

物聯網導論期末專題：航道水深即時量測與預測系統

許定為 R10942074

陳遠傑 R09921037

一、動機：

之前長榮海運的貨船於蘇伊士運河擱淺，造成航道壅塞，蘇伊士運河掌管全球約 12% 的貿易流量，若航道壅塞，可能會造成全球消費者日常品運輸延遲，嚴重影響全球經濟，故關於港灣、航道的水深是保障船舶安全航行和提高航運效益之關鍵。

現今有的量測水深的裝置，如聲納、光學設備，都具備精準的量測水深能力，但對於未來水深的變化，光靠量測水深是不夠的，港灣水深牽涉範圍廣大，如當日風向、天氣等，都是影響水深變化的重要元素之一，故只有蒐集過去水深的資料來預測未來沒辦法達到精準的預測。

而現有的 AI 技術可以做到精準預測，各大港口都有長時間累積的統計資料，可以藉由將這些資料搭建 AI 模型，從這些資料找出港灣水深變化的關係，故可以藉由現在的資訊來預測未來，讓使用航道者有一個很好的港灣水深參考來源。

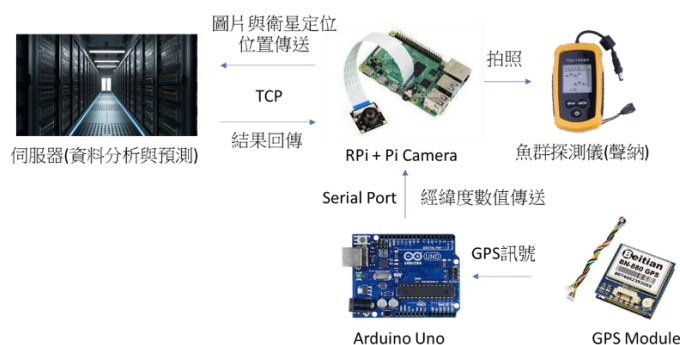
而這次的專題，理想上是希望能夠搭建無人船上安裝聲學裝置量測港灣水深，並搭配開放天氣資料建立 AI 預測模型，預測未來之港灣水深之變化，同時透過預測誤差判斷港灣內泥沙淤積情形，而這個理想的根基，即是將量測值與預測值做一個結合，也就是量測值需要與中心的資料庫作互動得到一個綜合出來的最佳水深預測值，以利船隻參考是否航行、停靠於所經港灣，故本次專題主要是在設計船隻與中心資料庫的通訊系統。

二、實驗器材

硬體設備：魚群探測儀(水深探測)、Raspberry Pi、Pi Camera、Arduino Uno、Beitian BN-880 GPS & Compass(GPS 模組)、筆電或桌電、Nvidia RTX 3060Ti(GPU)

軟體環境：Python3.8、Arduino、Pytorch、OpenCV

三、系統架構



圖一、系統架構[1][2][3][4][5]

這次的專題搭建了前面敘述的船隻與資料庫中心的通訊系統架構，以下分成船隻、資料庫中心、通訊。

a. 船隻：

船隻上會用到的裝置為 RPi、Pi Camera、魚群探測儀、Arduino、GPS Module，而各裝置的功能如下表。

裝置	RPi	Pi Camera	GPS Module	Arduino
功能	彙整各量測資訊並與資料庫中心通訊	拍攝魚群探測儀的畫面	讀取衛星信號並回傳給 Arduino	解讀衛星信號成經緯度並傳給 RPi

表一、系統個裝置功能

如表一所述，GPS Module 負責做定位，讀取衛星資料將訊號傳給 Arduino 解讀出經緯度資訊，再透過 Serial Port 將經緯度資料傳給 R Pi。而 Pi Camera 會將魚群探測儀的畫面拍下讓 R Pi 讀取，R Pi 會將 GPS 與圖片一起傳給資料庫中心最佳水深預測。

b. 資料庫中心：

資料庫中心為電腦機房，提供服務如下表所述：

	影像模型	水深模型	結合模型
功能	辨識圖像中的數字	根據資料庫預測未來水深	將影像模型與水深模型的結果作結合

表二、

RPi 傳送給資料庫中心的資料有圖片與經緯度座標，資料庫中心的電腦會先將圖片中的數字以影像模型辨識，將辨識的結果當作量測值，接著會根據經緯度座標從資料庫中抓出該地點的資料，透過水深模型預測出水深，最後會用結合模型將量測的水深與預測的水深做結合，輸出一個最佳的水深，並回傳給 RPi。

