

UNIVERSITY OF CAPE TOWN



EEE3097S Final Design Project

Group 24: Bertha Molai

Umutesa Munyurangabo

October 2021

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.

2. I have used the ...**IEEE**.....convention for citation and referencing.

3. The text of this essay/project/report is my own work, using my own words (except where attributed to others). This essay/project/report uses the work of the group (list names). ...
mlxbat001...mnyumu002..... I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.

Each group member has undertaken to submit his/her own essay/project/report and acknowledge the work of other group members.

I acknowledge that by copying someone else's assignment or essay, or a part of it, is wrong, and declare that this essay/report/project is my own work, and is based on the work of the group.

Table of Contents

Declaration	2
Table of Contents	3
1. Admin Documents	6
2.Introduction	9
2.1 Background to Investigation	9
2.2 Project Statement	9
2.2.1 Project initiations and Scope	10
2.2.2 Limitations and Constraints	10
2.2.3 Assumptions	10
3. Requirements Analysis	10
3.1. Analysis of Requirements	10
3.2. Overall System Requirements	11
Operational Requirements	11
Electronic Requirements	12
Functional Requirements	12
3.3. Subsystem Requirements	14
3.3.1. Compression Block	14
3.3.2. Encryption Block	14
3.4. Specifications	15
3.4.1. Overall System Block	15
3.4.2. Compression Block	16
3.4.3. Encryption Block	17
3.5. Acceptance Test Procedures (ATPs)	18
3.5.1. Metrics to Observe	18
3.5.2. Overview of Experiment Design for Testing	18
3.5.3. Acceptance Performance Definitions	19
4. Paper Design	20
4.1. Specifications & Requirements	20
4.2. Comparisons of Available Compressions and Encryptions Algorithms	20
4.2.1. Compression	20
4.2.2. Encryption	21
Algorithm 1: RSA Algorithm	24
Algorithm 2: Encoding and Decoding using Base64 Strings	24
Algorithm 3: Cryptocode	24
Algorithm 4: Cryptography Fernet	24

Algorithm 5: Hash Function	25
4.3. Subsystems Design	25
Inter – Subsystem and Inter Sub-subsystems Interactions	26
4.4. Feasibility	27
4.5. Bottlenecks	27
5. Validation using Old Data	28
Data Used	29
5.1. Experiment Setup	34
5.1.1. Overall System functionality	34
5.1.2. Experiment Design: Compression	34
Test 1: Compression Ratio	34
Test 2: Speed of Algorithm	34
Test 3: Data integrity	34
5.1.3. Experiment Design: Encryption	35
Test 1: Data Check	35
Test 2: Algorithm Speed Complexity	35
Test 3: Correctness of Data	35
Test 4: Data Loss	35
Test 5: Strength of Algorithm	36
Test 6: Size of Encrypted and decrypted data	36
5.2. Results	36
5.2.1. Overall System functionality	36
5.2.2. Results: Compression	37
<i>Output of test results</i>	37
5.2.3. Results: Encryption	38
Test 1 : Data Check	38
Test 2: Algorithm Speed Complexity	39
Test 3: Correctness of Data	39
Test 4: Data Loss	40
Test 5: Strength of Algorithm	41
Test 6 : Size of Encrypted and decrypted data	42
6. Validation using a different IMU	43
6.1. Simulated model vs Hardware model:	52
Experiment Setup	55
Overall system functionality: Raspberry pi including IMU Sense Hat	55
Experiment Design: Compression	57
Experiment Design: Encryption	58

Test 1: Data Check	58
Test 2: Algorithm Speed Complexity	58
Test 3: Correctness of Data	58
Test 4: Data Loss	59
Test 5 : Strength of Algorithm	59
Test 6 : Size of Encrypted and decrypted data	59
Results	60
Results: Overall system functionality	60
Test 1 : Overall Execution Time of System	60
Test 2: Measure the Power of the System	61
Test 3: Measure the Temperature and Pressure of Both Systems (Before and After Execution)	61
Test 4: "Stress" testing Raspberry pi & IMU Sense Hat	62
Results: Compression	62
Results: Encryption	64
6. Consolidation of ATPs and Future Plan	70
7. Conclusion	72
8. References	73

1.Admin Documents

Contribution:

Name of Student	Contribution	Page Number
mlxbat001	Inter-subsystem interactions & putting system together	pg 25 - 27
mlxbat001	Compression Algorithm sections	pg 14, 16, 19, 67
mlxbat001	ATP Tests	Pg 17 –19, 69
mlxbat001	Old Data Analysis	Pg 27 -41
mlxbat001	Specifications, UML & Requirements Analyse	Pg 60 – 62, pg 10-16
mnyumu002	Admin A&=	Pg 3
mnyumu002	Introduction	Pg 9 -12
mnyumu002	Encryption Algorithm sections	Pg 9 –10, 19, 20
mnyumu002	ATP Tests	Pg 17 –19, 69
Mnyumu002, m	User Requirements & Specifications	Pg 10 -16
mnyumu002	New Data Analysis	Pg 41 - 63

Project Management Tool:

A snapshot of our project management page is shown below:

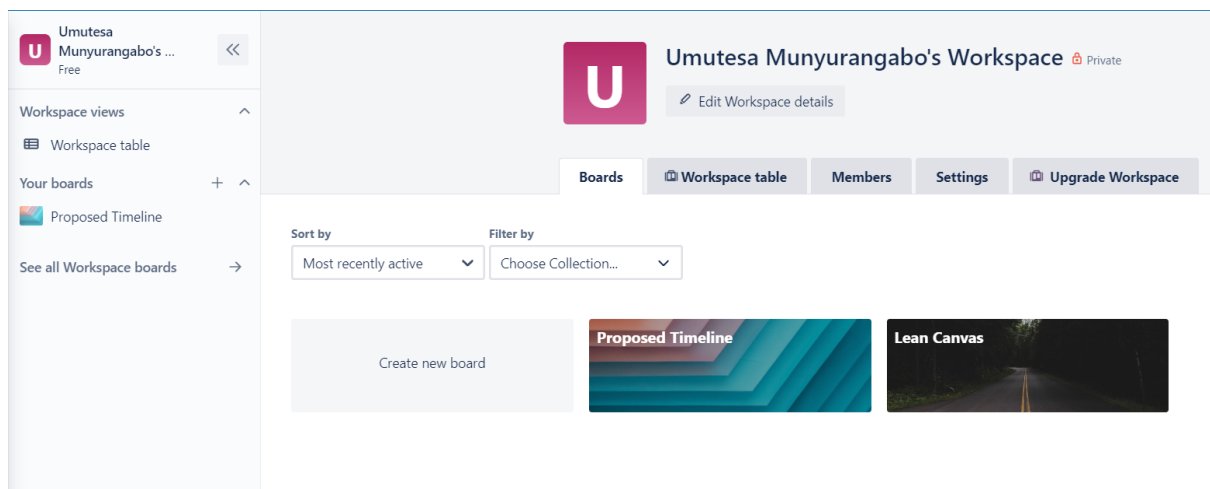


Figure 1: Trello Workspace

This page has been used to track the progress throughout the project, as well as plan the work ahead. This page has also aided in dividing the work up into manageable chunks between each member.

Development Timeline:

An overview of the development timeline for the first month of the project is shown below:

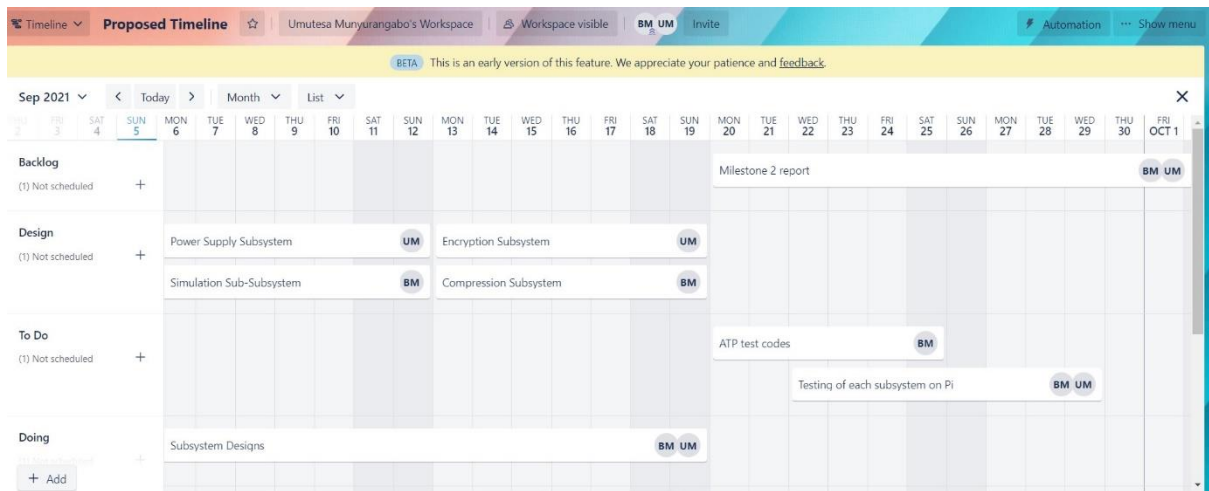


Figure 1: Dev Timeline in calendar form

A closer view of our timeline to date is shown in the image below. Each task is divided into subcategories (To-Do, Doing, testing etc.) and the progress on each task is tracked this way.

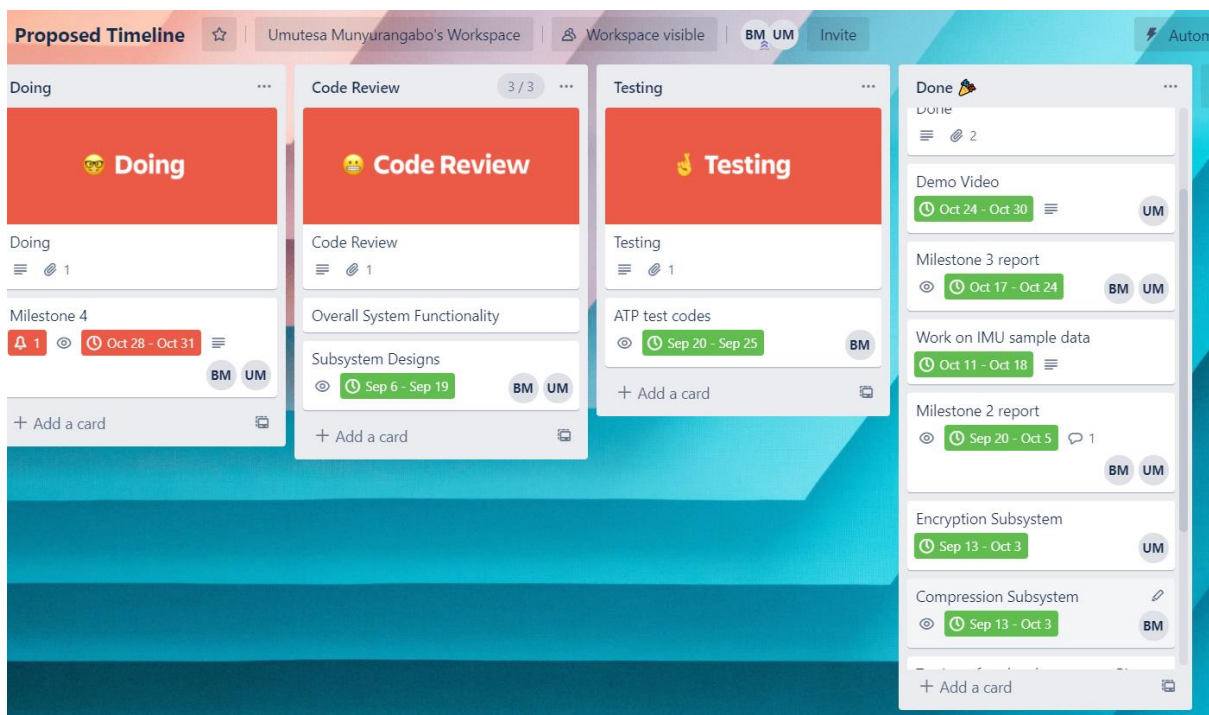


Figure 2: Current progress timeline

The Overall System code is under code review to fix a few bugs and optimise it. Apart from this, everything else is up to date. As can be seen in Figure 2, there is no Power Supply Subsystem accounted for as this was removed during the refinement process of the project. Further, all the due tasks have been completed.

In the overall project, there were a few delays due to the inclusion of development using a Raspberry Pi Sense Hat (B). The dates were adjusted accordingly and instead of finishing the project on the 29th October 2021 as intended, the final completion is the 31st October 2021.

In this final milestone, we continued working with the Sense Hat B, and implemented further refinements into our design. The aim of this milestone was compiling all the data gathered over the past milestones into a final document. Further testing was also done on our system to ensure correctness.

GitHub Link:

The git repository for our project can be found here:

<https://github.com/BerthaMolai/EEE3097S>

Trello Link:

The link to our project management tool platform:

<https://trello.com/umutesamunyurangabosworkspace/home>

2.Introduction

2.1 Background to Investigation

Global warming has been a growing issue in today's world. One such problem is clearly seen in the Antarctic where the polar cyclones and sea ice have become alarming. In 2016, the seasonal melting of ice took place over a much shorter time than usual [1]. These Antarctic conditions directly affect South African Weather as well, and are thus of greater interest. Although many teams have been doing research in the Southern Ocean, there are many constraints and setbacks due to the weather conditions so the information on the environmental conditions is limited. The Southern Hemisphere Antarctic Research Collaboration has recently developed their own buoy (SHARC Buoy) and research is currently underway. This buoy contains many sensors, one of which is used to measure the wave dynamics and other information about the ice in the environment such as the size and motion.

2.2 Project Statement

This project is an extension of the buoy model design which would be installed on the pancake ice in the Southern Ocean in order to gather environment data such as positions, force etc. One of the main major sub- systems of the design is the IMU sensor. An IMU is a specific type of sensor that measures force, magnetic field, position and angular rotation. The IMU is crucial, as it is able to provide information about the ice including the wave dynamics of the environment. Hence, the Project aims to design a system that is able to transmitted the data (i.e., compress and encrypt) using a low - cost alternative solution. Initially, one of the solutions was to use an Iridium, however this approach was too costly. By Identifying and engaging with different techniques, this project aims to design an arm base digital IP system that is able to transmit data at a low-cost while still maintaining high performance and accuracy for the ice buoys in the Southern Ocean in Antarctica.

2.2.1 Project initiations and Scope

The project was initiated by Prof Mishra, for the third Year ECE design Project (EEE3097S). focuses on the transmitting of the information using IMU sensors. The outcome would be a software implementation of the data transmitted using a Raspberry Pi Zero.

2.2.2 Limitations and Constraints

The biggest limitation of the project is the time period allocated to the project. This project starts in August 2021 and needs to be completed by the end of October 2021. This means 3 that the project needs to be completed within 3 months. Thus, good management skills and planning will be required to ensure that the project is completed on time. The IMU sensors are constrained, as these will not be available or provided during the period of the project. The lack of resource availability can increase the difficulty of the project as one needs to find alternative methods to access the data without the real hardware. The cost of the project has been constrained, since the main objective of this project is to implement a low-cost solution. Hence, the project is restricted in budget and spending extra money on resources is prohibited.

2.2.3 Assumptions

The following assumptions were made regarding the project: Although it would be extremely impossible for project designers to travel to Antarctic to collect real - time data, The data that were collected, for instance in the Labs is assumed be equivalent to the data that one would have collected in Antarctica, and thus can be used as real time data available in the Southern Ocean.

3. Requirements Analysis

3.1. Analysis of Requirements

As highlighted in the project statement, the digital IP will be used on a SHARC Buoy to collect data that will aid in investigating the weather phenomena in the Antarctic.

At a high level, the designed IP should use data files collected from an IMU sensor and compress and encrypt the files for them to be later transmitted. This process should be executed with the least amount of loss in information possible.

The design requires that the processing of this data consume as little power as possible as the raspberry pi will be running on a finite source of power. The speed of the system will also determine how much power is consumed by the overall execution of the system. In addition, the storage space on the device is also limited and thus compression of the data files should be maximised without compromising the integrity of the information transmitted. At least 25% of the lower Fourier coefficients should be retrieved for the data to be useful. The

system will need to satisfy this condition with noise added to the input as it is highly likely that noise will be picked up during transmission.

These high level and design requirements are further explored in more detail to derive the system requirements that are detailed in the next two subsections.

3.2. Overall System Requirements

Operational Requirements

User Requirement ID	Requirements	Description
UR001	Low-cost solution	The system must not exceed the budget provided; hence it should only make use of the resources provided, i.e., IMU sense hat & Raspberry pi
UR002	Minimal Resources Solution.	The system should not exceed the power, hence become too hot
UR003	Security Protection of Data	The data needs to be protected, and should allow users with valid access to gain access to the data
UR004	Transmit Data Remotely	The system should be able to compress and encrypt data remotely
UR005	No exchange of Sensitive Data	The system should not allow any leakage of sensitive information.
UR006	Robust Reliability	The system should be strong and withstand any possibilities of hackers.

