

AIoT 空氣品質監測 系統-期末報告

專題目標

- ▶ 設計低成本空氣監測裝置
- ▶ 利用感測器收集資料
- ▶ 網路傳輸到雲端或網頁
- ▶ AI 判斷空氣品質狀態

系統架構

- ▶ 感測空氣品質、溫濕度
- ▶ 資料傳送到雲端 (如 Google Sheets)
- ▶ 用 Python 做資料接收與簡單 AI 迴歸
- ▶ 網頁顯示結果或通知

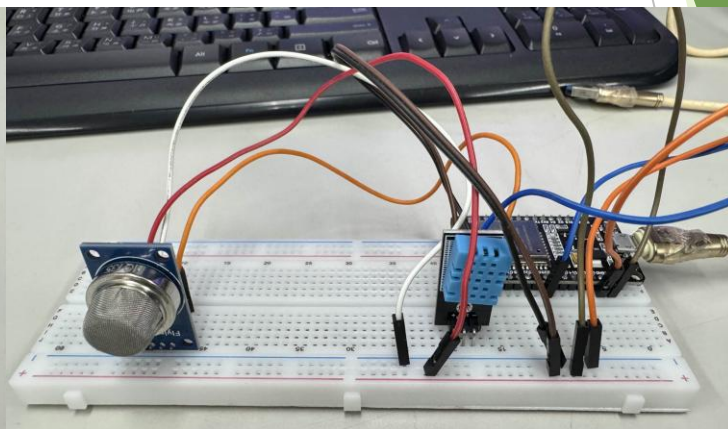
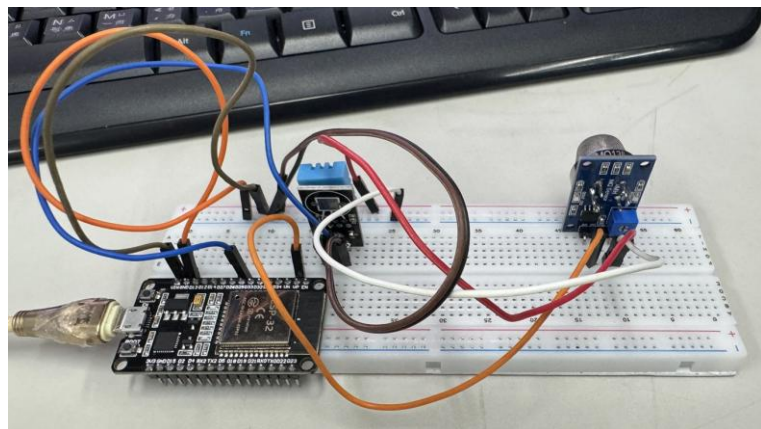
感測空氣品質、溫濕度

- ▶ - ESP32：控制板 + Wi-Fi
- ▶ - DHT11：溫濕度感測
- ▶ - MQ135：空氣品質感測器

硬體設備介紹與接線說明

- ▶ **ESP32 DevKit V1**
 - ▶ 多功能 Wi-Fi 控制板，支援 Arduino IDE
 - ▶ 提供多個 GPIO 腳位供感測器使用
- ▶ **DHT11 (溫濕度感測器) 輸出：數位訊號**
 - ▶ VCC → ESP32 5V
 - ▶ GND → ESP32 GND
 - ▶ DATA → ESP32 GPIO25
- ▶ **MQ135 (空氣品質感測器) 輸出：類比訊號**
 - ▶ VCC → ESP32 5V
 - ▶ GND → ESP32 GND
 - ▶ AO (類比輸出) → ESP32 GPIO36 (VP)

接線圖



1. Wi-Fi 設定區塊
2. Google Script URL 區塊
3. DHT11 與 MQ135 硬體設定
4. `setup()` 初始化階段
5. `loop()` 讀取感測器資料

```
1 #include <WiFi.h>
2 #include <HTTPClient.h>
3 #include "DHT.h"
4
5 //===== WiFi 設定 =====
6 const char* ssid = " ";
7 const char* password = " ";
8
9 //===== Google Apps Script Web App URL =====
10 const char* scriptURL = "https://script.google.com/macros/s/AKfycbz_Sn8oru0WwFwJ4YU10P1Xg0I-1nD5T2gnrVh048Sb4Db5snfGaDttk8RnJUCTVA/exec";
11
12 //===== DHT11 設定 =====
13 #define DHTPIN 25
14 #define DHTTYPE DHT11
15 DHT dht(DHTPIN, DHTTYPE);
16
17 //===== MQ135 設定 =====
18 #define MQ135_PIN 36 // GPIO36 是 ESP32 的 ADC 腳 (VP)
19
20 void setup() {
21   Serial.begin(115200);
22   delay(1000);
23
24   Serial.println("🔌 連線到 WiFi...");
25   WiFi.begin(ssid, password);
26
27   while (WiFi.status() != WL_CONNECTED) {
28     delay(500);
29     Serial.print(".");
30   }
31
32   Serial.println("\n✅ WiFi 已連線");
33   Serial.print("📍 IP 位址: ");
34   Serial.println(WiFi.localIP());
35
36   dht.begin(); // 啟動 DHT11 感測器
37   analogReadResolution(12); // 設定 ADC 解析度 (精確12-bit, 0~4095)
38 }
39
40 void loop() {
41   float temperature = dht.readTemperature();
42   float humidity = dht.readHumidity();
43   int mq135_value = analogRead(MQ135_PIN); // 讀取 MQ135 類比值 (0~4095)
44
45   if (isnan(temperature) || isnan(humidity) || isnan(mq135_value)) {
46     Serial.println("❌ 感測器讀取失敗，跳過這次上傳。");
47   } else {
48     Serial.print("🌡️ 溫度: ");
49     Serial.print(temperature);
50     Serial.print(" °C 💧 濕度: ");
51     Serial.print(humidity);
52     Serial.print(" % 🌿 MQ135: ");
53     Serial.println(mq135_value);
54
55     sendToSheet(temperature, humidity, mq135_value);
56
57   }
58   delay(60000); // 每 60 秒更新一次
59 }
```

