物聯網導論期末專題

貨櫃追蹤解決方案

R09631003 鄭子揚
R09631050 詹閔棋

一、動機與目的

近年隨著 Covid-19 肆虐，導致世界各國政府為了圍堵疫情擴散，執行分流上班等措施。而商港碼頭的卸貨人員的人力短缺，在貨物需求沒有因疫情減少的情形下，形成塞港的情形，而這也導致市場供應不足和貨櫃運輸成本上升等後續效應。塞港問題至今因為新型變異株 Omicron 的出現沒有緩解的現象，特別是在遠東地區至美國西岸的航線，在 11 月中期一天至少有接近 90 艘貨櫃船在洛杉磯外海等待卸貨，而這個情形可能在農曆年後才能得到緩解(台灣新生報, 2021)。

本專題的目標希望透過建立一個貨櫃追蹤的物聯網系統，在每個貨櫃集裝箱外側貼上全球衛星定位的模組，並透過低功耗網路的技術將貨櫃的位置資訊上傳至網路，讓貨運駕駛、業者和碼頭調度人員，可以透過網頁監控卸貨區的貨櫃數量及位置，提高駕駛和工人找尋貨櫃的效率，以方便下一艘貨輪入港時有多餘空間進行卸貨以解決塞港現象。

二、系統架構

圖 2.1 是本專題的系統架構圖。將含有 GPS 模組的感測器黏貼至貨櫃上，當 Arduino 在得到 GPS 的經緯度、時間資訊後，會透過 Lora 的方式將含有裝置名稱、日期、時間以及經緯度的資訊傳送給 LoRa Gateway。Gateway 除了扮演 LoRa Server 的功能之外，也同時為 MQTT Broker 的角色，負責將末端節點傳送的資料 publish 給 AWS 的雲端資料庫，在雲端資料庫中我們會建立一個資料的儲存規則，讓資料排列成我們想要的形式並儲存在 AWS Dynamodb 中。在使用者端則可以透過請求的方式，讓所有末端節點最新的位置，也就是最後備讀取的經緯度顯示在網頁地圖中。在第 3 章會詳細解釋我們系統架構中四大部分所使用到的材料與方法，以及程式流程。

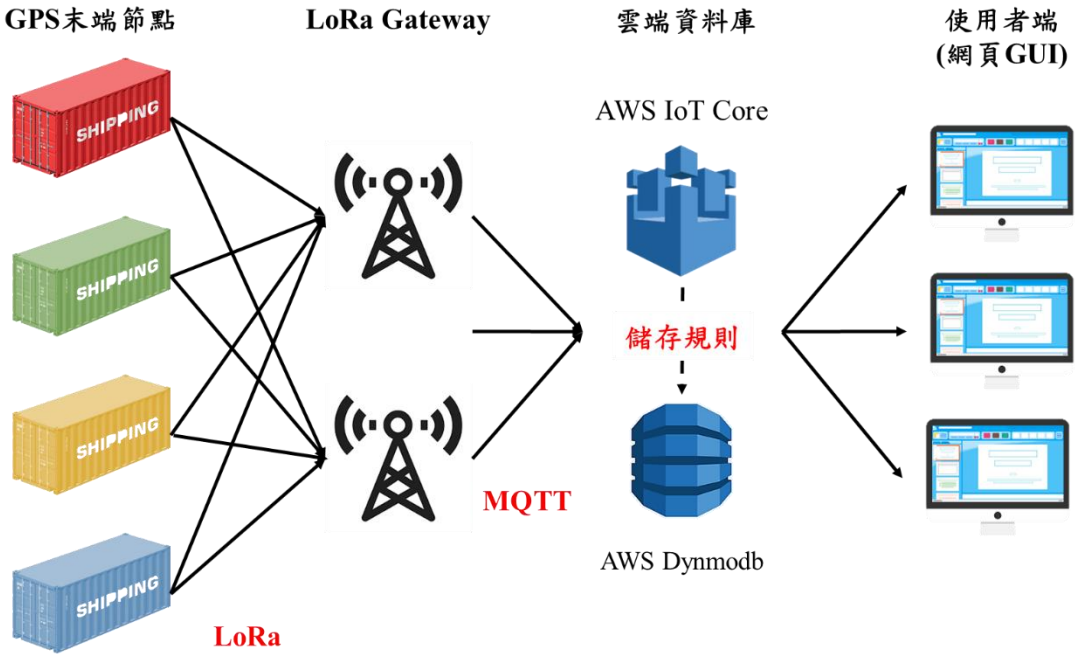

圖 2.1 貨櫃追蹤解決方案系統

三、材料方法

1. GPS 末端節點

在末端節點上我們使用的是 Arduino 作為主要的控制器，定位則採用 neo-8m 的 GPS 模組，通訊則透過低功號廣域網路技術中 LoRa 的通訊方法與 Gateway 傳遞末端節點的裝置資訊，使用的模組是 Dragino sheild 的擴充版。為了方便在戶外進行位置的量測，我們將整組末端節點的裝置放入長 20cm*寬 13cm*高 6.5cm 的盒子中，並在其中一側開口讓 LoRa 和 GPS 模組的天線外露，裝置的示意圖如圖 2.2，表 2.1 是 LoRa 內部參數設定值。在實驗過程中我們採取手持的方式拿著盒子在校園中移動進行經緯度的量測。

