Critical Design Review

Submitted to

Dr. Janiszewska

Prepared by

Matthew Geiger

Nick Stassen

Ben Bazan

Engineering 1182

The Ohio State University

College of Engineering

Newark, OH

April 21st, 2022

Table of Contents

List of figures

List of Tables

Executive Summary:

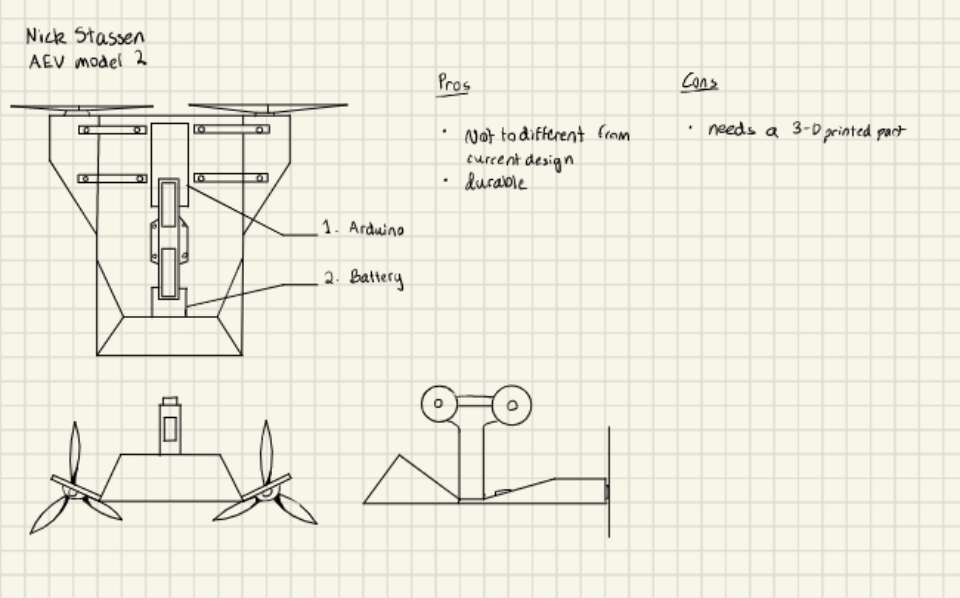
The purpose of this project is to learn about energy management, operational efficiency, and operational consistency while applying coding skills. To learn about these aspects, a situation was presented that required a small Advanced Energy Vehicle (AEV) to be created that would automatically hook up to a cart that can be transported to different stations along the track. The goal was to create one of these AEVs to be the most efficient and consistent vehicle possible. The AEV will use absolute position tracking to acutely position itself in the required spots while being the most efficient as possible. In this certain situation, an energy efficient AEV is required because the park this would be built for is secluded and does not have much access to electricity. Because the vehicle has limited access to electricity, it must use as little energy as possible hence why it must be as efficient as possible. To produce a design for an energy efficient AEV, 2 different AEVs were built and tested, while various aspects were then inspected to determine which one was more efficient. The first AEV design, design 1, was based partly from the original model given at the beginning of the lab. Design 1 was built and modified to use the 3-bladed propellers and as well was made for components to be attached on the bottom. Design 1 was then tested, and the data was taken from the EEPROM and converted into useful data using a MATLAB script (see appendix), Design 1 was then taken apart and Design 2 was constructed. Design 2 took the same energy efficient approach as Design 1 making sure to use the 3-bladed propellers as well as a bottom mounted Arduino. However, Design 2 was made for its stability as well as looks using the X-Wing from *Star Wars.* Design 2 program was then run, and the data was taken from the EEPROM and converted once again into useful information using the MATLAB script. The datasets from the 2 designs were then compared by looking at the energy per kilogram (EPK) for each design. Design 1 used 257.7 J/Kg and design 2 used 326.2 J/Kg. With Design 1 using nearly 69 joules of energy per kilogram less than design 2 it was chosen to continue in the design process. Design 1 then completed two tests in which the AEV would traverse the entire length of the track (Full-track run), including the stop on the incline. One test would be with the weighted cart and one would be with the unweighted cart. The mass used to calculate the EPK would be the mass of the AEV as the goal is to measure the efficiency of the AEV itself. With the weighted cart the AEV used 1,346.28 J/Kg of energy and with the unweighted cart the AEV used 1408.03 J/kg of energy (Table 1).

Introduction:

The AEV project was created to develop an energy efficient and autonomous vehicle to transport people and cargo along a monorail inside a national park. The vehicle must conserve as much energy as possible since the power at the park is limited and stranding the tourists or cargo is not an option. The project is important because creating a model AEV allows for rapid prototyping and configurations that if otherwise started on a large scale would take months or even years to complete, however using small scale AEVs, major changes can be completed in days and smaller changes can be completed in hours. The AEV will focus on energy management, operation efficiency, and operational consistency. The AEV will go slow enough to allow for the passengers and cargo to have a comfortable ride ensuring that passengers do not fall off the cart. The AEV will have energy per kilogram optimized to prevent energy waste.

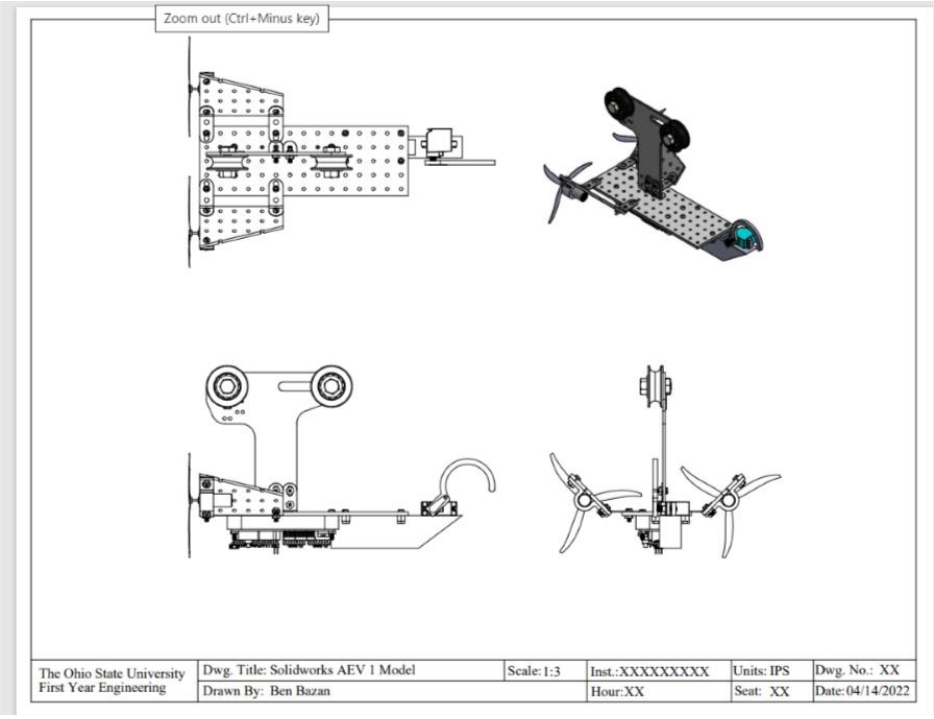
AEV Initial Concepts:

In Performance Test 1 two designs were compared to see which was better suited to the task explained above. Design 1 was inspired by the sample AEV given at the beginning of the lab, from there it was adapted for the highest energy efficiency and the lowest mass, as well improving the aerodynamic drag experienced by the model through the addition of a battery carrier which was 3-d printed. Design 2 attempted to further these improvements made through, attempting to further decrease drag by decreasing the width of the airframe used. This also attempted to lower the mass used and decrease the energy used per kilogram. Design 2 was made for its striking resemblance to the x-wing from *Star Wars.* Additionally, it would be more stable. The designs originally created in lab 1 were all abandoned besides Design Model B (Figure 1), this was because Design Model B incorporated the best features known at that time. However, since then experiments have improved the knowledge and better choices have been made. For instance, the choice to put the Arduino on the bottom to allow for an optimal center of gravity location. Or the incorporation of an aerodynamic battery carrier which decreases drag and optimizes C.G location. All changes were based off experience gained in experiments. The prototypes allowed for the AEV to have the best starting configuration possible that would then be edited to a more efficient model. For instance, the aerodynamic shield in Design Model B allowed for the thought of the aerodynamic shield on the battery carrier. Which will also be used to hold the servo that will hook onto the cart (shown at the start of the AEV in figures 2 and 3.

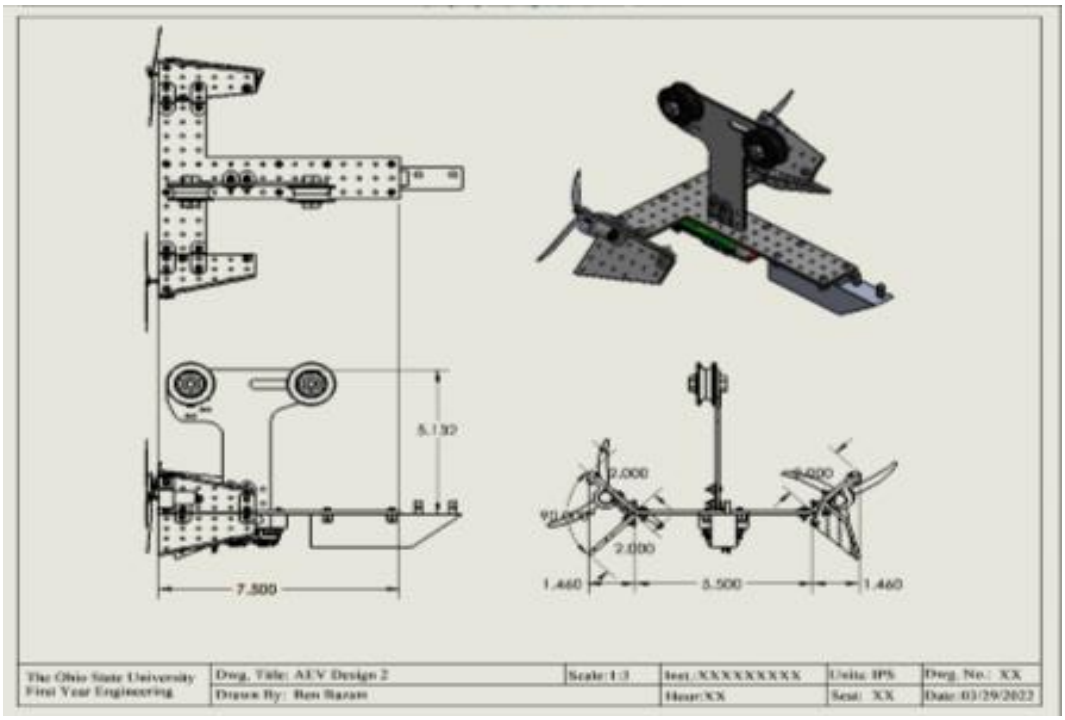


(Figure #1): Orthographic projection of design model B (Nick Stassen's design)

Results and Discussion:

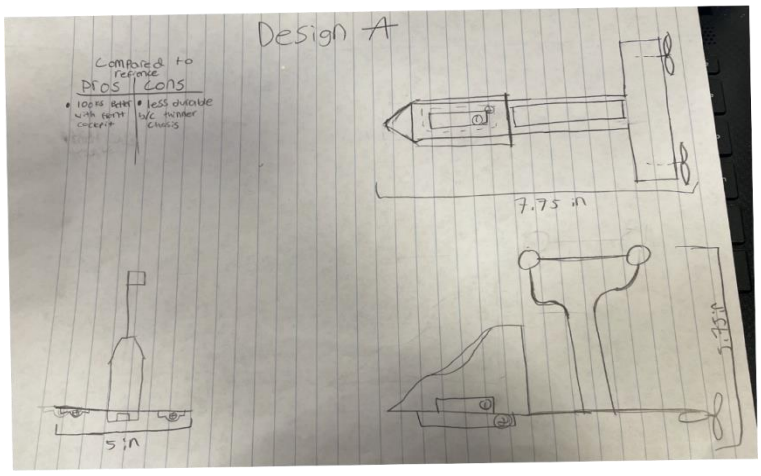


(Figure #2): AEV Design 1 Solidworks model

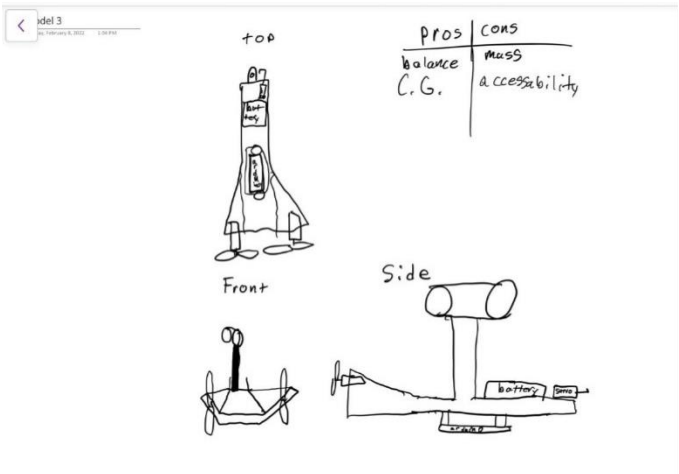


(Figure #3): AEV Design 2 Solidworks model

Performance Test 1 generated valuable information in determining the most energy efficient AEV (Advanced Energy Vehicle) design. Both designs in figures #2 and #3 above were similar in many aspects: monorail hangar attachment placement, hangar attachment used, Arduino assembly controller placement, and others. The differences between the vehicles were the number of wings, the central plastic base used for holding all the components, the motor placement, and the battery placement. This 3-D printed part on both vehicles seen above as the pale gray object on the nose of the vehicles was created to holster the battery on the front of the vehicle while allowing better aerodynamics. The servo which connected the AEV to the cart was also fitted on the 3-D printed part. Three designs were originally created by the group in experiment 3 (with the addition of a sample design for a total of 4) which encouraged creative thinking regarding combinations of parts for energy efficiency. These combinations in experiment 3 influenced the structures of designs 1 and 2 in several ways- most notably the need for a 3-D printed part (See figures #4, #1, and #5 for design concepts in experiment 3).



(Figure #4): Orthographic projection of design model A (Ben Bazan's design)



(Figure #5): Orthographic projection of design model C (Matthew Geiger's design)

Concept Screening and Scoring				
Success Criteria	Refrence	Design A	Design B	Design C
Stability	0	0	+	+
Eco-fuel	0	0	0	0
Look	0	+	+	0
Maintenance	0	0	0	0
Durability	0	-	+	0
Cost	0	-	-	0
Environmental	0	0	0	0
Sum +'s	0	1	3	1
Sum 0's	7	4	3	6
Sum -'s	0	2	1	0
Net Score	0	-1	2	1
Continue?	Combine	No	Yes	No

(Table #2): Concept screening for designs A, B, & C

		Refrence		Design A		Design B		Design C	
Success Criteria	Weight	Rating	Weighted score	Rating2	Weighted score	Rating3	Weighted Score4	Rating5	Weighted Score6
Balance	5%	3	0.15	4	0.6	4	0.6	4	0.6
Minimal blockage	15%	3	0.45	3	1.35	3	1.35	3	1.35
Center-of-gravity location	10%	2	0.2	4	0.8	4	0.8	4	0.8
Maintenance	25%	3	0.75	3	2.25	3	2.25	3	2.25
Durability	15%	2	0.3	2	0.6	4	1.2	4	1.2
Cost	20%	3	0.6	2	1.2	2	1.2	3	1.8
Evironmental	10%	3	0.3	3	0.9	3	0.9	3	0.9
Total Score			2.75		7.7		8.3		8.9
Continue?			No		no		yes		no

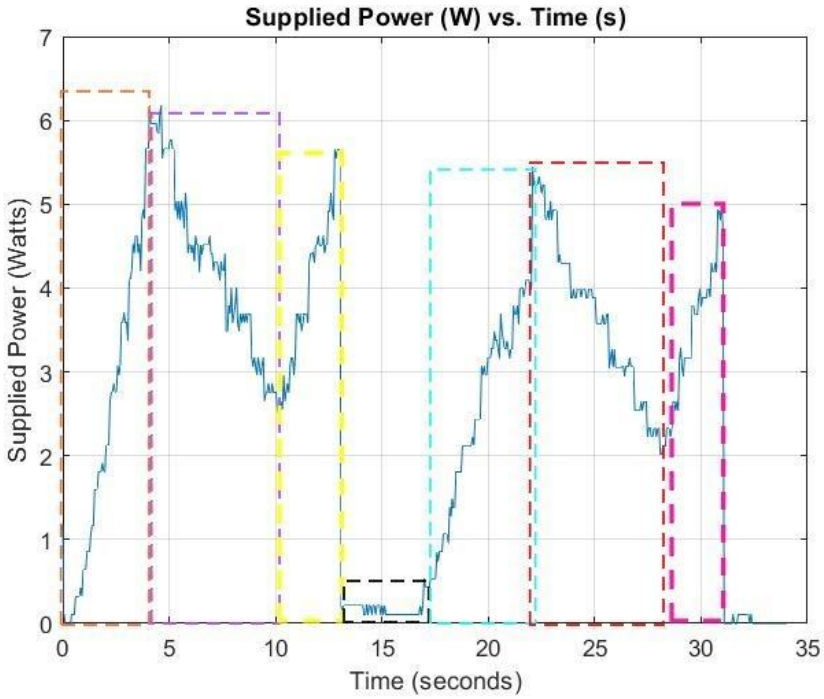
(Table #3): Concept scoring for designs A, B, & C

Design 1 was selected by judging other designs based off a design concept screening and scoring sheet (refer to tables #2 and #3 for concept screening and scoring sheets respectively). A design was created by each team member with the addition of the sample design (from experiment 1) for a total of 4 (refer to figures #4, #1, and #5 for orthographic projections of the 3 original designs). These designs were then judged on characteristics that were vital to the success of the AEV: balance, center-of-gravity location, durability, cost, environmental impact, look, etc. Design B from the concept screening and scoring was chosen to continue onto Performance Test 1 under the new name of “Design 1”. Design 2 in Performance test 1 was made to incorporate the fundamental qualities of design 1 based off the concept screening and scoring sheet, such as low energy usage and center of gravity location however it would also add stability and looks to the AEV. The goal with Design 2 was to create a better looking AEV more stable, as well as improving the aerodynamic characteristics of the design by decreasing the width of the base section.

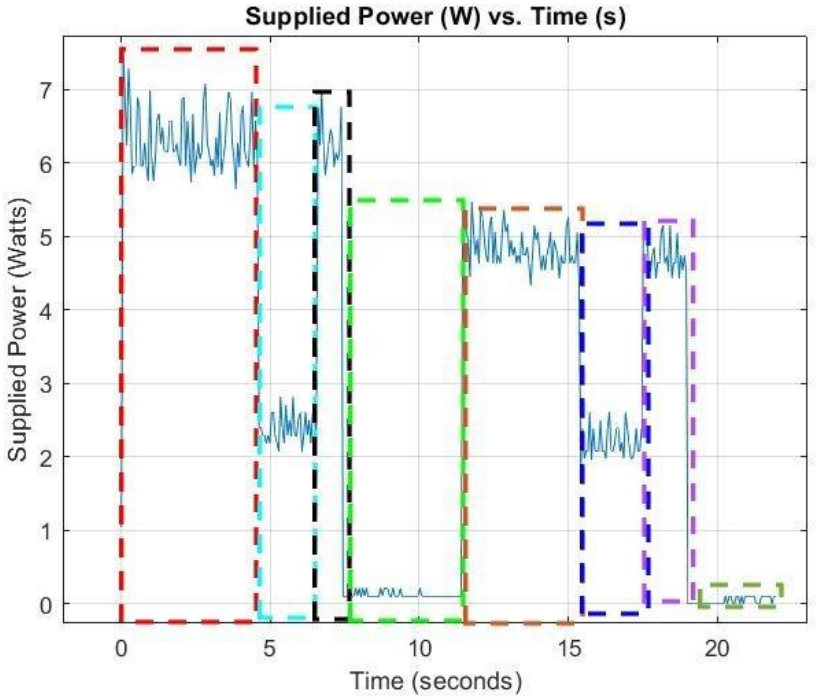
Ultimately, the double wing structure of Design 2 (figure #3) led to an increase in weight as well as a decrease in the thrust experienced by the AEV because of an increase in drag between the

propeller slipstream and wing structure, making Design 2 less efficient. While the graphs and data alone support this energy inefficiency hypothesis, Design 2 was unable to traverse the incline portions of the track with the code that Design 1 ran on. This comparison allowed a simple conclusion to be made about the energy usage of Design 2. However, subsequent tests were done to ensure this, the half-track run code used was then modified to allow Design 2 to complete the half-track run. Modifications consisted of increasing motor power percentages and changing the absolute position values where the AEV needed to stop. Design 1 consumed a total of 69.323J of energy completing the half-track run (Table #4), while Design 2 consumed a total of 114.18J on the same run (Table #4). This was a significant difference which the team did not expect. The team made Designs 1 and 2 to be similar, however Design 2 performed poorly in comparison. This information contributed to the decision to continue with Design 1.

