University of Cape Town

EEE3097S

ENGINEERING DESIGN: ELECTRICAL AND COMPUTER ENGINEERING

EEE3097S First Progress

Author:	*Student Number:*
Gerald Maswoswere	MSWGER001
Thabelo Tshikalange	TSHTHA094

September 12, 2022

Contents

1 Table listing individual contributions 3
1.1 Github section . . . 3
1.2 Project reoprt progress . . . 3

2 Data 4
2.1 Discuss which data you will use and why . . . 4
2.2 justification of that the data to use is good enough to run all the ATPs 5
2.3 Initial analysis of the data . . . 7

3 Experiment Setup 8
3.1 Simulations to check the overall functionality . . . 8
3.2 Compression Block . . . 9
3.2.1 Input and output of the compression block . . . 10
3.2.2 Input and output of the decompression block . . . 10
3.3 Encryption . . . 10
3.3.1 Input and output of the encryption block . . . 12
3.3.2 Input and output of the decryption block . . . 12

4 Results 13
4.1 Compression Block . . . 13
4.2 Encryption Block . . . 13
4.3 Total run time . . . 13

5 ATPs 14

List of Figures

1 Snapshot of project management tool 4
2 Snapshot time line 4
3 Sample code 7
4 FFT for X_axis 7
5 FFT for Y_axis 8
6 Encryption/Decryption sub-system 11
7 AES 256 testing 12

List of Tables

1 Compression specifications 5
2 Encryption specifications 6
3 Updated specifications 14
4 old specifications 14
5 atps met or not 15

1 Table listing individual contributions

Each group member was assigned a subsystem to investigate, and determine how their respective subsystem will work in the greater design. Thabelo was assigned the encryption subsystem and Gerald was assigned the compression subsystem

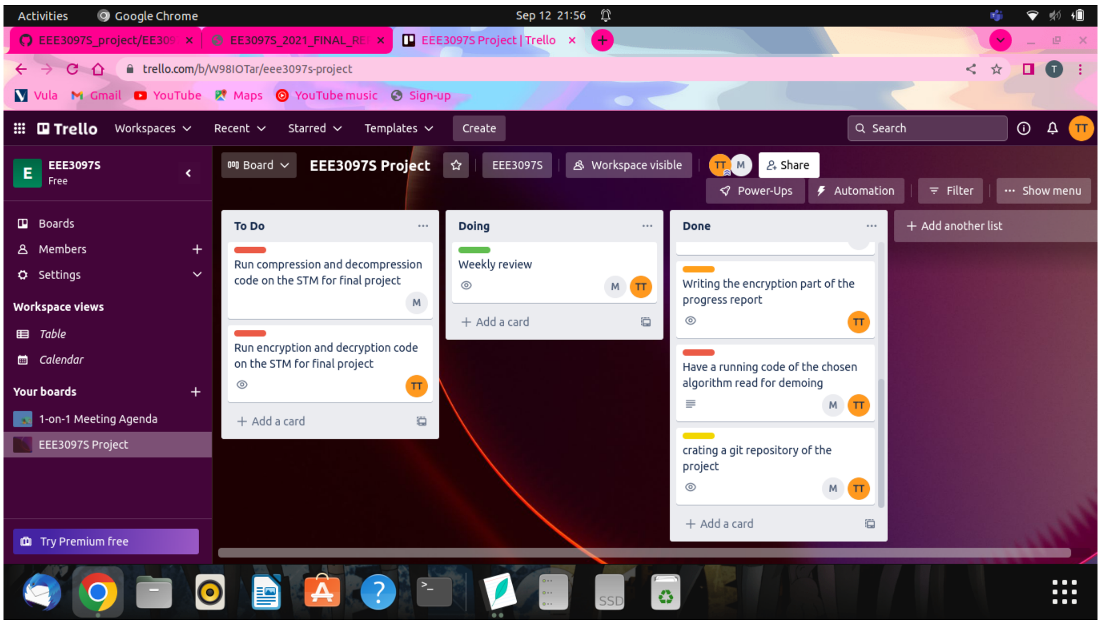
Contributions:	Contributer:
Requirement Analysis	Thabelo
Subsystem design	Gerald
Development Timeline	Thabelo and Gerald
Compression/decompression algorithm	Gerald
Compression Benchmarking	Gerald
Encryption/Decryption algorithm	Thabelo
Encryption Benchmarking	Thabelo
System Benchmarking	Gerald and Thabelo
GitHub Admin	Gerald and Thabelo
Report Writing	Gerald and Thabelo
IMU Module	Thabelo
Experiment Setup	Gerald
Results	Gerald and Thabelo
Acceptance Test Procedure (ATP)	Gerald and Thabelo

