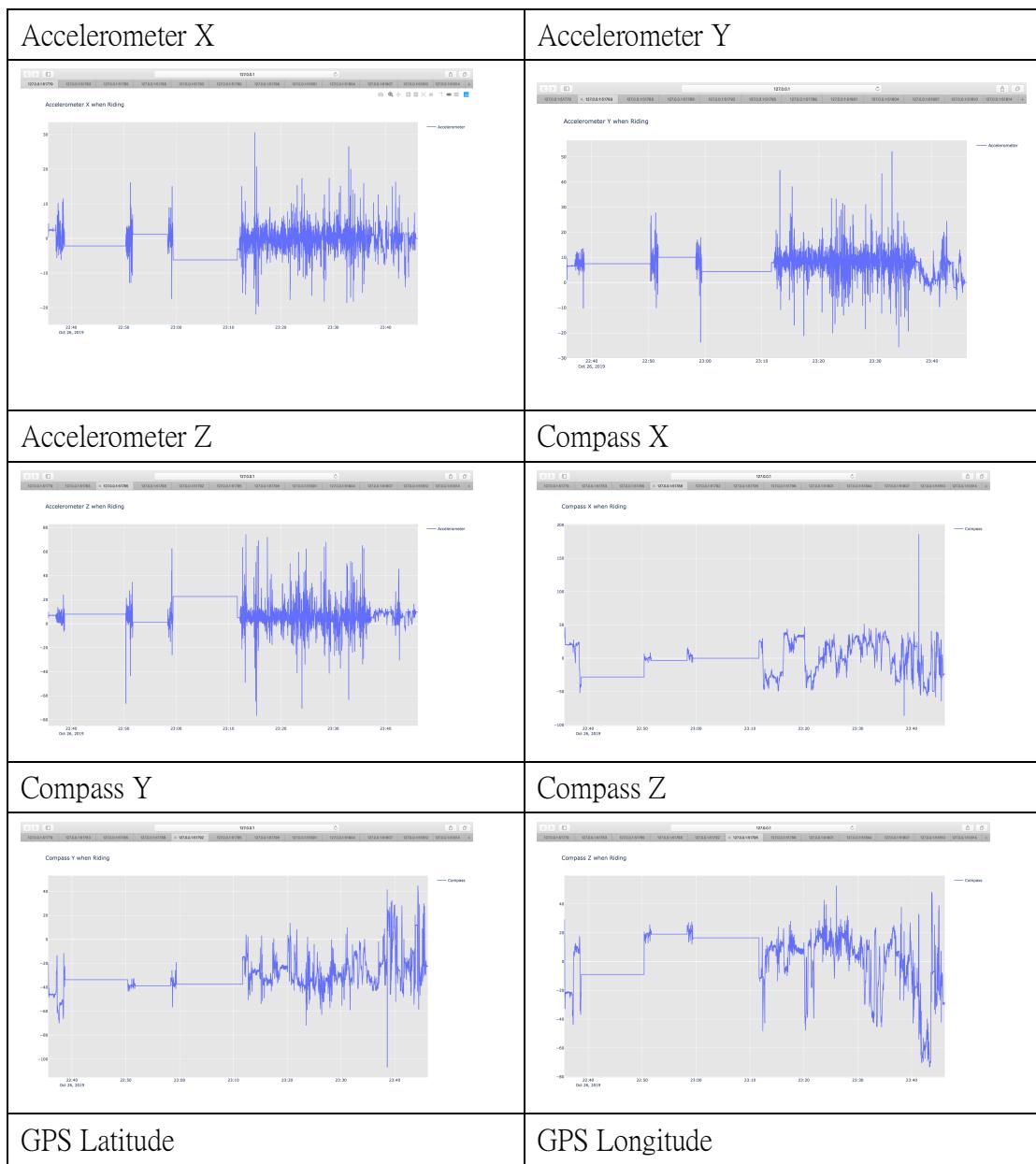


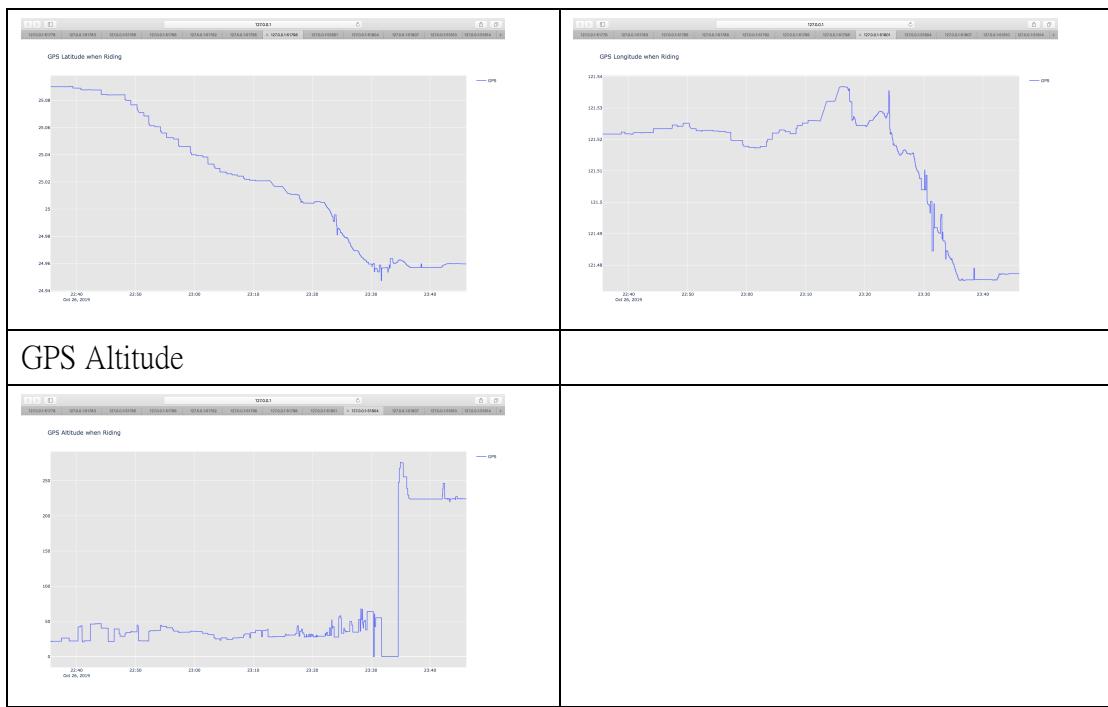
HW1_2019_part-B

(b)

