5-2016

Embedded Graphical User Control Interface for an Advanced Battery Management System

Nicole C. Bugay
Florida International University

Florida International University
Department of Electrical and Computer Engineering

Embedded Graphical User Control Interface for an Advanced Battery Management System

An ARCH Undergraduate Thesis

Submitted in Partial Fulfillment of the
Honors College Curriculum

By
Nicole C. Bugay

Advisor
Osama A. Mohammed, Ph.D.

Miami, May 2016

Table of Contents

Abstract

Energy storage systems have many applications including making the power grid more efficient, reliable, and economical and assisting in efficiently integrating renewable energy. Wind and solar energy are intermittent and therefore provide varying unpredictable power to the grid. Energy storage systems can be used to store energy while it is available for later use, thereby playing a significant role in the increased implementation of renewable energy. A battery management system is necessary to monitor and maintain safe, optimal operation of each battery stack in the energy storage system. The purpose of this research is to create and implement an advanced graphical user interface for a battery management system (BMS). The BMS will allow each battery into the stack to be individually monitored and managed while allowing different charging profiles to be applied. The graphical user interface will be created on the Linux QT platform and displayed on a BeagleBone board. This embedded board will then be implemented into the physical circuitry of the battery management system, a STMicroelectronics microcontroller. The battery management system being designed for is unique in the fact that individual batteries can be isolated. This means that while some batteries are charging, others can be supplying the load. As a result, the system is more reliable which can help it economically and quickly meet peak loads or eliminate short term power outages, for example. Currently peak loads are met by operating power plants at a higher capacity than what is needed to meet unexpected surges. To manage measurements and control commands between the graphical user interface and the microcontroller, a custom communication scheme will be implemented using Matlab Simulink to pass query, command, and configuration messages.

Chapter 1: Introduction

Electricity demands are constantly fluctuating. It can change from year-to-year, day-by-day, and even minute-to-minute. In order to be sure that the demands will always be met and to ensure uninterrupted service to consumers, power plants usually generate about 20% more power than necessary. As a result, only about 55% of installed generation capacity is used over the course of a year. This inefficiency could be reduced if the amount of energy storage in the grid was increased. Energy storage would allow more plants to run at full capacity with unused energy being stored for future use and peak demands being met by this stored energy [1]. Also, compared to conventional generators, energy storage systems have a faster ramp rate. The ramp rate is the rate at which a generator changes its output usually expressed in megawatts per minute. This allows energy storage to quickly meet load fluctuations, thereby increasing the stability and reliability of the power system.

Increasing the number of energy storage systems would also allow for increased implementation of renewable energy. The wind is not always blowing and the sun is not always shining. Since energy must be used immediately after it is generated, if there is an abundance of wind, for example, but low electricity demand, the energy would be wasted. Figure 1 shows that available wind and solar do not necessarily meet the demands of the customers. For the figure, data for wind, solar, and demand was collected over thirty days, superimposed on top of one

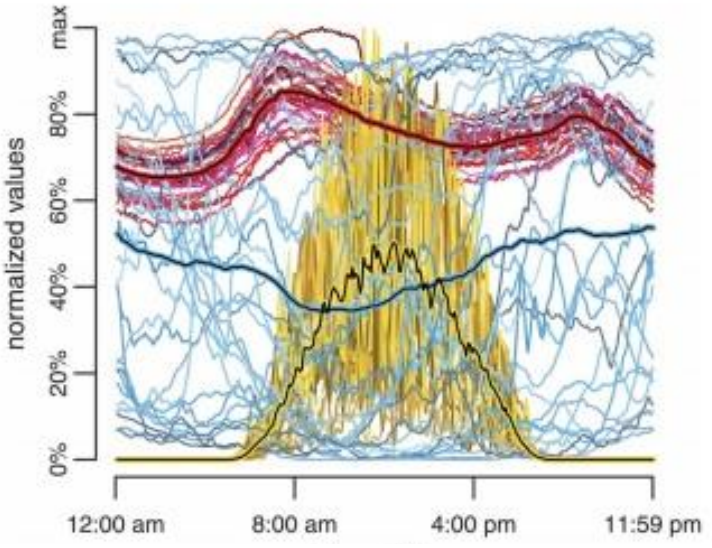

Figure 1: Wind –power generation (blue), insolation (gold), and power demand (red) taken over 30 days in April 2010. Averages in color-highlighted black lines [5].

another, and then normalized to its maximum values. Since supply does not necessarily meet demand, energy storage systems are necessary to store the energy while it is available so that it can be used at a later time when it is needed [2]. Increasing renewable energy will not only benefit the environment, but will also provide energy security by diversifying the sources of available energy [1].

In 2014, there were 1088 energy storage projects worldwide. Of these, operating projects only made up about 1.8% (145 GW) [3] of global power capacity, most of which came from pumped hydro plants. Batteries account for only a small portion (about 0.3%) of existing energy storage (456 MW) [3]. The term "batteries" in this case refers to the following technologies: flow, lithium ion, sodium sulfur, nickel cadmium, lead acid, electrochemical capacitors, and ultra-capacitors [3]. Despite this, batteries are considered the best choice for energy storage due to their versatility, high energy density, and efficiency.

In order to increase the amount of battery storage in the grid, a battery management system (BMS) is needed in conjunction with the battery storage system. This system is necessary to ensure optimal performance of the battery system. The BMS provides detailed monitoring including voltage, current, power, c-rate (Coulombic rate), and energy information. It can also predict the state of charge and state of health of the batteries [1]. This information can then be used to detect faulty batteries. The battery management system can be used to control charging and depth-of-discharge parameters, which are necessary to increase the battery's useful life [4]. A significant amount of energy is required to produce and maintain batteries, from mining for materials to installation. Batteries, therefore, must produce enough energy in their lifetime to make them worthwhile. Increasing the lifespan and efficiency of batteries can help offset the energy required to produce them [5].

In this work, a graphical user interface (GUI) has been created to allow the user to monitor and manage each battery or battery stack in the system from a supervisory controller. Qt Software under Linux is being used to create the GUI which is displayed on a BeagleBone with an attached touchscreen. The GUI was designed to have a user-friendly, expandable interface consisting of a home screen, a setup window, an informative window for each battery, and an informative window for the load. Chapter 2 provides a literature review. Chapter 3 describes the methods used to create this project. The results of this project will be given in Chapter 4 and the results are discussed in Chapter 5. The conclusion of this thesis can be found in Chapter 6.

Chapter 2: Literature Review

Microgrids play an important role in the evolution of our current power grid into a Smart grid, one that is efficient and economically friendly. Microgrids are isolated grids that can connect to the main grid for the purpose of exchanging energy and information. They consist of three parts: distributed generators (such as PV cells and wind turbines) and energy storage systems, control and communication systems, and the electrical load which consumes the energy [6]. Energy storage consists of any device that stores energy and there are many different types. Examples include lead acid batteries, lithium ion batteries, and ultra-capacitors.

First, consider the current global situation in which energy storage systems are being used. Huff (2014) states that the top services of energy storage systems is to provide an electric energy time shift, supply electric capacity, and firm renewable capacity, respectively [3]. Electric energy time shift refers to storing energy when the price is low, and discharging when the price is high. Supplying electric capacity means using energy storage instead of or to reduce the need for buying a new generator. The third service, firming renewable capacity, refers to using energy storage to mitigate the rapid changes in output from renewable energy sources. This includes accounting for sudden gusts of wind or the sun going behind clouds, for example [7].

As of 2014, Huff noted that energy storage only made up about 1.8% (145 GW) of the global power capacity, most of which came from pumped hydro plants. Batteries only account for about 0.3% (456 MW) of existing energy storage. Huff's information comes from the Department of Energy Global Energy Storage Database which was made in conjunction with Sandia National Laboratories [3].

In March 2016, Green Tech Media (GTM) Research and the Energy Storage Association (ESA) released U.S. Energy Storage Monitor: 2015 Year in Review. According to their research, the United States had a record year for energy storage deployments, deploying a total of 221 MW. This number is up 243% from 2014 which had 65 MW of energy storage deployed. In MWh, this is about 161 MWh (an 88% increase from 2014). The total U.S energy storage deployment from 2012-2015 can be seen in Figure 2. The energy storage utility-scale system price in 2015 was around $700-$1,200 per kWh, an 8%-13% decrease from the previous year. Green Tech Media (GTM) Research also expects significant growth in the U.S market over the next five year, resulting in a 1,662 MW annual market by 2020. This is about 8 times the size of the 2015 market and 26 times that of the 2014 market. As a result, the U.S. energy storage market is expected to be worth $2.5 billion by 2020. This is about 6 times the 2015 market of $432 million [8].

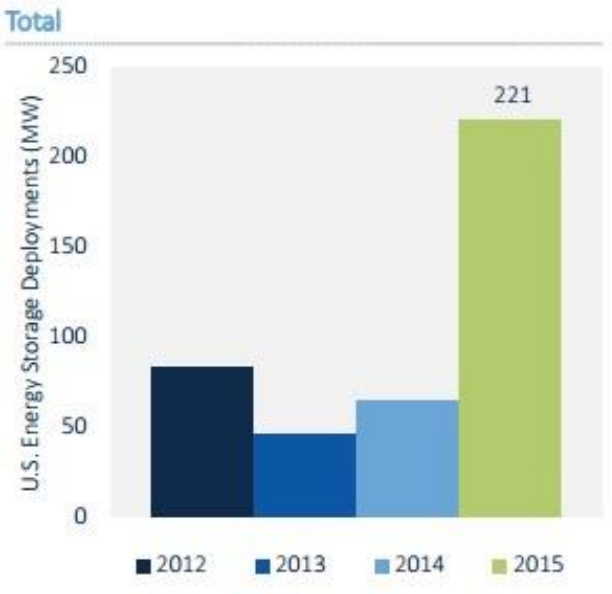

Figure 2: Total U.S Energy Storage Deployments (in MW) per year [8]

2.1 Benefits of Energy Storage Systems

Energy storage can be used to help improve the power grid. Yeleti and Fu (2010) examined the benefits of energy storage and how they can be used to make the current power system more efficient, reliable and economical. They found that electrical energy storage systems can be used to store inexpensive energy during off-peak times. This stored energy can then be used to meet peak loads quickly when the energy is expensive, thereby improving the stability and reliability of the power system. Energy storage systems will also play a significant role in the increased

implementation of renewable energy. Since wind and solar are intermittent, energy storage systems are necessary to store energy while it is available so that it can be used at a later time. Fu et al. also believe that energy storage systems can be used to decrease congestion on transmission lines. Fu is an assistant professor at Mississippi State University who got his Ph.D. in electrical engineering from Illinois Institute of Technology. Yeleti is working on his Master's in electrical engineering at Mississippi State University. Yeleti and Fu support the assertion that it is necessary and beneficial to have an efficient energy storage and management system [2].

