EEE3097S PAPER DESIGN

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

Requirements Analysis

User Story

The UCT electrical engineering department in collaboration with the oceanography department is carrying out research on the wave and ice dynamics in Antarctica using smart buoys. These comprise an IMU (ICM20649) which will collect data from the surrounding environment (accelerometer and gyroscope).

Project Requirements

- U1-> The IMU to be used is an ICM-20649. However, in this current project an ICM-20948 must be used, but the IP designed must also work for the ICM-20649.

- U2-> Oceanographers must be able to extract at least 25% of the Fourier Transforms of the data.

- U3-> Reduce the amount of computation done on the processor, by reducing the computations and data sent.

Project Bottlenecks

- Difficulty implementing floating point arithmetic using a fixed point device.
- The Serial Communication of the stm32 does not allow sending of files to the PC, which will make it difficult to get the data from the microcontroller to the laptop.
- The STM32 can only be coded in C, and therefore limits the algorithms that can be implemented on the microcontroller.

Compression algorithms and feasibility

Comparison and feasibility of algorithms

Huffman Compression Algorithm

The Huffman algorithm is a lossless compression algorithm. A variable-length code is assigned to each input character based on the frequency of corresponding characters. The most frequent character gets the smallest code and the least frequent character gets the largest code. The Huffman algorithm is easy to encode and decode, and fast to run, however, the algorithm uses more bits in fixed point variables and therefore might consume more space on the microcontroller.

LZW Compression Algorithm

The LZW algorithm is also a lossless compression algorithm. LZW compression works by reading a sequence of symbols, grouping the symbols into strings, and converting the strings into codes. The code takes up less space than the string it replaces, so compression is achieved. The algorithm is very simple to implement and has a very low compression rate which would save memory on the microcontroller. However, the algorithm is ineffective if there is no duplicate data, and storage requirements are indeterminate since they depend on the total length of all strings.

LZ77 Compression Algorithm

Like the algorithms above, the LZ77 algorithm is lossless. The LZ77 is a dictionary coding technique. This algorithm attempts to replace a string of symbols with a reference to its location in a dictionary. Due to its small window into previously seen text, the LZ77 algorithm continuously throws away valuable dictionary entries by slipping them out of the dictionary. This can be mitigated by increasing the search window, however, this increases the computational time in the CPU and increases the computational usage of the microcontroller.

Burrows-Wheeler Transform Compression Algorithm

Character strings are rearranged by the Burrows-Wheeler transform into runs of similar characters. A string that contains runs of repeated characters can be compressed easily by using techniques such as move-to-front and run-length encoding. However, the BWT takes time to run which might increase the computational power usage in the microcontroller, and it is difficult to implement the algorithm in c programming.

Chosen algorithms for implementation

The Huffman and LS77 compression will be used. The algorithms' ability to be fast, while having a low compression ratio and also being easy to implement, will help reduce the amount of computation done on the processor and increase the efficiency and accuracy of the IP design.

Encryption algorithms and feasibility

Comparison

As micro-controllers have limited resources, the choice of algorithms to use was restricted to block cypher algorithms. As they are more lightweight(use less resources) than stream cypher algorithms. The 4 most block suitable algorithms for the project are SIMON,SPECK,PICCOLO and PRESENT as they are the most compact in hardware and software implementation[1].

