

Maximum Power Point Tracking

A Major Qualifying Project Report:

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Yejee Choi

Monineath Khun

Giselle Verbera

Date: March 23th, 2017

Approved:

Professor Alexander Emanuel, Advisor

Acknowledgements

First and foremost we would like to thank our advisor, Professor Alexander Emanuel for assisting us throughout this project and always giving us words of encouragement. We would also like to thank Professor Yousef Mahmoud for his extensive knowledge about Maximum Power Point Tracking, solar panels and continued support. Our fellow classmate, Denver Cohen, also assisted in this project by giving us design suggestions and troubleshooting advice.

Abstract

The goal of this project was to design a Maximum Power Point Tracking system for solar panels that utilized a DC/DC converter and a microcontroller. The Perturb and Observe method was used to calculate and maintain the maximum voltage for a PV source with various voltage inputs ranging from 5V to 25V DC. The system was both simulated using Matlab Simulink and built using a converter and Arduino.

Contents

- Acknowledgements..... II
- Abstract III
- Contents IV
- Table of Figures VI
- Table of Tables VII
- Introduction..... 1
 - Brief History of Solar Panels 1
 - Our Project 1
- Background..... 2
 - How Solar Panels Work 3
 - Solar Panel Efficiency and Availability 4
 - Maximum Power Point Tracking 7
- Methodology 8
 - Perturb and Observe 8
 - Types of DC/DC Converters 11
 - Microcontroller Selection..... 12
- Design 14
 - Design Introduction..... 14
 - Matlab Code 16
 - Choosing values for our Buck converter..... 17
 - MOSFET decisions and Parameters..... 20
 - Design Modification..... 21
 - MOSFET Switch..... 21
 - Voltage Divider 24
- Simulation Results..... 25
 - PV Source 25
 - PV Source with MPPT Code 26
 - DC/DC Converter 29
 - Complete MPPT Simulation..... 31
- Implementation/ Monitoring Performance 35
 - Final Product 40

Result	41
Conclusions and Improvement	47
Appendices.....	49
A. Overall schematic.....	49
B. Arduino Data Sheet.....	50
C. Matlab Code.....	60
D. Arduino Code.....	60
E. Bill of Materials	63
References.....	64

Table of Figures

Figure 1 - Basic V-I Curve.....	6
Figure 2 - MPPT Algorithm Flowchart	9
Figure 3 - Solar Panel and Buck Converter Schematic.....	10
Figure 4 - Basic Schematic Buck Converter.....	12
Figure 5 - Flowchart for Overall Project.....	15
Figure 6 - Schematic of Buck Converter	17
Figure 7 - Buck Converter with Switch On and Off.....	22
Figure 8 - A High-Side P-Channel and Low-Side N-Channel MOSFET.....	23
Figure 9 - The Modified of High-Side and Low-Side MOSFET Switch	24
Figure 10 - Model of PV Source.....	25
Figure 11 - Output Voltage vs. Time Plot.....	25
Figure 12 - Output Current vs. Time Plot.....	26
Figure 13 - Output Power vs. Time Plot.....	26
Figure 14 - Model of PV Source with MPPT Matlab Code	27
Figure 15 - Output Voltage Graph with MPPT Code.....	27
Figure 16 - Output Current Graph with MPPT Code	28
Figure 17 - Output Power Graph with MPPT Code	28
Figure 18 - PV Source and DC/DC Buck Converter	29
Figure 19 - Output Voltage of DC/DC Converter vs. Time	30
Figure 20 - Close up of voltage output	30
Figure 21 - Inductor Voltage.....	31
Figure 22 - Complete MPPT Simulation	32
Figure 23 - Final Output Current	32
Figure 24 - Final Output Voltage.....	33
Figure 25 - Final Output Power	33
Figure 26 - Final Output Power with Input Voltage of 5V.....	34
Figure 27 - Final Output Power with Input Voltage of 10V.....	34
Figure 28 - Final Output Power with Input Voltage of 15V.....	34
Figure 29 - Final Output Power with Input Voltage of 25V.....	35
Figure 30 - PWM at 50% Duty Cycle.....	38
Figure 31 - Circuit side of Prototype	40
Figure 32 - Arduino side of Prototype	41
Figure 33 - Output Voltage of DC/DC Converter	42
Figure 34 - Ripple of the Output Voltage	43
Figure 35 - Output Power Curve at 20V.....	45
Figure 36 - Final Output Voltage from Oscilloscope	46
Figure 37 - Power Curve with Various Input Voltages	46
Figure 38 - Final Schematic.....	49

Table of Tables

Table 1 - Time Divisors and Corresponding Frequencies	36
Table 2 – Final Output Data from Arduino	44

Introduction

Brief History of Solar Panels

Solar Panels have become an increasingly popular form of renewable energy. With a simple beginning in the 1800s, photovoltaics (PVs) have continued to be improved and used as a form of energy. In the 1800s, Edmond Becquerel discovered the photovoltaic effect, which led to further research and the eventual invention of the first Photovoltaic Solar Cell in 1883 by Charles Fritts [28]. Since then, improvements have been made to increase efficiency, make solar cells more affordable, and increase their flexibility. This is further discussed in the background of the report. The overall project focus was to increase the efficiency of a basic solar panel.

Our Project

The objective of this project was to design a Maximum Power Point Tracker (MPPT) to constantly calculate and maintain the maximum amount of power from a solar panel. By using a DC/DC converter and an Arduino, our team was successfully able to create a system to reach this maximum power.

The solar panel was modeled using a DC voltage source, which then was connected to the DC/DC converter. A buck DC/DC converter was used to step the voltage down. This was required to have the voltage be in the acceptable input range for the arduino. Using the Arduino, a code was created for the Pulse Width Modulation (PWM), which determines the frequency of the PV source. The Perturb and Observe (P&O) method was used to calculate the maximum power the 'PV' source outputs, and the necessary duty cycle for the PWM. That information

would then be relayed back to the PV source and adjust it accordingly, to maintain the PV source at the peak power. To test our design, the DC source was adjusted to various voltage inputs, and the maximum power was successfully calculated each time.

This project was created using Matlab Simulink software and implemented using a breadboard then solderable board, an Arduino, and a DC voltage source.

Background

As the world continues to develop, the search for renewable energy continues. Not only is renewable energy popular with companies, but with advancements in technology, everyday people are beginning to utilize it. Even the government is taking notice. As the US Department of Energy states, “We’re taking responsible steps to cut carbon pollution, develop domestic renewable energy production and win the global race for clean energy innovation. We’re also working to dramatically increase the efficiency of appliances, homes, businesses and vehicles...” [8].

“As global temperature rise, wildfires, drought and high electricity demand put stress on the nation’s energy infrastructure” [8]. Today, 26% of the world is powered by renewable energy [28]. There are five main types of renewable energy being used. These five are solar, geothermal, bioenergy, wind, and water (hydroelectric). Solar energy is one of the more popular renewable energies, as it accounts for a growing portion of all renewable energy used throughout the US.

As the US department of energy has stated, the development of domestic renewable energy has begun [8]. Solar energy has taken off, as companies such as Solar City have made solar panels more affordable for homeowners. The term “living off the grid” has become more

popular and people are using solar power to aid in their sustainable lifestyles. According to Dictionary.com, Living off the Grid means “not requiring utilities such as electricity, water, etc” [18]. Solar power is so popular among renewable energy because it is essentially free energy. The energy comes from the sun; as long as you live in a location that gets sunlight you can have solar power. It is a form of energy that should be accessible for everyone, and that is why solar panels are so important. They harness the power that everyone is already exposed to and give us the capability of using it.

How Solar Panels Work

Solar panels are made of PV cells and other components such as inverters. PV cells are two layers of semiconductor material with opposite charges. When sunlight enters the cell, its energy knocks electrons, which spread in both layers. Due to the opposite charges of the layers, the electrons should move from the N-type layer to P-type layer, however the P-N junction creates an electric field that prevents this from happening and enforces electrons to flow only from the P to N. However, an external circuit connected to the PV cells allows the electrons in the N to travel to P. This electrons flowing through the circuit provide the direct current (DC) electricity. The inverter then converts this DC into alternating current (AC) electricity. The electrical panel then transmits power to lights or other appliances the panels are connected to. Depending on the company that installs the solar panels, the power generated either directly powers appliances/your home or gets fed back to the grid. Most of the time, the power meter or utility meter measures the energy you consume and feeds back to the grid [16].

Solar Panel Efficiency and Availability

As solar panels are becoming increasingly more popular, more research is being done to improve upon them. As of early 2016, the average solar cell is only 15% efficient [9]. As of the fourth quarter of 2016, the efficiency of solar panels is recorded at 22.1% by the First Solar's Cells Break [20]. This means that there is still over 80% of untouched energy that could be used. It is often pointed out that the efficiency of a solar panel varies by brand. For example, according to ScienceAlert, scientists have been able to develop a solar panel with 34.5% efficiency [19]. Regardless of the brand type, there is still more work to be done to make solar panels more efficient. Even though this is a huge improvement in the efficiency of solar panels, this is little compared to the efficiency of wind turbines, which are about 79-85% efficient [20]. In spite of the efficiency being so low, solar panels are being implemented and used in many places because it is easy to install, does not require maintenance and is residential and commercially friendly. People also tend to choose solar panels more so than wind turbines because of their accessibility.

Efficiency is determined by the reflection of the solar panel, and the bandgap energy of the semiconductor within the panel. If the panel has too much reflection, the sunlight isn't absorbed. If the electrons hitting the solar panels have energy higher than the bandgap energy, the photons could pass right through without forming an electron hole pair, which is needed to get energy. The sun puts out a spectrum of radiation and not all of it is light, only some, so it is not possible to harvest 100% of the sun's light with photovoltaics cells. According to physics, the maximum sunlight that can be captured by a photovoltaic cell is approximately 33.7%. The efficiency factor of a solar panel can be because of the silicon that is used to make the solar cell, which is part of a monocrystalline cell. The current commercial monocrystalline cells are only about 24% efficient. This is one of the constraints of making solar panels more than 24%

efficient, as it is impossible to make it more efficient than its materials. Other constraints are tilt angle, and any possibility of shade. Most solar panels are set at the optimum angle based on the angle the sun hits the home (or other location where the panels are installed). As the sun moves throughout the course of the day, the amount of sun the solar panels will get varies on the angle to which they are set. If the sun is blocked by clouds, the amount of sun hitting the solar panels would decrease, thus decreasing the power converted. So in order to get the best performance of the solar panels, it would be best to make sure that the solar panels get the maximum hours of sunlight during the day, to do that it would need to be clear of shades and the right orientation depending on the time of the day. All these factors can be determined by looking at a solar panel's characteristic curves, which are the output current and voltage graphs. The maximum power in an ideal situation for a solar panel can be found using the IV curve. This is the graph of the current vs. the voltage. The peak would be the optimum point of power for the panel [29]. Making all the various adjustments previously discussed would help try and obtain the maximum power, but it would not maintain it. Maintaining the maximum power is the goal of this project.

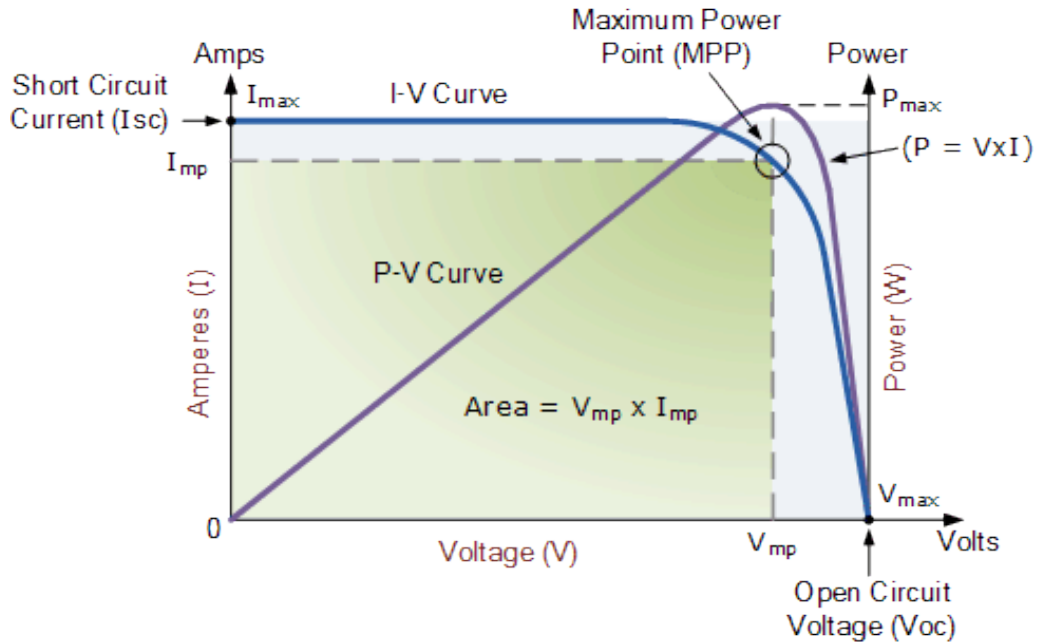


Figure 1 - Basic V-I Curve

Besides improvement on efficiency, solar panels are also becoming more affordable. This is partially due to the efficiency and partially due to the materials the solar panels are now made of. The University of Toronto has developed light sensitive nanoparticles that are more flexible, less expensive version of the original solar panels. Cheaper types of solar panels mean that more and more people will find them affordable and would consider using them as part of the power to their house. This will increase the amount of renewables that are used throughout the world.

Even with all of these advancements, solar panels are still sensitive to certain things. Sun intensity can cause the solar panels to become too hot, or can be so weak that the solar panels barely produce any amount of power. Solar panels can also work poorly if they are badly insulated. Bad insulation would cause heat to escape the solar panel and power to be lost. Other methods must be used to monitor the peak efficiency of solar panels to ensure that they are as efficient as they can be.

Maximum Power Point Tracking

As previously stated, other methods need to be employed to monitor and improve efficiency of solar panels. The most popular method is Maximum Power Point Tracking, or MPPT. MPPT is measuring the power of the solar panel at given intervals and making sure it is always at its maximum power. A measurement is taken from the solar panel and the power is calculated. After a specified interval, another measurement is taken. These two measurements are compared, and adjustments are made to the solar panel to ensure that the most recent measurement will lead to the maximum power.

MPPT is not a new technology. Some companies have been designing solar trackers for years. Most solar trackers move with regard to the angle of the sun, and do not constantly calculate power. Linak, for example, has two different types of solar tracking systems that both use integrated control actuators. The solar panels can either move 180 degrees (single axis) or can tilt in all different angles using dual access [23]. First Solar and Solar Flexrack have similar trackers that follow the movement of the sun throughout the day [20] [31]. These trackers are controlled by MPPT controllers. Controllers such as the MPPT Tracer Solar Charge Controller are installed and read a solar panel. Based on the information read, all solar panels are adjusted to follow the sun's path [22].

Though there are some MPPTs already on the market, our team has decided to make our own MPPT using a converter and microcontroller. The following paper illustrates our thought process during the design of our MPPT, along with results of our working product.

Methodology

Perturb and Observe

The main objective of maximum power point tracking is to read the voltage and current from the solar panel, perform the calculation for power and then display the power at its maximum. There are many algorithms available to execute this process. Some examples include Perturb and Observe (P&O), Incremental Conductance, Parasitic Capacitance and Constant Voltage Method. Of all the available algorithms, P&O is the most widely used algorithm because of its easy implementation. As for the Incremental Conductance method, it is more complex. However, a pro with this method is that it can be more accurate than the P&O method. As for the Parasitic Capacitance, it is much more complicated since the effect of the solar cells' parasitic junction capacitance matters [14].

