Introduction to IoT Final Project

ToF 山屋水情偵測

作者：B08901097 徐有齊

B08901132 任瑨洋

業師：廖書漢 博士

夏維良 經理

1. 概要

本專案旨在使用 LoRa 技術開發一套便宜、低功耗且可靠之物聯網通信網絡，在缺乏網路設施之偏遠地區提供各種物聯網感測設備回傳資料，並即時將這些資料呈現在網際網路。作為示範，我們使用一 ToF 感測器量測山屋中儲水池之水位，回傳至伺服器後，即時於前端展示給使用者參考。

1. 背景與發想

業師提供了我們一組體積小且功耗低的 ToF 單點測距模組，此模組特色為其可以測量透明物體，例如水位面。業師並提供我們「水位監測」這個概念作為發想，例如測量城市的下水道水位等。

我們以業師提供的想法進行延伸，以組員個人的登山經驗和觀察，決定以「山屋水情監測」作為題目，並結合課程所教授的 LoRa 技術，解決山區因通訊不良而導致登山客難以事先掌握環境資訊的窘境。由圖(一)中可以看到，國家公園內山屋水情回報日期參差不一，有些山屋的資料常常相當過時，因此我們希望設計一個裝置即時回報水情，給登山客更有參考價值的資訊。此外此類裝置還能提供求救功能，在便利之餘更能夠守護登山客的生命安全。



圖(一)：玉山國家公園管理處於 111/03/14 發布的水情資訊 [1]，當下其中有幾筆資料的更新時間已經是一個月以上。下次發布的時間已經是 111/12/08 [2]，中間長達近 10 個月的時間皆沒有更新資料。

1. 價值創造與分析

我們的解決方案有以下欲達成之目標：

1. 我們希望可以幾乎即時監測山屋中的儲水槽水位。
2. 水位資料將儲存於雲端。
3. 使用者可以輕鬆的查詢儲水槽水位。

我們的解決方案預計客戶為登山者與國家公園管理單位，向這些需要山屋水位資料的單位提供資料。

我們在市場上目前沒有明顯的競爭者，唯一可以討論的是會友熱心山友向國家公園管理處回報山屋水位。我們的解決方案比起熱心山友的被動回報，只需要付出極小的成本，便可以達成即時監測的目標，在利益權衡方面相當有優勢。

