Engineering Assignment Coversheet

Please note that you:

- Must keep a full copy of your submission for this assignment
- Must staple this assignment
- Must NOT use binders or plastic folders except for large assignments

Student Number(s)	671116
Group Code (if applicable):	

Assignment Title:	Project 3: Fault detection & data fusion.
Subject Number:	MCEN90032
Subject Name:	Sensor Systems
Student Name:	Akshay Kumar Bharat Kumar
Lecturer/Tutor:	Mitchell Khoo
Due Date:	21/10/18

For Late Assignments Only

Has an extension been granted? Yes / No (circle)

A per-day late penalty may apply if you submit this assignment after the due date/extension. Please check with your Department/coordinator for further information.

Plagiarism

Plagiarism is the act of representing as one's own original work the creative works of another, without appropriate acknowledgment of the author or source.

Collusion

Collusion is the presentation by a student of an assignment as his or her own which is in fact the result in whole or in part of unauthorised collaboration with another person or persons. Collusion involves the cooperation of two or more students in plagiarism or other forms of academic misconduct.

Both collusion and plagiarism can occur in group work. For examples of plagiarism, collusion and academic misconduct in group work please see the University's policy on Academic Honesty and Plagiarism: http://academichonesty.unimelb.edu.au/

Plagiarism and collusion constitute cheating. Disciplinary action will be taken against students who engage in plagiarism and collusion as outlined in University policy. Proven involvement in plagiarism or collusion may be recorded on my academic file in accordance with Statute 13.1.18.

STUDENT DECLARATION

Please sign below to indicate that you understand the following statements:

I declare that:

- This assignment is my own original work, except where I have appropriately cited the original source.
- This assignment has not previously been submitted for assessment in this or any other subject.

For the purposes of assessment, I give the assessor of this assignment the permission to:

- Reproduce this assignment and provide a copy to another member of staff; and
- Take steps to authenticate the assignment, including communicating a copy of this assignment to a checking service (which may retain a copy of the assignment on its database for future plagiarism checking).

Student signature [illegible] Date 21/10/18

MCEN90032 Sensor Systems 2018:

Project 3: Fault Detection and Data Fusion

Project Completed by:

Akshay Kumar Bharat Kumar, 671116

Due by:

21st October 2018

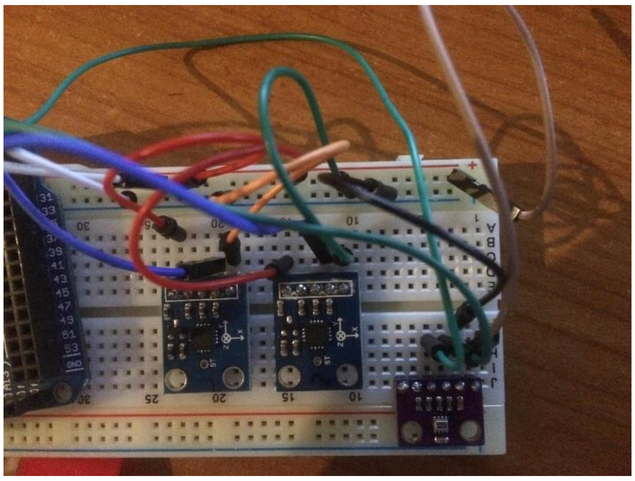
A.) Introduction:

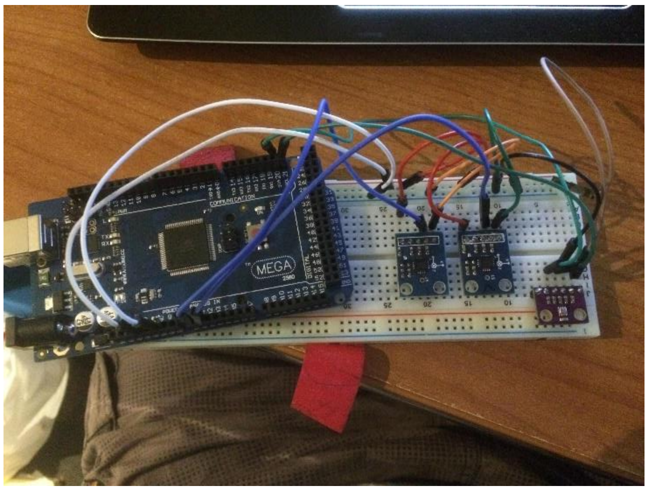
With the previous 2 projects displaying how important sensors are and how they can be effectively managed and corrected even if they have noise, the sensors still lack additional accuracy. The altimeter that had previously been designed with the Kalman filter is not very robust, especially if either pressure sensor or accelerometer were to fail (power loss or input loss). As such, this 3rd project was setup to detect a fault (or multiple faults) with the system, in addition to adding in an additional accelerometer to improve the accuracy and redundancy of the altimeter.

As such, the project was divided into 3 major parts:

1.) Fault Detection
    a. This involved collecting the data from the 2 accelerometers and the pressure sensor and being able to identify when a fault occurred. This fault could be either from an input loss or a power loss.
    b. The system should be able to inform the user of the problem, while still being able to operate under normal conditions.
2.) Data Fusion and Kalman Filter (simulation)
    a. This involved determining a new state space model for the 2 accelerometer inputs and pressure sensor input.
    b. In addition to this, the system would have white noise, simulating a fault within the system.
    c. This section was to find the discrete time state space model for this system and subsequently create a Kalman Filter much like Project 2.
    d. This was to be done via simulations initially, which were provided in a Simulink model for initial testing.
3.) Data Fusion and Kalman Filter (Arduino Implementation)
    a. Following the Kalman filter design and test, the Arduino was to be programmed to perform a live and real-time Kalman filter implementation.
    b. The system was to be able to operate nominally even if there was a fault in the system (such as input or power loss)

The images below are of the Arduino and sensors setup and wiring for Project 3.





