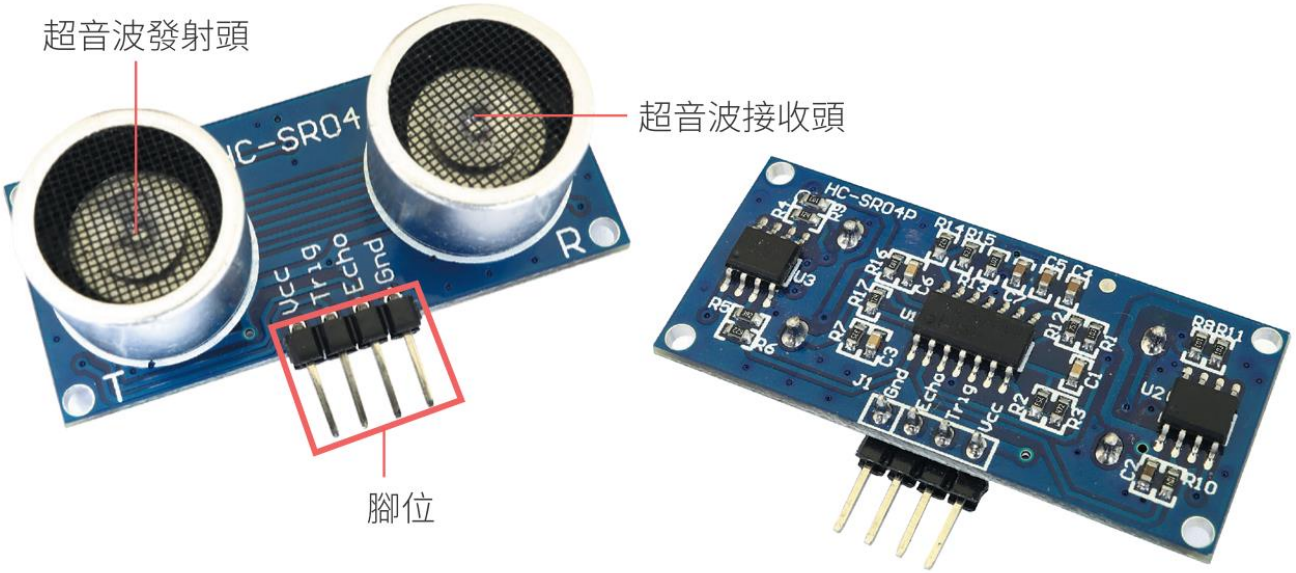
Ch03 判斷障礙物距離
– 超音波模組

3-1 認識超音波模組

- **超音波測距原理**

一邊發射頻率 40k 赫茲的超音波，另一邊接收撞到物體反彈的超音波；空氣中超音波傳播速度為每秒 340 公尺。

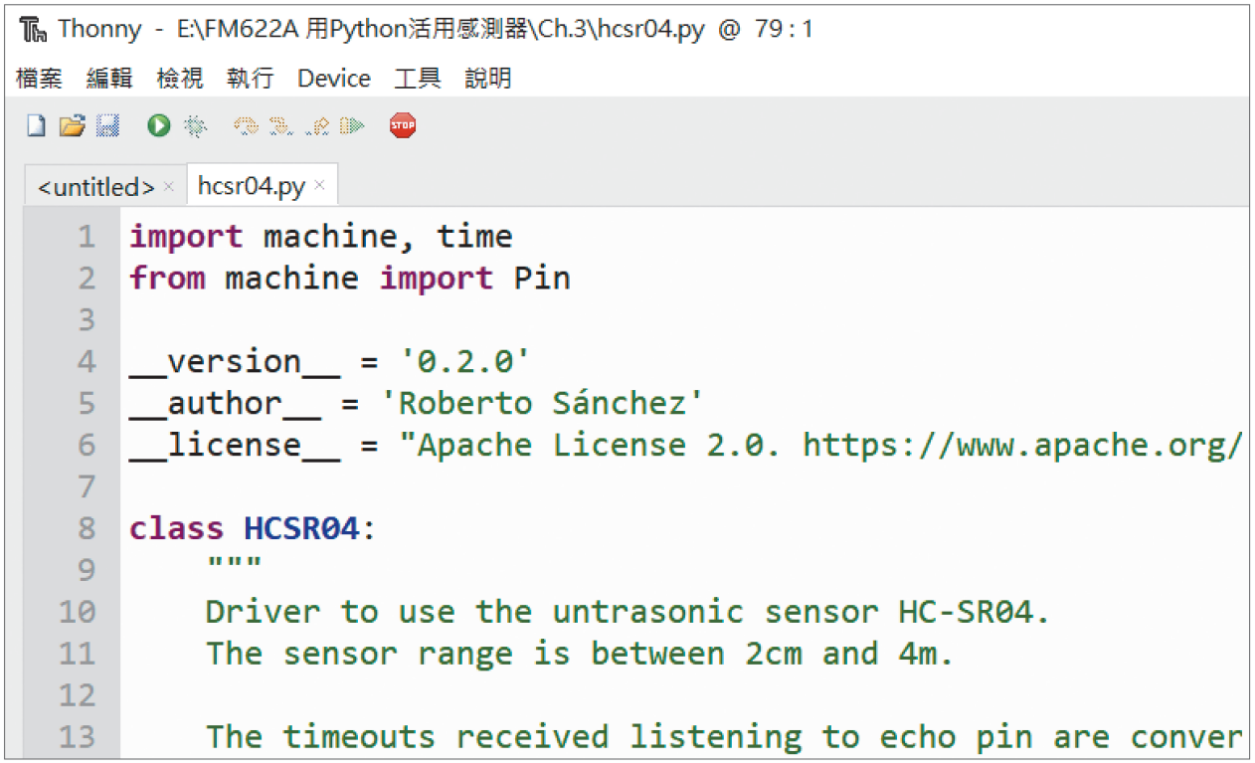
HC-SR04P 可偵測的範圍為 2 公分至 4 公尺。



3-1 認識超音波模組

- **安裝函式庫**

D1 mini 必須先安裝 HC-SR04P 的函式庫。



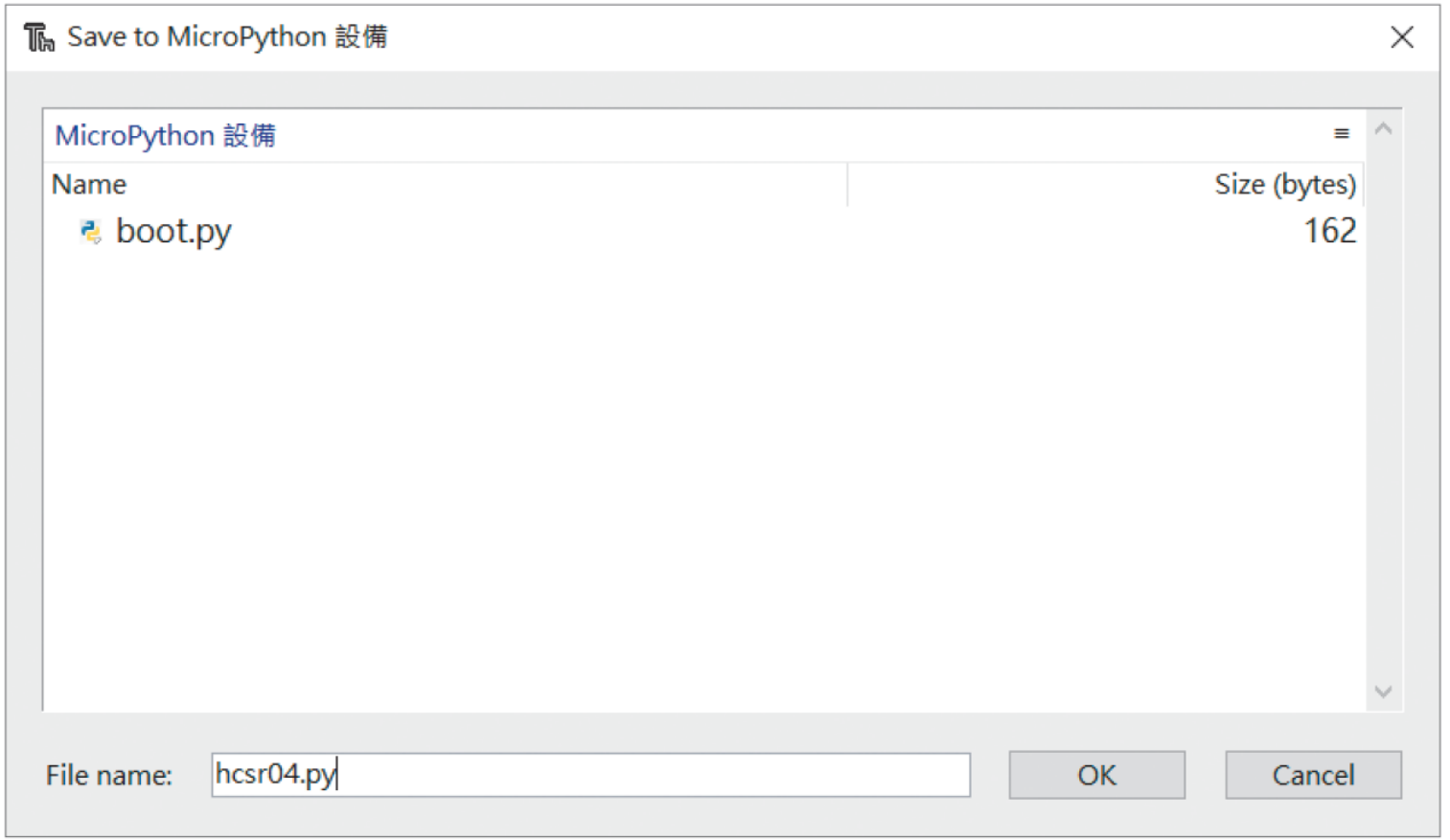
3-1 認識超音波模組

執行『檔案 / 儲存複本』：

選擇 MicroPython 設備：

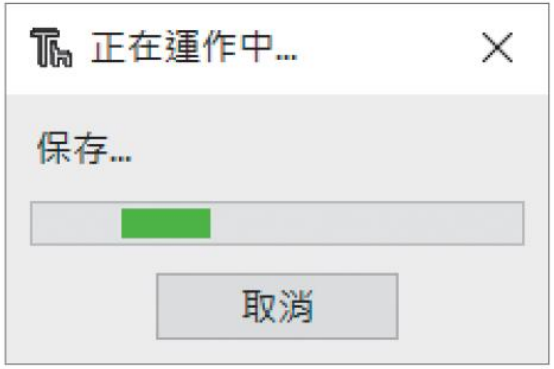
3-1 認識超音波模組

看到 D1 mini 上所有檔案。在交談窗輸入 hcsr04.py 並按 OK:



3-1 認識超音波模組

函式庫上傳到 D1 mini 上



在編輯器下方互動環境視窗確認是否上傳成功：

互動環境(Shell)

```
MicroPython v1.11-8-g48dcbbe60 on 2019-05-29; ESP module with E
SP8266
Type "help()" for more information.
>>> import os
>>> os.listdir()
['boot.py', 'hcsr04.py']
>>>
```

