University of Cape Town

EEE3097S

Design

Progress Report 02

Rory Schram
SCHROR002

Natasha Soldin
SLDNAT001

05/10/2022

Contents

1 Introduction **1**

2 Admin **1**
2.1 Contributions 1
2.2 Project Management Tool and Timeline 1
2.3 GitHub 2

3 IMU Module **3**
3.1 IMU Comparison 3
3.2 IMU Validation Tests 4
3.3 IMU Validation Results 6

4 Experiment Setup **10**
4.1 Overall System 11
4.2 Compression Block 11
4.3 Encryption Block 12

5 Results **13**
5.1 Overall System 13
5.2 Compression Block 13
5.3 Encryption Block 15

6 Acceptance Test Protocols **16**

1 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 progress report which states the practical focused experiment or final attempt of project software and hardware implementation. It includes an IMU module, experiment setup, results and acceptance test procedures.

2 Admin

2.1 Contributions

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

Rory Schram	Compression Algorithm
	Experiment Setup
	Results
Natasha Soldin	Encryption Algorithm
	IMU Module
	Acceptance Test Procedure

Table 1: Contribution Table

2.2 Project Management Tool and 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 are attached in figure 2.2.1 below. The project is on track.

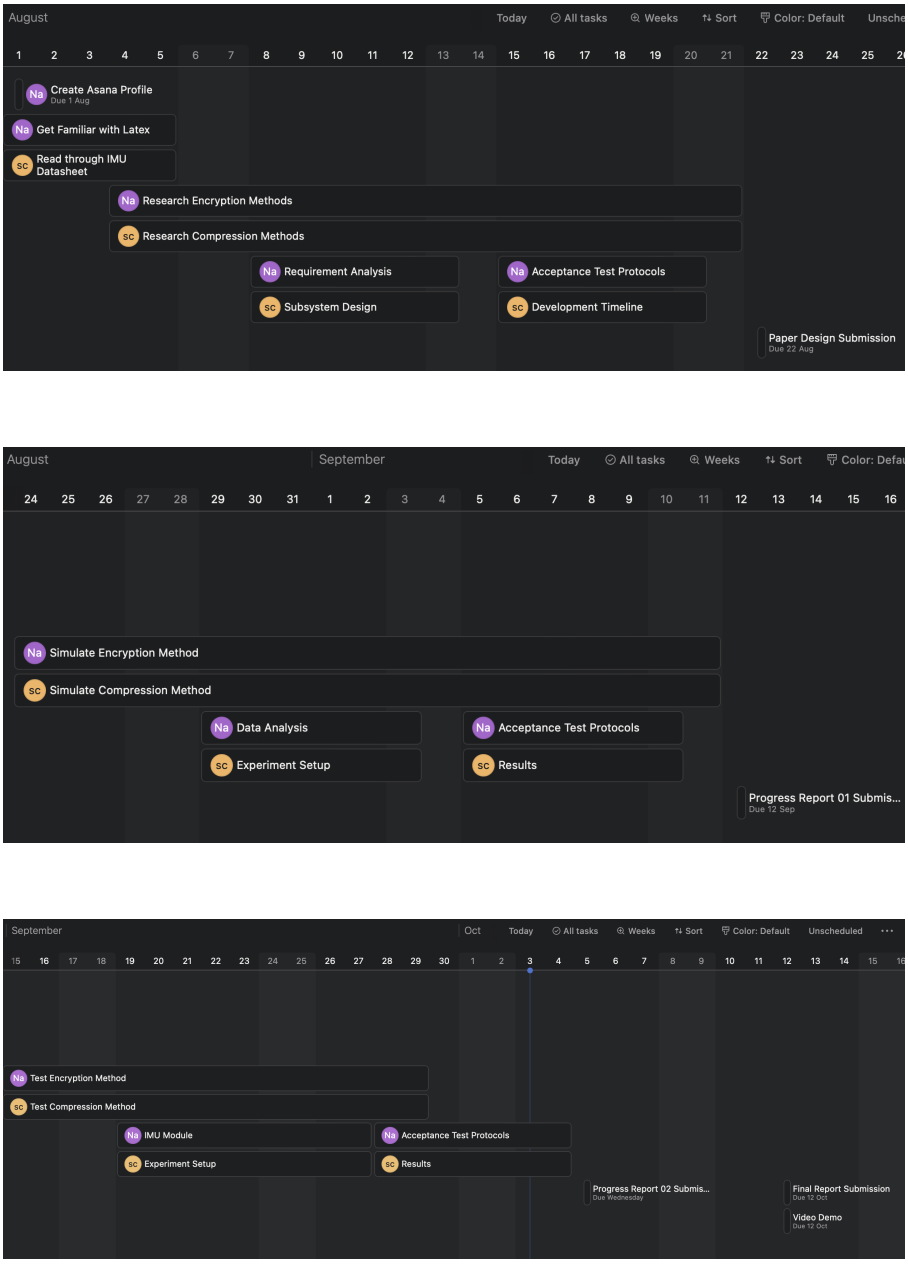


Figure 2.2.1: Asana Timeline

2.3 GitHub

The git repository can be accessed here.

