以 Raspberry Pi 為核心的遙測平台開發：面向太空遙測的替代驗證

作者：詹大力
指導教授：劉宗哲

September 2025

摘要

本研究旨在建置一套以 太空遙測為最終應用目標之低成本 IoT 遙測平台，並分別以 大氣遙測（探空氣球）與 海洋遙測（太空遙測替代驗證）作為實驗場域。系統採用 Raspberry Pi Zero 2 W、UPS HAT(C) 與 IoT Node(A)[1]，實現長距離、低功耗的資料蒐集與回傳。實驗包含暑期課程之教學實作與維也納 Donau Park 的戶外測試，獲得軌跡與連線表現等實證數據；另以保麗龍空心浮球與 PVC 水管構成海面漂浮載具，並於接收端加裝自製 Yagi-Uda 天線，用以驗證遠距接收能力。目前已完成系統整合，預計於 **10** 月中旬前往台東富岡漁港進行海上實測，並由當地漁船協助投放漂浮裝置。目前結果顯示，本平台具備跨場域可移植性與教育推廣價值，對未來延伸至太空遙測任務具有可行性。

關鍵詞：Raspberry Pi、Wireless Transmission、Data Flow、Housekeeping、Antenna Design

[1]IoT Node(A) 內含 LoRa 與 GPS，本文不再重複列出元件。

目錄

簡介

遙測（Remote Sensing）是利用感測器在遠距位置量測環境參數並進行資料回傳的技術，廣泛應用於大氣觀測、海洋監測與太空科學任務。然受限於成本、法規與場域門檻，直接於太空環境進行驗證不易；因此，本研究以「太空遙測」為最終願景，採用可快速迭代的替代場域，包含大氣與海洋。來驗證系統的可行性與可移植性。

本研究建置一套以 Raspberry Pi Zero 2 W、UPS HAT(C) 與 IoT Node(A) 為核心的低成本 IoT 遙測平台，透過模組化堆疊與雙裝置互傳設計，達成長距離、低功耗的資料蒐集與回傳。平台首先在「大氣遙測」場域以探空氣球進行驗證；研究過程除於維也納 Donau Park 進行戶外實測並獲得軌跡圖與連線表現外，亦在出國前以本系統支援為期五日的高空載台暑期課程，擔任講師並編撰技術手冊，累積教學與場測經驗。其後，為了貼近太空遙測任務中「長距漂移、能源受限、通聯困難」等挑戰，進一步將平台延伸至「海洋遙測」替代驗證：以保麗龍空心浮球配合 PVC 水管構成漂浮載具作為發送端，接收端則於原平台加裝自製 Yagi-Uda 天線，以提升長距離接收能力，模擬地面站對遙測載具之通聯情境。

綜合而言，本研究的主要貢獻如下：

- 通用遙測平台建置：以 Raspberry Pi 與 IoT Node(A) 為核心之模組化架構，兼顧低成本與快速擴充。
- 大氣場域驗證：完成探空氣球硬體與軟體流程，實測獲得軌跡與通聯數據，驗證平台於近地大氣的可行性。
- 海洋替代驗證：以漂浮載具與 Yagi-Uda 強化接收端，驗證長距離、低功耗、無人值守之通訊能力，作為太空遙測的替代試驗。
- 教育推廣與知識產出：以系統為基礎開設暑期課程並編寫技術手冊，展現平台於教學與實作上的移植性。

本報告後續章節將依序說明研究動機與目的、相關研究與文獻、系統架構與設計、兩個應用場域之實作與測試（大氣遙測與海洋遙測）、綜合討論，最後提出結論與未來展望。

研究動機

遙測任務在大氣、海洋與太空場域皆面臨長距離通聯、能源受限與載具無人值守等挑戰；但太空環境的驗證成本高、法規與場域門檻亦高。為了在可負擔的條件下驗證系統可行性，本研究以可快速迭代的替代場域切入：在近地大氣以探空氣球實作流程，在海面以漂浮載具模擬長距漂移資料回傳，兩者皆以相同之 IoT 平台為核心，以評估跨場域可移植性與未來延伸至太空遙測的可能。

平台選型方面，Raspberry Pi Zero 2 W 具備社群生態成熟、周邊擴充容易與軟體開發門檻低等優點；搭配 UPS HAT(C) 可提升續航與斷電保護；IoT Node(A) 則提供長距離低功耗通訊與定位能力。過去在維也納 Donau Park 的戶外測試與為期五日的高空載台暑期課程實作，顯示該平台有良好的教學推廣性與實作可得性，進一步支持以此為基礎展開跨場域驗證。

