# Line Following Delivery Bot

---

By-
Avirat Varma - N17296397
Rishi Kavi - N15231052
Pavan Chowdary Cherukuri - N10938396

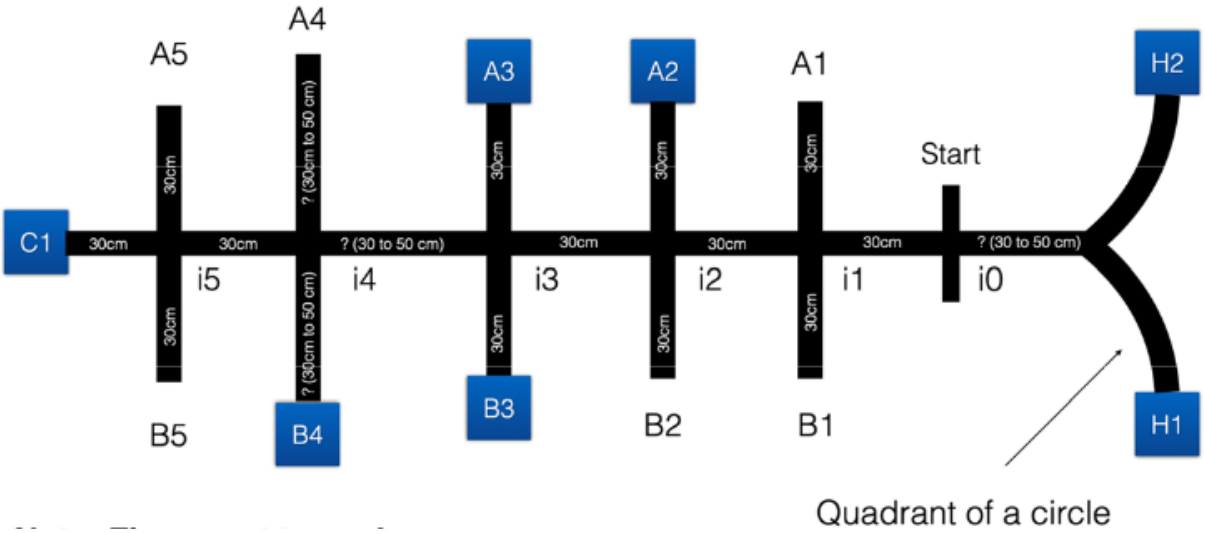
# Table of Contents


| | |
|---|---|
| INTRODUCTION | 3 |
| OBJECTIVES | 3 |
| METHODOLOGY | 4 |
| CIRCUIT DESIGN | 5 |
| MECHANICAL DESIGN | 6 |
| PSEUDO CODE | 7 |
| BILL OF MATERIALS | 10 |
| DEMONSTRATION | 11 |
| RESULTS | 11 |
| APPENDIX - CODE | 11 |

## INTRODUCTION

COVID pandemic has disrupted normal life. Given these unusual circumstances, there is a need for a contact less delivery system that can allow a COVID test kit provider to drop off the test kits at desired locations in a parking lot. This robot has been designed to perform a set of tasks to accomplish this goal.

## OBJECTIVES




- You will be directed to place your robot at one of the home locations (H1 or H2).
- Your robot must always follow the path (black tape).
- Starting from the assigned home position, the robot must first reach i0 (start position) and should NOT look around for an object (car) at the start position.
- Next, as the robot continues along the path, it must detect intersections and provide an indication of their detection by some means (e.g., LED, piezoelectric buzzer, LCD display, etc.).
- Upon arriving at an intersection (other than i0) the robot must look for objects (if any) on either side/ahead of the intersection.

- Upon detecting the object/objects, the robot must provide an indication of the object detection by some means (e.g., LED, piezoelectric buzzer, LCD display, etc.)
- The robot must then move towards the object to reach it, must stop for a few seconds within 5 to 8cms of the object, and must give an indication of reaching the object by some means (e.g., LED, piezoelectric buzzer, LCD display, etc.)

## METHODOLOGY

---

The key logic that decides the maneuvers of the bot is that it must, at all times, be centered on a black line. The IR array gives a value of 0 for minimal reflectance (white surface) and 2500 for maximum reflectance (black surface). The algorithm aims to sense the line using a comparison of the values from all sensors. Using this, at all hierarchies of the logic, the bot must be centered on the black line.

The IR sensor array was meticulously selected to ensure that the IR LEDs are equidistant at 0.95mm to avoid the array to bounce between minimum and maximum detection values i.e the IR LED is not on the edge of the tape. The bot then follows the behavioral hierarchy to do the following in a loop:

1. Follow the line
2. Detect and indicate intersections
3. Detect and indicate for objects at each intersection (except the first one)
4. Turn at the intersection towards the object detected
5. Move towards the object until 8cm apart
6. Indicate object reached
7. Take a U-turn
8. Return back to the original track
9. Repeat from Step-1
10. This iteration stops after 6 counts, i.e. after servicing the last intersection

Since our algorithm was efficient, the loops are executed very quickly and the adjustments to the control of the bot are done at a fast rate, thereby eliminating the need to add a PID controller to refine the maneuvers of the bot. The same objective calls for avoiding the use of any function

calls in our code which would consume more machine time and would have made the motion jittery.