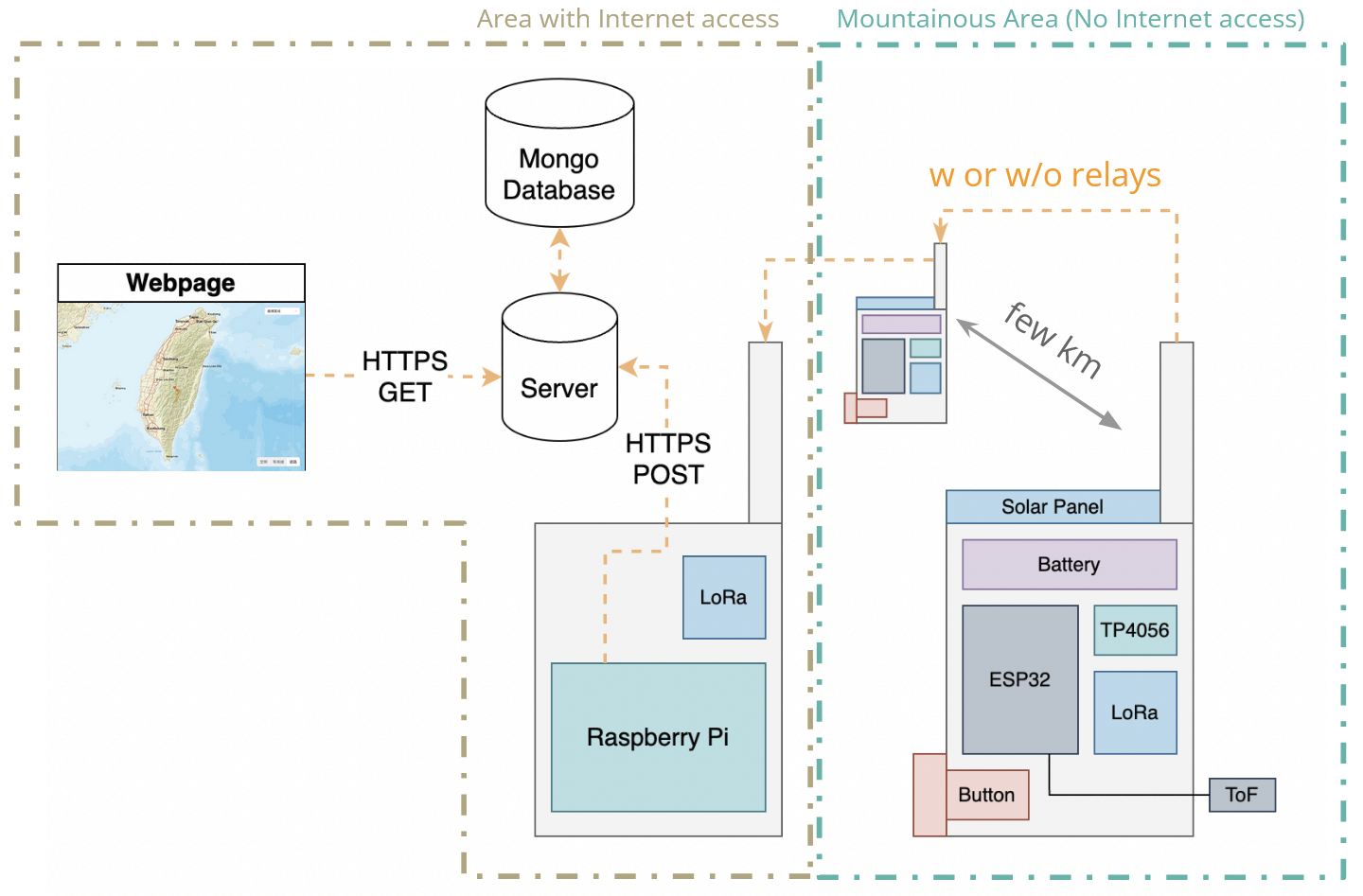
以下我們就兩個方面進行價值分析：

1. 就經濟產值方面，根據中華民國健行登山會的統計 [1]，臺灣每年頻繁登山的有約 3 萬到 5 萬人次，曾經到過 3000 公尺以上的登山客更達 50 萬人次。由此可以看出，登山在臺灣風氣盛行，此解決方案有許多潛在客戶。
2. 就登山客救援方面，我們的解決方案可以在山區中提供登山客手機信號、GPS 信號及無線電以外的通訊方案，可以在上述方案失效時試圖與登山客取得聯繫，提高遇到緊急狀況的人的生還機率。
3. 解決方案
4. 總覽

圖(二)為我們系統的總覽，主要分成兩個部分：山區中沒有網路通信的部分，以及山腳下至使用者端等有網路通訊的部分。

在山區中，我們將設計一個整合的模組，此模組可以測量山屋水位後以 LoRa 回傳。若是一發 LoRa 訊號距離不足，可以透過多個整合模組接力傳送。

山腳下有網路的地方，我們計劃放置一個 Gateway 模組，負責在接收 LoRa 訊號並處理後，以 HTTPS POST 的形式發至伺服器。伺服器接收資料後，會將資料送入資料庫儲存，並提供使用者前端頁面查詢水情資料。



圖(二)：山屋水情監測解決方案系統總覽。以 [draw.io](https://app.diagrams.net/) 協助繪製。

1. 硬體端與韌體端

以下介紹我們的解決方案中使用到的硬體內容。

* MCU - ESP32

我們在微控制器的部分採用 ESP32，除了其成本低廉而計算效能強大外，其還有提供 deep sleep mode 功能，可以在不需要傳送資料的空窗期進入睡眠節電，功耗只需要 10 uA [4]，非常省電。在這個專案中由於演示需求，每 10-20 秒更新一次水情資料，實務層面則只需要一小時以上，甚至到一天才更新一次資料即可。資料將會以 在讀取後，使用 ArduinoJson 套件 [5] 協助將資料變成 JSON 形式，再生成字串等待傳送。