本研究的核心動機在於：以低成本、可重複、可教學推廣的方式，建立一套可於不同場域反覆驗證與迭代的遙測平台，並以「海洋遙測作為太空遙測的替代驗證」來降低開發風險與成本，最終支撐未來朝太空任務規格前進。

研究目的

本研究以「太空遙測為最終願景」為指引，設定以下總目標與可量測的子目標／成功準則：

總目標

建立一套以 Raspberry Pi Zero 2 W、UPS HAT(C) 與 IoT Node(A) 為核心的遙測平台，完成於大氣與海洋兩個替代場域的整合實證，並形成可複製之設計準則，作為未來進一步朝太空遙測應用之基礎。

子目標與量化指標

- 通訊效能：在開闊地/海面環境達到有效鏈路距離 $\geq$ 20 km，封包成功率（PSR）在 15 km 距離下 $\geq 75\%$。
- 能源續航：平均功耗 $\leq$ 1 W，使用 UPS HAT(C) 於實測任務中連續運作時間 $\geq$ 12 小時。
- 定位與軌跡：GPS 定位平均誤差 $\leq$ 10 m；能重建完整漂移/飛行軌跡，缺失片段比例 $\leq 5\%$。
- 海洋替代驗證：漂浮載具於海面連續運作 $\geq$ 24 小時；接收端加裝自製 Yagi-Uda 天線後，實測接收靈敏度或最大可用距離相較基準提升，天線增益 $>$ 11dBi。
- 可重現性與教學價值：整理裝配與軟體部署文件，外部人員依文件可於 $<$ 24 小時內完成系統部署並重現基本實驗。

驗證方法與成功準則

- 測試設計：以地面開闊地點對點量測為基準，分別在國立屏東大學操場、Donau Park 戶外、海面漂流情境進行任務測試；控制變因包含發送功率、天線高度/型式、封包間隔等。
- 資料指標：記錄 RSSI/SNR、封包遺失率、功耗曲線、GPS 取樣率與軌跡完整度；以任務距離、時間與資料量進行歸一化比較。
- 成功定義：當上述量化指標同時達成各自門檻，且系統可於兩場域穩定運行並產出可重現的流程文件，即視為達成研究目的。

相關研究與文獻探討

LPWAN 與 LoRa 調變技術

低功耗廣域網路（Low-Power Wide-Area Network, LPWAN）致力於在極低功耗下達成公里級以上傳輸距離 [1, 2]。LoRa 採用線性掃頻擴頻（chirp spread spectrum, CSS）調變，透過可調式擴頻因子（SF）、頻寬（BW）與碼率（CR），在接收靈敏度與資料率之間取捨 [3, 4, 5]。鏈路預算（link budget）常以

$$P_r = P_t + G_t + G_r - L_{\text{fs}} - L_{\text{misc}}$$

估算，其中自由空間路徑損失（dB）近似為

$$L_{\text{fs}} \approx 32.45 + 20\log_{10}(f_{\text{MHz}}) + 20\log_{10}(d_{\text{km}})[6].$$

在較高 SF（如 SF10/SF11）下可獲得較佳接收靈敏度，代價是空口占用時間（ToA）上升，需於功耗與佔空比之間取得平衡。

模組與頻段選擇（本研究配置）

本研究採用 Ai-Thinker **RA-01** 模組（晶片為 **Semtech SX1278**），工作於 433 MHz ISM 頻段 [7, 5]。目前規劃的初始參數如下：

表 1: LoRa 無線電設定

參數	值
模組 / 晶片	Ai-Thinker RA-01 / SX1278
頻率 f	433 MHz
擴頻因子 SF	11
頻寬 BW	125kHz
碼率 CR	4/7
發射功率 P_t	+18dBm
封包長度	16 bytes
封包間隔 / 佔空比	15 秒
接收端天線（型式/增益）	Yagi-Uda / 11-13dBi

LoRa 通訊拓撲與應用案例

LoRa 生態可分為點對點（P2P）與符合 LoRaWAN 的星狀網路兩類 [8]。P2P 部署快速、延遲低，適合研究原型與定制協定；LoRaWAN 透過閘道器與網路伺服器實現多

節點管理，更適合長期佈建。對於本研究之探空氣球與海洋替代驗證，P2P 能簡化系統與縮短時延，並保留對封包格式與重傳策略的掌控。

GNSS 定位於遙測中的角色

GNSS（如 GPS）支援軌跡重建與資料地理標註。為兼顧功耗與資料完整度，實務上常採間歇取樣、基於品質指標（如 HDOP）之過濾，以及冷／熱啟動策略優化 [9, 10]。於高動態（氣球）與海面漂移（漂流器）場景，天線佈局與遮蔽條件對定位品質影響顯著，需與機構設計共同權衡。

