Control of Mobile Robotics
Spring 2016

Lab 1

Sensors and Actuators

8 February, 2016

Gerardo Sanchez
Adeoluwa Oyenusi

Task Description

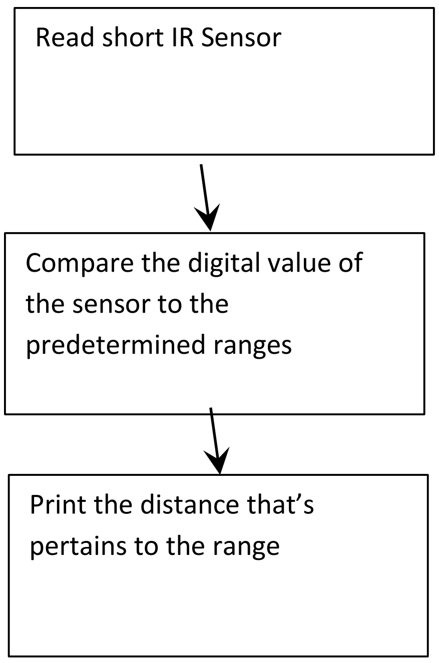
The objective of this lab is to get the students familiar with their robots. The lab consists of 5 tasks.

1. Utilize all three IR short distance sensors to evaluate the distance of an object. Digital values from the sensor will be converted to represent a distance from the object. The distance will range from 2-10 inches.
2. Implement the use of the IR long distance sensor and the IR short distance sensor to evaluate the distance of an object in the range of 2 to 59 inches. The IR short distance sensor will evaluate distances from 2- 8 inches and the IR long distance will evaluate from 8 – 59 inches.
3. The students are to make the robot perform four different open loop motions without using any sensor information. The robot will move 50 inches straight forward, 50 inches straight backward, clockwise circle and counter clockwise circle. The circle will be 20 inches in diameter.
4. The robot will move in a square trajectories which are 20 inches on every side. One traversal will feature right turns and one the other will use left turns to complete the square.
5. The robot will move along a figure “8” path with the circles having a diameter of 20 inches.

Short IR Range Sensor

Video link: https://www.youtube.com/watch?v=XTL1sy3CNY0

Since the front short sensor, the right short sensor, and the left short sensor are all the same devices, we utilized the same set of values to determine the distance from the analog readings. We averaged 10 IR value measurements at each distance from 2 inches to 10 inches and assigned ranges for the each distance.


Read short IR Sensor
Compare the digital value of the sensor to the predetermined ranges
Print the distance that’s pertains to the range

