MECH 452 - Mechatronics Engineering
Department of Mechanical and Materials Engineering
Faculty of Engineering and Applied Science, Queen's University, Kingston

Group #24
Project Stage 1- Neutralization of Point Radiation Source with a Mobile Robot

Student Names: Zach Elman and Cristiano Iocabelli
Instructor Name: Prof. Surgenor
Date Report Submitted: November 14, 2021

Summary:
A LynxBot and code from the previous lab programs Group9Lab6WallFollow and Group9Lab8Target, were used in this project stage. The project challenge required the LynxBot to perform a wall follow during the first step of the challenge and then perform a sweep of the area in order to locate one of the three lamps which would be turned on. The lamps were two, three, or four metres away from the sweeping location. The goal of the challenge was to locate the illuminated lamp, drive towards it and cover the lamp with the LynxBot blocker. Activities involved included, a) modifying a wall following program to incorporate two sharps in order to allow the LynxBot to follow the wall specific to this task, b) modifying a sweeping program to allow the LynxBot to detect which one of the three lamps was illuminated, and c) troubleshooting the LynxBot and analyzing data in-situ in order to program appropriate solutions to issues with detecting the four-metre light source.

Design

The LynxBot robot used for the trial had several main components. These components included an ardunio board, a six-volt battery, a twelve-volt battery, two motors for its sets of wheels, front bumper buttons, two sharps, and a photoresistor. Fig. 1 shows the entire LynxBot design used throughout the challenge. Fig. 2 is a close-up of one of the sharps that was placed at the front of the LynxBot in order to detect when it was close to the final wall during the wall following segment. It was also used as a second method of detection to stop the LynxBot once it was close enough to one of the three lamps. The second sharp used can be seen in Fig. 3. It was used for the wall following segment and was placed on an angle in order to detect turns more in advance. The photoresistor used can also be seen in Fig. 2 and was placed at the front of the LynxBot, stacked on 10 pieces of Lego. It was used for detecting the lamps during the challenge. Fig.4 displays the testing and challenge setup for the project.

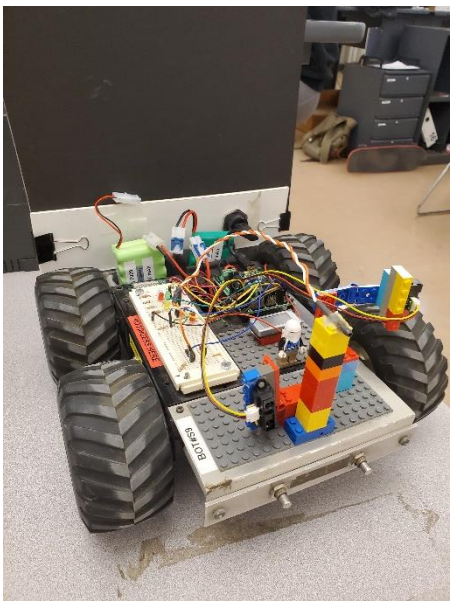

Fig. 1- Front view of the LynxBot design used during the challenge

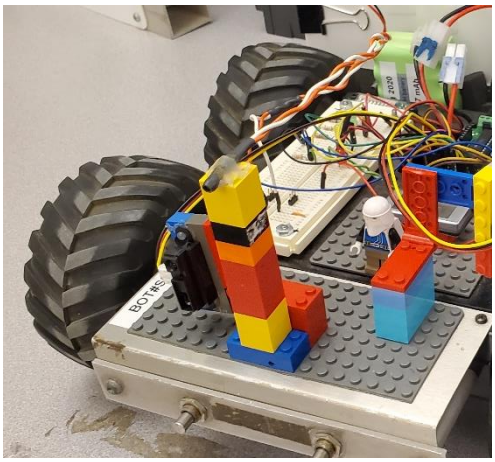

Fig. 2 – Close-up view of the front sharp and photoresistor used

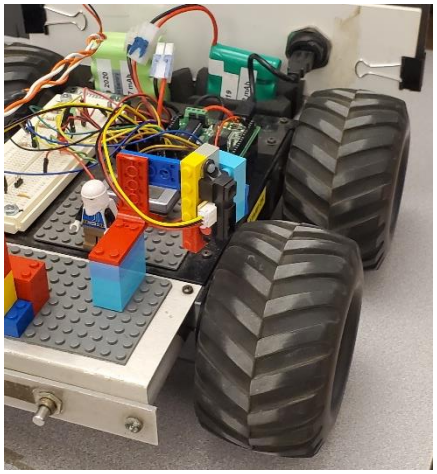

Fig. 3- Close up view of the angled sharp used for the wall follow

Fig. 4 – Stage 1 challenge setup with LynxBot at sweeping position

Program:

A listing of the program ***Group24Lab1IntroSwitch*** used in the lab is given in **Appendix A**. The flowchart corresponding to this program is given as **Fig. 5** . The program starts with the green LED flashing until the button is pressed which commences the wall following segment. During the wall following segment, the LynxBot completes two turns and then drives along the final straight-away until it reaches the sweeping area. Here, it completes a 180 degree sweep in order to locate the position of illuminated lamp. If lamp is four metres away, the LynxBot would rotate a fixed amount to improve its bearing to the lamp and would drive forward in order to complete its position correction sequence at a closer distance so that it is more accurate. It would then keep driving forward until the photoresistor detects a high enough brightness, or three seconds have passed. The code is then the same between the main branch and this offset branch which is why it connects back to the main segment in the flowchart. If the illuminated lamp was not detected to be the four metre lamp, the position correction sequence is completed. The LynxBot then advances forward until either the brightness level is high enough to indicate that the robot is close enough to complete the second sweep, or after two seconds have passed. At this point the LynxBot completes its sweep to improve its bearing and then it begins its bearing or position correction sequence in case the sweep was slightly off. After this, the LynxBot continues to advance forward until either the brightness is high enough to indicate that it is at the final location, or once the sharp detects that the LynxBot is close to the box behind the lamp. Once this is achieved, the LynxBot will pause for 2 seconds, turn 180 degrees counterclockwise, and move backwards for 0.5 seconds to block the light. The program will end after this point.