探空氣球相關研究

探空氣球（HAB）在教育與近太空實驗廣為應用。以 LoRa 為核心之輕量化酬載可在成本、功耗與可重複性上取得優勢；隨高度提升之視距（LOS）有助於延伸通聯距離，但仍需遵守頻譜與發射功率之在地規範 [11, 12]。

海洋漂流器與替代驗證

海洋表層漂流器用於量測洋流與海面參數，設計上追求隨流性（minimize wind slip）、穩定浮力與抗波動 [13, 14, 15]。海洋環境在覆蓋稀疏、節點不易回收、能源受限與長距通聯等面向，與太空遙測具相似挑戰；因此以海洋遙測作為替代驗證，可在可控成本下驗證鏈路穩健性、電源管理與封包策略。

Yagi–Uda 天線與接收距離

Yagi-Uda 天線提供高指向性與主瓣增益，能在不增加發送端功耗下提升有效通聯距離與抗干擾能力 [16]。視距條件下，無線電地平線距離可由天線高度粗估為

$$d_{\mathrm{km}} \approx 3.57(\sqrt{h_{1,\mathrm{m}}} + \sqrt{h_{2,\mathrm{m}}})[6],$$

顯示提高天線高度與主瓣指向對實際可用距離的重要性。

研究缺口與本研究定位

既有工作多在單一場域（僅大氣或僅海洋）驗證；較少研究以同一套低成本 IoT 平台跨場域（大氣與海洋）連續驗證，並特別以「海洋遙測」作為「太空遙測」替代手段來設計鏈路與能源策略。本研究以 Raspberry Pi 與 IoT Node(A) 為核心，採用 RA-01（SX1278, 433 MHz）模組，首先於大氣場域驗證，再於海洋場域針對長距漂移與地面站接收進行替代驗證，累積設計準則以支撐未來太空遙測任務。

系統架構與設計

LoRa 無線電設定

表 2: LoRa 無線電設定

參數	值
模組 / 晶片	Ai-Thinker RA-01 / SX1278
頻率 f	433 MHz
擴頻因子 SF	11
頻寬 BW	125kHz
碼率 CR	4/7
發射功率 P_t	+18dBm
封包長度	16 bytes
封包間隔	15 秒
接收端天線（型式/增益）	Yagi-Uda / 11-13dBi

硬體架構

表 3: 硬體元件與連接

模組	介面/連接	備註
Raspberry Pi Zero 2 W	—	主控；作業系統為 Raspberry Pi OS Lite
UPS HAT(C) + 電池	電源	Waveshare LiPo WS-803040，3.7 V，1000 mAh；供電/保護
IoT Node(A)	I^2C（與 RPi）	疊板連接；LoRa 433 MHz；GPS 由 `gpsd` 提供
發送端天線	SMA	IoT Node(A) 隨附天線
接收端 Yagi-Uda	SMA	8 元素、433MHz、11-13dBi

電源與功耗設計

表 4: 功耗與續航估算

項目	數值	備註
平均功耗	$\approx$ 1W	任務時序估算
電池容量	1000 mAh	LiPo，3.7V
預估續航	9-12 小時	以裝上太陽能板後的平均日照時間計算

韌體與軟體架構

- 作業系統／語言：Raspberry Pi OS Lite、Python 3。
- 系統服務：GPS 使用 `gpsd`。
- 主要模組：GPS 解析、封包化、記錄/重傳、電源管理、設定檔。

通訊協定與封包格式

表 5: 遙測封包格式

欄位	長度	單位	說明
Latitude	4 B	10^{-5} 度	有號整數，緯度 $\times 10^5$
Longitude	4 B	10^{-5} 度	有號整數，經度 $\times 10^5$
Timestamp	4 B	s（UTC+8）	GPS 或系統時間（epoch 格式）
Device ID	1 B	—	裝置識別碼（例：0x01）
Flags	1 B	bitfield	GPS fix 狀態（0/1）等
Reserve 1	1 B	—	保留欄位，可依裝設感測器決定用途
Reserve 2	1 B	—	保留欄位，可依裝設感測器決定用途

任務時序

INIT $\rightarrow$ GPS_FIX $\rightarrow$ PACKETIZE $\rightarrow$ TX $\rightarrow$ LOG $\rightarrow$ SLEEP；上報間隔 15 秒。

