First Progress Report: Milestone 2

Yashil Luckan* and Pooja Jugnarayan*
Department of Electrical and Computer Engineering
University of Cape Town

September 4th, 2022

* Email addresses lckyas002@myuct.ac.za, and jgnpoo001@myuct.ac.za.

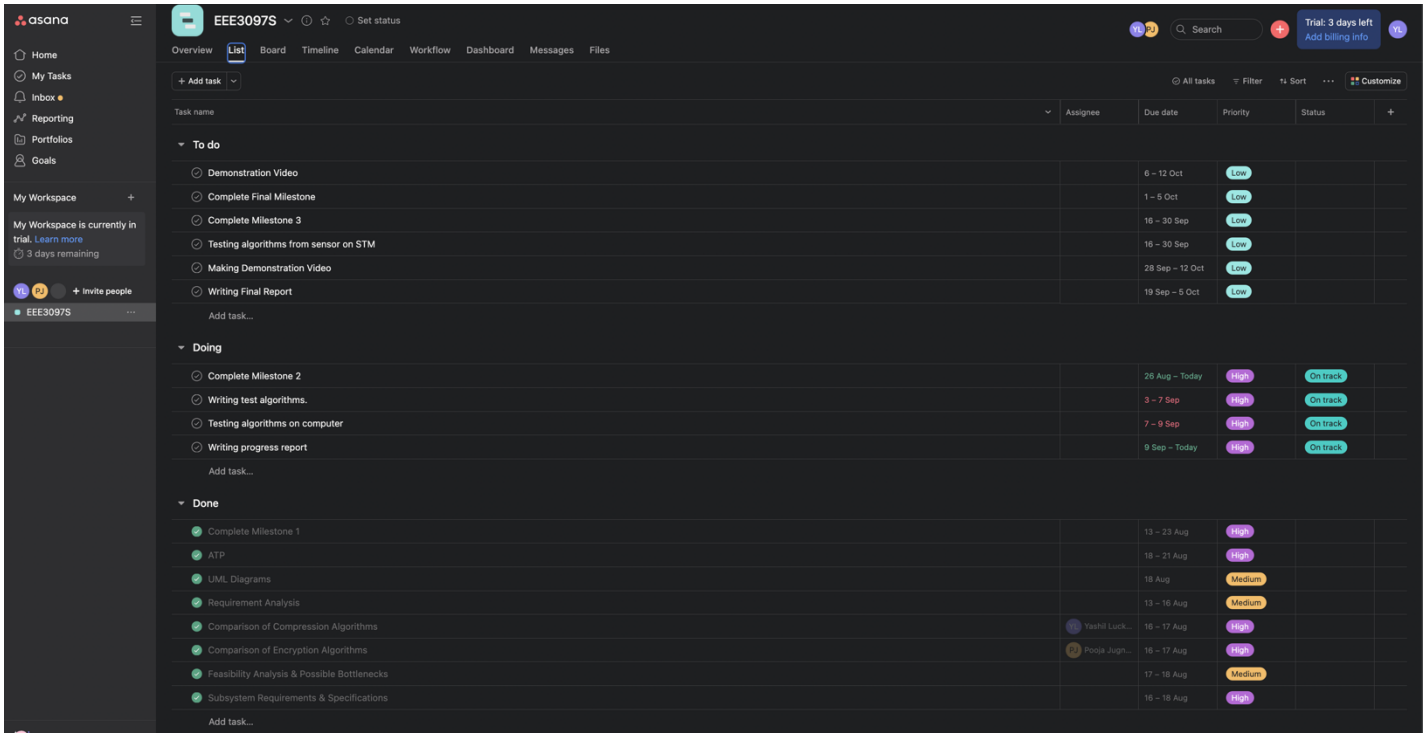
Table of Contents

1. Admin documents

1.1. Table of individual contributions

Item in table of contents	Contributors
Compression	Yashil
Encryption	Pooja
Data used	Yashil and Pooja
Data justification	Yashil and Pooja
Initial analysis	Yashil and Pooja
Overall experiment 1	Yashil
Overall experiment 2	Pooja
Compression experiments	Yashil
Encryption experiments	Pooja
Data types	Yashil and Pooja
Overall functionality experiment 1	Yashil
Overall functionality experiment 2	Pooja
Compression functionality	Yashil
Encryption functionality	Pooja
Acceptance performance definition	Yashil and Pooja
Traceability Matrix	Yashil and Pooja

