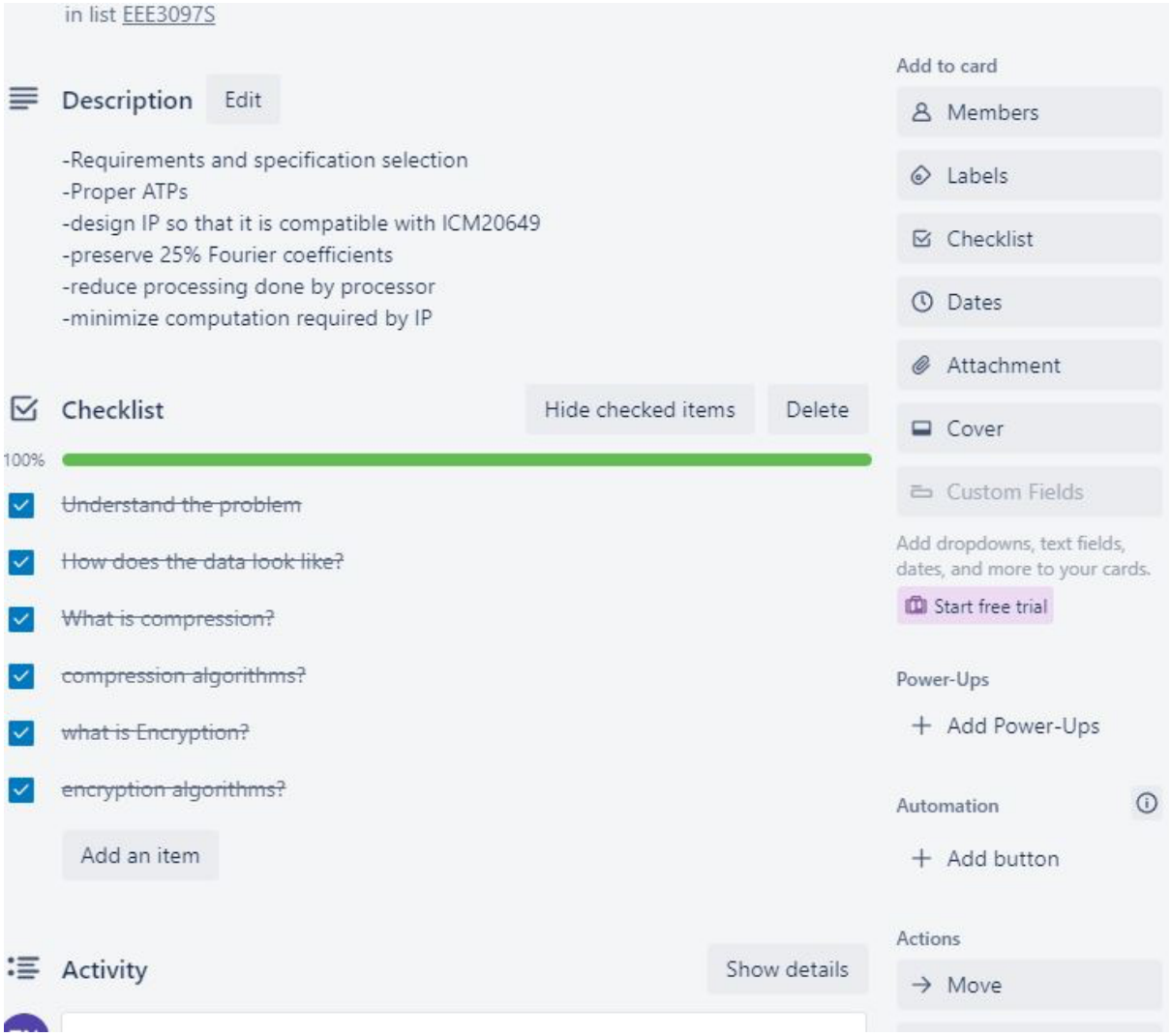
2022

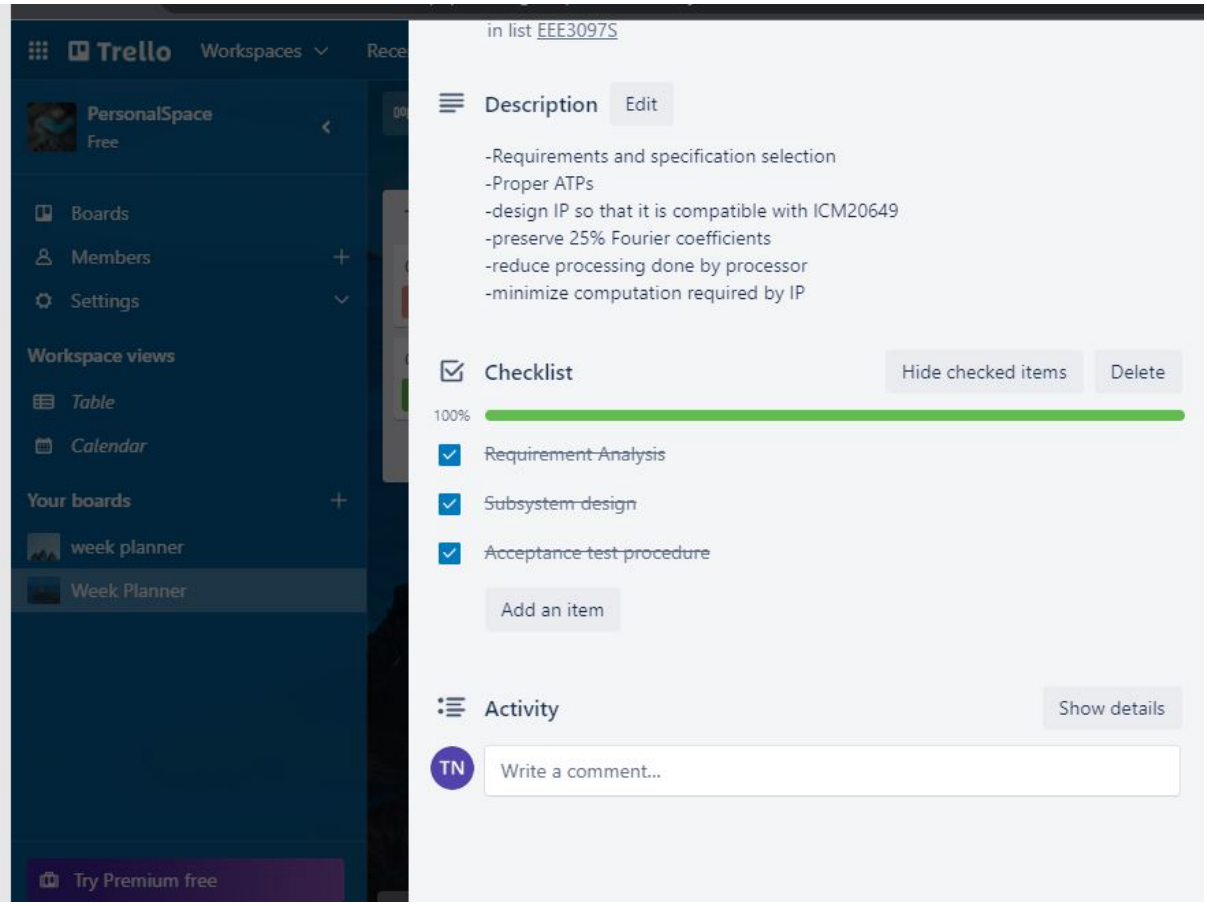
Paper Design

MILESTONE 1

TLOKOMELO NKOSI(NKSREG002) & ASHLEY RATAU (RTXASH001

Management Tool Used





The link to Trello: https://trello.com/c/MCHFe8JK/9-paper-designrequirement-analysis