Lawder, Suthar, Northrop, De, Hoff, Leitermann, Crow, Santhanagopalan, and Subramanian (2014) also believe that the current power grid is inefficient. Due to the fact that electricity demands are constantly fluctuating, power plants usually generate about 20% more power than necessary to ensure adequate supply so that customers' service is not interrupted. Lawder et al. concluded that by using energy storage systems, this inefficiency can be removed by allowing more plants to run at full capacity with peak demands being met by stored energy that can be immediately used by the consumer when it is needed. Lawder et al. agree with Yeleti and Fu [2] that increasing energy storage will increase the implementation of renewable energy. This is beneficial not only environmentally, but it also provides energy security by diversifying the energy market [1].

Barnhart, Dale, Brandt, and Benson (2013) agree with Lawder et al. [1] and Yeleti et al. [2] that the world needs increased implementation of affordable, and sustainable energy resources. However, Barnhart et al. disagree with them on the fact that energy storage systems are beneficial for all types of renewable energy. In their research, they calculated the Energy Return on Investment (EROI) for both solar and wind farms using energy storage and curtailing. EROI is the amount of energy produced by a technology, divided by the amount of energy it takes to build and

maintain that technology. In the case of batteries, this takes into account how much energy is used throughout its lifetime, from mining for materials to its installation. Barnhart et al. concluded that for solar, grid-scale batteries were beneficial when storing surplus solar energy after the energetic costs were factored in. This is because the energy required to build a solar farm is comparable to the energy required to build the batteries. However, they found that this was not the case for wind energy and that it would actually be more energetically efficient to shut down a wind turbine (curtail it) than it is to store the surplus electricity it generates. This is because wind is energetically cheaper than the energetically expensive batteries required to store the energy. Barnhart et al. concluded that the best ways to improve the battery energy return on investment is to increase their life cycle, efficiency and depth-of-discharge.

This study was done only to compare the energy efficiency of storing electricity versus curtailing it and the authors note that energy storage has other benefits not quantified or analyzed in this study, such as ensuring reliable power supplies, benefiting power grid operations and electricity market economics, and applications in disaster relief and war zone scenarios. Charles Barnhart and Michael Dale are postdoctoral scholars at Stanford's Global Climate and Energy Project (GCEP). Sally Benson is the director of GCEP and a professor of energy resources engineering, while Adam Brandt is an assistant professor of energy resources engineering in Stanford's School of Earth Sciences [5].

2.2 Batteries and the Importance of Battery Management Systems

Lawder et al. believe that batteries are the best choice for energy storage due to their versatility, high energy density, and efficiency, even though battery energy storage systems currently account for only a small portion of the existing energy storage in the grid. To encourage

their integration, they believe a battery management system is necessary to monitor and maintain safe, optimal operation of each battery stack.

The battery packs of the system degrade during cycling. This can be worsened by extreme charging patterns such as undercharging and overcharging. A smart battery management system can be used to reduce these causes of degradation by providing optimal charging patterns. Detailed monitoring is also useful to predict the battery state of charge (SOC) and state of health (SOH). Leitermann and Crow have Ph.D.'s in electrical engineering, while Santhanagopalan and Subramanian have Ph.D.'s in chemical engineering. Lawder, Suthar, Northrop and De are all working on their Ph.D.'s in energy, environment, and chemical engineering at Washington University. Hoff, on the other hand, works in industry and is Vice President of Research and Technology at A123 Energy Solutions. Lawder et al. support my research by providing evidence of the importance of a battery management system [1].

Jurgen Garche and Andreas Jossen (2000) agree with Lawder et al. that a battery management system is necessary to monitor the SOC and SOH of the batteries which can then be used to detect battery failures. Garche and Jossen also looked at the parameters that influence a battery lifecycle. They conclude that operating temperature, the method used to charge the battery, maintenance, and depth-of-discharge affect how fast the battery ages and that some of these parameters, such as charging mode and depth-of-discharge, can be controlled by a battery management system to increase the battery's useful life. As previously mentioned, Barnhart et al. have said that increasing the battery's useful life is necessary in order to increase the energetic efficiency of batteries in wind farms [5]. Garche and Jossen both work for the Center of Solar Energy and Hydrogen Resources in Ulm, Germany [4].

Battery management systems can also be used to improve microgrids. Microgrids can either stand alone (off-grid) or be connected to the main grid and play an important role in integrating distributed energy resources. Distributed energy resources consist of both generators (for example PV cells and wind turbines) and energy storage systems. Research has been done to develop control strategies for batteries in microgrids. However, in this research, the batteries are usually treated as an ideal power source. Xu, Miao and Fan (2012) point out that in reality, a battery has operational limits, such as the fact that the state of charge cannot be lower than a certain threshold. The microgrid and battery management system were then simulated. Xu et al. concluded that it is necessary to develop adequate control strategies using battery status information that was collected by the battery management system [9].

2.3 Improving Battery Management Systems

Since a battery management system requires different controls and modeling in order to produce the optimal results previously described, Miao, Xu, Disfani, and Fan (2014) investigated what these models should be. Miao et al. agree with Lawder et al. that the battery management system must take into consideration the battery state of charge in order prevent damage due to extreme charging profiles. Since state of charge cannot be directly measured, Miao et al. derived a way to estimate it based on coulomb counting. This works because the charge accumulated by the battery can be represented as an integration of the current going into the battery. Miao et al.'s battery management system also has a way of identifying which mode the battery should be in based on the state of charge and whether or not the microgrid is isolated from the main grid. Miao and Fan have Ph.D.'s in electrical engineering from the University of West Virginia and both currently work at the University of South Florida. Xu and Disfani are pursuing their Ph.Ds. at the University of South Florida [10].

The battery management system (BMS) that I worked on creating a GUI for is described in a paper by Elsayed, Lashway and Mohammed (2015). The BMS was designed to allow for each battery in an energy storage system to be individually monitored and managed. The system is able to monitor the voltage, current, power and energy of each battery along with its C-rate [11], the rate at which a battery is charged or discharged relative to its maximum capacity [12]. This normalized current allows for the behavior of the voltage to be anticipated no matter the size of the battery. For this reason, Elsayed et al. believe that C-rate is crucial for designing a predictive and advanced management system, and also because it can be used to analyze each battery's performance. For example, as the C-rate increases, the amount of usable capacity available will decrease.

There are three modes of operation in the battery management system which can be specified for each individual battery. These include: load connected mode in which the battery is connected to the stack and is discharging as it supplies the load, charging mode in which the battery is disconnected from the load and is charging, and maintenance mode in which the battery is not connected to the stack nor is it charging. Different charging profiles can also be applied to individual batteries in the charging mode, in either constant or pulsed mode. Pulsed charging is a relatively new tactic which can be used to repair faulty batteries, particularly lead acid batteries. As lead acid batteries discharge, sulfate crystals form. This is a normal phenomenon, but if the battery remains uncharged, the crystals become stable crystalline deposits. These deposits decrease the battery's capacity. Pulsed charging can be used to crack sulfation and break down deposits, so that the battery can once again charge to full capacity. Elsayed and Lashway are Ph.D. candidates and Mohammed is a professor and director of the Energy System Research Laboratory at Florida International University [11].

Batteries in an energy storage system can go through hundreds to thousands of cycles in their lifetime. For this reason, Christensen et Adebusuyi (2013) believe that is important that the state of health (SoH) of a battery be known. SoH is a generic term used to describe the performance of a battery compared to its performance when it was new. Though there is currently no set formula for finding state of health, there is agreement on what parameters affect it. The main factor involved is the internal resistance of the battery. As a lithium ion battery is charged and discharged, the electrolytes in the battery react with both the anode and the cathode to form a film, the solid electrolyte interface (SEI). As it continues cycling, this film thickens, increasing the internal resistance of the battery. This reaction occurs at any temperature but is worse at lower temperatures [13]. For a lead acid battery, the internal resistance is a result of the sulfation previously discussed. When a lead acid battery remains uncharged, the naturally occurring sulfate crystals become stable crystalline deposits and increase the internal resistance of the battery [11]. As the battery resistance increases, the battery's ability to deliver power decreases. Christensen is working on an industrial PhD for Lithium Balance and the Technical University of Denmark. Adebusuyi is the Products and Applications Manager at Lithium Balance [13].

2.4 Implementation of a Graphical User Interface for a Battery Management System

To design the graphical user interface for the battery management system, the Qt Linux platform was used. Molkentin (2007) provides a good foundation for building Qt applications which will be necessary in the GUI creation. He explains the contexts and basic techniques needed to use Qt. There is also a chapter about input/output interfaces which describes Qt's base class, QIODevice. This class provides a platform for sending data and accessing external processes no matter what type of Input/Output. Molkentin worked on the KDE project for Qt Company, formally known as Trolltech. The KDE project was to develop an entire desktop based on Qt.

Consequently, he has a good understanding of how Qt works, however Molkentin does not discuss Qt's application in embedded systems. The GUI for the battery management system will eventually be uploaded onto a BeagleBoard, an embedded board. Therefore, it is necessary that whatever program is used in the creation of the GUI be executable on an embedded system [14].

Rischpater (2013) believes that Qt is a very suitable program for creating graphical user interfaces for embedded environments. In fact, Rischpater believes that for embedded platforms it is actually better to use Qt's classes over straight C++ programming. This is because Qt classes take less memory and are more readable and portable on all platforms. The reason Qt classes save memory is because, unlike STL collections, Qt's are reference based and use copy-on-write. Unfortunately, the tradeoff for this is typically slower performance speed. However, Rischpater points out that in practice, there is rarely enough to cause a problem. Rischpater is a senior engineer at Microsoft and has over 20 years of experience writing about and developing for computing platforms. His book teaches the reader how to build applications using Qt Designer and how to design GUIs using Qt Creator with methods that are clear and effective. Rischpater's book supported research efforts by helping in the creation and optimization of the GUI for the battery management system [15].

Thelin (2007) agrees with Rischpater that Qt is acceptable for use in embedded systems. Thelin is currently a consultant focusing on embedded systems with 20 years of experience in software development. The main differentiating factor between his book and other Qt books is that it includes a chapter on databases. Understanding databases helped in learning how to store data managed within the Qt application, such as measurements from the physical battery system, and will also play a major role in future work on this project. Thelin only provides basic information on the subject and notes that separate sources must be used to gather more information on the

specific database engine one chooses. The author believes that no matter what database engine is chosen, Qt's module makes it easy to integrate [16].

