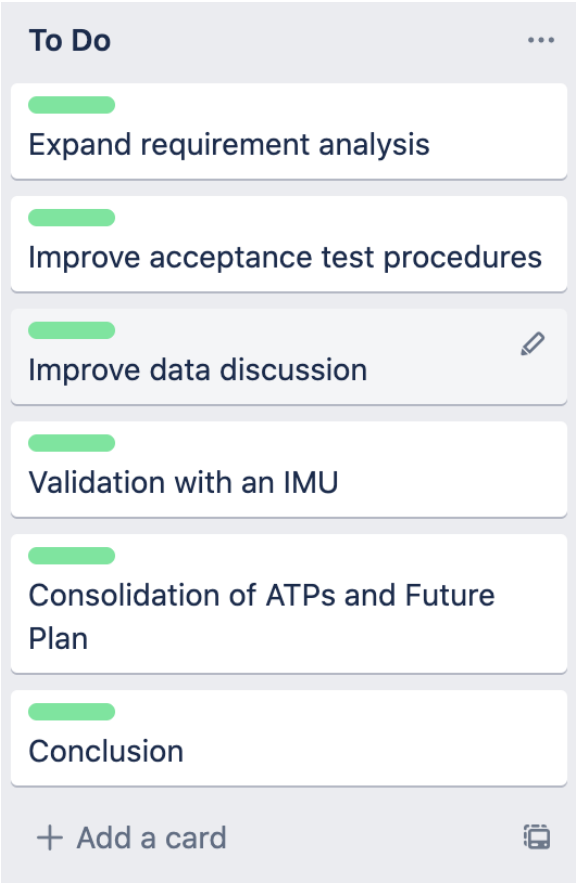
EEE3097 Final

PGRSAM001, BRDJAK002

1.
TABLE OF CONTRIBUTIONS

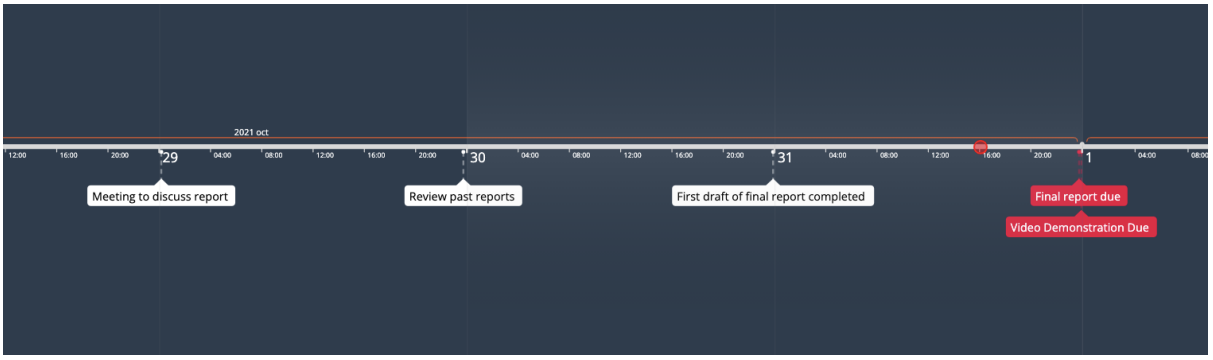
GROUP MEMBER	CONTRIBUTIONS	Section numbers	Page numbers
Jake Burditt	(compression subsystem), Admin, Requirement Analysis,Paper Design, Data	1,2,3,4	1,2,3,4,5,6
Sam Pogrund	(encryption subsystem), Admin, Paper Design, IMU, Consolidation, Conclusion	1,3,5,6,7	1,2,3,4,5,9,10,11

Screenshot of project management tool:



GitHub: https://github.com/spogrund/97-imu-data-manip.git

Timeline



2.

REQUIREMENT ANALYSIS

Interpretation of the requirements

Design an ARM based digital IP that will encrypt and compress inertial measurement unit (IMU) data. The data, which comes from an ocean buoy, will be provided by the ICM-20649 IMU. Due to data constraints, the measured data needs to be compressed, with 25% of its Fourier Series coefficients remaining after compression. The data also needs to be encrypted to ensure privacy. Due to the limited power available, the computation done by the processor needs to be limited as much as possible.

