EEE3097S - Final Report

Ethan Meknassi[†] and Ethan Morris[‡]
EEE3097S

Class of 2022
University of Cape Town
South Africa
[†]MKNETH002 [‡]MRRETH003

Contents

I Admin Documents 1

II Requirement Analysis 4
- II-A General Requirements . . . 4
  - II-A1 Requirement R1 . . . 4
  - II-A2 Requirement R2 . . . 4
  - II-A3 Requirement R3 . . . 4
  - II-A4 Requirement R4 . . . 4
  - II-A5 Requirement R5 . . . 5
  - II-A6 Requirement R6 . . . 5
- II-B Overall System specifications and associated ATPs . . . 5
  - II-B1 Overall Specifications: . . . 5
  - II-B2 Overall system ATPs: . . . 5
- II-C Compression Subsystem specifications and associated ATPs . . . 6
  - II-C1 Compression subsystem Specifications: . . . 6
  - II-C2 Compression subsystem ATPs: . . . 6
- II-D Encryption Subsystem specifications and associated ATPs . . . 6
  - II-D1 Encryption subsystem Specifications: . . . 6
  - II-D2 Encryption subsystem ATPs: . . . 7

III Paper Design 8
- III-A Compression algorithms comparison . . . 8
- III-B Encryption algorithms comparison . . . 8
- III-C Feasibility Analysis . . . 9
  - III-C1 Economic . . . 9
  - III-C2 Technical . . . 9
  - III-C3 Operational . . . 9
  - III-C4 Schedule . . . 9
- III-D Possible Bottlenecks . . . 10
- III-E Inter-Subsystem and Inter-Sub-Subsystem Interactions . . . 10
- III-F UML Diagram . . . 10

IV Validation using Simulated or Old Data 12
- IV-A Data being used and why . . . 12
- IV-B Data used relating to the ATPs . . . 12
- IV-C Initial Analysis of the data using Jupiter Notebook . . . 12
- IV-D Experimental Setup for Simulated Data . . . 15
- IV-E Experiments for the Compression subsystem . . . 15
- IV-F Experiments for the Encryption subsystem . . . 16
- IV-G Results for Simulated Data . . . 16
- IV-H Overall functionality Results . . . 16
- IV-I Results for the Compression subsystem . . . 18
- IV-J Results for the Encryption subsystem . . . 20

V Validation using the IMU 23
- V-A Comparison between the IMU sensors . . . 23
- V-B Salient Features . . . 23
- V-C Steps to ensure working functionality of IMU on buoy . . . 23
- V-D Validation tests to ensure working functionality of IMU . . . 24
- V-E Results of the Validation tests . . . 25
- V-F Experimental Method for Final Results . . . 27
  - V-F1 Experiments for the Overall Functionality of the System . . . 28
  - V-F2 Experiments for the Compression subsystem . . . 28
  - V-F3 Experiments for the Encryption subsystem . . . 29
- V-G Final Results and Analysis . . . 29
  - V-G1 Initial analysis of the data in the files . . . 30

V-G2 Overall functionality Results . . . 32
V-G3 Results for the Compression subsystem . . . 35
V-G4 Results for the Encryption subsystem . . . 37

VI Consolidation of ATPs and Future Plan 41
VI-A Original ATPs from the requirement analysis: . . . 41
VI-A1 Previous Overall Functionality ATPs . . . 41
VI-A2 Previous Compression subsystem ATPs . . . 41
VI-A3 Previous Encryption subsystem ATPs . . . 41
VI-B New and revised ATPs: . . . 42
VI-B1 Revised Overall Functionality ATPs . . . 42
VI-B2 Revised Compression subsystem ATPs . . . 42
VI-B3 Revised Encryption subsystem ATPs . . . 42
VI-C Future plan: . . . 43

VII Conclusion 44

References 45

VIII Appendices 46
VIII-A Links to the IMU sensors datasheets: . . . 46
VIII-B Appendix A: Simulated Data (Initial Analysis) . . . 46
VIII-C Appendix B: IMU Data (Initial analysis) . . . 55
VIII-D Appendix B: (results of the experimentation for ”delay200.csv” and ”delay1000.csv”) . . . 59

I. Admin Documents

The contribution of each student is displayed in Table I.

SECTIONS	PERSON RESPONSIBLE
Admin Documents:	
Trello Board	MRRETH003
Gantt Chart	MKNETH002
Requirement Analysis:	
General Requirements	MKNETH002 & MRRETH003
Overall System Specifications and Associated ATPs	MKNETH002 & MRRETH003
Compression Subsystem Specifications and Associated ATPs	MKNETH002
Encryption Subsystem Specifications and Associated ATPs	MRRETH003
Paper Design:	
Compression algorithms comparison	MKNETH002
Encryption algorithms comparison	MRRETH003
Feasibility Analysis	MKNETH002 & MRRETH003
Possible Bottlenecks	MKNETH002 & MRRETH003
Inter-Subsystem and Inter-Sub-Subsystem Interactions	MKNETH002 & MRRETH003
UML Diagram	MRRETH003
Validation using the IMU:	
Comparison between the IMU sensors	MKNETH002 & MRRETH003
Salient Features	MKNETH002 & MRRETH003
Steps to ensure working functionality of IMU on buoy	MKNETH002 & MRRETH003
Validation tests to ensure working functionality of IMU	MKNETH002 & MRRETH003
Results of the Validation tests	MKNETH002 & MRRETH003
Experimental Method for Final Results	MKNETH002 & MRRETH003
Final Results and Analysis	MKNETH002 & MRRETH003
Consolidation of ATPs and Future Plan:	
Original ATPs from the requirement analysis	MKNETH002 & MRRETH003
New and revised ATPs	MKNETH002 & MRRETH003
Future Plan	MKNETH002
Conclusion	MKNETH002 & MRRETH003
Appendices:	
Appendix A	MRRETH003
Appendix B	MKNETH002

TABLE I: Table showing the contribution of each student for each section.

Ethan Meknassi, MKNETH002, was in charge of the compression subsystem. This included all the experimental methods, testing, result, implementation and ATPs of the compression subsystem. Ethan Morris, MRRETH003, was in charge of the encryption subsystem. This included all the experimental methods, testing, result, implementation and ATPs of the encryption subsystem. The overall system parts as well as all the researches and writing were done together.

Git repository link: https://gitlab.com/mrreth003/data-transfer-microcontroller

The progress of the project has been monitored using Trello wherein all the tasks relating to the Milestones are marked in order of their progress. This website can be accessed on https://trello.com/b/8IhJEUIB/project-board.

The updated Project Task Management board is displayed in Figure 1.

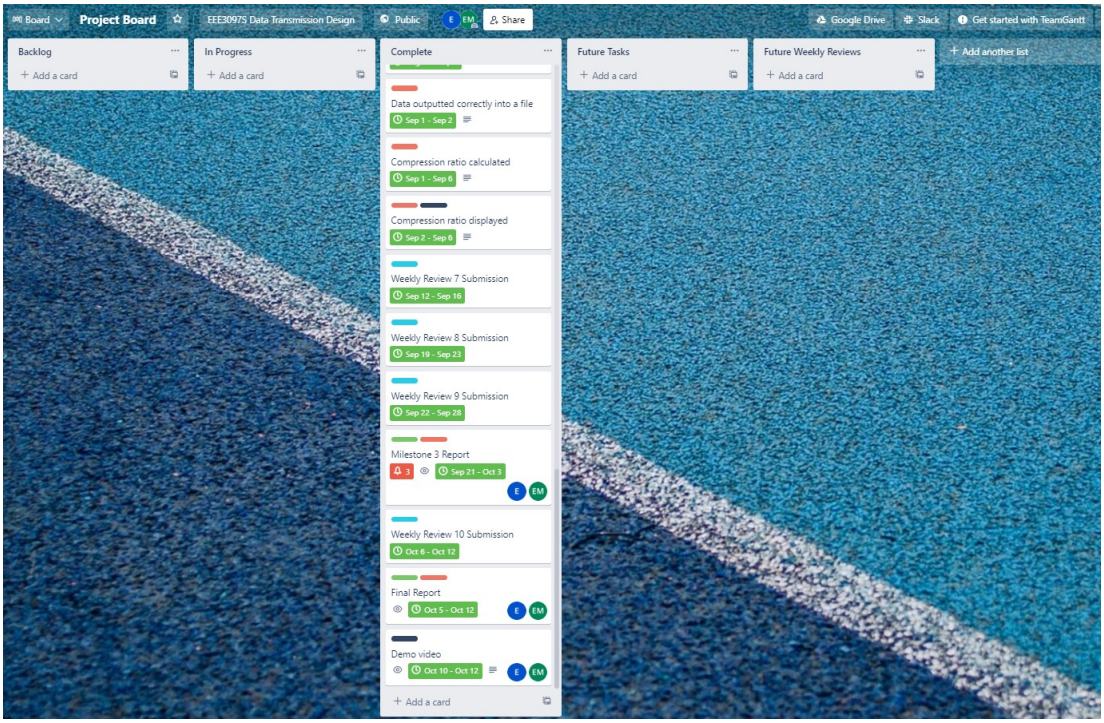

Fig. 1: Project Management Board using Trello

The timeline for this project is displayed using a Gantt Chart in Figure 2.

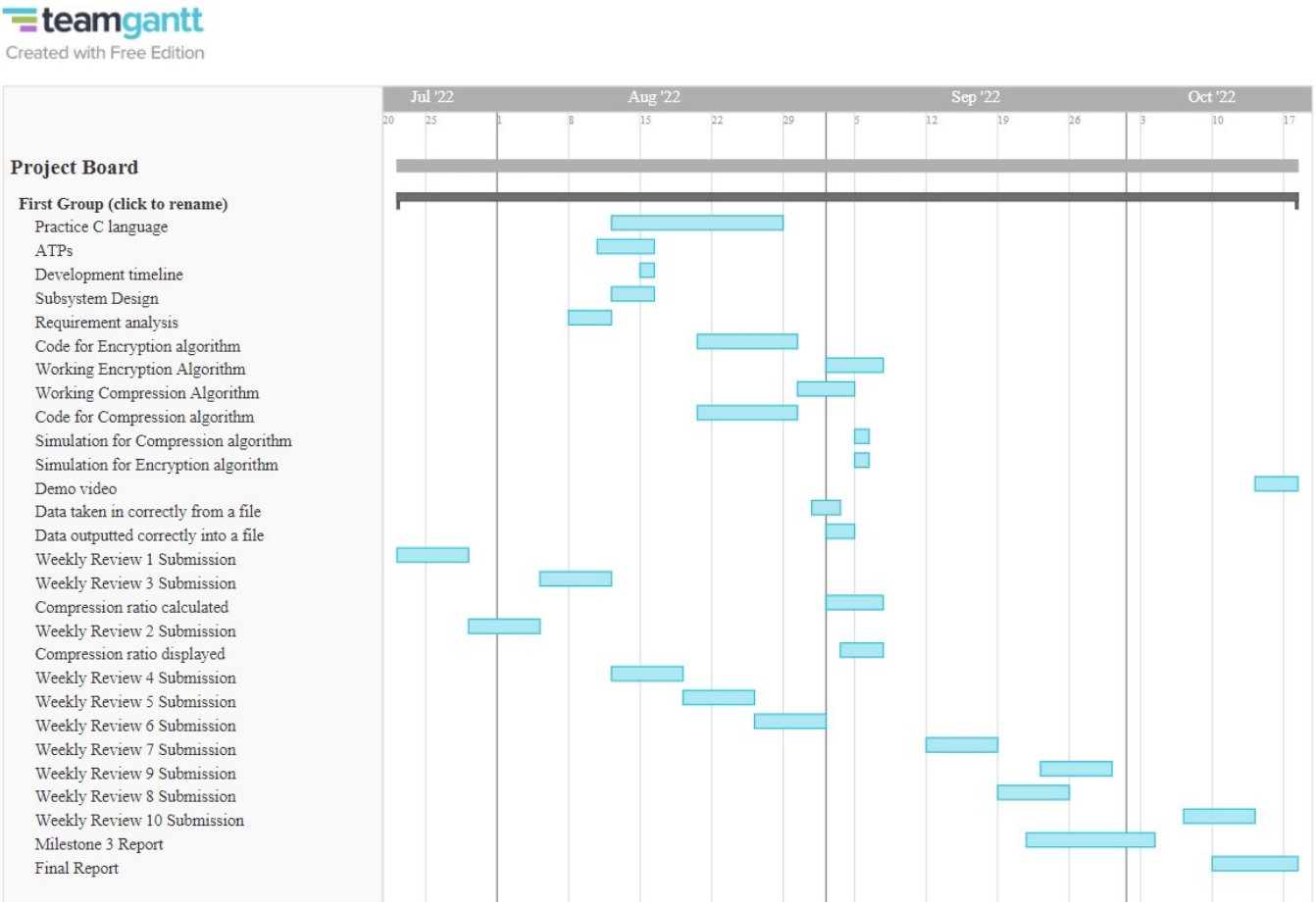

Fig. 2: Gantt chart displaying the timeline for the project

In terms of the progress of the project, the final report and demonstration video was completed in time. All the progress was monitored using a gantt chart and a project management tool (trello) which allowed the team to meet the deadlines for each milestones.

II. Requirement Analysis

A. General Requirements

1) Requirement R1

The compression algorithm used must allow for at least 25% of the Fourier coefficients to be extracted [1].

The most important aspect of this is that only the lower 25% of the Fourier coefficients are required to be extracted and not just any 25% [2]. This requirement allows for the data being transmitted to be reduced and for the compression to be more efficient. It is necessary in order for the crucial parts of the data to not be removed as a sacrifice when trying to reduce the time spent on compression.

2) Requirement R2

The IMU sensor to be used is **ICM-20649**.

The following are features of the ICM-20649:

- Triple-axis MEMS gyroscope (X, Y and Z) with a user-programmable full-scale range of ±500 dps, ±1000 dps, ±2000 dps, and ±4000 dps and integrated 16-bit ADCs with a digital output.
- Triple-axis accelerometer (X, Y and Z) with a programmable full-scale range of ±4g, ±8g, ±16g, and ±30g and integrated 16-bit ADCs with a digital output.
- Wake-on-motion interrupt to allow for the applications processor to function in low power operation.
- User-programmable interrupts
- Digital Microprocessor (DMP) enables low power calibration of accelerometer, gyroscope and compass. DMP is optimised for SMD, Step Count, Step Activity, Step Detect, Activity Classification, Rotation Vector and Gaming Rotation Vector.
- I2C up to 100 kHz (standard) or 400 kHz (fast) and SPI up to 7 MHz.
- Auxiliary master I2C bus for reading data from external sensors.
- VDD operating range 1.71V - 3.6V.

The sensor used in the testing of the device is the Waveshare Sense HAT (B). The functionality that comes with this sensor is very similar to the ICM-20649 and everything required from the ICM is also featured on the Sense HAT (B).

3) Requirement R3

The compression algorithm must also reduce the size of the data enough so that it becomes quicker for the data transmission and reception. This will lead in an increasing amount of data being able to be transmitted from the IMU sensor.

4) Requirement R4

Minimising the computation required for the microcontroller is necessary to reduce the power consumption caused by the amount of processing done. The time taken from data transmission to data reception must be no longer than 60 seconds.

The reduction of the power consumption will allow for the buoy to operate for longer without its battery needing to be charged or replaced. The more frequent battery changes would allow for the buoy to capture larger sets of data as well as reducing the amount of user interference in the data captured. The time spent on the compression and encryption processes affect the power consumption of the device as well as the transmission and reception speed of the data. Therefore, the processes need to be kept to a minimal execution time to maximise the efficiency of the project.

5) Requirement R5

The compression and encryption algorithms must be ran together and must be completed quickly. The quicker the data is being transmitted the more data can be gathered in the specified period of time.

6) Requirement R6

The memory and CPU usage of the entire process must be low enough so that the system can work efficiently and give out the desired output.

From these general requirements, specifications for the overall system, the compression subsystem and the encryption subsystem were derived below.

B. Overall System specifications and associated ATPs

1) Overall Specifications:

For the overall system, multiple specifications are gathered from requirement R2.

Requirements	Specification associated to it
Requirement R2	S1.1: Connection between the IMU sensor and the STM microcontroller is successful and correctly initialized
Requirement R2	S2.1: The data must be fully recovered and no data must be added or removed. The values must still be the same after decompression
Requirement R4	S3.1: The execution time for the compression and decompression must be less than 20 seconds
Requirement R6	S4.1: The compression and decompression must each take less than 10% of the total RAM pf the entire system

TABLE II: Table showing the overall system specifications and which requirements they come from

From these specifications, the ATPs for the compression subsystem can be derived.

2) Overall system ATPs:

Specification	Figures of Merit	Experiment	Acceptable Performance Definition
S1.1	Connection between the IMU sensor and the STM microcontroller	Test O1 tests whether the file sent by the IMU sensor is provided to the STM correctly with no loss and in the correct format	Successful transmission between the IMU and the STM microcontroller.
S2.1	Correctness of the data	Test O2 measures the size of the compressed file and compares it to the size of the original file. The compression ratio is determined from this test.	None of the data must be lost or added during the compression and decompression The values must all be the same.
S3.1	Execution Time	Test O3 measures the time before and after the compression algorithm runs and determines the time taken for the process.	The execution time must be less than 60 seconds
S4.1	Memory usage	Test O4 identifies how much RAM is used for different data sizes.	The values must be all the same.

TABLE III: Table showing the overall system specifications and which requirements they come from

C. Compression Subsystem specifications and associated ATPs

1) Compression subsystem Specifications:

Requirements	Specification associated to it
Requirement R1	S1.2: The lower 25% of the Fourier coefficients of the original file must be present and available from the decompressed file
Requirement R2	S2.2: The data must be fully recovered and no data must be added or removed. The values must still be the same after decompression
Requirement R3	S3.2: The compression must be greater or equal to 5
Requirement R4	S4.2: The execution time for the compression and decompression must be less than 20 seconds
Requirement R6	S5.2: The compression and decompression must each take less than 2.5% of the total RAM

TABLE IV: Table showing the compression specifications and which requirements they come from

From these specifications, the ATPs for the compression subsystem can be derived.

2) Compression subsystem ATPs:

Specification	Figures of Merit	Experiment	Acceptable Performance Definition
S1.2	Percentage of the Fourier Coefficients of the data	Test C1 is executed to compare the data in the decompressed file with the original file and will identify any possible differences and how much differences there is.	The lower 25% of the Fourier coefficients of the original file must be present and extractable from the decompressed file.
S2.2	Correctness of the data	Test C1 is again used to compare the data in the decompressed file with the original fileand identify possible differences.	None of the data must be lost or added during the compression and decompression. The values must be all the same.
S3.2	Compression Ratio	Test C2 measures the size of the compressed file and compares it to the size of the original file. The compression ratio is determined from this test.	Compression ratio must be equal or grater than 5 .
S4.2	Execution Time	Test C3 measures the time before and after the compression algorithm runs and determines the time taken for the process.	The execution time can be measured using the execution speed which must be faster than 0.75Mb/s.
S5.2	Memory usage	Test C4 identifies how much RAM is used for different data sizes.	The entire Compression and decompression operation must not use more than 10% of the total RAM.

TABLE V: Acceptance Test specifications for the Compression subsystem

D. Encryption Subsystem specifications and associated ATPs

1) Encryption subsystem Specifications:

Requirements	Specification associated to it
Requirement R1	S1.3: The encrypted data must be fully secure and cannot be accessed without the key
Requirement R2	S2.3: The data must be fully recovered and no data must be added or removed. The values must still be the same after decompression
Requirement R4	S3.3: The execution time for the encryption and decryption must be less than 20 seconds
Requirement R6	S5.2: The encryption and decryption must each take less than 2.5% of the total RAM

TABLE VI: Table showing the compression specifications and which requirements they come from

From these specifications, the ATPs for the compression subsystem can be derived.

2) Encryption subsystem ATPs:

Specification	**Figures of Merit**	**Experiment**	**Acceptable Performance Definition**	**ATPs Met/Unmet**
S1.3	Security of the data	Test E1 tests whether if the encrypted data is accessible and can be cracked.	The encrypted data must be fully secure.	ATP Met
S2.3	Correctness of the data	Test E2 compares the decrypted file with the original file and identifies possible differences.	The decrypted data must be the exact same as the original file and no data must be added/lost or corrupted.	ATP Met
S3.3	Execution Time	Test E3 measures the time before and after the encryption algorithm run and determines the execution time.	The execution time must be less than 10 seconds.	ATP Met
S3.4	Memory Usage	Test E4 identifies how much RAM is used for different data sizes	The entire Encryption and Decryption operation must not use more than 10% of the total RAM	ATP Met

TABLE VII: Acceptance Test specifications for the Encryption subsystem

III. Paper Design

A. Compression algorithms comparison

TABLE VIII: Table showing the different compression algorithms found with their description/features.

Algorithm	Description	Key features
Run length Encoding Algorithm	Simplest data compression algorithm to use [3]. Uses redundancies to compress the data.	Very simple to use but not the most efficient.
LZ77	Lossless, asymmetric compression algorithm which uses the single window method. Dictionary based, it converts the ASCII characters in the data into binary numbers.	Fast and efficient with a good compression ratio and is used in addition to other tools in other compression algorithms.
Huffman Encoding	Lossless, it uses probability distribution of the alphabet of the source to create code words for certain symbols.	Efficient but better used in parallel with other compression algorithms [3].
DEFLATE	Variation of the LZ77 algorithm but also uses Huffman encoding.	Simple to implement, Good compression ratio and combines benefits of both LZ77 and Huffman coding.
Bzip2	Makes use of the Burrows-Wheeler block-sorting text compression algorithm as well as Huffman encoding.	Better compression ratio than DEFLATE however lags behind in using modern compression algorithm.
LZW	Dictionary based it also is an adaptive compression algorithm. [4]	Good compression ratio but not optimized size. [5]

When comparing the different compression algorithms it is crucial to understand the following terms:

Compression time: This is measured using the compression ratio (ratio between the size of the compressed file and the size of the source file) and gives information on how fast the compression and decompression is [6].

The algorithm that will be used for this design is the **DEFLATE** algorithm []. This is due to its easy implementation as well as a good compression ratio compared to other algorithms. Furthermore, it combines the benefits of other compression algorithm into a single one. The LZ77 and the Huffman encoding algorithm are featured in the DEFLATE algorithm. DEFLATE also provides a reduced size of compressed data. This compression algorithm has a fast compression and decompression time which will help meet the requirements of the design.

B. Encryption algorithms comparison

The comparison for the encryption algorithms are based on multiple crucial factors. The speed of encryption and decryption, the key size, the rate of key generation, security, efficiency on implementation, complexity of the algorithm and the number of users accommodated by the model are all very important to look at when choosing an encryption algorithm [8].

It is also very important to consider the idea of symmetric and asymmetric encryption. Symmetric encryption allows for the encryption and decryption of the data using one single encryption key. Whereas asymmetric encryption method requires multiple keys for the encryption and decryption. Symmetric encryption is much easier to implement while still allowing security of the data.

The two best encryption algorithms are Blowfish and AES-256 [3]. Due to its easier implementation and its larger block size, **AES-256** will be used for this design. However, Blowfish could be an alternative if issues are be faced using AES-256.

TABLE IX: Table showing the different encryption algorithms found with their features [7]

Encryption algorithm	Key length	Cypher type	Speed	Block size	Security
RSA	depends	Asymmetric	Extremely Slow	Variable	Not secure
DES	56 bits	Symmetric	Slow	64 bits	Low security
3DES	168 bits	Symmetric	Slow	64 Bits	Moderate security
Blowfish	Varies from 32 to 448 bits	Symmetric	Fast	64 bits	Excellent security
AES-256	128,192 or 256 bits	Symmetric	Fast	128 bits	Excellent security

AES-256 provides the adequate key length, as well as the best encryption time which is dictated by the block size and key size of the algorithm.

C. Feasibility Analysis

1) Economic

The economic feasibility of the project can be determined using a cost-benefit analysis.

The cost of the project is based entirely on the price of the two elements of the system - the microcontroller and the sensor. The STM32F051R8T6 costs R150 and the ICM-20649 costs R180 bringing the total cost per unit for the materials to R330. The amount to be spent on labour would depend on the number of hours spent on development and configuration. The number of hours spent on this project will be 80 hours including all research and report based work as well as the construction of the device.

The benefits of this project when related to alternatives in the same market are evident in the large saving that can be seen when using this device over other options. This project also brings awareness to the conditions in the environment, whereby the changes in the data can be modelled to measure the changes to the earth that come with Climate Change.

2) Technical

The technical feasibility of the project is based on the risks and technical limitations attached to it.

The algorithms chosen for the project are the DEFLATE compression algorithm and the AES-256 encryption algorithm. The good compression ratio of the DEFLATE algorithm makes it very fast and efficient for both the compression and decompression of the data. DEFLATE is also memory independent and uses a combination of two already efficient algorithms which allows for a reduced file size once the data has been compressed. Additionally, this algorithm is rather simple to implement which makes it more feasible. The AES-256 encryption algorithm uses a symmetric approach which makes use of a single cryptographic key for both the encryption and the decryption of the data. This makes the implementation of the encryption aspect of the project much simpler. Furthermore, AES-256 is very fast, secure and is easier to implement which makes it the most feasible for this project.

3) Operational

The operational feasibility of the project is assessed by looking at the efficiency of the project and system controls.

The system is designed to only produce at least 25% of the Fourier transforms of the data points in order to increase the efficiency by reducing the processing power required to process all the data with with full accuracy. The time spent on the processing is therefore also reduced allowing for better functioning of the whole system. The time spent in each section of the data processing needs to be completed within one clock cycle as the data is measured from the sensor every cycle. The reduction in the amount of data requiring processing allows for data to be processed more frequently and for the clock cycle period to be reduced. The compression and encryption algorithms used are two of the fastest available allowing for no unnecessary delay in the data transfer process due to data manipulation.

4) Schedule

The schedule based feasibility is made up of time based limitations wherein the time spent on each task within the project is assessed to determine if the time has been used effectively.

80 hours have been dedicated to the design and implementation of this project. This time has been split into three different milestones: The Paper Design, Simulation and Implementation. One limitation in terms of time is based on how quickly it takes to fully understand the task at hand and solve it. Once a solution has been determined, the design and implementation of the system is based on the coding proficiency in the language of choice as well as the time spent on the development of the functioning project includes the construction of the code structure and the communication between the sensor and the microcontroller.

