Solar Powered Window Blinds

Group 11

Artis Coleman - Computer Engineer
Sean Diamond - Optical Engineer
Dakota Jordan - Computer Engineer
Stephen Walsh - Electrical Engineer

EEL 4914 – Senior Design I – Summer 2015
EEL 4915 – Senior Design II – Fall 2015

Table of Contents

1.0 Executive Summary

This capstone design project will adapt the traditional window blinds to provide additional useful functionality for a residential home, with its main feature being the collection of solar energy. Far too much electricity that powers the world today comes from the consumption of fossil fuels, a non-renewable resource. While it is not a current necessity, it will soon become increasingly important to find alternative methods of electricity generation before our limited resources run out.

This project will introduce just one of many envisioned products that will be commonly found in residential homes in the future which will collect energy to be stored in a central power storage unit which will power the entire residence. These central electricity storage units are essentially large rechargeable batteries, prototypes of which are already emerging in our current economy (e.g. Tesla's Powerwall).

To this end, our project will modify the traditional residential window blinds to complement the power infrastructure of a modern home.

2.0 Project Description

2.1 Project Motivation and Goals

Since the main focus of this project is to provide additional natural energy to a residence, it is only logical that it should maximize the energy it collects by minimizing the energy it consumes while still providing features that a consumer expects from a technologically-advanced product in the current market. To meet these goals, the blinds will aim to be lightweight, relatively low cost, and use minimal energy to perform basic operations such as opening and closing.

The current state of technological development is seeing an increasing amount of products developed for the "Internet of Things". These blinds aim to be a member of this category of products by being interconnected wirelessly with the user's mobile device to allow for greater control over their home. Motor operation for these blinds opens up the ability to widen our demographic to the elderly or disabled by making the device easier and more convenient to operate.

2.2 Objectives

The main objective of this project is to provide a supplementary power source for an energy-conscious home. The operation of the product itself will be self-powered so that it will not use up disposable batteries or require external electricity. Additionally, the excess energy gained through the solar cells can be stored in a battery for use in other areas of the home. A USB charge port is included on the product as the only provided

interface for accessing this excess energy. However, if a central home battery storage device is used, it is assumed that it will have additional interface ports to utilize the energy provided. Multiple installations of this product can provide a significant boost in the home's electricity reserves.

Secondary objectives include convenient operation of the blinds and a thermometer. The ability to open and close the blinds with the push of a button allows for more convenient use of the product. This is an especially useful feature for disabled persons who may not be able to open and close traditional blinds manually. Also, with the use of the mobile application, even less effort is required from the user. Information from the thermometer will be displayed on a screen on the product itself, as well as transmitted to the user's phone.

2.3 Requirements Specifications

2.3.1 Hardware Specifications

A list of hardware specifications has been established in order to clearly define the product and assist in design of the hardware. These specifications, listed in Table 1, must be met in order for this project to be considered complete.

#	Description
1	As many solar cells as possible mounted to the housing to maximize surface area
2	Install additional cells on the light-blocking material of the blinds if possible
3	Fit window with width of at least 30 inches
4	Ability to recharge an external battery source
5	Operation of motor from both buttons and mobile application
6	USB port to act as charging station for USB-compatible devices
7	Attached thermometer on the housing
8	Display screen for thermometer and battery life
9	Communicate wirelessly with mobile phone app
10	Battery source will be interchangeable
11	Adjustable height to work with multiple window sizes.
12	Housing should be durable and resistant to warping from sunlight

Table 1 - Hardware Specifications

2.3.2 Software Specifications

A list of software specifications has been established in order to clearly define the product and assist in design of the software. These specifications, listed in Table 2, must be met in order for this project to be considered complete.

#	Description
1	Mobile application to work in Android environment
2	Controls opening and closing of blinds
3	Displays the battery's current charge % of maximum
4	Displays temperature read by the thermal sensor
5	If battery is expending more energy than it's charging, display time remaining until battery is empty if current rate is maintained
6	Should be able to interact with the blinds from at least 25 feet away
7	Includes secure "sync" option to associate your blinds with your device
8	Track and display charging statistics
9	Display the rate of battery charging from the solar cells
10	Display time until battery is fully charged
11	Display dollar amount saved based on local energy rates
12	Compatible with at least 3 most recent API versions of Android

Table 2 - Software Specifications

The specifications listed in Tables 1 and 2 are the minimum requirements for our project. As development progresses, more features will be added if they are determined to be useful and fit the constraints placed upon us for this project.

3.0 Research Related to Project Definition

3.1 Existing Similar Projects and Products

3.1.1 Similar Products: Motorized Window Blinds

Biochemtronics has an article titled "Automatic Window Blinds Controller (PICAXE)" published on instructables.com [1]. The project involves adding a microcontroller-powered motor to a regular set of window blinds. Motorized window blinds are available for sale at home improvement stores; however, they are usually quite expensive. An inexpensive PICAXE Micro Controller is used for this project and the total cost of the project is only about $20.00. Biochemtronics used a light dependent resistor (LDR) to activate the motor and open the blinds when a pre-set level of light is present on the window. This feature will not be useful in our project because we want the blinds to be closed when sunlight is present; however, the overall design of the blind controller is similar to what we want to achieve in our project. The parts list for this project is shown in Table 3.

Part	Manufacturer	Cost
PICAXE -08M Micro Controller	Spark Fun Electronics	$2.95
ULN2003A Darlington Array	BG Micro #ICSULN2003	$0.59
DPDT 5.0V Relay	BG Micro # REL1106	$1.29
Solarbotics GM3 Gear Motor, 224:1 6V	Solarbotics	$5.50
3.5mm Stereo Jack	BG Micro #AUDCA017	$0.36
4 X 1.5V AA Battery Holder	BG Micro #BAT1068	$0.79
Battery Snap (9V style)	BG Micro #BAT1033	$0.25
LM7805T 5.0V, 1A, Regulator	BG Micro #REG7805T	$0.40
Small Project Box	BG Micro #ACS1157	$1.95
Small Proto Board (2 3/8)	BG Micro #ACS1433	$0.89
8 Pin Dip Socket	BG Micro #SOC1036	$0.10
(2) 16 Pin Dip Sockets	BG Micro #SOC1038	$0.08
Light Dependent Resistor	Radio Shack #276-1657 (5 pk)	$2.99
(2) SPST Switches	BG Micro #SWT1043	$0.20

Table 3 - Biochemtronics Automatic Window Blinds Controller BOM

The PICAXE-08M serves as the brain for the controller. This chip is an 8 pin DIP that features several inputs and outputs, analog to digital converters, a pulse width modulator, and an IR transceiver. It was originally designed for educational use and is easy to program. It is also more affordable than other microcontrollers because it runs on freeware. The LDR attaches to the analog to digital converter (ADC) input of the PICAXE to monitor the light level outside. The ADC sets a variable to a value between 0

and 255 based on the intensity of the light on the LDR. This variable is used to set the light level at which the PICAXE outputs will turn on and open or close the blinds. The Darlington array is used to amplify the low current of the PICAXE outputs so that they can switch the high currents required by the relay and motor. The Darlington array also protects the rest of the circuit from voltage spikes caused by the inductive loads. The 5V DPDT relay is used to reverse the polarity of the motor so that the blinds can both open and close. The pulse width modulator of the PICAXE can be used to control the motor speed; however, this is not required when using the Solarbotics gear motor. The circuit diagram for the controller is shown in Figure 1, printed with permission from Biochemtronics.

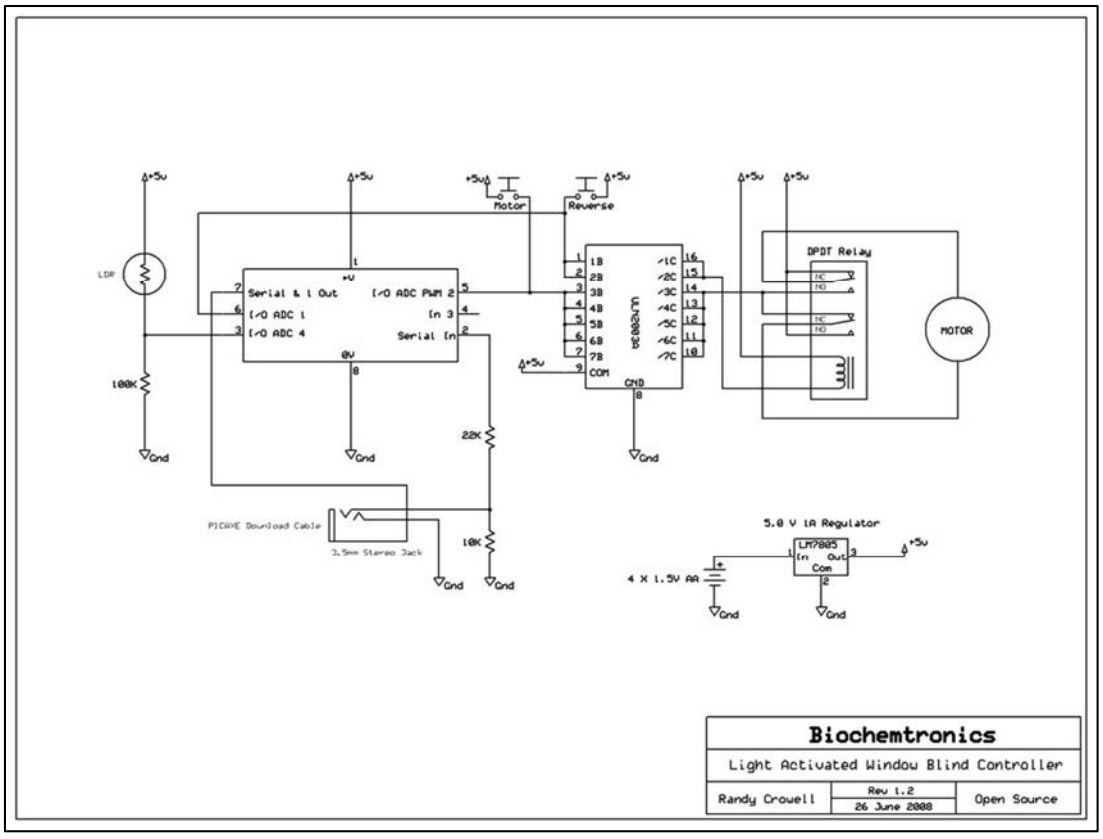

Figure 1 - Biochemtronics Light Activated Window Blind Controller

In order for the motor to open and close the blinds a small hole needs to be drilled in the motor shaft. After removing the rod from the blinds the small hook at the top of the rod used to open the blinds can be put through the hole that was drilled in the motor shaft. When the motor is activated it will cause the hook to spin and open and close the blinds. Position adjustment switches can be added to the circuit in order to manually open and close the blinds. A 360° servo or stepper motor could be used in place of the gear motor to increase the position accuracy; however, this would increase the cost of the project.

Once the circuit is constructed, the PICAXE needs to be programmed. The code reads the outside light level from the ADC and defines the levels at which the blinds will open and close. Code is also included in order to keep the blinds from trying to open again once they are already open and from trying to close once they are already closed. The open and close position needs to be adjusted so that the motor will not under or over-rotate. Connecting a computer to the controller that is mounted to the blinds allows the variables in the code to be fine-tuned. If the controller gets off track over time the adjustment buttons can be used to tweak the blinds back to the desired position. A picture of the completed project is shown in Figure 2, printed with permission from Biochemtronics. The battery pack is attached to the top of the project box so that the batteries can be easily replaced or recharged.

Figure 2 - Completed Biochemtronics Project

This project provides a good example of how a simple microcontroller and inexpensive motor can be used to motorize blinds. There are several adjustments that need to be made in order for this design to be incorporated in our project. The main difference between this project and our design is the blind type. We want to use a roller shade system instead of a conventional blind system with slats. Using a roller blind will allow a large, bendable solar panel to be attached to the shade. The solar power collected by the shade could then be used to power the motor as well as the microcontroller and a built in USB charger. Another feature that we want to incorporate in our project is remote controllability. This will allow the user to control the blinds using a smart phone application. The LDR could be repurposed to monitor light levels over a certain period of time. This will allow a mobile user to monitor the best times for collecting solar energy.

3.1.2 Solar Powered USB Charger

Honus has an article titled “How to make a solar iPod/iPhone charger –aka MightyMintyBoost” published on instructables.com [2]. The project involves converting an Adafruit MintyBoost charger, which usually runs off of AA batteries, to a solar powered charger that recharges a lithium polymer battery. The MintyBoost kit ($19.50) includes the following parts:

- 5V boost
- converter
- 8-pin socket
- 220uF
- power supply capacitors
- 0.1uF bypass capacitors
- 3.3kΩ, 75kΩ, and 49.9kΩ resistors
- 1N5818 Schottky diode
- 10uH power inductor
- USB type A female jack
- 2 x AA battery
- holder
- PCB

The Mintyboost is designed to provide a 500mA charge rate to the USB chargeable device. The 49.9kΩ and 75kΩ resistors are used in order to adjust the bias on the four pins of the USB so that iPhone charging is supported. Many USB devices that do not require data transfer will charge with pins 2 and 3 of the USB floating; however, the iPhone requires these pins to be biased at 2 V to attain a 500mA charge rate. The circuit diagram of the Mintyboost’s USB pin configuration is shown below in Figure 3, printed with permission from Honus.

The 3.3kΩ resistor is used to improve the high current capability of the boost converter chip. One of the ceramic capacitors is used in order to stabilize the output voltage as well as filter out high frequency noise. The other capacitor is used to stabilize the internal reference of the boost converter chip. The Schottky diode is used to ensure energy is transferred in only one direction, from the battery to the USB port. The boost (or step-up) converter chip is a DC/DC power converter that has an output voltage greater than the input voltage. The IC socket protects the boost converter chip and can easily be replaced if there are any problems with the circuit. The power inductor is used by the DC/DC converter chip to store and convert power from low to high voltages. The electrolytic capacitors are used to stabilize the input and output voltages during up-conversion. They also act to filter low frequency noise. The circuit diagram for the MintyBoost v3.0 is shown below in Figure 4, printed with permission from Honus.

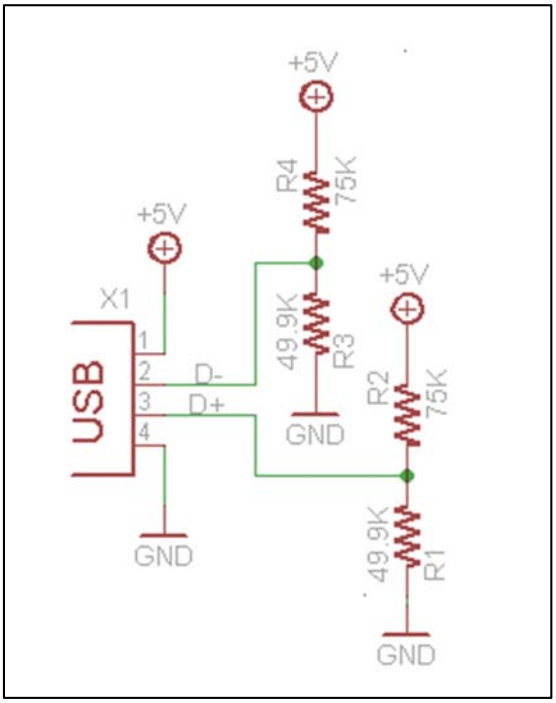

Figure 3 - Mintyboost USB Pin Configuration

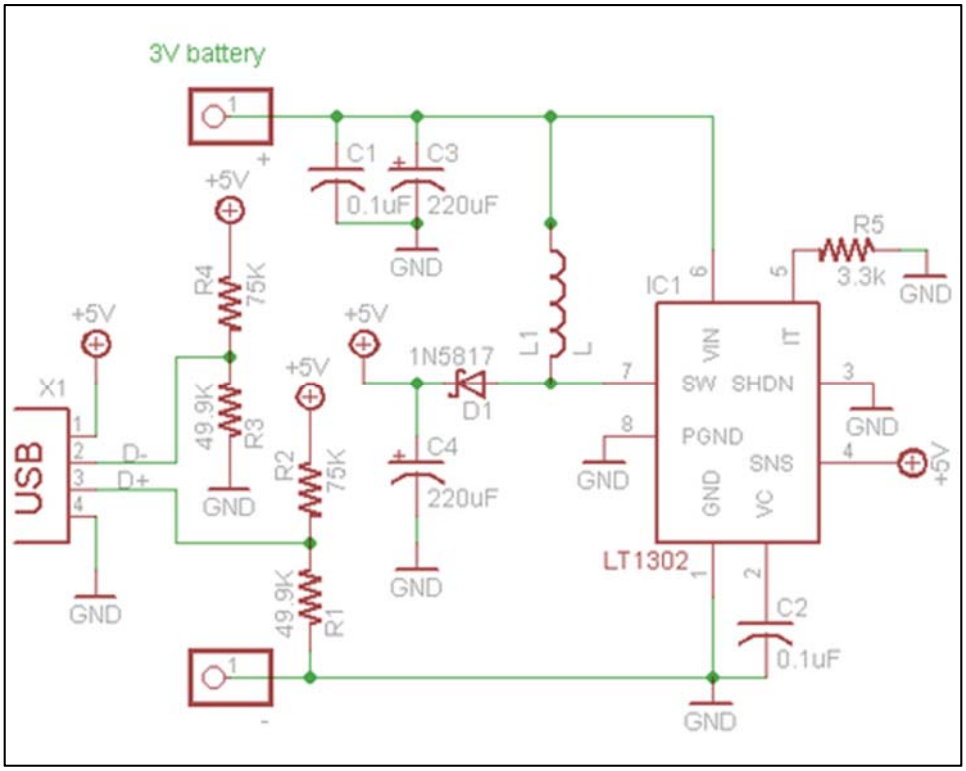

Figure 4 - Mintyboost Circuit Diagram

Once the MintyBoost kit is assembled, a 2-pin JST cable can be soldered to the PCB in place of where the AA battery holder would normally be attached. Figure 5, printed with permission from Honus, shows the MintyBoost circuit with the added JST cable. The JST can then be connected to the load output port of the Adafruit USB LiPoly Charger ($17.50), which was specially designed for solar use. If a regular charger were used the efficiency would be poor due to the charger constantly turning on and off in varying light conditions. The MCP73871 Voltage Proportional Charge Control chip is used to draw as much current as possible while maintaining a constant voltage point. Resistors are used in order to set the voltage point and a large capacitor is used to stabilize the rapid voltage collapse of the panel. In addition, a Schottky diode charges the capacitor from the panel and prevents the capacitor from draining back into the panel. If the amount of power generated by the solar panel where larger, a Max Power Point Tracker (MPPT) circuit would be needed. This circuit uses a DC/DC converter to keep the charger operating at the maximum power. Since adding the DC/DC converter is expensive, and only increases the efficiency by 30% for small panels, a linear converter is the right choice for this application.

Figure 5 - MintyBoost Circuit with Added JST Connector

An additional feature offered by the MCP73871 chip is load sharing. When the panel is producing power, the load current goes directly from the input voltage to the output. The li-poly battery will supply up to 1.8A if the current required by the load is higher than what is provided by the panel. An extra USB mini-B and DC jack are included on the board so that the battery can be charged when solar energy is not available. Three indicator LEDs allow the user to easily monitor the status of the charger. The red LED indicates that there is power connected to the charger, the orange LED indicates when the battery is being charged, and the green LED indicates when the battery is fully charged.

The last step for this project is to connect the 3.7V 2.5A-h Adafruit Li-Poly Battery ($14.95) and the Adafruit 6V 3.4W solar panel ($44.95). A 2.1mm terminal block adapter can be used to connect the panel to the DC jack on the charger. This panel is made of monocrystalline silicon and has a cell efficiency of 17%. It outputs 6V at 530 mA and is

waterproof, scratch resistant, and UV resistant. The Li-poly battery can be connected to the battery port on the charger by a 2-pin JST cable. The output ranges from 4.2V to 3.7V and has a capacity of 2.5A-h for a total of about 10W-h. Protection circuitry is built into the battery, which prevents over-charging, over-use, and output shorts. A 10k Precision Epoxy Thermistor ($4.00) can be added on to the board in order to monitor the temperature of the battery and shut down charging if the battery becomes too hot. This is useful if the battery will be used outside. Figure 6 shows how the Li-Poly battery and USB charger connect to the charge controller and Figure 7 shows the completed project with the solar panel, both figures 4 and 5 were printed with permission from Honus.

This DIY project provides good insight into the kind of circuitry that is needed in order to construct a solar power USB charger. The parts list for this project is shown in Table4. The total cost for this project is $100.90. A larger solar panel and battery could be used in order to scale this project so that more solar energy could be captured and stored. This system would require an MPPT charge controller in order to maximize the amount of energy transmitted from the solar panel to the battery. A lead acid battery could be used instead of a Li-Poly battery in order to reduce cost.

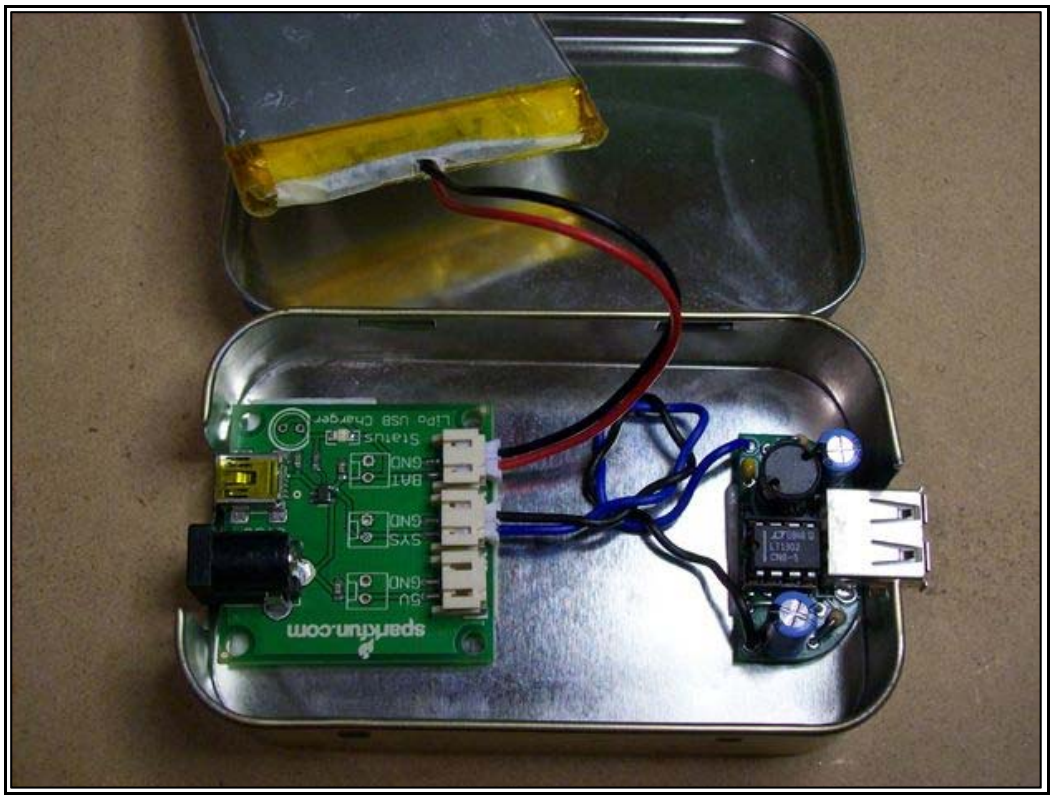

Figure 6 - Battery and USB Charger Connected to Charge Controller

Figure 7 - Completed Solar USB Charger

Part	Manufacturer	Cost
MintyBoost Kit v3.0	Adafruit	$19.50
USB/DC/Solar Lithium Ion/Polymer Charger v2	Adafruit	$17.50
Lithium Ion Polymer Battery – 3.7v 2500mAh	Adafruit	$14.95
Large 6V 3.4W Solar Panel – 3.4 W	Adafruit	$44.95
10K Precision Epoxy Thermistor	Adafruit	$4.00
	Total	$100.90

Table 4 - Solar USB Charger Parts List

3.2 Relevant Technologies

3.2.1 Solar Cells

3.2.1.1 Photovoltaic Technology Considerations

There are a variety of solar cell technologies to choose from when designing an application that will use solar energy as its main source of power. One of the most important characteristics to assess is the efficiency of the cells. The efficiency of a solar cell refers to the ratio of incident light energy to the converted electrical energy. The spectrum of incident light, semiconductor material used, and device structure all have an effect on the efficiency of the cells [3]. The most efficient photovoltaic technology

should be chosen so in order to collect the most energy possible; however, the efficiency is often limited by cost.

Most solar cells are silicon based because the manufacturing process to produce silicon wafers is mature [3]. Cells that are made with other materials that are rare to earth, such as tellurium, can be much more expensive. When selecting a photovoltaic technology, a tradeoff between efficiency and cost should be considered. The size and shape of the photovoltaic should also be considered. Certain photovoltaic technologies use semiconductor wafers, which are very thick compared to alternative thin-film devices, which can be flexible. Thinner cells may be suitable for applications where space and weight limitations are present.

3.2.1.2 Crystalline Silicon Photovoltaics

Crystalline silicon photovoltaics are currently the most widely used photovoltaic cells on the market. This technology represents about 90% of the world's total photovoltaic cell production [4]. These cells have high efficiency and can be connected together under high-transmittance glass to produce reliable, weather resistant photovoltaics. Monocrystalline and polycrystalline silicon are two technologies used to produce the silicon wafers for these photovoltaics. Figure 8 shows a monocrystalline cell on the left and a polycrystalline cell on the right.

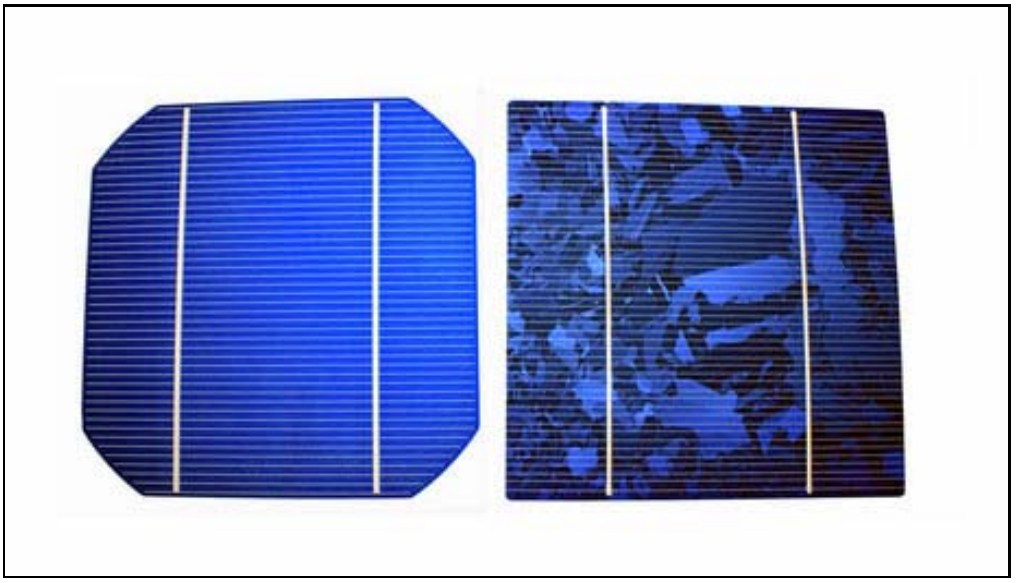

Figure 8 - Monocrystalline (left) and Polycrystalline (right) Solar Cells

Monocrystalline silicon solar cells yield the highest efficiencies, but are also more expensive than other photovoltaic technologies. Standard industrial cells yield efficiencies ranging from 16-18% and high-efficiency cells are capable of efficiencies greater than 20% [4]. The best research cells are currently capable of 25% efficiency levels, though the manufacturing technologies needed to create these cells are

expensive and time-consuming because specialized equipment is needed. The Czochralski process is typically used to grow the crystals and then the silicon wafers are cut from cylindrical ingots.

Polycrystalline, or multicrystalline, silicon solar cells are more common than monocrystalline cells because they are less expensive. Standard industrial cells yield efficiencies ranging from 15-17% and currently make up about 48% of the world's photovoltaic market [4]. They are produced from metallurgical-grade silicon by a chemical purification process called Siemens process and the wafers are cut from square ingots. The efficiency of these cells is less than that of monocrystalline cells due to the presence of grain boundaries, which reduces the overall minority carrier lifetime for the material [5].

Cell module efficiencies for both technologies are about 14%. This is due to the fact that the square polycrystalline cells have a higher packing factor than the pseudo-square monocrystalline cells. Wafers for both technologies range in thickness between 160 and 240 μm. Cell modules containing crystalline silicon cells are relatively thick because of the thickness of the wafer and the glass casing.

3.2.1.3 Thin-Film Photovoltaics

Thin-film photovoltaics are the thinnest and least expensive photovoltaics produced. Unfortunately, they are also less efficient than traditional crystalline silicon photovoltaics. These devices are made by depositing thin layers of photovoltaic material on a substrate. The thickness of the film varies from the nanometer range to the micrometer range. This is much thinner than the silicon wafers used in crystalline silicon photovoltaics. When deposited on a flexible substrate, such as plastic, the photovoltaics can be applied to curved surfaces. Figure 9 shows a thin-film cell that has been deposited on a flexible plastic substrate.

Figure 9 - Flexible Thin-Film Cell

Amorphous silicon photovoltaics were the first thin-film solar cells to be made commercially available. They are made by vapor-depositing a 1 μm thick layer of silicon

on a plastic, glass, or metal substrate [6]. Unfortunately, these cells suffer from the Staebler-Wronski effect which describes the degradation in photoconductivity with prolonged exposure to intense light [7]. This causes the efficiency of these cells to decrease over time until a stabilized efficiency is reached. Typical efficiencies for amorphous silicon modules drop from 10% to 7% due to the Staebler-Wronski effect [6]. Although the efficiencies of these cells are limited, they are still competitive due to their affordability and flexibility.

Cadmium telluride photovoltaics are another kind of thin-film solar cells that solve the efficiency problem present in the amorphous silicon technology. First Solar recently reported its average commercial module efficiency to be 14.7% [8]. This is competitive with the 14% efficiencies found in polycrystalline silicon modules. Cadmium telluride cells currently represent 7% of the world's photovoltaic market [8]. Although these cells yield high efficiencies at a low cost, there are concerns about both the scarcity of tellurium and the toxicity of cadmium. Since this technology is newer and somewhat controversial, it is not as available as amorphous silicon photovoltaics.

3.2.2 Display Screens

Due to the fact that our project will require the use of a display screen to provide a visual source to output data we must make an analysis of different relevant technologies. There are currently two major types of display screen technology. First there is CRT, cathode ray tube, and technology. This is used mostly in older TV models apparent in the 1980s through 1990s. Over the past few years a newer type of TV emerged and quickly taken controlled of the display screen market. This technology is called flat panel display [12]. This analysis will go over the advantages and disadvantages in each of the possible display screen technologies.

The first option for a display screen would be a cathode ray tube TV. Even though the technology used in this device is fairly outdated the price to acquire these devices is extremely cheap and fairly accessible. However actually using this technology would bring a lot of disadvantages. Major disadvantages would be its size, weight, and energy consumption is all considerably large. Building a solar powered blinds project requires optimization in all of these fields.

The next type of display screen would be using flat panel display technology. Flat panel displays are the newer of the two technologies and can be found pretty commonly now. Different models of flat panels exists LCDs, LEDs, and E-Paper displays are all examples of a few types of flat panel displays [14]. Using a flat panel display has many advantages over using CRT. For example, they can be readily found in various sizes, the energy consumption in flat displays is substantially less, and the they are much lighter so they can be easily mounted to a blinds structure.

LCD and LED display screens would be very ideal if the output we choose to display has constantly changing data. This data would need to be changing every few seconds to a few minutes to really benefit from the screen's power consumptions. For instance, if we displayed the time on the blinds, having an LCD or LED screen would be very useful for energy consumption. Reasons why we would need to consider displaying images or data that is constantly changing is because LCD and LED screens all normally have a high refresh rate. This refresh rate is used to update the images on the display. For example, let's consider that a screen is operating at a 60Hz refresh rate. That's means the display is attempting to refresh the screen 60 times per second. If the images are not changing, that is a complete waste of energy. So the previous statement holds that this type of display would be useful if the display is constantly outputting a different variety of data. Another option of LCD/LED screens would be to use a segmented display but this would limit what we can output as visual data [13].

A new and very appealing technology to use is electronic paper displays. These e-paper displays allow data to be retained on screens for several hours even after the power sources are turned off completely. This feature is called "bistable" which, in practice, means the display is only truly consuming power when the display is active (changing or refreshing). This is in contrast to using a LCD/LED screen where even non-changing screens require a minimum refresh rate of 30Hz. The e-paper display will be refreshing at a much slower rate, closer to 0.00007Hz, which will nearly eliminate all power consumption. Adding times when we know the images are worth changing, we could consider even turning off the voltage to the display while it is utilizing the bistable feature [9].

In our project, let's say, for example, we would like to display the charging battery icon, weather status, and temperature status most of the time these images will not need to be updated a substantial amount of times. Using an e-paper display would drastically reduce the energy drain on the system unlike a LCD/LED. The only downside to actually using an e-paper display is there is a slight increase and price, especially if we wanted to consider getting a multi-colored display model. Being that it's a newer technology the average price is higher than something of similar standards like an LCD/LED.

A comparison of various electronic display technologies can be seen in the chart in Figure 10, reprinted with permission from Persuasive Displays INC. For this project, we initially decided to go with an e-paper display for practical reasons. Since the overall goal of this project is to conserve energy and provide an alternate renewable energy source, having such a low-power-consuming display such as e-paper was very appealing. The specific e-paper display we wanted was the Persuasive Displays 2.7 E-Paper display. The specific Digikey part number is E1270CS021-ND and its price before tax is $21.72 [11].

	E-ink	TFT-LCD	OLED	TN-LCD	Ch-LCD
Mechanism	Charged particles movement in capsules	Twist of liquid crystal	Emissive layer of organic compounds	Twist of liquid crystal	Twist of cholesterol liquid crystal
Bistable	◎	X	X	X	○
Reflectivity	◎	△	X	△	△
Power	◎	X	X	△	◎
Contrast Ratio	○	◎	◎	△	X
Response Time	△	○	◎	○	△
Color	△	◎	◎	○	△
Viewing Angle	◎	○	◎	△	△

◎:Great ○:Good △:Acceptable X:Bad/don't support

Figure 10 - Comparison Table of Display Technologies

Although the specifications of the e-paper display are exactly what we wanted to find in a display for our project, once development began, we realized that there was a large concern with e-paper. Specifically, there was a considerable lack of documentation for this technology, even in its own datasheets from multiple e-paper manufacturers. We saw that the e-paper had a 40-pin I/O connection, but there was no information on what each pin was intended to do. Because of the uncertainty that this lack of documentation presented, and the time constraints we faced, we decided to explore other options.

After we had ruled out e-paper as a viable option for our project, we looked into LCD. The reason we had ruled out LCD before was because there was a concern that backlighting would be a very large power drain. After more research, we confirmed that the backlighting is the most power-hungry feature of LCD screens. Fortunately, we were able to find some screens that allow us to disable the backlighting. This significantly reduced the power requirements of the display to the point where it became a good option for our project.

We chose to go with a 16x2 character LCD model NHD-C0216AZ-FN-GBW made by Newhaven Display International, shown in Figure 11. This screen has plenty of space to display information for both the battery charge and temperature measured from our probe.

Figure 11 – 16x2 Character LCD Screen

We were able to find this LCD screen at Digikey for only $9.11. More details on the specifications can be seen in the data sheet in Appendix D. One nice difference between this and the e-paper display is that the LCD only has 14 pins, instead of the e-paper's 40 pins, and the documentation clearly describes the function of each one. The data sheet also includes a sample initialization program which will make the embedded programming much easier. As a bonus, it also supports Japanese characters, so we could add a language change option if we so desired.

3.2.3 Motors

3.2.3.1 Motors Overview

We have been looking for used motorized blinds to see if there is a way to take the motor out of an old set and use them for our project. The reason for this is because we assumed it would be much cheaper to just take apart an old set of blinds and use the motor, than to pay for a brand new motor. However, as we have been conducting research on these, it seems that this option is approximately the same price and some can be quite expensive. The roller blind motor seems to be the most commonly used motor and seems to be the most inexpensive.

These motors seem to range in price anywhere from $70 to over a $100. This seems to be a bit expensive for a motor when you could just buy the whole blinds assembly for about $120. There are also motor kits that are sold online that are about a total of $115, which allow a user to make and install their own motor.

Originally, we had theorized that a vertical slat design would be the easiest and most effective to implement for our project. A concept rendering of this design can be seen in Figure 12, courtesy of BlindsParts. With this in mind, we started working on the design of our project based on this format.

