University of Cape Town

EEE3097S

Design

Paper Design

Rory Schram
SCHROR002

Natasha Soldin
SLDNAT001

22/08/2022

Contents

1 Contributions

Each team member's contributions for the following paper design are tabulated in table 1 below.

Rory Schram	Compression Algorithm Compariosn
	Subsystem Design
	Timeine Development
Natasha Soldin	Encryption Algorithm Compariosn
	Requirement Analysis
	Aceptance Test Procedure

Table 1: Contribution Table

2 Introduction

The following project design is for an ARM based digital IP using an STM32F051-microcontroller. This design aims to retrieve, compress and encrypt data from an Inertial Measurement Unit (IMU) sensor. This type of sensor includes an Accelerometer, a Gyroscope, a Magnetometer and a Barometer. This design will be implemented as a buoy installed on an ice 'pancake' in the Southern Ocean to collect data about the ice and wave dynamics.

This data will then be transmitted using the Iridium communication network, which is a global satellite communications network. However, this is extremely costly and therefore the data would need to be compressed to reduce its size. The data is also encrypted for security purposes.

The following report includes a paper design which states the basic planning laid out for the aforementioned project. This involves a requirement analysis, subsystem design, acceptance test procedure and development timeline.

3 Requirement Analysis

The requirement analysis will include user cases and applications of IMU sensors, derivation of user requirements and their accompanying technical specifications, a comparison of compression techniques, a comparison of encryption techniques, a feasibility analysis and possible bottlenecks of the project.

3.1 User Cases

There are multiple user cases for a system that contains an IMU sensor [1] and reads off its data. A few of them are:

- A GPS system requires movement data to determine the location and speed of a car to plot routes to a destination. The IMU's accelerometer would provide information regarding the speed of the car and the magnetometer would provide information regarding the direction of the car. This data would require compression as it involves satellite transmission and would require encryption as it is private information.
- Virtual reality (VR) or augmented reality (AR) systems require movement data to describe the head movement and orientation of the 'player'. The IMU's accelerometer would provide information regarding the speed of the head movements and the gyroscope would provide information regarding the orientation of the device. This data requires compression as it is large in size.
- Drones require movement data to aid in flight control, stability and guidance. The IMU's accelerometer would provide information regarding the speed of the drone, the magnetometer would provide information regarding the direction of the drone and the gyroscope would provide information regarding the orientation of the drone. This data would require compression as it often involves satellite or wireless transmission and would require encryption as drone application often involves the use of sensitive data (i.e. military applications).

3.2 Requirements

1. The project should be designed to utilize the ICM-20649 IMU sensor even though that sensor is not available to use during the project's planning stage.
2. The project should be designed to utilize the STM32F051 microcontroller.
3. The data obtained from the IMU sensor should be compressed to reduce the cost of transmission because the transmission of data using Iridium is extremely costly.
4. The data obtained from the IMU sensor should be encrypted to increase security of the data.
5. Minimal loss/ damage should apply to the decrypted and decompressed data.
6. Limit power consumption by reducing the amount of processing done in the processor and minimizing the computation required.

3.3 Specifications

1. The design should adhere to the electrical specifications in figure 3.3.1 below as obtained from the ICM-20649's datasheet.

Typical Operating Circuit of section 4.2, VDD = 1.8 V, VDDIO = 1.8 V, T_A = 25°C, unless otherwise noted.

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
SUPPLY VOLTAGES						
VDD		1.71	1.8	3.6	V	1
VDDIO		1.71	1.8	3.6	V	1
SUPPLY CURRENTS						
Gyroscope Only (DMP & Accelerometer disabled)	Low-Noise Mode		2.67		mA	2
Accelerometer Only (DMP & Gyroscope disabled)	Low-Noise Mode		760		µA	2
Gyroscope + Accelerometer (DMP disabled)	Low-Noise Mode		3.21		mA	2
Gyroscope Only (DMP & Accelerometer disabled)	Low-Power Mode, 102.3 Hz update rate, 1x averaging filter		1.23		mA	2, 3
Accelerometer Only (DMP & Gyroscope disabled)	Low-Power Mode, 102.3 Hz update rate, 1x averaging filter		68.9		µA	2, 3
Gyroscope + Accelerometer (DMP disabled)	Low-Power Mode, 102.3 Hz update rate, 1x averaging filter		1.27		mA	2, 3
Full-Chip Sleep Mode			8		µA	2
TEMPERATURE RANGE						
Specified Temperature Range	Performance parameters are not applicable beyond Specified Temperature Range	-40		+85	°C	1

Figure 3.3.1: ICM-20649 Electrical Specifications

2. Both algorithms needs to be coded in C/C++ in STM32CUBEIDE to be compatible with the STM32F051 microcontroller.

3. The compression specifications are as follows:

   (a) The compression and decompression of the IMU data will be done using a dictionary compression method as described in section 3.4 below.

