ROB 6103 : ADVANCED MECHATRONICS
Fall 2021

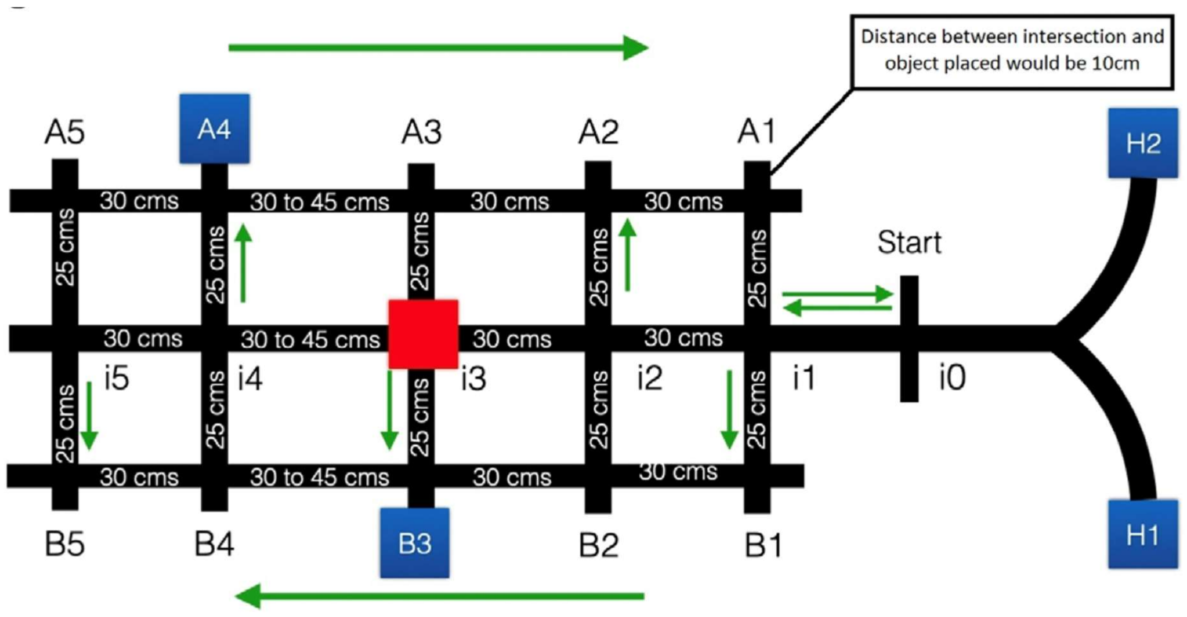
Project 2
Autonomous COVID Testing Kit Delivery Robot
Grid Style Lane Following Robot

DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING
NEW YORK UNIVERSITY

Student Name: Xinzhou Jiang (xj2106)
Penmetsa Tejaswi Raju (pr2258)
Rishi Eswar Varma Vegesna (rv2210)
Instructor: Prof. Vikram Kapila

Table Of Contents

OBJECTIVES



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. We were given the task of making a delivery bot for a grid style lane which is reminiscent of the streets of Manhattan. The figure above is the map of the objective that we must be complete.

Tasks:

1. We will start from either H1 or H2
2. Robot must follow the black line with which the track is made
3. The bot must reach the Start position I0 and should not stop at any movement looking for objects.
4. The robot should detect the intersections and provide an indication which we have done using a Blue LED.
5. The red box as shown in the figure is the obstacle our robot must avoid and take routes accordingly and following the traffic rules while doing the same.
6. The robot must reach the desired location detect the object (in our case used a Red LED) and not stop.
7. The robot must follow the traffic rules at all times while searching for the objects.

ELECTRONIC DESIGN

To meet the given objectives the bot was equipped with various electronic components. The list of electronic components used are as follows:

1. Parallax continuous rotation servo motors
2. QTR- 8RC Reflectance Sensor Array
3. HC-SR04 Ultrasonic Sensors
4. Propeller Activity Board WX
5. Power Supply Module
6. LED

1. Parallax Continuous Servo

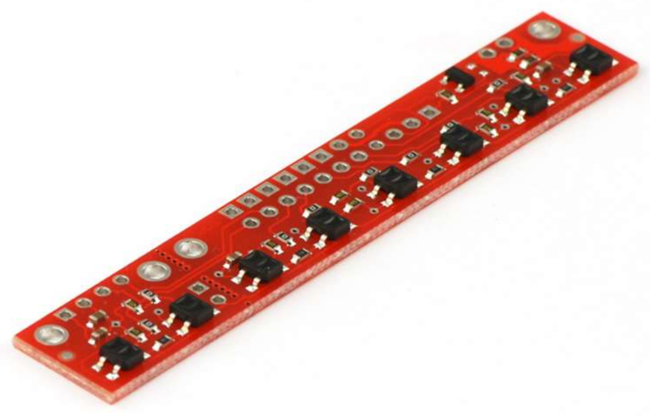
Continuous rotation servos are standard servos that have been modified to offer open-loop speed control instead of their usual closed-loop position control. The modification effectively turns them into motors with integrated motor drivers in a compact, inexpensive package. Just throw on a wheel and you have a drive system for your robot that can be controlled using an RC signal or a simple direct connection to a single microcontroller I/O line. The details of the Parallax Continuous Servo is:

a. 0 to 50 RPM with Pulse Width Modulation ramp up option
b. Weight of each servo is 42.5 grams
c. Power Requirements: 4 to 6 VDC and 15-200 mA according to the load the motors are on

2. QTR-8RC Reflectance Sensor Array

This sensor module has 8 IR LED/phototransistor pairs mounted on a 0.375" pitch, making it a great detector for a line-following robot. Pairs of LEDs are arranged in series to halve current consumption, and a MOSFET allows the LEDs to be turned off for additional sensing or power-savings options. Each sensor provides a separate digital I/O-measurable output**.** You can then read the reflectance by withdrawing that externally applied voltage on the OUT pin and timing how long it takes the output voltage to decay due to the integrated phototransistor. Shorter decay time is an indication of greater reflection. The specifications of this IR sensor array are:

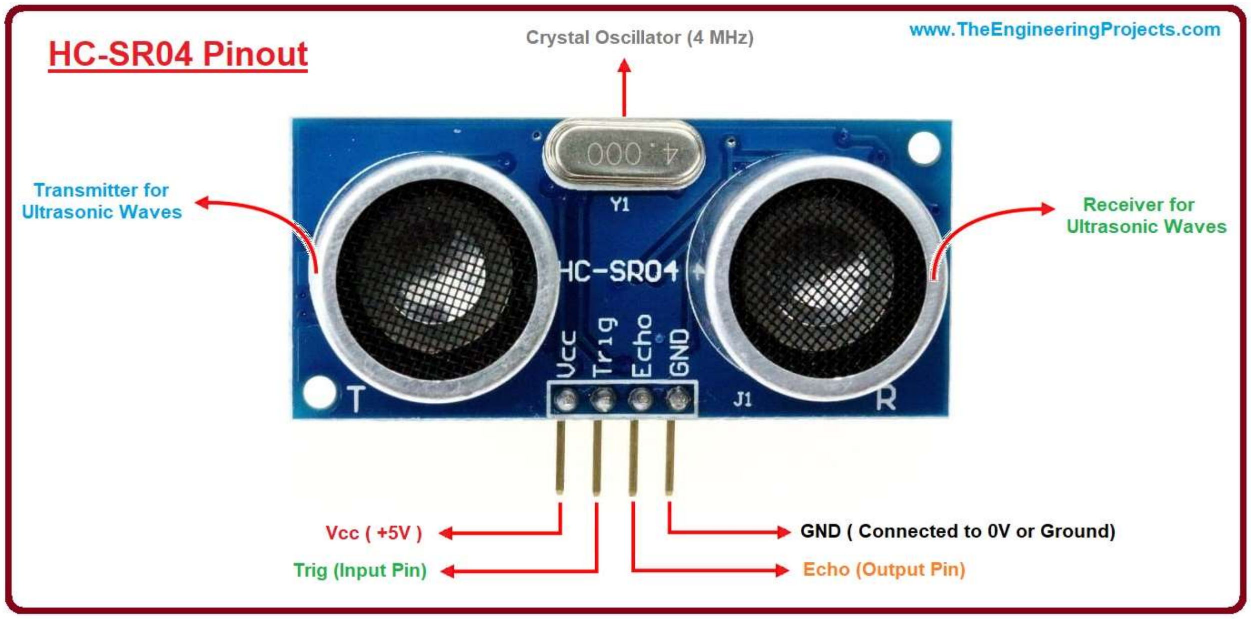
a. Operating Voltage: 3.3-5 V
b. Supply Voltage: 100 mA
c. Optimal Sensing Distance: 3mm



3. Ultrasonic Sensors

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity. For our project we are using the HC-SR04 ultrasonic sensor. It has two ultrasonic transducers. The one acts as a transmitter which converts electrical signal into 40 KHz ultrasonic sound pulses. The receiver listens for the transmitted pulses. If it receives them, it produces an output pulse whose width can be used to determine the distance the pulse travelled. The specifications of this Ultrasonic Sensors are:

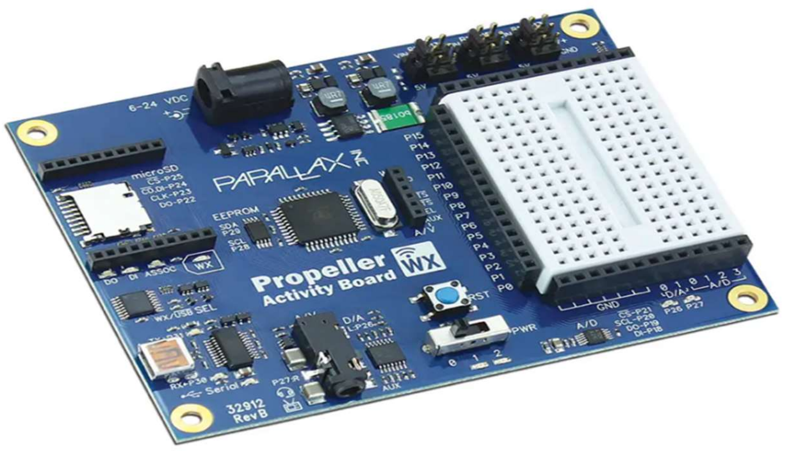
a. Operating Voltage: 5 VDC
b. Operating Current: 15 mA
c. Operating Frequency: 40 KHz
d. Max Range: 4 m
e. Minimum Range: 2 cm


HC-SR04 Pinout
Crystal Oscillator (4 MHz)
www.TheEngineeringProjects.com
Transmitter for Ultrasonic Waves
Receiver for Ultrasonic Waves
HC-SR04
Y1
Vcc
Trig
Echo
GND
T
R
J1
Vcc (+5V)
Trig (Input Pin)
GND (Connected to 0V or Ground)
Echo (Output Pin)

4. Parallax Activity Board WX

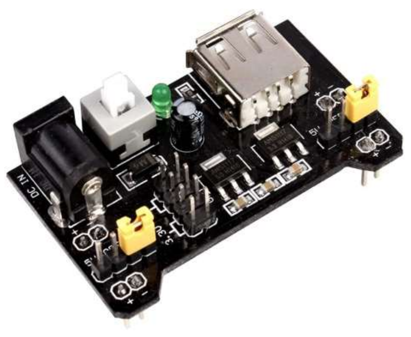
Parallax Activity Board is an activity board which features an 8 core Propeller microcontroller which has 32 Propeller I/O pins which is available for prototyping. It uses a Propeller P8X32A-Q44 microcontroller and has 8 cores. This activity board includes a RF module socket with Wi-Fi support, a microSD card holder which can be used to log sensor data, Mini TRRS audio/video jack. Each cog of the propeller board was called as cog and the board can achieve true multi-core functionality. Each Cog has a memory of 2KB and the board also consists of a central hub which provides 32KB ROM and 32KB RAM for every cog. Each Cog can access the central hub in a round-robin method. The following are the specifications of the board:

- Power Requirements: 6 to 15 VDC from an external power supply
- Storage: 64 KB I2C EEPROM
- Clock: 5 MHz crystal oscillator



5. Power Supply Module

A Power Supply Module is a Voltage regulator is a system which is designed to maintain a certain volage rating be it a Step Down Voltage or a Step Up. The voltage regulator we are using in our project is a 9V to 5V regulator that we are using to reduce the voltage supply from a 9 Volt battery to 5 V so as to supply the ultrasonic sensors with the required power that it needs. A voltage regulator is being used to power them as we found out that ultrasonic sensors were giving wrong readings when they were being powered simultaneously so we connected the servos and the ultrasonics differently just so that we are able to get consistent results.