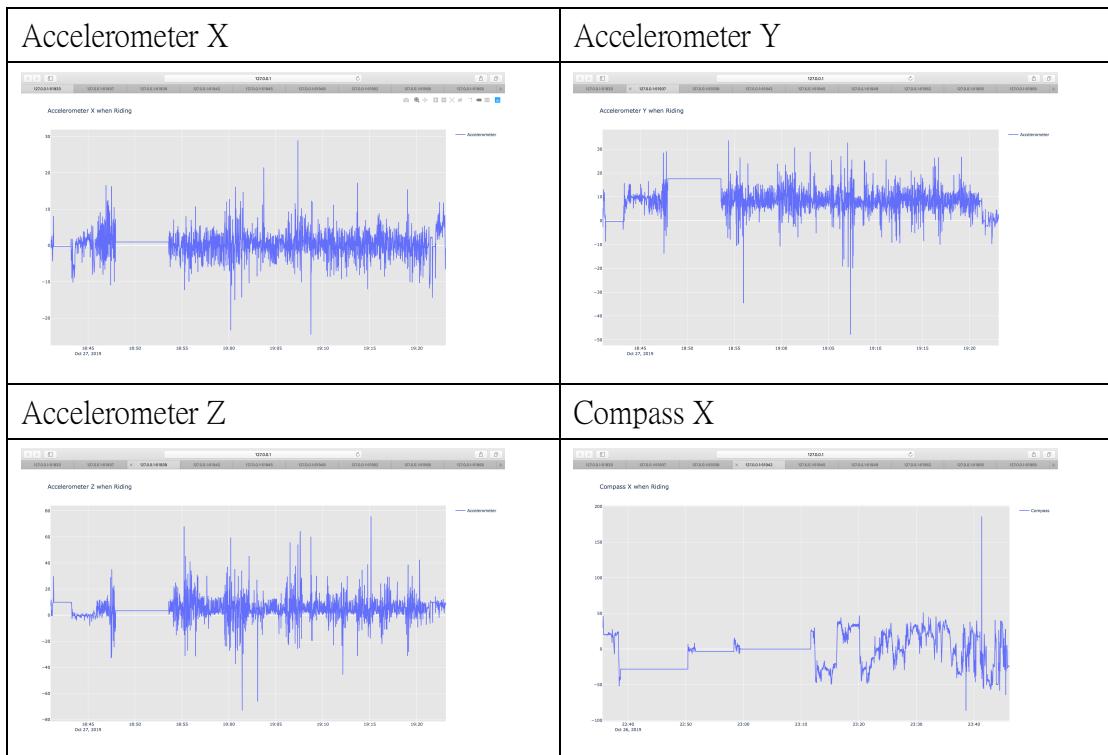
由於我在搜集資料時，一直發生中斷，不然就是在拿到資料在查看時，才發現測量的 sensor 的數值，太不合理，所以我只拿其中算是比較合理可用的 data 來使用。設定 sensor 為每一秒紀錄一次。