3 IMU Module

3.1 IMU Comparison

The project is intended to use the ICM-20649 IMU sensor but the project utilizes the ICM-20948 IMU sensor due to availability constraints. The following table 2 summarizes the key differences between to the sensors:

	ICM-20649 IMU	**ICM-20948 IMU**
Device	6-Axis MEMS Motion Tracking Device	9-Axis MEMS Motion Tracking Device
Supply Voltage	1.71V - 3.6V	1.95V - 3.6V
Temperature Range	- 40 °C to +85 °C	- 40 °C to +85 °C
Gyroscope	Digital-output X-, Y-, and Z-axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ± 500 dps, ± 1000 dps, ± 2000 dps, and ± 4000 dps and integrated 16-bit ADCs	Digital-output X-, Y-, and Z-axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ± 250 dps, ± 500 dps, ± 1000 dps, and ± 2000 dps, and integrated 16-bit ADCs
Accelerometer	Digital-output X-, Y-, and Z-axis accelerometer with a programmable full scale range of $4g$, $8g$, $16g$, and $30g$ and integrated 16-bit ADCs	Digital-output X-, Y-, and Z-axis accelerometer with a programmable full scale range of ± 2g, ± 4g, ± 8g, and ± 16 g, and integrated 16-bit ADCs
Magnetometer	N/A	3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator. Output data resolution of 16-bits and full scale measurement range is ± 4900 μT
Low Pass Filter	Digitally-programmable	User-selectable
Communications	I2C at up to 100 kHz (standard-mode) or up to 400 kHz (fast- mode), or SPI at up to 7 MHz.	I2C at up to 100 kHz (standard-mode) or up to 400 kHz (fast- mode), or SPI at up to 7 MHz.
Shock Tolerance	10,000g shock tolerant	20,000g shock tolerant
Application	High Impact Application	N/A

Table 2: IMU Sensor Comparison

Specification 1 (S1) in table 5 below ensures IMU sensor compatibility between the project and the intended ICM-20649 IMU sensor. This specification ensures that the electrical specifications of the project align with the electrical specifications of the intended ICM-20649 IMU sensor as shown by figure 3.1.1 below.

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

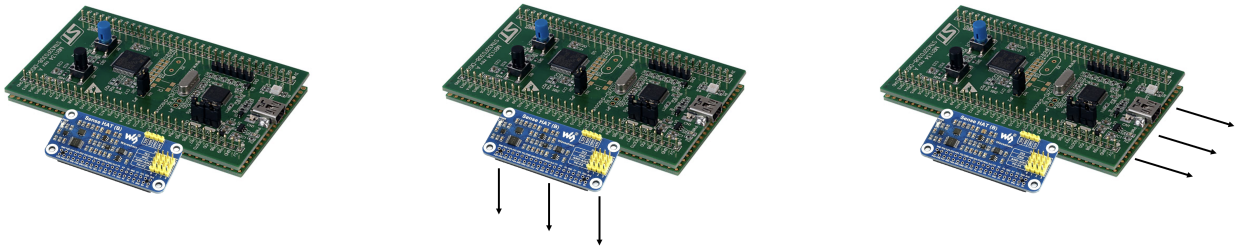
Figure 3.1.1: ICM-20649 Electrical Specifications

The above ensures that extrapolation between the projects current implementation and future intended implementation is as seamless as possible.

3.2 IMU Validation Tests

The sensor data validated and analyzed in the following IMU module was the accelerometer data as this is one of the sensors on the ICM-20649 IMU sensor that this project is concerned with. Therefore the additional sensors on the ICM-20948 IMU can be ignored at this stage of the design.

- **V1 - Hardware Validation:** when correctly connected the LED on the IMU should be illuminated signaling correct configuration. After the powering up of the STM32, the I2C interface writes to certain memory addresses to take the IMU module out of sleep mode. This follows the 'start-up' sequence and then data should be able to be extracted from the IMU.
- **V2 - Sensor Validation - Accelerometer Validation:**
  - **V2.1 - Motion Validation:** the acceleration readings should be read for three cases: the system should be left stationary on a flat surface, dropped and allowed to free fall and moved from a stationary position with effective acceleration. As shown by figure 3.2.1 below:



(a) Stationary Position (b) Free Fall (c) Accelerated in one direction

Figure 3.2.1: Accelerometer Validation

– **V2.2 - Rotation Validation:** the acceleration readings should be read for two cases: the system should be rotated along the x-axis and then rotated along y-axis. As shown by figure 3.2.2 below:

