ROB 6103 : ADVANCED MECHATRONICS
Fall 2021

Term Project
Autonomous Grid Style Lane Following Robot with Enemy/Friendly Detection and Knocking Mechanism

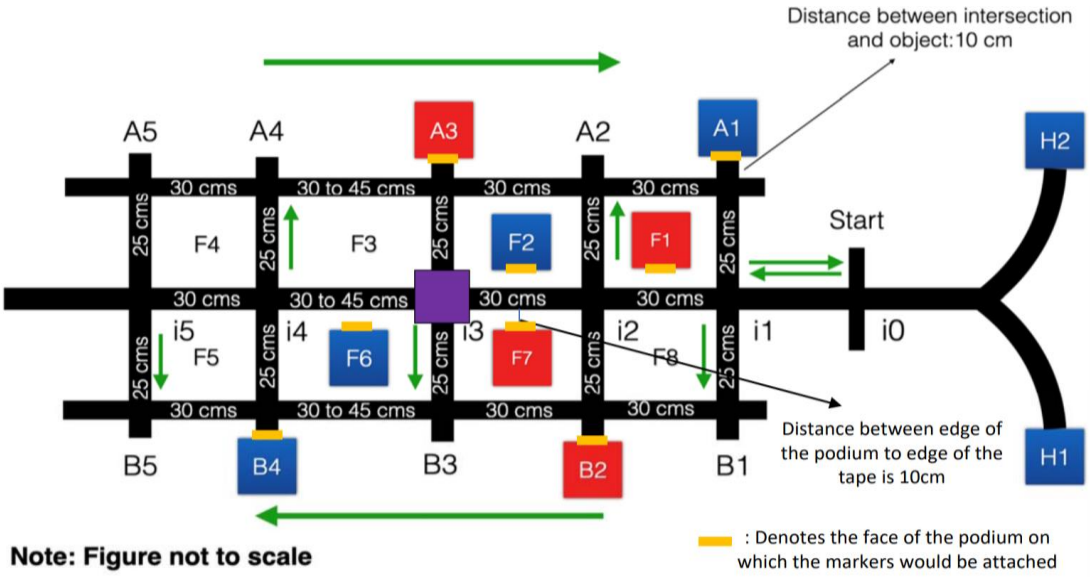
DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING
NEW YORK UNIVERSITY

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

Table of Contents

OBJECTIVES

We were tasked with the responsibility of designing an autonomous robot that will move around the streets of Manhattan Grid like path consisting of intersection A1 to A5, B1 to B5, and I0 to I5, to find enemy intruders (the enemies) in the path who are troubling the civilians (the friendlies) in the city. The robot must lookout for the enemies and eliminate them by scouring through the streets while following all the traffic rules and should avoid obstacles.



Task descriptions

1. The robot must use 2 microcontrollers: 1st- Raspberry Pi and 2nd- Propeller Activity Board.
2. The robot will start at either one of the safe houses H1 or H2.
3. The robot must always follow the path (black tape) in a smooth but fast enough speed.
4. The robot must reach i0 (start position) and should NOT stop at any time during its full run unless it detects an enemy/friendly.
5. The robot must move towards the friendlies/enemy's locations to reach them while avoiding obstacles on its path. It should provide an indication of reaching each location by some means like LED
6. The robot should knock off the enemies from their car (or podium) by a mechanism attached on their robot. It should NOT hurt the friendlies. It can detect whether an object was a friendly or an enemy by using the aruco tag present on the front face of the podium

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. Parallax Propeller Activity Board WX
5. Raspberry pi 3B+
6. Micro-servo SG90
7. Power Supply Module
8. LED

