

MSU Solar Car Racing Maximum Power Point Tracker

Michigan State University
Senior Design Capstone
ECE 480, Team 8
Fall 2014

Project Sponsor

Michigan State University Solar Car

Project Facilitator

Bingsen Wang

Team Members

Daniel Chen
Yue Guo
Luis Kalaff
Jacob Mills
Brenton Sirowatka



Executive Summary

The MSU Solar Car Racing Team is a student run organization which creates solar powered vehicles. They create these vehicles and then race other universities at various solar car racing events across the country. The objectives for the team include performing well at competition and also enhancing the knowledge of its participants. The work which is required to make the car is split up among members and senior design group 8 was tasked with creating the maximum power point tracker.

The maximum power point tracker (MPPT) is a device inside the solar car which interfaces between the solar array and the battery of the vehicle. The MPPT is required because the low voltage of the solar array needs to be synchronized with the high voltage of the battery. The device will locate the voltage level which will provide maximum power for the solar car. The team previously bought a MPPT but now desire an MPPT that will be completely student designed and built. The final prototype built by our team will be expanded upon once the semester is over and eventually implemented into the MSU solar car.

There are three main components of the design. These components include a DC/DC booster, a microcontroller, and a final PCB which our design will be implemented on. The DC/DC booster will do the physical aspect of boosting the low voltage of the solar array while the microcontroller will control the booster in order to provide the vehicle with maximum power.

Acknowledgements

The following people played a critical role throughout the design process for the project. The team would like to give tremendous thanks to all of these people; we could not have done it without them.

Professor Bingsen Wang: Dr. Wang advised the team on the prototyping process and encouraging us to be resourceful.

Steve Zajac: Steve gave helpful advice on where to obtain free samples and helped clarify the team's needs.

Ian Grosh: Ian helped give advice on which software to use for simulation and help clarifying the Solar team's needs.

Scott O'Conner: Scott gave us helpful advice and pointed the team in the right direction by giving us background on MPPT systems and the solar car's current infrastructure.

Solar Car Racing Team: The Solar Car Racing team came up with project idea and funded our endeavor.

Roxanne Peacock: Roxanne ordered the parts for the team

Table of Contents:

Executive Summary	2
Acknowledgements	3
Chapter 1 - Introduction to Solar Vehicles	6-11
1.1 Solar Panels	6
1.2 Battery	9
1.3 Maximum Power Point Tracker	9
1.4 Applications of MPPT Implementations	10
1.4.1 Solar Farms	10
1.4.2 Solar Vehicles	10
1.4.3 Residential/Commercial Energy	11
1.4.4 Power Transmission (Fiber Optics)	11
Chapter 2	12-17
2.1 FAST Diagram	12
2.2 House of Quality (CCRs)	13
2.3 Feasibility & Decision Matrix	13
2.4 Budget	16
2.5 Gantt Chart	17
Chapter 3	18-42
3.1 Overview of Conceptual DC-DC Booster Converter	18
3.2 Capacitor	19
3.3 Inductor	22
3.4 Diode	24
3.5 Switch	25
3.5.1 Ideal Switch	25
3.5.2 MOSFET	26
3.6 MOSFET Driver	28
3.7 Ideal DC-DC Booster PSpice Simulation	30
3.8 Sensing	32
3.8.1 Voltage Sensing	32
3.8.2 Current Sensing	33
3.9 Microcontroller	35
3.9.1 Device Initialization	36
3.9.2 Analog to Digital Converter	36
3.9.2 Pulse Width Modulator	37
3.10 Algorithm	38
3.10.1 Perturb and Observe	39
3.11 MPPT CAD Prototype	40
Chapter 4	43-57
4.1 Testing - DC/DC Booster	43
4.2 Testing - MOSFET Driver	45
4.3 Testing - MOSFET, Driver, & DC-DC Booster	46
4.4 Testing - Light Bulb Load	48
4.5 Testing - C2000 MPPT P&O Algorithm	52
4.6 Testing - MSP430 MPPT P&O Algorithm	52
4.7 Testing - Current Sensor	54

4.8 Testing - MOSFET, Driver, Booster, Microcontroller, & Sensing (Entire Design)	55
Chapter 5	
5.1 Summary	58
5.2 Final Cost and Budget	59
5.3 Final Thoughts	60
5.4 Schedule	61
Appendix 1 - Technical Roles	62-66
Appendix 2 - Bibliography	67-68
Appendix 3 - Technical Attachments	69-81
A3.1 C2000 MPPT Code	70
A3.2 MSP430 MPPT Code	79

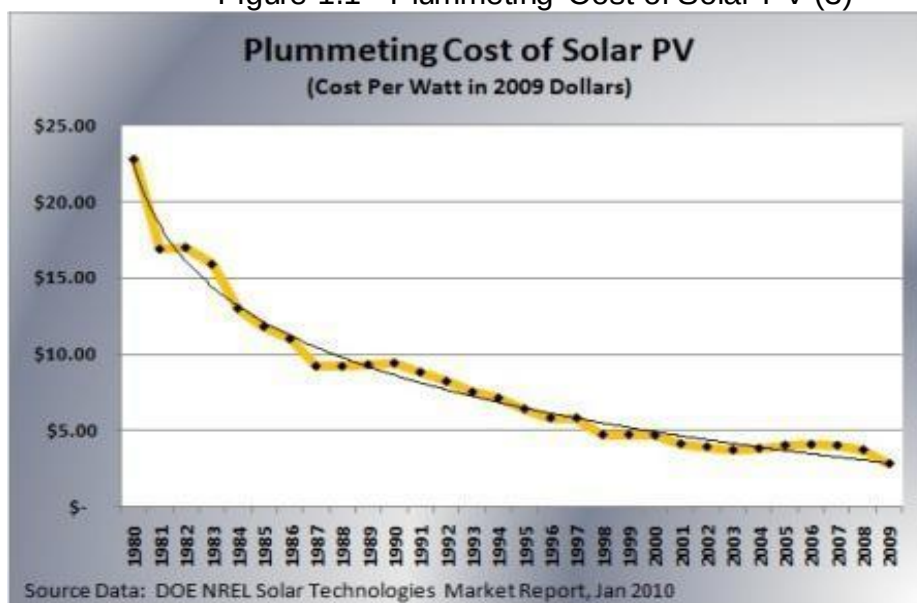
Chapter 1 - Introduction to Solar Vehicles

1.1 - Solar panels

The photovoltaic effect is described as the creation of voltage or current in a material when said material is exposed to sunlight. This effect was first observed in 1839 by Alexandre Bacquerel. This was the first step towards being able to harvest energy from the sun and then using it to power human technology. The first solar cell came about in the late 19th century and was built by Charles Fritts. Fritts was an American inventor who made his solar cell from selenium. He was able to achieve 1% efficiency with his design. Although this is an unimpressive number in today's highly technological world, Fritts' invention was crucial in the advancement of solar energy. The next major milestone in solar energy was when Bell labs created the first high powered cell with an efficiency of 6% (1). This cell was made of silicon, which is still used for most solar cells today. Modern solar cells now achieve much higher efficiency, sometimes reaching almost 45%.

The push by many for renewable energy has led to widespread usage of solar cells. They can be found in anything from homes, businesses, vehicles, satellites, etc. Modern solar cells are also very durable in that most companies today guarantee 80% power output for the first 25 years of use and usually replace the cells if they fall below the expected power output. Another positive trend in solar energy is that the average cost of a solar cell has steadily declined over the last 35 years (2). Figure 1.1 shows the pricing of solar cells per watt from 1980 to 2009. In 1980 the price per watt was upwards of \$23.00. By 2009 the price per watt had fallen well below \$5.00 and today in 2014 the price per watt is less than a dollar. This pricing decline is an encouraging sign for an industry that may be the most viable option when trying to replace non-renewable energy.

Figure 1.1 - Plummeting Cost of Solar PV (3)

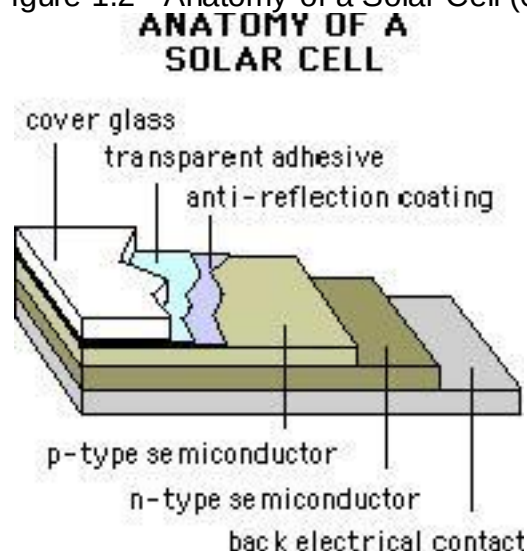


Solar cells utilize the photovoltaic effect in order to generate their electricity (4). What happens is that the electrons in the valence band of the material, usually silicon, absorb energy. These electrons become extremely excited and break free. They then diffuse into a different material and travel to the connected device. The configuration of solar cells is very important when implementing them into an electrical system. Solar cells add their voltage in series and add their current in parallel. Thus, giving an individual solar cell low open circuit voltage (approximately 0.5V) and a very high short circuit current (approximately 6A).

There are several different types of solar cells on the market today. Polycrystalline cells are most common because they are the most inexpensive to make. Although, there is a trade off when using these cells because there are much more efficient options available. The polycrystalline cells are made from several small silicon crystals as opposed to monocrystalline cells which are made from one large silicon crystal. Monocrystalline cells are more efficient but are more expensive to manufacture (5). There are several other types of solar cells available including multi-junction cells and organic cells but these designs are much less common. Multi-junction cells utilize several layers of semiconducting materials in order to increase the wavelength which can be absorbed by the cell. While organic cells are a product of improving molecular engineering practices. They are a relatively new product which uses a polymer solar cell and organic electronics for light absorption.

The typical anatomy of a solar cell consists of six layers. As shown in Figure 1.2 from top to bottom solar cells consist of a sheet of protective glass, an adhesive, an anti-reflection coating, a p-type semiconductor, a n-type semiconductor and then finally the back electrical contact.

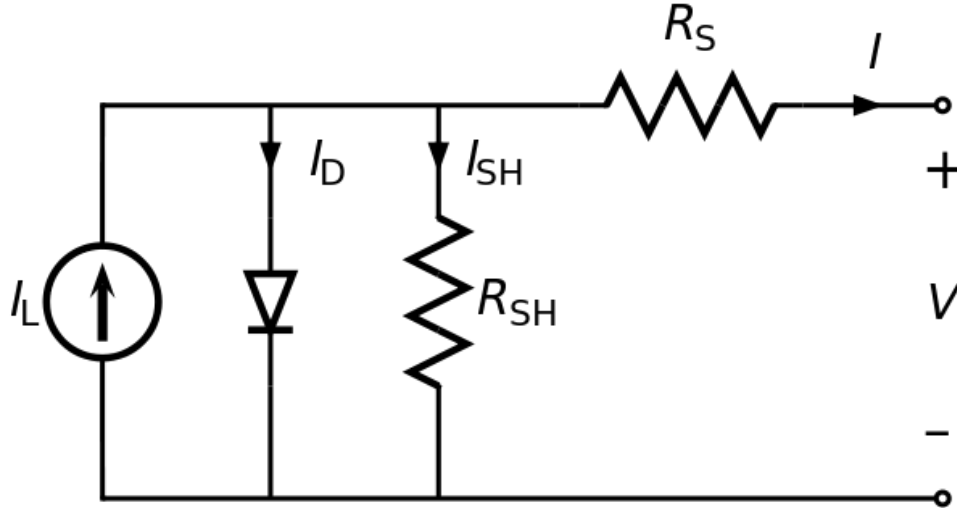
Figure 1.2 - Anatomy of a Solar Cell (6)



The purpose of having the two semiconductors is so that the electrons being diffused inside the cell have somewhere to travel to. The n-type silicon has an excess of electrons, while the p-type has an excess of holes. When the electrons break free, they recombine with the holes in the p-type silicon. The point in the cell when the p and n type silicon meet is called the p-n junction.

Understanding the electrical function of a solar cell can be made much easier by analyzing the schematic. The schematic for a typical solar cell is very basic and can be seen below in Figure 1.3.

Figure 1.3 - Solar Cell Schematic



This figure gives a much clearer picture of a cell's function. The current source shown is generated by the illumination of the sun. While the diode represents the simple p-n junction behavior. A series resistance and shunt resistance are added to the schematic because no solar cell is ideal. A schematic for a solar cell with just the current source and diode would be an ideal circuit for a cell. Several equations can be derived from the equivalent circuit in Figure 1.3. The characteristic equation for the current produced by the solar cell is shown below in Figure 1.4.

Equation 1.4 - Solar Cell Current

$$I = I_L - \underbrace{I_0 \left\{ \exp \left[\frac{q(V + IR_S)}{nkT} \right] - 1 \right\}}_{\text{Current Through Diode}} - \underbrace{\frac{V + IR_S}{R_{SH}}}_{\text{Shunt Current}}$$

In this equation I_L represents the photogenerated current. The current through the diode is then subtracted from that. The diode current is found by using the Shockley diode equation. Finally the shunt current is subtracted. This current is found simply by using Ohm's law with the voltage across the elements and the resistance of the shunt. Doing this calculation will give the user the total output current.

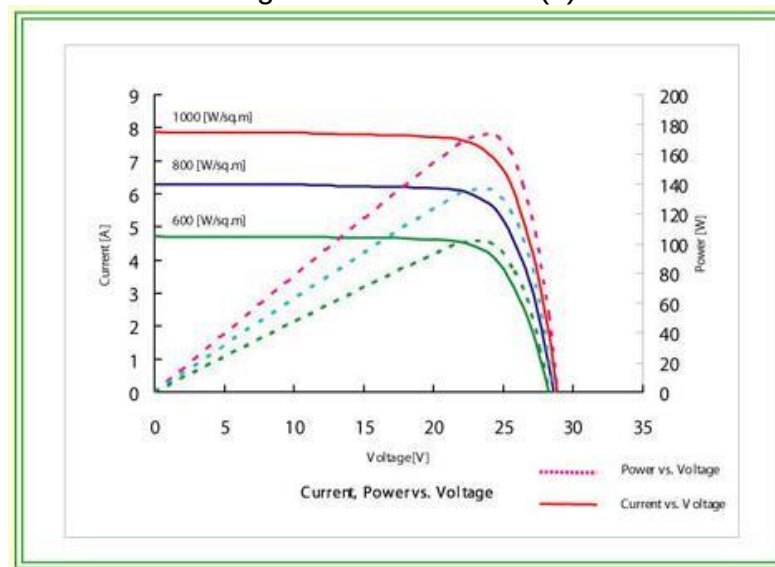
1.2 Battery

The battery of a solar car typically can allow the car to travel 250 miles on a full charge at a constant speed of 40 miles per hour. The cars travel at slow speeds because of the low efficiency of solar cells. As an increase in efficiency of the panels is made, the average speed of a solar car increases. One consideration as it pertains to the battery is that the user must include blocking diodes at the end of each panel used in the design. Otherwise, the battery itself could force current backwards through the array.

1.3 Maximum Power Tracker

A Maximum Power Point Tracker (MPPT) is a device which maximizes the power generated from photovoltaic (or solar) cells (7). It can also be recognized as an electronic circuit that links the solar array and the battery. This device is necessary because it matches the relatively low voltage of the solar array to the high voltage of the battery. Maximum power point trackers are usually digital devices which control the power by looking at the output of the solar panels and then comparing that output to the battery voltage (8). It then decides the optimum power output which will most effectively charge the battery. The maximum power point is found at the “knee” of what is called the I-V curve. The I-V curve is a graph which displays the illumination current versus the voltage being generated by the solar array. An example of this curve can be seen in Figure 1.5 below.

Figure 1.5 - I-V curve (9)



Solar cells have a very complex relationship between solar illumination, temperature, and resistance. This is what causes the nonlinear form of the I-V curve. In circuit theory, the approximate maximum power point could be calculated by finding the point at which the derivative of the I-V curve is equal and opposite the I/V ratio.

There are several components that must be implemented when creating a maximum power point tracker. The most important component probably being the DC/DC booster. The DC/DC booster boosts the low voltage of the array. The MPPT then uses voltage and current sensing so that the microcontroller can then use that information to calculate at what point the maximum power is being transferred.

1.4 Applications of MPPT Implementations

1.4.1 Solar Farms

Solar farms are large scale solar power generators which supply energy to the electricity grid. These farms supply power at a utility level unlike most solar applications which supply power on a smaller residential level. For a power station to be viable they must be created to supply a power of at least one megawatt. Although many solar farms can supply hundreds of megawatts. These farms are expensive to produce and are usually developed by privately owned, independent power companies.

Figure 1.6 - Solar Farm (11)



1.4.2 Solar Vehicles

Over the past few decades solar powered vehicles have made huge strides in improving performance but auto companies are still trying to overcome the many limitations solar vehicles face. One of these is limitations is power density. The power being generated from the solar array is heavily dependent on the size of the vehicle and area on the vehicle that can be exposed to light. Currently there are not any strictly solar powered vehicles available commercially but there is hope that solar power could help supplant the power used by modern electric vehicles which charge using the electrical grid. Another possible commercial application for solar vehicles is golf carts. Golf carts are fairly lightweight vehicles which spend most of their time in the sun. Maximum power point trackers in solar vehicles make sure that the battery is receiving maximum power when it is charging. This can improve the speed and driving distance of the vehicle.

Figure 1.7 - Solar Car



1.4.3 Residential/Commercial Energy

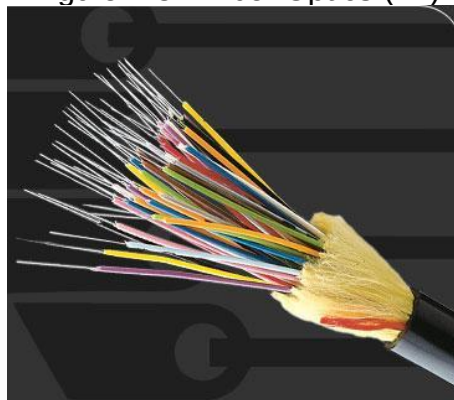
Just over the past decade the number of homes which utilize solar power has increased greatly. This increase in solar use has been helped by increased tax incentives from the federal and local governments, as well as improved efficiency and power production in current solar panels.

Installation of solar panels to a home today can greatly reduce a family's carbon footprint as well as save them up to \$84.00 per month on their electricity bill.

1.4.4 Power Transmission (Fiber Optics)

Maximum power point tracking can also be used in optical power transmission systems. Optical power transmission is a sufficient way of replacing copper wiring with fiber optic cables when a conventional power supply is challenging to implement. In optical power transmission power can be transmitted with light through an optical fiber. A light source, most likely a laser, generates light and then a photovoltaic cell converts the power back into electricity. This is a very efficient way of converting monochromatic light into electricity. When the photovoltaic cell converts the light energy back to electricity an MPPT is used to make sure the proper power is being sent to the device. This is very similar to our project just on a much smaller scale.

Figure 1.8 - Fiber Optics (11)

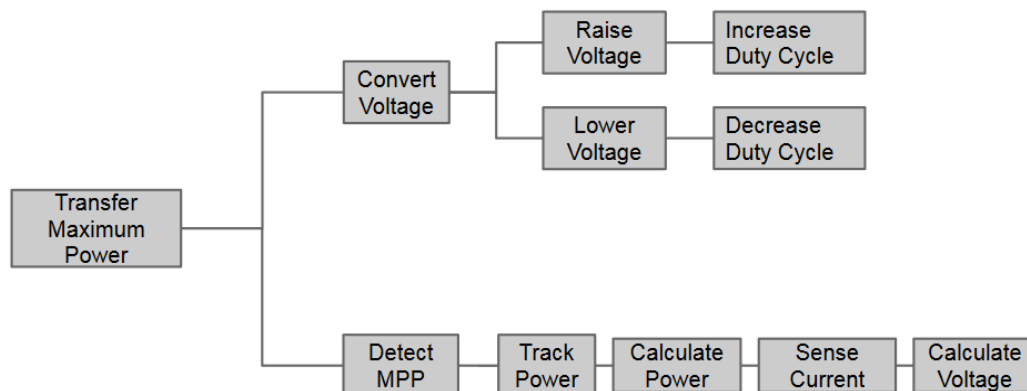


Chapter 2 - Exploring the Solution and Selecting a Specific Approach

2.1 FAST Diagram

The following FAST diagram displays how the team intends to implement the maximum power point tracker. The leftmost part of the diagram highlights what the overall design needs to accomplish which is to transfer maximum power to the solar vehicle's battery. As the diagram moves further to the right the diagram gives a more detailed view of how the MPPT will be created. The design can be separated into two major components, converting (or changing) the voltage and detecting at what point the maximum power is located. In order to convert the voltage the MPPT will raise and lower the voltage value. This will be accomplished by increasing and decreasing the duty cycle. The second main function of the design is to actually detect the maximum power point. This point is found by tracking the power. That power is monitored by first sensing the current and voltage values. Once this is accomplished those values are used in order to calculate the power at that point.

Figure 2.1 - Fast Diagram



One of the main components of the design is the electrical portion. This part of the project is composed of a power supply going into the booster as well as an additional power supply that will power the MOSFET driver and current sensor. Voltage and current sensors before and after the booster will give the readings necessary in order to calculate the power at both of those points.

The second component of the design is the programming portion. The programming of the microcontroller is crucial to the design because it will use the readings from the sensors in order to tell the MOSFET driver the approximate duty cycle for which the device is calculating maximum power. The only other portion of the design is to create a PCB for the final design and use a box for protection when dealing with voltages above 50 V. This was not necessary during most of the testing because the team tried to test the components on a smaller scale first.

2.2 House of Quality (CCRs)

The team created a House of Quality in order to define the needs of the sponsor. This House of Quality diagram can be seen in Appendix 3. The requirements listed for the project are shown below and were provided by MSU Solar Car.

- 3 Stages
 - Prototype of DC/DC booster ((20-60V), 6A input, ~100v)
 - Program a Microcontroller to track and adjust the Maximum Power Point
 - Create PCB combining these two requirements.
- Design Requirements
 - 95% Efficiency
 - Size preferred to be less than the size of a credit card
 - Weight should be no more than 100g

After the first meeting with the sponsor the requirements were modified slightly. MSU solar car made it clear that the most important requirement would be to keep the efficiency above 95%. The other two requirements (size and weight) were not as important to MSU solar car so as long as the final design met the efficiency requirement. Taking all of the information provided by our sponsor into consideration the team decided that the critical customer requirements were to implement the 3 stages provided and to keep the design at or above 95% efficiency.

There were also extra requirements which were specified by MSU solar car that could be added to the design if there was enough time to implement them. One of these requirements being that the MPPT would have CAN functionality. This is something the solar car team will try to implement once the prototype is complete. Getting this done would provide the solar car team with less work to do in the future. Another extra feature which could be implemented into the design would be to create multiple channels. The sponsor said that they would only require a one-channel prototype. Although the previous MPPT used by the MSU solar car team had 4 channels. If the team were able to complete a one channel design it would be beneficial to expand on that and create a 4 channel MPPT so that it would be a fully functioning prototype.

2.3 Feasibility & Decision Matrix

Using the critical customer requirements the team separated the project into 5 stages. The stages include component design and selection, separate component implementation, low voltage prototype, high voltage prototype, and complete prototype with extra features.

For the first stage (component design and selection) research is to be completed as well as the ordering of all parts. This is important for the functionality of the design as well as the safety. Since the device deals with high voltages, a miscalculation in this stage could be disastrous when testing the design in lab.

The second stage is to build and test the components separately to ensure they are working as expected. This will include the testing of the DC/DC booster, gate driver, current sensor, voltage sensor, and microcontroller.

The third stage in the design process is to complete a functional prototype of the design. All of the individually tested parts will be brought together and the team will then test the design using low voltage. Once the testing is complete improvements will be made as needed.

