Memo

Date: November 13, 2015
To: Inst. Schrock and GTA Yang
From: Group A - Nick Bova, Bryan Check, Tyler Sargent, Jordan Scully, Brad Sievers
Subject: Lab 09 Performance Test 2 - Code

Introduction

The purpose of this lab was for the team to modify the code executed by the Arduino in the AEV in order to successfully fulfill the objectives laid out in the Mission Concept Review (MCR). Based on the design that was decided upon in Lab 08 - Performance Test 1, the code was required to guide the vehicle to the gate and stop for seven seconds to allow the gate to open. From there, the AEV was supposed to travel to the cargo and slow down to pick it up. After five seconds, the vehicle traveled back towards the gate, stopped again for seven seconds, and then returned to the starting point. The goal was to develop two functioning codes during lab time so that the AEV could be tested using both and the efficiency of each run (and in turn each code) could be compared and analyzed. This memo was created in order to outline the thought process of the team while conducting the performance test and examining the data afterwards. The results are displayed and discussed below as well as the individual conclusions and recommendations drawn by each group member.

Results & Discussion

The main strategy the team used for the first Arduino code originally involved having the AEV utilize coasting. The team reasoned that this would help the AEV be more energy efficient, for gliding along portions of the track would require no power from the motors and therefore less overall energy would be exerted. However, in practice this method proved to be too inconsistent. It was far too difficult to get the vehicle to coast to a stop at the gate open sensor while also making sure that it did not trip the sensor that initiated the gate's closing. Slight variations in battery placement on the body of the AEV, bumps in the track, and power lossage in the battery after each run forced the team to find a more consistent, practical code. The group brainstormed and decided instead to utilize the "reverse()" and "celerate()" commands in conjunction to stop the AEV more abruptly. As it reached the first gate sensor, the propellers quickly rotated in the opposite direction of the AEV's movement generating negative thrust and halting the vehicle with improved accuracy. The general outline of the run can be seen in Table 1 below.

Table 1: Arduino Code Basic Algorithm

Stage	Distance	Wait Time
Travel to Gate	18.3 ft	-
Wait for Gate	-	8 sec
Travel to Cargo	20.3 ft	-
Collect Cargo and Wait	-	7 sec
Travel to Gate	18.3 ft	-
Wait for Gate	-	8 sec
Travel to End	20.3 ft	-

With the tracks being different enough to cause inconsistencies with different test runs and the batteries also being inconsistent, it was decided that it would be easier to adjust this code in the future rather than the initial coasting-dependent code described above. The team first got a rough outline of the full program and then continually tweaked the mark values used in "motorSpeed()" and "celerate()" as well as the times used in "goFor()" until the AEV performed as desired (Figure A1). This was achieved through dozens of test runs and keen observations from the group to see how far the AEV stopped from where it was supposed to ultimately be stopping. The vehicle ran smoothly during testing, exhibiting superb center of balance and an optimal performance speed. No physical characteristics of the AEV behaved differently than expected by the team.

Unfortunately, the team was only able to develop one successful code during the lab instead of two. This was partly due to the fact that the team spent so much of their time developing a coast-heavy code which ultimately proved ineffective (as discussed above). Also, for the entirety of one of the work days the group's reflectance sensors were malfunctioning. This issue was not realized until the end of class, and therefore all of the day's progress had to be scrapped and the sensors recalibrated. Ultimately, the team did not plan their time as efficiently as possible which resulted in this task not being completed. Two codes will be developed in Performance Test 3 so that the group does not fall behind.

After the AEV was tested with the final code, the data collected by its reflectance sensors was uploaded to MATLAB and the AEV Analysis Tool was used to convert the values to physical parameters (just like in Performance Test 1). A graph of power exerted vs time was generated by the software, as seen in Figure 1 below. Similar to the procedure in System Analysis 2, the group then examined the graph and divided it into twelve phases based on the Arduino commands utilized at these points.