1.Parallax Continuous Servo

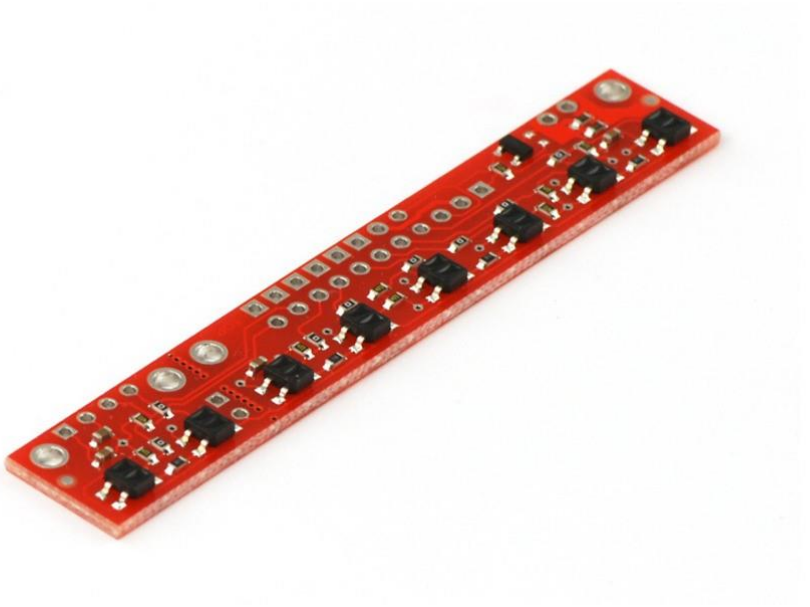
Continuous rotation servos are standard servos that have been modified to offer open-loop speed control instead of their usual closed-loop position control. The modification effectively turns them into motors with integrated motor drivers in a compact, inexpensive package. Just throw on a wheel and you 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 are:

- 0 to 50 RPM with Pulse Width Modulation ramp up option
- Weight of each servo is 42.5 grams
- 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:

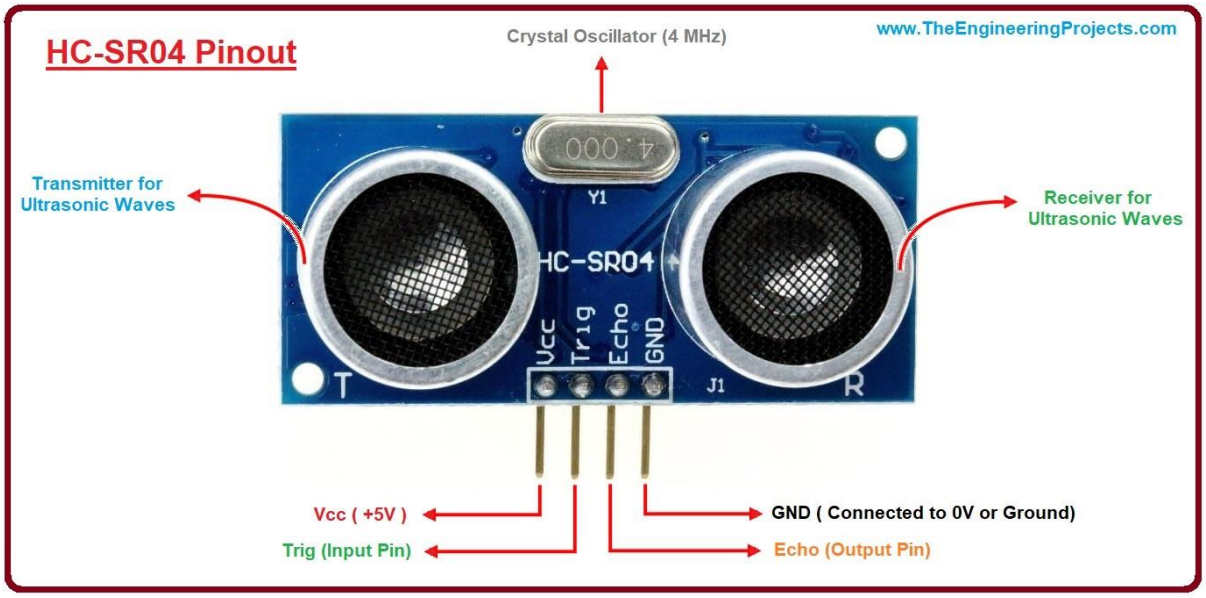
- Operating Voltage: 3.3-5 V
- Supply Voltage: 100 mA
- Optimal Sensing Distance: 3mm



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:

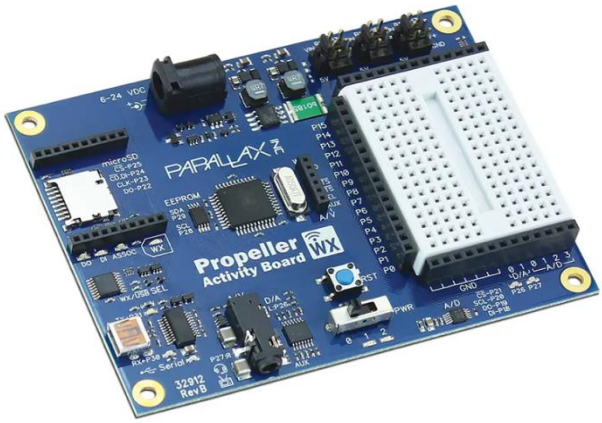
- Operating Voltage: 5 VDC
- Operating Current: 15 mA
- Operating Frequency: 40 KHz
- Max Range: 4 m
- Minimum Range: 2 cm