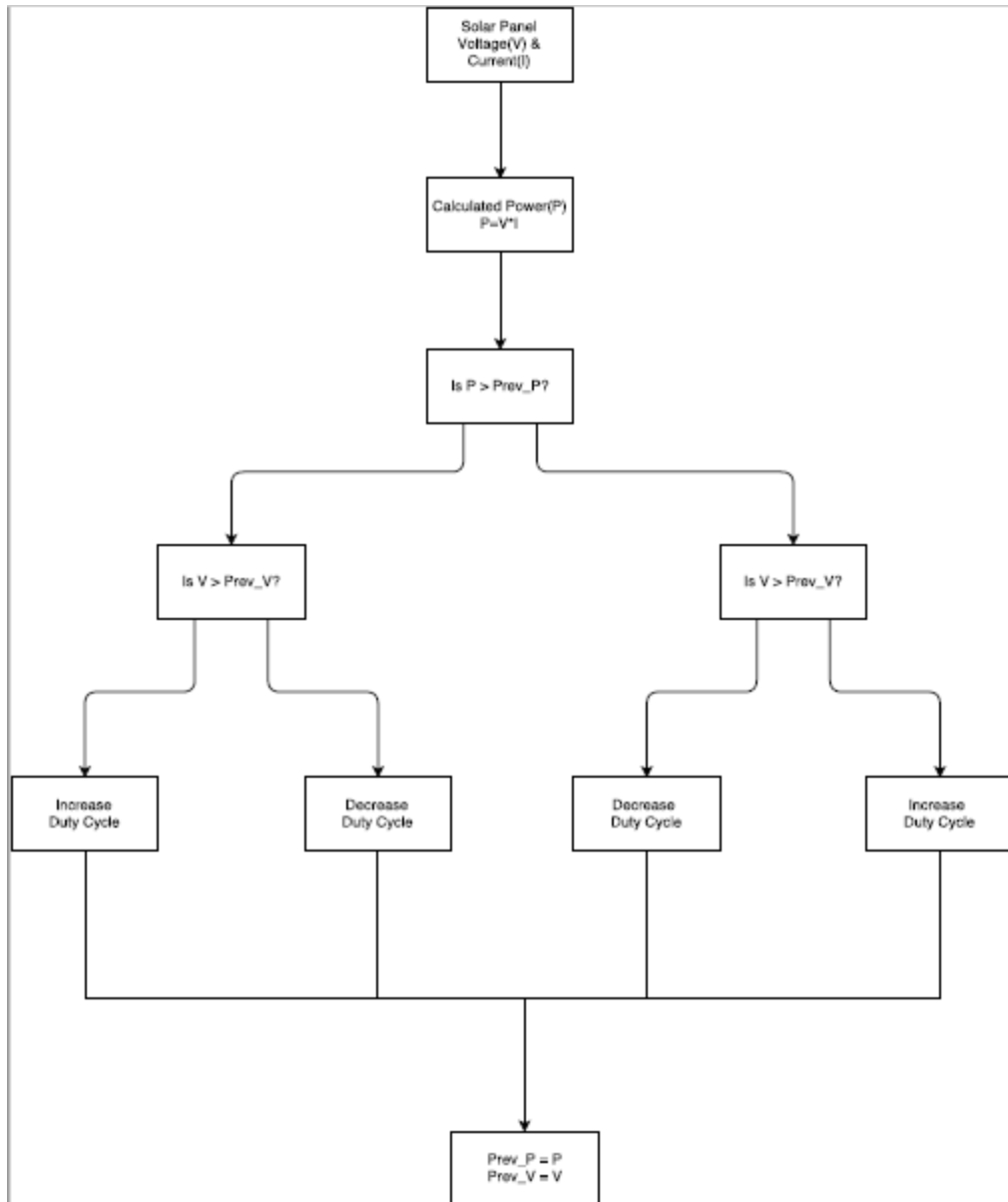


Figure 2 - MPPT Algorithm Flowchart

For this project, P&O method was chosen to perform the MPPT algorithm. The output voltage and current from the solar panel constantly feeds the input of the DC/DC converter.

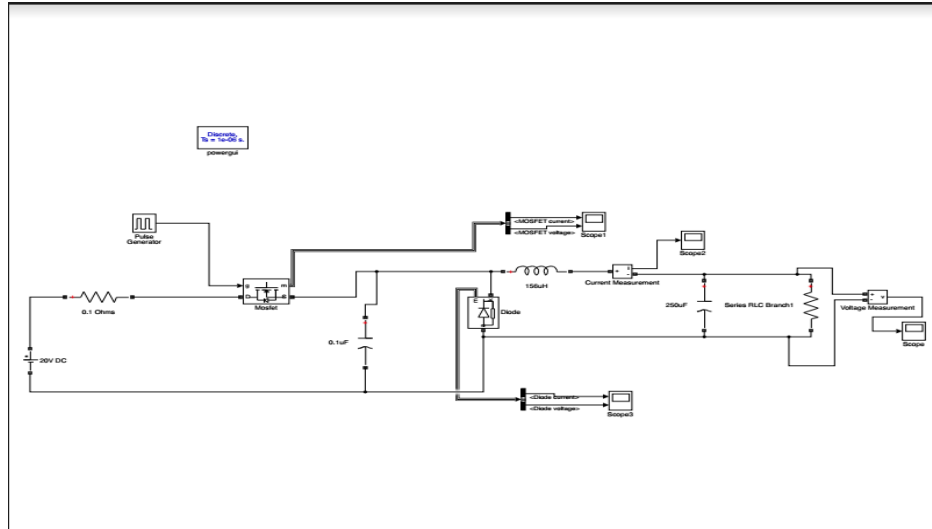


Figure 3 - Solar Panel and Buck Converter Schematic

Since the type of DC/DC converter chosen for this project is the buck converter, it will step the input voltage down based on the duty cycle. The preset duty cycle for the simulation is 50%, so the voltage from the input of the buck converter will be reduced by half when measure the output of the converter.

The output voltage and current of the buck converter gets sampled and then calculated for power. The calculated power then gets compared to the previous maximum power, if there is one. For the comparison, if the current power is greater than the previous maximum power and the current voltage is also greater than the previous voltage, then the converter's duty cycle increases. Likewise, if the current voltage is less than previous voltage then the duty cycle decreases. As for the case that current power is less than the previous power, and the current voltage is greater than the previous voltage, according to P&O algorithm, the duty cycle of the buck converter should decrease. However, if the current voltage is less than the previous voltage, then increase the duty cycle. The process described above is in one loop of the code. Before starting the next loop, the current power is set to the previous power if it is greater than the

previous power. Then a new loop of the same process which is by starting to measure the new voltage and current that is resulted from the newly set duty cycle of the buck converter. Refer to the flowchart in above for a clearer understanding [25].

Types of DC/DC Converters

For this project we need a product that converts and regulates a certain DC voltage level to different DC voltage level. There are many possible DC/DC converters that can achieve this condition, some examples of these converters are buck converter, boost converter, buck-boost converter, and the cuk converter. A buck converter is a switching converter that operates opening and closing an electronic switch periodically, and it is a step down voltage converter since its output voltage is smaller than its input voltage. A boost converter is similar to the buck converter except it functions as a step up converter instead of a step down. Its output voltage is larger than the input voltage. As for buck-boost converter, the output voltage can be either higher or lower than the input voltage. Lastly, the cuk converter is similar to the buck-boost converter, the output voltage can be either larger or smaller than the input voltage. The difference between the cuk and buck-boost is that the output of the cuk converter has a polarity reversal [15].

The Buck converter was chosen for this project due to its simpler concept than the rest. In general, buck converters in steady state should have the following properties: inductor current should be periodic and continuous, the average inductor voltage should be zero, the average capacitor current should be zero, and most importantly, the power delivered to the load should be the same as power supplied by the source for ideal components. For non-ideal components, the power losses should be taken into the account as well [15].

The figure below shows the topology for buck converter. The source connects to the switch, which supposed to control the switching period, T , which relates to the duty cycle of the buck converter. From the switch, the inductor is connected in parallel to the diode, which the diode is connected to ground, and the other end of the inductor is connected to the capacitor in parallel to the load resistor [15].

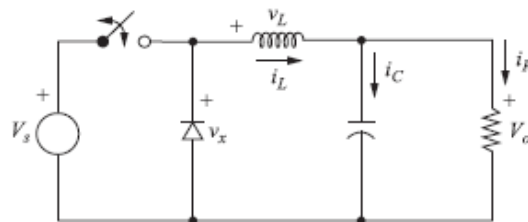


Figure 4 - Basic Schematic Buck Converter

Microcontroller Selection

Our project required a device that can control the duty cycle based on the output power and modify the pulse width of an input square wave. The input square wave signal is sent to the MOSFET which act as an electronic switch, the switch then controls the duty cycle of the buck converter which affects the converter output voltage. A microcontroller is used to partially serve these purposes. It calculates the power from the samples of the voltage and the current every second. By comparing a previous power to a new power, an appropriate modification of duty cycle would occur and so a change in PWM would happen.

A microcontroller was chosen rather than a microprocessor because it required less external hardware, reducing the final product size. Being compact makes it very efficient. It also has power saving mode which helps reduce the power consumption of the whole system. The idea of a

microcontroller can be comparable to a mini computer. It includes many other components that can be used in so many applications like our project [21].

Although there were many options for microcontroller, the Arduino Mega 2560 was chosen to accomplish these requirements because it allows fast and easy prototyping and due its large online community it has been made easier to get resources to help writing Arduino programs. This Arduino Mega 2560 consists of the 54 digital pins, 16 analog inputs and many other pins such as a power, input of 5V or 3.3V, ground (GND) and PWM [6]. One factor that needed to be consider when choosing a microprocessor was the electrical specification. The power supply of 20V was used as a source voltage and DC/DC converter was used to step down the 20V input to 10V output. Hence, it was very important for our microprocessor to have an input voltage requirement of 10V. The Arduino Mega 2560 has a recommended input voltage of 7-12V where the voltage input that our DC/DC buck converter produced fall within. In addition to it, the operating voltage of this Arduino is 5V [5].

The Arduino communicates with the user interface such as laptop and circuit board through the Arduino Software (IDE) and the USB port connected to the laptop. The IDE includes a feature known as a serial monitor which allows the data to be sent from the board and print it out on the laptop [5].

Design

Design Introduction

Once the DC/DC converter component values were decided and the MPPT method chosen, the next step was to simulate our design for the overall project. This was done in stages based on the main sections in the following flowchart. The entire project would consist of a PV source that acts as the input for the DC/DC converter. This is the input for the microcontroller, which receives the voltage and current values and calculates the maximum power based on that information. Once that is gathered, the duty cycle of the DC/DC converter must be adjusted to maintain that maximum voltage. The microcontroller adjusts the converter duty cycle by the PWM, which is another input for the converter. The following sections explain the code used to calculate and maintain the power, and how it was designed and implemented in Matlab Simulink.

Flow Chart for Project's Prototype

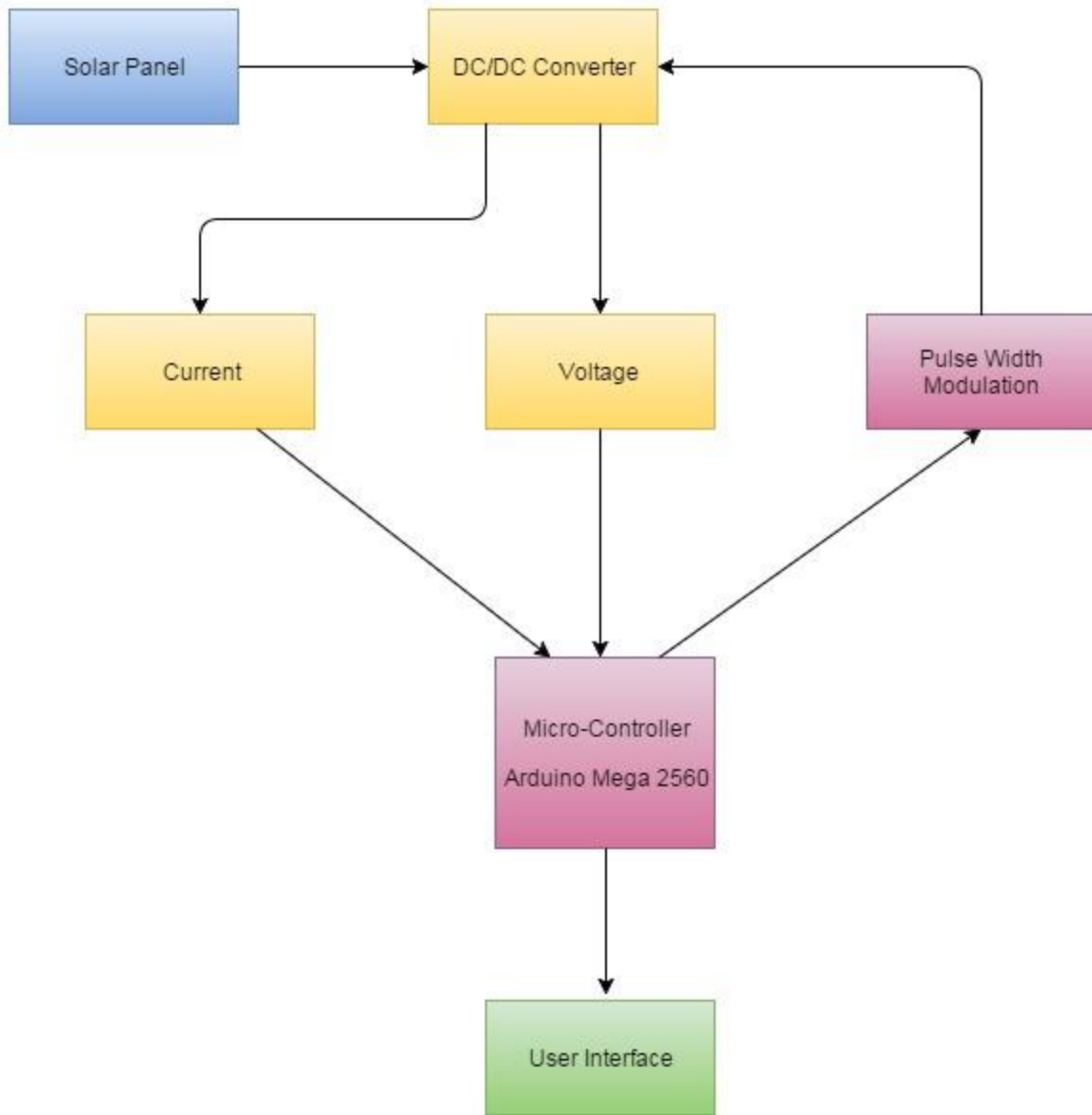


Figure 5 - Flowchart for Overall Project

Matlab Code

One of the key factors of this project was the MPPT algorithm and the way this algorithm implemented on our system. As it was mentioned before, perturb and observe method was chosen to perform the tracking of maximum power point. This method was based on the electrical power that was calculated with voltage and current from DC/DC converter and then, it compared the power at each moment and tracked the maximum power by modifying the duty cycle of the system. To implement this algorithm, Matlab software was used to simulate the whole system and test out the code before implementing it practically. Matlab software was chosen due to its ability to simulate and code in one file. Using the flowchart of the algorithm, the code was written step by step [25].

The instantaneous power delivered by the solar panel at each moment was calculated by measuring the output voltage and current across the DC/DC converter and multiplying both quantities. This function executed in a loop so that it would be repeated and produced multiple power readings: $P_1, P_2, \dots, P_n, P_{n+1}$. If the new power, P_{n+1} is higher than the previous value P_n , then the operating voltage is compared. If the new voltage V_{n+1} is greater than the previous voltage V_n , then the duty cycle decreased, else the duty cycle increased. Otherwise, if the new power is less than the previous value and the new voltage is greater than the previous voltage, then the duty cycle increased, else the duty cycle decreased. Modifying the duty cycle would change the operating voltage and the power would be calculated again, which would always result in maximum power point.

Choosing values for our Buck converter

Once it was decided to use a buck converter, we knew the basic schematic of what our converter would look like.