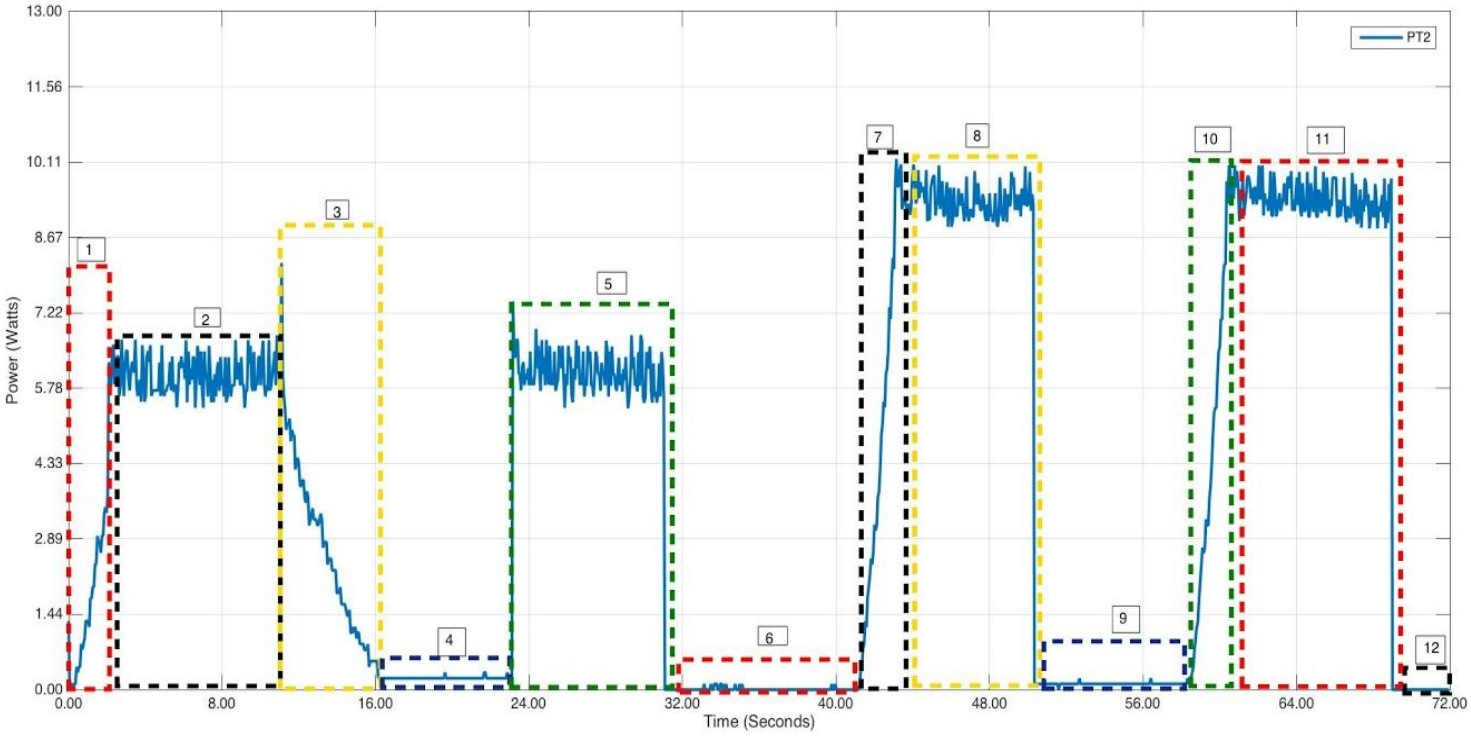

Figure 1: Graph of AEV Power vs. Time with Phase Divisions

As one can see, the graph of the run looks as though it follows a rough pattern. The AEV accelerates, moves at a constant speed, and then decelerates before braking. This same basic movement is seen four times in the above graph, for the "celerate()", "motorSpeed()", "celerate()", "brake()" command combination was used multiple times within the code for symmetry and ease of duplication. According to the chart, the entire run took about 70 seconds to complete which was very close the group's estimation of 65 seconds. To further examine the different phases of the AEV's run and the energy exertion during each, the elapsed time and and amount of energy used in joules was calculated for each command. This data was gathered from the "Performance Analysis" tool of the AEV Analysis application and organized in the table below for clarity.

Table 2: Phase Energy Breakdown Data

Phase	Arduino Code	Time (s)	Total Energy (J)
1	celerate(4,0,23,2)	2	3.1178
2	motorSpeed(4,23)	5.52	32.9382
3	celerate(4,23,0,5)	5	28.4128
4	brake(4)	7	7.21
5	motorSpeed(4,23)	11.5	27.1768
6	brake(4)	10	0.0921
7	celerate(4,0,35,2)	2	6.194
8	motorSpeed(4,35)	7.4	68.4441
9	brake(4)	8	0.8979
10	celerate(4,0,35,2)	2	5.1353
11	motorSpeed(4,35)	9.06	83.9184
12	brake(4)	2.4	0
		Total Energy:	285.6431 J