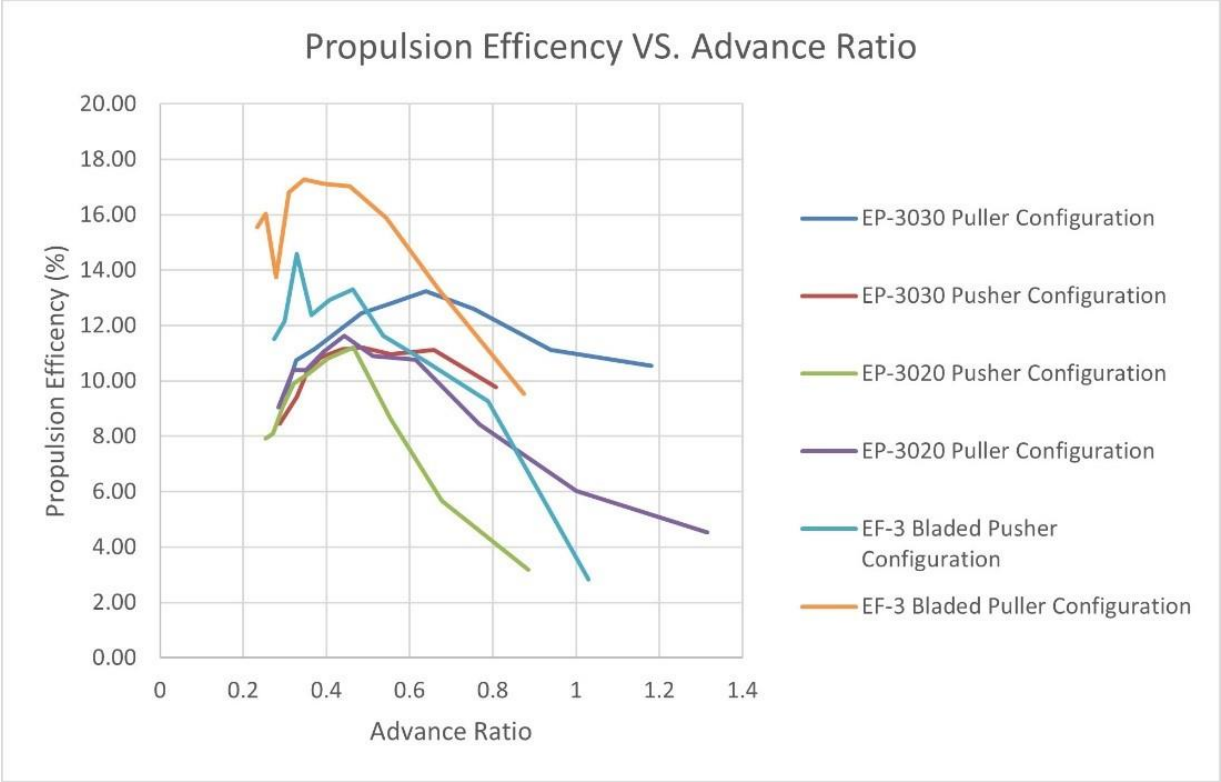
Performance Test 1 was important to the design process because it demonstrated the need for a light vehicle. Future changes made to the AEV would need to help make the vehicle lighter or stay the same weight and improve aerodynamic characteristics. Design 2 was less aerodynamic than design 1 for one reason, the double winged structure increased the drag between the wings and the propeller slipstream, which caused the AEV to benefit from less force than the motors were producing. Furthermore, one of design 2's motors were offset a couple degrees. While it was mostly unnoticeable to the naked eye, this error likely impacted the motors contribution to the overall thrust of the vehicle. However, this offset motor was an innate part of the vehicle based on the parts available, meaning it could not be fixed without an improvement in available building parts or manufacturing new building parts. This process would likely be a waste of time and possibly add more weight to the vehicle. The System Analysis tests 1 and 2 gave the team direction with how to code the designs for Performance Test 1. This experiment involved testing different coding functions which have the same outcome but different ways of getting there. Most notably, these tests compared using the celerate command and the motorSpeed command. Figures #6 and #7 show the celerate and motorSpeed performances during their respective flat track runs. The motorSpeed command uses less total energy in this run and as well takes less time. Additionally, a fatal error could occur when using the celerate command, if the cellerate command were to be active (meaning the code is still increasing the power of the motors) and the AEV were to go past a position that was used in the next line of code to stop the AEV (in Arduino it would look like this: celerate(10,40,4,4); goToAbsolutePosition(10*(8/3.901))), then the AEV would never stop and the safety of passengers would be at risk. This information showed the team that the motorSpeed command should be used primarily rather than celerate, however small time values for celerate could be used to reduce strain on motors. The half-track code was created using this style. The full track coded design and final vehicle design were heavily influenced by performance tests.



(Figure #6): describes the AEV flat track run, supplied power versus time, using celerate commands, along with a phase breakdown.

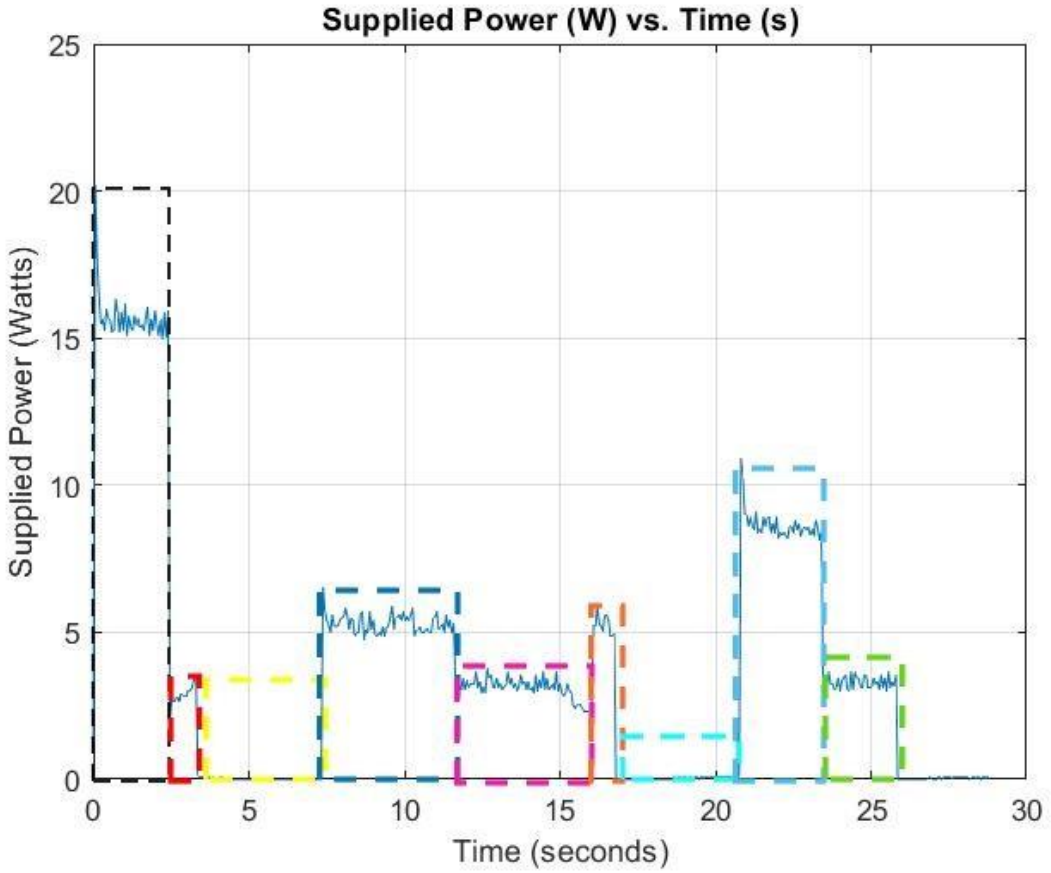


(Figure #7): describes the AEV flat track run, with supplied power versus time, using motorSpeed commands. Along with a phase break down.



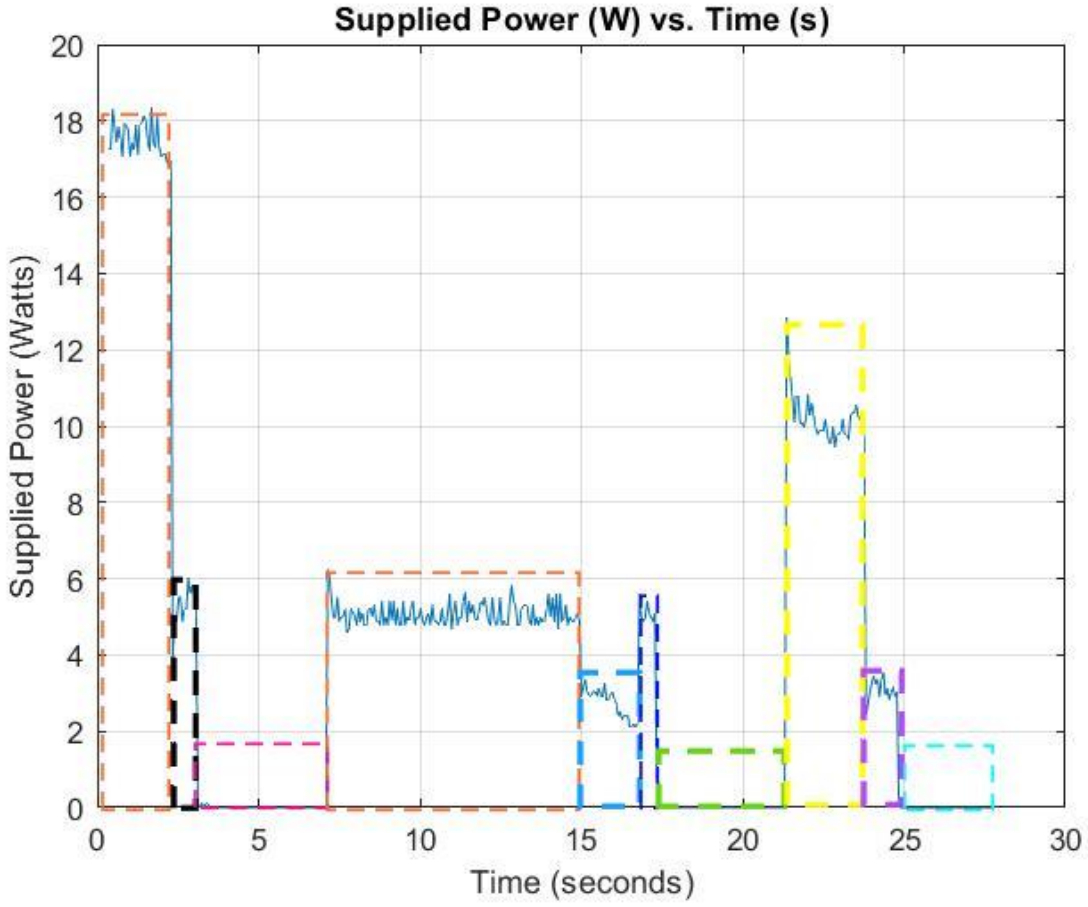
(Figure #8). System Efficiency vs. Advance Ratio

Experiment 4 tested three different propellor types each in 2 different configurations. These two configurations were pusher and puller (tractor). As seen above in figure #8, EF-3 bladed puller configuration has the best propulsion efficiency for most of the advance ratio. In addition, the EF-3 blade pusher configuration comes second place until around 0.5 advance ratio. This showed that the EF-3 bladed propellors will give the highest efficiency at lower advance ratios (approximately 0.7 and below), but in both configurations. These advance ratios tend to occur at motor power percentages around 30-40%. The code used in all AEV runs have motor power percentages primarily around 10-40% meaning EF-3 was the most suitable pick for the propellors.



(Figure #9): Supplied Power vs. Time (with phase breakdowns) for Design 1 Half-track run

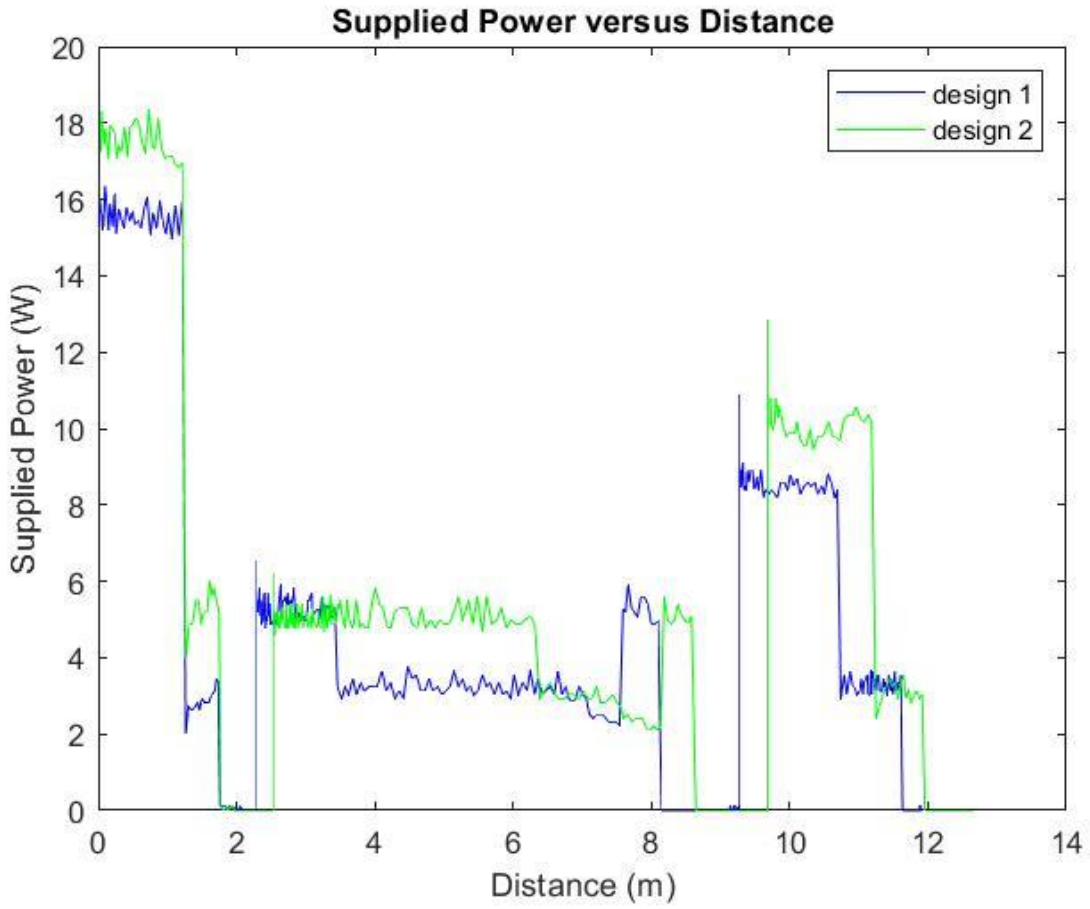
Figure #9 shows the supplied power to the motors vs. time on the half-track run for design 1. The different phases of the run are indicated by colored dashed boxes. The highest energy phase was observed at the start of the run in phase 1. High energy was observed due to the AEV needing to get up an incline in pusher configuration- the less optimal configuration. Phase 2 represents the vehicle traveling on the horizontal after the incline and getting over to the pickup station at a low speed. Phase 3 is where the vehicle stopped at the station for 4 seconds and then reversed. When the vehicle is braked, energy levels are approaching zero. Phase 4 is where the vehicle accelerated along the horizontal before going down the now decline. Phase 5 shown in magenta is where the vehicle reduced its power not only because it was going down a decline and thus had more speed but also because it was going around a turn near the end of the phase. Phase 6 in orange shows the power after the vehicle made it around the turn and needed more power. In Phase 7 power was cut to come to a stop at “the waves”, this lasted 4 seconds. In Phase 7 shown in a cyan dashed box is where the vehicle increase power to make it up the incline, note how its is significantly less power because it is in puller configuration. In Phase 8 the vehicle uses a small amount of power to traverse the horizontal for another 3 feet.



