To: Professor Jolanta

From: Heath Meyers, Nate Johnson, and Jason Kibler II

Date: 1-29-2020

Subject: Lab 2: Arduino Programming Basics

Acknowledgments:

The group would like to thank The Ohio State University for providing the space necessary and equipment for the lab. The group would also like to thank Dr. Jolanta the help with the installation, configuration, implementation of the Arduino program.

Introduction

The experiment performed in this lab was to familiarize the engineering groups with Arduino coding and to help them better understand the capabilities of the Arduino itself. The Arduino board will be an essential component to the AEV design and in the completion of the main park transportation task. The Arduino has the capabilities to control the motors, as well as sense and react to color changes thanks to a scanner attached to the board. To get the board to work, one must code the board. The coding language used is similar to C#/C++, a standard coding language. The code itself is what tells the Arduino board what to do, it is a combination of commands and algorithms that are designed to complete a specific task. Once the Arduino has the code, the board executes it and from there performs the task(s) described. The physical Arduino board is limited in what it can perform by the code itself.

The group was given a pre-made sketchbook that contained several completed “Classes” or bits of code that they were able to call from in order to complete the actions required in the lab. Calling from a “Class” is done by typing out the class name and sending its arguments within parenthesis. For example, if the speed of a motor need to be changed then the class “motorSpeed” would be called the user would type, motorSpeed(1,80), which would set motor 1’s power to 80%.

The group was also given two separate scenarios. Each scenario had its own set of instructions of what the AEV was supposed to do. To complete those instructions the group wrote two different sets of code, one for scenario 1, and one for scenario 2. Once the code was completed by the group, the Arduino was connected to their computer via USB and the code was uploaded through the Arduino program in which the code was written in.

Results/Discussion

Scenario 1: The first scenario of this lab was a demonstration of how the Arduino programming was to be used in order to create necessary motor functions that will be used in the overarching mission of the labs. The steps that were followed can be seen in figure 1 and the code that was uploaded to the to the Arduino board can be seen in the appendix under scenario 1.

1. Accelerate motor one from start to 15% power in 2.5 seconds.
2. Run motor one at a constant speed (15% power) for 1 second.
3. Brake motor one.
4. Accelerate motor two from start to 27% power in 4 seconds.
5. Run motor two at a constant speed (27% power) for 2.7 seconds.
6. Decelerate motor two to 15% power in 1 second.
7. Brake motor two.
8. Reverse the direction of only motor 2.
9. Accelerate all motors from start to 31% power in 2 seconds.
10. Run all motors at a constant speed of 35% power for 1 second.
11. Brake motor two but keep motor one running at a constant speed (35% power) for 3 seconds.
12. Brake all motors for 1 second.
13. Reverse the direction of motor one.
14. Accelerate motor one from start to 19% power over 2 seconds.
15. Run motor two at 35% power while **simultaneously** running motor one at 19% power for 2 seconds.
16. Run both motors at a constant speed (19% power) for 2 seconds.
17. Decelerate both motors to 0% power in 3 seconds.
18. Brake all motors.
19. Save Program as (Save As:) PrgmBasics

Figure 1: Steps for scenario 1.

Once the code was written in the Arduino programming software using the given sketchbook, the code was then uploaded to the Arduino board and the code was then run. The code did as specified in figure 1.

Scenario 2: Scenario 2 was one that was much less practical compared to scenario 1 but still allows for the continued understanding of the Arduino code. The steps followed for scenario 2 can be seen in figure 2 and the code can be seen in the appendix under scenario 2.