Table Listing shared and Individual Contributions

Team Member	Contribution
Tlokomelo Nkosi	1.1,1.2,1.3, 2.1.1,2.1.3,3.1,3.2
Ashley Ratau	1.2,1.4, 2.1.2,2.1.4,2.1.5,3.3
Shared	2.2,2.3

Development Timeline

The group meet twice a week at 18:00 on Tuesdays and Thursdays

01/08/2022-05/08/2022- Requirement analysis

08/08/2022-12/08/2022- Subsystem Design

15/08/2022-19/08/2022- Acceptance test procedure

19/08/2022-22/08/2022- Writing report

Introduction

The aim of the project is to design an ARM based digital IP, utilizing the STM32F0 specifically to compress and encrypt the Inertial Measurement Unit (IMU) data containing information about the ice and wave dynamics in the southern Antarctic Ocean.

1. Requirement Analysis:

1.1 Interpretation of the requirements

Requirement 1: The ICM to be used is the ICM-20649 however these will not be provided because they are expensive instead the ICM-20948 will be provided. The important task is to design our IP so that it is compatible with ICM-20649 even though will be testing on ICM-20948.

Requirement 2: 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 the compression satisfies this.

Requirement 3: In addition to reducing the amount of data we also want to reduce the amount of processing done in the processor (as it takes up power which is limited). Try to minimize the computation required for your IP.

The table below shows the user requirements derived from the project description document. These user requirements will be used to derive the functional requirements and eventually the specifications. The headings are the user requirement ID and description.

Requirement ID	Description
UR001	The user requires that they be able to extract at least 25% of Fourier Coefficients of the data
UR002	The data must be compressed
UR003	The data must be encrypted
UR004	Minimize processing done by the processor to preserve the limited battery life
UR005	Use a programming language that gives the best benefit
UR006	Data is transferred after the completion of processing
UR007	Use an ARM based board

1.2 Comparison of compression and encryption algorithms

Compression:

Compression is used to reduce file sizes [1]. Compression can be classified into two categories which are *lossy* and *lossless* [1]. Lossy compression permanently removes parts of the data in reducing the size of a file from which the removed data cannot be recovered. In lossless compression, as the data size is compressed the data is not compromised or lost, as you are still able to recover the original data during decompression. There are trade-offs to which category of compression is used. Lossless compression is better suited for text files and lossy compression is better suited for image files. Below are some of the lossless compression algorithms available.

The LZ77 is the fundamental base of other lossless compression algorithms [1]. LZ77 uses the sliding window method. How the algorithm works is that it refers the same byte sequences to its first occurrence. The algorithm manages a dictionary using triples to represent the offset which is the distance between the start of a phrase and the beginning of a file, Run-length which is the number of characters that make up a phrase and the deviating characters which are markers indicating a new phrase [2]. The dictionary updates dynamically to show what is contained in the file and what is the size of the file whenever a file is parsed.

The sliding window analyses the input data sequence and keeps a record of the data as part of the dictionary. The sliding window has a search buffer which contains the dictionary (recent encoded data) and a look-up buffer which has the portion of the input sequence to be encoded next. The performance of the compression algorithm is dependent on the size of sliding window [3]. If the size of the sliding window is too small, that means less repeated data sequences found by the compressor resulting in a larger file size and if the size of the sliding window is too big, that means the compressor takes a while longer to find repeated sequences, thus resulting in slow compression speeds.

The LZR is a linear alternative to the LZ77[2]. LZR requires a massive amount memory when used in non-linearity and is therefore not a better option compared to LZ77.

The LZSS [2] is also a lossless compression algorithm derived from LZ77 and aims to improve on it. It comes with an extra feature that checks if a substitution decreases file size and if it doesn't it left in its original form. LZSS uses only offset -length pairs through the elimination of deviating characters making the size of the dictionary small.