During the fourth stage of the design process a second prototype will be built that will be able to withstand much higher voltages and give a good depiction of how the final product functions. This stage is important to the sponsor because it will ensure that the prototype can be implemented onto the MSU solar car. Safety needs to be a priority in this stage because the team will be dealing with high voltages.

The final stage of the design will be to complete a working prototype that would be able to go into the solar car and function properly. It would also have all of the extra features operational. Using these design stages as a guide a feasibility matrix was created. The matrix gave the team a better understanding on how complex each design stage was going to be to complete. The matrix can be seen in Figure 2.2 below.

Figure 2.2 - House of quality

Design Stage	Cost (5-High, 1-Low)	Complexity (5-Easy, 1-Difficult)	Time (5-Least, 1-Most)	Mean Feasibility
1 - Component design and selection	5	5	5	5
2 - Initial Component Implementation	4	5	5	4.67
3 - Low Voltage Prototype	4	3	3	3.33
4 - High Voltage Prototype	3	3	3	3
5 - Completed Prototype with Extra Features	2	2	2	2

Initially the minimum limit for completion was set at design stage four. Stage five would only be completed time permitting. Unfortunately the team was only able to get through design stage three. Problems were encountered with the current sensing capabilities. This stalled the completion of stage 3. The problems with the design became apparent in this stage. The team was able to test some of the components at a higher voltage but never an entire prototype. Proposed solutions will be explained in detail later on in the report.

Two more feasibility matrices were created which focused more on the selection of the microcontroller and which method the controller would use to program the tracker. While researching the team found that there are several methods for programming a MPPT. Eight methods have been used in the past but the matrix was created using only four of these methods. This was done because several of the methods had little documentation, limited success, and high cost. The feasibility matrix for the selection of

the programming method can be seen in Figure 2.3 below. The perturb and observe method was chosen because it proved to be the most feasible method. Also the MSU solar car team previously used this method in order to program their MPPT. More information on these control methods can be found in Chapter 3.

Figure 2.3 Feasibility Matrix of Control Method

Control Method	Efficiency	Simplicity	Tracking ability	Feasibility
Perturb and Observe	5	5	4	14
Incremental Conductance	3	1	5	9
Current Sweep	3	2	3	8
Constant Voltage	2	3	1	6

The feasibility matrix below (Figure 2.4) compares the MSP430 microcontroller to the TI Piccolo microcontroller. The MSP430 would be easier to implement due to past experiences but the piccolo microcontroller proved to be the better choice because the features are much better suited for the design.

Figure 2.4 Selection Matrix of Microcontroller

	TMS320F28027	MSP430G2553
Clock Speed	60MHz	16MHz
ADC	12 bits	10 bits
# of ADC Channels	7/13	8
Operate Temperature(C)	-40 to 105	-40 to 85
Interface	UART/I2C/SPI	UART
Cost	3.05/1ku	0.90/1ku
Rating	5	1

2.4 Budget

This project was funded with a budget of \$500.00. Over the course of the project tracking the products was crucial so that the team did not go over the provided budget. The current design is well under the cap and can be seen in the chart below.

Figure 2.6 - Budget

Items	Vendor	Amount	Order Number	Arriving Date	Sample?	Cost(\$)	Budget Left(\$)
Toroid	Dexter Magnetics Technologies	2	716954	24-Sep	Yes	0	500
TI C2000 piccolo lachpad	Mouser	1	36140895	30-Sep	No	17.08	482.92
Capacitors	Digi-Key	2	47218057	9-Oct	No	8.66	474.26
Diodes	Digi-Key	2	47218057	9-Oct	No	26.94	447.32
MOSFET Driver	Digi-Key	5	47340754	21-Oct	No	11.9	435.42
Heat Sinks	SparkFun Electronics	4	47600004	10-Nov	No	6.24	429.18
Half Bridge Driver	Digi-Key	4	47600004	11-Nov	No	8.84	420.34
High Side Current Sensor	Digi-Key	4	47600004	11-Nov	No	17.56	402.78
Op-amp for sensor	Digi-Key	4	47600004	11-Nov	No	5.24	394.54
Diode	Digi-Key	4	47600004	11-Nov	No	0.72	396.82
MOSFET	Digi-Key	8	47600004	11-Nov	No	14.88	381.94
Capacitor	Digi-Key	6	47600004	11-Nov	No	8.82	373.12
TI-C2000 PicoLo Launchpad	Newark	2	25870140	20-Nov	No	34.16	339.12
					Total Amount Spent = 160.88		

The data above is only pertinent to this project. I think with what the team now knows there are certain things that could have been cut out of the budget in order to reduce cost. On the other hand the team did not complete the project and certain components may need to be added to the budget in order for the design to work sufficiently. Another factor to consider is that some of the components used were given to the team for free. Most of the free samples received are not listed in the budget above.

2.5 Gantt Chart

As shown in Figure 2.7 below we outlined the entire timeline of our project using a Gantt chart. The three main portions of the schedule are design, prototyping, and refinement. During the design portion the team ordered parts, ran simulations, and designed schematics regarding how the MPPT was supposed to function. During prototyping the team built prototypes to try and implement the design. During refinement the team was supposed to improve upon the prototype and make sure all components in the design were working.

Figure 2.7 - Gantt Chart

Task Mode	Task Name	Duration	Start	Finish	Predecessors
1	Design	18 days	Mon 9/15/14	Wed 10/8/14	
2	DC-DC boost converter	10 days	Mon 9/15/14	Fri 9/26/14	
3	Parameter Values Selection	5 days	Mon 9/15/14	Fri 9/19/14	
4	Research on Boost Converter	2 days	Mon 9/22/14	Tue 9/23/14	3
5	Efficiency calculations	3 days	Wed 9/24/14	Fri 9/26/14	4
6	Microcontroller	2 days	Mon 9/29/14	Tue 9/30/14	
7	Choose Microcontroller	1 day	Mon 9/29/14	Mon 9/29/14	5
8	Study of Microcontroller	1 day	Tue 9/30/14	Tue 9/30/14	7
9	PCB layout	6 days	Wed 10/1/14	Wed 10/8/14	8
10	Cadence Simulation	6 days	Wed 10/1/14	Wed 10/8/14	
11	Final Propoasal	0 days	Wed 10/8/14	Wed 10/8/14	9
12	Simulation	12 days	Thu 10/9/14	Fri 10/24/14	11
13	DC-DC Boost Converter	12 days	Thu 10/9/14	Fri 10/24/14	
14	Labview Simulation efficiency	5 days	Thu 10/9/14	Wed 10/15/14	
15	Pspice simulation	7 days	Thu 10/16/14	Fri 10/24/14	14
16	Prepare Oral Presentation	3 days	Thu 10/9/14	Mon 10/13/14	11
17	Oral Presentation	0 days	Wed 10/15/14	Wed 10/15/14	16
18	Prepare Progress Report	4 days	Wed 10/15/14	Mon 10/20/14	17
19	Progress report	0 days	Wed 10/22/14	Wed 10/22/14	18
20	Prototype	22 days	Sat 10/25/14	Mon 11/24/14	12
21	DC-DC boost converter	3 days	Sat 10/25/14	Tue 10/28/14	
22	Low Voltage	2 days	Sat 10/25/14	Mon 10/27/14	12
23	High Voltage	1 day	Tue 10/28/14	Tue 10/28/14	22
24	First Demo	0 days	Tue 10/28/14	Tue 10/28/14	23
25	Gate Driver Solo Test	4 days	Wed 10/29/14	Mon 11/3/14	24
26	High-Speed Low-Side MOSFET	2 days	Wed 10/29/14	Thu 10/30/14	
27	Half Bridge Driver	2 days	Fri 10/31/14	Mon 11/3/14	26
28	Current Sensor Solo Testing	4 days	Tue 11/4/14	Fri 11/7/14	25
29	Microcontroller Solo Test	4 days	Mon 11/10/14	Thu 11/13/14	28
30	MPPT test	7 days	Fri 11/14/14	Mon 11/24/14	29
31	Refinement	7 days	Tue 11/25/14	Wed 12/3/14	30
32	Parameter Optimization	5 days	Tue 11/25/14	Mon 12/1/14	
33	MPPT sensitivity test	2 days	Tue 12/2/14	Wed 12/3/14	32
34	Desgin Issues	0 days	Wed 11/5/14	Wed 11/5/14	
35	Progress report	0 days	Wed 11/5/14	Wed 11/5/14	34
36	Second Demo	0 days	Wed 11/26/14	Wed 11/26/14	30
37	Poster	0 days	Sun 9/14/14	Sun 9/14/14	
38	Final Report	0 days	Wed 12/3/14	Wed 12/3/14	31
39	Project Revision	1 day	Thu 12/4/14	Thu 12/4/14	38
40	Design Day	0 days	Fri 12/5/14	Fri 12/5/14	39

Chapter 3 – Technical Description of Work Performed.

3.1 Overview of Conceptual DC-DC Boost Converter

A DC-DC Boost converter is composed of a switch, a transistor, two capacitors and an inductor. The simplest explanation which can be given for a DC/DC booster is that it is a circuit which boosts a low voltage into a much higher voltage. An overview of the conceptual design is shown in Figure 3.1.

Figure 3.1 - Schematic of DC-DC Boost Converter

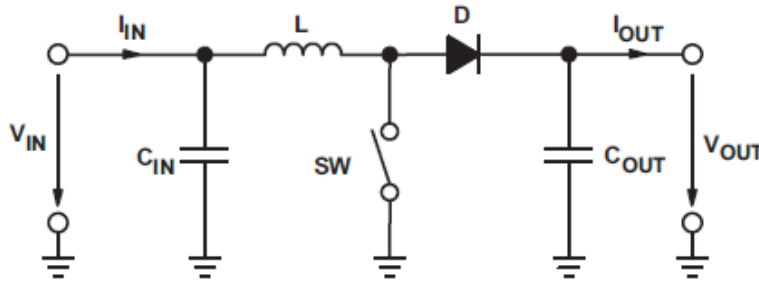


Figure 3.2

On state

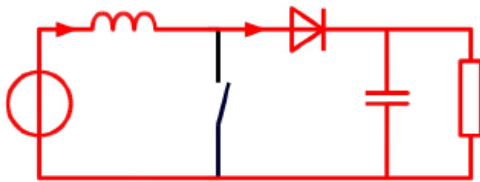
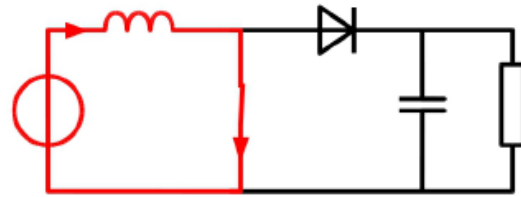


Figure 3.3

Off state



The tendency of the inductor to resist current changes, by creating and destroying a magnetic field, is the key factor in a DC-DC Boost converter. Figure 3.3 shows the Boost converter when the switch is closed. In this scenario, current flows in the clockwise direction through the inductor which stores energy by creating a magnetic field. After the switch is opened, as in Figure 3.2, current is reduced because of the higher impedance seen at this state. Then the magnetic field created in the on state is destroyed to maintain the current flow towards the load. Therefore the output voltage becomes higher than the input voltage.

There are four main parameters that need to be established in order to design a DC-DC Boost converter: the minimum input voltage, the maximum current, the maximum input voltage and the desired voltage output. The range of the input voltage was set by the MSU Solar Car Team to be between 20V and 60V. The maximum current output is equal to maximum input current. In this case the maximum current is 6 A because that is the highest amount of current which can be provided by the solar array. The output voltage is set by the duty cycle of the switch in the DC/DC boost converter. The duty cycle is calculated from the minimum input voltage, the desired

output voltage, and the efficiency, which in this case is 95%. The equation for the duty cycle can be seen in Equation 3.4 below.

Equation 3.4 - Duty Cycle (12)

$$D = 1 - \frac{V_{in(min)} \times \eta}{V_{out}}$$

The duty cycle is a critical component for the MPPT. Once the maximum power point is calculated by the microcontroller, the duty cycle is then set to the value which will provide the booster with appropriate voltage for maximum power transfer.

3.2 Capacitors

Theory

In the conceptual design, there is a capacitor for the input voltage and a capacitor for the output voltage. While the inductor is being charged, the only element on the circuit connected to the inductor is the output capacitor. So the output capacitor needs to be able to handle the output voltage. In addition there will be some ripple output due to the discharge of the output capacitor and the ESR (effective series resistance) in the capacitor. The additional output voltage ripple caused by the capacitor can be calculated with the ESR, the maximum output current, the duty cycle, and the inductor ripple current. The equation for output voltage ripple is shown in Equation 3.5 below.

Equation 3.5 - Output Voltage Ripple (12)

$$\Delta V_{out(ESR)} = ESR \times \left(\frac{I_{out(max)}}{1-D} + \frac{\Delta I_L}{2} \right)$$

There is minimum value for the output capacitance of a DC-DC Boost converter. If this minimum capacitance is not used, the result could cause serious damage to the capacitor and possibly an explosion. To reduce ripple voltage, the team decided to go with a value much higher than the minimum output capacitance. The minimum capacitance can be computed from the maximum output current, the minimum switching frequency of the converter, the duty cycle, and the desired output voltage ripple. The equation for minimum output capacitance can be seen below.

Equation 3.6 - Output Capacitance (12)

$$C_{out(min)} = \frac{I_{out(max)} \times D}{f_s \times \Delta V_{out}}$$

Also, by the definition of ESR, a capacitor has a resistance in series associated with it and therefore there is power dissipation in the capacitors. The power losses in the capacitor can be expressed using the switching frequency, the capacitance, the voltage, and the dissipation factor. This last parameter can be found in the datasheet of the given capacitor. The equation for power losses within a capacitor can be seen in

Equation 3.7. Monitoring the power losses in our design was crucial for this project. Our main requirement is efficiency so any loss in power needs to be noted.

Equation 3.7 - Power Loss in a Capacitor (13)

$$Power\ Losses = (i^2) \times (ESR)$$

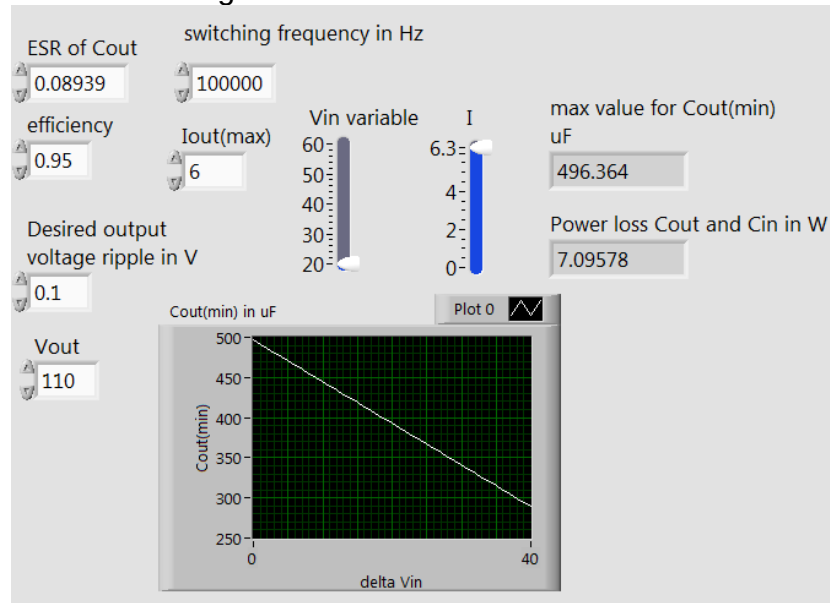
The most important parameter when choosing a capacitor for a DC-DC Boost converter, is its voltage rating. The voltage rating has to be bigger than the expected voltage across the capacitor. In this project the voltage rating needs to be much higher than the output voltage.

In a simple DC-DC Boost converter, the input capacitor is not necessarily needed. However, in order for the design to be efficient, a capacitor is used to stabilize the input voltage due to the peak current requirement of a switching power supply. This capacitor helps increase the overall system efficiency and helps reduce the current peaks drawn from the input supply and noise injection. The capacitance value is determined by the source impedance of the input supply. The team decided to make the value of the input capacitor the same as the one for the output capacitor. In addition, the power loss by the input and the output capacitors are equivalent.

Even though the ultimate goal is to maximize the efficiency of the DC-DC Boost converter, the power losses in both input and output capacitors is an unfortunate necessity.

In the early stages of the brainstorming process, a LabVIEW project was made to determine the most suitable component values of the DC-DC Boost Converter, which could meet the requirements set by the MSU Solar Car Team. The LabVIEW simulation used the equations provided earlier in this report as well as several other equations in order to give an accurate calculation of minimum capacitance. The team chose a ripple voltage of 100 mV for the capacitor. The voltage could fluctuate between 20V up to 60V. In the simulation we set the current value to 6 A because that is the highest current the solar panel can provide. As the voltage is increased in the simulation the required output capacitance decreases. By setting the simulation to its maximum current we could then determine the worst case scenario our device may see. The minimum output capacitance calculated for this simulation was $496.364\mu F$. This value corresponds to a voltage of 20V and the maximum current pulled from the solar cell. Figure 3.8 shows the LabVIEW simulation made. Under these conditions the power dissipation for both the input and output capacitance was found to be 7.095 W.

Figure 3.8 - LabVIEW Simulation



Component Selection

To reduce power losses, it was decided to use two capacitors in parallel for both the input and output capacitance. This reduced the equivalent ESR and therefore lowered the power dissipation.

Two different kinds of capacitors were used. One of them was a EKXG201ELL101ML20S $100\mu F$ electrolytic capacitor from Digi-Key. This capacitor can withstand up to 200V so it can tolerate the 110V output voltage expected. One of the other advantages of this capacitor is the low ESR of $270m\Omega$. Also, the operating temperature of the product goes up to $85^{\circ}C$ which should tolerate the temperature inside the solar car. An image of the capacitor used is shown in Figure 3.9.

Figure 3.9 - EKXG201ELL101ML20S Capacitor



The other type of capacitor bought for the project was a ESK477M200AN3 $470\mu F$ capacitor from Kemet. This product can also withstand up to 200V and its operating temperature is also set to be at $85^{\circ}C$. Its ESR is around $130m\Omega$. An image of this capacitor is shown in Figure 3.10.

Figure 3.10 - ESK477M200AN3 Capacitor



The two capacitors in parallel form an equivalent capacitance of $570\mu F$ which is much higher than the minimum output capacitance of $496.364\mu F$ calculated in the LabVIEW simulation. Also the equivalent ESR consists of a $270m\Omega$ resistance in parallel with a $130m\Omega$ resistance giving an ESR of $87.75m\Omega$. This parallel combination will cut the power dissipation in half.

3.3 Inductor

Theory

An inductor in a DC-DC Boost converter has to store energy. Inductance occurs due to a magnetic field that forms around a current-carrying conductor. Magnetic flux proportional to the current is then created. A change in the current results in a change in the magnetic flux which generates an electromotive force that acts against the change. Inductance is a measure of the amount of this force (EMF) generated for each unit change in current. Parameters such as the material wrapped around the inductance, the type of conductor, number of windings or turns, as well as the size of each turn, have an influence in the inductance.

For our design, there are a few parameters that needed to be accounted for in order to select a suitable inductor. The inductance need in a DC-DC Boost converter can be estimated by Equation 3.11.

Equation 3.11 - Required Inductance (12)

$$L = \frac{V_{in} \times (V_{out} - V_{in})}{\Delta I_L \times f_s \times V_{out}}$$

where V_{in} is the input voltage, f_s the minimum switching frequency, V_{out} the output voltage and ΔI_L the estimated inductor ripple current. A good estimation for the inductor ripple current is 20% to 40% of the output current. So it is defined by Equation 3.12.

Equation 3.12 - Inductor Current Ripple

$$\Delta I_L = (0.2 \text{ to } 0.4) \times I_{OUT(max)} \times \frac{V_{OUT}}{V_{IN}}$$

where $I_{OUT(max)}$ is the maximum output current.

An inductor was designed in the lab using a free toroid sample from Dexter Magnetics Laboratories and enamel wire from the ECE Shop. The theoretical value of inductance can be calculated by the following formula:

Equation 3.13 - Induction

$$L \approx \frac{\mu N^2 A}{2\pi r}$$

A = cross-sectional area
 r = toroid radius to centerline

Where μ is the relative permittivity, N is the number of turns, A is the cross-sectional area of the toroid, and r is the toroid radius to centerline. There are power losses in the inductor due to eddy currents, resistivity of the wire and capacitance between the wires.

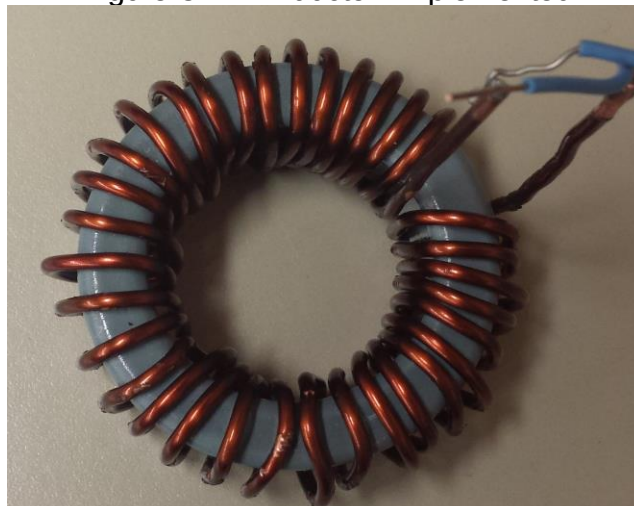
Component Selection

According to the Luis's LabVIEW simulation calculations, the minimum value of the inductance should be somewhere around 6.6mH, which is a relatively large inductance for an inductor. Since the volume of the MPPT is limited in size, the cross-sectional area (A) needed to be taken into consideration. In order to increase the inductance, an increase in the number of turns of wire (N) and the relative permittivity (μ) is required. The user could also decrease the toroid radius.

Another design constraint to account for is that the ESR of the inductor should be limited to a small value so there would be little loss. This can be controlled by increasing the diameter of the wire used to make the coil.

After these considerations, the team decided to choose the toroid currently being used, which has a relative permittivity of 5,000, radius of 2.0725cm, and cross sectional area of 1.18cm^2 . When there are 32 turns of wire wrapped around the core, the theoretical inductance is 6.58mH, which was fairly close to the desired inductance. A LCR meter in the lab allowed the team to measure the constructed inductor, the resulting inductance turned out to be 6.9mH. Since the inductor has gage 10 enameled wires, the mechanical property of wires are stronger and this made the cross-sectional area actually much bigger than the area of toroid. A picture of the inductor made is shown in Figure 3.14.

Figure 3.14 - Inductor implemented



3.4 Diode

Theory

A diode in a DC-DC Boost converter is used to prevent the output current from flowing back to the inductor (12). It should be able to handle the boost converter operating at peak voltage and current as well as be able to dissipate the least amount of power possible.

In terms of power losses, Schottky diodes are the most suitable option in these circumstances because of their low voltage forward and their very fast switching capability. Power dissipation in the diode can be described by:

Equation 3.15 - Power dissipation in the diode

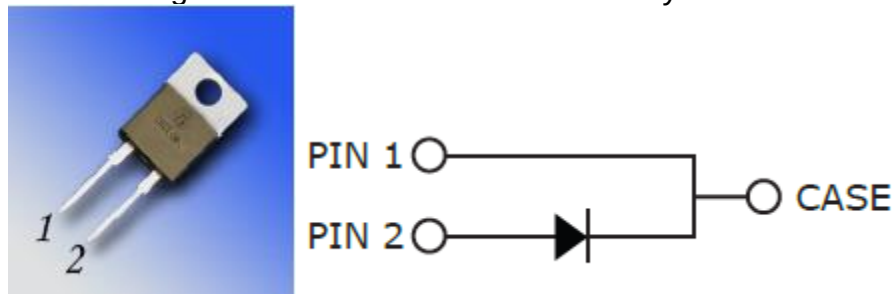
$$P_D = I_F \times V_F$$

where I_F is the average forward current of the rectifier diode and V_F is the forward voltage of the rectifier diode. These diodes have a much higher peak current rating than average rating meaning that the high peak current in the system is not a problem. Therefore the Schottky diode needs to have a forward current rating equal to that of the maximum output current.