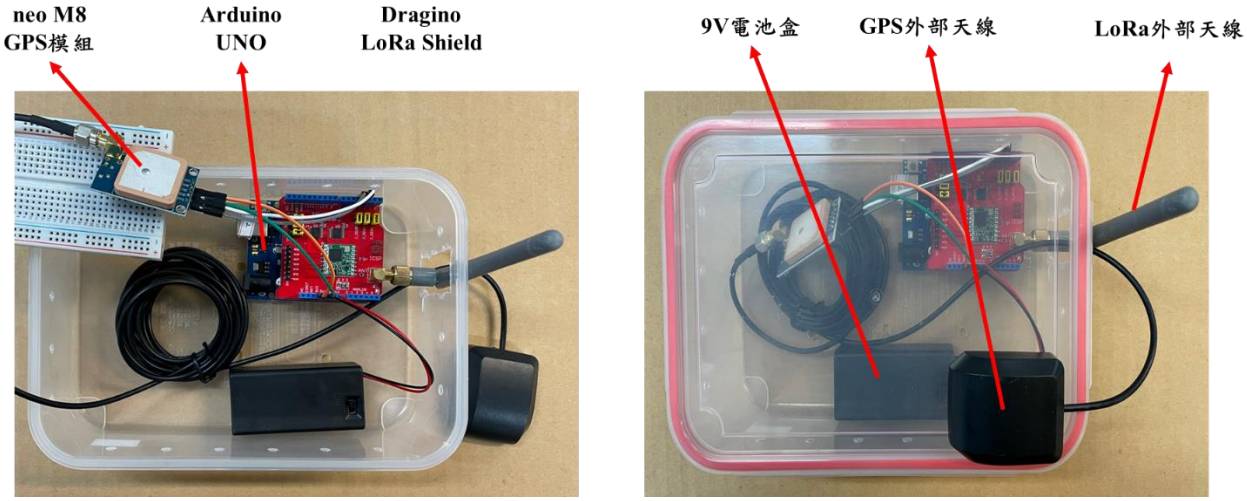

圖 GPS 末端節點硬體使用
左圖為內部構造，右圖是上蓋後

表 2.2 LoRa 內部參數設置

參數	設定值
頻率(MHz)	905.1
頻寬(kHz)	500
Spreading Factor	12
Tx Power(dBm)	23

圖 2.3 是 GPS 末端節點的程式運作流程，程式一開始會等待 GPS 模組抓取固定的衛星，抓取衛星後，會獲得一組度分秒格式的經緯度訊號，經過 TinyGPS 函式庫的轉換後，程式會將末端接點的裝置名稱、經緯度以及時間資訊整理成一組封包，並透過 LoRa 模組將封包傳送給 LoRa Gateway 做接收，封包的內容及格式詳見表 2.2。在封包被傳送後 Arduino 會進入 1 小時省電狀態的深度睡眠模式，待 Arduino 被喚醒後則會重複上述流程。

前面提到 Arduino 會於傳送完封包後，進行一個小時的深度睡眠模式， 其目的是為了要符合末端節點的長時間使用，所以必須使能耗下降。Arduino 所使用的 ATmega328 微控制器共有 4 種睡眠模式分別為 idle, power save, standby 以及 power down。而我們採取了最省電的 power down 模式，在這個模式中除了關閉 MCU 中未使用到的模組外，也會關閉 ADC 以及 BOD，僅剩 Timer 和看門狗計時器，以達到最佳省電狀態。圖 2.4 是我們使用三用電表，對 Arduino UNO(使用 9V 電池供電)所量測電流的結果，而根據結果顯示在深度睡眠模式下的 UNO 板電流消耗下降約 27%。而要喚醒睡眠中的 UNO 板有兩種方式 1.使用外部訊號 2.使用內部 Timer。而在實驗過程中，我們注意到 GPS 模組所連接到 UNO 的 Rx 腳位，會不斷給予外部訊號，導致 UNO 在設定的睡眠模式時間前提早被喚醒，因此在進入深度睡眠模式前，我們設定將 Tx 及 Rx 腳位的訊號讀取先關閉，待進入工作模式前再開啟。