ACCEPTANCE TEST PROCEDURE

Figures of merits, based on which you would validate your final design

- Percentage of Fourier Series Available at output
- Compression Ratio
- Power Usage
- Encryption Strength

Experiment design to test these figures of merit

Feed data with a known output and compare the outputs.
Measure the power usage when completing a full cycle of compression and encryption.
Pass the data though basic encryption cracking algorithms to test the secureness.
Feed the data using a waveshare IMU and check the output.
Compare file sizes before and after compression.

Acceptable performance definition

- The device must be capable of scanning data from the IMU (ICM-20469) and converting it to a readable file
- From input data, the device should be able to produce a Fourier Series

- 25% of the Fourier Series coefficients should be available after compression and encryption
- Device should consume less than 3W of power when processing
- Data must be encrypted
- File size must decrease after compression

3.

Paper Design

Comparison of compression algorithms [1],[2],[3]

Run Length Encoding Algorithm is a simple compression algorithm which identifies consecutive sequences of symbols as runs or non runs. Runs are sequences with repeating symbols, while non-runs are those sequences not identified as runs. The algorithm keeps track of the length, position and repeating symbol in each run. Although simple and effective, the run-length algorithm is best used for plain text data, which our data is not.
Huffman Encoding uses the probabilities of the source data's symbols to construct code words. These codewords are shorter for more frequent symbols. The Huffman algorithm creates a binary tree of the source data and then traverses the tree and replaces the symbols with their codes.

Lempel-Ziv compression
The LZ77 algorithm replaces repetitions in data with references to an earlier existence of the repetition in the input data. The reference is a length-distance pair. The length specifies how many of the next characters are copies and the distance specifies how far back the 'original' characters are. In order to look back for originals, the LZ77 algorithm makes use of a sliding-window, which keeps track of a window of data to look into.

The LZ78 algorithm creates a dictionary based on the input data and replaces each repetition within the data with a reference to the dictionary. Each entry in the dictionary contains an index and a character, the index is the index of a previous dictionary entry and the character is use to append a string represented by the previous entry. Thus a string, for example, would be stored in reverse order, with the first letter in the string holding the index 0.

All LZ algorithms are based off of the LZ77 and LZ78.

Deflate Compression breaks the input data into blocks and then makes use of the LZSS algorithm and Huffman encoding to deflate and inflate the input data. The deflate algorithm has the majority of the control of how the data and its blocks are compressed and therefore a smart program is required if you want to have any control over the compression. Due to the use of multiple compression types, this algorithm is often very efficient.

Comparison of encryption algorithms [4]

There are two kinds of encryption types:
Symmetric encryption is a type of encryption where only one secret key is to to encrypt and decrypt the data.
Asymmetric encryption is a type of encryption that uses a private and public key that are mathematically linked. The public key is used to encrypt the data and the private key is used to decrypt the data

Triple DES encryption is a symmetric block cipher which uses 3 keys with 56 bits each.

Advanced encryption standard is a symmetric block cipher encryption algorithm that has a 128/192/256 bit key.

RSA encryption is a public key encryption algorithm that is generally used for sending data over the internet.

Feasibility analysis

The viability of the project being successful must be assessed. This is done by looking at various factors that have an effect on the degree of success of the project. These factors include:

- Cpu processing speed of the pi0
- Size of the key
- Power consumption
- Input size and rate and how long it takes to process.
- Complexity of the design and programs used(matching our skills to the level of the project)
- Level of accuracy and correctness

Possible bottlenecks

- The stream of input data received from the sender may arrive at too large of a size per unit time for the system to handle at the same rate, this would cause a slowdown of the system.
- A difference in processing time between encryption and compression sub-modules, this would mean that when data is finished with one process it would have to wait before going through the next process.