天線與射頻設計

- 發送端天線：IoT Node(A) 隨附天線、ANT433-SMA-J-RG174-1.5M
- 接收端天線：IoT Node(A) 隨附天線、自製 Yagi-Uda 天線：8 元素，433MHz，增益 11-13dBi。
- **SMA** 連接：發送端與接收端均以 SMA 接頭連接。

機構與環境防護

- 漂浮載具：保麗龍空心浮球直徑 25 cm；PVC 管外徑 2.5 cm，內徑 2.3 cm，長度 45 cm。

測試計畫與量測指標

- 場測：屏東大學操場、Donau Park、海面測試。
- 指標：RSSI、SNR、PSR、最大可用距離、平均功耗、GPS 定位成功率與精度。
- 資料紀錄：CSV 欄位（lat, lon, timestamp, Device ID, flags），檔名格式為.txt。

應用一：大氣遙測（探空氣球）

系統設計與組裝

本應用以乳膠氣球搭配氦氣作為「大氣遙測」的載台。暑期課程中採繫留（**tethered**）方式測試：以釣魚線一端固定在鐵架、另一端固定在裝置，避免氣球漂移至遠處。酬載核心為 Raspberry Pi Zero 2 W、UPS HAT(C)、IoT Node(A)，電源採 Waveshare WS-803040 鋰電（3.7 V，1000 mAh）。無外殼，整機倒掛以使發送端天線朝下（近似垂直極化），裝置與天線之間以 SMA-SMA 線連接；發送端天線為 IoT node(A) 隨附天線。其餘機構與配重配置見表 6。

表 6: 探空氣球載台與硬體配置

項目	規格 / 說明
氣球	乳膠；填充氦氣；繫留測試（釣魚線）
酬載總重	250g（含電池/天線）
電池	Waveshare WS-803040，3.7 V，1000 mAh
主控與模組	RPi Zero 2 W + UPS HAT(C) + IoT Node(A)
發送端天線	IoT node(A) 隨附天線
安裝與極化	倒掛，天線朝下（近似垂直極化）

軟體流程與數據收集

作業系統為 Raspberry Pi OS Lite，主要以 Python 3 撰寫；GPS 由 `gpsd` 服務提供；433 MHz，SF=7，BW=125kHz，CR=4/5。流程：開機自動執行程式 → 取得 GPS fix → 封包化（經緯度、UTC+8 時間戳、裝置識別碼、GPS fix 狀態）→ LoRa 發送 → 記錄日誌；異常時記錄錯誤並重試。

測試流程

暑期課程繫留測試

日期時間：**7/05** 約 **14:00**；使用釣魚線綁在裝置上繫留抬升，高度變化有限。概述如表 7。

表 7: 暑期課程繫留測試條件與結果概要

項目	設定 / 現場情況
地點	國立屏東大學操場
時間	7/05 約 14:00
抬升方式	繫留（釣魚線，一端鐵架、一端裝置）
地面站	Raspberry Pi 和 IoT node(A) 疊版連接
上報設定	SF=7；BW=125kHz；CR=4/5；間隔 =15 秒傳一次
任務時長	約 40 分鐘
資料接收	成功接收到資料

結果與初步觀察

本次繫留測試以驗證升空與資料回傳流程為目標。酬載成功升空，並透過 LoRa 完成資料傳輸，地面站順利接收到有效 GPS 封包。傳輸設定為 SF7、BW=125 kHz、CR=4/5、間隔 15 秒、發射功率 1 W。此結果證實系統設計具備可行性。

改進計畫

後續測試將持續強化資料穩定性，方向包含：

- 進行地面鏈路前測，確認無線參數最佳化。
- 改善酬載穩定性以降低旋轉與姿態影響。
- 地面站搭配指向性天線與放大器，以提升接收率。

維也納 **Donau Park** 步行測試（固定接收／移動發送）

目的　驗證 LoRa 在戶外開闊地之基本長距離通訊可用性（功能性測試），非高空氣球任務。

方法　將接收端裝置置於定點（地面），發送端裝置佩掛於身上開始步行；持續上報位置與基礎遙測欄位（時間、經緯度、高度等），以評估不同分離距離與方位下的連線情況。

初步觀察　於約 0.33 km 的分離距離下可完成基本連線。後續可加入 Yagi 提升鏈路餘裕，並記錄更完整的封包成功率與距離關係。

表 8: Donau Park 步行測試條件與結果概要

項目	設定 / 現場情況
地點與時間	Donau Park；7/26；約 14:00-15:00
拓撲	接收端定點；發送端隨人步行
接收端天線	IoT Node(A) 隨附天線
發送端天線	IoT Node(A) 隨附天線
無線參數	433 MHz；SF=11；BW=125kHz；CR=4/7
起點座標	(48.2400, 16.4180)
終點座標	(48.2418, 16.4144)
最大分離距離	約 **0.33 km**（大圓距估算）
備註	人體遮擋、手持角度與極化影響明顯；園區建物/樹木造成多徑

