Smart IoT Communication : Context Specific Data Pruning in Smart IoT Applications

A THESIS

submitted in partial fulfilment of the requirement for the degree of

Master of Technology

in

Computer Technology

By

Wadood Ahmad Khan

(2017EET2308)

Under the guidance of

Dr. Swades De

Electrical Engineering Department,

Indian Institute of Technology, Delhi

June 2020

CERTIFICATE

This is to certify that the work contained in this thesis titled **"Smart IoT Communication : Context Specific Data Pruning in Smart IoT Applications"** by **Wadood Ahmad Khan** (2017EET2308) in partial fulfilment of the course work requirement of *Master of Technology* in *Computer Technology* program from *Electrical Engineering Department*, Indian Institute of Technology, Delhi is a bonafide work carried out by him under my guidance and supervision. The matter submitted in this thesis has not been admitted for an award of any other degree anywhere unless explicitly referenced.

Signed :
Dr. Swades De
Project Supervisor
Department of Electrical Engineering
Indian Institute of Technology Delhi
Dated :

ACKNOWLEDGEMENT

I would like to thank Dr. Swades De for his continual guidance and support throughout the duration of the project. His continuous mentoring enabled me to constantly move in the right direction that I needed for the completion of this work.

I would also thank my family and friends for their constant support and encouragement throughout my life. Finally, I thank the Almighty, who instills the strength inside me to keep moving forward.

Wadood Ahmad Khan

16 Jun 2020

ABSTRACT

KEYWORDS: Internet of Things, Advanced Metering Infrastructure, Big Data, Smart Meter.

In the age of connected devices i.e., Internet of Things (IoT), a significant area of concern is **Big Data**. Sensing devices installed in an IoT infrastructure generate vast chunks of data which is forwarded to the main system e.g., a server hosted on a cloud platform through a gateway. Smart electricity meter is such an example of IoT device which are now-a-days being installed in domestic and industrial premises in order to monitor the power consumption along with various other services like billing, load forecasting, dynamic pricing etc. Usually these IoT devices sample at high rate and report the whole data to the cloud. In this process large amount of energy is consumed at the IoT node for transmission of the data, large bandwidth is used for transmission over the Internet, and also storage requirement to save this enormous data is very high. Our studies reveal that all of this data is not of use, i.e., considerable portion of it is redundant. Hence, in the interest of energy sustainability of the wireless IoT nodes, efficient channel bandwidth usage, as well as cost-efficient cloud storage, compression or pruning of IoT data at the end node by providing some intelligence is of great interest.

To this end, the IoT devices measuring/reporting single parameter or multiple parameters need to be judiciously pruned (leading to single-variate pruning/compression or multivariate pruning/compression) keeping in mind the time-(non)criticality of the data type, without compromising on the information content.

The nature of data usage points to the appropriate adoption or devise of data-driven learning techniques at the IoT(end) node. For example, if the nature of data is very time-critical, the learning and pruning/compression technique has to be purely online, whereas if the data is of somewhat non-real-time in nature, a time-staggered pruning/compression technique may be more appropriate. Likewise, if the data is a single-variable, the exploration on data reduction has to be fetched from that time-series itself, whereas in a multivariate IoT data collection scenario (which is more

prevalent in real-life IoT applications) additional intelligence can be built from the spatio-temporal information correlation. The associated data communication pipeline as well as the cloud intelligence can facilitate efficient transmission of the pruned content and extraction of source information at the cloud for its usage from any remote location in the network. This work aims to develop a series of cost-effective Smart Energy Meter prototypes which incorporate the intelligence of judiciously pruning the sensed data and converting it into smart data before transmission. Another aim of this work is to demonstrate the reconstruction accuracy of the proposed architecture in real-life scenarios.

Acronyms

SG	Smart Grid
AMI	Advanced Metering Infrastructure
IoT	Internet of Things
AI	Artificial Intelligence
sMAP	Simple Measurement and Actuation Profile
RPi	Raspberry Pi
OS	Operating System
USB	Universal Serial Bus
SBC	Single Board Computer
TTY	Teletype Writer
JSON	Java Script Object Notation
NTP	Network Time Protocol
DB	Database
PHP	Hypertext Preprocessor
QoS	Quality of Service
MMIoT	Massive Machine Type IoT
API	Application Programming Interface
GPIO	General Purpose Input Output
M2M	Machine to Machine
LwM2M	Lightweight M2M
CoAP	Constrained Application Protocol
ToT	Transfer of Technology

Contents

Acknowledgement i
Abstract ii
Acronyms iv
Contents v
List of Figures viii
List of Tables ix

1 Introduction **1**
1.1 Internet of Things 1
1.2 Motivation 3
1.3 Objective 5
1.3.1 Project Plan 7
1.4 Outline of the thesis 7

2 Background **8**
2.1 Advanced Metering Infrastructure 8
2.2 Data-driven Framework in IoT 11
2.3 Conclusion 12

3 Data Driven Resource Optimization Approaches **14**
3.1 Non-real time univariate approach 15
3.2 Non-real time multivariate approach 17
3.2.1 Principal Component Analysis 17
3.2.2 Compressive Sensing 18

4 Methodology, Approach and Tools **20**

4.1 Smart Energy Meter 20

4.1.1 Overview of the Hardware Components 20

4.1.2 Budget Estimation 31

4.1.3 Hardware Architecture 34

4.1.4 Software Architecture 40

4.2 IoT Application Server 52

4.2.1 Prototype - 1 52

4.2.2 Prototype - 2, 3 and 4 56

5 Conclusions and Future Scope **62**

5.1 Conclusion 62

5.2 Future Scope 63

Bibliography **65**

List of Figures

1.1 Footprints of IoT applications 2
1.2 Conventional Energy Meter versus Smart Energy Meter 4

2.1 Overview of AMI Architecture 10
2.2 An example of IoT network 12

3.1 Flow Diagram of Univariate Approach 17
3.2 Flow Diagram of Multivariate Approach 18

4.1 RS 485 Single Point Connection (Source : Enersol Official Website) . . 21
4.2 RS 485 - USB Converter 22
4.3 RS 485 - RS 232 Converter 22
4.4 Modbus Slave Device : Memory Registers 24
4.5 Modbus Message : Function Codes 24
4.6 Modbus Message : READ HOLDING REGISTERS 25
4.7 Modbus Message : Slave Response 25
4.8 WiFi Module : ESP8266-01 26
4.9 FTDI FT232 Interfacing with ESP8266-01 27
4.10 NB-IoT HAT for RPi : SIMCOM7020E 28
4.11 NB-IoT Module : BC 66 28
4.12 Raspberry Pi Development Board 29
4.13 STM32L476RG Nucleo-64 Development Board 30
4.14 Hardware Components of Prototype - 1, 2 and 4 34
4.15 Single Phase Connection Diagram 35

4.16 Interfacing Diagram of Prototype - 1, 2 and 4 36
4.17 Hardware Components of Prototype - 3 37
4.18 Interfacing Diagram of Prototype - 3 38
4.19 Smart Energy Meter using sMAP framework 40
4.20 sMAP Source Components 42
4.21 Smart Energy Meter with Data Pruning Subsystem 45
4.22 Smart Energy Meter with Data Pruning Subsystem : Task view 46
4.23 Smart Energy Meter with Multivariate Data Pruning Subsystem 50
4.24 Smart Energy Meter Prototypes 51
4.25 sMAP Archiver 52
4.26 Health Parameter : Plot of CPU Temperature Time Series 54
4.27 Energy Consumption Parameter : Plot of Voltage Time Series 55
4.28 Plot of Original Time Series 58
4.29 Plot of Reconstructed Time Series 59
4.30 Plot of Original and Reconstructed Time Series 60
4.31 Smart Energy Meter Lab Setup 61

List of Tables

4.1 Address Mapping for Enersol MFR 2810. 21
4.2 Cost of the Components. 31
4.3 Hardware Cost of Prototype 1, 2 and 4. 32
4.4 Hardware Cost of Prototype 3. 33

Chapter 1

Introduction

1.1 Internet of Things

Internet of Things has attained tremendous significance due to the recent advancements in embedded systems, electromechanical systems, and networking capabilities. IoT is likely to impact every aspect of human activity by automation, control, and networked monitoring. However, the mass deployment of IoT devices poses certain challenges in storage and communication requirements, security, and energy sustainability. Also, the nature of IoT traffic is dependent on the type of application. Each application has its Quality of Service (QoS) requirements.

IoT framework is an amalgamation of sensors, networking, big data, and AI to achieve multiple benefits. Paper [1] mentions some of the key application areas of this technology :

- Regulation of energy consumption
- Streamlining operations in factories
- eHealthcare services
- eEducation

- Agriculture and Transportation
- Sensor networks driven process monitoring etc

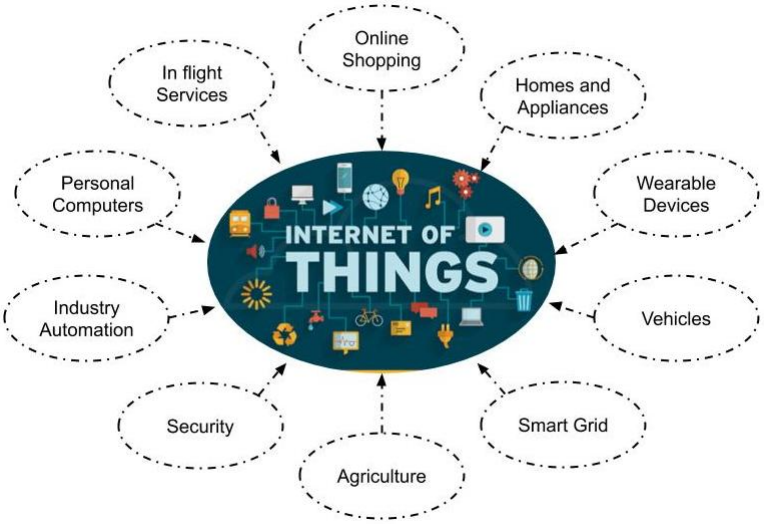

Figure 1.1: Footprints of IoT applications

In order to support the massive deployment of IoT devices, studies on security features, reliable and efficient network architectures, access, and network protocols have gained considerable attention. In this work, we focus on the implementation aspect of *Smart IoT Communication* which is less explored and relatively new.

A smart meter is the next generation of the electricity meter. It includes various sensors for identifying the parameters and sends the data to the control center. It may also control the operation of the devices in the customer premises through command signals. On a regular basis, the utility provider receives the electricity consumption data from the customers' various devices. It helps in managing the electricity demand-response in an efficient manner. It also provides useful information to the consumers about their energy usage patterns. The integration of smart meters in the smart grid will help utility companies to identify unauthorized consumption and electricity thefts, thereby improving the distribution efficiency and quality of power. Irrespective of the type or quantity of the parameter to be measured, Smart Meters should have the following six basic functionalities [2]:

- **Display :** In-home display of real-time energy consumption which aids the consumer in managing their demand efficiently.
- **Synchronization :** Synchronization of time for reliable billing and data analysis.
- **Power Management :** Auxiliary source of power to keep the device working to handle the loss of primary source of energy.
- **Secure Communication :** Sending energy consumption information and receiving control commands must be through a secure channel as well as upgrades for the trustworthiness of firmware.
- **Control and calibration :** It must be able to compensate for the variations across each type of system.
- **Quantitative measurement :** Accurate measurement of the physical quantity using various established physical principles and approaches.

1.2 Motivation

With the explosive growth of smart devices in the age of the Internet of Things and the advancements in electric metering in Smart grid has resulted in the generation of large quantities of data. **Advanced Metering Infrastructure(AMI)** has emerged as a powerful means for two-way communication between the end consumers and the utility service providers. Smart meters act as sensing devices in IoT. These devices are deployed in industrial and domestic environments and regularly sample different parameters of electricity usage. Hence, smart meters generate vast amounts of fine-grained electricity consumption data. The transmission of these massive chunks of data certainly creates issues of storage and bandwidth requirements in the context of IoT. This data is used by various applications such as demand-side management, billing, dynamic pricing, load forecasting, etc. The number of IoT devices is expected to rise in the near future when the massive deployment of IoT takes place. Hence, compression of smart meter data has recently captured the interest of the scientific community.

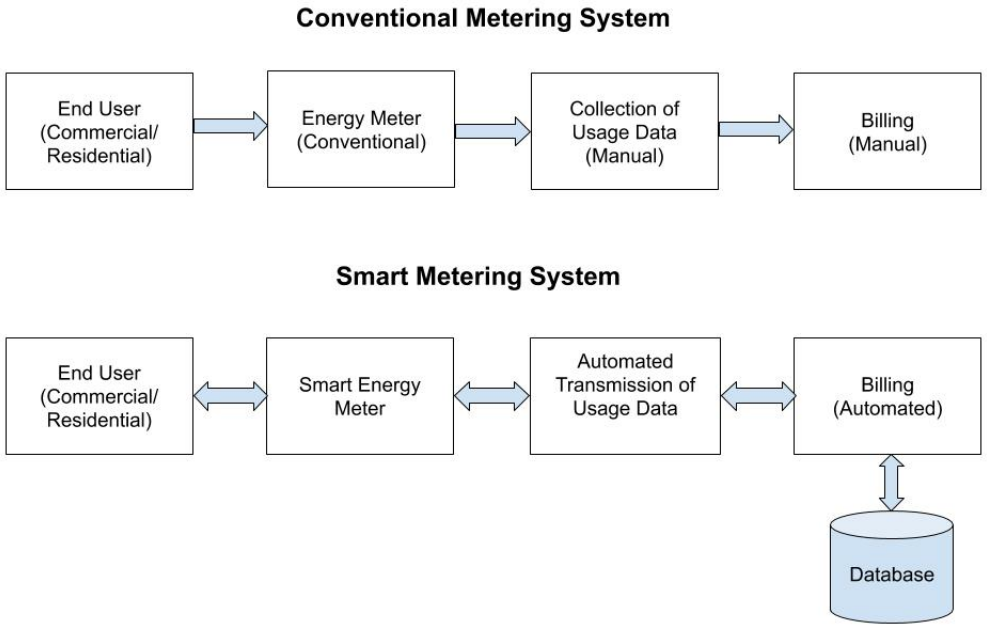

Figure 1.2: Conventional Energy Meter versus Smart Energy Meter

A smart meter is an advanced device that supports bi-directional communication compared to a conventional energy meter. Smart meters read electricity consumption parameters like current, voltage, frequency, power, etc. and communicate it to the control center in a secure manner. Figure 1.2 shows the architectural difference between a conventional energy meter and smart energy meter. Generally, Smart Meters are equipped with two major modules, which are sensing and communication. Hence, each smart meter can be broadly broken down into two subsystems as Metrology and Communication, respectively. The metrology depends on multiple parameters such as technical requirements, the accuracy of data required, application, measured phenomenon, etc. The communication subsystem includes data encryption and security. Data collected by the Smart Meter is a combination of sensed parameters, the current timestamp, along with a unique device identifier.

The huge amount of data generated by the Smart Metering framework helps in

understanding and modeling the patterns of energy usage. However, resource-efficient transmission and storage of this vast data remain a challenging task. Due to limitations in the handling of this vast amount of data, methodologies for smart meter data reduction need to be exploited. However, Data-Driven Resource Optimization techniques have not been exploited in Smart Meters. At each smart meter, the communication bandwidth required for transmitting smart meter data can be considerably reduced by making use of the sparse nature of data, with minimum loss of information. Therefore, it is suggested to integrate a Data Pruning Subsystem in Smart Meters, which will enable to refine data from the IoT devices and turn bulk data into smart data.

1.3 Objective

The focal objective of this work is to develop a *Cost Effective End-to-End Smart Energy Metering Solution with Data Pruning Subsystem at the Edge Node*. The Data Pruning Subsystem employs data driven resource optimization techniques depending on the level of compression required, and also the amount of data available to operate on. As a use case towards commercialization, smart energy meter has been chosen to demonstrate the concept. Smart energy meter has been chosen since it makes available multiple parameters that can be used to achieve high levels of compression. This concept can be easily extended to similar IoT devices generating time-series data consisting of single or multiple parameters. In order to make the prototypes commercially viable, the following challenges need to be mitigated.

- **Cost Minimization** : The choice of the hardware components needs to be judiciously done so as to have a reasonable performance at a minimal cost.
- **Minimal change to existent infrastructure** so that the proposed system can be easily integrated, and it has wider acceptability in the market. Also, the existing devices can be upgraded at a reasonable cost, wherever possible.
- **Implementation Complexity** of the data pruning subsystem. It can be further sub divided into :

- **Space complexity** of the algorithm since the system needs to work in near real-time, extra memory will be required to store the intermediate values generated during the operation of the algorithm. So, optimized implementation of the algorithm will play a crucial role in reducing the cost as it will have a small memory footprint. Also, in order to meet the configured sampling rate accuracy, the system will need to buffer the measured parameters as well. Since it cannot be guaranteed that the algorithm will finish execution and also transmission will take place within one sampling period.
- **Time complexity** will be a lesser challenge. The introduction of the new subsystem will introduce some delay, the magnitude of which depends on the processing power of computing resources.

