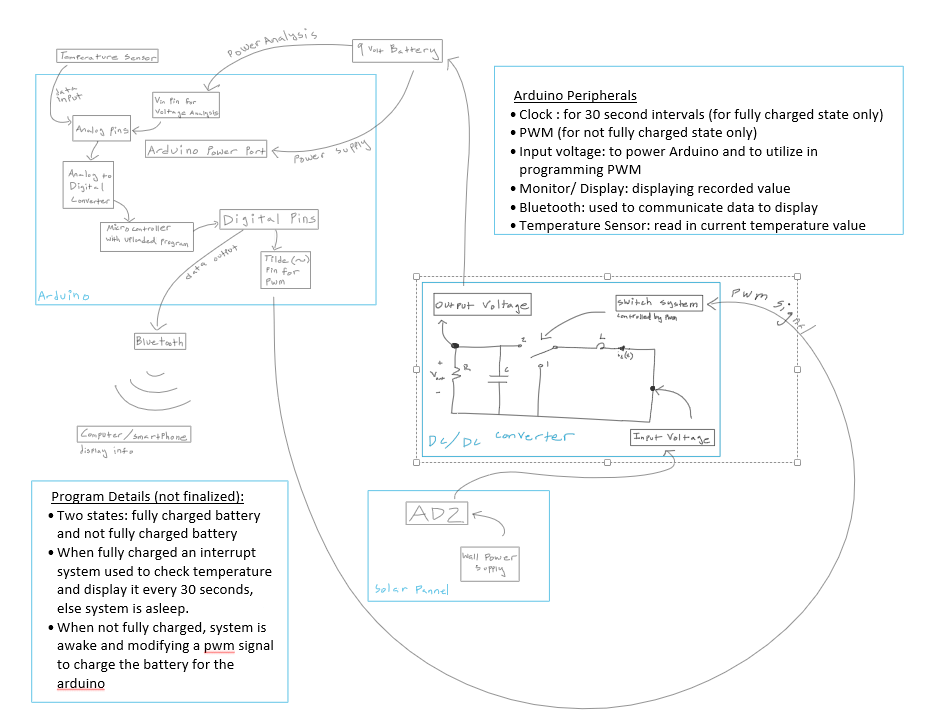
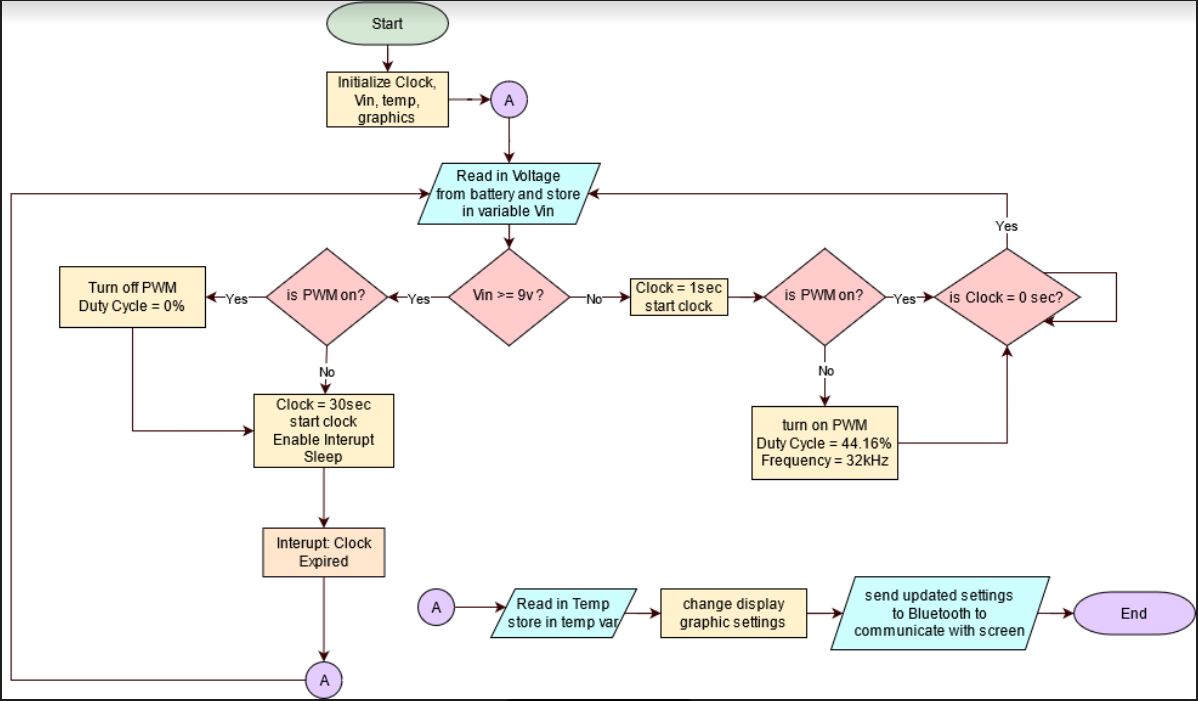
Milestone Report 3

