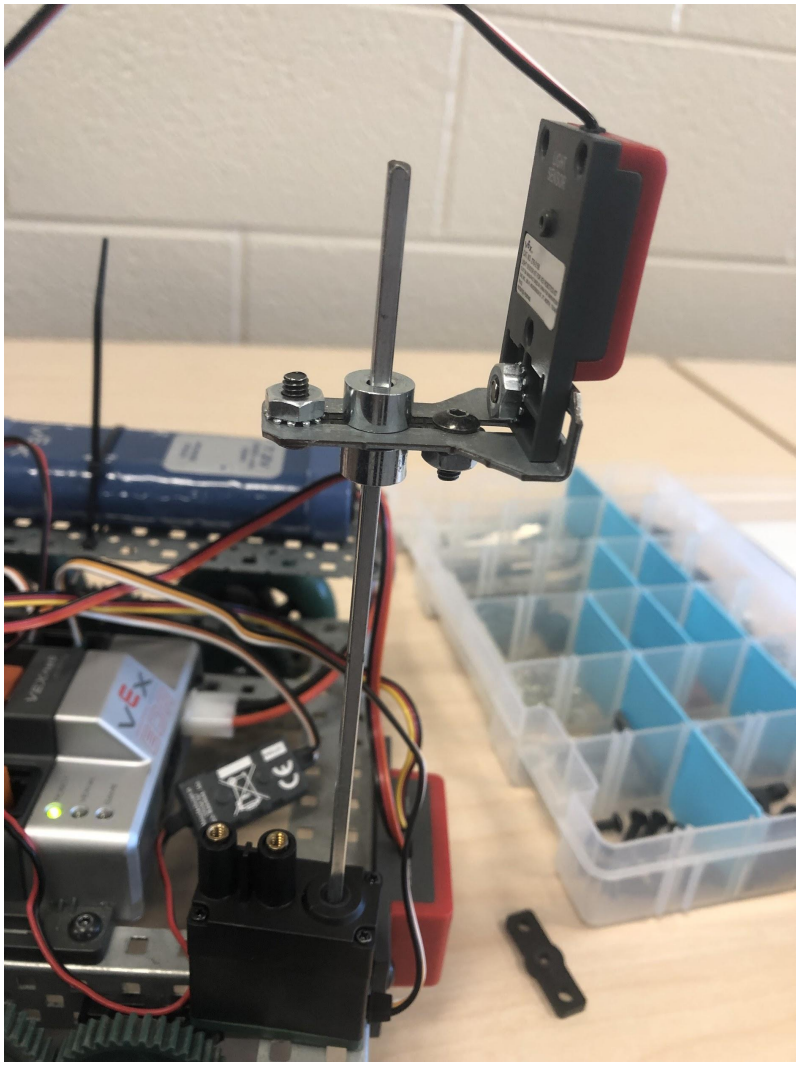
Peter Atsaves
Assignment 3

Introduction

Our task was to create a squarebot that is able to actively scan for a light source and go towards it on its own. To accomplish this, we had to have our robot sense light so we needed to add a light sensor. We also had to find the accuracy of the light sensor and its direction. To do this, we had to attach a protractor and a potentiometer and see the relation between the two. These would help us calibrate the light relative to the potentiometer. After our light sensor was calibrated, we had to code our robot so that it could sense light and travel in the direction where the light is the strongest so that it could eventually reach the light source.