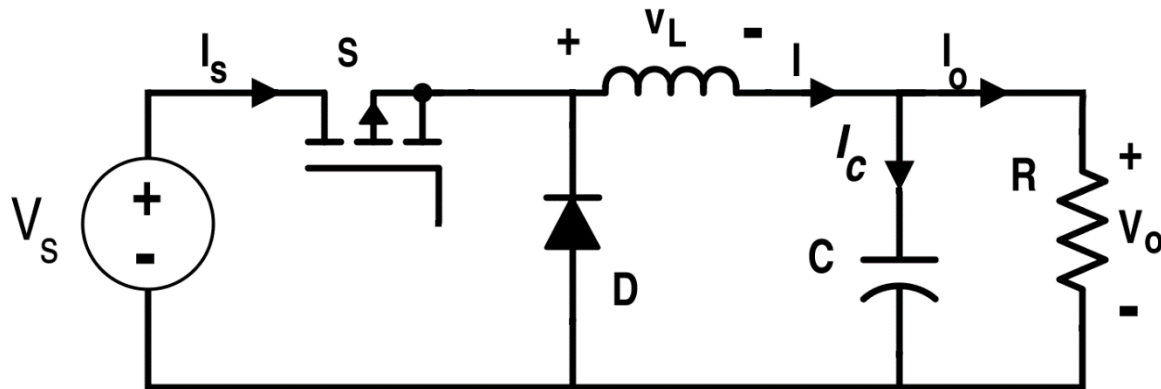


Figure 6 - Schematic of Buck Converter

As shown in the buck converter schematic, a resistor, capacitor, diode, and inductor are needed. The values for these components are based off the frequency and desired duty cycle. The duty cycle is the percentage of time that the waveform is ‘on’ or high. For example, if the duty cycle was 80%, then the waveform that duty cycle pertains to would be on 80% of the time and off 20% [27]. According to Power Electronics, the desired frequency for most buck converters is 20 kHz. When the switching frequency increases, “the minimum size of the inductor to produce continuous current and the minimum size of the capacitor to limit out ripple both increase....High frequencies are desirable to reduce the size of both the inductor and capacitor” [15]. Due to this reason and avoiding audio noise, the frequency had to have been at least 20 kHz. As our voltage source is 20V (an arbitrary amount to represent a solar panel), and our Arduino has an input voltage range of 7-12V, the buck converter must step down the voltage within that range. It was decided the initial output voltage of the buck converter should be 10V. Since

$$V_{out}=V_{in}*D, D=\frac{V_{out}}{V_{in}}=\frac{20}{40}=0.5$$

The duty cycle is therefore 0.5, or 50%.

Once the duty cycle and frequency were determined, the next step was to find a resistor value. According to Power Electronics, a common value for a load resistor in a converter is 10 ohms [15]. The 10 ohm resistor that we were planning to use could only handle a small amount of power, and when tested with the rest of our converter it burnt out. To resolve this issue, instead of using one 10 ohm resistor, three 31.6 ohm resistors were used in parallel. These resistors are able to handle 30W of power, which is more than is created. Adding these resistors in parallel gives:

$$R_{total} = \frac{R_1 R_2 R_3}{R_1 + R_2 + R_3}$$

$$R_{total}=10.533$$

With the frequency, duty cycle, and resistor value decided, the inductance value could be calculated. The equation to calculate inductance is found as follows:

$$L = (1 - D) \frac{R}{2f} = 1.25 * 10^{-4} \text{H}$$

This value is the minimum value of the inductor that could function properly within our buck converter. To provide a safety net and ensure that the inductor works properly within our converter, Power Electronics suggests to multiply the minimum inductor value by 1.25, to obtain continuous inductor current [15].

$$L=1.56*10^{-4}=0.156\text{mH}$$

As the inductor we are using is made with a circular metal ring and coil, the number of turns the coil needs is necessary to get the correct 0.156mH needed. The equation below finds the number of turns needed.

$$L = \frac{r^2 \text{turn}^2}{9r + 10l}$$

Where r is radius of the hole of the metal ring, l is length/thickness of the ring, and L is inductor value. Rearranging this, the number of turns can be found:

$$\text{turn} = \sqrt{\frac{L(9r + 10l)}{r^2}}$$

Using r=1.5cm, l=1cm, and L=0.156mH, the number of turns is found to be 64 turns. This means our inductor has 64 windings on the metal ring.

To find the capacitor value, the voltage ripple was needed. Voltage ripple is the change in voltage between the highest peak and the lowest peak. The lower the ripple voltage is, the more accurate and precise the overall voltage is. The goal is to have as small a ripple as possible.

Voltage ripple is found using the equation

$$\Delta V_o = \frac{T \Delta i_L}{8C}$$

Where T = time, i_L = inductor ripple current, and C = capacitor value. As seen in the above equation, the inductor ripple current is needed in order to find the voltage ripple. Using equations from the textbook we know

$$\Delta i_L = \frac{V_o(1 - D)}{Lf}$$

Plugging in the values we have already, we get

$$\Delta i_L = \frac{V_o(1 - D)}{Lf} = \frac{10(1 - .5)}{(6.25e - 5)(20e3)} = 4$$

As we want to have a very small voltage ripple, and so we arbitrarily choose to have our ripple be 0.1V. Using this, we can rewrite the voltage ripple equation to obtain

$$C = \frac{T\Delta i_L}{8\Delta V_o}$$

The values we have can be substituted as seen below to get a total of 0.00025F, so C= 0.25F, or 250uF.

$$C = \frac{T\Delta i_L}{8\Delta V_o} = \frac{1}{\frac{20000}{8 * 0.1} * 4} = 0.00025F$$

Having a capacitor of 250uF would give a voltage output ripple of 0.1V, which is a small output. This was tested, and the results appear below in the Simulation Results section.

Lastly, the diode had to be chosen. As the diode's purpose was to regulate the energy transfer within the DC/DC converter. The diode helps prevent the capacitor from discharging to ground when the converter is in the 'off' mode [10]. To meet these functionalities within the converter, the diode had to have a high enough forward voltage that it would not be cut off and allow the converter to function. For this reason, a basic 1N4004 diode was used.

MOSFET decisions and Parameters

In order to complete the buck converter, a transistor had to be selected. A transistor is defined by Dictionary.com as “a semiconductor device that amplifies, oscillates, or switches the flow of current between two terminals by varying the current or voltage between one of the terminals” [32]. There are different types of transistors, such as Bipolar Junction Transistors (BJT) or Field Effect Transistors (FET). Our team decided to use an N-MOSFET (NMOS for short). The parameters we needed to pay close attention to where the drain to source breakdown voltage, resistance, and drain current. The drain to source breakdown voltage needed to be greater than 20V, because the solar panel would have an output of 20V. The resistance needs to

be low in order to reduce conduction losses. To find the needed minimum value for the drain current, the current flowing through the inductor needed to be calculated. The output power for the solar panel is 0.2W. The output voltage for the DC/DC buck converter is 10V. Using the equation

$$i_L = \frac{P}{V}$$

the above numbers were filled in to find the inductor current.

$$i_L = \frac{P}{V} = \frac{.2W}{10V} = 20mA$$

Given this information, the NMOS IRF 521 was chosen. From the data sheet, the NMOS has a drain to source breakdown voltage of 80V, which is much more than the 20V minimum. The drain current is 9.2A, which exceeds our 20mA minimum. The resistance is 0.27, which is a low value. Therefore this NMOS exceeds our minimum values for everything.

Design Modification

Although during the design and simulation phases everything went smoothly with the original design, during implementation phase it was necessary for the design to be slightly modified due to the chosen components and its manufacturer. The modifications that were made to the original design are the buck converter switch, and the output reading from the buck the converter to the Arduino.

MOSFET Switch

Within the buck converter there is an electronic switch that closes and opens based on the input PWM, refer to the figure below for the equivalent circuit when the switch is closed and

open. During the simulation, at the switching state of the buck converter, since the chosen component was the ideal MOSFET, there wasn't any signs of design errors. However, during implementation phase, the gate-source voltage of the MOSFET was not compatible with the oscilloscope that were used to check the output voltage. This caused the output voltage to be limited at 7V, which was lower than the expected output voltage of 10V. So in order to fix this problem, the team decided to use a MOSFET Switch Driver, a High-Side P-Channel Drive and a Low Side N-Channel Drive.

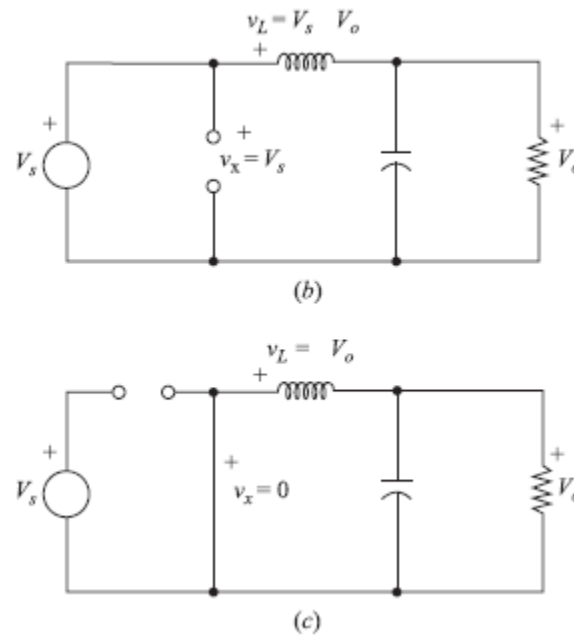


Figure 7 - Buck Converter with Switch On and Off

A High-Side Switch is controlled by an external signal and it connects or disconnects the power source to the load and it source the current to the load. As for a Low-Side Switch, it connects or disconnects the load to the ground and it sinks the current from the load [12]. The

connection is a slight modification of a Half-Bridge circuit, which is a High-Side P-Channel MOSFET and a Low-Side N-Channel MOSFET tied with common drain, refer to the figure below [24].

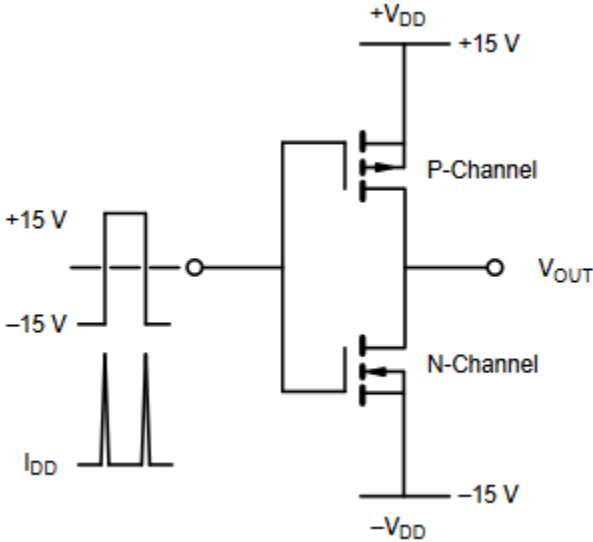


Figure 8 - A High-Side P-Channel and Low-Side N-Channel MOSFET

The figure below shows the modified circuit of the Half-Bridge. The gate of the P-Channel, IRF9520, is connected to the drain of the N-Channel MOSFET. Similarly to the figure above, the figure below has PWM input into the gate of the N-Channel and the drain is sourced from the P-Channel MOSFET.

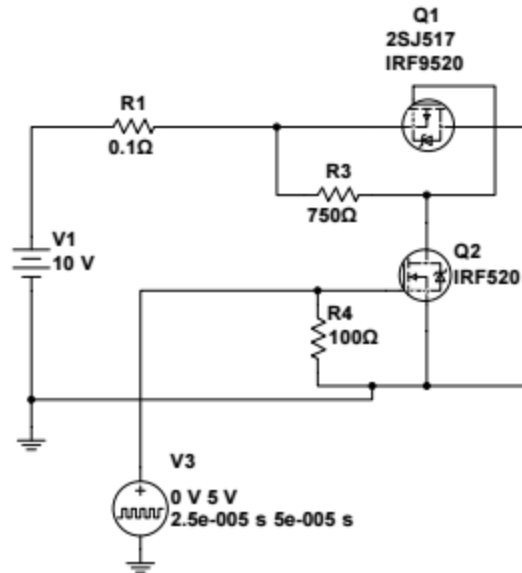


Figure 9 - The Modified of High-Side and Low-Side MOSFET Switch

Voltage Divider

When trying to read the output voltage of the buck converter, there was a problem that the team encountered. The Arduino Mega 2560 was the chosen component to perform the power calculation and the MPPT algorithm. The analog pins of the Arduino can only read the voltage from a range of 0-5V. However, the requirement for the prototype is to be able to read the voltage with a range of 0-25V. Therefore, to solve this problem, the team decided to use voltage divider to change the voltage going into the input pin of the Arduino to no larger than 5V. The reduced voltage will get converted back to its original value based on the voltage divider ratio before the Arduino uses that value to compute the power.

Simulation Results

PV Source

To test our design with Matlab Simulink before starting to build it, the first step was to create a PV source. This was modeled using a DC voltage source and a shunt resistor of 0.1 ohms.

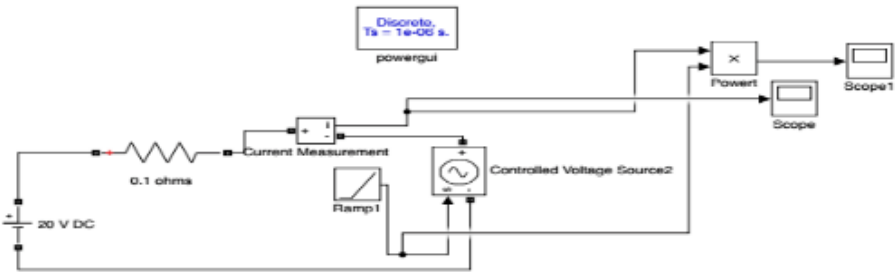


Figure 10 - Model of PV Source

To test the accuracy of the PV source, the voltage, current, and power graphs were found.

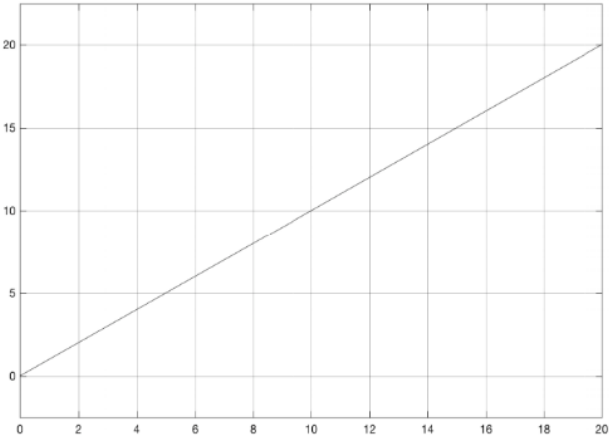


Figure 11 - Output Voltage vs. Time Plot

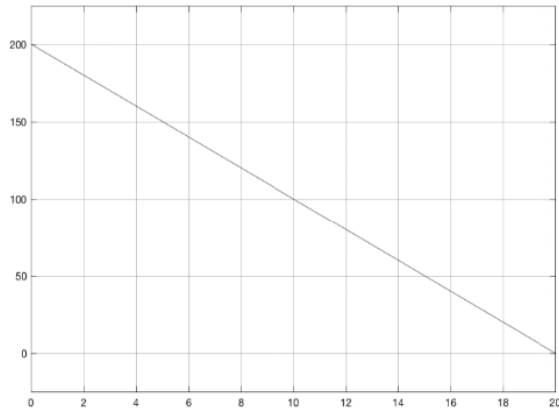


Figure 12 - Output Current vs. Time Plot

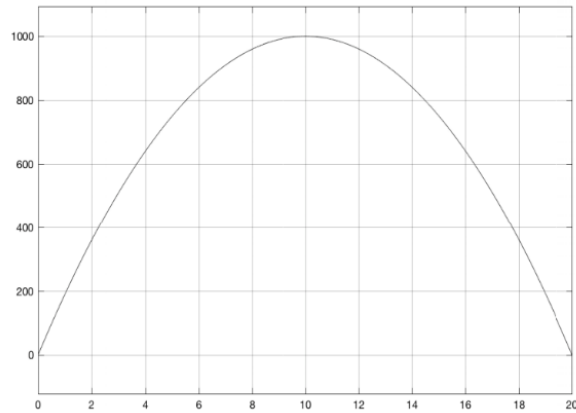


Figure 13 - Output Power vs. Time Plot

This graph corresponds to the typical power graphs for PV sources, and so our model of a PV source is accurate.

PV Source with MPPT Code

After confirmation of an accurate PV model, the MPPT code was added to model. The Matlab MPPT code was discussed in a previous section. It is used to calculate the maximum amount of power that the PV source can output and maintain.

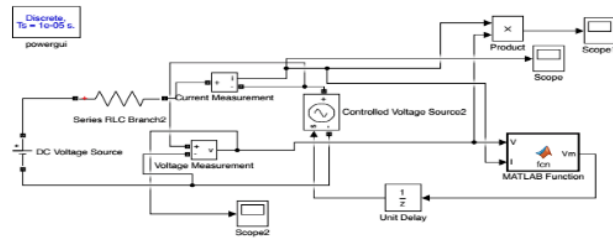


Figure 14 - Model of PV Source with MPPT Matlab Code

In the image above, the code is connected using the Matlab Function block in Simulink. The Unit Delay block is what adjusts the duty cycle and maintain the maximum power. In order to maintain the maximum power, the voltage and current output graphs must also maintain a constant value at whichever respective point will create the maximum power. In figures below, the constant values achieved at the end are the resulted maximum power.