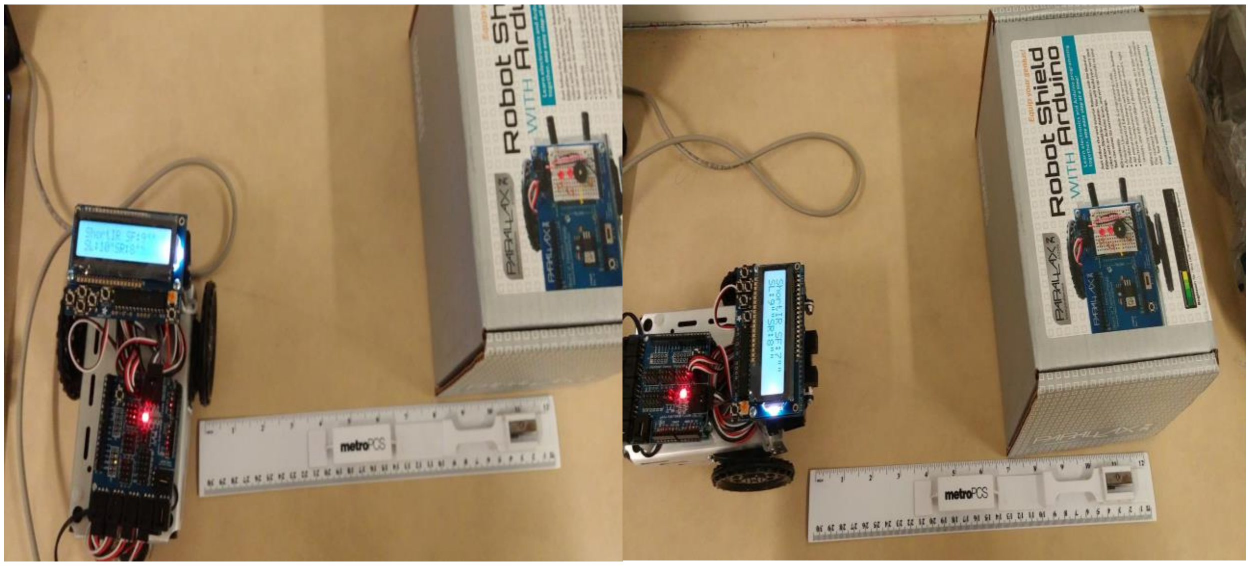
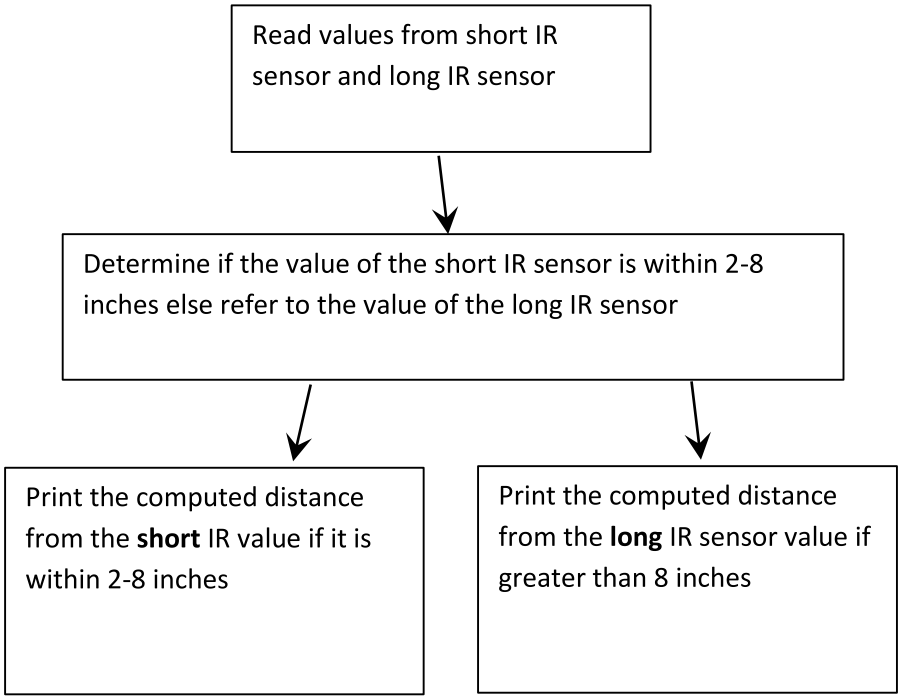


Figure 1 - Front and Right IR Sensor Test

Long IR Sensor

Video link: https://www.youtube.com/watch?v=Ce3zzJZdNEc

For the long IR sensor, we broke up the distances into groups and averaged 10 measurement samples at each distance for each group. Each group is assigned a function to determine the distance based on the sensor reading. For example, a group consists of distance 9 inches to 17 inches and they use the same linear function. Each distance was sampled 10 times to get an average value.



Read values from short IR sensor and long IR sensor

Determine if the value of the short IR sensor is within 2-8 inches else refer to the value of the long IR sensor

Print the computed distance from the **short** IR value if it is within 2-8 inches

Print the computed distance from the **long** IR sensor value if greater than 8 inches