LED Indications are as follows:

- Yellow LED- Intersection detected
- Green LED- Object Detected
- Red LED- Object Reached

## CIRCUIT DESIGN

---

List of components are as follows:

1. 3 HC-SR04 Ultrasonic Sensors
2. 2 Continuous Servo motors
3. 8-InfraRed Sensor Array
4. Arduino UNO R3
5. Arduino Shield & Bread Board
6. 2 - 6V(1.5Vx4) Battery packs
7. 3 Indicator LEDs

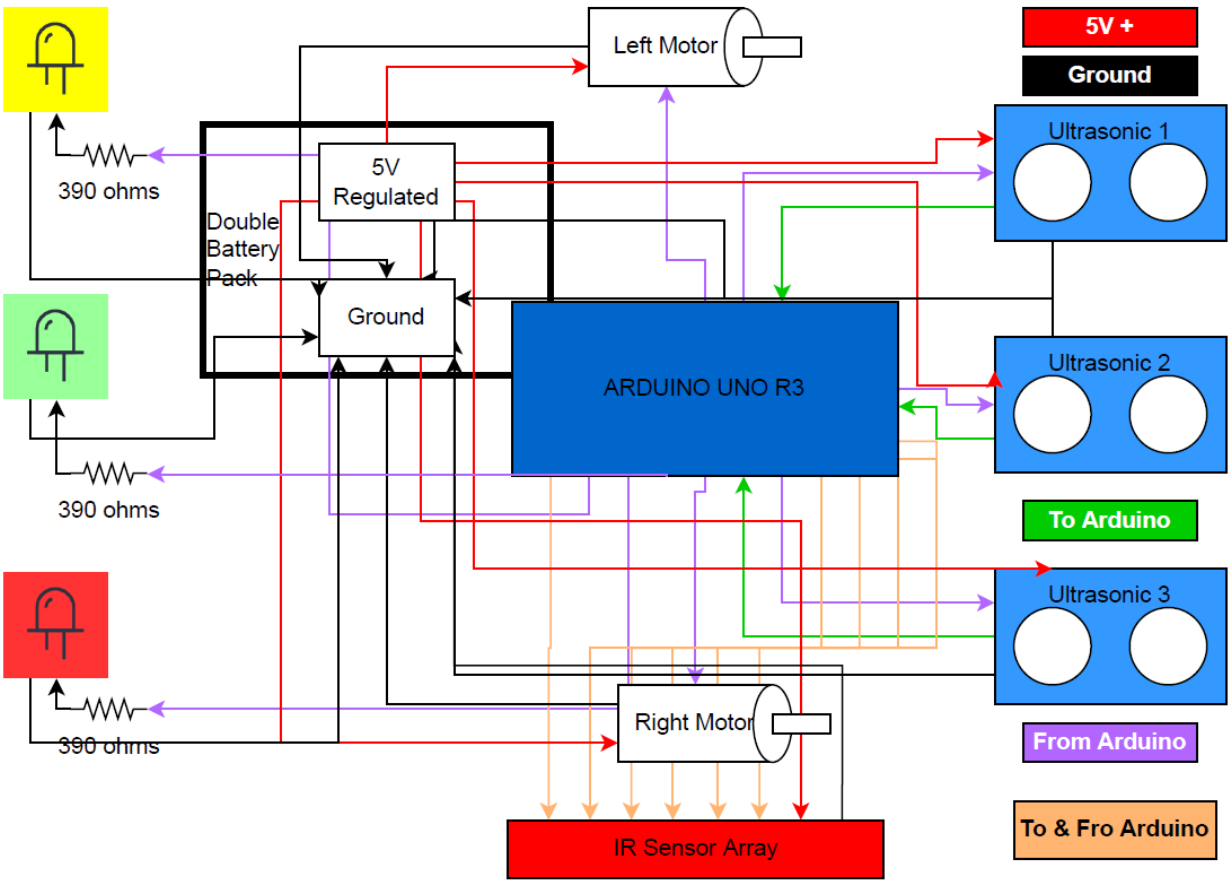



Figure 1: Circuit Design

## MECHANICAL DESIGN

The 3 Dimensional CAD model for the Sensor Housing, as seen in the Figure 2 and Figure 3 below was modeled in AutoDesk Fusion 360 software. This design was 3D printed using PLA 18 material which is lightweight and has considerable strength.

