University of Cape Town

Department of Electrical Engineering

Electrical and Computer Engineering

EEE3097S: Engineering Design

Compression and Encryption blocks for IMU generated data

FINAL REPORT

H.G Ngwenya and M.S Mwandla

NGWHOP001, MWNMSI001

1. <u>Admin Documents</u>

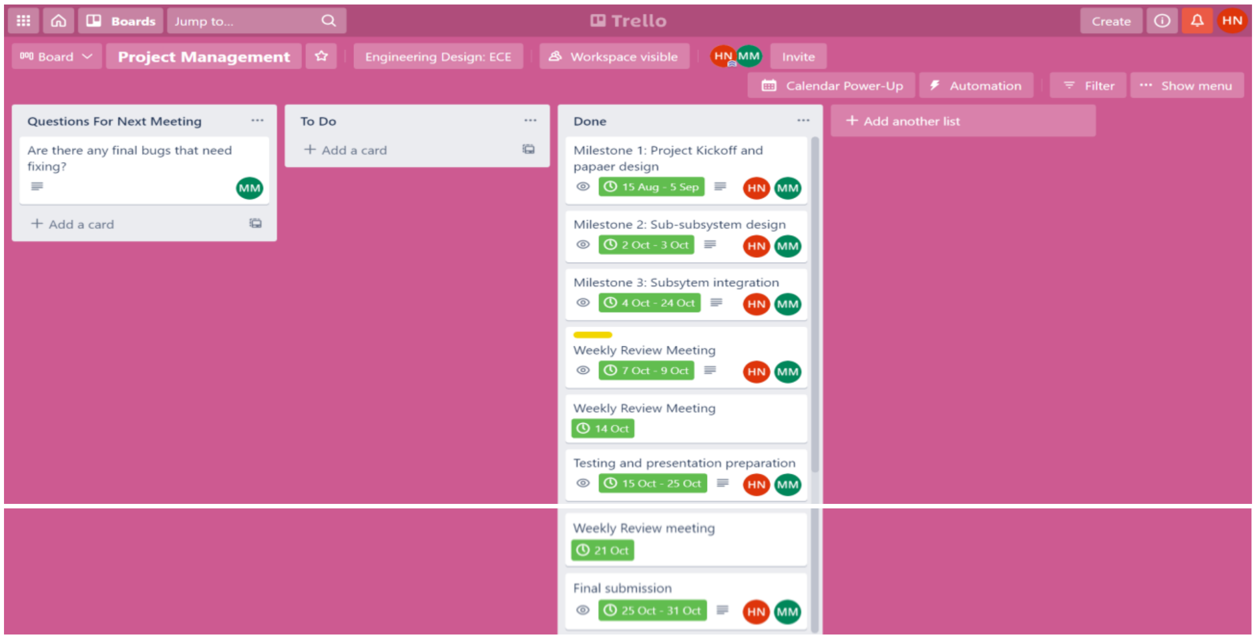
a. Team member contribution

Task	Group member	Section Number	Page number
Project administration	Hope and Msimamisi	1	2
Requirement analysis	Msimamisi	2	4
Paper Design	Hope	3	10
Design validation using old data	Msimamisi	4	12
Design validation using the IMU module	Hope	5	16
Consolidation of ATPs and future plan	Msimamisi	6	24
Conclusion	Hope	7	25

Table 1: Contribution table

b. Project management tool

The Trello project management board is shown in *Figure 1 and the project calendar is also shown in Figure 2* below:



Questions For Next Meeting

Are there any final bugs that need fixing? MM

To Do

Done

Milestone 1: Project Kickoff and papaer design — 15 Aug - 5 Sep — HN MM

Milestone 2: Sub-subsystem design — 2 Oct - 3 Oct — HN MM

Milestone 3: Subsytem integration — 4 Oct - 24 Oct — HN MM

Weekly Review Meeting — 7 Oct - 9 Oct — HN MM

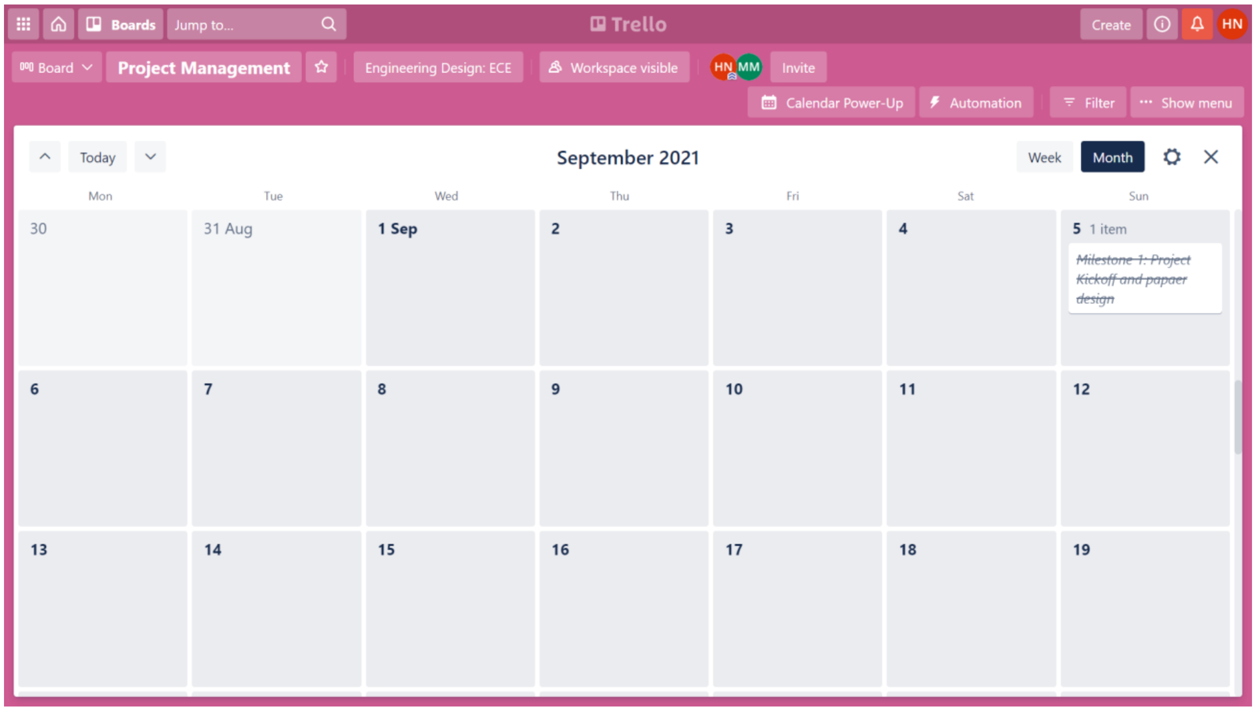
Weekly Review Meeting — 14 Oct

Testing and presentation preparation — 15 Oct - 25 Oct — HN MM

Weekly Review meeting — 21 Oct

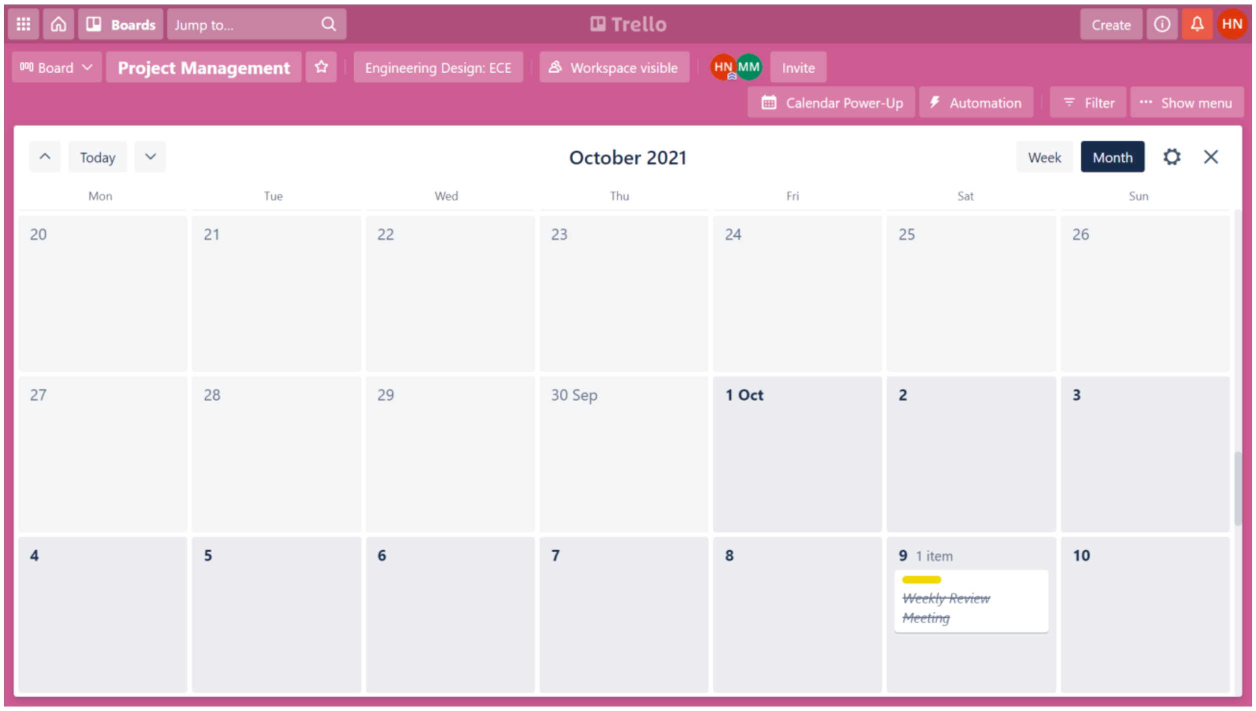
Final submission — 25 Oct - 31 Oct — HN MM