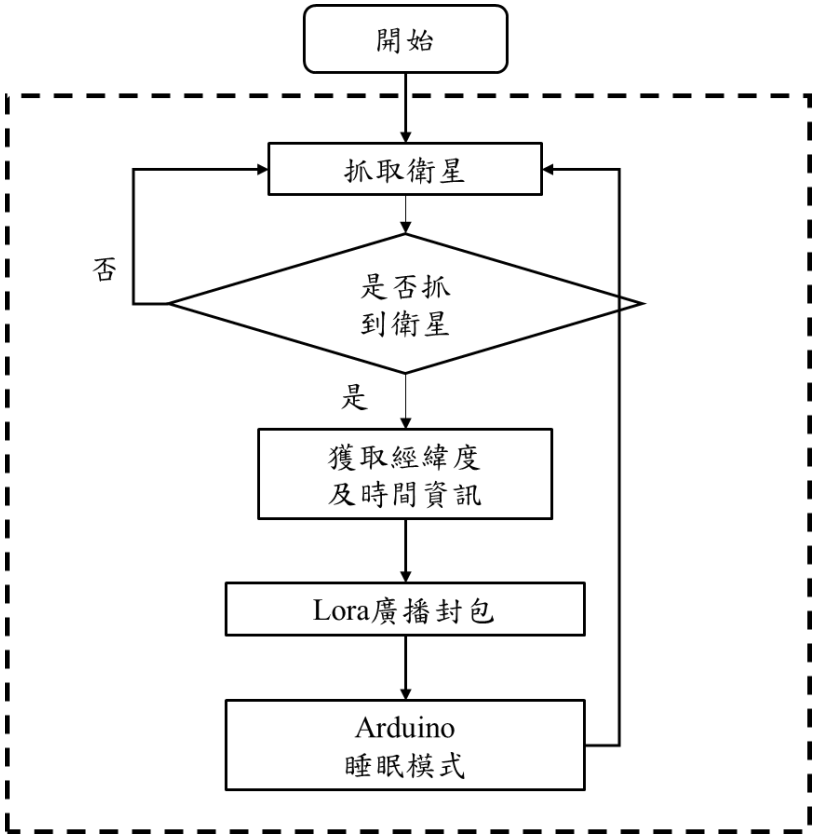

圖 2.3 GPS 節點運作流程

表 2.2 GPS 節點透過 LoRa 廣播資訊

資料名稱	備註
日期	“YYYY/MM/DD”
時間	“hh/mm/ss”. 24 小時制
裝置名稱	e.g.”001”. 可供設置貨櫃編號
緯度	浮點數格式 小數點以下 6 位
經度	浮點數格式 小數點以下 6 位

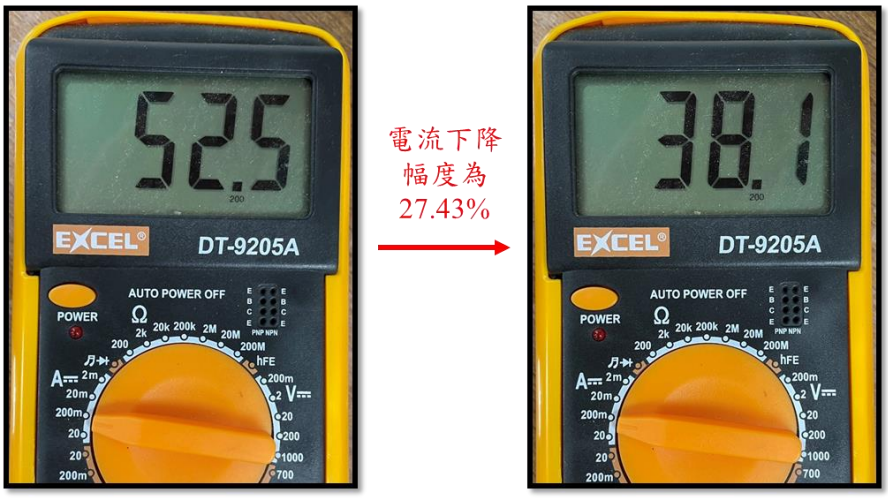

圖 2.4 使用三用電表量測 Arduino 電流

左圖為 Arduino 工作時間量測電流 52.5mA

右圖為 Arduino 睡眠時間量測電流 38.1mA

2. LoRa Gateway

Gateway 負責收集末端節點的資料，並透過 WiFi 上傳到雲端儲存。Gateway 可以分為兩個部份分別是 LoRa Server 和 MQTT Broker。LoRa Server 的硬體使用 Arduino UNO 和 Dragino sheild 擴充板，而 MQTT Broker 的硬體則使用 Raspberry Pi 3b+，兩者之間會透過一條 USB 相接，這個 Gateway 在實驗過程中被放置於台大鄭江樓南棟 404 研究室的窗台上，以獲得最佳的接收訊號品質，硬體與放置圖如圖 2.5。

