MBKMUN001

Nathan Mboko

TSHDAV008

David Tshimbalanga

EEE3097S Submission 3: Final Report

Table of Contents

1 . Admin section

1.1. Table of contributions:

	MBKMUN001	TSHDAV008
Contribution	Compression	Encryption
Section numbers	**1 - 8**	**1 - 8**
Page numbers	1 - 45	1 - 45

Table1: Showing the contribution of each of the team members

1.2. Snapshot of your project management tool:

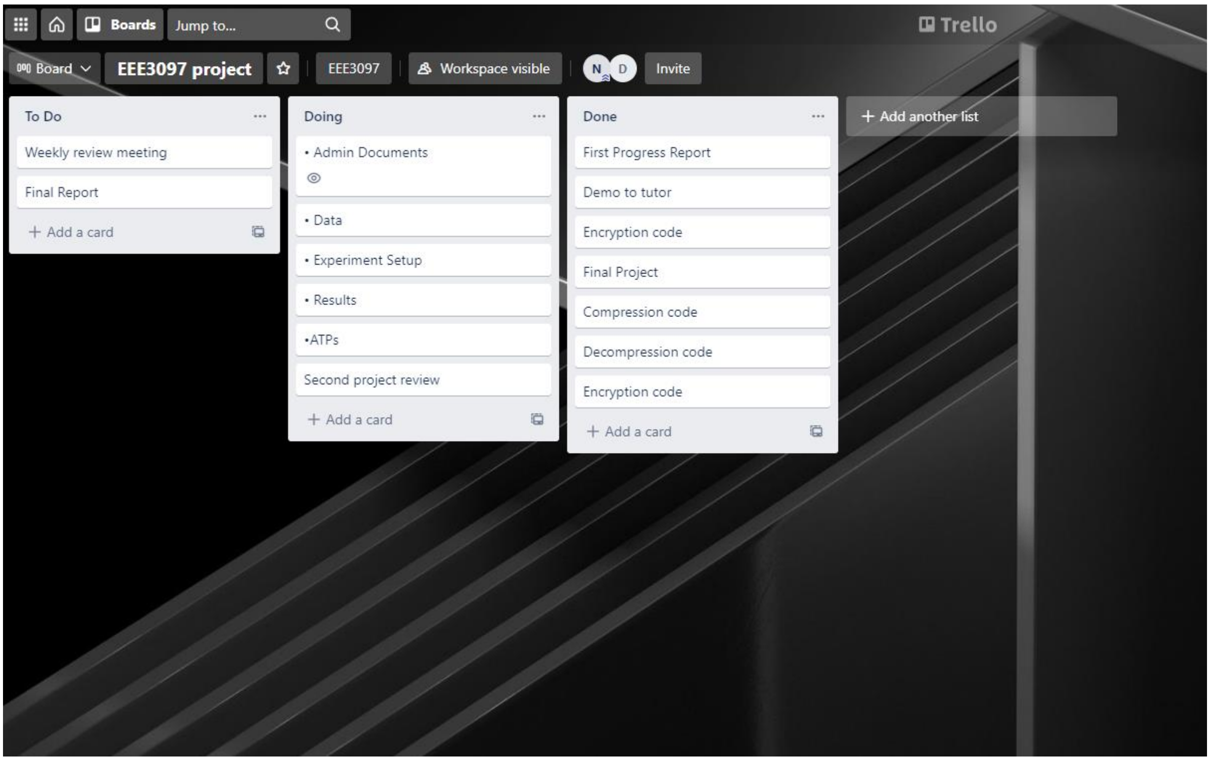


Figure1: Showing project management tool

1.3. Link to GitHub page:

The link to the Git of the project is <https://github.com/Nathan-mboko/EEE3097S-Project >

1.4. Timeline of the project:

Figure2: Showing timeline diagram

2. Requirement Analysis

2.1. Requirements

Initially 3 requirements were given for the design of the digital IP. They are as follows:

Req1. The IMU to be used is ICM-20649.

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. Minimize the computation required for your IP.

From these requirements the specifications were derived.

2.2. Analyse of the requirements using existing research or reports.

In a study to monitor high-frequency Ocean Signals a small, low-cost and self-assembly autonomous Inertial Measurement Unit (IMU) that independently collects continuous acceleration and angular velocity data is mounted on a GNSS buoy to provide the positions and tilts of the moving buoy.The main idea is to integrate the Differential GNSS (DGNSS) or Precise Point Positioning (PPP) solutions with IMU data, and then evaluate the performance by comparing with in situ tide gauges.The validation experiments conducted in the NCKU Tainan Hydraulics Laboratory showed that GNSS and IMU both can detect the simulated regular wave frequency and height, and the field experiments in the Anping Harbor, Tainan, Taiwan showed that the low-cost GNSS buoy has an excellent ability to observe significant wave heights in amplitude and frequency. [10]

2.3. List of requirements

1. The design of this project will be centred around the parameters and the specifications of the ICM-20649. The ICM-20649 will be wirelessly connected to the Raspberry Pi where the data will be compressed and encrypted to be used at a later date.

2. The compression algorithm should allow the client to have at least 25% of the data initially sent from the system.

3. The algorithm should be kept simple to reduce the amount of computation done by the processor to reduce the power consumption

2.4. List of specifications

1. Compression ratio should be more than should be above 50%

2. The encrypted data should not be in a readable format

3. The Decrypted data file should contain all the data from the Input Data file

4. The Big O notation of the compression and decompression algorithms should at most be n squared.

5. The Big O notation of the encryption and decryption algorithms should at most be n squared.

2.5. Design acceptance test procedure (ATP)

Specification	ATP
Compression ratio should be more than should be above 50%	Compare the compressed file size to the file size of the Input data and ensure that the Compressed file is less than half the size of the Input Data file
The encrypted data should not be in a readable format	Inspect the contents of the encrypted file to ensure it is not in a readable format
The Decrypted data file should contain all the data from the Input Data file	Run a program which compares the data in the output file of the system to the data present in the input data file
The Big O notation of the compression and decompression algorithm should at most be n squared.	Observe how the Compression and Decompression Algorithms performs when large input data files are put into the algorithm
The Big O notation of the encryption and decryption algorithm should at most be n squared.	Observe how the Encryption and Decryption Algorithms performs when large input data files are put into the algorithm

Table2: Showing the list of ATPs

3. Paper Design

● Requirement Analysis

○ *Interpretation of the requirements*:

1. The design of this project will be centred around the parameters and the specifications of the ICM-20649. The ICM-20649 will be wirelessly connected to the Raspberry Pi where the data will be compressed and encrypted to be used at a later date.

2. The compression algorithm should allow the client to have at least 25% of the data initially sent from the system.

3. The algorithm should be kept simple to reduce the amount of computation done by the processor to reduce the power consumption

○ *Comparison of some available compression and encryption algorithms*

1. Compression

- Run Length Encoding: Classified as a lossless compression algorithm as in this case there is close to no loss in the data compression when one wants to retrieve it. They are reversible. It runs on sequences with the same value occurring many consecutive times. It encodes the sequence to store only a single value and its count.

- Huffman coding: Also classified as a lossless compression algorithm. It is reversible. Input characters are assigned variable-length codes based on the frequencies of corresponding characters with the most frequent characters getting the smallest code.

- Block transform coding: Classified as a lossy compression algorithm. Some data values are lost in the compression. Transform coding compresses image data by representing the original signal with a small number of transform coefficients.

2. Encryption:

- Reverse Cipher Algorithm

● Reverse Cipher uses a pattern of reversing the string of plain text to convert as cipher text.

● The process of encryption and decryption is same.

● To decrypt cipher text, the user simply needs to reverse the cipher text to get the plain text.

- Transposition Cipher Algorithm

Transposition Cipher is a cryptographic algorithm where the order of alphabets in the plaintext is rearranged to form a cipher text.

- XOR Algorithm

XOR algorithm of encryption and decryption converts the plain text in the format ASCII bytes and uses XOR procedure to convert it to a specified byte. It offers the following advantages to its users

○ ***<u>Feasibility analysis</u>***

<u>1. Compression</u>

- Run Length Encoding:

This type of algorithm is reversible therefore all the data is kept however there are more disadvantages as they have very low compression ratios which is bad for this system as the amount of data received is high. Also, there is a high number of computations needed for this algorithm. Therefore, it is a bad choice.

- Huffman coding:

Similarly, to the previous one, it has good data recovery but a very low compression ratio which makes it a bad choice.

- Block transform coding:

Although there is a loss in data in this algorithm, the requirement for the system is to have at least 25% recovery which is in the range of the algorithm. Also, the compression ratios are very high which is perfect as the amount of data received is very high. It also has a relatively low amount of computation. These facts make this algorithm the best choice.

<u>2. Encryption:</u>

Reverse Cipher Algorithm:

While this algorithm is relatively simple and requires the least amount of computation, one major drawback of the reverse cipher algorithm is that it is very weak. A hacker can easily break the cipher text to get the original message. Hence, reverse cipher is not considered as a good option to maintain a secure communication channel.

Transposition Cipher Algorithm:

This algorithm provides a high level of data security but a large amount of computation making inefficient for the Raspberry Pi processor

XOR Algorithm:

This algorithm provides fast computation, it is easy to understand and analyze and it allows us to choose the encrypted file size. Due to these reasons we've decided to implement this algorithm in our encryption process

○ ***Possible bottlenecks***

1. A lack of good synchronization between the compression and encryption algorithm can lead to a bottleneck in the system.

2. The difference in specs between the Wi-Fi antenna on the pi and the ICM-20649 can lead to a bottleneck of the component with the higher specs.

3. The compression algorithm could compress too much and lead to a very high loss in data quality

● Subsystem Design

○ ***Subsystem and Sub-subsystems Requirements***

1. Compression: At least 25% of the data should be recovered

2. Encryption: The encryption should be done on the Pi and it should minimize the amount of computation needed and the description should produce the previous non-encrypted data.

○ ***Subsystem and Sub-subsystems***

Specifications

1. Compression:

40% of the data should be recovered

2. Encryption:

The encrypted data is decryptable

○ ***Inter-Subsystem and Inter-Sub-subsystems Interactions***

The output data from the compression program will be used as the input for the encryption program.

● UML

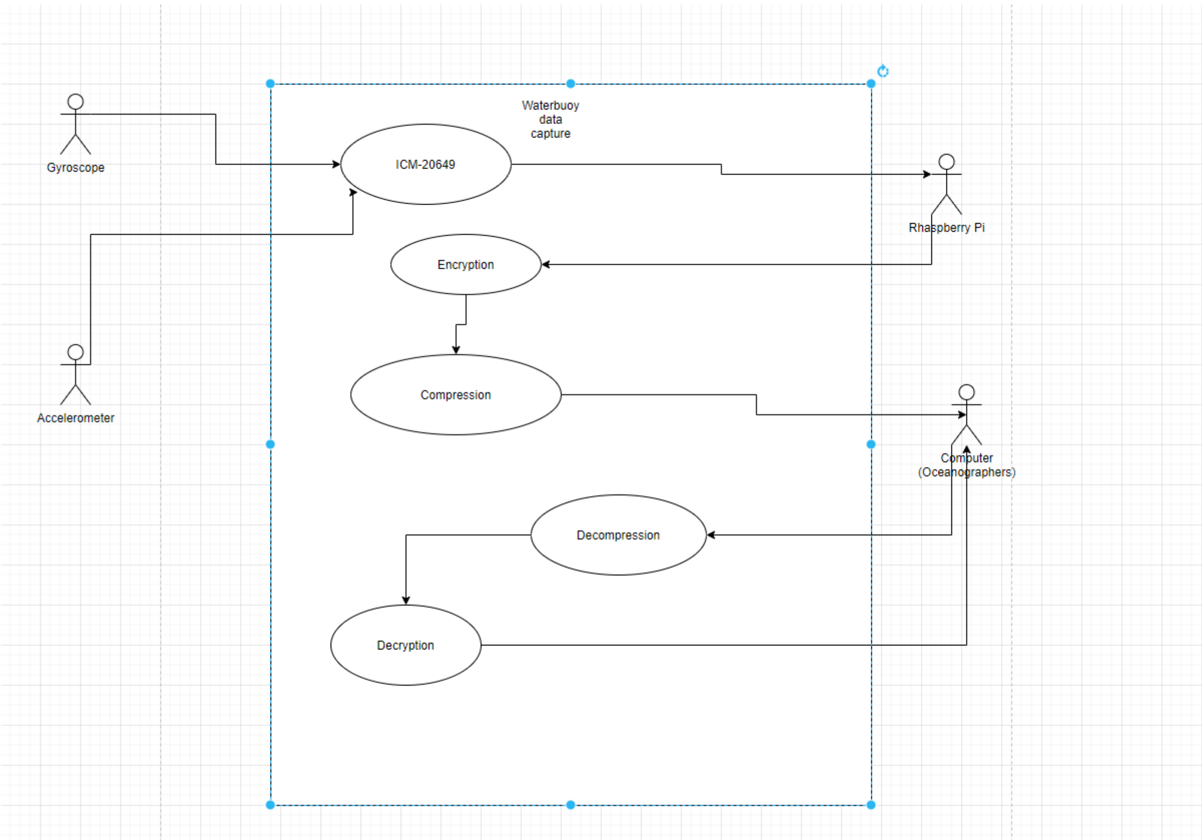


Figure2: Showing the UML diagram of the system

● Acceptance Test Procedure

○ ***<u>Figures of merits, based on which you would validate your final design</u>***

40% of fourier coefficients are recovered.