(a) x-Axis Rotation (b) y-Axis Rotation

Figure 3.2.2: Gyroscope Validation

– **V2.3 - Pendulum Movement:** a common movement for IMU validation is to attach the system to a pendulum and allow it to swing. As shown by figure 3.2.3 below:

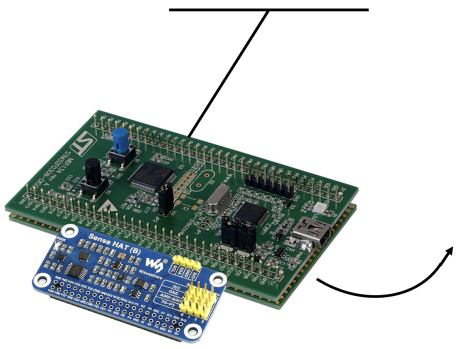


Figure 3.2.3: Pendulum Movement

3.3 IMU Validation Results

- **V1 - Hardware Validation:** The results of the hardware validation can be seen in figure 3.3.1 below. The IMU's light is illuminated when connected signalling correct configuration:

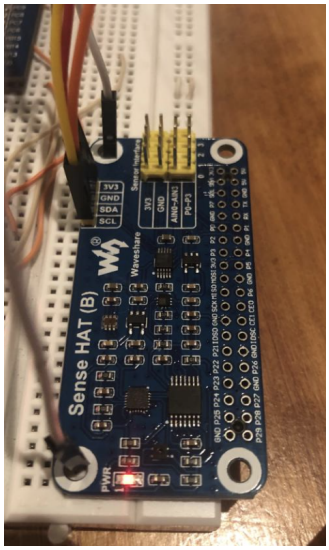


Figure 3.3.1: IMU Connection

- **V2 - Sensor Validation - Accelerometer Validation:**
  - **V2.1 - Motion Validation:** the results from experiment a) the stationary position experiment above can be graphically plotted as shown by the graphs in figure 3.3.2 below:

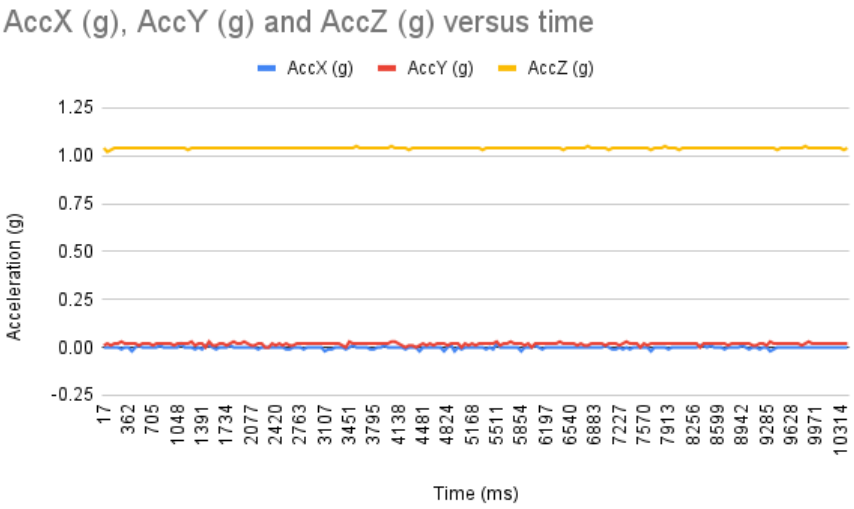


Figure 3.3.2: Accelerometer Validation Results 2.1a - Stationary Position

Figure 3.3.2 above demonstrates that when the IMU module is in a stationary, stable environment with no external forces except for gravitational force, the x and y acceleration is 0 and the z acceleration is constant due to gravity.

The results from experiment b) the free fall validation experiment above can be graphically plotted as shown by the graphs in figure 3.3.3 below:

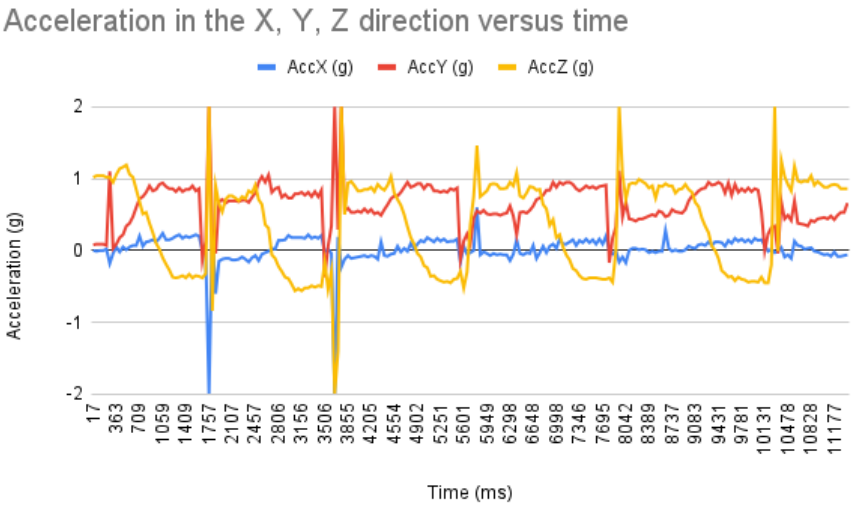
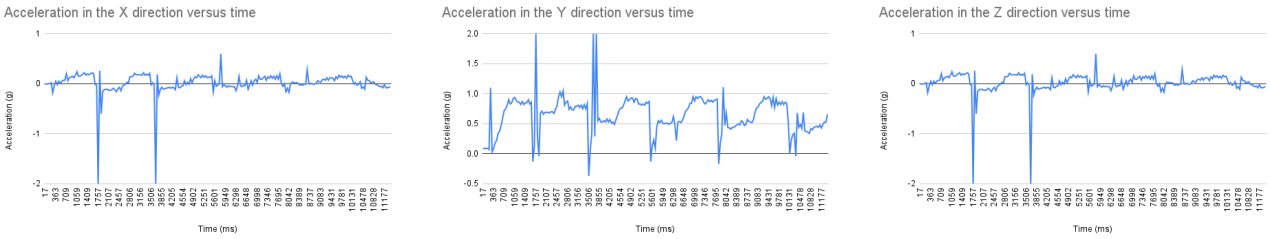


Figure 3.3.3: Accelerometer Validation Results 2.1b - Free Fall

The acceleration values can be separated into their x, y and z components and plotted separately shown in figure 3.3.4 below:



(a) x component of acceleration (b) y component of acceleration (c) z component of acceleration

Figure 3.3.4: Accelerometer Validation Results 2.1b - Free Fall

Above are the x, y, and z components of acceleration after a simple drop test was applied to the IMU module. The module was lifted and dropped onto a hard surface repeatedly over a time period of about 10 seconds. This resulted in hard spikes in the acceleration in the 3 different axis directions as a result of free fall's sudden halt.

The results from experiment c) the uni-directional acceleration validation experiment above can be graphically plotted as shown by the graphs in figure 3.3.5 below:

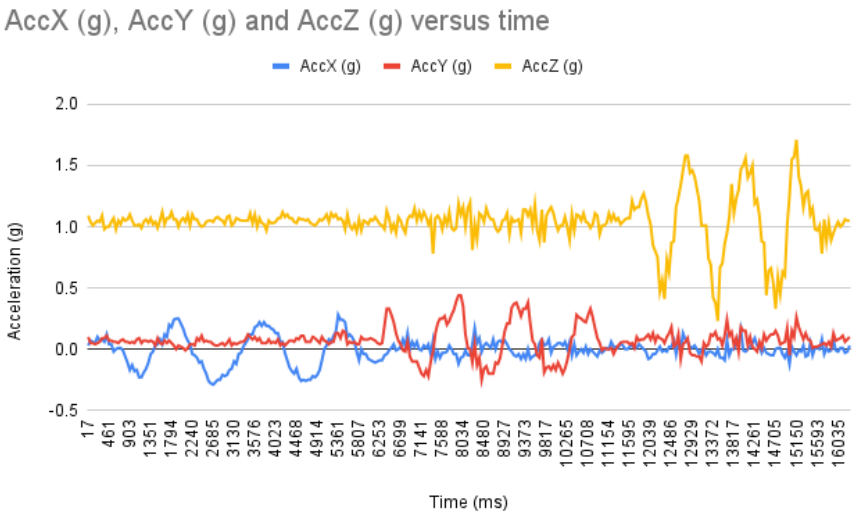


Figure 3.3.5: Accelerometer Validation Results 2.1c - Straight Line Movement

As can be seen, the acceleration in the different x, y, and z directions is increasing and decreasing as linear movement is experienced by the IMU at varying accelerations.

– **V2.2 - Rotation Validation:** the results from the rotation validation experiment above can be graphically plotted as shown by the graphs in figure 3.3.6 below:

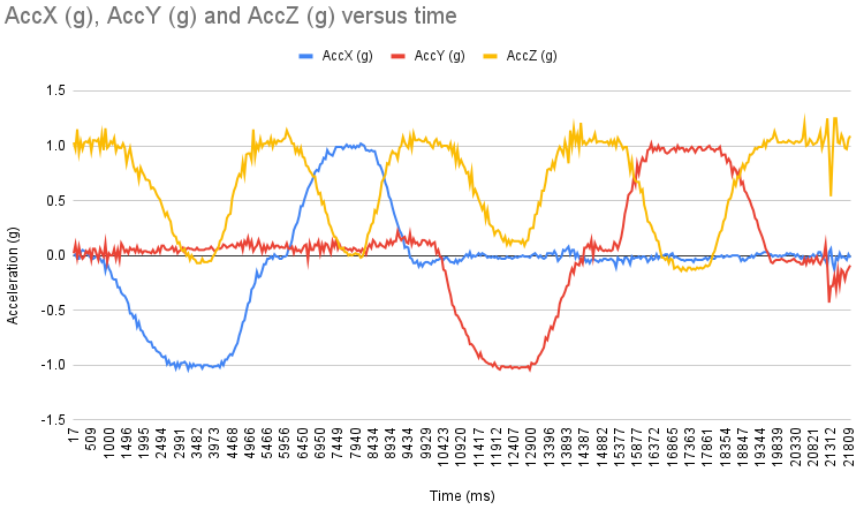


Figure 3.3.6: Rotation Validation Results

Both rotations (around x and y axis) are combined in a single experiment and plotted in figure 3.3.6 above. It can be seen that angular acceleration results in oscillatory motion in the x, y and z components of acceleration.

– **V2.3 - Pendulum Movement:** the results from the pendulum movement validation experiment can be graphically plotted as shown by the graphs in figure 3.3.7 below:

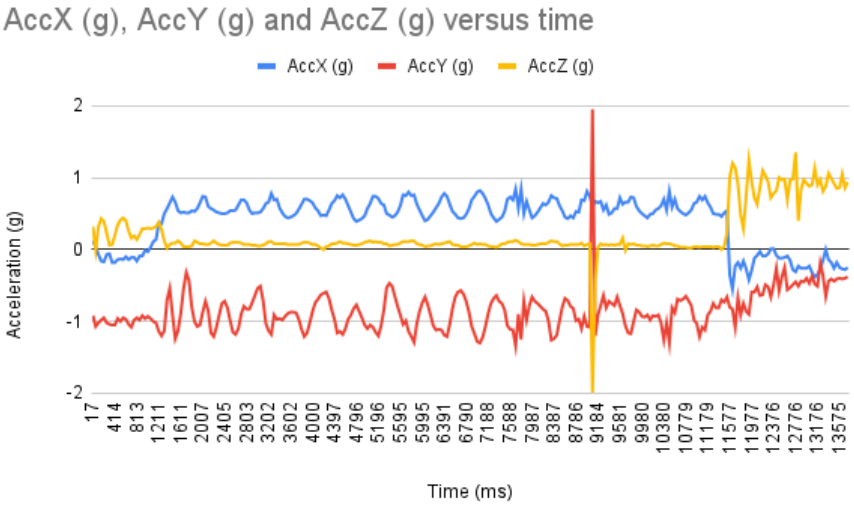


Figure 3.3.7: Pendulum Movement Results

4 Experiment Setup

In the practical implementation, the code needed to be written in C/C++. C was chosen as the preferred language for the project as it is more compatible with the STM32F051 microcontroller and STMCubeIDE. Although in this progress report, the algorithms were not run on the STM32F051 microcontroller itself, the project needs to keep its specifications and limitations in mind going forward.

The compression algorithm choice was unchanged from the simulated portion of the project and was implemented using a C versions of lz4 dictionary compression. The encryption method however was changed to a basic ASCII shift encryption algorithm for the following reasons:

1. The STM32F051 has extremely limited memory space and is hypothesised to be unable to handle two complex methods each with their own header and inclusion files. Therefore, there needed to be priority of the compression algorithm over the encryption algorithm.

2. Compression takes precedence over encryption in this project's context because the data transmission using iridium is extremely costly, compression of the data is vital to ensure cost reduction. The data is not sensitive in nature and therefore extensive encryption of the data is not necessary - the group has instead chosen to implement a simpler encryption algorithm and rely somewhat on the Iridium's network security.

3. The compressed data is already reasonably different enough from the original data due to the manipulations performed by a dictionary compression algorithm. Therefore a simple encryption method suffices to ensure minimal similarities to the original data.

The specifications and ATPs were changed accordingly and are reflected in the new table in table 6 below.

The compression and encryption experimental blocks below follow a similar structure to those in the simulated based experiment in progress report 1. The biggest difference however is due to the space constraints on the STM32F051 microcontroller, fewer and smaller dataset sizes were chosen and the arising results are less exhaustive than before.

The compression algorithm's effectiveness is hypothesised to be hindered as the initial dataset size may be too small to be effectively compressed. Thus, it is hypothesises that the compression ratio will be significantly lower than previously but the hope is that the dataset can still be compressed to meet the project's specifications.

Another difference from progress report 1 is that the decryption and decompression algorithms will not be tested or timed as they do not directly relate to the project and will occur post-data transmission on the receiving PC.

The data retired from the IMU for the following experiment is the accelerometer data taken whilst the IMU is stationary. It will therefore closely resemble the data taken from the stationary position accelerometer validation test above (plotted in figure 3.3.2 above).

4.1 Overall System

The aim of the overall system experiment block is to assess the overall functionality of the system.

