FINAL REPORT

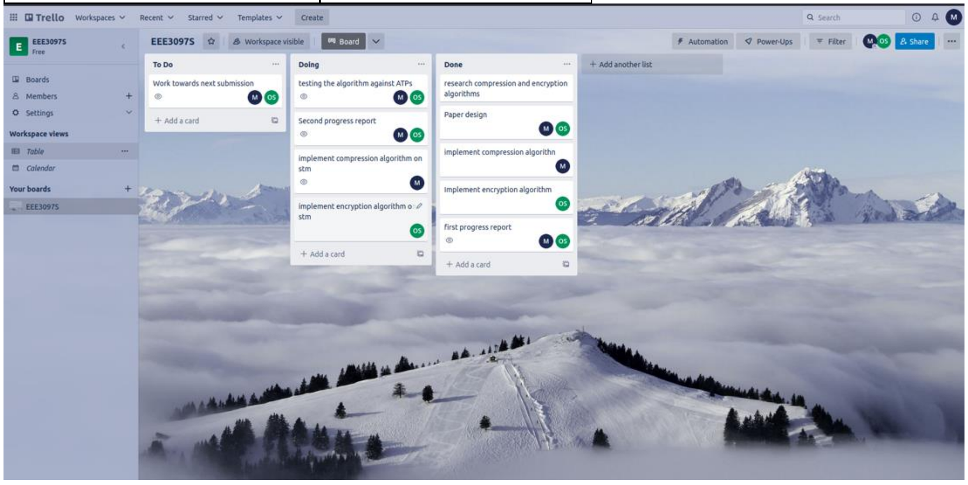
Oliver Shaw *Mbasa Mguguma*
SHWOLI002 *MGGMBA003*

1.1 Table of Contributions

Mbasa Mguguma	Compression Subsystem
Oliver Shaw	Encryption Subsystem

We both worked on everything together except our separate blocks

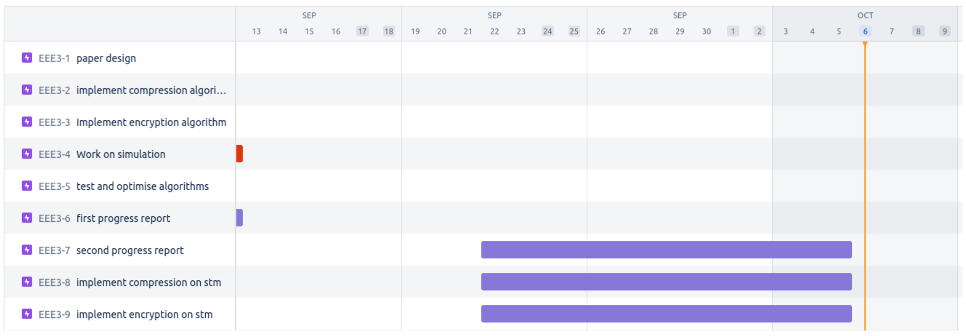
1.2 Project Management Tool



1.3 GitHub Link

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

1.4 Timeline



2 Introduction

The project is to design an ARM based digital IP using the STM32 to encrypt and compress IMU data that is from SCALE's flexible buoy system operating over the pancake ice of the Antarctic.

3 Requirement Analysis

The user requirements are derived from the project description, these user requirements identify the aim of the project and from them the specifications that the system needs to meet.

User requirement ID	Description
U1	Extract 25% of Fourier Coefficients of the data
U2	Minimize computation required
U3	Encrypt and compress IMU data

U1-Extract 25% of Fourier Coefficient of the data

The compression of data deals with decreasing the size of data by altering the data. The system's main objective is to collect as much data as it can, hence compressing the data is a viable option. Compression takes large amount of data and returns a relatively small amount of data, the compressed data. The size of this compressed data is smaller and it is easier to transmit this data, but it does not hold the information of the original data. Decompression returns the original data. Depending on the algorithm used for the compression and decompression the data might be lost.

Therefore, a compression algorithm that has minimal loss has to be implemented in order to keep the information of the original data.

