Lab 2: The Demobot

*ECE 564: Fundamentals of Autonomous Robots Lab*

*Team 1: Jacob Cassady, Chase Crutcher, and Olalekan Olakitan Olowo*

*August 28, 2019*

*The group members have worked together and face-to-face at all stages of this project work. The contributions of members to the report and to the codes are equal.*

*(Initials of group members)*

Contents

[Introduction 3](#_Toc18671470)

[Part 1: (Building of a Demobot) 3](#_Toc18671471)

[Part 2: (Command Test) 7](#_Toc18671472)

[Test 1 7](#_Toc18671473)

[Test 2 8](#_Toc18671474)

[Test 3 9](#_Toc18671475)

[Test 4 10](#_Toc18671476)

[Data for powerLevel = 50 10](#_Toc18671477)

[Data for powerLevel = 100 11](#_Toc18671478)

[Test 5 13](#_Toc18671479)

[Test 6 14](#_Toc18671480)

[Test 7 15](#_Toc18671481)

[Test 8 16](#_Toc18671482)

[Test 9 17](#_Toc18671483)

[Test 10 18](#_Toc18671484)

[Conclusion: 19](#_Toc18671485)

[Suggestions: 20](#_Toc18671486)

# Introduction

The goal of this lab was to use the Wallaby Robot Controller with the KISS Web IDE and the assorted parts given to build a Demobot and have it perform various tasks. C programs were created and tested in order to get the Demobot to complete the assorted tasks given.

# Part 1: (Building of a Demobot)

The Chassis and Motors shown in Figure 1 are mounted together using medium bolts. This will act as the engine for the Demobot.

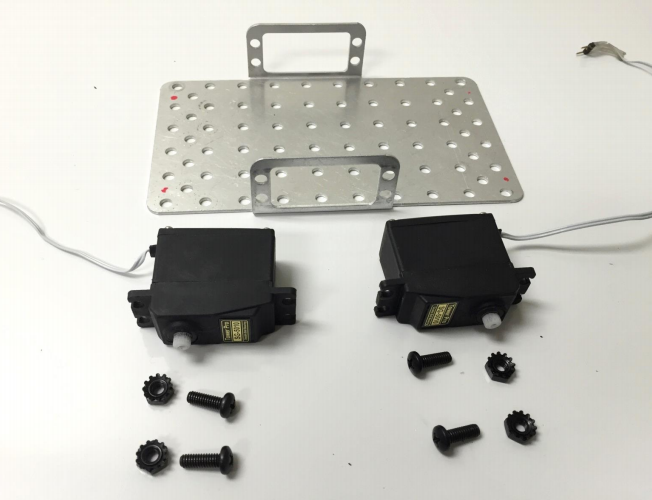


Figure 1: Chassis and Motors

The Wheel Assembly and Caster Ball in Figure 2 are attached to the Demobot using screws, brackets, and nuts. This will allow the Demobot to move forward and backward as well as be able to turn due to the manipulation allowed from the Caster Ball.

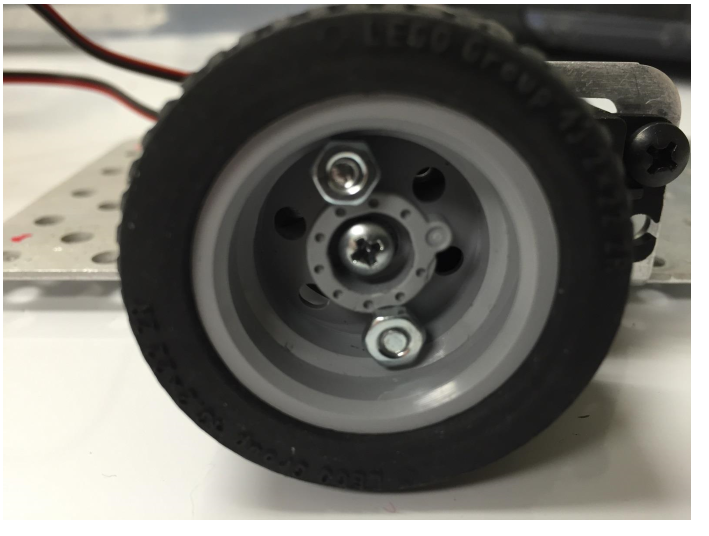


Figure 2: Wheel Assembly

With the wheel assembled earlier, the round metal servo horn was push unto the servo spindle. Securing the wheels in place with a long screw it was repeated for both sides.

