Second Progress Report

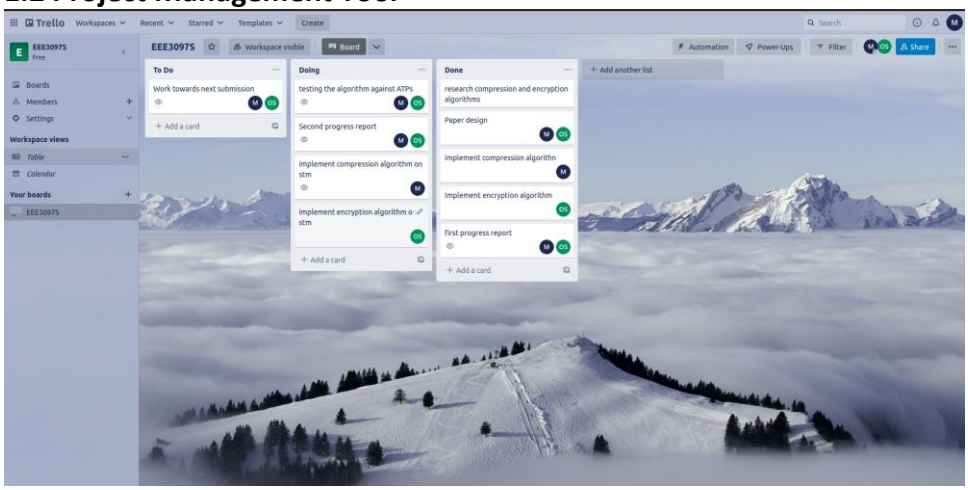
Mbasa Mguguma *Oliver Shaw*
MGGMBA003 *SHWOLI002*

1 Admin Documents

1.1 Table of Contributions

Mbasa Mguguma	Compression Subsystem
Oliver Shaw	Encryption Subsystem

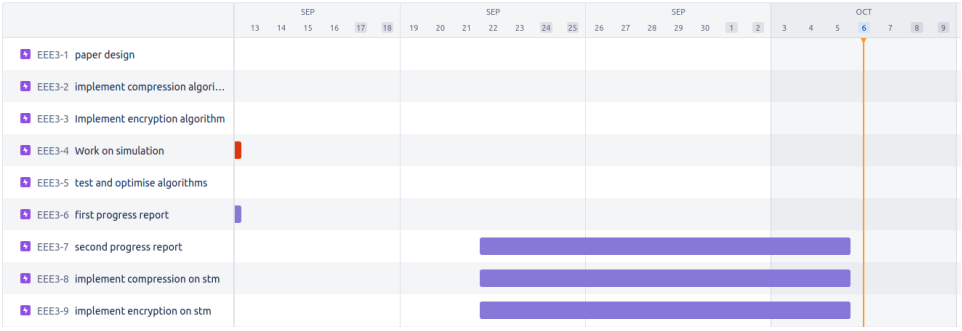
1.2 Project Management Tool



1.3 GitHub Link

https://github.com/Gagasii/EEE3097S-Project

1.4 Timeline



Our progress was delayed by a day.

2 IMU Module

2.1 Sensor Features and Differences between Sense HAT (B) and ICM-20649

This section will discuss the differences between the sensor we have used, the Sense HAT (B), and the IMU used in the real-life implementation in Antarctica, the ICM-20649. The initial difference is that the Sense HAT (B) offers a less specialised set of sensors. The Sense HAT (B) offers a 3-axis accelerometer, 3-axis gyroscope, magnetometer, barometer as well as a temperature and humidity sensor. The ICM-20649 , however, is a more specialised sensor as it offers a 3-axis accelerometer, 3-axis gyroscope as well as an on board temperature sensor. The essential differences that are important here are between the ICM-20649 and the ICM-20948 (the accelerometer, gyroscope and magnetometer device on the Sense HAT(B)). The ICM-20948 is a 9-axis device meaning it includes a magnetometer as well as the accelerometer and gyroscope. The ICM-20948 only has Full-Scale Range up to $\pm$2000 degrees per second for its gyroscope meaning it has 16.4 least significant bits per degree per second whereas the ICM-20649 has $\pm$4000 degrees per second meaning it has 8.2 least significant bits per second. This means that the device used in the final application is more sensitive and able to detect and convey smaller changes in the gyroscope's data. The accelerometer of the ICM-20948 has Full-Scale Range of up to $\pm$16G with a sensitivity scale factor of 2048 least significant bits per G whereas the ICM-20649 has Full-Scale Range of $\pm$30G and sensitivity scale factor of 1024 least significant bits per G. This again shows us that the sensor used in the actual implementation of this project has a much more sensitive set of sensors. Both the devices are manufactured by the same company and operate with the same drain power voltage, V_{dd}, of 1.8 V.

