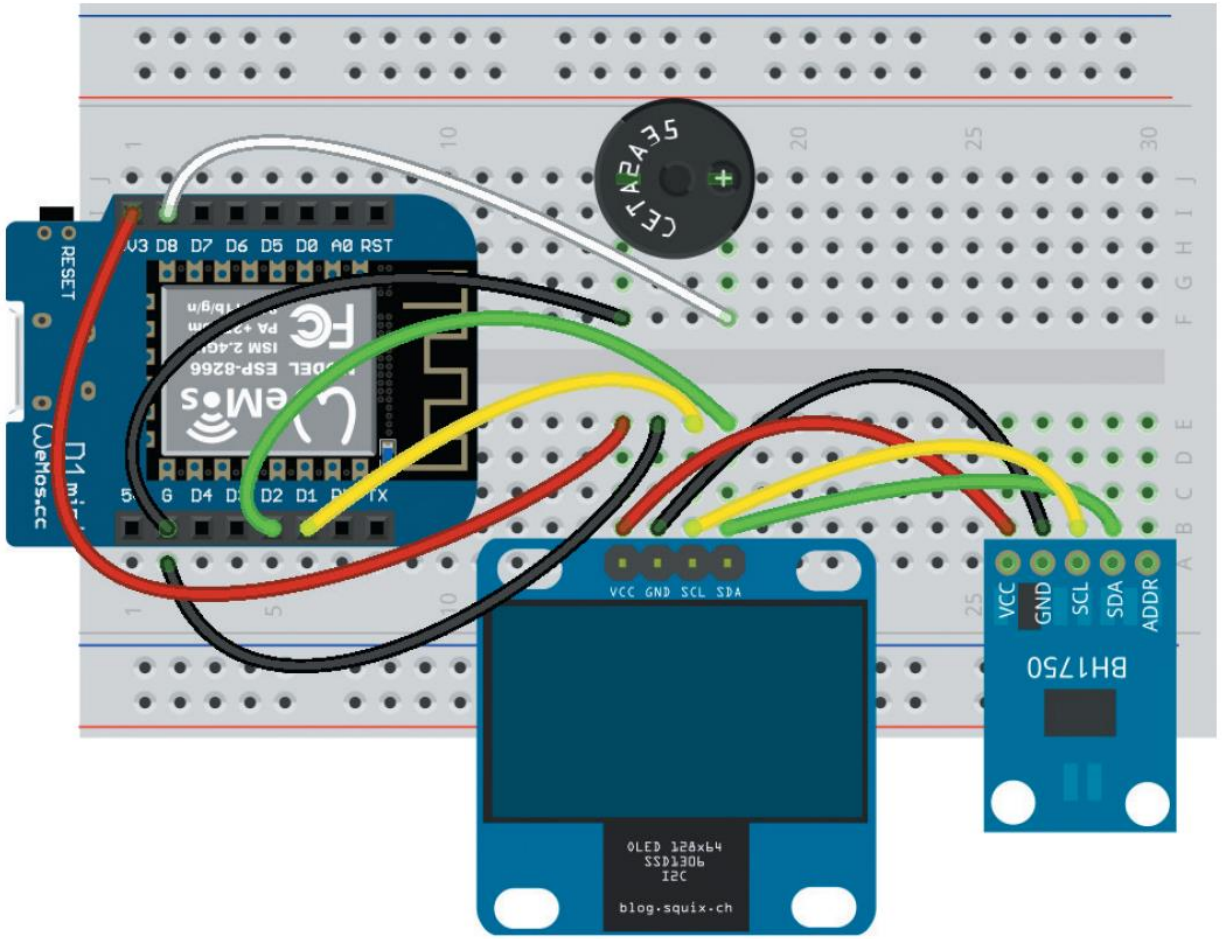
- **Lab11**

照明不足提醒裝置	
實驗目的	在環境亮度太暗時發出警告聲
材料	• D1 mini • OLED 模組 • 亮度感測模組 • 無源蜂鳴器

- **接線圖**

亮度感測模組接線方式請參閱 Lab 10，
無源蜂鳴器接線請參閱 Lab 07。

4-3 照明警示


RESET
D1 mini
WeMos.cc
D8 D7 D6 D5 D0 A0 RST
G D4 D2 D1
TX
VCC GND SCL SDA
0.96" OLED 128x64
SSD1306
I2C
blog.squix.ch
VCC
GND
SCL
SDA
ADDR
BH1750