3-2 讀取障礙物距離

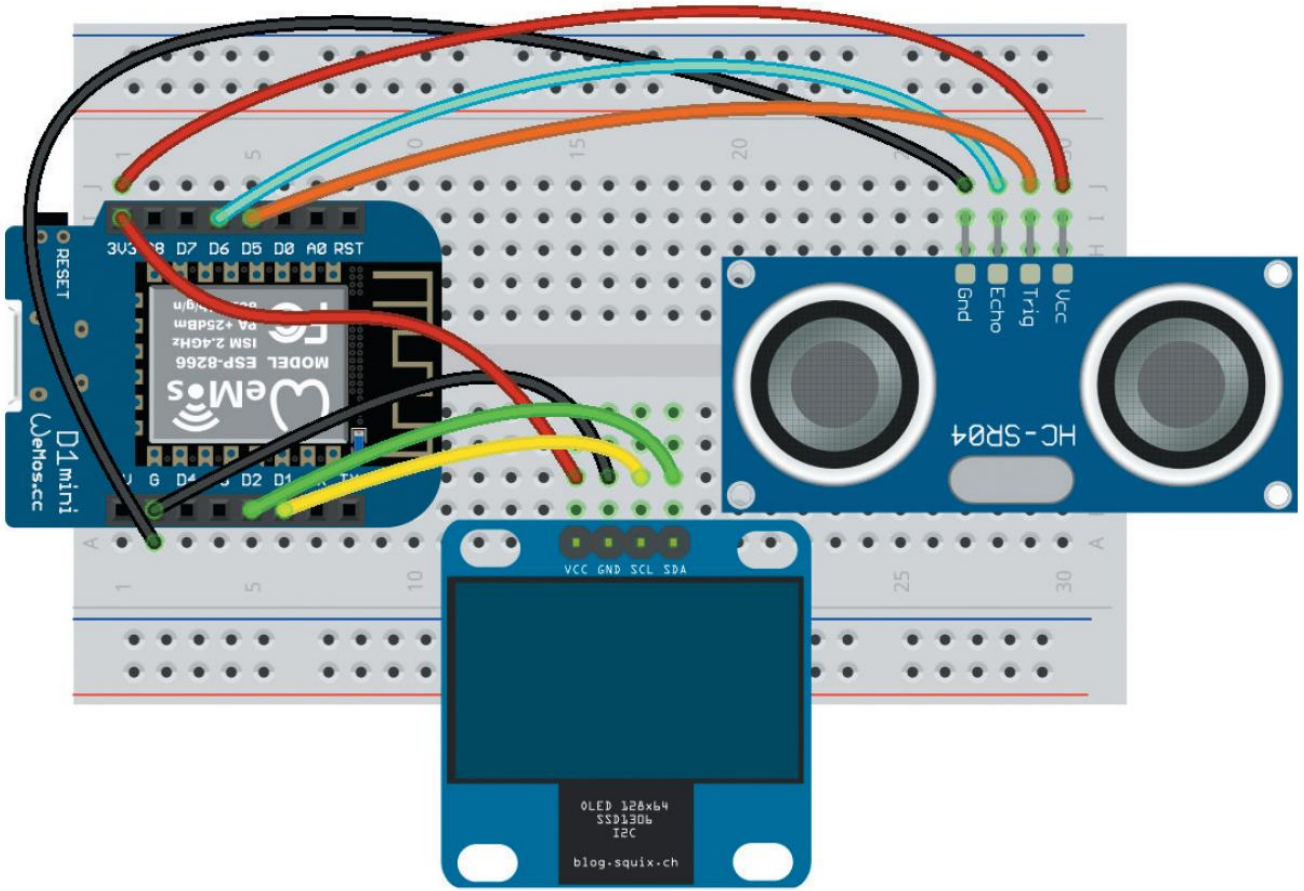
- **Lab06**

讀取並顯示障礙物距離	
實驗目的	讀取超音波測距值並顯示在 OLED 上
材料	• D1 mini • OLED 模組 • 超音波模組

- 接線圖

超音波模組腳位	意義	D1 mini 對應腳位
Vcc	電源	3V3
Trig	觸發腳	D5 (14 號腳位)
Echo	接收腳	D6 (12 號腳位)
Gnd	接地	G

3-2 讀取障礙物距離


RESET
D1 mini
(WeMos.cc
3V3 D7 D6 D5 D0 A0 RST
G D4 D2 D1
MODEL ESP-8266
ISM 2.4GHz
PA +25dBm
Gnd
Echo
Trig
Vcc
HC-SR04
VCC GND SCL SDA
OLED 128x64
SSD1306
I2C
blog.squix.ch

3-2 讀取障礙物距離

- **設計原理**

匯入函式庫：

```
from hcsr04 import HCSR04
```

建立超音波模組物件：

```
sonar = HCSR04(trigger_pin=14, echo_pin=12)
```

trigger_pin 是觸發腳，用來讓模組發出超音波。
echo_pin 是接收腳，讀取模組收到的超音波。

3-2 讀取障礙物距離

呼叫 sonar.distance_cm() 方法，讀取偵測到的障礙物距離：

```
distance = sonar.distance_cm()