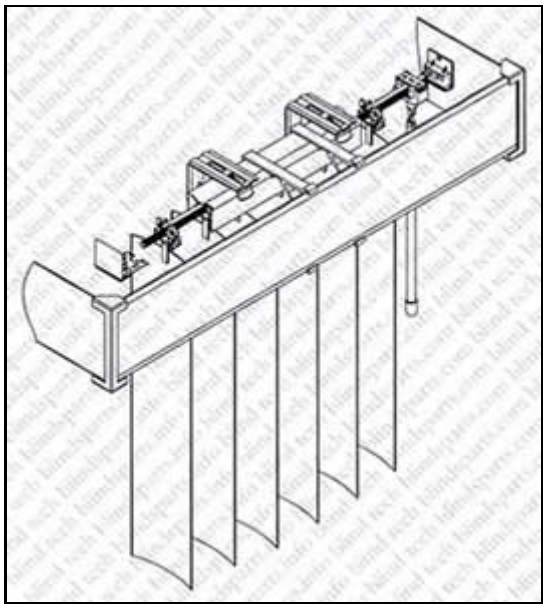

Figure 12 - Vertical Blinds Concept

The simplest solution that we found through research, is to just get two small D/C motors, that are pretty inexpensive and will have enough torque to rotate the slates and then slide them all away from the window. One of the D/C motors will be to spin the blinds to open the slats. The second motor will be to draw the blinds open all the way. The motors will have a limit switch that will let the motor spin only a certain amount to stop the blinds from overturning or opening them too far.

Just about as we had decided on the blind and motor layout, we saw another architecture that seems to be the most plausible way to go. This architecture uses a pull down blind shade which can be automated by a single motor. This architecture provides multiple benefits for our design. First, it will significantly simplify the process of attaching the motor(s), which was a concern for us since none of us are mechanical engineers. Second, it will cut the price in half as we will only have to use a single motor instead of two. Third, since we are using fewer parts, it will lessen the overall weight of the blinds. And finally, since we would be using a canvas instead of slats, the weight of the window covering would be lighter; a lighter canvas means less energy required by the motor to raise it. Figure 13 depicts an example of the assembly anticipated having for our project design.

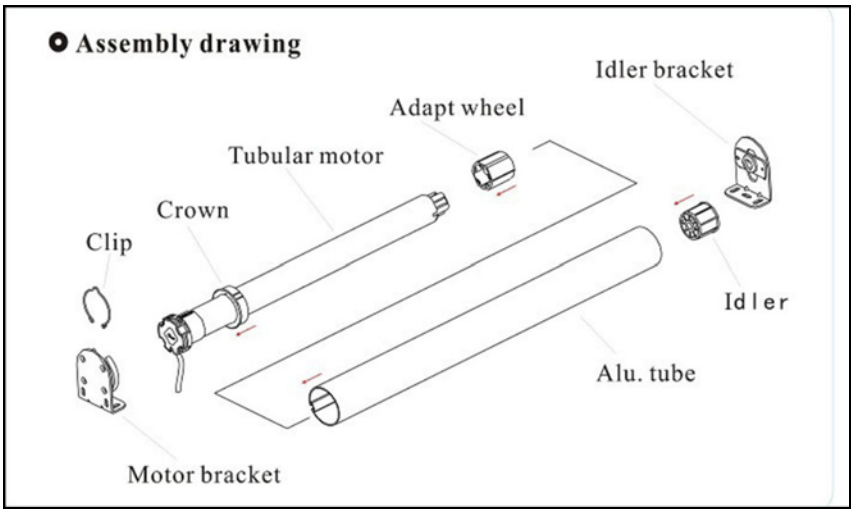

Figure 13 - Motor for Rolling Shades Architecture

3.2.3.2 Servo Motors

We also looked into another motor option while we were still considering using vertical slats. We looked at a servo motor, but it seemed to be a bit of overkill. The price for this motor seemed financially unaffordable. This motor is retailed at $620. This is very high compared to some other options. The motor is definitely more than enough to get the job done though. It has enough power to turn the blinds and to open the blinds all of the way.

The motor has the some of the following specs. The power that is used to run this motor seems to be way too unattainable. It requires 400W of power to run. The rated voltage for this part is 150V. This motor has a rated speed of 5000 RPM. This is more than enough for this project especially with a peak speed of 8000 RPM. The torque is much more than believed to be needed at 20 in-lbs [15].

The motor seems to be a bit heavier than what would be desired. It weighs in at a little bit over 3.5 lbs. This is a little too heavy since there will be two of the motors in these blinds, as one motor would be to turn the blinds, and one would be to open the blinds all of the way. The motors need to be mounted in a fixture on the wall which means that a lighter motor would be desired as to not stress the drywall.

After looking at all of these specs on the servo motor, it is apparent that it is not the right motor for this project. It is too big and bulky, is very unaffordable, and uses too much power for our solar powered battery.

3.2.3.3 DC Motors: Leeson Motors DC Motor-.05 - .1HP, 12-24V, 1750-4200RPM, TENV, Sq. flange

This next motor is a DC motor that seemed to be a bit more affordable than the servo motor. This motor is priced at approximately $135. This motor also offers a square mounting surface on the sides of it that make it much easier to mount inside of the housing. This motor has a horsepower of 0.5 to 1. This is, once again, more than enough to do the job we need it to in our project. The revolutions per minute are 1750 [16]. This is still overkill on the blinds. The blinds can open much slower, if needed, to save energy. The working voltage is also much more attainable because the voltage required to run this is in the range of 12V to 24V. A drawback on this motor that was the same as the servo motor is that the motor weighs 4 lbs. Adding 8 lbs into the fixture hanging from the wall is going to loosen the mounting mechanism over time much faster than a lighter motor would.

This motor is also very large. It is almost half of a foot long. This is another drawback because we did not want the fixture on the wall to be too bulky. This motor having been still too expensive, too large, and weighing too much was not a motor that would be used for our project.

3.2.3.4 DC Motor – 13800 RPM 3VDC 350mA

This motor seemed to have a lot more promise than the previous motors that have been researched. This one is much more affordable to say the least. It is also retailed at only $1.39. The only issue that may be ran into with this motor is that it may not have enough torque, but this will have to be determined later on in the testing procedure.

3.2.3.5 12V Motor with Drive Shaft

A drive shaft motor seems to be the closest to what we have been looking for. It is designed specifically for motorized blinds and is used to roll up the blind shade for a rollable blinds architecture. The housing on the motor is a long tube and it is a relatively low-profile housing. It seems to be pretty easy to install and looks like it will get the job done. It is a 12V, low powered motor, which is exactly what we were looking for since we are running our entire project off of a battery. The smaller the amount of power used, the better. Also, this motor is rated to pull up a load of up to 6 lbs.

This seems to be way more than enough for a thin blind shade with a thin-film solar panel. The torque rating on the motor is .7 nm which seems to be a perfect amount for this project. It is also able to rotate at 34 RPM. The standby current for the motor is 18mA and the most current it can possibly draw is 1200mA, which is when it stalls [17]. The normal current draw when in operation is approximately 800mA. This is definitely viable since we will have a 60W-h battery and the motor will only be active for brief, and sporadic, periods of time. The tube length for the motor comes in any size from 12

inches and up. This definitely seems to be the right motor for us and our needs for this project. A breakdown of parts for this motor can be seen in Figure 14, courtesy of RollerTrol Automation Systems.

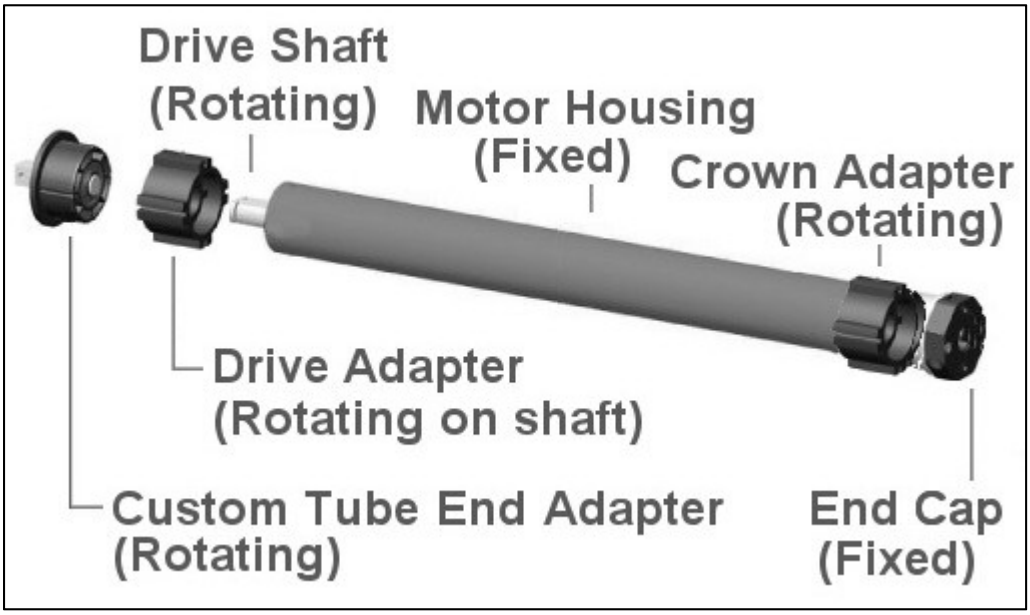

Figure 14 - Drive Shaft Motor

After a little further design into this rolling architecture some more, we decided that we would purchase this motor for our project. We ended up buying it from RollerTrol Automation Systems for $118.00. Although it is only 14.5 inches long, it can be extended by putting a longer PVC pipe over it. The actual product, with a PVC pipe acting as an extender is shown in Figure 15. It is top-mounted, and is mounted inside the housing. There was a concern as to how much space the drive shaft, PVC pipe, canvas, and solar panel would consume in the limited volume we have inside the housing, but we were unable to know for certain until all the products were physically in hand.

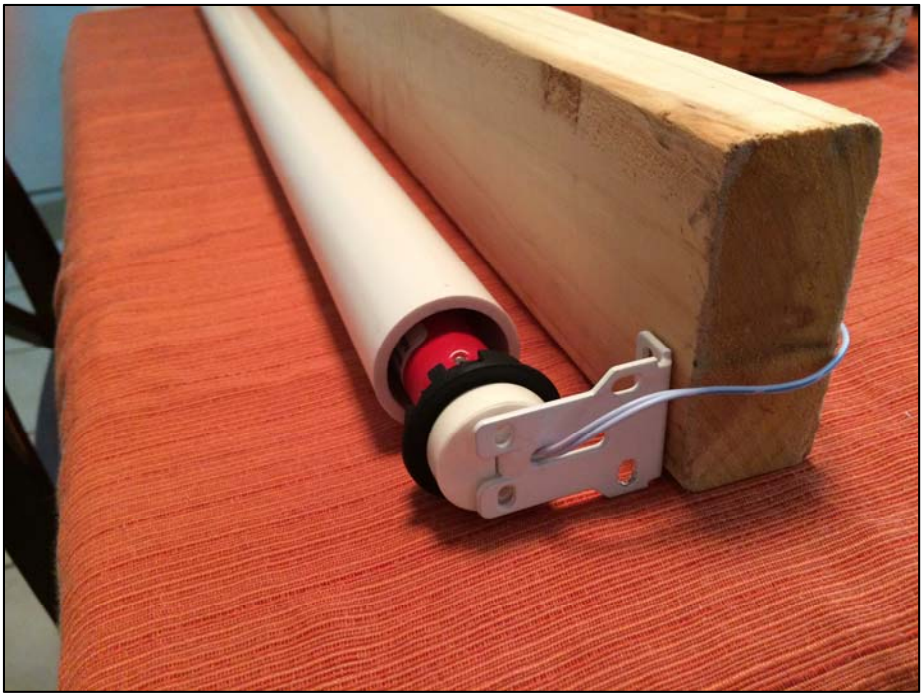

Figure 15 – Drive Shaft Motor that was Purchased

3.2.4 Remote Connectivity

A wireless connection was required to communicate between our device and a mobile phone equipped with our application. There were several different options available to us to accomplish this goal, and we primarily considered: Wi-Fi, Bluetooth, and Zigbee [18]. Although power consumption is the primary focus of our project, we also needed to take into account other requirements. We looked into metrics for each connection method such as range and bandwidth as well. These metrics have been gathered and compiled into Table 5 for easy comparison [19].

Connect Method	Active Power Consumption	Idle Power Consumption	Max Range	Bandwidth
Wi-Fi	750 mW	negligible	32-95 m	54 Mb/s
Bluetooth	100 mW	negligible	10 m	1 Mb/s
Zigbee	80 mW	negligible	10 m	250 kb/s

Table 5 - Comparison of RF Specifications

3.2.4.1 Wi-Fi

Common Wi-Fi frequency ranges are around 2.4 GHz and 5 GHz. The main benefit in the 2.4 GHz system is that it has a longer wavelength meaning it can bend around corners a

little better than its 5GHz equivalent. This makes it good for large homes or apartment buildings. However, since the 2.4 GHz standard is more common, it is prone to interference it here are multiple other routers using the same frequency within a certain proximity. It is also prone to interference from other devices such as cell phones.

If we choose to use a Wi-Fi chip on our microcontroller, it will require the user to purchase and set up their own wireless router. Also, it would take significantly more power to operate than either Bluetooth or Zigbee options. It also takes additional steps to set up through a router. The main benefit of this is the significantly longer range. If it's connected to a router, it can potentially be accessible from anywhere that has a wireless connection. However, the power drain from Wi-Fi does not make this a good option.

3.2.4.2 Bluetooth and Zigbee

Bluetooth and Zigbee are both very similar in terms of power consumption, range, and security protocols. However, with Zigbee being the newest standard to enter the remote connectivity lineup, there are still problems to be worked out. Some of these problems include interoperability [20] and a lack of development resources.

Bluetooth has a new standard they refer to as Bluetooth Low Energy (BLE). BLE has a bit lower range and bandwidth than traditional Bluetooth technology, but we will not be requiring frequent or large data transfer between the MCU and mobile application. We also expect it to be used from within the same room as the blinds. Additionally, Bluetooth technology has been around for many years, so there are ample resources available on the web to assist us in software development. For all of these reasons, we will choose to go with a Bluetooth connection for convenience and power consumption.

3.2.4.3 RFID

Our project will require a "sync" feature that will allow the user to register the device with their mobile phone in order to control operation of the blinds with the mobile application. One of the methods being considered for this synchronization is a radio frequency identification (RFID) tag. The tag would be located inside the control housing for the blinds and the phone would act as the RFID reader.

A product that uses RFID to connect to smart phones is Assa Abloy Hospitality Mobile Access (formerly VingCard Elsafe). This technology allows the user to make a reservation at a participating hotel location through their mobile app. When the user arrives at the hotel, they can just go straight to their room where an RFID-compatible locking mechanism is installed on the door. The user simply has to put the phone in proximity to the door to unlock the door and the check-in process is completed automatically [23].

The main focus for us in Assa Abloy's product is its compatibility with RFID technology. Specifically, it uses near-field communication (NFC) which utilizes the high frequency band of RFID [21]. This requires very close proximity between the tag and the reader, approximately 4 inches. Additionally, NFC communication can go both ways since either device can act as a tag or reader. This may allow us to display a "synced" notification on the integrated display screen on the blinds.

Additionally, smart phones that are NFC compatible can be used as electronic wallets. There are an increasing number of retail checkout card scanners that support the use of NFC readers to pay for products. This is usually indicated by a prompt similar to, “Tap or swipe your card”. The “tap” portion of this prompt makes use of NFC built into a chip on certain credit cards. The technology in compatible smart phones can be used in the same manner.

Compatibility might be a problem for using NFC technology with our mobile phones. However, cell phones from most major manufacturers such as Acer, BlackBerry, HTC, LG, Motorola, Nokia, and Samsung have come equipped with NFC capabilities for several years now [22]. One notable exception to this are that all iPhones older than the iPhone 6. This is not a major setback for us since we are currently only planning to support Android devices with our mobile application. Also, NFC is supported after Android 2.3.3, SDK version 10, which was released on February 9, 2011. As we can see in Figure 16, the amount of people that would not be able to use this feature is only 0.3% (Froyo) of Android users (as of June 1, 2015) [24].

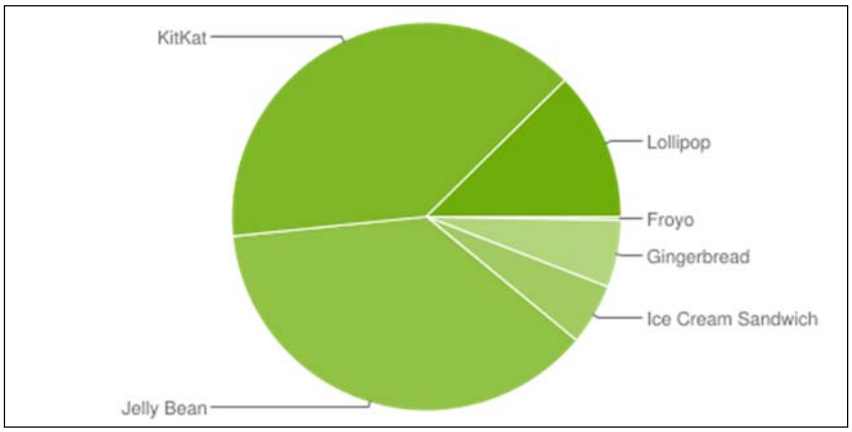

Figure 16 - Android API Distribution as of June 1, 2015

One feature of the Android API is that a setting can be specified to only allow Google Play to display the application to users with NFC-compatible phones. However, we would still keep it in Google Play even for non-compatible phones for publicity.

3.3 Possible Architectures and Related Diagrams

3.3.1 Microcontroller Units Architecture

There are several possibilities when deciding on a microcontroller unit. The first consideration is the type of architecture to use. At the highest level, the possible MCU architectures we have to consider are the Von Neumann architecture and the Harvard architecture, with Von Neumann being the older of the two. The main difference is the way the memory is set up [26].

Getting more specific, some common MCUs are MSP430, 8051, PIC, AVR, and ARM. We want to use the newer Harvard architecture for this project because it is the more popular of the two in modern microcontrollers, and it would provide us with more relevant experience for future projects. MSP430 uses Von Neumann architecture, so we will rule that out as one of our possibilities. Older ARM units use Von Neumann as well [24]. This leaves us with 8051, PIC, AVR, and newer ARM processors.

Out of these four remaining architectures, AVR and ARM are the newest. As a result, they are generally faster than the 8051 and PIC MCUs [28]. Additionally, 8051 and PIC have small stack space and can be a bit challenging for C programming [29]. Assembly programming is still viable, but would take a bit more time.

Focusing on AVR and ARM, we can see that both are very good options. There are multiple free compilers available for both, and they're both very widely used in modern applications. However, AVR is designed specifically for use with video codecs and for use in devices such as Apple's iPod. ARM on the other hand, has a wide range of uses and is heavily used in smart phones. As a very flexible and powerful option, we choose to go with an ARM MCU. Also, since ARM processors are used in a wider variety of products in the current economy, this would provide us with good experience which is highly probable to be used in future applications.

The ARM processor has more processing capabilities than we require for this project. If we were developing this product for actual marketing for a company, I would probably not use the ARM because of the (slightly) extra power consumption and additional cost. However, we want the learn how to use the ARM processor to allow us to explore more career opportunities in the future.

3.3.2 Motors Architecture

Eagle and LtSpice will be the programs that we will be using to design our schematic for our board. Each part will have its own library with its own package, device and schematic symbol. We will be aiming to use 603’s as much as possible because the cavity in the housing is relatively small, so we will try to make the board as small as possible. There will not be very many parts on the physical PCB layout since we will be using a battery controller externally to charge the battery from the solar energy. The only circuits on the board itself will be the regulation circuit from the battery to the

necessary voltage to operate the micro and to charge any USB device that will be plugged in as well as operate the motor.

3.3.3 Power Distribution Architecture

The basic power distribution layout is shown below in Figure 17. The solar module is the only power input into the battery. A solar charge controller was necessary in order to maximize the amount of power being input into the battery and also to provide protection circuitry for the solar module input and load outputs. The loads include the roller blind motor, microcontroller, USB charger, and e-paper display. Appropriate circuitry was required to deliver the correct amount of power to each sub-system and to protect each load from receiving too much power. Since the solar module would only charge the battery during the day, the loads were chosen to draw as little power as possible while still accomplishing the desired function.

In order to monitor and report the current charge on the battery, we required a battery monitor to accomplish this task. We found a very affordable battery monitor from Texas Instruments, model BQ34Z100-G1. It is compatible with many different batteries including Li-Ion, LiFePO4, PbA, NiMH, and NiCd and a voltage range of 3V-65V.

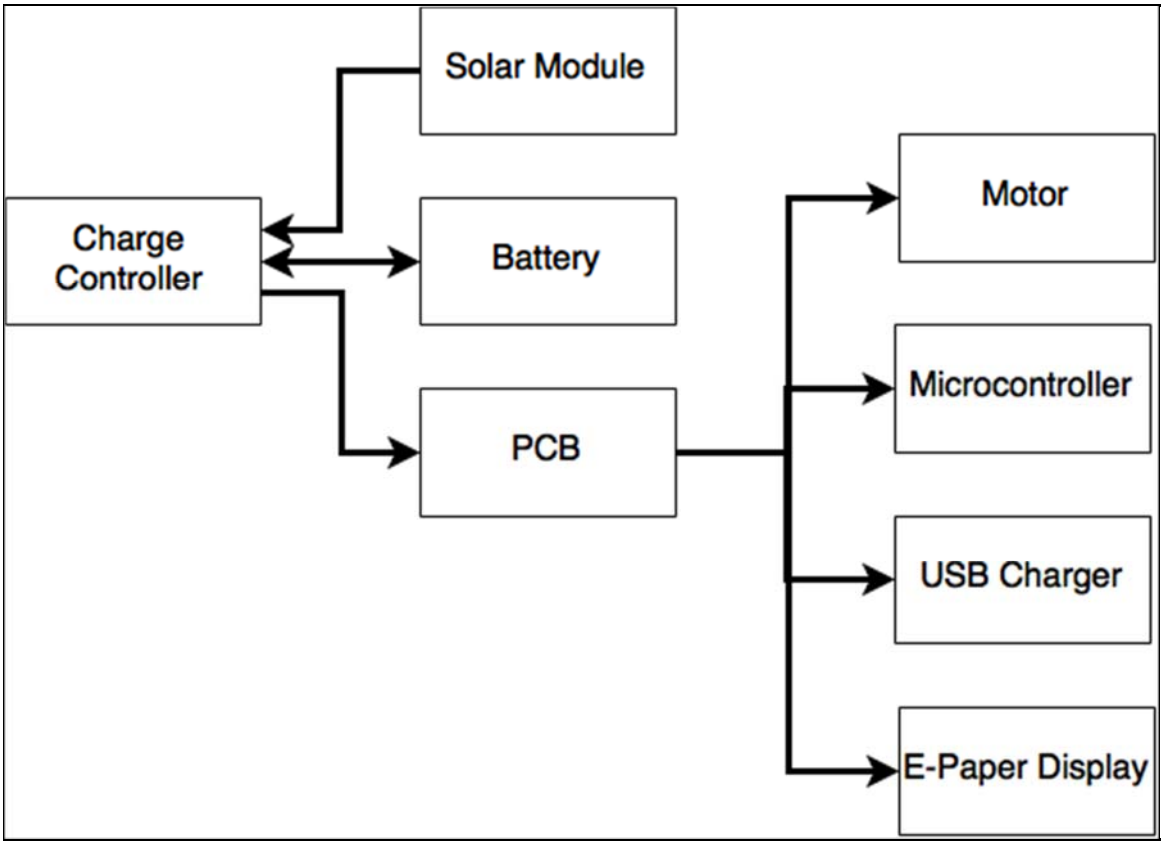

Figure 17 - Power Dissipation Architecture

3.3.4 Battery Options

When selecting a battery to use for solar energy storage it is important to consider the cost, lifetime, size, environmental impact, and reliability of the different technologies available. A deep cycle battery is required for applications where the battery will be regularly discharged at a low rate over several hours. This is in contrast with a shallow cycle, or starter, battery, which delivers a high power pulse over a short period of time. The battery selected for this project needed to be a deep cycle battery because USB charging requires the battery to be discharged at a low rate. Two battery chemistries that are commonly used for solar energy storage are lead acid and lithium-ion. Lead acid batteries are the best option when working with a limited budget because they are more affordable than lithium-ion batteries.

Lead acid batteries can be divided into two main types: flooded and sealed. Flooded lead acid batteries are more affordable than sealed lead acid (SLA) batteries; however, they require upright orientation to prevent leakage, a ventilated environment, and routine maintenance. SLA batteries can be further divided into Gel and absorbed glass mat (AGM) categories. AGM batteries are more common because they offer higher current rates and display long life cycles when discharged to less than 60% of max capacity between recharges [33]. They are also less expensive than Gel batteries. Gel batteries have longer life cycles than AGM and perform better in higher temperature environments.

Lithium-ion batteries are the best option when a high life cycle is desired. They are more expensive than lead acid batteries; however, they are lightweight and less sensitive to variations in temperature. The Powerizer 12V 5A-h LiFePO4 rechargeable battery costs $79.95. This battery has a built in PCB to protect it from being overcharged (>15.6V), over discharged (<8.8V), or over drained (>40~60A). An LED indicator is also built in, which shows the capacity status. The battery is rated to have a cycle life of more than 1000 cycles. The maximum charging rate is 2.5A and the maximum discharging rate is 10A. The battery measures 3.5”x2.8”x3.98” and weighs 1.48 lbs. It has T1 terminal connectors.

The Universal Power Group’s rechargeable battery, UB250, is a 12V 5Ah SLA AGM and costs $9.93. The battery is rated to have a cycle life of 200 cycles when the battery is completely discharged (0%), 500 cycles when the battery is half discharged (50%), and over 1200 when the battery is discharged about a third discharged (30%) before recharging. The lifetime for this battery is shown graphically in Figure 18, printed with permission from UPG. The battery measures 3.54”x2.76”x3.98” and weighs 3 lbs. It has F1 terminal connectors.

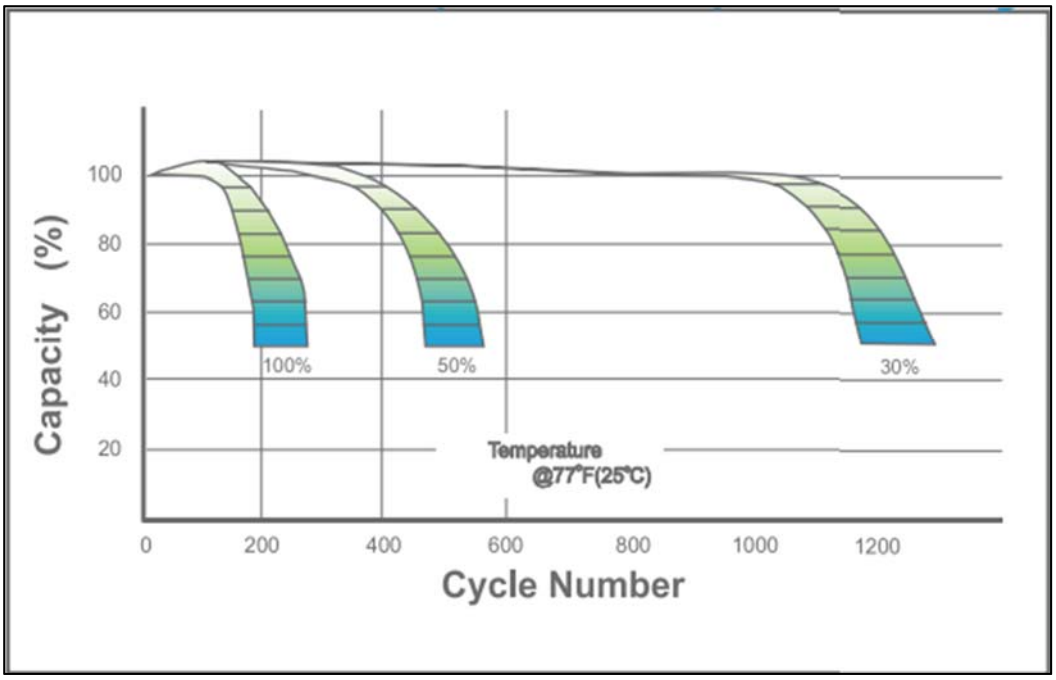

Figure 18 - Cycle Life vs Depth of Discharge for UB250 Battery

The lithium-ion battery is about eight times more expensive than the lead acid battery with similar specifications; however, there are several safety features that are built in to the lithium-ion battery that are not included in the lead acid battery. Since a solar charge controller was planned to be used in our blinds regardless of the battery chemistry, the added circuitry of the lithium-ion battery that provides protection from overcharging, over discharging, and over draining was unnecessary. There is no indicator on the lead acid battery to show the capacity status; however, a simple battery monitor chip could be implemented to keep track of this information.

The cycle life for the lithium-ion battery is about twice that of the lead acid battery when the battery is discharged 50% before it is recharged. If the lead acid battery is only discharged 30% before it is recharged then the lifetime is about the same as the lithium-ion battery. If the battery will be discharged completely every night then it would be most beneficial to spend more upfront on the lithium-ion battery. If the battery will be discharged less than 50% before it is recharged, then the lead acid is a much more affordable option. Even if the lead acid battery needed to be replaced up to seven times, in the time that it would take the lithium-ion battery to retire it would still be the more economical option; however, it would have a larger environmental impact. It is important to correctly recycle batteries of all chemistries in order to reduce the environmental impact.

Later into the development of our project, we realized that the higher self-discharge rate of a lead acid battery might cause problems. However, after investigating further into this issue, we found that the self-discharge rate of the lead acid battery we selected

was still within an acceptable range for our application. The details for the self-discharge rate of our lead-acid battery can be seen in Table 13, below, courtesy of Universal Power Group [53].

3 Months	6 Months	12 Months
91%	82%	64%

Table 6 – Self Discharge Rate (i.e. Shelf Life) of Sealed Lead-Acid Battery at 77°F

The self-discharge rate is relatively significant compared to some alternatives such as lithium ion or nickel metal hydride. However, there are a few other factors that lead us to decide to continue with the lead-acid battery. Switching from lead-acid to a different type of battery would require us to get a new charge controller too, which would cost a significant amount of money. Our project is only using this battery in order to demonstrate its various functions, and is not intended to be the main battery source for the product, as explained in the goals laid forth for the project. As described in Section 1 of this document, the product is intended for use with a large centralized home battery. Besides these considerations, there is also the fact that this discharge rate is still low considering that it will be recharged on a daily basis. Due to the low capacity and energy consumption of our product, the battery will almost always be at maximum capacity unless the USB charger is heavily utilized. This emphasizes the fact that these blinds are intended to be used as part of a larger renewable energy home power system, and that the final state of this project is simply to demonstrate its capabilities.

3.3.5 Power Input

3.3.5.1 Solar Module Options

There are a variety of solar cell technologies to choose from when designing an application that will collect solar energy. These include monocrystalline silicon, polycrystalline silicon, and amorphous silicon photovoltaics. Crystalline silicon photovoltaics are non-flexible and would need to be installed to blinds having large vertical or horizontal slats. The advantage to using crystalline silicon photovoltaics is that they yield higher efficiencies. This means they will provide more power to the battery for every hour of incident sunlight using the same amount of space. The size of each module would need to match the size of each slat. The modules could be connected in series or in parallel in order to increase the output voltage or current of the system. Amorphous silicon photovoltaics are flexible and can be attached to roller shades, but are less efficient than crystalline photovoltaics. Several larger modules could be used and attached directly to the back of the shade. This would reduce the complex wiring necessary when using crystalline photovoltaics.

ML Solar carries a set of forty 3”x6” polycrystalline photovoltaics for $33.95. These cells are shown below in Figure 19, printed with permission from ML Solar. Each cell has an average power of 1.8W operating at a maximum voltage of 0.5V and a maximum

current of 3.6A. These cells can be connected in series in order to achieve the higher voltage necessary for charging a 12 V lead acid battery. A voltage of 13.5 to 13.8 V is recommended for charging a 12V lead acid battery; however, the charge controller would regulate the voltage to match that of the battery if the voltage of the solar module exceeds that of the battery. Using thirty of these cells connected in series would provide a total average power of 54W operating at a maximum voltage of 15V and a maximum current of 3.6A.

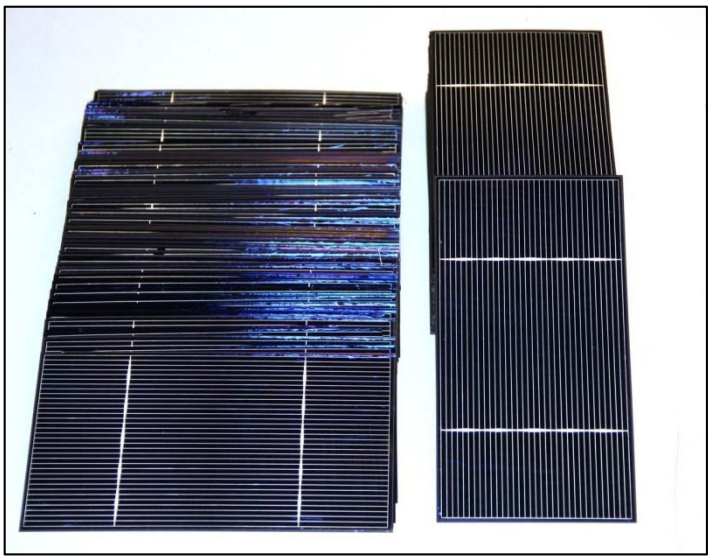

Figure 19 - ML Solar 3”x6” Polycrystalline Cells

Blocking diodes would need to be used in order to prevent the current from flowing back into the cells. Bypass diodes would also be needed in order to bypass a cell that is shaded. Figure 20 shows how the cells can be connected in series when using the blocking and bypass diodes.

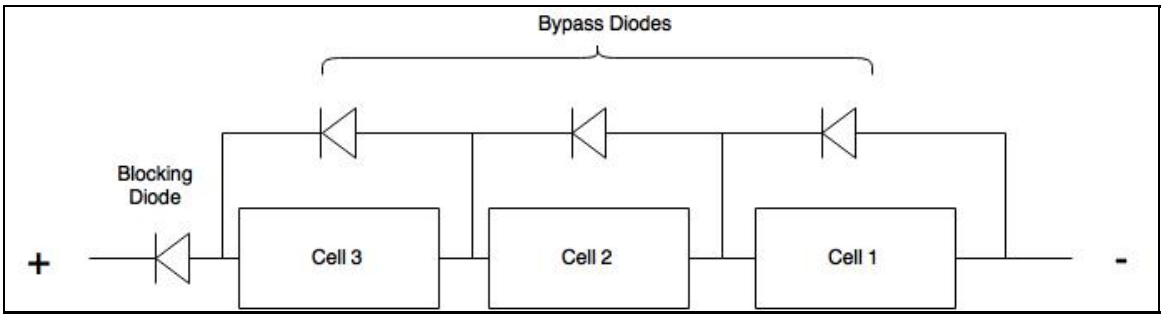

Figure 20 - Connecting Solar Cells in Series

Five cells can be connected in series on each of six vertical 6” slats. This would create six 9W modules that can be connected in series to form the final 54W module. This module would be capable of capturing the amount of energy capable of being stored by the 60W-h battery in about 1.1 hours under ideal lighting conditions. While the amount of power capable of being captured by these cells far exceeds that of the amorphous

silicon cells, the wiring required would be more intricate and blinds with large slats would need to be used, increasing the weight and power required by the motor in order to move them.

Sunpower carries a set of 20 6”x6” monocrystalline photovoltaics for $89.99. These cells are shown below in Figure 21, printed with permission from cicibizhk (eBay). Each cell has an average power of 3.3W operating at a maximum voltage of 0.58V and a maximum current of 5.93A. Similar to the polycrystalline cells, these cells can be connected in series in order to achieve the higher voltage necessary for charging a 12V lead acid battery. Using 18 of these cells connected in series would provide a total average power of 59.4W operating at a maximum voltage of 10.4V and a maximum current of 5.93A. The charge controller would need to be able to boost the voltage and drop the current in order to safely charge the battery. Three cells can be connected in series on each of the six vertical 6” slats. This will create six 9.9W modules which can be connected in series to form the final 59.4 W module. This module would be capable of capturing the amount of energy capable of being stored by the 60W-h battery in one hour under ideal lighting conditions.