(Figure #10): Supplied Power vs. Time (with phase breakdowns) for Design 2 Half-track run

Figure #10 shows the supplied power to the motors vs time with the different phases being indicated by dashed boxes (same as figure #9). As seen above, design 2 reached a maximum of 18 Watts of supplied power whereas in Figure #9 design 1 reached a maximum of only 15 Watts (both maximums being in phase 1). This trend of design 2 using more power was consistent throughout the entire comparison of the two runs. Like design 1, phase 1 was the highest energy phase because it involved the AEV going from stopped to ascending an incline. This required a significant amount of power and design 2 was both heavier and less aerodynamic. There two reasons most likely account for the increase in supplied power in design 2 compared to design 1. Phase 2 shows the AEV using a lower motor speed to glide to the pickup station. Phase 3 is the AEV breaking at the pickup station. Phase 4 is the largest difference between Figures #9 and #10 because it combined phases 4 and 5 from Figure #9 into one phase. This change happened because design 2 couldn't make the half-track run on the lower power of phase 5 from design 1's code. This meant the power needed to be increased during that time interval of the run for design 2. Ultimately the team just continued the phase 4 from design 1's code into phase 5 of design 2's code, and this worked. Phase 5 shows the deceleration around the bend on the half-track run. Phase 6 is where design 2 needed a little more power after going through the bend. Phase 7 is the AEV stopping at the waves location along the half-track run. Then for phase 8, the second most energy-intensive phase, the vehicle

went up the second incline in puller configuration. Phase 9 shows the vehicle being at the top of the incline and inching toward the stop. Phase 10 shows the vehicle stopping.



(Figure #11): Supplied Power vs. Distance (both designs plotted together)

As mentioned above, design 2 clearly used more energy than design 1. Figure #11 above shows the supplied power to the motors in designs 1 and 2 vs the distance they traveled. Figure #11 agrees with the observation that design 2 used more energy. Interestingly, they show the same levels of supplied power in some places. However, This happened because they are coasting at those points. Around the 2-8 meters mark, the graphs looks somewhat the same because the designs are both coming down from the first incline and already have potential energy just from the height. This means less energy was needed to travel further and they cruised up until they stopped around 8-9 meters.

Phase	Arduino Code	Energy per Phase (J)
1	motorSpeed(4,45)	37.86
2	motorSpeed(4,14)	3.0633
3	brake(4) goFor(4)	0.949
4	motorSpeed(4,22)	22.47
5	motorSpeed(4,16)	13.92
6	celerate(4,15,12,1) motor Speed(4,22)	3.64
7	brake(4) goFor(4)	0.492
8	motorSpeed(4,30)	23.115
9	motorSpeed(4,16)	30.894
		Total energy per kilogram 507.07

(Table #4): Supplied Energy for each line of code for design 1

Figure #9 visualized the supplied power vs time with phase breakdowns graphically for design 1. Table #4 above shows the same concept in the form of total joules per phase and the code executed at the phase. As mentioned before, phase 1 required the most energy for designs 1 and 2. This is because the motors need to be ran at higher powers to get the designs from stopped to moving up an incline. As seen above in Table #4, the AEV starts off with the motorSpeed function to start the motors. The First parameter represents both motors being activated and the second represents the power percentage the motors are to be ran at. 45% is the highest motor power percentage the team used for design 1. The energies per phase where the AEV was braked equaled around 0 which was expected. The AEV motors were only ran at 30% for the second incline because it was in puller configuration which is the optimal configuration to be in.

Phase	Arduino Code	Energy per phase (J)
1	motorSpeed(4,50)	34.303
2	motorSpeed(4,20)	4.6043
3	brake(4) goFor(4)	0.022
4	motorSpeed(4,22)	39.727
5	motorSpeed(4,16)	5.8493
6	celerate(4,15,12,1)	2.435
7	motorSpeed(4,22)	2.281
8	brake(4) goFor(4)	24.14
9	motorSpeed(4,35)	3.4416
10	brake(4)	0.09102
		326.232 Total energy per kilogram

(Table #5): Supplied Energy for each line of code for design 2.

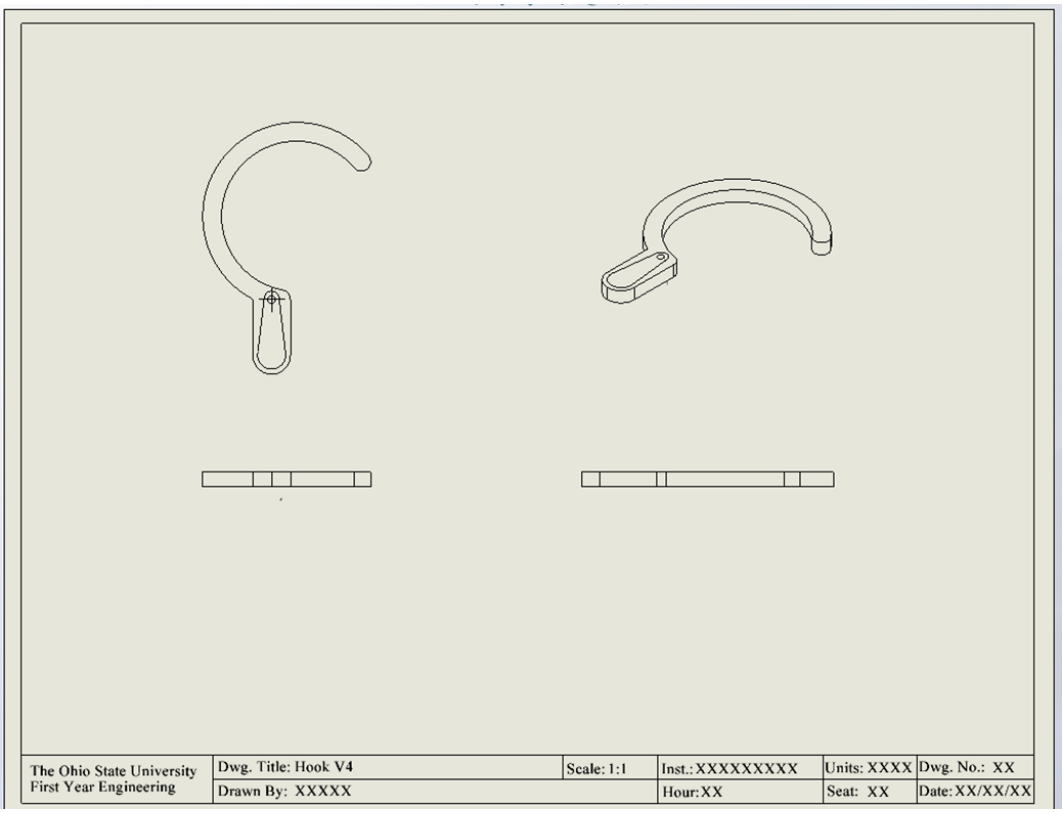
Figure #10 visualized the supplied power vs time with phase breakdown graphically for design 2. Table #5 above shows the same concept in the form of total joules per phase and the code executed at the phase. Table #4 and #5 show practically the same code. However, design 1 in Table #4 tends to use less joules per phase than design 2 with a few exceptions. In addition, the

code in Table #5 for design 2 generally has larger power percentages for the motors. This is due to design 2 being less aerodynamic and heavier (thus needing more power to move).

Design	Energy (J)	Weight (kg)	Energy per kilogram (J/kg)
1	69.323	0.269	257.7
2	114.18	0.35	326.2

(Table #6): Energy per kilogram for each design

Table #6 shows the total energy of both designs on the half-track run, the weight of each design, and the energy per kilogram of each design. As seen above, design 1 uses roughly 45 less total joules than design 2 making it a clear winner in terms of total energy cost. As mentioned earlier in the discussion, design 2 weighed more which likely contributed to the high energy consumption. Also, the energy per kilogram of design 1 is a clear winner as seen above with its energy per kilogram being 257.7 J/kg compared to design 2's 326.2 J/kg.



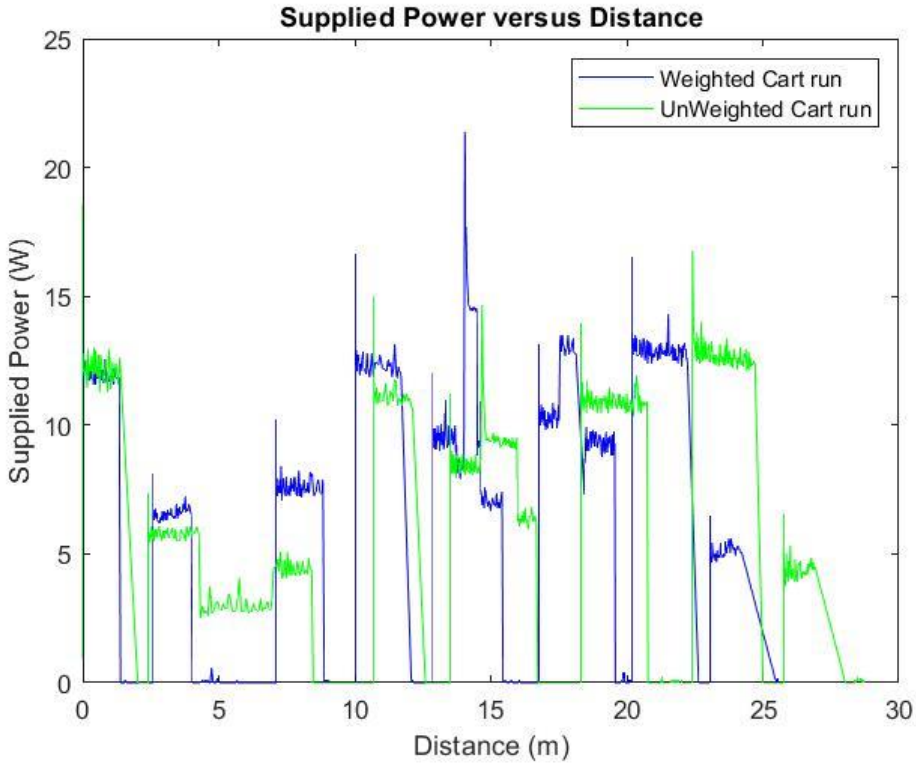
(Figure #12) details the solidworks model of the hook used by the AEV.