   (b) The compression ratio should be between 40% and 60%.

4. The encryption specifications are as follows:

   (a) The encryption and decryption of the IMU data will be done using an AES encryption method as described in section 3.5 below.

   (b) The encryption key should be either 128, 192 or 256 bits to ensure adequate security.

5. The decrypted and decompressed data should reflect 25% of the Fourier coefficients of the original IMU data

6. The C/C++ programs should be of time complexity order n [O(n)] to reduce the amount of processing done and thus the power consumption.

3.4 Comparison of Compression Algorithms

Time series data is becoming more and more prevalent in today's data-centric world. The single largest issue with this is how to deal with the massive amounts of data produced, and consequently, how to handle storing this data. Systems with high write rates, such as IoT sensors,

can cause massive storage concerns. Any device that is writing constantly and producing time series data, over time may need a large amount of storage space. For example, a Boeing 787 may generate half a terabyte of sensor data per flight. [2] This fact reiterates the importance compression when it comes to utilizing time series data in your projects. Compression of the IMU data is especially important for this project, because the cost of data transmission is so incredibly high due to the Iridium Satellite Communication network being utilized for data transmission. While this network has the impressive capability of being able to communicate with every remote locations all around the globe, it comes at a high monetary cost. This cost can be reduced dramatically by proper implementation of compression algorithms on the time series data that has been produced by the IMU module. The following comparison will delve into what exactly compression is, how it it is implemented and which algorithms where considered for this project.

Generally, compression algorithms take advantage of particular properties of time series data, that can then be exploited to reduce the overall size of the data set. These properties can be broken into 3 categories, as seen below. [3]

- **Redundancy**: Sometimes certain portions of data appear more often than other portions of data in time series data, this can be exploited for our purposes by leaving out these repeated portions of data and instead representing them with a piece of data that takes up much less space.
- **Approximabilty**: Often sensor can end up producing time series data that can be approximated by function that have been designed/calculated beforehand.
- **Predictabilty**: In today's ever more advanced digital world, sometimes it is possible to predict future time series data using deep learning neural networks.

There are a large number of compression algorithms available for use, with every one differing drastically in complexity, functionality and efficiency. Every compression algorithm can be described as one of the following. [3]

- **Non-adaptive or adaptive**: A non-adaptive algorithm does not need to train on a data set beforehand to work efficiently and properly, whereas an adaptive algorithm learns from past data to compress the newer data being produced.
- **Lossy or Lossless**: To explain this concept, lets assume that you have a data set D_1, which has a size determined to be the number of bits required to store the data set, defined as s_1, and a compression algorithm defined as the function F, which takes in a data set and returns that same data set after the compression algorithm has been applied. Therefore, $F(D_1) = D_1^{'}$ and the size of $D_1^{'}$ is $s_1^{'}$. For a compression algorithm to be of any use, $s_1^{'} < s_1$. A lossless compression algorithm is one where the inverse compression algorithm produces the same data set as the one that was originally input into the algorithm. Therefore, $F^{-1}(D_1^{'}) = D_2$, where $D_2 = D_1$ the size of the output $s_2 = s_1$. If, after compression and decompression, $D_2 = D_1$ and $s_2 = s_1$, the compression algorithm is said to be lossless. If $D_2 \neq D_1$ and $s_2 < s_1$ then the algorithm is said to be lossy.

- **Symmetric or Non-symmetric**: If a compression system uses the same algorithm to compress and decompress data, only changing the direction of data flow, then it is said to be symmetric. If a compression system uses completely different algorithms for compression and decompression, then it is said to be non-symmetric.

