



# EEE3097S Final Report

## October 2022

Group Member Name	Student Number
Kimmy Sithole	STHKIM002
Matshego Kgafela	KGFMAT002

## Contents

1. Administration Documents .....	4
1.1. Contribution table .....	4
1.2. Project management software screenshot .....	5
1.3. Link to GitHub Repository .....	5
1.4. Timeline .....	5
1.5. Analysis of Timeline .....	6
2. Introduction .....	7
3. Requirement Analysis .....	8
3.1. Requirement Derivation and Analysis .....	8
3.1.1. Analysis of UR001 .....	8
3.1.2. Analysis of UR002 .....	9
3.1.3. Analysis of UR003 .....	9
3.1.4. Analysis of UR004 .....	9
3.1.5. Analysis of UR005 .....	9
3.1.6. Analysis of UR006 .....	10
3.1.7. Analysis of UR007 .....	10
3.1.8. Analysis of UR008 .....	10
3.2. Specifications .....	10
3.3. Acceptance Test Protocols .....	12
4. Paper Design .....	14
4.1. Available Compression Algorithms .....	14
4.1.1. Integer Compression .....	14
4.1.2. Float Compression .....	15
4.1.3. Any Data type compression .....	15
4.2. Available Encryption Algorithms .....	15
4.2.1. Advanced Encryption Standard (AES) .....	15
4.2.2. Triple Data Encryption Standard (3DES) .....	16
4.2.3. Blowfish algorithm .....	17
4.3. Chosen Algorithms .....	17
4.3.1. Compression .....	18
4.3.2. Encryption .....	18
4.4. Subsystem Design .....	18
4.4.1. Graphical Representation of inter-subsystem design .....	19
5. Validation using Simulated or Old Data .....	21
5.1 Importance for Simulation based validation .....	21
5.2 Steps for validation using old data .....	21
5.3 Data .....	22
5.4 Discussion of data .....	22

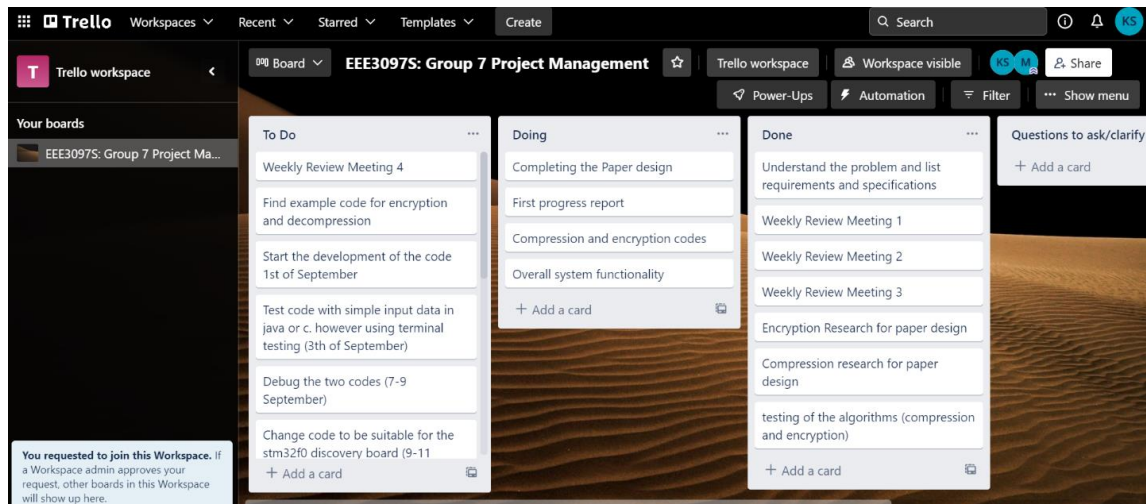
5.5 Validity of data .....	22
5.6 Graphical Representation the Overall data.....	23
5.7 Analysis of the overall data .....	29
5.8 Steps followed for collecting and formulating the data .....	29
5.9 Experimental setup .....	30
5.9.1 Overall system .....	30
5.9.2 Compression block .....	30
5.9.3 Encryption Block.....	31
5.10 Results .....	32
Overall system .....	32
Compression block .....	38
Encryption Block.....	40
6. Validation using IMU .....	41
Importance of hardware-based validation .....	41
Steps for validation using data from IMU in real time .....	41
IMU Module .....	41
Salient features on ICM 20948 .....	41
Steps to take to ensure ICM 20948 can be extrapolated to ICM 20649 .....	42
Validation tests for ICM 20948.....	42
Results for ICM 20948 .....	43
Analysis Results for ICM 20948 .....	44
Compression block .....	44
Experimental setup .....	44
Results .....	44
Analysis of results .....	46
Encryption block.....	46
Experimental setup .....	46
Results .....	47
Consolidation of ATPs and Future Plan .....	49
Recreated ATPs.....	49
ATPs met or not .....	51
Future work before implementation .....	52
Conclusion .....	52
References.....	53
Appendix.....	55
Appendix A .....	55
Appendix B.....	56
Appendix C.....	58

# 1. Administration Documents

## 1.1. Contribution table

<b>Kimmy (STHKIM002) - COMPRESSION</b>	<b>Matshego (KGFMAT002)- ENCRYPTION</b>
<b>Compression experiment setup</b> – Discussed how the simulations/tests of the compression block will be performed.	<b>IMU Module</b> – Worked on the description of the IMU module. This includes listing the modules features, anticipated functionality tests and how we will extend its functionality to the ICM20948
<b>Compression experiment results</b> – provided and analyzed the results of the compression block tests/simulations.	<b>Encryption experiment setup</b> – Discussed how the simulations/tests of the encryption block will be performed.
<b>ATPs</b> – Worked together with Matshego on the refined Acceptance Test Procedures and descriptions of whether these tests have been met or not. This includes the implications of the tests not being met	<b>Encryption experiment results</b> – provided and analyzed the results of the encryption block's tests/simulations.
<b>Overall system experiment setup</b> - Worked together with Matshego on the experiments that will be performed to check the functionality of the overall system. Specifically, worked more on the compression section of the overall system experiment.	<b>ATPs</b> - Worked together with Kimmy on the refined Acceptance Test Procedures and descriptions of whether these tests have been met or not. This includes the implications of the tests not being met.
<b>Overall system experiment results</b> - Worked on checking that the compression section of the overall section manages to compress input data. This includes providing and analyzing graphical results.	<b>Overall system experiment setup</b> – Worked together with Kimmy on the experiments that will be performed to check the functionality of the overall system. Specifically worked more on the encryption section of the overall system experiment.
<b>Validation hardware-based-</b> Worked together with Matshego for hardware-based tests and results.	<b>Overall system experiment results</b> – Worked on checking that the encryption section of the overall section manages to encrypt compressed data. This includes providing and analyzing graphical results
<b>Validation Simulation-based</b> Worked with Matshego for this section of the report	<b>Validation hardware-based-</b> Worked together with Kimmy for hardware-based tests and results.
<b>Requirements analysis</b> Worked with Matshego on this part of the document	<b>Validation Simulation-based</b> Worked with Kimmy for this section of the report
<b>Paper design</b> Worked with Matshego on this part of the document	<b>Paper design</b> Worked with Kimmy on this part of the document

## 1.2. Project management software screenshot



*A screenshot of the updated Trello project management software utilised*

## 1.3. Link to GitHub Repository

[KimmySithole/EEE3097S\\_Compression\\_Encryption\\_Project: An ARM based digital IP using the STM32F0 to encrypt and compress the IMU data \(github.com\)](https://github.com/KimmySithole/EEE3097S_Compression_Encryption_Project)

## 1.4. Timeline

To Do	Doing	Done
Final report by the 17 of October 2022	Final report	Understand the problem and list requirements and specifications
Final video demonstration by the 17 of October 2022	Reading data from IMU on the stm32 and writing data into a txt file	Weekly Review Meeting 1
	Using alternate compression algorithm	Weekly Review Meeting 2
	Final video demonstration	Weekly Review Meeting 3
		Weekly Review Meeting 4
		Paper design
		First progress report
		Weekly Review Meeting 5
		Weekly Review Meeting 6
		Weekly Review Meeting 7
		Weekly Review Meeting 8
		Weekly Review Meeting 9
		Second progress report

## 1.5. Analysis of Timeline

The table above shows the timeline for this project. We were supposed to be done with the second progress report on the 5<sup>th</sup> of October, however we are a day behind and still busy with it. The implementation of the encryption and compression algorithms onto the stm32 is a bit more complex and therefore taking longer than expected hence we are still implementing that part and working on it. Reading the IMU data from the stm32 is also taking longer with unexpected errors hence we are still busy with that. The goal was then to have everything working on the 12<sup>th</sup> of October 2022. The project was then extended to the 17<sup>th</sup> of October. Stm32 implementation was also changed to not be a requirement when it comes to encryption and compression. On the 12<sup>th</sup> of October we found a new compression algorithm as the previous one was giving problems. Our goal changed to focusing on implementation on PC and stm32 only for the IMU. The weekly reviews are all done one time and completed. We have also completed the Paper design and the first progress report. Overall, our progress is delayed, however we are pushing for everything to be done on the 17<sup>th</sup> of October 2022.

## 2. Introduction

Remote sensing is the use of an autonomous device in a remote area to communicate with another network wirelessly. Remote sensing is critical in the Southern Ocean Seasonal Experiment's (SCALE) endeavour to understand various environmental characteristics of the marginal ice zone in the Antarctic. The University of Cape Town's (UCT) electrical engineering research group part of SCALE have built a remote sensing buoy that obtains data from the marginal ice zone. This buoy has been termed SHARC buoy and is built with an Inertial Measurement Unit (IMU) and a host of other sensors. The data from the sensors is transmitted through an iridium satellite modem. Transmission through an iridium modem is, however, quite costly. Therefore, the research group would like to compress the data from the IMU and, furthermore, encrypt the data. Our task is to design an ARM based digital IP (Intellectual Property) to compress and encrypt the data from the IMU to alleviate the costs of iridium modem transmission and ensure the data is protected in transit.

## 3. Requirement Analysis

### 3.1. Requirement Derivation and Analysis

The user requirement analysis was instigated with a review of the brief user requirements stipulated in the projects design requirements. From there, a table of requirements was created. This table is represented in Table 1 below.

*Table 1: User Requirements table*

User requirement ID	Description
<b>Compression subsystem requirements</b>	
UR001	The system should be able to successfully compress the IMU data
UR002	The system should be able to successfully decompress the compressed data
<b>Encryption subsystem requirements</b>	
UR003	The system should be able to successfully encrypt the IMU data
UR004	The system should be able to successfully decrypt the encrypted data
<b>Whole system requirements</b>	
UR005	25% of the Fourier coefficients of the IMU data should be maintained throughout the encryption and compression process
UR006	The accuracy of the data should be maintained throughout the encryption and compression process
UR007	The system should work on minimal processing power due to the SHARC buoy working on limited power
UR008	The system should be able to work for both the ICM-20948 and ICM-20649

#### 3.1.1. Analysis of UR001

*“The system should be able to successfully compress the IMU data”*

Appendix A showcases the problem statement and design requirements of the overall system. Part of the requirements is to “compress the IMU data” which is transmitted via the iridium satellite modem. This is to reduce the costs of data transfer. The first



user requirement is, therefore, to compress the IMU data. This should be done successfully without fault/error in the transition from raw data to compressed data

### 3.1.2. Analysis of UR002

*“The system should be able to successfully decompress the compressed data”*

Since the compressed data will need to be used for further analysis by the SCALE research group, the data will need to be decompressed after transmission and receipt. This motivates the need for the second user requirement to decompressing the compressed data successfully without fault/error

### 3.1.3. Analysis of UR003

*“The system should be able to successfully encrypt the IMU data”*

Part of the design requirements in Appendix A is to “encrypt the data.” This motivates the third requirements. The system should be able to encrypt/plain text raw data such that it is not easily decipherable in its transmission. This should be done successfully with error/fault in the process.

### 3.1.4. Analysis of UR004

*“The system should be able to successfully decrypt the encrypted data”*

Since the data is needed for further analysis, it will need to be in a form that is understandable. The system, should also be able to decrypt the data successfully such that it can be analysed by SCALE.

### 3.1.5. Analysis of UR005

*“25% of the Fourier coefficients of the IMU data should be maintained throughout the encryption and compression process”*

The SCALE “Oceanographers have indicated that they would like to be able to extract at least 25% of the Fourier coefficients” as indicated in Appendix A. This is

to ensure that some of the frequency domain data is not lost throughout the data transmission process. This motivates the inclusion of UR005.

### 3.1.6. Analysis of UR006

*“The accuracy of the data should be maintained throughout the encryption and compression process”*

Since the data will be needed for further research, it is imperative that, most of the data is preserved throughout the compression and encryption process. The data used could be sensitive to the compression-encryption process, therefore, it is important to ensure that it does not get altered as it traverses through the process.

### 3.1.7. Analysis of UR007

*“The system should work on minimal processing power due to the SHARC buoy working on limited power”*

The SCALE researches have indicated that they “want to reduce the amount of processing done in the processor,” as stipulated in Appendix A. By ensuring that the system works on minimal processing power, the amount of processing done in the processor is limited, therefore, reducing the overall processing done in buoy’s processor.

### 3.1.8. Analysis of UR008

*“The system should be able to work for both the ICM-20948 and ICM-20649”*

The design requirements have indicated that the IMU on the SHARC buoy is the ICM-20649. However, we will not get this IMU, rather the Waveshare Sense HAT which incorporates the ICM-20948. The designed system should be able to work for the ICM-20649. Therefore, it is imperative that the system works for both IMU’s to fulfil this.

## 3.2. Specifications

After careful analysis of the user requirements, the following specifications were derived:

Table 2: Specifications table

Specification ID	Description	User Requirement Addressed
SP001	The designed system should work for both the ICM-20649 and the ICM-20948	UR008
SP002	When performing the Discrete Fourier Transform (DFT) on the decompressed data, at least 25% of the Fourier coefficients of the input data should be present	UR005
SP003	At least 90% of the accuracy of the input data into the system should be successfully maintained after decompression and decryption	UR002 UR004 UR006
SP004	The system should use less than 50% of the total power of the microcontroller on the buoy. It has to use as minimal power as possible.	UR007
SP005	The system should be able to encrypt 100% of the data such that the raw data file and encrypted file have 0% resemblance	UR003
SP006	The system should be able to compress the IMU data such that the compressed file's size is 60% of the original data.	UR001

### 3.3. Acceptance Test Protocols

A summary of the Acceptable Test procedures is presented in Table 3.

*Table 3: Summary of Acceptance Test Protocols*

ATP ID	ATP Name	Specification Addressed
ATP001	Accuracy Test	SP003
ATP002	Encryption Test	SP005
ATP003	Compression Test	SP006
ATP004	Frequency Test	SP002
ATP005	Low power data retrieval	SP004
ATP006	Low power system	SP005

<b>ATP Name</b>	ATP001: Accuracy Test
<b>Description</b>	This test checks whether the decrypted and decompressed data is similar to the original input data to the system. A python script will be written to compare the two files. The script will output a binary True/False after the comparison
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The decrypted/decompressed data receives a <b>True</b> when compared to the input data to the system
<b>Fail Conditions</b>	The decrypted/decompressed data receives a <b>False</b> when compared to the input data to the system

<b>ATP Name</b>	ATP002: Encryption Test
<b>Description</b>	This test checks whether a file has been successfully encrypted. The same python script used for ATP001 will be used for this test
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The encrypted data receives a <b>False</b> when compared to the input data to the system
<b>Fail Conditions</b>	The decrypted/decompressed data receives a <b>True</b> when compared to the input data to the system

<b>ATP Name</b>	ATP003: Compression Test
<b>Description</b>	This test checks whether a file has been successfully compressed. After compression, a compression ratio will be outputted

<b>Figure of Merit</b>	Compression saving space of atleast 40%
<b>Pass Conditions</b>	The compressed file has a compression space saving of 40% or more
<b>Fail Conditions</b>	The compressed file has a compression space saving of less than 40%

<b>ATP Name</b>	ATP004: Frequency Test
<b>Description</b>	This test checks whether 25% of the Fourier coefficients have been preserved. For this test the DFT of the original data is compared to the DFT of the decompressed/decrypted data. The same python script that has been used for previous test will be used.
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The decrypted/decompressed data receives a <b>True</b> when compared to the input data DFT. This signifies that 100% of the Fourier coefficients have been preserved.
<b>Fail Conditions</b>	The decrypted/decompressed data receives a <b>False</b> when compared to the input data DFT of the system to show that 0% of the Fourier coefficients have been preserved.