Figure 1: Trello project management board



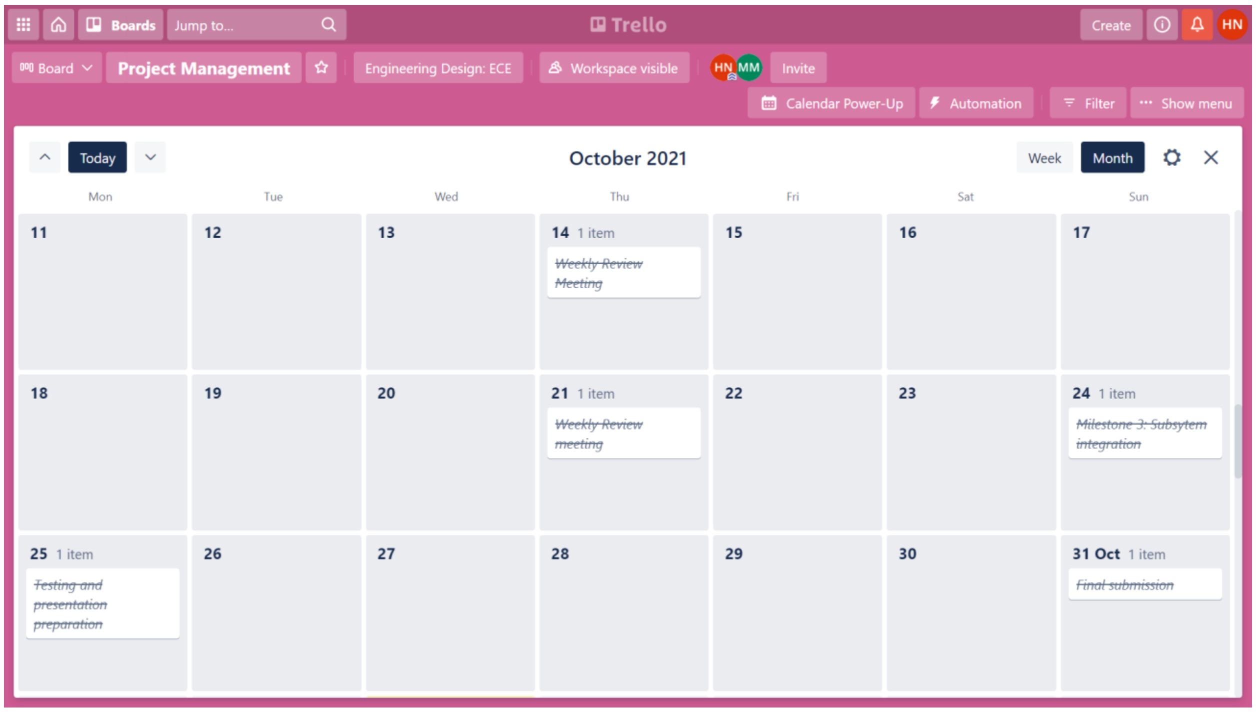
September 2021

Mon	Tue	Wed	Thu	Fri	Sat	Sun
30	31 Aug	1 Sep	2	3	4	5 1 item ~~Milestone 1: Project Kickoff and papaer design~~
6	7	8	9	10	11	12
13	14	15	16	17	18	19



October 2021

Mon	Tue	Wed	Thu	Fri	Sat	Sun
20	21	22	23	24	25	26
27	28	29	30 Sep	1 Oct	2	3
4	5	6	7	8	9 1 item ~~Weekly Review Meeting~~	10



Mon	Tue	Wed	Thu	Fri	Sat	Sun
11	12	13	14 1 item ~~Weekly Review Meeting~~	15	16	17
18	19	20	21 1 item ~~Weekly Review meeting~~	22	23	24 1 item ~~Milestone 3: Subsytem integration~~
25 1 item ~~Testing and presentation preparation~~	26	27	28	29	30	31 Oct 1 item ~~Final submission~~

Figure 2 : Trello project calendar

The project management tool used can be found here .

c. Git repository:

The GitLab project repository can be found here.

2. Requirement Analysis

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

Since the ICM-20649 was unavailable, the Waveshare sense HAT [1] was used. The sense is a board that has multiple sensors including sensors with IMU capabilities. The advantages and disadvantages of using this device as a suitable measure of the behavior of the intended component were considered.

Advantages :

- The sense HAT can be interfaced with the Raspberry Pi like the required device.
- The sense HAT has IMU capabilities as the required device.
- The sense HAT supports the I2C communication protocol like the required component.

Disadvantages:

- Unlike the required IMU, the Sense HAT does not support the SPI communication protocol. This means that the scope of communication of the sense HAT is limited compared to the required IMU.
- The required IMU and the sense HAT are manufactured by different companies therefore their operating standards may differ.
- The ADC the Sense HAT uses has a 12-bit resolution, while the ICM-20649's on-chip ADC has a resolution of 16 bits.The Sense HAT has a lower resolution than the required IMU.
- The main requirement is a device that has IMU functionality which at least supports the SPI or I2C protocol. The WaveShare HAT provides for this requirement.

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

This requirement has it that if you take the Fourier transform off the data and analyse the Fourier coefficients, 25% of all Fourier confessions from the original dataset should be maintained after compression.

The chosen compression algorithm is Huffman encoding. This algorithm is classified as no loss compression algorithm. This means that the integrity of the original data is kept even after compression has been achieved. As a result, all of the Fourier coefficients will be retained. The compression algorithm satisfies this requirement.

Spec: Algorithm must achieve lossless decryption

ATP: Algorithm returns original data that is readable.

c. **Req3**. In addition to reducing the amount of data we also want to reduce the amount of processing done in the processor (as it takes up power which is limited).

The Raspberry Pi is a low specification computational device with limited memory and power. In the remote environment in which this system will be executed, generating power is not the easiest thing to do therefore power must be conserved during program execution.

Spec3:

The chosen encryption and compression algorithm must be primarily chosen because it is the fastest algorithm to execute. Quicker execution times mean less time spent on computation therefore less power used during computation. This watch is the requirement of reducing the power consumed during program execution.

