EEE3097S PROGRESS REPORT 2

Admin Documents

GitHub Link: [lukhanyoVena808/IMU_DESIGN (github.com)]

Table 1: Contribution

CONTRIBUTORS	SUBSYSTEM
Lukhanyo Vena (VNXLUK001)	Compression
Bina Mukuyamba (MKYBIN001)	Encryption

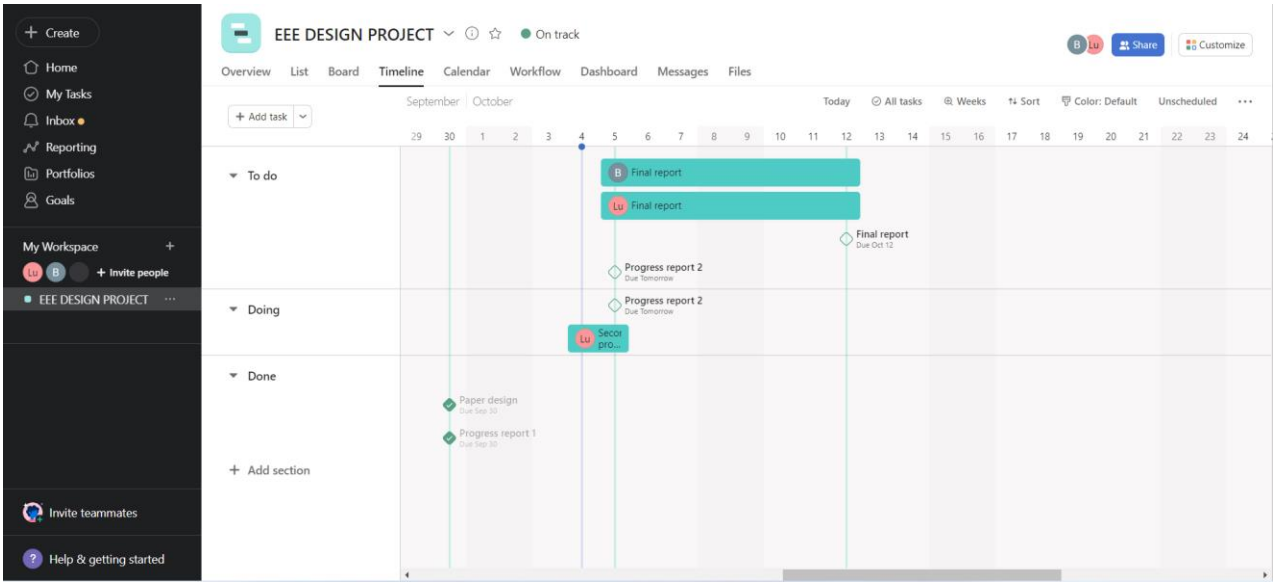

Figure 1: Project timeline

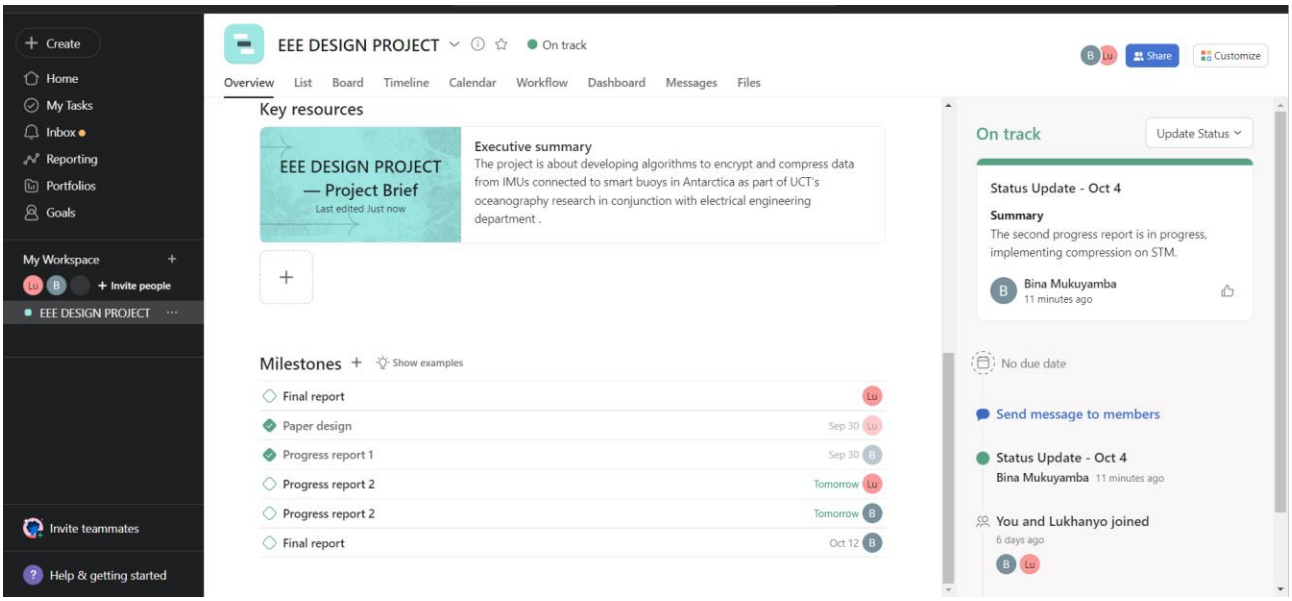

Figure 2: Project overview

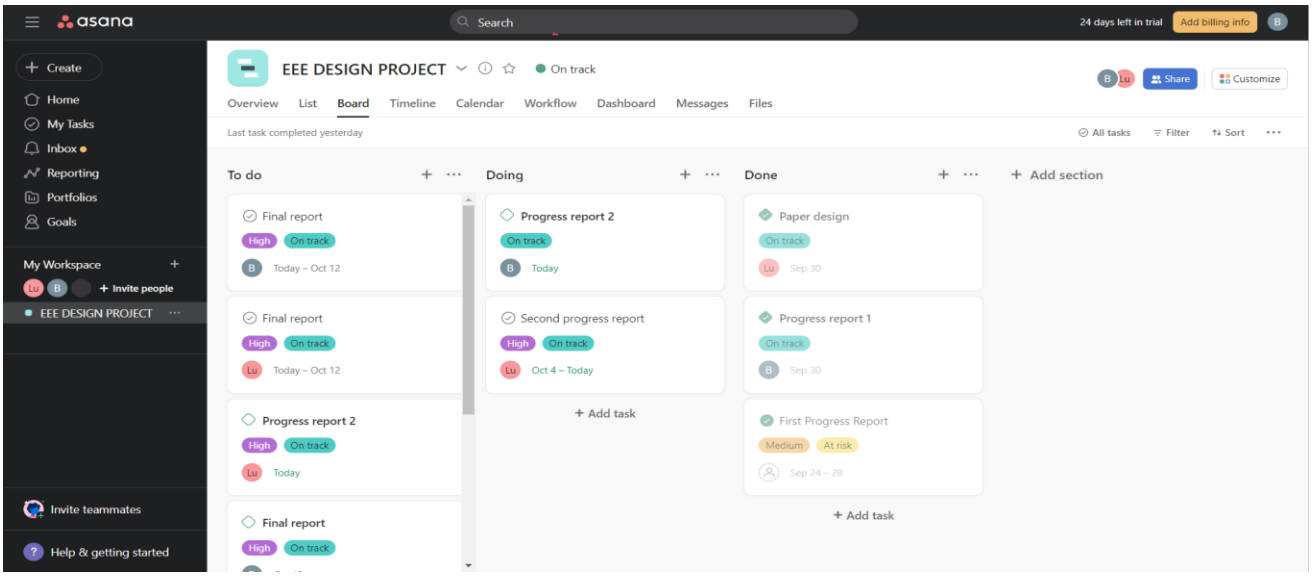

Figure 3: Project board

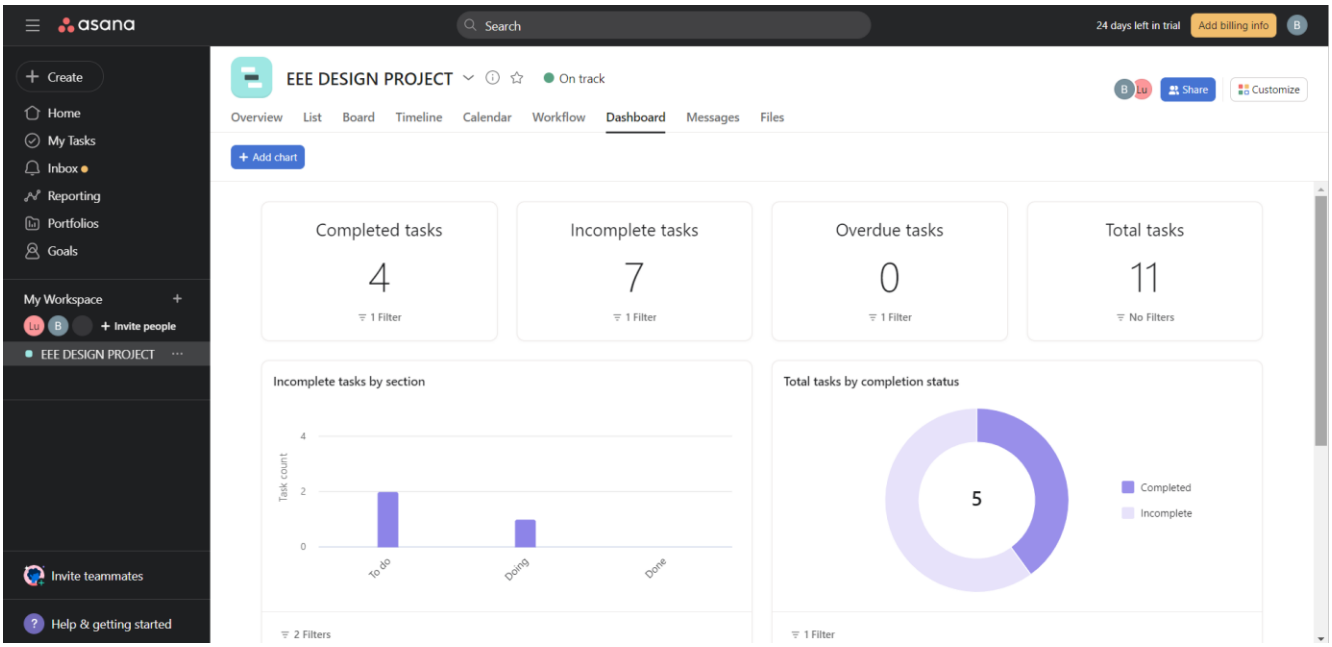

Figure 4: project dashboard

IMU Module

Salient features

Key features of this IMU (20948) that distinguish it from the ICM-20649 is that the ICM-20948 has an onboard 3-axis magnetometer whereas the 20649 does not. However, the 20649 has a larger full-scale range (FSR) for both the accelerometer and gyroscope than the 20948. The gyroscope of the 20649 has dps (degrees per second) ranging from (500 to 