System output file size should be the same size as Input Data file

The Big O notation of the compression and encryption algorithms should at most be n squared.

○ ***<u>Experiment design to test these figures of merit</u>***

-Run a comparison test between initial data from the ICM and the final encrypted data to ensure that at least 40% of the fourier coefficients are compatible

-Check to see if the system output file size is equal to input data file size

-Calculate the Big O notation of the compression and encryption algorithms to test whether they have a Big O notation less than n squared

○ Acceptable performance definition

- System output file size should be the same size as Input Data file

- Big O notation of the compression and encryption algorithms must be less than n squared

- 40% Fourier coefficients must be recovered in the final encrypted file

4. Validation using Simulated or Old Data

4.1. Simulation-based validation.

Simulation modelling provides an efficient way to test and explore how a system responds to different types of data. Simulated experiments are less expensive and take less time than experiments with real hardware.

A simulation model can capture many more details than a physical model, providing increased accuracy and more precise results. Uncertainty in operation times and can be easily modelled, allowing you to quantify and for more robust solutions to be found.

4.2. List of steps.

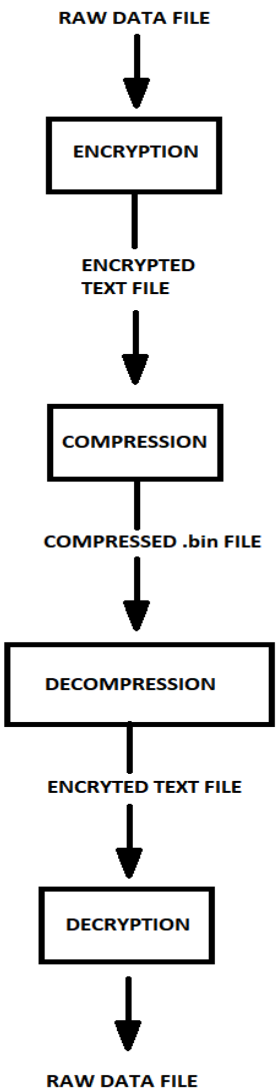


Figure3: Block diagram Showing list of steps

4.3. Data used the steps followed

The Data that was used in these experiments is the raw data received from an IMU on a boat.

Though the data used is from an IMU on a boat instead of an IMU on SHARC buoy as the Project is designed for, both the IMU on the boat and the IMU on the SHARC buoy have a similar function as they're both used to capture geographical positioning data.

The file size of the IMU boat Data that will be used can also be comparable to that file size of the Data of the IMU on the SHARC buoy.

4.4. Present the experimental setup and results

• Experiment Setup

System Experiments

Experiment 1:

This experiment will input various Input Data file sizes into the system and record how fast the system processes the various sized input data files to test the speed of the system. A timer module will be implemented in the program to determines its runtime.

The aim of this experiment is to identify the Big(O) value of the system and to observe how efficiently the system performs when a large amount of data is put into the system.

Experiment 2:

Different amount of Data inputs will be used in this experiment to test how much of the data is preserved in the output file of the system.

A program that compares the contents of the input data file and the systems output file will be run to observe how much of the Input data is present in the systems output file. The aim of this experiment is to ensure that all the data in the Raw data input file is found in the Systems output file.

Encryption

Experiment 1:

This experiment will input various Input Data file sizes into the encryption algorithm and record how fast the algorithm processes the various sized input data files to test the speed of the algorithm. A timer module will be implemented in the algorithm to determines the encryption algorithms runtime.

The aim of this experiment is to identify the Big(O) value of the encryption algorithm and to observe how efficiently the system performs when a large amount of data is put into the system.

Experiment 2:

This Experiment will test how the encryption algorithm changes the format Raw Data Input file to the unreadable encrypted format.

The aim of this experiment is to determine how well the Encryption algorithm can mask the data to ensure the security of the data captured by the IMU on the Sharc Buoy.

Compression

Experiment 1:

Different file sizes of the input data are used to test the speed of the compression algorithm. The files used were the outputs of the encryption code that is used before this one. In the compression algorithm a timer module was used to determine how fast it is,

The goal of this experiment is to identify the Big (O) value of the compression algorithm and observe how the compression algorithm behaves when a large amount of data is entered into the algorithm.

Experiment 2:

This experiment tests how the compression algorithm compresses the encrypted file.

The goal of this experiment is to determine how well the compression algorithm can reduce the data size to make the data transfer faster as the there is a lot of data.

Decompression

Experiment 1:

Different file sizes of the input data are used to test the speed of the decompression algorithm. The files used were the outputs of the compression code that is used before this one. In the decompression algorithm a timer module was used to determine how fast it is,

The goal of this experiment is to identify the Big (O) value of the decompression algorithm and observe how the decompression algorithm behaves when a large, compressed file is entered.

Experiment 2:

This experiment tests how the decompression algorithm changes the format of the compressed file to the encrypted file.

The aim of this experiment is to compare the compressed data Input file to the output decompressed file to ensure that these two files are same.

Decryption

Experiment 1:

Various Input Data file sizes will be put into the decryption algorithm and record how fast the algorithm processes the various sized input data files to test the speed of the algorithm. A timer module will be implemented in the algorithm to determines its runtime.

The aim of this experiment is to identify the Big(O) value of the decryption algorithm and to observe how the decryption algorithm performs when a large amount of data is put into the algorithm.

Experiment 2:

This experiment will be used to check that the decrypted file is the same as the Raw Data Input file.

The aim of this experiment is to compare the Raw Data Input file to the output decrypted file to ensure that these two files are same.

• Results

System Experiment 1: Testing the performance of the entire system by measuring the runtime of system program using different file sizes

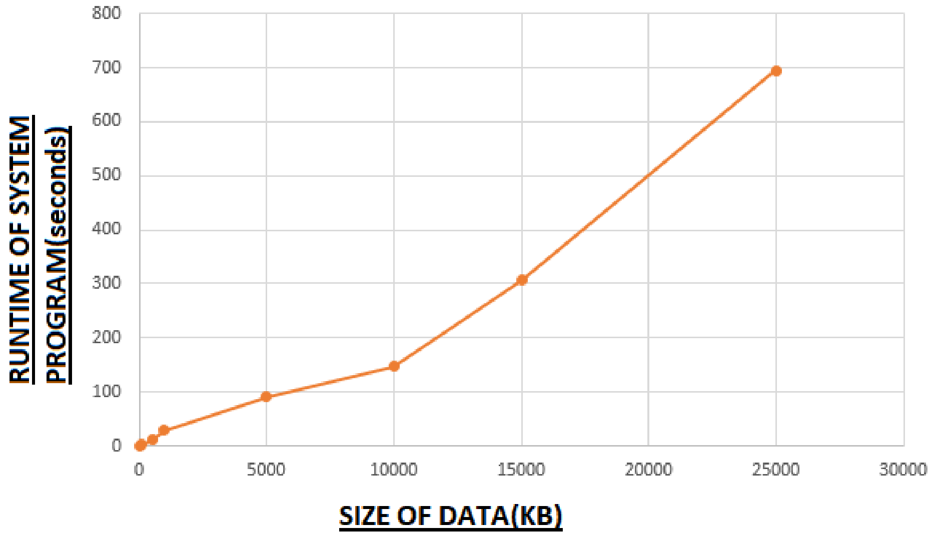


Figure4: Showing the result of System on a runtime vs bytes graph

The results above show that the system's Big(O) value is equal to N. This means that the best and worst-case order of complexity of the entire system is linear. While a linear order of complexity is not the most efficient algorithm the system is still capable of meeting the requirements of the Project.

System Experiment 2: This experiment will be used to verify how much of the Raw input data is preserved in the output data file of the system

Amount of Data in Raw Data Input file (kilobytes)	Percentage of Data found in Output System File (%)
10340	100
91193	100
576404	100
1246242	100
4996744	100
8225821	100
17248183	100
25004631	100

Table 3: Showing the amount of Data in Raw Data Input file compared to amount of data in System output file

The Table show that all the data in the Raw Data input file can be found in the System output file. This means that no data is lost within the system. This ensures that all the data that captured by the IMU Sharc Buoy will processed by our System.

The Comparison Program compares the position and value of the items with the Input Data file and the System Output Data file to ensure that the content of both files are the exact same.

Link to Comparison Program: https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/Comparison_Program.py

ENCRYPTION

Experiment 1: Testing the performance of the encryption algorithm by measuring the runtime of the encryption program using different file sizes

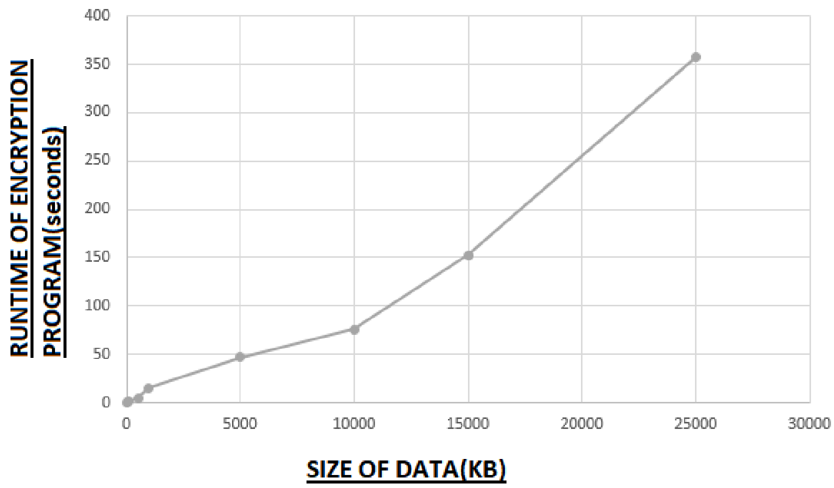


Figure5: Graph showing runtime of encryption algorithm for different file sizes

The encryption algorithm's Big(O) value is equal to N though from the values that can be seen above. The best and worst-case order of complexity of the encryption algorithm is linear.

The Big(O) value of the Encryption algorithm is N which meets the required specification that the Big(O) of the Encrypted algorithm does not exceed N squared.

