



EEE3097S Paper Design
Group 3
Shameera Cassim and Tristyn Ferreira
CSSSHA020, FRRTRI001

5 September 2022

1 Introduction

1.1 Scope

This project deals with implementation of encryption and compression of data from sensors on the SHARC buoy system [1]. This project considers different compression algorithms to apply to the data. Different types of encryption and encryption algorithms to apply to the compressed data are also considered. Organised storage of the data and satellite transmission fall outside of the scope of this project. This project aims to provide a solution to the data size problems highlighted in the SHARC Buoy design by Jamie Nicholas Jacobson.

1.2 Limitations

The system must be designed to use data from the ICM-20649 IMU however at the time of this project, it is not available. Instead the ICM-20948 is used for the duration of this project and design decisions take ICM-20649 compatibility into account.

Additionally, at the time of this project, Raspberry Pis (floating point processors) are not available for use in development and testing. Instead an STM32F0 with a fixed point processor is used, this limits functional testing of the algorithms on a floating point processor architecture. Further, this limits the amount of data that is able to be processed at any time.

Contributions

Task	Completed by:
Introduction: Scope	Both
Introduction: Limitations	Both
Requirements: Traceability Matrix	Tristyn Ferreira
Requirements: User Requirements	Both
Requirements: Functional Requirements	Both
Requirements: Specifications	Shameera Cassim
Requirements: Possible Bottlenecks	Tristyn Ferreira
Requirements: Feasibility Analysis	Shameera Cassim
Subsystem Design: Compression	Tristyn Ferreira
Subsystem Design: Encryption	Shameera Cassim
Subsystem Design: Inter-Subsystem and Inter-Sub-Subsystems Interactions	Both
Acceptance Test Procedures	Both
Development Timeline	Tristyn Ferreira
Bibliography	Both
Research	Both

2 Requirements

Table 1: Requirement Traceability Matrix

UR	FR	Specifications	ATP
UR01	FR01	SP01	AT02
UR02	FR06	SP07	AT07
UR03	FR05	SP04, SP06	
UR04	FR03, FR04	SP05	AT04, AT05
UR05	FR01, FR02	SP02	AT03
UR06	FR07	SP08	AT01

2.1 User Requirements

Table 2: User requirements used to determine the design and functionality of the desired IP

User Requirement	Description
UR01	At least 25% of lower Fourier coefficients must be recoverable after compression
UR02	System must be compatible with ICM-20649 and ICM20948 IMU chips
UR03	System must use minimal computation to be power efficient
UR04	System must encrypt the data
UR05	System must compress data
UR06	System must be able to transmit data to another device

2.1.1 Analysis of UR01

At least 25% of Fourier coefficients must be recoverable after compression.

While some losses due to compression are acceptable, a minimum of 25% of the lower Fourier coefficients of the data must be recoverable after compression.

2.1.2 Analysis of UR02

System must be compatible with ICM-20649 and ICM-20948 IMU chips

The final system should work with an ICM-20649 IMU chip. However, at the time of this project, the ICM-20649 is not available and so the ICM-20948 IMU chip will be used for testing and the duration of this project. The solution should still be relevant for the ICM-20649 and its data.

2.1.3 Analysis of UR03

System must use minimal computation to be power efficient

The system, for which the IP is designed, has a limited source of power. The system needs to conserve energy in order to survive once deployed [1]. Therefore, the IP needs to be computationally inexpensive.

2.1.4 Analysis of UR04

System must encrypt the data

The data will be sent over the internet and so requires a layer of encryption to protect the contents.

The data is not highly sensitive and is unlikely to be hacked so encryption does not need to be complex. The algorithm used needs to be lightweight as discussed in UR03.

2.1.5 Analysis of UR05

System must compress the data

The data gathered on the SHARC buoy will be transmitted using Iridium satellite connection, this is expensive and so the number of transmissions made needs to be minimised. Additionally, number of packets to be transmitted needs to be minimised in order to preserve battery life [1] as discussed in UR03.

2.1.6 Analysis of UR06

System must be able to transfer data to another device for decryption and decompression

When the SHARC bouy is deployed it will transmit the data via Iridium satellite networks [1] however, this transmission process is outside the scope of this project. For this project, the data needs to be transferred following encryption to another device to be decrypted and decompressed.

2.2 Functional Requirements

Analysis of the user requirements resulted in the following functional requirements which describe how the system will function.