One such option is to use SQLite to incorporate a database into Qt. SQLite is a software library that implements a server-less Structured Query Language (SQL) database engine. Junyan, Shiguo, and Li (2009) feel that traditional databases do not meet the needs of embedded systems. Databases for embedded systems need to be small, fast, reliable and easy-to-port. Out of all the embedded databases Junyan et al. compared, they found SQLite to be the best because it was powerful yet small and simple to use. Because it is server-less, it does not rely on an external system, making it easy to utilize for embedded systems. As a result, the complete database can be stored in a single cross-platform disk file. Junyan et al. implemented SQLite with a graphical user interface design based on embedded Qt and therefore may be a possibility in the GUI for the battery management system. Junyan et al. were published as part of the IEEE International Forum on Information Technology and Applications [17].

Qt is a popular choice in embedded systems and has been used to create the GUI used to manage a variety of systems. For example, Shengwen, Chen, and Wen (2011) designed a wind power supervisory control system. It utilized an embedded development platform based on Linux and an ARM microprocessor along with Qt embedded software. The microcontroller used to perform the tests of the GUI in this project also used an ARM microprocessor. The authors felt that embedded Qt was the best to use due to its high performance, small footprint and to the fact that it can be deployed across different systems without changing the source code. For communication, the authors used qextserialport which is a third party serial port control class [18].

At the time of Shengwen et al.'s paper, Qt did not provide a serial port control class. In order for the GUI to communicate with the physical battery management system, a serial port must

be used. According to the Qt Documentation, for Qt to be able to read from and write to a serial port, QSerialPort class must be used. QSerialPort provides functions to access serial ports. QSerialPortInfo can be used in conjunction with QSerialPort to provide information about the available serial ports. After setting the port, QSerialPort can be used to open the port in read-only, write-only, or read-write mode using the open() method. Once the port is open, QSerialPort tries to determine the current configuration of the port and initialize itself. However, this can be overwritten and reconfigured using setBaudRate(), setDataBits(), setParity(), setStopBits(), and setFlowControl() methods. Once the port is open and ready to read or write, the read() and write() methods can be used respectively [19].

Qt, however, does not have a unique pre-made widget to create graphs of data. To make up for this, Emanuel Eichhammer created QCustomPlot for others to use under the GNU General Public License. QCustomPlot is a Qt C++ widget for plotting and data visualization. The focus of this plotting library is to make quality 2D plots, charts, and graphs, as well as offer high performance for real time data visualization [20].

Energy storage systems play a major role in transforming the current power system into a smart grid, a power grid that is more efficient, reliable and economical, especially by facilitating the creation of microgrids. Microgrids can connect to the main grid or operate on their own. Energy storage allows them to separate from the main grid and they also make it possible to more efficiently integrate renewable energy. Though there are many types of energy storage, batteries were determined to be the best choice for energy storage due to their versatility, high energy density, and efficiency. However, despite all this, battery energy storage systems currently account for only a small portion of the already limited existing energy storage in the grid. To encourage their integration, a battery management system is necessary to monitor and maintain safe, optimal

operation of each battery stack. Accurate measurements and monitoring prevent from damaging the batteries and the overall system along with making sure that the system is operating efficiently.

The graphical user interface for the battery management system designed is able to effectively manage and monitor the batteries in the energy system. It must be reliable, accurate and user-friendly in order for the system to run optimally. The graphical user interface was uploaded onto a BeagleBoard, an embedded board. Qt was chosen to design the graphical user interface for the embedded system. It was selected because of its small footprint, portability, and ease-of-use due to its specially designed Qt classes, such as QSerialPort and QSerialPortInfo.

Chapter 3: Methods

The battery management system for which this GUI is being created will consist of the battery module and a microprocessor. The microprocessor will be used to collect measurements from the battery module and control switches (for switching the battery's mode of operation). The battery management system will then be connected to a BeagleBone with an attached touchscreen where the GUI will be displayed. Figure 3 shows the hierarchy of the entire battery management system.

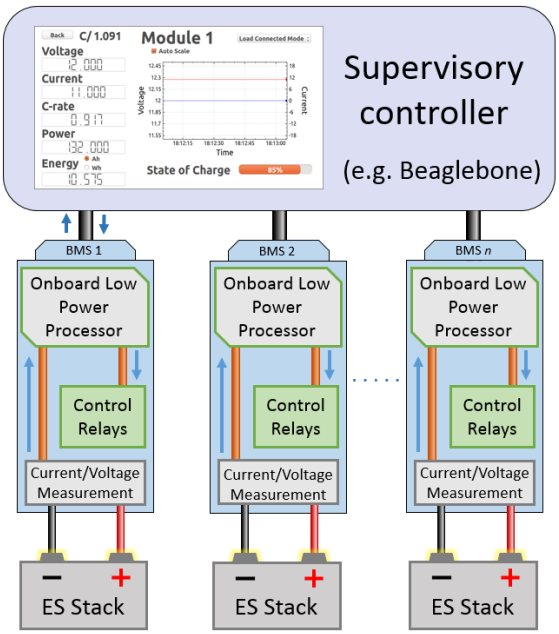

Figure 3: Hierarchy of the Battery Management System

The components will be connected by a serial port using a custom communication scheme. This communication scheme is able to pass query, control, and configuration messages. A functional block diagram of the communication in the battery management system can be seen in Figure 4. Control commands will take priority over the other commands, because they will be used to control the switches that determine what mode that battery is in. In order to represent these

switches on the microcontroller, LEDs will be used. Both LEDs and switches operated by either being high or low (on or off). Since LEDs and switches take similar commands, each LED will represent one of the switches in the battery system.

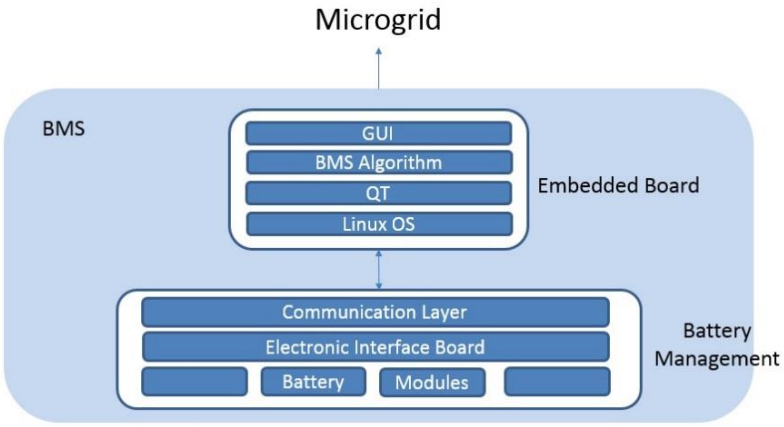

Figure 4: Functional Block Diagram of BMS Communication

In order to create the graphical user interface (GUI) for the battery management system, Qt software under Linux was used. The community version of Qt Creator version 5.5 was used to design the individual screens and link them to the program code. The GUI was designed to be an expandable interface consisting of a home screen, setup, and information windows for each individual battery module as well as an information window for the load.

To design the screens, the Qt creator has pre-made basic widgets for direct and easy use. To add widgets to the dialog boxes, users can drag and drop the appropriate widget from the list into the display box. These widgets can then be programmed in the source code. In the GUI designed, some commonly used widgets for displaying information were, QLabel to display text, QLCDNumber to displays numerical information, and QProgressBar to display the state of charge in an easy to read manner. In order for the user to make a selection, such as going to a new window

or going back, the QPushButton and QComboBox widgets were used. In order for the user to input numerical information, QDoubleSpinBox widget and QDial were used.

The Qt Creator software does not include any pre-made widgets for making graphs to plot data. As a result, a third party open source Qt C++ widget called QCustomPlot was used. These files (qcustomplot.h and qcustomplot.cpp) were downloaded and then added into the project. Since the version is higher than 5.0, printsupport command had to also be added to the QT variable in the .pro file. Creating a QCustomPlot widget in the GUI, was done by promoting the Qt specific widget QWidget [20].

To connect these widgets to user input and create some sort of reaction, a Qt method called signals and slots was used. Signals and slots allows for communication between objects and allows for functions to be called as a result of a certain user input [21]. For example, the signal could be a one second timer ending and the slot could be the energy function that needs to be called every second. This will be discussed more later on. This method was also used to update the graph, perform operations in real time, and program the function of buttons.

The communication between the physical circuitry of the battery management system, specifically the microprocessor used to collect measurements and control switches, and the graphical user interface was done using a custom serial port scheme. In order for Qt to write and read data from a serial port, the Qt classes QSerialPort was used. The port used was ttyS0 and it was opened in read-write mode. In Qt, using QSerialPort's public functions, the baud rate was set to 9600, the data bit size was set to 8 bits, parity was turned off, the stop bits were set to one stop, and the flow control was turned off [19].

The measurements that are collected by the microprocessor are voltage and current. The GUI will then use these values to calculate other information about the battery management system. The power of the battery module was calculated using the following formula:

$$Power = Voltage \times Current \ \rightarrow P = VI$$

The capacity of the battery is set by the user in the setup window upon initiating the GUI. This value is then used to calculate the C-rate of the battery using the following formula:

$$CRate = \frac{Current\ (A)}{Capacity\ (Ah)}$$

It is important to note that since the current and voltage are updating in real time, so are all the calculations using these parameters. The state of charge of the battery will be calculated from the initial voltage of the battery. In the practical system, the initial voltage will be taken as an average while the user is in the setup window. The GUI will request 100 samples of the initial voltage and then average it in order to get the most accurate reading. This value will then be used in the state of charge calculation, which depends on the battery type. If the battery is lead acid (denoted by subscript Pb), the following formula is used:

$$SoC_{Pb} = 1128V^3 - 5512V^2 + 8933V - 4797$$

If the battery is lithium ion (denoted by subscript Li), the following formula is used [22]:

$$SoC_{Li} = (4.162\ x\ 10^{16})e^{-\frac{V-17.85}{2.343}} + 61.36e^{-\frac{V-3.985}{0.2229}}$$

The energy management calculations were also integrated into the GUI. Using the capacity set by the user and the initial state of charge from the microprocessor, the initial energy of the battery is then calculated using the following formula:

$$Energy_{Initial} = \frac{Capacity \times State\ of\ Charge}{100}$$

This value is then stored in persistent memory as Energy. A function is then created that will be called every one second. This function is called every one second, because an Ampere is one Coulomb times 1 second. In order to convert this into the same units as energy (Ah), the current must be divided by 3600. In this function the global Energy is assigned to the variable energyInitial. energyInitial is then used to calculate energyNow using the following formula:

$$Energy_{Now} = Energy_{Initial} + \frac{Current}{3600}$$

energyNow is then assigned to the permanent memory space Energy, overwriting the original initial energy that was there. The calculation is then repeated every one second, continuously adding the energy calculations. A flowchart of the procedure for energy management calculations can be seen in Figure 5.