1.2. Project management tool

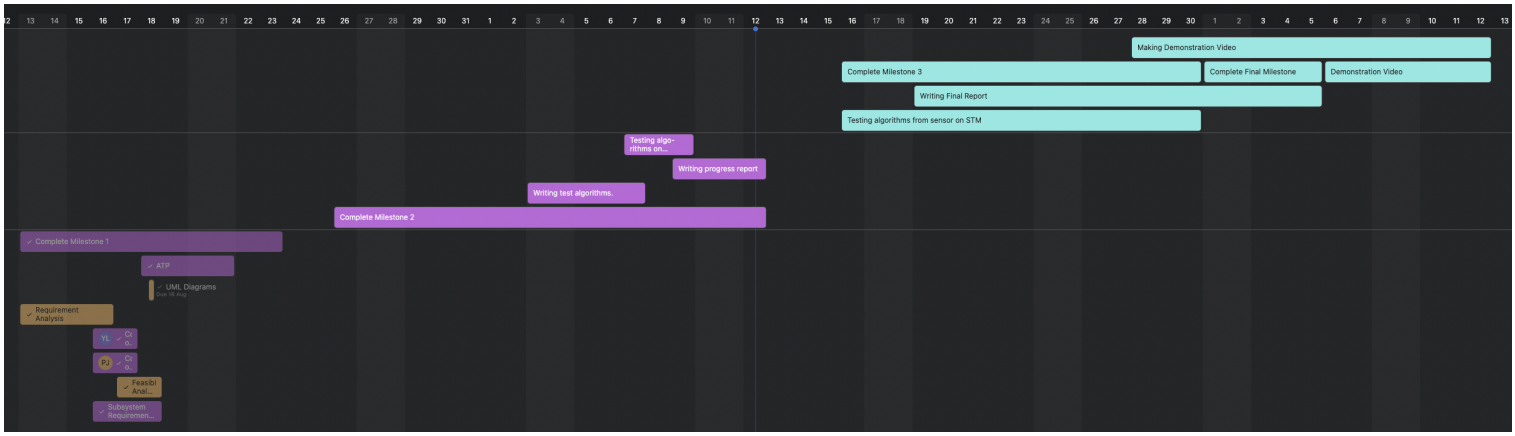


The project management tool of choice is Asana. The image above shows the breakdown of the tasks at hand.

1.3. Github link

https://github.com/yashilluckan/EEE3097-Team6.git

1.4. Timeline



The above picture shows the Asana timeline, and we are on track and the tasks at hand at this point in time have been completed i.e., progress is on time.

2. Data

2.1. Data used

The data to be used are CSV files that were communicated to us via email from Humphrey Chiramba. The following CSV files were chosen because it is data that was captured by the IMU, and the raw data is in a readable format.
The CSV files were modified to contain only 1000 lines of data, as this size still allowed the compression to work effectively whilst being small enough to mimic the size of the data files that will be created from the data received from the IMU.
The CSV files include:

1. EEE3097S 2022 Walking Around Example Data.csv
2. EEE3097S 2022 Turntable Example Data.csv
3. EEE3097S 2022 Turntable Example Data 2.csv.