<b>ATP Name</b>	ATP005: Lower Power data retrieval
<b>Description</b>	This test checks that the communication between the stm32f0 discovery board and PC is minimal. This is done through inspection by checking the CPU usage of the PC
<b>Figure of Merit</b>	Numerical figure of 10%
<b>Pass Conditions</b>	The communication between the PC and microcontroller should less than 10% of the CPU usage on the PC
<b>Fail Conditions</b>	The communication between the PC and microcontroller uses more than 10% of the CPU usage on the PC

<b>ATP Name</b>	ATP006: Low Power System
<b>Description</b>	This test is similar to ATP005. Instead of monitoring the power usage, it monitors the power usage of the encryption/compression algorithms on the PC. This is also done via inspection of the CPU usage.
<b>Figure of Merit</b>	Numerical value of 50%
<b>Pass Conditions</b>	The encryption and compression algorithms use less than 50% of the CPU usage
<b>Fail Conditions</b>	The encryption and compression algorithms use more than 50% of the CPU usage.

## 4. Paper Design

### 4.1. Available Compression Algorithms

#### 4.1.1. Integer Compression

The algorithms below are light weight compression algorithms that compress integer data values. These algorithms help with reducing storage size while also helping with the reducing of computational power when running the algorithms.[5]

##### *Delta Encoding*

Reduces amount of information required to represent certain data by storing the difference with the one or more of the reference data. This algorithm works well where there is a lot of similar data.[5]

##### *Delta-of-Delta Encoding*

This algorithm is the same as delta-encoding, however the only difference is that this algorithm subtracts the difference on two levels instead of one. This reduces size a lot more than the delta-encoding.[5]

##### *Simple-8b*

This algorithm further compresses in that the storing of the difference, the size of storage allocated to it will vary depending on what the difference is. Therefore, further compressing the data. This can be applied on its own, however it is more effective with delta having been applied before.[5]

##### *Run-length encoding*

This algorithm is the furthering of the simple-8b in that the repeated numbers are grouped and stored in value and data instead of storing all the repeated data as individuals. RLE is a classic compression algorithm, there is even simple –8b RLE which combines the two algorithms.[5]

### 4.1.2. Float Compression

#### *XOR based*

Since not all data is integer. One can compress float data using the above integer algorithms. This, however, is usually messy and results in lossy data. XOR- based is a float compression algorithm which does float compression more efficiently in that the float point numbers are stored in that only the difference in them is stored. The first floating point number is used as the reference. This algorithm improves the integer compression delta encoding [5]

### 4.1.3. Any Data type compression

#### *Dictionary*

Dictionary compression compresses all data types. This is very reliant on the redundancy of data to work well. Therefore, dictionary compression works well with data with lots of repetitions. This algorithm works in that it creates a dictionary with all the possible values or data that can appear. It is that the data to be compressed is given an index of the dictionary where it appears. An example of dictionary algorithms is LZW that uses string character representation. Other examples are LZ77 and LZ78. Many more examples are available for this type of compression as this is one of the classic compression algorithms used [5].

## 4.2. Available Encryption Algorithms

### 4.2.1. Advanced Encryption Standard (AES)

The Advanced Encryption Standard (AES), also called the Rijndael algorithm, is a symmetric encryption algorithm that encrypts 128-bit data blocks with encryption keys that are either 128 bits, 192 bits or 256 bits. The input data goes through several cycles of encryption and these cycles depend on the key sizes being used. For a 128-bit key, the input data will go through 10 cycles of encryption. For a 192-bit key, the data will go through 12 cycles of encryption and finally for a 256-bit key the data will go through 14 cycles of encryption. In every cycle of encryption, the data will traverse through 4 rounds, namely: sub-bytes, shift-rows, mix-columns and finally add-round-key. [9]

Every cycle of encryptions begins with the sub-bytes action. Here, every byte of the 128-bit input data is traversed through as the name suggests. As the algorithm traverses through every byte, the bytes are divided into nibbles, therefore, resulting in 2 nibbles for every byte. Each nibble is then replaced with a corresponding nibble on a look up table. These replacements occur for every nibble in the input data. [9]

After the input data has passed through sub-bytes it will then enter shift-rows. The shift-rows operation traverses through all the rows of the 128-bit block the comes out of the sub-bytes operation. As it traverses through every row, it shifts the rows based on an offset. The offset is dependent on the row number being shifted. [9]

After the shift-rows operation has been completed, the data then enters mix-columns. Here every column of the input data block is mixed up as the name suggests. To do this, each column is multiplied with a matrix and the outcome of that matrix is a new column that will replace the old column. [9]

Finally, after the input data's nibbles, rows and columns have been manipulated, the data goes through the add-round-key operation. Here, a logical XOR operation is performed between the manipulated data and the user generated key. After this is performed the data will enter a new cycle depending on the cycle number it is on. [9]

#### *Advantages of the algorithm*

- It uses high key sizes (128, 192 and 256), therefore, making it difficult to penetrate encrypted data. [8]
- It is one of the most common encryption algorithms, therefore, making it a credible and trustworthy algorithm. [8]
- Since it is commonly used there are many open-source resources that can be used to implement it. [8]

#### *Disadvantages of the algorithm*

- Every block of data is encrypted the same way. Therefore, whilst unlikely, a brute force attack with much more powerful computers than today's standard would find the key. [8]
- There are many mathematical operations that are performed in this algorithm. This demands a considerable amount of computing power. The algorithm might be heavy for usage on a micro controller with limited power. [8]
- For hardware purposes that require real-time responses, it could be a little slow due to the mathematical operations being performed.
- 

### 4.2.2. Triple Data Encryption Standard (3DES)

The Triple Data Encryption Standard (3DES) is an improved version of the old Data Encryption Standard algorithm (DES). It is also a symmetric block cipher encryption algorithm which was designed due the DES being increasingly susceptible to brute force attacks. When 3DES is used, the input data block goes through 3 cycles. First the data is encrypted with a 56-bit key. Next data is decrypted with another 56-bit key. Finally, the data is encrypted again with a third 56-bit key. [4]



#### *Advantages of the algorithm*

- Easy to implement as it only has 3 cycles of operation [1]
- There are open-source resources that can be used to aid implementation [1]
- Does not need much computing ability due to its ease of use [1]
- 

#### *Disadvantages of the algorithm*

- The input data only does through 3 rounds of encryption; therefore, it is a lot less secure than other algorithms such as AES. [1]
- Since it only uses 56-bit keys it is a lot less secure than algorithms that use higher key sizes [1].
- For hardware purposes, it could be a little slow as it takes some time for the data to traverse through the 3 rounds [1].

### 4.2.3. Blowfish algorithm

The Blowfish algorithm is a symmetric encryption algorithm that was created in response to the DES's vulnerability. The algorithm takes in a 32-bit plain text input and encrypts the data with an array of 18 keys. To achieve this first a flexible key size of up to 448 bits is processed. In the processing, the key is divided into 18 32-bit subkeys and added to an array of size 18. After this the encryption begins and will for through 16 iterations before it is complete. The plain text is first divided into 2, resulting in 2 32-bit halves. A logical XOR operation is then performed with one of the sub-keys and the left half to produce a new left half. The new left half is then logically XORed with the initial right half to produce a new right half. The new left and right halves are then swapped and a new iteration begins.[7]

#### *Advantages of the algorithm*

- Unpatented algorithm, therefore, can be used by anyone for commercial use [6]
- Faster than the DES algorithm making it suitable for embedded applications
- The algorithm has less operations than AES which means it doesn't need a lot of compute power to operate [6]

#### *Disadvantages of the algorithm*

For blowfish to work, when two endpoints are communicating the key for decryption cannot be accessed through unsecure means. Every endpoint pair communicating will therefore, need a unique key and as the number of endpoints increase, the larger the key management. [6]

### 4.3. Chosen Algorithms

#### 4.3.1. Compression

The initial chosen compression algorithm was the XOR based compression. This compression however when implemented posed to have limitations that were unforeseen. The first limitation is in only being able to compress csv file. When later on the realisation that data might also be in txt format this posed as a limitation to the system. Another limitation was that there is little information on XOR based compression this resulted in very limited resources on the implementation of the code. Thereby this resulted in the debugging of the found code difficult. In the end we could not implement the decompression with the compression algorithm and therefore an alternate algorithm had to be found.

The next best option algorithm to use was the dictionary algorithm as this is widely used and many information on it is found on the internet. With that being said we were able to find a dictionary algorithm that is able to compress all different types of files and decompress them. The code used for compression is the FastLZ compression which implements the LZ77 compression method part of dictionary. Although dictionary uses a more power than XOR compression, it is more efficient in that it works. It also allows for versatility and therefore accommodates the fact that for testing we use an ICM which is different from the Sharc buoy ICM.

#### 4.3.2. Encryption

The initial chosen algorithm was the Blowfish algorithm due to its lightweight characteristics. However, due to it not being the current standard, the AES encryption algorithm was then chosen. It is much more secure with a multitude of resources to aid with implementation.

### 4.4. Subsystem Design

The overall systems can be broken down into an Encryption subsystem and Compression subsystem.

*Table 4: Encryption sub-system requirements*

Requirement ID	Description
ER001	The subsystem interaction should maintain low power
ER002	Data should remain accurate in subsystem interaction
ER003	Encryption and compression subsystems should interact well
ER004	The encrypted data should not have resemblance of the original data
ER005	Subsystem independent of the compression subsystem

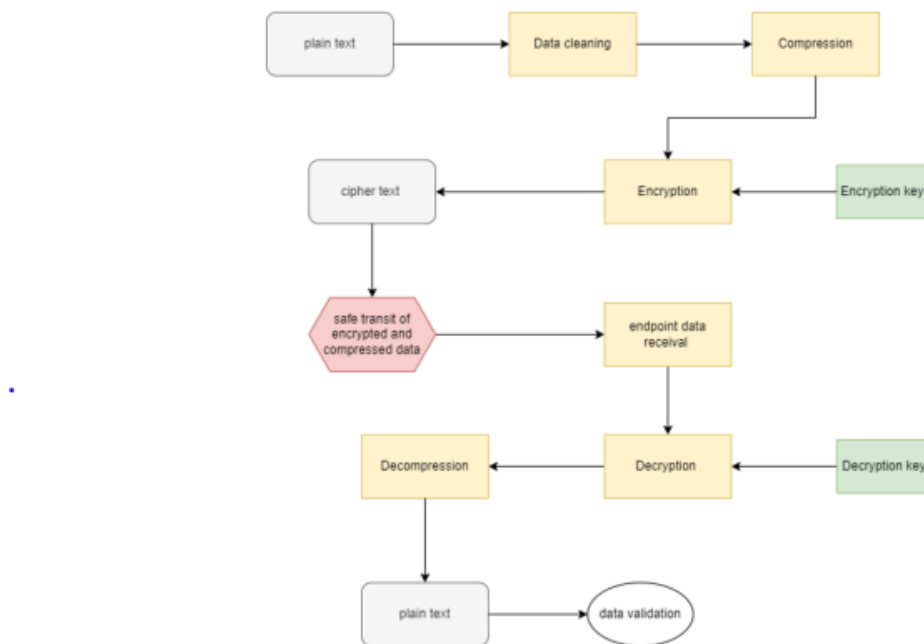
Table 5: Compression sub-system requirements

Requirement ID	Description
CR001	Throughout subsystem interaction of Fourier coefficients should be conserved
CR002	System must run independently to the encryption subsystem
CR003	compression and decompression should be quick to complete
CR004	compression should reduce file size of input data
CR005	decompressed data should mostly match input data

Table 6: Inter sub-system specifications

Specification ID	Description
CS001	90% of decompressed and decrypted data should match the input data
CS002	at least 25% of input data Fourier coefficients should be present in the decompressed/decrypted data
CS003	Compression should at least reduce data by 40%
CS004	Compression/encryption and decompression/decryption should not take more than 10 seconds to complete respectively

#### 4.4.1. Graphical Representation of inter-subsystem design



#### Discussion of Graphical Representation

Data is received as plain text from the IMU. This data is then passed on to the compression block where it is compressed. The compressed data is then passed to the encryption block to be encrypted. Encryption requires an encryption key in order to

encrypt the compressed file. This key is pre-programmed in the encryption algorithm. The output of encrypted data is cipher text. This is then passed to decryption to be decrypted. Decryption requires a decryption key to decrypt. This key is related to the encryption key. This is given to the user receiving the encrypted data. The decrypted data is then passed on back to the compression block where it will be decompressed. Decompression does not require a key however decompression should occur in the same compression block for it to work. The output of decompressed data is the plain text again. This is the same original data that was inputted in the first place.

## 5. Validation using Simulated or Old Data

### 5.1 Importance for Simulation based validation

Simulation validation is running the system with old data to test the various systems implemented in the system before building the system as a whole and running it in real-time. Simulation based validation is important in that it allows for debugging of the system before running it in real-time which in that case debugging can be difficult and not easily traceable. Simulation based validation also gives an idea of how the system will run, discarding all the anomalies that might happen during run time. It is in that, that simulation validation provides a good average estimation of power usage and speed of the system.

In this project, it is particularly important to use old data as that allows for the testing of the compression and encryption algorithms regardless of what the IMU system is outputting. , therefore, allowing for testing to happen before the configuration of the IMU and even during processing and debugging the IMU. This, therefore, allows for the development compression and encryption systems independently. Old data in this project also helps in giving a good average estimation of the running speeds for both the compression and encryption systems. Old data also allows for testing using different data and therefore understanding how compression and encryption systems behave with different data. This in conclusion allows for the checking of the compression and encryption system to see if they run as expected given the different data.

### 5.2 Steps for validation using old data

**Step 1:**

The first step is identifying the various parts of the compression and encryption system to be testes

**Step 2:**

Identify the different data that will be used for each part to be tested

**Step 3:**

Collect or formulate the different data to be used

**Step 4:**

With each data collected, test each part pf the system with its designated test data

## 5.3 Data

Overall data used is sample data given, which is of simple walking around motion in a lecturer room. This data is data from the IMU ICM20948 and it is a time-based series. The data displays x, y, z coordinates for each measurement (acceleration, quat, gyro, yaw, roll, pitch, gravity) and therefore capturing full motion movement experienced. T

From this sample of data, the data is divided four files that will be used to test various parts of the compression and encryption system.

### **Data 1(size-13997KB):**

This will be the csv file as it is given.

### **Data 2 (size-13997 KB):**

This will be the csv file converted into a txt file.

### **Data 3 (size- 13538 KB):**

This will be the csv file reduced in size by deleting some of the contents in the file.

### **Data 4 (size- 9394 KB):**

A reduced csv of the reduced csv file.

## 5.4 Discussion of data

Data 1 is used to test if the system can compress and encrypt normal or expected data from the IMU ICM20948. This is mainly to test the functionality of the system as a whole and if it can work with ICM data as expected.

Data 2 is then used to seem if the system can handle same data of a txt file format. Looking at the unavailability of the ICM20649, this data is used as way to see if the system can still operate as normal ven with the IMU writing data into a txt file instead of a csv file. And this is to accommodate the fact that the ICM20649 might work differently or be configured differently however still taking into consideration that the system is designed for data from that ICM.

Data 3 lastly is used to test system speed differences when data is of a smaller size. This is also used for test the different compression ratios with different data size.