c. 通訊：

船隻與資料庫中心的通訊裝置是透過 TCP，因為需要傳送影像，故需要 Reliable transmit，而 TCP 是個很好的選擇。

四、實驗方案的討論

原本神通資訊的蘇經理建議我們使用 FW-SFCC 聲納探測儀(圖二)，此裝置有精準度的量測且有信號輸出，可以直接用 Arduino 讀取其信號，換算水深，但其裝置一個要價 25000，十分昂貴，故我們提出使用魚群探測儀當作量測裝置，魚群探測儀廣受釣客喜愛，可以探測出水深也可以找出魚群所在的深度，且又因釣客身處的環境，裝置有防水的設計，故很適合這次專題使用，可以使成本降低，但其缺點是沒有輸出原，只有顯示畫

面，如圖三所示，故若要使用魚群探測儀，就需要額外架設相機拍畫面並做數字辨識。



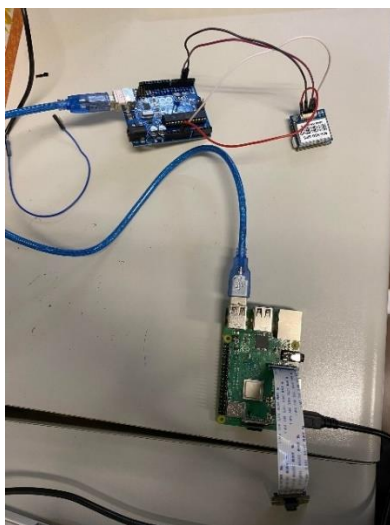
圖二、FW-SFCC[6]



圖三、魚群探測儀[7]

而我們因為經費考量，故決定用魚群探測儀並自己額外搭建視覺辨識模型，以節省成本。

五、裝置組裝實圖



圖四、實驗裝置實圖

實驗裝置實圖如圖四，此圖代表的部分為船上的量測裝置，魚群探測儀的圖片我們上網找圖用截圖的方式，故到最後也沒有買，實驗方式就是將 Pi Camera 拍電腦螢幕播放的魚群探測儀截圖。

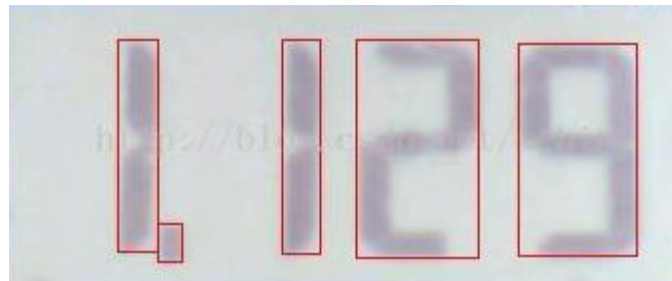
六、各裝置的實作

a. 影像模型：

因實驗裝置為相機拍攝魚群探測機的數字當作量測結果，故要建立一個視覺辨識模型，而我們在這裡有參考幾個模型：

1. 傳統視覺辨識方法：

傳統電腦視覺會先將影像作切割，如圖五所示，將數字一個個切出來，最後再根據數字 0~9 有的圖像特徵，如開口方向或數字是否有洞，從這些特徵來預測數字的類別。



圖五、影像數字分割[8]

但由於傳統電腦視覺的影像切割需要自行定義參數，參數的數量也不太多，故很難調整到一個很萬用的狀態，以至於辨識效果都不太好。

2. 深度學習模型：

深度學習應用於電腦視覺在 GPU 的平行運算能力越強後，廣泛被研究及應用，在很多任務上都有很好的表現，如影像分類、影像切割、物件偵測等，且網路上已有很多的開源程式可以下載來使用，也有許多前人訓練好的模型可以用，故深度學習是個很好方法來做數字辨識。