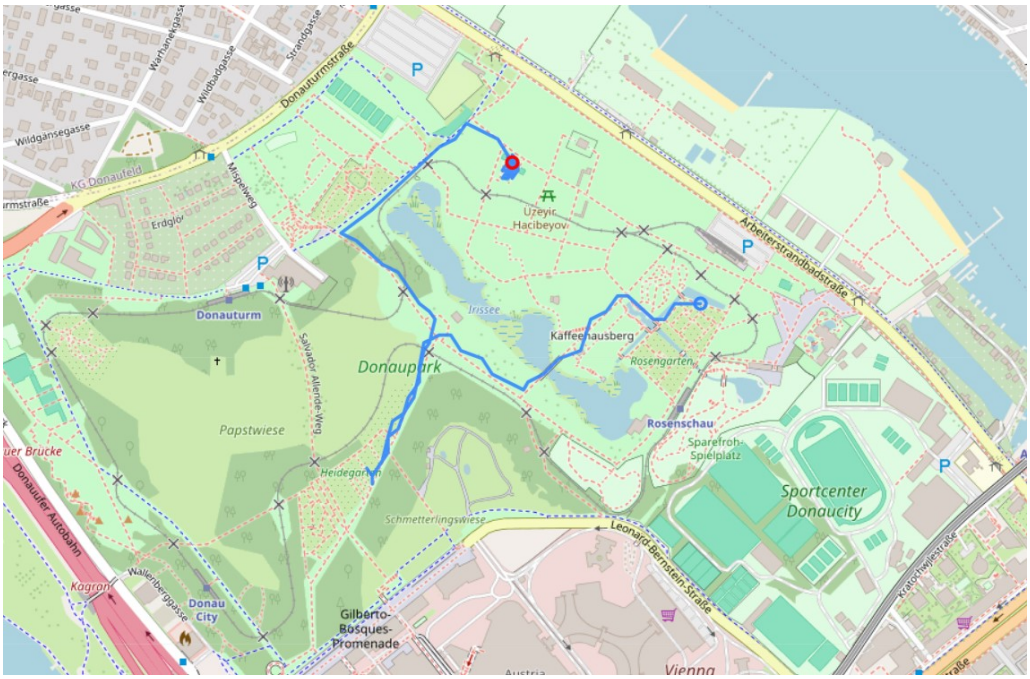

圖 1: Donau Park 步行測試軌跡

應用二：海洋遙測（太空遙測替代驗證）

漂浮裝置設計

結構配置

本裝置以上半球與下半球的功能分工為核心設計理念。上半球為一個中空保麗龍浮球，PVC 水管貫穿其中並提供固定支撐，外部黏附 ANT433-SMA-J-RG174 天線，並搭配低雜訊放大器（+30 dB）以強化接收效能。下半球則以金屬棒貫穿作為配重，使裝置在海上漂浮時能維持直立，確保天線朝上並維持穩定方位。

核心電子設備

酬載電子元件配置如下：

- Raspberry Pi Zero 2 W
- UPS HAT(C)
- IoT Node(A)
- 四片 1 W 太陽能板與穩壓模組

透過太陽能持續供電，裝置可延長於海上漂浮的作業時間，提升實驗效益。

地面站配置

基礎硬體

地面站以 Raspberry Pi 4 搭配 IoT Node(A) 疊板組成，負責接收並記錄由漂浮裝置傳回的資料。

自製天線系統

地面站搭配一具自製的八元素 Yagi-Uda 指向性天線，工作於 433 MHz。透過該天線與低雜訊放大器，本系統可顯著提升鏈路餘裕並擴展接收範圍。

測試方法

漂浮裝置預計投放於近海區域，隨洋流漂移並持續以 GPS 定位回報座標。傳輸設定如下：

- 頻率：433 MHz
- 擴頻因子（SF）：11
- 頻寬（BW）：125 kHz
- 編碼率（CR）：4/7
- 上報間隔：每 15 秒一次

地面站利用 Yagi-Uda 天線指向漂浮裝置方向，接收並記錄所有回傳的 GPS 資料。

目前進度：Yagi-Uda 天線測試

於高屏溪舊鐵橋及其南方約 30 公里處進行自製八元素 Yagi-Uda 天線測試，確認本系統至少具備完成 30 公里級別資料傳輸的能力。此結果為後續海上實驗提供了鏈路性能的初步驗證依據，顯示天線設計足以支援長距離漂流器資料回傳之需求。