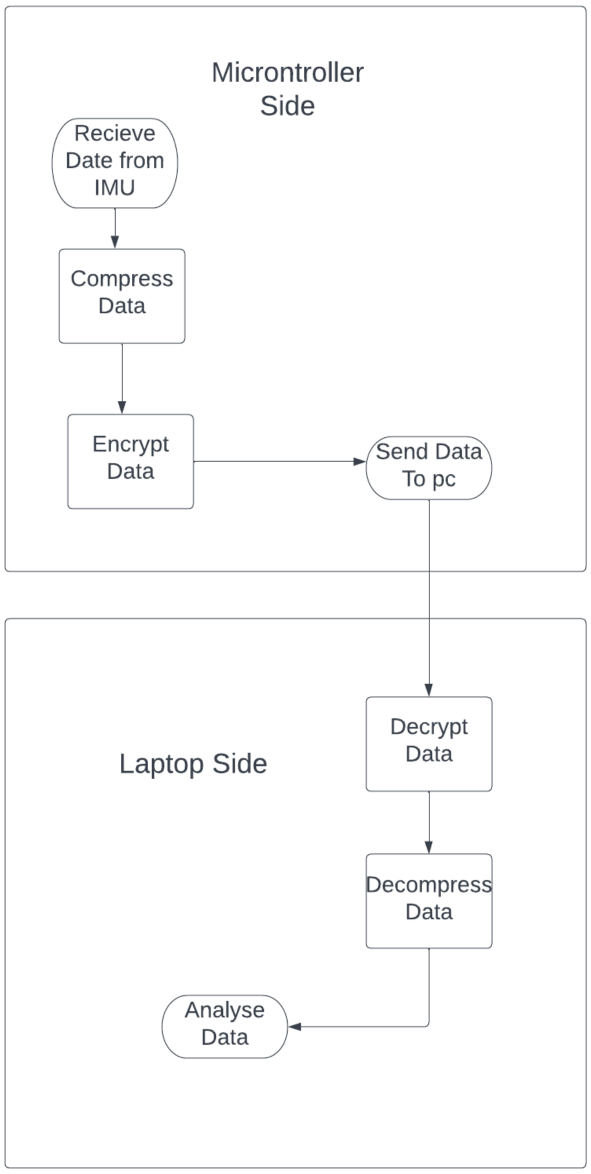
Of the 4 Speck and Simon take up the least physical area (GE<1000) and are the most software efficient(>3500kbps/kB) but are hardware inifficient. Whereas Piccolo is the most hardware and energy efficient(4uJ/bit) however it is not software efficient[2]. Present is the most flexible and common algorithm of the 4 and is the easiest to implement as it is based on AES encryption standard.

Therefore **Present** will be used as it is an SPN block cipher algorithm. SPN is more efficient than FN and GFN algorithms(Simon and piccolo) for the same level of security and energy costs[3]. Also speck will not be used as it is an ARX block cypher so requires more resources.

Feasibility of algorithms

For the design Present(AES) is the most feasible to implement as it has plenty of implementations in different programming language libraries, and more extensive documentation online. Therefore the design process will be simplified as there are plenty of algorithm examples which can be referred to, to aid the design.

Sub-System Designs



Data Flow graph

Compression

Requirements

- RC1 -> The algorithm needs to be fast.
- RC2 -> The algorithm needs to save space.
- RC3 -> Must use low computational power.

Specifications

- SC1 -> The compression algorithm should have a compression ratio of less than 90%.
- SC2 -> Compression time must be less than 90 seconds for a csv file.
- SC3 -> Compressed data must have 25% Fourier Transform of original data.

Encryption

Requirements

- RE1 Encrypt and decrypt data from IMU(ICM20649)
- RE2 Minimal computation (Processes should consume as little power as possible)
- RE3 IP should work for ICM-20649 even though using ICM-20948

Specifications

General:

- SE1 Algorithm used should be fixed point implementation
- SE2 Algorithm must be compatible with a 32-bit ARM processor (STM32F0)

Security:

- SE3 Algorithm must use symmetric key encryption
- SE4 Algorithm must use block cyphers

Performance:

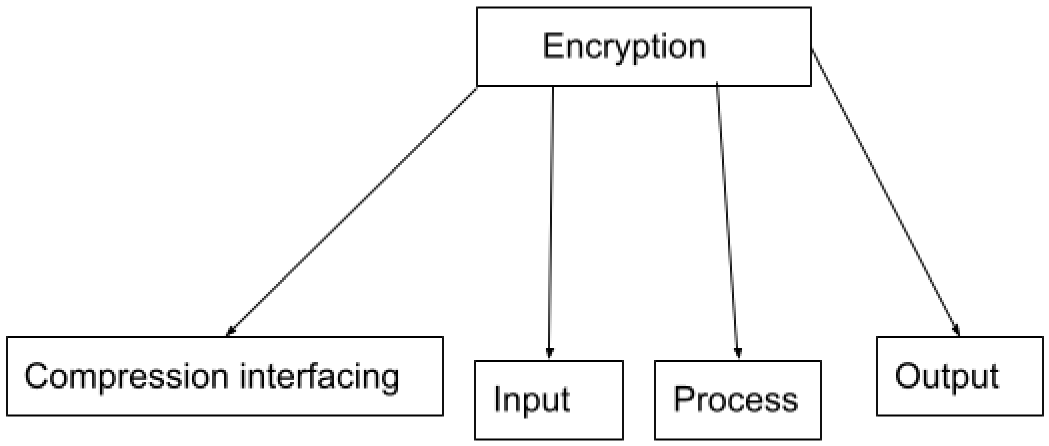
- SE5 Algorithm must have a throughput of 23.7kbps
- SE6 Algorithm must have a latency no more than 10792 cycles per block
- SE7 Algorithm must have an efficiency greater than 50%

Physical:

- SE8 Algorithm must occupy memory less than 8kB RAM, 64kB ROM

Sub-subsystem block diagram

The encryption sub-system algorithm has been broken down into the following sub-subsystems
NB: Decryption has the same sub-subsystems as shown below.



Acceptance Test Procedures

Compression

Figures of Merit

1. ATP-C1 -> When the data is compressed, the compression ratio must be at least under 90%.

2. ATP-C2 -> The compression data should have at least 25% of the Fourier Transform of the data.

3. ATP-C3 -> Compression does not take more than 90 seconds to run.

4. ATP-C4 -> Test if communication between IMU and microcontroller is working.

5. ATP-C5 -> Test if the compressed file is sent from the microcontroller to the PC.

Experiments

1. Compress a file with data, and measure the size of the compressed file to that of the uncompressed data (Size of Compressed / Size of Uncompressed * 100).
2. Plot Fourier Transforms for the compressed data and that of the original data. Then compare the FFTs.
3. Measure the execution time when compressing a file.
4. Read data from IMU and print on the terminal.
5. Send a compressed file from the microcontroller and check it is received on the PC.

Encryption

Figures of Merit

Security of algorithm
Performance of algorithm
Physical cost of the algorithm

Experiments

ATP2.1:Simulation of Algorithm
The program(encryption and decryption) will be run on a PC using provided test data from csv files.
Security:
Visual inspection will be made after encryption to see if the plain text has been converted to cipher text. We can only test security by checking if data has been encrypted. Evaluating how secure the algorithm is, requires vigorous statistical analysis which is outside the scope of this project.

Performance:
This will be tested by determining the throughput, latency, and efficiency of the algorithm. The execution time will be measured and the latency and throughput will be calculated using the formulas

Latency = $clockCycles/block\ size$ [4]

Throughput = text Processed/clock Cycle @ 4MHz [5]

Efficiency = ThroughPut/Code size [6]

ATP2.2:Applying algorithm with actual sensor data(IMU data) using STM32 discovery
Sensor data will be passed to the STM32 and the encryption program will run on the STM.
The output will then be sent to a computer for decryption and the resulting data will be analyzed for any discrepancies from the original.

The performance and security will be assessed in the same way as the simulation. The physical cost will be assessed by ensuring that the size of the program is not too large.

Acceptable performance definition

Algorithm performance will be deemed acceptable if the results obtained in all the experiments meet the above specifications for the sub-system. This will be done by using the above figures of merit to compare the specifications with the test results obtained.

Overall design

Figures of Merit

Size of the decompressed file
Data content of the decrypted file

Experiments

ATP3.1:Testing the overall(both compression and encryption) programs on PC only (simulation) using test data.

ATP3.2:Test the overall programs on the STM32 using actual sensor data.
Sensor data will be passed to the STM32 and the encryption and compression programs will run on the STM.
The output will then be sent to a computer for decryption and decompression and the resulting data will be analyzed for any discrepancies from the original.