ATP: Required execution time must be below 30 seconds

ATP: RAM usage must be less than 25%

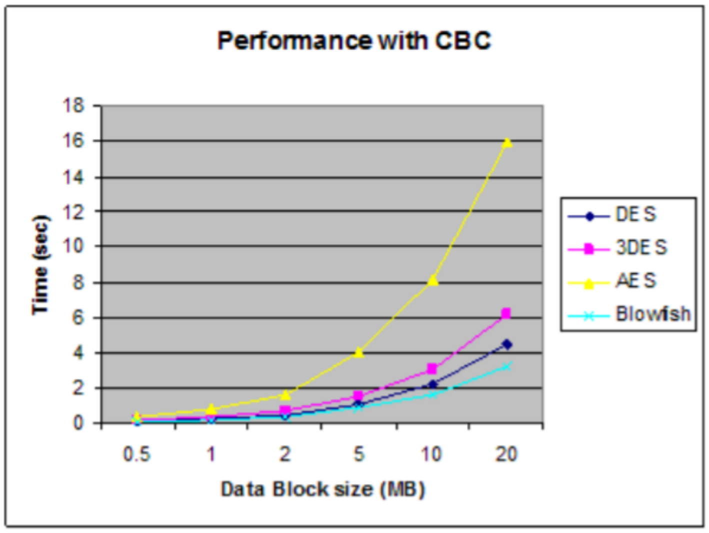
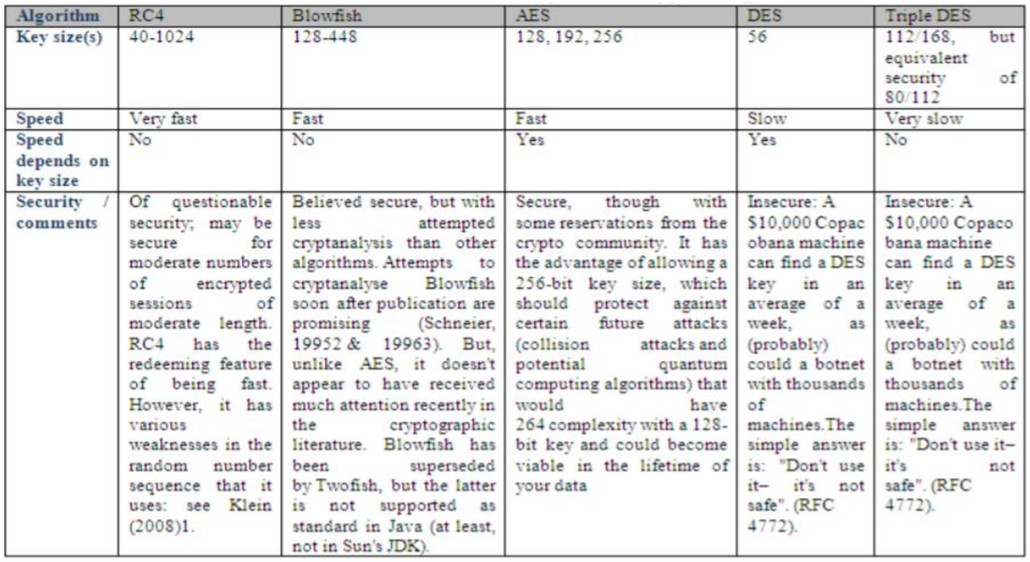


Figure 3: Comparison between encryption algorithms

Adopted from a study by Dr. Kiramat ullah, Bibi Ayisha, Farrukh Irfan, Inaam Illahi and Zeeshan Tahir. [3]



Algorithm	RC4	Blowfish	AES	DES	Triple DES
Key size(s)	40-1024	128-448	128, 192, 256	56	112/168, but equivalent security of 80/112
Speed	Very fast	Fast	Fast	Slow	Very slow
Speed depends on key size	No	No	Yes	Yes	No
Security / comments	Of questionable security; may be secure for moderate numbers of encrypted sessions of moderate length. RC4 has the redeeming feature of being fast. However, it has various weaknesses in the random number sequence that it uses: see Klein (2008)1.	Believed secure, but with less attempted cryptanalysis than other algorithms. Attempts to cryptanalyse Blowfish soon after publication are promising (Schneier, 19952 & 19963). But, unlike AES, it doesn't appear to have received much attention recently in the cryptographic literature. Blowfish has been superseded by Twofish, but the latter is not supported as standard in Java (at least, not in Sun's JDK).	Secure, though with some reservations from the crypto community. It has the advantage of allowing a 256-bit key size, which should protect against certain future attacks (collision attacks and potential quantum computing algorithms) that would have 264 complexity with a 128-bit key and could become viable in the lifetime of your data	Insecure: A $10,000 Copac obana machine can find a DES key in an average of a week, as (probably) could a botnet with thousands of machines.The simple answer is: "Don't use it– it's not safe". (RFC 4772).	Insecure: A $10,000 Copaco bana machine can find a DES key in an average of a week, as (probably) could a botnet with thousands of machines.The simple answer is: "Don't use it– it's not safe". (RFC 4772).

Table 2: Encryption algorithms performance comparison

Adopted from a study by Soheila Omer AL Faroog Mohammed Koko, Dr.Amin Babiker A/Nabi Mustafa.[2]

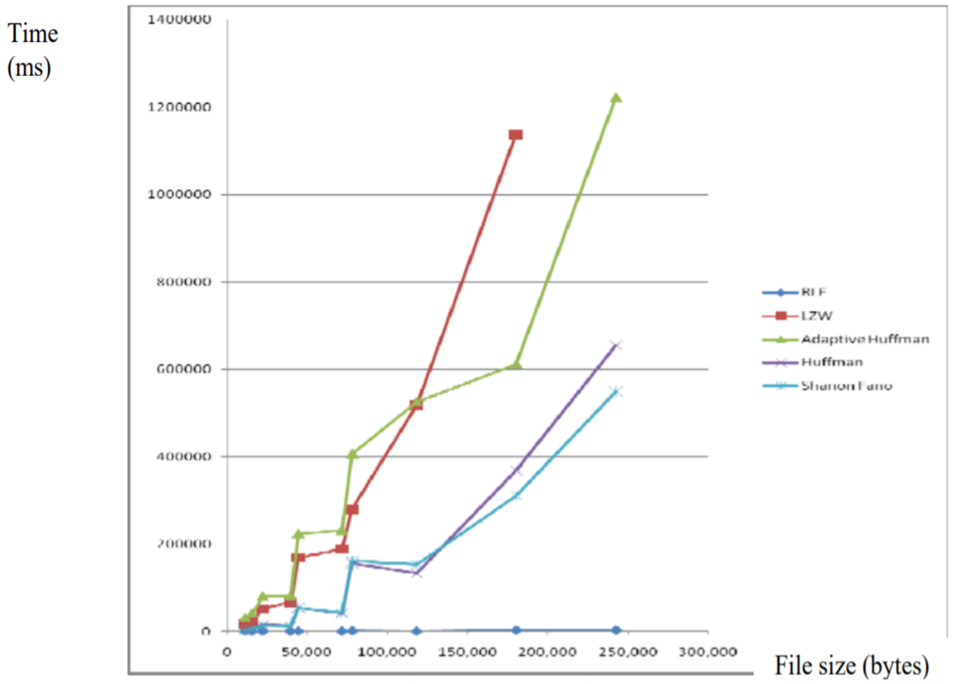


Figure 3: An example from an experiment of a comparison between the performance of different lossless compression algorithms.

Note: The twofish algorithm is designed to be quicker, accept a greater bit length and achieve greater security than the Blowfish algorithm and maintain a greater level of security [4].

The requirements, constraints and specifications of the system and derived subsystems are summarized in the table below.

Type of subsystem	**Requirement**	**Constraint**	**Specification**
Overall system	**R1.1:** The IMU to be used is ICM-20649.	**C1.1**: IMU-20649 hardware is not available.	**S1.1**: A close replica of IMU-20649 must be applied and used to simulate system behaviour.
	R1.2: 25% of the Fourier coefficients of the data must remain	**C1.2**: The lower 25% of Fourier coefficients cannot not be lost.	**S1.2**: Lossless encryption and compression algorithms must be used to process the data.
	R1.3: Computation required for the IP must be minimized.	**C1.3**: There is limited power available on the Pi.	**S1.3**: The choice algorithms must complete operation in the quickest possible time to minimise processor engagement time.
Compression subsystem	**R2.1**: The subsystem must incur minimal data loss.		**S3.1:** A lossless compression algorithm must be used to ensure little or no distortion in data after reconstruction.
	R2.2: The algorithm used to compress the data must achieve a		**S3.2:** The data file size should be reduced enough to allow storage in the 512 MB

	plausible amount of file size reduction or have a low compression ratio. **R2.3:** It must also have a fast compression time.		memory space of a raspberry pi zero. **S3.3:** Compression time of the algorithm must be no more than 60 seconds.
Encryption subsystem	**R3.1**: The encryption block must ensure that data is secured. **R3.2:** The overall processing power utilized in the encryption process must be minimized. **R3.3**: The integrity and readability of the data must be maintained.		**S3.1:** An algorithm with proven experimental results of resisting decryption or with no successful cryptanalysis must be used so that the possibility of unauthorized decryption is illuminated and the data is secured. **S3.2:** The execution time of the encryption process must be in the order of seconds so that processing power is only utilized for a short amount of time and the overall usage of processing power is kept at a minimum. **S3.3:** Encrypted data must go to a random routine decryption test, before transmission, to

		▨	ensure that there is no corruption of data.

Table 3: System requirements and specifications

3. <u>Paper Design</u>

a. UML Use-Case Diagram

A UML Use-Case Diagram detailing the high level interaction between subsystems and how each process is used.