Link to Encryption program: https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/encrypt-new.py

EXPERIMENT 2: Ensuring that the encrypted file is not in a readable format

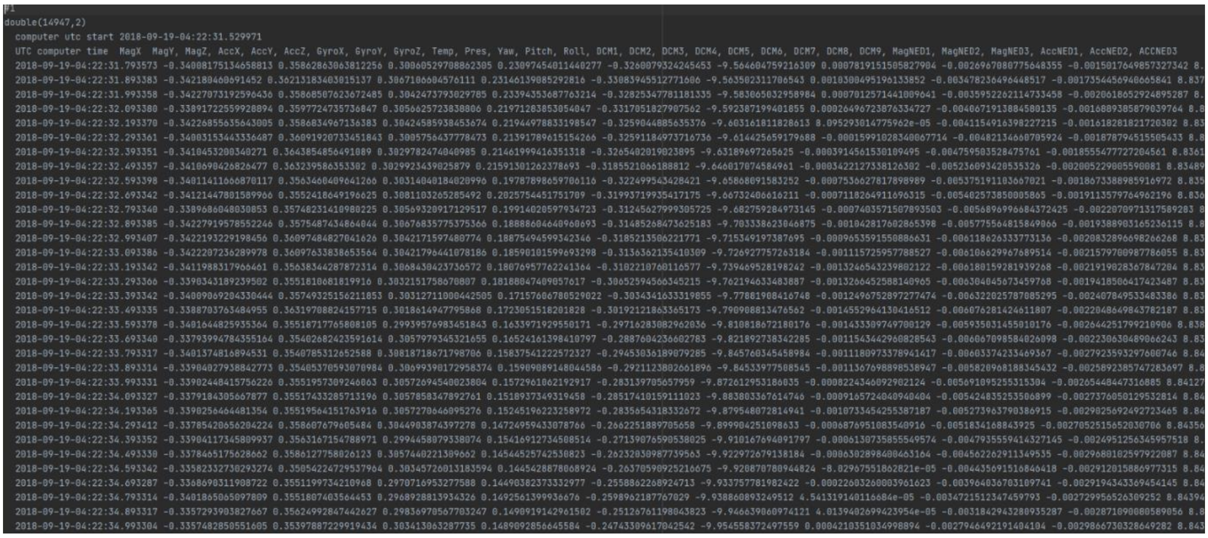


Figure6: Raw Data Input file

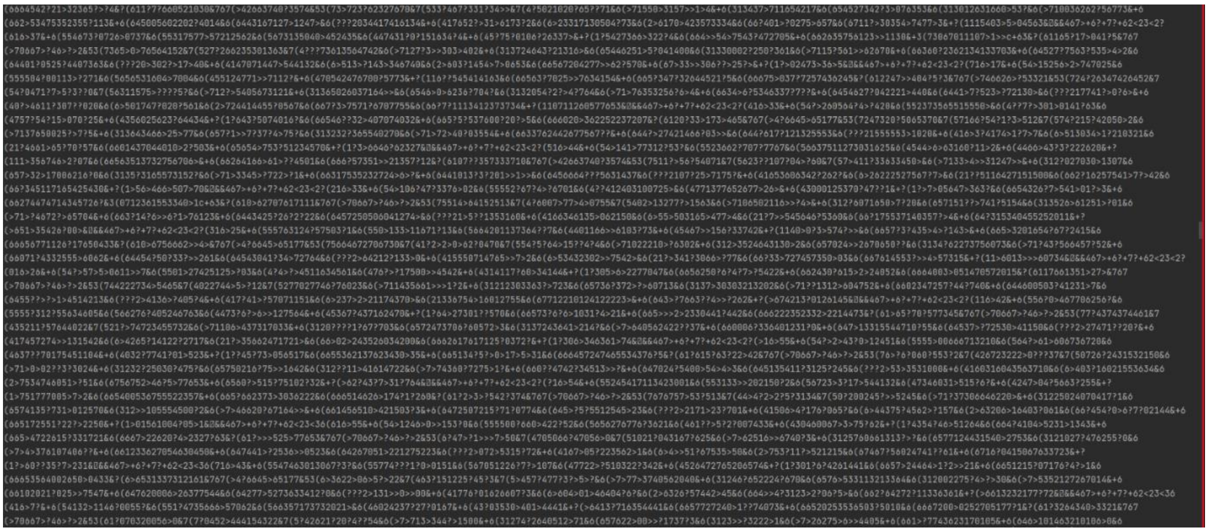


Figure7: Encrypted program output

From figure 10 it can be seen that the encrypted file has a particular format which makes it impossible to read off any data values without decrypting the file. This ensures that the data can only be seen by those who have access to the decryption key thus ensuring the privacy and safety of the data.

Link to Input Data files folder:< https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/data>

Link to the encrypted data file folder:< https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/Encrypt >

COMPRESSION

Experiment 1: Testing the performance of the compression algorithm by measuring the runtime of the compression program using different file sizes

The followings are the result of the compression algorithm using the simulated data. The input were text files that were generated from < https://onlinefiletools.com/generate-random-text-file>

The experiment was made with input files of weight 100B, 10KB, 100KB, 500KB, 1MB and 50MB. The results were:

Bytes	Percentage of compression (%)	Time elapsed	Compressed bytes size
100	62.00	0.0000287	62
1e4	62.56	0.0004117	6256
1e5	63.42	0.0033887	63420
5e5	63.53	0.0298400	317672
1e6	63.55	0.0346070	6355156
5e7	64.91	0.3868000	32456545

Table4: Showing the result of decompression

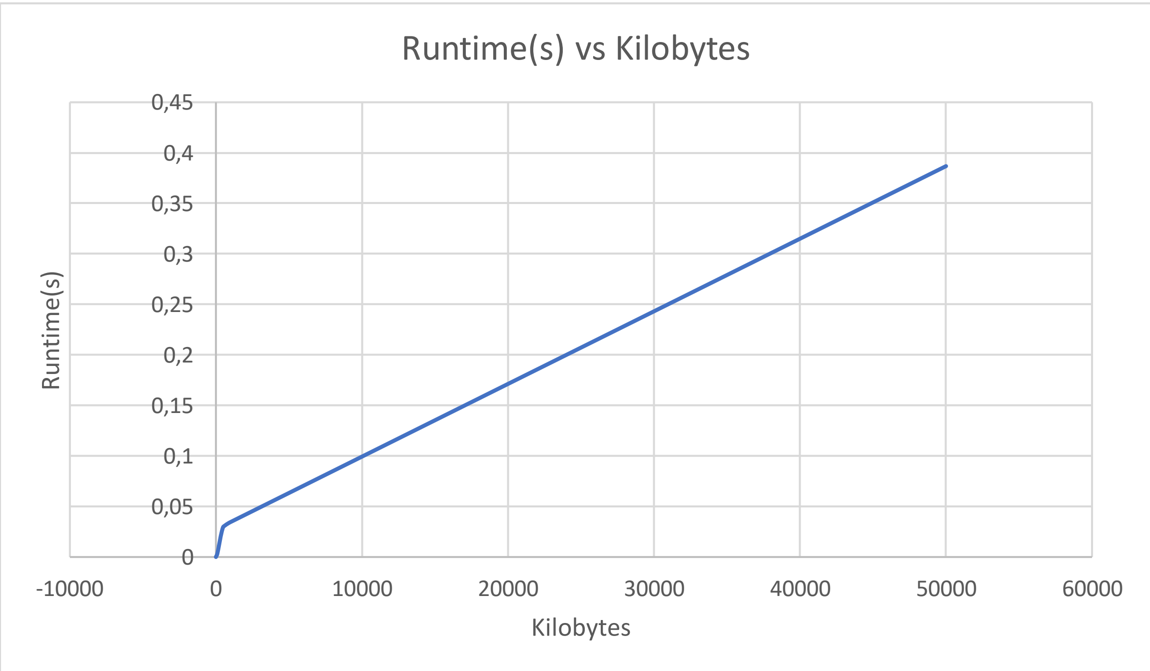


Figure8: Showing the result of compression on a runtime vs bytes graph

From the results above it can be seen that the encryption algorithm's Big(O) value is equal to N which corresponds to the algorithm that was used. This representation express that the bigger the size of the data, the longer (linearly) the time taken will be.

```
Time to compress:
0.0389872
size of original: 1000000
size of compressed: 635579
size of decompresse-d: 1000000

Process finished with exit code 0
```

Figure9: Showing the size of compression and decompression at the output

<u>Experiment 2: Ensuring that the compressed file is different than the encrypted one</u>

The following is a screenshot of one of the files that were encrypted

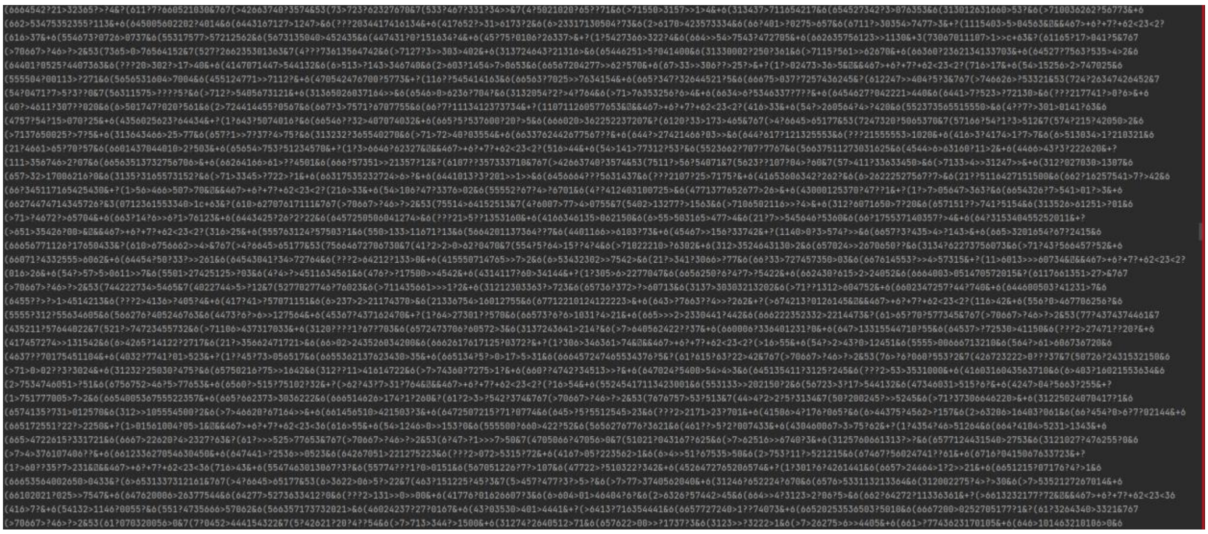


Figure10: Encrypted file content

Next, is a screenshot of the compressed files

Figure11: Output after compression

It can be seen that the compressed file characters are different than the encrypted file.

The output has to match with the requirements:

- The compression algorithm as more than 25% of the original data, therefore this requirement is met
- The algorithm that was used was simple and fast and this can be confirmed by the average time to compress a file with 0.3868s for a 500MB file containing around 5e7 characters. Therefore, the requirement for the algorithm to be fast is respected

Link to the code used for compression: < https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/compress.py>

Link to the compressed folders

DECOMPRESSION

Experiment 1: Testing the performance of the decompression algorithm by measuring the runtime of the decompression program using different file sizes

The followings are the result of the decompression code used on the previously compressed data.

These are the following files after the compression in the previous step:

Bytes	Time elapsed	Bytes decompressed
62	0.0000030	100
6256	0.0000187	1e4
63420	0.0002570	1e5
317672	0.0124600	5e5
6350000	0.0285000	1e6

Table 5: Showing the result of decompression

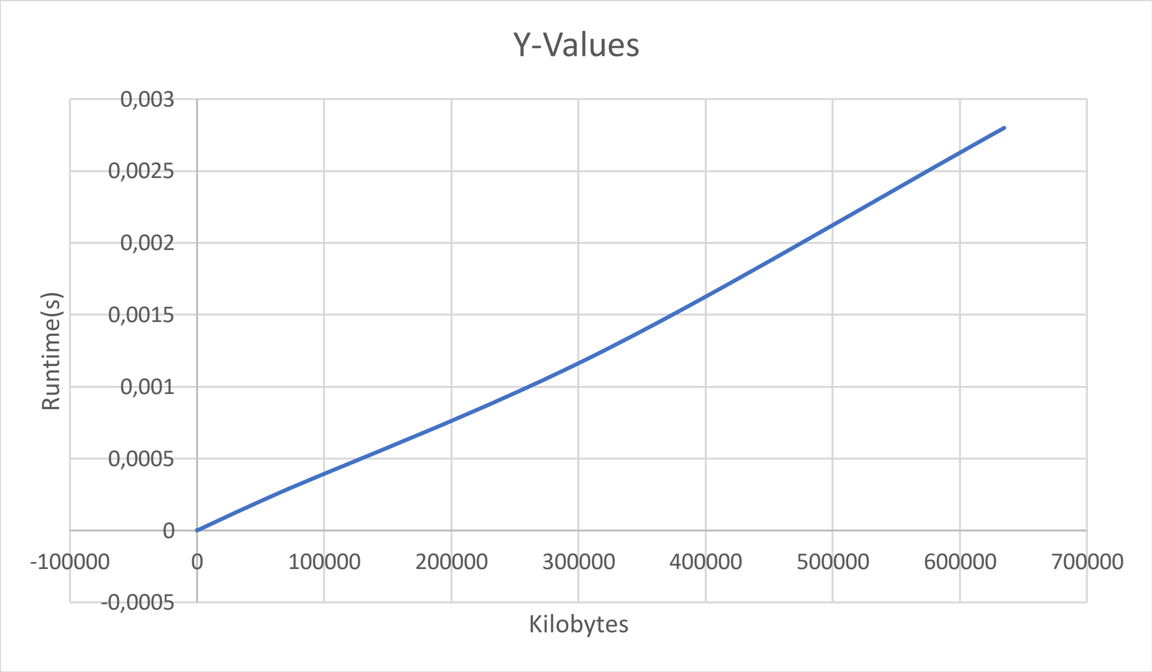


Figure12: Showing the result of the decompression on a runtime vs bytes graph

Similarly, to the compression data analysis, it can be seen from the results above that the decompression algorithm's Big(O) value is equal to N which corresponds to the algorithm that was used. This representation express that the bigger the size of the data, the longer (linearly) the time taken will be.