As mentioned above, there are multiple types of compression algorithms. Below is an overview of different types of compression algorithms, each one incorporating various features mentioned above. [3]

1. **Dictionary Based**: This compression principle is based off of the fact that over a long time, time series data sets will have repeated common segments. After an initial training phase on a similar data set, a dictionary of segments will be created. The compression algorithm will then search through this dictionary and if a segment is found in the data that matches what is in the dictionary, then that length of text will be be extracted, and instead it will be replaced by an index that can be traced back to the dictionary.

2. **Functional Approximation**: The idea behind functional approximation is that time series data can be represented by a mathematical function. Finding a mathematical function for the entire time series would not be possible however, so what happens is the data set is split up into little segments, and an approximation for these segments is made. Since devising a function that would perfectly match the data set is a very process heavy procedure, the best that can be done is a close approximation. This method greatly reduces the data set size, because the only data that needs to be stored are the functions of approximation themselves, however this results in lossy compression, because of the multiple functions only roughly approximating the time series data. The strength behind this method however, is that no training needs to take place, the algorithm will work out of the box.

3. **Autoencoders**: Autoencoders are unsupervised machine learning networks that apply backpropagation. [4] Autoencoders aim to take input data and compress it into a smaller deconstructed form, called the code, which can then be rebuilt at a later stage with the same neural network, just operating in the reverse direction.

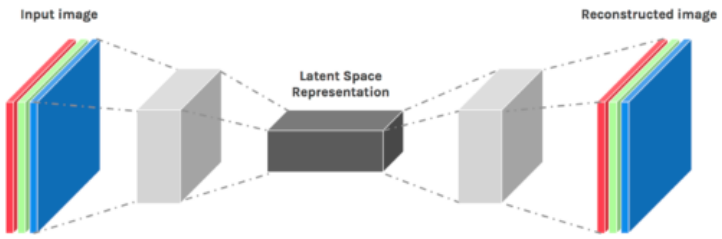

Figure 3.4.1: Overview of autoencoded compression [4]

4. **Sequential Algorithms**: Sequential algorithms, at their base level, incorporate several of the best compression techniques into one large algorithm. Some of the most popular techniques include, but are not limited to, the following:

- Huffman coding
- Delta encoding
- Run-length encoding
- Fibonacci binary encoding

Combining these algorithms, whether using many of them, or simply two of them, results in a very robust and efficient compression algorithm.

After much consideration, it was determined that the most important compression aspects were the following:

1. The compression algorithm needs to be able to run on the STM32F051 in a C/C++ implementation.

2. The compression needs to be lightweight, in other words, it cannot take up the limited resources available for use on the STM32F051.

3. The compression must be simple. This is to reduce the time of development, and to save on resources on the STM32F051 micro-controller.

Bearing all of the prior information in mind, the compression technique that fits all of the requirements best would be the dictionary type compression. This is because dictionary compression is lightweight, fast, and simple to implement. The main types of dictionary compression are Tristan, Conrad, A-LZSS, and D-LZW. Due to these techniques being the most suitable for the compression needs of this projects, one of them will be used.

3.5 Comparison of Encryption Algorithms

Encryption methods can be subdivided into two categories [5]:

- Symmetric Encryption: encryption that involves a single key to encrypt and decrypt data. This a slighlty older technique that offers less security of data.

- Asymmetric Encryption: encryption that involves two keys, a public key and a private key that are mathematically linked together. Data is encrypted using the public key but can only be decrypted by the private key. This more modern method offers a greater level of security to the data but is more complicated to implement.

Although Asymmetric encryption is safer and offers a greater level of encryption to the data, it is more complicated to implement and for this context unnecessary. It is not necessary as the system will be deployed in a remote area of the Southern Ocean and the data collected by the sensor is not sensitive or at risk of tampering/ theft. Thus, Symmetric encryption will suffice.

There are several Symmetric encryption methods to consider, of which three will be examined in this comparison: Advanced Encryption Standard (AES), Data Encryption Standard (DES) and Triple Data Encryption Standard (3DES). Some of the key factors of the above encryption methods are tabulated in table 2 below: [6] [7]

Key	**AES**	**DES**	**3DES**
Definition	Advanced Encryption Standard	Data Encryption Standard	Triple Data Encryption Standard
Designed By	Vincent Rijmen and Joan Daemen	IBM	IBM/ Walter Tuchman
Year	2001	1978	1981
Key Length	128, 192, 256 bits	56 bits	112, 168 bits
Size	128 bits	64 bits	64 bits
Security Level	Most secure	Less secure	More secure than DES
Derived from	Square Cipher	Lucifer Cipher	DES
Memory Used	14,7 KB	18,2 KB	20,7 KB