## 5.5 Validity of data

The data used is good enough to run the ATPs (Acceptance Test Procedure) in that as aforementioned the data allows for testing functionality of the system on expected data. The data also allows for testing the functionality of the system given a different data file format which might be the case considering the limitation in the ICM used for the Bouy. Lastly, the data used is valid enough to run the ATPs as it allows for testing of the system speeds and seeing the behaviour of the system with different data file sizes

## 5.6 Graphical Representation the Overall data

### Plots of Accelerometer data

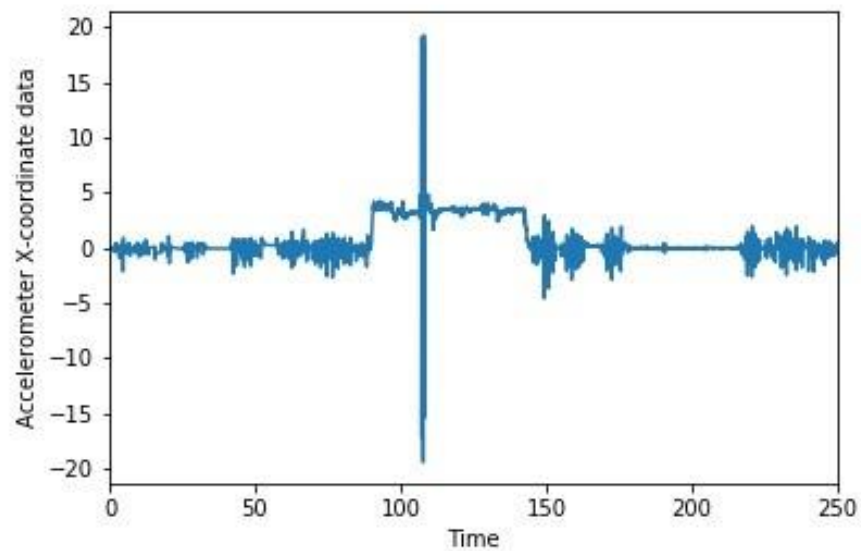


Figure 1: Accelerometer X-Coordinate data vs Time

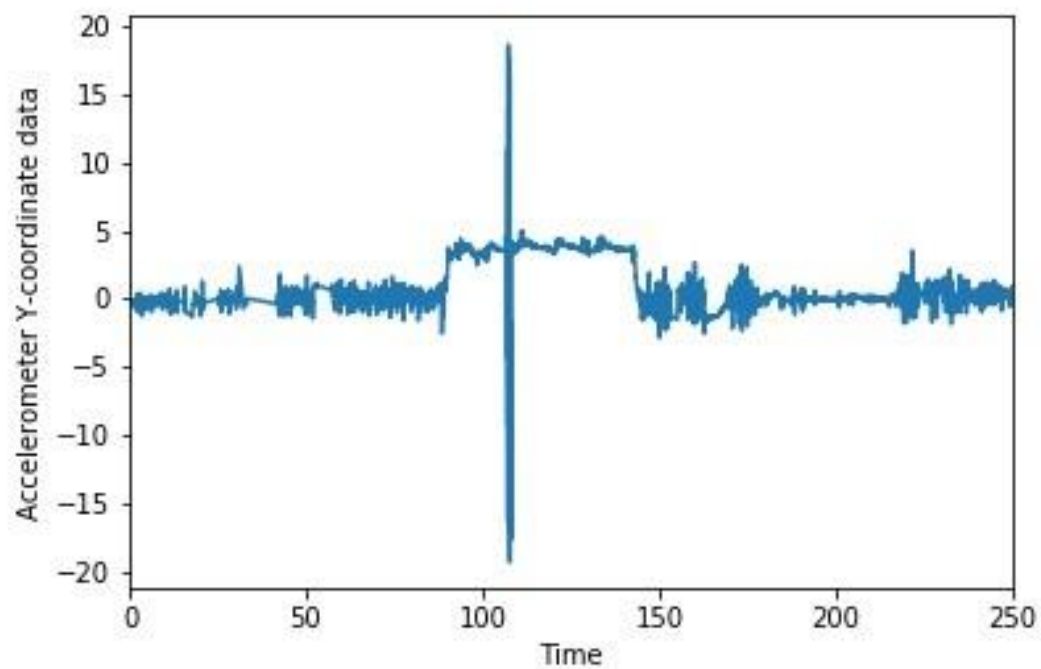


Figure 2: Accelerometer Y-Coordinate data vs Time

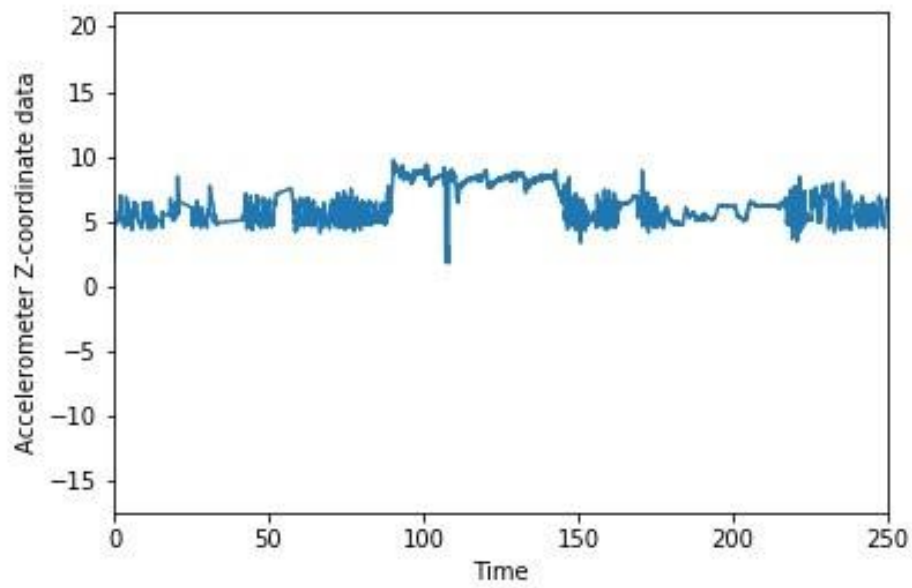


Figure 3: Accelerometer Z-Coordinate data vs Time

#### Fourier Transforms of Accelerometer Data

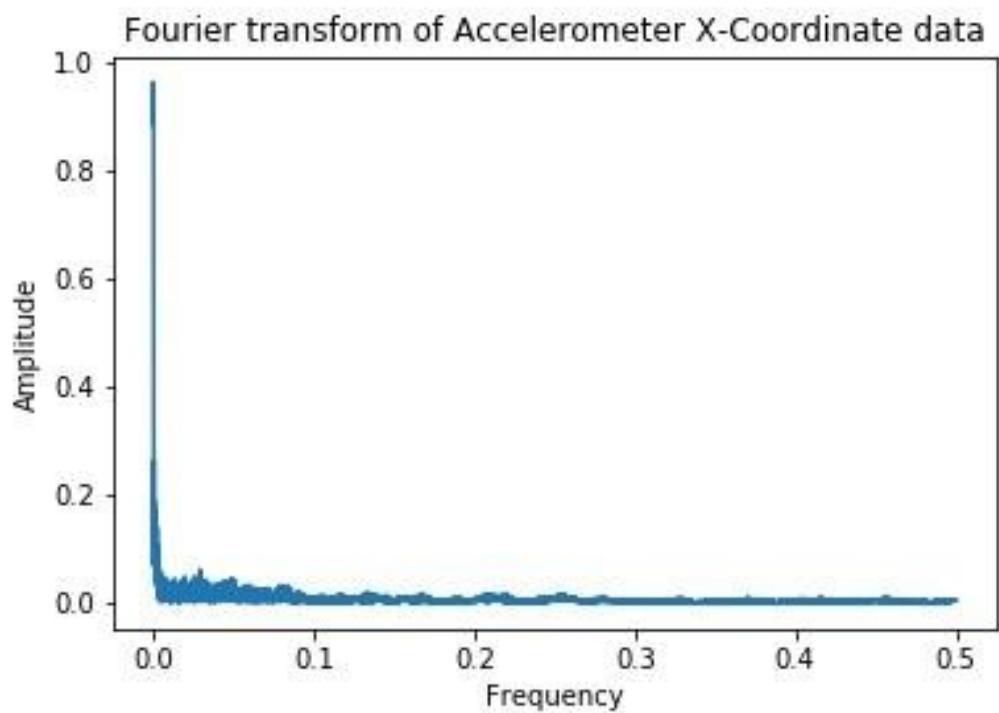


Figure 4: Fourier Transform of Accelerometer X-Coordinate data



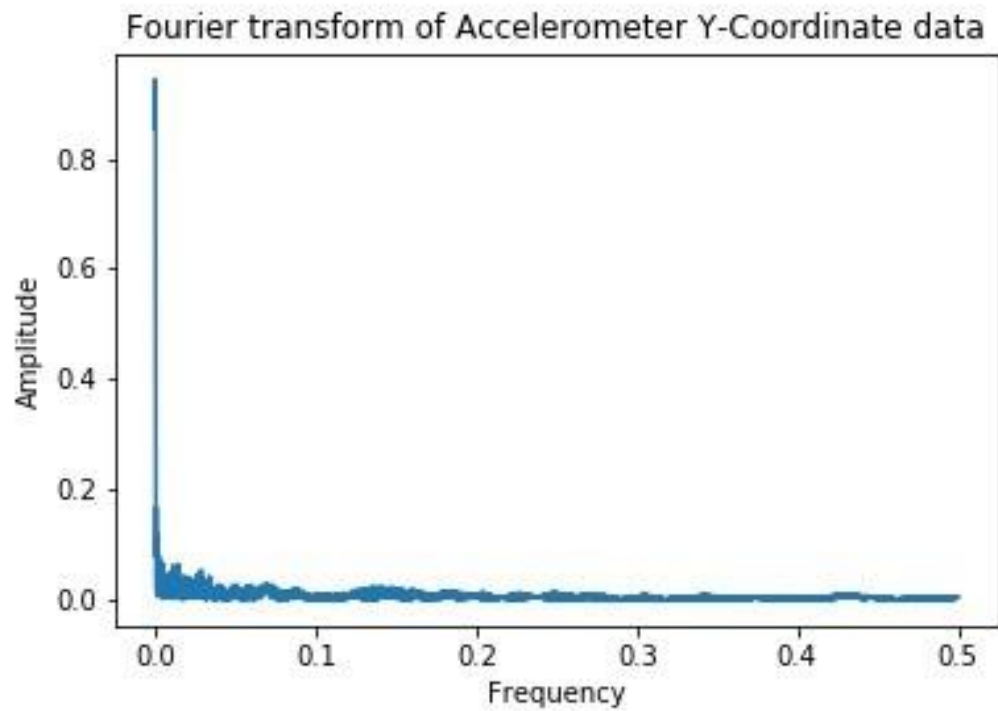


Figure 5: Fourier Transform of Accelerometer Y-Coordinate data

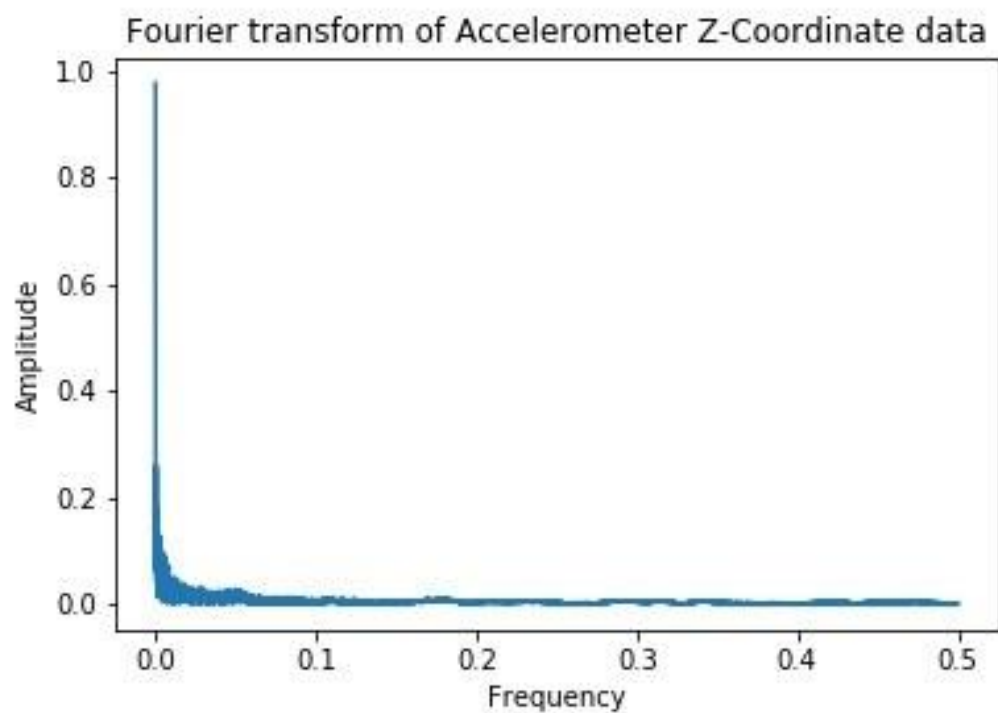


Figure 6: Fourier Transform of Accelerometer Z-Coordinate data

### Plots of Gyroscope data

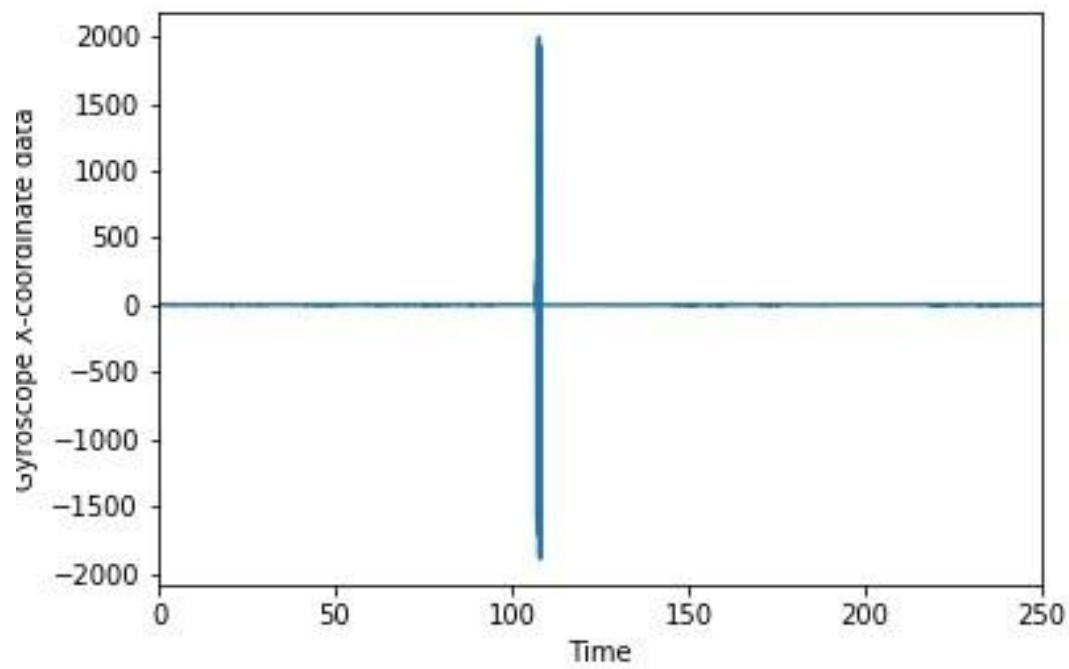


Figure 7: Gyroscope X-Coordinate data vs Time

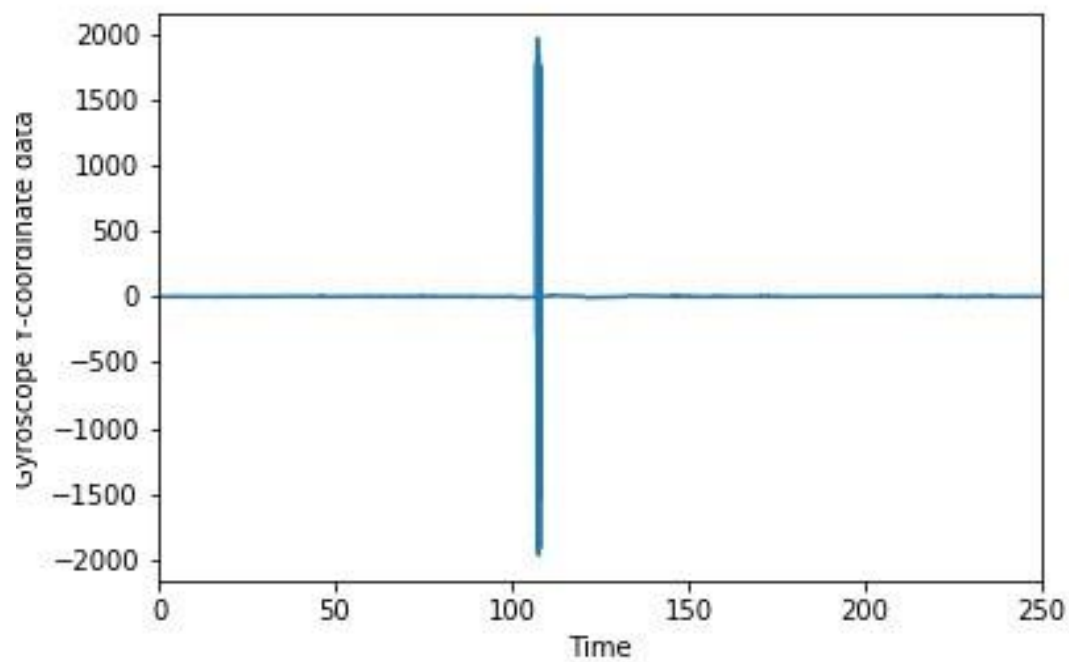


Figure 8: Gyroscope Y-Coordinate data vs Time

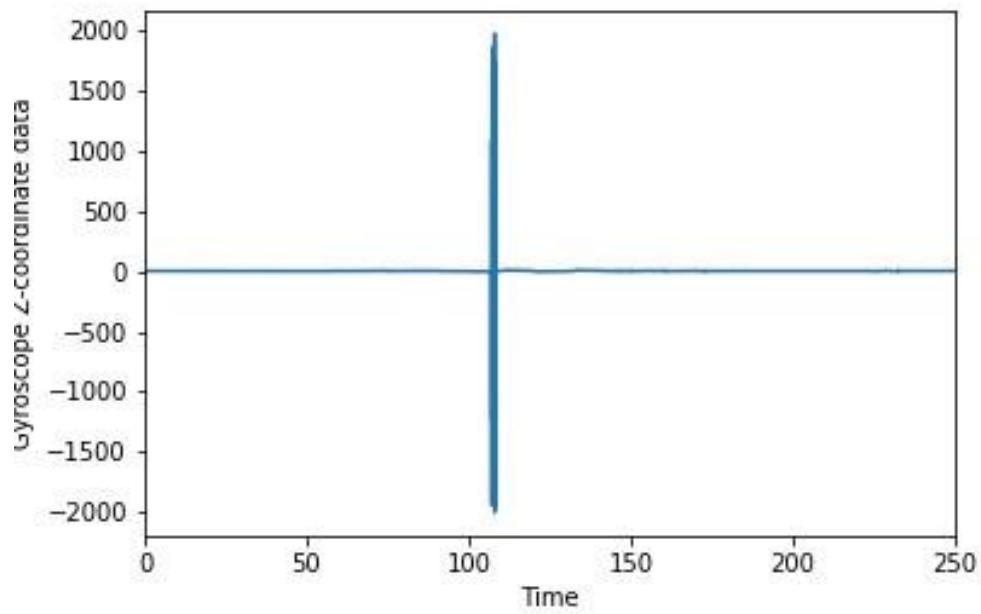


Figure 9: Gyroscope Z-Coordinate data vs Time

#### **Fourier Transforms of Gyroscope Data**

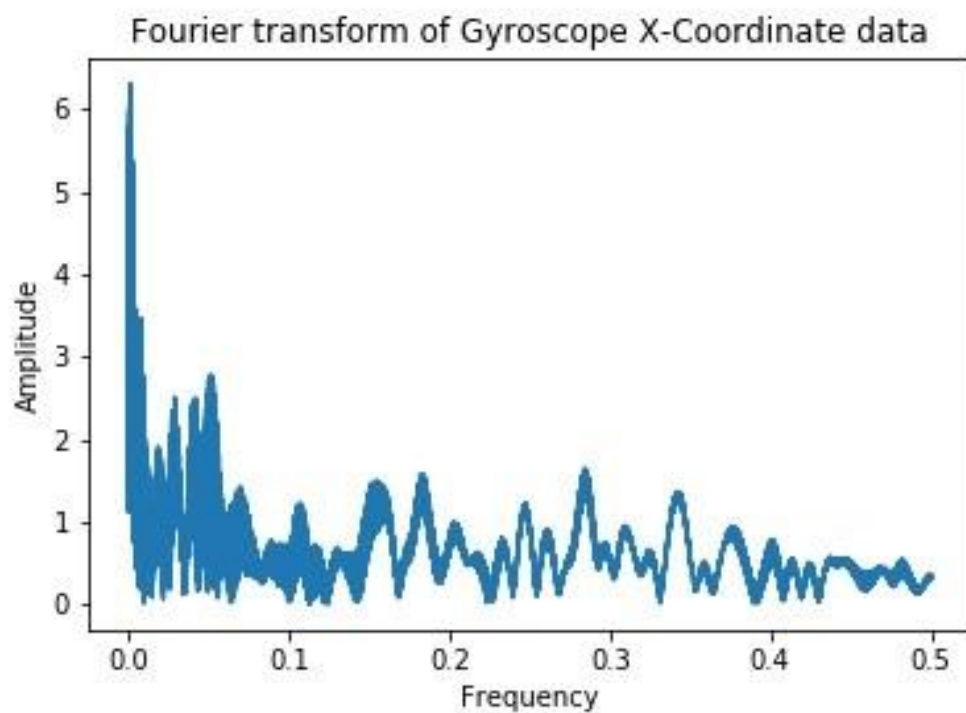


Figure 10: Fourier Transform of Gyroscope X-Coordinate data

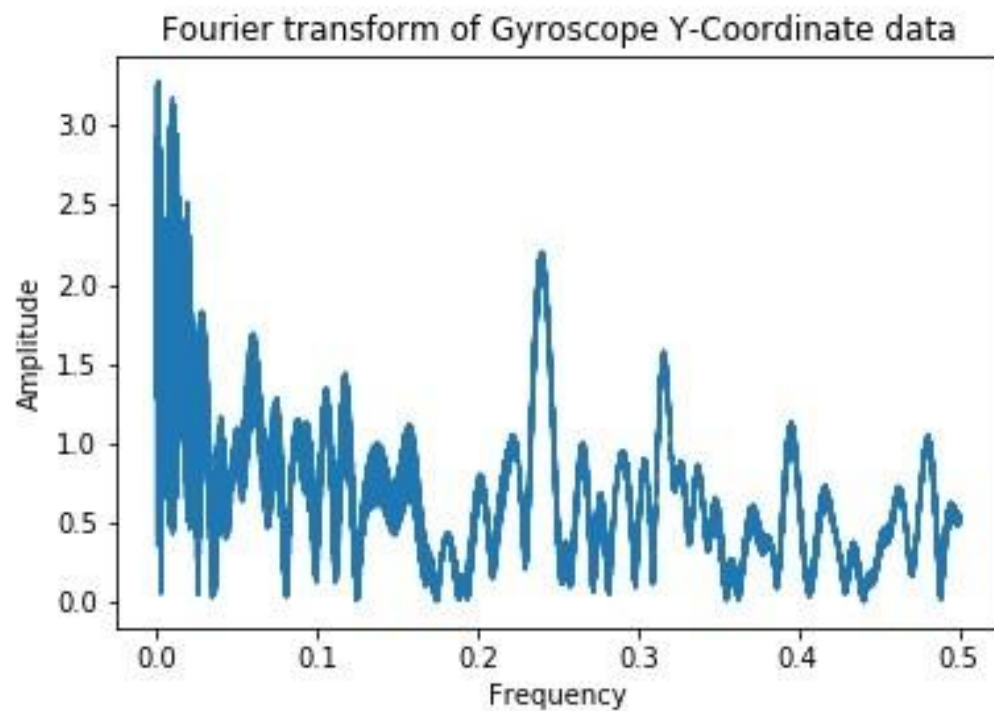


Figure 11: Fourier Transform of Gyroscope Y-Coordinate data

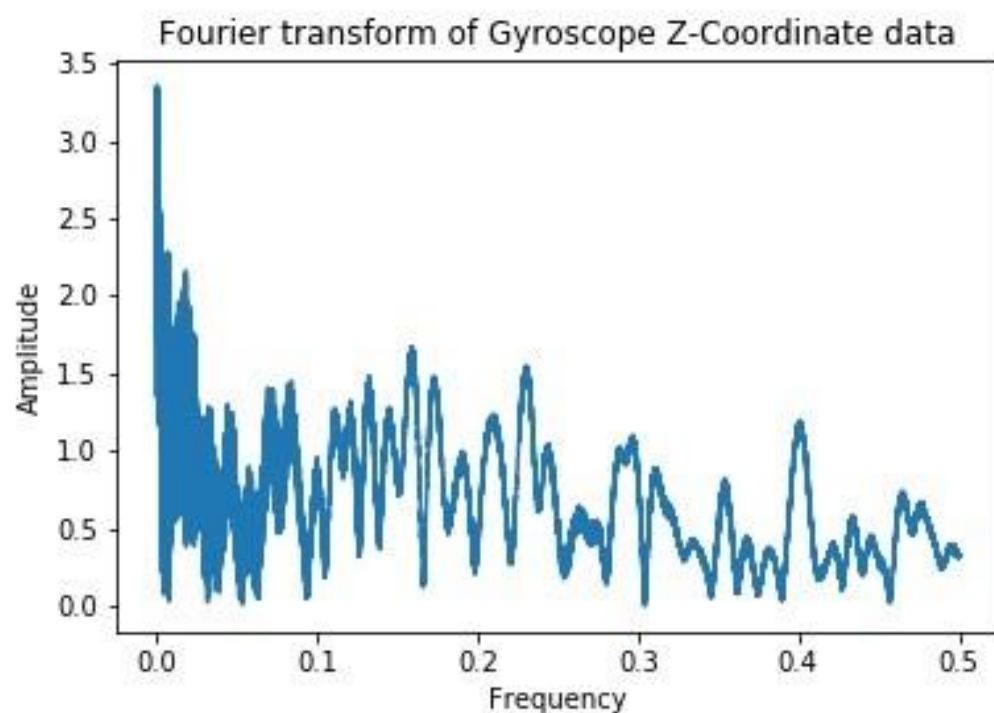


Figure 12: Fourier Transform of Gyroscope Z-Coordinate data

## 5.7 Analysis of the overall data

### **Accelerometer data**

Figures 1, 2 and 3 depict the data obtained from the accelerometer. In the X and Y planes the accelerometer appears to remain stationary at 0. This is depicted by Figures 1 and 2 respectively. However, the data jumps to 5 between 100 and 150 seconds for both the X and Y plane data. The Z plane data differs slightly from the aforementioned 2 planes. The data remains stationary at 5 and jumps to 10 between 100 and 150 seconds. This is depicted by figure 3. There also appears to be noise in the data of all 3 planes. This is depicted by the high frequency fluctuations of the data in all 3 graphs (see figure 1, 2 and 3). There is also a spike in the data of all 3 planes at the 110 second mark. For the X and Y plane data the spike is the long vertical line from -18 to 18 (See figures 1 and 2 respectively). The spike in the Z plane differs from the spike in the X and Y plane as it has a much smaller vertical height.

### **Fourier transform of the accelerometer data**

The Fourier transforms of the 3 accelerometer coordinate planes are similar. They all include a single frequency component at  $\omega = 0$ . After this there are no other observable frequency components. This is depicted by figures 4, 5 and 6.

### **Gyroscope data**

The Gyroscope data can be seen in figures 7, 8 and 9. These figures represent the data for the gyroscopes X, Y and Z planes respectively. The 3 figures are similar as they are stationary throughout the logging time. However, they all incorporate a spike from around -1800 to 1800 at around 110 seconds. This is similar to the spike in the accelerometer data.

### **Fourier transform of the gyroscope data**

The Fourier transforms of the X, Y and Z plane data of the gyroscope is very noisy. This is depicted by figures 10, 11 and 12 respectively. Unlike the accelerometer data which has a single frequency component at  $\omega = 0$ , there are many frequency components for the data of all 3 planes.

## 5.8 Steps followed for collecting and formulating the data

1. Downloaded the csv file given
2. Save a copy as txt file
3. Save a different copy with deleted contents
4. Save a copy of the deleted contents copy with more deleted contents from that copy

## 5.9 Experimental setup

### 5.9.1 Overall system

The functionality of the overall system is to compress and encrypt IMU data in real time. The input data will be the IMU data which can be in the form of a .csv or .txt file. The output data should be a compressed and encrypted version of this file. This data can then be fed into the decryption/compression subsystem. The output of this subsystem is a file identical to the original data from the IMU. If a .csv file were used as an input then a .csv file identical to the original should be outputted. However, if the two are not similar then the overall system still needs modification.

