CONTENTS

Executive Summary ........................................................................................................................ 5
Introduction - Project Overview ..................................................................................................... 6
LPRDS Project Status ..................................................................................................................... 9
LPRDS 2010 - Technical Overview ............................................................................................. 10
  Raw Power Interface (RPI) ....................................................................................................... 11
  Energy Storage System (ESS) .................................................................................................. 12
    ESS Overview ....................................................................................................................... 12
    Per Cell Battery Management Research ............................................................................. 13
  Switch Controller (SC) ............................................................................................................. 15
    SC Overview ......................................................................................................................... 15
    Battery Management Control Algorithm ............................................................................. 17
    Characterization of Batteries ............................................................................................... 19
    Snubber Circuitry .................................................................................................................. 23
    PV modeling ......................................................................................................................... 25
  Filter Inverter Box (FIB) ........................................................................................................... 27
    Inverter .................................................................................................................................. 28
    Filter ...................................................................................................................................... 35
Supervisory Control and Data Acquisition (SCADA) .................................................................... 46
  Data Acquisition Board Design Changes ............................................................................. 46
  C++ API ................................................................................................................................ 47
  LPRDS Error Codes ............................................................................................................... 57
  SCADA Block Diagram Description ..................................................................................... 58
  Pico-LCD .............................................................................................................................. 62
  Database .................................................................................................................................. 63
  Demo App ............................................................................................................................... 64
System Safety ............................................................................................................................... 68
  Safety Loop ............................................................................................................................. 68
  SCADA Interface Box (SIB) Hardware ............................................................................... 69
  Tower Hardware: .................................................................................................................... 70
  Safety to Software Interface Board ....................................................................................... 71
EXECUTIVE SUMMARY

This report documents detailed technical sections describing the design decision that were made throughout the project. Even though not all system requirements were met, you will find that much research and progress has been made for the LPRDS system by the 2010 design team. The technical overview documents design decisions made by the team, as well as other aspects of the design changes made to the LPRDS system. The technical overview begins with descriptions of the legacy subsystems and the design changes made to them throughout the semester. The legacy subsystems are followed by the bulk of the technical overview. The bulk of the overview includes the subsystems and aspects of the LPRDS that were designed by the current design team. The subsystems designed by the 2010 design team include the FIB, SC, and SCADA. System safety and the SIB designs are also included in the technical overview.

The report then goes into a general project requirements section. This section of the final report consists of the documentation of some of the general project requirements associated with the LPRDS system. Included in this section are the budget, power budget, and the “ilities” reports. The ilities reports included within this section are environmental, EMI, hazmat, safety, reliability, maintainability, manufacturability, sustainability, and ethics.

The next section of the report is a maintenance manual that explains the unique technical principles and details of system operation. The maintenance manual also includes information on any advanced maintenance or calibration techniques that could be applied by an expert maintainer. A set of schematics, pin outs, interface control documents, communication protocols, cables assignments, PCB board layouts, and the semantics of all interfaces are documented. The report concludes with Appendices that document work that was referenced throughout the report. The appendices include Matlab files and spice decks that were used in our designing of the system. These files specifically address the filter and inverter design.
INTRODUCTION - PROJECT OVERVIEW

The Lafayette Photovoltaic Research and Development System is a multi-year, multi-team, senior Electrical and Computer Engineering capstone project that consists of the designing, implementing, and testing of a 2kW solar energy system. The main requirement of the system is to convert high voltage DC from the photovoltaic array to a 120V RMS AC signal of 60Hz. The project began in 2009, when students started designing a system that would use the energy acquired from the solar panel array mounted on the roof of the engineering building to power an AC load. Any excess energy not being used to power the load is stored within the system in the battery bank so that when the PV arrays are not producing enough energy to power the load, the difference is drawn from the energy stored in the battery bank.

The system’s main elements consist of a solar panel array, an Energy Storage System containing batteries, a Filter-Inverter Box containing an AC conversion system, a Raw Power Interface that connects the solar panels to the rest of the system, and a Switch Controller that regulates the high voltage path between the PV, the batteries, and filter/inverter. The system also includes a commercial grid-tie inverter that converts the DC electricity to AC so the photovoltaic power can be used by Lafayette College even when the system is not being used for student project work.

The LPRDS system is a requirements driven, team based, design project that is subject to a 30 page Statement of Work that includes 256 technical and management based requirements. There are dozens of project requirements for each component of the system with an emphasis on system integration. The main requirements include the automatic charge and discharge of the batteries, the delivery of 120V RMS, 60Hz AC electricity, the monitoring, storing and displaying of real time temperature, voltage, and current data from all subsystems, and demonstrated safety precautions to safely shut down the system if a fault is detected.

The 2010 design team has organized itself into smaller teams, typically of four or fewer people, to focus on the technical design of each subsystem. At the same time, the team members also fill management and systems engineering roles. This project has required constant effort from management and systems engineers to maintain communication between subsystem teams and make system level decisions, all while maintaining focus on the main project goals. Much of today’s engineering is system engineering. This design team has been given a unique opportunity that not many undergraduates get a chance to do. This project has provided many members of the team with leadership opportunities as well as exposure to a design team environment that smaller projects cannot replicate. At the beginning of the semester, the team was not always organized and on track to complete the project. However, the team developed a clear project schedule, strict budget, distinct goals, and a better idea of how to work efficiently both individually and as a whole and because of this, we were very close to our goal of system integration.

Not only are members of the design team being exposed to a large design team dynamic and leadership roles, but we were also exposed to both mechanical and safety issues. Most Electrical engineers have little exposure to mechanical issues such as board layout, subsystem box layout and assembly drawings. This project exposed and familiarized the design team to these important issues. The design team has also been exposed to electrical safety issues. Students are limited to only 30V and the lock out tag out system has been adopted by the professors to ensure that students cannot mess with the high voltage. Each subsystem includes
high voltage isolation that consists of a protective shield over the high voltage sections in each subsystem box.

The PV array consists of ten modules that were provided to the team. The PV modules are connected in series on top of the team’s engineering building and are used to convert solar energy to electrical energy. The power that the panels output varies based on temperature and insolation.

The Raw Power Interface accepts high voltage DC from the roof-mounted PV array and delivers it to the rest of the system. The RPI also is the safety hub of the system; it monitors current on high voltage lines and sets off a safety alarm if it detects a ground fault. All the subsystems are connected to a safety interface, and when a safety fault is detected anywhere in the system, the subsystems disconnect from high voltage and enter a fault state. Safety faults include a ground fault interruption, overheating in any subsystem, or the failure of any subsystem. The safety interface also includes high voltage isolation relays to prevent the conduction of high voltage.

Within the ESS is a battery bank containing sixty-four 3.2V Lithium Iron Phosphate Batteries organized in sixteen packs of four. These batteries are connected in series to produce a nominal voltage of 205V and are used to store the excess energy from the PV array that is not delivered to the load. The ESS also creates 12V and 5V from the batteries and provides these voltages to the other subsystems, where they are used to power chips and other circuitry.

The Filter Inverter Box serves as the AC conversion system. It consists of an H-bridge, a low-pass filter, and a microcontroller. Using these components, the FIB receives the high voltage DC from the Switch Controller and converts it to a 120V RMS AC signal of 60Hz. The Inverter is made up of an H-bridge and a microcontroller. The H-bridge is made up of four high-power IGBTs. It allows voltage to be applied across the filter input with two opposite polarities. Turning on the IGBTs in a specific order will apply a positive polarity and a negative polarity. The switching of the IGBTs is controlled by the signal applied at their gates, which is a pulse width modulated signal produced by the microcontroller. This switching creates the positive-negative pattern of AC current. The filter was added to the H-bridge in order to reduce the harmonic distortion in the output to produce a well-formed sinusoidal AC signal. An isolation transformer was also included between the FIB and the load. The transformer is necessary to reference the output of the FIB to ground and to isolate the connection from the circuitry of the FIB to the load.

The Switch Controller directs power between the RPI, the ESS, and the FIB via two high voltage switches. One switch disconnects the RPI and powers the load directly from the batteries, and the other disconnects the FIB in order to recharge the batteries. The Switch Controller switches are operated based on an algorithm. This algorithm is executed by the Supervisory Control and Data Acquisition subsystem using custom designed software. The Switch Controller algorithm is based on the voltage measured across the battery pack.

Switch A will remain closed until the batteries are charged up to 100%, where Switch A opens to prevent the overcharging of the batteries. Also as long as the battery charge is above 55%, Switch B will be closed to conduct power to the FIB, which means there will be enough power present to run the FIB. If the batteries are fully charged at 100% of their capacity, Switch A between the RPI and ESS will be opened in order not to overcharge the batteries. Switch A will remain open until the batteries are discharged to 65%, when it will close to recharge the
batteries before over-discharging occurs. If the battery pack discharges below 20%, Switch B will be opened in order not to over-discharge the batteries, directing all incoming power to the battery stack, enabling the batteries to charge. If the system goes into a fault state both switches will remain open until the fault is cleared. Once the fault is cleared, the system will continue operating based on the state of the system before the fault.

The Supervisory Control and Data Acquisition subsystem controls the higher-level operation and data collection of the other subsystems. The control of system functions is handled in four applications, the battery management app, the maintenance app, the demo app, and the system states app. Each subsystem contains a Data Acquisition Board that measures the relevant current, voltage, and temperature within the subsystem. Using a FitPC, SCADA polls each subsystem’s Data Acquisition Board for data and stores it in a database. Graphs and analyses are then generated to view and evaluate system operation and performance over time. SCADA also runs a website that provides a description of the system and displays the current system state.
LPRDS Project Status

The project currently consists of a completed and tested Raw Power Interface and Energy Storage System. The Filter Inverter Box and Switch Controller have been both designed and built along with a tower layout to hold the system. A low voltage integration test that involves all the subsystems except the FIB was begun. However, due to time constraints, we were not able to complete our testing plan. The FIB has also been tested. It has demonstrated the major requirement of 120VRMS and 60Hz, but it does have problems when changing the load of the output.

For more information on system testing, please see the ATP Report and the QA Report on the LPRDS 2010 website at sites.lafayette.edu/ece492-sp10 under the project documents section. These reports give complete documentation of the testing that was done on the LPRDS system by the 2010 design team.

The major requirements the 2010 design team achieved were the completion of the Raw Power Interface (contains main logic for safety), completion of the ESS, the 120VRMS 60Hz AC signal, the performing of supervisory functions on all subsystems, the logging of system data (sensors, states, and Sunny Boy) into the database and the website, the connecting of all subsystems to the safety interface, and the demo and system display.

The major requirements that we began to test, but were unable to achieve due to time constraints include the switch controller battery management algorithm, and the monitoring and usage of the operation states. We also wanted to incorporate power independence, but did not have enough time to implement.

The requirements we did not achieve include per-cell battery management within the ESS, standalone operation within the ESS, the measuring of the phase angle between voltage and current, and power factor, and the high voltage photovoltaic integration.

If given more time the design team would have obviously tried to meet all system requirements. We would also include some improvements in the system such as adding snubbers for when incorporating the PV array, system control via the website, maximum power point tracking, and demo touch sensors that allow the user to cycle through the demo app at their own pace and leisure.
LPRDS 2010 - Technical Overview

The Technical overview consists of the aspects of the Lafayette Photovoltaic Research and Development System that were designed by the 2010 design team. This section documents design decisions made by the team, as well as other aspects of the design changes made to the LPRDS system. The technical overview begins with descriptions of the legacy subsystems and the design changes made to them throughout the semester. The legacy subsystems are followed by the bulk of the technical overview. The bulk of the overview includes the subsystems and aspects of the LPRDS that were designed by the current design team. The subsystems designed by the 2010 design team include the FIB, SC, and SCADA. System safety and the SIB are also included in the technical overview.
Raw Power Interface (RPI)

The Raw Power Interface was designed by the 2009 LPRDS design team. Last year, the design team never tested its function, but we were led to believe the system worked by Professor Nadovich. With Professor Nadovich’s recommendation we decided to test the RPI to ensure its functionality. These tests showed that the Raw Power interface worked with correct functionality. Also we tested the Data Acquisition Board inside the RPI and found it had correct operation and did not need a redesign by the current design team. These tests can be found in the QA report.

Since the Raw Power Interface was designed by last year’s design team, this year’s design team will not give a technical overview of the design decisions made when creating the subsystem. However, a brief description of the system is necessary.

The Raw Power Interface (RPI) provides an interface between the solar panel array and the rest of the system. The RPI is the safety hub of the system; it monitors current on high voltage lines and sets off a safety alarm if it detects a ground fault. All subsystems are connected to a safety interface, and when a safety fault is detected anywhere in the system, all subsystems disconnect high voltage from their outside terminals and enter a fault state. Safety faults include a ground fault (detected in the RPI), overheating (detected by temperature sensors in each subsystem), and the failure of any subsystem (detected in each subsystem). The safety interface also includes high voltage isolation relays that are normally open to prevent the conduction of high voltage. These relays close only when there are no faults in the system. A more in depth safety report is included later within the system safety section in this report.
Energy Storage System (ESS)

ESS Overview

The Energy Storage System was designed by the 2009 LPRDS design team. Professor Nadovich told us that the subsystem had correct operation, but the data Acquisition Board had a few problems. The design team made minor changes to the Data Acquisition Board and fabricated a DAQ board PCB. More can be read about the design changes of the Data Acquisition Boards within the SCADA section of the report. The design team then tested the functionality of this subsystem with the Data Acquisition Boards. Professor Nadovich altered the battery bank to consist of a single battery stack of 12V and a multiple stack that consisted of 24V. Using this configuration within the battery bank, the design team tested the functionality of the ESS with the incorporated software within the DAQ boards. These tests showed that the Energy Storage System worked with correct functionality. For more information on these tests please visit the QA report.

Since the Energy Storage System was designed by last year’s design team, this year’s design team will not give a technical overview of the design decisions made when creating the subsystem. However, a brief description of the system is necessary.

The Energy Storage System (ESS) stores excess energy from the photovoltaic array that is not delivered to the load. Within the ESS is a battery bank containing sixty-four 3.2V Lithium Iron Phosphate Batteries organized in sixteen packs of four. These batteries are connected in series to produce a nominal voltage of 205V. The ESS also contains a DC/AC to DC converter used with the high-voltage DC from the battery bank or 120V AC from a wall outlet to produce 12V DC. The Data Acquisition board then steps the 12V DC down to 5V using a zener diode and a resistor. The 12V and 5V are provided to the other subsystems, where they are used to power chips and other circuitry.

Because the batteries should ultimately be protected from being either over-charged or over-discharged, the design team characterized the batteries by testing a single cell. The batteries should never be charged to over 100% of their capacity or discharged to less than 20% of their capacity. Without charge/discharge protection the batteries would be damaged, reducing their lifespan and capacity. The 2010 design implemented a new subsystem call the Switch Controller to prevent this from occurring. The Supervisory Control and Data Acquisition subsystem monitors the state of charge of the batteries, and runs the Battery Management Application that controls the charging and discharging of the batteries through the Switch Controller (SC).
Per Cell Battery Management Research

At the beginning of the Spring 2010 semester, the team considered implementing per-cell battery monitoring and per-cell battery management. Monitoring each cell rather than monitoring the battery pack as a whole provides more information about the battery pack, and allows an observer to pinpoint any problems more easily. Also, managing the charging and discharging of individual battery cells rather than the entire battery pack allows for more efficient use of each battery, since each can be charged or discharged to its fullest safe extent, rather than being limited by the weakest cells in the pack. If cells are not charged and discharged to their fullest safe limits, they actually lose efficiency because they lose the ability to be charged and discharged so fully. In addition, adding a means to bypass a cell once it becomes fully charged or discharged would permit the other cells to be used to their limits, once again using power more efficiently.

The team also researched methods of implementing per-cell monitoring and management. There exist some commercial systems that monitor and manage a large stack of batteries, but they can be very costly. It is also possible to design the circuitry and software necessary for accurate battery monitoring and management, but doing so would be a very complicated, difficult task. The third option is using a commercially available IC and designing the accompanying electronics and software. After some initial research, the team focused on this third approach. A chip that stood out was Linear Technologies’ LTC6802, which is capable of monitoring and balancing up to 12 Lithium-ion cells in series. Only 6 of these chips would be necessary to monitor and manage the entire 64-cell battery pack. The accompanying circuitry would also have to be designed, but the LTC6802 chip automatically handles much of the complicated process of monitoring and managing each cell.

The effects of implementing per-cell monitoring and management are clearly beneficial to the LPRDS project. However, there are drawbacks to implementing them as well. Specifically, the ESS and the wiring of the battery pack had already been designed and completed in 2009. The existing ESS provided connections at four points within the battery pack, which didn’t provide quite as much information about individual cells, but still provided more information than would be available from a single measurement across the battery pack. Adding per-cell capabilities would entail not only a completely new PCB design for the battery monitoring/managing chips such as the LTC6802, but also extensive redesign of the ESS wiring to allow for connections to each cell, and of the ESS itself to fit the new circuitry in the casing.

Additional methods for increasing efficiency would also be beneficial to the operation of the ESS. One option is maximum power point tracking (MPPT), which essentially adjusts the input to the batteries to maximize the power accumulated from the PV array. Implementing MPPT would require research on tracking algorithms as well as design of complicated tracking and compensating circuitry.

If these changes were made, the team would also have to test the operation of the redesigned ESS by prototyping at low voltage before moving to integrate per-cell monitoring and management with the entire ESS at high voltage. This would require a significant commitment of time and manpower from the project team.

After researching battery monitoring and battery management, considering the costs and benefits of implementing either or both on a per-cell basis, and considering the time and
manpower available this semester, the design team made the decision not to implement either per-cell battery monitoring or per-cell battery management this semester. Therefore, the team designed for SCADA to monitor voltage and current of the aggregate battery pack, but not of individual cells. Battery cells are neither charged nor discharged individually, and no method is provided to bypass individual cells.

However, the team recommends the use of the LTC6802 for use in the future. Based on the research and preliminary stages of design conducted this year, using this chip or another similar to it would allow for per-cell monitoring and management at relatively low monetary cost. Provided that a future design team can devote the effort to per-cell battery monitoring and management, implementing both would increase the efficiency of the operation of the ESS.
Switch Controller (SC)

SC Overview

The Switch Controller was designed by the 2010 design team. The purpose of the Switch Controller (SC) is to prevent the batteries in the Energy Storage System (ESS) from being overcharged or undercharged and also to dictate the flow of energy among the Photovoltaic (PV) array, ESS, and Filter Inverter Box (FIB). The SC design implements two high voltage switches on either side of the high voltage line going to the ESS. The state of switch A controls whether energy from the PV array can flow to the ESS and FIB and the state of switch B controls whether energy from the ESS can flow to the FIB. The Figure 6: Basic Switch Controller System block diagram below illustrates the location of switches A and B in relation to the rest of the system.

![Figure 7: Basic Switch Controller System](image)

The design as shown was decided after the consideration of adding a third switch between the ESS and the SC. This third switch would allow for situations in which it would be advantageous to disconnect the ESS while the PV array could still provide power to the FIB. For example, a situation where the batteries in the ESS are fully charged, yet the PV array is still providing sufficient power to supply a load connected to the FIB. In this example it would be more efficient to use the energy from the PV array to supply the load while disconnecting the ESS for later use.

The first problem this design raises is that of power point tracking. Power point tracking refers to setting the point on the PV array voltage-current curve that maximizes the power output. The IV curve relevant to the solar cells used in this design project is shown below.

As shown in the figure, changes in insolation and cell temperature can shift the IV curve. Due to the shifting nature of the IV curve, some method for adjusting the output voltage and current of the PV array would be preferred. One method of setting this point is to adjust the Thevenin resistance seen by the PV array to hold the voltage and current supplied by the PV
array, at maximum. This method requires additional circuit design to handle the implementation of such an algorithm and is not required by the statement of work. Another solution to maximizing the power supplied by the PV array is to hold the voltage at a point on the curve where the current will not change significantly with changes in cell temperature. This method provides a rough means of maximizing the power output of the PV array because the voltage will be set close to the maximum voltage and the current will change only with changes in insolation.

![Figure 8: PV IV Curve](image)

This method of maximizing power output from the PV array was preferred by the 2009 design team and so the ESS was designed with this in mind. The voltage across the battery stack within the ESS is used to hold the voltage supplied by the PV array at this point. The voltage across the ESS should vary between 165V and 235V depending on the state of charge (SoC), thus holding the voltage supplied by the PV array between this range on the flat portion of the PV array IV curve. With this design, the ESS can never be disconnected from the PV array while the PV array is supplying power because the ESS is required to set the voltage of the PV array. This constraint eliminated the possibility of adding a third switch within the SC controlling the flow of energy to and from the ESS. It is important to note however, that there is a switch within the ESS that can interrupt the connection of the ESS with the rest of the system in the case of a fault condition.
Battery Management Control Algorithm

The SC algorithm to be implemented in this design controls the two switches A and B, on either side of the ESS positive high voltage terminal. The logic for the algorithm is written in software within the FIT PC and communicates with the SC data acquisition board (DAQ) via an RS-485 Ethernet cable. The SC DAQ board relays control signals to switches A and B based on commands sent from the FIT PC. Also, the SC DAQ board relays voltage measurements from the ESS DAQ board to the FIT PC.

The logic for the SC algorithm is based in part on Tyson DenHerder’s work, described in his senior project paper *Design and Simulation of Photovoltaic Super System Using Simulink*. The algorithm considers the voltage of the battery stack within the ESS and determines the state of switches A and B based on predefined voltage threshold values. Switch A, controlling the flow of energy from the PV array to the ESS, is closed if the battery stack voltage discharges to 205V (65%) and is opened if the battery stack voltage charges to 235V (100%). Switch B, controlling the flow of energy from the ESS to the FIB, is closed if the voltage charges to 195V (55%) and is opened if the voltage discharges to 165V (20%). After switch B is opened, the FIB must ensure that the PWM generator is off until 10 ms after switch B closes again. The next states of switches A and B are based on the previous state of the switches. The table below shows the truth table used to determine the state of switches A and B based on the voltage measured on the battery stack.

<table>
<thead>
<tr>
<th>Condition</th>
<th>A’</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>V &gt;= 235</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>205 &lt; V &lt; 235</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>205 &lt; V &lt; 235</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>V &lt;= 205</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>B’</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>V &lt;= 165</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>165 &lt; V &lt; 195</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>165 &lt; V &lt; 195</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>V &gt;= 195</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 9: SC Truth Table*

Note: A’ and B’ denote the previous state of switch A and B, respectively.

It is important to note that these voltage thresholds were determined based on specifications from the battery cell manufacturer and verified through testing. Also, the testing done to verify the voltage thresholds was performed on only one 3.2V battery cell. The low voltage thresholds determined from testing were then extrapolated to determine the high voltage battery stack voltage thresholds. The figure below shows a complete system state transition diagram for switches A and B based on the SC algorithm with startup, shutdown, fault, and PWM generation conditions considered as well.

During safe shutdown, the order in which to open the switches is critical. The PWM must be turned off first, then switch B opened, then switch A opened. In the implementation of this algorithm, it was decided that there would be a 10 second delay between each event happening to ensure safety.

In the event of a fault, switch A and switch B open as well as a third switch within the ESS disconnecting the battery stack. When recovering from a fault, the switch within the ESS must be close first before all others. Then, depending on the battery voltage, if switch B is to be
closed, there must be a minimum of a 10 ms delay between the ESS switch and switch B closing to prevent spikes in current that may damage the high voltage switches.

*Figure 10: Complete SC Switching Diagram*
Testing of the SC algorithm idea described above was performed on a single 3.2V LiFePO₄ battery of the same type that is used in the ESS battery stack to test the idea’s plausibility. Figure 10: Charge/Discharge of Battery Test below shows the test setup. An adjustable discharge load box was used to discharge the battery cell as well as simulate a variable load attached to the ESS such as the FIB. A DC power supply was used to charge the battery cell as well as simulate power coming from the PV array. The battery voltage was monitored and plotted using a LeCroy Oscilloscope.

 Tests were performed to verify that the voltage thresholds corresponding to the SoC of the battery cell were accurate. Those voltage thresholds chosen for the algorithm proved to be approximately representative of the battery cell SoC. Similar tests were also performed to simulate various functions of the SC algorithm. It was determined that given the charge and discharge curves under various loading and supplying conditions, the algorithm would prove effective.

 Below are the results from testing done on 3.2V LiFePO₄ battery cells S-69 and S-70. The figure below shows the results from charging a single battery cell at 8A to the 3.65V (100%) threshold and then removing the DC power supply while beginning to discharge at 5A. At the 2.57V (20%) threshold, the load was removed and the open circuit (OC) voltage was measured after settling. At 3.04V (55%), the SC algorithm would close switch B (thinking the battery has charge) reconnecting the load simulated here at 5A. This test shows that at a discharge rate of 5A the OC voltage settled to 2.94V (47%) and therefore will not creep above the 55% threshold causing a false trip of switch B. Also, prior to discharge, this battery cell was charged at 8A to the 100% threshold and then the OC voltage was measured after settling. The 3.33V (76%) point

![Image](image-url)
is where the OC voltage settled to; which ensures that the SC algorithm would not incorrectly close switch A had the OC voltage fallen below the 3.20V (67%) threshold. This ensures that the SC algorithm would not incorrectly close switch A because 8A is the maximum charge current given the PV array characteristics.

**Figure 12: 8A Charge OC Voltage and 5A Discharge OC Voltage Test**

The figures below show the results of the same test performed on battery cells S-69 and S-70 respectively. In both tests the batteries were charged at 8A to the 3.65V (100%) threshold after which the supply current was removed. In the case of the first test, S-69, after the supply current was removed the battery’s open circuit voltage settled to 3.31V (74.8%) before the battery was discharged at 10A. At 2.57V (20%) the load was disconnected until the open circuit voltage became greater than 3.04V (55%) at which point the 10A load was reconnected simulating a false trip of switch B. At the 2.57V (20%) threshold the load was again disconnected and the open circuit voltage settled to 2.94V (47%). This test indicated that discharging the battery at a rate of 10A or greater would produce one false trip of switch B, but that any kind of oscillatory behavior would not be possible. The test was repeated on battery S-70 and more time was provided for the open circuit voltage of the battery to settle. Again, only one false trip of switch B resulted.
BATTERY: S-69

Charged Battery to 3.65V (100%), Voltage settled to 3.31V (74.8%),
Discharge @ 10A to 2.57V (20%)

Figure 13: 10A Discharge OC Voltage Test

BATTERY: S-70

Charge @ 8A to 3.65V (100%), Discharge @ 10A to 2.57V (20%),
@ 3.04V (55%) Discharge @ 10A to 2.57V (20%),
@ 3.04V (55%) Discharge @ 10A to 2.57V (20%)

Figure 14: 10A Discharge OC Voltage Test
The figures below show the results of overdischarging a single 3.2V LiFePO4 battery cell. After these tests were performed on battery S-69, no further testing was performed on this battery to ensure that a healthy battery was used in each test.

**Figure 15: 10A Discharge OC Voltage Test**

**Figure 16: 10A Overdischarge Test 2**
Snubber Circuitry

Snubbers are components such as inductors, capacitors, and diodes, added to circuits to prevent damaging current or voltage spikes from when a component is turned on or off. This is especially important in large voltage and high current circuits, such as those in the LPRDS system in the SC. If there is a large capacitive load attached to a switch, the initial current rush into the capacitor could be much larger than any of the components (especially the switch) can handle, and can cause large damage to the system and possibly things or people in the vicinity when it explodes.

Snubber circuits were considered and simulated for various conditions to prevent both current and voltage spikes. A Simulink simulation is shown below with the snubber composed of the inductor, resistor, and diode, as well as the throwback diode across the ground and the switch.

![Figure 17: Snubber](image)

The results of this simulation are shown in the charts below. These simulations show the effectiveness of adding a large inductor in series with the switch in preventing an extremely large initial current spike.
Figure 18: Snubber Simulation – Current through switch without snubber

Figure 19: Snubber Simulation – Current through switch with snubber

However, the practical aspect of using this inductor required a core of over four inches in diameter and hundreds of winds of large gauge (#11 or 12) wire, taking up a large amount of space. Further, it was shown in some simulations that by adding an inductor in front of the inverter, the Pulse Width Modulation scheme used would cause the output from the inverter to fall out of spec and cause distortion. Therefore, snubbers were not used in our system and another solution was devised in order to prevent large transients.

Instead, a resistor was placed in parallel with Switch B in the Switch Controller and the timing of the switches and the Pulse Width Modulation in the Filter Inverter Box were carefully controlled. The resistor keeps a large capacitor in the FIB charged so that when Switch B is closed, there is no large current spike to charge the capacitor.
PV modeling

We attempted to simulate the PV characteristics using Matlab. We found PV research by the University of Colorado in which they had modeled PV panels in Matlab. The basic PV cell model they used was a parallel connection of a current source, diode, and a resistor, plus another series resistor.

![PV Cell Model](image)

\[ I_D = I_0 \left( e^{V_D/V_T} - 1 \right) \]

The diode characteristics are:

Solving the KCL and KVL equations give us:

\[ I_{SC} - I_D - \frac{V_D}{R_P} - I_{PV} = 0 \]

\[ V_{PVcell} = V_D - R_s I_{PV} \]
The Matlab model of this system

![Figure 21: PV Model in Matlab](image)

However, there were two problems we had with implementing this model for our system. This model required both insolation and PV Current as inputs, and our system only had insolation as an input. We expected the PV to generate voltage and current based on insolation. The second problem we had was that we couldn’t connect this PV model to the rest of our modeled system (the batteries and the switches). We used Matlab’s SimPowerSystems model, which is an extension to Matlab’s Simulink system. Objects in SimPowerSystems do not naturally connect to Simulink objects.