1.1 Github section

https://github.com/mswger001/EEE3097S_TSHTHA094_MSWGER001

1.2 Project reoprt progress

Link of project management: https://trello.com/b/W98IOTar/eee3097s-project


To Do

Run compression and decompression code on the STM for final project

Run encryption and decryption code on the STM for final project

Doing

Weekly review

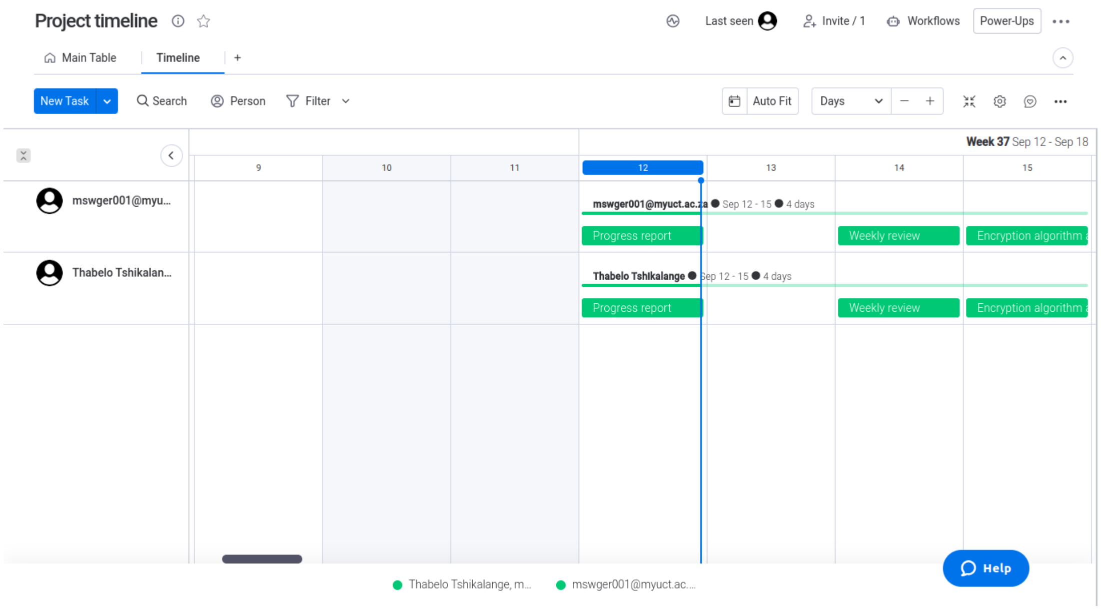
Done

Writing the encryption part of the progress report

Have a running code of the chosen algorithm read for demoing

crating a git repository of the project

Figure 1: Snapshot of project management tool


Project timeline

mswger001@myuct.ac.za ● Sep 12 - 15 ● 4 days: Progress report, Weekly review, Encryption algorithm

Thabelo Tshikalange ● Sep 12 - 15 ● 4 days: Progress report, Weekly review, Encryption algorithm

Figure 2: Snapshot time line

2 Data

2.1 Discuss which data you will use and why

We have chosen to use the given data int the anouncements as well as in the vula resources where we are sure it contains all the required fields and is the actual data

specification found on the bouy, the link is below
https://vula.uct.ac.za/access/content/group/4542c49f-47ed-40a0-8b3c-efed3756d216/03. the sensors measures force, angular force and magnetic field. IMU are a consist of 3-axis of accelerometer, 3 – axis of gyroscope, which could be 3 – axis of magnetometer, would be a 9-axis IMU. Accelerometer, is measure of change of velocity in the imu sense hat across a single axis. Accelerometer, measures linear acceleration in a single direction. Gyroscope measures the angular velocity about three axes, hence pitch, yaw and roll. Lastly, the magnetometer, measures the fluctuations in the Earth's magnetic field by measuring the air's magnetic flux density. All these are represented in the example csv file as well as the raw file provided by the proffesor.

some of the raw data extracted from the above is also available on the github repository. Running the parser and we will able to view the files and opt for the most recent files which and we then able to perform some simple analysis on. This data allows us to model the size of the data we are expecting to work with and run using on the stm32.