To test the functionality of the overall system the following setup is carried:

Data is separated into the different copies required

#### Overall test 1:

Test if compression and encryption system can successfully compress, encrypt, decrypt, and decompress data 1.

#### Overall test 2:

Test if compression and encryption system can successfully compress, encrypt, decrypt, and decompress data 2.

#### Overall test 3:

Compare Overall test 1 results with the graphical representation of data as aforementioned.

The graphs of the output are expected to be the same as the graphs of the input data above. Conservation of 25% of the Fourier coefficients is required.

#### Overall test 4:

Compare Overall test 2 results with the graphical representation of data as aforementioned.

The graphs of the output are expected to be the same as the graphs of the input data above.

Conservation of 25% of the Fourier coefficients is required.

### 5.9.2 Compression block

#### *How it works*

Code for compression is written in c. The code runs on windows using the terminal. For compression, a csv or txt data format that only consists of numbers with the first block representing time is used as an input file. On Windows terminal make all is run to compile all the compression and decompression code blocks. To run compression 6pack –compression level input file output file is run on the terminal. This converts the csv or txt file to a the output file of any format (for testing a txt file was used) and therefore representing the compressed file. The compression code allows for benchmarking which can be indicated by inputting –mem where followed by compression level and input file. Decompression occurs by running 6unpack Compressed file.

### *Functionality*

To test the functionality of compression block, data is collected and formulated into the different data used as Input data.

Compression code has different level in which it can compress. For testing functionality of the block both levels must be tested.

### *Compression level 1*

#### **Compression test 1:**

Test to see if data 1 compresses.

#### **Compression test 2:**

Test to see if compressed file of data 1 decompresses to data 1.

#### **Compression test 3:**

Repeat compression test 1 with data 2.

#### **Compression test 4:**

Repeat compression test 2 with compressed file of data 2.

#### **Compression test 5:**

Benchmark compression of data 1. Achieve at least 40% space saving and processing of less than 10 seconds.

#### **Compression test 6:**

Benchmark compression of data 2. Achieve at least 40% space saving and processing of less than 10 seconds.

#### **Compression test 7:**

Benchmark compression of data 3. Achieve at least 40% space saving and processing of less than 10 seconds.

#### **Compression test 8:**

Benchmark compression of data 4. Achieve at least 40% space saving and processing of less than 10 seconds.

### *Compression level 2*

The same tests as compression level 1 apply.

## 5.9.3 Encryption Block

### *Experimental setup*

The functionality of the encryption block can be checked by manual means or through the use of automation through python. To ensure the encryption block functions as expected, the system will be checked against the following requirements:

The encrypted data should not be the same as the original data

After data has passed through the encryption block, the data should not resemble the original data. If the input file is in text/string format, the output of the encryption block

should not resemble that text. This applies to other input file types such as files containing numbers and even images.

#### The encrypted data should not be readable to the human eye

This applies particularly to text data that can be read. After the data has passed through the encryption block, it should be hard to read. This ensures that if the data were unexpectedly intercepted it should not be deciphered easily/quickly.

#### The decrypted data should be the same as the original data that went into encryption block

When the encrypted data is decrypted, the data should be exactly the same as the original compressed data that entered the encryption block. The data should be the same to ensure that no meaningful/important data is lost in the encryption process

## 5.10 Results

### Overall system

#### **Overall test 1:**

Data 1 successfully compresses, encrypts, decrypts, and decompresses.

#### **Overall test 2:**

Data 2 successfully compresses, encrypts, decrypts, and decompresses.