SUBSYSTEM DESIGN

Compression subsystem requirements

- Ensure data size is significantly reduced to reduce cost of transmission
- Data compression needs to result in less power consumed by the CPU
- The input data needs to be converted to a Fourier series
- 25% of Fourier series coefficients need to be obtained at the receiver
- Sufficient device memory is required for compression process
- Sufficient processing power is required to compress the data

Compression subsystem specifications

- Use C programming language
- Deflate compression using zlib
- Compute Fourier series using C

Encryption subsystem requirement

1. Encrypt the data securely

2. Ensure compatibility on the pi
3. Not use lots of power

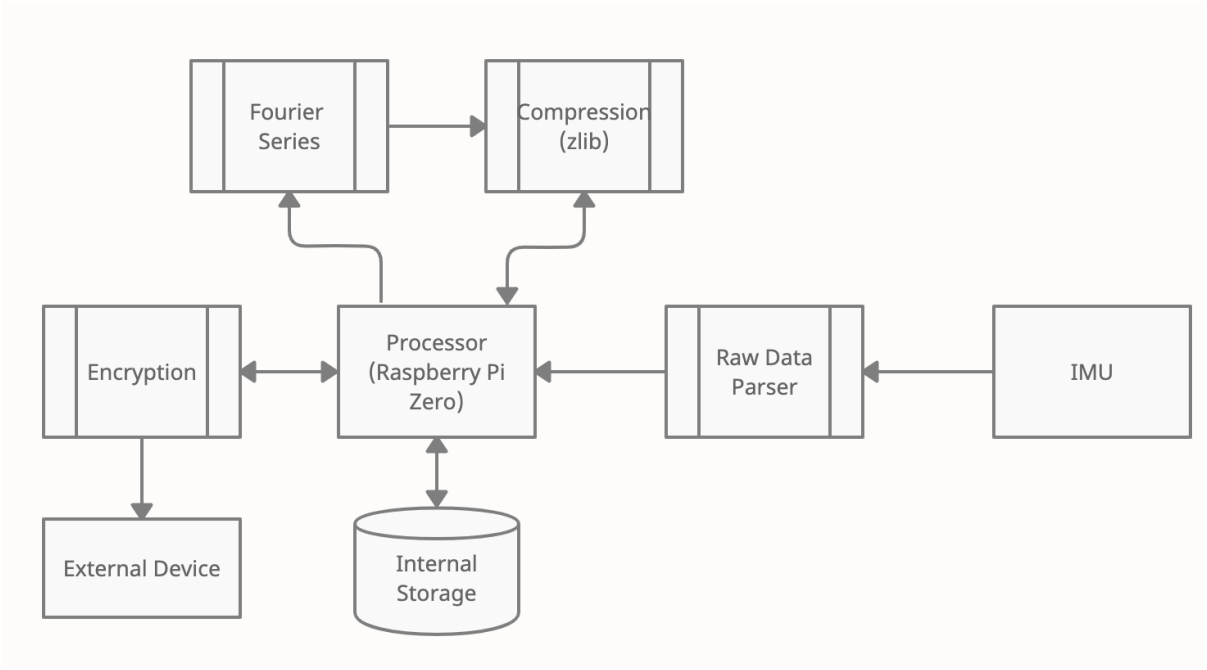
Encryption subsystem specifications

1. Use a 128-bit AES algorithm to encrypt data
2. Use python as the programming language
3. Use a 128 bit key as the bigger the key, the more processing power used.

Inter-Subsystem and Inter-Sub-subsystems Interactions

The IMU will feed data into the system where the first stage is to parse the data into a format that is readable to our system.
The data is fed to through the pi where it undergoes a fourier transform and the least 25% of fourier coefficients are extracted.
The data is then passed to the compression subsystem to be compressed.
The data then goes to the encryption sub-system to encrypt the data.
Finally the compressed and encrypted data is fed to the external device.

UML diagram



4.
Validation using simulated data

Simulation based validation is an important step in the design of any project. It is expensive and risky, in terms of both cost and time, to skip this step and attempt to carry out final stage tests. The simulation phase allows one to completely understand their own design, and if it has been modelled correctly enough, will allow for possible improvements and development to make itself

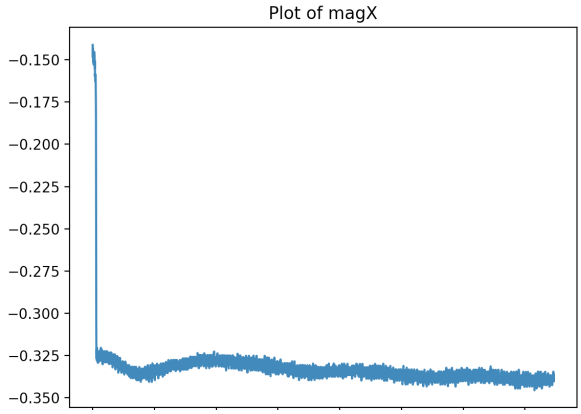
known. This optimisation of the design saves time and cost when moving onto the next stages of design.