後續計畫

本研究之海洋遙測測試預計於 **2025** 年 **10** 月中旬進行，地點為台東富岡漁港。屆時將在劉宗哲教授協助下，由當地漁船將裝置投放至外海，以模擬漂流器隨洋流漂移並進行 GPS 定位回報。相關數據（如通訊距離、封包接收率、漂流軌跡）將於後續試驗完成後補充與分析。

本次海上投放亦被視為一項低成本、可重複驗證的太空遙測替代驗證實驗，不僅有助於鏈路性能與系統穩定度之檢驗，更可為後續拓展至大氣與太空領域提供實證基礎。

參考文獻

[1] U. Raza, P. Kulkarni, and M. Sooriyabandara, “Low Power Wide Area Networks: An Overview,” *IEEE Communications Surveys & Tutorials*, 19(2), 855–873, 2017.

[2] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, “A Study of LoRa: Long Range & Low Power Networks for the Internet of Things,” *Sensors*, 16(9), 1466, 2016.

[3] Semtech Corporation, *LoRa Modulation Basics (Application Note AN1200.22)*, 2015.

[4] Semtech Corporation, *SX1272/3/6/7/8 LoRa Modem Designer’s Guide (AN1200.13)*, 2013.

[5] Semtech Corporation, *SX1276/77/78/79 Low Power Long Range Transceiver Datasheet*, Rev. 4, 2015.

[6] ITU-R, “Recommendation P.525-4: Calculation of Free-Space Attenuation,” 2019.

[7] Ai-Thinker, *RA-01 LoRa Module Specification*, 2019.

[8] LoRa Alliance, *LoRaWAN Specification v1.1*, 2017.

[9] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*, 2nd ed., 2011.

[10] E. D. Kaplan and C. J. Hegarty (eds.), *Understanding GPS/GNSS: Principles and Applications*, 3rd ed., Artech House, 2017.

[11] D. Akerman, “Receiving From LoRa High Altitude Balloons,” 2021. (Online article)

[12] J. Krueger, “Testing Next-Generation Telemetry Using LoRaWAN on High-Altitude Balloons,” *Academic High Altitude Conference (AHAC)*, 2024.

[13] R. Lumpkin and M. Pazos, “Measuring Surface Currents with SVP Drifters,” in *Lagrangian Analysis and Prediction of Coastal and Ocean Dynamics*, Cambridge Univ. Press, 2007.

[14] S. Elipot, R. Lumpkin, G. Prieto, et al., “A Global Surface Drifter Data Set at Hourly Resolution,” *Journal of Geophysical Research: Oceans*, 121(7), 3987–4002, 2016.

[15] NOAA/AOML, “The Global Drifter Program (GDP),” program web page, accessed 2025.

[16] C. A. Balanis, *Antenna Theory: Analysis and Design*, 4th ed., Wiley, 2016.

附錄

程式碼節錄範例

以下本研究有使用的程式：(使用 Python3 執行)