- **End-to-End Solution for commercial deployment :** The system should support popular communication technologies. It should also be up-gradable to support technologies to be deployed in the near future. This will tend to increase the cost of the product. This should be made flexible in order to cater to different markets.

- **Hardware Assembly :** Identification of appropriate hardware components and standalone testing. Interfacing them to build the final product. Off the shelf, components to be preferred for prototyping in order to reduce the product development time.

- **Intermittent Network Connectivity** needs to be handled. It needs to be ensured that measurements acquired and compressed are not lost due to the network outage. These values should be locally stored and sent to the central server whenever the network connectivity is restored. The amount of local required will dictate the choice of the hardware. This will impact the cost of the product.

1.3.1 Project Plan

The project was planned to be executed in stages. Each stage consisting of the following phases :

- Building Proof of Concept prototype using RPi.
- Commercially viable microcontroller-based implementation.

Stage - 1 was aimed at developing a conventional smart energy meter using an open-source python framework. It helped in gaining an insight into the architecture and actual working of IoT devices, thereby helping in contemplating the feasibility of incorporating any kind of intelligence at the edge device, which will be a part of Stage - 2.

Stage - 2 Based on the evaluation carried out in Stage - 1, understand the requirements, and propose a software architecture for incorporating a univariate data compression algorithm in smart energy meters. Develop a software framework/library which will serve as a basis to build more intelligent IoT devices.

Stage - 3 Having gained some insight and expertise in the development of Smart Energy Meters with a data compression subsystem and a working version of the software framework for edge intelligent devices, incorporation of multivariate data compression algorithm in the above devices was attempted.

1.4 Outline of the thesis

The remaining thesis is arranged as follows.

Chapter 2 discusses the Advanced Metering Infrastructure in Smart Grid and data-driven resource optimization approaches in IoT devices.

Chapter 3 provides an overview of the data pruning algorithms.

Chapter 4 contains the crux of this thesis. It provides a detailed discussion of the hardware and software implementation of the prototypes developed.

Chapter 5 provides a conclusion to the thesis and discusses the future scope of the project.

Chapter 2

Background

2.1 Advanced Metering Infrastructure

An electric power grid [3] is an interconnection of power generation sources, electric transmission network, transformers, and distribution network to deliver power to the consumers. Electrical energy that is generated by power plants and transported over very long distances using the transmission network and distribution network to reach the end consumers is referred to as an electric grid. The operation of electric grid has become obsolete. Due to the lack of automated analysis, it is unsuitable to the fast-growing demand for electricity. Hence, the concept of **Smart Grid (SG)** has gained prominence merged which is touted as the next generation of electric power system.

SG can be considered as interconnecting the various components of electric grids using communication networks. It aims to provide electricity supply to the end-consumers in a safe, reliable, efficient, and secure way. It also enables the unification of various renewable as well as alternative sources of energy through communication networks and automated control. In SG, two-way communication is established between the various components and allows the end-user to not only consume power but also to supply excess power to the SG. This is achieved with the the help of **smart meters** that monitor and measure the bi-directional flows. Since the various components are

interconnected via communication links in SG, it provides an infrastructure where multiple sources can produce electric energy and dynamically load balance depending on the sensing information exchanged between the components.

Smart Meter is the key to realising Smart Grid. **Advanced Metering Infrastructure(AMI)** is a collective term to describe the whole infrastructure that collects and analyzes in near real-time data from smart meters and control center equipment using two-way communication. This data is used to manage various applications and services intelligently. AMI can be considered as the first step towards the digitization of the electric grid. AMI is the backbone of the smart grid and has gained great attraction due to accuracy in online meter reading. It includes smart meters, e.g. gas, electric meters, communication network between utility providers and customers, and data management systems for managing and analyzing the data for further processing. A smart meter is capable of measuring power consumption in detail as compared to a conventional meter. It periodically sends the collected information to the utility provider for billing, monitoring of load, and demand-response mechanisms by the control center. It also enables consumers to monitor and control their power consumption. Hence, through customer participation, it may be possible for the utility provider to provide electricity at dynamic and relatively low prices. The objectives of AMI can be:

- Remotely available meter reading with minimal errors
- Identification of problem in the network.
- Load profiling
- Energy Audit

Figure 2.1 shows the various components of the AMI architecture [3]. AMI comprises all hardware and software components that participate in the measurement of energy consumption and the transmission of information to utility providers and customers. All-encompassing technological components of AMI include :

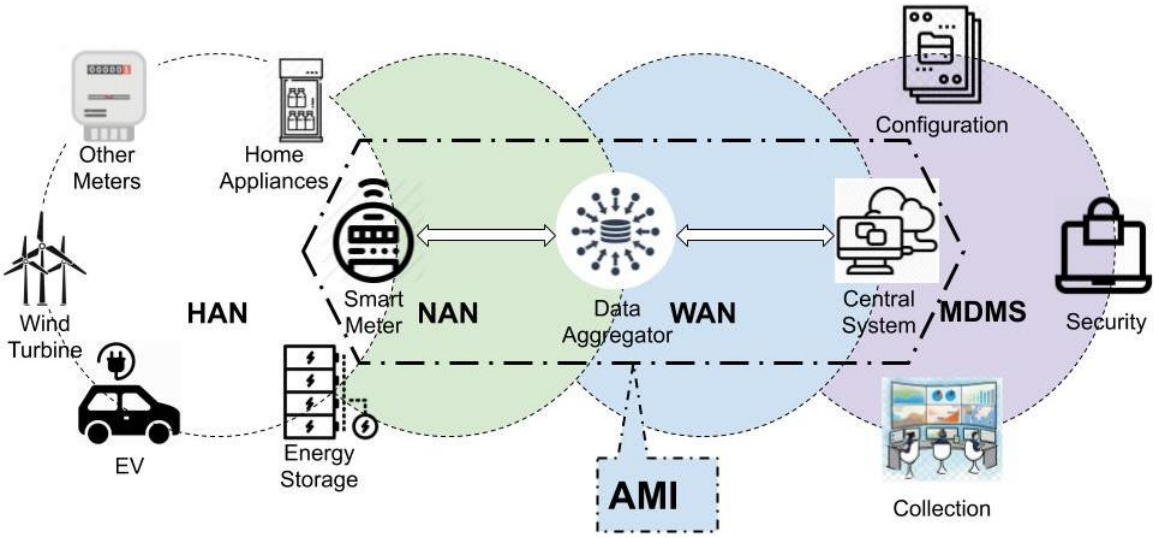

Figure 2.1: Overview of AMI Architecture

- **Smart Meters** - These are advanced metering devices that are capable of sampling information about water, electrical energy, and gas usage at regular intervals and conveying the data to a central entity through a dedicated communication network. It also receives information from the utility provider and transmits it to the consumer.

- **Communication Network** - It is an advanced communication network supporting bi-directional communication between smart meters and the utility companies. It enables information exchange between them. Networks such as Power Line Communications, Fiber Optic Communication, Wireless Communication are used for this purpose.

- **Meter Data Acquisition System** - These are software applications running on the hardware at the Control Center, and the **Data Concentrator Units (DCUs)**. These are used to receive data from the smart meters via a communication network and forward it to the Meter Data Management System (MDMS).

- **Meter Data Management System (MDMS)** - The Head-end system or the cloud service which receives, stores, and analyzes the metering information.

2.2 Data-driven Framework in IoT

In the context of 5G, IoT is a platform for collecting and transmitting data collected by the sensors. Data analysis provides vital information about the dynamic nature of real-world devices. It enables the development of automated processes intending to improve the efficiency of the system. However, due to the generation of fine-grained data within a short period, a vast amount of data is sent in the communication network. This is referred to as **Big Data**. Therefore, the major challenges posed by this rapidly evolving technology are :

- Efficient archival of data in the cloud
- Efficient data transmission

Smart IoT communication proposes a data-driven approach. In this framework, each IoT device has been imparted some intelligence in order to fulfill a certain objective. One approach of doing so is to impart the capability to the edge IoT device to understand the dynamics of the underlying process with the aim of converting bulk data into smart data. This conversion does not compromise on the information content essential for decision making at the control center. Imparting data pruning intelligence at the edge node level has two benefits :

- Reduced traffic in the communication network. This assumes significance in MMIoT(Massive Machine Type IoT) communication whereby a large number of IoT devices communicate information simultaneously.
- Reduces the data storage requirements at the cloud. It also aids in faster fetching and caching of content used by various applications.

Paper [4] mentions the improvement in the energy sustainability of the edge IoT node by incorporating data pruning intelligence. Paper [5] suggests that by learning the channel variability and using a dynamic channel coding scheme, the transmission reliability of IoT data can be improved. Paper [6] suggests a data-driven approach for reducing the bandwidth required for transmission of data generated by the phasor measurement

unit. The current-day availability of powerful and energy-efficient computing resources has made it feasible to carry out computation-intensive tasks at a meager cost. Also, the computation is carried out at each edge device in a distributed mode, thereby reducing the bottleneck in simultaneous computation.

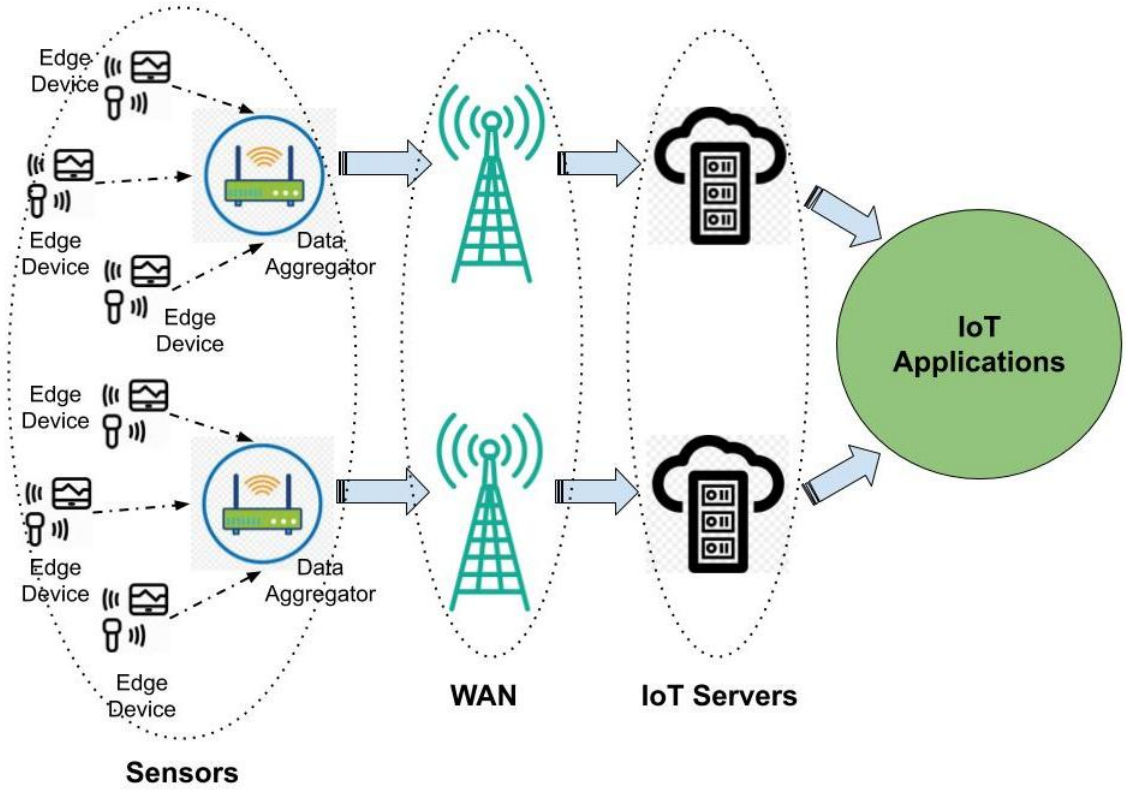

Figure 2.2: An example of IoT network

2.3 Conclusion

In conclusion, data-driven resource optimization techniques can be employed in commercial smart meter in order to make them intelligent in the context of data bandwidth saving. Currently, deployed commercial IoT devices lack the intelligence to exploit data reduction techniques at the edge node and transmit all the sensed data, thereby sending redundant information content. These IoT devices can be considered as conventional sensing devices with network connectivity. With the advent of 5G, there is expected to

be a very dense deployment of edge devices, and the data sampling rates are going to increase. Hence, the need arises to make these devices smarter in some context. This work makes an attempt to make these devices smarter from the context of data pruning at the edge node. Therefore an attempt is made to incorporate the data bandwidth saving strategies in real smart energy meters and develop a series of prototypes in this direction.

Chapter 3

Data Driven Resource Optimization Approaches

Commercial smart meters sense various energy consumption parameters such as current, voltage, apparent power, frequency, energy, etc. They may also measure meter health-related parameters. Unlike conventional analog meters, these automated meters follow a high-speed data sampling approach, thereby generating a huge chunks of data. The data consists of multiple sensed variables. This data is forwarded to the nearest data aggregator using a wired or wireless communication network. Fine-grained data is useful for near real-time applications such as dynamic pricing, short term load forecasting, demand response, energy feedback.

Because of the limitations posed by massive chunks of data **(Big Data)**, techniques for converting smart meter data into smart data have attained significance. Data Compression approaches employed in smart meters can be classified based on whether they operate at the edge device or the aggregator in the AMI. Techniques employed at the aggregation point have a high compression ratio since the aggregation point has access to vast amounts of data from many meters connected to it. The sampling rate is typically half-hour, and hence it aids in identifying various energy consumption patterns such as daily, weekly, seasonal, or behavioral. Recent studies have noted that

in order to support near real-time applications, a fine-grained smart meter is more useful. But it results in the generation of a considerable amount of data even in the first leg of IoT communication. Various compression techniques have been suggested in literature and can be studied under two heads :

- **Lossless Compression Techniques** : These techniques lead to an accurate reconstruction of data, and there is no information loss. Four loss-less data compression techniques have been(adaptive Markov chain Huffman, adaptive trimmed Huffman, Lempel Ziv Markov chain Huffman and tiny Lempel Ziv Markov chain) studied in [7], [8]. In [9], a differential coding based resumable compression method has been suggested.

- **Lossy Compression Techniques** : These techniques report a higher compression ratio and are suitable for error-tolerant scenarios. In a recent study [10], authors have proposed an adaptive data pruning algorithm based on compressive sampling for bandwidth saving with minimal loss of information. It exploits temporal data stochasticity of a single time series data at the smart meter. In paper [11], authors have proposed a data compression technique that makes use of multivariate data. It utilizes the cross-correlation among different variables acquired by the smart metering devices. Once decorrelated, sparsity in each generated stream is exploited to achieve temporal compression.

Our study reveals that a data pruning subsystem employing these algorithms has not been integrated into commercial smart meters.

3.1 Non-real time univariate approach

Paper [10] proposes a lossy data compression technique based on compressive sampling on a single time-series data. Compressive Sensing makes use of the sparsity in the given data or time-series to compress it and also provides fairly accurate reconstruction from fewer samples as compared to that required by Nyquist sampling theorem. Consider,

a data transmission window of n number of samples. The i^{th} sample is denoted by x_i. Hence, $\{x_1, x_2, \cdots, x_n\}$ are the samples comprising x. So x can be represented as:

$$x = \psi f \tag{3.1}$$

where ψ is a matrix of size $n \times n$ representing the sparse basis matrix and the column vector of coefficients corresponding to ψ are denoted by f. Out of n samples in a given batch of data, only m ($m \ll n$) samples are randomly chosen for transmission. In order to save transmission bandwidth over the communication channel, this data downsizing is performed. Using (3.1), the samples which are transmitted can be represented by,

$$y = \phi x = \phi \psi f \tag{3.2}$$

where ϕ is sensing matrix of size $m \times n$. In order to accurately reconstruct the signal using the m received samples, the problem of underdetermined system of linear equations needs to be solved. Subspace pursuit algorithm has been employed in this work to reconstruct the original signal at the cloud server. For signal reconstruction, Random Gaussian Matrix and Discrete Fourier Transform (DFT) are chosen as sensing matrix and sparse basis, respectively, such that restricted isometry and incoherence property are satisfied. In this scheme, the sparsity is computed for each batch of data by calculating the number of DFT coefficients comprising 99.99% energy of samples. This helps in capturing the erratic nature of smart meter data and also plays an important role in compressing the data by decreasing the number of samples which are transmitted without comprising the information content or the reconstruction accuracy. Hence, this scheme is adaptive as compared to the conventional technique of compressive sampling, where the sparsity is inferred and remains the same. Accordingly, $m = s \log n$ are randomly chosen and transmitted.