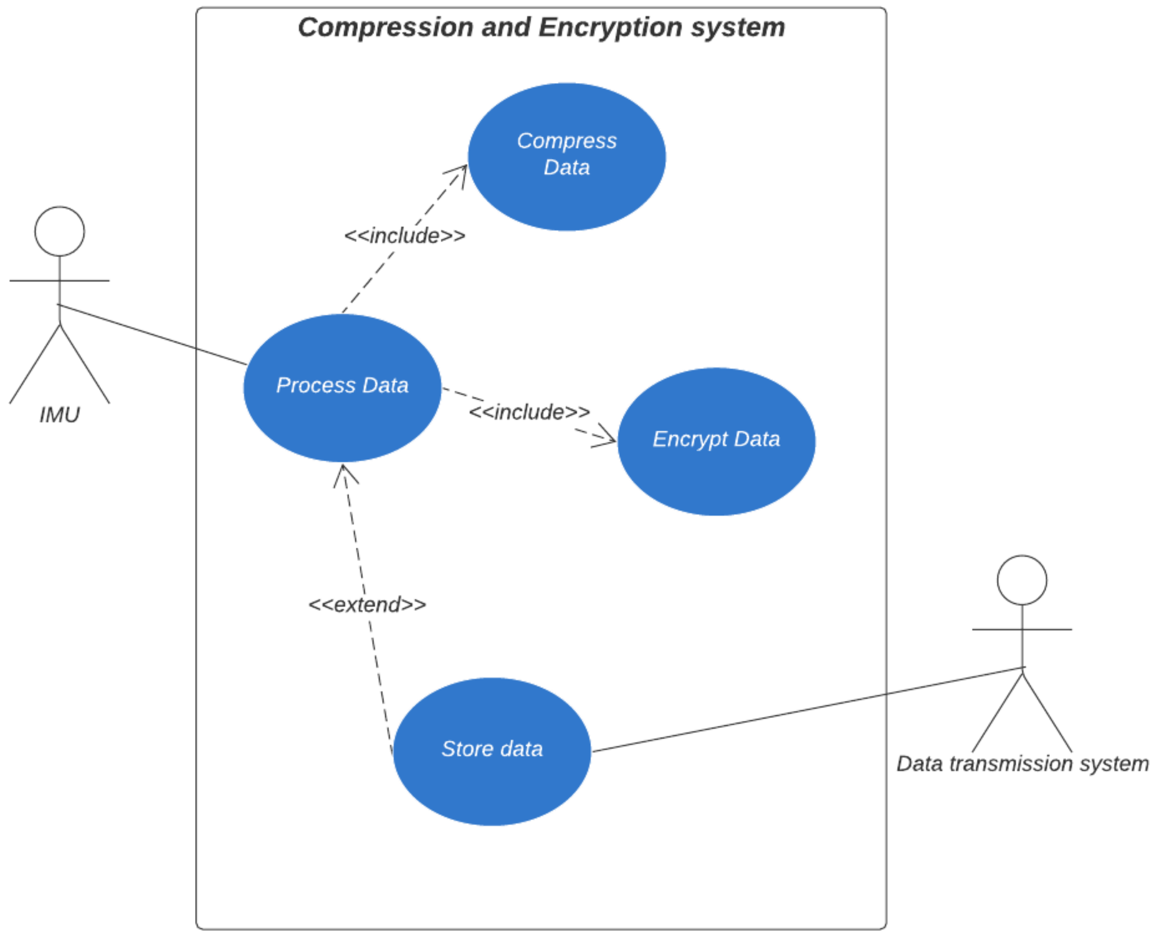


Figure 4: UML diagram of the system

b. System Block Diagram

The system block diagram showing the flow of data and process sequence for the compression and encryption is shown in *Figure 5.*

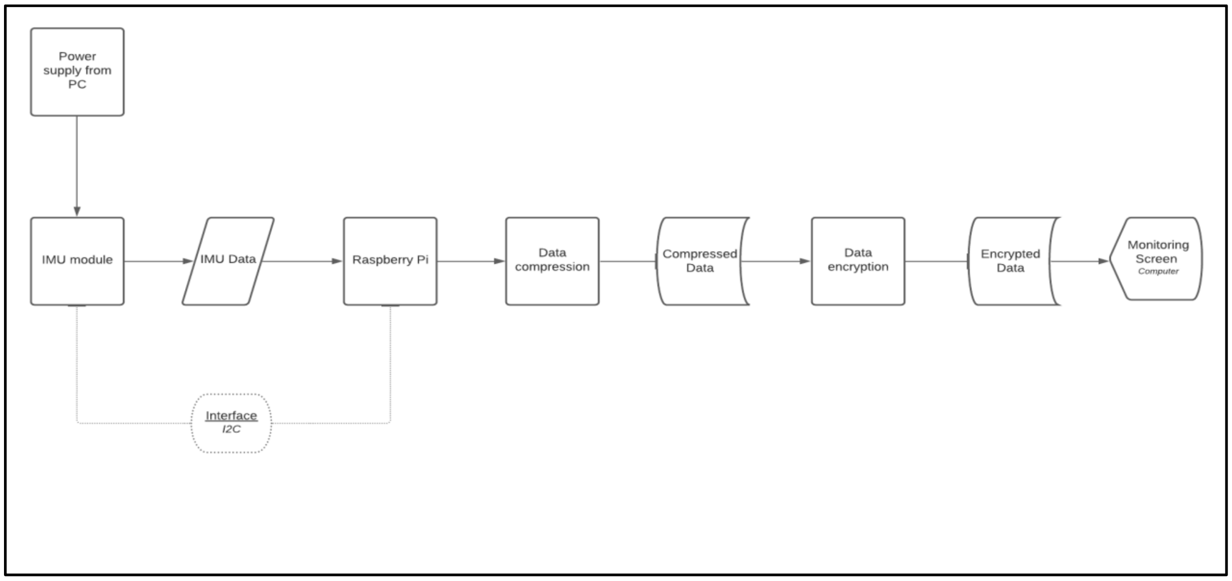


Figure 5: System block diagram

c. Design Acceptance Metrics

The following performance metrics were defined to give the design implementation merit for acceptance.

Compression	**Encryption**
Extract at least 25% of the data Fourier coefficients	Less that 25% of the total RPi RAM space must be utilised
Achieve a compression time of less than 30 seconds	It must take less than 30 seconds to complete the encryption process
Compress data file size must be at least 20% less than the original	The encrypted data must recovered without any losses

Table 3: Acceptance Test Metrics

d. Design process

The design approach that was decided upon for the implementation of the system was to take the modular development route and divide the system into two subsystems, the compression and encryption subsystems. These system blocks would be developed and tested for functionality independently, and later imported into a single program file to integrate them.

The initial step of design once the requirements, specifications and acceptance metrics were defined was to do research on the different algorithms that can be used to implement the encryption and compression. Different algorithms for each submodule of the system were looked into and compared to find implementations that would be suitable and efficient for the project use case. The final choices were to use the Huffman Encoding and Two-Fish algorithm for compression and encryption, respectively. Python was chosen as the programming language that would be used to write code for these algorithms as it is easy to use and the team is more familiar and comfortable with this language.

The next step before the actual implementation could begin was to decide on how the system blocks would tested later on and it was decided that old IMU data collected from a previous project would be used for the first stage of testing since no IMU module was available to do so at the time and later on the Waveshare Sense HAT for Raspberry Pi would used to extract original IMU data, which will be used for the final testing procedure.

4. Validation using Simulated or Old Data

a. Importance of doing simulation based validation

When designing a system, creating an initial model for the purpose of testing is often quite difficult to achieve. This can be attributed to the cost of acquiring the components, the availability of those components as well as the time it would take to create a working initial model. This is often undesirable and inefficient for the purpose of testing. working models often left for the final stages of the design. The solution to achieve testing during initial stages is through simulations. simulations provide a low cost and convenient avenue to validate system behavior without actually spending any monetary value on the system. simulations also provide an efficient way to quickly change design requirements and specifications without any physical repercussions which are often costly. simulations save time and money and ensure that a design proceeds correctly and efficiently.

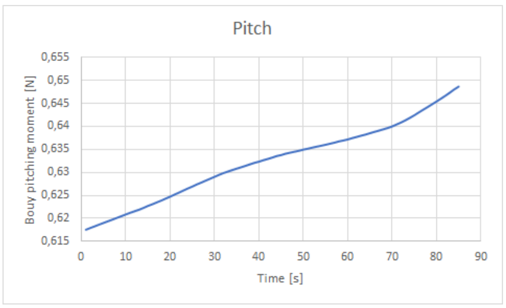
b. Validation steps

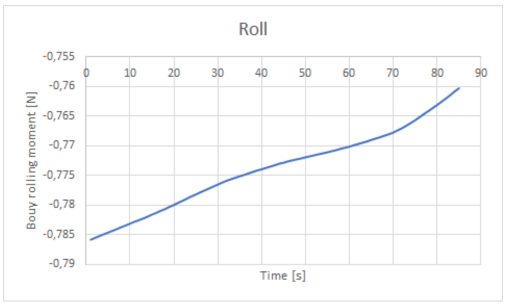
After the simulated data was compiled, the following steps were followed in order to verify that the simulated data indeed illustrates physically realizable behavior. This test would be used to ensure that the simulator daughter represents a true reflection of what would be reasonably expected from the behavior of the real system.The two main aspects that were validated were the integrity of the data (data represents reasonable system behavior) and the ability to be transferable between subsystems.