4000dps) [1] but the 20948 gyroscope has dps ranging from (250-2000dps) [2]. The accelerometer of the ICM-20649 has gravitational acceleration ranging from (4g to 30g) but the accelerometer of the ICM-20948 has gravitational acceleration ranging from (2g to 16g). Therefore, the ICM-20649 can obtain a wider range of values from these sensors than ICM-20948.

The ICM-20948 requires a greater supply current (3.11mA) than the ICM-20649(1.27mA) due to having an additional sensor (magnetometer).

Testing extrapolation to actual IMU

To extrapolate our testing to the actual IMU (ICM-20649) we will test this ICM-20948 in a controlled environment with environmental conditions similar to the IMU in Antarctica by mimicking the rocking motion of buoys on sea ice and doing the experiments in an air-conditioned room (lower temperature).

Since the actual IMU(ICM-20649) does not have a magnetometer, we will not take readings from the ICM-20948 magnetometer because Antartica is at the poles, therefore the magnetometer data will not provide interesting information and there will be no point adding

code to read from a non-existent sensor when the programs are run on the ICM-20649. So only the accelerometer and gyroscope will be used since they provide measurements of the sea ice dynamics as required by the project requirements.

The same equipment (hardware) will be used (STM, PC, cables) and software (same algorithms will run on STM connected to actual IMU) to extrapolate the experiments to the actual IMU to improve reliability. The experiments on the actual IMU will be the same as described below.

A note about data; Figure 5 below shows all data that could be obtained from the IMU, however due to the limited storage on STM not all will be collected.