**Introduction:**

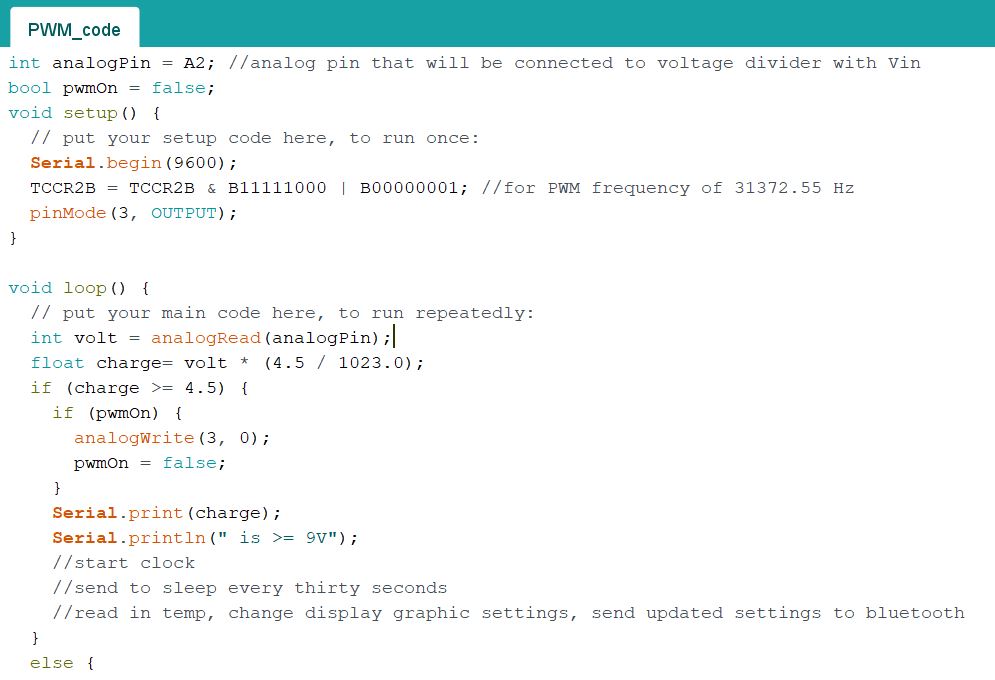
At the beginning of this project, we decided on certain goals and deliverables for each milestone. For milestone three, our deliverables were to show the battery charging and discharging waveforms as well as refining our simulation and building the physical DC/DC converter. Our goal for the hardware was that the DC/DC converter should be able to recharge the 9V battery when the battery is less than fully charged. The DC/DC converter should also have PWM feedback control, i.e. the arduino will output a PWM signal to charge the battery only when the voltage is below the needed threshold. Our goal for the software was for the Arduino to communicate with the DC/DC converter through PWM, and the Arduino should operate based on the charge of the 9V battery. As a result, our deliverable was to show battery charging and discharging waveforms.