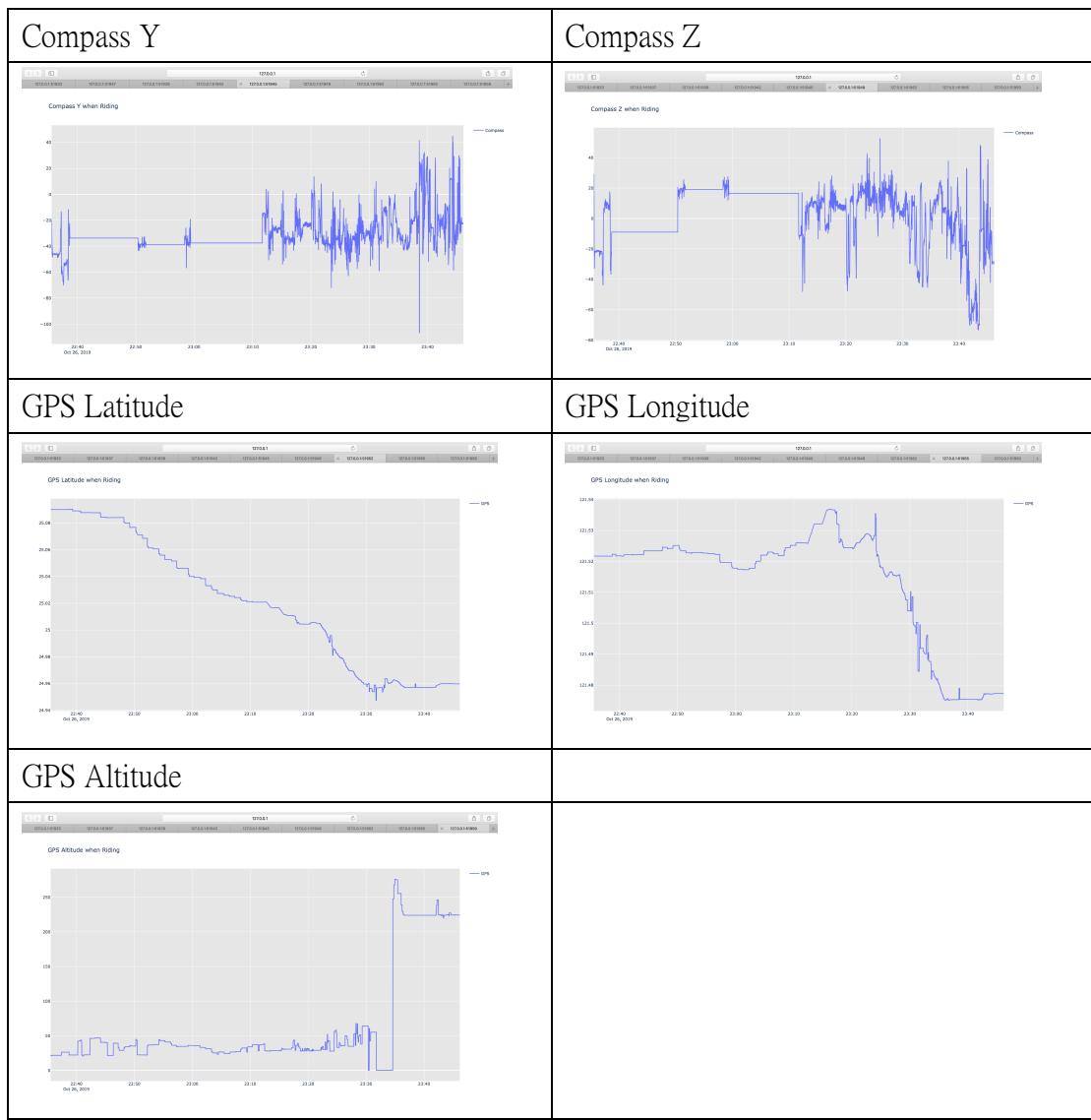
2019-10-26_22-35-35-Riding