B.) Part 1: Fault Detection

Following on from the previous 2 projects, the accelerometer and pressure sensor had been used and been calibrated before. With the introduction of a new accelerometer, this was calibrated and subsequently coded into ArduinoIDE.

Data was then collected using all 3 sensors. The breadboard and Arduino were left on a flat surface. At a random point during the data collection, a wire was removed from the power input of one Arduino and then placed back. The input pin for the Z axis was then removed and then replaced. This was subsequently repeated for the second accelerometer. The graphs below show the data which was collected during this time for both accelerometers.

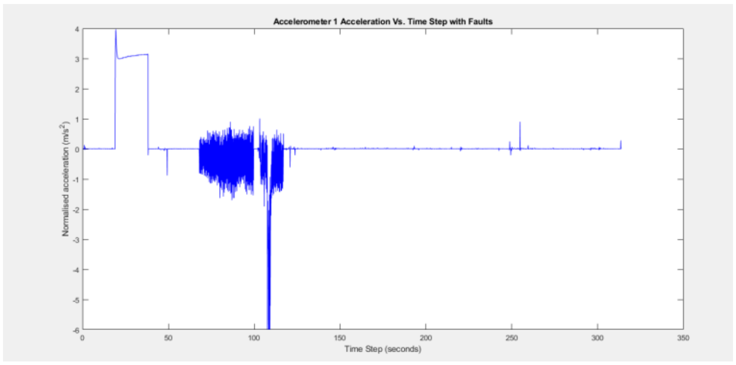


Figure 1: Data being collected from the 1st accelerometer as the power and input pins were removed

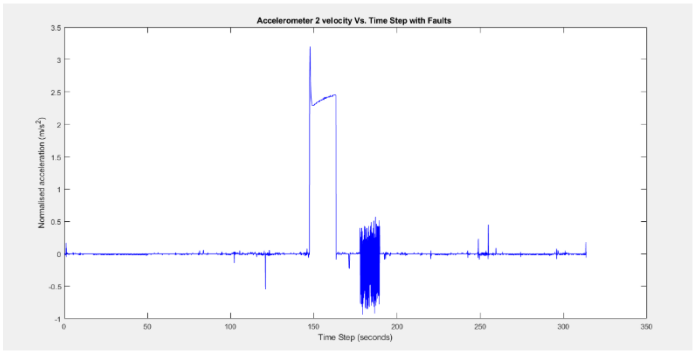
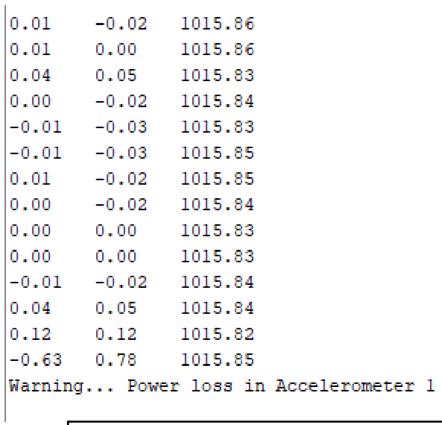


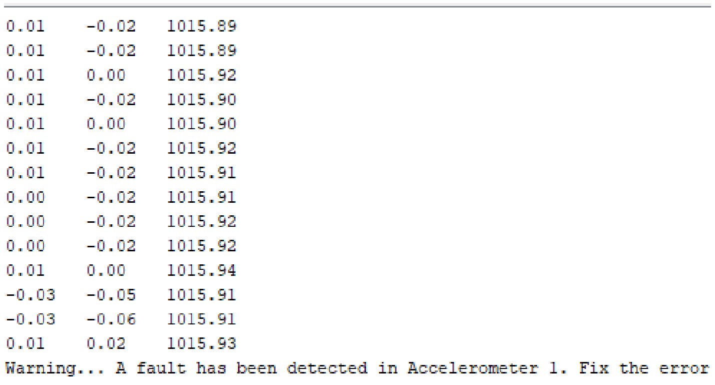
Figure 2: Data being collected from the 2nd accelerometer as the power and input pins were removed

The huge spikes in the beginning of each graph represent when the power supply was removed from the accelerometer. The hairy (high frequency noise) which is seen a few time steps later shows when the input pin was removed from the accelerometer.

From these graphs, a simple, yet effectively algorithm was devised to account for faults in the system. To account for power loss, if the readings spiked greater than 4, it would trigger a fault warning. Similarly, for the second accelerometer, if the readings spiked above 3, a fault would be registered. In the case for input loss, the optimal algorithm which was found was that if between any 2 subsequent data readings, the normalised acceleration exceeded 1 m/s^2, there was a fault present. This was true for both accelerometers. Following this, implementing these algorithms into the Arduino IDE allowed for easy identification of errors. The following snapshots are from the Serial Monitor upon removing the respective power and input wires from both accelerometers.