Time	Count	Quat W	Quat X	Quat Y	Quat Z	Accel X (g)	Accel Y (g)	Accel Z (g)	Gyro X (dç	Gyro Y (dç	Gyro Z (dç	Yaw	Pitch	Roll	X	Y	Z	Grav X	Grav Y	Grav Z	Temp
0.094	12	0.9999	0.0037	-0.0003	0	-0.0018	-0.0024	4.4889	-0.061	-0.061	0	0	0.0006	0.0073	0	0.0006	-0.0073	0.0006	0.0073	0.9999	30.3771
0.124	15	0.9999	0.0035	-0.0004	0	-0.0042	-0.0084	4.7008	0	-0.061	-0.061	0	0.0007	0.0071	0	0.0007	-0.0071	0.0007	0.0071	0.9999	30.3947
0.134	16	0.9999	0.0035	-0.0005	0	-0.0036	-0.0108	4.8044	-0.061	-0.061	0	0	0.001	0.007	0	0.001	-0.007	0.001	0.007	0.9999	30.3888
0.138	17	0.9999	0.0035	-0.0005	0	-0.0036	-0.0108	4.8044	-0.061	-0.061	0	0	0.001	0.007	0	0.001	-0.007	0.001	0.007	0.9999	30.3888
0.146	18	0.9999	0.0035	-0.0005	0	-0.0036	-0.0108	4.8044	-0.061	-0.061	0	0	0.001	0.007	0	0.001	-0.007	0.001	0.007	0.9999	30.3888
0.15	19	0.9999	0.0035	-0.0005	0	-0.0036	-0.0108	4.8044	-0.061	-0.061	0	0	0.001	0.007	0	0.001	-0.007	0.001	0.007	0.9999	30.3888
0.162	20	0.9999	0.0034	-0.0006	0	-0.003	-0.0102	4.8613	-0.061	-0.061	0	0	0.0012	0.0067	0	0.0012	-0.0067	0.0012	0.0067	0.9999	30.3829
0.166	21	0.9999	0.0034	-0.0006	0	-0.003	-0.0102	4.8613	-0.061	-0.061	0	0	0.0012	0.0067	0	0.0012	-0.0067	0.0012	0.0067	0.9999	30.3829
0.176	22	0.9999	0.0034	-0.0006	0	-0.003	-0.0102	4.8613	-0.061	-0.061	0	0	0.0012	0.0067	0	0.0012	-0.0067	0.0012	0.0067	0.9999	30.3829

Figure 5: EEE3097S 2022 Turntable Example Data 2

The following is a sample of the data we retrieved and plan to compress and encrypt from the ICM-20948:

```
accel X    accel Y    accel Z    gyro X    gyro Y    gyro Z
0.067871, -0.000488, 0.975586, -0.060976, -0.060976, -0.121951
0.000000, -0.118652, 1.000000, -0.060976, -9.939025, 0.000000
0.124512, -0.000488, 1.000000, -0.060976, -0.060976, 15.609756
```

Figure 6: Group 13 data retrieved from ICM-20948

Validation tests

To check that the IMU works, in our main.c file we print the values read from the different sensors to putty via a UART connection.

To validate that the sensors are working we take readings every 1.5 seconds. To see the accelerometer and gyroscope in action we move the IMU connected to the STM horizontally and vertically and observe the readings from the sensors via putty. To observe each axis,the IMU is moved in the direction of each individual axis. I.e., the IMU is moved up and down to observe changes in one axis of both sensors, moved left and right to observe changes in another axis (at constant height), moved forward and backwards to observe changes in the other axis (at constant height).

The validity of the readings from each axis of the sensor can be checked by moving the IMU in opposite directions such that one direction should give negative values for each axis (e.g. backwards, left, or down). Can also check validity by showing that the readings should be close to 0 if there is no movement of the IMU.

Experiment Setup

Overall System

The overall system experiment comprised of these steps in the following order: compression -> encryption -> decryption -> decompression. The sensor data from the accelerometer and gyroscope was read into a buffer (char array) of variable size (depending on how much data we wanted to read) using the STM. The reading of data was observed via UART (on putty). The main.c file containing the compression and encryption algorithms was flashed onto the STM using STM32cube IDE .The data in the buffer was then compressed using a Huffman compression algorithm on the STM. The compressed data was then encrypted using a block cipher ARX (Add rotate XOR) algorithm also on the STM. The compressed and encrypted data was sent to the PC which carried out decryption and decompression using the above 2 algorithms to obtain the original data. A python script was used to automate the execution of this process and compute the execution time of the overall process from the start of compression to the end of decompression.

This procedure was repeated for input data of different sizes by varying the size of the buffer in the main.c file. The python scripts computed the execution time for each size of data being processed. The scripts also computed the compression ratio (size ratio of the compressed data to original data).