Deflate is another compression algorithm that combines Huffman coding with either LZ77 or LZSS pre-processor. It is an entropy encoding that assigns codes based on the frequency of a character [2]. The codes are based on varying lengths depending on how frequently the characters represented appear and the more frequent or modal the characters are the shorter the code [3]. To prevent ambiguity when decrypting, the codes need to be decoded in a specific way. Huffman codes can be implemented in two ways, namely static and dynamic encoding. The same codes are used for all input data in the static implementation. In dynamic encoding the input data is analysed and

categorized into its typical code size by breaking the data into chunks that are processed separately.

ZLIB [3] is a data compression algorithm that is utilized when it is required that entirety of compressed data is recovered when decompressing a file. The ZLIB uses the deflate method to compress the data and the inflate method to decompress the data back to its original form. Off all the algorithms ZLIB is more realistic and efficient in its implementation. The ZLIB has 10 compression levels with each levels offering a different performance in relation to the compression ratio and speed. The different levels range from 0-9, with level 0 representing no compression. Level 1 is the fastest while having the lowest compression ratio and Level 9 is the slowest while having the highest compression ratio [3]. In order to determine small strings, the Rabin-Karp string matching algorithm is used. The advantages of the algorithm are that insertions are easy, and it avoids deletions.

Encryption:

Encryption can either be symmetric or asymmetric [4]. In symmetric encryption there is only one key used for encryption and decryption, while in asymmetric encryption there is more than one key for encryption and decryption. There are keys that are specific to encrypting data and keys that are specific to decrypting data, these are called ciphers in the context of asymmetric encryption. There are two types of ciphers namely the stream ciphers and block ciphers [4].

Stream Cipher creates an encrypted cipher text where the plain text is XOR'd bit by bit with the key, for the to work properly the key size and data must be the same. It then uses the exact same process to decrypt the cipher text. The stream cipher is made realisable through the introduction of a pseudorandom generator. A pseudorandom generator is a process that takes an input and returns longer result deterministically. To ensure that there is only one shared key a nonce is used to enforce the notion of keeping data secure and never repeating keys.

Block cipher iteratively encrypts the input data using a different key resulting in same length texts [4]. 3DES and AES are the epitome of block ciphers which take an input of 48 bits and 128 bits respectively. In a block cipher each new round generates a different key through the key expansion mechanism. AES is built as a substitution permutation network [4].

AES works of a 16 bytes block. The process of operation is that the round key is XOR'd with the current message and then follows the substitution process where blocks of data interchange and then permutation happens where bits are permutated and shuffled around. This process is repeated until the final is received. In order to decrypt the block cipher, the same process is done in reverse.

Of the available modes of operation of block ciphers, the Electronic Code Block (ECB) [4] is the most used. The modes of operation are used work around the fact block ciphers only take a fixed length input. ECB divides the data into 16 byte blocks and performs AES encryption uniformly [6]. ECB can also be done in parallel although it doesn't guarantee security. Another mode of operation is the Cipher Block chaining (CBC) a mode of operation that chains together each 16-byte block of plaintext by conducting block cipher encryption first and then XORing the previous plaintext's ciphertext into our current plaintext.
The other mode of operation is the Randomised Counter Mode (RCM) [1], which is the safest of all the available options.

Since the STM32FO already has libraries for the encryption and decryption functionalities, implementing the AES encryption method on it is simple.
This library uses the Cipher Block Chaining mode of operation, which permits the input of messages of arbitrary size.

1.3 Feasibility Analysis

Schedule:

To finish the project before the deadline, a definite timetable covering the upcoming weeks has been established. A milestone that calls for completing specific phases of the project plan must be submitted every three weeks. The individuals working on the project meet once a week to go over each step in the plan and how it can be carried out. Thus, despite time constraints, the project can still be finished on time thanks to the established timeline.

Technical:

All requirements will be met except for not working with the actual IMU since it will not be provided, although a reasonable attempt will be made to make sure that the IP tested on a different is compatible with the IMU on the SHARC Buoy. Many of the techniques required for the two submodules, compression, and encryption, may be downloaded into the STM boards and are readily available online. There is an issue with memory of the STM that could almost halt the project which would require looking for an alternative board like the Raspberry PI zero.