print(distance)
```

變數 distance 會儲存超音波模組的測距數值，單位為公分。

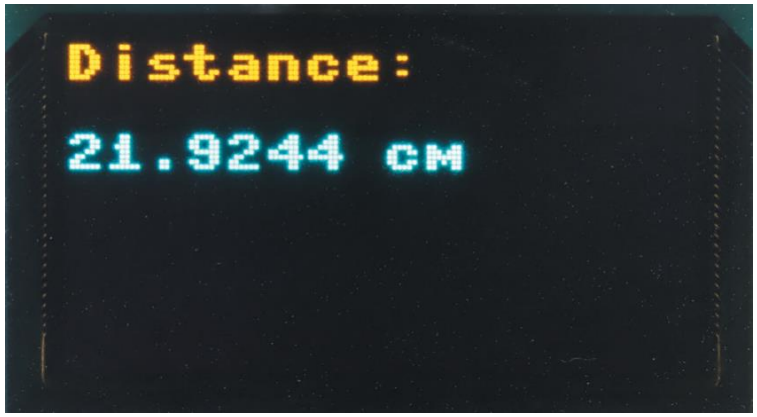
3-2 讀取障礙物距離

- 程式設計

```
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
from hcsr04 import HCSR04
import utime

oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))
sonar = HCSR04(trigger_pin=14, echo_pin=12)

while True:

    distance = sonar.distance_cm()
oled.fill(0)
oled.text("Distance:", 0, 0)
oled.text(str(distance) + " cm", 0, 16) # 和 OLED 上一行字隔一行
oled.show()
print("偵測距離: " + str(distance) + " 公分")

utime.sleep_ms(25)
```

3-2 讀取障礙物距離

- **實測**



編輯器的互動環境視窗會顯示類似以下結果：

```
偵測距離: 26.0653 公分
偵測距離: 26.4433 公分
偵測距離: 25.945 公分
偵測距離: 25.9622 公分
偵測距離: 25.9107 公分
```