Equation 3.16 - Forward Current

$$I_F = I_{OUT(max)}$$

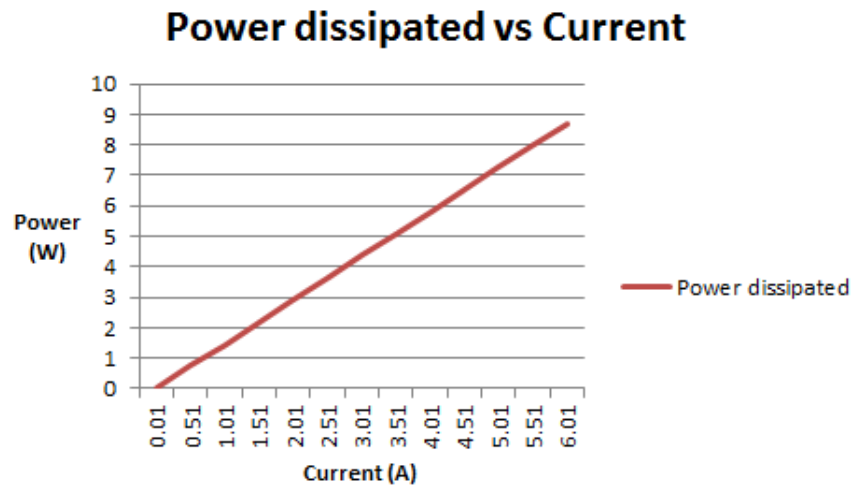
Figure 3.17 - Silicon Carbide Schottky Diode



Component Selection

The diode chosen for this project was a CVFD20065A Silicon Carbide Schottky bought from CREE. A picture and block diagram of the component is shown in Figure 3.17. Its forward current is 26A which is large enough to handle the maximum current. The diode also has a max operating temperature of 175°C and a forward voltage ranging from 1.35 to 1.65 depending on the temperature through the diode. The power dissipation of the diode implemented in the DC-DC Boost converter was calculated in Figure 3.18. At 6A, the maximum current, the power dissipated in the diode is equal to 8.71W.

Figure 3.18 - Power Dissipation vs Input Current



3.5 Switch

3.5.1 Ideal Switch

Theory

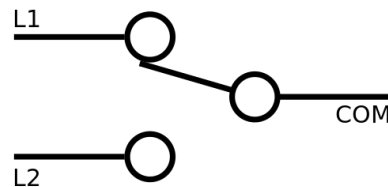
An electrical switch is a representation of a device which can connect and disconnect conductors. A switch is made up of poles and throws. A pole is the number of conductors, such as a copper wire, which can be controlled. A throw is the number of paths that can be connected. Figure 3.19 shows a few examples of switch symbols. In the single pole, single throw, the connection is either on (connected), or off (disconnected). In the single pole, double throw configuration, the “COM” wire is only ever connected to either L1 or L2, but never at the same time. The double pole, single throw configuration is like two separate single pole, single throw circuits, but are controlled by the same throw. Either the top two and the top bottom are connected or they are both disconnected.

Figure 3.19 - Switching Methods

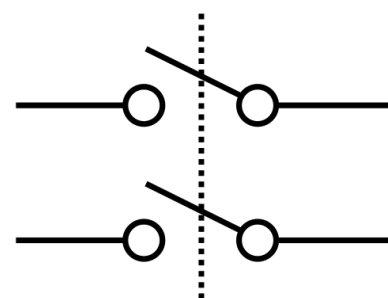
Single Pole, Single Throw



Single Pole, Double Throw



Double Pole, Single Throw



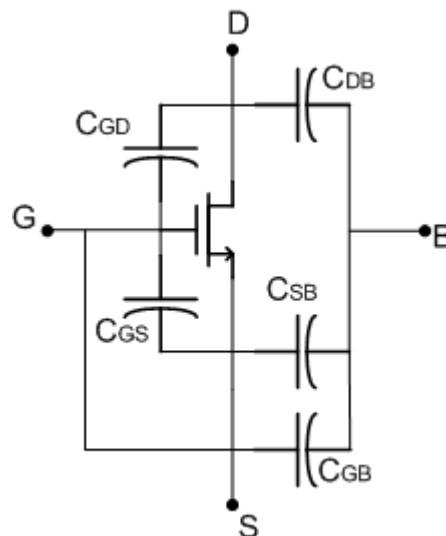
An ideal switch allows circuits with switches to be easily simulated. Naturally, the circuit performance will deviate from the actual circuit, but can provide in some cases, a realistic performance. An ideal switch was used in the PSpice simulations as discussed in the Ideal DC-DC Booster PSpice Simulation section.

3.5.2 MOSFET

Theory

The team decided to use a MOSFET in the design. There are two sets of requirements needed to choose a MOSFET for a given circuit: operating requirements, which ensure proper operation, and performance requirements, which reduce device losses as much as possible. Let's discuss the operating requirements first. The MOSFET needs to be able to handle the peak current of the circuit. It is important to keep in mind that the current rating decreases with increasing temperature. The Drain-Source voltage of the MOSFET has to handle the maximum voltage supplied. Also this voltage rating varies with the temperature. The threshold voltage of the MOSFET must be less than both the input voltage and the maximum gate-source voltage specified in the datasheet. The MOSFET needs to operate within its SOA (Safe Operating Area) (14). This is where the performance requirements come into play. The SOA is defined by the junction temperature, the breakdown voltage, and the maximum drain current. In order for the MOSFET to work properly, losses have to be minimal for a specific thermal junction. There are two types of losses: conduction and switching losses.

Figure 3.20 - Schematic of MOSFET with internal capacitances



A MOSFET has various internal capacitors that play a large part in the switching operations. A schematic of the MOSFET with its capacitances is shown in Figure 3.20

The values of these capacitors can be found by looking at the datasheet and using the following equations:

Equation 3.21 - Capacitances values from Datasheet

$$\begin{aligned}C_{gd} &= C_{rss} \\C_{gs} &= C_{iss} - C_{rss} \\C_{ds} &= C_{oss} - C_{rss}\end{aligned}$$

Another parameter to look for is the total gate charge. Along with the switching frequency this will give you the current needed to charge the gate capacitance of the MOSFET. The higher the gate charge, the higher the dissipation losses. Switching losses are also affected by the drain-source voltage, the drain current, the switching frequency, and the rise and fall times.

Equation 3.22 - Switching Power Losses

$$P_{SW} = \frac{1}{2} \times V_{DS} \times I_D \times f_{sw} \times (t_{rise} + t_{fall})$$

Conduction losses are the second type of loss. The MOSFET has a small drain-source on-resistance. This is one of the most crucial parameters. It has to be as low as possible. The conduction losses mainly depend on the drain-source on-resistance, the input current, and the duty cycle.

Equation 3.23 - Conduction Power Losses

$$P_C = R_{DS(on)} \times D \times I_{in}^2$$

When dealing with high temperature the value of the resistance increases by a factor determined by the current temperature and a temperature, $T_{J(hot)}$, given by the manufacturer.

Equation 3.24 - Drain-Source On-Resistance in High Temperatures

$$R_{DS(on)hot} = R_{DS(on)spec} \times \left(1 + 0.005(T_{J(hot)} - T_{spec})\right)$$

So the total losses are given by the following equation :

Equation 3.25 - Total Power Losses

$$P_M = \frac{1}{2} \times V_{DS} \times I_D \times f_{sw} \times (t_{rise} + t_{fall}) + R_{DS(on)hot} \times D \times I_{in}^2$$

These two types of losses are dissipated as heat, increasing the junction temperature. The manufacturer specifies the maximum junction temperature (150-200 Celsius) that the MOSFET can handle. The maximum power dissipation for a given maximum junction temperature can be seen below:

Equation 3.26 - Maximum Power Dissipation Junction

$$P_{Dmax} = \frac{T_{Jmax} - T_A}{\theta_{JA}}$$

where θ_{JA} is the thermal junction-to-ambient resistance.

This determines if the MOSFET actually needs a heat sink or not. The temperature rise of the MOSFET junction relative to the ambient temperature, ΔT_{JA} , is defined as :

Equation 3.27 - Change in Temperature at Junction

$$\Delta T_{JA} = \theta_{JA} \times P_D$$

with P_D being the dissipated power. When a heatsink is used, the expression for θ_{JA} changes to:

Equation 3.28 - Thermal Junction with Heat sink

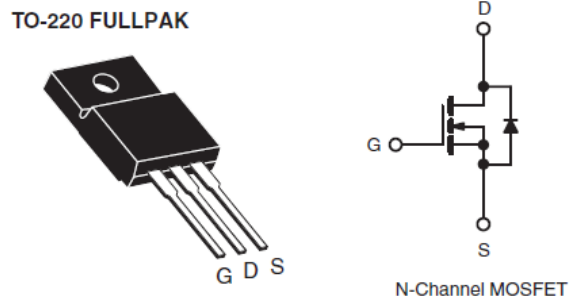
$$\theta_{JA} = \theta_{JC} + \theta_{CS} + \theta_{SA}$$

where θ_{JC} is the manufacturer specified MOSFET junction-case thermal resistance, θ_{CS} is the case-to-sink thermal resistance, and θ_{SA} is the sink-to-ambient thermal resistance.

Component Selection

For the design the team chose to use the Vishay IRF1640G Power MOSFET. This MOSFET met all of the previously specified requirements. The drain to source voltage limit is 200 V which is more than enough for the MPPT. Also if the gate to source voltage is set at 10 V then the continuous drain current can handle up to 6.2 A. This is also above the maximum current the maximum power point tracker will receive. The Drain-Source Resistance is equal to 0.18Ω . A picture of the MOSFET chosen is shown below in Figure 3.29.

Figure 3.29 - IRF1640G Power MOSFET



3.6 MOSFET Driver

Theory

In order for the MOSFET to work with high frequency, a drive circuit needs to be implemented in order to quickly switch the MOSFET between its on and off state. By doing so, the time spent by the MOSFET in its active region is limited and the dissipation losses are reduced. The drive circuit is basically amplifying the pulse modulated signal coming from the microcontroller to a voltage high enough so it can charge the gate capacitance in the MOSFET and turn it on.

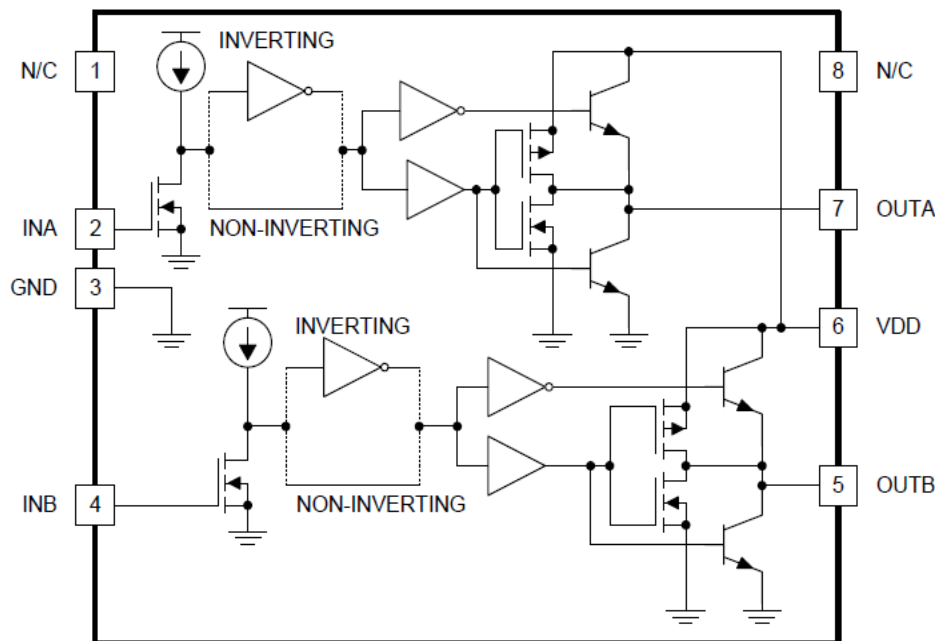
Component Selection

The first driver tested was the Dual 4-A Peak High-Speed Low-Side Power-MOSFET Driver produced by Texas Instruments as shown in Figure 3.30. These drivers can handle high frequency and large peak currents. The specific driver used was the UCC27324, which has a non-inverting configuration. Figure 3.31 shows a block diagram of the driver, which shows two circuits on the chip. Pins 1, 3, and 8 should always be connected to ground. When a pin is not being used on the chip, it should be connected to either ground or the voltage high to prevent floating voltages. INA/INB is connected to the function generator or microcontroller. VDD should be a voltage between 4.5V and 15V. The driver will be powered with the solar car's 12V line. The OUTA/OUTB pin should be connected to the gate of the MOSFET. This will allow switching between 0V and slightly less than 12V.

Figure 3.30 - UCC27324 MOSFET Driver

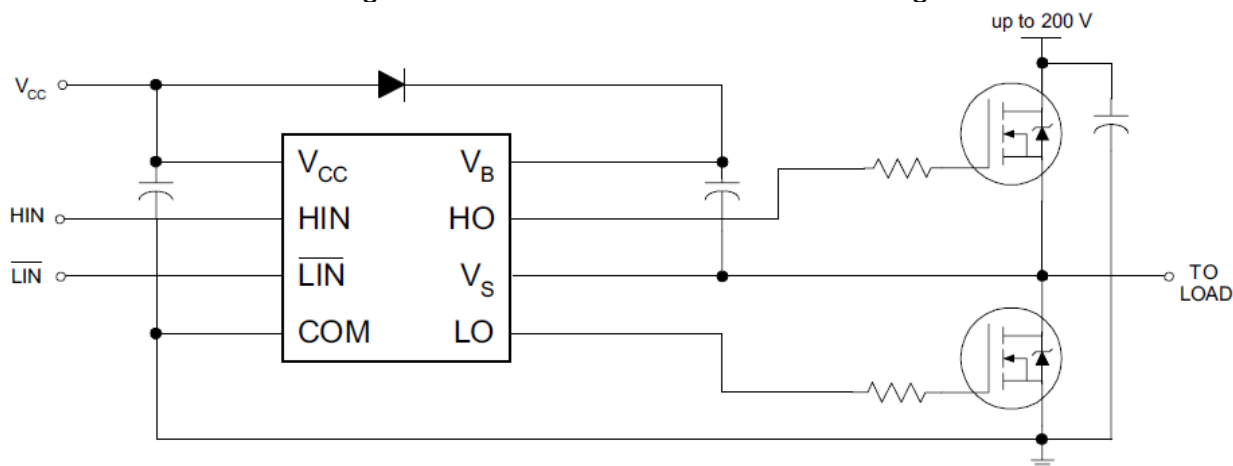


Figure 3.31 - UCC27324 MOSFET Driver Block Diagram
BLOCK DIAGRAM



The second driver that was tested was a IRS2003(S)PbF Half-Bridge Driver bought from International Rectifier. This component is a high voltage, high speed power MOSFET driver with dependent high- and low-side referenced output channels. It also is 3.3V logic compatible for its logic input. The driver is fully operational to up to 200V and has a gate drive supply range from 10V to 20V which is large enough to turn on the MOSFET. The driver was implemented with two MOSFETs, one for the high-side and one for the low-side. A typical connection for this specific driver is shown in Figure 3.32. Both resistors chosen were $1k\Omega$ and the diode chosen was a 1N4148. The bootstrapping and bypass capacitors were selected to be $0.1\mu F$.

Figure 3.32 - Circuit Schematic for Testing



3.7 Ideal DC-DC Booster PSpice Simulation

PSpice is an analog circuit simulator. An ideal DC-DC booster circuit was coded and simulated using the program. Figure 3.8 shows the specific schematic. Component values which were determined using LabVIEW were used for the simulation. Figure 3.33 shows the source code for the PSpice circuit. A 20V DC source (V_IN) was used to simulate the low end voltage range the MPPT is designed for. Then the switch was modeled using a current controlled voltage source. A current pulse has a $1\mu s$ rise and $1\mu s$ fall delay. Its pulse width is $5.6\mu s$ and a period of $10\mu s$. This means the pulse will have a 56% duty cycle. The duty cycle can be changed to simulate different output voltages. A $1k\Omega$ load resistor was used..

Figure 3.33 - PSpice Simulation Script

```
MSU Powerpoint Tracker Team by Brenton Sirowatka
.SUBCKT Ideal_Diode anode cathode
Vx anode int DC 0V
Ed int cathode TABLE {I(Vx)} = (-0.01A,-650V) (0A,0V) ;characteristics of diode
.ENDS
V_IN 1 0 DC 20V
C_IN 1 0 460uF
```

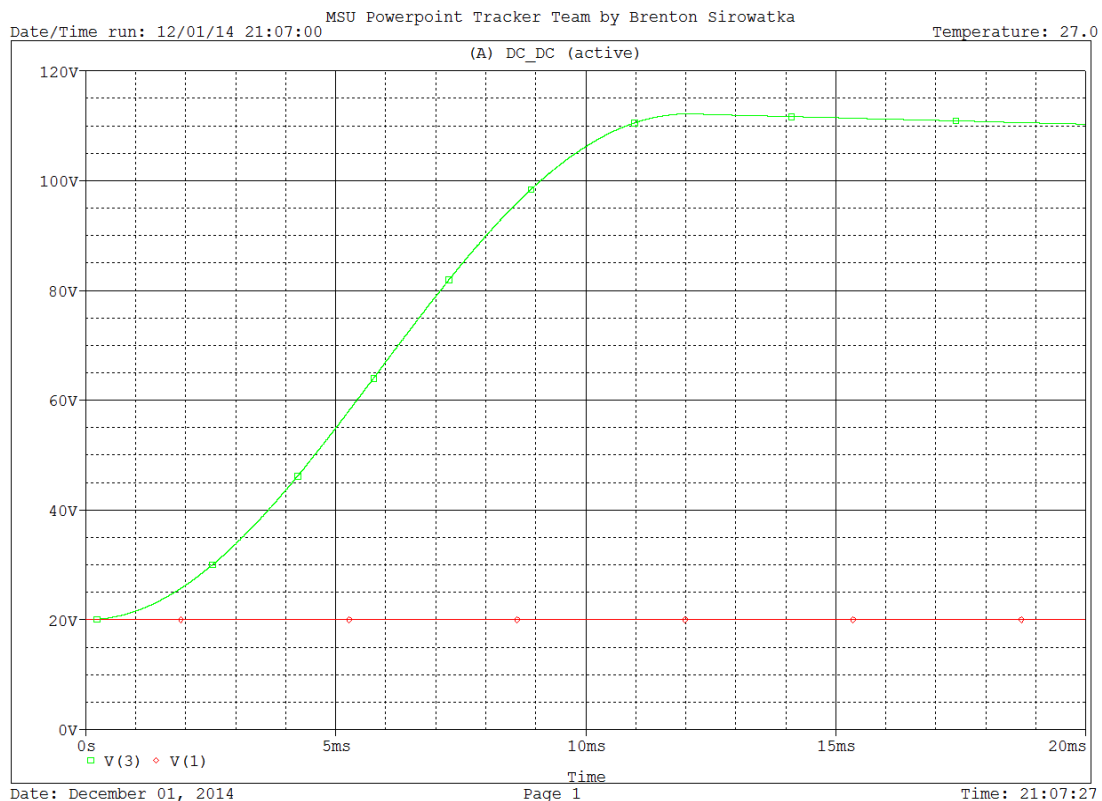
```

***Switch
Ipls 0 10 PULSE(0A 2mA 0s 1us 1us 5.6us 10us) ;PULSE(A1 A2 Td Tr Tf Tw Period)
Vctrl 10 11 DC 0V
Rpls 11 0 1m
***
L1 1 2 2.75mH ;IC=9.4A
Wsw 2 0 Vctrl Wmod
;http://www.uta.edu/ee/hw/pspice/pspice10.htm
Xd 2 3 Ideal_Diode
.MODEL Wmod ISWITCH(Ron=1m Roff=1MEG Ion=1mA Ioff=0A)
C_OUT 3 0 460uF ;IC=10.25V
RL 3 0 1K
.PROBE
.TRAN 1ms 20ms 0us 1ms ;UIC
.END

```

Figure 3.34 shows the transient response of the PSpice code. The pulse width was adjusted to make the output voltage 110V, which is the battery's voltage of the solar car. Under these ideal conditions, it would take ~10ms for the voltage to rise from 20V to 110V, as indicated by the green line. The circuit shows a slight 2V overshoot at 10ms and settles on 110V at 18ms.

Figure 3.34 - Ideal DC-DC Boost Converter Simulation



A more realistic simulation would require modeling of the MOSFET to be used, which requires either finding PSpice code for the component, or inputting the values manually from the datasheet.

3.8 Sensing

The design needs to measure the current and voltage from the solar arrays that are connected to the MPPT. Therefore, there is a need to utilize voltage and current sensing modules (14).

3.8.1 Voltage Sensing

Information

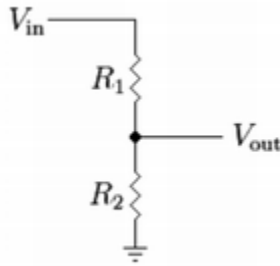
The perturb and observe algorithm requires voltage and current readings from the DC-DC converter in order to calculate the power at any given instance. Therefore, a voltage sensing module is required so the actual input and output voltages of the DC-DC Boost Converter can be monitored. Voltage measurement can be done in several ways. The table below shows some of the ways in which voltage can be monitored.

Table 3.35 - Voltage Sensor

Class	Principle of Operation	Application Field
Electrodynamic	Interaction between currents	DC and AC current
Electromagnetic	Interaction between magnets and magnetic fields	DC current
Electronic	Signal Processing	DC and AC current
Electrostatic	Electrostatic interactions	DC and AC current
Induction	Magnetic induction	DC and AC current
Thermal	Current thermal effects	DC and AC current

There are six different ways to measure voltage from the above table. However, only the electronic method is feasible. This is because electrodynamic, electromagnetic, electrostatic, induction and thermal measurement requires larger components which are not suitable for this project. Therefore, the voltage can be measured using an analog-to-digital(ADC) converter that is built into the microcontroller. However, the ADC only accepts voltages between 0 to 5 V. This needs to be taking into consideration because the input voltage of the main circuitry is expected to be between 20 to 60 V. Therefore, the voltage needs to be scaled.

Figure 3.36 - Voltage Divider



A voltage divider is a serial connection between two resistors as shown in Figure 3.36 . The voltage is divided between the resistors because of Kirchhoff's voltage law. Thus, it can be used to scale down the voltage from the DC-DC Boost Converter.

In terms of choosing the resistor combination, there are two factors to consider. The first factor is to make sure that the voltage divider will draw minimum current from the main circuitry because efficiency is a major design requirement. Second, the resistor combination needs to be selected so that at maximum input voltage the output from the voltage divider will not exceed 3.3V (i.e. maximum input voltage to the ADC modules of the microcontroller).

In Equation 3.37 the relationship between V out and V_{in} is shown:

Equation 3.37

$$V_{out} = V_{in} \frac{R_2}{R_1 + R_2}$$

R1 is selected to be 3.3MΩ. This is to make sure that the voltage divider will not draw significant current from the DC-DC Boost Converter. As for the selection of R2, the value is determined by using equation 3.37. Since the maximum output voltage from the voltage divider needs to be less than 3.3V when the voltage at the DC-DC Boost Converter is 60V. The result of the calculation of R2 yields an approximate value of 192kΩ. The final value of R2 is chosen to be 180 kΩ because it is the closest available resistance and it gives some tolerance for the DC-DC Boost Converter's voltages. This is because we want to make sure that the microcontroller will still be able to track the voltages even if the DC-DC Boost Converter's voltage goes slightly above 60V. Based on the selection of R1 and R2 the expected sensing range of the voltage divider is between 0 to 63.8V.