而深度學習的模型，在這邊嘗試了以下的模型：

i. CNN：

CNN 為影像辨識最常用的模型，因具有擷取區塊性的特徵，因此在影像辨識上會有很好的表現。

ii. ResNet34：

ResNet[9]為現今很常用的深度學習模型，是基於 CNN 改良出的模型，其特點在於資料丟入 CNN 前會先保留，並與 CNN 後的結果相加，此機制稱為 Residual，如圖六所示，經實驗證實，能夠有更好的表現，且其模型參數量較少，故很適合作需要及時辨識的任務。

iii. ResNet50：

ResNet50 為前面提到的 ResNet34 中 CNN 層數更深的模型，效果會更好但只有多 10MB 左右的參數。

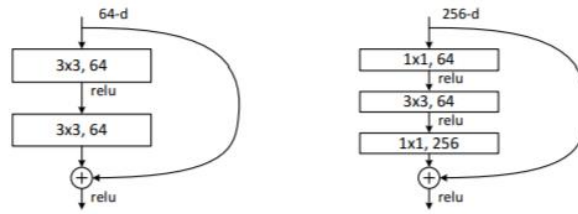


Figure 5. A deeper residual function \mathcal{F} for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a “bottleneck” building block for ResNet-50/101/152.

圖六、Residual 機制

而因 ResNet 的效果比一般 CNN 好，且 ResNet50 的參數量只比 ResNet34 多 10MB 左右的大小，故這裡以 ResNet50 當作訓練模型，並自己上往截圖，如圖七所示，來當作訓練資料。



圖七、訓練資料

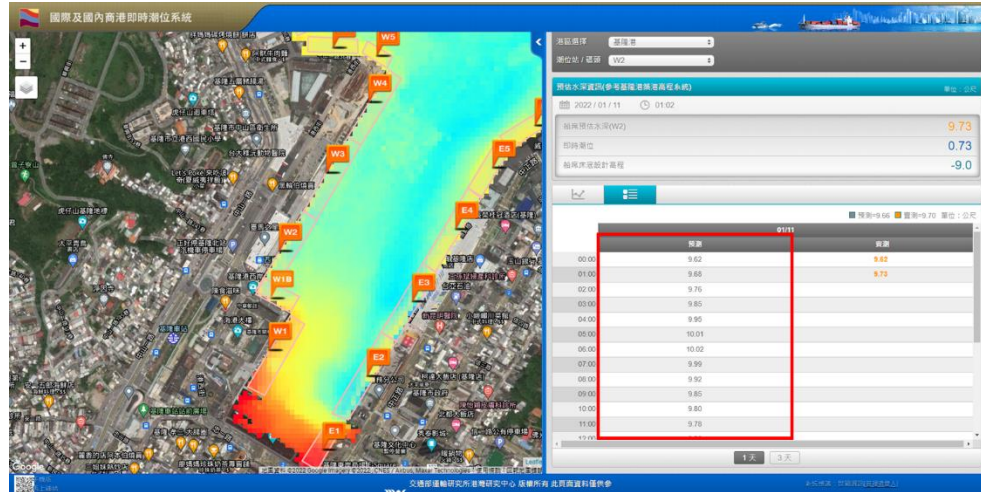
但由於需要從一張圖片中，讀取出數個數字，這與一般的手寫數字辨識不同，一般的手寫數字辨識是一個分類的任務，要分的類別為 0~9，可以使用 cross-entropy loss 來訓練，但這邊很難這樣做，因這樣又要訓練另一個模型來做數字切割，故這邊使用 Mean Square Error Loss 來做訓練，使模型輸出一個數字，再用這個數字跟 Label 做差異的計算來更新模型參數。

而 ResNet 的實作，這裡使用 Pytorch[10]的 torchvision model zoo 中的 pretrained resnet50 來訓練。

b. 水深模型：

水深模型需要有港灣水深的資料，以利進行機器學習分析，這個部分神通資訊說是她們會做的任務，故這邊並沒有自己實作這塊，而為了要有一個有預測港灣水深的數值，這裡實作一個 python 爬蟲，從基隆港的官方網站將預測水深的資料抓下來，如圖八中的紅框所示，紅框內的資料為不同時間點的預測水深。

而這邊的爬蟲是使用 Python 的 requests[11]和 BeautifulSoup[12]套件來實作，因裡面的資料都有自己的網址，故只需使用 Http Post 將所需資料的網址送出去，即可得到網站的回應，而回應可以用 BeautifulSoup 來讀取轉換成所需的格式。