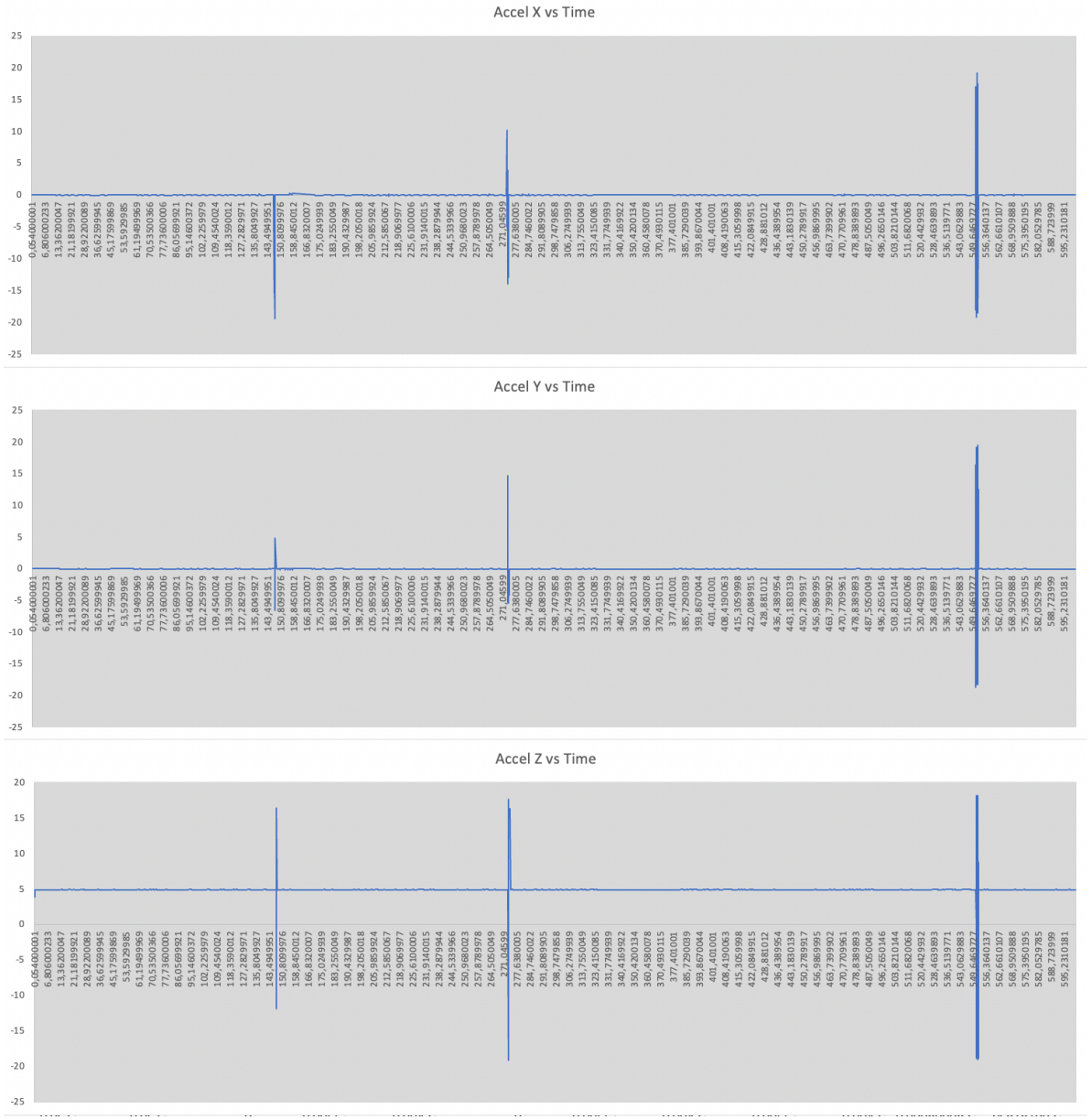
The primary tests will be used on "Turntable Example Data.csv" and the other two will be used to verify our results.