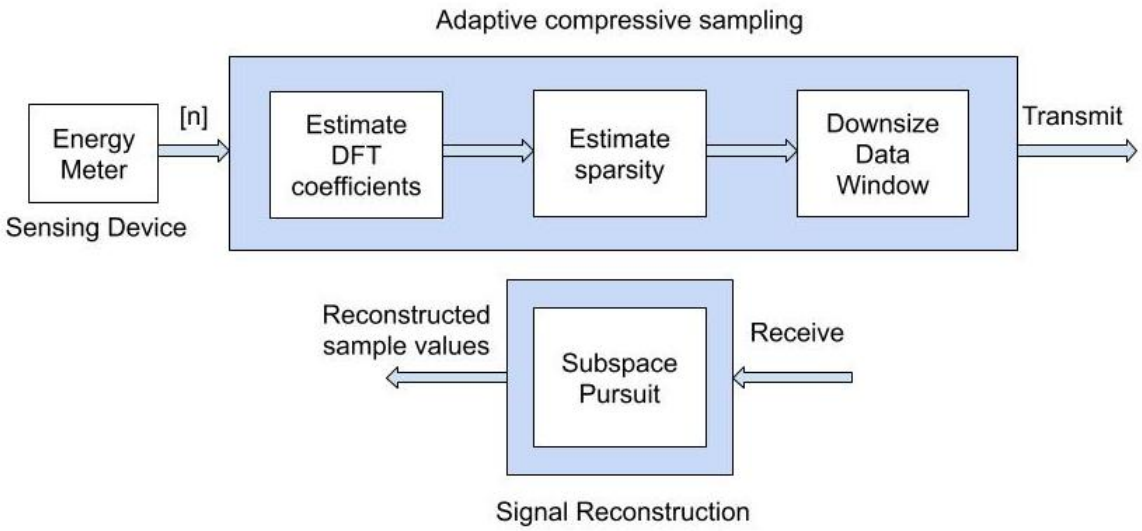

Figure 3.1: Flow Diagram of Univariate Approach

3.2 Non-real time multivariate approach

This technique utilizes the cross-correlation among different variables of the multivariate data sensed by a smart meter to decrease the dimensionality of the data. Once the dimensionality is reduced, for each stream/dimension, temporal compression is exploited. Since the data is fluctuating in real-world conditions, the vital parameters, namely temporal sparsity and a minimum number of required dimensions are computed for each batch. The algorithm operates in two steps as explained below.

3.2.1 Principal Component Analysis

PCA is a technique to reduce the dimensionality of the data by transforming it from n-dimensions to p-dimensions. With this operation, the strongly related features in the input and a major portion of the variance of the whole multivariate data is preserved in p-dimensions. From the input data matrix, eigen-vector eigen values are computed. In the transformed space, eigen-vector serves as an orthogonal basis. The basis vectors are dependant on the input data, unlike other transformation techniques. The principal

components obtained are arranged in decreasing order of variance and are uncorrelated.

3.2.2 Compressive Sensing

It is a technique that provides a compressed representation of data without much loss of information content. The signal is reconstructed by finding a solution to an under-determined linear system. The condensed representation provides saving in data storage and transmission. The properties of sparsity and incoherence are used in mathematical algorithms to reconstruct the signal from a few number of measurements. For reconstruction purpose, Subspace Pursuit Algorithm [12] has been used for data reconstruction since it is accurate and has low computational complexity.

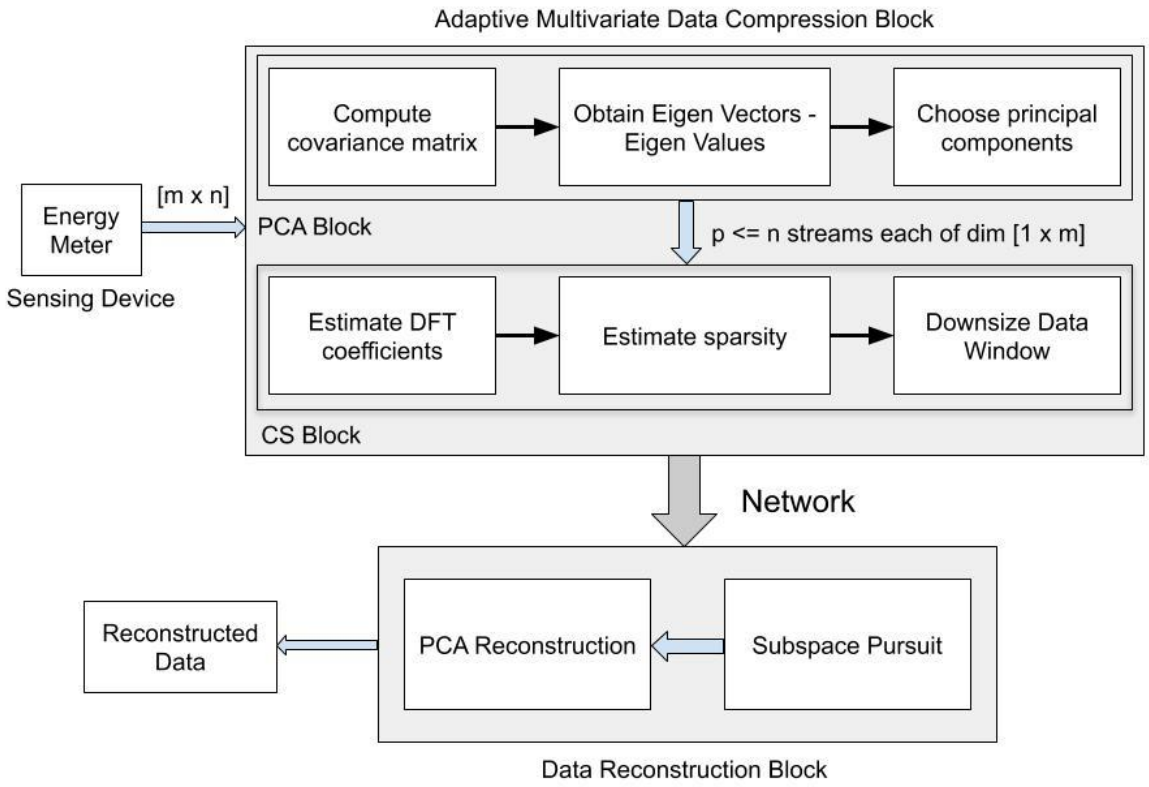

Figure 3.2: Flow Diagram of Multivariate Approach

Adaptive multivariate data compression(AMDC) [11] executes in two stages at the transmitter. It processes data in batches. It operates on 2-D data, where

one dimension represents the parameter type, and the other dimension represents the time scale. PCA is applied to decorrelate the input variables. Using the eigen value-eigen vector combination, the principal components are obtained. Since this operation preserves the variance in the data in a few dimensions, so only those principal components are considered for the purpose of recovering the data depending on a predefined threshold. Then each stream of principal component is sent for temporal compression where sparsity is determined by the number of DFT coefficients that contribute at least 99.99% energy of the samples. The adaptive part of the algorithm in each stage is the number of principal components, and the number of transmitted samples are determined for each batch.

Reconstruction at the receiver is also a two-stage process. The compressed version of the principal components is recovered using Subspace pursuit algorithm to yield the principal components. These are further fed to a PCA reconstruction block, which yields the actual data with reasonable accuracy.

Chapter 4

Methodology, Approach and Tools

4.1 Smart Energy Meter

4.1.1 Overview of the Hardware Components

This section covers the hardware components that were selected for building the prototypes. Cost-effectiveness was the primary factor in the selection of a particular hardware module. Off-the-shelf components were preferred in order to develop the prototypes quickly and in a time-bound manner.

Enersol MFR 2810 with RS485 Port

This power meter is cost-effective, easy to operate, and compact in size. It is capable of measuring basic parameters that are required to monitor an electrical installation. This meter provides a RS 485 Modbus RTU compliant port using which data from various registers can be read. A programming manual [13] is provided to configure the various parameters of the meter, including the Modbus slave configuration. The meter can operate in a single point connection or multi-dropped configuration using RS 485 twisted pair. It supports baud rate varying from 1200 to 9600 bps. It supports

two-wire RS 485 Half Duplex serial channel. Either a single/individual parameter or a block of parameters can be accessed through RS 485 communication port. Table 4.1 shows few entries of the address mapping provided by the manufacturer for the selected model of energy meter. The complete list is available at [14].

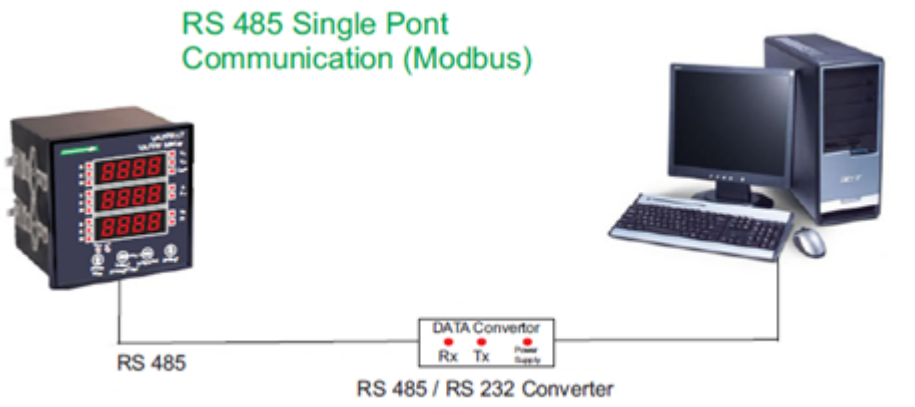

Figure 4.1: RS 485 Single Point Connection

(Source : Enersol Official Website)

Description	**Address**	**Datatype**
Apparent Power - avg	3840	32-bit Float
Active Power - avg	3842	32-bit Float
Reactive Power - avg	3844	32-bit Float
Average PF	3846	32-bit Float
Line to Line Avg Voltage	3848	32-bit Float
Line to Neutral Voltage	3850	32-bit Float

Table 4.1: Address Mapping for Enersol MFR 2810.

RS 485 Convertor - Modbus RTU Protocol

The **USB-RS485** is a hardware module for USB and RS 485 Protocol conversion. It is small in size and fully automatic plug-and-play USB module. It does not require external power to operate. It plugs into Personal Computer via the USB(Universal Serial Bus) port. It features automatic RS-485 flow control and does a robust USB to RS-485 protocol conversion.

Figure 4.2: RS 485 - USB Converter

RS 485-RS 232 is a hardware module to interface TTL and RS-485. It is a robust converter that provides bi-directional serial communications signal conversion between the TTL and RS-485. It supports multi-drop configuration and can connect 32 nodes in a network. This converter is suitable for office and industrial applications and provides superior characteristics/features. It has automatic directional, thereby making it easier to use as a replacement for TTL/RS-232 Serial Interface.

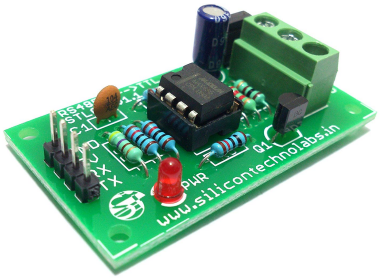

Figure 4.3: RS 485 - RS 232 Converter

Overview of Modbus Protocol

The Modbus communication protocol is the most common and popular protocol in the field of Supervisory Control and Data Acquisition for process automation. It is a communication protocol that was published by Modicon in the year 1979 to be used with its Programmable Logic Controllers. It provides a common language for the devices to exchange messages among themselves. It is an open standard that describes the specifications and messaging format to be used. The original Modbus interface ran on RS-232 serial communication. But most of the Modbus implementations developed later on, use RS-485 because it provides the following benefits :

- Longer distances.
- Higher speeds.
- Support for multi-drop configuration.

The Modbus communication over a two-wire RS 485 serial physical media makes use of the Master-Slave technique. In this technique, only the master device initiates the transactions or queries. The master device then waits a specified duration of time for the slave devices to respond. The slave devices provide the requested data to the master device or by performing the action as requested in the message. This protocol supports unicast as well as broadcast communication mechanisms. Slave devices return a response to all the messages, but they do not respond to messages destined to broadcast address. Slave devices only reply to queries from the master device and do not initiate messages on their own. The master's query consists of :

- Target Device Address (Slave/Broadcast Address).
- Function code specifying a read or write command.
- If it is a write command, then the data to be written.
- Field carrying error checking information.

The error checking field aids the master device in verifying the integrity of the message.

A slave device response consists of :

- Fields confirming that the request is received.
- The data which needs to be provided as requested.
- Field carrying error checking information.

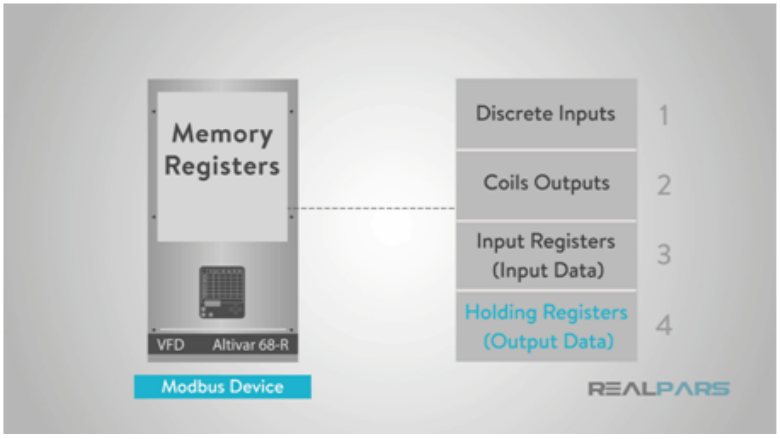

Figure 4.4: Modbus Slave Device : Memory Registers

Figure 4.4 shows the layout of the memory registers of a typical Modbus slave device demarcating where the configuration information, output, and input data can be written to and read from [15].

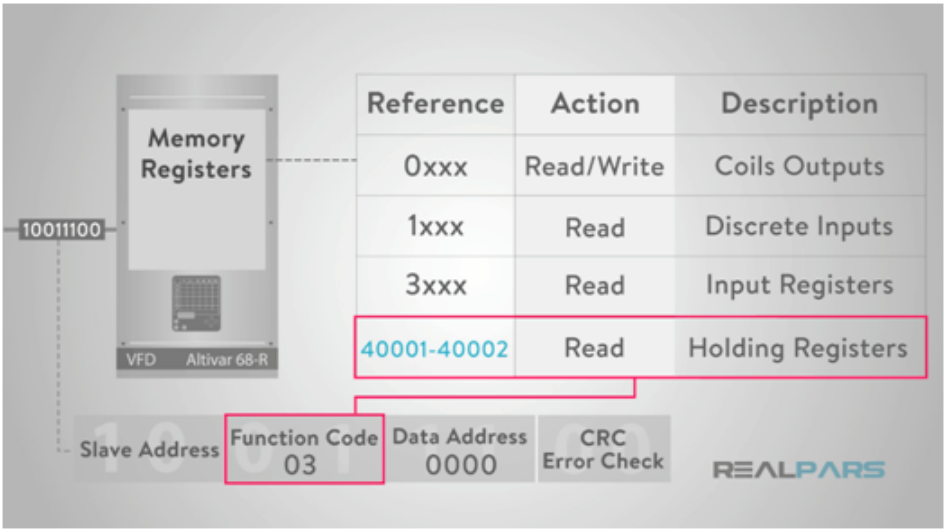

Figure 4.5: Modbus Message : Function Codes

Figure 4.5 shows the typically supported function codes by the Modbus slave devices and a short description of each function code [15].