1. Reverse all motors.
2. Power all motors at 25% power for 0.5 second.
3. Brake all motors for 0.1 seconds.
4. Repeat steps 2 and 3 at total of 2 times
5. Power all motors at 15% power for 0.3 second.
6. Brake all motors for 0.05 seconds.
7. Power all motors at 40% power for 0.3 second.
8. Power all motors at 25% power for 0.5 second.
9. Power all motors at 15% power for 0.3 second.
10. Brake all motors for 0.05 seconds.
11. Power all motors at 40% power for 0.3 second.
12. Power all motors at 25% power for 0.5 second.
13. Brake all motors for 0.5 seconds.
14. Power all motors at 55% power for 0.5 second.
15. Brake all motors for 0.1 seconds.
16. Repeat steps 14 and 15 a total of 2 times
17. Power all motors at 65% power for a total of 0.3 second.
18. Brake all motors for 0.05 seconds.
19. Power all motors at 40% power for 0.3 second.
20. Power all motors at 20% power for 0.5 second.
21. Power all motors at 15% power for 0.3 second.
22. Brake all motors for 0.05 seconds.
23. Power all motors at 40% power for 0.3 second.
24. Power all motors at 25% power for 0.5 second.
25. Brake all motors.
26. Save Program as (Save As:) PrgmBasicsStarWars

Figure 2: Steps for scenario 2.

Once the code was created it was, once again, uploaded to the Arduino board and was executed. When executed the code utilized the sound created by the motors to produce the Star Wars theme.

AEV Behavior

Once the Arduino code was uploaded to the Arduino and the program began to run, there was a very clear resistance that took place on several instances throughout the lab. This resistance not only could be seen by the minimal motion at the beginning of each startup, but the resistance could as be heard with the engines screeching but with no movement.

This resistance did not only take place during the initial startup but during almost every single motor start up during the scenarios.

Scenario 1 would be more useful towards the overall goal of trying to transport the tourists to varying sections of the park. This is because scenario 2, although producing music, would cause a much more of a jolting action on the AEV than scenario 1. Yet neither of the two scenarios would be ideal for the given objective because the codes have not been programed to run the specific track smoothly.

Resolving Error

Several difficulties were faced through the process of the lab. The first issue that arose was that the given Arduino sketchbook had to be uploaded on to the Arduino software which took help form the instructor to complete. Next, the Arduino code was new to members of the group and there was an adjustment period that took place before any real strides were made towards the goal of this lab. Continually, even after the code was gotten used to, mistakes continued to be made that were caused by habits from other programing software.

Even more issues occurred once the code was written. One issue was that the Arduino program was not able to recognize the USB connector form the computer to the physical Arduino which also took some help from the instructor to fix. However, once the code was uploaded to the Arduino the code ran smoothly baring the undesirable behaviors mentioned in the AEV behavior section.

Despite the plethora of difficulties that were faced in the is lab, both scenarios were able to be completed with scenario 2 producing the Imperial March form Star Wars using the sound of the fan blades.

Changes Made

Because of the several difficulties that were faced in the lab some changes also had to be made in order to execute both scenarios properly. One change that had to be made was uploading the given sketchbook to the Arduino software. The Arduino board also had to be moved on the AEV so that when the turbines began to spin, they would not hit the board.

Limitations:

Even after being given the sketchbook and using it to produce a similar code to what will be used in the final stages, as well as using the code to produce music; there are still limitations that are on what the code can do. One limitation is that, once the AEV gains momentum it cannot be stopped using the “brake” function because the function only stopes the motor, rather than the AEV itself. In order to stop the AEV by using the motor and turbines, the “reverse” function will have to be used which will also not

instantly stop the AEV because the motors still have to switch direction and counteract the forward momentum. Another limitation is that, although it may not be needed, the motors are not able to operate at 100% power. The operational power limits also limit the weight that the AEV can be and in turn the number of passengers that it can carry when the final goal is reached.

Recommendations

There was not much need for guidance in the actual coding process but rather, as stated in the resolving error section, with the input of the sketchbook and the uploading of the code to the Arduino board.

As for recommendations for making the lab better, the only thing that can be done is to create a third scenario that will allow for the groups to become even more familiar with the code making process and the code uploading process. Having a third scenario will also allow for the development of more ideas of what can be done with the code towards the main goal.

Conclusion

The goal of this lab of becoming familiar with the Arduino programing and the board itself was accomplished through the completion of the two scenarios presented. The first scenario demonstrated how the Arduino board can be coded in order to move the AEV along the monorail. The second scenario furthered this understanding while creating the Imperial March theme when ran. Moreover, this lab also allowed for the understanding of the capabilities of the Arduino and the turbines. The group now has another steppingstone of knowledge towards the completion of the AEV monorail.