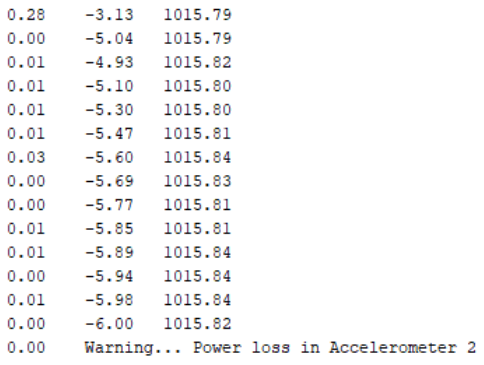
```
0.01    -0.02   1015.86
0.01    0.00    1015.86
0.04    0.05    1015.83
0.00    -0.02   1015.84
-0.01   -0.03   1015.83
-0.01   -0.03   1015.85
0.01    -0.02   1015.85
0.00    -0.02   1015.84
0.00    0.00    1015.83
0.00    0.00    1015.83
-0.01   -0.02   1015.84
0.04    0.05    1015.84
0.12    0.12    1015.82
-0.63   0.78    1015.85
Warning... Power loss in Accelerometer 1
```



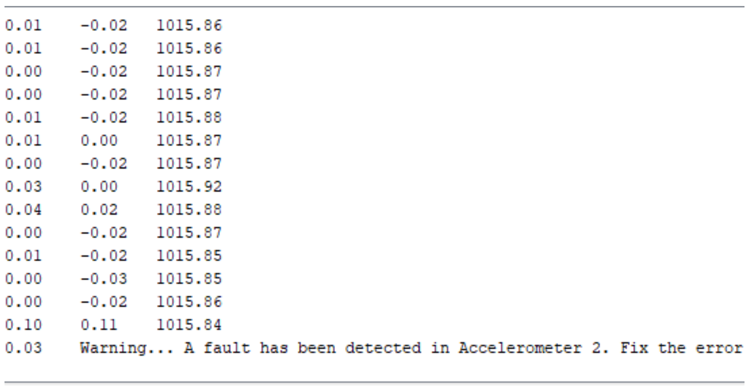
```
0.01    -0.02   1015.89
0.01    -0.02   1015.89
0.01    0.00    1015.92
0.01    -0.02   1015.90
0.01    0.00    1015.90
0.01    -0.02   1015.92
0.01    -0.02   1015.91
0.00    -0.02   1015.91
0.00    -0.02   1015.92
0.00    -0.02   1015.92
0.01    0.00    1015.94
-0.03   -0.05   1015.91
-0.03   -0.06   1015.91
0.01    0.02    1015.93
Warning... A fault has been detected in Accelerometer 1. Fix the error
```

Figure 3: Power loss (left) and Input loss (right) fault detection in the Arduino for Accelerometer 1 working successfully accelerometer as the power and input pins were removed

A similar process was required for the pressure sensor. By first collecting the raw data, the pressure sensor automatically fails to run if the power is turned off, and so no data is collected. This is an easy means to identify a fault within this sensor. In terms of an error with the input pins, the following graph shows the change.



```
0.28    -3.13   1015.79
0.00    -5.04   1015.79
0.01    -4.93   1015.82
0.01    -5.10   1015.80
0.01    -5.30   1015.80
0.01    -5.47   1015.81
0.03    -5.60   1015.84
0.00    -5.69   1015.83
0.00    -5.77   1015.81
0.01    -5.85   1015.81
0.01    -5.89   1015.84
0.00    -5.94   1015.84
0.01    -5.98   1015.84
0.00    -6.00   1015.82
0.00    Warning... Power loss in Accelerometer 2
```



```
0.01    -0.02   1015.86
0.01    -0.02   1015.86
0.00    -0.02   1015.87
0.00    -0.02   1015.87
0.01    -0.02   1015.88
0.01    0.00    1015.87
0.00    -0.02   1015.87
0.03    0.00    1015.92
0.04    0.02    1015.88
0.00    -0.02   1015.87
0.01    -0.02   1015.85
0.00    -0.03   1015.85
0.00    -0.02   1015.86
0.10    0.11    1015.84
0.03    Warning... A fault has been detected in Accelerometer 2. Fix the error
```

Figure 4: Power loss (left) and Input loss (right) fault detection in the Arduino for Accelerometer 2 working successfully accelerometer as the power and input pins were

Establishing the algorithm for the pressure sensor is much easier, as the condition for a fault to be detected is that the pressure sensor begins reading negative pressures, which highly impossible for its purposes. Implementing this algorithm results in the following fault detection response in the Serial Monitor.