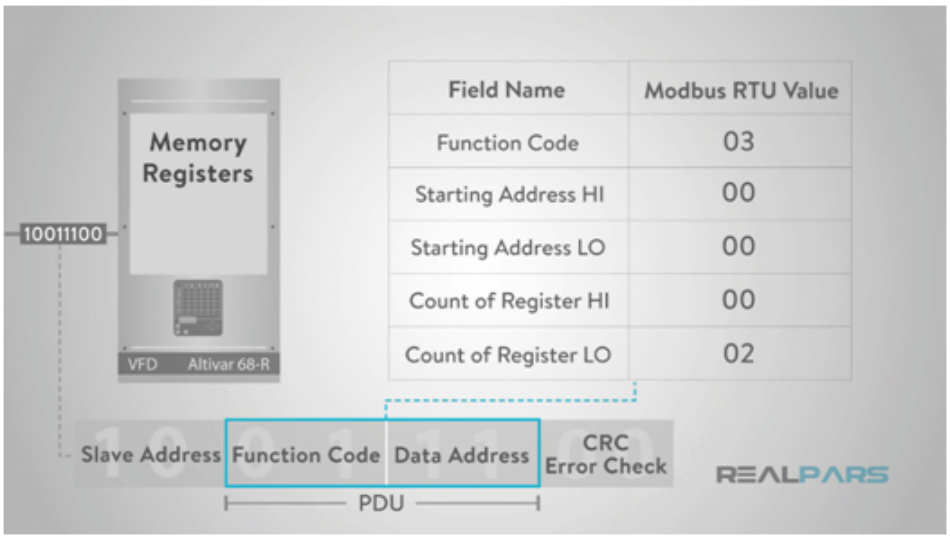

Figure 4.6: Modbus Message : READ HOLDING REGISTERS

Figure 4.6 shows an example of a request message to fetch the registers 40001 to 40002 in the Holding Register Area from slave device 1 [15] and Figure 4.7 shows the response message to the above query [15].

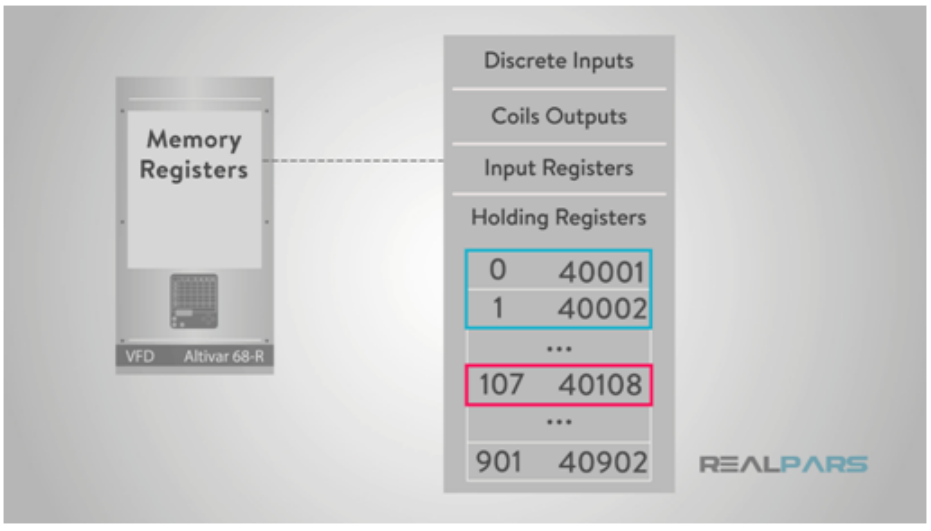

Figure 4.7: Modbus Message : Slave Response

ESP8266-01 WiFi Module

The ESP8266-01 is a small and cheap WiFi chip that has a microcontroller unit. It has I/O digital pins to control it and provides a simple and almost pseudo-code like programming language. It has full TCP/IP capability and supports up to 5 TCP/UDP simultaneous connections. This device is manufactured by Shanghai-based Chinese manufacturer, Espressif Systems.

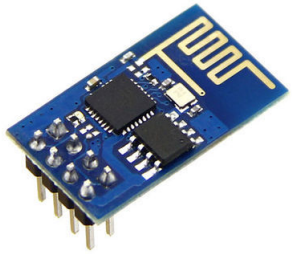

Figure 4.8: WiFi Module : ESP8266-01

The ESP-01 version of this device was first seen in the year 2014. This module has an MCU that is capable of connecting to a WiFi network and also establish TCP/IP connections. This module provides UART pins for communication but can also be accessed over the air. This module comes already programmed, and it is ready for use via AT commands without any initial configuration or any hardware setting. Few examples of basic commands :

- AT – response OK
- AT+CWLAP – list nearby available WiFi networks
- AT+GMR – check the firmware version
- AT+CIFSR – get current allocated IP address

Figure 4.9 shows the interfacing of FTDI FT232 and ESP8266-01 modules [16]. FTDI module can be used to test the AT commands provided by the ESP8266-01

modules. It provides to USB interface which can be connected to PC, and AT commands can be issued using the Serial Console. Hence, it helps in understanding the functioning of the WiFi module.

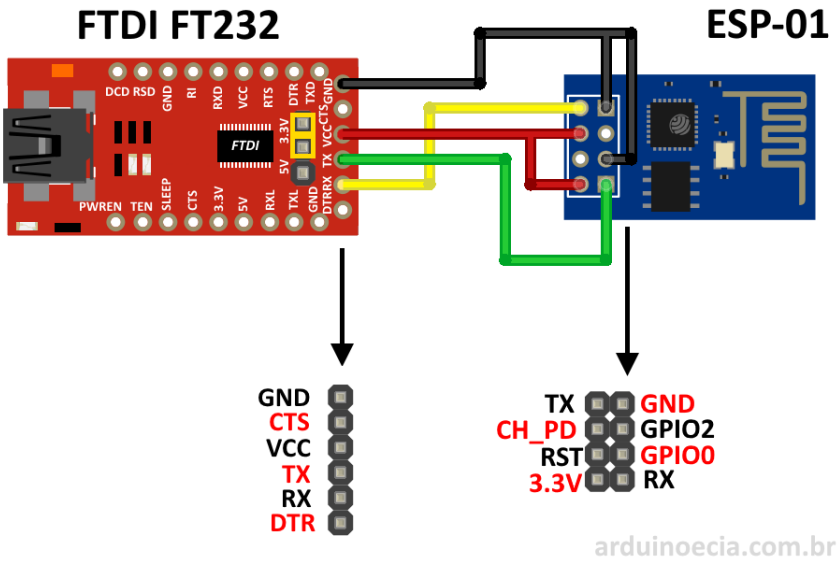

Figure 4.9: FTDI FT232 Interfacing with ESP8266-01

SIMCOM 7020E NB-IoT Module

The SIM7020 is a Multi-Band NB-IoT(NarrowBand-Internet of Things) module. This module provides a rich interface like UART, GPIO etc. for issuing AT commands. It also supports communication protocols like MQTT/LWM2M/COAP. It comes as a HAT for RPi [17]. It comes with end-user development resources to aid in application development. It can be easily integrated with the user hardware modules. Therefore, it reduces the investment of the customer and also shortens the time-to-market. TCP/IP of SIM7020 is a multiple client structure, supports up to 5 sockets (like TCP or UDP). SIM7020 also supports two types of HTTP communicating, HTTP GET, and HTTP POST.

Figure 4.10: NB-IoT HAT for RPi : SIMCOM7020E

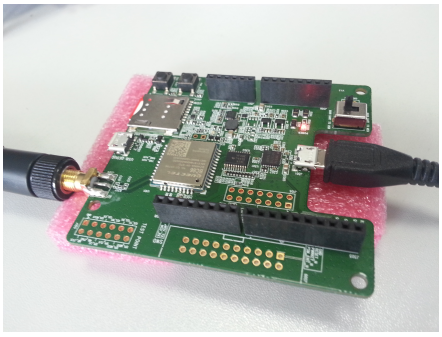

Figure 4.11: NB-IoT Module : BC 66

BC-66 NB-IoT module has been designed for applications that need low throughput data communication, low latency, and low power consumption in a variety of environments. It is suited for M2M applications. It is flexible, secure, and has reasonable performance. Few application areas are asset tracking, E-health, remote monitoring, metering, etc. It cannot support transparent and server mode.

Raspberry Pi Development Board

The Raspberry Pi is a cheap and small-sized single-board computer that runs a full-fledged OS. It provides connectivity to common peripheral devices like a monitor, a standard mouse and keyboard. This device was developed with the intent of enabling

everyone to explore computing and learn programming languages like Scratch, Python, C, etc. The Raspberry Pi is a very cheap computer that runs Raspbian OS, a variant of Linux OS. It also provides GPIO pins to control electronic devices for physical computing and develop IoT devices. It also provides UART pins for serial communication.

Figure 4.12: Raspberry Pi Development Board

STM32L476RG Nucleo-64 Development Board

The STM32L476RG [18] is a microcontroller based on the high-performance Arm® Cortex®-M4 32-bit RISC core. It is an ultra low power microcontroller operating at a frequency of up to 80 MHz. The core features a single precision Floating point unit (FPU), which supports all data types and single-precision data-processing instructions. It also features a memory protection unit (MPU), which enhances security of the application.It also implements a full set of DSP instructions. This development board also features common and advanced communication interfaces. It enables prototyping and trying out new concepts, affordably and flexibly. These boards are available in various combinations of power and performance. STM32 Nucleo-64 board comes with

a built-in ST-LINK debugger or programmer, and a separate probe is not required.

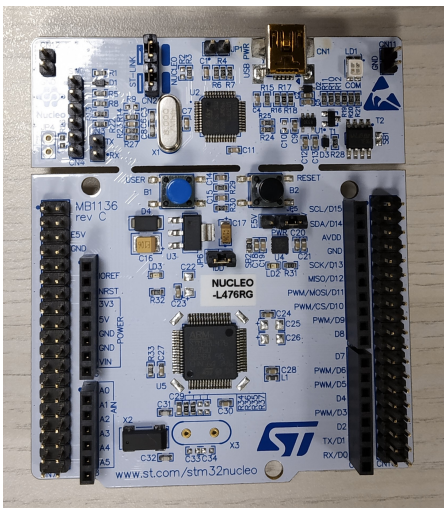

Figure 4.13: STM32L476RG Nucleo-64 Development Board

4.1.2 Budget Estimation

Budget is one of the main aims and motivations of the project, which is to reduce the cost of the series of prototypes developed. The super-set of the main components that have been utilized across the prototypes are shown in the table below, along with their cost.

Component	**Quantity**	**Price(in Rs/-)**
Enersol MFR 2810 3 Phase Meter with RS 485 port	1	3634
R-Pi Development Board	1	2898
PVC Meter Enclosure	1	500
RS 485 - RS 232 Bi-dir Converter	1	330
RS 485 - USB Converter	1	190
STM32 Nucleo-64 Board	1	1100
SIMCOM NB-IoT HAT for RPi	1	1900
BC-66 NB-IoT Module	1	1800
ESP8266-01 WiFi Module	1	250

Table 4.2: Cost of the Components.

Cost of Prototype 1, 2 and 4

Table 4.3 shows the list of components utilized in building the prototype and along with the cost of each component and the total cost. This cost is expected to reduce significantly when the product enters the mass production stage.

Component	**Quantity**	**Price(in Rs/-)**
Enersol MFR 2810 3 Phase Meter with RS 485 port	1	3634
R-Pi Development Board	1	2898
PVC Meter Enclosure	1	500
RS 485 - USB Converter	1	190
Total*		**7222**

Table 4.3: Hardware Cost of Prototype 1, 2 and 4.

* additional cost of SIMCOM NB-IoT HAT for RPi (Rs. 1900)

Cost of Prototype 3

Table 4.4 shows the list of components utilized in building the prototype and along with the cost of each component and the total cost.

Component	**Quantity**	**Price(in Rs/-)**
Enersol MFR 2810 3 Phase Meter with RS 485 port	1	3634
STM32 Nucleo-64 Board	1	1100
PVC Meter Enclosure	1	500
RS 485 - RS 232 Bi-dir Converter	1	330
ESP8266-01 WiFi Module	1	250
Total*		**5814**

Table 4.4: Hardware Cost of Prototype 3.

* additional cost of BC-66 NB-IoT Module (Rs. 1900)

4.1.3 Hardware Architecture

This section focuses on the hardware aspect of the prototypes, and it delves into the details of the interfacing of the various components.

Prototype - 1, 2 and 4

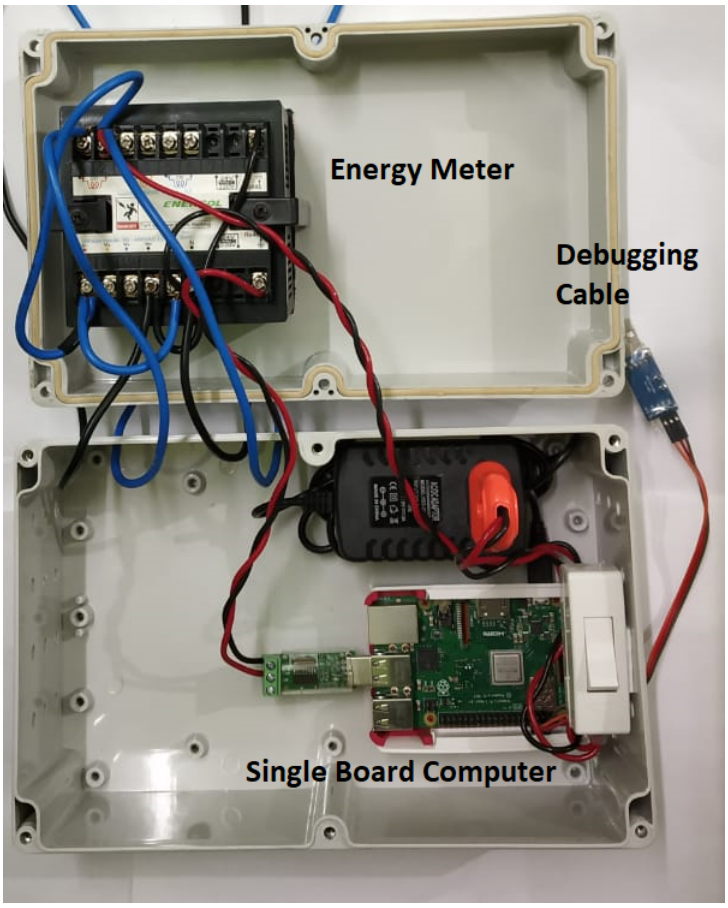

Figure 4.14: Hardware Components of Prototype - 1, 2 and 4

To shorten the cycle of prototyping, off-the-shelf components were preferred. Prototypes 1, 2, and 4 share the same hardware components since all of them are RPi based. The RS-485 port of the energy meter is connected to the USB port of the RPi Development Board using an RS-485 to USB converter. This converter does the required protocol conversion. All the components are enclosed in a PVC enclosure for protec-

tion. The RPi draws the power from the input mains supply using a DC adapter. A debugging port is also provided, which can be accessed from outside the PVC box. This debugging port uses an RS-232 to USB converter to provide a debugging console to the maintenance team in case there is some fault in the meter. Few holes have been provided in the PVC enclosure for ventilation. A miniature DC cooling fan can also be placed inside the PVC enclosure in case the meter is installed in a hot environment. Figure 4.14 shows the actual image of the hardware interfacing of the components.

Since the meter is a 3-Phase meter but for testing purposes, it has been connected to a single-phase power supply. The connections have been made in accordance with the connection guide [19] supplied by the manufacturer. An incandescent/LED bulb has been used as a load. The meter is connected in a single-point connection for RS-485 Modbus communication.

Figure 4.15 shows the connections to be made for single phase power supply.

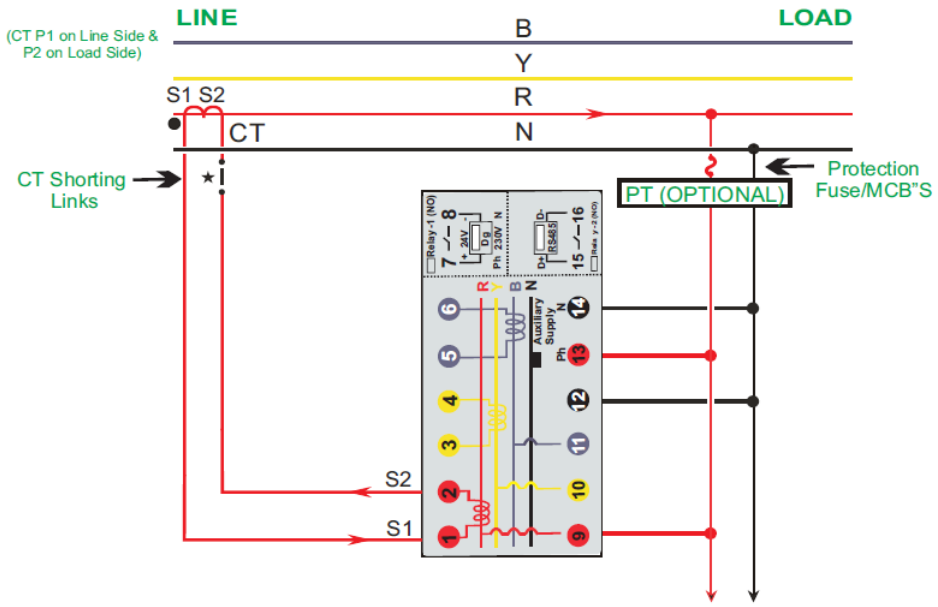

Figure 4.15: Single Phase Connection Diagram

Figure 4.16 shows the circuit diagram for connecting the various hardware components.