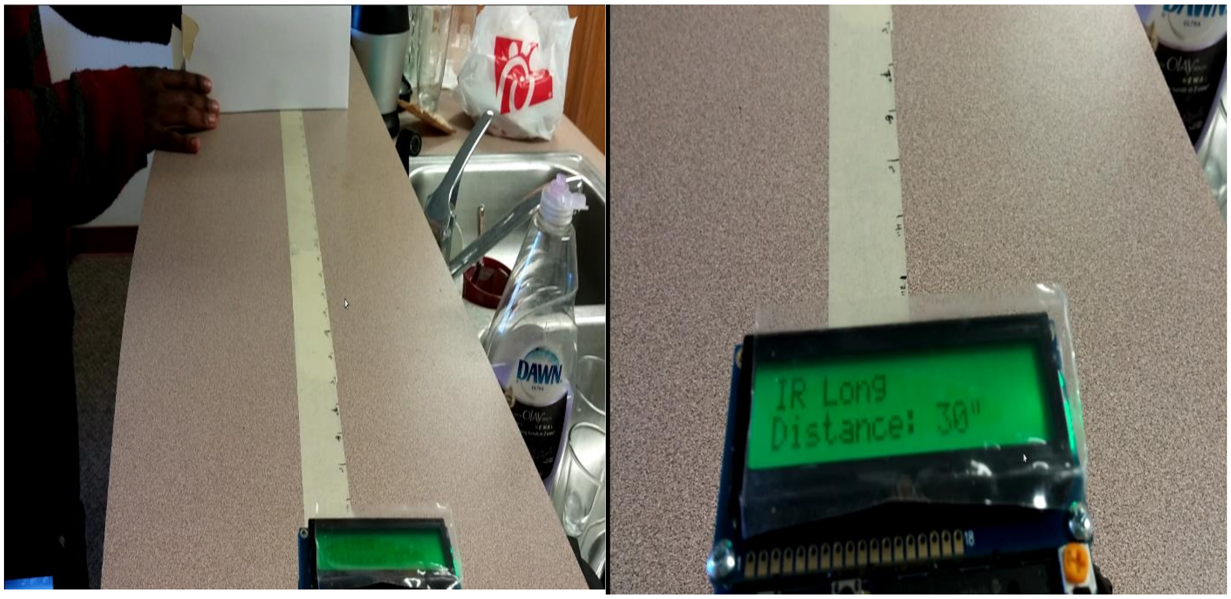
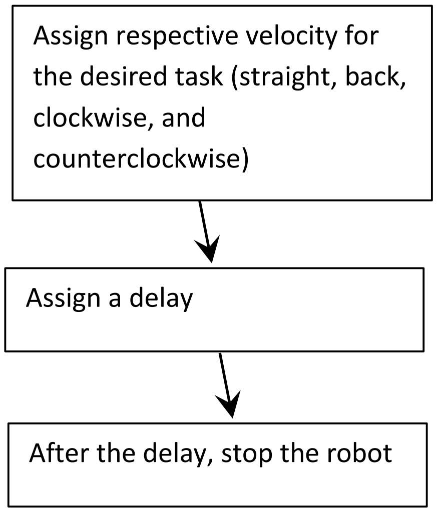


Figure 2 – Testing the long and short IR sensors

Open Loop Motor

Video link: https://www.youtube.com/watch?v=GV4dfQR1Avk

We implemented different timing values to get the robot to move 50 inches forward, backward, complete 20 inches clockwise and counterclockwise circles.


Assign respective velocity for the desired task (straight, back, clockwise, and counterclockwise)
Assign a delay
After the delay, stop the robot