3.8.2 Current Sensing

Information

The perturb and observe algorithm requires the readings voltages and current from the DC-DC converter to be able to calculate the power. Therefore, a current sensing module is required so the actual input and output currents of the DC-DC Boost Converter can be monitored. There are different methods for sensing current. Several of the common current sensing methods are shown below:

Table 3.38 - Current Sensor

Type	Sensing Range	Isolation	Accuracy	Intrusive	Cost
Magnetic	Medium - Very High	Yes	Medium	No	High
Optically Isolated Resistive	Medium - High	Yes	Low - Medium	Yes	Medium
Resistive	Very low - High	No	High	Yes	Low

Magnetic current sensing can be done in several ways:

- Current Transformer
- Flux-Gate magnetometer
- Magneto diode
- Rogowski coil
- Search-coil magnetometer

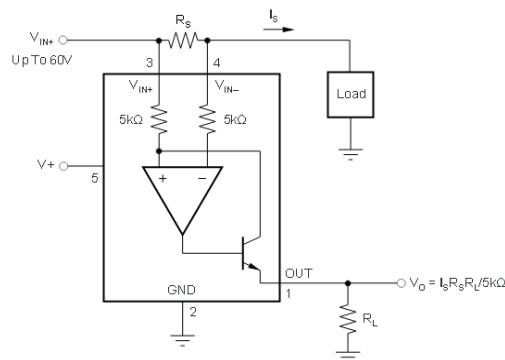
The above methods are commonly used to measure AC current. Therefore, the team has discarded the magnetic current sensing option because the design only operates in DC.

Optically isolated resistive current sensing is commonly used in high-current systems. However, optical sensors are not practical for the design due to its complexity, size, and cost.

Since both the magnetic and optically isolated resistive current sensing methods are not feasible, the resistive method is going to be implemented. A resistive measurement, or shunt measurement utilizes Ohm's law in order to measure current. Shunt resistors are low value resistances that are placed in series with the path of the current the user is trying to sense. The current can be determined because the voltage across the shunt resistor is proportional to the current flowing through.

INA168(High-Side Measurement Current Shunt Monitor)

Figure 3.39 - INA168



The team has chosen the INA168 current shunt monitor as the current sensing module. The most important reason for selecting this particular module is because of its large input range (i.e. V_{in+} and V_{in-} can have a maximum common-mode input of 60V). As shown in figure INA168, the output from the current sensing module is governed by the following equation:

Equation 3.40 - Voltage across Load Resistance

$$V_o = \frac{I_s R_s R_L}{5k\Omega}$$

Where I_s is the current that is being measured, R_s is the shunt resistance, R_L is the gain resistance, and the $5k\Omega$ resistance is the reciprocal of the internal transconductance of INA168 (i.e. $\frac{1}{200\mu A/V}$). The voltage across R_s has to be large enough in order for the sensor to get an accurate reading of the current. For the INA168 the minimum voltage value across the shunt resistor is 50mV.

3.9 Microcontroller

A microcontroller is a small computer, usually packaged into a small chip, which has the ability to run programs and interface with internal and external peripheral such as C programs, ADCs, and sensors. Microcontrollers are extremely useful because of their ability to perform these actions coupled with the fact that they are economical. Microcontrollers usually operate at lower frequencies (MHz range) and consume much less power than consumer computers, and have smaller magnitudes.

There are numerous microcontrollers on the market, each with its own features to be considered when choosing a controller for a design. The Texas Instruments C2000 Piccolo microcontroller was chosen in order to run the Perturb and Observe algorithm in our design. The Piccolo is on a Launchpad which has a USB port for easy connection to a computer. The C2000 Piccolo Launchpad can be seen in figure 3.41.

Figure 3.41 - C2000 Piccolo Launchpad



The Piccolo has a 32-Bit CPU, 60MHz clock, and operates on 3.3V supply. It has 22 individually programmable GPIO pins with Enhanced Pulse Width Modulation

(ePWM), Analog to Digital Converter (ADC), and Enhanced Capture Module (eCap). To use any of these modules, they must first be setup. Texas Instruments has split up the different components of the microcontroller into individual user guides, which can be found on their website. See Appendix 3 to reference the C2000 MPPT code and MSP430 MPPT code. Code Composer Studio, which is an integrated development environment (IDE), was used to load the program onto the microcontroller and debug operation.

3.9.1 Device Initialization

Microcontrollers require their peripherals to be setup before they can be used by the program. The lines of code previously mentioned are discussed in Table 3.42.

Table 3.42- Initialization Code

WD OG_disable(myWDog);	A watchdog timer forces a device reset after a set number of clock cycles. This is used when a microcontroller is not easily reset and requires added code. Our implementation does not require this feature, which is enabled by default.
PLL_setup(myPII, PLL_Multiplier_12, PLL_DivideSelect_ClkIn_by_x);	This sets the frequency of the phase locked loop.

3.9.2 Analog to Digital Converter

An analog to digital converter (ADC) allows continuous signals to be discretized, which allows digital logic to be performed by computer like a microcontroller. An analog signal can take on any amplitude at any instant of time. When a signal is being measured by a microcontroller, the amplitude must be within the limit of the ADC. This means, for example, if a 20V sine wave is to be measured by a 5V ADC, a voltage divider with a ratio of $\frac{1}{4}$ could be used to ensure any value is within 0V and 5V. The number of bits of an ADC determines the resolution of the measurement.

The C2000 Piccolo has a 12bit ADC, which means a signal measured can have a value between 0 and 4095 in decimal (0 and FFF in Hexadecimal). If the C2000 ADC is set to 3.3V, the ratiometric value, which is $\frac{\text{Resolution of the ADC}}{\text{System Voltage}}$, will be $\frac{4096}{3.3} \approx 1241$.

The ratiometric value also equals $\frac{\text{ADC Reading}}{\text{Analog Voltage Measured}}$. Knowing the ratiometric value equals 1242, a measured voltage of 2V will give a decimal value of 2730. This decimal value can be converted back to the measured analog voltage using the ratiometric value and then it can be scaled to the unscaled analog value ratio to get the unscaled analog value. For example, a 60V signal scaled down to 3.3V gives a ratio of $\frac{60V}{3.3V} = 18.18$. An ADC reading of 2730 gives a scaled voltage of 2V, which can be multiplied by 18.18 to give an unscaled voltage of 36.36V.

The Texas Instruments Piccolo Analog-to-Digital Converter and Comparator Reference Guide and Real-Time Control Peripheral Reference Guide describe which pins have ADC functionality and how they are setup. A few of these lines of code are shown in Table 3.43.

Table 3.43 - ADC Code

ADC_setSocChanNumber (myAdc, ADC_SocNumber_x, ADC_SocChanNumber_Ax);	This selects which IO pin will be sampling. Only certain pins have ADC functionality and can be viewed in the Piccolo Launchpad Experimenter Kit User's Guide under Device Pin Out. Example: A0 is pin 6 on J5.
sum_of_ADC_samples_Array[x] += AdcResult.ADCRESULTx;	ADCRESULTx is the register where the measured ADC value is stored after a conversion. Each new conversion will overwrite the last.

3.9.2 Pulse Width Modulator

A pulse width modulation (PWM) signal can be used to turn a switch on and off. Common applications include dimming an LED or driving a motor. A pulse width is defined as the time a signal is high divided by the period of the signal ($Pulse\ Width = \frac{Period\ High}{Period}$) where Period = Period High + Period Low. Another way to describe a PWM signal by using the concept of a Duty Cycle. A duty cycle is the pulse width multiplied by 100%. For example, a 50% duty cycle is on half the time, and off the other half.

Texas Instrument's Piccolo Enhanced Pulse Width Modulator Module guide describes which pins have PWM functionality and how to use the C2000 Piccolo's PWM module. The PWM setup is shown in the C2000 MPPT code in Appendix 3. A few lines to note include the lines which set up the PWM frequency and duty cycle, as shown in Table 3.44.

Table 3.44 - PWM Setting example

GPIO_setPullUp(myGpio, GPIO_Number_x, GPIO_PullUp_Disable);	In order for a microcontroller pin to be used as a PWM pin, the pin pull-up should be disabled since it is enabled by default and is required for PWM.
PWM_setPeriod(myPwm1, 150);	This changes period (or frequency). 150 corresponds to a frequency of 100kHz. In the code, this value is static.
PWM_setCmpA(myPwm1, 75);	This changes the pulse width (function of duty cycle) of the

signal, and will change when change to either increase or decrease the array voltage. 75/150 gives a 50% duty cycle.

It is important to ensure the 'setCmp' value, which sets the pulse width, always stays under the 'setPeriod', which sets the period, or else the PWM module will not work. 'setCmpA' is not setting the duty cycle, but the pulse width.

3.10 Algorithm

There are eight methods that can be used to find the maximum power point of an photovoltaic array (15).

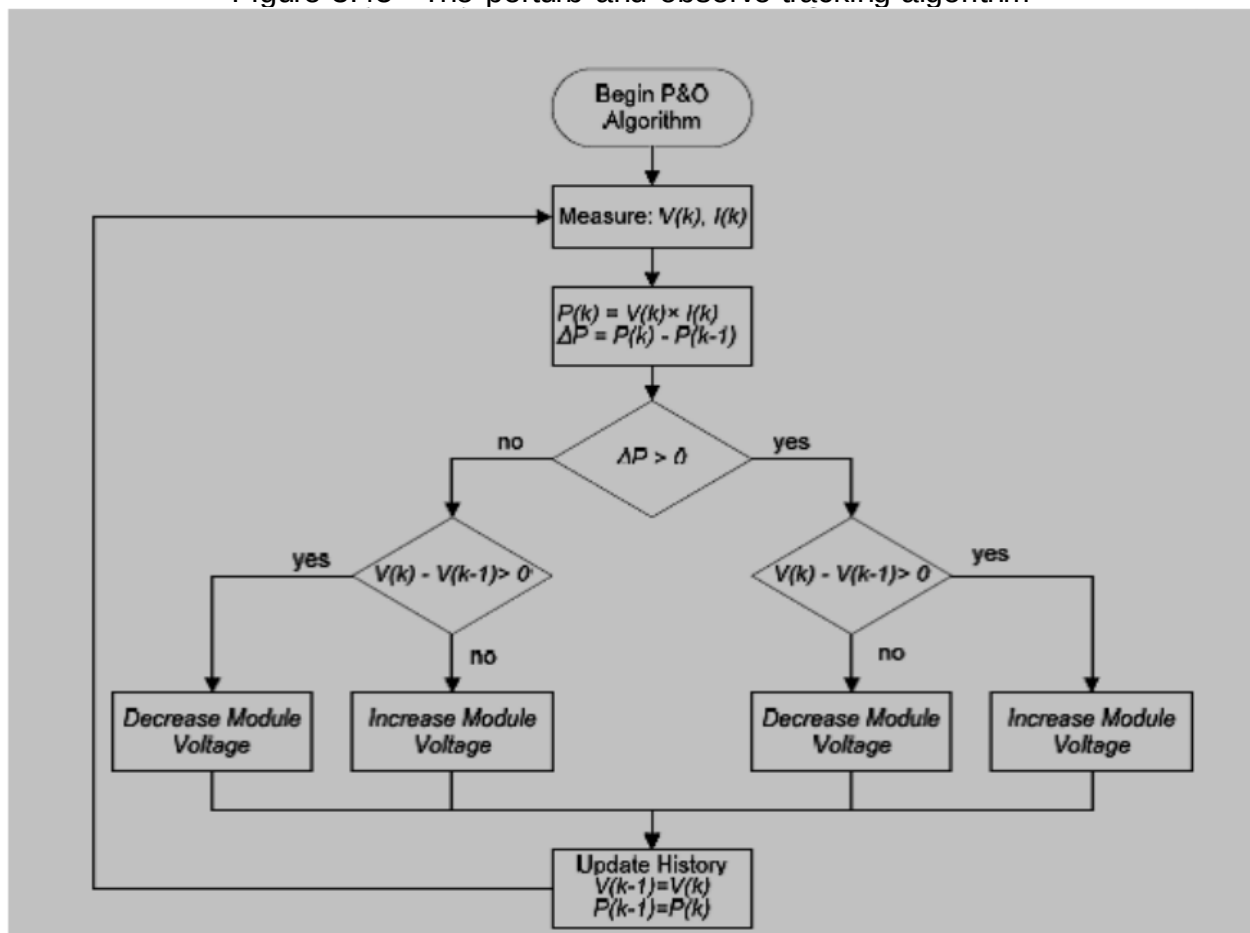
- Constant Voltage
 - A single predetermined voltage represents the maximum voltage point(V_{MP}). It has an estimated efficiency of 80%
- Open Circuit Voltage
 - The system finds the open circuit voltage (V_{OC}) and uses this to find the V_{MP} . This is calculated using the equation $V_{MP} = k * V_{OC}$, where k is between 0.7 and 0.8.
- Short Circuit Current
 - The system uses a short load pulse to generate a short circuit condition. The short circuit current (I_{SC}) is used to estimate the maximum point current (I_{MP}) using the equation $I_{MP} = k * I_{SC}$.
- Current Sweep
 - The current sweep method uses a sweep waveform for the PV array current such that the I-V characteristic of the PV array is obtained and updated at fixed time intervals.
- Perturb and Observe
 - The system searches for the maximum power point by changing the PV voltage or current and detecting the change in the photovoltaic power output.
- Incremental Conductance
 - The system uses incremental conductance to locate the maximum power point when $\frac{dI_{PV}}{dV_{PV}} + \frac{I_{PV}}{V_{PV}} = 0$.
- Temperature [3]
 - This method employs a sensor in order to obtain a sample of the photovoltaic surface temperature and then uses that temperature to find the optimal voltage that should be pushed across the device.
- Temperature Parametric [3]
 - The temperature parametric method uses the below equation to calculate the maximum power point voltage instantly by measuring time and solar irradiation.

$$V_{MP} \cong \left[(u + S \cdot v) - T \cdot (w + S \cdot y) \right] \cdot V_{MP_SC}$$

3.10.1 Perturb and Observe

Perturb and Observe (P&O), as shown in Figure 3.45, is the algorithm the team selected. It is one of the most commonly used methods due to its ease of implementation and relatively high efficiency. To implement the P&O algorithm, voltage and current must be known. Voltage and current sensors are placed before the boost converter and must be able to be read by the microcontroller, as explained in the voltage and current sensors section. After the voltage and current measurements are made as described in the ADC section, these values can be used in the algorithm. The power of the photovoltaic array is found by multiplying the voltage and current. An initial value should be set in software for the first iteration, for example the power can equal zero. This new power value will be compared to the last power value. The voltage is then compared to the old voltage value. When an increase in module voltage is required, duty cycle should be increased. When a decrease in module voltage is required, the duty cycle should be decreased. The new voltage, current, and power values should now be saved as old values and the cycle is repeated.

Figure 3.45 - The perturb and observe tracking algorithm



The increment at which the duty cycle should be changed should be set in software. If the increment is large, the algorithm will overshoot the actual maximum power point and will continue to overshoot around this point. One advantage of this is that it can find the estimated power point quickly, which is beneficial for fast changing environments, but is less efficient in slow changing environments. By setting the increment small, the algorithm will take much longer to find the maximum power point, but will not overshoot this point as much. This is optimal for slow changing environments. With the increased speed of most microcontrollers, a smaller increment is advantageous to ensure efficiency.

3.11 MPPT CAD Prototype

The following figures are CAD prototypes of a MPPT system. They were made using Siemen's NX software. These drawings reflect our current design status and how it could still be assembled. Stacks of PCBs could be implemented for design testing and interchangeability. For example, the voltage and current sensor, which are more sensitive, would be placed on their own PCB to allow removal if they get damaged.

Figure 3.46 - Conceptual CAD Drawing of Current Sensor Module

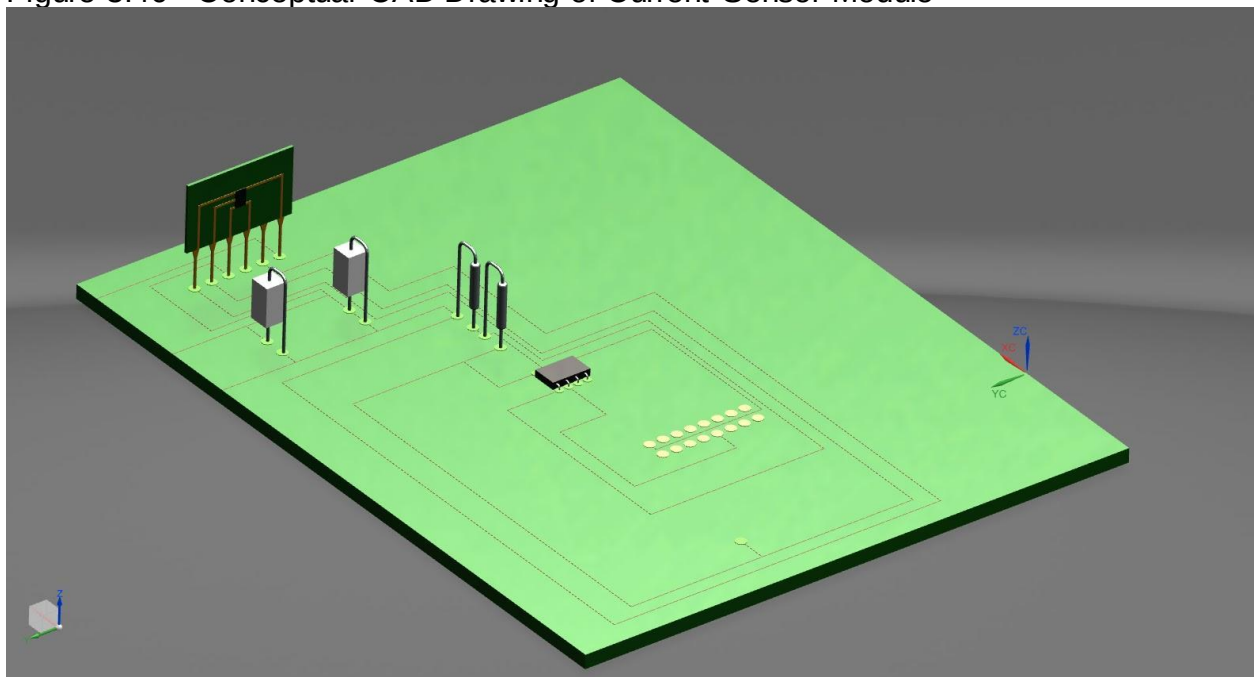


Figure 3.47 - Conceptual CAD Drawing of the DC-DC Boost Converter

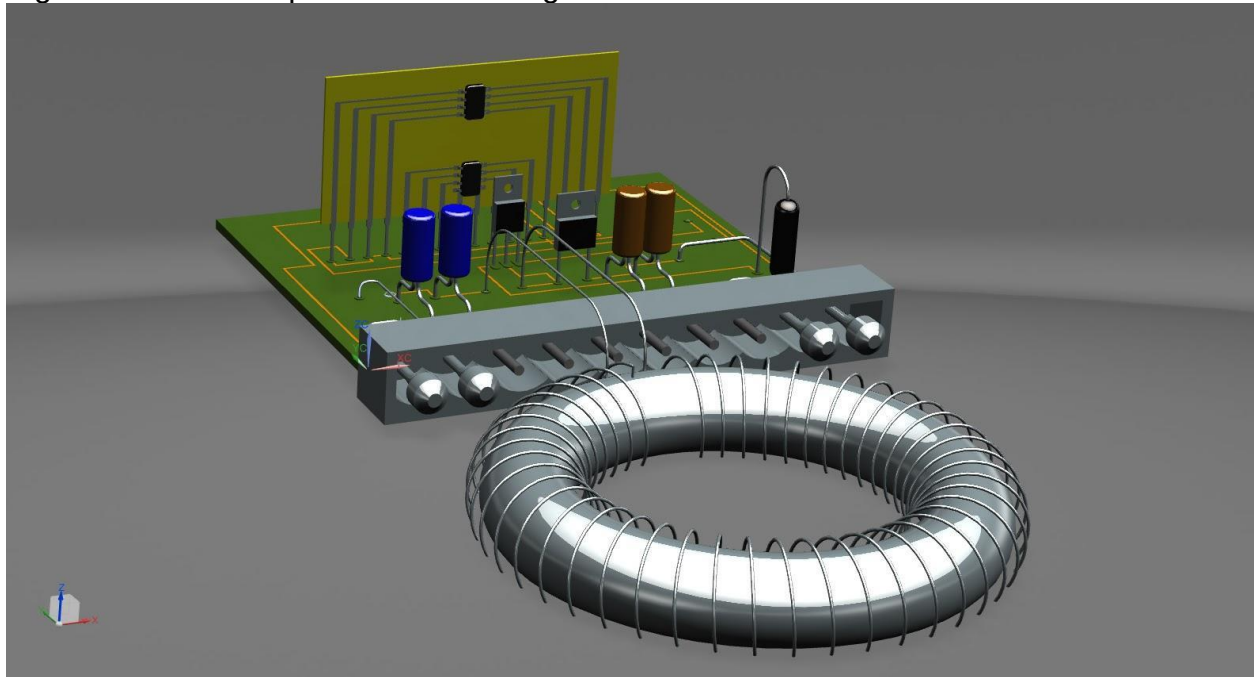


Figure 3.48 - Conceptual CAD Drawing of the MPPT

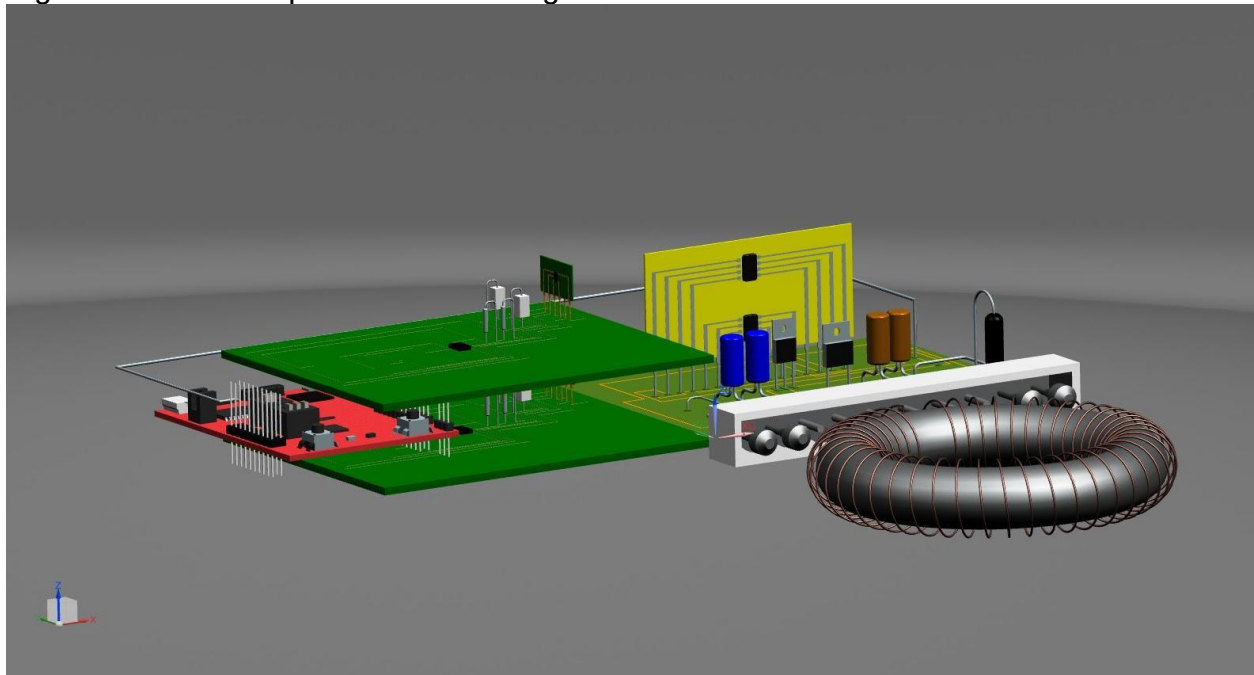
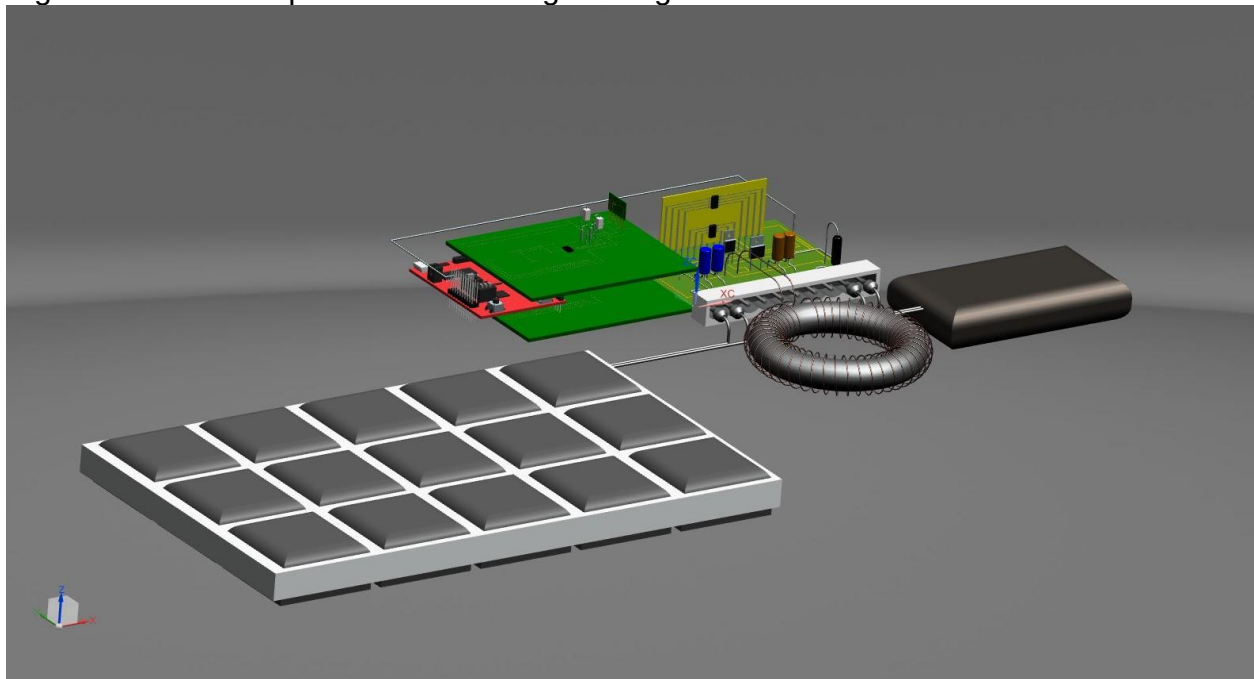