<u>Experiment2: Making sure that the decompressed file content looks similar to the encrypted one that was the input previously</u>

The following is a screenshot of on of the compressed file contents, the one used in the previous experiment:

Figure13: compressed file content

Next, is a screenshot of the output of the decompression program.

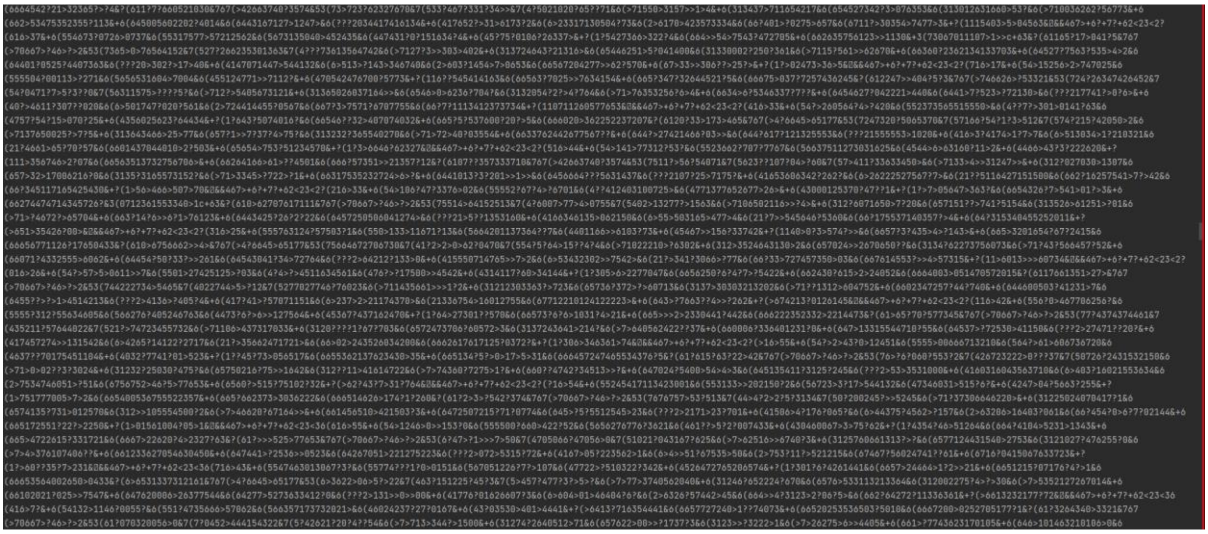


Figure14: Output after decompression

It is clear that the output after decompression (Figure12) is similar to the one in Figure8 therefore, the decompression algorithm works as it should.

The output must match with the requirement:

The algorithm that was used was simple and fast and this can be confirmed by the average time to decompress a 6.35MB file in 0.0285 seconds. Therefore, the requirement for the algorithm to be fast is respected

Here is a link to the code used for decompression: < https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/decompress.py>

DECRYPTION

Experiment 1: Testing the performance of the decryption algorithm by measuring the runtime of the decryption program using different file sizes

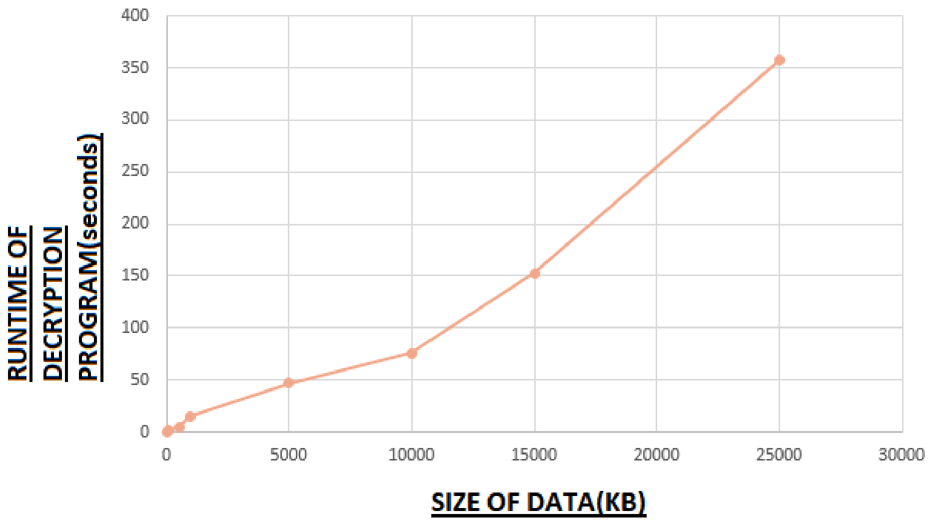


Figure15: Graph showing runtime of decryption algorithm for different file sizes

Like the encryption algorithm the decryption algorithm's Big(O) value is equal to N though from the values that can be seen above the decryption algorithm is slightly faster than the encryption algorithm. The best and worst-case order of complexity of the decryption algorithm is linear.

As previously stated, the Big(O) value of the Decryption algorithm is N which meets the required specification that the Big(O) of the Decrypted algorithm does not exceed N squared.

Link to Decryption Program:< https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/decrypt.py >

<u>EXPERIMENT 2: Ensuring that the decrypted file is the exact same as the raw data input file</u>

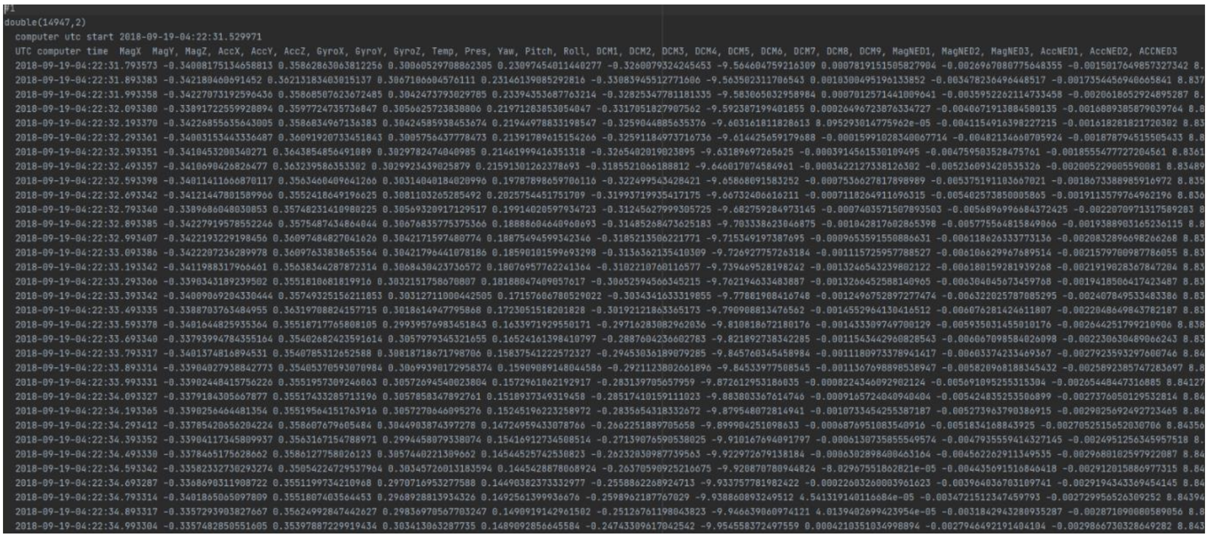


Figure16: Raw Data Input file

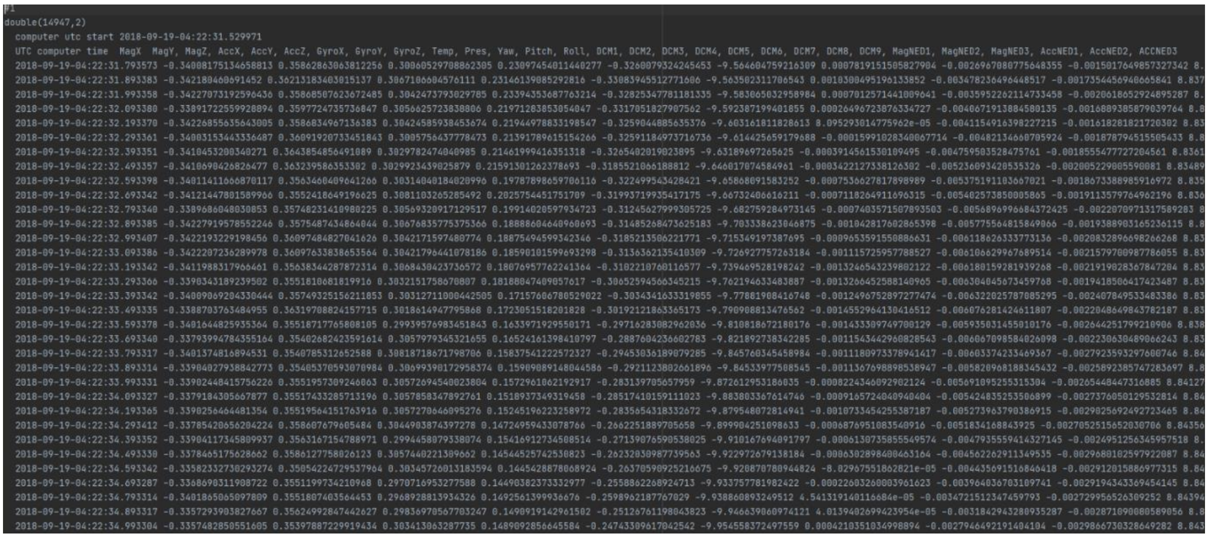


Figure17: Output file after Decryption

The output file of the decryption algorithm mirrors the Raw Data input file exactly this shows that all the data in the Raw data input file of the system is found in the decrypted output file of the system.

5. Validation using a different-IMU

5.1. Need to do a hardware-based validation.

Hardware based experiments allow the physical limitations of a system to be tested. To ensure that the program is compatible with the physical system. Hardware based Validation tests how external factors in the environment affect the operation of the physical system. It provides a more accurate measure of a systems performance as opposed simulation-based testing as hardware testing actually observes how the system perform sin the physical world where the system will be implemented as opposed to the theoretical environment in simulation-based testing.

5.2. List of steps.

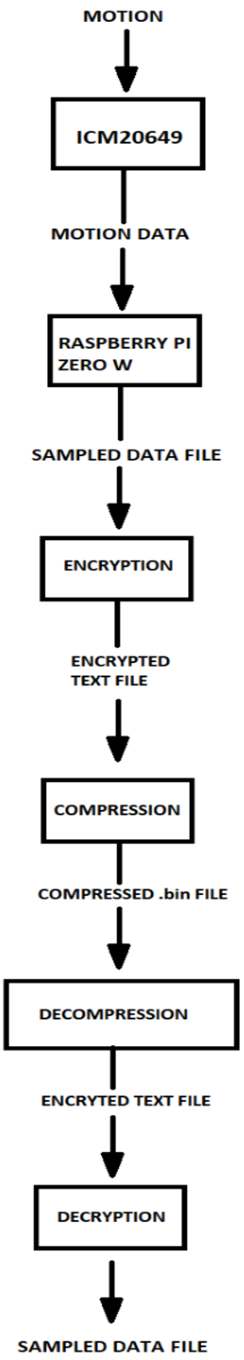


Figure18: Block diagram showing steps

5.3. Discuss the data you have used and the steps you followed.

An IMU is a specific type of sensor that measures angular rate, force and sometimes magnetic field. IMUs are composed of a 3-axis accelerometer and a 3-axis gyroscope, which would be considered a 6-axis IMU. The data that was used was sampled data received directly from the different sensors on the ICM20649. This sampled data was then put into a text file which was used as the Input Data file of the system.

