智慧垃圾分類系統

組員:B11223222陳力愷、B11223219 鍾永平

日期:2026/01/06

目錄

題目發想與動機

- 核心問題：在日常生活中，民眾/學生常因不確定垃圾類別（如瓶罐是否可回收）而導致分類錯誤。這使得回收人員需耗費大量人力重新分類，增加後續處理的麻煩 。
- 痛點情境：
  - 校園/家庭分類混亂：例如人們貪圖一時的方便隨意亂丟，或者不確定分類規格（如：複合材質容器）而猶豫或誤丟，導致資源回收物被丟入一般垃圾桶。
  - 非接觸需求：傳統垃圾桶需手動開啟，可能存在一些衛生隱憂。
- 解決方案：開發一套結合 Line Bot 介面、雲端 AI 辨識與 IoT 實體致動的智慧垃圾桶。使用者只需拍照上傳，系統便會自動判斷類別並驅動實體垃圾桶開蓋，達成「自動化分類」與「非接觸式衛生」兩大目標。

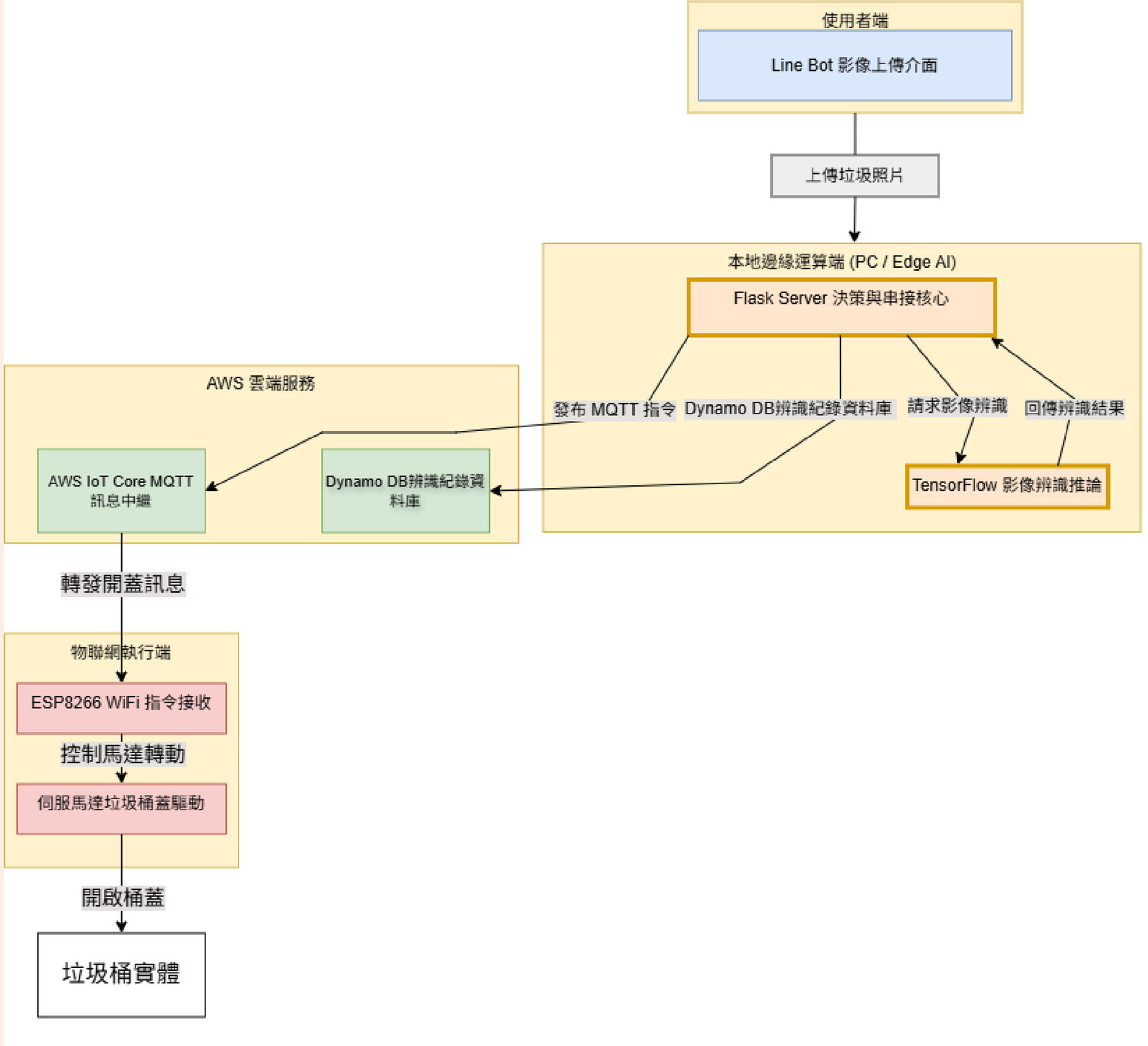