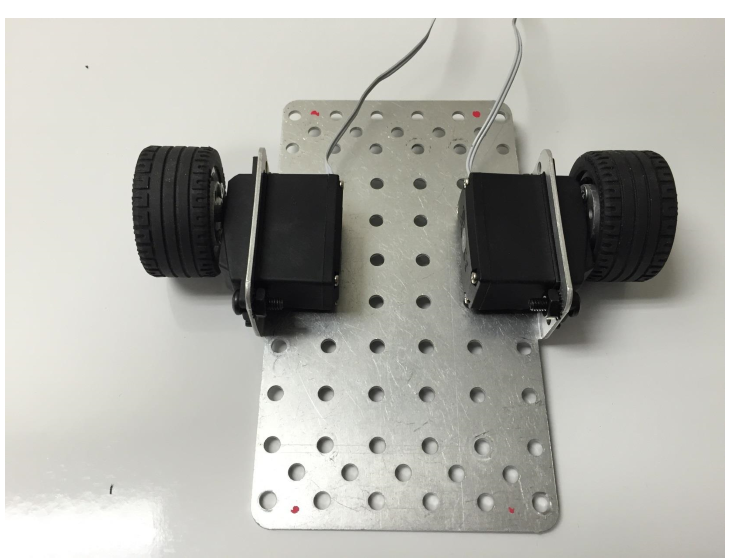


Figure 3: Wheel Assembly unto the chassis

The ball bearing caster and two long bolts were assembled, put through the thick washer and 3 hole lego attached underneath with the ball initially pop out to allow for the fixing.

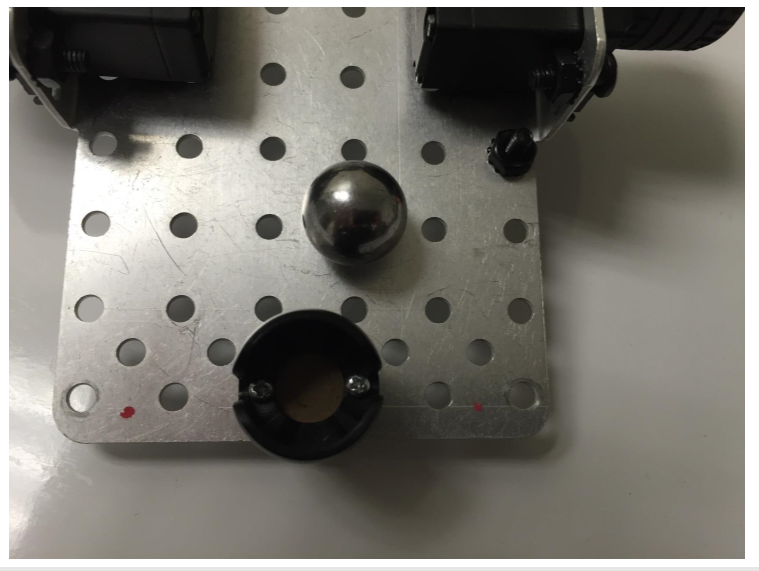


Figure 4: Wheel Assembly and Caster Ball

The Tophat Sensor, a reflectance sensor in short range is mounted at a fixed distance above the ground level and attached to the front of the chassis facing down. The sensor was plugged into the analog port of the wallaby. On the right side of the chassis is servo bracket with the servo motor attached using the same medium bolts to hold it together. Finally, the servo horn is attached to the servo motor as seen in Figure 3.