To check if the output data retains at least 25% of the Fourier coefficients, the ratio of the decompressed data Fourier transform to the original data Fourier transform was plotted.

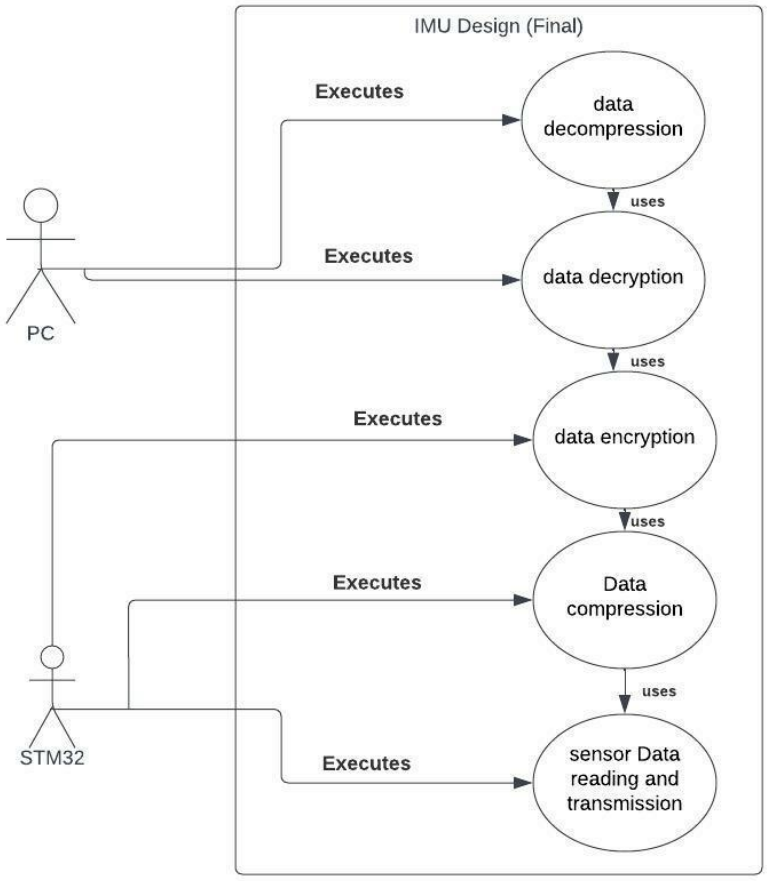

Figure 7: Overall system UML use case diagram

Compression

To test if the compression algorithm works, the execution time for compressing and decompressing data blocks of varied sizes will be computed using the python scripts.

The compression ratios (compressed string array length/original string array length * 100) are also calculated using the scripts (pc side) and using clock functions in main file of the STM. The decompressed data will also be plotted in the time and frequency domain to compare it to the original data. Based on the previous simulation Huffman compression will be used to compress and decompress the IMU data.

The input data to the subsystem is the raw IMU data from the sensors. The compression part of the Huffman algorithm will be done on the STM, the decompression done on the PC. The STM will send the compressed data to the PC for decompression via UART.

To observe compression the buffer size before and after compression is compared. And the compression ratio calculated.

Encryption

To test the subsystem functionality alone, IMU data from STM will be written to a char array (buffer)that will be processed by the encryption algorithm on the STM. To test the subsystem's interfacing with the compression algorithm the input data to the subsystem will be the compressed data from the compression subsystem.

The encryption algorithm used will be the ARX (Add Rotate XOR) block cipher encryption algorithm. The (fenc.c) main file which implements the algorithm was modified to also encrypt strings of data directly instead of reading in from a file since the STM has no filesystem in order to run the encryption using the STM. The encryption portion of the algorithm will be run on the STM while the decryption will be run on the PC.

The expected output of the subsystem is compressed data/pure data with the contents encrypted. Similarly, the expected output of the decryption should be the plaintext(un-encrypted) data.

To test the algorithm efficiency buffers of data (char arrays) of varied sizes are passed into the program and the execution time computed using a python script. The various times obtained for each size of data are compared.

To check if encryption occurred the data is visually inspected before and after encryption to make sure the text is unreadable by printing the buffer contents to putty via UART. Likewise, decryption is validated by checking the original data against the decrypted data.

The security aspect of the algorithm will not be tested (breakability of the encryption) due to time constraints. It was assumed that since the algorithm was lightweight the goal was efficiency over security, so we already know it is not very secure.