After spending a significant amount of time unsuccessfully trying to fix these problems, we ultimately decided to stop modeling the PV and continue with the rest of the project.
Filter Inverter Box (FIB)

The Filter Inverter Box serves as the AC conversion system. It consists of an H-bridge, a low-pass filter, a microcontroller, and an isolation transformer. Using these components, the FIB receives the high voltage DC from the Switch Controller and converts it to a 120V RMS AC signal of 60Hz.

The Inverter is made up of an H-bridge and a microcontroller. The H-bridge is made up of four high-power IGBTs. It allows voltage to be applied across the filter input with two opposite polarities. Turning on IGBTs 1 and 4 will apply a positive polarity and turning on IGBTs 2 and 3 will apply a negative polarity. The switching of the IGBTs is controlled by the signal applied at their gates, which is a pulse width modulated signal produced by the microcontroller. This switching creates the positive-negative pattern of AC current. A low pass filter is added to the H-bridge in order to reduce the harmonic distortion in the output to produce a well-formed sinusoidal AC signal. An isolation transformer is also included between the FIB and the load. The transformer is necessary to reference the output of the FIB to ground and to isolate the connection from the circuitry of the FIB to the load.

Figure 22: Completed Switch Controller
Inverter

The inverter consists of four Insulated Gate Bipolar Transistors (IGBT’s). These are arranged in an H-bridge configuration. Each half of the bridge has a driver which drives both the low and the high side of the bridge. The PWM is a unipolar design, reference last year’s final report for details.

The circuit was simulated with Simulink. The SimPowerSystems package allowed us to add IGBTs and switches as well as passive elements to the simulation. We used the included PWM generation block to control the IGBTs. We simulated the whole system including filter, transformer, and a small load. We used this simulation to look at things like inrush current, as well as voltage drops and THD of the circuit. After running these simulations, we were confident in the operation of the h-bridge circuit. These simulations are attached.

We also tried to simulate in Smart Spice. We ran into some problems with the file for the driver we were using. The spice file was written in PSpice, and the simulator we have is closer to HSpice. We made some minor modifications to the deck until there were no errors when we tried to simulate. The circuit did not simulate at all, the output of the driver chips did change at all, even with the input changing. After a while of trying to simulate with the driver chips, we modified the deck to generate our own signals to ensure that the H Bridge was working, and we tried several things such as a step on the DC input to see what would happen. We were fairly confident in our circuits to proceed to making the PCB. These simulations are attached.

The PCB was first drawn as a schematic in DxDesigner. Once the schematic was completed, we were able to import the schematic into PADS. Since the two programs can connect easily we were able to import the schematics directly, and import any changed made to
the schematic after the board began to be designed. The specifications for the board were that there had to be a separation between the high and low voltage, and space around any of the high voltage lines. Keeping the biggest problem was making space for the high voltage traces. I tried to run these trace on both the top and bottom of the board so that they would not have to be as wide. The problem with this is that we could not run traces for the gate drives to some of the transistors. We remedied this by cutting the trace on the top and adding stitching vias where possible. We also tried to make the trace slightly wider at these areas to keep heat down. We figured out what the trace width needed to be for 20A, which is twice what the RMS current of the system was supposed to be. We kept close to this width were we could. We did mistakes, but nothing that needed a respin of the board.

The team last year built a prototype of this circuit, which had the problem of blowing up when a step was applied to the high voltage input. When the circuit did not blow up, the inverter created 120V at 60.5Hz. The circuitry on the high side of the board was reused all the way up to the optocouplers. The feedback circuit was redesigned to a simpler circuit from the complex circuit from last year that was untested.

The microcontroller that is the brains of the inverter is an Atmel Atmega128. This chip includes onboard PWM generation as well as UART and ADC. The PWM generator is used for controlling the inverter. The chip runs at 16MHz, so it is fast enough to do the PWM generation as well as communicate by RS485 with SCADA. Since the chip has a built-in UART, we only need to add a transceiver that has a level converter to work with the RS485. There is also an ADC on the chip, which we would have liked to use for feedback for closed loop design. This feedback is not yet functional because we did not have time to implement this closed loop design.

The big problem with the sine wave generation last year was that the frequency was a little higher than the specification set forth in the statement of work. The specification for frequency is 60 Hz ± 0.05%. This means the frequency needs to be within 0.03Hz. The previous design used a microcontroller providing a 10 kHz PWM, with the PWM updated by going sequentially though the sine table. 10 kHz does not divide evenly by 60Hz, so they were not able to get the precision needed to stay within specs. For this reason we had to find a different method to generate the sine wave for PWM. The fist idea that we had was to put a longer sine table into the device. We calculated that 3 full waves in a sine table would divide down exactly to 60Hz from the 10kHz PWM. Other ideas we had were to change the frequency of the PWM to something that would divide to 60 Hz. With our clock frequency at 16MHz, it was difficult to find a frequency that put us within the spec. The final idea that we implemented was the use of a Direct Digital Synthesizer.

The current design of the PWM uses a Direct Digital Synthesizer (DDS), which can programmed to any frequency in its range. A DDS consists of an adder, a register, and a signal table. The register and adder create an integrator. The output of the register is fed into the adder along with a value that will determine the output frequency. This is then fed back into the register. The upper bits of this register were used to index the signal table. In our case, we used a 10 bit sine table to make our signal. The DDS used had a 32 bit register, and at 10 kHz, we were able to achieve a resolution of:

\[ f_{step} = \frac{f_{clk}}{2^N}\frac{10kHz}{2^{32}} = 2.33\mu Hz \]
With this result, we can see that we can get well within our spec of 30 mHz. Next the value for adding to the register needed to be found to get our output frequency to right value. This value can be found with:

\[
\frac{\text{Increment} \times f_{\text{clk}}}{2^N} = \frac{\text{Increment} \times 10kHz}{2^{32}} = 60Hz
\]

\[\text{Increment} = 25,769,800\]

This result can be tweaked easily to get the correct value to account for any difference in the clock frequency in the processor.

The frequency specification was verified by the use of a signal generator and oscilloscope. The signal generator had a precision of around 100ppm worst case, which corresponds to .006 Hz. To make sure that we were within the .05% specified, we had to get our measurement to within .024Hz of the signal generator. The differential output from the filter and the signal generator output were compared on the scope. The signal generator output was help still by using the sync output of the generator. The other waveform slowly moved across the other one, due to the differences in frequency. By timing the duration of the waveform moving across the still waveform, we can get the difference in time. The time it took the waveform to make one full period across the other was at first around 30 seconds, which gave us a difference of just about our specification of .03 Hz. We changed the increment of the integrator slightly so that the time increased, meaning that we were closer to our desired frequency. The final difference was around 204 seconds, which corresponds to .005 Hz, well within the specification. The following is the setup of the frequency test:
The sine table in the device was generated with a Matlab script that accepted the maximum value and length of table, as well as modulation factor. The script then printed this out to a text file. We chose a 10 bit table to keep the quantization noise low in the device. The device has 128k of program memory that we could use for a sine table, so we were nowhere near the limit of the processors memory. The sine table was originally put into the data memory, which is only 4k. This was a limit on the size, but once we discovered how to get the processor to access program memory for data we had a much larger limit on the size. The table as generated can be put directly into the program.

The PCB was built to isolate anything that could have high voltage from the low voltage side. This was done using optical isolation, as well as a transformer and an isolated power supply. The power supply took the 12 volts from the ESS to 12V on the high side. The PWM signals had to be isolated as well. This was easily accomplished using optical isolators. We also had a feedback path for if we would have time to implement closed loop design. This was done using a transformer, since the primary and secondary coils are insulated from each other. The line dividing the low and high voltage was straight, to enable the requirement of a physical barrier to be placed between the high and low voltage sides.

The high side of the PCB consisted mostly of the H-bridge. This structure allows us to change the direction of voltage across a load, which is connected across the center of the structure. Each half of the bridge is driven by a driver. The gates on the low side of the H-bridge are easy to drive with the 12 volts on that side of the board. The problem is when we
have to drive the high side, the gate-emitter voltage must be high enough to turn the transistor on. This is a problem since when the high side is on, the emitter voltage is within a few volts of the high voltage coming in. This will cause the gate voltage to need to be higher than high voltage power coming in. The driver will allow that gate to be held high enough to keep the high side transistor on. The driver will also keep us from turning the high side and low side on at the same time. The driver incorporated dead time, a time when neither high nor low side is being driven. This will keep us from having two transistors in series from high voltage to ground, causing us to short out the high voltage power supply. They also allow us to have one control signal from the low side to the high side of the board, simplifying getting signals from low side to high side.

The IGBTs we originally used were rated for 600V at 39A. We thought that this would be high enough, as the rated current draw is around 10A for the system. The transistors ran as expected under low load, but did not function well with a load step. We tested the inverter with a switchable load, and when the load was switched we saw a large output current spike of three times the peak current. This was easily enough to break the IGBTs. We tested them again, but this time we did not switch a load at high voltage. We were able to run the inverter up to high voltage with no load, but when we loaded the inverter with a small load, the IGBTs shorted. We tried to parallel two of these parts, but the waveform was not a clean sine wave. We attributed this to uneven current sharing between the two parts, and ordered some 600V 90A parts. We put these in and ran the inverter up. They did not work well at first, but after a few modifications, the new parts worked the same as the old ones under no load. We ran the supply up with the low load again and we got a clean sine wave out at high voltage. We then went to the next higher load and ramped up the voltage. At about 160Vdc on the input we began to see a very erratic waveform on the HV input. This was centered at some zero crossing distortion. It seemed to not have anything to do with the amount of load, since we tried higher loads and the waveform was about the same. With this change in the IGBTs, we were pushing our gates to the limit of their current output. The gates charge and discharge just in time for the other side of the bridge to turn on or off. In the future, we can look at getting some different drivers with a programmable dead time or with a larger current output. We had also looked at getting some with a disable input to stop the PWM, but this can be done in the inverter easily. The following waveforms are for output and HV input. The yellow waveform is the HV input, magenta is current, and green in the output waveform.
With both of these IGBTs we saw distortion on the down slope of the sine wave at the zero crossing. We did not have time to find where these were coming from. My thoughts are that it is caused by something with the PWM. To test this I would take a close look at the PWM around the zero crossing. From this I would try to see if there was anything that might account for the sine wave distortion such as PWM dead time violated or something else. If this is not the problem, then we might have to take a closer look at the algorithm of the PWM generation. The output wave looks like the following under a load:
The system used to supply 12V and 5V power to the system. The 5V power was generated from the ESS DAQ board. We wanted to not have to rely on the DAQ board in the ESS for power, so we added a 5V converter. This converter was sized to be able to supply over twice the current needed. This way, we only need 12V power to power our whole system.
The filter design that was picked was a low pass filter in order to filter out the 10 KHz PWM wave generated by the microcontroller chip. To create this low pass filter, first we connected inductors on both output nodes of the H-Bridge. Then these inductors are connected together by a capacitor. The nodes of the capacitor are the outputs of the FIB box, which will go to the transformer. A schematic of the filter and transformer is shown below. In the schematic we added 60 milliohms in front of the inductor to represent the internal resistance of the inductor.
The process of creating the filter is as followed. First, the filter and transformer was simulated in an AC sweep mode using SmartSpice. As we simulated the filter, the size of the capacitor was kept constant as we varied the value of the inductors as well as varied the resistive load. The reason for keeping the capacitor constant was not to have too many varying variables. The value of the capacitor that was used was 30 µF, which was chosen from last year’s filter. Our goal for the filter was to have a -45dBV at 10 KHz. The resistive load was changed in order to simulate a load with both maximum and minimum current, which means the maximum and minimum power delivered. The resistor value for simulating maximum current is 12 ohms and the value for simulating minimum current is 360 ohms. The value of inductors that returned reasonable result was 500 µH. The plots are shown below.

![Figure 29: Simulation of filter and transformer with max current](image)

![Figure 30: Simulation of filter and transformer with minimum current](image)
As seen in the figure above, there is a large peak at 669 Hz with a value of 28 dB. This peak was ignored due to the fact that the peak is located around the eleventh harmonic of the 60Hz, which would be irrelevant. Also the peak is not near the 10 KHz frequency that the filter has to filter out. This large peak will not affect the output of the FIB box.

The next thing that was tested was if the filter met the total harmonic distortion requirement. A requirement for the filter is that the THD has to be below 3 percent. In order to accomplish this we first simulated the H-Bridge without a filter in Matlab. The goal was to figure out how much the THD actually is without the filter. The THD is 25 percent without the filter. The simulation result is shown below. Next, we simulated the H-Bridge with a filter in Matlab. The goal was to figure out the THD with all the components of the FIB box. From the simulation, the THD with the filter is equal to 1.52 percent. This means that the filter meets requirement of 3 percent THD. The simulation result is shown below.

![Figure 31: THD simulation without filter](image1)

![Figure 32: THD simulation with filter](image2)
The final process was to simulate the H-Bridge with a filter in Matlab again. The goal was to figure out if the filter meets GPR0003: EMI/EMC requirement. This requirement states that at 150 KHz, the output voltage has to be below -60 dBmV. This means that the voltage at 150 KHz has to be below 2mV. The simulation result is shown below.

![Figure 33: Output Sign Wave of the FIB Box and Power Spectrum](image)

From our simulations we found that the highest dBV around that area was at -85.12 dBV. This translates to .055mV. This means that we have met the spec for the GPR0003: EMI/EMC requirement.

After the size of the inductor was selected and passed all the tests, we had to look up vendors that sold that specific size inductor core. Selecting a core and how to wind it was a fairly involved decision. In the end, we chose the highest grade material for the core and a thick wire to wind the cores with. The drawbacks to this choice are clear: it is expensive to buy and difficult to wind. However, we felt it was the choice that gave us the best chance to meet the specifications for which the filter was responsible.

To make our selection, we used a program available off of micrometals.com that, given user input as to the requirements, returns a number of results that use different cores and windings to meet the requirements entered. Looking through this list we immediately were able to eliminate most of the options due to temperature concerns (our box is not supposed to get hotter than 40 degrees C). Another thing we looked at was the inductance vs. current graph. In an ideal inductor this would be a flat line, however in reality it does change and so we looked for a core with a line as flat as possible so that the performance of our filter would not change based on the
load. Core loss and copper loss were also a concern, though this had no impact on any specific requirement.

One of our main concerns was hysteresis. The filter designed last year introduced a number of harmonics and distortion into the sine wave that came from the inverter. This was shown in an experiment by Professor Nadovich last year, as he hooked the inverter up to wall power and then recorded and analyzed the output. Knowing the filter was responsible for introducing distortion, he spent time in the “off-season” researching the problem. He was able to model the effects of the hysteresis of an arbitrary inductor on a sine wave in SmartSpice, and the simulation showed a distorted wave that very closely resembled the output from the filter last year.

So this year in deciding what core to purchase, we intended to first model the effects of hysteresis in Matlab. Our model for the filter design is shown below:

![Figure 34: Model of the Filter and Transformer](image)

We then decided to model the effects of hysteresis from a cheap core against the more expensive one to determine if it made a difference. We were able to model the effects of the hysteresis of our specific cores by using the electrical and magnetic properties that were given in the core selection program from Micrometals (this takes into account how the core is wound), as well as the generic B-H curves for the material we were modeling. The results are shown below.
The results were decidedly inconclusive. The distortion in the beginning could just be start-up transients, as both of them smooth out after a few milliseconds. However, the only difference between the simulations was the hysteresis model, and there is some improvement in the expensive core over the cheap one.

We decided to go with the expensive core. Professor Nadovich’s off-season experiment gave us good reason to believe that hysteresis of the inductors was to blame for the distortion introduced by the filter last year. We attempted to model this, and did not see results that looked similar to the output last year, but even in our simulation the more expensive core performed better. There is also the possibility that our software might not be accurately simulating the physical phenomenon of hysteresis. Either way, the hysteresis model we calculated for the more
expensive core was more linear and extended out to a much higher saturation current. So if hysteresis is really to blame for the distortion last year, then this is the core that has the highest probability of resolving that problem.

The core material and winding that were selected had the lowest temperature rise, the lowest core and copper loss, one of the flattest inductance vs. current curves, and gives us the best chance of getting a clean sine wave out of our filter. The cost, while high in relation to the cheaper inductors, was still only $13.25, well within the budget allotted for the filter and inverter box.

Testing:

With our cores selected, the next step was to build the filter and begin testing. In the first stage of testing we wanted to see the frequency response of the filter to see if it matched our simulations. We tested for two condition, max current and minimum current. We changed the input peak-to-peak voltage through the filter to see if it had an effect on the frequency response, and it was seen that it did not have any significant effect on the output. So with that being said, below is the graph as well as the data we used for the maximum current experiment.

Comparing this to the simulation of the filter and transformer with max current shown above, it can be seen that the simulated -3dB frequency and overshoot were very close to the experimental
results. We then tried with the 360 Ohm load to simulate the minimum current. These results are below.

<table>
<thead>
<tr>
<th>Freq.</th>
<th>Amplitude of Input</th>
<th>Amplitude of Output</th>
<th>Freq.</th>
<th>Ratio in dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>8.8</td>
<td>8.06</td>
<td>30</td>
<td>-0.762952607</td>
</tr>
<tr>
<td>60</td>
<td>8.1</td>
<td>7.96</td>
<td>60</td>
<td>-0.173298229</td>
</tr>
<tr>
<td>90</td>
<td>7.2</td>
<td>7.19</td>
<td>90</td>
<td>-0.012072121</td>
</tr>
<tr>
<td>300</td>
<td>3.4</td>
<td>3.44</td>
<td>300</td>
<td>0.1013590511</td>
</tr>
<tr>
<td>600</td>
<td>1.6</td>
<td>1.85</td>
<td>600</td>
<td>1.400757332</td>
</tr>
<tr>
<td>700</td>
<td>0.625</td>
<td>1.96</td>
<td>700</td>
<td>8.482521524</td>
</tr>
<tr>
<td>800</td>
<td>0.294</td>
<td>1.47</td>
<td>800</td>
<td>13.97940097</td>
</tr>
<tr>
<td>900</td>
<td>0.088</td>
<td>1.34</td>
<td>900</td>
<td>23.65244252</td>
</tr>
<tr>
<td>1000</td>
<td>0.27</td>
<td>1.2</td>
<td>1000</td>
<td>12.95049464</td>
</tr>
<tr>
<td>1100</td>
<td>0.497</td>
<td>1.1</td>
<td>1100</td>
<td>6.900725928</td>
</tr>
<tr>
<td>1200</td>
<td>0.713</td>
<td>1</td>
<td>1200</td>
<td>2.938209403</td>
</tr>
<tr>
<td>1300</td>
<td>0.919</td>
<td>0.918</td>
<td>1300</td>
<td>0.17774654</td>
</tr>
<tr>
<td>1400</td>
<td>1.094</td>
<td>0.844</td>
<td>1400</td>
<td>-2.53497657</td>
</tr>
<tr>
<td>1500</td>
<td>1.28</td>
<td>0.813</td>
<td>1500</td>
<td>-3.94238848</td>
</tr>
<tr>
<td>1600</td>
<td>1.44</td>
<td>0.78</td>
<td>1600</td>
<td>-5.32535778</td>
</tr>
<tr>
<td>1700</td>
<td>1.64</td>
<td>0.76</td>
<td>1700</td>
<td>-6.68665115</td>
</tr>
<tr>
<td>1800</td>
<td>1.8</td>
<td>0.7</td>
<td>1800</td>
<td>-8.20348032</td>
</tr>
<tr>
<td>1900</td>
<td>1.95</td>
<td>0.63</td>
<td>1900</td>
<td>-9.61388123</td>
</tr>
<tr>
<td>2000</td>
<td>2.11</td>
<td>0.625</td>
<td>2000</td>
<td>-10.5604876</td>
</tr>
<tr>
<td>3000</td>
<td>3.41</td>
<td>0.45</td>
<td>3000</td>
<td>-17.5903837</td>
</tr>
<tr>
<td>4000</td>
<td>6.09</td>
<td>0.281</td>
<td>4000</td>
<td>-26.7382194</td>
</tr>
<tr>
<td>5000</td>
<td>7.6</td>
<td>0.25</td>
<td>5000</td>
<td>-29.6574767</td>
</tr>
<tr>
<td>6000</td>
<td>9.8</td>
<td>0.188</td>
<td>6000</td>
<td>-34.3419404</td>
</tr>
<tr>
<td>7000</td>
<td>10</td>
<td>0.188</td>
<td>7000</td>
<td>-34.3103401</td>
</tr>
<tr>
<td>8000</td>
<td>10.19</td>
<td>0.188</td>
<td>8000</td>
<td>-34.6803256</td>
</tr>
<tr>
<td>9000</td>
<td>10.19</td>
<td>0.188</td>
<td>9000</td>
<td>-34.6803256</td>
</tr>
<tr>
<td>10000</td>
<td>10.19</td>
<td>0.188</td>
<td>10000</td>
<td>-34.6803256</td>
</tr>
</tbody>
</table>

*Figure 38: Simulation Results for 360 Ohm load*

These results also closely match the simulations for overshoot, -3dB frequency, and attenuation at 10KHz. So our stand alone testing of the filter was considered a success in that we saw the exact kind of response from our filter that we were expecting.

Our next tests were part of the acceptance criteria. We wanted the filter to introduce less than 1% THD into the system (as measured by using the wall outlet as a zero-THD input) and also to remain below 40C. A summary explaining exactly how this experiment was set up is available in the filter ATP. The results are below.
The results of this experiment were also good as the THD introduced by the filter was calculated to be less than 0.1% and the temperature of the inductors after one hour of operation was less than 40°C.

In the last stage of testing, we tested the filter and inverter together to ensure that it met the acceptance criteria for the FIB as a whole. To do this, the THD out of the FIB needed to be less than 3%, and pass the FCC regulations for conducted emissions. The results of the high voltage testing for THD are below.
These results show a THD of $0.157\%$, well below our limit of 3%. As for the conducted emissions requirements, these stipulate that there should be no signal above 66dB μV (2mV) over 150KHz when viewed at 9KHz resolution bandwidth. These results are seen below.
These results show that we do not pass this requirement. However, we have several reasons to believe that this was a problem with our testing equipment rather our design. For one, a 100mV noise was seen coming out of the transformer with no input. This is more than enough noise to throw off our results. When the power spectrum is viewed with a finer resolution bandwidth, the sidebands become apparent and the spectrum begins to resemble noise. And it also makes sense that it would be a problem with our equipment because the signal out past 150KHz in our simulations was nowhere near 2mV.

Conclusion:

Overall, the design and testing of our filter and the results of our complete FIB system were a success. We met all the important specifications (with the possible exception of the conducted emissions requirement) and in many cases exceeded the requirements by a significant amount. Our experimental data matched the results of our simulations, providing us with a good deal of confidence in the current and future performance of the FIB system.
Supervisory Control and Data Acquisition (SCADA)

Data Acquisition Board Design Changes

The Data Acquisition Boards (DAQ) used in the LPRDS-BMS 2010 are very similar to the DAQs used in the 2009 project. Changes from the 2009 DAQ boards include:

- An isolated DC-DC converter to provide 5V High from the 12V Low
- An op amp to boost the signal from the current sensor to the PIC processor
- A diode to protect the circuit from voltage fluctuations
- A leopard paw print
- The removal of the high-voltage side DC-DC converter

Further, it was found that additional changes to the board had to be made after they were fabricated. These changes include:

- R38, R39, R40 not installed (open circuit)
- C9, C10 jumpered across (short circuit)
- Zener diode used instead of the DC-DC converter (U4) to generate 5V Low
  - 750 Ohm, 0.25W resistor from 12 Low on U4 to 5V Low on U4
  - 5V Zener diode in place of R15 (oriented with line on diode away from RTN)
C++ API

The Lafayette Photovoltaic Research and Development System has a fully functional C++ API that can be used by applications to completely control the LPRDS system. These API functions are listed and explained below.

- **connect(address)**
  
  *Return Type:* int
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.
  
  *Parameters:*
  
  - **address:** A string object to be used as in creation of the communication pipe between the application and the LPRDS Kernel.
  
  *Description:*
  
  - This method must be called immediately upon startup of an application that uses the LPRDS Kernel. The LPRDS Kernel must already be running in order for this method to work fully. When this method is called, a communication pipe is created to receive messages from the LPRDS Kernel. Additionally, a message is sent to the LPRDS Kernel through the kernel communication pipe in order to register the application with the kernel process.

- **disconnect()**
  
  *Return Type:* none
  
  *Parameters:*
  
  - none
  
  *Description:*
  
  - This method must be called before termination of an application that is registered with the kernel process. When called, this method sends a packet through the kernel communication pipe in order to remove the application from the kernel registry. Additionally, the communication pipe that was created at the start of the application by use of the connect method is removed.

- **getAddress()**
  
  *Return Type:* string
  
  - The returned value is a string object containing the application address that was used by the connect method.
  
  *Parameters:*
  
  - none
  
  *Description:*
  
  - This method checks for and returns the address used by the application when communication with the LPRDS Kernel process.
• getDeviceList()

Return Type: list<string>

- The returned value is an STL List container of string objects. The string objects are the names of each active device recognized by the LPRDS Kernel process. The device names are those specified in the LPRDS Hardware Description XML file.

Parameters:
- none

Description:
- This method communicates with the LPRDS Kernel process through the kernel communication pipe. A packet is sent requesting the name of each active device (sensors and switches) in the system. After sending this packet, the application waits for a response from the kernel process. The response is sent from the kernel process via the application’s communication pipe that was established upon use of the connect method. The received packet is then parsed into the STL List of string objects, containing the device names, which is available for use by the application. This method has two corollary methods, getSensorList and getSwitchList, that perform the same actions except requesting only sensor names and only switch names, respectively.

• getSensorList()

Return Type: list<string>

- The returned value is an STL List container of string objects. The string objects are the names of each active sensor recognized by the LPRDS Kernel process. The sensor names are those specified in the LPRDS Hardware Description XML file.

Parameters:
- none

Description:
- This is a corollary method to the getDeviceList method. It communicates with the LPRDS Kernel process through the kernel communication pipe. A packet is sent requesting the name of each active sensor in the system. After sending this packet, the application waits for a response from the kernel process. The response is sent from the kernel process via the application’s communication pipe that was established upon use of the connect method. The received packet is then parsed into the STL List of string objects, containing the sensor names, which is available for use by the application.

• getSwitchList()

Return Type: list<string>

- The returned value is an STL List container of string objects. The string objects are the names of each active switch recognized by the LPRDS Kernel process. The switch names are those specified in the LPRDS Hardware Description XML file.
Parameters:
- none

Description:
- This is a corollary method to the getDeviceList method. It communicates with the LPRDS Kernel process through the kernel communication pipe. A packet is sent requesting the name of each active switch in the system. After sending this packet, the application waits for a response from the kernel process. The response is sent from the kernel process via the application’s communication pipe that was established upon use of the connect method. The received packet is then parsed into the STL List of string objects, containing the switch names, which is available for use by the application.

- **getDetails(deviceID)**

  **Return Type:** vector<string>

  - The returned value is an STL Vector container of string objects. The string objects are the device details. Depending upon what type of device the details were requested for, the vector will be of two different sizes. If the device details were requested for a sensor device the vector will be of length 10 and if the details were requested for a switch the vector will be of length 5. The following diagram, Figure 1, shows a visual representation of the possible formats of the vector returned by this method.

  ![Diagram showing formats of getDetails return vectors](image)

  **Figure 43: Format of getDetails return vectors**
- **deviceID**: A string object representing the name of the device for which details are to be requested.

**Description:**
- This method sends a packet to the LPRDS Kernel process through the kernel communication pipe requesting the details of the device specified by the parameter `deviceID`. Valid device names are those specified in the LPRDS Hardware Description XML file and are available by using the `getDeviceList` method or its two corollary methods, `getSensorList` and `getSwitchList`.

- **getUnits(sensorID)**

  **Return Type**: string
  - The returned value is a string object of the units of the sensor device specified by the parameter `sensorID`.

  **Parameters**:
  - **sensorID**: A string object representing the name of the sensor for which the units are requested.

  **Description**:
  - This method communicates with the LPRDS Kernel process to request the units of the specified sensor. The units parameter of the sensor is established in the LPRDS Hardware Description XML file. In the case that the specified sensor name is not valid, the string object returned will be “no_device”. If the device specified exists but is not a sensor the returned string will be “not_sensor”. If the device is a sensor but has no units specified the returned string will be “no_units”.

  **CHECK THE RETURNED VALUES IN CASE OF ERROR HERE**

- **getValue(sensorID)**

  **Return Type**: float
  - The returned value is either the sensor reading, after adjustment by the scale and offset fields (found in the LPRDS Hardware Description XML file) by the kernel process, or an error code; see LPRDS Error Codes in Appendix A.

  **Parameters**:
  - **sensorID**: A string object representing the name of the sensor for which the units are requested.

  **Description**:
  - The method communications with the LPRDS Kernel process to poll a sensor, specified by the parameter `sensorID`, in the system. The method returns either the sensor reading or an error code (as a float value). The error codes can be found in Appendix A.

- **getValue(daq,channel)**
Return Type: float
- The returned value is either the sensor reading, before adjustment, or an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- daq: An int value which is the address of the data acquisition board (DAQ) with the sensor to be read.
- channel: An int value which is the A-D channel number on the data acquisition board (DAQ) where the sensor is connected.

Description:
- The method communications with the LPRDS Kernel process to poll a sensor, specified by the parameters daq and channel, in the system. The method returns either the sensor reading or an error code (as a float value). The error codes can be found in Appendix A.