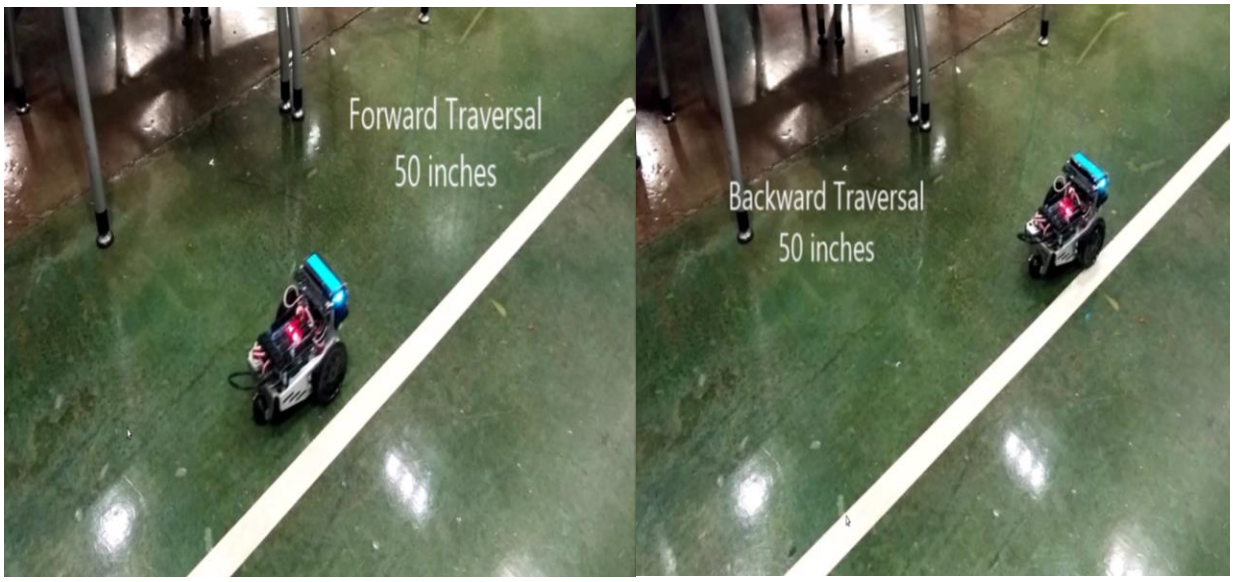


Figure 3 – Forward and Backward Traversal

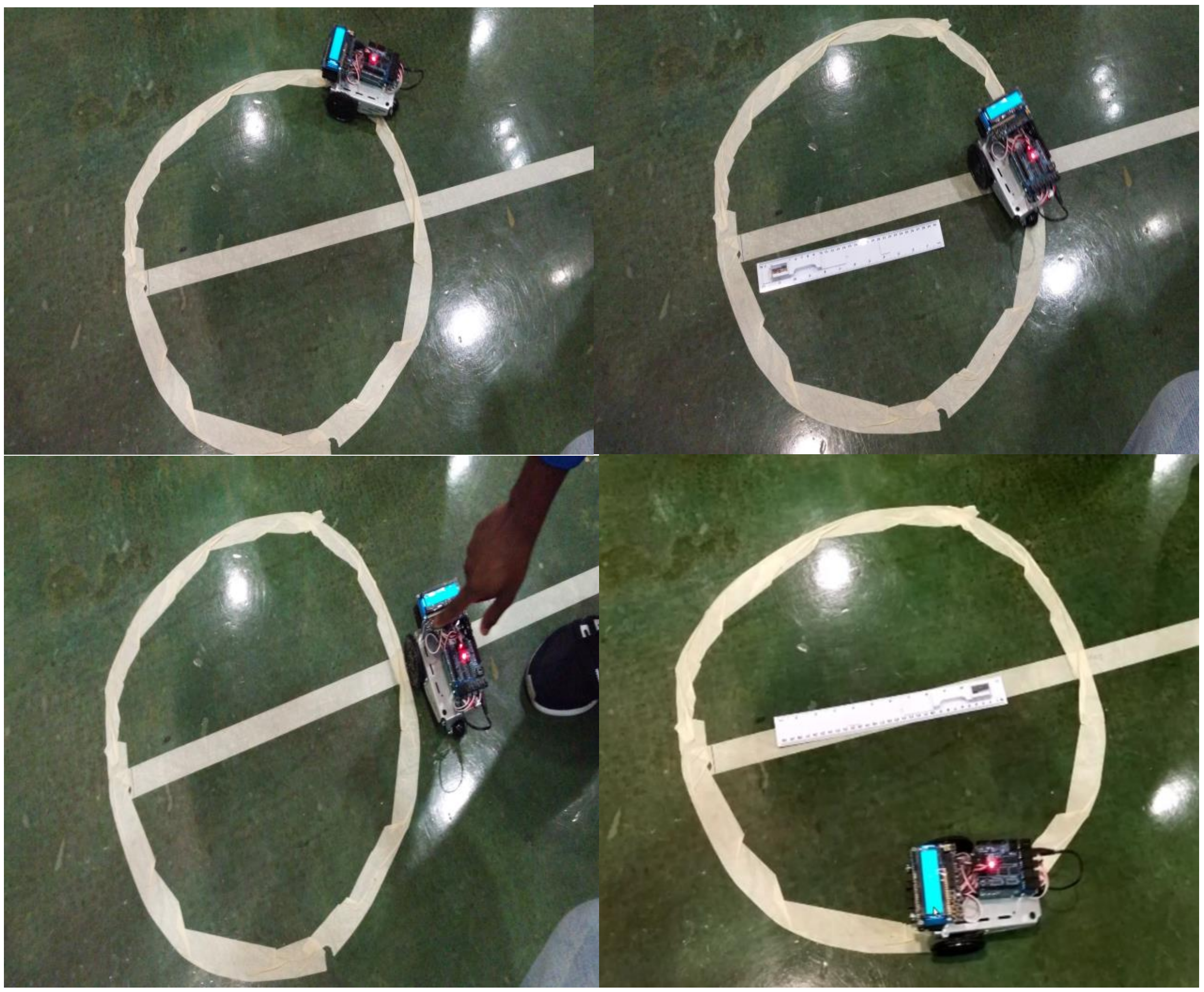
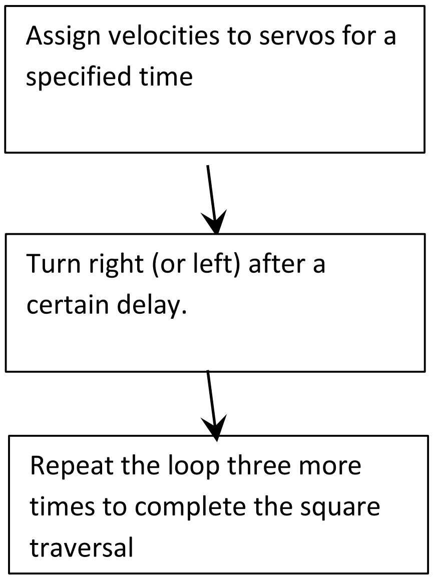