資料傳送到雲端 (硬體端)

```
void sendToSheet(float temp, float hum, int mq) {  
  if (WiFi.status() == WL_CONNECTED) {  
    HTTPClient http;  
    http.begin(scriptURL);  
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");  
  
    String postData = "temp=" + String(temp) + "&hum=" + String(hum) + "&mq=" + String(mq);  
    int httpCode = http.POST(postData);  
  
    if (httpCode > 0) {  
      Serial.println("✅ 上傳成功: HTTP " + String(httpCode));  
      String payload = http.getString();  
      Serial.println("伺服器回應: " + payload);  
    } else {  
      Serial.println("⚠️ 上傳失敗: HTTP " + String(httpCode));  
    }  
  
    http.end();  
  } else {  
    Serial.println("❌ WiFi 已斷線, 無法上傳");  
  }  
}
```

🌡️ 溫度: 28.40 °C 💧 濕度: 40.00 % 🌬️ MQ135: 1

✅ 上傳成功: HTTP 302

伺服器回應: <HTML>

<HEAD>

<TITLE>Moved Temporarily</TITLE>

</HEAD>

<BODY BGCOLOR="#FFFFFF" TEXT="#000000">

<!-- GSE Default Error -->

<H1>Moved Temporarily</H1>

The document has moved <A HREF="https://script.googleusercontent.com/macros/echo?user_content_key=

</BODY>

</HTML>

資料傳送到雲端 (網路端)

- ▶ 新增一個google sheet，作為資料庫
- ▶ 每一列有5個data：Time，Temperature，Humidity，MQ135，Quality，前4個是要上傳的參數，最後一個是要預測的參數

資料傳送到雲端 (網路端)

- ▶ 為了實作接收功能, 必須要使用google sheet的內建功能:APP Script
- ▶ APP Script新增doPost函數, 使其能夠每接收一次資料就新增一次row

```
function doPost(e) {  
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("ESP32");  
  var time = new Date();  
  var temp = e.parameter.temp;  
  var hum = e.parameter.hum;  
  var mq = e.parameter.mq;  
  
  sheet.appendRow([time, temp, hum, mq]);  
  return ContentService.createTextOutput("Success");  
}
```

簡單 AI 迴歸

- ▶ - 基本模型：Random Forest Regressor
- ▶ - 執行功能：使用溫度、濕度、MQ135
預測粉塵密度

簡單 AI 迴歸

- ▶ 為了讓Google sheet調用訓練的模型, 必須要將模型包成API
- ▶ 使用Flask模組將模型包成 API

```
# ===== 建立 Flask API =====  
app = Flask(__name__)  
  
@app.route('/')  
def home():  
    return '✅ Flask API is running'  
  
@app.route('/predict', methods=['POST'])  
def predict():  
    try:  
        data = request.get_json()  
        features = np.array(data["features"]).reshape(1, -1)  
        prediction = model.predict(features)  
        return jsonify({"prediction": float(prediction[0])})  
    except Exception as e:  
        return jsonify({"error": str(e)}), 400  
  
# ✅ 啟動 Flask  
if __name__ == '__main__':  
    app.run(host='0.0.0.0', port=5000)
```