4-3 照明警示

- **設計原理**

這裡製作亮度警示裝置，在光線不足時發出提醒。

考量到關燈休息的可能，
程式設定在照度低於20 lux 時也不會響警報。

- **程式設計**

```
from machine import Pin, PWM, I2C
from ssd1306 import SSD1306_I2C
import bh1750fvi, utime

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

buzzer = PWM(Pin(15, Pin.OUT), freq=110, duty=0)
```

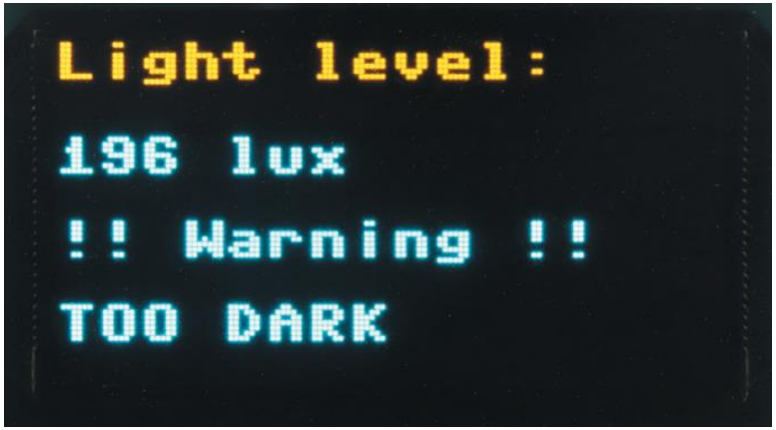
4-3 照明警示

```
while True:

    light_level = bh1750fvi.sample(i2c, mode=0x23)
    print("偵測亮度: " + str(light_level) + " lux")

    oled.fill(0)
    oled.text("Light level:", 0, 0)
    oled.text(str(light_level) + " lux", 0, 16)

    if 30 < light_level < 300:
        oled.text("!! Warning !!", 0, 32)
        oled.text("TOO DARK", 0, 48)
        print("警告 !! 亮度不足 !!")
        buzzer.freq=(110)
        buzzer.duty=(512)

    else:
        buzzer.duty(0)

    oled.show()
```