More importantly the raw data is in the same format as the data that will be produced by the actual IMU

2.2 justification of that the data to use is good enough to run all the ATPs

specID	*Specification*
SP01	STM Communication using I2C/ UART interface
SP02	The Lz77 will be used to compress the data from the IMU as well as decompress
SP03	compression ratio ≥ 1.25
SP04	The compression algorithm should account for at least 25% of the lower fourier coefficients if there is data loss at decompression.
SP05	The compression algorithm must take no longer than 5 seconds per 10 kilobytes of data.

Table 1: Compression specifications

For the compression specifications, the given data can be added to the stm via UART or I2C inteface which can then accounnt for SP01
applying a compression algorithm and being able to compress and decompress the file while preserving the state of the contents of the data file would test SPO2
by checking the data size after and before compression the ratio for compression can

be asserted thus SP03 is checked
by then working on the uncompressed data to perform fourier transforms the lower fourier coeficients can be validated if they are there
the time taken to compress the data can also be timed using the stm on the data to find how long it takes to compress it
Thus covering the data is good enough for the compression ATPs

specID	*Specification*
SP06	The execution time of the algorithm should not exceed 10 seconds per 10 kilobytes of data.
SP07	No data must be lost during the encryption and decryption of the data file.
SP08	The original data file's contents must be identical to the decrypted data file's contents

Table 2: Encryption specifications

2.3 Initial analysis of the data

Using Matlab the following code was used to generate the plotting of the data given to get a more clearer representation.

```
>> Array=csvread('EEE3097S 2022 Turntable Example Data.csv',2);
col1 = Array(:, 1);
col2 = Array(:, 18);
y = fft(col2);
Ts = 1/50;
fs = 1/Ts;
f = (0:length(y)-1)*fs/length(y);
plot(f, abs(y))
title('frequency vs fft(Z\_axis)');
xlabel('frequency');
ylabel('fft(Z\_axis)');
```

Figure 3: Sample code

This was the data that was observers before it was compressed and the aim is that the same data will be preserved after it gets compressed.

