TEAM A8

FINAL REPORT

Gursewak Braitch - 21730163
Chen Wang - 17130162
Yuedong (Rick) Cheng - 24250169
Cheung Hiu Lok - 51149169
Harmanveer Cheema - 18842161

TABLE OF CONTENTS

1. List of Figures

2. List of Tables

3. Project Description

Project Description
The objective of this project is to design and build a small-scale wind turbine capable of maximizing the output power by tracking the wind direction and controlling the rotational speed of the blades within the span of one semester.

Requirements
- Deliver 12V DC (10-15V)
- Rotates a minimum 90°around vertical axis in response to change of wind source direction
- Maximum dimensions of 50cm x 50cm x 60cm
- Be able to maintain output voltage at different wind speeds
- Be able to control the rotational speed of the blades
- Maintains maximum output power under changing wind direction and speed
- Final revision cannot be on a breadboard

Constraints
- Sizes / volume of the wind turbine is set in requirements
- Budget of 500 Canadian dollars
- Availability of components from digikey
- Only 5 waterjet laminations

Goals(all met)
- To create a wind-turbine capable of producing and maintaining output of 2 W
- To design and implement sensors which can detect wind direction
- To create a feedback loop that takes inputs from the wind direction sensor and adjusts the fan direction to attain maximum power
- To be able to change blade speed using boost converter

4. Tasks

4.1 Generator

Overview

The electric machine used in this project is a Permanent Magnet Synchronous Generator (PMSG). The machine utilizes neodymium magnets to create a rotating magnetic field that is used to generate 3 phase electric power.

Requirements

The generator needs to be capable of supplying sufficient current so that the output voltage isn't limited when an appropriate load is attached. The output voltage also needs to be high enough so that it can be stepped up to the required 10-15V with minimal efficiency losses. The core of the generator must be made of some material with high magnetic permeability.

Voltage Generation

The voltage generated by a changing magnetic flux through a coil can be shown by Faraday's law

$$emf = -N\frac{d\phi}{dt}$$

where is the number of turns and is the magnetic flux through the coil.

We approximate the flux coming off the permanent magnets as

$$\phi = BA$$

where B is the surface magnetic flux density and A is the area of the magnet being used.

Substituting the second expression into the first, we obtain

$$emf = -N\frac{d(BA)}{dt}$$

which when averaged over time becomes

$$emf_{avg} = -\frac{NA}{\Delta t}\left(B_{final} - B_{initial}\right) = -\frac{2NAB_{\max}}{\Delta t}$$

where Δt is the time taken for opposite poles to reach the same position.

The design features 6 rotor poles, and 9 stator poles. Each rotor pole consists of 3 Neodymium magnets in series. Each stator pole consists of 70 turns, with 3 coils in series for a total of 210 turns per phase. The flux rating of each magnet is with an area of

$$25.4 \times 10^{-3}\, m \ \times 12.7 \times 10^{-3}\, m = 32.258 \times 10^{-5}\, m^2$$

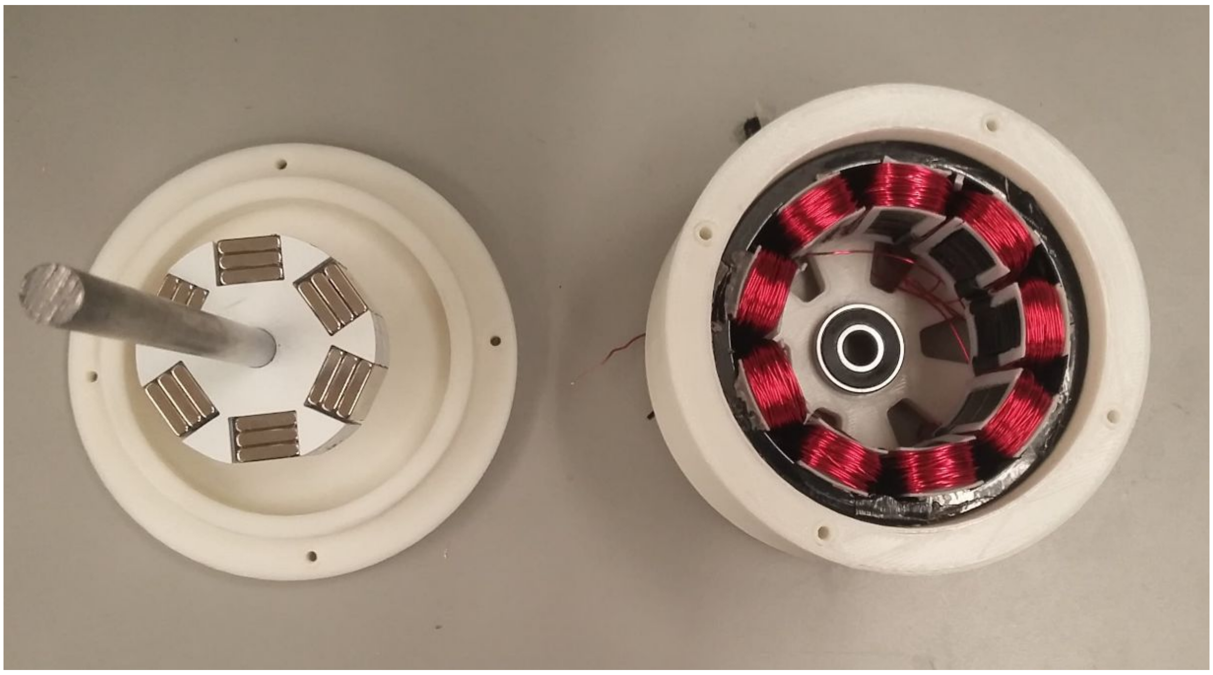


Figure 1: Permanent Magnet Synchronous Generator

Evaluating at these values at 330 rpm, we see that

$$emf_{avg} = 6.354\, V$$

for each phase. This value is very close to what we achieved at the same speed.

The stator is made of steel laminations that have been waterjet cut to reduce magnetic reluctance.

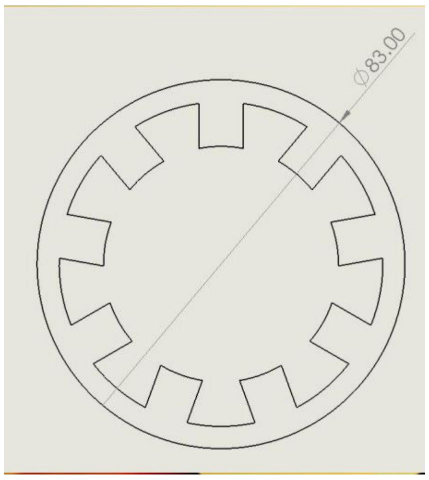


Figure 2: Stator Waterjet Design

Constraints

Due to the physical nature of the generator there are many constraints that have to be worked around.

The power output of the generator is directly proportional to the size of the blades which is limited by the given volume of 50cm × 60cm × 50cm.

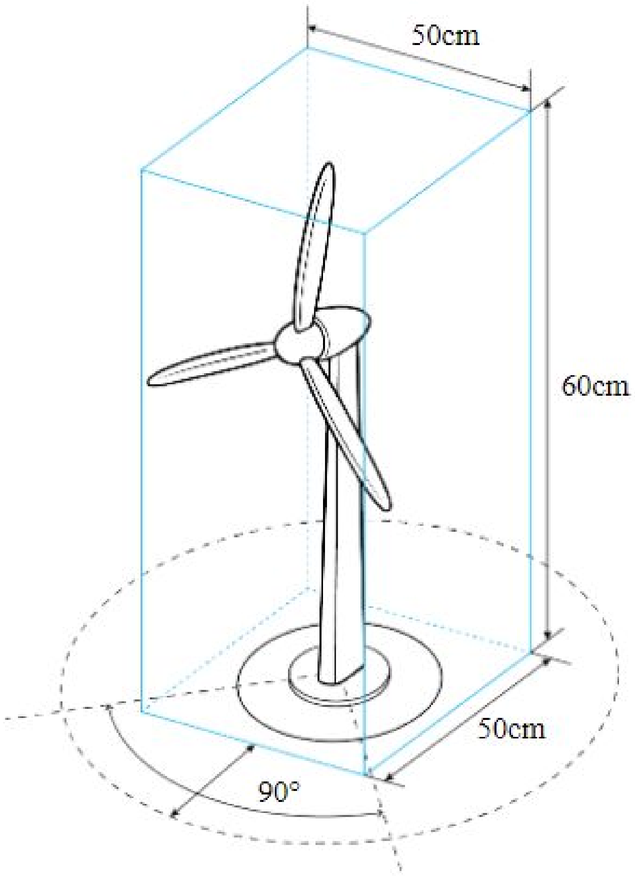


Figure 3: Turbine Size Constraint

The power output of the generator is also proportional to the cube of wind velocity, which is supplied by small fans.

$$P_{wind} = \frac{1}{2}Av^3\rho$$

where A is the area the blades sweep, v is the wind velocity and ρ is the air density.

The Betz limit constrains our energy transfer efficiency to a theoretical maximum of 59.3%.

Leakage flux and mechanical friction results in further inefficiencies in energy conversion when converting energy from the mechanical domain to the electrical domain.

Air gaps in the generator limit transmission of flux due to their high reluctance.

Design Validations

There are many instances where certain characteristics offer tradeoffs and optimization is needed.

Magnets tend to stick to the ferromagnetic core, making it difficult for the rotor to rotate, this is known as magnetic cogging. Due to our magnets being stacked on each other, it is very significant. This cogging is dependent on the air gap between the magnets and the ferromagnetic material.

As shown earlier, the voltage generated is proportional to

$$\frac{d\phi}{dt}$$

Increasing the air gap decreases cogging and therefore increases the speed of rotation which increases the voltage generated. However, an increase in air gap also increases magnetic reluctance, resulting in a decrease in flux and furthermore the voltage generated.
To optimize this problem, we tested the performance of our generator, utilizing rotors with a series of different air gaps and utilized the one with the best performance. In this case, our air gap is 2.5mm resulting in a phase reluctance of $19.25*10^6H^{-1}$.

Although the air gap had been optimized for speed, a major problem was the start-up torque needed to start the rotation. In order to improve the torque generated by the wind, we went with a 5 blade design. The 5 blade design provided just enough torque to start up the generator without having to give it a manual push.

The voltage generated is also proportional to the number of turns N. N is dependent on the amount of magnetic wire utilized. Due to the impedance of the magnetic wire, when we increase the number of turns N, the impedance also rises, limiting the current that we can supply. The size of the wire affects its impedance, as wire with a larger cross section has a lower impedance. This however, impacts the number of turns that is possible in a confined space.To optimize this, we used a length of wire with the lowest AWG rating possible that would generate just enough voltage to fulfill our requirements. We used 26 AWG resulting in a phase impedance of just 3Ω.

The generator was wired in Y-formation due to its simplicity, and tests not showing improvement when delta formation was used.

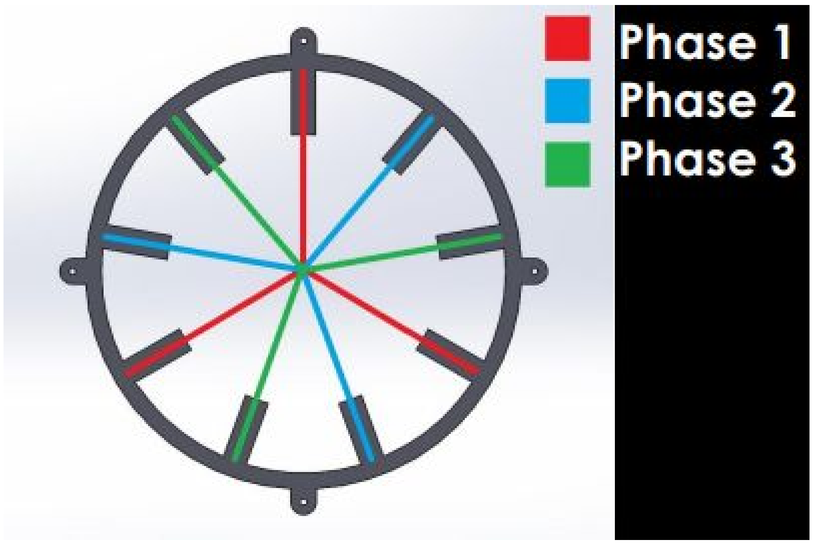


Figure 4: Phase Diagram

4.2 Power Electronics

Overview

The main goal of this subtask is to transform the n-phase AC voltage waveforms from the generator and convert them into a steady 10-15V DC. This involves a two-stage conversion: an n-phase rectifier and a boost converter.

The purpose of the n-phase diode bridge is to rectify the AC sinusoids into only positive peaks. This positive voltage will be supplied to the boost converter which will step the voltage up to the required range of 10-15V DC.

Requirements

For the n-phase rectifier, the only requirement is that it converts the positive and negative peaks of the generated sinusoid into purely positive peaks. This way, only positive voltage will be supplied to the boost converter ensuring that reverse voltage polarity will not damage any of the boost converter components.

There are four major requirements that the boost converter must meet: the output voltage must be between 10-15V, the input voltage must be lower than the output voltage, the converter allows for some form of current control, and the boost converter must be able to support a resistive load.

The output voltage must be kept between 10-15V for proper operation of the MPPT algorithm. The general idea is that the PWM duty cycle will be changed so that the maximum power can be extracted from the generator while maintaining 10-15V.

The current control requirement is so that the controller can change the PWM duty cycle to the boost converter so that maximum power can be extracted from the generator and consequently, the blade speed will be controlled to extract the maximum power from the wind.

Goals

There are several goals for this subtask. The main subtask goal is to reduce losses of the rectifier and the boost converter. As one of the main goals of the final project is to extract maximum power from the wind, decreasing power losses in the components will help achieve that goal.

Another goal is to ensure modularity of the rectifier and boost converter. As the voltage and current sensors will most likely not be ready, there needs to be a simple way of connecting the sensors in the correct positions.

3-Phase Rectifier

The generator is designed as a 3-phase PMSG so the 3-phase rectifier schematic is shown below.