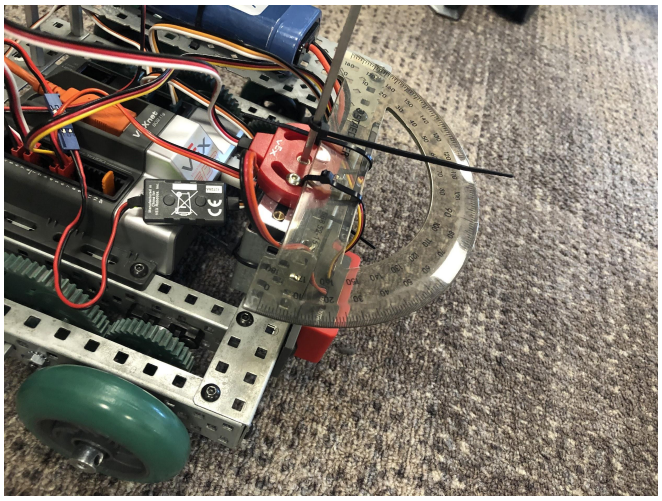
Active Directionality Sensing

For the design, we used a servo and put a rod into the cube shaped hole. We then used a thin metal slate which we attached perpendicular to the rod. We held this in place with a washer on top and on the bottom so it wouldn't fall. We then attached the light sensor perpendicular to the metal slate and screwed it in. Our design looks like an upside down h, with the rod going up, a metal piece attached, and then the light sensor on top of that. For the code, we simply had the servo rotate from position -127 to 127. We did this in increments of 10. We set an initial value for the position of the strongest light, and the value of the strongest light. After each run in our for loop, we compare the current light to the strongest light. If the light was stronger, we reassigned the strongest light and position values. Once the servo rotated all the way, we had the servo move to the position with the strongest light and stay there for a couple of seconds. The only main problem with our approach was that we used a very long rod and because of this, we have to take longer amounts of time between each servo move. If the rod was small, we would be able to scan for light much quicker because of the torque.



Accuracy Measurement

For the design, we had to attach the servo to the robot and add a potentiometer. To do this, we put the servo on top of one of the long rails of the robot. We then found a 4-sided piece that looks like a cage which fit perfectly around our servo. We screwed this into the rail on our robot. Once we had this part sturdy. We started by attaching a thin slate on top of the servo which had holes in it. We screwed this down into the servo and also screwed the red part of the potentiometer into it. This was so the red part of the potentiometer wouldn't move when scanning. Once this was in place, we put the rod in the white part of the potentiometer. We then attached a protractor. We had the protractor face flat, and we attached it to the potentiometer. We used 2 twist-ties to hold it in place. The protractor was perpendicular to the rod also. We also attached a twist tie to the rod so that when the servo moved positions, we could accurately see the angle that the servo position was currently at.

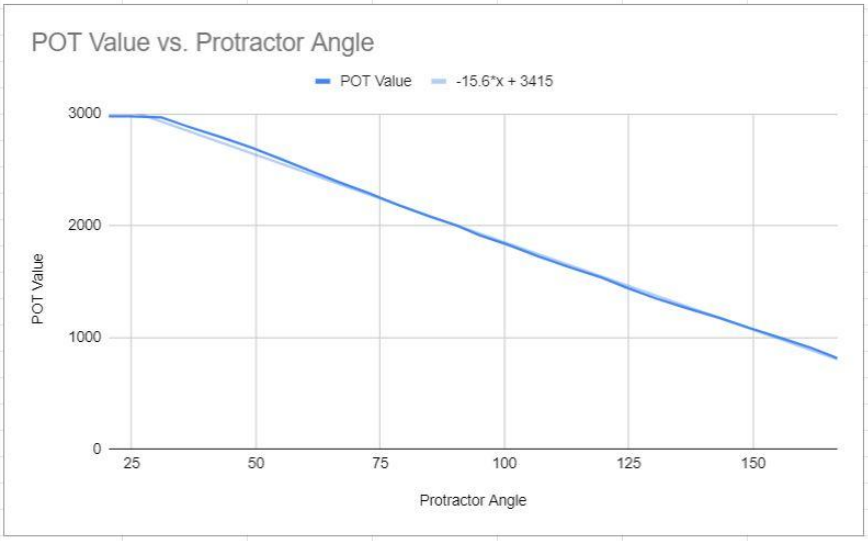


The only change we made to our code was increasing the wait time after each time the servo moved. We made this 3 seconds so we had enough time to get the correct angle. We took 3 rounds of data. We measured the protractor angle and potentiometer value at each servo position. We did this in increments of 10 from position -127 to 127. The results we got for this are in the picture below.

Servo Position	Protractor Angle	POT Value	Protractor Angle	POT Value	Protractor Angle	POT Value
-127	21	2977	20	2983	20	2968
-117	25	2979	25	2982	24	2969
-107	32	2955	31	2980	30	2970
-97	38	2873	36	2889	35	2894
-87	44	2775	42	2787	44	2794
-77	50	2685	48	2692	50	2703
-67	56	2578	55	2584	55	2606
-57	63	2460	60	2485	61	2500
-47	67	2368	67	2386	67	2394
-37	73	2276	73	2294	72	2302
-27	79	2172	79	2181	78	2194
-17	85	2076	85	2083	84	2094
-7	91	1989	90	2002	90	2014
3	95	1897	95	1915	95	1925
13	101	1813	100	1824	102	1831
23	106	1718	106	1722	108	1740
33	112	1617	113	1629	114	1637
43	120	1522	120	1531	119	1544
53	124	1432	125	1445	125	1451
63	131	1339	130	1351	130	1361
73	137	1248	136	1260	137	1273
83	142	1160	143	1170	145	1187
93	148	1074	150	1081	150	1093
103	155	983	156	992	156	1005
113	160	896	163	904	162	919
123	165	808	168	808	168	823