系統架構與流程

本系統採用混合式 AIoT 架構，整合了電腦視覺、物聯網與雲端資料庫 。

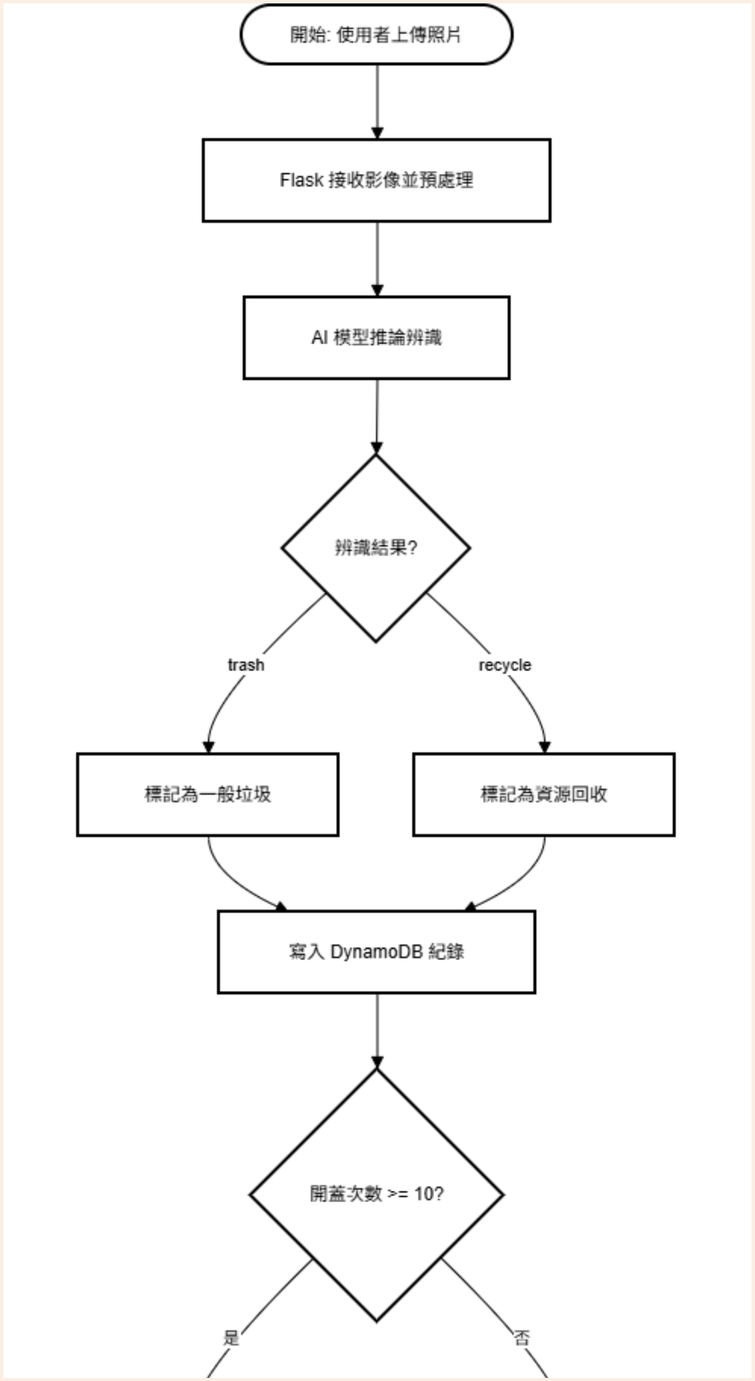
系統運作邏輯與流程

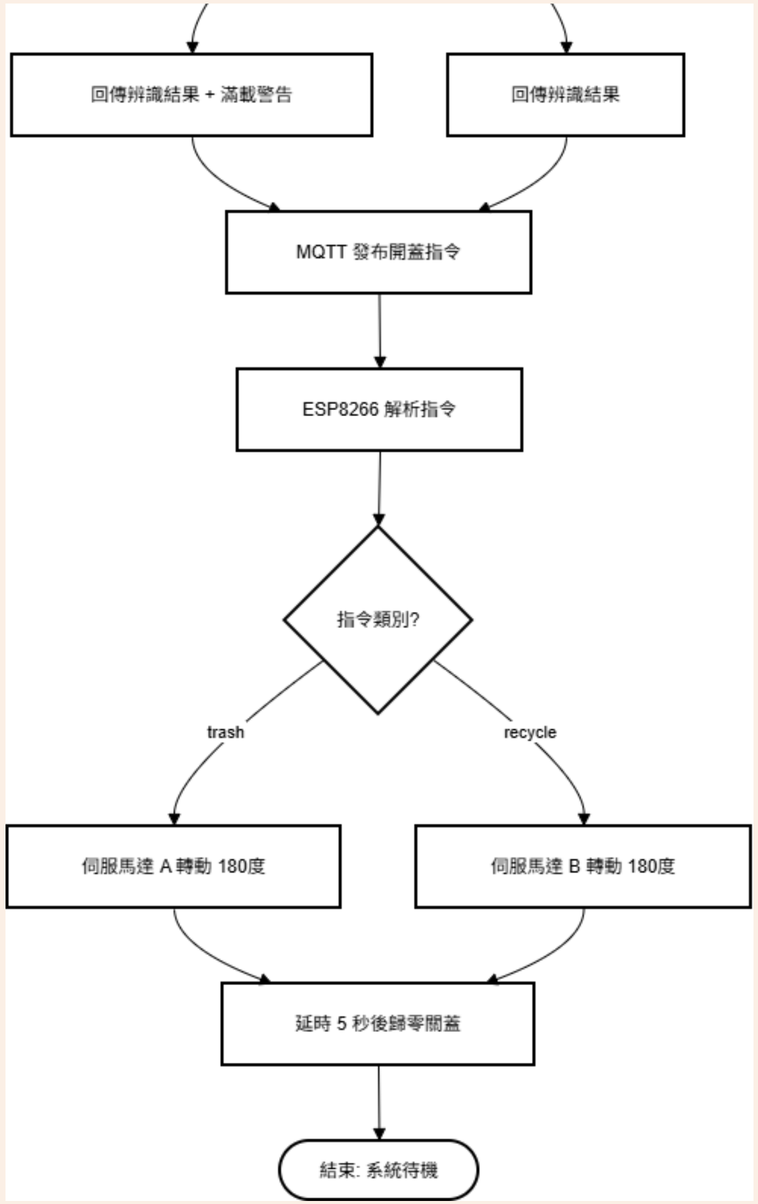
1. 使用者端 (Line Bot)：使用者透過 Line 上傳垃圾照片，Flask 後端接收照片並儲存。
2. AI 運算端 (PC/Flask)：
    - 調用本地 Keras 模型 (.h5) 進行推論。
    - 根據推論結果 (trash 或 recycle) 更新 AWS DynamoDB 雲端資料庫。
    - 透過 MQTT 協定將控制指令發送至 AWS IoT Core。
3. 通訊層 (AWS IoT Core)：擔任 MQTT Broker，負責將訊息傳遞給訂閱了 bin/control 主題的硬體終端。
4. IoT 控制端 (ESP8266)：
    - 解析 JSON 負載中的 target 欄位。
    - 根據辨識結果驅動對應的伺服馬達轉動至 180 度，延時 5 秒後自動關閉。

系統架構與資料流程圖


使用者端
Line Bot 影像上傳介面
上傳垃圾照片
本地邊緣運算端 (PC / Edge AI)
Flask Server 決策與串接核心
發布 MQTT 指令
Dynamo DB辨識紀錄資料庫
請求影像辨識
回傳辨識結果
TensorFlow 影像辨識推論
AWS 雲端服務
AWS IoT Core MQTT 訊息中繼
Dynamo DB辨識紀錄資料庫
轉發開蓋訊息
物聯網執行端
ESP8266 WiFi 指令接收
控制馬達轉動
伺服馬達垃圾桶蓋驅動
開啟桶蓋
垃圾桶實體