5.4. Present the experimental setup and results

- **Experiment Setup**

<u>System Experiments</u>

Experiment 1:

This experiment will use various sizes of sampled Input Data from the ICM20649 to test the speed of the system. A timing module will be used to record the amount of time it takes for the system to process the data. The aim of this experiment is to identify the Big(O) value of the system and to observe how the system performs when a large amount of data is put into the system

Experiment 2:

Different amount of sampled input data from the ICM20649 will be used in this experiment to test how much of the data is preserved in the output file of the system program. The aim of this experiment is to ensure that all the data in the Raw data input file is found in the Systems output file. A program that compares the contents of the input data file and the systems output file will be run to observe how much of the Input data is present in the systems output file

<u>Encryption</u>

Experiment 1:

This experiment will input various Input Data file sizes into the encryption algorithm and record how fast the algorithm processes the various sized input data files to test the speed of the algorithm. A timer module will be implemented in the algorithm to determines the encryption algorithms runtime.

Experiment 2:

This Experiment will test how the encryption algorithm changes the format Raw Data Input file to the unreadable encrypted format.

The aim of this experiment is to determine how well the Encryption algorithm can mask the data to ensure the security of the data captured by the IMU on the Sharc Buoy.

Compression

Experiment 1:

The IMU will be used for various periods of time to get different file sizes of the input data which were encrypted in the previous experiment to be used to test the speed of the compression algorithm. This was done by adding a timer module to the code to determine how long it took for it to run.

This will help in determining the Big (O) value of the compression algorithm and observe how the compression algorithm behaves when a large amount of data is entered into the algorithm.

Experiment 2:

This experiment tests how the compression algorithm compresses the encrypted file.

This is meant to test if the output compressed file content is different that the input

Decompression

Experiment 1:

Different file sizes of the input data (compressed data) are used to test the speed of the decompression algorithm. This was done by adding a timer module to the code to determine how long it took for it to run.

The goal of this experiment is to identify the Big (O) value of the decompression algorithm and observe how the decompression algorithm behaves when a large, compressed file is entered.

Experiment 2:

This experiment tests how the decompression algorithm changes the format of the compressed file to the encrypted file.

This Experiment will test how the encryption algorithm changes the format Raw Data Input file to the unreadable encrypted format.

The aim of this experiment is to determine how well the Encryption algorithm can mask the data to ensure the security of the data captured by the IMU on the Sharc Buoy.

- **Results**

System Experiment 1: Testing the performance of the entire system by measuring the runtime of system program using different file sizes

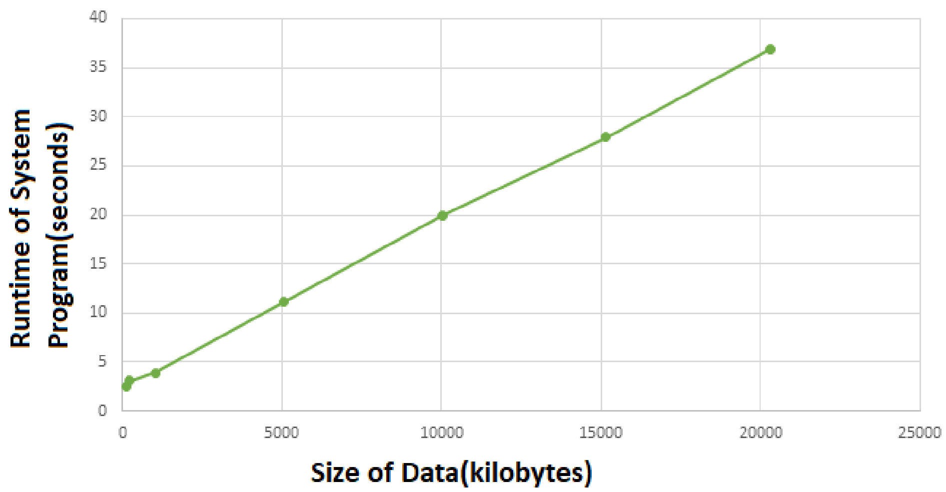


Figure19: Showing the result runtime of system program vs number of sampled input data graph

The results above show that the system's Big(O) value is equal to N. This means that the best and worst-case order of complexity of the entire system is linear. While a linear order of complexity is not the most efficient algorithm the system is still capable of meeting the requirements of the Project. The system also processes data faster than in the simulation as data is stored and processed in real time.

System Experiment 2: This experiment will be used to verify how much of the Raw input data is preserved in the output data file of the system

Amount of Data in Sampled Input Data file	Percentage of input data present in System output data file
105	100
1018	100

5033	100
10046	100
15151	100
20333	100

Table 6: Amount of Data in Raw Data Input file compared to amount of data in System output file

The Table show that all the data in the Sampled Input Data file can be found in the System output file. This means that no data is lost within the system. This ensures that all the data that captured by the IMU ICM20649 on the Sharc Buoy will processed by our System.

ENCRYPTION

Experiment 1: Testing the performance of the encryption algorithm by measuring the runtime of the encryption program using different file sizes

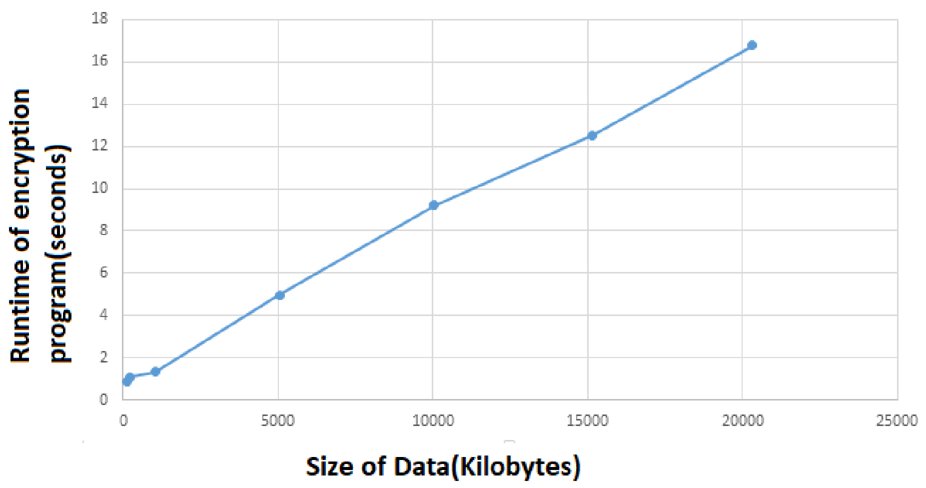


Figure20: Graph showing runtime of encryption algorithm for different file sizes

The results above show that the system's Big(O) value is equal to N. This means that the best and worst-case order of complexity of the encryption algorithm is linear. While a linear order of complexity is not the most efficient algorithm the system is still capable of meeting the requirements of the Project.

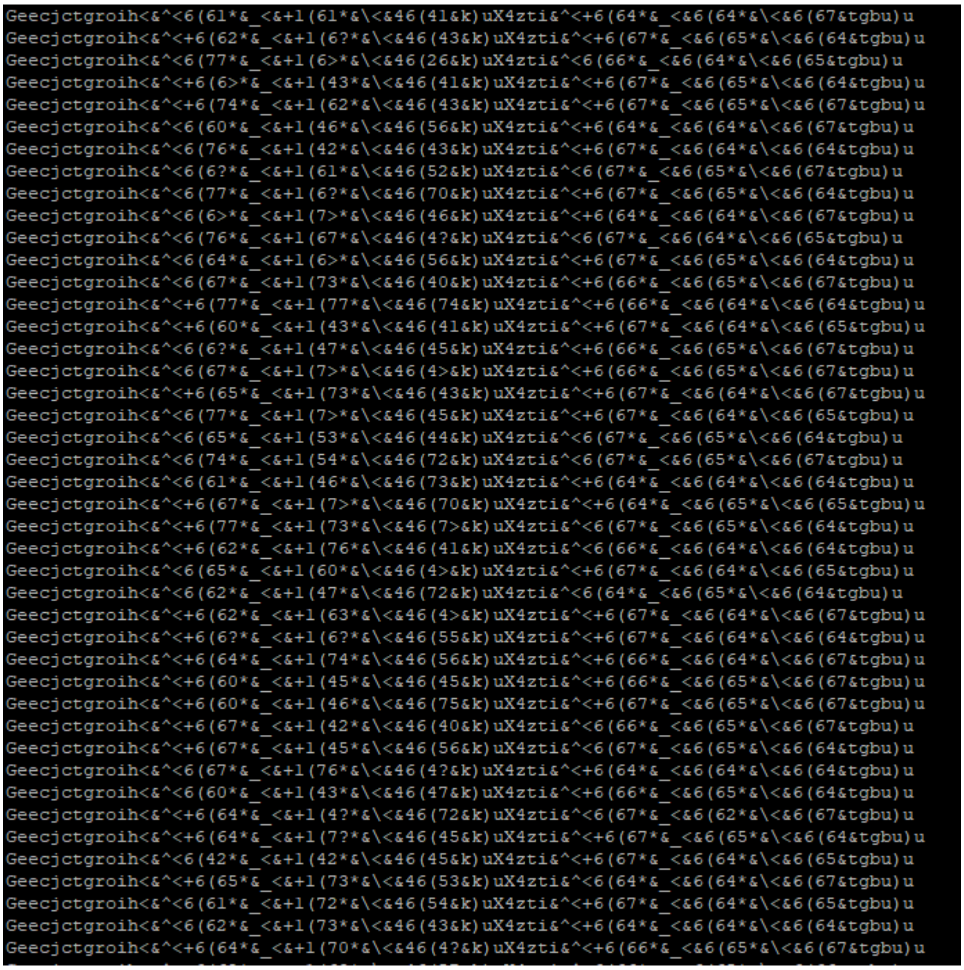
The Big(O) value of the Encryption algorithm is N which meets the required specification that the Big(O) of the Encrypted algorithm does not exceed N squared.

Link to Encryption program:< https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/encrypt-new.py >

EXPERIMENT 2: Ensuring that the encrypted file is not in a readable format

```
Acceleration: X:-0.02, Y: -7.27, Z: 19.97 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.07, Y: -7.07, Z: 20.27 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.04, Y: -7.09, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.11, Y: -7.08, Z: 20.40 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:-0.08, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.12, Y: -7.04, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.06, Y: -7.20, Z: 20.30 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.24, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.09, Y: -7.07, Z: 20.34 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.09, Z: 20.16 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.08, Y: -7.18, Z: 20.20 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.01, Z: 20.29 m/s^2|Gyro X:0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.02, Y: -7.08, Z: 20.30 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.15, Z: 20.26 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.11, Y: -7.11, Z: 20.12 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.06, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.09, Y: -7.21, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.01, Y: -7.18, Z: 20.28 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.03, Y: -7.15, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.18, Z: 20.23 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.03, Y: -7.35, Z: 20.22 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.12, Y: -7.32, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.07, Y: -7.20, Z: 20.15 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.01, Y: -7.18, Z: 20.16 m/s^2|Gyro X:-0.02, Y: 0.03, Z: 0.03 rads/s
Acceleration: X:-0.11, Y: -7.15, Z: 20.18 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.10, Z: 20.27 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.03, Y: -7.06, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.04, Y: -7.21, Z: 20.14 m/s^2|Gyro X:0.02, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.05, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.09, Y: -7.09, Z: 20.33 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.12, Z: 20.30 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.23, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.20, Z: 20.13 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.24, Z: 20.26 m/s^2|Gyro X:0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.23, Z: 20.30 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.10, Z: 20.29 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.06, Y: -7.25, Z: 20.21 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.29, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.04, Z: 0.01 rads/s
```

Figure21: Sampled Input Data file



```
Geecjctgroih<&^<6(61*&_<&+l(61*&\<&46(41&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(62*&_<&+l(6?*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(77*&_<&+l(6>*&\<&46(26&k)uX4zti&^<6(66*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<+6(6>*&_<&+l(43*&\<&46(41&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(74*&_<&+l(62*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(60*&_<&+l(46*&\<&46(56&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(76*&_<&+l(42*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(6?*&_<&+l(61*&\<&46(52&k)uX4zti&^<6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(77*&_<&+l(6?*&\<&46(70&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(6>*&_<&+l(7>*&\<&46(46&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(76*&_<&+l(67*&\<&46(4?&k)uX4zti&^<6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(64*&_<&+l(6>*&\<&46(56&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(67*&_<&+l(73*&\<&46(40&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(77*&_<&+l(77*&\<&46(74&k)uX4zti&^<+6(66*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(60*&_<&+l(43*&\<&46(41&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(6?*&_<&+l(47*&\<&46(45&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(67*&_<&+l(7>*&\<&46(4>&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(65*&_<&+l(73*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(77*&_<&+l(7>*&\<&46(45&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(65*&_<&+l(53*&\<&46(44&k)uX4zti&^<6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(74*&_<&+l(54*&\<&46(72&k)uX4zti&^<6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(61*&_<&+l(46*&\<&46(73&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(67*&_<&+l(7>*&\<&46(70&k)uX4zti&^<+6(64*&_<&6(65*&\<&6(65&tgbu)u
Geecjctgroih<&^<+6(77*&_<&+l(73*&\<&46(7>&k)uX4zti&^<6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(62*&_<&+l(76*&\<&46(41&k)uX4zti&^<6(66*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(65*&_<&+l(60*&\<&46(4>&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(62*&_<&+l(47*&\<&46(72&k)uX4zti&^<6(64*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(62*&_<&+l(63*&\<&46(4>&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(6?*&_<&+l(6?*&\<&46(55&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(74*&\<&46(56&k)uX4zti&^<+6(66*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(60*&_<&+l(45*&\<&46(45&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(60*&_<&+l(46*&\<&46(75&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(67*&_<&+l(42*&\<&46(40&k)uX4zti&^<6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(67*&_<&+l(45*&\<&46(56&k)uX4zti&^<6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(67*&_<&+l(76*&\<&46(4?&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(60*&_<&+l(43*&\<&46(47&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(4?*&\<&46(72&k)uX4zti&^<6(67*&_<&6(62*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(7?*&\<&46(45&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(42*&_<&+l(42*&\<&46(45&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<+6(65*&_<&+l(73*&\<&46(53&k)uX4zti&^<6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(61*&_<&+l(72*&\<&46(54&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(62*&_<&+l(73*&\<&46(43&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(70*&\<&46(4?&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
```

Figure22: Encrypted file output

From figure 8 the encrypted file has a particular format which makes it impossible to read off any data values without decrypting the file. This ensures that the data can only be seen by those who have access to the decryption key thus ensuring the privacy and safety of the data.

