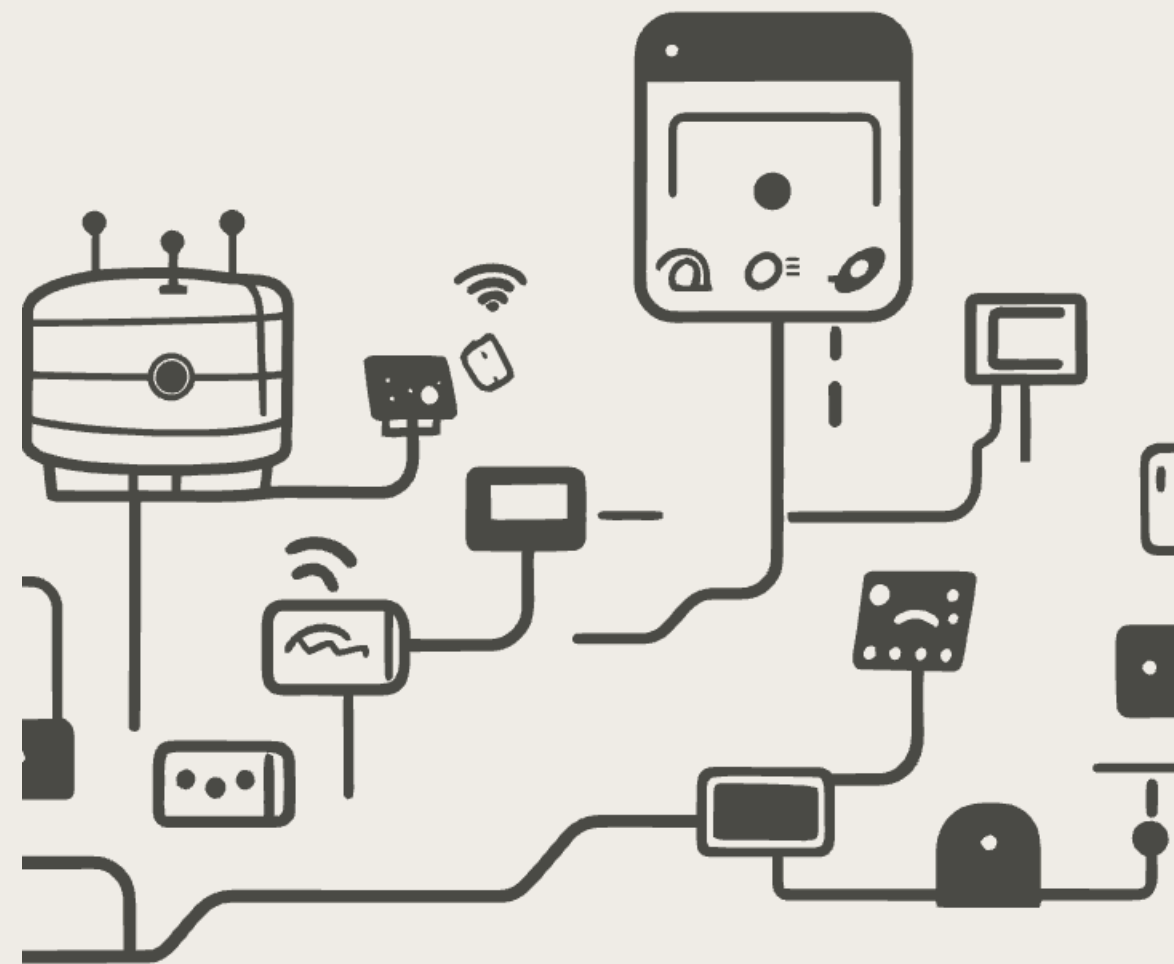


# 通訊與 AI 模型整合應用期末 實作報告

本次期末專題實作主題為：「感測器數據分析與 LINE Bot 輔助通報系統」。  
我們使用 Arduino Uno 搭配 DHT11 溫溼度感測器、HC-SR501 人體紅外線感測器與 HC-05 藍牙模組，實現一套具備環境偵測、即時 LINE 通知與自然語言互動功能的智慧通訊系統。

B11123206 陳冠欣



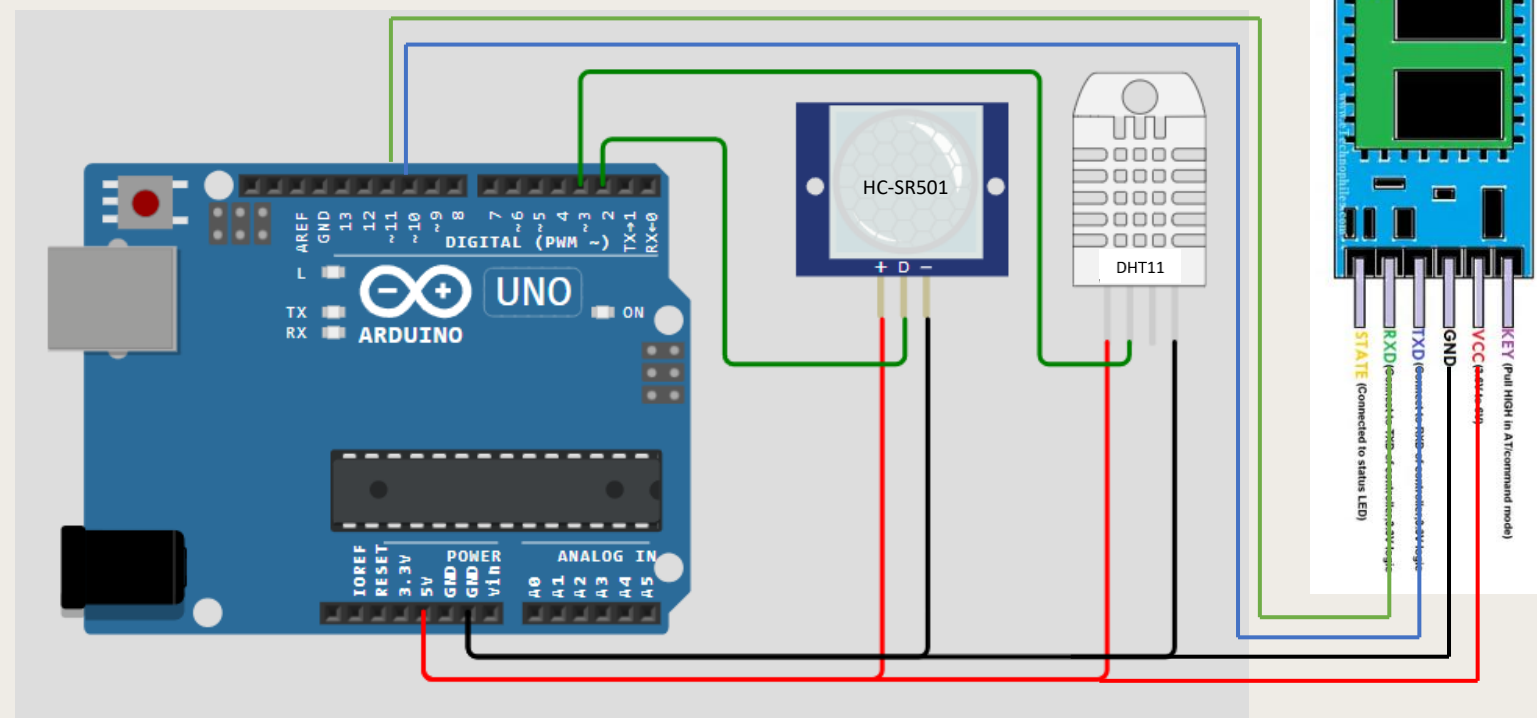
# 實作流程：Arduino 感測器讀值與藍牙傳送

## 接線設定

DHT11: VCC → 5V, GND → GND, DATA → D2

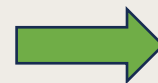
HC-SR501: VCC → 5V, GND → GND, OUT → D3

HC-05: VCC → 5V, GND → GND, TX → D10, RX → D11 (使用 SoftwareSerial)





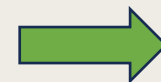
先使用 Arduino IDE 管理函式庫安裝 DHT sensor library 1.4.6



```
1 #include <SoftwareSerial.h>
2 #include <DHT.h>
```