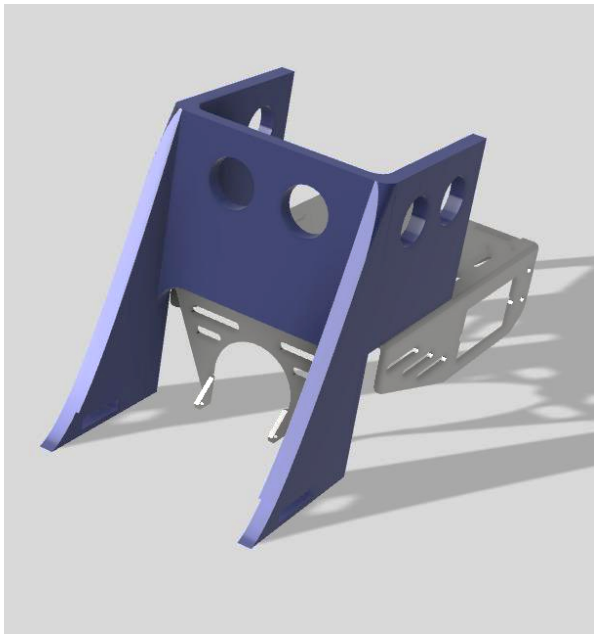


Figure 2: Isometric View

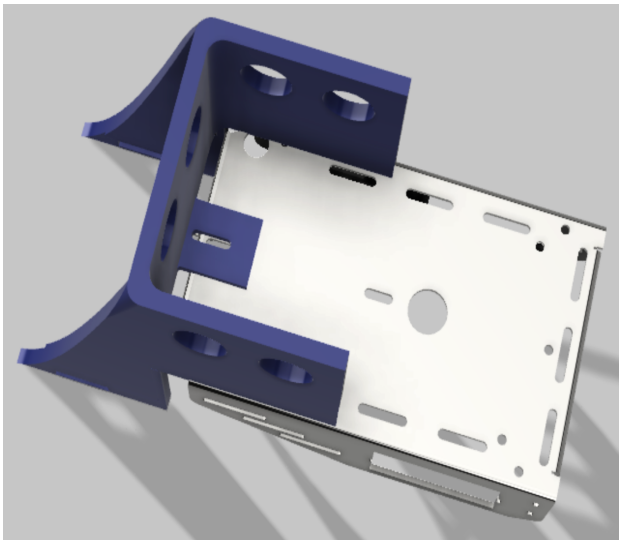


Figure 3: Top View

The Sensor Housing was aesthetically designed keeping in mind - aesthetic appeal and accurate positioning for the 3 HC-SR04 ultrasonic sensors and the IR sensor array in a combined housing design. The thickness and lengths of the design were optimized to provide the required strength and support. A groove is provided as shown in Figure 3 to mount onto the Boe-Bot Chassis. Two slots of thickness 4mm were provided at the bottom to hold the Infrared sensor array at an optimal height of 5mm from the ground.

## PSEUDO CODE

```
Define all IR Sensor Pins
Define al Ultrasonics Trigger and Echo Pins
Define Left & Right motor pins
Define LED indicator pins
Define Values delay values for Push, Right, Left & U-turn,

Initialise SensorValues array
Initialize Ultrasonic Ping values
Initialize Line Following index value and intersection counter (inter)
```

```
void setup()
{
  Setting pinMode for all pins;
  Serial.begin(9600);
}

void loop() {
  //Parent loop (if no intersection keep following line else...)
  Read sensorValues;
  Initialize variable 's' to 0;
  For(i=0; run through all 6 IR sensors)
  {
     s=s+sensorValues[i];
  }
  if(check if bot is on the line and number of intersections<7)
  {
    for(i=0; run through all 6 IR sensors)
    {
      check which IR sensor has minimum value (black line);
    }
   if(middle two sensors are on the line)
    {
      bot goes straight;
    }
    else if (line is on the left)
    {
      bot turns left;
    }
    else if (line is on the right)
    {
      bot turns right;
    }
  }
  else if (bot is in-between intersections 3 and 6)
  {
    increment intersection counter;
    glow yellow led to indication intersection;

    push the bot to align ultrasonic sensors;

      check for object on the left;
      check for object on the right;

    if(object is detected on the left)
    {
      glow green led to indicate object;
        push to align wheels;
```

```
        turn left;
    move close to the object using line following;
      glow red led to indicate object reached;
    take U-turn;
    back up;
    move forward to the intersection from where it took a left;
      push to move beyond intersection;
      turn left;
  }

  if(object is detected on the right)
  {
    glow green led to indicate object;
      push to align wheels;
      turn right;
    move close to the object using line following;
      glow red led to indicate object reached;
    take U-turn;
    back up;
    move forward to the intersection from where it took a right;
      push to move beyond intersection;
      turn right;
  }

  if(object is detected on both sides)
  {
    glow green led to indicate object;
      push to align wheels;
      turn left;
    move close to the object using line following;
      glow red led to indicate object reached;
    take U-turn;
    back up;
    move forward to the intersection from where it took a turn;
      push to move beyond intersection;
    use front ultrasonic to move to other object using line following;
      glow red led to indicate object reached;
    take U-turn;
    back up;
    move forward to initial intersection;
      push to align;
      turn right;
      push to move beyond intersection;
  }
}

if (bot has reached 6 intersections)
```

```
  {
    increment intersection counter;
    if(object detected in front)
    {
      glow green led to indicate object;
      move close to the object using line following;
      glow red led to indicate object reached;
      take U-turn;
    }
  }
  else if (to ignore false intersection detection)
  {
    if (check if time elapsed is enough for detection)
    {
      increment intersection counter;
      glow yellow led to indicate intersection;
    }
      push to align wheels;
  }
}
```