D. Possible Bottlenecks

The use of the Waveshare Sense HAT (B) instead of the ICM-20649 might be a possible bottleneck during the implementation of the project. The hardware differences between the two could cause problems.

Another possible bottleneck could be that the Waveshare Sense HAT (B) might produce data too fast for the next subsystem to handle. This would lead to the loss of crucial data during the transmission.

The lack of resources about the project could be a possible bottleneck further down the line of the project.

Additionally, a possible bottleneck that could be faced is the use of the STM32F051R8T6 microcontroller rather than a Raspberry Pi. This comes with several issues, the code for the algorithm will need to be written in the C language rather than an easier language like python. The engineering team will therefore need to learn and practice the C language for the implementation to be successful. Furthermore, the implementation of the code onto the microcontroller is done through a complicated IDE (STM32Cube) that the team is not familiar with.

E. Inter-Subsystem and Inter-Sub-Subsystem Interactions

In the **Data Processing Subsystem**, the data is generated by the IMU sensor, this data is retrieved through the **Data Generation Sub-Subsystem** and then passed onto the **Compression Sub-Subsystem** inside the **Data Compression Subsystem**. The **Compression Sub-Subsystem** is made up of processing the input data from the IMU sensor which is formatted as a csv data file. The compression of this data uses the DEFLATE algorithm, creating a new text file of the compressed data ready to be sent to the encryption algorithm in the **Data Encryption Subsystem**. The **Encryption Sub-Subsystem** is made up of processing the compressed data from the text file stored in the microcontroller storage. The encryption uses the AES-256 algorithm - which makes use of one primary key for encryption and decryption. The data goes through the **Encryption Sub-Subsystem** and is then sent back to the **Data Reception Sub-Subsystem** (part of the Data Processing Subsystem). In this Sub-Subsystem, the data is sent from the microcontroller to the system controller. The data is then transferred back to **Decompression Sub-Subsystem** and the **Decryption Sub-Subsystem** where it will be translated back to readable content. Finally the lowest 25% Fourier coefficients of the data is capture and validated in the **Data validation Sub-Subsystem**.

F. UML Diagram

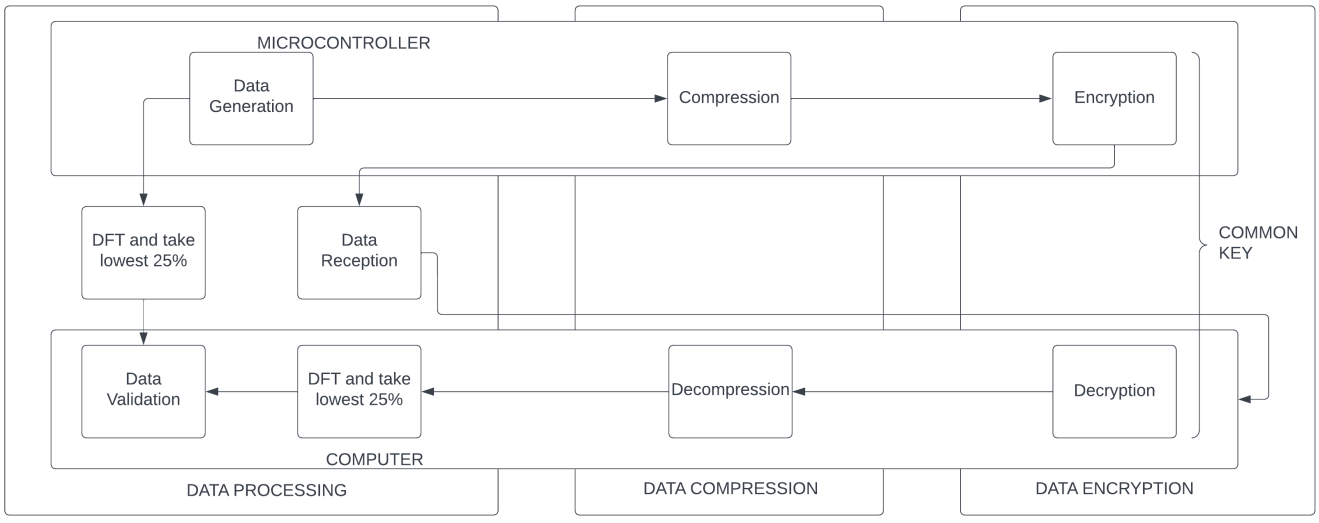

Fig. 3: UML Diagram showing the structure of the Subsystems

The UML Diagram displaying the functioning of the the system can be seen above in Figure 3. This diagram starts with the input data captured in the Data Generation sub-subsystem. This data is sent in two directions, one copy is sent through a DFT function where the lowest 25% of the Fourier transforms are chosen and then through to the Data Validation Sub-Subsystem while the

other copy of the data is taken in by the Data Compression subsystem, more specifically the Compression sub-subsystem. This compressed data is sent to to the Encryption Sub-Subsystem - of the Data Encryption Subsystem - where it is encrypted using the public key. The encrypted data is sent to the Data Reception Sub-Subsystem where it is sent from the microcontroller storage to the laptop upon which it goes through the Decryption Sub-Subsystem and the Decompression Sub-Subsystem. The decompressed data is sent processed through a DFT where the lowest 25% of the Fourier transform values are chosen to be sent to the Data Validation Sub-Subsystem. The two copies of the data in the Data Validation Sub-Subsystem are compared and any differences are noted. The compression and decompression ratios are also calculated and recorded. The final output data can be accessed from the Data Validation Sub-Subsystem.

IV. Validation using Simulated or Old Data

A. Data being used and why

The data that was used was the data gathered from example ".csv" files. These files are the "EEE3097S 2022 Turntable Example Data.csv", "EEE3097S 2022 Turntable Example Data 2.csv" and "EEE3097S 2022 Walking Around Example Data.csv". The files contain various values and data regarding some measurements including accelerate positions, gyroscope, time, etc. This data has the same format and is very similar to the data that will be sent from the IMU sensor to the microcontroller. The example files contain different sets of values and have various sizes. This allows for the testing of the system to be as close as possible to what the actual data will look like. This provides a clear indication of the behavior of the system and if there are any anomalies. It is also important to note the size of these files as it will play an important role in the experiment.

"EEE3097S 2022 Turntable Example Data.csv" = 21Mb

"EEE3097S 2022 Turntable Example Data 2.csv" = 19Mb

"EEE3097S 2022 Walking Around Example Data.csv" = 14Mb

The "EEE3097S 2022 Turntable Example Data.csv" is biggest file out of all those being tested and therefore should be the one taking more time and using more memory storage.

B. Data used relating to the ATPs

Using these example files, the correctness of the final outputs must be determined. This will be achieved by using the data provided in the example files and comparing the original files. Therefore, using this data will be adequate for this ATP. This includes the ATP regarding the correctness of the data for the overall functionality of the system as well as the compression subsystem and encryption subsystem. This aspect also fulfills the ATP based on the specification that at least 25% of the Fourier Coefficients of the data must be retrieved. However, an additional comparing tool is needed to make this comparison possible. A python script will be used to determine differences between the input and output files of the whole system.

For the compression ratio ATP, the example data used will be adequate as the file size was provided. This file size will therefore be compared to the output of the compression algorithm and the compression ratio will be determined.

Regarding the ATP based on the execution time specification, the data used is good enough to run this ATP as it will provide an indication of how much time it takes to run the process for various sizes of data. This is why using all three example data given will be a good indication of the running time of the system. The ATPs regarding the overall system, the compression subsystem and also the encryption will therefore be correctly tested using the three example data files.

The example files will also need to provide adequate testing for the security of the data for the encryption subsystem. The encrypted data will be able to be analyzed from inputting the three different example files in the encryption algorithm and analyze the encrypted data.

Finally, the three example files need to be used for testing the ATP regarding the memory usage of the system. The three files have different sizes which will have a different effect on the memory usage. Using this data for the testing of the RAM will be good enough as they will provide a range of RAM usage that depends on the size of the file. The data, being similar to the data that the sensor will provide, will therefore give an indication of how much RAM is being used by the system. This ATP applies for the overall system, the compression subsystem and the encryption subsystem. The data used is therefore adequate for this ATP.

C. Initial Analysis of the data using Jupiter Notebook

The data that was in the test files were all analysed prior to the experimentation in order to have an idea of what to expect. The FFTs for the Accelerometer, the gyroscope as well as the temperature values were plotted using python.

Initial Analysis of the data contained in the ”EEE3097S 2022 Turntable Example Data.csv” file or ”1.csv”:

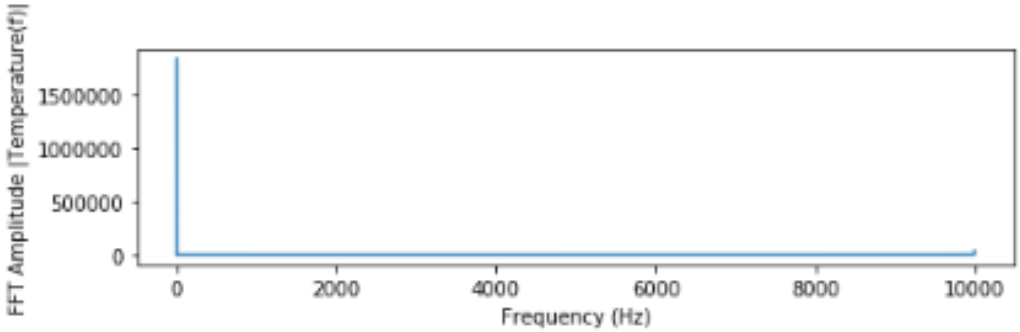

Fig. 4: Graph showing the FFTs of the accelerometer values of the IMU sensor straight from the file ”1.csv” retrieved from the STM

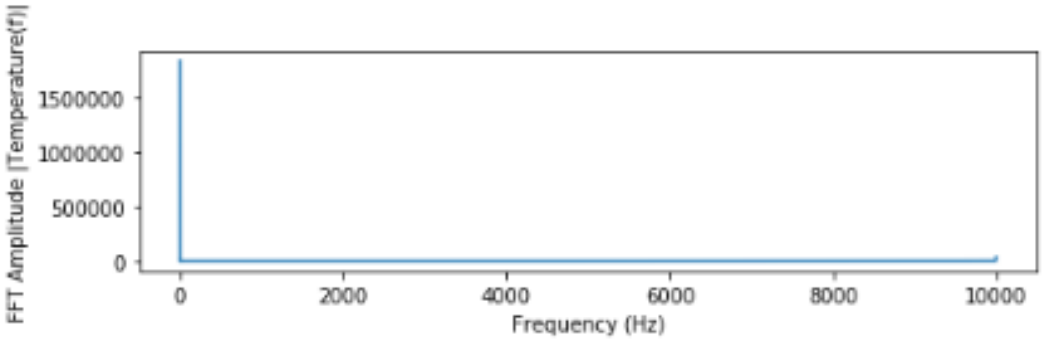

Fig. 5: Graph showing the FFTs of the accelerometer values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The accelerometer values have not been changed by the processing system, meaning the transmission was successful.

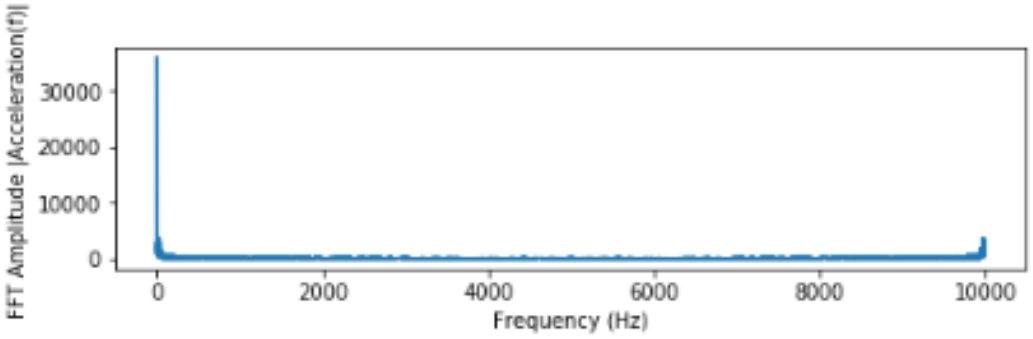

Fig. 6: Graph showing the FFTs of the gyroscope values of the IMU sensor straight from the file ”1.csv” retrieved from the STM

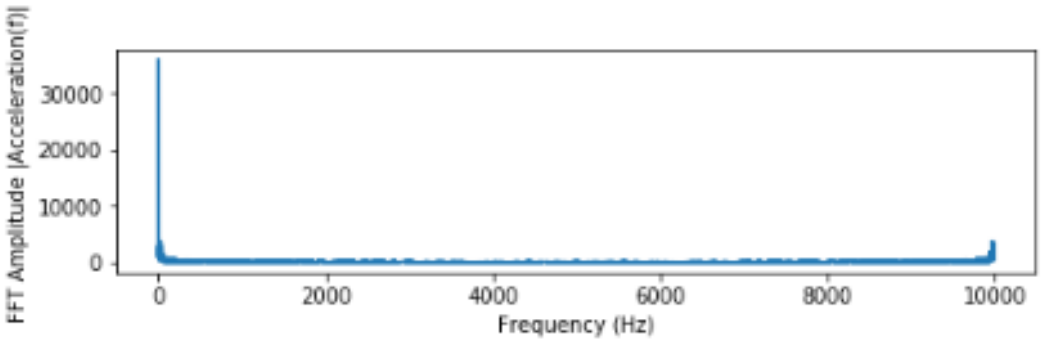

Fig. 7: Graph showing the FFTs of the gyroscope values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The gyroscope data values have not changed.

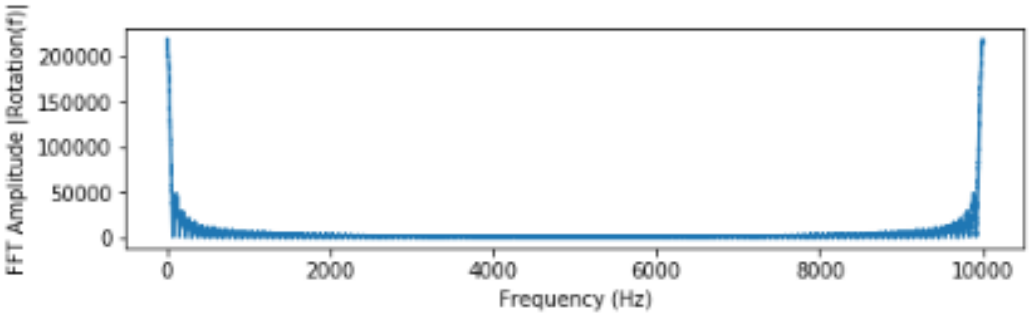

Fig. 8: Graph showing the FFTs of the temperature values of the IMU sensor straight from the file ”1.csv” retrieved from the STM

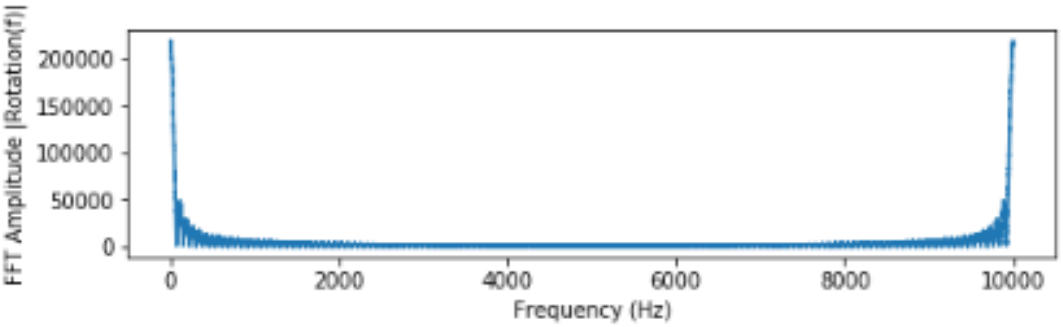

Fig. 9: Graph showing the FFTs of the temperature values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The temperature data values have not changed.

D. Experimental Setup for Simulated Data

The compression algorithm in use when testing this data has been changed from ”DEFLATE” to ”bzip2”. This change is in effect for all results shown for the rest of the report.

Test O1: This test was conducted to identify and analyse the ability of the IMU sensor to send data to the STM microcontroller in the adequate format. The format that must be delivered to the STM microcontroller is a ”.csv” file as it is what the compression algorithm takes in as an input. Therefore, the expected input for this test was a ”.csv” file from the IMU sensor. To test this, data from the sensor was sent over to the STM microcontroller using example data. The format of the file received at the STM was then identified and whether the data is the same (no loss or corruption) was determined. The output was expected to be a ”.csv” file.

Test O2: This test was effectuated to determine the accuracy and correctness of the data throughout the overall process. The original file that is passed onto the compression subsystem is compared to the final output of the entire process. This comparison was done using a Python script named ”CompareFiles.py” which is available on the git repository of the project. The Python script compares the two files and checks if there are any differences or missing parts. It then outputs a similarity percentage, with 100% meaning that the data in the files are identical. The expected output of this experiment is that the original data and the final output data are the same with no data lost/corrupted.

Test O3: This test was done in order to determine the time taken for this entire system to run. The example file was inputted into the all the algorithms which run one after the other using the one command and the file goes through each systems. The time taken for each step is recorded and outputted. After the final output file was received, the different times for each steps are then combined to obtain the execution time of the entire process. This process was then repeated for each of the example files in order to obtain accurate results. It was then compared to the specifications that were fixed to the system prior the experiment. The expected output must be an execution time that is less than the specifications (less than 60 seconds).

Test O4: This test was completed in order to determine the memory usage of the overall functionality of the system. This was determined based on how much RAM was used during the execution of the system. This experiment was done on a laptop with 8GB of RAM. The percentage RAM used by the system was determined by recording how much memory was used by the entire system. The STM microcontroller has a static RAM of 192KB, so the experiment will take this in consideration as well.

E. Experiments for the Compression subsystem

Test C1: This test was effectuated to determine the accuracy and correctness of the data throughout the compression subsystem. The original file, which is the example ”.csv” file was inputted into the compression and then the decompression algorithms. The final output after the decompression, a ”.txt” file, is then compared back to the original file. This comparison was done using the comparison Python script, ”CompareFiles.py”. The Python script compares the two files and checks if there are any differences or missing parts. It then outputs a similarity percentage, with 100% meaning that the data in the files are identical. This is then compared to the specifications which required the lowest 25% of the Fourier coefficients.

Test C2: This test aims to determine the compression ratio. This ratio was calculated using the original file size and dividing it by the compressed file size.

$$compression\ ratio = \frac{original\ file\ size}{compressed\ file\ size}$$

This test was conducted onto all 3 example files to determine the compression ratio for various sizes of data. The input for this block is a ”.csv” file and the output is a ”.csv.bz2” file. This compression ratio was then compared to the compression ratio for each example file which was first set during the paper design specifications. The expected output was a compression ratio that respects the specifications (greater than 5).

Test C3: This test was done in order to determine the time taken for the compression and decompression to run. The example files were inputted (”.csv”) into the compression algorithm, which outputs a ”.csv.bz2” file, and then the decompression algorithm - which takes in a ”.txt.bz2” file and outputs a ”txt” file. The time taken for each step is recorded and outputted. After the final output file was received, the different times for each steps were then combined to obtain the execution time of the entire compression subsystem. This process was then repeated for the 3 different example files in order to obtain accurate

results. It was then compared to the specifications that were fixed to the system prior the experiment. The expected output of this test was an execution time of less than 20 seconds.

Test C4: This test was completed in order to determine how much memory usage the compression subsystem was taking. This was determined using how much RAM was used during the execution of the subsystem. The RAM used was recorded from the task manager of the computer. The range of memory used by the system was then put into a percentage with respect to the overall RAM and compared to the specifications of the system.

F. Experiments for the Encryption subsystem

Test E1: This test was done to evaluate what the encryption algorithm was doing to the data and whether the encrypted data could be read without the cryptographic key or not. The encrypted data was analyzed to check if it was secure. The example file was inputted through the encryption algorithm as ".csv.bz2" file. The encrypted data was the output of this test as a ".txt.bz2" file which was then analyzed to see if it was secure and properly encrypted.

Test E2: This test was effectuated to determine the accuracy and correctness of the data throughout the encryption subsystem. The original file, which is the example ".csv.bz2" file was inputted into the encryption and then the decryption algorithms. The final output after the decryption, in the form of a ".txt.bz2" file, is then compared back to the original file. This comparison was done using the comparison Python script, "CompareFiles.py". The Python script compares the two files and checks if there are any differences or missing parts. It then outputs a similarity percentage, with 100% meaning that the data in the files are identical.

Test E3: This test was done in order to determine the time taken for the encryption and decryption to run. The files that are inputted for this test are coming from the compression subsystem and therefore the input is a ".txt.bz2" file. The time taken for each step is recorded and outputted. After the final output file (which is a ".txt" file) was received, the different times for each steps were then combined to obtain the execution time of the entire encryption subsystem. This process was then repeated for the 3 different example in order to obtain accurate results. It was then compared to the specifications that were fixed to the system prior the experiment.

Test E4: This test was completed in order to determine how much memory usage the encryption subsystem was taking. This was determined using how much RAM was used during the execution of the subsystem. The RAM used was recorded from the task manager of the computer. The range of memory used by the system was then put into a percentage with respect to the overall RAM and compared to the specifications of the system.

G. Results for Simulated Data

All tests time-based tests done in this project were done using the Linux time module. The module produces three times: the real time, user time and system time. the real time is the length of time from start to finish of the computation and therefore this time is the one which will be used to measure the length of each test computation.

Just a reminder, the commands refer to the example files as what they were renamed as and will follow the naming convention mentioned in the experimental method:

"EEE3097S 2022 Turntable Example Data.csv" = "1.csv"

"EEE3097S 2022 Turntable Example Data 2.csv" = "2.csv"

"EEE3097S 2022 Walking Around Example Data.csv" = "3.csv"

The results that were tabulated in this section were taken from the results for "1.csv" (for which a screenshot was provided).

H. Overall functionality Results

Test O1 Results: Due to the lack of availability of the IMU sensor at this stage of the project, this test was not able to be conducted. It was however mentioned in the experimentation setup as it is a crucial aspect for the functioning of the overall system.

Test O2 Results:

```
murphy@ThinkpadT450:/mnt/c/Users/ethan/OneDrive - University of Cape Town/2022/2022S/EEE3097S/data-transfer-microcontroller/Testing$ python3 CompareFiles.py
Enter the first filename: 1.csv
Enter the second filename: 1.txt
----------------------------------
Comparing files
 > 1.csv
 < 1.txt
----------------------------------
The files are 100% the same.
```

Fig. 10: Terminal showing file comparison for the first file tested.

	1.csv	2.csv	3.csv
Test 02 Results of comparison	100%	100%	100%

TABLE X: Table displaying the results of Test O2 on each of the example files

From Figure 45, it can be seen that when using the first file, the input ".csv" data file and the corresponding output ".txt" file are 100% the same. The same can be seen in Table XLI, relating to the second and third files respectively. For all three simulations, the ATP relating to this test has been met.

Test O3 Results:

```
murphy@ThinkpadT450:/mnt/c/Users/ethan/OneDrive - University of Cape Town/2022/2022S/EEE3097S/data-transfer-microcontroller/Testing$ time (bzip2 -vk 1.csv && openssl enc -ae
s-256-cbc -pbkdf2 -in 1.csv.bz2 -out 1out.txt.bz2 -pass pass:ethan && rm 1.csv.bz2  && openssl enc -d -aes-256-cbc -pbkdf2 -in 1out.txt.bz2 -out 1.txt.bz2 -pass pass:ethan &
& rm 1out.txt.bz2  && bzip2 -d 1.txt.bz2)
  1.csv:   11.490:1,  0.696 bits/byte, 91.30% saved, 22429420 in, 1952059 out.

real    0m9.909s
user    0m4.188s
sys     0m0.211s
```

Fig. 11: Terminal simulation time for the first file tested.

	1.csv	2.csv	3.csv
Execution Time	9.909 seconds	9.109 seconds	7.800 seconds

TABLE XI: Table displaying the results of Test O3 on each of the example files

From Figure 46, the first file is the largest of the three tested and it has a simulation time for the entire process which is under 10 seconds. As can be seen from Table XI, the simulation times are all less than the required 10 seconds. These tests were conducted based on a simulation of the compression, encryption, decryption and decompression, For all 3 simulations, the ATP for this test has been met.

Test O4 Results:

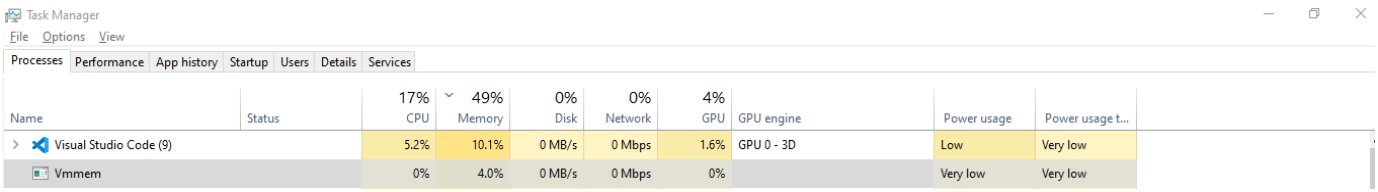

Fig. 12: Terminal showing the RAM in use by the relevant process before the first file was tested.

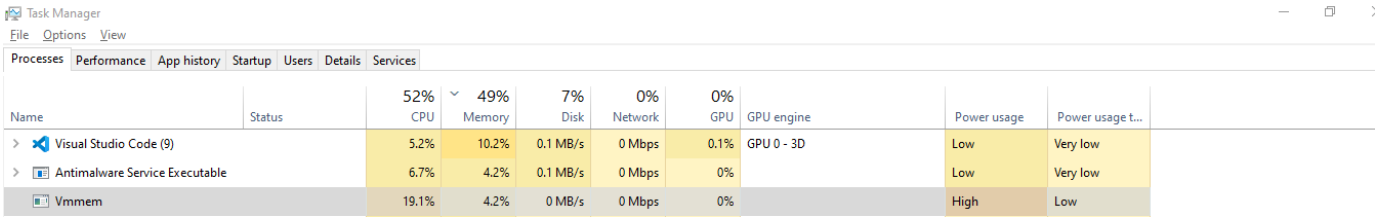

Fig. 13: Terminal showing the RAM in use by the relevant process while the first file was tested.