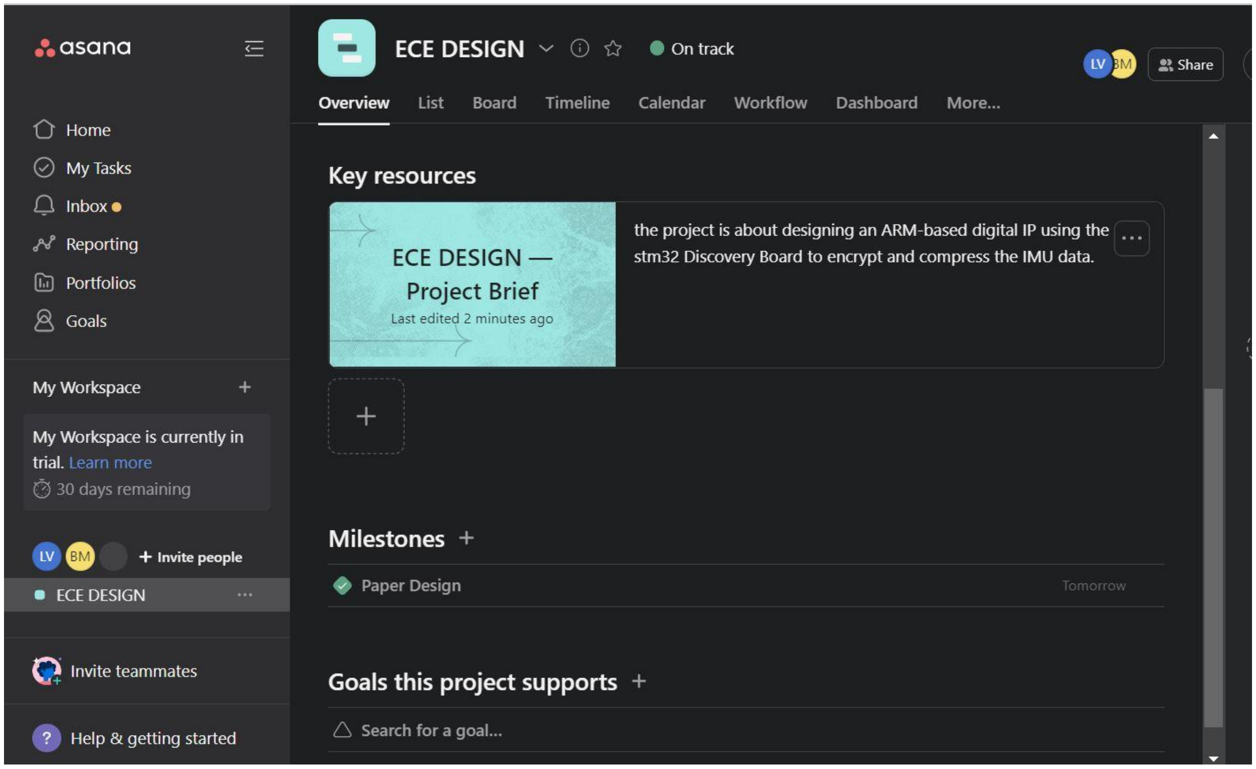
Acceptable performance definition

Overall system will be considered acceptably performed if all the original data is recovered after decryption and decompression. The size of the data before compression and after decompression should be the same, and the data content in plain text after decryption should appear the same as before encryption.