Table 3: Requirements addressing the needs of the system

Requirement	Description
FR01	System to compress data such that 25% of the lower Fourier coefficients are recoverable after decompression
FR02	Compression subsystem to pass processed data to the encryption subsystem
FR03	All compressed data must be encrypted
FR04	Encryption subsystem to encrypt data in a manner that allows efficient and complete decryption
FR05	System must use minimal computation and processing
FR06	All code must be implementable for a system using either ICM-20649 or ICM-20948
FR07	System to transmit data via USB to another device for decryption and decompression

2.3 Specifications

Table 4: User requirements used to determine the design and functionality of the desired IP

Specification ID	Description
SP01	A minimum of 25% of the lower Fourier coefficients must be present after decompression
SP02	The compression ratio must be greater than 1, i.e. the compressed file must be smaller than the original file.
SP03	100% of the compressed data must be recovered after decryption.
SP04	Less than 20mAh of power must be used in encrypting the data.
SP05	Less than 10% of the original data can be the same as the original file after encryption.
SP06	All code should be written in C
SPO7	I2C must be run on fast mode when developing and testing (ICM-20649 standard)
SP08	System must transfer the data over UART following encryption

2.4 Possible Bottlenecks

Data is being transferred and processed throughout the system, key bottlenecks in the process are identified as:

2.4.1 I2C communication lines

Between the sensor and the STM32: This connection is made using I2C which is a two-wire protocol. In order to transmit over I2C the voltage of the wire needs to be driven high and low using a resistor. Additionally the wires have internal capacitance which slows down the process. This would cause a bottleneck between the flow of data in and the compression algorithm.

2.4.2 Compression and Encryption

Depending on the encryption algorithm used, there is potential for a bottleneck between compression and encryption of the data. If the encryption is significantly faster than the compression algorithm, there will be idle time where the encryption waits for the compression to complete.

2.4.3 STM32F051

The STM32F051 has limited RAM (8 Bytes) and the tasks of reading data from the HAT, processing the data and then transferring it to the computer are process heavy. Ultimately the STM32F051 will be a bottleneck in the entire system.

2.5 Feasibility Analysis

Table 5: Feasibility Analysis

Specification ID	Level of Feasibility	Reasoning
SP01	HIGH	There are many lossless and low loss compression algorithms that exist and could be used
SP02	HIGH	There are many compression algorithms that can compress files much smaller than 30% of the original size that could be implemented
SP03	HIGH	Provided the key is correct, 100% of the decrypted data should match the original data
SP04	MED	The encryption algorithms to be tested have an expected battery drainage of between 3mAh and 7mAh on similar architectures
SP05	HIGH	Provided the encryption algorithm is implemented correctly, very little of the original data should match the cipher text in the encrypted file
SP06	HIGH	Unless there are unforeseen circumstances that require otherwise, all code will be written in C
SP07	HIGH	Unless there are unforeseen circumstances that require otherwise, I2C will be run on fast mode when developing and testing
SP08	MED	While this is a key specification, it is unclear whether we will have access to an FTDI or another method of transferring encrypted data over UART

3 Subsystem Design

3.1 Compression

3.1.1 Requirements & Specifications

This subsystem must meet the requirements set out in FR01, FR02 and FR05 and the specifications set out in SP01, SP02 and SP06 as described above.

3.1.2 Algorithm Comparisons

Three algorithm types are considered namely Delta, Delta-on-Delta and Dictionary compression:

Delta compression only stores the difference between an object and a reference object, this reduces the amount of information needed to represent a data object. This compression works best with numerical data that represents something slowly changing over time. [2] The data recorded by the IMU is numerical and slowly changes over time. This is a suitable compression algorithm.

Delta-of-delta compression applies a second layer of Delta compression to the Delta-compressed data [2]. This further decreases the size of the data however it increases the compression time and it increases the source code. Since the STM32F051 used for testing has limited memory, this algorithm takes up valuable space needed for data.

Dictionary compression makes a list of the possible values that can appear in the data and then stores an index to the dictionary containing the unique values. This works well when there is repeating data. [2]. The LZSS compression is dictionary based.

The algorithm chosen must be small as the memory available on the STM32F051 is limited. This significantly reduces the options available:

Algorithm	Compression Ratio	Source code size
Delta Compression	1.48	9kB
LZSS	1.41	8kB
LZW	1.42	36kB

Figure 1: Comparison of Compression Algorithms [3], [4]

3.2 Encryption

3.2.1 Requirements & Specifications

This subsystem must meet the requirements set out in FR03, FR04 and FR05 and the specifications set out in SP03, SP04, SP05 and SP06 as described above.