U2-Minimize Computation required

The system will be operating away from a constant power supply. If the processor does a lot of computation, it will quickly drain the power supply. This is not desirable for a system that will be operating in the Southern Ocean with limited power supply. For the system to stay in operation for a reasonable amount of time, the software running on the system should be optimized to run at the shortest amount of time. This mean that the algorithm that will be implemented should run at high speed to minimize the computation required.

U3-Encrypt and compress IMU data

For the system to collect and transmit as much IMU data as it can in a cheaper manner, it must collect large amounts of data then reduce the data in size for cheaper transmission. This is possible by the use of a compression algorithm. As seen from U1, the compression algorithm must have minimal loss. To avoid the data being stolen or accessed by unauthorised individuals, it should be encrypted. When the data is encrypted the information that it holds cannot be accessed until it is decrypted.

3.1 Encryption Subsystem

Symmetric Encryption

A symmetric encryption algorithm uses just a single cryptography key for encryption and decryption. Using symmetric encryption algorithm are significantly faster than asymmetric encryption algorithms. They also require less computational power than asymmetrical encryption algorithms. This means that symmetric encryption techniques are very useful for large data sets.

Advanced Encryption System (AES)

AES uses a family of block cypher that consists of different key lengths and block sizes. This encryption technique uses substitution and permutation. During the encryption process various sub-processes are used. This includes sub bytes, row shifts, column mixes and adding round keys. This process has many rounds depending on the size of the key.

Compared to other symmetric algorithms, such as DES, AES is a faster and safer algorithm. The option to use multiple key lengths are another big advantage to using AES.

Asymmetric Encryption

An asymmetric encryption algorithm uses multiple keys for encryption and decryption. There is a distinct key for the encryption and decryption stages. This type of encryption is more secure than symmetrical encryption. The use of a private decryption key means that a system is secure against man-in-the-middle attacks. This type of encryption also has the benefit of offering authentication. This is due to the private key that ensures the only entity able to see and decrypt the data is the entity its intended for.

RSA Encryption Algorithm

This system of encryption uses prime factorisation in which two large prime numbers are multiplied together. Figuring out the original prime numbers is the encryption puzzle. The full length algorithm is virtually impossible to crack.

RSA Encryption is extremely scalable and the level of protection is based directly on the key length.

Feasibility Analysis

One necessity for the project is to reduce the power consumption of the unit. The IMU gives information about the ice as well as the wave dynamics and this data needs as many data points as possible. Due to this system being a standalone (in terms of having limited power in the system. Therefore, the smartest design decision would be to use symmetric encryption algorithm of AES. The other requirement of ensuring that at least 25% of the Fourier coefficients will also be ensured by using AES.

Possible Bottlenecks

The process of encryption is bottlenecked by the size of the microcontroller. The microcontroller is only 32 bit which means the encryption is limited in terms of data transfer as well as key size. This means that the encryption is less secure. The other bottleneck is the speed of the microcontroller. When implementing the system in the simulation phase on a laptop the speed of the process will be much faster than when actually implemented exclusively on the microcontroller.

3.2 Compression Subsystem

Compression algorithms

Since we will be using fixed point numbers, Integer compression algorithms will be efficient for the project.

Run-length encoding

This algorithm is lossless and is efficient when there is a lot of repeats in the data that is being compressed. It replaces repetitive data by keeping one copy of the data and a count [1]. But it does not take into account the number of bytes used to represent that number.

Delta

This algorithm is also lossless and it reduces the number of bytes to store a number. Instead of storing all the big numbers from a data set, it stores the differences from [1]. storing all the deltas takes up bits.

Huffman coding

This algorithm is also lossless, it checks how often symbols appear in a data set. Symbols that appear more often are encoded as short-bit string and Symbols that barely appear are encoded as long-bit strings.

For the project initially a combination of delta and Run-length encoding algorithms, was going to be used compress the data. After a searching for implementations and available resources, Huffman compression algorithm was one algorithm that is implemented in C and it also fits the requirement for an algorithm that has minimal loss of data. This will be suitable for the requirements of the project which is to keep 25% of the Fourier coefficients.