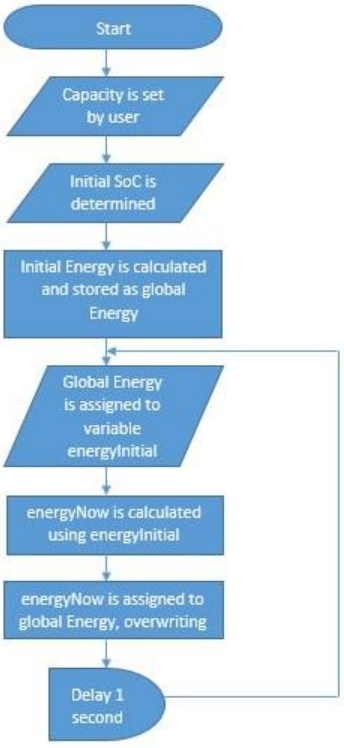

Figure 5: Energy Management System Flowchart

Once the GUI was designed and created, it was downloaded and displayed on a BeagleBone with an attached touchscreen. A serial port was then be used to connect the BeagleBone to the microprocessor of the battery management system. Matlab Simulink was used to set up the communication scheme on the microcontroller. A Simulink Blockset called Waijung was used to generate C code from the Simulink simulation mode for the microcontroller. Waijung was used because it was designed specifically to support STM32F4 family of microcontrollers from STMicroelectronics [23].

In order to verify the accuracy of the voltage and current measurements being transmitted over the serial port and being displayed on the GUI, an oscilloscope was connected directly to the battery module. Voltage and current measurements were requested by the GUI every second by exchanging query commands to the microcontroller via serial port. The measurements to be displayed on the GUI were taken by a STMicroelectronics Discovery microcontroller. The STMicroelectronics Discovery microcontroller was programmed to behave the same as the chip on the final battery management system, which is still in development. In the first accuracy test, the microcontroller collected measurements at a rate of 2 samples per second (a period of 0.5 seconds). In the second accuracy test, the measurements were collected at a rate of 100 samples per second (a period of 0.01 seconds). The measurements are then passed from the microcontroller over a serial port back to the BeagleBone using the same syntax that will be implemented into the final product. A 12V lead acid battery was used for the source and a 10Ω resistor was used as the load. The analog to digital converter on the STMicroelectronics Discovery microcontroller can only accept voltage values between 0-3.3V. As a result, the battery voltage was first passed through a voltage divider consisting of a 1kΩ and 4.7kΩ resistor scaling the voltage by a factor of approximately 0.175 to adhere to the range of the microcontroller. The current was measured using a shunt resistance of 1.2Ω. A shunt resistor is a high precision resistor. Since the microcontroller measures voltage, knowing the exact resistance along with the voltage allows for the current to be calculated using Ohm's Law as follows:

$$Current = \frac{Voltage}{Resistance}$$

Compensations for the voltage divider and the shunt resistance were made on the microcontroller before sending the information to the GUI. To scale the voltage back to its appropriate form, it

was divided by 0.175 and the current was calculated using the above formula. These calculations were programed in Matlab Simulink before sending the information over the serial port.

Chapter 4: Results

This project has three outcomes. The first outcome is the graphical user interface that was designed for the battery management system. The second is the design and implementation of a communication scheme between the BMS board and the microcontroller. The third outcome is the results of the verification test to determine the accuracy of the measurements the GUI is receiving. These results will be provided in the following sections.

4.1 The Graphical User Interface

The GUI was designed to be an expandable interface consisting of a home screen, setup, and information windows for each individual battery module as well as an information window for the load. When the GUI is initiated, the user will be taken to the main window. At first, the only button available to the user here will be the setup button. This button will take you to the setup screen which allows the user to input information about the battery before the system can start running. The user can specify the maximum and minimum voltage limit, the maximum current limit, the type of batteries, the number of cells in the batteries, and the capacity of the batteries. The selections made about the type of battery here will affect which formula is used to calculate the state of charge, as previously discussed. After the batteries are set up, the home screen will allow the user to choose if he/she wants to look at the individual batteries or the load. Figure 6 shows the main window after the user has setup the batteries and Figure 7 shows the setup window.

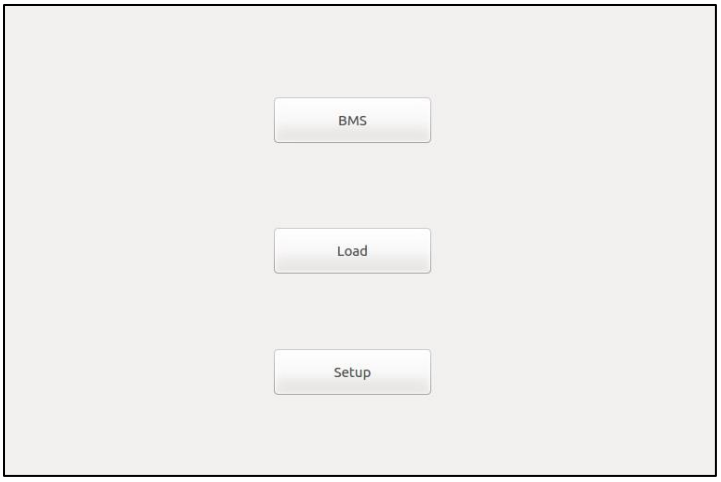

Figure 6: Main Window after Batteries have been Setup

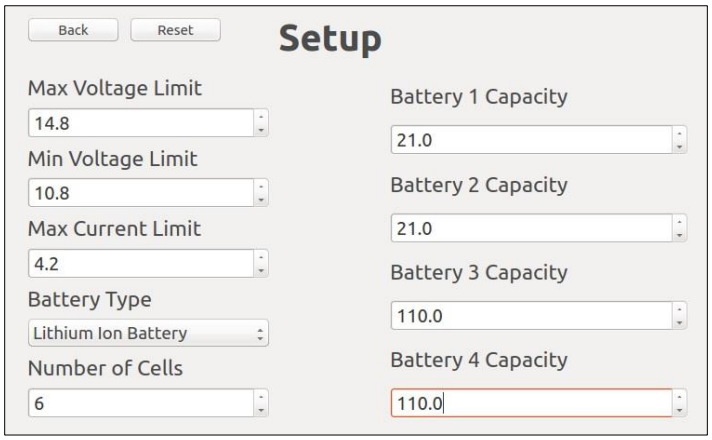

Figure 7: Setup Window

If the user chooses the load button, they will be taken to the load information window which can be seen in Figure 8. This screen digitally displays the battery voltage, DC bus voltage, DC bus current, current stored energy, and state of charge. All the displays are accurate up to three decimal places and all the graphs in the project can be auto-scaled to best match the data if desired. The state of charge at the load is the lowest state of charge of the array of batteries connected to it.

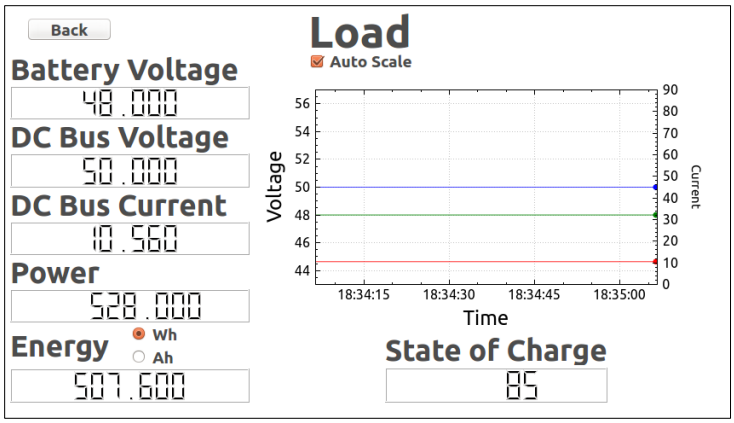

Figure 8: Load Information Window

The user can also view information about the individual batteries in the system by selecting the BMS button on the main window and then selecting which battery they would like to look at on the BMS home window as shown in Figure 9. Although this version has only four battery modules, the program could easily be extended to accommodate as many batteries as desired.

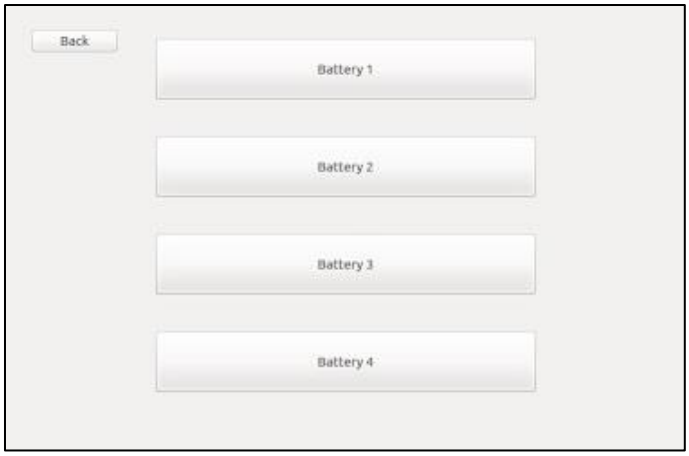

Figure 9: Battery Management System (BMS) Home Window

Once a battery is selected, the information for that battery will be displayed as shown in Figure 10. This information will be displayed digitally and includes voltage, current, state of charge, C-rate, power, and energy. The voltage and current of the battery will also be displayed in

a graph to show changes over time. As previously mentioned, the graph can be auto-scaled if desired.