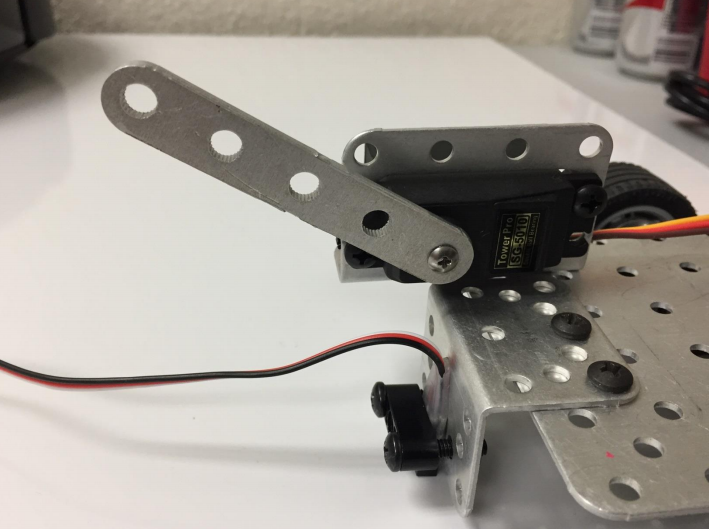


Figure 5: Tophat Sensor and Servo Bracket/Motor with Horn attached

A 7 hole lego beam is used to mount the Wallaby microcontroller to the chassis itself as shown in Figure 4. This is needed to allow space for the battery and wiring to lay underneath the Wallaby when it is being used.

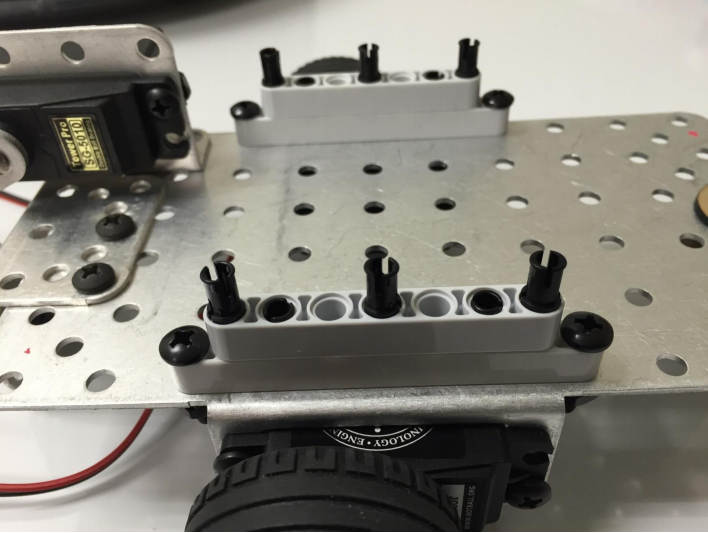


Figure 6: Mounting Bracket

The Wallaby and battery pack is attached to the mounting bracket as shown in Figure 5. This allows the Wallaby and battery to ride on top of the Demobot as the robot performs various assigned tasks.

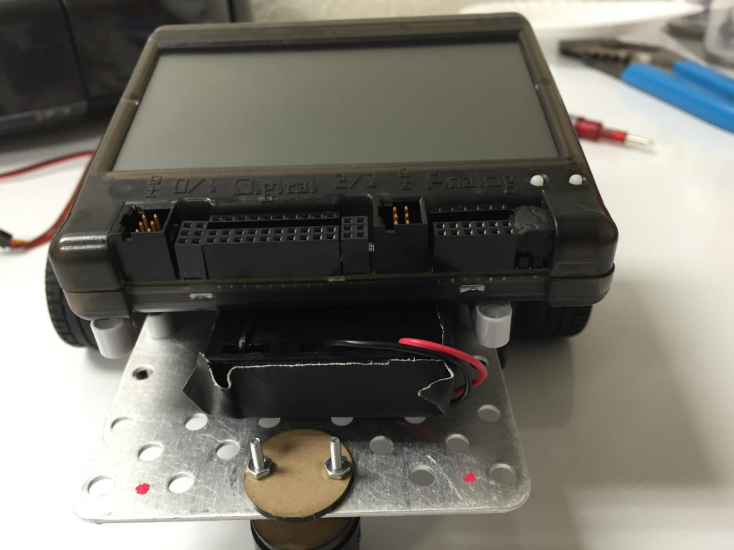


Figure 7: Wallaby and Battery Pack mounted to the Chassis

For the complete assembly of the Demobot, the wires were connected to the appropriate ports on the Wallaby. The tophat sensor to the analog port, the motors for driving the wheels to the motor port and the servo motor was connected to the servo motor port which appears to be at the far end of the Wallaby.