2.2. Data justification

The following justifications are based off the updated ATP's found under Section 5.1:

A1	The CSV files can be read in C without errors being shown, thus this ATP can be tested fully.
A2	*Unapplicable as tests using the aforementioned data are not conducted on STM yet. Only on a computer. However, code that causes the STM LED to flash is sent to the STM to ensure that data can be sent and interpreted by the STM. Timing of this data would serve no purpose as this data is irrelevant to the scope of this project.*
A3	The CSV files can be read in C without errors being shown, thus this ATP can be tested fully.
A4	Contents (characters) of the CSV files can be compared to each other natively in C and hence this ATP can be tested fully.
A5	The compression algorithm will take time to run and hence this ATP can be fully tested.
A6	The encryption algorithm will take time to run and hence this ATP can be fully tested.
A7	*Unapplicable as tests for compression and encryption are not conducted on STM yet. Only on a computer. Thus, it is not possible to check if the output file is the same as the original.*

2.3. Initial analysis



The is no need to show the frequency domain of the IMU readings. This is because, later it would have been used to compare the input and output data to ensure that 25% of lower Fourier Coefficients are retained after extraction. However, GZIP, which is the algorithm of choice is lossless and hence there is no need to compare Fourier Coefficients.

3. Experiment setup

3.1. Simulations and experiments to determine overall functionality

The following experiments were performed to determine overall functionality:

- Experiment 1: To test the reading and writing to the STM, we wrote code to cause an LED on the STM to flash
- Experiment 2: In order to determine the overall functionality of the program, a file was encrypted then compressed and thereafter decrypted and decompressed. A test script was written in the C programming language to check the integrity of the CSV file after this process. It was compared to the original CSV file.

3.2. Subsection experiments

3.2.1. Compression experiments

The following experiments were performed on TurntableData.csv to test if the GZIP algorithm works:

- Experiment 1: To test if the file was successfully being read by the algorithm, GZIP was set to run on the CSV file, if the file was successfully read then an output was produced in the same directory e.g., TurntableData.csv.gz. If the file was unsuccessful the output would be a response in terminal describing the error e.g., not in gzip format.
- Experiment 2: The timing mechanism built into Unix based systems was used to time the algorithm (time) when the gzip algorithm was called from terminal. E.g., time gzip TurntableData.csv.
- Experiment 3: A test script was written in the C programming language to check the integrity of the CSV file after extraction. It was compared to the CSV file before compression.

3.2.2. Encryption experiments

The following experiments were performed on TurntableData.csv to test if the AES algorithm works:

- Experiment 1: To test if the file was successfully being read by the algorithm, a test was written to display an error if the CSV file was NULL in C and to display the plaintext that will be encrypted in terminal to further prove that the file is being read.
- Experiment 2: The timing mechanism built into Unix based systems was used to time the algorithm (time) when the AES algorithm was called from terminal. E.g., time gcc -o aes aes.c main.c.
- Experiment 3: A test script was written in the C programming language to check the integrity of the CSV file after decryption. It was compared to the CSV file before encryption.
- Experiment 4: The same test script as experiment 3 will be run on the encrypted and original file to show that there are differences in the file and it is sufficiently encrypted.

3.3. Data types

After compression the output file should be of type: .csv.gz.
After extraction the file will be the original file before compression and will be of type: csv.
The file type after encryption as well as the file type after decryption are .csv.