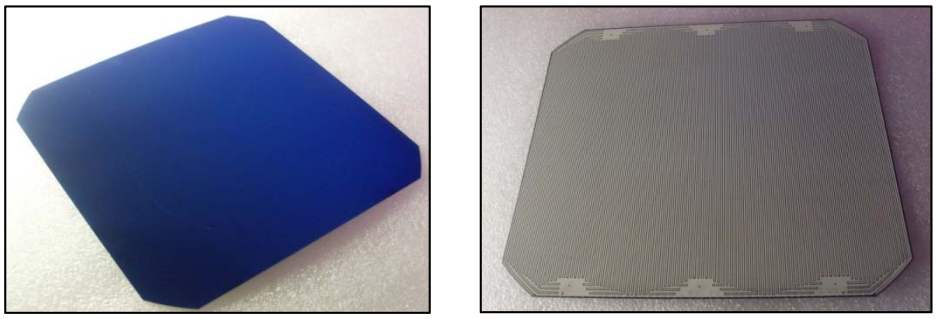

Figure 21 - Sunpower 6”x6” Monocrystalline Cells

Powerfilm makes a series of rollable, amorphous silicon panels, which come in a variety of sizes. They are made using a proprietary roll process, which makes them much more flexible then other thin film technologies. The R-14 model is 14.5”x42” and costs $185.99. This panel is shown in Figure 22, printed with permission from FlexSolarCells. The panel has a total average power of 14W operating at a maximum voltage of 15.4V and a maximum current of 900mA. A blocking diode is built into the unit to prevent back flow of current. If more power is desired, panels can be connected in parallel to achieve a higher current. The voltage of the panel is ideal for charging the 12V lead acid battery being used in our project. This module is capable of capturing the amount of energy capable of being stored by the 60W-h battery in 4.29 hours under ideal lighting conditions. Although the panel is more expensive and less efficient than the crystalline silicon modules, the rollable nature of the product allows for it to be easily attached to a roller shade. This drastically reduces the amount of solder connections needed, which reduces the amount of failure points in the system. The panel also weighs 0.98 lbs, which does not put much stress on the motor used to roll the shade.

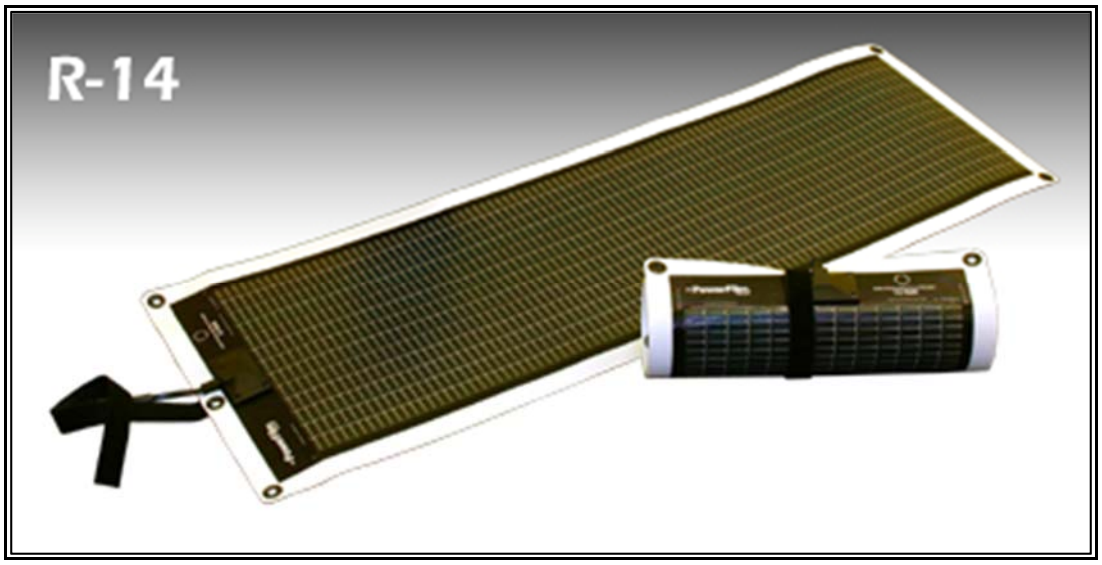

Figure 22 - Powerfilm R-14

In order to maximize the power capable of being captured by the system while maintaining the roller shade design, a combination of crystalline solar cells and amorphous silicon solar panels is used. The 3.3W 6”x6” monocrystalline photovoltaics are attached to the back of the housing unit used to store the battery and circuit boards.

Connecting six of the cells in series produces 20.6 W operating at a maximum voltage of 3.48V and a maximum current of 5.93A. Adding the 14W amorphous silicon rollable solar panel brings the total power of the system to 34.6W. This system will be capable of capturing the amount of energy capable of being stored by the 60W-h battery in 1.73 hours under ideal lighting conditions. A diagram of the system described is shown in Figure 23.

The only tricky part of this configuration is that we needed to connect the rollable panel with the hard panels. Because the panels would be receiving varying amounts of power over time, we were concerned about some current flowing from one panel and into the other, rather than into the charge controller. To fix this, we added a boost converter to the hard panels to match the voltage of the rollable panel. The rollable panel has a blocking diode built-in so we do not have to worry about that one, but we added a diode to the output of the boost converter from the crystalline panels. After that, we were able to easily connect them in parallel and feed the output into the charge controller to be forwarded on to the battery.

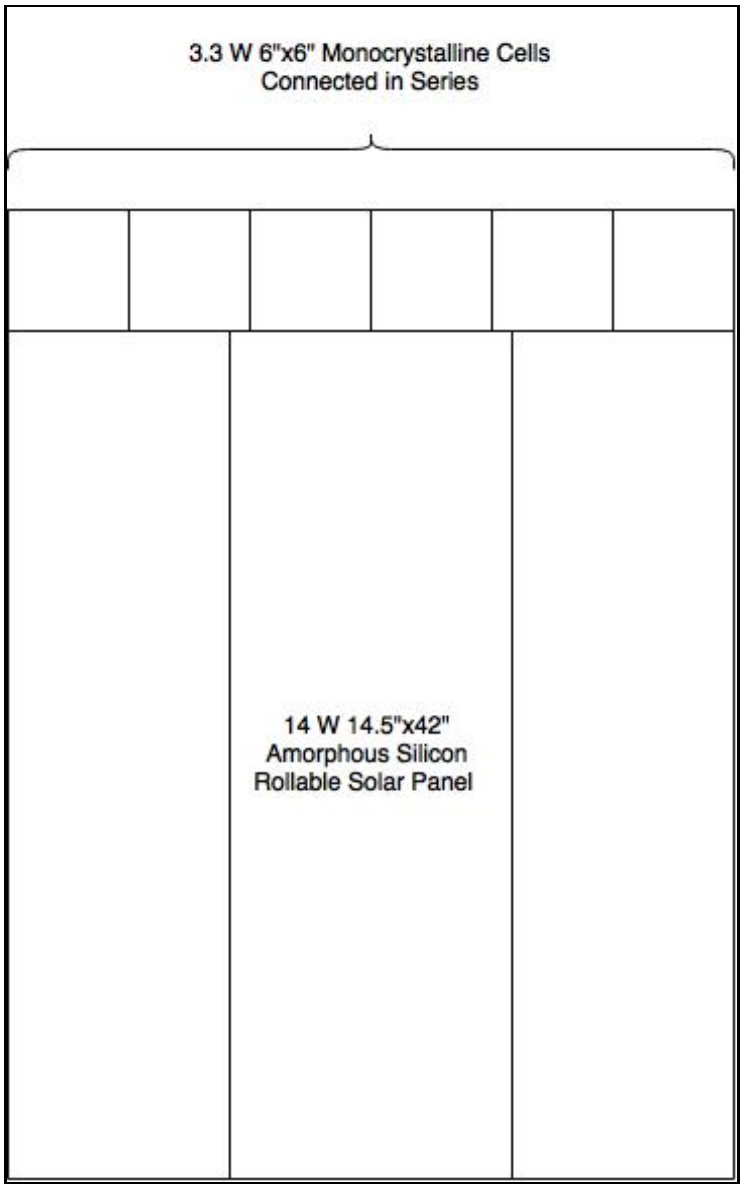

Figure 23 - Crystalline and Amorphous Silicon Design

3.3.5.2 Maximum Power Point Tracking

Maximum Power Point Tracking (MPPT) compares the output voltage of the solar module to the terminal voltage of the battery. From this information, the maximum power that can be output by the solar module into the battery is calculated. The optimized voltage is then selected in order to maximize the current flowing into the battery. The efficiency is increased most when temperatures are below average. A 20-45% power gain in winter and a 10-15% power gain in summer can be achieved when an MMPT is used [30].

The MMPT circuit is a high frequency DC-to-DC converter. It takes the DC input from the solar modules and converts it to high frequency AC signal. It then runs the AC signal

through a transformer and a rectifier to convert it back to a different DC voltage and current that matches the battery voltage. Since light and temperature conditions vary continuously throughout the day, most MMPT charge controllers are now digital microprocessor controlled devices. Several companies make MPPT charge controllers with varying costs, peak efficiencies, speeds, max panel voltages, battery voltages, and max load currents.

The Genasun GV-5 is a 65W 5A MPPT solar charge controller for lead-acid batteries. The controller costs $75.00 and has a maximum recommended panel power of 65W. Inputs include a 27V maximum panel voltage input, a 12V battery input, and a load input with a continuous rated load current of 5A. Although it is rated for 27V, a 22V maximum panel voltage is recommended under standard operating conditions. The minimum battery voltage for normal operation is 7.2V and the maximum input current is 9A. A computer controlled four-stage battery charging profile is used in order to increase the battery life and maximize capacity. This includes an absorption voltage of 14.2V, a float voltage of 13.8V, and a load disconnect voltage ranging from 11.4V to 12.5V. A battery temperature compensator is also included which adjusts the voltage at -28mV/℃. The controller utilizes an MPPT tracking speed of 15Hz in order to adapt quickly to changing light conditions. This results in an electrical efficiency of 96%-99.85% and a tracking efficiency of over 99%. The controller uses ceramic components instead of electrolytic components in order to increase the product lifetime. A 10-year warranty is included with purchase. The GV-5 consumes 0.150mA when operating and 0.125mA when asleep. It weighs only 2.8 ounces and has dimensions of 4.3”x2.2”x0.9”. A built-in electronic protection circuit cuts power when a short circuit is detected. This will protect the controller from damage if the polarity of any of the inputs is mistakenly reversed.

Genasun also makes the GV-4 and the GV-10, which are similar controllers that are designed for smaller and larger panel powers respectively. The GV-4 costs $65.00 and has a maximum recommended panel power of 50W. The GV-10 costs $109.00 and has a maximum recommended panel power of 140W. Inputs include a 27V maximum panel voltage input and a 12V battery. There is no load input on either of these controllers. The warranties on the GV-4 and GV-10 are five years, which is half the warranty of the GV-5. All of the other characteristics of these charge controllers are identical to those of the GV-5 [31].

3.3.5.3 Pulse Width Modulation

A pulse width modulation (PWM) charge controller is less expensive than an MPPT controller and is more ideal for small systems. It can be used if the solar module voltage is matched to that of the battery under normal operating conditions. Unlike the MPPT controller, which operates at above the battery voltage, the PWM operates at the battery voltage. It performs best in warm weather and when the battery is near full [32].

In the PWM charging method, the charge controller sends out a short charging pulse of energy to the battery. The controller then checks the state of the battery to determine if more pulses should be sent and how wide the pulses should be. If the battery is fully discharged, the controller will send out longer pulses at a steady rate. When the battery is nearly full the pulses will become shorter and shorter. The controller checks the charge of the battery between pulses and adjusts the pulse accordingly.

The Morningstar SS-6L-12V is a 12V 6A PMW solar charge controller. An image of this controller is shown in Figure 24, printed with permission from MorningStar. The controller costs $45.00 and has a maximum recommended panel power of 72W. Inputs include a 30V maximum panel voltage input, a 12V battery input, and a load input with a continuous rated load current of 6A. The controller consumes 8mA when operating which is many times more than the MMPT controller. The minimum battery voltage for normal operation is 1V and the load in-rush capability is 45A. A four-stage battery charging profile is used in order to increase the battery life and maximize capacity. This includes an absorption voltage of 14.1V, a load-disconnect voltage of 11.5V and a load-reconnect voltage of 12.6V. A battery temperature compensator is also included, which adjusts the voltage at a rate of -30mV/℃. A 5-year warranty is included with purchase. It weighs 8 ounces and measures 6”x2.2”x01.3”. A built-in electronic protection circuit protects the controller from overload, short-circuit, and high voltages from the solar and load inputs as well as high voltages from the battery.

Figure 24 - Moringstar SunSaver-6

Morningstar also makes the SS-10L-12V, which is a similar controller that is designed for larger panel powers. This controller costs $55.00 and has a maximum recommended panel power of 120W. Inputs include a 30V maximum panel voltage input, a 12V battery input, and a load input with a continuous rated load current of 10A. The controller consumes 8mA when operating, which is also many times more than the MMPT controller. The minimum battery voltage for normal operation is 1V and the load in-rush

capability is 65A. All of the other characteristics of these charge controllers are identical to those of the SS-6L-12V.

We decided to go with the MPPT because it consumes less power during operation. Although the PWM works better for small systems like ours, and has good efficiency in hot environments, the power consumption is far greater than the MPPT.

3.3.6 Power Generation

When we combine the monocrystalline panels and the thin film panels in parallel and feed it through the MPPT, we end up with significant power generation. The power generated by the solar panels obviously varies based on the weather, angle of exposure to the sun, and other environmental obstructions. Overall, the power generation from our combined solar panels should be very effective. See table 7 for our estimates on power generation with the panels operating at varying levels of efficiency.

Light Condition	Power Generated	Time to Fully Charge 60 W-h Battery
100% Ideal	36.96 W	1.62 hr
75% Ideal	27.72 W	2.16 hr
50% Ideal	18.48 W	3.25 hr
25% Ideal	9.24 W	6.49 hr

Table 7 – Combined Power Generation

3.3.7 Power Output

The load output of the charge controller is connected to the PCB which is designed to deliver a specified amount of power to each of the loads. The loads include the roller blind motor, microcontroller, USB charger, LCD display, and temperature sensor.

3.3.7.1 Roller Blind Motor

The roller blind motor operates at 12V and draws around 1.5A of current when operating. This is the largest current required by any of the components. However, the motor will only have very infrequent use. It will only take about 15 seconds of continuous operation to roll the canvas up or down. The amount of power the motor will take is determined by the amount of use by the user, but it will be far from constant.

3.3.7.2 Microcontroller

We seriously considered only two options for our MCU. The Atmel SAM3N00A is one of the low power options available by Atmel that was considered. Normal operation requires 1.62-3.6 Volts, with an active current consumption of 18mA and sleep mode current of 8.4mA. This is very much acceptable for the power requirements of our project.

The other option we considered for our MCU was the Texas Instruments CC2640. This chip takes 3.8V max voltage, with an active current of 2.938mA and sleep current of 0.550mA. The power consumption of the Atmel was good, but the CC2640 is over a dozen times more efficient. This was clearly the option we wanted to go as far as power consumption was concerned.

3.3.7.3 USB Charger

The biasing of the four USB pins is important when deciding on what kind of devices will be capable of charging. Pin 1 is defined as Vcc and is always biased at 5V. A 5V voltage regulator will be needed in order to reduce the 12V battery voltages to 5V. The TI L7805 voltage regulator will maintain a fixed output voltage of 5V with a maximum output current of 1.5A. Pin 4 is defined as ground and is always connected to ground. The charging port of a USB chargeable device contains hardware that detects when a charger is connected. Traditional USB ports have a maximum output of 0.5A; however, newer devices are capable of using higher currents, which results in faster charging times. Pins 2 and 3 are defined as the data pins. Pin 2 is the negative data pin and pin 3 is the positive data pin. If the data pins of a charger are left floating then the device will default to the traditional 0.5A rate. If the data pins are shorted then the device will be capable of drawing larger currents. This current will depend on the resistors used. We decided to leave the data pins floating to simplify this aspect of the project and also to simplify the current draw requirements, which makes it easier to determine average power consumption since it will always discharge the battery at 0.5A.

3.3.7.4 Display Screen

Table 8 shows the recommended operational conditions for the Pervasive Displays 2.7 inch e-paper display. This is higher than expected, but the statistics themselves are a bit misleading. The reason we liked the e-paper is that it only requires power to change the display. All figures displayed on the screen will persist even without any power. This is perfect for our project because the only time we need to update the screen is when there is a temperature change or a significant change in the battery charge.

Parameter	Symbol	Value			Unit
		Min	Typ	Max	
Digital Power	V_{DD}	2.3	3.0	3.6	V
Analog Power	V_{cc}	2.3	3.0	3.6	V
Input Voltage	V_{IH}	$0.8V_{DD}$	-	V_{DD}	V
	V_{IL}	V_{SS}	-	$0.2V_{DD}$	V
Output Voltage	V_{OH}	$0.8V_{DD}$	-	V_{DD}	V
	V_{OL}	V_{SS}	-	$0.2V_{DD}$	V
Input Current	$I_{DD} + I_{CC}$	-	5	10	mA

Table 8 - Operational Conditions for E-paper Display

We had to move away from using the e-paper display because there was a significant lack of documentation for developing with it. E-paper is still a relatively new technology, so this should have been expected, but the benefits of it caused us to spend a lot of time considering it as an option for our display.

After we had ruled out e-paper, we decided to look at alternative displays. The screen we chose to use, Newhaven Display International model NHD-C0216AZ-FN-GBW does not have any backlighting. Because of this, it operates at a lower power level than other LCD screens we have seen, which was the main reason for choosing this particular model. According to the data sheet, the LCD operates with an average voltage of 5.0V and a current of 1.0mA.

3.3.7.5 Temperature Sensor

We needed to add a temperature sensor to the project, since we had outlined it as one of our main requirement specifications. The probe does not take much power at all, and is actually the least power-consuming component of the project. It operates at 3V with a current of 1.5mA.

3.3.7.6 Cumulative Power Consumption

Table 9, below, shows the breakdown of the power consumption of each component. These values were taken straight from the components' corresponding data sheets. When we look at the combined power consumption of all the various components, it is apparent that the most power-hungry components are the motor and USB charger. Fortunately, the motor will only operate for about 15s at a time, and the USB charger will only draw current while a device is connected.

Component	Voltage (V)	Current (mA)	Power (mW)
Motor	12	1500	18000
USB	5	500	2500
MCU	3.8 max	2.94	11.17
Bluetooth RX	3.8 max	5.9	22.42
Bluetooth TX	3.8 max	9.1	34.58
Display	5	1	5
Temperature Sensor	3	1.5	4.5

Table 9 – Cumulative Power Consumption

As an additional analysis, we took a look at how long it would take for a phone to fully charge with the USB charger. Table 10, below, shows this analysis based on a phone with one of the largest batteries, the Samsung Galaxy S5.

USB Current Drain (Ah)	USB Power Drain (Wh)	Time to Fully Charged (h)
2.8	14	5.6

Table 10 – Samsung Galaxy S5 Power Consumption via USB

Since some of these components would not be running full time, we decided to normalize the component power drain by comparing them all in Watt-hours. The table to use for comparing can be seen in Table 11.

Component	Current Usage (A-h)	Power Usage (W-h)
Motor (operating for 15s)	0.00625	0.075
USB Charger	2.8	14
MCU	0.00294	0.011172
Bluetooth RX	0.0059	0.02242
Bluetooth Tx	0.0091	0.03458
Display	0.001	0.005
Temperature Sensor	0.0015	0.0045

Table 11 – Power Consumption in Watt-Hours

As we can see, the USB charger takes a significant amount of power to use and would not be able to fully charge a Samsung Galaxy S5 with the battery capacity we have. However, regular operations take very minimal power to operate. Considering that our battery has a capacity of 60Wh, if we don't use the motor or USB charger, the product will continue to operate normally for about a month without recharging.

3.4 Possible Software Development Environments

3.4.1 Embedded Microchip Environments

Since we decided to use an ARM microcontroller unit as part of our requirements, we needed development software that was compatible. First, we needed to find out what language would be best to use. There are several languages we could use such as assembly, C, C++, Pascal, Basic, Java, and some others. However, when developing for an MCU, we will want to focus on assembly or C for portability and efficiency reasons. With C being a more intuitive language than assembly, we will go with that language instead. Most modern compilers can translate C language to the MCU almost as efficiently as writing assembly code directly. The only reason to ever program an MCU with assembly anymore is if the response time of the application needs to be as real-time as possible, such as with some devices used in the medical field or the military, but this does not apply for our project so C would work fine.

After we decided to use the C software language, we needed an IDE that supports both C and ARM. There were many choices, and in retrospect, IAR probably would've been the best program to use since we went with a Texas Instruments MCU and IAR is

specifically made for programming their chips. Regardless, we considered the following IDEs for our embedded development:

- Code Composer Studio
- Crossworks
- DS-5 (Eclipse plugin)
- Keil MDK

Code Composer Studio has limitations on its free software, and still requires an annoying download and setup process. However, it is easy to use after you go through the download process and our computer engineers have prior experience with it. Crossworks does not appear to be very user-friendly. Keil MDK appears to function well and without much of a learning curve. DS-5 has a free community version, which is a plugin for the Eclipse IDE. This is an alluring option because we have used the Eclipse IDE for Java development during previous courses using Java.

We decided to go with the Code Composer Studio for our embedded software development. At first, the DS-5 plugin seemed like a good option, but upon reflection it seemed that it might cause more complications due to issues faced with other Eclipse plugins. The risk of delaying our project development is not worth the convenience and familiarity it may give. Since Code Composer Studio is specifically geared toward MCU programming, and the fact that we are familiar with it, it did not pose many unforeseen complications.

3.4.2 Mobile Application Environments

The mobile application for this project was developed for Android devices due to budget constraints and our prior experience with Android programming. The core programming language we decided to use was either going to be Java or C#. With this in mind a few software development platforms can be considered.

First on the list is Google's recently revamped Android Studio. This would serve as the best ideal development program if we would chose Java as the primary application development. The reason is that it supports the development of Android apps, has a vast amount of resources and guides to help prevent problems, and is directly related with Android's creators. With Android Studio's built in features like intelligent code editor, GitHub integration, multi-screen app development, virtual devices, and Android builds. Having all of these features in one platform can save the developers the trouble of exporting code and uploading the different build versions [36].

Another considerable software development platform for the mobile application would be Xamarin. This platform utilizes C# as the core programming language instead of common languages like Java, Objective C, and Swift used in most other mobile application development platforms. Other features in Xamarin are built in user

interfaces, API Access, integrated GitHub version control, and the Xamarin Test Cloud [35]. If we were to choose Xamarin, the developers would be required to learn and use C#. This was not a necessarily bad thing, but could pose potential setbacks as they would need to have experience and knowledge. Not having this experience can create a possibly large learning curve, but the developers would have an opportunity to both learn and use a new programming language and platform.

Another benefit of Xamarin would be the increase in accessibility of the mobile application. This is because Xamarin allows iOS development readily unlike Android Studio. The disadvantage is that it would cost money to actually place the application on Apple's Appstore. This is just a consideration though since we actually had no plans to develop for iPhone or even place it on the Appstore, so this is more of a feature that is only nice to boast.

The last option for a mobile development platform is Eclipse IDE. This one would require the most behind the scenes setup to get started but allows many of the same features as the other two platforms. Like Android Studio, it is designed for Java programming. The main reason we decided not to go with the Eclipse plugin is that our developers had previous experience with it and had a lot of drawbacks. For one thing, GitHub integration is not included like it is in the other IDEs. The built-in Android emulator is also very unresponsive, which forces use of a 3rd party emulator for testing.

Based on the computer engineers' preference, we decided to develop the mobile app in C# for the experience with the new language and IDE. The main drawback to this choice was the lack of development resources for Xamarin. Since it appears to be the only IDE that allows Android programming in C#, the only help we could find was on the Xamarin forums, which were not always complete.

4.0 Related Standards

Standards greatly affect how new technologies are spread in the United States. The purpose of forming standards is to simplify product development and to ensure safety in the development and use of new products. Standards make products more interoperable, which can lead to faster development and cost reduction. Many standards were utilized in our project including safety standards, reliability standards, communication standards, programming language standards, connector standards, and battery standards. Some of these standards will be pre-established by organizations such as IEEE, ANSI and UL and others will be project specific, defined by our team members. Below are a few of the standards that will be used in our project.

4.1 Safety Standards

Table 12 lists the safety standards relevant to our project. As always, safety is a primary concern in any project. Our project does not involve any hazardous materials or sharp objects, but safety should always be considered when working with electrical parts. For example, the housing in which the electronics and the rolled up canvas will be housed will possibly reach high temperatures since it will be in direct contact with the window, at least on one side. We have to consider the fact that this may pose a fire hazard.

Standard	Description
IEC 61508	Functional safety of electronic and programmable safety-related systems.
NFPA 70	National Electrical Code that specifies safe electrical design, installation, and inspection to protect people and property from electrical hazards.
UL 50	Performance requirements for enclosures to provide a degree of protection to personnel against incidental contact with enclosed equipment
UL 62109-1	Describes standard for safety of power converters used in photovoltaic power systems.

Table 12 - Safety Standards

4.2 Reliability Standards

Table 13 lists the reliability standards relevant to our project. The purpose of a project would be meaningless if the product is not designed to be reliable. In this case, our largest reliability risk will be the solar cells we use to absorb energy from the sun. We

will need to know the exact characteristics of our cells in order to properly utilize them and ensure maximum performance and, therefore, reliability.

Standard	Description
UL 61215	States requirements for testing crystalline silicon photovoltaic modules to determine the electrical and thermal characteristics, which are used to determine if the module is capable of withstanding prolonged exposure to a specified climate.
UL 61646	States requirements for testing thin-film photovoltaic modules to determine the electrical and thermal characteristics, which are used to determine if the module is capable of withstanding prolonged exposure to a specified climate.

Table 13 - Reliability Standards

4.3 Communications Standards

Table 14 lists the standards for communications. The three possible protocols we used were Bluetooth and RFID. We included Wi-Fi here since we originally explored it as a possibility for our remote connectivity protocol.

Standard	Description
IEEE 802.11b	Defines wireless-networking specifications up to 11 Mbits/s using a 2.4GHz band (Wi-Fi).
IEEE 802.15.1	Defines wireless medium access control and physical layer specifications for wireless personal area networks (Bluetooth).
IEEE 802.15.4f	Defines protocol for active RFID and sensor applications

Table 14 - Communications Standards

4.4 Programming Language Standards

In reference to the standards of the programming languages we are following many of the standardizations of ISO/IEC 9899:2011 also known C11. However, the personal preferences outlined below are some case specific guidelines the developers were responsible for following in order to help keep the code legible for debugging purposes.

4.4.1 Version Control

In order to handle the version control of the mobile application development the developers integrated a code collaboration tool into the development cycle. There are

quite a few collaboration tools to choose from but we limited the choices to the following four tools: Beanstalk App, Cloud9, GitHub, and Jira [37]. This following discussion will analyze the advantages, disadvantages, and the overall cost association with using these tools.

A distributed software company called Wildbit makes the Beanstalk App. This company's focus is to improve the way web apps are built, ran, and deployed. The overall idea behind the Beanstalk App is to create a hosting service that can offer GitHub and SVN version control, developer collaboration tools, and deployment tools. Beanstalk is also packed full with multiple useful features designed to help developers track code changes, error tracking, and deploy working code [38].

Beanstalk's pricing plans varies depending on the needs of the projects and developers. However they do offer a free plan that is available for the first two weeks of a subscription. The prices for continuing use after the end of the two weeks go for $15 per month up to 200 per month. There are also two different levels of pricing plans. One being made for freelancers and startups that as the prices increase the only changes in the plans are to the total storage space, repositories, users, and servers. The other level is made for businesses and enterprises that add more features altogether to the plans. These features are bonuses like added security tools, priority support, and custom backups. For our projects development these added features were a little bit extreme in terms of what we actually planned to build for the mobile application [39].

Overall Beanstalk is a great tool the developers could use to code and work together with features like version control, code editing using their built-in editors, and many more. The only downside is that the tool costs money monthly to use during the development phase. This is an extra expense that we wanted to try to minimize.

The next tool for discussion is called Cloud9. This code collaboration tool differs from the other tools by the fact that it is a cloud-based IDE. This feature also gives Cloud9 the ability to have real time coding and chatting unlike most of the other tools [43]. Cloud9 is able to do this by actually creating developer environments that can support a variety of languages. Cloud9 offers a free account with limited features and a maximum of one premium workspace. The developers would be able to function completely with just a free account alone. However, if more functionality is desired, the pricing for Cloud9 base plans start at $9 per month going up to $79 per month for large plans [44]. The base plan would certainly offer enough resources to operate with maximum efficiency for our project.

One of the most popular, if not the most popular, collaboration software tools available is GitHub. With the ability to work with the tool almost anywhere, GitHub possesses a huge amount of advantages over its competition. Built as an open source platform, many companies have also created add-ons or even integrated GitHub's services into their own products. With powerful collaboration tools, the ability to perform code

reviews along with code management, issue tracking, and completely free access to the entire feature without a paid plan GitHub may be the best pick as far as an all-in-one collaboration tool. The only disadvantage to using GitHub is the free plan limits the developers to a public repository. This means anyone can access the project to view our files or even use our code and work. The only way to gain access to a private repository is if we choose an upgraded plan starting at $7 per month. All of their plans each offer unlimited collaborators and public repositories. The only things that change as the prices increase is the total amount of private repositories available to the collaborators. The plan we would purchase would be the micro level plan that offers 5 private repositories. Also GitHub offers education discounts to even reduce the monthly costs even more [42].

Last but not least is Jira. This is a project management software product for collaborative development. It is made by a company called Atlassian, which specializes in making an assortment of products for developers trying to work together. The purpose of using Jira, in contrast with some of the other collaboration tools, is that it's great for planning, tracking, and reporting progress of the developers [40]. This is a great product for the developers who plan to use agile methodologies. The features built into Jira are knowledge management, development workflow, continuous integration, and real-time collaboration. Jira offers a free 7 days then prices vary depending on if you choose to be a cloud based or server based system. Cloud based systems range from $10 to $1,000 per month while server based systems range from $10 to $24,000 per month. The price increases as more users join your team, but all of the other features do not change. For the amount of developers for our project we would only need a plan costing $10 per month if we were to proceed and use Jira as a program management tool [41].

After considering all of the tools we decided that we would be using GitHub to handle our version control needs. We used the micro plan level, which would normally cost $7 per month. One of the developers in our team already had this plan prior to the start of the course and had an extra private repository that we could use for this project. In doing so, we were able to save us the cost of paying $7 per month in our project budget. Additionally, since we were able to use a private repository, we could ensure that our intellectual property will not be vulnerable to others. It is important to protect the privacy of our work to avoid potential plagiarism claims if another group of people decided to steal our research and claim us as the violators.

4.4.2 Version Control Standards

The following list contains all of the version control coding preferences the developers must adhere too.

- The naming convention to keep track of forks in the repositories is as follows:
    - Ex: [1.00]ClassName.filetype

- A scenario of every time we upload a file with new changes, we will increment the version count of the name of the file. If the original file name is [1.01]ClassName.filetype. After commiting code changes we would then upgrade the version count of the file to [1.02]ClassName.filetype.
    - Ex: Original = [1.01]ClassName.filetype
      New = [1.02]ClassName.filetype

- In order to avoid working on the same files at the same time we will follow this rule in certain scenarios. If we ever have to switch over the file we're working on, we'll formalize everything to the next version. So instead of [1.XX], we'll push it to [2.XX] so we don't have conflicting changes in the same file.

4.4.3 Embedded Language Standards

The developers are to abide by these standards set below at all times for the purpose of standardizing all code in the C language for embedded programming of the projects hardware applications.

4.4.3.1 Naming Conventions Standards (Embedded Programming)

Follow these naming conventions to ensure the developers know what the variable scope is along whether it's a function of file being used.

- Constants will use all caps.
    - Ex: int CAPS = 1000;
- Constants will use underscores to replace the spaces in the name.
    - Ex: int ALL_CAPS = 1000;
- Functions will use camel case with the first word always lowercase.
    - Ex: functionNamesLikeThis();
- File names will be camel case with the first word always uppercase.
    - Ex: FileNameLikeThis();
- Variable names will be all lowercase.
    - Ex: int lemons = 0;
- Variable names will use underscores to replace the spaces in the name.
    - Ex: int num_of_lemons = 0;
- All names should be descriptive enough to leave no ambiguousness to the other developers when reviewing code during a code review.

4.4.3.2 Blank lines Standards (Embedded Programming)

The use of blanks line should be standardize in a way the developers can easily scan the code to search and find what they are looking for. These rules should always be used in the following cases to ensure that the minimum extra white space will be in the code.

- Three lines inclusive should be in between sections in order to help separate the two areas of code.
    - Ex: After a method and before a new method there should be blank lines.
- One line break may be used to separate code inside of a function or method whenever it would improve readability.
    - Ex: Immediately after variable declarations and conditional statements
- All return statements will be separated from the code by a line break.
    - Ex: A code sample is shown below

```
int main(){
        printf("Hi");
        return 0;
}
```

- In all other cases there should be no line breaks unless it falls into one of the categories listed above.

4.4.3.3 Function Formatting Standards (Embedded Programming)

All functions in the code are to be standardized by the following rules.

- Immediately after naming or declaring a function an open bracket is to follow. Then an immediate line break wit the content of the function starting one indent inside of the previous indent mark. The closing bracket will be placed on a new line with the same indentation level as the original indentation at the beginning of the function.
    - Ex: A code sample is shown below

```
private double functionNameHere(int some, int variable){
        printf("Group: ");   // Some example code
        printf("11");   // Indented like this
        return 0;
}  // The closing brackets will always be on a newline.
```

4.4.3.4 Comments Standards (Embedded Programming)

The developers should avoid using multi line comments and follow these standards in regards to comments

- Each file will have multiple section headers (done with the built-in commenting syntax) in the following format:
    - Ex. A code sample is shown below

```
//*********************************************************
//*    Title of section, e.g. Header, Constants, Main Loop, or Functions      *
//*              Up to two lines of text inside - Centered                    *
//*********************************************************
```

- Every function will have a description of what it does, as well as a short description of each input and output (avoid using global variables besides constants), pass all inputs and outputs to/from the main loop with function calls and returns:
    - Ex: A code sample is shown below

```
// Prints text to the screen
// x: The text that is to be printed
void printToScreen(String x) {
        printf("%s\n", x);
}
```

- Besides the section headers and the function headers, all other developer comments will be end-of-line comments. First apply three spaces after the code. Then "//" followed by a space. Then immediately after the space you can type the details you wanted to address in the comment.
    - Ex: A code sample is shown below

```
printf("Hello World");   // Print statement for Hello world
```

4.4.3.5 Conditional Statements Standards (Embedded Programming)

All types of statements and loops should be written as follows.

- First the type of statement or loop should be written. Then a space will follow separating the type from the opening parentheses. After the comparison statements are made a closing parentheses should be placed. Then a space will be in between the closing parentheses and the opening bracket.
    - Ex: A code sample is shown below

```
for (i = 1; i < 10; i++) {
        printf("i = %d",i);
}
```

- If statements, for loops, and while loops along with any other types shall be written with a space in between each of the elements and the operator inside of the parentheses.
    - Ex: A code sample is shown below

```
if (a = b) {
        a++;
}
```

- If they use brackets the open bracket should follow the closing parentheses with a space in between them. They closing bracket should be on a newline lined up with the beginning conditional statement indentation.

4.4.3.6 Miscellaneous Standards (Embedded Programming)

These are miscellaneous standards for embedded programming that did not fit into any other specific areas.

- Try to avoid lines longer than 100 characters. This standard depends on the IDE we choose. Also this character count does not include comments.
- General progression: header > libraries > global variable declaration > main loop > function calls

4.4.4 Object-Oriented Language Standards

The developers are to abide by these standards set below at all times for the purpose of standardizing all code in the object-oriented language for mobile programming of the projects software applications.

4.4.4.1 Naming Conventions Standards (in object-oriented language)

Follow these naming conventions to ensure the developers know what the variable scope is along with whether or not it is a function or file being used.

- Constants will use all caps.
    - Ex: int CAPS = 1000;
- Constants will use underscores to replace the spaces in the name.
    - Ex: int ALL_CAPS = 1000;
- Methods will use camel case with the first word always lowercase.
    - Ex: methodNamesLikeThis();
- Class names will be camel case with the first word always uppercase.
    - Ex: FileNameLikeThis();
- Variable names will be all lowercase.
    - Ex: int lemons = 0;
- Variable names will use underscores to replace the spaces in the name.
    - Ex: int num_of_lemons = 0;
- All names should be descriptive enough to leave no ambiguousness to the other developers when reviewing code during a code review.

4.4.4.2 Blank lines Standards (in object-oriented language)

The use of blanks line should be standardized in a way the developers can easily scan the code to search and find what they are looking for. These rules should always be used in the following cases to ensure that the minimum extra white space will be in the code.

- Three lines inclusive should be in between sections in order to help separate the two areas of code.
    - Ex: After a method and before a new method there should be blank lines. One line break may be used to separate code inside of a function or method whenever it would improve readability.
    - Ex: Immediately after variable declarations and conditional statements
- All return statements will be separated from the code by a line break.
    - Ex: A code sample is shown below

```
int main(){
        printf("Hi");
        return 0;
}
```

- In all other cases there should be no line breaks unless it falls into one of the categories listed above.