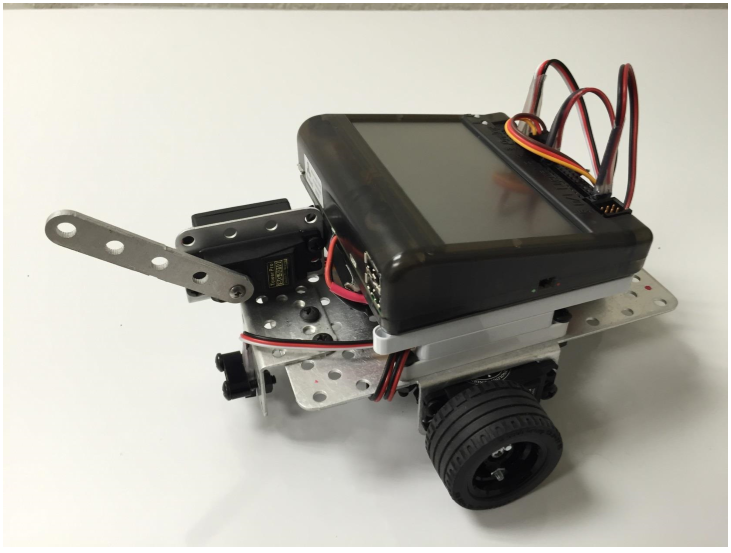


Figure 8: A complete Demobot

# Part 2: (Command Test)

## Test 1

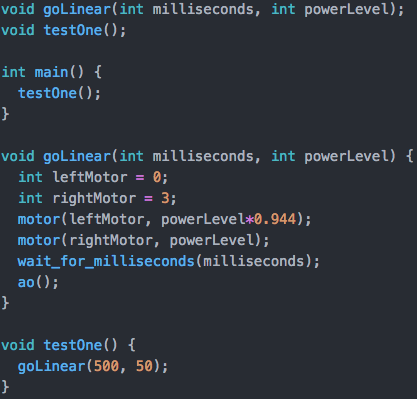


Figure 9: Test 1 src

This program is very simple. It runs forward at a powerLevel of ~50 for half a second. A factor was added to reduce the powerLevel of the left motor to make the robot go more straight.

## Test 2



Figure 10: Test 2 src

This program has the robot start and end in the same location. It drives forward, drives backwards, turns to the right while driving forward, and then backs up in the opposite direction. All commands were created using the goDemobot function.

## Test 3

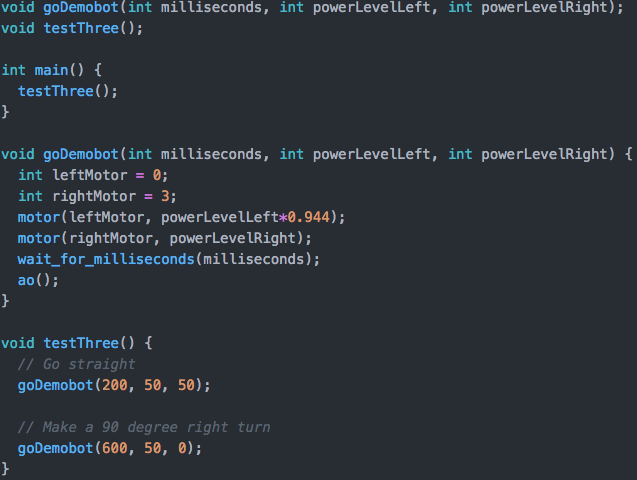


Figure 11: Test 3 src

This program has the robot go straight and then make a 90 degree turn. The sharpness of the 90 degree turn can be altered by increasing/decreasing the magnitude of difference between the left and right motors. The sharpest possible right turn would have the left motor at a power level of 100 while the right motor has a power level of -100.