• getLastError(sensorID)

Return Type: float
- The returned value is either the previously recorded sensor reading from the LPRDS database or an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- sensorID: A string object representing the name of the sensor for which the units are requested.

Description:
- The method communications with the LPRDS Kernel process to request the most recent reading from a sensor, specified by the parameter sensorID, in the LPRDS database. The method returns either the sensor reading or an error code (as a float value). The error codes can be found in Appendix A.

• setSwitch(switchID,level)

Return Type: int
- The returned value is an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- switchID: A string object representing the name of the switch to be set.
- level: The level is an integer 1 or 0 representing on or off, respectively.

Description:
- This method operates a switch in the LPRDS system, specified by switch name, by sending a packet to the LPRDS Kernel through the kernel communication pipe. The value returned by this method is an integer error code.

• setSwitch(daq,pin,level)

Return Type: int
- The returned value is an error code; see LPRDS Error Codes in Appendix A.
Parameters:
- **daq**: An int value that is the address of the DAQ that has the switch to be set.
- **pin**: An int value that is the pin number of the switch on the DAQ.
- **level**: The level is an integer 1 or 0 representing on or off, respectively.

Description:
- This method operates a switch in the LPRDS system, specified by DAQ and pin number, by sending a packet to the LPRDS Kernel through the kernel communication pipe. The value returned by this method is an integer error code.

- **addDevice(switchID,daq,pin,details)**

  Return Type: int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- **switchID**: A string object representing the name of the switch.
- **daq**: An int value that is the address of the DAQ that has the switch.
- **pin**: An int value that is the pin number of the switch on the DAQ.
- **details**: A string object that contains a brief human readable description of the device. No spaces are permitted in the description.

Description:
- This method allows an application to add a new switch to the LPRDS system. In order to do this a packet is sent through the kernel communication pipe containing the device parameters specified by switchID, daq, pin, and details. The LPRDS Kernel process updates the LPRDS Hardware Description XML file and updates the internal device lists to include the new switch. The kernel process sends a packet to through the application’s communication pipe acknowledging the success or failure of the addition. The packet contains an error code; these error codes can be found in Appendix A.

- **addDevice(sensorID,type,units,offset,scale,daq,channel,max,min,details)**

  Return Type: int
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

Parameters:
- **sensorID**: A string object representing the name of the sensor to be added to the system.
- **type**: A string object representing the type of sensor to be added. Valid sensor types are temperature (“temp”), voltage (“volt”), or current (“curr”).
- **units**: A string object representing the units of the sensor to be added.
- **offset**: A float value that is the offset to be applied to sensor readings.
- **scale**: A float value that is the scale factor to be applied to sensor readings.
- **daq**: An int value that is the address of the DAQ that has the sensor.
- **pin**: An int value that is the channel number of the sensor on the A-D on the DAQ.
- **max**: An int value that is the maximum safe reading of the sensor. A reading greater than this value will trigger a fault in the LPRDS system.
- **min**: An int value that is the minimum safe reading of the sensor. A reading below this value will trigger a fault in the LPRDS system.
- **details**: A string object that contains a brief human readable description of the device. No spaces are permitted in the description.

**Description:**

- This method allows an application to add a new sensor to the LPRDS system. In order to do this a packet is sent through the kernel communication pipe containing the device parameters specified by sensorID, type, units, offset, scale, daq, channel, max, min, and details. The LPRDS Kernel process updates the LPRDS Hardware Description XML file and updates the internal device lists to include the new sensor. The kernel process sends a packet to through the application’s communication pipe acknowledging the success or failure of the addition. The packet contains an error code; these error codes can be found in Appendix A.

- **removeDevice(deviceID)**

  **Return Type**: int

  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  **Parameters**:

  - **deviceID**: A string object representing the name of the device to be removed from the system.

  **Description**:

  - This method allows an application to remove a device from the LPRDS system. To do this a packet is sent through the kernel communication pipe containing the name of the device to be removed, specified by deviceID. An error code is sent back to the application through the application’s communication pipe. The error codes for the LPRDS system are explained in Appendix A.

- **updateDevices()**

  **Return Type**: int

  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  **Parameters**:

  - none

  **Description**:

  - The method sends a packet to the LPRDS Kernel process instructing it to re-scan the LPRDS Hardware Description XML file. This method can be used to update the kernel process’ internal device lists if the hardware description file has been updated manually or by another means than the supplied addDevice and removeDevice methods. (WARNING: Changing the LPRDS
Hardware Description XML file by any means other than the supplied addDevice and removeDevice methods is not recommended and may result in corruption of the device lists and/or the hardware description file!

- **getPollingInterval()**
  
  *Return Type:* int
  
  - The returned value is the current system polling interval in seconds.

  *Parameters:*
  
  - none

  *Description:*
  
  - The method requests the current system polling interval from the LPRDS Kernel process. The polling interval is how often the system polls each sensor for a reading.

- **setPollingInterval(seconds)**
  
  *Return Type:* int
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  *Parameters:*
  
  - **seconds:** An int value that is the number of seconds the polling interval will be changed to.

  *Description:*
  
  - This method instructs the LPRDS Kernel process to change how often each sensor in the LPRDS system is polled and how often sensor readings are saved in the LPRDS database. An error code is returned to the application by the LPRDS Kernel process.

- **getState()**
  
  *Return Type:* int
  
  - The returned value is an integer representing the current operational system state.

  *Parameters:*
  
  - none

  *Description:*
  
  - The method requests the current operational system state from the LPRDS Kernel process. The system state is managed by the LPRDS Operational State Manager application. The LPRDS operational states and corresponding integers are explained in Appendix B.

- **setState(state)**
  
  *Return Type:* int
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.
Parameters:
- state: An int value representing the next operational state of the LPRDS system.

Description:
- The method is used to change the current operational state of the LPRDS system in the LPRDS Kernel process. The LPRDS operational states and corresponding integers are explained in Appendix B.

- **checkSafety()**
  
  Return Type: int
  
  - The returned value is 1 or 0, representing closed or open, respectively.

  Parameters:
  - none

  Description:
  - This method can be used by an application to check the status of the safety loop. The status of the loop is returned as an integer value, a 1 for closed or a 0 for open.

- **operateSafety(level)**
  
  Return Type: int
  
  - The returned value is an error code; see LPRDS Error Codes in Appendix A.

  Parameters:
  - level: An integer 1 or 0 representing closed or open, respectively.

  Description:
  - This method can be used by an application to operate the SCADA safety loop relay. The input parameter, level, as an integer 1 or 0 determines what action is performed. If level = 0 the safety loop relay is opened. If it is a 1 the safety loop relay is closed.

- **killApps()**
  
  Return Type:
  
  Parameters:
  - none

  Description:
  - This method communicates with the LPRDS Kernel process via the kernel communication pipe. The purpose of the method is to provide a means of terminating all applications registered with the LPRDS Kernel process through the connect method. The only application permitted to use this method is the LPRDS Operational State Manager application. The only applications that are not terminated by use of this method are the LPRDS
Kernel process and the LPRDS Operational State manager. The applications are terminated by the SIGURG signal, which should be handled by the application.

- **getErrorMsg(int err)**
  
  **Return Type:** string
  
  - The error message corresponding to the specified error code.

  **Parameters:**
  
  - **err:** The error code

  **Description:**
  
  - This method is provided as a means of standardizing error codes and error messages. The error codes handled are described in Appendix A.

- **getFails()**

  **Return Type:** vector<string>
  
  - The LPRDS system devices that have experienced an LV fault.

  **Parameters:**
  
  - none

  **Description:**
  
  - The method communicates with the LPRDS Kernel process via the kernel communication pipe in order to get a list of the devices that have experienced an LV fault. An LV fault consists of a sensor reading that is out of the bounds specified in the XML Hardware Description file for that particular device.

- **intToString(int c)**

  **Return Type:** string
  
  - The specified integer value converted to a C++ string.

  **Parameters:**
  
  - **c:** The integer to be converted

  **Description:**
  
  - This method converts the specified integer to the string type.

- **stringToInt(string s)**

  **Return Type:** int
  
  - The specified string converted to an integer value.

  **Parameters:**
  
  - **s:** The string to be converted

  **Description:**
  
  - This method converts the specified string to an integer value, if possible.
LPRDS Error Codes

These error codes may be returned if the method return type is a float or integer type. If the return type is of type float, the error codes possible are found in the Error Code (float) column. If the return type is of type int, the error codes possible are found in the Error Code (int) column. If the return type of a method is not of type int or float, there are different possibilities for errors. If the return type is a string, the error will be described in the string. If the return type is an STL container, such as a List or Vector, the container will be empty if there was an error.

<table>
<thead>
<tr>
<th>Error Code (int)</th>
<th>Error Code (float)</th>
<th>Error Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>-99991</td>
<td>Device does not exist</td>
</tr>
<tr>
<td>2</td>
<td>-99992</td>
<td>Wrong device type</td>
</tr>
<tr>
<td>3</td>
<td>-99993</td>
<td>DAQ communication timeout</td>
</tr>
<tr>
<td>4</td>
<td>-99994</td>
<td>DAQ board communication error</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Pin no. is out of bounds</td>
</tr>
<tr>
<td>6</td>
<td>-99996</td>
<td>A-D channel no. is out of bounds</td>
</tr>
<tr>
<td>7</td>
<td>-99997</td>
<td>Sensor reading exceeds maximum value</td>
</tr>
<tr>
<td>8</td>
<td>-99998</td>
<td>Sensor reading below minimum value</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>Unable to access LPRDS Hardware Description XML file</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Device ID already exists</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>LPRDS Kernel is already running</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>LPRDS Kernel is already shutdown</td>
</tr>
<tr>
<td>13</td>
<td>-99913</td>
<td>Unable to perform action, LPRDS Kernel is shutdown</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>SQL command conversion error</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>SQL query error</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>Unknown SQL error</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>SQL invalid name</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>SQL invalid DAQ/pin combination</td>
</tr>
<tr>
<td>19</td>
<td>-99919</td>
<td>Write to pipe error</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>Kernel &amp; Results pipe do not exists, LPRDS Kernel may not be running</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>Kernel pipe does not exists, LPRDS Kernel may not be running</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>Results pipe does not exists, LPRDS Kernel may not be running</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>Requested operation is not permitted</td>
</tr>
</tbody>
</table>
Unable to connect to LPRDS Kernel
SIB is not connected
SIB communication error
-99927 API Timeout error

SCADA Block Diagram Description

**LPRDS Core System Software**

The core system software refers to all the software to the left of the Applications Programming Interface (API) line. This includes the XML file, the XML Parser, XCR, I/O Manager, Sunny Boy, Kernel, Database, MySQL++ and the MySQL database. This software is responsible for interacting with the hardware, which includes the SCADA Interface Box (SIB), the Data Acquisition boards, the Sunny Boy Inverter, and relays to interact with the safety loop.

**Hardware Desc. XML**

This block in the LPRDS Software Architecture block diagram represents the LPRDS Hardware Description XML file. This file, written using the XML format, is used to store all important parameters for each hardware device in the LPRDS BMS. The file can be modified manually to add, remove, and edit hardware devices in the system, however, this is highly discouraged and, if
performed improperly, this may result in system failure. It is recommended that any hardware changes be performed via the Maintenance application, represented in this diagram by the “Maint.” block.

**XML Parser**

The purpose of this component of the LPRDS Software system is to interpret the Hardware Description XML file. The XML Parser scans the file for the LPRDS hardware devices and sends the data to the LPRDS Kernel process, represented here by the “Kernel” block. Additionally, the XML Parser performs additions and removals in the Hardware Description XML file as requested from the Maintenance application via the Kernel.

**Kernel**

The LPRDS Kernel process is the core component of this software system. The kernel acts as the communication hub for all parts of the system, as well as performing safety monitoring that is integral to proper operation of the LPRDS BMS. The C++ API, available for application development, communicates exclusively with the kernel to perform critical system tasks. Additionally, the Kernel handles all data processing and storage by sending the sensor readings to the MySQL database using the MySQL++ wrapper package.

**Sunny Boy (blue)**

This block in the LPRDS Software Architecture represents the C++ code that handles communication with the Sunny Boy Grid-tie Inverter, which is a commercially purchased inverter that is used when the LPRDS BMS is not in use. The code converts data requests from the Kernel process into a serial data stream that can then be sent to the SIB through a USB port where it is then relayed to the actual Sunny Boy inverter via RS-485 communication protocol.

**I/O Manager**

The I/O Manager is a C++ module that processes requests from the Kernel to read a sensor or set a switch/relay in the LRPDS system. The requests are converted into data packets that are then transferred to the XCR module for transmission to the data acquisition (DAQ) boards. The packet structure used is the same as was used by the 2009 LPRDS team since the software on the data acquisition boards was reused. In addition to generating packets, the I/O Manager also interprets packets received from the data acquisition hardware. The packets are received by the XCR module and then the important data is extracted and relayed to the LPRDS Kernel for processing.

**XCR**

The XCR module handles USB communication to and from the data acquisition hardware. The packets to be sent are generated by the I/O Manager and then sent to XCR for transmission. XCR then transmits the packets through USB to the SIB, where they are then sent via RS-485 to the appropriate location. Data is received from the USB port and converted into packet form where it is then sent to the I/O Manager for further processing.
SIB
The SCADA Interface Box (SIB) is a commercial product which connects to the USB port of the computer. It has relays which are used to trip the safety loop as well as check if the safety loop has been tripped by another subsystem. It also has two serial ports which allow us to communicate (using the RS-485 protocol) to the Data Acquisition boards.

Database
This block represents the C++ software interface that was designed to handle all communication between the LPRDS Kernel and the MySQL database. The software used the MySQL++ software package in order to integrate with the database.

MySQL++
MySQL++ is a commercial software package for Linux that was installed in order to be able to use C++ to operate the LPRDS MySQL database.

MySQL
The Database block represents the MySQL database used by the LPRDS software to store all data and log information from the system. The information in the database is viewable from the LPRDS SCADA website, which is represented by the Website block.

Apache2
In order to be able to display the SCADA website, the LPRDS software system incorporates the Apache2 web server, a commercial package for Linux.

State Mgmt.
The State Mgmt. block in the LPRDS Software Architecture represents the LPRDS Operational State Manager application. The application uses the C++ API to manage the LPRDS system according to operational states specified in the final report.

Batt. Mgmt.
The Battery Management application is the implementation of the battery charge and discharge algorithm. This application can only be run when the LPRDS system is in the Operational state, as specified by the LPRDS Operational State Manager. Should a user attempt to start the application when the LPRDS system is not in the correct state, the application will display an error to the user and terminate. If the application is running properly and the system state changes, the LPRDS Kernel will terminate the application using Linux signals.

Maint.
The Maintenance application is a piece of software written in C++ that was designed to be used by an experienced LPRDS operator to manually control the system. The program implements the LPRDS C++ API and is capable of performing switch operations, reading sensors, adding/removing devices from the hardware description XML file, and manually setting the system state, among other things.
**Demo**

This block represents the slideshow application that displays some current system values as well some educational information about the LPRDS system on an LCD display. The slideshow is intended to be viewed by passersby as a brief description and demonstration of the system, in addition to the LPRDS poster.

**Website**

The purpose of the SCADA website is to provide an interface to view the information stored in the LPRDS MySQL database. The website consists of four pages, each with a specific purpose. The main page contains a brief synopsis of the SCADA website and a link to the project development website. The system page shows the most recent system state along with the status of a few critical system components. The logs page allows the user to view event, fault and system state logs within a selected date range. The date range is chosen by using the drop down menus to select the range to within hour precision. The data page allows the user to view the stored data for every device that has been a part of the LPRDS system. Additionally, the website allows the user to plot the data as well. Like on the log page, the data is displayed or plotted from the user specified date range within hourly precision. On both the data and log pages, the default date range is the current day. For security purposes, the website is accessible only from the 4th floor of the Acopian Engineering Center at Lafayette College.

**Display**

The Display block is a standard LCD display that is used by the Demo application.

**Misc.**

The Misc. block represents the safety relays used by the SCADA system.

**Data Acq.**

This block represents the data acquisition hardware used in the system by the ESS, FIB, RPI, and SC. Communication to these hardware components is directed by the SIB and handled by the XCR and I/O Manager modules.

**Sunny Boy (pink)**

The Sunny Boy is a commercial Grid-Tie Inverter that is used when the LRPDS system is shutdown.

**C++ API**

The LPRDS C++ API was designed to enable applications programmed in C++ to fully control the LPRDS BMS. The C++ API separates the core system processes from the system management applications. The LPRDS Kernel process handles all commands that are processed via the C++ API and executes the appropriate actions, such as reading a system sensor or setting a switch, relay, or digital output.
SQL API

The SQL API represents the extensive PHP-SQL communication interface. The full set of commands, however, is limited to query only permission. This was done to prevent accidental or malicious actions that could delete data or change database structure. The only way to change or add to the database is from behind the C++ API and is handled automatically by the LPRDS Kernel process via the MySQL-C++ wrapper.

Pico-LCD

Viewing the configuration file and the contents:

The file that determines the layout for the pico-lcd is located in /usr/local/lprds/etc/, the file is named lcd4linux-lprds.conf. By modifying this file, the displayed elements of the pico-lcd screen can be changed. In the beginning of the file there are various widgets that are currently being displayed on the pico-lcd screen. There are additional widgets available to add and a list of available widget is located at http://ssl.bulix.org/projects/lcd4linux/wiki/Layout along with their respective fields and examples. Near the bottom of the screen is the layout which describes where the widgets will be displayed on the pico-lcd screen. Here is an example screen shot of how the widgets are put into the file:

![LCD Configuration File Example](image)

The exec expression executes a terminal command and prints the result on the lcd screen at a refresh rate specified by the second argument.
**Starting and Stopping the configuration file:**

The configuration file mentioned above is currently set to be run on startup so when the fit pc is
booted up the layout described in the file will be displayed. The startup script, lcd4linux, located
in the /etc/init.d directory on the fit pc is responsible for calling the correct commands to
configure the layout. It has a start and stop command which can be called manually from the
terminal by sudo /etc/init.d/lcd4linux start and  sudo /etc/init.d/lcd4linux stop. These are useful if
the configuration file has been changed and the changes are desired to be displayed immediately.

**Database**

The LPRDS software has the ability to automatically log system information to a MySQL
database. This was accomplished by creating a MySQL database locally on the fitPC and using
MySQL++, which is a commercial package that allows communication from C++ to MySQL.
The current database can be viewed on the fitPC with a graphical user interface within firefox. A
‘database’ link is available in the bookmark bar, or you can type in the following address:

```
http://localhost/phpmyadmin/
```

Username: root                      Database: SCADAdata
Password: 111111                      Server = localhost

The database is arranged into subsections known as tables with each entry in the table
known as a row. The SCADAdata database is arranged into 9 tables, each with an auto-
incrementing index named ‘id’. Tables such as event_log, fault_log, state_log and data have
additional fields ending in ‘_id’ (event_id, hw_id etc.) that correspond to the ‘id’ number of the
table named in the prefix. For example, the ‘hw’ table stores detailed information about each
sensor and assigns each a unique index ‘id’ number. When a sensor reading is stored in the ‘data’
table, it only needs to store the matching ‘id’ number from the ‘hw’ table in the ‘hw_id’ field to
identify the sensor. This was done to minimize the amount of space used by the database.
To run the demo app navigate to ‘scada/demoApp’ within terminal and type ‘./demoApp’.

The Demo App was designed to provide a passerby with information about LPRDS and demonstrate system capabilities. It is to be viewed on the LCD display located on top of the LPRDS tower and is currently programmed to automatically cycle through pages of information, similar to a slide show. It can also be manually navigated using the ‘+’ and ‘-’ keys on the numpad and was intended to eventually be controlled by the touch sensor system described in the following section. The demo app was programmed in C++ using QT4 for the user interface and LPRDS API functions to read from the database. Graphics were created using Adobe Illustrator.

Files for the demo App can be found within a subfolder of the ‘scada’ software folder. The ‘demo.ui’ file defines the user interface and can be edited graphically using QT4 Designer. Modifications beyond the interface appearance should be made within the ‘demoGUI.cpp’ file.

The pages of information and their ordering are described in the page transition diagram shown below. The app is designed to display different information depending on which system is running- LPRDS or Sunny Boy. Hitting the ‘L’ key while running the demo app toggles the ‘LPRDS’ variable and changes the screens that are shown. Future expansion of the demo app should use API methods to automatically determine if LPRDS or the Sunny Boy is running.
Figure 45: Demo App Page Transition Diagram
**Touch Sensor**

Preliminary work was started on a touch sensor system to be used with the demo app. The goal was to allow simple navigation (left or right) from outside of the room. Capacitive touch technology is an ideal solution because it allows user input through the glass while keeping all hardware within room 401, making it tamper proof. Also, it's just cool.

The product being used is the QT113 Charge-Transfer Touch Sensor (Digikey part# 427-1138-1-ND). They are designed to make touch sensing simple and automatically calibrate to ensure proper operation. These chips were smaller than expected so two of them have been soldered to a standard chip holder for use with a bread board. The chips require an external capacitor and metal electrode to operate and run on 2.5 to 5 V. Its output is active low with an occasional heartbeat pulse that can be view on an oscilloscope. Actual implementations of the chip may require the output to be inverted and filtered to remove the heartbeat pulse. Refer to the spec sheet for more details.

For the computer to respond to input from the touch sensor, we had planned on emulating a key press and sending the signal via usb to the fitPC. This can be done by either using a PIC processor with custom software or using a product such as the I-Pac PC interface board [link](http://www.ultimarc.com/ipac1.html). The demo app is currently set up to move forward or step back upon pressing the ‘+’ or ‘-‘ keys on the numpad.

**Demo Outlet**

Within the Output box is a 30A normally open AC relay connected to a standard wall outlet. When completed, this will allow software control over an AC load to demonstrate that the system is generating a usable AC signal. The outlet has been tested with high voltage AC and works with loads up to 960W. When the relay is switched on, it draws 143 mA from the 5V supply.

The control signal was left unfinished due to time constraints but can be provided by the SIB using one of the spare relay ports. The 5V power and 5V control signal must share a common ground for the relay to operate properly. The wires for demo outlet control are coded as follows:

- Red - 5 Vdc power
- Blue – 5 Vdc control signal
- Black - ground
Once completed the demo outlet can be turned on for a brief moment during the progression of the demo App to demonstrate a working AC signal from the FIB. Ideas for electronics to plug into the demo outlet include light bulbs, heat lamps or Christmas lights during the holiday season.
System Safety

Safety Loop

As shown in Figure 42, the main logic for the safety interface exists in the RPI, but each subsystem is part of the safety loop. The ground fault monitor in the RPI operates by monitoring the current on the high voltage lines and detecting a ground fault if the current entering and leaving do not match. It opens the safety loop if a ground fault is detected.

Temperature sensors in each subsystem open automatically when overheated, breaking the safety loop. Also in each subsystem are controllable relays that can be opened via SW commands, allowing SCADA to break the safety loop if it detects unfavorable conditions internal to a subsystem.

If there are no faults or failures and the safety loop is closed, then the “Safety 12V” created in the RPI is provided to all subsystems. Isolation relays are used wherever high voltage is present (except in the SC, which contains two switches that can also be used to isolate high voltage). These high-voltage isolation relays are normally open, and require the 12V to stay closed. Therefore, if a fault occurs, the safety loop opens, stopping delivery of 12V to the high-voltage isolation relays and breaking all high voltage connections.

On the RPI are two buttons, one green and one red. The red button is a shutdown button that provides a way to open the safety loop manually in case a fault is suspected, or in case the system fails to automatically detect a fault. The green button is a safety reset button, which
clears any faults and allows the safety logic to operate. If a fault is detected, the safety loop will be open and “Safety 12V” will not be delivered, but the safety loop logic is still operating—i.e., once the fault is cleared the system will continue to operate. If the safety loop is broken manually with the red shutdown button, however, it must also be cleared manually with the green safety reset button for the system to continue operating.

**SCADA Interface Box (SIB) Hardware**


Manual (USB-IIRO4-2SM.PDF) and Datasheet (USB-IIRO4-2SM_DATASHEET.PDF) are located in SIB Documentation folder.

**Features:**

- 4 isolated digital inputs (3-31V is logic high)
- 4 SPDT relays
- 2 serial ports configured to RS-485
- USB connected & powered

We are using both serial ports, one relay, and one isolated input.

**Cables:**

**WPC1**

This USB power cable is coming from the FIT PC.

**WFS1**

This is a modified Cat5e Ethernet cable from the FIB. The connector on the SIB side has been modified to be a standard 9 pin connector. The white-brown wire (of the cable) is wired to pin 2 and the brown wire is wired to pin 3.

**WSS1**

This is a modified 4 conductor wire from the Sunny Boy inverter. The connector on the SIB side is a standard 9 pin connector with the A wire wired to pin 2 and the B wire wired to pin 3.

**WSS2**

This is a 4 conductor wire from the Safety to Software Interface Board. The connector on the SIB side is a standard 25 pin connector. This wire contains the safety loop and safety 12. The safety loop is wired to pins 8 and 20 and the safety 12 is wired to pins 1 (+12V) and 14 (Ground). Pin 8 is the common for relay 0 on the SIB and pin 20 is the normally open contact of relay 0. Pin 1 is the positive terminal for the digital input channel 0 on the SIB and pin 14 is the negative terminal for channel 0.
Three objects live on the right side of the tower.

**Emergency Shutdown Button**

This is a big mushroom button that is used as an emergency stop button. It is connected directly into the safety loop. It has two positions: pulled out (input is connected to output) and pushed in (input is not connected to output). To manually break the safety loop push this button.

**Alarm**

Alarm is an AudioLarm II from Floyd Bell Inc. (Part Number: XC-09-212-QR)

Website: [http://www.floydbell.com/products/specifications/XC-09-212-QR](http://www.floydbell.com/products/specifications/XC-09-212-QR)

**Alarm Acknowledge Button**

A button used to acknowledge that the alarm is going off.


Datasheet is in SIB Documentation folder (287-1029.PDF)
Safety to Software Interface Board
This board is designed to be the interface between the tower hardware, SIB, and the safety loop.

Board Operation:
- On power-up the alarm will beep for approximately 1/3 of a second. This is both to indicate that the alarm is working and that the safety loop is broken.
- Once the safety loop is closed the board will wait for it to be broken.
- Once the safety loop is broken the alarm will beep until either the safety loop is closed or the alarm acknowledge button is pushed.
- Once the alarm is silenced it will not beep until the safety loop is closed and then broken again.

The board schematic & layout are included below.

Cables
Note: All cables must match their label to the label on the alarm board. The labels go as close together as possible.

WSS2
This is a 4 conductor wire from the SIB. The connector on the alarm board side is a 2 two pin connectors. This wire contains the safety loop and safety 12. The safety loop is wired to connector X and the safety 12 is wired to connector Z.

WCS1
Safety Loop wire from the SC is connected into WCS2. Connector J58A connects to SA.

WCS2
Safety Loop wire adapter for the alarm board. This connects the normal safety loop connector (SA) to the four pin connector (S) on the board.

Connections from Tower Hardware

Big Red Button
Two wire connector labeled E on the alarm board.

Alarm
Two wire connector labeled A on the alarm board.

Alarm Acknowledge Button
Two wire connector labeled B on the alarm board.

WEF1B
One branch of a Y cable, this is a 12 volt power supply from the ESS.
Here is the layout of the alarm board this is a Visio file in the SIB Documentation folder (AlarmBoard.vsd).

**Notes on layout:**
- All connections are labeled with their respective letters.
- All solder paths are color coded based on their type.
- The Visio document is rendered so one grid space represents one hole on the proto-board.
- The dotted lines are solder paths that cross under the GAL chip.
- A wire acts as a jumper to the left of the GAL chip.

**6N135**
This is an optoisolator that separates Safety 12 from the GAL. The top two pins of S are Safety 12 and Safety 12 ground. Safety 12 runs through a resistor then through the 6N135. The connections on the other side act as a switch. When the Safety 12 is present the out signal is high.

**MCP130**
This is a supervisory circuit that holds the output signal low for 350 milliseconds. The flat part of this chip goes to the left in this diagram.

**LM317**
This is a voltage regulator that takes System 12V and turns it into 5V for all of the chips on the board.

Note: The GAL program is detailed in the SIB Software Word document in the SIB Documentation folder (SIB_Software.docx).

SCADA Interface Box (SIB) Software

SIB Firmware
The SIB utilizes the supplied firmware. The instruction document was followed with no deviations. FXLOAD must be installed on a Linux machine in order to use this firmware.

NOTE: I ran into trouble with this on the Linux desktop machine (not the FIT) I was writing software on. After many successful firmware downloads it stopped automatically downloading the firmware to the SIB. I do not know why this occurred but it has not been reproduced in the FIT as of yet.

SIB Software

SIB Class
There is a custom SIB class in the SCADA software files (sib.c & sib.h). The SIB class consists of a constructor, deconstructor, relay method, and input method.

Constructor
The constructor performs the necessary operations to be able to communicate with the SIB. This code was taken from the provided sample program. It also configures the SIB’s channels to act as relays or digital inputs depending on their actual function.

Relay Method
The relay method will open and close one of the four relays (labeled 0-3) using a given value (0 for close and 1 for open).

Input Method
The input method will return a logic value for the given input channel. A 1 corresponds to greater than 4V (up to 30V) and a 0 corresponds to 0-4V. The method will return a -1 if the operation failed.

AIOUSB Libraries
The libraries that the SIB class use are found in the libraries folder of the SCADA software code. There are four total; two are C libraries and two are C++ libraries.
NOTE: The libraries in the compressed tar file must be recompiled on a 32 bit machine because they were originally compiled on a 64 bit machine. You will get errors if you do not use the already compiled ones in the SCADA software code file or freshly recompiled ones.