Figure 4 – Clockwise and Counter Clockwise Traversal

Square

Video link: https://www.youtube.com/watch?v=9T3SwUbUimk

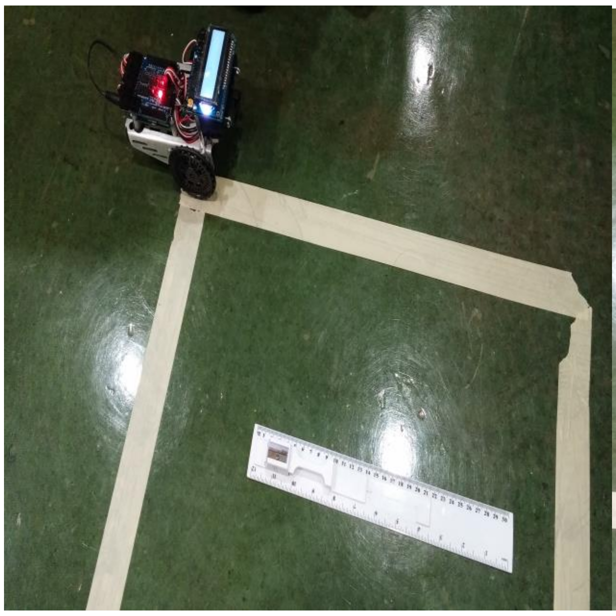
We tested different time values to know when the robot moved 20 inches. We calculated how long it would take to spin clockwise for right turn and how long it would take to spin counter clockwise for left turn.



Assign velocities to servos for a specified time

Turn right (or left) after a certain delay.