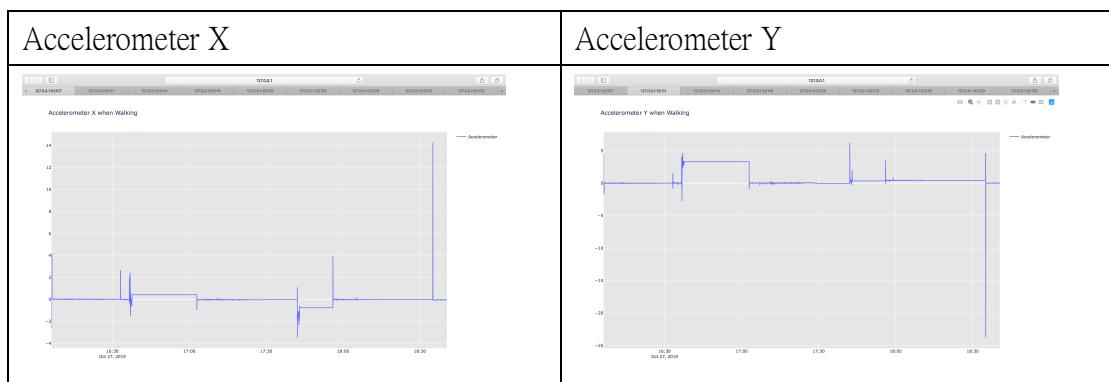


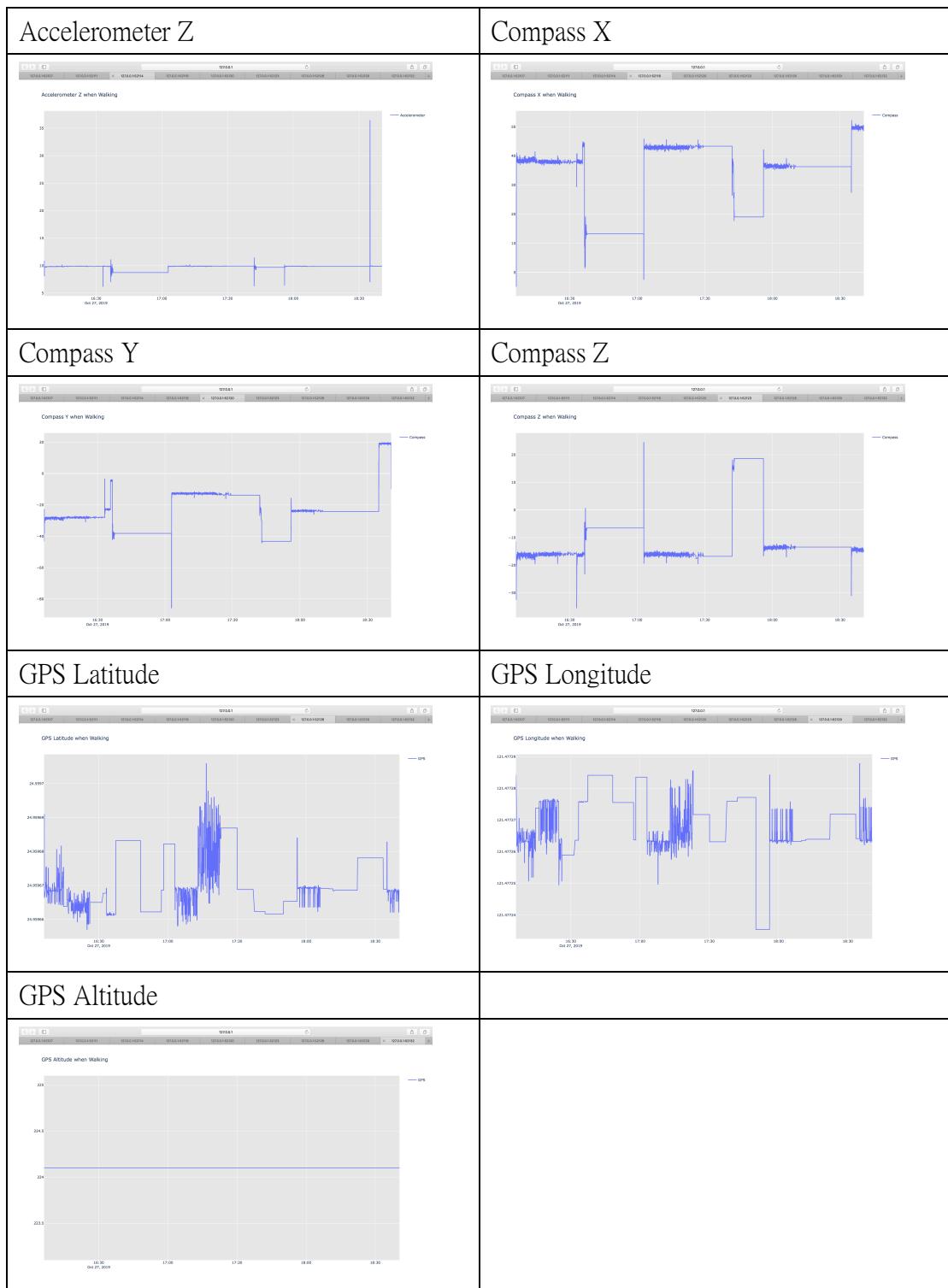
2019-10-27_18-41-01-Riding