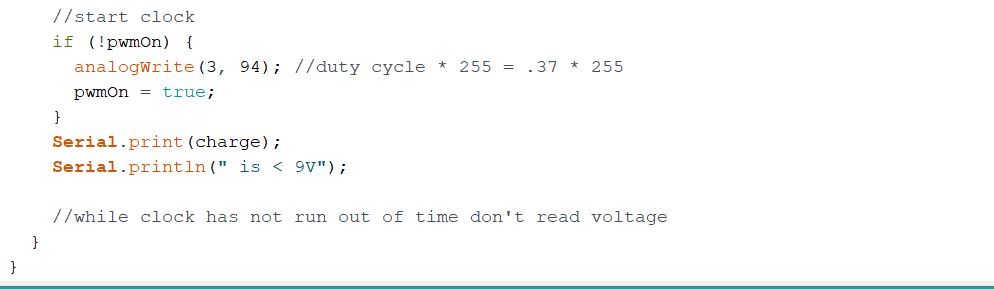
**High Level Design:**

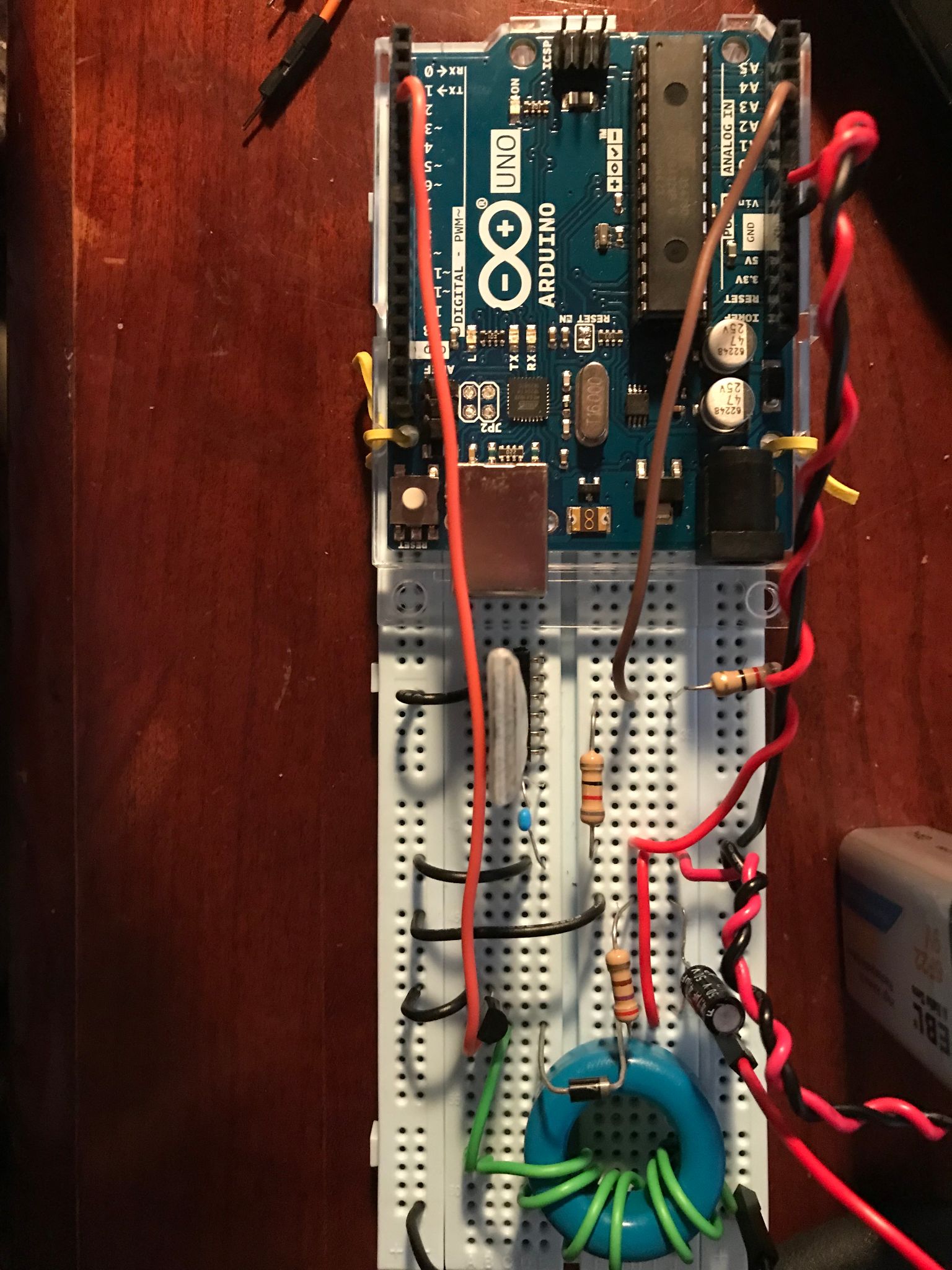
Shown below is the most recent and up to date version of the entire system presented as a block diagram. The focus of this milestone was to complete the programming needed to begin charging the battery and to build a physical circuit to further refine the values of the chosen components. Below is the flowchart that will be used as a blueprint for the code related to this project. It outlines the two main states of the code: when the battery is above 9 volts, and when the battery is beneath this threshold. The details of this process are discussed in the following sections. 