1. Connect the system fully including: IMU, STM32F051 micrcontroller and PC
2. Configure communication protocols: USB to UART between the STM32F051 micrcontroller computer and I2C between the STM32F051 micrcontroller and the IMU.
3. Take in data for a specified amount of time and send to the PC.
4. Analyse this data in the frequency domain by taking the Fourier Transform.
5. Pass the data through the compression and encryption blocks and obtain the data ready for transmission.
6. Transmit this data to the PC
7. Decrypt and Decompress the data on the PC and ensure no loss has occurred.
8. Analyse this data in the frequency domain by taking the Fourier Transform to ensure that the first 25% of Fourier co-efficients have been retained.

4.2 Compression Block

The aim of the compression experiment will be to determine in the compression algorithm meets the specifications and corresponding ATPs. The following method will be followed to test the compression algorithm:

1. The compression algorithm will run on a dataset of known size and a time measurement will be taken for the compression part of the program's runtime.
2. This will be repeated on 5 datasets of varying sizes. The varying sizes of data input are determined by the number of lines inputted into the program
3. A graph will be plotted to determine the time complexity of the compression algorithm. This graph will plot dataset size vs. runtime.
4. The compression ratio will be calculated for each of the 5 compression cases and this will also be plotted.
5. The compressed file will then be decompressed.
6. The decompressed data will then be compared to the original data using a comparison algorithm to test for signs of data loss.
7. The Fourier transform of the original data and the decompressed data will be verified to have the same first 25% fourier coefficients.

The compression block expects to receive raw data and produce compressed data of reduced size.

4.3 Encryption Block

The aim of the encryption experiment will be to determine in the encryption algorithm meets the specifications and corresponding ATPs. The following method will be followed to test the encryption algorithm:

1. The encryption algorithm will run on a compressed dataset of known size and a time measurement will be taken for the program's runtime.
2. This will be repeated on 5 datasets of varying sizes.
3. A graph will be plotted to determine the time complexity of the encryption algorithm. This graph will plot dataset size vs. runtime.
4. The encrypted file will then be decrypted.
5. The encrypted file will then be decrypted and compared to the original data using a comparison algorithm to test for signs of data loss or tampering.

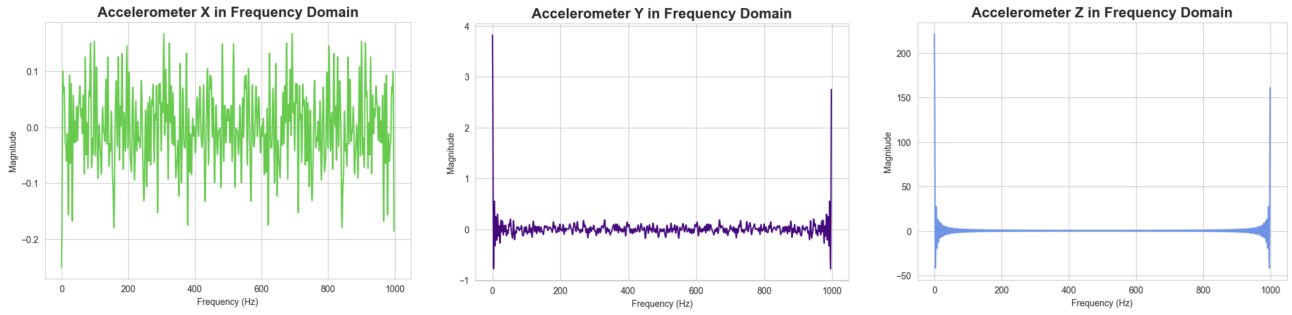
The encryption block expects to receive compressed data and produce encrypted data of increased randomization.

5 Results

The results of the above mentioned experimental setups are shown below.

5.1 Overall System

The only results that can be drawn from the overall system functionality is to analyse the pre-compressed/encrypted data against the post-decompressed/decrypted data in the frequency domain and compare the Fourier co-efficients to ensure that the first 25% have been retained. Because the algorithms used are lossless, the data is exactly the same and therefore its frequency domain data will be the same thus the Fourier co-efficients are retained. The frequency domain plots of the accelerometer data are shown in figure 5.1.1 below:



(a) x component of acceleration (b) y component of acceleration (c) z component of acceleration

Figure 5.1.1: Accelerometer Frequency Domain

5.2 Compression Block

Data pertaining to the compression block of the experiment is tabulated in table 3 below and plotted graphically below.

Dataset Size (bits)	Time (s)	Ratio
700	0.000079	2.31
1400	0.000118	2.54
2800	0.00014	2.08
4000	0.000168	2.11
5900	0.000169	2.29

Table 3: Compression Block Results

Figure 5.2.1 below shows how the runtime of the compression algorithm changes over varying dataset sizes.

