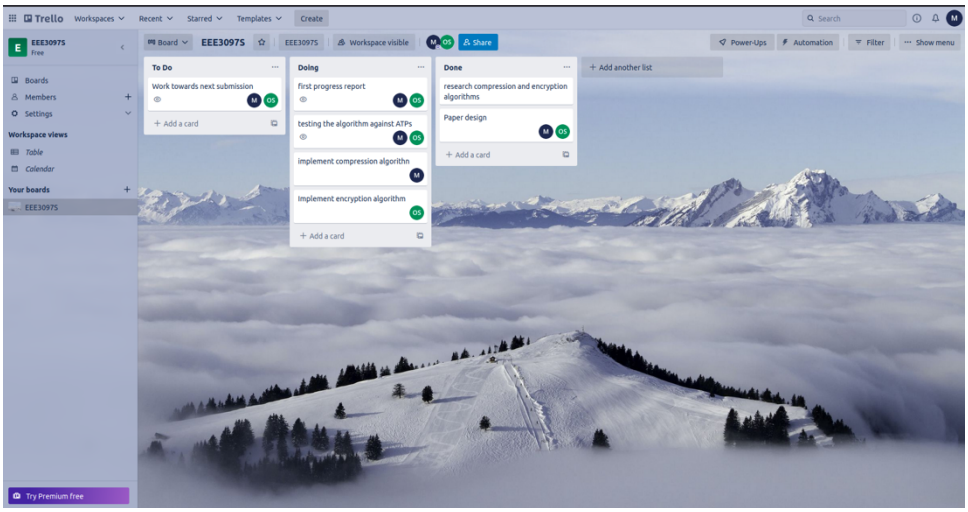
First Progress Report

Oliver Shaw *Mbasa Mguguma*
SHWOLI002 *MGGMBA003*

Oliver Shaw	Encryption Subsystem
Mbasa Mguguma	Compression Subsystem

1 Admin Documents

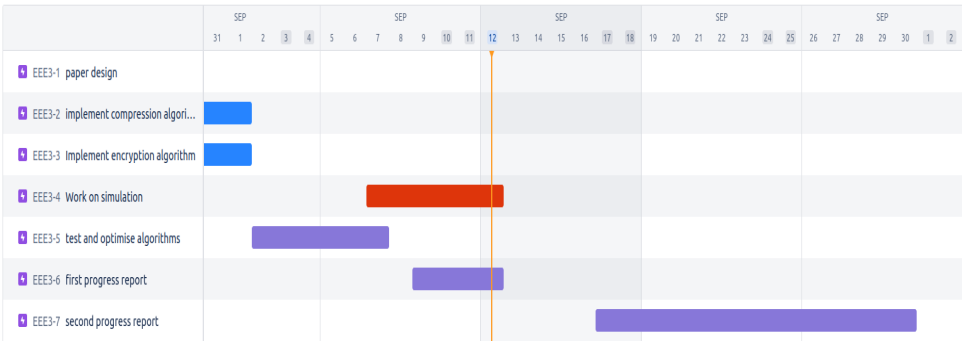
1.1 Project Management Page



1.2 Link to Git Repo

https://github.com/Gagasii/EEE3097S

1.3 Timeline



Our progress was delayed by a day.