系統運作邏輯流程圖


開始: 使用者上傳照片
Flask 接收影像並預處理
AI 模型推論辨識
辨識結果?
trash
recycle
標記為一般垃圾
標記為資源回收
寫入 DynamoDB 紀錄
開蓋次數 >= 10?
是
否


回傳辨識結果 + 滿載警告
回傳辨識結果
MQTT 發布開蓋指令
ESP8266 解析指令
指令類別?
trash
recycle
伺服馬達 A 轉動 180度
伺服馬達 B 轉動 180度
延時 5 秒後歸零關蓋
結束: 系統待機

AI 模型與硬體配置

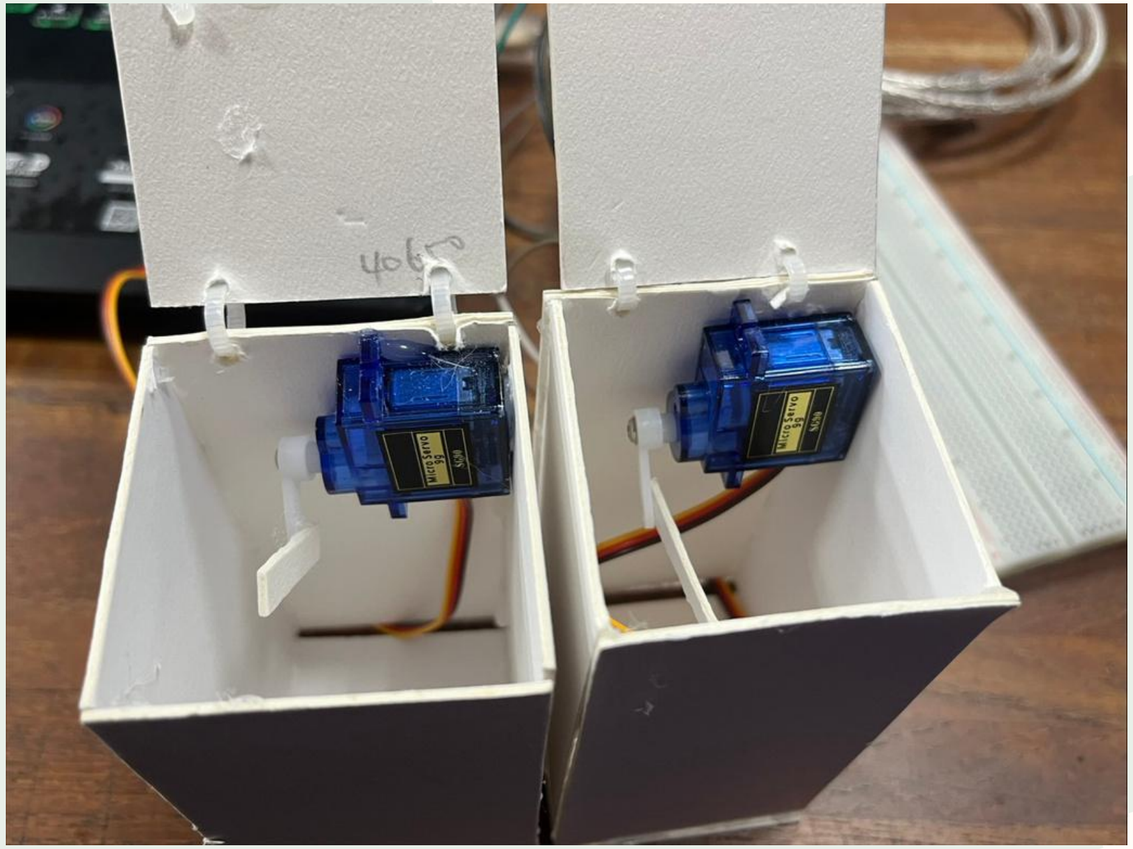
AI 策略

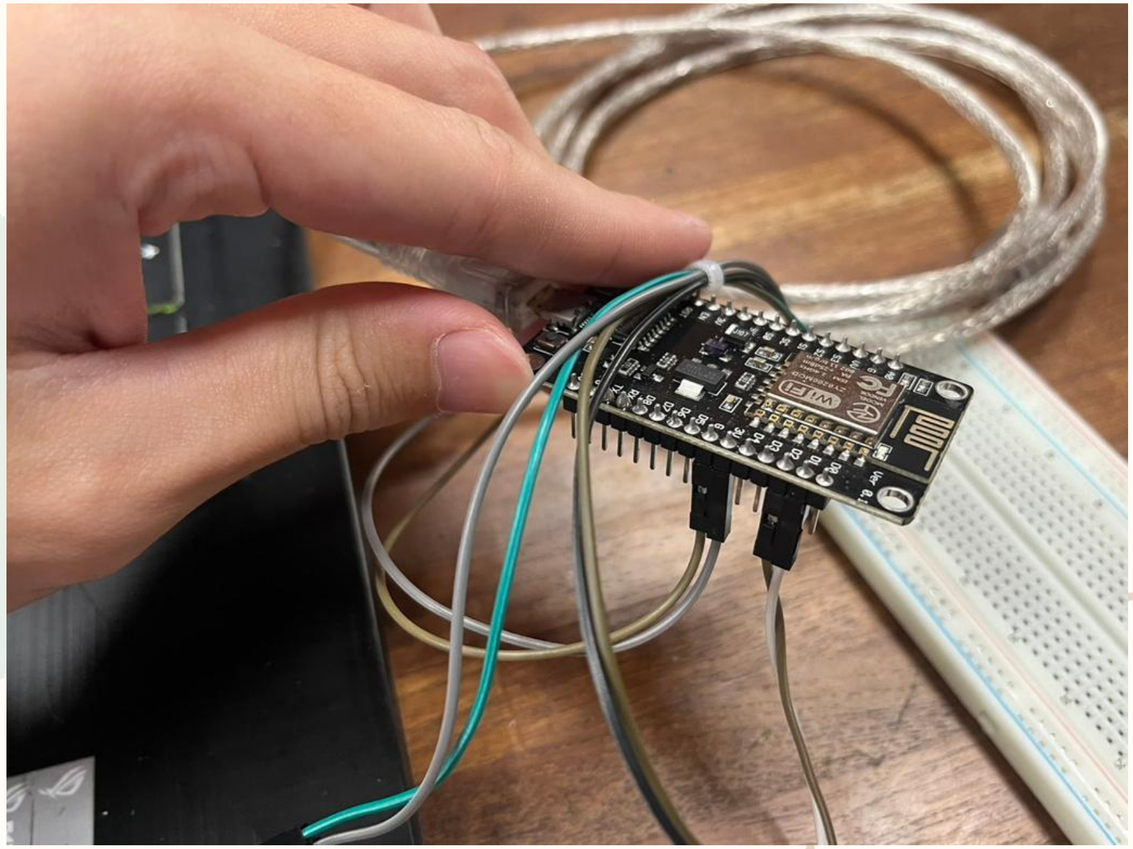
- 模型類型：影像分類模型 (Image Classification)。
- 技術細節：使用 TensorFlow / Keras 架構，透過 Teachable Machine 進行遷移學習訓練，導出 .h5 模型。
- 選擇理由：模型輕量化且能與 Python Flask 高度整合，適合在邊緣端 (PC/Gateway) 進行快速推論。

硬體配置與接線

- 控制器：ESP8266 (NodeMCU)。
- 致動器：
  - Servo A (一般垃圾)：訊號線接 GPIO 5 (D1)。
  - Servo B (資源回收)：訊號線接 GPIO 4 (D2)。
- 接線注意：馬達需接 5V 電源，且必須與 ESP8266 共地 (Common Ground) 以確保訊號穩定。