4 Subsystem Design

4.1 Encryption Subsystem

Encryption Subsystem Requirements

U1.F2	The encryption process must not lose any data
U2.F3	The encryption algorithm must be high speed
U2.F4	The encryption algorithm must be low power

Encryption Subsystem Specifications

U1.F2.S1	The algorithm must maintain 100% integrity of the data
U2.F3.S1	Encryption should take 1.5 seconds per Gb of data
U2.F3.S1	Decryption should take 1.5 seconds per Gb of data

Encryption Subsystem Acceptance Test Procedures

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

4.2 Compression Subsystem

Compression Subsystem Requirements

U1.F1	The algorithm must not lose the data
U2.F1	Use high speed algorithm
U2.F2	Use low power

Compression Subsystem Specifications

U1.F1.S1	25% of Fourier Coeffients of the data must be retained
U2.F1.S1	Compression and decompression must take not more than 2 seconds each
U2.F1.S1	Compression ratio of the algorithm should be less than 0.25

Compression Subsystem Acceptance Test Procedures

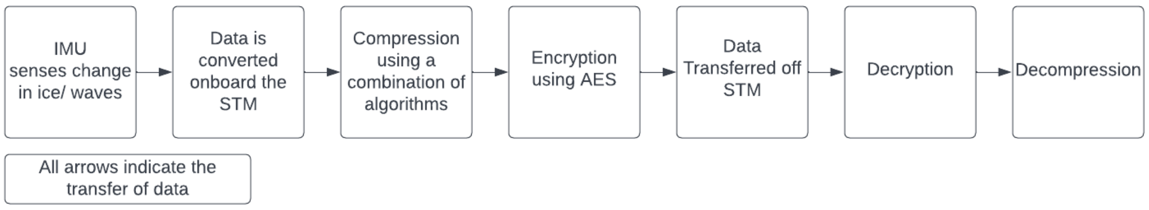
ATP	Description

U1.F1.S1.A1	25% of Fourier coefficients present
U2.F3.S1.A1	Encoding/decoding under 2 seconds each
U2.F3.S1.A1	Compression ratio greater than 2

4.3 Subsystem Interaction

Interaction protocol

For the sending of the data from the IMU to the STM we will use SPI protocol since it is faster than I2C [2]. We are prioritizing speed to minimize power usage in order to stay within the brief as well as be able to have the system running for as long as possible on a limited power supply.


IMU senses change in ice/ waves
Data is converted onboard the STM
Compression using a combination of algorithms
Encryption using AES
Data Transferred off STM
Decryption
Decompression
All arrows indicate the transfer of data

5 Validation of Simulated 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 and 3-axis magnetometer. 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. This type of data was an adequate choice as the final version of the project will use an IMU that has the same 9-axis system being implemented.

Using data that has accelerometer, gyroscope and magnetometer data is very useful for our project because these are the parameters needed for wave dynamics and the melting of ice in the Antarctic region. These different parameters can help us better understand the impacts of climate change on the Antarctic ice. Using direct data methods is very important as it allows researchers to gauge more accurately the effects of different weather phenomena in the area and their impacts on the ice. The gyroscope and accelerometer allow us to track the motion in the waves and the effects of weather events on them as well as their intensity. This use of data makes sense as we would then be able to track through different seasons as well as through different years how these weather events (and the broader consequences of climate change) are directly affecting the ice in Antarctica. Previous methods only used satellite imaging to determine changes in Antarctic ice but using direct measurement methods is a great advance in the discovery of the effects. This is why we have decided to use data that reflects this by including gyroscope and accelerometer data. Our choice in data also reflects a similar type of data as we have chosen data of a moving person. This is similar to the effects seen on ice as the data has phases of movement (both vertically in terms of standing as well as walking data) and phases of static data. This is important as it reflects the type of data, we may see from the IMU on the pancake ice in Antarctica that moves and remains static depending on waves as well as weather.

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. This data can give us an accurate set of data that is similar to what we can expect from the IMU that would be situated on the implemented project on the Antarctic pancake ice. The second is a 10-minute-long recording of the data of the IMU rotating about one of its axes. This data can show us the impacts of wave movement on the IMU and ensure that our implemented system works on this data as well. This file is taken at a high sample rate. The final data file is the same as the second data file, but the sampling rate is lower. Using different data sampling rates allows us to ensure that the sampling rate doesn't impact our system output in a way that may hinder the project in the full implementation phase.