2.2 Validation Testing of the Sense HAT (B)

In order to ensure our IMU is working correctly we will test it through a set of different tests. These tests are broken down into static tests and dynamic tests.

The Static Testing is to test our IMU for drift rates. Drift rates are essentially a description of the rate at which the value of the IMU deviates from the wanted value over time. Ideally these tests would be taken over long periods of time however for our tests we decided to use shorter periods of time simply to simplify the process. The Static Tests are also designed to try and test that at if the IMU is not moving the value holds true and doesn't return false values in the most basic set up. Our set up involved setting the IMU at the base of a home made pendulum as well as stable on a 'flat' surface. This was followed by a test where we moved the IMU around and vibrated (shook) it and return it to the original position to gauge if when returned the values stayed constant.

The Dynamic Tests are designed to try and test the accuracy of the IMU. Unfortunately we do not have as accurate a set up as we would want. The ideal set up would involve the use of a metal pendulum to ensure the most accurate results. Our setup used a homemade pendulum which would act to establish a set base position of known position and from there when the IMU is released from this height we are able to compare our expected results with the actual data given by the IMU. These tests will be repeated three times in order to get an accurate understanding of any errors that presented themselves.

For these tests we wanted to compare our sensors data with the equivalent sensors that exist on the ICM-20649 and therefore will be using the accelerometer, gyroscope and

temperature sensors. We set the ICM-20649 to have gyroscope range of ±2000 degrees per second and the accelerometer to ±16G.

2.2.1 Static Testing

For the first Static Test in which the IMU is left at rest to test the effects of drift the values of the accelerometer, gyroscope and temperature before the test was started were approximately:

Table 1.1 showing the values of the 3-axis gyroscope, in radians per second, and the values of the accelerometer, in meters per second, of the Sense HAT (B) at rest at the beginning of the testing procedure for Static Test 1

	x	y	z
Gyroscope	0.00	0.02	0.00
Accelerometer	0.04	0.01	9.79

Where the gyroscope is returning in radians per second and the accelerometer is returning meters per second. This was felt to be an easier gauge and of easier use for when the data needs to be processed.

After an hour at rest in a stable position the values of the accelerometer, gyroscope and temperature were approximately:

Table 1.2 showing the values of the 3-axis gyroscope, in radians per second, and the values of the accelerometer, in meters per second, of the Sense HAT (B) at rest after one hour of the testing procedure for Static Test 1

	x	y	z
Gyroscope	0.00	0.01	0.00
Accelerometer	0.04	0.01	9.79

From this we only saw very minor changes to the gyroscope, 0.01 radians per second change is the Y-axis measurement, but no major changes in any field. The temperature obviously couldn't be factored into this test as the temperature was not able to be controlled throughout this process. This showed us that the drift error (although only tested for one hour) shouldn't affect our IMU and therefore we can confirm that under static or rest conditions our IMU is working effectively.

The second Static Test saw the IMU vibrated/ shaken and then replaced to the exact location in which is started before it was shaken the values of the accelerometer, gyroscope and temperature before the test was started were approximately:

Table 1.3 showing the values of the 3-axis gyroscope, in radians per second, and the values of the accelerometer, in meters per second, of the Sense HAT (B) at rest at the beginning of the testing procedure for Static Test 2

	x	y	z
Gyroscope	0.00	0.01	0.01
Accelerometer	0.04	0.01	9.80

After a vigorous amount of movement for 30 seconds and replaced in the same location and orientation as when it started the values of the accelerometer, gyroscope and temperature before the test was started were approximately:

Table 1.4 showing the values of the 3-axis gyroscope, in radians per second, and the values of the accelerometer, in meters per second, of the Sense HAT (B) at rest at the end of the testing procedure for Static Test 2

	x	y	z
Gyroscope	0.03	0.02	-0.02
Accelerometer	0.04	0.02	9.79

From this we are able to see that the rapid movement of the IMU resulted in some minor changes in the various parameters of each of the sensors (accelerometer and gyroscope). This however could not be definitively proven as the placement of the IMU after the test may have changed and affected the value. The second Static Test was then repeated three times in order to see if the affect persisted or if through the tests different errors were found that may be attributed to the rudimentary nature of the testing procedure.