Results

Overall System

Using the above methods in the experimental set up the results below were obtained. Figure 8 below shows raw data obtained from the IMU before processing occurs.

NB: Due to time constraints and the fact that our algorithms became corrupted after being sent through to the pc using serial communication, we were unable to decompress and decrypt the data on the pc. The current results were obtained on the algorithms executed on the STM (compression and encryption). We unfortunately also ran out of time to perform the Fourier analysis of the IMU data.

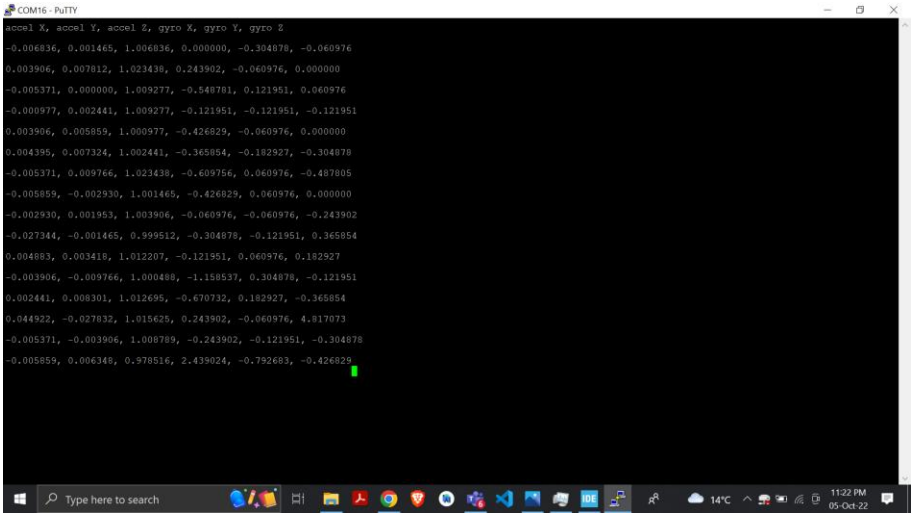

Figure 8: Before Compression

Figure 9 below shows the output of the compression algorithm.

