Mälardalen University
School of Innovation Design and Engineering
Västerås, Sweden

Thesis for the Degree of Master of Science in Computer Sciences with Specialization in Embedded Systems 15.0 credits

BATTERY SENSORY DATA COMPRESSION FOR ULTRA NARROW BANDWIDTH IOT PROTOCOLS

Ahmed Karic
akc17002@student.mdh.se
Ivan Loncar
ilr17002@student.mdh.se

Examiner: Moris Behnam
Mälardalen University, Västerås, Sweden

Supervisor: Tran Hung
Mälardalen University, Västerås, Sweden

Company supervisor: Ralf Strömberg,
ADDIVA AB, Västerås, Sweden

June 11, 2018

Abstract

Internet of Things (IoT) communication technology is an essential parameter in modern embedded systems. Demand for data throughput drastically increases, as well as the request for transmission over considerable distance. Considering cost-efficiency in the form of power consumption is unavoidable, it usually requires numerous optimization and trade-offs. This research tends to offer a solution based on data compression techniques. In this way, problems caused by data throughput are mostly eliminated, still varying with the field of application. Regardless of having both lossy and lossless techniques, the focus is on lossy algorithms due to immensely larger compression ratio (CR) factor, which is not the only but usually the most crucial factor. There are also numerous other quality metrics described. In the experiment part, LoRa long-range wireless communication protocol is used, with an accent on battery sensory data transmission. Temperature and current are the signals of interest. The research offers detailed information of the impact on compression parameters by four target algorithms: fractal resampling (FR), critical aperture (CA), fast Fourier transform (FFT) and discrete cosine transform (DCT).

List of plots

Tables

Acknowledgements

We take this opportunity to express our gratitude to the company ADDIVA ELEKTRONIK AB for offering an opportunity to perform a research in a wonderful environment. Addiva supplemented us with all necessary equipment and offered a support by great staff. We hereby thank to Peter, Ralph, Nills, Oly, Rasmus and Madalen. We also owe a huge gratitude to our supervisor Tran Hung for unselfish dedication to our work and support in the report.

Acronyms

IoT Internet of Things

CR Compression Ratio

FR Fractal Resampling

CA Critical Aperture

FFT Fast Fourier Transformation

DCT Discrete Cosine Transformation

RTOS Real Time Operating System

IDE Integrated Development Environment

RT Real Time

CPU Central Processing Unit

MAC Media Access Protocol

OSI Open System Interconnection

ISM Industrial, Scientific and Medical

CH Chebyshev transform

WPD Wavelet Packet Decomposition

ECG Electrocardiogram

PPG Photoplethysmogram

DWT Discrete Wavelet Transform

FWHT Fast Walsh-Hadamard Transform

LDE Lossy Delta Encoding

ML Machine Learning

ADC Analog to Digital Conversion

MWC Modulated Wideband Converter

ASEC Analysis by Synthesis Coding

RMS Root Mean Square

PRD Percentage RMS Difference

WDD Weighted Diagnostic Distortion

DFT Discrete Fourier Transform

SVD Singular Value Decomposition

PAA Piecewise Aggregate Approximation

APCA Adaptive Piecewise Constant Approximation

PLA Piecewise-Linear Approximation

VLSI Very Large Scale Integration

DSP Digital Signal Processing

PCA Principal Component Analysis

MCU Microcontroller Unit

SoC State of Charge

IC Integrated Circuits

NTC Negative Temperature Coefficient

PTC Positive Temperature Coefficient

RLE Run Length Encoding

BS Backward Slope

BC Box Car

CF Curve Fitting

NASA National Aeronautics and Space Administration

STD Normalized Standard Deviation

TTN The Things Network

PCF Power Consumption Factor

Table of Contents

1 Introduction

In the world of the IoT technology, demand for low power - long range sensory data transmission is rapidly growing. "In IoT, numerous and diverse types of sensors generate a plethora of data that needs to be stored and processed with minimum loss of information "[1]. Networked embedded devices are generating a vast amount of data that need to be sent to the cloud, and then processed and stored. Application areas are mainly related to transportation, medical, health-care, military and agriculture. Some of the applications, like car battery monitoring and smart farms, require the data to be sent over a long distance with some power restriction. This thesis propose an analysis of a battery management system concerning all essential parameters, which can be processed and controlled to prolong battery life.

Data is created in the embedded networked devices, or more commonly known as IoT devices usually, need to be analyzed by a remote server computer. However, communication problems arise on different stages, mostly on end-point-devices, where all the information is gathered. The data is supposed to be frequently sent via long range-low power wireless network, low bandwidth limits that. These types of network can fail in sending the data, or in some cases succeed, but with lower transmission rates. Such data would be stored in the memory of the end-device until it is getting resent. Considering the worst cases, the memory may get saturated in some instance. Problem with the lack of memory could be solved using data compression technique. Regardless of the presence and accessibility of the future communication protocols, the data compression process will be an inevitable and unavoidable factor.

Depending on the use-case of networked embedded devices different IoT protocols can be utilized. Low power-long range protocols are needed for the energy efficient devices that depend on batteries and are transmitting data over long distances, which may be several kilometers. Pure representatives of wireless IoT protocols, which are long range and low power are *Ingenu*, *LoRa*, *SigFox*, listed in the article [2], as the examples of company developed, *Symphony* based on LoRa physical layer, and *Weightless* as examples of open source protocols. *Ingenu* protocol is not universal in a sense it is only situated in the North American territory and proposed for local needs. Thus it will not be evaluated in this research. Open source variants of wireless protocols, mentioned above, also will not be considered due to poor support from the community and not well developed. To the best of our knowledge, *SigFox* and *LoRa* are feasible for protocols that will be analyzed and compared for different purposes in implementation and the research.

2 Problem Formulation

The core problematic revolves around the comparison and analysis of specific data compression algorithms for optimizing of battery system features. Essential feature constraints are put on power consumption, real time performance, memory and processor demand.

At the very beginning, it is necessary to analyze sampling rate of multiple sensory data, to evaluate compression algorithm complexity, communication properties,

as well as the selection of real time operating system (RTOS), to determine which microcontroller and central processing unit (CPU) is best for the specific requirements.

Many different compression algorithms will be tested and compared. Desired comparison properties of the algorithms are mainly related to the algorithm execution complexity, losses, level of compression and need for memory extension. Hence, the following research questions are in the primary focus of the thesis:
RQ1: What is the CR factor of different compression algorithms applied on battery data?
RQ2: What is the effect of specific compression algorithms on memory capacity and power consumption?

3 Method

To achieve an effective research on the topic, *case study* is employed in the first half of the research, where multiple papers will be reviewed. Numerous compression techniques, applied in different areas, are analyzed in order to acquire the most suitable ones for the target application.

After the analysis, appropriate algorithms are selected and tested in an experiment setup. The *experiment* is obtained to give answers to RQ2. It was mainly based on observing current behaviour in the transmitting node. Thus the research is represented as a combination of case study method with the experiment.

Research process is obtained through the following steps:

1 - Analysis of selected data compression algorithm.

2 - Select the most suitable algorithms for the field of application. It is not mandatory to apply only existing compression algorithm. The research will be supplemented by some of the well-known numerical methods or different approaches in data reduction techniques.

3 - Analysis of all essential parameters for development environment: a processing unit, development board and RTOS . The parameters that will be considered are the memory, power consumption, clock frequency, the case of implementation and integrated development environment (IDE).

4 - System setup: setting Lora network, hardware and software setup, testing the system. To attain valid results, signal measurements on current, voltage and temperature data will be measured and stored in a file on a personal computer. The data is used for testing, to have reproducible and consistent results.

5 - Implementation and testing of selected compression algorithms. Tests compare algorithms power consumption, real time (RT)[1] capabilities and compression ratio (CR), with respect to error margin for lossy compression.

[1]Real time property of an algorithm represents its capability to obtain compression during sampling process (run-time). The algorithm should not wait for a data set to be entirely accumulated.

6 - Analyzing and comparing the results based on different metrics to determine the effectiveness of the data reduction techniques and providing a significant suggestion.

4 Background

In the field of transportation, battery technology is known as future energy system solutions. There are many vehicle types powered by battery technology. To prolong battery lifetime, sustainable maintenance is required, and accurate monitoring of energy measurements can achieve this. Importantly, we need to measure and control battery electrical energy consumption, regarding current, voltage, and temperature, as important battery-life parameters. To have access and to be able to manipulate with the battery parameters, it is necessary to make them available on the servers via the appropriate wireless communication protocol. The communication should contain an end-device, also called node, a low power device that performs data acquisition. The data are further sent to the antenna, called gateway, that receives information from the node, and can feedback the data to the node. It is necessary to keep in mind that the *end-device* must belong to some of the three possible end-node classes: A, B or C. Detailed information about the classes are available on [3]. An *end-device*, belonging to any of the classes, must support bi-directional communication with the *gateway* [3]. In the implementation part (see chapter *System setup)*, the role of the *node-device* is delivered to AVR32's EVK1100 board, which for the experimental part of the research will not include any sensor, due to confidentiality reasons.

Particular wireless protocols provide the low power transmission of data over long distance (up to 15 kilometers), and their well-known representatives are LoRa and SigFox. The protocols fit the description of possible applications, but on the other hand, have huge limitations on data throughput.

LoRaWAN is a low power-long range media access protocol (MAC), which belong to the second (data-link) and the third (network) layer of Open Systems Interconnection (OSI) reference model [4]. Its operating frequency varies with location. It operates in 863-870 MHz in Europe, US 902-928 MHz, China 779-787 and 470-510 MHz. Data bandwidth of the protocol rises to maximal 500 kHz, still depending on the region of the application. Its modulation is based on Chirp [2] technology[5], which ensures the communication to work properly and immune to noise and Doppler effect even at low power consumption. In the US, LoRa has dedicated uplink and downlink channels. Eight uplink channels are operating on 125 kHz, one 500 kHz uplink channel and one 500 kHz down-link channel[3].

Its first concurrent, SigFox, has also established wide application area, being part of *Industrial, Scientific and Medical* (ISM) radio band. It operates on 868 MHz in Europe and 902 MHz in the US. The network has a star-topology and ability to cover a huge area. "Sigfox operate in the 200 kHz of the publicly available band to exchange radio messages over the air. Each message is 100 Hz wide and transferred

[2]CHIRP (Compressed High-Intensity Radiated Pulse) is a technology that has been used in military applications, such as in radars and sonars.

at 100 or 600 bits per second a data rate, depending on the region. Hence, long distances can be achieved while being very robust against the noise"[6].

To preserve the quality of the information subject to constraints of transmission rates, it is necessary to reduce the size of data by applying compression algorithms. Compression can be either lossless or lossy. The lossless principle is the one where no data is lost, but all the redundancies are removed. In the case of lossy algorithms, data fidelity is being lost within predefined error margins. Many use cases allow data to be lost, so lossy compression is usually implemented when higher compression ratios are required.

To achieve effective research on the topic, mixed methods of case study and experiment will be combined. The case study provides an investigation and quantitative comparison of state of the art compression algorithms and a general way a data reduction can be achieved. On the other hand, when testing and comparing specific algorithms, experiment setup yields a quantitative approach for answering the research questions. As previously stated in the research questions, RQ1 and RQ2 can only be answered by executing an experiment and comparing obtained results.

Regarding the final expectations, a goal is to find optimal compression algorithm for networked embedded devices or IoT solutions that require data reduction. Optimum solution in this investigation refers to keeping the real time capabilities of the system within small boundaries of error tolerance.

5 Related Work

There exists a wide variety of compression algorithms with different classifications ranging from lossless and lossy ones, from the time domain to transfer domain ones, and from general ones to data specific ones like algorithms for the image, video, and text. This thesis aims towards lossy algorithms with a limited amount of loss of data fidelity. Each type of sensor has, a unique signal characteristic, and they are time-varying one-dimensional signals. A paper published by Tulika Bose *et.al.* [1] shows a study that compares four compression algorithms on a set of six types of signals. The target algorithms are CA and FR from time domain and *Chebyshev transform* (CH) and *Wavelet Packet Decomposition* (WPD) from transform domain. Data types processed for algorithm testing are Electrocardiogram (ECG), Photoplethysmogram (PPG), bearing data as quasi-periodical signals, accelerometers and different signals form solar irradiation measurement. The authors have concluded that there is a correlation between specific types of signals and compression ratio and error margins as metrics for the number of certain compression algorithms.

The first research question is mainly related to CR factor, which highly varies depending on compression type and field of application. According to Mamaghanian et. al. [7] and the research done on compression of ECG signals, the CR (measured in percentage) ranges between 90% and 51 %, depending on the quality of the reconstruction. The research was performed on Discrete Wavelet Transform (DWT). More on CR in health application has been presented in [8] where CR factor varies between 17.6 and 44.5 and with precision up to 2.0 %. Traditionally lossy metrics perform CR between 3 and 15, whereas lossless ones range between 1 and 3 times [9].

In the research [9], hybrid model encountering both lossy and lossless compression, applied on ECG system, result in CR of 2.1 and 7.8 for lossless and lossy compression respectively. Hence, the aim is to investigate CR factors in CA applications on battery management system and a huge challenge to accomplish more than 90% of CR.

IoT devices generate huge amounts of data, that should be handled appropriately. According to Moon et al. it is a big challenge to find an optimal balance between a CR factor and information loss[10]. In the proposed work [10] a feasibility analysis of lossy compression is performed on agricultural sensor data by comparing the original and reconstructed signals. Sensors in weather stations are used as the main data resources. Among various existing algorithms the focus is put on main representatives of frequency domain compression algorithms, DCT, *Fast Walsh-Hadamard Transform* (FWHT), DWT and *Lossy Delta Encoding* (LDE). Results show that DCT and FWHT have higher compression ratio than other ones. Observing the results for higher compression factors, error rates exhibit a significant increase. On the other side, LDE, compared to DCT, LDE, and DWT, allows for smaller compression factors, maintaining the error within a smaller range.

One of the most important factors in efficient operating of IoT embedded solutions is power consumption. The power efficiency factor can be increased in lots of communication elements, mostly related to hardware and system setup, but the core problematics revolve around the amount and speed of the data transfer. According to Stojkoska et al. [11] power consumption in preprocessing is much lower than power consumption during the transmission process. The focus of the research was put on the development of a new principle for *delta compression*, that results in good lossless data compression. Implementation of such an algorithm with a high level of execution complexity, demand more memory, capacity and processor power. The coding scheme proposed by Stojkovska and Nikolovski achieved up to 85% energy saving, resulting in higher compression ratio and lower memory capacity than the conventional *delta coding*.

As already discussed in the previous section, nodes mainly consist of sensor and controller with quite low computational power. The end-device is supplied by an external battery that is not easy to replace, in some cases even tough to access.To prolong the lifetime of the battery and the supplied node, it is necessary to reduce the data traffic, since the radio communication elements exhibit very high power consumption. In [12], efficient data compression algorithm suitable for commercial wireless sensor networks has been proposed by exploiting the principles of entropy compression.

A novel hybrid data compression scheme that reduces power consumption has been introduced by [13]. This scheme reduces power consumption in the wireless sensor by focusing on ECG application and compresses data with lossy and lossless techniques. Lossy compression with CR averaging around 7.8 saves power in wireless transmission by 18% with Bluetooth transceiver, whereas lossless algorithms based on entropy coding compress residual error stored in local memory and saves 53% for raw data transmission. The residual error is extracted from raw data and reconstructed lossy data and coded with Huffman coding. They have shown that this approach can be attractive to bio-medical applications since it allows prelim-

inary signal analysis on a lossy signal, where there is no need for precise readings when the heart rhythm is stable. If abnormal heart rhythm is detected, residual data is retrieved wirelessly. This hybrid transmission method reduces energy consumption in all IoT applications that do not require precise data for preliminary analysis. Further improvements in power reduction can be made by using dedicated hardware and low energy wireless protocols. Also, it is possible to achieve indirect improvements by using state of the art compression algorithms, which show better performance than Fan compressor chosen in [13].

If there is no need for absolute precision, especially in applications that generate a lot of data that needs to be sent to the server via IoT networking, lossless compression exhibits negligible impact and show no difference, as stated by Aekyeung Moon *et.al.* in [14]. Their paper explores the impact of lossy compression on smart farm analytic which is highly important when making local weather predictions using machine learning (ML) techniques. The importance lies in the fact that if error bounds in lossy compression algorithms discard too much useful data from the original sequence, classifying ML algorithms will not be able to give right predictions, rendering the concept of smart farming obsolete. They have shown that by achieving CR of more than 99.9% with frequency domain lossy algorithms, like DCT and FWHT, ML algorithms respond with better prediction, compared to the prediction for raw data. Error bounds for lossy algorithms remove high frequency data points. Hence, removing most of the outliers makes classifiers better for predictions, and at the same time achieves high compression rates with allowed data quality loss.

Moshe Mishali and Yonina C. Eldar have introduced a way for compression algorithm of analog signals in the stage of analog to digital conversion (ADC) using so-called Xampling [15], where the usual scheme states that sampling rate of an ADC should be higher than the maximum occurring frequency in the signal we want to sample [3]. Xampling is a method that reduces the amount of data in the sampling stage by sampling the analog signal with sub-Nyquist frequency without losing the information. One approach is a combination of sampling theory with compressed sensing. Compressed sensing exploits the fact that vectors in undetermined linear systems are spars (mostly zeros) and combination with random demodulation extends the idea by recovering discreet harmonics. This method although being efficient is, on the other hand, challenging to implement and has high computational complexity. Xampling achieves sub-Nyquist sampling with a modulated wideband converter (MWC) which is more practical in terms of implementation and computational complexity.

