Energy Management System Infrastructure for Solar Photovoltaic Energy System

by

Joey Zheqiang Li

A thesis

presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2022

© Joey Zheqiang Li 2022

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Joey Li was the sole author for Chapters 3-8 which were written under the supervision of Dr. John Wen and Dr. Jatin Nathwani and were not written for publication. Dr. John Wen, Dr. Jatin Nathwani, and Dr. Munur Herdem are co-authors on any publications relating to this works.

This thesis consists in part of two manuscripts that were written for publication. Exceptions to sole authorship of material are as follows:

Research presented in Chapter 2:

Dr. John Wen and Dr. Jatin Nathwani were the primary co-investigators on research grants which supported conducting this work. This research was conducted at the University of Waterloo by Joey Li under the supervision of Dr. John Wen and Dr. Jatin Nathwani. Dr. John Wen contributed to study design. Dr. John Wen, Dr. Jatin Nathwani, and Dr. Munur Herdem provided research guidance and feedback. Joey Li wrote the draft manuscripts, which all co-authors contributed intellectual input on.

Citations:

Chapter 2:

Li J., Herdem M., Wen J., and Nathwani J, Using Big Data, Artificial Intelligence, and Digitalization in Smart Energy Networks, 9th Global Conference on Global Warming: 164-167

Li J., Herdem M., Wen J., and Nathwani J, "Methods and applications for Artificial Intelligence, Big Data, Internet of Things, and Blockchain in smart energy management," *Energy and AI*, vol. 11, p. 100208, Jan. 2023, doi: 10.1016/J.EGYAI.2022.100208.

Abstract

In recent years, more distributed energy resources (DERs) such as solar PV, wind and more are being installed to compensate for the growing energy demand. It is vital to properly monitor and integrate these new energy resources. An Energy Management System (EMS) can be used to achieve this objective. This thesis is focuses on an EMS an infrastructure that can be used for a residential solar PV system. An EMS should be manufacturer independent, affordable, and easy to integrate.

This thesis presents an EMS consisting of a Monitoring and Control Platform (MCP), a data management infrastructure, and a client application. The MCP includes a Smart Energy Controller (SEC) and Motes that are placed at the output of the solar PV system, the input to the battery, and the input to the load. The user can send commands to the SEC via the web application. Then, the SEC sends commands to the Motes via Zigbee. The Motes can block or allow the flow of power using a power relay. Thus, the Motes allow precise control over the flow of power. The Motes also contain sensors that can measure the power. The measurements are sent to the SEC via Zigbee. Commands and alarms are logged by SEC. The system uses a neural network to predict the power production of a solar PV energy system. The SEC uploads the power measurements, the predicted power production, and system logs to a database where it is stored.

The client application allows users to access the Information stored in the database. The client application displays the measured power from each Mote, the predicted power production, alarms, and system control logs. The EMS was tested on a solar PV system with a single panel, a 12 V battery, and a DC load. It demonstrates the potential for low cost and independent energy management systems.

The thesis has designed an operational hardware infrastructure for an EMS that can be integrated into the existing solar PV systems. The EMS infrastructure is manufacturer independent, affordable, and easy to integrate. These features will help with faster integration of solar PV energy systems into a Smart Energy Network. Furthermore, it provides a platform that can host different energy scheduling and energy prediction algorithms. The functionality has been demonstrated through integration into a solar cart. Furthermore, a software and hardware architecture consisting of a sensor data collection system, control system, and a neural network has been successfully developed. The performance of these components was evaluated through multiple experiments. The sensor data collection system and control system were each tested separately and then tested concurrently. For the sensor data

collection, 500 sensor samples were collected and analyzed to demonstrate the reliability of the sensor collection system. The testing results indicate that the system is able to consistently and accurately collect the data. The performance of the remote energy flow control has also been tested. 500 commands were sent from the web application. The response times from the system were evaluated. The results show that the system executes the command and responds within 0.9 to 1.4 seconds. The neural network was trained and tested on actual data collected from a 250-kW solar PV installation in Gaithersburg, Maryland, USA. The mean absolute error of the power production forecasts was 6 kW or 2.4%. Furthermore, the cost of the presented system is estimated to be almost 18 times cheaper compared to existing energy management systems. This thesis provides valuable information for research and development of future energy management systems.

Acknowledgements

I wish to express my sincere gratitude and appreciation to my supervisor Dr. John Wen. He gave me an amazing opportunity to do research at this prestigious institution. His support has let me explore my research ideas to the fullest. Whatever I needed, Dr. Wen would try to accommodate me the best he could. I would also like to thank Dr. Jatin Nathwani for his suggestions in shaping my thesis ideas and providing funding support. I would also like to thank Dr. Zhongchao Tan and Dr. James Tung for being my thesis readers.

Next, I would like to thank my family for providing motivation and guidance. They have always been available for contact and supported my mental health. I would also like to thank Dr. Munur Sacit Herdem for providing career advice and helping me write my papers.

Lastly, special thanks to my close friends: Josh Martone, Clark Kneabone, Dylan Martone, Derek Churchill, Jason Kitoko and Anand Sandagdorj for keeping my spirits up and providing fun times.

Table of Contents

Author's Declaration ii

Statement of Contributions iii

Abstract iv

Acknowledgements vi

List of Figures ix

List of Tables xii

Chapter 1 : Introduction 1

1.1 Preface 1

1.2 Motivation 2

1.3 Thesis Objective 5

1.4 Thesis Contributions 5

1.5 Thesis Outline 5

Chapter 2 : Literature Review 6

2.1.1 Data Mining for Artificial Intelligence 6

2.2 Intelligent Algorithms 8

2.3 Big Data 22

2.3.1 Dimensions of Big Data 23

2.4 Digital Technologies 26

2.4.1 IoT Computing 28

2.5 Summary 34

Chapter 3 : Methodology and Materials 38

3.1 High Level Design 38

3.2 Data Management Infrastructure 42

3.3 Smart Energy Controller 47
3.3.1 Development of a Neural Network for Predicting Energy Production 47
3.3.2 Hardware 50
3.3.3 Software 53
3.4 Motes 59
3.5 Client Application 63
3.6 Summary 68
Chapter 4 Experimental Setup 69
Chapter 5 Results and Discussion 76
5.1 Scenario 1: System Validation 76
5.2 Scenario 2: Sensor Data Collection 83
5.3 Scenario 3: Control Performance 84
5.4 Scenario 4: Asynchronous Processing Performance 92
5.5 Scenario 5: The Neural Network Performance 97
5.6 Scenario 6: Cost Evaluation 107
5.7 Summary 111
Chapter 6 Future Research 113
Chapter 7 Conclusion 114
References 116
Appendices 129
Appendix A Specifications of the Digi Xbee 3 129
Appendix B Development of the Relay Control Circuit 129

List of Figures

Figure 1: Keyword Co-occurrence Network made from Query 1 results from Scopus 10
Figure 2: The popularity of major intelligent algorithms in literature from the Scopus database. 11
Figure 3: Keyword Co-occurrence Network made from Query 2 results from Scopus. 25
Figure 4: Keyword Co-occurrence Network made from Query 3 results from Scopus. 26
Figure 5: A high level diagram depicting the IoT architecture for an energy management system. 32
Figure 6: A diagram showing an example architecture for an energy management system. 35
Figure 7: High level diagram showing the key components and connections in a solar PV system. Inspired from [87]. 39
Figure 8: High level diagram showing the EMS in this thesis. The key components and flow of information are outlined. 42
Figure 9: A sample control log entry in AWS. The system is a string representing the EMS system ID. The epoch timestamp is the epoch time of when the action was recorded. The "ctrl_log_data" field is a JSON object. 44
Figure 10: Diagram summarizing the interaction between clients and the broker. 45
Figure 11: Diagram summarizing the AWS services used in the cloud layer and their interaction. 46
Figure 12: The pseudo code of how the ANN training data was processed. 49
Figure 13: The Smart Energy Controller Architecture. 53
Figure 14: Diagram showing how the asynchronous control loop functions in the SEC. 55
Figure 15: Pseudo code for the SEC function that checks commands and executes them. 56
Figure 16: Pseudo code for the SEC function that collects sensor readings. 57
Figure 17: Pseudo code for the JNXB3 58
Figure 18: High level diagram of the circuit used to control the relay. 60
Figure 19: Diagram showing the internal Mote architecture. Note that the Wattmeter and SSR and placed in series on the positive power line. 62
Figure 20: Pseudo code for the programming running inside the Mote. 63
Figure 21: Dashboard for the client application. This is the default view. 64
Figure 22: The control options window. Note that the 'COMMAND TEST' and 'Download CSV file' buttons are for testing purposes only. 65
Figure 23: The alarm logs table. 65

Figure 24: The graph for the PV power predictions. Note that this prediction is made from the NIST Campus data on July 20, 2017 66
Figure 25: The control logs table 66
Figure 26: Sample power measurements in real time power measurements graphs. 67
Figure 27: The inside of a Mote enclosure. 69
Figure 28: The inside of the Smart Energy Controller enclosure. 70
Figure 29: The Solar Cart circuit and the Mote placement 71
Figure 30: A picture of the experimental setup 72
Figure 31: A closeup picture of the Motes and the inside of the Solar Cart 73
Figure 32: The flow of expected behavior in the system 77
Figure 33: The “Toggle PV Mote” command as seen in the Jetson Nano terminal 78
Figure 34: The “Toggle Battery Mote” command as seen in the Jetson Nano terminal 78
Figure 35: The “Toggle Load Mote” command as seen in the Jetson Nano terminal 78
Figure 36: The “Success” message as seen in the web application 79
Figure 37: Transmit Failure as seen in the Jetson Nano 79
Figure 38: Transmit Failure as seen in the web application 80
Figure 39: Mote Error and Transmit Failure in the Alarm Logs 80
Figure 40: The Mote sensor readings in the command line 81
Figure 41: The PV Mote and Load Mote sensor readings on the dashboard. Note that the battery state of charge in the picture is a placeholder since the experiment did not actually use a separate battery. 81
Figure 42: The real time graphs of the Mote sensor readings in the web application. 82
Figure 43: The sensor readings obtained from the PV Mote, Battery Mote, and Load Mote in Case 2. 83
Figure 44: A plot depicting typical range of time that the system takes to send a command and receive a response 89
Figure 45: Sample measurements of the time it takes for the controller to process a command before sending it out to the Motes 90
Figure 46: Sample measurements of the time it takes for the Xbee communications portion of the control system to finish 91
Figure 47: Sample measurements of the time it takes for the controller to process the responses before sending it to AWS IoT Core. 92

Figure 48: The sensor readings obtained from the PV Mote, Battery Mote, and Load Mote in Case 4. 94
Figure 49: Plot depicting the range of processing times for the controller. The X in the middle represents the mean of the samples. 96
Figure 50: Visual depiction of the time it takes for each part of the system to complete. Note that the numbers used here represent the 95th percentile. 97
Figure 51: The predicted power vs actual power for data from April 2016 99
Figure 52: The predicted power vs actual power for data from July 2016 100
Figure 53: The predicted power vs actual power for data from October 2016 101
Figure 54: The predicted power vs actual power for data from December 2016 102
Figure 55: Bland-Altman plot comparing the Neural Network's predictions vs the actual measured values. 103
Figure 56: The program loading the model and making the predictions as seen in the Jetson Nano terminal. 106
Figure 57: The predicted power as seen in the dashboard. 106

List of Tables

Table 1: Cost of sample Smart Solar PV devices. 3
Table 2: Data mining algorithms that have been categorized according to their manipulation of data . 8
Table 3: Summary of intelligent algorithms that can be used for scheduling and control of energy resources 14
Table 4: Summary of key intelligent algorithms used for forecasting. 18
Table 5: A comparison of Zigbee, Bluetooth Low Energy, and Wi-Fi 5. 30
Table 6: Cloud services for IoT computing and data mining 33
Table 7: Supplement to Figure 6. Describes the digital entities, data, and processes in each layer 35
Table 8: The services used from AWS 42
Table 9: Some microcontrollers that can be used for IoT applications [94]–[97] 50
Table 10: The list of scenarios to be tested or evaluated 75
Table 11: The 95th percentiles and standard deviations for each time measurement for various sample ranges 86
Table 12: The 95th percentiles and standard deviations for each time measurement. Data is calculated from 500 samples. 87
Table 13: The 95th percentiles and standard deviations for each time measurement for various sample ranges for Case 4. 92
Table 14: The standard deviations and 95th percentiles for the command time measurements for Case 4. 95
Table 15: The metrics used to evaluate the ANN performance 98
Table 16: A performance comparison of various Neural Networks designed to forecast solar PV power prediction 104
Table 17: Cost of major components in the EMS developed in this Thesis. 108
Table 18: The cost of each energy management system after adjusting for the mark up and material cost 110
Table 19: The specifications of the Digi Xbee 3 [106], [127] 129

Chapter 1:
Introduction

1.1 Preface

Global renewable energy consumption is projected to grow by 147 % in the next 30 years [1]. In 2019, new global investments in clean energy were nearly ten times the amount invested in 2004 [2]. Furthermore, the share of renewable power in global energy generation has increased from 5.2% in 2007 to 13.4% in 2019 [2] . Allocating energy resources rationally and exploiting renewable energy play more and more important roles in today's life. Renewable energy technologies, such as solar photovoltaics (PV), wind turbines, and biomass-based conversion processes are excellent distributed energy resource (DER) solutions because they can be installed on roofs, plots of land, and even offshore. Therefore, there is high demand for the current electric grids, energy systems, and markets to be compatible with renewable energy generation systems.

To accommodate the demand, the grid must transition to a Smart Grid (SG) or Smart Energy Network (SEN) [3]. A Smart Grid and Smart Energy Network both have the following common goals: (1) Integrate, accommodate, and automate energy systems and energy storage; (2) Enable secure, real-time two-way communications between energy systems and stakeholders; (3) Encourage and allow participation from consumers; (4) Resiliency against attacks and disasters; (5) Promote adoption of renewable technologies; (6) Enable existing energy resources to be retrofitted with new technologies; (7) Seamless simultaneous implementation of digital technologies, stakeholders, and energy resources. It should be noted that a SEN is more comprehensive than an SG by accounting for additional energy resources within the information and communication technology infrastructure: natural gas, heating and cooling systems, hydrogen systems, and biomass [4]. However, integrating distributed energy resources will be more focused on renewable energy (versus blue hydrogen and natural gas) due to its prevalence and growth.

To integrate distributed energy resources within a SEN or SG, Information and Communication Technology (ICT) is required to monitor and co-ordinate the needs and capabilities of all resources and suppliers, energy transformers, infrastructure operators, end users, and energy market stakeholders [4]. Furthermore, the grid needs more automated systems that can manage these resources as well as generate quantitative estimates of site energy demand and production. Control

and projection of renewable energy generation, however, require a large volume of information, such as solar irradiance, humidity, windspeed, and so on. Therefore, Big Data, artificial intelligence, and digital technologies are important to build and implement an extremely sophisticated and advanced energy management system (EMS). An EMS may be designed for various sites such as Solar Photovoltaic (PV) farms, wind farms, and microgrids. The key component of an EMS is microgrids. A microgrid consists of a self-sustaining network of interconnected DERs and loads and it may be connected to a grid. A microgrid is a self-contained electrical network that allows end users to generate their own electricity on-site and use it when they need it most. Therefore, microgrids are valuable in SGs and SENs, because it helps integrating distributed energy resources effectively (4-6).

Based on the requirements for an ICT platform [3], an EMS should achieve one or more of the following functionalities: (i) monitor, forecast and schedule usage of site energy resources and loads; (ii) maintain bidirectional communication with the wider grid and local energy resources and loads; (iii) facilitate energy trading between the grid and other sites; (iv) store, process and communicate relevant data to stakeholders including site owner, utilities, grid operators etc. Efficient scheduling of resources reduces monetary costs, while real time monitoring allows for a quick response to any equipment faults. Bi-directional communication is essential because grid operators need to know how much energy is being consumed or how much is available; without real time data an imbalanced grid can result in system disturbances and in extreme scenarios blackouts. Furthermore, communication is vital to enable energy trading between multiple sites; if the grid is constrained and if a user's own supply is inadequate, the EMS has the flexibility to source energy from a neighbor's energy resources. This idea supports a robust and resilient system. Lastly, all produced or collected information should be made available to help utilities understand consumption/supply patterns, help users adjust their energy usage, and more. These benefits are the reasons why EMSs are important for the future of energy.

1.2 Motivation

An EMS can provide reliable access to power required for daily's life. Thus, it is even more essential for residential areas in rural or remote locations. A popular type of distributed generation is solar PV. In 2020, there was 138.2 Gigawatts of new Solar PV systems installed [5]. It is more attractive because it can be installed on houses in various capacities. Unlike geothermal or wind power which requires massive funds and land. Table 1 outlines the cost of energy management systems from

various companies. When looking for energy management systems, the following functionality was desired:

- Can be used for residential solar PV energy systems
- Can control the flow of electricity to loads or from energy resources
- Can monitor the flow of electricity to loads or from energy resources

Table 1: Cost of sample Smart Solar PV devices.

Manufacturer	Price (USD)	Components	Description
Enphase	$518 [6] + $1707 [7] = $ 2225	IQ Gateway + Enphase IQ System Controller	These are part of the Enphase Energy system [8][9].The IQ Gateway collects data from Enphase microinverters, Enphase batteries, and loads. While the IQ system controller connects Enphase batteries, and Enphase microinverters; the relevant feature is that it enables control of electricity to loads and from the solar PV system [10].
Tesla	$14 650 [11]	Powerwall	The Powerwall provides information on the battery usage and solar PV energy production. Note that the Tesla solar roof must be installed with the battery if buying from Tesla. The PV system is connected to an inverter built into the Powerwall. Using the Tesla Mobile application, users can view the collected information. They can also use it to manually charge or discharge the battery.

SolarEdge	$3250 [12] - 4040 [13]	SolarEdge Energy Hub Inverter	The Energy Hub is one of the devices in the SolarEdge smart energy management system [14][15]. The inverter acts as the central controller and aggregates data from other SolarEdge peripherals. Through the inverter, users can monitor energy consumption and production. It also enables users to schedule energy usage and control EV charging. Note that the inverter is only compatible with the SolarEdge battery bank or the LG Chem RSU battery series. Note that the inverter pricing scales with the rated power; higher rated power means it costs more money.

Note: The price references are included next to the number.

As shown in Table 1 these systems are expensive. With the cheapest option still being almost $2400 USD. However, it should be noted these prices are composed of multiple components. The prices factor in labor cost, materials cost, overhead costs, testing costs and more. A detailed breakdown on the pricing will be discussed in Section 5.6. It is important to keep in mind that the listed prices are composed of multiple costs.

These are not standalone energy management systems. For users with existing solar PV systems, it would require them to remove and re-install core components such as inverters or batteries. Users must buy into the whole system and cannot mix these parts together. Furthermore, the maintenance and installation services for these systems may not be available in all places. Remote areas should be given the same opportunities and benefits of grid modernization. Lastly, much of the reviewed literature discusses the development of key algorithms and theoretical processes within an EMS. However, there has not been much discussion on the design of the underlying hardware and software infrastructure. The next section outlines the objectives of this thesis.

1.3 Thesis Objective

This thesis aims to explore the infrastructure and operation of an energy management system for a residential Solar PV energy system. The objectives of the thesis are as follows:

I. Explore and develop a hardware and software infrastructure for an energy management system that is manufacturer independent, financially affordable, and easy to integrate
II. Test and evaluate the performance of the developed energy management infrastructure

1.4 Thesis Contributions

This thesis provides an infrastructure that can help with large scale integration of solar PV energy systems. The developed infrastructure is a low cost and practical system that is independent of the manufacturer. These features allow for faster global scalability. Solar PV is a valuable part of a SEN, thus solar PV integration helps transform the grid into a SEN. The proposed system also provides an infrastructure that can be modified to host different energy scheduling and energy prediction algorithms such as Neural Networks. This accelerates energy management system integration by allowing researchers to focus on developing algorithms, rather than the underlying infrastructure.

1.5 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 reviews the literature for AI, Big Data, and digital technology in energy management, Chapter 3 describes the methodology used to build the hardware and software infrastructure for a Solar PV energy management system, Chapter 4 describes the experimental setup used to test the energy management system, Chapter 5 discusses the experiments and the results, Chapter 6 presents future research directions for the EMS , and finally Chapter 7 presents a conclusion to the research in this thesis.

Chapter 2:
Literature Review

2.1.1 Data Mining for Artificial Intelligence

Data mining, in the context of energy management, involves data collection, preparation, and analysis [52]. Data mining aims to extract new information and relationships from existing raw data. Data preparation involves filtering outliers in the data set, filling in missing data, or transforming the data into specific data required by intelligent algorithms. For data analysis, a model or algorithm is used to abstract trends or relationships in the data and provide a desired output. For smart energy management, AI would be the model that performs the abstraction.

As reviewed in Section 2.1, intelligent algorithms have been used for load predictions and energy storage optimization in energy management. However, these types of models are especially reliant on proper data preparation. Ku and Jeong utilized data mining methods to help with the energy storage charging and discharging schedule for a building [53]. Their paper considers 4 unique loads: lighting, office equipment, fans, and a cooling coil. For the data preparation phase, they used a Self-Organizing Map (SOM) to determine which weather factors have the largest impact on the energy usage. SOM is an unsupervised NN model that non-linearly maps high dimensional data onto a low dimensional map space [53]. They used SOM to generate maps for each load and 25 weather factors. The colouration between the load and weather factor maps was used to determine their correlation. They complemented the SOM results using the Analysis of Variance (ANOVA) method. ANOVA is used to quantify the impact of certain factors on the variance of other variables. They used it to determine the impact of the weather factors on the variance of total building energy usage. This paper is interesting because: (1) It used two data mining methods to complement and verify each other; (2) It used a visual method, SOM, to prepare the data; thus, visual analysis can also be useful for understanding relationships between features.

Dou et. al, also used data mining to help in forecasting energy consumption, but also extended it to predict energy production [54]. The goal is to accurately predict wind power generation, PV power generation, and loads. The raw data was the historical load and generation for a specific site. The data is normalized and clustered via K-means clustering. Finally, variational mode decomposition is used to reconstruct the original data points of each cluster. The data is then used by a NN to predict energy

consumption. While data mining is relevant for forecasting energy resources, it is also vital in helping with demand response.

Babaei et. al used a data mining method called density-based spatial clustering of application with noise (DBSCAN) to determine which loads in a smart home can be shifted [55]. DBSCAN is a method for clustering data based on the density of the formed clusters. They claim that their method is more resilient to outliers and noise than other methods such as K-means clustering. Furthermore, their method does not require the use of data from sensors. The clustering determines at which time intervals, which appliances are considered non-essential and can be shifted. The clustering method is performed periodically, with updated load information, to update its calculation. Although user energy usage should generally obey a pattern, unexpected lifestyle changes may lead to changes in energy usage. Therefore, it is valuable to periodically understand and analyze energy usage patterns to obtain the most accurate schedule. artificial intelligence can also use the data on shiftable and non-shiftable loads to determine an energy schedule.

The data preparation is important to the data mining process and for training intelligent algorithms. If the input data is inaccurate or has errors, then the result will be inaccurate even for an optimized algorithm. The results from data mining are also valuable to stakeholders. The refined data can be displayed on the client application. For example, the load clusters from the DBSCAN method can be shown as categories to help utilities quickly understand the high demand times. Furthermore, researchers can reuse the refined data from collected inputs as training data for the AIs. Therefore, data mining also provide value to stakeholders

As seen in Figure 3, data mining is closely associated with analytical algorithms and forecasting. The keyword network is consistent with the literature. Furthermore, the diverse content from the literature show how data mining can be applied to detection, classification, and prediction. If the training or input data is not properly prepared, then it may not bring out the full potential of intelligent algorithms. In essence, the use of intelligent algorithms should be combined with proper data mining. Table 2 outlines some of the algorithms in the literature. In summary, data mining is an integral part of energy management as it can enhance fault detection, demand response, and energy forecasts.

Table 2: Data mining algorithms that have been categorized according to their manipulation of data

Transformation	Feature Selection / Extraction	Categorization
• Wavelet Packet Transform • Hilbert Transform • Z-score normalization	• Pearson Correlation • Relief Method • Correlation-based feature selection (CFS) method • Analysis of Variance • Self-Organizing Map • Variational mode decomposition • Weighted Least Squares	• density-based spatial clustering of application with noise • K-means clustering

2.2 Intelligent Algorithms

In this thesis, an "intelligent algorithm", is regarded as one that can autonomously give the best output based on varying inputs. Without the intelligent algorithms, the system would have to rely on fixed mathematical formulas with a well-defined process. The AI gains its adaptive abilities from multiple intelligent algorithms working together. There is not a fixed formula that the AI relies on. As will be explained, some of these algorithms rely on training and can be updated to improve their performance unlike a set mathematical formula. While others can adjust their actions based on inputs and outputs, thus making them more widely applicable. Algorithms such as deep learning and machine learning algorithms fall under this category. McKinsey Global Institute analyzed hundreds of use cases and claimed that Ais based on intelligent algorithms, such as deep neural networks, can improve performance over non-AI algorithms in approximately 70% of their use cases [16]. Furthermore, they claimed that deep learning techniques can generate between $3.5 trillion and $5.8 trillion in value each year. Intelligent algorithms have already been beneficial in commercial projects. Google used a machine learning algorithm to help forecast energy production of a wind farm. Using these predictions, Google was able to boost the value of the produced wind energy by 20 percent [17]. While IBM used machine learning to predict solar irradiation for a given solar PV site. Their model was 30% more accurate compared to the second-best model [18]. As will be shown in this section, intelligent algorithms extend beyond machine or deep learning. The primary idea is that

intelligent algorithms can be continuously improved and variably adjust their produced output based on inputs.

For each topic (i.e., AI, Big Data, and digital technology), a keyword of co-occurrence network (KCON) is generated from the literature search to help overview the trends. All keyword networks are generated from journal articles from 2017-2022. VOSviewer was used to generate each KCON. Note that separate KCONs were generated from Scopus and Web of Science (WoS) literature, however, they were very similar. Thus, only the KCONs generated from the Scopus data are presented in this thesis.

Figure 1 is a KCON for AI papers in renewable energy management from Scopus. The AI KCON was generated from 1927 journal articles. KCONs visualize the relationship between keywords used in papers. The bubble associated with each keyword represents its occurrence in the literature; a bigger bubble means the keyword was used more often. While the colouring of the bubble represents its relationship with other keywords; the closer the colouration is, the more strongly keywords are associated with each other. Note that strong relationships can still exist between bubbles of different colours. The KCON can provide a high-level view of the popular ideas and combinations of ideas in the literature. The keywords are essentially topics. KCONs are useful for bibliometric analyses and to understand the relationships between topics and discover new ones. Thus, a researcher can use a KCON to find new research topics or further refine their search criteria. For example, an author may want to see how neural networks are commonly used. From Figure 1, it can be observed that neural networks are used for solar power generation and wind power applications.

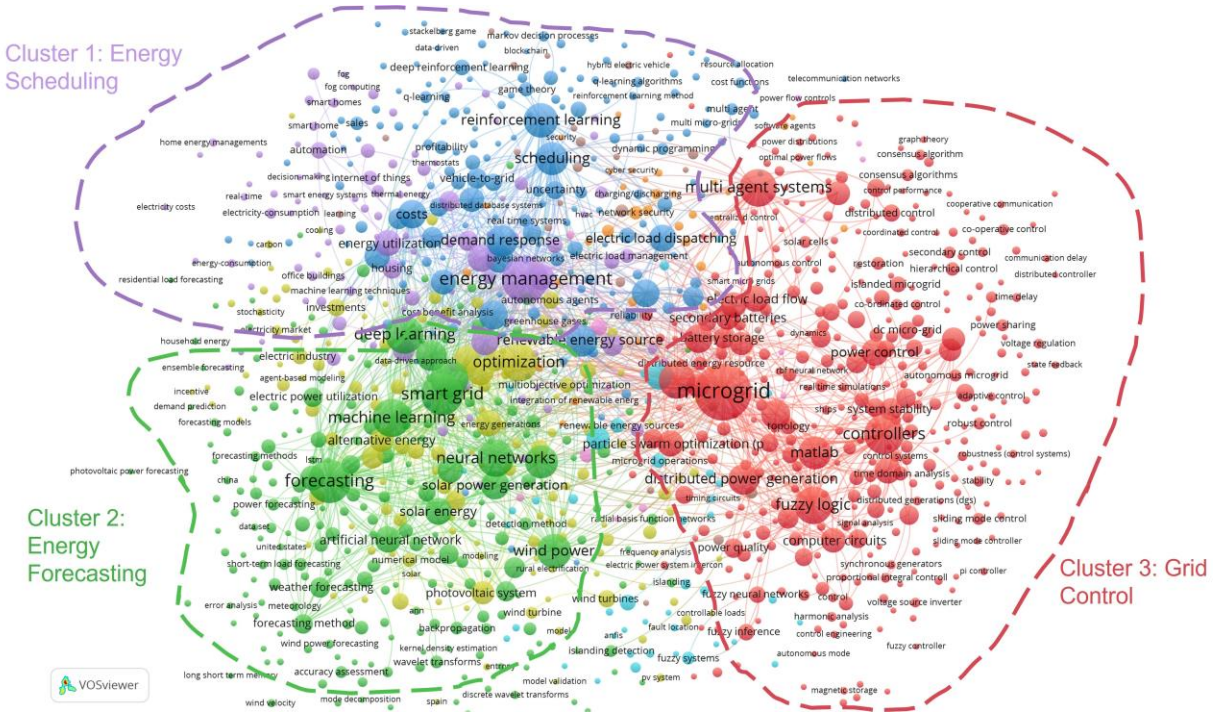


Figure 1: Keyword Co-occurrence Network made from Query 1 results from Scopus

Figure 1 gives a high-level idea of how AI is used in literature. The keywords selected all appeared in the literature at least 5 times. Furthermore, several keywords were omitted to avoid redundancy. For the artificial intelligence KCON the following keywords were removed: “electric power transmission networks”, “microgrids”, “micro grid”, “energy management systems”, “renewable energy resources”. The keywords skewed the relationships in the network; thus they were removed. There are three primary clusters: Energy Scheduling, Energy Forecasting, and Grid Control. The first cluster, Energy Scheduling, is associated with using intelligent algorithms to help plan the consumption of energy. The key applications for AI-based energy scheduling include demand response, electric load dispatching, and charging/discharging of energy resources. Looking at the fringes of the cluster, there are many keywords that relate to intelligent algorithms or energy consumption.

The next cluster is energy forecasting. It focuses on using intelligent algorithms to predict the production of energy; either directly or via weather phenomena. Based on the keywords, forecasting applications include load, solar power, and wind power forecasting. Furthermore, some of the AI

related keywords include neural networks, deep learning, and machine learning. Most of the keywords on the fringes are intelligent algorithms or relate to forecasting.

Lastly, Cluster 3, Grid Control, is concerned with the operation of grid assets with intelligent algorithms. Based on the keywords, the control of grid assets involves multi-agent systems, distributed control, and hierarchal control; the focus is moving away from centralized systems. It appears the goal of these methods is to enable autonomous operation of energy resources and ensure power quality. The edge of the cluster has many keywords that relate to energy control. Based the Scopus keyword co-occurrence networks, there are three primary areas where intelligent algorithms are used: Energy Scheduling, Energy Forecasting, and Grid Control.

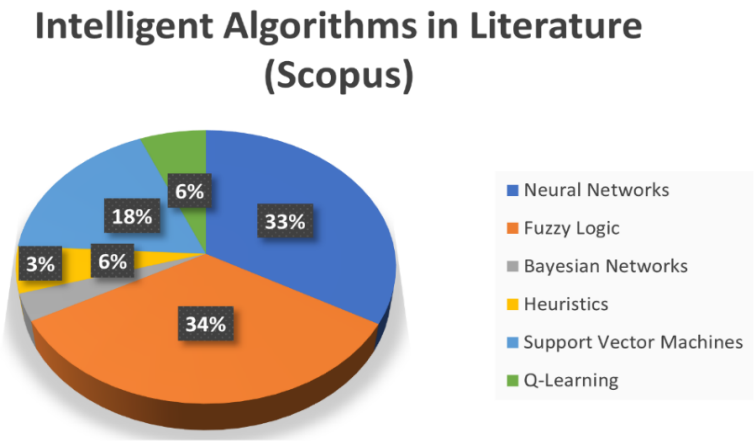


Figure 2: The popularity of major intelligent algorithms in literature from the Scopus database.

Figure 2 visualizes the popular intelligent algorithms or models in the literature. However, not all the intelligent algorithms are used in the literature during manual review and checking keyword occurrences these algorithms came up often. Furthermore, these algorithms encompass many subtypes as well. Thus, these chosen algorithms were the most representative and current ones in the literature. The most prevalent algorithms appear to be Fuzzy Logic and Neural Networks. The discussion will begin with Fuzzy Logic. Fuzzy Logic is used to give a degree of certainty to elements. The elements are assigned a value between 0 and 1, with 0 being “completely false” and 1 being “completely true” [19]. In a Fuzzy Logic system, the inputs first are put through a fuzzifier, which creates a fuzzy set from the inputs. Then the rules of the system are applied to the fuzzified inputs by

an inference engine. The rules help determine what action should be taken. Based on the evaluation, the firing strength of each rule with respect to the input is determined. The fired rules alter the fuzzy set, which is then defuzzified to obtain the output; in this case, the output is an action [20][21]. The popularity of Fuzzy Logic (FL) can be primarily attributed to its technical adaptability and wide range of applications.

Kofinas et. al use a hybrid Fuzzy Q-learning model. They apply it to a multi-agent system (MAS) to manage a microgrid [20]. The hybrid model uses a Q-learning function to help determine the rules for the fuzzy logic controller. Based off the Q-value and state, the action is chosen. After an action, the state is updated, then a reward is calculated. Based off the rewards, the q-value associated with the action is updated. Every agent controls a load or resource. The cost functions for the agents are primarily based off the current demand and power generation. Agents representing the energy storage also factor in the current state of charge. The calculated reward for each agent is based on the evaluation of the combined actions of all agents. The general idea is that each agent needs to balance its consumption or supply with the other agents.

Alfaverh et. al. combines Reinforcement Learning and Fuzzy Logic in a home energy management system for residential demand response [22]. The aim of the system is to minimize the power consumption. It shifts the operation of appliances from peak hours to low demand hours. Instead of an MAS, Al-Sakkaf et. al use Fuzzy Logic in a central energy management system to control supply and consumption in a microgrid [23]. The performance is optimized with Ant Bee Colony Optimization (ABCO). The inputs and outputs of the Fuzzy Logic System (FLS) are scaled by the ABC to compensate for errors in the estimation of power needed.

The literature shows that Fuzzy Logic is used as the central control algorithm for energy resources. Thus, the actions should be choosing when to supply the load and when to supply the grid. Both actions include deciding which resources are used to supply the electricity. It needs information like the power demand (kW), electricity price ($/kWh), State of Charge (SoC, %), and power output from the site energy resources (kW). The FLS can use measured real time data or forecasted data. A FLS can use different time intervals to perform actions. The forecasted data can be given anywhere between 10 minutes to 24 hours; specific numbers are shown in Table 4. The FLS can schedule its actions based on the forecasts. Alternatively, as real time data comes in, it can perform its actions

based on the current state. Calculated estimates of the power demand, electricity price, State of Charge, and energy resource power output can also be used as inputs to an FLS.

Examining these papers reveal a pattern starts that can explain the popularity of Fuzzy Logic. Fuzzy Logic is mathematically flexible because it has been combined with MDP, Q-learning, and neural networks. Furthermore, it is adaptable to different applications including energy forecasting, demand response, and energy balancing. It can be said that its flexibility can be partially owed to its simplicity; if it was a complex algorithm with too many factors to consider or modify, then it would not be as widely combined or modified. Fuzzy Logic is based on an action and reward system, it is therefore suitable for the applications that involve controlling an observable state, like charging and discharging energy resources. Although Fuzzy Logic is useful, it is worthwhile to examine another intelligent algorithm that can be used for energy scheduling.