In reference to our acceptance test procedures, we need to ensure that our data maintains 25% of its Fourier coefficients. To ensure this we have chosen a longer as well as shorter period of data as well as data sampled at higher and lower rates. These will allow us to make sure that when we implement both the compression as well as encryption, we do not lose important data that would affect the findings.

We also need to ensure that our system performs fast enough so that it can be implemented on the MCU and not require too much power that means that the final implementation isn't feasible as it needs to be able to perform using only an onboard battery and be able to fetch data for long periods of time. By using different sets of data we are able to evaluate the different levels of power usage and determine if it will be feasible in the final implementation.

5.1 Experiment Setup

5.1.1 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.

5.1.2 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 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 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.

5.1.3 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 algorithm.

6 Results

6.1 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.

6.2 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
4	10.4	15.60 ms	15.56 ms
5	15.9	24.73 ms	24.73 ms
6	26.8	40.20 ms	40.18 ms
7	29.2	43.78 ms	43.80 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.

6.3 Compression Block

Test number	File size (Mb)	Compressed file size	Compression ratio	Time taken to compress (s)	Time taken to decompress (s)
1	22.4	9.7	2.31	1.16	0.6
2	19.8	8.7	2.28	0.98	0.59
3	16.2	7.6	2.13	0.85	0.52
4	26.3	11	2.39	1.25	0.63
5	20.2	9	2.24	1.03	0.58
6	24.5	10.3	2.37	1.22	0.61
7	18.4	8.1	2.27	0.89	0.55

The input csv files and the output csv files from all the experiments have the same size, meaning the content of the files is more or less the same, to verify this an online resource to check the differences from files was used and it found no differences from all the files.

7 Validation using the IMU Module

7.1 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 ±2000 degrees per second for its gyroscope meaning it has 16.4 least significant bits per degree per second whereas the ICM-20649 has ±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 ±16G with a sensitivity scale factor of 2048 least significant bits per G whereas the ICM-20649 has Full-Scale Range of ±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.

7.1.1 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 homemade 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.

7.1.2 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:

<u>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</u>

	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.

7.1.3 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 we will attempt to swing in a single direction. From this test we were able to see both the IMU data changing dynamically as the IMU was swung from the rest position to its maximum at the bottom of the pendulums arc. We were also able to see that as the IMU reached the top of its pendulum arc that the IMU data starting trending towards zero and at the peak of its arc we were able to see that the data reached almost the base values we were expecting. From this test we were able to see that the IMU was in working order.

7.2 Experimental Setup

7.2.1 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 transmitted to a laptop using serial communication where the data is put in a file for processing. The file is then sent to each of the blocks for compression, encryption, decryption and then finally decompression.

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 take note of the time 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.

7.2.2 Compression Subsystem

The compression algorithm the system uses is Huffman's compression algorithm which is implemented in C. The goal of this experiment is to check whether the compression block runs in the shortest possible time, each under 2 seconds according to the acceptable test procedures. The block takes in a file that consists of data collected from the IMU by the STM and it compresses the data. The compression of this data should yield a compression ratio of more than 2, meaning the size of the compressed file should be less than half of the original file.

The experiment also involves varying the size of the data. This is to ensure that the compression block yields the desired results at different cases and for us to take note of cases where the results deviate from the expected results. Since the STM is used to collect the data, smaller data sets are being used for the experiment.

To check for functionality, a number of tests were run, where the input to the subsystem, the data read from the IMU, is compared to the output, the data decompressed on the laptop. For this to be done properly and a solid conclusion to be drawn, 5 tests cases were run. For each of these cases the size of the input data to the subsystem was different. This comparison was done using Fourier analysis and an online resource that checks difference on a file.

The subsystem was timed to determine its speed and to roughly approximate the power consumption of the subsystem. The timing of the subsystem is implemented in code. The desired runtime of the subsystem is specified as one of the acceptable test procedures.

7.2.3 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.