We have only been learning how to utilize the STM boards for over a month, so we may encounter problems that call for in-depth investigation. We both have never used an STM before until a month ago, we both did EEE2046F in 2020 during covid where we couldn't receive the STM boards we started learning now. However, we do have the Pi boards.

We believe that it may prove difficult to filter the IMU data so that 25% of the coefficients are recovered. We know from our scant investigation that the IMU output data can be quickly transformed using the Fast Fourier transform in the programming language C that we have chosen by design of using the STM. Despite this, we believe that our lack of C experience may hinder us. Overall, we have the tools necessary to fulfil every user requirement for the design project.

1.4 Possible Bottlenecks

There are numerous trade-offs, some of which were already described above and others which have not yet been mentioned. Time is a factor in all trade-offs in one way or another because we want our system working in real time. some of the trade-offs are choosing a compromise between the compression level and time needed to compress the data.

Different encryption techniques are better suited for different situations. While some algorithms are better suited to limited length data, others can handle streams of data of various lengths. The time it takes to perform the encryption can be impacted by this decision. The data may be vulnerable when it is being transmitted since the file might not entirely or correctly encrypted. Because of possible slowness in the encryption and compression, the 25% Fourier preservation may not be met.

To extract only the necessary data from the data set, pre-processing is required. The more precisely this processing is done, the longer it will take, and thus more power consumed. It is necessary to determine how much data should be included in each transfer packet. How long does the code wait till there is enough information for processing before transferring? The amount of time needed to analyse and transmit each packet will increase with the size of the data collected per packet. If the wait time is too long and the buoy only transfers periodically and disappears before it can transfer a sizable amount of data, the data that could have been sent may be lost.

2 Subsystem Design

2.1 subsystem and sub-system Requirements and Specifications

2.1.1 Compression of data

One of the various compression methods that are available on the internet is used to compress data, ZLIB to be specific. The various methods have been compared, and the most effective approach for the SHARC BUOY application was chosen using the speed and compression ratio. For a variety of reasons, data compression comes before data encryption. The most popular response is that, after data has been encrypted, the file creates a random stream of data that is nearly hard to attempt to compress. In order to reduce the excess data size, compression also rely on the discovery of compressible patterns.

Data compression is necessary for the SHARC BUOY project in order to transmit data from the buoy across the current transmission network in Antarctica [7]. The satellite network this transmission method uses imposes restrictions on it. Because of the network's expensive data transfer fees, erratic transmission dependability, and bandwidth and data structure requirements, portions of the transmission are constrained. Because there is less data to transfer, compression will reduce the cost of doing so. It will also raise the likelihood that the transfer will be successful because it can be done in a shorter amount of time.

Functional Requirement ID	Description	User Requirement satisfied
FR001	Data sent to the STM must compressed. Data received from STM must also be compressed	UR002

Design Specification	Description	Functional Requirement satisfied
DS001	The deflate algorithm for compression and the inflate method for inflation are implemented by the Zlib library. The STM32FO is compatible with this library. The STM32FO must compress the data it processes to the fewest amount of bits possible without losing any integral data.	FR001

2.1.2 Encryption of data

Data is encrypted using the AES algorithm. This type of encryption is trusted, utilized, and widely recognized. The STM32FO already provides a library for the encryption and decryption functionalities, making it simple to apply the AES encryption approach [4]. This library's Cipher Block Chaining method of operation enables the entry of messages of arbitrary size. With this method of operation, keys can be used several times without endangering the security of the encryption. Therefore, even if the data set is repeated, it is impossible to figure out the key.

Data encryption before transmission is a smart technique to ensure data security. For the purpose of communication between the buoy, the transmission tower in Antarctica [8], and the lab that will capture the data, this information will be encrypted.