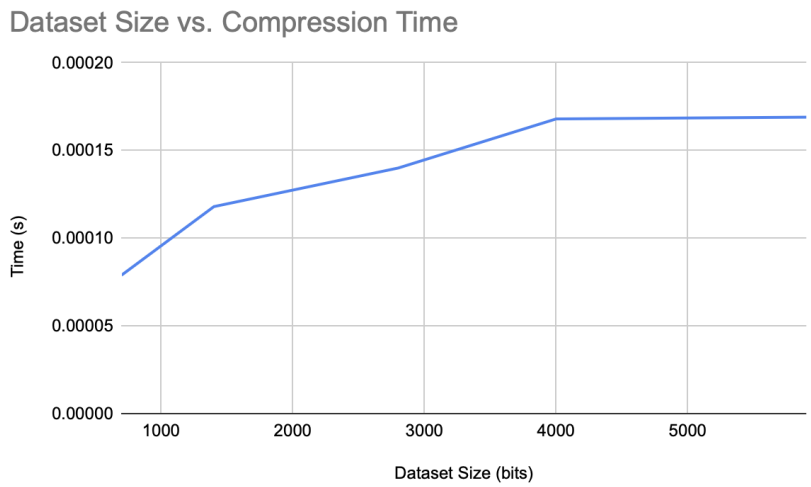


Figure 5.2.1: Graph showing the time required to compress varying sizes of input data.

It can concluded from figure 5.2.1 above that the lz4 compression algorithm implemented in this particular setup has a time complexity of order n [O(n)]. This is to be expected from the lz4 compression library.

Figure 5.2.2 below shows the change in compression ratio over varying dataset sizes.

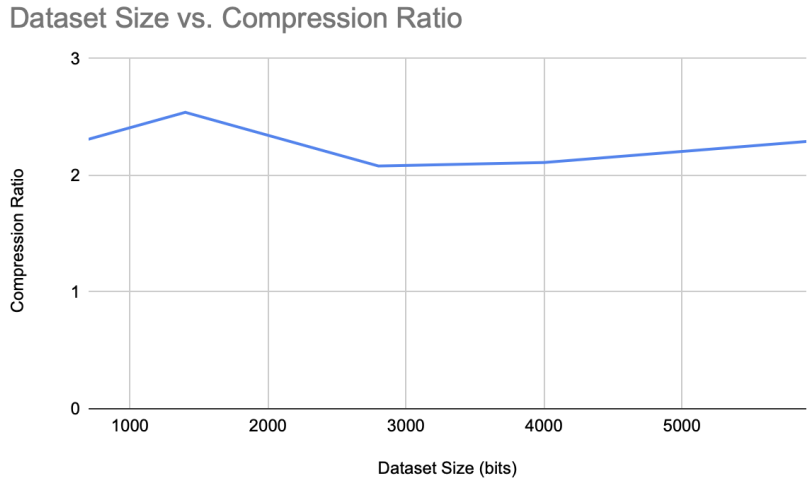


Figure 5.2.2: Graph showing the compression ratio over varying sizes of input data.

It can be seen that the compression ratio is around 2,2. This is less than in the simulated experiments but can be expected because of the smaller dataset sizes and thus reduced efficiency of the compression algorithm as explained above. However, the compression ratio is still adequate and within the project's specifications.

5.3 Encryption Block

Data pertaining to the encryption block of the experiment is tabulated in table 4 below and plotted graphically below.

Dataset Size (bits)	Time (s)
700	0.00006
1400	0.000064
2800	0.000107
4000	0.000128
5900	0.000153

Table 4: Encryption Block Results

Figure 5.3.1 below shows how the runtime of the encryption algorithm changes over varying dataset sizes.

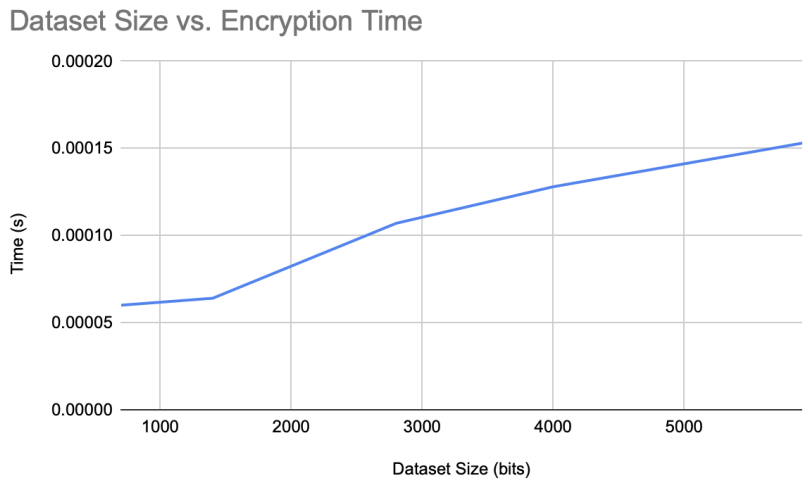


Figure 5.3.1: Graph showing the time required to encrypt varying sizes of input data.