	1.csv	2.csv	3.csv
Percentage RAM usage	0.2%	0.1%	0.1%

TABLE XII: Table displaying the results of Test O4 on each of the example files

As can be seen from the figures above, the maximum change in the ”Vmmem” process (which is responsible for running the computation from the terminal) is 0.2% of the total RAM which equates to 16MB on a 8GB device. This would be scaled down when running the system on the microcontroller. The RAM on the microcontroller is 192KB, but the data will be transmitted one row at a time and therefore there is enough RAM available to compute what is required. The 0.2% of the RAM utilised when running the system does met the relevant ATP.

I. Results for the Compression subsystem

Test C1 Results:

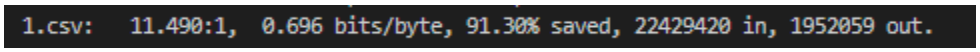

Fig. 14: Terminal showing percentage of data compressed for the first file tested.

	1.csv	2.csv	3.csv
Percentage of data compressed	91.3%	92.31%	85.22%

TABLE XIII: Table displaying the results of Test C1 on each of the example files

As can be seen from the table XIII, the percentage of the data compressed was between 85.2% and 92.3% which meets the ATP pertaining to the percentage of Fourier coefficients being retained being above 25%. All three of the percentages in the figures are well above 25% and therefore the ATP relating to the compression of the data is met.

Test C2 Results:

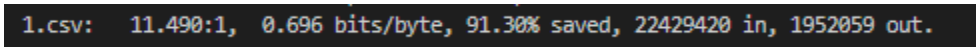

Fig. 15: Terminal showing the compression ratio for the first file tested.

	1.csv	2.csv	3.csv
Compression ratio	11.490	13.008	6.7666

TABLE XIV: Table displaying the results of Test C2 on each of the example files

From Table XIV it can be see that the compression ratio ranges between 6.7 and 13. The compression ratio is dependent on the set of data used, and correlates to the percentage of compression - figure 138. The file with the lowest percentage compression has the lowest compression ratio as the new compressed version of the data was not able to reduced the file size in as many places as the alternate files used. All of the compression ratios are above 5 and therefore the relevant ATP has been met.

Test C3 Results:

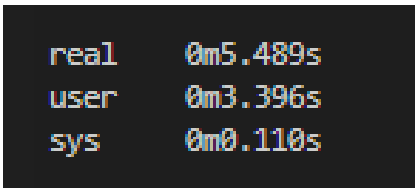

Fig. 16: Terminal showing the time taken for compression while the first file was tested.

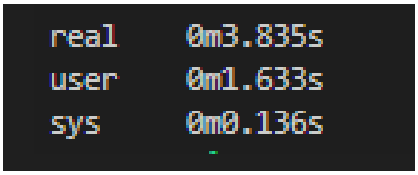

Fig. 17: Terminal showing the time taken for decompression while the first file was tested.

	1.csv	2.csv	3.csv
Execution Time Compression	5.489 seconds	4.727 seconds	3.133 seconds
Execution Time Decompression	3.835 seconds	3.566 seconds	3.566 seconds

TABLE XV: Table showing the results from Test C3 for each of the example files

The total duration spent on compression for the first file test was therefore 9.324 seconds.

The total duration spent on compression for the second file test was therefore 8.293 seconds.

The total duration spent on compression for the third file test was therefore 6.699 seconds.

As can be seen from the above figures, the compression and decompression processing times were measured separately and then added together to calculate the total time spent on the compression algorithm. The maximum time spent on the compression algorithm occurred when testing the largest file, as this file has the most values that require compression. The time taken in this computation was 9.324 seconds which falls within the relevant specification of 20 seconds. Therefore, the ATP relating to test C3 has been met.

Test C4 Results:

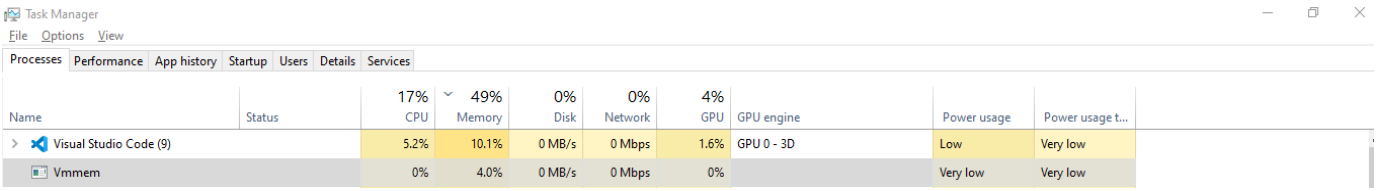

Fig. 18: Terminal showing the RAM in use by the relevant process before the first file's compression was tested.

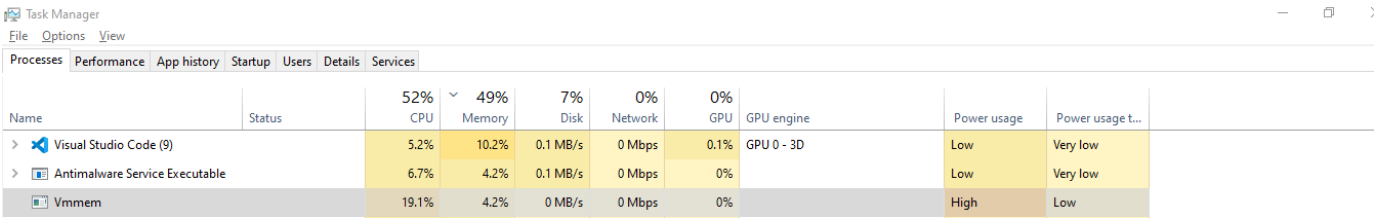

Fig. 19: Terminal showing the RAM in use by the relevant process while the first file's compression was tested.

	1.csv	**2.csv**	**3.csv**
Percentage RAM usage	0.2%	0.1%	0.1%

TABLE XVI: Table displaying the results of Test C4 on each of the example files

As can be seen from the preceding figures, the maximum change in the "Vmmem" process 0.2% of the RAM which is the same as 16MB on a 8GB device. This would be scaled down when running the system on the microcontroller. The RAM on the microcontroller is 192KB, but the data will be transmitted one row at a time and therefore there is enough RAM available to compute the compression and decompression. The 0.2% of the RAM utilised when running the system does met the relevant ATP.

J. Results for the Encryption subsystem

Test E1 Results:

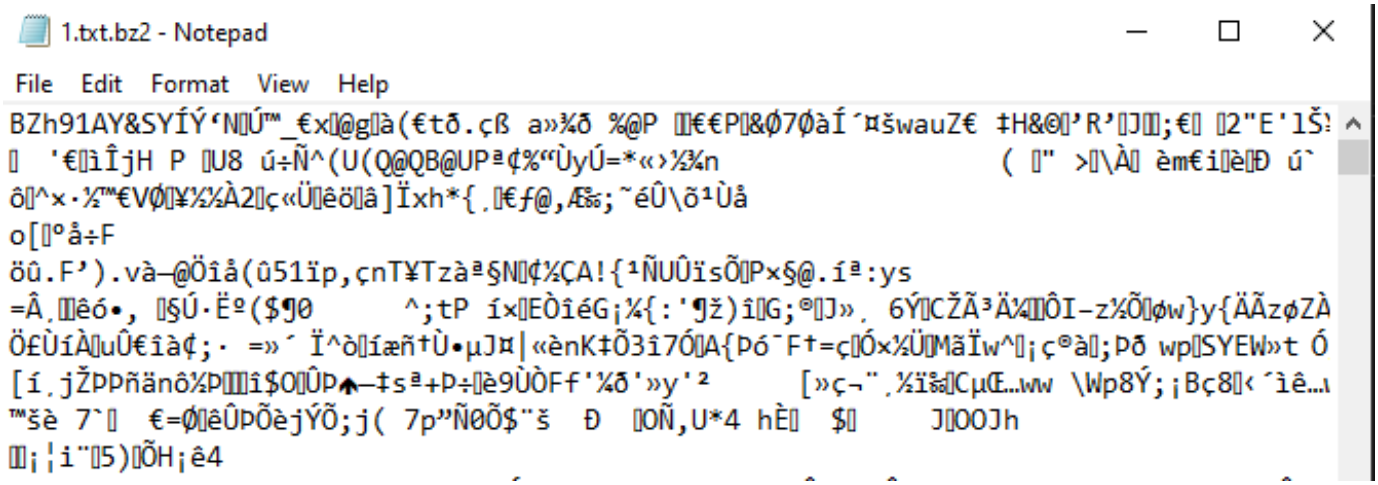

Fig. 20: File showing the encrypted data for the first file tested.

	1.csv	2.csv	3.csv
Encryption Results	Successful	Successful	Successful

TABLE XVII: Table showing the results from Test E1 for each of the example files

Table XXXI shows a sample of the encrypted data for each of the three test after the files have been encrypted. The ATP which indicates a working encryption key ensures the data is safe as it cannot be read and accessed without the key. This ATP has been met.

Test E2 Results:

Fig. 21: Terminal showing file comparison before and after encryption for the first file tested.

	1.csv	2.csv	3.csv
Python Script Results	100%	100%	100%

TABLE XVIII: Table displaying the results of Test E2 on each of the example files

The above table shows that the integrity and the contents of the each of the files in their respective tests is not compromised and that the files before and after encryption and decryption are 100% the same and therefore, the ATP relating to the encryption integrity is met.

Test E3 Results:

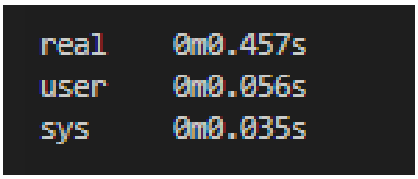

Fig. 22: Terminal showing the time taken for encryption while the first file was tested.

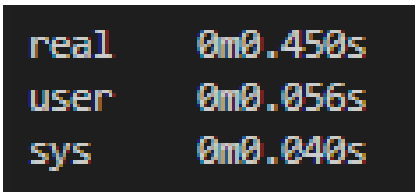

Fig. 23: Terminal showing the time taken for decryption while the first file was tested.

The total duration spent on encryption for the first file test was therefore 0.907 seconds.

The total duration spent on encryption for the second file test was therefore 0.9 seconds.

The total duration spent on encryption for the first file test was therefore 1.151 seconds.

The duration spent on encryption and decryption, as can be seen in the figures above, for each of the three tests totaled a value that was always lower than the specified 20 seconds required computation time. As the maximum computation time was smaller than 1.2 seconds for both encryption and decryption combined, the ATP is therefore met.

Test E4 Results:

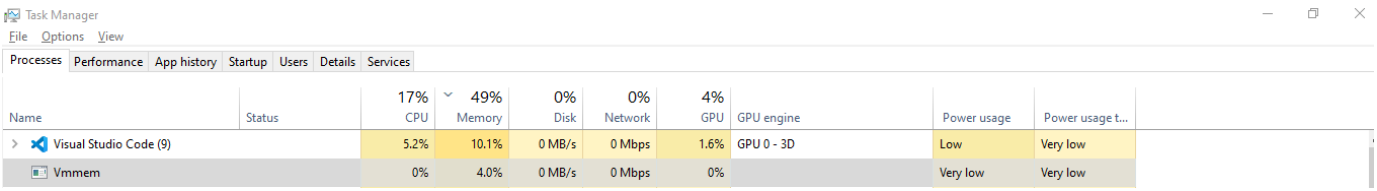

Fig. 24: Terminal showing the RAM in use by the relevant process before the first file's encryption was tested.

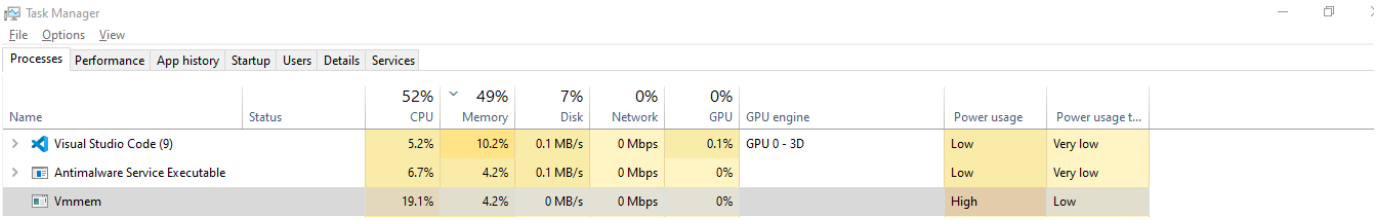

Fig. 25: Terminal showing the RAM in use by the relevant process while the first file's encryption was tested.

	1.csv	**2.csv**	**3.csv**
Execution Time for Encryption	0.457 seconds	0.547 seconds	0.500 seconds
Execution Time for Decryption	0.450 seconds	0.353 seconds	0.651 seconds

TABLE XIX: Table showing the results from Test E3 for each of the example files

	1.csv	2.csv	3.csv
Percentage RAM usage	0.2%	0.1%	0.1%

TABLE XX: Table displaying the results of Test E4 on each of the example files

As can be seen from the preceding figures, the maximum change in the ”Vmmem” process 0.2% of the RAM which is the same as 16MB on a 8GB device. This would be scaled down when running the system on the microcontroller. The RAM on the microcontroller is 192KB, but the data will be transmitted one row at a time and therefore there is enough RAM available to compute the encryption and decryption. The 0.2% of the RAM utilised when running the system does meet the relevant ATP.

V. Validation using the IMU

A. Comparison between the IMU sensors

It was decided to first include a comparison between the ICM20948 IMU sensor that is used during the experiments of the design and the ICM20649 which is the IMU sensor which will be used on the actual SHARC Buoy design. These differences are important to note as they have certain features which are different and that will possibly affect the design. Using the datasheet of the two sensors, the following table was constructed showing the features of each sensors.

Features of the IMU	**ICM20948**	**ICM20649**
Gyroscope	3-Axis Gyroscope with Programmable FSR of ±250 dps, ±500 dps, ±1000 dps, and ±2000 dps	3-Axis gyroscope with programmable FSR of ±500 dps, ±100 dps, ±2000 dps, and ±4000 dps
Accelerometer	3-Axis accelerometer with programmable FSR of ±2g, ±4g, ±8g, and ±16g	3-Axis accelerometer with programmable FSR of ±4g, ±8g, ±16g, and ±30g
Interface Protocol	Host interface: 7 MHz SPI or 400 kHz I2C	Host interface: 7 MHz SPI or 400 kHz I2C
Temperature sensor	Digital-output temperature sensor with range of 40°C to +85°C (Compass:-30C to +85C)	Digital-output temperature sensor with range of 40°C to +85°C
ADC and filters	On-Chip 16-bit ADCs and Programmable Filters	On-Chip 16-bit ADCs and Programmable Filters
Compliants	RoHS and Green compliant	RoHS and Green compliant
VDD range	VDD operating range of 1.71V to 3.6V	VDD operating range of 1.71V to 3.6V
MEMS structure	MEMS structure hermetically sealed and bonded at water level	MEMS structure hermetically sealed and bonded at water level
Magnetometer	AK09916 3-axis magnetometer from Asahi Kasei Microdevices Corporation	No magnetometer

TABLE XXI: Table showing the features and differences of both the ICM2098 IMU sensor and the ICM20649 IMU sensor

Both of the IMU sensors are very similar and have the operation for most of their features. It is important to note, however, that the programmable FSR for both the gyroscope and the accelerometer is lower for the ICM20948 than the ICM20649. This can cause a problem in the experimenting of the data as it might not fully represent what the ICM20649 is capable of doing with regards to these sensors' values. It is therefore assumed that the ICM20649 will operate in a lower range from its maximum range in order to match the FSR of the ICM20948.

The second major difference between the two sensors is the fact that the ICM20948 has a magnetometer which will not be present in the actual sensor used on the SHARC Buoy model (ICM20649). This means that the magnetometer sensor values gathered from the experiments can be disregarded as they will not be available in the final design of the project.

For further information about the features and differences of both of the IMU sensors links to the datasheets were provided in the Appendix Section.

B. Salient Features

The salient features are defined as characteristics of the IMU sensor currently being used for testing that are vital to the intended functioning of the device. In this case, the salient features for both sensors are the Gyroscope, Accelerometer, Temperature and the Interface Protocol used for data transmission. The salient features of both of the IMU sensors are displayed in table XXI. The differences include the disparity in the Full Scale Range (FSR) between the two sensors which is discussed in the comparison above. Additionally, the lack of a magnetometer in the ICM20649 is also something to take in consideration. As a result of this, the magnetometer values gathered in the experimentation will be disregarded. This is done in order for the test files and the experimentation to represent as accurately as possible what the actual buoy will be able to read and produce.

C. Steps to ensure working functionality of IMU on buoy

In order to make sure the system will be working on the actual buoy it is crucial that no extra features from the ICM20948 are being tested or are taken into consideration. This is due to the fact that the ICM20649 IMU sensor will be the one used for the actual buoy and therefore those extra features will not be available. Using the comparison between the two IMU sensors, the experimentation can be adapted to make sure the overall functionality of the system will be the same when it is implemented on the actual buoy. These differences are made more clear in table XXI and help distinguish what must be taken into consideration when testing the IMU sensor.

During the experiments, the gyroscope and the accelerometer Full Scale Range will be set to a maximum of +/- 4000dps and +/-30g respectively. This is doe in order to mimic what the ICM20649 will be capable of and get an accurate representation of what the actual buoy will be able to do. The magnetometer values gathered were also completely disregarded and were not even provided in the files that were used for the testing.

D. Validation tests to ensure working functionality of IMU

The following tests performed regarding the IMU were effectuated in order to make sure that the IMU is working as expected. These validation tests include what the expected output is and what the behavior of the sensor should look like.

Test I1:

This validation test 1 for the IMU operation consists of testing the overall functionality of the IMU sensor. This means that the IMU sensor must receive power from the STM microcontroller and that it is able to correctly interact with it. This test is validated if the IMU sensor has its light switch on signifying it is receiving power. It is also validated if the STM microcontroller is successfully connected to it. This is checked by initializing the connection in the STM microcontroller coding interface and it is expected that the connection is successful when the code is built.

Test I2:

This second validation test for the IMU operation is to test the ability of the IMU sensor to provide its data to the serial monitor through the STM microcontroller. This data must be outputted in the correct format. This expected output is a string containing the data displayed on the serial monitor.

Test I3:

This third validation test for the IMU operation is testing the gyroscope data. In order to test this, the IMU sensor STM microcontroller system is rotated around for multiple rotations. The expected output for these tests are some dps value which is not zero as the IMU sensor was moved from its original position. This same test is also performed while keeping the IMU sensor completely still. For this test, the expected output is to be 0dps. Using both of these tests will ensure that the gyroscope sensing is fully functional.

Test I4:

This validation test for the IMU operation is regarding the accelerometer data. This test can be completed first by testing whether there is any output when the IMU sensor is kept entirely still. The expected output of this test is 0g. This test is then repeated to check whether an output is detected when the device undergoing an acceleration. The device is being moved first steadily and then its movement is faster and faster. This expected output is now some output in ”g”.

Test I5:

This validation test for the IMU operation consists of testing the temperature data. First, the temperature is being recorded in the room using a separate device (thermometer). The sensor is then used to record the temperature and the temperature value is analysed in order to see if it was recorded properly. The expected output is a value in degrees which must be similar to the one recorded with the thermometer. Additionally, the sensor can be placed in a cooler spot, such as a fridge, and the temperature can now be recorded again. This time the expected output is a lower temperature in degrees.

Test I6:

This validation test for the IMU operation consists of testing the sensor’s reaction to noise. Noise is everywhere and it very important to consider when dealing with components such as sensors which carry sensible data. There are various types of noise such as Thermal Noise (white noise) or shot noise which could have an effect on the value gathered by the sensor. To test this, air was blown onto the sensor and seeing if the data fluctuated by a large margin or not. Additional heat was put onto the pins and the sensor to see if it would lead to different data fluctuations (except for the temperature data). This was then compared a set of value in the same conditions but without the added noise. The expected output was a change in the data gathered that is not too noticeable and that the noise applied does not affect the values results drastically.

E. Results of the Validation tests

Test I1 Results:

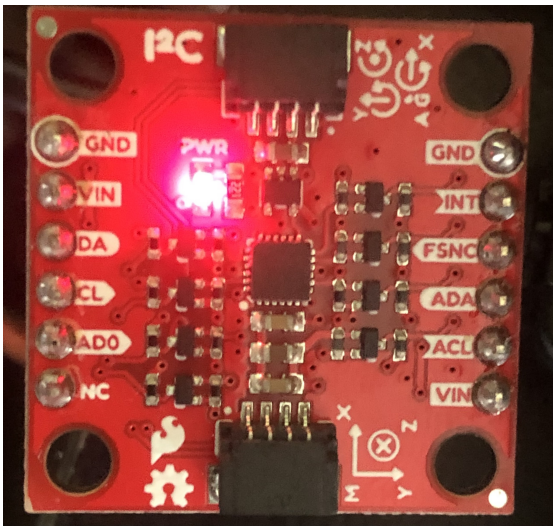

Fig. 26: Image showing the red power light on the sensor shining.

As can be seen from the figure above, the red LED on the sensor to indicate the power being active was on while the sensor was in use, indicating the input voltage supplied to the sensor is sufficient to support the reading of data from any of the available sensors.

Test I2 Results:

Temperature (°C)	Gyroscope (X)	Gyroscope (Y)	Gyroscope (Z)	Accelerometer (X)	Accelerometer (Y)	Accelerometer (Z)
0.06146	-0.060976	-0.060976	0	-0.004883	-0.000488	0.998535
17.444843	-0.060976	0.060976	-0.182927	-0.001953	-0.006836	1.008789
17.396921	0	-0.182927	-0.182927	-0.004395	0	1.002441
17.253153	-0.121951	0	-0.182927	-0.000977	-0.00293	1.007324
17.540689	-0.365854	-0.182927	0.304878	0.000977	-0.007324	1.007324
17.205229	-0.121951	0	0	-0.004395	0	1.009766
17.157307	-0.182927	0.060976	-0.121951	-0.003418	-0.005371	1.005371
17.253153	0.060976	-0.060976	0.243902	-0.004883	-0.008301	1.004395

Fig. 27: Image showing the data being displayed in a file after being read from the IMU sensor.

This figure shows the format of the data that has been read from the IMU sensor and also shows that the data has been read correctly from the serial monitor used - pyserial - which each of the lines of data to a ”.csv” file.

Test I3 Results:

19.170067	-0.365854	0.243902	0.121951	-0.071289	0.095703	1.024414
19.265911	-0.243902	-0.060976	-0.182927	-0.069336	0.099609	1.023926
18.978374	-0.060976	0.060976	-0.060976	-0.063477	0.099121	1.025879
19.122143	0	0.060976	-0.182927	-0.065918	0.095703	1.02832
19.122143	0	0	0	-0.065918	0.09668	1.026367
19.265911	-0.304878	0.304878	-0.243902	-0.072266	0.098145	1.028809
19.265911	-0.121951	-0.243902	0	-0.070801	0.095703	1.025879
19.170067	-0.365854	-0.304878	0.182927	-0.063477	0.09668	1.031738

Fig. 28: Image showing a reading where the gyroscope is set to the origin.

19.697218	-0.182927	-0.060976	0.731707	-0.180664	0.172852	1.083496
19.649294	0.97561	0.548781	-1.463415	-0.181641	0.170898	1.073242
19.505526	0.304878	-0.182927	-1.280488	-0.177734	0.170898	1.084473
19.840986	-9.939025	96.768295	359.817078	0.134277	0.22998	1.399414
19.265911	14.695123	2.5	10.304878	-0.029785	0.18457	1.015137
19.505526	-1.341463	0.731707	0.914634	-0.06543	0.130371	1.026855
19.55345	-4.634146	-0.731707	0.914634	-0.064941	0.12207	1.027344

Fig. 29: Image showing readings where the gyroscope is a non-zero value in each of the axis directions.

As can be seen in the above figures, the gyroscope readings are centered around the origin (0, 0, 0) and do change in different directions which indicates that the gyroscope sensor on the IMU is configured correctly and connected properly as well.

Test I4 Results:

18.115763	120.243904	92.68293	70.914635	0.004395	0.100586	0.94873
18.259533	120.609756	92.865852	71.219513	0.002441	0.104492	0.949219
18.307455	120.121956	92.865852	71.219513	0.006348	0.098145	0.940918
18.451223	119.939026	92.804878	71.036591	0	0	0
18.115763	119.634148	92.68293	70.853661	0.010742	0.102051	0.947754
18.403301	120.121956	92.865852	71.097565	0.003418	0.100098	0.949219
18.067841	120.18293	92.865852	71.036591	0.006348	0.096191	0.942383

Fig. 30: Image showing a reading where the accelerometer is equal to zero in all directions.

22.62051	119.390244	66.219513	69.451218	0.709961	-0.341797	2.65918
22.524666	128.353668	57.012196	39.878048	0.491699	-0.364746	2.688965
23.003893	113.658539	72.865852	92.621956	0.696777	-0.486328	2.58252
23.003893	113.353661	102.865852	36.524391	0.977539	-0.538086	2.479004
23.051817	122.439026	114.390244	40.42683	1.134277	-0.677734	2.493164
23.003893	141.341461	69.634148	85.914635	0.680176	-0.500488	2.553223
22.955971	126.402443	89.695122	77.317078	0.613281	-0.539551	2.554199

Fig. 31: Image showing a sample of data showing the non-zero and changing values of the accelerometer.

Figures 30 and 31 show the accelerometer values set all zero values in the possible directions of motion when the sensor has not been moved. These figures also show the accelerometer values as non-zero in the case of an acceleration in any of the three directions (x, y, z).

