Power Rangers
Low Level Design

ANJOLA LANRE-LADENEGAN, MEAGHAN HANNON,
SARAH RITTER, EVAN SYERS

6 MAY 2015

Contents

1 Introduction

This project was pursued primarily based on the constituent members' interest in large power systems and energy solutions. The microcontroller requirement for the project and considerations of cost and safety made a project based on power electronics a sensible path. Professor Michael Lemmon at the University of Notre Dame was approached with these interests in mind and a potential project using certain resources already in the power electronics lab sparked the group's interests.

The fundamental phenomenon leading to the necessity of this project is dependence of power generated by a solar array on the voltage it is being operated at. Depending on the irradiance of light, the point at which the solar array outputs maximum power occurs at a different voltage. The power curves for the array that will be used for the project were achieved by sweeping the voltage with a rheostat connected to the output. The result is shown in Figure 1.

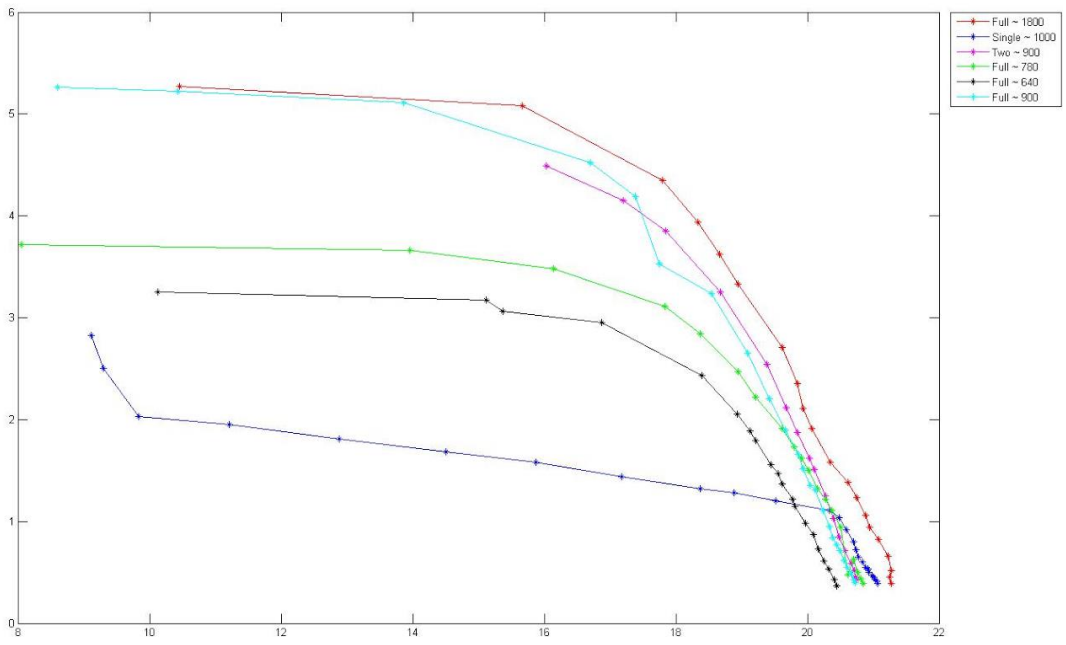


Figure 1: Current/Voltage Relationship for Solar Array

The problem that the group will attempt to solve is a problem of optimizing the power that is outputted from the array. If the solar array were simply attached to the battery, it would

be driven at the battery voltage, which is 12V; at maximum irradiance this would charge the battery with a current of about 5A. A better solution would be to have a buck converter that establishes the voltage of the solar array at a point closer to the maximum power point, such as 18V. At 18V the output power of the array would be considerably higher, which when converted to 12V would charge the battery at a rate higher than it would be without a converter. The goal of this project is to have a converter that tracks the maximum power point, and drives the converter so that maximum power is always being input to the battery.

This project is extremely relevant today because solar energy is highest during the day from around 11am-2pm. The problem, as seen in Figure 2, is that the highest energy consumption occurs before the sun is up or after it goes down. High consumption occurs around 9am and from 6-9pm. Using batteries to store solar power for high consumption times would prevent overloading the grid.

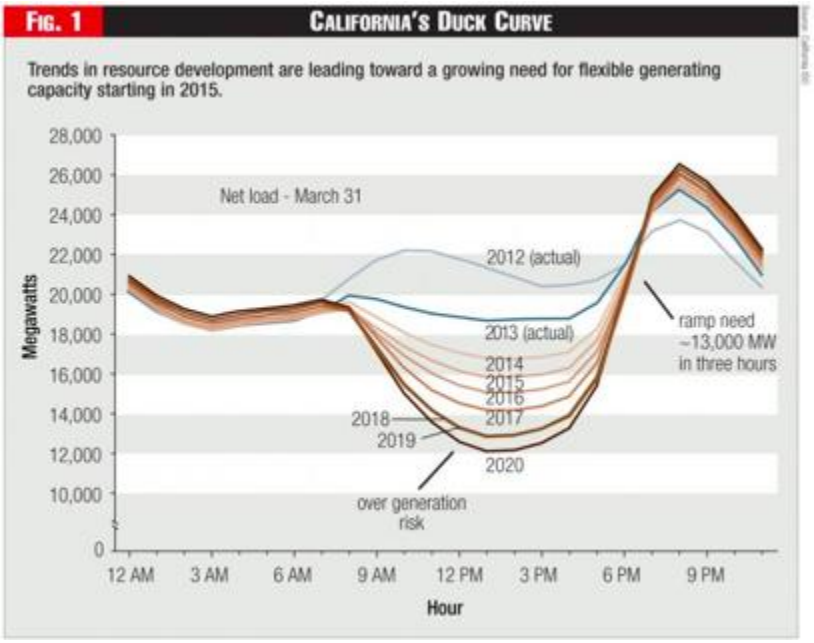


Figure 2: Typical Power Usage Curve

A specific example of a problem with solar generation is that it produces peak power at around 3 p.m., while the max demand on a distribution network typically occurs at 5 p.m. There is no way a project could be done involving an entire distribution network, but the problem of

how to effectively store solar energy is very central to solving this problem and could be achieved on a much smaller basis. Because efficiency is the most commonly used parameter to compare the performance of solar cells we will be working to maintain the maximum efficiency. Solar energy is difficult to work with because it gives off varying power that isn't consistent. Most solar cells have an efficiency around 20% which means a lot of energy is lost in the process. The IV characteristics of solar cells change depending on the time of the day. For instance, clouds cause shadowing which shifts the IV curve down. By maximizing the IV characteristics the battery would charge more efficiently.

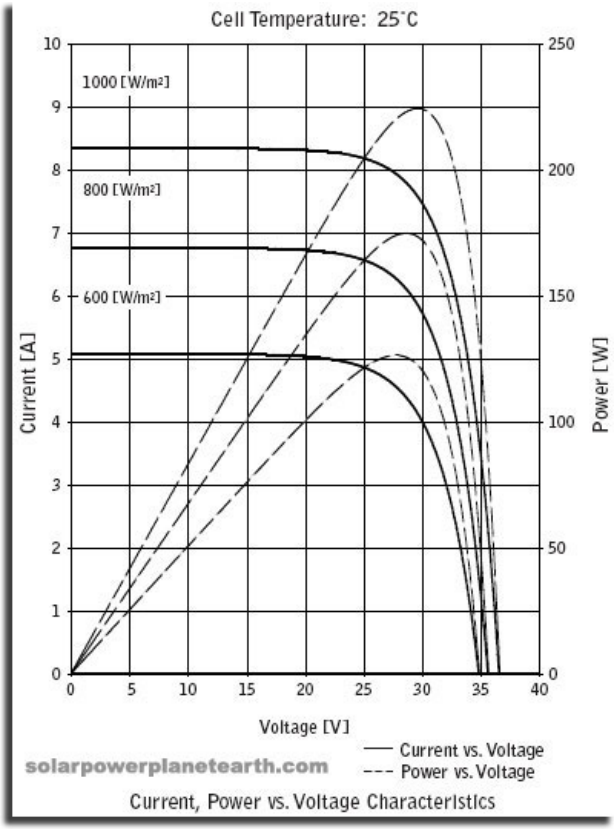


Figure 3: I/V Curves with the Maximum Power Points

This project will charge a sun xtender battery using a converter to increase efficiency to increase the use of solar power. We will be using the solar panel that Dr. Lemmon has in his lab in Figure 4.