Table 1: Operational Requirements

Electronic Requirements

User Requirement ID	Requirements	Description
UR007	System will run off a portable power source with a limited charging capability to survive for at least one month	The system needs to operate without a power source.
UR008	Minimal amount of computation Power.	The system should not exceed the power, hence become too hot
UR009	Minimal amount of computation heat & temperature during simulation	The system should not overheat, because this could damage the IMU sensor, as well as the raspberry pi.
UR010	Back - up storage supply	Additional Back up supply should be supplied.

Table 2: Electronic Requirements

Functional Requirements

User Requirement ID	Requirements	Description
UR011	Extract at least 25% of the Compression data	The system needs to decompress, decrypt and recover at least 25 % of the lower Fourier coefficients
UR012	Optimization of the data i.e., how fast is the algorithm	The system needs to compress and encrypt data at a reasonable speed, i.e., no longer than 60 seconds
UR013	Accuracy and consistency in data after encryption and compression of system	The data that is compressed and encrypted in the system design needs to be accurate

		and hence, no data must be lost.
UR014	Contain sufficient storage for the data	The IMU and raspberry pi needs sufficient storage space for data storage.
UR015	Encryption allowed with no key or public key	The system should allow encryption to occur openly with a public key access to all users.
UR016	Decryption denied if invalid Key provided	The system needs to decrypt the data only if user has access to the private key.
UR017	Using a unique encrypted code to encrypt the data	The data needs to be encrypted using a unique encryption algorithm
UR018	Sampling and Compression are constant in time	The sampling and compression time of the data, hence needs to be consistent.
UR019	Size of compressed file less than specified max size per file	The size of compressed data, needs to at its minimum.
UR020	Compression ratio of file	The compression ratio needs to be sufficient and not too large.
UR021	Decompressed data file produces same results as original uncompressed file	Consistency in data supplied.
UR022	The size of the decrypted file needs to be equivalent to the original text	Consistency in Size of data
UR023	Encryption strength sufficient to protect the information from disclosure	Encryption data needs to enclosed with a private key.

Table 3: Functional Requirements

3.3. Subsystem Requirements

Further dividing the system requirements into smaller categories specific to the subsystems, the requirements for each of these are derived from the scope of the project as well as the high-level requirements stated in the project brief.

3.3.1. Compression Block

The table below shows the requirements for the compression block.

Requirement ID	Requirements	Description
CR001	Minimal sampling and compression time	The sampling rate, needs to not exceed the minimum.
CR002	Substantial decrease in size of file	The decompressed file needs to be smaller than compressed file.
CR003	Sufficient representation of data from data sampled	The data needs to a representation of the original data.
CR004	Minimal loss of data	No data must be lost.
CR005	Compression ratio of file	Minimum ratio size that is compressed

Table 4: Compression Block Requirements

3.3.2. Encryption Block

The table below shows the requirements for the encryption block.

Requirement ID	Requirements	Description
ER001	Security & Automatic Data Protection	The data needs to be protected.

ER002	Effective encryption to protect the data	The data needs to be protected, and should allow users with valid access to gain access to the data
ER003	Password Protection and Zipping	Use of private key to decrypt and decompress the data.
ER004	Access Remotely	The system should be able to compress and encrypt data remotely
ER005	Authentication Verification	The system needs to decrypt the data only if user has access to the private key.
ER006	Integrity of data is maintained	The data needs to be consistent and correct
ER007	Fast and Easy encryption of data	The system needs to compress and encrypt data at a reasonable speed, i.e., no longer than 60 seconds

Table 5: Encryption Block Requirements

3.4. Specifications

After analysing the requirements for the entire system, a list of specifications can be developed to aid in the design of the system. These are divided into two subcategories: a list specific to the compression block, and one specific to the encryption block.

3.4.1. Overall System Block

Spec ID	Related Req ID	Specifications

OS001	UR001	No additional resources were brought and only the resources supplied by UCT were used. Hence cost of project is R0.00
OS002	UR002	The System is designed using only two resources: IMU Sensors & Raspberry pi
OS003	UR003	A strong encryption algorithm should be implemented, so that data is protected, I.e. Hash Function
OS004	UR004	Data needs to encrypted remotely.
OS005	UR005	No data should be shared without permission or access to private key.
OS006	UR007	Power supplied to IMU and Raspberry pi, i.e., 3.3 to 5 volts
OS007	UR008	Additional Data supplied by PC port, i.e. 12 to 24 volts
OS008	UR009	A built in “cooling “system implemented in design when it gets too hot, or above the optimal average temperature.
OS009	UR010	Power of system should not exceed 10 Watts.

Table 6: Table of specifications for the overall system

3.4.2. Compression Block

The table below shows the specifications derived from the specific requirements stated in Section 3.3.1 which details the requirements for the Compression block.

Spec ID	Related Req ID	Specifications
CS001	CR001	Compression time should not exceed standard gzip compression time for the same data size
CS002	CR001	Sampling rate should be twice as fast as compression rate

CS003	CR003	Sampled data should represent at least 25% of the lower Fourier coefficients in the data
CS004	CR002	compression ratio(must be at least 10)
CS005	CR004	Gzip CRC-32 CheckSum to ensure data integrity
CS006	CR001	Efficient sampling and compression algorithm implemented in C for less CPU time when gathering data

Table 7: Table of specifications for the Compression Subsystem

3.4.3. Encryption Block

The table below shows the specifications derived from the specific requirements stated in Section 3.3.2 which details the requirements for the Encryption block.

Spec ID	Related Req ID	Specifications
SE001	ER001	Low Percentage Data loss
SE002	ER002	Faster timing Algorithm, I.e less than 60 seconds
SE003	ER003	Using a certain unique encrypted code, ie hash function
SE004	ER004	Size of encrypted file needs to be larger than original data file
SE005	ER005	Size of encrypted file needs to be equivalent to original data file
SE006	ER006	No decryption allowed without the valid private key

Table 8: Table of specifications for the Encryption Subsystem

3.5. Acceptance Test Procedures (ATPs)

3.5.1. Metrics to Observe

Specific figures of merit were measured to test the functionality of the system and validate it. In particular, the following metrics were paid attention to:

- Sampling time of simulated data
- Compression time
- Encryption time
- Size of compressed and encrypted data
- Contents of output files
 - Accuracy
 - Percentage of Fourier coefficients contained
- Encryption keys

3.5.2. Overview of Experiment Design for Testing

For files of different sample sizes, the subsystem is tested via a program for the following parameters:

- Compression rate in MB/s
- Compression ratio (original size/ compressed file size)
- Integrity (compressed file will be decompressed and its contents compared to those of original file)
- Accuracy of the encryption algorithm, hence an algorithm is used to compare the output of the encryption subsystem as well against the original data.
- Percentage of data that is loss in the decryption algorithm, this is to ensure that no data is lost in the encryption process.
- Security protection of the data in the encryption algorithm
- Uniqueness of encryption algorithm code

For each of these subsystems, the time taken to run the algorithm was measured and compared.

3.5.3. Acceptance Performance Definitions

The ATP tests will only be met if the following figures of merit are measured successfully, as shown in Table 9.

ATP Number	Related Spec ID	Acceptance Test
1	CS001, CS002, CS006	Total sampling and compression time are consistent in proportion to the original dataset size and does not exceed the max time specified (no more than 1minute)
2	CS003	File of sampled data before encryption compared against original data and has at least 25% of its original lower Fourier coefficients
3	CS003	Size of compressed file less than specified max size per file depending on the number of data files on Raspberry Pi 0 SD card.
4	CS004	Compression ratio of file greater than 2
5	CS005	After gzip CRC-32 checksum, decompressed data file produces same results as original uncompressed file
6	SE001	No data loss after encrypted file has been decrypted
7	SE002	Timing the algorithm, and obtaining a faster time complexity algorithm to encrypt & decrypt data needs to be faster than the average time complexity of a simple encryption algorithm, i.e 60 seconds
8	SE003	The encrypted text needs to be unique & different the original data
9	SE004	The size of the decrypted file needs to be equivalent to the original text, hence equal to 8874 KB.
10	SE005	The size of the encrypted text needs to larger than the original IMU data, hence larger than 8878 KB
11	SE006	A valid message needs to be outputted when the user tries to decrypt the data with an invalid key.

4. Paper Design

4.1. Specifications & Requirements

Thorough analysis of the user requirements and specifications was carried out. The results of this research are shown in Sections 3.2: Overall System Requirements , 3.3: Subsystem Requirements and 3.4: Table 5: Encryption Block Requirements

Specifications.

4.2. Comparisons of Available Compressions and Encryptions Algorithms

In order to optimise the design, the different encryption and compression algorithms were researched and compared. The features of these algorithms were examined closely to determine which algorithm would work best for the system.

4.2.1. Compression

There are several types of data compression standards, the most common one being Deflate [2]. This is the core algorithm used for gzip, zlib and Zip which are the most common types of compressions.

The algorithms for data compression implementations are compared with:

- ❖ Compression speed
- ❖ Decompression speed, and
- ❖ Compression ratio

There is normally a trade off with ratio and compression speed, which means when choosing which type of compression to use, the most favourable outcome may not be the smallest data size, or the compression the speed but rather has to include a compromise on both the time taken and size for the best results in our design.

The comparison of these three types of compression is effectively illustrated in the image below:

Comparison

	zlib	GZIP	ZIP
Headers	0x78(01/9C/DA)	1F8B	504B0304
Compression format	DEFLATE	DEFLATE	DEFLATE
Checksum	None	CRC-32	CRC-32
Lossless?	Yes	Yes	Yes
Data	Stream / single file	Stream / single file	Archive files and directories

Figure 3: Table of comparison of gzip, zlib and zip [3]

As shown above, the main differences between the three are the headers, and the fact that zlib does not have an integrity check at the end of the compression (Checksum). Further, zlib and gzip algorithms only compress one file at a time.

For this design, gzip will be used as it offers an easier way to check for data integrity. Further, it can be used in conjunction with tar.xz to compile multiple files, and the size is normally smaller than zip.

4.2.2. Encryption

The most common techniques of encryptions are illustrated in the figure below:



Figure 4: Types of Encryptions

- Symmetrical Encryptions: This is when the key used to encode and decode are identical
- Asymmetric Encryptions: The encode and decode keys are different, but are linked mathematically.

In modern days, the data is represented in a string of bits. Hence, the encryption algorithm needs to convert these bits into readable textss.

The encryption process can also be explained by means of illustration.

How Encryption Works

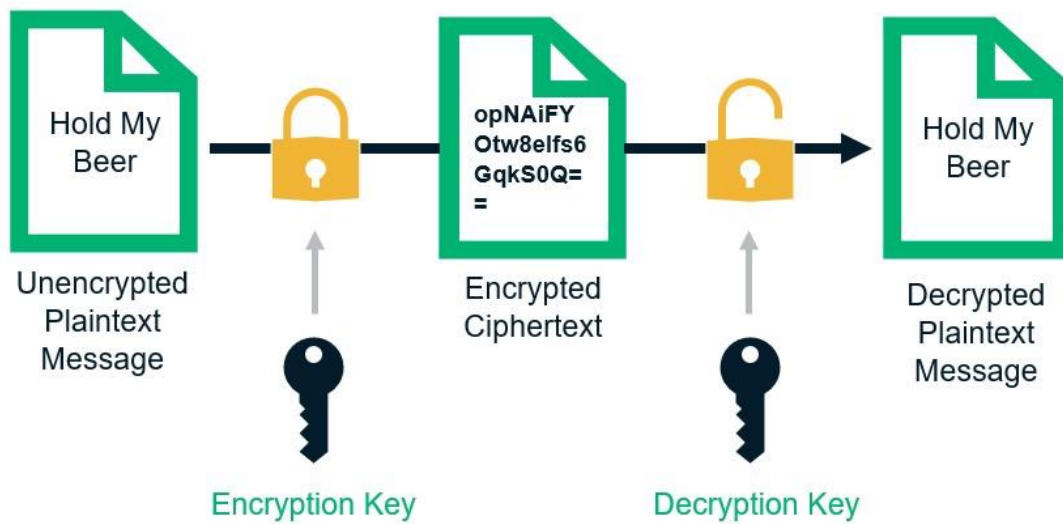


Figure 5: Encryption Process [4]

As shown in figure 5, the encryption process will receive an input from the users, i.e., plain text, encrypt the data using a specific key as an Encrypted ciphertext. Thus, in order to decrypt the encrypted text, the users need the decrypted key, i.e private key to unlock the code.

Different Encryption Algorithms

Encryption involves the process of cryptography; this is the process where the "secret code" of the encrypted text is formed.

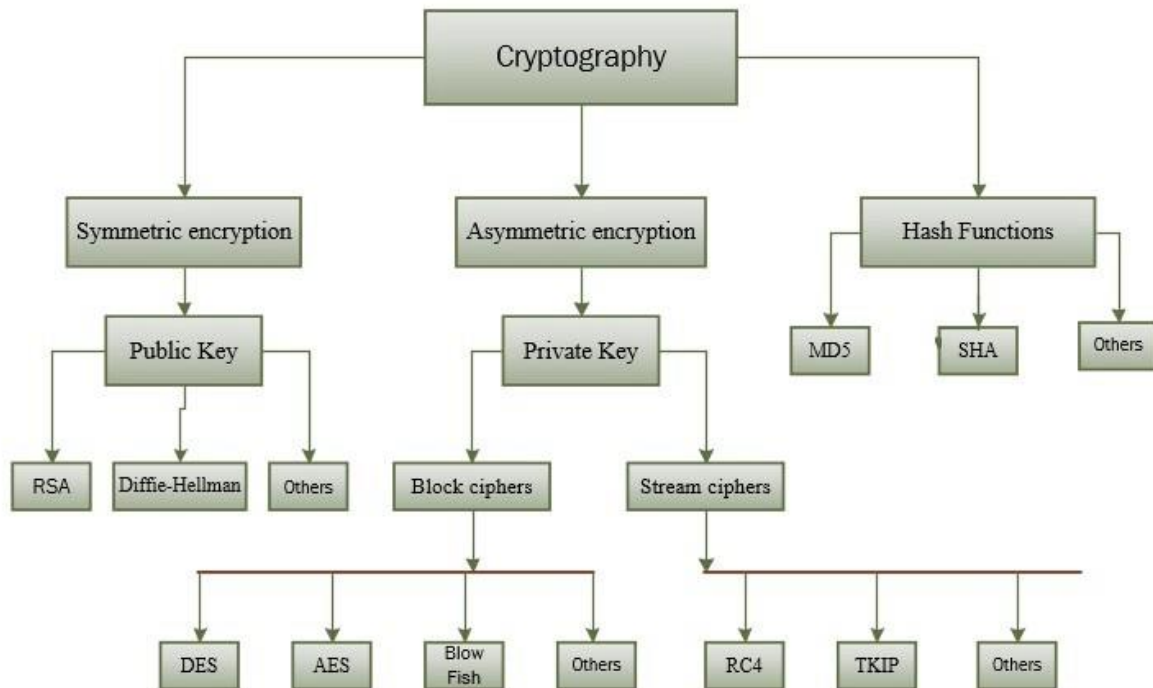


Figure 6: Breaking down cryptography [5]

Hence, depending on whether the method is symmetrically or asymmetrically, there exists different ways to encrypt the data, the “secret code.”

Algorithm 1: RSA Algorithm

The RSA Algorithm is an asymmetrical cryptography algorithm that makes use of two different keys, hence a public key to encrypt the data and a private key to decrypt the data. The keys are distributed using two large prime numbers. The encryption and decryption texts are represented as a symbol using the modulus function [1].

Algorithm 2: Encoding and Decoding using Base64 Strings

It usually works when a base64 encoding is a type of algorithm that converts a string of bytes into ASCII characters and vice versa, which is used for both encryption and decryption.

Algorithm 3: Cryptocode

This is the simplest way to encrypt and decrypt the python files, which uses built in python libraries. It makes use of a string encoded as a primary key.

Algorithm 4: Cryptography Fernet

Cryptography makes use of mathematical operations for securing information. The fernet allows for symmetrical encryption, also known as the “secret key”, which allows data to be encrypted and decrypted.

Algorithm 5: Hash Function

Lastly, the hash table algorithm is a mathematical algorithm that maps the plain text to data of an array fixed size, known as the hash value. The figure below shows an example of how hash functions are used in cryptography.