Appendix

Scenario 1

```
// Program between here----------------------------------------------------------------------

celerate(1, 0, 15, 2.5); //Increase power of motor 1 from 0% to 15% over 2.5 seconds
goFor(1); //Wait 1s
brake(1); //Brake motor 1
celerate(2, 0, 27, 4); //Increase power of motor 2 from 0% to 27% over 2.5 seconds
goFor(2.7); //Wait 2.7s
celerate(2, 27, 15, 1); //Decrease power of motor 2 from 27% to 15% over 1 second
brake(2); //Brake motor 2
reverse(2); //Reverse motor 2
celerate(4, 0, 31, 2); //Increase power of all motors from 0% to 31% over 2 seconds
motorSpeed(4, 35); //Set all motors power to 35%
goFor(1); //Wait 1s
brake(2); //Brake motor 2
goFor(3); //Wait 3s
brake(4); //Brake all motors
goFor(1); //Wait 1s
reverse(1); //Reverse motor 1
celerate(1, 0, 19, 2); //Increase power of motor 1 from 0% to 19% over 2 seconds
motorSpeed(2, 35); //Set motor 2 power to 35%
motorSpeed(4, 19); //Set all motors power to 19%
goFor(2); //Wait 2s
celerate(4, 19, 0, 3); //Increase power of all motors from 19% to 0% over 3 seconds
brake(4); //Brake all motors

// And here---------------------------------------------------------------------------------
```

Scenario 2

```
// Program between here----------------------------------
reverse(4); //Reverse all motors
motorSpeed(4, 25); //Set all motors power to 25%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.1); //Wait 0.1s

motorSpeed(4, 25); //Set all motors power to 25%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.1); //Wait 0.1s

motorSpeed(4, 25); //Set all motors power to 25%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.1); //Wait 0.1s

motorSpeed(4, 15); //Set all motors power to 15%
goFor(0.3); //Wait 0.3s
brake(4); //Brake all motors
goFor(0.05); //Wait 0.05s

motorSpeed(4, 40); //Set all motors power to 40%
goFor(0.3); //Wait 0.3s

motorSpeed(4, 25); //Set all motors power to 25%
goFor(0.5); //Wait 0.5s

motorSpeed(4, 15); //Set all motors power to 15%
goFor(0.3); //Wait 0.3s
brake(4); //Brake all motors
goFor(0.05); //Wait 0.05s

motorSpeed(4, 40); //Set all motors power to 40%
goFor(0.3); //Wait 0.3s

motorSpeed(4, 25); //Set all motors power to 25%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.5); //Wait 0.5s

motorSpeed(4, 55); //Set all motors power to 55%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.1); //Wait 0.1s

motorSpeed(4, 55); //Set all motors power to 55%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.1); //Wait 0.1s

motorSpeed(4, 55); //Set all motors power to 55%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors
goFor(0.1); //Wait 0.1s

motorSpeed(4, 65); //Set all motors power to 65%
goFor(0.3); //Wait 0.3s
brake(4); //Brake all motors
goFor(0.05); //Wait 0.05s

motorSpeed(4, 40); //Set all motors power to 40%
goFor(0.3); //Wait 0.3s

motorSpeed(4, 20); //Set all motors power to 20%
goFor(0.5); //Wait 0.5s

motorSpeed(4, 15); //Set all motors power to 15%
goFor(0.3); //Wait 0.3s
brake(4); //Brake all motors
goFor(0.05); //Wait 0.05s

motorSpeed(4, 40); //Set all motors power to 40%
goFor(0.3); //Wait 0.3s

motorSpeed(4, 25); //Set all motors power to 25%
goFor(0.5); //Wait 0.5s
brake(4); //Brake all motors

// And here---------------------------------------------------------------------------
```

Lab Participation

Nate Johnson wrote the introduction and discussed the two scenarios. Jason Kibler II commented on the code and attached the code. Heath Myers wrote the resolving error, limitations, AEV behavior, recommendations, the conclusion, and the lab participation sections.