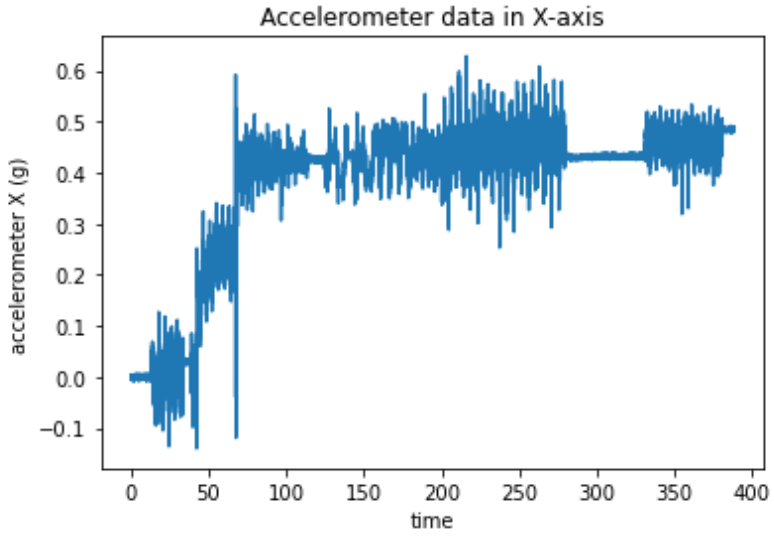
2 Data

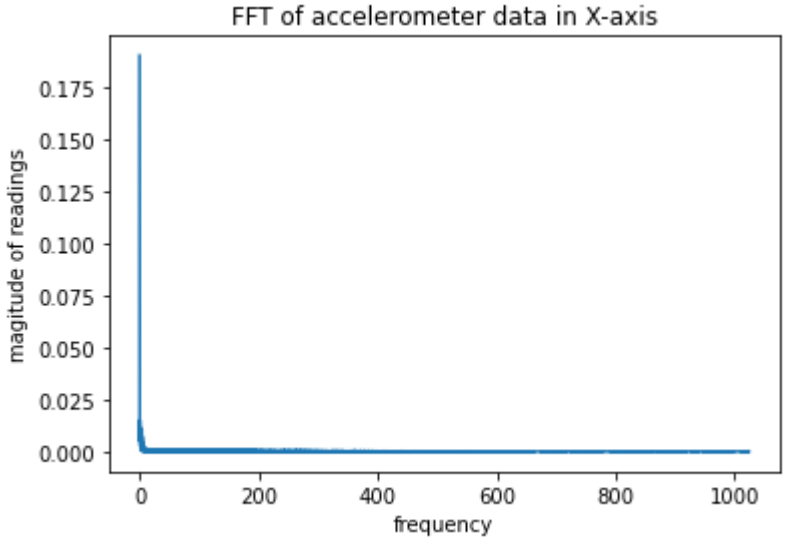
The data that we used for the simulation is somewhat the same as the one that we expect from the IMU, a csv file that contains all the data measured by the sensor, including all the measurements from the 3-axis Gyroscope, 3-axis Accelerometer, 3-axis magnetometer etc. The data used for simulation can be seen in the git repository for the project. Since we want as much of the IMU data as possible a csv file is suitable and has readings up to 4000 rows which is a lot.

There are three different sets of data to be used for our simulation phase. The first is a data file coming from IMU that is 5 minutes of a user walking, sitting and standing. The second is a 10-minute long recording of the data of the IMU rotating about one of its axes. This file is taken at a high sample rate. The final data file is the same as the second data file put the sampling rate is lower.

The data is good enough to run all the specifications and ATPs, since we are able to check the input data against the output data of the system to check if the data is the same to some extent. There are a lot of readings, this will allow us to check if our algorithms will run at the lowest possible time in order to keep power usage low.

The following plots were made from the data in the csv file. More plots are available in our Git Repository (https://github.com/Gagasii/EEE3097S).





3 Experiment Setup

3.1 Overall Functionality

The experiment for overall functionality of the system is made up of the compression subsystem, encryption subsystem. The system first compresses the data, then the compressed data is taken as input to the encryption block where it is encrypted, the encrypted file is then decrypted then decompressed. The output data from the system is checked against the input data to the system. This is done by parsing through the data and putting each piece into an array. The arrays are then compared in order to verify that there are no discrepancies in the system. This confirms firstly that through the processes the data is the same as well as through the communication between the systems.

Since the system has to minimize power usage, the overall system has to take lowest possible time processing. The compression and encryption blocks are timed and this allows us to know how fast the system, as a whole, is processing the information.

3.2 Encryption Subsystem

To determine the speed of the encryption we will time the individual encryption and decryption for different data classes. To do this is relatively easy through timing directly in C. For the simulation stage of the project we will be using the time function in C which implements Epoch time. The timing will be set up for both the encryption and decryption subsystems individually.

The changing of the size of the data class as well as changes in sampling rate. This is important as it can help us verify that the code works under different cases. This ensures integrity in the output through rigorous testing.

For testing that the encryption process works we will be implementing a testing process where the data blocks are compared before encryption and after decryption. This is important and it allows us to verify the accuracy of the encryption/ decryption code. This will be done by comparing the blocks at input (before encryption) and at output (after decryption).

This is done by using array comparison and deducting each element from the related element in the other array. An array is then created with the results of the deductions. The array is then checked by finding if any element in the is non-zero and then it returns that the algorithm is creating errors.