引入兩個函式庫：

- SoftwareSerial：建立模擬的序列埠給 HC-05 使用
- DHT：讀取 DHT11 溫溼度感測器資料



```
4 // 感測器接腳定義
5 #define DHTPIN 2 // DHT11 資料腳位接到 D2
6 #define DHTTYPE DHT11 // 感測器型號
7 #define PIRPIN 3 // HC-SR501 輸出接 D3
```

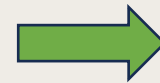
定義接腳：

DHT11 資料線接到 D2  
HC-SR501 的輸出線接到 D3  
DHT 感測器類型設為 DHT11

```
9 DHT dht(DHTPIN, DHTTYPE);
10 SoftwareSerial BTSerial(10, 11); // HC-05 TX接D10, RX接D11 (需降壓)
```

宣告 DHT 與 HC-05 用的序列通訊：

DHT11 感測器使用腳位 D2  
HC-05 模組接到 D10 ( RX ) 和 D11 ( TX )



```
12 void setup() {
13     dht.begin();
14     pinMode(PIRPIN, INPUT);
15
16     Serial.begin(9600); // Debug用
17     BTSerial.begin(9600); // 與 HC-05 通訊
18
19     Serial.println("系統啟動，開始傳送感測資料...");
20 }
```

初始化：

啟動 DHT 感測器  
設定 PIR 感測器為輸入  
開啟 Serial 和藍牙的通訊 ( 鮑率皆為 9600 )  
印出初始化訊息

```

void loop() {
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  int motion = digitalRead(PIRPIN); // 1: 有人, 0: 沒人

  // 檢查讀值是否成功
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println(" ! 無法讀取 DHT11 資料");
    return;
  }
}

```



```

33 // 組合要傳送的資料
34 String data = String(temperature, 1) + "," + String(humidity, 1) + "," + String(motion);
35
36 // 傳送到藍牙模組與序列監控視窗
37 BTSerial.println(data);
38 Serial.println("傳送: " + data);
39
40 delay(2000); // 每 2 秒傳送一次
41 }
42

```

組合傳送資料為格式：  
溫度,濕度,是否有人 例如：28.4,58.2,1  
傳送資料給電腦（藍牙）與 Serial Monitor（除錯）



程式上傳到Arduino執行

讀取感測器數值：  
溫度、濕度由 DHT11 提供  
motion 為人體移動狀態（1 = 有人，0 = 無人）  
如果溫溼度無法正確讀取（讀到 NaN），跳過此次迴圈並顯示警告

```

系統啟動，開始傳送感測資料...
傳送: 28.4,61.0,0
傳送: 28.6,61.0,0
傳送: 28.6,61.0,0
傳送: 28.4,61.0,0
傳送: 28.5,61.0,0
傳送: 28.5,61.0,0
傳送: 28.8,61.0,0
傳送: 28.6,61.0,0
傳送: 28.6,61.0,0
傳送: 28.6,61.0,0
傳送: 28.6,61.0,0

```

```

傳送: 28.6,72.0,1
傳送: 29.4,76.0,0
傳送: 28.9,77.0,0
傳送: 29.0,80.0,0
傳送: 29.4,81.0,0
傳送: 29.9,81.0,0
傳送: 29.9,81.0,1
傳送: 30.3,78.0,0
傳送: 29.9,71.0,1
傳送: 30.4,67.0,1
傳送: 30.8,68.0,1
傳送: 30.6,72.0,1

```

第一欄：溫度（℃）  
第二欄：濕度（%）  
第三欄：人體感測結果（0 = 無人，1 = 有人）

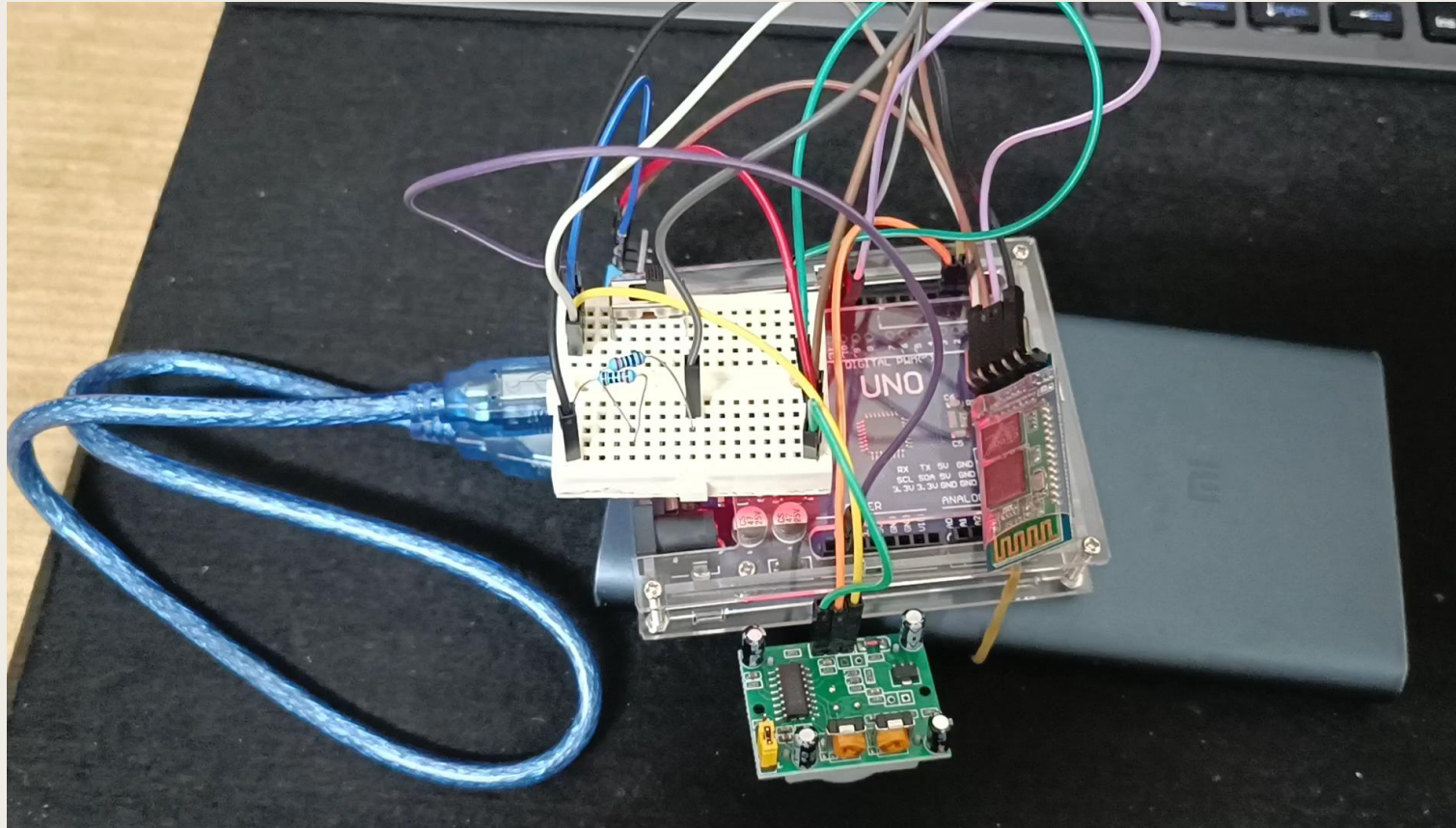


# 感測器數據格式與驗證

數據項目	範例值	說明
溫度	28.1	攝氏溫度 (°C)
濕度	60.0	相對濕度 (%)
是否有人	1	1 = 有人 / 0 = 無人

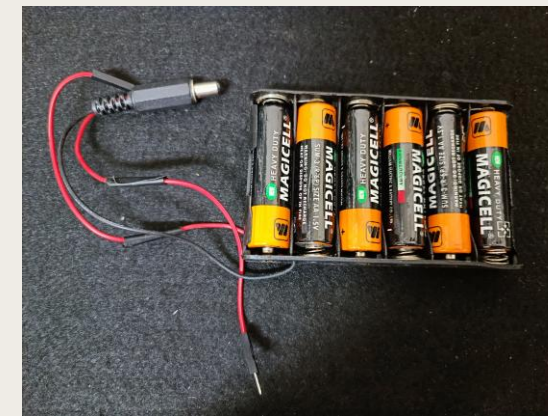
我們確保 Arduino 傳送的數據格式為「溫度,濕度,是否有人」，並透過 Serial Monitor 進行即時驗證，確保數據的準確性與一致性，為後續的 Python 處理奠定基礎。