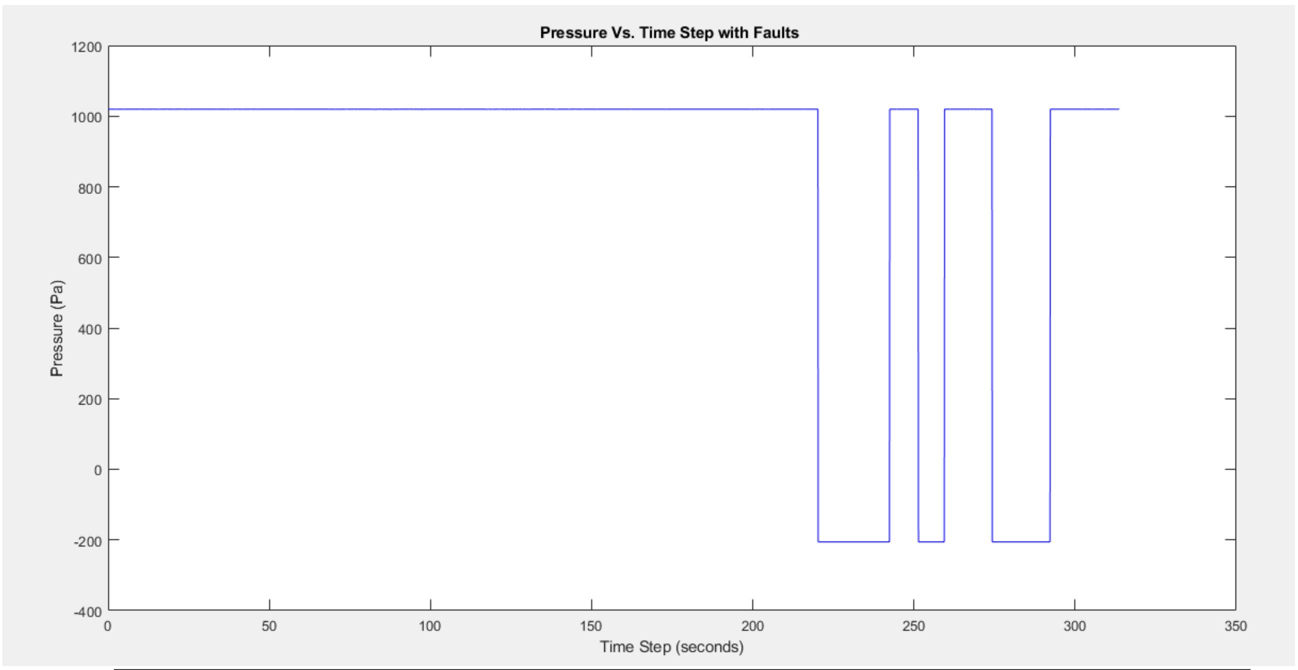
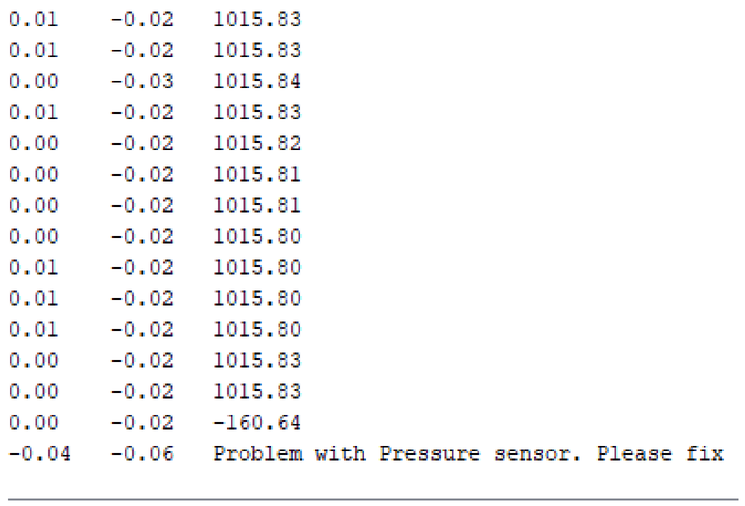


Figure 5: Pressure sensor raw readings as the input pins are removed

Algorithms may not be entirely robust, however, given that the project is only to track fluctuations as the student lifts and lowers the sensors and Arduinos, it is considered suitable enough for the project.

The testing procedures are all utilised with the IDE file named: 'Project3_FaultDetection_Part1'



```
0.01    -0.02   1015.83
0.01    -0.02   1015.83
0.00    -0.03   1015.84
0.01    -0.02   1015.83
0.00    -0.02   1015.82
0.00    -0.02   1015.81
0.00    -0.02   1015.81
0.00    -0.02   1015.80
0.01    -0.02   1015.80
0.01    -0.02   1015.80
0.01    -0.02   1015.80
0.00    -0.02   1015.83
0.00    -0.02   1015.83
0.00    -0.02   -160.64
-0.04   -0.06   Problem with Pressure sensor. Please fix
```

Figure 6: Pressure sensor fault detection when the input reading is removed

C.) PART 2: DATA FUSION AND KALMAN FILTER (SIMULATION)

i) ~~a)~~ With the autocorrelation function given as $R(\tau) = \frac{\beta}{2} e^{-\beta|\tau|}$ we can use PSD + find the ~~st~~ state equations. by ~~first saying~~

~~$x_1(t) = s(t)$~~

$$\dot{x}_1(t) = -\beta_s x_1(t) + \sqrt{2\beta_s}\,\sigma_s\,\omega_1(t).$$
$$\dot{x}_2(t) = -\beta_N x_2(t) + \sqrt{2\beta_N}\,\sigma_N\,\omega_2(t)$$

where $\omega_1(t)$ & $\omega_2(t)$ are unity white noise

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}-\beta_s & 0\\ 0 & -\beta_N\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}\sqrt{2\beta}\,\sigma & 0\\ 0 & \sqrt{2\beta}\,\sigma\end{bmatrix}\begin{bmatrix}\omega_1(t)\\ \omega_2(t)\end{bmatrix}$$

ii). ~~b)~~ Discretising the system in per lectures

$$\begin{bmatrix}x_{k+1,1}\\ x_{k+1,2}\\ \xi_{k+1}\end{bmatrix} = \overbrace{\begin{bmatrix}1 & \tau & 0\\ 0 & 1 & 0\\ 0 & 0 & e^{-\beta\tau}\end{bmatrix}}^{A}\begin{bmatrix}x_{k,1}\\ x_{k,2}\\ \xi_k\end{bmatrix}$$