- To test data integrity, the data points from each measure were represented on a graph. From these graphs, two things were observed:
  - The general waveform was observed and compared to the physical behavior of a well-known and verified system such as ship or boat. The waveform observed should present similar behavior to that of the verified system under certain physically realizable conditions such as a storm or calm ocean.
  - The presence of noise was observed. realistic waveforms are expected to have a significant amount of noise as physical conditions are not noise free. Therefore a waveform that has the presence of noise gives a more realistic indication of how the system behaves in the physical environment.
- To test how the detail behaves when being transferred between 2 devices, a wireless client and server system was created. The data was expected to arrive in the same condition in which it was sent, without data being corrupted. the data would be sent using socket programming.

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

The simulated data that was expected to model the data gathered by the sensor, was plotted on the graphs illustrated below. The simulated data was then compared to the behavior of physical systems similar to the system on which the sensor rests upon, which is a buoy.The data pertaining to the supposed physical motion of the buoy as a result of wave action is illustrated in various graphs below. The pitch, roll and acceleration in the X, Y and Z direction are considered in the illustration.





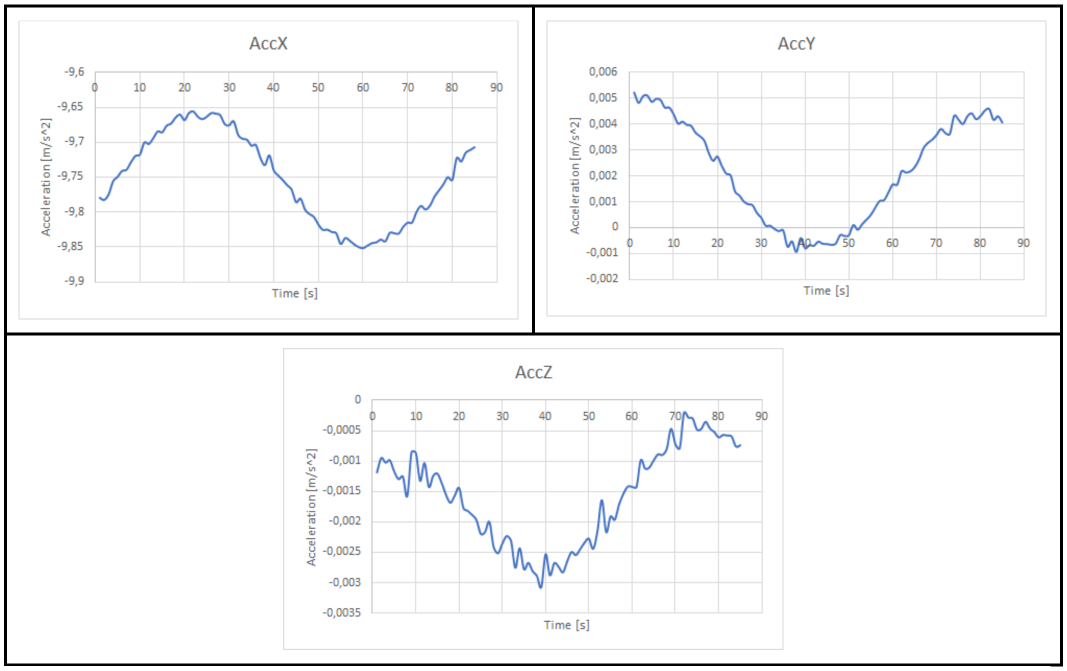


Figure 6: Illustration of the plots of various simulated data gathered by the sensor

Pitch

The waveform of the pitch shows an initially increasing pitch which settles and then increases again. This can be attributed to how a ship experiences pitch when riding over subsequent waves [6]. This means that the data simulated is physically realizable and therefore can be admissible as data that simulates what the IMU senses on the buoy due to wave action.

Roll

The waveform depicted by the plot shows a constantly increasing roll on the sensor situated on the buoy. This can be interpreted as the buoy being constantly elevated on one of the flanking sides thus rolling towards the opposite side. This is an acceptable motion of the buoy as it can be referenced to a ship that also experiences roll (called parametric roll) when traveling in the ocean [5]. The simulated data was compared to the motion of a ship in the ocean and presented a feasible result. This then results in the deduction that the simulated data detailing the roll due to wave action, is admissible.

Acceleration in the X-direction

The acceleration in the X direction was comparable to a surge motion on a ship [6]. This implied that the behavior of the simulated data was extendable to the behavior of a physical system that operates in the ocean which is the same environment in which the buoy and the sensor is expected to exist in. There was also a presence of noise in the data which is typical of the physical system. This implied that the data is admissible as a closer presentation of real system behaviour.

Acceleration in the Y-direction

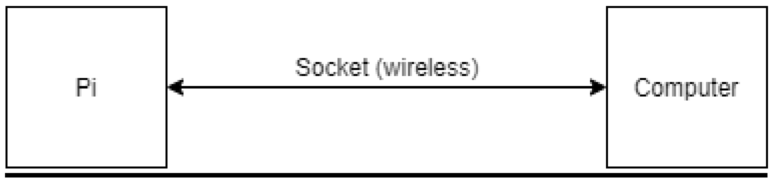
The acceleration in the Y direction was comparable to a heave motion on a ship [6]. This implied that the behaviour of the simulated data was extendable to the behaviour of a physical system that operates in the ocean which is the same environment in which the buoy and the sensor is expected to exist in. There was also a presence of noise in the data which is typical of the physical system. This implied that the data is admissible as a closer presentation of real system behaviour.

Acceleration in the Z-direction

The acceleration in the Z direction was comparable to a swaying motion on a ship [6]. This implied that the behaviour of the simulated data was extendable to the behaviour of a physical system that operates in the ocean which is the same environment in which the buoy and the sensor is expected to exist in. There was also a presence of noise in the data which is typical of the physical system. This implied that the data is admissible as a closer presentation of real system behaviour.

d. Present the experimental setup and results

Setup



- Python code is written to establish a client and server connection via socket programming between the Pi and a computer(laptop).
- Data was then processed on the Pi according to the subsystem descriptions and then sent to the computer via the socket for observation.

Observations and results

- The socket connection is established by an IP address and port combination. Since a TCP connection was established in this design (due to its reliable security during data transfer), an unallocated port had to be found in order for successful transmission as certain ports are reserved for certain operations.The port number used was 1234 as it was available.
- A socket has a configurable throughput. This means that the daughter transferable through the socket is determined by the programmer.
- When the validation tests were executed. initially a portion of the data was transferred as the amount of data that had to be transferred between the devices was unknown. from that the recommendation was drawn that the amount of data that needs to be transferred must be predetermined before the creation of the socket to prevent loss of data or over allocation of throughput.
- The validation test resulted in the data being able to be transmitted and received without any distortions thus validating the operation of gathering data and transmitting it between devices.

5. Validation using a different IMU

a. Importance of doing hardware-based validation

Testing the functionality of the design on hardware or using the Sense HAT(B) IMU module is an important method of validating the design as it allows us to determine if the design application can be extended to the actual IMU that will be used for the buoy. It is important to know that the design can be used in a real system like the one the project is intended for and not just be limited to application on simulated data or old test data.

b. Validation steps

Before conducting any experimentation on the IMU to make sure that the design works, the following steps were taken to validate the IMU module and to ensure that it worked as expected:

A physical connection between the Sense HAT and the Raspberry Pi was first made using connector wires to connect the voltage supply, ground, serial data line and serial clock pins of the module to their corresponding Raspberry Pi GPIO pins, that is the 3V3, GND, SDA and SCL pins, respectively.

To enable data transmission between the IMU module and the Raspberry Pi, the next step was to enable the I2C interface connection between the two, so that the Raspberry Pi would be able to detect the Sense HAT and communicate with it.

Python programs which can be found in the GitLab repository link shared in the Admin documents section of this paper were written to extract data from the IMU and transmit it to the Raspberry Pi.

A set of test cases were developed to check the functionality of the ICM20948 and LPS22HB on-board sensors as a way of monitoring the IMU module's behaviour. These tests included:

- Temperature response through exposing the module to different temperatures, such as normal room temperature with no external factors, hot temperature created using a heater to increase the temperature in the room and lastly, cold temperature created using ice cubes put in a container which was brought near the IMU.
- Getting pressure readings from the LPS22HB barometer to check if the readings would correspond to the expected standard atmospheric pressure.
- Applying a magnetic field to the space around the IMU by bringing a magnetic object near it to observe how the data changes with changes in magnetic strength.
- Lastly, the Sense HAT was moved around to check the IMU motion sensor's response to movement and the accelerometer readings were observed.