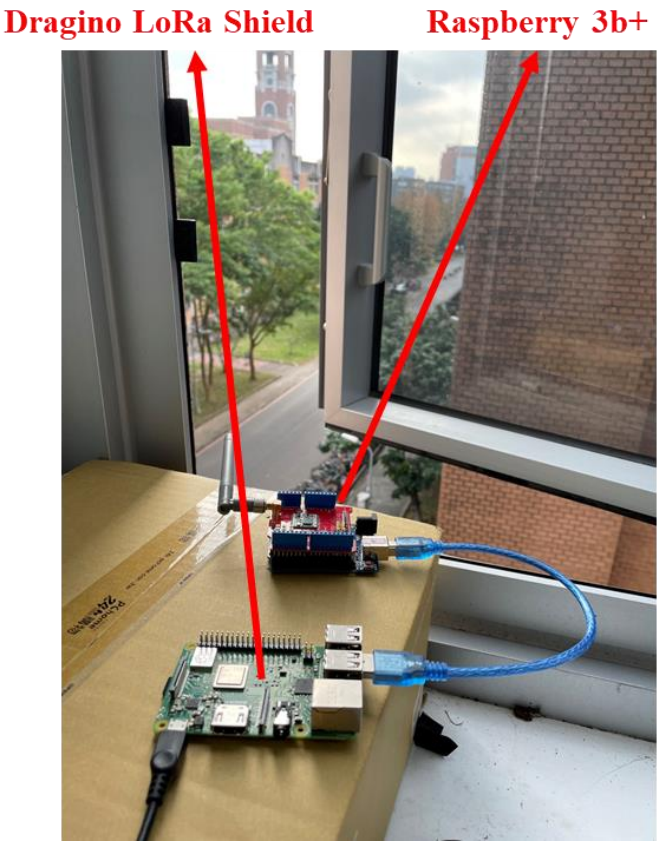

圖 2.5 LoRa Gateway 硬體使用與實驗設置

Gateway 運作流程如圖 2.6，在 LoRa Server 得到末端節點的廣播訊號後，會透過 USB 將資料送到樹莓派，樹莓派中的程式在讀取到 port 中有新資料進入後，會將送進來的資料整理成 JSON 的形式，並透過 MQTT 的方式，將這串 JSON 的訊息 Publish 給 AWS 的雲端。在傳遞的訊息中，裝置名稱和經緯度資料一樣維持字串的方式傳遞，而時間方面則將日期與時分秒做相加，形成 14 位數的整數型態，其用意是上傳至雲端時可以透過時間大小對資料進行排列，詳細的樹莓派 Publish 訊息的格式如表 2.3。

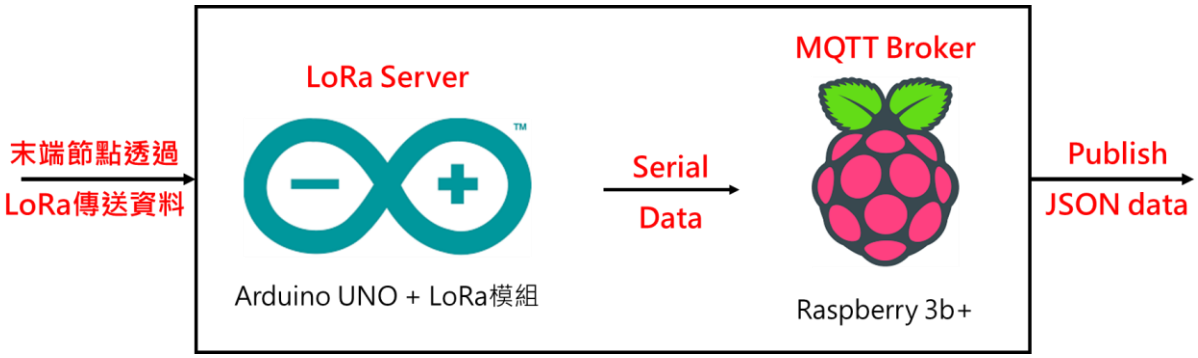

圖 2.6 LoRa Gateway 運作流程

表 2.3 Gateway 訊息上傳雲端格式

資料名稱	資料型態	備註
Device	String	裝置名稱
Time	Int	“2021/12/30” + “20:43:24” →20211230204324
Lat, Lng	String	經緯度資訊

3. 雲端資料庫

將 Gateway 的 Raspberry Pi 連線至 AWS IoT，建立安全的雙向連線，註冊完實物後，下載連線需要的套件放入裝置中，成功建立通訊後，便可以透過 MQTT，將末端節點的資訊從 Raspberry Pi 傳送到雲端空間中。我們使用 Python 來設置、執行指令，設置好 root、驗證金鑰、私鑰的路徑，並修改 clientID 和 topic 名稱，並在 AWS IoT 的政策文件中修改我們實物的資訊。圖 2.7 是 AWS IoT 成功接收 LoRa Gateway 上傳 message 的畫面。

圖 2.7 AWS 雲端接收 Gateway message 畫面

為了將實物的座標資訊即時呈現在地圖上，製作清楚簡潔的地圖供使用者使用，我們要將裝置儲存在雲端的資料擷取下來，所以我們使用 AWS IoT 中的規則引擎來整合資料庫的服務。我們使用的是 Dynamodb，先根據我們的需求建立資料表，Dynamodb 這個資料庫中會將資料儲存成表格的形式，所以在建立時最重要的就是先設置好 Partition Key 以及 Sort Key。Partition Key 是用來對接收到的資料進行分類的，Partition Key 相同的資料會被儲存在一起，可以用來對資料做分類；Sort Key 則是用來決定資料的儲存順序，可以針對相同 Partition Key 的資料進行排序。