4.4.4.3 Method Formatting Standards (in object-oriented language)

All functions in the code are to be standardized by the following rules.

- Immediately after naming or declaring a method an open bracket is to follow. Then an immediate line break with the content of the method starting one indent inside of the previous indent mark. The closing bracket will be placed on a new line with the same indentation level as the original indentation at the beginning of the method.
    - Ex: A code sample is shown below

```
private double methodNameHere(int some, int variable){
        printf("Group: ");   // Some example code
        printf("11");   // Indented like this
        return 0;
}  // The closing brackets will always be on a newline.
```

4.4.4.4 Comments Standards (in object-oriented language)

All types of statements and loops should be written as follows.

- Each file will have multiple section headers (done with the built-in commenting syntax) in the following format:
    - Ex. A code sample is shown below

```
//*************************************************************
//*    Title of section, e.g. Header, Constants, Main Loop, or Functions      *
//*              Up to two lines of text inside - Centered                    *
//*                                                                           *
//*************************************************************
```

- Every function will have a description of what it does, as well as a short description of each input and output (avoid using global variables besides constants), pass all inputs and outputs to/from the main loop with function calls and returns:
    - Ex: A code sample is shown below

```
// Prints text to the screen
// x: The text that is to be printed
void printToScreen(String x) {
        printf("%s\n", x);
}
```

- Besides the section headers and the function headers, all other developer comments will be end-of-line comments. First apply three spaces after the code. Then "//" followed by a space. Then immediately after the space you can type the details you wanted to address in the comment.
    - Ex: A code sample is shown below

```
printf("Hello World");   // Print statement for Hello world
```

4.4.4.5 Conditional Statements Standards (in object-oriented language)

All types of statements and loops should be written as follows.

- First the type of statement or loop should be written. Then a space will follow separating the type from the opening parentheses. After the comparison statements are made a closing parentheses should be placed. Then a space will be in between the closing parentheses and the opening bracket.
    - Ex: A code sample is shown below

```
for (i = 1; i < 10; i++) {
        printf("i = %d",i);
}
```

- If statements, for loops, and while loops along with any other types shall be written with a space in between each of the elements and the operator inside of the parentheses.
    - Ex: A code sample is shown below

```
if (a = b) {
        a++;
}
```

- If they use brackets the open bracket should follow the closing parentheses with a space in between them. They closing bracket should be on a newline lined up with the beginning conditional statement indentation.

4.4.4.6 Miscellaneous Standards (in object-oriented language)

These are miscellaneous standards for embedded programming that did not fit into any other specific areas.

- Try to avoid lines longer than 100 characters. This standard depends on the IDE we choose. Also this character count does not include comments.
- If you have to break up a method call to multiple lines, indent up to the start of the function call + 8 spaces (not an indent). Also this character count does not include comments.
- General progression: header > libraries > global variable declaration > main loop > function calls
- Avoid using multi-line comment notations
    - Ex: A code sample is shown below

```
/*
* Avoid using this format if possible. This commenting notation is not
* standardized between all languages and IDEs
*/
```

- Always use end of the line comments or refer back to the top of the list.

4.5 Connector Standards

4.5.1 USB Standards

Table 15, below, shows a list of relevant standards for the Universal Serial Bus.

Standard	Description
UL 6703	Outline of Investigation for Connectors for Use in Photovoltaic Systems
ANSI C119.6-2011	Standard for electrical connectors used in non-sealed, multiport connector systems rated 600 V or less for aluminum and copper conductors.
USB Battery Charging 1.2 Compliance Plan	Defines standards used for USB chargers.

Table 15 - USB Standards

4.5.2 Battery Connector Standards

Table 16, below, shows a list of relevant standards for the battery connector.

Standard	Description
IEEE 937-2007	Recommended Practice for Installation and Maintenance of Lead-Acid Batteries for Photovoltaic (PV) Systems
IEEE 1013-2007	Recommended Practice for Sizing Lead-Acid Batteries for Stand-Alone Photovoltaic (PV) Systems
IEEE 1361-2014	Guide for Selecting, Charging, Testing, and Evaluating Lead-Acid Batteries Used in Stand-Alone Photovoltaic (PV) Systems
IEEE 1526-2003	Recommended Practice for Testing the Performance of Stand-Alone Photovoltaic Systems

Table 16 - Battery Connector Standards

4.6 Battery Standards

Table 17, below, contains IEEE standards are relevant to our project regarding batteries.

Standard	Description
IEEE 937-2007	Recommended Practice for Installation and Maintenance of Lead-Acid Batteries for Photovoltaic (PV) Systems
IEEE 937-2007	Recommended Practice for Installation and Maintenance of Lead-Acid Batteries for Photovoltaic (PV) Systems
IEEE 1013-2007	Recommended Practice for Sizing Lead-Acid Batteries for Stand-Alone Photovoltaic (PV) Systems
IEEE 1361-2014	Guide for Selecting, Charging, Testing, and Evaluating Lead-Acid Batteries Used in Stand-Alone Photovoltaic (PV) Systems
IEEE 1526-2003	Recommended Practice for Testing the Performance of Stand-Alone Photovoltaic Systems

Table 17 - Battery Standards

4.7 Design Impact of Relevant Standards

Implementing the standards listed above made the project safer to the end user.

Using products that are compliant with the NFPA 70 standard ensures that the electronics will not introduce any hazardous conditions that could result in fire.

The UL 50 was considered when selecting the enclosure that the electronics will be stored in. This is to ensure that users are protected when incidentally coming into contact with the enclosure.

The UL 62109-1 standard was considered when selecting our boost converter, charge controller, and other power converters used in the solar module sub-system.

UL 61215 and UL 61646 were used when selecting the photovoltaics for the project. This ensures that any crystalline silicon or thin-film photovoltaics will be able to provide reliable power in the specified installed environment.

The Wi-Fi and Bluetooth standards were used in order to determine which remote connectivity method was most suitable for the project. Additionally, the RFID standard was used when designing the security sub-system for the project.

UL 6703 was used in order to choose compliant connectors used in the solar module sub-system. This includes the connections from the solar module to the charge controller, the charge controller to the battery, and from the charge controller to the PCB.

The USB Battery Charging 1.2 Compliance Plan was used in order to choose the pin assignment for the USB charger so that it operates in accordance with the standard. The battery standards were used as a guide for selecting the appropriate battery size for the project, charging the battery, and testing the battery to ensure it is performing well.

5.0 Realistic Design Constraints

5.1 Economic Constraints

There were some economic issues to take into account when designing the product for this project. We did not have any sponsorship for this project, so all expenses were paid by members of the team. The electronic pieces were not much of an issue, but the solar cells were the largest expense that we had to take into consideration.

One way we have been able to lower the cost of the solar cells is to use less of the flexible panels and more of the solid crystalline ones. The crystalline cells are much cheaper, but the only place we could fit them into our design was on the housing, which had limited exposure to the window. If we used only the crystalline cells, we estimated that it would not be able to gather enough energy on its own to meet our requirements. As a result, we decided to use a combination of the two architectures to maximize the energy gathering while trying to minimize cost.

Another effect of economic constraints can be seen in our choice of blinds. We had originally decided to use a vertical blind design that would operate with two motors, one to rotate the blinds open and another to pull them to the side. We discovered that it would be cheaper, not to mention easier to implement, to use a rolling canvas instead. The rolling canvas operates only on one motor and has a more simple design, which reduced the overall cost of our project.

Financial constraints have also affected our software design decisions as well. We had originally conceived of this product to be a complement to the Apple HomeKit [45]. However, to develop mobile applications for Apple, you have to get a license which costs $99 per year [46]. This is a decent price if you are a very active developer and will be producing multiple applications annually. However, we only had plans to use the license for this project alone and this cost was a big detriment to developing the application on Apple's platform. Android, on the other hand, was free to develop as long as it is not distributed through the Google Play store. If we wished to put the application on Google Play, it is only a $25 one-time registration fee [47]. Because of this, among other reasons, we decided to develop the mobile application for Android.

As important as economic constraints were to us in the beginning of our development process for this project, our priorities eventually shifted. Our original focus was on lessening the economic impact of our project, but as we grew closer to the end of the time frame given to use for this course, we soon shifted our focus from financial issues to temporal ones. For example, we ended up paying a few hundred dollars extra for overnight shipping of new parts in order to try to complete our project by the given deadline.

5.2 Time Constraints

The time constraint is also a large factor in our design choices. We only had one semester to build this product. This meant we had to build/buy the canvas, housing, and motor. We also had to acquire all the parts for our PCB and have it all integrated into the board. We also had to attach and wire all the solar cells to the blinds and get the charge controller working properly. We also expected some possible complications with the USB charger being able to charge at appropriate speeds across different phone models.

All of those physical requirements had to be met within our completion milestone of November 30th. In addition, we had to have software development going on concurrently. We needed to program the microcontroller to handle all the mechanical operations, and also to process commands from the mobile application. We also had to develop the mobile application itself, meaning its user interface and database system for device registration. Development of one part of the project sometimes depended on the prior completion of another part, further delaying overall completion.

These requirements and our time constraints compounded into affecting our design decisions in order to meet our deadline with a finished product. Instead of building everything from scratch, we had to buy a MPPT. Instead of soldering the parts to the PCB ourselves, we sent out the parts to a third party to integrate them all for us (at additional cost). We also considered the fact that a data link was required for the USB charger to determine the optimal current to maximize charge time for a cell phone, but due to time constraints we simply shorted the data pins in order to charge all devices at a constant 500 mA (or less, if the device had a maximum input current lower than that).

As far as the software development goes, it was affected by the time constraints as well. We were hoping to be able to include additional features in our program such as expanding the weather API to include an alert system to warn the user of approaching high sun-exposure times and reminding them to close the blinds to maximize energy generation. We also wanted to be able to display the rate at which the battery is charging at any point in time, and then to be able to display a graph for prior charging history. However, these are superfluous features, and we had to get basic operations working first. Given enough time and resources, this product could turn out to be very impressive.

The limited time given to complete this project has also affected our chances to learn more about new technology. We are fortunate to be able to learn a bit about e-paper, Bluetooth, and get more hands-on experience with RFID, and a new object-oriented language (C#), but we had hoped to be able to expand our skill set more than that with this project. Ideally, we would have also liked to have learned more about e-paper, Wi-Fi, and metalworking to really make the product shine, but we had to cut out some features to meet the time requirement.

5.3 Ethical, Health, and Safety Constraints

This project contains risks that had to be handled and taken into major consideration. This portion of the document will analyze the ethical, health and safety risks associated with the operation of Solar Blinds.

To answer the question of why we decided to pursue such a device as Solar Blinds is because we see that the demand for a new fuel source is at an all-time high. Seeing that the issues on air pollution and the ozone layer disappearing are always on the news. It's quite alarming that nothing can really be down about it until a new, easily accessible resource is found. Therefore the overall goal of this project is to present a device that can be used to reclaim one of the world's natural resources and turn it into an energy source we can use at our own discretion.

This product was designed to operate on a small scale for a normal everyday house. It can easily be converted into a product that will work with large business skyscrapers that face the sun most of the day, but since very few of them exist where we are located we will limit the project to everyday households.

The only health risks associated with the development of this device are the use of batteries and the heat generated by the product. In the section describing the use of batteries will elaborate more on how we decided to counteract this design constraint. The heat consideration is discussed in the Housing section.

Safety to the user is something that must not be taken lightly. To handle this design constraint we adhered to the standards determined by engineering professionals when we began building and coding. We also ensured that the parts that we had out sourced to be developed also adhered to the same engineering standards to ensure the proper requirements are met.

6.0 Project Hardware and Software Design Details

6.1 Electrical Hardware

Figure 25 is our block diagram displaying how all the different components of the project are connected. As we can see, everything goes directly through the PCB. The PCB is the heart of our project, so it is important that we design it properly and that it is completed as soon as possible. This was a potential bottleneck in our development, so it was something we had to give special attention as we went into the semester working on this project.

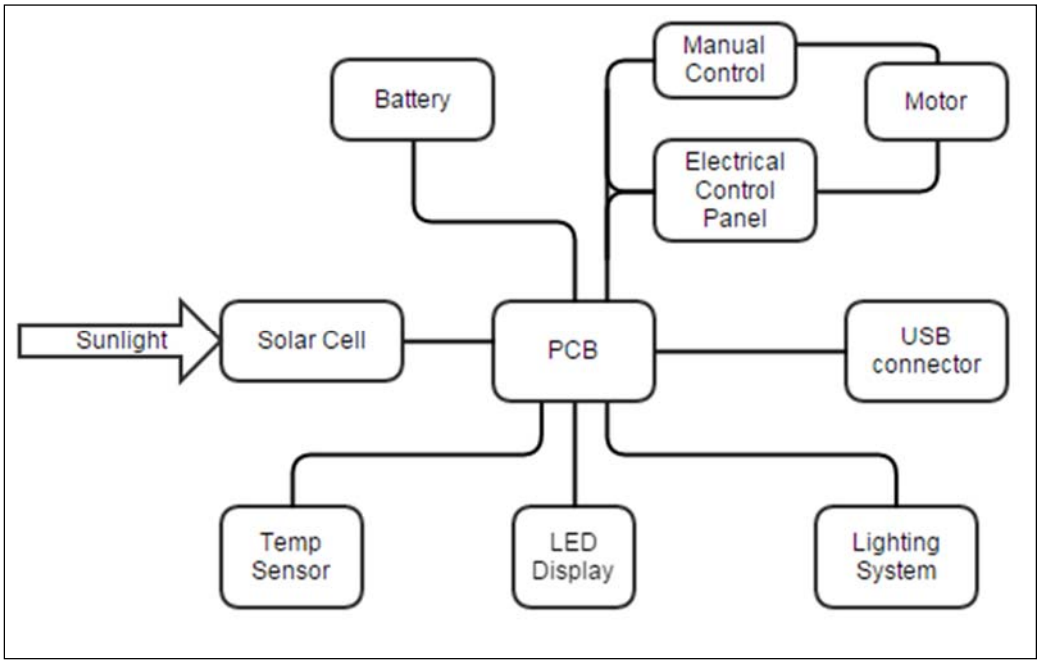

Figure 25 - Hardware Block Diagram

6.1.1 Charging Circuit

When the sunlight hits our solar panel and begins to send a voltage across these terminals, we needed to regulate the current and voltage that runs through the battery at any given time. This voltage would be fluctuating throughout the day and could reach voltages much higher than the battery and can harm the battery if it is not regulated correctly. We considered many different options to charge our battery correctly and safely.

One of the ways we considered was buying an aftermarket, already built battery charger from solar cells to a lithium battery. This would eliminate any errors that we could make by designing a circuit ourselves that would be quite complicated to ensure maximum power efficiency. Also, given the short time that we had to design, build and test this project, it would be much easier going with a premade battery controller that

eliminates a lot of time to build and test. We decided to go with the Gena Sun Gv-5 65W 5A Solar Charge Controller with MPPT. This controller monitors the voltage fifteen times per second and continues to keep a steady output voltage and current [48].

6.1.2 PCB

Eagle is the program we used to create our PCB layout and create all of our gerbers to send out to the manufacturers for our PCB. We designed all of the individual components into separate libraries with their own packages, symbols and device names. We then took these parts and dropped them into a schematic that looked like our previously layout schematic in LtSpice. This drops all of our components on a PCB and then we had to place them and route each part. The PCB was made to be as small as possible to reduce cost and be able to fit in the housing we needed it to fit into. We also deleted out many of the layers from Eagle as we did not need all of these layers that they gave us. Also, we did not need to do a panel drawing for our boards because we would not be ordering our boards by panel, but if this ever went into mass production, there would have to be a panel layout made as well as just the board layout.

The program we used to prototype our board was LtSpice. This program has you design in parts into different files and then allows you to bring them together and build schematics. This allowed us to build an accurate model of our schematic and assign the necessary pins and voltages to all of the different components that we needed in our PCB. You were also able to run mock circuits and simulate how the circuit will work when completed and put together.

6.1.3 Microprocessor

The first part that was designed in is the CC2640. This is a microprocessor that supports Bluetooth and has enough I/O pins to complete the job that we need it to complete. This MCU has a crystal hooked up to it, as well as a voltage of 3V to power it. This voltage was regulated down from 12V to 3V through a linear voltage regulator. There is also a motor hooked up to it that will spin in two different directions depending on which pin of the motor is pulled high or low. Also attached to this microcontroller is an LCD. This is the display that will be showing the temperature readings from the temperature sensor. The temperature pin will also be hooked up to the MCU and there will be a regulated 5 volts on the LCD as well. The pinout for the CC2640 is shown in Figure 26 [49].

Table 4-2. Signal Descriptions – RHB Package

Pin Name	Pin	Pin Type	Description
RF_P	1	RF I/O	Positive RF input signal to LNA during RX Positive RF output signal to PA during TX
RF_N	2	RF I/O	Negative RF input signal to LNA during RX Negative RF output signal to PA during TX
RX_TX	3	RF I/O	Optional bias pin for the RF LNA
VDDS	28	Power	1.8 V to 3.8 V main chip supply(1)
VDDS2	11	Power	1.8 V to 3.8 V GPIO supply(1)
VDDS_DCDC	18	Power	1.8 V to 3.8 V DC/DC supply
VDDR	29	Power	1.7 V to 1.95 V supply, typically connect to output of internal DC/DC(2)(3)
VDDR_RF	32	Power	1.7 V to 1.95 V supply, typically connect to output of internal DC/DC(4)(3)
DCOUPL	12	Power	1.27 V regulated digital-supply decoupling(3)
DCDC_SW	17	Power	Output from internal DC/DC(1)
EGP		Power	Ground – Exposed Ground Pad
RESET_N	19	Digital input	Reset, active-low. No internal pullup
DIO_0	6	Digital I/O	GPIO, Sensor Controller
DIO_1	7	Digital I/O	GPIO, Sensor Controller
DIO_2	8	Digital I/O	GPIO, Sensor Controller, High drive capability
DIO_3	9	Digital I/O	GPIO, Sensor Controller, High drive capability
DIO_4	10	Digital I/O	GPIO, Sensor Controller, High drive capability
DIO_5	15	Digital I/O	GPIO, High drive capability, JTAG_TDO
DIO_6	16	Digital I/O	GPIO, High drive capability, JTAG_TDI
DIO_7	20	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_8	21	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_9	22	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_10	23	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_11	24	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_12	25	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_13	26	Digital/Analog I/O	GPIO, Sensor Controller, Analog
DIO_14	27	Digital/Analog I/O	GPIO, Sensor Controller, Analog
JTAG_TMSC	13	Digital I/O	JTAG TMSC, High drive capability
JTAG_TCKC	14	Digital I/O	JTAG TCKC
X32K_Q1	4	Analog I/O	32 kHz crystal oscillator pin 1
X32K_Q2	5	Analog I/O	32 kHz crystal oscillator pin 2
X24M_N	30	Analog I/O	24 MHz crystal oscillator pin 1
X24M_P	31	Analog I/O	24 MHz crystal oscillator pin 2

(1) See Section 9.2, technical reference manual for more details

Figure 26 - CC2640 Pinouts

We originally assumed that this MCU had more than enough digital and analog I/O pins to get the job done. Later on, we realized that we were running out of pins and had to get a little creative in order to implement all of our components, such as putting both motor control buttons on one pin using a voltage divider. The MCU also has an ARM Cortex-M3 processor in it which seemed a bit overkill, but it was good experience to learn how to code it. This MCU also needs a 24MHz crystal to operate. Usually microcontrollers need some external capacitors but this specific micro has internal capacitors to tune the crystal. In the LtSpice design for this part, there is the outline of the part which is a square and there are the 32 pins that are attached to them. The pins are all named the same as the pinout above. The spice model is depicted in Figure 27.

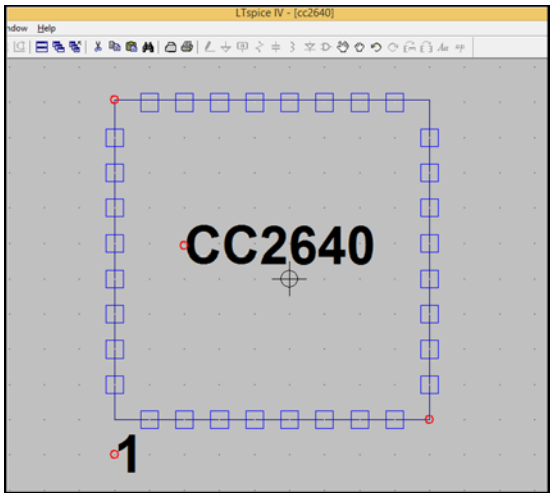

Figure 27 - LTSpice Pinout of CC2640

Designing this part into Eagle was not as simple as most parts. This part, since it is a microprocessor, has a much smaller pitch than most parts and the pads were shaped oddly. They were not your standard looking rectangular pads. The pads looked like a rectangle with a rounded top as shown in Figure 28. Also, another specification that made this part a bit more involved for creating in Eagle was that the package had a ground pad directly under it and had a pattern of vias that needed to be followed. This pattern is crucial for the correct heat sinking of the part.

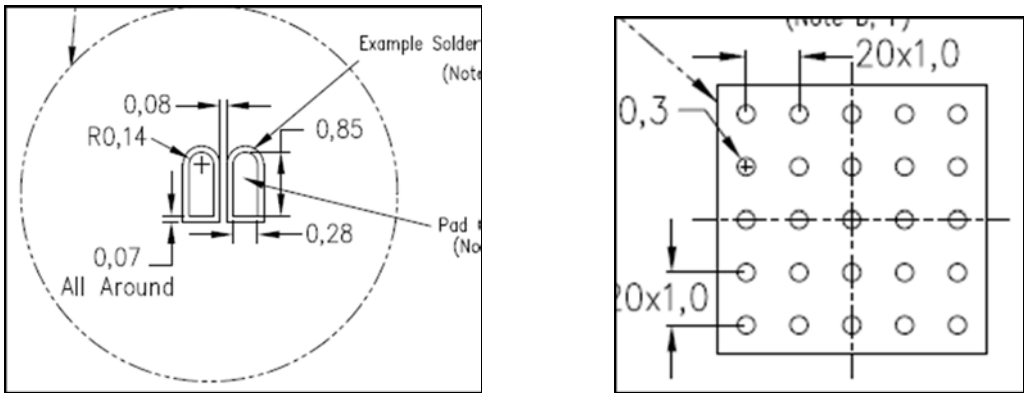

Figure 28 - Eagle CC2640 pins

Figure 29 is the actual package drawn into the Eagle libraries so we can use it on the PCB itself. The pins and all the package design was created by following the datasheet as closely as possible.

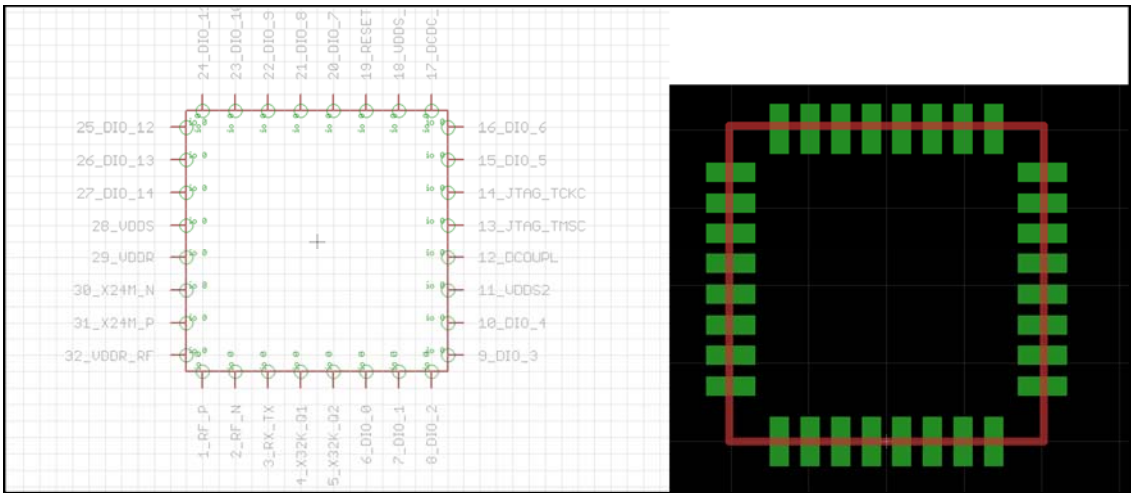

Figure 29 - Eagle PCB Board

6.1.4 Motor

The next part that was designed into the library in LtSpice was the motor. A screenshot of the motor as represented by LtSpice can be seen in Figure 30. This motor is operated by applying a high voltage from the MCU on one of the I/O pins to turn the motor on and either spin the motor clockwise or counterclockwise as labeled below in the picture of the motor.

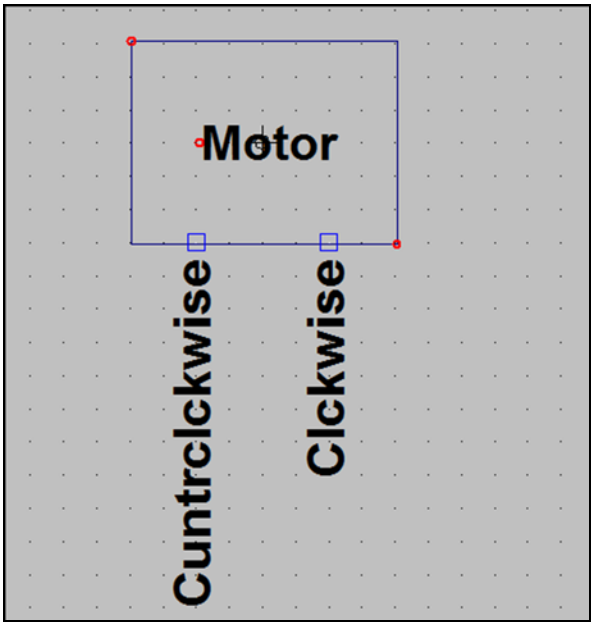

Figure 30 - LTSpice Motor Representation

These pins were hooked up to the I/O pins labeled eight and nine in the schematic of the CC2640. These are just digital I/O pins to either supply a high or a low.

6.1.5 E-Paper

This was the next most-complicated component to design into LtSpice after the microprocessor. This element of our project was intended to be used to tell the current temperature of the room and also display the charge of the battery from time to time. This display works very uniquely compared to many other displays. This uses significantly less power than other displays because it does not need constant current like a seven segment display to keep the LEDs lit. This display writes only when instructed and can hold the image that has been written for a long period of time after being written to without having to be updated for a while. This keeps power consumption down on this display and this is a major advantage seeing that we were using a battery-operated controller.

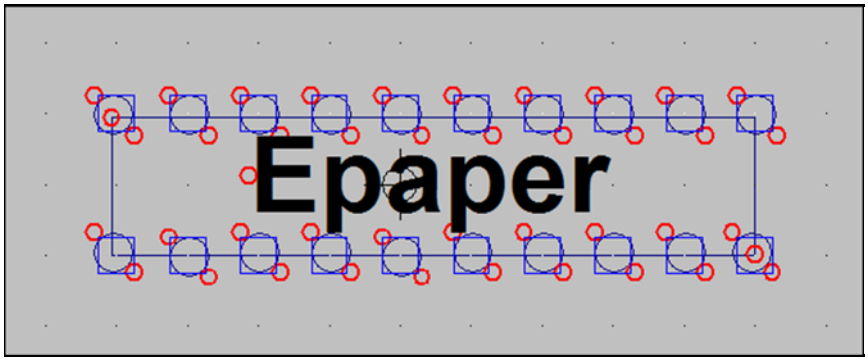

Figure 31 - LTSpice E-Paper Representation

This schematic symbol in Figure 31 represents the 24 pins on e-paper sub board. The e-paper plugs into a PCB sub board which has pin outs that will be connected to the MCU on our PCB.

6.1.6 USB Charger

We designed in a USB into LtSpice, as can be seen in Figure 32, and assigned it the necessary pins as a typical USB device. This USB is the charger to charge any non-Apple phone. The only pins that are connected in the schematic are the voltage pin and the ground pin. This will allow phones to charge but at a slower rate as to not drain the battery too fast.

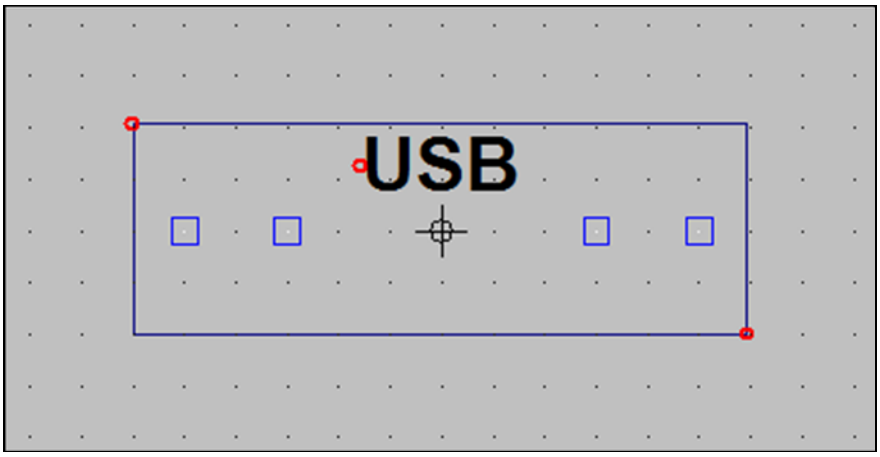

Figure 32 - LTSpice USB Charger Representation

There is another option we explored instead of buying a battery charging controller. This is designing our own circuit that would regulate the voltage of the solar panel and keep the battery charging at a constant current. The challenge would be doing this in the most efficient way. The best way to design the circuit ourselves would be to use a current limiting diode and a Zener diode. The current limiter diode would be in series with the solar panel voltage and it would drop the necessary voltage across the diode to supply a constant rated current to charge the battery at a constant current no matter what the solar panel voltage is at. The Zener diode would be connected in parallel with the battery that will be charging because it will keep the voltage in check that the battery is charging too as to make sure it doesn't over charge the battery. The battery will continue to charge until it hits the voltage of the Zener diode and then it will allow the current to run through the Zener diode and keep the battery from overcharging.

In Figure 33, we can see the diode that keeps the current constant. It does so by dropping the necessary voltage across it. This will continue to regulate the current until it hits a threshold that it will not be able to regulate the current anymore.

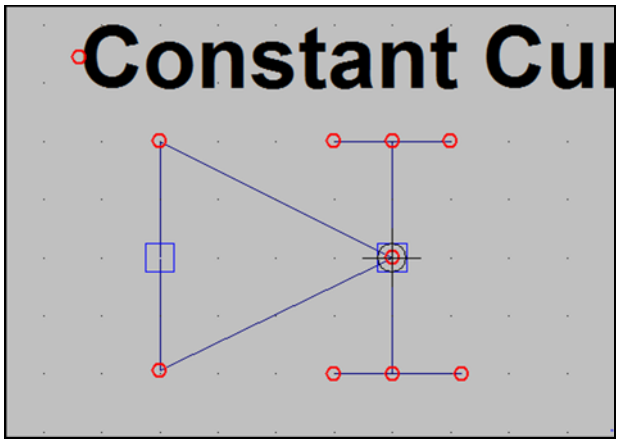

Figure 33 - LTSpice Current Stabilizing Diode

Figure 34 shows the schematic that could be used to charge the battery at a constant current and protect the battery from overcharging. The constant current diode as previously described will keep the correct current to charge the battery no matter the voltage across the solar panel. The Zener diode takes care of protecting the battery from overcharging and damaging the cells in the battery.

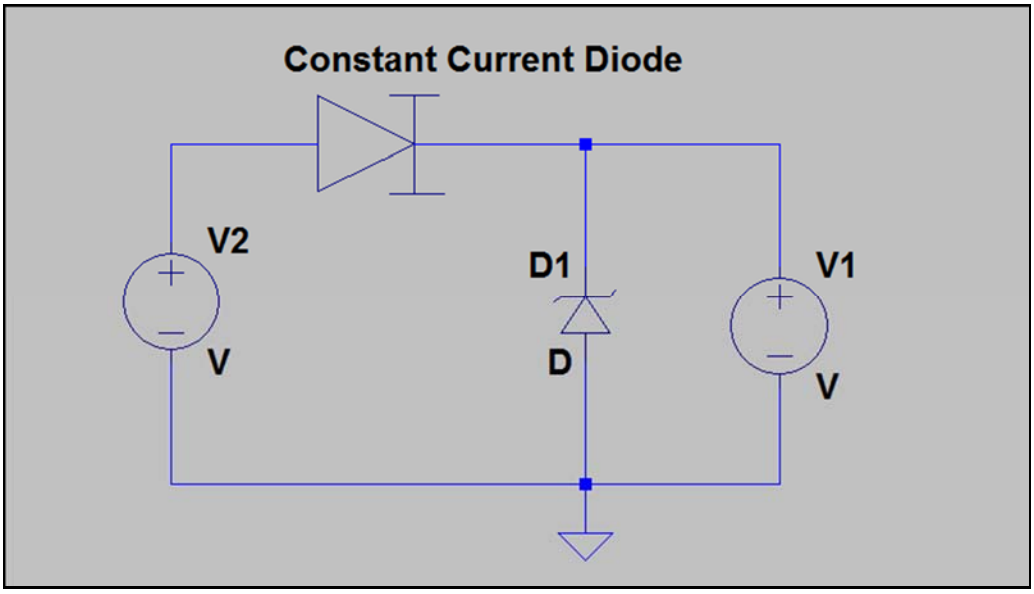

Figure 34 - LTSpice Current Stabilizing Circuit

Figure 35 is a USB jack which is a connector for the USB charger that we added to our circuit. This is attached directly onto the PCB and we used a USB extension cable (male to female) to extend the port out to the side of the housing for the user to plug their phone in.

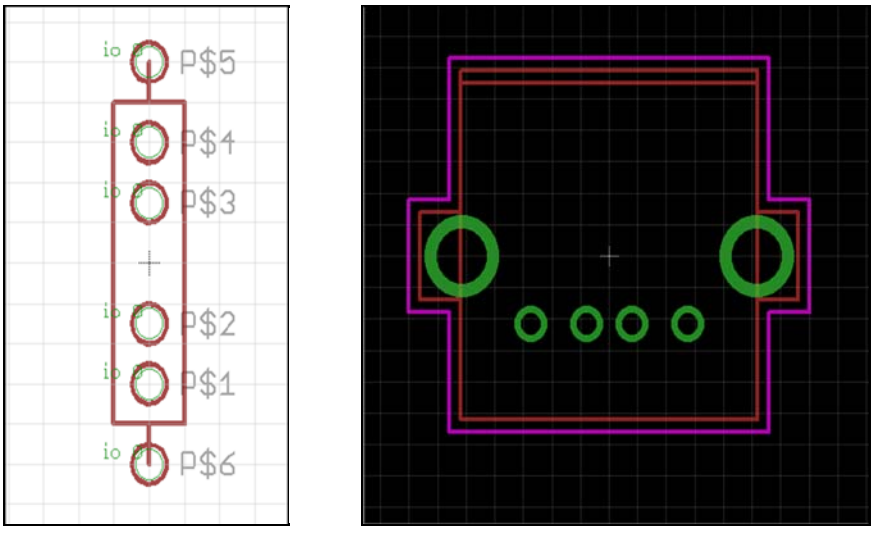

Figure 35 - Eagle USB Charge Port

6.1.7 Crystal (24 MHz)

The 24 MHz crystal is hooked up to our MCU via the two inputs for the 20MHz crystal. The LtSpice representation of this part is displayed in Figure 36. Usually, a crystal needs to have some capacitors hooked up to it to let it oscillate correctly, but this microcontroller has internal capacitors so there is no need to connect external capacitors.

Figure 36 - Eagle 24 MHz Crystal

This is the crystal that we will be using for the MCU to operate with. The schematic symbol is one that we drew up to represent the crystal and separate it from the other parts. It uses a Y to designate the crystal because there is no other part that uses this prefix and it is a commonly used notation for the crystal on many types of PCBs.

The package that we are using for the crystal can be seen in Figure 37. It is the same for all the different values of crystals with the supplier from whom we are purchasing it. The microcontroller itself can also be ran off of a 32 KHZ crystal, but we do not have the time or resources to test out both crystals values.