7.3 Results

7.3.1 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:

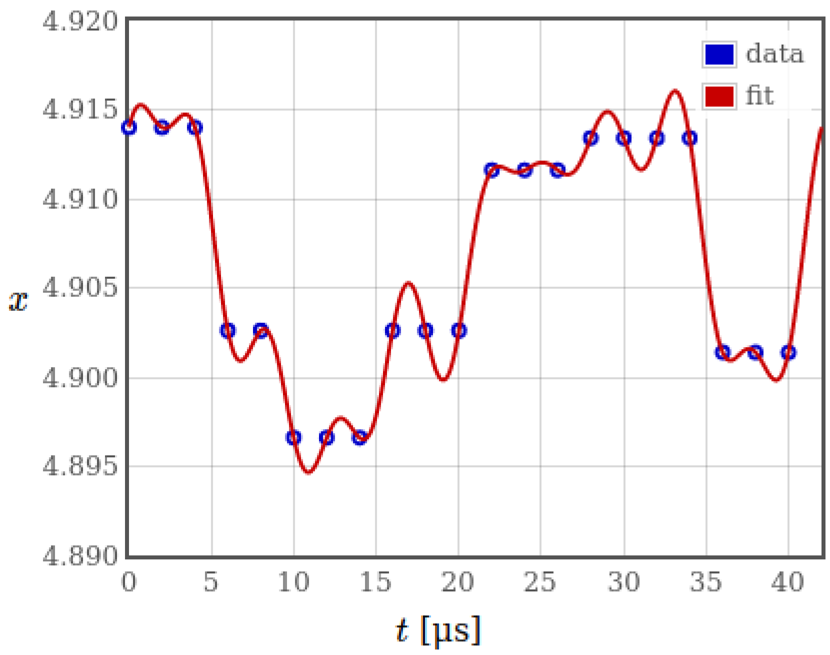


Figure 1.1: showing the input time domain array of the accelerometer in the Z-axis

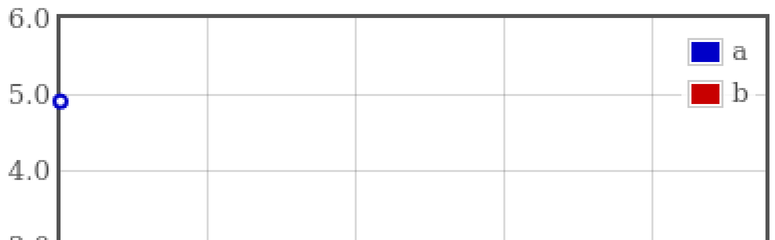


Figure 1.2: showing the discrete Fourier transform showing the Fourier coefficients of the data

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

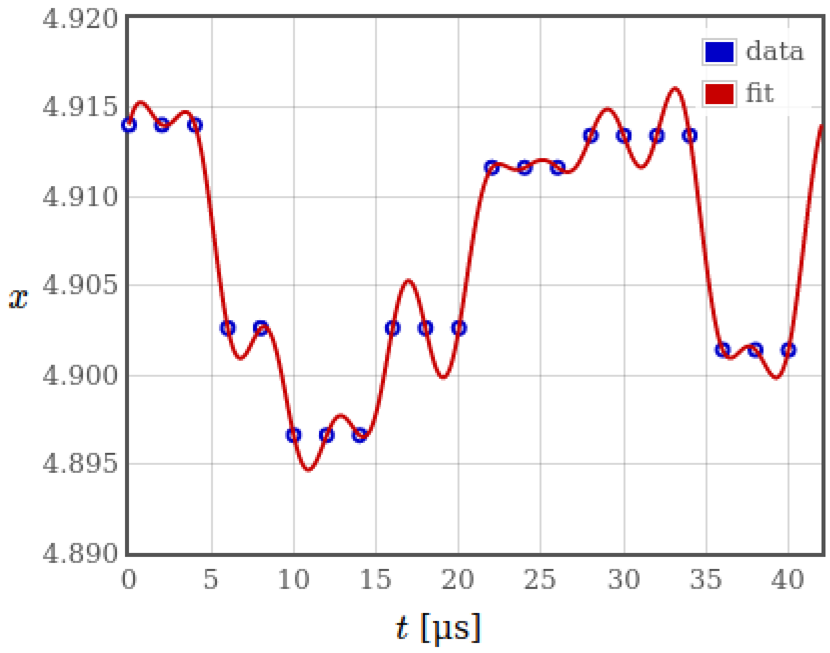


Figure 1.3: showing the time domain plot of the output data

From comparison to the input data, we can see that the output data is identical.