Figure 3.49 - Conceptual CAD Drawing of Integration of MPPT



Chapter 4 –Test data with proof of functional design:

4.1 Testing - DC/DC Booster

In order to build our DC/DC booster the team used the simple schematic provided to us by the project description. The team used the parts specified in chapter 3 to create the booster. Although, on the first attempt the team used the function generator in order to switch the MOSFET instead of the gate driver. The team tested this separate at first to ensure that the DC/DC booster was working before anything else was implemented into the design. For testing purposes we used a resistor as our load which was able to withstand several watts. This high wattage resistor was necessary because otherwise the resistor would heat up very quickly and burn. A photo of the circuit displayed on a protoboard can be seen in Figures 4.1 and 4.2 below.

Figure 4.1 - DC/DC booster

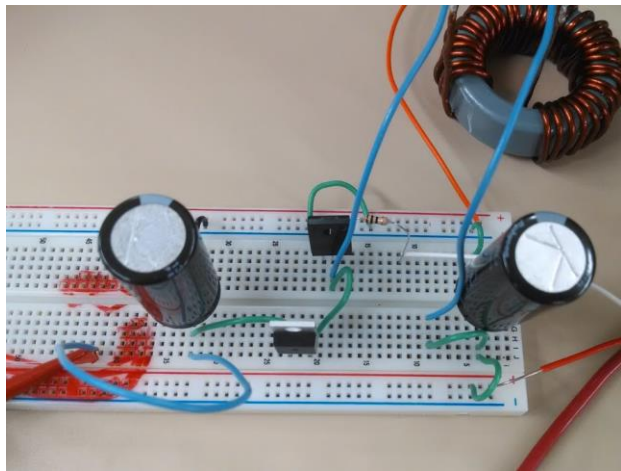
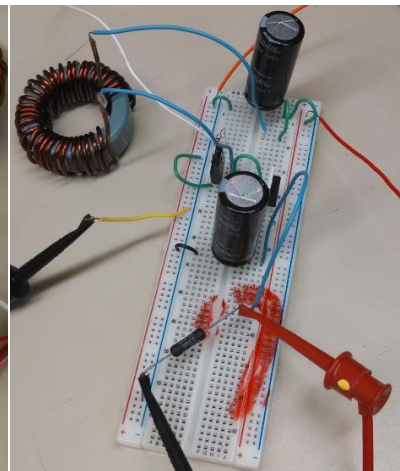
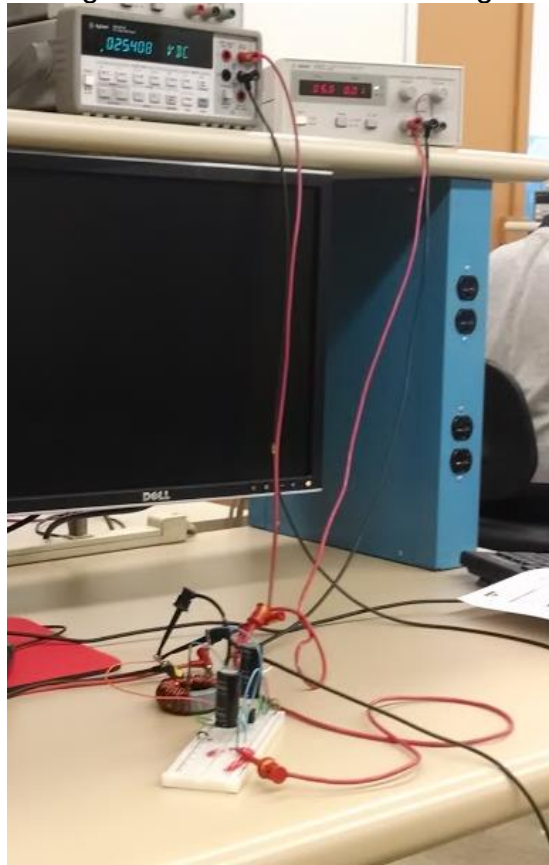


Figure 4.2 - DC/DC booster with load



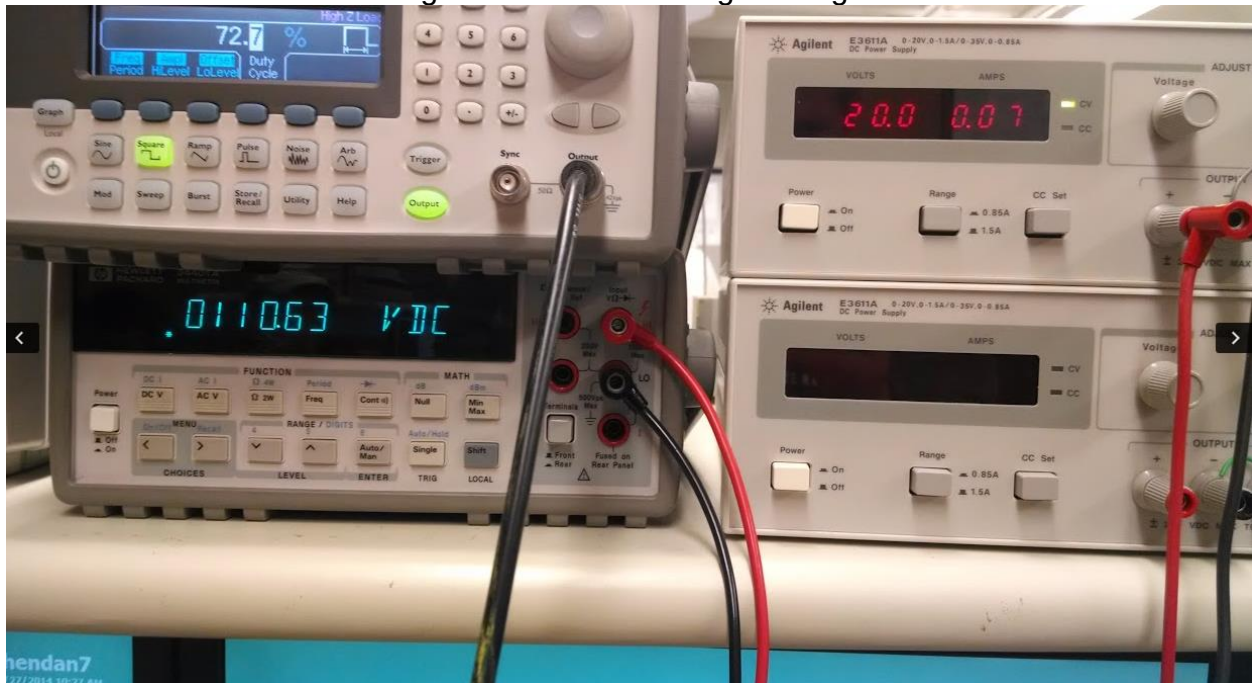
On the first attempt the team decided to use a low voltage of 5V as the V_{in} for the circuit so that the output voltage did not exceed 50 V. A photo of this first attempt can be seen in figure 4.3 below.

Figure 4.3 - DC/DC Low Voltage



In the figure the power supply is connected to V_{in} while the multimeter is connected to the load. As shown the team was able to boost a 5 V supply into 25.408 V using our booster. Yet the team was only able to draw a very small current of approximately 0.01 A and that is because the team used a high resistance at the load of $12\text{ k}\Omega$. The team then increased the voltage of the power supply to see if the booster could give the required voltage of 110V. Next a protective box was used in order to cover the design for safety purposes. The team was successful in getting to the required voltage at an input voltage of 20 V and a duty cycle of 72.7%. A current of only 0.07 A was being drawn. A photo of these results can be seen below.

Figure 4.4 - DC/DC High Voltage



4.2 Testing MOSFET Driver

Two MOSFET drivers were tested for the project. The UCC27324 by Texas Instruments and the IRS2003 by International Rectifier. The IRS2003 is a half bridge driver which is supposed to be a more effective method for driving the MOSFET in MPPT applications. The team attempted to use the half bridge by wiring the HIN and LIN pins to the same input from the function generator. This method was ineffective and the team was unable to get the driver to successfully drive the current so it was decided going back to our original plan of using the UCC27324 driver was the best option.

Figure 4.5 shows testing of the UCC27324 driver. “A” is showing an Agilent 33250A Function Waveform Generator producing a frequency of 1kHz, voltage of 3.3Vpp, DC offset of 1.65V, and a duty cycle of 50%. The frequency was chosen as a starting point and later increased to 100 kHz. The function generator acted in place of the C2000 microcontroller’s pulse width modulation feature. According to the UCC27324 datasheet, the input high had a minimum of 2V and an input low had a maximum of 1V. 3.3V is operating voltage of the C2000 microcontroller, so it could switch the driver without any extra hardware peripheral. The UCC27324 required a voltage between 4.5 to 15V. As shown in “B” this testing used 7.2V. The final setup would be powered by the solar car’s 12V line. Testing of 12V was later done to verify it would work. A voltage between 10V and 20V is needed to turn the MOSFET on. “C” shows the output waveform of the driver. The 1/ΔX under “X” shows the waveform is 1 kHz and the Δ under Y shows a voltage difference of 6.96V. The scope’s marker feature was used to ignore the overshoot voltage. “D” shows the circuit being tested with a 1kΩ load resistor.

Figure 4.5 - MOSFET driver Test

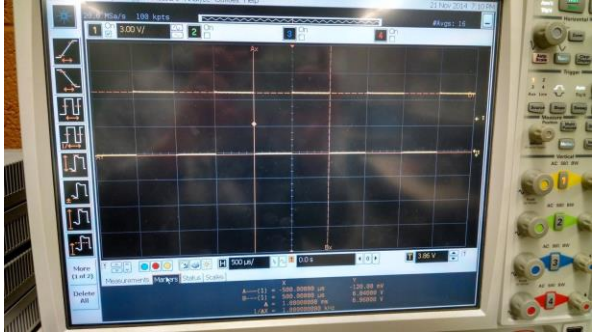
A - PWM from Function Generator



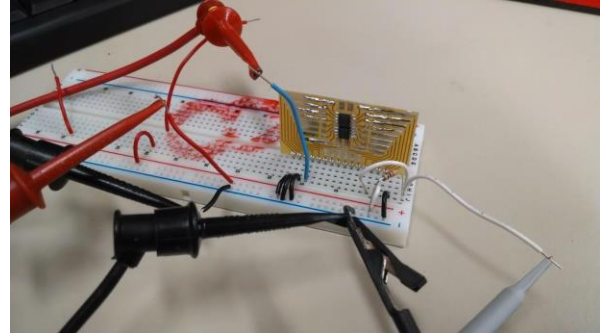
B - Bias Voltage



C - Amplified Signal



D - MOSFET driver test set up



4.3 Testing - MOSFET, Driver, & DC-DC Booster

After the MOSFET and driver were successfully tested together, the circuit was connected to the DC-DC booster circuit. Figure 4.6 shows the instruments used to power and measure the circuits. "C" in Figure 4.7 shows the function generator sending a waveform with a 100 kHz frequency, 50% duty cycle, 3.3Vpp amplitude, and 1.65V offset. This is used in order to mimic the pulse width modulation provided by the Microcontroller. The power supply next to the function generator was powering the MOSFET driver at 10.2V. "A" in Figure 4.7 shows the waveform generated by the MOSFET driver when it is being powered by roughly 10.2V. "B" shows the waveform generated by MOSFET driver when it is being powered by roughly 15.4V. "D" in Figure 4.7 shows the input voltage of the DC-DC booster as 18.6V being boosted to 50.8V, as being read on the HP 34401A Multimeter.

Figure 4.8 shows the circuit connections. The protoboard on left holds the MOSFET driver circuit

Figure 4.6 - Instruments for Testing

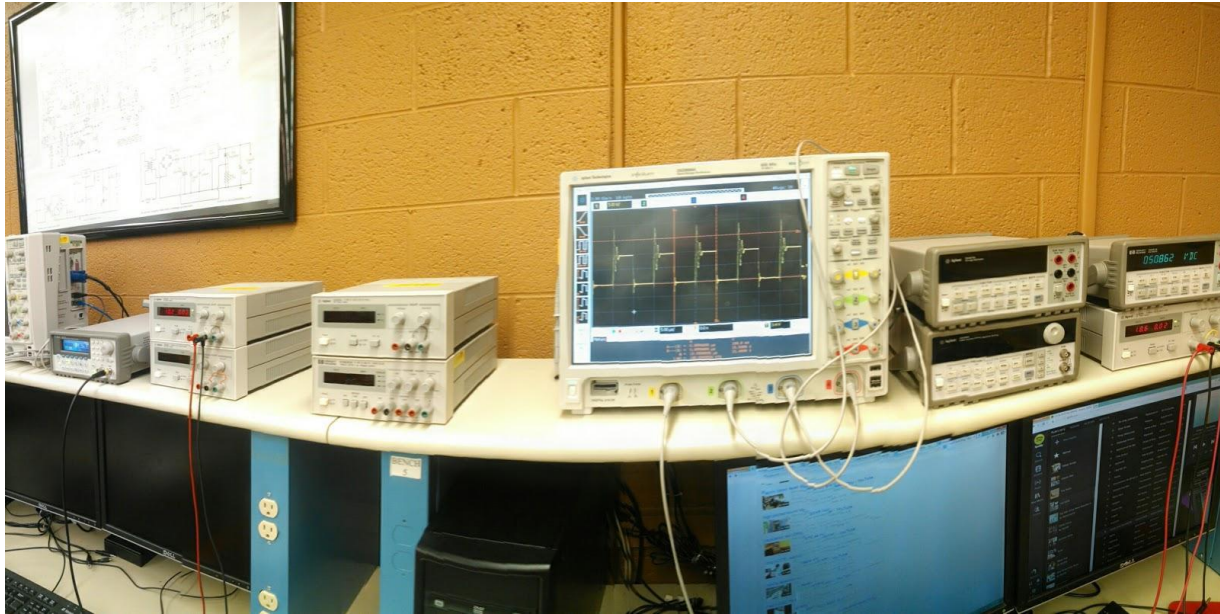
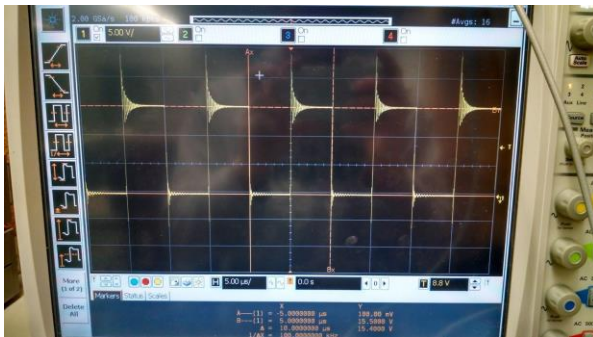
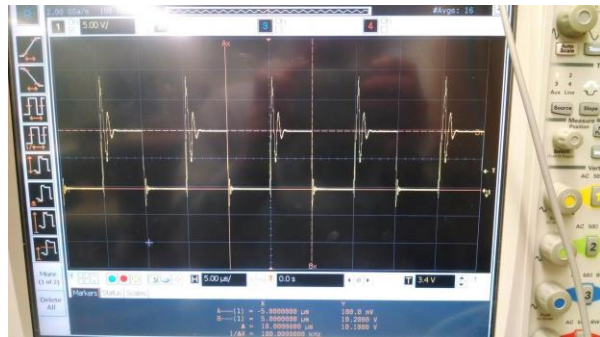


Figure 4.7 – Booster Testing

A



B



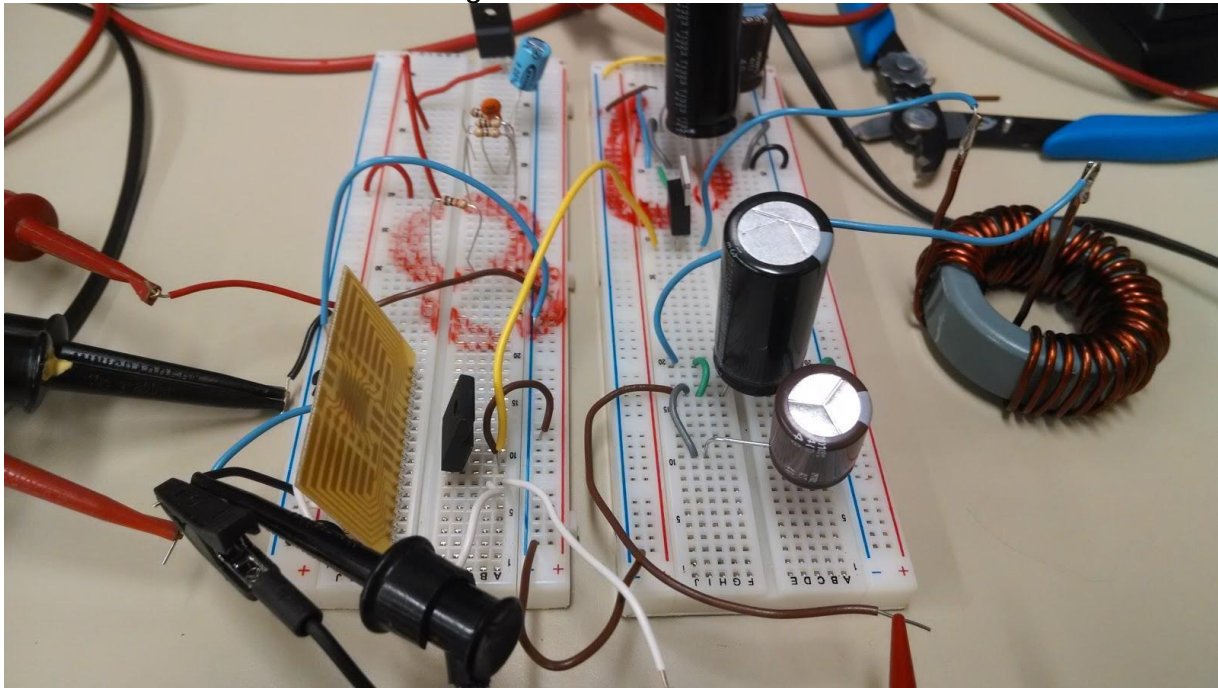
C



D



Figure 4.8 – DC/DC Circuit

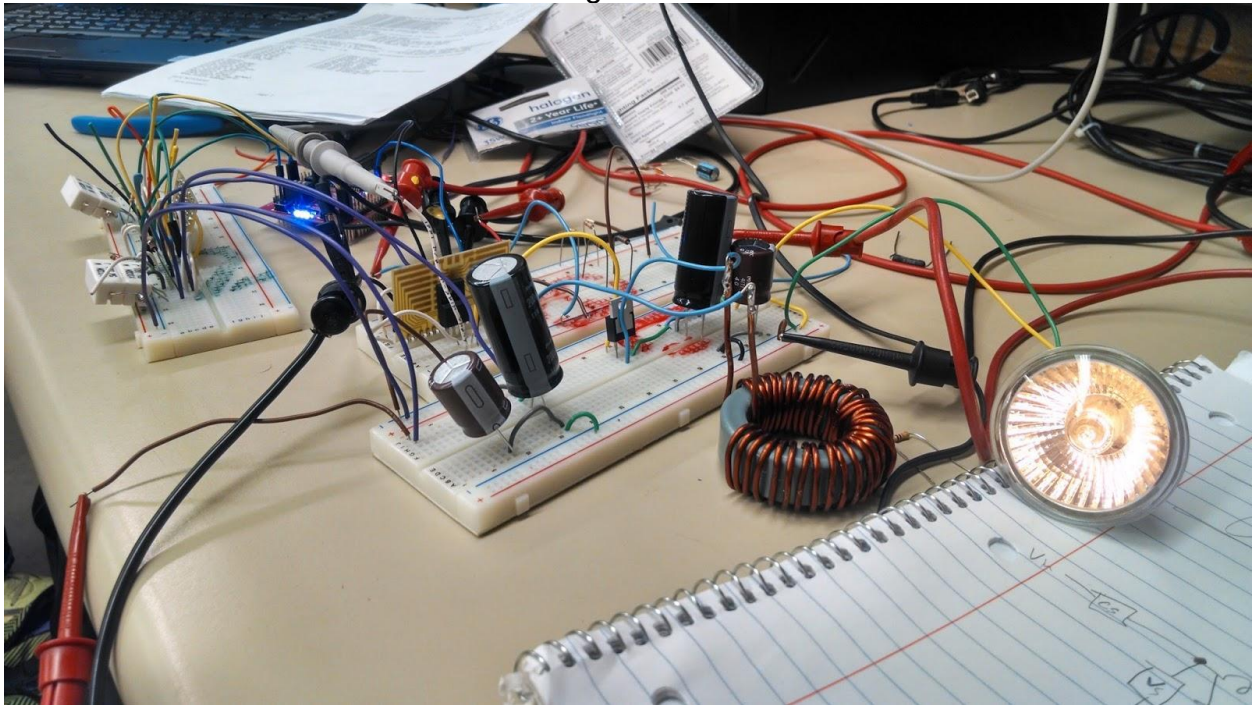


4.4 Testing - Light Bulb Load

Instead of using a 5 Watt resistor as a load, a 35 Watt halogen bulb was used. This is shown in Figure 4.8. The bulb used was a GE GU10 base halogen bulb with 35W rating at 120V. This is necessary for testing the current sensor. The current sensor requires enough current through its circuit load to read accurately. By replacing the resistor with the bulb, more current was able to be drawn. The bulb can also handle much more power. When using a $12k\Omega$ resistor, only about 0.01A of current was being read by the DC power supply. The Agilent E3611A has a resolution of 0.01A on its display. This current drawn makes sense because if the booster is working correctly and outputting 110V across the load, the current consumed would be