4. Results

4.1. Overall functionality

- The first experiment showed the correct LED flashing on the STM board. This shows that data and instruction can be successfully written to the STM.
- The results from the second test yielded that there were no errors or mismatches between the files before encryption and compression and after extraction and decryption. The terminal output displayed: “Errors: 0”. This means that the compression algorithm is lossless and that no data was lost during the encryption and decryption process

4.2. Subsection functionality

4.2.1. Compression functionality

- The results from test one shows the CSV file was able to be successfully compressed as no error messages such as “not in gzip format” were thrown to the terminal. Instead, the compressed file was delivered to the same directory as the original file and had the following name: TurntableData.csv.gz. The .gz extension signifies that the file was successfully compressed. The input file size was: 22 429 420 bytes. The output file size was: 2 830 055 bytes. This gives an approximate compression ratio of 0.126 and is 7.925 times smaller than the original file.
- The results from the second test yield that GZIP compression on TurntableData.csv took a real-world clock time of 0.455 seconds. The CPU clock cycles 0.011 seconds. Extraction took 0.051 seconds of real-world clock time and 0.011 seconds of CPU cycle time. The above values are an average after repeating the second experiment 2 times.
- The results from the third test yielded that there were no errors or mismatches between the files before compression and after extraction. The terminal output displayed: “Errors: 0”. This means that the compression algorithm is lossless. Errors were intentionally introduced by deleting values in order to ensure the IntegrityChecker.c file works correctly and the following output was produced after removing the headings of each value in the CSV file:

  Total Errors: 160

Line Number: 1	Error Position: 1
Line Number: 1	Error Position: 2
Line Number: 1	Error Position: 3
.	.
.	.
.	.
Line Number: 1	Error Position: 160

4.2.2. Encryption functionality

- There was no error message displayed in terminal when the "TurntableData.csv" was run through the testing algorithm. The plain text displayed in terminal matched the text in the CSV file which showed that the file was being read accurately.
- The timing algorithm was run on for only encryption of the "TurntableData.csv" file. This experiment was repeated three times and the real time average was 0m0.082 which is sufficient. The timing algorithm was run another three time for both encryption and decryption of the file which took an average 0m0.103s in real time. This shows that the algorithm runs in a reasonable amount of time and that encryption alone is quicker than both encryption and decryption.
- The results from the third test yielded that there were no errors or mismatches between the files before encryption and after decryption. The terminal output displayed: "Errors: 0". This shows that no data was lost during the encryption and decryption process. Errors were intentionally introduced by deleting a row in the CSV file in order to ensure the IntegrityChecker.c file works correctly and the following output was produced after removing the headings of each value in the CSV file:

  Line Number : 1 Error Position : 1
  Line Number : 1 Error Position : 2
  .
  .
  .
  Line Number : 1 Error Position : 451
  Total Errors : 450

- The results from the final test showed a 360504 Errors which indicates that little to no values in the encrypted and original file matched. This proves the validity of the encryption algorithm.

4.2.3. Additional Testing

- When all above tests were run with Walking Around Example Data.csv and Turntable Example Data 2.csv, identical functionality was observed and it was clear that the algorithms were both effective.

5. ATP

5.1. Acceptance performance definition

Previous ATPs:	
A1	A test will be run to check if the file is being read in by the compression algorithm.
A2	A test will be done to check the read and write speeds of the STM.
A3	A test will be run to check if the file is being read in by the encryption algorithm.
A4	A test script will be run to check if the output file after extraction and decryption is exactly the same as the raw data file.
A5	A test will be run to determine if the compression algorithm executes in minimal time and use minimal power.
A6	A test will be run to determine if the encryption algorithm executes in minimal time and use minimal power.
A7	A test will be done to check if the correct data is on the STM.