In order to connect to the cart a hook was designed using solidworks (see figure ##). The hook was made to essentially snap into the arm of the servo a screw was used to secure the arm into the gear of the servo. The hook was designed to have a small width so it could enter the loop attached to the cart with no problem and also have a medium radius so that the AEV didn't need to be too close to the cart where other parts of the AEV could interfere with the cart/hook. The

medium radius also allowed the AEV-cart configuration to pass around bends without the hook pulling the cart off of the track. Another advantage of the medium radius of the hook was the allocation of changes in height, when changing from the flat portion of track to an incline portion the AEV would be at an angle compared to the cart, while this only occurred for a short amount of time (until the cart reached the incline portion of the track) it was essential that the hook would still be able to pull the cart at these angles. The only problem the AEV experienced when connecting to the cart was the speed at which the servo was moving the hook to connect to the cart. The servo would write itself to zero to quickly and if the cart was in the wrong position the hook would hit the loop of the cart and fall off the track. To change this a servo write function was created. Although simple it proved to be an extreme benefactor when connecting to the cart. The function would initialize the integer i at a starting value when connecting to the cart the value would be 70 degrees. Using a for loop the integer would decrement every 20ms using i— until i was equal to 0. This meant the servo would take around 1.4 seconds to go from its initial position to its 0 position, allowing the team ample time to move the cart if in the wrong position. The only problem experienced during the full track run was with the weighted cart, one of the "dogs" ears' would smack into a bracket holding the track on the bend. To get around this issue the power of the AEV was shortly increased and then would be turned off, this allowed for a separation distance between the cart and the hook which allowed the cart to pass without the ear of the dog smacking into a bracket. Although this does slightly increase the power consumption for the AEV around the bend it is necessary to complete the track.

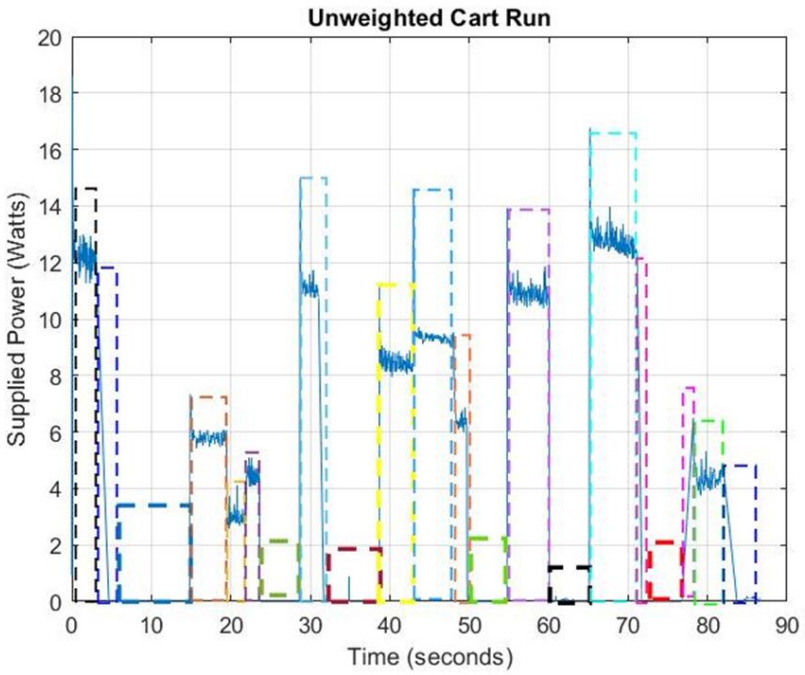
The main change for the energy optimization that was performed was the inclusion of the incher function. This incher function that was included in the unweighted cart run proved to be an integral part of the code in the weighted cart run. With the mass of the cart being higher other coding strategies needed to adjust motor speeds and braking distances in order to account for this change in momentum. However using the incher function the braking distance values changed only slightly. This is because the incher function operates at what is essentially the bare minimum to move forward, so when a braking distance is reached the AEV stops almost immediately.

Although no changes occurred to the physical structure of the AEV several changes were made to the code of the AEV. The main change required was coming from the grand canyon down toward the waves, a test proved that if the AEV-cart configuration came into the bend at too great of a speed the ear of one of the dogs would smack into the bracket holding the track and the AEV and cart derailed. This led to the motor power being cut near the beginning of the inline so the AEV-cart configuration would coast toward the beginning of the bend, from there motor speed was written to a value that would prevent the AEV-cart configuration from going too fast and derailing. The other main change also dealt with the dog ear-bracket collision problem. When coming from the rocky mountains towards hocking hills the ear of the dog would collide with the bracket, it had to deal with the angle at which the AEV was pushing the cart. In order to avoid this the AEV would increase the power to the motors at the beginning of the bend and then cut the power to the motors approximately 4 inches before the collision area. This allowed for the AEV to gain speed before the collision area in which it would then coast through the collision area. This led to the loop of the cart and the hook of the AEV (the contact points of both vehicles) to gain some separation.

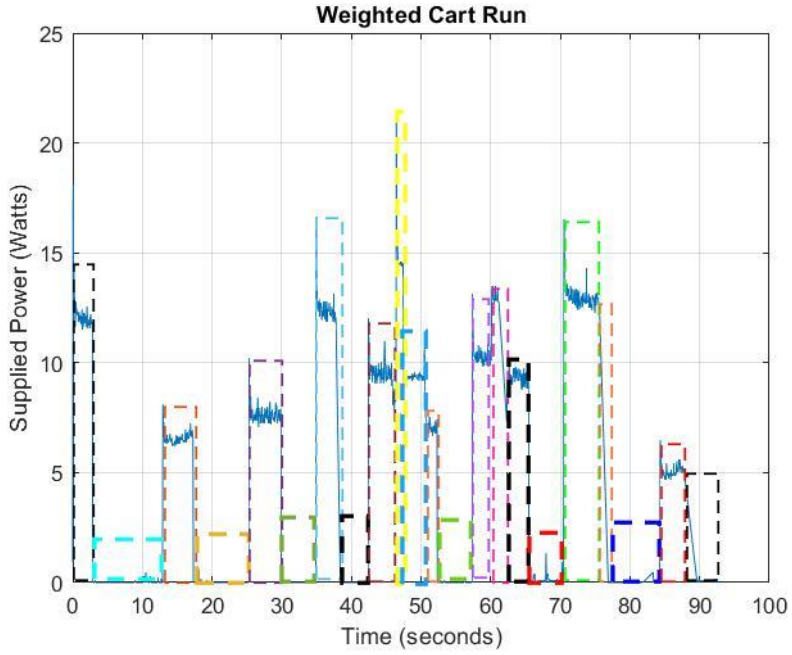


(Figure #13): Details the Supplied Power vs. Distance of the Weighted and Unweighted cart runs.

In the above figure the weighted cart run is shown to have a higher maximum usage of energy. This spike to upwards of 20 watts occurs when the AEV reverses its motors to stop on the incline portion near Alaska. This is because the weighted cart has a higher force of gravity in the x direction that requires a higher motor speed and thus a higher power percentage to offset.



(Figure #14): details the Supplied power vs. time for the unweighted cart run with energy phase breakdown.



(Figure #15): Describes the Supplied Power vs. Time for the Weighted cart run with phase breakdown.

When comparing Figures 14 and 15 there are a couple of important distinctions to be made. In figure 15 the fourth phase shows little to no power being supplied to the motors. This is because during this time the vehicle is coasting down the incline portion and headed towards the bend. Whereas the unweighted cart was currently supplying power to the motors. This was one of the main reasons for the energy per kilogram decreasing between the two runs. Another difference between the runs is in the violet box, or the 15th box from the left of figure 15. This momentary spike in power correlated to when the AEV was going around the bend. As mentioned above this increase in power was necessary for the AEV-cart configuration to make it around the bend and not get stuck from the ear of the dog hitting the bracket that supports the track. Similarities between the two runs include the main burst of power at the beginning of the run to make it up to the grand canyon to pick up the cart, as well as the braking maneuver done by the AEV when going down the incline towards the maintenance station.

Phase	Code	Time	Energy (J)
1	motorSpeed(4,40);	3.232	47.164
2	incher function	1.657	10.397
3	brake(4); rotateServo(0);	9.944	0.22
4	motorSpeed(4,24); goToAbsolutePosition(48);	4.47	25.98
5	motorSpeed(4,15); goToAbsolutePosition(-168);	2.65	8.637
6	motorSpeed(4,20); goToAbsolutePosition(-280);	1.988	7.577
7	brake(4); goFor(5);	4.8066	2.004
8	motorSpeed(4,40); goToAbsolutePosition(-588);	2.155	24.7636
9	incher function to -510	0.829	4.69
10	brake(4); goFor(7);	6.7955	0.065
11	motorSpeed(4,30); goToAbsolutePosition(-512);	4.8066	41
12	reverse(4); motorSpeed(4,36);	4.641	43.677
13	brake(4); reverse(4); motorSpeed(4,24);	1.49	10.452
14	goToAbsolutePosition(-420); brake(4); goFor(5);	5.304	2.93
15	motorSpeed(4,37); goToAbsolutePosition(-100);	5.14	56.65
16	incher Function to twelve	4.972	2.348
17	motorSpeed(4,43); goToAbsolutePosition(130);	6.961	80.31
18	brake(4); goFor(5); rotateServo(60);	4.807	0
19	motorSpeed(4,20); goToAbsolutePosition(120);	1.988	7.067
20	incher function but modified to brake on descent	2.9834	13.13
21	same incher function	3.149	5.64
Total Energy per kilogram			1,408.00

(Table #7): describing the energy phase breakdown with code for the AEV during the unweighted cart run.

Phase	Code	Time	Energy (J)
1	rotateServo(60); motorSpeed(4,40); goToAbsolutePosition(54*(8/3.902));	2.855	34.61
2	brake(4); goFor(5); rotateServo(0); goFor(4);	10.13	1.273
3	reverse(4); motorSpeed(4,26); goToAbsolutePosition(48*(8/3.902));	4.24	27.875
4	brake(4); goFor(8);	8.1	1.54
5	motorSpeed(4,29); goToAbsolutePosition(-142*(8/3.902));	4.788	35.11
6	brake(4); goFor(5);	4.788	0.51
7	motorSpeed(4,44); goToAbsolutePosition(-256*(8/3.902)); inching function	3.5	39.422
8	brake(4); goFor(4);	3.87	0.008
9	reverse(4); motorSpeed(4,33); goToAbsolutePosition(-268*(8/3.902)); motorSpeed(4,30); goToAbsolutePosition(-258*(8/3.902));	4.42	43.511
10	motorSpeed(4,52); goFor(1);	0.55	8.708
11	motorSpeed(4,36); goFor(3);	3.315	32.511
12	brake(4); reverse(4); motorSpeed(4,26); goToAbsolutePosition(-202*(8/3.902));	1.66	12.453
13	brake(4); goFor(5);	4.97	0.682
14	motorSpeed(4,35); goToAbsolutePosition(-120*(8/3.902));	3.13	32.28
15	motorSpeed(4,44); goToAbsolutePosition(-96*(8/3.902));	0.55	7.8
16	brake(4); delay(750); motorSpeed(4,65); delay(350);	1.29	15.84
17	motorSpeed(4,33); goToAbsolutePosition(-40*(8/3.902));	3.13	28.44
18	brake(4); inching function(-12*(8/3.902)); brake(4); goFor(5);	5.16	2.52
19	motorSpeed(4,44); goToAbsolutePosition(66*(8/3.902));	5.16	74.39

20	inchingFunction(79*(8/3.902));	1.105	7.71
21	brake(4); goFor(5); rotateServo(70); goFor(1);	7.73	1.59
22	reverse(4); motorSpeed(4,22); goToAbsolutePosition(55*(8/3.902));	3.315	21.388
23	modification to incher code allowing us to slow down on incline	4.6	4.99
total			total energy per kilogram 1346.28

(Table #8): describing the energy phase breakdown with code for the AEV with the weighted cart final test.