## Test 4

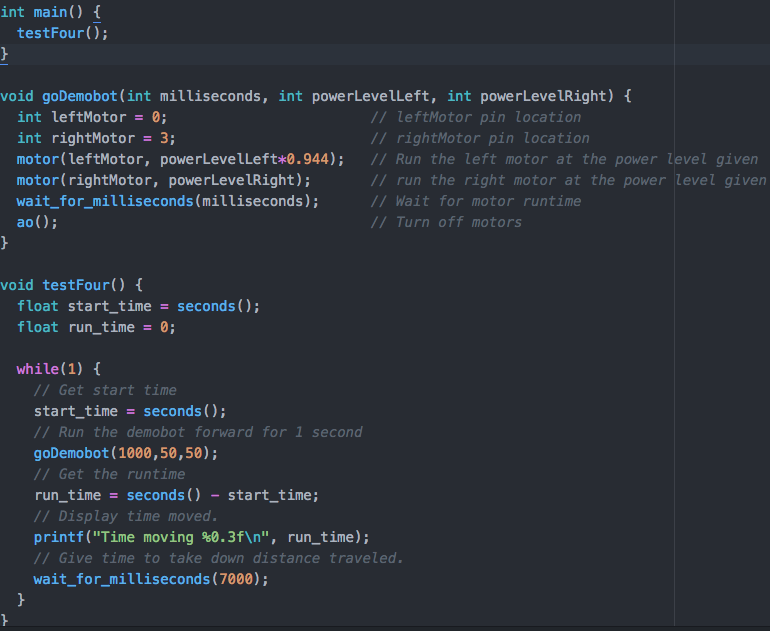


Figure 12: Test 4 src

As you can see from **Figure 9** and **10** below, the speed at different moments of time during the test is very linear in nature for both 50 and 100 power levels. These graphs may not be as linear as the power level drops on the Wallaby as the functions used for movement are based off the power-dependent motor() command.

### Data for powerLevel = 50

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time Differential (s)** | **Time (s)** | **Differential Distance Traveled (inches)** | **Distance Traveled (inches)** | **Speed (inch/sec)** |
| 1 | 1 | 6.4 | 6.4 | 6.4 |
| 1.06 | 2.06 | 6.45 | 12.85 | 6.084906 |
| 1.05 | 3.11 | 6.48 | 19.33 | 6.171429 |
| 1 | 4.11 | 6.4 | 25.73 | 6.4 |
| 1.02 | 5.13 | 6.48 | 32.21 | 6.352941 |

Table 1: Distance Traveled @ PowerLevel = 50

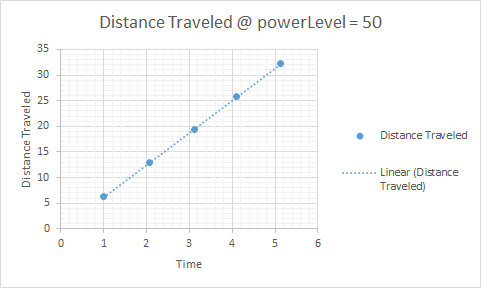


Figure 13: Distance Traveled @ powerLevel = 50 Graph

### Data for powerLevel = 100

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time Differential (s)** | **Time** | **Differential Distance Traveled (inches)** | **Distance Traveled (inches)** | **Speed (inch/sec)** |
| 1.02 | 1.02 | 11.9 | 11.9 | 11.66667 |
| 1.01 | 2.03 | 11.95 | 23.85 | 11.83168 |
| 1.03 | 3.06 | 11.92 | 35.77 | 11.57282 |
| 1.04 | 4.1 | 11.95 | 47.72 | 11.49038 |
| 1 | 5 | 11.93 | 59.65 | 11.93 |