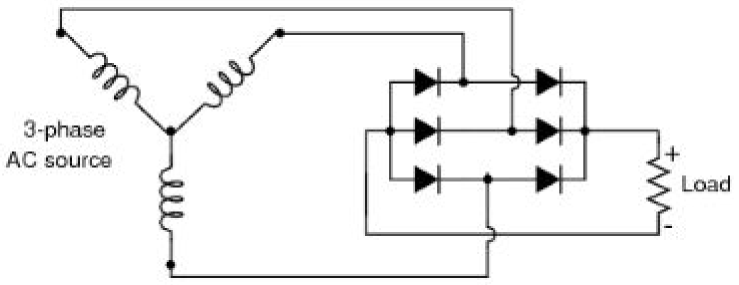


Figure 5: 3-Phase Rectifier Schematic

The three phases of the generator are connected to the three-terminal input of the rectifier and the two terminal output will be connected to the boost converter. A capacitor was placed in parallel to the output to smooth the output and provide a continuous input to the boost converter.

There were two choices of components in place of the diodes: regular diodes or thyristors. The component choice will dictate the cost and complexity of the rectifier.The use of thyristors would require 6 separate control signals; these signals would be dependent on the frequency of the generated voltage sinusoid. In addition, the thyristor controls would have to be integrated with the MPPT. These two problems add many levels of complexity to the system and tuning the control signals would be difficult given the time frame of the project.

Due to the reasons above, I decided that the best choice is a passive rectifier. Also, it is best suited to our needs and would allows us to better meet the time constraint.

Boost Converter

The schematic of the most common configuration of a boost converter is shown below.

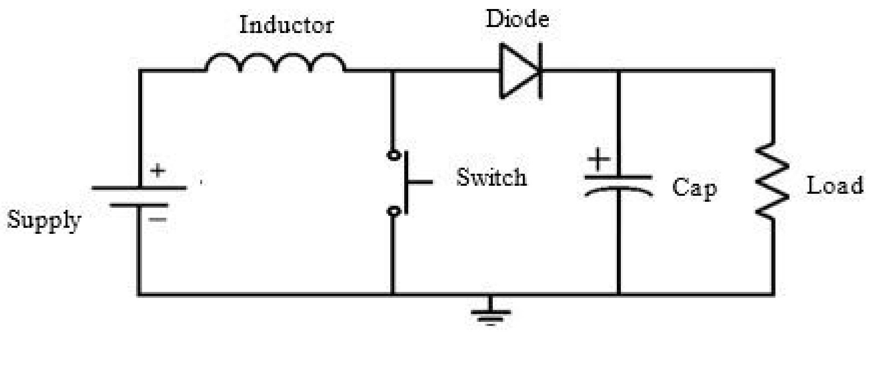


Figure 6: Boost Converter Schematic

There is a component choice as well. The diode can also be changed to a MOSFET that would be complementary to switch. This would allow for better control of converter but would add more control complexity. The diode was chosen due to cost, decreased control complexity, and the fact that the boost converter will be operating in constant conduction mode for this project.

The component value calculations below are used as a general guideline for the first prototype.

Boost Converter Parameters

$$V_{in} = 6 - 12V$$
$$V_o = 12V$$
$$f_{sw} = 100kHz$$
$$R_{o,min} = 15\Omega$$
$$\Delta i_L = 20\%$$
$$\Delta V_o = \pm 5\%$$

Component Value Calculations

$$D = 1 - \frac{V_{in}}{V_o} = 1 - \frac{V_{in}}{12V} \boxed{= 50\%\ (6V\ input) \quad 0\%\ (12V\ input)}$$
$$I_L = \frac{V_o}{R_o V_{in}} = \frac{12V^2}{15\Omega * 6V} = 1.6A$$
$$\Delta i_L = 0.2 * I_L = 0.32A$$
$$L = \frac{V_{in}D}{\Delta i_L f_{sw}} = \frac{6V * 0.5}{0.32A * 100kHz} \boxed{= 93.75\mu H}$$
$$C_{min} = \frac{V_{in}D}{\Delta V_o (1-D) R_o f_{sw}} \boxed{= 13.3\mu F}$$

An LTSpice simulation is completed to give me an idea of whether the calculated component values would work properly. Further tuning is required as the final boost converter parameters may not be the same.

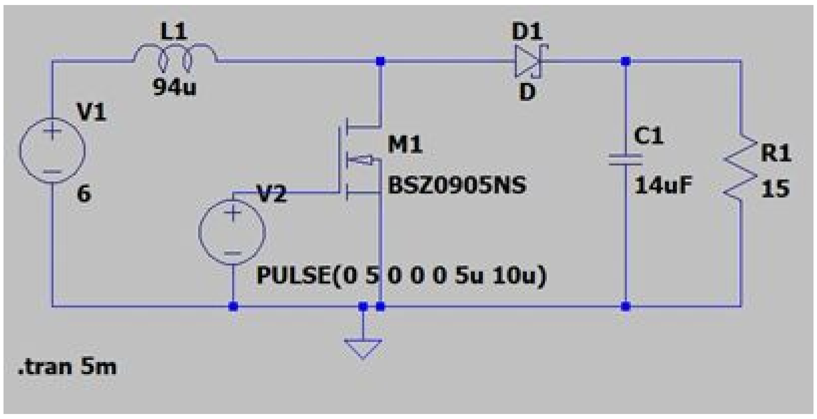


Figure 7: Boost Converter Simulation Circuit

The first simulation is with the calculated parameters, the duty cycle at 50% and the input voltage of 6V.

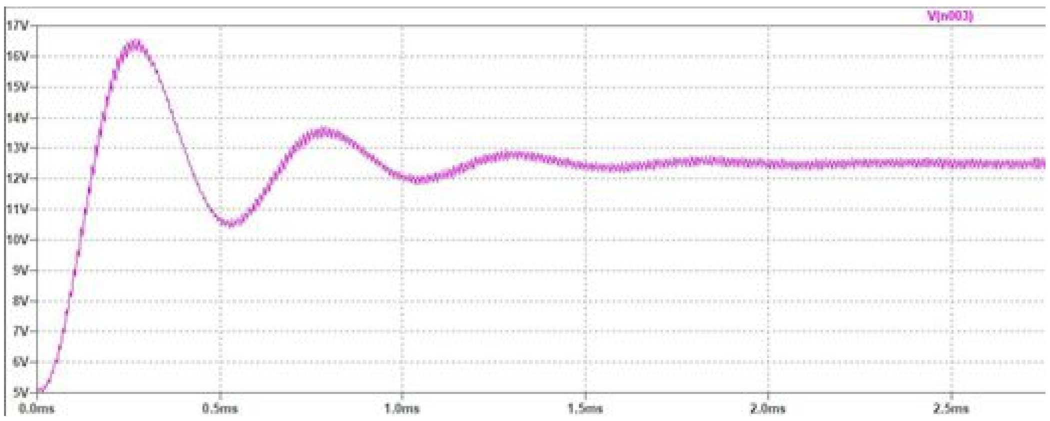


Figure 8: Boost Converter Simulation (C = 14µF)

In the waveform, the output voltage settles to ~12V after the transient as expected. The output voltage ripple is very evident (as I used the capacitor value for 5% ripple) so the simulation shows that the calculated values do work. I increased the capacitor value to 200µF in order to decrease the output voltage ripple.

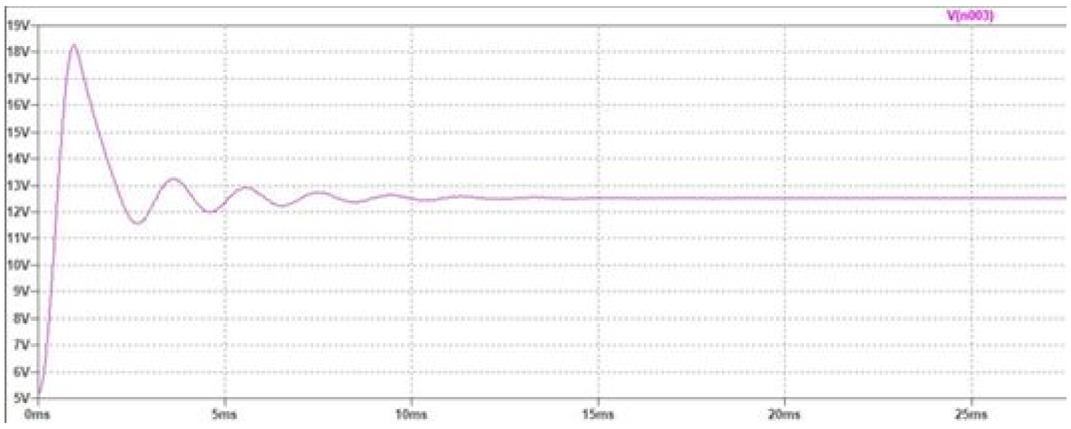


Figure 9: Boost Converter Simulation (C=200µF)

Now, the output voltage ripple was removed since the capacitor discharges across the load when the MOSFET switch is on and charges when the MOSFET is off.

First Prototype

The first prototype is on perf board as it is very simple to create a circuit and the circuit would not suffer from parasitic capacitances of the breadboard. This allows the high switching frequency as well as a higher safe current. This prototype was used for preliminary integration with the generator before the PCB is completed.

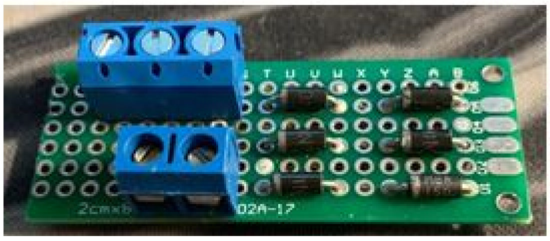


Figure 10: 3-Phase Rectifier Prototype 1

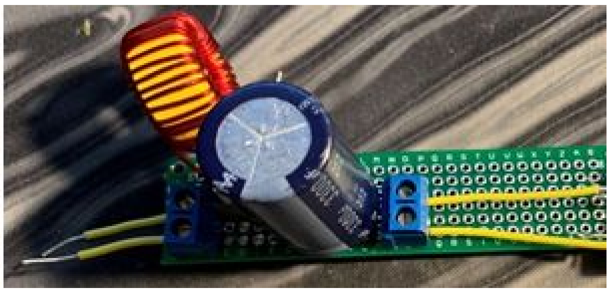


Figure 11: Boost Converter Prototype 1

The input of the rectifier is the 3-hole screw terminal and the output is the 2-hole screw terminal. These connect to the PMSG and the boost converter respectively. The input of the boost converter is the 2-hole screw terminal on the left and the output is the 2-hole screw terminal on the right. These connect to the 3-phase rectifier and the resistive load respectively.

The 3-phase rectifier was tested using the signal generator and measuring the output voltage which does output a constant DC voltage. The boost converter was tested using a power supply and the signal generator to generate the PWM required for boost converter to operate.

PCB Design

The PCB is designed using Altium Designer. First, the two schematics were created, and then components are chosen. However, some components are not available in the Altium library so I substituted those with components with the same footprint as the required component. The two schematics are shown below.

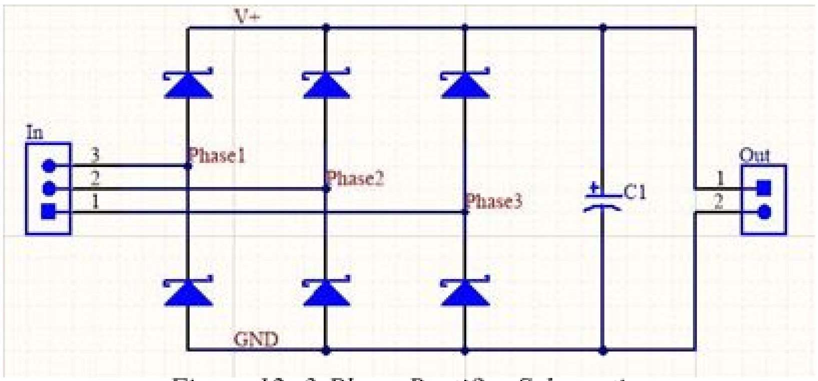


Figure 12: 3-Phase Rectifier Schematic

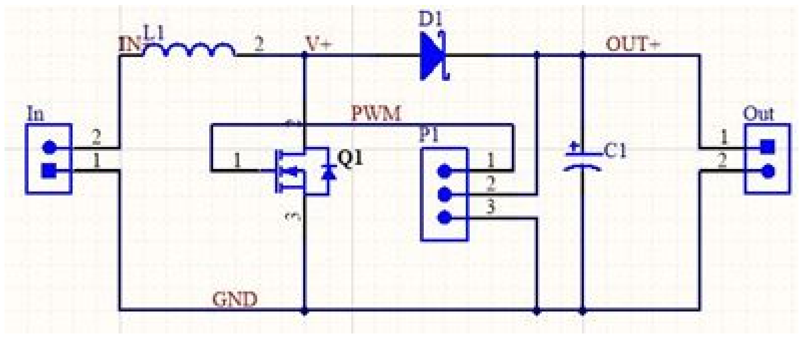


Figure 13: Boost Converter Schematic

An additional 3-pin male header is placed on the board which are connected to the gate of the MOSFET to supply a PWM signal, the positive output, and ground. This will simply the tasks of controlling the MOSFET and measuring the output voltage.

Then, both schematics are exported to a PCB file and after placement, the final PCB layouts are shown below.