Table 1.5 showing the average change in values of the 3-axis gyroscope, in radians per second, and the values of the accelerometer, in meters per second, of the Sense HAT (B) at after three rounds of testing for Static Test 2

	x	y	z
Gyroscope	0.01	0.00	-0.01
Accelerometer	0.01	0.01	0.02

From this set of testing, it was clearer to us that the IMU was in fact in working order and not being drastically affected by the tests. Because of this we could conclude that the IMU was in working order.

2.2.2 Dynamic Testing

For the dynamic testing we repeated a simple pendulum swing from a known height of 10 cm. The weight of the IMU and STM (along with the wiring was 100g). We will set the IMU and STM at the bottom of a solid centre pole which

3 Experimental Setup

3.1 System Functionality

The experiment for overall functionality of the system is made of up the compression subsystem, the encryption subsystem as well as the input of IMU Data as well as the STM. The data is read from the IMU using SPI. The IMU Data is read into a buffer on the STM. The Data is in the form of an array. This array is then compressed and passed onto the Encryption block which encrypts as well as decrypts the data. The data is then sent back to the compression block which decompresses it.

To check the data has kept integrity we do a simple array compare on the input and output data arrays. This can easily be done through a simple program or through an online resource that checks the data against each other and gives exact differences where any exist. Our initial set up was to use the two arrays and subtract them from each other to identify any discrepancies that existed between the two. We found that the online resource allowed us to compare the data much more quickly and gave us exact discrepancies much more accurately as we would see at which position the issue was and the input and output values instead of having to implement that in code or by rechecking the arrays ourselves. We also did a Fourier analysis on the data to check if the data is kept the same, by taking the discrete Fourier transform of the data and then checking the Fourier coefficients.

In order to ensure another vital part of our project, power consumption, we will be counting the clock cycles it takes for the data to be processed by each block individually as well as the system as a whole. From this we are able to deduce the time for the data to transferred as well as how much power we should be using based off of some rough calculations for this stage of the project.

3.2 Compression Subsystem

The compression algorithm the system uses is Huffman's compression algorithm which is implemented in C. The experiment includes compressing the data on the STM, transmitting the compressed data, then on the laptop decompress the data. The data is in the form of an array, it includes the 3-axis accelerometer data and 3-axis gyroscope data. There is a constraint of space since the STM only has 64Kb, so the data is not a lot compared to the previous data we used. The subsystem takes in an array and it gives out a compressed array, the compressed array is then transmitted to be decompressed and tested on the laptop.

To check for functionality, a number of tests are run, where the input buffer to the subsystem, the data read from the IMU, is compared to the output, the data decompressed on the laptop. This comparison is done using Fourier analysis and an online resource that checks difference on a file. The size of the input buffer is varied to test the performance of the system.

The subsystem is timed to determine its speed and to roughly approximate the power consumption of the subsystem. The timing of the subsystem is implemented in code.

3.3 Encryption Subsystem

The encryption algorithm used in our project is AES encryption. 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 on the STM or through C. For this 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 classes 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 this we have now used smaller data pieces as we have used a smaller amount to data in our IMU readings in order to reduce the workload of the STM due to its small amount of RAM.

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).