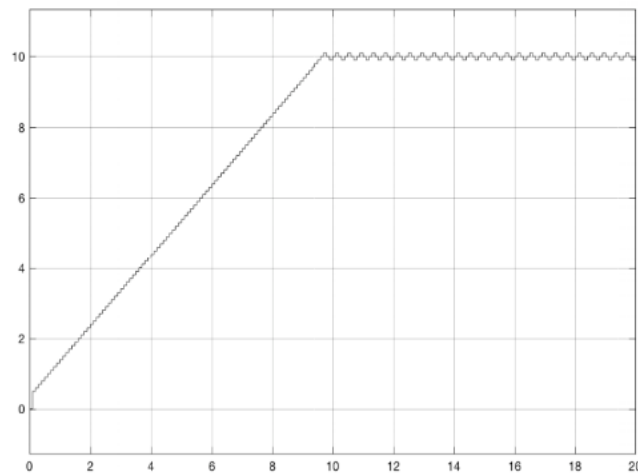


Figure 15 - Output Voltage Graph with MPPT Code

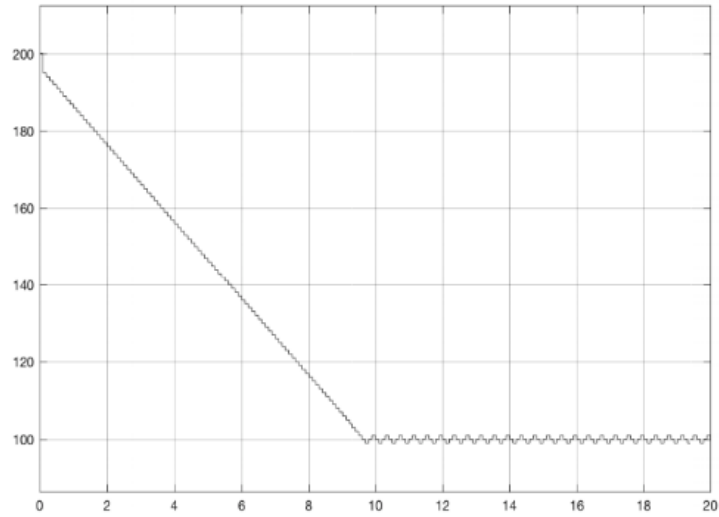


Figure 16 - Output Current Graph with MPPT Code

Based on these graphs, the expected maximum voltage should be $100A \cdot 10V = 1000W$.

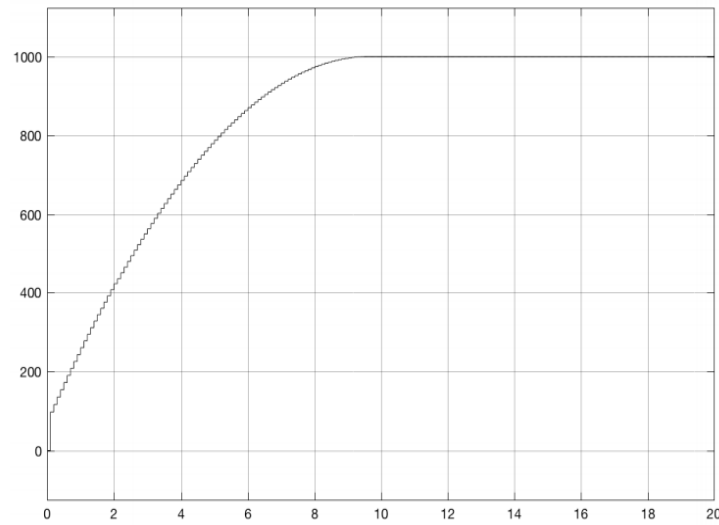


Figure 17 - Output Power Graph with MPPT Code

As seen on the figures above, the correct maximum output voltage has been found and maintained, and so the MPPT Matlab code is correct.

DC/DC Converter

After gathering all the specific values for the DC/DC converter, Matlab Simulink was used to design it. Below is our final DC/DC buck converter schematic.

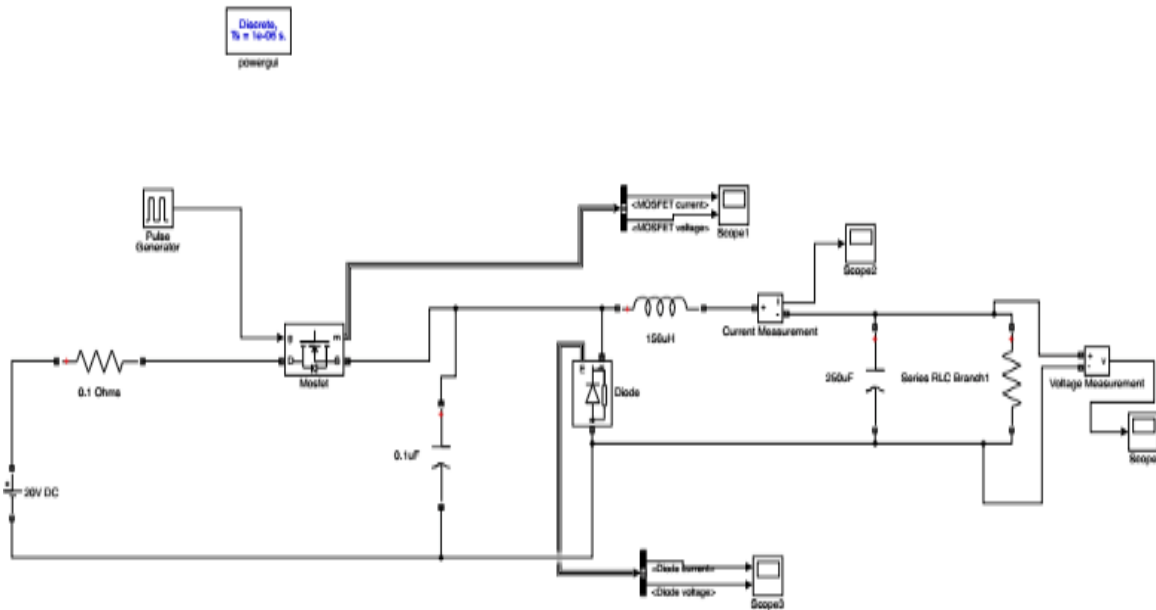


Figure 18 - PV Source and DC/DC Buck Converter

The PV source was used as the input to the converter. To test the converter simulation, an input of 20V was used. Here, a pulse generator acts as a simpler version of the PWM that would be added later on. Based on the previous calculations and a 50% desired duty cycle, the expected output voltage should be roughly 10V.

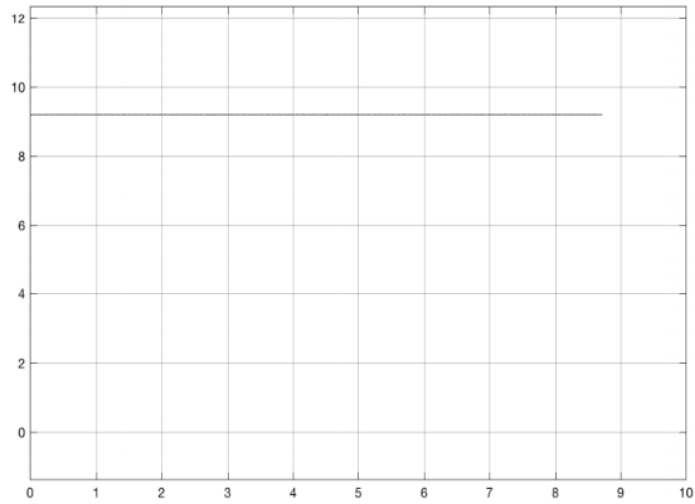


Figure 19 - Output Voltage of DC/DC Converter vs. Time

As seen on the graph, the output voltage was a consistent line around 9V. The reason for the slight difference between the expected output voltage and simulated output voltage was due to some loss within the system. The calculations did not take into account the resistance of the PV source, which lowered the input voltage to the converter.

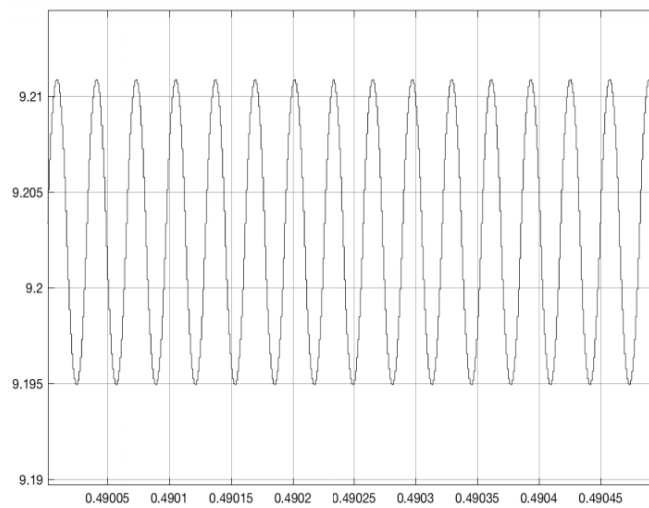


Figure 20 - Close up of voltage output

The figure above is a zoomed-in snapshot of the output voltage. As previously discussed, the goal for the output voltage ripple was 0.1V. The output voltage we achieved through the converter in simulation oscillates between 9.195V and 9.21V, which is a ripple of 0.015V. This ripple is even smaller than calculated, which is beneficial as the smaller the ripple the better in converters. Lastly, the inductor voltage was measured and can be seen below.

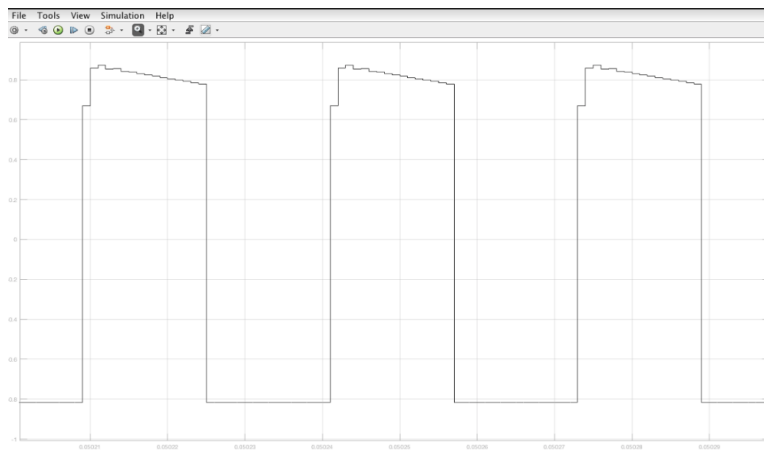


Figure 21 - Inductor Voltage

As mentioned in the Methods section, the average inductor voltage should be zero. The inductor voltage ranges between 0.8 and -0.8V, which averages out to zero. As the voltage acts correctly, it is further proof that our DC/DC buck converter is working successfully.

Complete MPPT Simulation

After all parts of the MPPT worked separately, they were combined and tested together. The figure below shows the PV Source, DC/DC converter, and MPPT Matlab code with the PWM.

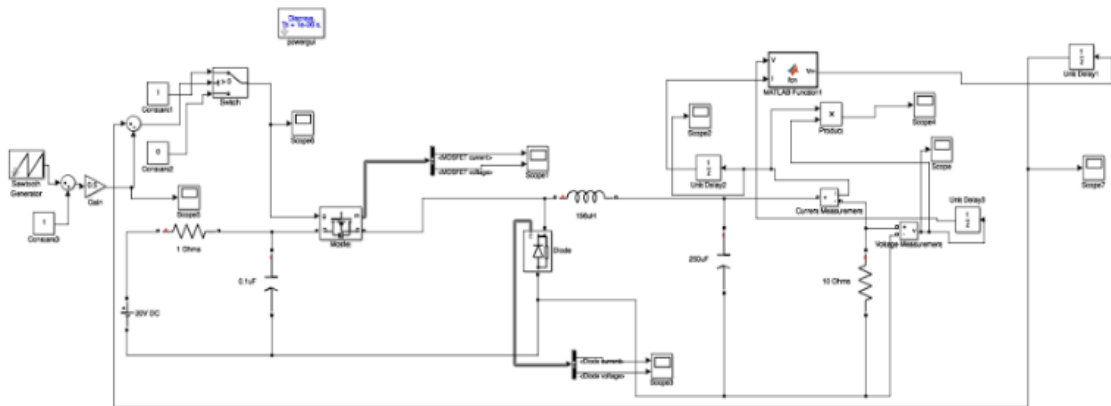


Figure 22 - Complete MPPT Simulation

By adding the MPPT code, the DC/DC converter would step down the PV voltage, relay the voltage and current values to the Matlab code, which then calculates what the voltage would be and sends that information to the PWM to adjust the duty cycle. This creates the steps within the graph figure below. The constant line is when the maximum power is finally achieved and maintained.



Figure 23 - Final Output Current

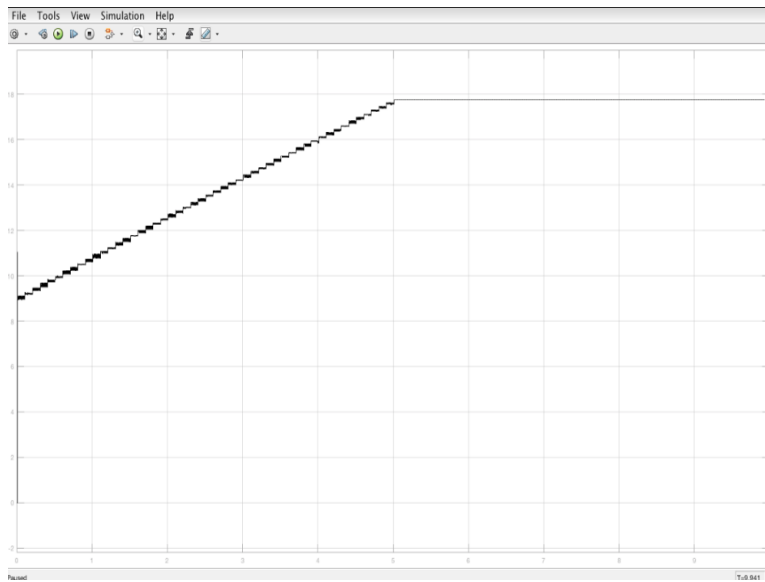


Figure 24 - Final Output Voltage

The current reached is around 1.8A, and the voltage is 18V. The maximum power should then be around 32W.

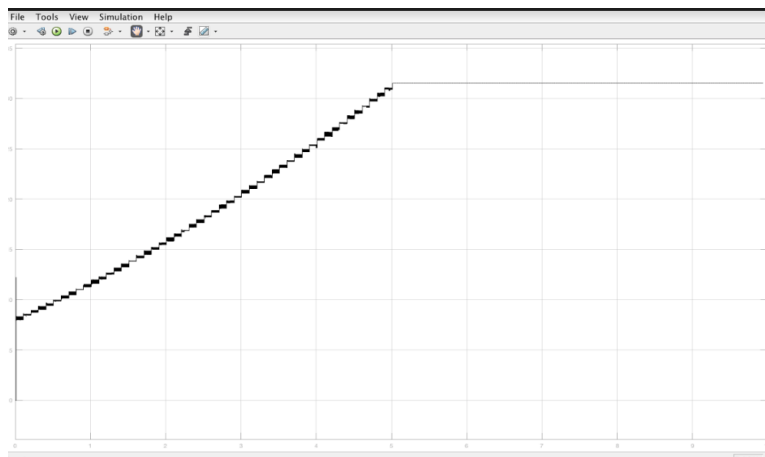


Figure 25 - Final Output Power

The maximum power is around 32W, which was the expected power value for the PV source of 20V. To confirm that our code indeed works at various voltages. The figures below show the final output power at various input voltages.