After we got these results, we took the average for the potentiometer value and protractor angle at each servo position. With the averages, we used linear regression to get an equation so we can find out the angle based on the potentiometer value and vice versa. The average values and graph are below.

Protractor Angle	POT Value	New POT Value
20	2976	3097.8
25	2977	3030.2
31	2968	2931.4
36	2885	2848.2
43	2785	2739
49	2693	2645.4
55	2589	2551.8
61	2482	2458.2
67	2383	2369.8
73	2291	2281.4
79	2182	2187.8
85	2084	2094.2
90	2002	2005.8
95	1912	1933
101	1823	1839.4
107	1727	1751
113	1628	1652.2
120	1532	1548.2
125	1443	1470.2
130	1350	1381.8
137	1260	1283
143	1172	1179
149	1083	1085.4
156	993	986.6
162	906	893
167	813	809.8



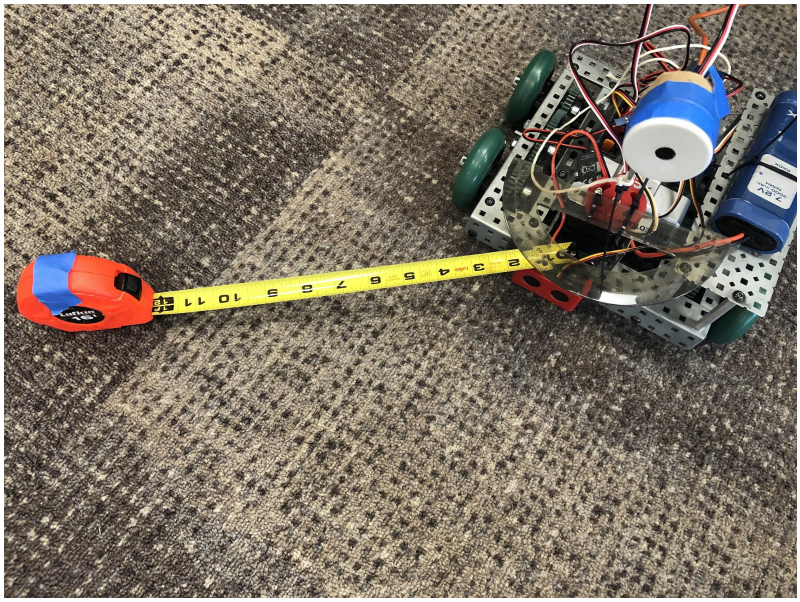
The equation we got was 3415 - 15.6*(protractor angle) = POT Value. For part B, we placed a light at 3 different angles, and for 2 different distances at each angle. We wanted to measure the accuracy of the light sensor. We picked angles 135, 90, and 30 degrees using the protractor. We then got the potentiometer reading of the strongest light value, and converted it to degrees using our equation from linear regression.

	Actual Angle (Protractor Angle)	POT Value (Device Angle where light is the brightest)	Angle(POT)	Final Angle (POT Linear Regression)
Angle 1 4 in	135	1115	155	147
	135	1478	135	124
	135	1560	125	119
Angle 1 8 in	135	1562	125	119
	135	1562	125	119
	135	1567	125	118
Angle 2 4 in	90	1930	100	95
	90	1930	100	95
	90	2113	85	83
Angle 2 8 in	90	2227	85	76
	90	2126	90	83
	90	2222	85	76
Angle 3 4 in	30	2959	35	29
	30	2909	40	32
	30	2913	40	32
Angle 3 8 in	30	2958	25	29
	30	2958	25	29
	30	2965	25	29

After collecting this data, we calculated the accuracy by taking the difference between actual angle (protractor angle) and expected angle (final POT after linear regression). We then squared these, summed them all up, and divided by 18 which was the total number of measurements. The average error we got was around 10 degrees.