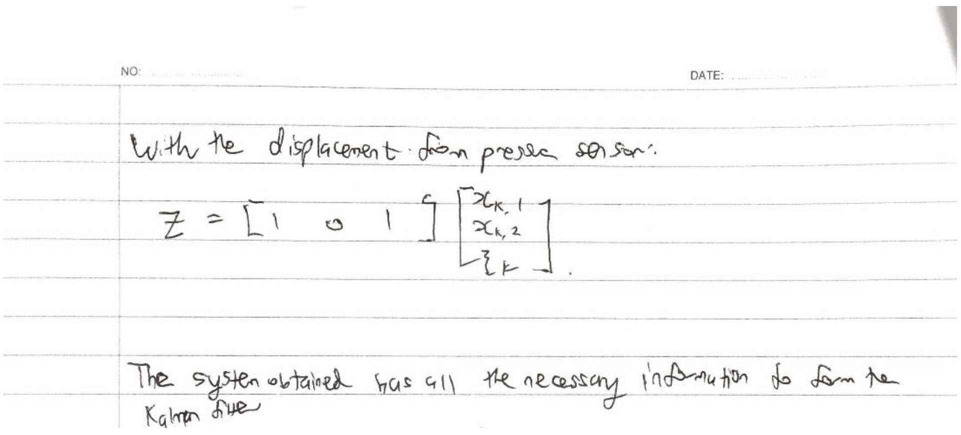
$$+ \underbrace{\begin{bmatrix}\tau^2/2\\ \tau\\ 0\end{bmatrix}}_{B.} u_k + \overbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & B_{2,k}\end{bmatrix}}^{\text{white noise}}\begin{bmatrix}\omega_{k,1}\\ \omega_{k,2}\\ \bar{v}_k\end{bmatrix}$$

where $B_{2,k} = \begin{bmatrix}-1/\beta_s & 0\\ 0 & -\frac{1}{\beta_N}\end{bmatrix}\left(\begin{bmatrix}e^{-\beta_s T}-1 & 0\\ 0 & e^{-\beta_N T}-1\end{bmatrix}\right)\begin{bmatrix}\sqrt{2\beta_s}\,\sigma_s & 0\\ 0 & \sqrt{2\beta_N}\,\sigma_N\end{bmatrix}$

This is the linear system.

$$= \begin{bmatrix}-\sqrt{\frac{2}{\beta_s}}\,\sigma_s\left(e^{-\beta_s\tau}-1\right) & 0\\ 0 & -\sqrt{\frac{2}{\beta_s}}\,\sigma_s\left(e^{-\beta_s\tau}-1\right)\end{bmatrix}$$

$$\therefore B_{2,k} = -\sqrt{\frac{2}{100}}\,\sigma_s\left(e^{-\beta_s\tau}-1\right).$$



NO:

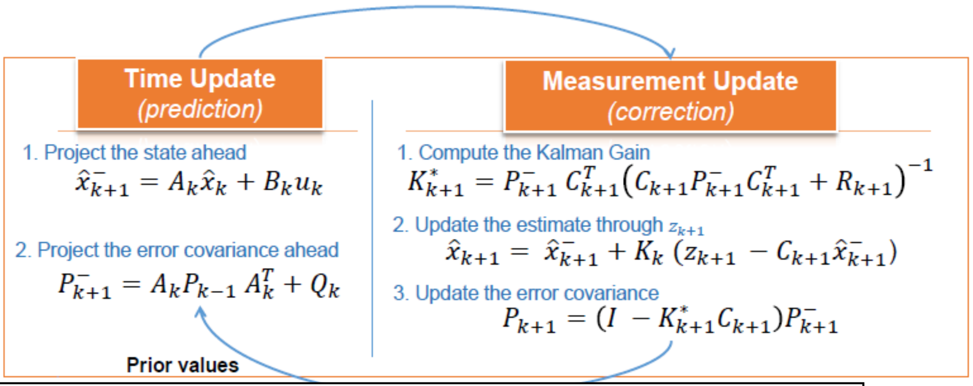
DATE:

With the displacement from pressure sensor:

$$Z = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k,1} \\ x_{k,2} \\ \xi_k \end{bmatrix}.$$

The system obtained has all the necessary information to form the Kalman filter

iii.) With the model above, A and B as per the Kalman Filter process are known. C is the covariance matrix which was selected to be [1, 0, 1]. K is the Kalman Gain which will help with the correction process to ensure to correct for the noise being produced.