3-3 防盜警報器 – 使用無源蜂鳴器

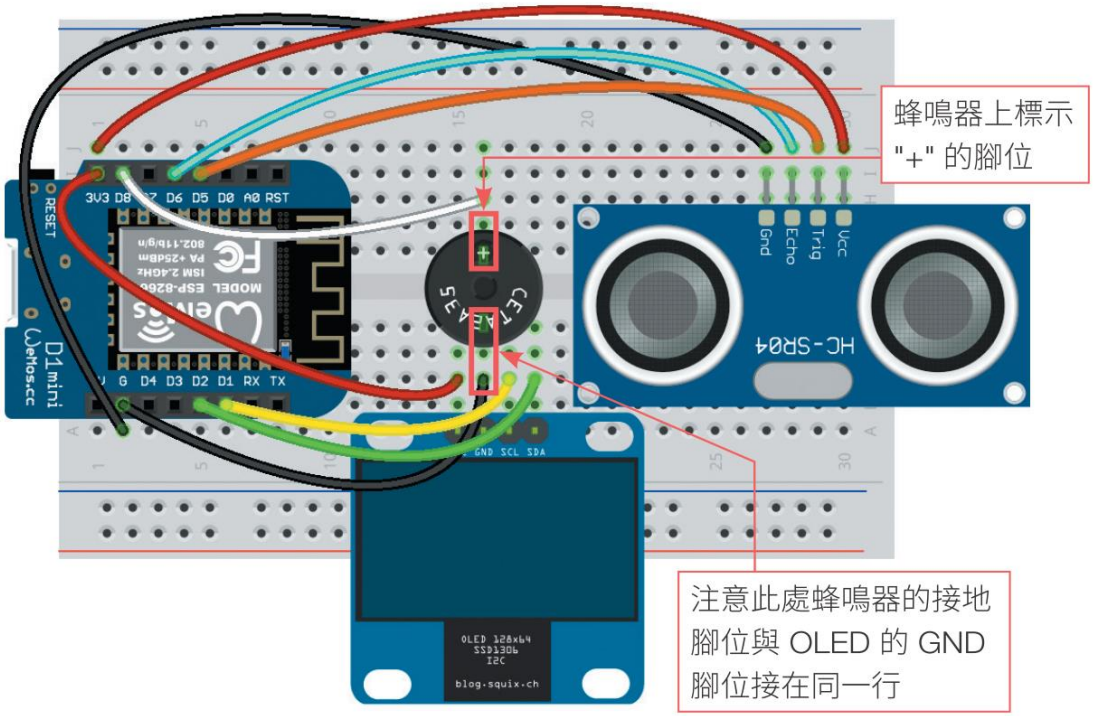
- **Lab07**

防盜警報器	
實驗目的	在超音波模組偵測到一定距離內有物體時，響起警報聲並顯示警報訊息
材料	• D1 mini　• OLED 模組 • 超音波模組　• 無源蜂鳴器