在本專題中，我們設置的 Partition Key 為 Device_Name，我們設計的 Server 可以同時接收多台裝置回傳的 GPS 資訊，所以這個 Partition Key 能夠讓我們在分析資料時更加方便；而我們設定的 Sort key 是 Time，就是 GPS 感測器收到資訊的時間，格式為 YYYYMMDDhhmmss(eg. 2021/12/30 20:36:55

會顯示為 20211230203655)，將收到的資訊案時間排序有利我們進行前端渲染的作業。圖 2.8 是設定資料表排序規則的畫面。

圖 2.8 Dynamodb 設定資料庫儲存規則

因為 AWS 學生方案帳號的限制，所以只能設定一個 Sort key，導致我們在整理資料庫資料的過程變得比較複雜。最重要的經緯度資訊，我們透過 Message 欄位來回傳，緯度和經度的精度可以達到小數點後面 6 位，圖 2.9 是我們資料表儲存末端節點上傳的資訊。

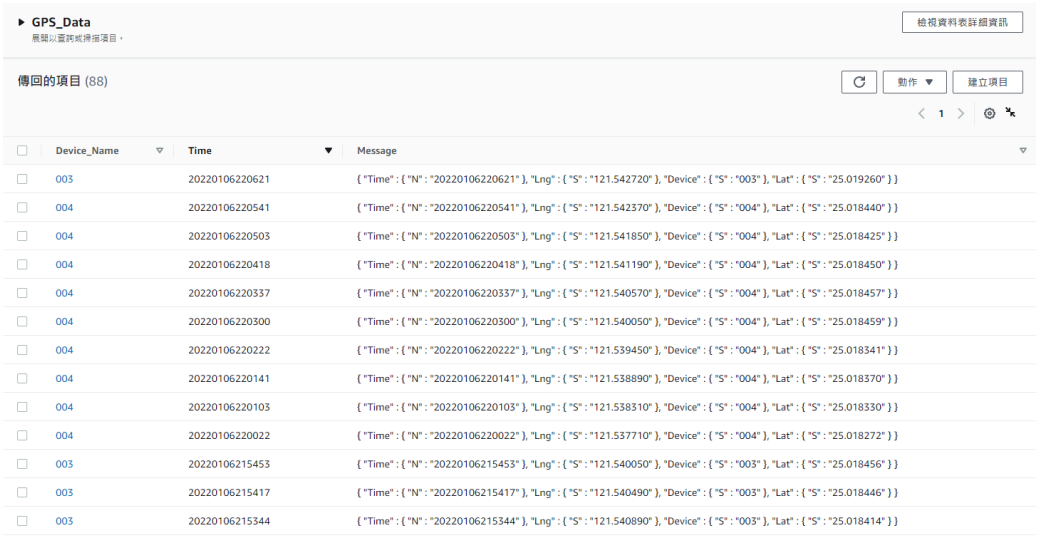

圖 2.9 儲存於 Dynamodb 的末端節點資料

4. 使用者端(網頁 GUI)

使用者端負責將儲存在 AWS Dynamodb 的資料做讀取並將資料以視覺化的方式呈現給使用者查閱。使用主端的撰寫主要透過函式庫 flask 完成，它是可以透過 Python 撰寫的輕量級網路框架。而網頁資料的獲取則是透過 boto3 函式庫，透過給予程式 Dynamodb 的 API Key 之後，我們就可以透過 Python 對雲端資料庫的內容進行讀取或者修改，而我們使用的方式是每次讀取都掃描(scan)一整個資料的資料並擷取至 local 端，並以 list 的形式暫存。在資料呈現方面，為了能夠讓使用者清楚了解貨櫃的實際位置，必須將讀取至 local 端的貨櫃經緯度在地圖中呈現，這邊我們使用到的是 OpenStreetMap 這項服務，它是由 Steve Coast 所提出的一套開放協做的地圖計畫，讓用戶可以在地圖中自由地加入

新內容。使用者端架構圖如圖 2.10。

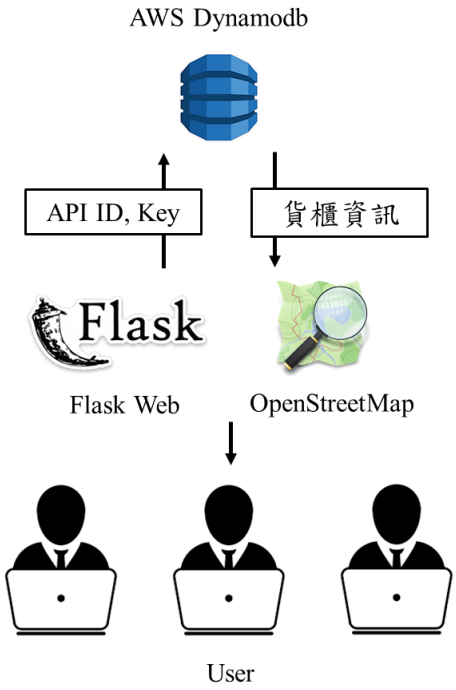

圖 2.10 使用者端架構

我們的目的是要讓貨運司機可以在讀取網頁時可以看到貨櫃最新的位置，因此對於從資料庫中暫存的資料，我們的處理方式是找到每個不同裝置在最新的時間的經緯度，並輸入到地圖上，並以 marker 的方式呈現。在地圖中使用者可以透過點選地圖上的 marker，會顯示這個貨櫃的詳細資訊，如貨櫃的編號及其在這個位置的日期以及時間，詳細的畫面見圖 2.11。