Listing 1: LoRa/GPS 資料上報流程（發送端）

```
import os, csv, time, struct
from datetime import datetime, timedelta
import serial
import smbus

# ===== 常數 =====
DEVICE_ID       = 0x01
LOG_PATH        = "/home/bigpower/ocean_log.txt"
SERIAL_PORT     = "/dev/ttySC0"
SERIAL_BAUD     = 115200
I2C_ADDR        = 0x16
CTRL_REG        = 0x23
SEND_PERIOD_SEC = 10.0

# LoRa 暫存器（依 EP-0105）
REG_SF     = 0x22
REG_BW     = 0x24
REG_CR     = 0x25
REG_LDO    = 0x26
REG_PRE_H  = 0x2A
REG_PRE_L  = 0x2B
REG_CRC    = 0x2C
REG_HDR    = 0x2D
REG_PA     = 0x29

BIT_TXTRIG = 0x01
BIT_RESET  = 0x40

PROFILE = {
    "SF": 11,
    "BW": 7,        # 假設 7 = 125 kHz（依廠商映射）
    "CR": 7,        # 4/7（依廠商映射）
    "PREAMBLE": 12,
    "HDR": 0,       # explicit
    "CRC": 1,       # on
    "PA": 17,
```

```
    "LDO": 1,
}

bus = smbus.SMBus(1)
ser = serial.Serial(SERIAL_PORT, SERIAL_BAUD, timeout=1)

def w(addr, val): bus.write_byte_data(I2C_ADDR, addr, val & 0xFF)
def r(addr): return bus.read_byte_data(I2C_ADDR, addr)

def apply_profile():
    w(CTRL_REG, BIT_RESET); time.sleep(0.6)
    w(REG_SF, PROFILE["SF"])
    w(REG_BW, PROFILE["BW"])
    w(REG_CR, PROFILE["CR"])
    w(REG_HDR, PROFILE["HDR"])
    w(REG_CRC, PROFILE["CRC"])
    w(REG_PRE_H, (PROFILE["PREAMBLE"] >> 8) & 0xFF)
    w(REG_PRE_L, PROFILE["PREAMBLE"] & 0xFF)
    w(REG_PA, PROFILE["PA"])
    w(REG_LDO, PROFILE["LDO"])

# ---- NMEA 轉十進位度數 ----
def convert_to_decimal(nmea_str, direction):
    if not nmea_str: return None
    try:
        if direction in ('N','S'):
            deg = int(nmea_str[:2]); minute = float(nmea_str[2:])
        else:
            deg = int(nmea_str[:3]); minute = float(nmea_str[3:])
        d = deg + minute/60.0
        return -d if direction in ('S','W') else d
    except:
        return None

# ---- 取一次 GPS ----
def get_gps_info():
    try: ser.write(b"AT+GPS=1\r")
    except: pass
    time.sleep(2)
    try:
        ser.reset_input_buffer()
        ser.write(b"AT+GPSRD=1\r")
    except: pass
    time.sleep(1.0)

    lat = lon = None
    ts_epoch = None
    fix_ok = False
    hh = mm = 0

    t0 = time.time()
```

```
    while time.time()-t0 < 8.0:
        if ser.in_waiting:
            line = ser.readline().decode(errors="ignore").strip()
            if line.startswith(("$GNRMC", "$GPRMC")):
                p = line.split(",")
                if len(p) > 9:
                    fix_ok = (p[2] == "A")
                    lat = convert_to_decimal(p[3], p[4])
                    lon = convert_to_decimal(p[5], p[6])
                    try:
                        hhmmss = p[1][:6] if len(p[1]) >= 6 else "
                            000000"
                        dt = datetime.strptime(p[9] + hhmmss, "%d%m%
                            y%H%M%S")
                        local = dt + timedelta(hours=8)
                        ts_epoch = int(local.timestamp())
                        hh, mm = local.hour, local.minute
                    except:
                        ts_epoch = int(time.time())
                        lt = time.localtime(ts_epoch)
                        hh, mm = lt.tm_hour, lt.tm_min
                break
        time.sleep(0.1)

    if ts_epoch is None:
        ts_epoch = int(time.time())
        lt = time.localtime(ts_epoch)
        hh, mm = lt.tm_hour, lt.tm_min

    return lat, lon, ts_epoch, bool(fix_ok), hh, mm

# ---- log 檔 ----
def init_log():
    if not os.path.exists(LOG_PATH):
        with open(LOG_PATH, "w", newline="") as f:
            csv.writer(f).writerow(["entry_id","lat","lon","
                timestamp","fix","hh","mm"])

def append_log(lat, lon, ts, fix_ok, hh, mm):
    if not os.path.exists(LOG_PATH):
        init_log()
    with open(LOG_PATH, "r") as f:
        entry_id = sum(1 for _ in f)  # header 也算一行
    lat_v = float(lat) if lat is not None else 0.0
    lon_v = float(lon) if lon is not None else 0.0
    with open(LOG_PATH, "a", newline="") as f:
        csv.writer(f).writerow([entry_id, f"{lat_v:.6f}", f"{lon_v
            :.6f}", int(ts), 1 if fix_ok else 0, hh, mm])
    return entry_id

def read_latest_row():
```

```
    if not os.path.exists(LOG_PATH): return None
    with open(LOG_PATH, "r") as f:
        rows = list(csv.reader(f))
        return rows[-1] if len(rows) > 1 else None

# ---- 發送（16B）>iiIBBBB ----
def send_latest_from_row(row):
    try:
        lat = float(row[1]); lon = float(row[2]); ts = int(float(row
            [3]))
        fix = int(row[4]); hh = int(row[5]); mm = int(row[6])
    except:
        lat = lon = 0.0; ts = int(time.time()); fix = 0; hh = mm = 0

    lat_i = int(lat * 1e5)
    lon_i = int(lon * 1e5)
    flags = 0x01 if fix == 1 else 0x00
    payload = struct.pack(">iiIBBBB", lat_i, lon_i, ts, DEVICE_ID,
        flags, hh & 0xFF, mm & 0xFF)

    bus.write_i2c_block_data(I2C_ADDR, 0x01, list(payload))
    w(CTRL_REG, BIT_TXTRIG)

# ---- 主程式 ----
if __name__ == "__main__":
    apply_profile()
    init_log()
    print("[INIT] Sender ready with matched LoRa profile")

    while True:
        lat, lon, ts, fix_ok, hh, mm = get_gps_info()
        eid = append_log(lat, lon, ts, fix_ok, hh, mm)
        latest = read_latest_row()
        if latest:
            send_latest_from_row(latest)
            print(f"[TX] id={eid} fix={latest[4]} lat={latest[1]}
                lon={latest[2]} ts={latest[3]} {latest[5]}:{latest
                [6]}")
        else:
            print("[WARN] no latest row")
        time.sleep(SEND_PERIOD_SEC)
```