$$\frac{110V}{12k\Omega} = 9.16mA \approx 0.01A.$$

Figure 4.8

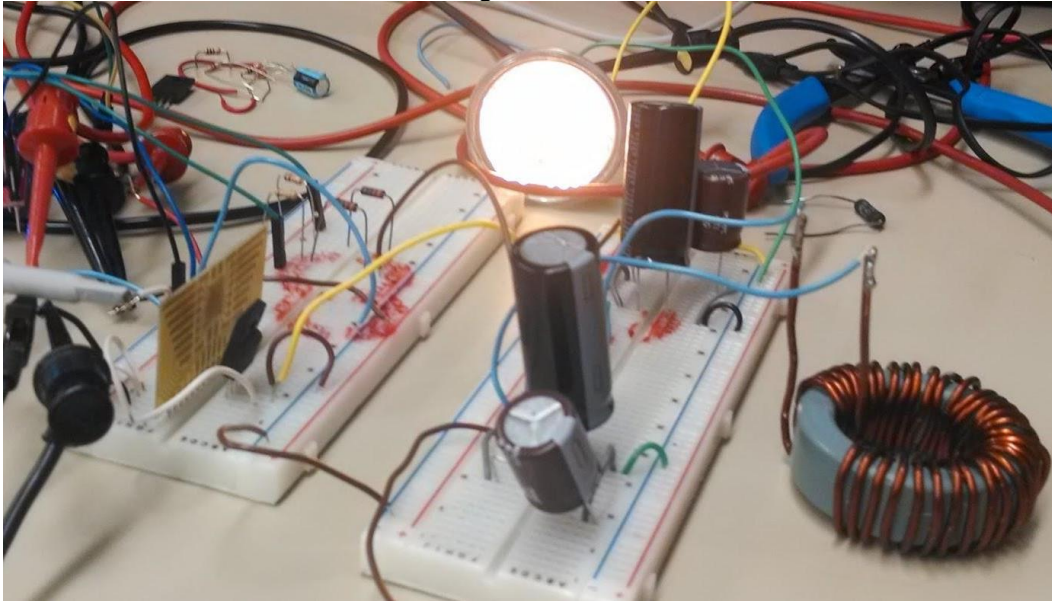


The function generator provided the PWM, producing a 100 kHz square wave with 3.3V amplitude and 1.65V DC offset and 50% duty cycle. By changing the duty cycle of the function generator, the halogen bulb's brightness could be observed. The halogen bulb was brightest at 60% duty cycle as shown in Figure 4.10. During testing, 18V was set for the power supply, but dropped to 4.7V. The boost converter was outputting between 30V to 40V across the bulb. At around 70% duty cycle, the voltage across the load was 37.3V as shown in Figure 4.9. As the duty cycle was increased above 70%, the bulb got dimmer and the circuit started malfunctioning. This is due to the fact that the DC power supply was being limited at its 1.58A maximum current (its specification states it is rated at 1.5A). The DC power supply voltage started falling. It should be noted that the Agilent 33250A function generator had a maximum duty cycle of 80%.

Figure 4.9



Figure 4.10



The team tested this portion of the design a second time using higher voltages. For this test the team used the $12k\Omega$ resistor at the load and a voltage of blank at the input. Table 4.11 displays the results of this test. The team used a the function generator to simulate the PWM of the microcontroller. It was set at 3.3 V peak to peak, 100 kHz, and 1.65 V. The MOSFET driver was supplied with a voltage of approximately 12 V and the input voltage was set at 20 V. The input resistor was a $0.0495\ \Omega$ resistor.

Table 4.11 - Data for Test Setup

Function Generator			MOSFET Driver (V)	Input Voltage (V)	Input Resistor (Ω)	Load Resistor (Ω)
Freq	Amplitude	Offset	11.8	20	0.0495	12000
100kHz	3.3V	1.65V				

Table 4.12 - Data from Second Test

Duty (%)	Cycle	Output Voltage (V)	Voltage Across		Power In (W)	Power (W)	Out Efficiency (%)
			Input (mV)	Resistor			
20		34.8	0.39		0.1575757576	0.10092	64.0453846
25		36.6	0.43		0.1737373737	0.11163	64.2521511
30		38.9	0.487		0.1967676768	0.12610083	64.0861524

				33	6
				0.14421333	64.5439421
35	41.6	0.553	0.2234343434	33	3
				0.16800083	64.3656443
40	44.9	0.646	0.261010101	33	5
					64.6378849
45	48.9	0.763	0.3082828283	0.1992675	9
				0.24210083	65.2012581
50	53.9	0.919	0.3713131313	33	6
55	60	1.125	0.4545454545	0.3	66
					66.7674823
60	67.8	1.42	0.5737373737	0.38307	9
					68.2670538
65	77.7	1.824	0.736969697	0.5031075	7
				0.68252083	69.0048636
70	90.5	2.448	0.9890909091	33	6
				0.99736333	
75	109.4	3.49	1.41010101	33	70.7299212
					73.1082480
80	138.3	5.396	2.18020202	1.5939075	1

As shown in Table 4.12 the efficiency for the booster is fairly low. Although, the current being drawn from the power supply was no more than 12 mA. With a low current and low power any power dissipation within the circuit can cause a dramatic drop in efficiency. If tested on a solar panel there would be an increase of power and current and most likely the efficiency would continue to improve. The efficiency losses in the circuit were calculated in Table 4.13 for an input voltage of 20V and a duty cycle of 20%.

Table 4.13 Power Losses Data

Power (W)							
Power In	MOSFET	Capacitors	Diode	Inductor	Power Out	Total Lost	Power
0.1575757 576	0.042144	0.003888	0.016 2	0.03045	0.10092	0.056655	

4.5 Testing - C2000 MPPT P&O Algorithm

The perturb and observe algorithm can be tested easily by monitoring the output of the PWM signals. To simulate the expected output from the voltage and current sensor, the input signal's voltage to ADC must be in the range of 0V to 3.3V. The testing strategy is to utilize two potentiometers which were supplied 3.3V input from the microcontroller. This setup will guarantee a maximum voltage of 3.3V to the ADC channel. Furthermore, the sampling voltages can be alternated upon the adjustment of the potentiometer. Based on the observation of the PWM signals, it is determined that the duty cycle changes with the changes of the sampled input from the ADC modules (i.e. changes in input current and voltage that leads to changes in input power). Thus the perturb and observe algorithm is indeed functional. However, the ADC section of the program is not working properly. With further testing and debugging, it is found that the ADC channel of the C2000 does not operate as expected. The expected reading from the ADC should be integers between 0 and 4095 because the C2000 has a 12bit ADC. However, the actual reading when using the Code Composer Debugger yields random negative numbers that are small in magnitude (i.e. -0.345, -1.0984 etc). Based on the testing and observation, the team recognizes the following potential problems with respect to the C2000 microcontroller.

When debugging the code using Code Composer Studio, it was found that the line `"PIE_setDebugIntVectorTable(myPie);"` was causing the compiler to give an error that the microcontroller's memory was out of available memory. Instinct is telling the team that this is the core issue why the ADC is not working as they may be interlinked because the ADC relies on an interrupt to initiate the ADC. The code was borrowed from a blogger source which claims the code works, but there were other areas found within the code to indicate otherwise. There simply was not enough time to debug this issue, and with sparse time left, the MSP430 solution was investigated.

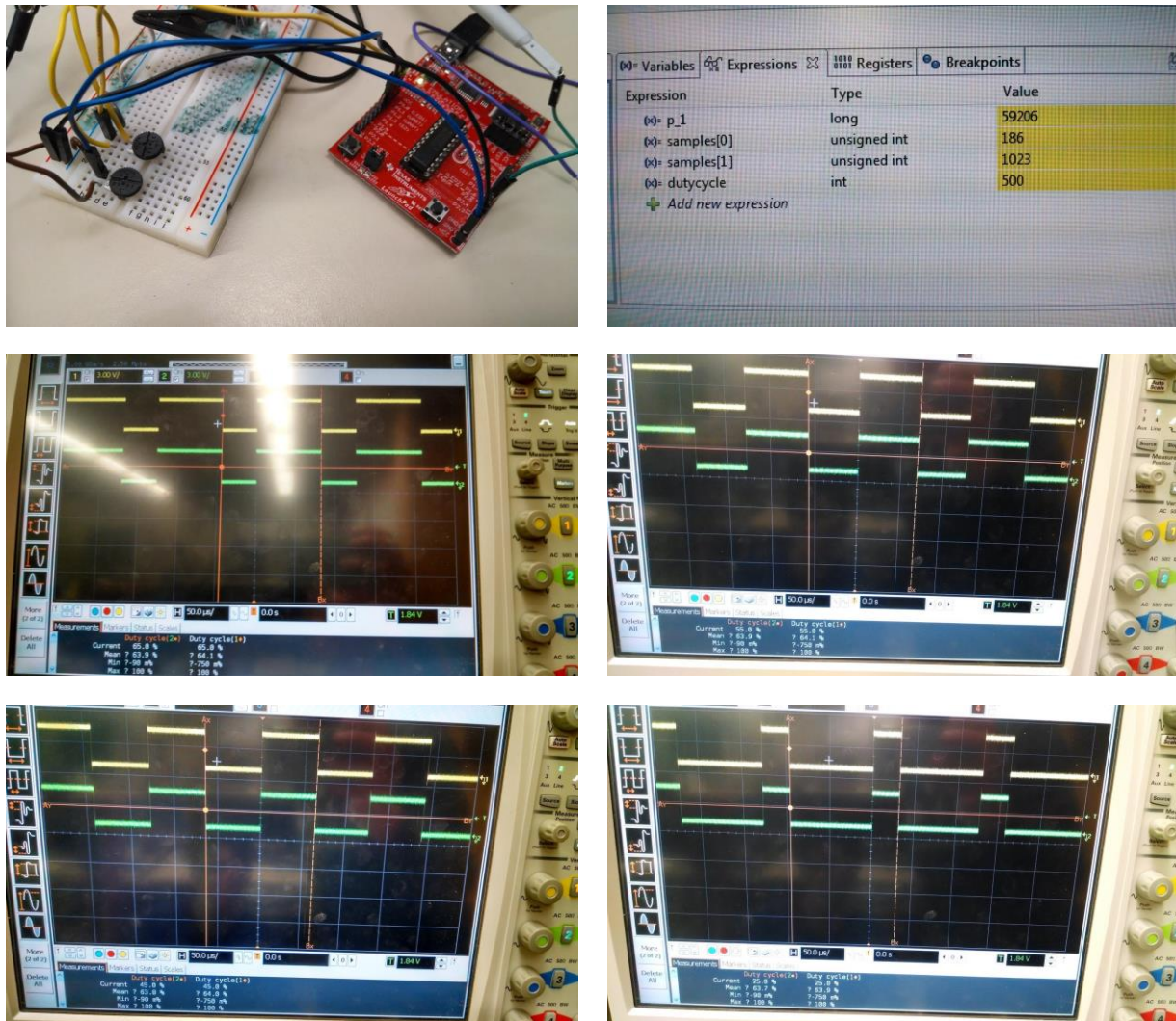
It should be noted that the microcontroller's PWM feature was working and was able to successfully drive the MOSFET driver with the C2000 Piccolo. When testing, the PWM would ramp up to its limited 90% duty cycle due to the ADC not working. The code has preventative measures to ensure the duty cycle does not fall below 10% or exceed 90% duty cycle.

4.6 Testing - MSP430 MPPT P&O Algorithm

Due to the fact that the Piccolo's ADC was not functional, the team approached the design expectation via an alternate method that is running MPPT program on the MSP430. Similar to the testing method of C2000, two potentiometers were used for each ADC channel, one to represent change in current and the other to represent change in voltage. Using the built-in debugger inside Code Composer Studio, the voltage being sampled through ADC could be read using the "Watch Expressions" feature. The variables being watched were 'P1', which is the instantaneous power, 'samples[0]' and 'samples[1]', which were the ADC values, and 'duty cycle', which was actually the pulse width. The period set in software was 1000, so a value of 500 for the 'duty cycle' would be 50%. This is shown in Figure 4.14. Also shown are varying duty cycles when the potentiometers were twisted (i.e. simulating the changing of the output parameters of the solar array). The code was modified to change the duty cycle in each

iteration an increment of 5% instead of the original 0.1% to better observe process of the perturb and observe algorithm. The bottom four pictures show varying duty cycles.

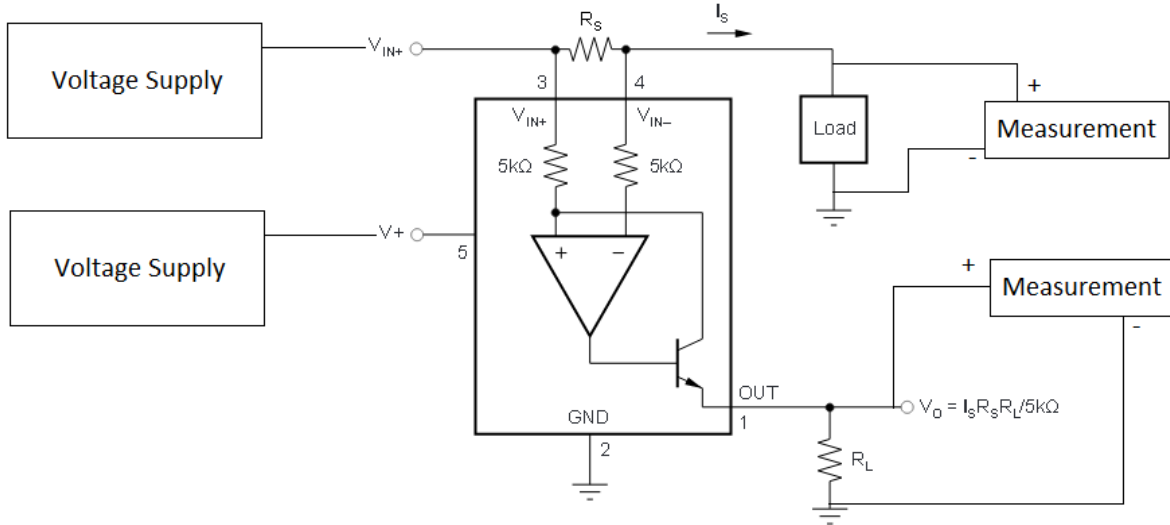
Figure 4.14



4.7 Testing - Current Sensor

The testing schematic of the current sensor module is shown in Figure 4.15.

Figure 4.15 - Current sensor Schematic



The load resistor, R_L , is held constant at a value of 10Ω . The voltage supply of the INA168 module, V_+ , is held constant at a value of $10V$. The sense resistor, R_s , is held constant at a value of 0.05Ω . The testing procedure is shown as follow:

- Step 1: Measure the actual resistance of R_L .
- Step 2: Measure the actual resistance of R_s .
- Step 3: Set V_{IN+} to desire voltage.
- Step 4: Measure the output voltage across R_L .
- Step 5: Calculate the current through R_L using measured resistance and voltage.
- Step 6: Measure the output voltage at R_L of the INA168 module.
- Step 7: Calculate the current sensed using the given formula in INA168 datasheet.
- Step 8: Calculate percent error.
- Step 9: Repeat step 3 through step 8 for different setup of R_L .

The testing data and plots are shown as follow:

Table 4.16 - Constant Variables For Testing

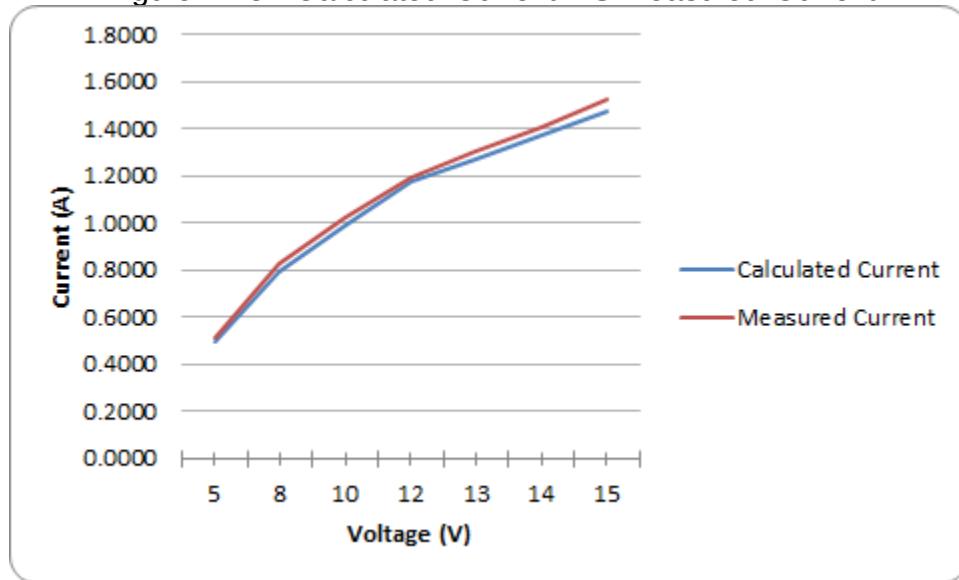
R_L Measured(Ω)	R_s Measured(Ω)	R_{INA168} Measured(Ω)
10.0640	0.0490	46345.0000

Table 4.17 - Data measured and calculated for current sensor testing

V_{IN+} (V)	R_L Measured(V)	R_s Measured(V)	Calculated Current	Measured Current	Error %
5	4.9746	0.2343	0.4943	0.5160	4.1990

8	7.9682	0.3757	0.7918	0.8273	4.2932
10	9.9241	0.4666	0.9861	1.0274	4.0229
12	11.8329	0.5429	1.1758	1.1953	1.6375
13	12.8100	0.5929	1.2729	1.3054	2.4903
14	13.8370	0.6409	1.3749	1.4111	2.5646
15	14.8270	0.6950	1.4733	1.5302	3.7221

Figure 4.18 - Calculated Current VS Measured Current



From the information above, it is observed that the current sensor can measure current accurately. This is essential for the MPPT because the current measurements will be used to calculate the input power. An inaccurate measurement may lead to the wrong determination from the perturb and observe algorithm.

4.8 Testing - MOSFET, Driver, Booster, Microcontroller, & Sensing (Entire Design)

This is the portion of testing where the team brought all of the working components together in order to try and get the maximum power point tracker functioning properly. First the shunt resistor of the current sensor was connected in series with the input voltage. Then the current sensor was powered with a 10 V source and controlled at a current of 10 mA while the MOSFET was powered with a 12V source. At this point everything was connected besides for the microcontroller. The microcontroller's pins were then connected to the required parts of the design. Namely the input of the gate driver. When the design was first tested the light bulb was used as the load and the circuit was powered with 15 V. The power supply's voltage began to

oscillate erratically once the supplies were turned on. These oscillations caused many different problems within the circuit which proved to be very difficult to troubleshoot.

After extensive troubleshooting, several problems were found within the design. The oscillations were actually causing current to surge back through some of the components. Most of the components can handle large amounts of current with the exception of the current sensor. The current sensor can withstand no more than 10mA to any of its pins. Once the current came back through the booster and into the sensor the pin shorted and caused the 12V supply of the sensor to drop dramatically to 1V. This was because a pin on the sensor was now trying to draw a lot more current than what it was controlled at. There were four current sensors ordered and ready for use. All of these began to malfunction because of the high current they received when troubleshooting the design. The team did not realize the current surge problems until after the initial four sensors were burnt out.

There were several reasons for the oscillations which were the cause of so many problems within the design. One reason being that the circuit wanted to draw a higher current than what the power supply would allow. When the DC/DC booster was first tested a 12k Ω resistance was used. This resistance was allowing the circuit to boost the voltage properly but it was not drawing enough current for the current sensor to measure anything substantial. For the design to be effective the shunt resistor needed to have 50 mV across it. The shunt resistance is so low that the only way to get 50 mV across it would be by drawing a high current from the source. This is the reason smaller resistances were then used as the load for the booster. Finding a low resistance load that can withstand a large amount of power is difficult but the light bulb seemed to be a viable option. Unfortunately, the light bulb was trying to draw too much current out of the supply, causing the voltage to drop and the power supply to oscillate. The resistance of the bulb was approximately 35 Ω so we needed to test our design on a more powerful voltage supply that could withstand a higher current and voltage.

Unfortunately the malfunctioning within the current sensors at the beginning of testing forced the team to purchase more sensors and waiting to configure the sensors was the only option. Once the new current sensors were configured it was much too close to our deadline for the team to try and use the high voltage power supplies in the facilitator's lab. The team considered using the power supply which Design Team 7 used but the minimum voltage was 50V, which was too high for the load. The 35 Ω resistance would draw a current above 3 amps, which is much higher than the 2.25 current limit of that power supply. Thus the team was unable to test the design with a fully functioning current sensor in series with the input voltage.

The team realized later that even if the current sensor had worked in unison with the booster and gate driver, testing the full design (using the microcontroller) with the higher voltage source still would have been ineffective. When using a voltage controlled power supply the current will change if the duty cycle changes. The team noticed that this is not the proper testing conditions needed when testing a maximum power point tracker. As stated previously a maximum power point tracker pulls a constant current from the solar panel and increases the duty cycle (and thus the voltage) until the current drops and the power is no longer maximum. The controller then decreases the duty cycle back to maximum. The controller then oscillates around the maximum power point. If a fully functioning maximum power point tracker was tested on a voltage

controlled power source then the current could change radically as the duty cycle is increased. This would again cause voltage and current fluctuations which could damage components, such as the current sensor. The design would need a power source that would generate a waveform similar to that of an I-V curve generated by a solar panel.

There are a few methods for MPPT testing which could have been implemented if there was more time allotted for the project. The first and most simple method would have been to utilize an actual solar panel for testing. The team did have access to a small solar array, but weather would not permit testing for a majority of the time anyway. The second method for testing MPPTs would be to purchase an actual solar panel simulator. The Agilent E4350B Solar Array Simulator is a viable solar panel simulator which would be able to produce the I-V needed. The major drawback to this method would be the expense. On average, a power supply with these capabilities currently costs \$5,000.00. This obviously would not be a realistic option. The best option for testing the MPPT would have been to use an actual solar array if the team would have been able to complete a viable prototype that functioned without the microcontroller.

As described, putting all of the components together was the most difficult part of the design. Lack of knowledge with current sensing set the team back and the only option from there was to play catch up. By the time the current sensor was functional the team was unable to test it on high voltage. The team also did not have time to acquire the proper testing supply for when the MPPT became fully operational. Although, the team is confident that through our efforts the solar car team will be able to build on the design and create a fully operational MPPT. The team has already provided proof that our design for the DC/DC booster and MOSFET driver circuit is functional. It could boost the voltage of the array to the necessary voltage required for the battery. Also the team has proven that our current sensing module is functional when tested on its own. The design is very close to operational and could be completed in three steps.

1. Use high voltage and high current supply in order to successfully test the functionality of the current sensor.
2. Successfully compile the code onto the microcontroller by conducting more research on the functionality of the C2000 Piccolo Microcontroller (emphasis on understanding the interrupt vector table and how it works with the ADC)
3. Using a solar panel in order to test the entire design and then troubleshoot MPPT accordingly.