Difference (Squared)	error for each angle+distance	
154.651545	13.30278799	
117.3611111		
258.8798488		
263.0218606	16.32548551	
263.0218606		
273.5207101		
26.96005917	5.676606898	
26.96005917		
42.75147929		
191.7159763	11.95829281	
54.34335963		
182.9429652		
0.5917159763	1.938683115	
5.933596318		
4.750164366		
0.4972057857	0.8804870845	
0.4972057857		
1.331360947		
10.19185973		

Some things that could have led to error are that we held the phone by hand so it could have moved by a couple of degrees which could have made a big difference. Another thing that could've caused error is surrounding light. When we first got our results, we had to redo them because we had the squarebot on top of the table and the light from the windows was affecting our results. We had to move our squarebot to the ground and block it from the light. For part C, we repeated the same process at the same angles but we used a 1 inch and 2 inch enclosure. We noticed a major difference in the results. With the 1 inch enclosure, we improved to an average error of 4.9 degrees, and there were no outliers in the data like in part b. With the 2 inch enclosure, the average error was 3.8 degrees. Almost all of the measurements were within a couple degrees of what was expected. We noticed a major difference because the enclosures block out a lot of extra light so it is focused on a more narrow path. When there is an enclosure, only lights that are really strong will be noticed. This makes sense because in part a we were getting data all over the place and we had to try and block out some of the light ourselves.



We used a tape measure to make sure we had the correct angle. We tried to use a string and attach it to the rod, but we didn’t want that to slow down the servo. If that worked though, we feel like that would have been a little more accurate.

Final Angle (POT Linear Regression)	Differnece (squared)	error for each angle+distance
139	17.36111111	3.485592273
139	17.89940828	
134	1.187541091	
140	21.30177515	3.721998973
139	19.56360947	
134	0.6944444444	
99	90.00657462	7.933625733
94	14.79289941	
99	84.02777778	
94	12.88625904	6.004805768
99	82.85667324	
94	12.43014464	
33	7.24852071	4.540133209
29	1.187541091	
37	53.40236686	
29	1.051939513	1.098506746
29	0.924556213	
29	1.64365549	
average error (1 inch enclosure)	4.946754257	

Final Angle (POT Linear Regression)	Difference squared	error for each angle+distance
127	57.2156476	4.62368674
133	3.22156476	
133	3.698224852	
133	3.455785667	6.630097764
127	63.18211703	
127	65.23668639	
93	9.077087442	3.059736545
93	10.68786982	
93	8.321005917	
87	10.27284681	3.183904087
87	9.866042078	
87	10.27284681	
29	1.64365549	0.9882334256
29	0.5917159763	
29	0.6944444444	
29	1.187541091	0.9861522016
29	0.80539119	
29	0.924556213	
average error (2 inch enclosure)	3.803178711	

After all of the testing, it was clear that it would be best to use the 2 inch enclosure for the rest of the assignment.

Phototaxis

For part 3, we didn't change any of the mechanical design of the robot except that we left the 2 inch enclosure attached to the light sensor. For the software design, our robot starts by scanning for light. It gets the strongest light value and servo position. If this light value is greater than 700, the robot will randomly wander. We used 700 because that was a value we determined for when the robot doesn't sense any significant light. The robot randomly wanders

by spinning the left motor for a second, and then going straight forward for 2 seconds. If the light value is less than 700, the robot doesn't randomly wander and instead runs through a series of if else statements where it compares the angle of the strongest light. If the angle is between 80 and 100, the robot goes straight. Between 100 and 135, it goes slightly left. Between 45 and 80, it goes slightly right. Greater than 135, it goes hard left. And less than 45, it goes hard right. The robot travels in one of these 5 positions for 2 seconds. Some strengths to our approach if the random wandering. Given how the robot spins, if there is a light source, the robot will eventually find it since it will rotate 360 degrees after a series of spins. It might take some time, but it will find the light source. Another is that our robot travels for 2 seconds so it makes significant progress each time it moves. Some weaknesses though are that it only travels in 5 directions and not more. It's possible that the robot can travel too far and actually lose the light after finding it. If it were to take smaller steps and travel in a more accurate direction, this would never happen. While they are definitely some weaknesses to our design, it will find the light source eventually. It may not be the most efficient, but it works every time.