* LoRa - SX1278

LoRa 方面我們使用 SX1278 模組，並使用 Sandeep Mistry 所撰寫的 arduino-LoRa 套件 [6] 進行驅動。注意到 LoRa 適合遠距離的小規模少資料通訊，在郊區的直線通訊距離可以達到 15 公里以上 [7]，非常適合我們的專案。在這個專案中，我們使用亞洲地區不須登記的 433 MHz 頻段，傳輸功率 17 dB，展頻因子為 7，訊號頻寬為 125 kHz。

* ToF - FSTOF2002C0D

此為虹晶科技所提供的 ToF 模組，可以透過 UART 或是 I2C 協定進行溝通，偵測距離在 10 cm 到 240 cm，且可以偵測水面高度 [8]。在這個專案中我們使用 UART 進行溝通，設定上為鮑率 9600 與8N1。

* Infrared - 2Y0A02

此為我們準備的備案方案，當 ToF 模組失效時可以透過該模組偵測水面高度，惟其耗電量較高且準度較差。

* Big Red Button - LA38

我們在整合模組上放置了一個大紅色按鈕，按下此按鈕會觸發一個 Rising Edge 給 MCU，將透過 LoRa 傳送一求救信號。

* Battery - LiPo 18650

我們使用 18650 鋰電池作為我們模組的電源，其標準參考電壓為 3.7 V，滿電約為 4.1 V，容量約為 2200 mAh。

* Charger - TP4056

由於我們希望此解決方案不需要頻繁照應，因此電池應有一充電方案，此為鋰電池充電模組，負責將其他電源降壓後充給鋰電池。

* Solar Panel

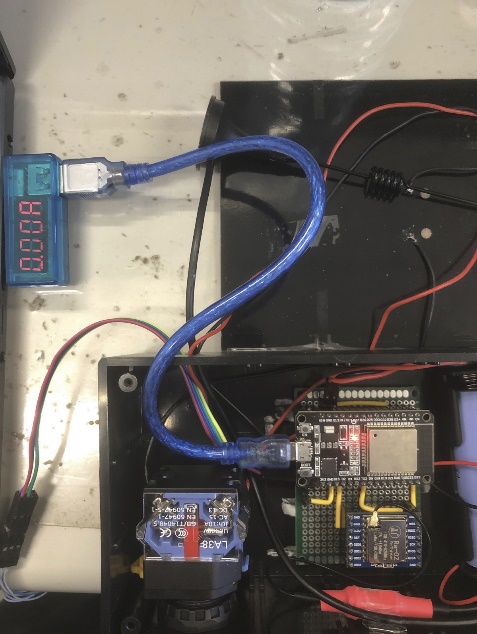
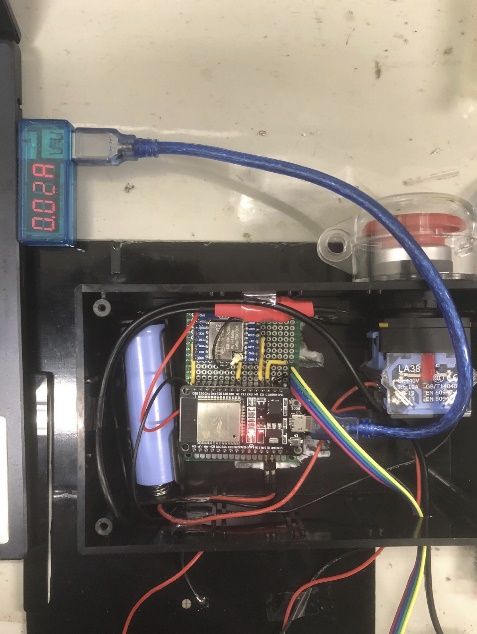
我們的充電方案使用兩片 2.5 W 的太陽能板供電，透過充電模組項鋰電池充電。