Time Update (prediction)
1. Project the state ahead
$\hat{x}^-_{k+1} = A_k\hat{x}_k + B_k u_k$
2. Project the error covariance ahead
$P^-_{k+1} = A_k P_{k-1} A_k^T + Q_k$
Prior values
Measurement Update (correction)
1. Compute the Kalman Gain
$K^*_{k+1} = P^-_{k+1} C^T_{k+1}\left(C_{k+1}P^-_{k+1}C^T_{k+1} + R_{k+1}\right)^{-1}$
2. Update the estimate through z_{k+1}
$\hat{x}_{k+1} = \hat{x}^-_{k+1} + K_k\left(z_{k+1} - C_{k+1}\hat{x}^-_{k+1}\right)$
3. Update the error covariance
$P_{k+1} = \left(I - K^*_{k+1}C_{k+1}\right)P^-_{k+1}$

Figure 7: Kalman filter algorithm as per the lectures

Now from the previous project,the Q and R matrices, which represent the acceleration and pressure sensor noise respectively, were selected based off research completed by the student (Shimkin, 2009). The article cited shows that the Q and R matrix is best defined according to the following description:

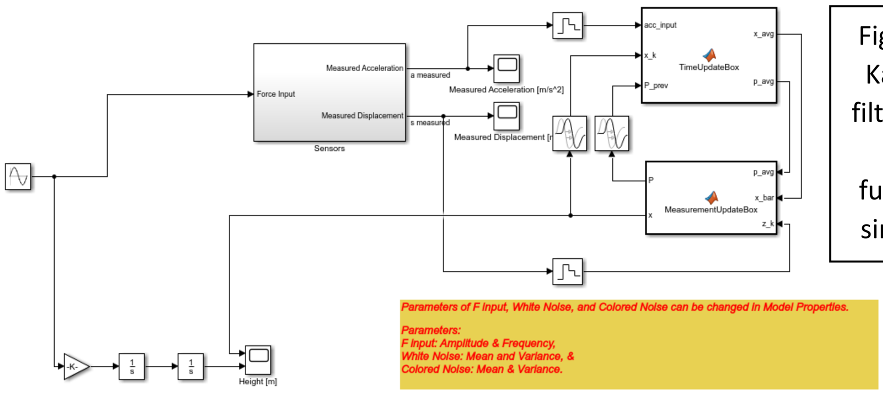
$$Q = BB^T\sigma^2_{acc} \qquad R = \sigma^2_{press}$$

However, given that the accelerometer has white noise affecting it, there is an additional factor to be added, which is shown below.

$$Q = BB^T\sigma^2_{acc}(white\ noise)(white\ noise)^T$$

With the R matrix however, there is no white noise included, so remains unchanged. The variances of both the accelerometer and pressure sensor have been set to 1 for simplicity.

iv.) Applying the Kalman Filter in the Simulink model (Figure 8) (also included in submission) produces a graph for distance as shown on the next page.



Force Input
Measured Acceleration
Measured Displacement
Sensors
a measured
s measured
Measured Acceleration [m/s^2]
Measured Displacement [
acc_input
x_k
P_prev
x_avg
p_avg
TimeUpdateBox
P
x
p_avg
x_bar
z_k
MeasurementUpdateBox
-K-
1/s
1/s
Height [m]
Parameters of F Input, White Noise, and Colored Noise can be changed in Model Properties.
Parameters:
F Input: Amplitude & Frequency,
White Noise: Mean and Variance, &
Colored Noise: Mean & Variance.

Figure 8: Kalman filter with data fusion in simulink

As can be seen, the Kalman filter (yellow) is able to track the integrated signal (blue) quite well, showing that it is quite adequate in tracking the overall height and distance. Moreover, it doesn't have as much of the noisy signal that the integrated function has, indicating a smoothness and removal of noise.

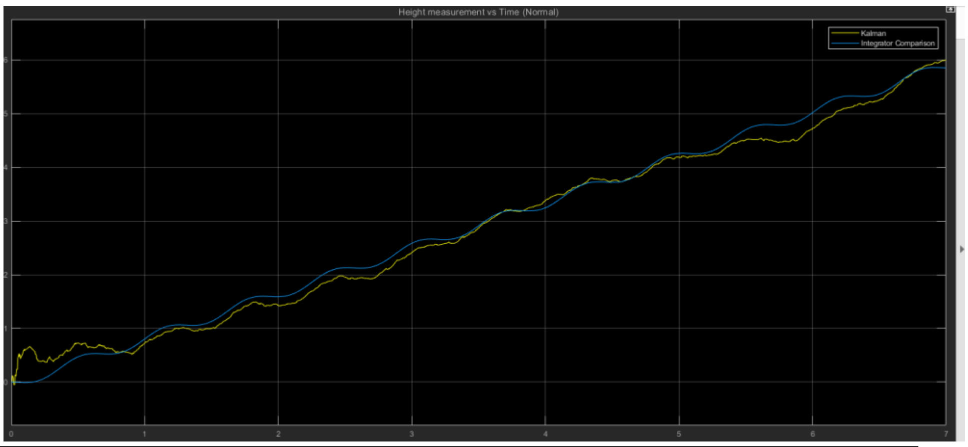


Figure 9: Kalman filter with data fusion and the integrated displacements, showing how well the Kalman filter tracks

v.)