In tables 7 and 8 it shown that the total energy per kilogram decreased between the unweighted cart run and the weighted cart run. This is due to a couple of factors the main reasons being that as stated above the weighted cart configuration (WCC) coasted down the incline from the grand canyon towards the bend whereas the unweighted cart configuration (UWCC) used power during that time. Another reason for the decrease in energy usage could be the change in batteries. The unweighted cart run used the battery that the AEV had been using for all previous tests. However, one of the wires came out of the terminal used to charge the battery, while at first it was thought this could be overcame either by holding the wire into the terminal by hand or by soldering new wires to another terminal it could not and the battery died. A new battery was then used which could account for the drop in energy usage as it might be more efficient at delivering power.

The main area in which energy could be saved would be when going around the bend coming from the rocky mountains toward the hocking hills stop. Although the increase in power to increase separation between the hook of the AEV and the loop of the cart was necessary there is a way in which it would no longer be necessary. If the weighted cart's cargo could configured to better match the specifications of the track that it goes along this maneuver would no longer be necessary and it would save a lot of energy. To do this either the housing of the cargo could be changed to have more length and less height, this would allow for the same volume of cargo to be transported yet get rid of the need for the maneuver. The other way in which this could be accomplished would be in increasing the length of the arm that connects the cart platform to the track. In order for the loop to remain at a constant height relative to the AEV the loop would need to be elevated using a 3-d printed part that would assist in the aerodynamics of the cart. An additional way in which energy could be conserved would be by a change to the track of the monorail, the distance between the incline portion of the track leading to the grand canyon station and the maitenance station location would allow the AEV to experience more frictional forces before it came to the maitenance station. This would decrease the amount of energy the motors use in braking to slow the AEV down. While the energy to get to the grand canyon station would increase coming from the maitenance station, a position could be found in which the energy saved would be the greatest when taking the difference between the energy saved from the braking maneuver and the energy lost using more power to traverse more distance between the grand canyon station and the maitenance station.

The main issue experienced during the weighted cart run was with the inching function. While the approximate power percentage was being discovered to make it up the incline toward the grand canyon station and not be left with too much power, occasionally the AEV would get stuck at the top of the incline. The AEV would then be freed however the inching function was

increasing motor speed while it was stuck, however the motor speed did not change until later. This resulted in the AEV being freed and continuing normally until all of a sudden the motor speed would be increased to about 70 percent. The AEV would then blast forward and run into the cart causing the cart, the AEV or both to derail and cause damages to both themselves and their surroundings. An if statement was implemented in an attempt to prevent this from happening however it was discovered that the if statement was detecting a motor speed greater than what it was supposed to even though the motor speed was not at that value. This resulted in the inching function being broke out of and not moving the AEV into place. Eventually the incher function at the grand canyon station was discarded due to these repeated issues. However other incher functions such as the one at Alaska and the incher function used when dropping the cart off at the grand canyon station were implemented in the code.

Overall the stop on the incline near Alaska was relatively easy to implement. The main problem that existed was moving the WCC from the Alaska stop towards the incline. If the motor speed and subsequently the speed of the WCC was too great then the WCC would surpass the stop on the incline and go all the way to the bottom of the incline where it would then begin to climb the incline. Obviously this wasted energy and was not desired, to account for this 2 major changes were made. First the motor speed that the WCC used to begin with at the Alaska stop was decreased halfway toward the beginning of the incline. This allowed for the WCC to gain some speed and then remain at a lower speed. Secondly the motor speed was briefly increased to a large value to slow the vehicle from its current direction (toward the bottom of the incline) and then after a second the motor speed was decreased to maintain a zero velocity value.

Conclusion and Recommendations:

Through the development of the Advanced Energy Vehicle (AEV) concepts over C++ programming, operational consistency, and operational efficiency were taught. Using concept screening and scoring sheets the less optimal AEV designs were weeded out. This left us with a design that excelled in multiple aspects including affordability, aerodynamics, and weight. This design went on to be known as design 1. When design 1 was compared to design 2 in terms of energy usage, it was more efficient. This led to the decision of design 1 being utilized for the final run. The coding style knowledge obtained from these tests was carried on throughout the rest of the AEV experiments in many ways: motorSpeed was used more than celerate, avoiding reversing and powering motors to break, and slowing down the servo. These coding styles can especially be seen in the final code. Overall, the efficiency of the AEV when compared to other teams was in the top 3 out of approximately 13. This was a huge success considering how important being energy efficient was for the mission.

The total energy usage per kilogram of the WCC was 1346 J/kg (Table #9), however there is no real reference for that amount of energy. So, by converting the total joules to its kilocalorie equivalent it is easier to gain a grasp on that amount of energy- which comes out to be 0.322 kilocalories (kcal) of energy. Since the average person burns between 0.3-0.6 kcal when doing a singular push up, the amount of energy consumed by the AEV is not that much. Especially when considering that an average human consumes around 2,250 kcal a day. This low consumption of energy is very surprising, which just goes to show how inefficient humans are at converting energy.

The main error incurred during the AEV's development was with the inching function. The function created was a while loop which iterated a difference in position over a time and increased or decreased the motor speed accordingly 5 to 10 times a second. The inching function operated under the assumption that all marks that the AEV travelled were accounted for. However, as is evident from the run this is not the case. In Table #9 it is shown in phases 1 and 19 that the goToAbsolutePosition function used to make it up the incline changes from phase 1 to 19. In phase 1 the goToAbsolutePosition function uses a value of 54 inches multiplied by the constant to convert into marks, however in phase 19 that value changes to 66 inches multiplied by the constant to convert into marks. This means that a total of around 12 inches were lost during the run from the wheel skidding and not measuring distance. Skidding for the incher function is extremely dangerous as lost marks result in drastic motor speed changes. Resulting from these high changes in motor speed was almost always the derailing of the AEV and physical damage to components such as the servo and battery carrier. These setbacks delayed progress due to the AEV being inoperative, and delayed progress meant having to rush other tasks.

Some recommendations to avoid these consequences would be to ensure a way to prevent the motor speed from being written to high values. Manufacturing better wheels that gripped to the track and did not slip or skid would ensure less marks lost during the run. An additional improvement would be adding more reflective tape patches, this would allow the reflectance sensors to measure more marks per second and improve the sampling time of the inching function.

Appendix:

Part	# of parts	Price per part	Total cost per part
Arduino	1	100	100
Electric motor	2	9.99	19.98
Servo motor	1	5.95	5.95
Count sensor	2	2	4
Propeller	2	0.45	0.9
Wheels	2	7.5	15
2.5" x 7.5" Rectangle	1	2	2
Trapizoids	2	1	2
T-Shape Arm	1	3	3
Angle Brackets	6	0.84	5.04
Motor clamps	2	0.59	1.18
Bulk screws and nuts	1	2.88	2.88
Battery Carrier	1	.84	.84
			Total 162.77

(Table #9): Price Breakdown for AEV Design 1

Part	# of parts	Price per part	Total cost per part
Arduino	1	100	100
Electric motor	2	9.99	19.98
Servo motor	1	5.95	5.95
Count sensor	2	2	4
Propeller	2	0.45	0.9
Wheels	2	7.5	15
T-Shape	1	2	2
Trapizoids	4	1	4
T-Shape Arm	1	3	3
Angle Brackets	10	0.84	8.4
Motor clamps	2	0.59	1.18
Bulk screws and nuts	1	2.88	2.88
Battery Carrier	1	.84	.84
			total 168.13

(Table #10): Price Breakdown for AEV Design 2

No.	Task	Start	Finish	Due Date	Ben	Matthew	Nick	% complete
1	AEV 1 Construction	2/2/2022	2/2/2022	2/2/2022	x	x	x	100
2	AEV 1 Wind Tunnel Testing	2/9/2022	2/9/2022	2/9/2022	x	x	x	100
3	Wind Tunnel Data Analysis	2/9/2022	2/16/2022	2/16/2022	x	x	x	100
4	Progress Report	2/9/2022	2/16/2022	2/16/2022	x	x	x	100

(Figure #16): System Analysis 1 Schedule

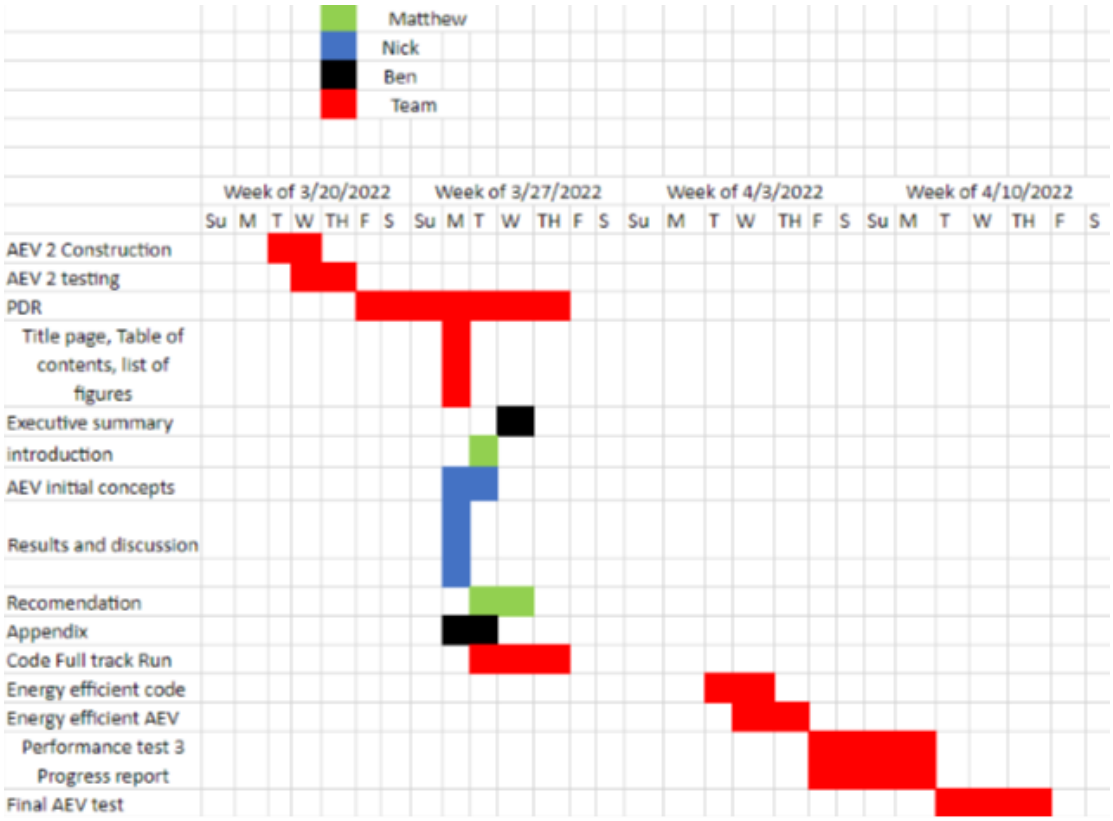
No.	Task	Start	Finish	Due Date	Ben	Matthew	Nick	% complete
1	Code Flat Track Run	2/16/2022	2/23/2022	3/8/2022	x		x	100
2	Matlab Code for Flat Track Run	2/16/2022	2/23/2022	3/8/2022		x		100