Q-learning uses the idea of Reinforcement Learning (RL). In Reinforcement Learning (RL), the agent aims to minimize or maximize a number. For energy management this can be energy costs or energy usage. In the case of energy management, the appropriate action may be influenced by the environment. It will perform an action that will change the environment. Then, it will adjust its actions based on the new environment state. The feedback from each action will produce a "reward" which helps the RL algorithm understand the correct actions to take. Thus, the RL algorithm will take actions that can maximize its reward [24]. For energy scheduling, the environment can be the energy resources and load. While the actions can be controlling the flow of electricity in the site. This is useful for managing energy resources because of the uncertainty in energy consumption and production.

In Reference [25], the Foruzan et. al discuss microgrid management and energy trading in a multi-agent system. The agents are based on model-free Q-learning algorithms. Q-learning uses the idea of Reinforcement Learning (RL). Q-learning is useful for managing energy resources because of the uncertainty in energy consumption and production. Autonomous energy trading is a vital part of a SEN, and it encourages integration of renewable energy systems. Suppliers are incentivized to have distributed renewable energy because they can profit. While consumers are encouraged to consume from these resources because they can save on energy expenses.

Reference [26] uses a model-free Batch RL algorithm. Unlike Q-learning which discards its history after each update, batch RL keeps its history and uses it to influence its next actions. The goal in this paper is to maximize the energy usage of the on-site storage and reduce energy bought from the gird or sold to the grid. An approach that considers history is useful because there are trends in energy; for example, a person's consumption schedule, the peak solar production, or electricity prices throughout the day. Ji et. al. also devised an energy management algorithm that considers energy production and consumption in a microgrid [27]. However, they used a hybrid deep reinforcement learning approach called Deep Q-Network (DQN). In essence, it uses the idea of actions and rewards from Q-learning to tune a neural network. This approach is interesting because it is seen how reinforcement learning can be combined with another algorithm.

There are a couple of key points from the selection of these well cited papers. Like Fuzzy Logic, RL is versatile. RL-based algorithms can consider history (Batch RL) or be integrated with a Neural Network (DQN). RL algorithms can be applied on a macro scale, where they are used to coordinate actions between multiple agents. They can also be applied on a smaller scale to help optimize energy costs for home appliances. RL algorithms can also be used to manage a single site. In summary, the popularity of Reinforcement Learning can be attributed to its flexibility in applications and algorithm creation. In addition, RL-based algorithms such as Q-learning are "model-free". This means that they do not require a priori information on the environment. Thus, the same model can be applied to multiple systems in different locations. Thus, the model-free nature of Q-learning encourages the widespread adoption of energy management systems. Table 3 summarizes intelligent algorithms that can be used for energy scheduling. Although Fuzzy Logic and Q-learning are useful for energy scheduling, it is not as suitable for applications like energy forecasting, where the focuses are on numerical analysis of relationships and reducing numerical errors on the output result. Alternatively, there are other algorithms that can do this.

Table 3: Summary of intelligent algorithms that can be used for scheduling and control of energy resources

Intelligent Control Algorithm	Inputs	Objective	Target Environment	Reference

Adaptive Neuro-Fuzzy Inference System (ANFIS) + Fuzzy Logic	• Daily power generation • Daily power consumption • Solar irradiance (W/m^2) • Temperature (°C)	Predict the power output of the load and PV system (15 minute intervals) and curtail the load to match the PV generation and remaining battery power	Distribution network with a solar PV system, battery, and load	[28]
Fuzzy Q-Learning	• Power consumption of each load • PV power production • Battery Charge or Discharge power	Each agent aims to balance supply and meet power demand of each load while minimizing usage of the diesel generator	• Sources: PV System, Fuel Cell, Diesel Generator • Loads: Electrolyzer unit, desalination plant, variable electric load	[20]
Fuzzy Q-learning	• Shiftable Appliances (with priority levels attached to each one) • Non-shiftable appliances • Numerical reward based on whether action is Bad, Good, or Very Good	Monitor, control, and optimize the energy consumption	• Household with various appliances (Washing machine, Dish Washer, TV, Laptop etc.)	[22]
Fuzzy Logic w/ Artificial Bee Colony Optimization	• Renewable power generation (kW) • Load demand (kW) • Battery SoC • Fuel Cell power output	Meet load demand while maximizing usage from renewable energy and the battery and minimizing generation cost from the diesel	DC microgrid with a PV system, wind turbine or diesel generator, fuel cell, and battery energy storage connected to a house	[23]

		generator and fuel cell		
LSTM	• Current SoC • PV energy generation • energy demand • time of use tariff	Generate Charge and Discharge Schedule for EVs in the building parking lot	Commercial building with PV solar system and EV charging stations	[29]
Model-free Q learning	• The bids on the market (if a load agent) • Numerical reward based on whether action increased profit (if supply) or lowered expenses (if load)	In a microgrid with an auction-based energy trading system, each agent, representing a load or supply, aims to maximize their revenue (if a supply) and minimize their expenses (if a load)	A microgrid with energy storage, PV power systems, wind turbines, diesel generators, and customer loads	[25]
Batch Q-learning	• Previous actions and rewards • Battery SoC • Load • Inverter efficiency • PV energy production • Electricity price	Schedule the charge or discharge of a battery storage system (in 15-minute intervals) while minimizing the energy bought from or sold to the main grid	Microgrid with a load, battery storage system, and PV power system	[26]
Deep feedforward neural network + deep Q-network	• Operational cost of the distributed generator • Operational cost of the ESS • Price of selling electricity to the main grid	Schedule the charge and discharge of the energy storage system and local generators while minimizing the daily operating expenses and	Microgrid with various renewable energy generators, loads, and energy storage system	[27]

	• Cost of buying electricity from the main grid • Load	satisfying the load		

Table 3 shows that Neural Networks and Support Vector Machines are also widely reported in the literature. A standard NN is comprised of an input layer, one or more hidden layers, and an output layer. First, the raw data enters the input layer. Then it is processed by the hidden layer; this is where it tries to find the “meaning" in the data. There can be multiple hidden layers. Each hidden layer has an activation function. The inputs must go through the activation function first and the output may be propagated to other layers. Each layer is constructed from one or more neurons. These neurons contain activation functions. These activation functions are used to process non-linearities within the data [30][31]. Finally, the output layer provides the conclusion or prediction from the hidden layers. To make these predictions, a dataset is made with features and a desired output. The features are the data that the NN uses to predict the output. In reference [32], the feature data are meteorological data; air temperature, global irradiance, and relative air humidity. While the output is the predicted PV generation, datasets should contain many different entries, so the NN can train from a wide variety of situations. A Neural Network (NN) is commonly used in energy forecasting for wind energy resources, solar energy resources, or load demand [33][34][35].They have also been used for forecasting weather phenomena such as solar irradiance or wind speed in energy management systems [36][37].

Knowing the predicted power generation helps determine whether there will be excess energy that can be sold or stored. It can also inform users if there will be insufficient energy and the load must be supplied by the grid. However, consistently accurate predictions can be difficult because of to the intermittent nature of solar power generation; this fact alone motivates us for intense research of using NNs for energy or weather forecasting. Furthermore, NNs are not rigid algorithms and can “learn” from errors, thus continually improving NN-based forecasting. NNs are as flexible as Fuzzy Logic. They can be combined with a Fuzzy logic algorithm or a Support Vector Machine (SVM) to further improve performance for a given goal [20][38]. Similar to Neural Networks, Support Vector Machines can be trained to learn relationships; they’ve also primarily been used for energy-related forecasting [39][40][41][42]. As observed from the literature, both NNs and SVMs can be used for

energy forecasting. Again, their strength is abstracting numerical relationships between inputs and outputs; this idea also makes them useful for classification. Table 4 summarizes the reviewed intelligent algorithms that can be used for forecasting.

However, unlike Fuzzy Logic, there is no inherent mechanism for defining the states and associated procedure. Furthermore, Fuzzy Logic has a reward system that guides the algorithm in each action. These reasons can explain why they are more popular for energy forecasting as opposed to energy control. Both Neural Networks and SVMs have numerical parameters that can be modified. These parameters adjust the way the models "learn". However, it is not feasible to attempt to manually adjust the parameters. Using optimization algorithms such as Particle Swarm, Cuckoo, Dragonfly, and Genetic Algorithms [38][39][40][41] can iteratively tune the model.

Table 4: Summary of key intelligent algorithms used for forecasting.

Intelligent Forecasting Algorithm	Inputs	Objective	Target Environment	Optimization	Reference
Efficient Deep Convolution Neural Network (EDCNN)	• Dew point temperature • Dry bulb temperature • Wind speed • Wind power (current hour-24) • Wind power (current hour-25) • Wavelet Decomposed wind power • Time of the day	Predicted Wind power (MW) at each hour for the next 24 hours	900 MW Wind Farm in Maine, New England		[33]
Hybrid Model (Long Short Term Memory (LSTM) + Convolutional Neural	• temperature • humidity • holiday status • day of the week • season	Predicted power demand (GW) for the next 7 days	Daily power demand in South Korea		[34]

Network (CNN))					
Attention Long Short Term Memory (ALSTM) Neural Network	• PV power • PV module temperature • Day of year (1-365) • Time step (7.5, 15, 30, 60minute intervals)	Predicted power demand (kW) at each time step (7.5, 15, 30, 60minute intervals) for each day	20 kW rooftop PV power station in Shaoxing city, Zhejiang Province, China		[35]
Elman Neural Network	Solar irradiation from 5 weather stations which is then transformed by Discrete Fourier Transform	Predict solar irradiation for the next 24 hours for a given city	Qingdao, Shandong Province, China		[36]
Hybrid Model (two Recurrent Neural Network s+ Backpropag ation Neural Network + autoregressi ve moving average model + improved complete ensemble empirical mode decompositi on)	• Maximum wind speed • Minimum wind speed • Mean wind speed • Standard deviation	Forecast wind speed at 15minute intervals	Wind turbine farm		[37]
Hybrid feed-forward particle swarm artificial neural network model	• Energy demand • Environmenta l Temperature • Carbon Dioxide	Predict the required Biomass flow (kg/h), Syngas flow, and gasification plant airflow	Biomass Gasification plant	Particle Swarm Optimizati on	[43]

	• Carbon Monoxide (%) • Methane gas (%) • Hydrogen gas (%) • Nitrogen gas (%) • Active Power • Power factor • Generator frequency • Lower heating value • Airflow to internal combustion engine	to meet the energy demand			
Hybrid Model (Wavelet Transform + Particle Swarm Optimization + Support Vector Machine)	• solar radiation intensity • temperature • cloud cover • humidity • pressure • wind speed	Forecast power generation of a PV power system every hour	480 kW PV power system with six arrays in Beijing, China	Particle Swarm Optimization	[39]
Support Vector Machine	Wind speed (m/s)	Predict power generation of a wind farm every 10 minutes	1500 kW wind Farm in Penglai, Shandong Province, China	Cuckoo Search Algorithm	[40]
Support Vector Machine	• historical load data • ambient temperature • weather data (unspecified)	Predict the power consumption of a microgrid every 40 minutes	Offshore oilfield microgrid	Dragonfly Algorithm	[41]
Hybrid Model (Support Vector	• Number of Household • High-Income households	Predict the hourly power consumption of a	Rural microgrid in South Africa		[42]

Regression+ LSTM)	• Medium-Income households • Low-Income households • Number of Water Pumping • Number of Milling Operations • Number of Small Shops • Number of Schools • Number of Clinics • Number of streetlights	microgrid containing residential and commercial loads			

Both models continue to see research because there are still many ways to approach parameter tuning. Furthermore, training these models also requires selection of data relevant to the site and local weather. When forecasting energy for loads, the input data can be the historical load demand for the given site. However, the input data differs depending on the resource due to the varying nature of energy resources. For example, PV energy forecasting may use solar irradiance (W/m^2), ambient temperature (° C), humidity (%), cloud cover index (%), and so on. Whereas wind energy predictions may use wind speed (m/s), wind direction, ambient temperature (° C), air pressure (kPa) and more. Furthermore, PV systems and wind turbines in each site have different specifications. This means a model trained on data from one site may not be suitable for a different site. The same weather conditions may not yield the same energy for different systems. Thus, a big challenge is to have these models work for any site in any location.

The papers presented in Table 3 and Table 4 show that the current trend of intelligent algorithms is heading towards hybrid models. Researchers are using algorithms to optimize other models to improve their performance. The literature also shows that the algorithms are being linked together and the outputs are being passed through multiple stages to refine the results. Furthermore, it appears that Fuzzy Logic, Neural Networks, and Support Vector Machines are all popular algorithms for use

in hybrid or multistage models. These models are commonly used for energy forecasting or controlling energy resources. Model performance can be improved via optimizing algorithms. In summary, hybrid and multi-stage models are the current trend for intelligent algorithms in energy management.

Examining popular intelligent algorithms shows a couple notable trends. First, versatility is a prime factor in current research for artificial intelligence in energy management. Algorithms should be easily integrated or linked with other ones. No single approach is perfect, thus combining different types of algorithms together enable a more well-rounded system. Hybrid and multi-stage learning systems are becoming more popular. Neural Networks and SVMs are widely employed for forecasting. While Fuzzy Logic and Q-Learning are used for energy resource control. All these algorithms are vital for energy management. There are different levels of management in a smart grid. It can range from controlling specific appliances in a home to coordinating energy trading across multiple entities.

In summary, utilizing hybrid and multi-stage intelligent algorithms for energy forecasting and energy resource control are the current and foreseeable future trends in energy management. While intelligent algorithms and automated analysis are vital to energy management, they are nothing without data. The reviewed literature shows various useful data. Moreover, large volumes of data are needed to effectively train these algorithms. Therefore, data mining is a vital step needed before training.

2.3 Big Data

Conceptually, Big Data (BD) is the management of large volumes of data. Big Data (BD) systems can process a large amount of data ranging from Terabytes (TB) to Petabytes (PB). The process includes organizing data, receiving data, storing data, and processing data requests [44]. Dimensional Research reported that about 44% of organizations in 2018 had a data repository of over 100 TB [45]. Ecolibrium Energy, an energy analytics firm in India, uses a data analytics platform to process over 100 million data points daily [46]. Their analytics have helped reduce power consumption by 5-25% in such industries as iron and textiles. These references indicate that the immense volume of data generated by digital technology in a Smart Energy Network. The benefit and importance of Big Data comes from the inherent utility provided by the data; training AIs, providing performance insights,

monitoring equipment, and more. Reference [47] reports that the analysis of data can help utilities generate 20 – 50 billion USD, reduce outages by 35 percent, and reduce maintenance and labour by 20 percent. In essence, an EMS cannot function effectively without data management. In 2016, IBM estimated that 'bad' data costs the USA approximately 3.1 trillion USD [48]. Bad data contain errors, biases, or irrelevant data. The report also states that 60% of time is spent on correcting and organizing the data. Thus, to minimize bad data, it is vital to analyse the data. Reference [49] surveyed participants asking which information technology is most impactful for organizations. More than half of the participants believe that Big Data Analytics is Ihe most impactful technology for organizations [49].

2.3.1 Dimensions of Big Data

To understand how BD can be applied to energy management, it is important to consider the following seven dimensions of BD [50][51]:

1. **Volume:** Volume represents how to effectively store and organize data. For this dimension, it is vital to consider the amount of data generated over certain period. The period may be minutes, hours, or days. With an understanding of the potential volume, it becomes easier to design systems and methods that can handle the load.
2. **Velocity:** Velocity represents the frequency that new data is coming in (units/second) and the size of the incoming data (bytes). Thus, researchers must consider whether their Big Data system can handle the incoming volume of data. The system should also minimize delays in data transfer. Some data in a SEN is time sensitive. For a proper grid operation, the grid operators and utilities need to know beforehand the amount of energy being fed into the grid and the expected energy demand to effectively balance the energy flow in the grid. In an energy management system, there is real time data from various sources, including the smart meters, sensors (e.g. solar irradiance, temperature, and humidity) and internally generated Data (ex. energy predictions).
3. **Variety:** Variety considers the data from site sensors, smart meters (semi-structured data), system status reports (un-structured data), database queries (structured data), user requests for data, etc. An EMS is part of the larger ICT platform in a SEN that integrates data for a neighborhood, city, and country.

4. **Veracity:** Veracity represents the accuracy of the collected data. For an energy management system, this means that the collected data should have minimal biases and no errors. For example, placing a sensor to measure ambient temperature right next to a fan would obviously artificially change the temperature and produce inaccurate measurements.
5. **Validity:** Validity represents whether the data is being appropriately used. The data may have a good veracity but could be invalid for a specific application such as substituting readings of PV panel temperature from an ambient temperature sensor. It may seem like a good approximation, but it is fundamentally different and the right sensor readings should be used.
6. **Volatility:** Volatility represents the length of data retention. This is deeply tied with Volume because it is unfeasible to retain data forever. The system continuously expands and could result in longer processing times because it must filter through more records. Volatility can be measured by time (e.g., data is retained for 30 days) or if the storage exceeds a certain amount of bytes. Therefore, there must be policies in place to erase data. This can be determined by examining the veracity of the data and external factors. For example, employees can delete their account and other personal data related to them when they leave the company.
7. **Value:** Value represents the benefits gained from using the data. The value generated from the data must be greater than the cost for facilitating the resources to store and extract the data. Energy management systems should consider the following factors for data usage:
    (a) Contribution to accurate prediction of energy production and energy demand. The data should be omitted if they do not significantly contribute to the model's predictions. Therefore, it is necessary to determine the impact of each factor by testing them in the intelligent algorithms using different combinations of feature data.
    (b) Usefulness to stakeholders. Even though it may not be immediately useful to the system operation, data can be extracted for others that can use it. For example, the local utilities may need a record of how many Solar PV systems are installed. Consumers may want to know their hours and days of highest energy use.
    (c) Accessible to governments. They may require access to data to set policies or help public programs.
    (d) Flexible to Stakeholders' needs. They may request additional data for their own applications.

These dimensions should help researchers develop and further understand the data requirements. They are useful for architecting Big Data platforms and datasets. With an overall understanding of the Big Data dimensions, it is now possible to discuss the KCON in Figure 3. In the Scopus KCON, "electric power transmission networks" and "smart power grids" were removed because they skewed the keyword relationships. The Big Data KCON was generated from 160 journal articles. As mentioned, the KCON can provide a high-level view of the popular ideas and combinations of ideas in the literature. In this case, we want to see how big data is applied to energy management.

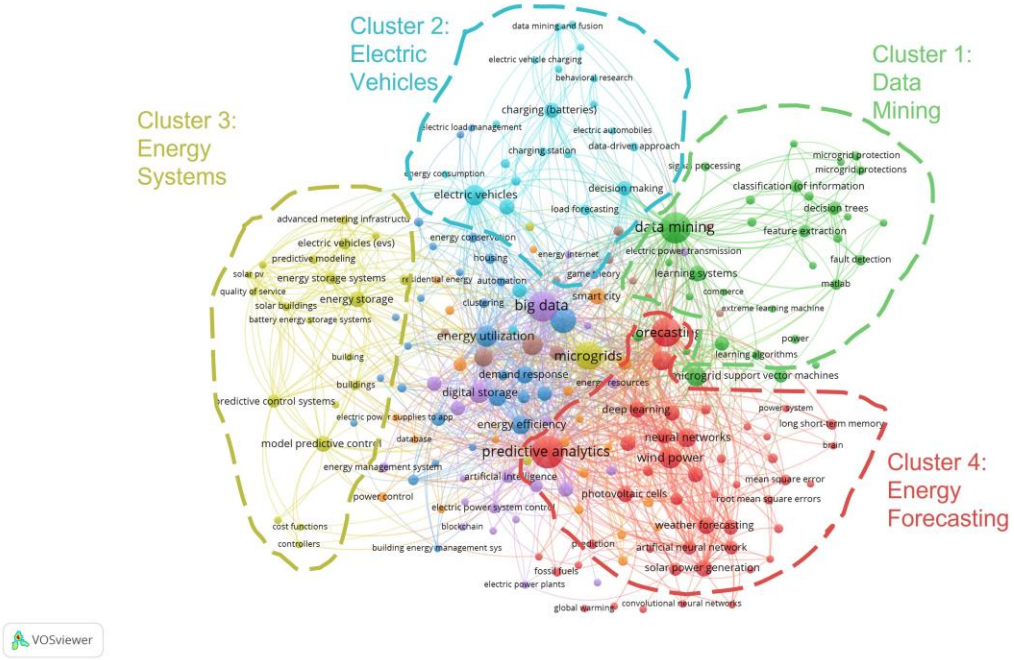


Figure 3: Keyword Co-occurrence Network made from Query 2 results from Scopus.

Data mining is one of the more coherent clusters in Figure 3. Data mining is used in feature extraction for AI training data and fault detection of electrical equipment. The data mining keyword bubble is directly connected to forecasting and energy storage as well. Section 2.1 already discusses energy forecasting. Cluster 2, electric vehicles are another type of energy load and resource, but do not directly relate to Big Data. Similarly, cluster 3 has many keywords related to energy systems. However, Clusters 2 and 3 shows that the concept of big data has been used with energy forecasting, electrical vehicles, and other energy resources. Cluster 1 warrants a deeper analysis since it has different applications. Thus, the most relevant topic to discuss would be data mining.

2.4 Digital Technologies

Digital technology has become common in our every day lives; laptops, smart phones, cloud storage, and so on. Therefore, it is important to look at how we can leverage and integrate digital technologies in a smart energy network. Digitalizing the grid involves integrating sensors, intelligent algorithms, and digital frameworks into the existing infrastructure to improve energy efficiency and plant operations, accelerate adoption of green transportation, and so on. Many of the intelligent algorithms used require a digital infrastructure and Big Data [56]. Using digitalization for building operations can potentially reduce total energy consumption. For example, Reference [57] states that total global building energy consumption can be reduced by 10% between 2017 and 2040. Additionally, Reference [58] surveyed stakeholders in the energy industry and 67% of them said that automation and digital workflow had a substantial impact in their organization and the sector.

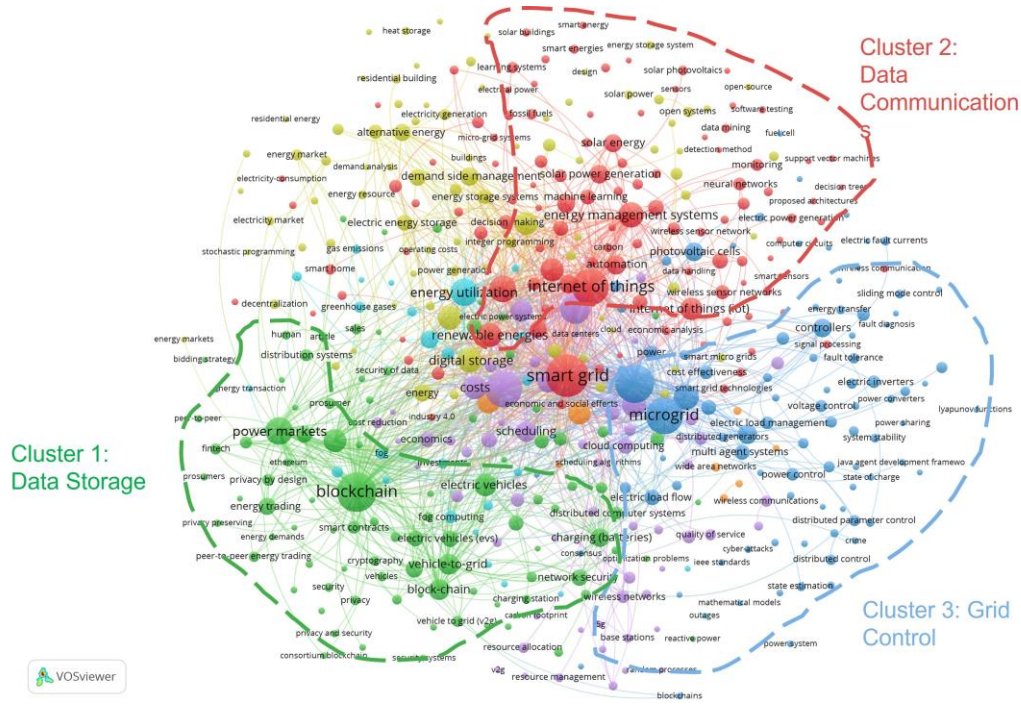


Figure 4: Keyword Co-occurrence Network made from Query 3 results from Scopus.

Figure 4, the KCON for Digital Technology, further explains the current digital technology research in smart energy management. The Digital Technology KCON was generated from 808 journal articles. The KCON shows that two clusters focus on data technologies. As discussed previously,

data mining is useful for improving performance of intelligent algorithms and refining data. Thus, there has been a focus on researching information technologies for data processing. Internet of Things (IoT) is a major keyword in the Data Communications cluster in Figure 4. As explained in Section 4.1, the primary value of utilizing IoT technology is enabling interoperability between components. Reference [59] estimates that utilizing IoT technology can potentially add billions to the global economic value of energy management. BeeBryte developed an energy management software that uses an IoT gateway to connect with HVAC and other digital systems like Programmable Logic Controllers. The interconnectivity provided by the IoT gateway enables the software to collect real time information and control various devices [60]. A DHL Logistics center in Singapore uses this system and managed to save 300 000 USD annually and avoided 1000 tonnes of CO_2 emissions per year [61].

The Data Storage cluster shows Blockchain as a major keyword. The others are closely associated keywords like power markets, energy trading, privacy, security, and smart contracts. Blockchain provides a public platform for transactions. Thus, it is useful for trading energy in a decentralized manner. The records are immutable and visible to all members of the platform. This enables a secure public marketplace for users to buy and sell energy. Latrobe performed economic modelling and their results showed that a blockchain-based marketplace could saved consumers can save an additional 6 to 12% on their electricity bills [58]. Furthermore, sellers could make an additional 15-25% on their sales. Reference [62] claims that Blockchain can boost GDP by 1.76 trillion USD by 2030 and enhance around 40 million jobs. Blockchain is one example of how digital technology can be used to facilitate energy trading. This cluster shows that researchers are looking into using digital technology to create secure marketplaces where electricity can be bought and sold.

The last cluster focuses on grid control. Some of the keywords include controllers, wireless networks, and distributed control. Thus, research is also focused on applying digital technology to control energy resources. An important part of control also involves monitoring. For example, General Electric (GE) power has utilized Amazon Web Services (AWS) to help monitor power plant operation [63]. Each day, their digital system can process 20 billion data records from power plant sensors and streams 500,000 data points per second. Furthermore, GE Renewable Energy uses AWS to increase their data streaming volume by 500% [64]. Using digital technologies allows for quick processing of a vast volume of information. Faster processing leads to increased data availability, thus

improving response times to problems and providing real-time insights that lead to more relevant actions.

The literature reviewed so far is focused on mathematics and modelling. In terms of energy management, this is a vital first step; the theoretical foundation must be sound. The next step is to verify the work after the simulations. In this case, verification requires physical implementations using digital technologies. For example, sensors and computer systems are needed to gather data and to process data or run the algorithms, respectively. Without these implementations, the value of the research becomes mitigated. Thus, increasing digitalization also means enabling the benefits provided by AI and data mining. Monitoring energy resources requires sensors and communication devices, and processing data needs a server to run on.

2.4.1 IoT Computing

This section outlines how digital technology can be used to integrate AI and data mining into an energy management system. Moreover, the clusters in Figure 4 reveal a focus on researching grid control and data communications. Within these clusters there are keywords like internet of things (IoT), wireless sensor networks, distributed control, fault detection, neural networks, and so on.

Many devices in a SEN are constantly streaming data. Thus, a data processing infrastructure is needed to store and transport this information (e.g., the smart meter, sensors, the renewable energy system, other digital sources like those from weather stations). The data can be used for improvement, real time detection, and analysing problems in the systems. For instance, identifying the cause of a blackout and isolating any damaged circuits. Furthermore, a proper data management infrastructure is needed to enable AI and data mining.

For example, an AI forecasting algorithm needs the current power consumption. Thus, each load may have a sensor that collects this data. Alternatively, a smart meter can be used. Sensors and smart meters belong in the “edge” layer of IoT computing. Mads et. al discuss the usage of Packetized Energy Management (PEM) to communicate and coordinate with multiple devices on the site [65]. Each load can make a request to use a ‘packet’ of energy. In this case, the loads are also edge devices. This message is sent to the local PEM coordinator, which can either reject or accept the request. The decision is based on the microgrid or grid conditions: electricity pricing, current local generation

output, Battery SoC, and more. The PEM performs some data processing and decisions on site. The processed information may be sent to an off-site data center for storage and further processing. In this case, the PEM is in the fog layer. Typically, multiple fog devices may be connected to a single data center. One of their purposes is to reduce traffic going to a data center. One fog device can be connected to the data center instead of having all the edge devices directly connected to the data center. Furthermore, pre-processing the data reduces the computational burden at the data center. This data center stores the information from all connected devices and may also perform data mining. Data centers typically reside in the cloud layer. edge, fog, and cloud computing paradigms fall under the concept of the "Internet of Things". Thus, a system that uses these paradigms can be regarded as an Internet of Things (IoT) Platform. The basic components of an IoT platform are:

1. **Collection**: Devices that gather the data and a framework to transmit said data. A common semantic structure is needed to ensure ease-of-maintenance and compatibility across all parts of the grid.
2. **Storage**: Collected information must be held in a data storage center to be organized and processed.
3. **Processing**: The collected data typically needs to be processed before being used in applications. Therefore, there needs to be a component that handles this processing. Usually, the processing is performed in the fog or cloud layer
4. **Visualization**: The data needs to be displayed to stakeholders to allow them to understand how much power they are using, equipment conditions and more.

Before performing data collection, the types of data should be determined. Examples include load demand, power produced, meteorological data and more. Then the data collection devices can be determined. For example, an AI may need a temperature, wind speed, and humidity. Reference [66] used a CR1000 acquisition card to collect weather and climate data. Technologies such as sensors and smart meters belong in the edge layer of IoT computing.

These devices would then be connected to a fog device which is connected to a cloud data center. Typically, the fog device would connect to the data center using the Internet. For example, a Wi-Fi gateway device can be used. This gateway serves as a link to the internet. For example, in reference [67] an Arduino microcontroller was used to monitor the incoming power from the inverter and grid.

An ESP8266 microcontroller served as the Wi-Fi gateway. The Arduino microcontroller transmitted this data to an ESP8266 microcontroller, which forwarded the data to a cloud server. The Wi-Fi gateway fits into the fog layer of IoT computing. Energy resources can cover wide areas; therefore, devices may communicate via wireless protocols. Thus, a local area network (LAN) can be used to transmit data from the edge devices to a fog device. The data communication protocols should be low power to minimize the impact on the site's energy consumption. If the components are powered by batteries, it is vital to ensure they can last a long time before needing replacement.

Table 5 gives a comparison of wireless communication protocols that can be used for LAN. Typically, an electronic module is needed to enable these wireless communications. An example is the Texas Instruments CC2530 chip which can enable ZigBee communications [78]. Once the data is collected it needs to be stored and processed.

Table 5: A comparison of Zigbee, Bluetooth Low Energy, and Wi-Fi 5.

Protocol	**Zigbee**	**Bluetooth Low Energy (BLE)**	**Wi-Fi 5**
Maximum Data Rate (Megabytes/second)	0.250 [68]	2 [69]	1300 [70]
Communication Range (Meters)	0-50 [71]	0-78 [72]	0-80 [73]
Transmit Power Consumption (Milliwatts)	29.7 – 247.5 [74], [75]	4.44 – 71.46 [76]	1266 - 4336 [77]**

*The power usage will vary depending on the electronic module and packet size

** The power usage was measured from an Access Point. Furthermore, it will vary depending on the electronic module, packet size, frequency, and signal strength in dBm.

All the collected information then needs to be sent a data center for storage and processing. For large volumes of data, parallel processing and distributed computing is recommended [79][80][81]. A popular system is the Hadoop Distributed File System (HDFS). HDFS allows computations to be spread across multiple computers, thus enabling parallel processing. Parallel processing allows multiple requests to be handled simultaneously, thus reducing the time it takes to get the needed information. Gupta et. al built a distributed computing system using HDFS [82]. Their system processed data 20 times faster than a monolithic database. Their system was designed to process data from multiple energy management systems across a region. Now with an understanding of each IoT layer, it is important to see how they are used in the literature.

As discussed in Section 2.1, intelligent algorithms can be used for energy trading or optimal energy scheduling. The authors of reference [83] implement a two-tier computing approach for demand side response. The environment is assumed to have multiple separate microgrids. The first tier is at the edge. Each microgrid has its own "Edge Cloud" that aggregates the production and consumption of each home. The Edge Cloud has an algorithm that schedules optimal energy usage for every load in the microgrid considering the energy cost. Then, each Edge Cloud will forward the energy usage schedule, total microgrid energy production, and storage to the next tier called the "Core Cloud". The Core Cloud then optimizes energy usage of each microgrid. The objective is to reduce the peak-to-average ratio for the entire grid network. In this example the devices collecting the data from the loads are in the edge layer. While the Edge Cloud is actually in the fog layer. Lastly, the Core Cloud would be in the cloud layer.

In reference [84], they also have a similar two-tier approach for energy optimization. In this paper, each microgrid can communicate with one other. The optimization in the 1st tier is performed by cooperation between individual agents within the microgrid. Each AI agent represents a load or energy resource. Furthermore, these AI agents would be in the edge layer. At the 2nd tier, each microgrid is regarded as a single agent and they attempt to trade energy so that each microgrid has the same approximate marginal cost. The microgrid agents would be in the fog layer. While the aggregated data would be in cloud layer.

It can be seen how taking advantage of the computing paradigms also allows for a unique approach to energy optimization by splitting the calculations to different stages. The multi-tier approach reduces

computational burden on each system. The systems in each tier only needs to consider the information directly sent to it. Furthermore, the separation of levels also allows for utilities to be able monitor and collect energy usage from less entities.

Figure 5 shows the high-level architecture of an EMS. Firstly, energy usage, consumption and other physical data need to be collected from sensors and other equipment. The collected information should be sent to a fog device, a Smart Energy Controller (SEC), with an AI to optimize energy usage, trade energy, and perform demand response. The SEC serves as a gateway for the microgrid to communicate with other microgrids or the cloud data center. The controller can also serve as the internet gateway since it will aggregate the data and contain the outputs from the AI. The central controller can be a microcontroller such as a Raspberry Pi, an NVIDIA Jetson Nano, a Beaglebone and more. The advantages of a microcontroller over a desktop computer are its small size and cheaper pricing. For example, the Raspberry Pi 4B measures 85 mm x 56 mm and can be bought off the shelf for 35 USD [85][86]. Each site sends the processed information to a cloud server. The cloud server stores the data. Furthermore, it will be connected to multiple sites. However, there will not be one cloud server, but rather multiple servers for a specific region or area.

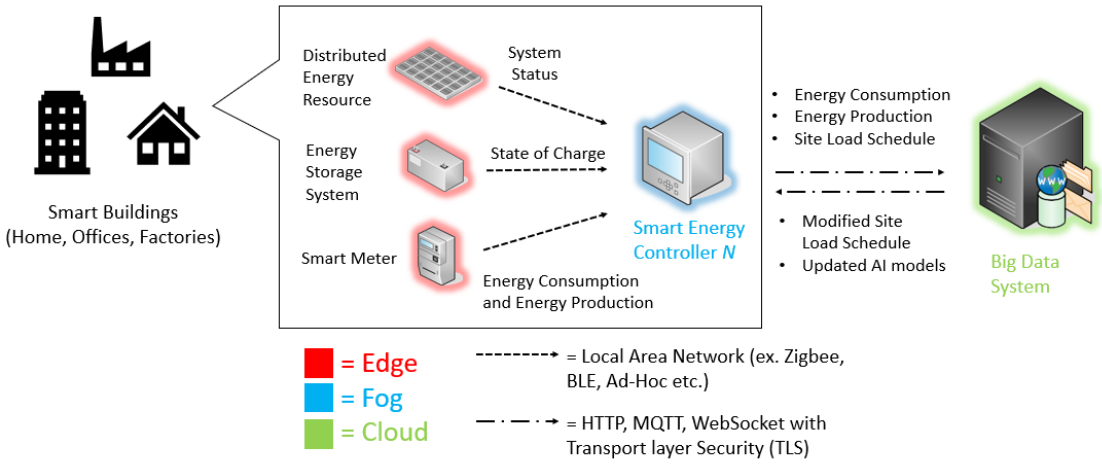


Figure 5: A high level diagram depicting the IoT architecture for an energy management system.