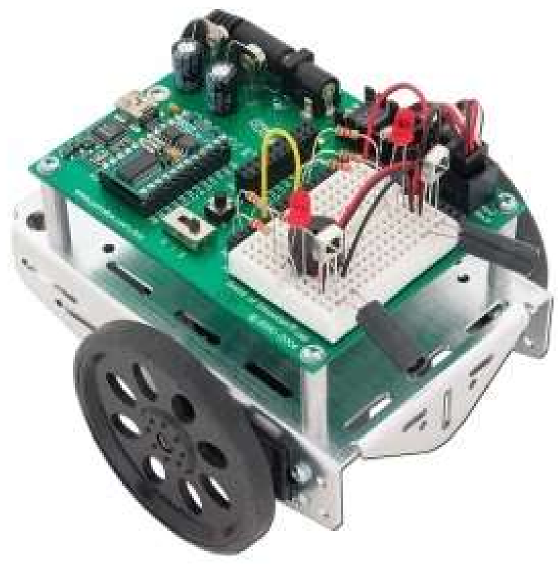


6. Light Emitting Diode

A LED or light emitting diode is a semiconductor light source that emits light when current flows through it. We are using an LED as an indicator when we detect an intersection and also when we detect an object.

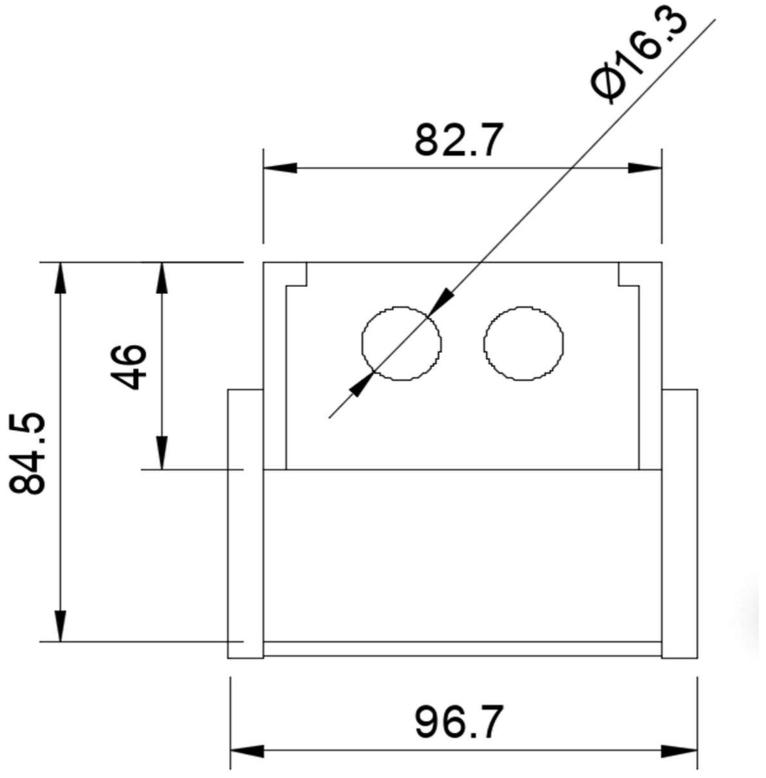
7. Boe-Bot Chassis

Boe-Bot Chassis is a aluminum made chassis that which is both durable and light weight. It also has numerous holes to fit in parts for both robot drive system and also the control systems for them. It also has slots to fit in the Continuous Servo motors which can be used for the motion of the robot.

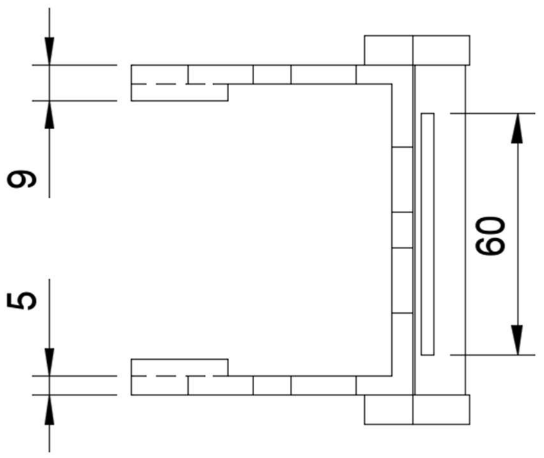


MECHANICAL DESIGN

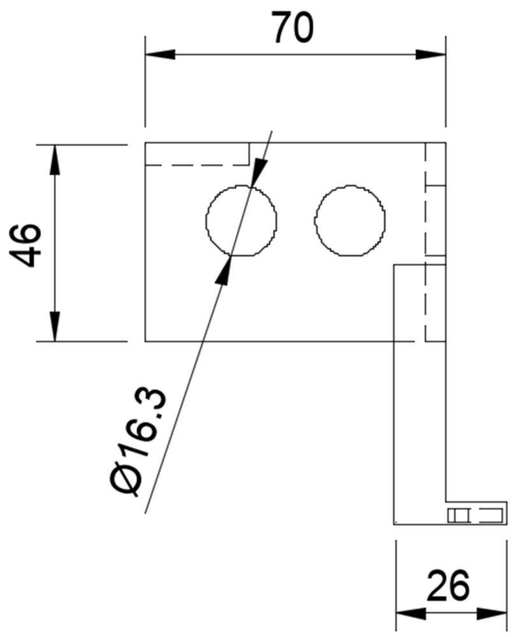
As the project uses 3 ultrasonic sensors to detect the presence and proximity of the object and the traffic congestion at the intersections a custom-built sensor mast was 3D printed from PLA to hold the ultrasonic sensors on the boe-bot chassis. The mast was printed in a way that it helps to holds the ultrasonic sensors at optimum height and angle to detect the 20 cm objects that are placed at the intersections. Apart from holding the ultrasonic sensors the mast was also made in a way that it could hold the QTR-8RC reflectance sensor. The CAD model design was designed in a way keeping in mind the limitations and specifications of the sensors. The reflectance array should by at a height of 3mm from ground for getting accurate values, so an extension was given in the design in such a way to hold the reflectance array in place. The CAD model diagrams of the Sensor mast are as follows:



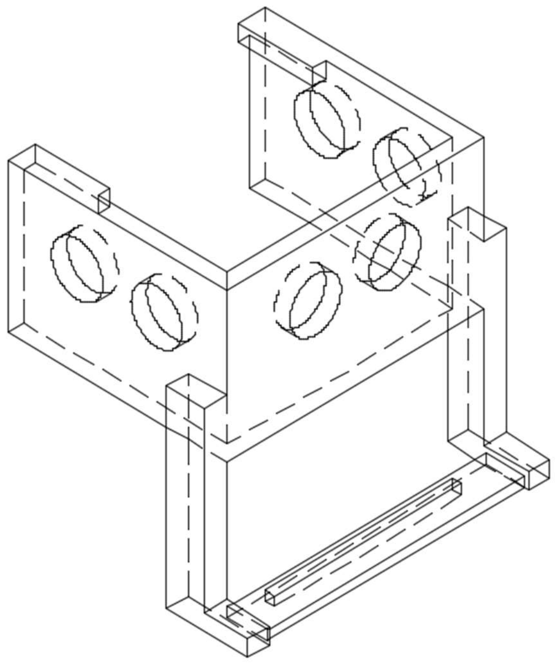
Front View



Top View



Side View



Isometric View

METHODOLOGY

The working of the bot consists of 2 major processes one was line following and other was object and traffic congestion detection. For the purposes of achieving this task We are using 4 cogs in total. Each cog was used for one of the following functions

- mainline following and route execution
- reading ultrasonic sensor data
- intersection check
- reading IR sensors

Intersection point check:

The intersection point check cog is continuously monitoring 8 IR sensors output. If all the sensors' values are greater than threshold 1000 and the counter value is greater than one, Blue LED will on, the joint counter will increase by one, and the joint flag will rise to high. The counter values are further monitored by the main cog 0 which determines the states.

IR sensor read:

IR sensor read cog is continuously reading the QTR IR sensor array and updating the global 8 sensors values for other cogs to monitor. According to the Parallax ultrasonic sensors examples, after setting the pins of the sensors high, the built-in function rc_time is used to read the sensors' values. In order to speed up the charging and discharging procedures which can increase the sensors' values updating speed, the emitter pin is connected to 3.3V

The code contains 2 major parts line follower and object detection. Each of these parts were discussed in detail below

Line Follower:

8 IR sensor array used, use middle three to detect whether the car is in the center, sensor 1, 2, 3 and sensor 6, 7, 8 is used to detect whether car is not in the middle. We used all of 8 sensors to determine the intersection point. We are using a COG just to follow line, this was done so as to achieve the objective that the bot should not stop at any moment. As there is no available library for QTR sensors on Propeller Board

we are using RC-Time function to get to know whether each of the above sensors are on the black line or not. By using this method, we found out that each sensor when facing the black surface has a sensor value above 400. We gave a thresh-hold of 400 as we found out from testing the array on the track. So as to detect the intersection when all eight sensors have values above 400 then and only then do, we say that an intersection is detected.

When the sensors 4 and 5 detect black surface and all other sensors detect white surface the bot will go forward. And when the sensors 6 sensor 7 and sensor 8 or any of them detect a black surface and sensors 1,2,3 detect white surface that means the bot have to move right so that it aligns with the line. So in this case the right servo turns a little faster while left servo moves at the same speed. This change in speeds of both servos help the bot to move left and align sensor 4 and 5 on the line making it go straight. Similarly when sensors 1,2, and or 3 detect a black line and sensors 6,7,8 detect a white line that means the bot is moving left and has to align so for this the left servo speed was increased a little compared to the right servo helping it to align with the black line. Each step of this servo motion was made small enough so it can replicate the process of PID control implementation in the line following and also to avoid the jerky motion of the robot. Speed and Timing of the bot was important to make the movement of the bot smooth. This is the gist of the method we are using to code the line follower part of this project.

Object Detection:

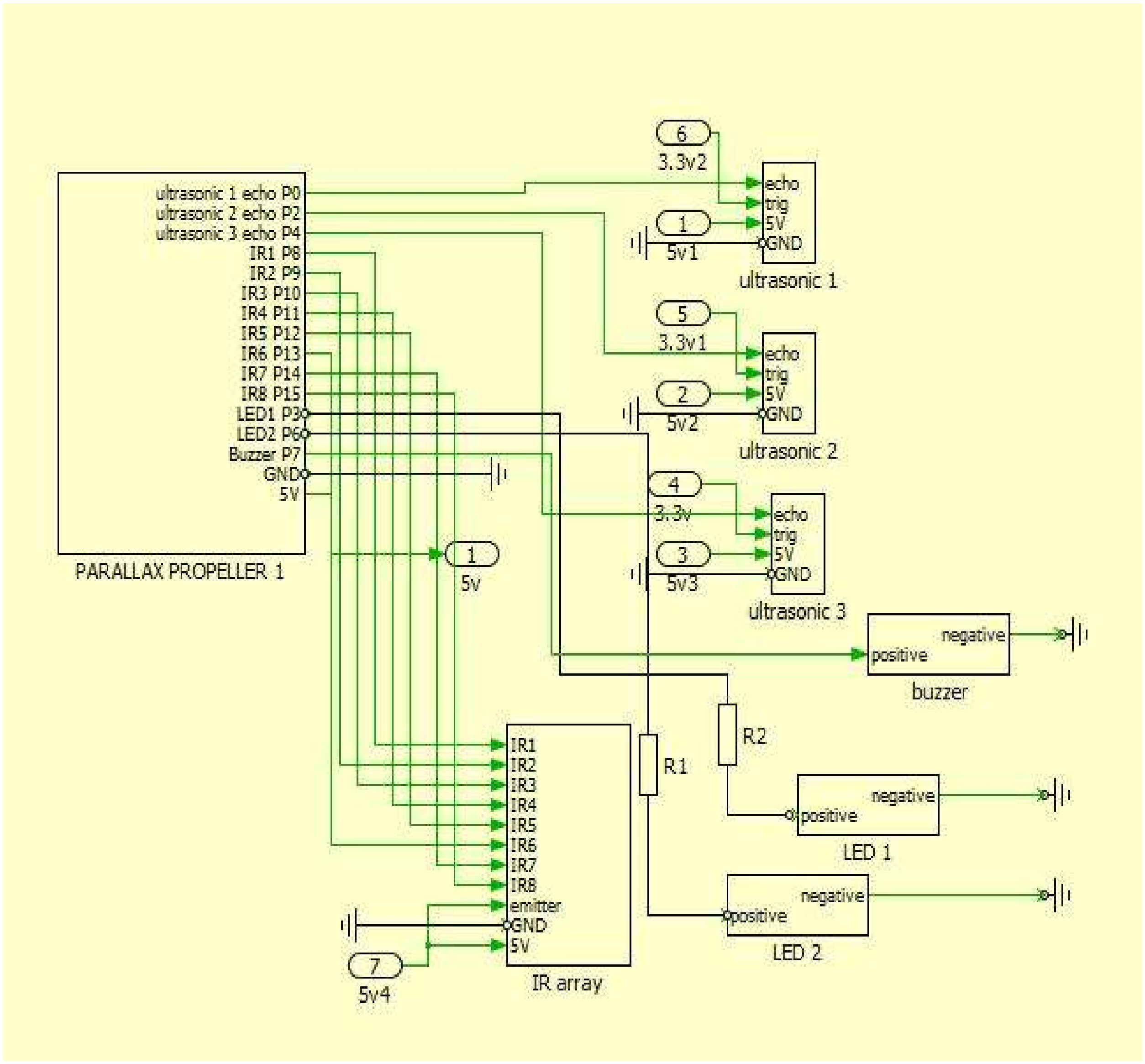
We have defined the right-side ultrasonic sensor to be Ultrasonic 1 and the forward ultrasonic to be Ultrasonic 3. We are using only two ultrasonic sensors as for the given layout we found out that the objects that needs to be detected is always on the left side.