## BILL OF MATERIALS

---

| Component | Cost/Unit | Quantity | Cost |
|---|---|---|---|
| Boe-Bot Chassis | $70 | 1 | $70 |
| Parallax Continuous Servo Motor | $15 | 2 | $30 |
| Arduino UNO R3 | $20 | 1 | $20 |
| HC-SR04 Ultrasonic Sensor | $2 | 3 | $6 |
| Pololu QTR-8RC IR Sensor Array | $15 | 1 | $15 |
| Custom Designed Sensor Holder | $0 | 1 | $0 |
| Arduino Shield | $2 | 1 | $2 |
| Battery Pack | $7 | 2 | $14 |
| Duracell Battery | $1.375 | 8 | $11 |

| | | | |
|---|---|---|---|
| LED | $0.2 | 3 | $0.6 |
| Switch | $0 | 1 | $0 |
| **Total Cost** | | | **$168.6** |

## DEMONSTRATION

Google Drive Links:

**Demo 1**

**Demo 2**

The two demo videos have different placement of objects on the track.

## RESULTS

1. The bot successfully follows a black line
2. The bot can successfully detect intersections
3. The bot can successfully detect objects using ultrasonic sensors at the intersections
4. The bot successfully stops after servicing all intersections
5. Changing of batteries can vary the speed and turning calibrations of the robot
6. The bot successfully gives indications for results 2 & 3 using 3 LEDs

