Arduino Coding AEV Executive Summary

Submitted to
Dr. Janiszewska

Prepared by

Nick Stassen,
Matthew Geiger,
Ben Bazan

Engineering 1182
The Ohio State University
College Of Engineering
Newark, OH

January 27, 2022

Purpose

The purpose of this lab was to introduce the lab group to coding with commands specific to the project. For instance, the Brake command a command which is not found in normal Arduino programming but makes it easier for the lab group to control the AEV and its motion down the monorail rather than using commands like digital or analog write.

Background

The lab group was given a box containing an Arduino nano with two motor control shields on it. The box also contained a component that has wheels and could operate on a single rail. The box contained building materials that attach to the component with wheels to build a structure for the Arduino nano to operate on. The lab group is responsible for building and coding an AEV or Advanced Energy Vehicle capable of completing a mission that entails moving itself along a rail which has both inclines and declines. The AEV must also stop at certain points along the rail, some of which are at an incline.

Procedure Summary

Procedure

The Arduino nano was sat near the electric motors and the motors wires were fastened into the motor controllers next to the Arduino nano. While this was being set up, the code for scenario one was being written. The code was created in a file called “PrgmBasics” and was then moved to the “MyCode” file in the sketchbook folder containing all the AEV Arduino files. This was important because the commands used in the lab are premade and to be able to call these commands the program needs to be in the same folder as them. One person read what the code needed to do line-by-line, and another coded it. Once the code was complete the Arduino cable plugged into the laptop with the code and the code was then compiled and pushed onto the Arduino microprocessor. Then the button on the Arduino was pressed and held for a second to execute the code. The process for making and loading the code onto the Arduino was then repeated for scenario two.

Comment on performance

The electric motors performed well and as expected in the lab. The only unexpected part of running the scenarios was how loud the motors were. While this was used in scenario two to play a song, it was alarming at first when executing scenario one. However all the other motors in the room were just as loud, meaning there was not anything wrong with our motors.

Another performance comment to be made is the motor’s response to the motorSpeed command. While it is physically impossible for any real object to go from stationary to moving instantaneously, the motor will need to accelerate to the speed which takes time. This was evident in the lab when using the motor speed function because its speed would ramp up and down when using the motorSpeed command. This could also be determined by the sound of the

motor when undergoing this command. The motor would get louder when getting to the speed and get quieter when the brake command was called.

Significant Findings

The specific commands used in this lab could limit the success of the AEV in its completion of the mission because using the brake command will only stop its motors from turning rather than applying a braking force to the AEV and stopping it. So rather than thinking of these specific commands in a traditional since the lab group needs to think creatively. So, when brake is used the AEV has time to slow down and stop. Tests with the actual track and the final edition of the AEV need to be conducted so that the lab group can time how long the AEV takes to come to a complete stop after stopping the motors. Also, since the brake command does not actually apply a braking force if the coefficient of friction for the wheels is low enough the AEV could possibly roll down the track when it is supposed to be stopped. To solve this problem a physical brake could be manufactured through 3d printing and implemented using a servo provided by a lab group member. The servo would allow the AEV to stop on inclines. Other solutions could have the motors turn to apply some amount of force to keep the AEV from moving down the incline.

One error that occurred was the code not uploading from the computer to the Arduino Nano. The lab group forgot to look at the green bar on the computer and unplugged the micro usb too soon. When the AEV did not work the lab group inspected the polarity configurations between the motors and the motor controller shields and inspected the polarity between the battery and the Arduino Nano. Once none of these solutions worked the lab group uploaded the code and let it fully upload thus solving the problem. Other minor errors that occurred were issues with the code before uploading like missing semicolons and missing uppercase letters. These issues were easily resolved upon finding them.

Result Descriptions

Both scenarios 1 and 2 called to create an Arduino code utilizing the basic commands to reflect the given scenarios. The scenario that would best suit the mission would be scenario 1. Scenario 1 would work best because it includes movement both forward and backward, but also has the motors run in different directions making the car be still, which the mission needs to be done. When it comes to how “smooth” the scenarios are, scenario 1 is “smoother” while scenario 2 is jerkier. Scenario 2 changes the speeds of the motors and breaks very quickly while scenario 1 gradually accelerates and brakes. So, scenario 1 would provide a better ride for pedestrians.

Implications For the Advanced Energy Vehicle

<u>Reccomendations</u>

While the physical hardware of the AEV will be important in its design, the software will be the backbone of the vehicle. The most important command for the AEV design will likely be celerate because this will allow a smooth transition from stop to go and go to stop. If timing can be perfected or close to perfect, the brake and motorSpeed commands may be useless. Celerate allows for realistic and natural movement which is important to negate skidding. A large obstacle in the design of the AEV will be preventing skidding. This means smart hardware, but smarter software. If the design software could be responsive to the AEV's path, the AEV could surpass a much wider range of obstacles and random errors.

Division of work statement

Matthew Geiger completed The Significant Findings portion and the Division of Work Statement of this Executive Summary. Ben Bazan completed the Results Descriptions section of this Executive Summary. Nick Stassen completed The Procedure Summary section, The Implications for the Advanced Energy Vehicle, and the Arduino Code for Scenarios 1 and 2, for this Executive Summary.

(Note: Ben Bazan was unable to assist with questions pertaining to lab activity because of COVID-19 quarantine procedure. Lab members Matthew and Nick were completely fine doing extra work and completely agree with OSU's current COVID-19 safety precautions.)

Arduino Code for Scenarios 1 and 2.

```
void setup() {
  // put your setup code here, to run once:
  reverse(4);
  motorSpeed(4,25);
  goFor(0.5);
  brake(4);
  goFor(0.1);
  motorSpeed(4,25);
  goFor(0.5);
  brake(4);
  goFor(0.1);
  motorSpeed(4,25);
  goFor(0.5);
  brake(4);
  goFor(0.1);
  motorSpeed(4,15);
  goFor(0.3);
  brake(0.05);
  motorSpeed(4,40);
  goFor(0.3);
  motorSpeed(4,25);
  goFor(0.5);
  motorSpeed(4,15);
  goFor(0.3);
  brake(4);
  goFor(0.05);
  motorSpeed(4,40);
  goFor(0.3);
  motorSpeed(4,25);
  goFor(0.5);
  brake(4);
  goFor(0.5);
  motorSpeed(4,55);
  goFor(0.5);
  brake(4);
  goFor(0.1);
  motorSpeed(4,55);
  goFor(0.5);
  brake(4);
  goFor(0.1);
  motorSpeed(4,55);
  goFor(0.5);
  brake(4);
  goFor(0.1);
  motorSpeed(4,65);
  goFor(0.3);
  brake(4);
  goFor(0.05);
  motorSpeed(4,40);
  goFor(0.3);
  motorSpeed(4,20);
  goFor(0.5);
  motorSpeed(4,15);
  goFor(0.3);
  brake(4);
  goFor(0.05);
  motorSpeed(4,40);
  goFor(0.3);
  motorSpeed(4,25);
  goFor(0.5);
  brake(4);


}
```

```
celerate(1,0,15,2.5);
goFor(1);
brake(1);
celerate(2,0,27,4);
goFor(2.7);
celerate(2,27,15,1);
brake(2);
reverse(2);
celerate(4,0,31,2);
//line 10 below
motorSpeed(4,35);
goFor(1);
brake(2);
goFor(3);
brake(1);
goFor(1);
reverse(1);
celerate(1,0,19,2);
motorSpeed(2,35);
goFor(2);
motorSpeed(4,19);
goFor(2);
celerate(4,19,0,3);
brake(4);
```