Repeat the loop three more times to complete the square traversal



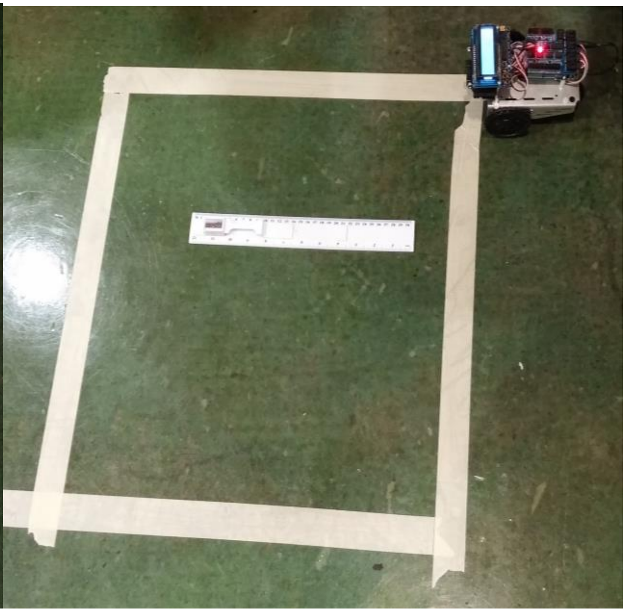
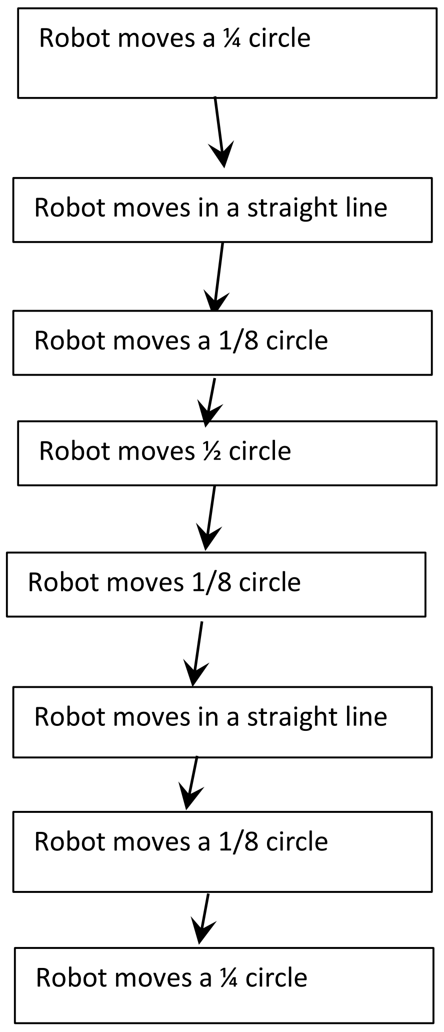
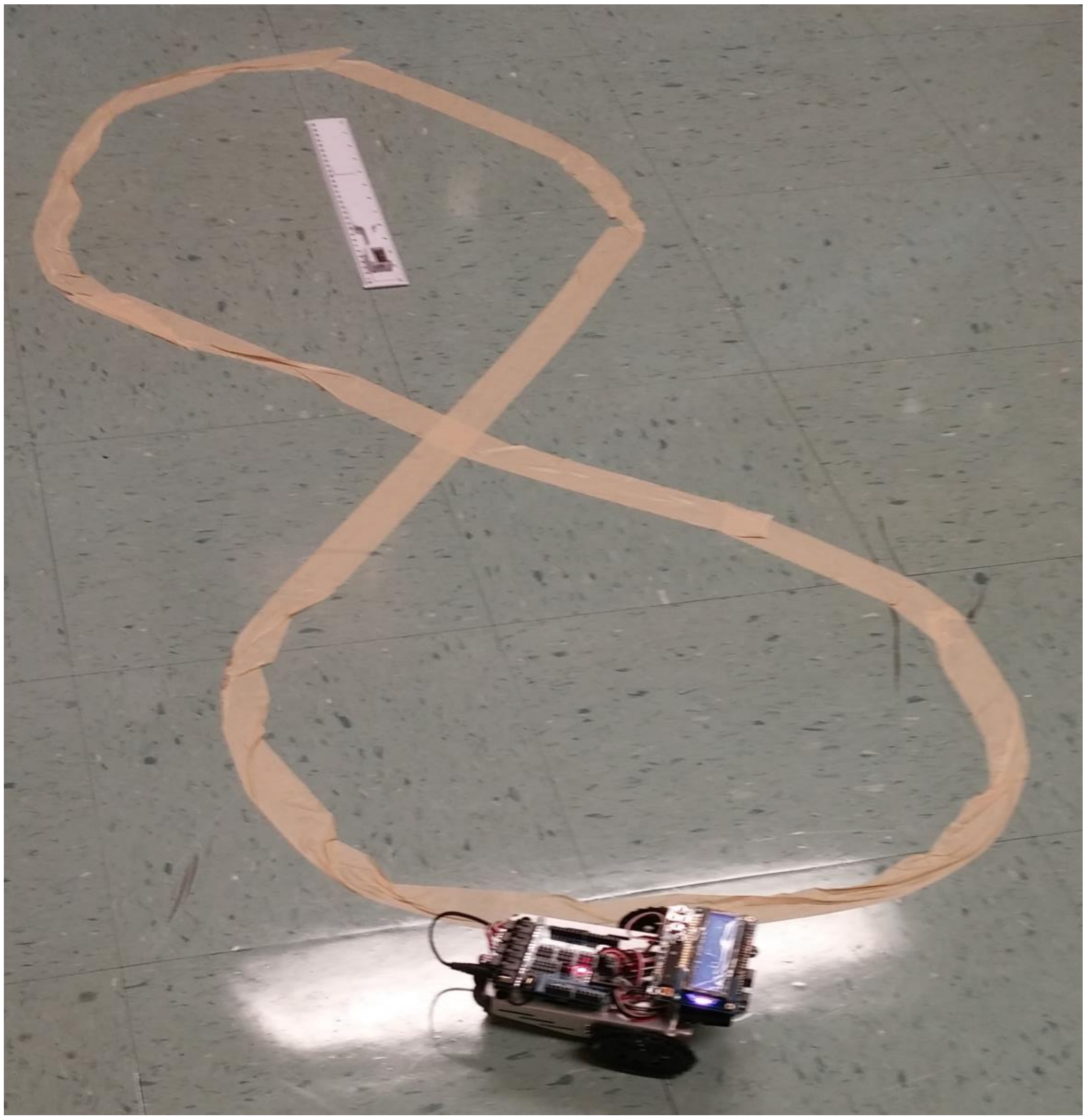


Figure “8” Loop

Video link: https://www.youtube.com/watch?v=PxYqiT7WUDc

We decided to break up the figure 8 into 2 circles and 2 diagonal lines. We calculated the time it would take to complete ¼ of the circle, ½ of the circle, and ⅛ of the circle. Then we used these timings to time how long the robots will turn for.