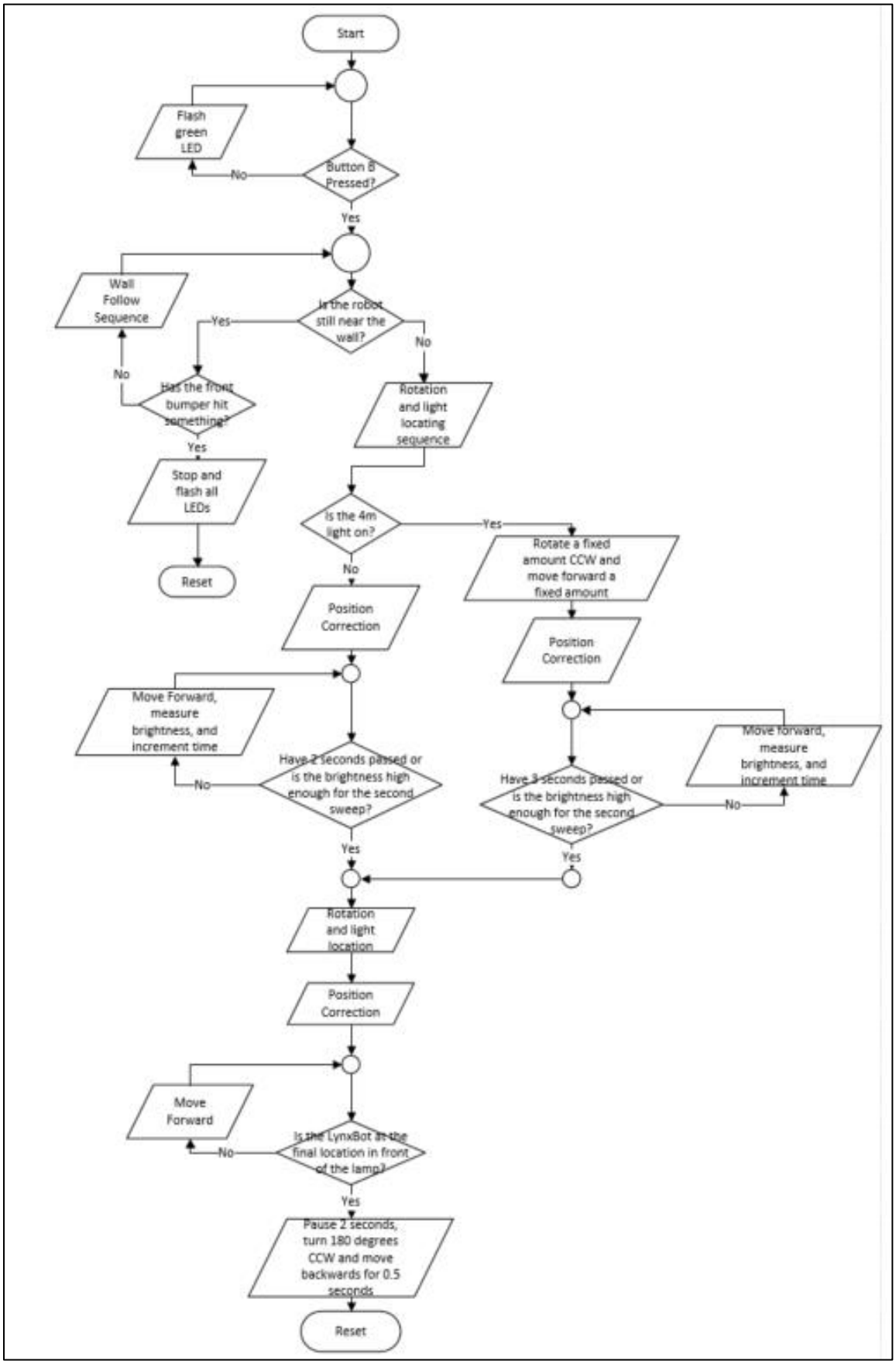

Fig. 5- Flowchart of the program used in stage 1 of the project

Results*:*

i) Beginning with the wall following sequence, code from the ***Group9Lab6WallFollow*** program was used and it worked well before the second turn without any adjustments. The first challenge was to turn right, since the previous program only worked for turning left. To solve this problem, a forward-facing sharp was added and used to detect when the LynxBot should begin to turn right. At first, open loop control was used where the robot would turn right for an arbitrary delay and then return to the wall following code. This was changed since the turning amount would differ greatly based on the battery voltage. To close the loop, the final code had the robot turning right until the forward-facing sharp did not read the wall anymore, where it would then return to the wall following code.
The next issue faced was that the light sensing code from the previous lab was not working well on the initial sweep. It was determined that the issue was because there was inconsistency between the sweeps to the right and the left. The sweeps to the right (the initial sweep) would often over rotate. To solve this issue an open-loop counterclockwise turn was added to compensate for the over rotation.
The last and most difficult issue faced was for the 4 metre light. The photoresistor was not able to detect the location of the 4-metre light since it was very close to the brightness level of the ambient light. This was tested in-situ and it was even found that there was a location in between the 3 metre and 4 metre light that produced slightly higher brightness readings than the actual lamp. This was most likely due to the light from the window. To solve this issue, an extra open loop clockwise turn was added to get closer to the correct bearing. This would only execute if the max brightness level recorded was below a certain threshold throughout the sweep which would indicate that the 4-metre light was on. The LynxBot would then drive forwards for a fixed amount of time to get closer to the light to complete its position correction sequence at a distance where it would be able to detect the light. After this position correction sequence, the LynxBot would then drive forward for 3 seconds (instead of 2 seconds for the other lights) or until the brightness reading reached a high enough level to complete its second sweep. This allowed it to get closer to the light before it would do its second sweep where the light was easily detected. This worked well, allowing for many successful runs, including two successful ones for the test.

ii) One of the teams had a very good strategy for the challenge which is currently being considered for stage 2 of the project. The method used a continuous sweeping method with the servomotor where the LynxBot would continuously correct its bearing as it was driving forward. This is a great solution as it avoids open-loop control since the LynxBot is constantly correcting itself according to the bearing relative to the light. The biggest advantage is that it eliminates the risk of a sweep being slightly off since it can correct itself shortly after. This was a regular problem that was experienced throughout the testing of the current program since the program only had two sweeps and thus only two chances to improve its bearing. The photoresistor did experience noisy readings and would sometimes send the LynxBot on a bearing quite off from the lamp. If this happened for the 3 metre or 4 metre light, the error would be propagated significantly as it moved forward. This would further affect the accuracy of the second sweep since the LynxBot would be facing the light on a very large angle. Therefore, this constant sweeping method is currently being investigated for stage 2 as it worked very well, achieving the other team a perfect score and the fastest time. Additionally, there was another team that would measure the angle of the lamp relative to the robot at the final step and adjust the amount that the robot would rotate in order to block the lamp. This proved to be more successful and allows for a

more precise method of blocking the light. This will be investigated for the stage 2 project as well to avoid the use of an arbitrary fixed rotation amount currently used to block the light.