3	Code Half Track Run	2/23/2022	3/2/2022	3/8/2022	x		x	100
4	Matlab Code for Half Track Run	2/23/2022	3/2/2022	3/8/2022		x		100
5	Progress Report	2/16/2022	3/8/2022	3/8/2022	x	x	x	100

(Figure #17): System Analysis 2 Schedule

No.	Task	Start	Finish	Due Date	Ben	Matthew	Nick	% complete
1	AEV 2 Construction	3/22/22	3/24/22	3/31/22	x	x	x	100
2	AEV 2 Testing	3/22/22	3/24/22	3/31/22	x	x		100
3	PDR	2/16/22	3/31/22	3/31/22	x	x	x	100
3a	Title page, Table of contents, & List of Figures	3/28/22	3/28/22	3/31/22	x	x	x	100
3b	Executive Summary	3/29/22	3/30/22	3/31/22	x			100
3c	Introduction	3/28/22	3/28/22	3/31/22		x		100
3d	AEV Initial Concepts	3/28/22	3/29/22	3/31/22		x		100
3e	Results and Discussion	3/28/22	3/28/22	3/31/22			x	100
3f	Conclusion and Recomendation	3/29/22	3/30/22	3/31/22		x		100
3g	Appendix	3/21/22	3/31/22	3/31/22	x			100

(Figure #18): Performance Test 1 Schedule



(Figure #19): Daily Schedule for Performances 1,2,3, and 4

Division of work statement

Ben Bazan completed Appendix and Executive Summary components. Matthew Geiger completed the Introduction, AEV initial Concepts, Results and Discussion: Weighted and Unweighted carts, and the Conclusion and Recommendations. Nick Stassen completed the Results and Discussion.

Unweighted cart run Arduino Code:

```
rotateServo(60);
  reverse(4);
  motorSpeed(4,40);
  goToAbsolutePosition(55*(8/3.902));
  int i=10;
  motorSpeed(4,i);
  while(getVehiclePosition()<(77*(8/3.902))){
    float current = getVehiclePosition();
    delay(200);
    float next = getVehiclePosition();
    if(abs(next-current)/0.2<2){
      i++;
      motorSpeed(4,i);
    }
    if((abs(next-current)/0.2)>2){
      i--;
      motorSpeed(4,i);
    }
  }
  brake(4);
  goFor(5);
  for(int i=60; i>=0; i--)
  {                                 // allows the servo to rotate slowly rather than quickly
    rotateServo(i);
    delay(20);
  }
  goFor(4);                         // waiting for humans to connect the cart to the AEV if self-connect fails.
  reverse(4);
  motorSpeed(4,24);
  goToAbsolutePosition(24*(8/3.902)); // traveling towards the now decline from the grandcanyon
  motorSpeed(4,15);
  goToAbsolutePosition(-84*(8/3.902));
  motorSpeed(4,20);
  goToAbsolutePosition(-140*(8/3.902)); // increasing speed to make it to the waves
  brake(4);
  goFor(5);
  motorSpeed(4,40); //increasing power to go up the incline towards alaska
  goToAbsolutePosition(-244*(8/3.902));
  i=12;
  motorSpeed(4,i);
  while(getVehiclePosition()>(-260*(8/3.902))){ //same incher code to make it to alaska
    float current = getVehiclePosition();
    delay(200);
    float next = getVehiclePosition();
    if(abs(next-current)/0.2<2){
      i++;
      motorSpeed(4,i);
    }
    if((abs(next-current)/0.2)>2){
      i--;
      motorSpeed(4,i);
    }
  }
  brake(4);
```

```
goFor(7);
reverse(4);
motorSpeed(4,30);
goToAbsolutePosition(-256*(8/3.902));
brake(4);
reverse(4);
motorSpeed(4,36);  //stopping on the incline portion by alaska
goFor(5);
brake(4);
reverse(4);
motorSpeed(4,24);
goToAbsolutePosition(-210*(8/3.902)); //coasting into the next stop
brake(4);
goFor(5);
motorSpeed(4,37);
goToAbsolutePosition(-50*(8/3.902)); //going towards the hocking hills stop
brake(4);
i=10;
motorSpeed(4,i);
while(getVehiclePosition()>(6*(8/3.902))){  //using incher to stop at hocking hills
  float current = getVehiclePosition();
  delay(200);
  float next = getVehiclePosition();
  if(abs(next-current)/0.2<2){
    i++;
    motorSpeed(4,i);
  }
  if((abs(next-current)/0.2)>2){
    i--;
    motorSpeed(4,i);
  }
}
brake(4);
goFor(5);
motorSpeed(4,43);   //powering up motors to make it up the incline towards the grand canyon
goToAbsolutePosition(63*(8/3.902));
i=10;
motorSpeed(4,i);
while(getVehiclePosition()<(74*(8/3.902))){ //same incher code
  float current = getVehiclePosition();
  delay(200);
  float next = getVehiclePosition();
  if(abs(next-current)/0.2<2){
    i++;
    motorSpeed(4,i);
  }
  if((abs(next-current)/0.2)>2){
    i--;
    motorSpeed(4,i);
  }
}
brake(4);
goFor(5);
for(int i2=0; i2<=70; i2++) { //same slow servo rotation code
  rotateServo(i2);
  delay(20);
```

```
  }

  reverse(4);
  motorSpeed(4,20);
  goToAbsolutePosition(60*(8/3.902));
  reverse(4);
  i=20;
  motorSpeed(4,i);
  while(getVehiclePosition()>(18*(8/3.902))){  //a modification to the incher code that allows it slow itself
down while traversing the incline portion of the incline
    float current = getVehiclePosition();
    delay(100);
    float next = getVehiclePosition();
    if(abs(next-current)/0.1>2){
      i++;
      motorSpeed(4,i);
    }
    if((abs(next-current)/0.1)<2){
      i--;
      motorSpeed(4,i);
    }
  }
  brake(4);  //stopping the motors and stopping at the maitenance station.
```

Weighted cart run Arduino Code:

```
rotateServo(60);
  reverse(4);
  motorSpeed(4,40);
  goToAbsolutePosition(54*(8/3.902));
  brake(4);
  goFor(5);
  for(int i=70; i>=0; i--)
  {                                 // allows the servo to rotate slowly rather than quickly
    rotateServo(i);
    delay(20);
  }
  goFor(4);                         // waiting for humans to connect the cart to the AEV if self-connect fails.
  reverse(4);
  motorSpeed(4,26);
  goToAbsolutePosition(48*(8/3.902)); // traveling towards the now decline from the grandcanyon
  brake(4);
  goFor(8);
  motorSpeed(4,29);
  goToAbsolutePosition(-142*(8/3.902)); // increasing speed to make it to the waves
  brake(4);
  goFor(5);
  motorSpeed(4,44); //increasing power to go up the incline towards alaska
  goToAbsolutePosition(-256*(8/3.902));
  int i=10;
  motorSpeed(4,i);
  while(abs(getVehiclePosition())<(267*(8/3.902))){ //same incher code to make it to alaska
    float current = getVehiclePosition();
    delay(200);
    float next = getVehiclePosition();
```

```
  if(abs(next-current)/0.2<2){
    i++;
    motorSpeed(4,i);
  }
  if((abs(next-current)/0.2)>2){
    i--;
    motorSpeed(4,i);
  }
}
brake(4);
goFor(4);
reverse(4);
motorSpeed(4,33);
goToAbsolutePosition(-268*(8/3.902));
motorSpeed(4,30);
goToAbsolutePosition(-258*(8/3.902));
brake(4);
reverse(4);
motorSpeed(4,52);  //stopping on the incline portion by alaska
goFor(1);
motorSpeed(4,36);
goFor(3);
brake(4);
reverse(4);
motorSpeed(4,26);
goToAbsolutePosition(-202*(8/3.902)); //coasting into the next stop
brake(4);
goFor(5);
motorSpeed(4,35);
goToAbsolutePosition(-120*(8/3.902));
motorSpeed(4,44);
goToAbsolutePosition(-96*(8/3.902));
brake(4);
delay(750);
motorSpeed(4,65);
delay(350);
motorSpeed(4,33);
goToAbsolutePosition(-40*(8/3.902)); //going towards the hocking hills stop
brake(4);
i=10;
motorSpeed(4,i);
while(getVehiclePosition()>(-12*(8/3.902))){  //using incher to stop at hocking hills
  float current = getVehiclePosition();
  delay(200);
  float next = getVehiclePosition();
  if(abs(next-current)/0.2<2){
    i++;
    motorSpeed(4,i);
  }
  if((abs(next-current)/0.2)>2){
    i--;
    motorSpeed(4,i);
  }
}
brake(4);
goFor(5);
```

```
  motorSpeed(4,44);   //powering up motors to make it up the incline towards the grand canyon
  goToAbsolutePosition(66*(8/3.902));
  i=10;
  motorSpeed(4,i);
  while(getVehiclePosition()<(79*(8/3.902))){ //same incher code
    float current = getVehiclePosition();
    delay(200);
    float next = getVehiclePosition();
    if(abs(next-current)/0.2<2){
      i++;
      motorSpeed(4,i);
    }
    if((abs(next-current)/0.2)>2){
      i--;
      motorSpeed(4,i);
    }
  }
  brake(4);
  goFor(5);
  for(int i2=0; i2<=70; i2++) { //same slow servo rotation code
    rotateServo(i2);
    delay(20);
  }
  goFor(1);
  reverse(4);
  motorSpeed(4,22);
  goToAbsolutePosition(55*(8/3.902));
  reverse(4);
  i=25;
  motorSpeed(4,i);
  while(getVehiclePosition()>(6*(8/3.902))){  //a modification to the incher code that allows it slow itself
down while traversing the incline portion of the incline
    float current = getVehiclePosition();
    delay(100);
    float next = getVehiclePosition();
    if(abs(next-current)/0.1>2){
      i++;
      motorSpeed(4,i);
    }
    if((abs(next-current)/0.1)<2){
      i--;
      motorSpeed(4,i);
    }
  }
  brake(4);
  reverse(4);//stopping the motors and stopping at the maitenance station.
  i=10;
  motorSpeed(4,i);
  while(getVehiclePosition()>(5*(8/3.902))){  //  trying to stop backwards
    float current = getVehiclePosition();
    delay(100);
    float next = getVehiclePosition();
    if(abs(next-current)/0.1>2){
      i--;
      motorSpeed(4,i);
    }
```

```
  if((abs(next-current)/0.1)<2){
    i++;
    motorSpeed(4,i);
  }
 }
 brake(4);
```