4-3 照明警示

- **實證**

執行程式後，當測得亮度低於 300 lux 、但仍高於 20 lux 時，蜂鳴器就會響起，OLED 也會顯示警告訊息：



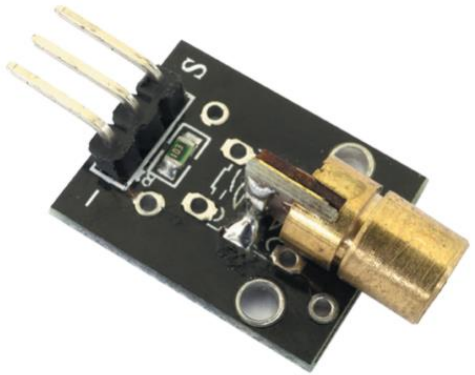
4-4 雷射模組

- **Lab12**

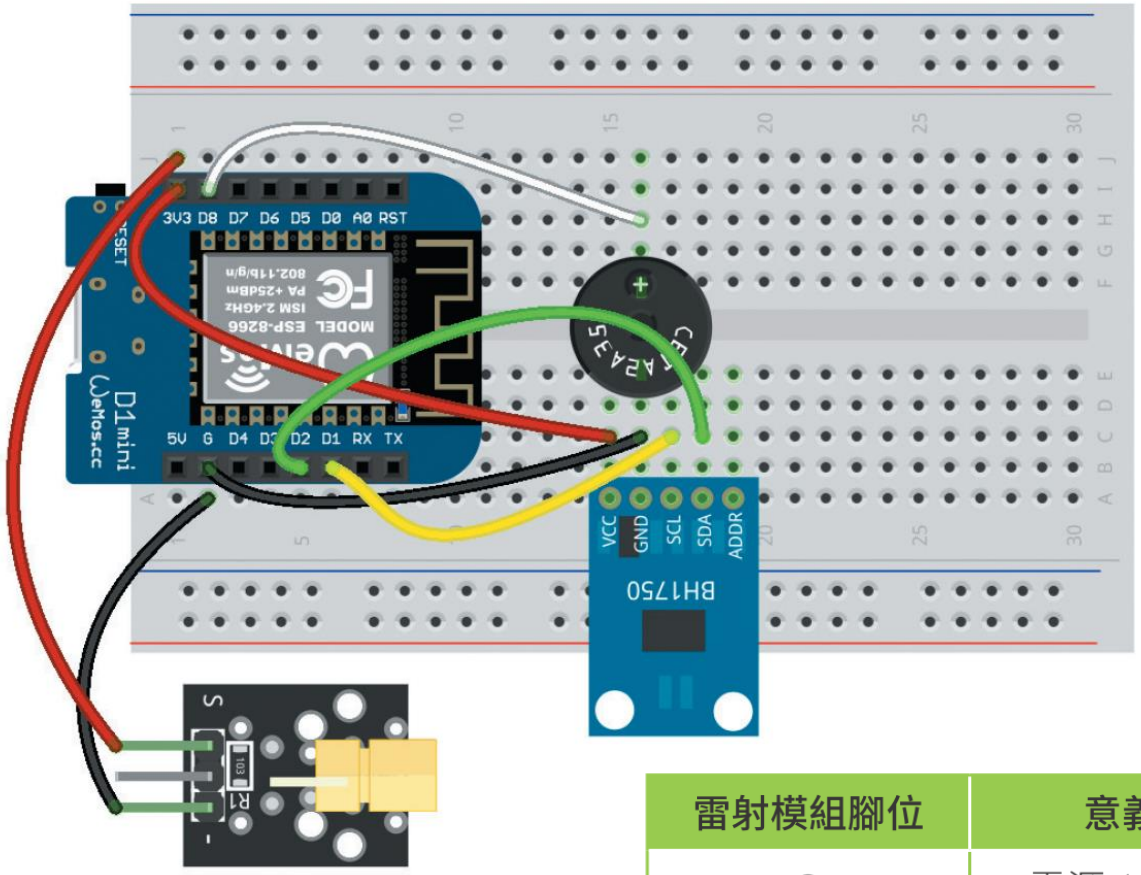
雷射打靶機	
實驗目的	做個能感應雷射光的雷射打靶機
材料	• D1 mini • 亮度感測模組 • 無源蜂鳴器 • 雷射模組

- **接線圖**

這裡將加入 KY-008 雷射模組，
通電後就會發出紅色雷射光：



4-4 雷射模組



雷射模組腳位	意義	D1 mini 對應腳位
S	電源 / 訊號	3V3
中央腳位	電源	不接線
-	接地	G

4-4 雷射模組

- **設計原理**

當亮度感測模組被雷射光照到時，分數加 1，蜂鳴器響起：

```
score += 1 # 分數遞增 1, 效果等同於 score = score + 1
```