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

To test the compression and encryption algorithms using the Sense HAT, the first step was IMU data extraction. This is the step where data is extracted from the IMU module using a python program and the program output is redirected into a text file to make comparisons between the original file with the compressed, encrypted, decrypted, and decompressed files easier as a way of validating the functionality of the compression and encryption algorithms.

The data being recorded into these output files is the 3-axis acceleration, pitch, and roll.

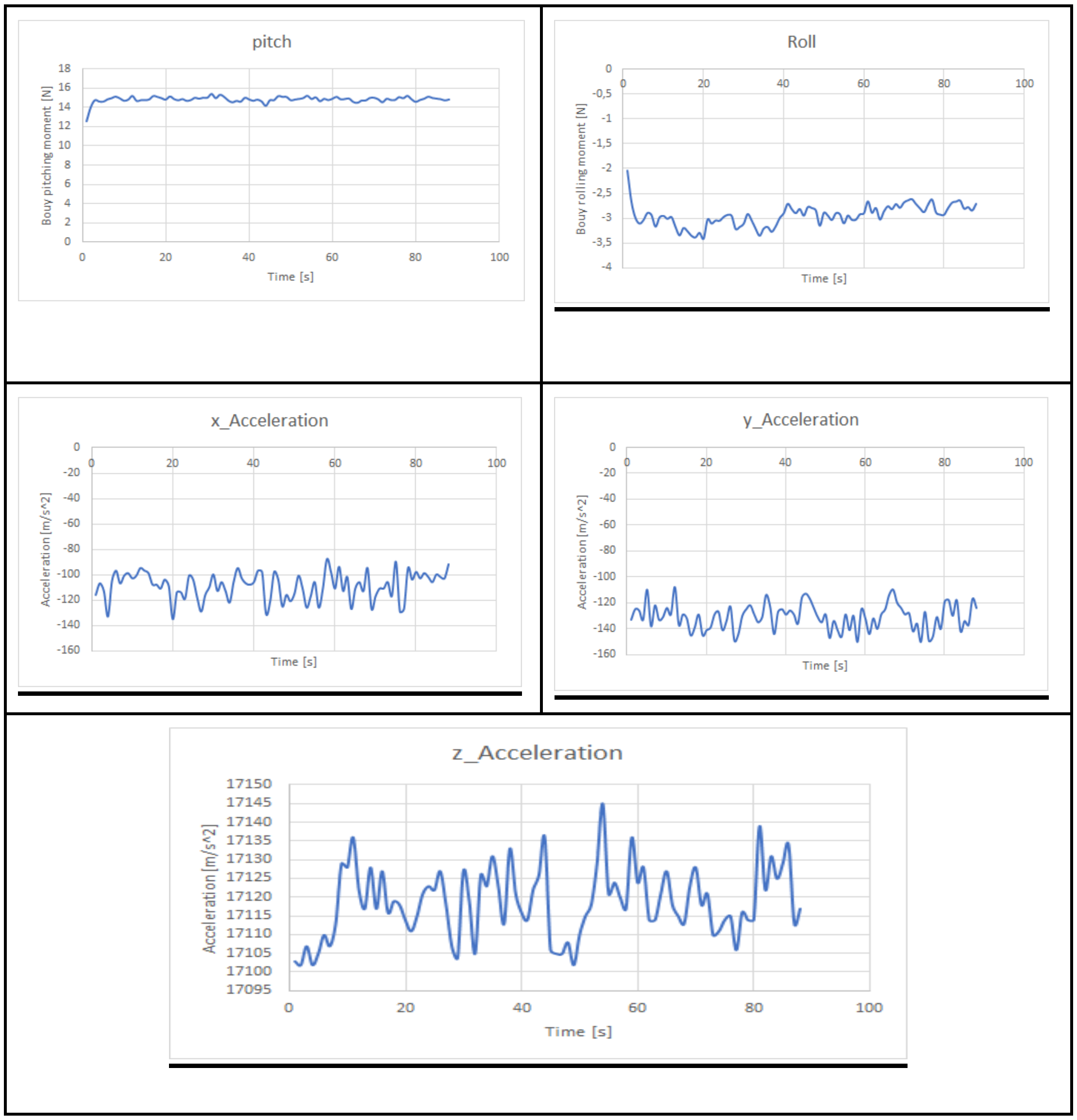


Figure 7: Data extracted from the IMU module

d. Experimental setup

Ten files of sample data of varying sizes were used for experimentation. Using the recorded data stored in files the following steps were followed to conduct the experiment:

- A bash shell script was created to automate the to run the compression and encryption algorithms.
- The original data files were first compressed using the Huffman encoding algorithm implementation which outputs a string array containing the compressed format of the data. The arrays were pushed into separate output files.
- The compressed data was then taken as an input to the two-fish algorithm program and encrypted. The output of the encryption block of the system was also redirected into encrypted data files.
- To analyse what the data looked like after both compression and encryption, the encrypted files were decrypted to check if the decrypted data exactly matched the compressed data files.
- The decrypted data which was also stored in files was then decompressed into different output files containing the decompressed version of the compressed data.
- While these processes were being executed, the execution time of the individual processes as well as that of the entire program were measured using programs that were specifically coded to calculate the run time of the compression, encryption, decryption, and decompression, respectively, these values were then recorded.
- Finally, a comparison between the output files that were saved for all the processes were compared. Each of the original files were first compared to the compressed version to check if compression was achieved by looking at the file sizes and observing if the compressed file was smaller than the original.
- The next step was to compare the compressed files with the encrypted files to ensure that encryption was successful, and the data was encoded securely.
- The decrypted files were checked to check if the decrypted data files looked like the compressed files as expected, and lastly, the data in the decompressed files was compared to the original data file to check if all the data was recovered successfully after all the processing.

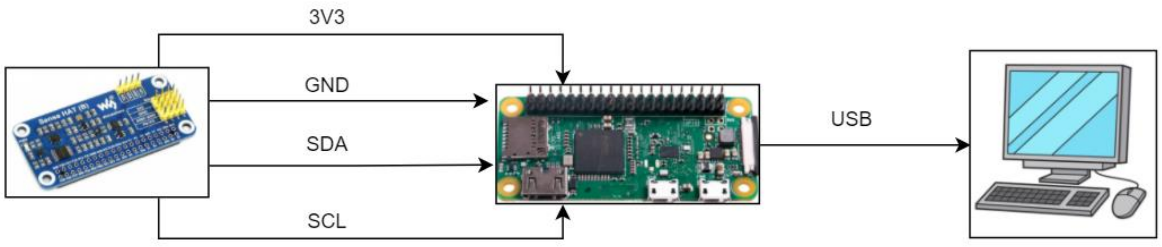


Figure 8: Diagram of Sense HAT connection to the RPi and pc

e. Results

The predefined merits of acceptance for the compression algorithm were that the implementation should most importantly reduce the data file size to show that there was indeed successful compression and the set benchmark for this measure was a saving percentage of more than 20%, the compression process should take less than 30 seconds to execute and since a lossless compression algorithm was used, all the originally extracted data must be recovered after decompression, which means over 25% of the Fourier coefficients of the data were successfully captured.

Figure 9 shows a graph of the size of the original data files which contain the IMU data and that of their compressed versions. From the graph, all the compressed files have sizes less than the original files which means compression was achieved.