Test I5 Results:

21.470364	123.292686	118.353661	41.646343	0.5625	0.598633	2.729004
21.949591	161.219513	119.329269	101.829269	0.60791	0.369141	2.851074
21.805822	132.987808	97.317078	82.256096	0.43457	0.23291	2.833008
21.997515	134.878052	120.121956	80.243904	0.472168	0.40918	2.601074
22.093359	124.939026	55.548782	70.487808	0.665039	0.433105	2.813477
22.62051	147.012192	96.707321	59.512196	0.450684	0.562012	2.758301
22.524666	119.207321	94.329269	69.451218	0.497559	0.687988	2.764648
22.668434	48.902439	65.426834	101.585365	0.618164	0.715332	2.625977

Fig. 32: Image showing the temperature sensor measuring the current temperature in the room it is in.

10.352262	0.182927	0	-0.182927	-0.241211	0.0625	0.986816
10.016804	0.121951	-0.121951	-0.182927	-0.244629	0.064941	0.986328
9.96888	0	-0.060976	0.121951	-0.237793	0.066406	0.97998
10.208494	0.182927	-0.121951	-0.182927	-0.243652	0.064941	0.978516
10.016804	0.060976	-0.365854	0	-0.245117	0.067383	0.983398
10.016804	0.182927	-0.304878	0.243902	-0.23877	0.070801	0.987305
9.96888	0.121951	-0.243902	0.060976	-0.239746	0.072754	0.982422

Fig. 33: Image showing the data when the sensor is put in the fridge.

20.607752	-125.792686	93.780487	282.865875	-1.433105	-1.296387	1.811035
20.511906	7.073171	-0.792683	1.52439	-0.209961	-1.76123	0.73877
21.278671	-7.743903	-6.219512	-4.634146	-1.046387	-1.630859	1.056152
21.805822	-1.280488	-12.560976	-4.207317	-0.982422	-1.626465	1.231445
22.668434	31.585367	29.268293	1.097561	-1.101562	-1.716309	1.109375

Fig. 34: Image showing the data when the sensor has been taken out of the fridge.

As can be seen from the above figures, the temperature sensor accurately measures the temperature of the its environment and does adapt to measure the cooler temperatures. The sensor does return to reading the room temperature once it has been taken out of the fridge.

Test I6 Results:

22.332973	-12.865854	33.353661	-13.47561	-0.655273	-0.435059	2.350098
21.949591	-171.402451	213.597565	-34.024391	-0.791016	-0.152344	2.287598
21.566208	76.402443	70.121956	107.439026	0.094238	0.157227	2.591309
21.805822	-2.256098	-57.439026	-0.97561	-0.535156	0.200684	2.491211
21.805822	30.121952	29.512196	113.353661	0.211914	0.320312	3.043945

Fig. 35: Image showing the data when excessive noise has been applied to the sensor.

24.307455	120.243904	92.743904	71.341469	0.007324	0.106445	0.951172
26.403301	120.426834	92.987808	71.219513	0.008789	0.10791	0.948242
28.971994	120.121956	92.926834	71.219513	0.006348	0.105469	0.948242
28.211609	120.121956	93.048782	71.036591	0.006836	0.110352	0.951172
28.115763	120.18293	92.926834	71.219513	0.00293	0.108398	0.947266
27.451223	120.36586	92.926834	71.219513	0.001465	0.11084	0.942383

Fig. 36: Image showing the data when heat has been applied to the IMU while applying the excessive noise.

27.307455	120.36586	92.804878	71.463417	0.006348	0.101562	0.946289
27.259533	120.36586	92.68293	71.036591	0.009766	0.106445	0.946777
28.307455	120.304878	92.743904	71.036591	0.005859	0.103516	0.949219
28.211609	120	92.865852	71.158539	0.006836	0.104492	0.946289
28.211609	120.243904	92.804878	71.036591	0.006836	0.103027	0.944824

Fig. 37: Image showing the data when heat has been applied to the IMU without the excessive noise.

The above figures show the change in the results read from the sensor when excessive noise is applied. The change seen shows the lack of an effect felt due to the noise. The change in the data values correlates with the rest of the data read from the IMU. The reading measured with heat applied, as can be seen in the Temperature sensor values, as well as the noise did not affect the values either - besides the temperature values.

F. Experimental Method for Final Results

Each test was conducted by reading the input data from the IMU sensor. The data sample collected from the sensor used for the testing contains one reading from all the sensors on the IMU. This test was repeated a total of 10 times, with the average computation time taken for each of the processes. This was done to record more accurate results and also to account for the effects of memory warming.

The data was read from the sensor in the form of a String which was stored in a variable on the microcontroller before being transmitted to the Laptop using a UART connection. The UART connection displayed the values for all the different features on PuTTY, a type of Serial Monitor, from which it was copied into a ”.csv” file ready for processing. The data that is outputted to PuTTY is a string of values which contain the IMU sensor data. The data in the ”.csv” was converted to a ”.txt” file format in the encryption process and therefore at the output as well because the ”.txt” file format allowed for faster file processing for a file of the same size as a ”.csv” file.

The files that were used for the experiments were all retrieved from the IMU sensor connected to the STM microcontroller. Three files were used and all had different delays between the readings of data taken from the IMU sensor. The information about these files are showed below:

The first file that has a delay 100ms between each retrieval of the IMU sensor data was named ”delay100” and has a file size of 85.06Kb. The second file that has a delay 200ms between each retrieval of the IMU sensor data was named ”delay200” and has a file size of 68.93Kb. The third file that has a delay 1000ms (or 1second) between each retrieval of the IMU sensor data was named ”delay1000” and has a file size of 42.24Kb.

1) Experiments for the Overall Functionality of the System

Test O1: This test was conducted to identify and analyse the ability of the IMU sensor to send data to the STM microcontroller in the adequate format. The format that must be delivered to the STM microcontroller is a string as it is what can be transferred from the microcontroller to the laptop over UART communications. Therefore, the expected input for this test was a string variable from the IMU sensor. To test this, data from the sensor was read and sent to the STM microcontroller before being transmitted to the laptop for processing. The format of the data received at the STM was then identified. The output was expected to be a set values which are separated with commas.

*Note: This test has been adapted from the First Progress Report to account for the change of the Milestone requirements. The previous version of this test referred to a ”.csv” file at the output of the sensor reading. This file was then sent to the compression system for processing. This has been altered so that the sensor will produce a string value that can be quickly transferred to the laptop used for compression and encryption testing.

Test O2: This test was effectuated to determine the accuracy and correctness of the data throughout the overall process. The original file, which was retrieved from the serial monitor, that is passed onto the compression subsystem is compared to the final output of the entire process. This comparison was done using a Python script named ”CompareFiles.py” which is available on the git repository of the project. The Python script compares the two files and checks if there are any differences or missing parts. It then outputs a similarity percentage, with 100% meaning that the data in the files are identical. The expected output of this experiment is that the original data and the final output data are the same with no data lost/corrupted.

Test O3: This test was done in order to determine the time taken for this entire system to run. The file, that was gathered from the STM microcontroller, was inputted into the all the algorithms which run one after the other using the one command and the file goes through each system. The time taken for each step is recorded and outputted. After the final output file was received, the different times for each steps are then combined to obtain the execution time of the entire process. This process was then repeated for each of the example files in order to obtain accurate results. It was then compared to the specifications that were fixed to the system prior the experiment. The expected output must be an execution time that is less than the specifications (less than 60 seconds).

Test O4: This test was completed in order to determine the memory usage of the overall functionality of the system. This was determined based on how much RAM was used during the execution of the system. This experiment was done on a laptop with 8GB of RAM. The percentage RAM used by the system was determined by recording how much memory was used by the entire system. The STM microcontroller has a static RAM of 192KB, so the experiment will take this in consideration as well. Additionally, this test can be used to check the CPU usage of the overall system. The percentage of CPU used can be retrieved and compared to the STM usages.

2) Experiments for the Compression subsystem

Test C1: This test was effectuated to determine the accuracy and correctness of the data throughout the compression subsystem. The original file, which is the example ”.csv” file was inputted into the compression and then the decompression algorithms. The final output after the decompression, a ”.txt” file, is then compared back to the original file. This comparison was done using the comparison Python script, ”CompareFiles.py”. The Python script compares the two files and checks if there are any differences or missing parts. It then outputs a similarity percentage, with 100% meaning that the data in the files are identical. This is then compared to the specifications which required the lowest 25% of the Fourier coefficients.

Test C2: This test aims to determine the compression ratio. This ratio was calculated using the original file size and dividing it by the compressed file size.

$$compression\ ratio = \frac{original\ file\ size}{compressed\ file\ size}$$

The input for this block is a ".csv" file and the output is a ".csv.bz2" file. The expected output was a compression ratio that respects the specifications (greater than 5).

Test C3: This test was done in order to determine the time taken for the compression and decompression to run. The example files were inputted (".csv") into the compression algorithm, which outputs a ".csv.bz2" file, and then the decompression algorithm - which takes in a ".txt.bz2" file and outputs a "txt" file. The time taken for each step is recorded and outputted. After the final output file was received, the different times for each steps were then combined to obtain the execution time of the entire compression subsystem. It was then compared to the specifications that were fixed to the system prior the experiment. The expected output of this test was an execution time of less than 20 seconds.

Test C4: This test was completed in order to determine how much memory usage the compression subsystem was taking. This was determined using how much RAM was used during the execution of the subsystem. The RAM used was recorded from the task manager of the computer. The range of memory used by the system was then put into a percentage with respect to the overall RAM and compared to the specifications of the system. Additionally, this test can be used to check the CPU usage of the compression subsystem and process. The percentage of CPU used can be retrieved and compared to the STM usages.

3) Experiments for the Encryption subsystem

Test E1: This test was done to evaluate what the encryption algorithm was doing to the data and whether the encrypted data could be read without the cryptographic key or not. The encrypted data was analyzed to check if it was secure. The example file was inputted through the encryption algorithm as ".csv.bz2" file. The encrypted data was the output of this test as a ".txt.bz2" file which was then analyzed to see if it was secure and properly encrypted. The data was tested against other possible encryption keys to ensure the safety of the data.

Test E2: This test was effectuated to determine the accuracy and correctness of the data throughout the encryption subsystem. The original file, which is the example ".csv.bz2" file was inputted into the encryption and then the decryption algorithms. The final output after the decryption, in the form of a ".txt.bz2" file, is then compared back to the original file. This comparison was done using the comparison Python script, "CompareFiles.py". The Python script compares the two files and checks if there are any differences or missing parts. It then outputs a similarity percentage, with 100% meaning that the data in the files are identical.

Test E3: This test was done in order to determine the time taken for the encryption and decryption to run. The files that are inputted for this test are coming from the compression subsystem and therefore the input is a ".txt.bz2" file. The time taken for each step is recorded and outputted. After the final output file (which is a ".txt" file) was received, the different times for each steps were then combined to obtain the execution time of the entire encryption subsystem. It was then compared to the specifications that were fixed to the system prior the experiment.

Test E4: This test was completed in order to determine how much memory usage the encryption subsystem was taking. This was determined using how much RAM was used during the execution of the subsystem. The RAM used was recorded from the task manager of the computer. The range of memory used by the system was then put into a percentage with respect to the overall RAM and compared to the specifications of the system. Furthermore, this test can be used to check the CPU usage of the encryption subsystem and process. The percentage of CPU used can be retrieved and compared to the STM usages.

G. Final Results and Analysis

All time-based tests done in this project were done using the Linux time module. The module produces three times: the real time, user time and system time. the real time is the length of time from start to finish of the computation and therefore this time is the one which will be used to measure the length of each test computation.

1) Initial analysis of the data in the files

The data that was in the test files were all analysed prior to the experimentation in order to have an idea of what to expect. The FFTs for the Accelerometer, the gyroscope as well as the temperature values were plotted using python.

The data in the files plotted and simulations were made using Jupiter Notebook. The following figures give an indication of what the data from these files are. By analysing this data, an additional assessment can be made on what can be expected from the time, frequency and temperature outputs of the IMU sensors. The simulations on the Jupiter Notebook were done for each of the three files. Only the graphs from the ”delay100.csv” file was shown below. The rest of the graphs can all be found in appendix A.

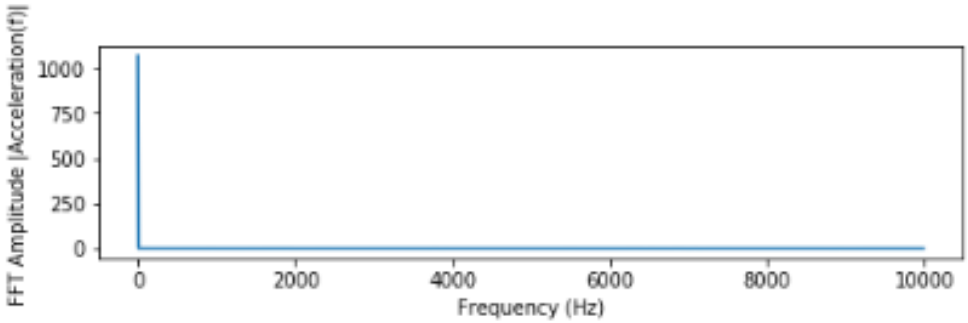

Fig. 38: Graph showing the FFTs of the accelerometer values of the IMU sensor straight from the file ”delay100.csv” retrieved from the STM

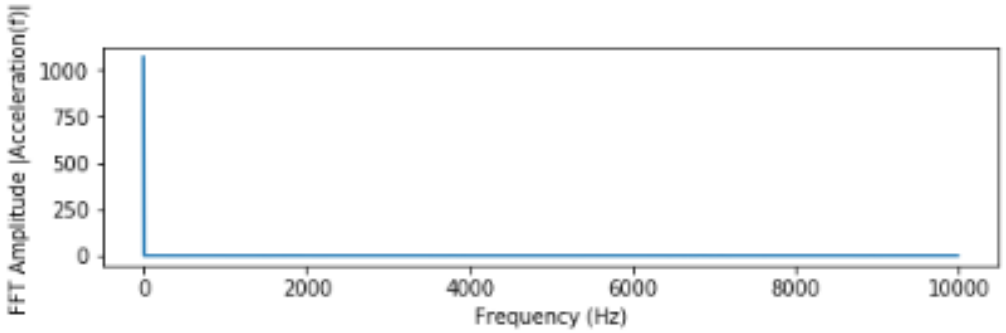

Fig. 39: Graph showing the FFTs of the accelerometer values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The accelerometer values have not been changed by the processing system, meaning the transmission was successful.

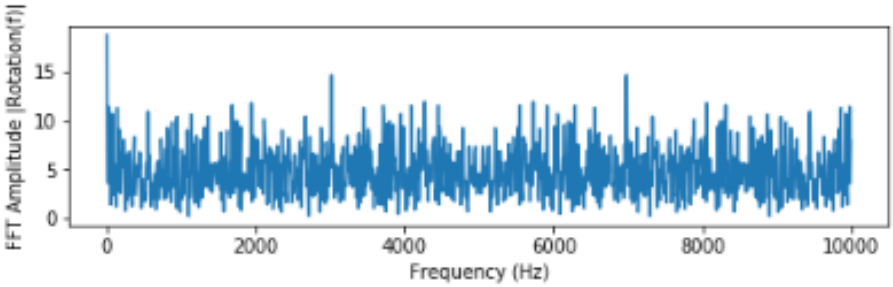

Fig. 40: Graph showing the FFTs of the gyroscope values of the IMU sensor straight from the file ”delay100.csv” retrieved from the STM

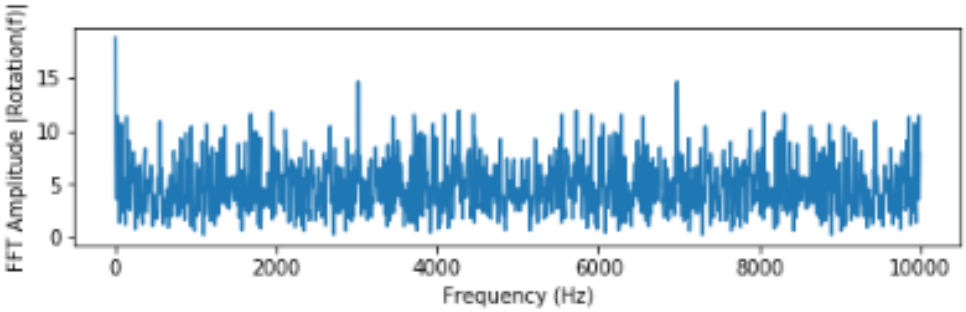

Fig. 41: Graph showing the FFTs of the gyroscope values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The gyroscope data values have not changed.

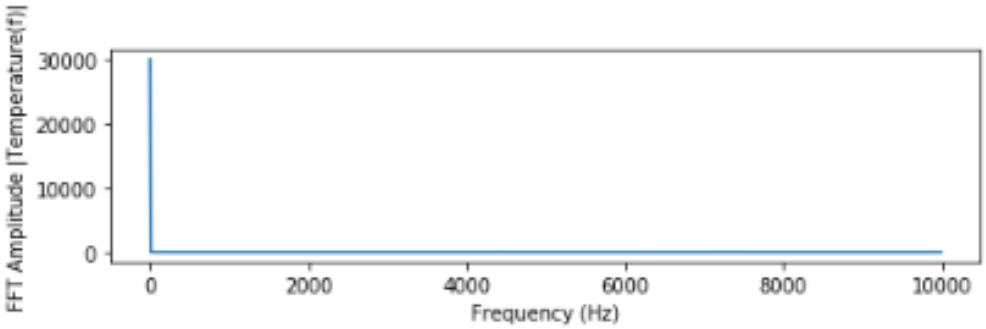

Fig. 42: Graph showing the FFTs of the temperature values of the IMU sensor straight from the file ”delay100.csv” retrieved from the STM

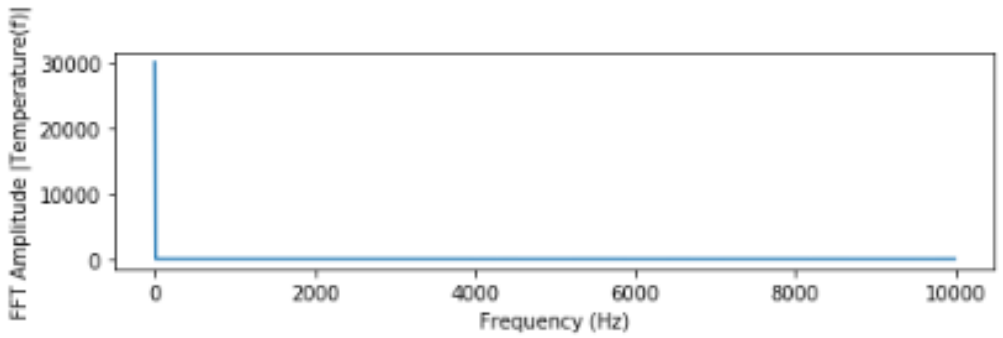

Fig. 43: Graph showing the FFTs of the temperature values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The temperature data values have not changed.

2) Overall functionality Results

Test O1 Results:

Temp	Gyro (x)	Gyro (y)	Gyro (z)	Accel (x)	Accel (y)	Accel (z)
27.98476	-0.18293	-0.06098	0.060976	-0.00098	0.000488	1.004883
28.22437	0.121951	0	0.304878	0.005859	-0.00684	1.007812
28.36814	-0.36585	0.121951	-0.30488	-0.00342	-0.00147	1.001953
28.36814	-0.18293	0.060976	0.121951	0.000977	-0.00391	1.012207
28.36814	0.243902	-0.2439	0.060976	-0.00293	0.001465	1.009277
28.0806	0	-0.30488	-0.18293	0.000488	0.004883	1.006836
28.03268	0	-0.06098	0.060976	-0.00098	-0.00098	1.011719
28.0806	0.243902	-0.12195	0.121951	-0.00098	0.002441	1.011719
28.22437	0	0.060976	-0.12195	-0.00293	-0.00049	1.01416
28.55983	-0.18293	-0.12195	-0.06098	0.000977	0.000977	1.012207
28.36814	0.365854	0.121951	-0.18293	-0.00586	-0.00293	1.011719
28.36814	0.304878	-0.12195	-0.06098	0.000488	-0.00195	1.011719
28.36814	0.060976	-0.30488	-0.12195	-0.00147	0.000977	1.008789
28.32022	-0.2439	0.182927	-0.06098	0	0.002441	1.008301
28.55983	-0.18293	-0.06098	0.243902	-0.00293	-0.00635	1.007324
28.32022	0.304878	-0.2439	-0.12195	0	-0.00391	1.01123
28.55983	-0.12195	-0.2439	0.121951	-0.00098	0.000977	1.009277
28.22437	0	-0.36585	-0.06098	-0.00293	0	1.003906
28.55983	0	-0.2439	0.060976	-0.00147	-0.0044	1.01123
28.36814	-0.06098	-0.12195	-0.06098	-0.0044	-0.00147	1.01123

Fig. 44: Picture showing that the IMU sensor and STM outputted data

From Figure 44, it can be seen that the IMU sensor provides data to the STM microcontroller and that it is retrieved in the correct format. For all three simulations, the ATP relating to Specification S1.1 has been met.

Test O2 Results:

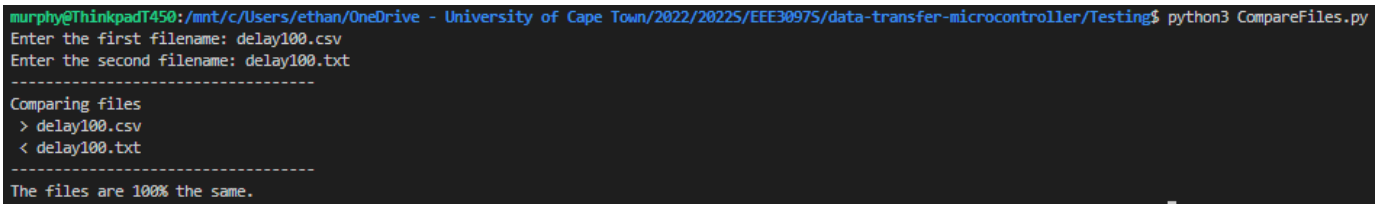

Fig. 45: Terminal showing file comparison for the first file tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Test 02 Results of comparison	100%	100%	100%

TABLE XXII: Table displaying the results of Test O2 on each of the example files

From Figure 45, it can be seen that when using the first file, the input ".csv" data file, and the corresponding output ".txt" file that the two files are 100% the same. The same can be seen in Table XXII, relating to the second and third files respectively. For all three simulations, the ATP relating to Specification S2.1 has been met.

Test O3 Results:

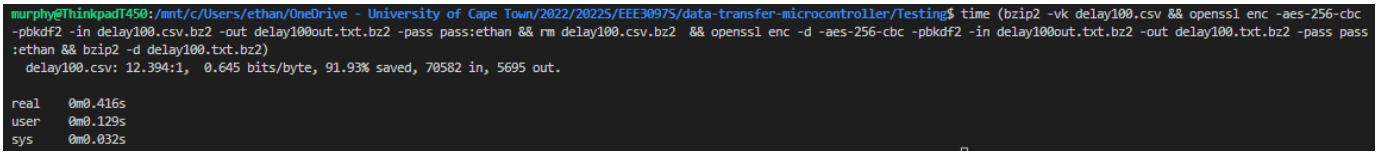

Fig. 46: Terminal simulation time for the first file tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Execution Time	0.416 seconds	0.393 seconds	0.246 seconds

TABLE XXIII: Table displaying the results of Test O3 on each of the example files

From Figure 46, the first file is the largest of the three tested and it has a simulation time for the entire process which is under 10 seconds. As can be seen from Table XXIII, the simulation times are all less than the required 10 seconds. These tests were conducted based on a simulation of the compression, encryption, decryption and decompression, For all 3 simulations, Specification S3.1's ATP has been met.

Test O4 Results:

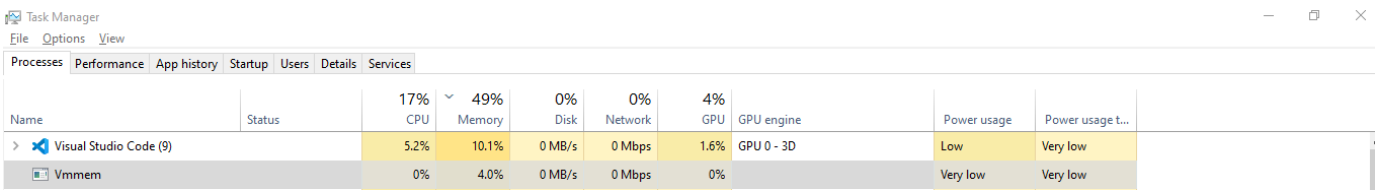

Fig. 47: Terminal showing the RAM in use by the relevant process before the first file was tested.

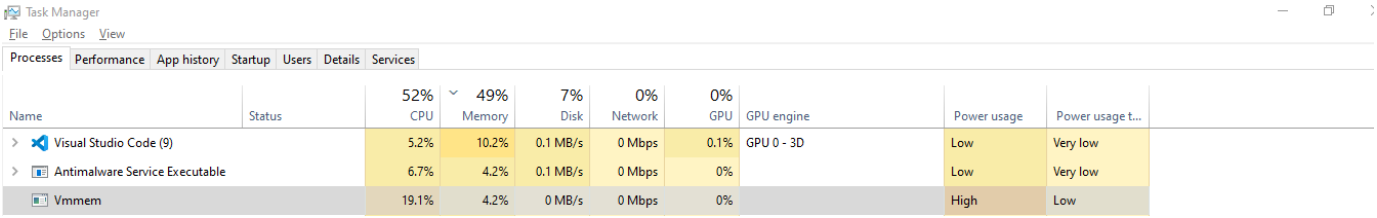

Fig. 48: Terminal showing the RAM in use by the relevant process while the first file was tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Percentage RAM usage	0.2%	0.1%	0.1%

TABLE XXIV: Table displaying the results of Test O4 on each of the example files

As can be seen from the figures above, the maximum change in the ”Vmmem” process (which is responsible for running the computation from the terminal) is 0.2% of the total RAM which equates to 16MB on a 8GB device. This would be scaled down when running the system on the microcontroller. The RAM on the microcontroller is 192KB, but the data will be transmitted one row at a time and therefore there is enough RAM available to compute what is required. The 0.2% of the RAM utilised when running the system does meet the relevant ATP - which relates to Specification S4.1.

