# Topic#3 Lab1 實驗結報

姓名：蔡佩蓉　　學號：109511286

## 一、 實驗目的

Q1: 使用 Raspberry Pi 控制 LED 燈依照摩斯密碼表發送 SOS 信號。
Q2: 使用 Raspberry Pi 和 DHT11 溫濕度感應器設計一個溫度警示燈。將使用者輸入的溫度設為門檻值，每秒顯示當前溫度及濕度，當溫度超過門檻值時將開啟 LED 燈。
Q3: 使用 Raspberry Pi 和 HC-SR04 超音波感測器設計一個距離感測警示燈。根據距離感測器的數據，控制 LED 燈的狀態。
Q4: 結合 Q2 和 Q3，使用 Raspberry Pi 、 DHT11 溫濕度感應器和 HC-SR04 超音波感測器設計一個距離（和溫度）感測警示燈 2.0。根據溫度和距離感測器的數據，控制 LED 燈的狀態，並顯示相關信息。

## 二、 實驗過程 (Code + 說明)

Q1: 用 LED 產生 SOS 的摩斯密碼(重複顯示)

```
import RPi.GPIO as GPIO
import time

# 設定 LED 腳位
GPIO.setmode(GPIO.BOARD)
LED_PIN = 12
GPIO.setup(LED_PIN, GPIO.OUT)

while True:
    # S
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.3)

    # O
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.3)
```

```
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.3)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.3)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.3)

    # S
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(LED_PIN, GPIO.LOW)
    time.sleep(0.7)

GPIO.cleanup()
```

Q2: 設計一個溫度警示燈

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

GPIO.setwarnings(False)

# 設定 LED 腳位
GPIO.setmode(GPIO.BOARD)
LED_PIN = 12
GPIO.setup(LED_PIN, GPIO.OUT)

# Setup DHT11
sensor_args = {'11' : Adafruit_DHT.DHT11,
               '22': Adafruit_DHT.DHT22,
               '2302': Adafruit_DHT.AM2302}
sensor = sensor_args['11']

# GPIO#, ex: GPIO4 = Pin7
gpio = 4
```

```
# 使用者輸入溫度
user_input = float(input("Input: "))

while True:
    # DHT 感應器讀取溫度與濕度
    humidity, temperature = Adafruit_DHT.read_retry(sensor,
gpio)
    if humidity is not None and temperature is not None:
        # 每秒輸出當前溫度與濕度
        print('Temp={0:0.1f}*
Humidity={1:0.1f}%'.format(temperature, humidity))
        # 當溫度大於使用者輸入溫度時，開啟 LED 燈
        if temperature > user_input:
            GPIO.output(LED_PIN, GPIO.HIGH)
        # 反之，關閉 LED 燈
        else:
            GPIO.output(LED_PIN, GPIO.LOW)
        time.sleep(1)
    else:
        print('Failed to get reading. Try again!')
        sys.exit(1)
GPIO.cleanup()
```

Q3: 距離感測警示燈

```
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)

v = 343
TRIG = 16
E = 18
LED_PIN = 12

print '1'
# 設定 LED 腳位，TRIG 發出超音波，Echo(E)接收反射回來的訊號
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(E, GPIO.IN)
GPIO.output(TRIG, GPIO.LOW)

def measure():
    # 開始測量距離時，發出 1 秒(delay)的超音波
    GPIO.output(TRIG, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(TRIG, GPIO.LOW)
    pulse_start = 0
```

```
    pulse_end = 0
    while GPIO.input(E) == GPIO.LOW:
        pulse_start = time.time()
    while GPIO.input(E) == GPIO.HIGH:
        pulse_end = time.time()
    # 用時間計算距離
    t = pulse_end - pulse_start
    d = t * v
    d = d / 2
    return d * 100

while(1):
    distance = measure()
    print(distance)
    # 控制 LED 燈
    # (距離<10，開啟 LED；10<=距離<=20，LED 閃爍；距離>20，關閉 LED)
    if distance < 10:
        GPIO.output(LED_PIN, GPIO.HIGH)
    elif distance >= 10 and distance <= 20:
        GPIO.output(LED_PIN, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(LED_PIN, GPIO.LOW)
        time.sleep(0.1)
    elif distance > 20:
        GPIO.output(LED_PIN, GPIO.LOW)