Conclusion

Overall, we were happy with our performance of the robot. It wasn't very quick, but it made its way to the light source in a good amount of time. Both times, we started out by having the robot face away from the light. It would take a turn, move forward, stop, and then scan. It did this a couple times until it found the light source. Once it did, the robot traveled towards the light for 2 seconds, stopped, and scanned. After initially finding the light source, it took roughly less than 10 stops to get to the light source. It did get to the light source, but we still feel like there could be improvements. Instead of moving forward for 2 seconds, we should move that to be 1 second or half a second. By moving for 2 seconds straight, it is possible that our squarebot can go too far and lose the light source. Another thing we would improve is to make the rod which holds the light sensor and potentiometer a lot smaller. The smaller it is, the faster the rod can turn. This would allow us to scan for light more frequently because we can compensate with a shorter scanning time. Once we calibrated our light sensor and potentiometer, it was too late to change this aspect of our robot. One more thing we could improve is to make our robot travel in 10 different directions instead of 5. This would just make the movement more accurate. With our current solution, if the light angle is between 135 and 100, the robot would move the same way. 35 degrees is a lot so our robot didn't always move in the best direction, it just went in the general direction. That didn't cause any problems for our current solution, but if we were to improve our code we would definitely add this.

Appendix

Part 1

```
#pragma config(Sensor, in4, POT, sensorPotentiometer)
#pragma config(Sensor, in6,    light_sensor,   sensorReflection)
#pragma config(Motor,  port9,           Servo,         tmotorServoStandard, openLoop)
    //int sensor_value;
    //sensor_value = SensorValue(POT);
int ServoMove(int servoPosition){

  writeDebugStreamLine("Servo Position = %d",servoPosition);
  int light = SensorValue(light_sensor); //gets light sensor value
  writeDebugStreamLine("POT = %d",SensorValue(POT));
  motor[Servo] = servoPosition; //moves servo to current position
  wait1Msec(200);
  return light;
}

task main()
{
    int max_position = -127; //sets default max position to first position
    int max_light = 800; //sets default max light to 800 (lowest light value)
    bool b = true;
    if(b){
        for(int i = -127; i<127 ; i +=10){ //checks all values in servo range incrementing by 10
        ServoMove(i);
        if (ServoMove(i) < max_light){ //if current light value is greater than the max,
                                       //max light value becomes current light value and
                                        //max position value is changed to current position
            max_light = ServoMove(i);
            max_position = i;
        }
            }
        }
        motor[Servo]=max_position; //moves servo to position with the most light
        //ServoMove(max_position);
        wait1Msec(5000);
        writeDebugStreamLine("%d",max_position);
}
```

Part 2

```
#pragma config(Sensor, in1,    POT,            sensorPotentiometer)
#pragma config(Sensor, in6,    light_sensor,   sensorReflection)
#pragma config(Motor,  port9,           Servo,         tmotorServoStandard, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

int ServoMove(int servoPosition){

  writeDebugStreamLine("Servo Position = %d",servoPosition);
  int light = SensorValue(light_sensor); //gets light sensor value
  writeDebugStreamLine("Light = %d",light);
  motor[Servo] = servoPosition; //moves servo to current position
  wait1Msec(100);
  return light;
}

task main()
{
    int max_position = -127; //sets default max position to first position
    int max_light = 800; //sets default max light to 800 (lowest light value)
    bool b = true;
    if(b){
        for(int i = -127; i<127 ; i +=10){ //checks all values in servo range incrementing by 10
        ServoMove(i);
        if (ServoMove(i) < max_light){ //if current light value is greater than the max,
                                       //max light value becomes current light value and
                                       //max position value is changed to current position
            max_light = ServoMove(i);
            max_position = i;
        }
            }
        }
        motor[Servo]=max_position; //moves servo to position with the most light
        //ServoMove(max_position);
        wait1Msec(5000);
        writeDebugStreamLine("%d",max_position);
}
```