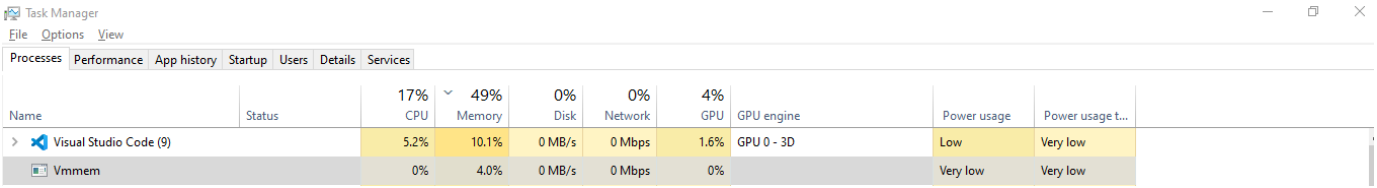

Fig. 49: Terminal showing the CPU in use by the relevant process before the first file was tested.

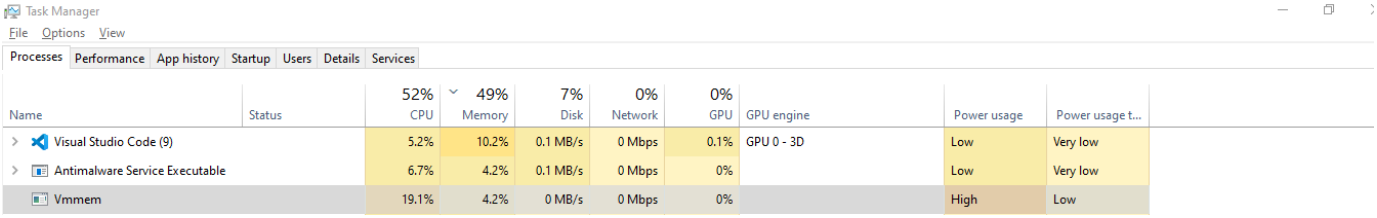

Fig. 50: Terminal showing the CPU in use by the relevant process while the first file was tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Percentage RAM usage	0.91%	0.46%	0.4%

TABLE XXV: Table displaying the results of Test O4 on each of the example files

As can be seen from the figures above, the maximum change in the ”Vmmem” process (which is responsible for running the computation from the terminal) is 0.91% of the total available CPU of the device used. The CPU available of the computer is however different than the one used for this experiment. This would therefore be scaled down when running the system on the microcontroller but the percentage usage would be the same. The ATPs were revised after relating them to the Specifications. An ATP for CPU usage has been added. The 0.91% of the CPU percentage utilised when running the system does meet the relevant ATP.

3) Results for the Compression subsystem

Test C1 Results:

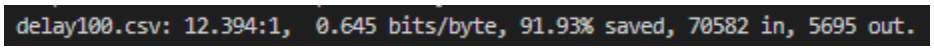

Fig. 51: Terminal showing percentage of data compressed for the first file tested.

	delay100.csv	delay200.csv	delay1000.csv
Percentage of data compressed	91.93%	92.03%	90.97%

TABLE XXVI: Table displaying the results of Test C1 on each of the example files

As can be seen from the table XXVI, the percentage of the data compressed was between 90.97% and 91.93% which meets the ATP pertaining to the percentage of Fourier coefficients being retained being above 25%. All three of the percentages in the figures are well above 25% and therefore the ATP for Specification S2.2 which relates to the compression of the data is met.

Test C2 Results:

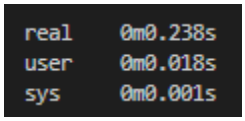

Fig. 52: Terminal showing the compression ratio for the first file tested.

	delay100.csv	delay200.csv	delay1000.csv
Compression ratio	12.394	12.548	11.072

TABLE XXVII: Table displaying the results of Test C2 on each of the example files

From Table XXVII it can be see that the compression ratio ranges between 11.072 and 12.394. The compression ratio is dependent on the set of data used, and correlates to the percentage of compression - figure 138. The file with the lowest percentage compression has the lowest compression ratio as the new compressed version of the data was not able to reduced the file size in as many places as the alternate files used. All of the compression ratios are above 5 and therefore the relevant ATP for Specification 3.2 has been met.

Test C3 Results:

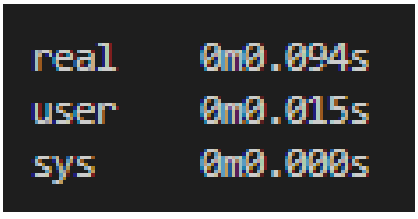

Fig. 53: Terminal showing the time taken for compression while the first file was tested.

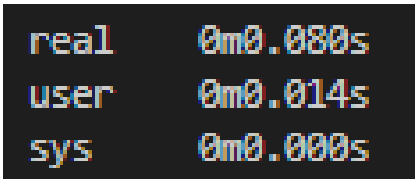

Fig. 54: Terminal showing the time taken for decompression while the first file was tested.

	delay100.csv	delay200.csv	delay1000.csv
Execution Time Compression	0.094 seconds	0.217 seconds	0.238 seconds
Execution Time Decompression	0.080 seconds	0.236 seconds	0.018 seconds

TABLE XXVIII: Table showing the results from Test C3 for each of the example files

The total duration spent on compression for the first file test was therefore 0.174 seconds.

The total duration spent on compression for the second file test was therefore 0.453 seconds.

The total duration spent on compression for the third file test was therefore 0.256 seconds.

As can be seen from the above figures, the compression and decompression processing times were measured separately and then added together to calculate the total time spent on the compression algorithm. The maximum time spent on the compression algorithm occurred when testing the largest file, as this file has the most values that require compression. The time taken in this computation was 0.174 seconds which falls within the relevant specification. Therefore, the ATP relating to test C3 and Specification S4.2 has been met.

Test C4 Results:

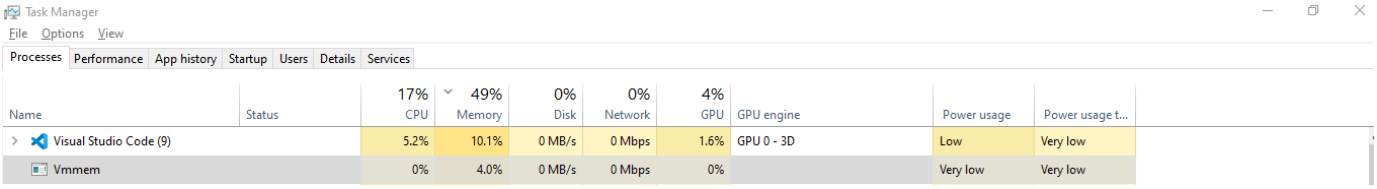

Fig. 55: Terminal showing the RAM in use by the relevant process before the first file's compression was tested.

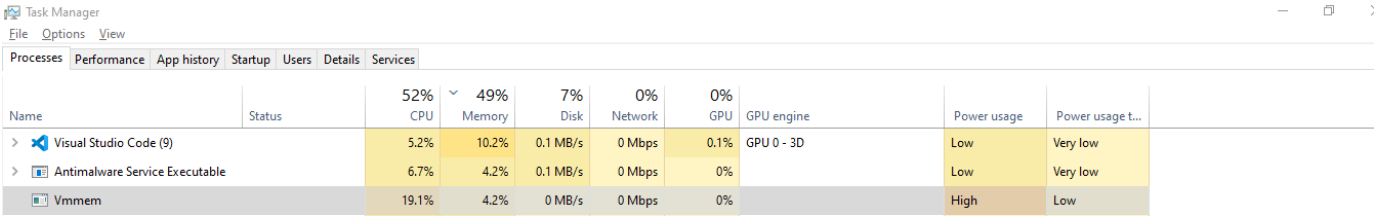

Fig. 56: Terminal showing the RAM in use by the relevant process while the first file's compression was tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Percentage RAM usage	0.2%	0.1%	0.1%

TABLE XXIX: Table displaying the results of Test C4 on each of the example files

As can be seen from the preceding figures, the maximum change in the "Vmmem" process 0.2% of the RAM which is the same as 16MB on a 8GB device. This would be scaled down when running the system on the microcontroller. The RAM on the microcontroller is 192KB, but the data will be transmitted one row at a time and therefore there is enough RAM available to compute the encryption and decryption. The 0.2% of the RAM utilised when running the system does meet the relevant ATP for Specification S5.2.

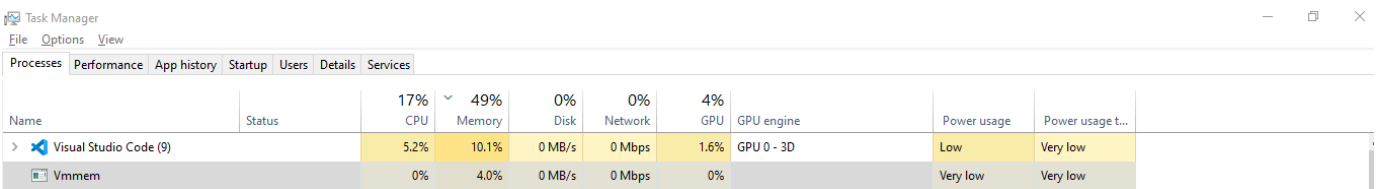

Fig. 57: Terminal showing the RAM in use by the relevant process before the first file's compression was tested.

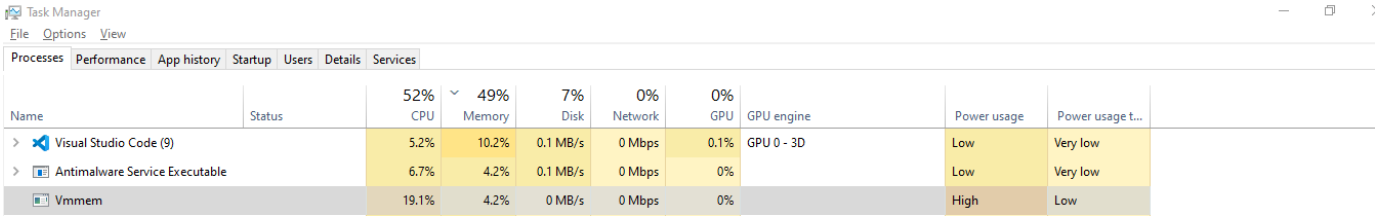

Fig. 58: Terminal showing the RAM in use by the relevant process while the first file's compression was tested.

	delay100.csv	delay200.csv	delay1000.csv
Percentage RAM usage	0.91%	0.46%	0.46%

TABLE XXX: Table displaying the results of Test C5 on each of the example files

As can be seen from the figures above, the maximum change in the "Vmmem" process (which is responsible for running the computation from the terminal) is 0.91% of the total available CPU of the device used. The CPU available of the computer is however different than the one used for this experiment. This would therefore be scaled down when running the system on the microcontroller but the percentage usage would be the same. The ATPs were revised after relating them to the Specifications. An ATP for CPU usage has been added. The 0.91% of the CPU percentage utilised when running the system does meet the relevant ATP.

4) Results for the Encryption subsystem

Test E1 Results:

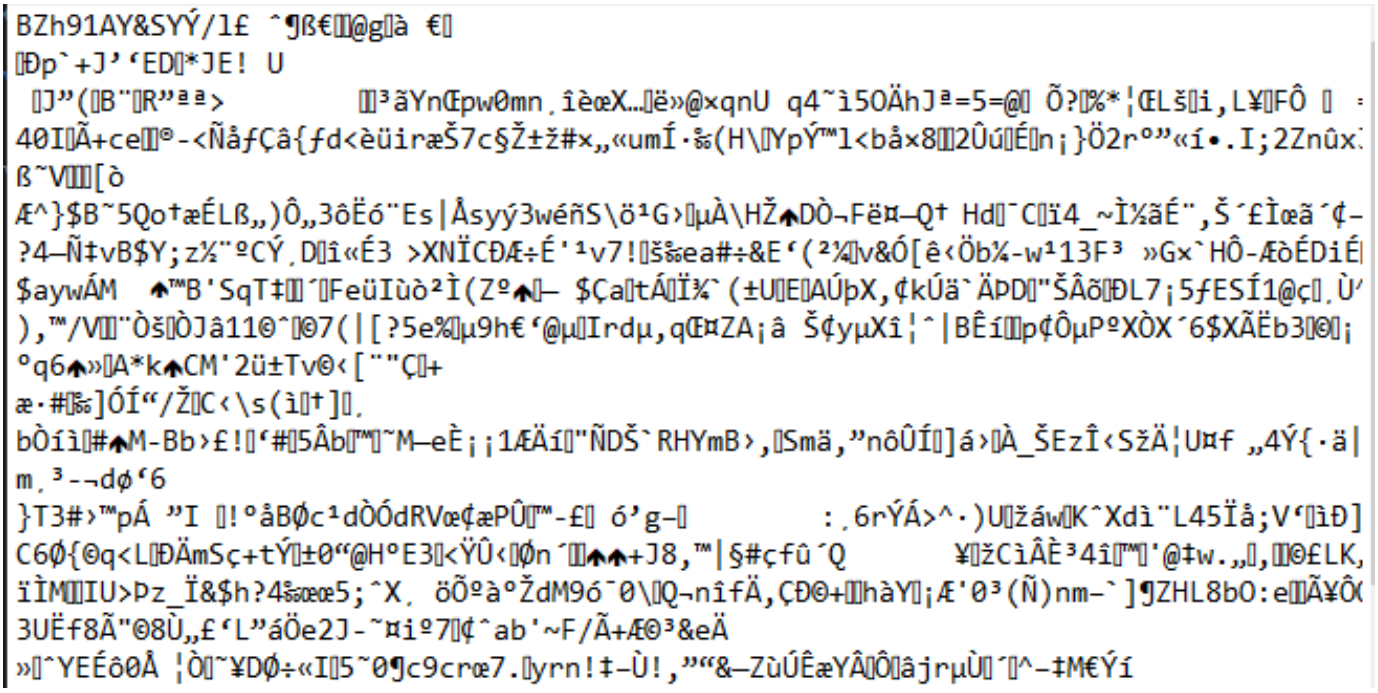

Fig. 59: File showing the encrypted data for the first file tested.

	delay100.csv	delay200.csv	delay1000.csv
Encryption Results	Successful	Successful	Successful

TABLE XXXI: Table showing the results from Test E1 for each of the example files

Table XXXI shows a sample of the encrypted data for each of the three test after the files have been encrypted. All three of the encrypted files were tested against alternate encryption keys and these keys were unsuccessful in trying to decrypt the data which ensures the safety of the data. The ATP which indicates a working encryption key ensures the data is safe as it cannot be read and accessed without the key. This ATP is related to Specification S1.3 and is ATP has been met.

Test E2 Results:

Fig. 60: Terminal showing file comparison before and after encryption for the first file tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Python Script Results	100%	100%	100%

TABLE XXXII: Table displaying the results of Test E2 on each of the example files

The above table shows that the integrity and the contents of the each of the files in their respective tests is not compromised and that the files before and after encryption and decryption are 100% the same and therefore, the ATP - for Specification S2.3 - relating to the encryption integrity is met.

Test E3 Results:

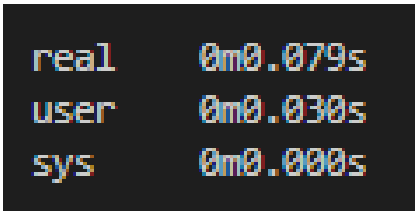

Fig. 61: Terminal showing the time taken for encryption while the first file was tested.

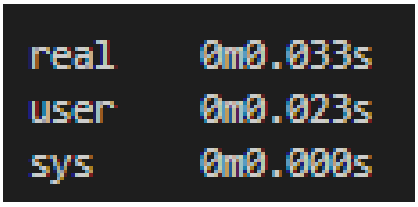

Fig. 62: Terminal showing the time taken for decryption while the first file was tested.

The total duration spent on encryption for the first file test was therefore 0.112 seconds.

The total duration spent on encryption for the second file test was therefore 0.157 seconds.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Execution Time for Encryption	0.079 seconds	0.086 seconds	0.082 seconds
Execution Time for Decryption	0.033 seconds	0.071 seconds	0.093 seconds

TABLE XXXIII: Table showing the results from Test E3 for each of the example files

The total duration spent on encryption for the first file test was therefore 0.175 seconds.

The duration spent on encryption and decryption, as can be seen in the figures above, for each of the three tests totaled a value that was always lower than the specified 20 seconds required computation time. As the maximum computation time was smaller than 0.175 seconds for both encryption and decryption combined, the ATP is therefore met. This ATP relates to Specification S3.3.

Test E4 Results:

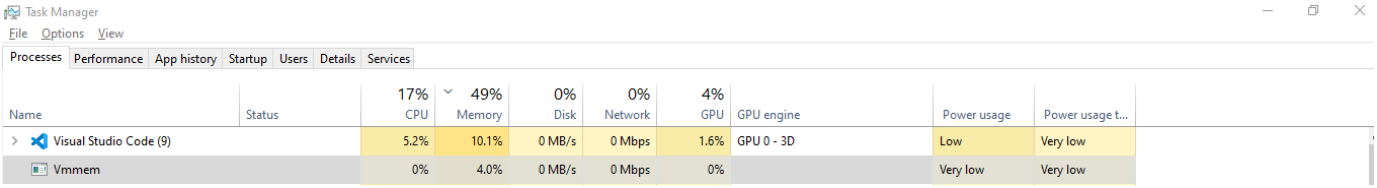

Fig. 63: Terminal showing the RAM in use by the relevant process before the first file's encryption was tested.

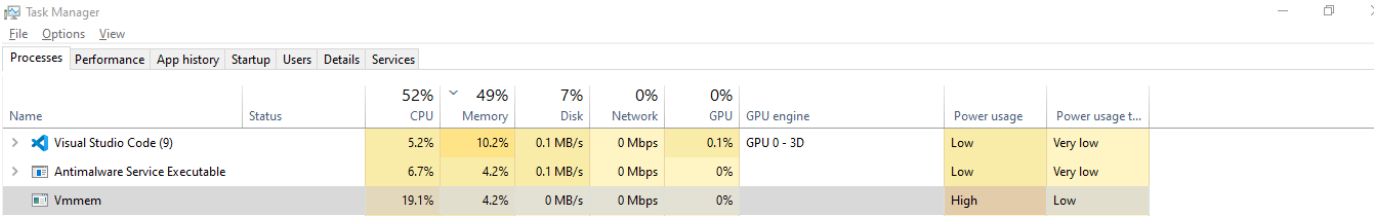

Fig. 64: Terminal showing the RAM in use by the relevant process while the first file's encryption was tested.

	delay100.csv	**delay200.csv**	**delay1000.csv**
Percentage RAM usage	0.2%	0.1%	0.1%

TABLE XXXIV: Table displaying the results of Test E4 on each of the example files

As can be seen from the preceding figures, the maximum change in the "Vmmem" process 0.2% of the RAM which is the same as 16MB on a 8GB device. This would be scaled down when running the system on the microcontroller. The RAM on the microcontroller is 192KB, but the data will be transmitted one row at a time and therefore there is enough RAM available to compute the encryption and decryption. The 0.2% of the RAM utilised when running the system does meet the relevant ATP for Specification S3.4..

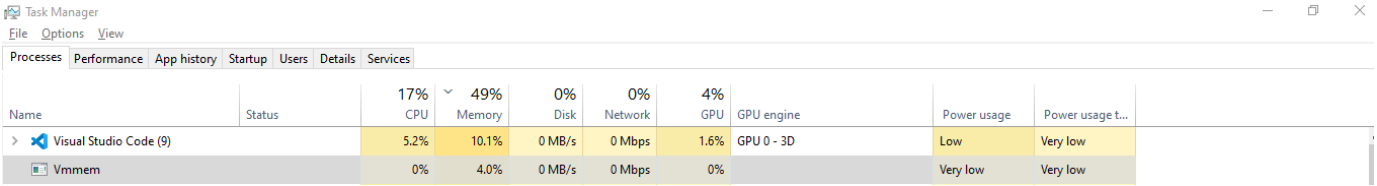

Fig. 65: Terminal showing the RAM in use by the relevant process before the first file's encryption was tested.

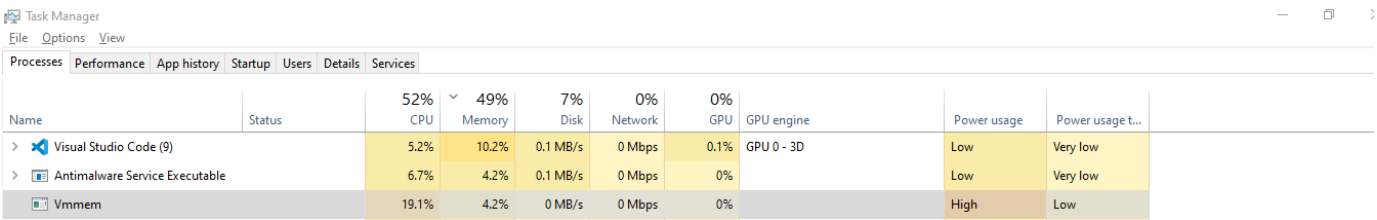

Fig. 66: Terminal showing the RAM in use by the relevant process while the first file's encryption was tested.

	delay100.csv	delay200.csv	delay1000.csv
Percentage RAM usage	0.91%	0.46%	0.46%

TABLE XXXV: Table displaying the results of Test E5 on each of the example files

As can be seen from the figures above, the maximum change in the ”Vmmem” process (which is responsible for running the computation from the terminal) is 0.91% of the total available CPU of the device used. The CPU available of the computer is however different than the one used for this experiment. This would therefore be scaled down when running the system on the microcontroller but the percentage usage would be the same. The ATPs were revised after relating them to the Specifications. An ATP for CPU usage has been added. The 0.91% of the CPU percentage utilised when running the system does meet the relevant ATP.

VI. Consolidation of ATPs and Future Plan

The following ATPs (Tables: XXXVI, XXXVII and XXXVIII) are from the requirement analysis. The tables needed to be updated as they were some additional ATPs that were established as well as the testing of each ATPs being redone using the sample data straight from the IMU sensor and STM. These new experiments and testing was already included in the earlier sections. The CPU usage was tested for the overall system, the compression subsystem and encryption subsystem.

A. Original ATPs from the requirement analysis:

1) Previous Overall Functionality ATPs

Figures of Merit	Experiment	Acceptable Performance Definition	ATPs Met/Unmet
Connection between the IMU sensor and the STM microcontroller	Test O1 tests whether the file sent by the IMU sensor is provided to the STM correctly with no loss and in the correct format.	Successful transmission between the IMU and the STM Microcontroller.	ATP Met
Correctness of the data	Test O2 compares the final file that is delivered with the original file provided by the IMU sensor.	The data must be the same as the original file and no data must be added/lost or corrupted.	ATP Met
Execution Time	Test O3 measures the time before and after the entire process has been run.	The execution time must be less than 60 seconds.	ATP Met
Memory Usage	Test O4 identifies how much RAM is used for different data sizes for the entire process.	The total RAM usage must be less than 10% of available RAM.	ATP Met

TABLE XXXVI: Acceptance Test specifications for the Overall functionality of the system

2) Previous Compression subsystem ATPs

Figures of Merit	Experiment	Acceptable Performance Definition	ATPs Met/Unmet
Percentage of the Fourier Coefficients of the data	Test *C1* is executed to compare the data in the decompressed file with the original file and will identify any possible differences and how many differences there are.	The lower 25% of the Fourier coefficients of the original file must be present and extractable from the decompressed file.	ATP Met
Correctness of the data	Test *C1* is again used to compare the data in the decompressed file with the original file and identify possible differences.	None of the data must be lost or added during the compression and decompression. The values must be all the same.	ATP Met
Compression Ratio	Test *C2* measures the size of the compressed file and compares it to the size of the original file.	Compression ratio must be equal or greater than 5.	ATP Met
Execution Time	Test *C3* measures the time before and after the compression algorithm runs and determines the time taken for the process.	The execution time must be less than 20 seconds.	ATP Met
Memory usage	Test *C4* identifies how much RAM is used for different data sizes.	The Compression and decompression operations must each use less than 2.5% of the total RAM.	ATP Met

TABLE XXXVII: Acceptance Test procedure for the Compression subsystem

3) Previous Encryption subsystem ATPs

Figures of Merit	Experiment	Acceptable Performance Definition	ATPs Met/Unmet
Security of the data	Test E1 tests whether the encrypted data is accessible and can be cracked.	The encrypted data must be fully secure.	ATP Met
Correctness of the data	Test E2 compares the decrypted file with the original file and identifies possible differences.	The decrypted data must be exactly the same as the original file and no data must be added/lost or corrupted.	ATP Met
Execution Time	Test E3 measures the time before and after the encryption algorithm run and determines the execution time.	The execution time must be less than 10 seconds each for encryption and decryption.	ATP Met
Memory Usage	Test E4 identifies how much RAM is used for different data sizes	The Encryption and Decryption operations must each use less than 2.5% of the total RAM	ATP Met

TABLE XXXVIII: Acceptance Test specifications for the Encryption subsystem

B. New and revised ATPs:

1) Revised Overall Functionality ATPs

Figures of Merit	Experiment	Acceptable Performance Definition	ATPs Met/Unmet
Connection between the IMU sensor and the STM microcontroller	Test O1 tests whether the file sent by the IMU sensor is provided to the STM correctly with no loss and in the correct format	Successful transmission between the IMU and the STM Microcontroller.	ATP Met
Correctness of the data	Test O2 compares the decrypted file with the original file and identifies possible differences.	The data must be the exact same as the original file and no data must be added/lost or corrupted.	ATP Met
Execution Time	Test O3 measures the time before and after the entire process has been run and determines the execution time.	The execution time must be less than 60 seconds	ATP Met
Memory Usage	Test O4 identifies how much RAM is used for different data sizes of the entire process	The total RAM usage must not use more than 10% of the total RAM	ATP Met
CPU Usage	Test O4 identifies how much CPU percentage is used for different data sizes	The entire operation must not use more than 25% of the total CPU available	ATP Met

TABLE XXXIX: Acceptance Test specifications for the overall functionality of the system

2) Revised Compression subsystem ATPs

Figures of Merit	Experiment	Acceptable Performance Definition	ATPs Met/Unmet
Percentage of the Fourier Coefficients of the data	Test C1 is executed to compare the data in the decompressed file with the original file and will identify any possible differences and how much differences there is.	The lower 25% of the Fourier coefficients of the original file must be present and extractable from the decompressed file.	ATP Met
Correctness of the data	Test C1 is again used to compare the data in the decompressed file with the original fileand identify possible differences.	None of the data must be lost or added during the compression and decompression. The values must be all the same.	ATP Met
Compression Ratio	Test C2 measures the size of the compressed file and compares it to the size of the original file. The compression ratio is determined from this test.	Compression ratio must be equal or grater than 5 .	ATP Met
Execution Time	Test C3 measures the time before and after the compression algorithm runs and determines the time taken for the process.	The execution time can be measured using 20 seconds.	ATP Met
Memory usage	Test C4 identifies how much RAM is used for different data sizes.	The entire Compression and decompression operation must not use more than 2.5% of the total RAM.	ATP Met
CPU usage	Test C4 identifies how much CPU is used for different data sizes	The entire Compression and decompression operation must not use more than 5% of the total CPU available.	ATP Met

TABLE XL: Acceptance Test specifications for the Compression subsystem

3) Revised Encryption subsystem ATPs

Figures of Merit	Experiment	Acceptable Performance Definition	ATPs Met/Unmet
Security of the data	Test E1 tests whether if the encrypted data is accessible and can be cracked.	The encrypted data must be fully secure.	ATP Met
Correctness of the data	Test E2 compares the decrypted file with the original file and identifies possible differences.	The decrypted data must be the exact same as the original file and no data must be added/lost or corrupted.	ATP Met
Execution Time	Test E3 measures the time before and after the encryption algorithm run and determines the execution time.	The execution time must be less than 10 seconds each for encryption and decryption	ATP Met
Memory Usage	Test E4 identifies how much RAM is used for different data sizes	The entire Encryption and Decryption operation must not use more than 2.5% of the total RAM	ATP Met
CPU Usage	Test E4 identifies how much CPU percentage is used for different data sizes	The entire Encryption and Decryption operation must not use more than 5% of the total CPU available	ATP Met

TABLE XLI: Acceptance Test specifications for the Encryption subsystem

C. Future plan:

Regarding the future of this project, the next step would be to start testing and implementing the the project into the S.H.A.R.C Buoy. This new set of testing would involve new parameters to take in consideration such as environmental uncertainties. These experiments will test whether the device is able to function adequately during extreme conditions such as very low temperatures. A possible protection would need to be designed to make the device waterproof as it will be exposed to water. Further testing will need to be done to make sure this protection does not obstruct with the device's ability to gather accurate data.