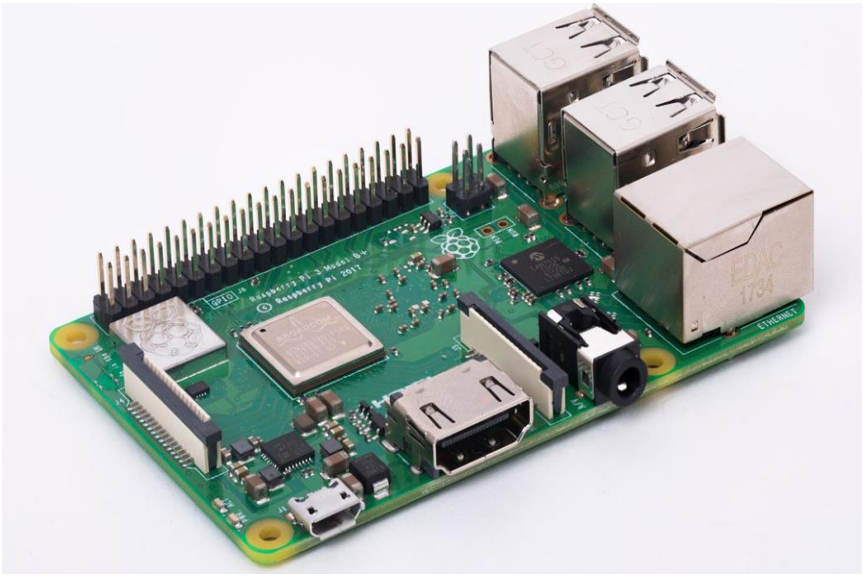
4.Parallax Activity Board WX

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.Raspberry Pi Model 3B+

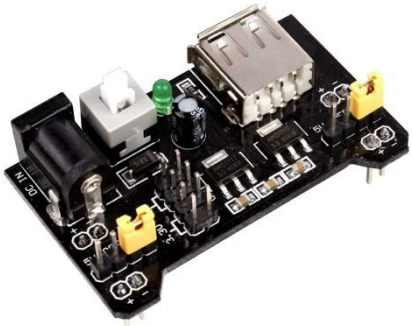


The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processer, 10x faster than the first-generation Raspberry Pi. Additionally, it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs. The technical specifications of the Raspberry Pi 3 Model B are:

- Power: Micro USB socket 5V1, 2.5A
- Storage: 1GB LPDDR2
- Processor: Broadcom BCM2387 chipset, 1.2GHz Quad-core ARM Cortex-A53

6.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.



7.Micro Continuous Servo (SG-90)

A micro-servo SG90 from Arduino kit was used in the knocking mechanism. It was a continuous servo motor but works as a standard servo base on gear feedback The specifications of the servo motor were as follows:

- Torque- 2.5 kg-cm
- Voltage- 4.8-6V

8.Light Emitting Diode

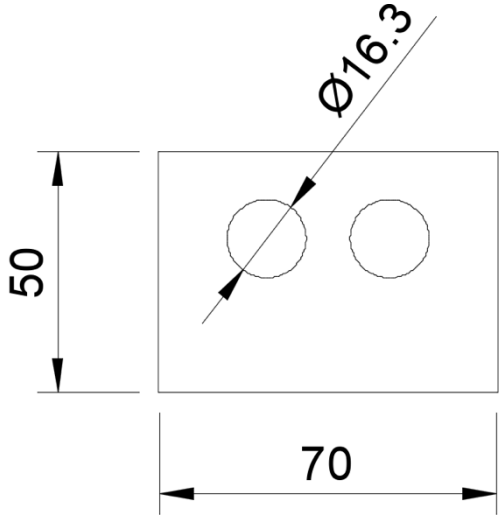
A LED or light emitting diode is a semiconductor light source that emits light when current flows through it. We are using an LED as an indicator when we detect an intersection and also when we detect an object.