Table 2: Distance Traveled @ PowerLevel = 100

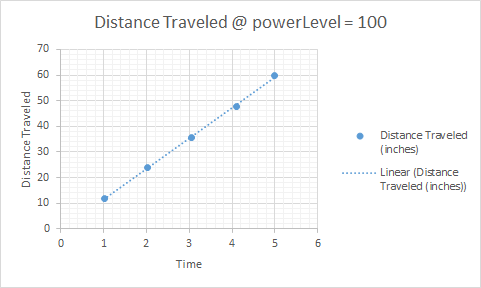


Figure 14: Distance Traveled @ powerLevel = 100 Graph

## Test 5

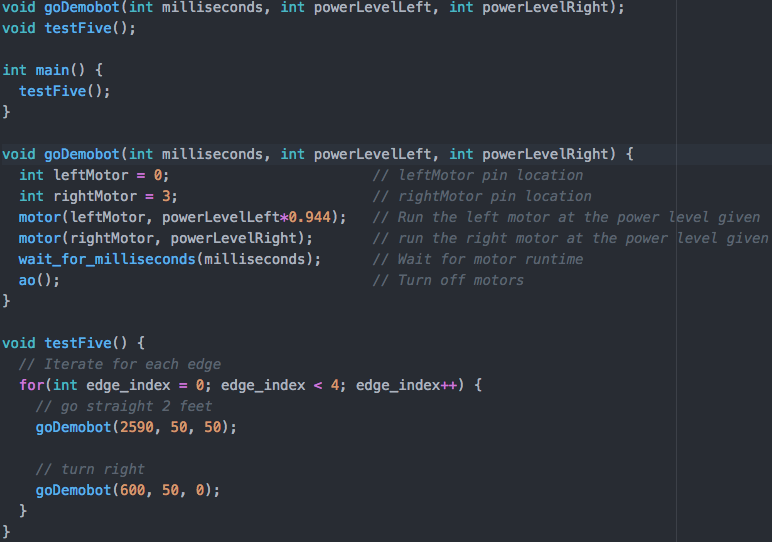


Figure 15: Test 5 src

This program tracks a 2x2 square. The robot did not perform the same each time during three different tests. We did our best to walk the millisecond parameter in to perfect 2 feet for each edge but as the robot ran out of power, the parameter would need to be increased. This is because the motor() command is dependent on the power level.

## Test 6



Figure 16: Test 6 src

This program does the same as Test 5 but for a 4x4 square. After repeating the experiment three times, it was clear the error was larger than with a smaller square. This is because slight errors in the straightness of the goDemobot command have more time to accumulate. Different surfaces caused for different averages in speed for a given powerLevel. This meant the algorithm did not easily transfer between test surfaces.

## Test 7

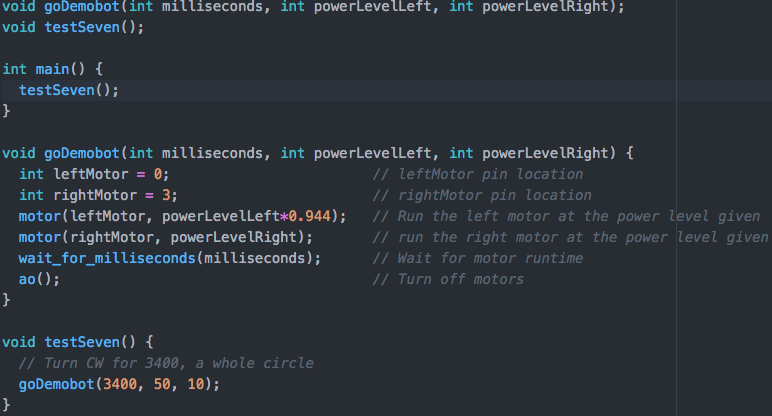


Figure 17: Test 7 src

This program has the robot transcribe a clockwise circle.