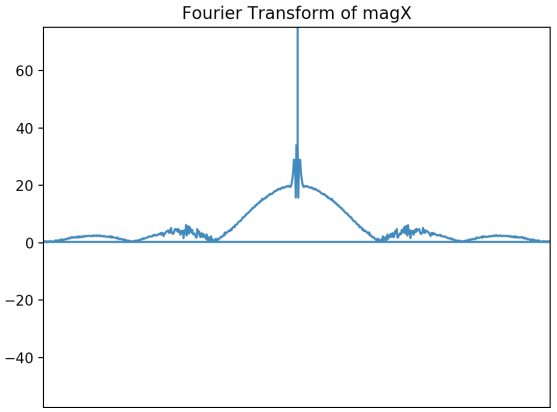
During the simulation phase, data taken from an IMU placed on a ship was used to replicate the use of an actual IMU. The steps taken in the simulation phase were as follows:

- The data was in a single CSV file
- The data was parsed in the main function of the simulation program. This data was converted to a number of arrays, each array representing a different IMU attribute
- The data was fourier transformed and printed to a CSV file
- The CSV file was encrypted and compressed
- The CSV file was then unencrypted and uncompressed
- The data was then reverse fourier transformed and printed to a CSV file
- The data at each stage was then compared to its original form to assess accuracy

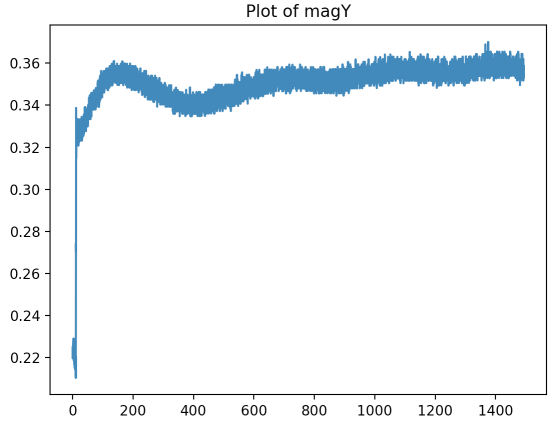
The Raw Set follows closely to the measurements that would be taken by the IMU, both in type and in value. The Data Set comes from an IMU, hence it is taking the correct type of measurements. The IMU, from which the Data has been taken, was placed on a ship, therefore the measurements it has taken will be very similar to those taken in our use case (on an iceberg).

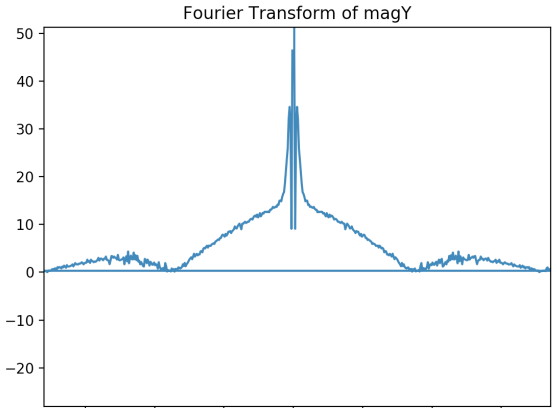
The figures of merits, in which we intend to validate our final design are Percentage of Fourier Series Available at output, Compression Ratio, Power Usage and Encryption Key Strength. Using the Raw Data set, a comparison of input and output Fourier Series is easily achievable and is an accurate representation of the same process with data measured from the IMU. The compression ratio can be simply measured. The power usage using the raw data set should closely follow that if the data was measured from our own IMU as their measurements are almost identical.