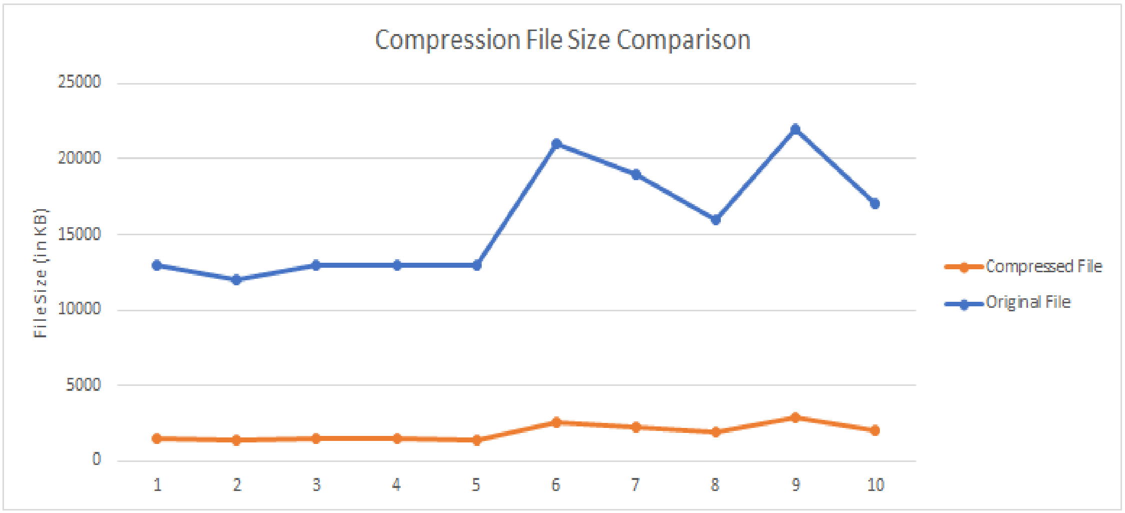


Figure 9: Comparison between compressed and uncompressed data file sizes

The figure below shows that the IMU data files were reduced by over 86,5% after compression, which is way better than the 20% that was initially set as an acceptance metric.

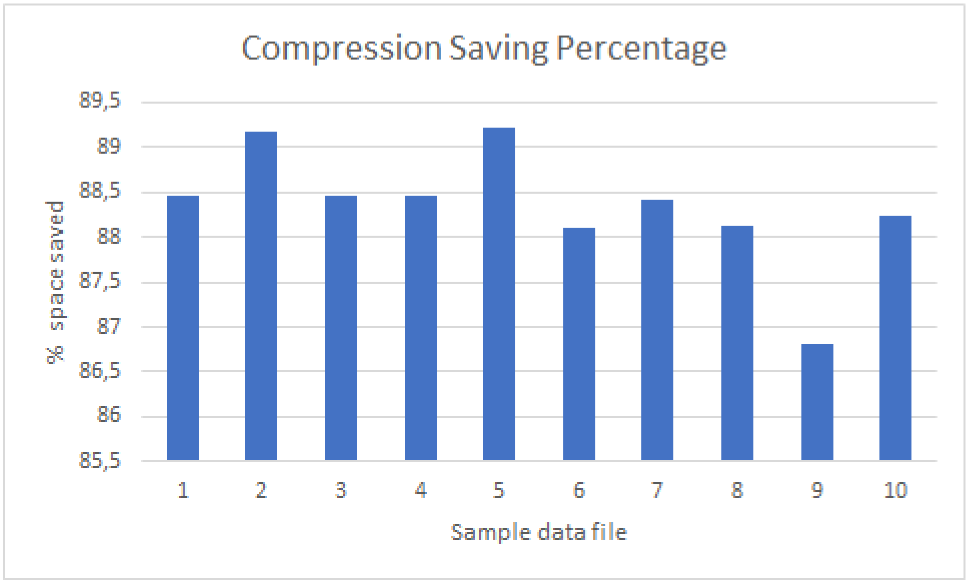


Figure 10: Bar graph for the saving percentage of the compression algorithm

The 30 seconds compression time goal that the design was aimed to meet was achieved as can be seen in *Figure 11*. It took less than 3 seconds to compress all the IMU sample data files.

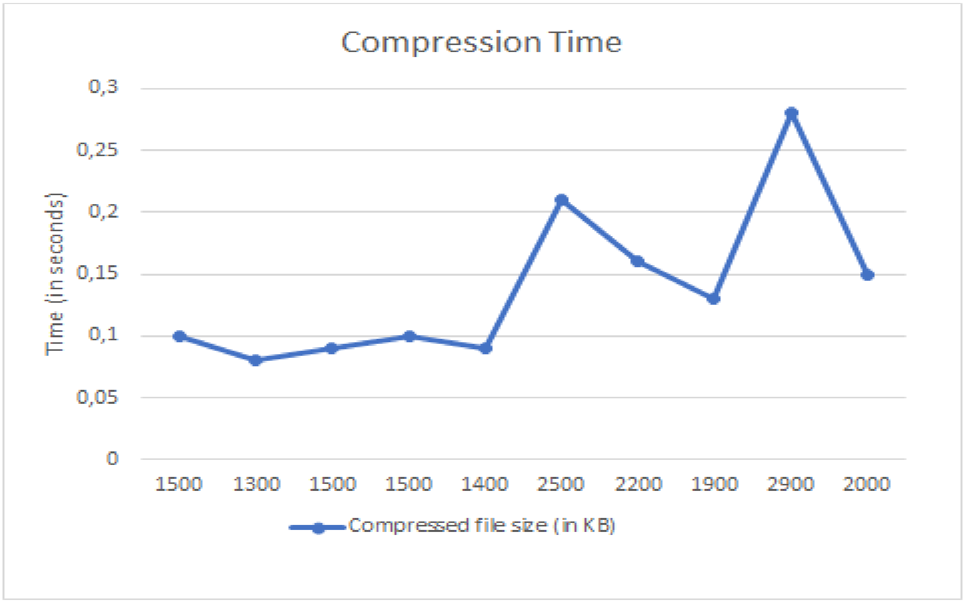


Figure 11: Execution time for compression the data files

The acceptable performance criteria set for the encryption algorithm was to achieve a RAM usage percentage of less than 25% of the 512MB total RAM space available in the Raspberry Pi Zero W, an encryption time of less than 30 seconds and a lossless output.

All these design goals were achieved as can be seen in *Figure 12, 13* and *14 shown below. Figure 12* shows a bar graph representing the amount of RAM used for each data file that was encrypted during the experiment and an average of 20,5% of RAM was used during the process.

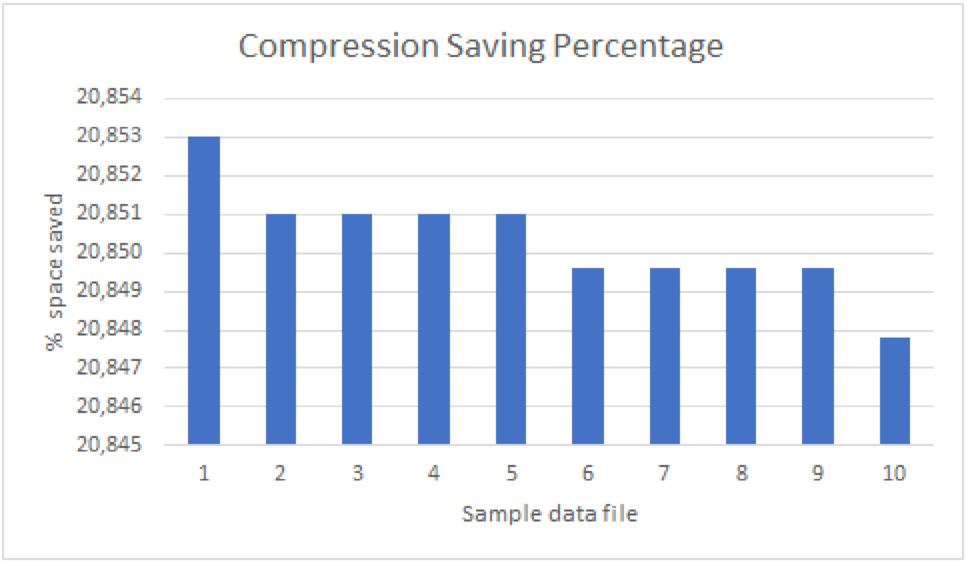


Figure 12: Bar graph of the % of RAM used

In *Figure 13*, representing the encryption time, it is observed that all the sample data files that were encrypted were all processed in less than 0,09 seconds, achieving an average program execution time of 0,051 seconds.

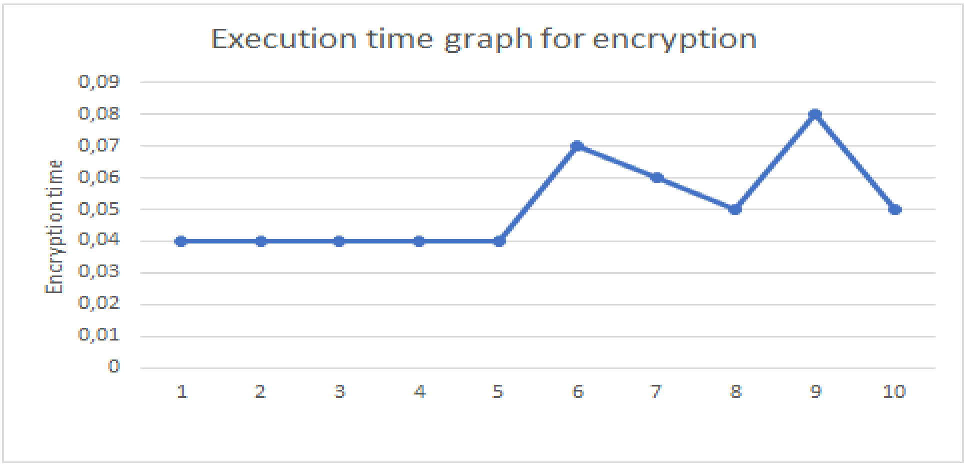


Figure 13: Encryption time for the different files

The figure below depicts the fourier transforms of the extracted IMU data and processed data that went through compression, encryption, decryption and decompression, respectively. The results show the exact same output as expected since lossless algorithms were used for both the encryption and compression, and these should retain all the Fourier coefficients of the data that was initially extracted from the Sense HAT.