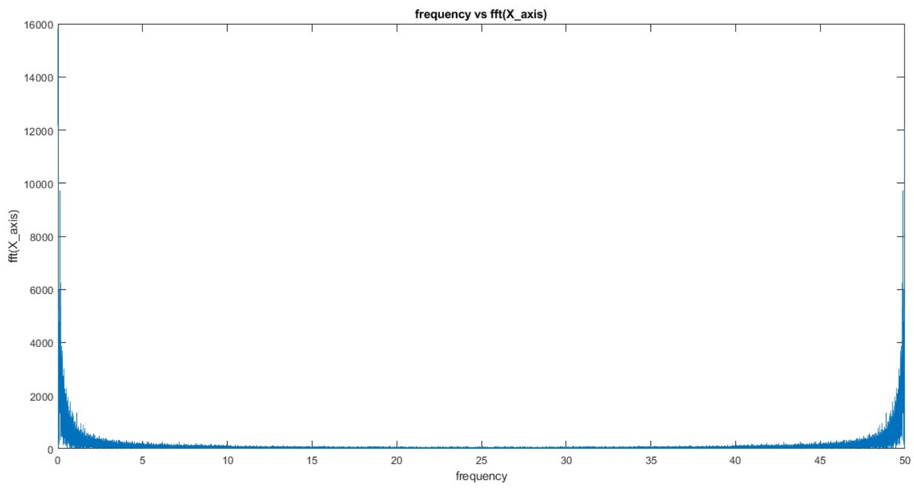


Figure 4: FFT for X_axis

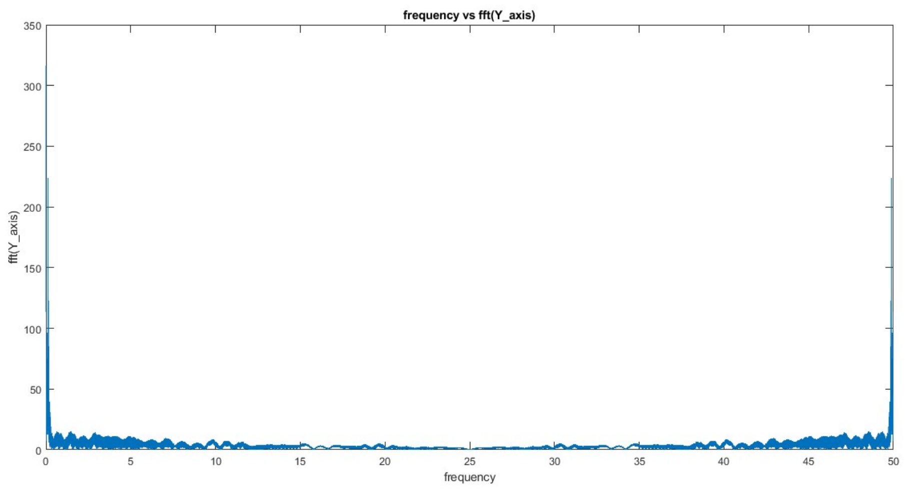


Figure 5: FFT for Y_axis

3 Experiment Setup

3.1 Simulations to check the overall functionality

By sending simple data to a modelled system on a laptop the data is sent as a stream which is taken and stored into a variable labelled ”data” which flow as stream of bits these are displayed back to the screen to see if data is not corrupted, the data through a main file goes through to the to the compression algorithm where its size is reduced and a copy of the compressed file is created ,stored as raw-Data and previewed on the format as well state. The raw-Data is then taken through to the encryption block which encrypts the raw-Data to a set of encrypts characters, the encrypted characters are also displayed to show difference between before and after encryption, the encrypted characters are then decrypted with an expectation to find identical characters prior to encryption, these characters are displayed to screen and compared to the encrypted as well as before encryption. the decrypted characters are taken back to the compressing and decompressing block for decompression, the characters are decompressed and are expected to be the same as the one prior to the movement of data from the compression block, the two are compared to check for integrity. the overall time taken for the basic implementation was timed using timestamps that were placed at the begging of the console application to the end

readdata– compressData– encryptData– transfer– decryptData– decompressData

3.2 Compression Block

The objective of the experiment is to test the compression ratios if they meet the specifications. The experiment seeks to check if the decompressed files are the same as the original file. The experiment will check the best compression level/algorithm to archive speeds required to avoid backlog and keep up with the transmission speeds.

initial experiments on the compression algorithm were done on a laptop, by testing out a couple of open source libraries and comparing the benifits of each it was chosen to change from the zstd algorithm to the Lz77 algorithm mainly due to ease of use of library when loading to the stm32.

the first step to compressing was compressing a simple txt file and get the file to reduce in size.

this was achieved by running a copy of the Lz77 algorithm from
https://github.com/cstdvd/lz77

the intructions followed were the use of a terminal to run and compress files from the terminal whilst specifying the arguments. this was a succes for the small file which came directly from the root folder where the algorithm resided in.

next step taken was to test if the algorithm could handle large amounts of data sent to it to compress by using a large dataset from the internet the process was seen to be able to withstand large amounts of data whilst being able to still reduce the file size

next step was to find if one could retrieve the original contents of the now compressed file, the algorithm still through the terminal was used to decompress the file and the files were inspected to check if the data was identical prior to compression. this was a success.

the algorithm was then tested for speed of execution to find if it could suffice for the acceptance testing, this was done by using time stampps when the data was begining compression and another one after one after it was done compressing, the time take for decompression was also measured by the same methodolody

the algoritm was timed to see how long it took to compress a 30mb file and it took 1.484375s which satisfied that on the ATPs
the algoritm was timed to see how long it took to decompress to 30mb file and it

took 0.312500s which satisfied that on the ATPs

The compressed file to original file ratio was averagely 0.5 which also satisfied the atps

3.2.1 Input and output of the compression block

The input is a raw file data characters
The output is also raw file data characters

3.2.2 Input and output of the decompression block

The input is a raw file data characters
The output is also raw file data characters

3.3 Encryption

The objective of the experiment is to test the encryption if they meet the specifications. The experiment seeks to check if the encrypted files are unusable and the then decrypted files are identical to the prior encryption state.