## APPENDIX - CODE

```
#include <QTRSensors.h>
#define ult1 12 //left ultrasonic trigger
#define ule1 13 //left ultrasonic echo
#define ult2 11 // center ultrasonic tigger
#define ule2 10 //center ultrasonic echo
#define ult3 8  //right ultrasonic trigger
#define ule3 9  //right ultrasonic echo
#define mleft A4 //left motor signal line
#define mright A3 //right motor signal line
#define i2 7   //IR sensor 2
#define i3 6   //IR sensor 3
#define i4 5   //IR sensor 4
#define i5 4   //IR sensor 5
```

```
#define i6 3   //IR sensor 6
#define i7 2   //IR sensor 7
#define l1 A0   //LED 1
#define l2 A1  //LED 2
#define l3 A2  //LED 3
#define sp 40
#define sp2 60
#define left 14
#define right 19
#define uturn 30
#define push 7
#define rev 8
#define thresh 10000
#define wait 5000


QTRSensors qtr;
unsigned int dl, dr, dc, duration1, duration2, duration3;
const int SensorCount = 6;
unsigned int sensorValues[SensorCount];
int maxind = 3;
int inter = 0;


void setup()
{
 qtr.setTypeRC();

 qtr.setSensorPins((const uint8_t[]) {
  7, 6, 5, 4, 3, 2
 }, SensorCount);

 DDRC |= B00011111;
 pinMode(ult1, OUTPUT);
 pinMode(ult2, OUTPUT);
 pinMode(ult3, OUTPUT);
 Serial.begin(9600);
}

void loop() {
 //father loop (if no intersection keep following line else...)
 qtr.read(sensorValues);

 int s = 0;
 for (int i = 0; i < 6; i++)
 {
  s += sensorValues[i];
 }
 if (s < 12800 && s > 3000 && inter < 7)
 {
  int bleh = 0;
  for (int i = 0; i < 6; i++)
  {

   if (sensorValues[i] > bleh)
   {
```

```
    bleh = sensorValues[i];

    maxind = i;
   }
  }

  if (maxind == 2 || maxind == 3) //if true, robot moves straight
  {
   digitalWrite(mright, HIGH);
   delay(0.6);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(2.3);
   digitalWrite(mleft, LOW);
   delay(sp);
  }
  else if (maxind == 0 || maxind == 1) //if true, robot turns left
  { for (int r = 0; r < 1; r++)
    {
     digitalWrite(mright, HIGH);
     delay(0.6);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
     delay(1.5);
     digitalWrite(mleft, LOW);
     delay(sp);
    }
  }
  else if (maxind == 4 || maxind == 5) //if true, robot turns right
  { for (int r = 0; r < 1; r++)
    {
     digitalWrite(mright, HIGH);
     delay(1.5);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
     delay(2.3);
     digitalWrite(mleft, LOW);
     delay(sp);
    }
  }
 }
 // if robot is at intersections 3 to 6
 else if (inter > 0 && inter < 6 && s > 3000)
 {
  inter++;
  //glow led1
  digitalWrite(l1, HIGH);
  delay(2000);
  digitalWrite(l1, LOW);

  //push to align ultrasonic
  for (int r = 0; r < push; r++)
  {
   digitalWrite(mright, HIGH);
   delay(0.6);
   digitalWrite(mright, LOW);
```

```
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
}
//check for object to left
delay(1000);
digitalWrite(ult1, LOW);
delayMicroseconds(2);
digitalWrite(ult1, HIGH);
delayMicroseconds(10);
digitalWrite(ult1, LOW);
int duration2 = pulseIn(ule1, HIGH);
dl = (duration2 * .0343) / 2;

//check for object to right
digitalWrite(ult3, LOW);
delayMicroseconds(2);
digitalWrite(ult3, HIGH);
delayMicroseconds(10);
digitalWrite(ult3, LOW);
int duration3 = pulseIn(ule3, HIGH);
dr = (duration3 * .0343) / 2;

//if object detected on left side
if (dl < 50 && dr > 50)
{
  //turn left and move close to object. uturn and move till the intersection and turn left and push
  //glow led2
  digitalWrite(l2, HIGH);
  delay(2000);
  digitalWrite(l2, LOW);

  //push to align wheels
  for (int r = 0; r < push; r++)
  {
    digitalWrite(mright, HIGH);
    delay(0.6);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(2.3);
    digitalWrite(mleft, LOW);
    delay(sp);
  }
  delay(1000);
  //turn left
  for (int r = 0; r < left; r++)
  {
    digitalWrite(mright, HIGH);
    delay(0.6);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(0.6);
    digitalWrite(mleft, LOW);
    delay(sp);
  }
```

```
delay(1000);
//move close to object
delay(1000);
digitalWrite(ult2, LOW);
delayMicroseconds(2);
digitalWrite(ult2, HIGH);
delayMicroseconds(10);
digitalWrite(ult2, LOW);
int duration1 = pulseIn(ule2, HIGH);
dc = (duration1 * .0343) / 2;

while (dc < 30 && dc > 8)
{
 qtr.read(sensorValues);

 int blah = 0;
 for (int i = 0; i < 6; i++)
 {

  if (sensorValues[i] > blah)
  {
   blah = sensorValues[i];

   maxind = i;
  }
 }

 if (maxind == 2 || maxind == 3) //if true, robot moves straight
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 else if (maxind == 0 || maxind == 1) //if true, robot turns left
 {
  for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(0.6);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(1.5);
   digitalWrite(mleft, LOW);
   delay(sp);
  }
 }
 else if (maxind == 4 || maxind == 5) //if true, robot turns right
 { for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(1.5);
   digitalWrite(mright, LOW);
```

```
      digitalWrite(mleft, HIGH);
      delay(2.3);
      digitalWrite(mleft, LOW);
      delay(sp);
    }
  }
  digitalWrite(ult2, LOW);
  delayMicroseconds(2);
  digitalWrite(ult2, HIGH);
  delayMicroseconds(10);
  digitalWrite(ult2, LOW);
  int duration1 = pulseIn(ule2, HIGH);
  dc = (duration1 * .0343) / 2;
}
//glow led3 after reaching object
digitalWrite(l3, HIGH);
delay(2000);
digitalWrite(l3, LOW);

//take uturn
for (int r = 0; r < uturn; r++)
{
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
delay(1000);
//back up