We are powering the ultrasonics at all times just to detect the obstruction and the objects; all these tasks are assigned to one COG. A function in the code is used to detect the objects at distance of 30cm. 30 cm is the distance where the object will be placed according to the objective given to us.

We have programmed the bot to first go down the main road and detect the road obstruction first. When doing the same a COG keeps calculating the intersections that it is passing through so that our bot does not disobey the traffic rules.

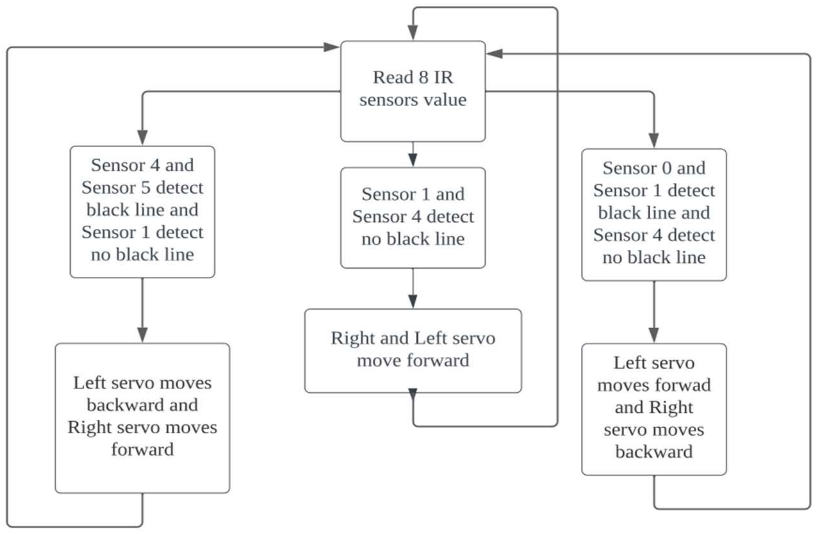
When the road obstruction is detected, our bot indicates it with a Red LED glow and takes a right or left according to the intersection it just detected while it detected the obstruction. Then it follows a particular path that we have written a code on. When it detects the objects on either lane on the track it again glows the Red LED just to indicate that it has sensed an object. We have given it a path to follow according to where the obstruction is detected on the track. By doing so we follow the traffic rules that are specified to us in the objective.