圖八、基隆港水深資料[13]

c. 結合模型：

結合模型的目的是要能將量測的水深與預測的水深做結合並輸出一個最佳的預測水深，這裡的作法是實作出一個 Kalman Filter[14]，其數學原理如下[15]：

$$X[k|k-1] = F \cdot X[k-1|k-1] + B \cdot U[k] \quad (1)$$

$$P[k|k-1] = F \cdot P[k-1|k-1] \cdot F^T + Q \quad (2)$$

$$X[k|k] = X[k|k-1] + Kg[k](Z[k] - H \cdot X[k|k-1]) \quad (3)$$

$$Kg[k] = \frac{P[k|k-1]H^T}{H \cdot P[k|k-1]H^T + R} \quad (4)$$

$$P[k|k] = (1 - Kg[k]H)P[k|k-1] \quad (5)$$

上面 5 條方程式中 X 為預測狀態、 F 狀態轉移矩陣、 U 為狀態輸入、 B 為狀態輸入的轉移矩陣、 P 為 X 的 covariance matrix、 Q 為預測狀態的變異數、 H 狀態轉成測量系統的轉移矩陣、 Z 為量測狀態、 R 為測量狀態的變異數、 Kg 為 Kalman Gain，而這 5 條式子分別代表的意義為

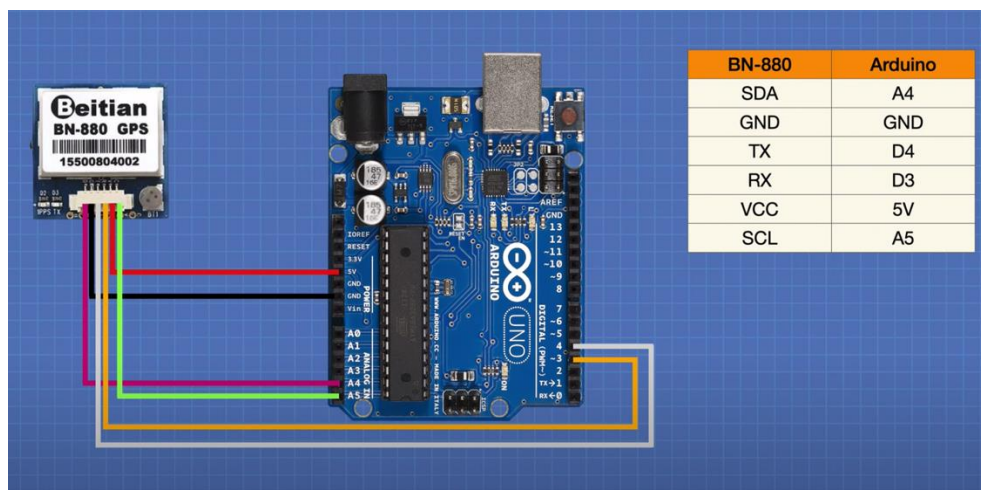
1. 利用 $k-1$ 的狀態，加上變化量，預估出 k 狀態。
2. 利用 $k-1$ 狀態的雜訊，配合預估 k 狀態的雜訊，預估出 k 狀態的雜訊。
3. 利用預估出的 k 狀態與 k 的測量狀態搭配權重值(Kalman Gain)完成更新 k 狀態。
4. 權重值(Kalman Gain)是 k 狀態的雜訊與測量雜訊 經計算而求得。
5. 最後利用權重值(Kalman Gain)更新 k 狀態的雜訊。

而這邊因為資料只有一維，故上述參數都是純量而非矩陣，且測量和狀態的數值都沒有經過運算，故 $F = 1$ 、 $H = 1$ ，而其餘的參數接為預測和測量的狀態變異數，需要經過統計而決定，故這邊就直接設定為： $P = 3$ 、 $Q = 4$ 、 $R = 20$ ，因測量較易有誤差，故將測量狀態變異