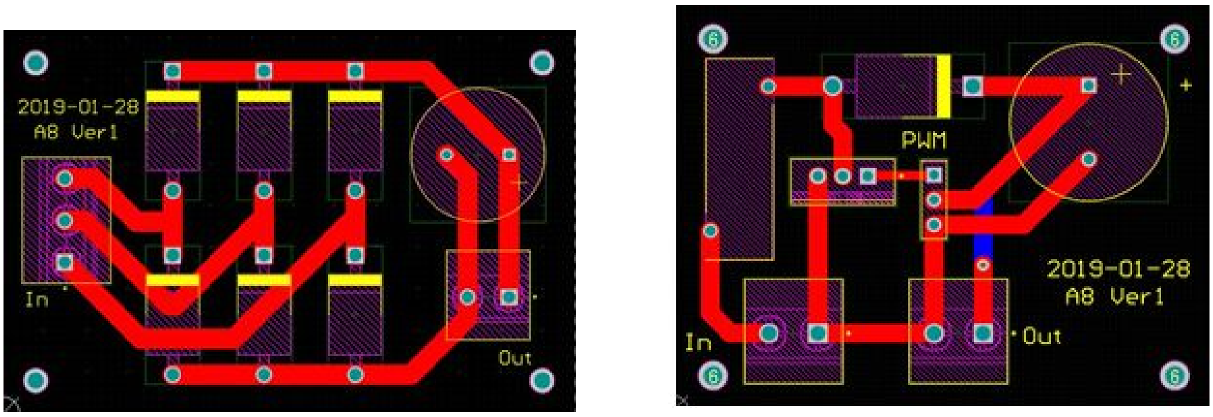


Figure 14: Final PCB Layout

PCB and Component Choices

Two separate boards are made so the voltage and current sensors can easily be placed in the required locations. In addition, any error in the board design will only require a smaller PCB to be remanufactured rather than a larger board if the rectifier and boost converter were combined. This satisfies the subtask goal of a modular design.

Screw terminals were used to facilitate the modularity of the two boards. The capacitors for both the rectifier and boost converter are very large to eliminate possible voltage ripples on the output. In addition, the diodes were changed to higher rated Schottky diodes. The reason for this was to decrease the power loss through the two stages and deliver maximum power to the resistive load. This satisfies the subtask goal of minimizing losses to extract maximum power.

Final PCB Testing

The 3-phase rectifier is tested using the supplied external rotor generator. The output was measured on the oscilloscope and was a steady DC voltage.

The boost converter is tested using an external power supply, the signal generator for the PWM signal, and a 45Ω resistive load. The three plots below were produced to give a general idea of operating point of the boost converter. This information was passed to the generator and controls subtasks for a general idea of the requirements to operate the components.

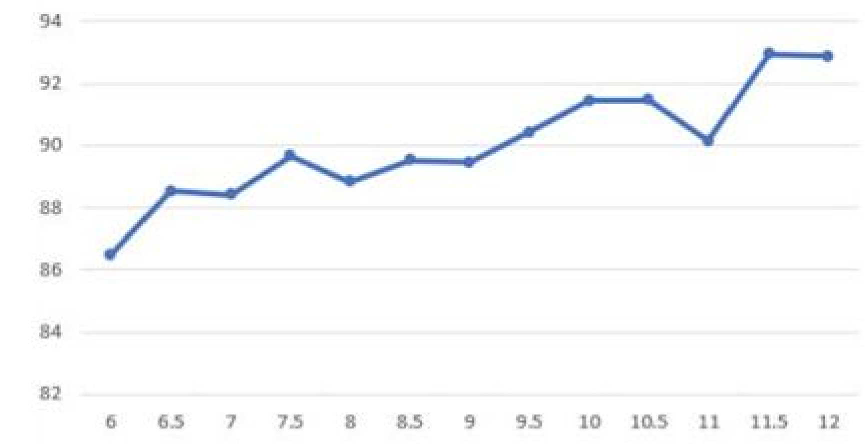


Figure 15: Input Voltage(V) vs. Output Voltage(V)

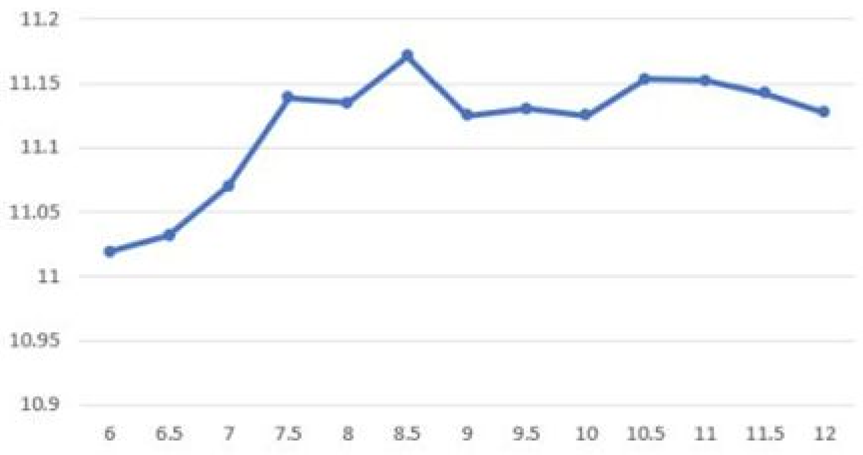


Figure 16: Input Voltage(V) vs. Efficiency (%)

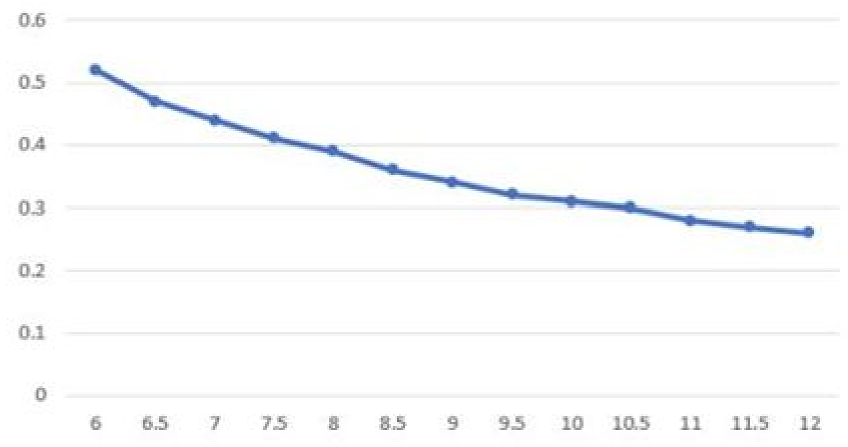


Figure 17: Input Voltage(V) vs. Input Current (A)

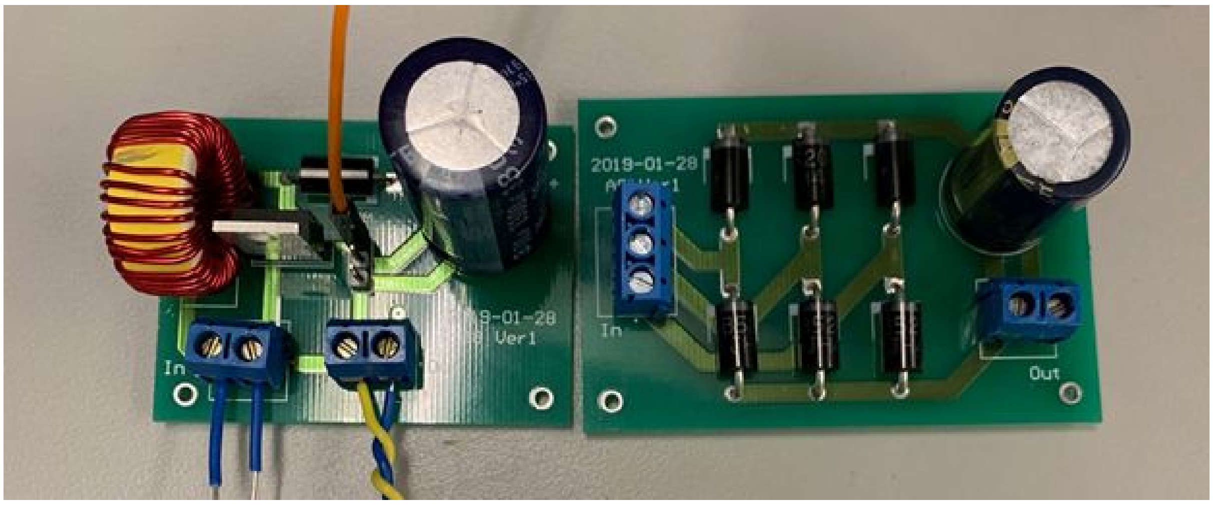


Figure 18: Final 3-Phase Rectifier and Boost Converter PCBs

Budget

The allotted budget is ~$50.00 for the components and ~$160.00 for the PCB manufacturing. The total cost of both PCBs and components (used and unused) is well within budget. The table below lists all costs of the subtask.

Component	Quantity	Cost
SB520 (Schottky Diode)	12	$9.07
3-Hole Screw Terminal	2	$1.85
Heat sink	2	$2.02
IRF540N (MOSFET switch)	2	$3.70
50V 1000µF Capacitor	2	$3.86
3-pin Male Header	5	$0.95
PCBs	NA	$109.93
Extra Parts	NA	~$30.00

Allotted Budget: $210.00
Total Cost: $161.38

Table 1: Total Cost of Power Electronics Completion

4.3 Sensing and Modelling

Overview

For this subtask, the main objective is to create two control loops, one for MPPT and another for wind direction tracking. To achieve this, current and voltage sensors must be implemented to measure power; and a wind direction sensor to perceive the direction of wind source to the wind turbine. These sensors are to be used in feedback loops.

The control loops are first simulated with Simulink to maximize effectiveness and to try different strategies of implementation. They are used for development before implementing on the wind turbine.

Requirements

The voltage sensor must be able to measure DC voltage coming from the generator after rectifying, and handle a range of 10 to 15V. It must convert the voltages to analog signals that are sent to the controller as one of the raw data for MPPT calculations.

The current sensor must be able to measure DC current coming from the generator after rectifying, and handle a range of 0-2A. It must convert the current to analog signals that are sent to the controller as one of the raw data for MPPT calculations.

The wind direction sensor is required to detect incoming wind source at least 90° around vertical axis, and generate analog signals to the controller to convert to position for wind direction tracking.

Subtask Goals

The main goal for sensing and modelling is to maximize the accuracy and precision of the sensors, and to give inputs for the Controls subtask to create the MPPT and wind direction tracking control loops.

A secondary goal is to reduce the power used by the sensors, to increase efficiency of the final project. Lastly, the final goal is to achieve modular sensing components, separating the voltage and current sensors and the wind direction sensor. This is to ensure swapping of components can be easily done if a certain part malfunctions.

Voltage Sensor

The voltage sensor is a voltage divider using 300kΩ and 75kΩ resistors. The schematic is shown below:

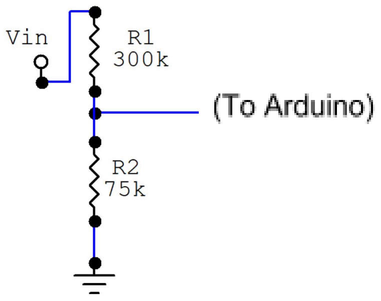


Figure 19: Voltage sensor schematic

The Arduino microcontroller is connected to the node between the 300kΩ and 75kΩ resistors, where voltage will be scaled by equation:

$$V_{arduino} = V_{in} \cdot \frac{R_2}{R_1+R_2} = V_{in} \cdot \frac{75k}{300k+75k} = V_{in} \cdot 0.2$$

From the equation, it can be seen that for any input voltage Vin, the voltage to the Arduino will be scaled by 0.2. So for the maximum required voltage range of 15V, the voltage input to Arduino would be 3V, well within the maximum of 5V for an Arduino ADC pin. Hence, the expected range for Vin (0-15V) will be scaled to (0-3V) to the microcontroller.

Using a test voltage from a voltage source and readings from the Arduino, a graph was made to ensure the linearity of the voltage divider and to measure the accuracy of the sensor. It was found that the measured voltage was linear with R^2 value of 0.9999, and that there was an average of 3.28% error in the measured scaled voltage compared to the expected. Hence for the expected voltage range, the voltage sensor is accurate to 0.05V.

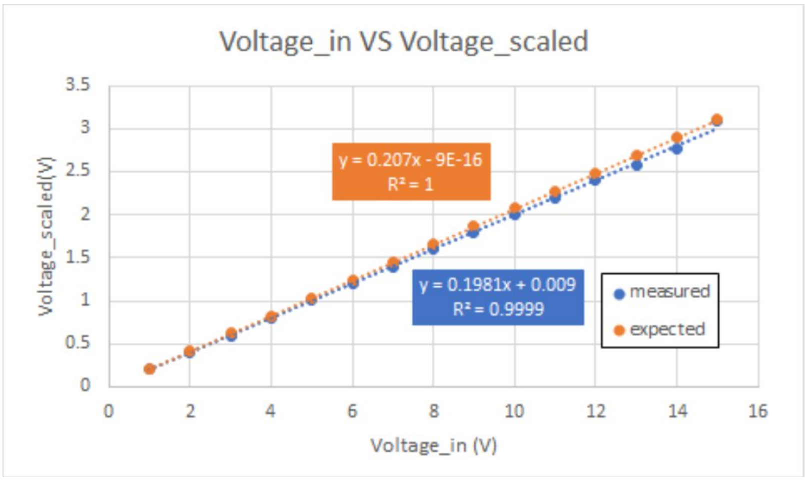


Figure 20: Voltage_in VS Voltage_Scaled

Current Sensor

The current sensor is constructed using a shunt resistor in parallel with an op-amp circuit fed into the arduino. The arduino will then convert the voltage into a corresponding current. The schematic is drawn below.