Alarm Board GAL Program
The GAL on the alarm board is running the logic for the safety hardware on the tower (the alarm and two buttons). It has five inputs, three outputs, and a few logic expressions.

Inputs
Clock
The clock is run off of the clock_out output signal.
R
An asynchronous reset signal that is tied directly to the safety loop indication.
Safety
This is the output from an optoisolator that indicates when the safety loop is closed.
Button
An input that drops whenever the Alarm Acknowledge Button is pressed.
Poweron
An input that will stay low for 350 milliseconds after power is supplied to the board.

Outputs
Alarm
The output that powers the alarm.
Ack
An unused output indicating that the alarm has been acknowledged.
Clock_out
The clock signal that will set the Ack signal.

Logic Functions
The Ack signal is asynchronously reset to zero by the inverse of the safety signal. On the rising edge of the clock Ack is set to one.

Alarm is equal to one if the safety loop is broken (safety is one) and Ack is not one.

Clock_out is the not of the exclusive or of Button and Poweron. This will give a rising edge as Poweron rises to one or if the button is pressed.
This section of the final report consists of the documentation of some of the general project requirements associated with the LPRDS system. Included in this section are the budget, power budget, and the “ilities” reports. The ilities reports included within this section are environmental, EMI, hazmat, safety, reliability, maintainability, manufacturability, sustainability, and ethics.
## Budget

<table>
<thead>
<tr>
<th>Order No.</th>
<th>Vendor</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>951948</td>
<td>Mouser Electronics</td>
<td>$149.73</td>
</tr>
<tr>
<td>951949</td>
<td>Jameco</td>
<td>$126.48</td>
</tr>
<tr>
<td>951951</td>
<td>Digi-Key</td>
<td>$167.30</td>
</tr>
<tr>
<td>953521</td>
<td>McMaster-Carr</td>
<td>$58.93</td>
</tr>
<tr>
<td>953524</td>
<td>Newark Electronics</td>
<td>$15.98</td>
</tr>
<tr>
<td>953525</td>
<td>Access I/O Products</td>
<td>$211.64</td>
</tr>
<tr>
<td>953527</td>
<td>Mouser Electronics</td>
<td>$31.53</td>
</tr>
<tr>
<td>953538</td>
<td>Marlin P. Jones</td>
<td>$24.51</td>
</tr>
<tr>
<td>953529</td>
<td>Jameco</td>
<td>$133.70</td>
</tr>
<tr>
<td>953530</td>
<td>Digi-Key</td>
<td>$430.63</td>
</tr>
<tr>
<td>953531</td>
<td>Mouser Electronics</td>
<td>$120.17</td>
</tr>
<tr>
<td>953532</td>
<td>Allied Electronics</td>
<td>$180.54</td>
</tr>
<tr>
<td>953533</td>
<td>Digi-Key</td>
<td>$90.62</td>
</tr>
<tr>
<td>953534</td>
<td>Allied Electronics</td>
<td>$14.55</td>
</tr>
<tr>
<td>953535</td>
<td>Newark Electronics</td>
<td>$8.11</td>
</tr>
<tr>
<td>953537</td>
<td>Digi-Key</td>
<td>$40.51</td>
</tr>
<tr>
<td>953543</td>
<td>Allied Electronics</td>
<td>$151.35</td>
</tr>
<tr>
<td>953542</td>
<td>Marlin P. Jones</td>
<td>$60.17</td>
</tr>
<tr>
<td>151400</td>
<td>Stokes</td>
<td>$23.32</td>
</tr>
<tr>
<td>3012339</td>
<td>Lodestone Pacific</td>
<td>$145.32</td>
</tr>
<tr>
<td>953548</td>
<td>Mouser Electronics</td>
<td>$72.92</td>
</tr>
<tr>
<td>953557</td>
<td>Albarell Electric Inc.</td>
<td>$80.00</td>
</tr>
<tr>
<td>644128</td>
<td>Advanced Circuits</td>
<td>$156.78</td>
</tr>
<tr>
<td>953547</td>
<td>Marlin P. Jones</td>
<td>$75.43</td>
</tr>
<tr>
<td>953550</td>
<td>Allied Electronics</td>
<td>$39.62</td>
</tr>
<tr>
<td>953555</td>
<td>Digi-Key</td>
<td>$35.87</td>
</tr>
<tr>
<td>953556</td>
<td>McMaster-Carr</td>
<td>$24.41</td>
</tr>
<tr>
<td>954141</td>
<td>Digi-Key</td>
<td>$132.73</td>
</tr>
</tbody>
</table>
As seen above, we were over budget by about 4% or $115.12, which was expected since we had many unexpected purchases at the last minute of the project. To see a more detailed list of parts see the Bill of Material List (BOM).
## Power Budget

Power Available = 37.5W

<table>
<thead>
<tr>
<th>System</th>
<th>Component</th>
<th>Watts</th>
<th># of components</th>
<th>Total Watts</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPI</td>
<td>Ground Monitoring</td>
<td>0.18</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>RPI</td>
<td>LEDs @ 12V</td>
<td>0.09</td>
<td>2</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>Relay</td>
<td>0.5291</td>
<td>1</td>
<td>0.5292</td>
</tr>
<tr>
<td>RPI</td>
<td>Total</td>
<td></td>
<td></td>
<td>3.8892</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>Current Sensor</td>
<td>0.18</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>Microchipthermistors</td>
<td>0.000375</td>
<td>1</td>
<td>0.000375</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>RS485 Isolator</td>
<td>0.225</td>
<td>1</td>
<td>0.225</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>PIC</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>Driver</td>
<td>0.125</td>
<td>1</td>
<td>0.125</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>Comparator</td>
<td>0.125</td>
<td>2</td>
<td>0.25</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>SM Components</td>
<td>0.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>Relay</td>
<td>0.05</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>RPI DAQ</td>
<td>Total</td>
<td></td>
<td></td>
<td>1.530375</td>
</tr>
<tr>
<td>ESS</td>
<td>LEDs @ 12V</td>
<td>0.18</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>ESS</td>
<td>LEDs @ 5V</td>
<td>0.09</td>
<td>2</td>
<td>0.18</td>
</tr>
<tr>
<td>ESS</td>
<td>Relay</td>
<td>0.5291</td>
<td>2</td>
<td>1.0582</td>
</tr>
<tr>
<td>ESS</td>
<td>Total</td>
<td></td>
<td></td>
<td>1.4182</td>
</tr>
<tr>
<td>SC</td>
<td>voltage meters</td>
<td>0.2</td>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>SC</td>
<td>temp meters</td>
<td>0.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>SC</td>
<td>display LEDs</td>
<td>0.09</td>
<td>2</td>
<td>0.18</td>
</tr>
<tr>
<td>SC</td>
<td>Relays</td>
<td>0.06</td>
<td>2</td>
<td>0.12</td>
</tr>
<tr>
<td>SC</td>
<td>Total</td>
<td></td>
<td></td>
<td>0.9</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>Current Sensor</td>
<td>0.18</td>
<td>1</td>
<td>0.18</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>Microchipthermistors</td>
<td>0.000375</td>
<td>4</td>
<td>0.0015</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>RS485 Isolator</td>
<td>0.225</td>
<td>1</td>
<td>0.225</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>PIC</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>Driver</td>
<td>0.125</td>
<td>1</td>
<td>0.125</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>SM Components</td>
<td>0.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>DAQ Boards</td>
<td>Total</td>
<td></td>
<td></td>
<td>1.2315</td>
</tr>
<tr>
<td>ESS &amp; SC</td>
<td>Total</td>
<td></td>
<td></td>
<td>2.463</td>
</tr>
</tbody>
</table>
FIB

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Power</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference diode</td>
<td>0.075</td>
<td>1</td>
<td>0.075</td>
</tr>
<tr>
<td>Optocoupler</td>
<td>0.015</td>
<td>2</td>
<td>0.03</td>
</tr>
<tr>
<td>Microcontroller</td>
<td>0.095</td>
<td>1</td>
<td>0.095</td>
</tr>
<tr>
<td>LEDs @ 12V</td>
<td>0.18</td>
<td>2</td>
<td>0.36</td>
</tr>
<tr>
<td>LEDs @ 5V</td>
<td>0.09</td>
<td>4</td>
<td>0.36</td>
</tr>
<tr>
<td>Drivers</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Transistor</td>
<td>0.5</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>Relay</td>
<td>0.53</td>
<td>1</td>
<td>0.53</td>
</tr>
<tr>
<td>RS485</td>
<td>0.275</td>
<td>1</td>
<td>0.275</td>
</tr>
<tr>
<td>Core</td>
<td>0.8</td>
<td>2</td>
<td>1.6</td>
</tr>
<tr>
<td>temp meter</td>
<td>0.2</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>display LEDs</td>
<td>0.09</td>
<td>2</td>
<td>0.18</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>6.205</strong></td>
</tr>
</tbody>
</table>

SCADA

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Power</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fit PC</td>
<td>6</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>SIB</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>7</strong></td>
</tr>
</tbody>
</table>

Safety Interface

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Power</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarm board</td>
<td>0.3</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>DC Soldi State Relays</td>
<td>0.336</td>
<td>2</td>
<td>0.672</td>
</tr>
<tr>
<td>Relay</td>
<td>0.5292</td>
<td>1</td>
<td>0.5292</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>1.5012</strong></td>
</tr>
</tbody>
</table>

Tower Display

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Power</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEDs</td>
<td>0.0125</td>
<td>7</td>
<td>0.0875</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td><strong>0.0875</strong></td>
</tr>
</tbody>
</table>

System Total

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Power</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>24.994475</strong></td>
</tr>
</tbody>
</table>

Available Power

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Power</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>12.505525</strong></td>
</tr>
</tbody>
</table>
**GPR002 – Environment**

An updated environmental analysis was not performed for the LPRDS-BMS2010 project; however the analysis from the 2009 team was understood and applied to the 2010 design. The 2009 environmental analysis is included in the Appendix.

**GPR003 – Electromagnetic Interference Compatibility**

An updated Electromagnetic Interference Compatibility analysis was not performed for the LPRDS-BMS2010 project; however the analysis from the 2009 team was understood and applied to the 2010 design. The 2009 Electromagnetic Interference Compatibility analysis is included in the Appendix.

**GPR004 – Hazardous Materials**

The Lafayette College Chemical Hygiene Plan lists carcinogens, reproductive toxins, and corrosive substances as hazardous materials. A list of these chemicals from the hygiene plan is included in . A Material Safety Data Sheet (MSDS) will be provided with the safety plan for all parts requiring one. The LiFePO$_4$ batteries do not require an MSDS. During the course of this project we did not explicitly purchase any chemicals listed in the Chemical Hygiene plan. Also, none of the parts we ordered had warnings or otherwise indicated that they contained these substances.

All parts in the design meet 2002/95/EC RoHS requirements, as only parts marked as RoHS compliant were selected for the design. The system includes only LiFePO$_4$ batteries (no NiCd or Lead-acid batteries are used).
<table>
<thead>
<tr>
<th>Carcinogens</th>
<th>Reproductive Toxins</th>
<th>Corrosive Substances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alkylating agents: -halo ethers</td>
<td>Alkylating agents: aziridines</td>
<td>Strong acids</td>
</tr>
<tr>
<td>- bis(chloromethyl) ether</td>
<td>- ethylenimine</td>
<td>- sulfuric</td>
</tr>
<tr>
<td>- methyl chloromethyl ether</td>
<td>2-methylaziridine</td>
<td>- nitric</td>
</tr>
<tr>
<td>Alkylating agents: epoxides</td>
<td>Hydrazines</td>
<td>- hydrochloric</td>
</tr>
<tr>
<td>- ethylene oxide</td>
<td>- hydrazine (and hydrazine salts)</td>
<td>- hydrofluoric</td>
</tr>
<tr>
<td>- Diepoxybutane</td>
<td>- 1,2-diethylhydrazine</td>
<td></td>
</tr>
<tr>
<td>- Epichlorohydrin</td>
<td>- 1,1-dimethylhydrazine</td>
<td></td>
</tr>
<tr>
<td>- Propylene oxide</td>
<td>1,2-dimethylhydrazine</td>
<td></td>
</tr>
<tr>
<td>- Styrene oxide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alkylating agents: diazo, azo, and azoxy compounds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- 4- dimethylaminoazobenzene</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acylating agents</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- propiolactone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- butyrolactone</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-nitroso compounds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- N-nitrosodimethylamine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- N-nitroso-N-alkylureas</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alkylating agents: sulfonates</td>
<td>Dimethylcarbamoyl chloride</td>
<td>Mercury compounds</td>
</tr>
<tr>
<td>- 1,4-butanediol dimethanesulfonate</td>
<td>Natural products (including antitumor drugs)</td>
<td></td>
</tr>
<tr>
<td>- Diethyl sulfate</td>
<td>- adriamycin</td>
<td>Nitrobenzene</td>
</tr>
<tr>
<td>- Dimethyl sulfate</td>
<td>- aflatoxins</td>
<td>Nitrous Oxide</td>
</tr>
<tr>
<td>- Ethyl methanesulfonate</td>
<td>- bleomycin</td>
<td>Phenol</td>
</tr>
<tr>
<td>- Methyl methanesulfonate</td>
<td>- cisplatin</td>
<td>Polychlorinated biphenyls</td>
</tr>
<tr>
<td>- Methyl trifluoromethanesulfonate</td>
<td>- progesterone</td>
<td>Polybrominated biphenyls</td>
</tr>
<tr>
<td>- 1,3-propanesultone</td>
<td>- reserpine</td>
<td>Toluene</td>
</tr>
<tr>
<td></td>
<td>- safrole</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>strong bases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sodium hydroxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- potassium hydroxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- ammonium hydroxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>dehydrating agents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sulfuric acid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sodium hydroxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- phosphorus pentoxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- calcium oxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>oxidizing agents</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- hydrogen peroxide</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- chlorine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- bromine</td>
</tr>
</tbody>
</table>

*Table 1: Hazardous Materials*
<table>
<thead>
<tr>
<th>Misc. Organic compounds</th>
<th>Organohalogen compounds</th>
<th>Vinyl Chloride</th>
</tr>
</thead>
<tbody>
<tr>
<td>- formaldehyde</td>
<td>- 1,2-dibromo-3-chloropropane</td>
<td>Xylene</td>
</tr>
<tr>
<td>- acetaldehyde</td>
<td>- Mustard gas</td>
<td></td>
</tr>
<tr>
<td>- 1,4-dioxane</td>
<td>- Carbon tetrachloride</td>
<td></td>
</tr>
<tr>
<td>- Ethyl carbamate</td>
<td>- Chloroform</td>
<td></td>
</tr>
<tr>
<td>- Hexamethylphosphoramid</td>
<td>- 3-chloro-2-methylpropene</td>
<td></td>
</tr>
<tr>
<td>- 2-nitropropane</td>
<td>- 1,2-dibromoethane</td>
<td></td>
</tr>
<tr>
<td>- Styrene</td>
<td>- 1,4-dichlorobenzene</td>
<td></td>
</tr>
<tr>
<td>- Thiourea</td>
<td>- 1,2-dichloroethane</td>
<td></td>
</tr>
<tr>
<td>- Thioacetamide</td>
<td>- 2,2-dichloroethane</td>
<td></td>
</tr>
<tr>
<td>- Arsenic / compounds</td>
<td>- 1,3-dichloropropene</td>
<td></td>
</tr>
<tr>
<td>- Chromium / compounds</td>
<td>- Hexachlorobenzene</td>
<td></td>
</tr>
<tr>
<td>- Thorium dioxide</td>
<td>- Methyl iodide</td>
<td></td>
</tr>
<tr>
<td>- Beryllium and certain beryllium compounds</td>
<td>- Tetrachloroethylene</td>
<td></td>
</tr>
<tr>
<td>- Cadmium / compounds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Lead / compounds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Nickel / compounds</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Selenium sulfide</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2,4,6-trichlorophenol</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Aromatic amines                                             |                                                             |                                                    |
| - 4-aminobiphenyl                                           |                                                             |                                                    |
| - Benzidine                                                 |                                                             |                                                    |
| - O-naphthylamine                                           |                                                             |                                                    |
| - Aniline                                                   |                                                             |                                                    |
| - O-anisidine                                               |                                                             |                                                    |
| - 2,4-diaminotoluene                                        |                                                             |                                                    |
| - O-toluidine                                               |                                                             |                                                    |
**GPR006 and GPR007– Reliability and Maintainability**

The system has been tested for reliability and maintainability (Test 4.1.2), but analyses of MTBF and MTTR were not performed. Therefore, it is unknown whether the system meets GPR006-01 of GPR007-01. However, the brief analysis below can help reduce MTTR.

The SoW specifies that the MTTR be less than one week. However, given that orders can take a week or longer to deliver, MTTR can be significantly decreased by maintaining an inventory of replacement parts available in case of a failure. Important parts to have on hand include high-voltage isolation relays, safety relays, fully populated DAQ PCBs, LiFePO4 batteries, and indicator LEDs.

**GPR008 – Manufacturability**

An updated manufacturability analysis was not performed for the LPRDS-BMS2010 project; however the analysis from the 2009 team was understood and applied to the 2010 design. The 2009 manufacturability analysis is included in the Appendix.

**GPR009/010 – Life Cycle Sustainability and Ethics Report**

In order to completely understand global sustainability and ethics, members of the LPRDS-BMS2010 team attended a lecture by Dr. Bill Lynn, visiting professor of Environmental Studies at Williams College. During this lecture James Cameron’s Avatar was examined from a critical perspective to explore the main themes of the movie and how they can be applied to our modern conception of global sustainability and ethics.

Dr. Lynn argued that that struggle of the Na’vi people against the destructive forces of the human mining corporation (RDA Corporation) highlights the dangers in operating in modern society within an anthropocentric view. In other words, the belief that humans are at the center of the moral community and that we have intrinsic value while other species only have instrumental value, can lead to disturbing and unsustainable outcomes. The RDA Corporation from Avatar embodies this moral view and as a result does not include the Na’vi people or the other natural species and ecosystems of the planet Pandora in their moral community. From this moral framework, RDA Corporation feels justified in destroying the lush natural habitats of Pandora and driving the Na’vi people out of their home in order to more effectively mine the precious natural resource Unobtainium on Pandora.

In contrast to the anthropocentric moral framework, is the geocentric moral framework that Dr. Lynn is advocating. Within this moral framework all species are included in the moral community and have intrinsic value. While operating within this framework a value is placed on maintaining biodiversity. Dr. Lynn argues that only from this moral view can we achieve sustainable and ethical practices.

The LPRDS-BMS2010 team operates within the geocentric moral framework as well as abiding by all that is included in the IEEE Codes of Ethics. In all design considerations both the IEEE Codes of ethics and the implications of operating within the geocentric framework are referenced. For a further discussion of sustainability and ethics the 2009 sustainability and ethics reports have been included in the Appendix.
MAINTENANCE MANUAL

The maintenance manual is a low level document that explains the unique technical principles and details of system operation. The maintenance manual also includes information on any advanced maintenance or calibration techniques that could be applied by an expert maintainer. A set of schematics, pin outs of all connectors, interface control documents, communication protocols, signal assignments of all cables, PCB board layouts, and the semantics of all interfaces are included within this manual.

System Operational States

Figure 46 shows the state transition diagram for the LPRDS as a whole. When the system is in the OFF state, there is no power delivered between the RPI, ESS, FIB, and SC; and the Fit PC is off; but power may still be provided to the DAQ boards from a wall outlet. Pushing the power button to turn on the Fit PC will cause the system to enter the BOOTING state, at which point the LPRDS software loads. If high voltage is detected at either the PV array or the ESS, the system automatically enters the HV UNSAFE state. Otherwise, SCADAS software checks the safety loop and as long as the safety loop is unbroken, the system automatically enters the LV STANDBY UNSAFE state. If any high voltage is detected in the LV STANDBY UNSAFE state, the system will transition to the HV UNSAFE state.

From the HV UNSAFE state, a user can manually turn off any high voltage, which results in a transition to the LV STANDBY UNSAFE state. From the LV STANDBY UNSAFE state, a user can clear faults by pushing the green button on the RPI, which will allow the system to enter the LV STANDBY SAFE state. From the LV STANDBY SAFE state, any low voltage fault will return the system to the LV STANDBY UNSAFE state, and any high voltage fault will send the system to the HV UNSAFE state. From the LV STANDBY SAFE state, turning on the ESS will allow the system to transition to the OPERATIONAL state, at which point the system is fully operational and apps can be run.

Apps can only be run from the operational state. The system should only be shut down from the LV STANDBY UNSAFE or LV STANDBY SAFE states. Turning off the Fit PC (by pushing its power button) from either of the LV states will send the system to the SHUTDOWN state, in which software shuts down. If this is the case, there is no need to disconnect hardware, because in order for the system to be in an LV state, high voltage must already have been disconnected. The Fit PC can also be shut down from other states, simply by turning it off via its power button, but this is inadvisable since high voltages may still be present. The system transitions automatically from the SHUTDOWN state to the OFF state.
Figure 51: LPRDS System Operational State Diagram
Interface Control Diagrams

Interface control diagrams are system drawings that document the exact way subsystem types are instantiated and interconnected. All connectors and cabling assignments are documented in the ICD’s.

Top Level System Diagram

![Image of Top Level ICD Diagram]

Figure 52: Top Level ICD
Raw Power Interface (RPI)

Figure 53: RPI ICD
Energy Storage System (ESS)

Figure 54: ESS ICD
Switch Controller (SC)

Figure 55: SC ICD
Filter Inverter Box (FIB)

Figure 56: FIB ICD
Figure 57: SCADA Top Level Diagram
Schematics/Subsystem Wiring Diagrams

The ESS and RPI were designed by the 2009 design team. The schematics/wiring diagrams are from last year’s design team.

Energy Storage System Schematic – from 2009 design team

![ESS Schematic 1](image_url)

Figure 58: ESS Schematic 1
Figure 59: ESS Schematic 2
Figure 60: ESS Schematic 3
Figure 61: RPI Wiring Diagram
Figure 62: RPI Wiring Diagram 2
Filter Inverter Box Wiring Diagram

Figure 63: FIB Wiring Diagram
Figure 64: FIB Box Layout

Filter Schematic

Figure 65: Filter Schematic
Filter Wiring Diagram

Figure 66: Filter Assembly Drawing

Figure 67: Filter Assembly Top View
Inverter Schematic

Figure 68: Inverter Schematic 1

Figure 69: Inverter Schematic 2
Inverter PCB Layout

Figure 70: Inverter Schematic

Figure 71: Inverter PCB
Silk Screen

Figure 72: Inverter PCB Silk Screen

Top View

Figure 73: Inverter PCB Top View
Figure 74: Inverter PCB Bottom View

Figure 75: ESS DAQ Board Schematic 1
ESS DAQ Board PCB 2010 Redesign

Figure 78: ESS DAQ Board Redesign Top Level
Figure 79: ESS DAQ Board Redesign Silk Screen
Figure 80: ESS DAQ Board Layout Bottom View
RPI DAQ Board PCB 2009 design team

Figure 81: RPI DAQ Board Layout

RPI DAQ Board PCB Schematic – 2009 design team

Figure 82: RPI DAQ Schematic
Figure 83: RPI DAQ Schematic 2
Switch Controller Wiring Diagram

Figure 84: Switch Controller Wiring Diagram
Figure 85: Safety to Software Interface Board Wiring Diagram
High Voltage Cable Diagram

**High Voltage (HV) ICD**
Last Modified: April 27, 2010

- **PV ARRAY**
- **RPI-A1**
- **SC-A5**
- **FIB-A2**
- **ESS-A3**

**CABLE PIN CONFIGURATION**
**RECEIVER PIN CONFIGURATION**

**TYPE-A**

**TYPE-B**

**TYPE-C**

**ADJUSTABLE LOAD BOX**

**HV SUPPLY**

---

**Figure 86: High Voltage Cable Diagram**
Cable Assignments

RPI Connections (A1)

In the RPI (Raw Power Interfacet) enclosure, there are five different external voltage and signal cables connected. The PV (Photo Voltaic array) high voltage DC (direct current) is hard wired through conduit (single wires through a piece of conduit mounted to the RPI enclosure). The wiring of the PV high voltage DC to the inside (top section of the enclosure panel) of the RPI is as follows:

10 AWG Red wire - connects to - HV+ terminal
10 AWG Black wire - connects to - HV- terminal
8 AWG Green wire - connects to - Ground terminal

The J13 connector, for High Voltage DC distribution to the SC, is a four pin Anderson Power “Power Pak®” connector, located on the bottom of the enclosure, configured for three contacts and wired with a 14 AWG wires:

Pin 1 a 14 AWG Red wire - connects to – HV4 terminal
Pin 2 a 14 AWG Black wire - connects to - HV- terminal
Pin 3 a 14 AWG Green wire - connects to - GND terminal

The J12 connector, for system low voltage DC supply from ESS, is a six pin Amp/Tyco “Mate-N-Lok” connector, located on the bottom of the enclosure and wired with a 18/6 cable (18 AWG wire, 6 conductor cable):

Pin 1 an 18 AWG Red wire - connects to - +12V terminal
Pin 2 an 18 AWG Black wire - connects to – 12VCOM terminal
Pin 3 an 18 AWG White wire - connects to – 5VCOM terminal
Pin 4 an 18 AWG Green wire - connects to - +5V terminal
Pin 5 an 18 AWG Orange wire - spare
Pin 6 an 18 AWG Blue wire – spare

The J11 connector, for the Safety system interface leaving RPI to the ESS, is a four pin Amp/Tyco “Mate-N-Lok” connector, located on the bottom of the enclosure and wired with a 18/4 cable (18 AWG wire, 4 conductor cable):

Pin 1 an 18 AWG White wire - connects to – 1D terminal
Pin 2 an 18 AWG Green wire - connects to – 1E terminal
Pin 3 an 18 AWG Red wire - connects to – 1H terminal
Pin 4 an 18 AWG Black wire – not connected

The J14 connector, for the system RS485 communication, is an L-COM Cat 5E RJ45 coupler connector, located on the bottom of the enclosure. An inexpensive Cat 5E cable (4 twisted pair cable – 8 24 AWG conductors) connects to the inside of the coupler. The other end of the Cat 5E cable will be cut to length and stripped (no more than 1/2” on the end) to connect the following wires:
Pin 7 a 24 AWG White/Brown wire - connects to – RPI DAQ RS485 B terminal
Pin 8 a 24 AWG Brown wire - connects to on RPI DAQ RS485 A terminal

**FIB Connections (A2)**

In the FIB (Filter Inverter Box) enclosure, there are eight different voltage and signal cables connected. The 205 HVDC (High Voltage DC - direct current) from the SC (Switch controller) enters the FIB section (through connector J21) where the voltage is inverted through the H- Bridge, passed through a filter and then sent to the 120 VAC (120 Volts AC - alternating current). This output 120 VAC is sent to the Xfmr (transformer) though the connector J28. The wiring of the FIB connectors follows:

The J21 connector (connected to cable WCF2 from SC-A5), for High Voltage DC distribution, is a four pin Anderson Power “Power Pak®” connector, wired with an 14/3 cable (3 cables consisting of 14 AWG wire):
Pin 1 a 14 AWG Red wire - connects to - High Voltage DC HV+ terminal
Pin 2 a 14 AWG Black wire - connects to - High Voltage DC Ground HV- terminal
Pin 3 a 14 AWG Green wire - connects to - GND terminal

The J22 connector (connected to cable WEF3 from ESS-A3), for the Safety system interface from ESS-A3, is a four pin Amp/Tyco “Mate-N-Lok” connector, located in the enclosure and wired with an 18/4 cable (4 cables consisting of 18 AWG wire):
Pin 1 an 18 AWG White wire - connects to – J36 Pin 1 (18 AWG White wire)
Pin 2 an 18 AWG Green wire - connects to – J36 Pin 2 (18 AWG Green wire)
Pin 3 an 18 AWG Red wire - connects to – J36 Pin 3 (18 AWG Red wire)
Pin 4 an 18 AWG Black wire - connects to – J36 Pin 4 (18 AWG Black wire)
The J23 connector (connected to cable WCF1 to SC-A5), for the Safety system interface to SC-A5, is a four pin Amp/Tyco “Mate-N-Lok” connector, located in the enclosure and wired with an 18/4 cable (4 cables consisting of 18 AWG wire):
Pin 1 an 18 AWG White wire - connects to – J57 Pin 1 (18 AWG White wire)
Pin 2 an 18 AWG Green wire - connects to – J57 Pin 2 (18 AWG Green wire)
Pin 3 an 18 AWG Red wire - connects to – J57 Pin 3 (18 AWG Red wire)
Pin 4 an 18 AWG Black wire - connects to – J57 Pin 4 (18 AWG Black wire)

The J24 connector (connected to cable W2), for the USB to Jtag communication, used for software downloads, is an L-COM USB (a) to USB (b) jack. A short USB cable connects the “Jtag to USB converter” to the J24 jack. The “Jtag to USB converter connects to the FIB PCB board.