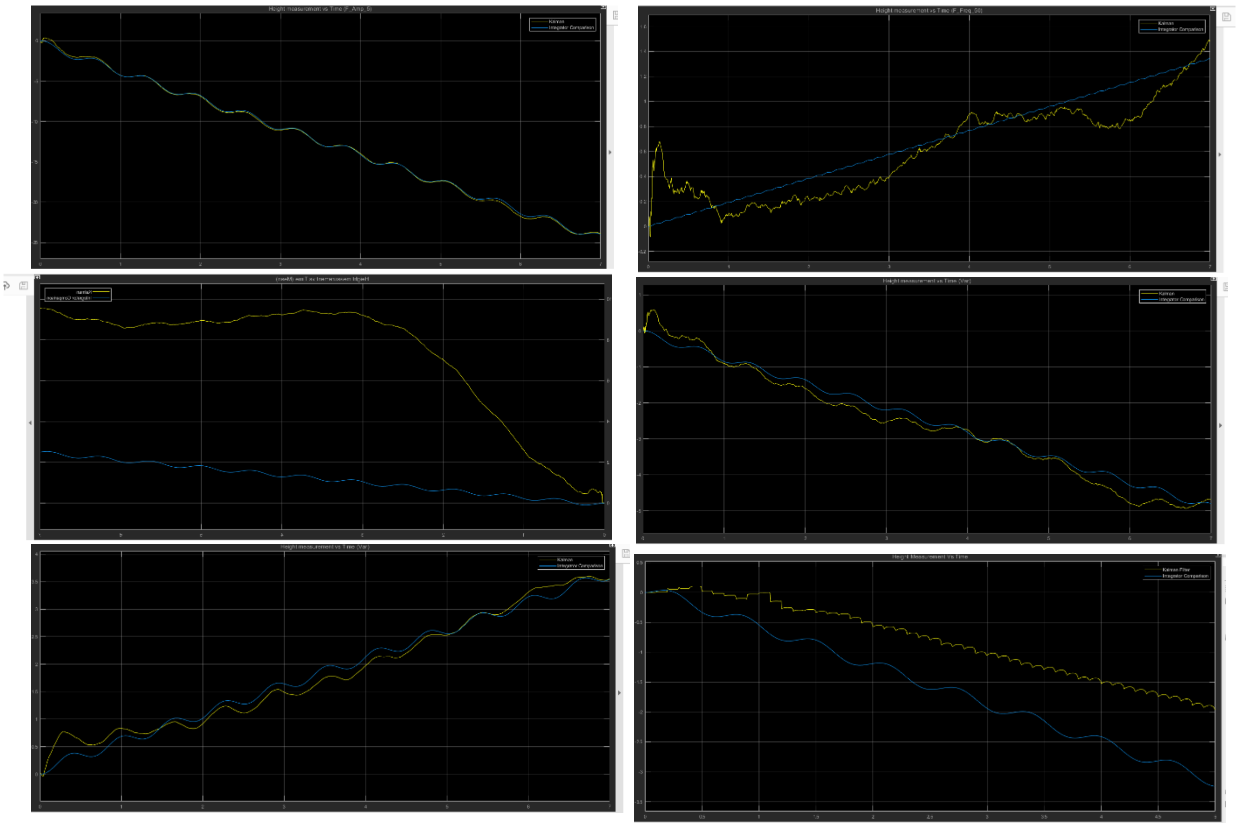


Figure 10: Kalman filter accuracy of fit with F_amplitude = 5 (Top Left), F_frequency = 50 (Top Right), WhiteNoise_Mean = 5 (Centre Left), WhiteNoise_Variance = 10 (Centre Right), Beta = 10 (Bottom Left) and Sampling Period = 0.1 (Bottom Right)

Increasing the amplitude of F appears results in the Kalman filter closely following the integrated displacement. This is not ideal given that if the displacement was very noisy, this would result in the Kalman filter being noisy as well. Increasing the F frequency results in a far noisier Kalman filter and it becomes out of sync too, being highly ineffective. Change the white noise variance and beta values don't appear to wildly change the Kalman filter read out in comparison to that of the integrated displacement. However, changing the white noise mean significantly affects the Kalman filter, resulting in no similarity or accuracy in tracking. This shows how significant the effect of white noise can be to the accuracy and effectivity of the Kalman filter.

With the sampling period, by turning it to 0.1 seconds, the Kalman filter is bumpier, but generally follows the same trend as displacement. However, it is much slower and less sensitive to changes and hence is much higher than the actual displacement.

D.) Part 3: Data Fusion and Kalman Filter (Arduino Implementation)

i.) The height estimation algorithm which is used initially is over-simplistic and inaccurate. It is based off the equations of motion:

$$s = 0.5at^2 + s_{prev}$$

Where t is the sampling time and s_{prev} is the previous height measurement which comes from the pressure sensor and a which is the combination of the accelerometers accelerations.

However, since the distance being calculated is dependent on 3 sensors: 2 accelerometers and 1 pressure sensor, the all contribute a little to the overall height being measured. As such, a constant factor of 0.33 is multiplied to all input measurements for this algorithm, i.e.:

$$s = 0.5(0.33a_1 + 0.33a_2)t^2 + 0.33s_{prev}$$

In doing so, the overall effect is of each sensor sums up to be 1 (0.33 + 0.33+0.33).

ii.) This algorithm also employs the fault detection algorithms shown in Part 1. Figure 11 shows the algorithm above with both accelerometers and Figure 12 shows it when only one accelerometer is working. The motion involved and being shown is moving the Arduino and sensors up and then back down to the table top. The code to test this is included in the submission as: 'DataFusion_v2_671116'