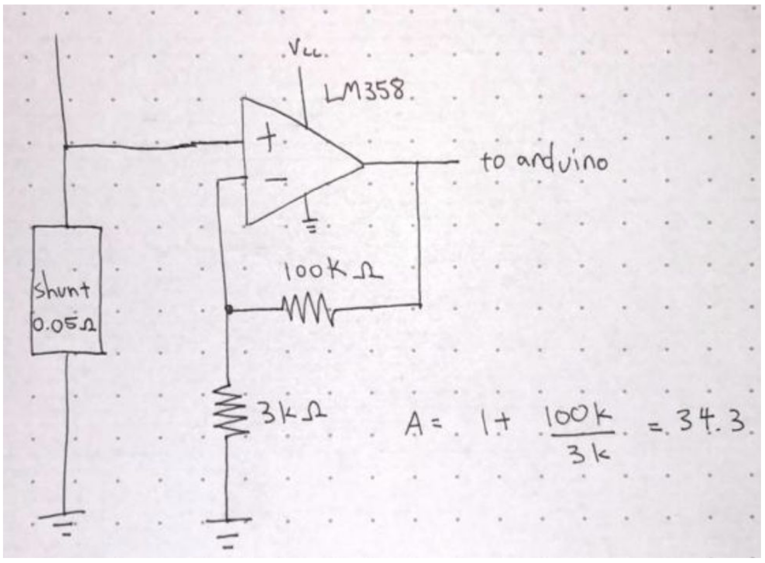


Figure 21: Current Sensor Schematic

As we can see in the schematic, the voltage across the shunt resistor is amplified by 34.3 times. The reason why we need to amplify our voltage drop is due to the resolution of the Arduino. The voltage resolution on the Arduino is 5/1024, which is around 5mV. Assuming the current we receive from our wind turbine to be 0.2A, the maximum amount of voltage drop we will detect across the shunt will be 0.2 * 0.05 = 0.01V = 10mV. In this case, the resolution is half of our maximum value, making our current sensor practically useless. However, if we boost the voltage drop by approximately 30x, 30*10mV = 300mV. Now, the 5mV resolution will only give us a uncertainty of 2%. Which is much better when it comes to MPPT.

We also do not have to worry about power loss as we can calculate the power loss from shunt using the following equation.

$$P = I^2R = 0.2^2 0.05 = 0.002W$$

0.002W can be considered to be negligible, potentially less than the power loss from wires.

After the voltage signal is received from the Arduino, we performed a test using a test current. By varying the test current, we can find the exact ratio of voltage measured to the actual current across the shunt resistor. The graph below illustrates our test current vs voltage, and the slope of graph(1.7592) is the ratio which we used for our conversions.

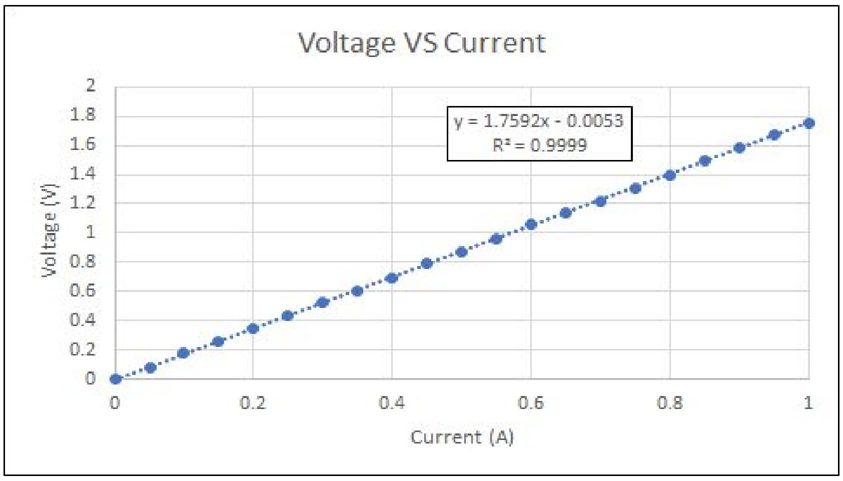
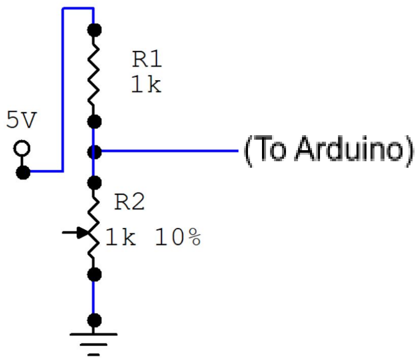


Figure 22: Current VS Voltage(to Arduino)

Wind Direction Sensor

The wind direction sensor uses a voltage divider similar to the voltage sensor, but with Bourns 81A1A-B28-A10L 1K potentiometer as R2. The schematic and final design is as shown below:



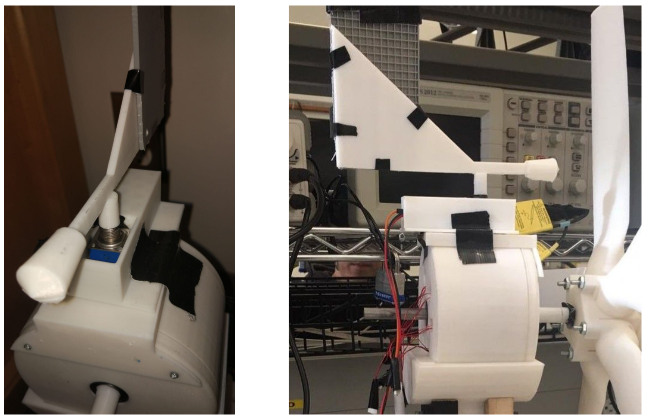


Figure 23: Wind direction sensor schematic *Figure 24: Wind direction final design*

Similar to the voltage sensor, the Arduino is connected between the two resistors. The input voltage for the divider is 5V, which can be taken easily from the Arduino. The voltage measured at the Arduino will follow the voltage divider equation:

$$V_{arduino} = V_{in} \cdot \frac{R_2}{R_1+R_2} = 5 \cdot \frac{R_2}{R_1+R_2}$$

So the maximum and minimum voltage measured in the Arduino is:

$$V_{arduino(max)} = 5 \cdot \frac{1k}{1k+1k} = 2.5\text{V}$$
$$V_{arduino(min)} = 5 \cdot \frac{0k}{1k+0k} = 0\text{V}$$

This range (0-2.5V) ensures that it is well within the range for the Arduino's ADC pins, and maps the direction of the wind based on the resistance on the potentiometer. As the potentiometer has rotation of 300° and resistance of 1kΩ, the wind source can be mapped as 0-150° off to the left or right of the vertical axis, with the midpoint (150°) at 1.25V, maximum offset to the right (300°) at 2.5V, and maximum offset to the left (0°) at 0V. This is better illustrated by the figure below:

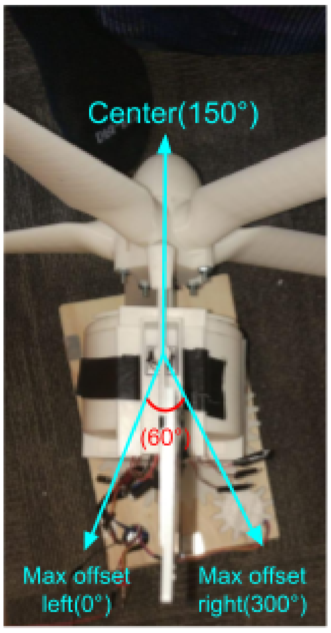


Figure 25: Mapping wind source to voltage

Simulink Models

To aid in creating the control loops for controlling the MPPT and wind tracking, Simulink models were made to simulate the behaviour and fine tune the algorithm before implementation. The final MPPT Simulink model is as follows:

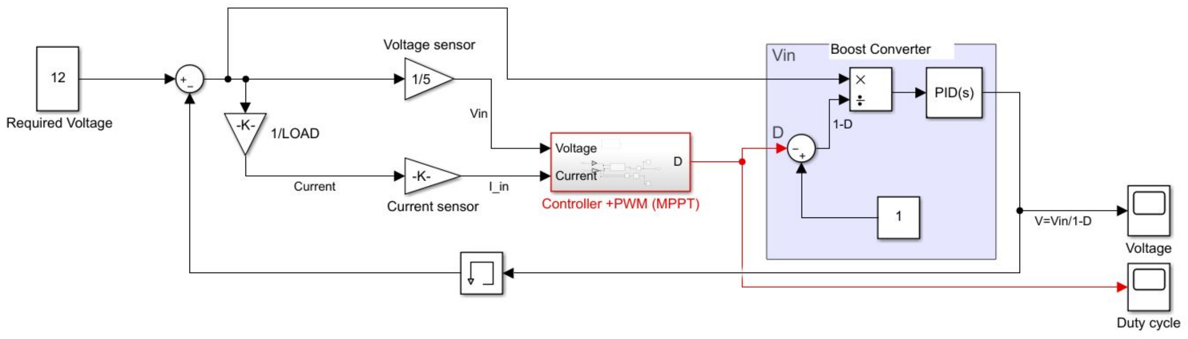


Figure 26: MPPT Simulink Model

The MPPT model uses a MATLAB user defined function to implement the MPPT algorithm, which will be explained in the Controls subtask. The function generates a duty cycle D to the Boost converter, which increases the Vin of the boost converter based on the duty cycle by the equation:

$$V_{out} = \frac{V_{in}}{1-D}$$

The resulting voltage is then subtracted from the voltage input for the system, creating a feedback loop. The voltage is then used as the input for the MPPT algorithm, which completes the loop. The graphs obtained are:

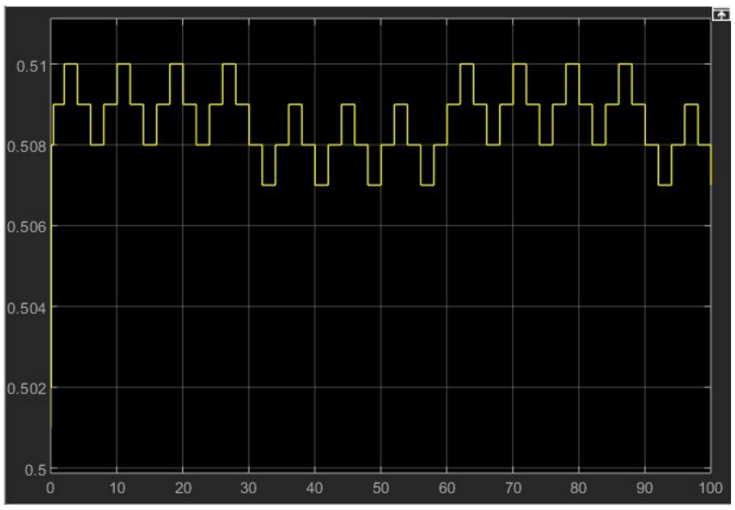


Figure 27: Duty Cycle

As seen from the graph, the duty cycle fluctuates slightly due to the MPPT algorithm implemented. The “perturb and observe” method meant that the duty cycle is altered constantly to see if changes need to be performed. Otherwise, it remains relatively stable within a certain range.

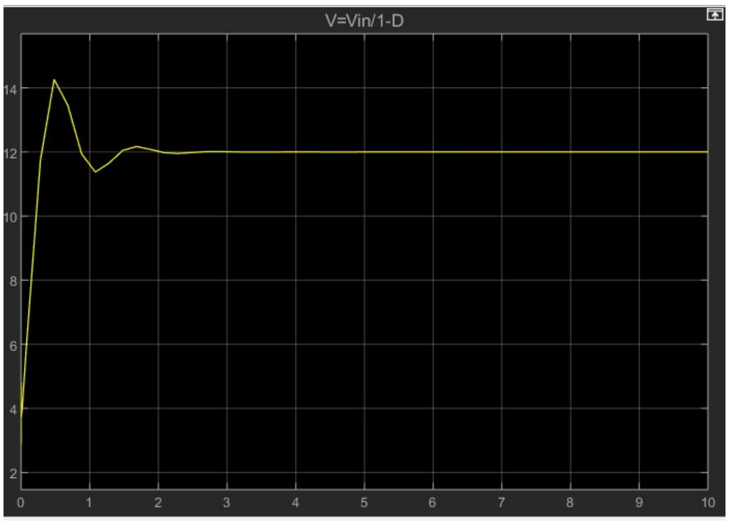


Figure 28: Voltage output from boost converter

From the graph above, it can be seen that the voltage resulted from the boost converter fluctuates before settling. This is due to the fluctuating duty cycle until it reaches a point close to stable, at which the duty cycle continues to fluctuate, as seen in figure X. However, since the fluctuation is very small, it will not be visible in this graph except for the larger ones at the start, before settling.

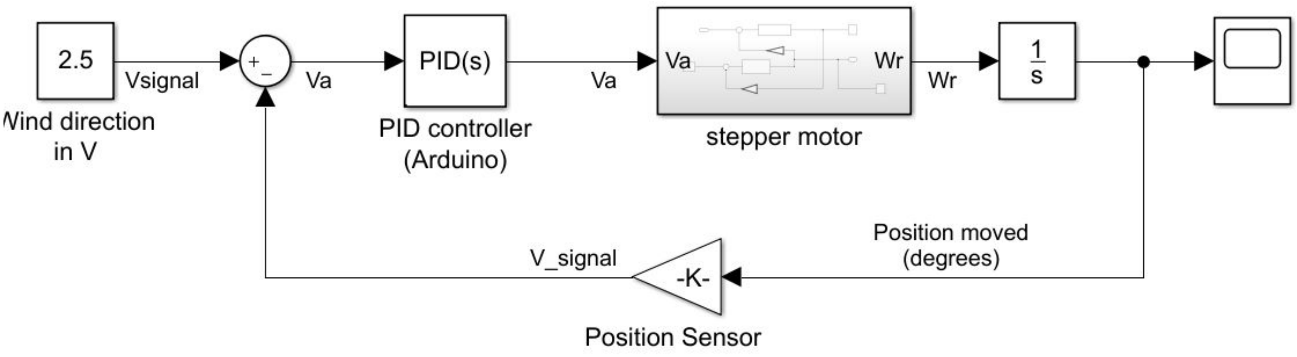


Figure 29: Wind tracking Simulink model

The wind tracking model takes a scaled voltage input to simulate a reading from the wind direction sensor. After scaling the voltage to a position through the Arduino, a voltage is sent to the stepper to turn the base. The resulting change in turbine position is then measured and a feedback loop connects it back to the original Vsignal through the position sensor. The graph obtained for the position moved(degrees) is as follows:

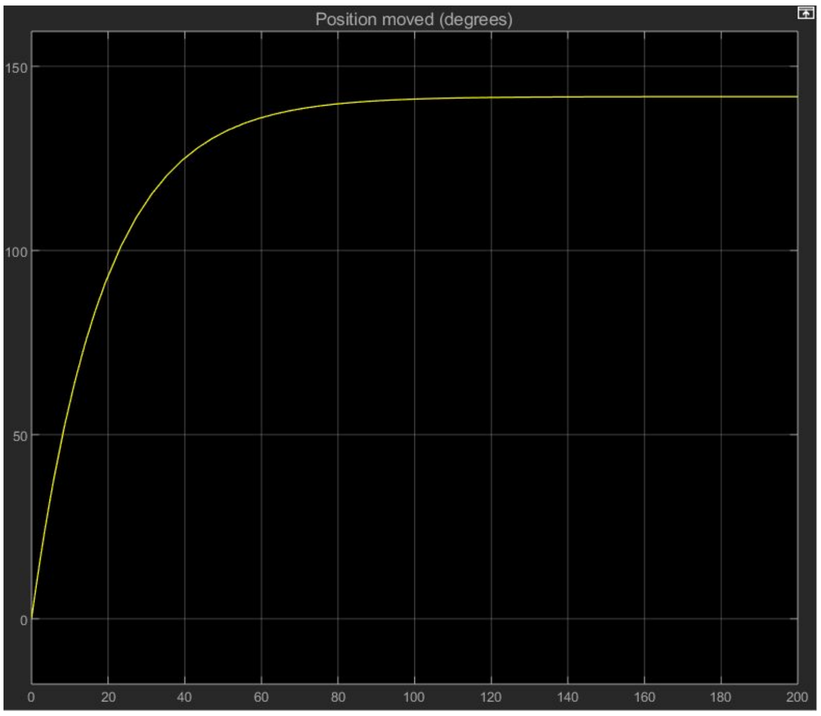


Figure 30: Position moved(degrees)

The resulting graph shows the gradual change for the position moved in order to reach the center. Since the maximum offset is +/- 150° for a wind sensor voltage of 0-2.5V, the position moved at 2.5V input is roughly 145°, as the center was marked at 1.3V to better simulate real life values. So the turbine position eventually moves 145° to get to center again.