簡單 AI 迴歸

- ▶ 包成API後，必須要將模型的API Deploy,使google sheet的app script 能夠調用模型
- ▶ 安裝 Ngrok , 執行 ngrok http 5000 , 會得到一個網址
- ▶ 該網址可以提供App script調用API

```
ngrok
♦ Want to hang with ngrok on our new Discord? http://ngrok.com/discord

Session Status      online
ccount              kia109062337@gapp.nthu.edu.tw (Plan: Free)
Update              update available (version 3.23.1, Ctrl-U to update)
Version              3.23.0
Region              Japan (jp)
Latency              46ms
Web Interface        http://127.0.0.1:4040
Forwarding           https://5aa2-140-114-26-118.ngrok-free.app -> http://localhost:5000

Connections          ttl      opn      rt1      rt5      p50      p90
396                0        0.00     0.00     0.03     0.03

HTTP Requests
-----
16:01:56.440 CST POST /predict          400 BAD REQUEST
16:00:17.267 CST POST /predict          200 OK
16:00:17.079 CST POST /predict          400 BAD REQUEST
16:00:17.357 CST POST /predict          400 BAD REQUEST
16:00:17.516 CST POST /predict          400 BAD REQUEST
16:00:17.176 CST POST /predict          200 OK
16:00:17.479 CST POST /predict          200 OK
16:00:17.229 CST POST /predict          400 BAD REQUEST
16:00:17.161 CST POST /predict          400 BAD REQUEST
16:00:16.439 CST POST /predict          400 BAD REQUEST
```

簡單 AI 迴歸

- ▶ 在App script 新增Predict AQI 函數
- ▶ Api Url設定為Ngrok得到的網址
- ▶ 接下來就可以在google sheet 調用Predict AQI函數

```
function PREDICT_AQI(inputArray) {  
  // 扁平化輸入，例如 [[300, 25, 60]] → [300, 25, 60]  
  const features = [].concat.apply([], inputArray);  
  
  // === 驗證輸入是否正確 ===  
  if (  
    !features ||  
    features.length !== 3 ||  
    features.some(val => val === "" || val === null || isNaN(val))  
  ) {  
    return null; // ✖ 不合法 → 不回應  
  }  
  
  const apiUrl = "https://5aa2-140-114-26-118.ngrok-free.app/predict";  
  const payload = JSON.stringify({ features });  
  
  const options = {  
    method: 'POST',  
    contentType: 'application/json',  
    payload: payload,  
    muteHttpExceptions: true  
  };  
  
  try {  
    const response = UrlFetchApp.fetch(apiUrl, options);  
    const result = JSON.parse(response.getContentText());  
  
    if ('prediction' in result) {  
      return result.prediction;  
    } else {  
      return null; // ✖ 有回應但無預測值 → 不回應  
    }  
  } catch (e) {  
    return null; // ✖ 呼叫失敗 → 不回應  
  }  
}
```

網頁顯示結果或通知

- ▶ - 在App Script新增doGet函數
- ▶ 回傳溫度、濕度、MQ135、品質等資料

網頁顯示結果或通知

- ▶ 用html寫網頁
- ▶ 用Web app回傳的溫度、濕度、MQ135、品質等資料顯示結果

```
body>  
<table border="1" id="data-table">  
  <tbody></tbody>  
</table>  
  
<script>  
  const apiUrl = "https://script.google.com/macros/s/AKfycbwTAZYI4AizWgYtORL7X0ba-UlPrX62WW9Gyx1kUotZEhZL0rM1cslJ7VdDdp57";  
  
  fetch(apiUrl)  
    .then(response => response.json())  
    .then(data => {  
      const tbody = document.querySelector("#data-table tbody");  
  
      // 取最後 10 筆 (從陣列尾部反推)  
      const last10 = data.slice(-10);  
  
      last10.forEach(row => {  
        (method) Document.createElement<tr>(tagName: "tr", options?: ElementCreationOptions):  
        HTMLTableRowElement (+2 overloads)  
        const tr = document.createElement("tr");  
  
        ["時間", "溫度", "濕度", "MQ135", "品質"].forEach(key => {  
          const td = document.createElement("td");  
          td.textContent = row[key];  
          tr.appendChild(td);  
        });  
  
        tbody.appendChild(tr);  
      });  
    })  
    .catch(error => {  
      console.error("資料載入失敗:", error);  
    });  
</script>  
</body>  
</html>
```


網頁顯示結果或通知

► 用 Netlify 服務器將網頁發布在網路上

連結：[最近 10 筆環境感測資料](#)

最近 10 筆感測資料

時間	溫度	濕度	MQ135	品質
2025-06-10T10:47:03.747Z	27.9	50	0	37.42
2025-06-10T10:48:08.092Z	27.3	51	0	94.93666666666667
2025-06-10T10:49:10.789Z	27.1	53	0	56.34168181818181
2025-06-10T10:50:13.657Z	27.7	53	0	56.34168181818181
2025-06-10T10:51:17.835Z	27.7	52	0	152.03783333333334
2025-06-10T10:52:26.633Z	27.1	51	0	94.93666666666667
2025-06-10T14:09:35.220Z	25.5	60.2	0	1.8324264708027866
2025-06-10T14:10:03.361Z	25.5	60.2	0	1.8324264708027866
2025-06-10T14:13:58.789Z	31.9	58	0	0.44993846723451986
2025-06-10T14:14:12.009Z	28.3	37	5	13.048833333333333

Demo影片