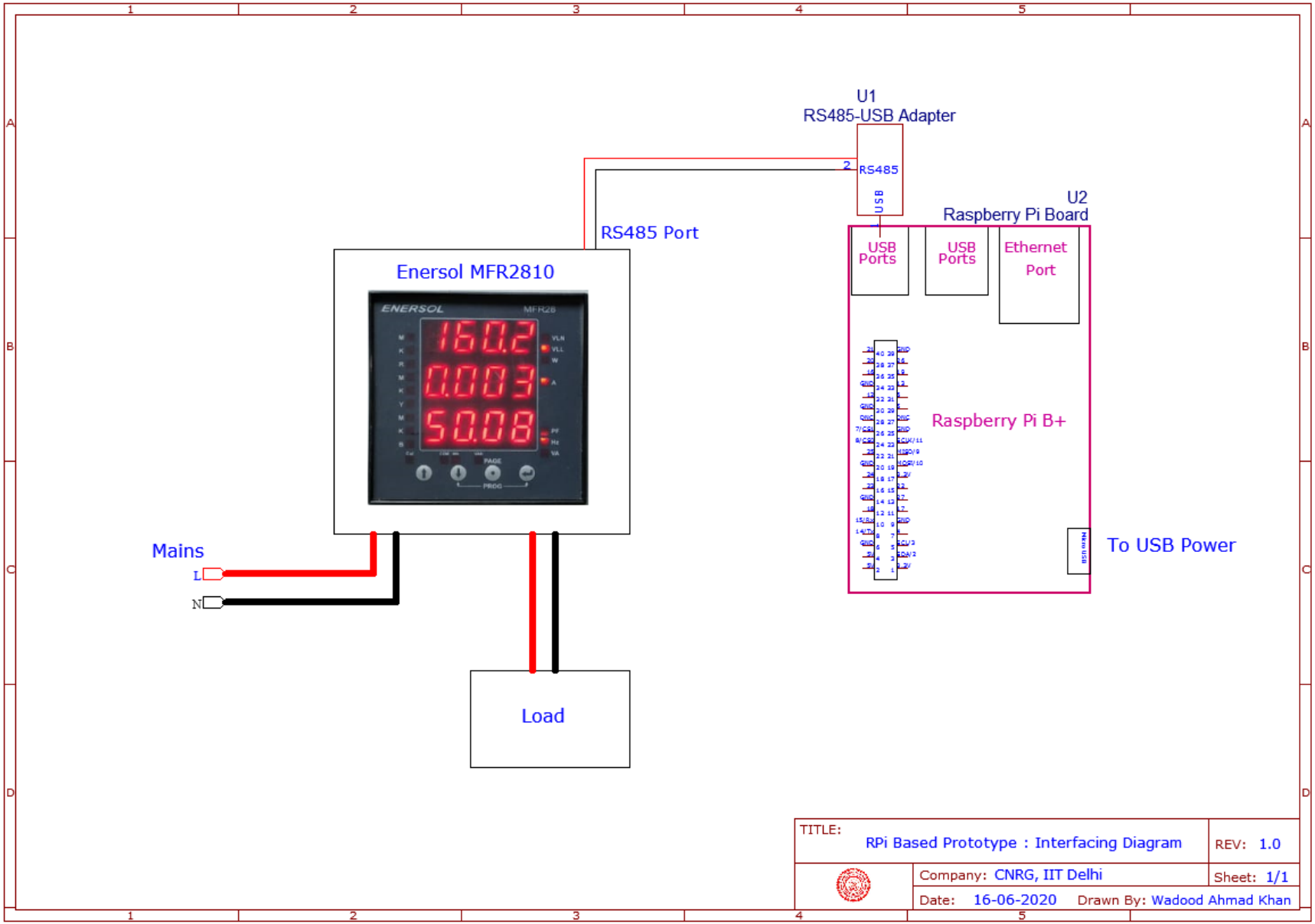

Figure 4.16: Interfacing Diagram of Prototype - 1, 2 and 4

Prototype - 3

Prototype - 3 is an STM32 microcontroller-based smart energy meter with a data pruning subsystem. In order to shorten the cycle of prototyping, the STM32 development board has been used. This microcontroller provides several UART interfaces for interfacing with various hardware modules. At least three UART interfaces are required to make a smart meter. One each to communicate with ESP8266-01 WiFi Module, RS 485 Modbus Communication with Energy Meter using RS485-RS232 converter and Debugging Console. This microcontroller was chosen since it is energy efficient and has sufficient memory to accommodate the data pruning algorithm. It also has an integrated debugger, which accelerates the development and testing of the application. One UART interface is reserved for this purpose in this development board. It can be connected to a PC using a USB Mini-B cable for flashing the executable file and accessing the debugging console. Figure 4.17 shows the actual hardware setup used.

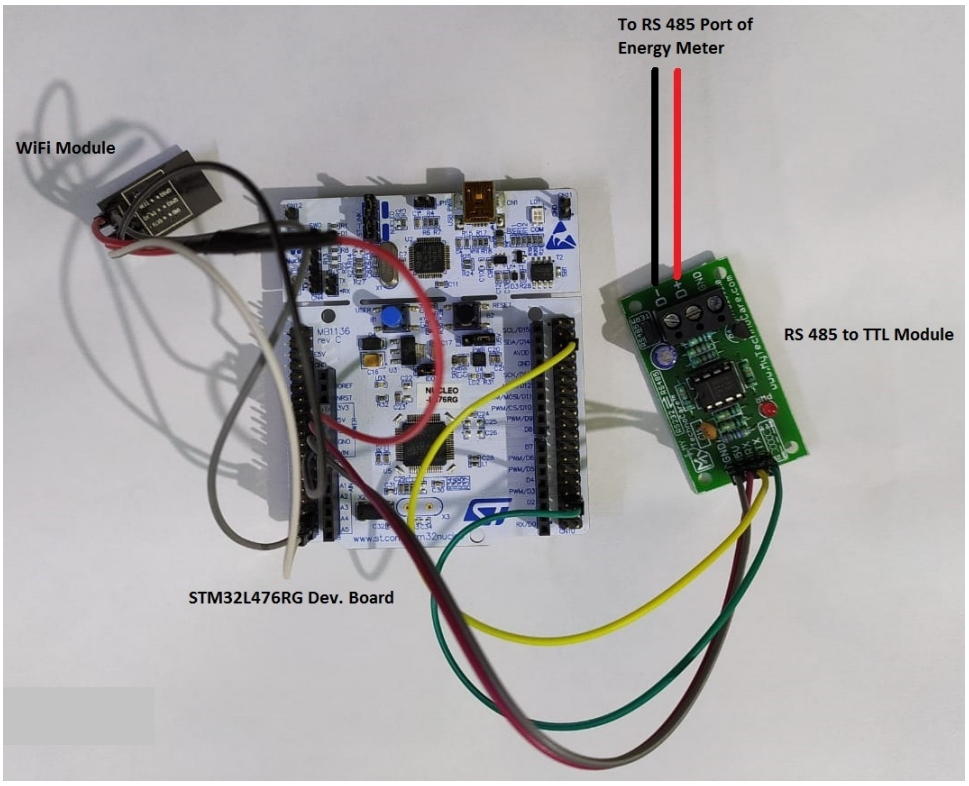

Figure 4.17: Hardware Components of Prototype - 3

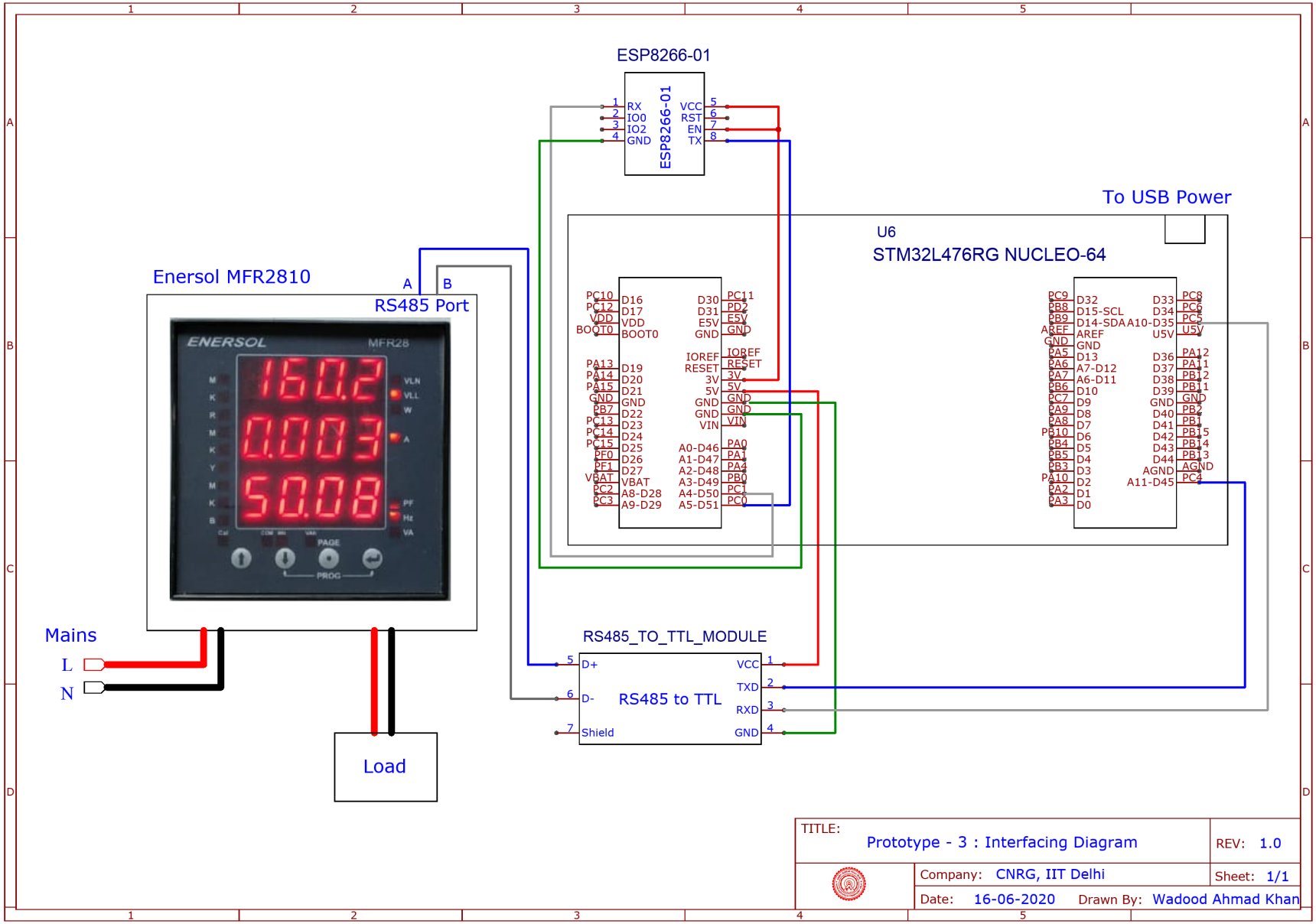

Figure 4.18: Interfacing Diagram of Prototype - 3

In this prototype, a RS 485-TTL bi-directional converter module has been used to communicate with the RS 485 port of the energy meter. This converter has automatic directional control, which relieves the programmer from controlling the direction from the microcontroller GPIO pin. The energy meter and the microcontroller communicate in a master-slave configuration. One UART interface is connected to the RS 485 converter, and another one is connected to the WiFi module using jumper wires. The board is powered by the USB Mini-B cable connected to a PC. Figure 4.18 shows the schematic of the various components and their interfacing. The pin diagram [20] for the board can be used to identify the UART interfaces to be used for connecting the various modules. Serial Interface 3 has been connected to the energy meter RS485 port using the RS-485 TTL converter. Serial Interface 1 has been connected to the ESP8266-01 WiFi Module.

4.1.4 Software Architecture

Overview of NTP Protocol

Network Time Protocol is a protocol which is used for synchronizing the clock of a computer client to a time source or a server in the internet. A time request is initiated by a NTP client and sent to the server. After a certain number of transactions with the server and using the time delay and its local offset, the client is able to synchronize it's clock with the server. The client sends a User Datagram Packet on Port 123 to the server. The server replies with the timestamp, which the client uses to determine the current value of time. NTP uses Coordinated Universal Time(UTC) to accurately synchronize the time across the various clients. The accuracy is dependent on the size of the network.

Prototype - 1

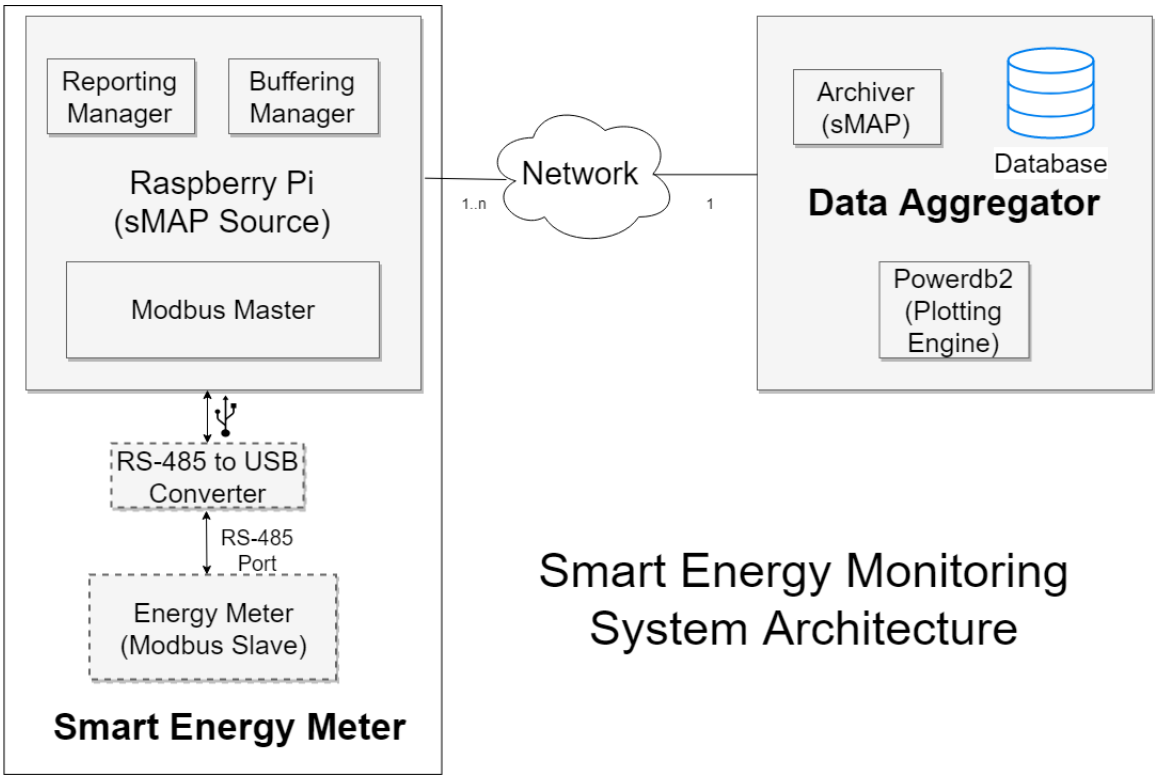

Figure 4.19: Smart Energy Meter using sMAP framework

Overview of sMAP :

The Simple Measurement and Actuation Profile (**sMAP**) [21] is a specification for a protocol which enables to expose and publish time-series data in a quick and easy manner. It supports data from a large variety of sensing equipment. The sMAP architecture comprises of several interconnected components that help in acquiring, storing, transmitting, and plotting time-series data : Sources, Archivers, and various Applications.

- **sMAP Sources** : This component establishes communication with the underlying existing instrumentation, and collects time-series data using a large variety of underlying protocols supported by sMAP library.
- **sMAP Archiver** is a component which is entrusted with the task of storing large volumes of data. It is a high-performance component which also provides an interface for accessing the stored data in a simple and powerful manner. It is also concerned with cleaning the raw data and locating the addresses of the data.
- **Applications** query the archiver to make use of real time or historical data. These applications may make use of this data to estimate an optimal control plan, providing feedback to the user and plotting the data in order to study the trend in the fetched data. Out of the box, powerdb2 is provided by sMAP project. It is a plotting engine and organizes time-series data.