#### **Overall test 3:**

Below is graphs of the decompressed and decrypted data. The graphs are simi

#### **Plots of Accelerometer data (Decrypted and Decompressed)**

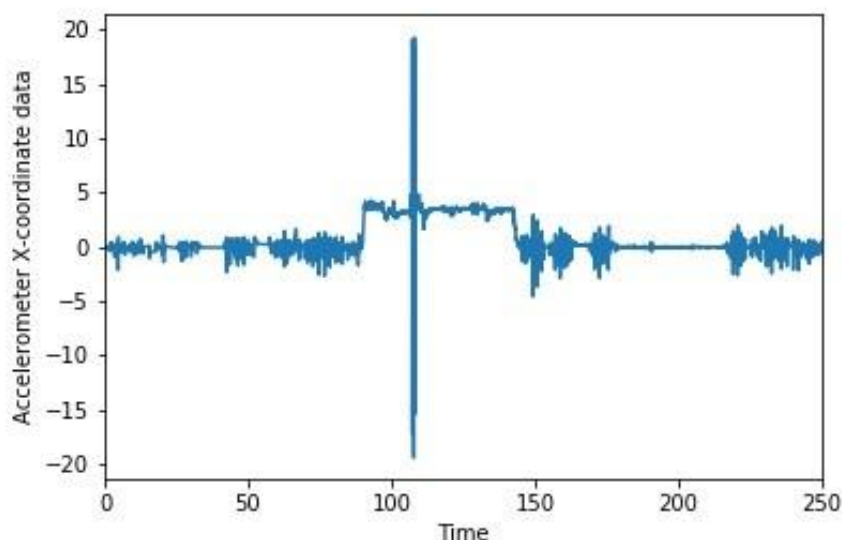
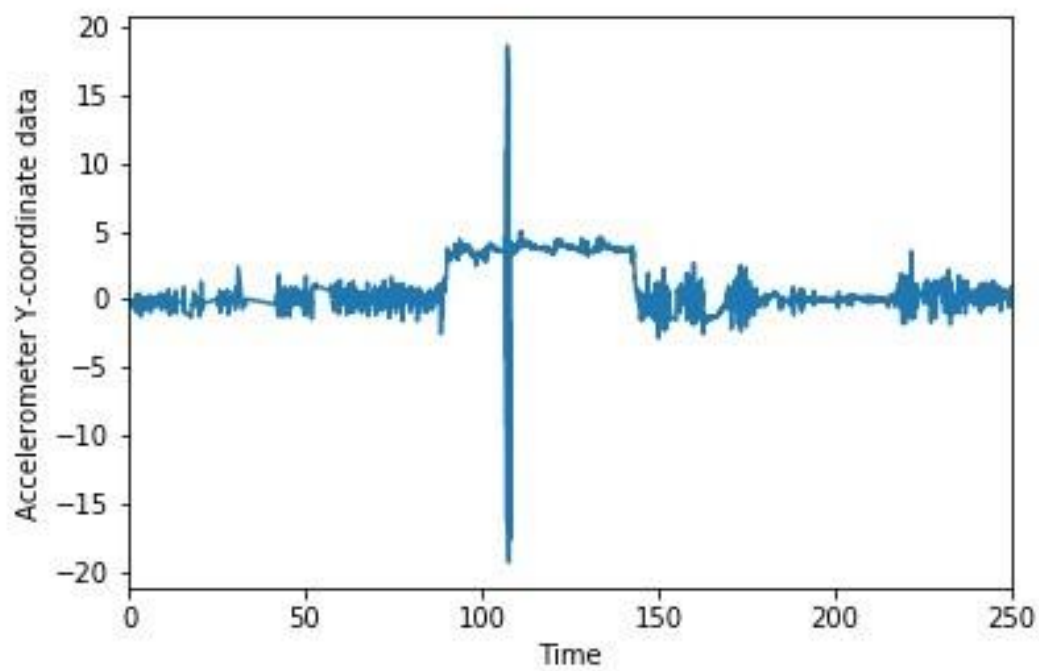
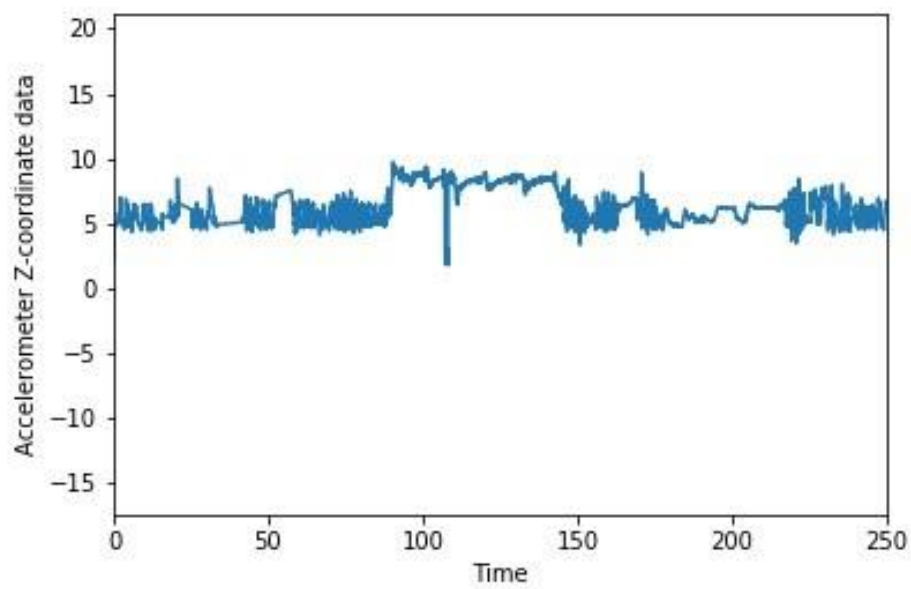


Figure 13: Accelerometer X-Coordinate data vs Time



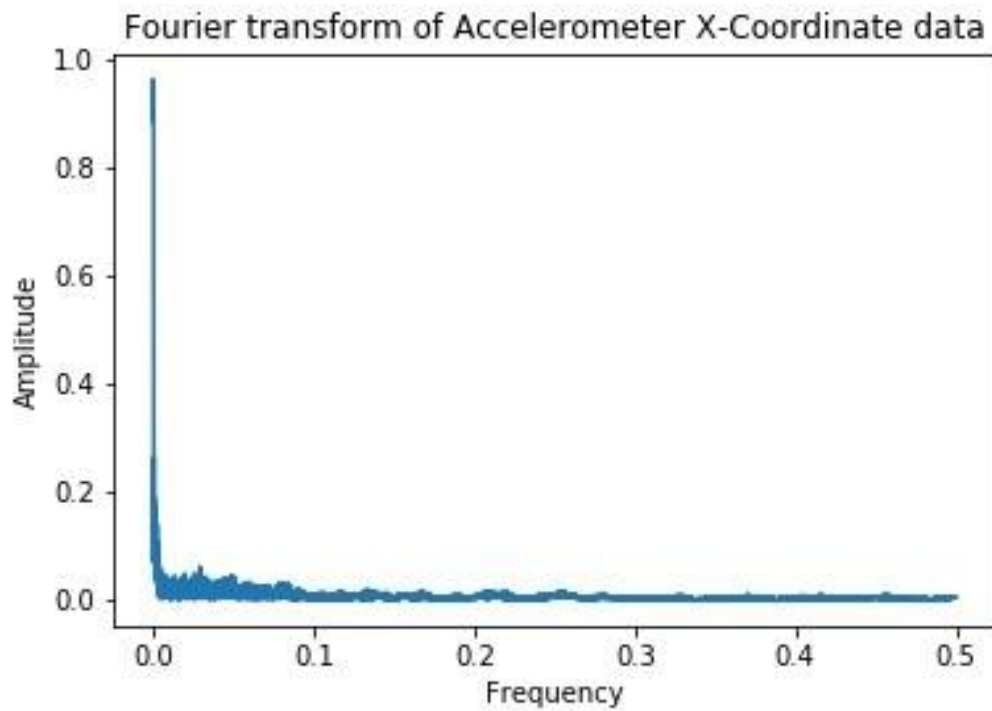


*Figure 14: Accelerometer Y-Coordinate data vs Time  
(Decrypted/Decompressed)*

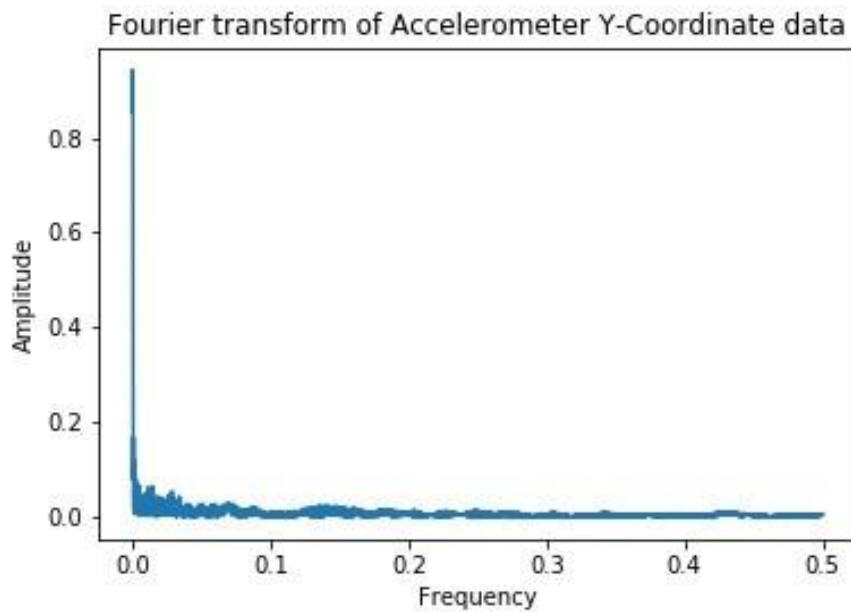


*Figure 15: Accelerometer Z-Coordinate data vs Time  
(Decrypted/Decompressed)*

### Fourier Transforms of Accelerometer Data (Decrypted and Decompressed)



*Figure 16: Fourier Transform of Accelerometer X-Coordinate data (Decrypted/Decompressed)*



*Figure 17: Fourier Transform of Accelerometer Y-Coordinate data (Decrypted/Decompressed)*

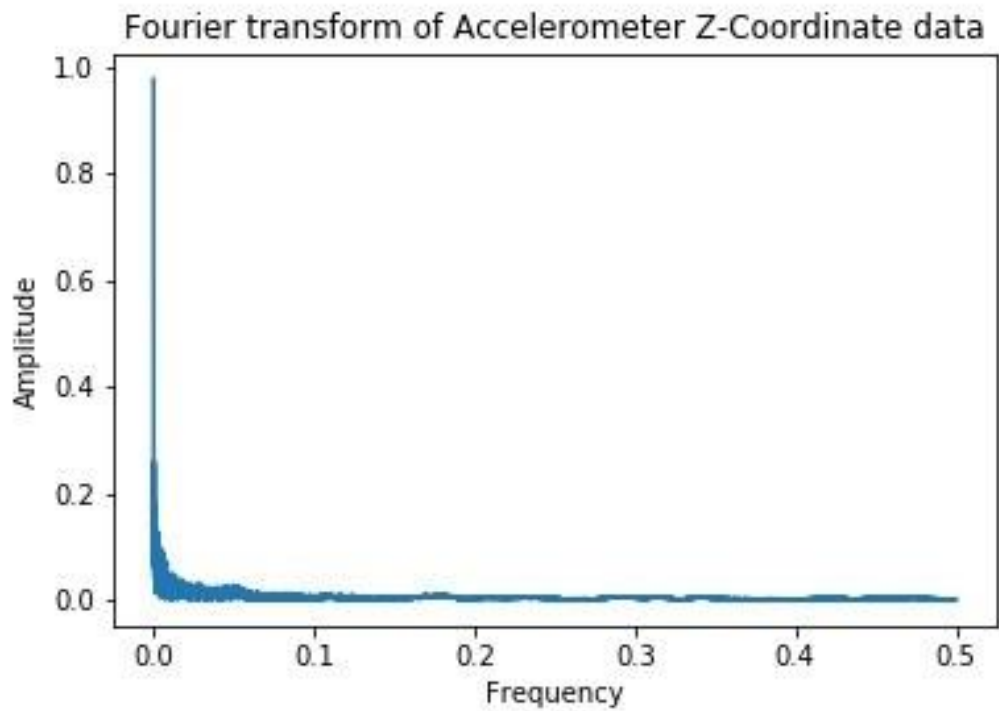


Figure 18: Fourier Transform of Accelerometer Z-Coordinate data (Decrypted/Decompressed)

**Plots of Gyroscope data (Decrypted and Decompressed)**

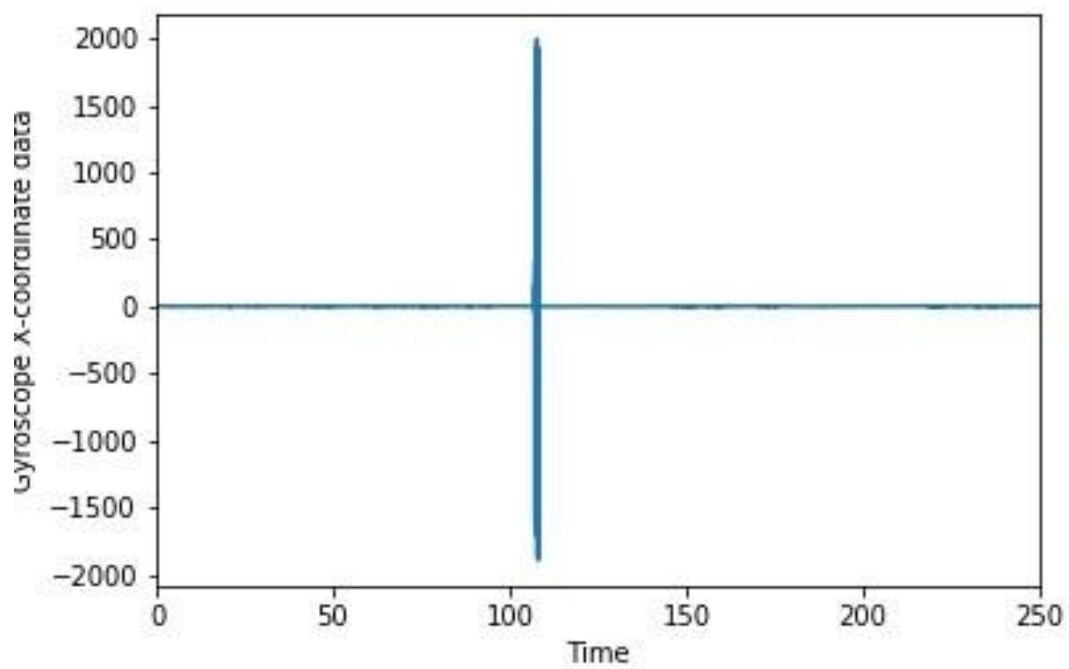
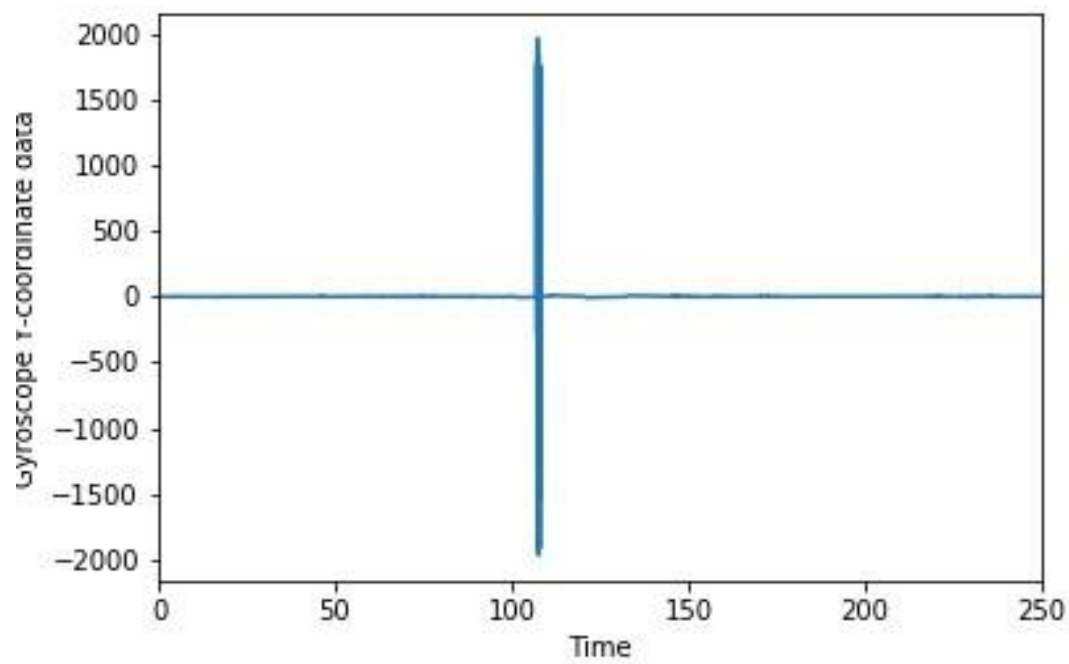
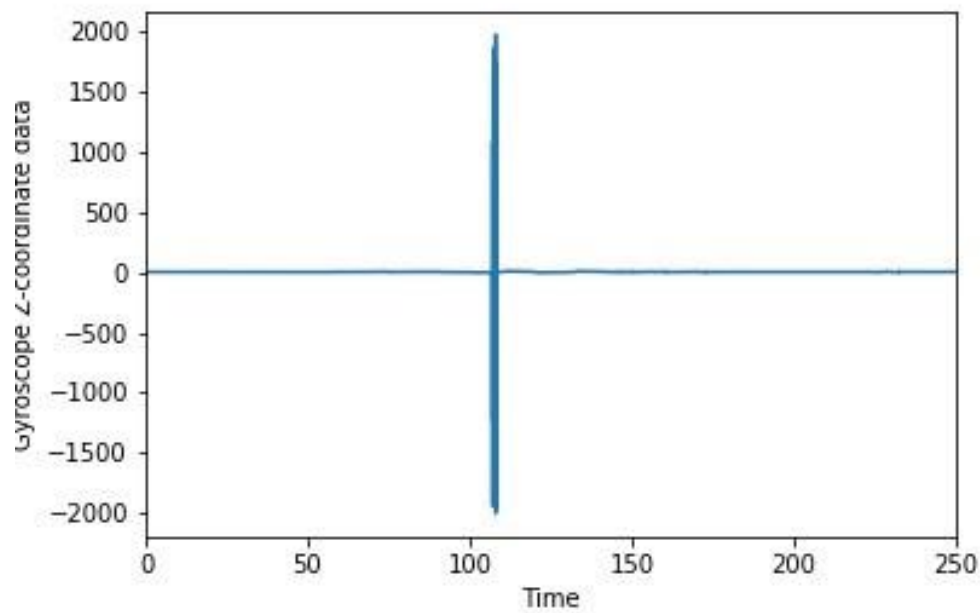