數R設定的比較大。

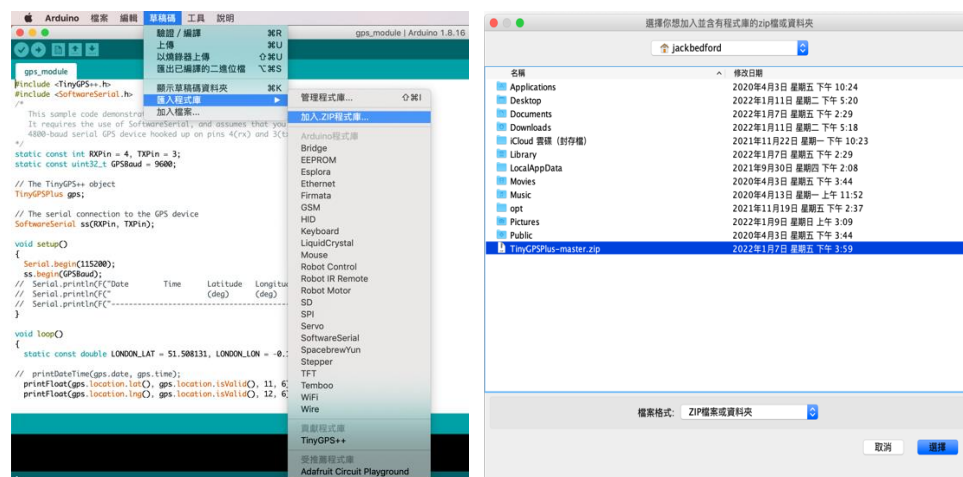
d. GPS

我們首先將 bn-880 與 Arduino UNO 組裝起來如圖九。其中，我們只需要連接 5V 電源、接地、Tx、Rx 四條線，另外兩條 analog 的 I/O 不需要用到。



圖九、BN-880 組裝圖

而在 Arduino 的控制代碼中，我們需要使用到 TinyGPS++ 的 Library，因此要先從以下連結 <https://github.com/mikalhart/TinyGPSPlus> 將 zip 檔下載下來，再將 Library 導入 Arduino 如圖十。



圖十、下載 TinyGPS++ Library

接著，我們修改範例代碼，將 GPS baud 設定為 9600，serial baud 設定為 115200，將 bn-880 所收到的資訊中，提取日期、時間、經度、緯度四個資訊，輸出到 serial monitor 中。最後，再寫一份 python code，利用 serial module，將 Arduino 的 serial output 讀進來，並 parse 出定位用的經緯度。

其中，值得一提的是，Arduino 代碼所設定的 delay 不能設置得太長，否則會嚴重影響到整個系統的效能，照相的功能和預測模型都會

變得相當遲鈍，甚至卡住。但也要設定一個足夠大的 **delay**（約為 0.05 秒），讓 **python** 較穩定地讀取 **serial output**，否則可能會出現難以預期的結果，導致 **parsing** 時無法順利轉換資料型態。

七、結果

這裡分成三個部分呈現，分別為資料庫中心介面(個人的筆電)、**RPi** 介面、**Arduino** 介面：

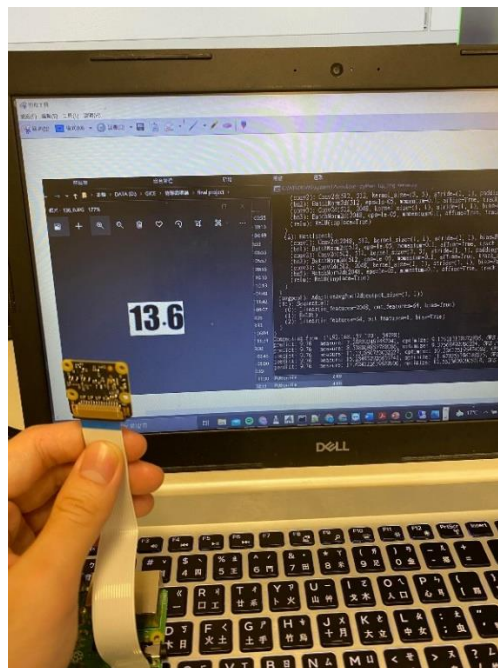
a. 資料庫中心介面：

對於數字辨識模型，我們用 **Pi Camera** 拍了大約 500 張的數字截圖，如圖十一，將這些圖片當作訓練資料丟入 **ResNet50** 進行訓練，得到的 **MSE** 可以為 8，也就是訓練資料上平均會有 $\pm\sqrt{8}$ 的誤差，這個結果算是令人滿意的結果。