Yaniv Zigel *et.al.* have published a paper [16] that proposes a compression algorithm for ECG signals. A newly proposed algorithm, analysis by synthesis coding (ASEC), exploits the predictability of measured ECG signals. ASEC adopts long and short-term predictors for compression of ECG signals with so-called beat codebook that records typical heartbeat signals. As a lossy compression algorithm that achieves CR of 30, it operates on metric for measuring distortion in the reconstructed signal with either Percentage Root Mean Square (RMS) Difference (PRD) or Weighted Diagnostic Distortion (WDD).

We mentioned some of the data reduction techniques that reduce dimensionality

[3] Nyquist criterion, briefly explained in chapter 5

such as discrete Fourier transform (DFT), DWT. There exist other techniques mentioned and compared in research done by Kaushik Chakrabarti *et.al.* in [17] such as Singular Value Decomposition (SVD), Piecewise Aggregate Approximation (PAA) and their newly developed method Adaptive Piecewise Constant Approximation (APCA) that is tailored for faster and better indexing and calculation. Calculation and dimensionality reduction is made at the expense of precision, based on metrics for distance calculations in different forms of norm calculation (usually Euclidean distance norm). This novel method APCA outperforms aforementioned sophisticated dimensionality reduction methods by two orders of magnitude and supports arbitrary norms while being able to be indexed in a multidimensional structure. This method is also applicable to other data sequences rather than just time-series data.

In some applications of signal recording lossy compression is a desirable way of manipulating by data as mentioned in [18] Konstantinos et al. write about a piecewise-linear approximation (PLA) technique for lossy compression that can compress signals in a given tolerance range with high compression ratio. PLA has linear computational complexity $O(N)$ and advantages over other techniques in frequency or time domain, not only speed-wise in the compression stage, but in reconstruction as well. It is reconstructed only by linear interpolation, and at the same time filters the data. As mentioned in work, the algorithm uses only a few mathematical operations and comparisons which makes it easily implementable in Very Large Scale Integration (VLSI) for a single chip integration. Furthermore, improvement of compression ratio has been proposed by introducing lossless Huffman coding scheme after the lossy one.

Time series data generated by wireless sensors poses a challenge when it comes to power and memory reduction as well as keeping the RT performance of the system. This problem has been tackled by many including researchers Iosif Lazaridis and Sharad Mehrotra in their work [19]. Their task was to capture sensory data with aforementioned system performance demands with the guarantee of and result being in the range of allowed error bound based on Euclidean distance norm. To be able to reduce the amount of data being generated, not to send the raw samples that use communication bandwidth and keep RT performance, data compression has been utilized, as well as prediction scheme.

Numerous power reduction methods have been invented as a result of data reduction. William R. Dieter *et.al* have worked on a method that changes the sampling rate of ADC by Nyquist rate [20]. Nyquist rate requires at least two times of the maximum existing frequency to produce good results. In a time-varying analog signal that frequency threshold changes depending on the signal. This variability in the signal of interest allows for the ADC to change its sampling rate accordingly and thus indirectly reduce the strain on next stage of digital signal processing (DSP), reducing the power consumption. Researchers in [20] have shown 40% decrease in power for static sampling rates, depending on processing per frame. This method uses buffers and dynamic voltage (or frequency) scaling, to ensure the data and power reduction.

As machine learning and other closely related fields are trying to find patterns in data, as such, they can be used to compress periodical signals. Auto-encoders as a

type of artificial neural net as a method of lossy compression of biometric data have been introduced by Davide Del Testa and Michele Rossi in [21]. Their work aimed to produce a lightweight lossy compression based on auto-encoders that are both memory and power aware. Auto-encoders have been shown to be a lightweight method of lossy compression compared to another state of the art counterparts like DCT, principal component analysis (PCA), wavelet transform and linear approximation. Neural net training needs to be done offline, but once that is done the algorithm can be easily put in a hardware solution on a microcontroller unit (MCU).

6 Battery Management System

In the recent years, batteries are becoming the solution to multiple problems around the world. Storing excess energy created by renewable power sources and electric vehicles that run on them, batteries are one of the fastest growing industries. Many modern systems need a battery monitoring, to have an insight into parameters of interest, which helps to develop better battery technology in the future and see what battery state and performance is. The battery parameters can be measured directly or estimated based on other measurements.

Battery parameters being monitored:

- Current (consuming and leaking)
- Individual battery voltage
- Individual battery impedance
- Individual battery temperature
- Ambient temperature
- State of charge (SoC)
- Ripple current

System specification influences the process of battery data acquisition. It is divided into several stages as shown in Figure 6.1. The first step is the measurement of the signal and conversion to an electrical - voltage signal since ADCs operate on converting voltage levels to discrete digital data. Next, the signal needs to be conditioned, so it adjusts the range of voltage in which ADC operates and filters out the noise. Finally, resolution and sampling frequency of ADC should be selected based on a required specification, precision, and signal type.

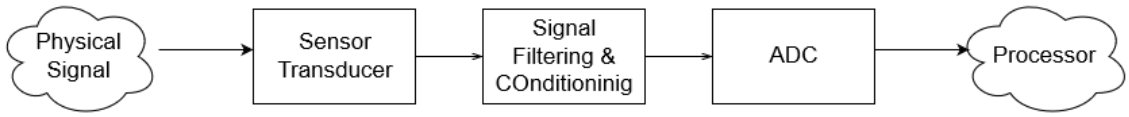

Figure 6.1: Data acquisition block diagram

Physical parameters being measured for battery monitoring system are current, voltage and temperature signals.

6.1 Sensors

Current can be measured in different ways depending on the magnitude, type, accuracy, and range. Current measuring techniques for digital systems convert current with some transducer to the voltage, that is later acquired by ADC.
Commonly used current measurement techniques:

- Shunt resistance that directly converts current to the voltage drop
- Transformer or coil measurement that creates proportional voltage according to Faraday's law
- Magnetic approach with Hall effect sensor that generates potential difference based on magnetic field strength around wire where the current passes and deflects the current flow inside the sensor
- Magnetic approach with magneto-resistive material that will generate a voltage drop in the resistor, proportional to the change of the resistance influenced by a magnetic field created by passing current.
- Opto-magnetic current sensor that creates voltage drop based on a light deflection for high currents with the order of magnitude 1000A

From all the mentioned current sensing techniques, shunt resistance was chosen as the easiest and cost-effective solution for demonstration and proof of concept purposes.

Temperature measurement for electrical systems is also done by converting the temperature to the voltage with a transducer. Practical approaches for the embedded system include measuring temperature indirectly by measuring voltage in a system that is linearly dependent on temperature. Semiconductor devices like diode and transistor integrated circuits (ICs) provide temperature measurement that can be proportional to absolute temperature and complementary to absolute temperature. Other practical methods include thermocouples that generate potential difference on the end of the sensor. Resistor methods with negative temperature coefficient (NTC) and positive temperature coefficient (PTC) are a resistive approach where the temperature is indirectly measured in the linear region of NTC and PTC resistors. Thermoelectric effects such as Seebeck, Peltier, and Thomson are also examples of temperature measurement, where temperature difference in certain materials generates a potential difference or current on the contact pads of sensors [22].

When measuring a physical parameter and converting it into a digital electrical signal, it is always being converted to voltage difference which is in the digital system with the help of ADC converted to a stream of ones and zeros. Voltage measuring in that sense is the easiest to do and requires fewer steps than other methods since it does not need a transducer to be converted, except for alternating and high voltages.

6.2 Signal Conditioning and Filtering

Figure 6.1 as a second step has signal filtering and conditioning. Practically signal conditioning is done first after that signal can be filtered using some of the analog filters. Conditioning is a process of converting the voltage from sensor transducer from one range to another one suitable for the ADC, usually $0 - 3.3V$ or $0 - 5V$. This conditioning is done with operational amplifiers in the configuration for linear or other remappings of voltage ranges based on sensor output signal. Sometimes it is needed to weaken the signal, so an attenuator is used. After the signal has been properly conditioned analog filters are constructed based of what kind of noise is excepted from the signal, so low-pass, high pass or band pass filters are selected and placed as close as possible to the ADC, so no additional noise is being induced on the electrical path to the ADC.

6.3 Signals

Sensors that measure time-series physical signal are mathematically two-dimensional but are viewed as one-dimensional signals when being processed with any digital signal processing tool or algorithm since time is being excluded in most of the cases when digital filtering is being done, or some encoding.

When the signal is being measured it is necessary to know the behavior of the signal. Based on the application and the physical nature of signals characteristics and requirements of the system it is necessary to determine sampling frequency that determines how many data point will be measured in one second.

Nature of the signal will determine the upper limit if sampling frequency necessary. The upper limit of the frequency is attained by spectral analysis of the signal being measured. Spectral analysis is usually done by applying Fourier transform on the signal to determine the largest occurring frequency in the signal that needs to be recorded. Nyquist theorem talks about sampling frequency is at least two times higher than the largest occurring frequency in the signal.

Application and requirement can now be considered in practical terms to reevaluate the sampling frequency since applying the highest possible sampling rate are not always needed, or it can even be unwanted. Reason for not applying highest sampling frequency lies in the fact that high sampling rate produces a lot of data and thus memory requirements go up, as well as power consumption and system cost. To avoid that, an expert determines what sampling rate is needed and enough for the application with consideration of limitations.

Our research is being done in the field of battery management and an expert from the company evaluated that sampling rate of voltage, current and temperature signals is enough to be 1Hz. The behavior of the current and voltage signal will in reality possibly has abrupt changes since battery system is powering an electric motor in the car that can induce high and sudden changes in the signal being measured.

Companies engineer-researcher have made this observation because in the application of SoC will not have a high influence on the battery consumption overall, hence allowing low sampling rates for signal measurement.

7 Data Compression

All surrounding information is constituted of analog signals, in the form of light, sound, heat. To be processed by a computer, it is necessary to map the signal into a corresponding digital form. This ADC conversion, during the process of data sampling, inherently suffers an inevitable loss of information that is unavoidable. Depending on the level of precision we want to achieve, it is necessary to choose a specific sampling frequency, according to well-known Nyquist-Shannon sampling theorem [23]. However, during the sampling process, in embedded systems usually obtained at the sensor level, many of the sampled data are not necessary to store and process, causing memory saturation and latency in program execution. The problems arise in embedded IoT solutions when the data are supposed to be sent over a communication network. The networks themselves are limited in the amount of data that can be transferred using specific wireless protocol. To reduce memory requirements and decrease power consumption, we consider a solution based on data compression. For the implementation of such a solution, the system itself should show a certain level of error tolerance, trading quality for cost-efficiency.

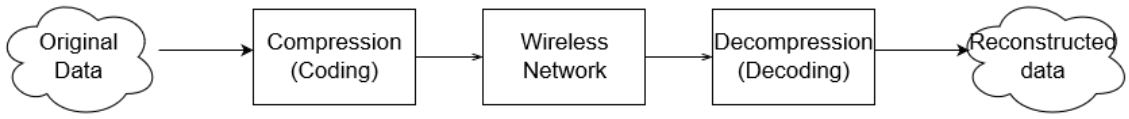

Figure 7.1: Data transmission block diagram

7.1 Algorithm Selection

First studies on data compression techniques appeared in the early 1940s, and gradually improved as needs for data transfer increased. Before any data transmission occurs, one of the crucial points in the design of a system is whether it is vital to compress the data. Based on the quality of reconstruction of the original data, we introduce lossy and lossless data compression. If the level of data sensitivity and confidentiality is very high and no information of the data should be lost, then one of the lossless techniques must take place. Examples of such occurrences are usually in the field of biomedical engineering, where no loss in graphical representation is allowed. Also, transmission of text messages is usually very sensitive causing considerable troubles in the transmission of a single letter. Usually, the information in the form of picture, sound or video allows for high level of compression, as soon as the loss of information does not seem disturbing to the observer. In that occasion, we call for lossy algorithms to obtain the compression.

The primary focus of the research is embedded system solutions implemented on battery, which are equipped with numerous measurement technologies. As in embedded IoT solutions, in general, the data are not extremely sensitive, especially in measuring battery parameters regarding SoC, we can afford a certain data loss traded for benefits in memory capacity and power consumption. Thus, in the research, the main focus is put on lossy compression techniques.

Usually, when the selection of suitable algorithm comes to the question, the first property of the technique that comes to our mind is CR factor, which stands for a relation of reduced (compressed) size of data to the original one. It can be expressed either as a ratio (see equation 7.1.1), or in a percentage form, as represented by equation 7.1.2.

$$CR = \frac{original\_data}{compressed\_data} \tag{7.1.1}$$

$$CR_P = (1 - \frac{compressed\_data}{original\_data}) * 100\% \tag{7.1.2}$$

CR factor is, without doubt, the most important factor in algorithm selection. To perform a thorough investigation, especially when CR of a candidate algorithm is not dominating over its competitors, it is necessary to encounter other relevant factors. Those factors are adherence to standards, algorithm complexity, speed and processing delay, error tolerance, quality, and scalability. [24]

7.2 Lossless Compression

Lossless techniques, as its name states, ensure no loss of information in a compression process. The reconstructed data is entirely identical to original one. Its benefits are mainly used in medical applications for analysis of different records for text compression where any loss might bring confusion. Lossless compression algorithms work on a principle for removing redundancies in the data. At first glance, it might look reasonable always to apply a lossless technique, but many applications do not require exact reconstruction of the signal. Lossless CR factor is much smaller compared to the CR achieved by lossy algorithms. As already mentioned in Section 3, its value ranges in most cases up to 3, what leads to 70% of compression, and the entropy limit bounds this CR factor. Entropy limit represents the lowest amount of data needed to encode data and is calculated 7.2.1 where p_i is the probability of occurrence of the character.

$$Data = -\sum p_i log_2(p_i) \tag{7.2.1}$$

7.2.1 Huffman Coding

Huffman coding is an example of lossless compression, used for text stream data reduction. Characters are coded to represent letters and other symbols in the text documents. Thus redundancies are removed. The text is coded in such a way that every symbol is coded with the amount of 8 bits from ASCII table, or some other way of representation. Most of the possible symbols in the table are not used or used less frequently than the others. The most frequent symbols are coded with fewer bits of data and less frequenter with a higher number of bits. Encoding is done by a traversal tree, where leaf nodes represent less frequent characters. Parent nodes represent high-frequency nodes. Going left in the tree, we assign bit value '0', and moving towards right we assign bit value '1', thus encoding each node in the tree.

The idea can be expanded upon by coding adjacent characters, word, sentences, or even paragraphs if their occurrence is frequent in the text [25].

7.2.2 Arithmetic Coding

Similarly to Huffman coding, arithmetic coding as a type of lossless data compression is used for text string compression, but can also be applied to other types of signals. It encodes data on a similar principle of probability distributions, but only for strings of text from a set of all symbols used in the document. Probability set is divided according to the probability distribution of the characters. The string being compressed is encoded from left to right by dividing each subset in the same manner as an initial set of probability distributions in the number of times how long the string is after which the final probability is given to the string that is represented with a float number between zero and one [26].

7.2.3 Other Lossless Algorithms

There are a lot of lossless compression algorithm schemes for different types of files or general purpose ones that can apply lossless compression on any file. Huffman and Arithmetic coding can be classified as general purpose algorithms. Some more famous and widely used algorithms are Run-length Encoding (RLE), dictionary-based general purpose algorithms where Lempel-Ziv compression also known as LZ77 and LZ78 is foundation ground for another dictionary-based compression. Lossless and general purpose algorithms are not the extents of this research and as such will not be included in the rest of the paper.

7.3 Lossy Compression

A compression technique is said to be lossy if some insignificant data may be omitted in data transmission. The level of significance is usually defined by the system developer, to meet the system requirements. In such manner, the term "reversibility" of the compression is introduced. The compression is called irreversible if it is impossible to reconstruct the original signal by decompression. Similarly, the compression is called reversible if the original data may be recovered perfectly by decompression [27]. Irreversible compression is usually satisfactory for data stored in the form of multimedia images, video or audio signals. They result in changes which are hardly recognized by human sense(receptive) organs. There are numerous algorithms and numerical methods widely used for lossy data compression. They may exist either in time or frequency domain. The most popular time domain compression algorithms are:

- Backward Slope (BS)
- Box Car (BC)
- Fractal Resampling (FR)
- Critical Aperture (CA)
- Curve Fitting (CF)

7.3.1 Box Car

BC method stores a data point only when there is a significant change with respect to the last record. The method does not give good compression results for the signals exposed to rapid changes. It provides excellent results on temperature data [1].

7.3.2 Backward Slope and Critical Aperture

BS is based on slope value calculated with the last two points with respect to the current sample. This method usually does not appear as very efficient compression. Another method based on slope calculation belongs to Historian Compression Methods. It is called Critical Aperture, sometimes "Rate of Change" and "Swing door" compression [28]. The technique gives an outstanding real-time performance, resulting in high compression ratio even for the signals with a rapid change of slope. Due to all the benefits are given by the algorithm, it is selected for detailed analysis and implementation on LoRa protocol (8.3). General Electric originally introduces the method.

7.3.3 Fractal Resampling

FR operates by calculating midpoint distance. The method requires all the data to be stored to achieve a full compression. These limitations are the main reason why this technique is not applicable on real-time processes. However, due to excellent compression ratio and applicability on signals of different rate of change, it can easily be tailored to some of the selected battery signals. (8.2).

7.3.4 Curve Fitting

CF method gives a precise estimation, but considering the execution complexity very high [4], it does not seem acceptable for our system. The complexity appears very noticeable on a vast number of samples. The method also does not support real-time operations, what makes it less favorable compared to other time-domain algorithms.