Prototypes

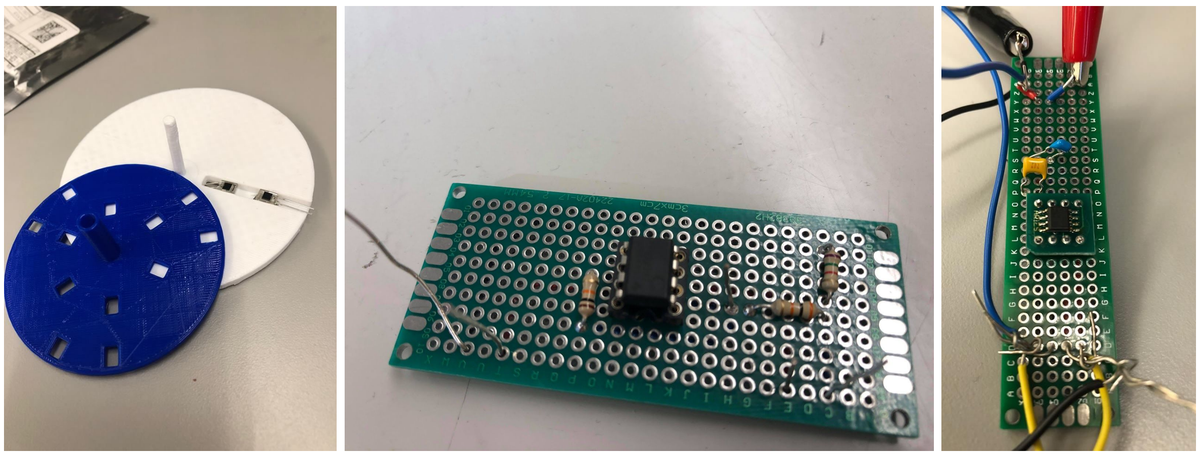


Figure 31: Prototypes for wind source sensor, voltage sensor, current sensor, respectively.

Originally, there there were plans to include a voltage follower between R1 and Vin to ensure high impedance to the voltage divider, minimizing the power loss in the sensor. However, to ensure saturation does not occur, a separate 15V source is needed to power the Op-Amp. This proved to be an issue towards the end of the project when it was decided that the turbine does not need to be powered by an external power source, and only a power bank. Therefore, the final design has increased resistor values to minimize power loss instead.

Hall effect sensors were originally used as the current sensor for the project. However, after one of them malfunctioned on our PCB, we opted to use the shunt current sensor from our new team member as it was more robust in design. This was first tested on a perfboard before finalizing the final version which included the voltage sensor.

Prototype for the wind direction sensor was an optical encoder, but after deciding that the sensor be placed on top of the generator, a potentiometer was quickly chosen in its place due to its compact design and simplicity. Furthermore, multiple iterations for the wind vane design were considered, with the final one being heavy enough to be both accurate and have reduced fluctuations on the potentiometer readings.

Budget

Most parts were bought for original designs but as project progressed were switched to use parts from previous projects, such as the potentiometer instead of the light sensors and the shunt current sensor instead of the hall effect. Resistors for the voltage sensor were obtained from the lab for free. Final budget used was within the allotted $40 for sensing.

In total, the following parts were ordered for sensors and modelling:

Quantity	**Name**	**Price(plus GST)**
2	Photodiode	2.26
2	Hall effect sensor	18.22
2	SOIC8 to DIP8 adaptor	9.48
7	Ambient Light Sensor	7.44
	Final Total:	37.4

Table 2: Budget usage for sensors

4.4 Controls

Overview

The goal of the controls part of the project is the implementation of the closed-loop wind-tracking system, as well as current control, blade-speed control and maximum power point tracking (MPPT) algorithms onto a microcontroller.

Requirements

For the wind-tracking system, the requirement is that the wind turbine maintain the direction of the rotational plane of the blade perpendicular to the airflow with a minimum 90° angle change of wind source direction. Using the potentiometer, we are able to achieve MPPT under a 300° angle change of wind source direction.

The requirements for current control and blade-speed control are related, as current control is used to maintain the rotational speed of the turbine blades at an optimum value relative to the changing wind speeds. This is done by controlling the PWM output to the boost converter, which must ensure the boost converter output is between 10-15V.

As far as MPPT, the requirements are to make sure the wind turbine maintains maximum output power for changing wind direction and speed.

In addition, it is required that PWM be done by hardware and not software, and to also ensure constant control time-step.

Goals

The main goal of this section is to create a closed feedback loop that takes inputs from the wind direction sensor and voltage/current sensors from the output of the boost converter and uses this information to maximize the output power of the wind turbine.

To do this, the large overarching goal is broken down into smaller manageable ones. First is the ensure interrupt driven programming was used to ensure non-blocking code. Another goal is to place an emphasis on modular code design, which allows for easier debugging and readability.

Microcontroller

The microcontroller used is the Arduino Mega 2560. Besides the fact that a team member owns one and thus we could save costs, the Arduino was chosen due to it having six timers, with five being available for use. This allows us to configure up to five different interrupts and five different PWM frequencies. Additionally, the board has 54 I/O pins, with 14 capable of PWM output. This is important as we were initially considering having a bonus involving changing the colour of RGB LEDs which would require at least 6 PWM pins. Another important reason for choosing the Arduino was that it has multiple shields which can be purchased to add additional functionality to the board, such as a Bluetooth or ethernet shield.

An Arduino Uno is also used but it is not programmed and instead used to power the stepper motor.

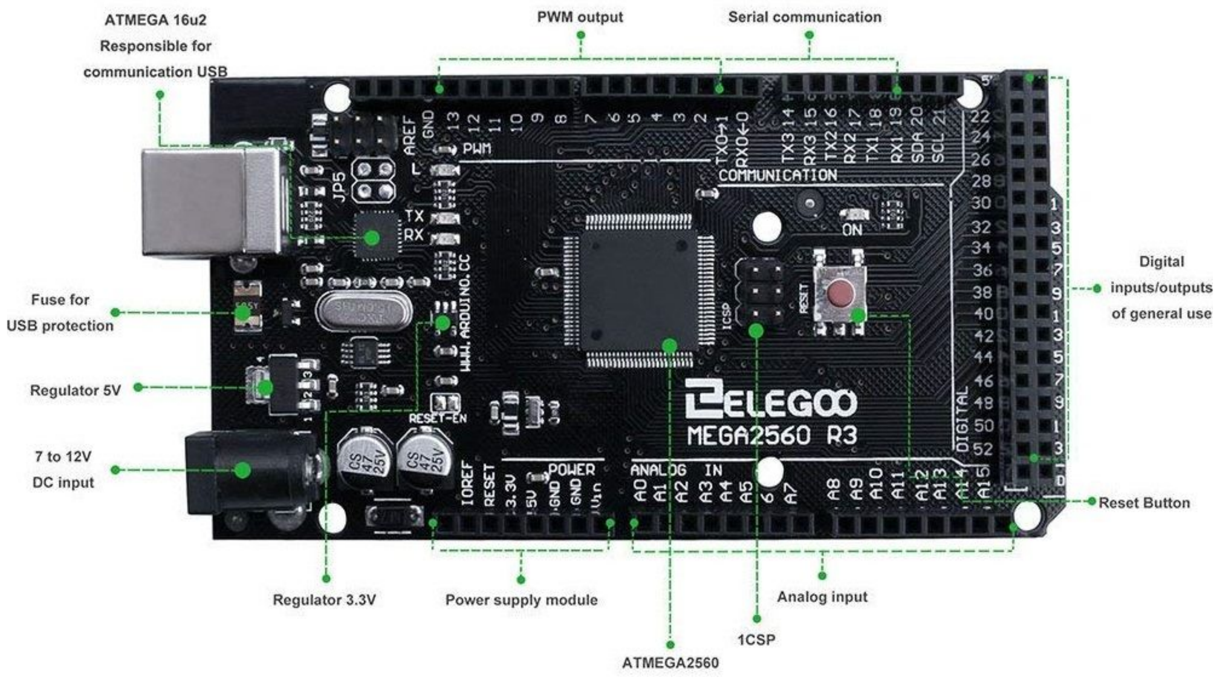


Figure 32: Overhead view of the ELEGOO MEGA 2560 R3 board used

To control the 5 timers, the open source TimerThree library is used and modified to also work for Timer1, Timer4, and Timer5. For Timer2, a different open source library called FlexTimer2 is used. With these libraries, a period can be assigned to each timer at which the interrupt will occur, and this will correspond to the period of the PWM output belonging to that timer. A duty cycle can also be assigned for that PWM and changed at a later time. Timer3 is used for the PWM output of 150kHz. Additionally, each timer can also have a function attached which will serve as the interrupt service routine, which is how the other four timers are used.

Timer	Function
Timer1	AnalogSensor
Timer2	MPPT
Timer4	BoostControl
Timer5	WindSensor

Table 3: ISRs for each timer

Analog Sensors

There are three different analog inputs to the Arduino; voltage from the potentiometer, voltage from the voltage sensor connected to boost converter output, and voltage from the current sensor which is a shunt with opamp. All three of these inputs have their readings taken in the AnalogSensor function.

The AnalogSensor function is called at a frequency of 200Hz, or every 5ms. The function will read from the analog input pin corresponding to each signal and add them to a stored sum. After 100 readings or every 0.5s, the average of the sums for each input will be taken and used. This allows us to get more accurate readings and not have to worry about random spikes in input voltage.

Since the Arduino can only take a max voltage input of 5V, the boost converter output is reduced using a voltage divider before being sent to the Arduino. Thus, after the average of the input is taken, the actual voltage will be recalculated by simply dividing it by the voltage divider equation of R2/(R2+R1). The voltage from the current sensor is converted to a current by simply dividing it by 1.75, which is determined by mapping the linear relationship of the output voltage to current. As for the voltage from the potentiometer, since its max value is ~2.5V, there is no need to do any additional calculations after finding its average value.

The output power of the wind turbine is simply calculated by multiplying the voltage and current readings from the boost converter output and stored in a global variable for use by the MPPT algorithm.

To ensure we are getting the correct readings, we first test the current and voltage sensor by using the DC power supply, so we can know what voltage and current we should detect. Then, we used a multimeter and compared the readings it has to the Arduino to figure out the degree of precision we have. With the voltage sensor we are within a 0.05V margin of error and the current sensor has a 0.01A error margin.

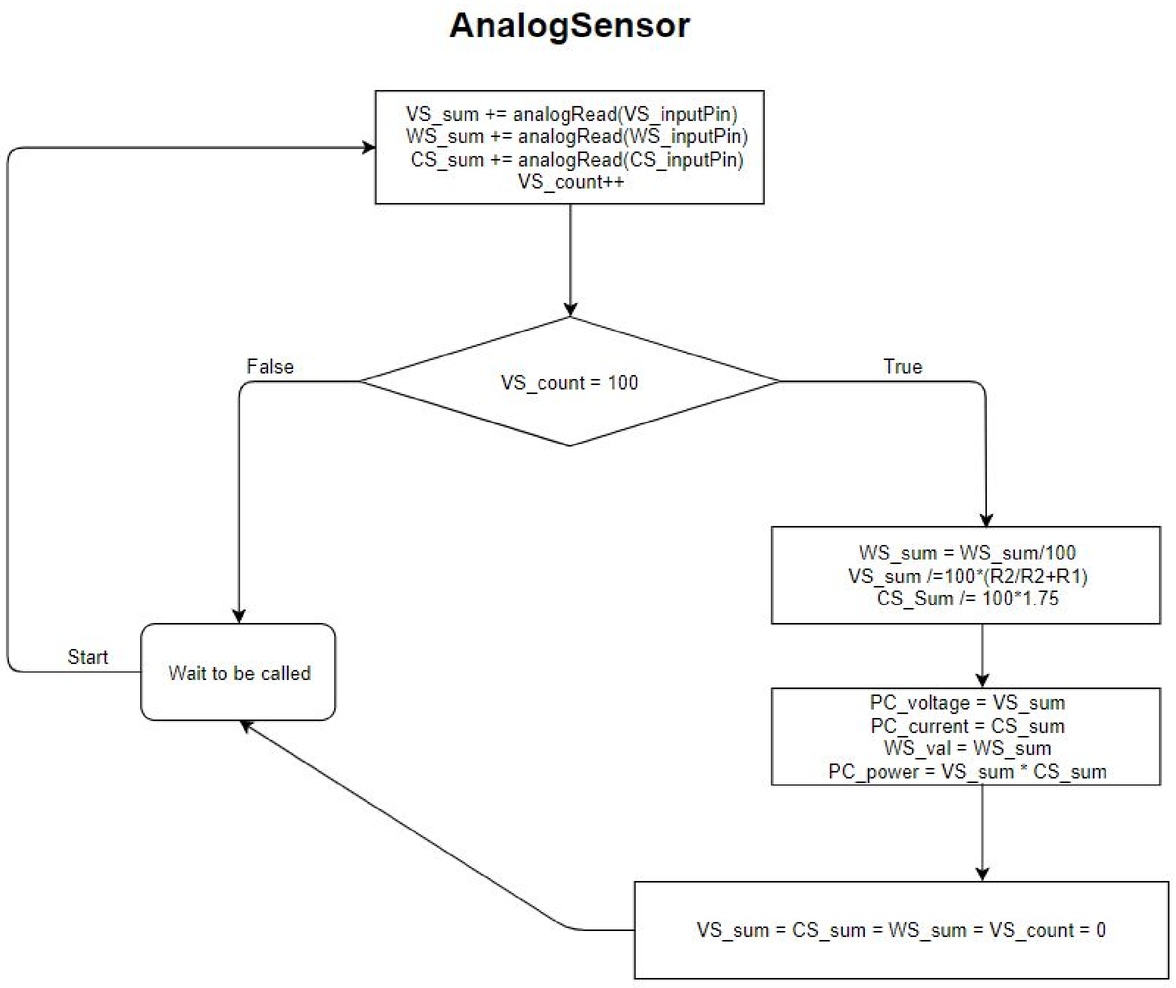


Figure 33: Diagram of AnalogSensor

WindSensor and Stepper Motor Control

The wind tracking component of controls has two main parts, detecting the wind direction and controlling the stepper motor.