AI 模型與硬體配置

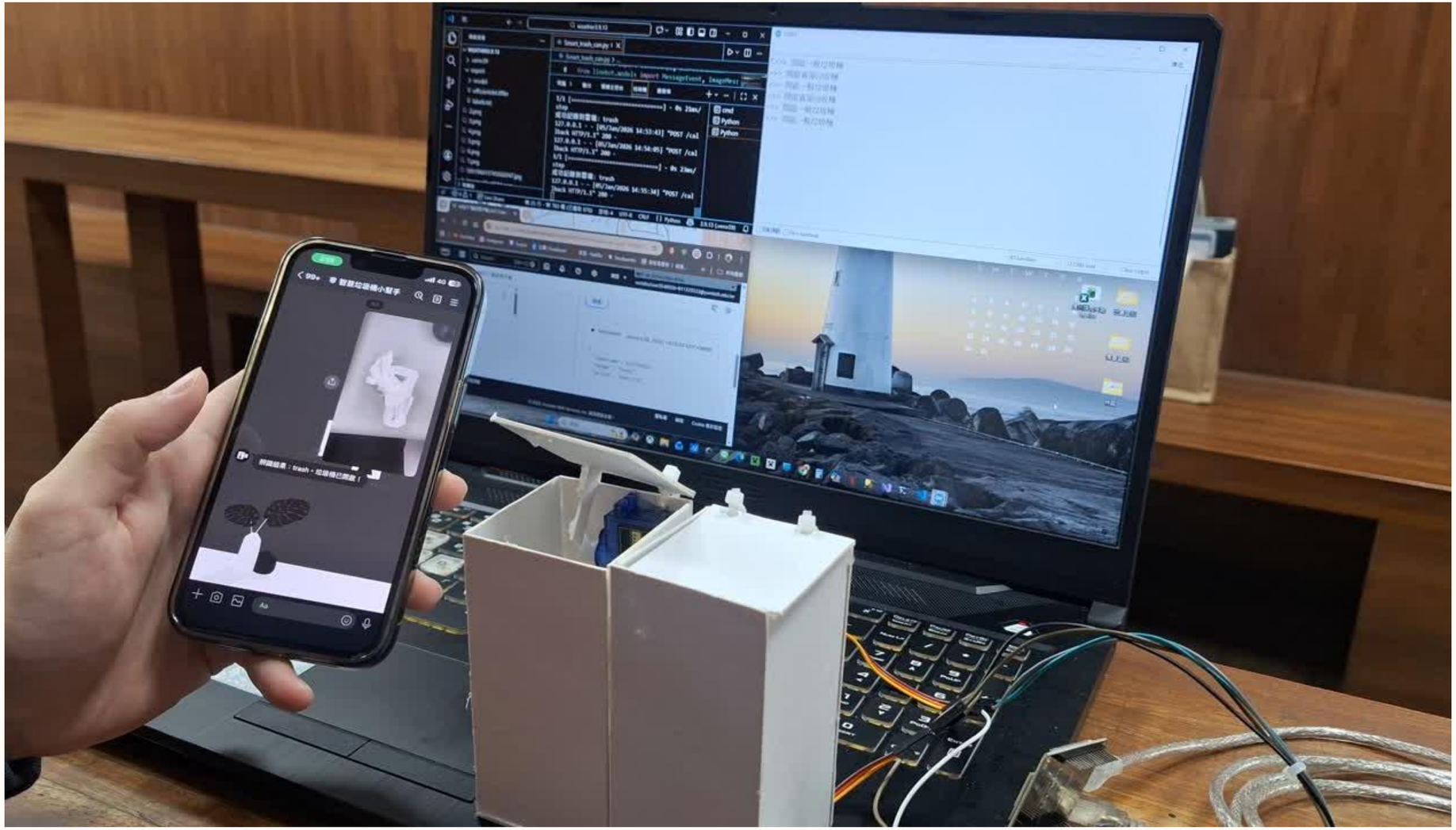




實測展示

- 核心功能表現
  - 即時辨識回饋：Line Bot 會回覆辨識結果（如：「辨識結果：recycle。垃圾桶已開啟！」）。
- 雲端統計與歸零：
  - 輸入「開蓋統計」可查詢目前的累計次數。
  - 輸入「歸零統計」可清除 DynamoDB 歷史紀錄。
- 滿載預警機制：當單一類別累計開蓋達 10 次時，系統會自動在 Line 訊息中發送「⚠ 滿載警告」，提醒清潔人員清理。