iii) The main lesson learned from this challenge was that open loop control should be avoided. The inconsistency that comes with using open loop control adds time for calibration and only works at an ideal voltage level whereas closed loop control should only need to be calibrated once. Closed loop control greatly reduces the risk of having noisy hardware which can result in errors that would be propagated significantly by only occasionally being fixed in an open loop system. A more precise method of control will also be investigated for the light blocking sequence of the trial since it will help to eliminate different turning inconsistencies experienced due to changes in battery voltage. Finally, the team learned that functions should be used more in order to make the program more organized and easier to troubleshoot since the program became extremely long throughout this phase. It often made it challenging to find certain parts within the code that needed to be changed since there were so many lines of code.

Appendix A - Program Listing*:*

```
/*****************************************
************************
Group9Stage1
Original by B. Surgenor, 23/10/2020
Modified by: Zach Elman and Cristiano
Iocabelli
(based on Group9Lab8Target)



Changes:

Added e-stop sequence during wall follow
which flashes all lights if bumper is
pressed

Added wall follow sequence

Added 4 metre lamp condition with
specific sequence involving additional
rotation and advance forward before
implementing the position correction
sequence.

Added sharp detection when LynxBot is
near lamp. (Lab 8 only relied on
photoresistor value)



*****************************************
************************/

// Pin Assignments
int RED = 4; //red LED Pin
int GRN = 5; //green LED Pin
int YLW = 6; //yellow LED Pin
//push button
int BUTTON_A = 7;
int BUTTON_B = 8;
int BUTTON_C = 9;
int MOTOR_L = 10; // left motor signal
int MOTOR_R = 11;
int servoPin = 12;
int LIGHT = A0; //photoresistor pin (can't
be lowercase a0)
int firstturn = 60;
int servoAngle = 0;
//sharp values
int SHARP1 = A1; // sharp input pin
int sensor1;
int SHARP2 = A2;
int sensor2;
float KP = 0.16;
// adjust stop speed and target distance as
appropriate
int WALL = 1300;
//global variables
float result; //A to D value from
photoresistor
float mvresult; //millivolt value for
photoresistor
float mvavg;
unsigned long time;
unsigned long start;
float maxangle;
float mvmax = 0;
int deltaf = 30;
int delta = 20;
int delta1= 20;
int deltat= 10;
int STOP_SPEED = 143;
int delta2 = 20;
float correctangle=0;
float middist=3100;
float mvtotal;
float mvtotal2;
int count;
int cond = 0;
int offset= 50;
int BUMPER = 13;
// Setup Routine
void setup()
{
// initialize led pins as outputs.
pinMode(GRN, OUTPUT);
pinMode(YLW, OUTPUT);
pinMode(RED, OUTPUT);

//initialize pins as inputs
pinMode(BUTTON_A, INPUT);
```

```
pinMode(BUTTON_B, INPUT);
pinMode(LIGHT, INPUT);
pinMode(MOTOR_L, OUTPUT);
pinMode(MOTOR_R, OUTPUT);
pinMode(BUMPER, INPUT);
//initialize serial printout
Serial.begin(9600);
Serial.println(4000); // set desired max
vertical scale
Serial.println(2700);

//Program Initalization
Serial.println("Program loaded.");
Serial.println("Press button to start.");
runMotors(0,0);
delay(50);
do {
toggleLED(GRN);
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
Serial.println(mvresult);
// Green LED flashing
}while(digitalRead(BUTTON_B)== HIGH);
Serial.println("Program running.");

delay(1000); // give user chance to get
ready before plotting starts after button
press
}
// Main Loop
void loop() {
delay(1000);
do{
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
digitalWrite(RED, HIGH);