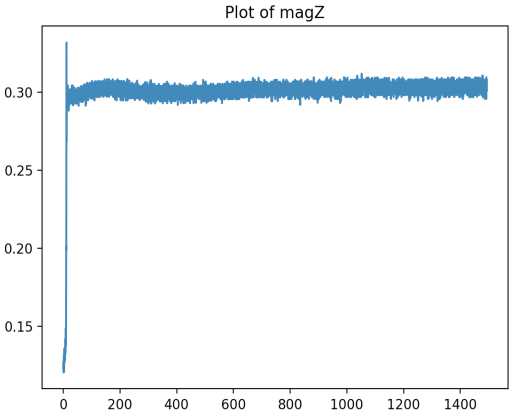


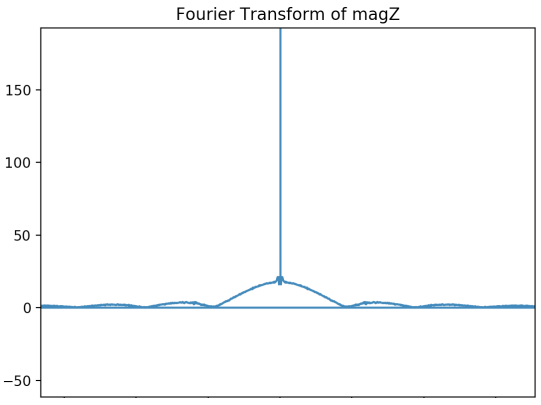
magX and the magnitude of the Fourier Transform



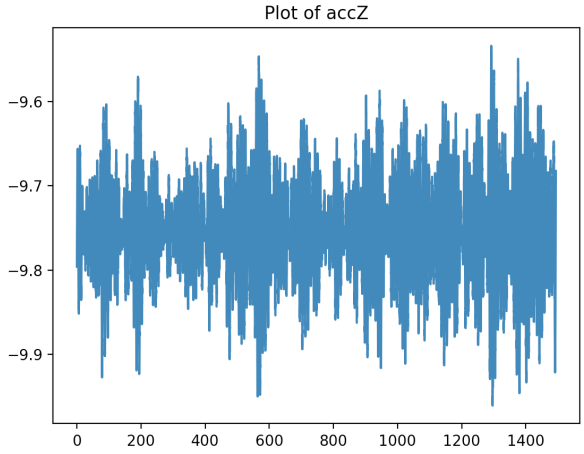


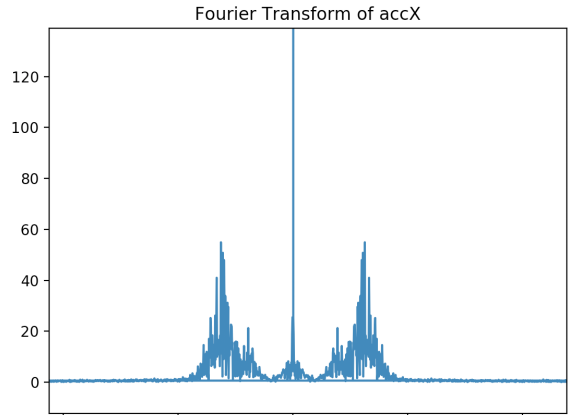
magY and the magnitude of the Fourier Transform