# Arduino 感測與藍牙傳送

電源輸入選擇以行動電源作為Arduino主要電源供應來源。  
如果附近有插座也可以直接使用USB電源供應器供電  
(也可以使用電池盒，就是有點費電池)



Arduino 感測與藍牙傳送影片 連接: <https://youtu.be/NUiDoJqFR8w>

# 實作流程：藍牙對接

## 配對與接收設定

HC-05 藍牙模組名為 105127。在 Windows 藍牙裝置管理員中成功配對，並分配 COM5 作為資料接收埠。後續使用 `pyserial` 成功建立連線。

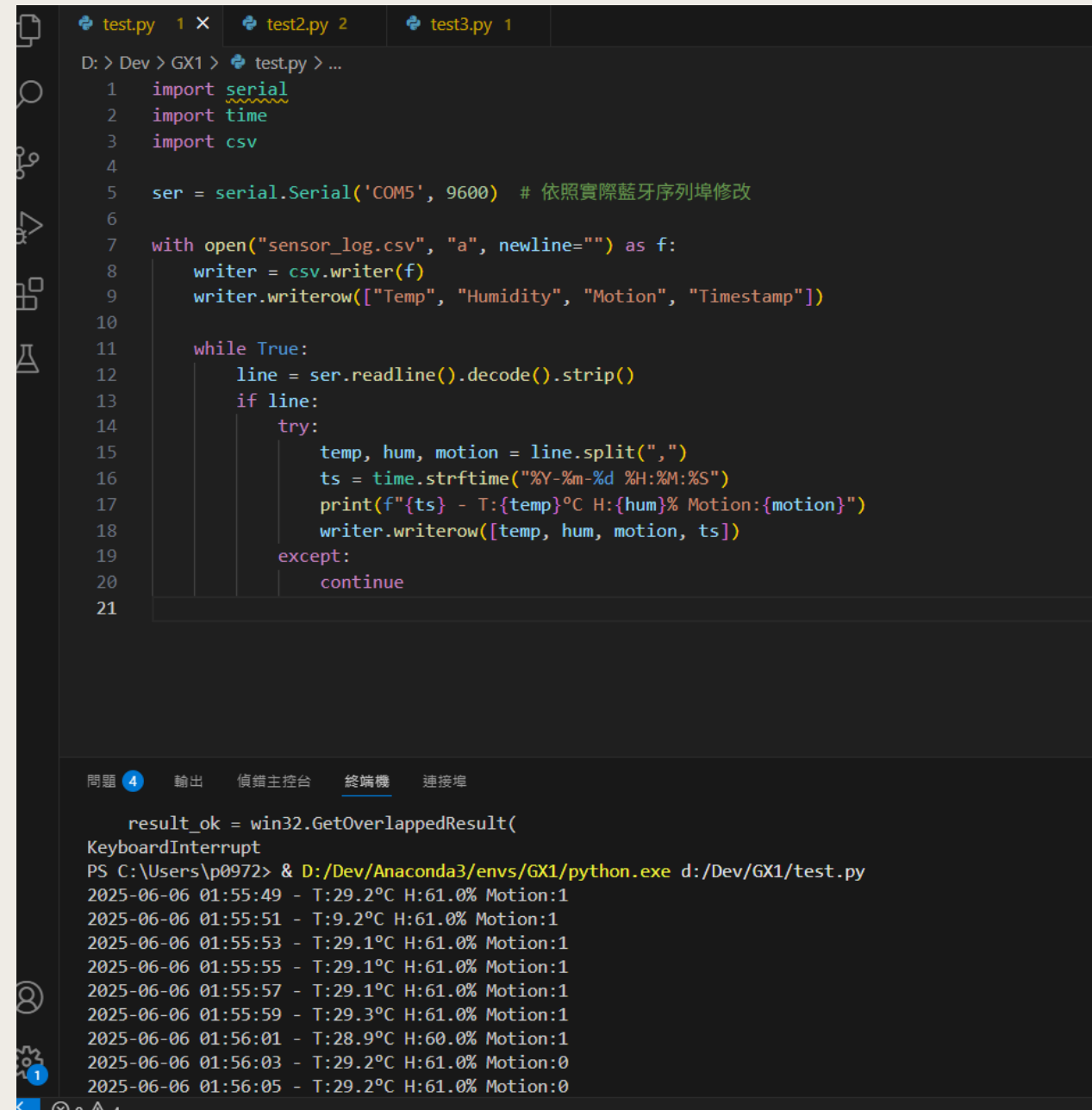


# 實作流程：Python 藍牙接收程式說明

從 HC-05 接收 Arduino 感測資料並寫入 CSV 檔案，同時在終端機輸出格式化資訊。

## ✓ 程式功能說明

- 與 Arduino 透過藍牙 ( COM5 ) 建立序列通訊
- 接收格式為 溫度,濕度,是否有人 的感測資料
- 將每筆資料記錄至 sensor\_log.csv ( 附加時間戳 )
- 並在畫面即時顯示資料內容 ( 含時間 )



```
test.py 1 x test2.py 2 test3.py 1
D: > Dev > GX1 > test.py > ...
1 import serial
2 import time
3 import csv
4
5 ser = serial.Serial('COM5', 9600) # 依照實際藍牙序列埠修改
6
7 with open("sensor_log.csv", "a", newline="") as f:
8     writer = csv.writer(f)
9     writer.writerow(["Temp", "Humidity", "Motion", "Timestamp"])
10
11 while True:
12     line = ser.readline().decode().strip()
13     if line:
14         try:
15             temp, hum, motion = line.split(",")
16             ts = time.strftime("%Y-%m-%d %H:%M:%S")
17             print(f"{ts} - T:{temp}°C H:{hum}% Motion:{motion}")
18             writer.writerow([temp, hum, motion, ts])
19         except:
20             continue
21
```

問題 4 輸出 偵錯主控台 終端機 連接埠

```
result_ok = win32.GetOverlappedResult(
KeyboardInterrupt
PS C:\Users\p0972> & D:/Dev/Anaconda3/envs/GX1/python.exe d:/Dev/GX1/test.py
2025-06-06 01:55:49 - T:29.2°C H:61.0% Motion:1
2025-06-06 01:55:51 - T:9.2°C H:61.0% Motion:1
2025-06-06 01:55:53 - T:29.1°C H:61.0% Motion:1
2025-06-06 01:55:55 - T:29.1°C H:61.0% Motion:1
2025-06-06 01:55:57 - T:29.1°C H:61.0% Motion:1
2025-06-06 01:55:59 - T:29.3°C H:61.0% Motion:1
2025-06-06 01:56:01 - T:28.9°C H:60.0% Motion:1
2025-06-06 01:56:03 - T:29.2°C H:61.0% Motion:0
2025-06-06 01:56:05 - T:29.2°C H:61.0% Motion:0
```