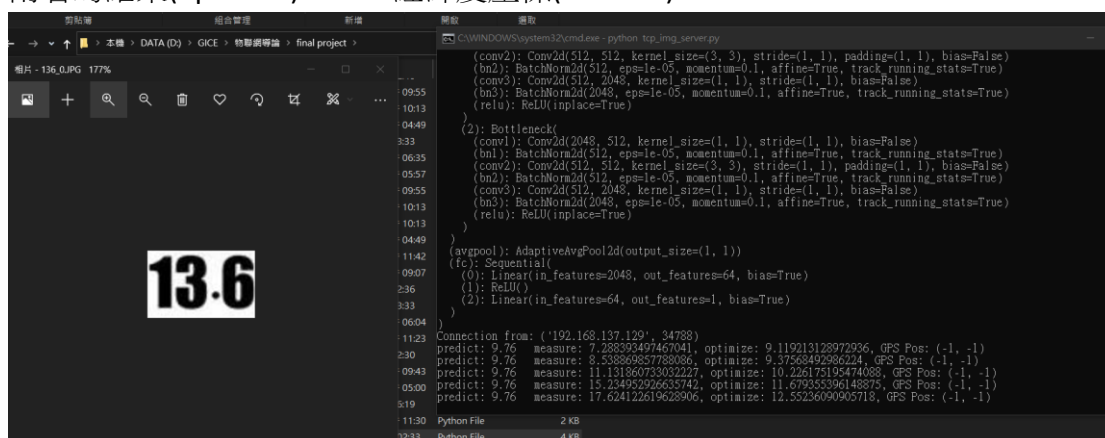
圖十一、訓練資料收集

而實測時，會用 **Pi Camera** 拍攝於螢幕上的數字截圖，如圖十二所示。



圖十二、實際測試實圖

而在筆電 CMD 上，如圖十三，出現的數字分別為從基隆港抓出來的預測資料(predict)、影像辨識出來的結果(measure)、Kalman filter 整合兩者的結果(optimize)、GPS 經緯度座標(GPS Pos)。