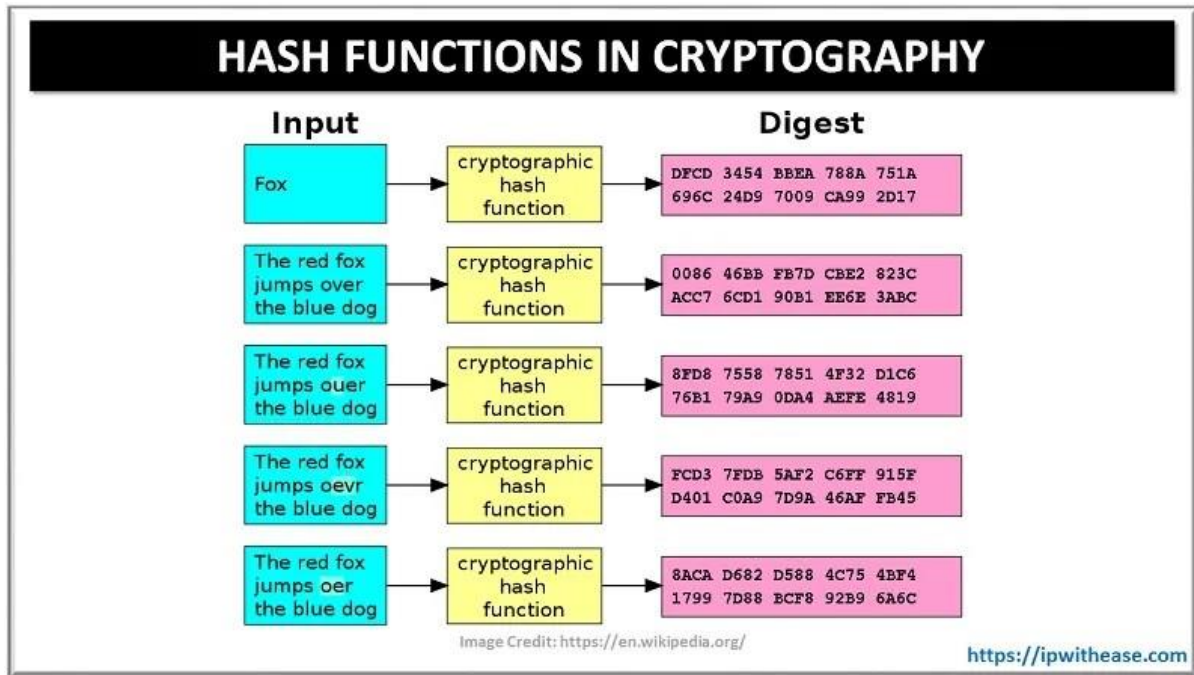


Figure 7: The Hash Algorithm [6]

As shown in the image above, the same words being used in a sentence, or message, will not result in similar hash values. The use of the same sentence with a variation in one or two letters results in a completely different hash value, with no similarities between them.

Properties of hash algorithm and why it is effective

- ❖ Non-predictable, hence guarantee safety of data
- ❖ Non-Revisable, hard to recreate original plain text
- ❖ Collision Resistances, hence a unique combination for each text

4.3. Subsystems Design

The shark Buoy is the main system, and the aim of this project is to implement a sub-system of the shark buoy- which is ARM based digital IP using the Raspberry-PI. The sub-sub systems are:

- Encryption block
- Compression block

The overall system includes an IMU block to sample data from the IMU. The UML diagram below illustrates how the system is intended to function.

As shown, the user will select an option between processing and retrieving the data. The 'process data' function makes use of the IMU data to send a file to the compression block that can then be compressed and encrypted. The 'retrieve data' function mimics later actions; after transmission has occurred. This function is necessary to ensure that the output of the system is retrievable.

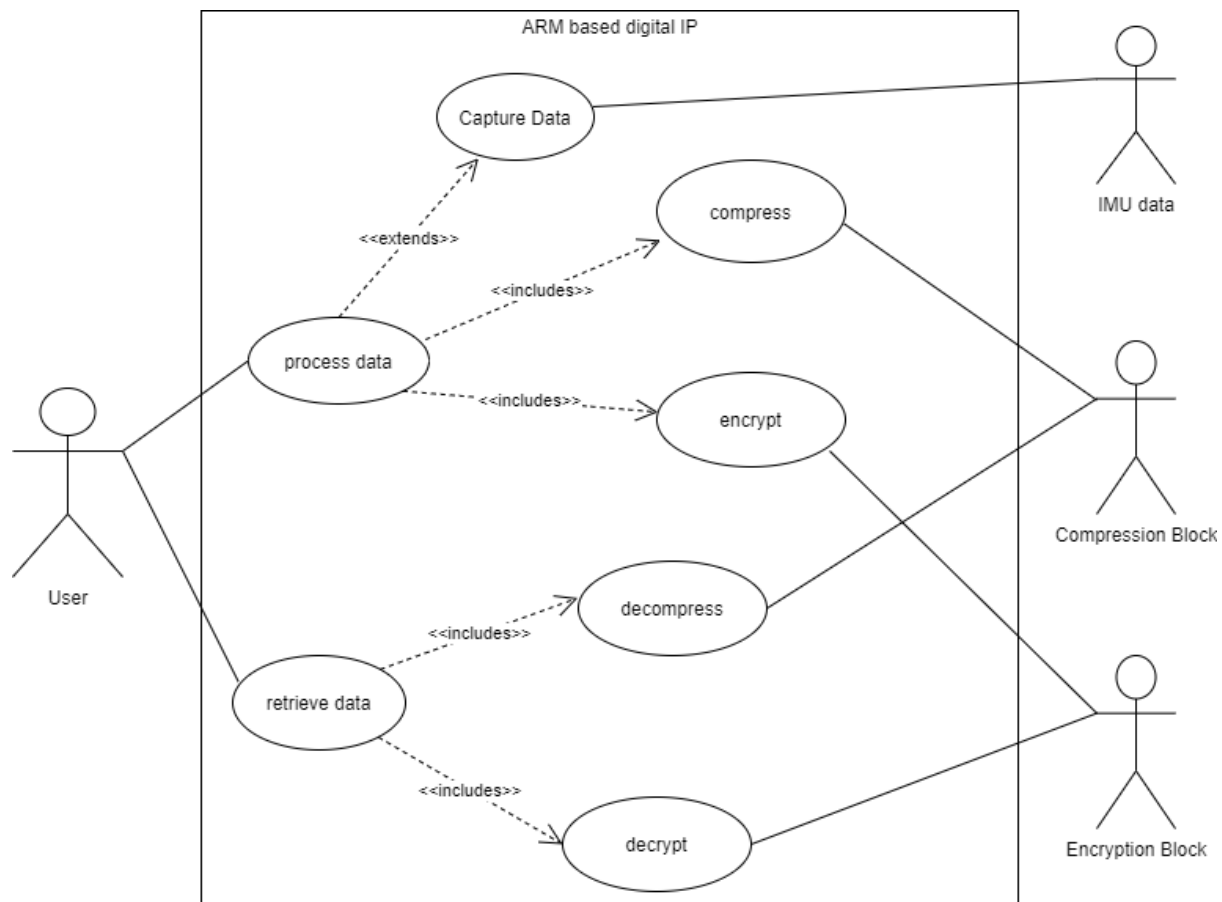


Figure 8: UML diagram for system

Inter – Subsystem and Inter Sub-subsystems Interactions

- IMU data is used to perform the tasks the IP is supposed to i.e. processing data from an IMU, compressing, and finally encrypting it.
 - The data read from the IMU (in real-time/pre-recorded) is passed to the compression block as a text file(.txt) or comma separated file(.csv)
 - The file to be compressed should be no more than 10MB
- The Compression subsystem handles the first part of the task:
 - data is input into this module and formatted accordingly. Next, the data is compressed according to the standards specified.
- The Encryption module handles the protection and formatting of data to be sent from the IP

- The compressed file (.gzip) from the compression subsystem is the input into this module. This compressed file is encrypted and a key is given to the user
- The output of this subsystem, and thus the output of the overall system, is an encrypted compressed .gzip file.

4.4. Feasibility

The datasheet for the ICM20469 contains enough information to use to design the system without having a physical IMU available for design testing. Matlab has a package that will make simulation of data much easier if the parameters are determined correctly. Further, there has also been enough research in the Antarctic to provide sample data to compare simulated data with. The old data from the already available research makes a good starting point for realistic data that the system can be tested with.

The Raspberry Pi Zero has libraries such as the AES encryption library which will aid in the design of the compression and encryption modules. Python is a language that is well suited for running on the device. Since most of the algorithms explored have already existing Python libraries, the designed IP should be able to compress and encrypt data in a considerably efficient time as compared to if new algorithms were designed from scratch. This will also mean that it is less likely to encounter bugs due to the implementation of the respective algorithms. Using already existing libraries to implement the algorithms also guarantees that an ideal compression ratio and encryption method are used in the design. The various algorithms can be interchanged to find the best combination to meet the size and security requirements.

With careful use of all the relevant information provided, the design of this IP is quite feasible.

4.5. Bottlenecks

The system will only be as efficient as it's least efficient component. Some challenges perceived for this project are:

- selecting a sampling frequency that will allow for accurate and realistic data to be recorded and used.
- Selecting a sampling frequency higher than the rate at which the compression block can process information may result in loss of information
- A lag in pushing the file outputs from the compression block into the encryption block.
- System might be prone to hackers and other cyber-attacks which may require even more security to ensure no one is able to access the data
- Acceptance Tests may take longer to produce and may not be as reliable when using the provided IMU due to different functionalities and the IMU not being used in the same conditions.

5. Validation using Old Data

Why do we need simulation-based validation?

Simulations models are an approximation of the real time data systems. Although the data never exactly meet the exact real time world system, it stills provides a close approximation of how the system reacts in real life.

The verification and validation of data is the process of confirmation that the data is correctly implemented with respect to the model system. During the verification step, the model is being tested to find error, and then fixed if found. Various methods and process are being used to assure the model's specifications and assumptions meets the expectations of the real model system.

The validation process, checks the accuracy of the model. Model Validation can define as the substantiation of computerized system that is within its domain of applicability, and possesses a satisfactory range of accuracy consistent with the intended application of the mode.

Simulations also involve assumptions being made about the model to estimate our approximation. There are two types of assumptions: Structural and Data Assumptions.

Structural Assumptions are assumption based on how the system operates and how it is physically arranged. Data assumptions are assumptions based on the appropriate data available to use.

And lastly, the testing is the last step. Hence, the tests involve comparison of the output from the system with the actual output from the real time model.

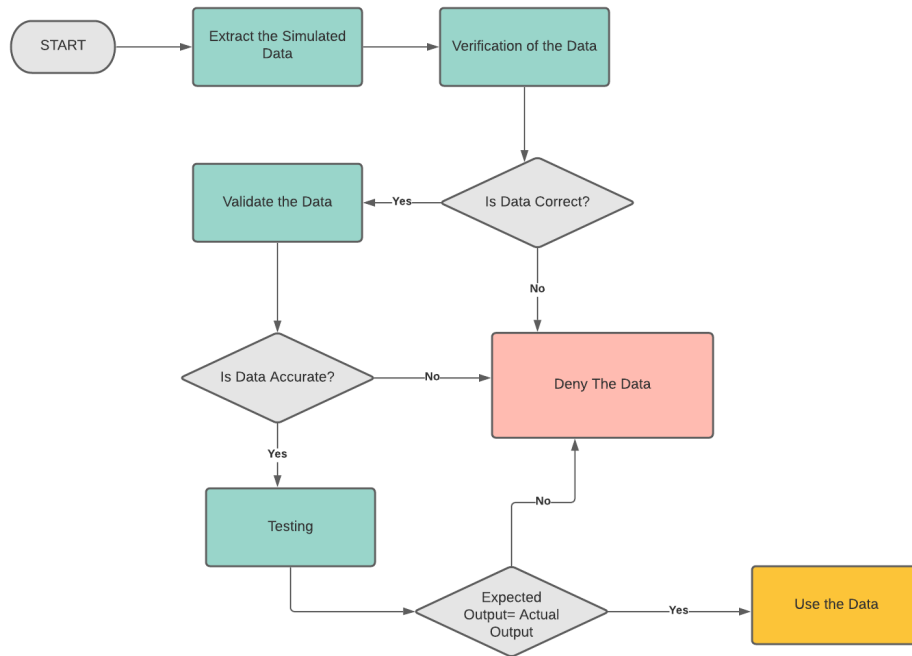


Figure 1: Data Verification

Figure 1, shows the verification process of data when decided to use Simulated data or not. Firstly, the data needs to be verified, then validated and tested. If the data passes all the tests, the data will be accepted and can be used to approximate the real time model. Otherwise, if the data fails any of those tests, it gets rejected and denied.

Data Used

According to the requirements, it is expected that we use data that one would gather from an IMU Sensor, hence the IMU-20948. The data includes a 3-Axis gyroscope (reading in x, y & Z direction) ,3 -Axis accelerometer, 3 axis Magnetometer, pitch, press, temperature of the IMU sensors, Yaw, Roll, and DMP Features. This data will be collected from the IMU sensor hat which will be attached to the raspberry board, but because we were not provided the IMU sensor yet, the raw excel data provided will be used as a substitution for the actual data.

This raw IMU data provided will be used for testing to validate the proposed design. This data was recorded by an IMU sensor with similar specifications, with the data being collected from a boat. This makes the data ideal as the environment of the captured data is similar to the use case for our design (i.e., data captured from a boat, and not from a different moving object which would produce different results).

Rationale for using chosen data

The time stamps recorded will allow for different sampling rates to be tested out, along with their corresponding performances in the system. Further, as the data resembles typical IMU sensor data, the Fourier coefficients for the respective measurements can be calculated through the use of available libraries. These coefficients will be used to confirm the system does preserve the lower 25% of the Fourier coefficients.

The reason why the data will be good enough to test our ATP and specifications, is because the raw data provided is an actual representation of the real time data that is expected to be extracted from the Sharc Buoy. Thus, any approximations or tests performed on raw data should be equivalent to any data analysis performed on real time data with minimum errors.

From our ATP table and specifications, the majority of our tests is testing whether the algorithms are effective and able to perform their function, e.g timing the time complexity of algorithm, calculating compression ratio or data loss, thus using data that represents the actual data used in Sharc Buoy should still provide a good approximation of what can be expected from the real time data.

Initial analysis of the data using Matlab

Initial analysis of the data produced the following time domain and frequency domain plots:

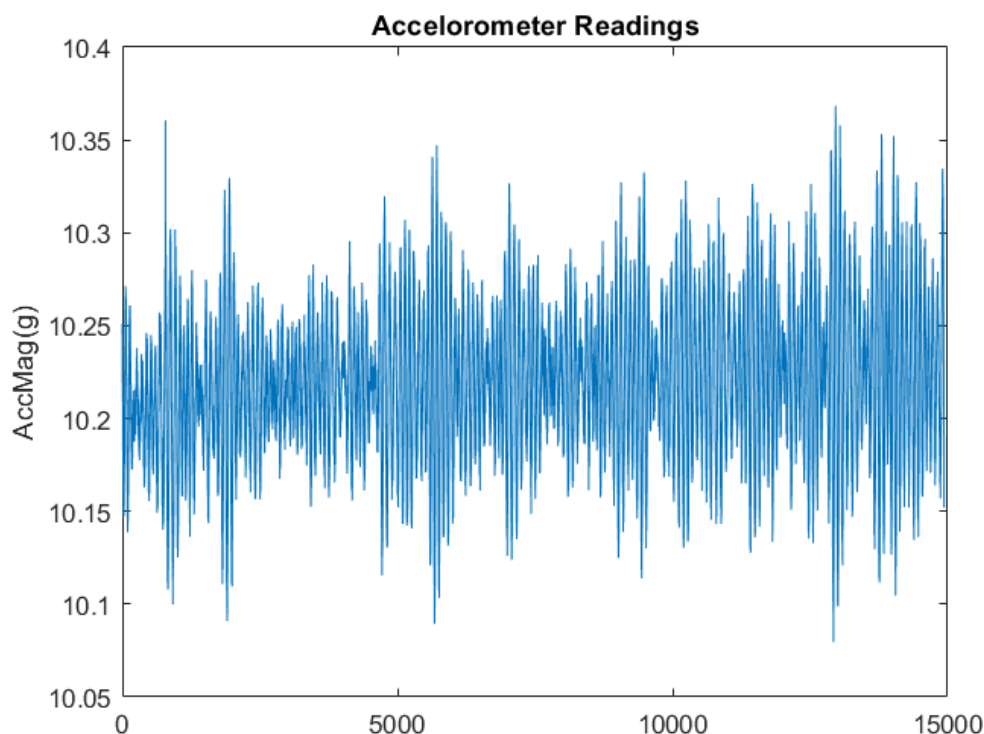


Figure 9: Time-domain plot of ACC magnitude

The overview of the accelerometer readings from the data captured in one of the data files provided is shown above. This plot shows the small range that the readings fluctuate over. It also reveals that there are many fluctuations over time.