It can concluded from figure 5.3.1 above that the ASCII shift encryption algorithm implemented in this particular setup has a time complexity of order n [O(n)].

6 Acceptance Test Protocols

The following table 5 summarises the project's requirements, specifications and acceptance test protocols (ATPs). It also highlights which correspond to which.

Requirements	**Specifications**	**Acceptance Test Protocols**
R1	S1	A1
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.	The design should adhere to the electrical specifications of the ICM-20649 IMU.	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).
R2	S2	A2
The project should be designed to utilize the STM32F051 microcontroller.	Both algorithms needs to be coded in C/C++ in STM32CUBEIDE to be compatible with the STM32F051 microcontroller.	Run the algorithms in a C/C++ IDE as well as on the STM32F051 to test that it works.
R3	S3.1	A3.1
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.	The compression and decompression of the IMU data will be done using a dictionary compression method.	Check that the file size of the compressed data is considerably less than the file size of the original data.
	S3.2	A3.2
	The compression ratio should be between 40% and 60%	Calculate the compression ratio between the uncompressed and compressed data.

Requirements	**Specifications**	**Acceptance Test Protocols**
R4	S4.1	A4.1
The data obtained from the IMU sensor should be encrypted to increase security of the data.	The encryption and decryption of the IMU data will be done using an AES encryption method.	Compare the encrypted data to the un-encrypted data to ensure minimal similarities which indicates reasonable level of encryption.
	S4.2	A4.2
	The encryption key should be either 128, 192 or 256 bits to ensure adequate security.	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.
R5	S5	A5
Minimal loss/ damage should apply to the decrypted and decompressed data.	The decrypted and decompressed data should reflect 25% of the Fourier coefficients of the original IMU data.	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.
R6	S6	A6
Limit power consumption by reducing the amount of processing done in the processor and minimizing the computation required.	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.	Test the time complexity of the compression and encryption programs by running multiple tests with different dataset sizes and plot the correlation.

Table 5: Requirements, Specifications and ATPs Old Version

However changes were made to the encryption method choice and thus the corresponding encryption requirement, specification and ATPs. The updates versions are tabulated in table 6 below:

Requirements	**Specifications**	**Acceptance Test Protocols**
R1	S1	A1
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.	The design should adhere to the electrical specifications of the ICM-20649 IMU.	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).
R2	S2	A2
The project should be designed to utilize the STM32F051 microcontroller.	Both algorithms needs to be coded in C/C++ in STM32CUBEIDE to be compatible with the STM32F051 microcontroller.	Run the algorithms in a C/C++ IDE as well as on the STM32F051 to test that it works.
R3	S3.1	A3.1
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.	The compression and decompression of the IMU data will be done using a dictionary compression method.	Check that the file size of the compressed data is considerably less than the file size of the original data.
	S3.2	A3.2
	The compression ratio should be between 40% and 60%	Calculate the compression ratio between the uncompressed and compressed data.

Requirements	**Specifications**	**Acceptance Test Protocols**
R4	S4	A4
The data obtained from the IMU sensor should be encrypted to increase security of the data.	The encryption and decryption of the IMU data will be done using an ASCII shifting encryption method.	Compare the encrypted data to the un-encrypted data to ensure minimal similarities which indicates reasonable level of encryption.
R5	S5	A5
Minimal loss/ damage should apply to the decrypted and decompressed data.	The decrypted and decompressed data should reflect 25% of the Fourier coefficients of the original IMU data.	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.
R6	S6	A6
Limit power consumption by reducing the amount of processing done in the processor and minimizing the computation required.	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.	Test the time complexity of the compression and encryption programs by running multiple tests with different dataset sizes and plot the correlation.

Table 6: Requirements, Specifications and ATPs Updated

From the results in section 5 above, the ATPs can be tested to determine whether they have been met or not. This is tabulated in table 7 below:

ATP Number	Has the ATP been met?	Reason
A1	✓	The data obtained from the IMU closely follows the sample data provided and is validated as 'correct' IMU data in the validation tests of the IMU module above.
A2	✓	The algorithms were coded in C/C++ to be compatible with the STM32F051 micrcontroller
A3.1	✓	The compression algorithm's efficiency although hindered by the smaller dataset sizes still results in adequate reduction of size of the data.
A3.2	✓	The compression ratio was calculated to be around 2,2. This is adequate as the project required a compression ratio between 40\% and 60\%
A4	✓	The similarity between the pre-encrypted data and post-encypted data is minimal. This enough to consider the encryption to be of adequate randomisation.
A5	✓	The lower 25% of the fourier co-efficients have been retained when comparing the fourier transform of the original data to the decrypted and decompressed data
A6	✓	The time complexity of the algorithms is O(n)

Table 7: ATP Meeting and Reasoning

The ATPs and corresponding specification were not changed - however one was omitted due to the encryption algorithm change. All ATPs (A1 - A6) were tested and met in this implementation of the project.