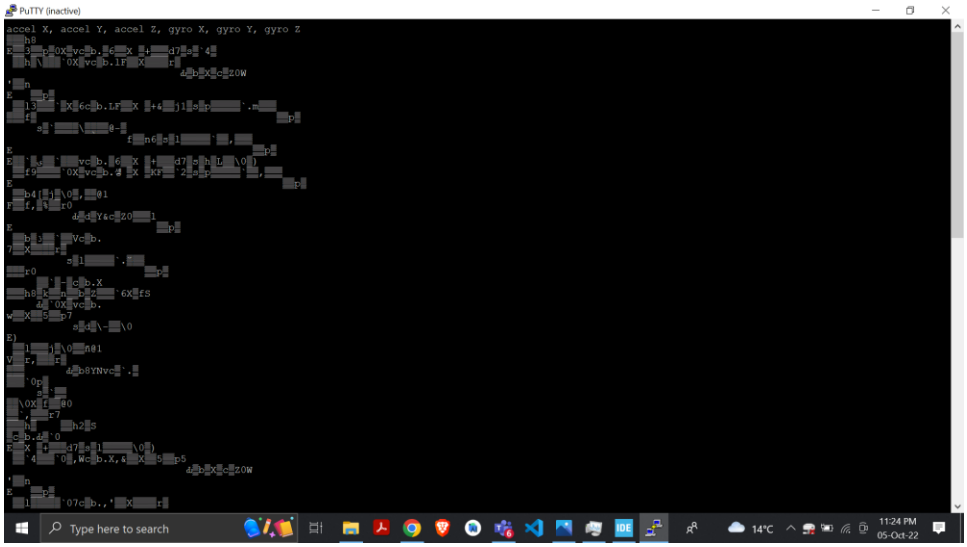

Figure 9: After Compression

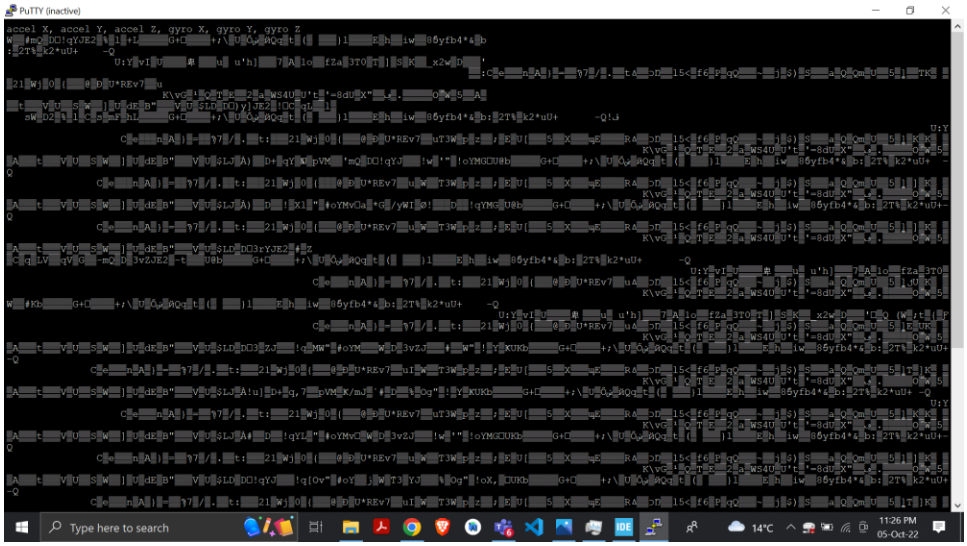

Figure 10: After Compression and Encryption on STM

Figure 10 above is the final output after both compressing and encrypting the data on the STM. Due to technical difficulties as mentioned above, we were unable to produce screenshots of the data after decompressing and decrypting the data on the PC. However, the programs on the STM ran to completion and the execution times computed were tabulated below.

Compression

The following results show the execution time and compression time the huffman algorithm used in the STM.

Table 2: Results of compression algorithm

Data Size (bytes)	Execution time (s)	Compression Ratio (%)
512	0.000011	87.5
1024	0.000021	87.5
2048	0.000107	87.5
4096	0.00037	87.5

The time execution does increase as the input size increases, however the compression rate stays the same, this is because the algorithm is and Adaptive Huffman compression which allows the us to set the compression ratio. The reason for choosing 87.5 is because it is does not compress the data to the point of corrupting it while still saving space on the STM.

Encryption

The following is the data is the time execution results of different bytes size under encryption.

Table 3: Results of encryption algorithm

Data size (bytes)	Execution time (s)
512	0.000010
1024	0.000017
2048	0.000104
4096	0.00025

As suggested by table 3 above, increasing the size of the input data increases the execution time of the algorithm as it has more data to process.

ATPs

Table 4: ATP summary (Original)

ATP	ATP Met (Y/N)	Why ATP hasn't been met	Changes
When the data is compressed, the compression ratio must be at least under 90%.	N, new ratio is 87.5%	This algorithm has a lower compression ratio	We changed the compression algorithm from processing files to processing arrays of characters. In order to work on STM
The compression data should have at least 25% of the Fourier coefficients of the data.	N	Ran out of time to test this	Implement in final report
Compression does not take more than 90 seconds to run.	Y	-	-
Test if the data is sent from the microcontroller to the PC and vice versa	Y	-	-
Test if communication between IMU and microcontroller is working	Y	-	-
Encryption does not take more than 90s to run	Y	-	-

Ciphertext different from plain text	Y	-	-
Original data obtained after decryption	N	Serial corrupted data from STM	Re-work algorithms to cater for this
all the original data is recovered after encryption, decryption compression and decompression	N	Decryption and decompression did not run as original data was lost	Re-work serial coms and interfacing with PC

Table 5: New ATPs

ATP	Overall System (now including IMU)	Compression	Encryption
1	Reading a block of data from IMU should not take longer than 1s.	Compression ratio must be ...	Algorithm must implement a block cipher
2	Amount of data read from IMU can be controlled	Compression algorithm runs on the STM32	Encryption algorithm runs on STM
3	Sensible values produced by sensors on IMU		
4	Overall system takes no longer than 30s to complete		
5	Original data obtained after processing data		
6			
7			
8			
9			
10			

References

[1] *Invensense.tdk.com*, 2022. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2021/07/DS-000192-ICM-20649-v1.1.pdf. [Accessed: 04- Oct- 2022].

[2]*Cdn.sparkfun.com*, 2022. [Online]. Available: https://cdn.sparkfun.com/assets/7/f/e/c/d/DS-000189-ICM-20948-v1.3.pdf. [Accessed: 04- Oct- 2022].