Figure 19: Gyroscope X-Coordinate data vs Time (Decrypted/Decompressed)

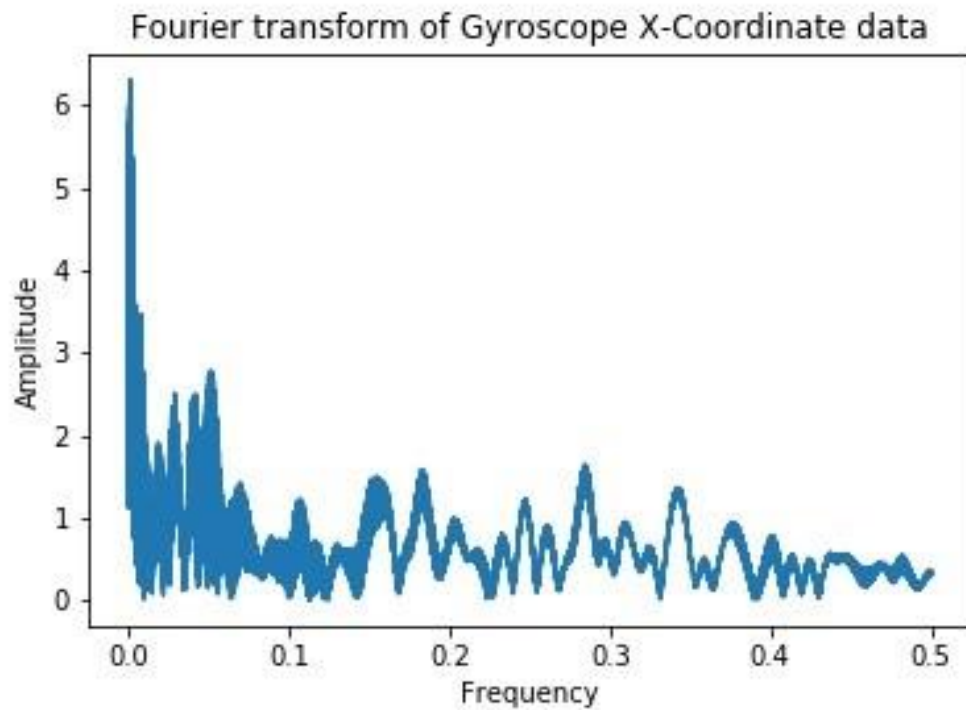


*Figure 20: Gyroscope Y-Coordinate data vs Time  
(Decrypted/Decompressed)*

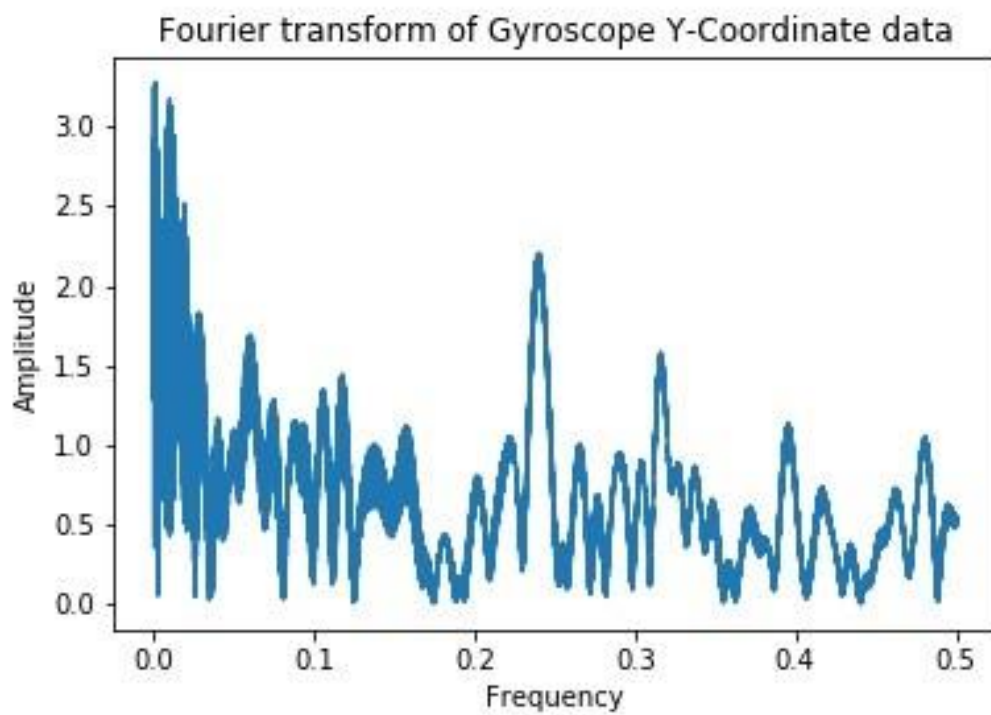


*Figure 21: Gyroscope Z-Coordinate data vs Time  
(Decrypted/Decompressed)*

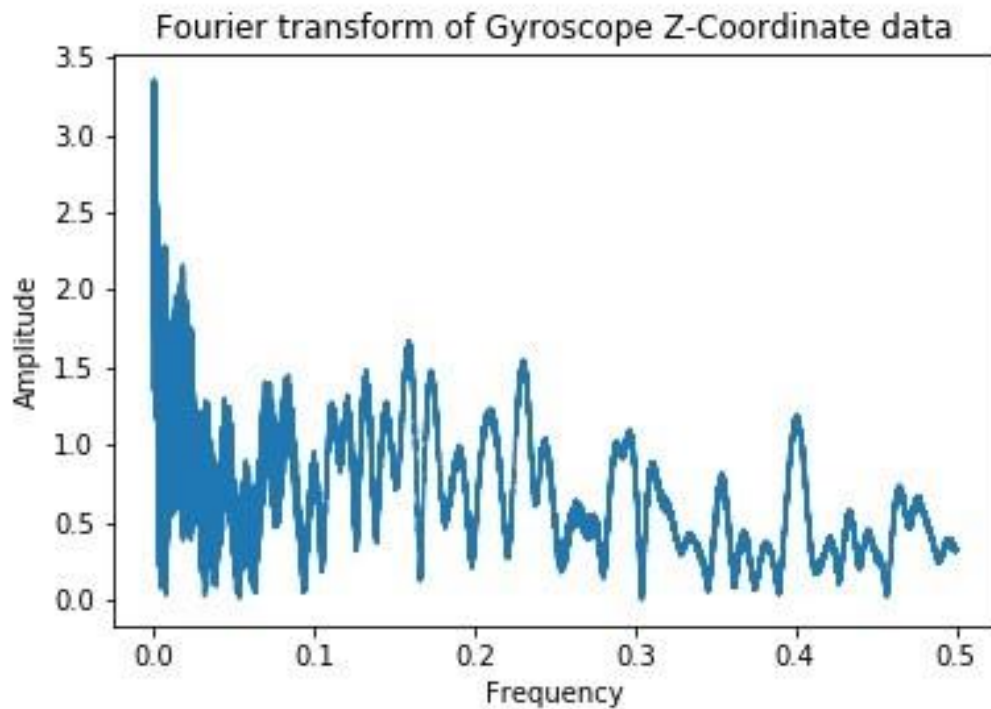
### Fourier Transforms of Gyroscope Data (Decrypted and Decompressed)



*Figure 22: Fourier Transform of Gyroscope X-Coordinate data (Decrypted/Decompressed)*



*Figure 23: Fourier Transform of Gyroscope Y-Coordinate data (Decrypted/Decompressed)*



*Figure 24: Fourier Transform of Gyroscope Z-Coordinate data (Decrypted/Decompressed)*

When comparing these graphs to the ones above under graphical representation data, the graphs are the same.

#### **Overall test 4:**

The decompressed data for data 2 was similar to the decompressed data for the overall test 3. The graphical representations were, therefore, the same i.e. The same time based graphs and frequency based graphs.

### *Analysis of results*

The overall system works as expected as all the tests passed. Conservation of 25% of the Fourier coefficients occurs as 100% of the input data is retrieved on decompression.

Compression block

	Compression level 1	Compression level 2
--	---------------------	---------------------

Compression test 1	Data 1 compresses successfully	Data 1 compresses successfully
Compression test 2	Compressed file of data 1 decompresses successfully	Compressed file of data 1 decompresses successfully
Compression test 3	Data 2 compresses successfully	Data 2 compresses successfully
Compression test 4	Compressed file of data 2 decompresses successfully	Compressed file of data 2 decompresses successfully
Compression test 5	Data 1 compression speed: 202.3 Mbyte/s. Data 1 decompression speed: 511.2 Mbyte/s. Data 1 compression space saving: 66.1 % saved	Data 1 compression speed: 201.3 Mbyte/s Data 1 decompression speed: 494.2 Mbyte/s Data 1 compression space saving: 66.6% saved
Compression test 6	Data 2 compression speed: 202.3 Mbyte/s. Data 2 decompression speed: 487.3 Mbyte/s Data 2 compression space saved: 66.1% saved	Data 2 compression speed: 198.6 Mbyte/s. Data 2 decompression speed: 503.7 Mbyte/s. Data 2 compression space saving: 66.5% saved
Compression test 7	Data 3 compression speed: 201.2 Mbyte/s. Data 3 decompression speed: 499.0 Mbyte/s. Data 3 compression space saving: 66.2% saved	Data 3 compression speed: 200.2 Mbyte/s Data 3 decompression speed: 496.6 Mbyte/s. Data 3 compression space saving: 66.6% saved
Compression 8	Data 4 compression speed: 197.8 Mbyte/s. Data 4 decompression speed: 487.4 Mbyte/s. Data 4 compression space saving: 65.9% saved	Data 4 compression speed: 197.7 Mbyte/s Data 4 decompression speed: 493.8 Mbyte/s. Data 4 compression space saving: 66.2% saved

#### *Analysis of results*

Looking at compression tests 1,2,3 and 4 the compression block functions well. The compression block can successfully compress and decompress data. This goes for both compression level 1 and level 2. When looking at compression level 1; a txt file and csv file are treated in the same way. This is seen in the similarity in the saved space and the compression speed. In compression level 2 however these are processed differently with the difference in compression speed and space-saving. Comparing compression level 1 and level 2, compression level 2 in general is faster than compression level 1 and therefore a better compressor to use. For both compression level 1 and 2, as data size reduces the compression speed is faster however saved space reduces. Decompression speed differs a lot however the patterns are that only the reduction of input file size affects the speed of decompression. Overall compression functions well, as compression and decompression occur successfully for both csv and txt files. Compression speeds reduce with file reduction as expected. Compression level 2 is better than compression level 2 which is also expected. Appendix B show screenshots of the compression tests performed on a windows computer using the command prompt.



## Encryption Block

An image of a csv file containing sample IMU

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	Time	Count	Quat W	Quat X	Quat Y	Quat Z	Accel X (g)	Accel Y (g)	Accel Z (g)	Gyro X (g/s)	Gyro Y (g/s)	Gyro Z (g/s)	Pitch	Roll	Yaw	Q	Z	Grav Y	Grav Z	Temp			
2	0.019	3	0.9999	0.0037	1.00E-04	-0.0004	0.006	-0.0683	2.2417	0	0	-0.122	0.0007	-0.0002	0.0074	0.0007	-0.0007	-0.0002	0.0074	34.9947			
3	0.023	4	0.9999	0.0037	1.00E-04	-0.0004	0.006	-0.0684	2.2417	0	0	-0.122	0.0007	-0.0002	0.0074	0.0007	-0.0007	-0.0002	0.0074	34.9947			
4	0.045	5	0.9999	0.0036	0.0002	-0.0006	0.003	-0.1	3.2135	0	0	-0.122	0.0012	-0.0004	0.0072	0.0012	-0.0004	-0.0072	-0.0004	0.0072	9.9999	34.1065	
5	0.053	6	0.9999	0.0035	0.0002	-0.0008	-0.0006	-0.1162	3.8278	0	0	-0.122	0.0017	-0.0004	0.007	0.0017	-0.0004	-0.007	-0.0004	0.007	9.9999	34.0947	
6	0.057	7	0.9999	0.0035	0.0002	-0.0008	-0.0006	-0.1162	3.8278	0	0	-0.122	0.0017	-0.0004	0.007	0.0017	-0.0004	-0.007	-0.0004	0.007	9.9999	34.0947	
7	0.075	8	0.9999	0.0034	0.0002	-0.0011	-0.006	-0.1299	4.2706	0	0	-0.122	0.0022	-0.0005	0.0067	0.0022	-0.0005	-0.0067	-0.0005	0.0067	9.9999	34.1124	
8	0.083	9	0.9999	0.0034	0.0002	-0.0013	-0.0078	-0.1341	4.4631	0	-0.061	-0.122	0.0027	-0.0005	0.0067	0.0027	-0.0005	-0.0067	-0.0005	0.0067	9.9999	34.1241	
9	0.087	10	0.9999	0.0034	0.0002	-0.0013	-0.0078	-0.1341	4.4631	0	-0.061	-0.122	0.0027	-0.0005	0.0067	0.0027	-0.0005	-0.0067	-0.0005	0.0067	9.9999	34.1241	
10	0.105	11	0.9999	0.0033	0.0002	-0.0016	-0.003	-0.1329	4.6128	0	-0.061	-0.122	0.0033	-0.0005	0.0066	0.0033	-0.0005	-0.0066	-0.0005	0.0066	9.9999	34.0947	
11	0.113	12	0.9999	0.0032	0.0002	-0.0019	0	-0.1305	4.7062	0	0	-0.122	0.0038	-0.0005	0.0065	0.0038	-0.0005	-0.0065	-0.0005	0.0065	9.9999	34.1065	
12	0.117	13	0.9999	0.0032	0.0002	-0.0019	0	-0.1305	4.7062	0	0	-0.122	0.0038	-0.0005	0.0065	0.0038	-0.0005	-0.0065	-0.0005	0.0065	9.9999	34.1065	
13	0.135	14	0.9999	0.0032	0.0003	-0.0022	-0.0012	-0.1311	4.7643	0	0	-0.122	0.0044	-0.0006	0.0065	0.0044	-0.0006	-0.0065	-0.0006	0.0065	9.9999	34.1124	
14	0.143	15	0.9999	0.0032	0.0004	-0.0024	-0.0066	-0.1263	4.802	0	0	-0.122	0.0049	-0.0006	0.0064	0.0049	-0.0006	-0.0064	-0.0006	0.0064	9.9999	34.1182	
15	0.147	16	0.9999	0.0032	0.0004	-0.0024	-0.0066	-0.1263	4.802	0	0	-0.122	0.0049	-0.0006	0.0064	0.0049	-0.0006	-0.0064	-0.0006	0.0064	9.9999	34.1182	
16	0.165	17	0.9999	0.0031	0.0004	-0.0027	-0.0066	-0.1257	4.8722	0	0	-0.122	0.0054	-0.0008	0.0062	0.0054	-0.0007	-0.0062	-0.0008	0.0062	9.9999	34.1006	
17	0.173	18	0.9999	0.003	0.0004	-0.003	-0.009	-0.1251	4.8463	0	0	-0.122	0.006	-0.0008	0.006	0.006	-0.0007	-0.006	-0.0008	0.006	9.9999	34.0947	
18	0.177	19	0.9999	0.003	0.0004	-0.003	-0.009	-0.1251	4.8463	0	0	-0.122	0.006	-0.0008	0.006	0.006	-0.0007	-0.006	-0.0008	0.006	9.9999	34.0947	
19	0.19	20	0.9999	0.0029	0.0004	-0.0032	-0.0066	-0.1257	4.8565	0	0	-0.122	0.0065	-0.0008	0.0057	0.0065	-0.0007	-0.0057	-0.0008	0.0057	9.9999	34.1124	
20	0.194	21	0.9999	0.0029	0.0004	-0.0032	-0.0066	-0.1257	4.8565	0	0	-0.122	0.0065	-0.0008	0.0057	0.0065	-0.0007	-0.0057	-0.0008	0.0057	9.9999	34.1124	
21	0.205	22	0.9999	0.0028	0.0004	-0.0035	-0.0066	-0.1251	4.8595	0	0	-0.122	0.007	-0.0009	0.0056	0.007	-0.0008	-0.0056	-0.0009	0.0056	9.9999	34.1124	
22	0.209	23	0.9999	0.0028	0.0004	-0.0035	-0.0066	-0.1251	4.8595	0	0	-0.122	0.007	-0.0009	0.0056	0.007	-0.0008	-0.0056	-0.0009	0.0056	9.9999	34.1124	
23	0.226	24	0.9999	0.0027	0.0004	-0.0037	-0.0066	-0.1275	4.8691	0	-0.061	-0.122	0.0074	-0.0009	0.0054	0.0074	-0.0008	-0.0054	-0.0009	0.0054	9.9999	34.1124	
24	0.234	25	0.9999	0.0026	0.0004	-0.004	-0.0048	-0.1281	4.8691	0	-0.061	-0.122	0.0079	-0.0008	0.0051	0.0079	-0.0007	-0.0051	-0.0008	0.0051	9.9999	34.0947	
25	0.238	26	0.9999	0.0026	0.0004	-0.004	-0.0048	-0.1281	4.8691	0	-0.061	-0.122	0.0079	-0.0008	0.0051	0.0079	-0.0007	-0.0051	-0.0008	0.0051	9.9999	34.0947	
26	0.255	27	0.9999	0.0024	0.0004	-0.0042	-0.0048	-0.1245	4.8709	0	-0.061	-0.122	0.0084	-0.0008	0.0049	0.0084	-0.0007	-0.0049	-0.0008	0.0049	9.9999	34.1006	
27	0.264	28	0.9999	0.0023	0.0003	-0.0045	-0.009	-0.1275	4.8703	0	-0.061	-0.122	0.0089	-0.0006	0.0045	0.0089	-0.0006	-0.0045	-0.0006	0.0045	9.9999	34.1241	
28	0.266	29	0.9999	0.0023	0.0003	-0.0045	-0.009	-0.1275	4.8703	0	-0.061	-0.122	0.0089	-0.0006	0.0045	0.0089	-0.0006	-0.0045	-0.0006	0.0045	9.9999	34.1241	
29	0.285	30	0.9999	0.0021	0.0003	-0.0047	-0.0024	-0.1305	4.8670	0	-0.061	-0.122	0.0094	-0.0006	0.0042	0.0094	-0.0006	-0.0042	-0.0006	0.0042	9.9999	34.1065	

[illegible]

The encryption subsection of the results in this report features two images. An image with the original sample data from the IMU and another image showcasing an encrypted version of the original sample data. The encrypted data has passed the first experiments to test the functionality of the encryption block. Firstly, it does not resemble the original data and finally it cannot be read/understood.