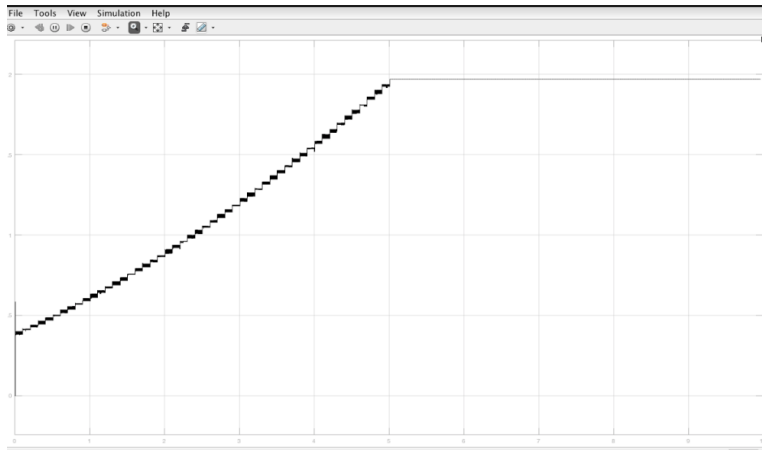


Figure 26 - Final Output Power with Input Voltage of 5V

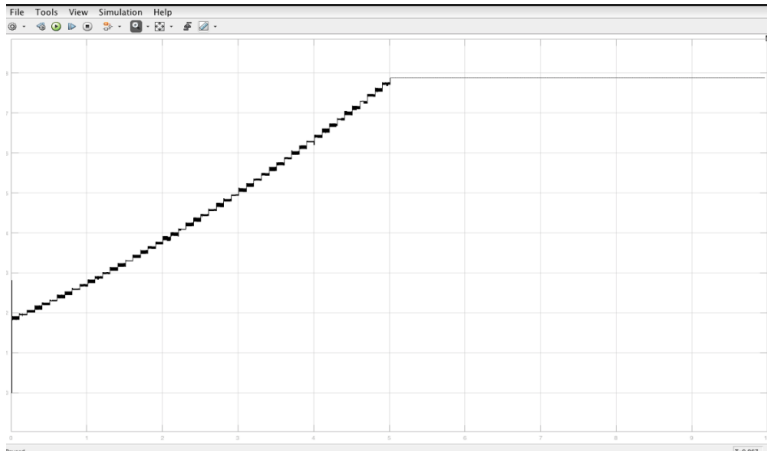


Figure 27 - Final Output Power with Input Voltage of 10V

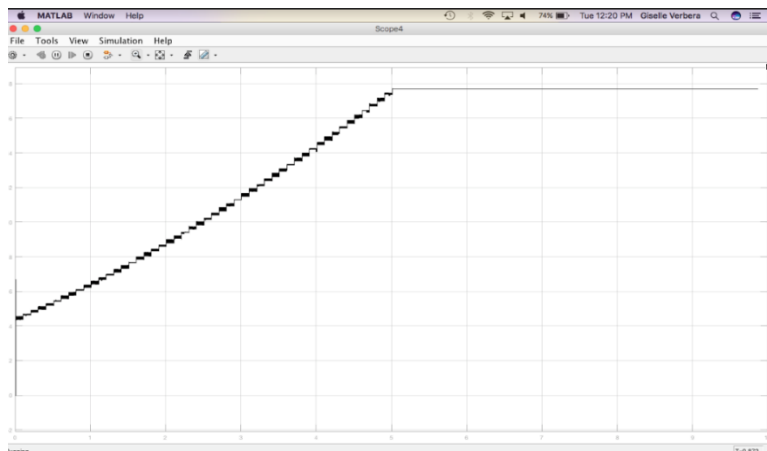


Figure 28 - Final Output Power with Input Voltage of 15V

It is therefore seen that no matter what the input voltage is from the PV source, our design accurately and successfully can calculate the maximum power and maintain this as a constant, until the input voltage changes again and the process is repeated.

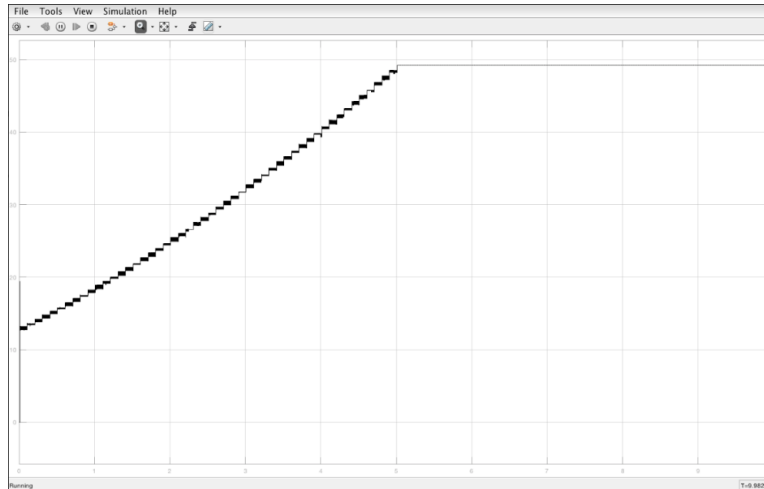


Figure 29 - Final Output Power with Input Voltage of 25V

Implementation/ Monitoring Performance

For our project, an Arduino needed to perform three distinct functions and those were as follows:

1. Provide the correct frequency of the PWM square wave,
2. Calculate the power using the current and voltage reading obtained from the DC/DC buck converter every time,
3. Implement 'Perturb and Observe' algorithm that compared the previous power to new power and modified the duty cycle of PWM to track the maximum power point successfully.

Firstly, pins were assigned for the PWM and the voltage reading. All the functions were written and uploaded to the Arduino by the Arduino IDE software. For the PWM signal, pin 9 was chosen. To set up pin 9 to provide the PWM signal to our DC/DC buck converter, the team first assigned the pin 9 to the PWM by writing the one line code,

```
int PWMPin = 9;
```

This code allowed the team to call `PWMPin` and assigned the specific duty cycle to it which then entered through pin 9.

Setting up the PWM of the Arduino can get complicated because each PWM pin was set with a default frequency that was much lower than our operating frequency. Pin 9, for example, had a default frequency of 490.2 Hz, while our system required an operating frequency of 20 kHz. To increase the operating frequency of the specific PWM pin, another line of code had to be written. Depending on which pin was used, the timer varied and depended on the timer divisor, the frequency varied as well. According to the Arduino page, pin 9 uses timer 2 [2].

Divisor	Frequency/Hz
1	31372.55
8	3921.16
32	980.39
64	490.20
128	245.10
256	122.55
1024	30.64

Table 1 - Time Divisors and Corresponding Frequencies

As the operating frequency of the project was 20 kHz, so the closest possible frequency to 20 kHz was 31372.55Hz for the Arduino pin 9. According to the table, 31 kHz is controlled by the timer 1 divisor. To set the timer 2 divisor to 1 for PWM frequency of 31372.55Hz \approx 31.4 kHz, a code was written in the setup function:

```
TCCR2B = TCCR2B & B11111000 | B00000001;
```

This code changed the frequency of pin 9 to 31.4 kHz. The duty cycle in the Arduino is defined on a scale of 0 – 255 where 0 is always off and 255 is always on 100%. The source voltage of 20V stepped down to 10V by the buck converter, which gave us a duty cycle of 50% and this corresponds to 127 in Arduino. To assign our `PWMPin` duty cycle to 50%, another line of code was written in the loop function:

```
analogWrite (PWMPin, 127);
```

This allowed the initial square wave to be at 50% duty cycle. To make sure that `PWMPin` successfully provided 50% duty cycle signal with 31kHz, the oscilloscope was connected and the frequency and pulse were monitored. The oscillogram of the `PWMPin` at 50% duty cycle is shown below:

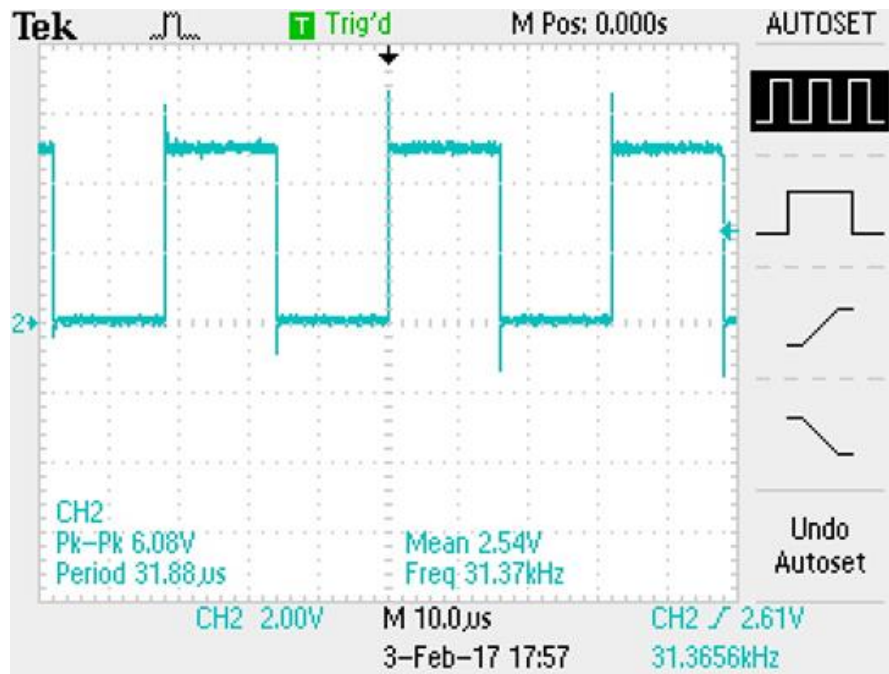


Figure 30 - PWM at 50% Duty Cycle

For our project, the duty cycle of the signal needed to be modified depends on the voltage input to track the maximum power point successfully rather than staying at 50% duty cycle the whole time. Hence, the code needed to be modified as follows:

```
analogWrite(PWMPin, duty_cycle);
```

The variable `duty_cycle` was declared and initially assigned to 127 and it would increase or decrease depends on the power value. Another variable `changed` was assigned and it represents the amount of change of the duty cycle. For this project, the step size was set to be 5%, which is corresponding to 12.7 in Arduino code. For example, if the duty cycle needed to increase, it would increase by 5% from whatever value `duty_cycle` was at and vice versa.

After successfully implementing the function for setting up the PWM signal for the switch of the buck converter, Arduino needed to calculate the power output by using the voltage and current from the buck converter. To read the voltage output from the buck converter, the analog

input pin, A1 was assigned to the voltage output port on DC/DC converter. As the voltage reading from Arduino would be in a raw data value (0 - 1023), it should be converted into a voltage reading (0 - 5V). This could be done by the following code [7]:

```
voltage = ((float)raw_voltage_data * 5.0) / 1023.0;
```

As the Arduino Mega 2560 can read the voltage up to 5.0V only, the voltage divider has been added. Hence, the voltage read from the analog input should be multiplied by the voltage ratio.

A current was calculated by dividing the voltage by the load resistor, 10.5 ohms as the current in the resistor is equal to the output current according to the Kirchoff's Current Law. These voltages and current values then entered through the Arduino to compute power.

To compute the power, a few variables were declared: P for power, V for voltage and I for current and using these variables, the power formula was written:

$$P = V * I;$$

After it executed the formula and calculated the power output, the P&O algorithm was implemented. The code for the algorithm was written based on the flowchart of the algorithm and the Matlab code that was written for the simulation [25].

The implementation of P&O algorithm was the major function of the Arduino. This algorithm processed by changing the duty cycle of the PWM that controls the MOSFET from 0% to 95% in increments of 5%. The change in the PWM would affect the output voltage of the DC/DC converter and alter the power output. As the voltage and current changes, the power calculations are made and recorded with each increment until it reaches maximum power. Once it reaches the maximum power point, it maintains the duty cycle value where the maximum point is recorded and maintains the maximum power point. If there is any change in the voltage or current value,

change the power output to be bigger than the previous maximum power point that was found, the algorithm will be executed again and change the previous maximum power point into the new power output and vice versa.

Final Product

The two figures below show the final prototype on a solderable board and the chosen microcontroller. The red alligator clip is connected to the positive side of the power supply/solar panel. And the white color alligator clip is connected to the ground/negative side of the power supply/solar panel.

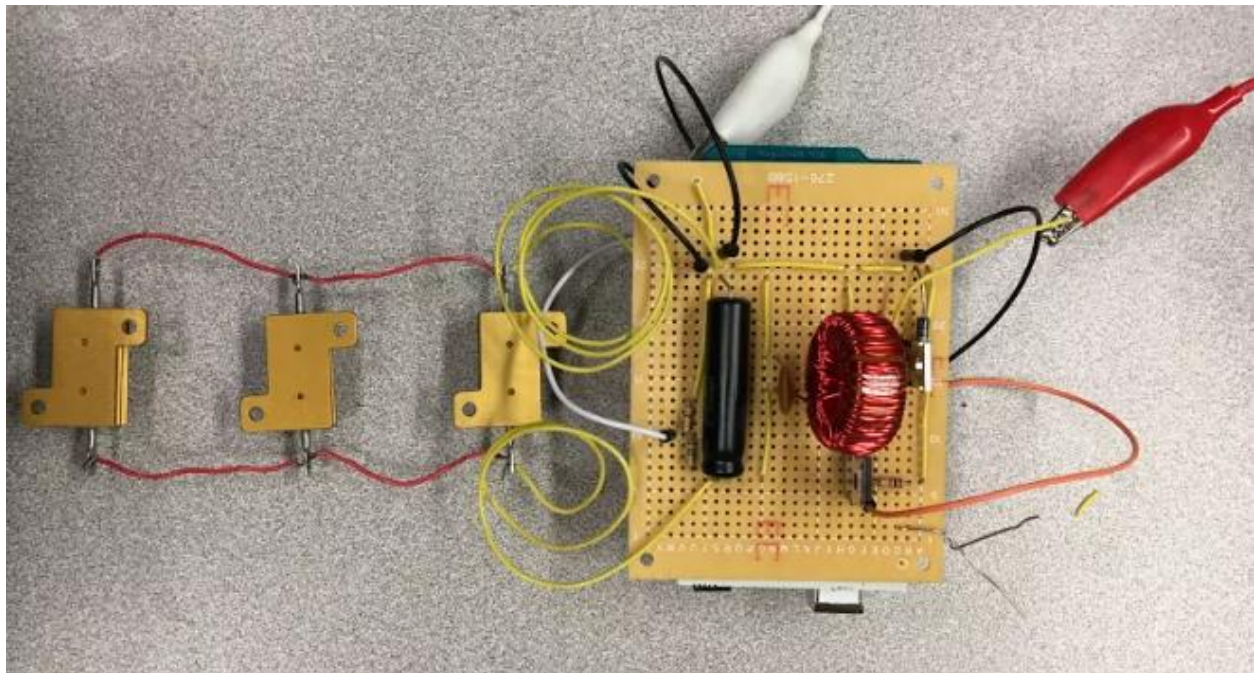


Figure 31 - Circuit side of Prototype

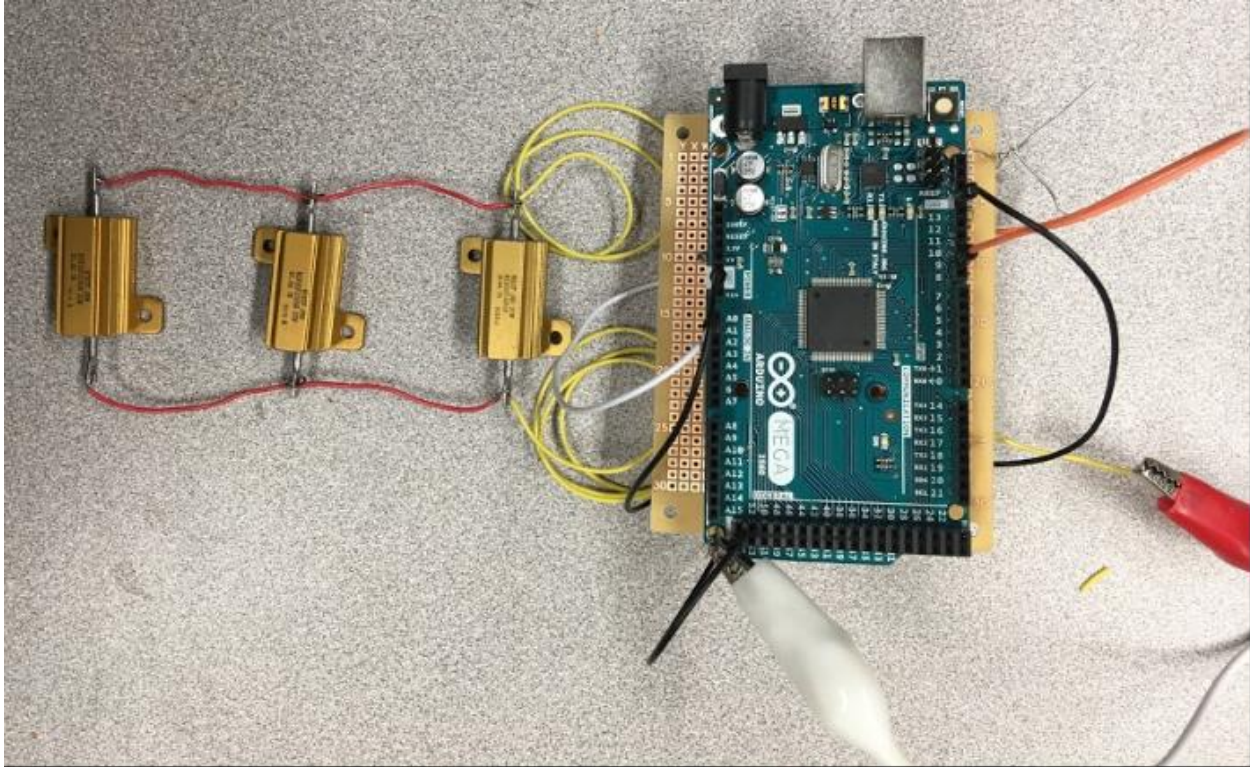


Figure 32 - Arduino side of Prototype

Result

To test out our DC/DC converter, the converter is connected to the PV source and the oscilloscope is connected to the output to display the output voltage. The input voltage of 20V was used.