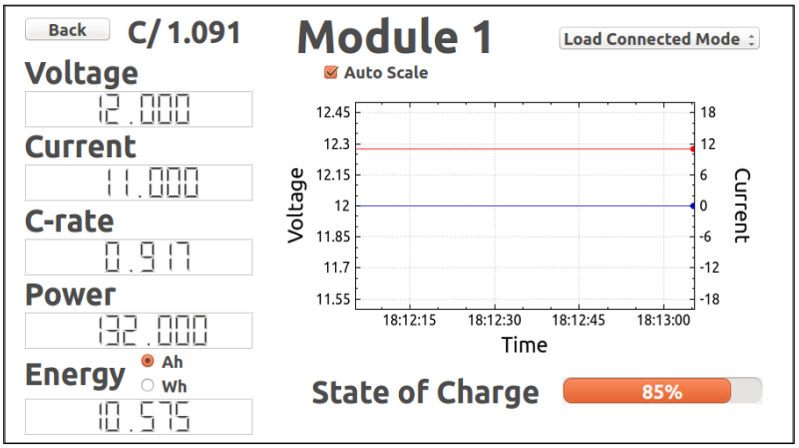

Figure 10: Battery Module 1 Information Window

The user is also able to apply the different charging profiles to the battery. As previously mentioned in Chapter 2 and Chapter 3, there are three modes of operation: load connected mode, maintenance mode, and charging mode. In the battery module information window, the user can specify which mode he/she would like by selecting it from the drop down box. On the microcontroller, three LEDs, orange, red, and green, were used to represent the three switches that control which mode of operation of battery is in. When the LED is on it represents the switch being closed. Alternatively, when the LED is off it represents a switch on the battery management board being opened. Figure 11 shows the three combinations of LEDs on the microcontroller. The LEDs were controlled by user selection of the mode in the drop down box.

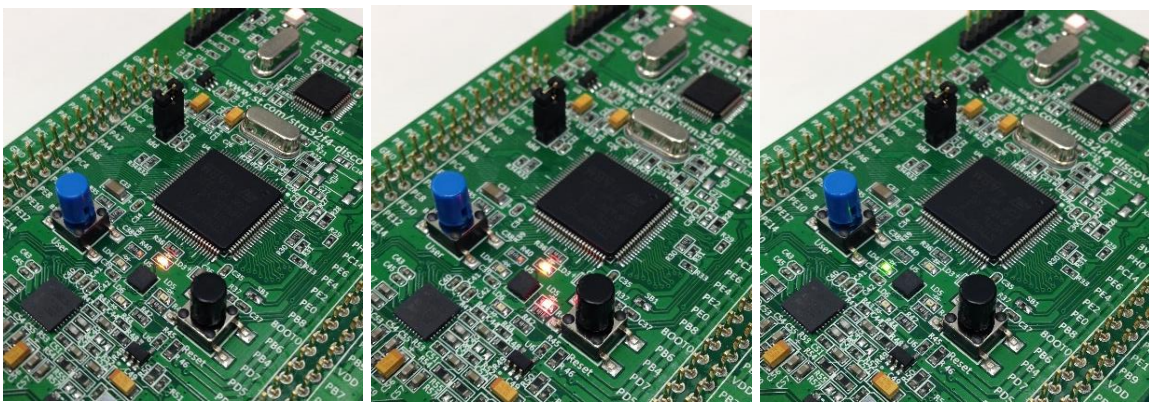

Figure 11: LEDs to represent the switching configurations of load connected, charging, and ideal mode (from left to right)

If charging mode is selected, a new window will appear to allow the user to specify the type of charging they want done, either constant or pulsed charging. This charge controller window can be seen in Figure 12.

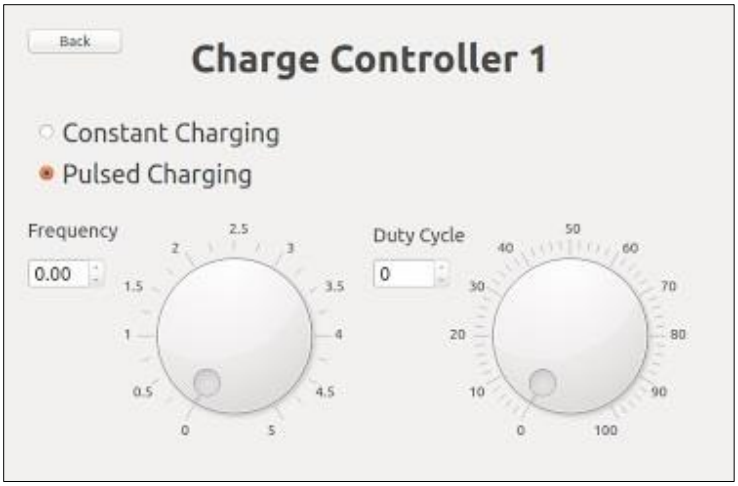

Figure 12: Charge Controller Window for Battery Module 1

4.2 The Communication Scheme

A custom communication scheme, referred to as the Energy Storage Modular Controller (ESMC) communication structure, was designed and implemented in order to pass commands

between the BMS board and the STMicroelectronics microcontroller. The STMicroelectronics microcontroller was programmed to interpret the strings of commands using Matlab Simulink.

Queries are sent from the GUI by the BMS board over the serial port to the microcontroller. The purpose of these queries is to request information back from the microcontroller about the battery system. A summary of queries used in this project can be seen in Table 1. The ESMC communication scheme assigns an address to each of the batteries in the system, which is denoted by x.

Function	**Query**	**Unit**
Voltage	ESMCVx	V
Current	ESMCIx	A

Table 1: Summary of Queries

Figure 13 shows how Matlab was programmed to allow the microcontroller to interpret and respond to the queries it receives.

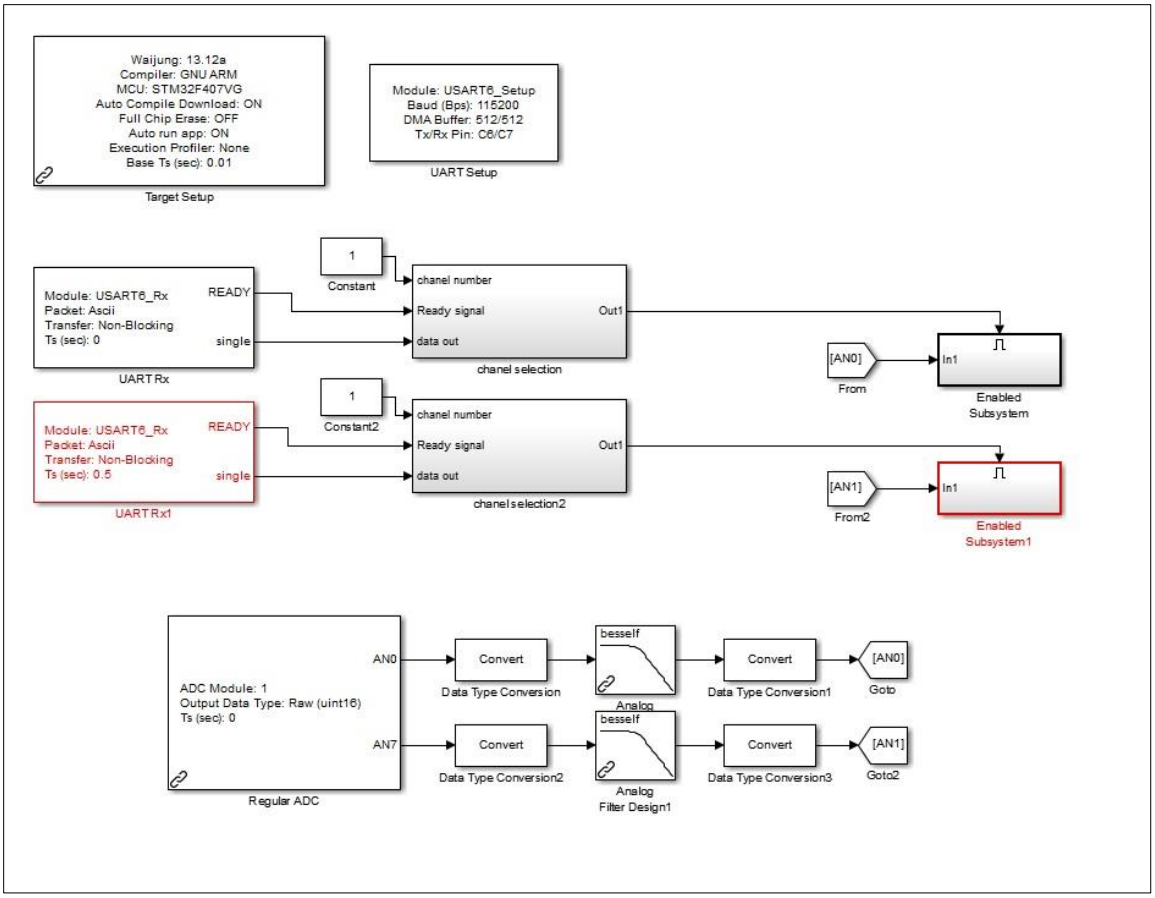

Figure 13: Target and Query Setup in Matlab

Commands are also sent from the GUI by the BMS board over the serial port to the microcontroller. The purpose of commands is to elicit an action from the microcontroller, specifically to set the mode of the battery by controlling the three switches previously discussed. A summary of the commands used can be seen in Table 2. As previously mentioned, an x is used to denote the number of the battery in the system for which the command is intended.

Function	**Command**
Load Connected Mode	ESMCmodeLOADx
Ideal Mode	ESMCmodeIDEALx
Charging Mode	ESMCmodeCHARGEx
Pulsed Charging Mode	ESMCmodePCHARGEx

Table 2: Summary of Commands

Similarly, configuration commands are sent from the GUI by the BMS board over the serial port to the microcontroller. The purpose of configuration commands is to set the parameters of a command and as a result contain a float integer. A summary of the configuration commands used can be seen in Table 3. As previously mentioned, an x is used to denote the number of the battery in the system for which the configuration is meant to affect.

Function	Command	Unit
Set Duty Cycle of Pulsed Charging Mode	ESMCmodeDUTYx=%f	%
Set Frequency of Pulsed Charging Mode	ESMCmodeFREQx=%f	Hz

Table 3: Summary of Configuration Commands

Figure 14 shows how the commands and the configuration commands were designed in Matlab Simulink to create an action out of the microcontroller.