## Test 8

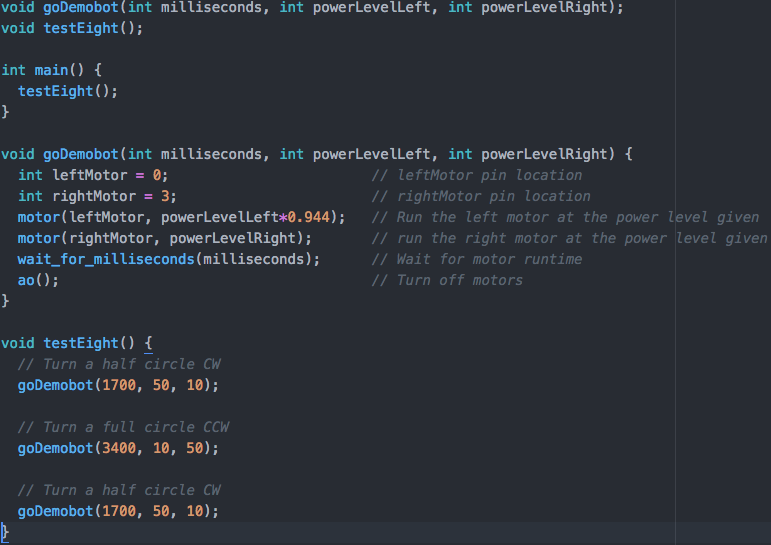


Figure 18: Test 8 src

This program has the robot transcribe half a clockwise circle, then a full counter clockwise circle, and then a final clockwise half circle. This produces a perfect figure 8.

## Test 9

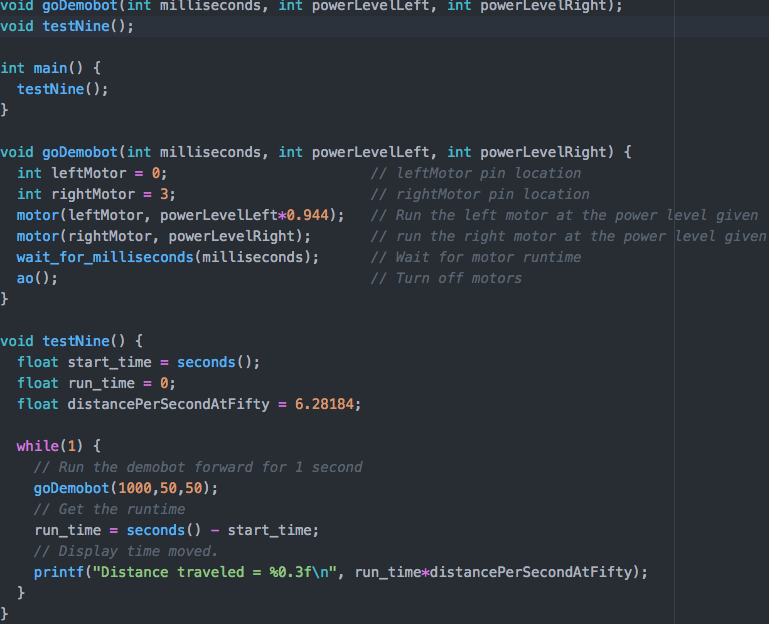


Figure 19: Test 9 src

The readings displayed were fairly accurate. We used the slope of the graph from test 4 to determine a factor to relate the runtime of the robot to the distance traveled at a given power level. Again, as noted, this speed constant is different for different surfaces. As a consequence, the distance traveled was less accurate when tested on a flat rubber surface than the smooth tile floor.

## Test 10



Figure 20: Test 10 src part 1



Figure 21: Test 10 src part 2

The differences between the mav, mrp, and motor commands are subtle. Mav command stands for Move At Velocity. This command takes a motor number and a velocity amount. It is similar to the Motor command, which moves the motor at a percentage of its output instead of a set velocity. On the other hand, the mrp command stands Move Relative Position. This command takes three parameters:

* A motor number
* A velocity
* A number of ticks

It rotates at the velocity for the number of ticks and then stops.

To recreate test 6 using the Mav and MRP commands, I made analogous goDemobot functions for each command. This allowed our team to walk in the values for how long the robot should run given a velocity (for mav) or a velocity and number of ticks (for Mrp). The wait\_for\_milliseconds command within the Mrp function ensures the motors are not cut off by other function calls.

# Conclusion:

In this particular exercise, under the supervision of the TA, a complete Demobot was built and programmed to observe the following actions. These actions required the Demobot to move in a straight line, turn going in the forward or backward directions, go in a straight line then turn 90 degrees to the right. Other tasks carried out were the programming of the Demobot to move in a 4\*4 feet square. The report also contained the programming code installed to execute the tasks.

# Suggestions:

It is suggested that more parts ideally suited for the project should be provided. As many students had to improvise with what is available and this might not provide the desired outcome expected.