A zoomed in representation is shown below:

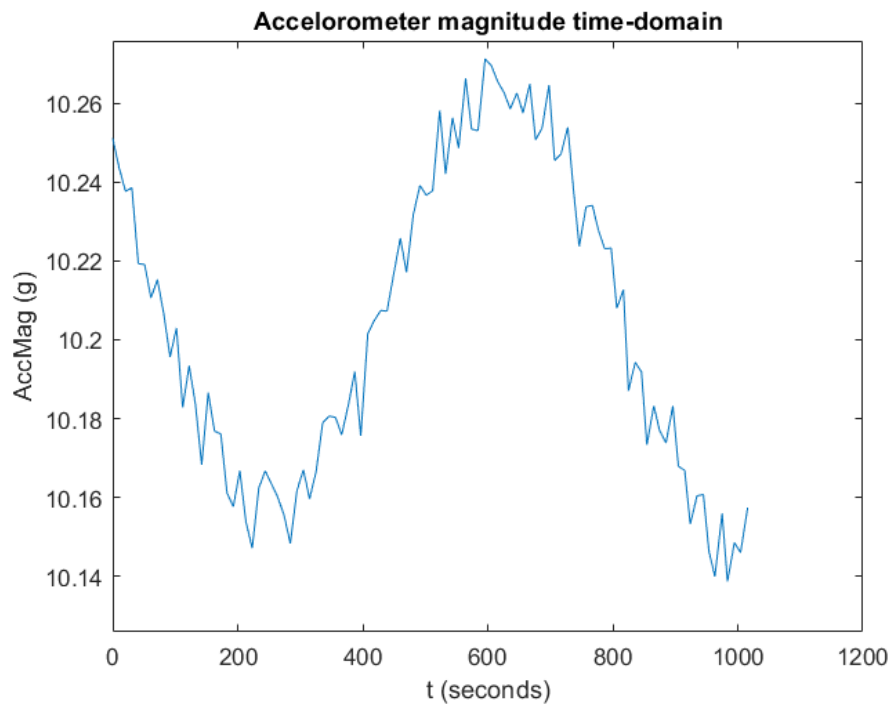


Figure 10: Time-domain plot of accelerometer readings

The figure above shows the time-domain plot for a selected period of time. As shown, the magnitude of acceleration fluctuates between 10.14 m/s^2 and 10.27 m/s^2 .

The overview of the gyroscope measurements is also plotted, showing both the overall data, and then a zoomed in representation in Figure 12:

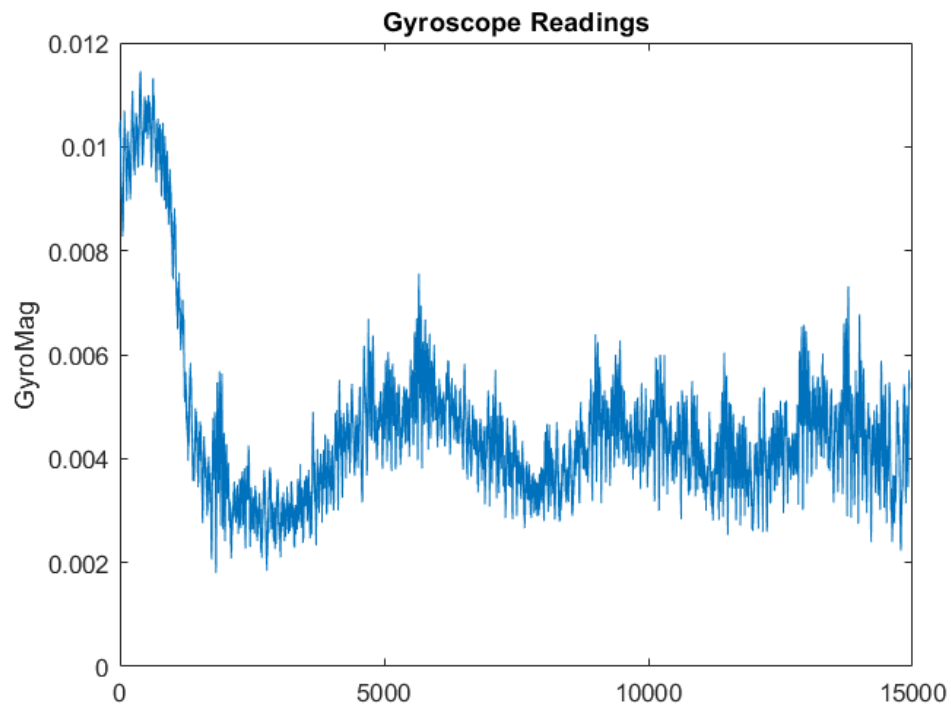


Figure 11: Overview of Gyroscope time-domain

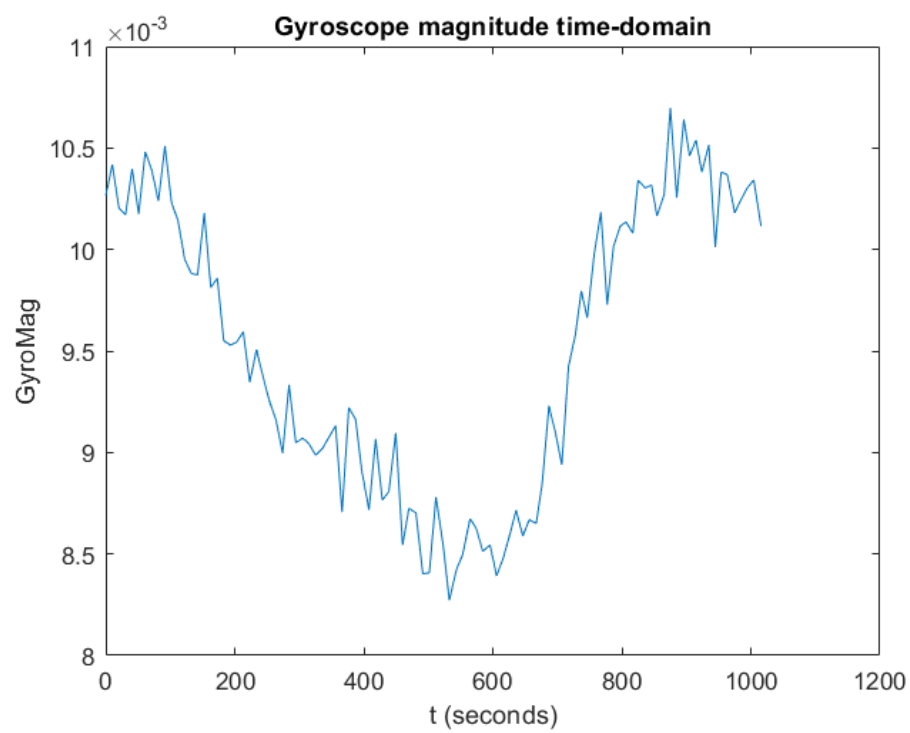


Figure 12: Time-domain plot of gyroscope readings

The gyroscope readings are shown to be much smaller in magnitude.

The Fourier transform of the accelerometer and gyroscope measurements were calculated and plotted as shown below. A zoomed in sample of these plots is conveyed in this report

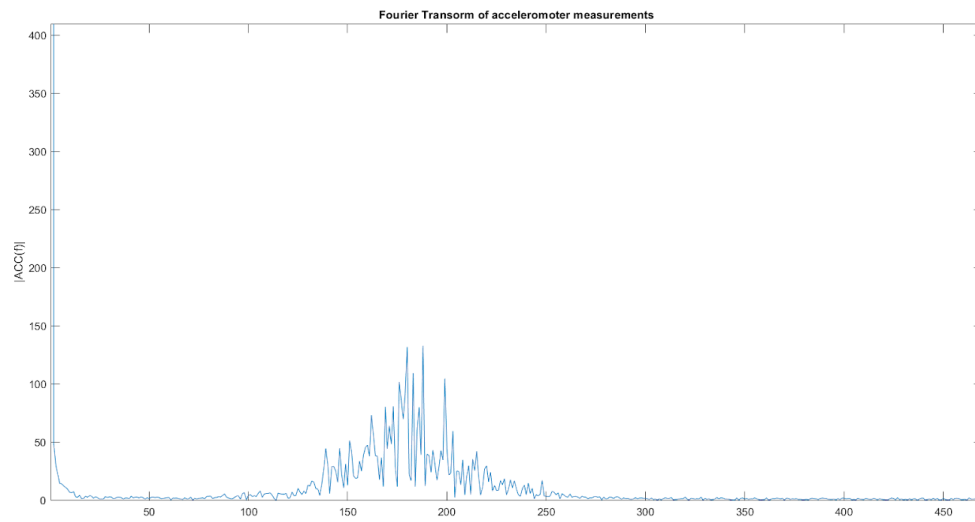


Figure 13: Acceleration Fourier Transform

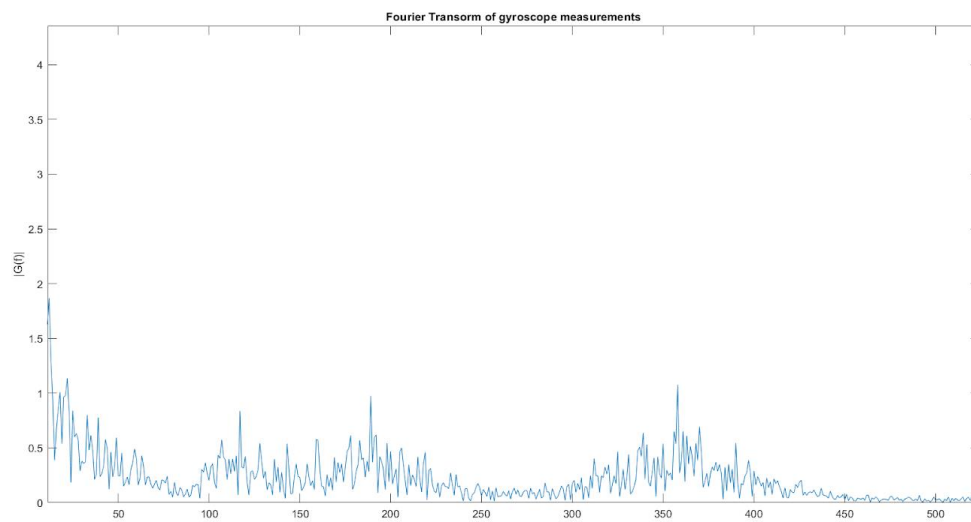


Figure 14: Rate of rotation (Gyroscope) Fourier Transform

5.1. Experiment Setup

5.1.1. Overall System functionality

As the system must run on a raspberry pi 0, sample data in CSV files is passed to the system through an ssh file transfer protocol (WinSCP). This allows for the real-time transfer of data. To check the overall functionality of the system, raw data files are fed into the processing block and read by the compression subsystem. The compression subsystem outputs a zipped file that the encryption subsystem will then encrypt, thus outputting an encrypted, compressed file.

The execution time for the processing of the file will be recorded, along with the initial and final file sizes. These tests will be run several times with different file inputs, each with either a different sampling rate or noise added.

Fourier coefficients of the data will be calculated using the original raw data, and the data from the decrypted, and decompressed file. These two Fourier samples will be compared and the mean squared error will be calculated.

5.1.2. Experiment Design: Compression

A python program is created to test for the different criteria that need to be met. These tests are driven by the compression ratio, speed and integrity requirements previously specified. Several algorithms are compared to ensure that the most efficient compression algorithm is chosen. For each data set, 10 iterations are run and the average of these results is taken.

Test 1: Compression Ratio

A sample file is fed into the program, and the program is run in a loop. The input file size is recorded (in bytes). Next, for each algorithm chosen for comparison: zlib, gzip and lzma, the same data file is compressed and written to an output file. The size in bytes of the contents of this output file is also recorded. These two are compared and the compression ratio, along with the % reduction are calculated and recorded:

$$\% \text{ reduction} = \left(1 - \frac{\text{compressed size}}{\text{original size}}\right) * 100$$

Test 2: Speed of Algorithm

During the same experiment, the time before and after each compression and decompression is recorded. The average of all 10 readings from the compression and decompression is recorded and these values are compared.

Test 3: Data integrity

Finally, once the output data has been decompressed, the contents of the original file and the decompressed file are compared in the python program to check whether any loss has occurred.

The results of these tests will determine which algorithm will be used for the system, taking into consideration a time-size trade-off.

5.1.3. Experiment Design: Encryption

Test 1: Data Check

Test whether the algorithm has the correct encryption code, hence when the IMU sensor data is supplied, an encrypted data needs to be output.

Input	Expected Output
The compressed data of the IMU sensor, i.e the accelerometer, gyroscope, etc	An encrypted text, which differs from the data sensor input

Table: Test for Encryption

Test 2: Algorithm Speed Complexity

Test the time each algorithm takes to compile the data from the IMU sensor into the appropriate encrypted text.

Input	Expected Output
Time the algorithm.	Algorithm speed needs to be faster than the average performance.

Table: Test for Encryption

Test 3: Correctness of Data

Test the correctness of the Data. Once the data has been encrypted, the data should be decrypted back to the “original” IMU sensor data. The original data should be comparable to the decrypted data. This experiment tests for the correctness of the decrypted algorithm. How close is the data to the original? Was there any data that was lost?

Input	Expected Output
Decrypt the encrypted text	Data needs to coincide with the original imu sensor data

Table: Test for Encryption

Test 4: Data Loss

How much data is lost. Once the data has been decrypted, calculate the data percentage that is loss, using the formula:

$$Data\ Loss = \frac{Data\ Loss}{Original\ Data} \times 100$$

Input	Expected Output
Decrypt the encrypted text	Calculate the data loss percentage, hence needs to be as low as possible

Table: Test for Encryption

Test 5: Strength of Algorithm

Every algorithm has its weak and strong points, this test is to test how easily the algorithm breaks, this test will check whether the data can be decrypted if the user does not have the correct "Key" to decrypt.

Input	Expected Output
Attempt to decrypt the encrypted text, with an invalid key.	A valid message, stating that the user cannot decrypt the text file without the key.

Test 6: Size of Encrypted and decrypted data

When data gets encrypted, its size changes. The aim of this experiment is to study how exactly the size has changed.

Input	Expected Output
Measure the original size of data and also measure the size of the encrypted and decrypted data	The size of encrypted data needs to be equal to the original data size, hence means no data is lost. The size of the encrypted file needs to be greater than the original data size.

Table: Test for Encryption

5.2. Results

5.2.1. Overall System functionality

A test of the whole system is carried out after both subsystems have been compiled. The overall time taken to complete the compression and encryption on a pi for a given data sample is 34.52s.

As expected, the output of the system block shows no loss, and thus the root mean square of the error in the Fourier coefficients of the input file and output decompressed file is 0. This however, does not take into account the noise that may be picked up from transmission. Thus, it can be concluded that the overall algorithm is effective as all the compressions ATP tests and encryption ATP tests were met as shown in Table 12: ATP Tests.

5.2.2. Results: Compression

The compression tests produced the results conveyed in the screenshot of the text file shown below:

```
| 3793914 output\sampleOut_gzip.csv.gzip
Average compression time: 1.1618026733398437
Compression ratio: 2 with reduction of 60.14%
Average decompression time: 0.055907368659973145
| 3809410 output\sampleOut_zlib.csv.zlib
Average compression time: 0.8714528918266297
Compression ratio: 2 with reduction of 59.98%
Average decompression time: 0.045366084575653075
| 2913444 output\sampleOut_lzma.csv.lzma
Average compression time: 3.48464613755544
Compression ratio: 3 with reduction of 69.39%
Average decompression time: 0.1371371348698934

The size winner is: lzma
The time winner is: zlib
```

Figure 15: Test results from comparing compression algorithms

Output of test results

As shown in the figure above, using the lzma library produced the slowest compression time, but had the largest compression ratio. This compression takes on average 3s to compress about 9Mb and reduces the file size. The compression ratio is 3.

The gzip and zlib libraries produce similar results, at half the time that the lzma algorithm takes. The compression ratio for both these algorithms is 2, with the difference in reduction percentages being only 0.16%.

When noise is added to the sample data, the compression ratio slightly decreases, and as when the sampling rate is changed, the time taken changes accordingly.

```

1763503 output\sampleOut_gzip.csv.gzip
Average compression time: 0.5527220726013183
Compression ratio: 2 with reduction of 58.37%
Average decompression time: 0.02597970962524414
1773485 output\sampleOut_zlib.csv.zlib
Average compression time: 0.41442830562591554
Compression ratio: 2 with reduction of 58.13%
Average decompression time: 0.021257245540618898
1412548 output\sampleOut_lzma.csv.lzma
Average compression time: 1.3821890274683635
Compression ratio: 2 with reduction of 66.65%
Average decompression time: 0.0704848051071167

The size winner is: lzma
The time winner is: zlib

```

Figure 16: Comparison of compression algorithms at half the sampling rate

For each of the algorithms tested, the time it takes to compress and decompress was halved when the sampling frequency was halved. This corresponds to the size of the data file which will also decrease.

Conclusions:

A compromise of speed and time need to be taken into consideration, along with ensuring maximum data integrity. As a result, the gzip library will be used for the design. This is as proposed in the initial design where the gzip library was preferred because of the CR32 checksum carried out. This algorithm is preferred as the higher compression will result in less storage space being taken when large amounts of data are required. This is a constraint that our design will have to satisfy.

5.2.3. Results: Encryption

For the encryption techniques, four algorithms will be tested and compared.

Test 1 : Data Check

The first experiment checks whether the encrypted data matches the expected data.

Algorithm	Data Check
Algorithm 1 : RSA Algorithm	The encryption algorithm works only if the IMU data received per line is less than 53 bytes, otherwise data larger than 53 bytes will not be properly encrypted.

Algorithm 2 : Base64 Strings	The encryption works effectively, all the data has been encrypted correctly.
Algorithm 3 :Cryptocode	The encryption works effectively, all the data has been encrypted correctly however the algorithm is very slow.
Algorithm 4 : Cryptography Fernet	The encryption works effectively, all the data has been encrypted correctly.