During the literature review it was observed that there was no semantic IoT framework, and there was little discussion on the actual implementation of the IoT computing infrastructure. To advance research in this field, an IoT framework needs to be discussed. Therefore, Table 6 presents some IoT platforms in the industry that can be used for edge, fog, and cloud computing. The scale and sensitivity of data in a SEN means that the IoT platforms should be hosted by reputable companies with many resources. Google Cloud, Amazon Web Services, and Microsoft Azure platforms offer many services. Table 6 summarizes the services of each platform and their purpose.

Table 6: Cloud services for IoT computing and data mining

Service	*Google Cloud*	*AWS*	*Azure*
Data Aggregation and Streaming	Google IoT Core	Amazon Managed Streaming for Apache Kafka, AWS Glue	Azure Stream Analytics
Device Management	Google IoT Core	AWS IoT Core	Azure IoT Edge
Data storage and Processing	Google BigQuery	Amazon S3, Amazon Elastic Kubernetes Service	Azure SQL
Applications	Google App Engine	Amazon Elastic Kubernetes Service	Azure App Service

A key part of any IoT platform is a customer facing application. This is needed to communicate key data to stakeholders. For example, utilities need to know how much energy may be generated and consumed in the next hour so they can ensure there is a sufficient energy supply available. While customers may want to see which loads are being shifted for demand response. Each application must be designed differently for each stakeholder. From the literature in this section, it is observed how various digital technologies can aid in energy management. IoT platforms can be used to connect stakeholders and devices at a site. Combining an IoT platform with AI and big data enables the creation of an energy management system.

2.5 Summary

In this chapter, a review of AI, big data, and digital technologies is presented. Researchers should utilize these concepts when developing an EMS. The role of the AI is to act as the site manager, ensuring optimal operation and facilitating interactions with other systems and stakeholders. Furthermore, it helps to decentralize grid operation through management of individual sites. Utilizing the dimensions of big data and data mining can help to train AIs. While usage of IoT computing provides an infrastructure for dataflow and interaction between entities and systems.

Artificial intelligence plays a major role in automating energy management. It is composed of multiple intelligent algorithms that enable energy prediction, energy scheduling, demand response, and energy trading. The popular algorithms for forecasting are Neural Networks and Support Vector Machines. They are used to forecast energy load and energy production. This information is important during energy scheduling and energy trading because it allows the system to allocate enough energy for demand or profit off excess energy. During training, these models have been combined with optimization algorithms like Cuckoo, Dragonfly, Ant Bee Colony, Genetic algorithms, and Particle Swarm to increase their performance. The popular choices for energy scheduling and energy trading are Fuzzy Logic and Q-learning. Fuzzy Logic acts based on the state of the microgrid and can be guided towards optimal choices through its action-reward mechanism. Similarly, Q learning adjusts its choices based on feedback from the environment. Q Learning is useful because it does not require a priori information on the environment, thus it can be used for different microgrids. In all cases, the research is heading towards hybrid and multi-stage models to enhance the performance.

To utilize the full potential of these algorithms, data mining is needed. Data mining can be used for enhancing demand response, and energy related predictions. Data mining for demand response, it can be used to periodically classify shiftable loads. While, for energy predictions, data mining is used to select the best input data. Thus, in the context of microgrid energy management, data mining is used to enhance the performance of AI.

To effectively implement AI, researchers need to consider the underlying digital technology. Designing a digital infrastructure using edge, fog, and cloud computing can enhance the flow of data and operation of the entire system. The edge is composed of sensors and data acquisition devices

which are then connected to a fog device. In the context of EMS, the fog device is a central controller that is programmed with an AI and can perform demand response and basic data mining. Furthermore, the EMS controller serves as a gateway to the wider internet and the cloud layer. The cloud layer is where bulk data mining is performed on data from multiple energy management systems. To handle such volumes of data, the cloud layer can utilize distributed and parallel computing to split up the work. Finally, all the data should be available to stakeholders to help them make data driven decisions. IoT platforms such as Google Cloud, AWS, and Azure can be used as common semantic frameworks for data processing. AI, big data, and digital technologies are key research fields for smart energy management and therefore their co-development and integrability are key to the future of the grid. Synthesizing the literature provides an architecture and description in Figure 6 and Table 7.

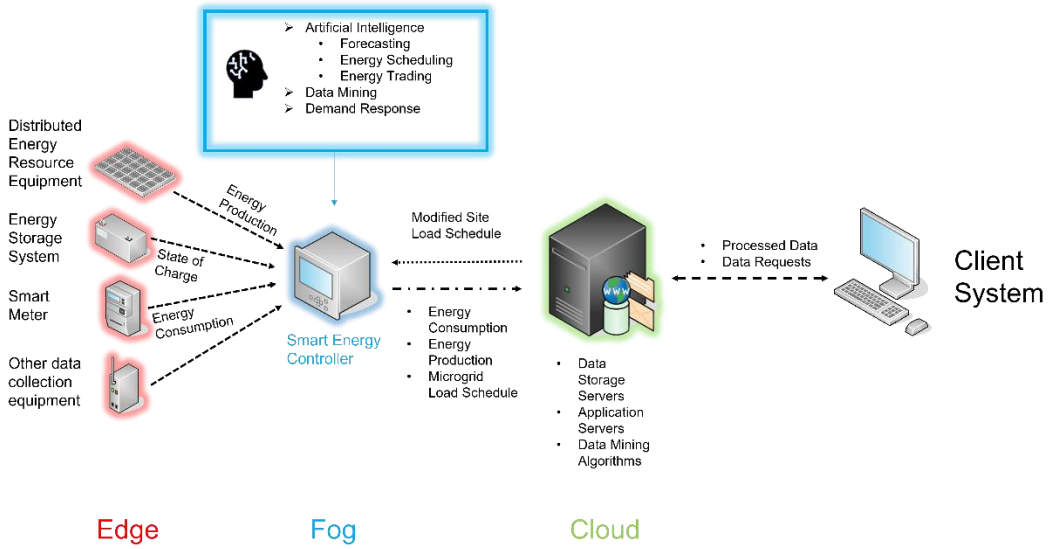


Figure 6: A diagram showing an example architecture for an energy management system.

Table 7: Supplement to Figure 6. Describes the digital entities, data, and processes in each layer

	Digital Devices / Entities	**Data**	**Processes**
Edge	• Sensors (Temperature,	• Physical *(temperature [°C],*	N/A

	irradiance, anemometer etc.) • Energy Storage System (Battery, Flywheel, Thermal Storage etc.) • Smart Meter • Distributed Energy Resource Equipment (smart inverters • Smart Inverters	*Irradiance [W/m^2], Wind speed [m/s])* • Power related *(Voltage [Volts], Current [Amperes], Power Consumption [kW/h], Power Generation [kW/h])* • System Status *(Fault, OK etc.)*	
Fog	• Smart Energy Management Controller • Internet Gateway	• Refined Edge Data *(Physical Data and Power related)* • Generated Data *(Energy Schedule – charge / discharge of energy resources, load shifting)* • Equipment Status for each edge device and plant	• Smart Energy Management Controller contains: ○ Artificial Intelligence ○ Data Mining Algorithm(s) to preprocess the data for the AI ○ Demand response may be performed by the AI
Cloud	• Data Storage Servers • Client Application Servers	• Data Storage Servers contains the following data from each connected EMS: ○ Refined Edge Data *(Physical Data and Power related)* ○ Initial and modified energy schedule ○ Equipment Status	• The cloud computing services will: ○ Perform data mining to reveal more relationships in data or prepare them for the client application ○ Host the data storage and application servers

As observed, the Energy Management System is comprised of an edge, fog, and cloud layer. The edge layer is made up of the sensors and actuators. This layer provides data to the fog layer or cloud

layer and can directly influence the behavior of the plant. The fog layer consists only of the microgrid energy management controller. This controller acts as the representative of the site. It communicates and controls the edge layer devices, while aggregating and uploading data to the cloud layer. The idea is to reduce the volume of data headed to the cloud by pre-processing it and having it sent by fewer sources. The cloud is where the data is stored and further processed to provide data analytics to the user via the client interface. This architecture fulfills the goals of a SEN by automating energy systems with a microgrid controller, providing bi-directional real time communication via the IoT layers, automation through an AI that can control the energy systems, and promote adoption of renewable technologies by providing a system to monitor energy. Therefore, this architecture can provide a solid foundation to develop and explore an EMS for rural or off-grid areas.

There is already so much content covering the key AI and data mining processes for an EMS. However, it is important to build a hardware and software infrastructure for these processes to work. Thus, the next chapter will detail the specific technologies and techniques that can be used to build an infrastructure for an EMS.

Chapter 3:
Methodology and Materials

3.1 High Level Design

An energy management system in this thesis is developed to achieve the following objectives:

i. Enable monitoring, energy production forecasting and control of site energy resources.
ii. Maintain bi-directional communication with the wider grid and local energy resources and loads; and
iii. Store, process and communicate relevant data required by stakeholders

Based on the above objectives, an outline of the architecture is developed. The energy management system is designed for an off-grid residential solar PV system. The primary requirements are as follows:

1. Can retrofit the EMS on existing solar PV energy systems
2. Can operate without equipment from a specific solar PV system manufacturer
3. Can connect to the Internet
4. Can run a Neural Network
5. Can store information digitally
6. Can allow or block the flow of electricity within the solar PV energy system
7. Can measure the power flow within the solar PV energy system
8. Can access information from the EMS via a central interface

Figure 7 shows the key components and electrical connections for a solar PV system.

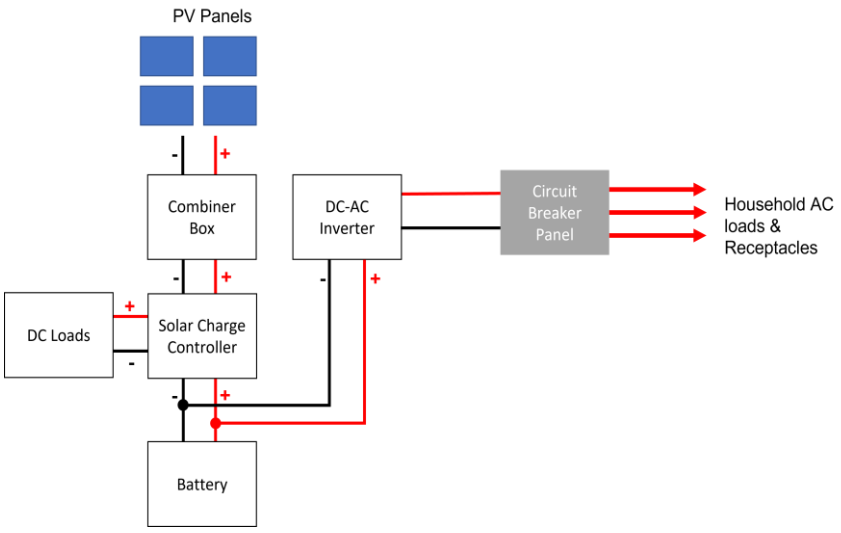


Figure 7: High level diagram showing the key components and connections in a solar PV system. Inspired from [87]**.**

A detailed energy management architecture below is developed from Figure 6 in Chapter 2.

1. User Application: This software takes the information from the data storage and presents it in a user-friendly format. This application also allows the users to have some manual control over the system.
2. Plant: The site with the distributed generators and the load.
3. Monitoring and Control Platform (MCP): This component enables control and performs automated monitoring of energy and equipment on the site. This system is a part of the edge and fog layers, including.
    a. Actuators: This component consists of hardware and software that is used to control the physical state of the plant. In this system, the actuators control the flow of electricity.
    b. Controller Software: This is the component that automates the sensor data collection and enables direct control over the actuators.
    c. Controller Hardware: The controller software is loaded onto the hardware. The hardware also enables connectivity to the rest of the MCP.
    d. Sensors: The sensors are the devices that monitor physical properties. Examples include the energy flow, the ambient temperature, the windspeed etc.

4. Data Management Infrastructure:
    a. Local communication network: This network allows the controller to exchange information with the actuators and sensors.
    b. External communication network: This network allows the controller to transmit information to components that is not reachable within the local communication network.
    c. Data Center: A component stores the information obtained from the MCP. The stored information includes site energy usage, production, equipment status, sensor data. It is also used as a data processing center to present information in a human friendly manner. This component resides within the cloud layer

Now that architecture has been outlined, the remainder of the section will discuss the properties of each component in the energy management system. The first component discussed will be the user application. The displayed data and controls should be labelled with numbers or words. Furthermore, the labels should be accurate. For example, a graph titled 'Predicted Daily Energy Production' should have the Y-axis units be in Watts and the X-axis units should represent the time; in 24-hour clock time or 12-hour clock time. This property is essential because the target users are personnel in the residential sector or utilities. The features included in the user application are control options (e.g. Turning the battery mote on or off), Alarms (e.g. Equipment problems), forecasted power production, and system logs (e.g. The changes in equipment status).

The MCP requires control mechanisms, which allow users or an AI to control the actuators. Another important part of the MCP is the data collection system. Without it, the AI and users cannot make decisions and are blind to what is happening in the system. The system should be able to measure the energy currently being produced. Also, energy forecasting requires specific physical data. Table 4 in Section 2.1 presents examples of physical data required to predict energy production for solar PV and wind turbines. Lastly, information on the equipment status needs to be collected. If a piece of equipment malfunctions, then the system can trigger alarms and inform the user to review and address the issues.

The MCP shall be separated from the energy resource system. A separate MCP requires additional costs and components. However, since it is isolated from the energy resource components, thus problem identification is simplified. For example, if the MCP reports zero power being generated and

no internal errors, then it is possible that the energy resource may have an issue. Furthermore, a separate system allows non-smart systems to be retrofitted without having to replace components.

Easier integration requires that the smart energy management system be hardware agnostic. The hardware and software that comprise the smart energy management system will be part of a closed system. However, the EMS should not rely on other external components within the plant to operate. This allows users with various equipment to be able to use the system.

This EMS is also designed to be relatively low cost for replacing individual components. Since this EMS is designed for a Solar PV system, the cost is within the range of the price of the solar PV systems in Table 1 in Chapter 1.

For the EMS to function effectively, real time operation is required. In this case, real time means within seconds. This includes real time communication within the MCP and between the MCP, the data center, and the user application. Issues shall be reported as soon as possible to avoid downtimes, accidents, and further malfunctions.

Now with a more thorough understanding of each component in the EMS, an architecture can be developed. The EMS is designed for a residential solar PV system. Figure 8 show a high-level overview of the components in the EMS. Note that the MCP configuration is made with respect to the experimental setup.

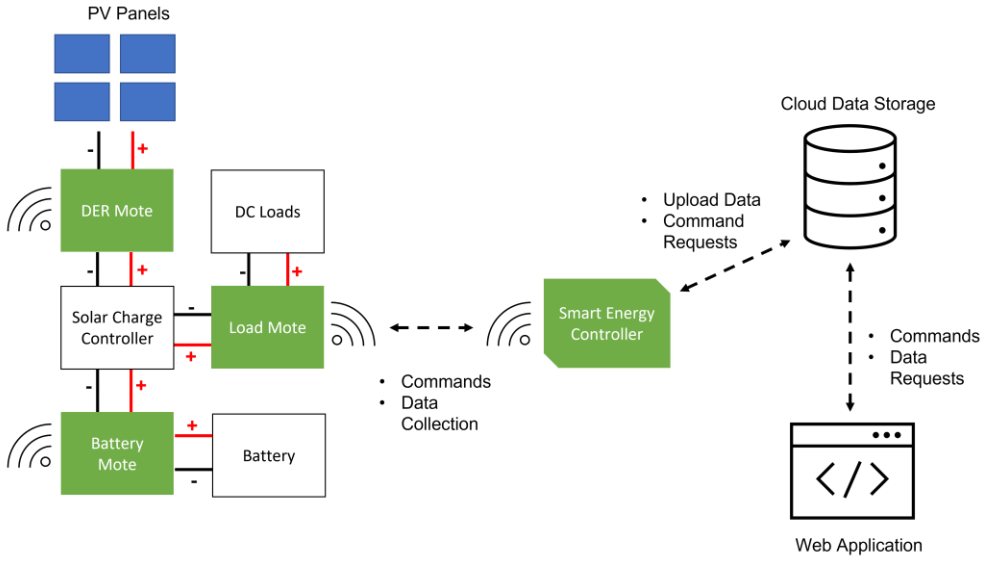


Figure 8: High level diagram showing the EMS in this thesis. The key components and flow of information are outlined.

The Smart Energy Controller wirelessly communicates with the Motes. The Motes can measure the amount of energy (kW) flowing through them, and block or allow the flow of energy as needed. The controller itself contains the AI, the control software, and the data collection software. All information is aggregated by the controller and is sent upstream via the internet to the data center. The data center stores and processes data from the MCP. Then the user application can be used to view the processed data and issue commands. The specific design choices for each component are discussed in the following sections. Before discussing the MCP, it is important to understand the data management infrastructure for the EMS.

3.2 Data Management Infrastructure

The data management infrastructure encompasses data storage and communication within and between the edge, fog, and cloud layers. Using some of the dimensions of big data, described in Section 2.3, can help guide the design of the infrastructure. Firstly, there is the Volume dimension. The data infrastructure needs to handle sensor readings and log data. Furthermore, the system should scale to manage data for multiple energy management systems. Therefore, a cloud service provider managed by a large, reputable business is ideal.

In this thesis, Amazon Web Services (AWS) is used to create the cloud layer in the data management infrastructure. As discussed in Section 2.4, there are other services that provide similar offerings. Any one of them can be used, but AWS is chosen due to its familiarity and easy to use. To minimize compatibility errors, all data is managed by AWS. Table 8 outlines the services used from AWS. Using a provider like AWS also considers the Velocity dimension. Their infrastructure is used by multiple large businesses worldwide [88]. Thus, it has been proved that their system can handle the frequency and volume of the incoming data.

Table 8: The services used from AWS

Service Name	*Description*
AWS Amplify	A service used to host the client application.

AWS Lambda	A service used to create a function that can retrieve and preprocess data from other AWS services. This function is called a 'Lambda function'. The function is invoked via a link generated by the API gateway service.
API Gateway	This service is used to create and manage Application Program Interfaces (APIs) to access other AWS services
DynamoDB	A NoSQL database used to store records.
IoT Core	Used to connect to and manage edge devices or other clients
Cognito	A service that controls permissions to access web applications and permissions for web applications access other AWS services.

Although there are several services available to construct the data management infrastructure. AWS Amplify allows the creation and hosting of web applications without needing to provision a web server. This accelerates development because now the user only needs to focus on the web application itself rather than the infrastructure to host it. AWS Lambda allows users to program functions that can access other AWS services. In this case, the Lambda functions are programmed in Python and are designed to retrieve and process data from DynamoDB. The web application can call the function through a Uniform Resource Locator (URL) link created by the API Gateway. This allows the web application access data from DynamoDB and display it. The type of data stored in DynamoDB includes control logs, alarm logs, and measured power from each Mote. The details of the data will be further discussed in the following paragraphs.

DynamoDB is a NoSQL database. NoSQL uses "tables" to store data entries. Each data entry is a row, and each column is an attribute of the data. NoSQL does not require the same columns for all entries. Missing or empty data columns does not affect the operation of the database. DynamoDB database partially covers the Variety dimension. DynamoDB can hold structured or semi-structured data. The only limitation is that it cannot store unstructured data. However, this is acceptable because in the proposed EMS, unstructured data is not used. DynamoDB also covers the Volatility dimension. Records can automatically be deleted after a certain period. For example, DynamoDB can be configured to store records only for a month.

There are two tables: control logs and alarm logs. In DynamoDB each table must have a primary key, which is a column in the table. This helps with querying the correct data. Ideally, the primary key is unique for each client. Therefore, System ID is selected. Where the System ID is a numeric string representing a specific EMS installation. Then there is the secondary index, which is also used to help with sorting. The secondary index represents the timestamp, which is recorded in epoch time. Epoch time is the time in seconds since January 1st, 1970, at midnight in Coordinated Universal Time (UTC). Thus, every entry has a unique secondary index. The newer the entry, the larger the epoch time number. When querying data, the secondary index can be used to sort the entries in order of newest to oldest. Both primary and secondary indices are vital when retrieving the data via the Lambda function. Lastly, each table has other columns specific to its purpose. The control and alarm logs ta‘le will ’ave a 'log data' column and measured powe‘ will have ’he 'sensor data' column. It is important to note that more columns can be added at any time. Figure 9 shows a sample of a data entry for the control logs table. While DynamoDB can store data, there still needs to be a system to transmit and receive data.



Figure 9: A sample control log entry in AWS. The system is a string representing the EMS system ID. The epoch timestamp is the epoch time of when the action was recorded. The “ctrl_log_data” field is a JSON object.

AWS IoT Core is used to connect the web application and SEC together. It fulfills this task by acting as a Message Queue Telemetry Transport (MQTT) broker. It handles the messages sent to it, via MQTT, and route them to their correct destination. MQTT is a resource light messaging system that allows for bi-directional communication [89]. The web app and SEC are both clients to the broker and can subscribe to topics or publish to them. Topics are like a communications channel for a specific subject. A client that subscribes to a topic client will receive new messages sent to that topic. Whereas publishing to a topic is sending a message to that topic. In the EMS, the topics are for commands, sensor data, acknowledgements, and predictions. Figure 10 visualizes the idea of subscribing and publishing to a topic.

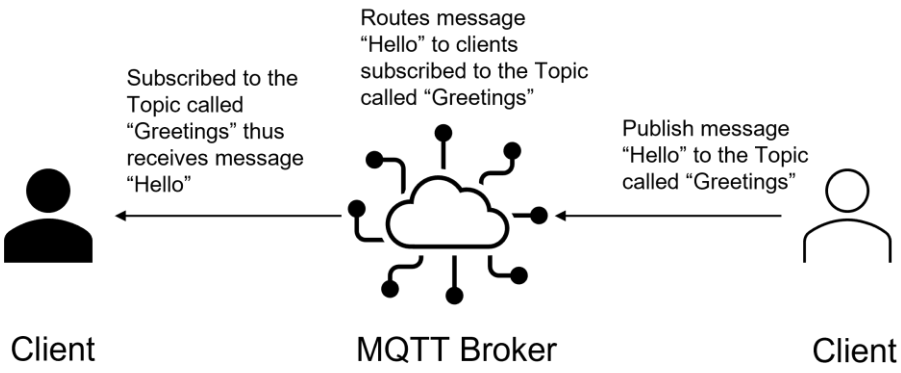


Figure 10: Diagram summarizing the interaction between clients and the broker.

IoT Core’s MQTT Broker was chosen because it provides encrypted communications, via TLS v1.2. Furthermore, MQTT is suitable for fog and edge devices where minimal power consumption is desired. The MQTT system is used to issue commands, receive acknowledgements, report sensor readings, and update the control logs table and alarm logs table. AWS Cognito is used to allow the web application to publish MQTT messages and subscribe to MQTT topics. Figure 11 visualizes the interaction between AWS services within the cloud layer. Now with an understanding of the cloud layer, it is possible to move onto to the fog and edge layer communications.

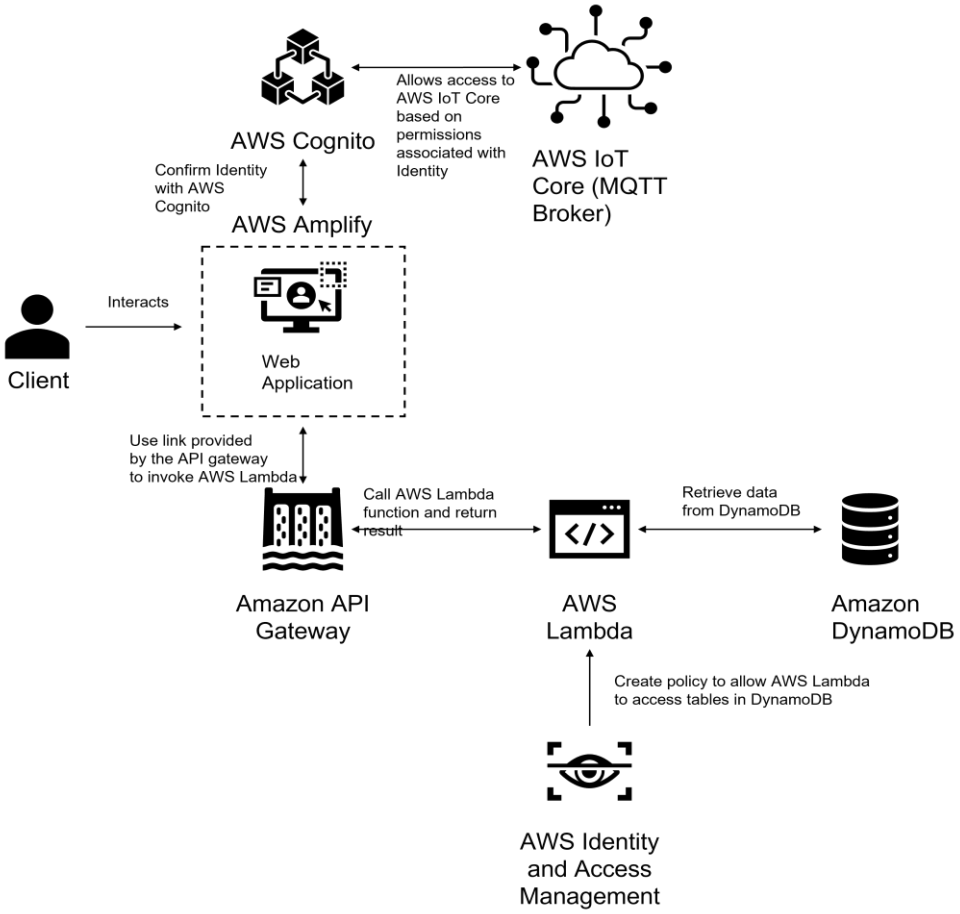


Figure 11: Diagram summarizing the AWS services used in the cloud layer and their interaction.

The edge devices are the motes, whereas the fog device is the Smart Energy Controller. As discussed in Section 3.1, these devices reside in the site. The controller and the motes communicate wirelessly. Wired communication is not flexible for different energy systems or sites due to the variability of each site and cost of wires with respect to length. Furthermore, wires can be physically damaged. Therefore, the preferred way for on-site communication is through a wireless network. As reviewed in Section 2.4, there are a few notable options; Wi-Fi, Zigbee, and Bluetooth Low Energy (BLE). The protocols are compared and presented in Table 5.

As observed, Wi-Fi consumes 426 mW, Zigbee consumes 100 mW, and BLE consumes 10 mW, The high-power consumption of Wi-Fi makes it unattractive for device communications. While BLE

boasts a better data rate and power usage, Zigbee is still used as the site communication protocol. The better data rate does not make a large difference because, the messages are simply text, thus the 1 Mb size will never be needed. While ZigBee generally uses more power, it is used in many IoT applications as well. There are over 4000 products that are Zigbee certified, with 530 products certified in 2021. Almost 17% of these products are used for smart energy applications, while around 3% are used for green power applications [90]. Furthermore, the power usage also depends on the hardware because each chip consumes a different amount of power. For the SEC to communicate with AWS, it would use Wi-Fi because communications with AWS must be done via the Internet. In places where there is no internet service provider, a cellular modem can be used. Communications with AWS remains the same, regardless if a landline or cellular modem is used. For fog and edge device communications, Zigbee is used. Wi-Fi is used for fog to cloud communications. Now that the data management has been discussed, it is possible to start exploring the operation and construction of the Smart Energy Controller.

3.3 Smart Energy Controller

The Smart Energy Controller (SEC) is responsible for hosting the AI, controlling the flow of energy in the energy system, aggregating data from the Motes, and uploading sensor data and logs to the cloud.

This section provides the underlying software and hardware infrastructure needed to achieve these objectives. It is important to understand how an intelligent algorithm can be implemented in software. Therefore, a neural network was developed to predict the power production for a solar PV energy system at the National. The developed neural network is used to help build a NN for predicting energy usage. This section also provides a hardware architecture that enables control and communications for the MCP. A software framework is provided to further understand how energy control, energy prediction, and data collection can be implemented for real time operation.

3.3.1 Development of a Neural Network for Predicting Energy Production

As reviewed in Section 2.1, either a Support Vector Machine (SVM) or a Neural Network (NN) can be used to predict energy production. In this thesis, a Neural Network was chosen due to the vast software support and documentation.

The desired output of the NN is the hourly power production from the solar PV system. The hourly interval was chosen because it lines up with hourly weather forecasts. Table 3 and Table 4 in Section 2.1 gives a list of input features used in the literature. From the literature, the chosen input features are Humidity (%), Ambient Temperature (°C), Wind Speed (m/s), and Solar Irradiance on the Plane of the Array (W/m^2).

To train this NN, a dataset from the National Institute of Standards and Technology (NIST) campus in Gaithersburg, Maryland, USA was chosen. The campus features four PV testbeds and a rooftop weather station. Due to the different specifications of each testbed, only one testbed dataset can be used. In this thesis, the testbed chosen was a ground mount solar PV array. The array is rated to produce 271 kW of DC power. Each array module is made from monocrystalline silicon and has a rated efficiency of 14.4 %. Both the rooftop weather station and ground mount array aggregate data every minute. The datasets used are from 2016-2018 [91]. The humidity data comes from the weather station data, while the rest comes from the ground mount array. Unfortunately, the collected data cannot be used right away.

Hourly measurements are needed for the input features and output. Furthermore, the data is saved as comma separated files (CSV) files for each day. There are 2922 total files. The ground array data includes 100 different measurements; however, only 4 of them are needed; Ambient Temperature (°C), Wind Speed (m/s), Solar Irradiance on the Plane of the Array (W/m^2), and Inverter Power Output (W). Note that the output power from the inverter was used because, it gives a more accurate representation of power after accounting for the losses from the other components. The weather station CSV files contain 49 measurements, but only the Humidity (%) is needed. Figure 12 shows the pseudo code of program used to process the data. The 2018 dataset is used for training. While the 2017 dataset is used for validation. Lastly, the 2016 dataset is used for testing. The NN never learns from the validation and testing data. However, the results from the validation data set are used to manually tune the NN neurons and layers. While the testing data set is exclusively used to see the performance of the NN. Each dataset was processed and has 8760 entries. With an understanding of the data itself, it is now to discuss the development of the NN.

Algorithm 1: The Python program used to process the training data for the neural network

Input: The CSV files containing the measurements from solar PV system and weather station
Output: A processed CSV file for each year containing the input features and associated output labels.

```
for each year do
    for PV system CSV file do
        Concatenate the average windspeed, solar irradiance, ambient temperature, and
        inverter power output data from the files to RAW_DATA data frame
    end for
    for weather station CSV file do
        Concatenate the Humidity data from the files to WEATHER_DATA data frame
    end for
    Compare the length of the data frames and drop the number
    of rows to match the one with the smaller amount
    Take the Humidity data from the WEATHER_DATA frame and
    insert it into RAW_DATA data frame
    for each feature data in RAW_DATA do
        for every 60 elements do
            Calculate the average
            Save the average to TEMP_LIST
        end for
        Save TEMP_LIST as a new column (with the feature name)
        into HOURLY_DATA data frame
        Clear the TEMP_LIST
    end for
    Export the HOURLY_DATA data frame to a CSV file
end for
```

Figure 12: The pseudo code of how the ANN training data was processed.

The NN is programmed in Python using the TensorFlow library. The architecture of the NN is 5 layers with 25,50,25,15, and 1 neuron(s) respectively. Note that experiments with different configurations were tried, but this configuration yielded one of the best results. Also, the output layer must always have one neuron since we only have one output. Layers 1-4 used Rectified Linear Units (ReLU) as the activation function, while the output layer used linear activation. ReLU was the default choice for activation functions and is good for dealing with non-linearities in the data; the function is defined as $f(x) = max(0,x)$ [92]. Also, the linear activation does not change the output and just sums the weighted outputs and multiplies the result by 1; this is desired since the true output is required. The Neural Network was trained with 10 Epochs with 24 samples per batch. The training and validation were performed within one day. It was done on an NVIDIA Jetson Nano 2 GB edition; it features a quad core ARM A57 at 1.43 GHz and 2 GB of RAM. Now that the Neural Network has been explained, the rest of the SEC architecture can be explored.

3.3.2 Hardware

The SEC shall have many different functions; thus, it must be programmable. Ideally, it needs to have a low power consumption, a low price point, be commercially available, be capable of communicating with other digital entities, and be capable of running a Neural Network. There are no strict requirements on processing performance; if it can perform the mentioned features, it will likely have enough processing power to support the programs. Therefore, a microcontroller is well suited to fulfill these conditions. Very simply, a microcontroller contains a programmable chip with memory and various Input/Output (I/O) capabilities [93]. There are many microcontrollers on the market, Table 9 gives a list of potential alternatives.

Table 9: Some microcontrollers that can be used for IoT applications [94]–[97]

	ESP32	***Raspberry Pi Zero W***	***Raspberry Pi 4B (2 GB Edition)***	***Jetson Nano (2 GB Edition)***
OS	N/A	Raspbian	Raspbian	Ubuntu 18.04 based OS
Connectivity	34 × programmable GPIOs (4 × SPI 2 × I2S 2 × I2C 3 × UART), 12-bit SAR ADC, 2 × 8-bit DAC, 10 × touch sensors, Motor PWM,	40 pin GPIO header	Gigabit Ethernet, 2x USB 3.0, 2x USB 2.0, 40 pin GPIO header	40-pin header (GPIO, I2C, I2S, SPI, UART) 12-pin header (Power and related signals, UART) 4-pin fan header 1 x4 (PCIe Gen2)

	LED PWM up to 16 channels, Hall sensor			
Wireless Communications	2.4 Ghz 802.11 b/g/n, Bluetooth 4.2, Bluetooth Low Energy (BLE)	2.4GHz 802.11n wireless LAN, Bluetooth Classic 4.1, BLE	2.4GHz 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE	Gigabit Ethernet
GPU	N/A	N/A	Broadcom VideoCore VI	128 CUDA Core Maxwell Architecture
Primary Programming Languages	C, MicroPython	C, C++, Python	C, C++, Python	C, C++, Python
Random Access Memory (RAM)	520 kB	512 MB	2 GB	2 GB
Price (CAD)	$ 6.37 [98]	$13.37 [99]	$ 64.11 [100]	$103.29 [101]

Note: The references for the price are indicated next to the number.

For this thesis, the Jetson Nano 2 GB version was chosen. The Jetson Nano was chosen because it features a Graphical Processing Unit (GPU) with Compute Unified Device Architecture (CUDA) cores. A GPU is especially useful for improve Machine Learning (ML) development and performance [102][103]. As discussed in Section 2.1, Neural Networks, a type of ML algorithm, can be used to predict energy production or consumption. Furthermore, CUDA cores are special processing units developed by NVIDIA that enable high speed parallel computations. NVIDIA CUDA core GPUs

have been used for AI-based video processing [104], high performance computing centers [105], and many more computationally intensive tasks. Furthermore, the Jetson Nano has more RAM than the Raspberry Pi W and ESP32. The closest competitor is the Raspberry Pi 4B, however the Jetson Nano has a better GPU. The only disadvantage is the price, which is much higher than the other devices.

Since one of the objectives of the SEC is to predict energy production, it would be beneficial to have device that can improve the performance of such algorithms. Furthermore, if more intelligent algorithms are added then this benefit becomes more valuable. It is important to note that the Jetson Nano does not feature an onboard Wi-Fi chip, however there are plenty of USB Wi-Fi dongles that can be used. In this case, the Wi-Fi adapter used a RTL8812BU chip for Wi-Fi communication.

As discussed in Section 3.2, the Motes and the Smart Controller communicate via Zigbee. However, the Jetson Nano does not feature Zigbee communication, so a separate module is needed. The Motes will need Zigbee communication capabilities and programmability for basic logic. The suitable approach would be to have Zigbee module with programmability. This design choice enables the Jetson Nano to have ZigBee communications, while serving as a platform for the Motes. Like the microcontroller, the Zigbee module needs to have low power consumption, a low price point, and be commercially available. After searching, the most suitable option is the Digi Xbee 3 (See Appendix A for the specifications of the Digi Xbee 3).

The notable features are serial data interfaces, ADC pins, GPIO pins, and the MicroPython stack [106]. The serial data interfaces are necessary for the module to communicate with the Jetson Nano and certain sensors. The Xbee 3 module connected to the Jetson Nano uses UART for communication. While the ADC pins enable various analog sensors to be attached. The GPIO pins allow configuration and can be used to trigger peripherals. Lastly, the MicroPython stack enables programming of basic logic into the Xbee 3 modules. For future reference, the Xbee 3 module attached to the Jetson Nano will be designated as JNXB3, while a module used for a Mote will be called MXB3. Now that the hardware components have been defined, the SEC architecture can be defined. Figure 13 shows the high-level architecture of the SEC.