由於雷射光強度強且不易散射，一照到模組的感測晶片 ，就會測得高達數萬 lux 的照度值。因此若在室內測試，亮度感測模組可分辨雷射光和其它光源。

設定打靶機打到 10 分後遊戲會結束，因此加入判斷式：

```
while score < 10:
# 迴圈內容
```

4-4 雷射模組

- 程式設計

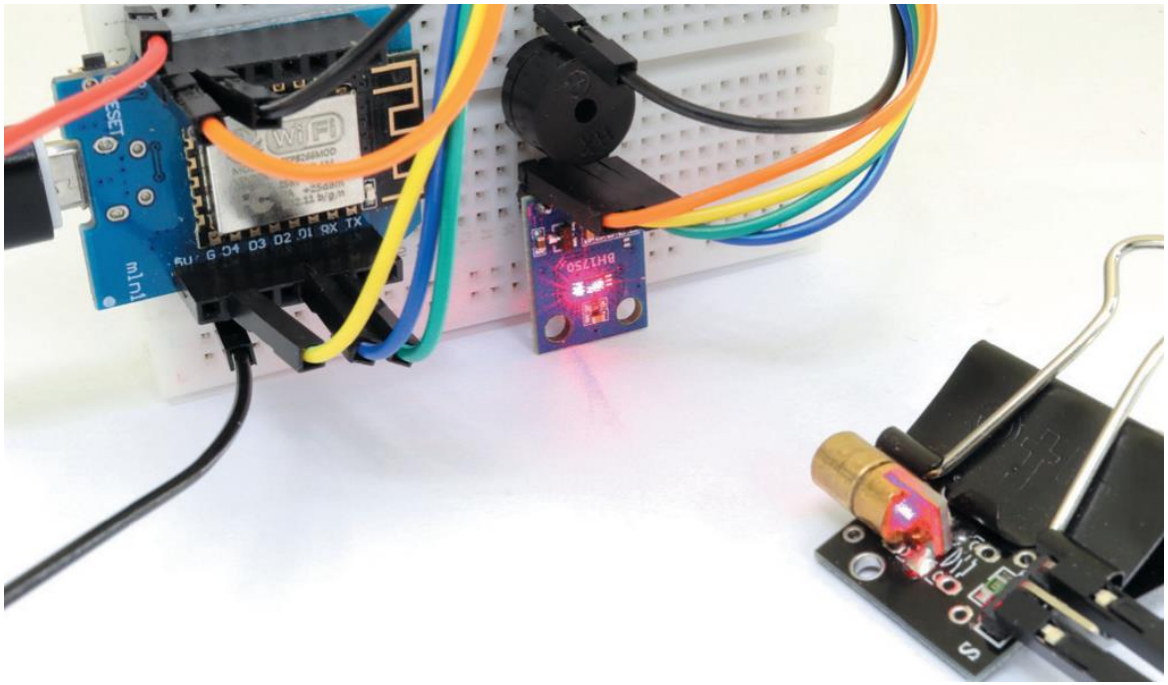
```
from machine import Pin, I2C, PWM
import bh1750fvi, utime

score = 0
buzzer = PWM(Pin(15, Pin.OUT), freq=784, duty=0)
while score < 10:

    light_level = bh1750fvi.sample(I2C(scl=Pin(5),
                                       sda=Pin(4)), mode=0x23)
    print("偵測亮度: " + str(light_level) + " lux, 得分: " +
          str(score))

    if light_level > 10000:

        print("==== 命中! ====")
        score += 1
```

4-4 雷射模組

```
            # 播放得分音效
            buzzer.freq(784)
            buzzer.duty(512)
            utime.sleep_ms(100)
            buzzer.freq(988)
            utime.sleep_ms(300)
            buzzer.duty(0)

        utime.sleep_ms(10)

print("=== 遊戲結束! ===")

# 播放遊戲結束音效
buzzer.freq(784)
buzzer.duty(512)
utime.sleep_ms(100)
buzzer.freq(659)
utime.sleep_ms(100)
buzzer.freq(523)
utime.sleep_ms(300)
buzzer.duty(0)
```