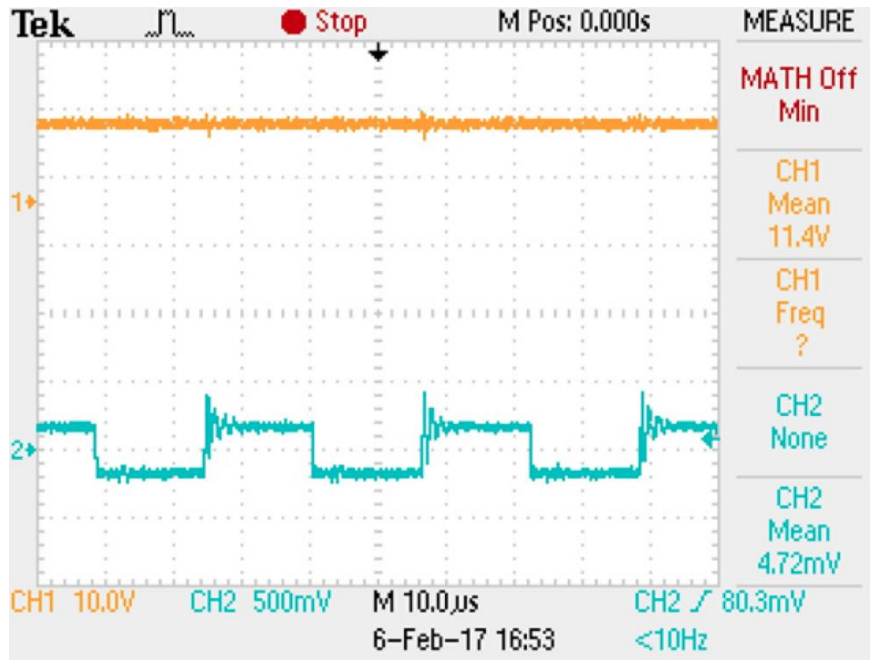


Figure 33 - Output Voltage of DC/DC Converter

As shown above, it successfully stepped down the 20V to roughly 11.4V, which is very close to the expected value of 10V. The ripple of the voltage output was very small, about 500mV.

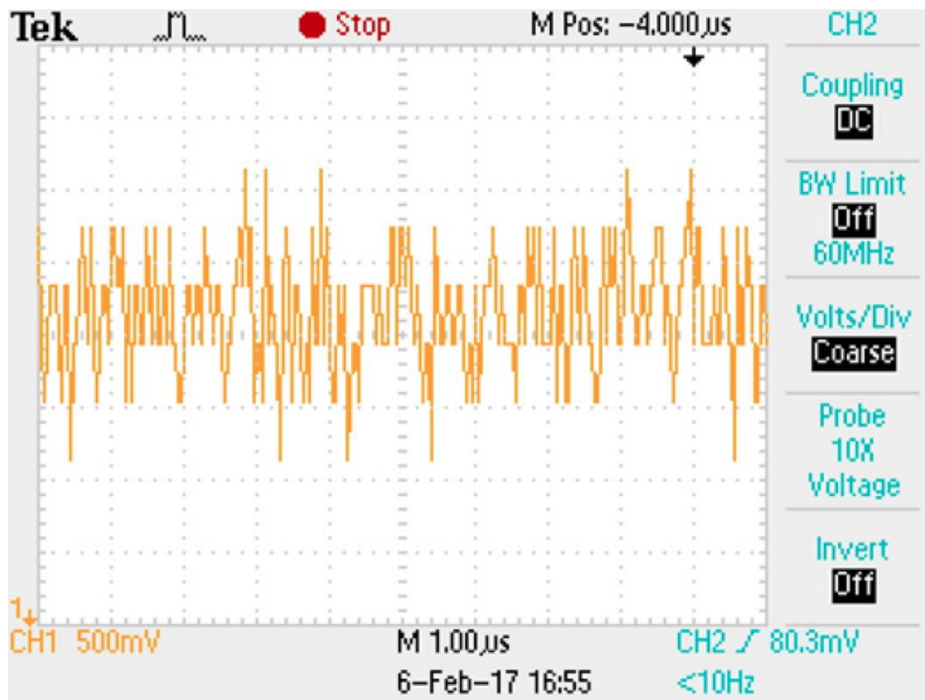


Figure 34 - Ripple of the Output Voltage

To test the final product, all the components were connected together and the Arduino was connected to the laptop with a USB. The code was uploaded to the Arduino through the IDE and the data of the output power was displayed on the laptop screen through the serial monitor feature of the IDE.

With the input of 20V, the voltage and current values were recorded, the power was computed and printed out as shown below. As the duty cycle increased, the voltage and the current increased and so the power output increased.

Voltage/V	Current/I	Power/W
0.00	0.00	0.00
3.01	0.29	0.86
3.26	0.31	1.01
4.30	0.41	1.76

4.87	0.46	2.26
5.91	0.56	3.33
6.99	0.67	4.65
6.99	0.67	4.65
7.74	0.74	5.70
8.42	0.80	6.75
9.78	0.93	9.11
10.57	1.01	10.64
11.64	1.11	12.91
12.29	1.17	14.38
13.29	1.27	16.83
14.33	1.36	19.56
15.30	1.46	22.29
16.16	1.54	24.86
17.05	1.62	27.70
17.84	1.70	30.32
18.02	1.72	30.93
18.02	1.72	30.93
18.02	1.72	30.93
18.02	1.72	30.93
18.02	1.72	30.93
18.02	1.72	30.93

Table 2 – Final Output Data from Arduino

Our final voltage was 18.02V, the current reached around 1.72A and the maximum power was 30.93W. Compare to the simulation results of about 32W, it is shown that our product worked very accurately.

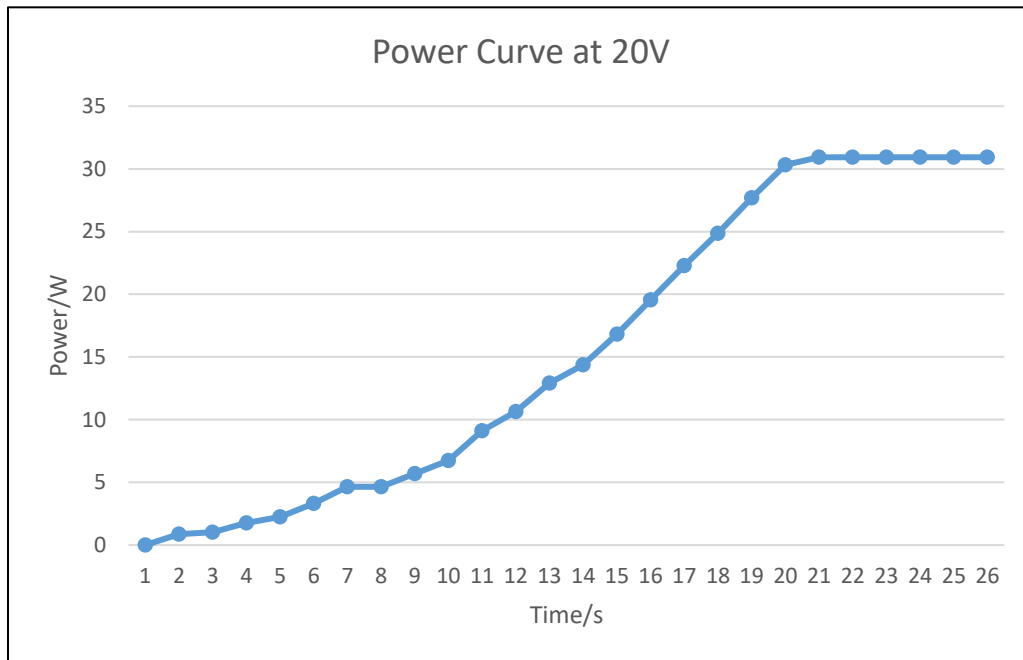


Figure 35 - Output Power Curve at 20V

As seen on the graph, the output power increased until it reached the maximum power and once it reached the maximum power, it stayed constant which proves that our P&O algorithm was successfully implemented. By comparing the final results with the results from the simulation, it is shown that our design worked accurately.

The final output voltage from the oscilloscope is shown below, and it has a mean value of 18.6V:

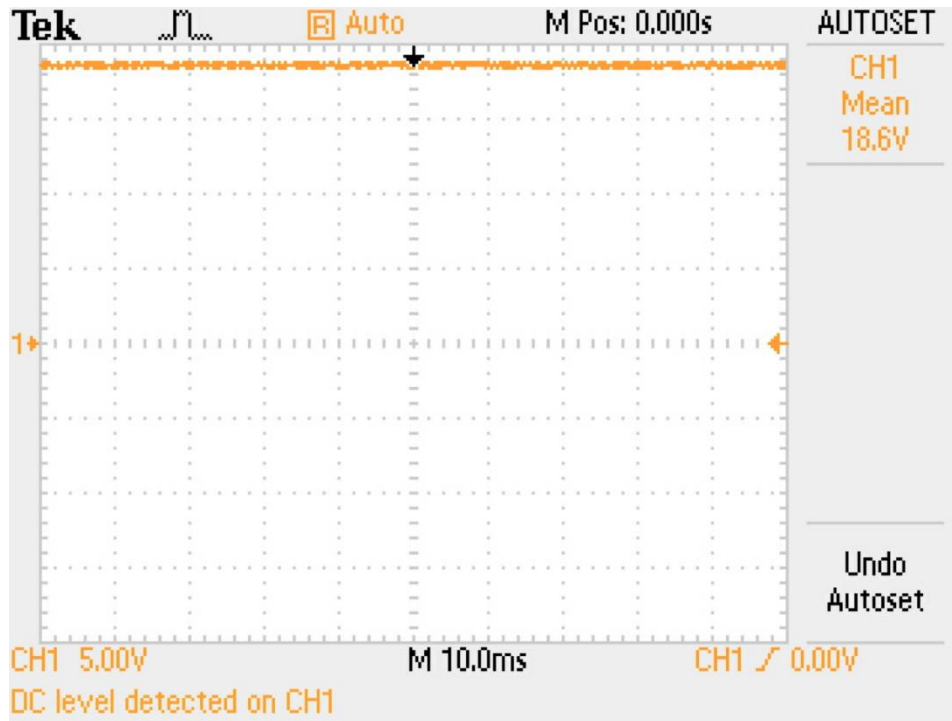


Figure 36 - Final Output Voltage from Oscilloscope

To make sure that our final product worked correctly, various input voltages were tested out and the result is shown below:

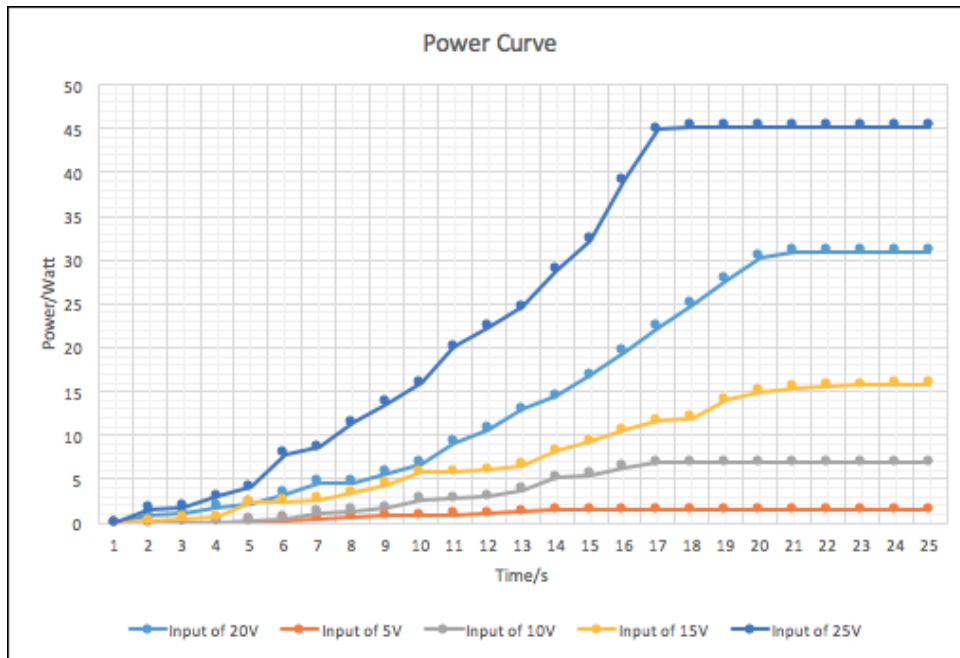


Figure 37 - Power Curve with Various Input Voltages

This result proves that our product correctly calculates the power, tracks and maintains the maximum power point until there is any change in voltage.

Conclusions and Improvement

Over the course of three terms with countless hours of research, applying theory and practice, and after many failures, the prototype finally functions as anticipated. The prototype successfully locates the maximum power regardless of what the solar panel voltage production may be, as long as it is within the range of the 0-25V, since 25V is the limitation of the available laboratory equipment. Hence, this prototype can be used to track the maximum power point of different solar panels and help customers determine the most efficient solar panels available out there in the market, reliably.

Although the product function as expected, there's still some room for improvements. There are few chosen components that were used for the prototype that could be chosen differently for better performance. For example, the load resistor can have the same resistance value but should be able to withstand higher power output. Currently the resistors' temperature can be very high if the maximum power is too high. Another adjustment could be to try out different types of capacitors, with and without parasitic, to see which type of capacitor would be better. Our team had hoped to do this but unfortunately ran out of time. The test can be done using oscilloscopes to monitor their voltage and current behavior. Another improvement would be to use a different microcontroller. The Arduino Mega 2560 uses a language that is very simple to program and there are many available resources, however, the actual chosen hardware was not very compatible with our circuit. So choosing a different microcontroller that better fits the circuit would help improve the accuracy of measuring and calculations as well as simplifying the

overall circuit and the code, as mentioned in design modification section. Lastly, to better the appearance of the prototype, the soldered circuit board should be replaced by a printed circuit board and the user interface should be a LCD screen that can display all the desired values and the user-desired plot. These are all possible improvement points that can be accomplished if there had been more time, or these can be done in the future.