Judging by the data in the table, the “celerate()” and “motorSpeed” commands performed with very similar efficiencies. During phase 2, the AEV was moved using the “motorSpeed()” command for 5.52 seconds and exerted about 34 J of energy during this time. The “celerate()” command in phase 3 exerted only 28 J of energy, however the vehicle ran for a half second less under this command- insinuating that the efficiencies are nearly equal. Another phenomenon that can be discerned from this table is that the AEV exerted much more energy when carrying the cargo than it did on the trip there. To illustrate this concept, phases 5 and 11 can be examined. Although the “motorSpeed()” command was issued for 1.5 seconds longer in phase 5 than in face 11, only 28 J were used when no R2-D2 was present and 84 J were used when he was. This was expected by the team, for an increase in mass requires an increase in thrust needed to move the AEV which in turn increases the energy consumed. To combat this effect, the team decided to have the AEV use the pull propeller configuration when hauling the cargo because it is more efficient. It was the group’s hope that this increase in efficiency could aid in offsetting the additional energy needed to pull the extra mass. One last interesting piece of information gained from the above table is the inconsistency of the “brake()” function’s power usage. In theory, the AEV should exert 0 J of energy while this command is being run because the command’s function is to cut all power to the motors. However, according to the phase breakdown the energy usage of the command fluctuates between 0 and 7 J throughout the run. This occurrence must be due to a sensor error of some type, because no matter how fast the AEV is traveling or how much mass is attached to the body, no energy should be used. The command does not actively halt the propellers from rotating, it simply stops the motors and lets the vehicle coast. The entire run used a total of 286 J of energy, which seemed reasonable to the group.

Aside from the reflectance sensor issue described above as well as inconsistencies between the two test tracks, another error present in the lab was that the AEV was slow to start when the battery was running low and therefore it required a push to get going- altering the intended speed of the vehicle. Otherwise, the AEV ran at a consistent speed though distance traveled also fluctuated for some runs. Another error that arose during testing was that the AEV initially had trouble connecting to the payload, as the metal bracket was too low to properly latch onto the R2-D2 magnet. One final potential error present during this procedure could have occurred between the time the AEV was removed from the track and the data was uploaded to MATLAB for analysis. If the wheels of the AEV were accidentally touched or spun by a group member during this time, the EEPROM data recorded by the vehicle would be thrown off and the results would be tainted.

Conclusions and Recommendations- Bryan

In this lab, an Arduino code was devised by the team and implemented so that the AEV could complete all operational requirements listed in the Mission Concept Review (MCR). The chosen AEV design from the previous lab was commanded to travel to the gate, pick up the R2-D2 cargo, return to the gate, and then travel back to the starting point. The purpose of this lab was to create two codes that accomplished these tasks and to compare the consistency of the runs to improve analysis and prototype testing skills. However, the team did not have time to finish a second code and therefore the contents of this memo focus on analyzing the run of the one successfully-coded AEV. It was found that the "celerate()" and "motorSpeed()" commands were quite similar in energy expenditure and that the AEV used more energy when hauling the cargo for obvious reasons. It was also found that relying on the AEV coasting is too inconsistent to be viable, and therefore a code that utilizes the "reverse()" and "celerate()" command to bring the vehicle to an abrupt halt is advisable.

Much of the error present in this lab can be resolved. To solve the problem of track inconsistencies between the overhead railings on the second and third floors of the building, the group can simply make sure that they adjust their mark values to account for the proper track before the final run. To ensure that no issues arise with regards to low battery power during testing, the team can acquire a fresh battery before attempting to complete the final mission. This would also solve the problem that occurred during some runs where the AEV needed a slight push to get started. In order to fix the error where the AEV failed to attach to the cargo, the team should extend the metal bracket further from the vehicle and attach it to a lower portion of the vehicle. The final potential source of error can easily be prevented. The team members must simply be conscientious of the wheel sensor and ensure that they do not move it in between the AEV's run and the data upload.

All in all, although the team did not complete a second code as intended this lab provided valuable information that will be used to optimize the completion of the mission. The procedure led the team to conclude that coasting should be scrapped entirely from their code, and also that the "celerate()" and "motorSpeed()" commands can be used interchangeably for the most part. As already predicted, the AEV exerts more energy when carrying the cargo and therefore it is recommended that the AEV is run in the pull configuration on the return trip. More efficiency testing will be conducted in future labs, however at this stage it is recommended that the team merely tweak the code devised in this lab to complete the remaining tasks.

Conclusions and Recommendations - Jordan

The experiment was performed by building the AEV design chosen during the last performance test and writing code to do the required tasks. This meant making the AEV stopping at the gate, picking up R2D2, stopping at the gate again and then returning to the start. The code the group ended up creating was one that used a reverse thrust to slow the vehicle. This resulted in a code that used 285J to complete the task. The main result of this lab was the group not using the coasting idea due to wild inaccuracy. This meant the team needed to change its plan of coding to one relying more on the got to absolute value command for the entire distance with no coasting.

During the course of this lab the team had to overcome many obstacles resulting in the code being reworked several times. This led the team to being behind schedule and rethinking the approach towards completion. A source of error was the AEV not reading marks correctly, the reason this happened was most likely poorly charged batteries. Another result of error was the different tracks causing variance in AEV runs. These problems were overcome by the end of the lab however through rewriting the code and making sure to grab a working battery. The initial idea of coasting to use as little energy as possible had to be scrapped due to the inaccuracies.