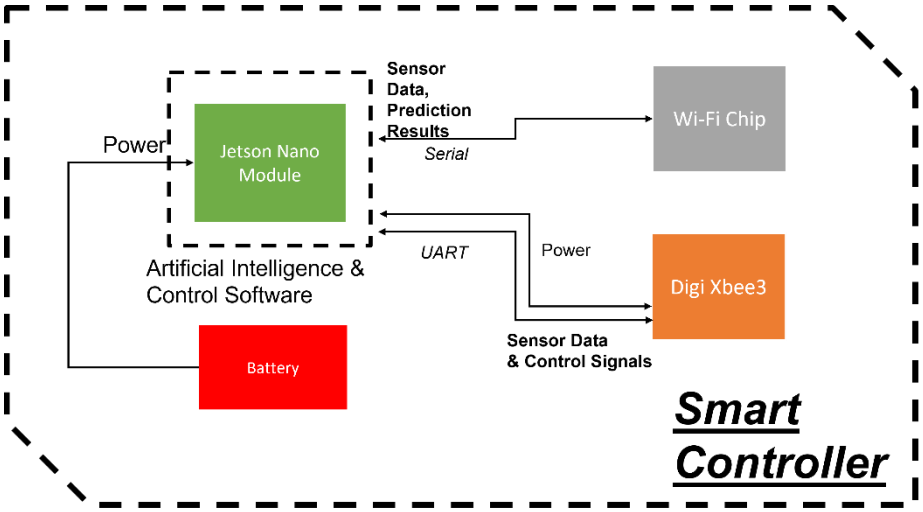


Figure 13: The Smart Energy Controller Architecture.

3.3.3 Software

In Section 4.3 the SEC responsibilities are discussed. For easier referencing, the responsibilities of the SEC are rearranged as follows:

1) Predicting the power production and power usage
2) Controlling the flow of energy through the entire solar PV energy system
3) Aggregating data from the Motes
4) Uploading aggregated data and system logs to the cloud Layer.

To accomplish these responsibilities, they must be executed through software logic. However, there are many programming languages and libraries that can be used to accomplish these responsibilities. Python is currently used to program the NN used for power production forecasts; this covers responsibility 1). Examining responsibility 4), the controller must be able to upload data to AWS. AWS has libraries for Python. Specifically, the 'boto3' module is used to upload data to DynamoDB. Furthermore, the controller must connect to IoT Core; AWS has made software development kits for Python and C++ [107]. Thus, considering these facts, responsibility 4) should be accomplished through Python as well. Responsibilities 2) and 3) do not require external services or software to be accomplished. Since responsibilities 1) and 4) already use Python, it would make sense to also use Python for responsibilities 2) and 3). Therefore, the entire software logic on the SEC has been programmed using Python.

The responsibilities provide a high-level overview of what the SEC should do, however these need to re-defined so they can be executed in software logic. Responsibilities 1 – 3 can be defined as separate tasks, while responsibility 4) is technically a necessary part that must be done at the end of each task. Thus, the SEC has three main tasks that it needs to perform within the execution loop:

a) Predicting power production using a Neural Network.

b) Executing commands from external inputs.

c) Collecting and aggregating sensor readings from the Motes.

Tasks a) and c) are executed periodically; a) is executed hourly, while c) is executed every 5 seconds. Task b) is unique because the SEC only executes command when receiving an external input; thus this task can be considered I/O bound. The external inputs can come from the energy scheduling algorithm or the web application. In this case, the only external entity is the web application. Sequentially executing these tasks in a continuous loop would not work because they have different execution conditions. Instead, it is more efficient if the SEC only executes these tasks when it is necessary. Therefore, these tasks need to be performed asynchronously and continuously.

Python features a module called 'asyncio' that can allow for asynchronous execution of functions. An infinite loop is created with the three tasks; where each task is defined as a function in the code. Each function has a sleep timer. This sleep timer temporarily moves the function off the queue and allows for the execution of other functions. Thus, for task a), the function will sleep for 1 hour and then

execute. Task c) sleeps for 60 seconds, then executes. Lastly, task b) sleeps only for 1 second; this brief sleep time is necessary because the program needs to check for external input, however there should still be an interval for the other tasks to have a chance to be put in the queue. The asynchronous data loop and the pseudo code for each function is defined from Figure 15 to Figure 17.

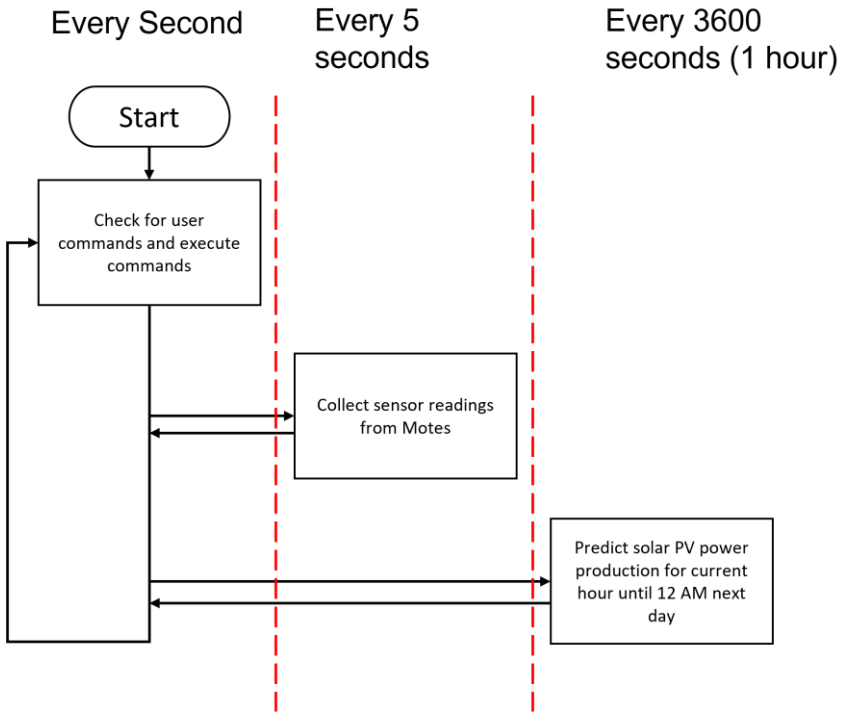


Figure 14: Diagram showing how the asynchronous control loop functions in the SEC.

Algorithm 2: The function that accepts commands from the web application and uploads the responses to AWS

Input: The asynchronous scheduling algorithm trigger and a command from the web application
Output: The response from the Mote or alarms, which are uploaded to AWS

```
1:  Initialize communications with AWS DynamoDB
2:  Initialize communications with AWS IoT Core
3:  Initialize UART communication object
4:  while true do
5:      sleep for 1 second
6:      if new message in the "COMMANDS/system_ID" MQTT topic then
7:          if message is "Toggle PV Mote" then
8:              Save the command "CTRL" for the PV Mote into COMMAND_SEQUENCE
9:          else if message is "Toggle Battery Mote" then
10:             Save the command "CTRL" for the PV Mote into COMMAND_SEQUENCE
11:         else if message is "Toggle Load Mote" then
12:             Save the command "CTRL" for the Battery Mote into COMMAND_SEQUENCE
13:         end if
14:     end if
15:     for each COMMAND in COMMAND_SEQUENCE do
16:         Send COMMAND via UART to the JNXB3
17:         ANSWER = check UART buffer
18:         if ANSWER == "CONFIRM" then
19:             end for
20:         end if
21:     end for
22:     Send "DONE" to JNXB3 via UART
23:     while RESPONSES < Number of commands do
24:         ACK = Check UART buffer
25:         if ACK != none then
26:             Save ACK into ACK_LIST
27:             RESPONSES +=1
28:         end if
29:     end while
30:     for each ACK in ACK_LIST do
31:         if ACK == "RELAY_TOGGLED" then
32:             Save the acknowledgement into CONTROL_LOGS dictionary with the Mote ID
33:             Update the state of the Mote relay in the DEVICE_LIST dictionary
34:         else if ACK is an error then
35:             Log the error and the Mote that had the error into ALARM_LOGS dictionary
36:         end if
37:         if ALARM_LOGS != NONE then
38:             Upload ALARM_LOGS dictionary to the alarm logs table in DynamoDB
39:             Send an MQTT messaged named "ALARM" to the "UPDATES" topic in MQTT
40:         end if
41:         Upload the CONTROL_LOGS dictionary to the control logs table in DynamoDB
42:         Send an MQTT messaged named 'CONTROL_LOGS_UPDATE' to the UPDATES topic in MQTT
43:     end for
44: end while
```

Figure 15: Pseudo code for the SEC function that checks commands and executes them.

Algorithm 3: The function that requests the sensor readings and uploads them to AWS

Input: The asynchronous scheduling algorithm trigger

Output: The collected sensor readings or alarms, which are uploaded to AWS

```
1:  Initialize communications with AWS DynamoDB
2:  Initialize communications with AWS IoT Core
3:  Initialize UART communication object
4:  while true do
5:      sleep for 5 seconds
6:      for each Mote do
7:          Send "SENS" for each Mote via UART to the JNXB3
8:          ANSWER = check UART buffer
9:          if ANSWER == "CONFIRM" then
10:             end for
11:         end if
12:     end for
13:     Send "DONE" to JNXB3 via UART
14:     while RESPONSES < Number of commands do
15:         ACK = Check UART buffer
16:         if ACK != None then
17:             Save ACK into ACK_LIST
18:             RESPONSES +=1
19:             time start = current CPU time
20:         end if
21:     end while
22:     for each ACK in ACK_LIST do
23:         if the response is a sensor reading then
24:             Save the sensor reading into SENSOR_READING dictionary with the Mote ID
25:         else if response is an error or unrecognized then
26:             Log the error and the Mote that had the error into ALARM_LOGS dictionary
27:         end if
28:         if ALARM_LOGS != NONE then
29:             Upload the alarm dictionary to the alarm logs table in DynamoDB
30:             Send an MQTT messaged named 'ALARM' to the UPDATES topic in MQTT
31:         Upload the SENSOR_READING dictionary to the PWRSENS topic in MQTT
32:         end if
33:     end for
34: end while
```

Figure 16: Pseudo code for the SEC function that collects sensor readings.

Algorithm 4: The MicroPython program inside the JNXB3 that acts as Zigbee communication interface between the Motes and the Jetson Nano

Input: A UART message from the Jetson Nano
Output: The response from the Motes that is sent to the Jetson Nano via UART

```
1:  while true do
2:      MESSAGE = Check UART buffer
3:      if MESSAGE has been received then
4:          if MESSAGE == command then
5:              Append the command with the Mote ID to COMMAND_SEQUENCE
6:              Send a confirmation message back to the Jetson Nano
7:          end if
8:          elif MESSAGE == "DONE" then
9:              for each COMMAND in COMMAND_SEQUENCE do
10:                 Check the Mote ID in COMMAND
11:                 Send message to the Mote
12:                 if JNXB3 fails to connect to one of the Motes then
13:                     Send error message to Jetson Nano
14:                     continue_comms = "NO"
15:                 else
16:                     continue_comms = "YES"
17:                 end if
18:             end for
19:             if continue_comms = "YES"
20:                 timeout = 5200
21:                 for each COMMAND in COMMAND_SEQUENCE do
22:                     time_start = current CPU time in milliseconds
23:                     while true do
24:                         RESPONSE = Check the ZigBee buffer
25:                         if RESPONSE has been received then
26:                             confirm_sent = "YES"
27:                             end while
28:                         else if current CPU time > time_start + timeout then
29:                             Transmit a 'no response' message to the Jetson Nano along with the Mote ID
30:                             end while
31:                         end if
32:                     end while
33:                 end for
34:             end if
35:         end if
36:     end if
37: end while
```

Figure 17: Pseudo code for the JNXB3

The key points that shall be noted are the redundancies built into the communications. Firstly, when receiving acknowledgements, both the JNXB3 and Jetson Nano will keep checking until the loop times out or an acknowledgement is received. Obviously, the system should not be blocked if there is an issue with the communications. There have been cases where the UART message got scrambled or clipped. Thus, when issuing commands, the Jetson Nano must receive confirmation that the JNXB3 got the message correctly. It will attempt to issue the same command up to five times if it does not

receive an appropriate response. In essence each stage of the communications will only continue if the previous stage cleared successfully.

Note for task b), each system will be subscribed to a sub-topic corresponding to their system ID. The main topic, ‘COMMANDS’, is the same for each system.

Since the loop executes each function asynchronously, it would be trivial to add another task for automated energy scheduling. The code and logic for issuing commands, triggering alarms and data uploads are already done.

3.4 Motes

In a Solar PV EMS, the Motes have two primary objectives, firstly collecting data on the energy flow and then blocking or allowing the flow of power. The primary requirements are as follows:

1. Can communicate with the Smart Energy Controller
2. Can connect to an electrical wire or cable
3. Can measure the power flowing through the connected electrical wire or cable
4. Can block or enable the flow of electricity through the connected electrical wire or cable
5. Can operate without needing power from other Motes or the SEC

The purpose of the first objective is to ensure that there are no connection issues (ex. damaged power cables) so users can see how much energy is being generated in real time. Furthermore, the collected energy data can be used to train the AI. In Figure 8 of Section 3.1, the Motes are placed at the output of the PV panels, the input to the batteries, and the input to the load. If the user wants to charge the battery, they can turn off Mote #2 and turn on Motes #1 and #3. This precise control enables the charge and discharge of energy resources to fit the user’s lifestyle.

The Motes need to ”end ’nergy readings to the SEC. To measure the flow of enIrgy, a Gravity I^2C Wattmeter is used. As the name suggests, it uses I^2C to communicate with the Xbee 3. I^2C stands for Inter-Integrated Circuit. It is a serial communication protocol that enables integrated circuits (The chips on the Xbee 3 and wattmeter respectively) to communicate [108]. The chip onboard the wattmeter measures current and voltage and uses it calculate the power. The wattmeter’s inputs are rated for 26 V at 8 A [109].

Being able to control the flow of energy allows for energy scheduling. As explained in the previous section, the SEC sends the signals to allow or block the flow of energy; the Motes simply enable the action. To control the flow of energy the Omron Electronics G5LE-1A4 DC3 Power Relay is used. This relay is an electromagnetic (EM) relay.

An EM relay operates by sending an electrical signal to an input coil which emits an electromagnetic field that opens or closes physical contacts. Another option is a Solid-State Relay (SSR). An SSR does not have any moving mechanical contacts and uses an LED to control a photocoupler which allows or blocks the flow of energy [110]. In both devices, the Xbee 3 is connected to the input control lines sends a signal to open or close the relay. Due to the usage of electronics, SSRs are more reliable and can switch faster [111]. As an example, the G5LE-1A4 DC3 Power Relay takes about 10ms to close the contacts and 5 ms to release them, whereas the SVDD series (DC Solid State Relays) by Sure Controls Inc. has a switching time of 25 μs [111][112]. However, when examining the prices on Digikey for EM relays and SSRs, the SSRs are more expensive. Thus, there is a trade-off between performance and price.

To control the relay, a circuit needed to be designed. Figure 18 shows the high-level circuit for the relay control (See Appendix B for the details of the experiment that led to this circuit).

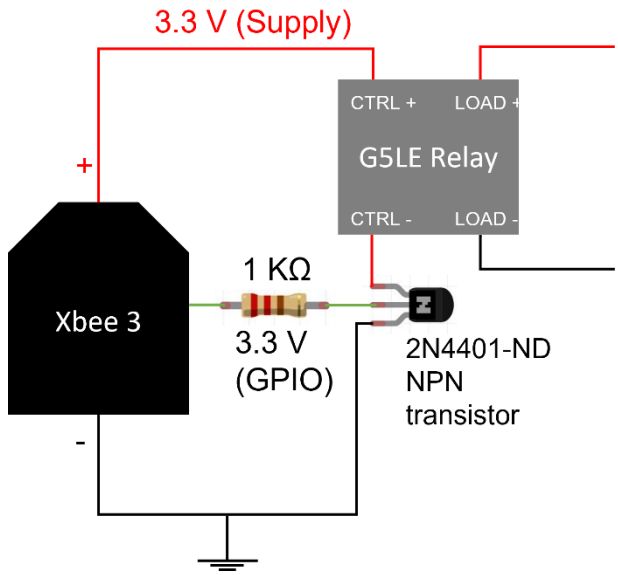


Figure 18: High level diagram of the circuit used to control the relay.

The Xbee's 3.3 V supply pin is directly connected to the control lines of the relay. However, there is no mechanism to control the amount of current being drawn from the supply pin. Thus, a 2N4401-ND NPN transistor was used to control the amount of current being drawn. To control the transistor, a GPIO pin was connected to its Base pin. In this circuit, the transistor acts a current amplifier. It pulls current through the transistor based on the amount of current being supplied to its Base pin. The 1 KΩ resistor is used to reduce the input current to the base. The formulas used to determine the correct transistor and resistor is as follows [113]:

$$\boldsymbol{Base\ Current\ (I_B)} = \frac{\boldsymbol{Collector\ Current\ (I_C)}}{\boldsymbol{Gain\ (\beta)}} \quad \textbf{(1)}$$

$$\boldsymbol{I_B} = \frac{\boldsymbol{Base\ Volt\text{-}ge\ (V_B) - Base\text{-}Emitter\ Voltage\ Drop\ (V_{Be})}}{\boldsymbol{Base\ Resistor\ (R_B)}} \quad \textbf{(2)}$$

The following substitutions are made; I_b = GPIO pin current = 0.003 A, I_c = 0.133 A, thus $\beta \approx 44$. The coil resistance is 25.2 Ω, however it will consider be considered negligible. The calculated gain is a bare minimum and does not guarantee correct operation of the transistor, thus the gain can be multiplied by 5 to get $\beta = 220$. To calculate R_b, it can be said that V_b = GPIO voltage = 3.3 V, and V_{be} = 0.7 V. Note that the value of 0.7 V is the typical value for silicon-based transistors. Thus, Rb ≈ 866 Ω. The closest resistor value available is 1 KΩ. The gain of the transistor and the maximum collector current must be greater than the calculated values. The 2N4401-ND NPN transistor has a maximum gain of 300 with a maximum collector current of 1000 mA. Therefore, it is suitable for the application.

The G5LE-1A4 DC3 output line is Normally Open, meaning it starts disconnected. Also, the relay configuration is Single Pole, Single Throw (SPST). This means there is only one circuit with only one ON position [114]. The relay is rated for 250 V DC/AC and can handle up to 10 A. Since the Mote is wirelessly controlled, it will require a battery. The only requirement is that the battery voltage should be around 2 – 3.6 V. Determining the correct battery sizing would involve determining the requirements with the utilities and the clients to determine how long the Motes batteries should last. Based on these requirements, multiple tests on sites would then need to be performed. Furthermore, there are rules to designing battery circuits. Design for battery circuits alternatives are not available. Given these facts, detailed battery design will not be discussed. Figure 19 shows the Mote hardware architecture.

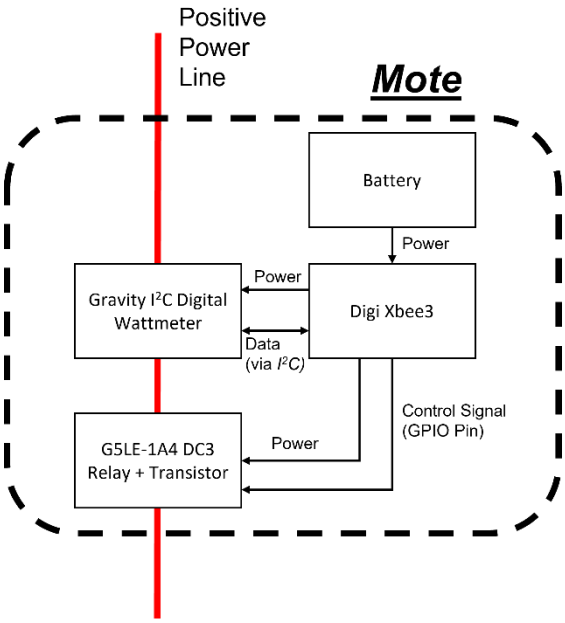


Figure 19: Diagram showing the internal Mote architecture. Note that the Wattmeter and SSR and placed in series on the positive power line.

The software within the Xbee 3 is straightforward. The program continuously polls its Zigbee message buffer. When it receives a new message, it decodes the message and then checks the payload. The payload is one of two key words: CTRL or SENS. If the payload is CTRL, then the Xbee 3 will send a signal to the pin connected to the input coil of the relay. This signal will either toggle it ON or OFF. The relay will then send an acknowledgement back saying the state has been changed. If the payload is "SENS", then the Xbee 3 sends the current reading from the wattmeter. If the Mote does not respond, then the SEC will raise an alarm. The pseudo-code is outlined in Figure 20.

Since the system can measure the power, it is possible to estimate the state of charge SoC of the battery. Battery SoC can be estimated from the Open-Circuit Voltage (OCV). Certain OCV values can be mapped to the SoC [115]. The OCV should be measured at least a couple hours after the last usage which makes it difficult to utilize. It can be combined with Coulomb Counting (CC) to make it more effective for real time counting [116]. For example, the authors in Reference [116] designed an improved CC algorithm using a piecewise SOC-OCV curve. Their algorithm requires real time measurements of the voltage, temperature, and current of the battery. The Battery Mote is useful for

any algorithm that require powers, voltage, or current measurements from the battery. It can send the measurements an algorithm stored in the SEC which can then calculate the battery SoC in real time.

Algorithm 5: The MicroPython program for the Mote

Input: Zigbee message that is a sensor reading request or a relay toggle request
Output: Response to request, either a success or error

```
1:  Initialize sensor
2:  Initialize relay pin
3:  while true do
4:      if (Zigbee message is received) then
5:          if (payload == 'ctrl') then
6:              toggle the relay
7:              get pin state
8:              response = pin state
9:          end if
10:         if payload is 'sens then
11:             perform sensor reading
                response = sensor reading
12:         end if
13:     end if
14:     Send response back to JNXB3 via ZigBee
15: end while
```

Figure 20: Pseudo code for the programming running inside the Mote.

3.5 Client Application

As mentioned in Section 4.1, the client application is hosted on AWS Amplify. The application was developed using HTML, CSS, and JavaScript. Figure 21 shows the dashboard for the client application. The center figure is a high-level diagram of the connections. As observed, the motes display the current power flowing through them.

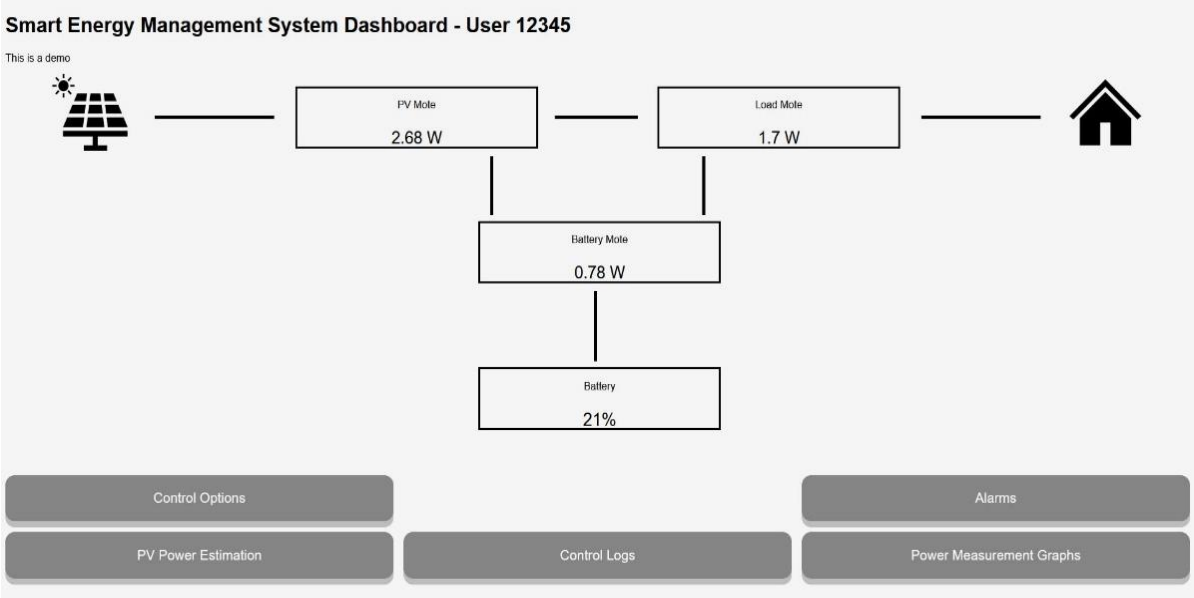


Figure 21: Dashboard for the client application. This is the default view.

Looking at Figure 22 to Figure 26, these show the screens for each button when they are clicked. Starting with the control options in Figure 22, it simply has the three buttons. When a button is clicked, it will publish an MQTT message to the "COMMANDS/*SystemID"* topic, where "*System ID"* is the numeric string that identifies the system. When the user logs in, the system ID will be saved and will be filled into the topic name. The application will then wait and poll the MQTT message buffer. The application is subscribed to the "ACKS/ *System_ID"* topic. The window will display a waiting message. It will receive one of two responses: 'Success' or 'Error'. In the case of the Error, the window will tell the user to check the Alarm logs. Figure 23 shows the table for the alarm logs. There will be a message that describes the alarm, and a date and time of occurrence for the alarm. The Alarm Logs and Control Logs table automatically update when the user clicks on the button. Furthermore, they update whenever the MCP sends an MQTT message indicating an Alarm or Control Logs update.

As discussed, the logic for the Alarms is handled by the SEC. However other problems may be detectable through the client application. If the relay state in the client application is ON, but there is zero power flowing through it, that means either the relay or the energy system may be having an issue. The important part here is that the user has access to real time information, which is partially provided by the Motes. Using this information, they can investigate right away if there is an issue.

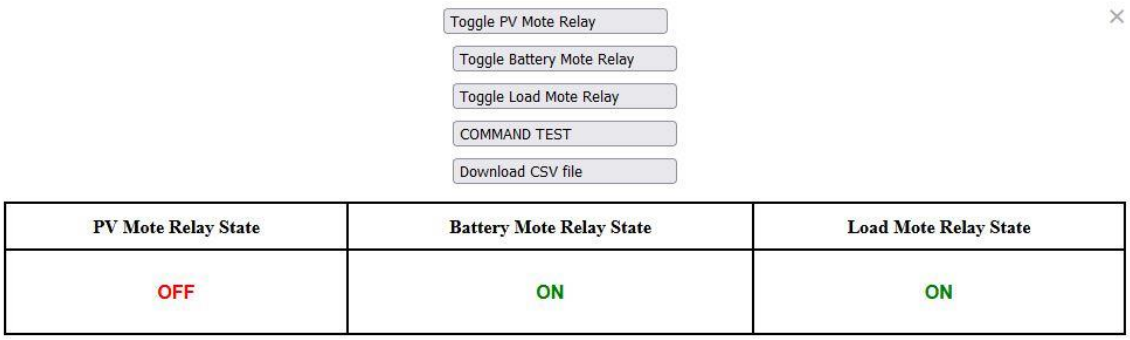


Figure 22: The control options window. Note that the 'COMMAND TEST' and 'Download CSV file' buttons are for testing purposes only.

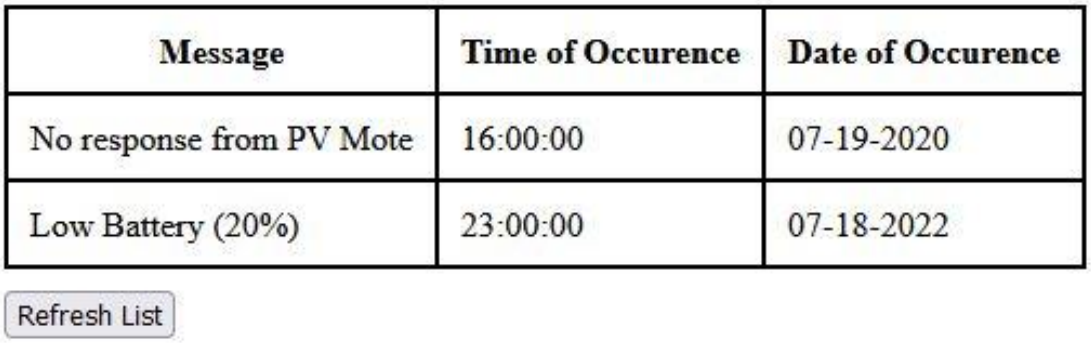


Figure 23: The alarm logs table.

Figure 24 shows the graph for the PV power predictions. For the PV power estimation, the application subscribes to the "PWRPREDICTIONS/ *System_ID"* topic. The neural network within the SEC will make its predictions and then it will be uploaded via MQTT; this is the same neural network presented in Section 3.3.1. Here it will receive an array of 24 values, where each value is represents the predicted power production; the values correspond to their Enumeration (ex. element 11 is represents 1100 Hrs). The graph's y-axis will scale according to the values provided. Figure 25 shows what the control logs table. The table will specify the control action, the time of occurrence, the data of occurrence, and lastly why the action was issued. Furthermore, the table can be manually refreshed

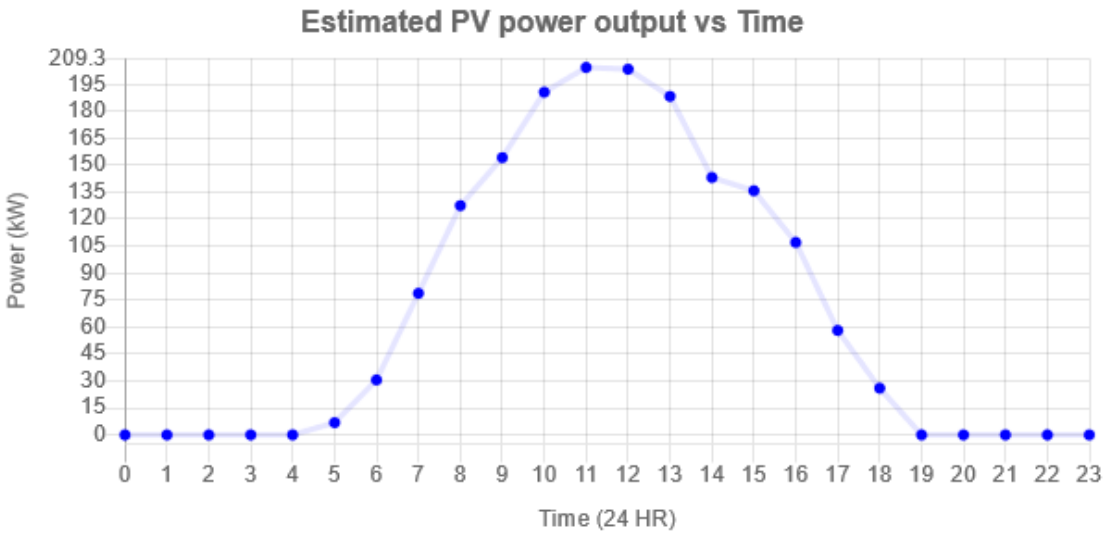


Figure 24: The graph for the PV power predictions. Note that this prediction is made from the NIST Campus data on July 20, 2017.

Action	Time of Occurence	Date of Occurence	Reason
PV Mote has been switched ON	17:50:47	2022-06-21	Insufficient Battery Charge
Battery Mote has been switched ON,	20:21:46	2022-06-16	Insufficient PV power

Refresh List

Figure 25: The control logs table.

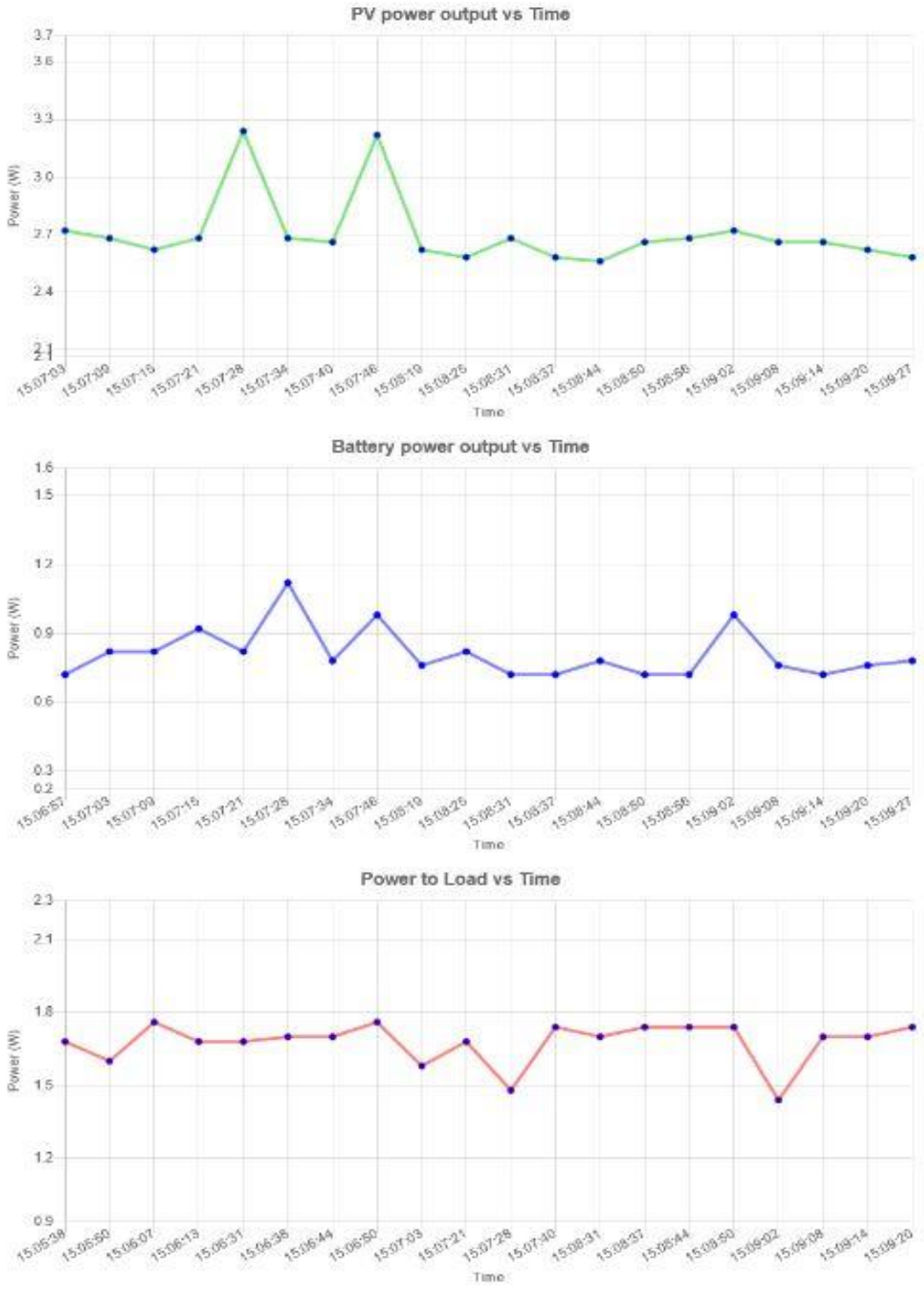


Figure 26: Sample power measurements in real time power measurements graphs.

Figure 26 shows what the power measurement graphs look like. There's a graph that shows the measurements from each Mote. These measurements are samples obtained when all the Motes were turned on. These are examples to show what the graphs look like with data. The timestamps are on the X-axis, while the power measured is on the Y-axis.