Figure 4: Solar Panel

2 Detailed System Requirements

1. The embedded intelligence required for the system will be a microcontroller. The microcontroller must be able to read in the voltage and current states from the array as well as the battery's state of charge. Based on those values, the microcontroller must indicate a duty ratio to achieve the maximum power point.
2. The microcontroller must be able to output this duty ratio in a fashion that the converter will be able to implement it on the switching device.
3. Since this project involves high power circuits as well as electronics, the system must be built such that the voltages and currents into the electronics are of appropriate magnitude. Isolation should also be carefully planned such that there is no chance of equipment being damaged.
4. It is a requirement of the system that it minimizes losses such that the final charging current is an improvement upon where it would be at a set point. This will involve using high-accuracy and high-performance devices in some part of the system.

5. There is a small inverter on the cart that will allow the necessary supply circuits to power the devices that are used. The microcontroller, current sensor, power MOSFET, and any op amps used will require supply circuits to run.
6. The final product will be a unit that should not need to be moved or interacted with in any way. To change where the solar power is being routed (for example, for the purpose of testing the converter) the sockets and banana clips already configured on the cart will be used.
7. The final product must be thermally stable. Any power dissipated may be exacerbated in certain environments.
8. The system will be installed on the cart for the solar array. Nothing will need to be used; the system should passively improve the charging efficiency of the battery. If desired, the power from the battery can be used to run various devices through the inverter, but that is a feature that is supplementary to the project.
9. There will be several precautions taken to ensure that the moderately high voltages and currents do not pose any safety hazards. Properly insulated wires will be used and secure connectors will be used to connect energized parts. These connectors will only be moved when the circuits are de-energized.
10. The size of the system should be such that is can be contained on the cart that the solar array is on. Weight is not a major consideration. The system should be mechanically robust enough that transporting the cart and exposing it to the elements will not compromise its operation.
11. The system must be efficient enough that it significantly improves the charging rate of the battery.

3 Detailed Project Description

3.1 System Theory of Operation

The system relies on DC-DC converter theory to achieve its desired purpose. By using storage devices in combination with switches, it is possible to transform a DC voltage to another DC voltage with high efficiency. The storage elements are realized by large inductors and capacitors and the switches are realized by MOSFETs and diodes. In this system, it is desired to step down a higher voltage to achieve a lower voltage with a higher current. To do this, a standard buck converter architecture is utilized. In this architecture, the source is connected in series with a switch that can be controlled (a MOSFET) and the storage element (the inductor). When the MOSFET is conducting, the current magnetizes the core of the inductor, which then discharges when the MOSFET is nonconducting. A figure showing these two states is shown below.

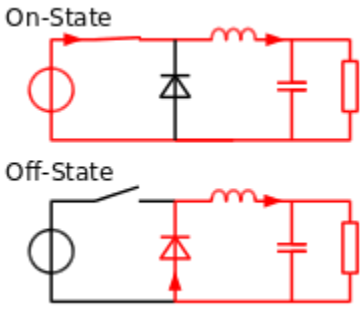


Figure 5: Buck Converter States

The conduction state of the MOSFET is controlled via a voltage applied to the gate. In practice, this voltage is sourced with a specialized driver chip. A diode from common to the node between the inductor and MOSFET allows a current path for this discharge to occur. Using standard circuit analysis equations and differential equations that describe the behavior of storage devices, it can be proved that in this architecture, the output voltage is related to the input voltage by the duty ratio, or the percentage of time the switch is conducting during a switching period. This relationship forms the basis for Pulse Width Modulation (PWM). The

converter can be carefully controlled to achieve different voltages by changing how long the MOSFET is conducting and nonconducting. This can be done either by using a PWM chip, which will create a duty cycle by comparing a reference voltage with a ramped oscillation, or by using a microcontroller that can output a similar signal.

The power point tracking element of the system operates by obtaining information about the input voltages and currents and comparing it to a table of measured values. The obtaining of the input voltage and current is done by a small-current voltage divider and a Hall-effect current sensor, respectively. The microcontroller will match the value to the closest point on one of several predetermined, interpolated I-V curves, and will determine the ideal voltage that the input should be driven at to achieve maximum power, which will change based on the irradiance. To achieve this voltage, the microcontroller must do a calculation to determine what duty ratio will achieve this voltage depending on the current output level. It is also possible that either the battery will be fully charged or the irradiance will be so low so as to not be able to have a voltage large enough to step down, in which case the converter will be turned off.

The solar panel can be represented by the following figure:

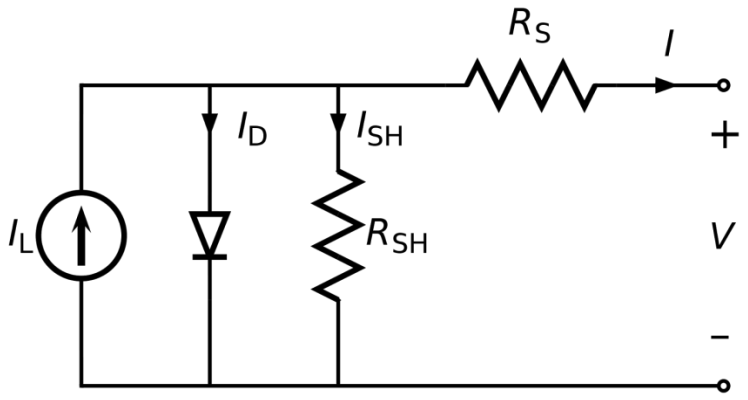


Figure 6: Solar Panel Equivalent

Note the diode characteristic of the solar panel equivalent. If the voltage of the battery exceeds the voltage supplied by the solar cell, as may occur during periods of low irradiation, the battery would be allowed to discharge through a relatively low impedance path, which is undesirable. The magnitude of I_L is directly related to the irradiation: the higher the irradiance (W/m^2) is, the more current will be generated. I_D is a loss mechanic that depends on the voltage

at the node and the panel. I_{SH} is another loss mechanic and is given by the voltage over R_{SH}. Ideally, the panel would have a low R_S, high R_{SH}, and low I_O (which is a factor in calculating the diode current).

3.2 System Block Diagram

The system block diagram is seen below in Figure 2.

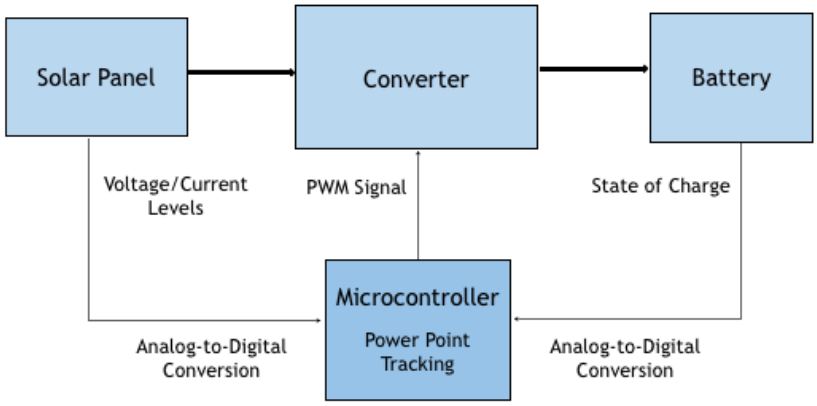


Figure 7: System Block Diagram

The solid lines represent the high power paths in the system. Between the solar panel and the converter, this could be up to 22V and 5A. After the converter, the voltage level will be the battery voltage, which is nominally 12V, at a higher current. The two inputs to the microcontroller block give the microcontroller the information it needs to execute the power point tracking feature. These inputs are voltages scaled to the ratings of the microcontroller. The PWM signal is calculated as a result of the power point tracking and is used to control the MOSFET in the converter via a driver.