In addition to this the device cannot run without being powered by a computer as of right now. It would be possible to add a new subsystem to the project which will be in charge of powering the device and allow it to run onto the S.H.A.R.C Buoy for a long period of time. This new subsystem will feature a battery, a battery protection, a voltage regulator (to be able to power the STM and sensor adequately) and possibly a reverse polarity protection.

Furthermore, optimization could be brought to the system. The algorithms used could be optimized to reduce memory (RAM) and CPU usages as well as the execution time. The compression ratio could also be improved which would allow for larger files of data to be transmitted more efficiently. The algorithms could also be completely redesigned in order to perfectly suit the specifications and be as fast as possible.

The effects of noise could also be reduced onto the system by using filtering. A moving average filter technique could be implemented in order to remove the random interference of noise. This filtering technique makes use of a low pass filter which will remove the high frequencies of the signal.

Finally, the project was purely based on the use of an STM microcontroller which could be replaced by a raspberry pi. A raspberry pi would allow for more optimized algorithms to be implemented without h=the restrictions brought by the STM. A raspberry pi is also more commonly used and would allow for the final device to be more accessible and easier to use for the client. Furthermore, a raspberry pi has access to network features. This will be very useful for the transmission of data which would be done in a wireless manner. The raspberry pi would be able to connect to a network and transmit the data immediately as it is received without needing the user to connect it to a computer. This would be much more effective especially in the context of the S.H.A.R.C buoy as the device would not need to be retrieved every time data needs to be gathered.

VII. Conclusion

The requirements and specifications of the project were constructed based on the brief and through research conducted on the components available for use. The specifications derived from the requirements allowed for the definition of Acceptable Test Procedures (ATPs) that could be used, along with their relevant experiments, as a measure to determine if the design has been successful in a certain function.

Research was then done to determine the best algorithms to use for both encryption and compression - those which best suit the ATPs that have been set out. The influence that the choice of algorithm has on the design is felt by the Specifications, ATPs and even the Results. The algorithms most suited to the project were ”DEFLATE” for compression and ”AES-256” for encryption. These algorithms were chosen because they combine high speed data processing with low overhead and low resource allocation required. Later in the project, the algorithm that would be used for compression of the data was changed to ”bzip2”. This change was made due to the struggle of interfacing with and making use of ”DEFLATE” and the differences between the two were minimal. The change allowed for the whole processing block to be run with the use of one command.

An integral part of the project was understanding the bottlenecks that come with data transmission and the use of processes which take different lengths of time to compute. The resources available on the STM microcontroller make it difficult to determine the optimal time for transmitting and receiving data across the UART connection between the microcontroller and the computer. The use of C as the programming language to code the algorithms for processing the data as well as reading from the IMU sensor resulted in a longer period spent on each element due to the lack of familiarity with the language compared to alternatives.

Once the algorithms had been selected, it was imperative to first test them on simulated data sets. This was done to ensure the ATPs were met for each of files and not just a single file. The compression ratio and rate of transmission resulted were factors accounted for in the ATPs that would severely affect the running of the project and they needed to be measured ensure the project would perform as was required. The simulated data chosen for this testing process was formatted in the same manner as was required from the sensor readings, which allowed for the testing of the processing block to include the integrity of the data as well as the formatting. The FFTs of the data were plotted to analysis if at least the 25% of the Fourier coefficients were present in both the input and output files.

The IMU sensor used for the reading of the values was tested to ensure the functionality performed as required for the data reading, and the formatting of the data read from the sensor allowed for it to be transmitted in the format specified within the code running on the STM microcontroller. The tests ensure the data was being produced and that it was centered around the correct values in order to produce accurate measurements. The file format used to read from the sensor was a ”.csv” as it was the most effective in sorting the formatting before the filtering required on the computer side could be implemented. All the other files used within the program were ”.txt” files which were able to hold the same data in the correct format, just without the reading and writing overheads that come with a ”.csv” file.

The overall system was tested along with each of the subsystems to ensure that the output data was in fact the correct data and the ATPs relating to this were all met. The tests were output from the processing blocks was the same as the data being read from the sensor to ensure the full functioning of the system. A future planning was then drawn up to account for the next steps that can be done in order to improve the project in different ways.

This was all done using the Agile Design Method to ensure frequent client involvement in order to develop a final product to their liking. Weekly meetings were had where minutes were recorded to ensure that anything meaningful mentioned in the meetings was noted and acted upon at a later date.

References

[1] Fourier transforms: Image compression, part 1 - YouTube. [Online]. Available: https://www.youtube.com/watch?v=g8h-d1IJfGk

[2] Fourier compression. [Online]. Available: https://reproducibility.org/RSF/book/geo391/hw3/paper_html/node3.html

[3] S. R. Kodituwakku and U. S. Amarasinghe, "COMPARISON OF LOSSLESS DATA COMPRESSION ALGORITHMS FOR TEXT DATA," vol. 1, no. 4, p. 10.

[4] DevPal, "LZW compression algorithm explained | an introduction to data compression." [Online]. Available: https://www.youtube.com/watch?v=KJBZyPPTwo0

[5] LZW (lempel–ziv–welch) compression technique - GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/

[6] Time-series compression algorithms, explained. [Online]. Available: https://www.timescale.com/blog/time-series-compression-algorithms-explained/

[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," vol. 3, no. 2. [Online]. Available: https://symbiosisonlinepublishing.com/computer-science-technology/computerscience-information-technology32.php

[8] A. Bhattacharya. Comparison of various encryption algorithms and techniques for securing data. [Online]. Available: https://www.encryptionconsulting.com/comparison-of-various-encryption-algorithms-and-techniques-for-securing-data/

VIII. Appendices

A. Links to the IMU sensors datasheets:

ICM20948 datasheet: https://invensense.tdk.com/wp-content/uploads/2021/10/DS-000189-ICM-20948-v1.5.pdf

ICM20649 datasheet: https://invensense.tdk.com/wp-content/uploads/2021/07/DS-000192-ICM-20649-v1.1.pdf

B. Appendix A: Simulated Data (Initial Analysis)

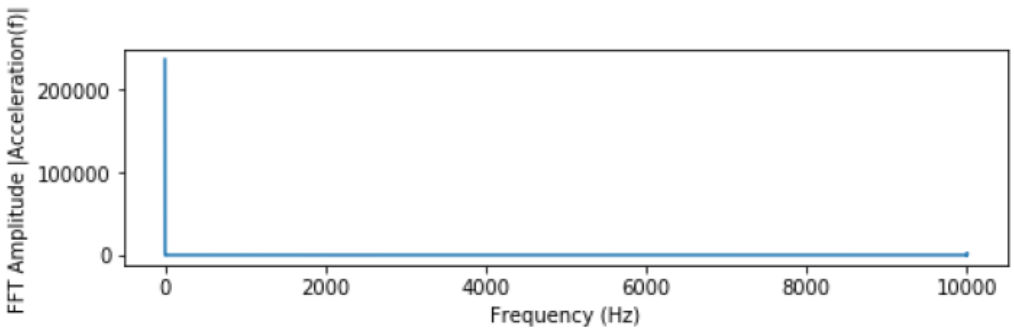

Fig. 67: Graph showing the FFTs of the accelerometer values of the IMU sensor straight from the file ”2.csv” retrieved from the STM

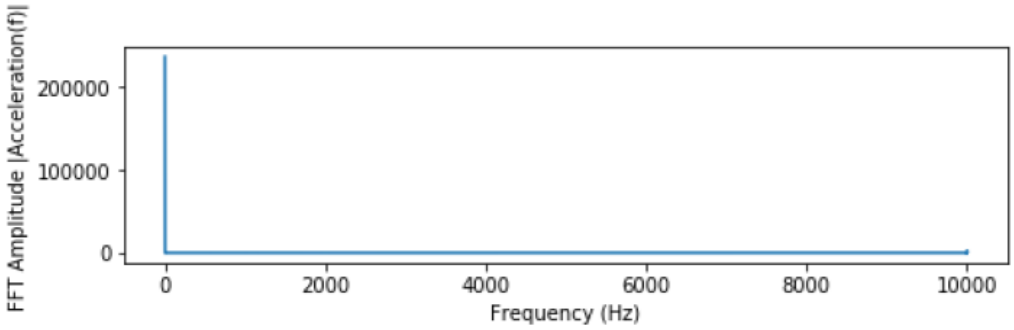

Fig. 68: Graph showing the FFTs of the accelerometer values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The accelerometer values have not been changed by the processing system, meaning the transmission was successful.

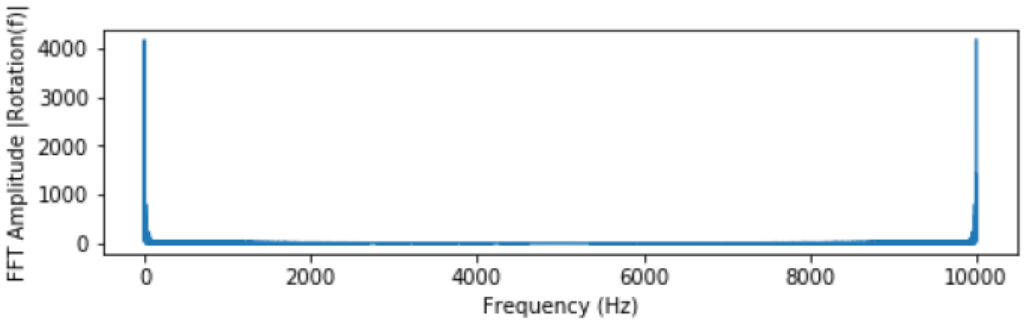

Fig. 69: Graph showing the FFTs of the gyroscope values of the IMU sensor straight from the file ”2.csv” retrieved from the STM

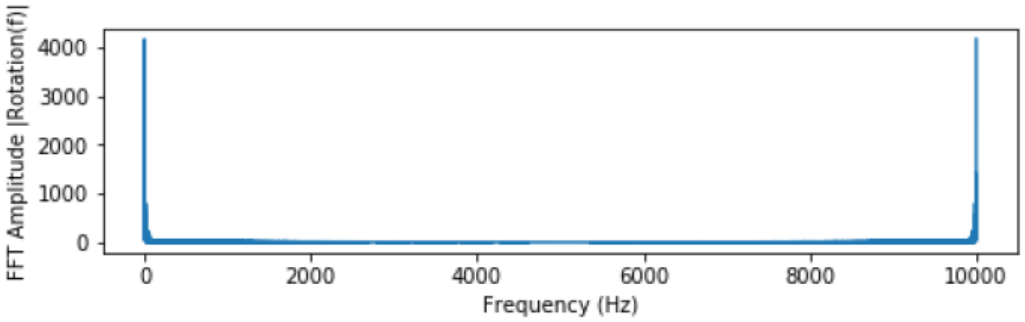

Fig. 70: Graph showing the FFTs of the gyroscope values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The gyroscope data values have not changed.

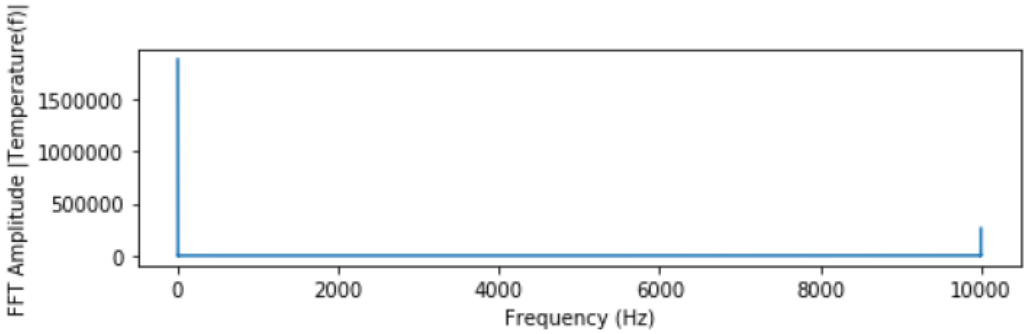

Fig. 71: Graph showing the FFTs of the temperature values of the IMU sensor straight from the file ”2.csv” retrieved from the STM

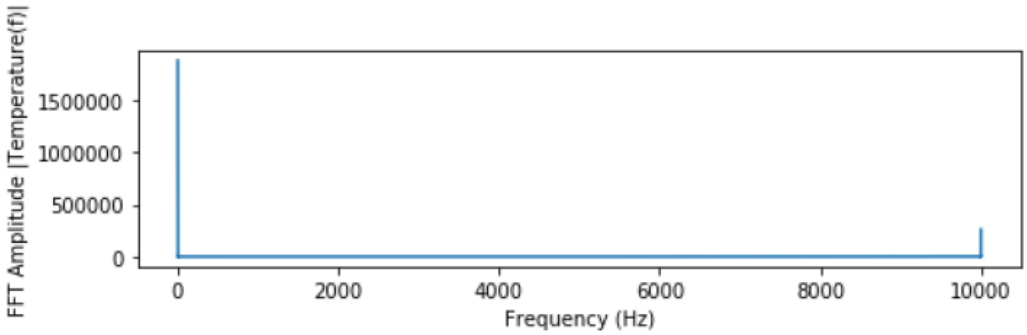

Fig. 72: Graph showing the FFTs of the temperature values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The temperature data values have not changed.

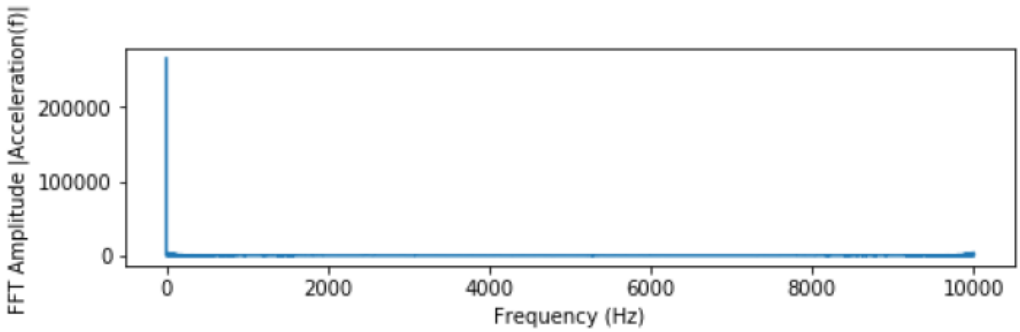

Fig. 73: Graph showing the FFTs of the accelerometer values of the IMU sensor straight from the file ”3.csv” retrieved from the STM

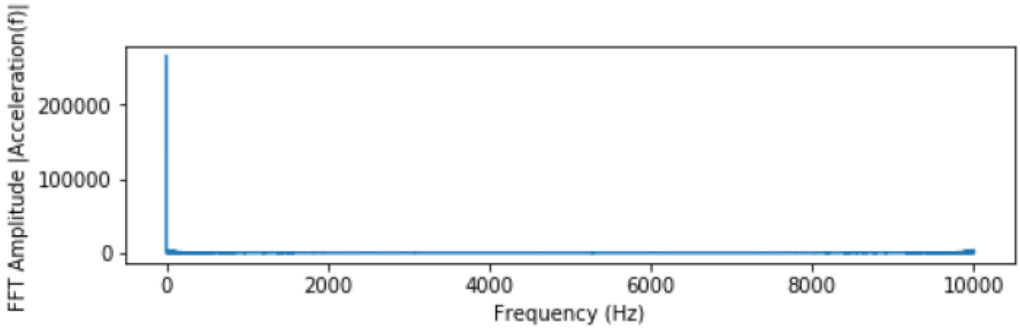

Fig. 74: Graph showing the FFTs of the accelerometer values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The accelerometer values have not been changed by the processing system, meaning the transmission was successful.

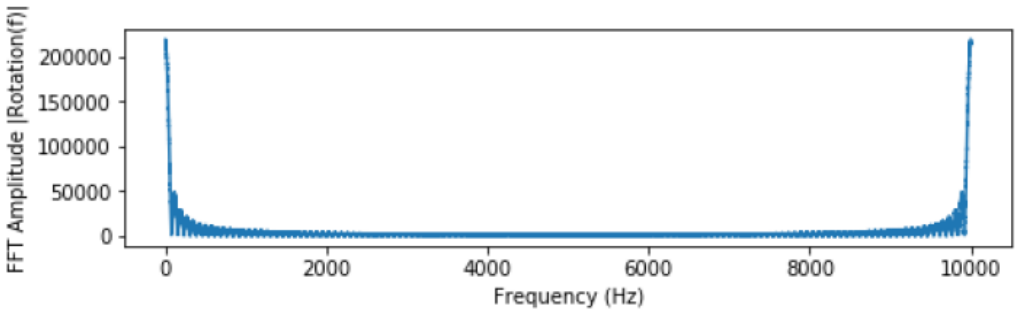

Fig. 75: Graph showing the FFTs of the gyroscope values of the IMU sensor straight from the file ”3.csv” retrieved from the STM

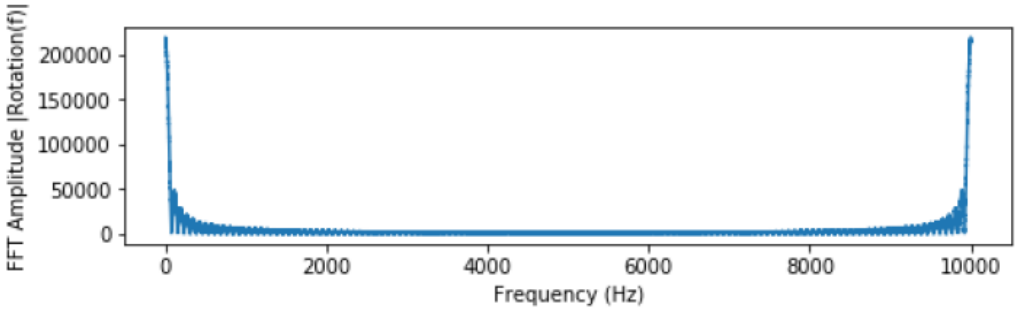

Fig. 76: Graph showing the FFTs of the gyroscope values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The gyroscope data values have not changed.

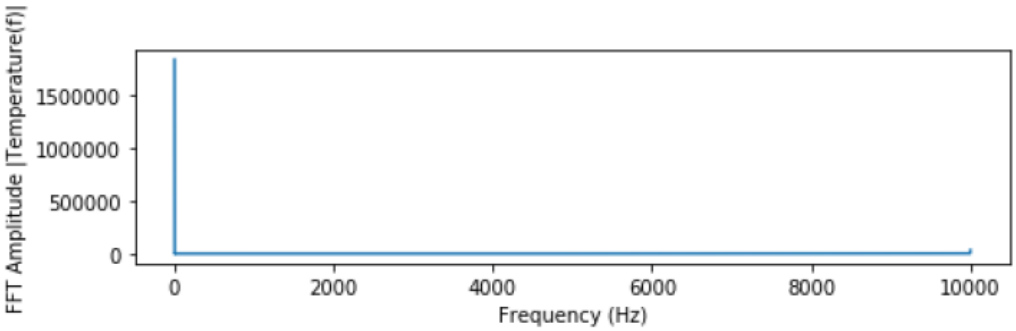

Fig. 77: Graph showing the FFTs of the temperature values of the IMU sensor straight from the file ”3.csv” retrieved from the STM

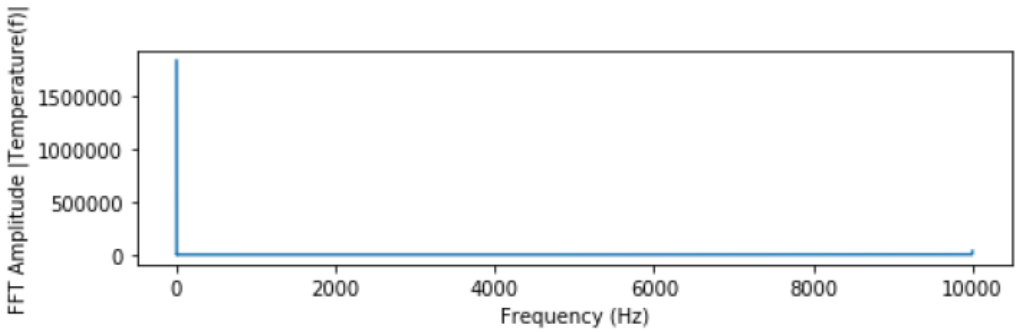

Fig. 78: Graph showing the FFTs of the temperature values after the data has been going through the overall system and been compressed, encrypted, decrypted and decompressed.

As it can be seen from both of these figures, the graph using the data before it has gone through the overall system and the graph using the data after it has gone through the system are very similar. This shows that no data has been lost or altered. The FFTs look the same and the peaks/troughs are all the same. The temperature data values have not changed.

The following are the results for files 2.csv and 3.csv for each of the tests conducted.

Test O2 Results:

```
murphy@ThinkpadT450:/mnt/c/Users/ethan/OneDrive - University of Cape Town/2022/2022S/EEE3097S/data-transfer-microcontroller/Testing$ python3 CompareFiles.py
Enter the first filename: 2.csv
Enter the second filename: 2.txt
-----------------------------------
Comparing files
 > 2.csv
 < 2.txt
-----------------------------------
The files are 100% the same.
```

Fig. 79: Terminal showing file comparison for the second file tested.

```
murphy@ThinkpadT450:/mnt/c/Users/ethan/OneDrive - University of Cape Town/2022/2022S/EEE3097S/data-transfer-microcontroller/Testing$ python3 CompareFiles.py
Enter the first filename: 3.csv
Enter the second filename: 3.txt
-----------------------------------
Comparing files
 > 3.csv
 < 3.txt
-----------------------------------
The files are 100% the same.
```

Fig. 80: Terminal showing file comparison for the third file tested.

Test O3 Results:

```
murphy@ThinkpadT450:/mnt/c/Users/ethan/OneDrive - University of Cape Town/2022/2022S/EEE3097S/data-transfer-microcontroller/Testing$ time (bzip2 -vk 2.csv && openssl enc -aes-256-cbc -pbkdf2
 -in 2.csv.bz2 -out 2out.txt.bz2 -pass pass:ethan && rm 2.csv.bz2  && openssl enc -d -aes-256-cbc -pbkdf2 -in 2out.txt.bz2 -out 2.txt.bz2 -pass pass:ethan && rm 2out.txt.bz2  && bzip2 -d 2.t
xt.bz2)
  2.csv:   13.008:1,  0.615 bits/byte, 92.31% saved, 19827842 in, 1524321 out.

real    0m9.109s
user    0m4.273s
sys     0m0.220s
```

Fig. 81: Terminal simulation time for the second file tested.

```
murphy@ThinkpadT450:/mnt/c/Users/ethan/OneDrive - University of Cape Town/2022/2022S/EEE3097S/data-transfer-microcontroller/Testing$ time (bzip2 -vk 3.csv && openssl enc -aes-256-cbc -pbkdf2
 -in 3.csv.bz2 -out 3out.txt.bz2 -pass pass:ethan && rm 3.csv.bz2  && openssl enc -d -aes-256-cbc -pbkdf2 -in 3out.txt.bz2 -out 3.txt.bz2 -pass pass:ethan && rm 3out.txt.bz2  && bzip2 -d 3.t
xt.bz2)
  3.csv:    6.766:1,  1.182 bits/byte, 85.22% saved, 14332601 in, 2118413 out.

real    0m7.800s
user    0m3.125s
sys     0m0.179s
```

Fig. 82: Terminal simulation time for the third file tested.

Test O4 Results:

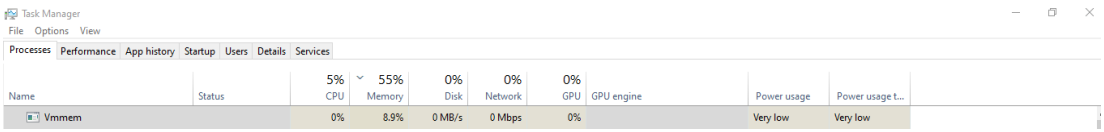

Fig. 83: Terminal showing the RAM in use by the relevant process before the second file was tested.