for (int r = 0; r < rev; r++)
{
  digitalWrite(mright, HIGH);
  delay(2.4);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.7);
  digitalWrite(mleft, LOW);
  delay(sp);
}
//move till intersection
qtr.read(sensorValues);

int s = 0;
for (int i = 0; i < 6; i++)
{
  s += sensorValues[i];
}
while (s < thresh && s > 3000)
{
  int bleh = 0;
  for (int i = 0; i < 6; i++)
  {
```

```
  if (sensorValues[i] > bleh)
  {
   bleh = sensorValues[i];

   maxind = i;
  }
 }

 if (maxind == 2 || maxind == 3) //if true, robot moves straight
 {
  digitalWrite(mright, HIGH);
  delay(0.9);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.1);
  digitalWrite(mleft, LOW);
  delay(sp2);
 }
 else if (maxind == 0 || maxind == 1) //if true, robot turns left
 { for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(0.9);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(1.5);
   digitalWrite(mleft, LOW);
   delay(sp2);
  }
 }
 else if (maxind == 4 || maxind == 5) //if true, robot turns right
 { for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(1.5);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(2.1);
   digitalWrite(mleft, LOW);
   delay(sp2);
  }
 }

 qtr.read(sensorValues);

 s = 0;
 for (int i = 0; i < 6; i++)
 {
  s += sensorValues[i];
 }
}
delay(1000);
//push to align for left turn
for (int r = 0; r < (push * 2); r++)
{
 digitalWrite(mright, HIGH);
```

```
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 delay(1000);
 //turn left
 for (int r = 0; r < left; r++)
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 delay(1000);
 //push to move beyond intersection
 for (int r = 0; r < push; r++)
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }

}
//if object detected on right side
else if (dr < 50 && dl > 50)
{
 //glow led2 turn right and move close to object. uturn and move till the intersection and turn right and push
 //glow led2
 digitalWrite(l2, HIGH);
 delay(2000);
 digitalWrite(l2, LOW);

 //push to align for right turn
 for (int r = 0; r < push; r++)
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 delay(1000);
 //turn right
 for (int r = 0; r < right; r++)
```

```
{
 digitalWrite(mright, HIGH);
 delay(2.4);
 digitalWrite(mright, LOW);
 digitalWrite(mleft, HIGH);
 delay(2.3);
 digitalWrite(mleft, LOW);
 delay(sp);
}
delay(1000);

//move close to object
digitalWrite(ult2, LOW);
delayMicroseconds(2);
digitalWrite(ult2, HIGH);
delayMicroseconds(10);
digitalWrite(ult2, LOW);
int duration1 = pulseIn(ule2, HIGH);
dc = (duration1 * .0343) / 2;

while (dc < 40 && dc > 8)
{
 qtr.read(sensorValues);

 int blah = 0;
 for (int i = 0; i < 6; i++)
 {

  if (sensorValues[i] > blah)
  {
   blah = sensorValues[i];

   maxind = i;
  }
 }

 if (maxind == 2 || maxind == 3) //if true, robot moves straight
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 else if (maxind == 0 || maxind == 1) //if true, robot turns left
 {
  for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(0.6);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(1.5);
   digitalWrite(mleft, LOW);
```

```
      delay(sp);
    }
   }
   else if (maxind == 4 || maxind == 5) //if true, robot turns right
   { for (int r = 0; r < 1; r++)
    {
     digitalWrite(mright, HIGH);
     delay(1.5);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
     delay(2.3);
     digitalWrite(mleft, LOW);
     delay(sp);
    }
   }
   digitalWrite(ult2, LOW);
   delayMicroseconds(2);
   digitalWrite(ult2, HIGH);
   delayMicroseconds(10);
   digitalWrite(ult2, LOW);
   int duration1 = pulseIn(ule2, HIGH);
   dc = (duration1 * .0343) / 2;
}
//glow led3 after reaching object
digitalWrite(l3, HIGH);
delay(2000);
digitalWrite(l3, LOW);

//take uturn
for (int r = 0; r < uturn; r++)
{
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
delay(1000);
//back up
for (int r = 0; r < rev; r++)
{
  digitalWrite(mright, HIGH);
  delay(2.4);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
//move till intersection
qtr.read(sensorValues);

int s = 0;
for (int i = 0; i < 6; i++)
```

```
{
 s += sensorValues[i];
}
while (s < thresh && s > 3000)
{
 int bleh = 0;
 for (int i = 0; i < 6; i++)
 {

  if (sensorValues[i] > bleh)
  {
   bleh = sensorValues[i];

   maxind = i;
  }
 }

 if (maxind == 2 || maxind == 3) //if true, robot moves straight
 {
  digitalWrite(mright, HIGH);
  delay(0.9);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.1);
  digitalWrite(mleft, LOW);
  delay(sp2);
 }
 else if (maxind == 0 || maxind == 1) //if true, robot turns left
 { for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(0.9);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(1.5);
   digitalWrite(mleft, LOW);
   delay(sp2);
  }
 }
 else if (maxind == 4 || maxind == 5) //if true, robot turns right
 { for (int r = 0; r < 1; r++)
  {
   digitalWrite(mright, HIGH);
   delay(1.5);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(2.1);
   digitalWrite(mleft, LOW);
   delay(sp2);
  }
 }
 qtr.read(sensorValues);

 s = 0;
 for (int i = 0; i < 6; i++)
 {
```

```
      s += sensorValues[i];
     }
    }
    delay(1000);
    //push to align for right turn
    for (int r = 0; r < (push * 2); r++)
    {
     digitalWrite(mright, HIGH);
     delay(0.6);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
     delay(2.3);
     digitalWrite(mleft, LOW);
     delay(sp);
    }
    delay(1000);
    //turn right