GPIO.cleanup()
```

Q4: 距離感測警示燈 2.0

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

GPIO.setwarnings(False)

TRIG = 16
E = 18
LED_PIN = 12

print '1'
# 設定 LED 腳位，TRIG 發出超音波，Echo(E)接收反射回來的訊號
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(E, GPIO.IN)
GPIO.output(TRIG, GPIO.LOW)

# Setup DHT11
```

```
sensor_args = {'11' : Adafruit_DHT.DHT11,
               '22': Adafruit_DHT.DHT22,
               '2302': Adafruit_DHT.AM2302}
sensor = sensor_args['11']

# GPIO#, ex: GPIO4 = Pin7
gpio = 4


def temperature():
    # DHT 感應器讀取溫度與濕度
    humidity, temperature = Adafruit_DHT.read_retry(sensor,
gpio)
    if humidity is not None and temperature is not None:
        return temperature
    else:
        print('Failed to get reading. Try again!')
        return

def measure():
    # 開始測量距離時，發出 1 秒(delay)的超音波
    GPIO.output(TRIG, GPIO.HIGH)
    time.sleep(1)
    GPIO.output(TRIG, GPIO.LOW)
    pulse_start = 0
    pulse_end = 0
    while GPIO.input(E) == GPIO.LOW:
        pulse_start = time.time()
    while GPIO.input(E) == GPIO.HIGH:
        pulse_end = time.time()
    # 用時間計算距離 (用公式 v = 331 + 0.6 * t)
    t = pulse_end - pulse_start
    temp = temperature()
    v = 331 + 0.6 * temp
    d = t * v
    d = d / 2
    # Print output
    print("============================")
    print('Temp: {0:0.1f}*C'.format(temp))
    print('V = 331 + 0.6 * {0:0.1f}'.format(temp))
    print('  = {0:0.1f}'.format(v))
    return d * 100

while(1):
    distance = measure()
    print('Distance: {0:0.1f}cm'.format(distance))
    # 控制 LED 燈
    # (距離<10，開啟 LED；10<=距離<=20，LED 閃爍；距離>20，關閉 LED)
    if distance < 10:
        GPIO.output(LED_PIN, GPIO.HIGH)
```

```
    elif distance >= 10 and distance <= 20:
        GPIO.output(LED_PIN, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.output(LED_PIN, GPIO.LOW)
        time.sleep(0.1)
    elif distance > 20:
        GPIO.output(LED_PIN, GPIO.LOW)

    time.sleep(1)
GPIO.cleanup()
```

## 三、 問題及解法

問題 1: DHT 感應器讀值不穩定

解法：

- 穩定電源：提供穩定的電源給感測器，避免電壓不穩定或供電不足的情況。(要記得檢查供電為 3.3V 還是 5V)
- 延遲與重試：在讀取感測器數值後，可以添加一些延遲並進行多次重試，避免因為讀取速度過快而導致的讀值不穩定情況。
  根據 ppt 裡 Q2 的 AdafruitDHT.py 寫 Demo code，會發生一個情況，就是 code 會在 if – else statement 那裡進入 else 裡然後 sys.exit(1)。為了解決這個狀況，我直接跳過那個 if – else statement，然後用 sensor = sensor_args['11'] 和 gpio = 4 定義腳位，除了解決 sys.exit 外也避免冗長的 code。

問題 2: 超音波感測距離差異大

解法：

- 環境干擾：降低環境干擾。例如，當有其他聲源（例如其他超聲波感測器）或障礙物時，可能會導致回波干擾。
- 超聲波波束：超聲波感測器通常具有一定的波束寬度，嘗試使其與測量物體的表面垂直對齊，以減少反射問題。
- 增加觸發訊號 pulse width: 增加延遲時間以確保發送的 trigger signal 可以被接收到。
  將原本 ppt 裡超音波感測的 TRIG 發出超音波的時間 (delay time)，0.0001 秒 (time.sleep(0.00001))，設成 time.sleep(1)也就是 1 秒的 delay，成功避免發生卡頓問題。

## 四、 心得

這次實驗讓我更深入了解了感測器和 Raspberry Pi 的連接和使用。

實驗中遇到的狀況是樹梅派有時候變太燙時會當掉，還有迴圈中 time.sleep()設太小時會斷斷續續的，所以不要設太小。還有 HC-SR04 在距離太過遙遠時會出現很大的誤差，不過短距離內都是正常的。

這次實驗並不難，但更多的是對於設備和 MobaXterm 的不熟悉，所以也花了不少時間來完成。首先遇到的狀況是筆電透過 USB 轉 TTL 序列傳輸線沒辦法連接到樹梅派，最後是跟助教換了一條全新的線，然後再上網重新安裝了符合版本的舊版驅動程式，最後才成功連接到。再來就是 MobaXterm 的複製貼上和很多快捷鍵都和平常使用的不太相同，常常不小心按錯鍵。