## 6. Validation using IMU

### Importance of hardware-based validation

Hardware based validation is when the hardware is to be used in a process or task that can have any effect on the quality, safety, efficacy of the product or product mandated data. This is therefore particularly important in that it is not always that old data will be used in conjunction with using the designed system. It is actually the case mostly that hardware is used in conjunction with a designed system. Hardware based validation is also important in that it allows for evaluation of the system in real time and therefore giving a realistic view of how the system behaves in general.

In this project hardware-based validation is important in that it allows for analysis of compression and encryption algorithms with direct IMU data. This therefore giving a realistic view of the system when running in real time.

### Steps for validation using data from IMU in real time

#### Step 1 :

The first step is configuring the IMU with the stm32

#### Step 2:

Reading data from the IMU

#### Step 3:

Saving data from IMU in a file

#### Step 4:

Sending over the data to compression block to be compressed

#### Step 5:

Sending data to the encryption block to be encrypted and decrypted

#### Step 6:

Sending data back to compression block to be decompressed

#### Step 7:

Analysing results of both the encryption and compression block

### IMU Module

#### Salient features on ICM 20948

The IMU used in the SCALE Sharc Buoy is the ICM-20649. The IMU used in the developed Intellectual Property (IP) is the ICM 20948. Some salient features of the ICM 20948 that differ from the ICM 20649 are presented in the table below.

Feature in ICM 20948	Difference to ICM 20649
----------------------	-------------------------

Gyroscope has a user programmable full-scale range of $\pm 250$ , $\pm 500$ , $\pm 1000$ and $\pm 2000$ degrees per second (dps)	The ICM 20649 on the other had has a user programmable full-scale range of $\pm 500$ , $\pm 1000$ , $\pm 2000$ , $\pm 4000$ degrees per second (dps)
Accelerometer has a user programmable full-scale range of $\pm 2g$ , $\pm 4g$ , $\pm 8g$ , $\pm 16g$ .	ICM 20649 has user programmable full-scale range of $\pm 4g$ , $\pm 8g$ , $\pm 16g$ , $\pm 30g$ .
I <sup>2</sup> C communication frequency of 100 KHz (standard mode) or up to 400 KHz (fast mode)	I <sup>2</sup> C communication frequency of only 400 KHz (fast mode)
Gyroscope has sensitivity scale factors of 131, 65.5, 32.8, and 16.4 Least Significant Bytes (LSB) per dps. This corresponds to the $\pm 250$ , $\pm 500$ , $\pm 1000$ and $\pm 2000$ dps respectively.	ICM 20649 gyroscope has sensitivity scale factors of 65.5, 32.8, 16.4, 8.2 Least Significant Bytes (LSB) per dps. This corresponds to the $\pm 100$ , $\pm 500$ , $\pm 1000$ , $\pm 2000$ dps respectively.

Steps to take to ensure ICM 20948 can be extrapolated to ICM 20649

1. **Configure the gyroscope to a full-scale range of  $\pm 500$  degrees per second** – The ICM-20948 and ICM-20649 have similar gyroscope full scale ranges of  $\pm 500$  degrees per second (dps) and  $\pm 1000$  dps. The  $\pm 500$  dps corresponds to a sensitivity scale of 65.5 LSB/dps for both IMU's. Furthermore, the  $\pm 1000$  dps corresponds to a sensitivity scale of 32.8 LSB/dps. Since it is more ideal to have a higher sensitivity scale, the ICM 20948 should be configured to a full-scale range of  $\pm 500$  dps to have the highest sensitivity scale and mimic the ICM 20649
2. **Configure the Accelerometer to a full-scale range of  $\pm 16g$**  – The highest common accelerometer full scale range on both IMU's is 16g. Therefore, it will be necessary to configure the ICM 20948 to an accelerometer full scale range of 16g to ensure that the results are similar to the ICM 20649.
3. **Configure the I<sup>2</sup>C communication frequency to 400 KHz** – Whilst the ICM-20948 has two communication frequencies (standard mode = 100KHz and fast mode = 400KHz) the ICM-20649 can only communicate in fast mode (400KHz). The ICM-20948's I<sup>2</sup>C communication should, therefore, be configured to fast mode in order to mimic the ICM-20649.

Validation tests for ICM 20948

Test ID	Test Name	Explanation of Test
TID001	Rotational test	The ICM-20948 has a gyroscope which tracks angular velocity. The rotational test will test this by rotating the IMU at different speeds, directions

		and angles and observing the behaviour of the output data. If the gyroscope data is changing, the gyroscope is working as expected, however, if it is not changing than it is inactive.
<b>TID002</b>	<b>Change of speed test</b>	The ICM-20948 also has an accelerometer which measures acceleration. The change of speed test will test this through shaking the IMU module at different velocities. If the accelerometer data changes with changes to the shaking, the IMU works as expected. If not, the IMU is inactive.
<b>TID003</b>	<b>Static angle test</b>	When the IMU is stationary, the gyroscope data should remain constant. The static angle test will test this by rotating the IMU to 6 different angles/positions. The IMU will be left in these angles for 1 minute and the output data will be observed. If the IMU gyroscope data remains constant then the IMU operates as expected.

## Results for ICM 20948

All tests were done by changing all axis of the IMU. However only the X coordinates results are recorded and seen in the demonstration video under gyroDataX. This data is again shows under compression results below.

### TID001

ICM20948 reacts to rotational movement and therefore test is passed. This can be seen under compression block results with the gyroDataX used showing reaction of the ICM to rotational movement.

### TID002

The ICM2094 reacts to speed changes. The accelerometer tests also pass.

### TID003

The IMU reacts to the constantly to the static angle test. Test was met. This is also seen under compression block results below under that screenshot showing the gyroDataX.

## Analysis Results for ICM 20948

The ICM20948 passes all the required tests. Meaning that the IMU functions well with the system and system input data is valid.

## Compression block

### *How it works*

Code for compression is written in c. The code runs on windows using the terminal. For compression, a csv txt data format that only consists of numbers with the first block representing time is used as an input file this is received from the IMU. On Windows terminal make all is run to compile all the compression and decompression code blocks. To run compression 6pack –compression level input file output file is run on the terminal. This converts the txt file to a the output file of any format (for testing a txt file was used) and therefore representing the compressed file. The compression code allows for benchmarking which can be indicated by inputting –mem where followed by compression level and input file. Decompression occurs by running 6unpack Compressed file.

### *Functionality*

To test the functionality of compression block, data is collected from the IMU.

Compression code has different level in which it can compress. For testing functionality of the block both levels must be tested.

### *Experimental setup*

For testing data from the IMU used is the Gyro X coordinate only, however the IMU can output all the other functionalities. It is only for testing that a sample of only the gyro X coordinate data is used.

### *Compression level 1*

#### **Compression test 1:**

Test to see if data from IMU compresses.

#### **Compression test 2:**

Test to see if compressed file of data from IMU decompresses.

#### **Compression test 3:**

Benchmark compression of data from the IMU. Achieve at least 40% space saving and processing of less than 10 seconds.

#### **Compression test 4:**

Compare decompressed file with input data. 90% of the data must be identical.

### *Compression level 2*

The same tests as compression level 1 apply.

## Results

#### **Compression test 1:**

IMU data compresses successfully for both compression level 1 and 2

#### **Compression test 2:**

Compressed IMU data/ decrypted data decompresses successfully for both compression level 1 and 2

#### **Compression test 3:**

	Compression level 1	Compression level 2
IMU input data	Data compression speed: 20.0 Mbyte/s. Data decompression speed: 642.1 Mbyte/s. Data compression space saved: 60.4% saved	Data compression speed: 22.0 Mbyte/s. Data decompression speed: 478.5 Mbyte/s. Data compression space saved: 60.4% saved

#### Compression test 4:

Belows is a screenshot of the IMU output data

```
===== PuTTY log 2022.10.16 17:01:32 =====
[0.06097561]
[-0.06097561]
[-0.12195122]
[0.00000000]
[0.12195122]
[0.12195122]
[0.00000000]
[-0.06097561]
[-64.32926941] |
[8.96341515]
[4.32926846]
[53.29268265]
[-9.26829243]
[-70.73171234]
[6.21951246]
[2.07317066]
[-100.18292999]
[10.91463470]
[-35.73170853]
[13.23170757]
[0.06097561]
[0.24390244]
[-14.87804890]
[84.87805176]
[-1.82926834]
[1.89024389]
[-24.63414764]
[12.50000000]
[-4.87804890]
[-0.24390244]
[-2.43902445]
[0.42682928]
[-2.92682934]
```

Below is a screenshot of the decompressed data

Size of data should be smaller than original

```

===== PuTTY log 2022.10.16 17:01:32 =====
[0.06097561]
[-0.06097561]
[-0.12195122]
[0.00000000]
[0.12195122]
[0.12195122]
[0.00000000]
[-0.06097561]
[-64.32926941] |
[8.96341515]
[4.32926846]
[53.29268265]
[-9.26829243]
[-70.73171234]
[6.21951246]
[2.07317066]
[-100.18292999]
[10.91463470]
[-35.73170853]
[13.23170757]
[0.06097561]
[0.24390244]
[-14.87804890]
[84.87805176]
[-1.82926834]
[1.89024389]
[-24.63414764]
[12.50000000]
[-4.87804890]
[-0.24390244]
[-2.43902445]
[0.42682928]
[-2.92682934]

```

### Analysis of results

Compression block functions as expected as both the compression and decompression run successfully. Decompressed file is identical to input data and therefore adding to the functionality of the compression block. When looking at compression test 3; compression level 2 is slower than compression level 1. The difference is not that much, and this is expected as with truly little size data the various levels operate very similarly. This again is seen in space saved being 60.4% for both compression level 1 and level 2. Decompression rate is not very much depended on anything as previously mentioned. Conservation of 25% of the Fourier coefficients occurs as 100% of the input data is retrieved on decompression. System works well as space saved is more than the minimum 40%. The processing also happens in less than 10 seconds.

## Encryption block

### Experimental setup

When the encrypted data is decrypted, the data should be exactly the same as the original compressed data that entered the encryption block. The data should be the same to ensure that no meaningful/important data is lost in the encryption process

### Results

To test the encryption block with the IMU data, ATP002 becomes relevant:

<b>ATP Name</b>	ATP002: Encryption Test
<b>Description</b>	This test checks whether a file has been successfully encrypted. This is tested by physically analysing the readability of the encrypted data.
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The encrypted data is <b>False</b> when compared to the input data to the system

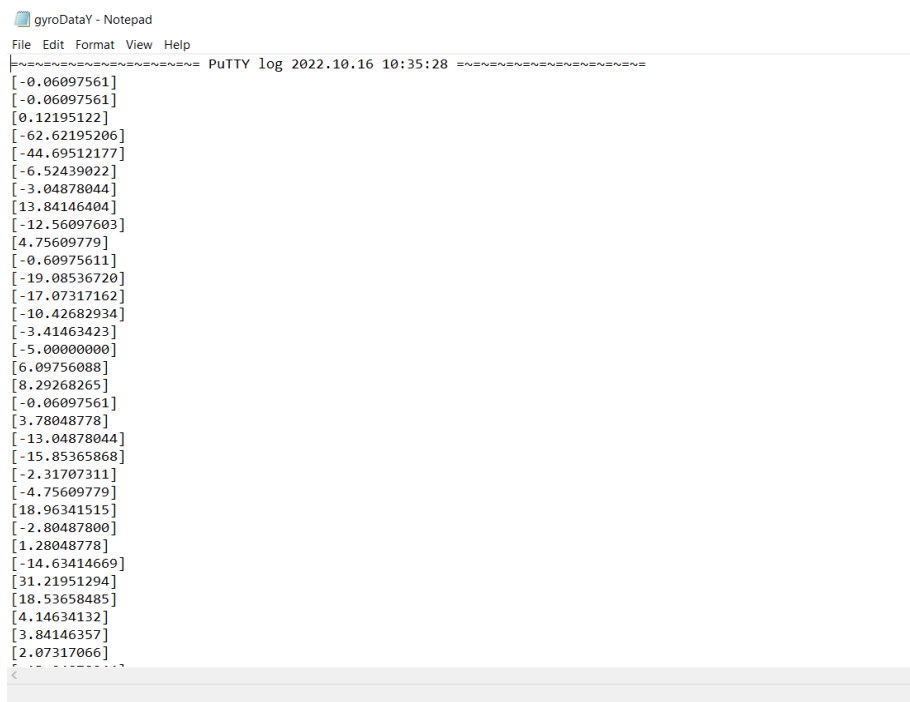
## Fail Conditions

The decrypted/decompressed data is **True** when compared to the input data to the system

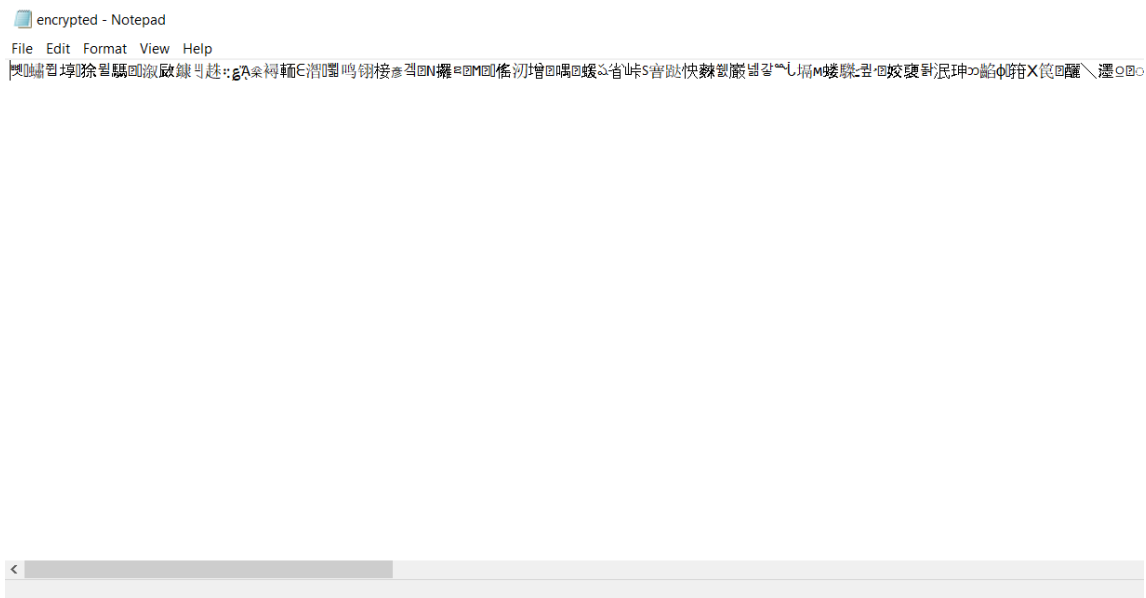
Using ATP002, a simple python script can be written to compare 2 files. One that contains raw text and another that is encrypted.