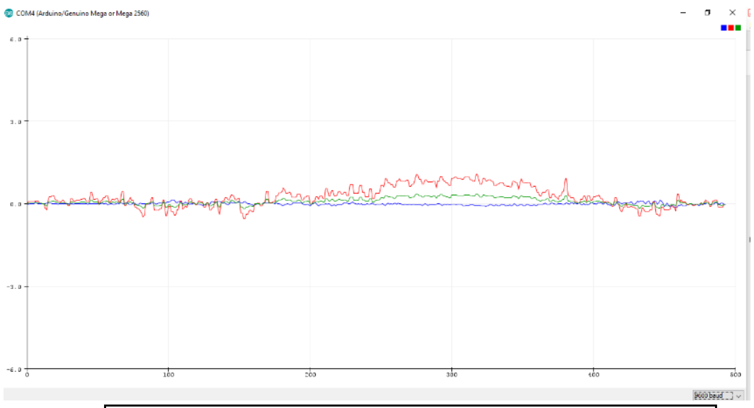


Figure 11: Height estimation algorithm when all 3 sensors are working

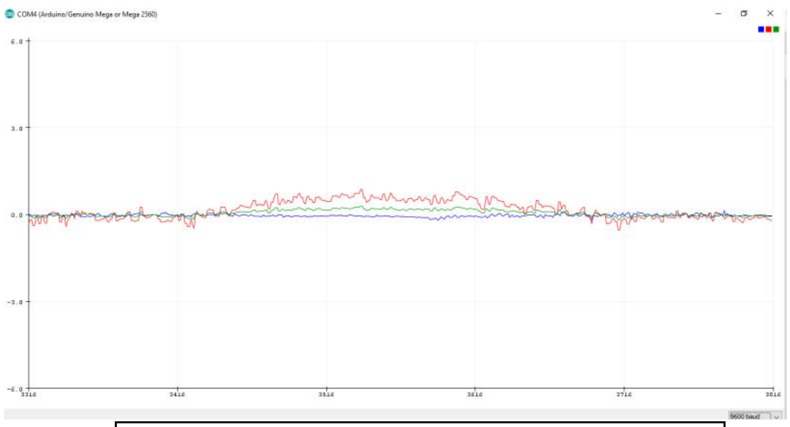


Figure 12: Height estimation algorithm when 1 accelerometer has failed

The red line represents the pressure sensor reading for the displacement. The blue line is the data fusion between the 2 accelerometers. The green line is the height estimation algorithm implemented.

iii.) As can be seen, the height tracking is not very good. It attempts to follow the pressure sensor but fails to do so accurately. As such, the Kalman filter is needed to be used. From Part 2, the Kalman filter designed was subsequently implemented into the Arduino. As can be seen in Figures 13 and 14, the Kalman filter is able to better track the displacements moved. This is even true when one of the accelerometers has failed (Figure 14). Comparing this to the Project 2, the results are far more robust now, while being as accurate, if not better. Even if one sensor goes faulty, the system is still operational and relatively accurate. Moreover, the added accelerometer has allowed for an overall better tracking of the actual displacement moved, which was severely lacking when the altimeter was designed in Project 2. The code used for this testing is included in the submission as: 'Kalman_Filter_Project3_671116'

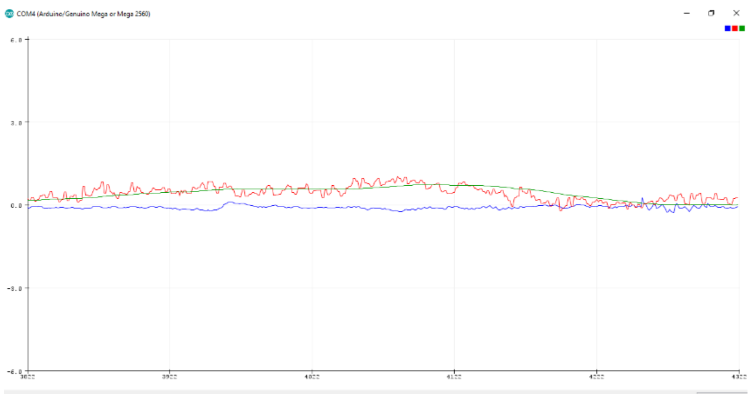


Figure 13: Kalman filter height estimation algorithm when all 3 sensors are working

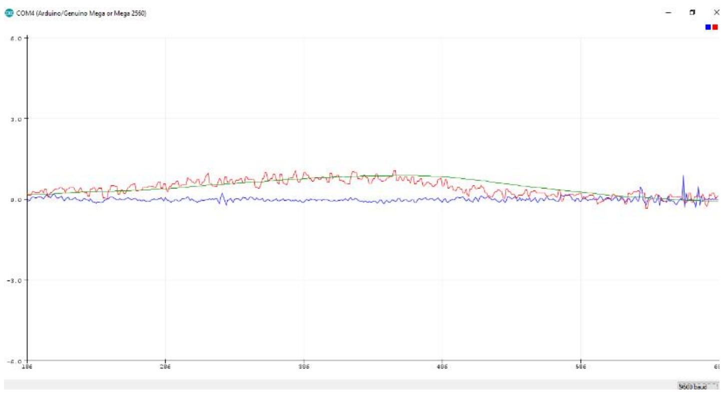


Figure 14: Kalman filter height estimation algorithm when 1 accelerometer has failed

The red line represents the pressure sensor reading for the displacement. The blue line is the data fusion between the 2 accelerometers. The green line is the Kalman filter tracking of the height of displacement.

E.) CONCLUSION:

From the project and tests conducted, introducing a 2nd accelerometer significantly improves the overall accuracy and ability for the Kalman filter to accurately track displacements. Moreover, this project enforced the concept and effect of white noise of sensors and how it can distort the actual signals being received. As a result, being able to account for it can significantly improve the overall accuracy that the altimeter is trying to achieve. By having 2 sensors instead of the 1 accelerometer, there is the added layer of redundancy that if one were to fail, the system would not cease to operate as the second accelerometer is still able to operate.