* RPi Zero W

原先我們打算也使用 ESP32 作為終端 Gateway 的 MCU，但後來發現使用 ESP32 發 HTTPS 協定會有一些設定上的困難，因此改用 RPi Zero W 加上 LoRa 模組 SX1278 作為終端回傳裝置。我們使用 PyLoRa 模組，並以 Python 進行開發，最終的裝置體積非常小巧，且使用非常容易，只要接上電源就會自動連上 WiFi 並接收 LoRa 訊息回傳伺服器。資料以 JSON 格式傳送。

此外，我們也有對我們的解決方案進行功耗的計算。偵測水位並傳輸資料時，MCU 採用 modem sleep mode，功耗約為 3 mA (我們有降 CPU 頻以更節省電量) [4]，ToF 模組的耗電量約為 10 mA [8]，LoRa 模組 SX1278 在發送訊息時的耗電量約為 93 mA [9]。

實務上測量電流時，由於工作時間實在太短，基本上都測到睡眠時的電流量，整個模組在使用 ToF 模組偵測時，儀器量不到電流，只能粗估耗電量約為 < 10 mA，如圖(三)─左所示。使用備用 Infrared 模組偵測時耗電量約為 20 mA，如圖(三)─右所示。

圖(三)：左圖為使用 ToF 模組時的耗電實測，右圖則為備用模組的。

實測以 20s 回報一次的頻率擺放 12 小時候 (遮住太陽能板不充電)，電池電壓由 4.036 V 掉至 4.002 V，由此估計由 4.1 V (滿電) 掉到 3.7 V (約一半電) 可使用 130 小時。

1. 軟體端

本專案的軟體部分係指監測設備終端回傳之伺服器，以及使用者瀏覽相關資訊的網頁。軟體部分之架構為經典的 MERN stack [10] 架構，即以 React [11] 作為前端框架，透過 Node.js [12] 和 Express [13] 運行伺服器，並以 MongoDB [14] 作為資料庫。

* 前端

我們使用 React 加上 TypeScript 進行開發，使用者介面主要使用的模組為 AntDesign [15]，地圖部分則是使用 IgniteUI [16] 的地圖模組，介面設計相當簡潔易懂，讓使用者可以直觀地檢視水情資料，並且若有人按下警急求救鈕，網頁也能夠立即顯示警急求救訊息。前端使用 Restful API 和 HTTPS 協定和伺服器溝通。

以下是前端的網址：<https://iot-term-project.netlify.app/>

備註：由於後端會自動休眠，因此第一次瀏覽時可能要花比較久的時間等待後端回傳資料。

* 後端

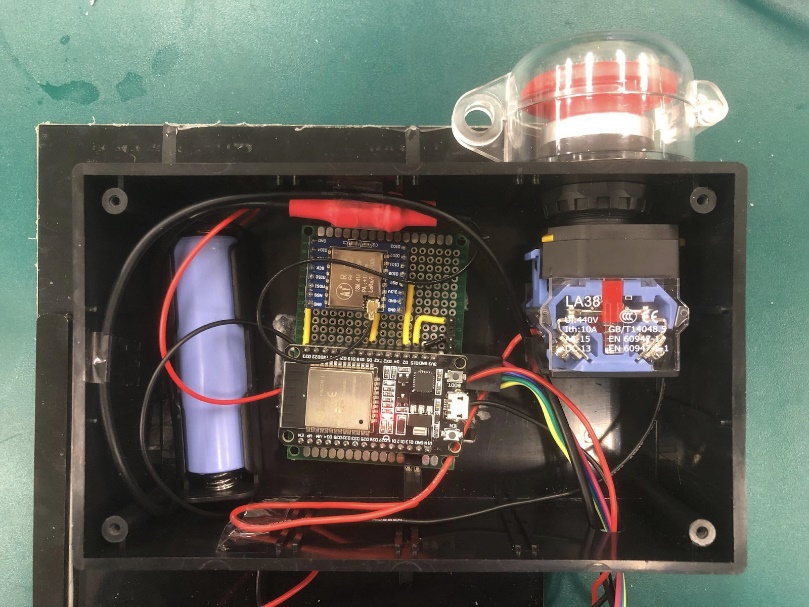
我們用 NodeJS 加上 Express 架設伺服器，伺服器 API 設計如下：