Table: Test for Encryption

As shown from the above table, all algorithms match the expected data of their particular encryption technique, expected for the RSA Algorithm which won't encrypt data that are larger than 53 bytes.

Test 2: Algorithm Speed Complexity

This is to test the time complexity of the algorithms, exactly how fast the algorithm executes.

As shown in below Figure 17, the RSA and Cryptocode Graph have the longest time complexity, whereas the Cryptography and Base64 Graph have the shortest time complexity of less than 2.5 milliseconds.

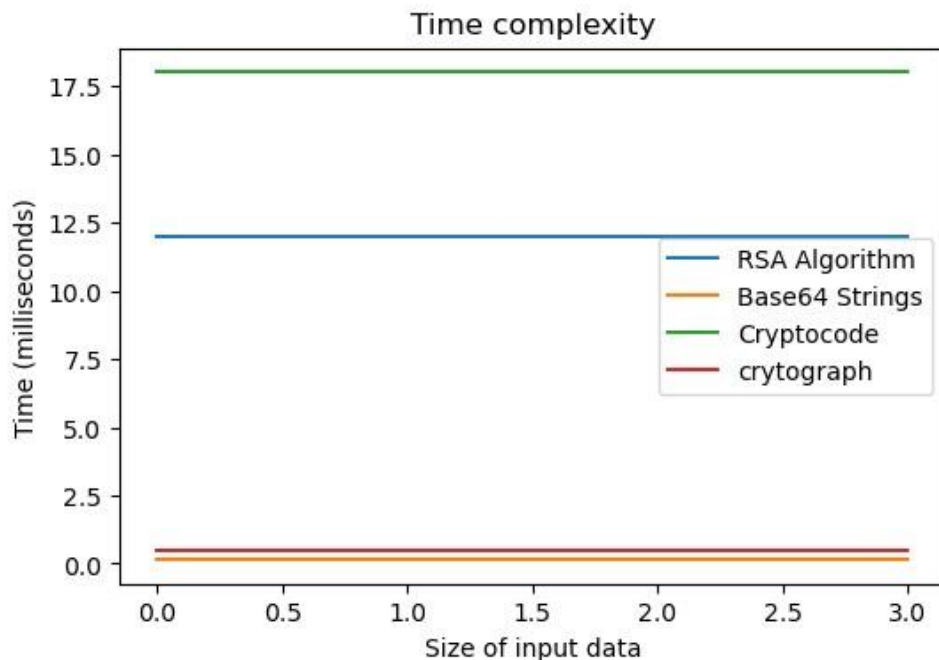


Figure 17: Time Complexity Graph for comparison of Encryption Algorithms

Test 3: Correctness of Data

This is used to ensure that the decrypted data matches the original IMU data.

Algorithm	Correctness of Data:
------------------	-----------------------------

Algorithm 1 : RSA Algorithm	The decrypted data contains the same exact data as the
	original text, however the format (e.g. new line, paragraph) is lost.
Algorithm 2 : Base64 Strings	The decrypted data is equivalent to the original data, hence no data is lost
Algorithm 3 :Cryptocode	The decrypted data is equivalent to the original data, hence no data is lost
Algorithm 4 : Cryptography Fernet	The decrypted data is equivalent to the original data, hence no data is lost

Table: Test for Encryption

Test 4: Data Loss

This calculates the percentage of data that was lost from test 3, using the formula:

$$Data\ Loss = \frac{Data\ Loss}{Original\ Data} \times 100$$

Algorithm	Data loss (Percentage)
Algorithm 1 : RSA Algorithm	20% (Format of data is lost)
Algorithm 2 : Base64 Strings	0%
Algorithm 3 :Cryptocode	0%
Algorithm 4 : Cryptography Fernet	0%

Table 10: Comparing data loss for the selected encryption algorithms

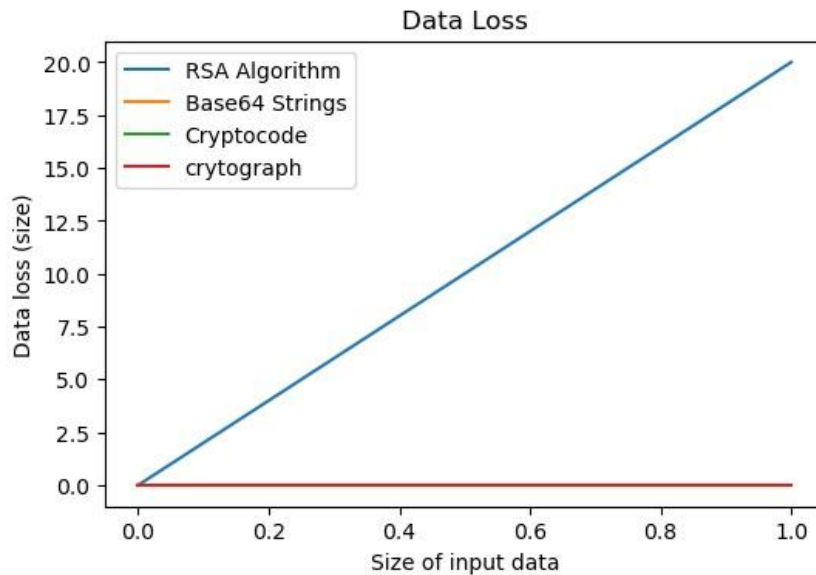


Figure 18: Comparing data loss for the selected encryption algorithms

As shown in Table 10 and Figure 18, the RSA algorithm is the only algorithm where their data is lost as the size of the input increases. The other algorithms are able to maintain a 0% data loss with an increase in data size.

Test 5: Strength of Algorithm

This is to test how algorithms behave with unexpected behaviour, i.e Trying to decrypt the data with an invalid key.

Algorithm	How do algorithms react with Invalid keys?
Algorithm 1 : RSA Algorithm	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided. Hence, an expectation error is thrown.
Algorithm 2 : Base64 Strings	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided.Hence, an expectation error is thrown.
Algorithm 3 :Cryptocode	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided.Hence, an expectation error is thrown.
Algorithm 4 : Cryptography Fernet	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided.Hence, an expectation error is thrown.

Table 11: Behaviour of selected encryption algorithms when invalid key is used

Test 6 : Size of Encrypted and decrypted data

This experiment is used to compare the size of the encrypted and decrypted file to the original file. The original file size is 8874 KB.

Algorithm	Encryption Size	Decryption Size
Algorithm 1 : RSA Algorithm	56KB	2000KB
Algorithm 2 : Base64 Strings	23648KB	8874 KB
Algorithm 3 :Cryptocode	18645KB	1143KB
Algorithm 4 : Cryptography Fernet	1245KB	8874 KB

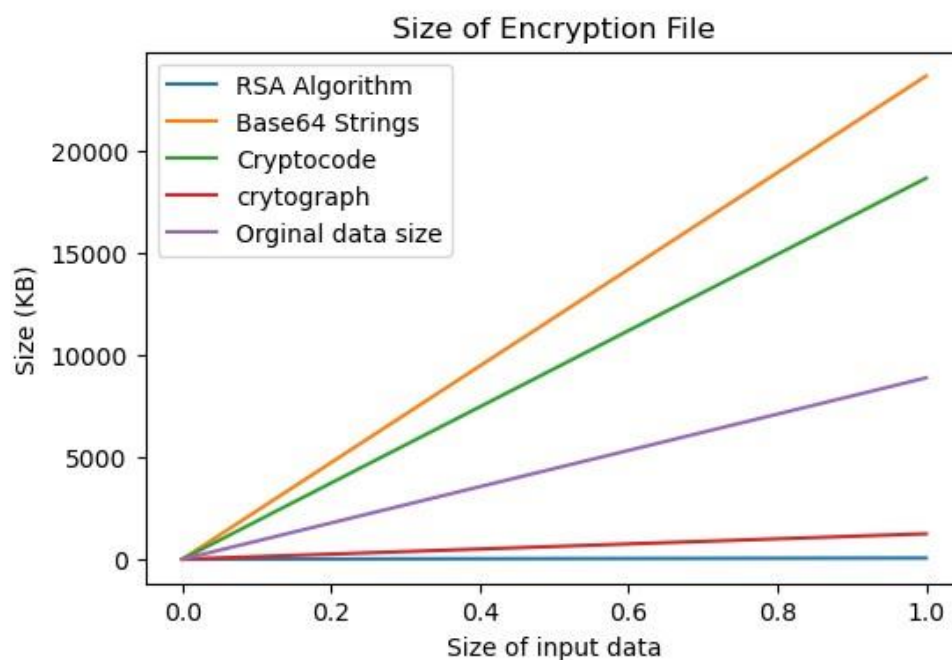


Figure 19: Size of encrypted Data against input data

For the encryption data size, the size of the algorithm needs to be greater than the original non -encrypted data size, as this shows that an effective encrypted algorithm has been implemented with major reductions. A smaller encrypted size file means that less minor reductions have been adjusted to the original text. In Figure 19, it is clearly shown that the Base64 Strings algorithm and the cryptograph algorithm are effective, due to their encrypted data being bigger, meaning more random and additional headers have been implemented in the algorithm.

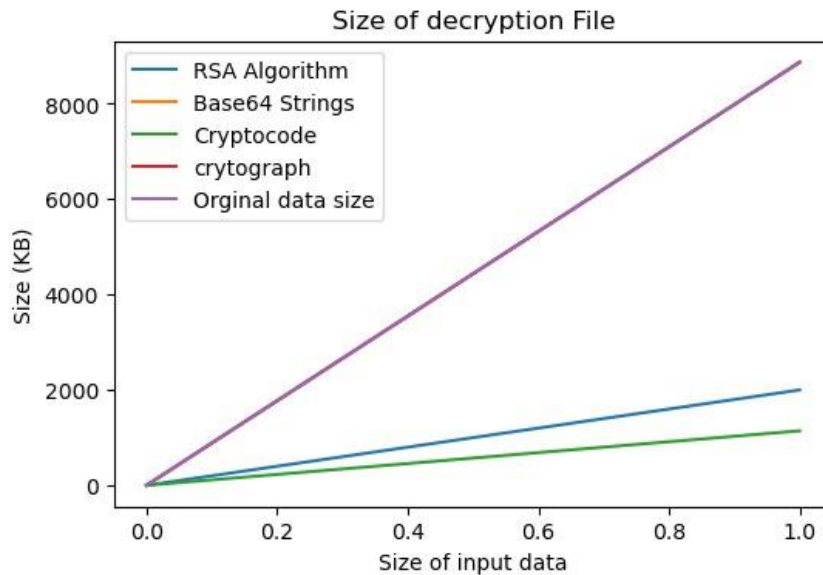


Figure 20: Size of decrypted data against input data

For the decryption data size, the size of the algorithm needs to coincide with the original data size. This will mean that no data has been lost. If a decryption data size is smaller than the original, it means some data have not recovered from the encryption.

6. Validation using a different IMU

Why do we need hardware-based validation?

Hardware validation is the ensuring of the accuracy of the hardware for a well-functioning system. This is to ensure that the hardware required for the system is able to function correctly as expected. This is important as to ensure that there are no defects in the IMU, and hence the model is working as expected, e.g., the imu is not broken or damage.

The hardware process involves three stages: Hardware verification, hardware testing and hardware maintenance. Hardware verification is the independent testing of algorithms to check whether the hardware is working as expected. Hardware verification could include

- Testing under hardware conditions simulating expected real-life conditions, such as storage, transportation
- Ensuring that the hardware performs in extreme weather conditions, in our case cold weather in ice buoys
- Ensure hardware is expected to perform in under hardware conditions
- Security measures are in place
- Quality assurances are in place

Hardware Testing is ensuring every component on the IMU Sense hat is operating as it should and performing exactly in accordance with the specifics. Example, include testing measures such as

- Developing a set of Test criteria
- Applying functional tests to ensure which tests criteria have been met
- Verify output is compatible with expected output
- Applying non-operating tests to ensure hardware can withstand any physical handling e.g., dropping IMU

After the hardware have been verified, tested and implemented, it must be correctly maintained. Most hardware items will come with a maintenance schedule, hence by the manufacturer or supplier. It important that the hardware system is maintained so that it can continue to perform at the level of expectations when needed.

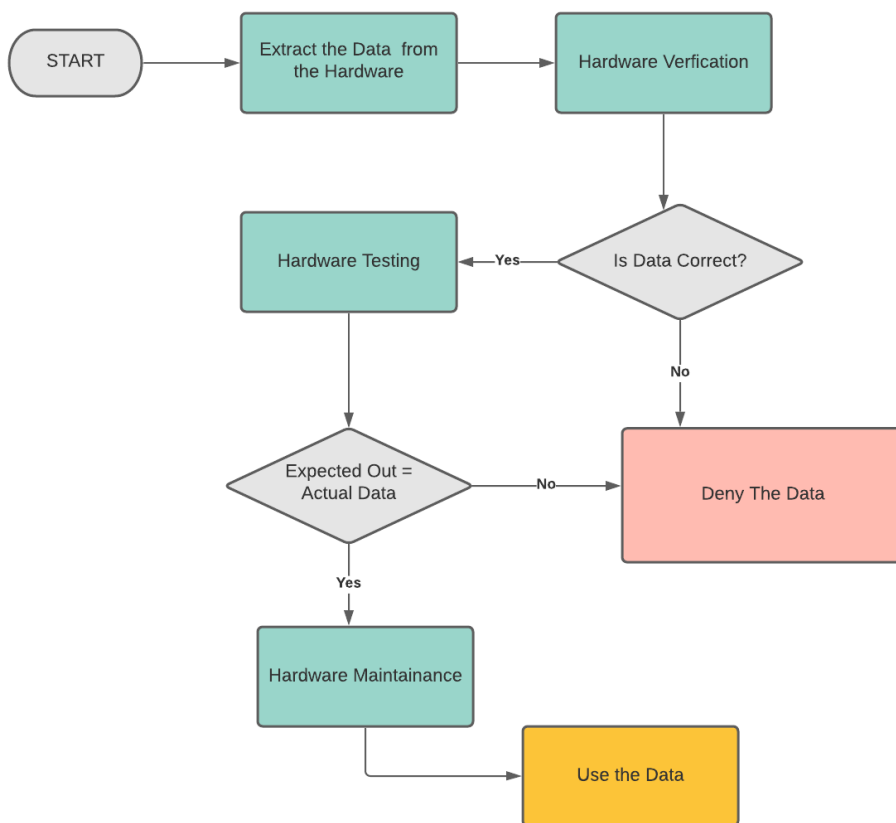


Figure 21: Hardware Process

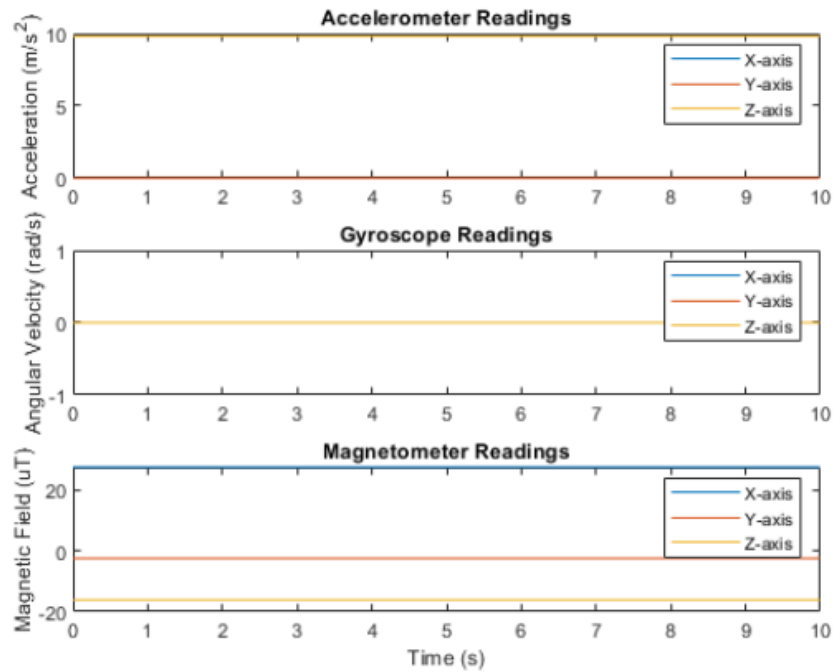
Figure 21, shows the hardware verification steps for the implementation. Hence, the hardware is first verified, then tested and thereafter it needs to maintained to be used continuously when required.

Data Used

IMU Sense Hat, are specifics types of sensors that measures force, angular force and magnetic field. IMU are a consist of 3-axis of accelerometer, 3 – axis of gyroscope, which could be 3 – axis of magnetometer, would be a 9-axis IMU.

Accelerometer, is measure of change of velocity in the imu sense hat across a single axis. Accelerometer, measures linear acceleration in a single direction. Gyroscope measures the angular velocity about three axes, hence pitch, yaw and roll. Lastly, the magnetometer, measures the fluctuations in the Earth's magnetic field by measuring the air's magnetic flux density.