Table 2: Comparison between Symmetric Encryption Methods

DES is no longer in use as it is considered to be insecure. This insecurity stems from its short 56-bit key. 3DES [8], although more secure than DES, is less commonly used today. The national Institute of Standards and Technology (NIST) recently released a proposal stating that all forms of 3DES will be deprecated through 2023 and disallowed from 2024. Therefore for longevity purposes this encryption method will not be used. Thus, AES will be used because it is commonly used today, it offers an adequate level of security and is lightweight in nature which reduces the resource usage on the STM32F051.

3.6 Feasibility Analysis

Upon consideration of the project description and its requirements, the project is decided to be feasible. For the following reasons:

- There are many readily available predefined compression and encryption libraries that can be implemented in this project. Which greatly reduces the complexity of the project and time needed to complete the project.
- The project description is relatively simple and straightforward.
- The timeline given for the project is reasonable.

- Similar projects have been completed and successfully implemented in the past.

The possible bottlenecks specified in section 3.7 below greatly impact the feasibility of the project. Majority are due to the fact that previous projects utilised raspberry pi and python which in combination offer more memory space, easier data retrieval and better algorithm variety.

3.7 Possible Bottlenecks

The project is vulnerable to the following possible bottlenecks:

- Student's lack of knowledge surrounding C/C++ programming.
- Limitations of the STM320F0 microcontroller's memory space and computing power.
- Data procurement from sensor resulting in faulty/ incorrect data.
- Data transmission to different sub-systems of the project could result in losses or faults.
- Incompatibility of data format with compression and encryption libraries.

4 Subsystem Design

For this project, the following major subsystems exist:

- The STM32F051 micro-controller unit
- The IMU (Inertial Measurement Unit)
- External computing device (personal laptop)

These subsystems and their relationships can be graphically shown in figure 4.0.1 below:

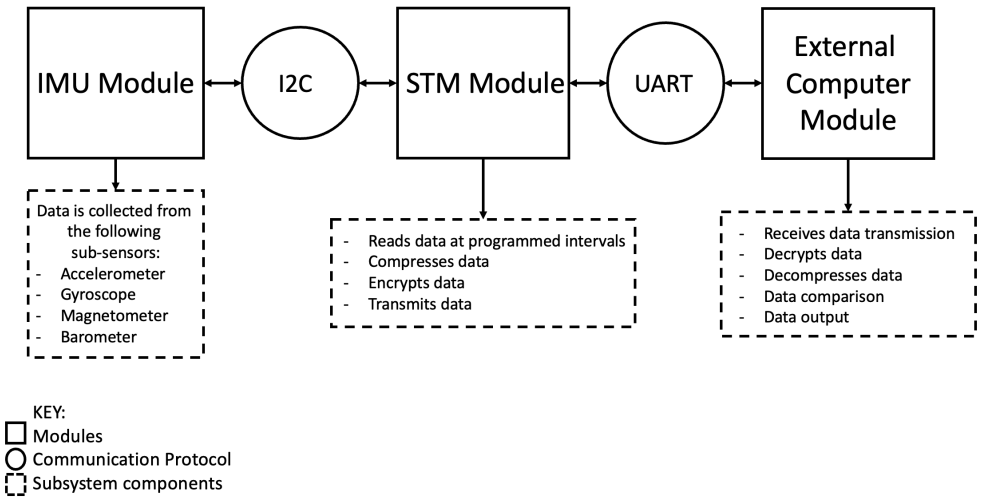

Figure 4.0.1: Block Diagram of Subsystems

4.1 Sub-subsystems

4.1.1 STM32F051 Micro-controller Subsystems

The STM32 micro-controller has many subsystems that will operate in this project. The two most important ones being the compression and encryption blocks. These are the basis behind the entire project and form the largest portion of required design work. Another large subsystem of the STM32 is the communication protocols. This can be broken up into the communication between the STM32 and the external computing device, and the communication between the STM32 and the IMU sensor itself.

4.1.2 The IMU Subsystems

The main subsystem of the IMU is the communication protocol between the IMU itself and the STM32. This communication protocol is vital for the reading of the data from the IMU module.