圖 2.11 可顯示貨櫃位置的地圖 GUI

四、核心函式(code)

1. GPS 末端節點

//末端節點工作階段程式碼(僅上部分程式)

//程式等待 GPS 抓取固定衛星

```
if(sleepFlag == false){
  ss.begin(GPSBaud);
  //powerSet(0x00);
  //delay(1000);
  findFlag = false;
  while (findFlag == false && sleepFlag == false && ss.available() > 0){
    gps.encode(ss.read());
    if (gps.location.isUpdated()){
      if(gps.satellites.value() > 0)
        findFlag = true;
      else
        findFlag = false;
    }

    if (findFlag == true){
```

//程式透過 LoRa 廣播封包

```
      String total_countString = dateCur + " " + timeCur + " " + device + " " + String(latCur,digit)+ " " + String(lngCur,digit);
      for (int i=0 ; i< 48; i++){
          data[i] = total_countString.charAt(i);
      }

      rf95.send(data, sizeof(data));
      rf95.waitPacketSent();
      uint8_t buf[RH_RF95_MAX_MESSAGE_LEN];
      uint8_t len = sizeof(buf);

      if (rf95.waitAvailableTimeout(3000)){
        if (rf95.recv(buf, &len)){
            Serial.print("got reply: ");
            Serial.println((char*)buf);

            String lineOut= (char*)buf;
            lineOut= lineOut+ "                          ";
            String Line0  = lineOut.substring(0,15);
            String Line1  = "Rx:"+lineOut.substring(19,42);
            int rssiRev = rf95.lastRssi();

        }
        else{
```

//GPS 節點深度睡眠程式

```
void deepSleepMode(int sleepTimes){
  LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
  if(sleepTimes == totalSleepTimes){
    Serial.println("Arduino wakes up");
  }
}
```

2. LoRa Gateway

//整理從 LoRa Server 送來的資料

```
    if len(strList) == 6:
        device = strList[2]
        time = int(str(strList[0]) + str(strList[1]))
        lat = str(strList[3])
        lng = str(strList[4])


        publish(device, time, lat, lng)
```

//MQTT Publish 實際函式，Message 以 JSON 格式傳送

```
def publish(device, time, lat, lng):
    if mode == 'both' or mode == 'publish':
        message = {}
        message['Device'] = device
        message['Time'] = time
        message['Lat'] = lat
        message['Lng'] = lng
        messageJson = json.dumps(message)
        myAWSIoTMQTTClient.publish(topic, messageJson, 1)
        if mode == 'publish':
            print('Published topic %s: %s\n' % (topic, messageJson))
```

3. 使用者端(網頁 GUI)

//使用 boto3 讀取 Dynamodb 上的資料，並讀取每個 device 最新的資料

```
class readDB:
    # 建構式
    def __init__(self):
        self.data = []
        self.dict = set()

    def current(self, raw):
        raw.reverse()

        for r in raw:
            if r["Device_Name"] not in self.dict:
                self.dict.add(r["Device_Name"])
                self.data.append(r)

            else:
                continue

        return self.data
```

//將 Dynamodb 上的資料(貨櫃集裝箱經緯度)畫在 OpenStreetMap 上，並呈現在 local 端網頁上

```
def map_marker():
    # this map using stamen terrain
    # we add some marker here

    fmap = folium.Map(
         location=[data[-1]["Message"]["Lat"], data[-1]["Message"]["Lng"]],
         tiles='Stamen Terrain',
         zoom_start=15
    )
    c = 0
    for d in newData:
        t = str(d["Time"])
        date = t[0:4] + "/" + t[4:6] + "/" + t[6:8]
        time = t[8:10] + ":" + t[10:12] + ":" + t[12:14]
        folium.Marker(
            location=[d["Message"]["Lat"], d["Message"]["Lng"]],
            #popup="{} \n {}".format(date, payload[3]),
            popup="<b>" + "<center>" + "Name: " + d["Device_Name"] + "<br>" + date + "<br>" + "<center>" + time + "</b>"
            icon = folium.Icon(icon = "cube", color = colorList[c], prefix='fa')

        ).add_to(fmap)
        c+=1
```

五、結果 - DEMO

圖 5.1 是使用 Python flask 讀取 AWS 資料庫的資料，並使用第四節使用者端的程式碼中 current 這個函式，可以讀出每個 device 在資料庫中最新的資料結果。