Rational Using Data



21

Figure 22: Readings from IMU

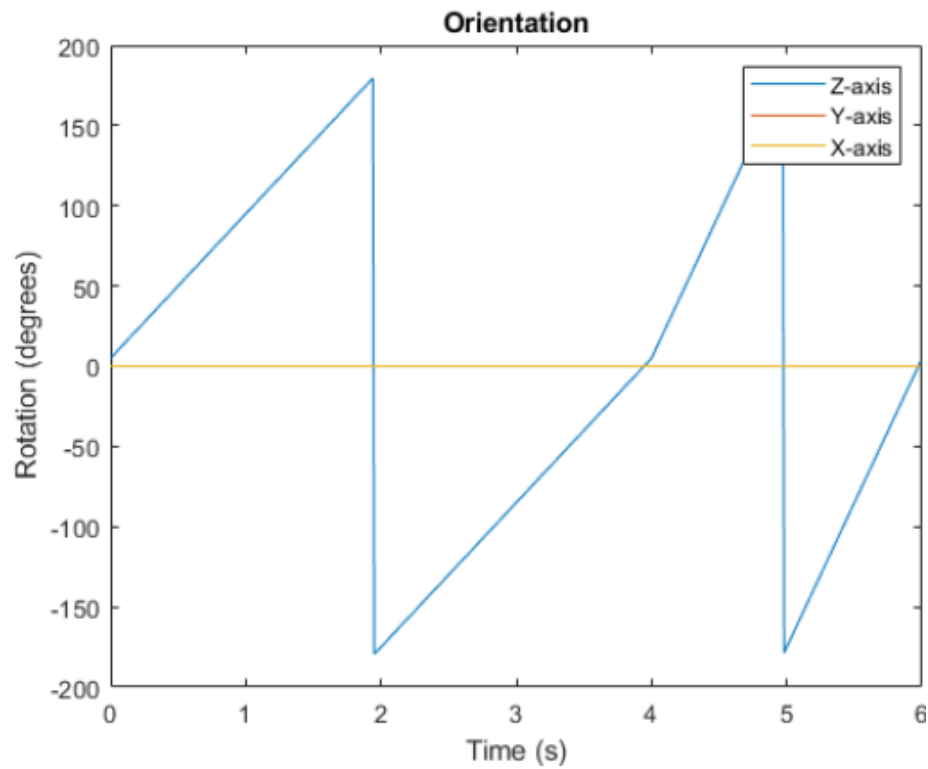


Figure 23: Readings from IMU

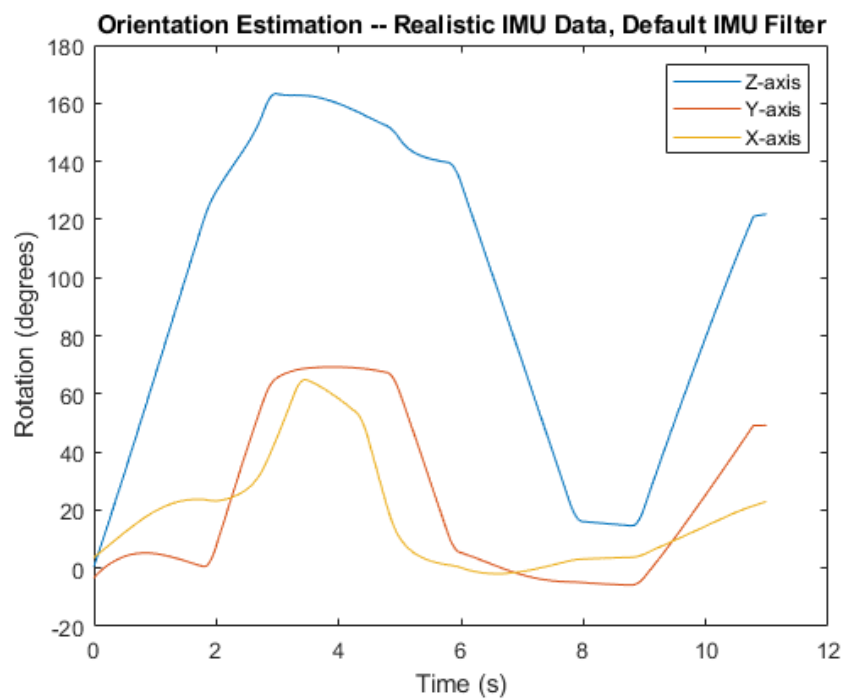
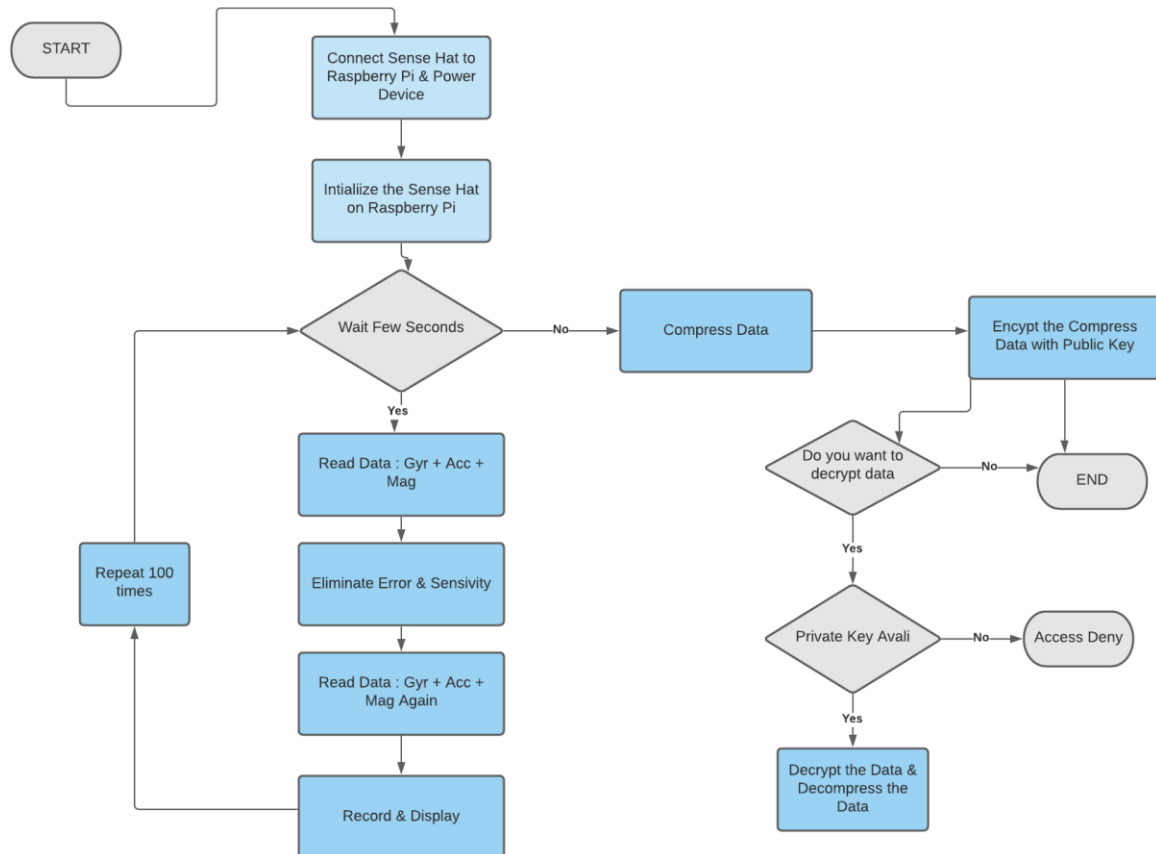


Figure 24: Readings from IMU

As shown in the above figures, the data collected, the data read from the IMU is ideal and matches the expected data, when the imu sensor is in motion.

Steps taken to make sure that testing with this IMU can be extrapolated easily to the IMU to be used in the actual buoy:



22 Figure 25: Initialise Steps

As shown in the image above, after initialisation of the IMU module, the data will be read after several seconds to allow for the system to start up. Further, several readings will be taken and compiled to send into the system. Although the ICM being used is a 9-axis MEMS device and has more functionality than the ICM that will be used in the final module, only the relevant data will be read and processed. This will include the gyroscope, accelerometer, magnetic field, yaw, roll, pitch, temperature, pressure, and time. Doing so will mean a direct translation of our results to the final system.

The 400kHz fast mode I2C interface will not be used as the ICM20649 does not support this mode.

Validation Tests for IMU Sense Hat B:

Validation Test 1: Temperature & Pressure & Humidity Check

Test whether the IMU produces the ideal temperature, pressure and humidity as expected if placed in a room environment

Input	Expected Output
Measure Temperature, Humidity and Pressure of the IMU sense Hat in a room environment.	Ideal Room Temperature: e.g., 25 - 30 degrees

Table: Test for System

Results:

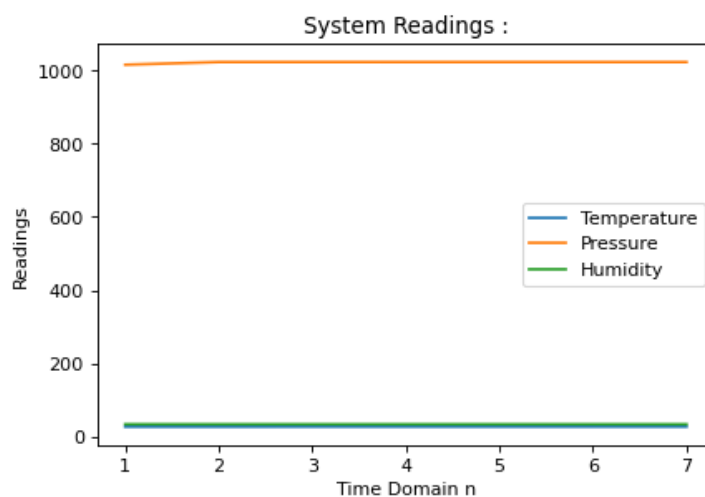


Figure 26: Readings

As shown in figure 26, the temperature and pressure reading of the IMU sensor is ideal for a room temperature environment.

Validation Test 2: Place a magnet near device

The magnetometer is used to read the readings of the magnet, hence in the presence of the magnet the reading of the magnet should be stronger than without.

Input	Expected Output
Record the readings with and without a magnet	Readings in the presence of a magnet should be stronger than without a magnet.

Table: Test for System

Results:


```

/-----/

Roll = -9.47 , Pitch = 4.54 , Yaw = 145.52

Acceleration:  X = 17786 , Y = -15725 , Z = 433

Gyroscope:     X = -2520 , Y = -6786 , Z = -2678

Magnetic:      X = 98 , Y = 25 , Z = -178

/-----/

Roll = 1.51 , Pitch = 0.33 , Yaw = 152.99

Acceleration:  X = 4172 , Y = 11818 , Z = -8340

Gyroscope:     X = 1171 , Y = -245 , Z = -248

Magnetic:      X = 24587 , Y = 145 , Z = -234

```

Figure 27: Accelerator Readings

Figure 27, shows the reading of the IMU sensor, first before the presence of a magnet and then with the presence of magnet. Hence, as shown in above figure, the magnetic field becomes stronger in presence of magnet.

Validation testing 3: Continuously Shake the device and Record Data

When the device is shaken, a random different sample of reading should be read from the IMU. Angular Velocity and acceleration are also observed

Input	Expected Output
Shake the device	Random Range of Output readings.
Shake device backwards and forwards in specific direction	Increase in readings on one axis

Table: Test for System

Results:

```

Roll = -12.41 , Pitch = 43.34 , Yaw = 154.37

Acceleration:  X = -12980 , Y = 2360 , Z = 3720

Gyroscope:     X = 4796 , Y = -29305 , Z = -1046

Magnetic:      X = -16492 , Y = 138 , Z = -262

/-----/

Roll = -7.15 , Pitch = 31.15 , Yaw = 157.28

Acceleration:  X = -11332 , Y = 1117 , Z = -4040

Gyroscope:     X = -2138 , Y = -17836 , Z = 166

Magnetic:      X = -495 , Y = 24623 , Z = 24656

/-----/

Roll = -14.06 , Pitch = 40.23 , Yaw = 148.62

Acceleration:  X = -22989 , Y = -2743 , Z = 5710

Gyroscope:     X = 1881 , Y = 1810 , Z = -6582

Magnetic:      X = 273 , Y = -55 , Z = 273

/-----/

```

Figure 28: IMU Readings

When the device is shaken randomly, the results are quite random as shown in figure 8. When the pi is moved flat along the various axes of the IMU, the acceleration oscillates on that particular axis.

Validation testing 4: Rotating the device along the IMU Sense Hat axes

When the Sense Hat is rotated, the values of x, y and z should change in proportion to a scale factor of 1 and -1.

Input	Expected Output
Record the readings in the forward direction, and then rotate the device in the opposite direction.	When the device rotates, reading should be in the opposite direction as the data recorded in forward direction.

Rotate about horizontal axis(x-axis)	Positive or negative gyroX reading depending on clockwise or anticlockwise reading respectively
Rotate about vertical axis (y-axis)	Positive or negative gyroY reading depending on clockwise or anticlockwise reading respectively

Table: Test for System

Results:

```

/-----/

Roll = 14.11 , Pitch = 31.14 , Yaw = -66.77

Acceleration:  X = -1451 , Y = 434 , Z = 23115

Gyroscope:     X = -2935 , Y = -11286 , Z = -2476

Magnetic:      X = 24578 , Y = 54 , Z = -123

/-----/

Roll = 8.59 , Pitch = 36.80 , Yaw = -63.41

Acceleration:  X = 1837 , Y = 2237 , Z = -22303

Gyroscope:     X = 378 , Y = 1821 , Z = -406

Magnetic:      X = -43 , Y = 64 , Z = -229

/-----/

```

Figure 28: IMU Readings

The clockwise rotations among the respective axes also show positive changes in the acceleration z direction reading, while the negative rotations show negative readings.

Validation testing 5: Detecting Pitch, Roll, yaw with the Sense Hat

The IMU Sense that should be able to detect a reasonable magnitude in all directions.

Input	Expected Output
Record the Pitch, Roll & Yaw of the device and rotate the device.	Reading should be ideal.

Table: Test for System

Results:

```

Roll = 1.51 , Pitch = 0.33 , Yaw = 152.99

Acceleration:  X = 4172 , Y = 11818 , Z = -8340

Gyroscope:      X = 1171 , Y = -245 , Z = -248

Magnetic:       X = 24587 , Y = 145 , Z = -234

```

Figure 29: IMU Readings

This test is run in conjunction with test 4. For the corresponding axes, the pitch, roll and yaw also change respectively when the device is rotated. There is very little change observed when the device is being moved (and not rotated) along a particular axis

6.1. Simulated model vs Hardware model:

Differences between the provided IMU sensor and the one specified in the final requirements


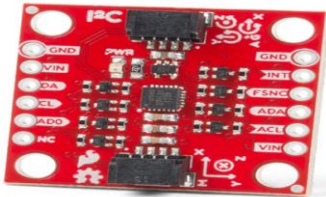
Sense Hat Type B(Hardware data)	ICM-20469 (Simulated Data)
	

Table: Test for System

As shown in above Table the IMU Sense hat B has more features, such as more GPIO pins, wave share, more built-in sensors and a voltage regulator. Hence, this provides an advantage as more data can be collected for use.