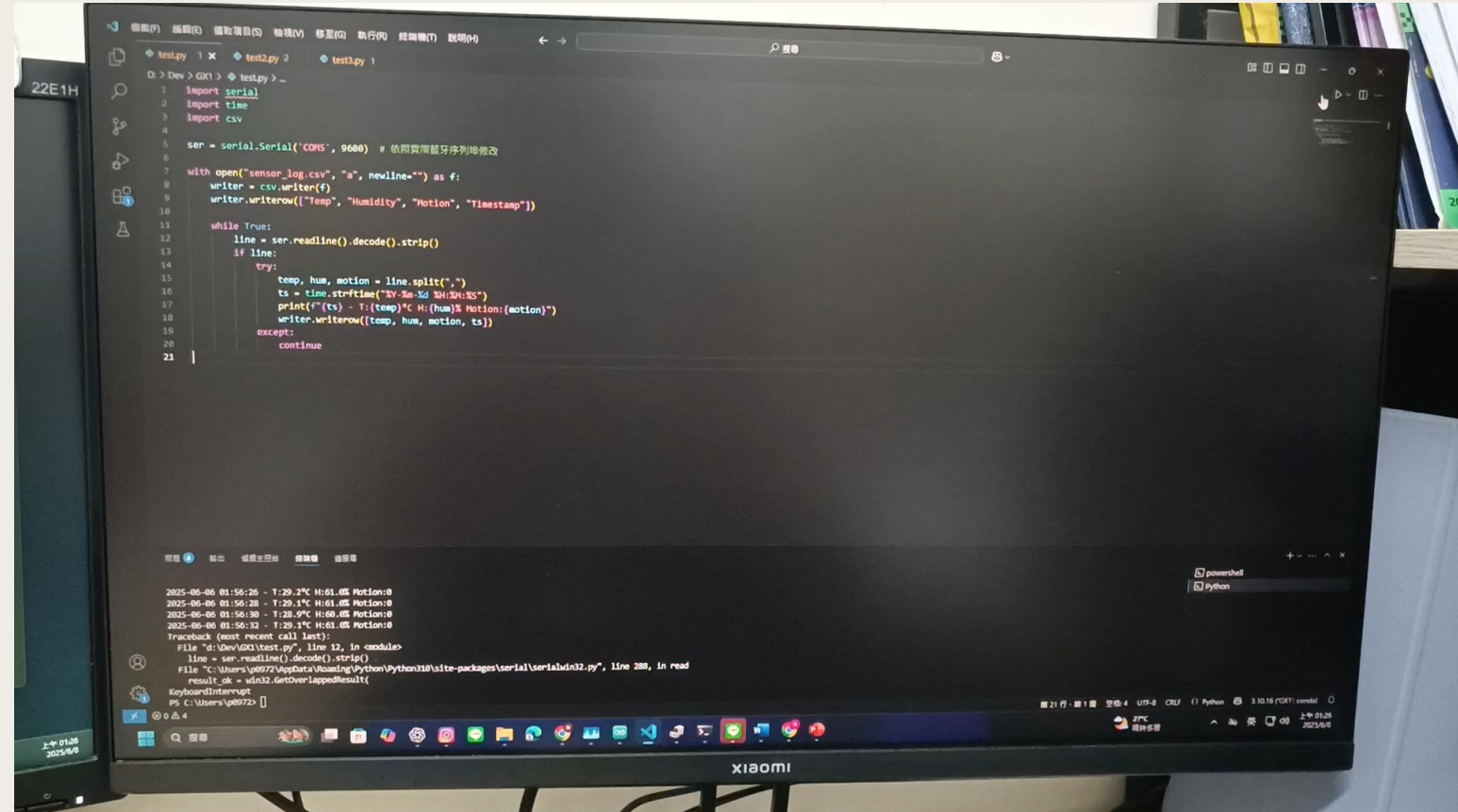


# 實作影片拍攝

Arduino 感測與藍牙傳送 + Python資料接收 影片連接: <https://youtu.be/NUiDoJqFR8w>

## 讀取與顯示資料

程式持續讀取 Arduino 傳送的資料，每筆資料以 "," 分解為三項數據：溫度 (°C)、濕度 (%) 與人體感測 (1 = 有人 / 0 = 無人)。初期版本會將這些數據顯示在終端機。



# 實作流程：Python 藍牙接收 並發送 LINE 推播-程式說明

## ✅ 程式總覽功能說明

- 透過 HC-05 藍牙模組接收來自 Arduino 的環境感測資料 ( 溫度、濕度、人體感測 )
- 判斷當下資料是否屬於「舒適」區間
- 根據是否有人與是否舒適的條件，自動產生 LINE 通知訊息並推送給指定使用者
- 將所有資料紀錄到 sensor\_log.csv 檔案中
- 每種情況具備冷卻時間，避免短時間重複通知

```
> Dev > CXX > testipy > _
1 import serial
2 import time
3 import csv
4 from linebot import LineBotApi
5 from linebot.models import TextSendMessage
6
7 # === LINE Bot 設定 ===
8 LINE_CHANNEL_ACCESS_TOKEN = _
9 TO_USER_ID = _
10
11 line_bot_api = LineBotApi(LINE_CHANNEL_ACCESS_TOKEN)
12
13 def send_linebot_message(message):
14     try:
15         line_bot_api.push_message(
16             TO_USER_ID,
17             TextSendMessage(text=message)
18         )
19         print("✅ 已發送 LINE Bot 通知")
20     except Exception as e:
21         print("❌ 傳送 LINE Bot 訊息失敗:", e)
22
23 # === 溫度是否舒適的判斷條件 ===
24 def is_comfortable(temp, hum):
25     return 25.0 <= temp <= 28.0 and 45 <= hum <= 65
26
27 # === HC-05 COM 埠設定 ===
28 PORT = 'COM5'
29 BAUD_RATE = 9600
30
31 try:
32     ser = serial.Serial(PORT, BAUD_RATE, timeout=1)
33     print(f"✅ 成功連接至 {PORT}")
34 except:
35     print(f"❌ 無法開啟 {PORT}，請確認 HC-05 是否配對成功並有對應 COM 埠")
36     exit()
37
38 # === 儲存資料 + 檢查條件 ===
39 with open("sensor_log.csv", "a", newline="") as f:
40     writer = csv.writer(f)
41     if f.tell() == 0:
42         writer.writerow(["Time", "Temperature", "Humidity", "Motion"])
43
44     print("⌘ 開始接收資料中，按 Ctrl+C 可停止")
45
46     try:
47         last_alert_time_person = 0
48         last_alert_time_empty = 0
49         cooldown = 60 # 通知冷卻時間 (秒)
50
51         while True:
52             line = ser.readline().decode().strip()
53             if line:
54                 try:
55                     temp, hum, motion = line.split(",")
56                     temp = float(temp)
57                     hum = float(hum)
58                     timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
59                     print(f"[{timestamp}] -> T:{temp}°C H:{hum}% 有人:{motion}")
60                     writer.writerow([timestamp, temp, hum, motion])
61
62                     comfort = is_comfortable(temp, hum)
63                     now = time.time()
64
65                     if motion.strip() == "1":
66                         if now - last_alert_time_person > cooldown:
67                             if not comfort:
68                                 send_linebot_message(
69                                     f"⚠ 有人進入環境\n⌚ 時間:{timestamp}\n🌡 溫度:{temp}°C\n💧 濕度:{hum}%\n⚠ 感測數據不舒服，建議開冷氣"
70                                 )
71                             else:
72                                 send_linebot_message(
73                                     f"⚠ 有人進入環境\n⌚ 時間:{timestamp}\n🌡 溫度:{temp}°C\n💧 濕度:{hum}%\n✅ 感測數據舒適，無需開冷氣"
74                                 )
75                             last_alert_time_person = now
76
77                     elif motion.strip() == "0":
78                         if now - last_alert_time_empty > cooldown:
79                             if comfort:
80                                 send_linebot_message(
81                                     f"⚠ 無人偵測中\n⌚ 時間:{timestamp}\n🌡 溫度:{temp}°C\n💧 濕度:{hum}%\n⚠ 感測數據舒適，是否忘記開冷氣?"
82                                 )
83                             else:
84                                 send_linebot_message(
85                                     f"⚠ 無人偵測中\n⌚ 時間:{timestamp}\n🌡 溫度:{temp}°C\n💧 濕度:{hum}%\n✅ 很棒，出門沒有忘記開冷氣!"
86                                 )
87                             last_alert_time_empty = now
88
89             except ValueError:
90                 print("⚠ 資料格式錯誤:", line)
91
92 except KeyboardInterrupt:
93     print("⏏ 已中止接收")
94     ser.close()
95
```





# 程式架構概述



## 舒適度邏輯判斷

系統根據溫濕度與人體感測數據，判斷環境舒適度。舒適條件為：  
 $26.0^{\circ}\text{C} \leq \text{溫度} \leq 28.0^{\circ}\text{C}$  且  $45\% \leq \text{濕度} \leq 65\%$ 。

## LINE Bot 主動推播整合

使用 line-bot-sdk 整合 LINE Bot，透過 `push_message()` 主動傳送感測分析結果至使用者 LINE 帳號，例如：「 有人進入環境   
溫度： $33.9^{\circ}\text{C}$   濕度：74.0%  
 感測數據不舒適，建議開冷氣」。