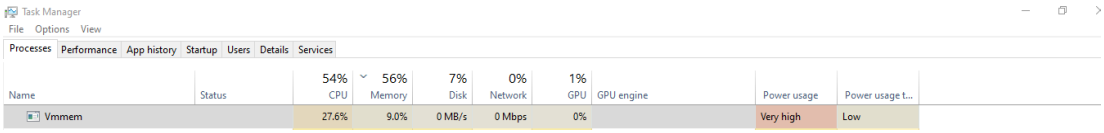

Fig. 84: Terminal showing the RAM in use by the relevant process while the second file was tested.

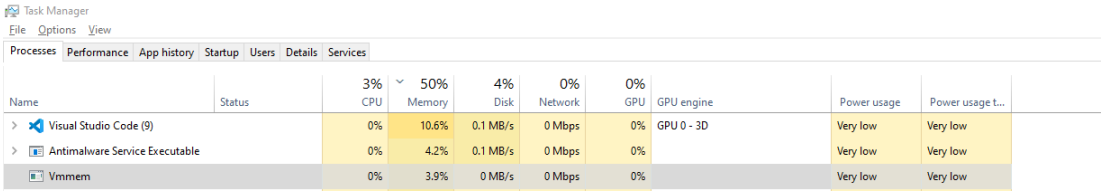

Fig. 85: Terminal showing the RAM in use by the relevant process before the third file was tested.

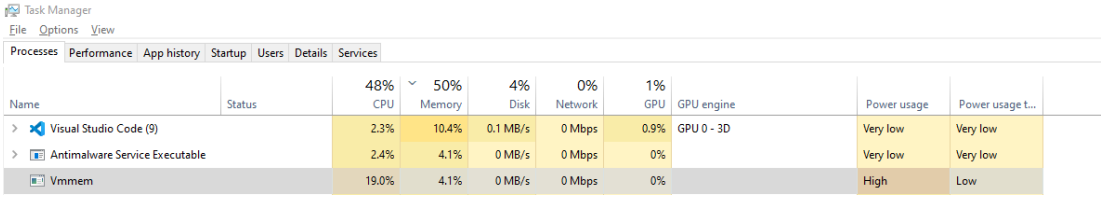

Fig. 86: Terminal showing the RAM in use by the relevant process while the third file was tested.

Test C1 Results:

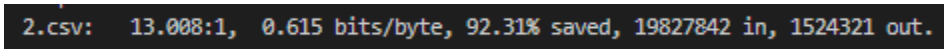

Fig. 87: Terminal showing percentage of data compressed for the second file tested.

Fig. 88: Terminal showing percentage of data compressed for the third file tested.

Test C2 Results:

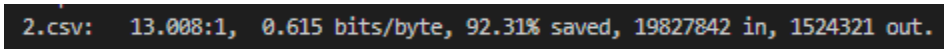

Fig. 89: Terminal showing the compression ratio for the second file tested.

Fig. 90: Terminal showing the compression ratio for the third file tested.

Test C3 Results:

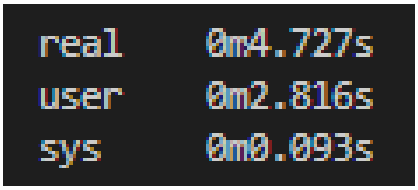

Fig. 91: Terminal showing the time taken for compression while the second file was tested.

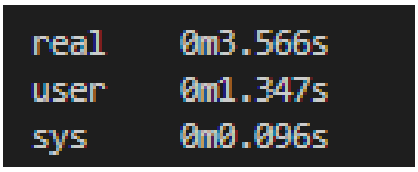

Fig. 92: Terminal showing the time taken for decompression while the second file was tested.

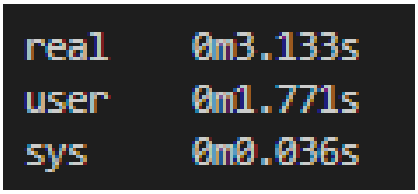

Fig. 93: Terminal showing the time taken for compression while the third file was tested.

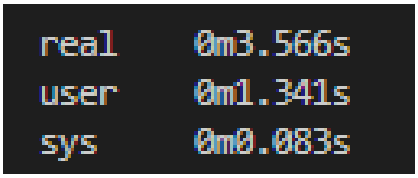

Fig. 94: Terminal showing the time taken for decompression while the third file was tested.

Test C4 Results:

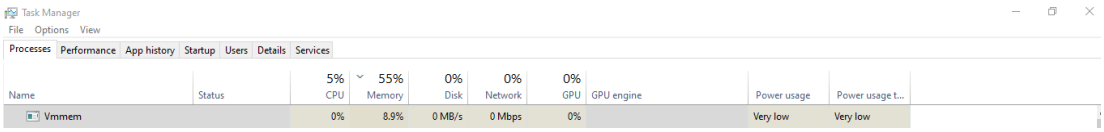

Fig. 95: Terminal showing the RAM in use by the relevant process before the second file's compression was tested.

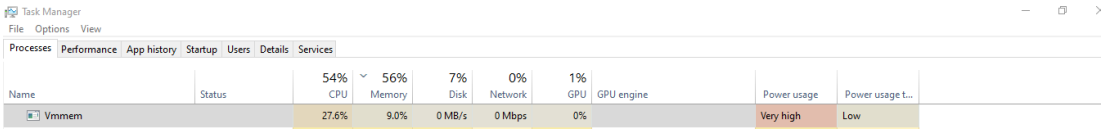

Fig. 96: Terminal showing the RAM in use by the relevant process while the second file's compression was tested.

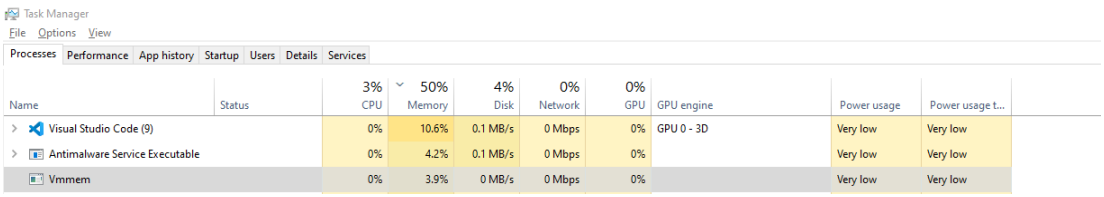

Fig. 97: Terminal showing the RAM in use by the relevant process before the third file's compression was tested.

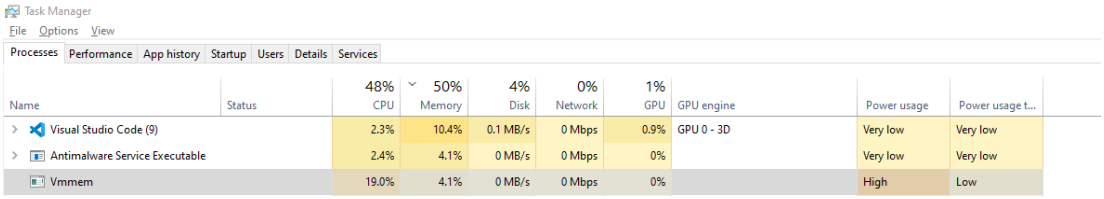

Fig. 98: Terminal showing the RAM in use by the relevant process while the third file's compression was tested.

Test E1 Results:

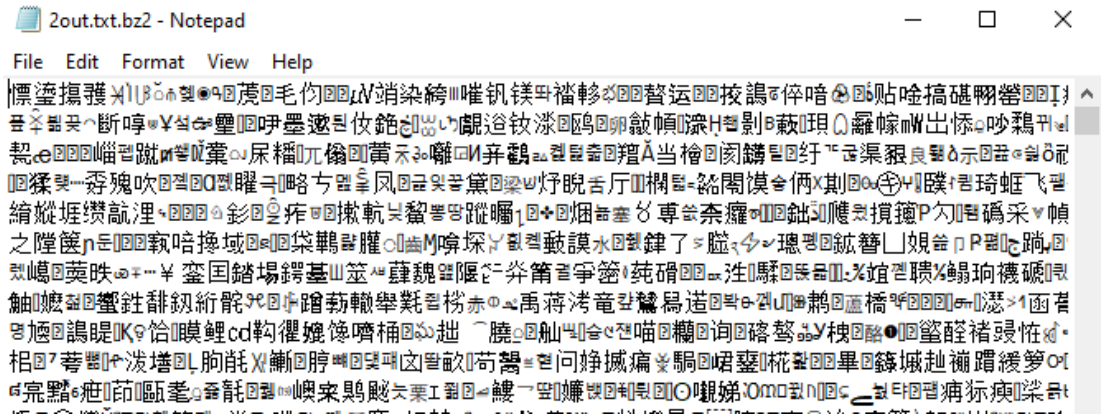

Fig. 99: File showing the encrypted data for the second file tested.

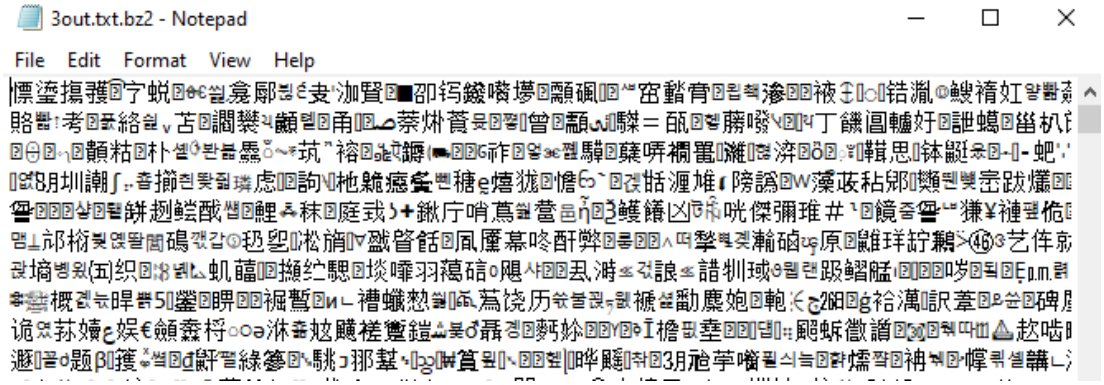

Fig. 100: File showing the encrypted data for the third file tested.

Test E2 Results:

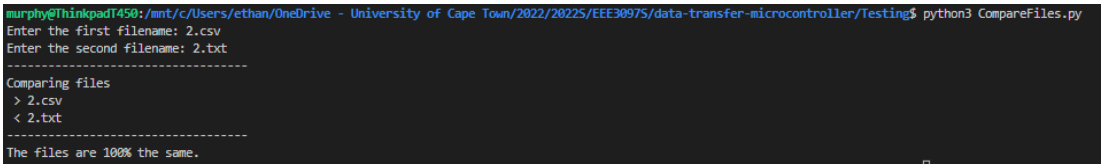

Fig. 101: Terminal showing file comparison before and after encryption for the second file tested.

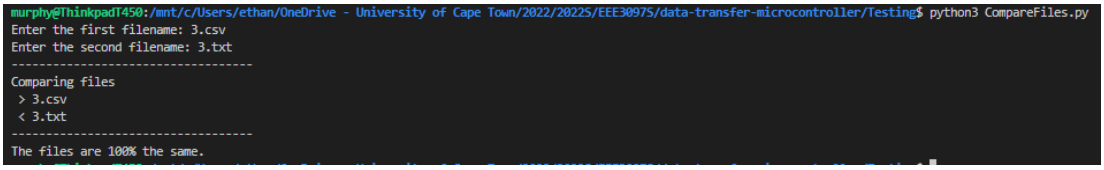

Fig. 102: Terminal showing file comparison before and after encryption for the third file tested.

Test E3 Results:

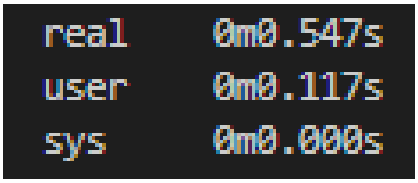

Fig. 103: Terminal showing the time taken for encryption while the second file was tested.

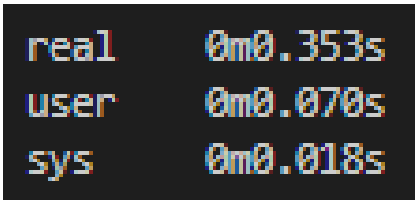

Fig. 104: Terminal showing the time taken for decryption while the second file was tested.

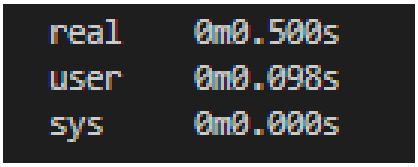

Fig. 105: Terminal showing the time taken for encryption while the third file was tested.

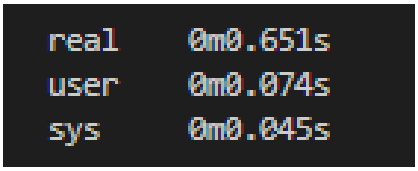

Fig. 106: Terminal showing the time taken for decryption while the third file was tested.

Test E4 Results:

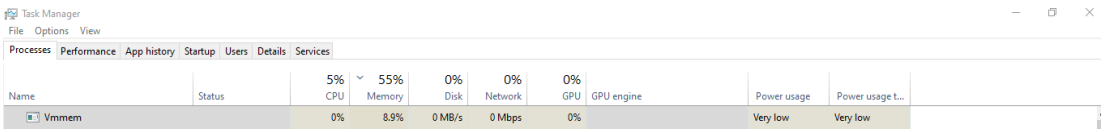

Fig. 107: Terminal showing the RAM in use by the relevant process before the second file's encryption was tested.

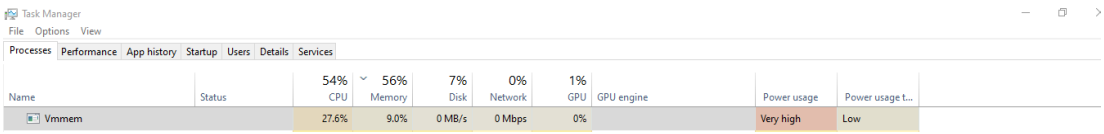

Fig. 108: Terminal showing the RAM in use by the relevant process while the second file's encryption was tested.

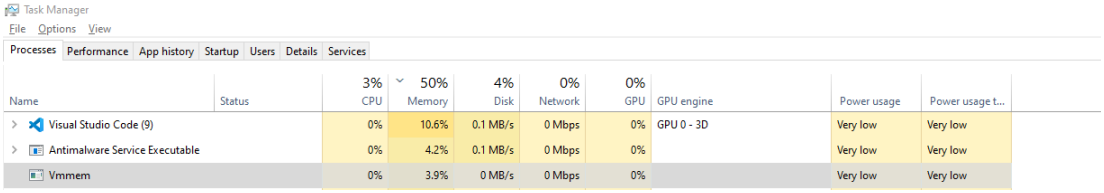

Fig. 109: Terminal showing the RAM in use by the relevant process before the third file's encryption was tested.

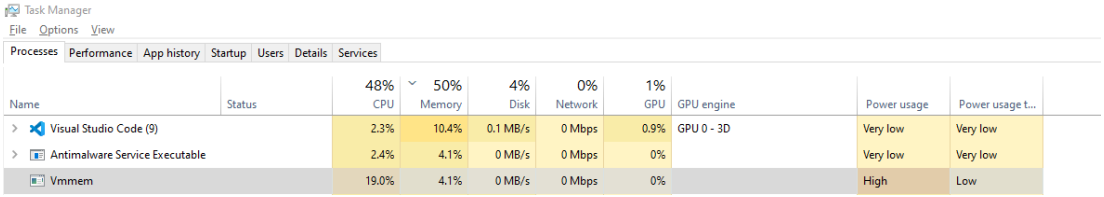

Fig. 110: Terminal showing the RAM in use by the relevant process while the third file's encryption was tested.

C. Appendix B: IMU Data (Initial analysis)

The following are data analysis simulations based on delay200.csv and delay1000.csv.

Initial Analysis of the data contained in the delay200.csv file:

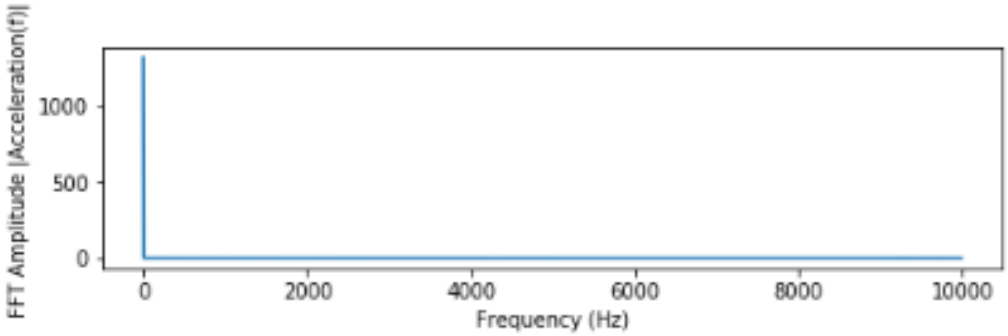

Fig. 111

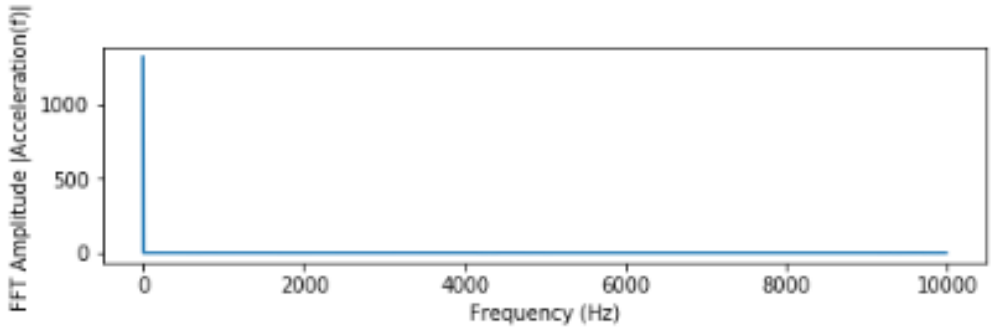

Fig. 112

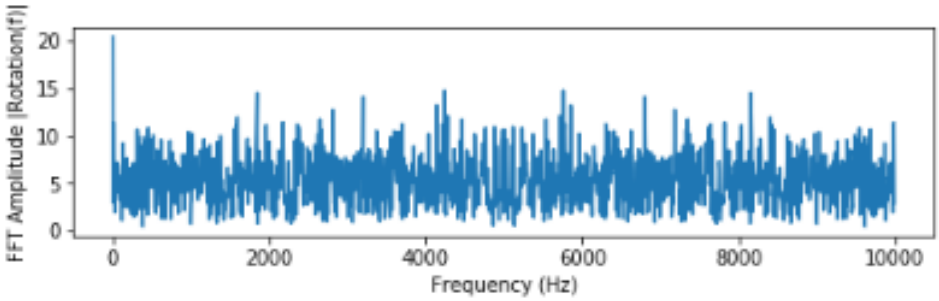

Fig. 113

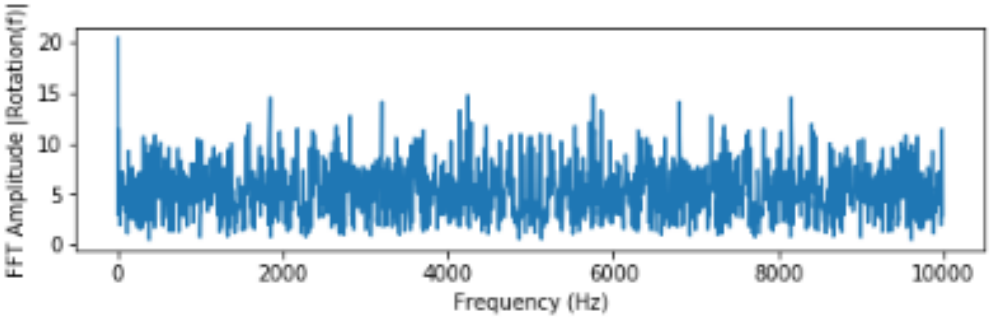

Fig. 114

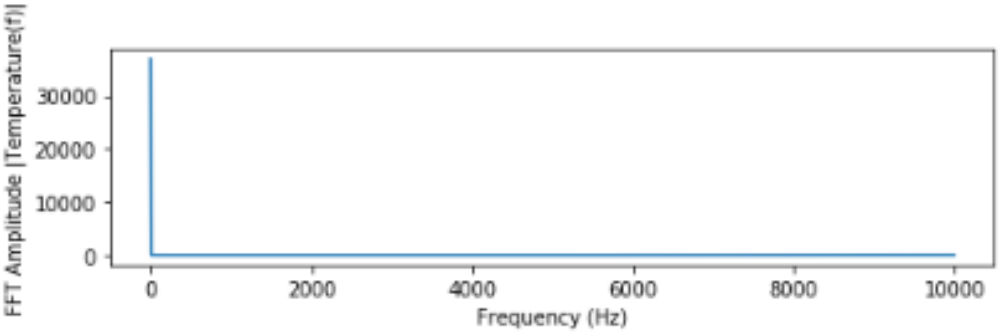

Fig. 115

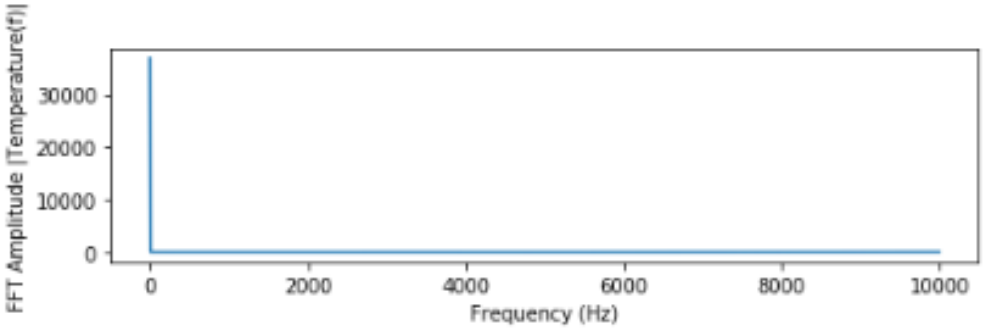

Fig. 116

Initial Analysis of the data contained in the ”delay1000.csv” file:

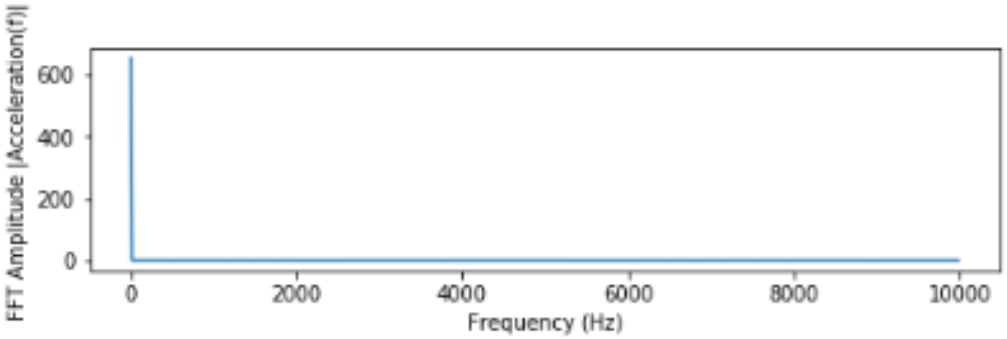

Fig. 117

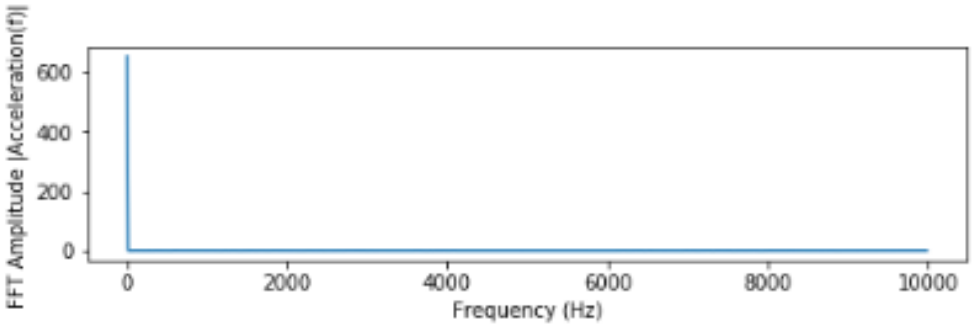

Fig. 118

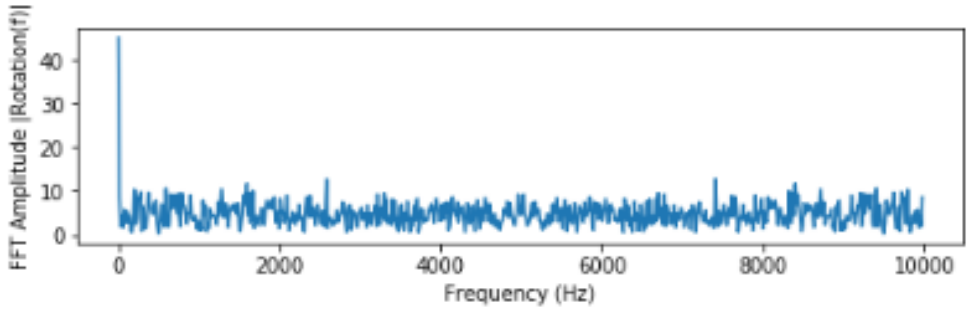

Fig. 119

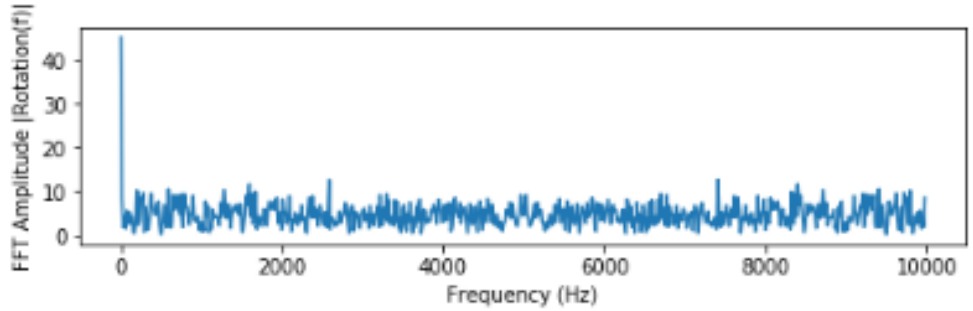

Fig. 120

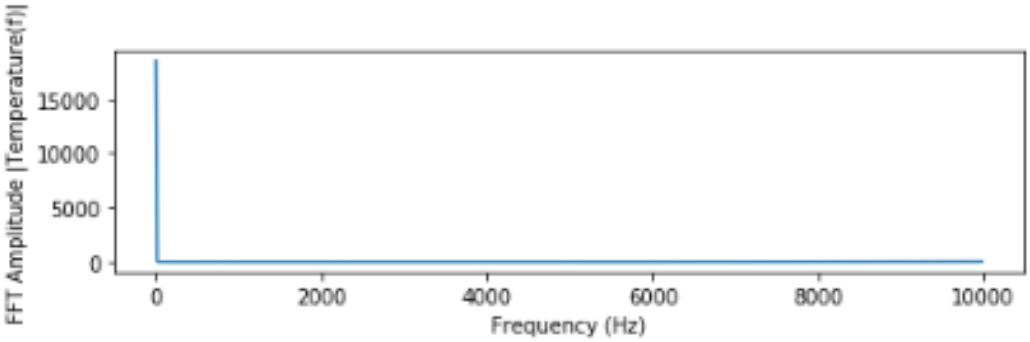

Fig. 121

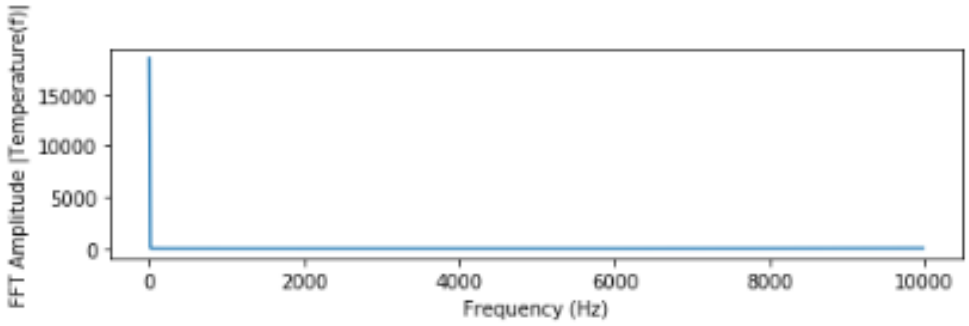

Fig. 122

D. Appendix B: (results of the experimentation for ”delay200.csv” and ”delay1000.csv”)

The following are the results for files delay200.csv and delay1000.csv for each of the tests conducted.