Chapter 5 – Final cost, schedule, summary and conclusions:

5.1 Summary

Overall the project was a great learning experience for the team. Over the course of the semester the team was able to successfully build a DC/DC booster that was able to boost to the required voltage. A gate driver was successfully implemented to drive a MOSFET with the voltage required. The current sensor was configured so that it would be successful under the right test conditions. Unfortunately the trouble encountered when trying to implement the current sensor into the rest of the project created a delay in our schedule that could not be overcome. The microcontroller was loaded with the perturb and observe algorithm, but lack of knowledge with the C2000 Piccolo microcontroller caused problems when compiling the program and onto the controller and troubleshooting the errors.

If more time was provided, there are numerous ideas the team would implement to successfully build a maximum power point tracking system. The team would test the new current sensor, with the current booster circuit, using a high voltage/high current supply so that the proper voltage could be drawn through the shunt resistor. Then the team would bring the microcontroller into the circuit and test the MPPT with an actual solar panel or supply which could simulate an I-V curve. This method for testing is the only method possible when testing a maximum power point tracker. Unfortunately the team was not able to complete the overall design of the circuit in order to acquire the appropriate testing solutions.

If the project could be done over again there are many things the team would have done differently. There would have been more of a focus on implementing the current sensor before testing it with the microcontroller. Also the team would have emphasized researching what testing methods were needed for a maximum power point tracker. The team rushed trying to put all of the components together towards the end and failed to notice the flaws in our design and testing methods. Another method for building the tracker would have been to scale the entire project down to small voltage so that a proof of concept could have been established before trying to implement our design with high voltages and current which ultimately cause many of our components to malfunction. The team would have started with the Arduino or MSP430 in place of the C2000 and improve from there. The lack of experience with the C2000, compared with our previous knowledge of Arduino and MSP430 microcontrollers set the team back.

5.2 Final Cost and Budget

Items	Vendor	Amount	Order Number	Arriving Date	Sample ?	Cost(USD)	Budget Left (USD)
Toroid	Dexter Magnetics Technologies	2	716954	24-Sep	Yes	0	500
TI C2000 Piccolo Launchpad	Mouser	1	36140895	30-Sep	No	17.08	482.92
Capacitors	Digi-Key	2	47218057	9-Oct	No	8.66	474.26
Diodes	Digi-Key	2	47218057	9-Oct	No	26.94	447.32
MOSFET Driver	Digi-Key	5	47340754	21-Oct	No	11.9	435.42
Heat Sinks	SparkFun Electronics	4	47600004	10-Nov	No	6.24	429.18
Half Bridge Driver	Digi-Key	4	47600004	11-Nov	No	8.84	420.34
High Current Sensor	Digi-Key	4	47600004	11-Nov	No	17.56	402.78
Op-amp for sensor	Digi-Key	4	47600004	11-Nov	No	5.24	394.54
Diode	Digi-Key	4	47600004	11-Nov	No	0.72	396.82
MOSFET	Digi-Key	8	47600004	11-Nov	No	14.88	381.94
Capacitor	Digi-Key	6	47600004	11-Nov	No	8.82	373.12
TI-C2000 Piccolo Launchpad	Newark	2	25870140	20-Nov	No	34.16	339.12

Overall Cost: 160.88 USD

Budget Left : 339.12 USD

5.3 Final Thoughts

Lessons Learned:

- Make sure when testing a current sensor the proper voltage is held across the shunt resistor. Otherwise the sensor could give inaccurate readings.
- When designing a high voltage circuit, scale down the design and test on low voltages first. This provides a circuit that will be easier to troubleshoot and a proof of concept for the design. From there, scale the design up until the requirement is met.
- When building a new device the methods needed for testing are just as important as the design of the device itself. There is no point in implementing a design if the methods for testing are not appropriate.
- When planning a project, ensure an even more detailed schedule and make sure to meet as many deadlines as possible. Once a deadline is missed it becomes increasingly difficult to hit the projected completion date.
- When dealing with microcontrollers first use a controller that is familiar. Then if there is time, use a microcontroller which allows for optimizing device performance.
- Do not spend large amounts of time trying to build the perfect prototype. Build a prototype that works and then improve upon the project from that point. Getting ahead of yourself and trying to build a perfect project in one attempt can slow a project down and impede progress.
- When ordering parts always check the size of the component before ordering the part. Some components are very small and need to be soldered onto a dip package. The current sensor for this project was much smaller than expected and this caused delay in our project because the team had to brainstorm ways for implementing it into the circuit. Using a silver soldering paste, a needle, and steady hands, the chips were able to be soldered to a PCB. Depending on the characteristics of the PCB, they were either able to be inserted directly into a breadboard or required wires to be soldered to the pads. Numerous PCB boards and chips were ruined due to pins being soldered together. Each of these boards required testing using a multimeter to see whether the pins were touching each other and whether the pins were touching the PCB pads. Both cases were encountered in the process, often resulting in having to rend the chip and PCB useless due to being unable to fix it.

5.4 Schedule

	Task Mode	Task Name	Duration	Start	Finish	Predecessors
1		Design	18 days	Mon 9/15/14	Wed 10/8/14	
2		DC-DC boost converter	10 days	Mon 9/15/14	Fri 9/26/14	
3		Parameter Values Selection	5 days	Mon 9/15/14	Fri 9/19/14	
4		Procedure for Vdd of Microcontroller	2 days	Mon 9/22/14	Tue 9/23/14	3
5		Efficiency calculations	3 days	Wed 9/24/14	Fri 9/26/14	4
6		Microcontroller	2 days	Mon 9/29/14	Tue 9/30/14	
7		Choose Microcontroller	1 day	Mon 9/29/14	Mon 9/29/14	5
8		Study of Microcontroller	1 day	Tue 9/30/14	Tue 9/30/14	7
9		PCB layout	6 days	Wed 10/1/14	Wed 10/8/14	8
10		Cadence Simulation	6 days	Wed 10/1/14	Wed 10/8/14	
11		Final Propoasal	0 days	Wed 10/8/14	Wed 10/8/14	9
12		Simulation	12 days	Thu 10/9/14	Fri 10/24/14	11
13		DC-DC Boost Converter	12 days	Thu 10/9/14	Fri 10/24/14	
14		Labview Simulation efficiency	5 days	Thu 10/9/14	Wed 10/15/14	
15		Pspice simulation	7 days	Thu 10/16/14	Fri 10/24/14	14
16		Prepare Oral Presentation	3 days	Thu 10/9/14	Mon 10/13/14	11
17		Oral Presentation	0 days	Wed 10/15/14	Wed 10/15/14	16
18		Prepare Progress Report	4 days	Wed 10/15/14	Mon 10/20/14	17
19		Progress report	0 days	Wed 10/22/14	Wed 10/22/14	18
20		Prototype	29 days	Sat 10/25/14	Wed 12/3/14	12
21		DC-DC boost converter	2 days	Sat 10/25/14	Mon 10/27/14	
22		Low Voltage	1 day	Sat 10/25/14	Sat 10/25/14	12
23		High Voltage	1 day	Mon 10/27/14	Mon 10/27/14	22
24		First Demo	0 days	Mon 10/27/14	Mon 10/27/14	23
25		Gate Driver Solo Test	4 days	Tue 10/28/14	Fri 10/31/14	24
26		High-Speed Low-Side MOSFET	2 days	Tue 10/28/14	Wed 10/29/14	
27		Half Bridge Driver	2 days	Thu 10/30/14	Fri 10/31/14	26
28		Current Sensor Solo Testing	2 days	Mon 11/3/14	Tue 11/4/14	25
29		Write Code Microcontroller Piccolo	2 days	Wed 11/5/14	Thu 11/6/14	28
30		Troubleshooting	21 days	Fri 11/7/14	Fri 12/5/14	29
31		DC-DC boost converter	2 days	Fri 11/7/14	Mon 11/10/14	29
32		Gate Drive & DC-DC Test	2 days	Tue 11/11/14	Wed 11/12/14	31
33		Piccolo Microcontroller & Gate Driver	1 day	Thu 11/13/14	Thu 11/13/14	32
34		MSP430 Microcontroller & Gate Driver Test	1 day	Fri 11/14/14	Fri 11/14/14	33
35		DC-DC Boost Converter & Gate Driver & Microcontroller Tes	4 days	Mon 11/17/14	Thu 11/20/14	34
36		MPPT Test	11 days	Fri 11/21/14	Fri 12/5/14	35
37		Current Sensor Tuning	11 days	Fri 11/21/14	Fri 12/5/14	35
38		Prepare Design Issues Report	6 days	Tue 10/28/14	Tue 11/4/14	24
38		Prepare Design Issues Report	6 days	Tue 10/28/14	Tue 11/4/14	24
39		Prepare Progress Report	5 days	Tue 10/28/14	Mon 11/3/14	24
40		Desgin Issues	0 days	Tue 11/4/14	Tue 11/4/14	38
41		Progress report	0 days	Mon 11/3/14	Mon 11/3/14	39
42		Second Demo	0 days	Tue 11/4/14	Tue 11/4/14	38
43		Desgin for Poster	4 days	Wed 11/5/14	Mon 11/10/14	42
44		Finished Poster	0 days	Mon 11/10/14	Mon 11/10/14	43
45		Final Report	0 days	Wed 12/3/14	Wed 12/3/14	20
46		Project Revision	1 day	Thu 12/4/14	Thu 12/4/14	45
47		Design Day	0 days	Fri 12/5/14	Fri 12/5/14	46

Appendix 1 - Technical Roles



Daniel Chen

My technical role for the design project was to share responsibility with Brenton in the implementation and prototyping of the C2000 microcontroller. Two of the main expectations for the microcontroller were to sample input voltage and current, and adjust the output PWM with respect to the perturb and observe algorithm.

The ADC, PWM initialization, and programming required thorough research on the particular microcontroller we were using. Brenton and I have spent a significant amount of time to go over several initialization examples. These examples helped us further understand the capability of the microcontroller and also helped us better troubleshoot the design. One of the main challenges when programming the microcontroller was that the C2000 ADC channel was not working as expected. However, Brenton and I were able to understand the problems that caused the C2000 ADC to malfunction by shifting gears to a more familiar microcontroller, the MSP430. The process of researching and troubleshooting will serve as the most beneficial lesson in this project. Through the continuous process of debugging and refining, we improved the design and also enhanced our knowledge of the engineering design process.

In addition to the microcontroller, I have also been helping with the design and testing of the current and voltage sensors because those modules serve as the bridge between the microcontroller and the DC-DC Boost Converter.



Yue Guo - My technical role for the project was the PCB designer. Besides that, I have been working on the circuit building. My work includes the first double channel current sensor with the Op-Amp based voltage sensor in the project under the help from my group mates, as well as the soldering of small-size components under microscope by using silver paste, and repairing of the shorted or disconnected chipset.

The experience I have gained in this project is precious. The MPPT project is very different from the previous labs I had participated, i.e. ECE 203, ECE303, ECE331 and CSE 251. The major difference is that, in the projects those designed by teachers, I was guided by the lab manuals or lectures and as long as I do not do anything wrong, a decent score will be given. Also, even if I did the lab procedures incorrectly, there were always professors and TAs, who are willing to help me and guide me in the right direction. However, this is not how engineers work in real life.

The MPPT project has provided me a chance to observe, learn from, and also participate in a simulation of an industrial environment. In addition, as the lab coordinator, I had to consider how to spend the funds provided for us. Also, I started to notice details in the orders which I had neglected before. For example, I realized that by selecting the right package and type of the components, the prototyping experiences can go smoother. If the wrong component packaging was ordered I would have to spend hours making them usable for prototyping. This affected the efficiency of the entire testing and troubleshooting process.

As the PCB designer, I learned how to use Eagle to design PCBs. I did this by watching videos online and viewing examples which had been successfully created in the past. Since Eagle is possibly the most popular PCB design tool today, the specialty of using it can bring me more opportunities in my career as an Electrical Engineer.



Luis Kalaff - My technical role for the MPPT project was Hardware implementation. I also wrote a LabVIEW program that implement all equations needed to build a DC-DC Boost converter. Even though that building the DC-DC Boost converter was a team effort, integrating the MOSFET driver and the current sensor fell was something mainly done by Jake, Daniel and I. The first component I worked on with Jake was the MOSFET driver.

This part of the Hardware implementation was straightforward to understand for me because of my current enrollment in a mixed-signal circuit design class. In order for the MOSFET to let any current flow, the gate capacitance of the MOSFET needs to be charged. The pulse modulated signal from the microcontroller was not large enough to do so. Two different drivers were tested. First, the Dual Low-Side Driver was implemented. This driver required no additional circuit. Then the Half-Bridge Driver required bootstrapping and bypass capacitors, resistors, a diode and two MOSFETs. After doing some troubleshooting, Jake and I implemented the first one into the DC-DC Boost converter due to its simplicity and efficacy.

Also I help Jake and Daniel to troubleshoot the current sensor. There is a range for the voltage across the shunt resistor which determines if the current sensor is giving an accurate reading. Therefore the value of the shunt resistor had to be determined. However the accuracy of the sensor also depends on the current going through the DC-DC Boost converter. So I calculated the range of input current in which the current sensor would give accurate readings for both the final product and the testing set up implemented by the team.

Finally it last part of the prototyping was to implement all the parts of the MPPT together. The MOSFET driver was implemented successfully. However the current sensor gave a lot of problems. The current limit on the pins of the chip made troubleshooting very difficult because any issues with the rest of the circuit would make more current than needed flow to the chip.



Jacob Mills - My technical role for the project was Hardware implementation. Specifically the current sensor and the gate drivers. Building the DC/DC was mostly a team effort so implementing the gate driver and current sensor fell to Luis, Daniel, and I. First I implemented the gate driver into the system. This required a knowledge of what our MOSFET needed at its gate and of what to give the gate driver at its input in order to test. Luis and I used a signal similar to that of the PWM which would be sent from the microcontroller.

Second I calculated the resistance required for the shunt resistance and the load resistance which would both be connected directly to the current sensor. Then Daniel and I tested the current sensor by calculating what the voltage across the load resistor should be and then measuring it with the multimeter. This was a fairly simple process which was complicated when trying to implement it into the rest of our design.

Third was the implementation of all parts into the final design. This was the portion of the project which took the most time and effort. The gate driver implemented into the system without fail. The current sensor was a much larger issue. Problems with implementing the small chip being used as well as current flowing to the pins created several issues with the design.



Brenton Sirowatka - My technical role for the project was to implement the microcontroller tracking using the C2000 microcontroller. This required acquainting myself user guides, example C code, and the fundamentals of P&O tracking. I also invest invested some time with MSP430, of which I had a plethora of previous experience with. Dan and I were able to successfully measure the PWM signal generated by each microcontroller tested.

I was able to get the MOSFET driver working that is currently used in the team's system as well as integrating it with the MOSFET, and later with the booster circuit. My resourcefulness allowed the testing of the booster circuit early in the design process by using the function generator to drive the MOSFET in place of the microcontroller as well as using potentiometers to test the microcontroller's ADC. Using PSpice, I simulated the booster circuit using ideal components calculated by Luis in LabVIEW to verify the component characteristics had merit.

My nontechnical role involved reviewing the documents to ensure consistent, quality writing. I also implemented a folder structure and maintained clean file organization in Google Drive to coordinate document collaboration.

Appendix 2 - Bibliography

1. "The History Highlight of Solar Sells (Photovoltaic Cells)." *Ntnu*. N.p., n.d. Web. 03 Dec. 2014. <<http://org.ntnu.no/solarcells/pages/history.php>>.
2. Toothman, Jessika, and Scott Aldous. "How Solar Cells Work." *HowStuffWorks*. HowStuffWorks.com, n.d. Web. 02 Dec. 2014. <<http://science.howstuffworks.com/environmental/energy/solar-cell.htm>>.
3. "Smaller, Cheaper, Faster: Does Moore's Law Apply to Solar Cells? | Guest Blog, Scientific American Blog Network." *Scientific American Global RSS*. N.p., n.d. Web. 03 Dec. 2014. <<http://blogs.scientificamerican.com/guest-blog/2011/03/16/smaller-cheaper-faster-does-moores-law-apply-to-solar-cells/>>.
4. Knier, Gil. "How Do Photovoltaics Work?" *NASA Science*. NASA, n.d. Web. 02 Dec. 2014. <<http://science.nasa.gov/science-news/science-at-nasa/2002/solarcells/>>.
5. Maehlum, Mathias. "Which Solar Panel Type Is Best? Mono-, Polycrystalline or Thin Film?" *Energy Informative*. N.p., n.d. Web. 03 Dec. 2014. <<http://energyinformative.org/best-solar-panel-monocrystalline-polycrystalline-thin-film/>>.
6. "HighTechScience.org Installs Solar Train System at Museum." *HighTechScience.org Installs Solar Train System at Museum*. N.p., n.d. Web. 03 Dec. 2014. <http://www.hightechscience.org/solar_express.htm>.
7. 2598 Fortune Way, Suite K • Vista, CA 92081 • Phone 760-597-1642 • Fax 760-597-1731 • *Www.blueskyenergyinc.com What Is Maximum Power Point Tracking (MPPT) and How Does It Work?* N.p.: Blue Sky Energy, n.d. PDF.
8. "Basics of MPPT Solar Charge Controller." *LEONICS*. LEONICS, n.d. Web. 3 Dec. 2014. <[http%3A%2F%2Fwww.leonics.com%2Fsupport%2Farticle2_14j%2Farticles2_14j_en.php](http%3A%2F%2Fwww.leonics.com%2Fsupport%2Farticle2_14j%2Farticles2_14j_en.php)>.
9. "PCM -INDEX-" *PCM POWERCOM*. N.p., n.d. Web. 03 Dec. 2014. <<http://pcmups.com.tw/product/solar/PPV175M6.html>>.
10. Mendick, Robert. "Eric Pickles' Ruling Kills off Large-scale Solar Farms." *The Telegraph*. Telegraph Media Group, 08 June 2014. Web. 03 Dec. 2014. <<http://www.telegraph.co.uk/earth/environment/10883759/Eric-Pickles-ruling-kills-off-large-scale-solar-farms.html>>.
11. "Why Is Fiber Optic Cable the Express Highway of Communication? - RP Companies, Inc." *RP Companies Inc.* N.p., n.d. Web. 03 Dec. 2014. <<http://rpcompaniesinc.com/telecommunication/why-is-fiber-optic-cable-the-express-highway-of-communication/>>.
12. Hauke, Brigitte. *Basic Calculation of a Boost Converter ' S Power Stage*. N.p.: Texas Instruments, n.d. PDF.
13. "Capacitors and ESR." - *Transwiki*. N.p., n.d. Web. 03 Dec. 2014. <http://wiki.xtronics.com/index.php/Capacitors_and_ESR>.
14. Hovens, M.G.P.; Prevoo, Y.K.L.M. "Maximum Power Point Tracking Topology, Sensor and Switch Design." *TUDeft*. N.p., n.d. Web. 3 Dec. 2014. <<http://discover.tudelft.nl:8888/recordview/view?recordId=TUD%3Aoai%3Atudelft.nl%3Auuid%3A4f8b71a1-7766-462f-b6ec-6b1605bb5218&language=en>>.

15. Freeman, Dave. *Introduction to Photovoltaic Systems Maximum Power Point Tracking*. N.p.: Texas Instruments, n.d. PDF.
16. Ahmed M. Atallah, Almoataz Y. Abdelaziz, and Raihan S. Jumaah. *IMPLEMENTATION OF PERTURB AND OBSERVE MPPT OF PV SYSTEM WITH DIRECT CONTROL METHOD USING BUCK AND BUCK - BOOST CONVERTERS*. N.p.: AIRCC Publishing Corporation, 1 Feb. 2014. PDF.
17. "C2000 Solar MPPT Tutorial Covering the Electronics & C Code." *CoderTronics*. N.p., n.d. Web. 03 Dec. 2014. <<http://coder-tronics.com/c2000-solar-mppt-tutorial-pt1/>>.
18. "Simple MPPT with MSP430." *MSP430™ Microcontroller Projects*. N.p., n.d. Web. 03 Dec. 2014. <<http://e2e.ti.com/group/launchyourdesign/m/msp430microcontrollerprojects/665420.aspx>>.

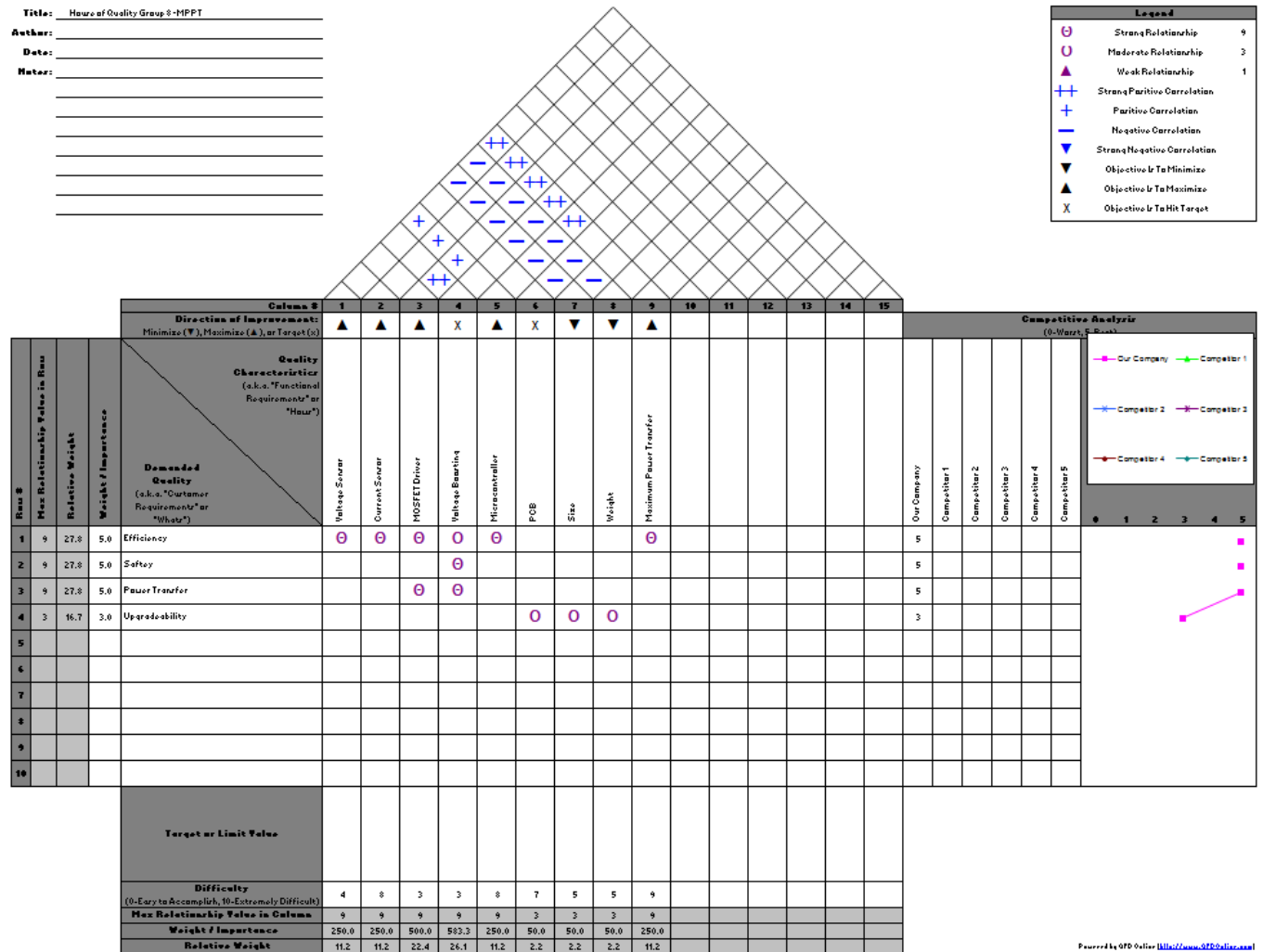
House of Quality:

Title: House of Quality Group 8 - MPPT

Author:

Date:

Name:



A3.1 C2000 MPPT Code

A few changes were made to get rid of errors. The source code was taken from (17).

```
// Device Headerfile and Examples Include File
#include <stdio.h>
#include "DSP28x_Project.h"
#include "f2802x_common/include/clk.h"
#include "f2802x_common/include/flash.h"
#include "f2802x_common/include/gpio.h"
#include "f2802x_common/include/pie.h"
#include "f2802x_common/include/pll.h"
#include "f2802x_common/include/adc.h"
#include "f2802x_common/include/pwm.h"
#include "f2802x_common/include/timer.h"
#include "f2802x_common/include/wdog.h"

// Function prototype for interrupt handler
interrupt void cpu_timer0_isr(void);

// Function prototypes
void init_system(void); // Initialise all handles, disable watchdog, clocks, disable the
                        // PIE and all interrupts
void init_Gpio(void); // Setup Gpio pins
void init_Timer(void); // Setup timers
void init_PIE(void); // Enable PIE and register ISR handlers
void init_ADC(void); // Setup and initialise ADC module
void Init_Pwm(void); // Setup and initialise PWM module
void Data_Update(void); // Update ADC values
void Adj_PWM(void); // Update PWM duty cycle

// Constants
const float ADC_12bit = 0.0008056; // 12bit ADC voltage resolution
const float IP_Volt_Const = 0.0517241379; // Calibrated constant input voltage
const float IP_Amp_Const = 0.5; // Calibrated constant input current
const float OP_Volt_Const = 0.0242460083; // Calibrated constant output voltage
const float OP_Amp_Const = 1; // Calibrated constant output current

// MPPT Global variables *Most can be placed inside the functions but for debugging
// and viewing the state in CCS they need to be Global
float IP_Amp;
float IP_Volt;
float Old_IP_Volt;
float OP_Amp;
float OP_Volt;
float New_PW_In;
float Old_PW_In;
```

```

float New_PW_Out;
int SysTick = 0;    // Used to control the PWM update frequency
int PWM_Temp_Temp;  // Used for graphing data
int Duty_Cycle;     // Used for graphing data

// System handles
CLK_Handle myClk;
FLASH_Handle myFlash;
GPIO_Handle myGpio;
ADC_Handle myAdc;
PIE_Handle myPie;
TIMER_Handle myTimer0, myTimer1;
CPU_Handle myCpu;
PLL_Handle myPll;
WDOG_Handle myWDog;
PWM_Handle myPwm1, myPwm2;

void main(void) {
    init_system();
    init_Gpio();
    init_Timer();
    init_PIE();
    init_ADC();

    // Disables the ePWM module time base clock sync signal
    CLK_disableTbClockSync(myClk);
    Init_Pwm();
    // Enables the ePWM module time base clock sync signal
    CLK_enableTbClockSync(myClk);

    Data_Update(); // initial read of current power in

    while(1)
    {
        if (SysTick == 1)
        {
            Data_Update();
            Adj_PWM();
            SysTick = 0;
        }
    }
}

// Function declaration for the interrupt handler
interrupt void cpu_timer0_isr(void)
{
    SysTick = 1;
}

```

```

        PIE_clearInt(myPie, PIE_GroupNumber_1);
    }

// Function declarations
void init_system(void)
{
    // Initialise all the handles needed for this application
    myPwm1 = PWM_init((void *)PWM_ePWM1_BASE_ADDR, sizeof(PWM_Obj));
    myPwm2 = PWM_init((void *)PWM_ePWM2_BASE_ADDR, sizeof(PWM_Obj));
    myAdc = ADC_init((void *)ADC_BASE_ADDR, sizeof(ADC_Obj));
    myClk = CLK_init((void *)CLK_BASE_ADDR, sizeof(CLK_Obj));
    myCpu = CPU_init((void *)NULL, sizeof(CPU_Obj));
    myFlash = FLASH_init((void *)FLASH_BASE_ADDR, sizeof(FLASH_Obj));
    myGpio = GPIO_init((void *)GPIO_BASE_ADDR, sizeof(GPIO_Obj));
    myPie = PIE_init((void *)PIE_BASE_ADDR, sizeof(PIE_Obj));
    myPll = PLL_init((void *)PLL_BASE_ADDR, sizeof(PLL_Obj));
    myTimer0 = TIMER_init((void *)TIMER0_BASE_ADDR, sizeof(TIMER_Obj));
    myTimer1 = TIMER_init((void *)TIMER1_BASE_ADDR, sizeof(TIMER_Obj));
    myWDog = WDOG_init((void *)WDOG_BASE_ADDR, sizeof(WDOG_Obj));

    // Disables the watchdog (WDOG) timer
    WDOG_disable(myWDog);
    // Enables the ADC clock
    CLK_enableAdcClock(myClk);
    // Calibrates the ADC and internal oscillators
    (*Device_cal)();
    // Sets the internal oscillator 1 as the clock source
    CLK_setOscSrc(myClk, CLK_OscSrc_Internal);
    // Setup the PLL for x12 /1 which will yield 60Mhz = 10Mhz * 12 / 2
    PLL_setup(myPll, PLL_Multiplier_12, PLL_DivideSelect_ClkIn_by_2);
    // Disables the peripheral interrupt expansion (PIE)
    PIE_disable(myPie);
    // Disables all of the interrupts
    PIE_disableAllInts(myPie);
    // Disables global interrupts
    CPU_disableGlobalInts(myCpu);
    // Clears all interrupt flags
    CPU_clearIntFlags(myCpu);
    // Enable Global realtime interrupt DBGMC
    CPU_enableDebugInt(myCpu);
    // If running from flash copy RAM only functions to RAM
    #ifdef _FLASH
    memcpy(&RamfuncsRunStart, &RamfuncsLoadStart,
(size_t)&RamfuncsLoadSize);
    #endif
}

```

```

void init_Gpio()
{
    // Initialize GPIO for EPWM1A
    GPIO_setPullUp(myGpio, GPIO_Number_0, GPIO_PullUp_Disable);
    //GPIO_setPullUp(myGpio, GPIO_Number_1, GPIO_PullUp_Disable); // Note
used
    GPIO_setMode(myGpio, GPIO_Number_0, GPIO_0_Mode_EPWM1A);
    //GPIO_setMode(myGpio, GPIO_Number_1, GPIO_1_Mode_EPWM1B);
// Note used
    // Initialize GPIO for EPWM2A
    GPIO_setPullUp(myGpio, GPIO_Number_2, GPIO_PullUp_Disable);
    //GPIO_setPullUp(myGpio, GPIO_Number_3, GPIO_PullUp_Disable); // Note
used
    GPIO_setMode(myGpio, GPIO_Number_2, GPIO_2_Mode_EPWM2A);
    //GPIO_setMode(myGpio, GPIO_Number_3, GPIO_3_Mode_EPWM2B);
// Note used
}

void init_Timer()
{
    // Sets the timer (TIMER) period
    TIMER_setPeriod(myTimer0, 6000000);
    // Sets the timer (TIMER) prescaler
    TIMER_setPreScaler(myTimer0, 0);
    // Reloads the timer (TIMER) value
    TIMER_reload(myTimer0);
    // Sets the timer (TIMER) emulation mode
    TIMER_setEmulationMode(myTimer0, TIMER_EmulationMode_RunFree);
    // Enables the timer (TIMER) interrupt
    TIMER_enableInt(myTimer0);
    //Starts the timer (TIMER)
    TIMER_start(myTimer0);
}

void init_PIE(void)
{
    // Initializes the vector table with Debug interrupt handlers
    //PIE_setDebugIntVectorTable(myPie); //Gives "Not enough space on uC "error
    // Enables the peripheral interrupt expansion (PIE)
    PIE_enable(myPie);
    // Registers a handler for a PIE interrupt
    PIE_registerPieIntHandler(myPie, PIE_GroupNumber_1,
PIE_SubGroupNumber_7, (intVec_t)&cpu_timer0_isr);
    // Enables a specified interrupt number
    CPU_enableInt(myCpu, CPU_IntNumber_1);
    // Enables the Cpu Timer 0 interrupt

```

```

    PIE_enableTimer0Int(myPie);
    // Enables global interrupts
    CPU_enableGlobalInts(myCpu);
    // Enables the debug interrupt
    CPU_enableDebugInt(myCpu);
}

void init_ADC(void)
{
    // Enables the ADC band gap circuit
    ADC_enableBandGap(myAdc);
    // Enables the ADC reference buffers circuit
    ADC_enableRefBuffers(myAdc);
    // Powers up the ADC
    ADC_powerUp(myAdc);
    // Enables the ADC
    ADC_enable(myAdc);
    // ADCCTL1 - ADC_ADCCTL1_ADCREFSEL_BITS
    ADC_setVoltRefSrc(myAdc, ADC_VoltageRefSrc_Int);
    // Clear ADCINT1 flag reinitialize for next SOC
    ADC_clearIntFlag(myAdc, ADC_IntNumber_1);
    // Enables the specified ADC interrupt
    PIE_enableAdcInt(myPie, ADC_IntNumber_1);
    // Enables the ADC interrupt
    ADC_enableInt(myAdc, ADC_IntNumber_1);
    // Sets the interrupt pulse generation mode
    ADC_setIntPulseGenMode(myAdc, ADC_IntPulseGenMode_Prior);
    // Enables ADC interrupt
    ADC_enableInt(myAdc, ADC_IntNumber_1);
    // Sets the interrupt mode
    ADC_setIntMode(myAdc, ADC_IntNumber_1, ADC_IntMode_ClearFlag);
    // Sets the interrupt source
    ADC_setIntSrc(myAdc, ADC_IntNumber_1, ADC_IntSrc_EOC7);
    // Sets the start-of-conversion (SOC) channel number
    ADC_setSocChanNumber (myAdc, ADC_SocNumber_0,
ADC_SocChanNumber_A0);
    ADC_setSocChanNumber (myAdc, ADC_SocNumber_1,
ADC_SocChanNumber_A1);
    ADC_setSocChanNumber (myAdc, ADC_SocNumber_2,
ADC_SocChanNumber_A2);
    ADC_setSocChanNumber (myAdc, ADC_SocNumber_3,
ADC_SocChanNumber_A3);
    ADC_setSocChanNumber (myAdc, ADC_SocNumber_4,
ADC_SocChanNumber_A0);
    ADC_setSocChanNumber (myAdc, ADC_SocNumber_5,
ADC_SocChanNumber_A1);

```

```

        ADC_setSocChanNumber (myAdc, ADC_SocNumber_6,
ADC_SocChanNumber_A2);
        ADC_setSocChanNumber (myAdc, ADC_SocNumber_7,
ADC_SocChanNumber_A3);
        // Sets the start-of-conversion (SOC) trigger source
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_0,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_1,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_2,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_3,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_4,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_5,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_6,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        ADC_setSocTrigSrc(myAdc, ADC_SocNumber_7,
ADC_SocTrigSrc_EPWM1_ADCSOCA);
        // Sets the start-of-conversion (SOC) sample delay
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_0,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_1,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_2,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_3,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_4,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_5,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_6,
ADC_SocSampleWindow_55_cycles);
        ADC_setSocSampleWindow(myAdc, ADC_SocNumber_7,
ADC_SocSampleWindow_55_cycles);
    }

void Init_Pwm()
{
    // Enables the pwm clock
    CLK_enablePwmClock(myClk, PWM_Number_1);
    // Enables the pulse width modulation (PWM) start of conversion (SOC) A pulse
generation
    PWM_enableSocAPulse(myPwm1);

```

```

        // Enables the pulse width modulation (PWM) start of conversion (SOC) A pulse
generation
        PWM_setSocAPulseSrc(myPwm1,
PWM_SocPulseSrc_CounterEqualCmpAIncr);
        // Sets the pulse width modulation (PWM) start of conversion (SOC) A interrupt
period
        PWM_setSocAPeriod(myPwm1, PWM_SocPeriod_FirstEvent);
        // TBCTL Time-Base Control Register - syncmode
PWM_TBCTL_SYNCSEL_BITS
        PWM_setSyncMode(myPwm1, PWM_SyncMode_CounterEqualZero);
        // TBPRD Time Base Period Register
        PWM_setPeriod(myPwm1, 150); // 60M /
        // Sets the pulse width modulation (PWM) phase
        PWM_setPhase(myPwm1, 0x0000);
        // Sets the pulse width modulation (PWM) count
        PWM_setCount(myPwm1, 0x0000);
        // Sets the pulse width modulation (PWM) counter mode
        PWM_setCounterMode(myPwm1, PWM_CounterMode_UpDown);
        // Disables the pulse width modulation (PWM) counter loading from the phase
register
        PWM_disableCounterLoad(myPwm1);
        // Sets the pulse width modulation (PWM) high speed clock divisor
        PWM_setHighSpeedClkDiv(myPwm1, PWM_HspClkDiv_by_2);
        // Sets the pulse width modulation (PWM) clock divisor
        PWM_setClkDiv(myPwm1, PWM_ClkDiv_by_1);
        // Writes the pulse width modulation (PWM) data value to the Counter Compare
A hardware
        PWM_setCmpA(myPwm1, 75);
        // Sets the pulse width modulation (PWM) object action for PWM A when the
counter equals CMPA and the counter is incrementing
        PWM_setActionQual_CntUp_CmpA_PwmA(myPwm1, PWM_ActionQual_Set);
        // Sets the pulse width modulation (PWM) object action for PWM A when the
counter equals CMPA and the counter is decrementing
        PWM_setActionQual_CntDown_CmpA_PwmA(myPwm1,
PWM_ActionQual_Clear);
        // Sets the pulse width modulation (PWM) object action for PWM B when the
counter equals CMPA and the counter is incrementing
        PWM_setActionQual_CntUp_CmpA_PwmB(myPwm1, PWM_ActionQual_Clear);
        // Sets the pulse width modulation (PWM) object action for PWM B when the
counter equals CMPA and the counter is decrementing
        PWM_setActionQual_CntDown_CmpA_PwmB(myPwm1,
PWM_ActionQual_Set);

        // Enables the pwm clock
        CLK_enablePwmClock(myClk, PWM_Number_2);
        // Sets the pulse width modulation (PWM) sync mode
        PWM_setSyncMode(myPwm2, PWM_SyncMode_EPWMxSYNC);

```

```

    // Sets the pulse width modulation (PWM) period
    PWM_setPeriod(myPwm2, 1000);
    // Enables the pulse width modulation (PWM) counter loading from the phase
register
    PWM_enableCounterLoad(myPwm2);
    // Sets the pulse width modulation (PWM) phase
    PWM_setPhase(myPwm2, 1000); // To calc
phase 2000 = 180deg therefore 2000 /180 = 11.11 so for 120deg = 120*11.11 = 1333.2
    // Sets the pulse width modulation (PWM) phase direction
    PWM_setPhaseDir(myPwm2, PWM_PhaseDir_CountUp);
    // Sets the pulse width modulation (PWM) counter mode
    PWM_setCounterMode(myPwm2, PWM_CounterMode_UpDown);
    // Sets the pulse width modulation (PWM) high speed clock divisor
    PWM_setHighSpeedClkDiv(myPwm2, PWM_HspClkDiv_by_2);
    // Sets the pulse width modulation (PWM) clock divisor
    PWM_setClkDiv(myPwm2, PWM_ClkDiv_by_1);
    // Writes the pulse width modulation (PWM) data value to the Counter Compare
A hardware
    PWM_setCmpA(myPwm2, 500);
    // Sets the pulse width modulation (PWM) object action for PWM A when the
counter equals CMPA and the counter is incrementing
    PWM_setActionQual_CntUp_CmpA_PwmA(myPwm2, PWM_ActionQual_Set);
    // Sets the pulse width modulation (PWM) object action for PWM A when the
counter equals CMPA and the counter is decrementing
    PWM_setActionQual_CntDown_CmpA_PwmA(myPwm2,
PWM_ActionQual_Clear);
    // Sets the pulse width modulation (PWM) object action for PWM B when the
counter equals CMPA and the counter is incrementing
    PWM_setActionQual_CntUp_CmpA_PwmB(myPwm2, PWM_ActionQual_Clear);
    // Sets the pulse width modulation (PWM) object action for PWM B when the
counter equals CMPA and the counter is decrementing
    PWM_setActionQual_CntDown_CmpA_PwmB(myPwm2,
PWM_ActionQual_Set);
}

void Data_Update(void)
{
    float ADC_A0, ADC_A1, ADC_A2, ADC_A3;
    int sum_of_ADC_samples_Array[4] = { 0 };
    int numberOfSamples = 128;
    int i = 0;

    for (i = 0; i < numberOfSamples; i++) {
        while (AdcRegs.ADCINTFLG.bit.ADCINT1 == 0) {}
        sum_of_ADC_samples_Array[0] += AdcResult.ADCRESULT0;
        sum_of_ADC_samples_Array[1] += AdcResult.ADCRESULT1;
        sum_of_ADC_samples_Array[2] += AdcResult.ADCRESULT2;
    }
}

```

```

    sum_of_ADC_samples_Array[3] += AdcResult.ADCRESULT3;

    sum_of_ADC_samples_Array[0] += AdcResult.ADCRESULT4;
    sum_of_ADC_samples_Array[1] += AdcResult.ADCRESULT5;
    sum_of_ADC_samples_Array[2] += AdcResult.ADCRESULT6;
    sum_of_ADC_samples_Array[3] += AdcResult.ADCRESULT7;
    AdcRegs.ADCINTFLGCLR.bit.ADCINT1 = 1;
}
// divide by the number of samples to find the average value
ADC_A0 = sum_of_ADC_samples_Array[0] / numberOfSamples;
ADC_A1 = sum_of_ADC_samples_Array[1] / numberOfSamples;
ADC_A2 = sum_of_ADC_samples_Array[2] / numberOfSamples;
ADC_A3 = sum_of_ADC_samples_Array[3] / numberOfSamples;

// Calculate values read on ADC GPIO
ADC_A0 = (ADC_12bit * ADC_A0);
ADC_A1 = (ADC_12bit * ADC_A1);
ADC_A2 = (ADC_12bit * ADC_A2);
ADC_A3 = (ADC_12bit * ADC_A3);

// Calculate correct circuit readings
IP_Volt = (ADC_A0 / IP_Volt_Const);
IP_Amp = (ADC_A1 / IP_Amp_Const);
OP_Volt = (ADC_A2 / OP_Volt_Const);
OP_Amp = (ADC_A3 / OP_Amp_Const);

// Calculate Power In and Power Out
New_PW_In = IP_Volt * IP_Amp;
New_PW_Out = OP_Volt * OP_Amp;
}

void Adj_PWM(void)
{
    int PWM_Temp;
    PWM_Temp = EPwm1Regs.CMPA.half.CMPA; // Assign the
current duty cycle value of PWM1 CMPA to PWM_Temp
    PWM_Temp_Temp = PWM_Temp; // Used for
graphing
    Duty_Cycle = 100 - ((PWM_Temp_Temp / 1000) * 100); // Used for graphing

    if (New_PW_In > Old_PW_In) { // New power larger than old power
        if (IP_Volt > Old_IP_Volt) // New PV volts greater than old
        {
            PWM_Temp = PWM_Temp + 2; // Increase Duty Cycle
        }
        else // New PV volts less than old

```

```

        {
            PWM_Temp = PWM_Temp - 2;          // Decrease Duty Cycle
        }
    }
    else {                                     // New power less than old
power
        if (IP_Volt > Old_IP_Volt)           // New PV volts greater than old
        {
            PWM_Temp = PWM_Temp - 2;          // Decrease Duty Cycle
        }
        else                                 // New PV volts less than old
        {
            PWM_Temp = PWM_Temp + 2;          // Increase Duty Cycle
        }
    }

    if (PWM_Temp < 15) {
        PWM_Temp = 15;                       // Necessary to prevent
too smaller PWM value
    }
    if (PWM_Temp > 135) {
        PWM_Temp = 135;                     // Necessary to prevent too
greater PWM value
    }
    EPwm1Regs.CMPA.half.CMPA = PWM_Temp;    // Adjust PWM1 duty cycle
    EPwm2Regs.CMPA.half.CMPA = PWM_Temp;    // Adjust PWM2 duty cycle

    Old_IP_Volt = IP_Volt;                   // Assign new input volts value to old
input volts value
    Old_PW_In = New_PW_In;                  // Assign new power value to old
power value
}

```

A3.2 MSP430 MPPT Code

The source code was taken from (18).

```

#include "msp430G2553.h"

/* Vpv max ADC = 1023 <-> 24V
 * Ipv max ADC = 1023 <-> 2A
 */

#define TPWM 75    // Period of PWM

```

```

void mppt_compute(void);
void ADC10_ISR(void);

unsigned int samples[2], loop = 0; //for holding the conversion results
long int p_0 = 0, p_1 = 0;
void main(void){
    WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
    //Configure Timer CPU
    BCSCTL1 = CALBC1_16MHZ;
    DCOCTL = CALDCO_16MHZ;

    //Configure PWM
    //Freq PWM = Freq clock /(x*TACCR0) x=2 if UP-DOWN, else x=1
    TA0CTL = TASSEL_2 + MC_3 + ID_0 ;// SMCLK, up-down mode, Timer A input
    divider /1
    TACCR0 = TPWM;                       // PWM Period

    TA0CCTL1 = OUTMOD_2;                 // Timer_A.OUT1 PWM Toggle/reset
    TACCR1 = 37;                         // PWM Duty Cycle

    TACCR2 = 1000-1;                     // Period of Timer_A.OUT2 to trigger
    ADC
    TA0CCTL2 |= OUTMOD_6;                // Timer_A.OUT2 Toggle/Reset

    P1DIR |= BIT2 + BIT6;                // P1.2 and P1.6 -> output
    P1SEL |= BIT2;                       // P1.2 = TA1 output
    P1SEL &= ~BIT6;                      // P1.6 -> I/O

    //Configure ADC
    ADC10CTL0 &= ~ENC;                   // Disable ADC
    ADC10CTL1 = INCH_1 + CONSEQ_3 + SHS_3 + ADC10DIV_0 + ADC10SSEL_3;
    // A4 A3 - A1 A0 -> Multi channel
    // Repeat sequence of channels
    // Timer_A.Output 2 trigger ADC
    ADC10CTL0 = SREF_1 + ADC10SHT_2 + MSC+ REFON + ADC10ON + ADC10IE;
    // VR+ = VREF+ and VR- = VSS
    // 16 x ADC10CLKs, Multi channel enabled

    ADC10AE0 = BIT0 + BIT1;              // P1.0 -> A0 (sample[1] -> lpv),
    P1.1 -> A1 (sample[0] -> Vpv)
    ADC10DTC1 = 0x02;                    // 2 conversions
    ADC10SA = (int)samples;              // Buffer
    ADC10CTL0 |= ENC+ADC10SC;            // Sampling and conversion
    ready

    __enable_interrupt();

```

```

    for (;;)    {
        // other code

    }
}

void mppt_compute(void){
    p_1 = samples[0] * samples[1];           // compute de new power
    if(p_1 > p_0)
        TACCR1 = ( TACCR1 >= (TPWM -1) ) ? TACCR1 : + (TACCR1+1); //
Increment duty cycle of PWM with up saturation
    else if (p_1 < p_0)
        TACCR1 = ( TACCR1 <= 1 ) ? TACCR1 : TA0CCR1 - 1;           //
Decrement duty cycle of PWM with low saturation
    p_0 = p_1;
        // Storage new power
    return;
}

#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void){
    loop++;
    if(loop >= 3906){
        mppt_compute();
        loop = 0;
    }
    ADC10CTL0 &= ~ENC;           // Disable ADC
    while (ADC10CTL1 & ADC10BUSY); // Wait if ADC10 core is active
    ADC10SA = (int)samples;      // Data buffer start
    ADC10CTL0 |= ENC+ADC10SC;    // Enable and start ADC
    ADC10CTL0 &= ~ADC10IFG;      // Clear interrupt

    return;
}

```