3.3 Detailed Design/Operation - Converter

The converter circuit must be capable of taking a high input voltage and reducing it to the voltage of the battery. The input voltage for which the converter is to be optimized is 17-

19V, which is where the maximum power lies based on Figure 1. The voltage of the battery is nominally 12V, although this will vary with the state of charge. The converter must do the conversion very efficiently (i.e. with minimum switching loss). The microcontroller will be responsible for disconnecting the power from the load in the case of low irradiance, but the converter must be designed such that reverse power flow through the diode characteristic of the array is prevented. The MOSFET and diodes must be sized to block whatever voltage spikes in switching scenarios. These spikes are more problematic when there are inductive circuit components that cause ringing, the reduction of which will be of high concern. The converter should also be able to disconnect and reconnect without any user adjustments. The driver circuit for the switch must be able to appropriately respond to changes in the duty ratio.

The buck converter circuit consists of a 12V input voltage to power the driver and an input voltage from the solar panel. For testing, a variable DC supply is used instead of the solar panel. The converter has an IR2127 driver chip that controls high power switches on the gate of the IRF640 MOSFET. The driver has two bypass capacitors around it to reduce the effects of noise. The switching between the MOSFET and diode is regulated by the duty cycle so that the converter outputs the desired 12V to the battery for most efficient charging. A 4Ω rheostat is being used as the load for testing. A PWM circuit is being used to control the converter and two potentiometers are used to adjust the duty cycle and frequency. When subsystems are combined the microcontroller will output the PWM signal with the desired duty cycle at set frequency of 100kHz.

Device Selections and Details

Input/Output Connectors: For the input and output connectors, Phoenix MSTBV3 horizontal connectors were used. These are screw terminal connectors that can be used to connect to different outputs. For the input, banana plugs were required, whereas for the output, it was easiest to connect using terminal clamps meant for batteries. The clamps chosen were Mueller copper coated terminations. The size of wire chosen to make both the input and output

connections was 10 AWG. The typical ampacity of 10 AWG wire is 30A, which ensured there would be no problem carrying the expected currents in the converter.

Input Diode: The input diode selected was the UF5402. The advantageous qualities of this diode were its low forward voltage drop and high reverse blocking voltage. Minimizing the forward voltage drop was of high importance since this drop directly impacts efficiency of the entire converter. Having a high reverse blocking voltage was important to prevent the case in which the battery discharges through the diode characteristic of the solar panel. The UF5402 is rated to block a DC voltage of 200 volts. This far exceeds the expected spikes of any situation in the products operation, so given more time, a more ideal diode may exist. The current rating of the UF5402 diode is 3A; since at high irradiance currents almost 4A could be observed, two UF5402 diodes were placed in parallel. So long as the forward drops of each diode are equal, the current should in operation be roughly split between the two diodes.

MOSFET: The MOSFET selected for the system was the IRF640. This device was ideal due to its simple drive requirements and its high ratings. The IRF640 was attached to a heat sink to ensure that it would dissipate power correctly. This device possessed a relatively high $R_{DS(ON)}$ of .18 , which could equate to significant power losses at higher operation currents. Other power MOSFETs may perform better in this regard, but the project proceeded with the IRF640 due to the ease of driving it.

Driver: The IR2127 was selected as the driver for the system. This driver was selected for its ability to perform the task of sourcing the current to turn on the MOSFET given a logic input signal as well as its current sense capabilities that helped in protecting the MOSFET. The driver was implemented in the circuit board using an 8-DIP socket, so that it could be easily replaced. During early testing, if the input was changed too rapidly, the driver would dissipate too much power and would become non operational. This event was rare or nonexistent in the final implementation of the circuit, but the socket remained such that the driver could be easily

changed out. Several capacitors and a bootstrap diode, 1N4148, were selected in accordance with the application notes of the IR2127.

Buck Diode: The diode selected for the diode in the buck circuit (see Figure 5) was the UMUR2020. The most important specification that was looked for in the diode was the recovery time. The switching occurs across this diode so a short time is desirable. The UMUR2020 boasts a recovery time of about 35 ns at the 1A level. Another important specification is the forward voltage drop, which for the UMUR2020 is .85V. The device is rated to carry an average current of 20A, which gives a comfortable margin compared to the levels expected in the system. The blocking voltage of the device is 200V which is adequate when switching between modes.

Inductor: The inductor used was a lab-wound inductor around a PQ32/20 core. AWG 18 transformer wire was used to complete the windings, which can carry roughly 14A. Seven windings were made, which equates to an inductance of 344 mH. Based on KVL and KCL switching steady state equations, this was determined to be adequate to maintain the output current at a very small ripple.

Buck Capacitor: The capacitor chosen was of value 680 uF. This was determined to be adequate based on the steady state equations to maintain the output voltage within a very small ripple. A larger capacitor could have been chosen, but this would have added to the cost and size of the system with very little benefit.

3.4 Detailed Design/Operation - Voltage/Current Sensing

3.4.1 Solar Array Voltage

The voltage reading circuit will utilize a voltage divider with high resistances to scale down the voltage to the correct level below the microcontroller's 3.3V. These resistances have

been decided on as 5.6kΩ and 47kΩ. The voltage across the 5.6 kΩ resistor will be accepted into the microcontroller. That voltage will then be multiplied by the equation:

$$V_m=(5.6*(5.6+47))*V_d$$

The measured voltage the stepped-down voltage from the actual voltage, Vd. The voltage reading circuit will be implemented on a printed board with wide traces. The connection will be made by a firm connector and housing.

3.4.2 Solar Array Current

The current will be measured by a LAH 25-NP current transducer by LEM. This unit will take a current of up to 25A and output a scaled-down current. The current sensor will be displayed on the LCD using the equation:

$$V_m = I_m\ (11000)*249$$

The measured voltage across the 249 Ω resistor will be inputted into the above equation to display the accurate current. The turns ratio that the transducer is being wired for is 1:1000. This current passing through a resistor will provide the voltage in the range that the microcontroller will use. The resistor will be a 1% accuracy resistor of value 249Ω for the project.

3.4.3 Battery Voltage

The battery voltage reading circuit will utilize a voltage divider to scale down the voltage from 12V to less than 3.3V. The voltage needs to be scaled down to be compatible with the microcontroller. The resistances used for the divider are 6K and 20K. The output voltage is taken across the 6K resistor. The battery will multiply the input voltage by the equation:

$$V_m=(6*(6+20))*V_d$$

The measured voltage is the stepped-down voltage from the actual voltage.

This divider circuit would be implemented on a printed board with wide traces. Connections would be made by a firm connector and housing.

3.5 Detailed Design/Operation - Microcontroller

3.5.1 Microcontroller - Hardware

The design of the printed circuit board containing the microcontroller for this project is based off the layout of the prototype kit board used in EE 41430. The microcontroller used is the PIC32MX270F265B which allows for a variety of applications. This microchip is smaller than the standard 64 pin microchip that was used on the kit board. In order to power this kit board, a voltage regulator is utilized a LD1117. This voltage regulator will step the 12V from the battery down to the 5V necessary to power the LCD display. Another LD1117 is also used to step down the 5V to 3.3V in order to power the microchip and other devices. An FT230XS is used to power the board from a USB port. Two MCP6002 IC opamps are used for the inputs of the sensing circuits and a 6N137 optocoupler is used to isolate the PWM signal

3.5.2 Microcontroller - Software

3.5.2.1 PowerPoint Tracking Algorithm

The PowerPoint Tracking algorithm receives the voltage and current readings from the analog to digital conversions and then outputs a duty cycle used to created a PWM signal for the converter circuit. This algorithm searches through a three-dimensional array containing previously measured values of voltage and current from the solar array in order to determine an ideal power. This power is associated with a certain ideal voltage that is combined with the voltage reading from the analog to digital conversion of the battery in order to calculate the duty cycle.