Implementation

This prototype has been implemented by making use of an open-source framework sMAP. This framework requires the implementation of the sMAP source driver. Source driver connects to the existing instrumentation for data, and sMAP provides tools to expose the data over http. sMAP source library is designed to take care of common scenarios like :

- Intermittent Network Connectivity
- Appending metadata

- Remote Actuation

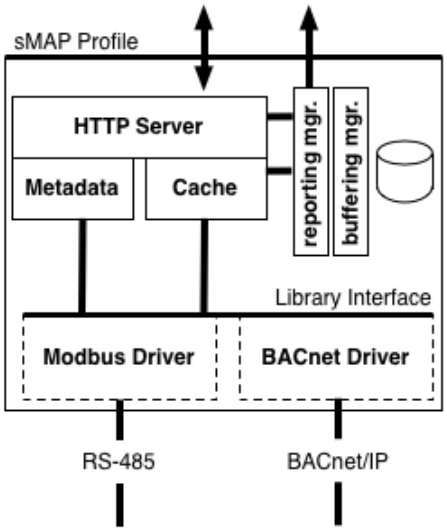

Figure 4.20: sMAP Source Components

The figure 4.20 shows the various components that make up the sMAP source [21]. These components aid in supporting the features mentioned above. In order to publish the energy consumption data, a configuration file needs to be written, which contains the run-time configuration of the edge device. This configuration file is passed to the sMAP daemon at startup. This configuration file is parsed by the sMAP source driver at startup. sMAP source driver communicates with the underlying energy meter using the RS 485 port and translates the data that is available into time series with some metadata. Then the daemon is started using the following command.

```
$ twistd --logfile abcd -n smap conf.ini
```

where "abcd" denotes the name of the logging file.

The source driver implements the **metrology subsystem**, which is the module primarily communicating with the energy meter using the Modbus protocol. Since an RS-485 to USB converter module has been employed for hardware connectivity, it appears as tty in Raspbian OS. The Vendor ID and Product ID of this converter is

used to fetch the right tty to be used for communication. It appears as *ttyUSBx* in our case. These values are passed using the configuration file, as mentioned earlier.

An example configuration file is shown below.

conf.ini

```
[report 0]
ReportDeliveryLocation = http://192.168.0.107:8079/add/GwjVoJy85K97c7Zi31naBkxJB6qG3iIn7gEt
[/]
Metadata/SourceName = Test Energy Meter New
uuid = 4a1ed488-a458-11e1-91af-00508dca5a06
[/example]
type = smap.drivers.myexample.Driver
METERS = 1
MODELS = MFR28
PARITY = E
BAUD_RATE = 9600
CYCLE_RATE = 2
VARIABLE_LOADTYPES = None
LOADTYPES = MS203
```

In order to compose, send and receive Modbus RTU messages over this interface, *modbus_tk* package has been used in the python script. This package also provides a logging feature using python logging module, CRC checking, error handling, message request timeout mechanism, etc. The source driver behaves as Modbus Master. It sends the message with function code "READ HOLDING REGISTERS" and the slave responds with the parameter values as requested in the message. Currently, all the parameters in the range are requested in a single transaction. Separate time-series information is maintained for each parameter. The values that are received in the slave

response message are converted into IEEE-754 floating-point decimal representation and added to their respective time series. In addition to the energy consumption parameters, device health-related parameters are also maintained as a time-series. These include data usage, system reboot count, virtual memory statistics, disk usage, signal strength, CPU temperature, etc. Knowledge about these parameters is likely to aid in establishing the possible cause when a device goes out of service. Hence, it provides device diagnostic information remotely.

The reporting manager can be considered as the **communication subsystem** of the edge device in the sMAP framework. It publishes this information to the central server when it is periodically scheduled. The time-series is also locally buffered, and a particular entry is removed only when it has been successfully published to the server and acknowledged as well. This helps in mitigating the issue of network or power outage. When the network or power is restored, all the pending entries are published to the server. The command to launch the daemon in the system startup phase has been incorporated so that it is not required to be manually triggered since the device will be deployed at the customer site.

Prototype - 2 and 3

Prototype 2 [22] has been developed using a proprietary python-based IoT framework developed as a part of this project. It currently supports any Linux based OS. This prototype uses Raspbian [23] OS to run the software modules. This framework aims to accelerate the development of IoT devices with data pruning subsystem such as pollution monitoring, temperature, and humidity monitoring devices, etc. It also provides an example IoT server, which can be modified depending on the use-case.

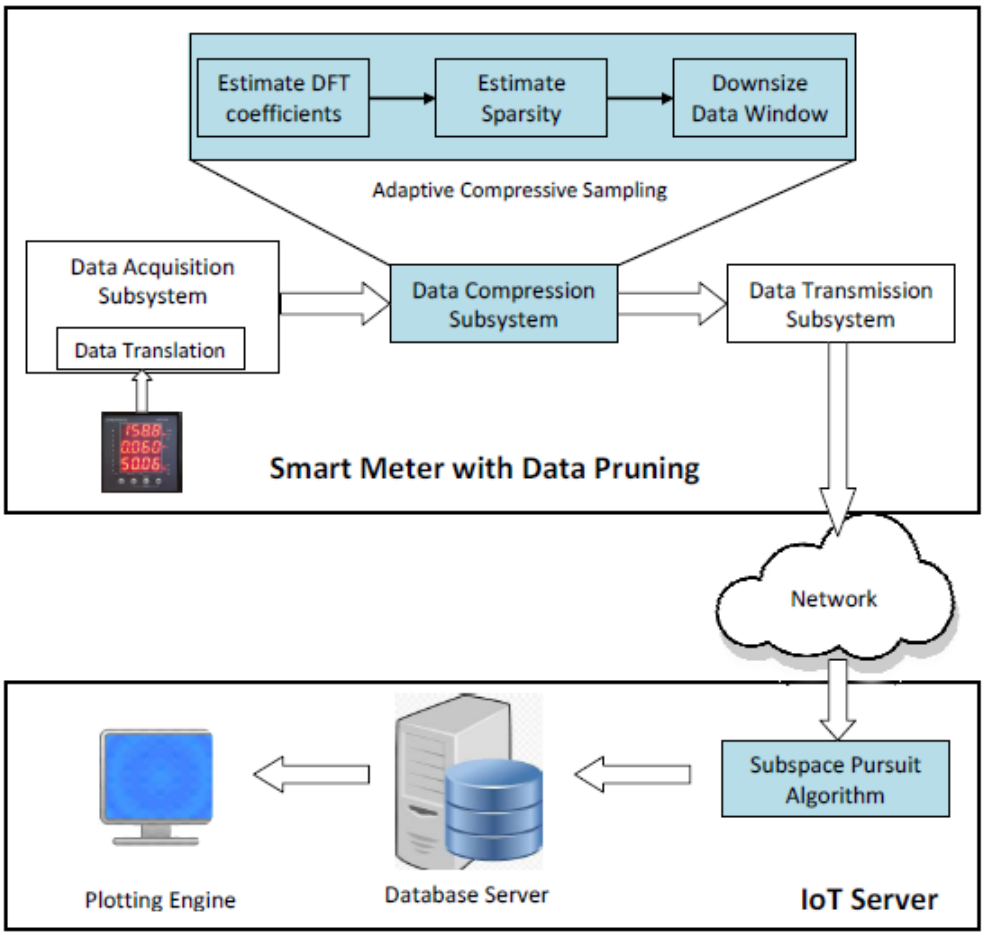

Figure 4.21: Smart Energy Meter with Data Pruning Subsystem

As shown in the upper part of the figure 4.21, the edge IoT device consists of 3 software modules broken down on the basis of their functionality. The three modules work in parallel in order to achieve the timing guarantees of data acquisition, compression,

and transmission.

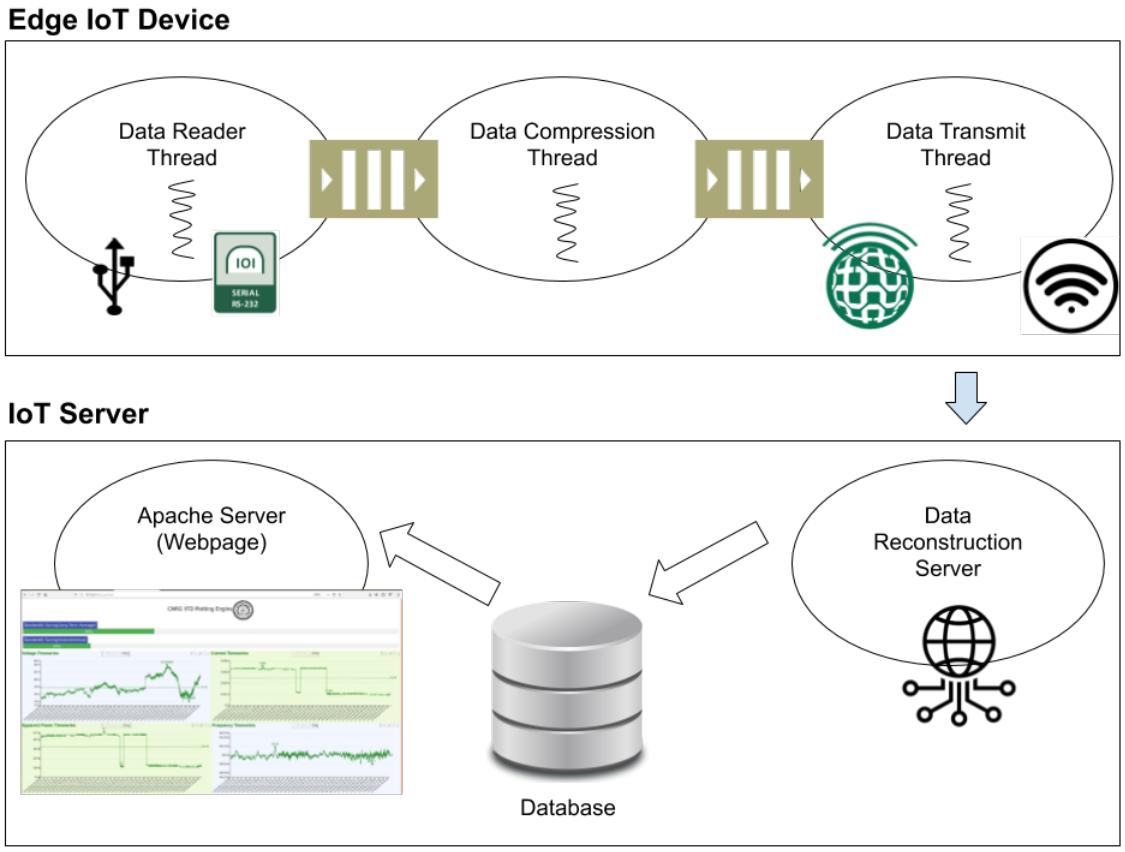

Figure 4.22: Smart Energy Meter with Data Pruning Subsystem : Task view

Figure 4.22 shows the decomposition of the various subsystems into OS tasks. For the prototype under consideration, each subsystem is implemented using the threading support provided by python language. Each module is a separate thread, and hence parallelism is achieved. The communication between each module is provided using messaging queues. The format of the message is predefined. Each IoT device needs to have three subsystems, namely, metrology, data compression, and communication subsystem in this architecture.

Metrology subsystem is entrusted with acquiring the data from the analog/digital device measuring the parameters depending on the context. It may provide the data in terms of analog/digital values, some sort of encapsulation pertaining to a data link layer level protocol, etc. It is also the job of this subsystem to do the required data translation

to the acquired data that will be enqueued to the data compression subsystem. In our context, IoT device is an energy meter, so energy consumption parameters will be measured. The off-shelf energy meter supports Modbus RTU protocol and is assigned a slave number, details of which have been explained in the previous section. The RS-485 port is connected to the USB port of the SBC. It appears a teletype terminal in the OS and has a unique number associated with it. Each USB device has a manufacturer and product code that is used in identifying this tty software port. This module makes use of *pymodbus* python package to handle Modbus communication and *threading* package for thread support.

The metrology subsystem constructs a Modbus RTU frame with a function code to read the holding registers and sends it to this tty. The slave responds with a frame. This frame is decoded using the specifications provided by the Modbus protocol. The off-shelf energy meter is a 3-phase meter supporting Modbus protocol; hence it provides an address mapping [14] for mapping parameters to their register addresses. Each parameter is stored in a 32-bit register, and the value is of type float adhering to IEEE-754 floating-point representation. So this conversion needs to be carried out by the metrology subsystem. Once the values of all the parameters of interest have been translated, metadata like meter identifier, geographical location, timestamp, etc. are appended to it, and it queued for further processing by the data pruning/compression subsystem.

Data Pruning/Compression Subsystem is entrusted with the task of intelligent data pruning of the data. This is a separate python thread. It dequeues the data sent by the metrology subsystem. The univariate data pruning algorithm that has been incorporated processes data in batches. This module makes use of *numpy* package for numerical operations. So this system keeps buffering the data until N items have arrived, where N is the optimum batch size. Therefore, this system requires some buffering capacity dependent on the value of N. Once the batch is ready, the algorithm is applied to the data. The output of the algorithm is the transformed data, which is encapsulated as a JSON packet. JSON related operations are performed using *simplejson* python package.

Communication Subsystem is entrusted with the task of establishment, communication, and termination of connection with the remote IoT Server. One of the key responsibilities of this module is the periodic transmission of data from the edge device. In the initialization phase of the edge device, this subsystem establishes communication with the remote server and also receives some dynamic configuration information. NTP daemon provided by the Raspbian OS is used to sync the time and timezone. This system is also required to have a much larger buffering capacity than the data pruning/compression subsystem since intermittent network connectivity needs to be mitigated. A reasonable value of 5000 entries has been chosen otherwise, the cost of the device is expected to rise significantly. This module makes use of *jsonsocket* package for sending JSON encapsulated data.

Prototype - 3 has been developed using a proprietary framework based on Arm® Mbed OS. This runs on STM32L476RG microcontroller as it is Arm® Mbed Enabled™. An alternate way of programming the controller is to use the *STM32 CubeMX* for generating the initialization code and thereafter using the *Keil IDE* to compile and flash the executable file. But Arm Mbed OS is preferred since it is an embedded OS designed specifically for developing devices in the Internet of things. This RTOS also provides threading support. It also provides memory pools and message queues natively. This prototype is based on the same architecture as the prototype - 2. It consists of the metrology, data compression, and data transmission subsystems. Each subsystem is a separate thread. Messages are exchanged between modules through the queues created using memory pools. In order to reduce the memory footprint of the code, the use of external libraries is totally avoided.

For the **Metrology Subsystem**, the Modbus RTU communication stack compliant to Arm Mbed OS has been developed as a part of this project. This stack currently supports READ HOLDING REGISTERS function code. It provides APIs to construct the Modbus RTU message frame along with CRC calculation. This module makes use of *UARTSerial* class for sending and receiving Modbus message frames. For converting the measured parameter values from the format as received in the Modbus response message to IEEE-754 floating-point decimal representation, an API has been developed.

Data Compression Subsystem implements the univariate data pruning algorithm. This is a separate thread running on Arm Mbed OS. FFT values for the optimum batch size are pre-computed and stored so as to reduce the processing time of the algorithm. This module has also been implemented without making use of external libraries in order to make it compact in terms of memory usage and also reduced execution time. It also converts the condensed information along with metadata to JSON format. An API has been implemented to convert numerical information to string representation as required by the JSON format.

Communication Subsystem interfaces with the WiFi module. It implements the semantics of communication with the ESP8266-01 WiFi module. ESP8266-01 WiFi module communication stack has been developed as a part of this project. This stack is compliant with ARM Mbed OS. It implements the state machine to decode the response generated by the WiFi module when a request is sent. The response may be a received packet or the execution status of a command. This module makes use of *UARTSerial* class for sending and receiving AT commands and data packets.

Prototype - 4

The proprietary python-based framework developed in Prototype 2 has been partly reused in implementing this prototype. The software module with multivariate data compression also runs on Raspbian [23] OS. The edge IoT device consists of 3 software modules broken down on the basis of their functionality. The modules work in parallel to meet the timing guarantees. Each module is implemented using the python threading framework, whereby communication across modules is achieved using message queues. The messaging format is proprietary. Each IoT edge device consists of three subsystems, namely, metrology, data compression, and communication system. The metrology and communication subsystem of prototype two have been reused. The major difference is in the implementation of the data compression subsystem, which incorporates the AMDC algorithm. In Prototype-2, each energy parameter data stream was treated independently, and an adaptive compressive sampling algorithm was employed. Here, multiple energy parameter streams are aggregated and then fed to a

PCA block followed by an adaptive compressive sensing block on the individual principal components. This subsystem receives a block of data comprising of a number of energy parameters for a particular sampling instance. It keeps buffering them until N number of samples are available. Hence, one batch consists of m-dimensional data for N sampling instances. The AMDC algorithm is applied to this batch, and data is compressed to a larger extent as compared to Prototype-2. The compressed data is translated into JSON format using *simplejson* python package and transmitted to the IoT server. Figure 4.23 depicts the software architecture of this prototype.