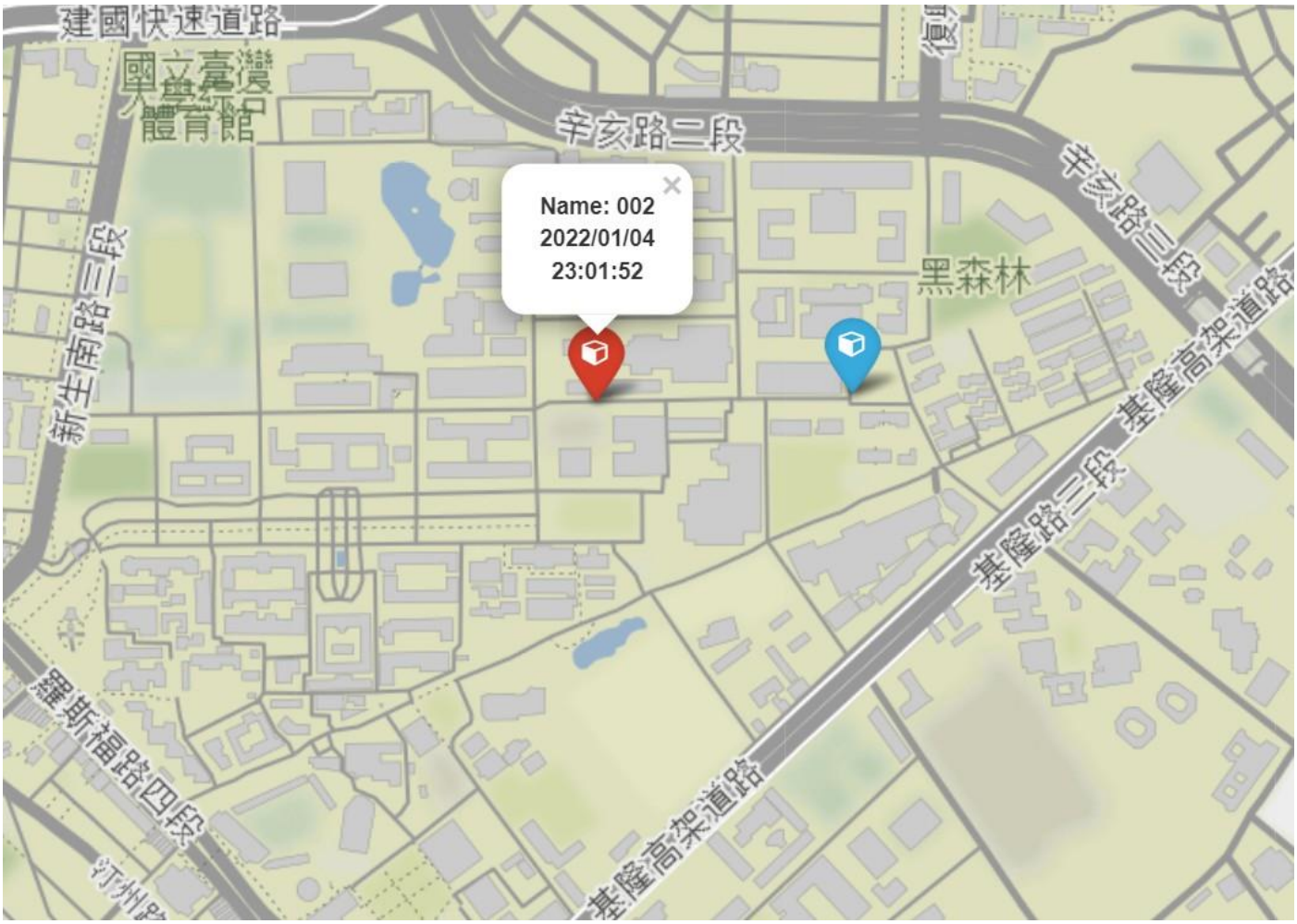

圖 5.1 裝置編號 001, 002 號在地圖上最新位置

我們也針對 LoRa 廣播距離做了一個實驗，實驗方法是將 LoRa Gateway 架設在台大鄭江樓南棟 404 研究是窗台上，而我們兩位同學一人各持一台 GPS 末端節點裝置在校園內行走，測試 LoRa 封包可以在多遠的距離傳輸，以及面對多個節點時能否正確回傳資訊。圖 5.2 是 google map 畫面的截圖，上方紅色路徑為裝置編號 003 的軌跡，藍色則為編號 004 的軌跡。

圖 5.2 實驗路線軌跡圖(google map 畫面截圖)

圖 5.3 與圖 5.4 分別是裝置編號 003 及 004 回傳經緯度並在我們網頁 GUI 呈現的結果。根據結果顯示，量測到最遠的距離為台大小福樓，這邊使用 google map 距離量測的功能，量測出的距離大約為 500 公尺。但是在其他相對距離較短的範圍內，LoRa Server 卻沒有接收到回傳的封包，推測原因可能為校園中的建築物、樹木等障礙物阻擋，導致訊號無法維持在視距(Line Of Sight)內傳播，因此無法達到 LoRa 理想中可以在 10~20km 的傳播距離。