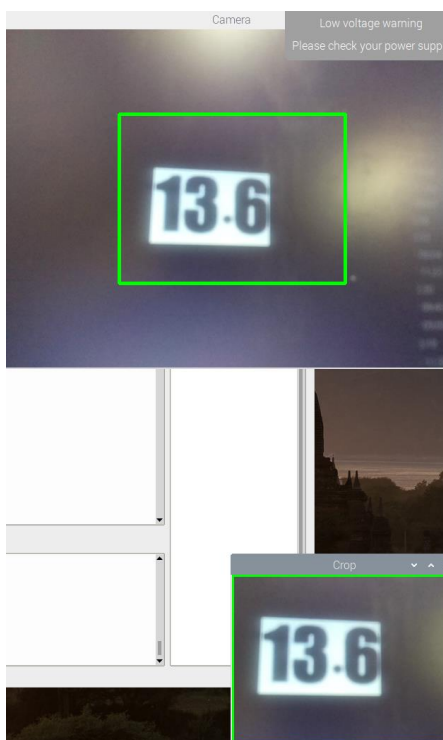


圖十三、筆電 CMD 截圖

可以看到視覺模型預測的結果還算準，且 Kalman Filter 也確實有做到整合兩者資訊的能力，而 GPS 座標皆為-1 是在室內拍攝，收不到 GPS 信號。

b. RPi 介面：

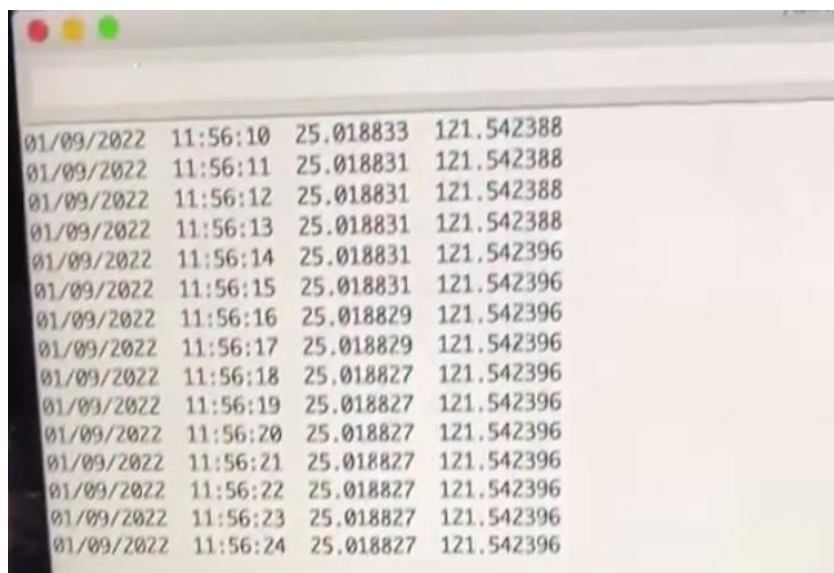
RPi 的介面會出現一個視訊視窗，視窗內有一個綠色方框，如圖十四所示，此方框是 Pi Camera 會截取的部分，故要將欲辨識的數字對準方框，方能將影像傳回筆電。



圖十四、RPi 介面

c. GPS

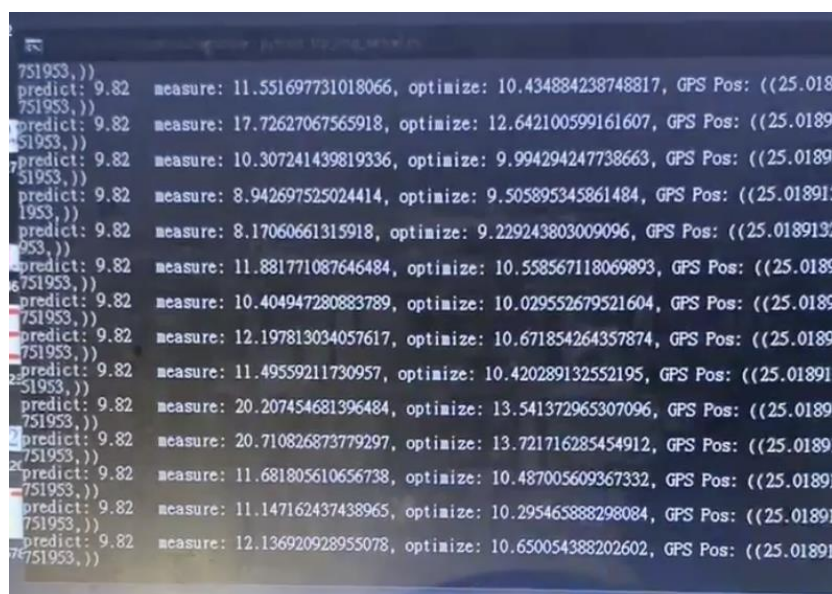
由於 **bn-880** 必須在戶外，而且需要在不錯的天氣條件下，才能收到 **GPS** 訊號，因此難以在教室中進行展示。實際跑出來的結果如圖十五所示，我們在測試的 **GPS** 訊號時，先設定每隔 1 秒輸出一次收到的訊號。由於拍攝影片時不方便移動電腦，因此經緯度幾乎是不動的。



01/09/2022	11:56:10	25.018833	121.542388
01/09/2022	11:56:11	25.018831	121.542388
01/09/2022	11:56:12	25.018831	121.542388
01/09/2022	11:56:13	25.018831	121.542388
01/09/2022	11:56:14	25.018831	121.542396
01/09/2022	11:56:15	25.018831	121.542396
01/09/2022	11:56:16	25.018829	121.542396
01/09/2022	11:56:17	25.018829	121.542396
01/09/2022	11:56:18	25.018827	121.542396
01/09/2022	11:56:19	25.018827	121.542396
01/09/2022	11:56:20	25.018827	121.542396
01/09/2022	11:56:21	25.018827	121.542396
01/09/2022	11:56:22	25.018827	121.542396
01/09/2022	11:56:23	25.018827	121.542396
01/09/2022	11:56:24	25.018827	121.542396