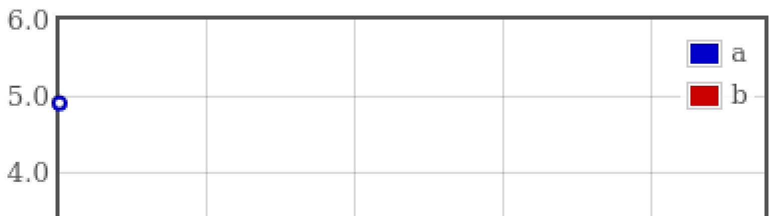


Figure 1.4: showing the discrete Fourier transform which shows the Fourier coefficients of the output data

From comparison to the input data, we can see that the output data is identical.

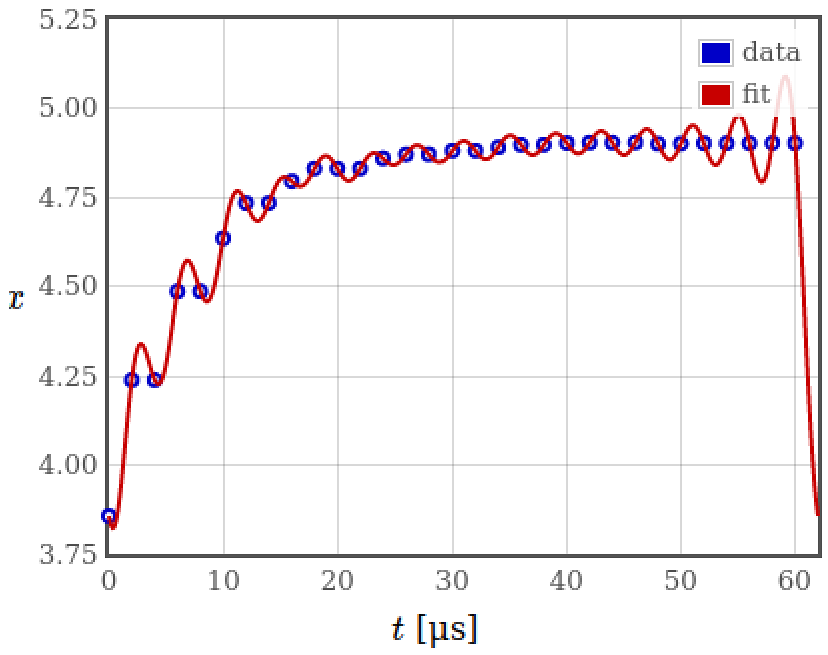
More tests were run, and the plots for the Fourier analysis will be included in the project git repository.

7.3.2 Compression Subsystem

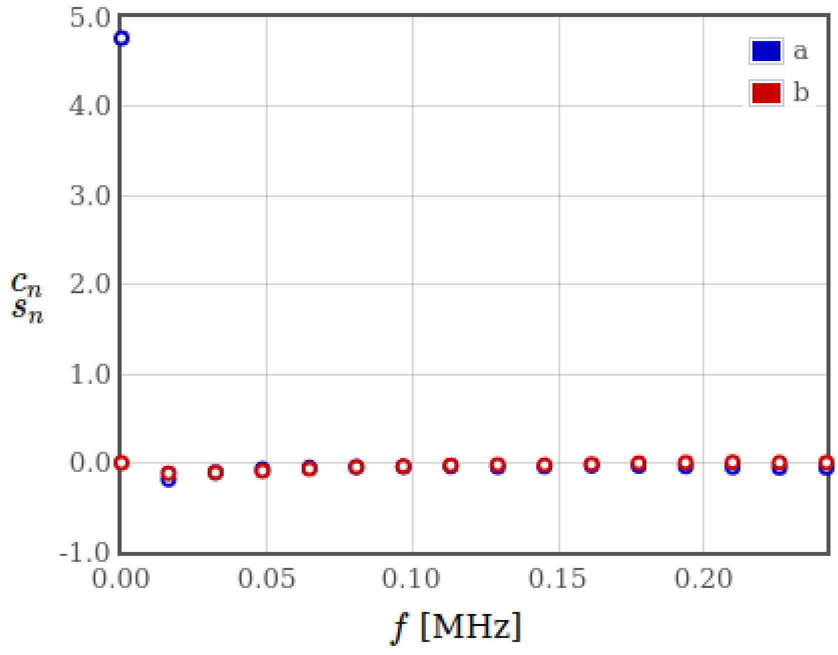
When testing the compression sub-system, a set of five tests were performed but with varying buffer size.