無源蜂鳴器使用壓電效應，當中的壓電陶瓷片會在以特定頻率通電後依該頻率震動、進而發出聲音

3-3 防盜警報器 – 使用無源蜂鳴器

- 接線圖

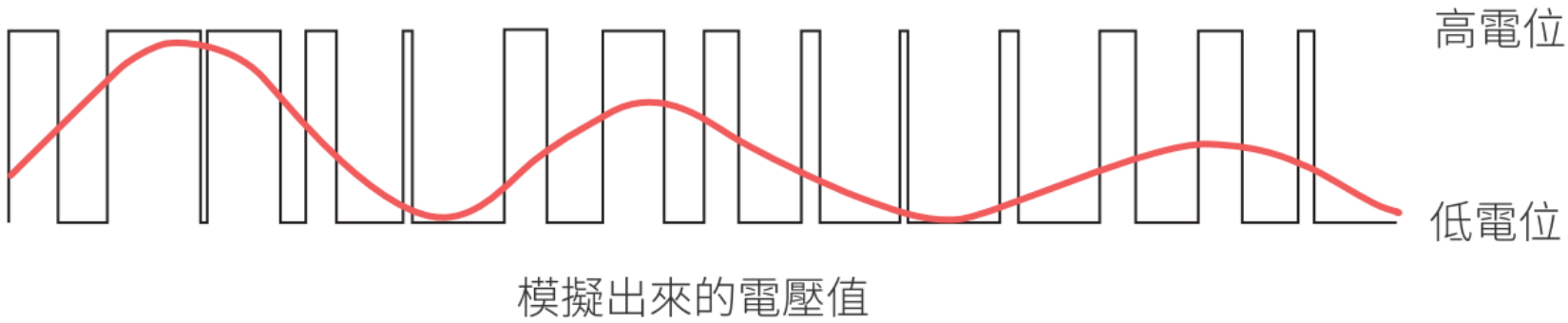


蜂鳴器腳位	意義	D1 mini 對應腳位
發音孔旁有 + 號那側腳位	電源	D8 (15 號腳位)
另一側腳位	接地	G

3-3 防盜警報器 – 使用無源蜂鳴器

- **設計原理**

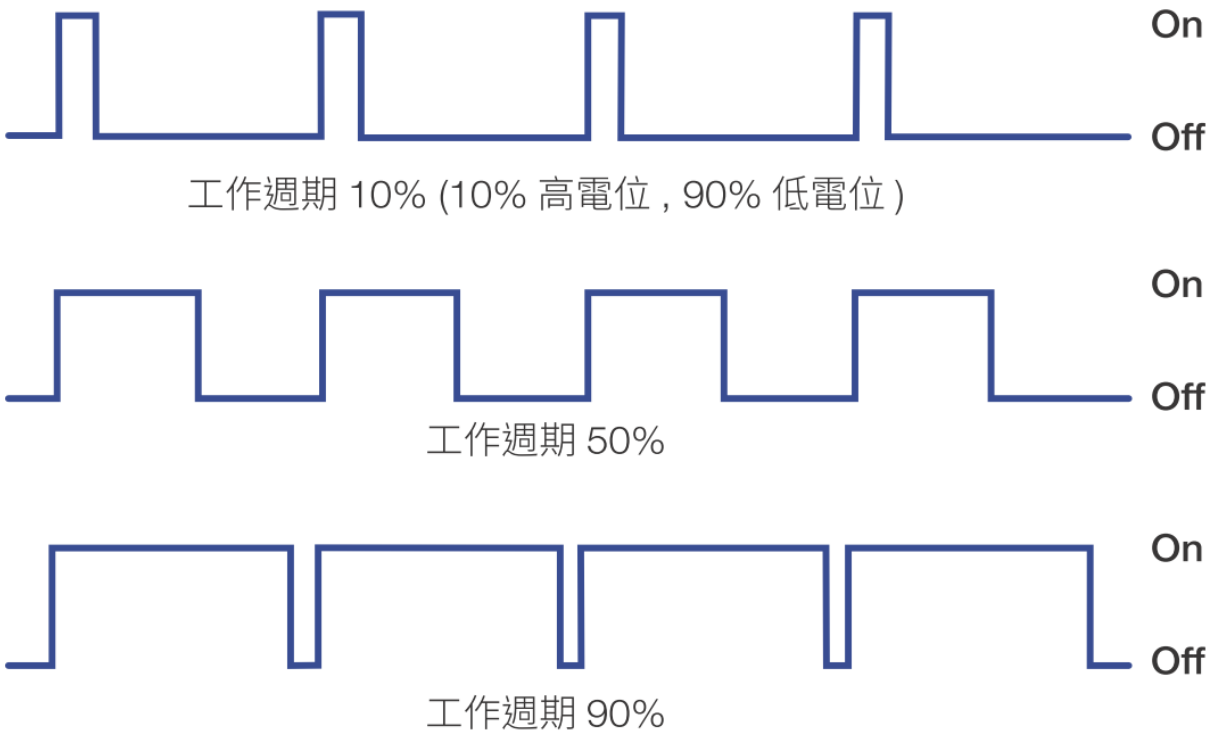
用 PWM (Pulse Width Modulation, 脈衝寬度調變) 調整蜂鳴器頻率，讓無源蜂鳴器發出聲音。