The J25 connector (connected to cable WEF1A from Safety to Software Interface Board), for system low voltage DC supply, is a six pin Amp/Tyco “Mate-N-Lok” connector, located on the bottom of the enclosure and wired with an 18/2 cable (2 cables consisting of 18 AWG wire):
Pin 1 an 18 AWG Red wire - connects to - +12V terminal
Pin 2 an 18 AWG Black wire - connects to – 12VCOM terminal

The J26 (connected to cable WFS1 to SIB), for the system RS485 communication, are L-COM Cat 5E RJ45 coupler connectors mounted in the enclosure. An inexpensive Cat 5E cable connects to the inside of the coupler and the other end of the cable will be cut to length and stripped (no more than 1/2” on the end) to connect the following wires:
Pin 7 a 24 AWG White/Brown wire - connects to – J45 RS485 SIBR2 (to RS485 -B)
Pin 8 a 24 AWG Brown wire - connects to – J45 RS485 SIBR2 (to RS485 -A)

The J27 (connected to cable WEF2 from ESS-A3), for the system RS485 communication, are L-COM Cat 5E RJ45 coupler connectors mounted in the enclosure. An inexpensive Cat 5E cable connects to the inside of the coupler and the other end of the cable will be cut to length and stripped (no more than 1/2” on the end) to connect the following wires:
Pin 7 a 24 AWG White/Brown wire - connects to – J39 ESS7 PCB J5 (to RS485 -B)
Pin 8 a 24 AWG Brown wire - connects to – J39 ESS7 PCB J5 (to RS485 -A)

The J28 connector (connected to cable WX1 to Xfmr), for delivery of 120 VAC to the Isolation Transformer, is a three connection mini AC receptacle and wired with 12/3 AWG wire (3 cables consisting of 12 AWG wire):
Gold Screw – 12 AWG Black wire - connects to – Filter PCB 120V terminal  
Silver Screw – 12 AWG White wire - connects to – Filter PCB Neutral terminal  
Green Screw – 12 AWG Green wire - connects to - GND terminal  

*ESS Connections (A3)*  
In the ESS (Energy Storage System) enclosure, there are nine different voltage and signal cable connections. High voltage DC enters the ESS section (through connector J38) where the voltage charges the storage batteries and provides system low voltage (+12V and +5V) through DC to DC converters. The stored battery energy will also leave the ESS section through connector J38. Wiring of the ESS connections follows:

The J38 connector for High Voltage DC distribution between batteries and SC, is a four pin Anderson Power “Power Pak®” connector, configured for three contacts and wired with a 14 AWG wire:  
Pin 1 a 14 AWG Red wire - connects to - High Voltage DC HV terminal  
Pin 2 a 14 AWG Green wire - connects to - GND terminal  
Pin 3 a 14 AWG Black wire - connects to - High Voltage DC Ground HVGND terminal

The J30 (connected to RPI), J37 (connected to SC) and J34 (connected to FIB) connectors, for system low voltage DC supply distribution, are six pin Amp/Tyco “Mate-N-Lok” connectors, located in the enclosure and wired with an 18/6 cable:  
Pin 1 an 18 AWG Red wire - connects to - +12V terminal  
Pin 2 an 18 AWG Black wire - connects to - COM terminal  
Pin 3 an 18 AWG White wire - connects to – 5VCOM terminal  
Pin 4 an 18 AWG Green wire - connects to - +5V terminal  
Pin 5 an 18 AWG Orange wire - spare  
Pin 6 an 18 AWG Blue wire - spare

The J31 connector (connected to RPI) and J36 connector (connected to FIB), for the Safety system interface, is a four pin Amp/Tyco “Mate-N-Lok” connector, located in the enclosure and wired with an 18/4 cable:  
Pin 1 an 18 AWG White wire - connects to - 1E  
Pin 2 an 18 AWG Green wire - connects to – blank terminal  
Pin 3 an 18 AWG Red wire - connects to - 1H terminal
Pin 4 an 18 AWG Black wire - connects to – C2 terminal

The J35 connector (connected to FIB), and J39 connector (connected to the SIB) for the system RS485 communication to FIB, is an L-COM Cat 5E RJ45 coupler connector mounted in the enclosure. An inexpensive Cat 5E cable connects to the inside of the coupler and the other end of the cable will be cut to length and stripped (no more than 1/2” on the end) to connect the following wires:

Pin 7 a 24 AWG White/Brown wire - connects to – ESS DAQ RS485 B terminal
Pin 8 a 24 AWG Brown wire - connects to – ESS DAQ RS485 A terminal

The J32 connector is an input from the AC Mains. That provides the system with 120V 60Hz signal that can be used to power the DC/AC to DC converter to provide system power so it can be tested even if the system is not creating its own power from the battery bank.

The J34 connector, for system low voltage DC supply from ESS, is a six pin Amp/Tyco “Mate-N-Lok” connector, located on the bottom of the enclosure and wired with a 18/6 cable (18 AWG wire, 6 conductor cable):
Pin 1 an 18 AWG Red wire - connects to - +12V terminal
Pin 2 an 18 AWG Black wire - connects to – 12VCOM terminal
Pin 3 an 18 AWG White wire - connects to – 5VCOM terminal
Pin 4 an 18 AWG Green wire - connects to - +5V terminal
Pin 5 an 18 AWG Orange wire - spare
Pin 6 an 18 AWG Blue wire – spare

SC Connections (A5)

In the SC (Switch Controller) enclosure, there are eight different voltage and signal cable connections.

The J51 connector (from RPI) and J53 connector (from ESS) ) for the system RS485 communication to SC, is an L-COM Cat 5E RJ45 coupler connector mounted in the enclosure. An inexpensive Cat 5E cable connects to the inside of the coupler and the other end of the cable will be cut to length and stripped (no more than 1/2” on the end) to connect the following wires:

Pin 7 a 24 AWG White/Brown wire - connects to – SC DAQ RS485 B terminal
Pin 8 a 24 AWG Brown wire - connects to – SC DAQ RS485 A terminal
The J52 connector (from RPI), J56 connector (to FIB), and J54 connector (to ESS) are for High Voltage DC distribution between batteries, RPI, and FIB, is a four pin Anderson Power “Power Pak®” connector, configured for three contacts and wired with a 14 AWG wire:

- Pin 1 a 14 AWG Red wire - connects to - High Voltage DC HV terminal
- Pin 2 a 14 AWG Green wire - connects to - GND terminal
- Pin 3 a 14 AWG Black wire - connects to - High Voltage DC Ground HVGND terminal

The J55 connector (from ESS), for system low voltage DC supply from ESS, is a six pin Amp/Tyco “Mate-N-Lok” connector, located on the bottom of the enclosure and wired with a 18/6 cable (18 AWG wire, 6 conductor cable):

- Pin 1 an 18 AWG Red wire - connects to - +12V terminal
- Pin 2 an 18 AWG Black wire - connects to – 12VCOM terminal
- Pin 3 an 18 AWG White wire - connects to – 5VCOM terminal
- Pin 4 an 18 AWG Green wire - connects to - +5V terminal
- Pin 5 an 18 AWG Orange wire - spare
- Pin 6 an 18 AWG Blue wire – spare

The J57 connector (to FIB), for the Safety system interface, is a four pin Amp/Tyco “Mate-N-Lok” connector, located in the enclosure and wired with an 18/4 cable:

- Pin 1 an 18 AWG White wire - connects to - 1E
- Pin 2 an 18 AWG Green wire - connects to – blank terminal
- Pin 3 an 18 AWG Red wire - connects to - 1H terminal
- Pin 4 an 18 AWG Black wire - connects to – C2 terminal

The J58 connector (to safety to software interface board) is a custom made cable. For more information on the wiring of this connection, please see the wiring diagram for the Safety to Software Interface Board.

Other System Connections

SIB

The SIB (safety interface box) has four cable connections. The J43 connection (to Sunny Boy) and the J45 connection (from FIB) are for the system RS485 communication. The J44 connection is a 25 pin connector that includes the safety 12V indicator and relay, as well as the relay for the demo outlet. The J46 connection is a USB connection that connects the FitPC to the SIB.
Safety to Software Interface Board

The A connection is connected to the alarm and is the signal that sets off the alarm of the system. The B connection is connected to the alarm acknowledge that is on the outside of the tower. When pressed, this signal shuts off the alarm. The X and Z connection is the safety 12V indicator and relay from the SIB. The S connection is the safety loop from the SC. The MP connection is the system 12V that is made by the ESS. It is the second half of a split cable WEF1B. The E connection is from the big red button on the side of the tower that tells the system to set off the alarm. It is the manual alarm switch.

Output Display Box

The PicoLCD is controlled by a USB connection that is attached to the USB hub through wire WDU1. The demo outlet is controlled by a relay that is connected to the SIB and wire WSO1. The frequency gauge and voltage gauges are controlled by the output of the transformer. This connection is done with wire WP4.

FitPC

The connection to the network is through Ethernet. This connection is through wire WPC4. The connection to the System Display is a VGA chord labeled WPC3. The connection to the USB hub is a USB connection that is labeled WPC2. The power being provided to the Fit is from the Mains Power Strip through chord WP2A.

USB Hub

The connection to the FitPC is a USB connection that is labeled WPC2. The connection to the Mouse is a USB connection that is labeled WUM1. The connection to the Keyboard is a USB connection that is labeled WUK1. The PicoLCD is controlled through a USB connection that is labeled WDU1.

Mains Power Strip

The power for the system display is from the Mains Power Strip and is labeled WP3. The FitPC is powered from the Mains Power Strip and is labeled WP2. The J32 on the ESS is an input from the AC Mains. That provides the system with 120V 60Hz signal that can be used to power the DC/AC to DC converter to provide system power so it can be tested even if the system is not creating its own power from the battery bank.

LPRDS Power Strip

The J62 connector from the transformer connects the power to the power supply from the transformer. It is done by cable WPS2. The frequency gauge and voltage gauges in the output display box are controlled by the output of the transformer, which is the same as the LPRDS power strip. This connection is done with wire WP4. The wall wort is directly plugged into the LPRDS Power Strip. Its purpose is in testing the output of the transformer.
# Bill of Materials

RPI – from 2009 design team

<table>
<thead>
<tr>
<th>Ref #</th>
<th>Description</th>
<th>Supplier</th>
<th>Price</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>RTL1045500 AE Enclosure</td>
<td>Stokes Electric</td>
<td>$161.12</td>
<td>$161.12</td>
</tr>
<tr>
<td>A1A</td>
<td>RTL2576000 AE Thumb Turn Handle w/ Key Lock</td>
<td>Stokes Electric</td>
<td>$17.42</td>
<td>$17.42</td>
</tr>
<tr>
<td>RP01</td>
<td>Fuse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP01A</td>
<td>CB1038-1 Fuse Block</td>
<td>Seagate Control Systems</td>
<td>$5.22</td>
<td>$5.22</td>
</tr>
<tr>
<td>RP01B</td>
<td>DCM-15 Fuses</td>
<td>Steven Engineering</td>
<td>$15.12</td>
<td>$15.12</td>
</tr>
<tr>
<td>RP02</td>
<td>W20AB Current Transformer</td>
<td></td>
<td>$235.00</td>
<td>$235.00</td>
</tr>
<tr>
<td>RP03</td>
<td>Safety Relay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP03A</td>
<td>682-0157 800V Solid State Relay</td>
<td>Allied Electronics</td>
<td>$151.18</td>
<td>$151.18</td>
</tr>
<tr>
<td>RP03B</td>
<td>682-0066 Heat Sink</td>
<td>Allied Electronics</td>
<td>$13.54</td>
<td>$13.54</td>
</tr>
<tr>
<td>RP04</td>
<td>RCMA420-D-1 Ground Fault Monitor</td>
<td></td>
<td>$460.00</td>
<td>$460.00</td>
</tr>
<tr>
<td>RP05</td>
<td>Safety System</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP05A</td>
<td>814-0442 Ice Cube Relay</td>
<td>Allied Electronics</td>
<td>$5.49</td>
<td>$5.49</td>
</tr>
<tr>
<td>RP05B</td>
<td>814-0467 Relay Base</td>
<td>Allied Electronics</td>
<td>$6.20</td>
<td>$6.20</td>
</tr>
<tr>
<td>RP06</td>
<td>Manual Switches</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP06A</td>
<td>814-5023 Extended Red Pushbutton</td>
<td>Allied Electronics</td>
<td>$15.21</td>
<td>$15.21</td>
</tr>
<tr>
<td>RP06B</td>
<td>814-5001 Flash Green Pushbutton</td>
<td>Allied Electronics</td>
<td>$10.04</td>
<td>$10.04</td>
</tr>
<tr>
<td>RP07</td>
<td>RPI Board</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RP08</td>
<td>511-1184 6V Pilot Light (Grn)</td>
<td>Allied Electronics</td>
<td>$3.08</td>
<td>$3.08</td>
</tr>
<tr>
<td>RP08</td>
<td>511-1190 6V Pilot Light (Yel)</td>
<td>Allied Electronics</td>
<td>$3.08</td>
<td>$3.08</td>
</tr>
<tr>
<td>Misc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M01</td>
<td>S-2214 High Voltage Warning Sticker</td>
<td>MySafetySign.com</td>
<td>$8.95</td>
<td>$8.95</td>
</tr>
<tr>
<td>M02</td>
<td>EP2.5/4UN End Section</td>
<td>Seagate Control Systems</td>
<td>$0.15</td>
<td>$0.45</td>
</tr>
<tr>
<td>M03</td>
<td>CGT4N Terminal Block</td>
<td>Seagate Control Systems</td>
<td>$3.54</td>
<td>$3.54</td>
</tr>
<tr>
<td>M04</td>
<td>CTS4U-N/BG Terminal Block</td>
<td>Seagate Control Systems</td>
<td>$6.48</td>
<td>$6.48</td>
</tr>
<tr>
<td>M05</td>
<td>58K7478 MOV</td>
<td>Newark Electronics</td>
<td>$1.00</td>
<td>$1.00</td>
</tr>
<tr>
<td>M06</td>
<td>2511120/1M 35 mm Aluminum DIN Rail</td>
<td>Seagate Control Systems</td>
<td>$2.50</td>
<td>$2.50</td>
</tr>
<tr>
<td>M07</td>
<td>4' 90F2840 Flexible Conduit</td>
<td>Newark Electronics</td>
<td>$4.88</td>
<td>$4.88</td>
</tr>
<tr>
<td>M07</td>
<td>2 SLU25 Ground Lug</td>
<td>ElecDirect</td>
<td>$2.56</td>
<td>$2.56</td>
</tr>
</tbody>
</table>

**TOTAL** $1,132.06
## ESS PCB BOM

<table>
<thead>
<tr>
<th>Description</th>
<th>Needed</th>
<th>Extra</th>
<th>Total</th>
<th>Total Cost</th>
<th>Vender</th>
<th>Vender Part#</th>
<th>Price Ea.</th>
</tr>
</thead>
<tbody>
<tr>
<td>400 HV cap - 4.7uF-part -unisolated hv-5v</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>$2.37</td>
<td>digikey</td>
<td>565-1411-ND</td>
<td>$0.79</td>
</tr>
<tr>
<td>63v cap - 100uf -part- unisolated hv-5v</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>$0.93</td>
<td>digikey</td>
<td>P5193-ND</td>
<td>$0.31</td>
</tr>
<tr>
<td>1mh inductor - part- unisolated hv-5v</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>$8.61</td>
<td>digikey</td>
<td>DN7750-ND</td>
<td>$2.87</td>
</tr>
<tr>
<td>4.3K ohm 1/4W axial resistor</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>4.3KQBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>510K ohm 1/4W axial resistor</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>510KQBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>270 ohm 1/4W axial resistor</td>
<td>12</td>
<td>12</td>
<td>24</td>
<td>$1.54</td>
<td>digikey</td>
<td>270QBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>10K ohm 1/4W axial resistor</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>$1.02</td>
<td>digikey</td>
<td>10KQBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>560 ohm 1/4W axial resistor</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>560QBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>150 ohm 1/4W axial resistor</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>$0.51</td>
<td>digikey</td>
<td>150QBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>120 ohm 1/4W axial resistor</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>120QBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>910 ohm 1/4W axial resistor</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>$0.38</td>
<td>digikey</td>
<td>910QBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>750 ohm 1/4W axial resistor</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>750QBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>1.5K ohm 1/4W axial resistor</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>1.5KQBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>30K ohm 1/4W axial resistor</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$0.32</td>
<td>digikey</td>
<td>30KQBK-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>IC HS CMOS DUAL OP AMP 8-DIP</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>$13.24</td>
<td>digikey</td>
<td>OPA2350PA-ND</td>
<td>$3.31</td>
</tr>
<tr>
<td>PIC - IC MCU FLASH 24KX16 40DIP</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>$4.50</td>
<td>digikey</td>
<td>PIC18F4525-I/P-ND</td>
<td>$8.90</td>
</tr>
<tr>
<td>IC COMPARATOR HS DUAL 14-DIP</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>$2.40</td>
<td>digikey</td>
<td>LM2901NNS-ND</td>
<td>$1.20</td>
</tr>
<tr>
<td>Current Transducer</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$14.00</td>
<td>digikey</td>
<td>398-1025-5-ND</td>
<td>$14.00</td>
</tr>
<tr>
<td>Thermistor I.C.</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>$3.44</td>
<td>digikey</td>
<td>MCP9701A-E/TO-ND</td>
<td>$0.43</td>
</tr>
<tr>
<td>TXRX ISO BUS 1MBS RS485 16SOIC</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>$12.80</td>
<td>digikey</td>
<td>390-1097-5-ND</td>
<td>$6.40</td>
</tr>
<tr>
<td>CAP 4700PF 50V CERAMIC X7R 0805</td>
<td>6</td>
<td>12</td>
<td>18</td>
<td>$1.10</td>
<td>digikey</td>
<td>399-1166-1-ND</td>
<td>$0.06</td>
</tr>
<tr>
<td>CAP 4700PF 50V CERAMIC X7R 0805</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>$0.47</td>
<td>digikey</td>
<td>399-1155-1-ND</td>
<td>$0.04</td>
</tr>
<tr>
<td>Testing LED for PIC outputs</td>
<td>12</td>
<td>0</td>
<td>12</td>
<td>$8.04</td>
<td>digikey</td>
<td>516-1404-ND</td>
<td>$0.67</td>
</tr>
<tr>
<td>2 terminal block</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>$1.30</td>
<td>digikey</td>
<td>A97984-ND</td>
<td>0.65</td>
</tr>
<tr>
<td>4 terminal block</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>$2.64</td>
<td>digikey</td>
<td>A97986-ND</td>
<td>0.88</td>
</tr>
<tr>
<td>6 terminal block</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$1.32</td>
<td>digikey</td>
<td>A97988-ND</td>
<td>1.32</td>
</tr>
<tr>
<td>12 terminal block</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>$7.32</td>
<td>digikey</td>
<td>1-1546217-2-ND</td>
<td>1.83</td>
</tr>
<tr>
<td>4 terminal block HV</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$1.33</td>
<td>digikey</td>
<td>A98115-ND</td>
<td>1.33</td>
</tr>
<tr>
<td>0.1UF SMD cap</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>$1.19</td>
<td>digikey</td>
<td>PCC1828CT-ND</td>
<td>$0.10</td>
</tr>
<tr>
<td>10UF smd cap</td>
<td>2</td>
<td>8</td>
<td>10</td>
<td>$2.75</td>
<td>digikey</td>
<td>587-1300-1-ND</td>
<td>$0.28</td>
</tr>
<tr>
<td>1UF smd cap</td>
<td>1</td>
<td>9</td>
<td>10</td>
<td>$0.95</td>
<td>digikey</td>
<td>399-1284-1-ND</td>
<td>$0.10</td>
</tr>
<tr>
<td>16pF Cap</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>$1.86</td>
<td>digikey</td>
<td>490-3630-ND</td>
<td>$0.31</td>
</tr>
<tr>
<td>Fault Indicator LED</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$0.15</td>
<td>Allied</td>
<td>670-1192</td>
<td>$0.15</td>
</tr>
<tr>
<td>Discharging Indicator LED</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$1.52</td>
<td>Allied</td>
<td>670-0169</td>
<td>$1.52</td>
</tr>
<tr>
<td>Charging Indicator LED</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$0.10</td>
<td>Allied</td>
<td>670-1057</td>
<td>$0.10</td>
</tr>
<tr>
<td>30V Indicator LED</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$0.12</td>
<td>Allied</td>
<td>670-1059</td>
<td>$0.12</td>
</tr>
<tr>
<td>Short Description</td>
<td>Value</td>
<td>Manufacturer</td>
<td>Man. Part Number</td>
<td>Total $</td>
<td>Vendor 1</td>
<td>Vendor $</td>
<td>Amount</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>------------------</td>
<td>---------</td>
<td>---------------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>Capacitor 33uF 25V</td>
<td>Nichicon</td>
<td>UVR1E330MDD1TD</td>
<td>$0.09</td>
<td>Mouser</td>
<td>$0.03</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Capacitor 1uF 50V</td>
<td>Panasonic - ECG</td>
<td>ECE-A1HKA010</td>
<td>$0.14</td>
<td>Digi-key</td>
<td>$0.14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Capacitor 10uF 25V</td>
<td>Panasonic</td>
<td>ECE-A1EKA100</td>
<td>$0.09</td>
<td>Mouser</td>
<td>$0.09</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Capacitor .001uF</td>
<td>Vishay/BC Components</td>
<td>A102K15X7RH5TAA</td>
<td>$0.08</td>
<td>Digi-key</td>
<td>$0.08</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Capacitor .1uF 50V</td>
<td>Vishay/BC Components</td>
<td>A104K15X7RF5TAA</td>
<td>$0.64</td>
<td>Digi-key</td>
<td>$0.08</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Capacitor 1.5uF 50V</td>
<td>Kemet</td>
<td>C330C155M5U5TA</td>
<td>$0.48</td>
<td>Allied Electronic</td>
<td>$0.48</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Capacitor .01uF 50V</td>
<td>Vishay/BC Components</td>
<td>A103K15X7RF5TAA</td>
<td>$0.07</td>
<td>Digi-key</td>
<td>$0.07</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Capacitor 100uF 450V</td>
<td>Nichicon</td>
<td>UPW2W101MRD</td>
<td>$3.85</td>
<td>Digi-key</td>
<td>$3.85</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Diode 400V 1A</td>
<td>Fairchild Semiconductor</td>
<td>1N4004</td>
<td>$0.81</td>
<td>Digi-key</td>
<td>$0.27</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Green LED</td>
<td>Lumex</td>
<td>SSL-LX3044GD</td>
<td>$0.40</td>
<td>Allied Electronic</td>
<td>$0.10</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Red LED</td>
<td>Lumex</td>
<td>SSL-LX3044SRD</td>
<td>$0.16</td>
<td>Allied Electronic</td>
<td>$0.16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Yellow LED</td>
<td>Lumex</td>
<td>SSL-LX3044YD</td>
<td>$0.12</td>
<td>Allied Electronic</td>
<td>$0.12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Amber LED</td>
<td>Panasonic - SSG</td>
<td>LN41YPH</td>
<td>$0.22</td>
<td>Digi-key</td>
<td>$0.22</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4pos header</td>
<td>Tyco Electronics</td>
<td>1776119-4</td>
<td>$1.88</td>
<td>Digi-key</td>
<td>$0.94</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2pos header</td>
<td>Tyco Electronics</td>
<td>1776119-2</td>
<td>$1.40</td>
<td>Digi-key</td>
<td>$0.70</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2pin hdr</td>
<td>$0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>IGBT - 600V 90A</td>
<td>S.T. Microelectronics</td>
<td>STGW45NC60WD</td>
<td>$18.80</td>
<td>Digi-key</td>
<td>$4.70</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Trans 40V</td>
<td>ON Semiconductor</td>
<td>P2N22222AG</td>
<td>$0.53</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>IGBT - 660V 40A</td>
<td>International Rectifier</td>
<td>IRG4PC30KD</td>
<td>$4.75</td>
<td>Digi-key</td>
<td>$4.75</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Terminal Quickfit</td>
<td>Keystone</td>
<td>1287</td>
<td>$1.10</td>
<td>Digi-key</td>
<td>$1.10</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 1.87k</td>
<td>Yageo</td>
<td>CFR-25JB-1K8</td>
<td>$0.06</td>
<td>Digi-key</td>
<td>$0.06</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 300ohm</td>
<td>Yageo</td>
<td>CFR-25JB-300R</td>
<td>$0.12</td>
<td>Digi-key</td>
<td>$0.06</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 10ohm</td>
<td>Yageo</td>
<td>CFR-25JB-10R</td>
<td>$0.24</td>
<td>Digi-key</td>
<td>$0.06</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 5k</td>
<td>Yageo</td>
<td>CFR-25JB-5K1</td>
<td>$0.24</td>
<td>Digi-key</td>
<td>$0.06</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 200k</td>
<td>Yageo</td>
<td>CFR-25JB-200K</td>
<td>$0.12</td>
<td>Digi-key</td>
<td>$0.06</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 1k</td>
<td>Yageo</td>
<td>CFR-25JB-1K0</td>
<td>$0.06</td>
<td>Digi-key</td>
<td>$0.06</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Item Description</td>
<td>Manufacturer</td>
<td>Part Number</td>
<td>Cost (Digi-key)</td>
<td>Cost (Other)</td>
<td>Qty</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------------</td>
<td>-------------------------------</td>
<td>----------------</td>
<td>-------------</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 7k</td>
<td>Yageo</td>
<td>CFR-25JB-6K8</td>
<td>$0.06</td>
<td>$0.06</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 3W 47k 3W</td>
<td>Tyco Electronics</td>
<td>ROX3J347K</td>
<td>$0.98</td>
<td>$0.33</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 3W 2.5k 3W</td>
<td>Vishay Dale</td>
<td>CW02B2K500JE12</td>
<td>$0.61</td>
<td>$0.61</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 750ohm</td>
<td>Yageo</td>
<td>CFR-25JB-750R</td>
<td>$0.26</td>
<td>$0.06</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>omitted Resistor 1/4W 10k</td>
<td>Yageo</td>
<td>CFR-25JB-10K</td>
<td>$0.13</td>
<td>$0.06</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 10k</td>
<td>Yageo</td>
<td>CFR-25JB-10K</td>
<td>$0.13</td>
<td>$0.06</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 25k</td>
<td>Yageo</td>
<td>CFR-25JB-24K</td>
<td>$0.12</td>
<td>$0.06</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Resistor 1/4W 1.5k</td>
<td>Yageo</td>
<td>CFR-25JB-1K5</td>
<td>$0.19</td>
<td>$0.06</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feedback Transformer</td>
<td>Triad Magnetics</td>
<td>F24-100</td>
<td>$6.74</td>
<td>$6.74</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microcontroller</td>
<td>Microcontrollershop.com</td>
<td>AVR-H128</td>
<td>$36.95</td>
<td>$36.95</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12V-5V</td>
<td>Lineage Power</td>
<td>AXA003A0XZ</td>
<td>$7.58</td>
<td>$7.58</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS485 Transceiver</td>
<td>Intersil</td>
<td>ISL8483EIP</td>
<td>$2.12</td>
<td>$2.12</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5V Shunt Regulator</td>
<td>LM136-5.0</td>
<td>2</td>
<td>$0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12V-12V</td>
<td>CUI Inc</td>
<td>VAWQ12-Q24-S12</td>
<td>$41.04</td>
<td>$41.04</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optocoupler</td>
<td>Lite-On Inc</td>
<td>6N135</td>
<td>$2.48</td>
<td>$1.24</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Half Bridge Driver</td>
<td>International Rectifier</td>
<td>IRS2111PBF</td>
<td>$5.94</td>
<td>$2.97</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24V Zener Diode</td>
<td>Vishay Semiconductors</td>
<td>1N4749A-TR</td>
<td>$0.04</td>
<td>$0.04</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feedback Transformer</td>
<td>Triad Magnetics</td>
<td>F24-100</td>
<td>$6.74</td>
<td>$6.74</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Microcontroller</td>
<td>Microcontrollershop.com</td>
<td>AVR-H128</td>
<td>$36.95</td>
<td>$36.95</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12V-5V</td>
<td>Lineage Power</td>
<td>AXA003A0XZ</td>
<td>$7.58</td>
<td>$7.58</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RS485 Transceiver</td>
<td>Intersil</td>
<td>ISL8483EIP</td>
<td>$2.12</td>
<td>$2.12</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5V Shunt Regulator</td>
<td>LM136-5.0</td>
<td>2</td>
<td>$0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>box</td>
<td>Case</td>
<td>Various</td>
<td>$59.95</td>
<td>$59.95</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>temp</td>
<td>Temp Sensor</td>
<td>GPD-032 ET3/30B-R</td>
<td>$3.95</td>
<td>$3.95</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hs</td>
<td>Heatsink</td>
<td>Aavid Thermalloy</td>
<td>$32.52</td>
<td>$32.52</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bergquist</td>
<td>$1.40</td>
<td>$0.28</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QC Female</td>
<td>Molex Connector Corp</td>
<td>19005-0010</td>
<td>$3.36</td>
<td>$0.42</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panel Mount</td>
<td>Anderson Power Products</td>
<td>1470G1</td>
<td>$1.53</td>
<td>$1.53</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Panel Mount Retainer</td>
<td>Anderson Power Products</td>
<td>110G9</td>
<td>$0.31</td>
<td>$0.31</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Housing - Black</td>
<td>Anderson Power Products</td>
<td>1327G6FP</td>
<td>$0.50</td>
<td>$0.50</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Housing - Red</td>
<td>Anderson Power Products</td>
<td>1327FP</td>
<td>$0.72</td>
<td>$0.72</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Housing - Green</td>
<td>Anderson Power Products</td>
<td>1327G5FP</td>
<td>$0.97</td>
<td>$0.97</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PowerPole Contact</td>
<td>Anderson Power Products</td>
<td>1845G1</td>
<td>$3.27</td>
<td>$1.09</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB connector</td>
<td>L-Com Connectivity</td>
<td>ECF504-BAS</td>
<td>$6.95</td>
<td>$6.95</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RJ-45 Connector</td>
<td>L-Com Connectivity</td>
<td>ECF504-SC5E</td>
<td>$21.62</td>
<td>$10.81</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terminal Blocks</td>
<td>Altech</td>
<td>CTS4U-N</td>
<td>$5.20</td>
<td>$0.65</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Term Block - End Plate</td>
<td>Altech</td>
<td>EP2.5/4UN</td>
<td>$0.40</td>
<td>$0.20</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC</td>
<td>Qty</td>
<td>Description</td>
<td>Vendor</td>
<td>Vendor Part</td>
<td>Mfg</td>
<td>Unit Price</td>
<td>Total Price</td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>--------------------------------------------------</td>
<td>-------------</td>
<td>----------------------</td>
<td>-----------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>DAQ</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>40A @ 480VDC Solid State Relay</td>
<td>MPJA</td>
<td>SDP4040D</td>
<td>Gold Control Tech</td>
<td>$16.95</td>
<td>$33.90</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Relay Heat Sink</td>
<td>Newark</td>
<td>93K7581</td>
<td>Crydom</td>
<td>$18.29</td>
<td>$36.58</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>DIN mounting rail</td>
<td>Allied</td>
<td>502-0156</td>
<td>Altech</td>
<td>$4.10</td>
<td>$4.10</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>Terminal Blocks</td>
<td>Mouser</td>
<td>845-CTS4U-N</td>
<td>Altech</td>
<td>$0.65</td>
<td>$10.40</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Terminal Blocks - End Stop</td>
<td>Allied</td>
<td>502-3212</td>
<td>Altech</td>
<td>$0.20</td>
<td>$0.40</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>Terminal Blocks - End Plate</td>
<td>Mouser</td>
<td>845-E0.2.5/4UN</td>
<td>Altech</td>
<td>$0.37</td>
<td>$0.74</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>Terminal Block - 3Pos Jumper</td>
<td>Mouser</td>
<td>845-CA722/3</td>
<td>Altech</td>
<td>$0.56</td>
<td>$1.68</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>PowerPole Housing - Panel Mount</td>
<td>Allied</td>
<td>803-0148</td>
<td>Anderson Power</td>
<td>$1.53</td>
<td>$4.59</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>PowerPole - Panel Mount Retainer</td>
<td>Allied</td>
<td>803-0158</td>
<td>Anderson Power</td>
<td>$0.31</td>
<td>$0.93</td>
</tr>
<tr>
<td>11</td>
<td>3</td>
<td>PowerPole Housing - Black</td>
<td>Allied</td>
<td>803-0116</td>
<td>Anderson Power</td>
<td>$0.50</td>
<td>$1.50</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>PowerPole Housing - Red</td>
<td>Allied</td>
<td>803-0114</td>
<td>Anderson Power</td>
<td>$0.72</td>
<td>$2.16</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>PowerPole Housing - Green</td>
<td>Allied</td>
<td>803-0265</td>
<td>Anderson Power</td>
<td>$0.97</td>
<td>$2.91</td>
</tr>
<tr>
<td>14</td>
<td>3</td>
<td>PowerPole Contact</td>
<td>DigiKey</td>
<td>A14283-ND</td>
<td>Anderson Power</td>
<td>$1.09</td>
<td>$3.27</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>Safety connector</td>
<td>DigiKey</td>
<td>A14283-ND</td>
<td>Tyco Electronics</td>
<td>$0.53</td>
<td>$1.06</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>RJ-45 Connector</td>
<td>Allied</td>
<td>665-0261</td>
<td>L-Com Connectivity</td>
<td>$10.81</td>
<td>$21.62</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>Power connector</td>
<td>DigiKey</td>
<td>A14283-ND</td>
<td>Tyco Electronics</td>
<td>$0.53</td>
<td>$0.53</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>Res 10 Ohm 1% 50W Alum Housed WW</td>
<td>DigiKey</td>
<td>RHRD-10-ND</td>
<td>Vishay/Dale</td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>Temp Sensor safety</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>$3.95</td>
<td>$3.95</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>LCD Temp Sensor &amp; Probe</td>
<td>MPJA</td>
<td>16370 ME</td>
<td>N/A</td>
<td>$8.95</td>
<td>$35.80</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>LCD display</td>
<td>MPJA</td>
<td>16177 ME</td>
<td>Colluck</td>
<td>$9.95</td>
<td>$39.80</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>LED Kit</td>
<td>MPJA</td>
<td>16313 OP</td>
<td></td>
<td>$9.95</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Part Desc.</td>
<td>Vendor Part No.</td>
<td>Manufacturer Part No</td>
<td>Supplier</td>
<td>Unit Cost</td>
<td>Qty</td>
</tr>
<tr>
<td>----</td>
<td>---</td>
<td>------------------------------------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>------------------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>Clear Polycarbonate Sheet, 1/8&quot; Thick, 24&quot;x24&quot;</td>
<td>McMaster-Carr</td>
<td>8574K53</td>
<td>Emerson Netwrk Pwr</td>
<td>$19.41</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>Cat 5e cable, yellow</td>
<td>Allied</td>
<td>S10-3512</td>
<td></td>
<td>$1.54</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>12 gauge wire</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>18 gauge wire</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>26 gauge</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1</td>
<td>19&quot; Rack Mount, 132.5mmx300mm</td>
<td>Jameco</td>
<td>132848</td>
<td>Various</td>
<td>$59.95</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>4&quot; Handle</td>
<td>Mouser</td>
<td>563-H-9111-B</td>
<td>Bud Industries</td>
<td>$6.40</td>
<td></td>
</tr>
</tbody>
</table>