3.6 Summary

In this chapter, the hardware and software infrastructure for an EMS is presented. The data management infrastructure is constructed using Amazon Web Services; AWS Amplify, AWS Lambda, AWS API Gateway, AWS DynamoDB, AWS IoT Core, and AWS Cognito. The Monitoring and Control platform consists of a central Smart Energy Controller and three Motes; the Motes are placed at the output of the solar PV system, the output of the battery, and the input to the load. The SEC platform is built from a NVIDIA Jetson Nano Microcontroller and a Digi Xbee 3. Using Python, a program is constructed to enable asynchronous execution of energy flow control, power sensor data collection, and predictions of a solar PV energy system. The Digi Xbee 3 is used to enable wireless communications with the Motes using the ZigBee protocol. Three Motes are used for energy flow control and power sensor data collection. The Digi Xbee 3 is used as the Mote control platform. A G5LE-1A4 DC3 power relay is used to block or allow the flow of energy. A Gravity I^2C Wattmeter is used to measure the power. Energy flow control and power sensor data collection are programmed in MicroPython. The client application displays real time power measurements, control logs, alarm logs, and predicted solar PV power generation for the next 24 hours. Furthermore, users can control the flow of energy and view the Mote relay states (ON or OFF) from the client application. This chapter forms the basis of this thesis; it contains the primary hardware and software used to build the energy management system. The next chapter details the experimental setup used to test the energy management system.

Chapter 4
Experimental Setup

The experiments were conducted on a prototype Solar Cart. Figure 27 and Figure 28 show the pictures of the Mote and controller respectively. Figure 29 shows the circuit diagram of the experimental set up. While Figure 30 and Figure 31 show pictures of the experimental setup. Note that in Figure 30 the flood light was used to simulate the Sun.

The battery on the solar cart is a 12 V battery rated at 4.5 Ah. Note in Figure 30 there are two loads, however, only Load 2 was used. The Digi Xbee 3 requires a 2-3.6 V power supply. Thus, a 3.7 V lithium polymer (LiPO) battery with 1100 mAh capacity was chosen. For the experiments, 3D printed enclosures were made, then they were insulated with antistatic foam.

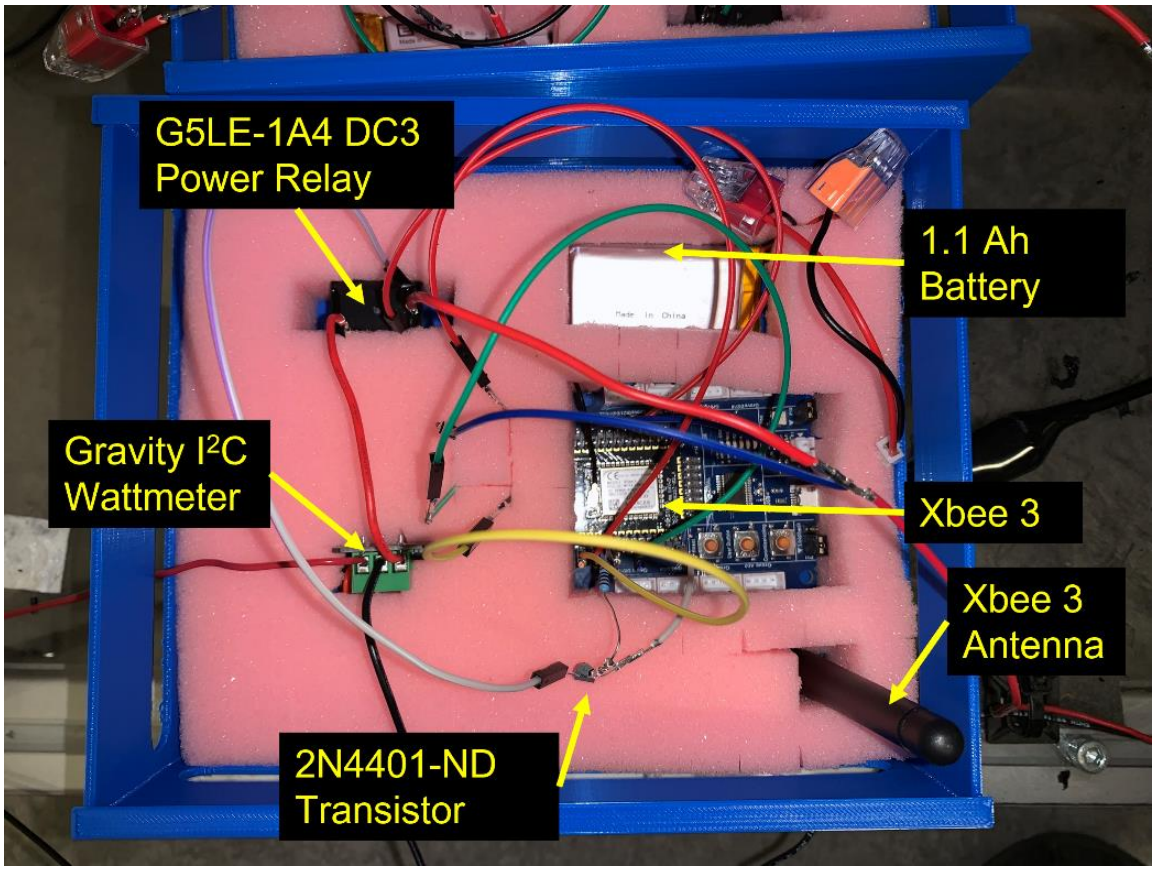


Figure 27: The inside of a Mote enclosure.

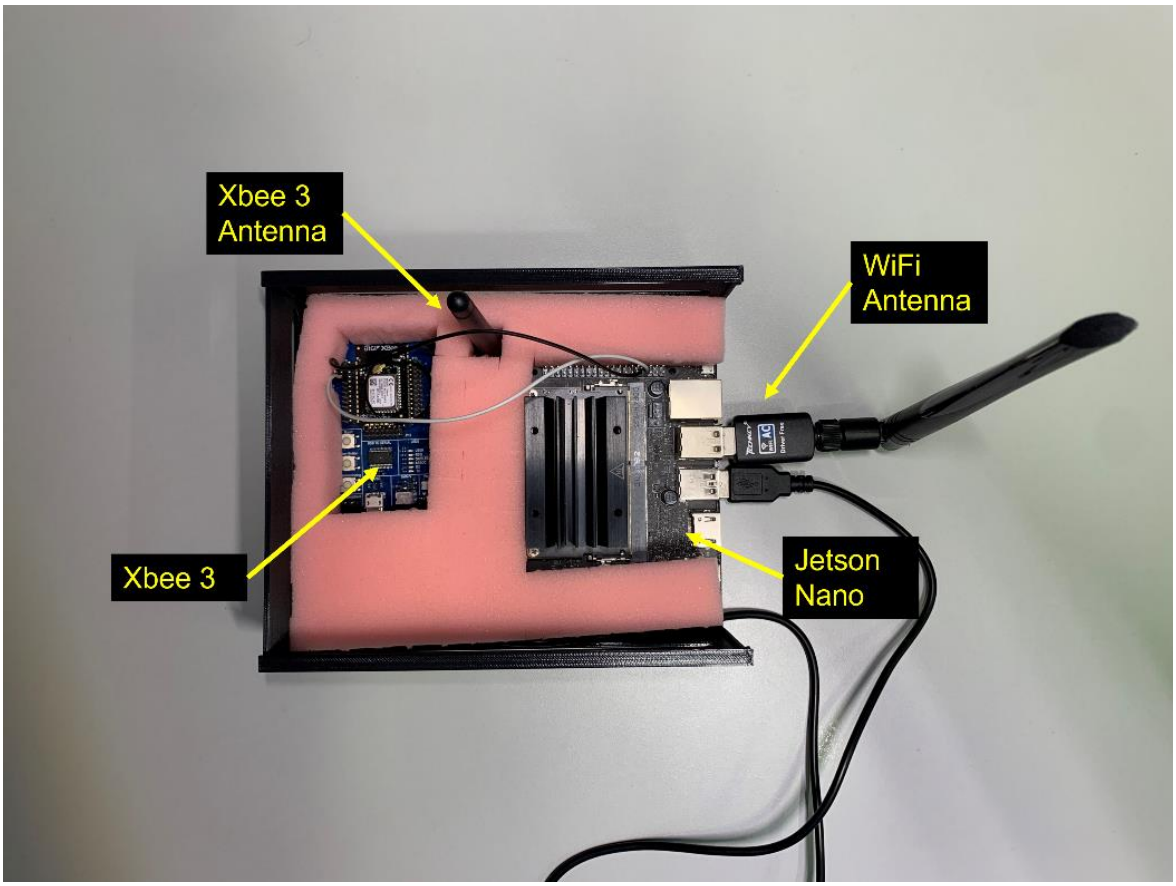


Figure 28: The inside of the Smart Energy Controller enclosure.

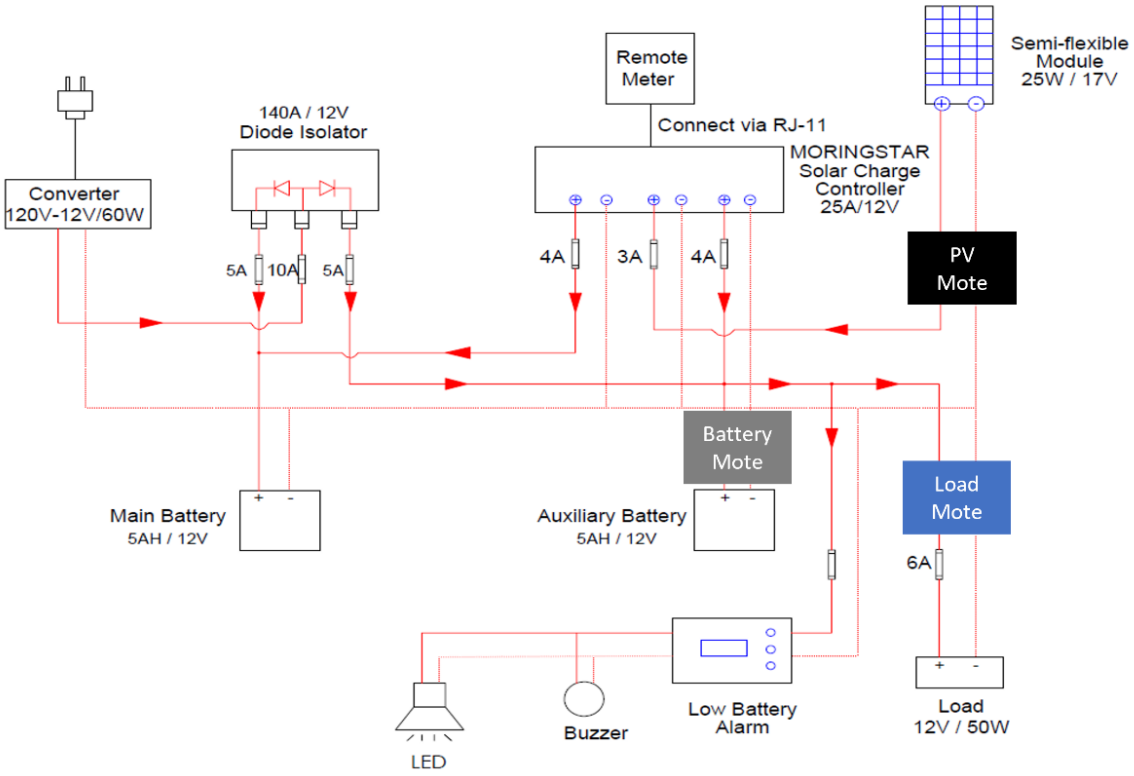


Figure 29: The Solar Cart circuit and the Mote placement

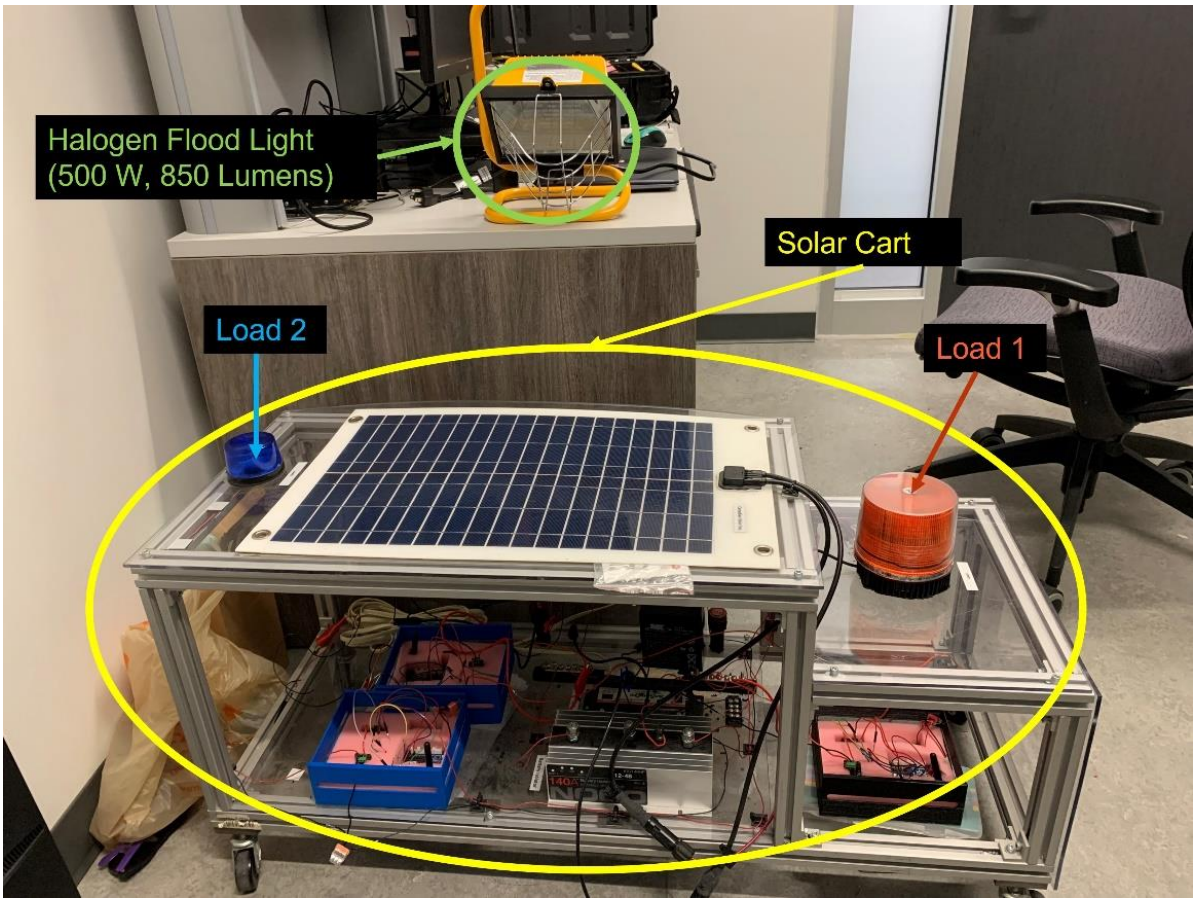


Figure 30: A picture of the experimental setup

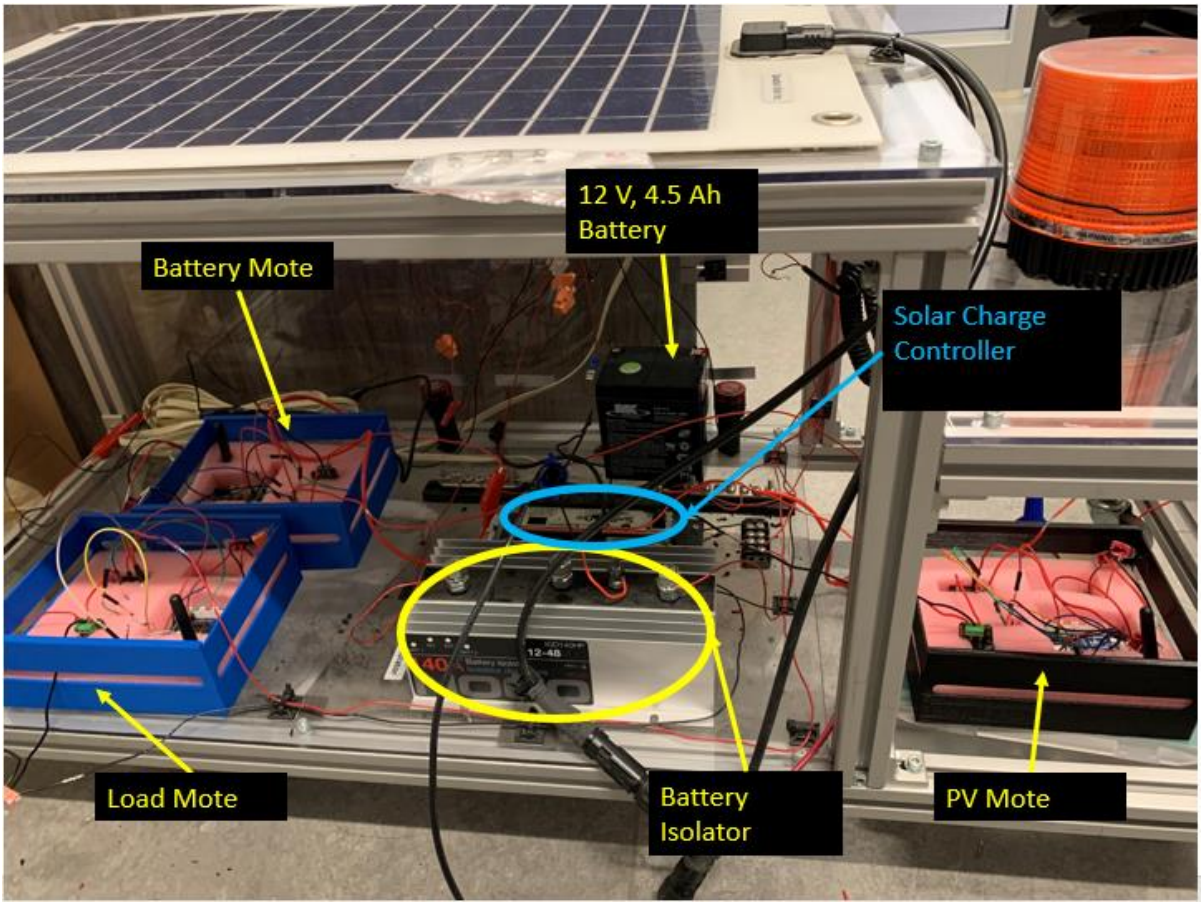


Figure 31: A closeup picture of the Motes and the inside of the Solar Cart.

The scenarios in

Table 10 are carried out to test the performance and verify the functionality of the system. In cases 2-4, all the experiments were automated. In each case, 5 sets of 100 samples were collected. Thus cases 2-4 were ran 5 times, with 100 samples per run.

Table 10: The list of scenarios to be tested or evaluated

Case Number	Description
1	System Validation: Ensuring that the flow of power can be controlled and can be measured.
2	Sensor Data Collection: Perform 5 tests each with 100 sensor measurement samples to gauge the measurement system performance.
3	Control Performance: Perform 5 tests, where each one executes 100 commands from the web app to gauge control system performance.
4	Asynchronous Processing performance: Perform 5 tests, where each one executes 100 commands from the web app and collects 100 sensor readings simultaneously to gauge system asynchronous processing performance.
5	Neural Network Performance: The Neural Network is loaded, makes the prediction, and uploads the data to the dashboard
6	Cost Evaluation: A price breakdown of the proposed EMS and a price comparison to existing energy management systems.

Chapter 5
Results and Discussion

In this chapter, six scenarios are performed to gain a further understanding of the proposed energy management system operation or properties. Scenario 1 checks the functions of each part of the system. Scenario 2 tests the sensor data collection system. Scenario 3 evaluates the execution time when issuing commands. Scenario 4 enables asynchronous scheduling; the functionality of the sensor data measurement is checked, and the command execution time is analyzed. Scenario 5 evaluates the prediction performance of the developed Neural Network. Lastly, Scenario 6 outlines the cost of the proposed energy management system and evaluates the pricing of other energy management systems.

5.1 Scenario 1: System Validation

In this scenario, there are five objectives:

1. Transmit and receive messages from the PV Mote, Battery Mote, Load Mote
2. The relays can be toggled ON and OFF
3. The sensors within the PV Mote, Battery Mote, and Load Mote can measure the flow of energy
4. Ensure the dashboard correctly displays the measured values and state of the system
5. Ensure that the alarms are triggered and will display on the screen

Prior to analyzing the results, the expected behaviour is outlined. Figure 32 shows the system flow and the key steps at each point.

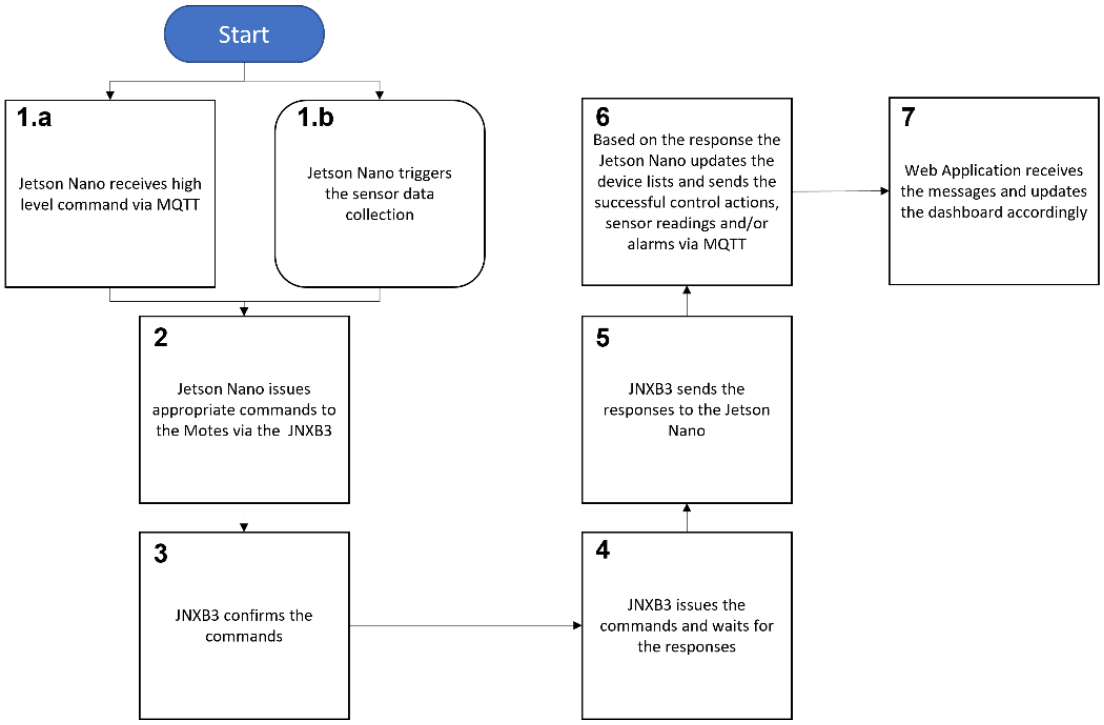


Figure 32: The flow of expected behavior in the system

To confirm and understand what is going on in the MCP, the actions had to be logged in some way. On the Jetson Nano, key actions were logged in the terminal. From Figure 33 to Figure 35, these show the terminal outputs from the Jetson Nano for each command. The expected behaviour can be evaluated compared to the outputs shown in Figure 32. As shown in each Figure, the first line states that the system has received a message from MQTT topic 'COMMAND/123', where 123 represents the system ID. Furthermore, the "msg" portion, represents the actual command which tells the Jetson Nano which Mote to toggle. This fulfills step 1.a from Figure 32. As observed the respective commands are "PVMote_Toggle_Relay", "BattMote_Toggle_Relay", or "LoadMote_Toggle_Relay". Furthermore, the "CONFIRMED" comes from the message from the JNXB3 indicating that it has received and recognized the command. This fulfills steps 2 and 3 from Figure 32.

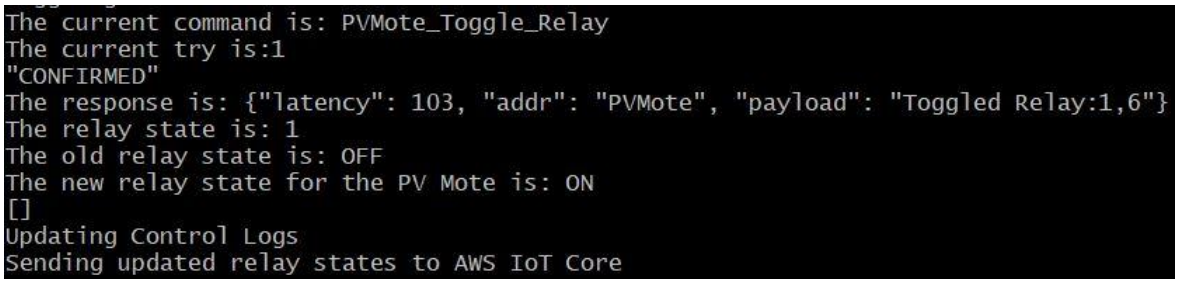


Figure 33: The "Toggle PV Mote" command as seen in the Jetson Nano terminal

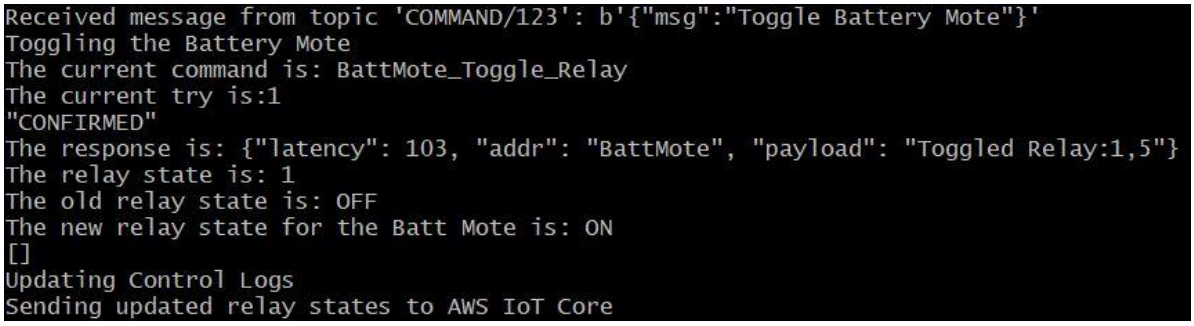


Figure 34: The "Toggle Battery Mote" command as seen in the Jetson Nano terminal

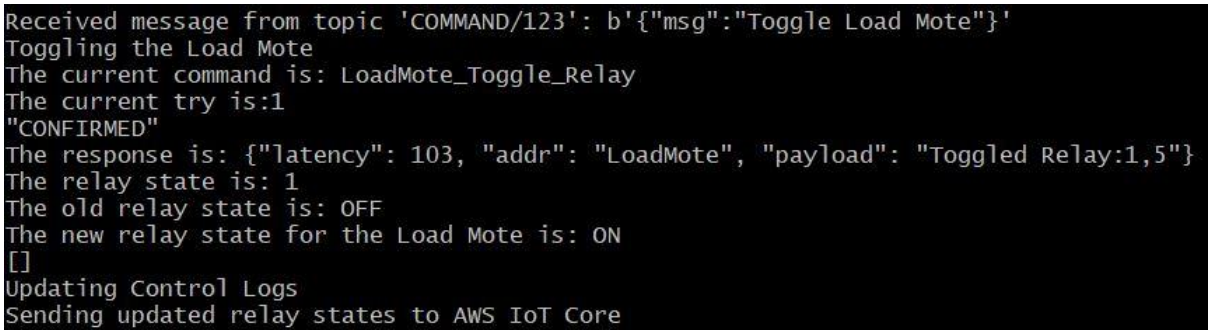


Figure 35: The "Toggle Load Mote" command as seen in the Jetson Nano terminal

Next, the responses from the Motes have been received. The object has two attributes; "addr" represents which Mote the response is from, while "payload" is the actual message from the Mote. The "Toggled Relay:1", indicates that the relay has been connected. Thus steps 4 and 5 have been fulfilled. Then the Jetson Nano prints the states of the old relays and reports the updated relay states; internally in the code, this means the Jetson Nano has updated the states of the device within its own

list. Then it must update the relay states by sending a message via MQTT. Lastly in Figure 36, it shows the ‘Success’ message, which indicates step 6 is completed.

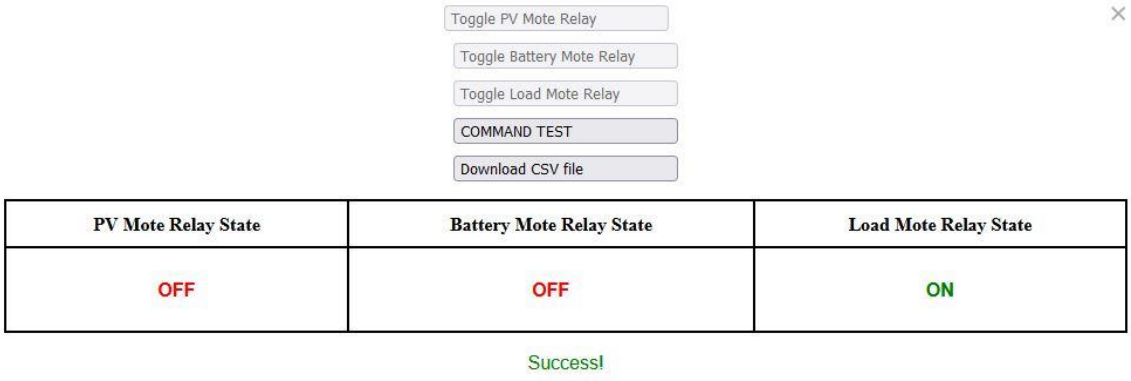


Figure 36: The “Success” message as seen in the web application

There may be times when the device is damaged and cannot respond, thus the Alarm functionality must also be tested as well. Looking at Figure 37 to Figure 39, these demonstrate the alarm system functionality. In Figure 37, the SEC fails to connect to one of the Motes, thus sends a notification via an MQTT message. Figure 38 shows the failure message in the window, while Figure 39 shows the alarm log message in the ‘Alarms’ window.

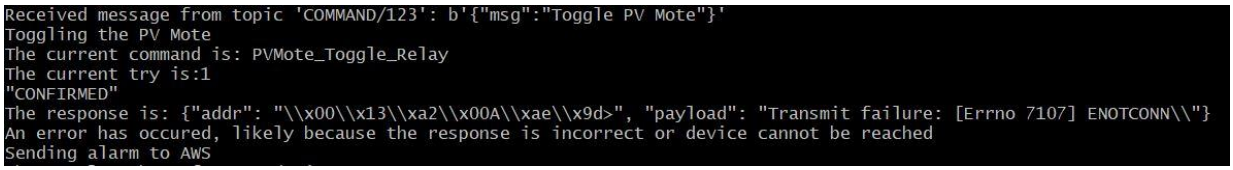


Figure 37: Transmit Failure as seen in the Jetson Nano

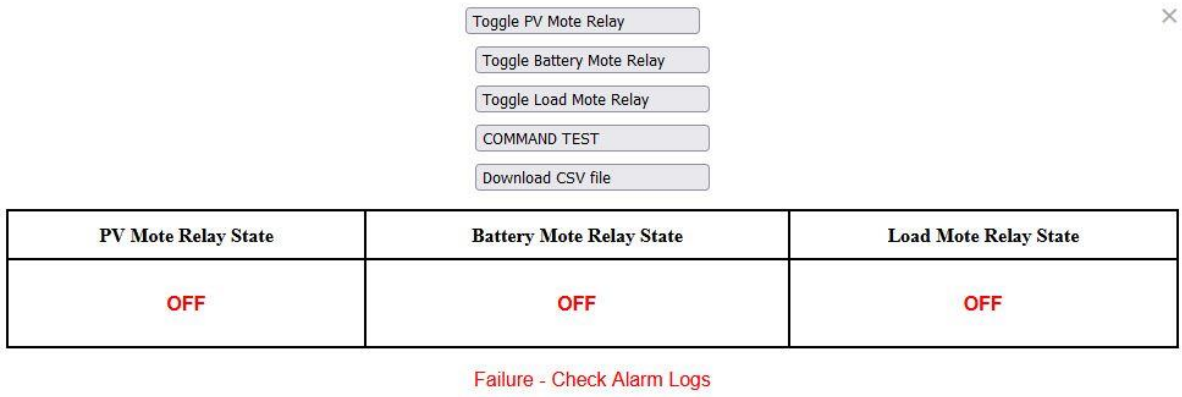


Figure 38: Transmit Failure as seen in the web application

Message	Time of Occurence	Date of Occurence
Transmit Failure: Device cannot be reached	19:30:12	2022-08-23

Figure 39: Mote Error and Transmit Failure in the Alarm Logs

Next the sensor reading functionality can be checked. Figure 40 displays the terminal output of the Jetson Nano when issuing sensor readings. It shows that it fulfills steps 1.b to 5 in a similar way. The responses show that the Battery Mote and Load Mote report 2.12 W and 2 W respectively while the PV Mote reports 0.0. Note that the power measurements are in watts (W).

Now examining the web application in Figure 41, it is observed that it does report the same values as measured by the sensors. While Figure 42 shows the real time measurements as seen from the web application. Furthermore, the similar measurements indicate that the sensor measurements are working correctly. The 0.12 W difference can be attributed to power losses during delivery. Note that in Figure 42, the graphs were reset. This was done so the measurements for this test could be quickly and easily distinguished.

```
The current command is: PVMote_sensor_reading
The current try is:1
"CONFIRMED"
The current command is: LoadMote_sensor_reading
The current try is:1
"CONFIRMED"
The current command is: BattMote_sensor_reading
The current try is:1
"CONFIRMED"
The response is: {"latency": 139, "addr": "PVMote", "payload": "sens_val:0.0,19"}
The response is: {"latency": 126, "addr": "LoadMote", "payload": "sens_val:0.0,3"}
The response is: {"latency": 113, "addr": "BattMote", "payload": "sens_val:40.0,8"}
{"sensor_readings": {"PVMote": "0.0", "LoadMote": "0.0", "BattMote": "0.04", "time": "19:52:22
"}}
Sending sensor readings to AWS IoT Core
```

Figure 40: The Mote sensor readings in the command line

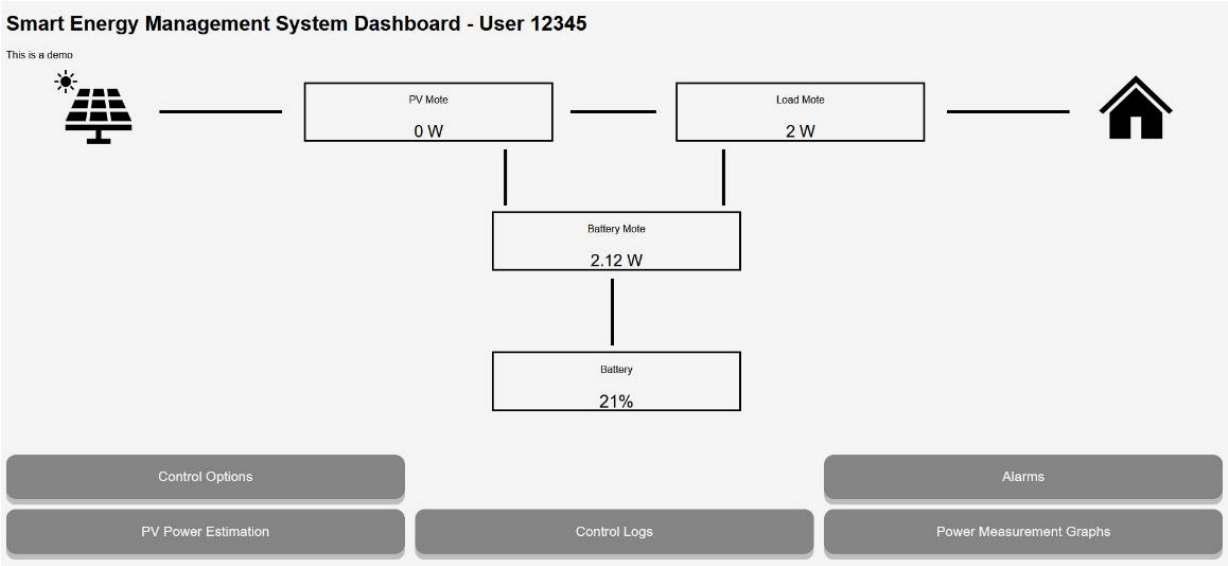


Figure 41: The PV Mote and Load Mote sensor readings on the dashboard. Note that the battery state of charge in the picture is a placeholder since the experiment did not actually use a separate battery.