MATLAB Code:

```
%==========================================================================
% Name: Matthew Geiger
% Date: April 20th 2022
% Class: 3:05 pm
%
% Last edited: 4/20/2022
%
%
%
% Program Title: Final Test Analysis
%
% Program Description: This program's purpose is to convert the
% EEPROMM data into usable values (inches velocity time(s)) as well as
% to determine the energy used.
%
%
%==========================================================================


clear;
clc;
data=xlsread("Final_Test_m");
tA=data(:,1);
iA=data(:,2);
vA=data(:,3);
mCum=data(:,4);
mPos=data(:,5);
D=0.0762; %Diameter of propeller
m=input('Please enter the weight of the AEV in kilograms: ');

%With the data now in variablwes we can calculate the variables we
want

T=(tA./1000); %Time in seconds
I=(iA./1024*2.46*(1/0.185)); %current in amps
V=((vA./1024)*15); %voltage in volts
Pos=(0.0124.*mCum); %distance traveled in meters
RelPos=(0.0124.*mPos); %relative positon in meters
SupP=(I.*V); %power in watts
for i=1:(length(SupP)-1)
```

```
    IncE(i)=(((SupP(i)+SupP(i+1))/2)*(T(i+1)-T(i))); %Incremental
energy in joules
end
E=sum(IncE);% Total energy in joules
TotE=E/m; %Total energy in joules per kilogram

for i2=2:length(T)
    vel(i2)=((RelPos(i2)-RelPos(i2-1))/(T(i2)-T(i2-1)));
end
KE=(1/2*m.*(vel.^2)); %calculates instantaneous kenetic energy
PropRPM=((-64.59.*(I.^2)+1927.25.*I-84.58)*0.8); % equation for
finding rpm of propeller based off current
J=(vel./((PropRPM'/60)*D));
for i3=1:length(J)
    if J(i3)<=0.15 && SupP(i3)==0
        J(i3)=0;
    elseif J(i3)<=0.15 && SupP(i3)>0
        J(i3)=0.15;
    end
end

PropEff=abs(-454.37.*(J.^3)+321.58.*(J.^2)+22.603.*J);
aa=find(PropEff>100);
for ac=1:792
    while PropEff(ac)>100
        PropEff(ac)=(PropEff(ac)-100);
    end
end

plot(T,SupP);
grid on
xlabel("Time (seconds)");
ylabel("Supplied Power (Watts)");
title("Supplied Power (W) vs. Time (s)");

ab=ginput(23);

figure;
plot(Pos,SupP);
grid on
xlabel("Position (meters)");
ylabel("Supplied Power (Watts)");
title("Supplied Power (W) vs. Position (m)");

figure;
plot(Pos,vel); %plots the velocity vs position of the AEV
grid on
xlabel("Position (meters)");
ylabel("Velocity (m/s)");
title("Velocity (m/s) vs. Positon (m)");

figure;
```

```
plot(Pos,KE);
grid on
xlabel("Position (meters)");
ylabel("Kinetic Energy (Joules)");
title("Kinetic Energy (J) vs. Position (m)");

figure;
plot(J,PropEff,'v');
grid on
xlabel("Advance Ratio");
ylabel("Propulsion Efficiency (%) ");
title("Propulsion Efficiency (%) vs. Advance Ratio");


%Phase 1 energy
xR1=ab(1);
xL1=0;
Ep1t=(xR1-xL1);
iL1=knnsearch(T,xL1);
iR1=knnsearch(T,xR1);
Ep1=IncE(iL1:iR1);
Ep1tot=sum(Ep1);
Ep1to=Ep1tot/m;

%Phase 2 energy
xR2=ab(2);
xL2=xR1;
Ep2t=(xR2-xL2);
iL2=knnsearch(T,xL2);
iR2=knnsearch(T,xR2);
Ep2=IncE(iL2:iR2);
Ep2tot=sum(Ep2);
Ep2to=Ep2tot/m;

%Phase 3 energy
xR3=ab(3);
xL3=xR2;
Ep3t=(xR3-xL3);
iL3=knnsearch(T,xL3);
iR3=knnsearch(T,xR3);
Ep3=IncE(iL3:iR3);
Ep3tot=sum(Ep3);
Ep3to=Ep3tot/m;

%Phase 4 energy
xR4=ab(4);
xL4=xR3;
Ep4t=(xR4-xL4);
iL4=knnsearch(T,xL4);
iR4=knnsearch(T,xR4);
Ep4=IncE(iL4:iR4);
Ep4tot=sum(Ep4);
```

```
Ep4to=Ep4tot/m;

%Phase 5 energy
xR5=ab(5);
xL5=xR4;
Ep5t=(xR5-xL5);
iL5=knnsearch(T,xL5);
iR5=knnsearch(T,xR5);
Ep5=IncE(iL5:iR5);
Ep5tot=sum(Ep5);
Ep5to=Ep5tot/m;

%Phase 6 energy
xR6=ab(6);
xL6=xR5;
Ep6t=(xR6-xL6);
iL6=knnsearch(T,xL6);
iR6=knnsearch(T,xR6);
Ep6=IncE(iL6:iR6);
Ep6tot=sum(Ep6);
Ep6to=Ep6tot/m;

%Phase 7 energy
xR7=ab(7);
xL7=xR6;
Ep7t=(xR7-xL7);
iL7=knnsearch(T,xL7);
iR7=knnsearch(T,xR7);
Ep7=IncE(iL7:iR7);
Ep7tot=sum(Ep7);
Ep7to=Ep7tot/m;

%Phase 8 energy
xR8=ab(8);
xL8=xR7;
Ep8t=(xR8-xL8);
iL8=knnsearch(T,xL8);
iR8=knnsearch(T,xR8);
Ep8=IncE(iL8:iR8);
Ep8tot=sum(Ep8);
Ep8to=Ep8tot/m;

%Phase 9 energy
xR9=ab(9);
xL9=xR8;
Ep9t=(xR9-xL9);
iL9=knnsearch(T,xL9);
iR9=knnsearch(T,xR9);
Ep9=IncE(iL9:iR9);
Ep9tot=sum(Ep9);
Ep9to=Ep9tot/m;
```

```
%Phase 10 energy
xR10=ab(10);
xL10=xR9;
Ep10t=(xR10-xL10);
iL10=knnsearch(T,xL10);
iR10=knnsearch(T,xR10);
Ep10=IncE(iL10:iR10);
Ep10tot=sum(Ep10);
Ep10to=Ep10tot/m;

%Phase 11 energy
xR11=ab(11);
xL11=xR10;
Ep11t=(xR11-xL11);
iL11=knnsearch(T,xL11);
iR11=knnsearch(T,xR11);
Ep11=IncE(iL11:iR11);
Ep11tot=sum(Ep11);
Ep11to=Ep11tot/m;

%phase 12 energy
xR12=ab(12);
xL12=xR11;
Ep12t=(xR12-xL12);
iL12=knnsearch(T,xL12);
iR12=knnsearch(T,xR12);
Ep12=IncE(iL12:iR12);
Ep12tot=sum(Ep12);
Ep12to=Ep12tot/m;



%Phase 13 energy
xR13=ab(13);
xL13=xR12;
Ep13t=(xR13-xL13);
iL13=knnsearch(T,xL13);
iR13=knnsearch(T,xR13);
Ep13=IncE(iL13:iR13);
Ep13tot=sum(Ep13);
Ep13to=Ep13tot/m;


%Phase 14 energy
xR14=ab(14);
xL14=xR13;
Ep14t=(xR14-xL14);
iL14=knnsearch(T,xL14);
iR14=knnsearch(T,xR14);
Ep14=IncE(iL14:iR14);
Ep14tot=sum(Ep14);
Ep14to=Ep14tot/m;
```

```
%Phase 15 energy
xR15=ab(15);
xL15=xR14;
Ep15t=(xR15-xL15);
iL15=knnsearch(T,xL15);
iR15=knnsearch(T,xR15);
Ep15=IncE(iL15:iR15);
Ep15tot=sum(Ep15);
Ep15to=Ep15tot/m;

%Phase 16 energy
xR16=ab(16);
xL16=xR15;
Ep16t=(xR16-xL16);
iL16=knnsearch(T,xL16);
iR16=knnsearch(T,xR16);
Ep16=IncE(iL16:iR16);
Ep16tot=sum(Ep16);
Ep16to=Ep16tot/m;


%Phase 17 energy
xR17=ab(17);
xL17=xR16;
Ep17t=(xR17-xL17);
iL17=knnsearch(T,xL17);
iR17=knnsearch(T,xR17);
Ep17=IncE(iL17:iR17);
Ep17tot=sum(Ep17);
Ep17to=Ep17tot/m;

%Phase 18 energy
xR18=ab(18);
xL18=xR17;
Ep18t=(xR18-xL18);
iL18=knnsearch(T,xL18);
iR18=knnsearch(T,xR18);
Ep18=IncE(iL18:iR18);
Ep18tot=sum(Ep18);
Ep18to=Ep18tot/m;

%Phase 19 energy
xR19=ab(19);
xL19=xR18;
Ep19t=(xR19-xL19);
iL19=knnsearch(T,xL19);
iR19=knnsearch(T,xR19);
Ep19=IncE(iL19:iR19);
Ep19tot=sum(Ep19);
Ep19to=Ep19tot/m;
```

```
%Phase 20 energy
xR20=ab(20);
xL20=xR19;
Ep20t=(xR20-xL20);
iL20=knnsearch(T,xL20);
iR20=knnsearch(T,xR20);
Ep20=IncE(iL20:iR20);
Ep20tot=sum(Ep20);
Ep20to=Ep20tot/m;

%Phase 21 energy
xR21=ab(21);
xL21=xR20;
Ep21t=(xR21-xL21);
iL21=knnsearch(T,xL21);
iR21=knnsearch(T,xR21);
Ep21=IncE(iL21:iR21);
Ep21tot=sum(Ep21);
Ep21to=Ep21tot/m;

%Phase 22 energy
xR22=ab(22);
xL22=xR21;
Ep22t=(xR22-xL22);
iL22=knnsearch(T,xL22);
iR22=knnsearch(T,xR22);
Ep22=IncE(iL22:iR22);
Ep22tot=sum(Ep22);
Ep22to=Ep22tot/m;

%Phase 23 energy
xR23=ab(23);
xL23=xR22;
Ep23t=(xR23-xL23);
iL23=knnsearch(T,xL23);
iR23=knnsearch(T,xR23);
Ep23=IncE(iL23:iR23);
Ep23tot=sum(Ep23);
Ep23to=Ep23tot/m;


fprintf("Phase 1 energy total %f, time %f \n",Ep1tot, Ep1t);
fprintf("Phase 2 energy total %f, time %f \n",Ep2tot, Ep2t);
fprintf("Phase 3 energy total %f, time %f \n",Ep3tot, Ep3t);
fprintf("phase 4 energy total %f, time %f \n",Ep4tot, Ep4t);
fprintf("Phase 5 energy total %f, time %f \n",Ep5tot, Ep5t);
fprintf("Phase 6 energy total %f, time %f \n",Ep6tot, Ep6t);
fprintf("Phase 7 energy total %f, time %f \n",Ep7tot, Ep7t);
fprintf("Phase 8 energy total %f, time %f \n",Ep8tot, Ep8t);
fprintf("Phase 9 energy total %f, time %f \n",Ep9tot, Ep9t);
fprintf("Phase 10 energy total %f, time %f \n",Ep10tot, Ep10t);
fprintf("Phase 11 energy total %f, time %f \n",Ep11tot, Ep11t);
```

```
fprintf("Phase 12 energy total %f, time %f \n",Ep12tot, Ep12t);
fprintf("Phase 13 energy total %f, time %f \n",Ep13tot, Ep13t);
fprintf("Phase 14 energy total %f, time %f \n",Ep14tot, Ep14t);
fprintf("Phase 15 energy total %f, time %f \n",Ep15tot, Ep15t);
fprintf("Phase 16 energy total %f, time %f \n",Ep16tot, Ep16t);
fprintf("Phase 17 energy total %f, time %f \n",Ep17tot, Ep17t);
fprintf("Phase 18 energy total %f, time %f \n",Ep18tot, Ep18t);
fprintf("Phase 19 energy total %f, time %f \n",Ep19tot, Ep19t);
fprintf("Phase 20 energy total %f, time %f \n",Ep20tot, Ep20t);
fprintf("Phase 21 energy total %f, time %f \n",Ep21tot, Ep21t);
fprintf("Phase 22 energy total %f, time %f \n",Ep22tot, Ep22t);
fprintf("Phase 23 energy total %f, time %f \n",Ep23tot, Ep23t);



fprintf("total energy %f \n", E);
fprintf("total energy per kilogram: %f \n",TotE);


figure;
plot(T,Pos);
title("Position vs. time");
xlabel("Time (s)");
ylabel("Position (m)");
```