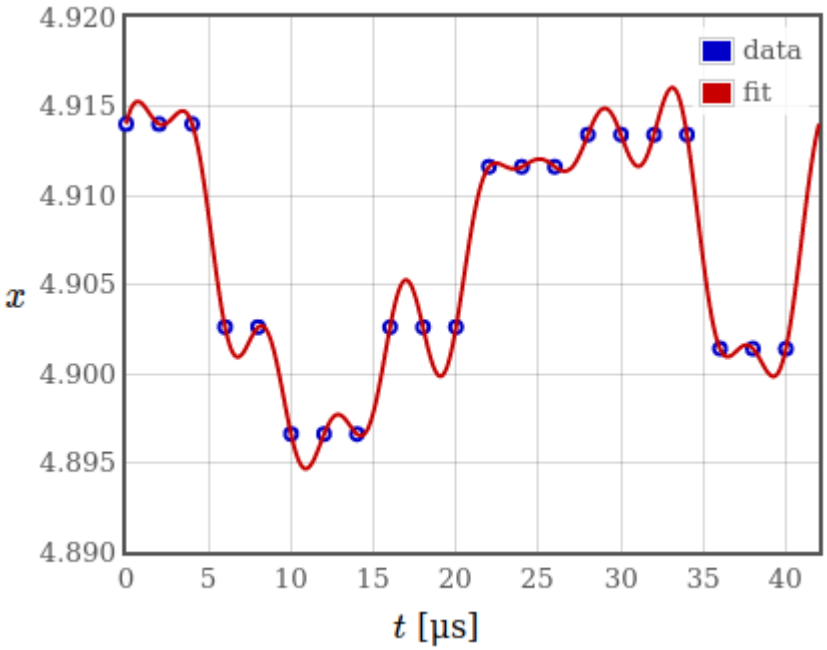
This is done by using array comparison and deducting each element from the related element in the other array. An array is then created with the results of the deductions. The array is then checked by finding if any element in the is non-zero and then it returns that the algorithm is creating errors.

The final test would be to test the breakability of the encryption algorithm however after much research it seems that due to the nature of AES encryption testing its breakability isn't really an option in a way other than brute force.

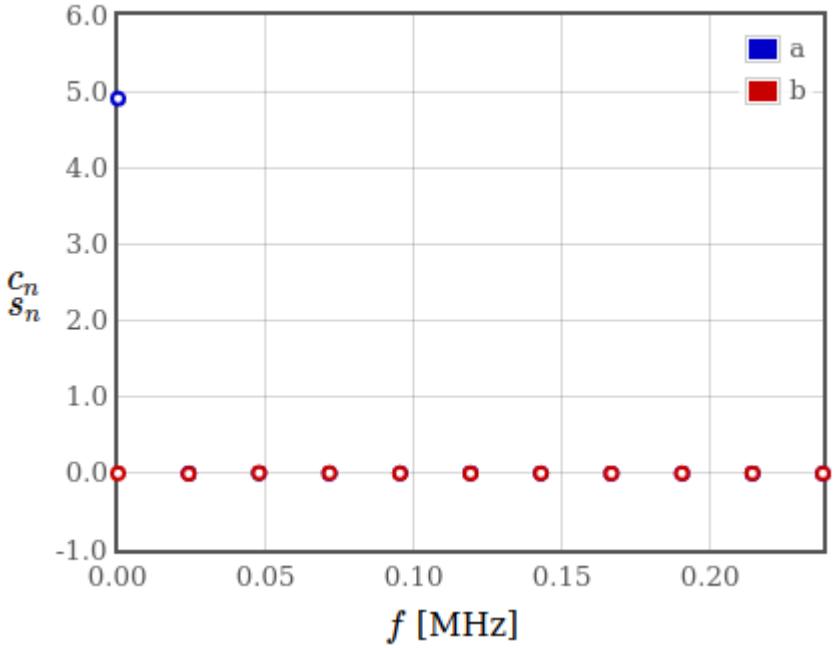
4 Results

4.1 System Functionality

In order for us to check the data we have created an input array and used an online array checking tool to determine any discrepancies between the input and output. The online array comparer found no discrepancies between any of our three test cases using different sets of data.
We also used Fourier analysis on the input and the output data of the system to check if the Fourier coefficients are present. To check this, we use the accelerometer data in the Z axis. The plots of the input data of the system are as follows:

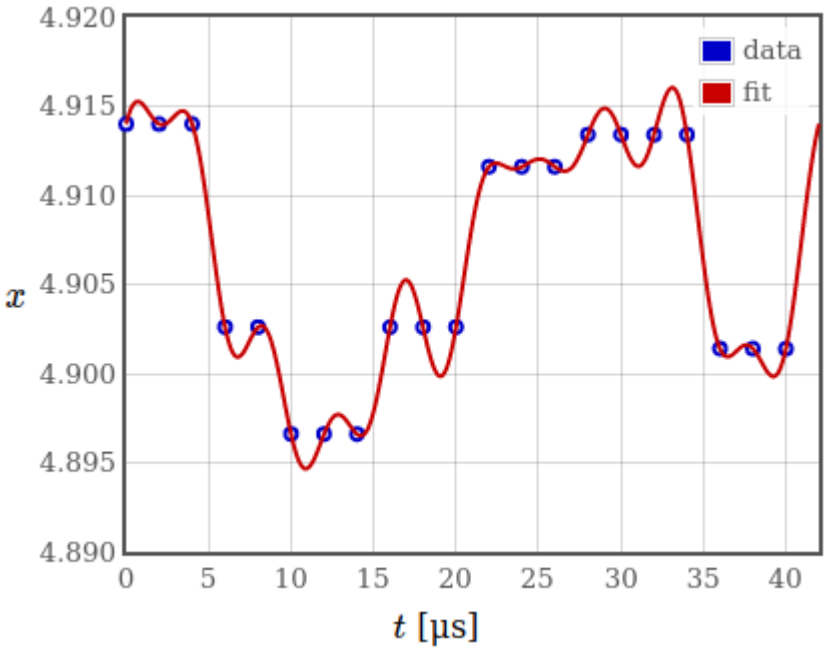


The plot above is the input time domain array of the accelerometer data in the Z-axis.

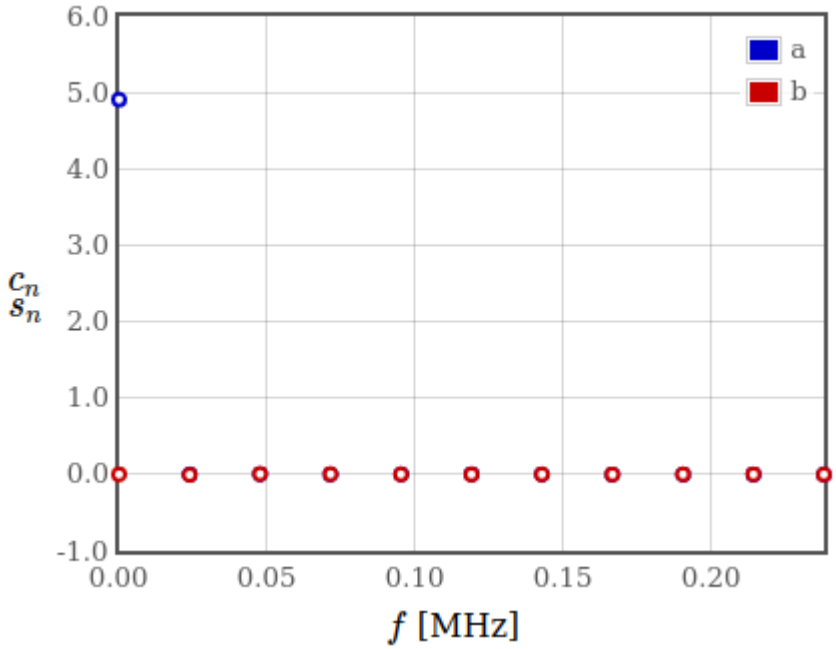


The plot above is the discrete Fourier transform which shows the Fourier coefficients of the data.

The plots of the output data of the system are as follows:



The plot above is the time domain plot of the output data, which is identical to the input data.



The plot above is the discrete Fourier transform that shows the Fourier coefficients of the data, which is also identical to the input data.

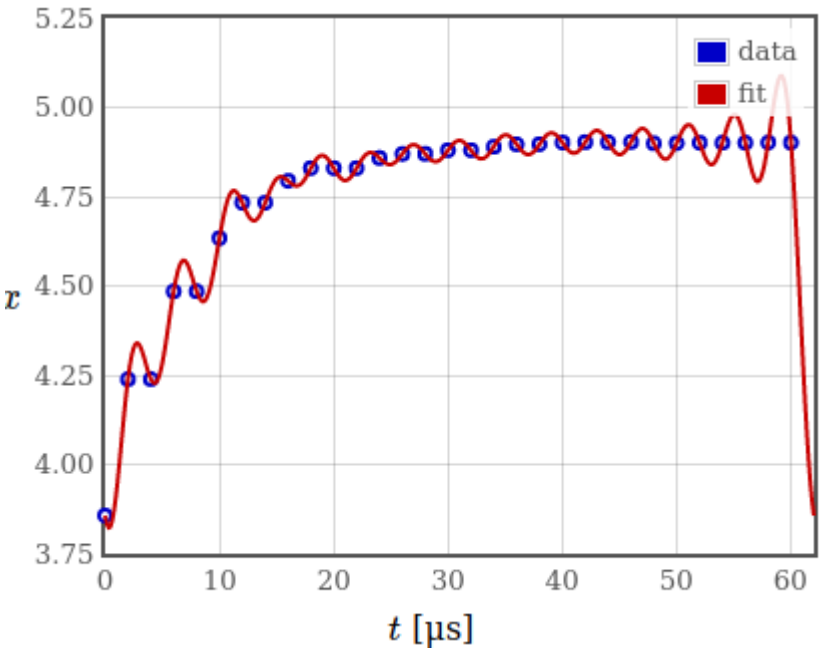
More tests were run, and the plots for the Fourier analysis will be included in the project git repository.