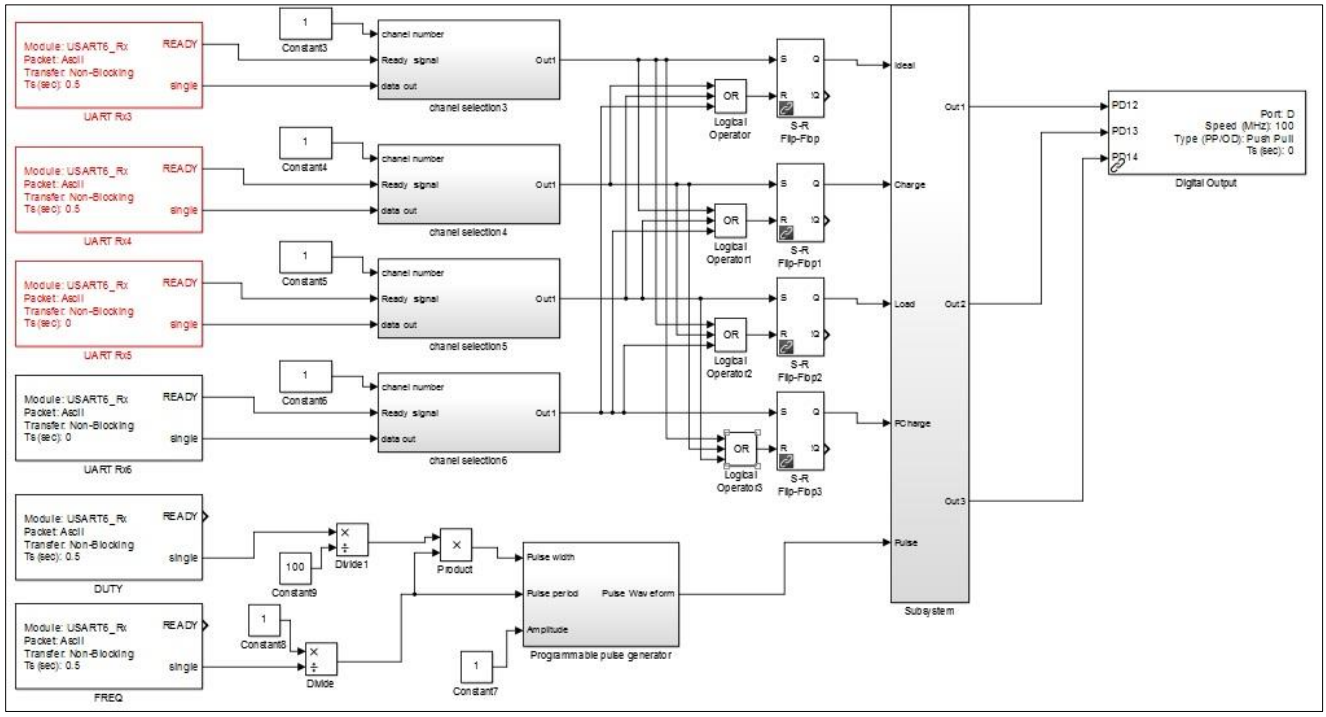

Figure 14: Command and Configuration Setup in Matlab Simulink

The inside of the subsystem shown in Figure 14 can be seen in Figure 15.

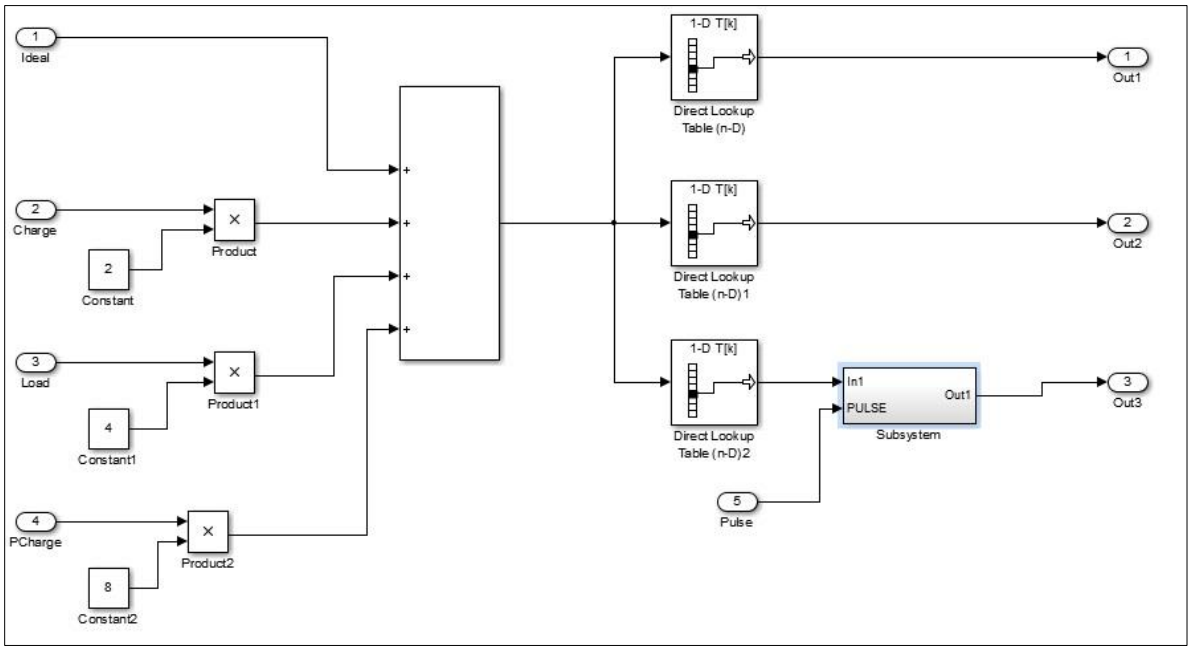

Figure 15: Command Subsystem in Matlab Simulink

The inside of the subsystem shown in Figure 15 can be seen in Figure 16.

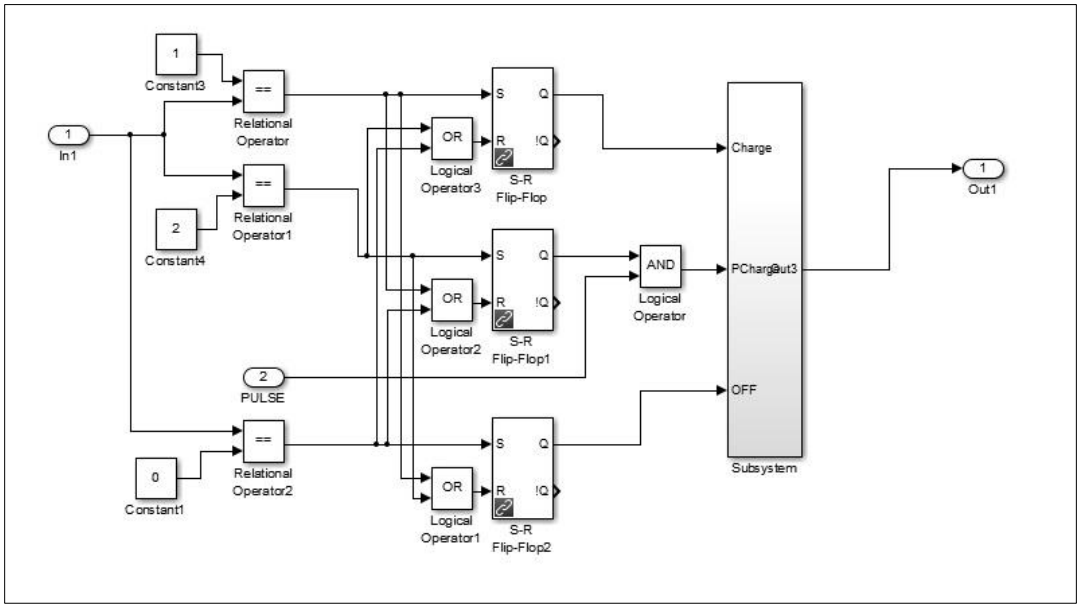

Figure 16: Pulse Subsystem in Matlab Simulink

4.3 Verification Test Results

Verification test results were obtained by comparing the voltage and current measurements displayed on the GUI with those obtained directly by the oscilloscope. The microcontroller samples measurements at a rate of 2 samples per second in verification test 1 and at a rate of 100 samples per second in verification test 2.

4.3.1 Results of Verification Test 1

A screenshot of the oscilloscopes measurements can be seen in Figure 17.

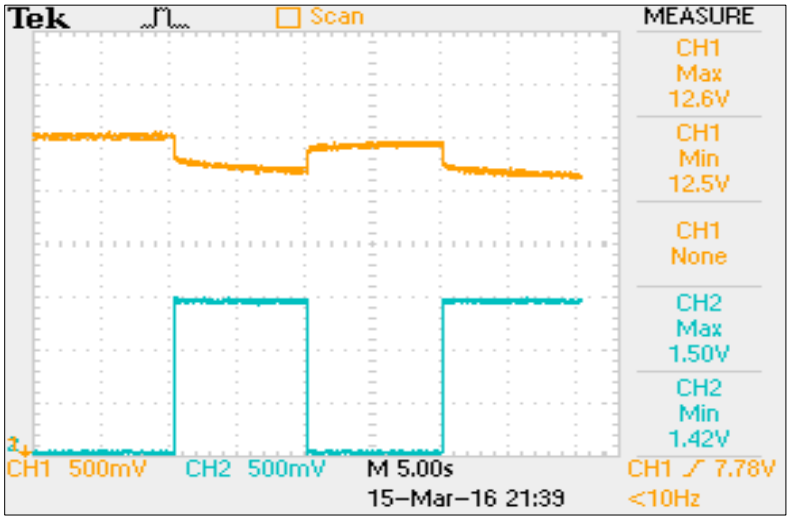

Figure 17: Screenshot of Oscilloscope Voltage and Current Measurements

The graph in Figure 17 was then compared with a screenshot of the GUI, as shown in Figure 18, which also includes a graph of the voltage and current.

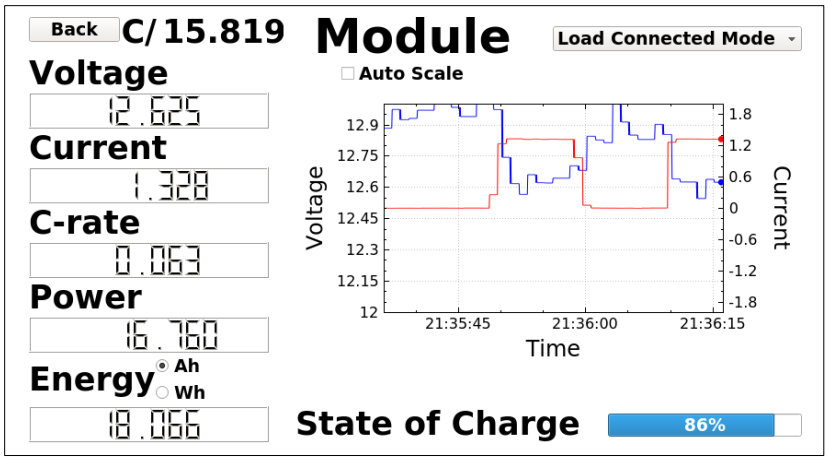

Figure 18: Screenshot of GUI Voltage and Current Measurements

4.3.2 Results of Verification Test 2

A screenshot of the oscilloscope measurements can be seen in Figure 19.