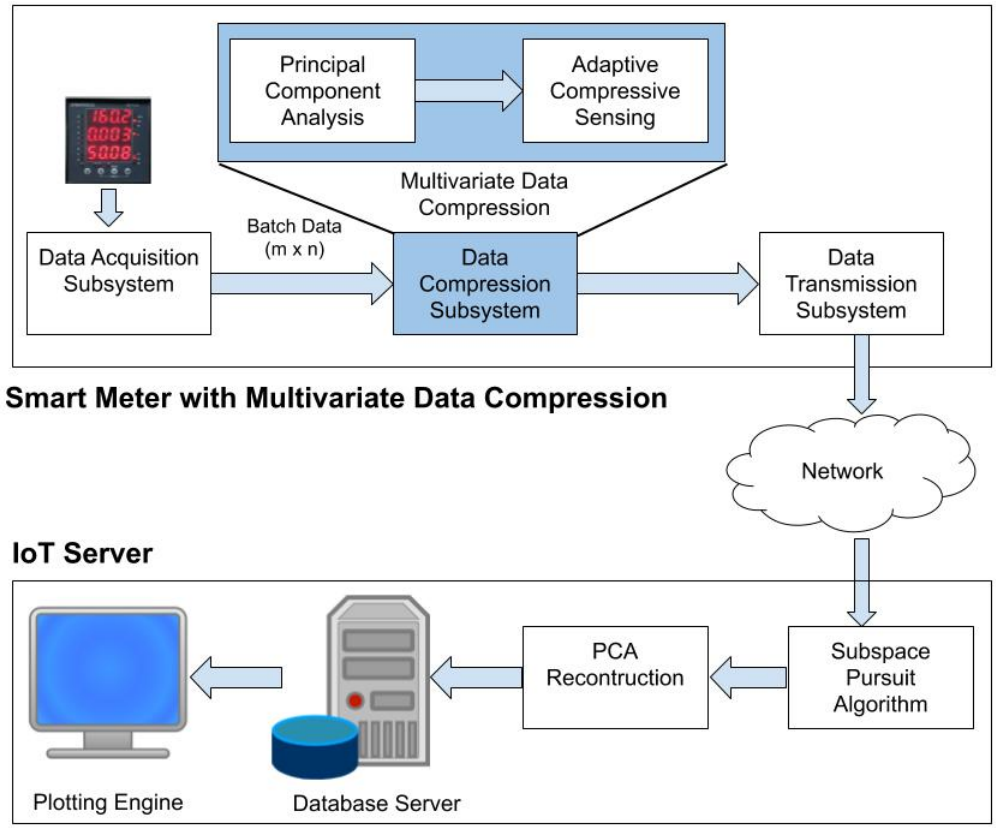

Figure 4.23: Smart Energy Meter with Multivariate Data Pruning Subsystem

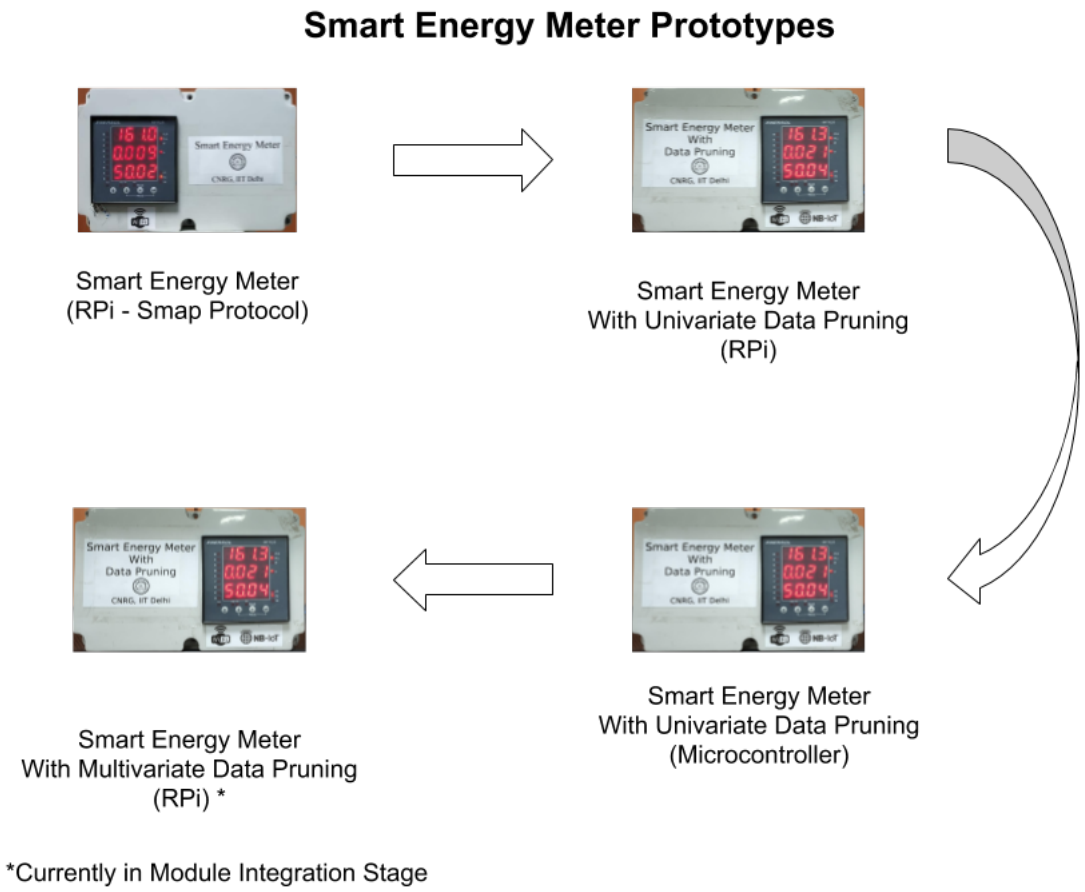

Figure 4.24: Smart Energy Meter Prototypes

Figure 4.24 depicts the series of prototypes developed as a part of this project.

4.2 IoT Application Server

4.2.1 Prototype - 1

The sMAP archiver [21] is a module that stores the time-series data in an efficient manner. It can be considered as a high-performance historian. It provides the various IoT devices, a storage facility to send their data. It supports :

- Time-series data storage and retrieval in an efficient manner.
- Usage of structured key-value pairs for storing the metadata.
- Provision to select the time series. This achieved by using the Archiver API and the Database Query Language.

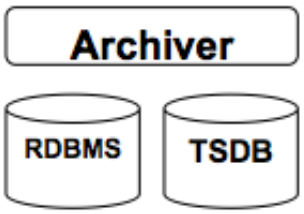

Figure 4.25: sMAP Archiver

The archiver is an application that stores data by making use of a relational database and time-series database. The relational database used is *postgres* and the time-series database is *readingdb*. It also provides a data dashboarding feature and a plotting engine. The powerdb2 project is provided out of the box. It provides features like time-series organization and plotting of the time-series. The front-end and the database are decoupled, and each can be run independently. It also provides query language support, which can be used by the end-user to develop front-end for displaying and organizing tree views of the data stream. The framework is quite flexible in this aspect.

Figure 4.26 shows the plot of a device health parameter i.e CPU Temperature. The front-end provides flexibility to select a particular parameter for a particular IoT edge

device from the drop-down menu, as shown on the right side of the figure. It also provides a button to auto-update the time-series, thereby making the plot dynamic. The time scale of the plot can also be selected by choosing the start and end date-time. It also shows the metadata for a particular parameter in the central portion of the web page. The selected time-series can also be downloaded as a CSV(Comma Separated Values) file by clicking on the button provided. Figure 4.27 shows the plot of a energy consumption parameter i.e Voltage.

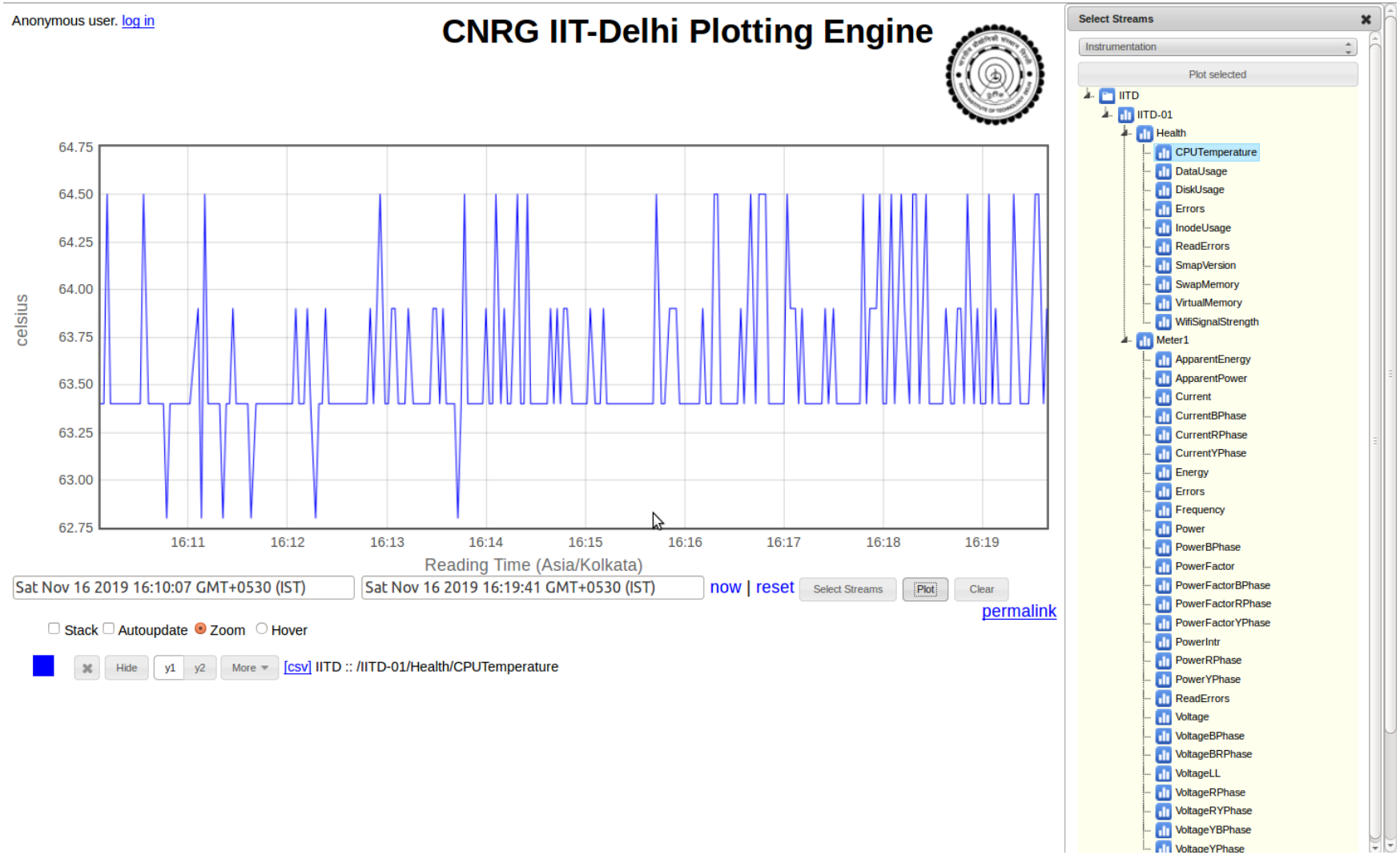

Figure 4.26: Health Parameter : Plot of CPU Temperature Time Series

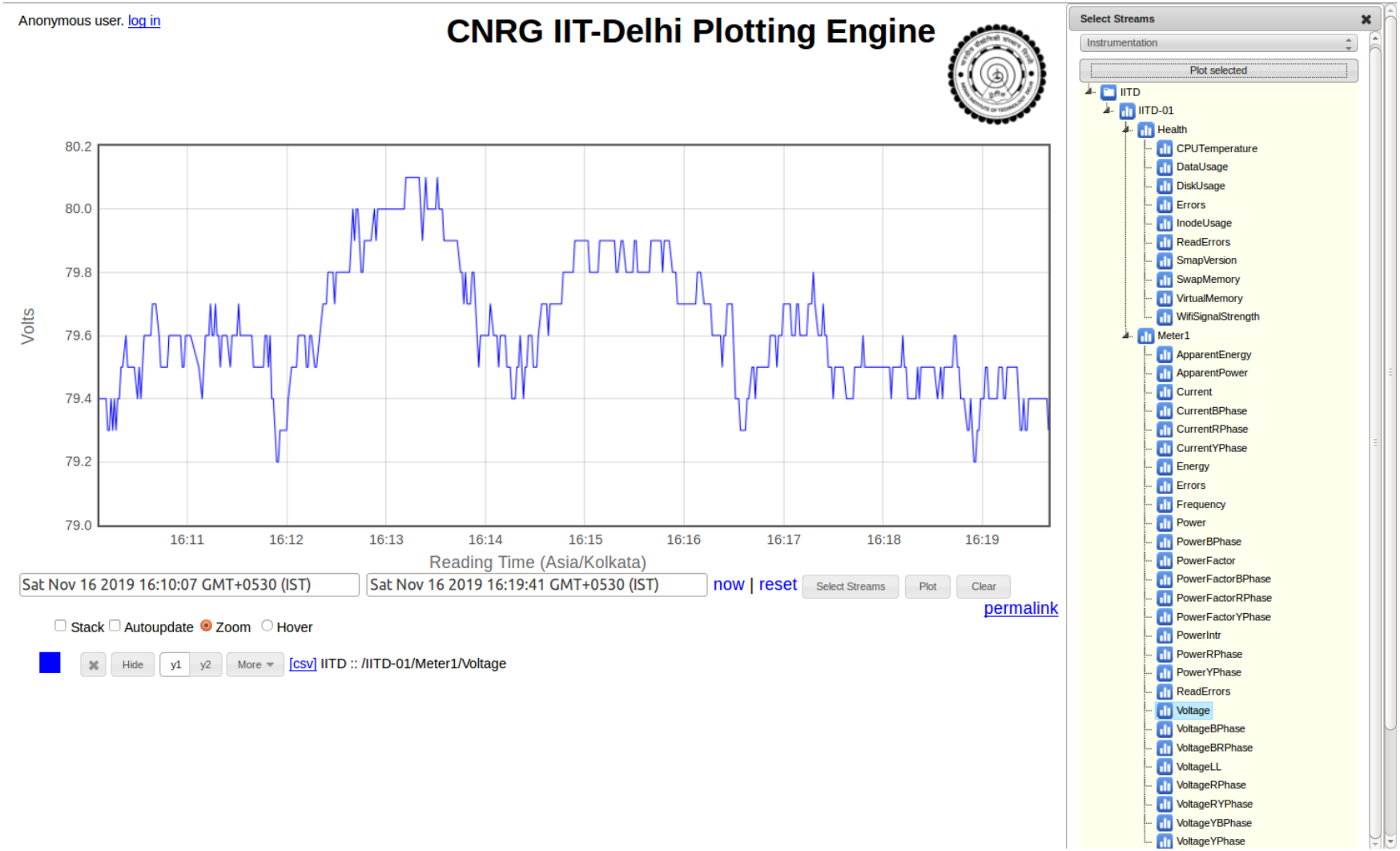

Figure 4.27: Energy Consumption Parameter : Plot of Voltage Time Series

4.2.2 Prototype - 2, 3 and 4

The IoT Application Server consists of an application accepting connection requests from the end nodes in the connection initiation phase. Once established, the server decompresses the data being sent by the various IoT devices, stores the timestamp, value tuple in the database indexed by the meter identifier. It also hosts a plotting engine which displays the time series to the end-user or the utility service provider. The IoT server consists of a server that is implemented using Python and a web page hosted using the Apache2 Web server. The two components are explained further.