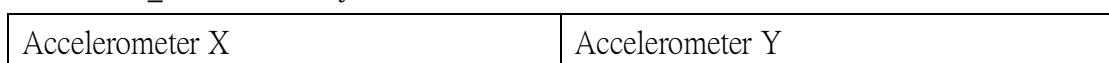


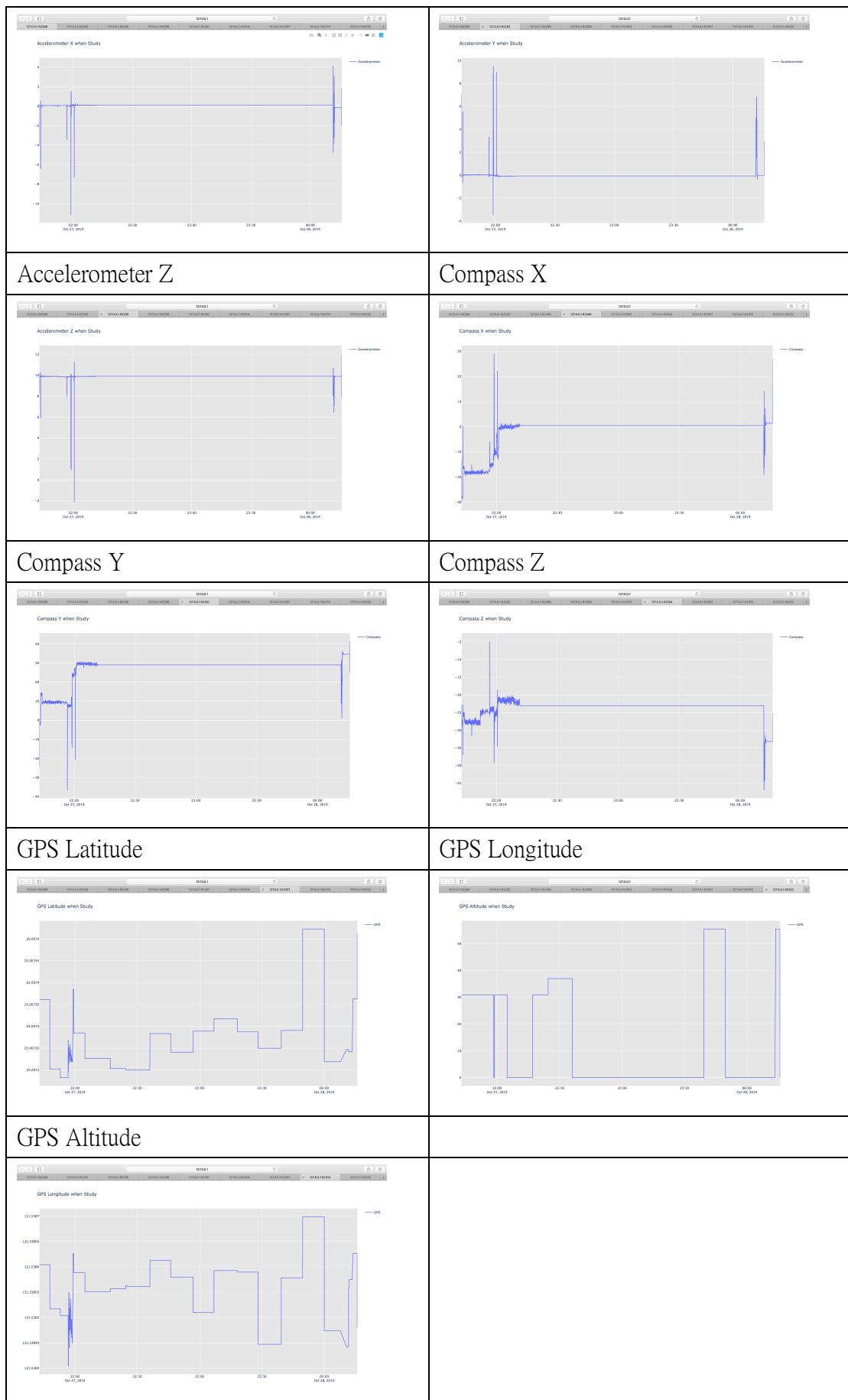