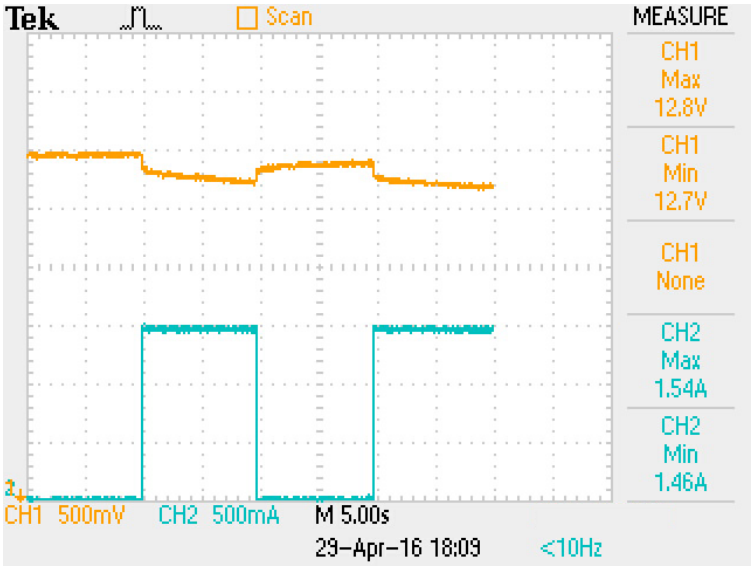

Figure 19: Screenshot of Oscilloscope Voltage and Current Measurements

The graph in Figure 19 was then compared with the screenshot of the GUI, shown in Figure 20, which also includes a graph of the voltage and current.

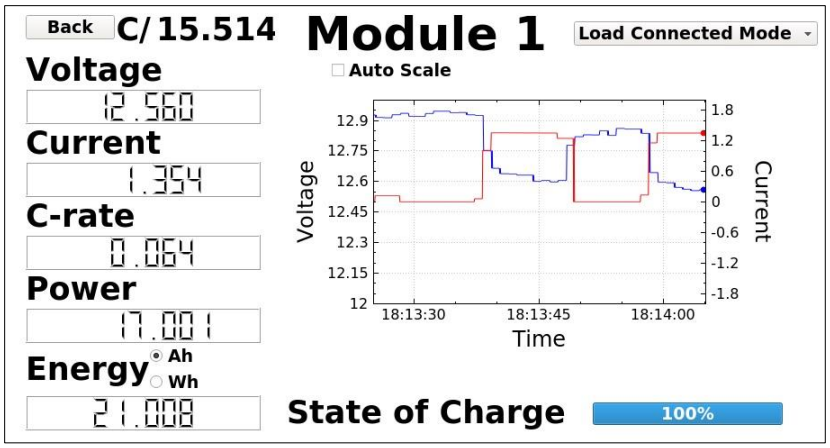

Figure 20: Screenshot of GUI Voltage and Current Measurements

Data was sent from the microcontroller to the GUI at a rate of once every second. Though the oscilloscope's data collection did not exactly meet up with the GUI's timing, the graphs comparing the results of the voltage and current measurements of the oscilloscope vs. the GUI can be seen in Figure 21 and Figure 22, respectively. In both figures, the blue line represents the measurements from the oscilloscope and the green line represents the measurements from the GUI.

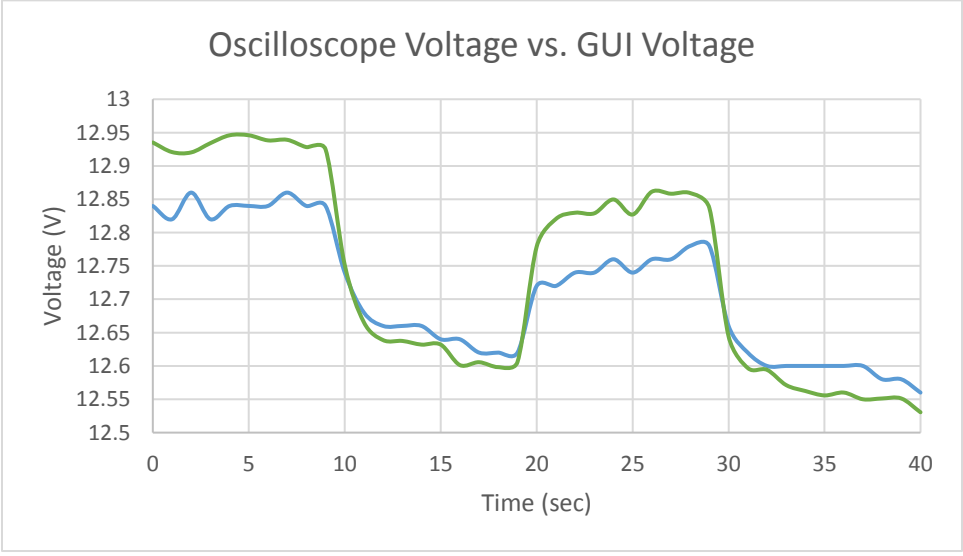

Figure 21: Voltage measurements from the oscilloscope (blue) vs. voltage measurements from the GUI (green)

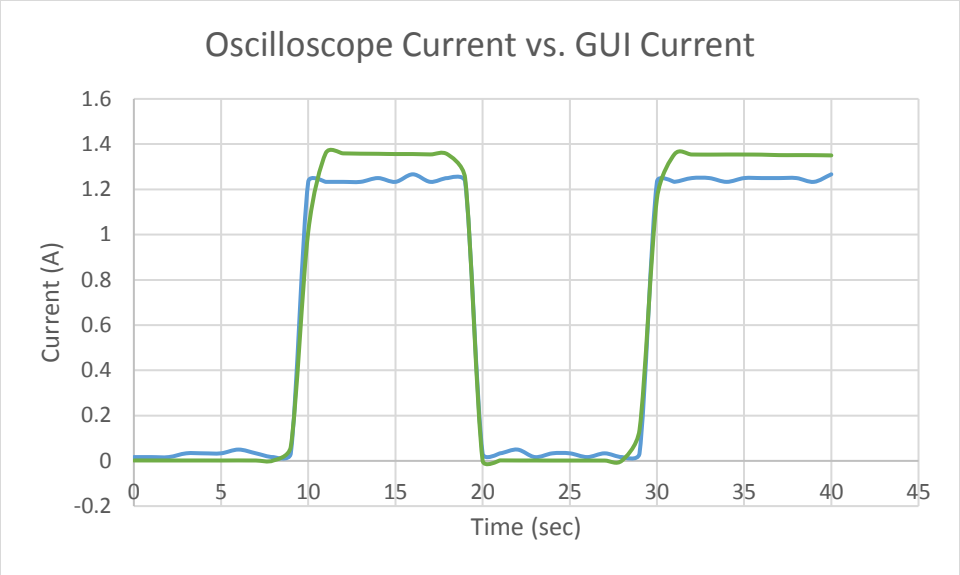

Figure 22: Current measurements from the oscilloscope (blue) vs. current measurements from the GUI (green)

Chapter 5: Discussion

This project has three major results: the designed graphical user interface for the battery management system, the designed communication scheme, and the results from the verification tests. These results will be discussed in the following sections.

5.1 Discussion of the Graphical User Interface

The graphical user interface clearly displays information about the load and each individual battery module. This allows the user to accurately manage the energy storage system by being able to make informed decisions.

The informative window for the load digitally displays the battery voltage, DC bus voltage, DC bus current, power, and energy at a precision of three decimal places. It also graphically displays the battery voltage, DC bus voltage, and DC bus current. If the user wishes, this graph could be auto scaled to best fit the given data. This window provides the user with an easy method of monitoring the entire system as a whole.

The informative windows for the individual battery modules digitally displays the battery's voltage, current, power, energy, and state of charge. The C-rate is provided in two formats to accommodate the preferences of all users. The standard decimal format in the digital box provides users with an accurate c-rate. The numerical number at the top of the screen (the reciprocal of the c-rate) provides the user with an easy method for determining how much longer until the battery is fully charged or discharged. For example, if the battery is discharging with a c-rate is C/20, the battery will be fully discharged in 20 hours. The voltage and current are also graphically represented so that changes can easily be seen over time. If the user wishes, this graph can also be

auto scaled to best fit the given data. With all the information available on one screen, this window provides the user with an easy method to monitor the individual battery modules in the system.

Based off this information, the user can then decide on the charging profiles for each individual battery module. Without leaving the modules informative window, the user can change the profile to any of the following: load connected mode, charging mode, and maintenance mode. If charging mode is selected, a new window will appear to allow the user to choice either pulsed or constant charging.

5.2 Discussion of the Communication Scheme

A custom communication scheme was designed and implemented in the battery management system. The ESMC scheme included three types of string commands: queries, commands, and configuration commands.

The query messages requested information about the battery system to be sent back to the GUI, specifically voltage and current information. Figure 13 shows how the target was set up using Waijung setup blocks. The figure also shows how the query messages are processed by the microcontroller. The UART Rx block looks for the specific query commands (ESMCVx and ESMCIx). Once the command is received it checks for the appropriate channel number, or the number of the battery the information is being requested on. Though the figure only shows one channel selection available for each query, the same procedure was easily replicated for multiple channels. Once the command is received and the channel specified, the enabled substation get the requested information off the appropriate pin from the Regular ADC block and sends the information back to the GUI using a UART Tx block (not pictured).

Command messages request an action by the microcontroller, specifically changing the mode that the specified battery is operating in. Figure 14 shows how Matlab Simulink was used to allow the microcontroller to process these commands. As previously mentioned, the modes are controlled by switch states being represented by LEDs. The UART Rx looks for the appropriate command and then checks for the battery number that the command applies for. Similarly, the channel selection was expanded to include multiple batteries though not pictured. S-R flip-flop blocks were used as a safety measure to hold the state of the batty and ensure that more than one state will never be selected. The modes are then sent to the substation shown in Figure 15.