4.2 Compression Subsystem

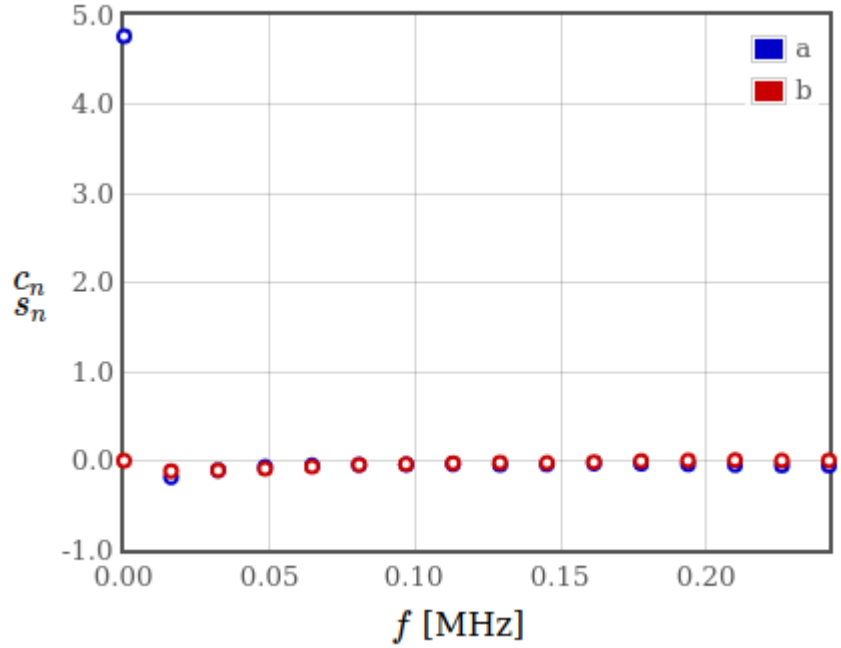
When testing the compression sub-system, a set of four tests were performed but with varying buffer size.

Test	Array size	Time taken to compress	Time taken to decompress	Compression ratio
1	8	0.01ms	0.01ms	2.1
2	16	0.01ms	0.01ms	2.1
3	32	0.02ms	0.02ms	2.2
4	64	0.02ms	0.02ms	2.3

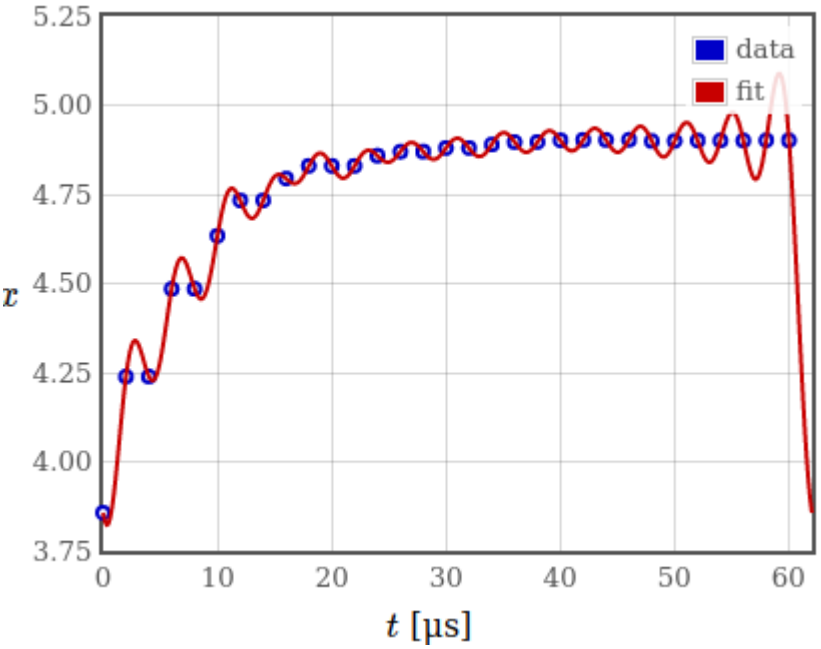
The time it takes to compress and decompress is even lower due to the decrease in the size of the data compressed.
The input data of the sub-system is then compared to the output, by means of a Fourier analysis. The data used is the accelerometer data in the in the Z axis.