If the two files pass the test i.e the script outputs “false,” then the encryption algorithm works with the ICM 20948 data.

## Results



*An Image of gyroDataY.txt with gyroscope data from ICM 20948*



*An Image of encrypted.txt with encrypted gyroscope data from ICM*

The two images above in this results section showcase the gyroscope y coordinate data and the encrypted version of that data. The python script written for ATP002 can now be used to test these two files.

#### Comparing the Initial and Encrypted Files

```
In [2]: >>> import filecmp
>>> filecmp.cmp('puttyData/gyroDataY.txt', 'puttyData/encrypted.txt')

Out[2]: False
```

*A Jupyter notebook snippet with the ATP002 script code*

In the above image, the ATP002 script code has been implemented on the two data files. The result is “False,” which signals that the encryption algorithm is working as expected on the ICM20948 data



## Consolidation of ATPs and Future Plan

### Recreated ATPs

Everything stayed the same, the only difference was in place of using a python script for some analysis; that is changed to analysis by physical comparison of graphs and data.

ATP ID	ATP Name	Specification Addressed
ATP001	Accuracy Test	SP003
ATP002	Encryption Test	SP005
ATP003	Compression Test	SP006
ATP004	Frequency Test	SP002
ATP005	Low power data retrieval	SP004
ATP006	Low power system	SP005

<b>ATP Name</b>	ATP001: Accuracy Test
<b>Description</b>	This test checks whether the decrypted and decompressed data is similar to the original input data to the system. This will be tested in comparing decompressed output graphs to the input data graphs. This is done by physical analyzation.
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The decrypted/decompressed data is <b>True</b> when compared to the input data to the system
<b>Fail Conditions</b>	The decrypted/decompressed data is <b>False</b> when compared to the input data to the system

<b>ATP Name</b>	ATP002: Encryption Test
<b>Description</b>	This test checks whether a file has been successfully encrypted. This is tested by physically analysing the readability of the encrypted data.
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The encrypted data is <b>False</b> when compared to the input data to the system
<b>Fail Conditions</b>	The decrypted/decompressed data is <b>True</b> when compared to the input data to the system

<b>ATP Name</b>	ATP003: Compression Test
<b>Description</b>	This test checks whether a file has been successfully compressed. After compression, a compression ratio will be outputted.
<b>Figure of Merit</b>	Compression saving space of at least 40%
<b>Pass Conditions</b>	The compressed file has a compression space saving of 40% or more

<b>Fail Conditions</b>	The compressed file has a compression space saving of less than 40%
------------------------	---

<b>ATP Name</b>	ATP004: Frequency Test
<b>Description</b>	This test checks whether 25% of the Fourier coefficients have been preserved. For this test the DFT of the original data is compared to the DFT of the decompressed/decrypted data. Only when the data is not identical is a python script used to compare decompressed data with input data. Otherwise this is analysed by physical analysis of the graphs of input and decompressed data.
<b>Figure of Merit</b>	True/False output
<b>Pass Conditions</b>	The decrypted/decompressed data is <b>True</b> when compared to the input data DFT. This signifies that 25% or more of the Fourier coefficients have been preserved.
<b>Fail Conditions</b>	The decrypted/decompressed data is <b>False</b> when compared to the input data DFT of the system to show that 25% or less of the Fourier coefficients have been preserved.

<b>ATP Name</b>	ATP005: Lower Power data retrieval
<b>Description</b>	This test checks that the communication between the stm32f0 discovery board and PC is minimal. This is done through inspection by checking the CPU usage of the PC
<b>Figure of Merit</b>	Numerical figure of 10%
<b>Pass Conditions</b>	The communication between the PC and microcontroller should less than 10% of the CPU usage on the PC
<b>Fail Conditions</b>	The communication between the PC and microcontroller uses more than 10% of the CPU usage on the PC

<b>ATP Name</b>	ATP006: Low Power System
<b>Description</b>	This test is similar to ATP005. Instead of monitoring the power usage, it monitors the power usage of the encryption/compression algorithms on the PC. This is also done via inspection of the CPU usage.
<b>Figure of Merit</b>	Numerical value of 50%
<b>Pass Conditions</b>	The encryption and compression algorithms use less than 50% of the CPU usage
<b>Fail Conditions</b>	The encryption and compression algorithms use more than 50% of the CPU usage.

## ATPs met or not

ATPs	Explanation
<u>ATP001: Accuracy Test</u> This test checks whether the decrypted and decompressed data is like the original input data to the system. A python script will be written to compare the two files. The script will output a binary True/False after the comparison	The ATP (Acceptance Test Procedure) was met. The decrypted, decompressed file is 100% identical to the input data file in all the tests performed (both simulation and hardware-based validation)
<u>ATP003: Compression Test</u>  This test checks whether a file has been successfully compressed. After compression, a compression ratio will be outputted	This ATP was met. All data files used compressed successfully with compression saving space more than the minim 40%. Processing also happens fast in less than maximum of 10 seconds.
<u>ATP004: Frequency test</u>  This test checks whether 25% of the Fourier coefficients have been preserved. For this test the DFT of the original data is compared to the DFT of the decompressed/decrypted data. The same python script that has been used for previous test will be used.	ATP was met. 100% identity was found between decompressed data and input data. Graphs of decompressed data also presented to give the same out as the graphs used for input data in the simulation set up.
<u>ATP002: Encryption test</u> This test checks whether a file has been successfully encrypted. The same python script used for ATP001 will be used for this test Encrypted data should not be readable at all. To test this a csv file will be used for testing as that can be easily analyzed physically if data encrypts or not. The readability of the data will be compared by physical comparison of the eyes. Data must be encrypted 100%.	The ATP was met. All data files after going through the encryption block are not readable. The numbers are represented by different symbols that are not understandable and therefore meaning the encryption code works. The data was 100% encrypted.
<u>ATP005: Low power data retrieval</u> This test checks that the communication between the stm32f0 discovery board and PC is minimal. This is done through inspection by checking the CPU usage of the PC	This was met in that data from the IMU was retrieved in less than 10 seconds without laptop crashes and overheating. This is seen in the demonstration video.

#### ATP006: Low Power System

This test is similar to ATP005. Instead of monitoring the power usage, it monitors the power usage of the encryption/compression algorithms on the PC. This is also done via inspection of the CPU usage.

This was met in that both compression and encryption codes ran in less than 10 seconds whilst also not causing any crash in the system. There was not observable overheating in the system.

#### Future work before implementation

According to the ATPs, ATPs 1,2,3 and 4 were met. Meaning the system can compress and encrypt properly without any errors. To ensure the validity of this even more before using the system with the bouy more work must be done in testing the accuracy of the decompressed data with the input data. Therefore, using a python script that compares the two files will allow for more sufficient analysis that will allow for even bigger sized data to be analysed. Based on our ATPs being met the algorithms for both compression and encryption do not need to change. Given that before the specifications where changed the system had to run on a stm32, this is what needs to happen with our system before using it on the bouy. This will also allow proper analysis of power consumption of the two algorithms. Before using the system with the bouy the system needs to be implemented in a way it would be used in the real world. In that decryption have in one device separately while encryption and compression occur in one device. All in all the system seems to be in good condition however before using it more tests need to be done whilst also being done computationally for more accuracy. Moreover, the system needs to be implemented in the stm32 while also being developed in a way that it would be used in the real world. Power tests would have to occur computationally as well way for more accuracy in that aspect.

#### Conclusion

In conclusion we developed a system that is able to compress and encrypt data from the IMU ICM 20948. We have however designed our system in that it can still compress and encrypt data from the ICM 20649 as well given that is the IMU the sharc bouy the system is designed for uses. Subsequently the system can decrypt and decompress the previously encrypted and compressed data. Steps we took into having a running system include, analysis of requirements and jotting down of specifications. It is through this step that we understood fully what is required of us in the project. We then took a step of creating paper design for the system which includes analysis of the different encryption and compression algorithm to be used. While also including different functional feasibilities of the system and bottlenecks. Lastly this section consists of the inter sub system design. We further took a step of validating the system using simulation data. This is done after all the compression and encryption code is coded. The simulation-based validation allows for analysis of the system and debugging. It is in this step that the initial XOR compression algorithm used failed. It was thereafter that an alternate dictionary algorithm was used. The dictionary algorithm used is the Fast LZ algorithm. With the new algorithm, simulation testing was done again. This all round completed the system in that it allowed for the system to work as a whole being able to compress and encrypt. We then moved to the hardware testing step. This is where IMU is implemented and tested. It is then that compression and encryption are tested using the data from the IMU. Moving on to the final step we evaluated our system with ATPs previously established. The system passed all the ATPs. It is however to be noted that a more computational comparison check can be used to ensure 100% accuracy in the result gotten in this project.

## References

[1] "Advantages and Disadvantages of Cryptography." Advantages And Disadvantages Of Cryptography - 1220 Words — 123 Help Me, <https://www.123helpme.com/essay/Advantages-And-DisadvantagesOf-Cryptography-PJ8CFSKM826>

[2] Application Notes on XOR Compression of Bit-Transposed Vectors, <http://bitmagic.io/bm-xor.html>.

[3] "Blowfish Algorithm: An Interesting Overview in 7 Points." Jigsaw Academy, 28 June 2022, <https://www.jigsawacademy.com/blogs/cyber-security/blowfishalgorithm/>.

[4] Callas, Jon. "Triple DES: How Strong Is the Data Encryption Standard?" SearchSecurity, TechTarget, 31 May 2017, <https://www.techtarget.com/searchsecurity/tip/Expert-adviceEncryption-101-Triple-DESexplained: :text=Triple%20DES%20encryption%20processtext=It%20works%20by%20taking%20three,the%20first%20and%20last%20steps> .

[5] Lockerman, Joshua. "Time-Series Compression Algorithms, Explained." Timescale Blog, Timescale Blog, 30 May 2022, <https://www.timescale.com/blog/time-series-compressionalgorithms-explained/>.

[6] Ltd, All Answers. "Blowfish Algorithm Advantages and Disadvantages." UK Essays, UK Essays, 29 July 2022, <https://www.ukessays.com/essays/computer-science/blowfish-algorithm-history-and-backgroundcomputer-science-essay.php>.

[7] Puneet. "What Is Blowfish in Security? Who Uses Blowfish?" Encryption Consulting, 17 Aug. 2022, <https://www.encryptionconsulting.com/education-center/what-isblowfish/>.

[8] "RF Wireless World." Advantages of AES — Disadvantages of AES, <https://www.rfwireless-world.com/Terminology/Advantagesand-disadvantages-of-AES.html>.

[9] Simplilearn. "What Is AES Encryption and How Does It Work? - Simplilearn." Simplilearn.com, Simplilearn, 29 July 2022, [https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption: :text=The%20AES%20Encryption%20algorithm%20\(also,together%20to%20form%20the%20ciphertext](https://www.simplilearn.com/tutorials/cryptography-tutorial/aes-encryption: :text=The%20AES%20Encryption%20algorithm%20(also,together%20to%20form%20the%20ciphertext).

[10] "What Are the Operations of Blowfish Algorithm?" Tutorials Point, <https://www.tutorialspoint.com/what-are-the-operations-ofblowfish-algorithm>.

# Appendix

## Appendix A

### Design Requirements: EEE3097S

**Background:** Some academics in the department of electrical engineering work in the larger RDI consortium called [SCALE](#). In one of the projects where we are involved, we are building a generic [flexible buoy that could be installed on the pancake ice in Southern Ocean to gather environmental data](#). You can read more about this buoy, which we have named as SHARC buoy (!) in [Jamie's MSc dissertation](#). His dissertation is also a good example of how to approach a design problem.

One of the crucial sensors in our buoy is the IMU. It gives information about the ice as well as wave dynamics. Ideally, we would like as much of the IMU data as possible. But, transmitting the data using Iridium is extremely costly. So, we want to compress the IMU data. We also want to encrypt the data.

**Problem Statement:** Design an ARM based digital IP using the Raspberry-Pi to encrypt and compress the IMU data. Further requirements are given below. These can be used to derive the exact specifications for your design.

Req1. The IMU to be used is [ICM-20649](#).

However, we shall not provide you these IMUs. We shall provide a different IMU. So, you shall need to find out ways to design your IP without the exact hardware! Here is a link to the device we will give you: [Waveshare Sense Hat \(B\)](#)

Req2. Oceanographers have indicated that they would like to be able to extract at least 25% of the Fourier coefficients of the data. Make sure that your compression satisfies this.

Req3. In addition to reducing the amount of data we also want to reduce the amount of processing done in the processor (as it takes up power which is limited). Try to minimize the computation required for your IP.

**Suggested Methodology:** This is a task with a few limitations. The biggest of which might be the lack of the actual sensor! Here is a suggested methodology.

- 1) **Requirement Analysis or Paper Design:** The first step is to understand the requirements and convert them into hard specifications. Then you must divide your IP into sub-systems and specify the inter and intra sub-system interactions. Lastly, you should work on detailed acceptance test procedures, i.e. how would you demonstrate that your design satisfies all the specifications.
- 2) **Implementation and Demonstration using Data from Computer:** In checking the functionality of your codes, you can set-up tests by sending data from your computer. These set of validations can be your next deliverable.

- 3) **Implementation and Demonstration using Data from an On- board Sensor:** Lastly, you need to show that the Pi can process data from an on-board sensor. Choose any sensors you think to be pertinent and use them to validate that your algorithms work.



Screenshot 1 below shows compression using compression level 1

```
C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -1 Data1.csv OutData1.txt
Data1.csv [#####] 66.1% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -1 Data2.csv OutData2.txt
Error: could not open Data2.csv

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -1 Data2.txt OutData2.txt
Error: file OutData2.txt already exists. Aborted.

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -1 Data2.txt OutData2.txt
Data2.txt [#####] 66.1% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -1 Data3.csv OutData3.txt
Data3.csv [#####] 66.2% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -1 Data4.csv OutData4.txt
Data4.csv [#####] 65.9% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>
```

Screenshot 2 below shows compression using compression level 1

```
C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -2 Data1.csv OutData1.txt
Data1.csv [#####] 66.5% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -2 Data2.txt OutData2.txt
Data2.txt [#####] 66.5% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -2 Data3.csv OutData3.txt
Data3.csv [#####] 66.6% saved

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -2 Data4.csv OutData4.txt
Data4.csv [#####] 66.2% saved
```

Screenshot 3 below shows benchmarking of compression level 1

```
C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -1 Data1.csv
Reading source file...
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 1, please wait...

Compressed 14332601 bytes into 4815368 bytes (33.6%) at 202.3 Mbyte/s.

Decompressed at 511.2 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -1 Data2.txt
Reading source file...
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 1, please wait...

Compressed 14332600 bytes into 4815853 bytes (33.6%) at 202.3 Mbyte/s.

Decompressed at 487.3 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -1 Data3.csv
Reading source file...
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 1, please wait...

Compressed 13862081 bytes into 4639048 bytes (33.5%) at 201.2 Mbyte/s.

Decompressed at 499.0 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -1 Data4.csv
Reading source file...
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 1, please wait...

Compressed 9619149 bytes into 3250540 bytes (33.8%) at 197.8 Mbyte/s.

Decompressed at 487.4 Mbyte/s.
```

Screenshot 4 shows benchmarking of compression level 2

```
C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -2 Data1.csv
Reading source file....
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 2, please wait...

Compressed 14332601 bytes into 4758686 bytes (33.2%) at 201.3 Mbyte/s.

Decompressed at 494.2 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -2 Data2.txt
Reading source file....
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 2, please wait...

Compressed 14332600 bytes into 4758763 bytes (33.2%) at 198.6 Mbyte/s.

Decompressed at 503.7 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -2 Data3.csv
Reading source file....
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 2, please wait...

Compressed 13862081 bytes into 4585869 bytes (33.1%) at 200.2 Mbyte/s.

Decompressed at 496.6 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -2 Data4.csv
Reading source file....
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 2, please wait...

Compressed 9619149 bytes into 3214663 bytes (33.4%) at 197.7 Mbyte/s.

Decompressed at 493.8 Mbyte/s.
```

## Appendix C

Below shows a screenshot of IMU data benchmarking

```
C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -1 gyroDataX.txt
Reading source file....
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 1, please wait...

Compressed 762 bytes into 460 bytes (60.4%) at 20.0 Mbyte/s.

Decompressed at 642.1 Mbyte/s.

(1 MB = 1000000 byte)

C:\Users\Kimmy\Downloads\FastLZ-master\FastLZ-master\examples>6pack -mem -2 gyroDataX.txt
Reading source file....
Setting HIGH_PRIORITY_CLASS...
Benchmarking FastLZ Level 2, please wait...

Compressed 762 bytes into 460 bytes (60.4%) at 22.0 Mbyte/s.

Decompressed at 478.5 Mbyte/s.

(1 MB = 1000000 byte)
```