2019-10-27_16-06-15-Walk



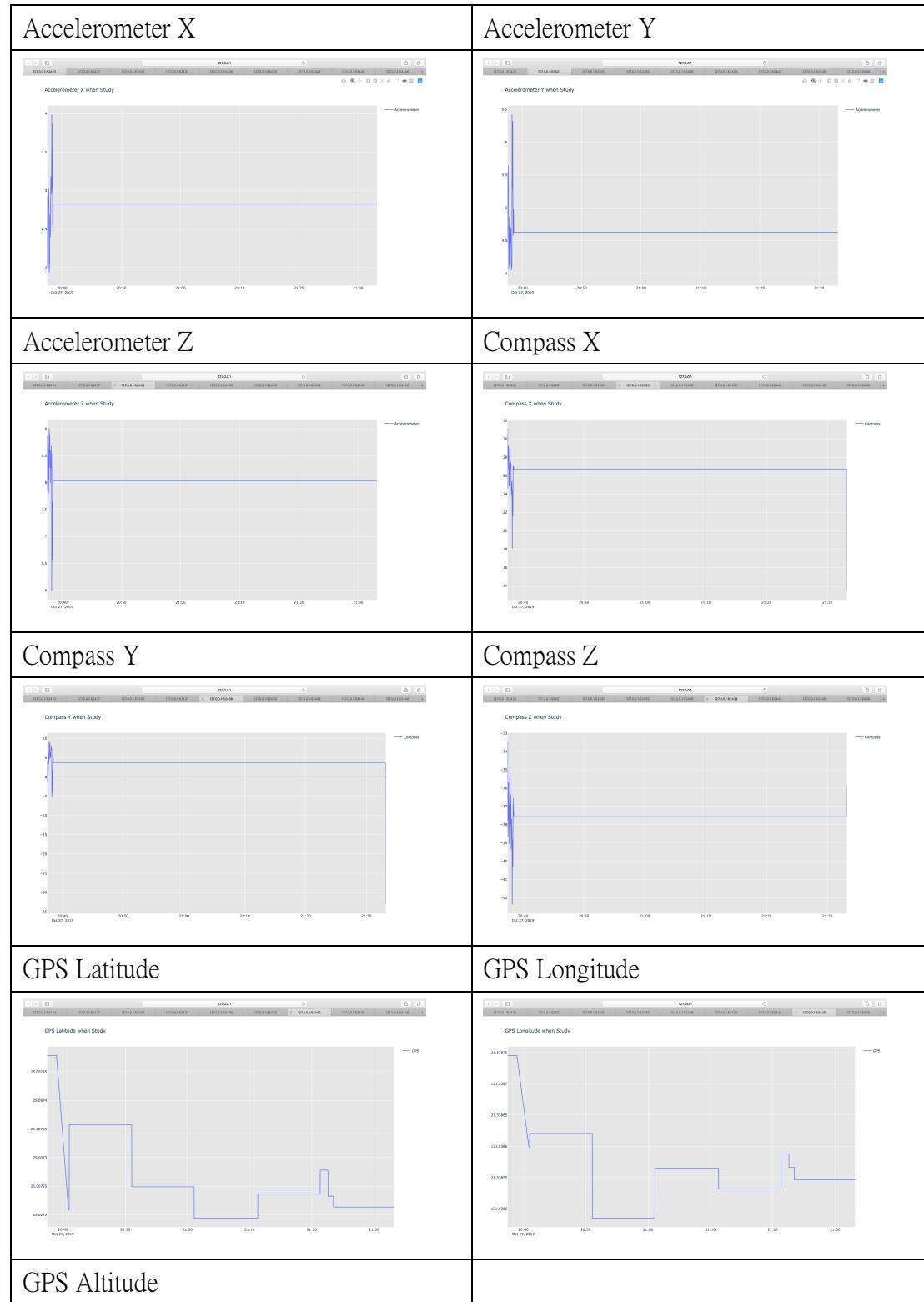