According to signal theories and Fourier transform every signal can be represented as a linear combination of sinusoids of different amplitudes and frequencies. However, to show such properties of the signal, it is necessary to perform the mapping of the data to the frequency domain. Direct and inverse Fourier transform techniques are represented by formulas 7.3.1 and 7.3.2 respectively.

$$F(i\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \tag{7.3.1}$$

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(i\omega)e^{i\omega t}d\omega \tag{7.3.2}$$

[4]Complexity of standard time-domain compression algorithms is of the order of N, where N is the number of samples of the input signal. Curve fitting method is based on matrix calculation, which results in NxN complexity

Once the signal is mapped into the frequency domain, its handling becomes easier and usually more efficient. We can see the contribution of each sinusoid, which allows the signal to be easily filtered and processed for any application. After analysis, processing and/or filtering of the signal is completed, it is remapped to a corresponding original in the time domain.

Hence, numerous frequency-domain data compression algorithms have been developed:

- Discrete Fourier Transform (DFT)
- Fast Fourier Transform (FFT)
- Discrete Cosine Transform (DCT)
- Fast Walsh-Hadamard Transform (FWHT)
- Discrete Wavelet Transform (DWT)
- Chebyshev Compression (CH)

All of the before-mentioned techniques are based on amplitude-frequency and phase-frequency characteristics, which are further manipulated to process the signal. The process of data mapping from time to frequency domain is performed by Fourier transform techniques. Initially, it has been applied to continuous time signals. However, entire technology is based on digital processing, where computation process is obtained on computers. Hence the time-data suffers limitations in amplitude and time resolution. This led to the invention of the methods that could be applied to discrete-time signals. In these writings, only the essential formulas are presented, since entire derivation of the methods is not in focus. Very first step in a sampling process is frequency selection. It is based on the highest existing frequency in the spectrum. Let ω_M be the maximal frequency in frequency spectrum $F(i\omega)$, and ω_s be the sampling frequency. According to the Nyquist sampling theorem, it is necessary to satisfy the equation 7.3.4

$$\omega_M = \frac{2\pi}{T_M}, \omega_s = \frac{2\pi}{T_s} \tag{7.3.3}$$

$$\omega_s \geq 2\omega_M \tag{7.3.4}$$

For a package of N sample data, sampled in a continuous time signal, equation 7.3.5 states

$$N = \frac{T_M}{T_s} \tag{7.3.5}$$

7.3.5 Discrete Fourier Transform (DFT)

Let us imagine a discretization of certain signal slot of $f(t)$, having Fourier transform $F(i\omega)$. Discretization process results in a discrete signal f(kT_s), and samples are taken at discrete time points $t = kT_s$. Applying Fourier transform, we get a continuous frequency domain signal. Reminding the fact that continuous signals cannot be computed on microprocessors without sampling process, we are interested in the relation between the samples f(kT_s) and corresponding discrete signal in frequency domain represented as $F(im\omega_0)$. The relation between the two data, in two different domains, is presented by the formulas 7.3.6 and 7.3.7, for direct and inverse discrete Fourier transform respectively.

$$F(im\omega_0) = \sum_{k=0}^{N-1} W_N^{mk} f(kT_s) \tag{7.3.6}$$

$$f(kT_s) = \frac{1}{N} \sum_{m=0}^{N-1} F(im\omega_0) W_N^{-mk} \tag{7.3.7}$$

where

$$W_N = e^{-\frac{i2\pi}{N}} \tag{7.3.8}$$

N - number of samples per transaction period
ω_0 - transaction frequency

$$\omega_0 = \frac{2\pi}{T_0} \tag{7.3.9}$$

Detailed derivation of the equations 7.3.6 and 7.3.7 are presented in the literature for signal and systems analysis (recommended to see [23]).

The process of mapping of the data into frequency domain gives the same amount of data as the original. To perform a process of compression, it is necessary to form a criterion, which decides whether a certain data should be saved or discarded. Samples with amplitude contribution smaller than a predefined value are neglected. Critical amplitude is left to the designer to choose. Recorded samples(amplitudes) are considered in the process of reconstruction (inverse transform) by a sinusoid of specific amplitude and phase.

7.3.6 Fast Fourier Transform (FFT)

Computation process of DFT implementation is very complex [5] and takes a huge processor execution time. This calculation time is the main reason why computers rely more on the algorithm called Fast Fourier Transform. The algorithm performs the same functionality as DFT, with a much smaller number of mathematical operations[23].

[5]With complexity order of N^2, where N is a number of samples for a signal time slot

7.3.7 Discrete Cosine Transform (DCT)

DCT is a method similar to DFT. It takes a finite data set and maps to a finite sequence, representing the data in frequency domain. The sequence represents an amplitude spectrum for cosine functions at different frequencies. Unlike DFT and FFT algorithms, which contain both real and imaginary coefficients, DCT frequency sequence is strictly real number sequence. Whereas DFT and FFT result in both amplitude-frequency and phase-frequency characteristics, DCT has no phase characteristics. Different forms of DCT are mentioned in chapter 6.

7.3.8 Chebyshev compression (CH)

This is a method for a signal approximation based on Chebyshev polynomial. The method applies to one, two and three-dimensional data. It is invented by National Aeronautics and Space Administration (NASA) when requirements for more data transmission from space-crafts increased. In the first stage, the algorithm splits the data into blocks, organized in the form of a matrix with predefined dimensionality Nx1. For applications on two-dimensional and three-dimensional data, the blocks would be organized as NxN and NxNxN dimensional, respectively. After the Chebyshev coefficients are calculated, thresholding is performed. It is a process of coefficient selection based on a predefined threshold value. Finally, samples which do not give a high contribution to polynomial approximation are discarded [29].

8 Application

Signals from different physical systems have different behaviors 6.1. As mentioned in 6.3 an expert determines needed sampling rate of an ADC to reduce the amount of data as much as possible. With signals that are not periodical and cannot be predicted redundancies can also be found. By removing the redundancies from the signals with lossy compression algorithms, data reduction can be much greater. For this research, four algorithms have been chosen to be implemented, tested and evaluated for data reduction, power consumption and RT performance on the embedded system for sending the data via LoRa:

- Fractal Resampling - FR
- Critical Aperture -CA
- Fast Fourier Transform - FFT
- Discrete Cosine Transform - DCT

Chosen algorithms are state of the art and practice. FR and CA are time domain, and FFT and DCT are frequency domain compression algorithms. Performance of each of the algorithm will be evaluated for different parameters. Reconstruction or decompression of the signal after an algorithm is applied will show how efficient the algorithm was. To evaluate the efficiency different metrics will be applied. When implementing the algorithm the error bound needs to be set, and an expert again chooses this error based on application. Error bounds are coded in relative terms compared to the range of the measured signal, and some applications tolerate an absolute error of up to 5% in respect to the amplitude range of the signal value. This error bound is easily calculated and determined for time-series compression algorithms since they work in the time domain and do not transform the signal. In the transform or usually frequency domain, it is difficult to determine this error with respect to the amplitude in the time domain. Usually, the approach is to exclude the coefficients that are to low compared to other coefficients again in relative terms where 5% error would again be considered an upper limit for data loss when applying these lossy algorithms. This approach where error bound is determined in the relative terms compared to the range of absolute value cannot be used for evaluation and comparison of both time domain and transform domain algorithms, so it is necessary to introduce a different metric for assessment of the algorithms.

8.1 Algorithm quality metrics

The first metric that can adequately test the algorithms applied in other papers [1], [9], [8] and used for evaluation of compression algorithms is PRD. This parameter is more commonly a normalized version of PRD with formula 8.1.1 comparing the reconstructed compressed signal with the original one. In the formula, x represents the original vector $\widetilde{x}$ the reconstructed version of the signal and $\bar{x}$ mean value of the original signal. N is number of samples for an entire data set, which is supposed to be compressed.

$$PRD = \sqrt{\frac{\sum_{n=1}^{N}(x(n) - \widetilde{x}(n))^2}{\sum_{n=1}^{N}(x(n) - \bar{x})^2}} \cdot 100\% \tag{8.1.1}$$

Normalized PRD is a more qualitative way of comparing compression algorithms, and it provides a clear picture of how well the signal is reconstructed and if the data fidelity is maintained.

Another important parameter for testing the effectiveness of the compression algorithm is CR with the formula given by 7.1.1 and for percentage CR by 7.1.2. CR is a crucial parameter since this factor will significantly influence the decision on implanting specific algorithm since it shows how many times has the data been reduced compared to the raw form.

Other parameters to look for after reconstructing the signal and compare it with original one are:

- Normalized Standard Deviation (STD)
  STD is a measure of the distribution of points in a, and it is calculated according to formula 8.1.2 where N is the number of points in data set, μ is the mean average value, and x_i is i_{th} point in data set. Low values for STD indicate that points in data set are closer to the mean average value and high values indicate that points are further from the mean.

$$S = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}|x_i - \mu|^2} \tag{8.1.2}$$

- Mean Average
  Mean value represents the average value of all points in data set and is calculated by summing all the values of points and dividing the sum with the number of points in data set as shown in formula 8.1.3

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{8.1.3}$$

- Median Value
  Median value tells what the value of a point that splits the data set into higher and lower halves is. Extreme values do not significantly influence it. Its value is used to see what is the typical value of point in the dataset.

- Data Skewness
  Skewness is a parameter that shows if a data set distribution is systematical or asymmetrical, and if it is asymmetrical it determines the amount and direction of asymmetry. If skewness is 0 then we have a normal distribution which is symmetrical, if skewness is negative, the distribution is leaning towards the left, and if it is positive, it leans towards the right. The formula for calculating skewness is given by 8.1.4

$$s = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^3}{(\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2})^3} \tag{8.1.4}$$

- Data Kurtosis
  In data set distribution kurtosis measures how much are points susceptible to outliers. A normal distribution has a kurtosis of 3, and with lower kurtosis, there are fewer outliers and vice versa. The formula for determining the kurtosis is:

$$k = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^4}{(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2)^2} \tag{8.1.5}$$

8.2 Fractal Resampling - FR

Fractal resampling is an offline compression algorithm which can be obtained only when all input data are present. The technique is based on midpoint displacement calculation, where midpoint distance is halved in every iteration. Algorithm performs maximum utility for $2^n + 1$ samples [6]. Compression quality highly depends on the predefined error tolerance. It is left to the system designer to define this parameter. In practice, for extremely accurate data transmission, sample value should not exceed the boundary of 2% of sensor output range.

Here we give a step-by-step demonstration of the fractal resampling compression, on a sequence of a nine sample data. Each sample is denoted as a pair of (x,y) coordinates, usually generated with an equal inter-arrival period. Note that blue lines represent original signal, whereas red dashed lines define region of tolerance for midpoint check. In the first step we draw a line between the first and the last point in the sequence, which are stored by default. Next we define an error tolerance region, bounded with upper and lower line, and check whether the midpoint belongs to the region. If it is inside, then the point is discarded, otherwise it must be stored and used in the process of signal reconstruction (decoding). The point splits the sequence into two sub-sequences (see figure 8.2. The same process is performed on each next sub-level. Third (last) level for a sequence of eight samples is presented on the figure 8.3. There are $log_2 N$ steps in total, for N samples in the data set, where N is selected as $2^n + 1$[7].

The reconstructed signal is completely composed of the samples contained in the time-sequence original. In this example, reconstructed signal is a time-series of x(0), x(2), x(4), x(6) and x(8). Calculated CR value, according to 7.1.2, is 45.5 %.

The core problematics of the research revolves around the compression ratio accomplished on battery data, particularly on current, voltage and temperature signals.

8.3 Critical Aperture - CA

Critical Aperture compression is based on slope calculation, and it is also known as "rate of change" compression. The method is based on slope calculation and strives to eliminate all the points situated within a certain slope boundary. The following example briefly describes the algorithm. There we have a sequence of eight samples, with an approximate slope error tolerance represented by vertical double-arrow.

[6] n is a natural number (positive integer, i.e. 0,1,2,3...)

[7] n is an integer number greater than zero

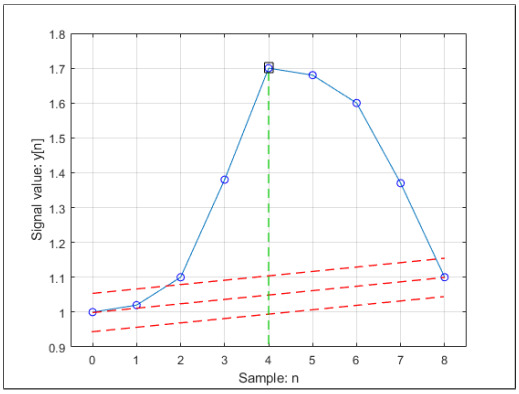

Figure 8.1: The first midpoint, arithmetic mean of the first and the last point in the sequence: x(4)

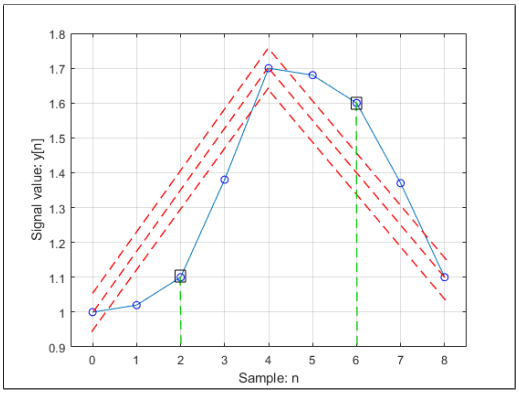

Figure 8.2: Sequence divided into two sub-sequences, arithmetic means at x(2) and x(6)

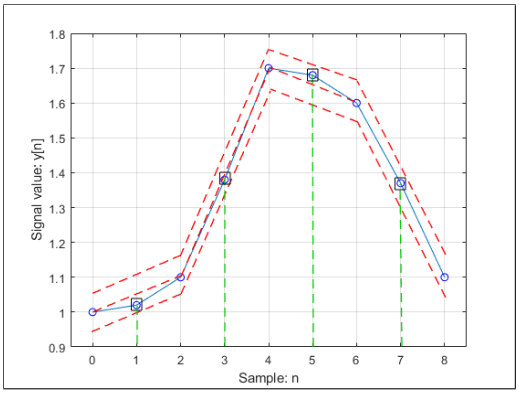

Figure 8.3: Arithmetic means at x(1), x(3), x(5) and x(7)

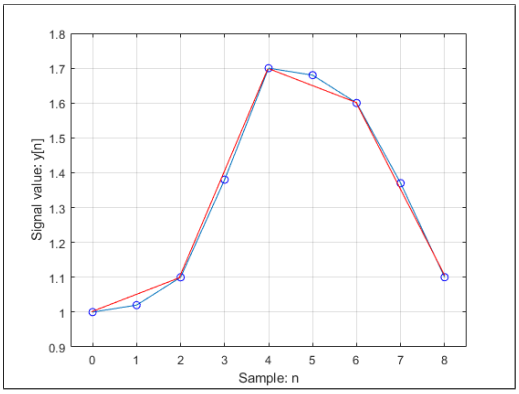

Figure 8.4: Signal plot of the original and compressed data

At the start, we define an initial slope value by use of x(0) and x(1). Next, it is necessary to fence allowed slope region, with regard to error tolerance. Initial slope field is marked with black straight lines A.19. Sample x(2) is the first to be tested, and slope of the point is lying within black boundaries, and the sample is discarded. In the next step, we take a look at the sample x(3) and mark the margins with respect to point x(2), represented by red region. Naturally, point x(3) is enclosed by red margins and therefore discarded for future use. Applying the same procedure on point x(4), represented by green lines, we see that the sample is out and therefore x(3) is stored as the last reference point.

In the next phase, we generate the reference slope using the points x(3) and x(4) expecting the x(5) lie in between black region, as shown on figure A.20. However, the point is out, and its predecessor x(4) is stored as a temporary reference. Going for the sample with index 6, and checking its affiliation for the black region, we see that it is not necessary to store the point. Eventually, the last sample, x(7), does not belong to the red region and therefore is stored in the sequence of compressing data (see figure 8.7). On the last graph, shown in figure 8.8, we see the comparison of the original data, represented by a blue and the compressed data depicted by a red plot.