| Method | Route | Description |
| --- | --- | --- |
| GET | /cabins | 取得各個山屋的監測資料 |
| GET | /devices | 取得各個裝置的資料（如電量、上次活動時間等） |
| POST | /messages | 由硬體裝置中的Gateway發送監測資料到伺服器的API，本route設有一些檢查機制確認上傳的訊息是由我們核可的設備所發送。本API也用於接收警急求救訊息。 |

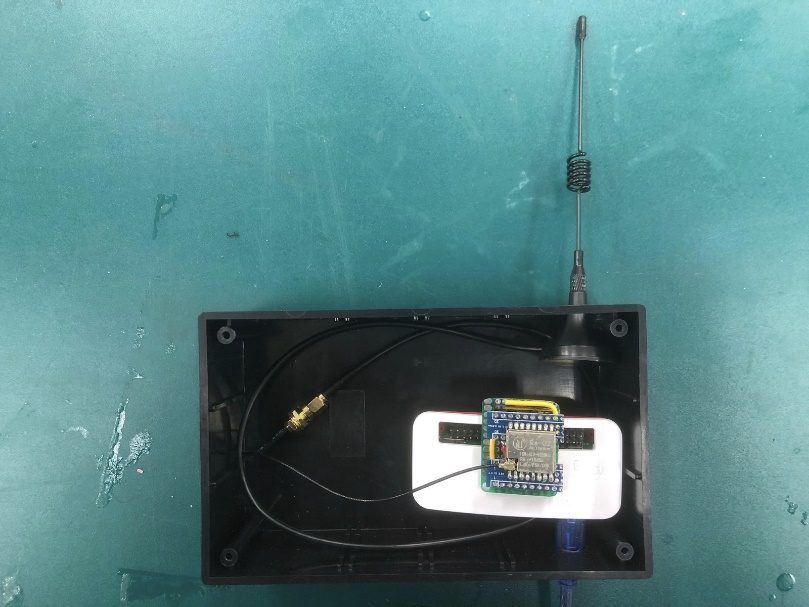
我們也把接收到的資料上傳到 MongoDB，因此這些資料都能受到更妥善的保護。

1. 實際演示

圖(四)為我們的解決方案中，山屋端會設置的模組內容。圖(五)則是 Gateway 模組的內容。



圖(四)：實作山屋端整合模組成果。注意到 ToF 模組與天線並未入鏡。



圖(五)：實作 Gateway 端整合模組成果，其中白色那顆是 RPi Zero W。

我們的程式碼可以在 Github 上存取，以下是連結：

<https://github.com/MartianSheep/Intro2IoT_FinalPrj/blob/main/RaspberryPi/lora.py>

以下連結則是我們的演示影片：<https://youtu.be/0AY_9QcEsWU>

1. 未來發展
2. LoRa 接力

由於一些偏遠山屋可能離山腳太遠，會導致一發 LoRa 打不到山腳。我們原本希望可以實現 LoRa 接力通訊與 Acknowledge 模式，然而由於使用的 sandeepmistry / arduino-LoRa 套件並不支援 LoRa 的 CAD (Channel Activity Detection) 模式，因此難以實現。未來希望使用其他套件加以實現。

1. 進階通訊

既然 LoRa 通訊網可以建立，則我們便可以加上一些基礎的文字通訊，包括文字輸入與顯示等，使救援隊更清楚目前緊急狀況為何，能夠規劃更完善的救援計畫，求救者也能即時得知救援情形。