Link to Sampled Input Data file:< https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/data >

Link to Encrypted output data files:< https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/Encrypt>

COMPRESSION

Experiment 1: Testing the performance of the decryption algorithm by measuring the runtime of the decryption program using different file sizes

The followings are the result of the compression code used on the previously encrypted IMU data.

The experiment was made with input files of weight ranging from 87KB to 1.7MB that were collected from the IMU after being encrypted. The results were:

KB	Percentage of compression (%)	Time elapsed (sec)	Compressed size (KB)
19	89.7	0.0004957	1.944
87	80.45	0.0024822	7.152
112	86	0.0029327	16.081
430	76.74	0.0084317	33
858	77.32	0.0173595	66.346
1291	81.66	0.0267503	105.351
1739	79.03	0.0357112	139.449

Table 7: Showing the result of decompression

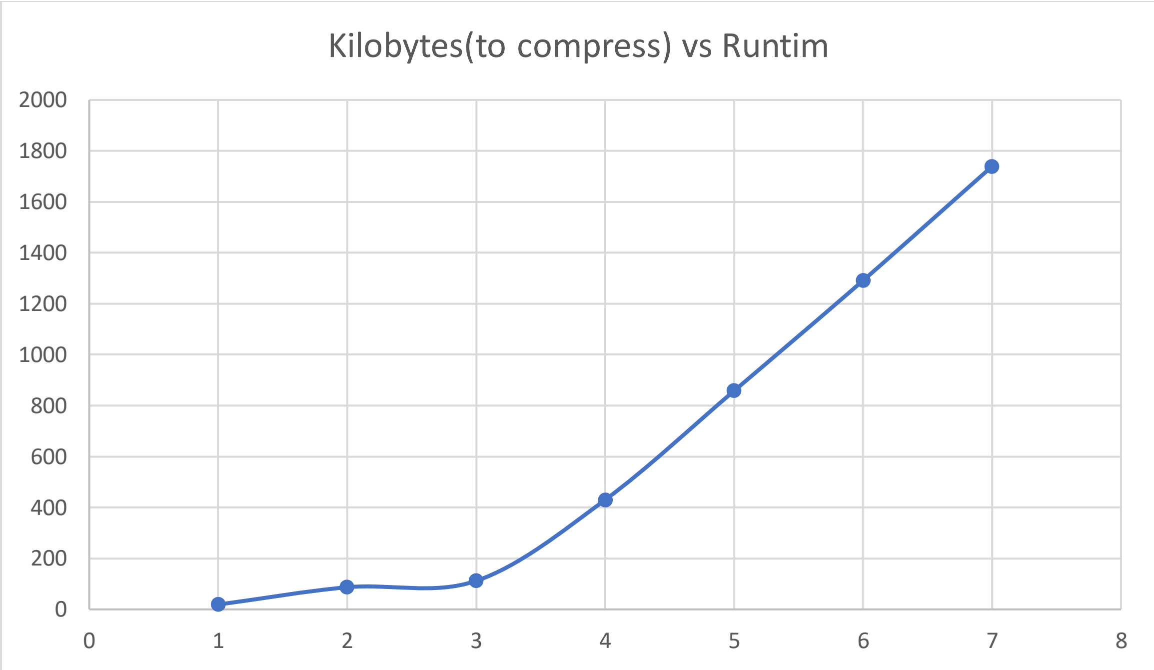
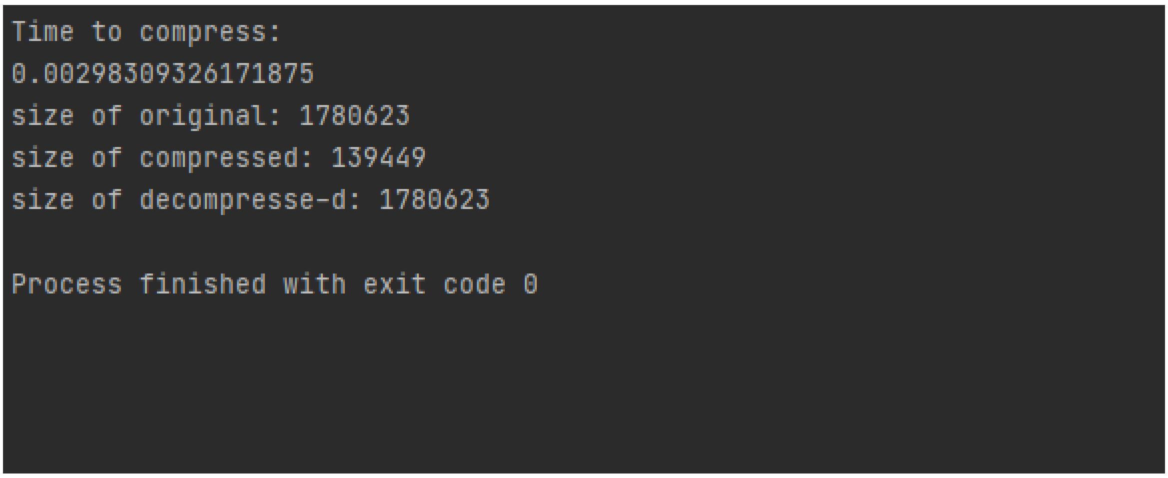


Figure23: Showing the result of compression on a runtime vs bytes graph

It can be seen from the results above that the compression algorithm's Big(O) value is equal to N for t>3 which corresponds to the algorithm that was used.

The Big(O) in this case as two different slopes at 100KB. From 0 to 100KB, the rate at which the time increases for an increase in file size is higher than the one after 100KB.



```
Time to compress:
0.00298309326171875
size of original: 1780623
size of compressed: 139449
size of decompresse-d: 1780623

Process finished with exit code 0
```

Figure24: Showing the size of compression and decompression at the output

Link to compression program:< https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/compress.py>

<u>Experiment 2: Ensuring that the compressed file is different than the encrypted one</u>

The following is a screenshot of one of the files that were encrypted

```
Geecjctgroih<&^<6(61*&_<&+l(61*&\<&46(41&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(62*&_<&+l(6?*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(77*&_<&+l(6>*&\<&46(26&k)uX4zti&^<6(66*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<+6(6>*&_<&+l(43*&\<&46(41&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(74*&_<&+l(62*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(60*&_<&+l(46*&\<&46(56&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(76*&_<&+l(42*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(6?*&_<&+l(61*&\<&46(52&k)uX4zti&^<6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(77*&_<&+l(6?*&\<&46(70&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(6>*&_<&+l(7>*&\<&46(46&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(76*&_<&+l(67*&\<&46(4?&k)uX4zti&^<6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(64*&_<&+l(6>*&\<&46(56&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(67*&_<&+l(73*&\<&46(40&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(77*&_<&+l(77*&\<&46(74&k)uX4zti&^<+6(66*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(60*&_<&+l(43*&\<&46(41&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(6?*&_<&+l(47*&\<&46(45&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(67*&_<&+l(7>*&\<&46(4>&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(65*&_<&+l(73*&\<&46(43&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(77*&_<&+l(7>*&\<&46(45&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(65*&_<&+l(53*&\<&46(44&k)uX4zti&^<6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(74*&_<&+l(54*&\<&46(72&k)uX4zti&^<6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(61*&_<&+l(46*&\<&46(73&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(67*&_<&+l(7>*&\<&46(70&k)uX4zti&^<+6(64*&_<&6(65*&\<&6(65&tgbu)u
Geecjctgroih<&^<+6(77*&_<&+l(73*&\<&46(7>&k)uX4zti&^<6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(62*&_<&+l(76*&\<&46(41&k)uX4zti&^<6(66*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(65*&_<&+l(60*&\<&46(4>&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(62*&_<&+l(47*&\<&46(72&k)uX4zti&^<6(64*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(62*&_<&+l(63*&\<&46(4>&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(6?*&_<&+l(6?*&\<&46(55&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(74*&\<&46(56&k)uX4zti&^<+6(66*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(60*&_<&+l(45*&\<&46(45&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(60*&_<&+l(46*&\<&46(75&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(67*&_<&+l(42*&\<&46(40&k)uX4zti&^<6(66*&_<&6(65*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(67*&_<&+l(45*&\<&46(56&k)uX4zti&^<6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(67*&_<&+l(76*&\<&46(4?&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(60*&_<&+l(43*&\<&46(47&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(4?*&\<&46(72&k)uX4zti&^<6(67*&_<&6(62*&\<&6(67&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(7?*&\<&46(45&k)uX4zti&^<+6(67*&_<&6(65*&\<&6(64&tgbu)u
Geecjctgroih<&^<6(42*&_<&+l(42*&\<&46(45&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<+6(65*&_<&+l(73*&\<&46(53&k)uX4zti&^<6(64*&_<&6(64*&\<&6(67&tgbu)u
Geecjctgroih<&^<6(61*&_<&+l(72*&\<&46(54&k)uX4zti&^<+6(67*&_<&6(64*&\<&6(65&tgbu)u
Geecjctgroih<&^<6(62*&_<&+l(73*&\<&46(43&k)uX4zti&^<+6(64*&_<&6(64*&\<&6(64&tgbu)u
Geecjctgroih<&^<+6(64*&_<&+l(70*&\<&46(4?&k)uX4zti&^<+6(66*&_<&6(65*&\<&6(67&tgbu)u
```

Figure25: Encrypted file content

Next, is a screenshot of the compressed files

Figure26: Output after compression

It can be seen that the compressed file characters are different than the encrypted file.

The output must match with the requirements:

- The compression algorithm as more than 25% of the original data, therefore this requirement is met

- The algorithm that was used was simple and fast and this can be confirmed by the average time to compress a file of 1.7KB is 0.0357s. Therefore, the requirement for the algorithm to be fast is respected

Link to compression program:< https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/compress.py>

Link to compression program output files< https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/Compress>

DECOMPRESSION

Experiment 1: Testing the performance of the decompression algorithm by measuring the runtime of the decompression program using different file sizes

The followings are the result of the decompression code used on the previously compressed data.