digitalWrite(GRN, HIGH);
digitalWrite(YLW, HIGH);
delay(500);
digitalWrite(RED, LOW);
digitalWrite(GRN, LOW);
digitalWrite(YLW, LOW);
delay(1000);
}while(-1);
}
sensor1 =
map(analogRead(SHARP1),0,1023,0,5000);
sensor2 =
map(analogRead(SHARP2),0,1023,0,5000);
int error = WALL - sensor1;
int dummy = min(abs(error)*KP, 100);
int deltaV = deltaf*(100-dummy)/100;

if(error > 130){
KP=0.18;
turnOnLED(YLW);
runMotors(deltaV, deltaf);
}
else if(error < -90){ //too close to the wall
turnOnLED(RED);
KP =0.20;
runMotors(deltaf, deltaV);
}
else{ //desired distance
KP=0.16;
turnOnLED(GRN);
runMotors(deltaf+10,deltaf+10);
}
Serial.print(error);
Serial.print("\t");
Serial.print(dummy);
Serial.print("\t");
Serial.println(deltaV);


}while(sensor2<1300);
do{
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
digitalWrite(RED, HIGH);
digitalWrite(GRN, HIGH);
digitalWrite(YLW, HIGH);
delay(500);
digitalWrite(RED, LOW);
digitalWrite(GRN, LOW);
digitalWrite(YLW, LOW);
delay(1000);
```

```
}while(-1);
}
sensor2 =
map(analogRead(SHARP2),0,1023,0,5000);
runMotors(20,-20);
delay(10);

}while(sensor2>1000);
do{
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
digitalWrite(RED, HIGH);
digitalWrite(GRN, HIGH);
digitalWrite(YLW, HIGH);
delay(500);
digitalWrite(RED, LOW);
digitalWrite(GRN, LOW);
digitalWrite(YLW, LOW);
delay(1000);
}while(-1);
}
sensor1 =
map(analogRead(SHARP1),0,1023,0,5000);
sensor2 =
map(analogRead(SHARP2),0,1023,0,5000);
int error = WALL - sensor1;
int dummy = min(abs(error)*KP, 100);
int deltaV = deltaf*(100-dummy)/100;

if(error > 130){
KP=0.18;
turnOnLED(YLW);
runMotors(deltaV, deltaf);
}
else if(error < -90){ //too close to the wall
turnOnLED(RED);
KP =0.20;
runMotors(deltaf, deltaV);
}
else{ //desired distance
KP=0.16;
turnOnLED(GRN);
runMotors(deltaf+10,deltaf+10);
}
Serial.print(error);
Serial.print("\t");
Serial.print(dummy);
Serial.print("\t");
Serial.println(deltaV);


}while(sensor1>1000);
runMotors(delta1,delta1);
delay(1700);
runMotors(0,0);
delay(2000);


for(int ii = 0; ii < (firstturn*2); ii += 1){

result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
//convert value to millivolts
if(mvresult>mvmax){
mvmax=mvresult;
correctangle=ii;
}
Serial.println(mvresult);
runMotors(-delta1,delta2);
delay(10);
}
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}

do{
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
```

```
float mvresult2 = map(result, 0, 1024, 0,
5000);
float mvresult3 = map(result, 0, 1024, 0,
5000);
float mvresult4=map(result, 0, 1024, 0,
5000);
mvavg = (mvresult + mvresult2 +
mvresult3 + mvresult4)/4;
runMotors(delta1,-delta2);

delay(10);

Serial.println(mvavg);
// Serial.print(mvresult);
}while(abs(mvmax-mvavg)>offset);
Serial.println(mvmax);
if (mvmax < 2500){
cond = 1;
for(int ii = 0; ii < (15); ii += 1){
runMotors(delta,-delta);
delay(10);
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
}
for(int ii = 0; ii < (60); ii += 1){
runMotors(delta,delta);
delay(10);
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
}
}
do{
for(int i = 0; i < 12; i += 1){
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
mvtotal = mvresult + mvtotal;
runMotors(0,0);
delay(10);
}
runMotors(deltat,-deltat);
delay(200);
runMotors(0,0);
delay(300);
for(int i = 0; i < 12; i += 1){
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
mvtotal2 = mvresult + mvtotal2;
runMotors(0,0);
delay(10);

}
mvtotal = mvtotal/12;
mvtotal2 = mvtotal2/12;
}while(mvtotal2 > mvtotal - 5);
runMotors(-deltat,deltat);
delay(300);
if (correctangle<40){
for(int i = 0; i < 150; i += 1)
runMotors(delta1,delta2);
delay(10);
}
if(cond==1){
middist = middist + 200;
count = -100;
}