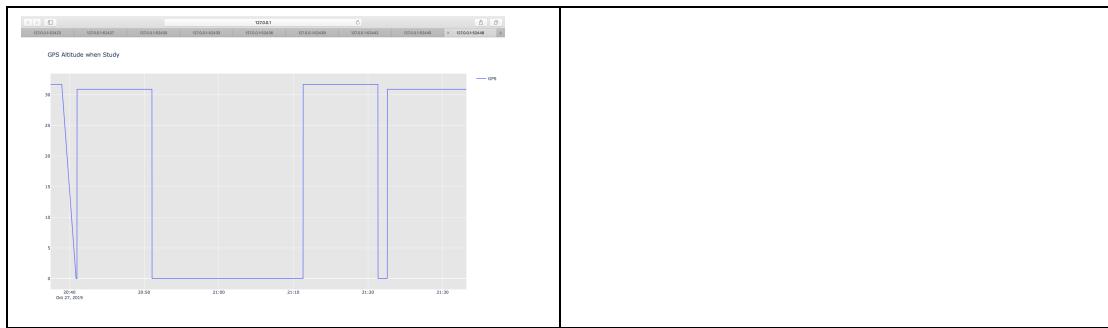
2019-10-27_21-42-51-Study





2019-10-27_20-37-27-Study





(c)