## 加入通知冷卻機制

為避免重複通知，系統導入冷卻機制，使用 `last_alert_time` 控制冷卻時間為 60 秒，確保通知的有效性與使用者體驗。

## ✅ LINE Bot 傳送訊息的需求

1. Channel Access Token ( 長長的一串 Bearer token )
2. 使用者 ID ( userId ) : 你想傳訊息給的對象

步驟：

1. 登入LINE Developers建立Messaging API Channel，找到 Channel Access Token、CHANNEL\_SECRET
2. 用 Flask 架設一個簡單的 Webhook，當你用戶端傳訊息給 LINE Bot 時，LINE 就會把該訊息送到你的伺服器，我們就可以從 event 裡抓出 userId。

```
PS C:\Users\p0972> & D:/Dev/Anaconda3/envs/GX1/python.exe d:/Dev/GX1/line_webhook_test.py
d:\Dev\GX1\line_webhook_test.py:12: LineBotSdkDeprecatedIn30: Call to deprecated class LineBotApi. (Use v3 class; linebot.
line_bot_api = LineBotApi(CHANNEL_ACCESS_TOKEN)
d:\Dev\GX1\line_webhook_test.py:13: LineBotSdkDeprecatedIn30: Call to deprecated class WebhookHandler. (Use 'from linebot.
since version 3.0.0.
handler = WebhookHandler(CHANNEL_SECRET)
* Serving Flask app 'line_webhook_test'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
✅ 你的 User ID 是：
U99735fe437538daac832e55513d08724
d:\Dev\GX1\line_webhook_test.py:34: LineBotSdkDeprecatedIn30: Call to deprecated method reply_message. (Use 'from linebot.
EADME.rst for more details.) -- Deprecated since version 3.0.0.
line_bot_api.reply_message(
127.0.0.1 - - [09/Jun/2025 14:23:29] "POST /webhook HTTP/1.1" 200 -
△ 4
```

```
D: > Dev > GX1 > line_webhook_test.py > ...
1 from flask import Flask, request, abort
2 from linebot import LineBotApi, WebhookHandler
3 from linebot.exceptions import InvalidSignatureError
4 from linebot.models import MessageEvent, TextMessage, TextSendMessage
5
6 app = Flask(__name__)
7
8 # ==== 換成你自己的 LINE Bot 資訊 ====
9 CHANNEL_ACCESS_TOKEN = '你的 Channel Access Token'
10 CHANNEL_SECRET = '你的 Channel Secret'
11
12 line_bot_api = LineBotApi(CHANNEL_ACCESS_TOKEN)
13 handler = WebhookHandler(CHANNEL_SECRET)
14
15 @app.route("/callback", methods=['POST'])
16 def callback():
17     signature = request.headers['X-Line-Signature']
18     body = request.get_data(as_text=True)
19
20     try:
21         handler.handle(body, signature)
22     except InvalidSignatureError:
23         abort(400)
24
25     return 'OK'
26
27 @handler.add(MessageEvent, message=TextMessage)
28 def handle_message(event):
29     user_id = event.source.user_id
30     message = f"✅ 你的 User ID 是：\n{user_id}"
31     print(message)
32
33     # 回傳給使用者
34     line_bot_api.reply_message(
35         event.reply_token,
36         TextSendMessage(text=message)
37     )
38
39 if __name__ == "__main__":
40     app.run(port=5000)
41
```



# 主程式結構與區塊解說

## 1. 套件匯入與 LINE 設定

```
1 import serial
2 import time
3 import csv
4 from linebot import LineBotApi
5 from linebot.models import TextSendMessage
```

- 使用 serial 接收藍牙資料
- 使用 line-bot-sdk 推送 LINE 訊息

```
# === LINE Bot 設定 ===
LINE_CHANNEL_ACCESS_TOKEN =
TO_USER_ID =

line_bot_api = LineBotApi(LINE_CHANNEL_ACCESS_TOKEN)
```

- 設定 LINE 機器人的 Access Token 與目標使用者 ID
- 使用 LineBotApi 初始化推播接口

## 2. 發送 LINE 訊息函式

```
13 def send_linebot_message(message):
14     try:
15         line_bot_api.push_message(
16             TO_USER_ID,
17             TextSendMessage(text=message)
18         )
19         print("✅ 已發送 LINE Bot 通知")
20     except Exception as e:
21         print("❌ 傳送 LINE Bot 訊息失敗:", e)
```

- 封裝 push\_message() 為簡易函式，輸入文字訊息即可發送
- 若發送失敗會印出錯誤資訊

## 3. 判斷舒適區間條件

```
23 # === 溫濕度是否舒適的判斷條件 ===
24 def is_comfortable(temp, hum):
25     return 26.0 <= temp <= 28.0 and 45 <= hum <= 65
```

- 若溫度與濕度同時落在範圍內，即判定為「舒適」

## 4. 建立藍牙串口連接

```
27 # === HC-05 COM 埠設定 ===
28 PORT = 'COM5'
29 BAUD_RATE = 9600
30
31 try:
32     ser = serial.Serial(PORT, BAUD_RATE, timeout=1)
33     print(f"✅ 成功連接至 {PORT}")
34 except:
35     print(f"❌ 無法開啟 {PORT}，請確認 HC-05 是否配對成功並有對應 COM 埠")
36     exit()
```

- 嘗試建立與 HC-05 的連線
- 若失敗則退出程式

## 5. 寫入 CSV 並接收資料

```
38 # === 儲存資料 + 檢查條件 ===
39 with open("sensor_log.csv", "a", newline="") as f:
40     writer = csv.writer(f)
41     if f.tell() == 0:
42         writer.writerow(["Time", "Temperature", "Humidity", "Motion"])
43
44     print("👉 開始接收資料中，按 Ctrl+C 可停止")
```

- 每次程式啟動會自動建立檔案與欄位（若是新檔案）
- 每次接收到有效資料即寫入一筆紀錄



# 程式結構與區塊解說

## 6. 主接收與通知迴圈、判斷邏輯與分支

```
while True:
    line = ser.readline().decode().strip()
    if line:
        try:
            temp, hum, motion = line.split(",")
            temp = float(temp)
            hum = float(hum)
            timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
            print(f"{timestamp} -> T:{temp}°C H:{hum}% 有人:{motion}")
            writer.writerow([timestamp, temp, hum, motion])

            comfort = is_comfortable(temp, hum)
            now = time.time()

            if motion.strip() == "1":
                if now - last_alert_time_person > cooldown:
                    if not comfort:
                        send_linebot_message(
                            f"👤 有人進入環境🕒 時間:{timestamp}\n🌡️ 溫度:{temp}°C💧 濕度:{hum}%🚨 感測數據不舒適，建議開冷氣"
                        )
                    else:
                        send_linebot_message(
                            f"👤 有人進入環境🕒 時間:{timestamp}\n🌡️ 溫度:{temp}°C💧 濕度:{hum}%✅ 感測數據舒適，無需開冷氣"
                        )
                    last_alert_time_person = now

            elif motion.strip() == "0":
                if now - last_alert_time_empty > cooldown:
                    if comfort:
                        send_linebot_message(
                            f"👤 無人偵測中🕒 時間:{timestamp}\n🌡️ 溫度:{temp}°C💧 濕度:{hum}%🚨 感測數據舒適，是否忘記關冷氣？"
                        )
                    else:
                        send_linebot_message(
                            f"👤 無人偵測中🕒 時間:{timestamp}\n🌡️ 溫度:{temp}°C💧 濕度:{hum}%✅ 很棒，出門沒有忘記關冷氣！"
                        )
                    last_alert_time_empty = now

        except ValueError:
            print("⚠️ 資料格式錯誤:", line)
```

- 每筆資料格式應為：溫度,濕度,是否有人
  - 若解析失敗則跳過該筆資料
- 有人 ( motion == 1 ) 時：
- 若 不舒適 → 發送建議開冷氣訊息
  - 若 舒適 → 發送「無需開冷氣」提示
- 無人 ( motion == 0 ) 時：
- 若 舒適 → 發送「可能忘記關冷氣」提示
  - 若 不舒適 → 發送「很棒，出門沒忘關冷氣」稱讚訊息

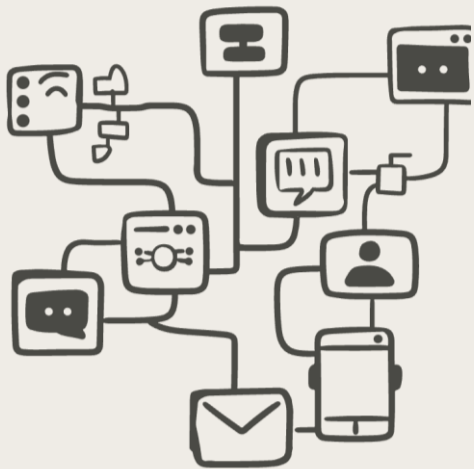
## 8. 通知冷卻控制

```
try:
    last_alert_time_person = 0
    last_alert_time_empty = 0
    cooldown = 60 # 通知冷卻時間 (秒)
```