do{
result = analogRead(LIGHT);
runMotors(delta1,delta2);
delay(10);
count=count+1;
mvresult = map(result, 0, 1024, 0, 5000);
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
}while(mvresult<middist && count<200);
```

```
runMotors(0,0);
delay(500);
for(int ii = 0; ii < (firstturn); ii += 1){
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
//convert value to millivolts
runMotors(-delta1,delta2);
delay(10);
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
}
```

```
for(int ii = 0; ii < (firstturn*2); ii += 1){

result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
//convert value to millivolts
if(mvresult>mvmax){
mvmax=mvresult;
correctangle=ii;
}
runMotors(delta1,-delta2);
delay(10);
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
}
do{
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
runMotors(-delta1,delta2);
}while(abs(mvmax-mvresult)>50);
```

```
do{
for(int i = 0; i < 12; i += 1){
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
mvtotal = mvresult + mvtotal;
runMotors(0,0);
delay(10);
}
runMotors(-delta,deltat);
delay(200);
runMotors(0,0);
delay(300);
for(int i = 0; i < 12; i += 1){
result = analogRead(LIGHT); //read the
value of photoresistor
mvresult = map(result, 0, 1024, 0, 5000);
mvtotal2 = mvresult + mvtotal2;
runMotors(0,0);
delay(10);
}
mvtotal = mvtotal/12;
mvtotal2 = mvtotal2/12;
}while(mvtotal2 > mvtotal - 5);
//runMotors(deltat,-deltat);
//delay(100);
//for(int ii = 0; ii < (firstturn*2-
correctangle); ii++){
//
// result = analogRead(LIGHT); //read the
value of photoresistor
// mvresult = map(result, 0, 1024, 0, 5000);
// runMotors(-delta1,delta2);
//
// delay(20);
//}
do{
sensor2 =
map(analogRead(SHARP2),0,1023,0,5000);
result = analogRead(LIGHT);
runMotors(delta1,delta2);
mvresult = map(result, 0, 1024, 0, 5000);
}while(mvresult<4400 && sensor2<900);
for(int i = 0; i < 50; i += 1){
runMotors(0,0);
```

```
delay(10);
if(digitalRead(BUMPER)== HIGH){
runMotors(0,0);
do{
turnOnLED(RED);
delay(500);
}while(-1);
}
}
for(int i = 0; i < 113; i += 1){
runMotors(delta1,-delta2);
delay(10);
}
for(int i = 0; i < 10; i += 1){
runMotors(0,0);
delay(10);
}
for(int i = 0; i < 40; i += 1){
runMotors(-delta1,-delta2);
delay(10);
}
runMotors(0,0);
turnOnLED(RED); // end of program
do{
toggleLED(RED);
}while(-1); // repeat forever
//}while(digitalRead(BUMPER) == HIGH);
runMotors(0,0);
turnOnLED(RED); // end of program
do{
toggleLED(RED);
}while(-1);
}
//********** FUNCTIONS (subroutines)
**************
//Turn on a single LED, and all other off
void turnOnLED(int COLOUR){
digitalWrite(GRN, LOW);
digitalWrite(YLW, LOW);
digitalWrite(RED, LOW);
digitalWrite(COLOUR, HIGH);
}
//Toggle an LED on/off
void toggleLED(int colour){
digitalWrite(colour, HIGH);
delay(125);
digitalWrite(colour, LOW);
delay(125);
}
void servoPulse(int servoPin, int myAngle){
int pulseWidth = (myAngle * 9.5) + 530;
digitalWrite(servoPin, HIGH); //set servo
high
delayMicroseconds(pulseWidth);
//microsecond pause
digitalWrite(servoPin, LOW);
}
void distanceLED(float mvResult){
if(mvResult > 3700){
turnOnLED(RED);
}
else if(mvResult < 3700 && mvResult >
3300){
turnOnLED(GRN);
}
else if(mvResult < 3100){
turnOnLED(YLW);
}
}
void runMotors(int delta_L, int delta_R){
int pulse_L = (STOP_SPEED + delta_L)*10;
//determines length of pulse in microsec
int pulse_R = (STOP_SPEED + delta_R)*10;
for(int i=0; i<3; i++){
pulseOut(MOTOR_L, pulse_L); //send pulse
to left motors
pulseOut(MOTOR_R, pulse_R); //send
pulse to right motors
}
}
void pulseOut(int motor, int pulsewidth){
digitalWrite(motor, HIGH);
delayMicroseconds(pulsewidth); //send
pulse of desired pulsewidth
digitalWrite(motor, LOW);
}
```