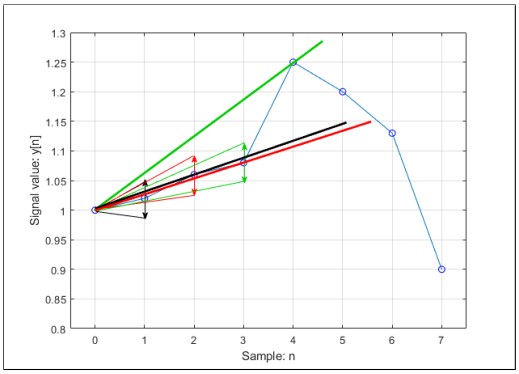

Figure 8.5: Initial slope with x(0) and x(1), further points x(2), x(3) and x(4) - x(3) stored

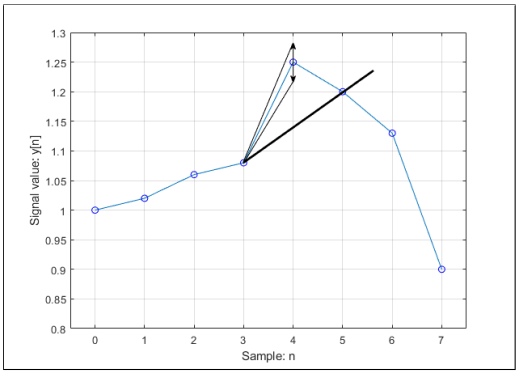

Figure 8.6: Slope generated with x(3) and x(4). Unsuccessful check at point x(5) - x(4) stored

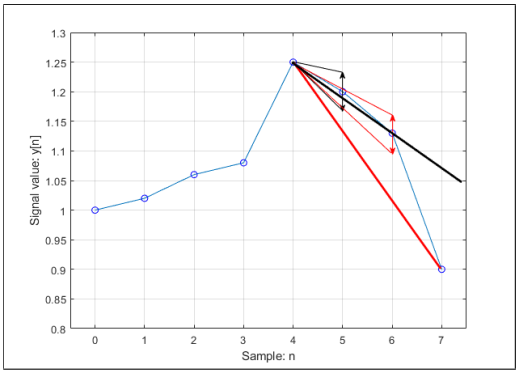

Figure 8.7: Referent slope with with x(4) and x(5) - x(6) stored

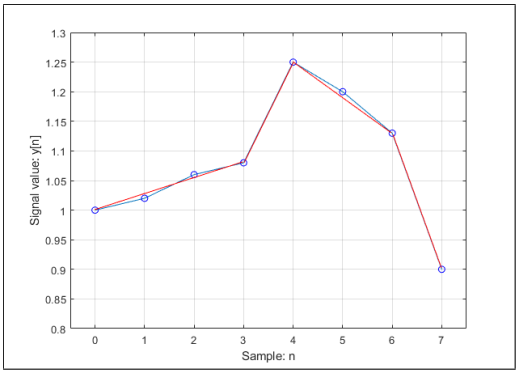

Figure 8.8: Original and compressed signal on the same plot

8.4 Fast Fourier Transform - FFT

FFT algorithm is mainly based on grouping (dividing) of the data-set into two subsets. Each subset is further halved until all the samples are distributed into pairs (see figure 8.9). Conventional DFT is applied on each pair as shown by the formula 8.4.1

$$X_3(m) = x(0) + W_2^m x(1) \tag{8.4.1}$$

$$X_3(0) = x(0) + x(1), W_2^0 = e^{\frac{2\pi}{2}0} = 1 \tag{8.4.2}$$

$$X_3(1) = x(0) - x(1), W_2^1 = e^{\frac{2\pi}{2}1} = -1 \tag{8.4.3}$$

Going a step back we calculate $X_2(m)$ and $X_1(m)$ [8].
Backward process is depicted on the figure 8.10

For a data set of N samples, there are $log_2 N$ steps of decomposition. Each step requires N complex summations and N/2 complex multiplications. Hence, the algorithm complexity is represented as $Nlog_2 N$. Size of the data in the compressed form is halved, since the frequency spectrum of the signal is symmetric, as a result of existing complex conjugates.

[8] See literature [23], Ch.8

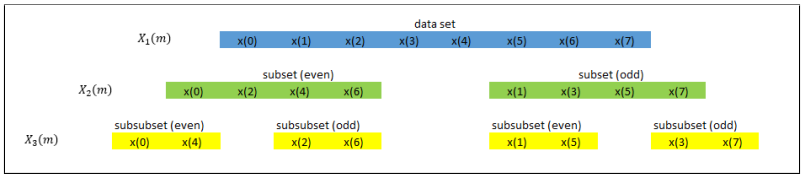

Figure 8.9: Data split process from the initial data set to the pairs

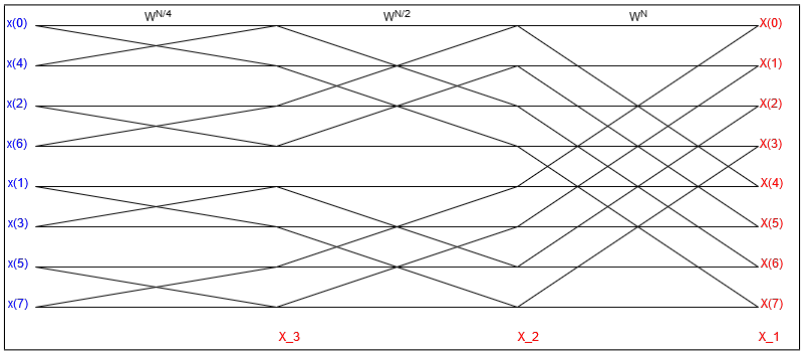

Figure 8.10: Backward calculation of the coefficients on sub-levels. Cooley-Tukey FFT with time decimation for N=8

8.5 Discrete Cosine Transform - DCT

DCT algorithm is based on similar calculation arithmetics as DFT. However, DFT considers an original time domain sequence to be in complex form, mapping it to corresponding complex form in the frequency domain. If a target signal is entirely in the time domain, then DFT execution complexity would be reduced, but still obtaining many unnecessary computations. In IoT sensory data acquisition, the data is always real. Hence applying DCT on the signal would result in the only real spectrum of coefficients. The algorithm can also be applied in "fast" form, similarly as FFT. For this implementation, the model of Fast Discrete Cosine Transform proposed by Lee (MIT) took place [30]. Here we present some of the DCT forms:

DCT-I

$$X_k = \frac{1}{2}(x_0 + (-1)^k X_{N-1}) + \sum_{n=1}^{N-2} x_n cos[\frac{\pi}{N-1} nk] \tag{8.5.1}$$

DCT-II

$$X_k = \sum_{n=0}^{N-1} x_n cos[\frac{\pi}{N}(n + \frac{1}{2})k] \tag{8.5.2}$$

DCT-III

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n cos[\frac{\pi}{N}(n + \frac{1}{2})] \tag{8.5.3}$$

DCT-IV

$$X_k = \sum_{n=0}^{N-1} x_n cos[\frac{\pi}{N}(n + \frac{1}{2})(k + \frac{1}{2})] \tag{8.5.4}$$

For a decompression process, it is necessary to obtain inverse transform, which slightly differs for all of the direct forms. Inverse transform of any formula is quite similar to the direct one. Hence:

$$Inv(DCT - I) = \frac{2}{N-1} DCT - I \tag{8.5.5}$$

$$Inv(DCT - II) = \frac{2}{N} DCT - III \tag{8.5.6}$$

$$Inv(DCT - III) = \frac{2}{N} DCT - II \tag{8.5.7}$$

$$Inv(DCT - IV) = \frac{2}{N} DCT - IV \tag{8.5.8}$$

9 System setup

To be able to test compression algorithms, and to answer the research questions asked in this thesis, some kind of hardware was necessary to do the evaluation. Development board EVK1100 with Atmels AT32UC3A0512 32 Bit microcontroller and FreeRTOS embedded operating system running on it, was used in conjunction with Sodaq Explorer development board with 32-Bit ARMs M0+ cortex ATSAMD21J18. Algorithms that were chosen in chapter 6 were implemented on EVK1100 board in Atmel Studio 7 IDE and programmed with Atmel's AVR Dragon debugger trough JTAG. The algorithms were first tested and implemented in `MATLAB`. Afterward, they are mapped into corresponding C code and downloaded to the board. Sodaq Explorer board was connected via USART with EVK1100 board. Compressed data is sent to the cloud via Sodaqs LoRa IoT module, which is part of Explorer board. Data is sent via LoRa wireless protocol from the end node to the router gateway.

Gateway hardware is Lairds Senturius RG186. Ethernet that was connected to the things network [31]. The gateway was registered by following the instructions after creating an account on the website mentioned above. Sodaq board, which is Arduino compatible, was programmed with instructions from official Sodaq support website [32], with some minor changes to the provided sketch program. Thus, it can be connected to EVK1100 board and assigned to Things Network as an application.

Entire system is presented on the figure 9.1

9.1 Implementation

Implementation of the algorithms is initially performed in `MATLAB`, where data handling is much faster and easier. After the testing is completed algorithm is downloaded to EVK1100 board. The detailed implementation of selected algorithms is described in Application section.

Testing points were put in MCU internal flash memory and stored as an array of type double, with actual values. However, there is an exception to FFT, which need a special data *treatment*. It is first mapped to range $[-1, 1]$ and then using macro converted from float type to signed fixed-point Q1.31 type, that is optimized for FFT and other DSP functions.

FFT compression algorithm is implemented with DSP library from Atmel Studio IDE, which contains a hardware-specific integrated example. This means that the algorithm for determining FFT coefficients has been optimized by writing a majority of the code in assembly language. Code optimization also includes a look-up table for twiddle factors, for up to 1024 coefficients. This makes the calculation process faster. FFT function requires three parameters: a vector for coefficients storing, an input vector, and a parameter that defines $log_2(Size)$ where Size is a number of samples for the data set. The function requires all the data points to be stored before a function call. This implies that the function does not exhibit RT capabilities. The compression process is performed by calculating the absolute value of a complex number, to determine the spectral coefficients. The error bound is defined after determining the range between the maximum and minimum value of the coefficients. Coefficients are removed from the final array and set to zero if they are smaller than

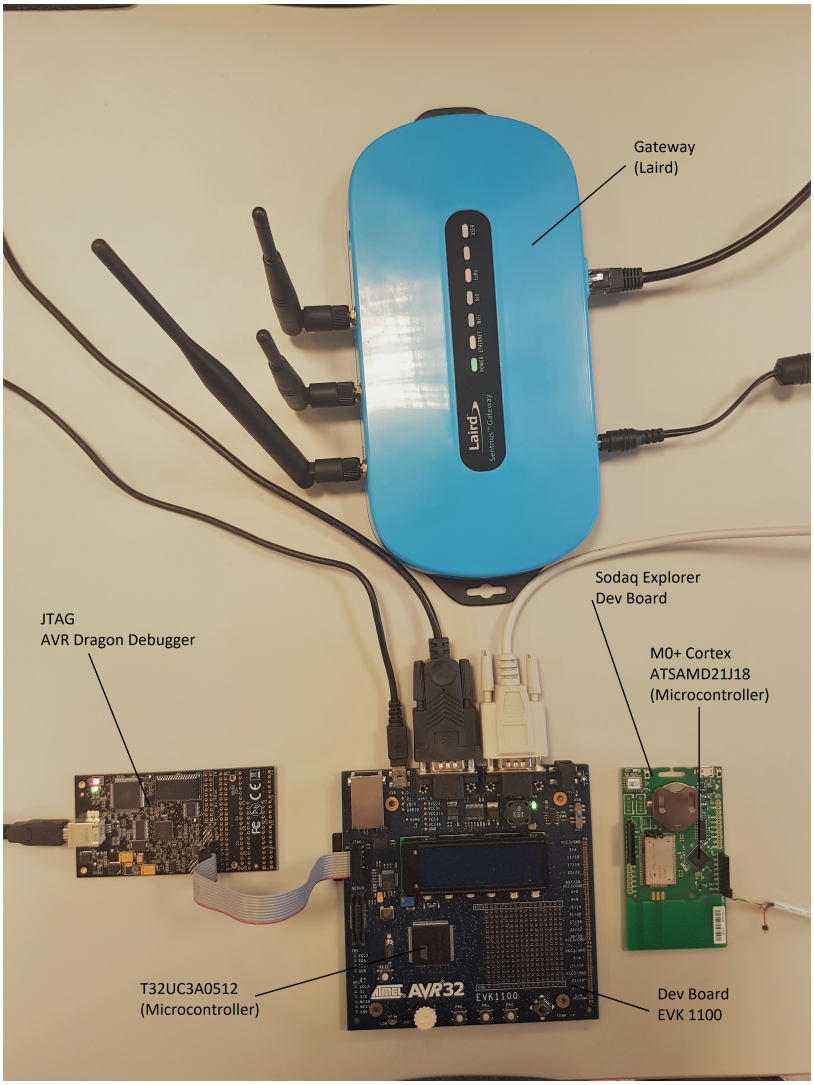

Figure 9.1: Overall representation of the system setup

predetermined error bound. There is another approach, where first few coefficients are saved, and the rest of them removed by calculating the percentage of power. Usually, more 95% of signal power must be transferred. However, this approach increases algorithm complexity and execution time. Size of the compressed data is not considered as for other algorithms, since it results in complex values with half of them being complex conjugates, except for the first one. When sending data, it is necessary to send the position of the coefficient as an integer type, and half of the complex outputs. Output (compressed) data can be of float or double type, since complex conjugates have real parts the same and imaginary ones with opposite sign. Reconstruction takes place on the server side with Inverse Fourier transform.

DCT or Fast DCT (FDCT) is another frequency domain compression algorithm, with essentially the same idea as DFT/FFT. DCT generates coefficients which define the amplitude of cosine of a specific frequency. Operation principle of for fast DCT is taken from [30], with C(99) "fast-dct-lee files". Lee's fast DCT is a recursive variant that similarly as FFT, splits the initial vector into halves until only pairs

of points are left. Recursive calculation of coefficients propagates backward and gives a vector that is the same size as the initial one with DCT coefficients. DCT output coefficients is a real numbers of type double, and it is later converted to float type with *casting* macro. Compression is accomplished in the same manner as in FFT algorithm, by removing insignificant coefficients. Removal of the coefficients is done by checking the range of normalized DCT coefficients, since the Fast DCT Lee transform returns non-normalized values. Normalization is done by multiplying first term by $\frac{1}{\sqrt{2}}$, and rest of the vector with $\sqrt{\frac{2}{N}}$, where N is the initial vector size.

FR is recently developed time-domain algorithm, which also requires a record of entire data set. European Space Agency has deployed this algorithm in some of its applications [33]. The time complexity of this algorithm is of the order of N. There are two nested loops: external with limit on $log_2 N$, and internal one increasing number of iterations from 1 to $log_2 N$-1. Compression is achieved by allowing a relative percentage error with respect to the amplitude range of the measured signal. This allowed error margin simultaneously act as some filter since it removes some of the *noise* from the data set. However, there are no convenient studies regarding filtering properties of the method, but it is interesting to see this unintentional visual effect. Also, reconstruction on the server side is unnecessary since the data can be represented on graphs with ordinary and straightforward linear interpolation.

CA is time-series algorithm that does not wait for data points to be saved and then processed. Thus, it is one of rare methods exploiting RT performance. The decision for saving next data point is based on critical aperture window 8.3. There is a small modification to the algorithm, in a way that it always stores M^{th} data point counting from the current referent point [9], if there is no point to be added.

$$M = \frac{N}{32} \tag{9.1.1}$$

where N is number of the samples in original data set. The modification is necessary since CA algorithm adds to much error in signals that are smooth and linear for a very long period. This behavior appears since the algorithm compares window of slope error tolerance, that at large *distances* all of the decaying points would be in that error bound, and further exclude them from reconstruction vector. This is a simple way of patching the algorithm, but another more reliable solution can be devised for future improvements. Those improvements, on the other hand, make the computational complexity of CA higher than the current complexity of $\mathcal{O}(N)$. CA acts as a filter, and it does not require reconstruction on the server side. Hence, it is sufficient to obtain linear interpolation.

9.2 Data Collection

Data used for the system setup have already been recorded and possessed by the company(industry). The reason is that comparison between the algorithms by using temporary sensory data would introduce uncertainties. Thus, confidentiality would be lost due to presence of noise in measurement process, and input data would never be the same. There are three types of the signal observed: current, voltage,

[9] once the data is stored it becomes referent point for the upcoming samples

and temperature. Sampling process had been performed simultaneously, with the frequency 1Hz, at the same time-stamp for each of the three signals.

9.3 Lora Limitations

As an emerging IoT protocol, LoRa has its pros and cons. Being able to send data over a long distance with low power consummation takes its toll in having the small bandwidth. Background 4 and introduction 1 chapter cover some of LoRa limitations regarding distance and number of channels. For this thesis, limitations of data rate, packet size, and up-link and down-link airtime are of greater importance. Mentioned limitations are defined by the things network (TTN) fair access policy, and they are different for the different geographical region of use. The focus is on European standard that operates in EU 863-870 MHz ISM (International Scientific Medical) radio band, which limits the duty cycle to a maximum of 1%. With duty cycle limitation, airtime of end device is also limited, and it is affected by the number of channels gateway has and the number of devices communicating with it. Up-link airtime is calculated by formula 9.3.1 and defines how much seconds can a device transmit a day where D is duty cycle. Furthermore, LoRa has a sub-band duty cycle limitation that does not allow a device to send data for certain amount of time, which is calculated by formula 9.3.2. Some experts include some practical ways of using the network on Things Network forum [34] and additional information about LoRaWAN technology can be found in [35] where transmission uncertainty is mentioned with a probability of successful transmission lower than 15%. LoRa can be configured for specific use in its physical bounds by selecting transmission parameters [36].

$$T_{air} = \frac{Num_{channels} \cdot 86400 \frac{sec}{day} \frac{D}{100\%}}{Num_{devices}} \tag{9.3.1}$$

$$T_{off} = (\frac{T_{air}}{D}) - T_{air} \tag{9.3.2}$$

9.4 Memory

Memory issues arise if a certain device or application generates vast data that requires processing. In all of the applications that require LoRa protocol, memory necessity has the limits. The culprit for memory limitation requirements is LoRa's airtime limitation and limitations in CR factor of deployed compression algorithms. There are several more dependencies to consider, where duty cycle and number of devices dictate the maximum amount of LoRa's airtime. Spreading factor and data transfer time will affect the limit of the amount of data that can be sent, while including a CR factor with a deployed compression algorithm. Calculations for amount of data which can be sent with LoRa is given by formula 9.4.1, where N is the size of data in B, t is the duration time of one transmission and n(t) is the amount of data excluding 14 B application payload.

$$N = \frac{T_{air}}{t} \cdot n(t) \cdot CR; \tag{9.4.1}$$

Example calculation will provide how to calculate the amount of data that can be sent daily over LoRa, under assumption that some parameters are fixed. With default settings for LoRa communication we have duty cycle

$$D = 0.35\%$$

and assumed number of devices for calculating T_{air} with 9.3.1 is 250 devices. After applying the formula, amount of time when we can send data is

$$T_{air} = 9.6768 \frac{sec}{day}$$

For sending a payload with maximum size packets of $240B$ it results in $400ms$ transmission time. If the average CR of an algorithm is 150, by 9.4.1 we find that amount of bytes we can send in one day is

$$N = 870912 \frac{B}{day}$$

For the application of battery monitoring system, one data point requires $4B$ for float and 2 B for unsigned short. This yields

$$N_d = \frac{N}{6} = 145152 \frac{points}{day}$$

These points can either be distributed and generated over the period of one day, or in smaller time slots. With this restriction, a signal can be sampled with a sampling frequency of

$$F_s = \frac{145152}{86400} = 1.68Hz$$

,if it works for a whole day. This results in $870912B$ that can be sent daily.