- 使用兩個冷卻計時器分別針對有人/無人狀況
- 60 秒內不會重複推播相同類型訊息，避免洗版



# LINE 推播實測畫面



## Python 串接與 LINE Bot 通知畫面

- 圖(一)為 Python 程式終端畫面：顯示藍牙連線成功、資料接收即時資訊與 LINE 通知發送紀錄。
- 圖(二)為 LINE App 中的聊天訊息實測畫面，系統根據不同環境條件，自動分析並產生對應提示

系統判斷邏輯混淆矩陣		
	不舒適環境 (X)	舒適環境 (✓)
有人 (1)	建議開冷氣 ❄️	無需開冷氣 ✓
無人 (0)	稱讚使用者沒忘記關冷氣 🎉	提醒是否忘記關冷氣 🔔

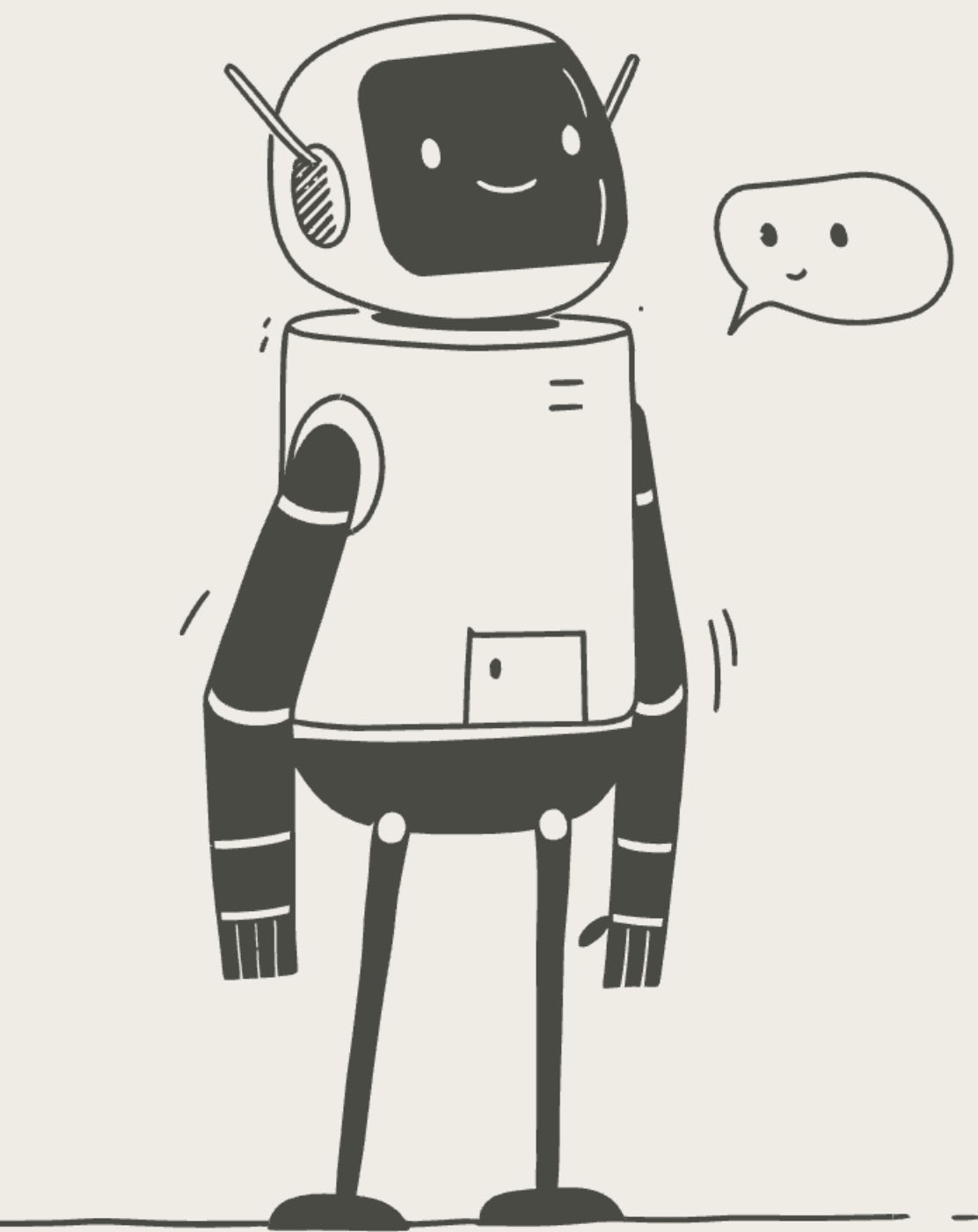


```
PS C:\Users\p0972> & D:/Dev/Anaconda3/envs/GX1/python.exe d:/Dev/GX1/test3.py
d:\Dev\GX1\test3.py:11: LineBotSdkDeprecatedIn30: Call to deprecated class LineBotApi
line_bot_api = LineBotApi(LINE_CHANNEL_ACCESS_TOKEN)
✓ 成功連接至 COM5
✎ 開始接收資料中，按 Ctrl+C 可停止
2025-06-06 00:20:59 -> T:33.2°C H:91.0% 有人:0
d:\Dev\GX1\test3.py:15: LineBotSdkDeprecatedIn30: Call to deprecated method push_message
more details.) -- Deprecated since version 3.0.0.
line_bot_api.push_message(
✓ 已發送 LINE Bot 通知
2025-06-06 00:21:01 -> T:33.3°C H:91.0% 有人:0
2025-06-06 00:21:03 -> T:33.4°C H:91.0% 有人:0
2025-06-06 00:21:05 -> T:33.4°C H:90.0% 有人:0
2025-06-06 00:21:07 -> T:33.4°C H:90.0% 有人:0
2025-06-06 00:21:09 -> T:33.6°C H:89.0% 有人:0
■ 已中止接收
```

圖(一) Python 程式終端畫面



圖(二) LINE App 中的聊天訊息實測畫面



# 實作流程：Python 藍牙接收並結合 Ollama 模型分析

## 整合目的

將系統從條件式邏輯升級為具備自然語言理解與回答能力，讓使用者可直接向 LINE Bot 發問，例如：「今天室內舒適嗎？」

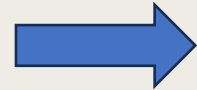
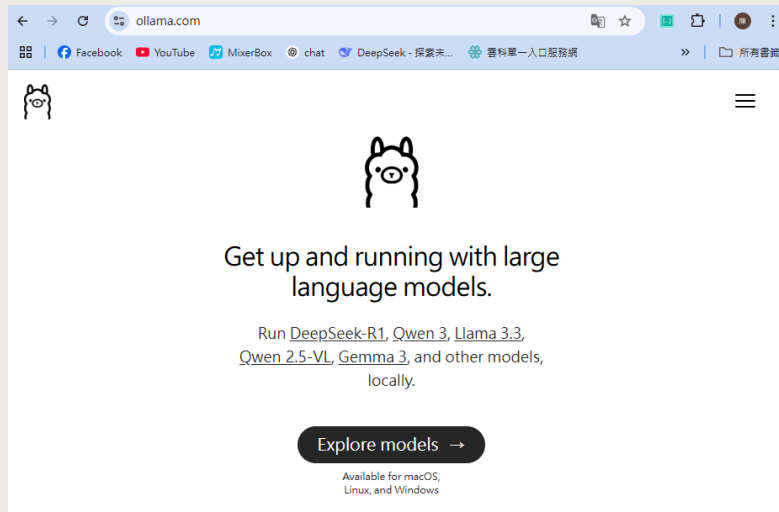
## 技術實作

安裝與執行本地語言模型：ollama run qwen3:8b。使用 Flask 架設 /webhook 接收使用者訊息，並讀取最新感測值組合 prompt 傳入模型。

## 實測效果

模型回覆更有彈性且具人性化說明，例如：「目前環境偏熱且濕度較高，建議打開冷氣並保持通風。」

# 安裝與啟用 Ollama



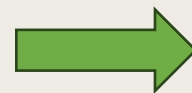
前置作業：

1. 打開終端機輸入ollama --version若顯示版本資訊即表示安裝成功。
2. 拉取指定模型 ( 以 Qwen3-8B 為例，適合顯存8GB的顯示卡 ) :ollama pull qwen3:8b
3. 執行本地模型進行測試 : ollama run qwen3:8b
4. 安裝 Ollama Python SDK : pip install ollama