3.3 Compression Subsystem

The experiment for the compression block is made up of encoding and decoding, and both these operations are timed to check runtime and determine roughly the power consumption of the block. Encoding is when the data taken from the csv file is compressed to give out a compressed csv file. Decoding is when the compressed csv file is decompressed to get the initial data from the input csv file.

During encoding a csv file is taken as input and a compressed version of the csv file is the output. During decoding a compressed csv file from encoding is the input and a decompressed csv file is the output (hopefully the same as the input csv file).

To check if the compression block works well, we will compare the input data before encoding and the output data after decoding. The size and amount of data was also varied to check the performance of the block for different data sizes.

The algorithm that I used for the compression block is Huffman's compression algorithm, I could not find the combination of delta and RLE implementation in C so I opted for Huffman 8-bit compression.

4 Results

4.1 Overall Functionality

When we compared the data at input to the output we found no discrepancies between the data sets. This is important as it allows us to see that through the simulation phase our system is working.

Through our timing experiments we are able to see that the system is working more quickly than our benchmarks. This should translate into our system being power efficient.

4.2 Encryption Block

The encryption testing was done using three cases. Each test was run 5 times and the average was taken to get an accurate result of the timing. It is important to note that these tests were run on the laptop side using the laptops CPU. The speed of the processors is approximately 3.2 GHz and has 8 cores (4 performance cores and 4 efficiency cores).

Test Number	File Size in Mb	Time taken to Encrypt	Time taken to Decrypt
1	9.7	14.55 ms	14.52 ms
2	8.7	13.04 ms	13.00 ms
3	2.8	0.042 ms	0.042 ms

Table 1: Encryption and Decryption time for three different file cases

The encryption and decryption are essentially identical (but in opposite order) so therefore the timing should take the same amount of time. The only difference between the two is the decryption process of AES can be parallelised and can therefore be faster than the encryption process.

The results are very good as they fall within the 1.5 seconds per gigabyte of data that was aimed for before implementing the algorithm.

The second experiment run for the simulation phase of the project showed that the encryption and decryption process was accurate.

4.3 Compression Block

The 1st encoding experiment took a csv file of size 22.4MB and the output is a csv file of size 9.7MB. The operation took 1.16 seconds and gave us a compression ratio of 2.31. After that the 1st decoding experiment took the output of the encoding experiment which is the 9.7MB csv file and gave an output of 22.4MB in 0.6 seconds.
The 2nd encoding experiment took a csv file of size 19.8MB and the output is a csv file of size 8.7MB. The operation took 0.98 seconds and gave us a compression ratio of 2.28. After that the 2nd decoding experiment took the output of the encoding experiment which is the 8.7MB csv file and gave an output of 19.8MB in 0.59 seconds.

The input csv files and the output csv files from both experiments have the same size, meaning the content of the files is more or less the same, to verify this I manually looked through the files and checked whether they are the same and yes, the files are the same.

5 Acceptance Test Procedures

5.1 Encryption Block

ATP	Description	Met?
U1.F2.S1.A1	Take a data set and compare it at encryption vs decryption to ensure 100% integrity for the encryption subsystem	Yes
U2.F3.S1.A1	Set up a timer for the encryption process with a speed of 1.5 seconds per gigabyte of data.	Yes
U2.F3.S1.A2	Set up a timer for the decryption process with a speed of 1.5 seconds per gigabyte of data.	Yes

The timing testing procedures will be very straight forward as they rely simply on code based timings that can be easily implemented into the code in order to ensure they are running efficiently and quickly. This will allow us to check for bottlenecks slowing down the encryption or decryption process.

A defined data set must be input into the encryption system in order to test the integrity of the code. This way we will be able to compare the output of the decryption and compare it directly to the defined data set.

The figures of merit for the encryption subsystem are that the encryption and decryption each take 1.5 seconds (maximum) to execute. As well as 100% integrity of the data after decryption.

5.2 Compression Block

U2.F1.S1, was changed from compression ratio should be less than 0.25 to compression ratio has to be greater than 2. This is a correction to the specification in the paper design. since a compression ratio of 0.25 means the compressed data is bigger than the original data and that is not correct.

ATP	Description	Met?
U1.F1.S1.A1	25% of Fourier coefficients present	Yes
U2.F1.S1.A1	Encoding/decoding under 2 seconds	Yes
U2.F1.S1.A2	Compression ratio greater than 2	Yes