展示影片



展示影片

⚠️ 滿載警告：【trash】桶已開蓋 10 次，請通知人員清理！

下午 2:58

問題與解決

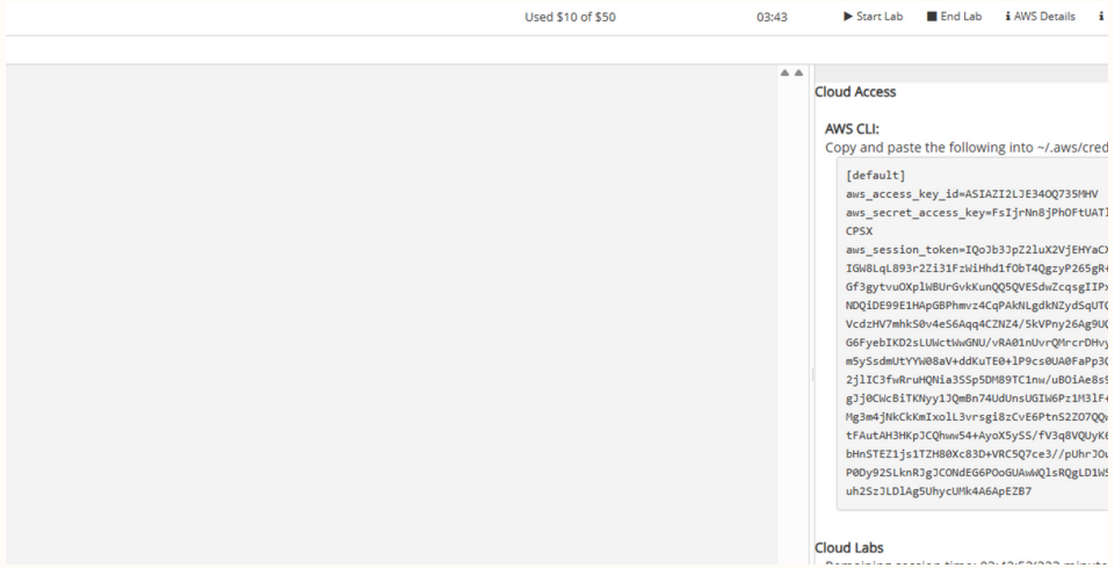
問題:
在硬體端開發時，ESP8266 能夠成功連接 Wi-Fi，但在嘗試連接 AWS MQTT Broker 時持續失敗，序列埠監控視窗顯示錯誤碼 狀態碼=-2(網路層連通但被伺服器拒絕)。

解決方法:
在Arduino程式碼中，改用ESP8266專用的 BearSSL函式庫，並嵌入憑證，將 AWS 下載的Root CA、Device Certificate (.crt) 與 Private Key (.key) 檔案內容，以正確格式嵌入程式碼中，成功通過 AWS 的雙向驗證。

問題與解決

```
t MessagingApiBlob' and 'MessagingApiBlob(...).get_message_content(...)' instead.
or more details.) -- Deprecated since version 3.0.0.
  message_content = line_bot_api.get_message_content(event.message.id)
1/1 [==============================] - 1s 612ms/step
DynamoDB 寫入失敗: Unable to locate credentials
c:\dev\weather3.9.13\Smart_trash_can.py:88: LineBotSdkDeprecatedIn30: Call to dep
agingApi' and 'MessagingApi(...).reply_message(...)' instead. See https://github.
 Deprecated since version 3.0.0.
  line_bot_api.reply_message(
127.0.0.1 - - [30/Dec/2025 11:20:07] "POST /callback HTTP/1.1" 200 -
1/1 [==============================] - 0s 22ms/step
DynamoDB 寫入失敗: Unable to locate credentials
127.0.0.1 - - [30/Dec/2025 11:20:22] "POST /callback HTTP/1.1" 200 -
```

問題:
Python後端依慣例配置ACCESS_KEY與SECRET_KEY後，執行DynamoDB寫入操作時仍持續報「Unable to locate credentials」。



解決方法:
本專案使用的AWS Academy實驗環境採用具時效性的臨時安全性憑證機制。此機制強制要求在標準金鑰外，必須額外包含 SESSION_TOKEN 才能通過驗證。最初使用的標準初始化方式因缺少此 Token，導致身分驗證失敗而被拒絕存取。

分工表

學號	姓名	負責內容
B11223222	陳力愷	AI 模型訓練 AIoT 硬體整合
B11223219	鍾永平	繪製流程圖 簡報製作