4.1.3 External Computing Device Subsystems

The external computing device, similar to the IMU, has to communicate with the STM32. This communication is one of the subsystems. The other subsystems are the de-encryption and de-compression of the received data from the STM32 module.

4.2 Subsystem and Sub-subsystems Requirements

4.2.1 STM32F051

Compression:

1. The STM32 must be able to compress the data that has been read from the IMU module.
2. The compression algorithm must be able to save storage space. This is necessary because time series data can take up a lot of storage space.
3. This compression must not be resource heavy so as not to take up to much computing power due to the limited resources available on the STM32.
4. The compression algorithm needs to preserve the first 25% of the Fourier coefficients that are contained within the raw data read from the IMU module.

Encryption:

1. The STM32 must encrypt the compressed IMU module data for security purposes.
2. This encryption should, similarly to the compression, not be to resource heavy, due to the STM32 having limited resources available.

Communication Protocols:

1. The STM32 must be able to communicate with the IMU module. This module is a digital sensor and so a digital communication protocol will have to be implemented to allow for this communication to happen.
2. The communication between the STM32 and the IMU module must happen at regular, controlled intervals.
3. Communication between the STM32 and an external computing device is a vital requiremnt for this project. This communication needs to cater for two-way communication.This communication must be two-way to allow for the compressed and encrypted data to be sent to the external computing device for processing, and on the other hand, to allow for data to be sent to the STM32 itself for acceptance test procedures.

4.2.2 IMU Module

Communication Protocol:

1. The IMU module needs to communicate with the STM32 module at regular intervals for data transmission purposes. As stated in the previous requirments, this communication must happen at regular, controllable intervals.

4.2.3 External Computing Device

Communication Protocol:

1. The external computing device needs to communicate with the STM32 module. As stated before, this communication protocol must allow for two-way communication for data retrieval and testing purposes.

De-encryption:

1. The external computer needs to be able to properly de-encrypt the transmitted data from the STM32. This decryption should return the compressed data.

De-compression:

1. The external computer needs to be able to properly de-compress the data after decryption. This decryption should be fast and efficient.

2. The de-compression should preserve 25% of the Fourier coefficients contained within the data.

4.3 Subsystem and Sub-subsystems Specifications

4.3.1 STM32F051

Compression:

1. The compression algorithm that will be used is part of the family of compression algorithms known as dictionary compression.

2. The compression algorithm will achieve a compression ratio of between 40% and 60%.

3. Dictionary compression is a very lightweight compression algorithm because it doesn't deal with mathematical operations as much as the other algorithms. Due to the limited resources available on the STM32, it makes sense to impliment a dictionary based compression algorithm.

4. The compression algorithm will be a lossless implementation, so as to perfectly preserve the first 25% of the Fourier coefficients contained within the raw data, and more.

Encryption:

1. AES encryption will be performed on the data obtained.

2. The AES encryption method is very lightweight and secure. This method only takes up 14,7KB of memory and offers multiple key length choices. 2

Communication Protocols:

1. The I2C communication protocol will be used in this project for communication between the STM32 and the digital IMU module.

2. I2C communication has many different speed modes. These are as follows, standard mode: 100 kbit/s, full speed: 400 kbit/s, fast mode: 1 mbit/s, high speed: 3,2 Mbit/s. [9]

3. The communication between the STM32 and the IMU module will be controlled by a timing structure governed by the internal clock of the STM32. This allows for precise timing of the data reading.

4. Communication between the STM32 and an external computing device will utilize the UART (Universal Asynchronous Receive Transmit) communication protocol. An FTDI device will be utilized to convert the UART signal to USB which is what our personal laptops utilize. As the name suggests, UART allows for two-way communication. UART allows for a maximum data transmission speed of around 5 Mbps.[10]

4.3.2 IMU Module

Communication Protocol:

1. As mentioned beforehand, the IMU module will be connected into the system via the I2C communication protocol with measurements being taken at pre-defined intervals controlled by the onboard oscillator of the STM32.

4.3.3 External Computing Device

Communication Protocol:

1. As mentioned beforehand, the communication between the STM32 and the external computing device will take place over the UART communication protocol.