These are the following files after the compression in the previous step:

Bytes	Time elapsed	Bytes decompressed
1.944	0.0001031	19

7.152	0.0004966	87
16.081	0.0007506	112
33	0.0009615	430
66.346	0.0014882	858
105.351	0.0024809	1291
139.449	0.002979	1739

Table 8: Showing the result of decompression

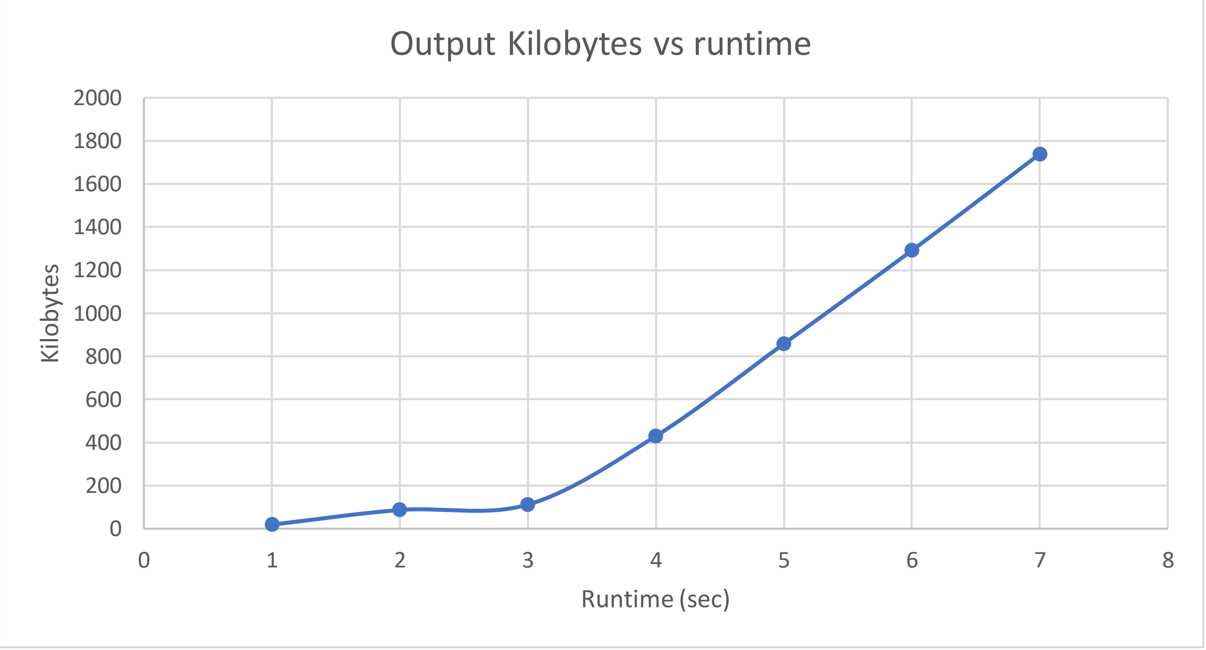


Figure27: Showing the result of the decompression on a runtime vs bytes graph

Similarly, to the compression data analysis, it can be seen from the results above that the decompression algorithm's Big(O) value is equal to N for t>3 which corresponds to the algorithm that was used.

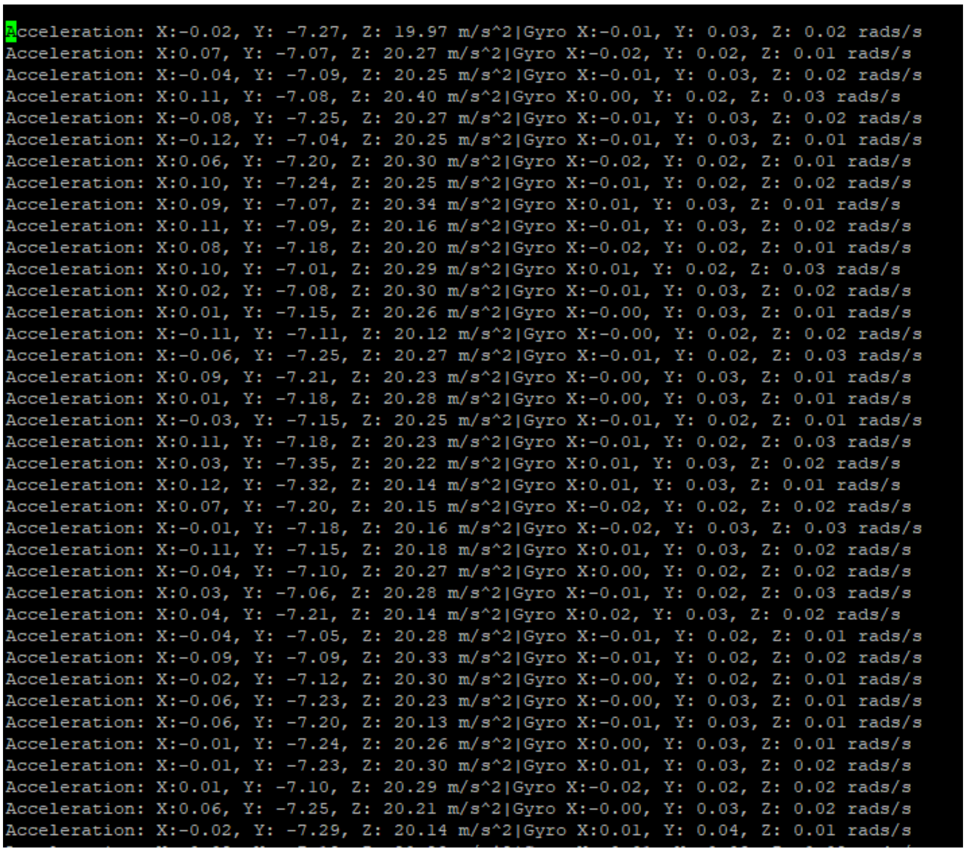
The Big(O) in this case as two different slopes at 100KB. From 0 to 100KB, the rate at which the time increases for an increase in file size is higher than the one after 100KB.

<u>Experiment2: Making sure that the decompressed file content looks similar to the encrypted one that was the input previously</u>

The following is a screenshot of on of the compressed file contents, the one used in the previous experiment:

Figure28: compressed file content

Next, is a screenshot of the output of the decompression program.



```
Acceleration: X:-0.02, Y: -7.27, Z: 19.97 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.07, Y: -7.07, Z: 20.27 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.04, Y: -7.09, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.11, Y: -7.08, Z: 20.40 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:-0.08, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.12, Y: -7.04, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.06, Y: -7.20, Z: 20.30 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.24, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.09, Y: -7.07, Z: 20.34 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.09, Z: 20.16 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.08, Y: -7.18, Z: 20.20 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.01, Z: 20.29 m/s^2|Gyro X:0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.02, Y: -7.08, Z: 20.30 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.15, Z: 20.26 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.11, Y: -7.11, Z: 20.12 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.06, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.09, Y: -7.21, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.01, Y: -7.18, Z: 20.28 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.03, Y: -7.15, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.18, Z: 20.23 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.03, Y: -7.35, Z: 20.22 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.12, Y: -7.32, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.07, Y: -7.20, Z: 20.15 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.01, Y: -7.18, Z: 20.16 m/s^2|Gyro X:-0.02, Y: 0.03, Z: 0.03 rads/s
Acceleration: X:-0.11, Y: -7.15, Z: 20.18 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.10, Z: 20.27 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.03, Y: -7.06, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.04, Y: -7.21, Z: 20.14 m/s^2|Gyro X:0.02, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.05, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.09, Y: -7.09, Z: 20.33 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.12, Z: 20.30 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.23, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.20, Z: 20.13 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.24, Z: 20.26 m/s^2|Gyro X:0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.23, Z: 20.30 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.10, Z: 20.29 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.06, Y: -7.25, Z: 20.21 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.29, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.04, Z: 0.01 rads/s
```

Figure29: Output after decompression

It is clear that the output after decompression (Figure12) is similar to the one in Figure8 therefore, the decompression algorithm works as it should.

The output must match with the requirement:

The algorithm that was used was simple and fast and this can be confirmed by the average time to decompress a 139.5B file in 0.002979 seconds. Therefore, the requirement for the algorithm to be fast is respected.

Link to decompressed program:< https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/decompress.py>

Link to decompression program output files:< https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/Decompress>

DECRYPTION

Experiment 1: Testing the performance of the decryption algorithm by measuring the runtime of the decryption program using different file sizes

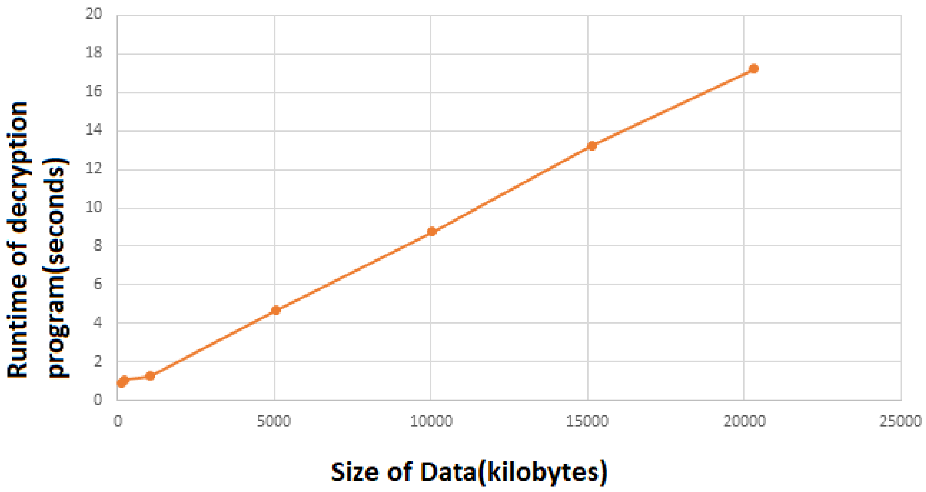


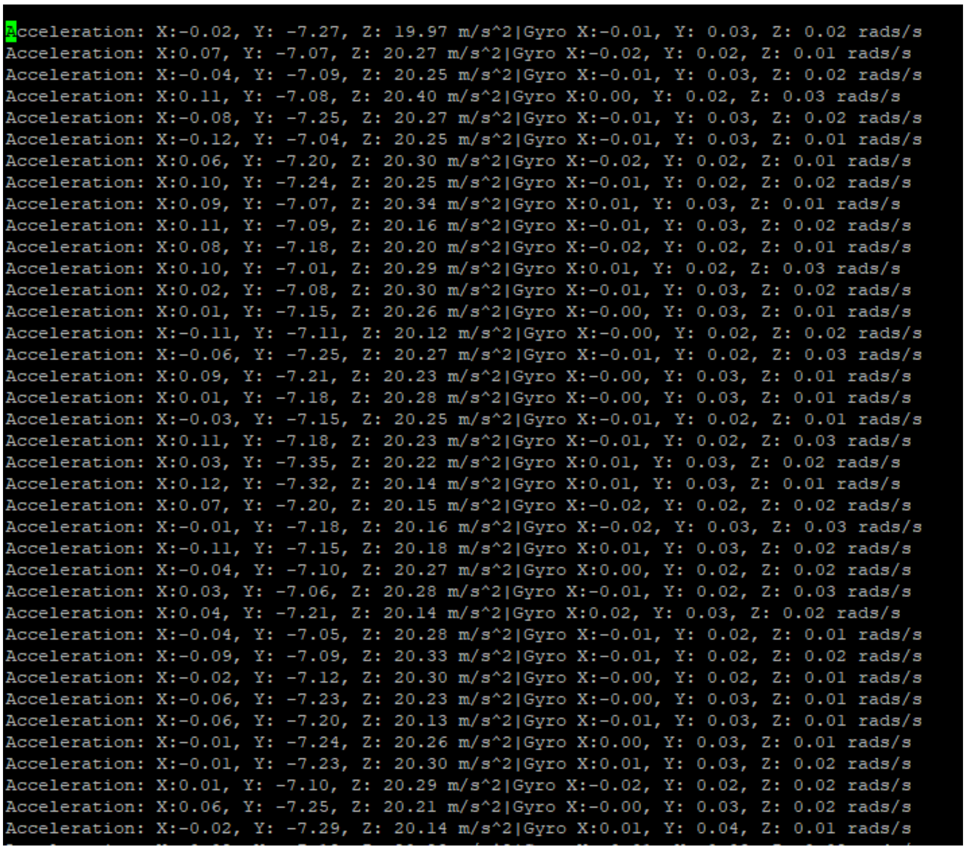
Figure30: Graph showing runtime of decryption algorithm for different file sizes

Like the encryption algorithm the decryption algorithm's Big(O) value is equal to N though from the values that can be seen above the decryption algorithm is slightly faster than the encryption algorithm. The best and worst-case order of complexity of the decryption algorithm is linear.

As previously stated, the Big(O) value of the Decryption algorithm is N which meets the required specification that the Big(O) of the Decrypted algorithm does not exceed N squared.