Test	File size (Mb)	Compressed file size (Mb)	Time taken to compress	Time taken to decompress	Compression ratio
1	4.4	2.1	78ms	41ms	2.1
2	3.5	1.7	63ms	33ms	2.1
3	6.4	2.9	109ms	56ms	2.2
4	7.3	3.2	113ms	79ms	2.3
5	8.1	3.5	127ms	83ms	2.3

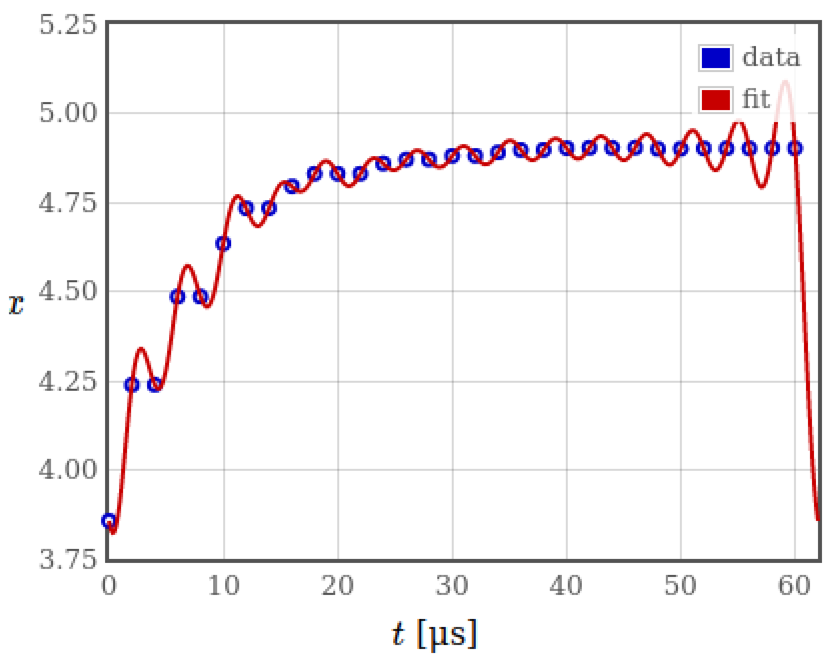
The time it takes to compress and decompress is even lower due to the decrease in the size of the data compressed.
The input data of the sub-system is then compared to the output, by means of a Fourier analysis. The data used is the accelerometer data in the in the Z axis.



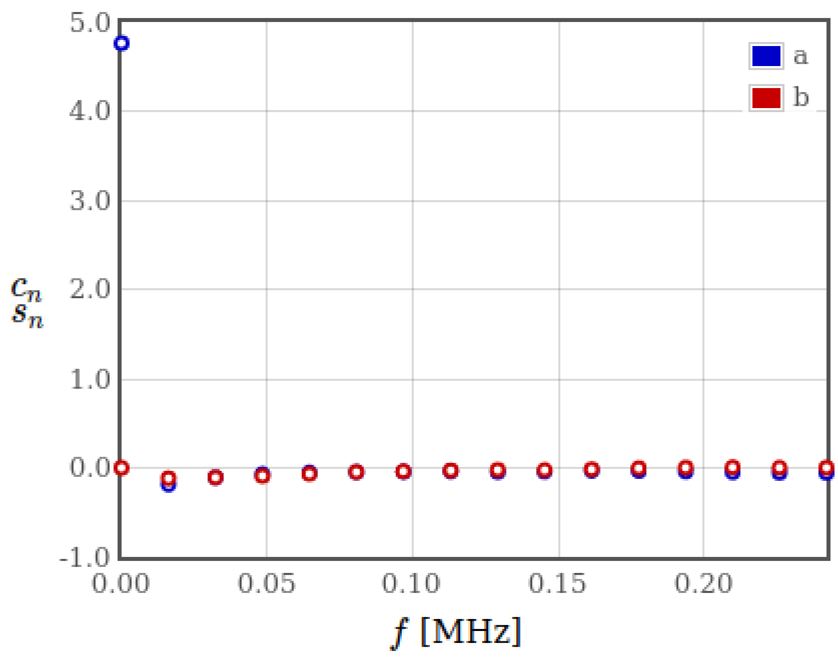
The plot above is a time domain plot of the Z axis of the accelerometer taken from the IMU and as input to the compression subsystem