Code Description

3 IR Short Sensors

The code below shows how we compute the distance of an object through the sensors. The function “computeDistance_SIR” calculates the distance and takes in the IR sensor value as an input. The variable “distance” is used to hold the distance calculated. We initialize distance with -1 so that in case the value does not fall within the ranges in the if else statements, the distance displayed will remain as the last one that was there. We have a nested if else statements which check the range the IR value falls in. If the value falls in a particular range, then the distance is assigned to the “distance” variable.

```
int computeDistance_SIR(int value){

  int distance = -1; //-1 signals that it did not qualify as a new distance
  if( value > 473 && value < 530)
  {
    //2 inches
    distance = 2;
  } else if( value > 335 && value < 366) distance = 3; //3 inches
      else  if(value > 246 && value < 268) distance = 4; //4 inches
          else if(value > 195 && value < 229) distance = 5; //5 inches
            else if (value > 175 && value < 191) distance = 6; //6 inches
              else if( value > 143 && value < 167) distance = 7; //7 inches
                else if(value > 121 && value < 136) distance = 8;//149 causes
an overlap with 7 & 8 inches, changed to 136
                  else if (value > 107 && value < 135) distance = 9;
                    else if(value < 100) distance = 10; //10 inches

  Serial.print(" sensor: ");
  Serial.print(distance);
  Serial.print(" dValue: ");
  Serial.print(value);
  Serial.print("\n");

  return distance;
}
```

Long IR Sensor

For this task, we used the short sensor to get distances from 2 to 8 inches and the long sensor to detect distances from 9 to 59 inches. The short sensor algorithm in this section is the same as the one explained above. However, a different approach was used to detect distances from 9 to 59 inches. An equation for each group was created from the sample values gotten from the long IR sensor. Similar to the short range algorithm, this algorithm used the long IR value to know which equation to use. Due to the large amount of noise and the long IR sensor quality, we were not able to read accurate values at distances greater than 49 inches. As a result, we did our best to estimate the distances at that range.

```
int computeLongDistance(int LValueF)
{
  int distance = -1;

  //if (LValueF > 235 && LValueF < 520)
  if(LValueF > 280 && LValueF < 520)
  {
    //distance = ((7E-05) * LValueF * LValueF) - (0.0942 * LValueF) + 36.943;
//8 to 20 inches
    //distance = ceil(distance) + 1;
    distance = ((-.0397) * LValueF) + 28.146; //good from 9-17

  } else if (LValueF > 160 && LValueF <= 240){ // 21 to 30
    distance = (-0.1341 * LValueF + 50.624);//y = -0.1341x + 50.624
  //} //else if(LValueF > 105 && LValueF <= 150){ //31 to 40
    //distance = (-0.2294 * LValueF) + 65 + 1;//y = -0.2294x + 65.054
    //distance = (-0.2334 * LValueF) + 66.127; // y = -0.2334x + 66.127
Worked
  } else if(LValueF > 83 && LValueF <= 115)
  {
    distance = (-0.3079 * LValueF) + 75.64;//y = -0.3079x + 75.64
  } else if (LValueF > 90 && LValueF < 96)
  {
    distance = 50;
  }
  else if (LValueF > 75 && LValueF < 85)
  {
    distance = 55;
  } else if (LValueF > 56 && LValueF < 75)
  {
    distance = 59;
  }
```

Open Loop Motor

For this task we had to control the motors by sending appropriate values to them. We created a library to help us interface with the motors and also used it to create other functions that would help with this task. The "cmd_vel" function was implemented in the library and is used to feed in values into the motors.

```
void robotMover::cmd_vel(int left, int right){

  if(left != -1) LServo.write(left);
  if(right != -1) RServo.write(right);

}
```

For the forward motion, we used the function "move_fwd" which simply passed the values into the "cmd_vel" function to make the robot go straight. This was achieved by passing the value 100 to the left motor and 80 to the right motor. The function has an input parameter which took in the time that the robot is to perform the forward motion. In order to move the robot 50 inches forward, we passed in the value 9000 which is 9 seconds.

```
void robotMover::move_fwd(int time){
    cmd_vel(100, 80);
    delay(time);
    cmd_vel(90, 90);
}
```