1. 更多感測器

除了水位以外，有許多內容都是可以透過感測器測得後以 LoRa 傳輸，例如山屋的當前電力，或是山屋位址的溫溼度等。

1. 太陽能板

目前實作成果的太陽能板效力不是很足，希望未來可以使用效率更好且更大的太陽能板。

1. 防水

山上多雨，所以此模組是否防水至關重要。在這次實作中我們已經盡量把模組包起來，然而還是有一些地方尚待改進。

1. 夜晚情境

夜晚若遭遇緊急狀況，欲求救者可能不知道求救鈕在哪。因此在大紅色按鈕上面塗上螢光塗料可能有幫助。

1. 關於水位量測

除了山屋儲水槽以外，此模組應該也可以量測其他水位，例如湖面或河流等。如此一來便可以透過河水水位高低警示山洪暴發。

1. 關於網頁

這次的專案中由於時間因素，我們並未製作手機板的頁面。此外網頁上也可以加入更多內容，例如登入儲存最關心的山屋，以及警示當前規劃的登山路線是否需要備水，又建議在何處備水等等。

1. 結論

我們針對此題目開發的成品品質符合我們的預期，可見一開始的構想有足夠的可行性，不過若要實際應用，仍需要更多的測試證明其可靠性和持久性。

1. 分工

任瑨洋：

* 硬體製作
* ESP32 韌體程式碼
* 製作報告
* 採購

徐有齊：

* 軟體前後端與資料庫
* RPi 程式碼
* 製作報告
* 採購

1. 參考資料
2. 玉山國家公園，*圓峰山屋蓄水池出水口日前因結冰導致管體爆裂，經山莊管理員修復完畢，惟目前無降雨呈枯水狀態，請近日前往後四峰的山友記得提前於排雲山莊備水前往。*，玉山國家公園，2022，網址：<https://www.ysnp.gov.tw/Announcement/C001000?ID=f085d164-23cf-475c-a501-fc940ec3c296&PageType=1>
3. 玉山國家公園，*【進入旱季，行走山林，提早備水】*，玉山國家公園，2022，網址：<https://www.ysnp.gov.tw/Announcement/C001000?ID=cab6150d-b2ea-477e-a987-f929a62b4aa3&PageType=1>
4. 葉金川，*台灣的登山運動*，中華民國健行登山會，2007，網址：<http://www.alpineclub.org.tw/front/bin/ptdetail.phtml?Part=ho-165-1&Rcg=36>
5. Last Minute Engineers, *Insight Into ESP32 Sleep Modes & Their Power Consumption*, Last Minute Engineers, 2022, URL: <https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/>
6. ArduinoJson, URL: <https://arduinojson.org/>
7. Sandeep Mistry, *sandeepmistry / arduino-LoRa*, Github, 2022, URL: <https://github.com/sandeepmistry/arduino-LoRa>
8. Semtech, *What are LoRa® and LoRaWAN®?*, LoRa Developer Portal, URL: <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>
9. Sharpsensoruser, *Application Guide for Foxconn FSTOF2002C0D Time of Flight Sensor Module*, Github, 2022, URL: <https://github.com/sharpsensoruser/sharp-sensor-demos/wiki/Application-Guide-for-Foxconn-FSTOF2002C0D-Time-of-Flight-Sensor-Module>
10. Nettigo, *LoRa RA-02 SX1278 433MHz*, Nettigo.eu, URL: <https://nettigo.eu/products/lora-ra-02-sx1278-433mhz>
11. MongoDB, MERN Stack Explained, MongoDB, URL: <https://www.mongodb.com/mern-stack>
12. React, URL: <https://reactjs.org/>
13. Node.js, URL: <https://nodejs.org/en/>
14. Express, URL: <https://expressjs.com/>
15. MongoDB, URL: <https://www.mongodb.com/>
16. AntDesign, URL: <https://ant.design/>
17. IgniteUI, URL: <https://github.com/IgniteUI/igniteui-react>