We used AES encryption from the AES library and an algorithm adapted from Tiny AES in C
The encryptor class contains all the methods necessary to encrypt and decrypt files.

initial experiments on the encryption algorithm were done on a laptop, by testing out a couple of open source libraries and comparing the benefits of each it was chosen to stick with the AES encryption algorithm.

the first step was to encrypting an array of characters that were stored in memory this was to see if there was an easily visible pattern which would make the encryption insecure.

Just as the file was encrypted to a set of different chars which had no meaning at all and compared to the original one when rendered to the screen. this satisfied the encryption part of the ATP's

the file was run as a console application which had all the arguments set already.

after encryption was tested on the algorithm the decryption part was the next point of focus taking the encrypted data and using the same key as was on encryption the contents were decrypted and the original array of characters were observed.

the encrypted data was also tested using wrong keys to check if the data integrity was safe and secure by using many wrong keys the strength of the encryption was tested.

the time taken to encrypt the file was also checked to see if it was a feasible and time effective. By checking how long it took to encrypt and and how long it took to decrypt the algorithm was tested to see if it satisfies the time requirements.

Figure[6] shows the inner workings of the AES algorithm.

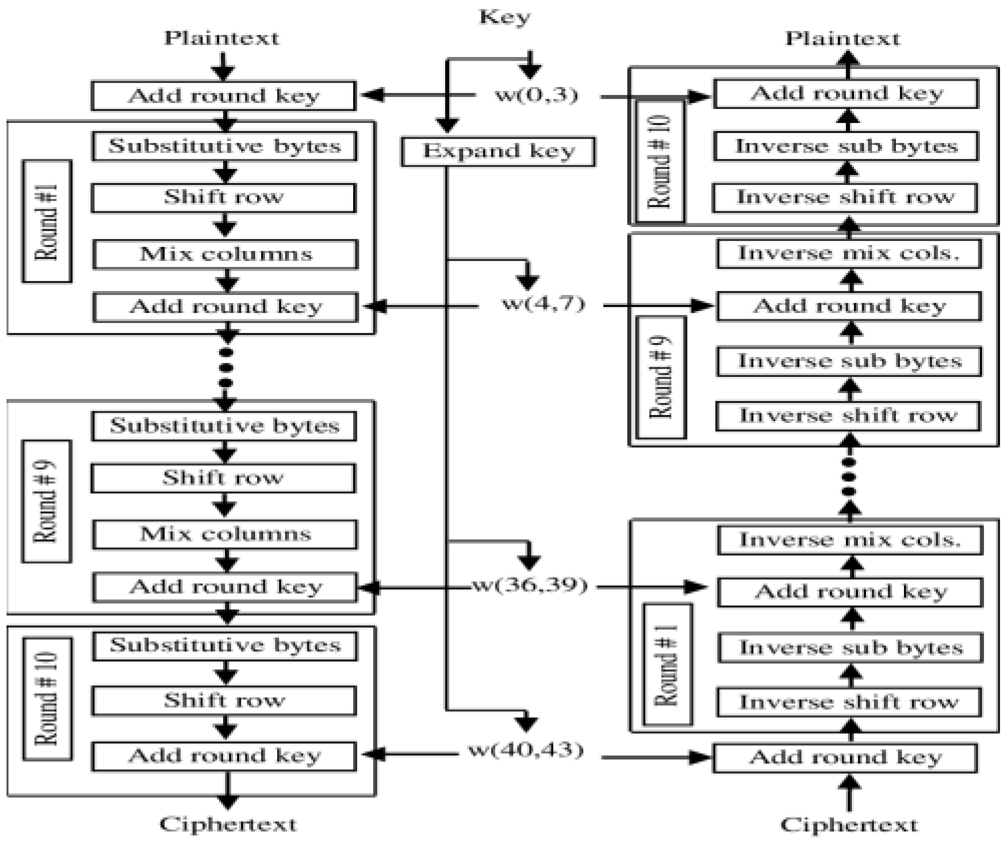


Figure 6: Encryption/Decryption sub-system

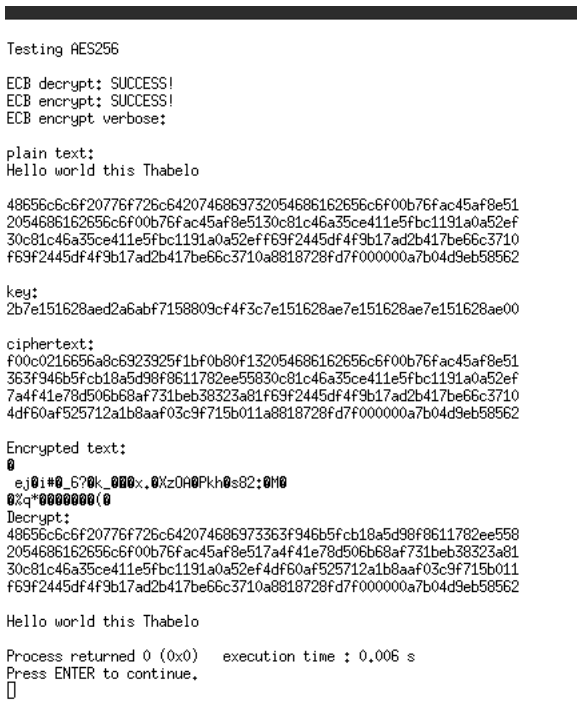
3.3.1 Input and output of the encryption block

The input is a raw file data characters
The output is also raw file data characters

3.3.2 Input and output of the decryption block

The input is a raw file data characters
The output is also raw file data characters

Some sample data was used to test functionality of the algorithm [7]



```
Testing AES256

ECB decrypt: SUCCESS!
ECB encrypt: SUCCESS!
ECB encrypt verbose:

plain text:
Hello world this Thabelo

48656c6c6f20776f726c642074686973205468616265 6c6f00b76fac45af8e51
```