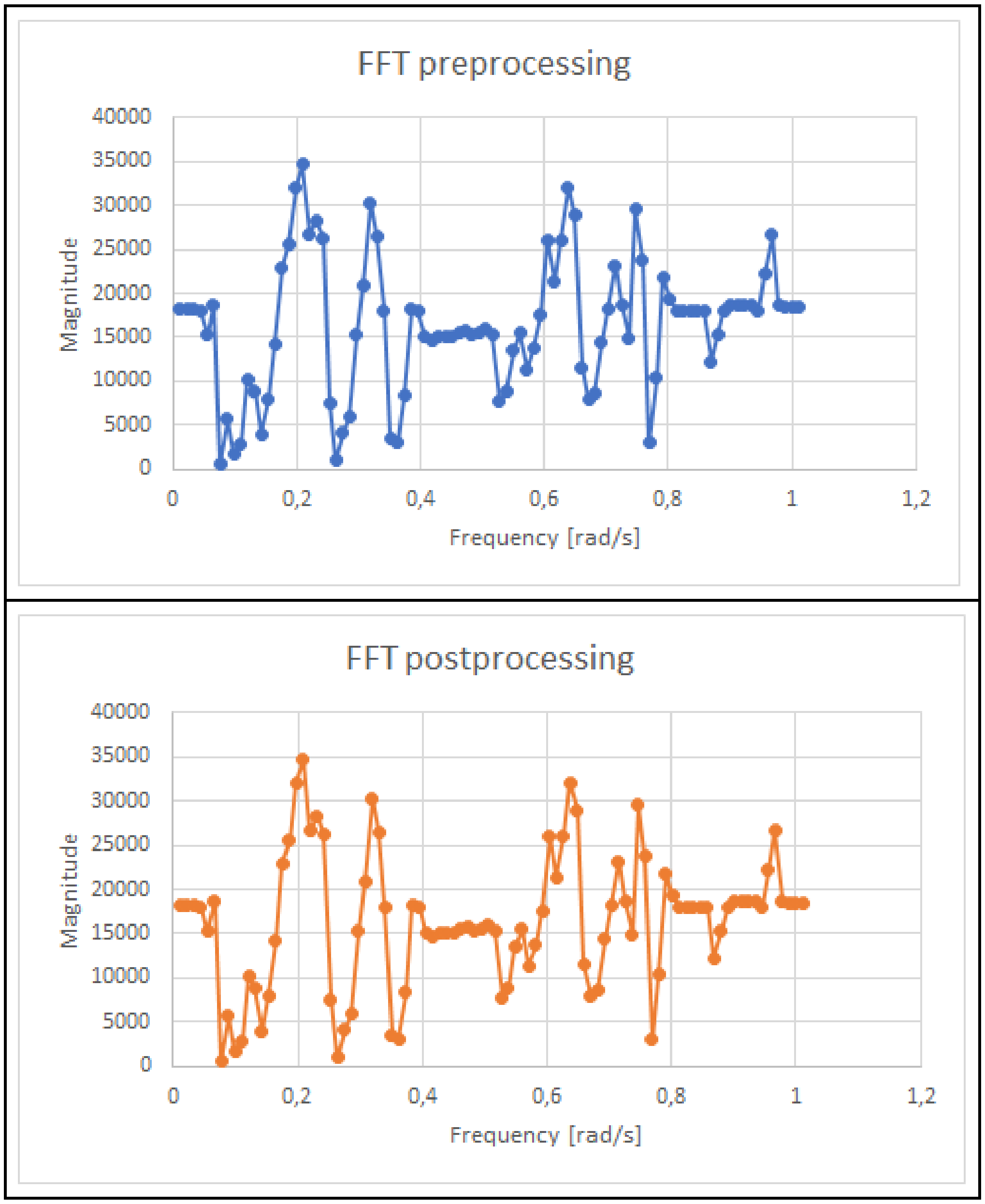


Figure 14: Fourier Transform plots of the original data and processed data

6. Consolidation of ATPs and Future Plan

Acceptance metric	Accepted?	If not, why and what change is required?	Comment
COMPRESSION			
Lossless compression	Yes		No data was lost during processing
Compression time	Yes		Required < 30 seconds Achieved an average of 0,139 seconds
Saving percentage	Yes		Required >= 20% Achieved over 86,5%
ENCRYPTION			
RAM usage	Yes		Required < 25% Achieved 20,5% (90Mb out of 512Mb)
Execution time	Yes		Required < 30 seconds Achieved 0,051 seconds on average
Lossless decryption	Yes		Readable data and resembled original data.

Table 4: ATPs consolidation table

Future recommendations:

- Include socket programming ATPs to increase the self reliability of this project as it will be able to transmit on demand via sockets.
- Increase automation in the system by adding timed executions and computational resource management systems eg. execute an instruction if execution exceeds a certain threshold.

- Create more realistic environments in order to achieve more reliable results eg. indoor tidal pools and refrigerators.
- Increase sample size of gathered data to further improve reliability of the study.

7. Conclusion

The aim of this report was to describe the design, experiment and approaches taken in creating a solution for the aforementioned design problem. Compression and encryption subsystems were designed and implemented in accordance with customer requirements and derived specifications. Interfacing and data exchange, between the systems, was also considered and priority was placed on secure data transfer. System limitations were considered for available system resources and for the harsh antarctic environment in which the system was expected to operate. Feasibility analyses were conducted to verify the viability of the suggested designs and possible bottlenecks were identified to reduce the possibility of unplanned system behaviour. The design was initially tested under simulated conditions as well as using a real model(close replica) of the intended system, in order to monitor the system response to real conditions. It was of significant importance to validate the data used in the simulations and the input parameters of the real model. From the result obtained, deductions were conducted about the expected behaviour of the system in the real antarctic environment. From the experiments conducted, the system responded positively and was in accordance with requirements and specifications. The system was able to operate within the defined limitations and conserve limited resources. Data integrity was maintained during subsystem data transfer. The compression and encryption subsystems performed their intended functions successfully. The design was a success according to the defined scope. Additional considerations are recommended to improve system functionality and automation. Timed programmatic triggers, environmental and internal resource management systems may be introduced to establish an intelligent control tower mechanism [7], where the entire system functionality is conducted with no human interaction or maintenance needed. Experiments may be conducted to investigate the significance of the effect of different interfacing protocols on the overall utilization of system resources. This would be done to ensure that the system can have finely tuned behaviour, without interfacing protocols negatively affecting the expected system design. Overall, the design performed well within the defined scope with the window for improvement always open.

8. References

[1]"Sense HAT (B) - Waveshare Wiki", *Waveshare.com*, 2021. [Online]. Available: https://www.waveshare.com/wiki/Sense_HAT_(B). [Accessed: 31- Oct- 2021]

[2] D. A. B. A. M. Soheila Omer AL Faroog Mohammed Koko, “Comparison of Various Encryption Algorithms and Techniques for improving secured data Communication,” IOSR Journal of Computer Engineering (IOSR-JCE), vol. 17, no. 1, p. 8, Feb. 2015.

[3] K. Ullah, B. Ayisha, F. Irfan, I. Illahi, and Z. Tahir, “Comparison of various encryption algorithms for securing data,” Rclis.org. [Online]. Available: http://eprints.rclis.org/33889/1/comparison%20report%20.pdf. [Accessed: 05-Sep-2021].

[4] "Twofish vs Blowfish | Encryption Differences | Gig Mocha", *Gig Mocha*, 2021. [Online]. Available: https://gigmocha.com/twofish-vs-blowfish-encryption-differences/. [Accessed: 02- Nov- 2021]

[5] 2021. [Online]. Available: https://www.wartsila.com/encyclopedia/term/parametric-rolling. [Accessed: 02- Nov- 2021]

[6] "Ship motions at sea and their effects on cargo ships", *FFQO US*, 2021. [Online]. Available: https://www.freightforwarderquoteonline.com/news/six-types-of-cargo-ship-motions-at-sea-and-their-effects/. [Accessed: 02- Nov- 2021]