Part 3

```
#pragma config(Sensor, in4,    POT,            sensorPotentiometer)
#pragma config(Sensor, in6,    light_sensor,   sensorReflection)
#pragma config(Motor,  port2,           rightMotor,    tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port3,           leftMotor,     tmotorVex393_MC29, openLoop)
#pragma config(Motor,  port9,           Servo,         tmotorServoStandard, openLoop)
//*!!Code automatically generated by 'ROBOTC' configuration wizard               !!*//

//Use your active directionality sensing device from Part 2 to allow Squarebot to find and approach a light source.
//You may find it necessary to periodically rescan for the light direction, although it is also possible to make
//continuous corrections as your robot drives toward the light source. If it cannot find the light source, it should
//wander randomly until the light comes into view.



    //int sensor_value;
    //sensor_value = SensorValue(POT);

//Functions for different turns

void moveHardLeft(){
    motor[leftMotor] = -40;
    motor[rightMotor] = 40;
    wait1Msec(700);
    motor[leftMotor] = 40;
    motor[rightMotor] = 40;
    wait1Msec(2000);
    motor[leftMotor] = 0;
    motor[rightMotor] = 0;
}
void moveSoftLeft(){
    motor[leftMotor] = 20;
    motor[rightMotor] = 80;
    wait1Msec(300);
    motor[leftMotor] = 40;
    motor[rightMotor] = 40;
    wait1Msec(2000);
    motor[leftMotor] = 0;
    motor[rightMotor] = 0;
}
void moveStraight(){
    motor[leftMotor] = 50;
    motor[rightMotor] = 38;
    wait1Msec(4000);
    motor[leftMotor] = 0;
```

```
    motor[rightMotor] = 0;
}
void moveSoftRight(){
    motor[leftMotor] = 80;
    motor[rightMotor] = 20;
    wait1Msec(300);
    motor[leftMotor] = 40;
    motor[rightMotor] = 40;
    wait1Msec(2000);
    motor[leftMotor] = 0;
    motor[rightMotor] = 0;
}
void moveHardRight(){
    motor[leftMotor] = 40;
    motor[rightMotor] = -40;
    wait1Msec(700);
    motor[leftMotor] = 40;
    motor[rightMotor] = 40;
    wait1Msec(2000);
    motor[leftMotor] = 0;
    motor[rightMotor] = 0;
}


int ServoMove(int servoPosition){

  //writeDebugStreamLine("Servo Position = %d",servoPosition);
  int light = SensorValue(light_sensor); //gets light sensor value
  //writeDebugStreamLine("POT = %d",SensorValue(POT));
  motor[Servo] = servoPosition; //moves servo to current position
  wait1Msec(200);
  return light;
}

task main()
{
    while(true){
        //SCANNING STATE
        int max_position = -127; //sets default max position to first position
        int max_light = 800; //sets default max light to 800 (lowest light value)
        int max_POT = 3000;
        bool b = true;
        if(b){
            for(int i = -127; i<127 ; i +=10){ //checks all values in servo range incrementing by 10
```

```
    ServoMove(i);
    if (ServoMove(i) < max_light){ //if current light value is greater than the max,
                                   //max light value becomes current light value and
                                   //max position value is changed to current position
        max_light = ServoMove(i);
        max_position = i;
        max_POT = SensorValue(POT);
    }
        }
    }
    motor[Servo]=max_position; //moves servo to position with the most light
    //ServoMove(max_position);
    wait1Msec(2000);
    writeDebugStreamLine("%d",max_position);
    int angle = (max_POT - 3415)/-15.6;
    writeDebugStreamLine("%d, <- ANGLE",angle);
    //DECISION
    if (max_light > 700){
        //int time = (rand() % (2000 + 1 - 500)+500);
        int time = 1000;
        writeDebugStream("%d, TIME1", time);
        motor[leftMotor]= -50; //Reverses the direction that Squarebot is going so it backs up.
        motor[rightMotor]= 0;
        wait1Msec(time);//Makes Squarebot stop for a random amount of time.
        motor[leftMotor]= 0;
        motor[rightMotor]= 0;
        wait1Msec(500);


        int time2 = 2000;
        motor[leftMotor]= 50; //Reverses the direction that Squarebot is going so it backs up.
        motor[rightMotor]= 38;
        wait1Msec(time2);
        motor[leftMotor]= 0;
        motor[rightMotor]= 0;

            }else if((80 < angle) &&(angle <= 100)){//Threshold for different turn radius
                moveStraight();
            } else if ((100 < angle) &&(angle <= 135)) {
                moveSoftLeft();
            }else if (135 < angle) {
                moveHardLeft();
            }else if ((45 < angle) &&(angle <= 80)){
                moveSoftRight();
            }else if(angle <= 45) {
                moveHardRight();
            }else {
                writeDebugStreamLine("SOL");
            }
    }
}
```