這邊我只使用我認為較有判斷價值的加速度。我先將 Ride, Study 和 walk 的 csv 各自建模型。建立模型方法是使用 sliding window 計算其標準差，window size 為 50，然後再取所有標準差的平均值。由於算完後的值有加速度的 X,Y,Z 三種，而 Y 軸的加速度是，Ride,Study 和 walk 中差異最大的，所以我採用 Y 作為這個模型的判定基準數值。

之後讀入要做判斷的 csv 檔，我們用一樣的方法，算輸入檔案的標準差平均值，然後再利用我們先前模型建立出的判定基準，來判斷說兩者相差最少的，就為該類別。

結果：

用來測試的資料中，有未參與建模的 dataset ,Riding2 和 Studing2。測試後，結果都為正確，並顯示內部運算的 error rate 。

```
shungfu@shungfude-MBP:~/Desktop/IOT$ python3 predict.py Riding1_Acc.csv
ride
Error rate: 0.08313651393481342 352/4234
shungfu@shungfude-MBP:~/Desktop/IOT$ python3 predict.py Riding1_Acc.csv
ride
Error rate: 0.08313651393481342
shungfu@shungfude-MBP:~/Desktop/IOT$ python3 predict.py Riding2_Acc.csv
ride
Error rate: 0.1163895486935867
shungfu@shungfude-MBP:~/Desktop/IOT$ python3 predict.py Walk_Acc.csv
walk
Error rate: 0.6351555747623163
shungfu@shungfude-MBP:~/Desktop/IOT$ python3 predict.py Study1_Acc.csv
study
Error rate: 0.006422817330720662
shungfu@shungfude-MBP:~/Desktop/IOT$ python3 predict.py Study2_Acc.csv
study
Error rate: 1.0
```

雖然說結果和每一欄的判斷結果不同的error rate 高達 1.0，但是由於我是用 sliding window 計算標準差，之後運算平均值。再拿來判斷，所以我依然能夠預測正確。

Python code:

https://hackmd.io/@L5UnFuBVQ12dUMp2E_zOw/Hy-gNCu_r

(f)

由於這邊需要將有標上 label 的檔案分群，所以我使用 knn 演算法來實作。knn 屬於監督式的學習，是一種 classification。再開始訓練模型之前，我們需要把原本分散的 csv 結合成一個，並為他們標上 label。(這邊我標記 0: riding motorcycle, 1: walking, 2: studying)

我們把上面結合好的資料做打亂，這樣我們訓練的結果才能更好。再把標好的 label 和原本的資料及切開，分為 data 和 target。這樣在訓練的時候，我們的判斷結果才不會被視為一種 attribute。接著再將 data 和 target 依照你的喜好比例，把他們都切成 test 和 train。

做好上述切割後，我們就可以匯入模型了。這邊我直接使用 sklearn 寫好的套件。`from sklearn.neighbors import KNeighborsClassifier`。把 test data 和 test target 丟入 fit function 裡面，模型就做好了。如果你想要看看正確率如何的話，你可以使用 score function，這邊的參數就要放你剛剛切割的 train data 和 train target 丟入，如果你還是丟 test 的話就會失去參考性了，因為你的模型和測試資料太過吻合了。

Knn models

```
In [43]: from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(data_train, target_train.values.ravel())

Out[43]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                               weights='uniform')
```

Eval Model

```
In [46]: print("Test Accuracy:", knn.score(data_test, target_test))

Test Accuracy: 0.9880896777206912
```

(準確率)

```
Data:  
[ 0.08140534 -0.07422252  9.90272 ]  
Real:  
2  
Predict:  
2  
  
Data:  
[2.4637089 6.5746784 7.0080423]  
Real:  
0  
Predict:  
0  
  
Data:  
[2.8252442 4.625739  8.0375805]  
Real:  
2  
Predict:  
2  
  
Data:  
[-0.0670397  0.08619389  9.895537 ]  
Real:  
1  
Predict:  
1  
  
Data:  
[-0.0287313  0.43815228  9.888355 ]  
Real:  
1  
Predict:  
1
```

(測試判斷)

Python code:

https://hackmd.io/@L5UnFuBVQ12dUMp2E_zOw/S1d-YpK9r