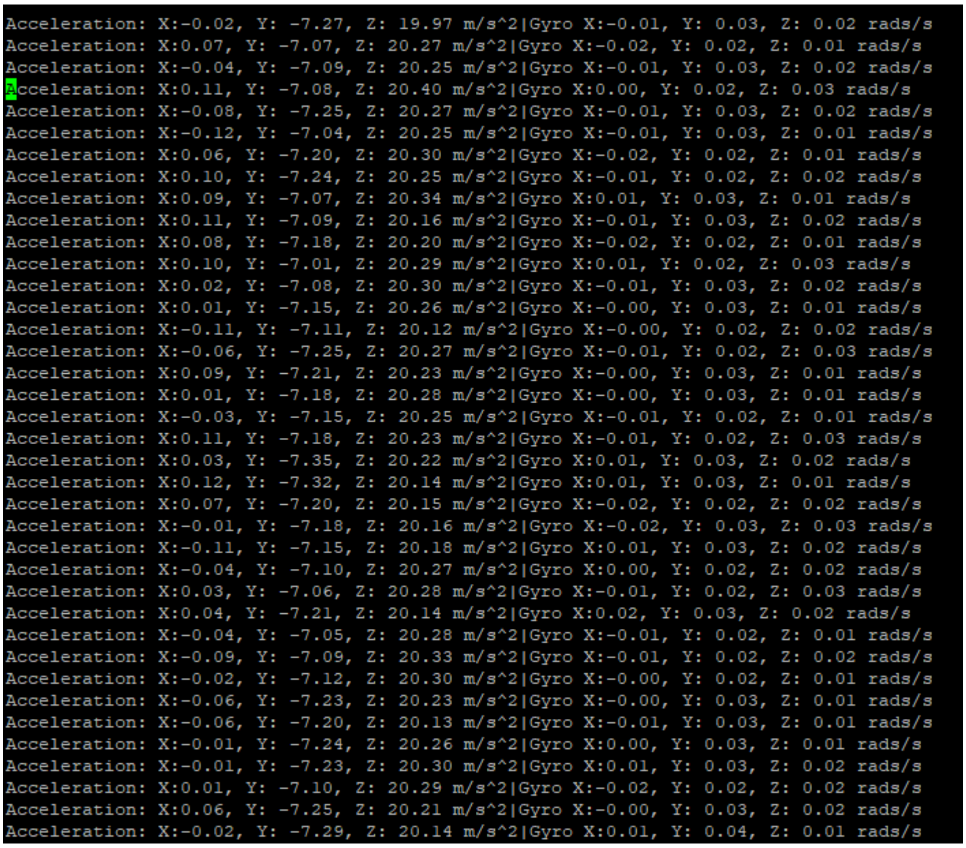
Link to Decryption Program:< https://github.com/Nathan-mboko/EEE3097S-Project/blob/main/decrypt.py >

EXPERIMENT 2: Ensuring that the decrypted file is the exact same as the raw data input file



```
Acceleration: X:-0.02, Y: -7.27, Z: 19.97 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.07, Y: -7.07, Z: 20.27 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.04, Y: -7.09, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.11, Y: -7.08, Z: 20.40 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:-0.08, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.12, Y: -7.04, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.06, Y: -7.20, Z: 20.30 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.24, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.09, Y: -7.07, Z: 20.34 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.09, Z: 20.16 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.08, Y: -7.18, Z: 20.20 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.01, Z: 20.29 m/s^2|Gyro X:0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.02, Y: -7.08, Z: 20.30 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.15, Z: 20.26 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.11, Y: -7.11, Z: 20.12 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.06, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.09, Y: -7.21, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.01, Y: -7.18, Z: 20.28 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.03, Y: -7.15, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.18, Z: 20.23 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.03, Y: -7.35, Z: 20.22 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.12, Y: -7.32, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.07, Y: -7.20, Z: 20.15 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.01, Y: -7.18, Z: 20.16 m/s^2|Gyro X:-0.02, Y: 0.03, Z: 0.03 rads/s
Acceleration: X:-0.11, Y: -7.15, Z: 20.18 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.10, Z: 20.27 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.03, Y: -7.06, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.04, Y: -7.21, Z: 20.14 m/s^2|Gyro X:0.02, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.05, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.09, Y: -7.09, Z: 20.33 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.12, Z: 20.30 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.23, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.20, Z: 20.13 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.24, Z: 20.26 m/s^2|Gyro X:0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.23, Z: 20.30 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.10, Z: 20.29 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.06, Y: -7.25, Z: 20.21 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.29, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.04, Z: 0.01 rads/s
```

Figure31: Sampled Input Data file



```
Acceleration: X:-0.02, Y: -7.27, Z: 19.97 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.07, Y: -7.07, Z: 20.27 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.04, Y: -7.09, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.11, Y: -7.08, Z: 20.40 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:-0.08, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.12, Y: -7.04, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.06, Y: -7.20, Z: 20.30 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.24, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.09, Y: -7.07, Z: 20.34 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.09, Z: 20.16 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.08, Y: -7.18, Z: 20.20 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.10, Y: -7.01, Z: 20.29 m/s^2|Gyro X:0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.02, Y: -7.08, Z: 20.30 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.15, Z: 20.26 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.11, Y: -7.11, Z: 20.12 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.06, Y: -7.25, Z: 20.27 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.09, Y: -7.21, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.01, Y: -7.18, Z: 20.28 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.03, Y: -7.15, Z: 20.25 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:0.11, Y: -7.18, Z: 20.23 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.03, Y: -7.35, Z: 20.22 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.12, Y: -7.32, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:0.07, Y: -7.20, Z: 20.15 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.01, Y: -7.18, Z: 20.16 m/s^2|Gyro X:-0.02, Y: 0.03, Z: 0.03 rads/s
Acceleration: X:-0.11, Y: -7.15, Z: 20.18 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.10, Z: 20.27 m/s^2|Gyro X:0.00, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.03, Y: -7.06, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.03 rads/s
Acceleration: X:0.04, Y: -7.21, Z: 20.14 m/s^2|Gyro X:0.02, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.04, Y: -7.05, Z: 20.28 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.09, Y: -7.09, Z: 20.33 m/s^2|Gyro X:-0.01, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.12, Z: 20.30 m/s^2|Gyro X:-0.00, Y: 0.02, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.23, Z: 20.23 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.06, Y: -7.20, Z: 20.13 m/s^2|Gyro X:-0.01, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.24, Z: 20.26 m/s^2|Gyro X:0.00, Y: 0.03, Z: 0.01 rads/s
Acceleration: X:-0.01, Y: -7.23, Z: 20.30 m/s^2|Gyro X:0.01, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:0.01, Y: -7.10, Z: 20.29 m/s^2|Gyro X:-0.02, Y: 0.02, Z: 0.02 rads/s
Acceleration: X:0.06, Y: -7.25, Z: 20.21 m/s^2|Gyro X:-0.00, Y: 0.03, Z: 0.02 rads/s
Acceleration: X:-0.02, Y: -7.29, Z: 20.14 m/s^2|Gyro X:0.01, Y: 0.04, Z: 0.01 rads/s
```

Figure32: Output file after Decryption

The output file of the decryption algorithm mirrors the Raw Data input file exactly this shows that all the data in the Raw data input file of the system is found in the decrypted output file of the system.

Link to Descripted files folder: https://github.com/Nathan-mboko/EEE3097S-Project/tree/main/Decrypt

6. Consolidation of ATPs and Future Plan

6.1. ATPs from previous section

Compare the compressed file size to the file size of the Input data and ensure that the Compressed file is less than half the size of the Input Data file
Inspect the contents of the encrypted file to ensure it is not in a readable format
Run a program which compares the data in the output file of the system to the data present in the input data file
Observe how the Compression and Decompression Algorithms performs when large input data files are put into the algorithm
Observe how the Encryption and Decryption Algorithms performs when large input data files are put into the algorithm

Table9: Table showing recreate the ATPs from your previous section

6.2. ATPs requirements

Specification	ATP	Has The ATP been met.
Compression ratio should be above 25%	Compare the compressed file size to the file size of the Input data and ensure that the Compressed file is less than half the size of the Input Data file	Yes, In experiment 1 of Compression section, the average compression ratio is 60% which is higher than 25%.
The encrypted data should not be in a readable format	Inspect the contents of the encrypted file to ensure it is not in a readable format	Yes, in experiment 2 of the Encryption section it can be seen that the encrypted files are in an unreadable format.
The Decrypted data file should contain all the data from the Input Data file	Run a program which compares the data in the output file of the system to the data present in the input data file	Yes, in experiment 2 of System result section the content in the system output file and the Input Data file were identical
The Big O notation of the compression and decompression algorithm should at most be n squared.	Observe how fast the Compression and Decompression Algorithms performs when large input data files are put into the algorithm	Yes, in experiment 1 of the Compression and Decompression result sections the two algorithms beave linearly. The Big(O) of the algorithms equal N which is less than N squared.
The Big O notation of the encryption and decryption algorithm should at most be n squared.	Observe how fast the Encryption and Decryption Algorithms performs when large input data files are put into the algorithm	Yes, in experiment 1 of the Encryption and Decryption result sections it can be seen that the two algorithms beave linearly. The Big(O) of the algorithms equal N which is less than N squared

Table10: Showing each ATPs and if it has been met the design

It can be observed from the table that all the specification and ATP have been met therefore, a table with specification changed will be the same as *Table10.*

6.4. Necessary set of work for a buoy team to use the project in the future

For the project to work fine, the team taking the project should:

- Ensure that the IMU is placed in a waterproof environment to avoid device malfunctioning
- The IMU should be well attached to the buoy to get accurate readings
- Run a test on the IMU to confirm if it fits the specifications that their work require

7. Conclusion

The task of the Project was to design a system which would encrypt, compress, decrypt and decompress data received from an IMU on a Sharc Buoy.

The Python XOR Algorithm was selected to implement the encryption and decryption aspects of the project. The Python Zlib library was used for compression and decompression. Experiments were ran to determine the speed and efficiency of each of the systems processes. Further experimentation was done to ensure the that the output file of the system was in the proper format and contains the proper content.

The Results of the experiment met each of the requirements and specifications that were stated and passed the Acceptance Test Procedures. Each of the system processes (Encryption, Decryption, Compression and Decompression) have a linear time complexity which is not the ideal in terms of the systems speed, but it meets the requirement of being faster than a system with a time complexity of N squared. Another crucial aspect of the system was to ensure that all the data sampled from the ICM20649 sensors has to be found in the output file of the system, this criterion was met by the system ensuring zero data loss.

From the results of the simulation-based experiments and the hardware-based experiments we can conclude that the designed system is a viable solution to the initial task presented as the designed system has met the requirements and specification of the task. Further improvements can be made to the systems design particularly the systems Encryption and Decryption algorithms which could be made faster and more efficient.

8. REFERENCES

1. Techie Delight (2021) Run Length Encoding (RLE) Data Compression Algorithm[online]. Available from:<https://www.techiedelight.com/run-length-encoding-rle-data-compression-algorithm/ >.

2. GeeksforGeeks (2021) Huffman Coding[online]. Available from:<https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/ >

3. Electrical and Computer Engineering UCSB (2017) Block Transform Coding [online]. Available from: <https://web.ece.ucsb.edu/~manj/ece178-Fall2009/e178-L14.ppt.pdf >

4. Tutorialspoint.com, 2021. [Online]. Available: https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_xor_process.htm. [Accessed: 05- Sep- 2021]

5. Tutorialspoint.com, 2021. [Online]. Available: https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_transposition_cipher.htm. [Accessed: 05- Sep- 2021]

6. "The importance of IMU Motion Sensors - CEVA's Experts blog," *CEVA's Experts blog*, Nov. 15, 2018. https://www.ceva-dsp.com/ourblog/what-is-an-imu-sensor/ (accessed Oct. 24, 2021).

7. 2021. [Online]. Available: https://www.mathworks.com/help/simulink/gs/create-a-simplemodel.html. [Accessed: 28- Oct- 2021].

9. D. G. Gryazin, L. P. Starosel'tsev, O. O. Belova, and A. N. Dzyuba, "Inertial measurement unit of waverider buoy. Development and test results," *Gyroscopy and Navigation*, vol. 7, no. 3, pp. 239–246, Jul. 2016, doi: 10.1134/s2075108716030056.

10. Y.-L. Huang, C.-Y. Kuo, C.-H. Shih, L.-C. Lin, K. Chiang, and K.-C. Cheng, "MONITORING HIGH-FREQUENCY OCEAN SIGNALS USING LOW-COST GNSS/IMU BUOYS," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B8, pp. 1127–1134, Jun. 2016, doi: 10.5194/isprs-archives-xli-b8-1127-2016.

11. "Code Faster with Line-of-Code Completions, Cloudless Processing," *Kite.com*, 2021. https://www.kite.com/python/answers/how-to-write-bytes-to-a-file-in-python#:~:text=Use%20open()%20and%20file,close%20the%20file%20using%20file. (accessed Oct. 31, 2021).