10 Results

Testing was obtained with recorded sensor data provided by the company. Direct current, voltage, and temperature were measured on Cyclon RF008, 5Ah 2V battery, where first 16384 samples were taken and compared to a compressed version of the data.

Power consumption was measured on Sodaq Explorer board, where LoRa module is placed. We measured current consumption of the board for different amounts of data packets sent. That made the device change its airtime, and thus changing the amount of current drawn for different size of data packets. The current was not measured for the whole data set, since it would require much time. LoRa limitations do not allow the device to establish up-link after three packets have been sent for T_{off} time. It is calculated by formula 9.3.2. Current profile for specific data packet size is always the same, with the same T_{off} and T_{air}[10] periods. Thus it is a practical solution to measure the current for only one data transfer, and afterward calculate the amount of current that would have been consumed after the transmission of whole data set. Current profile is acquired by portable digital oscilloscope, that is connected to a shunt resistor of 18.4Ω, which generates a voltage drop. To obtain actual current value in Amps, the acquiesced signal was exported from PicoScope software and processed in `MATLAB`. Measured voltage was divided by the resistance value to get the current value. Afterward, the signal was execrated into two parts, a part where data is transmitted (figure 10.1), and a part where the device is dormant (figure 10.2). Current consummation was calculated by integrating the signal over time, from the start of one transmission to the start of the next transmission. We conclude that different size of data packets correspond to different air times. Distinctive data packet sizes were chosen, 254 B and 100 B, that yield 400 ms and 174.3 ms respectively. It is necessary to mention that 14 B are reserved for communication frame configuration and the rest are useful bytes. To see the effects of power reduction, a plot is created to show power reduction ratio for compression from 0 to 750 in figure 10.3. This was done for both 240 B and 86 B of data. Six bytes are required to send one data point or coefficient, where four bytes represent point value as float type, and two bytes stand for unsigned short value for point position. One transmission, as it can be seen in figure 10.1, is constituted of three transactions on different sub-bands. This means we can send

$$240 \cdot 3 = 720B$$

, or

$$86 \cdot 3 = 258B$$

in one transmission. Considering the node is turned on all the time, power consumption comparison is obtained under consideration that entire data set is going to be sent. Power consumption ratio (PCF) is the ratio of power consumption of entire data set, to the power consumption of the compressed data set, considering rest of the time in idle (OFF) state. It is thoroughly represented by equation 10.0.1 where P_t is power of on transmission P_{idle} is power when the device is idle, N is size of data set, $Data_t$ is amount of data transmitted in one transaction.

[10] Air time is the time during which a broadcasting is occurring

$$PCF = \frac{P_t \frac{N \cdot 6}{Data_t}}{P_t \frac{N \cdot 6}{Data_t \cdot CR} + P_{idle}(N - \frac{N \cdot 6}{Data_t \cdot CR})} \tag{10.0.1}$$

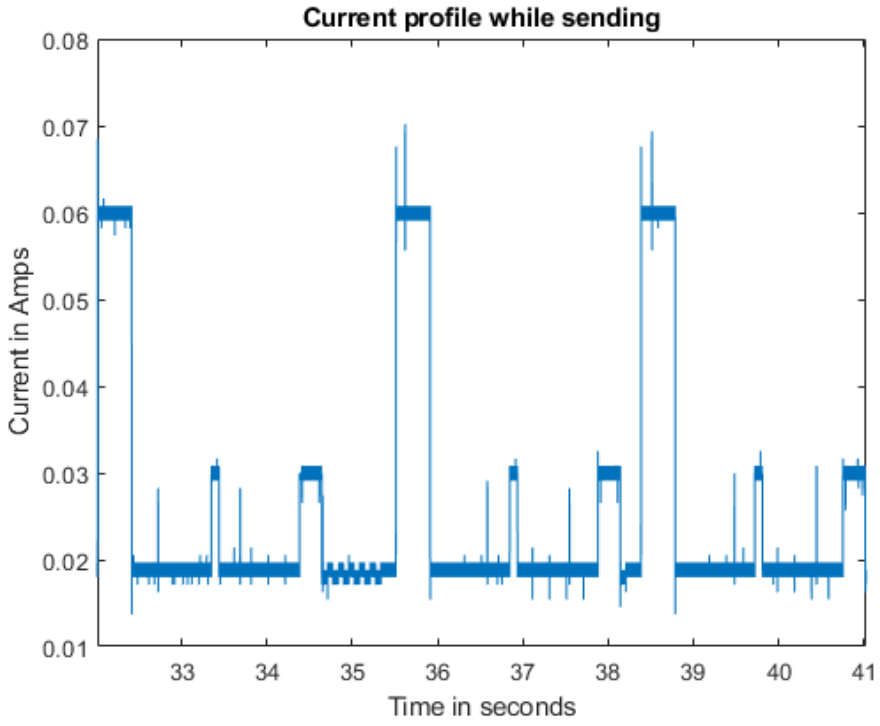

Figure 10.1: Current consumption profile of Sodaq board while sending data with 400 ms air time

In chapter 7 there is a brief discussion on different metrics for evaluation of compression algorithms that can be observed. Figures 10.4 to 10.6 represent dependence of CR factor on PRD value, which are considered crucial for algorithm selection.

The experiment was performed on three types of signals: temperature, voltage, and current. Quality of a transferred signal is usually not defined by the absolute value of its amplitude, but rather with PRD factor. It calculates overall distortion impacted by each point separately, hence en-counting error contribution of every sample. PRD values of 5% are considered excellent, whereas practical applications can be achieved with PRD between 10-15% depending on the use-case. It is important to note that CA and FR exhibit a discontinuity in CR(PRD) function.

The current signal had considerably better CR for CA, for all PRD values. The difference is enormously noticeable for PRD values in range 0% to 5%. Figures 10.7 to 10.10 show plots of the reconstructed current signal, for different compression algorithms, with PRD values of 5% [11]. Other parameters go in favor of FR although PRD for it was 6.4598%. Figure 10.9 visually proves that it has the closest reconstruction with table 1 showing the closest values for every parameter compared to the original signal. Frequency domain algorithms as it can be seen do not perform well on signals that have sharp, sudden changes since to reconstruct them to a 5%

[11] It is very hard to accomplish exact desired value of PRD

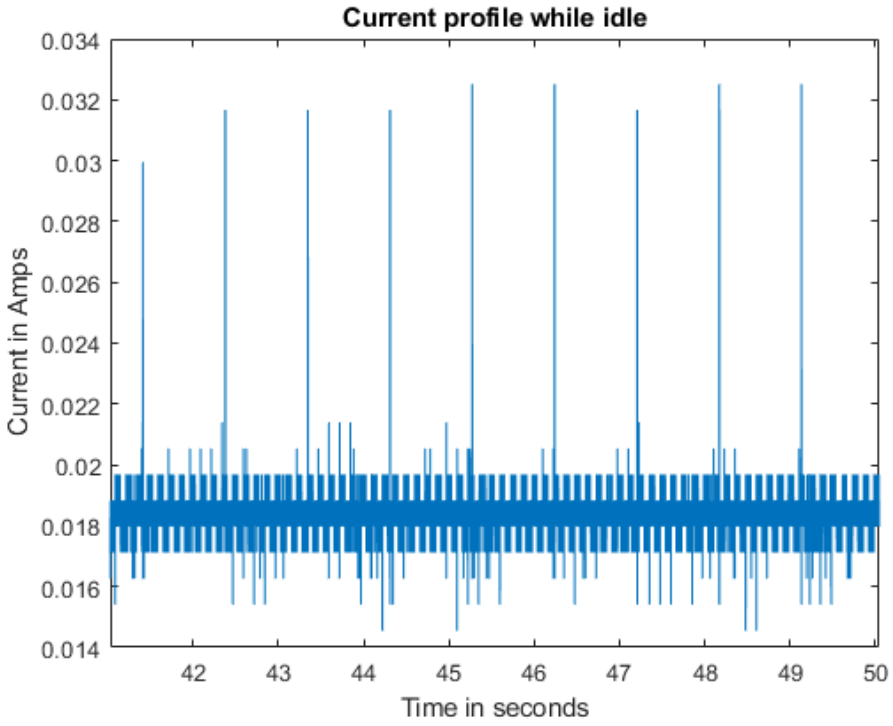

Figure 10.2: Current consumption profile of Sodaq board while idle

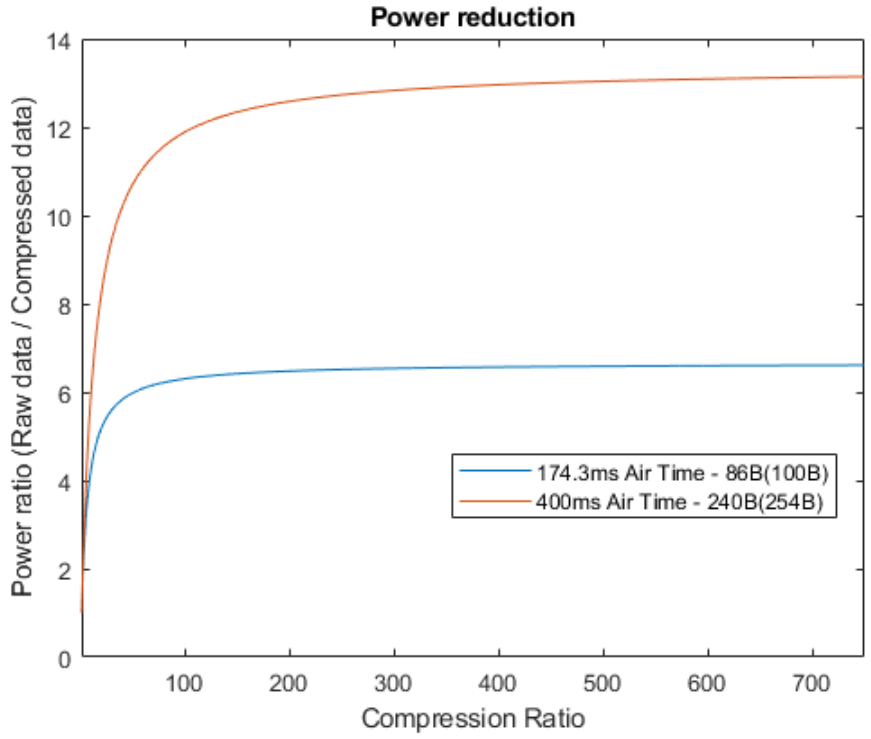

Figure 10.3: Power reduction ratio for different packet size and air times for Sodaq board

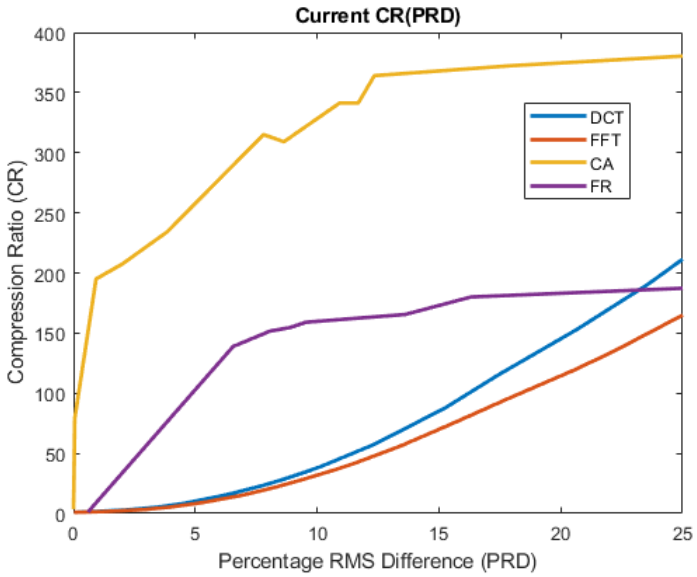

Figure 10.4: CR - PRD dependence for current signal

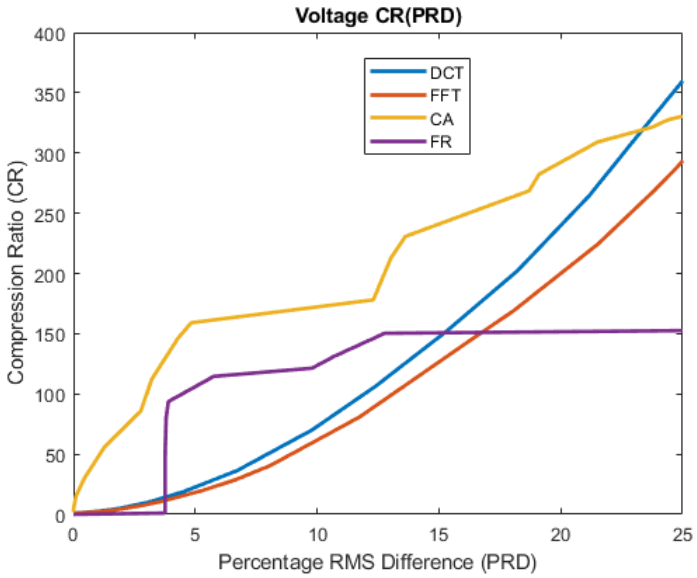

Figure 10.5: CR - PRD dependence for voltage signal

PRD value requires a lot more coefficients to be saved. The behavior of deployed algorithms was anticipated.

When analyzing the voltage signal, CA again exhibited the best results, especially for the compression of high order of precision. Time series voltage signal is smooth and flat in most places. Thus, it is better compressed with time domain algorithms. However, frequency domain algorithms demonstrated improvement in other parameters that were closest original ones, as shown in table 2. Figures 10.11 to 10.14 show visual representation of reconstructed voltage signal, along deployed algorithms with PRD values close to 5%.

Temperature signal exhibited an oscillating wave-like behavior enclosed with a

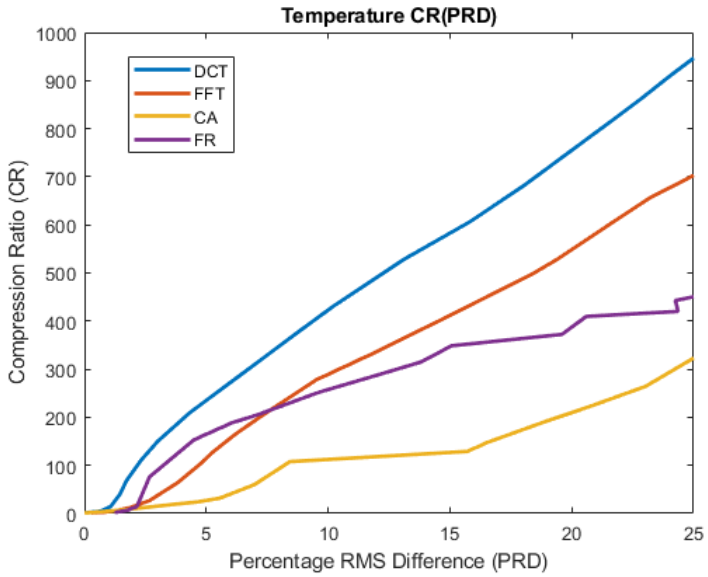

Figure 10.6: CR - PRD dependence for temperature signal

.	Original	FR	CA	DCT	FFT
PRD	NaN	6.4598	5.8554	5.0064	5
Normalized STD	6.602	6.5994	6.6681	6.5935	6.5936
Skewness	-0.0817	-0.0836	-0.0912	-0.0793	-0.0794
Kurtosis	2.719	2.7183	2.6503	2.7276	2.726
Mean	0.1783	0.1779	0.2632	0.1777	0.1777
Median	-1.0165	-1.0161	-1.0165	-0.9926	-0.9948
CR	NaN	121.37	273.067	10.0763	8.1472
$CR_P[\%]$	NaN	99.1761	99.6388	90.0757	87.7258

Table 1: Table comparing parameters of original current signal to reconstructed signal with different compression algorithms (Sample size = 16384)

.	Original	FR	CA	DCT	FFT
PRD	NaN	4.9367	4.877	4.9997	5.0017
Normalized STD	0.2051	0.2062	0.2071	0.2049	0.2049
Skewness	1.2903	1.2653	1.2275	1.2893	1.2902
Kurtosis	3.7975	3.7379	3.6861	3.783	3.7876
Mean	2.1246	2.1243	2.1234	2.1246	2.1246
Median	2.0686	2.0675	2.0676	2.0681	2.068
CR	NaN	113	159.068	22.2005	18.0639
$CR_P[\%]$	NaN	99.115	99.3713	95.4956	94.4641

Table 2: Table comparing parameters of original voltage signal to reconstructed signal with different compression algorithms (Sample size = 16384)

dose of noise. As expected, frequency domain algorithms achieved better performance. DCT showed the highest CR, followed by FR. Parameters can be seen in