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.

7.3.3 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
4	10.4	15.60 ms	15.56 ms
5	15.9	24.73 ms	24.73 ms
6	26.8	40.20 ms	40.18 ms
7	29.2	43.78 ms	43.80 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 breakability 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.

8 Acceptance Test Procedures and Future Plan

8.1 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.

8.2 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 as when compared to the pre-encryption case.

Our module would need to be able to take constant streams of data as opposed to our array based method or be have data transfer where the data is sent and used as arrays which can be processed easily by our system. For our system would need to implement its own battery instead of being powered by a laptop. This is important as an integral part of this project is ensuring that it is a low power device that is able to work for long periods of time and be able to gather data that has meaningful use. According to the use case outlined the actual implemented system may have to last a month without interruption. The IMU we have on our system has a sleep mode that it can be placed in if the device is inactive. Testing this system is necessary before full scale implementation. In order to ensure further that it is able to hold enough data for it to be meaningful it is also needs to have enough memory. This may include adding an extra memory unit. Our system also only allows for on board memory in order to lower the power usage when compared to a satellite based system for sending data back for analysis without interrupting the system. Another assurance we have to make is that the system is able to work at the temperatures that are expected in Antarctica. This may reach as low as -40° C. In order to do this further testing must be done ensuring that all parts of the system operate in their ideal way at these temperatures. Another assurance we have to make is that our system can operate in a wet environment. This may include testing further systems that hold the IMU and STM set up securely as well as keep them dry. A further test we could implement is using desiccant inside our housing system in order to maintain dryness. Our system also doesn't have anything to monitor power usage. To implement this system effectively we would want a way of actively assessing if our system is using the correct amount of power and not overusing power unexpectedly.

Conclusion

In conclusion, our system reads 3-axis accelerometer and 3-axis gyroscope data through the IMU which is sent to the MCU where the data is compressed, encrypted, decrypted and then decompressed. The system comprises of a compression subsystem and an encryption subsystem. The IMU was validated using static dynamic testing in order to ensure that the IMU was in working order and producing results that were accurate. The system is designed to be implemented in the future on the Antarctic pancake ice in order to detect wave dynamics and the effects of climate change on the ice. The system we have uses the Sense HAT (B) IMU to gather the accelerometer and gyroscope data as well as the STM32 Discovery board for the compression and encryption of the data. Our system is designed to be a low power system with onboard memory for data storage. We ensured that it is low power by optimising each subsystem to run as quickly and efficiently as possible. We needed to ensure that the data was compressed with a compression ratio of greater than 2 in order to ensure that the amount of data that the encryption system works on stays as low as we could. This helps to reduce the workload needed to be done by the encryption system. We also ensured that the compression subsystem completed it workload within two seconds. Furthermore, we wanted the encryption subsystem to be encrypting/ decrypting a gigabyte of data per 1.5 seconds. Beyond this we needed to ensure that we would keep a good standard of data integrity in order to ensure that the data that the system outputted was usable for research purposes. In order to achieve this, we wanted to make sure we were able to keep 25% of the Fourier Coefficients from our input phase to the outputted data. To ensure this within the encryption section we wanted 100% data integrity within the subsystem. We were able to achieve this. We achieved this and therefore ensured data integrity stayed at our desired level. The project as it could move forward in the future would see it being implemented on the pancake ice in Antarctica. The current system needs some modifications in order to ensure that it is working and a feasible options for use in Antarctica.