In order to pick the correct maximum power point, nested for loops were used to search through each value in the arrays. The distance formula was then used to find the closest point to the curve. When this point is found, an algorithm is used to find the maximum power on that

curve. This algorithm used a for loop to cycle through the needed curve and a comparison technique to find the largest value of power on the curve. The voltage corresponding to this maximum power is then used to calculate the duty cycle by being divided by the nominal voltage of the battery which is 12V.

The flowchart for the algorithm is shown below:

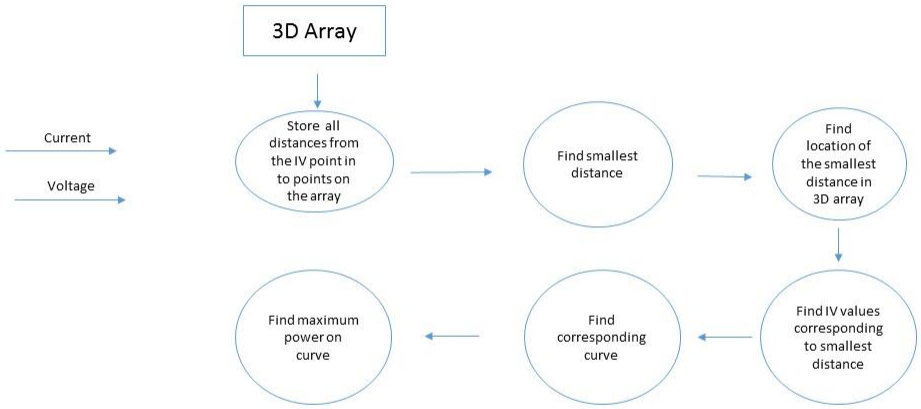


Figure 8: PowerPoint Tracking Flow Chart

3.5.2.2 Analog to Digital Conversion

In order to provide the PowerPoint Tracking algorithm with values of the voltage and current of the solar array and the battery voltage, it is necessary to perform analog to digital conversions. The microcontroller has this functionality and when programed correctly can sample the analog signals ranging from 0 to 3.3V and output a digital value used in the PowerPoint Tracking algorithm. These values are also displayed on the LCD screen using printf functions.

3.5.2.3 Pulse Width Modulation

A Pulse Width Modulation signal is needed as an input to the controller in this design. In order to accomplish this, the Output Compare Module of the microcontroller is utilized to created a PWM signal. The algorithm for this module requires a duty cycle which will be the output of the PowerPoint Tracking algorithm. The frequency of this signal has been preset to based on the specifications of the converter. The system clock is also set at 80MHz as originally defined in the Configuration Bits. Based on these inputs, certain registers are set in order obtain the correct PWM signal.

3.6 Interfaces

The Microcontroller and the converter will need many connections to properly work together. The converter will use the sensing circuitry to output voltage and current in as well as the battery voltage out. The converter uses voltage dividers to step the voltage down to a level that the microcontroller can handle below 3.3V. The stepped-down voltage coming in from the solar panel will be sent to the microcontroller which will display its value to the LCD screen by calculating the actual value. The voltage from the current transducer is below 3.3V so it does not need to be stepped down. The LCD will display the current coming from the solar panel using this equation:

$$V_m = I_m\ (11000)*249$$

The output voltage to the battery will work the same way as the solar input voltage. The sensing circuit will step it down and the microcontroller will calculate the actual value to send to the LCD screen.

For the converter to work, it will need the pulse width modulator (PWM) signal. Many converters use a PWM chip and circuitry. These use potentiometers to vary the duty ratio and frequency. Because the converter will be continuously changing the duty ratio due to the

maximum power point tracking algorithm the Microcontroller will need to send the pwm signal to the converter using jumper cables. The signal will vary the duty ratio based on the power point curve that the input values are closest to and will be set at a frequency of 100 kHz. The PWM signal will be sent to the driver which will power the MOSFET.

4 System Integration Testing

4.1 System Integration Testing

The systems were first tested together the week of March 30, 2015. The testing consisted of using the microcontroller board to provide the duty cycle for the converter board. The DC voltage source was used to imitate the solar cell and the rheostat was used as the load. The voltage was slowly increased to a point of 18V with careful attention that none of the device ratings were exceeded. For the integration test, the microcontroller provided a fixed duty ratio amid concerns that the duty cycle cutting out could damage the driver chip. It was successfully demonstrated that the microcontroller board could effectively be used to provide the logic input to the driver, and that the converter could adjust the input/output voltage ratio based on this duty cycle.

4.2 System Integration Testing Compared to Requirements

The integration test introduced several problems that must be addressed before the project requirements are met. The first is that the duty ratio must actively change in response the adjustments in the input power. The integration demonstrated that this will be easily achieved once the algorithm is in place on the microcontroller board. Some issues of the system's robustness in the event of sudden input power were introduced during the system integration test. Various filters to reduce the effects of transients were considered and a low pass RC filter was added on the second revision of the board to address this issue.

5 User's Manual/Installation manual

5.1 How to install your product

The 2015 Notre Dame EE Power Ranger's Efficient Solar Charging system consists of two printed circuit boards. It is intended for use with a single 100W solar panel and 12V deep discharge battery. To install the product, the boards should be mounted in close proximity to the solar panel and battery in the most sheltered area possible.

5.2 How to setup your product

Several connections must be made for the system to work correctly:

- The output of the converter board must be connected to terminals of the battery with the correct polarity using the insulated battery terminal clamps.
- The voltage from the battery must be connected to the regulator on the converter board.
- The solar panel power must be connected to the input plug of the converter board.
- The connections between the converter and microcontroller boards must be made. There is both a connection for the PWM signal from the microcontroller board and a connection for the regulated 12V from the converter board to the microcontroller board.
- A driver chip should be placed in the socket.

5.3 How the user can tell if the product is working

To test whether the system is working correctly or not, the user can use a high impedance probe to check the voltages in the system. On the input side of the MOSFET the voltage should be anywhere from 17-22V and may vary depending on the irradiance. On the output side of the MOSFET, the voltage should be about 12V, although this may vary with the

state of charge of the battery. Another point of interest is the gate voltage. If the system is operating correctly, this should be a square wave with a magnitude of maximally 30V. To test the tracking capabilities of the system, the user can probe either the gate or the PWM signal and gradually shade the solar panel. The duty ratio should adjust to match the maximum power point shown in Figure 1.

5.4 How the user can troubleshoot the product

If there is a problem with the system, the first thing that should be done is to replace the driver chip, after the boards have been disconnected from the power. The driver chip is very sensitive to large changes in power and has a tendency to fail. If replacing the driver is not sufficient, the user should attempt to identify the device that is problematic. First investigate the input PWM signal to the driver and the MOSFET gate voltage. If there is an issue with the PWM signal in, there is likely an issue with the microcontroller and sensing circuitry. If the gate voltage does not show the expected square wave to match the PWM signal, there is likely an issue with the driver setup. If the driver has not been replaced, do so, and ensure that the bootstrap diode and capacitors are correctly connected.

6 To-Market Design Changes

Several changes would be made to the product to make it more viable to market.

The product would be greatly simplified to allow for easier installation. This would most likely involve combining the boards to reduce the number of connections that must be made. For this to be done, isolation must be introduced to protect the microcontroller from the high currents present on the actual converter.

Several devices would have been changed to lower the cost of the product. The current sensing device had specifications that far exceeded the demands of the system. A device with lower cost closer to the system needs could have been found. The -12V/+12V regulator was a

somewhat expensive device and could possibly be replaced by a different device or a clever divider. In a market version of the product, a factory-wound inductor would have been used. This would have reduced costs and improved performance.

Similarly, several devices could have been replaced to improve the performance of the system (possibly at higher cost). The ultrafast recovery diodes used for the input introduced a large potential drop. These were necessary to prevent the battery from discharging through the diode characteristic of the solar panel but the need for them could have been eliminated with controls. The MOSFET used for switching had a relatively large $R_{DS,On}$ which would have impacted the performance of the system. Several of the resistor values could have been more carefully chosen to reduce power loss in sensing and improve accuracy of measurements.