Functional Requirement ID	**Description**	**User Requirement satisfied**
FR002	The data sent to and received from the STM32F0 must be encrypted using a suitable algorithm	UR003

Design Specification	Description	Functional Requirement satisfied
DS002	AES algorithm is to be used because it is compatible with the STM32FO	FR002

2.1.3 Data Processing and filtering

At least 25% of the Fourier Coefficients must pass filtering as a user requirement. This subsystem is crucial, as it collects all pertinent information from the IMU data output stored in the STM32F0 memory. Additionally, this data reduction makes it possible for transmission, compression, and encryption techniques to operate more effectively. Since the data from the IMU is measured in the time domain, a Fourier transform must be applied to the dataset in order to transform it to the frequency domain. This will enable the implementation of the fast Fourier transform (fft) method together with the coefficient equivalent filtering of the data with the appropriate sampling time. The STM32FO predominantly uses the C language which is a good thing because it highly recommended due to its energy efficiency, speed of execution and data rate

Functional Requirement ID	Description	User Requirement satisfied
FR003	Remove any potential aliasing from the data before extracting the 25% of Fourier coefficients.	UR001
FR004	The programming will be done in C	UR005

Design Specification	Description	Functional Requirement satisfied
DS003	To extract 25% of the data's Fourier coefficients, the IMU's data output will undergo the coding equivalent of a low pass filter.	FR003 & FR004

2.1.4 Data Transmission

The broader STM32F0 subsystem involves data transfer, although this subsystem mainly concentrates on sending the compressed and encrypted data to the iridium network. The I2C (a serial communication protocol) and SPI (Serial Peripheral Interface) are two sub-subsystems that are utilized to send data from the IMU to the STM and from the STM to the iridium network.

Functional Requirement ID	Description	User Requirement satisfied
FR005	The iridium network receives the data that has been compressed and encrypted.	UR006

Design Specification	Description	Functional Requirement satisfied
DS004	Use of the iridium modem to transmit data packets	FR005

2.1.5 STM32FO usage

The STM32FO is the point reference for all data transmission and manipulation.

Functional Requirement ID	Description	User Requirement satisfied
FR006	Use an ARM based processor	UR007

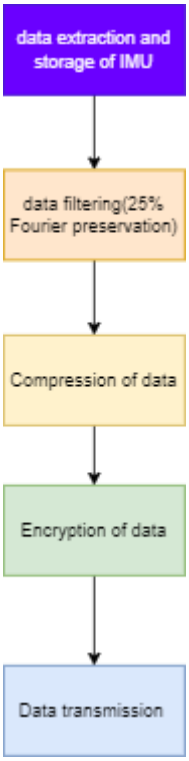
Design Specification	Description	Functional Requirement satisfied
DS005	Use an STM32FO	FR006

2.2 Inter-Subsystem and Inter-Subsystem Interactions

The STM32FO is the focal point of the entire system because everything gets through it at some point. The STM is the focus of the design however it is a subsystem of the SHARC buoy. The ICM-20649 sends the data it collects to the STM32F0. Within in the STM after receiving the data from the IMU filtering of the data takes places and 25% of the Fourier coefficients are extracted before any compression and encryption. Immediately after the filtering is done the data is compressed using the ZLIB library. Then after the data is encrypted using the AES algorithm. From there onwards the data is transmitted to the iridium network.

2.3 UML Diagram

The UML Diagram displays the general Buoy Subsystems' flow as well as the interactions between individual system elements.



3 Acceptance Test Procedures

In order to evaluate the performance of the subsystem designs acceptance test procedures will be used to assess whether they fulfil the user and functional requirements.

3.1 Figures of merit

The first test will be to test that the STM processes the data correctly when working with the IMU and compare the running speeds with the simulated data. Then after the data has been processed it will need to be filtered and attenuate unwanted noise. A test to demonstrate data filtering speed will be done. This considers both the processing time required to compute the 25% value of the Fourier coefficients necessary to successfully filter the data as well as the time spent filtering the data packet itself.