The "move_bkwd" function worked similar to the "move_fwd" function, only that different values were passed in cmd_vel. 9000 was also passed in as the time.

```
void robotMover::move_bkwd(int time){
  cmd_vel(80, 100); //move the robot backwards
  delay(time);
  cmd_vel(90, 90);
}
```

The clockwise circle used the function "circle_20". The function calls the "cmd_vel" function and passes 96 and 87. The time passed in was 20000 (20 sec).

```
void robotMover::circle_20(int time){
    cmd_vel(96, 87);
    delay(time);
    cmd_vel(90,90); //stop the robot
}
```

The counter clockwise circle used the function "counter_circle_20". It passes in the values 96 and 87 into "cmd_vel" and delays for time 20000 (20 sec).

```
void robotMover::cnt_circle_20(int time){
    cmd_vel(94, 83);
    delay(time);
    cmd_vel(90,90); //stop the robot
}
```

Square

The algorithm for the square used the functions spin_clkwse, spin_cntclkwse, and move_fwd which was described above. We made the robot move forward make a left turn and made it repeat those steps four times to complete the square. The forward motion used the “move_fwd” function and the left turn used the “spin_cntclkwse”. The “spin_cntclkwse” function

```
for(int i = 0; i < 4; ++i){
        lcd.print("moving forward");
        myMover.move_fwd(3800);
        lcd.clear();
        lcd.print("left turn");
        myMover.spin_cntclkwse();
        lcd.clear();
}

void robotMover::spin_cntclkwse(){
  cmd_vel(90, 80);
  delay(1350);
  cmd_vel(90, 90);
}
```

The same process was used for the square making left turns was used for making right turns, only that it used “spin_clkwse” to make the right turns.

```
for(int i = 0; i < 4; ++i){
        lcd.print("moving forward");
        myMover.move_fwd(3800);
        lcd.clear();
        lcd.print("right turn");
        myMover.spin_clkwse();
        lcd.clear();
}

void robotMover::spin_clkwse(){
  cmd_vel(100, 90);
  delay(1250);
  cmd_vel(90,90);
}
```

Figure 8 Open Loop

Our mobile robot utilizes the fundamental motion functions to complete the figure 8 motion. The code below describes the method of how the robot traverses the figure 8. The main idea is to split the track into different sections. The figure 8 loop consists of rudimentary motions previously developed earlier in this lab. Utilizing these motions with specific timing delays, we can achieve the closest representation of the figure 8 loop.

```
lcd.print("Making 8 loop");
lcd.clear();
lcd.print("1/4 clk");
myMover.fig8_circle_20(6000);// 1/4 of a circle
lcd.clear();
lcd.print("Straight");
myMover.move_fwd(4860);//move forward 15inches
lcd.clear();
lcd.print("1/8 cnt_clk");
myMover.fig8_cnt_circle_20(2500); // 1/8 of a circle
lcd.clear();
lcd.print("1/2 cnt_clk");
myMover.fig8_cnt_circle_20(9000); // 1/2 of a circle
lcd.clear();
lcd.print("1/8 cnt_clk");
myMover.fig8_cnt_circle_20(2400); // 1/8 of a cirlce
lcd.clear();
lcd.print("Straight");
myMover.move_fwd(3800);//move forward 15inches
lcd.clear();
lcd.print("1/8 clk");
myMover.fig8_circle_20(2500);// 1/8 of a circle
lcd.clear();
lcd.print("1/4 clk");
myMover.fig8_circle_20(6000);// 1/4 of a circle
lcd.clear();
lcd.print("Done");
```

Conclusion

The introductory lab proved to be a good stepping stone for learning about the interface techniques of the robot. Many of the tutorials proved to be helpful in setting up the robot; however, we encountered many issues trying to configure our robot for the desired tasks. Many of the tasks are seemingly trivial but the development and testing of the robot must be done from start to finish in one session. For example, our clockwise traversal was successful upon testing but we had to reconfigure our motion library to have the robot move along the track the next day. The requirements seem to be unclear on how precise we are to make measurements on the Long Range IR Sensor, because we ran into issues. For testing purposes we changed the lighting environment for the long IR in order to get more accurate results, but we were limited at about 49 inches. For engineering purposes, the IR sensors are accurate to about five inches so it a discrepancy of 1-2 inches would not affect motion planning algorithms. This lab was good because it allowed us to see how much EMI can affect noisy and fuzzy data.