7 Conclusions

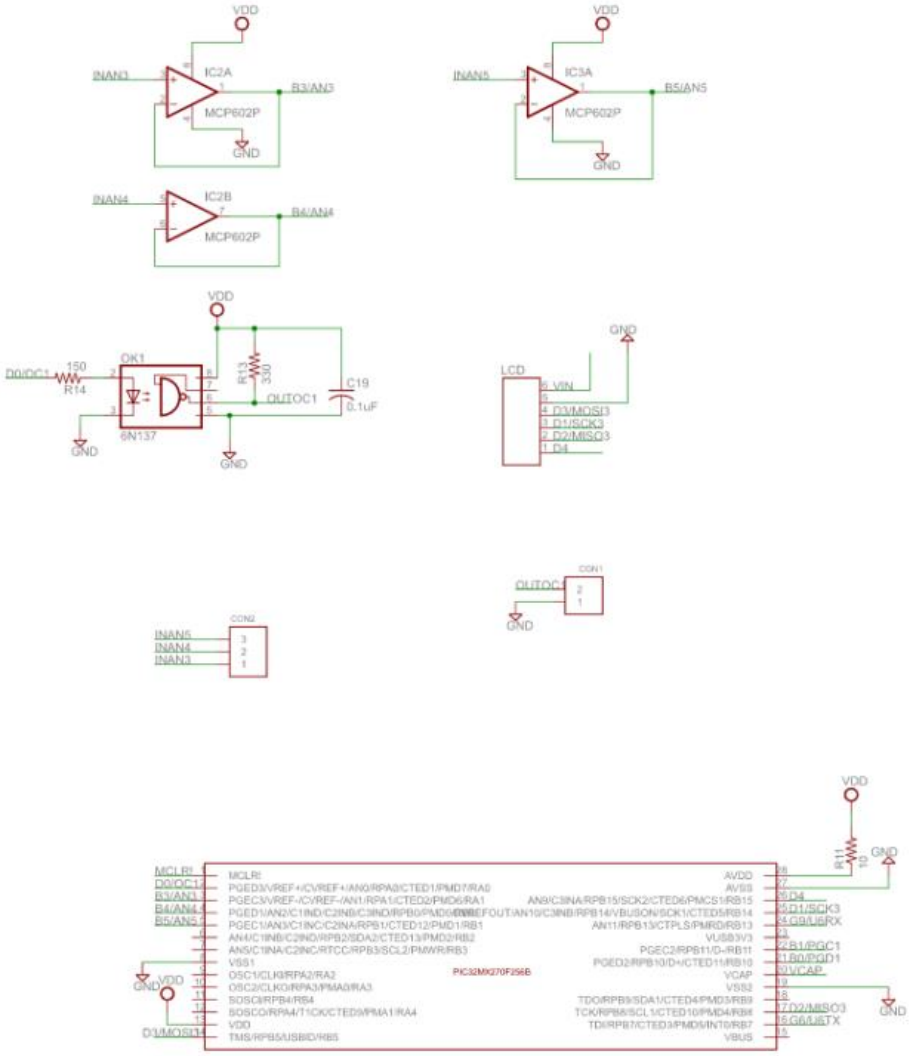
The system results in a viable improvement to charging a battery using a solar panel. The actual act of charging the battery is desirable as the maximum power a user will get from a panel occurs in early afternoon, when it is unlikely that the user would have a use for the power. The maximum power measured during data collection approached 85W, slightly over 20V with a current of about 4A. Running this through an ideal converter would have resulted in a charging current of (20/12)*(4A)=6.66A. The actual implementation of the converter was found to have an efficiency of about 80-85%, which would put the current in the example at about 5.5A, an improvement of almost 40%. The power point tracking aspect of the system is much more difficult to assess the value of, but qualitatively it only further improves the charging efficiency. The differences between an implementation of a “static” buck converter and one that tracks the power point is the sensing circuitry and the modulation of the pulse width that the switch is fired at. The sensing circuitry introduced almost no loss and added minimal cost to the system, and the effect of changing the pulse width is that the input is driven at a voltage that idealizes its power on the I-V characteristic of the solar panel.

There would be several difficulties in widespread use of the system. The most prominent would be the differing I-V characteristics between different users' panels. Even if data on specific panels was known, it could vary based on the level of shading and the angle of the panel.

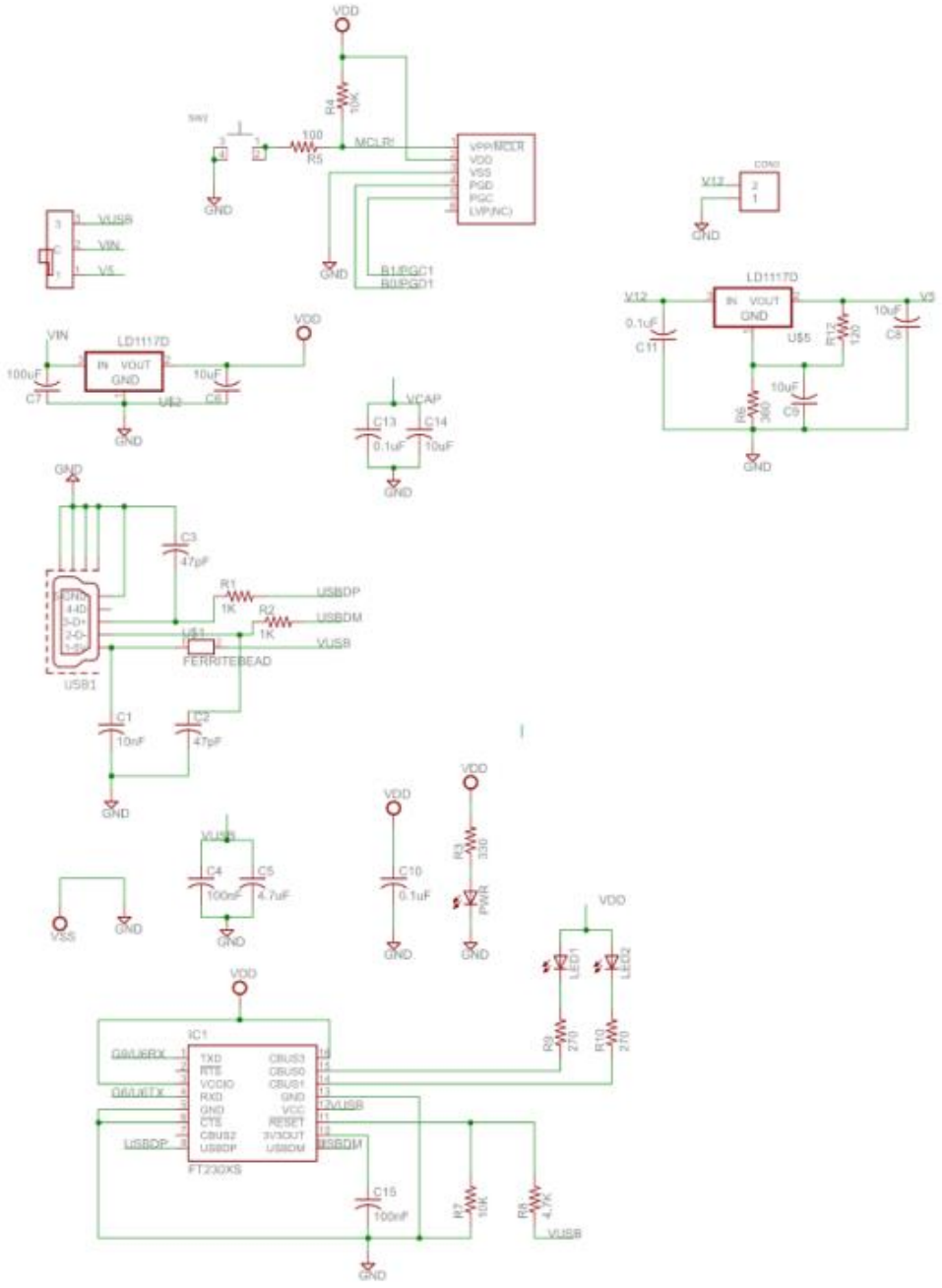
8 Appendices

Appendix A:

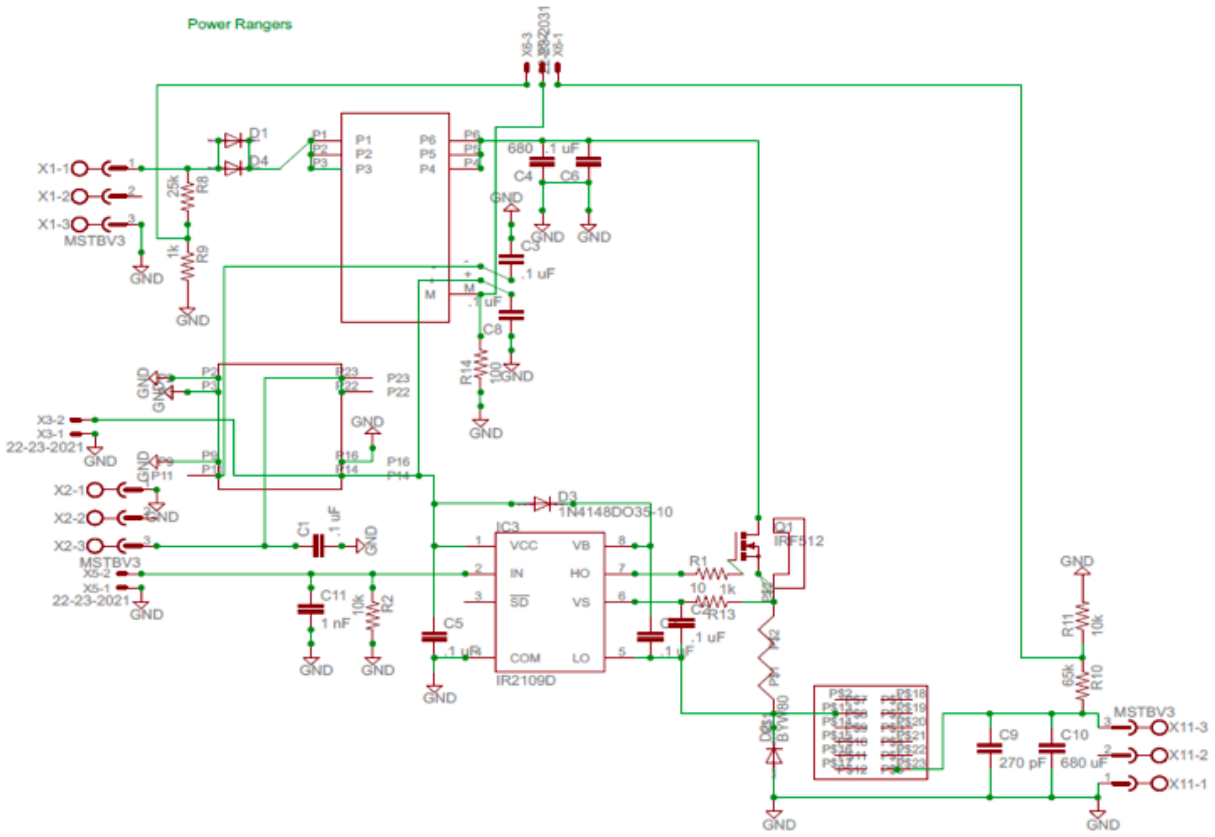
Eagle Schematics, Board Files and PCB boards



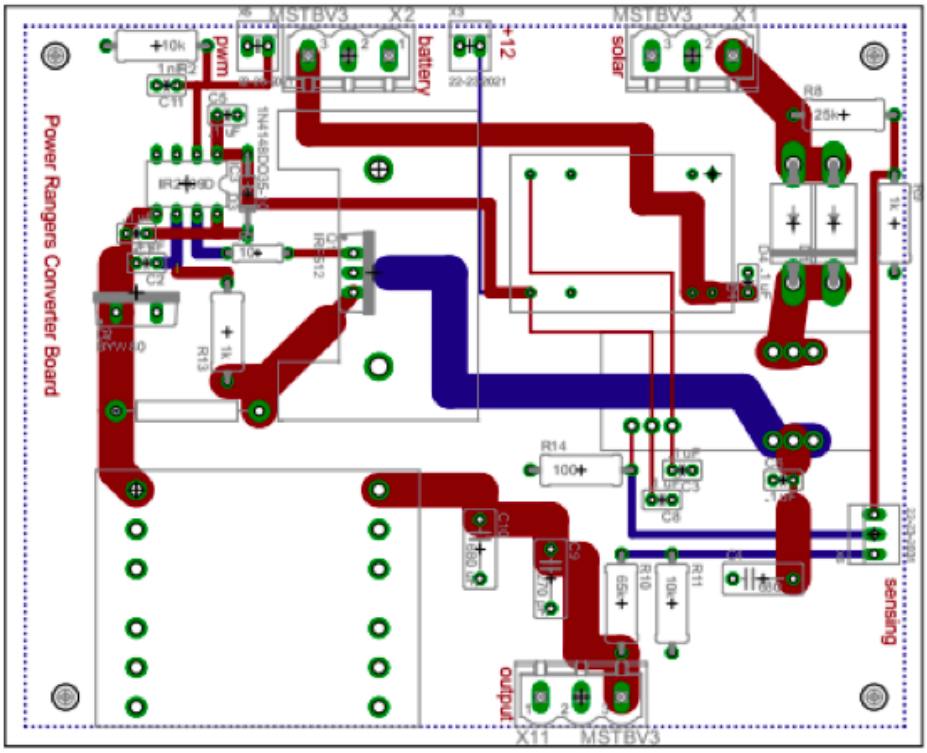
Appendix Figure 1: Microcontroller Schematic I



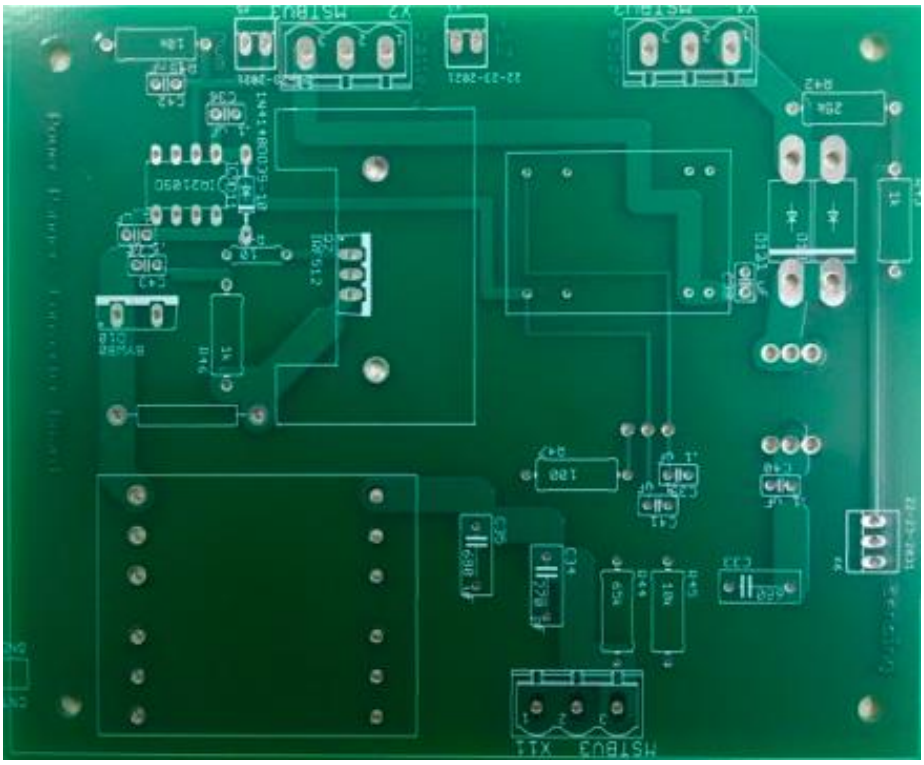
Appendix Figure 2: Microcontroller Schematic 2