    for (int r = 0; r < right; r++)
    {
     digitalWrite(mright, HIGH);
     delay(2.4);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
     delay(2.3);
     digitalWrite(mleft, LOW);
     delay(sp);
    }
    delay(1000);
    //push to move beyond intersection
    for (int r = 0; r < push; r++)
    {
     digitalWrite(mright, HIGH);
     delay(0.6);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
     delay(2.3);
     digitalWrite(mleft, LOW);
     delay(sp);
    }
   }
   //if object detected on both side
   else if (dr < 50 && dl < 50)
   {
     //turn left and move close to object. uturn till the intersection. move straight and close to opp object. uturn and
move to intersection. turn right and push
    //glow led2
    digitalWrite(l2, HIGH);
    delay(2000);
    digitalWrite(l2, LOW);
    //push to align for left turn
    for (int r = 0; r < push; r++)
    {
     digitalWrite(mright, HIGH);
     delay(0.6);
     digitalWrite(mright, LOW);
     digitalWrite(mleft, HIGH);
```

```
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
delay(1000);

//turn left
for (int r = 0; r < left; r++)
{
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
delay(1000);

//move close to object
digitalWrite(ult2, LOW);
delayMicroseconds(2);
digitalWrite(ult2, HIGH);
delayMicroseconds(10);
digitalWrite(ult2, LOW);
int duration1 = pulseIn(ule2, HIGH);
dc = (duration1 * .0343) / 2;

while (dc < 40 && dc > 8)
{
  qtr.read(sensorValues);

  int blah = 0;
  for (int i = 0; i < 6; i++)
  {

    if (sensorValues[i] > blah)
    {
      blah = sensorValues[i];

      maxind = i;
    }
  }

  if (maxind == 2 || maxind == 3) //if true, robot moves straight
  {
    digitalWrite(mright, HIGH);
    delay(0.6);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(2.3);
    digitalWrite(mleft, LOW);
    delay(sp);
  }
  else if (maxind == 0 || maxind == 1) //if true, robot turns left
  {
```

```
   for (int r = 0; r < 1; r++)
   {
    digitalWrite(mright, HIGH);
    delay(0.6);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(1.5);
    digitalWrite(mleft, LOW);
    delay(sp);
   }
  }
  else if (maxind == 4 || maxind == 5) //if true, robot turns right
  { for (int r = 0; r < 1; r++)
   {
    digitalWrite(mright, HIGH);
    delay(1.5);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(2.3);
    digitalWrite(mleft, LOW);
    delay(sp);
   }
  }
  digitalWrite(ult2, LOW);
  delayMicroseconds(2);
  digitalWrite(ult2, HIGH);
  delayMicroseconds(10);
  digitalWrite(ult2, LOW);
  int duration1 = pulseIn(ule2, HIGH);
  dc = (duration1 * .0343) / 2;
}
//glow led3 after reaching object
digitalWrite(l3, HIGH);
delay(2000);
digitalWrite(l3, LOW);

//take uturn
for (int r = 0; r < uturn; r++)
{
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
delay(1000);
//back up
for (int r = 0; r < rev; r++)
{
  digitalWrite(mright, HIGH);
  delay(2.4);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
```

```
  digitalWrite(mleft, LOW);
  delay(sp);
}
//move till intersection
qtr.read(sensorValues);

int s = 0;
for (int i = 0; i < 6; i++)
{
  s += sensorValues[i];
}
while (s < thresh && s > 3000)
{
  int bleh = 0;
  for (int i = 0; i < 6; i++)
  {

    if (sensorValues[i] > bleh)
    {
      bleh = sensorValues[i];

      maxind = i;
    }
  }

  if (maxind == 2 || maxind == 3) //if true, robot moves straight
  {
    digitalWrite(mright, HIGH);
    delay(0.9);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(2.1);
    digitalWrite(mleft, LOW);
    delay(sp2);
  }
  else if (maxind == 0 || maxind == 1) //if true, robot turns left
  { for (int r = 0; r < 1; r++)
    {
      digitalWrite(mright, HIGH);
      delay(0.9);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(1.5);
      digitalWrite(mleft, LOW);
      delay(sp2);
    }
  }
  else if (maxind == 4 || maxind == 5) //if true, robot turns right
  { for (int r = 0; r < 1; r++)
    {
      digitalWrite(mright, HIGH);
      delay(1.5);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(2.1);
      digitalWrite(mleft, LOW);
```

```
    delay(sp2);
   }
  }
  qtr.read(sensorValues);

  s = 0;
  for (int i = 0; i < 6; i++)
  {
   s += sensorValues[i];
  }
}
delay(1000);
//push to move beyond intersection
for (int r = 0; r < push; r++)
{
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
}
delay(1000);
//move close to object
digitalWrite(ult2, LOW);
delayMicroseconds(2);
digitalWrite(ult2, HIGH);
delayMicroseconds(10);
digitalWrite(ult2, LOW);
duration1 = pulseIn(ule2, HIGH);
dc = (duration1 * .0343) / 2;

while (dc < 40 && dc > 8)
{
  qtr.read(sensorValues);

  int blah = 0;
  for (int i = 0; i < 6; i++)
  {

   if (sensorValues[i] > blah)
   {
    blah = sensorValues[i];

    maxind = i;
   }
  }

  if (maxind == 2 || maxind == 3) //if true, robot moves straight
  {
   digitalWrite(mright, HIGH);
   delay(0.6);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(2.3);
```

```
    digitalWrite(mleft, LOW);
    delay(sp);
  }
  else if (maxind == 0 || maxind == 1) //if true, robot turns left
  {
    for (int r = 0; r < 1; r++)
    {
      digitalWrite(mright, HIGH);
      delay(0.6);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(1.5);
      digitalWrite(mleft, LOW);
      delay(sp);
    }
  }
  else if (maxind == 4 || maxind == 5) //if true, robot turns right
  { for (int r = 0; r < 1; r++)
    {
      digitalWrite(mright, HIGH);