圖十五

最終，我們會將 **GPS** 的收發模組，整合到系統架構之中，輸出的結果如圖十六。可以看到模組成功收到 **GPS** 訊號，並將座標值標註在最後兩個欄位。



751953,))	predict: 9.82	measure: 11.551697731018066,	optimize: 10.434884238748817,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 17.72627067565918,	optimize: 12.642100599161607,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 10.307241439819336,	optimize: 9.994294247738663,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 8.942697525024414,	optimize: 9.505895345861484,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 8.17060661315918,	optimize: 9.229243803009096,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 11.881771087646484,	optimize: 10.558567118069893,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 10.404947280883789,	optimize: 10.029552679521604,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 12.197813034057617,	optimize: 10.671854264357874,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 11.49559211730957,	optimize: 10.420289132552195,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 20.207454681396484,	optimize: 13.541372965307096,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 20.710826873779297,	optimize: 13.721716285454912,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 11.681805610656738,	optimize: 10.487005609367332,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 11.147162437438965,	optimize: 10.295465888298084,	GPS Pos: ((25.0189
751953,))	predict: 9.82	measure: 12.136920928955078,	optimize: 10.650054388202602,	GPS Pos: ((25.0189

圖十六

八、未來展望：

a. 視覺模型：

首先是關於視覺辨識模型，目前的訓練資料實在是太少了，故還是有很多時候辨識誤差會很大，故希望若此系統要落地，必定需要廣泛蒐集訓練資料，且需要考量多種不同的變化，如圖片旋轉、光線變化、數字的樣式等，必須要將這些變化的資料都蒐集完全，才可能訓練出一個完整的模型。

b. 聲納系統：

第二是聲納模組的部分，因為成本因素，才選擇使用魚群探測儀搭配數字辨識系統，但若使用聲納模組，如此一來量測結果必定能夠更精準，會有更好的結果。

c. 水深模型：

我們很期待神通資訊能夠完整建立出一個根據港灣水深預測的機器學習模型，若使用到此模型，這次的專題完整度會更好。

d. 搭建 UI：

這次因為時間緣故，並沒有使用上課教的 Node Red 架設 UI，故若有機會希望可以補上這一塊。

九、參考資料

- [1] <https://www.ruten.com.tw/item/show?21401298636653>
- [2] <https://www.roxtec.com/en/industries/infrastructure/data-centers/>
- [3] <https://shopee.tw/-3DPW-Arduino-UNO-R3-%E6%8E%A7%E5%88%B6%E6%9D%BF-%E5%8E%9F%E5%BB%A0%E6%AD%A3%E7%89%88%E6%99%B6%E7%89%87-%E9%9D%9E%E6%89%80%E8%AC%82%E7%9B%B8%E5%AE%B9%E7%89%88-%E5%90%ABUSB%E7%B7%9A-3D%E5%8D%B0%E8%A1%A8%E6%A9%9F-i.168569720.2684465342>
- [4] <https://www.compresdairpa.co/%E9%AD%9A%E7%BE%A4%E6%8E%A2%E6%B8%AC%E5%99%A8-%E6%8E%A2%E9%AD%9A%E5%99%A8/>
- [5] <https://shopee.tw/BN-880PIX4%E7%84%A1%E4%BA%BA%E6%A9%9FGLONASS%E9%9B%99%E6%A8%A1GPS%E6%A8%A1%E5%A1%8AAPM-HMC5883%E7%A3%81%E7%BE%85%E7%9B%A4%E5%9C%B0%E7%A3%815983-i.464502191.6592516334>
- [6] <https://www.ruten.com.tw/item/show?22145932461199>
- [7] <https://www.pchomeus.com/item/show?30214234687606%E7%9A%84>
- [8] <https://www.twblogs.net/a/5c39d86abd9eee35b3a5e9c7>
- [9] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

- [10] <https://pytorch.org/>
- [11] <https://docs.python-requests.org/en/latest/>
- [12] <https://www.crummy.com/software/BeautifulSoup/>
- [13] <https://tide.twport.com.tw/maps/?area=kl>
- [14] <https://zh.wikipedia.org/wiki/%E5%8D%A1%E5%B0%94%E6%9B%BC%E6%BB%A4%E6%B3%A2>
- [15] <https://silverwind1982.pixnet.net/blog/post/167680859>

十、附錄

程式碼：<https://github.com/TingWeiHsuTW/Intro-to-IOT.git>