Updated ATPs:	
A1	The GZIP algorithms has built in file checking functionality. If the file contents are read successfully before the compression algorithm is run then then an output file of format: .csv.gz is outputted into the same directory. If the file was unsuccessfully read or any other reason to that led to the file not being compressed, then an output to the system out will say “File not in gzip format”.
A2	A timing mechanism in Unix will be used to calculate the read and write speeds. The mechanism will encapsulate the reading from or the writing to STM block of code. The Read/write speed will be calculated by taking file size and dividing by the time taken to read/write. A pass will be indicated by a speed greater 100B/s then and a fail will be indicated as a speed less than 100B/s
A3	An if-statement will be used to test if data was read in from the file by testing if the file was empty within a testing algorithm written in C. If the file contents are read before the encryption algorithm is run successfully then then a “success” message will be displayed which will qualify as a pass. Else, a “Error” message will appear if the file was unsuccessfully read which will qualify as a fail.
A4	A test script will be written and will be run to check if the output file after extraction and decryption is the same as the raw data file. The test script will be written in C and will include and algorithm that compares every character of the output and input file and verifies that it is the same.
A5	A timing mechanism in Unix will be used to calculate the runtime of the compression algorithm. The mechanism will encapsulate the entire algorithm. A pass will be indicated by a time less than 1.1s and a fail will be indicated as a time greater than 1.1s.
A6	A timing mechanism in C will be used to calculate the runtime of the encryption algorithm. The mechanism will encapsulate the entire algorithm. A pass will be indicated by a time less than 1s and a fail will be indicated as a time greater than 1s.
A7	Once the file is written onto the STM, it will be sent back to the computer to ensure that the file sent to the STM is indeed correct. A test script will be written and will be run to check if the input file is the same as the raw data file sent to the STM. The test script will be written in C and will include and algorithm that compares every character of the two files and verifies that it is the same.

5.2. ATP Performance

Updated ATPs:	
A1	When the compression algorithm was run, no errors were observed and thus this ATP is being met.
A2	This ATP could not be fully tested as no encryption or compression algorithms were written to the STM at this stage in testing.
A3	When the encryption algorithm was run, no errors were observed and the original text was displayed to be the same as the text entered and thus this ATP is being met.
A4	The file before encryption and compression was exactly the same as the file after decryption and extraction and thus this ATP is met as integrity is fully preserved.
A5	The time taken for the whole compression algorithm to run was less than 1s and thus this ATP is met.
A6	The time taken for the whole encryption algorithm to run was less than 1s and thus this ATP is met.
A7	This ATP could not be fully tested as no compression or encryption algorithms were written to the STM at this stage in testing.

5.3. Traceability matrix

UR	FR	Specs	ATP	Pass/Fail
U1	U1.F1	U1.F1.S1	U1.F1.S1.A1	Pass
	U1.F2	U1.F2.S2	U1.F2.S2.A2	-
	U1.F3	U1.F3.S3	U1.F3.S3.A3	Pass
	U1.F4	U1.F4.S4	U1.F4.S4.A2	-
U2	U2.F5	U2.F5.S5	U2.F5.S5.A4	Pass
	U2.F6	U2.F6.S6	U2.F6.S6.A4	Pass
	U2.F7	U2.F7.S7	U2.F7.S7.A5	Pass
	U2.F8	U2.F8.S8	U2.F8.S8.A5	Pass
U3	U3.F9	U3.F9.S9	U3.F9.S9.A6	Pass
	U3.F10	U3.F10.S10	U3.F10.S10.A6	Pass
	U3.F11	U3.F11.S11	U3.F11.S11.A6	Pass
	U3.F12	U3.F12.S12	U3.F12.S12.A6	Pass
U4	U4.F13	U4.F13.S13	U4.F13.S13.A7	-
	U4.F14	U4.F14.S14	U4.F14.S14.A7	-
	U4.F15	U4.F15.S15	U4.F15.S15.A7	-
	U4.F16	U4.F16.S16	U4.F16.S16.A7	-

6. References

[1]Research.cs.wisc.edu, 2022. [Online]. Available: https://research.cs.wisc.edu/wpis/examples/pcca/gzip/gzip.c. [Accessed: 12- Sep- 2022].

[2]"GitHub - openluopworld/aes_128: Implementation of AES-128 in pure C. No modes are given. Only one block of encryption and decryption is given here.", GitHub, 2022. [Online]. Available: https://github.com/openluopworld/aes_128. [Accessed: 11- Sep- 2022].