3.2.2 Algorithm Comparisons

There are 2 main classes of encryption algorithms, symmetric and asymmetric. Symmetric algorithms have a single key for encryption and decryption while asymmetric algorithms have a public encryption key and a private decryption key. [5] This means that asymmetric algorithms are generally more secure. Given that the data for this specific application is not highly sensitive, using an asymmetric algorithm is not necessary. Further, symmetric algorithms have lower resource utilization and are typically much faster than asymmetric algorithms [6] which is necessary for this application. For those reasons, the algorithms considered below are all symmetric.

The algorithm chosen heavily impacts the battery drainage caused by encryption. This is illustrated in the table below:

Algorithm Type	Avg encryption throughput (MiB/s)			Battery drainage (mAh)
	1 MiB	5 MiB	10MiB	
ChaCha20-Poly1305	36.805	38.77	38.951	3.9
Twofish	16.593	17.258	17.309	8.77
AES	11.073	11.286	11.313	13.40
RC6	18.738	17.237	17.290	8.68
SPECK128	22.776	23.518	23.847	6.32

Figure 2: Comparison of Encryption Algorithms

This table was put together using data from tables 2 and 3 in the article PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices by Saraiva et al. [7]. Due to the low battery drainage, STM32 libraries and other source code currently available, ChaCha20-Poly1305 and RC6 will be tested for the purpose of this project.

3.3 Inter-Subsystem and Inter-Sub-subsystems Interactions

The data passed into the system will be compressed and then encrypted before it is passed out of the system. In order to use this data, it must be decrypted and then decompressed. This is illustrated in the diagram below:

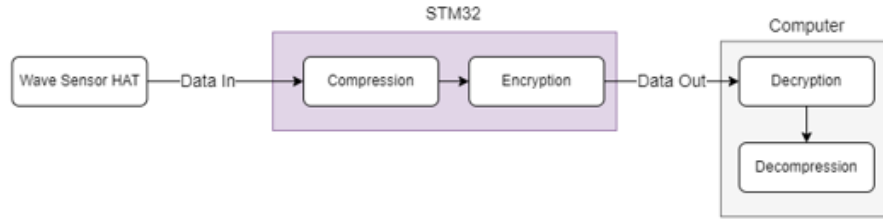


Figure 3: Diagram of Inter-Subsystem interactions

However, for the development and testing phases of this project, the system will be setup as follows:

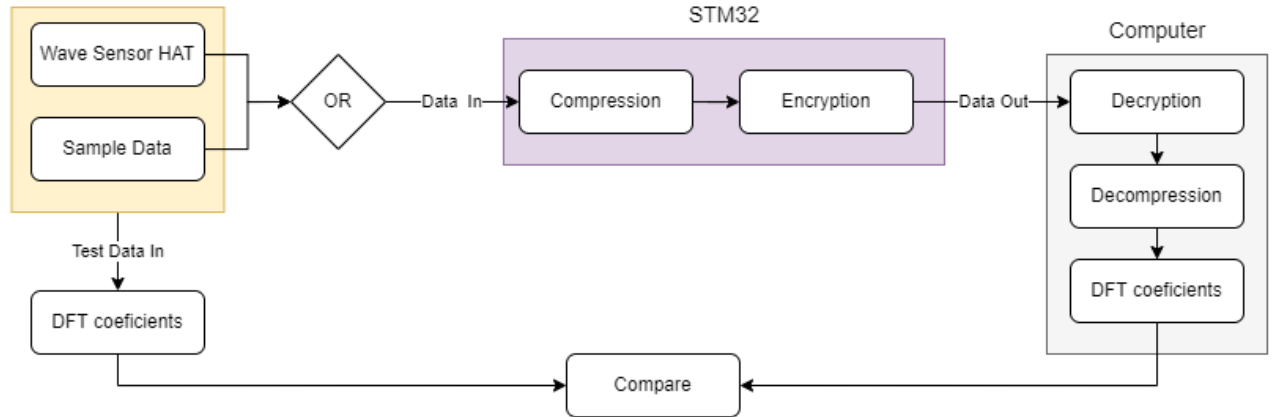


Figure 4: Diagram of Inter-Subsystem interactions - Testing and Development

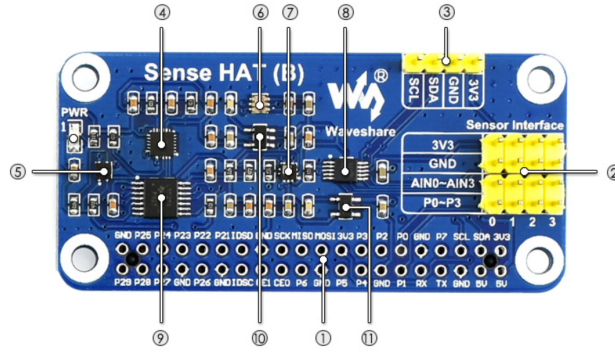
3.3.1 Interfacing

The subsystems need to be connected physically for testing and will be interfaced as below:

Table 6: Requirement Traceability Matrix

Interface	Description	Pins/Output
I001	Waveshare Sense HAT (B) to STM for data transfer	<ul style="list-style-type: none"> • SCL: HAT SCL* to STM PB10 • SDA: HAT SDA* to STM PB11 • Power: HAT 3V3* to STM VDD • GND: HAT GND* to STM VSS
I002	STM to FTDI for data transfer to computer	<ul style="list-style-type: none"> • STM PA10(RX) to FTDI TX • STM PA9(TX) to FTDI RX
I003	FTDI to computer for data transfer from STM	<ul style="list-style-type: none"> • Assuming the FTDI is connected to a Micro B USB port (which can be used to connect to a USB cable to connect to a computer), output via cable

*referring to pins labeled '3' on the diagram below: [8]



4 Acceptance Test Procedures

Table 7: User requirements used to determine the design and functionality of the desired IP

Acceptance test	System	Description
AT01	Input	Ensure data is transferred correctly and completely from computer to STM32F051
AT02	Compression	Ensure that at least 25% of lower Fourier coefficients of data are preserved
AT03	Compression	Ensure file size is decreased after compression
AT04	Encryption	Ensure that the data is not readable without the key after encryption
AT05	Encryption	Data is readable and the same as the original once decrypted
AT06	Full System (no HAT)	Ensure compression and encryption works when implemented together
AT07	Full system	Ensure that data read from the sensor completes a full cycle meeting all requirements and above ATPs

4.1 AT01

AT01	Connection Test
Evaluation type	Comparison
Target	STM32F051
Test Protocol	<ol style="list-style-type: none">1. Set up test case using a sample set of data2. Pass data from device to STM32F0513. Pass data from STM32F051 to computer.4. Compare data read and passed to computer to original sample set of data.
Pass Condition	<ul style="list-style-type: none">• Data is in the same format as original data• Data values are the same as original data• All original data is present in the new data
Fail condition	<ul style="list-style-type: none">• Data format differs from original• Any data value is different from the original data• Any data values are missing Extra data values are present in the new data

4.2 AT02

AT02	Fourier Coefficients Test
Evaluation type	Comparison
Target	Compression Subsystem
Test Protocol	<ol style="list-style-type: none">1. Set up test case using a sample set of data and take 25% the lower Fourier coefficients.2. Compress the original data using the compression algorithm on the STM32F051.3. Decompress the data and compute the Fourier coefficients on a computer.4. Compare the Fourier coefficients to the test case.
Pass Condition	<ul style="list-style-type: none">• All of the Fourier coefficients present in test case are present in decompressed data (extra coefficients may be present but not less).
Fail condition	<ul style="list-style-type: none">• Fourier coefficients present in the test case are not present in the decompressed data.

4.3 AT03

AT03	Compression Ratio Test
Evaluation type	Comparison
Target	Compression Subsystem
Test Protocol	<ol style="list-style-type: none">1. Set up test cases using sample set of data and make note of file sizes.2. Compress the original data using the compression algorithm on a computer.3. Compare the size of the new file to that of the original file.
Pass Condition	<ul style="list-style-type: none">• New file size is less than original file size.
Fail condition	<ul style="list-style-type: none">• New file size is greater than or equal to old file size.

4.4 AT04

AT04	Encryption key Test
Evaluation type	Algorithm Validation
Target	Encryption Subsystem
Test Protocol	<ol style="list-style-type: none">1. Set up test cases using sample set of data.2. Encrypt data using the encryption algorithm and key on the STM32F051.3. Check if data can easily be deduced or read after encryption on a computer.
Pass Condition	<ul style="list-style-type: none">• Data is not deducible or human readable after encryption.
Fail condition	<ul style="list-style-type: none">• Data can easily be deduced or read after encryption.

4.5 AT05

AT05	Decryption validation Test
Evaluation type	Algorithm Validation
Target	Encryption Subsystem
Test Protocol	<ol style="list-style-type: none">1. Set up test cases using sample set of data.2. Encrypt data using the encryption algorithm and key on the STM32F051.3. Decrypt data using the decryption algorithm and key on a computer.4. Compare new data to original data set.
Pass Condition	<ul style="list-style-type: none">• Decrypted data is identical to original sample set data.
Fail condition	<ul style="list-style-type: none">• There are differences between sample set data and decrypted data.