**Total** $269.77

---

## RPI DAQ

<table>
<thead>
<tr>
<th>Part Desc.</th>
<th>Vendor Part No.</th>
<th>Manufacturer Part No</th>
<th>Supplier</th>
<th>Unit Cost</th>
<th>Qty</th>
<th>Min Qty</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC - IC MCU FLASH 24X16 40DIP TRANSUDER CURRENT OPEN LOOP 20A</td>
<td>PIC18F4525-I/P-ND</td>
<td>PIC18F4525-I/P</td>
<td>Mouser</td>
<td>$5.60</td>
<td>1</td>
<td>1</td>
<td>$5.60</td>
</tr>
<tr>
<td>IC SENSOR THERMAL 3.1V TO-92-3</td>
<td>398-1025-S-ND</td>
<td>MCP9701A-E/TO-ND</td>
<td>Digi-key</td>
<td>$0.34</td>
<td>2</td>
<td>2</td>
<td>$0.68</td>
</tr>
<tr>
<td>TXRX ISO BUS 1MBS RS485 1650IC</td>
<td>390-1097-S-ND</td>
<td>IL3285E</td>
<td>Digi-key</td>
<td>$4.11</td>
<td>1</td>
<td>1</td>
<td>$4.11</td>
</tr>
<tr>
<td>IC SOCKET STRAIGHT 40POS TIN</td>
<td>AE10008-ND</td>
<td>A40-LC-TT-R</td>
<td>Digi-key</td>
<td>$1.03</td>
<td>1</td>
<td>1</td>
<td>$1.03</td>
</tr>
<tr>
<td>CONV DC/DC 10W SNGL 12V 5V PCB</td>
<td>445-2435-ND</td>
<td>CC10-12055F-E</td>
<td>Digi-key</td>
<td>$26.66</td>
<td>1</td>
<td>1</td>
<td>$26.66</td>
</tr>
<tr>
<td>IC COMPARATOR QD -40-85 C 14-DIP</td>
<td>LM2901NS-ND</td>
<td>LM2901N</td>
<td>Mouser</td>
<td>$0.36</td>
<td>1</td>
<td>1</td>
<td>$0.36</td>
</tr>
<tr>
<td>IGBT W/DIOE 600V 28A TO247AC</td>
<td></td>
<td>IRG4PC30KDPBF-ND</td>
<td>Digi-key</td>
<td>$4.75</td>
<td>1</td>
<td>1</td>
<td>$4.75</td>
</tr>
<tr>
<td>DIOE ZENER 24V 1W 5% DO-41</td>
<td>1N4749ADICT-ND</td>
<td>1N4749A-T</td>
<td>Mouser</td>
<td>$0.36</td>
<td>1</td>
<td>1</td>
<td>$0.36</td>
</tr>
<tr>
<td>Terminal Block - 5.08mm x 4</td>
<td>A97986-ND</td>
<td>1776119</td>
<td>Digi-key</td>
<td>$0.94</td>
<td>2</td>
<td>2</td>
<td>$1.88</td>
</tr>
<tr>
<td>Terminal Block - 10.16mm x 4</td>
<td>A98086-ND</td>
<td>282843-4</td>
<td>Digi-key</td>
<td>$1.25</td>
<td>2</td>
<td>2</td>
<td>$2.50</td>
</tr>
<tr>
<td>Terminal Block - 2.54mm x 12</td>
<td>A98075-ND</td>
<td>1-282834-2</td>
<td>Digi-key</td>
<td>$3.90</td>
<td>2</td>
<td>2</td>
<td>$7.80</td>
</tr>
<tr>
<td>CAP 4700PF 50V CERAMIC X7R 0805</td>
<td>399-1155-1-ND</td>
<td>C0805C472KSRACTU</td>
<td>Mouser</td>
<td>$0.05</td>
<td>1</td>
<td>1</td>
<td>$0.05</td>
</tr>
<tr>
<td>CAP 10UF 10V CERAMIC X5R 0805</td>
<td>587-1300-1-ND</td>
<td>LMK212BJ106KG-T</td>
<td>Digi-key</td>
<td>$2.75</td>
<td>4</td>
<td>10</td>
<td>$27.50</td>
</tr>
<tr>
<td>CAP 47000PF 50V CERAMIC X7R 0805</td>
<td>399-1166-1-ND</td>
<td>C0805C473KSRACTU</td>
<td>Digi-key</td>
<td>$1.06</td>
<td>18</td>
<td>20</td>
<td>$1.06</td>
</tr>
<tr>
<td>CAP CER 16PF 100V 5% C0G 0603</td>
<td>490-1332-1-ND</td>
<td>GRM1885CA160J010D</td>
<td>Digi-key</td>
<td>$0.64</td>
<td>3</td>
<td>10</td>
<td>$6.40</td>
</tr>
<tr>
<td>RES 4.99M OHM METAL FILM .50W 1%</td>
<td>PCHF4.99MCT-ND</td>
<td>HVR3700004994FR500</td>
<td>Digi-key</td>
<td>$3.99</td>
<td>2</td>
<td>5</td>
<td>$19.95</td>
</tr>
<tr>
<td>RES 49.9K OHM 1/4W 1% METAL FILM</td>
<td>49.9KXBK-ND</td>
<td>MFR-25FBF-49K9</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>2</td>
<td>5</td>
<td>$2.65</td>
</tr>
<tr>
<td>RES 1.87K OHM 1/4W 1% METAL FILM</td>
<td>1.87KXBK-ND</td>
<td>MFR-25FBF-1K87</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>1</td>
<td>5</td>
<td>$2.65</td>
</tr>
<tr>
<td>RES 196K OHM 1/4W 1% METAL FILM</td>
<td>196KXBK-ND</td>
<td>MFR-25FBF-196K</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>2</td>
<td>5</td>
<td>$2.65</td>
</tr>
<tr>
<td>RES 499 OHM 1/4W 1% METAL FILM</td>
<td>499XK8K-ND</td>
<td>MFR-25FBF-499R</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>1</td>
<td>5</td>
<td>$2.65</td>
</tr>
<tr>
<td>RES 4.99K OHM 1/4W 1% METAL FILM</td>
<td>4.99KXBK-ND</td>
<td>MFR-25FBF-4K99</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>1</td>
<td>5</td>
<td>$2.65</td>
</tr>
<tr>
<td>RES 1.00K OHM 1/4W 1% METAL FILM</td>
<td>1.00KXBK-ND</td>
<td>MFR-25FBF-1K00</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>1</td>
<td>5</td>
<td>$2.65</td>
</tr>
<tr>
<td>RES 100K OHM 1/4W 1% METAL FILM</td>
<td>100KXBK-ND</td>
<td>MFR-25FBF-100K</td>
<td>Digi-key</td>
<td>$0.49</td>
<td>1</td>
<td>5</td>
<td>$2.45</td>
</tr>
<tr>
<td>RES 121 OHM 1/4W 1% METAL FILM</td>
<td>121XK8K-ND</td>
<td>MFR-25FBF-121R</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>1</td>
<td>5</td>
<td>$2.65</td>
</tr>
</tbody>
</table>
ESS DAQ

<table>
<thead>
<tr>
<th>Part Desc.</th>
<th>Vendor Part No.</th>
<th>Supplier</th>
<th>Unit Cost</th>
<th>Qty</th>
<th>Min Qty</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES 562 OHM 1/4W 1% METAL FILM</td>
<td>562XBK-ND</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>2</td>
<td>5</td>
<td>$0.53</td>
</tr>
<tr>
<td>RES 750 OHM 1/4W 1% METAL FILM</td>
<td>750XBK-ND</td>
<td>Digi-key</td>
<td>$0.53</td>
<td>4</td>
<td>5</td>
<td>$0.53</td>
</tr>
<tr>
<td>LED DOME S83NM YW DFF AXL LO CUR</td>
<td>S16-1404-ND</td>
<td>Jameco</td>
<td>$0.45</td>
<td>4</td>
<td>4</td>
<td>$1.80</td>
</tr>
<tr>
<td>LED GREEN DIFFUSED ROUND LONG</td>
<td>P375-ND</td>
<td>Digi-key</td>
<td>$2.15</td>
<td>1</td>
<td>10</td>
<td>$2.15</td>
</tr>
<tr>
<td>LED AMBER DIFFUSED ROUND</td>
<td>P372-ND</td>
<td>Digi-key</td>
<td>$2.24</td>
<td>1</td>
<td>10</td>
<td>$2.24</td>
</tr>
<tr>
<td>PCB Board</td>
<td></td>
<td>PCB4.com</td>
<td>$33.00</td>
<td>1</td>
<td>1</td>
<td>$33.00</td>
</tr>
</tbody>
</table>

Total: $126.74
<table>
<thead>
<tr>
<th>Description</th>
<th>Catalog Number</th>
<th>Part Number</th>
<th>Supplier</th>
<th>Price</th>
<th>Quantity</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1UF smd cap</td>
<td>399-1284-1-ND</td>
<td>C0805C105K4ACTU</td>
<td>Mouser</td>
<td>$0.09</td>
<td>1</td>
<td>$0.09</td>
</tr>
<tr>
<td>16pF Cap</td>
<td>490-3630-ND</td>
<td>RPES5C2A160J2P1203B</td>
<td>Mouser</td>
<td>$0.23</td>
<td>3</td>
<td>$0.69</td>
</tr>
<tr>
<td>Fault Indicator LED</td>
<td>670-1192</td>
<td>SSL-LX3044SRD</td>
<td>Allied</td>
<td>$0.16</td>
<td>1</td>
<td>$0.16</td>
</tr>
<tr>
<td>Discharging Indicator LED</td>
<td>670-0169</td>
<td>SSL-LX3044USBD</td>
<td>Allied</td>
<td>$1.52</td>
<td>1</td>
<td>$1.52</td>
</tr>
<tr>
<td>Charging Indicator LED</td>
<td>670-1057</td>
<td>SSL-LX3044GD</td>
<td>Allied</td>
<td>$0.10</td>
<td>1</td>
<td>$0.10</td>
</tr>
<tr>
<td>30V Indicator LED</td>
<td>670-1059</td>
<td>SSL-LX3044YD</td>
<td>Allied</td>
<td>$0.12</td>
<td>1</td>
<td>$0.12</td>
</tr>
<tr>
<td>MOSFET N-CH 60V 500MA TO-92</td>
<td>BS170-ND</td>
<td>bs170</td>
<td>Mouser</td>
<td>$0.23</td>
<td>1</td>
<td>$0.23</td>
</tr>
<tr>
<td>optoisolator - 6n135</td>
<td>6N135QT-ND</td>
<td>6n135</td>
<td>Mouser</td>
<td>$0.73</td>
<td>1</td>
<td>$0.73</td>
</tr>
<tr>
<td>EDS protection</td>
<td>296-6596-S-ND</td>
<td>SN75240P</td>
<td>Mouser</td>
<td>$1.15</td>
<td>1</td>
<td>$1.15</td>
</tr>
<tr>
<td>220uF Al Electrolytic Cap</td>
<td>P5183-ND</td>
<td>ECA-1HM221</td>
<td>Digi-Key</td>
<td>$0.37</td>
<td>1</td>
<td>$0.37</td>
</tr>
<tr>
<td>47uF Al Electrolytic Cap</td>
<td>P834-ND</td>
<td>ECE-A1EKA470</td>
<td>Digi-Key</td>
<td>$0.15</td>
<td>1</td>
<td>$0.15</td>
</tr>
<tr>
<td>10uF Al Electrolytic Cap</td>
<td>P813-ND</td>
<td>ECE-A1EKA100</td>
<td>Digi-Key</td>
<td>$0.14</td>
<td>1</td>
<td>$0.14</td>
</tr>
<tr>
<td>1uF Al Electrolytic Cap</td>
<td>P824-ND</td>
<td>ECE-A1HKA010</td>
<td>Digi-Key</td>
<td>$0.14</td>
<td>1</td>
<td>$0.14</td>
</tr>
<tr>
<td>3.5A fuse Size:2AG</td>
<td>F2463-ND</td>
<td>022503.5HXP</td>
<td>Mouser</td>
<td>$0.37</td>
<td>1</td>
<td>$0.37</td>
</tr>
<tr>
<td>Size:2AG fuse clips</td>
<td>F058-ND</td>
<td>01110501H</td>
<td>Mouser</td>
<td>$0.15</td>
<td>2</td>
<td>$0.30</td>
</tr>
<tr>
<td>high voltage to 5V dc-dc</td>
<td>835-1032-ND</td>
<td>M57184N-715B</td>
<td>Digi-Key</td>
<td>$7.81</td>
<td>1</td>
<td>$7.81</td>
</tr>
<tr>
<td>12V-5V dc-dc 25W</td>
<td>555-1058-ND</td>
<td>AXA005A0XZ</td>
<td>Digi-Key</td>
<td>$12.10</td>
<td>1</td>
<td>$12.10</td>
</tr>
<tr>
<td>PCB Board</td>
<td></td>
<td></td>
<td>PCB4.com</td>
<td>$33.00</td>
<td>1</td>
<td>$33.00</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>$143.29</strong></td>
</tr>
</tbody>
</table>
**Data Acquisition Protocol**

Set up the Display and Port tabs respectively in RealTerm

![DAQ Communication Port Parameters][1]

Send commands in decimal, will receive them in hexadecimal

Order of sent and received data for commands, the checksum is calculated by xoring all of the previous bytes sent

**Get Analog Value:**

Destination/source/number of bytes sent/0/channel/checksum

Destination/source/number of bytes sent/Error Type/MSB/LSB/checksum

**Get Digital Value:**

Destination/source/number of bytes sent/1/pin/checksum

Destination/source/number of bytes sent/Error Type/Value/Byte Stuff/checksum

**Set Pin High:**

Destination/source/number of bytes sent/2/pin/checksum

Destination/source/number of bytes sent/Error Type/Byte Stuff/Byte Stuff/checksum

---

[1]: Figure 87: DAQ Communication Port Parameters
**Set Pin Low:**

Destination/source/number of bytes sent/3/pin/checksum

Destination/source/number of bytes sent/Error Type/Byte Stuff/Byte Stuff/checksum

List of Port Configuration for the PIC, there are only channel values for parts that go through the A-D Converter:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin Destination</th>
<th>A-D Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5V High through R41</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Temp1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Temp2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Temp3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Temp4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Discharges (LED D2)</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Vbatt4</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Vbatt3</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>Vbatt2</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Vbatt1</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>5VH Power</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>HVGround</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>Fault(Red LED)</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>HV Present(Yellow LED)</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>Dig out 5(small LED D6)</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>Dig out 6(small LED D7)</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>Dig out 7(small LED D8)</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>Dig out 8(small LED D9)</td>
<td>17</td>
</tr>
<tr>
<td>19</td>
<td>Dig out 9(small LED D10)</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>Dig out 10(small LED D11)</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>Dig in 6</td>
<td>20</td>
</tr>
<tr>
<td>22</td>
<td>Dig in 5</td>
<td>21</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>25</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>Dig in 4</td>
<td>27</td>
</tr>
</tbody>
</table>
## RPI Data Acquisition Board Sensor Equations

Board # 3

**List of Port Configuration for the PIC, there are only channel values for parts that go through the A-D Converter:**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin Destination</th>
<th>A-D Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Tsense_out2</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>Vsense_out1</td>
<td>10</td>
</tr>
<tr>
<td>35</td>
<td>Vsense_out2</td>
<td>8</td>
</tr>
<tr>
<td>36</td>
<td>Isense_Vout</td>
<td>9</td>
</tr>
<tr>
<td>37</td>
<td>Tsense_out1</td>
<td>11</td>
</tr>
</tbody>
</table>

**Commands for the Board:**

Get Analog Value:
Destination/source/number of bytes sent/0/channel/checksum
Destination/source/number of bytes sent/Error Type/MSB/LSB/checksum

**Conversion equations for the sensors:**

Voltage Sensors:
Convert the Hexadecimal value from the terminal to a decimal value.
\[ x = \text{decimal value coming from the terminal} \]
.0049x + .0103 = voltage before it enters the voltage divider
.5x + .5 = voltage at the output

Temperature Sensors:
Convert the Hexadecimal value from the terminal to a decimal value.
x = decimal value coming from the terminal
.247519x − 19.9367 = Degrees Celsius

Current Sensor:
Convert the hexadecimal value from the terminal to a decimal value.
X = decimal value coming from the terminal
.0396x − 19.944 = current
ESS Data Acquisition Board Sensor Equations

Board #2

List of Port Configuration for the PIC, there are only channel values for parts that go through the A-D Converter:

<table>
<thead>
<tr>
<th>Pin</th>
<th>Pin Destination</th>
<th>A-D Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5V High through R41</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Temp1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Temp2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Temp3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Temp4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>Discharges (LED D2)</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>Vbatt4</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>Vbatt3</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>Vbatt2</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Vbatt1</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>Fault(LED D3)</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>HV Present(LED D4)</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>Dig out 5(small LED D6)</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>Dig out 6(small LED D7)</td>
<td>13</td>
</tr>
<tr>
<td>17</td>
<td>Dig out 7(small LED D8)</td>
<td>14</td>
</tr>
<tr>
<td>18</td>
<td>Dig out 8(small LED D9)</td>
<td>15</td>
</tr>
<tr>
<td>19</td>
<td>Dig out 9(small LED D10)</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>Dig out 10(small LED D11)</td>
<td>17</td>
</tr>
<tr>
<td>21</td>
<td>Dig in 6</td>
<td>18</td>
</tr>
<tr>
<td>22</td>
<td>Dig in 5</td>
<td>19</td>
</tr>
<tr>
<td>27</td>
<td>Dig in 4</td>
<td>20</td>
</tr>
<tr>
<td>28</td>
<td>Dig in 3</td>
<td>21</td>
</tr>
<tr>
<td>29</td>
<td>Dig in 2</td>
<td>22</td>
</tr>
<tr>
<td>30</td>
<td>Dig in 1</td>
<td>23</td>
</tr>
<tr>
<td>33</td>
<td>LED 14</td>
<td>24</td>
</tr>
<tr>
<td>34</td>
<td>Analog 11</td>
<td>25</td>
</tr>
<tr>
<td>35</td>
<td>Analog 10</td>
<td>26</td>
</tr>
</tbody>
</table>
### Commands for the Board:

**Get Analog Value:**
Destination/source/number of bytes sent/0/channel/checksum
Destination/source/number of bytes sent/Error Type/MSB/LSB/checksum

**Get Digital Value:**
Destination/source/number of bytes sent/1/pin/checksum
Destination/source/number of bytes sent/Error Type/Value/Byte Stuff/checksum

**Set Pin High:**
Destination/source/number of bytes sent/2/pin/checksum
Destination/source/number of bytes sent/Error Type/Byte Stuff/Byte Stuff/checksum

**Set Pin Low:**
Destination/source/number of bytes sent/3/pin/checksum
Destination/source/number of bytes sent/Error Type/Byte Stuff/Byte Stuff/checksum

### Conversion equations for the sensors:

**Voltage Sensors:**
Convert the Hexadecimal value from the terminal to a decimal value.

\[ x = \text{decimal value coming from the terminal} \]
\[ .0049x + .0103 = \text{voltage before it enters the voltage divider} \]
\[ .5x + .5 = \text{voltage at the output} \]

**Temperature Sensors:**
Convert the Hexadecimal value from the terminal to a decimal value.

\[ x = \text{decimal value coming from the terminal} \]
\[ .0247519x – 19.9367 = \text{Degrees Celsius} \]

**Current Sensor:**
Convert the hexadecimal value from the terminal to a decimal value.

\[ X = \text{decimal value coming from the terminal} \]
\[ .0309x – 15.674 = \text{current} \]
Sunny Boy Communication Protocol

System Specifications

The Sunny Boy model that is installed in the project room is a model 3000US. It is equipped with an RS-485 communication module that allows communication through the SMA data protocol. This protocol is implemented using the YASDI (yet another SMA data implementation) API that was provided by the Sunny Boy manufacturer. This code is located in the Sunny Boy directory on the fit pc.

Preliminary Requirements

Before communication with the Sunny Boy is possible, the RS-485 wires that are coming from the Sunny Boy must be connected using an RS-485/422 to USB convertor which must be connected to the fit pc. The connected device’s location must be tty/USB1, this can be verified by typing the following command into the terminal on the fit pc:

>tail var/log/messages

This command will display where the most recent connected USB devices have been allocated to. If it does not match tty/USB1 then the yasdi.ini file must be modified to match the allocated USB number.

How to Communicate with the Sunny Boy

First, cd into the Sunny Boy Code directory located on the Desktop of the fit pc. Then cd into the Communication_Code directory, then into the most recent version available (currently version 1). Once there run the executable sunnyBoy to start the communication process.

Errors in Communication

If no Sunny Boy can be detected, there will be a message on the terminal that will explicitly say that the program cannot find the device. If it is found it will display that the device has been found.

Communication Commands

Once the initialization is complete the program will prompt the user for a channel name, a list of known channel names and its return value are:

Error – Gives a number that corresponds to a specific type of error which can be found in the documentation. A zero means there is no error.

E-Total – Returns the total energy in kWh delivered to the grid.

Fac – Returns the frequency of the AC voltage delivered to the grid

Iac- Returns the amount of current in Amps delivered to the grid
Ipv – Returns the amount of Amps being drawn from the PV array
Mode – Returns a number that corresponds to the mode of the system. This list is in the Sunny Boy documentation
Pac – Returns the amount of power delivered to the grid in kW
Temperature – Returns the temperature in degrees Celsius
Vac – Voltage in Volts delivered to the grid
Vpv – Voltage in Volts currently being drawn from the PV array

This is a list of all currently known command for communication with the Sunny Boy. As more are found, they will be added to the list.
**LPRDS-BMS Tower Layout**

The LPRDS tower was arranged in a way to optimize the display available from the window of AEC401. The SC and FIB boxes were mounted vertically with clear plexiglass covers to make the inner circuitry visible with vinyl lettering applied over it to point out key components. The ESS has 4 new LEDs to indicate active, charging, discharging and fault states. The Output box has analog meters to display the voltage and frequency of the generated AC signal as well as the picoLCD for displaying the current state of the system. A LPRDS sign was machined with the help of the mechanical engineering department, particularly professor Helm and senior Tim Hatch.

The back of the rack provides access to the fitPC keyboard and mouse, SIB, alarm interface board and mains power strip. An LCD monitor was placed at the top of the rack for use with the fitPC and to display the demo app. This is not meant to be a permanent installation, as it does not meet the energy independence requirement. Future project teams should invest in a low power (possibly LED backlit) LCD display that can be used with the system 12V supply within the system’s power budget.