CIRCUIT DIAGRAM

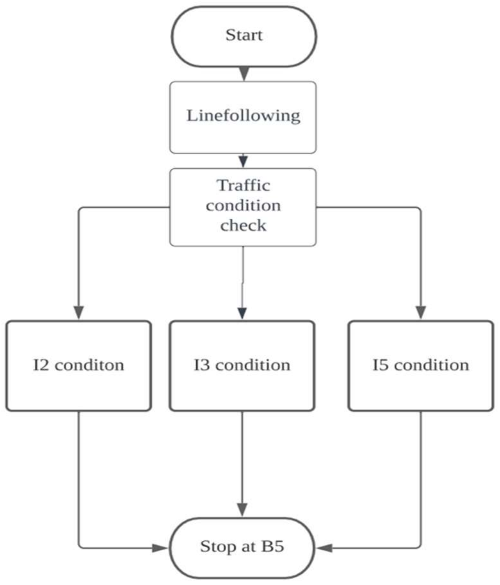


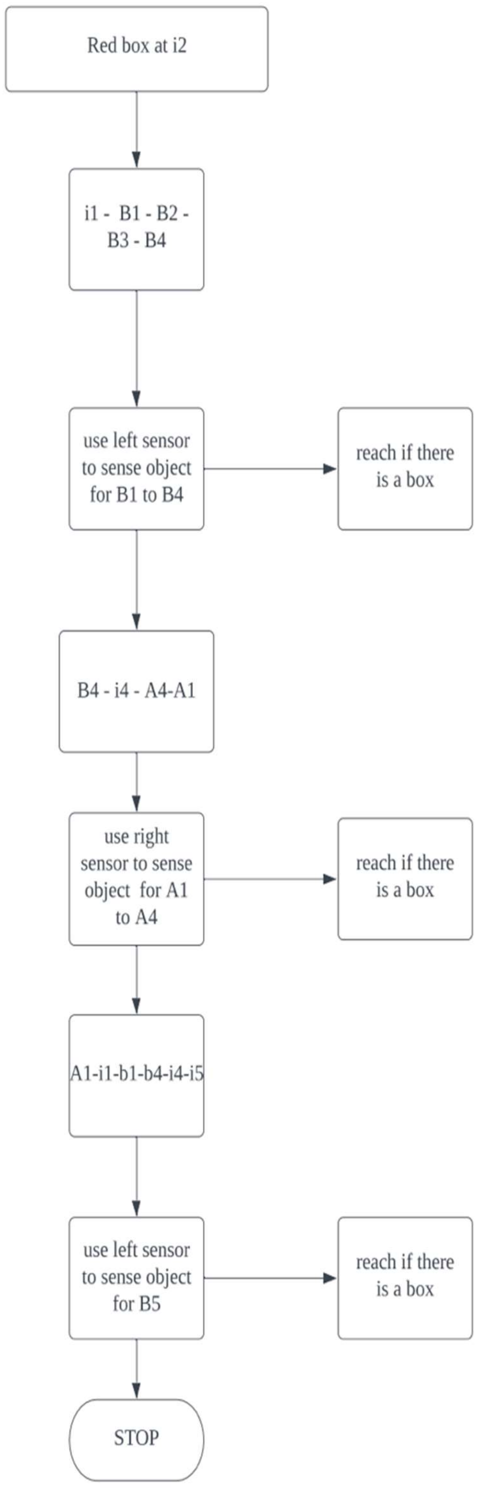
The electronic components are connected to one another in the most optimum way possible. The 8 pins of the 8RC Reflectance sensor are directly interfaced to the Propeller Activity WX board. The Continuous Servo Motors pins were directly connected to the servo pins on the Propeller Activity WX board which can be powered by using mode 2 on the board. The left servo signal pin was connected to pin 17 and right servo signal pin was connected to pin 17. Pins 2 and 4 were used to connect the ultrasonic sensor echo pins and trigger pins of the ultrasonic sensors were directly connected to 3.3V on the board. The LEDs were connected to pin 3 and 5 these LED were used to signal when objects and intersections were detected.

BLOCK DIAGRAM

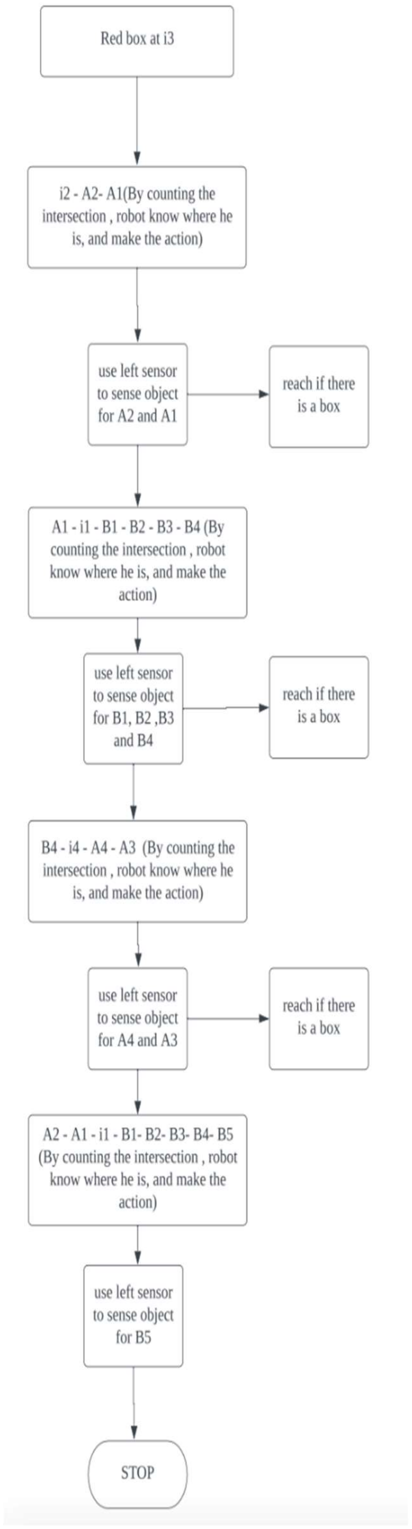

Read 8 IR sensors value
Sensor 4 and Sensor 5 detect black line and Sensor 1 detect no black line
Sensor 1 and Sensor 4 detect no black line
Sensor 0 and Sensor 1 detect black line and Sensor 4 detect no black line
Left servo moves backward and Right servo moves forward
Right and Left servo move forward
Left servo moves forwad and Right servo moves backward

Line Following Block Diagram

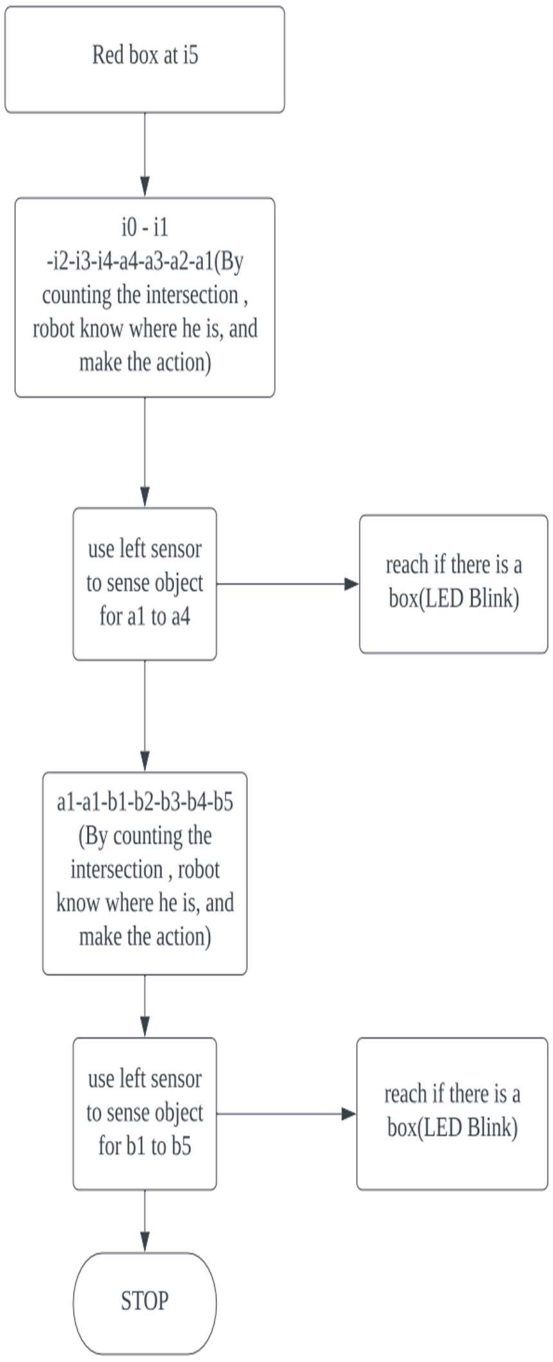



Red box at i2
i1 - B1 - B2 - B3 - B4
use left sensor to sense object for B1 to B4
reach if there is a box
B4 - i4 - A4-A1
use right sensor to sense object for A1 to A4
reach if there is a box
A1-i1-b1-b4-i4-i5
use left sensor to sense object for B5
reach if there is a box
STOP