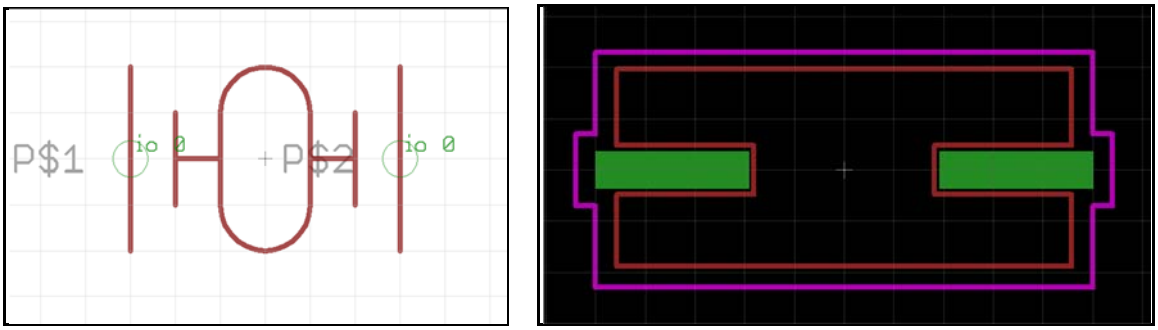

Figure 37 - Eagle 32 KHz Crystal

6.1.8 Voltage Regulator

The last element that was designed into LtSpice to complete the schematic was the MCP 1702 Linear Voltage regulator, shown in Figure 38. There were two different linear regulators that had to be added to the schematic. One of the linear regulators was a 12V to 3V regulator to power the microcontroller itself. The second linear regulator that was added was the 12V to 5V regulator. This regulator is used to regulate the 5V to power our USB and power our LCD.

The packages and pin outs for the two regulators are the, they just are ordered with different part numbers to get the two different desired outputs. Pin one is the ground, while pin two is the voltage input, and pin three is the voltage output. So in the case of the 3V regulator, the voltage output pin would have the desired 3V output [50].

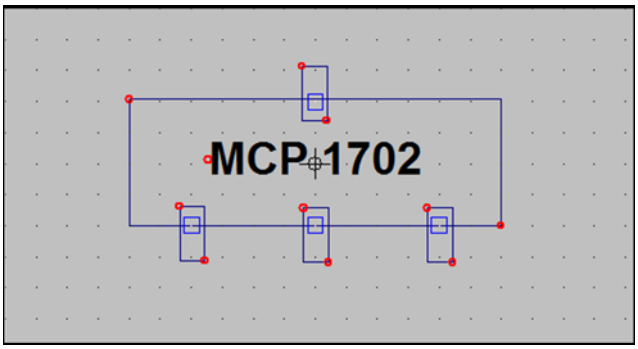

Figure 38 - Eagle's Representation of MCP 1702 Pins

The diagram in Figure 39 is a layout of the regulators pins and the package.

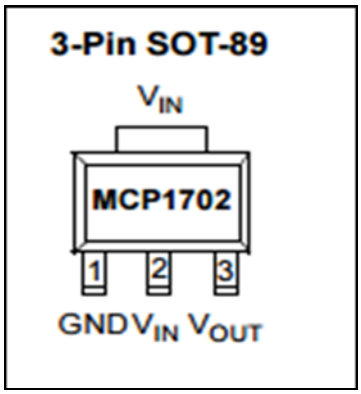

Figure 39 - Voltage Regulator Pins and Package

The MCP 1702 is another component that had to be added into our library. We were able to draw it up in Eagle, as can be seen in Figure 40. This component has two different devices that have the same symbol for the schematic and the same layout for the PCB.

There are two different values of the regulators which is the reason why there will be two different devices. The first device will be the version of the MCP 1702 that will be a linear regulator that regulates from 12V to 3V. The second device will be another version of the MCP 1702 that will regulate the 12V to 5V. The packages will be the same for each of these.

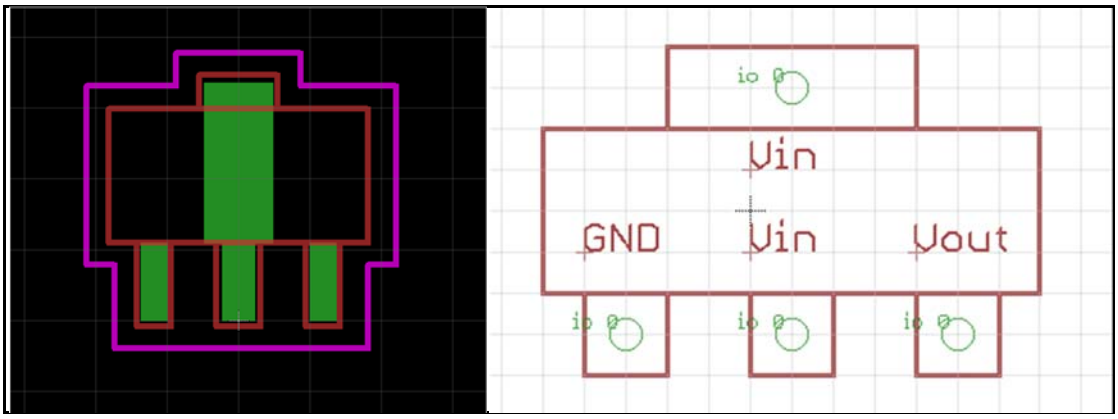

Figure 40 - Eagle Representation of MCP 1702

6.1.9 Antenna Circuit

The diagram in Figure 41 shows different options for the circuit that can be used for the different antenna schemes. We looked into the different ones, but we were not sure which one we were going to use for our design. In the end, we decided to use a trace antenna which is built into the MCU. This meant we did not need to worry about the design for an external antenna circuit.

The first type of antenna circuit we had to choose from was the Differential Operation circuit. This type of circuit is the most favored because it is the best performance wise, but it requires external biasing which can take up more board space and may be a bit more costly. The Single-ended Operation is another option we have to consider and it is biased internally and has the lowest power consumption which seems like something that we want the most. This seems to be the option that most suits our needs because it uses the least amount of power to run, but once again we will need to wait to test the different schemes because since it uses less power, it may not have the same range as the other option and we made need the range depending on how far the Bluetooth reaches [51].

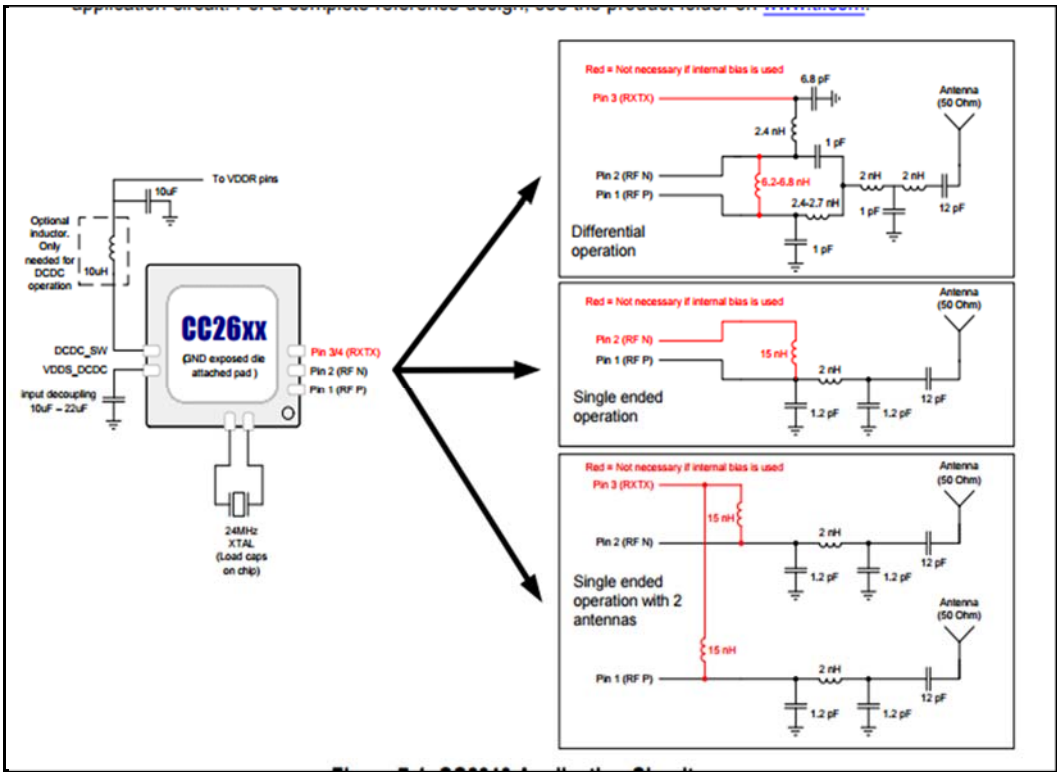

Figure 41 - Antenna Configurations

6.1.10 Complete Schematic

The schematic in Figure 42 shows all of the required connections from all of the previous components that were designed into this LtSpice Schematic. Together, this schematic should be able to do all of the functions that we were hoping to complete correctly and most efficiently. The only parts that had to be tested to add to this schematic were the antenna circuit and a few different connections to the e-paper display. Fortunately, we decided to use the trace antenna and an LCD instead, so this ended up being a non-issue.

The e-paper pinouts were displayed on Digikey on the sub board [52], but there was no documentation as to what pins needed to be hooked up where and what pins do what other than the one word descriptions of the pins function. This is the main reason we decided not to use the e-paper display. We simply did not have enough time to mess with it while getting all the other components operational.

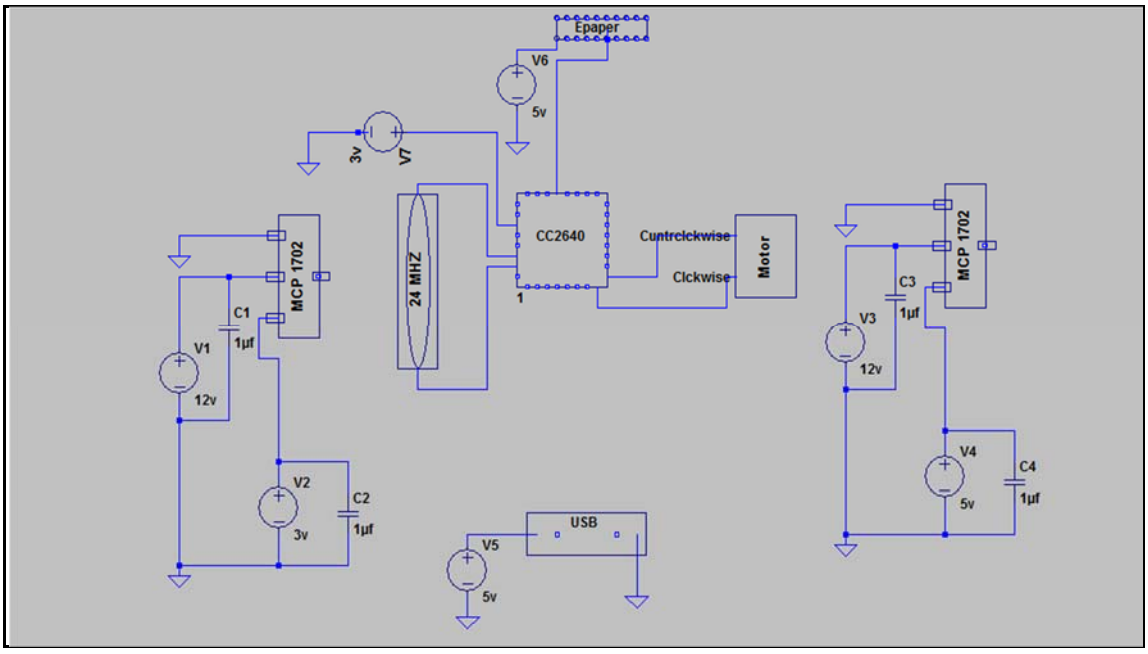

Figure 42 - Complete Schematic Represented in Eagle

6.2 Solar Technologies

In order to maximize the power capable of being captured by the system while maintaining the roller shade design, a combination of crystalline solar cells and amorphous silicon solar cells were used. The total power of the system is about 34.6 W.

Sunpower 3.3 W 6”x6” monocrystalline photovoltaics were attached to the back of the project box used to store the battery and circuit boards. A set of 10 of these cells can be purchased on eBay for $46.99. Connecting six of the cells in series produces a 20.6 W solar module operating at a maximum voltage of 3.48 V and a maximum current of 5.93 A. The cells needed to be tabbed using 2mm x 0.15mm tabbing wire (MISOL 10m: $6.28 on Amazon), a rosin flux pen (MG Chemicals 835-P: $8.95 on Amazon), and lead-free silver solder (Trakpower Rosin Core Lead Free Silver Solder: $11.93 on Amazon).

A DC-DC boost converter was needed in order to step up the voltage of the solar module to match that of the amorphous silicon solar module. This allows the two modules to be connected in parallel without suffering significant power losses. The DC-DC LTC1871 step up module worked well for this conversion. The circuit costs $10.04 and allows a user to select the input and output voltages. Table 18 shows the specifications of this device.

Measure	Value
Input voltage	3.5 – 30 V
Maximum output power	100 W
Output voltage	3.5 – 30 V
Maximum input current	10 A
Power consumption	15 mA

Table 18 - LTC1871 Specifications

The output of the LTC1871 has a maximum voltage of 15.4 V and a maximum current of 1.34A. Figure 43 shows the complete diagram of the monocrystalline solar module including the boost converter.

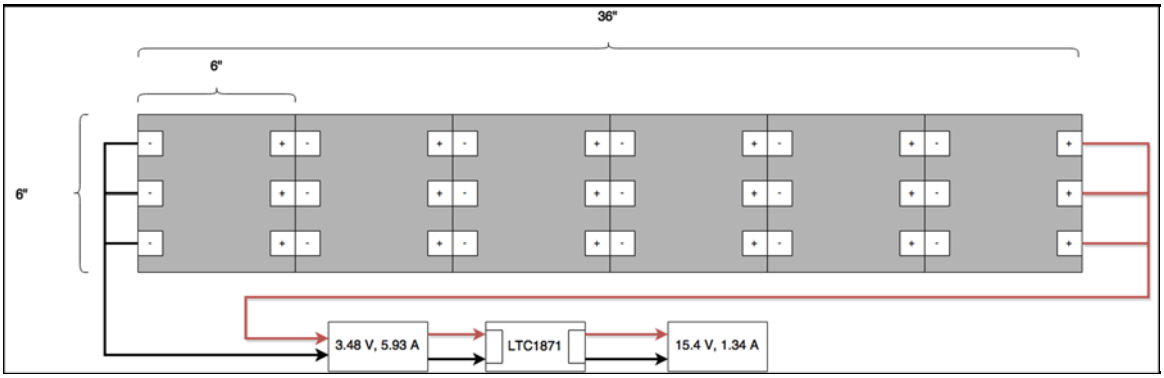

Figure 43 - Monocrystalline Solar Module

The Powerfilm R-14 rollable amorphous silicon solar module measures 14.5”x42” and costs $185.99. The panel has a total average power of 14 W operating at a maximum voltage of 15.4 V and a maximum current of 900 mA. Adding the 14 W amorphous silicon rollable solar panel brings the total power of the system to 34.6 W. Connecting the modules in series provides a maximum voltage of 15.4 V and a maximum current of 2.24A. A diagram of the system described is shown in Figure 44.

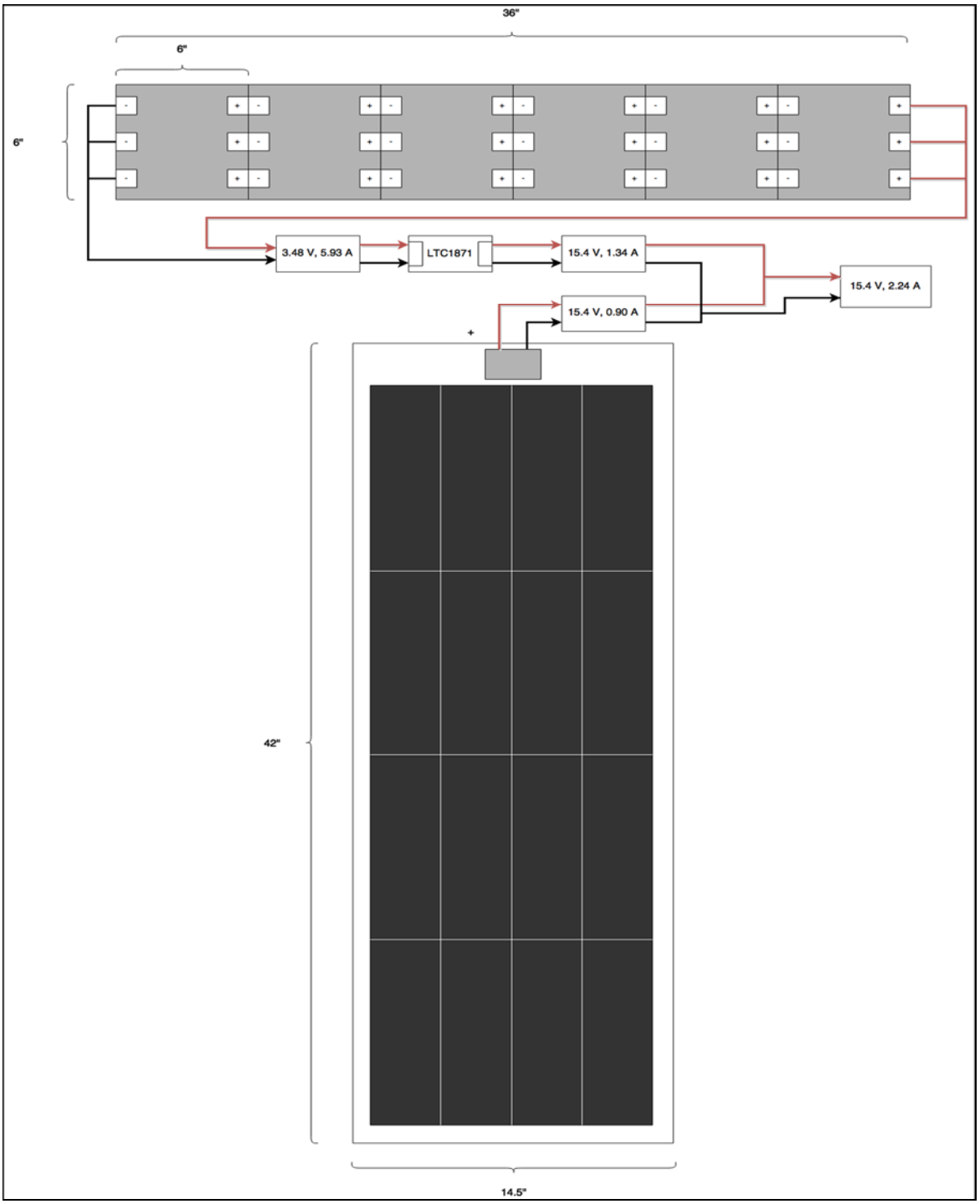

Figure 44 - Combined Solar Module

This system is capable of capturing the amount of energy capable of being stored by the 60Wh battery in 1.73 hours under ideal lighting conditions. The Universal Power Group UB250 12V 5Ah SLA AGM rechargeable battery will be used for the project. The battery costs only $9.93 which is much more affordable than the equivalent lithium-ion batteries on the market. The battery is rated to have a cycle life of 200 cycles when the

battery is 100% discharged, 500 cycles when the battery is discharged 50%, and over 1200 when the battery is discharged 30% before recharging. Assuming the battery is discharged to 50% every night and recharged to 100% of its capacity each day, the battery should last about 500 days before it needs to be replaced. Although the lifetime of the lithium-ion battery would be about twice as long, it costs eight times as much as the lead acid battery. Thus, the lead acid battery is the more economic choice. The nominal capacity of the battery varies with the discharge current. Figure 45 shows this relationship graphically. If the loads require a constant high current then the battery will perform much worse than 5 Ah.

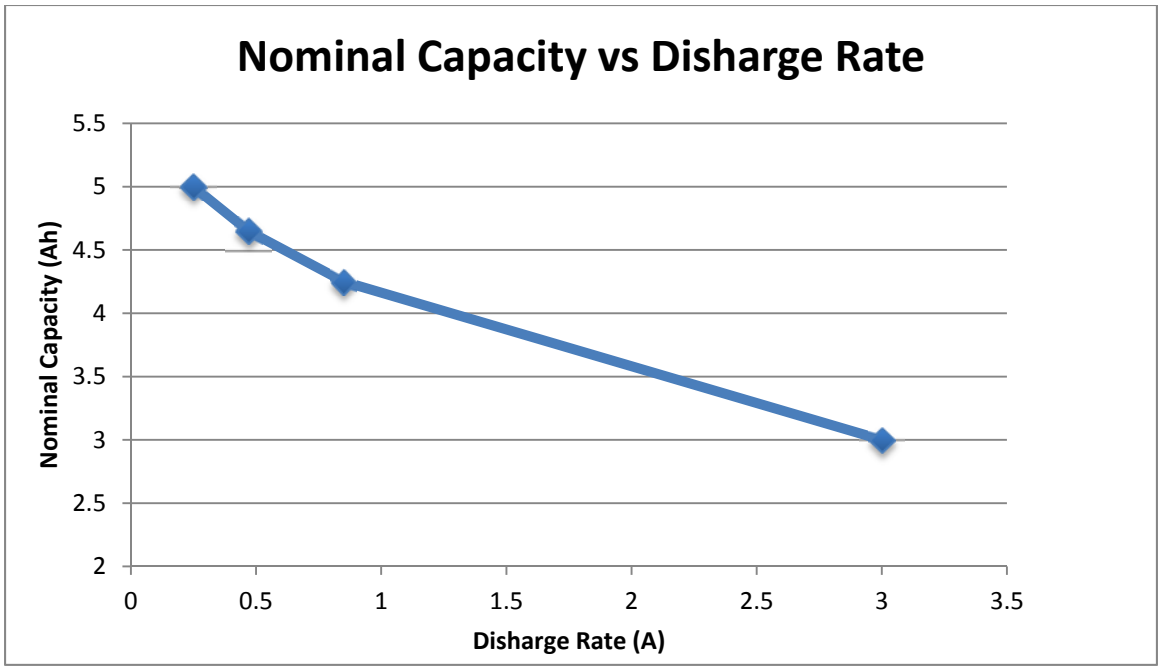

Figure 45 - Nominal Capacity vs Discharge Rate for SLA Battery

The battery measures 3.54”x2.76”x3.98” and weighs 3 lbs, which does not fit well in the 6”x6”x36” project box. For this reason, we did not include space for it inside the housing and simply use it as an external battery. The maximum charge current for the battery is 0.35C where C is the number of cells in the battery. A 12V SLA has six cells, so the maximum charge current for this battery is 2.1A. This is slightly less than the 2.24A maximum output current of the solar module. The MPPT charge controller was used to limit the output current of the solar module so that the maximum charge current of the battery is not exceeded.

The Genasun GV-5 65W 5A MPPT solar charge controller for lead-acid batteries was used. The controller costs $75.00 and has a maximum recommended panel power of 65 W. Inputs include a 27V maximum panel voltage input, a 12 battery input and a load input with a continuous rated load current of 5A. Although it is rated for 27V, a 22V maximum panel voltage is recommended under standard operating conditions. This is well above the 15.4V designed output voltage of the solar module. The minimum

battery voltage for normal operation is 7.2V. The maximum input current is 9A which is well above the 2.24A maximum output current of the solar module. A computer controlled four-stage battery charging profile is used in order to increase the battery life and maximize capacity. This includes an absorption voltage of 14.2V, a float voltage of 13.8V, and a load disconnect voltage ranging from 11.4V to 12.5 V. A battery temperature compensator is also included which adjusts the voltage at a rate of -28mV/℃. The controller utilizes an MPPT tracking speed of 15 Hz in order to adapt quickly to changing light conditions. This results in an electrical efficiency of 96% - 99.85% and a tracking efficiency of 99+%. The controller uses ceramic components instead of electrolytic components in order to increase the product lifetime. A 10-year warranty is included with purchase. The GV-5 consumes 0.150 mA when operating and 0.125 when asleep. It weighs only 2.8 ounces and measures 4.3”x2.2”x0.9”. A built-in electronic protection circuit cuts power when a short circuit is detected. This will protect the controller from damage if the polarities of any of the inputs are accidentally reversed.

Table 19 shows the bill of materials for the complete solar sub-system.

Manufacturer	Item	Seller	Price
Sunpower	3.3 W 6”x6” Monocrystalline Solar Cells (10)	eBay	$46.99
MISOL	2mm x 0.15mm Tabbing Wire (10 m)	Amazon	$6.29
MG Chemicals	835-P Rosin Flux Pen	Amazon	$8.95
Trakpower	Rosin Core Lead-Free Silver Solder	Amazon	$11.93
DROK	DC-DC LTC1871 Step Up Module	GearBest	$10.04
Powerfilm	R-14 Rollable Solar Module	FlexSolarCells	$185.99
Universal Power Group	UB250 12 V 5 Ah SLA AGM Battery	1000Bulbs	$9.93
Genasun	GV-5 MPPT Charge Controleller	Genasun	$75.00
Total			$355.12

Table 19 - Solar Sub-System Bill of Materials

Figure 46 shows the diagram of the complete solar sub-system. This connects all of the solar cells together and with the housing.

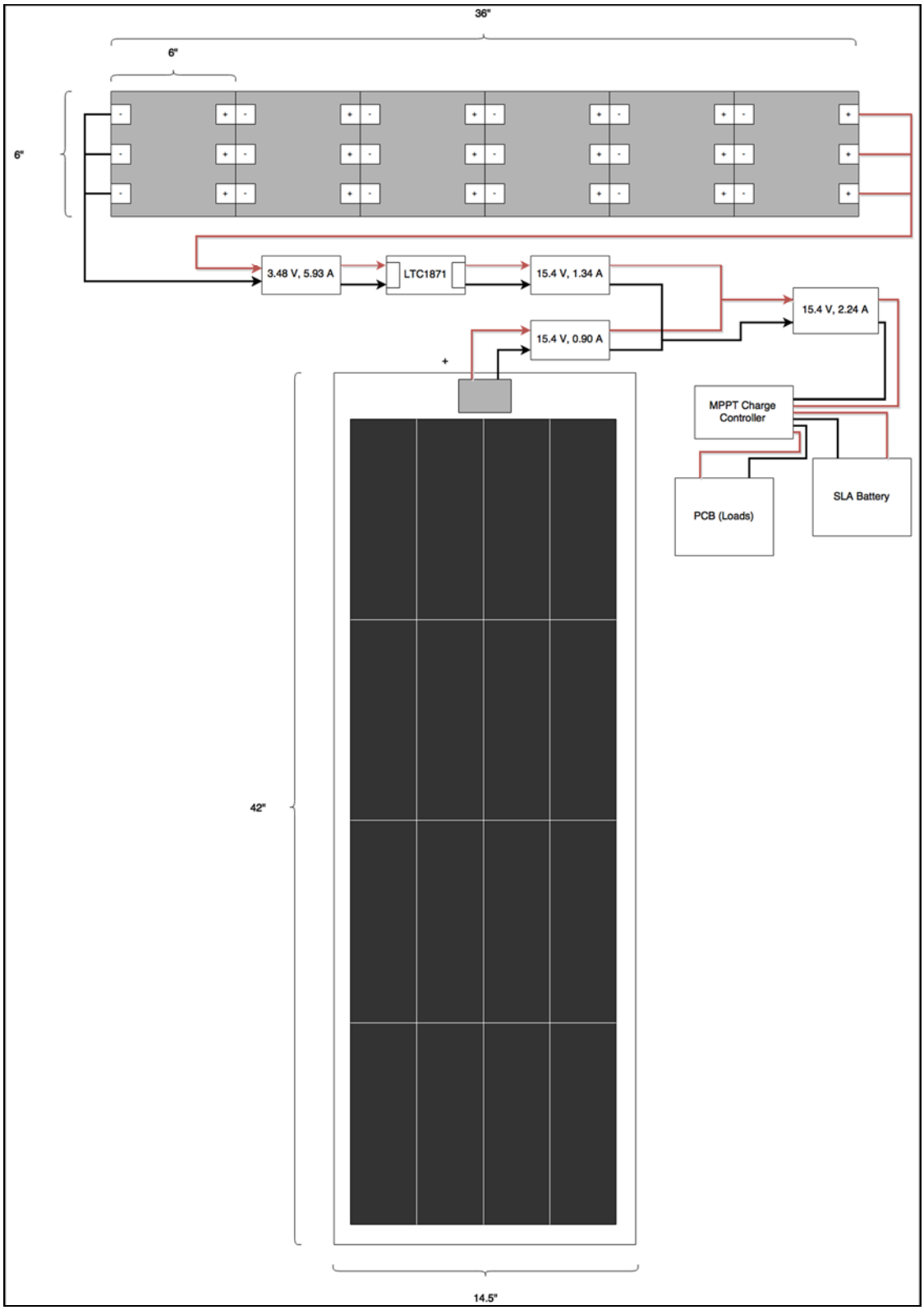

Figure 46 - Complete Solar Sub-System

6.3 Embedded Hardware

As discussed previously, in section 3.4.1, we decided to use ARM architecture. With one of the main goals of this project focusing on energy efficiency and power conservation, these were large factors for our decision of an MCU. Additionally, we also knew that we needed remote connection capability.

A good energy-efficient option presented itself in the Atmel SAM3N00A. This utilizes the ARM Cortex-M3 architecture, which falls into the category of Harvard architectures used by ARM. It is also one of the low power options available by Atmel. Normal operation requires 1.62-3.6 Volts, with power consumption shown in Figure 47, courtesy of Atmel.

Mode	SUPC, 32 kHz Osc., RTC, RTT, GPBR, POR (Backup Region)	Regulator	Core Memory Peripherals	Mode Entry	Potential Wake-up Sources	Core at Wake-up	PIO State While in Low Power Mode	PIO State at Wake-up	Consumption (2) (3)	Wake-up Time(1)
Backup	ON	OFF	OFF (Not powered)	WFE + SLEEPDEEP = 1	WKUP0–15 pins RTC alarm RTT alarm SM alarm	Reset	Previous state saved	PIOA & PIOB & PIOC inputs with pull ups	3 µA typ(4)	< 0.1 ms
Wait	ON	ON	Powered (Not clocked)	WFE + SLEEPDEEP = 0 + LPM bit = 1	Any event from - Fast startup through pins WKUP0–15 - RTC alarm - RTT alarm - SM alarm	Clocked back	Previous state saved	Unchanged	5 µA/15 µA (5)	< 10 µs
Sleep	ON	ON	Powered(7) (Not clocked)	WFE or WFI + SLEEPDEEP = 0 + LPM bit = 0	Entry mode = WFI Interrupt Only; Entry mode = WFE Any enabled interrupt and/or any event from - Fast startup through pins WKUP0–15 - RTC alarm - RTT alarm - SM alarm	Clocked back	Previous state saved	Unchanged	(6)	(6)

Figure 47 - SAM3N00A Power Consumption

The SAM3N00A did not have remote connection capability. In order to achieve this, we would require an additional module to supply this functionality. Atmel has several chips available for this. The most energy-efficient Wi-Fi module by Atmel is the ATWILC1000, whose power consumption can be seen in Figure 48, courtesy of Atmel.

Device State	CHIP_EN	VDDIO	Power Consumption[1]	
			I_{VBATT}	I_{VDDIO}
ON_Transmit	VDDIO	On	230mA @ 18dBm	29mA
ON_Receive	VDDIO	On	68mA	29mA
ON_Doze	VDDIO	On	280µA	<10µA
Power_Down	GND	On	<0.5µA	<0.2µA

Figure 48 - ATWILC1000 Power Consumption

An alternative to the Atmel MCU is the CC3200 made by Texas Instruments. The CC3200 uses an ARM Cortex-M4 processor. The main advantage of this chip over Atmel is that it has a built-in Wi-Fi module. This would significantly reduce the time and effort required for installation of the MCU and setup of Wi-Fi. The tradeoff with this chip is that it uses

more power than the Atmel. The active power consumption for the CC3200 can be seen in Figure 49, courtesy of Texas Instruments.

PARAMETER		TEST CONDITIONS[1] [2]			MIN	TYP	MAX	UNIT
MCU ACTIVE	NWP ACTIVE	TX	1 DSSS	TX power level = 0		278		mA
				TX power level = 4		194		
			6 OFDM	TX power level = 0		254		
				TX power level = 4		185		
			54 OFDM	TX power level = 0		229		
				TX power level = 4		166		
		RX	1 DSSS			59		
			54 OFDM			59		
	NWP idle connected[3]					15.3		

Figure 49 - CC3200 Active Power Consumption

As we can see, the current can reach pretty high values while the chip is active. The power consumption for the CC3200 while in sleep mode can be seen in Figure 50, courtesy of Texas Instruments.

PARAMETER		TEST CONDITIONS[1] [2]			MIN	TYP	MAX	UNIT
MCU SLEEP	NWP ACTIVE	TX	1 DSSS	TX power level = 0		275		mA
				TX power level = 4		191		
			6 OFDM	TX power level = 0		251		
				TX power level = 4		182		
			54 OFDM	TX power level = 0		226		
				TX power level = 4		163		
		RX	1 DSSS			56		
			54 OFDM			56		
	NWP idle connected[3]					12.2		
MCU LPDS	NWP active	TX	1 DSSS	TX power level = 0		272		mA
				TX power level = 4		188		
			6 OFDM	TX power level = 0		248		
				TX power level = 4		179		
			54 OFDM	TX power level = 0		223		
				TX power level = 4		160		
		RX	1 DSSS			53		
			54 OFDM			53		
	NWP LPDS[4]					0.25		
	NWP idle connected[3]					0.825		
MCU hibernate[5]	NWP hibernate[6]					4		µA
Peak calibration current [7]		V_{BAT} = 3.3 V				450		mA
		V_{BAT} = 2.1 V				670		
		V_{BAT} = 1.85 V				700		

Figure 50 - CC3200 Sleep Power Consumption

For Wi-Fi options, the Atmel is the better choice as far as power-consumption is concerned. However, for Bluetooth options, Atmel does not have any Bluetooth modules to connect to its MCUs. However, Texas Instruments has six wireless MCUs with Bluetooth capability built-in. Of these six chips, the lowest-power option is the CC2640 which is marketed as "ultra-low power wireless MCU for Bluetooth Smart". The current usage of the CC2640 can be seen in Table 20, courtesy of Texas Instruments.

Active	Idle	Standby	Shutdown	Reset Held
1.45 mA + 31 µA/MHz	550 µA	1 µA	0.15 µA	0.1 µA

Table 20 - CC2640 Power Consumption

Clearly, the Bluetooth MCU is hundreds of times more power-efficient than the Wi-Fi option. We decided to use the CC2640 for our project, more specifically, the CC2640F128RHBR because it is the cheapest model at the sacrifice of a few IO pins. With 33 pins, we did not expect to run into any problems. We later learned that not all of these pins can be used as IO, as many of them are labeled as “ground” or “no connect” but we still had enough pins to complete our goals.

6.4 Mobile Application

This section of the report will establish the concepts of operation for the Solar Blinds mobile application. Described below are the design details and processes in the mobile application design. We will be discussing which coding language was used, the environment it was built in, the high level architecture, and the specific features and functionality of the mobile application at a low level.

6.4.1 Coding language

For our mobile application system the developers chose to pursue the development of the Solar Blinds application using C# as the primary language. If time constraints were going to be too strict then we would fall back to using Java since we were more familiar with it. However, the difference between C# and Java was very small and we were able to use it without many problems.

Mobile applications are normally developed in an object-oriented language, and we felt that expanding our experience into a new programming language would be a great benefit to us. Up until now, we had only been taught MIPS assembly, C programming, and Java programming as part of our standard curriculum. Although C# may not be much different from Java, it is used for many different applications and potential employers would be happy to see that we have prior experience in this language.

6.4.2 Build Environment

The build environment of the mobile application was done in the free version of Xamarin Studio. This environment allowed the developers to use C# as their primary language. All source code and any other files will be managed using Xamarin Studio and preexisting folder structure since it is able to readily deploy code into working app bundles.

6.4.3 High Level Architecture

For our application's high level architecture we used a mix of layering and repository styles. The layering style representation is used because the data has to go through multiple layers and processed individually at each specific layer. Upon reaching the lowest layer, the I/O, Bluetooth, NFC, and RFID management components begin to interface with the actual mobile device and Solar Blinds. When the information is finally processed it goes to the user layer which can branch out into many differ states depending on the option screens selected. Using a GUI, an assortment of screens are used to display the different menu items the users can select from the home screen.

This makes the GUI one of the highest levels the data will reach. This is because the screen actually displays the data to the user but it does not allow user options to make changes to the solar blinds data internally. The data is only represented in these screens. The user will use the GUI to send signals to get or push data to the Solar Blinds. The only options the user will have on the app screen are to enter the app or double press the back button to close the app. At the menu layer the scheme will become linear in manner since it starts to act as a repository for the different types of data being pass to and from many different classes. The diagram in Figure 51 shows how all of the interaction between the mobile app and the physical components will take place.