Appendices

A. Overall schematic

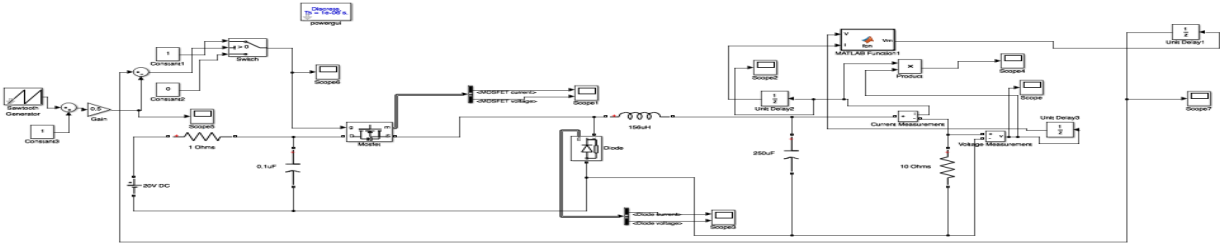


Figure 38 - Final Schematic

B. Arduino Data Sheet



Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V

8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash

DATASHEET

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
 - 135 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 16 MIPS Throughput at 16MHz
 - On-Chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 64K/128K/256KBytes of In-System Self-Programmable Flash
 - 4Kbytes EEPROM
 - 8Kbytes Internal SRAM
 - Write/Erase Cycles:10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/ 100 years at 25°C
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program
 - True Read-While-Write Operation
 - Programming Lock for Software Security
 - Endurance: Up to 64Kbytes Optional External Memory Space
- Atmel® QTouch® library support
 - Capacitive touch buttons, sliders and wheels
 - QTouch and QMatrix acquisition
 - Up to 64 sense channels
- JTAG (IEEE® std. 1149.1 compliant) Interface
 - Boundary-scan Capabilities According to the JTAG Standard
 - Extensive On-chip Debug Support
 - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
 - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
 - Four 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
 - Real Time Counter with Separate Oscillator
 - Four 8-bit PWM Channels
 - Six/Twelve PWM Channels with Programmable Resolution from 2 to 16 Bits (ATmega1281/2561, ATmega640/1280/2560)
 - Output Compare Modulator
 - 8/16-channel, 10-bit ADC (ATmega1281/2561, ATmega640/1280/2560)
 - Two/Four Programmable Serial USART (ATmega1281/2561, ATmega640/1280/2560)
 - Master/Slave SPI Serial Interface
 - Byte Oriented 2-wire Serial Interface
 - Programmable Watchdog Timer with Separate On-chip Oscillator
 - On-chip Analog Comparator
 - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
 - Power-on Reset and Programmable Brown-out Detection
 - Internal Calibrated Oscillator
 - External and Internal Interrupt Sources
 - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby
- I/O and Packages
 - 54/86 Programmable I/O Lines (ATmega1281/2561, ATmega640/1280/2560)
 - 64-pad QFN/NLFF, 64-lead TQFP (ATmega1281/2561)
 - 100-lead TQFP, 100-ball CBGA (ATmega640/1280/2560)
 - RoHS/Fully Green
- Temperature Range:
 - -40°C to 85°C Industrial
- Ultra-Low Power Consumption
 - Active Mode: 1MHz, 1.8V: 500µA
 - Power-down Mode: 0.1µA at 1.8V
- Speed Grade:
 - ATmega640V/ATmega1280V/ATmega1281V:
 - 0 - 4MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
 - ATmega2560V/ATmega2561V:
 - 0 - 2MHz @ 1.8V - 5.5V, 0 - 8MHz @ 2.7V - 5.5V
 - ATmega640/ATmega1280/ATmega1281:
 - 0 - 8MHz @ 2.7V - 5.5V, 0 - 16MHz @ 4.5V - 5.5V
 - ATmega2560/ATmega2561:
 - 0 - 16MHz @ 4.5V - 5.5V

2549Q-AVR-02/2014

1. Pin Configurations

Figure 1-1. TQFP-pinout ATmega640/1280/2560

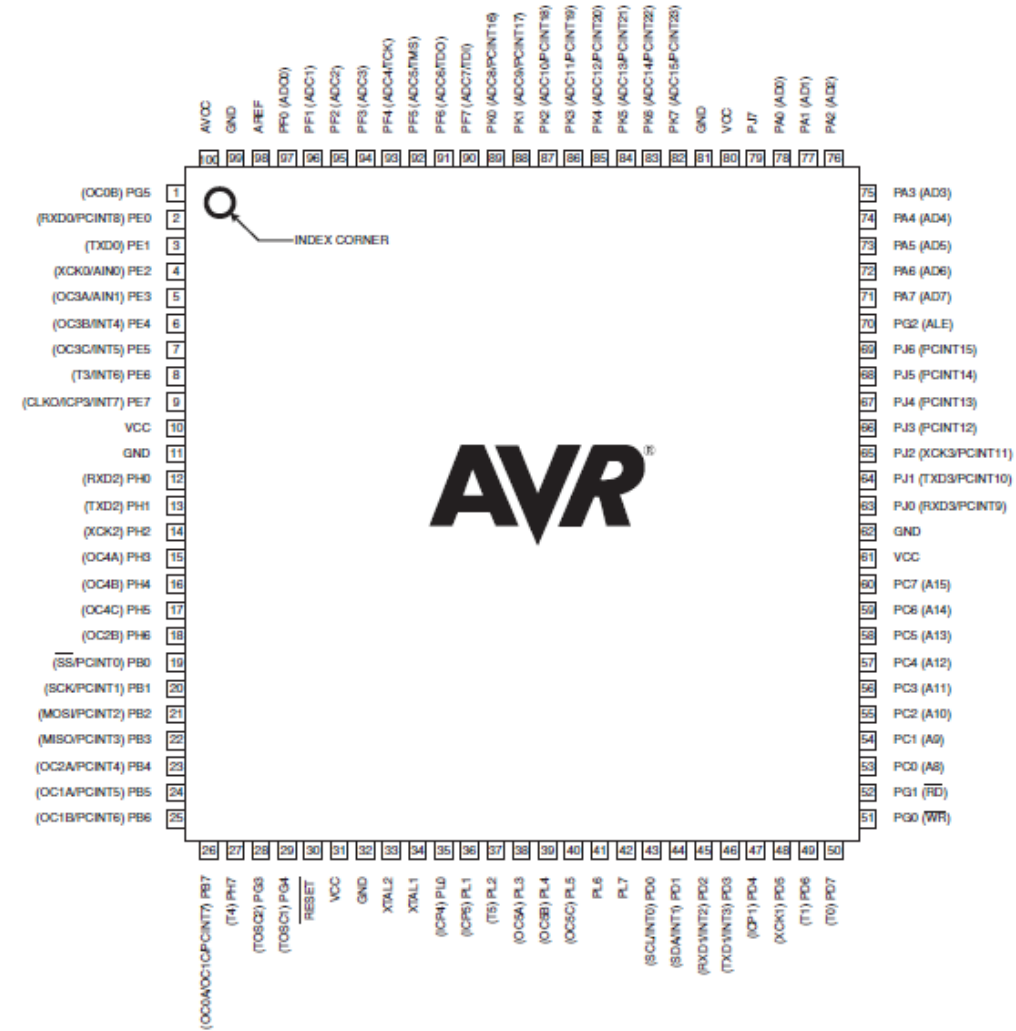


Figure 1-2. CBGA-pinout ATmega640/1280/2560

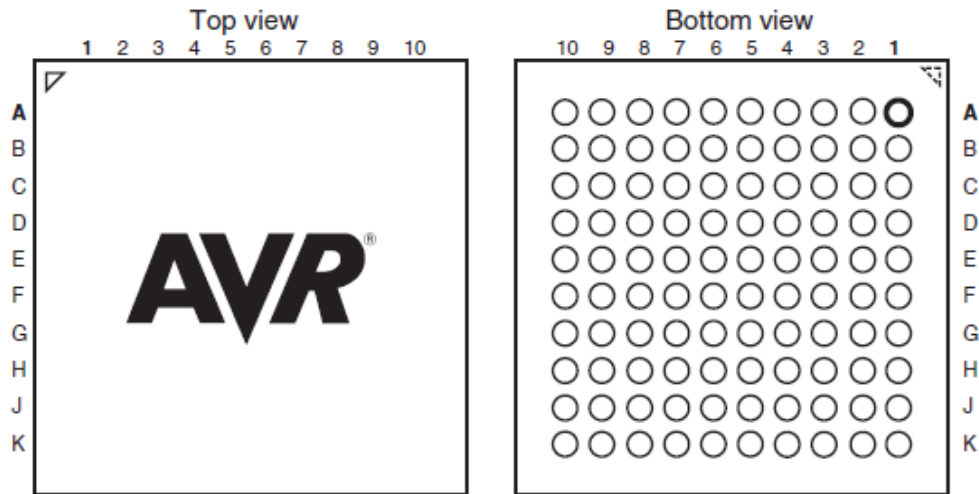
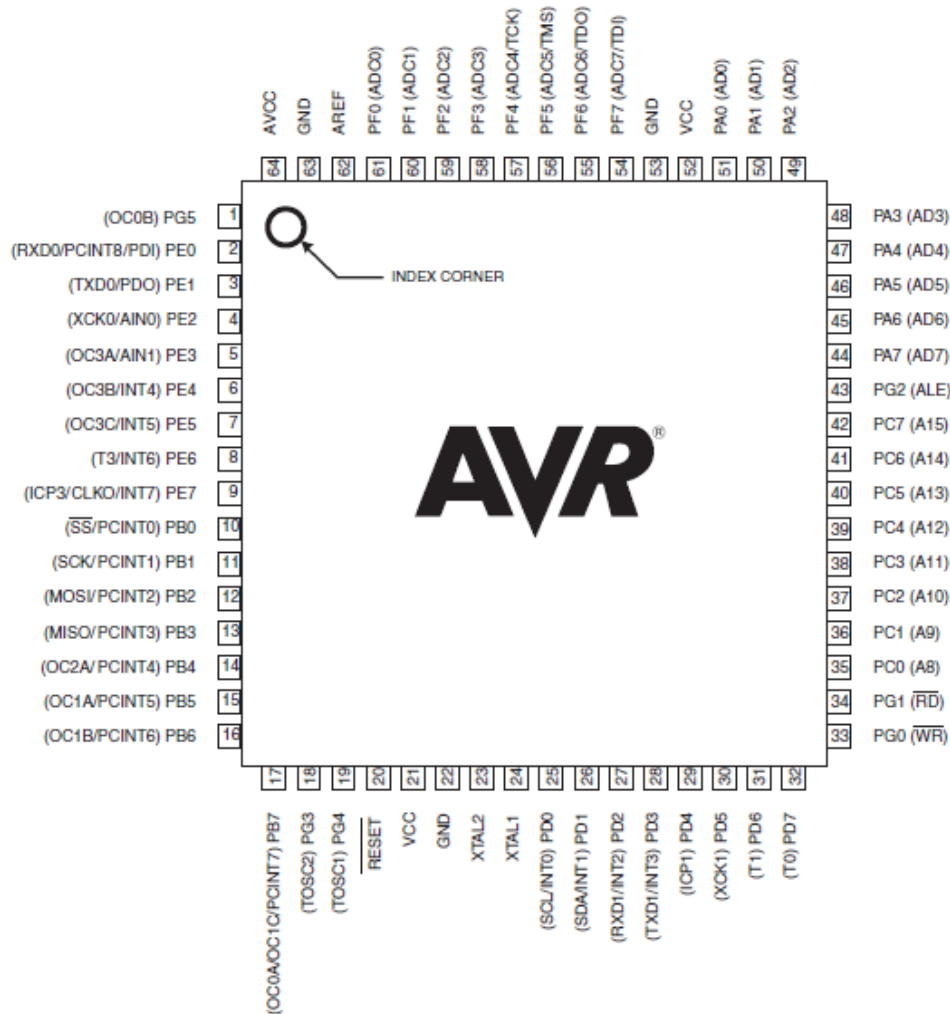


Table 1-1. CBGA-pinout ATmega640/1280/2560

	1	2	3	4	5	6	7	8	9	10
A	GND	AREF	PF0	PF2	PF5	PK0	PK3	PK6	GND	VCC
B	AVCC	PG5	PF1	PF3	PF6	PK1	PK4	PK7	PA0	PA2
C	PE2	PE0	PE1	PF4	PF7	PK2	PK5	PJ7	PA1	PA3
D	PE3	PE4	PE5	PE6	PH2	PA4	PA5	PA6	PA7	PG2
E	PE7	PH0	PH1	PH3	PH5	PJ6	PJ5	PJ4	PJ3	PJ2
F	VCC	PH4	PH6	PB0	PL4	PD1	PJ1	PJ0	PC7	GND
G	GND	PB1	PB2	PB5	PL2	PD0	PD5	PC5	PC6	VCC
H	PB3	PB4	RESET	PL1	PL3	PL7	PD4	PC4	PC3	PC2
J	PH7	PG3	PB6	PL0	XTAL2	PL6	PD3	PC1	PC0	PG1
K	PB7	PG4	VCC	GND	XTAL1	PL5	PD2	PD6	PD7	PG0

Note: The functions for each pin is the same as for the 100 pin packages shown in [Figure 1-1 on page 2](#).

Figure 1-3. Pinout ATmega1281/2561



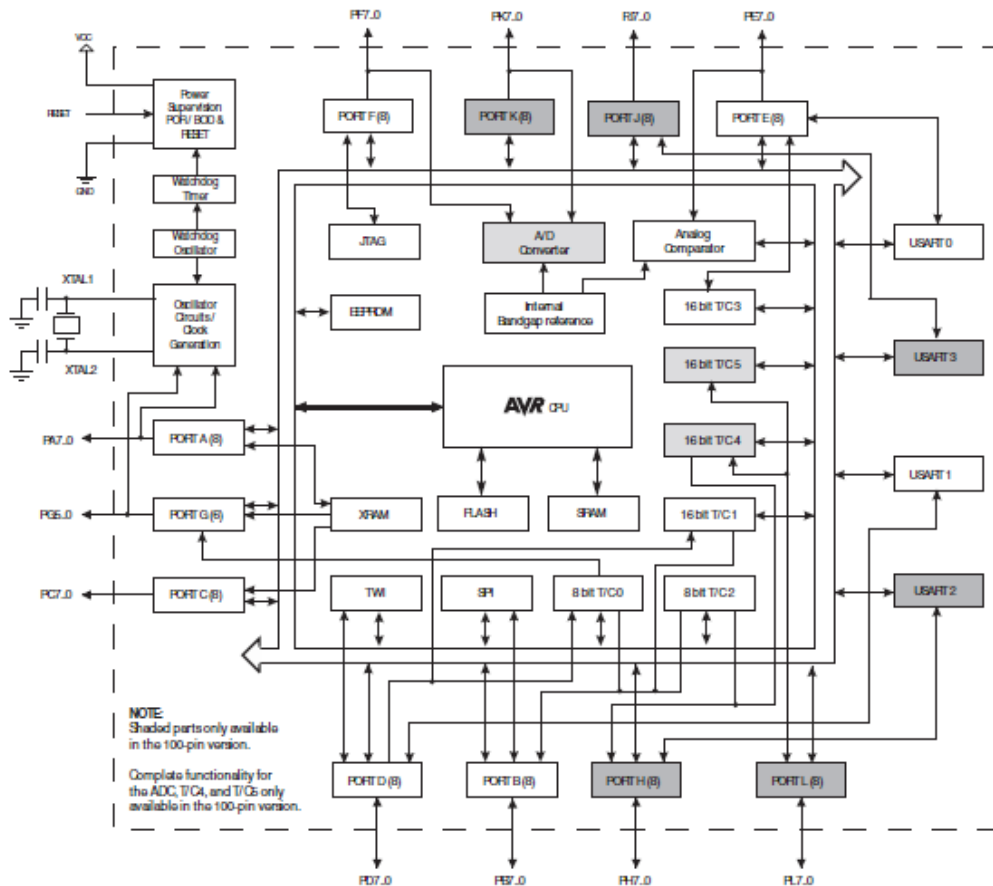
Note: The large center pad underneath the QFN/MLF package is made of metal and internally connected to GND. It should be soldered or glued to the board to ensure good mechanical stability. If the center pad is left unconnected, the package might loosen from the board.

2. Overview

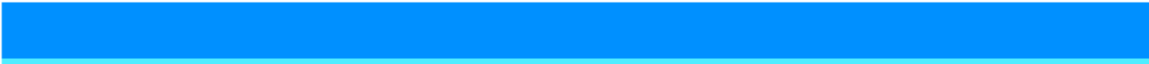
The ATmega640/1280/1281/2560/2561 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega640/1280/1281/2560/2561 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The Atmel® AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.



The ATmega640/1280/1281/2560/2561 provides the following features: 64K/128K/256K bytes of In-System Programmable Flash with Read-While-Write capabilities, 4Kbytes EEPROM, 8Kbytes SRAM, 54/86 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), six flexible Timer/Counters with compare modes and PWM, four USARTs, a byte oriented 2-wire Serial Interface, a 16-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE® std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS®) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications.