The function WindSensor is attached to Timer5 and called at a frequency of 10Hz, or every 0.1s. The method chosen to detect wind was to select the voltage at which the potentiometer was directly perpendicular to the wind source, and use it as a reference. This reference voltage was approximately 1.35V. Then, the difference between the current voltage of the potentiometer and the reference is calculated. An error margin of ±0.1V is added as it will be impractical to try and get the potentiometer to achieve a voltage of exactly 1.35V. If the calculated difference is greater than 0.1, then the variable dir will be set to -1, which means the stepper motor will turn counter clockwise. If the difference is less than -0.1, then dir will be set to 1, corresponding to a clockwise error correction. Otherwise, the stepper motor will not move.

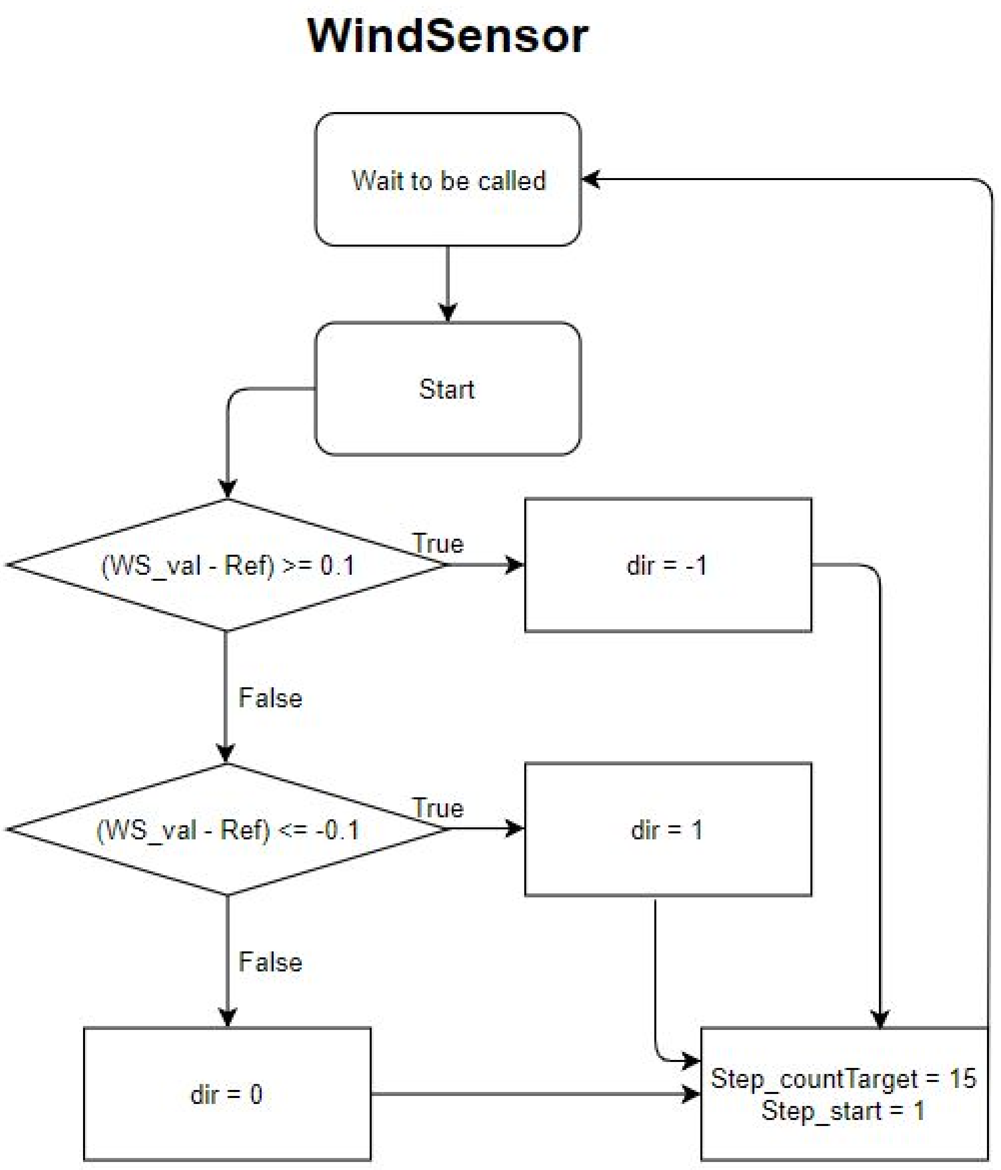


Figure 34: Diagram of WindSensor

Next is the stepper motor control. To control the stepper motor, the Stepper library, which is part of the basic Arduino IDE, is used. The stepper motor used is the included 28BYJ-48 Stepper Motor with ULN2003 driver. An Arduino Uno is used to supply 5V supply to the motor, and the four wires from the ULN2003 are connected to the Arduino pins 47, 49, 51, and 53. From reading the datasheet of the motor, we know the stepper motor needs 2048 steps to complete 1 revolution, so 1 step = 0.176°.

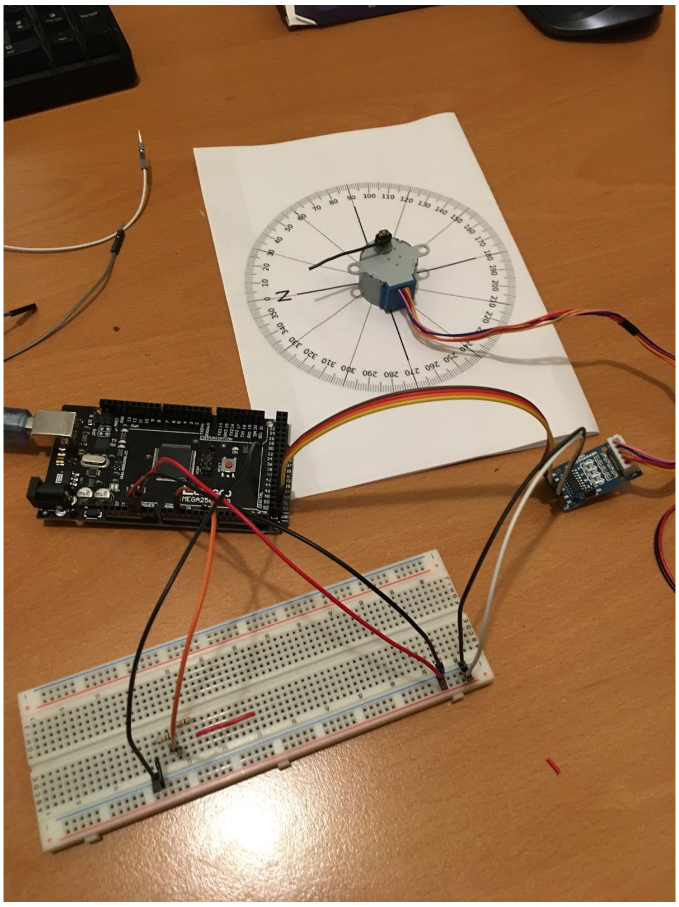


Figure 35: Test setup for stepper motor control

The StepperControl function is in the main loop function, as it has the lowest priority. This function simply makes the stepper motor turn 15 steps (2.64°) by calling the '.step' function of the object instantiated from the Stepper.h library. The direction is dependent on the variable 'dir' which is controlled by WindSensor.

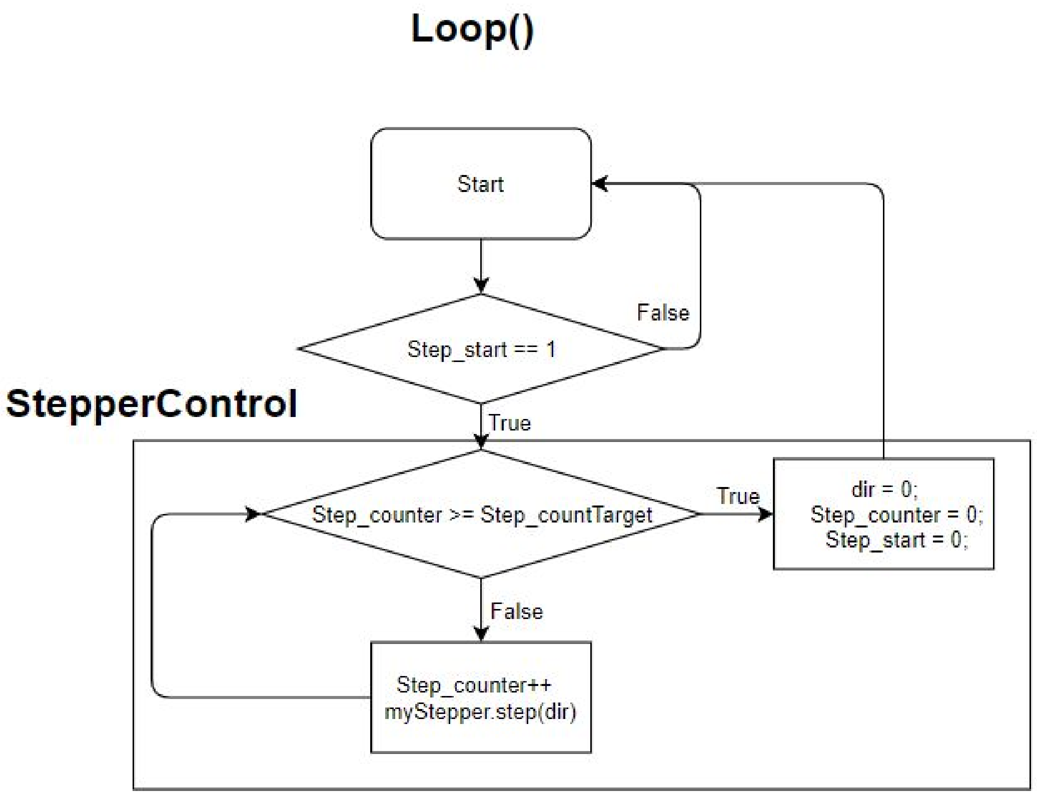


Figure 36: Diagram of Stepper Control

Maximum Power Point Tracking (MPPT) Algorithm

For MPPT, the perturb and observe method is chosen. The basis of this method is that the controller makes a small adjustment, in our case for the duty cycle, and measures power. If power increases, adjustments are made in that direction, either increasing or decreasing duty cycle, until power no longer increases. We chose this method because not only is it simple, but it also depends on “hill climbing”, which is the rise in power against duty cycle before the maximum power point and then the decline after that point. When we were manually adjusting the duty cycle to test that the MPPT algorithm was working, we noticed this trend in the graph and it further renewed our belief that this method was best suited for our purposes.

In our case, the MPPT function checks if the current power stored in a global variable is greater than or equal to the past voltage stored in the variable ‘old_Power’. If true, then we simply update the ‘old_Power’ variable with the current power and exit, which lets the boost control keep decreasing or increasing the duty cycle. If false, then we multiply the ‘boostup’ variable by -1, which either makes it 1 or -1. This variable is used by boost control to choose whether to increase or decrease duty cycle. If 1, then it increases duty cycle, else it decreases it. Then ‘old_Power’ gets current power and exits.