De-encryption:

1. Decryption of the AES level encrypted data will take place on the external computing device.

De-compression:

1. De-compression will take pace on the external computer. This process will be as fast as the compression process and will produce the exact same data as was initially compressed, due to the fact that dictionary, lossless compression is being used. A fully lossless compression algorithm choice means that all of the first 25% of the Fourier components contained within the data can be extracted.

4.4 Inter-Subsystem and Inter-Sub-subsystems Interactions

The main subsystems in the project are as follows:

1. STM32F051 Micro-controller.
2. The IMU module.
3. External computer.

Contained within these subsystems are the following sub-subsystems:

1. Compression/de-compression
2. Encryption/decryption
3. Communication protocols

The main interactions of the entire system are as follows. The IMU module performs measurements that are then stored in the memory addresses of the module itself. The measurements are read into the STM32 via the I2C communication protocol. The data is then processed on the STM32, this processing involves formatting the raw data, compressing this formatted data, and then encrypting this data using the aforementioned compression and encryption algorithms. The encrypted data is then to be transmitted to the external computing device. This communication takes place via the UART communication protocol. Once the encrypted data is on the external computing device, decryption and de-compression takes place. The data is then stored for viewing and even more processing to take place. Additional processing could be, for example, performing Fourier transforms to analyse the data in the frequency domain.

The last interactions that may take place are for acceptance test procedures. This involves taking the IMU module out of the system itself and feeding data to the STM32 directly. This would involve the following. Raw data on the external computing device is sent over the UART communication protocol to the STM32. The STM32 compresses and encrypts the data sent from the external computer and then sends it back to the external computer whereupon decryption and de-compression can take place. The original data can then be compared to the recently received data and a comparison can take place.

4.5 UML Diagram

The UML diagram in figure 4.5.1 below is similar to the block diagram in figure 4.0.1 above. It differs in that it is only concerned with the STM32F051 micro controller and external computer modules and incorporates more technical coding-related terms and structures.

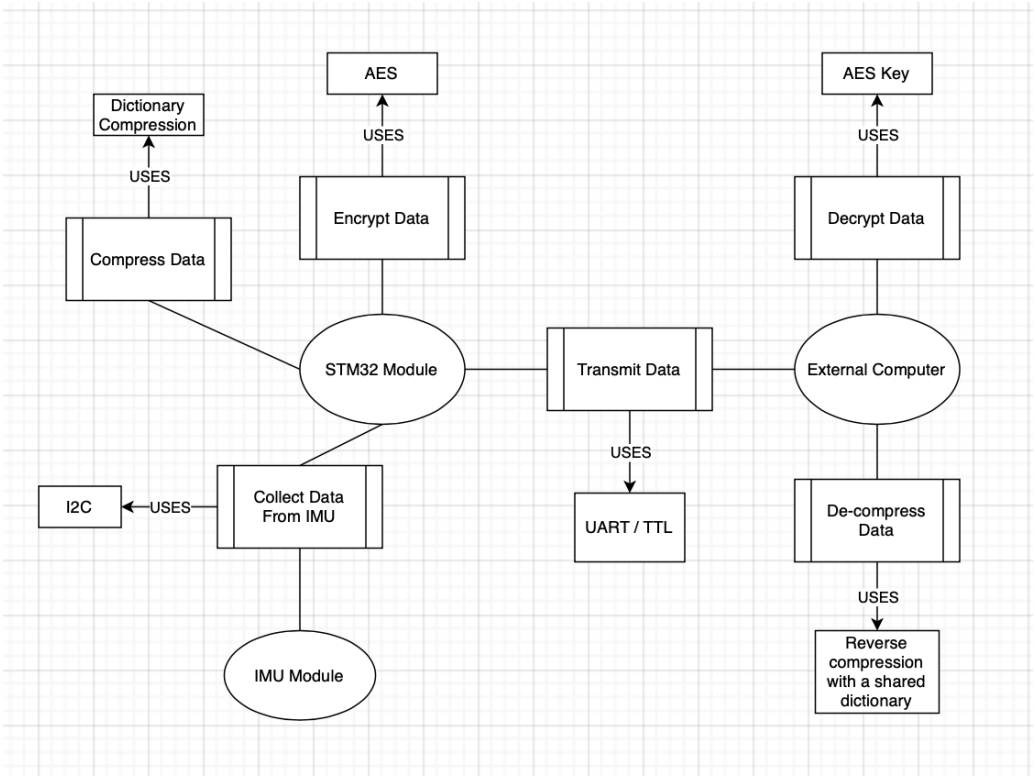

Figure 4.5.1: UML diagram depicting system interactions

5 Acceptance Test Procedure

5.1 Acceptance Performance Definition

An acceptance test procedure (ATP) was derived for each specification to test whether or not the project meets the specifications.