Listing 2: LoRa/GPS 資料上報流程（接收端）

```
import os, csv, time, struct
import smbus

I2C_ADDR, CTRL_REG = 0x16, 0x23
BIT_RXNE, BIT_TXTRIG, BIT_RESET = 0x02, 0x01, 0x40
LOG_PATH = "/home/bigpower/ocean_log.txt"

# LoRa 暫存器（依 EP-0105）
```

```
REG_SF     = 0x22
REG_BW     = 0x24
REG_CR     = 0x25
REG_LDO    = 0x26
REG_PRE_H  = 0x2A
REG_PRE_L  = 0x2B
REG_CRC    = 0x2C
REG_HDR    = 0x2D
REG_PA     = 0x29

PROFILE = {
    "SF": 11,
    "BW": 7,
    "CR": 7,
    "PREAMBLE": 12,
    "HDR": 0,
    "CRC": 1,
    "PA": 17,
    "LDO": 1,
}

bus = smbus.SMBus(1)
def w(a,v): bus.write_byte_data(I2C_ADDR, a, v & 0xFF)
def r(a):   return bus.read_byte_data(I2C_ADDR, a)

def apply_profile():
    w(CTRL_REG, BIT_RESET); time.sleep(0.6)
    w(REG_SF, PROFILE["SF"])
    w(REG_BW, PROFILE["BW"])
    w(REG_CR, PROFILE["CR"])
    w(REG_HDR, PROFILE["HDR"])
    w(REG_CRC, PROFILE["CRC"])
    w(REG_PRE_H, (PROFILE["PREAMBLE"] >> 8) & 0xFF)
    w(REG_PRE_L, PROFILE["PREAMBLE"] & 0xFF)
    w(REG_PA, PROFILE["PA"])
    w(REG_LDO, PROFILE["LDO"])

def init_log():
    if not os.path.exists(LOG_PATH):
        with open(LOG_PATH, "w", newline="") as f:
            csv.writer(f).writerow(["entry_id","lat","lon","
               timestamp","fix","hh","mm"])

def append_row(lat, lon, ts, fix, hh, mm):
    with open(LOG_PATH, "r") as f:
        entry_id = sum(1 for _ in f)
    with open(LOG_PATH, "a", newline="") as f:
        csv.writer(f).writerow([entry_id, f"{lat:.6f}", f"{lon:.6f}"
           , int(ts), int(fix), int(hh), int(mm)])
    return entry_id

```

```
58 if __name__ == "__main__":
59     apply_profile()
60     init_log()
61     print("[INIT] Receiver ready with matched LoRa profile")
62 
63     seq = 0
64     while True:
65         if r(CTRL_REG) & BIT_RXNE:
66             raw = bus.read_i2c_block_data(I2C_ADDR, 0x11, 16)
67             w(CTRL_REG, 0x00)
68             try:
69                 # 封包: >iiIBBBB
70                 lat_i, lon_i, ts, dev_id, flags, hh, mm = struct.
                       unpack(">iiIBBBB", bytes(raw))
71                 lat = lat_i / 1e5
72                 lon = lon_i / 1e5
73                 fix = 1 if (flags & 0x01) else 0
74                 eid = append_row(lat, lon, ts, fix, hh, mm)
75                 print(f"[RX] id={eid} dev={dev_id} fix={'OK' if fix
                       else 'NO'} "
76                       f"{lat:.6f},{lon:.6f} ts={ts} {hh:02d}:{mm:02d
                          }")
77             except Exception as e:
78                 print("Parse error:", e, raw)
79         time.sleep(0.01)
```