- **Application Server** is entrusted with establishing a connection with the clients and maintaining the client information. It is also the job of this server to decompress the data and feed the data to a back-end DB. As soon as the edge device comes up, it tries to establish a connection with this server. When the connection has been setup, the first information that is exchanged in the capability message. This helps the server to maintain some meta-data of the client and associate the connection information with it. After the initial exchange of messages, the client starts sending the data as per the configured interval of time or sampling interval. Another component of this server works to decapsulate the payload received from the client, which contains the compressed time series along with other meta-data. For prototype-2 and 3, to reconstruct the original time series, the batch of data needs to be fed to a data decompression algorithm, Subspace Pursuit Algorithm in our case. The application of this algorithm yields a time series which is stored in the back-end DB. For prototype 4, the data decompression algorithm needs to be applied, followed by PCA reconstruction. The data decompression algorithm used in Subspace Pursuit Algorithm and the PCA reconstruction is achieved using the formula shown in 4.1.

$$PCA\ reconstruction = PC\ scores\ .\ Eigen\ Vectors^{\intercal} + Mean \tag{4.1}$$

- **Web Server** is a component that makes up the front-end of the IoT Server. The web page is hosted using Apache2 Server. The web page is implemented

using JavaScript and PHP. Currently, the webserver is a β-version. The web page provides the following information to the end user :

- Long Term Average Bandwidth Saving - A button is provided to refresh it.
- Instantaneous Bandwidth Saving - A button is provided to refresh it.
- Time Series Plot of :
  - Apparent Power
  - Current
  - Voltage
  - Frequency

The graphical plot is interactive in the sense that original and reconstructed values have been superimposed to demonstrate the reconstruction accuracy of the proposed algorithms. Buttons have been provided for each one, to switch on/off the plots. The plot for each meter can be viewed by accessing their respective web page, which is indexed by a meter identifier. A button is also provided to view the values stored in the time-series. By hovering the mouse pointer over the plots, value for a particular time instant can be viewed. It also marks the maximum and minimum value in the time-series that is currently displayed. The average value for each parameter is also displayed. Figure 4.28 shows the plot of the original values for four energy consumption parameters. Figure 4.29 shows the plot of the reconstructed values for the four energy consumption parameters. Figure 4.30 shows the plot of the original and reconstructed values for the four energy consumption parameters which demonstrates the reconstruction accuracy of the algorithm. Figure 4.31 shows the demonstration lab setup.

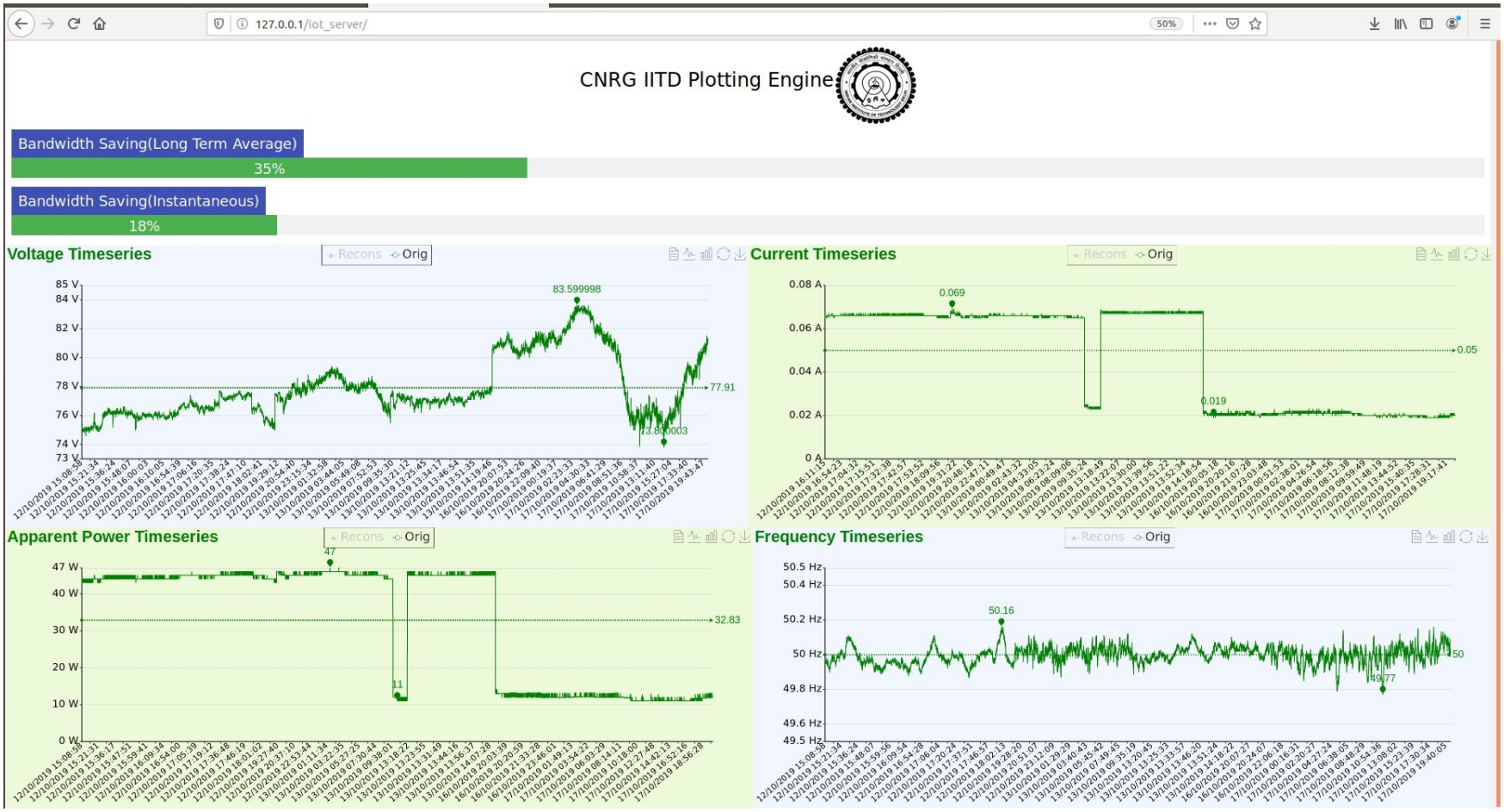

Figure 4.28: Plot of Original Time Series

Figure 4.29: Plot of Reconstructed Time Series

Figure 4.30: Plot of Original and Reconstructed Time Series

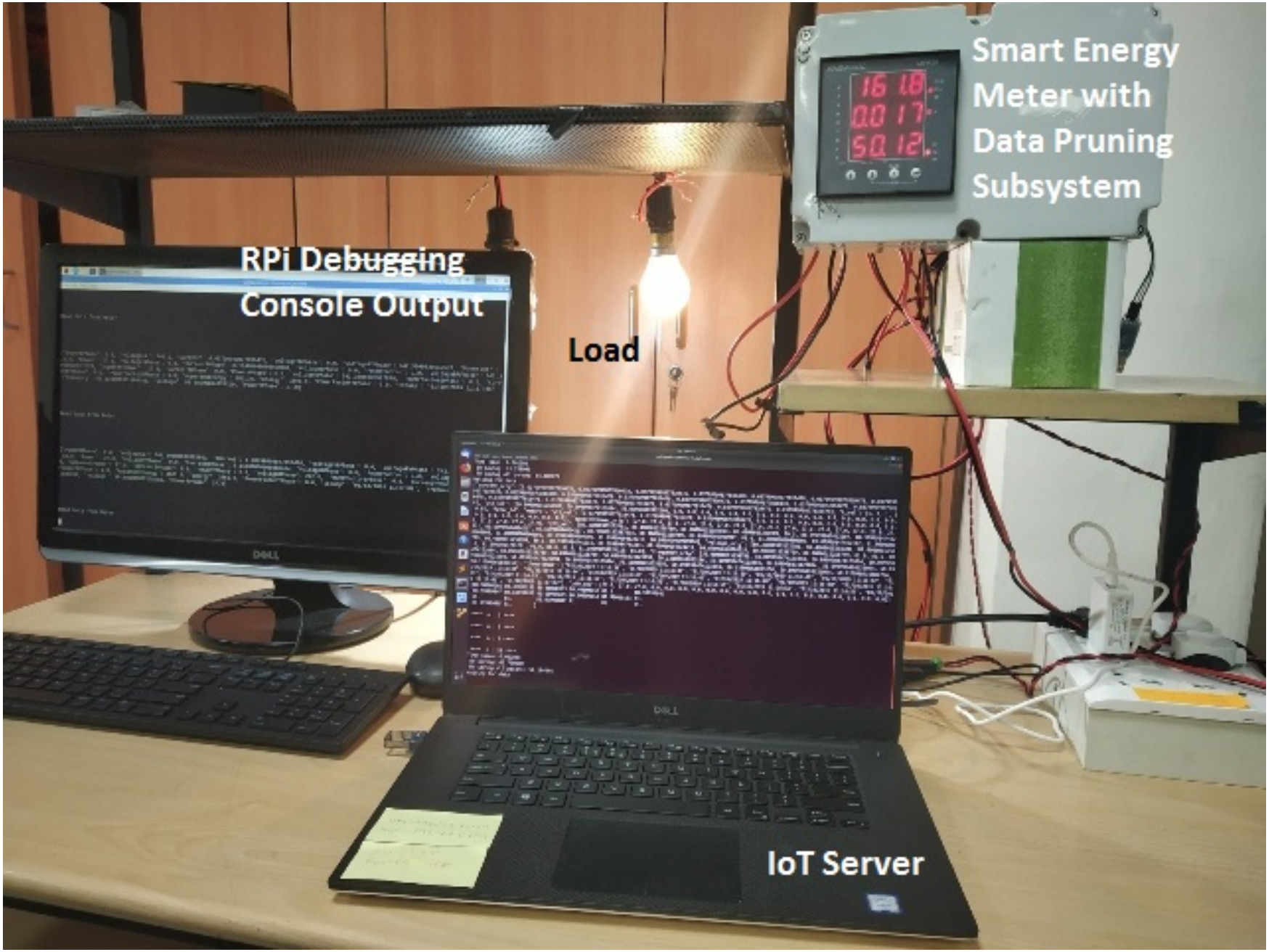

Figure 4.31: Smart Energy Meter Lab Setup

Chapter 5

Conclusions and Future Scope

5.1 Conclusion

IoT has emerged as a leading-edge technology for automation, process monitoring, and system control. Hence the growth of IoT is undeniable. In this work, we have tried to highlight that IoT devices should not be dumb devices sitting at the periphery of the IoT network. IoT is beyond providing network connectivity to existing analog/digital sensing devices. This work proposes a new software architecture to make the edge IoT devices intelligent in the context of data compression. Another contribution of this work is the development of a cognitive aware framework for carrying out further research in this direction. This framework will enable the quick transfer of technology to interested commercial organizations. This work exploits the state of the art data-driven resource optimization approaches mentioned in literature and makes use of it in actual real-world implementations. It demonstrates that the proposed algorithms can be used in real-world scenarios, and the benefits can be reaped. The application of this technology is limited not only to smart energy metering but can be easily extended to other domains by tuning the hyper-parameters of the data pruning subsystem.

5.2 Future Scope

From the discussions so far, it is quite evident that the incorporation of a data pruning subsystem in the commercially deployed smart meter will bring about a change in how edge IoT devices are implemented. It also encourages to evolve newer optimization techniques for different applications and make scalable, cost-effective, and secure solutions for IoT deployment. The proposal to integrate a data pruning subsystem in the processing pipeline of the IoT devices opens up a window of opportunities to make provisions in protocols like LwM2M, CoAP, etc. Also, the multivariate data compression algorithm can be implemented in a microcontroller to bring down the cost of the prototype and facilitate easy ToT to interested commercial parties. However, few challenges still need to be overcome namely,

- **Latency :** Currently, IoT traffic relies on existing wireless or cellular communication networks, which were not designed to support millions of IoT devices. But latency requirements need to be met in IoT scenario. The introduction of the data pruning subsystem aids in reducing the traffic to a considerable extent, a trade-off exists between the execution time of these algorithms and the reconstruction accuracy.

- **Security and privacy :** Nowadays, a massive number of IoT devices are being deployed in order to reap the benefits of the connected world. However, cost-effective and energy-efficient devices do not possess strong security measures, making them easy target for security breaches. Also, the devices that are already deployed hardly receive security patches for newly identified threats. Legacy devices that are being upgraded to be IoT compliant lack the in-built mechanism for modern threats. This demands the development of an industry-accepted security framework for IoT.

- **Energy efficiency :** An extensive range of IoT devices is being developed for various applications areas. Most of them are battery-powered, and incorporation of data pruning subsystems incurs an additional computational cost, thereby drain-

ing the battery further. This opens up new research avenues to make the computing resources energy-efficient and incorporate energy harvesting techniques.

- **Protocol specification :** In the current scenario, there is a lack of protocol specification for developing IoT devices. The integration of multiple platforms, multiple protocols at different layers, and a host of APIs poses technological risks and lacks cost-effectiveness. Also, the rapid evolution of IoT devices with proprietary feature-set in the absence of IoT standards has further complicated the situation. It is, therefore, the need of the hour to come up with a standard for IoT devices, which specifies the basic set of features all the IoT devices must provide, the scope for add-on features, basic security, and privacy mechanisms, etc.

Bibliography

[1] S. Tripathi, S. De. "Data-driven optimizations in IoT: a new frontier of challenges and opportunities", *CSI Transactions on ICT,* 35-43, 2019.

[2] Silicon Laboratories, Inc. Smart Metering Brings Intelligence and Connectivity to Utilities, Green Energy and Natural Resource Management. Rev.1.0. Available at: https://www.silabs.com/documents/public/application-notes/Designing-Low-Power-Metering-Applications.pdf, April, 2020.

[3] Intechopen, "Advanced Metering Infrastructure Based on Smart Meters in Smart Grid," https://www.intechopen.com/books/smart-metering-technology-and-services-inspirations-for-energy-utilities/advanced-metering-infrastructure-based-on-smart-meters-in-smart-grid, Jun 15, 2020.

[4] V. Gupta, S. Tripathi, and S. De, "Green Sensing and communication: A step towards sustainable IoT systems," *J. Indian Inst. Sc.*, (accepted, Mar. 2020).

[5] S. Tripathi and S. De, "Channel-adaptive Transmission Protocols for Smart Grid IoT Communication," *IoT J.*, (accepted, Apr. 2020).

[6] S. Tripathi and S. De, "Dynamic prediction of powerline frequency for wide area monitoring and control," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 2837-2846, Jul. 2018.

[7] M. Ringwelski et al., “The hitchhiker’s guide to choosing the compression algorithm for your smart meter data,” in *Proc. IEEE Int. Energy Conf. Exhib.*, Sep. 2012, pp. 935–940.

[8] M. Zeinali and J. S. Thompson, “Impact of compression and aggregation in wireless networks on smart meter data,” in *Proc. IEEE 17th Int. Workshop Signal Process. Adv. Wireless Commun.*, Jul. 2016, pp. 1–5.

[9] A. Unterweger and D. Engel, “Resumable load data compression in smart grids,” IEEE Trans. Smart Grid, vol. 6, no. 2, pp. 919–929, Mar. 2015.

[10] S. Tripathi, S. De, "An efficient data characterization and reduction scheme for smart metering infrastructure", *IEEE Trans. Ind. Informatics,* vol. 14, issue 10, pp. 4300-4308, October 2018.

[11] M. Roy Chowdhury, S. Tripathi, and S. De, "Adaptive multivariate data compression in smart metering Internet of Things," *IEEE Trans. Ind. Informat.,* (accepted, Mar. 2020)

[12] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing signal reconstruction,” *IEEE Trans. Inf. Theory,* vol. 55, no. 5, pp. 2230–2249, May 2009.

[13] Enersol Systems Official Website, “Programming Manual,” http://enersolsystems.com/Programming-Manual-Display-Parameter.pdf, Jun 15, 2020.

[14] Enersol Systems Website, “Address Mapping,” http://enersolsystems.com/address_mapping.pdf, Jun 15, 2020.

[15] Realpars Website, “Realpars Modbus Documentation,” https://realpars.com/modbus/, Jun 15, 2020.

[16] Pinterest, “FTDI FT232 ESP-01 Interfacing Diagram,” https://i.pinimg.com/236x/01/bb/7b/01bb7b8030df0c0d58ade7898042fd20.jpg, Jun 15, 2020.

[17] Waveshare, "SIM7020E NB-IoT HAT," https://www.waveshare.com/wiki/SIM7020E_NB-IoT_HAT, Jun 15, 2020.

[18] ST Official Website, "STM32L476RG," https://www.st.com/en/microcontrollers-microprocessors/stm32l476rg.html, Jun 15, 2020.

[19] Enersol Systems Official Website, "Connection Diagram," http://enersolsystems.com/connection_diagram.pdf, Jun 15, 2020.

[20] Zephyr Project, "ST Nucleo L476RG," https://docs.zephyrproject.org/1.9.0/boards/arm/nucleo_l476rg/doc/nucleol476rg.html, Jun 15, 2020.

[21] sMAP 2.0, "sMAP 2.0 Documentation," https://pythonhosted.org/Smap/en/2.0/, Jun 15, 2020.

[22] W. A. Khan, P. Das, S. Ghosh, M. Roy Chowdhury, S. Tripathi, S. Kaur, S. Chatterjee, and S. De, "Smart IoT Communication: Circuits and Systems" Research exhibit, in *Proc. COMSNETS* , pp. 1-2, Bangalore, India, Jan. 2020.

[23] Raspberry Pi Official Website, "Raspbian," https://www.raspberrypi.org/documentation/raspbian/, Jun 15, 2020.