The device is manufactured using the Atmel high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega640/1280/1281/2560/2561 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega640/1280/1281/2560/2561 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

2.2 Comparison Between ATmega1281/2561 and ATmega640/1280/2560

Each device in the ATmega640/1280/1281/2560/2561 family differs only in memory size and number of pins. [Table 2-1](#) summarizes the different configurations for the six devices.

Table 2-1. Configuration Summary

Device	Flash	EEPROM	RAM	General Purpose I/O pins	16 bits resolution PWM channels	Serial USARTs	ADC Channels
ATmega640	64KB	4KB	8KB	86	12	4	16
ATmega1280	128KB	4KB	8KB	86	12	4	16
ATmega1281	128KB	4KB	8KB	54	6	2	8
ATmega2560	256KB	4KB	8KB	86	12	4	16
ATmega2561	256KB	4KB	8KB	54	6	2	8

2.3 Pin Descriptions

2.3.1 VCC

Digital supply voltage.

2.3.2 GND

Ground.

2.3.3 Port A (PA7..PA0)

Port A is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on [page 75](#).

2.3.4 Port B (PB7..PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B has better driving capabilities than the other ports.

Port B also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on [page 76](#).

2.3.5 Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of special features of the ATmega640/1280/1281/2560/2561 as listed on [page 79](#).

2.3.6 Port D (PD7..PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on [page 80](#).

2.3.7 Port E (PE7..PE0)

Port E is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port E output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port E pins that are externally pulled low will source current if the pull-up resistors are activated. The Port E pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port E also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on [page 82](#).

2.3.8 Port F (PF7..PF0)

Port F serves as analog inputs to the A/D Converter.

Port F also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port F output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port F pins that are externally pulled low will source current if the pull-up resistors are activated. The Port F pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PF7(TDI), PF5(TMS), and PF4(TCK) will be activated even if a reset occurs.

Port F also serves the functions of the JTAG interface.

2.3.9 Port G (PG5..PG0)

Port G is a 6-bit I/O port with internal pull-up resistors (selected for each bit). The Port G output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port G pins that are externally pulled low will source current if the pull-up resistors are activated. The Port G pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port G also serves the functions of various special features of the ATmega640/1280/1281/2560/2561 as listed on [page 86](#).

2.3.10 Port H (PH7..PH0)

Port H is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port H output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port H pins that are externally pulled low will source current if the pull-up resistors are activated. The Port H pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port H also serves the functions of various special features of the ATmega640/1280/2560 as listed on [page 88](#).

2.3.11 Port J (PJ7..PJ0)

Port J is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port J output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port J pins that are externally pulled low will source current if the pull-up resistors are activated. The Port J pins are tri-stated when a reset condition becomes active, even if the clock is not running. Port J also serves the functions of various special features of the ATmega640/1280/2560 as listed on [page 90](#).

2.3.12 Port K (PK7..PK0)

Port K serves as analog inputs to the A/D Converter.

Port K is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port K output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port K pins that are externally pulled low will source current if the pull-up resistors are activated. The Port K pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port K also serves the functions of various special features of the ATmega640/1280/2560 as listed on [page 92](#).

2.3.13 Port L (PL7..PL0)

Port L is a 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port L output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port L pins that are externally pulled low will source current if the pull-up resistors are activated. The Port L pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port L also serves the functions of various special features of the ATmega640/1280/2560 as listed on [page 94](#).

2.3.14 RESET

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in "System and Reset Characteristics" on [page 360](#). Shorter pulses are not guaranteed to generate a reset.

2.3.15 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.3.16 XTAL2

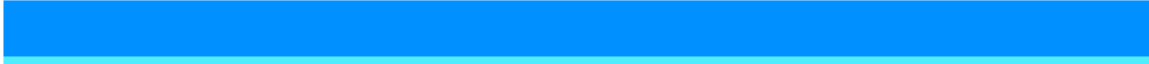
Output from the inverting Oscillator amplifier.

2.3.17 AVCC

AVCC is the supply voltage pin for Port F and the A/D Converter. It should be externally connected to V_{CC} , even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

2.3.18 AREF

This is the analog reference pin for the A/D Converter.



3. Resources

A comprehensive set of development tools and application notes, and datasheets are available for download on <http://www.atmel.com/avr>.

4. About Code Examples

This documentation contains simple code examples that briefly show how to use various parts of the device. Be aware that not all C compiler vendors include bit definitions in the header files and interrupt handling in C is compiler dependent. Confirm with the C compiler documentation for more details.

These code examples assume that the part specific header file is included before compilation. For I/O registers located in extended I/O map, "IN", "OUT", "SBIS", "SBIC", "CBI", and "SBI" instructions must be replaced with instructions that allow access to extended I/O. Typically "LDS" and "STS" combined with "SBR", "SBR", "SBR", and "CBR".

5. Data Retention

Reliability Qualification results show that the projected data retention failure rate is much less than 1 ppm over 20 years at 85°C or 100 years at 25°C.

6. Capacitive touch sensing

The Atmel® QTouch® Library provides a simple to use solution to realize touch sensitive interfaces on most Atmel AVR® microcontrollers. The QTouch Library includes support for the QTouch and QMatrix acquisition methods.

Touch sensing can be added to any application by linking the appropriate Atmel QTouch Library for the AVR Microcontroller. This is done by using a simple set of APIs to define the touch channels and sensors, and then calling the touch sensing API's to retrieve the channel information and determine the touch sensor states.

The QTouch Library is FREE and downloadable from the Atmel website at the following location: www.atmel.com/qtouchlibrary. For implementation details and other information, refer to the [Atmel QTouch Library User Guide](#) - also available for download from the Atmel website.

C. Matlab Code

```
function Vm = fcn(V,I)
%#codegen
persistent prev_PWM prev_P prev_V
if isempty(prev_PWM)
    prev_PWM = 0.5; % set initial duty cycle to be 50% = 0.5
    prev_P = 0;
    prev_V = 20;
end
% Initialize the increment of decrease of the duty cycle
changed = 0.01;
% Calculate the power
P = V*I;
% Increase or Decrease duty cycle depends on the conditions
if (P-prev_P) ~= 0
    if (P-prev_P) > 0
        if (V-prev_V) > 0
            Vm = prev_PWM + changed;
        else
            Vm = prev_PWM - changed;
        end
    else
        if (V-prev_V) > 0
            Vm = prev_PWM - changed;
        else
            Vm = prev_PWM + changed;
        end
    end
end
else
    Vm = prev_PWM;
end
% Update the values
prev_PWM = Vm;
prev_V = V;
prev_P = P;
end
```

D. Arduino Code

```
int PWMPin = 9; //PWM Connect to pin 9
int changed = 12.7; //change duty by 5%
int duty_cycle = 127.0; //start pwm with 50% duty cycle
float Vm;
float P;
float prev_P = 0;
float prev_V;
int sum = 0;
float voltage = 0.0;
```

```

void setup() {
  TCCR2B = TCCR2B & B11111000 | B00000001; //change the pwm pin
frequency to 31.372 kHz
  Serial.begin(9600);
}
void loop() {
  analogWrite(PWMPin, duty_cycle);
  Serial.print(duty_cycle);
  Serial.println("Duty Cycle");
  sum = analogRead(A1); //read the input on analog pin A1
  voltage = ((float)sum * 5.0) / 1023.0; //convert the analog reading
to a voltage
  sum = 0;
  float prev_V = voltage * 7.33; //multiply the voltage divider ratio
  Serial.print("prev_V ");
  Serial.println (prev_V); //print out the voltage
  float prev_I = (voltage * 7.33) / 10.5; //divide voltage by load
resistor to get current value
  Serial.print("prev_I ");
  Serial.println (prev_I); // print out the current

  P_and_O(prev_V, prev_I);
  Serial.print(P);
  Serial.println("Watt");
  Serial.println(" ");
  delay(1000);
}
float P_and_O(float V, float I) {
  P = V * I;
  float changeP = P - prev_P;
  float changeV = V - prev_V;
  if (changeP != 0) {
    if (changeP > 0) {
      if (changeV > 0) {
        if (0 <= duty_cycle && duty_cycle < 241.3) {
          duty_cycle = duty_cycle + changeD;
        }
      }
      else {
        duty_cycle = 254;
      }
    }
    else {
      if (0 <= duty_cycle && duty_cycle < 255) {
        duty_cycle = duty_cycle - changeD;
      }
      else {
        duty_cycle = 241.3;
      }
    }
  }
}

```

```
    else {
    if (changeV > 0) {
    if (0 <= duty_cycle && duty_cycle < 255) {
    duty_cycle = duty_cycle - changedD;
    }
    else {
    duty_cycle = 241.3;
    }
    }
    else {
    if (0 <= duty_cycle && duty_cycle < 241.3) {
    duty_cycle = duty_cycle + changedD;
    }
    else {
    duty_cycle = 254;
    }
    }
    }
else {
    duty_cycle = duty_cycle;
}
prev_V = V;
prev_P = P;
return P;
}
```

E. Bill of Materials

Items No.	Description	Values	Name on Schematics(RefDes)	Quantity	Cost
1	Resistor	750	Resistor	1	0.59
2	resistor	100	Resistor	1	0.59
3	IRF 9520 PMOS	na	Mosfet	1	0.91
4	IRF 520 NMOS	na	Mosfet	1	0.95
5	1N4004 Diode	na	Diode	1	0.13
6	Capacitor	250F	250 μ F	1	3.30
7	Capacitor	0.1 μ F	0.1 μ F	1	0.10
8	Resistors	31.6 Ω	Load Resistor	3	6.00
9	Resistor	1M Ω		1	0.95
10	Resistor	100k Ω		1	0.95
11	4m of coil	na	na	1	1.00
12	metal donuts	na	Inductor	2	1.50
13	Arduino Mega 2560			1	44.25

References

- [1] “Advantages and disadvantages of a solar tracker system” (2015, January). Retrieved November 27, 2016, from <http://www.solarpowerworldonline.com/2016/05/advantages-disadvantages-solar-tracker-system/>
- [2] “Adjusting PWM Frequencies” Arduino Playground - TimerPWMCheatsheet. (n.d.). Retrieved February 16, 2017, from <http://playground.arduino.cc/Main/TimerPWMCheatsheet>
- [3] Alberti, H. E., & Ghani, M. A. (2014, March 26). “Custom Maximum Power Point Tracker Major Qualifying Project” Retrieved October 17, 2016, from https://web.wpi.edu/Pubs/E-project/Available/E-project-032114-223145/unrestricted/Maximum_Power_Point_Tracker.pdf
- [4] “AN6076 Design and Application Guide of Bootstrap Circuit for High Voltage Gate Drive IC” (n.d.). ON Semiconductor. Retrieved November 24, 2016, from <https://www.fairchildsemi.com/application-notes/AN/AN-6076.pdf>
- [5] “Arduino - ArduinoBoardMega2560” (n.d.). Retrieved October 15, 2016, from <https://www.arduino.cc/en/Main/ArduinoBoardMega2560>
- [6] “Arduino - ATmega2560-Arduino Pin Mapping” (n.d.). Retrieved January 17, 2016, from <https://www.arduino.cc/en/Hacking/PinMapping2560>
- [7] “Arduino - ReadAnalogVoltage” (n.d.). Retrieved January 10, 2017, from <https://www.arduino.cc/en/Tutorial/ReadAnalogVoltage>
- [8] “Climate Change” (n.d.). US Department of Energy. Retrieved November 3, 2016, from <http://energy.gov/science-innovation/climate-change>

- [9] Davison, A. (n.d.). “The Latest in Solar Technology: What are the latest solar panels and solar inverters” Alternative Energy. Retrieved November 4, 2016, from <http://www.altenergy.org/renewables/solar/latest-solar-technology.html>
- [10] “DC-DC Converter Tutorial” (n.d.). Maxim Integrated. Retrieved March 22, 2017, from <https://www.maximintegrated.com/en/app-notes/index.mvp/id/2031>
- [11] De Brito, M. A., Sampaio, L. P., Luigi, G., Melo, G. A., & Canesin, C. A. (2011, October 10). “Comparative Analysis of MPPT Techniques for PV Applications” Retrieved October 3, 2016, from <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6036361>
- [12] Deng, Q. (2007, May 3). “A primer on high-side FET load switches (Part 1 of 2)” EE Times. Retrieved March 23, 2017, from http://www.eetimes.com/document.asp?doc_id=1272366
- [13] “Detailed Model of a 100-kW Grid Connected PV Array” (n.d.). Retrieved January 23, 2017, from <https://www.mathworks.com/help/physmod/sps/examples/detailed-model-of-a-100-kw-grid-connected-pv-array.html>
- [14] Harish, A., Prasad, M.V.D. “Microcontroller Based Photovoltaic MPPT Charge Controller” (April 2013). Retrieved from International Journal of Engineering Trends and Technology
- [15] Hart, D. W. (2011). “Power Electronics” P. 196-256. New York: McGraw-Hill.
- [16] “How Solar Panels Work” (2015, December 18). Retrieved October 4, 2016, from <http://www.ucsusa.org/clean-energy/renewable-energy/how-solar-panels-work#.V9L-qJMrKRt>

[17] “IRF520, IRF521, IRF522, IRF523 N-Channel Power MOSFETS” (n.d.). Retrieved October 20, 2016, from <http://docs-asia.electrocomponents.com/webdocs/0031/0900766b80031235.pdf>

[18] “Living off the Grid: Definition” Received February 23, 2017, from Dictionary.com

[19] MacDonald, F. (2016, May 18). “Engineers Just Created the Most Efficient Solar Cells Ever” Science Alert. Retrieved November 2, 2016, from <http://www.sciencealert.com/engineers-just-created-the-most-efficient-solar-cells-ever>

[20] Martin, R. (2016, March 03). “First Solar's Cells Break Efficiency Record” Retrieved February 23, 2017, from <https://www.technologyreview.com/s/600922/first-solars-cells-break-efficiency-record/>

[21] “Microcontroller Tutorial (1/5): What is a Microcontroller?” (2014, December 04). Retrieved March 23, 2017, from <http://www.build-electronic-circuits.com/microcontroller-tutorial-part1/>

[22] “MPPT TRACER 3215RN Solar Charge Controller 30A 12V 24V EP : Renewable Energy” Charge Controllers : Patio, Lawn & Garden. (n.d.). Retrieved March 23, 2017, from <https://www.amazon.com/TRACER-3215RN-Solar-Charge-Controller/dp/B008KWPGAE>

[23] Neitzel, F. (n.d.). “Solar Tracking, Solar Tracking Systems, Solar Trackers | We Catch the Sun” Retrieved March 23, 2017, from <http://www.solar-tracking.com/>

[24] “P-Channel MOSFETS, the best choice for High Side Switching” (n.d.). Retrieved December 3, 2016 from: <http://www.vishay.com/docs/70611/70611.pdf>

- [25] “Perturb & Observe” Archives. (n.d.). Retrieved March 23, 2017, from <http://coder-tronics.com/tag/perturb-observe/>
- [26] Plate, Christopher M., "Maximum Power Point Tracking and solar power for developing countries around the world" (2015). Electrical Engineering Undergraduate Honors ESES. Paper 38.
- [27] “Pulse-width Modulation” (n.d.). Retrieved February 5, 2017, from <https://learn.sparkfun.com/tutorials/pulse-width-modulation/duty-cycle>
- [28] “Renewables to lead world power market growth to 2020” (2015, October 2). Retrieved March 23, 2017, from <https://www.iea.org/newsroom/news/2015/october/renewables-to-lead-world-power-market-growth-to-2020.html>
- [29] “Solar Cell I-V Characteristic and the Solar Cell I-V Curve” (n.d.). Retrieved March 23, 2017, from <http://www.alternative-energy-tutorials.com/energy-articles/solar-cell-i-v-characteristic.html>
- [30] “Solar Panel Comparison Table” (n.d.). Retrieved November 27, 2016, from <http://sroeco.com/solar/table/>
- [31] “Tracker” Solar FlexRack. (2016, November 11). Retrieved February 23, 2017, from <http://solarflexrack.com/products/tracker/>
- [32] “Transistor: Definition” Received February 23, 2017, from Dictionary.com

[33] “Types of Transistors” (n.d.). Retrieved February 27, 2017, from Learnin About Electronics
<http://www.learningaboutelectronics.com/Articles/Types-of-transistors.php>

[34] “1N4001Thru 1N4007: General Plastic Rectifier” (n.d.). Retrieved October 20, 2016, from
<http://global.oup.com/us/companion.websites/fdscontent/uscompanion/us/pdf/microcircuits/students/diode/1n4004-general.pdf>