CASE-1


Red box at i3
i2 - A2- A1(By counting the intersection , robot know where he is, and make the action)
use left sensor to sense object for A2 and A1
reach if there is a box
A1 - i1 - B1 - B2 - B3 - B4 (By counting the intersection , robot know where he is, and make the action)
use left sensor to sense object for B1, B2 ,B3 and B4
reach if there is a box
B4 - i4 - A4 - A3 (By counting the intersection , robot know where he is, and make the action)
use left sensor to sense object for A4 and A3
reach if there is a box
A2 - A1 - i1 - B1- B2- B3- B4- B5 (By counting the intersection , robot know where he is, and make the action)
use left sensor to sense object for B5
STOP

CASE-2


Red box at i5
i0 - i1 -i2-i3-i4-a4-a3-a2-a1(By counting the intersection , robot know where he is, and make the action)
use left sensor to sense object for a1 to a4
reach if there is a box(LED Blink)
a1-a1-b1-b2-b3-b4-b5 (By counting the intersection , robot know where he is, and make the action)
use left sensor to sense object for b1 to b5
reach if there is a box(LED Blink)
STOP

CASE-3

DISCUSSION AND CONCLUSION

The Robot achieves the given objective i.e., starting from home position and reaching the start position. After the start position the robot should follow the path and signal whenever it detects an intersection. It should continue its path in the central lane and should indicate when it detects the traffic congestion at i2, i3, or i5 once it detects the traffic congestion and indicated the congestion it should maneuver around the congestion by adhering to the traffic rules. It should also look for objects in A1 to A4 and B1 to B5 and should signal when an object was seen. And one of the given objective was that the robot should never stop at any intersection. The Robot achieves all the given tasks and objectives. One of the prominent problems of the robot was it should be supplied with enough power source to run the high peripherals like servos. Powering the servo motors using Propeller activity WX board may affect the working of the Robot as motors use 6V DC at max speed no load conditions but propeller board can only supply a 5V output as it is connected to 6V portable battery pack, so the working of servos may be limited and if the peripherals are not properly connected without enough power, they may end up causing propeller board to go into brown- out mode which may affect the functionality of the Robot. The Robot can be seen as a scaled down prototype of contactless delivery mobile Robot which has potential applications in the pandemic situation. Furthermore, it can also be used as a In-house delivery bot in large office spaces to replace the man force deployed to transport various documents in the workspace. Finally, by using a proper power source and calibrating the sensors used in the Robot properly, the Robot can efficiently and perform the given tasks in the known environments.

APPENDIX

Code:

The programming of the Propeller Activity WX board was done using simpleIDE. Libraries like Servo, ping was used to interface the continuous servo motors and ultrasonic sensors

```
/*
Blank Simple Project.c
http://learn.parallax.com/propeller-c-tutorials
*/
#include "simpletools.h" // Include simple tools
#include "servo.h"
#include "ping.h" // Include ping tools


#define true 1
#define false 0
#define I2_STATE 2
#define I3_STATE 3
#define I5_STATE 4
#define FORWARD 6
#define LEFT 7
#define RIGHT 8





// Forward declaration
void Stop();
void LEDblink();
void LEDON();
void joint_check();
void redbox_check();
void turnright();
void turnleft();
void i5_state();
void i2_state();
void i3_state();

void forward();
void followline();
void readdata();
```

```
int location_check();
void IRsense();
void BohemianRhapsody();


// predesign path array for each condition

// statks for different cogs

// For storing process ID
int *cog1;
int *cog2;
int *cog3;
int *cog4;
int *cog5;
int *cog6;
int *cog7;

//static volatile int count ; // Global vars for cogs to share

int redbox_flag2 = 1;
int count3;
int count2;

//int intcheckflag = 0;
int joint_flag;
int count_flag;

//count1 is intersection count
//count4 is object count
static volatile int time1, time2, time3, time4, time5, time6, time7, time8, count1=0, count4;

//ultrasonic sensor values
int cmDist1;
int cmDist2;
int cmDist3;


void IRsense(){
while(1)
{
//read values from all the 8 sensors
high(8);
high(9);
high(10);
high(11);
high(12);
high(13);
high(14);
high(15);
pause(100);
time1=rc_time(8,1);
time2=rc_time(9,1);
time3=rc_time(10,1);
time4=rc_time(11,1);
time5=rc_time(12,1);
time6=rc_time(13,1);
time7=rc_time(14,1);
time8=rc_time(15,1);

}
}



void turnright()
{
//tunrright90 deg
servo_speed(16,0);
servo_speed(17,100);
pause(1200);
}
void turnleft()
{
//tunrleft90 deg
servo_speed(16,-100);
servo_speed(17,-10);
pause(1250);
}

void joint_check(){//check if the interseciton point found
while(1)
{
//print("%csensor2 = %d, sensor3 = %d%, sensor6 = %d%, sensor7 = %d%c", HOME, sensor2,
sensor3,sensor6,sensor7, CLREOL);
pause(100);
//intersection point found condition
if ( time2 > 900 && time3 > 900 && time4 > 900&& time5 > 900&& time6 > 900 && time7 >
900 && time8>900 )
{
//count1 ++;
//count3 = count1 - 3;
//servo_speed(16, -40);
//servo_speed(17, 20);
if (count1>=0)
{
count1 ++;
joint_flag = 1;
pause(500);
high(5);
pause(500);
low(5);
}
//avoid first interscetion which is the curve one
else if (count1 == 0 )
{
pause(100);
count1 ++;
joint_flag = 1;
}
if(count1 == 3 && cmDist3 < 32 ) //i2 condition
{

redbox_flag2 = I2_STATE;
}
else if(count1 == 4 && cmDist3 < 32) //i3 condition
{
redbox_flag2 = I3_STATE;
}
else if(count1 == 6 && cmDist3 < 32) //i5 condition
{
redbox_flag2 = I5_STATE;
}
else{
redbox_flag2 = 1; //meets no conditions
}
}
else
{ joint_flag = 0;
low(6);
}
}
}
void followline(){


if ((time4 > 400 && time5 > 400) && (time3 < 400 || time6 < 400) )
{ // sees no line on either side, goes straight until a line is seen
```

```
servo_speed(16, -40);
servo_speed(17, 20);
}
else if ((time3 > 400 && time4 > 400) && (time2 < 400 || time7 < 400) )
{ // sees line on right side, is making a right turn
servo_speed(16, -20);
servo_speed(17, 20);
}
else if (((time6 > 400 && time7 > 400)||(time7>400 && time8> 400)) && ( time2 < 400||
time1<400) )
{ // sees line on left side, is making a left turn
servo_speed(16, -40);
servo_speed(17, 0);
}

else
{ // both are seeing line, goes straight until light is seen
servo_speed(16, -40);
servo_speed(17, 20);
}
}

void readdata()//
{
while(1)
// Repeat indefinitely
{
//rightsensor =cmDist1, leftsensor= cmDist2, middlesensor = cmDist3

pause(100);
cmDist1 = ping_cm(0);
pause(200);
cmDist2 = ping_cm(2);
pause(200);
cmDist3 = ping_cm(4); // Get cm distance from Ping)))
pause(200);
while(cmDist3 < 30 )
{
//object at the front detected
count4++;
high(3);
pause(500);
low(3);
pause(500);
break;
}
```

```
while(cmDist2 < 20 )
{
//location object detected
high(3);
pause(500);
low(3);
pause(500);
count4++;
break;
}
// Wait 1/5 second
}

}




void i2_state(){ //
//int count4;
while(1){
//count4 = count1 - 3;

if ( count1 == 3 && joint_flag == 1)
{
turnleft();
followline();
continue;

}

else if (count1 == 4 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 7 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 9 && joint_flag == 1)
```

```
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 12 && joint_flag == 1)
{// turn left
turnright();
followline();
continue;
}

else if (count1 == 14 && joint_flag == 1 )
{//turn right
turnright();
followline();
continue;

}

else if (count1 == 18 )
{
servo_disable(16);
servo_disable(17);
pause(10000);

}
else
{followline();
}
}
}

void i3_state(){ //
//int count4;
while(1){
//count4 = count1 - 3;

if ( count1 == 4 && joint_flag == 1)
{
turnright();
followline();
continue;

}
```

```
else if (count1 == 5 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 6 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 8 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 11 && joint_flag == 1)
{// turn left
turnright();
followline();
continue;
}

else if (count1 == 13 && joint_flag == 1 )
{//turn right
turnright();
followline();
continue;

}

else if (count1 == 16 && joint_flag == 1 )
{//turn right
turnright();
followline();
continue;

}

else if (count1 == 18 && joint_flag == 1 )
```

```
{//turn right
turnright();
followline();
continue;

}

else if (count1 == 22 )
{
servo_disable(16);
servo_disable(17);
pause(10000);

}
else
{followline();
}
}
}

void i5_state() { //
//int count4;
while(1){
//count4 = count1 - 3;

if ( count1 == 6 && joint_flag == 1)
{
turnright();
followline();
continue;

}

else if (count1 == 4 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 7 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}
```

```
else if (count1 == 10 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 12 && joint_flag == 1)
{// turn left
turnright();
followline();
continue;
}


else if (count1 == 16 )
{
servo_disable(16);
servo_disable(17);
pause(10000);

}
else
{followline();
}
}
}




int main() // Main function
{
//int cog1 = cogstart(&IRsense, NULL, stack1, sizeof(stack1));
//int cog2 = cogstart(&followline, NULL, stack2, sizeof(stack2));
cog_run(IRsense, 128);
//cog1 = cogstart((void*)IRsense, NULL, stack1, sizeof(stack1));
cog_run(joint_check, 128);
//cog3 = cogstart((void*)joint_check, NULL, stack2, sizeof(stack2));
cog_run(readdata, 128);
//cog4 = cogstart((void*)readdata, NULL, stack4, sizeof(stack4));
//cog_run(BohemianRhapsody, 128);
//cog6 = cogstart((void*)redbox_check, NULL, stack6, sizeof(stack6));
pause(50);
```

```
while(1){
if ((time4 > 400 && time5 > 400) && (time3 < 400 || time6 < 400) )
{ // sees no line on either side, goes straight until a line is seen
servo_speed(16, -40);
servo_speed(17, 20);
}
else if ((time3 > 400 && time4 > 400) && (time2 < 400 || time7 < 400) )
{ // sees line on right side, is making a right turn
servo_speed(16, -20);
servo_speed(17, 20);
}
else if (((time6 > 400 && time7 > 400)||(time7>400 && time8> 400)) && ( time2 < 400||
time1<400) )
{ // sees line on left side, is making a left turn
servo_speed(16, -40);
servo_speed(17, 0);
}

else
{ // both are seeing line, goes straight until light is seen
servo_speed(16, -40);
servo_speed(17, 20);
}
if(redbox_flag2 != 1)
{//redbox detected at certain intersection , robot go to predesign path
switch(redbox_flag2){
//(one core)do design path for i + (one core)joint_check()+(one core)location_check()+(one
core)play cool music
case I2_STATE: i2_state(); break;
//cogstart(&joint_check, NULL, stack, sizeof(stack)); cogstart(&location_check, NULL, stack,
sizeof(stack));break;
case I3_STATE: i3_state(); break;
//cogstart(&joint_check, NULL, stack, sizeof(stack)); cogstart(&location_check, NULL, stack,
sizeof(stack));break;
case I5_STATE: i5_state(); break;
//cogstart(&joint_check, NULL, stack, sizeof(stack)); cogstart(&location_check, NULL, stack,
sizeof(stack));break;
}
//printf("rc_time = %d ,%d ,%d ,%d ,%d ,%d ,%d , %d \n",
time1,time2,time3,time4,time5,time6,time7,time8);
//printf("count1 = %d \n",count1);
}
//printf("count1 = %d \n",count1);
print("right =%d, left= %d, middle = %d, redflag2 =%d, count = %d \n", cmDist1,
cmDist2,cmDist3,redbox_flag2,count1 ); // Display result
pause(100); // Wait 1/10 of a second
}
}
```