****

**Detailed Design Components:**

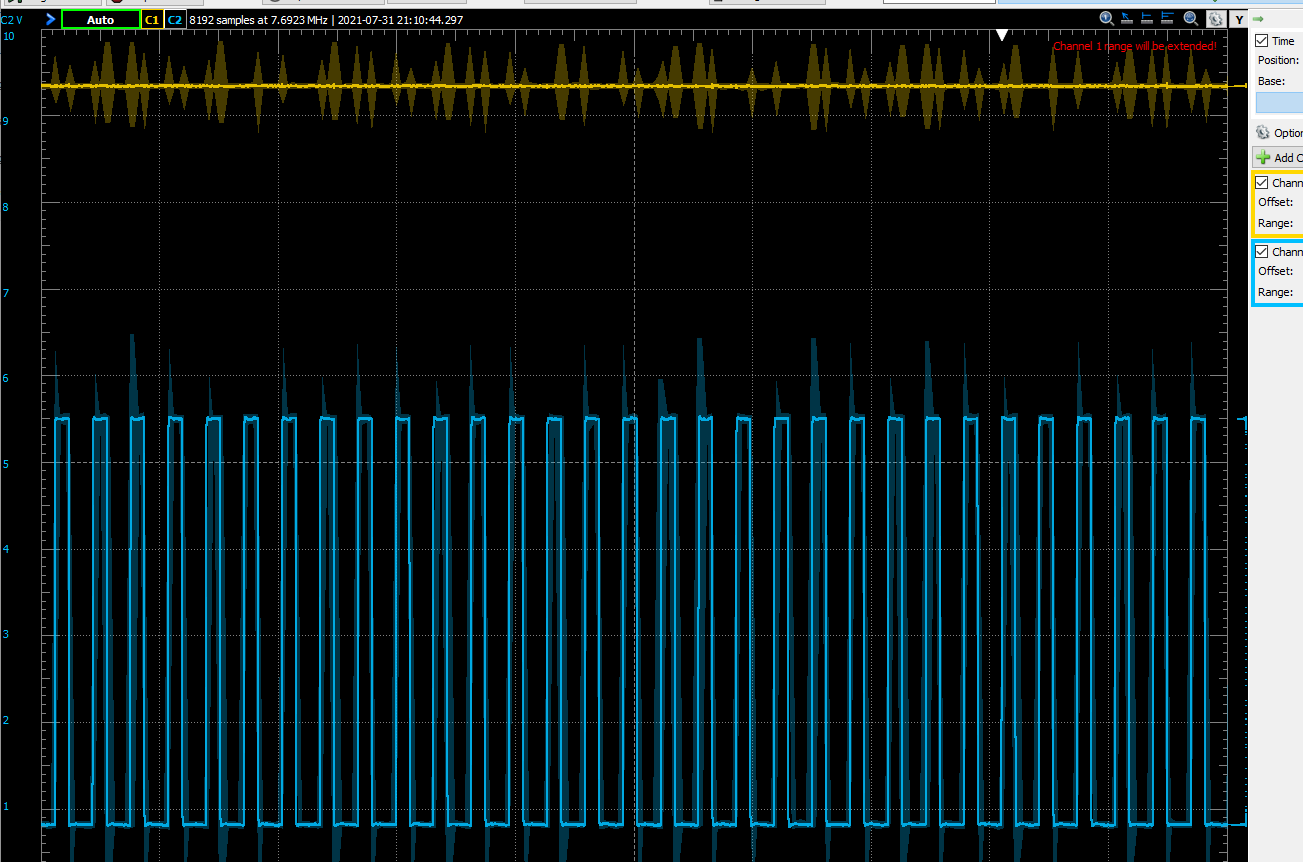
Shown below is the Arduino code by which the Arduino is able to communicate with the DC/DC converter through PWM and operate based on the charge of the 9V battery. To start off, we initialized and defined certain global variables that the code can access and change later on. We defined our analog pin as A2 and created a boolean to signify whether the PWM is on or off. We had to read the voltage from an analog pin instead of the Vin pin directly because Arduino does not have a method to read from Vin. We also found that the highest voltage an analog pin could read is 5V. For these reasons, in order to read the voltage coming from the Vin pin, or the voltage that the battery is sending into the Arduino, we had to use a voltage divider with two equal resistance values in which the input voltage of the voltage divider is the voltage going into the Arduino and the output voltage was the voltage going into the A2 pin. Using this method, if the battery is fully charged at 9V, the analog pin will read a voltage of 4.5V. Additionally, through incremental testing, we found out that the analogRead function maps input voltages into integer values between 0 and 1023. In order to see if the voltage is greater than or equal to 4.5V, we had to multiply the value that we got back from analogRead by 4.5/1023. If the charge is greater than or equal to 4.5, we checked if the PWM was on. If the PWM was on, we turn the PWM off by setting the duty cycle equal to 0. Similarly, if the charge was less than 4.5 and the PWM is not on, we turn the PWM on by writing the output with a duty cycle of 0.37, which when multiplied by 255 is 94.