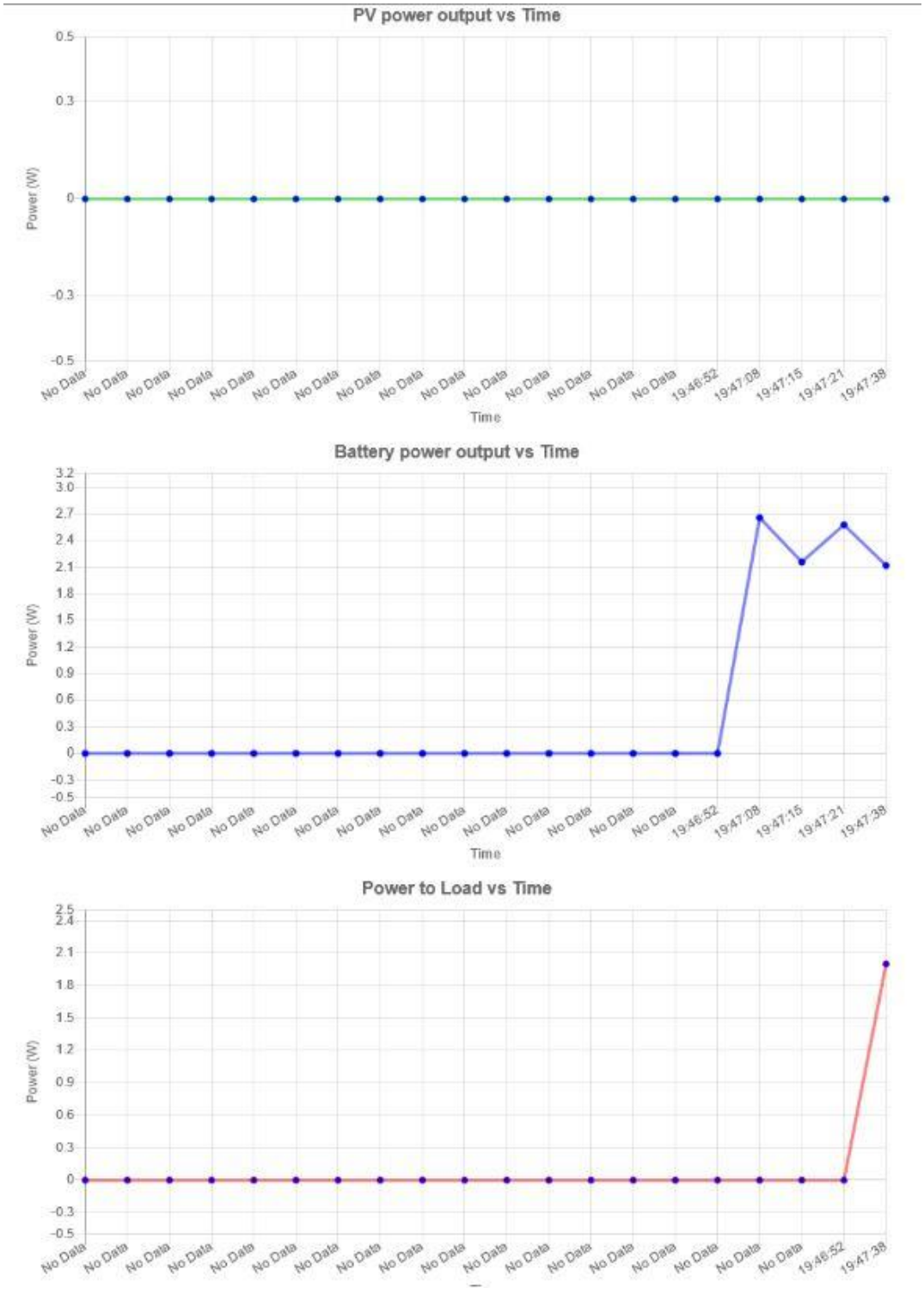


Figure 42: The real time graphs of the Mote sensor readings in the web application.

5.2 Scenario 2: Sensor Data Collection

The objective of this scenario is to determine the reliability and consistency of the data based on the number of tests. This will help to understand the functionality of the data measurement system. Furthermore, it will help provide a baseline for Scenario 4. This is to ensure that it can collect data repeatedly. To get an idea of the performance, 5 sets of 100 sensor reading requests per mote were conducted. Thus, bringing a total of 1500 data points.

Figure 43 shows the collected measurements. The diamonds represent the readings from the PV Mote. The circles represent the readings from the Load Mote. Lastly, the triangles represent the readings from the Battery Mote. There are enough scatter points that a consistent pattern can be observed. The sum of the magnitudes of the load mote and battery mote sensor readings are approximately equal the PV Mote reading. This indicates the PV panel is charging the battery and while supplying the load. There are some outliers, in the 400 – 500 sample number range, and these are believed to be caused by power fluctuations. While the battery readings are slightly more spread out, they still consistently hover around the 0.6-0.8 Watts and clear trend can still be observed. Furthermore, from these sets it can be observed that this behaviour is normal form the battery.

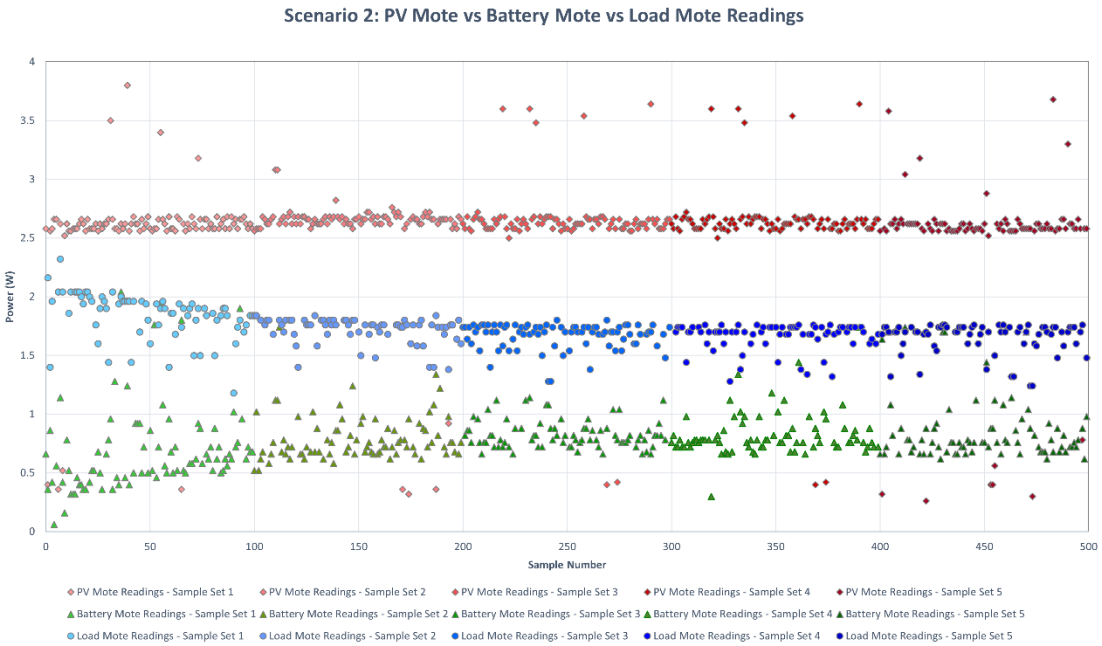


Figure 43: The sensor readings obtained from the PV Mote, Battery Mote, and Load Mote in Case 2.

It should be noted that there were 190 missing Mote readings out of the 1500 readings requested. Each test had approximately 40 missing data entries. Examining the terminal logs revealed that the JNXB3 would sometimes send a sensor reading from a Mote a second time; thus, there were three responses, but one of them was a duplicate. Therefore, when the Jetson Nano checked the responses, it would not be able to record a reading for one of the Motes. Note that this behaviour did not stop the program from continuing to collect data. For future improvements, a confirmation system can be put in place to verify that each response is unique; if it is not, then it should re-issue a sensor reading request. The challenge with this idea is minimizing the time delay. Additionally, a logic analyzer or oscilloscope can also be used to examine the UART signals. However, these missing points help establish a baseline for the consistency, which will be used in Scenario 4.

Despite this observation, it was still able to consistently report enough readings to understand the current power flow in the system. Furthermore, there is enough data available to see a trend and to confirm the readings between each set. This is closely tied to its consistency as well because the data and trends were similar across the 5 sets. The experiment here is not for testing the service life or the sensor material itself. The purpose is for consistency in terms of running further experiments and understanding the operation of the data collection system. While the reliability can be improved, the results demonstrate that the architecture it can serve as a foundation for the data collection system.

5.3 Scenario 3: Control Performance

The objective of this scenario is to determine and understand the processing time for sending and receiving commands

To achieve this objective, a test was performed, and 500 command samples were taken and analyzed. The test was automated from the web application and would send out a command every $5 + n$ seconds, where n is a random integer between 1 and 3. The data points are:

1. Request Processing Time: The time that is measured from when the Jetson Nano receives a command via MQTT until it confirms the last command with the JNXB3
2. Xbee Communication Processing Time: The time measured from when the Jetson Nano confirms the last command with the JNXB3 up until it receives the last response from the JNXB3. The confirmation of command is regarded as part of sending the commands,

furthermore the Jetson Nano and JNXB3 will be regarded as a monolith since they are both a part of the SEC

3. Response Processing Time: The time that is measured from when the Jetson Nano receives the last response from the JNXB3 until it sends a response via MQTT.
4. Total Processing Time: The time measured from when the web application sends the MQTT message until it receives an MQTT message
5. The MQTT Latency: It is equal to: Total processing time – Total control system processing time.

Various metrics were calculated across a range of 500 samples to determine if they were representative. In the 500 samples, there were 50 samples that could not be obtained likely due to noise corruption. The noise may have corrupted the encoding or the payload in the UART communications. It is acknowledged that since the system is connected by wires that it is more susceptible to noise. An improvement could be making a retry system if responses are unreadable or if the responses take too long; however, adding this mechanism may cause more time delays. Furthermore, a logic analyzer can be used to log and further examine the contents. Despite the unusable data, there is still plenty of remaining data to make calculations.

Table 11 shows the standard deviation and 95th percentile for the various time measurements. The 95th percentile was calculated because for time-based metrics it is more valuable to understand the upper bound for the slowest possible time. Thus, improvements that can reduce the upper bound will likely also reduce the average time.

The system operates In ‘real time’, however the definition of ‘real time’ varies from system to system. Therefore, to get a baseline of comparison, boundaries from reference [117] will be used:

1. Times that are 0.1 seconds or less is considered instantaneous.
2. Between 0.1 and 1 second is considered the limit for a user’s thought to be uninterrupted, however the delay is observed.
3. 10 seconds is regarded as the upper bound before the user will move on to do other things. Based off these boundaries, the system processing time for commands should be around 1 second to be considered ‘real-time’.

In this scenario, these numbers provide a good comparison for determining the latencies with respect to the commands. Since the commands are something that the users consciously perform, it is something that they will observe. As observed in Table 11, the results do not deviate too much, thus it was deemed that taking 500 samples is enough and the obtained data is representative of the system performance.

Table 11: The 95th percentiles and standard deviations for each time measurement for various sample ranges.

Sample Range	*100 Samples*	*200 Samples*	*300 Samples*	*400 Samples*	*500 Samples*
Total Controller Processing Time–- Standard Deviation (seconds)	0.3013	0.2991	0.3027	0.3046	0.3036
Total Controller Processing Time–- 95th Percentile (seconds)	1.4997	1.5055	1.5257	1.5283	1.5264
Request Processing Time–- Standard Deviation (seconds)	0.2994	0.2997	0.3026	0.3045	0.3036
Request Processing Time–- 95th Percentile (seconds)	1.1607	1.1672	1.1876	1.1899	1.1882
Xbee Communication Time–- Standard Deviation (seconds)	0.0001	0.0001	0.0001	0.0053	0.0047

Xbee Communication Time–- 95th Percentile (seconds)	0.3038	0.3039	0.3039	0.3039	0.3039
Response Processing Time–- Standard Deviation (seconds)	0.0283	0.0235	0.0209	0.0205	0.0195
Response Processing Time–- 95th Percentile (seconds)	0.0550	0.0501	0.0477	0.0416	0.0404
Mote Processing Time–- Standard Deviation (seconds)	0.0019	0.0016	0.0017	0.0017	0.0017
Mote Processing Time–- 95th Percentile (seconds)	0.0110	0.0083	0.0110	0.0110	0.0110

Table 12: The 95th percentiles and standard deviations for each time measurement. Data is calculated from 500 samples.

Measurement	**Standard Deviation (seconds)**	**95th Percentile (seconds)**
Total Processing Time	0.40368	1.61755
Total Controller Processing Time	0.30362	1.52643
Request Processing Time	0.30360	1.18821
Mote Processing Time	0.00165	0.01100
Xbee Communication Time	0.00474	0.30390
Response Processing Time	0.01949	0.04037

In Table 12, the 95th percentile for the total processing time is approximately 1.6 seconds. It is a little higher than the target time of 1 second. Figure 44 shows a box and whisker plot of the total processing time. The top and bottom "whiskers" represent the minimum and maximum of the data; in this case it is 0.6 s and 1.8 s respectively. While the number at the bottom of the box represents the upper bound for the first quartile of data. Approximately 25% of the measured processing times are within 0.9 seconds. The middle number represents the median. So roughly half of the measure processing times fall within 1.19 seconds. Lastly, the number at the top of the box represents the third quartile. Thus, about 75% of the processing times are 1.4 s or less. It should be noted that the 95th percentile is calculated from the measured data and that Figure 44 omitted the outliers. There were 5 outliers that were beyond the whiskers. Data points were considered outliers if the difference between them and the third quartile or first quartile was larger than 1.5 times the interquartile range. The interquartile range is the third quartile value minus the first quartile value; in this case it is 1.3249. The outliers were also removed to increase viewability. The standard deviation is large relative to the median and is roughly 33% of the value. This indicates that there may be some improvements to the control system that can be made.

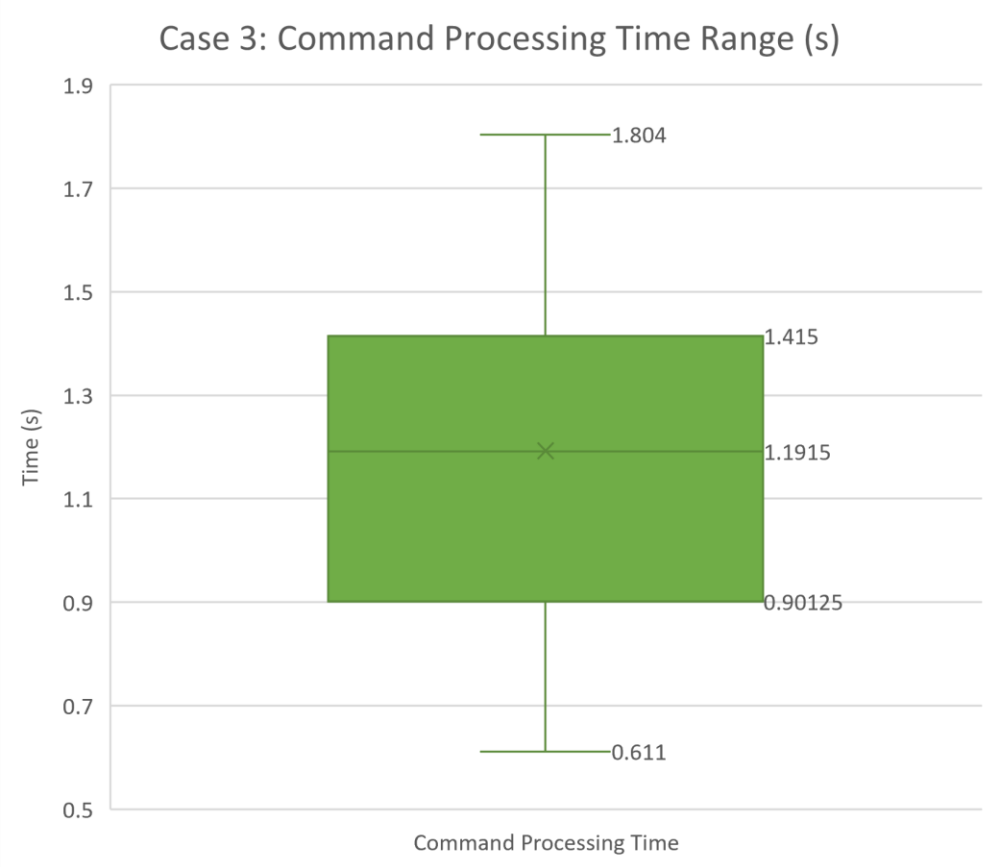


Figure 44: A plot depicting typical range of time that the system takes to send a command and receive a response

To understand what the improvements can be made, the processing time is broken down. Obviously, the control system processing time constitutes most of the total processing time; the MQTT latency is negligible in this case and is not something that can be controlled. Breaking down the controller processing time, it is observed that most of the time is taken during the initial processing of the request. Furthermore, the standard deviation of the total processing time is also heavily influenced by the standard deviation of the controller processing time.

Examining the metrics for the Mote processing time and Xbee communication time, it is observed that the time delays are small and have small standard deviations. This indicates that the programming for the Xbees is stable and consistent. While the response processing time is also low at around 40 ms, but the standard deviation is almost 20 ms. This standard deviation is about 50% of the

calculated upper bound. However, the standard deviation may be heavily influenced by outliers. Based off the definition presented earlier, there are 24 outliers.

These numbers give an approximate idea of performance of each part of the system. To get a better understanding, Figure 45 and Figure 46 display the actual samples vs the time measurements. Firstly, Figure 45 plots the request processing time vs the samples. As observed, the times vary wildly between 0.2 s and 1.4 s. This figure does help explain the large standard deviation. Figure 46 shows the Xbee communication time measurements of each sample. As observed, this section of the system is very stable at approximately 0.3 seconds with only one outlier. Lastly, in Figure 47 it is observed that the response processing time is relatively stable and hover around 0.03 s. However, there are some spikes where it reaches around 0.2 s; these spikes may explain the large standard deviation calculated in Table 12. As observed from the tables and the graphs, the improvements to the total processing time would be focused on the request processing component of the program.

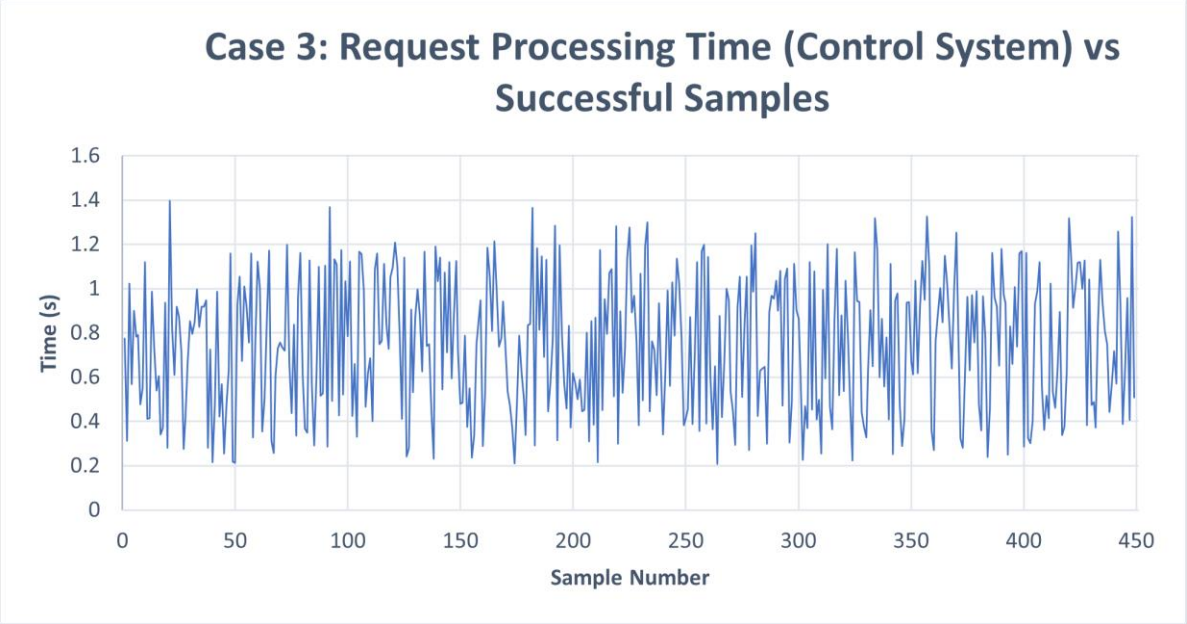


Figure 45: Sample measurements of the time it takes for the controller to process a command before sending it out to the Motes

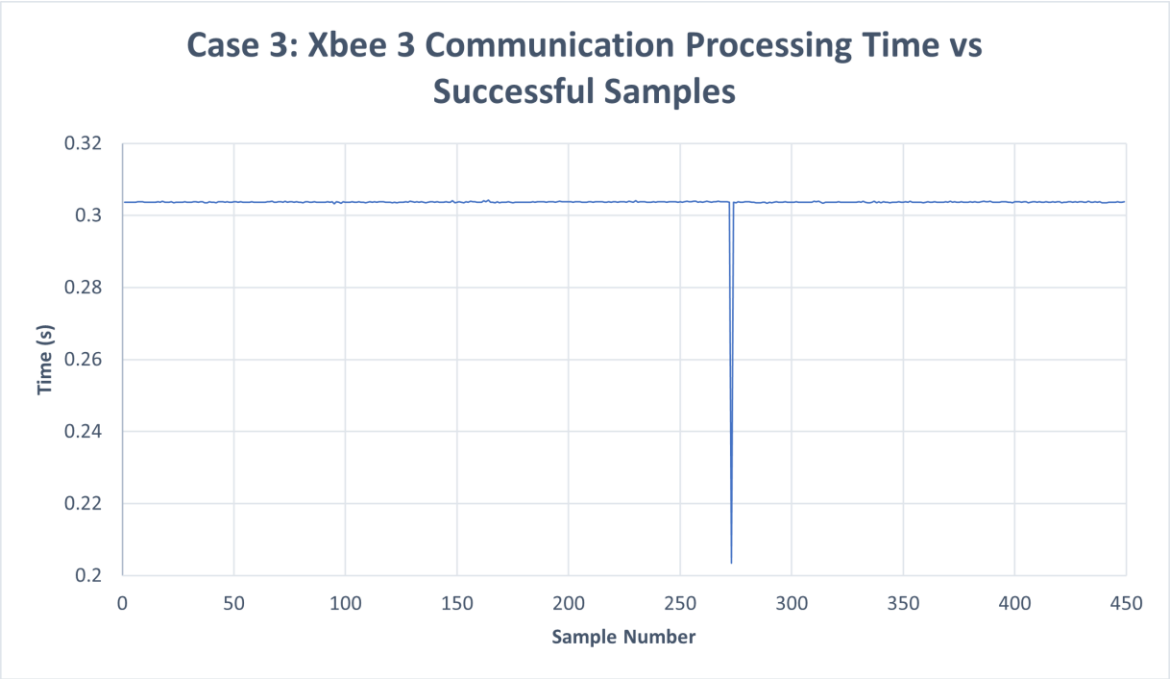


Figure 46: Sample measurements of the time it takes for the Xbee communications portion of the control system to finish

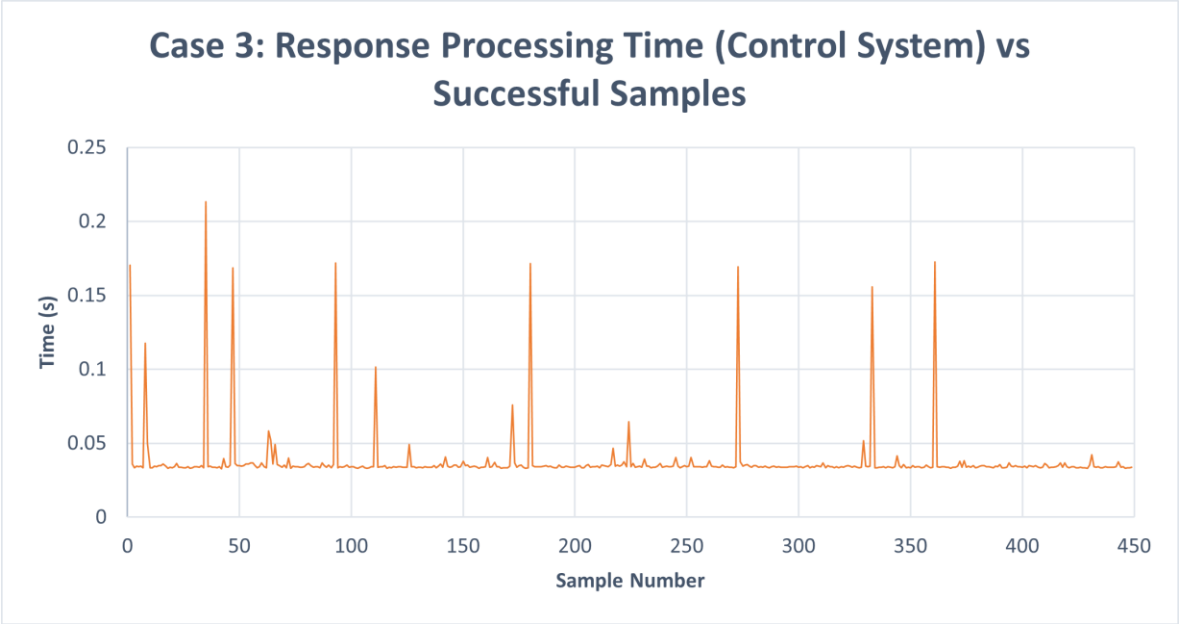


Figure 47: Sample measurements of the time it takes for the controller to process the responses before sending it to AWS IoT Core.

It is believed that the section with the most variability would be when the Jetson Nano confirms the command with the JNXB3. This sequence is contingent on the JNXB3 processing time and processing the UART communications. The JNXB3 has to go through the list of possible commands, so if the command is at the top of the list, then it will finish quickly, otherwise it has to iterate down the list and check it against each known command. Also, the Jetson Nano must strip the encoding on the UART messages before it can read it. Thus, due to the amount of interaction needed here, it is believed the variability in time comes from this section.

To reduce this sequence, the confirmation could potentially be skipped, given how reliable the communications infrastructure and hardware has been. Furthermore, another way to check the commands without comparing it against each case could be devised.

5.4 Scenario 4: Asynchronous Processing Performance

The objectives of this scenario are:

a) Determining the functionality of the data collection system when asynchronous scheduling is active

b) Determining the performance impact of asynchronous scheduling on the control processing time

The collected and calculated data points are the same as the ones discussed in Cases 2 and 3. There were 5 tests conducted, each with 100 command requests. In the 500 command samples, there were 53 samples that could not be obtained due to unrecognized responses. Again, this is likely caused by the noise corruption. There was enough data to perform an analysis and get an understanding of the system performance. As shown in Table 13, the results do not deviate too much, thus it was deemed that taking 500 samples is enough and the obtained data is representative of the system performance.

Table 13: The 95th percentiles and standard deviations for each time measurement for various sample ranges for Case 4.

Sample Range	*100 Samples*	*200 Samples*	*300 Samples*	*400 Samples*	*500 Samples*

Total Controller Processing Time–- Standard Deviation (seconds)	0.7616	0.6417	0.5922	0.5902	0.5803
Total Controller Processing Time–- 95th Percentile (seconds)	2.5765	2.3485	2.2569	2.2718	2.3566
Request Processing Time–- Standard Deviation (seconds)	0.7701	0.6481	0.5973	0.5942	0.5845
Request Processing Time–- 95th Percentile (seconds)	2.2345	2.0112	1.9193	1.9342	2.0166
Xbee Communication Time–- Standard Deviation (seconds)	0.0105	0.0105	0.0087	0.0075	0.0082
Xbee Communication Time–- 95th Percentile (seconds)	0.3039	0.3039	0.3039	0.3039	0.3039
Response Processing Time–- Standard Deviation (seconds)	0.0114	0.0179	0.0169	0.0162	0.0159
Response Processing Time–- 95th Percentile (seconds)	0.0376	0.0413	0.0394	0.0396	0.0401

Mote Processing Time–- Standard Deviation (seconds)	0.0022	0.0020	0.0018	0.0019	0.0019
Mote Processing Time–- 95th Percentile (seconds)	0.0080	0.0080	0.0080	0.0080	0.0080

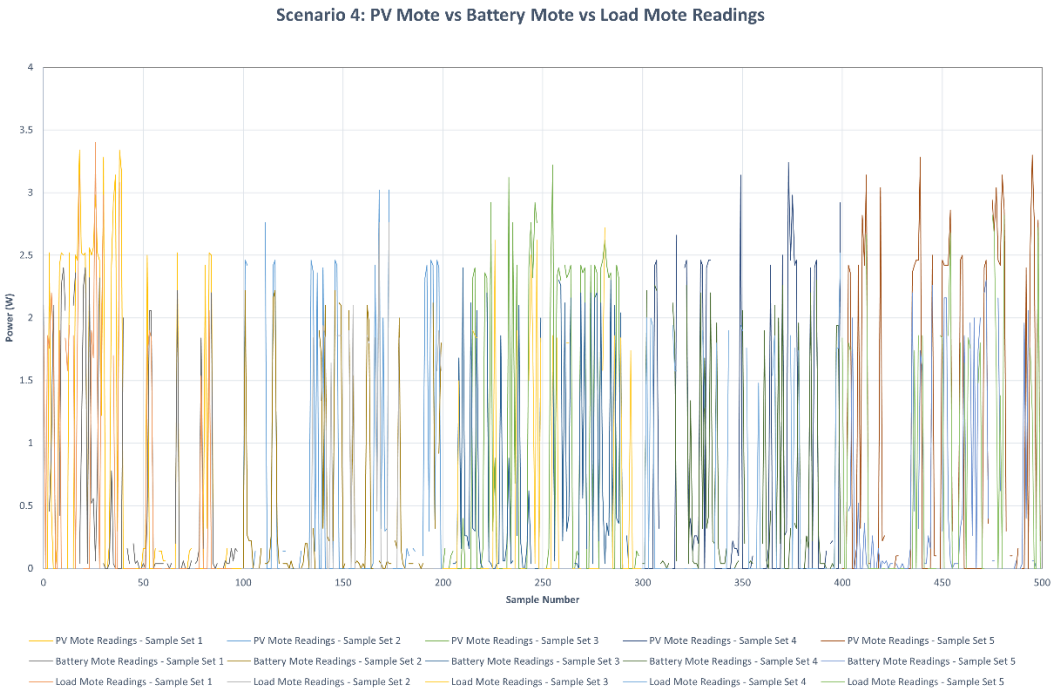


Figure 48: The sensor readings obtained from the PV Mote, Battery Mote, and Load Mote in Case 4.

As shown in **Error! Reference source not found.**, despite the asynchronous scheduling and p rocessing the system was still able to collect the sensor data. There were 178 missing data points out of the 1500 collected. There were about 45 missing data points per test. This is not much different than the 190 missing data points reported in Scenario 2. As described earlier, it is primarily due to the JNXB3 sending the same sensor reading twice. While 13 samples, or 39 data points, could not be obtained due to noise corruption. Examining the plot, the system is still able to consistency collect the

data; we can tell when power is flowing and when it is turned off. From Scenarios 2 and 4, the reliability is acceptable enough to serve as base for further development, however it can be improved by fixing the repeated sensor readings.

The asynchronous processing affected the commands differently. As shown in Table 14, the Xbee 3 communication time, Mote processing time, and request processing time are not affected by the asynchronous processing. The Motes are not affected by the asynchronous scheduling since their software does not feature any asynchronous processes. Furthermore, the request processing time is not affected either because this happens after all the information has been gathered; the system is already locked into processing the sensor requests or command requests so it cannot be interrupted. Obviously, the request processing time is affected by the asynchronous scheduling because it competes with the sensor readings for priority in the queue. Compared to Case 3, the 95th percentile has increased to 2 seconds and the standard deviation has increased to 0.5 seconds. This in turn has affected the total processing time. This is consistent from the results in Case 3, where most of the total processing time came from the request processing time.

Table 14: The standard deviations and 95th percentiles for the command time measurements for Case 4.

Measurement	Standard Deviation (seconds)	95th Percentile (seconds)
Total Processing Time	0.6940	2.5326
Total Controller Processing Time	0.5803	2.3566
Request Processing Time	0.5845	2.0166
Xbee Communication Time	0.0082	0.3039
Mote Processing Time	0.00165	0.01100
Response Processing Time	0.0159	0.0401

Figure 49 presents the true distribution of the processing times. Discounting outliers, the maximum limit appears to be approximately 2.5 seconds. While most of the processing times fall between 1 and

1.6 s. Compared to Case 3, this is no significant increase as the previous distribution was between 0.9 s and 1.4 s.

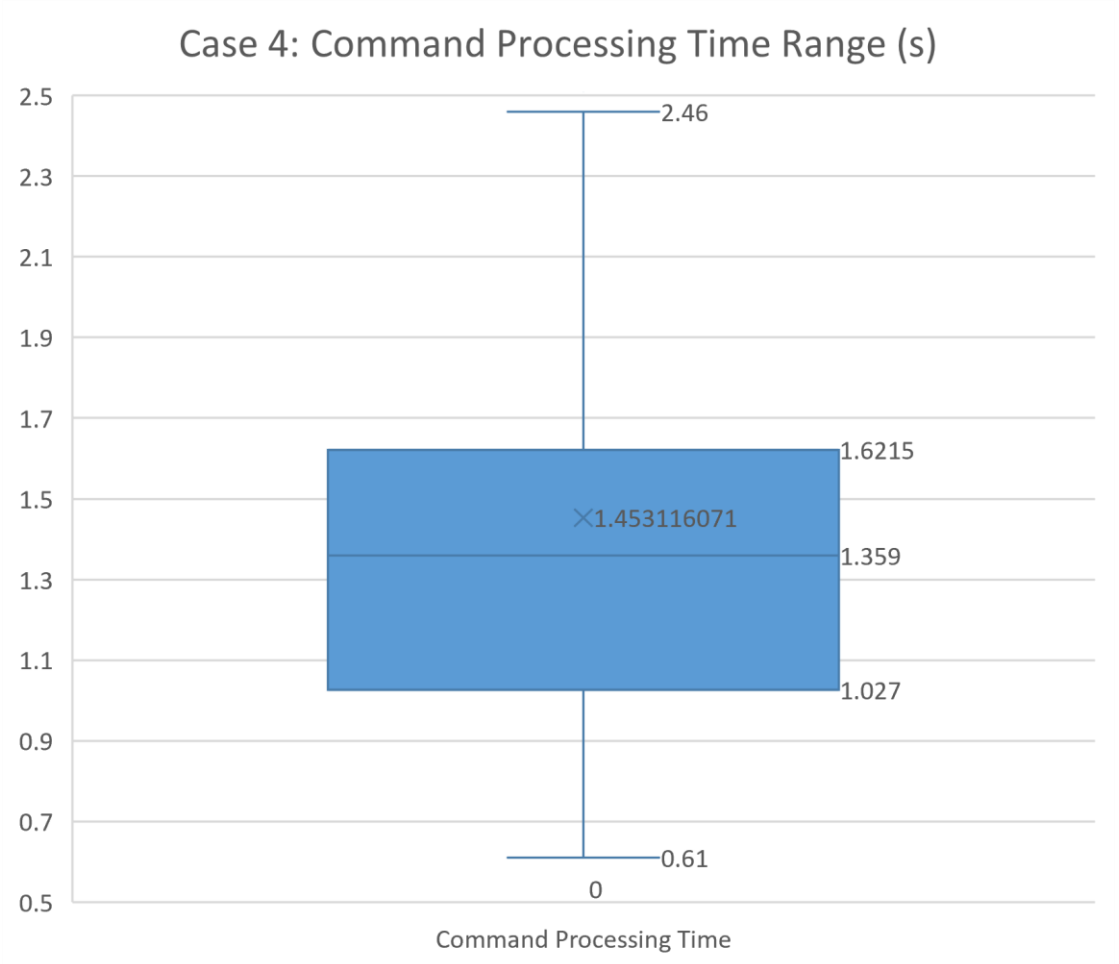


Figure 49: Plot depicting the range of processing times for the controller. The X in the middle represents the mean of the samples.

From this scenario, it is observed that the asynchronous scheduling has an impact on the command processing times. The controller could be in the middle of performing one task, when the next one triggers. Since the system can only do one task at a time, the other task gets placed onto the queue. That is why the controller processing times have increased. Overall, the asynchronous scheduling is effective to allow both sensor readings and commands to be issued simultaneously. Based on the results from Scenarios 2-4, the SEC needs to be improved and further investigated; in particular, the

UART communications between the Jetson Nano and JNXB3. Figure 50 visualizes the flow of information when issuing a command and the respective times for each sequence.