1. Ensure that the data obtained from the sensor hat closely follows the structure of the sample data provided as this comes from the design's intended sensor (ICM-20649 IMU).
2. Run the algorithms in a C/C++ IDE as well as on the STM32F051 to test that it works.
3. The compression ATPs are as follows:
   (a) Check that the file size of the compressed data is considerably less than the file size of the original data.
   (b) Calculate the compression ratio between the uncompressed and compressed data.

$$ratio = \frac{uncompressed}{compressed}$$

4. The encryption ATPs are as follows:
   (a) Compare the encrypted data to the un-encrypted data to ensure minimal similarities which indicates reasonable level of encryption.
   (b) Ensure decrypted data is exactly the same as the data before encryption to make sure that no data loss has occured during the encryption phase.
5. The fourier transform of the decrypted and decompressed data should be compared to the fourier transform of the original data to test if 25% of the lower fourier co-efficients have been retained.
6. Test the time complexity of the compression and encryption programs by running multiple tests with different dataset sizes and plot the correlation.

5.2 Figures of Merit

Figures of merit are the quantities used to describe the efficiency of a device. It is challenging to set numerical values to the experiment during the planning phase and therefore these figures of merit will act as an estimate but are subject to change when project implementation begins. The firgures of merit are:

- The decrypted and decompressed data should reflect 25% of the Fourier coefficients of the original IMU data.
- The encryption algorithm should use a 128, 192 or 256 bit key size.
- The compression algorithm should have a compression ratio of between 40% and 60%.

5.3 Experiment Design

5.3.1 Compression

The compression algorithm will be tested to determine its efficiency and accuracy.

Method

- The compression algorithm will run on a dataset of known size and a time measurement will be taken for the program's runtime.
- This will be repeated on 5 datasets of varying size.
- A graph will be plotted to determine the time complexity of the compression algorithm. This graph will plot dataset size vs. runtime.
- The compressed file will then be uncompressed and compared to the original data to test for signs of data loss.
- The compression ratio will be calculated and compared to the ideal case.
- The Fourier transform of the compressed and uncompressed data are compared to ensure that 25% of the Fourier coefficients are retained.

5.3.2 Encryption

The encryption algorithm will be tested to determine its its efficiency and accuracy.

Method

- The encryption algorithm will run on a dataset of known size and a time measurement will be taken for the program's runtime.
- This will be repeated on 5 datasets of varying size.
- A graph will be plotted to determine the time complexity of the encryption algorithm. This graph will plot dataset size vs. runtime.
- The encrypted file will then be decrypted and compared to the original data to test for signs of data loss or tampering.
- The Fourier transform of the encrypted and decrypted data are compared to ensure that 25% of the Fourier coefficients are retained.

6 Development Timeline

This project was managed using Asana as its project management software. With which a timeline was created that helped the team members stay up to date on tasks and deliverables. Screenshots of the timeline 6.0.1 and task list 6.0.2 are attached below:

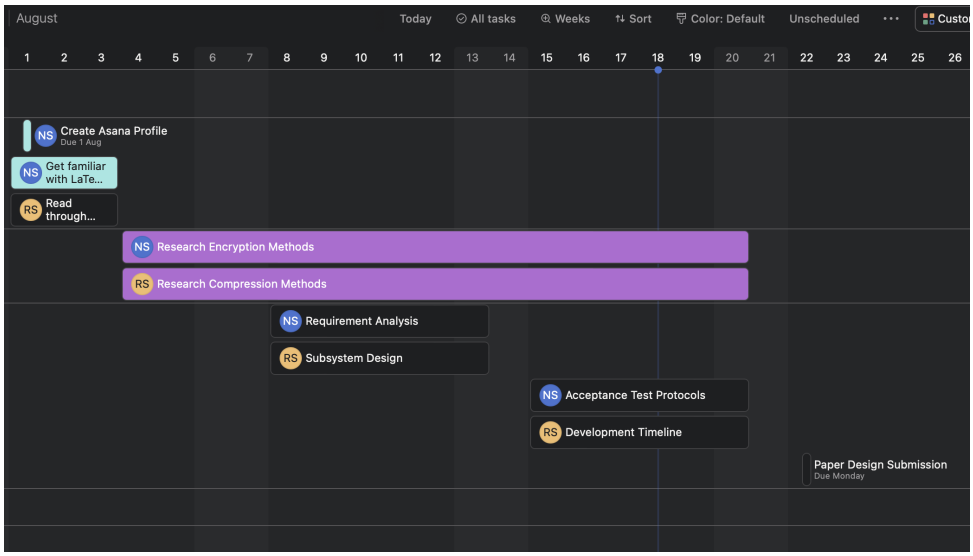

Figure 6.0.1: Asana Timeline

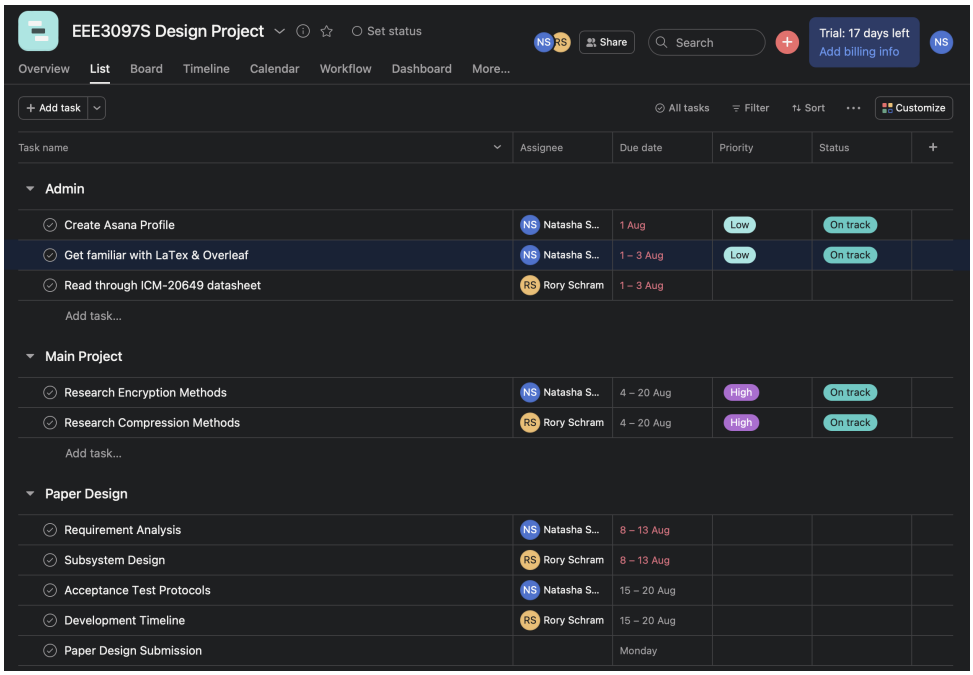

Figure 6.0.2: Asana Task list

References

[1] "The importance of imu motion sensors," Sep 2019.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of things applications: A systematic review," *Computer Networks*, vol. 148, pp. 241–261, 2019.

[3] G. Chiarot and C. Silvestri, "Time series compression: a survey," *ArXiv*, vol. abs/2101.08784, 2021.

[4] S. Deb, "How to perform data compression using autoencoders?," Sep 2020.

[5] SSL2BUY Wiki, "Symmetric vs. asymmetric encryption - what are differences?." `https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences`, 2021.

[6] Tutorials Point, "Difference between AES and DES ciphers." `https://www.tutorialspoint.com/difference-between-aes-and-des-ciphers`, 2021.

[7] M. N. A. Wahid, A. Ali, B. Esparham, and M. Marwan, "A comparison of cryptographic algorithms: Des, 3des, aes, rsa and blowfish for guessing attacks prevention," *Journal of Computer Science Applications and Information Technology*, 2018.

[8] J. Lake, "What is 3des encryption and how does des work?," Feb 2022.

[9] I2C-Bus, "Speed."

[10] S. Mishra, N. K. Singh, and V. Rousseau, "Chapter 10 - sensor interfaces," in *System on Chip Interfaces for Low Power Design* (S. Mishra, N. K. Singh, and V. Rousseau, eds.), pp. 331–344, Morgan Kaufmann, 2016.