Feature	IMU Sense Hat B	IMU Data (Supplied)
Type of IMU Sense Hat	ICM20948 Sense Hat B	ICM-20649
Operating Voltage	1.71 - 3.6V	1.71 to 3.6V
Gyroscope	<p>Ranging: $\pm 250/500/1000/2000$ dps</p> <p>Resolution: 16-bit</p> <p>Type (B) provides more angular velocity range option</p>	Digital-output X-, Y-, and Z-axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ± 500 dps, ± 1000 dps, ± 2000 dps, and ± 4000 dps and integrated 16-bit ADCs, digitally-programmable low-pass filter, factory calibrated sensitivity scale factor and uses Self-test
Accelerometer	<p>Ranging: $\pm 2/4/8/16$ g</p> <p>Resolution: 16-bit</p>	Digital-output X-, Y-, and Z-axis accelerometer with a programmable full-scale range of ± 4 g, ± 8 g, ± 16 g, and ± 30 g and integrated 16-bit ADCs. User-programmable interrupts, Wake-on-motion interrupt for low power operation of applications processor and Self-test
DMP Features	The DMP enables ultra-low power run-time and background calibration of the accelerometer, gyroscope, and compass, maintaining optimal performance of the sensor data for both physical and virtual sensors generated through sensor fusion. This enables the best user experience for all sensor enabled applications for the lifetime of the device	<p>The DMP can be used to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in applications. Optimized for Android Lollipop for low power features (AP suspended) including SMD, Step Count, Step Detect, Activity Classification, Rotation Vector, and Gaming Rotation Vector</p> <p>The DMP will also batch data from externally connected sensors such as a compass, or pressure sensor.</p>

		The DMP enables ultra-low power run-time and background calibration of the accelerometer, gyroscope, and compass, maintaining optimal performance of the sensor data for both physical and virtual sensors generated through sensor fusion. This enables the best user experience for all sensor enabled applications for the lifetime of the device
Temperature & Humidity	<p>Accuracy (Humidity): $\pm 2\%$ rH</p> <p>Ranging (Humidity): 0% ~ 100% rH</p> <p>Accuracy (Temp): $\pm 0.2^{\circ}\text{C}$</p> <p>Ranging (Temp): $-30 \sim 100^{\circ}\text{C}$</p> <p>Type (B) has higher precision and wider range</p>	Sensitivity Scale Factor Tolerance 25°C $\pm 0.5\%$, 3 Sensitivity Scale Factor Variation Over Temperature -40°C to $+85^{\circ}\text{C}$ $\pm 2\%$ 2 Nonlinearity Best fit straight line; 25°C
Magnetometer	<p>Ranging: ± 49 gauss</p> <p>Resolution: 16-bit</p>	Ranging: ± 30 gauss
Barometer	<p>Ranging: 260 ~ 1260 hPa</p> <p>Accuracy (ordinary temperature): ± 0.025hPa</p> <p>Speed: 1 Hz - 75 Hz</p> <p>Type (B) measures the barometric pressure more precise and faster</p>	None
Misc	<p>Color Sensor</p> <p>High precision 12-bit ADC</p> <p>Type (B) features color sensor, and supports more external AD sensors</p>	MEMS structure hermetically sealed and bonded at wafer level • RoHS and Green compliant
Communication Interface	I2C interface	The ICM-20649 communicates to a system processor using either a SPI or

		an I2C serial interface
--	--	-------------------------

Similarities Between old IMU Data and IMU Sense Hat B:

Feature	IMU Data (Supplied)	IMU Sense Hat B
Time	The IMU Data consist of the date, day and time which data was fetched.	Build in code is written to keep track of the time data is fetched.
Data Captured	3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer	3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer
Sensitivity to Movements	The IMU Sense that requires continuously “moving” of device to detect movement, orientation and magnetic	The IMU Sense that requires continuously “moving” of device to detect movement, orientation and magnetic
Sensitivity to External Environment	The IMU data captured are affected by external environments, such as temperature, humidity, pressure which may affect the values of the data	The IMU data captured are affected by external environments, such as temperature, humidity, pressure which may affect the values of the data

Experiment Setup

Overall system functionality: Raspberry pi including IMU Sense Hat

Test 1: Overall Execution Time of System

Instead of simulated data, the Sense Hat (B) containing the ICM20948 IMU is used. Data is sampled straight from the sensor in batches of 100 and written to a file to be later compressed and encrypted. To check the overall functionality of the system, raw data files (output files from the IMU block) are fed into the processing block and read by the compression subsystem. The compression subsystem outputs a zipped file that the encryption subsystem will then encrypt, thus outputting an encrypted, compressed file.

The execution time for the processing of the file will be recorded, along with the initial and final file sizes. These tests will be run several times with different file inputs, each with either a different sampling rate or noise added.

Fourier coefficients of the data will be calculated using the original raw data, and the data from the decrypted, and decompressed file. These two Fourier samples will be compared and the mean squared error will be calculated

Input	Expected Output
Record time taken to read samples from IMU	Ideal, not longer than a minute.
Calculate total time taken to execute the parts of the program that require heavy processing	No more than 60 % of total time for the whole system

Table: Test for System

Test 2: Measure the Power of the System

Measure the overall power of the system.

Input	Expected Output
Measure the Voltage and Current, and calculate Power.	Not too high power, as this might affect the functionality of the device.

Table: Test for System

Test 3: Measure the Temperature of Both Systems

It is important when working on a device that the temperature is not too high and it does not overheat the system, as this might affect the accuracy of the data.

Input	Expected Output
Measure the temperature of the system before and after executing program: Raspberry pi and IMU Sense Hat	A bit of heating is expected

Table: Test for System

Test 4: “Stress” testing Raspberry pi & IMU Sense Hat

Stress testing is simply running a series of processes on your system which are designed to run the CPU at full power, and monitor the temperature and stability of the system.

80 degrees is normally the high-temperature limit for most computer systems, so we need to run a stress test to make sure that our cooling is adequate to keep it under 80C. The hotter a processor gets, then less stable it becomes, so a cooler processor will lead to a more stable system

So, the following steps should be implemented:

1. Implement Cooling Methods
2. Run Stress Test and ensure System is stable and cool for 10 mins of run time
3. Run cpuburn-a53 and ensure system is stable and cool for 10 mins of run time
4. Push the clock speed and overclock settings until the system is no longer stable (crash or freeze), then dial overclocking back to the last successful settings
5. If the system is running too hot, consider adding extra cooling capabilities before trying again.

Experiment Design: Compression

A python program is created to test for the different criteria that need to be met. These tests are driven by the compression ratio, speed and integrity requirements previously specified. Several algorithms are compared to ensure that the most efficient compression algorithm is chosen. For each data set, 10 iterations are run and the average of these results is taken.

Test 1: Compression Ratio

A sample file (data sampled from IMU) is fed into the program, and the program is run in a loop. The input file size is recorded (in bytes). Next, for each algorithm chosen for comparison: zlib, gzip and lzma, the same data file is compressed and written to an output file. The size in bytes of the contents of this output file is also recorded. These two are compared and the compression ratio, along with the % reduction are calculated (IBM Corporation, 2021) and recorded:

$$\% \text{ reduction} = \left(1 - \frac{\text{compressed size}}{\text{original size}}\right) * 100$$

Test 2: Speed of Algorithm

During the same experiment, the time before and after each compression and decompression is recorded. The average of all 10 readings from the compression and decompression is recorded and these values are compared.

Test 3: Data integrity

Finally, once the output data has been decompressed, the contents of the original file and the decompressed file are compared in the python program to check whether any loss has occurred.

The results of these tests will determine which algorithm will be used for the system, taking into consideration a time-size trade-off.

Experiment Design: Encryption

Test 1: Data Check

Test whether the algorithm has the correct encryption code, hence when the IMU sensor data is supplied, an encrypted data needs to be output.

Input	Expected Output
The compressed data of the IMU sensor, i.e., the accelerometer, gyroscope, etc	An encrypted text, which differs from the data sensor input

Table: Test for System

Test 2: Algorithm Speed Complexity

Test the time each algorithm takes to compile the data from the IMU sensor into the appropriate encrypted text.

Input	Expected Output
Time the algorithm.	Algorithm speed needs to be faster than the average performance.

Table: Test for System

Test 3: Correctness of Data

Test the correctness of the Data. Once the data has been encrypted, the data should be decrypted back to the “original” IMU sensor data. The original data should be comparable to the decrypted data. This experiment tests for the correctness of the decrypted algorithm. How close is the data to the original? Was there any data that was lost?

Input	Expected Output
Decrypt the encrypted text	Data needs to coincide with the original imu sensor data

Table: Test for System

Test 4: Data Loss

How much data is lost. Once the data has been decrypted, calculate the data percentage that is loss, using the formula:

$$Data\ Loss = \frac{Data\ Loss}{Original\ Data} \times 100$$

Input	Expected Output
Decrypt the encrypted text	Calculate the data loss percentage, hence needs to be as low as possible

Table: Test for System

Test 5 : Strength of Algorithm

Every algorithm has it's weak and strong points, this test is to test how easily the algorithm breaks, this test will check whether the data can be decrypted if the user does not have the correct "Key" to decrypt.

Input	Expected Output
Attempt to decrypt the encrypted text, with an invalid key.	A valid message, stating that the user cannot decrypt the text file without the key.

Table: Test for System

Test 6 : Size of Encrypted and decrypted data

When data gets encrypted, it's size changes. The aim of this experiment is to study how exactly the size has changed.

Input	Expected Output
Measure the original size of data and also	The size of encrypted data needs to be

measure the size of the encrypted and decrypted data	<p>equal to the original data size, hence means no data is lost.</p> <p>The size of the encrypted file needs to be greater than the original data size.</p>
--	---

Table: Test for System

Results

Results: Overall system functionality

Test 1 : Overall Execution Time of System

The execution time is measured for reading the IMU data through use of a program run on the pi. The results are as follows

```

pi@berthasrpi:~/Documents/EEE3097S $ vim ICM20948.py
pi@berthasrpi:~/Documents/EEE3097S $ python3 ICM20948.py

Collecting data from the IMU

Time taken to read 100 sample: 33.84s

```

Figure 30: Readings from IM

As shown above, time taken to sample data from the IMU is on average 33.84s for 100 samples. This results in a sampling frequency of about 2.955 Hz.

A test of the whole system is carried out after both subsystems have been compiled.

```

pi@berthasrpi:~/Documents/EEE3097S $ python3 system.py
Enter the filename to open:
IMUdata.csv
output/sampleOut_.csv.gzip
Done compressing
now encrypting
Successfully Encrypted!
Decryption private key : amtkx3CpQdINJhOP940X
Time taken to compress and encrypt: 0.44s
Would you like to access your file now?
yes
Enter the private key:
amtkx3CpQdINJhOP940X
Successfully Decrypted!
Compressed and decompressed data the same? True

```

Figure 31 : Sample Output

The overall time taken to complete the compression and encryption on a pi for the given data sample from the IMU data is 0.44s. After decompression and decryption, the file contents are compared to the contents of the input IMU data file.

As expected, the output of the system block shows no loss, and thus the root mean square of the error in the Fourier coefficients of the input file and output decompressed file is 0. This however, does not take into account the noise that may be picked up from transmission.

Thus, it can be concluded that the overall algorithm is effective as all the compressions ATP tests and encryption ATP tests were met as shown in table 12

Test 2: Measure the Power of the System

Voltage: Approximately 0.64 Volts

Maximum Current of IMU Sense Hat P: 16mA

Hence, max Power: $P = V \cdot I$

$$P = 0.64 \cdot 16\text{mA}$$

$$P = 0.01024 \text{ W}$$

Power is ideal, not too high.

Test 3: Measure the Temperature and Pressure of Both Systems (Before and After Execution)

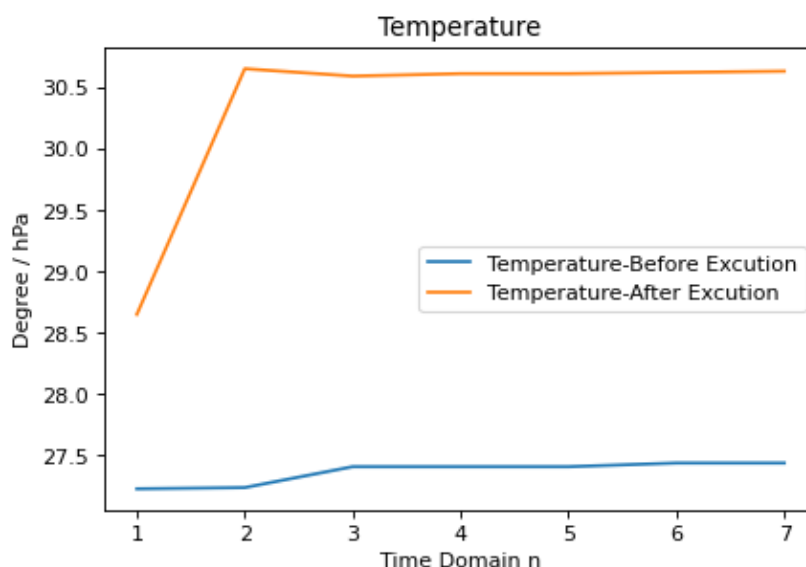


Figure 32: Readings from IMU

Pressure:

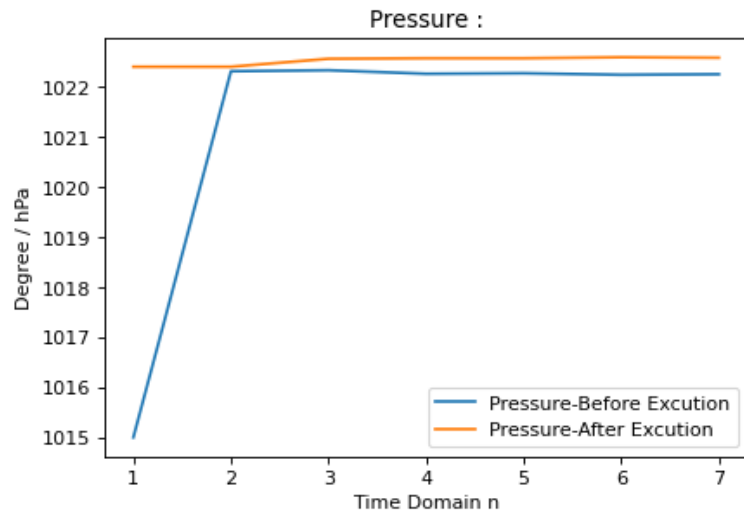


Figure 33: Readings from IMU

As expected, the temperature and pressure increased but not greatly.

Test 4: “Stress” testing Raspberry pi & IMU Sense Hat

```

pi@raspberrypi:~/SenseHat $ while true; do vcgencmd measure_clock arm; vcgencmd
measure_temp; sleep 10; done & stress -c 4 -t 900s
[1] 1137
stress: info: [1138] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
frequency(48)=1000104000
temp=36.3'C
frequency(48)=1000000000
temp=39.0'C
^C

```

Figure 34: Readings from IMU

Device passed the stress test as shown in above figure.

Results: Compression

The compression tests produced the results conveyed in the screenshot of the text file shown below:

```
output/sampleOut_.csv.gzip
Read 1615 words, 9392 bytes.
  2832 output/sampleOut_gzip.csv.gzip
Average compression time: 0.00655217170715332
Compression ratio: 3 with reduction of 69.85%
Average decompression time: 0.002297663688659668
  2854 output/sampleOut_zlib.csv.zlib
Average compression time: 0.005796802043914795
Compression ratio: 3 with reduction of 69.61%
Average decompression time: 0.001373136043548584
  2480 output/sampleOut_lzma.csv.lzma
Average compression time: 0.031412291526794436
Compression ratio: 3 with reduction of 73.59%
Average decompression time: 0.0018930673599243165

The size winner is: lzma
The time winner is: zlib
pi@benharpi: ~/Documents/55520076 $
```

Figure 35: Readings from IMU

As shown in the figure above, using the lzma library produced the slowest compression time, but had the largest compression ratio. This compression takes on average 0.3s to compress about 9.4kb and reduces the file size by up to 73%. The compression ratio is 3.

The gzip and zlib libraries produce similar results, at half the time that the lzma algorithm takes. The compression ratio for both these algorithms is also 3 for the given sample set, with the difference in reduction percentages being about 0.14%.

As when the sampling rate is changed, the time taken changes accordingly.

```
output/sampleOut_lzma.csv.lzma
output/sampleOut_zlib.csv.zlib
output/sampleOut_gzip.csv.gzip
Read 5197 words, 29736 bytes.
  7636 output/sampleOut_gzip.csv.gzip
Average compression time: 0.05080482959747314
Compression ratio: 3 with reduction of 74.32%
Average decompression time: 0.0038064956665039063
  7769 output/sampleOut_zlib.csv.zlib
Average compression time: 0.03702062368392944
Compression ratio: 3 with reduction of 73.87%
Average decompression time: 0.002476620674133301
  6372 output/sampleOut_lzma.csv.lzma
Average compression time: 0.08172096411387125
Compression ratio: 4 with reduction of 78.57%
Average decompression time: 0.0037940820058186848

The size winner is: lzma
The time winner is: zlib
```

Figure 36: Readings from IMU

For each of the algorithms tested, the time it takes to compress and decompress was doubled when the sampling frequency was doubled. This corresponds to the size of the data file which will also increase. The size of the file did however triple in size (9.4kB to 29.7kB). As shown in the results above, the compression ratio increases slightly for each of the algorithms.

It is of note to observe that when the sample size is small, the compression ratio is 3 for all the algorithms. The time taken is also considerably much smaller (by an order of 10) than for a system with larger samples. This is more favourable for the use case as it allows for more battery power, and a greater spread of range of times when the data is being read.

Conclusions:

A compromise of speed and time need to be taken into consideration, along with ensuring maximum data integrity. As a result, the gzip library will be used for the design. This is as proposed in the initial design where the gzip library was preferred because of the CR32 - checksum carried out. This algorithm is preferred as the higher compression will result in less storage space being taken when large amounts of data are required. This is a constraint that our design will have to satisfy.

Results: Encryption

For the encryption techniques, four algorithms will be tested and compared.

Test 1: Data check

The first experiment checks whether the encrypted data matches the expected data .

Algorithm	Data Check
Algorithm 1: RSA Algorithm	The encryption algorithm works only if the IMU data received per line is less than 53 bytes, otherwise data larger than 53 bytes will not be properly encrypted.
Algorithm 2: Base64 Strings	The encryption works effectively, all the data has been encrypted correctly.
Algorithm 3 :Cryptocode	The encryption works effectively, all the data has been encrypted correctly however the algorithm is very slow.
Algorithm 4: Cryptography Fernet	The encryption works effectively, all the data has been encrypted correctly.

Table: Test for System

As shown from the above table , all algorithms match the expected data of their particular encryption technique, expected for the RSA Algorithm which won't encrypt data that are larger than 53 bytes.

Test 2 : Algorithm Speed Complexity

This is to test the time complexity of the algorithms, exactly how fast the algorithm executes.