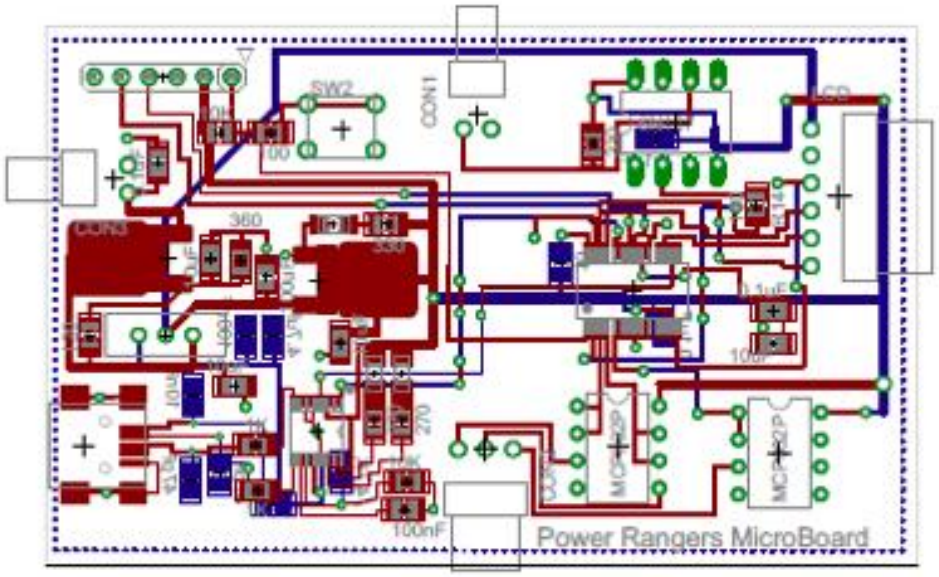
Appendix Figure 3: Converter Schematic



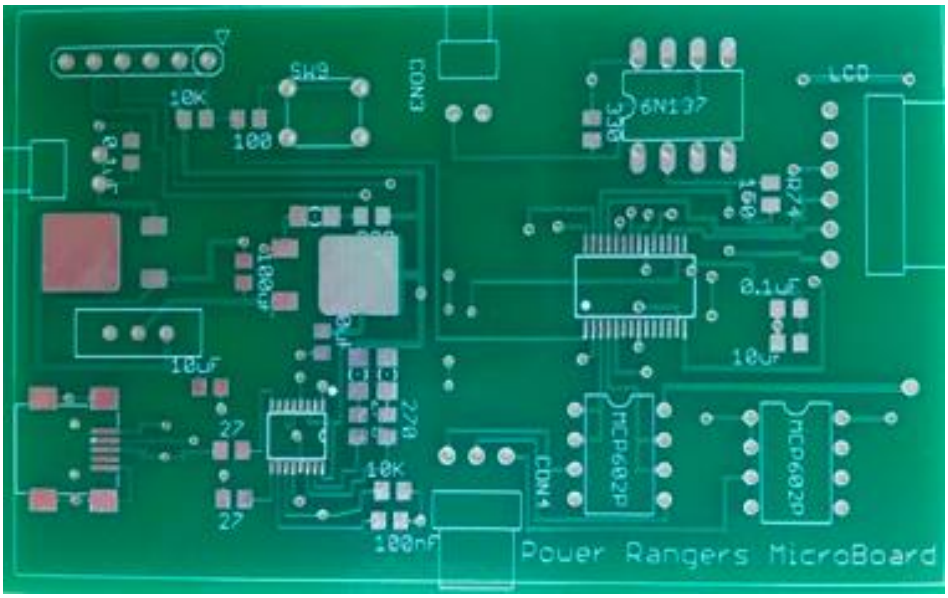
Appendix Figure 4: Converter Eagle Board



Appendix Figure 5: Converter PCB



Appendix Figure 6: Microcontroller Eagle Board



Appendix Figure 7: Microcontroller PCB

Appendix B:

Relevant Code

```
//Power Rangers
//Major Contributors: Anjola Ladenegan & Sarah Ritter
//Other Contributors: Evan Syers & Meaghan Hannon


//This file contains the reference curves in form of 3 dimensional arrays.
//This file also contains the power point tracking algorithm
#include <stdio.h>
#include <stdlib.h>
#include "SDlib.h"
#include "SDlib_LCD.h"
#include "configbitsrev2014vSarah.h"
#include <xc.h>
#include "Array_File.h"
#include <math.h>
#include <plib.h>




/*
 *
 */

void A2D_init(void);
int A2D_conv_voltage(void);
int A2D_conv_current(void);
int A2D_conv_battery(void);
double PowerPoint(double, double);
double DutyCycle (double, double);
void PWM_Generator(double);


int main(int argc, char** argv) {



    double voltage;
    double current1 = 4.75;
    double current2 = 4.75;
    double battery;
    double reqvoltage;
    double dutycycle_des;
    double dutycycle_act=0;
    double voltageread;
    double currentread1;
    double currentread2;
    double batteryread;
    double currentout;

    LCD_init();
    serial_init(57600UL);
    set_output_device(2);
```

```
    A2D_init();

    double counter1=0;
    double counter2=0;



    while (1){

       /* dutycycle_des = 0;

            while(fabs(dutycycle_des-dutycycle_act)>0.001){
                if (dutycycle_des>dutycycle_act){
                    dutycycle_act = dutycycle_act + 0.0005;
                }
                else if(dutycycle_des<=dutycycle_act){
                    dutycycle_act = dutycycle_act - 0.0005;
                }
                PWM_Generator(dutycycle_act);
            }

        PWM_Generator(dutycycle_des);

        batteryread = A2D_conv_battery();
        //battery = (double) batteryread*((3.3/1023)*7);


        if(battery>12){
            for (counter1=0; counter1<100; counter1++){
                /*LCD_display_on();
                LCD_clear();
                LCD_setpos(0,0);
                printf("Battery Voltage > 12. Not Charging");
                dutycycle_act = dutycycle_des;
                delay_ms(250);
            }
        }*/
        //else{


            //for(counter2=0; counter2<500; counter2++){

            battery = 12;

            voltageread = A2D_conv_voltage();
            voltage = (double) voltageread*((3.3/1023)*9.3);

            currentread1 = A2D_conv_current();
            current1 = (double) currentread1*((3.3/1023)*(4.03));

            currentread2 = A2D_conv_current();
            current2 = (double) currentread2*((3.3/1023)*(4.03));

                if (fabs(currentread1-currentread2)>.5){
                    if (currentread1>currentread2){
                        currentout = current1;
                    }
                    else if (currentread2>currentread1){
```

```
                    currentout = current2;
                    }
                }
                else if(fabs(currentread1-currentread2)<.5){
                    currentout = current1;


        reqvoltage=PowerPoint(voltage, currentout);

        dutycycle_des=DutyCycle(reqvoltage,battery);

        if(dutycycle_act!=0){
           while(fabs(dutycycle_des-dutycycle_act)>0.001){
                if (dutycycle_des>dutycycle_act){
                     dutycycle_act = dutycycle_act + 0.0005;
                }
                 else if(dutycycle_des<=dutycycle_act){
                     dutycycle_act = dutycycle_act - 0.0005;
                }
            PWM_Generator(dutycycle_act);
            }
        }

        //dutycycle_des = .5;
        PWM_Generator(dutycycle_des);

        dutycycle_act = dutycycle_des;

        LCD_display_on();
        LCD_clear();
        LCD_setpos(0,0);

        printf("Voltage : %.4f\r   Current: %.4f\r    Battery : %.4f\r   DutyRatio:
%.4f\r", voltage, currentout, battery, dutycycle_act);
        delay_ms(250);
        }



    }

    return (EXIT_SUCCESS);
}

void A2D_init(void){
//    AD1PCFG = 0; // This statement was giving us errors

    AD1CON1 = 0;
    AD1CON1bits.FORM = 4; // 32-bit unsigned Integer
    AD1CON2 = 0;
    AD1CON3 = 0;
    AD1CON3bits.ADCS = 3; // Set for the clock source
    AD1CON1bits.ON = 1;

}

 int A2D_conv_voltage(void){
```

```
    int data;
    AD1CHSbits.CH0SA = 4;
    AD1CON1bits.SAMP = 1; // Start Acquisition
    delay_us(1);
    AD1CON1bits.SAMP = 0; // Start Conversion

    while(!AD1CON1bits.DONE);
    data = ADC1BUF0; // Return Samples


    return (data);
}

 int A2D_conv_current(void){


    int data;
    AD1CHSbits.CH0SA = 5;
    AD1CON1bits.SAMP = 1; // Start Acquisition
    delay_us(1);
    AD1CON1bits.SAMP = 0; // Start Conversion

    while(!AD1CON1bits.DONE);
    data = ADC1BUF0; // Return Samples


    return (data);
 }


 int A2D_conv_battery(void){

         int data;
    AD1CHSbits.CH0SA = 6;
    AD1CON1bits.SAMP = 1; // Start Acquisition
    delay_us(1);
    AD1CON1bits.SAMP = 0; // Start Conversion

    while(!AD1CON1bits.DONE);
    data = ADC1BUF0; // Return Samples


    return (data);

 }

 double PowerPoint (double voltage, double current)
 {

    double incurr = current; //0.7;  //INCOMING CURRENT
    double involt = voltage; //20.5; //INCOMING VOLTAGE
    double max=0.0;
    double volt=0.0;
    double distarray[120]={0};
    int count=0;
    int location;
    int w=0;
```

```
    int position;
    int c=0;


    int i;
    int j;

    for (i = 0; i < 6; i++)
    {
        for( j=0; j<20; j++)
       {
            distarray[count]=sqrt(pow((Array1[i][j][0]-involt),2)+(pow((Array1[i][j][1]-
incurr),2)));

            //printf("distance is %f\n", distarray[count]);
            count=count+1;
        }
    }


    double min = distarray[0];
    for ( w = 1 ; w < 120 ; w++ )
    {
        if ( distarray[w] < min )
        {
           min = distarray[w];
           location = w+1;
        }
    }
        //printf("The closest is : %f\n",min );
        //printf("located at %d\n",location);



        if (location<=20)
        {
                position=location-1;
                    //printf("Interpolated Voltage is = %g \n", Array1[0][position][0]);
                    //printf("Interpolated Current is = %g \n", Array1[0][position][1]);
                    //printf("Interpolated Power is %f\n",Array1[0][position][2]);
                    //printf("This is on curve 1\n");
                    for (c = 0; c < 20; c++)
                    {
                        if(Array1[0][c][2]>max)

                            {
                                max=Array1[0][c][2];
                                volt=Array1[0][c][0];

                            }

                    }



                    //printf("The Maximum Power on that curve is %f\n", max);
                    //printf("The required voltage is %f\n",volt);
```

```
    }


    if(location>20 && location<=40)
    {
    position=(location-20)-1;

                //printf("Interpolated Voltage is = %g \n", Array1[1][position][0]);
                //printf("Interpolated Current is = %g \n", Array1[1][position][1]);
                //printf("Interpolated Power is %f\n",Array1[1][position][2]);
                //printf("This is on curve 2\n");

                for (c = 0; c < 20; c++)
                {
                    if(Array1[1][c][2]>max)

                        {
                            max=Array1[1][c][2];
                            volt=Array1[1][c][0];

                        }

                }



                //printf("The Maximum Power on that curve is %f\n", max);
                //printf("The required voltage is %f\n",volt);

    }


  if(location>40 && location<=60)
   {
    position=(location-40)-1;

                //printf("Interpolated Voltage is = %g \n", Array1[2][position][0]);
                //printf("Interpolated Current is = %g \n", Array1[2][position][1]);
                //printf("Interpolated Power is %f\n",Array1[2][position][2]);
                //printf("This is on curve 3\n");
                for (c = 0; c < 20; c++)
                {
                    if(Array1[2][c][2]>max)

                        {
                            max=Array1[2][c][2];
                            volt=Array1[2][c][0];

                        }

                }
```

```
                //printf("The Maximum Power on that curve is %f\n", max);
                //printf("The required voltage is %f\n",volt);

  }



if(location>60 && location<=80)
 {
  position=(location-60)-1;

                //printf("Interpolated Voltage is = %g \n", Array1[3][position][0]);
                //printf("Interpolated Current is = %g \n", Array1[3][position][1]);
                //printf("Interpolated Power is %f\n",Array1[3][position][2]);
                //printf("This is on curve 4\n");
                for (c = 0; c < 20; c++)
                {
                    if(Array1[3][c][2]>max)

                        {
                            max=Array1[3][c][2];
                            volt=Array1[3][c][0];

                        }

                }




                //printf("The Maximum Power on that curve is %f\n", max);
                //printf("The required voltage is %f\n",volt);

  }


if(location>80 && location<=100)
 {
  position=(location-80)-1;

                //printf("Interpolated Voltage is = %g \n", Array1[4][position][0]);
                //printf("Interpolated Current is = %g \n", Array1[4][position][1]);
                //printf("Interpolated Power is %f\n",Array1[4][position][2]);
                //printf("This is on curve 5\n");
                for (c = 0; c < 20; c++)
                {
                    if(Array1[4][c][2]>max)

                        {
                            max=Array1[4][c][2];
                            volt=Array1[4][c][0];

                        }

                }
```

```
                    //printf("The Maximum Power on that curve is %f\n", max);
                    //printf("The required voltage is %f\n",volt);

        }


       if(location>100 && location<=120)
        {
        position=(location-100)-1;

                    //printf("Interpolated Voltage is = %g \n", Array1[5][position][0]);
                    //printf("Interpolated Current is = %g \n", Array1[5][position][1]);
                    //printf("Interpolated Power is %f\n",Array1[5][position][2]);
                    //printf("This is on curve 6");
                    for (c = 0; c < 20; c++)
                    {
                        if(Array1[5][c][2]>max)

                            {
                                max=Array1[5][c][2];
                                volt=Array1[5][c][0];

                            }

                    }

                    //printf("The Maximum Power on that curve is %f\n", max);
                    //printf("The required voltage is %f\n",volt);
        }

        if (min>=10)
            {
                //printf("This point is not close enough to any curve\n");
                volt=0;
            }
                    return volt;

    }



double DutyCycle (double reqvoltage, double batteryvoltage)
 {
     double duty;

    /*if(batteryvoltage < 3.000 || reqvoltage < 10.00){
        duty = 0;
    }*/
    //else {
        duty = batteryvoltage/reqvoltage;
    //}

    return duty;
 }
```

```
void PWM_Generator(double dutycycle)
{
    //SYSTEMConfigPerformance(SYSCLK); // function that optimizes performance of PIC32
    double systemclock = 80000000;
    double pwm_freq = 100000;


    OC1CON = 0x0000; // Turn off the OC1 when performing the setup
    OC1R = 0x0064; // Initialize primary Compare register
    OC1RS = 0x0064; // // Initialize secondary Compare register
    OC1CON = 0x0006; // Configure standard PWM mode for output compare module 1

    // A write to PRy configures the PWM frequency
    // PR = [FPB / (PWM Frequency * TMR Prescale Value)] ? 1
    // : note the TMR Prescaler is 1 and is thus ignored
    PR2 = (systemclock / pwm_freq) - 1;

    // A write to OCxRS configures the duty cycle
    // : OCxRS / PRy = duty cycle
    OC1RS = (PR2 + 1) * (dutycycle);

    T2CONSET = 0x8000;      // Enable Timer2, prescaler 1:1
    OC1CONSET = 0x8000;     // Enable Output Compare Module 1


}
```