3-3 防盜警報器 – 使用無源蜂鳴器

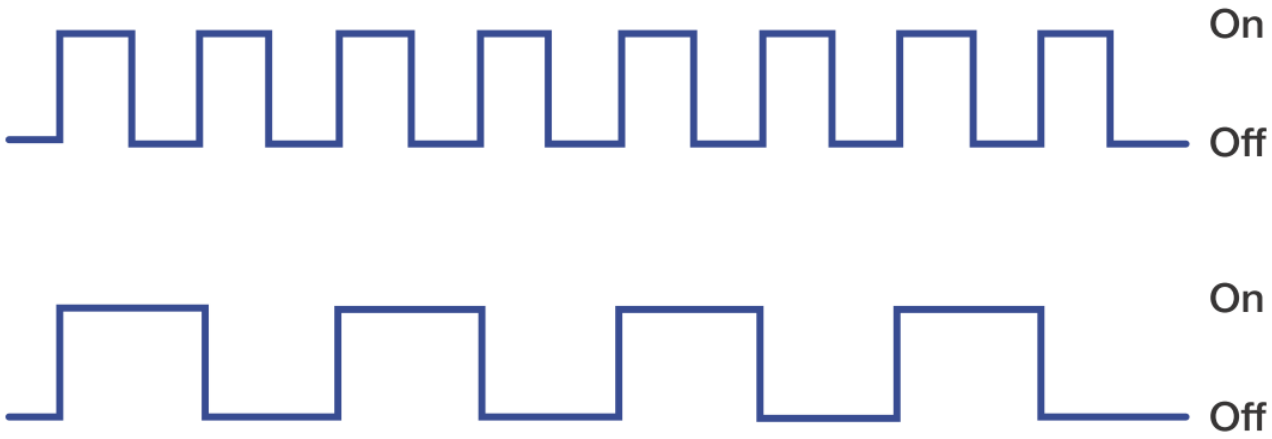
PWM 有兩種參數：

1. 工作週期的值越高，產生的平均電壓就越高。



3-3 防盜警報器 – 使用無源蜂鳴器

2. 頻率，也就是每秒電壓變高與變低的次數：



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

當無源蜂鳴器的壓電片以特定頻率振動時，就會發出該頻率的聲音。

3-3 防盜警報器 – 使用無源蜂鳴器

由於壓電片要有「通電 - 不通電」的循環才能產生振盪，控制蜂鳴器時的輸出電壓是固定的，重點在於調整 PWM 的頻率。

為在 MicroPython 使用 PWM 功能，必須匯入相關函式庫：

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

3-3 防盜警報器
– 使用無源蜂鳴器

建立 PWM 物件，指定給 15 號腳位 (D8)，
設頻率為 0 、工作週期為 512:

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

參數 freq是 PWM 的輸出頻率，設為 0 。
duty 是 PWM 工作週期；最大值為 1023，50% 相當於 512 。
上面程式可合併 :

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

3-3 防盜警報器 – 使用無源蜂鳴器

音高頻率範例：

音符	中音 C	D	E	F	G	A	B
唱名	Do	Re	Mi	Fa	Sol	La	Ti
頻率 (Hz)	261	294	330	349	392	440	494

```
from machine import Pin, PWM
import utime
buzzer = PWM(Pin(15, Pin.OUT), freq=0, duty=512)

buzzer.freq(261) # 頻率設為 261Hz->中音 C (Do)
utime.sleep_ms(1000) #讓聲音持續 1000 毫秒
buzzer.deinit() # 結束, 關閉 PWM (完全關閉蜂鳴器)
```

注意 PWM 頻率必須設為整數。

3-3 防盜警報器 – 使用無源蜂鳴器

- 程式設計

```
from machine import Pin, I2C, PWM
from ssd1306 import SSD1306_I2C
from hcsr04 import HCSR04
import utime

alarm_distance = 10 # 觸發警報距離
```

```
oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))
sonar = HCSR04(trigger_pin=14, echo_pin=12)

buzzer = PWM(Pin(15, Pin.OUT), freq=880, duty=0)

while True:

    distance = sonar.distance_cm()

    oled.fill(0)
    oled.text("Distance:", 0, 0)
    oled.text(str(distance) + " cm", 0, 16)

    if 2 <= distance <= alarm_distance:
        buzzer.duty(512)
        oled.text("!!! ALARM !!!", 0, 40) # 也在 OLED 顯示警報訊息
        print("!!! 觸發警報 !!!")
    else:
        buzzer.duty(0)
        print("無警報")

    oled.show()

    utime.sleep_ms(25)
```

3-3 防盜警報器 – 使用無源蜂鳴器

- **實測**


```
無警報
無警報
!!! 觸發警報 !!!
!!! 觸發警報 !!!
!!! 觸發警報 !!!
```

3-4 倒車雷達

- **Lab08**

倒車雷達	
實驗目的	讀取超音波模組的測距值，除了顯示在 OLED 上，也拿來控制喇叭發出聲音的間隔長短
材料	• D1 mini • 超音波模組 • OLED 模組 • 無源蜂鳴器

- **接線圖**

同 Lab 07

3-4 倒車雷達

- **設計原理**

改變while 迴圈內的延遲時間，調整蜂鳴器發出聲響的間隔，以反映偵測距離遠近。

以下程式設定超音波的測距範圍為 2 到 50 公分，
距離值乘 10 就是 while 迴圈停頓的時間值。

偵測到物體時讓蜂鳴器發出短促聲響，響 10 毫秒後關閉

3-4 倒車雷達

```
while True:

distance = sonar.distance_cm()

    if 2 <= distance <= 50:
        buzzer.duty(512)
        utime.sleep_ms(10)
        buzzer.duty(0)
        sound_delay = int(distance) * 10 - 10

utime.sleep_ms(sound_delay)
```