Test O2 Results:

Fig. 123: Terminal showing file comparison for the second file tested.

Fig. 124: Terminal showing file comparison for the third file tested.

Test O3 Results:

Fig. 125: Terminal simulation time for the second file tested.

Fig. 126: Terminal simulation time for the third file tested.

Test O4 Results:

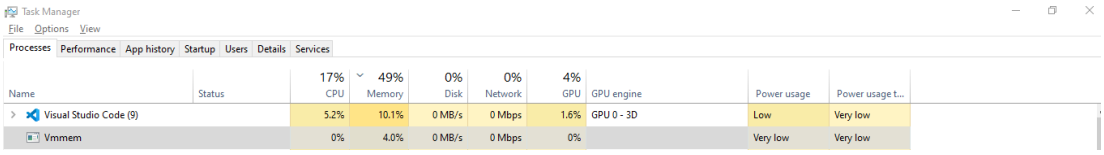

Fig. 127: Terminal showing the RAM in use by the relevant process before the second file was tested.

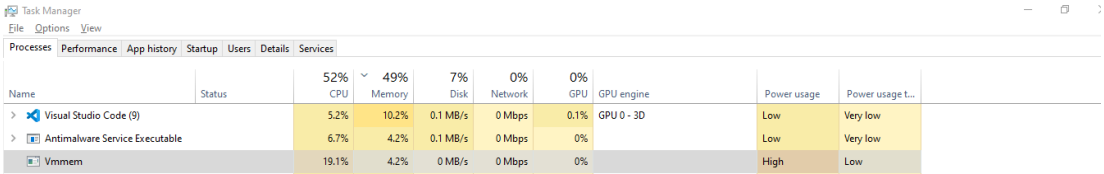

Fig. 128: Terminal showing the RAM in use by the relevant process while the second file was tested.

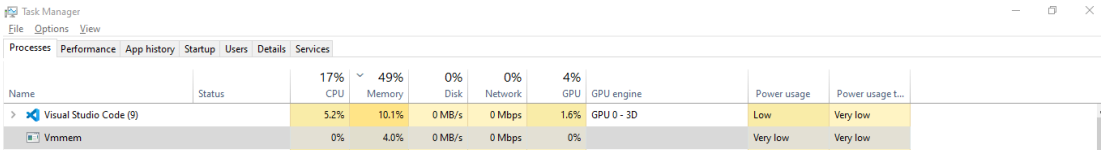

Fig. 129: Terminal showing the RAM in use by the relevant process before the third file was tested.

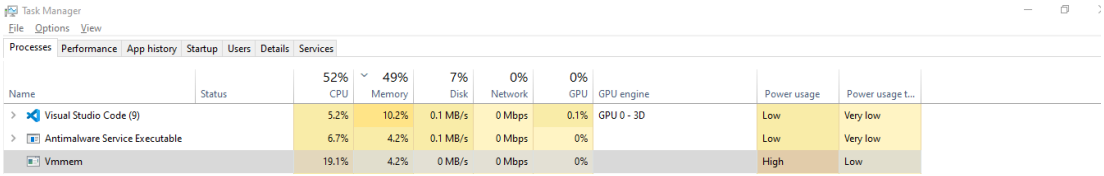

Fig. 130: Terminal showing the RAM in use by the relevant process while the third file was tested.

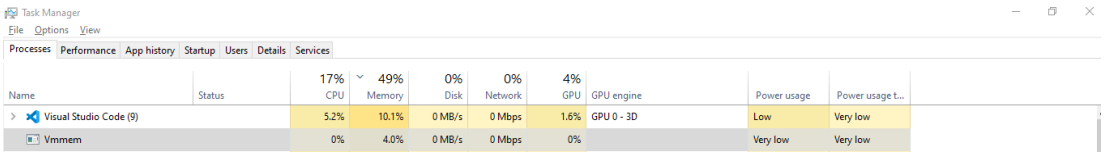

Fig. 131: Terminal showing the CPU in use by the relevant process before the second file was tested.

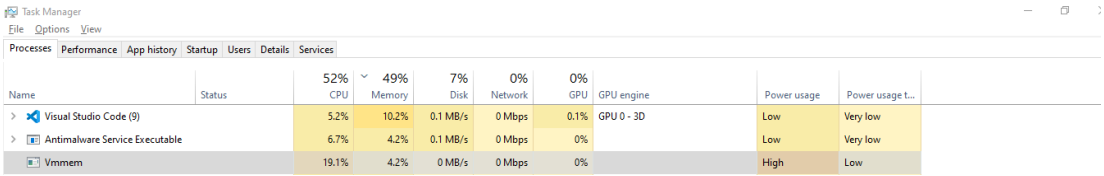

Fig. 132: Terminal showing the CPU in use by the relevant process while the second file was tested.

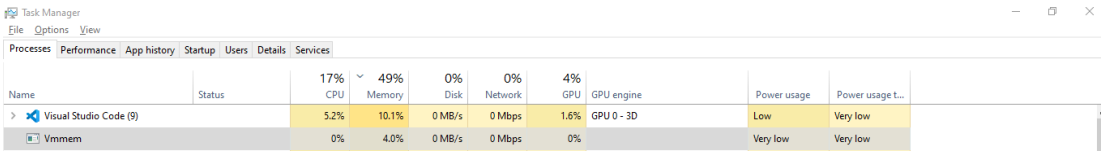

Fig. 133: Terminal showing the CPU in use by the relevant process before the third file was tested.

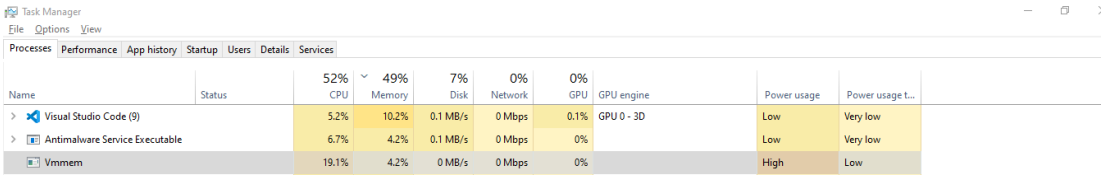

Fig. 134: Terminal showing the CPU in use by the relevant process while the third file was tested.

Test C1 Results:

Fig. 135: Terminal showing percentage of data compressed for the second file tested.

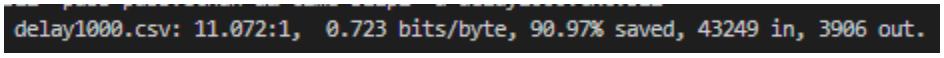

Fig. 136: Terminal showing percentage of data compressed for the third file tested.

Test C2 Results:

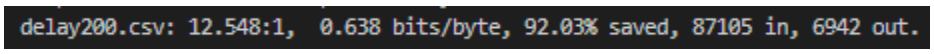

Fig. 137: Terminal showing the compression ratio for the second file tested.

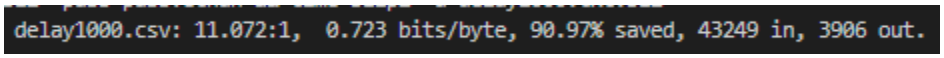

Fig. 138: Terminal showing the compression ratio for the third file tested.

Test C3 Results:

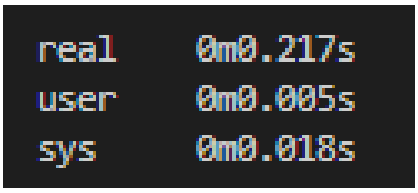

Fig. 139: Terminal showing the time taken for compression while the second file was tested.

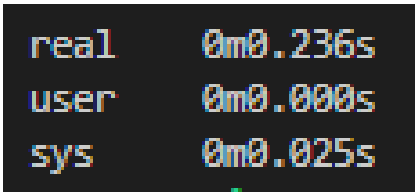

Fig. 140: Terminal showing the time taken for decompression while the second file was tested.

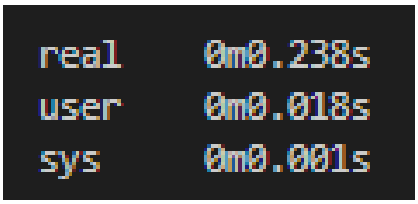

Fig. 141: Terminal showing the time taken for compression while the third file was tested.

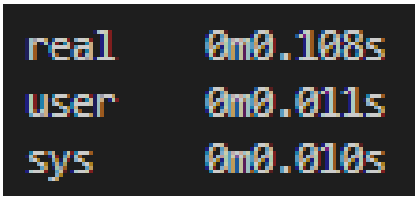

Fig. 142: Terminal showing the time taken for decompression while the third file was tested.

Test C4 Results:

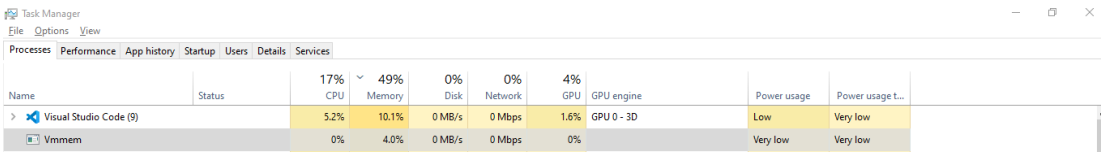

Fig. 143: Terminal showing the RAM in use by the relevant process before the second file's compression was tested.

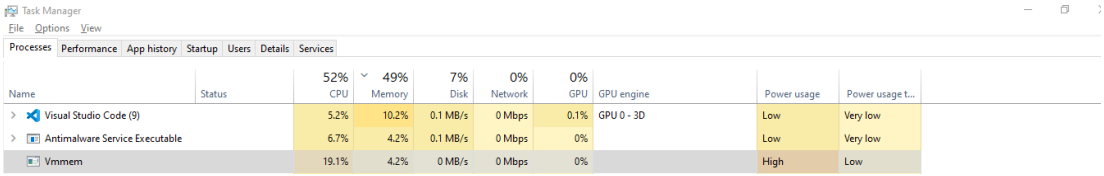

Fig. 144: Terminal showing the RAM in use by the relevant process while the second file's compression was tested.

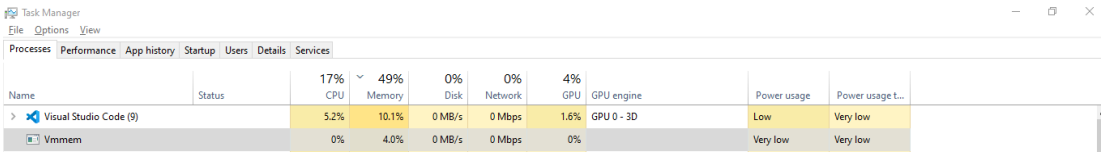

Fig. 145: Terminal showing the RAM in use by the relevant process before the third file's compression was tested.

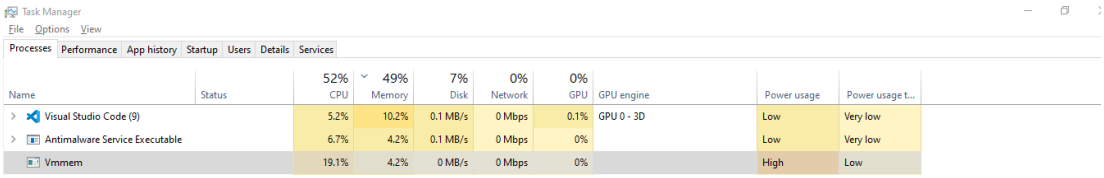

Fig. 146: Terminal showing the RAM in use by the relevant process while the third file's compression was tested.

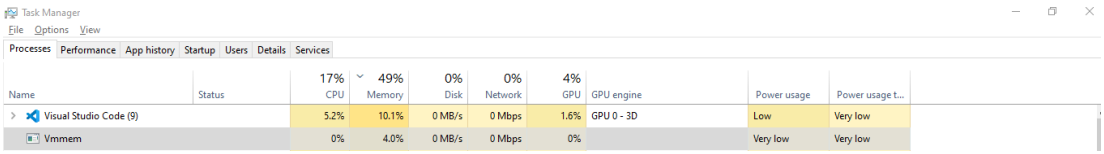

Fig. 147: Terminal showing the CPU in use by the relevant process before the second file's compression was tested.

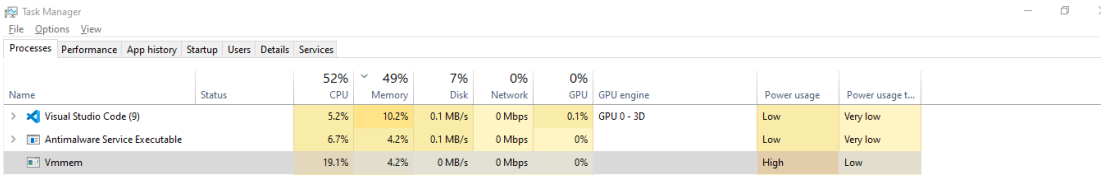

Fig. 148: Terminal showing the CPU in use by the relevant process while the second file's compression was tested.

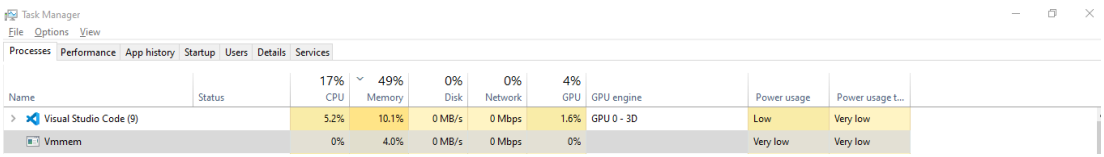

Fig. 149: Terminal showing the CPU in use by the relevant process before the third file's compression was tested.

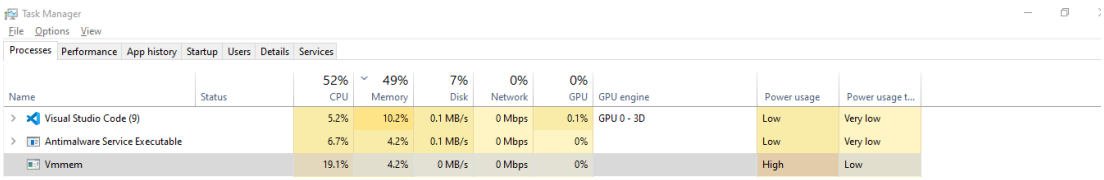

Fig. 150: Terminal showing the CPU in use by the relevant process while the third file's compression was tested.

Test E1 Results:

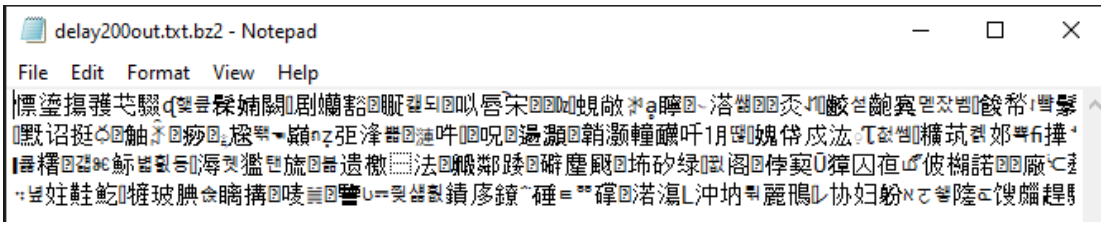

Fig. 151: File showing the encrypted data for the second file tested.

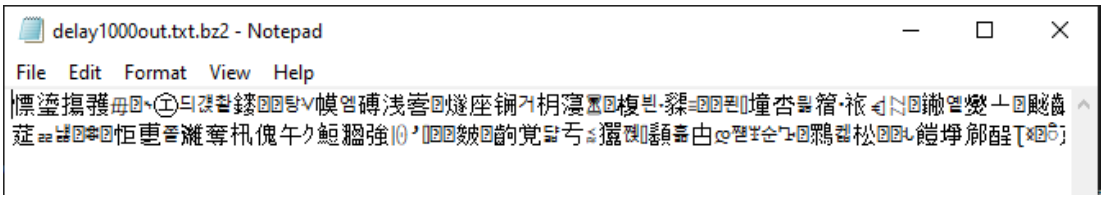

Fig. 152: File showing the encrypted data for the third file tested.

Test E2 Results:

Fig. 153: Terminal showing file comparison before and after encryption for the second file tested.

Fig. 154: Terminal showing file comparison before and after encryption for the third file tested.

Test E3 Results:

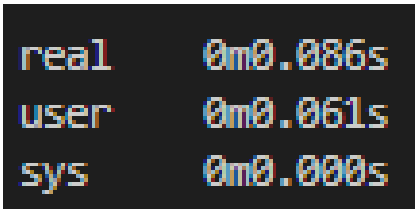

Fig. 155: Terminal showing the time taken for encryption while the second file was tested.

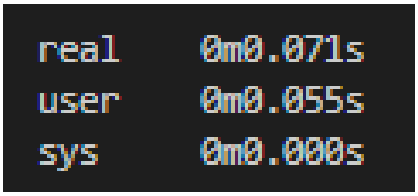

Fig. 156: Terminal showing the time taken for decryption while the second file was tested.

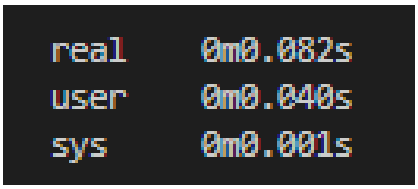

Fig. 157: Terminal showing the time taken for encryption while the third file was tested.

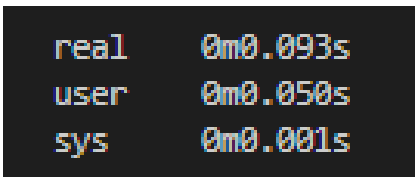

Fig. 158: Terminal showing the time taken for decryption while the third file was tested.

Test E4 Results:

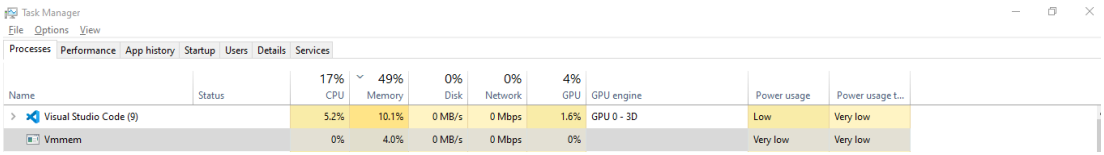

Fig. 159: Terminal showing the RAM in use by the relevant process before the second file's encryption was tested.

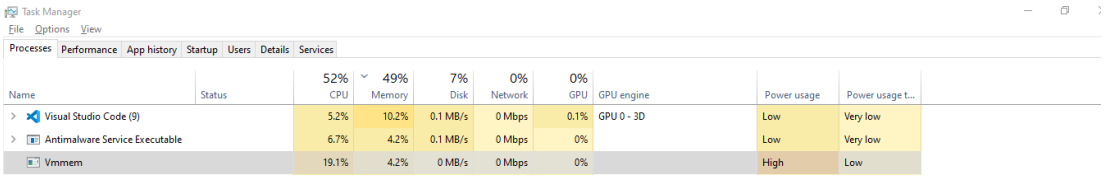

Fig. 160: Terminal showing the RAM in use by the relevant process while the second file's encryption was tested.

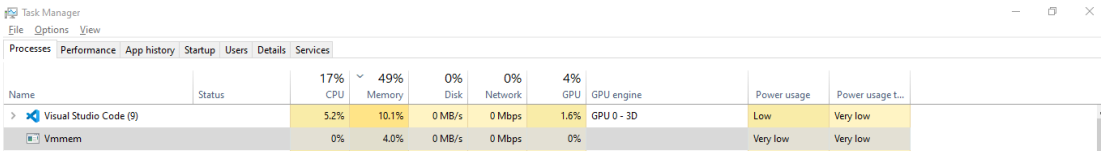

Fig. 161: Terminal showing the RAM in use by the relevant process before the third file's encryption was tested.

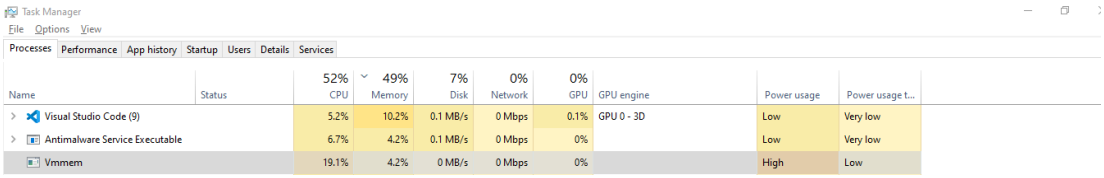

Fig. 162: Terminal showing the RAM in use by the relevant process while the third file's encryption was tested.

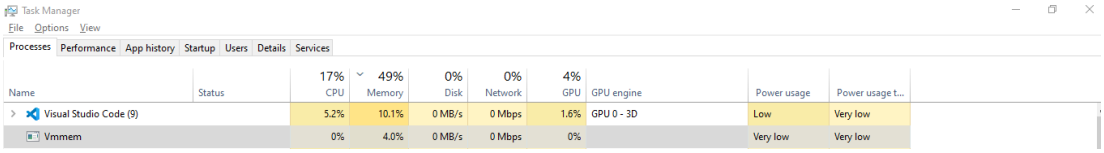

Fig. 163: Terminal showing the CPU in use by the relevant process before the second file's encryption was tested.

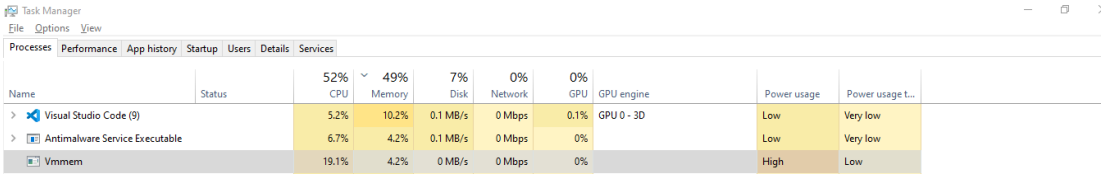

Fig. 164: Terminal showing the CPU in use by the relevant process while the second file's encryption was tested.

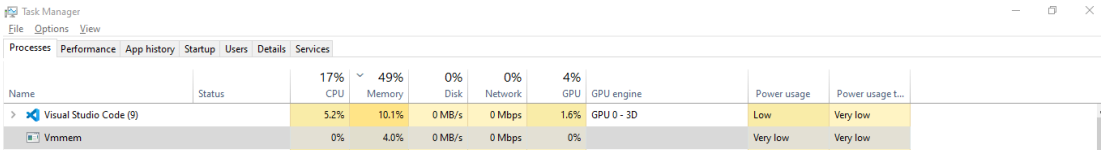

Fig. 165: Terminal showing the CPU in use by the relevant process before the third file's encryption was tested.

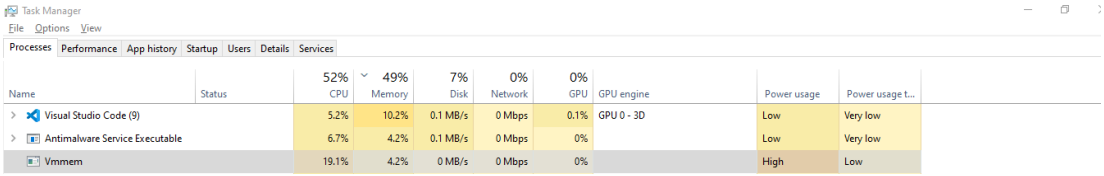

Fig. 166: Terminal showing the CPU in use by the relevant process while the third file's encryption was tested.