4-4 雷射模組

- 實測



4-5 光遮斷器

- **Lab13**

雷射保全系統	
實驗目的	修改 Lab 12 做成遮斷雷射光會觸發警報的保全系統
材料	• D1 mini • 亮度感測模組 • 無源蜂鳴器 • 雷射模組

- **接線圖**

同 Lab 12 。

4-5 光遮斷器

- **設計原理**

下面程式先進入第一個 while 迴圈，確定雷射光已對準亮度感測模組 ; 接著進入第二個 while 迴圈，檢查是否有物體遮斷雷射光，有的話就發出警報。

本例的 def getLightLevel() 可將會重複用到取得亮度的一段程式碼打包並命名為具有意義的 getLightLevel，就可以用 getLightLevel() 來方便取得亮度。

4-5 光遮斷器

getLightLevel 函式內最後一行 return data 可傳回此函式內讀取到的亮度感測模組數值。

下列程式碼：

```
Light_level = bh1750fvi.sample (I2C(scl=Pin(5),
                                 sda=Pin(4)), mode=0x23)
```

就可縮短成：

```
Light_level = getLightLevel()
```

4-5 光遮斷器

- 程式設計

```
from machine import Pin, I2C, PWM
import bh1750fvi, utime

def getLightLevel():
    data = bh1750fvi.sample(I2C(scl=Pin(5),
                                sda=Pin(4)), mode=0x23)
    return data

buzzer = PWM(Pin(15, Pin.OUT), freq=768, duty=0)

count = 0
print("系統校準中, 請讓雷射持續照射在亮度感測器5秒鐘...")

while count < 5: # 雷射校準迴圈

    light_level = getLightLevel()

    if light_level > 10000: # 偵測到雷射光
        count += 1
        print("已校準 " + str(count) + " 秒...")
    else:
        count = 0           # 歸零重新校準
```

4-5 光遮斷器

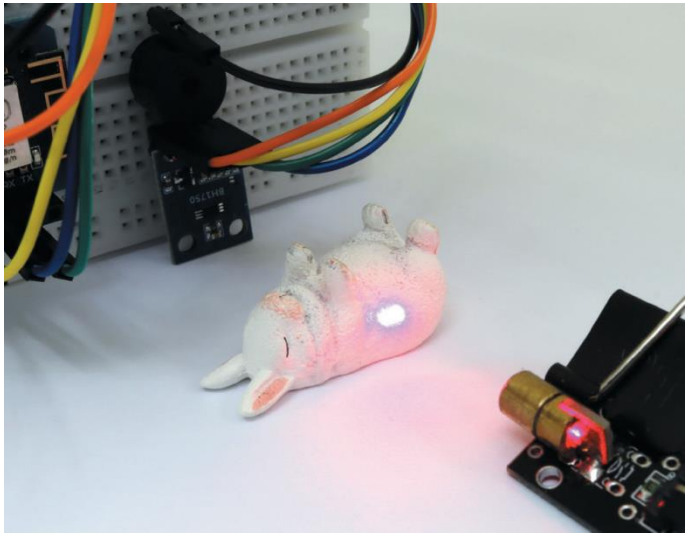
```
    utime.sleep_ms(1000)

while True: # 雷射偵測迴圈

    light_level = getLightLevel()

    if light_level < 10000: # 雷射光被遮斷
        print("!! 警報觸發 !!")
        buzzer.duty(512)

    else:
        buzzer.duty(0)
        print("-- 待命中 --")
```

4-5 光遮斷器

- **實測**

在執行程式前，固定好亮度感測模組和雷射模組，讓雷射光穩定打在亮度模組的感測晶片上。

執行程式，互動環境窗格顯示「 -- 待命中 -- 」，代表保全系統已啟動，這時擋住雷射光就能觸發警報：



```
系統校準中, 請讓雷射持續照射在亮度感測器5秒鐘...
已校準 1 秒...
已校準 2 秒...
已校準 3 秒...
已校準 4 秒...
已校準 5 秒...
-- 待命中 --
-- 待命中 --
-- 待命中 --
-- 待命中 --
!! 警報觸發 !!
```