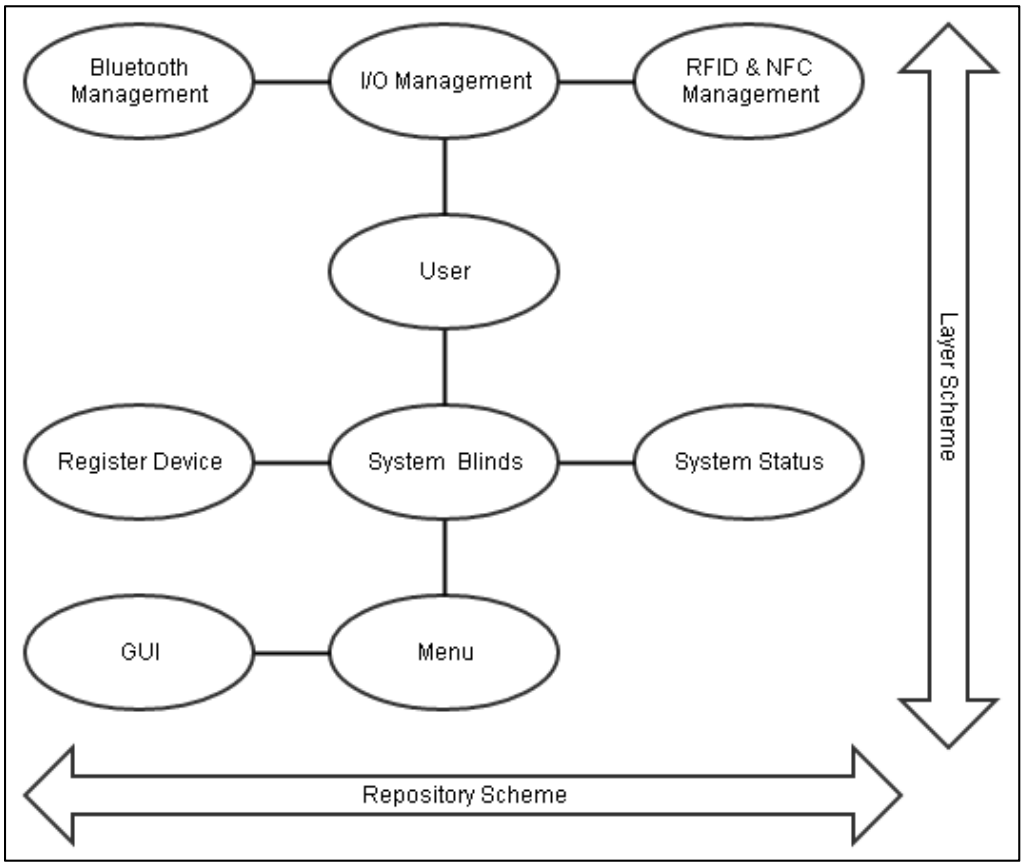

Figure 51 - Major Components and System Interfaces

6.4.4 Activity Descriptions

The activity diagram shows all of the user's actions they are able to accomplish using the mobile application and the Solar Blinds together, shown in Figure 52. These actions also represent the requirements needed to take this action and scenarios when the actions are available to be used by the user. If the user attempts any unpermitted actions the mobile application will inform the user that there is an error and show the steps that are needed to fix this.

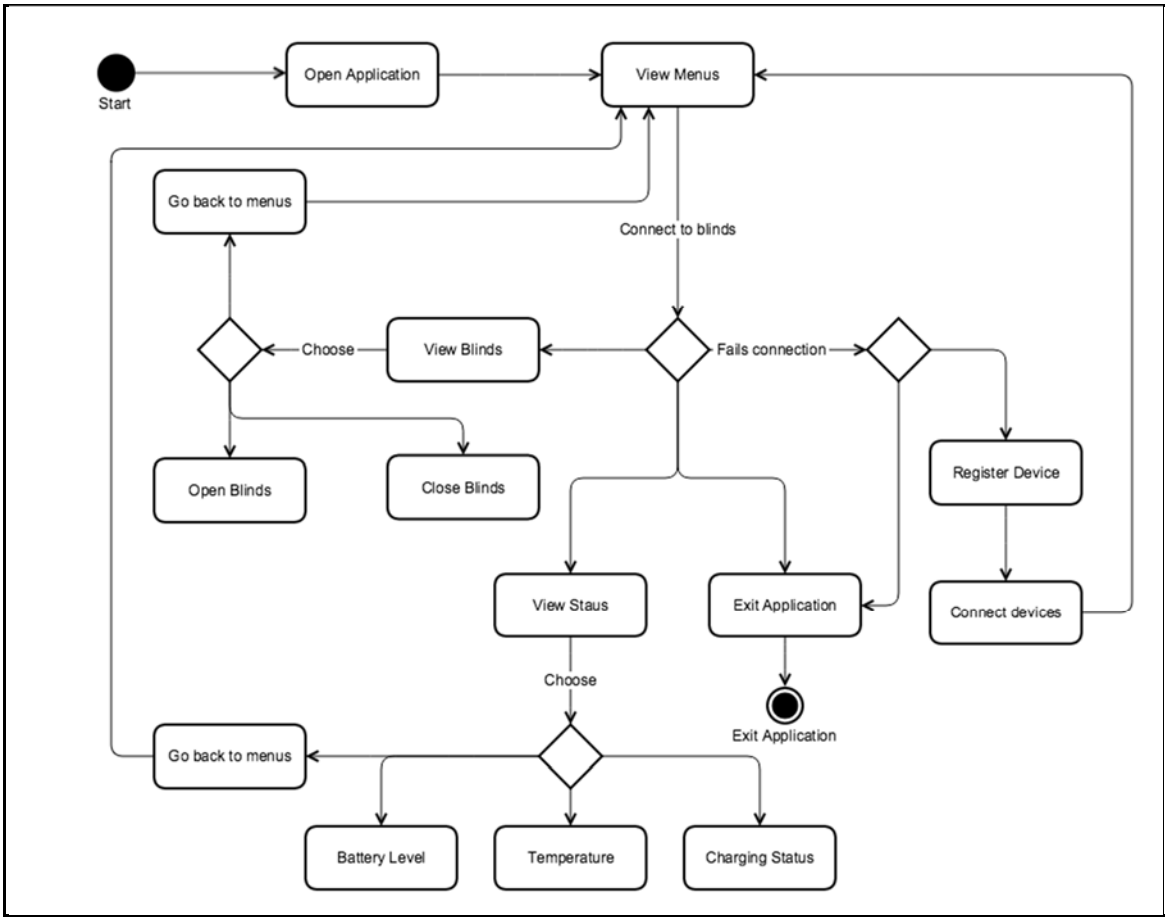

Figure 52 - Activity Diagram of the Mobile Application

6.4.5 Use Case Descriptions

This system will allow a user to operate directly with the Solar Blinds via our mobile application. In order to do this, the user must maintain certain requirements and cannot alter these specified procedures. Any user can access the mobile application by simply opening it up and viewing the menus. However if the user attempts to do anything besides just viewing the menu the action will fail. This is because in the use case the user can open the application and view the menu but when they do those tasks they must connect the device. This is to prevent anyone from randomly controlling the Solar

Blinds without proper authorization. When registering the device the user will have two options they can either select connect their device or disconnect their device. Remember that in order to actually operate the blinds the user must be registered to the Solar Blinds via RFID. This is why in the use case diagram in Figure 53 includes "connect device" for any action that will either operate or show the internal data of the Solar Blinds.

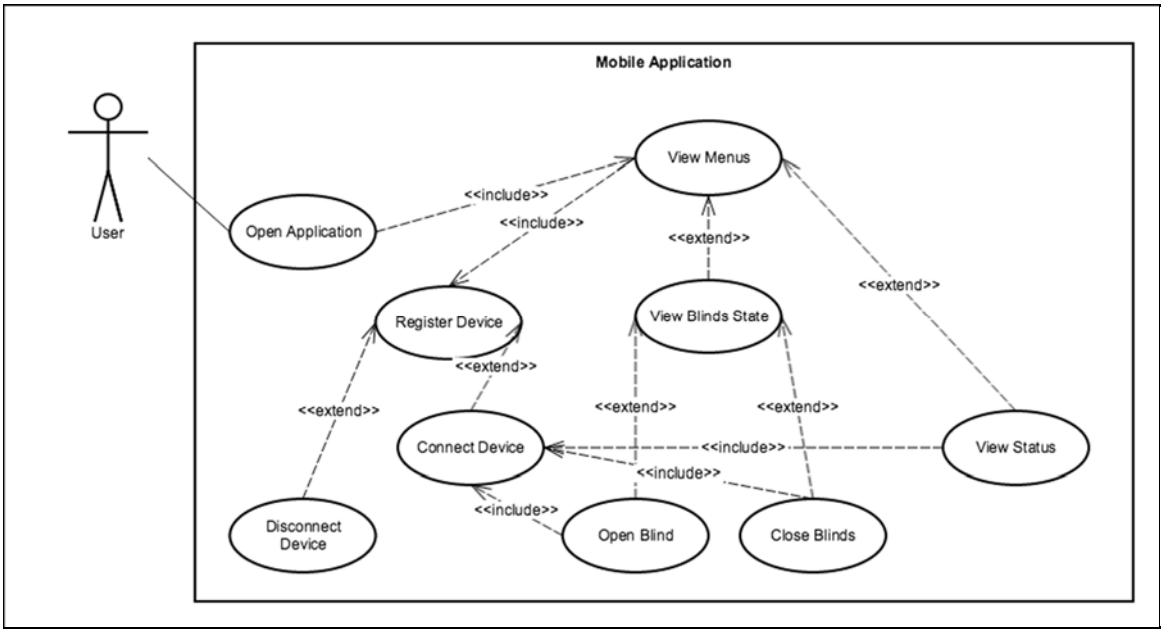

Figure 53 - Use Case Diagram of the Mobile Application

6.4.6 Mobile Application Screen

6.4.6.1 Menu Screen

After showing the app cover screen the user is taken directly to the menu screen without the need of user input. The menu screen shown in Figure 54 acts as the GUI navigation point to reach all of the other features of the mobile application. All of the buttons feature a normal state and an active. The buttons shown on the menu screen are:

- Register Device
- System Blinds
- System Status

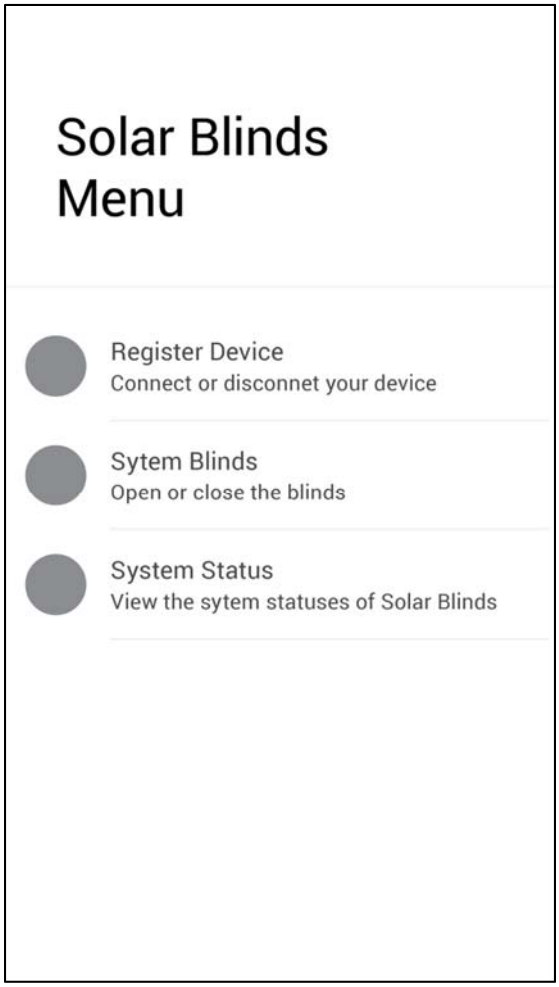

Figure 54 - Menu Screen Concept

6.4.6.2 Register Device Screen

This screen, shown in Figure 55 gives the user the ability to register the Solar Blinds device and mobile application with their mobile phone in order to view the output information and to send data back to the Solar Blinds with their mobile device. Using NFC and Bluetooth technology pressing on the buttons will initiate the mobile application to begin sending signals via Bluetooth to the Solar Blinds. The buttons shown on the register device screen are:

- Connect Device
- Disconnect Device

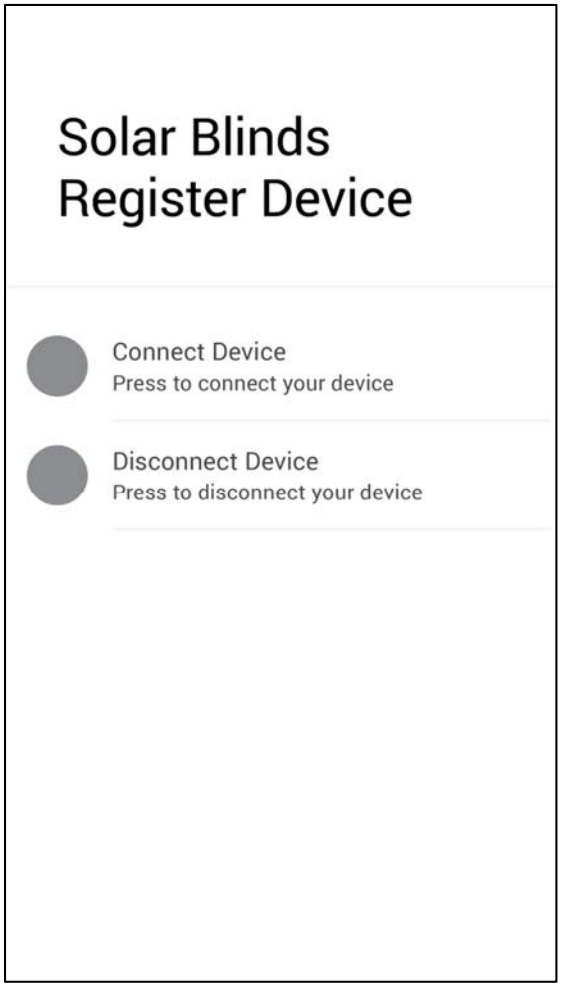

Figure 55 - Device Sync Screen Concept

6.4.6.3 System Blinds Screen

This screen, shown in Figure 56, is used to actually control the blinds from your mobile application. When the mobile application is connected to the Solar Blinds the screen indicates the current status of the blinds with a green dot (shown in the appendix). Pressing the button will switch between the two possible states. If the Solar Blinds are already in the open state, pressing the open button again will do nothing. This is to prevent malfunction of the motor. The buttons shown on the system blinds screen are:

- Open Blinds
- Close Blinds

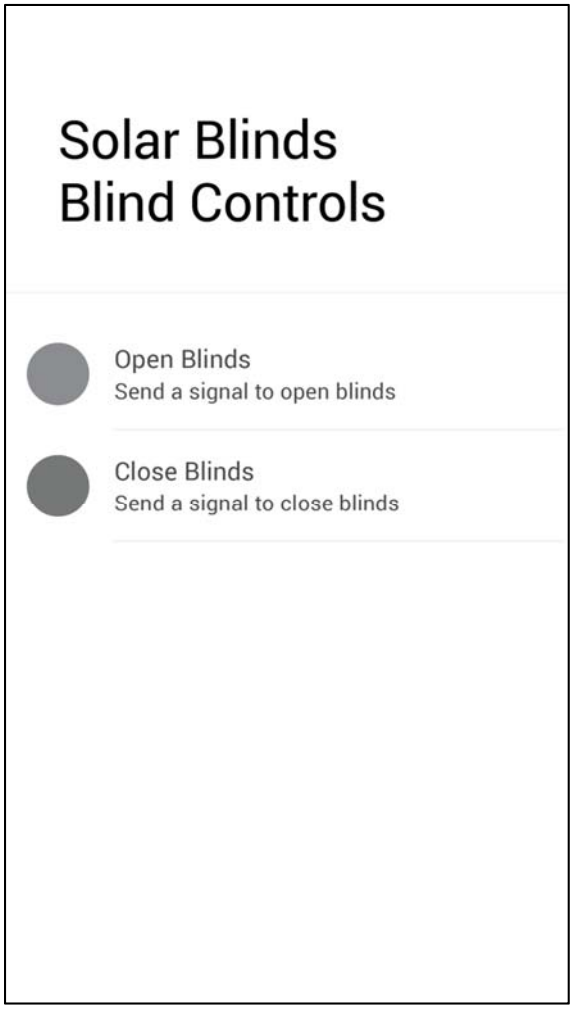

Figure 56 - Solar Blinds Screen Concept

6.4.6.4 System Status Screen

This screen, shown in Figure 57, is used to display the status of the internal components of the Solar Blinds system. The user would not be truly interacting directly with the Solar Blinds device at this point because the mobile application is only receiving data as opposed to before when it was doing both. The buttons shown on the system blinds screen are:

- Battery Level
- Temperature
- Charging Status

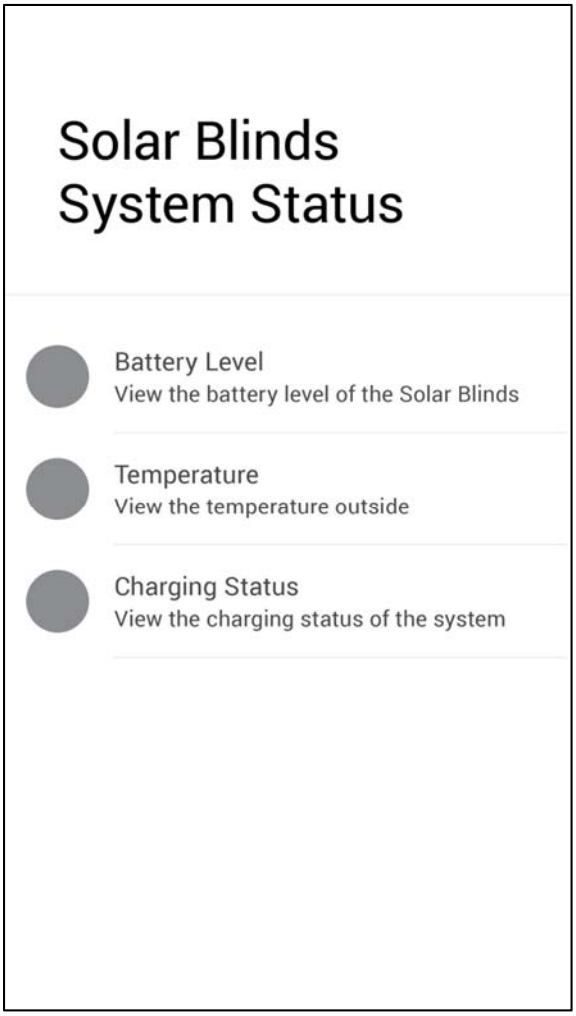

Figure 57 - System Status Screen Concept

6.4.7 Low Level Design

6.4.7.1 Graphic User Interface

This handles all of the graphics displayed. The different screens were broken down more in depth above. Please reference them to see more information. The input, output, attributes, and actions associated are:

- Inputs: Sensors (touchscreen technology), Solar Blinds Statuses
- Outputs: Graphics of the Solar Blinds data and statuses
- Attributes: The different screens
- Actions: Display all of the graphic data onto the mobile application screens

6.4.7.2 Input/Output

This handles the raw data collection and sends it to the other components in the system. For example, the touchscreen sending information to the Solar Blinds when the

user presses the close blinds button on the System Blinds screen. The input, output, attributes and actions associated are:

- Inputs: Data from the device in regards to the touch screen, Bluetooth, RFID and NFC
- Outputs: The data in form of graphic representation and forms to be transferred via Bluetooth and NFC
- Attributes: Touch sensor
- Actions: Get the data from the device and send to the read of the system

6.4.7.3 Mobile Application Prototype

This section contains a graphical overview of the concept design for the Solar Blinds mobile application. This prototype includes images of the mobile application along with the screens representing the active state buttons (toggled on) for each of the buttons that can be used to take the user to another screen or even cause an interaction signal to be sent to the Solar Blinds or the mobile application. All of these screens are discussed in detail in the 9.2 Software Operation section of the report.

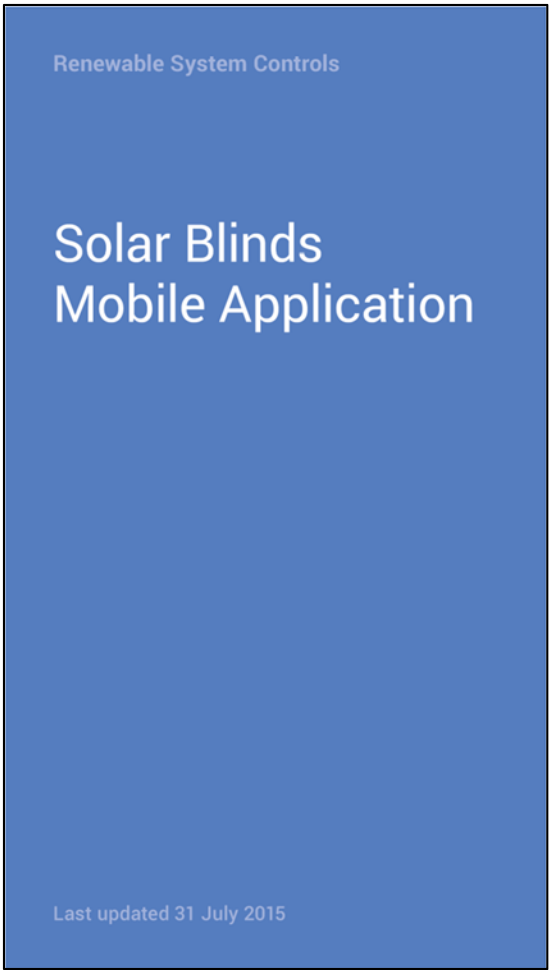

Figure 58 - App Cover/Intro Screen

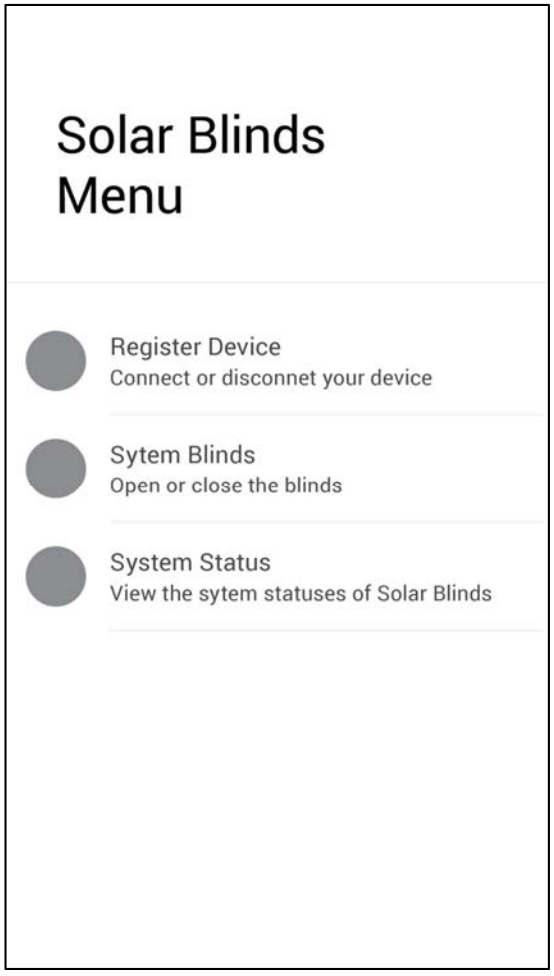

Figure 59 - Main Menu

The mobile application's cover screen is shown in Figure 58. This screen is displayed when the app starts and is displayed until the application loads.

In Figure 59, next to the cover screen, is a concept of the main menu. The menu screen of the mobile application provides a GUI for the main navigation of the mobile application.

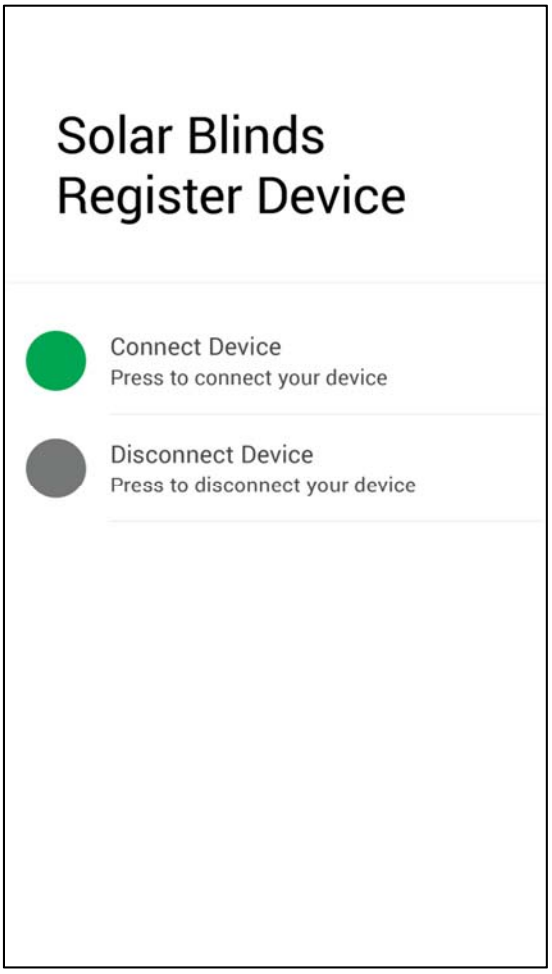

Figure 60 - Sync Screen

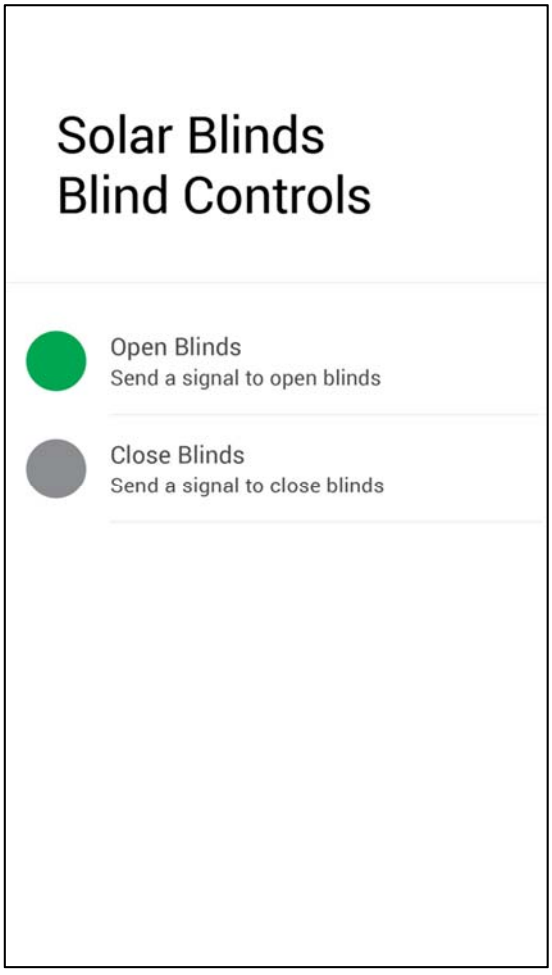

Figure 61 - Blind Control Menu

The register device screen in Figure 60 is used to connect or disconnect the user's mobile device. The green button represents that the button is currently being pressed.

The system blinds screen in Figure 61 is where the user will go to open and close the blinds. Likewise here, the green button represents it is being pressed.

We later added a list of devices synced to the phone. This list allows the user to sync to multiple devices and then select which one to control. A long press of the list items opens a menu that allows adding a nickname or deleting the registration.

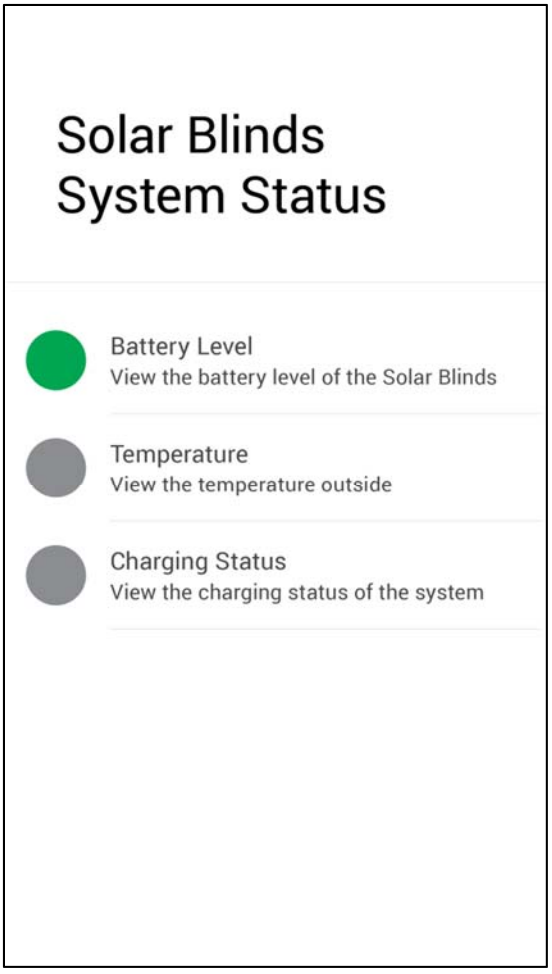

Figure 62 - System Status Menu

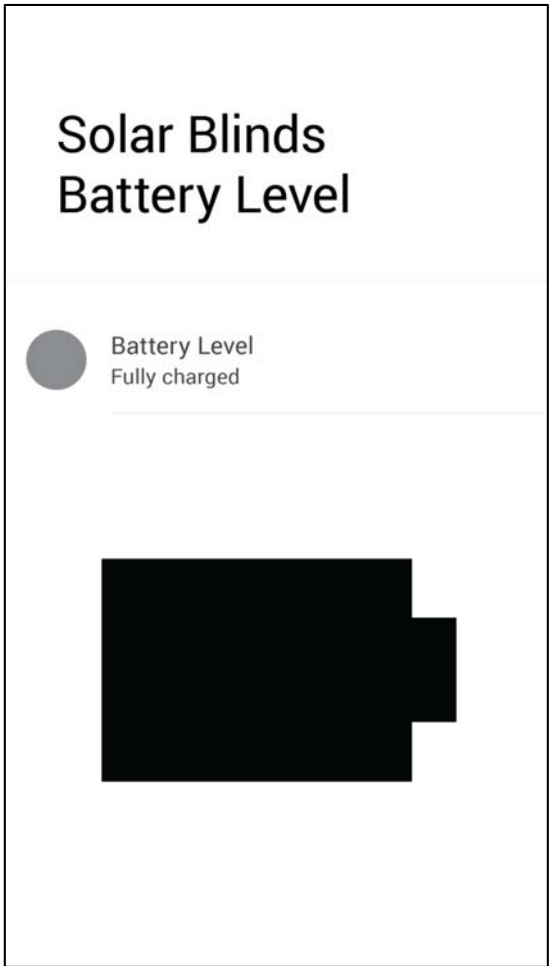

Figure 63 - Battery Menu

The system status screen in Figure 62 provides the user with navigation button to the battery level, temperature, and charging status of the mobile application. The green button represents that it is being pressed.

The battery level screen in Figure 63 shows the current battery level of the solar blinds system. The battery has 5 states that can be shown to the user. The state displayed here represents a fully charged battery.

Due to complications with the battery monitor, we ended up having to remove this feature from the project. Because of this, we felt that it was more appropriate to have the temperature readings moved to the Weather screen instead. Since we had no more use for the battery screen, this was removed from the application.

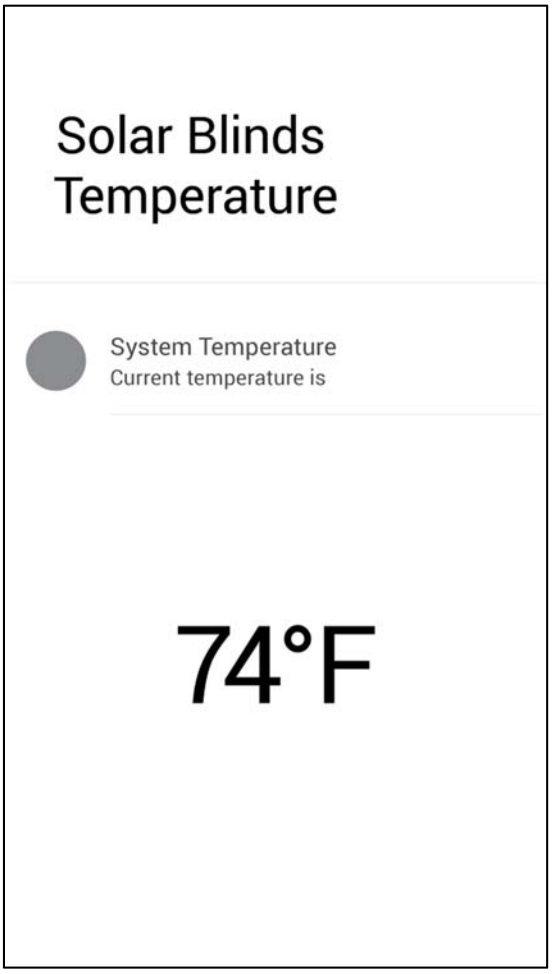

Figure 64 - Temperature Menu

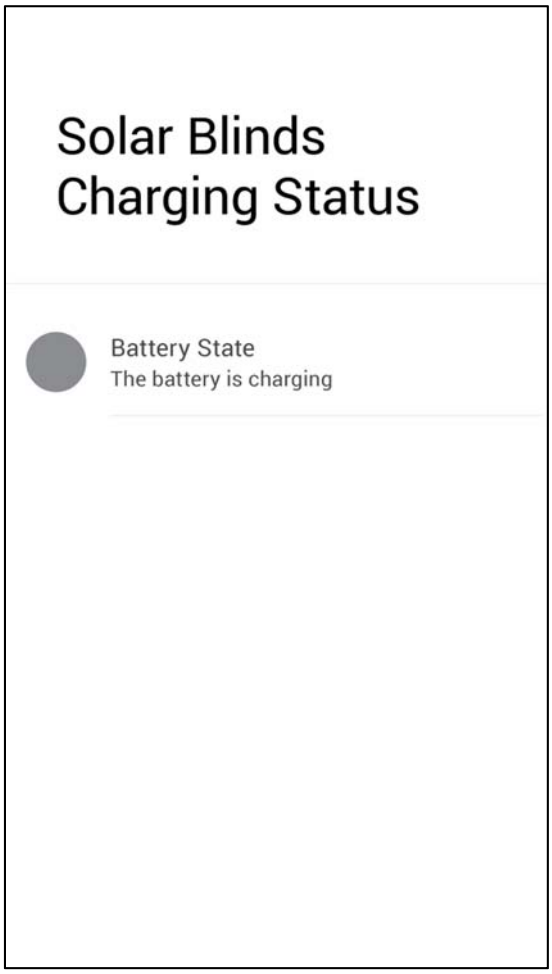

Figure 65 - Battery State Menu

Figure 64 shows the temperature screen. This screen shows the user the current outside temperature and weather. We also implemented real-time weather updates from the internet. This allows the user to see up to a 5 day forecast to decide when they should leave the blinds open or closed.

The charging status screen in Figure 65 shows the user if the blinds are currently charging or discharging energy. The battery would be discharging energy if there is less power incoming from the solar cells than is being expended to power the various functions. While a phone or other electronic device may be charged through the USB port, the battery of the blinds will display as discharging here.

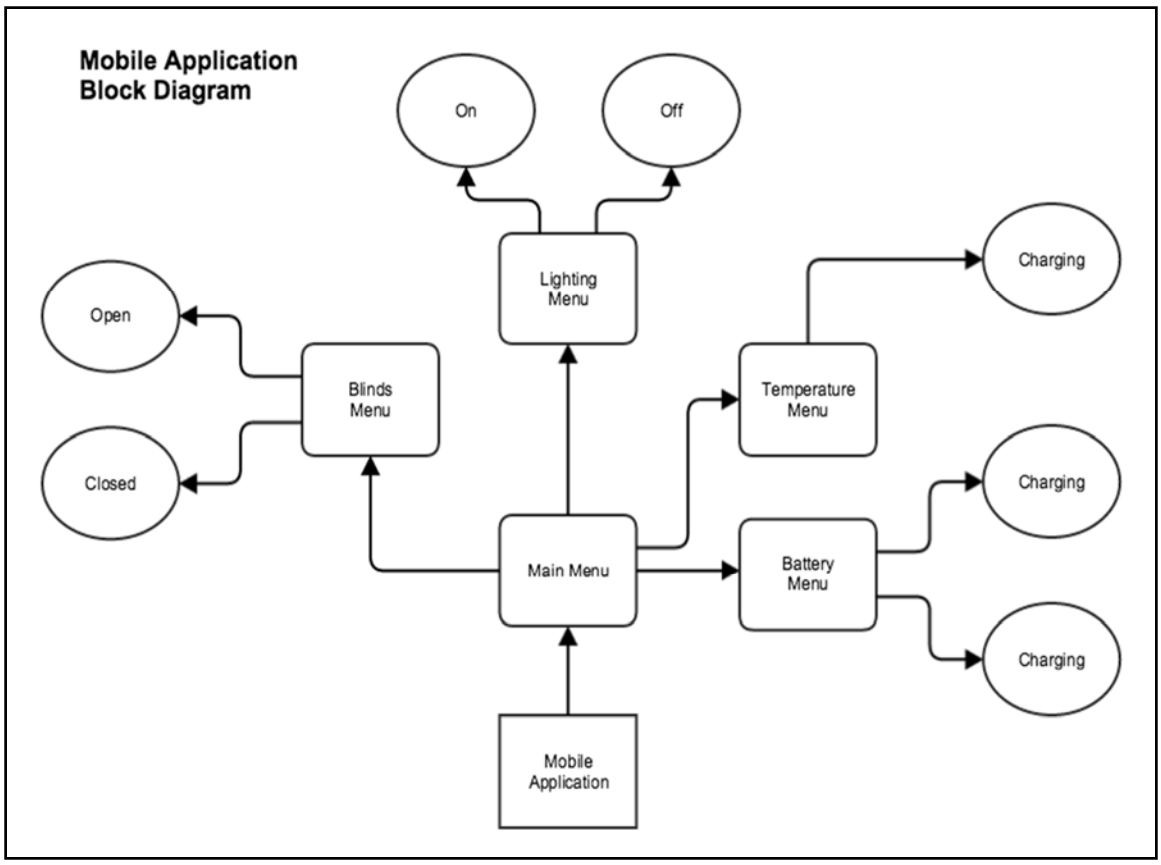

Figure 66 - Mobile Application Block Diagram

In Figure 66, we have a block diagram which shows the basic configuration of the entire mobile application. This is a visual representation of the connections between the various screens in the application and their various functions. It can be noted that we originally included a “lighting” menu because we considered adding LEDs to the room-side of the housing in order to provide light to the interior of the building while the blinds were closed. This removes the inconvenience of having the blinds closed to charge the battery with solar energy by still providing the function the sunlight normally serves.