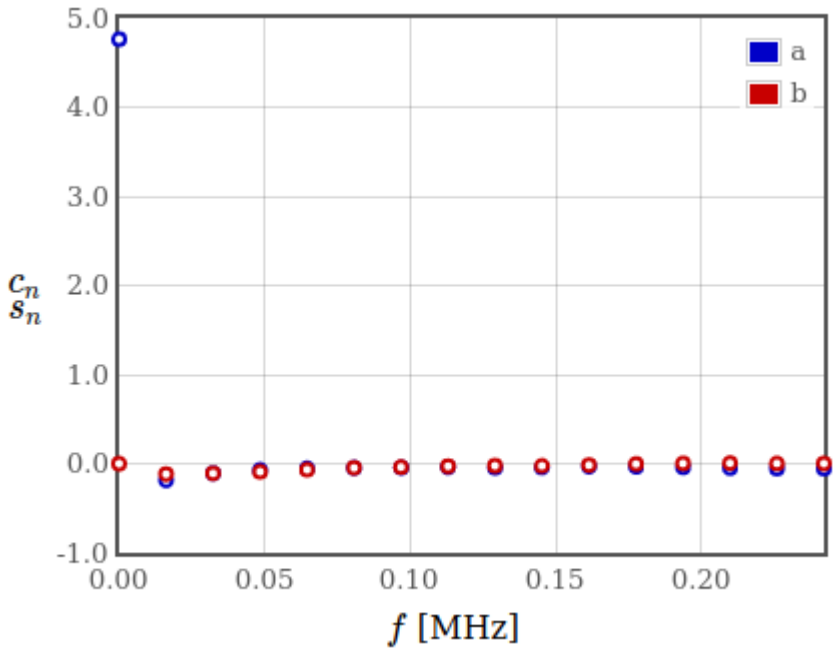
The plot above is a time domain plot of the Z axis of the accelerometer taken from the IMU and as input to the compression subsystem



The plot above is the Fourier coefficients plot from the input time domain data above.



The plot above is the output of the compression subsystem, as we can see from the plot it is as if it's the same plot.



The plot of the Fourier coefficients of the output data is shown above.

From the plots shown above, it shows that compression subsystem retains its lower frequency Fourier coefficients. From this ad other tests taken and Fourier analysis made it is safe to conclude that the compression algorithm is lossless.

4.3 Encryption Subsystem

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). For this it is also important to note that the size of the encryption files have been reduced as our IMU input data size was reduced in order to reduce the amount of work needed to be done by the STM. By reducing this we are able to utilise the STM more and not overload it which causes it to be slowed down and reduce the efficiency (speed) of our system as a whole.

Table 2.1: Encryption and Decryption time for three different file cases

Test Number	File Size in Mb	Time taken to Encrypt	Time taken to Decrypt
1	2.8	0.082 ms	0.081 ms
2	1.5	0.046 ms	0.046 ms
3	2.2	0.066 ms	0.064 ms

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. This was done using a simple online array comparison tool.

In order to test the breaksbility of the AES encryption algorithm implemented we would have to undertake a brute force approach. Using this it would take us 2^{128} complexity of exhaustive search in the fastest possible attack on the system. This is not feasible to undertake for a regular system or computer as the time to implement is too extreme.

5 Acceptance Test Procedures

5.1 Compression Subsystem

ATP from previous document:

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

For this document the ATPs are as follows:

ATP	Description	Met?
U1.F1.S1.A1	25% of Fourier coefficients present	Yes
U2.F3.S1.A1	Encoding/decoding under 2 seconds	Yes
U2.F3.S1.A1	Compression ratio greater than 2	Yes

The ATPs were all met, no changes were made.

5.2 Encryption Subsystem

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.A1	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.