![Tower Layout Picture](image-url)
Figure 89: Tower Layout
APPENDICES

Matlab simulation files:

generateSinTable.m
This script will generate a text file with a sine table to be used by the LPRDS inverter PWM generator. The table is generated in hex and can be put directly into the microcontroller program without modification. The table generated is a 10bit table, as the code is written.

```matlab
%% Generate Table
bits = 10;
modulation = 120*1.414/205;
maximum = 3*16*16+2*16+0;

ph = 0:2*3.14/2^bits:2*3.14;
ddswave = modulation*sin(ph);
ddswave = ddswave*maximum/2+maximum/2;
min(ddswave);
max(ddswave);
I = uint16(ddswave);
sinTable = dec2hex(I);

%% Write To File
fileID = fopen('sinTable.txt','w');

for a=1:length(sinTable)
    fprintf(fileID,'%e','0');
    fprintf(fileID,'%e','x');
    for b=1:3
        fprintf(fileID,'%e',sinTable(a,b));
    end

if mod(a,13) == 0
```
fprintf(fileID,'%c\n',');
else
    fprintf(fileID,'%c',');
end
end

close(fileID);

**THD.m**

The THD is calculated of the input waveform and output waveform of the filter with no load. The two files that are loaded are of the same simulation just different in time. This was done to make sure that the THD stayed constant even with a change in time.

```
input60Hz = load('input60Hz.dat');
subplot(2,1,1),plot(input60Hz(:,1),input60Hz(:,2));
title('Input Sine Wave');
t = input60Hz(:,1) + abs(min(input60Hz(:,1)));
amp = input60Hz(:,2);
N = length(input60Hz(:,1));
T = max(t);

wamp = amp.*hamming(N);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(wamp))/(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
```
subplot(2,1,2),plot(freq,p);
title('Power Spectrum of Input Sine Wave');

axis([0 1200 -80 20]);

% Get powers of 60Hz
% freq - frequencies used to index p
% p - fft
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59 || abs(mod(freq(a),60)) < 1)
        if (freq(a) < 61 && freq(a) > 59)
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
THDpercent = Et/(Et+Ef)*100

% output
figure
output = load('output60Hz.dat');
subplot(2,1,1),plot(output(:,1),output(:,2));
title('Output Sine Wave');
t = output(:,1) + abs(min(output(:,1))); 
amp = output(:,2);
N = length(output(:,1));
T = max(t);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(amp))/(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
subplot(2,1,2),plot(freq,p);
title('Power Spectrum of Output Sine Wave');

axis([0 1200 -80 10]);

% Get powers of 60Hz
%freq - frequencies used to index p
%p - fft
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59 || abs(mod(freq(a),60)) < 1)
        if (freq(a) < 61 && freq(a) > 59)
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
end
THDvalue = Et/(Et+Ef)

%%

%input
input = load('input60Hz2.dat');
subplot(2,1,1),plot(input60Hz(:,1),input60Hz(:,2));
title('Input Sine Wave');
t = input60Hz(:,1) + abs(min(input60Hz(:,1)));
amp = input60Hz(:,2);

N = length(input60Hz(:,1));
T = max(t);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(amp))/(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
subplot(2,1,2),plot(freq,p);
title('Power Spectrum of Input Sine Wave');
axis([0 1200 -80 10]);

% Get powers of 60Hz
%freq - frequencies used to index p
%p - fft
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59 || abs(mod(freq(a),60)) < 1)
        if (freq(a) < 61 && freq(a) > 59)
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
THDvalue = Et/(Et+Ef)

%%

%output
figure
output = load('output60Hz2.dat');
subplot(2,1,1),plot(output(:,1),output(:,2));
title('Output Sine Wave');
t = output(:,1) + abs(min(output(:,1)));
amp = output(:,2);
N = length(output(:,1));
T = max(t);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(amp))(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
subplot(2,1,2),plot(freq,p);
title('Power Spectrum of Output Sine Wave');

axis([0 1200 -80 10]);

% Get powers of 60Hz
%freq - frequencies used to index p
%p - fti
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59 || abs(mod(freq(a),60)) < 1)
        if (freq(a) < 61 && freq(a) > 59)
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
THDvalue = Et/(Et+Ef)

THD205VDC.m
THD calculation of the 205VDC input with a 300W load
clear all;
input60Hz = load('C1FIB_205vdc00001.dat');
t = input60Hz(:,1) - min(input60Hz(:,1));
subplot(2,1,1),plot(t,input60Hz(:,2));
title('Input Sine Wave');
amp = input60Hz(:,2);

N = length(input60Hz(:,1));
T = max(t);
wamp = amp.*hanning(N);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(wamp))/(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
subplot(2,1,2),plot(freq,p);
%axis([0 1200 -60 40]);

%Calculating THD
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if ((abs(mod(freq(a),60)) < .01) || (abs(mod(freq(a),60)) >59))
        if (Ef == 0)
\[ Ef = 10^{(p(a)/10)}; \]
\[ \text{else} \]
\[ Et = Et + 10^{(p(a)/10)}; \]
\[ \text{end} \]
\[ \text{end} \]
\[ \text{end} \]

\[ \text{THDpercent} = \text{Et/(Et+Ef})*100 \]

\text{title(}'\text{Power Spectrum of Input Sine Wave with THD = ',num2str(THDpercent), '}'\text{);} \]

\% Checking to see if the what's the value in dBV in the 150 KHz range to see
\% if the FIB passed the conduction/emmission requirement
\text{clear all}
\text{input60Hz = load('C1FIB_205vdc00001.dat');}
\text{t = input60Hz(100000:100112,1) - min(input60Hz(100000:100112,1));}
\text{subplot(2,1,1),plot(t,input60Hz(100000:100112,2));}
\text{title('Input Sine Wave');}
\text{amp = input60Hz(100000:100112,2);}

\text{N = length(input60Hz(100000:100112,1));}
\text{T = max(t);}
\text{wamp = amp.*hanning(N);}
\text{% subplot(2,1,1),plot(t,amp);}
\text{% subplot(2,1,2),plot(t,wamp);}

\text{dt = T/N;}
\text{fs = 1/dt;}
\text{Fmax = fs/2;}
\text{dF = fs/N;}

\text{p = abs(fftn(wamp))/(N/2);}
\text{p = 10*log10(p(1:N/2).^2);}
\text{% pwatts = (p(1:N/2).^2);}
\text{freq = [0:N/2-1]/T;}
**THDFilterAndWallOutlet.m**

Calculating the THD of the wall outlet

```matlab
input60Hz = load('wallOutput.dat');
%input60Hz = wallInput;
subplot(2,1,1),plot(input60Hz(:,1),input60Hz(:,2));
title('Wall Output Sine Wave');
t = input60Hz(:,1) + abs(min(input60Hz(:,1)));
amp = input60Hz(:,2);

N = length(input60Hz(:,1));
T = max(t);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(amp))/(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
subplot(2,1,2),plot(freq,p);
axis([0 1200 -80 30]);

% Get powers of 60Hz
%freq - frequencies used to index p
%p - fft
```
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59 || abs(mod(freq(a),60)) < 1)
        if (freq(a) < 61 && freq(a) > 59)
            a
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
THDPercent = Et/(Et+Ef)*100

title(['Power Spectrum of Wall Output Sine Wave With THD = ', num2str(THDPercent)]);

%%

% Calculating the THD of the filter with the wall outlet as the input.
input60Hz = load('wallfiltered.dat');
%input60Hz = wallInput;
subplot(2,1,1),plot(input60Hz(:,1),input60Hz(:,2));
title('Filter Output With The Wall As The Input Sine Wave');
t = input60Hz(:,1) + abs(min(input60Hz(:,1)));
amp = input60Hz(:,2);
N = length(input60Hz(:,1));
T = max(t);
dt = T/N;
fs = 1/dt;
\[ F_{\text{max}} = \frac{f_s}{2}; \]
\[ dF = \frac{f_s}{N}; \]
\[ p = \text{abs}(\text{fft}(\text{amp}))(N/2); \]
\[ p = 10 \times \log_{10}(p(1:N/2)^2); \]
\[ \% \text{ pwatts} = (p(1:N/2)^2); \]
\[ \text{freq} = [0:N/2-1]/T; \]
\[ \text{subplot}(2,1,2), \text{plot}(\text{freq},p); \]
\[ \text{title}('[\text{Power Spectrum of Filter Output With The Wall As The Input Sine Wave With THD} = ', \text{num2str}(\text{THDPercent}))]; \]

\[ \text{axis}([0 1200 -80 30]); \]

\[ \% \text{ Get powers of 60Hz} \]
\[ \% \text{freq} - \text{frequencies used to index } p \]
\[ \text{Et} = 0; \]
\[ \text{Ef} = 0; \]
\[ \text{first} = 1; \]
\[ b = 0; \]
\[ \text{for } a=2:1: \text{length(freq)} \]
\[ \quad \text{if } (\text{abs(mod(freq(a),60))} > 59 \| \text{abs(mod(freq(a),60))} < 1) \]
\[ \quad \quad \text{if } (\text{freq(a)} < 61 \&\& \text{freq(a)} > 59) \]
\[ \quad \quad \quad \text{Ef} = 10^{\times}(p(a)/10); \]
\[ \quad \quad \text{else} \]
\[ \quad \quad \quad \text{Et} = \text{Et} + 10^{\times}(p(a)/10); \]
\[ \quad \text{end} \]
\[ \text{end} \]
\[ \text{THDPercent} = \frac{\text{Et}}{\text{Et}+\text{Ef}} \times 100 \]

**THD-getPeaks.m**

This file gets the powers of 60 Hz from the power spectrum of a sine wave.
%% Get powers of 60Hz
%freq - frequencies used to index p
%p - fft
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59 || abs(mod(freq(a),60)) < 1)
        if (freq(a) < 61 && freq(a) > 59)
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
THDvalue = Et/(Et+Ef)

THDSimulation.m
This program calculates the THD of the simulated waveform.
clear all
input60Hz = load('SimulatedOutput.mat');
input60Hz.ans = input60Hz.ans';
title('Simulation Output Sine Wave');
t = input60Hz.ans(:,1) + abs(min(input60Hz.ans(:,1)));
samples = 112 + 100000;
t = t(100000:samples);
amp = input60Hz.ans(100000:samples,2);
subplot(2,1,1),plot(t,amp);

N = length(input60Hz.ans(100000:samples,1));
T = max(t) - min(t);
wamp = amp.*hamming(N);

dt = T/N;
fs = 1/dt;
Fmax = fs/2;
dF = fs/N;

p = abs(fft(wamp))/(N/2);
p = 10*log10(p(1:N/2).^2);
% pwatts = (p(1:N/2).^2);
freq = [0:N/2-1]/T;
subplot(2,1,2),plot(freq,p);
title('Power Spectrum of Simulation Output Sine Wave');
axis([140000 160000 -200 50]);

% Get powers of 60Hz
%freq - frequencies used to index p
%p - fft
Et = 0;
Ef = 0;
first = 1;
b=0;
for a=2:1:length(freq)
    if (abs(mod(freq(a),60)) > 59.9 || abs(mod(freq(a),60)) < 0.1)
        if (Ef == 0)
            Ef = 10^(p(a)/10);
        else
            Et = Et + 10^(p(a)/10);
        end
    end
end
THDpercent = \frac{E_t}{E_t + E_f} \times 100

Matlab Simulink simulation files:

**discretePWM.mdl**

This simulation model shows how a PWM can be generated in simulink using the PWM Generator in the SimPowerSystems package.

**FilterModel.mdl**

This is a simulation of the filter and transformer with a 'BadHysteresis2.mat' as the hysteresis model for the inductors. This was made in order to study whether or not a bad inductor material changes the outcome of the transformer. Other mat files include 'BadHysteresis.mat' which models another material that an inductor can be made out of.
**InverterModelDiscrete.mdl**
This is a simulation of the inverter, filter and transformer using simulink. This simulation has scopes on the output of the battery, the IGBTs and the output.

**InverterModelDiscreteLite.mdl**
This is a simulation of the inverter, filter and transformer using simulink. The only difference between this simulation and the previous simulation is that a lot of the scopes and irrelevant elements were taken out due to that small amount of memory. This way the output of the wave from was saved into a .mat file.
**InverterModelDiscreteLite_controlled.mdl**

This is a simulation of the inverter, filter and transformer using simulink. This simulation was made in order to observe different outputs. This simulation has scopes on the ideal switch, the voltage across the Cdc, and the voltage at the output.

**InverterModelDiscreteLite_snub.mdl**

This is a simulation of the inverter, filter and transformer using simulink. A lot of the scopes and irrelevant elements were taken out due to that small amount of memory. A snubber was added at the input of the H-Bridge.
InverterModelDiscreteLite_snubRCD.mdl

This is a simulation of the inverter, filter and transformer using simulink. A lot of the scopes and irrelevant elements were taken out due to that small amount of memory. A snubber was added at the input of the H-Bridge as well as an RLC circuit across two ideal switches.
**Smartspice simulation files:**

**Filter.cir**
Simulation of the filter and transformer with a resistive load called Rload.

VSIG vsrc vsrc2 AC SIN(0 1 60Hz 0sec 0 0)
*Vreson xtra vsig2 AC SIN(0 450Hz 0sec 0 0)

*source resistance of signal generator
*Rsrc vsrc vsig1 50

*R1ohm vsig1 vsig2 1

R1top vsrc ltop 60m
L1top ltop captop 500u
R1bot vsrc2 lbot 60m
L1bot lbot capbot 500u
C1top captop capbot 30uF

R2topl captop l2top 0
R2botl capbot l2bot 0
L2 l2top l2bot 35mH
R2topr l2top rtop 0
R2botr l2bot rbot 0
Rload rtop rbot 360
.AC DEC 1000 10 100K
FullInv.cir
Simulation of the IGBTs and H-Bridge

.INCLUDE irg4pc30kd.spi
.INCLUDE IR2184-test.sub
.MODEL Da1N4004 D(IS=10N N=1 BV=1200 IBV=10E-15 VJ=0.1)
.MODEL 1N4148 D(IS = 4.352E-9 N = 1.906 BV = 110 IBV = 0.0001 RS = 0.6458 CJO =
7.048E-13 VJ = 0.869 M = 0.03 FC = 0.5 TT = 3.48E-9)

Vpulse1 IN1 0 AC PULSE(0 10 0 10u 10u .001 .0025)
Vpulse2 IN2 0 AC PULSE(0 10 .00125 10u 10u .001 .0025)

*Transistors in H-bridge
XIGBTTop1 ncollec HO11 VS1 irg4pc30kd
XIGBTBot1 VS1 LO11 0 irg4pc30kd

XIGBTTop2 ncollec HO22 VS2 irg4pc30kd
XIGBTBot2 VS2 LO22 0 irg4pc30kd

*Drivers and needed components
C5 VS1 VB1 10u IC=15
C6 VS2 VB2 10u IC=15

Rsd1 VCC SD1 1k
Rsd2 VCC SD2 1k

Xdriver1 VCC IN1 SD1 0 VB1 HO1 VS1 LO1 IR2184
Xdriver2 VCC IN2 SD2 0 VB2 HO2 VS2 LO2 IR2184

R6 HO1 HO11 10
R5 LO1 LO11 10
R4 HO2 HO22 10
R3 LO2 LO22 10

D5 VCC VB1 Da1N4004
*D6 VCC VB2 Da1N4004

*PWM.CIR - PULSE WIDTH MODULATION
*
* INPUT VOLTAGE
VIN 1 0 SIN(5V 4V 60HZ) ;SIN(VOffset VPeak Frequency)
VIN2 13 0 SIN(5V 4V 60HZ 0 0 180) ;SIN(VOffset VPeak Frequency)
RIN 1 0 1K
RIN2 13 0 1K
*
* 10KHZ TRIANGLE WAVE
* (GENERATED USING PULSE SOURCE WITH LONG RISE/FALL TIMES)
VTRI 2 0 PULSE(0V 10V 0 49US 49US 1US 100US)
RTRI 2 0 1MEG
*
* COMPARATOR, INPUT = V(1,2)
* FOR V(1,2) < -1MV, OUTPUT = 0V
* FOR V(1,2) > +1MV, OUTPUT = 10V
ECOMP 3 0 TABLE {V(1,2)} = (-1MV 0V) (1MV, 10V)
RCOMP 3 0 1MEG
ECOMP2 14 0 TABLE {V(13,2)} = (-1MV 0V) (1MV, 10V)
RCOMP2 14 0 1MEG
*
* PWM OUTPUT STAGE
VCCpwm 10 0 DC 10V
Q1 10 3 IN1 QNOM
RL1 IN1 0 20
*VCCpwm2 15 0 DC 10V
*Q2 15 14 IN2 QNOM
*RL2 IN2 0 20
.model QNOM NPN
*Load
Lload VS1 VS2 35m
*Rload VS1 VS2 100
VCC ncollec 0 Pulse(0 200 1m .1u .1u 1 2)
VCC2 VCC 0 15V
.TRAN .001 .5 UIC; 0 .000001s UIC
*.Options ITL1=10000 ITL2=10000 ITL4=10000 Method=trap
.probe
.END

HBridgeStep.cir
Simulation for the IGBT and H-Bridge

.INCLUDE irg4pc30kd.spi
.INCLUDE IR2184-test.sub
.MODEL Da1N4004 D(IS=10N N=1 BV=1200 IBV=10E-15 VJ=0.1)
.MODEL 1N4148 D(IS = 4.352E-9 N = 1.906 BV = 110 IBV = 0.0001 RS = 0.6458 CJO = 7.048E-13 VJ = 0.869 M = 0.03 FC = 0.5 TT = 3.48E-9)

Vpulse1 IN1 0 AC PULSE(0 10 0 10u 10u .001 .0025)
Vpulse2 IN2 0 AC PULSE(0 10 .00125 10u 10u .001 .0025)

*Transistors in H-bridge
XIGBTTop1 ncollec HO11 VS1 irg4pc30kd
XIGBTBot1 VS1 LO11 0 irg4pc30kd
XIGBTTop2 ncollec HO22 VS2 irg4pc30kd
XIGBTBot2 VS2 LO22 0 irg4pc30kd

*Drivers and needed components
C5 VS1 VB1 10u IC=15
C6 VS2 VB2 10u IC=15
Rsd1 VCC SD1 1k
Rsd2 VCC SD2 1k
Xdriver1 VCC IN1 SD1 0 VB1 HO1 VS1 LO1 IR2184
Xdriver2 VCC IN2 SD2 0 VB2 HO2 VS2 LO2 IR2184
R6 HO1 HO11 10
R5 LO1 LO11 10
R4 HO2 HO22 10
R3 LO2 LO22 10
D5 VCC VB1 Da1N4004
*D6 VCC VB2 Da1N4004

*PWM.CIR - PULSE WIDTH MODULATION
* INPUT VOLTAGE
VIN 1 0 SIN(5V 4V 60HZ) ;SIN(VOffset VPeak Frequency)
VIN2 13 0 SIN(5V 4V 60HZ 0 0 180) ;SIN(VOffset VPeak Frequency)
RIN 1 0 1K
RIN2 13 0 1K
*
* 10KHZ TRIANGLE WAVE
* (GENERATED USING PULSE SOURCE WITH LONG RISE/FALL TIMES)
VTRI 2 0 PULSE(0V 10V 0 49US 49US 1US 100US)
RTRI 2 0 1MEG
*
* COMPARATOR, INPUT = V(1,2)
* FOR V(1,2) < -1MV, OUTPUT = 0V
* FOR V(1,2) > +1MV, OUTPUT = 10V
ECOMP 3 0 TABLE {V(1,2)} = (-1MV 0V) (1MV, 10V)
RCOMP 3 0 1MEG
Last Year’s ‘illities

Environmental

8.14 Memo 14: GPR002– Environmental Testing

To: Senior Management

From: LPRDS Team

Date: 04/24/09

We are required by GPR002 in the Lafayette Photovoltaic Research and Development (LPRDS) Statement of Work to demonstrate “reliable and normal functional operation” in ambient lab and storage temperatures and relative humidities. Specifically, our project must operate in temperatures ranging from 15 °C to 30 °C when in a lab environment, and 0 °C to 60 °C when in a storage environment. The relative humidity requirements state that we must be able to operate between 10% and 80%, non-condensing, when in a lab setting, and between 5% and 95%, non-condensing, when in a storage setting. Finally, we are required to use electronic components rated for commercial temperature ranges (0 °C to 70 °C).
In order to show that our electronic components are all rated for commercial temperature ranges, I have created a spreadsheet with each component listed along with its high and low temperature ratings. It can be seen on the following page. Note that all components fall within the needed operational temperature range, and therefore we can conclude that room temperature will have no effect on operation.

All cables and connector components are covered with weather resistant plastic. That is, they are designed to reduce the effects of humidity on the hardware. Because of this design, I posit that the wear-and-tear effects caused by a high humidity storage and lab environment will be negligible.

In the same manner, I can assure you that all subsystem containers, as well as the overall system container will also be resistant to the effects of high humidity. The team’s subsystem containers are made out of either steel or aluminum (RPI is steel, ESS and EDS are aluminum, SCADA has no enclosure). These materials are highly resistant to corrosion due to the oxide layer that forms on the surface. Therefore I conclude that there is no need to worry about damage to the containers.

Through this analysis, I have determined that our system specifications fall within normal environmental fluctuations and therefore we can expect the system to operate when within the above ranges.

**Sustainability and Ethics**

8.2 Memo 2: GPR009 – Sustainability

To: Senior Management
From: LPRDS Team
Date: 04/24/09

1. PV Sizing and Economic Analysis

The following analysis will estimate the size and cost of a PV system that would support the common household in Easton, PA. Geographic location obviously plays a critical role in the design and economics of solar panel systems. This is because the input energy from the sun, which is called the insolation, varies greatly with geographic region and season. Therefore the
first step in designing a PV solar system is to understand the insolation data for the installation site.

**Insolation**

Insolation is the measure of solar radiation energy received on a surface over a given time and has units of W/m$^2$. Insolation data for a given geometric region can be obtained using software such as Virtual Test Bed (VTB) [10] or from sources like the National Solar Radiation Database (NSRD) [11]. Figure 1, shows how the insolation varies throughout the day in our region of Pennsylvania on the first of March, July, September, and December. As shown in the figure, in the summer months the sun energy is present more hours of the day and it is also more intense. The daily insolation curves were integrated for 365 days to find an average of 3.86kWh/m$^2$/day for Easton, PA. This data will be important to size a solar panel system.

In sizing a system, it is also important to account for cloudy days. For this analysis it will be assumed that on cloudy days the solar panel system is shut down and the house is powered by the grid. An estimate for the average monthly insolation can be determined by derating the insolation based on the average number of cloudy days per month which can be obtained from the National Oceanic and Atmospheric Administration (NOAA) [12].

<table>
<thead>
<tr>
<th>Month</th>
<th>Average Monthly Insolation (kWh/m$^2$/day)</th>
<th>Non-Cloudy days form NOAA [12]</th>
<th>Total Insolation in a Month (kWh/m$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan</td>
<td>1.89</td>
<td>15</td>
<td>28.35</td>
</tr>
<tr>
<td>Feb</td>
<td>2.7</td>
<td>15</td>
<td>40.5</td>
</tr>
<tr>
<td>Mar</td>
<td>3.69</td>
<td>16</td>
<td>59.04</td>
</tr>
<tr>
<td>Month</td>
<td>Jan</td>
<td>Feb</td>
<td>Total</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
</tr>
<tr>
<td>Jan</td>
<td>4.71</td>
<td>16</td>
<td>75.36</td>
</tr>
<tr>
<td>Feb</td>
<td>5.44</td>
<td>17</td>
<td>92.48</td>
</tr>
<tr>
<td>Mar</td>
<td>5.96</td>
<td>19</td>
<td>113.24</td>
</tr>
<tr>
<td>Apr</td>
<td>5.87</td>
<td>20</td>
<td>117.4</td>
</tr>
<tr>
<td>May</td>
<td>5.33</td>
<td>20</td>
<td>106.6</td>
</tr>
<tr>
<td>Jun</td>
<td>4.19</td>
<td>18</td>
<td>75.42</td>
</tr>
<tr>
<td>Jul</td>
<td>3.06</td>
<td>19</td>
<td>58.14</td>
</tr>
<tr>
<td>Aug</td>
<td>1.95</td>
<td>15</td>
<td>29.25</td>
</tr>
<tr>
<td>Sept</td>
<td>1.57</td>
<td>14</td>
<td>21.98</td>
</tr>
<tr>
<td>Oct</td>
<td>1.57</td>
<td>14</td>
<td>21.98</td>
</tr>
<tr>
<td>Nov</td>
<td>1.57</td>
<td>14</td>
<td>21.98</td>
</tr>
<tr>
<td>Dec</td>
<td>1.57</td>
<td>14</td>
<td>21.98</td>
</tr>
<tr>
<td>Annual</td>
<td>3.86</td>
<td>Total: 204 days</td>
<td>Total: 817.76 kWh/m²</td>
</tr>
</tbody>
</table>

\[
\frac{(817.76 \text{ kWh/m}^2)}{204 \text{ days}} = 4 \text{ kWh/m}^2/\text{day}
\]

Table 1. Insolation adjustments for cloudy days.

In table 1, the total insolation for non-cloudy days is found in kWh/m² and then is divided by the number of non-cloudy days. This gives the average insolation in 4 kWh/m²/day for our region given it is a non-cloudy day.

![1995 Illumination curves](image)

Figure 1. Daily insolation curves given by VTB at Lafayette
System Sizing

A solar panel system appropriate for a given location can be sized given the average household energy consumption, insolation data, and the solar panel efficiency. The Department of Energy (DOE) reports that the typical household in our region uses an average of 28.7 kWh/day. The particular General Electric panels chosen for this project are 7.3% efficient. A system which would provide an average of 28.7 kWh/day can be sized using equation 2. Note that the average insolation given it is non-cloudy is used in the equation since that is the condition in which the system will be on.

\[
\text{size} = \frac{\text{AveEnergyConsummed}}{\text{AveInsolation} \times \text{PVefficiency}} = \frac{28.73 \text{kWh/day}}{4 \text{kWh}/m^2/\text{day} \times 0.073} = 98.4 \text{m}^2
\]

Eq. 2

In addition to the physical size, the maximum capacity of such a system can be calculated. When referring to the capacity of a solar panel system, the maximum capacity is usually given. The maximum capacity, in the unit Watts-peak (Wp), can be calculated according to equation 3. The maximum insolation is 1000 W/m², which can occur on a clear sunny day when the sunlight is perpendicular to the arrays [13].

\[
\text{capacity} = \max \text{Insolation} \times \text{area} \times \text{PVefficiency} = 1000 \text{W}/\text{m}^2 \times 98.4 \text{m}^2 \times 0.073 = 7150 \text{Wp}
\]

Eq. 3
These findings are consistent with the normal household solar array systems installed in this area. Trinity Solar, who installed the panels for this project, reports they install 5-10KWp systems for houses.

**Economic Analysis of Solar Energy**

Economic viability is a large concern when dealing with solar energy. The main expenses when dealing with solar energy include initial system cost, time value of money, and replacement of critical items. Below is an economic analysis over the course of a solar panel system’s 20 year lifespan to determine if solar energy is viable in our geographical region. The analysis is based on a 7kW system determined above with a panel area of 98.4 m².

Solar panels have an estimated cost of 4.5$/Wp and the inverter has an estimated cost of $0.85/Wp [14]. The panels and inverter will cost approximately $32,850 and $6,200 respectively. The panels will last 20 years, but the inverter must be replaced after 10 years. A modest battery system lasting for 2.5 days with an average energy use of 28.7 kWh/day and a battery cost of $150/kwh will cost $10,800. The batteries for this system will need to be replaced every 7 years, but it is estimated the battery cost will go down by 15% every 7 years due to technological advances [14]. These costs are given in table2.

<table>
<thead>
<tr>
<th>Initial costs = $57,830</th>
<th>Replacements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solar Panel Cost ($)</td>
<td>Installation ($)</td>
</tr>
<tr>
<td>32850</td>
<td>8000</td>
</tr>
</tbody>
</table>
Table 2. Photovoltaic system estimated costs.

On the 161 cloudy days in the year, an average of 28.7 kWh/day at $0.1095/kWh [15] will be bought from the grid. This means $475/year of electricity will be bought from the utility. Equation 3 shows the present day cost of the system assuming 5% interest per year over a 20 year system lifetime is $74,200. The photovoltaic energy will cost 59.7 cents/kWh, which is 5.45 times the region’s average. The cash flow diagram for the project is shown figure 2. Clearly at this time it is not economical to run a household off a solar panel system.

![Cash flow diagram for a household solar panels system in this region](image)

$$P = \frac{57,830}{(1 + 0.05)^{20}} + \frac{475}{(1 + 0.05)^{20}} + \frac{9,160}{(1 + 0.05)^{7}} + \frac{6,205}{(1 + 0.05)^{10}} + \frac{7,790}{(1 + 0.05)^{14}} = 74,200$$

Eq. 3

Although solar panels today are not economically viable, they will play an important part in the future of energy. Solar panel technology has not reached its full potential and there is ongoing work to improve photovoltaic cell efficiencies and to use low cost materials such as
polycrystalline solar cells. New power algorithms are being explored to maximize the energy that can be obtained from photovoltaic arrays. One of the limiting factors on the lifespan of a solar panel system is the electrolytic capacitors used in the inverter. Research is ongoing to find alternatives to using these capacitors. One of the goals of this project was to gain experience with power electronics and the subsystems involved in this upcoming technology.
2. Solar Panel Sustainability

Solar panels are one of the most promising sources of alternative energy. The amount of clean energy produced quickly outweighs any energy spent during the production phase. In addition, they are relatively easy to install compared to other energy sources. An article this past year in Scientific American stated “There are approximately 30 billion square feet (2.8 billion square meters) of expansive, flat roofs in the U.S., an area large enough to collect the sunlight needed to power 16 million American homes, or replace 38 conventional coal-fired power plants.[5]” The cleanliness and great potential of solar panels will keep them as a key technology for decades to come.