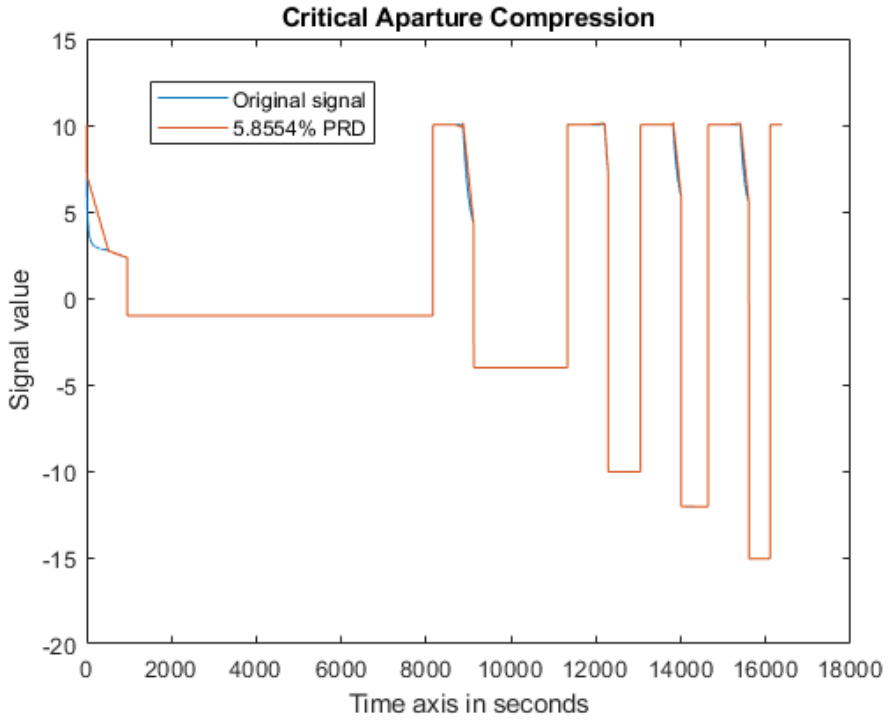

Figure 10.7: Reconstruction of current data compressed by CA, with amperes(A) on Y-axis and PRD close to 5%

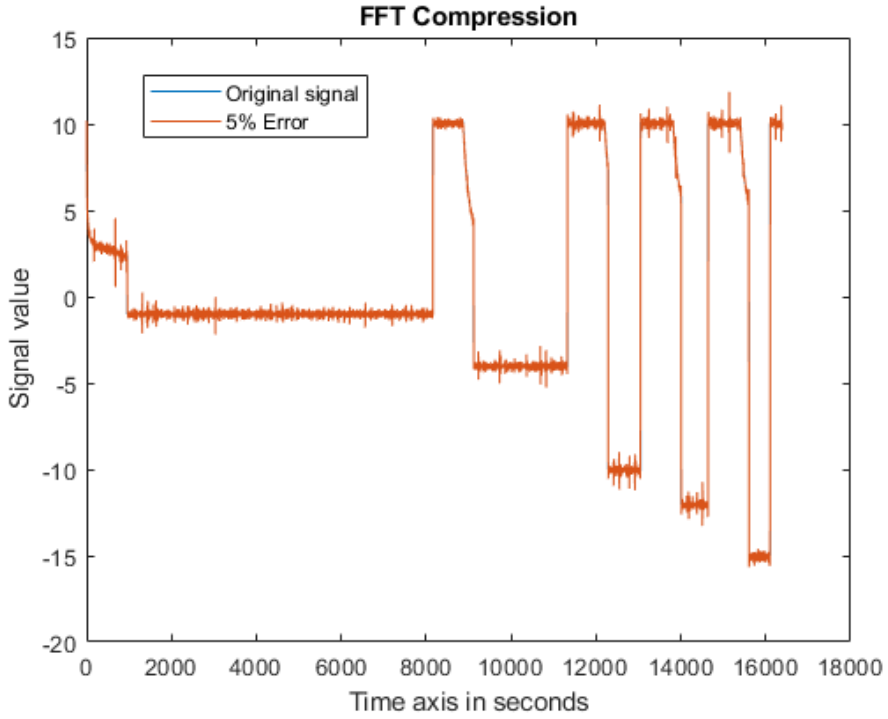

Figure 10.8: Reconstruction of current data compressed by FFT, with amperes(A) on y-axis and PRD close to 5%

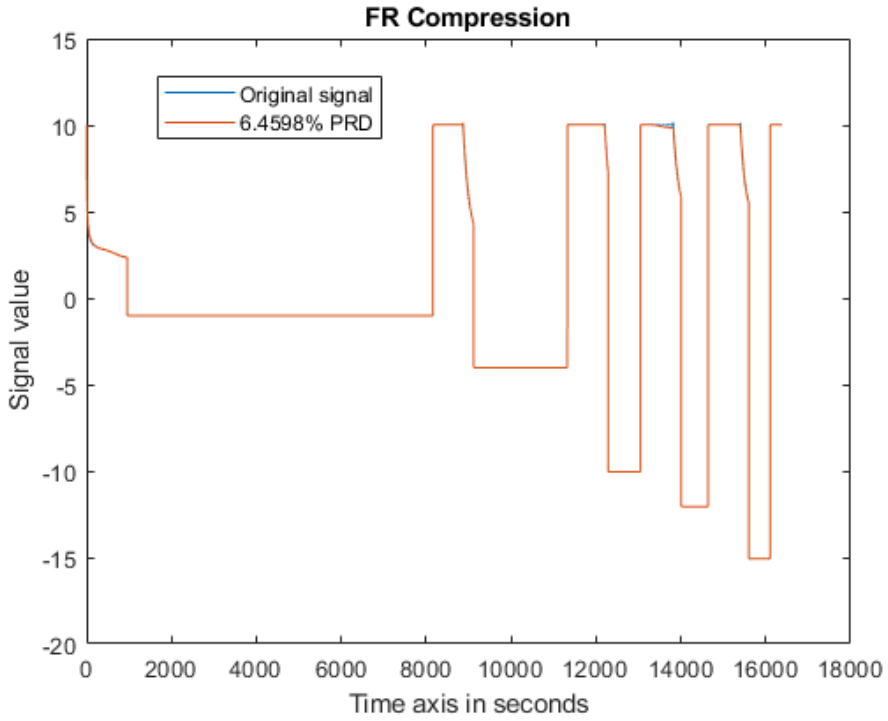

Figure 10.9: Reconstruction of current data compressed by, with amperes(A) on y-axis and PRD close to 5%

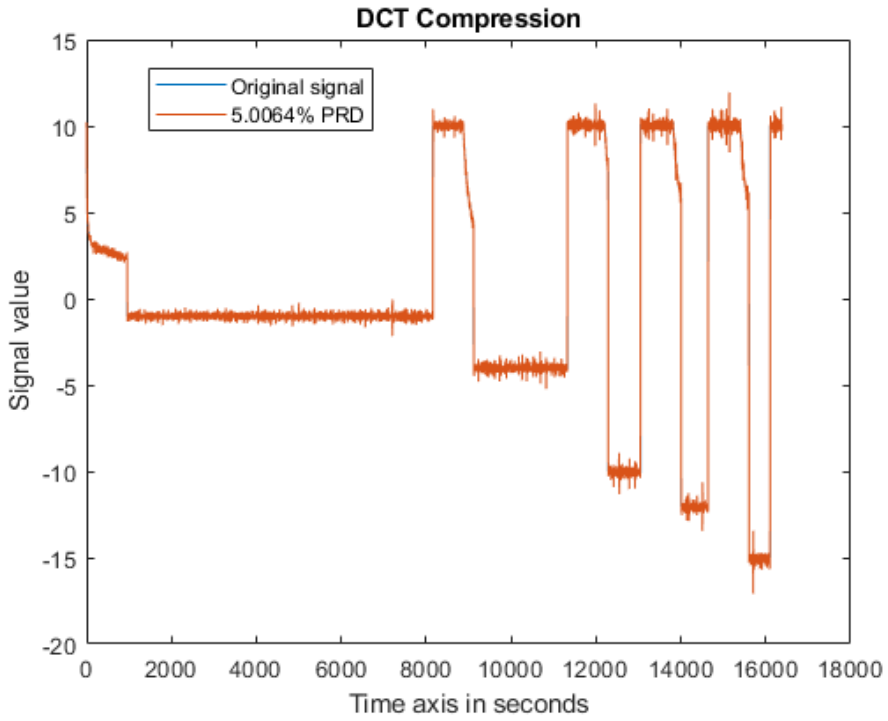

Figure 10.10: Reconstruction of current data compressed by DCT, with amperes(A) on y-axis and PRD close to 5%

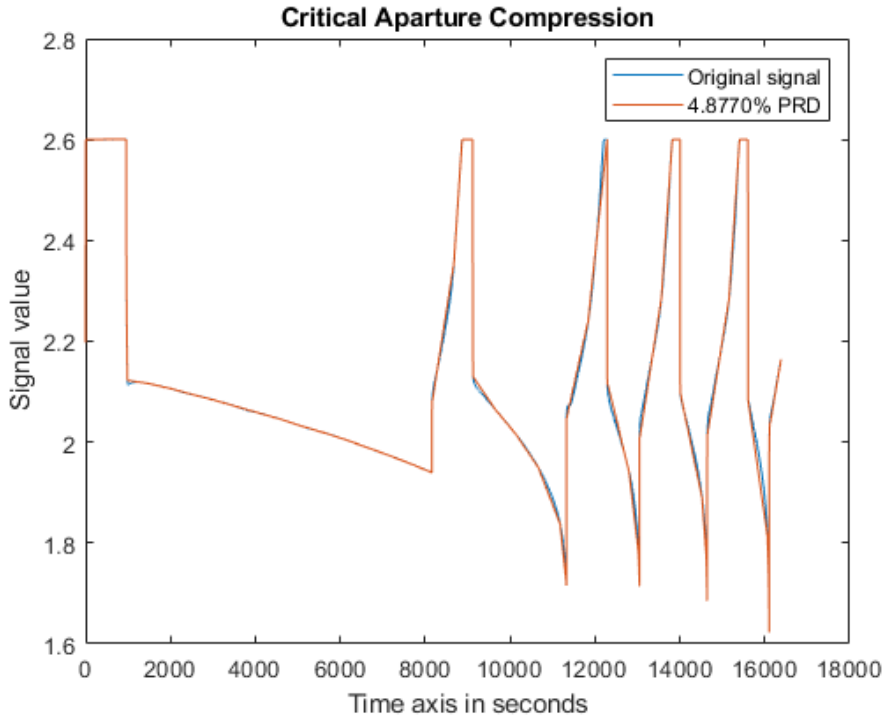

Figure 10.11: Reconstruction of voltage data compressed by CA, with volts on y-axis and PRD close to 5%

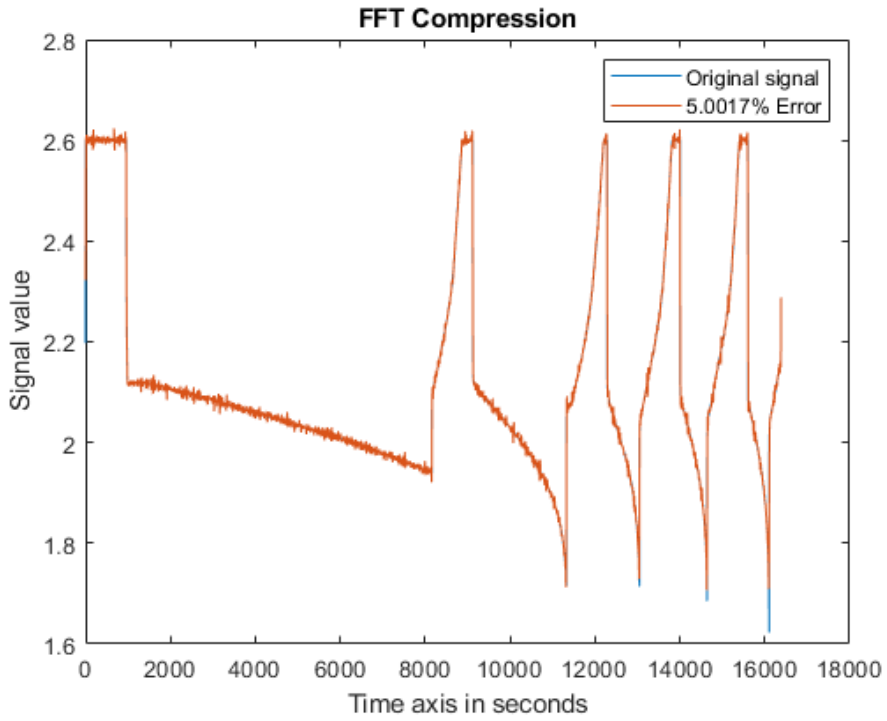

Figure 10.12: Reconstruction of voltage data compressed by FFT, with volts on y-axis and PRD close to 5%

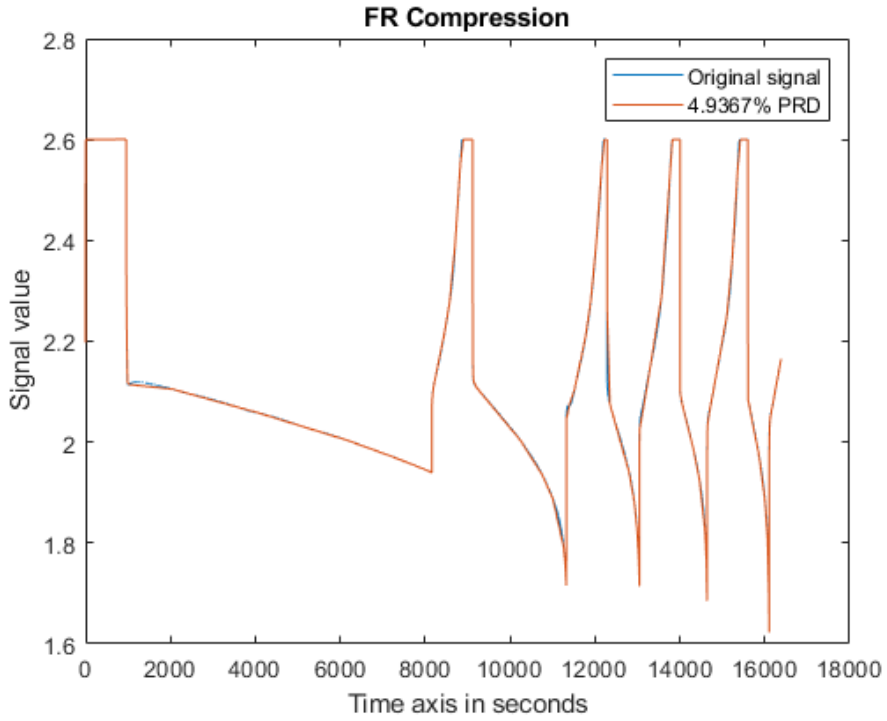

Figure 10.13: Reconstruction of voltage data compressed by FR, with volts on y-axis and PRD close to 5%

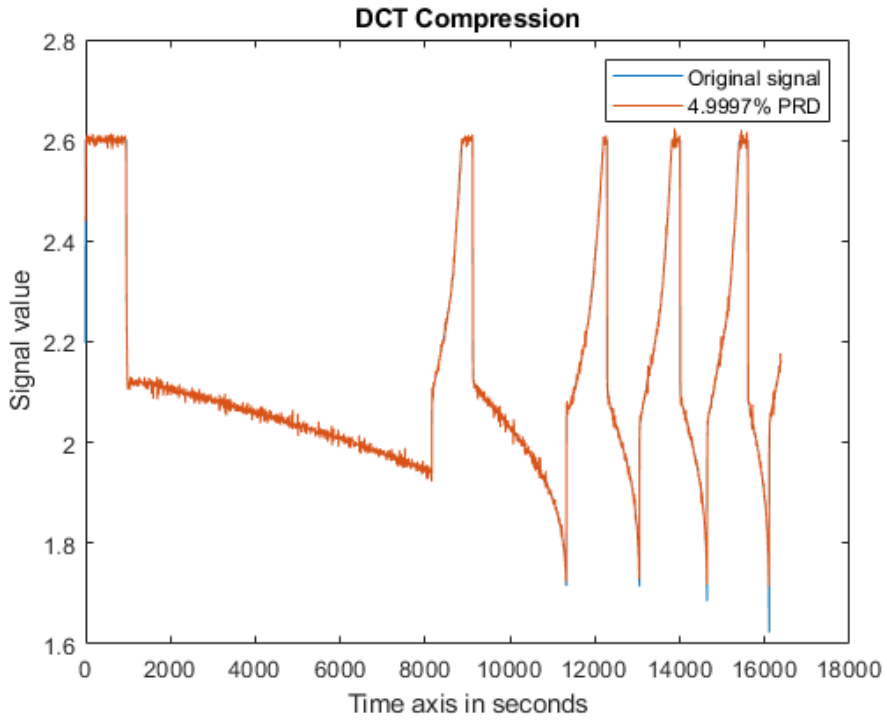

Figure 10.14: Reconstruction of voltage data compressed by DCT, with volts on y-axis and PRD close to 5%

table 9. It is important to notice that CA does not show CR as high as rest of the algorithms. Figures 10.15 to 10.18 are visual representation of temperature signal reconstruction with all of the deployed algorithms.