The compression of data. The algorithm chosen uses very little power and optimizes to preserve battery life when processing the data. The aim with the lossless compression is so that no data is lost and that 25% of the Fourier coefficients can be extracted after decompression. To test for this the compressed file size should be around ±50% of the original file size. The time taken to compress must low so as to lower the processing done by the processor and thus saving battery.

The encryption of data also the run time of the algorithm must be low so as limit the processing done by the processor so as to save battery. The encryption of the data must be secure and should be only accessible to those possessing the unique key.

In terms of the transmission of data, the data rates are important, so the speed of transmission will be tested. The congestion of packets will also be tested, and packet loss will also be tested.

The STM must be able to function properly in the southern Antarctic Ocean and process the data correctly but the must a margin of error that must be considered.

3.2 Experiment Design of ATPs

Compression

In order to test the figures of merit for the ZLIB compression algorithm IMU datasets will be used. To test for simulation for the comparison data MATLAB will be used to test whether the simulated data matches the actual data. Using the deflate and inflate methods of the ZLIB library to compress and decompress the data for analyzation in relation to the original file. The C library has a time measuring unit which will be used as a benchmark for execution time and processes. To measure how much battery is used with each compression run, the voltage level of the battery will measure to assess how much battery the compression algorithm uses.

Encryption

Also, with the Encryption to test the figures of merit IMU datasets will be used and compared to the simulated datasets. AES algorithm is the best feasible encryption technique available for encryption and decryption of data. To test the validity of the encryption keys similar to the unique keys will be used to try and hack the file. From there is any of the guessing keys work then a new and even secure data encryption algorithm will be implemented. Just like the compression the C library clocks will be used to check the execution time of the algorithm. When the data is sent back to the computer after the loop in validation it will be decrypted and compared with the original data file.

3.3 Acceptable Performance definition

For the system design to be acceptable all the client's requirements must be satisfied. To recap some of the requirements first, the data needs to compressed using lossless compression to ensure that no data is lost, and it should be able to preserve 25% of the Fourier coefficients. The processing done by the processor must be limited so as to save battery life. The successful completion of implementation and demonstration using computer data through the setup of tests to verify the functionality of the code as well as implementation and demonstration using data from an on-board sensor demonstrating that the STM can process data from an on-board sensor will constitute satisfactory performance in this unit.

4. References:

[1]
"Compression - Encoding images - GCSE Computer Science Revision - BBC Bitesize," *BBC Bitesize*, 2022. https://www.bbc.co.uk/bitesize/guides/zqyrq6f/revision/4#:~:text=Compression%20can%20 (accessed Aug. 22, 2022).

[2]

Leah Fainchtein Buenavida, “Crunch Time: 10 Best Compression Algorithms,” *dzone.com*, May 28, 2020. https://dzone.com/articles/crunch-time-10-best-compression-algorithms (accessed Aug. 22, 2022).
[3]

“Understanding zlib,” *Euccas.me*, 2019. https://www.euccas.me/zlib/ (accessed Aug. 22, 2022).

[4]
E. Williams, “Cryptography 101: Symmetric Encryption - Emily Williams - Medium,” *Medium*, Mar. 30, 2020. https://medium.com/@emilywilliams_43022/cryptography-101-symmetric-encryption-444aac6bb7a3 (accessed Aug. 22, 2022).

[5]
“What is 3DES encryption and how does DES work? | Comparitech,” *Comparitech.com*, 2022. https://www.comparitech.com/blog/information-security/3des-encryption/ (accessed Aug. 22, 2022).

[6]
C. Bernstein and M. Cobb, “Advanced Encryption Standard (AES),” *SearchSecurity*, 2021. https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard (accessed Aug. 22, 2022).

[7]
“AES Encryption Library for Arduino and Raspberry Pi: AES library for Arduino and Raspberry pi.,” *Github.io*, 2015. http://spaniakos.github.io/AES/ (accessed Aug. 22, 2022).

[8]

J. Jacobson, “SHARC BUOY: Remote Monitoring of Ice Floes in Southern,” University of Cape Town, Cape Town, 2020.