The table below summarizes the environmental effects of Solar Panels.
<table>
<thead>
<tr>
<th>Energy Use and Greenhouse Gas Emissions</th>
<th>Material Inputs</th>
<th>Manufacturing and Production</th>
<th>Use</th>
<th>Disposal and/or Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Major impact:</td>
<td></td>
<td>Major impact:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Fossil fuels (diesel) are</td>
<td></td>
<td>• Life-cycle greenhouse gas emissions are 40 - 55 grams per kilowatt-hour of generation capacity for standard silicon panels and 25 - 32 grams per kilowatt-hour for the newer thin-film technologies [2] and [1].</td>
<td>No impact:</td>
<td>• It takes 1/3 of the energy to make a solar panel from a recycled one rather than using new materials [7].</td>
</tr>
<tr>
<td>used for materials</td>
<td></td>
<td>• 45% of the total energy usage is from the production of polycrystalline silicon [2].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>extraction and for</td>
<td></td>
<td>• It is anticipated that with new technologies, life-cycle emissions will be reduced to 15 grams per kilowatt-hour [1].</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transporting those materials to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>manufacturing plants.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Energy from the electrical grid is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>also used for refining those materials.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Habitat Impacts                       |                | Major impact:               |     |                       |
| Major impact:                          |                | • Solid waste production is minimal. |     |                       |
| • Extraction of natural resources,    |                | • The fabrication of silicon solar cells requires large volumes of high purity water for silicon water cleaning. |     |                       |
| such as quartz, silicon carbide,      |                | • Many plants are designed to minimize waste consumption through recycling. |     |                       |
| glass and aluminum can cause habitat  |                | • All wastewater is treated and monitored prior to discharge under a Department of Environmental Quality (DEQ) water permit. |     |                       |
| disturbances analogous to sand and    |                |                             |     |                       |
| gravel pit mining but there is no    |                |                             |     |                       |
| leasing or precipitation process      |                |                             |     |                       |
| involving acids.                      |                |                             |     |                       |

| Local Air Impacts                     |                | Minimal impact:            |     |                       |
| Minimal impact:                        |                | • Fluorine and chlorine are emitted resulting from the neutralization of etching and texturing solutions and from flux gases. | No Impact: | • PV materials are usually encased in glass or plastic and do not release particles to the air. |
| • Emissions of solvents and           |                | • Fluorine and chlorine may be emitted to the air as a component of dust particles. |     |                       |
| alcohols contribute to                 |                | • 80% of the emissions of production are released during material processing related to solar glass manufacturing and soldering. |     |                       |
| photochemical ozone                    |                | • All air emissions are routed to pollution control equipment and covered under a Department of Environmental Quality (DEQ) air permit. |     |                       |
| formation and both direct (the       |                |                             |     |                       |
| solvent: tetra) and indirect (ozone)  |                |                             |     |                       |
| respiratory problems.                 |                |                             |     |                       |

| Occupational Health and Safety         |                | Minimal impact:            |     |                       |
| and Safety Impacts                    |                | • Silicon panel production can include fluorine, chlorine, nitrates, isopropanol, sulfur dioxide, nitrogen oxide, carbon dioxide, silica particles, oxides, acids and solvents. Some of which are considered to pose acute and/or chronic hazards to occupational safety. | No Impact: | • Recycling technologies for removing silicone from solar cells (from production waste or after module decommissioning) are not commercially available in the United States. |
| Minimal impact:                        |                | • The hazards of these substances are controllable with standard safety protocols usually employed in semiconductor industries. |     |                       |
| • Silica particles can be released in  |                | • Solar cells require very little maintenance, though they can be difficult to repair when maintenance is needed due to the risk of electrical shock. |     |                       |
| the mining and refining stage. If they |                |                             |     |                       |
| are small enough to be inhaled they    |                |                             |     |                       |
| may cause the lung disease            |                |                             |     |                       |
| silicosis—one that can easily be      |                |                             |     |                       |
| prevented with safety equipment.      |                |                             |     |                       |
3. Battery Sustainability

Batteries are essential in supporting the lifestyle of the current and upcoming generations. There are countless electronic gadgets that need portable power and due to the increasing importance of these portable tools for the normal functioning of the world, more attention has to be given to how they are affecting the sustainability of our world.

There are two types of batteries, disposable and rechargeable. The most widely used type of battery is the disposable battery; they are easy to use and a lot of electronics are designed to use them. Disposable batteries once contained toxic heavy metals, but due to a law passed in 1996, most battery manufacturers eliminated toxic heavy metals from their disposable batteries. Unfortunately this gives the manufacturers less motivation to recycle disposable batteries if there is no harm to the environment. This means more batteries end up in landfills wasting not only the materials in the batteries, but also land.

The other type of battery is rechargeable batteries. These batteries can get multiple uses from one cell, implementing one form of recycling. The chart below shows an estimate as to how many times a battery can be used before it can no longer hold a satisfactory charge. Along with the battery type is an estimated price of a single AA cell, a commonly used battery size.

<table>
<thead>
<tr>
<th>Cell Type:</th>
<th>Lead Acid</th>
<th>Ni-Cd</th>
<th>Ni-MH</th>
<th>Lithium Ion</th>
<th>Lithium Phosphate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cycles:</strong></td>
<td>500-800</td>
<td>1500-2000</td>
<td>500-1000</td>
<td>~1500</td>
<td>~2000</td>
</tr>
<tr>
<td><strong>AA price</strong></td>
<td>N/A</td>
<td>$1</td>
<td>$1.50</td>
<td>$4</td>
<td>$4</td>
</tr>
</tbody>
</table>
A standard AA alkaline battery costs about $0.75 and is done after one use. This shows that rechargeable batteries save enormous amounts of natural resources and save the consumer money. The drawback to rechargeable batteries is their use of toxic heavy metals. In order for rechargeable batteries to be better for the environment, the consumer has to take the responsibility of properly disposing of the battery. This can be done by taking the batteries to one of the many recycling centers for rechargeable batteries. This not only will save the environment, but the materials can be reused in more rechargeable batteries. “Since roughly 80 percent of batteries sold are disposable alkaline, it is up to consumers to transition to rechargeable batteries when possible.” [6]

4. Other Concerns

Additional sustainability concerns are mainstream to most electronic devices. “It is estimated that 70% of the heavy metals found in landfills (including mercury and cadmium) come from discarded electronic products.[9]” Our design also includes a small LCD display. LCD screens surround us everywhere in every day life, but they are easily disposed of. The liquid crystal material in LCD screens can contain 15-20 different chemicals[8]. Currently research efforts are being conducted to help recycle LCD screens, but it will continue to be a serious problem as the industry increases.
As our society begins focusing on alternative forms of energy, we also have to consider can our current system handle this energy. “We need an interstate transmission superhighway system,” states Suedeen G. Kelly, a member of the Federal Energy Regulatory Commission[7]. Currently, wind energy captured by windmills in the Midwest cannot be transmitted to the Northeast where power is needed. If our society were to adopt solar panels as a primary source of electricity, the electrical grid would have to be completely redone. Clearly, there would be several challenges if solar energy became widely adopted.
Sources


<http://tonto.eia.doe.gov/ask/electricity_faqs.asp#electricity_use_home>


Manufacturability

8.1 Memo 1: GPR008: Manufacturability

To: Senior Management
From: LRPDS Team
Date: 04/24/2009

Manufacturability is an important consideration in system design. The Statement of Work defines a production design as one which could be manufactured in large numbers (1000 units per year or greater). Part requisition is a large part of manufacturability. To ensure that the design is still constructible in the case of vendor closure, each item on the BOM must be available from at least two vendors. In the case that a part is only available from one manufacturer, an equivalent should be located.

An additional facet of the manufacturability of a system involves the tolerances that the system can withstand and still perform successfully. Often the values used in simulation are ideal values that either are not exactly realizable in hardware or do not take into account the tolerance which all components have. The Statement of Work requires that “any component which has a value that determines a critical voltage, time constant, frequency, or other parameter shall have a tolerance such that the system requirements are met with 99% yield in manufacturing.” To this end, three parts of the system have been identified as having a critical priority in the system’s manufacturability and are described below.

Part 1: Filter

The filter in the EDS sub module is used to filter the signal from the H-bridge to remove the undesired components, leaving only the desired 60 Hz. The Statement of Work indicates that the total amount of THD must not exceed 3%. The filter is comprised of an inductor and a capacitor with a resistor acting as the load. The inductor has a value of 1 mH and the capacitor has a value of 100μF. From the datasheets, the inductor has a tolerance of 15%, meaning that the possible values range from 0.85 to 1.15mH. Additionally, the value of the capacitor can range from 80 to 120μF.

Part 2: Sensors

Another crucial component of the design is the sensors. The sensors’ feedback is used in the monitoring of the system. For this reason, the accuracy of the sensors is important. Unlike the filter, the Statement of Work does not explicitly require the sensors to meet a certain threshold. Regardless, an analysis of the accuracy of the sensors allows the designers to be aware of the tolerances that must be dealt with. The sensors can be broken down into three types: voltage, current, and temperature.

Current Sensors
The current sensors used in the system are LEM HXS20-NPs which has an accuracy of 3%.

**Voltage Sensors**

The voltage sensors are simply resistor dividers with the value fed into an analog to digital converter. The resistors have a tolerance of 1% to ensure compliance.

**Temperature Sensors**

The Microchip MCP9700/9700A will be used for measuring the temperature in locations throughout the system. The datasheet specifies that the MCP9700/9700A has an accuracy of \( \pm 2^\circ C \) over the range of 0°C to 70°C which covers the expected operating temperatures of the system.

**Part 3: H-Bridge**

The H-Bridge is ultimately controlled through the EDS microcontroller, an ATmega128-16MC. The oscillator on the microcontroller runs at 16MHz, with a 0.005% tolerance according to the datasheet, indicating that the actual range is from 15.9992 MHz to 16.0008 MHz. The transistors on the H-Bridge are switched at 10 kHz, a much slower speed than the clock. This means the fluctuations in the speed of the oscillator will be negligible in the performance of the H-Bridge switching.

**EMI**

8.8 Memo 8: GPR003 – EMI/EMC Analysis

To: Senior Management

From: LPRDS Team

Date: 04/24/09

A General Project Requirement, and also a necessary step in any design project is the EMI (Electromagnetic Interference)/ EMC (Electromagnetic Conformance) analysis. It is within this document that we will analyze if the LPRDS-ETS-2009 design will meet all US CFR Title 47 Part 15 subpart B regulations for Class A digital equipment. While electronic control systems and power systems utilized by a public utility or in an industrial plant would be exempt from the specific technical standards contained in Part B, due to both specific requirements and safety precautions, an analysis was performed.

Conducted emissions must meet the FCC requirements for our system design. In order to meet these requirements, we must analyze the conducted emissions from all outputs of the system.

The first output of the system is the network connection coming from the SCADA computer. For the SCADA subsystem, our PC will be the Slim-Fit PC. Because we plan to use the PC as intended, and will be unmodified, we can use the companies regulatory certificate. This PC
satisfies all Subpart B regulations, as the certificate is shown here (http://www.fit-pc.com/files/fit-PC-Slim-certificate.pdf).

The second output of the system is the main 120 V AC. The requirements for conducted emissions are shown below:

(b) For a Class A digital device that is designed to be connected to the public utility (AC) power line, the radio frequency voltage that is conducted back onto the AC power line on any frequency or frequencies within the band 150 kHz to 30 MHz shall not exceed the limits in the following table, as measured using a 50 μH/50 ohms LISN. Compliance with the provisions of this paragraph shall be based on the measurement of the radio frequency voltage between each power line and ground at the power terminal. The lower limit applies at the boundary between the frequency ranges.

<table>
<thead>
<tr>
<th>Frequency of Emission (MHz)</th>
<th>Conducted Limit (dBμV)</th>
<th>Conducted Limit (dBV)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Quasi-peak</td>
<td>Average</td>
</tr>
<tr>
<td>0.15-0.5</td>
<td>79</td>
<td>66</td>
</tr>
<tr>
<td>0.5-30</td>
<td>73</td>
<td>60</td>
</tr>
</tbody>
</table>

The two main objects of concern in our design is the buck converter and the H-bridge. The buck converter will be switching at a frequency of 50 MHz and the H-bridge will be switching at a frequency of 10 KHz. Both of these objects are prior to the output low pass filter. In order to meet FCC compliance, we used PSPICE to create a model of the H-bridge followed by the low pass filter. Also in series is the model of the 50 μH/50 ohms LISN.

The results indicate that the projected output of the filter will limit the conducted EMI to meet the FCC Regulations shown. Figure 1 shows the FCC regulation limits, and the conducted emission based on frequencies.
The green line depicts the quasi-peak limit. This term references the spikes shown away from the trend line shown in deep red. The blue line shows the average limit required to be FCC compliant. This line must be higher than the deep red trend line, and is certainly at a safe level. One note worth mentioning is that our simulation software was only capable of simulating up to a frequency of 1.35MHz. Requirements state levels for up to 30MHz. I have assumed that past 1.35MHz, the same trend of the curve will continue, which would be compliant.

The second half of the EMI requirements is the radiated emissions. For unintentional radiators, the FCC has the following regulations to meet:

(b) The field strength of radiated emissions from a Class A digital device, as determined at a distance of 10 meters, shall not exceed the following:

<table>
<thead>
<tr>
<th>Frequency of Emission (MHz)</th>
<th>Field Strength (microvolts/meter)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 - 88</td>
<td>90</td>
</tr>
<tr>
<td>88 - 216</td>
<td>150</td>
</tr>
<tr>
<td>216 - 960</td>
<td>210</td>
</tr>
</tbody>
</table>
For radiated emissions, our main points of concern are again the buck converter as well as the H-bridge, both switching at high frequencies. This will lead to higher radiation, which cannot thoroughly be predicted, but can certainly be contained. We plan to house all of our major parts within metal casings to reduce the emitted radiation.

While we believe that this will be enough containment in the room to keep the radiated emissions below FCC requirements, we cannot be completely certain until a test has been completed. While the statement of work explicitly states that empirical data is not necessary, if shown to be a problem, we must be able to proceed to a test procedure. If we were to conduct a test, a large antenna able to pick up frequencies of 30MHz+ would be required, along with a spectrum analyzer able to measure in (microvolts/meter). This test, however, would not be done in an EMI chamber, therefore making it a non-compliant test. We would be able to test for a baseline with the system off, and compare to a test with the system running for a general idea of the system, but would be unable to classify the system as compliant.

In the case that the radiation emissions are too large, there are several techniques and approaches we may use to reduce the emissions. One technique is using shielding solutions. We may choose to upgrade to die cast boxes, sheet metal assemblies, conductive polymers, conductive painting and plating, or a molded grid shield. Another option is if the conductive emission is higher than expected, we may add ferrite cores to the end of our cables. These big cylindrical bumps on the cable will reduce the amount of undesired signals.
**SCADA Read Me**

**Introduction**

**Goal**

The purpose of the Supervisory Control and Data Acquisition (SCADA) software was to allow monitoring and control of the Lafayette Photovoltaic Research and Development System (LPRDS). The software would run on a computer called the FIT PC which is located in the LPRDS Tower. One general requirement was that all monitoring and control interactions with the system from the applications would be through a defined Application Programming Interface (API). For monitoring, the software had to provide the ability to poll Voltage, Current, and Temperature readings and store them in a Database, which could be accessed by applications as well as by a website. For control, the software had to provide the ability to control the switches and relays in the system from applications. Here is a simple block diagram of our initial view of the software to control the system:
**Basic Ubuntu Commands**

The software is running on the FIT PC, which has Ubuntu 9.04 installed. For interacting with the software, there are various useful terminal commands listed below.

`ssh –Y address`

This command allows a user to connect to another computer using the Secure Shell protocol. For example, if the user wanted to connect to the FIT PC, the command would be `ssh –Y fit@lprds.aec.lafayette.edu`. This is essentially the same as opening a terminal on the FIT PC itself, which can be useful for interacting with the software and the system from outside the project room. The password for the FIT PC is 111111. An example is shown below.

`ls`

This command lists all of the files in a directory. Adding `–l` switches it to a “long listing format”, essentially putting in a list format.
This command allows a user to search the list of all running processes for a specific term. For example, if the user wanted to see if any process with “lprds” in the name was running, the command would be `ps aux | grep lprds`. The command to view all running process is `ps aux`, and the command to search for a specific term is `grep searchterm`, and the `|` “pipes” the results of `ps aux` into `grep searchterm`. An example is shown below.

This command moves a file or files from the `fromLoc` to the `toLoc`. For example, if you were in the `/Desktop/scada` directory, and typed `mv kern /usr/local/lprds/bin/lprdskernd`, it would move the file “kern” in the `/Desktop/scada` directory to the `/usr/local/lprds/bin/` directory, and would rename it to `lprdskernd`. An example is shown in the `sudo` section below.

Putting `sudo` before a command allows the command to be run with root access (runs it as administrator). For example, if the directory you are trying to move a file to has restricted
permissions, the mv command might give you a permissions error. However, if you put “sudo” before it, it won’t give you an error about permissions. Sudo often asks for a password before executing the command, which, on the FIT PC, is 111111. An example is shown below.

```
tail filename
```

This command shows the last 10 lines of the file specified by `filename`. If you add the `–f` option, it continues to show the end of the file. An example is shown below.

```
apt-get install packagename
```

This command is used to install many packages in Ubuntu. It normally requires `sudo` before the command. For example, if you were to type `sudo apt-get install randomPackage`, it would install `randomPackage`. 
Starting, Stopping, Compiling the Software

Starting and Stopping Kernel and the Operational State Manager

Both Kernel and the Operational State Manager are supposed to automatically start when the FIT PC is turned on. You can check if they are running using the `ps aux | grep lprds` command, and looking for `lprdskernd` and `lprdsoperd`.

Manually Starting and stopping the Kernel or the Operational State Manager is done using scripts located in the `/dev/init.d/` directory on the FIT PC. Always check to make sure they are not running before trying to start either Kernel or the Operational State Manager.

The command to start the kernel is `sudo /etc/init.d/lprdskernd start`, and the command to start the Operational State Manager is `sudo /etc/init.d/lprdsoperd start`. Stopping the kernel or Operational State Manager is done through `sudo /etc/init.d/lprdskernd stop` and `sudo /etc/init.d/lprdsoperd stop`.

Note: Any application, including the Operational State Manager, must be started after Kernel, and must be stopped before Kernel for the software to work correctly.

Starting and Stopping Applications

All applications other than the Operational State Manager should be started from the `/Desktop/scada` directory on the FIT PC. To start them, you simply need to type `./AppName`. For example, the commands to start the existing applications are: `./maint`, `./mgmt`, `./clear_hv_unsafe`. Keep in mind, all applications require Kernel to be started in order for them to work properly.

Stopping the applications varies slightly between the applications. The maintenance application has an interface, so that the user can simply type “exit” to close the application. The battery management application has no interface, so to stop it the user would need to hold Ctrl and press C to stop it. The clear HV Unsafe application exits automatically after the user input request.
Compiling the Software

Compiling the software is done through a Makefile located in the /Desktop/scada directory. Before compiling an application, you should make sure that the application is not currently running. Once in the directory, simply type “make app” in order to compile the specific application. The make kern command will compile the kernel, and should be followed by sudo mv kern /usr/local/lprds/bin/lprdskernd. The make lprds command will compile the Operational State Manager, and should be followed by sudo mv /usr/local/lprds/bin/lprdsoperd. The make maint command will compile the maintenance application, and does not need to be followed by a mv command. The make mgmt command will compile the battery management application, and does not need to be followed by a mv command. Finally, the make clear_hv_unsafe command will compile the clear HV Unsafe application, and does not need to be followed by a mv command.

Troubleshooting

One possible problem with starting an application can arise from the application being shut down improperly. When this happens, the pipe which connects the application to the kernel will be left, and can interfere with the application starting up correctly. The pipes are located in the /usr/local/lprds/var/ directory, and the application specific pipes have the names “Results_AppName”. So, if the Maintenance application isn’t starting correctly, and typing ls –l /usr/local/lprds/var/ reveals that the Results_Maint pipe still exists, even though the Maintenance Application isn’t running, you should remove the pipe with sudo rm /usr/local/lprds/var/Results_Maint.

If the programs are running correctly, but the DAQ boards aren’t responding, one possible problem is that the SIB is using different USB ports than specified in the xcr.h file. For example, the software may be set to use /dev/ttyUSB0 (for the DAQ boards) and /dev/ttyUSB1 (for the Sunny Boy), while the SIB is set to use /dev/ttyUSB1 and /dev/ttyUSB2. To check the ports the SIB is set to use, type tail –f /var/log/messages, and then unplug the SIB from the USB port on the FIT PC, and then plug it back in. In the xcr.h file, the USB port should be set to the lower numbered USB port, and in the yasdi.ini file the Device should be set to the high numbered USB port. If these numbers have changed, the software will need to be stopped, recompiled, and started again.

If after using the scripts to stop the Operational State Manager or Kernel, the program is still running (you can still see lprdsoperd or lprdskernd using ps aux | grep lprds), you can still shutdown the process using the kill command. This can be used for any application which refuses to shut down. The command to shut it down would be sudo kill -9 pid, where pid is the Process ID. The Process ID can be found using ps aux | grep <name>, it is the number in the second column.

To fully shutdown and restart all LPRDS software, here are the steps to follow:
1. Stop any applications that are running using the methods listed under Starting and Stopping Applications and Starting and Stopping Kernel and the Operational State Manager. If these methods don’t work, then shutdown the processes using the kill command mentioned above.
2. Type sudo rm /usr/local/lprds/var/* to ensure that all pipes have been removed.
3. Start the Kernel, then the Operational State Manager, and then any applications that you want to run (assuming they are allowed to run: Note that the battery management application is only allowed to run in the Operational State).
Using the Software

Software Block Diagram

For a description of the individual blocks, see the SCADA Software Architecture document.

When an application wants to get data from the Data Acquisition boards, it calls the appropriate C++ API function. This request goes through the Kernel pipe (explained in a later section) to the Kernel. Kernel then calls the appropriate function in IOMgr, which formats the packet that needs to be sent to the Data Acquisition (DAQ) boards. XCR is responsible for physically writing the packet to the USB port, which gets sent to the SIB. The SIB passes the packet along to the DAQ boards using RS-485, which have sensors that are controlled by a microcontroller. The microcontroller sends a response back with the requested value, which is passed through the SIB to the USB port. XCR reads the packet, and passes it to IOMgr which decodes the packet, sending only the requested value back to Kernel. Finally, Kernel returns the requested value through Results pipe to the API, and to the application.

The data flow for controlling the digital outputs on the DAQ boards is exactly the same as for requesting data. Sunny Boy requests are also done in a similar fashion, except that the Sunny Boy class is used instead of the IOMgr and XCR.

Every time that data is received from the DAQ boards or Sunny Boy, as well as when switches are set, the data or switch position is recorded in the database. This data is available to the website using PHP to access the database.
**Pipes**

The C++ API uses a UNIX feature called pipes to allow communication between the applications and the Kernel, which is constantly running. Pipes are essentially a FIFO between two running processes. One application can write to the pipe, and then another application can read from it. Writing to and reading from pipes is very similar to interacting with a file, as pipes are essentially special files that are optimized for having multiple processes write to them.

In the interactions between the Kernel and the applications, there are two kinds of pipes: the kernel pipe, and the Results pipes. There is only 1 kernel pipe, which is meant to be read by Kernel, and written to by applications. There is one Results pipe for each application, which is created when the application calls the Connect function in the API, and deleted when the application calls the Disconnect function in the API. All pipes are created in the /usr/local/lprds/var/ directory.

**Database**

The LPRDS software has the ability to automatically log system information to a MySQL database. This was accomplished by creating a MySQL database locally on the fitPC and using MySQL++, which is a commercial package that allows communication from C++ to MySQL. The current database can be viewed on the fitPC with a graphical user interface within firefox. A ‘database’ link is available in the bookmark bar, or you can type in the following address:

http://localhost/phpmyadmin/

Username: root Database: SCADAdata
Password: 111111 Server = localhost

The database is arranged into subsections known as tables with each entry in the table known as a row. The SCADAdata database is arranged into 9 tables, each with an auto-incrementing index named ‘id’. Tables such as event_log, fault_log, state_log and data have additional fields ending in ‘_id’ (event_id, hw_id etc.) that correspond to the ‘id’ number of the table named in the prefix. For example, the ‘hw’ table stores detailed information about each sensor and assigns each a unique index ‘id’ number. When a sensor reading is stored in the ‘data’ table, it only needs to store the matching ‘id’ number from the ‘hw’ table in the ‘hw_id’ field to identify the sensor. This was done to minimize the amount of space used by the database.
**SQL Commands**

To manipulate data in the database we issue commands written in the Structured Query Language (SQL). In the LPRDS system, SQL commands are commonly used to retrieve data, write new data, and update previous entries. Below are a few SQL command examples:

\[
\text{SELECT} \ \ast \ \text{FROM} \ \text{hw} \ \text{WHERE} \ \text{daq} = 4 \ \text{AND} \ \text{pin} = 7
\]

The ‘SELECT’ command is used to retrieve information back from the database. The ‘\ast’ means retrieve every field from the table. ‘FROM hw’ specifies the table to pull data from and ‘WHERE’ is used to filter information by finding entries that match the following specifications.

\[
\text{SELECT} \ \text{data.id, data.hw_id, hw.daq, hw.pin, hw.name, data.value, hw.units, hw.type, hw.active, data.time} \ \text{FROM} \ \text{hw, data} \ \text{WHERE} \ \text{data.hw_id} = \ \text{hw.id}
\]

In this ‘SELECT’ command, information is gathered from two separate tables. To do this, each desired field must be prefixed with its table name, (ex. data.id for the id field in the data table)
and ‘FROM’ is used to specify the tables. In this case, WHERE is used to match up each ‘data’ entry with its detailed sensor information in the ‘hw’ table by matching data.hw_id to hw.id.

\[
\text{INSERT INTO state_log (state_id, time) values ( 3, NOW())}
\]

The ‘INSERT INTO’ command allows us to add a new entry to the database. It is best to specify the fields you intent to populate after the table name within parenthesis, leaving out ‘id’ to avoid problems with the automatically incremented index field in each table. Use NOW() for the timestamp value to write the current time. Any text must be enclosed in single quotes ‘’.

\[
\text{UPDATE hw SET active = 1 WHERE name = 'sensor_name'}
\]

To change the value of information already in the database, use the ‘UPDATE’ and ‘SET’ command. The values to be updated can be specified using ‘WHERE’ as seen before. Any text must be enclosed in single quotes ‘’.

\[
\text{ORDER BY event_log.time ASC}
\]

The ‘ORDER BY’ option can be added to the end of a ‘SELECT’ command to arrange the results by their timestamp field. Use ‘ASC’ for ascending time and ‘DESC’ for descending.

**Tip:** When formulating a new SQL command, it is convenient to use the phpmyadmin user interface in firefox. You will find a tab near the top of the page labeled ‘SQL’ where you can type in commands and get feedback if there is an error in the formatting.

**Installing MySQL and MySQL++**

MySQL and MySQL++ are already configured on the fitPC and right-most Ubuntu computer along the windows in AEC400. To install MySQL from scratch (as well as PHP and Apache for the website) follow this tutorial:

https://help.ubuntu.com/community/ApacheMySQLPHP

To install MySQL++, run the following commands in terminal:

```
wget http://www.tangentsoft.net/mysql++/releases/mysql++-3.0.9.tar.gz
```

```
tar xvfz mysql++-3.0.9.tar.gz
```
To use MySQL++ in a C++ program, you first will need to modify the Makefile with the following lines:

```
CFLAGS := -I /usr/include/mysql -I /usr/local/include/mysql++
LDFLAGS := -L /usr/local/lib -lmysqlpp -lmysqlclient -lnsl -lz –lm
$(CFLAGS) $(LDFLAGS)
```

Then within the .h file include…

```
#include <mysql++.h>
#include <stdlib.h>
using namespace mysqlpp;
```

**Sunny Boy**

**Using the yasdi library:**

The Sunny Boy communication protocol is implemented using the YASDI (yet another SMA data implementation) library which was provided by the Sunny Boy manufacturer. The Sunny Boy must be connected to the FIT PC in order to be properly communicated with via an RS-485 connection. The current Sunny Boy class utilizes only a few of the provided functions from the YASDI library; however, only a select few are required in order to poll data from it. These methods and their definitions are located within the sunnyBoy.cpp file. Since these methods draw upon the YASDI library, those libraries must be compiled along with the Sunny Boy code. Consult the Sunny Boy Communication Manual to see how to use this code to communicate with the Sunny Boy.

**Pico-LCD**

**Viewing the configuration file and the contents:**

The file that determines the layout for the Pico-LCD is located in /usr/local/lprds/etc/, the file is named lcd4linux-lprds.conf. By modifying this file, the displayed elements of the Pico-LCD screen can be changed. In the beginning of the file there are various widgets that are
currently being displayed on the Pico-LCD screen. There are additional widgets available to add and a list of available widget is located at [http://ssl.bulix.org/projects/lcd4linux/wiki/Layout](http://ssl.bulix.org/projects/lcd4linux/wiki/Layout) along with their respective fields and examples. Near the bottom of the screen is the layout which describes where the widgets will be displayed on the Pico-LCD screen. Here is an example screenshot of how the widgets are put into the file:

```
The exec expression executes a terminal command and prints the result on the LCD screen at a refresh rate specified by the second argument.

Starting and Stopping the configuration file:

The configuration file mentioned above is currently set to be run on startup so when the fit pc is booted up the layout described in the file will be displayed. The startup script, lcd4linux,
located in the /etc/init.d directory on the FIT PC, is responsible for calling the correct commands to configure the layout. It has a start and stop command which can be called manually from the terminal by `sudo /etc/init.d/lcd4linux start` and `sudo /etc/init.d/lcd4linux stop`. These are useful if the configuration file has been changed and the changes are desired to be displayed immediately.

**Package List**

Here is a list of the packages that were installed on the FIT PC to run the SCADA software. Many packages can be installed using the Synaptic Package Manager found in the System>Administration menu.

<table>
<thead>
<tr>
<th>Package</th>
<th>Package</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNUPLOT</td>
<td>MySQL++</td>
<td>G++</td>
</tr>
<tr>
<td>php5-mysql</td>
<td>mysql-server</td>
<td>mysql-client-5.0</td>
</tr>
<tr>
<td>phpmyadmin</td>
<td>mysql-server-core-5.0</td>
<td>mysql-client</td>
</tr>
<tr>
<td>mysql-common</td>
<td>mysql-server-5.0</td>
<td>libmysqlclient15off</td>
</tr>
<tr>
<td>libqt4-sql-mysql</td>
<td>libsoci-mysql-gcc</td>
<td>libmysqlclient15-dev</td>
</tr>
</tbody>
</table>