.	Original	FR	CA	DCT	FFT
PRD	NaN	4.841	4.9405	5.0654	5.0013
Normalized STD	2.0798	2.0743	2.0752	2.077	2.0771
Skewness	0.5023	0.4989	0.5152	0.5037	0.4977
Kurtosis	2.6638	2.6494	2.6552	2.6761	2.6659
Mean	25.7898	25.7848	25.8067	25.7896	25.7896
Median	25.426	25.4099	25.4307	25.4551	25.4429
CR	NaN	168.9175	27.2159	237.4493	114.5734
CR[%]	NaN	99.408	96.3257	99.5789	99.1272

Table 3: Table compares parameters of original temperature signal to reconstructed signal for different compression algorithms (Sample size = 16384)

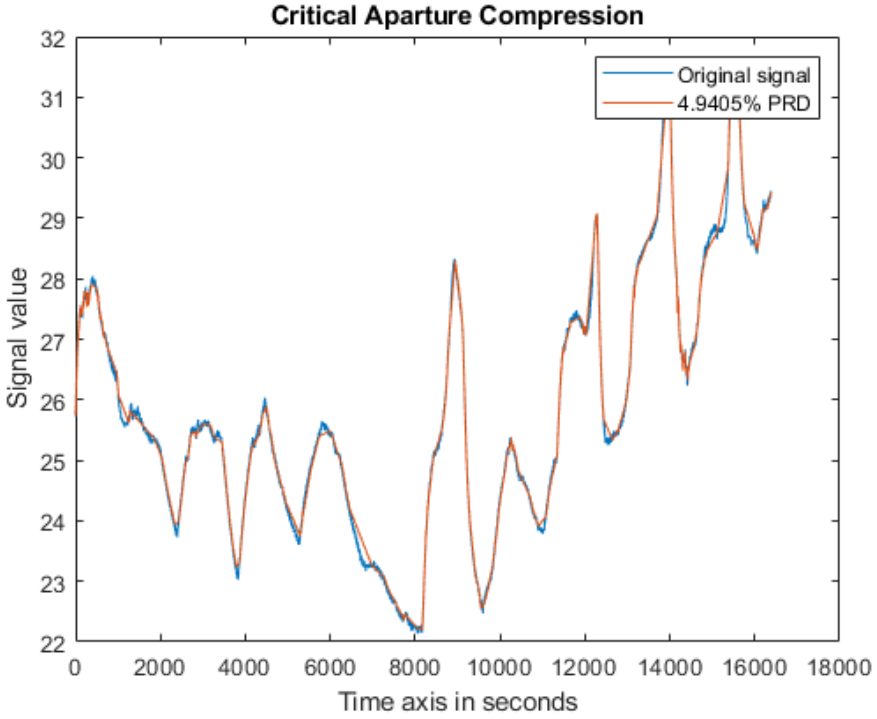

Figure 10.15: Reconstruction of temperature data compressed by CA, with celsius on y-axis and PRD close to 5%

This chapter provided evaluation of the algorithms in terms of similar PRD values since as explained in subsection 8.1 compares the compression algorithms in the same level of precision. In practical terms error bounds of these algorithms is not defined in terms of PRD since it is impractical, but in absolute values as depicted in section 8 and subsection 9.1. Effects of compression and reconstruction can be seen in figures and tables presented in appendix A.

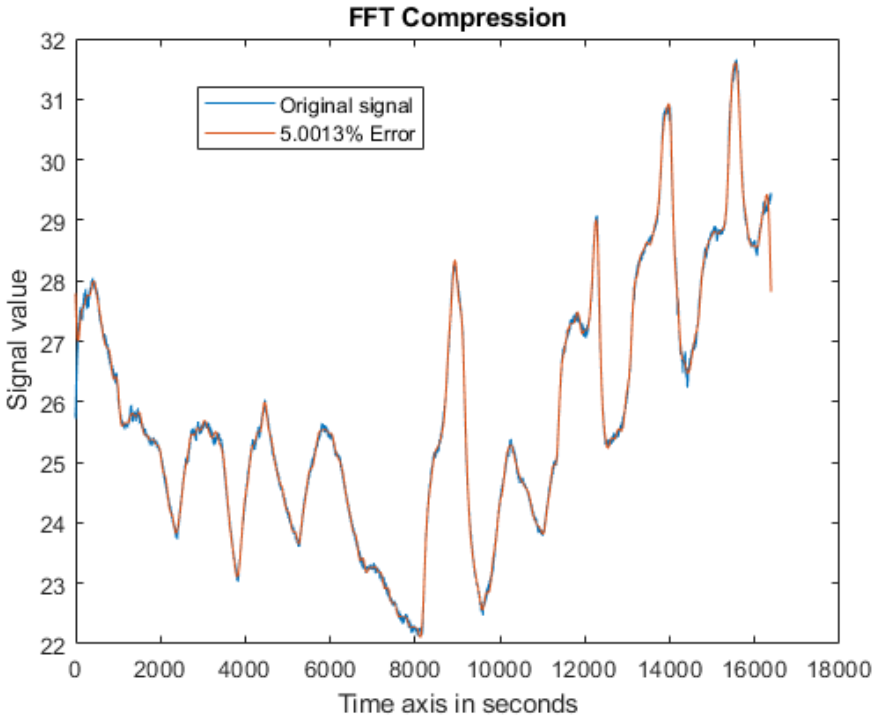

Figure 10.16: Reconstruction of temperature data compressed by FFT, with celsius on y-axis and PRD close to 5%

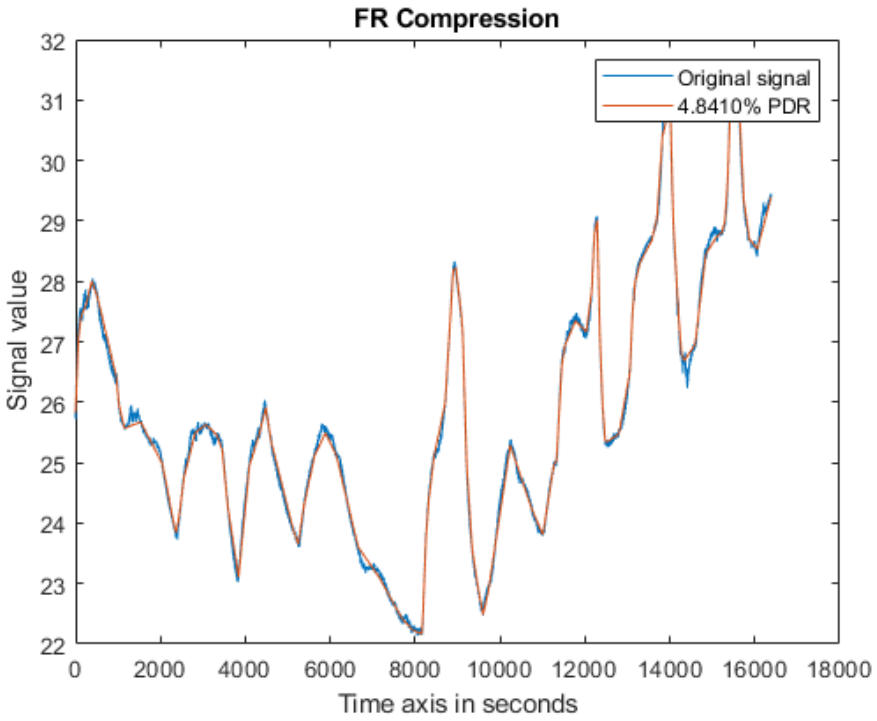

Figure 10.17: Reconstruction of temperature data compressed by, with celsius on y-axis and PRD close to 5%

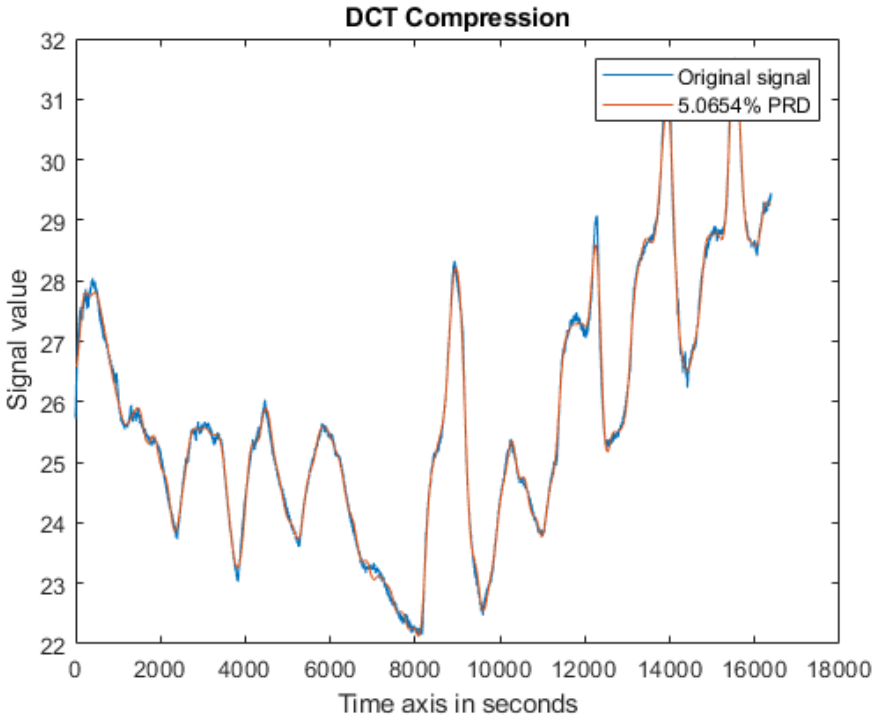

Figure 10.18: Reconstruction of temperature data compressed by DCT, with celsius on y-axis and PRD as close to 5%

11 Discussion & Conclusion

In this thesis, four lossy compression algorithms are deeply investigated. Compression was performed on battery data signals. The signals that were measured on the battery are current, voltage and temperature. Algorithms were tested for power reduction while sending raw and compressed data via wireless LoRa protocol. This IoT protocol has a long range-low power transmission, and it has an ultra narrow bandwidth. Narrow bandwidth limits the speed of data transfer and indirectly increases power consumption. This is why compression is needed. Regardless of the compression algorithm used, CR factor dictates power savings of the transceiver. We have shown that different sensor signals yield different CR for the different compression algorithm. Power reduction asymptotically approaches saturation, and cannot give better power savings with a larger value of CR. Saturation is achieved for CR around 200, regardless of the airtime spent by LoRa transceiver. When looking at CR(PRD) graphs in Figures 10.4 to 10.6 allowing 5% PRD, we notice that CA and FR are suitable for compression of current and the voltage signal, and DCT, FR, and FFT could be selected for use in compression of temperature signals. If we assume all of the parameters are important to bring a decision on the most efficient algorithm, we claim that FR is the best option. It can universally be used for any signal, yields considerable power savings and results in high CR, with minimal loss of data fidelity.

None of the algorithms would considerably improve power consumption, since FreeRTOS is always active and gathers data from a sensor, making computational

consumption negligible. The computational complexity of algorithms ranges from $\mathcal{O}(N)$ for CA and FR and $\mathcal{O}(NlogN)$ for FFT and DCT. An important notice is that CA is the only implemented algorithm in this thesis which performs RT calculations.

As mentioned in 9.3 LoRa protocol limitations restrain the Embedded system from RT capabilities. Depending on the application and the reliability of the system, memory expansion is optional depending on the use case as explained in subsection 9.4.

Future work could be done in exploring the new state of the art algorithms with a different set of signal types. Some improvements could also be developed for specific application of algorithms on specific signal types. One idea of ours was to test multiple compression schemes on the same data set to see if additional compression of already compressed data would yield any improvements in CR. A way this could be achieved is to apply lossy compression on the data set and then obtain a lossless compression.

References

[1] T. Bose, S. Bandyopadhyay, S. Kumar, A. Bhattacharyya, and A. Pal, "Signal characteristics on sensor data compression in IoT - an investigation," in *IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, London, UK, June 2016, pp. 1–6.

[2] Long-range iot on the road to success. [Online]. Available: http://www.electronicdesign.com/embedded-revolution/long-range-iot-road-success

[3] Lorawan. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/

[4] Osi model. [Online]. Available: https://searchnetworking.techtarget.com/definition/OSI

[5] T. Burden. Chirp technology. [Online]. Available: https://www.westmarine.com/WestAdvisor/Understanding-CHIRP-Scanning-Sonar

[6] Sigfox technology overview. [Online]. Available: https://www.sigfox.com/en/sigfox-iot-technology-overview

[7] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, September 2011.

[8] J. Ma, T. Zhang, and M. Dong, "A novel ECG data compression method using adaptive fourier decomposition with security guarantee in e-health applications," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 3, pp. 986–994, May 2015.

[9] C. J. Deepu, C. H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for IoT," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, April 2017.

[10] A. Moon, J. Kim, J. Zhang, and S. W. Son, "Lossy compression on IoT big data by exploiting spatiotemporal correlation," in *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, USA, September 2017, pp. 1–7.

[11] B. R. Stojkoska and Z. Nikolovski, "Data compression for energy efficient IoT solutions," in *25th Telecommunication Forum (TELFOR)*, Belgrade, Serbia, November 2017, pp. 1–4.

[12] F. Marcelloni and M. Vecchio, "A simple algorithm for data compression in wireless sensor networks," *IEEE Communications Letters*, vol. 12, no. 6, pp. 411–413, June 2008.

[13] C. J. Deepu, C. H. Heng, and Y. Lian, "A hybrid data compression scheme for power reduction in wireless sensors for IoT," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 2, pp. 245–254, April 2017.

[14] A. Moon, J. Kim, J. Zhang, H. Liu, and S. W. Son, "Understanding the impact of lossy compressions on IoT smart farm analytics," in *IEEE International Conference on Big Data (Big Data)*, Boston, MA, USA, December 2017, pp. 4602–4611.

[15] M. Mishali and Y. C. Eldar, "Xampling: Analog data compression," in *2010 Data Compression Conference*, Snowbird, UT, USA, March 2010, pp. 366–375.

[16] Y. Zigel, A. Cohen, and A. Katz, "ECG signal compression using analysis by synthesis coding," *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 10, pp. 1308–1316, Oct 2000.

[17] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," *SIGMOD Rec.*, vol. 30, no. 2, pp. 151–162, May 2001. [Online]. Available: http://doi.acm.org/10.1145/376284.375680

[18] K. Konstantinides and B. K. Natarajan, "An architecture for lossy compression of waveforms using piecewise-linear approximation," *IEEE Transactions on Signal Processing*, vol. 42, no. 9, pp. 2449–2454, Sep 1994.

[19] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *International Conference on Data Engineering (Cat. No.03CH37405)*, March 2003, pp. 429–440.

[20] W. R. Dieter, S. Datta, and W. K. Kai, "Power reduction by varying sampling rate," in *International Symposium on Low Power Electronics and Design*, August 2005, pp. 227–232.

[21] D. D. Testa and M. Rossi, "Lightweight lossy compression of biometric patterns via denoising autoencoders," *IEEE Signal Processing Letters*, vol. 22, no. 12, pp. 2304–2308, December 2015.

[22] U. Lachish, "Thermoelectric effect peltier seebeck and thomson," *Guma Science*, vol. 12, August 2017.

[23] M. A. Dokic, *Signals and Systems.* Faculty of Electrical Engineering Sarajevo, 2010, vol. 410.

[24] R. Hoffman, *"Data Compression in Digital Systems".* Chapman Hall, 1997, vol. 423.

[25] S. D., *A Concise Introduction to Data Compression.* Springer, 2008, vol. 314.

[26] R. M. N. IAN H. WIllEN and J. G. CLEARY, "Thermoelectric effect peltier seebeck and thomson," *Communications of the ACM*, vol. 30, June 1987.

[27] I. M. Pu, *Fundamental Data Compression*. Elsevier, 2006, vol. 269.

[28] S. Friedental. Ge proficy historian data compression. [Online]. Available: http://www.evsystems.net/

[29] Andrew F. Cheng *et al.*, "Data compression using chebyshev transform," *United States Patent*, vol. 13, no. 1, pp. 1–13, July 2007.

[30] B. G. Lee. Fast discrete cosine transform. [Online]. Available: https://www.nayuki.io/res/fast-discrete-cosine-transform-algorithms/fast-dct-lee.c

[31] The things network. [Online]. Available: https://www.thethingsnetwork.org/

[32] Sodaq explorer support. [Online]. Available: http://support.sodaq.com/sodaq-one/lorawan-starter-kit/

[33] T. F. Jos-Antonio Martnez-Heras and A. Donati. Fractal resampling, europian space agency. [Online]. Available: http://www.esa-tec.eu/space-technologies/from-space/fractal-resampling-time-series-compression/

[34] Arjan. The things network limitations. [Online]. Available: https://www.thethingsnetwork.org/forum/t/

[35] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, September 2017.

[36] M. Bor and U. Roedig, "Lora transmission parameter selection," in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, June 2017, pp. 27–34.