****

Another facet of this project that was completed during this milestone was the building and testing of the DC/DC converter and gathering the charging and discharging waveforms. Below is an image of the physical circuit: 

The green wires are used solely for the inductor and transferring it’s voltage, the black and red represent the standard ground and power respectively, orange is the PWM waveform, and the brown wire is the value being read by the arduino to determine the voltage of the battery. At first we were uncertain as to how we would read the value, as the arduino needs 9 volts of power to operate, but the pins can only tolerate a lower value. The solution to this problem was a voltage divider. Two equal valued resistors of 80Ω were used to cut the 9 volts in half. The resulting 4.5 volts is used to determine if the battery is fully charged or not. Once the physical circuit was built, some values changed. For example,the inductor has a value of 414𝛍H instead of 390𝛍H and the value of the duty cycle for the pwm was changed to 37%.

**Validation:**

Once the physical circuit was built, and the program was completed it was time to begin testing and refining our values. First we began by testing the code on it’s own. This was to avoid possibly harming the physical circuit and the battery. The program was tested by powering the arduino via it’s power cable and changing the numeric values within the code to have it output the PWM if the read voltage was below 5 volts instead of 9 volts. The reason for this choice was due to the AD2 not being able to naturally output a DC voltage of 9 volts. It was simpler to just change a few numeric values in the code after confirming that the concept worked. The PWM was also tested by checking the outputted waveform with the waveform generator. This waveform is one of the two displayed in the image below (the blue channel two signal). The yellow waveform is the charging waveform of the battery. Our circuit switched between charging and discharging very quickly so it was difficult to showcase the curves via waveforms, however, we temporarily connected the Arduino to the computer so that we could see the output voltage values (shown below). Print statements were included in our program so that validation of the program and the circuit were more streamlined and straightforward. In future milestones we intend to polish our code and gain a more refined charging and discharging waveform. A potential plan is to implement a lower threshold voltage to begin charging the battery, and a higher value at which to stop charging the battery. This is due to the fact that the battery wouldn’t charge above 3.76 or so. This may be to how the physical circuit was wired, however this will be further tested and discussed in the next milestone report. 

**Conclusion:**

By using an incremental design approach, we were able to reach our goals for both hardware and software as well as produce our intended deliverables. Through milestone three, we learned about how to charge a rechargeable battery through PWM feedback control and how to validate the PWM feedback control through charging and discharging waveforms. Furthermore, we learned how to implement a PWM with a specified duty cycle in the arduino code. Overall in the arduino code, we learned how to read an analog pin and turn on and off the pwm based on the voltage of the analog pin. In future milestones we intend to refine our current results and to complete the overall project.

**Authorship Page**

Introduction

Written By: Saakshi Naveen

Edited By: Illa Rochez

Highlevel

Written By: Illa Rochez

Detailed design components

Written By: Saakshi Naveen and Illa Rochez

Edited By: Illa Rochez

Validation

Written By: Illa Rochez

Conclusion

Written By: Saakshi Naveen

Edited By: Illa Rochez