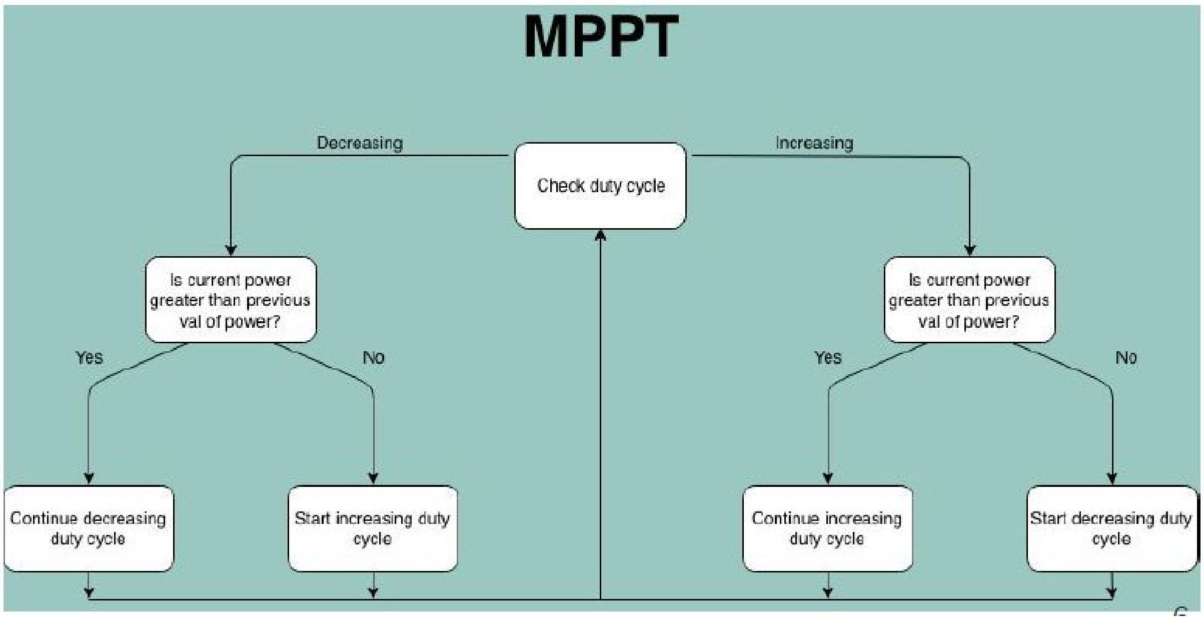

MPPT
Decreasing
Check duty cycle
Increasing
Is current power greater than previous val of power?
Yes
No
Is current power greater than previous val of power?
Yes
No
Continue decreasing duty cycle
Start increasing duty cycle
Continue increasing duty cycle
Start decreasing duty cycle

Figure 37: Overview of MPPT method

As mentioned before, to ensure our MPPT algorithm is working as expected, we manually adjusted the PWM input to the boost converter. We plotted the power vs duty cycle graph to see at what point the maximum output power was being achieved, which seemed to be between a duty cycle of 0.3 and 0.35. Then, we let the MPPT automatically adjust the PWM duty cycle to see which duty cycle it determined gave the maximum power output, and it hovered between the before mentioned range of 0.3 to 0.35 for the duty cycle.

We adjusted the frequency of the MPPT from the original 200Hz to 20Hz as we found we got better results from this. Although it did take longer to reach and maintain the maximum power output level, from 15 seconds to 30 seconds, we found it was worth the trade off as the duty cycle wouldn’t oscillate too much once it reached a stable level.

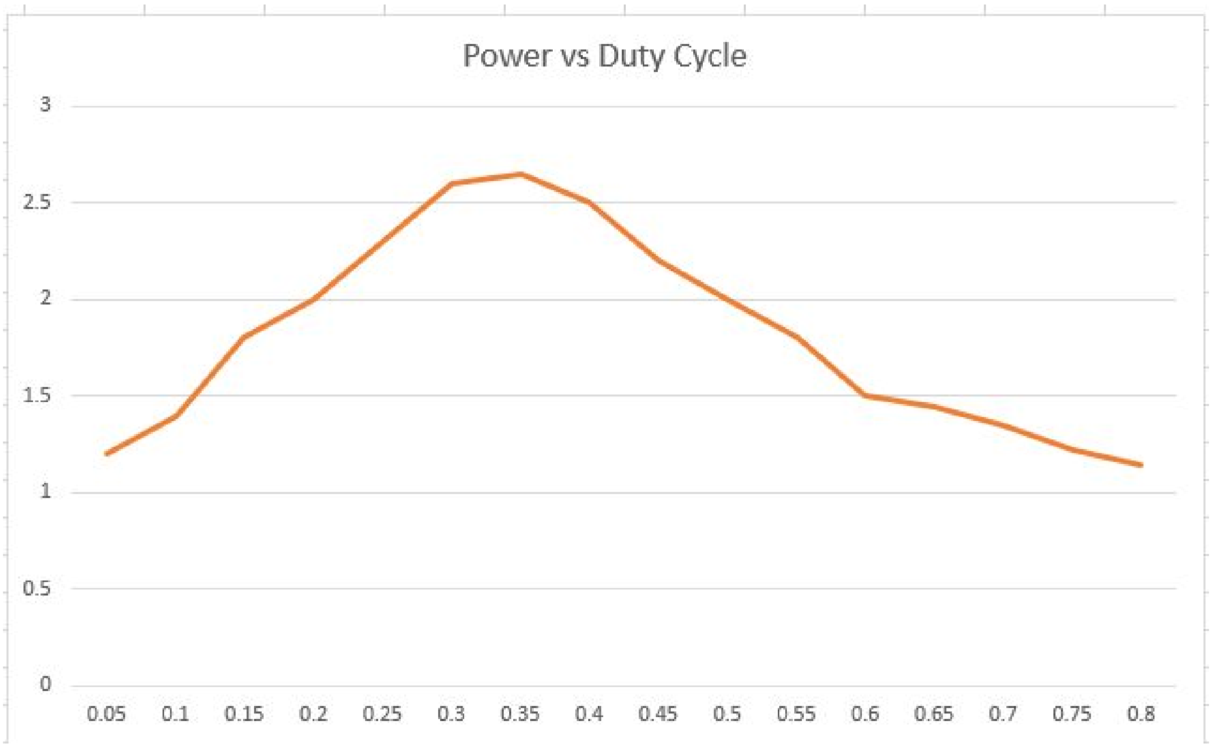


Figure 38: Plot of Power vs Duty Cycle when manually adjusting duty cycle

Budget

The allotted budget for controls was~40.00 for components. However, nothing was spent due to a team member already possessing the Arduino Mega and myself having an Arduino Uno. Additionally, the Bluetooth module used for the bonus was also reused from a past project. Thus, the total costs used by controls were 0.00.

Final Setup

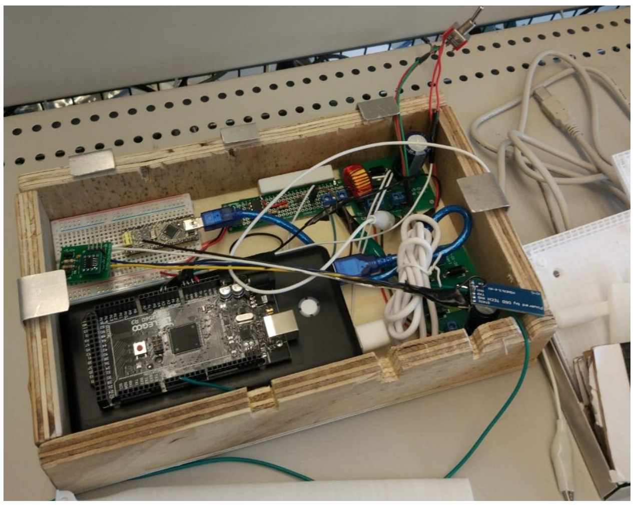


Figure 39: Final configuration of Arduino inside case

5. Integration

Overview

The process of integration involves three major subtask integrations. Power electronics and the generator will be the first integration alongside sensors and controls. Second, controls and sensors will work with power electronics and the generator with the main goal of a functioning MPPT algorithm.

Power electronics and the generator are separately integrated because power electronics is the only subtask that interfaces with the generator. Controls and sensors work together since those two subtasks work together the most. This way, we can begin the integration process without any major dependencies of another subtask.

Power Electronics and Generator

The first decision is the inclusion of a gearbox to either increase or decrease the RPM of the generator. We start by mounting the blade hub directly to the generator and measuring the open circuit voltage and short circuit current following rectification.

With the blades directly mounted to the generator, the measured open-circuit voltage is approximately 10V rectification which is the ideal input voltage into the boost converter. From this, we decided that the blades mounted directly to the generator is the best choice as the generator produces an ideal 10V rectified and the addition of a gearbox will be an issue.

The first possibility is a gearbox ratio that would increase the generator speed which may result in the generator not turning and if the generator did turn, the open-circuit voltage would be too high. The second possibility is a gearbox ratio that would decrease the generator speed which will result in a decreased open-circuit voltage which will result in the inability of the boost converter to boost the voltage to the required 10-15V. Additionally, a gearbox would require more time to design and more points of mechanical failure. Thus, the drawbacks of a gearbox outweighed its possible benefits.

Controls and Sensors

The first part for integration between controls and sensors is to get the Arduino microcontroller to acquire readings from the sensors. To do this, we start by obtaining the exact equations to scale the voltages from the sensors to the measured voltage or current. This is done experimentally as shown in figures 20 and 22. These equations are then used to calculate the actual voltage or current from the scaled voltage within the Arduino control loops. For wind direction readings, a similar process is used. The range of the wind sensor output(0-2.5V) is mapped to the rotation(0-300°), and is experimentally confirmed that the positions are accurate and matching to the readings.

With the controls able to read from the sensors correctly, the next step is to begin adding outputs from the Arduino based on different inputs from the sensor. A program was written to make the stepper motor mimic the movements of the potentiometer in the wind direction readings. This was done by changing the direction of the stepper based on the potentiometer readings.

Once the wind direction control loop was complete, the last step was to finish the MPPT control loop. The first step is to calculate power from the voltage and current sensors, and change the duty cycle of the PWM outputted by the Arduino to reach MPPT. The goal is to keep the output voltage of the boost converter within the range of 10-15V, and then focus on maximizing power output. As mentioned before in the MPPT section of the controls task, a perturb and observe method was chosen.

For this testing, we began to integrate controls, sensors, and power electronics. Sensors would take voltage and current readings from the boost converter and send them to the Arduino. The Arduino calculates voltage and current, then multiplies them to find power. With the voltage, current, and power, the Arduino adjusts the PWM output that it sends to the power electronics in order to keep the voltage within the aforementioned range.

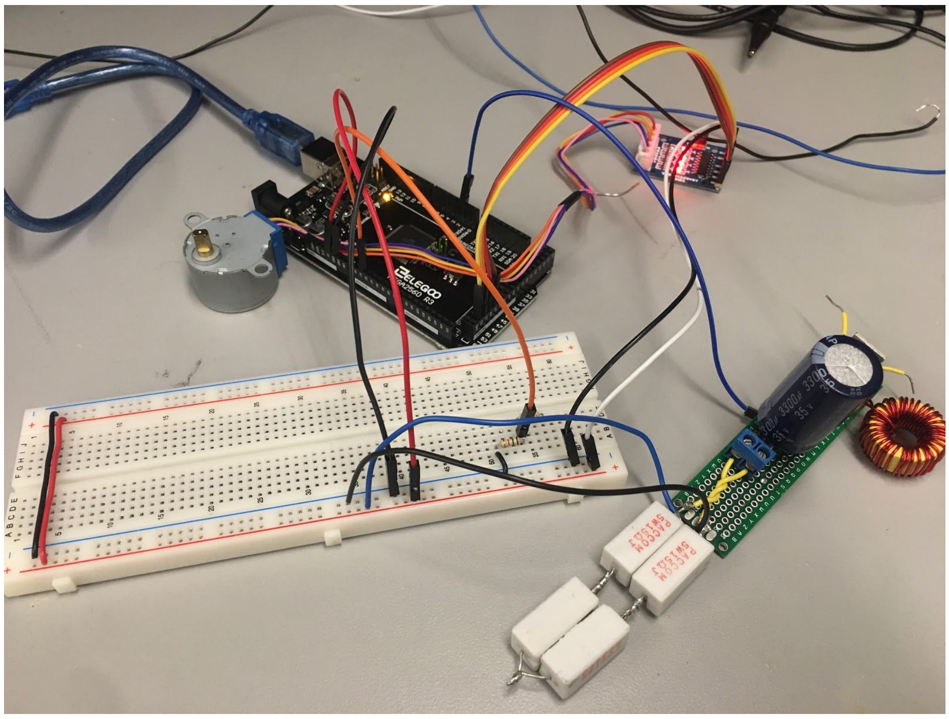


Figure 40: Test setup for controls and power electronics

Combining everything together

For the final part of integration, the placement of all components is finalized. In order for the wind direction sensor to work without obstructing the fan blades, the sensor is placed above the generator. Doing so also allowed the increase of the size of the wind vane, since it no longer risked hitting the fan blades compared to placing it in front.

Furthermore, a suitable external power source must be included to power the microcontroller and other electronic components. We chose the Poweradd Pilot X7 Power Bank 20000mAh portable charger due to its huge capacity and dual USB outputs. The huge capacity meant that the wind turbine can stay on for a longer period of time without external aid; and the dual output meant that the Arduino and sensors stepper motor are powered separately, and have a separate ground. Thus, reducing the noise generated by the stepper which may affect sensor readings.

Lastly, we had to determine the placement of all components and build the final structure. As all our components had modularity in mind, the placement was simple as each part was placed according to the flow of components, from the generator to the load. Wood was chosen as our material of choice to build our base and stand for the wind turbine, as it was cheap and easy to work with compared to metals. Furthermore, we used a box shape with a sliding lid to ensure all components may fit, and tight metal braces to hold down the top to reduce vibrations of the lid.

Testing and Validation

Following the physical construction of the full system, we performed various tests to ensure the requirements of the project were met. The main motivations of the test are to validate the performance of the system with regards to power output, wind direction tracking, and the MPPT algorithm.

Power Performance

To validate the power performance of the system, we start by choosing the correct resistor value to extract the maximum power from the generator. The process involved keeping only connecting the generator to the power electronics and testing various resistive loads to the boost converter output. For each resistive load, we did manually adjusted the PWM duty cycle to obtain the maximum power that could be supplied to the resistor. The following table shows the duty cycle and corresponding maximum power for each resistor value at hand.

Resistor Value (Ω)	PWM Duty Cycle (%)	Output Power (W)
75	32	1.89
82	36	2.74
91	40	2.31
100	38	2.23
120	42	2.07

Table 4: Multimeter Measured Maximum Power with Duty Cycle (Manual)

From these validation tests, we decided that a 82Ω resistive load would allow for highest power extraction from the generator.

All subtasks were then connected for a full test. To ensure that MPPT and voltage and current sensors are working, we attached all the above mentioned resistive loads and allows the MPPT algorithm to control the PWM duty cycle. There is a slight error in the power measurements which directly affected the MPPT algorithm, but the steady-state duty cycle and power measurement does satisfy the power performance requirement. The resulting steady-state PWM duty cycle and measured output is shown in the table below.