A Graphs & Tables

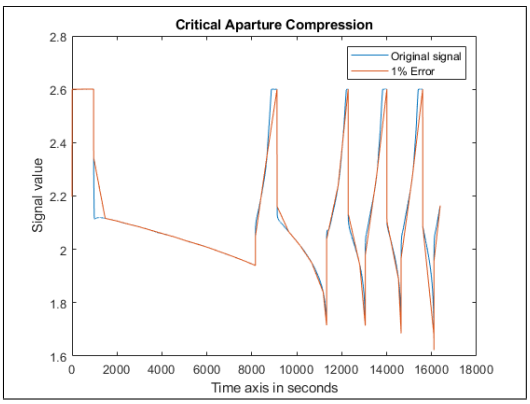

Figure A.1: CA compression of voltage signal with 1% absolute error

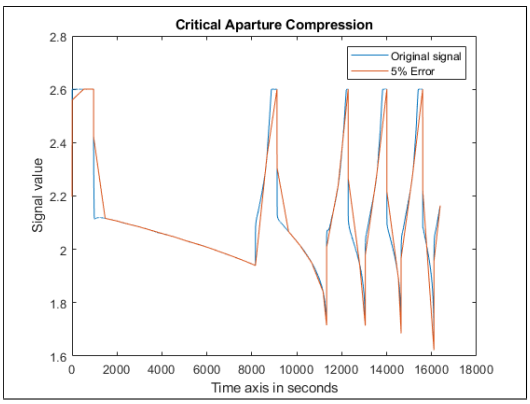

Figure A.2: CA compression of voltage signal with 5% absolute error

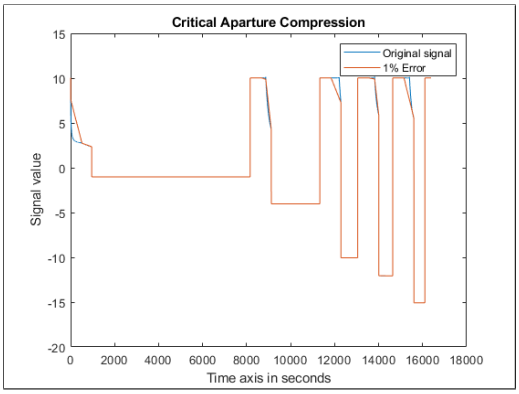

Figure A.3: CA compression of current signal with 1% absolute error

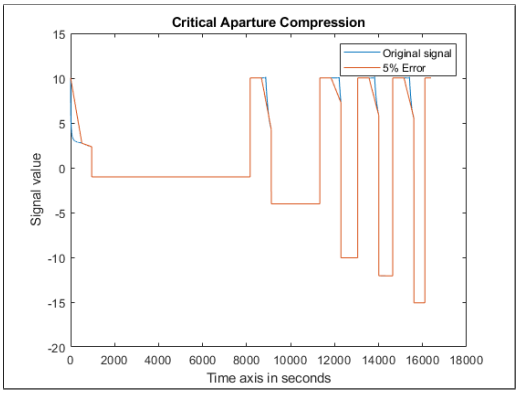

Figure A.4: CA compression of current signal with 5% absolute error

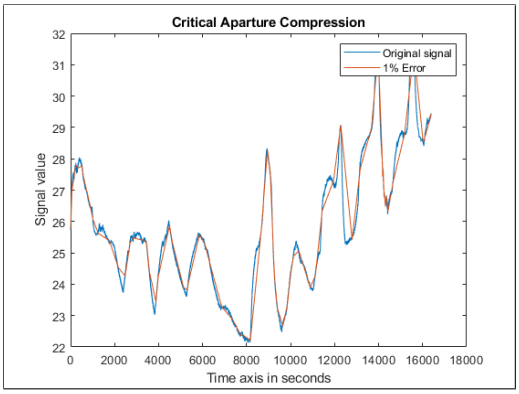

Figure A.5: CA compression of temperature signal with 1% absolute error

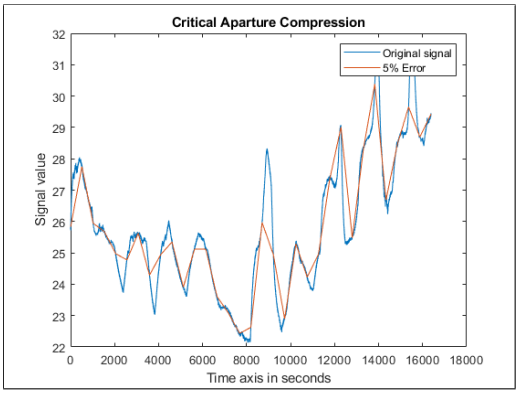

Figure A.6: CA compression of temperature signal with 5% absolute error

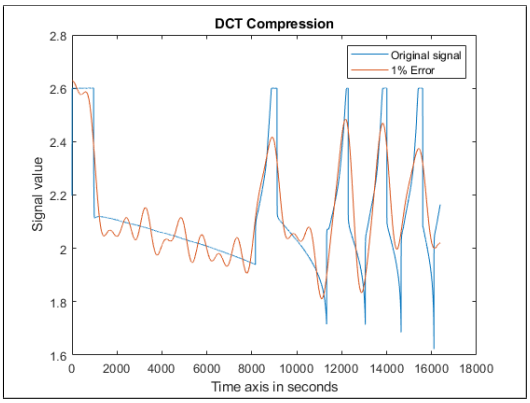

Figure A.7: DCT compression of voltage signal with 1% absolute error (in amplitude spectra)

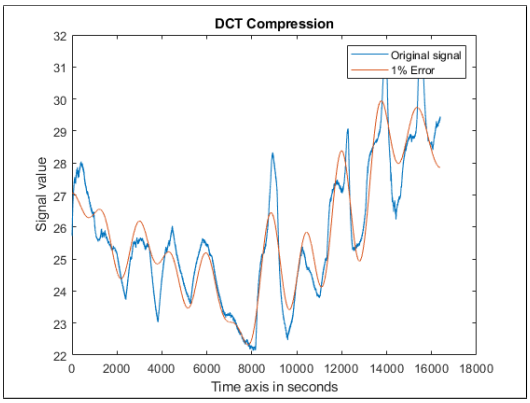

Figure A.8: DCT compression of temperature signal with 1% absolute error (in amplitude spectra)

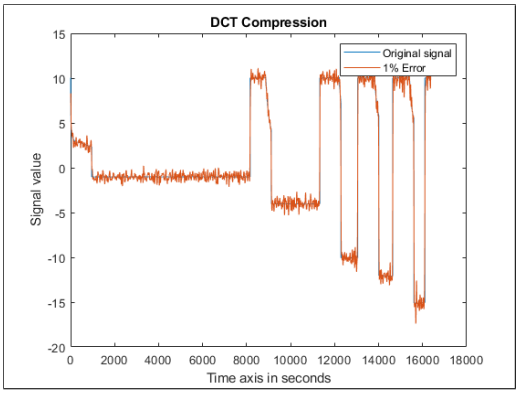

Figure A.9: DCT compression of current signal with 1% absolute error (in amplitude spectra)

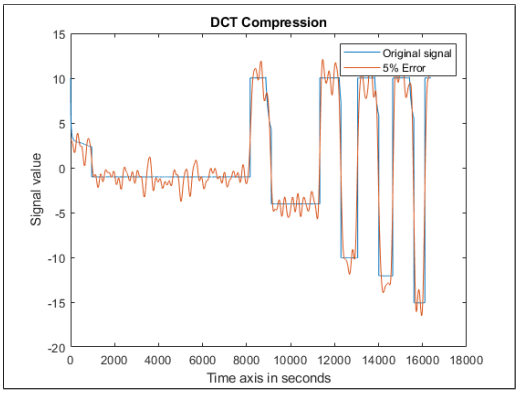

Figure A.10: DCT compression of current signal with 5% absolute error (in amplitude spectra)

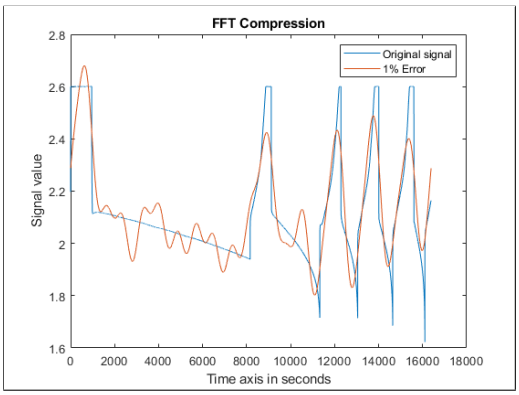

Figure A.11: FFT compression of voltage signal with 1% absolute error (in amplitude spectra)

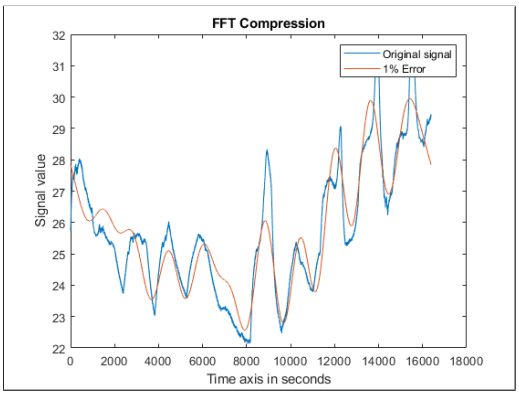

Figure A.12: FFT compression of temperature signal with 1% absolute error (in amplitude spectra)

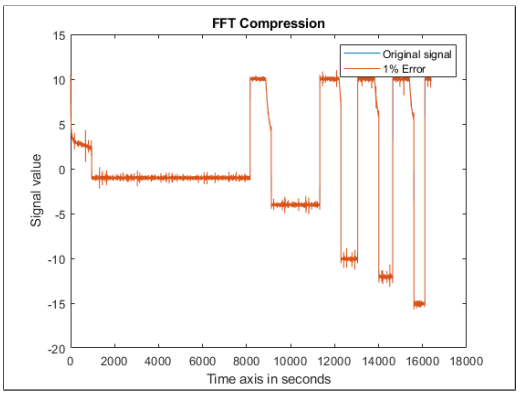

Figure A.13: FFT compression of current signal with 1% absolute error(in amplitude spectra)

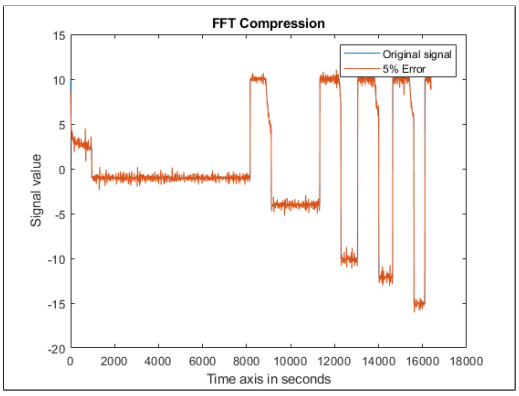

Figure A.14: FFT compression of current signal with 5% absolute error(in amplitude spectra)

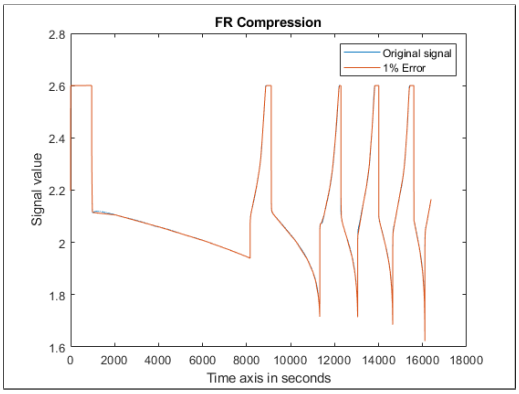

Figure A.15: FR compression of voltage signal with 1% absolute error

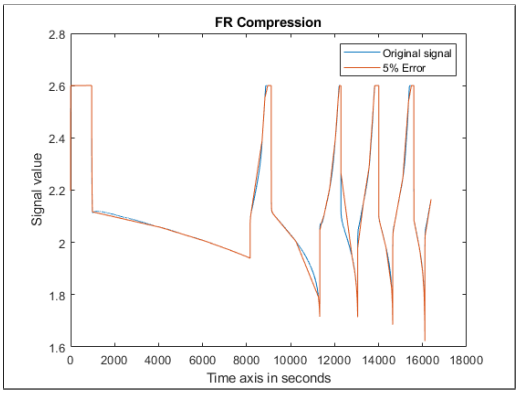

Figure A.16: FR compression of voltage signal with 5% absolute error

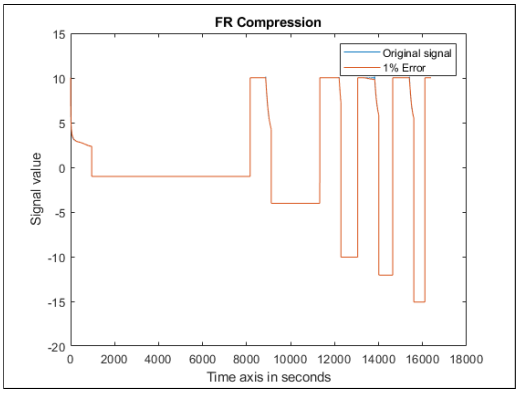

Figure A.17: FR compression of current signal with 1% absolute error

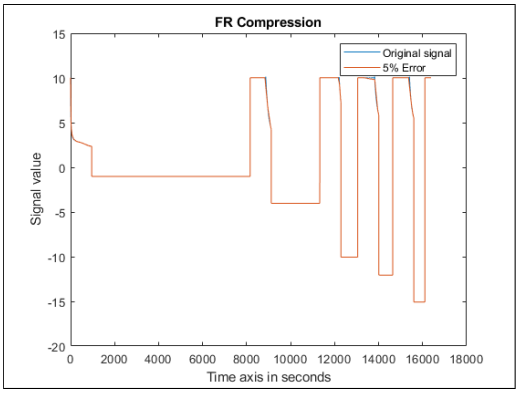

Figure A.18: FR compression of current signal with 5% absolute error

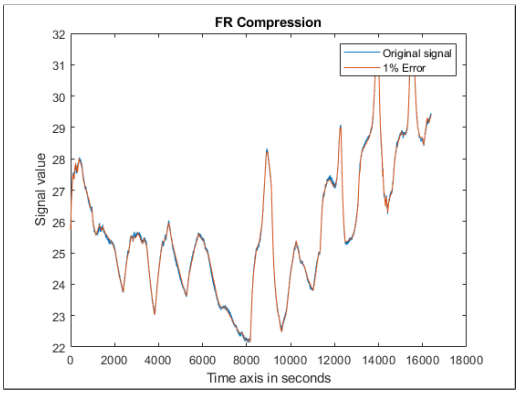

Figure A.19: FR compression of temperature signal with 1% absolute error

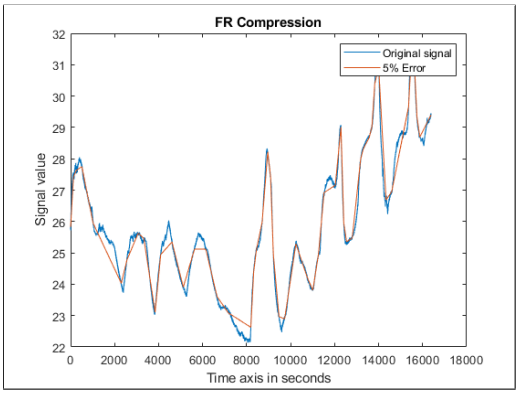

Figure A.20: FR compression of temperature signal with 5% absolute error

.	Original	FR	CA	DCT	FFT
PRD	NaN	3.9577	18.701	48.5236	49.1549
Normalized STD	0.2051	0.2057	0.2009	0.1845	0.1841
Skewness	1.2903	1.2823	1.0608	0.979	0.9211
Kurtosis	3.7974	3.7774	3.6132	3.2194	3.3958
Mean	2.1246	2.124	2.1175	2.1246	2.1246
Median	2.0686	2.0682	2.0649	2.0583	2.0883
CR	NaN	98.1138	268.59	1024	963.765
CR[%]	NaN	98.9808	99.6277	99.9023	99.8962

Table 4: Table compares parameters of original voltage signal to reconstructed signal for different compression algorithms with absolute error bound of 1% (Sample size = 16384)

.	Original	FR	CA	DCT	FFT
PRD	NaN	10.6919	26.4073	X	X
Normalized STD	0.2051	0.2085	0.2006	X	X
Skewness	1.2903	1.19	0.9443	X	X
Kurtosis	3.7974	3.5869	3.4146	X	X
Mean	2.1246	2.1248	2.1231	X	X
Median	2.0686	2.0684	2.0682	X	X
CR	NaN	134.303	348.596	X	X
CR[%]	NaN	99.2554	99.7137	X	X

Table 5: Table compares parameters of original voltage signal to reconstructed signal for different compression algorithms with absolute error bound of 5% (Sample size = 16384)

.	Original	FR	CA	DCT	FFT
PRD	NaN	6.4613	7.8036	12.3472	4.2526
Normalized STD	6.602	6.5987	6.5813	6.552	6.5958
Skewness	-0.0817	-0.0839	-0.1191	-0.075	-0.0798
Kurtosis	2.719	2.7185	2.6911	2.67817	2.724
Mean	0.1783	0.1774	0.1991	0.1777	0.1777
Median	-1.0165	-1.0161	-1.0165	-0.9459	-1.0002
CR	NaN	125.076	315.077	57.4877	6.0213
CR[%]	NaN	99.2005	99.6826	98.2605	83.3923

Table 6: Table compares parameters of original current signal to reconstructed signal for different compression algorithms with absolute error bound of 1% (Sample size = 16384)

.	Original	FR	CA	DCT	FFT
PRD	NaN	6.5534	11.6058	28.8083	9.7467
Normalized STD	6.602	6.5923	6.5388	6.3461	6.5707
Skewness	-0.0817	-0.0867	-0.1369	0.0047	-0.0725
Kurtosis	2.719	2.7203	2.7064	2.9553	2.7405
Mean	0.1783	0.0.1734	0.1713	0.0001	0.1777
Median	-1.0165	-1.0161	-1.0165	-0.968	-0.9691
CR	NaN	138.856	348.596	268.5902	30.2847
CR[%]	NaN	99.2798	99.7131	99.6277	96.698

Table 7: Table compares parameters of original current signal to reconstructed signal for different compression algorithms with absolute error bound of 5% (Sample size = 16384)

.	Original	FR	CA	DCT	FFT
PRD	NaN	2.6738	18.7886	36.977	40.8548
Normalized STD	2.0798	2.0785	2.104	1.9506	1.9252
Skewness	0.5023	0.502	0.5075	0.4167	0.5112
Kurtosis	2.6638	2.6691	2.5958	2.3572	2.3874
Mean	25.7898	25.787	25.812	25.7896	25.7896
Median	25.426	25.4287	25.4192	25.4087	25.5652
CR	NaN	76.2093	190.512	1638.4	1092.3
CR[%]	NaN	98.6878	99.4751	99.939	99.9084

Table 8: Table compares parameters of original temperature signal to reconstructed signal for different compression algorithms with absolute error bound of 1% (Sample size = 16384))

.	Original	FR	CA	DCT	FFT
PRD	NaN	13.8205	34.1922	X	X
Normalized STD	2.0798	2.0484	1.951	X	X
Skewness	0.5023	0.6319	0.4726	X	X
Kurtosis	2.6638	2.6226	2.3324	X	X
Mean	25.7898	25.7808	25.7858	X	X
Median	25.426	25.2043	25.1956	X	X
CR	NaN	315.096	496.485	X	X
CR[%]	NaN	99.6826	99.7986	X	X

Table 9: Table compares parameters of original temperature signal to reconstructed signal for different compression algorithms with absolute error bound of 5% (Sample size = 16384)