We originally planned to use LEDs to provide the functionality that sunlight usually provides while the blinds are open. This allows you to close the blinds and still have some light on the interior of the room. However, this was decided to be a waste of power, since the main focus on the project was to collect surplus energy to charge up a battery. LEDs were a power drain that did not fit with the focus of the project. In its place, we added a Weather screen to the mobile application. This provided a 5 day weather forecast pulled from the internet to allow the user to plan accordingly for when to open or close their blinds for maximum charge performance and usability.

7.0 Project Prototype Construction and Coding

7.1 Parts Acquisition and Bill of Materials

Shown below in Table 21 is the bill of materials for this project. This consists of all of the costs and parts that were used in the production of our design.

Part	Manufacturer	Budget	Actual
Monocrystalline Solar Cells (7)	Genasun	$ 46.99	$ 46.99
Thin-Film Solar Cells	PowerFilm Solar	$ 185.99	$ 185.99
Boost Converter	RioRand	$ 10.04	$ 15.99
Battery	1000Bulbs	$ 9.93	$ 14.93
Battery Monitor	Avnet	$ 0.00	$ 5.44
MPPT Charge Controller	Genasun	$ 75.00	$ 75.00
MCU CC2640	Mouser Electronics	$ 13.08	$ 14.24
MCU Development Board	Mouser Electronics	$ 37.17	$ 156.99
USB Charger	Donated	$ 19.50	$ 0.00
PCB Assembly	Donated	$ 30.00	$ 0.00
Temperature Sensor	Maxim	$ 0.00	$ 4.89
RFID Tag Type 2 NTAG213	TagStand	$ 0.00	$ 3.99
Housing	Home Depot	$ 0.00	$ 20.00
LCD Screen	Digikey	$ 21.72	$ 12.85
Motor	RollerTrol Automation	$ 100.00	$ 118.00
Canvas	Lowe's	$ 0.00	$ 7.00
Wooden Test Frame	Lowe's	$ 0.00	$ 4.00
Debugger	Digikey	$ 0.00	$ 162.83
Components for 2 PCBs	Digikey	$ 0.00	$ 77.00
Components for 3rd PCB	Digikey	$ 0.00	$ 77.00
Replacement LCD	Digikey	$ 21.72	$ 12.85
Replacement Temp Sensor	Maxim	$ 0.00	$ 4.89
	Total:	$ 549.42	$ 1,010.87

Table 21 - Bill of Materials and Manufacturers

7.2 PCB Vendor and Assembly

We considered many different options on what to do to order and construct our PCB. We used Eagle to design our schematic and PCB layout. As far as vendors go for sending out the PCB board to be made, here are some of the vendors we were considering: 4PCB, Sunstone, Avanti Circuits, OSH Park, PCB Pool, Olimex, and PCB Train. Each of these vendors all make PCB prototype boards but some do at much higher cost and some make them at much longer lead times.

Sunstone seemed to be a bit pricey with a cost for two boards that are 0 – 9 square inches being about $150.00, but the lead time of these boards are only a week.

Avanti does not provide a price until the actual board has been submitted, but the standard lead time on these boards is about two weeks.

OSH Park seemed to be a reasonably priced option as well, even though the lead time is over two weeks. OSH Park says that the boards will ship in less than 12 days of receiving the order. This means that although they may ship within 12 days, there is still added time for the delivery. However, the cost for these boards for two layers is only $5.00 per square inch, which is much cheaper than the other options we had seen.

PCB Pool offers four 2x2 PCB boards for a total price of $80.00. This did not seem too bad if we needed multiple boards which would be nice for testing in case something went wrong and we needed more boards. The lead time for these boards was about the average of what we had been finding, as it is about two weeks.

Olimex seems like it would have been a good company to have used, but their website was down while we were trying to decide which manufacturer to use. They seem to only operate during certain months out of the year.

PCB Train seemed to be the most expensive. The cost of manufacturing PCB boards starts at 200 euros as a baseline price. In American dollars is about $221.00. This seems much more expensive than our other options and was not really worth it unless we were creating a board that was much more intricate than this.

It seemed that the best option as far as lead times and price was OSH Park. This was the company that we decided to send our PCB out to.

Figure 67 is the list of the components that we needed for the PCB Assembly. They were put into an excel spreadsheet and have the attached symbols that will appear on the actual PCB boards Silkscreen Layer.

Component	Symbol	Quantity	Value
24 Mhz Crystal	Y1	1	32 Mhz
Battery Controller	B1	1	N/A
Capacitor	C1	1	1uf
Capacitor	C2	1	1uf
Capacitor	C3	1	1uf
Capacitor	C4	1	1uf
MCP 1702	R1	1	12V to 3V
MCP 1702	R2	1	12V to 5V
USB Jack	J1	1	N/A
CC2640	M1	1	N/A
Epaper display	E1	1	N/A
Motor	N1	1	N/A

Figure 67 - PCB Components List

7.3 Final Coding Plan

Table 22 in this section outlines the calendar for the completion of different parts of the mobile application. The developers were responsible for ensuring that they met all of the following deadlines in order to provide ample amount of time for code reviews, testing, debugging, and optimization.

Since we were using agile methods for development as the developers are coding they would also be testing and verifying the mobile application both virtually and physically. In case any issues arose during development, they could be addressed and alterations could be made then and there if deemed necessary. This way the mobile application would be completed for testing by the end of November.

Week	Task	Due Date
1	Finalize design prototype	8/28
2	Complete navigation screen	9/4
3	Complete the toggle states for the buttons along with how they are to be implemented	9/11
4	Complete the android device connection setup	9/18
5	Analysis report go back and ensure completion of tasks	9/25
6	Complete the setup for opening and closing the blinds	10/2
7	Complete the setup to pull in data from the Solar Blinds	10/9
8	Analysis report go back and ensure completion of tasks	10/16
9	Ensure all tasks are in testing complete. If tasks are not in that state reassign them to developers.	10/23
10	Refractor and optimize the code	10/30

Table 22 - Final Coding Plan

7.4 Housing

The housing we decided to use had to be a small enclosure which means that our PCB will have to be smaller than usual and the amount of components we could hide in the enclosure is limited. Most importantly, we had to make sure that the mounting brackets were installed correctly and securely so that the blinds would not fall out of the wall. This could have been a major issue because we could not have any safety concerns and we did not want the system to fall and get damaged.

The housing was designed to be able to fit within a window enclosure of 36” in width. However, the other dimensions were determined based on the internal components. The compression of the rolled-up thin film solar panel was not as compact as we had hoped. This caused our housing to be larger than we initially designed it to be. In future designs of these blinds, this should be given more focus in order to decrease the size of the rolled up canvas and solar panel. With the components we had, we had to make the housing 4.5”x6.5”x36”. This still ended up not being large enough and had to extend the height by another inch to fit the boost converter and MPPT.

We had a lot of debate about what material to use in the construction of the housing. We would have preferred to use aluminum, but we had a few concerns with that. None of our team members had any skills or experience in metalworking, and we did not know any resources to utilize for outsourcing the work. Additionally, we were concerned with how we would need to insulate certain electrical components in order to avoid short circuits.

An alternative material for the housing was wood, which we ended up using. A few of our members knew how to work with wood, and had the appropriate tools, so we decided to look into that option. Wood might be heavier and unnecessarily thicker than aluminum, but the weight of the housing is not very relevant because it will not affect the power requirement of the motor, only the weight of the canvas would affect that.

Another concern with using wood is that it might warp from sun exposure. Fortunately, this was a just a prototype to be used for the demonstration of the project, and significant warping would not occur in such a short period of time. Also, there would be minimal exposure of the wood to direct sunlight since the side of the housing facing the window would be clear acrylic covering the monocrystalline solar cells.

Our final concern with using wood was the potential fire hazard. To help reduce this issue, we changed the design of the housing to move the battery out of the housing to use externally. We also considered adding a heat-resistant coating called TemperKote INS which is intended for “any surfaces which require insulation or condensation control.” [54] The TemperKote would be applied to all internal wood surfaces to help with heat and moisture control. However, since we simply removed the battery from the housing, we ended up not needing to purchase the TemperKote.

The only concern we had that we were unable to address is the possibility of condensation inside the housing. None of our team members possess the proper knowledge to determine what preventative measure to take against condensation. Our hope is that the acrylic covering coupled with the glass on the window will provide enough protection to prevent condensation on the interior of the housing.

7.5 Design Concept

Below, in Figure 68, is the concept image created to represent our project. The left image is what would be seen from the inside of the room. There are buttons for motor control, LCD, slide out tray for the phone holder, USB port, slide panel to access the battery compartment, and ventilation slits along the housing to allow for air flow through the inside of the housing. On the right side is the view of the blinds from the point of view of someone looking at it from outside, on the other side of the window. The blue squares along the top are the crystalline cells that are mounted to the housing, and the grey represents the thin-film solar panel mounted to the canvas.

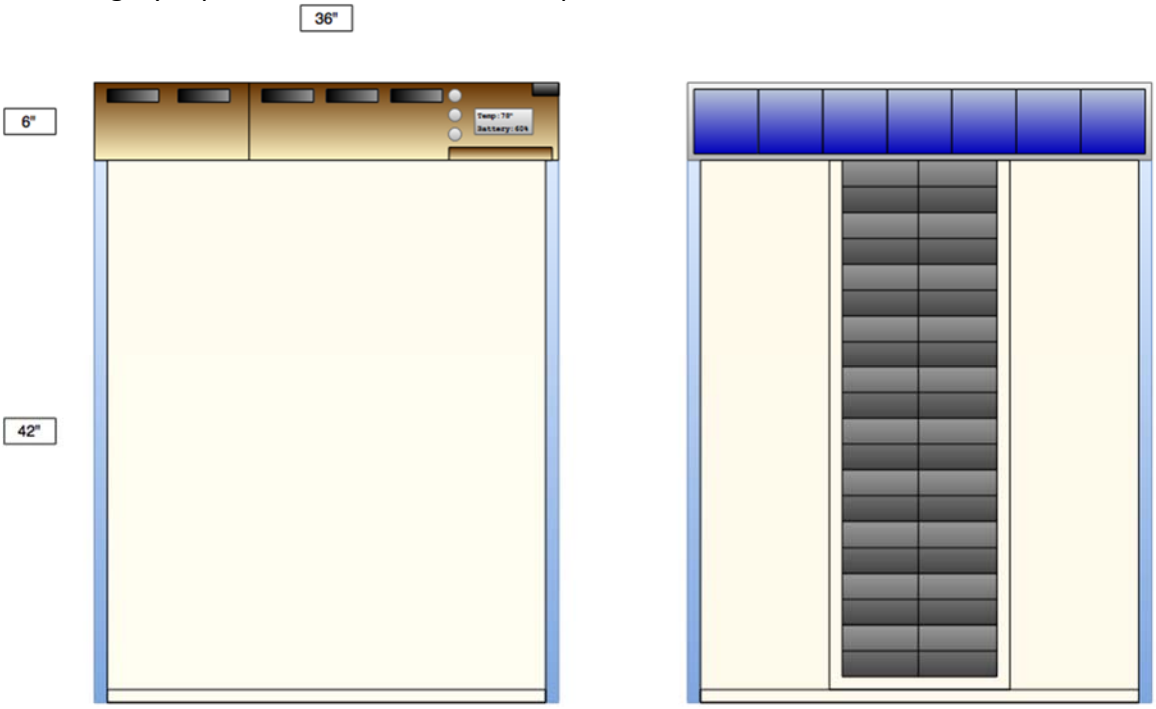

Figure 78 – Design Concept of Solar Blinds

8.0 Project Prototype Testing

Once our project was completed, there was a necessary procedure that needed to be followed to make sure all aspects were working correctly. There were many parts to this project and if any individual one failed, the whole project would not work.

8.1 Hardware Test Environment

A 3'x4' model window was built using two-by-fours. The solar shade unit was mounted to the frame. This allowed the unit to be moved around and tested as needed. The following environments were used to test the window.

- Glass pane removed
- Single glass pane tinted glass pane
- Single glass pane non-tinted glass pane
- Double glass pane tinted glass pane
- Double glass pane non-tinted glass pane

A multimeter was used to measure the amount of voltage and current that was being supplied to the battery under each test condition. A two-hour time window was used for each test condition. The quality of available light was intended be the same for each test, but that is not really within our control. The data captured in each of these tests were compared in order to find which environment is most effective for capturing light.

The final test is comparing the efficiency of the solar panel when the glass pane is on to when the glass pane is removed. The solar panel should yield higher efficiencies when the glass pane is removed because some of the incident light will be reflected when the glass pane is on.

Since the blinds where going to be inside of a window, a test was designed to determine the effects of two kinds of energy efficient glass on solar power. In this test a high intensity illuminator, a power meter, and a spectrometer where used to determine the transmittance of light through the glass. Single and double pane AGC tinted glass and Solarban 60 Low-E glass where tested. These types of glass repel heat, which makes the home more energy efficient by lowering cooling costs. Figure 79 shows a graph of the transmittance of light through the glasses. Solarban 60 Low-E single pane glass displayed the highest transmittance at 61.8%. This glass type would be best paired with the solar blinds in order to capture the greatest amount of solar energy.

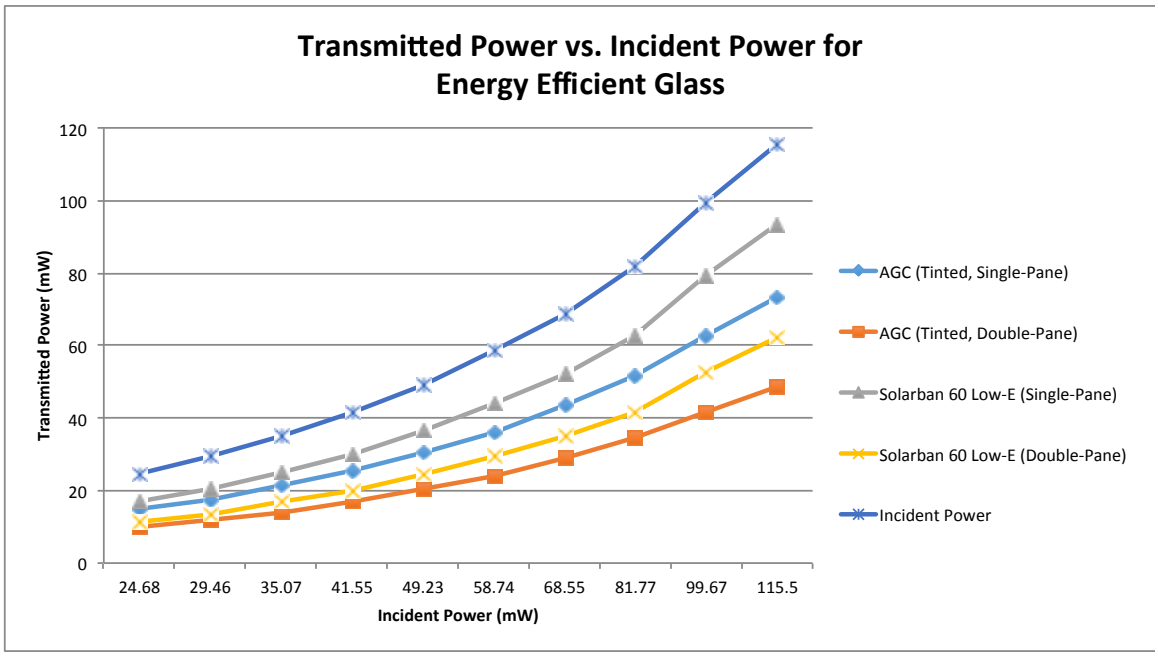

Figure 79 – Transmitted Power for Energy Efficient Glass

The transmission spectrums of the glasses clearly show why the Solarban 60 glass allows for more solar energy to pass through. Figures 80 and 81 show the transmission spectrums for the AGC tinted and Solarban 60 glass respectively. The intensity of the transmission for the Solarban 60 is much greater than that of the AGC tinted at visible light wavelengths. This means that the Solarban 60 allows for more visible light to pass through the window while still blocking the infrared radiation. This is ideal for our solar blinds application.

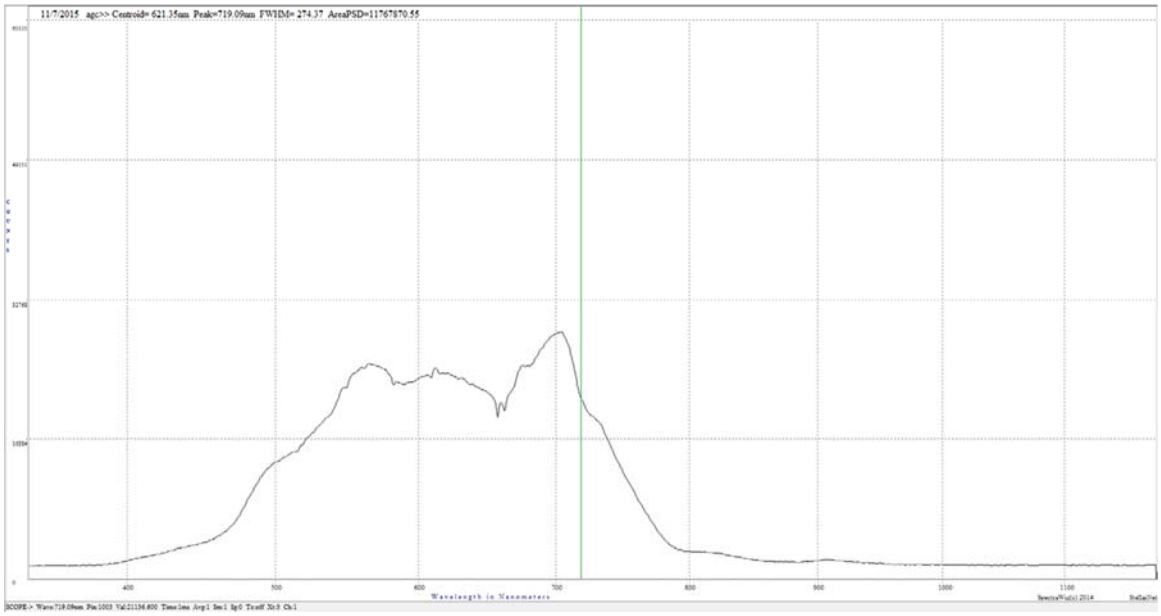

Figure 80 - AGC Tinted Glass Transmission Spectrum

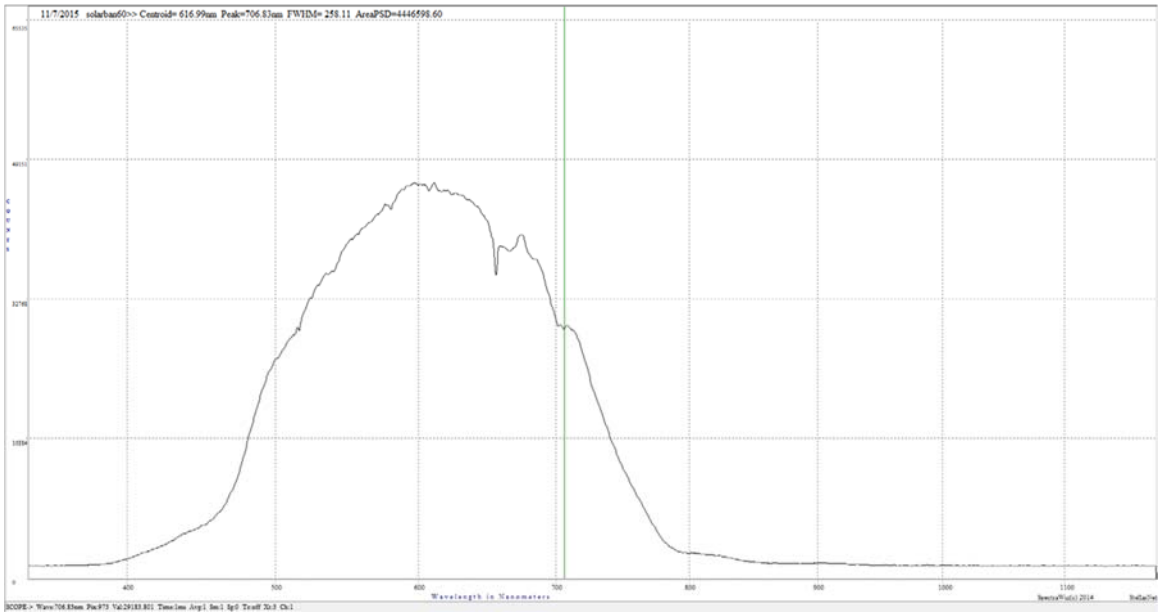

Figure 81 – Solarban 60 Low-E Glass Transmission Spectrum

8.2 Hardware Specific Testing

The Hardware test is critical to ensure the project is running at optimum performance. Our testing efforts will also ensure the project will not malfunction and cause harm to anyone that is using the product in a real life scenario. This portion of the report will be describing our projects test environment and procedures for testing the Solar Blinds hardware systems. We are going to be describing the objective for each hardware activity and what's expected to be accomplished from it. The hardware code was programmed in Code Composer Studio. The details listed below shows what we expected to occur in the software.

8.2.1 Stopping Criteria

The testing procedures must be continued non-stop unless a specific problem occurs that hinders further testing. These stopping criteria may be either functional or cosmetic. Before moving on to the next issue, the tester would consult with the other developers to make sure the issue is not a simple fix. If it is, then we would have the developers fix the issue and redeploy the code or hardware.

Once all of the test cases are passed the code and hardware is considered deliverable. This state means that there are no known bugs or issues to our knowledge and all of the system is fully working.

8.2.2 Hardware Test Cases

This section will provide a step-by-step procedure for each test case that occurred during the software testing activity. The test cases format will all be standardized into the following format.

Test Objective: Define the test case's goal.
Test Description: Define the step-by-step guidelines.
Test Conditions: Define any specific conditions that should be applied to the environment when testing.
Expected Results: Define the resulting actions that should be seen by the tester.

Table 23 - Example Table of a Test Case

Following the format of Table 23, the following Tables 24-28 describe the various test cases which we explored in order to test that our device's hardware functions properly. This testing was done at the end of all hardware development toward the end of November 2015. See the milestones in section 10.1 for more details.

Test Objective: Ensure the Solar Blinds turn on
Test Description: Using solar energy make the Solar Blinds turn on
Test Conditions: See test environment
Expected Results: The Solar Blinds should turn on.

Table 24 - Test Case: Power On

Test Objective: Make sure the Solar Blinds display screen turns on and functions
Test Description: • After the Solar Blinds turns on ensure the Solar Blinds display screen is functioning • Ensure the screen displays all of the correct details
Test Conditions: See test environment
Expected Results: The application cover screen will be displayed.

Table 25 - Test Case: Display Screen Functions

Test Objective: Verify the USB charger functions and capabilities
Test Description: • Check and make sure the USB charger is outputting the correct amount of the energy • Ensure the charger is able to charge a device
Test Conditions: See test environment
Expected Results: The USB charger is supposed to be able to power a small device via a USB cable.

Table 26 - Test Case: USB Charger Functions

Test Objective: Ensure that motors are able to fully open the blinds
Test Description: • Used the mobile application to send a signal to the Solar Blinds to open them. • Used the manual option to send a signal to the motors to open the blinds.
Test Conditions: See test environment
Expected Results: The Solar Blinds motors should be able to open the blinds

Table 27 - Test Case: Retracting the Blinds

Test Objective: Ensure that motors are able to fully close the blinds
Test Description: • Used the mobile application to send a signal to the Solar Blinds to close them. • Used the manual option to send a signal to the motors to close the blinds.
Test Conditions: See test environment
Expected Results: The Solar Blinds motors should be able to close the blinds

Table 28 - Test Case: Lowering the Blinds

8.2.3 Battery and Controller Testing

To test that the battery controller is charging the battery correctly, we needed to leave the solar panel film out in the sun for an extended period of time and have it hooked up to the battery controller which is in turn supposed to be charging the battery. We can let it continue to charge for some time, and as it is charging, we can use a multimeter and make sure that it is charging correctly and that the battery controller and the battery itself is working.

8.2.4 Voltage Regulation

After the battery and the controller had been verified to be working correctly, we needed to move to the regulation part of the circuit on the board itself. We needed to make sure that everywhere specified in the schematic, there is the correct voltage on each node there. Once this had all been verified and we know that there was enough voltage being applied to each node, then we moved on to test the motor.

8.2.5 Motor Test

First, we needed to check the motor by itself. We should see that if we just apply the voltage correctly the D/C motor should spin one way continually until it either hit the limit switch or we disconnect the voltage being applied to it. Next, we need to apply the voltage in the opposite direction, and see if the motor itself spins the opposite way. Once all this had been verified, then we know that the motor itself is working correctly so anything else that goes wrong with the motor later on in the project, we know isn't caused from the motor itself not functioning. We waited until later to check to see if the motor was correctly operated by the MCU.

8.2.6 USB Charger

The next step in the testing procedure was to test the USB charger itself. This is going to be the number one power consumer in our project. It is going to use 5V and draw about 500mA to charge compatible devices. This can be tested by checking the voltage on the USB header itself and making sure that it is five volts. Also, we can make sure that a phone or iPad or any device that is plugged in is charging. Also, we need to make sure as we are charging any device that is plugged into the USB slot that the battery is not draining too fast. We need to make sure that our battery is large enough and has enough capacity that it can supply the correct current and voltage needed to charge any of the devices plugged into that port.

8.2.7 Microprocessor Testing

This is the most complicated and extensive testing that was done in our project. The MCU is the heart of the whole project. First, we needed to make sure that the I/O pins we are working with are functioning correctly and are doing exactly what we are telling them to do. A way for us to check this was to tell the micro to put a high or a low on a certain pin, and making sure we see the pin itself go high or low using a voltmeter. This ensured that the pins that we thought we are talking to were indeed the correct pins we were talking to. This is important because we can be telling the micro to pull the wrong pin low and we can do something we didn't intend to do, or we can actually hurt the circuit itself.

After we have verified that we can correctly target the pins that we need to assert a high and a low on, then we could start testing the pins that the motor is hooked up to. We needed to make sure that we could assert a high on the right pins to turn the motor on in the direction we want it to spin to either rotate the blind open or closed. Once we have verified that we can assert highs and lows on our motor in the spots that we need to and can have it turning the blinds, we needed to check the second motor and make sure that the motor is going to act the same way for drawing the blinds open and closed. This all came before we were able to start testing the motors with the limit switch capability. Either we needed to test the limit switch so once the blinds rotate or open enough, they stop, or we needed to timeout how long it takes and assert a high or a low on that pin for however long is necessary.

8.2.8 Final Test

For the final testing after all of the individual parts have been tested, we tested the project as a whole. After we checked the charge, we began trying to drain it. We operated the blinds open and closed and see how much charge is left. Then, we charged one of our phones. After this has been accomplished, we could see how much charge once again is left in the battery. This gives us a ballpark of what to expect from a random day of sun and how much capacity it takes to run each part of our project.

8.3 Software Test Environment

Using agile methodologies, the developers were able to code the mobile application and test the functionality of the code as they progressed through the development process. Initially, most of the testing was done via Xamarin Cloud. Xamarin Cloud allowed us the ability to test the functionality of the mobile application on multiple different android operating systems easily even if with do not have the physical devices ourselves. For the testing with physical devices we have the following devices readily available:

- Samsung Galaxy Note 4
- Sony Xperia Z3V
- Access to test models from Verizon

Additionally, we used Genymotion as another platform for testing our application. Xamarin includes the capability of virtual testing on multiple platforms, but it does so through a cloud server system. This may introduce some potential complications related to typical internet latency which would skew test results. Genymotion could be installed locally so latency was not an issue.

By using both software, we were able to thoroughly test the compatibility of our application on multiple devices. This allowed us to see any critical flaws that may be related to API that has been deprecated, as well as the scalability of our GUI.

8.4 Software Specific Testing

The software test effort is a crucial part of the project's software development process. Testing our software ensured that the mobile application is functioning correctly. This portion of the report will be describing our projects test environment and procedures for testing the Solar Blinds mobile application. We are going to be describing the objective for each software activity and what was expected to be accomplished in our application. The mobile application will be specifically for the Android mobile operating system only since we developed the application solely for Android. The details listed below show what we expected to occur in the software.

8.4.1 Stopping Criteria

The testing procedures must be continued non-stop unless a specific problem occurs that hinders further testing. These stopping criteria may be either functional or cosmetic. Before moving on to the next test, the tester should consult with a developer to make sure the issue is not a simple fix. If it is, we would have the developer fix the issue and redeploy the application.

Once all of the test cases are passed, the code would be considered deliverable. This state means that there were no known bugs or issues to our knowledge and all of the application is fully working.

8.4.2 Software Test Cases

This section will provide a step-by-step procedure for each test case that will occur during the software testing activity. The test cases format will all be standardized into the following format.

Test Objective: Define the test case's goal.
Test Description: Define the step-by-step guidelines.
Test Conditions: Define any specific conditions that should be applied to the environment when testing.
Expected Results: Define the resulting actions that should be seen by the tester.

Table 29 - Example Table of a Test Case

Following the format of Table 29, the following Tables 30-56 describe the various test cases which we explored in order to test that our device functions properly. This testing will be done at the very end of all development toward the end of November 2015. See the milestones in section 10.1 for more details.

Test Objective: Click on the application icon on the android device.
Test Description: Using a virtual or physical android device click on the application icon
Test Conditions: See test environment
Expected Results: The android device should open the Solar Blinds Application.

Table 30 - Test Case: Opening the Application

Test Objective: Make sure the application cover screen appears when the application start.
Test Description: After clicking on the application icon the application cover screen should appear automatically.
Test Conditions: See test environment
Expected Results: The application cover screen will be displayed.

Table 31 - Test Case: View the Application Cover

Test Objective: View the menu screen of the Solar Blinds mobile application.
Test Description: After the cover screen the mobile application should display the Solar Blinds menu screen automatically.
Test Conditions: See test environment
Expected Results: The Solar Blinds menu screen should be displayed automatically.

Table 32 - Test Case: View the Menu Screen

Test Objective: Ensure that double pressing the back button from the view menu screen exits the application.
Test Description: Double press the device's back button to exit the application from the view menu screen
Test Conditions: See test environment
Expected Results: The mobile application should close after the double press.

Table 33 - Test Case: Exit Application

Test Objective: Confirm that the application transitions to the correct screens
Test Description: Click on the view menu screen in the following order: Case 1) • Register Device • System Blinds • System Status Case 2) • System Blinds • Register Device • System Status Case 3) • System Blinds • System Status • Register Device
Test Conditions: See test environment
Expected Results: The screens should always change to and display the correct screens.

Table 34 - Test Case: Transition Between Screens

Test Objective: Device should receive a success message from disconnecting a device.
Test Description: Once the mobile application receives the success signal from the Solar Blinds it should display “The Solar Blinds and your mobile device are now disconnected”.
Test Conditions: See test environment
Expected Results: The users device should receive a success message from disconnecting a device.

Table 35 - Test Case: Success Message from Disconnecting Device

Test Objective: The users device should receive a success message from opening the blinds.
Test Description: Once the mobile application receives the success signal from the Solar Blinds it should display “The Solar Blinds have been successfully opened. Enjoy the sunlight”.
Test Conditions: See test environment
Expected Results: The users device should receive a success message from opening the blinds.

Table 36 - Test Case: Verify the Success Message from Opening Blinds

Test Objective: Ensure clicking on the System Status button takes the user to the sub screens.

Test Description:

1. Click on the System Status button.
2. Verify the mobile application displays the sub screens
    a. Battery Level
    b. Temperature
    c. Charging Status
3. Click on each of the previously listed screens in the following order:

Case 1)

- Battery Level
- Temperature
- Charging Status

Case 2)

- Temperature
- Battery Level
- Charging Status

Case 3)

- Temperature
- Charging Status
- Battery Level

Case 4)

- Charging Status
- Temperature
- Battery Level

Case 5)

- Charging Status
- Battery Level
- Temperature

4. Verify that clicking on each button activates the button toggle.

Test Conditions: See test environment

Expected Results: The user will be able to navigate to each of the listed sub system status screen. On each button press the button should be displayed in the active state.

Table 37 - Test Case: Navigate through System Status Submenus

Test Objective: When the user presses on a button the button should toggle to the active state.
Test Description: Press on the buttons of different menus to see the buttons change state. Check the buttons on the following screens to verify the states. • Menu • Register Device • System Blinds • System Status
Test Conditions: See test environment
Expected Results: Verify the buttons change to a to a green button on press.

Table 38 - Test Case: Verify the Toggle Buttons

Test Objective: The users device should connect directly to the Solar Blinds system
Test Description: • Navigate to the Register Device screen by following this path: o Menu > Register Device • On the Register Device screen click on the Connect Device button • Verify the button toggle and the Connect Device button displays its active state • Verify that a success message is giving when the devices sync
Test Conditions: See test environment
Expected Results: The devices should connect with each other. The connect device button should display the active state button when toggled.

Table 39 - Test Case: Register Device Successfully Syncs

Test Objective: The users device should disconnect directly to the Solar Blinds system
Test Description: • Navigate to the Register Device screen by following this path: o Menu > Register Device • On the Register Device screen click on the Disconnect Device button • Verify the button toggle and the Disconnect Device button displays its active state • Verify that a success message is giving when the devices disconnects.
Test Conditions: See test environment
Expected Results: The devices should disconnect with each other. The disconnect device button should display the active state button when toggled.

Table 40 - Test Case: Register Device Successfully Disconnects

Test Objective: The users device should receive a success message from connecting a device.
Test Description: Once the mobile application receives the success signal from the Solar Blinds it should display “The Solar Blinds and your mobile device are now connected”.
Test Conditions: See test environment
Expected Results: The users device should receive a success message from connecting a device.

Table 41 - Test Case: Success Message from Connecting Device

Test Objective: The users device should receive a fail message from connecting a device if the following scenarios are present.
Test Description: • The device NFC technology is not turned on or not available. • The device Bluetooth technology is not turned or not available. • The devices are too far apart. • There is an error sending the signal
Test Conditions: See test environment
Expected Results: The users device should receive a fail message from connecting a device if the listed scenarios are present.

Table 42 - Test Case: Fail Message from Connecting Device

Test Objective: The users device should receive a fail message from disconnecting a device if the following scenarios are present.

Test Description:
• There is an error dropping the connection.
• If the device is in the following scenario it will automatically be disconnected.
  o Bluetooth technology is turned off or not available
  o The devices are two far apart

Test Conditions: See test environment
Expected Results: The users device should receive a fail message from disconnecting a device if the listed scenarios are present.

Table 43 - Test Case: Fail Message from Disconnecting Device

Test Objective: The device should receive a success message from closing the blinds.
Test Description: Once the mobile application receives the success signal from the Solar Blinds it should display “The Solar Blinds have been successfully closed. Welcome to the dark side”.
Test Conditions: See test environment
Expected Results: The device should receive a success message from closing the blinds.

Table 44 - Test Case: Verify the Success Message from Closing Blinds

Test Objective: The users device should receive a fail message from opening the blinds if the following scenarios are present.
Test Description: • The device NFC technology is not turned on or not available. • The device Bluetooth technology is not turned or not available. • The devices are too far apart. • There is an error sending the signal
Test Conditions: See test environment
Expected Results: The users device should receive a fail message from opening the blinds if the listed scenarios are present.