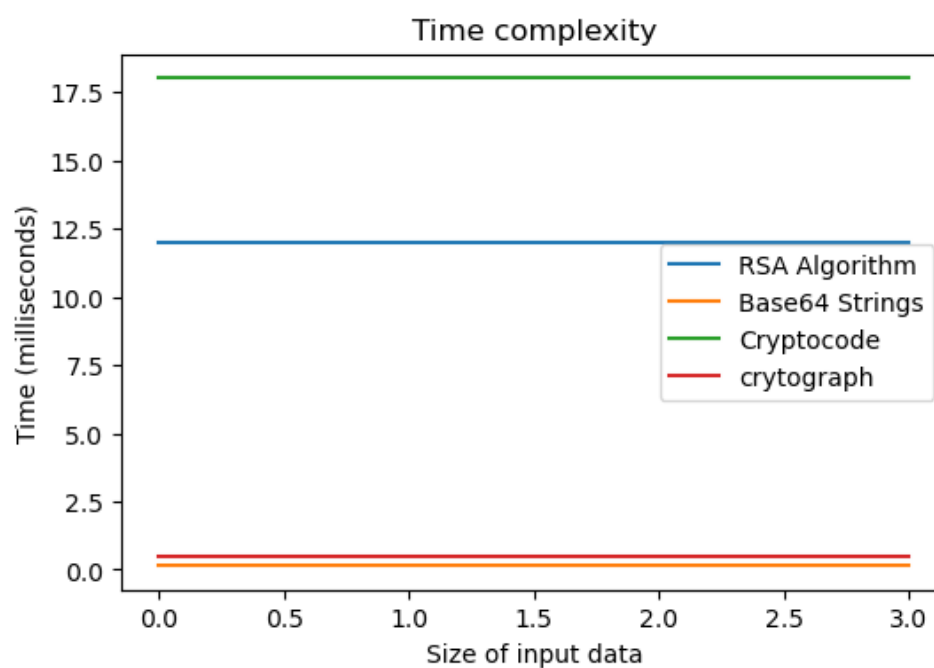


Figure 37: Speed of Algorithm

As shown in figure 37 the RSA and Cryptocode Graph have the longest time complexity, whereas the Cartography and Base64 Graph have the shortest time complexity of less than 2.5 milliseconds.

Test 3 : Correctness of Data

This is used to ensure that the decrypted data matches the original IMU data.

Algorithm	Correctness of Data:
Algorithm 1 : RSA Algorithm	The decrypted data contains the same exact data as the original text, however the format (e.g new line, paragraph) is lost.
Algorithm 2 : Base64 Strings	The decrypted data is equivalent to the original data, hence no data is lost
Algorithm 3 :Cryptocode	The decrypted data is equivalent to the original data, hence no data is lost
Algorithm 4 : Cryptography Fernet	The decrypted data is equivalent to the original data, hence no data is lost

Table: Test for System

Test 4: Data Loss

This calculates the percentage of data that was lost from test 3, using the formula:

$$Data\ Loss = \frac{Data\ Loss}{Original\ Data} \times 100$$

Algorithm	Data loss (Percentage)
Algorithm 1 : RSA Algorithm	20% (Format of data is lost)
Algorithm 2 : Base64 Strings	0%
Algorithm 3 :Cryptocode	0%

Algorithm 4 : Cryptography Fernet	0%
-----------------------------------	----

Table: Test for System

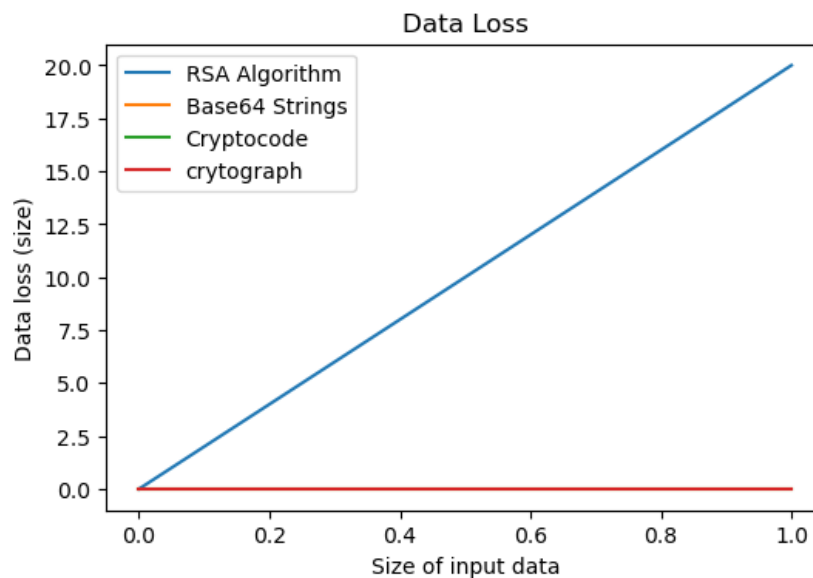


Figure 38: Percentage of Data Loss

As shown in figure 5 and table 9, the RSA algorithm is the only algorithm where their data is lost as the size of the input increases. The other algorithms are able to maintain a 0% data loss with an increase in data size.

Test 5: Strength of Algorithm

This is to test how algorithms behave with unexpected behaviour. ie Trying to decrypt the data with an invalid key.

Algorithm	How do algorithms react with Invalid keys?
Algorithm 1 : RSA Algorithm	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided. Hence, an expectation error is thrown.
Algorithm 2 : Base64 Strings	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided. Hence, an expectation error is thrown.

Algorithm 3 :Cryptocode	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided. Hence, an expectation error is thrown.
Algorithm 4 : Cryptography Fernet	The algorithm does not allow the user to access or decrypt the data if the correct key is not provided. Hence, an expectation error is thrown.

Table: Test for System

Test 6: Size of Encrypted and decrypted data

This experiment is used to compare the size of the encrypted and decrypted file to the original file. The original file size is 8874 KB.

Algorithm	Encryption Size	Decryption Size
Algorithm 1 : RSA Algorithm	56KB	2000KB
Algorithm 2 : Base64 Strings	23648KB	8874 KB
Algorithm 3 :Cryptocode	18645KB	1143KB
Algorithm 4 : Cryptography Fernet	1245KB	8874 KB

Table: Test for System

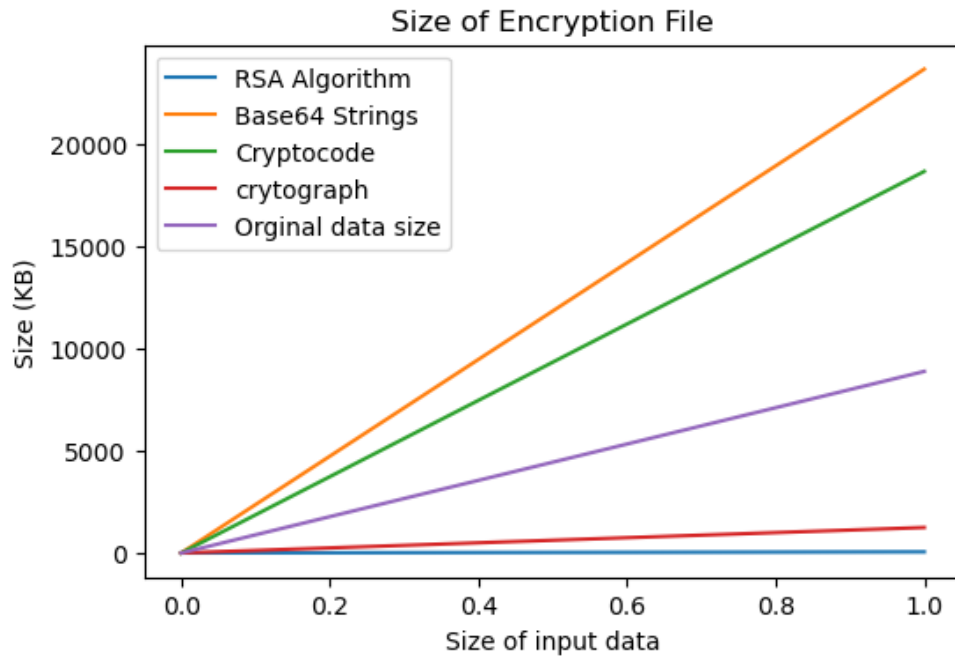


Figure 39: Size of Data

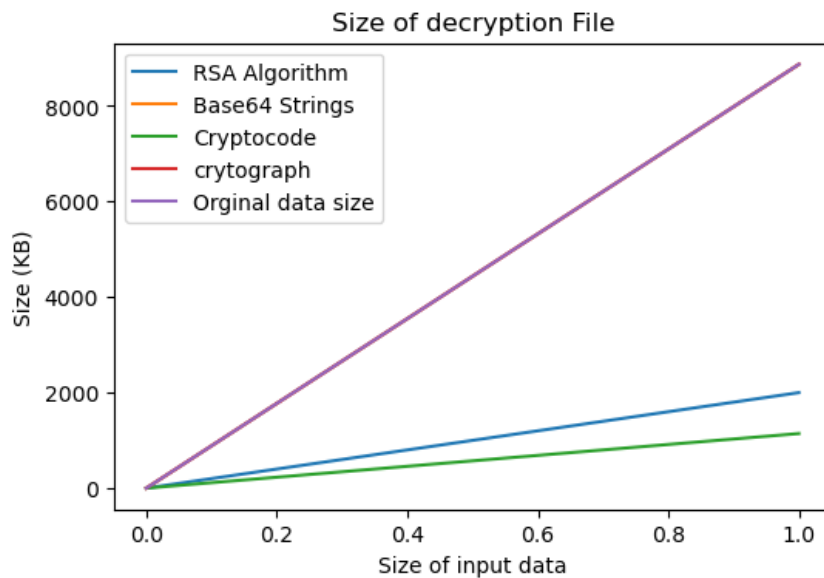


Figure 40: Size of Data

For the encryption data size, the size of the algorithm needs to be greater than the original non -encrypted data size, as this shows that an effective encrypted algorithm has been implemented with major reductions. A smaller encrypted size file means that less minor reductions have been adjusted to the original text. In Figure 4, it is clearly shown that the Base64 Strings algorithm and the cryptograph algorithm are effective, due to its encrypted data being bigger, meaning more random and additional headers have been implemented in the algorithm.

For the decryption data size, the size of the algorithm needs to coincide with the original data size. This will mean that no data has been lost. If a decryption data size is smaller than the original, it means some data have not recovered from the encryption.

Conclusions:

From the finding of the results, it is decided that the Base64 String Algorithm will be implemented to decrypt and encrypt our IMU sensor data. This is because it has the best time complexity, with no data loss.

6.Consolidation of ATPs and Future Plan

ATP Number	Related Spec ID	Acceptance Test
1	CS001, CS002, CS006	Total sampling and compression time are consistent in proportion to the original dataset size and does not exceed the max time specified (no more than 1minute)
2	CS003	File of sampled data before encryption compared against original data and has at least 25% of its original lower Fourier coefficients
3	CS003	Size of compressed file less than specified max size per file depending on the number of data files on Raspberry Pi 0 SD card.
4	CS004	Compression ratio of file greater than 2
5	CS005	After gzip CRC-32 checksum, decompressed data file produces same results as original uncompressed file
6	SE001	No data loss after encrypted file has been decrypted
7	SE002	Timing the algorithm , and obtaining a faster time complexity algorithm to encrypt & decrypt data.Needs to be faster than the average time complexity of a simple encryption algorithm, i.e 60 seconds
8	SE003	The encrypted text needs to be unique & different the original data

9	SE004	The size of the decrypted file needs to be equivalent to the original text, hence equal to 8874 KB.
10	SE005	The size of the encrypted text needs to be larger than the original IMU data, hence larger than 8878 KB
11	SE006	A valid message needs to be outputted when the user tries to decrypt the data with an invalid key.

Table 12: ATP Tests

Results of ATP testing

ATP	Criteria status	Comments
1	met	Data read from IMU sensor in under a minute
2	met	No error/loss recorded
3	met	Raspberry pi 0 will run for a max of about 12 hours on battery power. Considering that the current storage on the pi being used is at least 50GB, there will be enough space to store compressed files for the max duration of the pi being on.
4	met	Reduction percentage = 69.85% (increased)
5	met	Input and output files have the same contents
6	met	0% data loss
7	met	Algorithm takes 0.1607 seconds to encrypt and decrypt the data
8	met	The encrypted data has its own unique code, which is different from the original data
9	met	The size of the decrypted data is 8874 KB.
10	met	The size of the encrypted data size is 23648 KB

11	met	An appropriate message is output when a user tries to decrypt the data without the private key.
----	-----	---

Table 13: ATP Test Results

Future Dev Work before IP can be used:

The time for the development of this design was limited and thus, there were some features that were not implemented. And some of which were not implemented as thoroughly.

More extensive research and analysis will need to be done to examine power consumption when the programme in the IP is running. Further, although this design did factor in some noise, the design was more focused on ensuring the correct results are obtained. To obtain a more realistic view of the output expected, noise-simulated data at each subsystem input will need to be included.

The foundation for the security has been set with the encryption subsystem. However, the IP design will need to be refined such that the encryption key is transmitted securely. A transmission block (not developed in this design) will aid in this regard, and will ensure the data is formatted in the right way, and sent to the correct location.

Lastly, automation of the reading of data, processing and passing it onto the system to be compressed and encrypted is still required. This will also need to be carried out at set intervals.

7. Conclusion

A compromise of speed and time need to be taken into consideration, along with ensuring maximum data integrity. As a result, the gzip library will be used for the design. This is as proposed in the initial design where the gzip library was preferred because of the CR32 checksum carried out. This algorithm is preferred as the higher compression will result in less storage space being taken when large amounts of data are required.

From the finding of the results, it is decided that the Base64 String Algorithm will be implemented to decrypt and encrypt our IMU sensor data. This is because it has the best time complexity, with no data loss.

Although not having the exact IMU sensor that was provided, the development of the design could be done through use of old IMU data that was collected beforehand, along with IMU data from a different model. Both sample data sets showed good results and the initial ATP Tests were met.

The design can be built upon and used as a skeleton to further explore how a similar IP design can be implemented to collect data from the SHARC Buoy.

8. References

- [1] UCT News, "Polar cyclones, Antarctic sea ice and a cruise to understand it all," University of Cape Town, 15 July 2019. [Online]. Available: <https://www.news.uct.ac.za/article/embed/js/-2019-07-15-polar-cyclones-antarctic-sea-ice-and-a-cruise-to-understand-it-all/>. [Accessed 31 August 2021].
- [2] C. T. Y. Collet, "Smaller and faster data compression with Zstandard," 30 August 2016. [Online]. Available: <https://engineering.fb.com/2016/08/31/core-data/smaller-and-faster-data-compression-with-zstandard/>. [Accessed 31 August 2021].
- [3] "Compression: Clearing the Confusion on ZIP, GZIP, Zlib and DEFLATE.," Dev Community, 23 October 2019. [Online]. Available: <https://dev.to/biellls/compression-clearing-the-confusion-on-zip-gzip-zlib-and-deflate-15g1>. [Accessed 31 August 2021].
- [4] InfoSec Insights, "What is Asymmetric Encryption & How Does it Work?," Sectigo Store, 3 November 2020. [Online]. Available: <https://sectigostore.com/blog/what-is-asymmetric-encryption-how-does-it-work/>. [Accessed 30 August 2021].
- [5] ResearchGate, "Development of Advanced Encryption Standard (AES) Cryptography Algorithm for Wi-Fi Security Protocol," [Online]. Available: https://www.researchgate.net/figure/Types-of-Cryptography-Various-Cryptographic-Algorithms-a-Data-Encryption-Standard_fig4_323369289. [Accessed 31 August 2021].
- [6] Wikipedia, "Cryptographic hash function," 2020. [Online]. Available: https://en.wikipedia.org/wiki/Cryptographic_hash_function. [Accessed 31 August 2021].
- [7] IBM Corporation, "Data Compression," IBM Corporation, 2021. [Online]. Available: from <https://www.ibm.com/docs/en/spectrum-symphony/7.3.0?topic=performance-data-compression>. [Accessed 2 October 2021].
- [8] Wikipedia, "RSA (cryptosystem)," Wikipedia, 1 October 2021. [Online]. Available: [https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)#Encryption](https://en.wikipedia.org/wiki/RSA_(cryptosystem)#Encryption). [Accessed 3 October 2021].
- [9] Hakoja, "StackExchange: How do AES and DES ensure data integrity?," 2020. [Online]. Available: <https://crypto.stackexchange.com/questions/79954/how-do-aes-and-des-ensure-data-integrity>.
- [10] Python Software Foundation, "gzip - Support for gzip files," Python Software Foundation, 2021. [Online]. Available: <https://docs.python.org/3.7/library/gzip.html?highlight=gzip#gzip.GzipFile>. [Accessed 2 October 2021].

- [11] J. Jacobson, "SHARC BUOY: Remote Monitoring of Ice Floes in SOuthern Ocean [MSc Dissertation Paper - intro]," University of Cape Town, Cape Town, 2020.