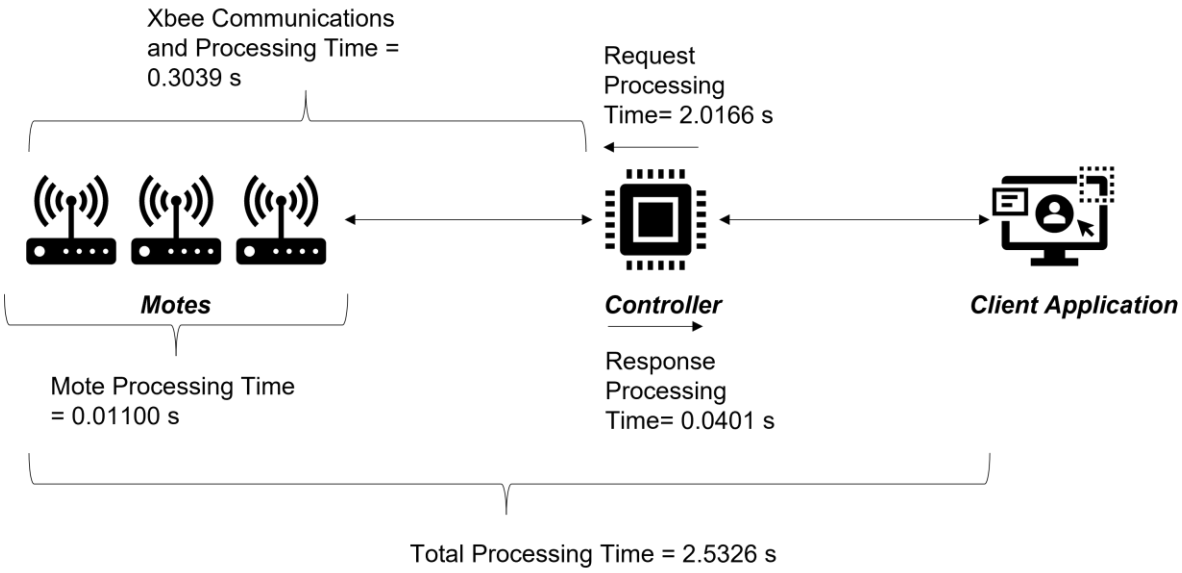


Figure 50: Visual depiction of the time it takes for each part of the system to complete. Note that the numbers used here represent the 95th percentile.

5.5 Scenario 5: The Neural Network Performance

The performance of the ANN was measured through Mean Absolute Error (MAE) and the 95th percentile of the absolute errors. The ANN made predictions on data collected from 2016. The data was filtered and processed to remove negative power; this does not make sense in the context of measuring energy produced. The processing produced 7756 entries. It should be noted that this data was never used for training or validation. It can be seen in Table 15 that the MAE is roughly 6.24. While the 95th percentile roughly 16. Note for the analysis, only entries with non-zero solar PV generation were considered; the predictions for zero PV power were always 0.0018. Removing these points left us with 4103 entries. It is important to make sure that the system is correctly predicting non -power producing hours. However, inclusion of these entries in the analysis skews the accuracy because then almost half of the entries are 0. Thus, having the same low error predictions will cause the accuracy to be much higher. The mean absolute error, including the entries with zero PV power production, was 4.44 kW or 1.7%; this justifies the removal of these points in the analysis.

Table 15: The metrics used to evaluate the ANN performance

Metric	Result (kW)	Result (%)
Mean Absolute Error	6.24	2.5
95^{th} Percentile of the Absolute Errors	16.91557148	6.4

Thus, on average, the Neural Network's prediction is off by 6 kW. In the context of the data provided, the system can produce up to 250 kW, thus that error is less than 2.5%. The 95^{th} percentile indicates that the largest upper bound for error is 16 kW which is more significant. Thus, the upper bound for error is about 6.4%. Even during high production days, 16 kW is a large quantity of energy. However, these metrics give a rough idea of the performance.

To better understand its performance, it is vital to examine its results compared to actual data. Looking at Figure 51 to Figure 54, these give samples of the performance of the ANN. Each day was chosen from a month in each season. It is observed that each prediction quite close to the actual result.

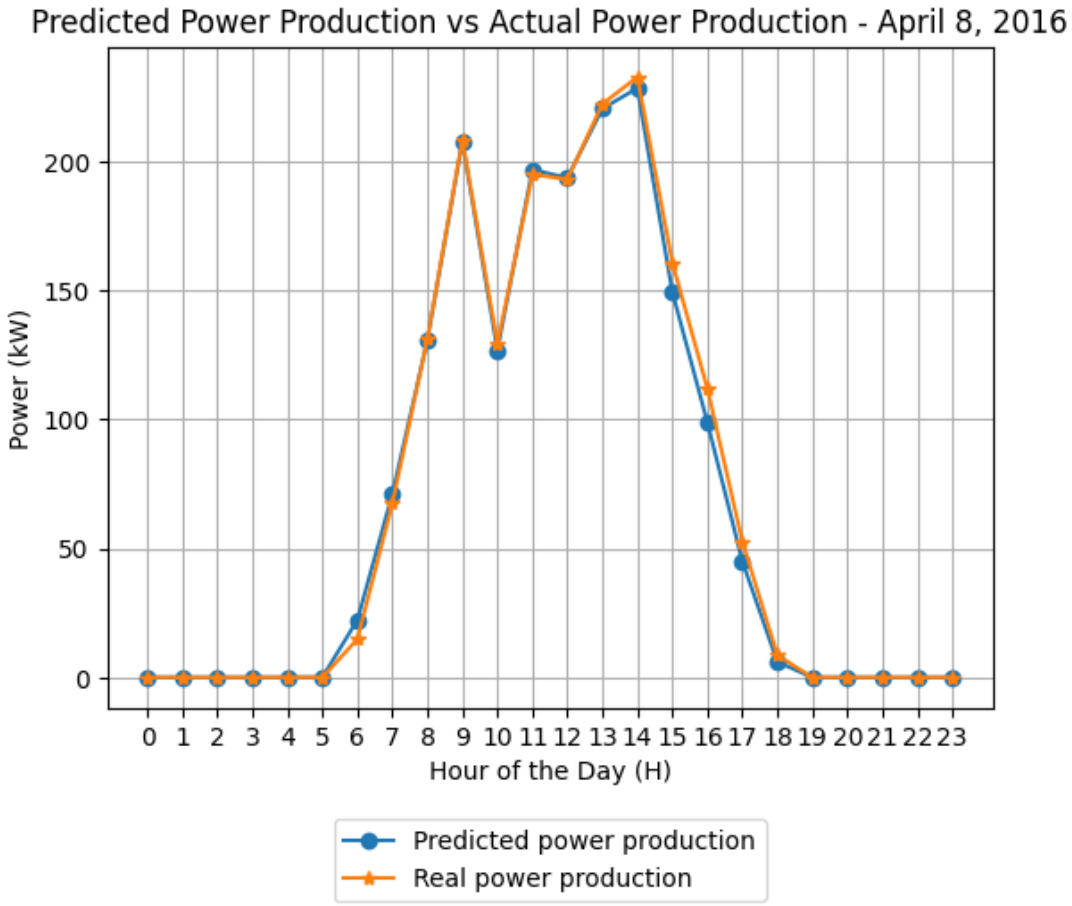


Figure 51: The predicted power vs actual power for data from April 2016

The first prediction is for April 8th. Figure 51 shows that Neural Network can use the inputs to get an accurate prediction even if the production is irregular. The drop at the 10th hour is likely due to shading because it goes back up the next hour.

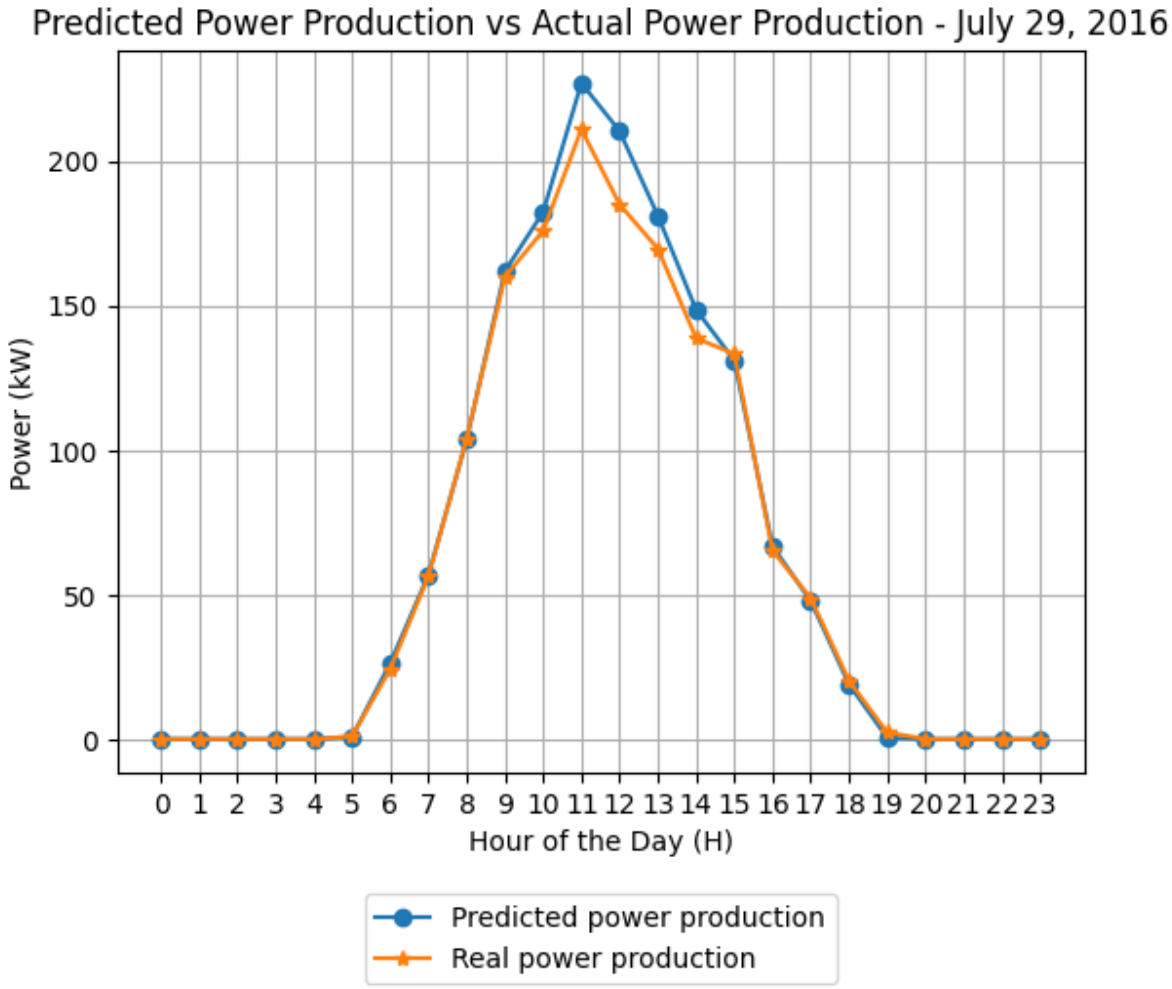


Figure 52: The predicted power vs actual power for data from July 2016

Figure 52 shows the prediction for a day in July. For this one we observe that the predictions are noticeably higher between the 11th-14th hours in the day. Examining the data set reveals the temperature during those times are the highest and reached 30 degrees Celsius. As we know higher temperatures decrease the efficiency of the solar panels. For comparison, the 13th hour on April 8th, 2016, has a solar irradiance of 800 W/m^2, but the panels produce 220 kW. While the temperature is about 8.8 °C. Looking at the 12th hour on July 29th, 2016, it produces 185 kW, and the solar irradiance is 880 W/m^2. However, the temperature is 29.5 °C. Thus, the high temperatures could be affecting the power output despite there being more solar irradiance. The model may have a bias and puts too much weight on the solar irradiance.

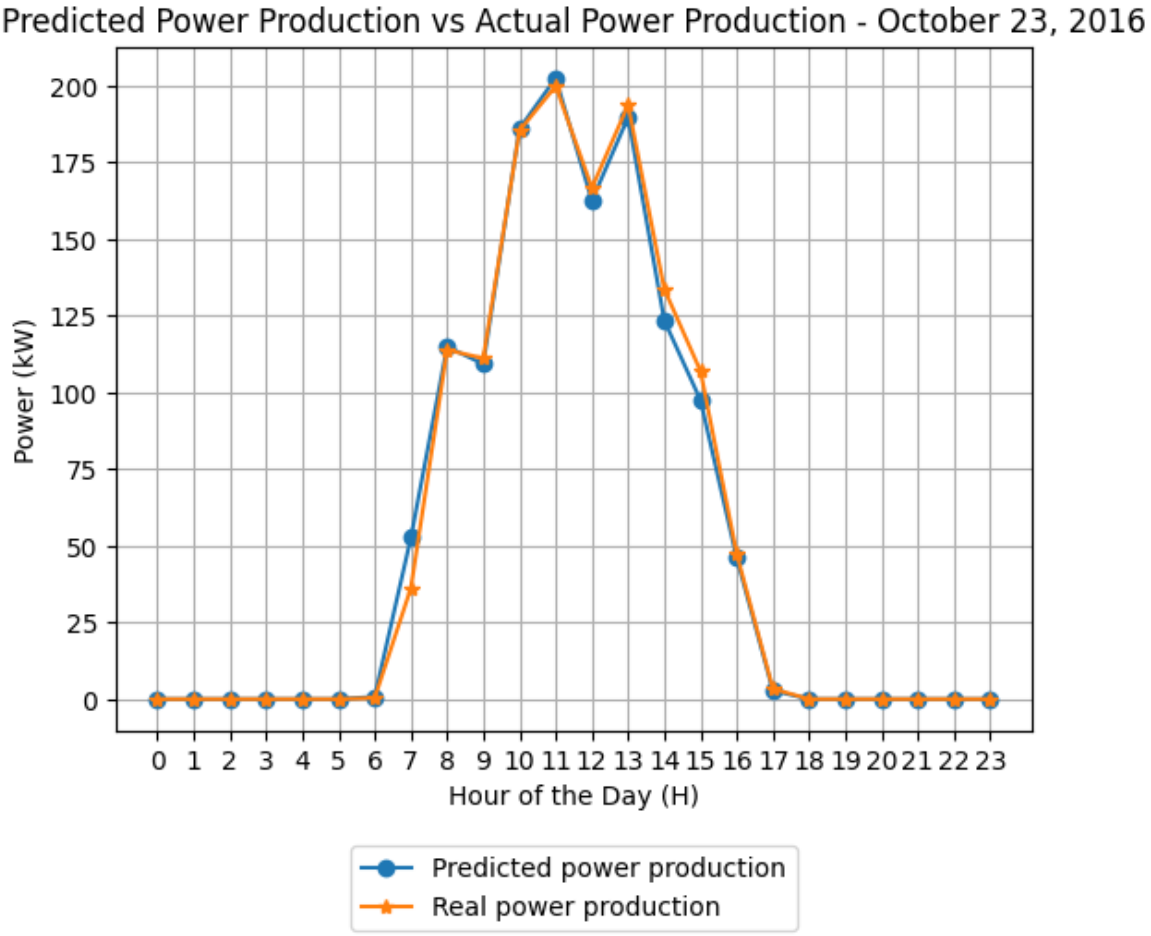


Figure 53: The predicted power vs actual power for data from October 2016

Figure 53 shows the prediction for a day in October. Like April it is still able to accurately predict the irregularities. The only point where it does not do as well is at 7 AM where the difference is almost 20 kW. When examining the dataset at the 7th hour, the solar irradiance is 181 W/m^2. However, at the 16th hour the solar irradiance is at 150 W/m^2, but the panels generate 10 kW more energy. It was noticed that the humidity is the biggest difference. It is almost 60% humidity at the 7th hour, whereas at the 16th hour it is 34% humidity. A high humidity can cause a build of moisture on the panel surface which reflects some of the sunlight. Thus, this will decrease the energy production. Again, this overestimation may be caused by the NN placing too much weight on the solar irradiance.

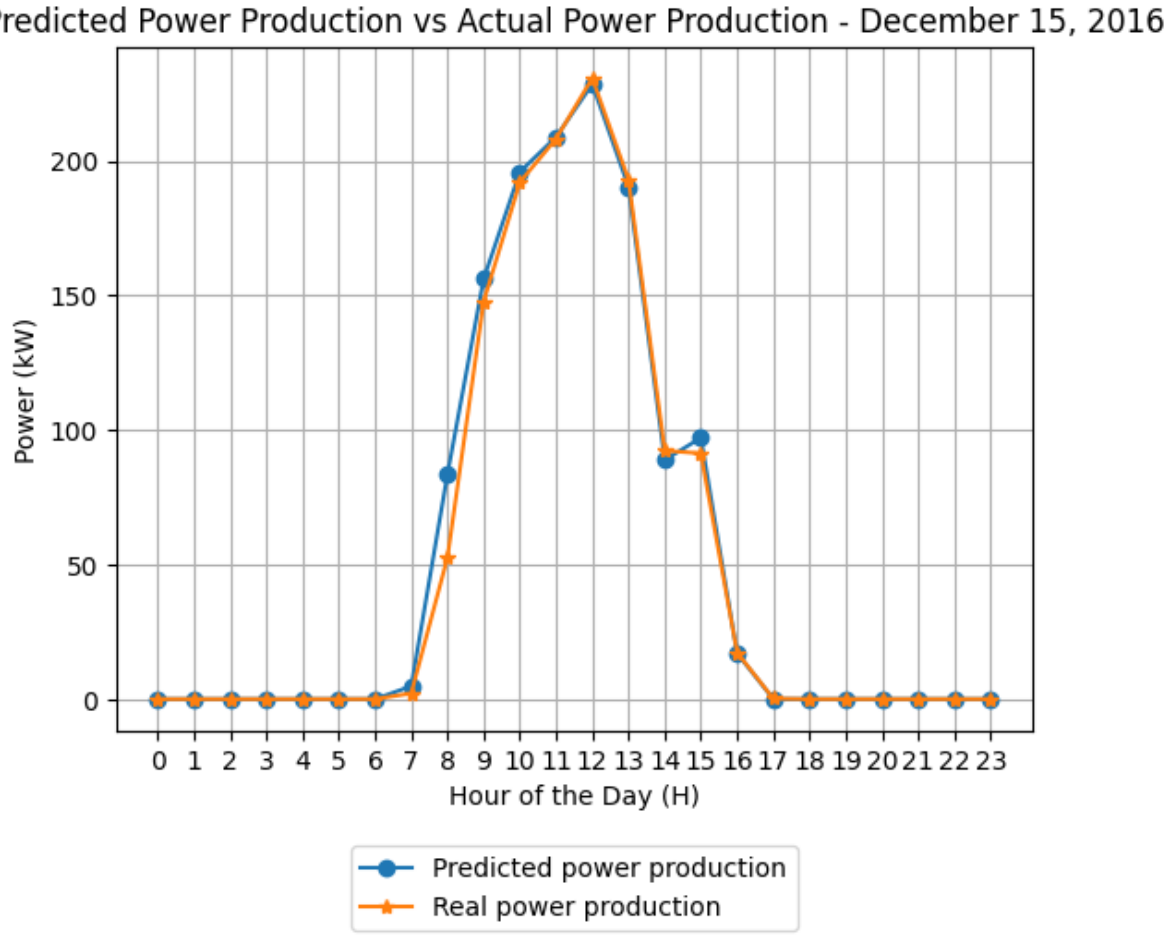


Figure 54: The predicted power vs actual power for data from December 2016

Finally, Figure 54 shows the prediction for a day in December. The prediction is accurate in almost every hour. The only point where it does not do as well is at 8 AM where the prediction nearly double that of the actual value. In the data that hour would be considered an anomaly because the average power generated is quite low, however the solar irradiance is 263 W/m^2. This doesn't scale properly when compared to the other data; at the 14th hour, the solar irradiance is W/m^2. Both have similar solar irradiances but the power production at the 14th hour is nearly double that; 92 kW versus 52 kW. All the other input data is approximately the same. It is important to remember that this is at 8 AM. Thus, there may have been snow which had not been cleared. Snow can cause shading which reduces the efficiency. If we look at 9 AM the produced power is nearly triple that of the power at 8 AM. Obviously, the sun is higher up but that alone should not cause a three times increase in produced

power. Also, this data comes from NIST, which is an office. Presumably the work starts at 9 AM, which means the snow may have not been cleared until close to 9 AM. To further understand the neural network performance, a Bland-Altman plot can be constructed.

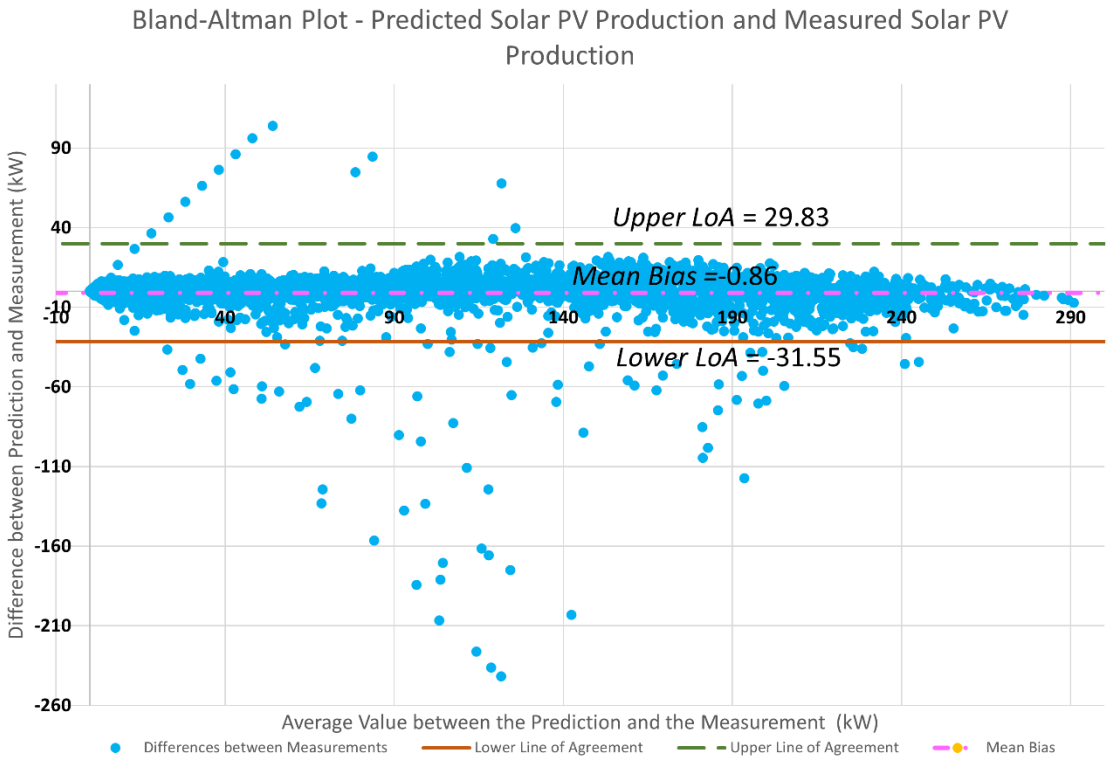


Figure 55: Bland-Altman plot comparing the Neural Network's predictions vs the actual measured values.

A Bland-Altman plot was constructed to understand any systemic biases. Figure 55 shows the Bland-Altman plot that was made from the measurements and the neural network predictions. The mean bias is -0.86, which indicates the system tends to slightly underestimate the actual power generation. Also, the magnitude of the mean bias is less than 1. Thus, the neural network has high accuracy when predicting the PV power generation over a wide range of values. Furthermore, there were only 89 data points that were outside the lines of agreement; these outliers correspond to 2.17% of the total predictions. This is a good indication of consistency as it means 97.83% of the data points are within the lines of agreements.

The upper line of agreement (LoA) is at 29.83, while the lower line of agreement is at -31.55. The magnitudes of the LoAs are approximately equal; this indicates the system is equally likely to overestimate or underestimate power production. These LoAs do indicate that the error can be up to 30 kW, which is significant in the context of low power production hours.

It also appears the system does have some points where it severely underestimated the power generated. Some of the magnitudes of the differences reach into the hundreds, however given the low occurrence, these data may have had external factors not accounted for; for example, the shading previously described. The low mean bias, equal LoA magnitudes, and low outlier values indicate the model is consistent in its performance. However, the lines of agreement should ideally be decreased to about 16 kW to fall in line with the 95^{th} percentile value in Table 15. The model's individual performance has been examined; however, its performance should be compared to other models in the literature. This will indicate whether the computed metrics are consistent with the rest of the developed models.

Table 16: A performance comparison of various Neural Networks designed to forecast solar PV power prediction

Reference	Type of Neural Network	Solar PV System Rating	Prediction Granularity	Root Mean Squared Error (RMSE)	Normalized Root Mean Squared Error (nRMSE)
[118]	Back Propagation Neural Network	36 kW	Hourly	4.24 kW	11.8 %
[119]	Back Propagation Neural Network	5 kW	Hourly	0.48 kW	9.5%
[120]	Bayesian Regularized	231 kW	Hourly	19.03 kW	8.2%

	Neural Network				
[121]	Back Propagation Neural Network	45.6 kW	Hourly	3.04	6.7%
[122]	Back Propagation Neural Network	10 MW	15 minutes	0.57 MW	5.7%
Thesis	Back Propagation Neural Network	250 kW	Hourly	15.68 kW	6.2%

Table 16 compares the performance of the model in the thesis to similar models in the literature. The idea was to pick models with similar prediction granularity and neural network architecture. Most importantly, the papers needed to calculate root mean squared error (RMSE) of their model. The main problem with comparing the predictions is that the PV system ratings are different; the same magnitude of error may be more significant for a system with a lower power rating. Thus, to make a more accurate comparison, the RMSE was normalized by the system's rated PV power. This scales the error based on the magnitude of the rated power. The new metric is called normalized Root Mean Squared Error (nRMSE) [118]. Equation (3) shows how the nRMSE is calculated [118].

$$\boldsymbol{normalized\ Root\ Mean\ Squared\ Error} = \frac{1}{P_{rated}} \times \sqrt{\frac{\sum_{i=1}^{N}(T_i - P_i)^2}{N}} \quad (3)$$

Where P_{rated} is the rated power of the solar PV system; N is the total number of data entries; T_i is the true measurement from the dataset; P_i is the predicted value. The model in this thesis is consistent with the nRMSEs from the literature selection. The nRMSEs range from 5.7 – 11.8%. Furthermore, the closest comparable model is from reference [120]. The model used is a back propagation neural

network, the solar PV system rating is 231 kW, and the prediction granularity is also hourly. Its nRMSE is 8.2%, while the model in this thesis is 6.2%. While the performance information is important, the model must be implemented to make use of the predictions.

As described in Chapter 3, the NN model predictions can be integrated in the asynchronous control loop along with the rest of the code. Figure 56 shows the Jetson Nano terminal output when the model is loaded and makes a prediction.

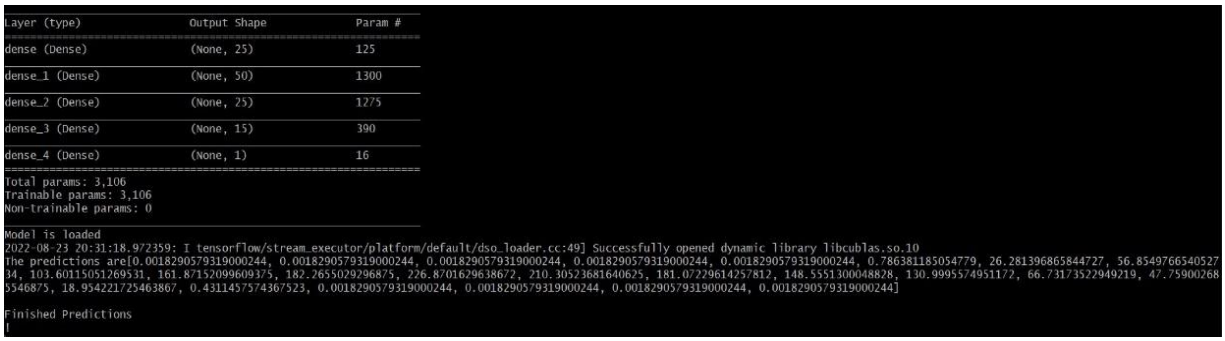


Figure 56: The program loading the model and making the predictions as seen in the Jetson Nano terminal.

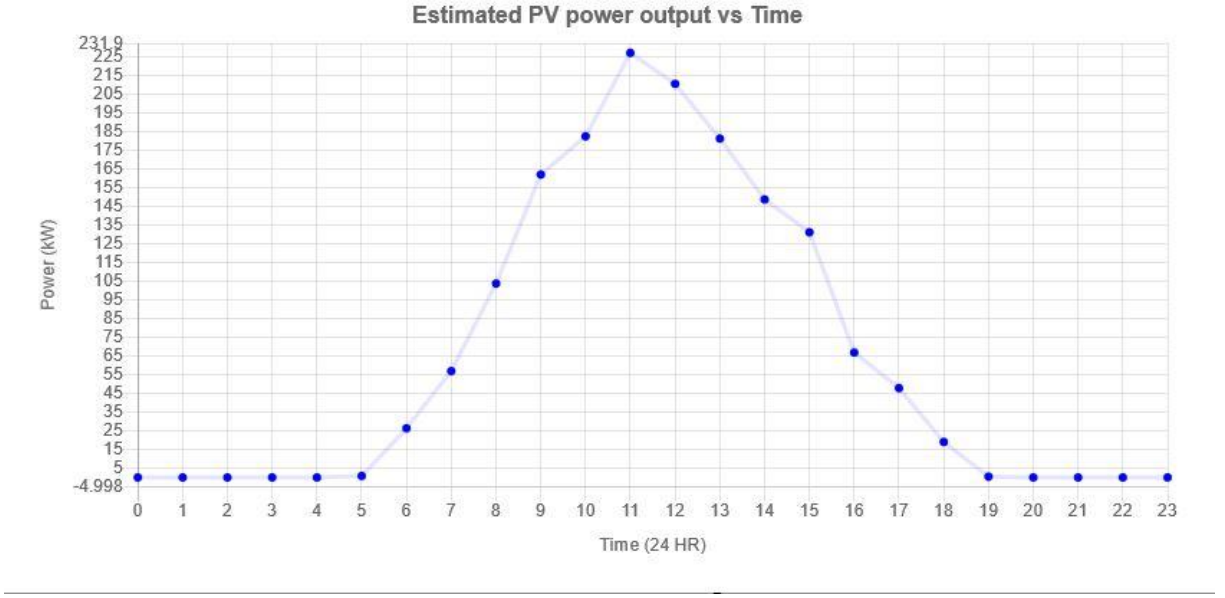


Figure 57: The predicted power as seen in the dashboard.

For real time predictions, there are weather APIs available. One such example is the OpenWeather API, which features Python support. OpenWeather also has a Solar Radiation API that provides the Global Horizontal Irradiance (GHI), Direct Normal Irradiance (DNI), and Diffuse Horizontal Irradiance (DHI). Using these measurements, the solar irradiance on the plane of the array can be calculated using the equations. Equations (4) - (6) show formulas to calculate the solar insolation on the plane of the array on a fixed solar PV panel [123].

$$\boldsymbol{Direct\ Irradiance\ on\ titled\ surface\ =\ DNI\ \times\ \frac{cos\,\theta}{sin\,\alpha}}\ \textbf{(4)}$$

$$\boldsymbol{Diffuse\ Irradiance\ on\ titled\ surface\ =\ DHI\ \times\ \frac{1+cos\,\tau}{2}}\ \textbf{(5)}$$

$$\boldsymbol{Reflected\ Irradiance\ on\ titled\ surface\ =\ HI\ \times\ \rho\times\frac{1-cos\,\tau}{2}}\ \textbf{(6)}$$

Where θ is the solar incident angle; sin α is the solar altitude; τ is the array tilt angle; ρ is the ground reflection coefficient. Note that these equations calculate the solar insolation (Wh/m^2), however since the calculation of the insolation is for a 1-hour time interval, the hour gets cancelled out and it just leaves (W/m^2); which is the solar irradiance.

Therefore, from the performance metrics and literature comparison, this neural network model provides a satisfactory performance for predicting power from solar PV generation system. Improving the model would involve reducing its reliance on the solar irradiance. Also introducing other features that affect solar panel generation, such as precipitation and cloud cover index, can improve the accuracy.

5.6 Scenario 6: Cost Evaluation

The objectives of this scenario are:

a) To determine the total cost of the proposed EMS.

b) Compare the cost of the proposed EMS to existing EMS solutions

Table 17: Cost of major components in the EMS developed in this Thesis.

Component	Quantity	Price/Unit (CAD)	Total Cost (CAD)
Jetson Nano	1	$103.29	$103.29
Digi Xbee 3, 2.4 Ghz Zigbee 3.0, U.FL Ant, SMT	4	$38.84	$155.36
G5LE Relay	3	$2.00	$6
Gravity I^2C Wattmeter	3	$10.16	$30.48
1100 mAh LiPO Battery	3	$18.07	$54.21
Total EMS Cost			$349.34

As seen from Table 17, this EMS costs approximately $349.34. However, before comparing this price to the other systems there are some factors to consider. As described in Section 1.3, the prices listed on the websites include more than just the cost of the materials. Using reference [124], we can get some examples of factors that influence the product cost:

1. Material Cost – The cost of procuring the materials needed to construct the product
2. Labour Cost – The cost of paying the personnel involved in the creation of the product
    a. Personnel directly involved in the manufacturing (e.g. factory workers)
    b. Personnel indirectly involved in the manufacturing (e.g. office workers)
3. Factory Overhead – The cost of operating the factory that assembles the product
    a. Utilities
    b. Mortgages

4. Equipment – The cost of using the equipment to produce an instance of the product
5. Warranty – A manufacturer guarantee stating that they will fix or replace products for a certain period under certain conditions. The cost of this service includes:
    a. Customer support
    b. Cost of repairs or replacements
6. Software development and support costs – Includes designing, developing, and maintaining the software for the system
7. Obsolescence management – Accounting for the loss of original manufacturer's, suppliers, or raw materials
8. Testing: Cost involves determining products standards, identifying defects, and fixing defects.
9. Burn in cost – Units are typically stress tested before being put in service. Used to weed out the products that would fail early. The cost includes:
    a. Developing the tests
    b. Performing the tests
    c. Cost of failed units during the test
    d. Opportunity cost of the failed units
10. Certification: Products need to be certified by a third party before they can be approved for release. Some groups that perform certification are:
    a. Canadian Standards Association (CSA)
    b. Underwriters Laboratory (UL)

Note that these factors are not an exhaustive list, but rather to demonstrate that the pricing listed includes many costs that are directly observed. Additionally, these costs can be estimated and modelled before the product's final release. These are factors that can be accounted for when pricing a product; not necessarily all of them or the only things. Furthermore, the pricing may also be influenced by the pricing strategy of the manufacturer. Additionally, prices may also include distributor costs as well. Which also include facility overhead.

In this thesis, only the materials cost is considered, thus a comparison based on the materials will be performed. A cost-plus pricing model will be used in this example. We can use approximations to get a better idea of the price. From reference [125], they state that the material cost can be 72% of the total product cost. However, the manufacturer markup must be considered as well. Reference [126]

gives mark up estimates for various manufacturing industries. In this case the mark up was chosen from the "Electrical Apparatus" industry in Canada. Note that these numbers were calculated from data from the 1970 – 1992. This information was from the most credible source available; the paper was published in the OECD Journal for Economic Studies. An approximate material cost can be calculated by equations (7) and (8).

$$\boldsymbol{Manufacturer\ Price = 1.14 * Total\ Product\ Cost} \quad \textbf{(7)}$$

$$\boldsymbol{Material\ Cost = 0.72 * Total\ Product\ Cost} \quad \textbf{(8)}$$

The cheapest system is the Enphase system; the listed price comes from the manufacturer's website. Using these equations, we get the material cost of the Enphase system to be $1911.16 CAD. The Enphase system is roughly 5.5 times the cost of the EMS in this thesis. However, these are estimates to demonstrate that price reflected and should not be taken a face value. Thus, the pricing of the EMS presented would change when accounting for the costs mentioned earlier. For a more accurate comparison, I would essentially have to open a business to manufacture the EMS, then calculate the cost.