Traceability Matrix

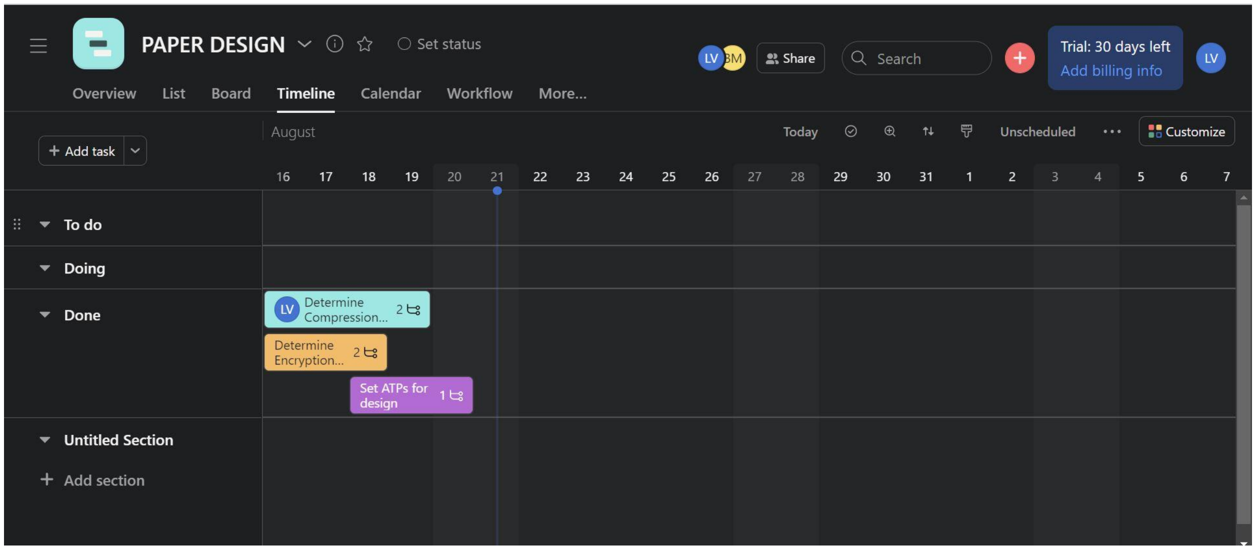
User requirements	Functional requirements	Specification	ATP
U1	RE3	SE1-SE4	ATP-C4
U2		SC3	ATP3.1, ATP3.2, ATP-C2
U3	RE1, RE2, RC1-RC3	SE5-SE8, SC1-SC2	ATP2.1, ATP2.2, ATP-C1, ATP-C3

TimeLine and PM Page



the project is about designing an ARM-based digital IP using the stm32 Discovery Board to encrypt and compress the IMU data.

Project Management page



Project Timeline

References

[1] [2] [3] [4] [5] [6] V. A. Thakor, M. A. Razzaque and M. R. A. Khandaker, "Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities," in IEEE Access, vol. 9, pp. 28177-28193, 2021, doi: 10.1109/ACCESS.2021.3052867.

“Db2 12 - Performance - Compressing Your Data.” Www.ibm.com, 1 Jan. 2019, www.ibm.com/docs/en/db2-for-zos/12?topic=performance-compressing-your-data . Accessed 22 Aug. 2022.

“Huffman Coding | Greedy Algo-3.” GeeksforGeeks, 3 Nov. 2012, www.geeksforgeeks.org/huffman-coding-greedy-algo-3/.

“Lossless Data Compression: LZ77.” Cs.stanford.edu, https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/lz77/shortcoming.htm#:~:text=Lossless%20Data%20Compression%3A%20LZ77&text=One%20of%20the%20main%20limitations,slide%20out%20of%20the%20dictionary. Accessed 22 Aug. 2022.

“LZW (Lempel–Ziv–Welch) Compression Technique - GeeksforGeeks.” GeeksforGeeks, 26 Apr. 2017, www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/.LZW /COMPRESSION and DECOMPRESSION. 2015.

“The Huffman Encoding.” Emory.edu, 2022, www.cs.emory.edu/~cheung/Courses/253/Syllabus/Compression/Huffman.html.Wikipedia Contributors.

“Burrows–Wheeler Transform.” Wikipedia, Wikimedia Foundation, 9 Nov. 2020, Accessed 22 Aug. 2022. en.wikipedia.org/wiki/Burrows%E2%80%93Wheeler_transform.---

“Lempel–Ziv–Storer–Szymanski.” Wikipedia, Wikimedia Foundation, 7 June 2022, en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Storer%E2%80%93Szymansk i.