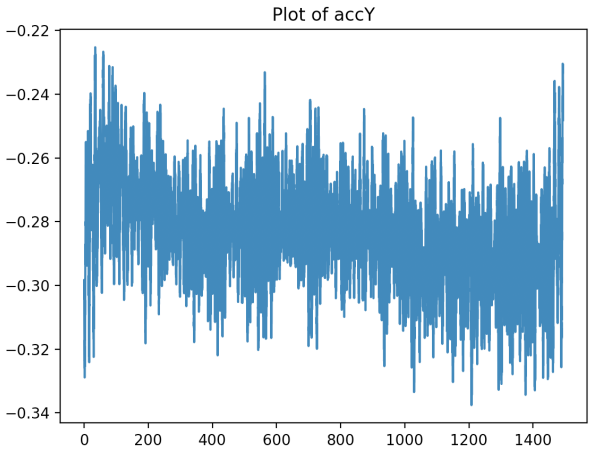


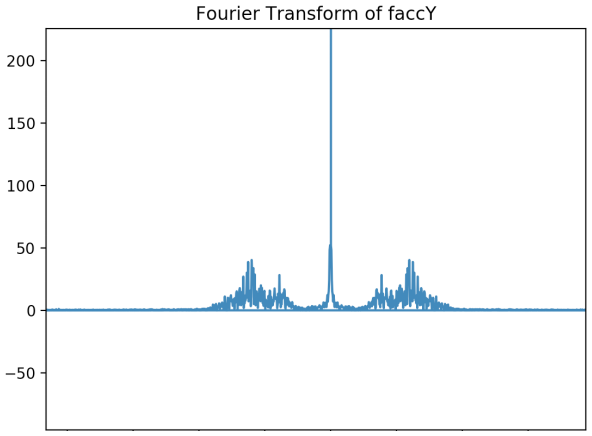
magZ and the magnitude of the Fourier Transform



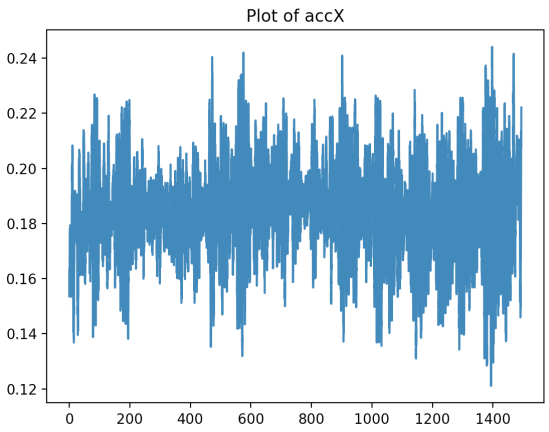


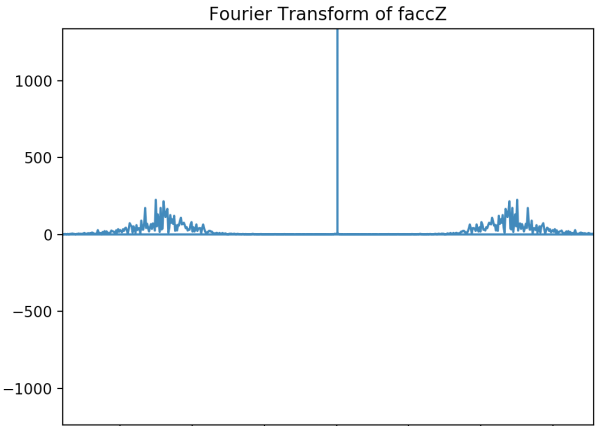
accX and and the magnitude of the Fourier Transform





accY and and the magnitude of the Fourier Transform





accZ and and the magnitude of the Fourier Transform

5.

Validation using a different-IMU

The software needed to be tested on a piece of hardware similar to the specified hardware to test whether it would work in reality as the transfer of working on a similar IMU to the real thing is much easier and one can more confidently say that it will work, than the tests on simulated data.

It is also important to test it on hardware to test the real time capabilities of the software and whether it runs efficiently and quickly enough.

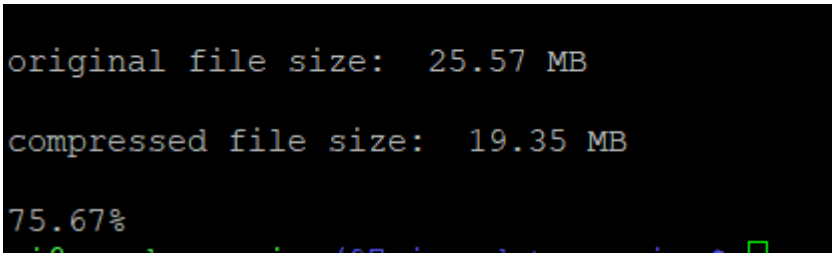
Steps taken to experiment with a different IMU to the one specified, more specifically using the waveshare sensehat are as follows:

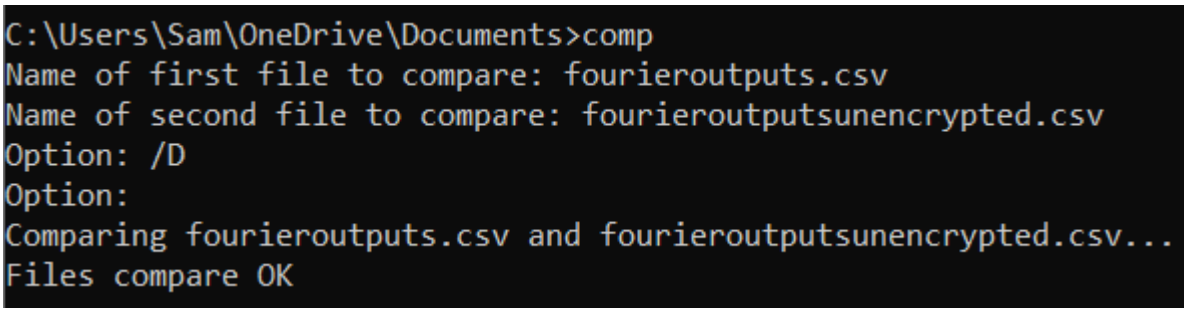
The IMU was tested by continuously outputting its features to the terminal and the sense hat was moved around to detect movement and acceleration.

It was covered with a warm towel and then with an ice pack to check that it detected a temperature change.

The real time IMU data collection was tested by

The main program was run and the sense hat was moved around and exposed to different temperature environments
The output files were checked to be encrypted and compressed and was unencryptred and decompressed to compare that the data was correct.





6.

Consolidation of ATPS

ACCEPTANCE TEST PROCEDURE

Figures of merits, based on which you would validate your final design

- Percentage of Fourier Series Available at output
- Compression Ratio
- Power Usage
- Encryption Strength

Experiment design to test these figures of merit

Feed data with a known output and compare the outputs.
Measure the power usage when completing a full cycle of compression and encryption.
Pass the data though basic encryption cracking algorithms to test the secureness.
Feed the data using a waveshare IMU and check the output.
Compare file sizes before and after compression.

Acceptable performance definition

- The device must be capable of scanning data from the IMU (ICM-20469) and converting it to a readable file
- From input data, the device should be able to produce a Fourier Series
- 25% of the Fourier Series coefficients should be available after compression and encryption
- Device should consume less than 3W of power when processing
- Data must be encrypted
- File size must decrease after compression

Acceptance Test Procedure	**Result**
Percentage Fourier Series at output	Procedure Met(100%)
Capable of scanning and converting IMU data	Procedure Met
Produce Fourier Series	Procedure Met
25% Foureir Series at output	Procedure Met
Less than 3W power consumption	Procedure not tested
Data encrypted	Procedure Met
Compression Ratio	Satisfactory

Encryption Key Strength	Satisfactory
Processing speed	Procedure met (code performs at more than 1 second of data per second)

From the above table, we think that our software is ready to be implemented with the real IMU.

7. Conclusion

Our software successfully runs on the pi0 and manipulates the data input both using the simulated and real time input from the sense hat and outputs the correct files.
All but one ATP has been met.
The compression module works, reducing the file size to an average of 75% the original size.
The encryption module works, sufficiently encrypting the files using the fernet library in python.
Our entire system has achieved full functionality and we believe that it can be implemented very easily in a real world application.

8. References

[1] Euccas.me. 2021. *Understanding zlib*. [online] Available at: <https://www.euccas.me/zlib/> [Accessed 5 September 2021].
[2] En.wikipedia.org. 2021. *Lempel–Ziv–Welch - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch> [Accessed 5 September 2021].
[3] Kodituwakku, S. R., & U, S. A. (2010, December 1). *(PDF) COMPARISON OF LOSSLESS DATA COMPRESSION ALGORITHMS FOR TEXT DATA*. ResearchGate. https://www.researchgate.net/publication/49616246_COMPARISON_OF_LOSSLESS_DATA_COMPRESSION_ALGORITHMS_FOR_TEXT_DATA
[4] Proofpoint. 2021. *What is Encryption? - Definition, Types & More | Proofpoint US*. [online] Available at: <https://www.proofpoint.com/us/threat-reference/encryption> [Accessed 5 September 2021].