Resistor Value (Ω)	PWM Duty Cycle (%)	Output Power (W)
75	35	1.95
82	34	2.65
91	42	2.47
100	39	2.15
120	40	1.97

Table 5: Sensor Measured Maximum Power with Duty Cycle (MPPT)

Wind Tracking Performance

To determine whether we met the requirement of detecting a change of 90° of wind source direction, we first began by placing the wind turbine directly in front of the fan and turning it on the highest speed. We noticed that the wind vane was not directly perpendicular to the source of wind, and thus tried lowering speeds to verify if this was correct. We deduced that although the fan faced the turbine head on, the wind direction was not directly perpendicular to the turbine and so the slight tilt of the wind vane was to be expected. After using two different fans and changing the speeds, we chose the reference voltage of 1.35V for the potentiometer the wind vane was attached to.

Next, we start the turbine directly facing the turbine and slowly rotate it clockwise to check if the stepper motor would turn the base counter clockwise to compensate, which it did. We did the same for the counterclockwise rotation and then changed wind fan speeds to check it met the requirements for changing wind speed.

Once that was verified, next we started the fan from a ± 45° offset from facing the fan, once again checking with different wind speeds to ensure it worked. After we knew that we met the minimum requirement of 90°, we tested for our bonus of 300° using the same methodology as before, except starting at ± 150° offsets.

To ensure the wind sensor was allowing us to reach maximum power output, we would start at an offset and let the stepper motor turn the turbine until it stabilized. Then, we would disable the wind sensor control on the Arduino through a keyboard command in the serial port. The Arduino still continued to display the current power in the serial monitor and so we would manually rotate the turbine in small increments to determine whether we were actually getting maximum output power.

From our findings, we saw that any adjustments we made would result in a lower power output, with negligible increases in power resulting when we made extremely small adjustments. This could be attributed to the ± 0.1V error margin we allow our potentiometer to have in regard to the reference voltage, but we find the tradeoff worth it as the stepper motor constantly moving makes the turbine constantly shake, which lowered our total power output.

Bonus Features

For our bonus, we implemented a wireless real time monitor that is hosted on a web server. First, we accessed the serial port on the Arduino using a bluetooth module that communicated with a local PC. Using Javascript, we open the serial port and read data that the Arduino is sending. We receive the voltage, current, and power at the output of the boost converter, voltage of the potentiometer, and the duty cycle of the PWM. This data is then sent to our web server. Next, we create an HTML/CSS page and use a ChartJS, a Javascript library, to create real time strip charts of the data being received and create a nice monitoring interface. Now the server, built using Node.js, uses a free node package called localtunnel to forward our localhost to a temporary domain, allowing anyone with the URL and an internet connection to view the information real time..

In addition to the website, we also had a bonus of having the wind turbine be able to rotate 300° around the vertical axis in response to a change in wind direction due to our choice of using a potentiometer for wind direction sensing.

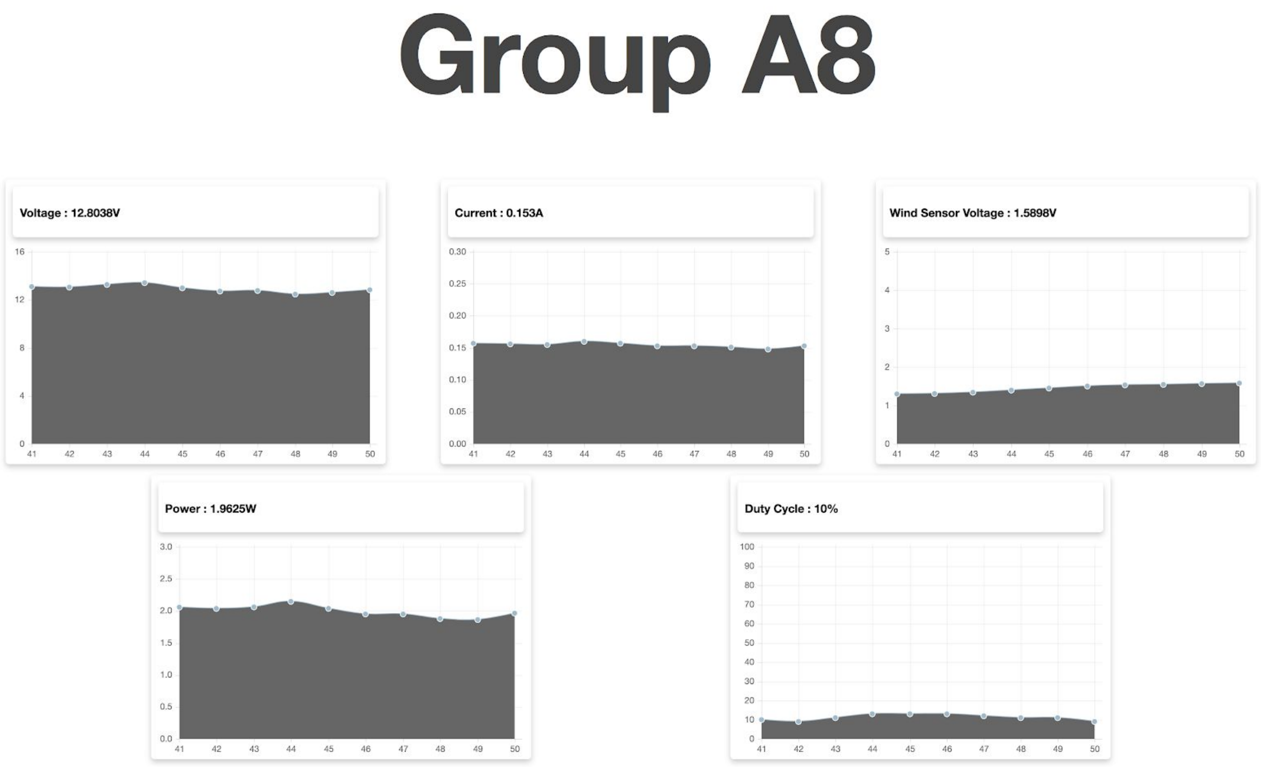
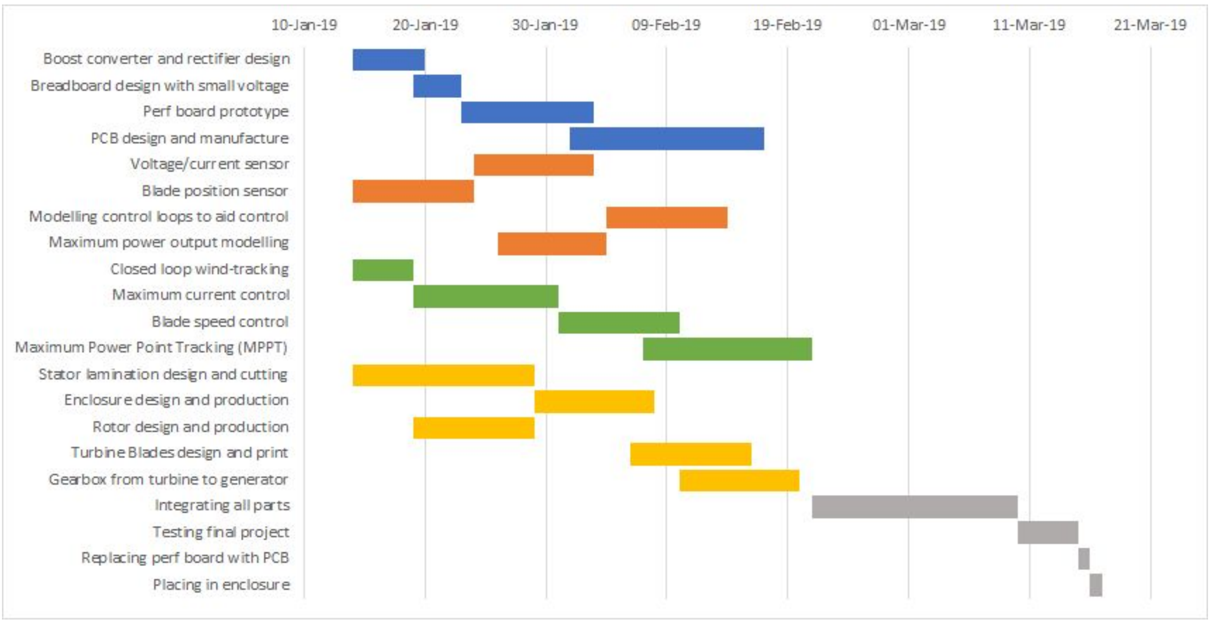


Figure 41: Website showing real time data

Appendix

Appendix A: Gantt Chart



Appendix B: Original Budget Allocation

<u>**PART**</u>	<u>**PRICE ($)**</u>
Controller Components (Arduino, Arduino Shields, ...etc.)	~40.00
Power Components (Inductors, diodes, mosfets, ..etc)	~50.00
Generator Components (permanent magnet, windings, etc.)	~55.00
Sensor Components (potentiometer, OpAmps, hall effect sensors, etc)	~40.00
Waterjet Cutting	~50.00
Power PCB	~160.00
Material (tower, base, etc.)	~35.00
Emergency funds (Replacement parts, new parts, etc.)	~70.00
Total	500.00

Appendix C: Final budget used

Items Bought

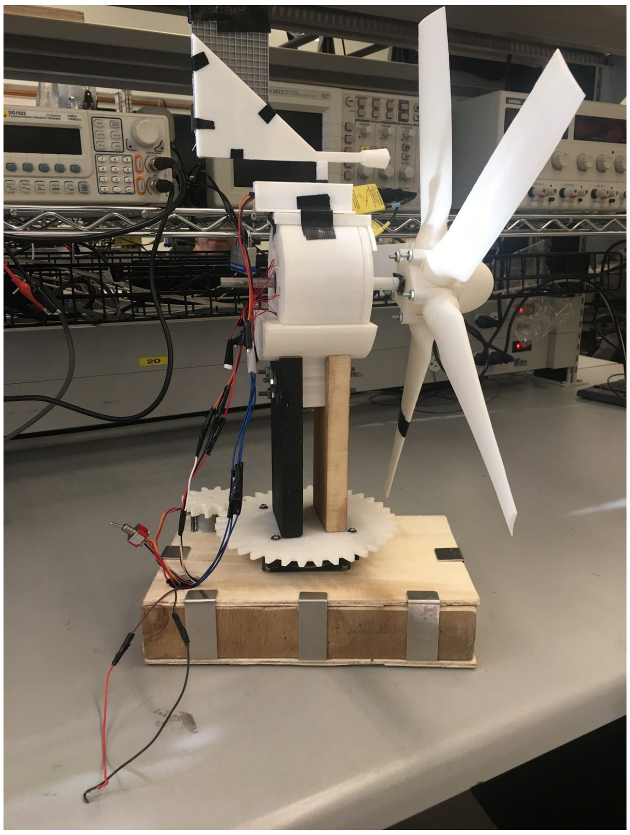
Quantity	**Name**	**Total Price(plus GST)**
2	Photodiode	2.26
1	Hall effect sensor	9.11
12	Round magnets	2.39
6	Rectangular magnets	25.14
4	Ball Bearings	6.13
1	SOIC8 to DIP8 adaptor	4.74
3	45V 10A Schottky diode	4.91
3	Heat sink	2.02
12	20V 5A Schottky diode	9.07
2	5K 10W Resistor	9.67
3	3 hole screw terminal	2.77
1	100uH Inductor	2.87
2	100V 33A MOSFET	3.70
5	Ambient Light Sensor	5.23
5	Ball Bearing Sealed	6.15
1	Jumper Wire (Male - Female)	2.90
1	Jumper Wire (Male - Male)	2.90
2	50V 1000uF Capacitor	3.86
5	3 Pin Male Header	0.95
1	M3 Screw	0.89
2	Ambient Light Sensor	2.21
12	Rectangular Magnet	37.50
1	Hall Effect Sensor	8.37
1	15 Ohm 35W resistor	4.59

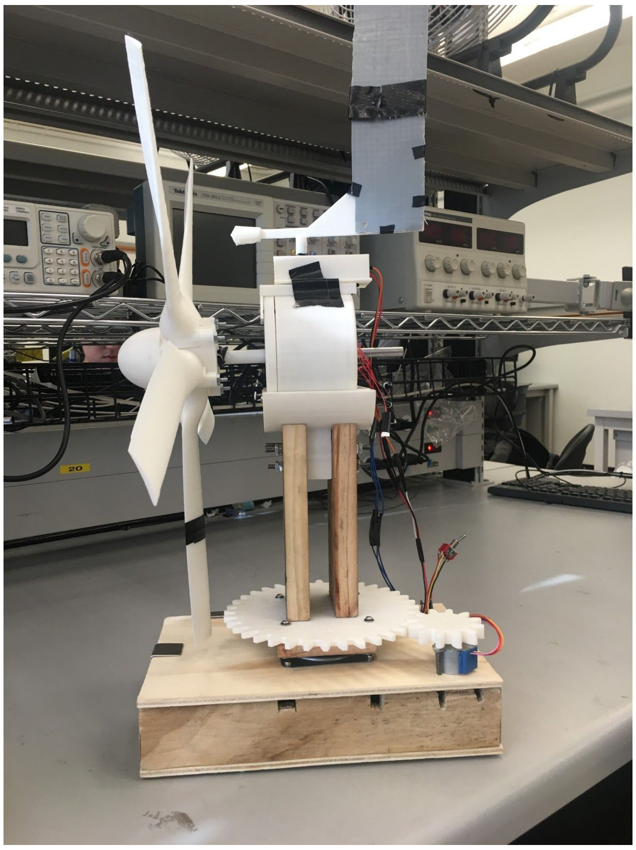
4	2 position header connector	1.47
1	SOIC8 to DIP8 adaptor	4.57
10	2 position header connector	1.47
1	15 Ohm 10W resistor	1.04
1	20 Ohm 10W resistor	1.21
1	25 Ohm 10W resistor	3.93
	Total	176.30

Total expenses incurred

Waterjet	PCB	Components	Total
112	**109.93**	**176.30**	**398.22**

Appendix D: Final turbine with all components assembled (Side views)





Appendix E: Final turbine with all components assembled (Front and back views)