Table 45 - Test Case: Verify the Fail Message from Opening Blinds

Test Objective: The users device should receive a fail message from closing the blinds if the following scenarios are present.
Test Description: • The device NFC technology is not turned on or not available. • The device Bluetooth technology is not turned or not available. • The devices are too far apart. • There is an error sending the signal
Test Conditions: See test environment
Expected Results: The users device should receive a fail message from closing the blinds if the listed scenarios are present.

Table 46 - Test Case: Verify the Fail Message from Closing Blinds

Test Objective: Use the mobile application to open the Solar blinds
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Blinds • On the System Blinds screen click on the Open Blinds button • Verify the button toggle and the Open Blinds button displays its active state • Verify that a success message is giving when the Solar Blinds recognizes the signal and they begin to open.
Test Conditions: See test environment
Expected Results: The mobile application and the Solar Blinds should connect and a signal is transmitted between them. The mobile application will display a success message when the signal is confirmed and the Solar Blinds will begin opening. The Open Blinds button should display the active state button when toggled.

Table 47 - Test Case: Opening the Solar Blinds

Test Objective: Use the mobile application to close the Solar Blinds
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Blinds • On the System Blinds screen click on the Close Blinds button • Verify the button toggle and the Close Blinds button displays its active state • Verify that a success message is giving when the Solar Blinds recognizes the signal and they begin to close.
Test Conditions: See test environment
Expected Results: The mobile application and the Solar Blinds should connect and a signal is transmitted between them. The mobile application will display a success message when the signal is confirmed and the Solar Blinds will begin opening. The Open Blinds button should display the active state button when toggled.

Table 48 - Test Case: Closing the Solar Blinds

Test Objective: Use the mobile application to view the Solar Blinds system battery levels

Test Description:
• Navigate to the System Blinds screen by following this path:
  o Menu > System Status > Battery Level
• Verify the correct battery levels are being displayed. The options in battery levels are:
  o Full
  o Almost Full
  o Half Full
  o Almost Empty
  o Empty
• Verify the words match the image of the battery being displayed respectively.

Test Conditions: See test environment
Expected Results: The mobile application will display the correct battery levels in reference to the systems current battery levels

Table 49 - Test Case: View the System's Battery Level

Test Objective: Use the mobile application to view the Solar Blinds system full battery level.
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Battery Level • Verify the correct battery levels are being displayed when the battery level is reading in between 100% - 80%. • Verify the words match the image of the battery being displayed respectively.
Test Conditions: See test environment
Expected Results: The mobile application will display the correct battery levels in reference to the systems current battery levels

Table 50 - Test Case: View the Full Battery Level

Test Objective: Use the mobile application to view the Solar Blinds system almost full battery level.
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Battery Level • Verify the correct battery levels are being displayed when the battery level is reading in between 79% - 60%. • Verify the words match the image of the battery being displayed respectively.
Test Conditions: See test environment
Expected Results: The mobile application will display the correct battery levels in reference to the systems current battery levels

Table 51 - Test Case: View the Almost Full Battery Level

Test Objective: To view the Solar Blinds system half full battery level.
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Battery Level • Verify the correct battery levels are being displayed when the battery level is reading in between 59% - 40%. • Verify the words match the image of the battery being displayed respectively.
Test Conditions: See test environment
Expected Results: The mobile application will display the correct battery levels in reference to the systems current battery levels

Table 52 - Test Case: View the Half Full Battery Level

Test Objective: Use the mobile application to view the Solar Blinds system almost empty battery level.
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Battery Level • Verify the correct battery levels are being displayed when the battery level is reading in between 39% - 20%. • Verify the words match the image of the battery being displayed respectively.
Test Conditions: See test environment
Expected Results: The mobile application will display the correct battery levels in reference to the systems current battery levels

Table 53 - Test Case: View the Almost Empty Battery Level

Test Objective: Use the mobile application to view the Solar Blinds system empty battery level.
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Battery Level • Verify the correct battery levels are being displayed when the battery level is reading in between 19% - 0%. • Verify the words match the image of the battery being displayed respectively.
Test Conditions: See test environment
Expected Results: The mobile application will display the correct battery levels in reference to the systems current battery levels

Table 54 - Test Case: View the Empty Battery Level

Test Objective: Use the mobile application to view the Solar Blinds system temperature reading levels
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Temperature • Verify the correct readings are being displayed. • The mobile application should display only reading in Fahrenheit degrees. • Verify the display is standardized to look like the following: o **74**°F
Test Conditions: See test environment
Expected Results: The mobile application should display the Solar blinds temperature reading in Fahrenheit degrees.

Table 55 - Test Case: View the System's Temperature Reading

Test Objective: Use the mobile application to view the Charging Status of the Solar blinds system.
Test Description: • Navigate to the System Blinds screen by following this path: o Menu > System Status > Charging Status • Verify the correct charging statuses are being displayed. • When the system is charging the charging status should say: o The battery is charging • When the system is discharging the charging status should say: o The battery is discharging
Test Conditions: See test environment
Expected Results: The mobile application should display the Solar blinds charging status correctly for the charging state and the discharging state.

Table 56 - Test Case: View the System's Charging Status

9.0 Project Operation

9.1 Hardware Operation

The way this project works is quite simple. The user takes their phone and has it sync to the blinds if they have a NFC chip in their phone and it will be linked to the blinds. This can then be connected through the Bluetooth chip and be controlled through an app on the phone. Once the phone has connected to the Bluetooth chip, you will be able to open and close the blinds with a simple push of a button. This will either open or retract the blinds based on the position of the blinds at the point of activation when the button is pushed. When the button is pushed, it will send a signal to the Bluetooth chip and

this will send a signal to the microprocessor. This then will be translated to the microprocessor that a button was pushed. Depending on which button was pushed on the app, this will send a signal to the micro that will either tell the motor to spin clockwise or counterclockwise. The micro will put a high on the right side of the motor that will tell t to open or close the blinds. The motor will then run and complete its task.

Also, there are other functions on these blinds. The blinds also come equipped with its own USB charger to charge any non-Apple device. It pulls approximately 500mA and has a voltage of 5V across it. This is more than sufficient to charge any device, even if it takes a bit longer than desired. There is also an LCD to display various information about the blinds.

The temperature gauge will read the current indoor temperature and it will send the results to the microcontroller. We then update the display to show the current temperature. This information is also transmitted over Bluetooth to the mobile application. We have the sensors take a temperature reading every few minutes as to use the least amount of power possible so we can keep the battery usage as low as possible.

9.2 Software Operation

The operation of the mobile application is simple and fluent enough for the user to not require a tutorial and allow easy understanding of what each control accomplishes and how to access them. The mobile application is not available for download from the Play Store due to the cost of licensing.

Once the icon has been clicked, the user is be presented with the view menu screen and here they are be presented with the view menu screen and here they are able to navigate to all of the following sub screens. The buttons listed each have its own toggle states to inform the user the mobile application recognizes the users touch input. The active states for all of the buttons will be that the normal button color changes. The available buttons on the menu screen are register device, system blinds, and weather.

In order the truly operate and interact with the mobile application and the Solar Blinds, the user must connect their mobile device to the Solar Blinds. To do this the user must navigate to the register device screen by pressing on the respective button. Once on the screen the user can choose to connect their device or disconnect it. Pressing connect device will initiate the mobile devices NFC and Bluetooth technology to establish a direct connection with the two devices will communicate via Bluetooth. Once the devices are successfully connected the mobile application will display an informational alert saying that the procedure was a success. Pressing on the disconnect device button will initiate the dropping of the active connections between the mobile device and the Solar Blinds thus disconnecting the two.

The next possible software operation the user can interact with the mobile application is to control the blinds motor via the application. The user must click on the Motor button on the menu screen to be taken to the respective sub screen. On the Motor screen the user will be able to select either open blinds or close blinds. The mobile application does not allow the user the option to send multiple open signals or multiple close signals when the Solar Blinds are already in the respective states. This prevents the mobile application from potentially causing damage to the motors and housing of the Solar Blinds. Once one of the buttons is pressed the application displays the active states for the button and begins to send the signal via Bluetooth to the Solar Blinds. This signal obtains the current state of the blinds and verifies that the state is not in the state the mobile application is trying to make it enter. If the states do not make the blinds motors will activate and either open or close the blinds. After finishing the Solar Blinds will send a response signal stating the blinds are fully open or closed. Then the mobile application will display a success alert saying that the procedure was a success. If the states do match then the Solar Blinds will do nothing and send a fail signal back to the mobile application. When the mobile application receives this signal it will alert the user that the blinds may be already in the suggested state or there was an error sending the signal.

The system status screen mainly provides data details of the Solar Blinds instead of providing direct user interaction. The system status has the following options the user can select to view: battery level, temperature, and charging status. Clicking on the battery level option transitions the screen to display the current Solar Blinds battery level from being charged by the solar energy. There are 5 different states of the battery level. The first state is full. This is where the battery level is completely charged. The mobile application displays this state for the battery if it's in between the range 100% - 80% and it will show a full battery icon. The next state is the almost full state. This state will be displayed if the battery is in between 79% - 60% and it will show a semi full (75%) battery icon. The next state after that is the half full state. This state will be displayed if the battery is in between 59% -40% and it will show a 50% full battery icon. The next state is the almost empty state. This state will be displayed if the battery is in between 39% - 20% and it will show an almost empty (25%) battery icon. The last state is the empty state. This state will be displayed if the battery is in between 19% - 0% and it will show an empty (0%) battery icon. When the battery icon is in the empty state some of the functionality of the Solar Blinds may be operational due to the low power reserves.

Viewing the temperature is also another part of the mobile application were the user has very minimal direct action on the Solar Blinds. Once the user transitions to this screen, the mobile application displays the temperature from the sensor in Fahrenheit and Celcius, along with the weather forecast.

10.0 Administrative Content

10.1 Milestones

Our group set a list of milestones for us to achieve in order to stay on track in our project development. By completing the sections described by the dates listed, we would be able to ensure completion and thoroughness of our project by the dates due.

These milestones were divided by month for legibility. The milestones for documentation are set for June 2015 to August 2015 in Tables 57 through 59.

June 2015	
Section	**Complete By**
Executive Summary	06/14/2015
Project Motivation and Goals	06/14/2015
Objectives	06/14/2015
Requirements Specifications	06/14/2015
Existing Similar Projects and Products	06/21/2015
Relevant Technologies: Solar Cells	06/21/2015
Relevant Technologies: Display Screens	06/21/2015
Relevant Technologies: Motors	06/21/2015
Relevant Technologies: Remote Connectivity	06/21/2015
Relevant Technologies: Microchip Processors	06/30/2015
Relevant Technologies: Motors	06/30/2015
Relevant Technologies: Power Distribution	06/30/2015
Relevant Technologies: Embedded	06/30/2015
Relevant Technologies: Mobile App	06/30/2015
Standards: Safety	06/30/2015
Standards: Reliability	06/30/2015
Standards: Communications	06/30/2015
Standards: Programming Languages	06/30/2015
Standards: Connectors	06/30/2015
Standards: Battery	06/30/2015
Standards: Design Impacts	06/30/2015

Table 57 - June 2015 Milestones

July 2015	
Section	**Complete By**
Design Constraints: Economic & Time	07/07/2015
Design Constraints: Ethical, Health, Safety	07/07/2015
Design Details: Electrical Hardware	07/21/2015
Design Details: Mechanical Hardware	07/21/2015
Design Details: Solar Technologies	07/21/2015
Design Details: Embedded Software	07/21/2015
Design Details: Mobile Application	07/21/2015
Prototype: Hardware Test Environment	07/31/2015
Prototype: Hardware Specific Testing	07/31/2015
Prototype: Software Test Environment	07/31/2015
Prototype: Software Specific Testing	07/31/2015
Milestones	07/31/2015
Budget & Financing	07/31/2015

Table 58 - July 2015 Milestones

August 2015	
Section	**Complete By**
Competency & Completeness Review	08/01/2015
Finalize Citations	08/02/2015
Finalize Formatting	08/03/2015
Documentation Binding	08/05/2015

Table 59 - August 2015 Milestones

Here, at the beginning of August, we finished our work on the documentation and began the work on actually creating our device. The lists of milestones for development are displayed in Tables 60-62, and only include deadlines by which we must accomplish various goals. However, it is implied that we will also begin working on the next section at the due date of the previous section. For example, below we can see that the battery status display screen of the mobile application will be completed by September 18th, and starting around that same time will be the development for the data communication with the MCU.

Since we have four members in our group, we will have a lot of concurrent work going on in order to complete the project within the time allotted to us. Because of this, a lot of the due dates are all intermingled among the different facets of our group. We have Sean and Stephen working on the "hardware" sections, and Artis and Dakota evenly divide the "embedded" and "mobile app" sections, and then whoever is free at the time

will be responsible for the "general" categories. The different sections have been color-coded for easy reference.

September 2015		
Category	**Section**	**Complete By**
Mobile App	Basic menu operation functional	09/01/2015
General	All hardware acquired	09/01/2015
Mobile App	RFID sync functional	09/04/2015
Hardware	Motor installed into blinds	09/04/2015
Hardware	Solar cells mounted	09/04/2015
Mobile App	Menus optimized	09/11/2015
Hardware	PCB Finished	09/11/2015
Hardware	Temperature gauge installed	09/11/2015
Hardware	Buttons mounted	09/11/2015
Mobile App	Battery status display complete	09/18/2015
Embedded	Battery charging operational	09/18/2015
Embedded	Manual operation functional (buttons)	09/18/2015
Hardware	Charging systems complete	09/18/2015
Mobile App	Send/receive data to/from MCU operational	09/25/2015
Embedded	Bluetooth connection to phone functional	09/25/2015
Hardware	Components wired (temp, buttons, e-paper)	09/25/2015
Hardware	E-paper display mounted and wired	09/25/2015

Table 60 - September 2015 Milestones

After the end of September, we had enough of the hardware completed that we were able to start focusing more on the software development. We needed the PCB to be assembled before we can do much with the embedded programming, and we could only do so much with the mobile application without having a Bluetooth link with the MCU.

October 2015		
Category	**Section**	**Complete By**
Mobile App	Motor functions controllable	10/02/2015
Embedded	E-paper display operational	10/02/2015
Hardware	USB charge port installed, wired, and powered	10/02/2015
Mobile App	Battery screen complete	10/09/2015
Embedded	E-paper displaying battery and temperature info	10/09/2015
Hardware	Phone holder mounted to housing	10/09/2015
Mobile App	Temperature screen complete	10/16/2015
Embedded	USB charge port operational	10/16/2015
Hardware	Wooden testing frame assembled	10/16/2015
Mobile App	Support for multiple blinds RFID registration	10/23/2015
Embedded	Reflect USB connection status on e-paper	10/23/2015
Hardware	Blinds able to mount into test frame	10/23/2015

Table 61 - October 2015 Milestones

November 2015		
Category	**Section**	**Complete By**
Mobile App	Battery statistics added based on past history info	11/06/2015
Hardware	Battery charge efficiency tests	11/06/2015
Mobile App	Import weather info from internet to temperature screen	11/13/2015
Mobile App	UI compatibility across devices achieved	11/13/2015
Hardware	Phone charge efficiency via USB tests	11/13/2015
Mobile App	Optimizations	11/20/2015
General	Demonstration/presentation material compiled	11/20/2015
Mobile App	Final software systems test	11/27/2015
General	Presentation material completed	11/30/2015

Table 62 - November 2015 Milestones

Unfortunately, we must admit that not all milestones for the research and documentation portion were completed by the set deadlines. However, this is true for most projects so we were not discouraged. We still had all milestones for August complete by the final deadline of the report submission. We had set additional

milestones to help ourselves stay on track of the actual development of the product with the goal of keeping a tighter schedule than we did for the research and writing of this report.

We had decided that it would be best to have a finalized product, both in hardware and software, by the end of November. In actuality, our development was delayed significantly due to issues in setting up debugging hardware that we were unaware was necessary for programming the embedded software. Once we had everything set up, development was done on a pretty tight time constraint. In the end, we managed to get everything working by our final deadline of November 30th. Unfortunately, just before our project demonstration, some of the circuitry burnt out and we required a new PCB.

10.2 Budget and Finance Discussion

We decided as a group that our total budget for this project would be $500, but with a goal of $300 or less. We had no sponsors so we paid out of pocket for this project. However, we also believe that anything worth doing should be done right, so we did not unnecessarily sacrifice quality for no other reason than expense.

After the final bill of materials was realized, it appears that we did not meet our $300 goal, at least in planning. Fortunately, by the end of the planning phase we were able to keep the cost under the $500 limit which we had set originally. Then, after we got into the development of the project, we realized there were additional materials and components we needed which added to the cost significantly. In the end, we were pretty far over budget.

Even though our costs were higher than estimated, some members are keeping certain parts such as the solar panels and MCU development board, so they paid for the full cost of those parts. This helped to mitigate cost of the project. The remaining cost was divided among all members of the team.

Appendices

Appendix A: Copyright Permissions

Texas Instruments

Texas Instruments is pleased to provide the information on these pages of the World Wide Web. We encourage you to read and use this information in developing new products.

TI grants permission to download, print copies, store downloaded files on a computer and reference this information in your documents only for your personal and non-commercial use. But remember, TI retains its copyright in all of this information. This means that you may not further display, reproduce, or distribute this information without permission from Texas Instruments. This also means you may not, without our permission, "mirror" this information on your own server, or modify or re-use this information on another system.

TI further grants permission to non-profit, educational institutions (specifically K-12, universities and community colleges) to download, reproduce, display and distribute the information on these pages solely for use in the classroom. This permission is conditioned on not modifying the information, retaining all copyright notices and including on all reproduced information the following credit line: "Courtesy of Texas Instruments". (http://www.ti.com/corp/docs/legal/copyright.shtml)

Atmel

Materials from this website www.atmel.com and any other website owned, operated or controlled by Atmel and/or its affiliated or subsidiary companies (together, Atmel) are owned and copyrighted by Atmel. Unauthorized use of such Materials (e.g., information, documentation and software), including these Terms, may be a violation of Atmel's intellectual property rights or other applicable laws. If you agree to these Terms, you may download (on a single computer), copy or print a single copy of all or a portion of the Materials for informational, non- commercial, lawful purposes only. You may distribute free copies of the documentation available at this website only to customers and prospective customers of Atmel's products. Any other distribution to third parties is strictly prohibited unless you obtain the prior written consent of Atmel. You may not modify in any way any of the Materials contained herein, or delete or modify any of Atmel's copyright, trademark or other proprietary notices.
(http://www.atmel.com/About/legal.aspx)

http://genasun.com/all-products/solar-charge-controllers/for-lead/gv-5-pb-5a-solar-charge-controller/
http://www.ti.com/lit/ds/symlink/cc2640.pdf
https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcR2C40O_XyoI_K9CtqM0xEoa0mp2ZMvp5lXEVkblk-zeSmlUsHo1D15KE8
http://www.ctscorp.com/components/Datasheets/008-0309-0.pdf
http://www.digikey.com/product-detail/en/S1144CS021/S1144CS021-ND/5046794
http://ww1.microchip.com/downloads/en/DeviceDoc/22008E.pdf

ML Solar

From: <sales@mlsolar.com>
Subject: RE: 'diamonds@knights.ucf.edu' submitted the form from your 'Contact Us' page
Date: August 3, 2015 at 12:09:57 PM EDT
To: <diamonds@knights.ucf.edu>

Hi Sean.
Yes. Feel free to use our images for your school design project. If you have any other questions please let us know.

ML Solar
Danny
408 583 8101

> -------- Original Message --------
> Subject: 'diamonds@knights.ucf.edu' submitted the form from your 'Contact Us' page
> From: "Contact Form" <contactform@bigcommerce.com>
> Date: Sun, August 02, 2015 3:03 pm
> To: sales@mlsolar.com
>
> **Full Name:** Sean Diamond
> Hello,
> I am a senior engineering student at the University of Central Florida and I am working on my senior design project. I would like to use a picture of the 3x6 solar cells from your website in my paper and I wanted to ask to make sure it was okay. This paper is for a design class and will not be published or used to make money.
>
> Thank you,
> Sean Diamond

Morningstar

Sean,

Thank you for contacting Morningstar with your inquiry. You can use any pictures from our website, but we would appreciate it if you noted that the pictures are courtesy of Morningstar in your work.

Please let me know if you have any other questions.

Thanks,

Patrick Smith
Technical Sales Engineer, Morningstar Corporation

P: 215-321-4457
E: psmith@morningstarcorp.com

Flex Solar Cells

Sean,

Sure. Just send me a copy of the final report. I'd love to read it.

Best Regards,
Fritz Meitzen
Sales Director
Electrical Engineer

FlexSolarCells.com
P: (512) 680-7034
fritzm@flexsolarcells.com

On Aug 2, 2015, at 5:30 PM, Sean Diamond <diamonds@knights.ucf.edu> wrote:
Hello,
I am a senior engineering student at the University of Central Florida and I am working on my senior design project. I would like to use a picture of the R-14 model from your website in my paper and I wanted to ask to make sure it was okay. This paper is for a design class and will not be published or used to make money.

Thank you,
Sean Diamond

Battery Space

Hi Sean,
Yes, it is ok, please indicate the picture is from BatterySpace.com
Best regards,
Jasmine Sun
Batteryspace.com/AA Portable Power

825 S.19th St.
Richmond, CA 94804
Tel: +1-510-525-2328
Fax: +1-510-439-2808
Email: sales@batteryspace.com
Site: http://www.batteryspace.com

We can now provide UN38.3 / IEC 62133 / UL 2054 / CE Test service.
By accepting our order, you agreed to our Sales Agreement

On Sun, Aug 2, 2015 at 3:56 PM, Sean Diamond <diamonds@knights.ucf.edu> wrote:
Hello,

I am a senior engineering student at the University of Central Florida and I am working on my senior design project. I would like to use a picture of the Powerizer 12 V 5 Ah LiFePO4 battery from your website in my paper and I wanted to ask to make sure it was okay. This paper is for a design class and will not be published or used to make money.

Thank you,
Sean Diamond

Pervasive Displays Inc.

---------- Forwarded message ----------
From: "Soren Jorgensen" <soren_jorgensen@pervasivedisplays.com>
Date: Aug 3, 2015 8:55 PM
Subject: Greetings from Pervasive Displays
To: <artiscolemanjr@gmail.com>
Cc:

Hello Artis,

Thank you for reaching out to us. Which university do you go to?

Please consider this email as permission to use material we have posted on our website or on repaper.org as long as the source is credited.

I would personally be interested in reading the report and hope to receive a copy.

Best regards,
Soren Jorgensen

| Soren Jorgensen | Pervasive Displays | Business Development |Portland, OR |
| +1917.553.6304 | Soren_Jorgensen@PervasiveDisplays.com| sjn_chidisp (Skype) |

Appendix B: Abbreviations

Below is a list of abbreviations used in this document and their full meanings.

Abbreviation	Meaning
DIY	Do It Yourself
PCB	Printed Circuit Board
USB	Universal Serial Bus
Wi-Fi	Wireless Internet for Frequent Interface
LED	Light-Emitting Diode
MCU	Microcontroller Unit
BOM	Bill of Materials
BLE	Bluetooth Low Energy
RFID	Radio Frequency Identification
NFC	Near Field Communications, a subset of RFID
GUI	Graphic User Interface
LCD	Liquid Crystal Display
MPPT	Maximum Power Point Tracker
LDR	Light-Dependent Resistor
DPDT	Double Pole, Double Throw switch
ADC	Analog-to-Digital Converter
AC	Alternate Current
DC	Direct Current
IC	Integrated Circuit
JST	Japan Solderless Terminal
CRT	Cathode Ray Tube
I/O	Input/Output
PWM	Pulse Width Modulator
SLA	Sealed Lead Acid
AGM	Absorbed Glass Mat
IDE	Integrated Development Environment

Appendix C: References

[1] Instructables.com, 'Automatic Window Blinds Controller (PICAXE)', 2015. [Online]. Available: http://www.instructables.com/id/Build-A-Motorized-Window-Blinds-Controller-For-Les/. [Accessed: 05- Aug- 2015].

[2] Instructables.com, 'How to make a solar iPod/iPhone charger -aka MightyMintyBoost', 2015. [Online]. Available: http://www.instructables.com/id/How-to-make-a-solar-iPodiPhone-charger-aka-Might/. [Accessed: 05- Aug- 2015].

[3] S. Kasap, *Optoelectronics and photonics*. Boston: Pearson, 2013.

[4] T. Saga, 'Advances in crystalline silicon solar cell technology for industrial mass production', *NPG Asia Materials*, vol. 2, no. 3, pp. 96-102, 2010.

[5] Pveducation.org, 'Multi Crystalline Silicon | PVEducation', 2015. [Online]. Available: http://www.pveducation.org/pvcdrom/manufacturing/multi-crystalline-silicon. [Accessed: 05- Aug- 2015].

[6] Solar-facts-and-advice.com, 'Amorphous silicon solar cells: Solar Facts and Advice', 2015. [Online]. Available: http://www.solar-facts-and-advice.com/amorphous-silicon.html. [Accessed: 05- Aug- 2015].

[7] 'Staebler-Wronski Effect in Amorphous Silicon and Its Alloys', *OPTO-ELECTRONICS REVIEW*, pp. 21-34, 2004.

[8] Energy.gov, 'Cadmium Telluride | Department of Energy', 2015. [Online]. Available: http://energy.gov/eere/sunshot/cadmium-telluride. [Accessed: 05- Aug- 2015].

[9] Eink.com, 'E Ink: Technology: Electrophoretic Techology', 2015. [Online]. Available: http://www.eink.com/technology.html. [Accessed: 05- Aug- 2015].

[10] Pervasivedisplays.com, 'Pervasive Displays - Why ePaper (EPD)', 2015. [Online]. Available: http://www.pervasivedisplays.com/technology/home. [Accessed: 05- Aug- 2015].

[11] P. Displays, 'E1271CS021 Pervasive Displays | E1271CS021-ND | DigiKey', *Digikey.com*, 2015. [Online]. Available: http://www.digikey.com/product-detail/en/E1270CS021/E1270CS021-ND/5046793. [Accessed: 05- Aug- 2015].

[12] Desktop Class - Online Classroom, 'Discuss Different Types Of Display Screens', 2011. [Online]. Available: http://www.desktopclass.com/education/computer-it/discuss-different-types-of-display-screens.html. [Accessed: 05- Aug- 2015].

[13] PCMAG, 'How to Buy an LCD Monitor', 2015. [Online]. Available: http://www.pcmag.com/article2/0,2817,2362048,00.asp. [Accessed: 05- Aug- 2015].

[14] HubPages, 'Screen Displays and Types', 2015. [Online]. Available: http://sdrahcir8.hubpages.com/hub/Screen-Displays-and-Types. [Accessed: 05- Aug- 2015].

[15] Applied-motion.com, 'V0400-211-C-000 | Applied Motion', 2015. [Online]. Available: http://www.applied-motion.com/products/servo-motors/v0400-211-c-000?gclid=CjwKEAjwiZitBRCy0pb3rIbG9XwSJACmuvvzQzOH6cBfJrGSHhzrXfUBWESLlE1E2sY2EzAUOLeb_RoCUMjw_wcB. [Accessed: 05- Aug- 2015].

[16] S. Leeson Motors DC Motor-.05 - .1HP, 'Leeson Motors DC Motor-.05 - .1HP, 12-24V, 1750-4200RPM, TENV, Sq. flange', *Global Industrial*, 2015. [Online]. Available:

http://www.globalindustrial.com/p/motors/ac-motors-2-phase/dc-motors/motor-25-tenv-1750-4200rpm-536f?infoParam.campaignId=T9F&gclid=CjwKEAjwiZitBRCy0pb3rIbG9XwSJACmuvvzwfAkcTsO3eZcoxMyUIgHq3r4EB38stGEwMGzNFVCghoC7Mvw_wcB. [Accessed: 05- Aug- 2015].

[17] Rollertrol.com, '12v DC Tubular Motors with Built-in Remote Control', 2015. [Online]. Available: http://rollertrol.com/store/en/tubular-motors/33-window-blind-motor.html. [Accessed: 05- Aug- 2015].

[18] Diffen.com, 'Bluetooth vs Wi-Fi - Difference and Comparison | Diffen', 2015. [Online]. Available: http://www.diffen.com/difference/Bluetooth_vs_Wifi. [Accessed: 05- Aug- 2015].

[19] Msdn.microsoft.com, 'Wi-Fi power management for connected standby platforms (Windows Drivers)', 2015. [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/hardware/dn757332(v=vs.85).aspx. [Accessed: 05- Aug- 2015].

[20] S. Higginbotham, 'ZigBee wants to be the Bluetooth of the internet of things. Too bad everyone hates it.', *Gigaom.com*, 2013. [Online]. Available: https://gigaom.com/2013/08/30/zigbee-wants-to-be-the-bluetooth-of-the-internet-of-things-too-bad-everyone-hates-it/. [Accessed: 05- Aug- 2015].

[21] J. Thrasher, 'RFID versus NFC: What's the difference between NFC and RFID?', *RFID insider*, 2013. [Online]. Available: http://blog.atlasrfidstore.com/rfid-vs-nfc. [Accessed: 05- Aug- 2015].

[22] NFC World+, 'List of NFC phones', 2015. [Online]. Available: http://www.nfcworld.com/nfc-phones-list/. [Accessed: 05- Aug- 2015].

[23] Hospitality Net, 'Hospitality Net - ASSA ABLOY Hospitality Revolutionizes Hotel Security and Convenience with Industryâ€™s Premier Mobile Access Solution', 2015. [Online]. Available: http://www.hospitalitynet.org/news/4070162.html. [Accessed: 05- Aug- 2015].

[24] Developer.android.com, 'Dashboards | Android Developers', 2015. [Online]. Available: https://developer.android.com/about/dashboards/index.html?utm_source=suzunone. [Accessed: 05- Aug- 2015].

[25] 2015. [Online]. Available: https://web.eecs.umich.edu/~prabal/teaching/eecs373-f10/readings/ARM_Architecture_Overview.pdf. [Accessed: 05- Aug- 2015].

[26] 2015. [Online]. Available: http://konstantin.solnushkin.org/teaching_reports/intro_to_hpc/2007/harvard_architecture.pdf. [Accessed: 05- Aug- 2015].

[27] M. Verle, *PIC Microcontrollers*. Beograd: Mikroelektronika, 2008.

[28] Ladyada.net, 'PIC vs. AVR smackdown', 2015. [Online]. Available: http://www.ladyada.net/library/picvsavr.html. [Accessed: 05- Aug- 2015].

[29] Quora.com, 'What is the difference between 8051, PIC, AVR and ARM? - Quora', 2015. [Online]. Available: http://www.quora.com/What-is-the-difference-between-8051-PIC-AVR-and-ARM. [Accessed: 05- Aug- 2015].

[30] Solar-electric.com, 'What is Maximum Power Point Tracking (MPPT)', 2015. [Online]. Available: http://www.solar-electric.com/mppt-solar-charge-

controllers.html. [Accessed: 05- Aug- 2015].
[31] Genasun.com, 'Genasun MPPT Controllers for Lead-Acid Batteries | GenasunGenasun', 2015. [Online]. Available: https://genasun.com/products-store/mppt-solar-charge-controllers/mppt-for-lead-acid/. [Accessed: 05- Aug- 2015].
[32] I. Solarcraft, 'PWM vs MPPT Solar Charge Controllers | Solarcraft', *Solarcraft.net*, 2015. [Online]. Available: http://solarcraft.net/articles/comparing-pwm-and-mppt-charge-controllers/. [Accessed: 05- Aug- 2015].
[33] Altenergymag.com, 'A Comparison of Lead Acid to Lithium-ion in Stationary Storage Applications | AltEnergyMag', 2015. [Online]. Available: http://www.altenergymag.com/content.php?post_type=1884. [Accessed: 05- Aug- 2015].
[34] 2015. [Online]. Available: http://datasheets.maximintegrated.com/en/ds/DS2438.pdf. [Accessed: 05- Aug- 2015].
[35] Xamarin.com, 'How do I share code across platforms with Xamarin? and other FAQs - Xamarin', 2015. [Online]. Available: http://xamarin.com/faq. [Accessed: 05- Aug- 2015].
[36] A. Overview, 'Android Studio Overview | Android Developers', *Developer.android.com*, 2015. [Online]. Available: http://developer.android.com/tools/studio/index.html. [Accessed: 05- Aug- 2015].
[37] A. Gajani, '4 Online Collaboration Tools for Developers', *Blog.monitor.us*, 2015. [Online]. Available: http://blog.monitor.us/2013/05/cc-in-review-4-online-collaboration-tools-for-developers/. [Accessed: 05- Aug- 2015].
[38] L. Wildbit, 'Wildbit â€“ We create web products to help business collaborate and communicate more effectively', *Wildbit.com*, 2015. [Online]. Available: http://wildbit.com/. [Accessed: 05- Aug- 2015].
[39] L. Wildbit, 'Beanstalk Plans & Pricing', *Beanstalkapp.com*, 2015. [Online]. Available: http://beanstalkapp.com/pricing. [Accessed: 05- Aug- 2015].
[40] Atlassian, 'JIRA - Issue & Project Tracking Software | Atlassian', 2015. [Online]. Available: https://www.atlassian.com/software/jira. [Accessed: 05- Aug- 2015].
[41] Atlassian, 'Pricing | Atlassian JIRA', 2015. [Online]. Available: https://www.atlassian.com/software/jira/pricing. [Accessed: 05- Aug- 2015].
[42] GitHub, 'Build software better, together', 2015. [Online]. Available: https://github.com/pricing. [Accessed: 05- Aug- 2015].
[43] C9.io, 'Cloud9 - Your development environment, in the cloud', 2015. [Online]. Available: https://c9.io/. [Accessed: 05- Aug- 2015].
[44] C9.io, 'Cloud9 - Plans and Pricing', 2015. [Online]. Available: https://c9.io/web/site/pricing. [Accessed: 05- Aug- 2015].
[45] Developer.apple.com, 'HomeKit - Apple Developer', 2015. [Online]. Available: https://developer.apple.com/homekit/. [Accessed: 05- Aug- 2015].
[46] Developer.apple.com, 'Choosing a Membership - Support - Apple Developer', 2015. [Online]. Available: https://developer.apple.com/support/compare-memberships/. [Accessed: 05- Aug- 2015].

[47]G. Publishing, 'Get Started with Publishing | Android Developers', *Developer.android.com*, 2015. [Online]. Available: http://developer.android.com/distribute/googleplay/start.html. [Accessed: 05- Aug- 2015].

[48] Genasun.com, 'Genasun GV-5 65W 5A Solar Charge Controller with MPPTGenasun', 2015. [Online]. Available: http://genasun.com/all-products/solar-charge-controllers/for-lead/gv-5-pb-5a-solar-charge-controller/. [Accessed: 05- Aug- 2015].

[49] Ti.com, 'CC2640 | Bluetooth / Bluetooth Low Energy | Wireless Connectivity | Description & parametrics', 2015. [Online]. Available: http://www.ti.com/product/cc2640. [Accessed: 05- Aug- 2015].

[50] 2015. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/22008E.pdf. [Accessed: 05- Aug- 2015].

[51] 2015. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2640.pdf. [Accessed: 05- Aug- 2015].

[52] E. Inc, '690-004-621-013 EDAC Inc | 151-1080-ND | DigiKey', *Digikey.com*, 2015. [Online]. Available: http://www.digikey.com/product-detail/en/690-004-621-013/151-1080-ND/806179. [Accessed: 05- Aug- 2015].

[53] Universal Power Group. 2015. [Online]. Available: https://a89b8e4143ca50438f09-7c1706ba3fabeeda794725d88e4f5e57.ssl.cf2.rackcdn.com/spec_sheets/files/000/003/520/original/ubD5741-spec.pdf?1440178672. [Accessed: 30- Sept- 2015].

Appendix D: Datasheets

Motor
http://rollertrol.com/store/en/12v-tubular/14-projector-screen-motor.html

Monocrystalline Solar cells
http://eshop.terms.eu/_data/s_3386/files/1379942540-sunpower_c60_bin_ghi.pdf

Thin-Film Solar Cells
http://www.powerfilmsolar.com/sitevizenterprise/website/cgi-bin/file.pl/powerfilmsolar/media/products/R14_Sales_Sheet_D6872305FE172.pdf

Boost Converter
http://cds.linear.com/docs/en/datasheet/1871fe.pdf

Battery
http://upgi.com/Themes/leanandgreen/images/UPG/ProductDownloads/D5741.pdf

Battery Monitor
http://www.ti.com/lit/ds/symlink/bq34z100-g1.pdf

MPPT Charge Controller
file:///D:/Documents%20and%20Settings/%C3%A4dmin/Desktop/GV-5_2012_manual_1112.pdf

Microcontroller Unit (MCU)
http://www.ti.com/lit/ds/symlink/cc2640.pdf

MCU Development Board
http://www.ti.com/lit/ug/swru321a/swru321a.pdf

Temperature Sensor
https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

LCD Screen
http://www.newhavendisplay.com/specs/NHD-C0216AZ-FN-GBW.pdf