不然會顯示ModuleNotFoundError: No module named 'ollama'

基本使用範例：

```
1 import ollama
2
3 response = ollama.chat(
4     model='qwen3:8b',
5     messages=[
6         {"role": "user", "content": "目前室內舒適嗎?"}
7     ]
8 )
9 print(response['message']['content'])
10 # response = ollama.chat(
```



```
PS C:\Users\p0972> & D:/Dev/Anaconda3/envs/GKI/python.exe d:/Dev/GKI/line_webhook_test.py
<think>
好的，用户问目前室内的舒适度如何。首先，我需要确定用户的具体情况。他们可能是在家、办公室，或者某个特定场所。因为没有提供具体信息，所以得先询问清楚。

接下来，用户可能想知道温度、湿度、空气质量，或者是否有噪音、光线等问题。不同的环境因素会影响舒适度，比如夏天可能更关注温度和空调，冬天则是取暖和湿度。

另外，用户可能没有明确说出需求，比如他们可能想调整环境但不知道如何操作，或者遇到某些问题却不知道如何解决。比如，如果温度过高，他们可能需要建议如何降温，或者如果空气干燥，可能需要加湿器的建议。

还要考虑用户可能的隐藏需求。比如，他们可能在工作或学习，需要一个安静、舒适的环境，或者他们可能有健康问题，比如过敏，所以空气质量很重要。

需要分步骤询问，先确认用户所在的环境，然后了解他们关心的具体方面，比如温度、湿度、噪音等。这样可以更有针对性地提供帮助，而不是泛泛而谈。

另外，要避免假设，比如不能直接说“室温是25度”，因为用户可能在不同的地方。需要先确认，再根据他们的回答给出具体建议。同时，保持回答简洁，避免信息过载，让用户容易理解和回应。
</think>

要判断目前室内的舒适度，我需要了解一些具体信息：

1. **您所在的环境**（例如：家中、办公室、公共场所等）
2. **您关注的舒适因素**（如温度、湿度、空气质量、噪音、光线等）
3. **是否有特殊需求**（如需要安静、通风、避免过敏源等）

例如：
- 如果您在家，可以告诉我室温、是否有空调/暖气、空气是否流通等。
- 如果您在办公室，可以描述是否闷热/寒冷，是否有异味或噪音等。

请提供更多细节，我会帮您分析当前的舒适度并给出建议！ 😊
PS C:\Users\p0972>
```

# 啟動 ngrok、設定 LINE Developers Webhook

執行指令：ngrok http 5000

```
D:\Dev\GX1\ngrok-v3-stable-v x
ngrok (Ctrl+C to quit)
Take our ngrok in production survey! https://forms.gle/aXiBFWzEA36DudFn6
Session Status online
Account GuanXin (Plan: Free)
Update update available (version 3.23.0, Ctrl-U to update)
Version 3.22.1
Region Japan (jp)
Latency 39ms
Web Interface http://127.0.0.1:4040
Forwarding https://4c85-36-232-219-64.ngrok-free.app -> http://localhost:5000
Connections
  ttl  opn  rt1  rt5  p50  p90
   28    0   0.00 0.00  1.99 2.01
HTTP Requests
-----
14:23:29.347 CST POST /webhook 200 OK
14:22:35.164 CST POST /webhook 404 NOT FOUND
01:40:33.891 CST POST /webhook
01:38:51.053 CST POST /webhook
01:37:41.483 CST POST /webhook
01:36:27.881 CST POST /webhook
01:34:39.898 CST POST /webhook
01:34:14.893 CST POST /webhook
01:32:34.505 CST POST /webhook 400 BAD REQUEST
01:31:31.695 CST POST /webhook
```

### Messaging API

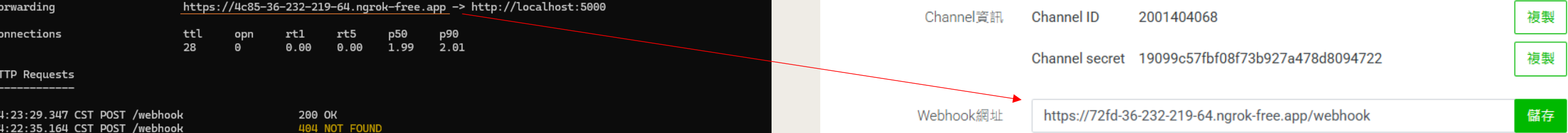
Messaging API為針對開發者所設計的進階功能。您可透過API收發訊息及動作，與LINE用戶進行更多互動。

[什麼是Messaging API ?](#)

[LINE Developers的API相關文件](#)

	狀態	使用中	
Channel資訊	Channel ID	2001404068	<button>複製</button>
	Channel secret	19099c57fbf08f73b927a478d8094722	<button>複製</button>
Webhook網址	<div>https://72fd-36-232-219-64.ngrok-free.app/webhook</div>		<button>儲存</button>