Conclusions and Recommendations - Tyler

The primary objective of the lab was to create and test two different codes for a singular AEV design. In this lab a series of problems prevented the second code from be completed and tested to the full extent. Some of the sources of error for the various problems that were faced were the varying battery level over time as well as differing batteries. Others included the differences between the tracks in either lab room. While troubleshooting the code and testing the AEV, the battery was consumed at a surprising rate making each run progressively more hard to predict the distances for the travel functions. This led to very inconsistent runs, many resulting in full failure. Once the battery was swapped it became apparent the the two tracks had some level of variation that was throwing off the calculations for the AEV code. A recommended fix for this is to minimize AEV runs and frequently exchange for fresh batteries. Also using functions that are based off of distance traveled rather than time pass allows for far more precise and controlled runs. The total power used by the AEV code can be found in Figure 1. This code that utilizes brakes for coasting along the track to minimize energy from the reversing of the motors and breaking in that fashion, uses a total of 285J to complete. As mentioned earlier there is no other code to compare this to however using an alternate code that uses power to stop the AEV would logically use more power therefore it is possible to conclude off that logic that coasting is more efficient when time is not important.

Conclusions and Recommendations - Brad

The purpose of this lab was to develop two Arduino codes to test, and determine which code completed the tasks required while using as little energy as possible. Many problems were encountered during this lab that the team had to overcome. Due to this, the team only had time to develop one Arduino code to test. After originally designing an Arduino code using coasting to save energy, the team found that the two tracks affect the AEV differently. Also, the batteries can be unreliable. As a result, it is recommended that an Arduino code using more precise functions, such as goToAbsolutePosition() and brake(), be used in order to better control the AEV. This will allow more precise distances to be travelled, regardless of the track conditions. Also, any changes that need to be made to the code in the future can be more easily done. Figure 1 above shows exactly how much energy is being used by the AEV during each phase. There are four general areas that use a lot of energy, comparatively. During the next lab, the team will need to determine how to make this code as energy efficient as possible while simultaneously maintaining simplicity.

Conclusions and Recommendations - Nick

It’s recommended, after what had happened in this lab, that care is made to make sure each part is maintained as well as possible. Any changes or imperfections in the parts, from the propellers and motors to the Arduino itself, will cause the AEV to run differently than from what is initially intended. In relation to the code, it is important as it is time consuming and inefficient to keep altering values in the code to accommodate these factors when other aspects of the vehicle need to be taken into account (e.g. energy efficiency, final testing).

Appendix A

```
reverse(4);                         //Reverse all motors
celerate(4,0,23,2);                         //Accelerate all motors from 0 to 23% power in 2 sec

motorSpeed(4,23);                 //Run all motors at 23% power.
goToAbsolutePosition(-362);       //Go to an absolute position of -362 marks.

reverse(4);                       //Reverse all motors.
celerate(4,23,0,5);                 //Accelerate all motors from 23 to 0% power in 5 sec

brake(4);                           //Brake all motors
goFor(7);                           //Go for 7 seconds
                                    //Stop at gate

reverse(4);                         //Reverse all motors
motorSpeed(4,23);                 //Run all motors at 23 % power
goToRelativePosition(-338);       //Go to a relative position of of -338 marks

brake(4);                             //Brake all motors
goFor(10);                          //Go for 10 seconds
                                    //Pick-up Cargo

reverse(4);                             //Reverse all motors.
celerate(4,0,35,2);               //Accelerate all motors from 0 to 35%  power in 2 sec

motorSpeed(4,35);                 //Run all motors at 35% power
goToRelativePosition(355);        //Go to a relative position of 355.

brake(4);                             //Brake all motors
goFor(8);                           //Go for 8 seconds
                                    //Stop at gate

celerate(4,0,35,2);               //Accelerate all motors from 0 to 35% power in 2 sec

motorSpeed(4,35);                 //Run all motors at 35% power
goToRelativePosition(355);        //Go to a relative position of 355 marks

brake(4);                           //Brake all motors
                                    //Run Finished
```

Figure A1: Arduino Code Used to Complete Mission

References

1. “Advance Energy Vehicle Design Project Lab Manual” https://eeiccourses.engineering.osu.edu/sites/eeiccourses.engineering.osu.edu/files/uploads/1182/AEVLab/AEVDocuments/LabManual/AEV_Lab_Manual_Rev_2015_08_07.pdf

2. “Technical Communication Guide” https://eeiccourses.engineering.osu.edu/sites/eeiccourses.engineering.osu.edu/files/uploads/resources/TechCommGuide/Tech_Comm_Guide_Rev_2015_07_16.pdf