Table 18: The cost of each energy management system after adjusting for the mark up and material cost

Manufacturer	Cost (CAD)
Enphase	$1911.16
Tesla	$12 583.58
SolarEdge	$2791.58 - $3470.15
Thesis EMS	$349.34

It shall be acknowledged that the Enphase, Tesla, and SolarEdge energy management systems are integrated into core components such as inverters. Thus, the pricing of these systems includes more than the EMS. However, the EMS in this thesis can be retrofitted onto a system without having to change or remove core components in a solar PV system. Buying, removing, and re-installing core components would almost certainly be more expensive than just installing on new parts.

The cost of the EMS displayed here may change depending on the rated power of the solar PV system and other stakeholder requirements. For example, a solar PV system with a 240 V solar panel would need sensors and relays that are appropriately rated. The only confirmed costs in this system are the Jetson Nano and Digi Xbee 3s. Note, that items whose price was less than a dollar, such as the wires, are considered negligible. To make an EMS affordable, the system shall be built from off the commercially available shelf parts. Typically, these off the shelf parts are designed for the average consumer; thus, are usually more affordable. Furthermore, off the shelf parts allow for rapid prototyping because they do not need to be manufactured every time they are bought. As discussed, having custom hardware made by another company should be avoided. It will likely drive-up costs and prevent rapid prototyping. The exception to this would be to homogenizing an existing design. In this case, the EMS would benefit from having a custom Printed Circuit Board (PCB) integrating all the components, instead of having parts loosely connected. The PCB would also make the design more compact and robust as the parts are soldered on.

5.7 Summary

In this chapter, six scenarios were presented to prove the feasibility and effectiveness of the proposed EMS. The first case validated the system operation. It verified the operation of the SEC, Motes, and client application. The second case only tested and evaluated the reliability of the sensor data collection. The test involved taking 500 sensor samples. The results showed a consistent pattern from the sensors. The third case only tested and evaluated the execution times for commands sent from the web application. The test involved executing 500 commands. The results showed that system could generally execute commands and respond within a range of 0.9 to 1.4 seconds. The longest part of executing the command is when the SEC receives and process the request. Thus, an improvement to this section of the code can reduce total execution time.

Scenario No.4 evaluated both the sensor data collection and the execution times for command simultaneously. The results demonstrated that the reliability of the sensor data collection was not majorly affected by the concurrent task execution. The execution times for the commands did increase; they generally fell in the range between 1 to 1.6 seconds. Scenario No.5 tested the integration of the neural network and the performance of its predictions. The performance was tested on a data taken from a 250-kW solar PV energy system in Gaithersburg, Maryland USA. The mean

absolute error of its predictions was 6 kW, while most of the errors fell within the magnitude of 16 kW. The final test case evaluated the cost of the EMS and compared it to exiting EMS solutions. The results indicated that the proposed EMS is much cheaper than the existing EMS systems. The proposed EMS costs 18 times less than the next cheapest system. The results demonstrate the prototype is operational.

Chapter 6
Future Research

This thesis is focused on the research and development of a hardware and software infrastructure for a Solar PV energy management system. It provides a suitable base for future research in energy management systems. The future research involves:

- Modifying and testing the system for different energy resources
- Creating a Printed Circuit Board (PCB) for the Motes and SEC
- Testing the system using an energy scheduling algorithm
- Improving the code and the UART communications in the Smart Energy Controller to be faster and more reliable.

Chapter 7
Conclusion

This thesis presents an innovative Energy Management System infrastructure for a solar PV system. The primary components of the proposed EMS is a data management infrastructure, a client application, a Smart Energy Controller, and three Motes. The data management infrastructure is primarily made using Amazon Web Services.

AWS is primarily used for data storage and data communications. The SEC sends system logs to DynamoDB where they are stored. Then the client application can access them. While AWS IoT Core enables MQTT communications between the client application and the SEC. AWS Amplify is used to host the client application.

Through the client application the user can observe alarms, changes in system state, predicted solar PV energy production, and the current flow and quantity of energy within the system. Errors can be logged by the SEC and then viewed and addressed in the client application. The user can also control the flow of energy through the client application. The client application sends an MQTT message to the SEC which then sends signals to the Motes which can stop or allow the flow of energy at key points in the system. The SEC also sends a record of the control actions and state changes to the AWS where it can be viewed in the client application

The Monitoring and Control Platform consisted of a SEC and three Motes. The Motes are placed at the output of the PV panels, the input to the batteries, and the input to the load. The SEC communicates with the Motes via ZigBee. The SEC can control the energy flow and collect power measurements using the Motes. Additionally, the SEC predicts daily energy production and updates the prediction every hour. The system can reliably and simultaneously collect data and execute commands. The MCP can receive and execute commands within 1-1.6 seconds.

The main contribution of this thesis is an innovative hardware and software infrastructure that can be used to help solar PV integration with main grid It provides a framework that can be modified to host different energy scheduling and energy prediction algorithms. This also helps energy management system integration by giving an underlying EMS infrastructure so researchers can focus on developing the algorithms to predict and schedule energy. The proposed system also costs much less than some of the brand name systems on the market. It costs nearly 18 times less than the Enphase EMS, which is the next cheapest option. The EMS can also be retrofitted to existing solar PV systems

without having to change core components. The Motes and the SEC communicate wirelessly via Zigbee; the components have no physical attachment to each other. Any part replacement is independent of other components in the EMS and in the energy system. Therefore, independence between components can make an EMS easy to integrate.

Future research for the EMS includes automated energy scheduling, compatibility with different energy resources, and developing PCBs for the SEC and Motes. In conclusion, this thesis provides an innovative EMS infrastructure that is manufacturer independent, affordable, and easy to integrate.

References

[1] Statista, “Global renewable energy industry,” 2019.

[2] Statista, “Energy supply -ISIC D- : Statista Industry Report – Global,” 2020. doi: 10.1787/factbook-2013-41-en.

[3] D. S. Chai, J. Z. Wen, and J. Nathwani, “Simulation of cogeneration within the concept of smart energy networks,” *Energy Convers Manag*, vol. 75, pp. 453–465, Nov. 2013, doi: 10.1016/j.enconman.2013.06.045.

[4] D. S. Chai, J. Wen, J. Nathwani, I. Rowlands, and W. I. for S. E. U. of Waterloo, “Concept Development for Smart Energy Networks,” 2011.

[5] Statista, “Solar energy industry worldwide,” 2020.

[6] Enphase Energy Incorporated, “IQ Gateway | Enphase Communication,” 2022. https://enphase.com/store/communication/iq-gateway (accessed Sep. 30, 2022).

[7] Enphase Energy Incorporated, “IQ System Controller | Enphase Storage,” 2022. https://enphase.com/store/storage/iq-system-controller

[8] Enphase Energy Incorporated, “Planning an Enphase Energy System,” 2022.

[9] Enphase Energy, “IQ System Controller 2 | Enphase Storage.” https://enphase.com/store/storage/iq-system-controller-2 (accessed Jun. 30, 2022).

[10] Enphase Energy Incorporated, “Load and Solar Circuit Control using IQ System Controller Auxiliary Contacts Contents,” 2021.

[11] T. Inc., “Request a Quote | Tesla.” https://www.tesla.com/en_au/powerwall/design (accessed Jun. 28, 2022).

[12] EcoDirect and SolarEdge Technologies Incorporated, “SolarEdge Energy Hub 3.8kW 240 Volt AC Single Phase Energy Hub HD-Wave Inverter with Prism Technology - SE3800H-USSNBBL14,” 2022. https://www.ecodirect.com/SolarEdge-Energy-Hub-Inverter-w-Prism-Technology-p/solaredge-se3800h-ussnbbl14.htm (accessed Sep. 30, 2022).

[13] EcoDirect and SolarEdge Technologies Incorporated, “SolarEdge Energy Hub Inverters | SolarEdge Solar Inverters | EcoDirect.com,” 2022. https://www.ecodirect.com/SolarEdge-Energy-Hub-Inverters-s/1144.htm (accessed Jun. 29, 2022).

[14] SolarEdge Technologies Incorporated, “Storage Estimator - SolarEdge Energy Bank | SolarEdge US,” 2022. https://www.solaredge.com/us/products/pv-inverter/the-solaredge-energy-hub (accessed Sep. 30, 2022).

[15] SolarEdge Technologies Incorporated, “Home Is Where the Smart Is - SolarEdge.” 2020. [Online]. Available: https://www.solaredge.com/sites/default/files/home-is-where-the-smart-is-usa.pdf

[16] M. C. J. Manyika, M. Miremadi, N. Henke, R. Chung, P. Nel, and S. Malhotra, “NOTES FROM THE AI FRONTIER: INSIGHTS FROM HUNDREDS OF USE CASES,” 2018.

[17] S. Witherspoo and W. Fadrhonc, “Machine learning can boost the value of wind energy,” *Google Blog - AI*, 2019. https://blog.google/technology/ai/machine-learning-can-boost-value-wind-energy/ (accessed May 13, 2022).

[18] IRENA, *Innovation landscape for a renewable-powered future: Solutions to integrate variable renewables*. 2019.

[19] Stanford University, “Fuzzy Logic (Stanford Encyclopedia of Philosophy).” https://plato.stanford.edu/entries/logic-fuzzy/ (accessed Mar. 03, 2022).

[20] P. Kofinas, A. I. Dounis, and G. A. Vouros, “Fuzzy Q-Learning for multi-agent decentralized energy management in microgrids,” *Appl Energy*, vol. 219, no. December 2017, pp. 53–67, 2018, doi: 10.1016/j.apenergy.2018.03.017.

[21] T. Mitiku and M. S. Manshahia, “Neuro Fuzzy Inference Approach : A Survey,” vol. 4, no. April, pp. 505–519, 2018.

[22] F. Alfaverh, M. Denai, and Y. Sun, “Demand Response Strategy Based on Reinforcement Learning and Fuzzy Reasoning for Home Energy Management,” *IEEE Access*, vol. 8, pp. 39310–39321, 2020, doi: 10.1109/ACCESS.2020.2974286.

[23] S. Al-Sakkaf, M. Kassas, M. Khalid, and M. A. Abido, “An Energy Management System for Residential Autonomous DC Microgrid Using Optimized Fuzzy Logic Controller Considering Economic Dispatch,” *Energies 2019, Vol. 12, Page 1457*, vol. 12, no. 8, p. 1457, Apr. 2019, doi: 10.3390/EN12081457.

[24] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Process Mag*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.

[25] E. Foruzan, L. K. Soh, and S. Asgarpoor, “Reinforcement Learning Approach for Optimal Distributed Energy Management in a Microgrid,” *IEEE Transactions on Power Systems*, vol. 33, no. 5, pp. 5749–5758, 2018, doi: 10.1109/TPWRS.2018.2823641.

[26] B. V. Mbuwir, F. Ruelens, F. Spiessens, and G. Deconinck, “Battery energy management in a microgrid using batch reinforcement learning,” *Energies (Basel)*, vol. 10, no. 11, pp. 1–19, 2017, doi: 10.3390/en10111846.

[27] Y. Ji, J. Wang, J. Xu, X. Fang, and H. Zhang, “Real-time energy management of a microgrid using deep reinforcement learning,” *Energies (Basel)*, vol. 12, no. 12, 2019, doi: 10.3390/en12122291.

[28] S. Nikolovski, H. R. Baghaee, and D. Mlakic, “ANFIS-Based Peak Power Shaving/Curtailment in Microgrids Including PV Units and BESSs,” *Energies (Basel)*, vol. 11, no. 11, Nov. 2018, doi: 10.3390/en11112953.

[29] H. Zhou *et al.*, “LSTM-based Energy Management for Electric Vehicle Charging in Commercial-building Prosumers,” *Journal of Modern Power Systems and Clean Energy*, vol. 9, no. 5, pp. 1205–1216, 2021, doi: 10.35833/MPCE.2020.000501.

[30] W. Hao, W. Yizhou, L. Yaqin, and S. Zhili, “The Role of Activation Function in CNN,” *Proceedings - 2020 2nd International Conference on Information Technology and Computer Application, ITCA 2020*, pp. 429–432, Dec. 2020, doi: 10.1109/ITCA52113.2020.00096.

[31] P. Marimuthu, “How Activation Functions Work in Deep Learning - KDnuggets,” *KDnuggets*, 2022. https://www.kdnuggets.com/2022/06/activation-functions-work-deep-learning.html (accessed Jun. 24, 2022).

[32] A. L. Klingler and L. Teichtmann, “Impacts of a forecast-based operation strategy for grid-connected PV storage systems on profitability and the energy system,” *Solar Energy*, vol. 158, pp. 861–868, 2017, doi: 10.1016/j.solener.2017.10.052.

[33] S. Mujeeb, T. A. Alghamdi, S. Ullah, A. Fatima, N. Javaid, and T. Saba, “Exploiting deep learning for wind power forecasting based on big data analytics,” *Applied Sciences (Switzerland)*, vol. 9, no. 20, 2019, doi: 10.3390/app9204417.

[34] M. Kim, W. Choi, Y. Jeon, and L. Liu, “A hybrid neural network model for power demand forecasting,” *Energies (Basel)*, vol. 12, no. 5, 2019, doi: 10.3390/en12050931.

[35] H. Zhou, Y. Zhang, L. Yang, Q. Liu, K. Yan, and Y. Du, “Short-Term photovoltaic power forecasting based on long short term memory neural network and attention mechanism,” *IEEE Access*, vol. 7, pp. 78063–78074, 2019, doi: 10.1109/ACCESS.2019.2923006.

[36] H. Lan, C. Zhang, Y. Y. Hong, Y. He, and S. Wen, “Day-ahead spatiotemporal solar irradiation forecasting using frequency-based hybrid principal component analysis and neural network,” *Appl Energy*, vol. 247, no. March, pp. 389–402, 2019, doi: 10.1016/j.apenergy.2019.04.056.

[37] J. Duan, H. Zuo, Y. Bai, J. Duan, M. Chang, and B. Chen, “Short-term wind speed forecasting using recurrent neural networks with error correction,” *Energy*, vol. 217, p. 119397, 2021, doi: 10.1016/j.energy.2020.119397.

[38] M. Sameti, M. A. Jokar, and F. R. Astaraei, “Prediction of solar stirling power generation in smart grid by GA-ANN model,” *International Journal of Computer Applications in Technology*, vol. 55, no. 2, pp. 147–157, 2017, doi: 10.1504/IJCAT.2017.082860.

[39] A. T. Eseye, J. Zhang, and D. Zheng, “Short-term photovoltaic solar power forecasting using a hybrid Wavelet-PSO-SVM model based on SCADA and Meteorological information,” *Renew Energy*, vol. 118, pp. 357–367, 2018, doi: 10.1016/j.renene.2017.11.011.

[40] T. Liu, S. Liu, J. Heng, and Y. Gao, “A new hybrid approach for wind speed forecasting applying Support Vector Machine with ensemble empirical mode decomposition and Cuckoo Search Algorithm,” *Applied Sciences (Switzerland)*, vol. 8, no. 10, 2018, doi: 10.3390/app8101754.

[41] A. Zhang, P. Zhang, and Y. Feng, “Short-term load forecasting for microgrids based on DA-SVM,” *COMPEL - The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 38, no. 1, pp. 68–80, 2019, doi: 10.1108/COMPEL-05-2018-0221.

[42] A. Moradzadeh, S. Zakeri, M. Shoaran, B. Mohammadi-Ivatloo, and F. Mohammadi, "Short-term load forecasting of microgrid via hybrid support vector regression and long short-term memory algorithms," *Sustainability (Switzerland)*, vol. 12, no. 17, 2020, doi: 10.3390/su12177076.

[43] C. Chiñas-Palacios, C. Vargas-Salgado, J. Aguila-Leon, and E. Hurtado-Pérez, "A cascade hybrid PSO feed-forward neural network model of a biomass gasification plant for covering the energy demand in an AC microgrid," *Energy Convers Manag*, vol. 232, no. November 2020, 2021, doi: 10.1016/j.enconman.2021.113896.

[44] Springer, *New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe*. Springer International Publishing AG Switzerland, 2016. doi: 10.1007/978-3-319-21569-3.

[45] Dimensional Research and Qubole, "An In-Depth Look at Big Data Trends and Challenges Results from a Global Survey by Dimensional Research," 2018.

[46] Energy Sector Management Assistance Program, "Energy Analytics for Development - Big Data for Energy Access, Energy Efficiency, and Renewable Energy," 2017. doi: 10.1596/28432.

[47] McKinsey Global Institute, "The Age of Analytics: Competing in a Data-Driven World," *McKinsey & Company, Brussels*, no. December, 2016.

[48] T. C. Redman, "Bad Data Costs the U.S. $3 Trillion Per Year," *Harvard Business Review*, 2016.

[49] Statista, "Industries & Markets: Big data," 2021.

[50] Doug Laney, "3D Data Management: Controlling Data Volume, Velocity, and Variety," *Application Delivery Strategies*, vol. 949, no. February 2001, p. 4, 2001.

[51] R. Patgiri and A. Ahmed, "Big Data : The V ' s of the Game Changer Paradigm," 2016. doi: 10.1109/HPCC-SmartCity-DSS.2016.8.

[52] IBM Cloud Education, "What is Data Mining? | IBM." https://www.ibm.com/cloud/learn/data-mining (accessed Mar. 06, 2022).

[53] K. Ku, “Building electric energy prediction modeling for BEMS using easily obtainable weather factors with Kriging model and data mining,” pp. 739–751, 2018.

[54] C. Dou, Y. Zheng, D. Yue, Z. Zhang, and K. Ma, “Hybrid model for renewable energy and loads prediction based on data mining and variational mode decomposition,” 2018, doi: 10.1049/iet-gtd.2017.1476.

[55] M. Babaei, A. Abazari, M. M. Soleymani, M. Ghafouri, S. M. Muyeen, and M. T. H. Beheshti, “A data-mining based optimal demand response program for smart home with energy storages and electric vehicles,” *J Energy Storage*, vol. 36, no. November 2020, p. 102407, 2021, doi: 10.1016/j.est.2021.102407.

[56] J. Bughin, J. Seong, J. Manyika, M. Chui, and R. Joshi, “Notes from the AI Frontier: Modeling the impact of AI on the world economy,” 2018.

[57] International Energy Agency, “Digitalization and Energy,” 2017.

[58] DNV GL, “Digitalization and the Future of Energy,” p. 50, 2021.

[59] M. Chui, M. Collins, and M. Patel, “The Internet of Things: Catching up to an accelerating opportunity,” 2021. [Online]. Available: https://www.mckinsey.com/~/media/mckinsey/business functions/mckinsey digital/our insights/iot value set to accelerate through 2030 where and how to capture it/the-internet-of-things-catching-up-to-an-accelerating-opportunity-final.pdf?shouldIndex=false

[60] BeeBryte, “BeeBryte | Solution.” https://www.beebryte.com/solution/ (accessed May 25, 2022).

[61] BeeBryte, “BeeBryte | References.” https://www.beebryte.com/references/ (accessed May 25, 2022).

[62] PricewaterhouseCoopers, “Time For Trust: How blockchain will transform business and the economy - PwC.” https://www.pwc.com/timefortrust (accessed May 25, 2022).

[63] Amazon Web Services, “GE Power Case Study – Amazon Web Services (AWS),” 2022. https://aws.amazon.com/solutions/case-studies/ge-power/ (accessed Apr. 01, 2022).

[64] Amazon Web Services and General Electric Renewable Energy, “GE Renewable Energy Case Study | Amazon EKS | AWS,” *Amazon Web Services*, 2021.

https://aws.amazon.com/solutions/case-studies/ge-renewable-energy/ (accessed Mar. 10, 2022).

[65] Mads Almassalkhi, Jeff Frolik, and Paul Hines, “How to Prevent Blackouts by Packetizing the Power Grid - IEEE Spectrum,” *IEEE Spectrum*, 2022.

[66] S. Lazgheb, B. Bayar, M. Belouda, H. Oueslati, and S. Ben Mabrouk, “Raspberry Pi-based smart platform for data acquisition, supervision and management of a hybrid PV/WT/Batteries system,” *Mediterranean Microwave Symposium*, vol. 2019-Octob, pp. 3–6, 2019, doi: 10.1109/MMS48040.2019.9157307.

[67] P. N. Krishna, S. R. Gupta, P. V. Shankaranarayanan, S. Sidharth, and M. Sirphi, “Fuzzy Logic Based Smart Home Energy Management System,” *2018 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018*, pp. 10–14, 2018, doi: 10.1109/ICCCNT.2018.8493744.

[68] M. Ghamari, B. Janko, R. S. Sherratt, W. Harwin, R. Piechockic, and C. Soltanpur, “A Survey on Wireless Body Area Networks for eHealthcare Systems in Residential Environments,” *Sensors (Basel)*, vol. 16, no. 6, Jun. 2016, doi: 10.3390/S16060831.

[69] M. Woolley, “The Bluetooth® Low Energy Primer,” 2022.

[70] Cisco, “802.11ac: The Fifth Generation of Wi-Fi,” *Cisco.Com*, no. 1, pp. 1–20, 2018, [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/wireless/aironet-3600-series/white-paper-c11-713103.pdf

[71] P. Dhillon and H. Sadawarti, “A Review Paper on Zigbee (IEEE 802.15.4) Standard,” *International Journal of Engineering Research & Technology*, vol. 3, no. 4, Apr. 2014, doi: 10.17577/IJERTV3IS040105.

[72] Bluetooth Special Interest Group, “Understanding Bluetooth Range | Bluetooth® Technology Website,” 2022. https://www.bluetooth.com/learn-about-bluetooth/key-attributes/range/ (accessed Sep. 24, 2022).

[73] F. Siddiqui, S. Zeadally, and K. Salah, “Gigabit Wireless Networking with IEEE 802.11ac:Technical Overview and Challenges,” *Journal of Networks*, vol. 10, no. 3, Apr. 2015, doi: 10.4304/jnw.10.3.164-171.

[74] D. Vaquerizo-Hdez, P. Muño, M. D. R-Moreno, and D. F. Barrero, “A low power consumption algorithm for efficient energy consumption in ZigBee motes,” *Sensors (Switzerland)*, vol. 17, no. 10, Oct. 2017, doi: 10.3390/s17102179.

[75] O. O. Kazeem, O. Akintade, L. O. Kehinde, O. O. Akintade, and L. O. Kehinde, “Comparative Study of Communication Interfaces for Sensors and Actuators in the Cloud of Internet of Things,” *International Journal of Internet of Things*, vol. 2017, no. 1, pp. 9–13, 2017, doi: 10.5923/j.ijit.20170601.02.

[76] E. Garcia-Espinosa, O. Longoria-Gandara, I. Pegueros-Lepe, and A. Veloz-Guerrero, “Power consumption analysis of bluetooth low energy commercial products and their implications for IoT applications,” *Electronics (Switzerland)*, vol. 7, no. 12. MDPI AG, Dec. 01, 2018. doi: 10.3390/electronics7120386.

[77] M. O. Demir, G. K. Kurt, and M. Karaca, “An energy consumption model for 802.11ac access points,” in *2014 22nd International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2014*, 2014, pp. 67–71. doi: 10.1109/SOFTCOM.2014.7039072.

[78] Texas Instruments, “A True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications,” 2009.

[79] N. H. N. Hashim and N. A. Ramli, “Smart Building Energy Management using Big Data Analytic Approach,” *MACS 2019 - 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics, Proceedings*, 2019, doi: 10.1109/MACS48846.2019.9024800.

[80] J. Choi, M. Kim, and J. Yoon, “Implementation of the Big Data Management System for demand side energy management,” in *Proceedings - 15th IEEE International Conference on Computer and Information Technology, CIT 2015, 14th IEEE International Conference on Ubiquitous Computing and Communications, IUCC 2015, 13th IEEE International Conference on Dependable, Autonomic and Se*, 2015, pp. 1515–1520. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.227.

[81] A. R. Al-Ali, I. A. Zualkernan, M. Rashid, R. Gupta, and M. Alikarar, “A smart home energy management system using IoT and big data analytics approach,” *IEEE Transactions on Consumer Electronics*, vol. 63, no. 4, pp. 426–434, 2017, doi: 10.1109/TCE.2017.015014.

[82] R. Gupta, A. R. Al-Ali, I. A. Zualkernan, and S. K. Das, “Big Data Energy Management, Analytics and Visualization for Residential Areas,” *IEEE Access*, vol. 8, pp. 156153–156164, 2020, doi: 10.1109/ACCESS.2020.3019331.

[83] M. H. Yaghmaee, M. Moghaddassian, and A. Leon-Garcia, “Autonomous Two-Tier Cloud-Based Demand Side Management Approach with Microgrid,” *IEEE Trans Industr Inform*, vol. 13, no. 3, pp. 1109–1120, 2017, doi: 10.1109/TII.2016.2619070.

[84] E. Harmon, U. Ozgur, M. H. Cintuglu, R. De Azevedo, K. Akkaya, and O. A. Mohammed, “The Internet of Microgrids: A Cloud-Based Framework for Wide Area Networked Microgrids,” *IEEE Trans Industr Inform*, vol. 14, no. 3, pp. 1262–1274, 2018, doi: 10.1109/TII.2017.2785317.

[85] Raspberry Pi Trading Ltd, “Buy a Raspberry Pi 4 Model B – Raspberry Pi.” https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

[86] Raspberry Pi Trading Ltd, “Raspberry Pi 4 Computer Model B- Product Brief.” 2021.

[87] C. Baldus-Jeursen and Y. Poissant, “CanmetENERGY Natural Resources Canada) Wes Johnston (Canadian Solar Industries Association, CanSIA)”.

[88] Amazon Web Services, “AWS: Customer Success Stories: Case Studies, Videos, Podcasts, Innovator stories.” https://aws.amazon.com/solutions/case-studies/?customer-references-cards.sort-by=item.additionalFields.sortDate&customer-references-cards.sort-order=desc&awsf.content-type=*all&awsf.customer-references-location=*all&awsf.customer-references-segment=*all&awsf.customer-references-industry=*all&awsf.customer-references-use-case=*all&awsf.customer-references-tech-category=*all&awsf.customer-references-product=*all (accessed Sep. 21, 2022).

[89] Zigbee Alliance, “Zigbee - Connectivity Standards Alliance.” https://zigbeealliance.org/solution/zigbee/ (accessed Aug. 12, 2021).

[90] Connectivity Standards Alliance, “Innovation & Adoption - Zigbee momentum in 2021 - CSA-IOT,” 2022. https://csa-iot.org/newsroom/innovation-adoption-zigbee-momentum-in-2021/ (accessed Jul. 01, 2022).

[91] Boyd M, Chen T, and Dougherty B, “NIST Campus Photovoltaic (PV) Arrays and Weather Station Data Sets,” *National Institute of Standards and Technology. U.S. Department of Commerce, Washington, D.C.*, 2017. https://pvdata.nist.gov/ (accessed Jul. 02, 2022).

[92] D. B, “Rectified Linear Units (ReLU) in Deep Learning | Kaggle,” 2018. https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook (accessed Jun. 24, 2022).

[93] A. Electronics, “What is a Microcontroller? A Look Inside a Microcontroller | Arrow.com | Arrow.com.” https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller (accessed Jun. 22, 2022).

[94] NVIDIA Corporation, “Buy the Latest Jetson Products | NVIDIA Developer.” https://store.nvidia.com/en-us/jetson/store/?page=1&limit=9&locale=en-us (accessed Jul. 02, 2022).

[95] Raspberry Pi Foundation, “Raspberry Pi 4 Model B specifications – Raspberry Pi.” https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (accessed Jul. 01, 2022).

[96] L. Hattersley, “Raspberry Pi 4, 3A+, Zero W - specs, benchmarks & thermal tests — The MagPi magazine,” *The MagPi magazine*.

[97] Espressif Systems, “ESP32 Series Datasheet,” 2022.

[98] DigiKey and Espressif Systems, “ESP32-WROOM-32D (4MB HIGH TEMP) Espressif Systems | RF/IF and RFID | DigiKey.” https://www.digikey.ca/en/products/detail/espressif-systems/ESP32-WROOM-32D-4MB-HIGH-TEMP/12342929 (accessed Sep. 21, 2022).

[99] Adafruit and Raspberry Pi Foundation, “Raspberry Pi Zero W : ID 3400 : $10.00 : Adafruit Industries, Unique & fun DIY electronics and kits.” https://www.adafruit.com/product/3400 (accessed Sep. 21, 2022).

[100] DigiKey and Raspberry Pi Foundation, “RASPBERRY PI 4B/2GB Raspberry Pi | Embedded Computers | DigiKey.” https://www.digikey.ca/en/products/detail/raspberry-pi/RASPBERRY-PI-4B-2GB/10258782 (accessed Sep. 21, 2022).

[101] Digikey, DFRobot, and NVIDIA Corporation, “NVIDIA Jetson Nano 2 GB | DFR0757 | Development Boards, Kits, Programmers | DigiKey.” https://www.digikey.ca/en/products/detail/dfrobot/DFR0757/13545232 (accessed Sep. 21, 2022).

[102] D. Steinkraus, I. Buck, and P. Y. Simard, “Using GPUs for machine learning algorithms,” *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2005, pp. 1115–1120, 2005, doi: 10.1109/ICDAR.2005.251.

[103] M. Wang *et al.*, “GPU-Accelerated Machine Learning Inference as a Service for Computing in Neutrino Experiments,” *Front Big Data*, vol. 3, p. 48, Jan. 2021, doi: 10.3389/FDATA.2020.604083/BIBTEX.

[104] D. Salvator, “JD Uses NVIDIA AI Platform to Put a Critical Eye on Video | NVIDIA Blog,” 2018. https://blogs.nvidia.com/blog/2018/02/13/jd-ai-video-inferencing/ (accessed Jul. 02, 2022).

[105] NVIDIA Corporation, “MAXIMIZING DATA CENTER PRODUCTIVITY WITH APPLICATION WORKLOAD ANALYSIS,” 2018.

[106] Digi International Inc., “Zigbee 3 RF Module | Digi XBee 3 Zigbee 3.0 | Digi International,” 2022. https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee3-zigbee-3#specifications (accessed Jun. 24, 2022).

[107] Amazon Web Services Inc., “AWS IoT Device SDKs, Mobile SDKs, and AWS IoT Device Client - AWS IoT Core.” https://docs.aws.amazon.com/iot/latest/developerguide/iot-sdks.html (accessed Jul. 02, 2022).

[108] Jonathan Valdez and Jared Becker, “Understanding the I2C Bus,” 2020.

[109] DFRobot, “Gravity: I2C Digital Wattmeter SKU: SEN0291-DFRobot.” https://wiki.dfrobot.com/Gravity%3A I2C Digital Wattmeter SKU%3A SEN0291 (accessed Jul. 01, 2022).

[110] Omron Corporation, “Technical Explanation for Solid-state Relays.”

[111] Eurotherm Inc. and Sure Controls, “Solid state relay selector guide,” 2007. https://www.surecontrols.com/wp-content/uploads/2014/07/HA136716-CII-Issue-2c-continental-1.pdf (accessed Jun. 21, 2022).

[112] OMRON Corporation, “G5LE PCB Power Relay Cubic, Single-pole 10A Power Relay.” 2018.

[113] B. Razavi, *Fundamentals of microelectronics*, 2nd ed. John Wiley & Sons, Inc., 2014.

[114] K. Rose, “Single Pole Triple Throw, Triple Pole Single Throw, SPST, DPDT, etc. How to tell the difference - Spemco Switches,” *Spemco*, 2015. https://spemco.com/blog/single-pole-triple-throw-triple-pole-single-throw-spst-dpdt-etc-how-to-tell-the-difference/ (accessed Jun. 21, 2022).

[115] Power-Sonic Corporation, “TECHNICAL MANUAL SEALED LEAD-ACID BATTERIES.”

[116] I. Baccouche, S. Jemmali, A. Mlayah, B. Manai, and N. E. Ben Amara, “Implementation of an improved Coulomb-counting algorithm based on a piecewise SOC-OCV relationship for SOC estimation of Li-ion battery,” *International Journal of Renewable Energy Research*, vol. 8, no. 1, pp. 178–187, 2018, doi: 10.20508/ijrer.v8i1.6686.g7292.

[117] J. Nielson, *Usability Engineering*, 1st ed. Cambridge, Massachusetts: AP Professional, 1993.

[118] L. A. Fernandez-Jimenez *et al.*, “Short-term power forecasting system for photovoltaic plants,” *Renew Energy*, vol. 44, pp. 311–317, Aug. 2012, doi: 10.1016/J.RENENE.2012.01.108.

[119] A. L. Klingler and L. Teichtmann, “Impacts of a forecast-based operation strategy for grid-connected PV storage systems on profitability and the energy system,” *Solar Energy*, vol. 158, no. October, pp. 861–868, 2017, doi: 10.1016/j.solener.2017.10.052.

[120] S. Al-Dahidi, O. Ayadi, J. Adeeb, and M. Louzazni, “Assessment of Artificial Neural Networks Learning Algorithms and Training Datasets for Solar Photovoltaic Power Production Prediction,” *Front Energy Res*, vol. 7, p. 130, Nov. 2019, doi: 10.3389/FENRG.2019.00130/BIBTEX.

[121] R. Azka, W. Soefian, D. R. Aryani, F. H. Jufri, and A. R. Utomo, “Modelling of photovoltaic system power prediction based on environmental conditions using neural network single and multiple hidden layers,” in *IOP Conference Series: Earth and Environmental Science*, Nov. 2020, vol. 599, no. 1. doi: 10.1088/1755-1315/599/1/012032.

[122] Y. Li, L. Zhou, P. Gao, B. Yang, Y. Han, and C. Lian, “Short-Term Power Generation Forecasting of a Photovoltaic Plant Based on PSO-BP and GA-BP Neural Networks,” *Front Energy Res*, vol. 9, p. 958, Jan. 2022, doi: 10.3389/FENRG.2021.824691/BIBTEX.

[123] S. Wang, “In-Plane Solar Irradiance Calculation for Various Type of PV Arrays,” *Conference Record of the IEEE Photovoltaic Specialists Conference*, pp. 1645–1648, Jun. 2019, doi: 10.1109/PVSC40753.2019.8981245.

[124] P. Sandborn, *Cost Analysis of Electronic Systems*, vol. 1. WORLD SCIENTIFIC, 2013. doi: 10.1142/8353.

[125] M. D. Miles, S. Electric, M. M. Dfma, and V. E. Manager, “The Design Engineer’s Guide for Total Product Cost Using DFA,” 2016. [Online]. Available: https://www.researchgate.net/publication/331234576

[126] J. Oliveira, M. S. Scarpetta, and D. Pilat, “MARK-UP PRICING. MARKET STRUCTURE AND THE BUSINESS CYCLE,” *OECD Economic Studies*, vol. 1996, no. 11, pp. 72–102, 1996.

[127] DigiKey and Digi International, “XB3-24Z8US-J Digi Xbee 3 | RF/IF and RFID | DigiKey.” https://www.digikey.ca/en/products/detail/digi/XB3-24Z8US-J/8130939 (accessed Sep. 21, 2022).

Appendices

Appendix A

Specifications of the Digi Xbee 3

Table 19: The specifications of the Digi Xbee 3 [106], [127]

Specification	Data
Indoor communication range	60 m
Outdoor communication range	1200 m
Transmit power	+8 dBm
Transmit current	40 mA
Receive current	17 mA
Serial data interface	SPI. UART, I^2C
Analog-to-Digital (ADC) inputs	4 x 10-bit ADC
GPIO pins	15
Programming languages	Custom Micropython package
Communication protocol	Zigbee 3.0
Encryption	128/256-bit AES
Price (CAD)	$38.84

Appendix B

Development of the Relay Control Circuit

The Digi Xbee 3 GPIO pins can only deliver 3 mA of current at 3.3 V. The relay control lines must be able to toggle at that rated current. Due to the restrictions on the control line current, initially the MCMX60D10 was chosen. It was discovered during testing that the relay did not fully close, and some current would leak through it. The relay was changed to the G5LE-1A4 DC3, which used

physical contacts instead of electronics. The G5LE requires roughly 133 mA of current, which is way above the capacity of the GPIO pins. Thus, an analog circuit was designed to overcome the limitations. Figure 18 shows the high-level circuit for the relay control.