您可由[LINE Developers Console](#)進行其他設定。



# 主程式結構與區塊解說

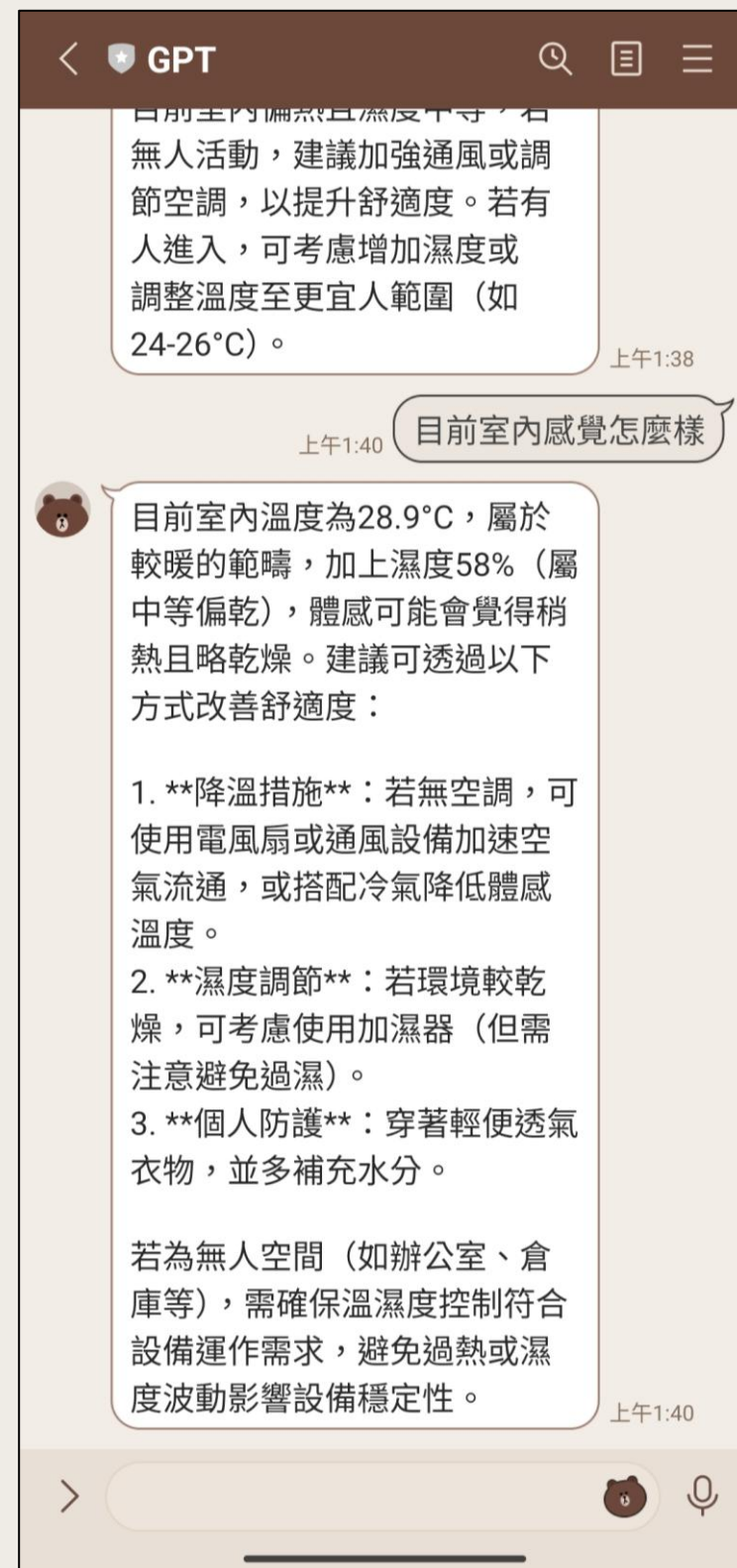
程式區塊	功能說明
<code>read_sensor_line()</code>	讀取並解析 Arduino 傳來的藍牙感測資料
<code>@app.route("/webhook")</code>	建立 Flask Webhook API，接收 LINE 的訊息事件
<code>handle_message()</code>	當使用者傳訊息給 Bot 時觸發，提取訊息內容與感測資料
<code>system_prompt</code>	將感測資料結合使用者訊息組成自然語言 Prompt 傳給語言模型
<code>ollama.chat()</code>	呼叫本地執行中的 <code>qwen3:8b</code> 模型進行回應推論
<code>re.sub(...)</code>	自動移除模型中 <code>&lt;think&gt;...&lt;/think&gt;</code> 思考提示內容，保持回覆乾淨
<code>line_bot_api.reply_message(...)</code>	將模型回應以文字訊息方式推送回給 LINE 使用者

```
1 from flask import Flask, request
2 import serial
3 import ollama
4 import re
5 from linebot import LineBotApi, WebhookHandler
6 from linebot.exceptions import InvalidSignatureError
7 from linebot.models import MessageEvent, TextMessage, TextSendMessage
8
9 app = Flask(__name__)
10
11 # --- LINE Bot 設定 ---
12 LINE_CHANNEL_ACCESS_TOKEN = 'KgbX3Tk6J80rhIj5RzV9km+eCkgJwX7Y0VgSuIV770rLKnltV0pG+LdMZvq2xz7skNo4i5FioPFj6WbSG0XVO Dc1ALutYYIk08F1x8I6T0Z4n4zfGp8+5Cu5+'
13 LINE_CHANNEL_SECRET = '198999c57fbf08f73b927a478d8094722'
14 line_bot_api = LineBotApi(LINE_CHANNEL_ACCESS_TOKEN)
15 handler = WebhookHandler(LINE_CHANNEL_SECRET)
16
17 # --- HC-05 藍牙 COM 設定 ---
18 PORT = 'COM5'
19 BAUD_RATE = 9600
20 ser = serial.Serial(PORT, BAUD_RATE, timeout=2)
21
22 def read_sensor_line():
23     try:
24         line = ser.readline().decode().strip()
25         if line:
26             temp, hum, motion = line.split(",")
27             return {
28                 "temp": temp,
29                 "hum": hum,
30                 "motion": motion
31             }
32     except:
33         return None
34
35 @app.route("/webhook", methods=['POST'])
36 def webhook():
37     signature = request.headers['X-Line-Signature']
38     body = request.get_data(as_text=True)
39
40     try:
41         handler.handle(body, signature)
42     except InvalidSignatureError:
43         return 'Invalid signature', 400
44
45     return 'OK', 200
46
47 @handler.add(MessageEvent, message=TextMessage)
48 def handle_message(event):
49     user_message = event.message.text
50     sensor_data = read_sensor_line()
51
52     if not sensor_data:
53         reply = "目前無法讀取感測資料，請稍後再試。"
54     else:
55         system_prompt = f"""
56 目前偵測到的環境資料如下：
57 - 溫度：{sensor_data['temp']}°C
58 - 濕度：{sensor_data['hum']}%
59 - 是否有人：{'有' if sensor_data['motion']==1 else '無'}
60
61 請根據以上資料回答使用者問題：{user_message}
62 """
63
64     try:
65         response = ollama.chat(
66             model='qwen3:8b',
67             messages=[
68                 ("role": "system", "content": "你是環境感測分析專家。"),
69                 ("role": "user", "content": system_prompt)
70             ]
71         )
72         raw_reply = response['message']['content']
73         # 移除 <think> ... </think> 區塊 (包含內容)
74         reply = re.sub(r"<think>.*?</think>", "", raw_reply, flags=re.DOTALL).strip()
75     except Exception as e:
76         print(f"❌ Ollama 錯誤：{e}")
77         reply = "❌ 模型處理錯誤，請稍後再試。"
78
79     line_bot_api.reply_message(
80         event.reply_token,
81         TextSendMessage(text=reply)
82     )
83
84 if __name__ == "__main__":
85     app.run(port=5000)
86     print(f"✅ Flask 伺服器已啟動，監聽端口 5000")
87     print(f"✅ 已連接至 HC-05 藍牙模組，COM 埠：{PORT}")
```



# LLM問答實測畫面

圖中顯示使用者透過 LINE Bot 發問「目前室內感覺怎麼樣」，系統即時讀取感測數據（28.9°C / 58% / 無人），並透過本地語言模型 Qwen3-8B 生成完整自然語言建議，包括降溫措施、濕度調節與個人防護建議等。此測試顯示模型已具備 contextual reasoning 能力，可根據即時環境資訊生成具體情境建議。





# 系統架構概覽

**Arduino 感測**  
DHT11 溫溼度、HC-SR501 人體感測

**LINE Bot 互動**  
通知與自然語言問答



**藍牙傳輸**  
HC-05 模組數據傳送

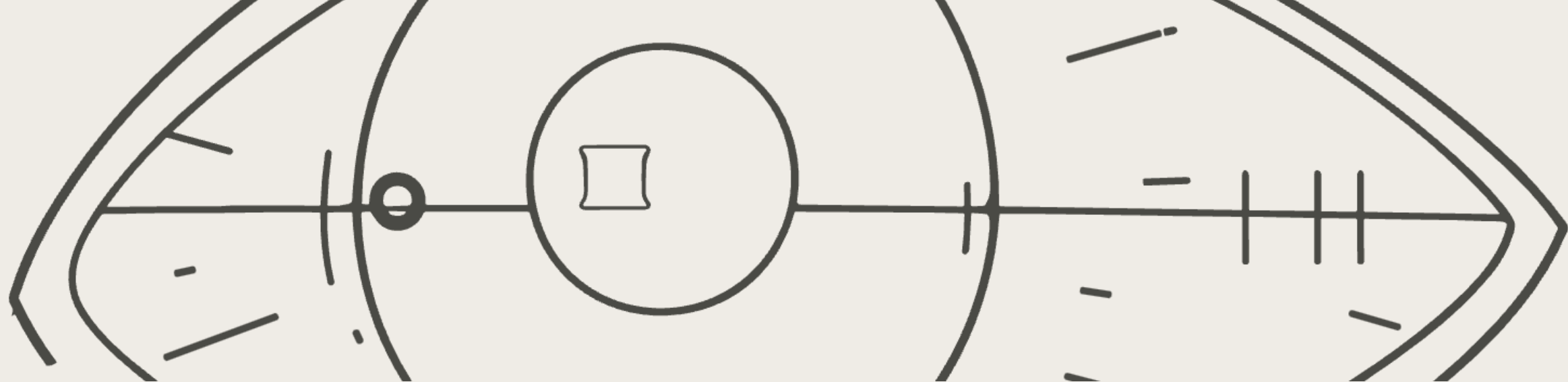
**Python 處理**  
數據接收與分析

整個系統的設計提供一個從硬體感測到軟體智慧互動的完整解決方案，實現環境監控與自動化通知，並透過 AI 提升使用者體驗。

# Ollama 模型自然語言互動



透過整合 Ollama 本地語言模型，LINE Bot 不僅能提供條件式通知，更能理解使用者提問並給出智慧回覆，例如針對「是否需要開冷氣？」的問題，模型會回覆：「目前環境偏熱且濕度較高，建議打開冷氣並保持空氣循環。」



# 未來展望與總結

## 系統穩定性

持續優化藍牙通訊穩定性與數據傳輸效率。

## 功能擴展

考慮加入更多感測器類型，如光照、空氣品質等。

## AI 模型優化

提升 Ollama 模型在環境控制建議方面的精準度與語義理解能力。

本次專題成功整合了感測器數據分析、無線通訊與 AI 模型的整合，實現智慧環境監測與互動通報。未來將持續改進系統，使其更加穩定、功能更豐富，並探索更多 AI 在物聯網應用中的可能性。