      delay(1.5);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(2.3);
      digitalWrite(mleft, LOW);
      delay(sp);
    }
  }
  digitalWrite(ult2, LOW);
  delayMicroseconds(2);
  digitalWrite(ult2, HIGH);
  delayMicroseconds(10);
  digitalWrite(ult2, LOW);
  int duration1 = pulseIn(ule2, HIGH);
  dc = (duration1 * .0343) / 2;
}
//glow led3 after reaching object
digitalWrite(l3, HIGH);
delay(2000);
digitalWrite(l3, LOW);

//take uturn
for (int r = 0; r < uturn; r++)
{
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
delay(1000);
//back up
for (int r = 0; r < rev; r++)
{
```

```
  digitalWrite(mright, HIGH);
  delay(2.4);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(0.6);
  digitalWrite(mleft, LOW);
  delay(sp);
}
//move till intersection
qtr.read(sensorValues);

s = 0;
for (int i = 0; i < 6; i++)

{
  s += sensorValues[i];
}
while (s < thresh   && s > 3000)
{
  int bleh = 0;
  for (int i = 0; i < 6; i++)
  {

    if (sensorValues[i] > bleh)
    {
      bleh = sensorValues[i];

      maxind = i;
    }
  }

  if (maxind == 2 || maxind == 3) //if true, robot moves straight
  {
    digitalWrite(mright, HIGH);
    delay(0.9);
    digitalWrite(mright, LOW);
    digitalWrite(mleft, HIGH);
    delay(2.1);
    digitalWrite(mleft, LOW);
    delay(sp2);
  }
  else if (maxind == 0 || maxind == 1) //if true, robot turns left
  { for (int r = 0; r < 1; r++)
    {
      digitalWrite(mright, HIGH);
      delay(0.9);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(1.5);
      digitalWrite(mleft, LOW);
      delay(sp2);
    }
  }
  else if (maxind == 4 || maxind == 5) //if true, robot turns right
  { for (int r = 0; r < 1; r++)
    {
```

```
      digitalWrite(mright, HIGH);
      delay(1.5);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(2.1);
      digitalWrite(mleft, LOW);
      delay(sp2);
    }
   }
   qtr.read(sensorValues);

   s = 0;
   for (int i = 0; i < 6; i++)
   {
    s += sensorValues[i];
   }
 }
 delay(1000);
 //push to align for right turn
 for (int r = 0; r < (push * 2); r++)
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 delay(1000);
 //turn right
 for (int r = 0; r < right; r++)
 {
  digitalWrite(mright, HIGH);
  delay(2.4);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
 delay(1000);
 //push to move beyond intersection
 for (int r = 0; r < push; r++)
 {
  digitalWrite(mright, HIGH);
  delay(0.6);
  digitalWrite(mright, LOW);
  digitalWrite(mleft, HIGH);
  delay(2.3);
  digitalWrite(mleft, LOW);
  delay(sp);
 }
}
if (inter == 6 && s > 3000)
{
```

```
inter++;

//read for object in front
digitalWrite(ult2, LOW);
delayMicroseconds(2);
digitalWrite(ult2, HIGH);
delayMicroseconds(10);
digitalWrite(ult2, LOW);
int duration1 = pulseIn(ule2, HIGH);
dc = (duration1 * .0343) / 2;
//if object detected in front
if (dc < 30)
{
 //glow led2
 digitalWrite(l2, HIGH);
 delay(2000);
 digitalWrite(l2, LOW);
 //move close to object
 delay(1000);
 digitalWrite(ult2, LOW);
 delayMicroseconds(2);
 digitalWrite(ult2, HIGH);
 delayMicroseconds(10);
 digitalWrite(ult2, LOW);
 int duration1 = pulseIn(ule2, HIGH);
 dc = (duration1 * .0343) / 2;

 while (dc < 30 && dc > 8)
 {
  qtr.read(sensorValues);

  int blah = 0;
  for (int i = 0; i < 6; i++)
  {

   if (sensorValues[i] > blah)
   {
    blah = sensorValues[i];

    maxind = i;
   }
  }

  if (maxind == 2 || maxind == 3) //if true, robot moves straight
  {
   digitalWrite(mright, HIGH);
   delay(0.6);
   digitalWrite(mright, LOW);
   digitalWrite(mleft, HIGH);
   delay(2.3);
   digitalWrite(mleft, LOW);
   delay(sp);
  }
  else if (maxind == 0 || maxind == 1) //if true, robot turns left
  {
   for (int r = 0; r < 1; r++)
```

```
          {
            digitalWrite(mright, HIGH);
            delay(0.6);
            digitalWrite(mright, LOW);
            digitalWrite(mleft, HIGH);
            delay(1.5);
            digitalWrite(mleft, LOW);
            delay(sp);
          }
        }
        else if (maxind == 4 || maxind == 5) //if true, robot turns right
        { for (int r = 0; r < 1; r++)
          {
            digitalWrite(mright, HIGH);
            delay(1.5);
            digitalWrite(mright, LOW);
            digitalWrite(mleft, HIGH);
            delay(2.3);
            digitalWrite(mleft, LOW);
            delay(sp);
          }
        }
        digitalWrite(ult2, LOW);
        delayMicroseconds(2);
        digitalWrite(ult2, HIGH);
        delayMicroseconds(10);
        digitalWrite(ult2, LOW);
        int duration1 = pulseIn(ule2, HIGH);
        dc = (duration1 * .0343) / 2;
      }
      //glow led3 after reaching object
      digitalWrite(l3, HIGH);
      delay(2000);
      digitalWrite(l3, LOW);
    }
    //take uturn
    for (int r = 0; r < uturn; r++)
    {
      digitalWrite(mright, HIGH);
      delay(0.6);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(0.6);
      digitalWrite(mleft, LOW);
      delay(sp);
    }
  }
}

else if (inter == 0 && s > 3000) //at the first intersection
{
  if (millis() > wait)
  {
    inter++;
    //glow led1
    digitalWrite(l1, HIGH);
```

```
      delay(2000);
      digitalWrite(l1, LOW);
    }
    for (int r = 0; r < push; r++)
    {
      digitalWrite(mright, HIGH);
      delay(0.6);
      digitalWrite(mright, LOW);
      digitalWrite(mleft, HIGH);
      delay(2.3);
      digitalWrite(mleft, LOW);
      delay(sp);
    }
  }
}
```