3-4 倒車雷達

- 程式設計

```
from machine import Pin, I2C, PWM
from ssd1306 import SSD1306_I2C
from hcsr04 import HCSR04
import utime

oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))
sonar = HCSR04(trigger_pin=14, echo_pin=12)

buzzer = PWM(Pin(15, Pin.OUT), freq=784, duty=0)
sound_delay = 500
buzzer_time = utime.ticks_ms()

while True:

    distance = sonar.distance_cm()
    oled.fill(0)
    oled.text("Distance:", 0, 0)
    oled.text(str(distance) + " cm", 0, 16)
    oled.show()
    print("偵測距離: " + str(distance) + " 公分")
```

3-4 倒車雷達

```
if 2 <= distance <= 50:
    buzzer.duty(512)
    utime.sleep_ms(10)
    buzzer.duty(0)
    sound_delay = int(distance) * 10 - 10
else:
    sound_delay = 500

utime.sleep_ms(sound_delay)
```

- **實測**

執行程式，把手慢慢靠近超音波模組，
即可聽到蜂鳴器以越來越急促的間隔發出聲音。

3-5 無弦空氣琴

- **Lab09**

無弦空氣琴	
實驗目的	用超音波模組動態調整喇叭音高
材料	• D1 mini • 超音波模組 • OLED 模組 • 無源蜂鳴器

- **接線圖**

同 Lab 07 。

3-5 無弦空氣琴

- **設計原理**

讓程式把超音波模組測到的距離換算成頻率:

頻率下限 220 Hz (低音 A)	頻率上限 880 Hz (高音 A)	差距 880 – 220 = 660
測距下限 40 公厘	測距上限 700 公厘	700 – 40 = 660

用另一函式從超音波模組讀取以公厘為單位的距離值 :

```
distance = sonar.distance_mm()
```

3-5 無弦空氣琴

由於選定的測距值和頻率值的範圍剛好相同，
只要把測得距離減去最低測距距離 40 公厘，加上最低頻率值 220 就是轉換後的頻率。

- 程式設計

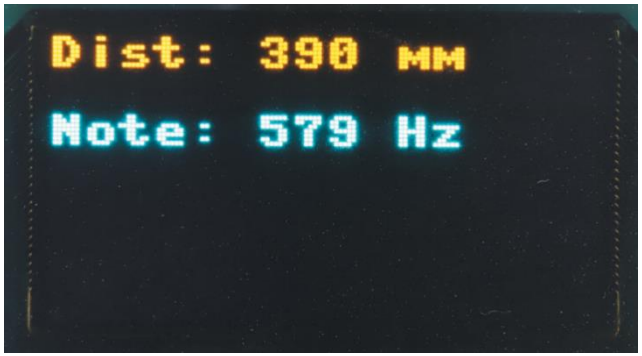
```
from machine import Pin, I2C, PWM
from ssd1306 import SSD1306_I2C
from hcsr04 import HCSR04
import utime

note_min = 220 # 最小頻率(Hz)
note_max = 880 # 最大頻率(Hz)
dist_min = 40  # 測距最小距離(公厘)
dist_max = dist_min + (note_max - note_min)
                # 算出測距最大距離(公厘)
```

```
oled = SSD1306_I2C(128, 64, I2C(scl=Pin(5), sda=Pin(4)))
sonar = HCSR04(trigger_pin=14, echo_pin=12)
buzzer = PWM(Pin(15, Pin.OUT), freq=0, duty=0)
buzzer.deinit()

while True:

    distance = sonar.distance_mm()

    if dist_min <= distance <= dist_max:
        key = note_min + (distance - dist_min)
        buzzer.freq(key)
        buzzer.duty(512)
    else:
        key = 0
        buzzer.duty(0)

    oled.fill(0)
    oled.text("Dist: " + str(distance) + " mm", 0, 0)
    oled.text("Note: " + str(key) + " Hz", 0, 16)
    oled.show()
    print("偵測距離: " + str(distance) + " 公厘, 播放頻率: " +
          str(key) + " 赫茲")

    utime.sleep_ms(10)
```

3-5 無弦空氣琴

- **實測**

執行程式，試著用手在超音波模組前來回移動




```
偵測距離: 115 公厘, 播放頻率: 325 赫茲
偵測距離: 112 公厘, 播放頻率: 322 赫茲
偵測距離: 128 公厘, 播放頻率: 338 赫茲
偵測距離: 138 公厘, 播放頻率: 348 赫茲
偵測距離: 136 公厘, 播放頻率: 346 赫茲
```