Configuration commands set the parameters of an action, specifically the duty cycle and frequency of the pulsed charging mode. Figure 14 also shows this in Matlab Simulink. The command is received by the appropriate UART Rx block. These values are then manipulated so that they are in the appropriate format to be used by the programmable pulse generator. The pulse generator uses this information to create a pulse that is then sent to the subsystem shown in Figure 15 which is then passed to the pulse subsystem shown in Figure 16.

This subsystem, shown in Figure 15, assigns a number to each mode. This number is then looked up in the Direct Lookup Table block to determine the state of LED 1, 2, and 3 for that mode. The third LED is the only LED specifically affected by pulsed charging. Therefore, this lookup table assigns a new number to switch and another subsystem is used to account for this, shown in Figure 16.

This pulse subsystem determines whether or not this switch should be pulsed. It has three possible states: on, off, or pulsed. This subsystem determines which state it is in based of the previously assigned number and then uses another S-R flip-flop block to hold the state and prevent errors. If the state is selected as pulsed charging, the signal will be ANDed with the pulse to create

the pulse. This then enters a third subsystem similar to that of the first subsystem. In this subsystem, the states are assigned a number, and then this number is looked up in a Direct Lookup Table block to determine the state of the third LED.

5.3 Discussion of the Verification Test Results

The voltage and current measurements were passed accurately from the microcontroller over the serial port. These measurements were then displayed on the GUI with a precision of three decimal points. They were then used to calculate all of the other parameters on the GUI including: c-rate, power, and energy.

5.3.1 Discussion of Verification Test 1 Results

The square wave in Figure 17 represents the current measured by the oscilloscope. In reality however, it is the voltage across the shunt resistor. The oscilloscope does not account for the fact that the resistance is 1.2Ω. The STMicroelectronics Discovery microcontroller has been programmed to account for this as it divides the voltage by 1.2Ω to get the current. Therefore, the current square wave on the oscilloscope reads slightly higher than what the current actually is/ what is displayed on the GUI in Figure 18.

As can be seen in Figure 18, the voltage measurement on the GUI was accompanied by a significant amount of noise, while the current measurement had almost no noise. The measurements were taken from the microcontroller, sent to the GUI, and displayed on the GUI using the same method. The only variable to change was the pin on the microcontroller. Therefore, variations in voltage measurements on the GUI are a result of noise from pin of the microcontroller. This type of noise is normal; however, it is made worse by the voltage divider. To scale the voltage down so that it can be used by the microcontroller, the voltage is multiplied by 0.175. Once the

voltage is read by the microcontroller it has to multiply the voltage by the reciprocal, 5.7, to get back to the original voltage. Therefore all noise from the microcontroller is also amplified by a factor of 5.7.

5.3.2 Discussion of Verification Test 2 Results

The square wave in Figure 19 represents the current measured by the oscilloscope. In reality however, it is the voltage across the shunt resistor. Similar to test 1, the oscilloscope does not account for the fact that the resistance is 1.2Ω. Therefore, the current square wave on the oscilloscope is slightly higher than what the current actually is/ what is displayed on the GUI in Figure 20.

As can be seen in Figure 20, the voltage measurement is accompanied by some noise, while the current measurement has almost no noise. When this figure is compared with its equivalent, Figure 18, in verification test 1, it can be seen that the overall amount of noise from the pins has decreased as a result of the faster sampling time.

However, the noise does become more evident when directly comparing the oscilloscope measurements vs. that of the GUI in Figures 21 and 22. The current measurement from the oscilloscope has already been divided by 1.2 to account for the shunt resistor. In both figures, the GUI measurements are higher than that of the oscilloscope. This is due to the magnification of the noise by the microcontroller. The microcontroller divides current by 1.2 and the voltage by 0.175. The result is the noise being scaled by a factor of 0.833 and 5.7, respectively.

Chapter 6: Conclusion

Energy storage systems will play a major role in transforming the current power grid into a system that is more efficient, reliable, and economical. Increasing energy storage will allow more power plants to run at full capacity and quickly meet fluctuating demands. Since wind and solar energy supplies are constantly changing, the increased energy storage will also allow for increased implementation of renewable energy. Though there are many types of energy storage, batteries were determined to be the best choice for energy storage due to their versatility, high energy density, and efficiency. However, despite all this, battery energy storage systems currently account for only a small portion of the already limited existing energy storage in the grid.

To encourage battery storage integration, a battery management system is necessary to monitor and maintain safe, optimal operation of each battery stack. Accurate measurements and monitoring prevent from damaging the batteries and the overall system, along with making sure that the system is operating efficiently.

The graphical user interface for the battery management system designed is able to effectively manage and monitor the individual batteries in the energy system. It is accurate and user friendly so that the system can run optimally. The graphical user interface was designed on QT because of QT's small footprint, portability, and ease-of-use. The GUI was then uploaded onto a BeagleBoard, an embedded board.

A custom communication scheme was created and implemented into the GUI in order to exchange queries, commands, and configuration commands to the microcontroller. The microcontroller was programed to interpret the messages from the GUI using Matlab Simulink. This communication took place via a serial port, which was proven to provide accurate

communication. Variations that did arise during the verification test were a result of noise from the microcontroller, a STMicroelectronics Discovery, and were amplified when accounting for the effects of the voltage divider and shunt resistor.

Future work will involve verifying the serial cable as an effective means of accurate communication over long distances. This would also include implementing the GUI and custom serial communication scheme into the final BMS microcontroller. The microcontroller is currently being designed and created by PhD candidates at the Energy System Research Laboratory. This should significantly reduce the noise and further improve the accuracy of the system. The microcontroller will also include a way of getting the state of health of the battery based on the internal resistance.

Other future work will involve implementing state of charge and energy calculations on the microcontroller side, instead of the GUI. To do this, a new command will be created called Reset. This command will be "ESMCReset=%f1, %f2, %f3, %f4, %f5, %f6" and will be sent every time the setup window is reconfigured by the user. In which f1 is the type of battery (0 for lead acid and 1 for lithium ion), f2 is the number of cells in the battery, f3 is the capacity of the battery, f4 is the maximum voltage of the system, f5 is the minimum voltage of the system, and f6 is the maximum current of the system. Sending all the information at once will decrease the memory necessary on the GUI. This information will then be used to calculate the energy and state of charge of the battery on the microcontroller using Matlab Simulink. These values will then be sent back to the GUI when the appropriate query is requested. Moving these calculations off of the graphical user interface will increase the accuracy of these measurements and increase its available overhead.

Acknowledgements

We would like to acknowledge the funding support from the Office of Naval Research and the United States Department of Energy. The author would like to thank her faculty mentor, Dr. Osama Mohammed, and everyone at the FIU Energy Systems Research Laboratory, especially Ahmed Elsayed, Christopher Lashway, and Tarek Yousef.

Bibliography

[1] M. T. Lawder, B. Suthar, P. W. Northrop, S. De, C. M. Hoff, O. Leitermann, M. L. Crow, S. Santhanagopalan, and V. R. Subramanian, "Battery energy storage system (bess) and battery management system (bms) for grid-scale applications," *Proceedings of the IEEE*, vol. 102, no. 6, pp. 1014-1030, 2014.

[2] S. Yeleti and Y. Fu, "Impacts of energy storage on the future power system," in *North American Power Symposium (NAPS), 2010*. IEEE, 2010, pp. 1-7.

[3] G. Huff, "DOE Energy Storage Database, The Role of Storage in Energy System Flexibility," International Energy Agency Workshop, 2014.

[4] J. Garche and A. Jossen, "Battery management systems (bms) for increasing battery life time," in *Telecommunications Energy Special Conference, 2000. TELESCON 2000. The Third International*. IEEE, 2000, pp. 81-88.

[5] C. J. Barnhart, M. Dale, A. R. Brandt, and S. M. Benson, "The energetic implications of curtailing versus storing solar-and wind-generated electricity," *Energy & Environmental Science*, vol. 6, no. 10, pp. 2804-2810, 2013.

[6] N. Dung, "Use of iec 61850 for low voltage microgrids internship report," University of Twente and Alliander.

[7] "Global energy storage glossary," U.S. Department of Energy and Sandia National Laboratory.

[8] *U.S. Energy Storage Monitor: 2015 Year in Review Executive Summary*. Green Tech Media Research, 2016.

[9] L. Xu, Z. Miao, and L. Fan, "Control of a battery system to improve operation of a microgrid," in *Power and Energy Society General Meeting, 2012 IEEE*. IEEE, 2012, pp. 1-8.

[10] Z. Miao, L. Xu, V. R. Disfani, and L. Fan, "A soc-based battery management system for microgrids," *Smart Grid, IEEE Transactions on*, vol. 5, no.2, pp. 966-973, 2014.

[11] A. T. Elsayed, C. R. Lashway, and O. A. Mohammed, "Advanced battery management and diagnostic system for smart grid infrastructure," *IEEE Transactions on Smart Grid*, 2015.

[12] *A Guide to Understanding Battery Specifications*, 1st ed. MIT Electric Vehicle Team, 2008.

[13] A. Christensen and A. Adebusuyi, "Using on-board electrochemical impedance spectroscopy in battery management systems," in *Electric Vehicle Symposium and Exhibition (EVS27), 2013 World*. IEEE, 2013, pp. 1-7.

[14] D. Molkentin, *The Book of Qt 4: The Art of Building Qt Applications*, No Starch Press, 2007.

[15] R. Rischpater, *Application Development with Qt Creator*, Packt Publishing Ltd, 2014.

[16] J. Thelin, *Foundations of Qt Development*, Apress, 2007.

[17] L. Junyan, X. Shiguo, and L. Yijie, "Application research of embedded database sqlite," in *Information Technology and Applications, 2009. IFITA '09. International Forum on,* vol. 2. IEEE, 2009, pp. 539-534.

[18] F. Shengwen, C. Licong, and W. Chunxue, "Software design of wind power supervisory control system based on embedded qt," in *Electrical and Control Engineering (ICECE), 2011 International Conference on.* IEEE, 2011, pp. 3628-3631.

[19] *Qt Documentation QSerialPort Class,* The Qt Company Ltd., 2016.

[20] E. Eichhammer, *QCustomPlot*.

[21] *QtDocumentation Signals and Slots,* The Qt Company Ltd., 2016.

[22] C. R. Lashway, O. A. Mohammed, "Adaptive Battery Management and Parameter Estimation through Physics based Modeling and Experimental Verification" accepted in the IEEE Transactions on Transportation Electrification Special issue on Energy Storage. 21 Apr 2016.

[23] *Waijung Blockset,* Aimagin, 2015.