4.6 AT06

AT06	System Test
Evaluation type	System execution
Target	Full System (no HAT)
Test Protocol	<ol style="list-style-type: none"> 1. Set up test case using sample data set and compute 25% of the lower Fourier coefficients. 2. Pass sample data through compression and then encryption algorithm on the STM32F051. 3. Perform decryption and then decompression on the computer. 4. Compute 25% of lower Fourier coefficients of the data. 5. Compare the coefficients to the test case.
Pass Condition	<ul style="list-style-type: none"> • Encryption and decryption complete without failure. • AT01 is met.
Fail condition	<ul style="list-style-type: none"> • Encryption or decryption fail. • AT01 is not met.

4.7 AT07

AT07	System Test with Wave sensor HAT
Evaluation type	System Execution
Target	Full System
Test Protocol	<p>The Wave sensor HAT is connected to STM32F051. The data from the HAT is first printed to the computer and then sent to be compressed (onboard the STM32F051).</p> <ol style="list-style-type: none"> 1. The data printed to the computer is used as the sample data set and 25% of the lower Fourier coefficients are computed. 2. Data is sent through compression and encryption on the STM32F051 and then transferred to the computer. 3. Perform decryption and then decompression on the computer. 4. Compute 25% of lower Fourier coefficients of the data. 5. Compare the coefficients to the test case.
Pass Condition	<ul style="list-style-type: none"> • Data is successfully read from the HAT and printed to the computer. • AT06 is met
Fail condition	<ul style="list-style-type: none"> • Data is not read from the HAT (there is not change in data or no data printed to the computer) AT06 is not met

1 Development Timeline


















Task name	Assignee	Due date
▼ To do		
▶  Paper Design Submission : Collate everything 6 𐌶		Tomorrow
▶  DFT script 3 𐌶		Aug 22 – Sep 4
▶  Encryption Algos: setup & test 4 𐌶	 Shameera	Aug 22 – Sep 7
▶  Compression Algos: Setup & test 3 𐌶	 Tristyn Ferre...	Aug 22 – Sep 7
 First Progress Report (Simulation)		Sep 12
▶  Encryption Algos: implement on STM 1 𐌶	 Shameera	Sep 8 – 24
▶  Compression Algos: implement on STM 1 𐌶	 Tristyn Ferre...	Sep 8 – 24
 Decide on best algorithm combination		Sep 24 – 30
▶  Interface with sensor HAT 1 𐌶		Sep 28 – Oct 1
 Second Progress Report (Practical)		Sep 30
 Run a full cycle		Oct 1 – 5
 Final Report Submission		Oct 5
 Demo Video		Oct 12

Figure 5: Screenshot of task list

The timeline shows a break down of milestones and suggested tasks for the remainder of the project. Some tasks have sub-tasks that need to be completed before the main task can be marked as completed. More tasks will be added as needed. Tasks not assigned to a specific person are either joint tasks or will be assigned in future.

References

- [1] J. N. Jacobson, “Sharc buoy: Robust firmware design for a novel, low-cost autonomous platform for the antarctic marginal ice zone in the southern ocean,” Ph.D. dissertation, Rondebosch, Cape Town South Africa, 2021.
- [2] Joshua Lockerman, Ajay Kulkarni, “Time-Series Compression Algorithms, Explained,” [Online]. Available: <https://www.timescale.com/blog/time-series-compression-algorithms-explained/>, [Accessed: 22 August 2022].
- [3] , “IzBench,” [Online]. Available: <https://github.com/inikep/lzbench>, [Accessed: 22 August 2022].
- [4] Glenn Fiedler, “delta compression,” [Online]. Available: <https://github.com/inikep/lzbench>, [Accessed: 22 August 2022].
- [5] Cloudflare, “What is encryption? — Types of encryption,” [Online]. Available: <https://www.cloudflare.com/learning/ssl/what-is-encryption/#:~:text=An%20encryption%20algorithm%20is%20the,by%20using%20the%20decryption%20key>, [Accessed: 19 August 2022].
- [6] Abhishek Tiwari, “Difference Between Symmetric and Asymmetric Key Encryption,” [Online]. Available: <https://www.geeksforgeeks.org/difference-between-symmetric-and-asymmetric-key-encryption/>, [Accessed: 19 August 2022].
- [7] D. A. F. Saraiva, V. R. Q. Leithardt, D. de Paula, A. Sales Mendes, G. V. González, and P. Crocker, “Prisec: Comparison of symmetric key algorithms for iot devices,” *Sensors*, vol. 19, no. 19, 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/19/4312>
- [8] Waveshare, “Sense HAT (B) for Raspberry Pi, Multi Powerful Sensors,” [Online]. Available: <https://www.waveshare.com/sense-hat-b.htm>, [Accessed: 17 August 2022].