圖 5.3 編號 003 回傳經緯度並於網頁 GUI 呈現

圖 5.4 編號 004 回傳經緯度並於網頁 GUI 呈現

六、結論與未來展望

在學期中剛要選擇期末專題的題目時，看到老師講解各式各樣不同領域的題目，感覺都是十分有發展性和技術成分的題目，和組員一時無法決定我們要選擇哪個題目進行研究。而因為先前實驗室的學長有做過針對氣調貨櫃中環境監控的物聯網感測裝置，因此我們認為若選擇貨櫃追蹤解決方案這個題目，如果成功做出貨櫃追蹤的系統，之後就可以將這套解決方案與先前學長的研究結果做結合，讓整個系統架構與功能更完整，可用性更高。所以我們先針對題目進行分析，擬定了之後的研究計劃和設計了大略的系統架構，接著就趕緊連絡了英業達股份有限公司的吳博祥處長，與他進行更深入的討論，確定了專題要求與技術細節的部份之後，便開始著手研究貨櫃追蹤解決方案。

整個系統的設計包含將 GPS 感測器做成 IoT 模組，透過 LoRa Gateway 傳輸資訊，Server 收到訊號後再傳輸到 AWS 雲端儲存空間。使用資料庫將 AWS 雲端資料按照所期望的格式新增到資料表，還有如何在本地端取得資料庫中所需要的資訊，最後在前端渲染，製作網頁 GUI，將座標在地圖上做可視化的呈現。這個專題可以說是用到了整學期物聯網課程中的所有內容，透過這個專題的實作，我們對物聯網相關領域的技術也有更深入的了解，自己也擁有了實作的經驗。當然在專題的製作過程中，我們也遇到了許多大大小小的問題。像是在 GPS 模組的選擇，在實際操作時常常遇到明明這次可以成功接收到定位訊號，過了一會兒再試一次就完全失靈；也因為硬體和實驗場域的限制，訊號最遠的傳遞距離也不如原先預想的那麼遠；還有關於 AWS 雲端資源和資料庫的使用，因為我們使用的是學生的帳戶，導致在操作的自由度上也受到了許多侷限，例如在 Sort key 的設定限制只能新增一個。而我們也盡量想辦法去解決遇到的問題，並嘗試許多替代方案，在物聯網的實作中，我們認為不斷的實驗以及收集數據是很重要的過程，實驗所得數據能夠直接反應出目前系統執行的效率，和結果是否符合預期，進而去修正其中的流程，才能使系統整體有更佳的表現。

最終我們也成功使用課堂所提供的器材設備，加上所學技術製作出了簡易的貨櫃追蹤系統。將定位感測裝置裝設在貨櫃上，Server 端可以同時接收多個貨櫃的經緯座標，並呈現在網頁地圖上，可以提供給使用者及時的貨櫃座標資訊還有完整的移動路徑，但也有些缺點仍未解決，像是 GPS 模組耗電問題，還有因為 LoRa 模組訊號在傳遞時，易受到環境障礙物影響，導致覆蓋面積過小，雲端功能和前端地圖呈現的部分也受限於付費機制。

而關於此專題，我們也規劃了未來還能夠再深入研究的方向。首先就是供電方面的問題，因定位

裝置在貨櫃中沒有額外電源供應，勢必得使用電池來供電，而貨櫃在海上的時間上則達到好幾個禮拜甚至幾個月，因此若能將此模組的功耗降得越低越好，解決的方向可以朝向更換更合適的 MCU、定位感測模組，以提升 GPS 末端節點的使用壽命；再來就是模組外殼的設計，此時專題因研究時間不長的因素，並未實際設計出外殼，未來可以根據此模組裝設在貨櫃上的方式，天線、電池擺放位置，去設計一個擁有小體積、輕便、防水等功能的合適外殼；最後就是可以再加強 GUI 提供給使用者的功能，呈現更完整的資訊，甚至提供 APP 讓使用者在戶外也能隨時追蹤貨櫃位置資訊。

參考資料

https://tw.news.yahoo.com/%E7%BE%8E%E8%A5%BF%E5%A1%9E%E6%B8%AF%E6%9C%AA%E7%B7%A9-%E6%81%90%E6%8C%81%E7%BA%8C%E8%87%B3%E6%98%8E%E5%B9%B42%E6%9C%88-121514793.html

https://randomnerdtutorials.com/guide-to-neo-6m-gps-module-with-arduino

https://diyi0t.com/arduino-reduce-power-consumption

https://medium.com/@NotSoCoolCoder/handling-json-data-for-dynamodb-using-python-6bbd19ee884e

https://docs.aws.amazon.com/zh_tw/amazondynamodb/latest/developerguide/GettingStarted.Python.04.html

https://blog.yeshuanova.com/2017/10/python-visulization-folium/