Figure 7: AES 256 testing

4 Results

4.1 Compression Block

the algorithm was timed to see how long it took to compress a 30mb file and it took 1.484375s which satisfied that on the ATPs

the algorithm was timed to see how long it took to decompress to 30mb file and it took 0.312500swhich satisfied that on the ATPs

a file size of 3,36 MB (3 527 103 bytes) was compressed to 1,50 MB (1 581 817 bytes)

4.2 Encryption Block

The algorithm was timed to see how long it took to encrypt a 2048 bit plain text and it took 0.012s which satisfied that on the ATPs

The algorithm was timed to see how long it took to decrypt the cyphertext and it took 0.012swhich satisfied that on the ATPs

4.3 Total run time

depending on the data that would be in the compressed file the system might take longer to encrypt. but the overall run time of the system would take an average time of 1.868875s

5 ATPs

specID	*Specification*
SP01	STM Communication using I2C/ UART interface
SP02	The Lz77 will be used to compress the data from the IMU as well as decompress
SP03	compression ratio ≥ 1.25
SP04	The compression algorithm should account for at least 25% of the lower fourier coefficients if there is data loss at decompression.
SP05	The compression algorithm must take no longer than 5 seconds per 10 kilobytes of data.
SP06	The execution time of the algorithm should not exceed 10 seconds per 10 kilobytes of data.
SP07	No data must be lost during the encryption and decryption of the data file.
SP08	The original data file's contents must be identical to the decrypted data file's contents

Table 3: Updated specifications

specID	*Specification*
SP01	Raspberry Communication using I2C interface
SP02	The huffman-algorithm will be used to compress the data from the IMU.
SP03	The adaptive mode of Zstandard is supposed to be set to the minimum required.
SP04	compression ratio ≥ 1.25
SP05	The compression algorithm should account for at least 25% of the lower fourier coefficients if there is data loss at decompression.
SP06	The subsystem shall do minimum computations to save power
SP07 & SP08	AES encryption library will be used.

Table 4: old specifications

specID	*Met or not*
SP01	✓
SP02	✓
SP03	✓
SP04	✓
SP05	✓
SP06	✓
SP07	✓
SP08	✓

Table 5: atps met or not