9.Boe-Bot Chassis

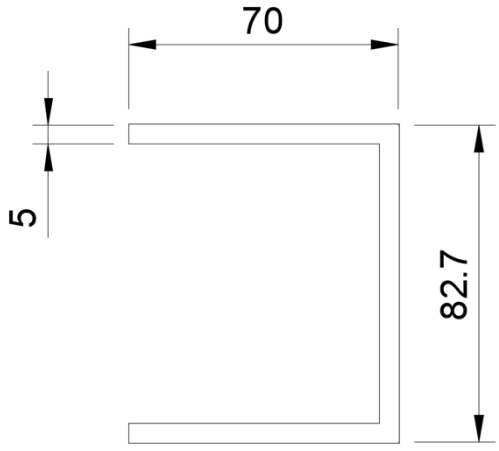
Boe-Bot Chassis is a aluminum made chassis that which is both durable and light weight. It also has numerous holes to fit in parts for both robot drive system and also the control systems for them. It also has slots to fit in the Continuous Servo motors which can be used for the motion of the robot.

MECHANICAL DESIGN

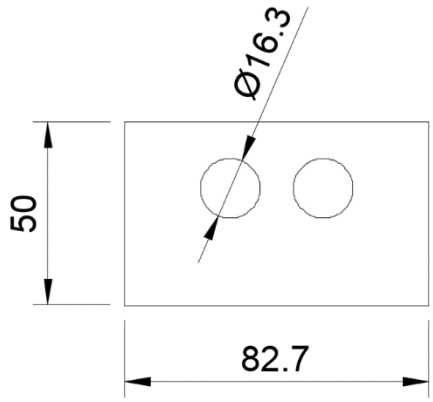
As the project uses 2 ultrasonic sensors to detect the presence and proximity of the object and the traffic congestion at the intersections a custom-built sensor mast was 3D printed from PLA to hold the ultrasonic sensors on the boe-bot chassis. The mast was printed in a way that it helps to holds the 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:



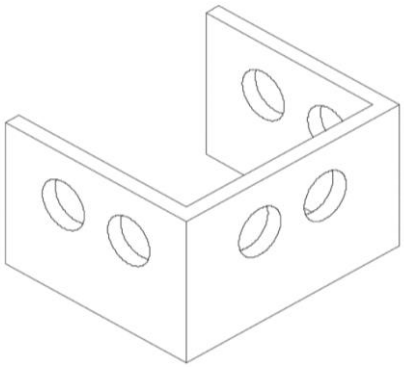
Front View



Top View



Side View



Isometric View

METHODOLOGY

The working of the bot consists of 4 major processes one was line following and other was object and traffic congestion detection. For the purposes of achieving this task We are using 5 cogs in total in propeller, and we are using the raspberry pi and the raspberry pi camera to detect whether the object detected was enemy or friend. Each of the 5 cogs were used for one of the following functions

- mainline following and route execution
- reading ultrasonic sensor data
- intersection check
- reading IR sensors
- to monitor and communicate with raspberry pi

And the main function of the raspberry pi is to detect whether the block kept on the left side of the bot is enemy or friend by checking the Aruco tag provided on it. After detecting an enemy the servo sends a high to the servo to move it to 180 degree and then back to 90 degree. After detecting an enemy, a blue LED lights and when a friend is detected a yellow LED light.

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 was used, and we use middle three to detect whether the car was on the center or not. And we were using sensors 3,4,5,6,7,8 to check whether the robot was at intersection or not. And sensors 3,4 were used to detect if the bot was moving towards right and sensors 6, 7, 8 were used to detect if the car was going to the left. 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.

Obstruction Detection:

Our bot has an ultrasonic sensor facing the forward direction, so when a traffic obstruction is detected, a desired path is followed so that the bot covers all the required area where the enemies and friends are placed. Enroute to the traffic obstruction the bot detects friendlies or enemies on the left side, stops, moves the servo to knock down if an enemy is detected, then takes a U-Turn after detecting the traffic obstruction and starts detecting the enemies and friendlies on the right side of the track and goes till position i1 and after reaching this intersection it takes a right onto the B- side of the track and goes from B1 to B4 all the time looking for both friendlies and enemies while following this path. After reaching B4 the bot takes a right turn again and goes to the A- side. Starting from A4 the bot goes all the way to A1 and then takes a right turn again towards point B1 where it takes a right again goes all the way to B5 where it stops. At normal positions like A2, A3, B3, B3, and F1 to F8 the bot detects the objects perfectly but at the fours corners A1, B1, A4, B4 the bot changes direction from centre to A lane and B to Centre Lane and From A to Centre Lane at this points since the IR sensor was at front and detects the intersection first it executes turn command first instead of object detection due to this the left Ultrasonic sensor was added to aid the Bot in this process. If the IR detects an object in these four corners it executes servo stop for a small duration during which the camera check whether it was an enemy/ friendly and hits if it was an enemy the turn command was executed. This ultrasonic aid was added because of the Low FPS of Raspberry Pi camera

Enemy and Friendly Block Detection:

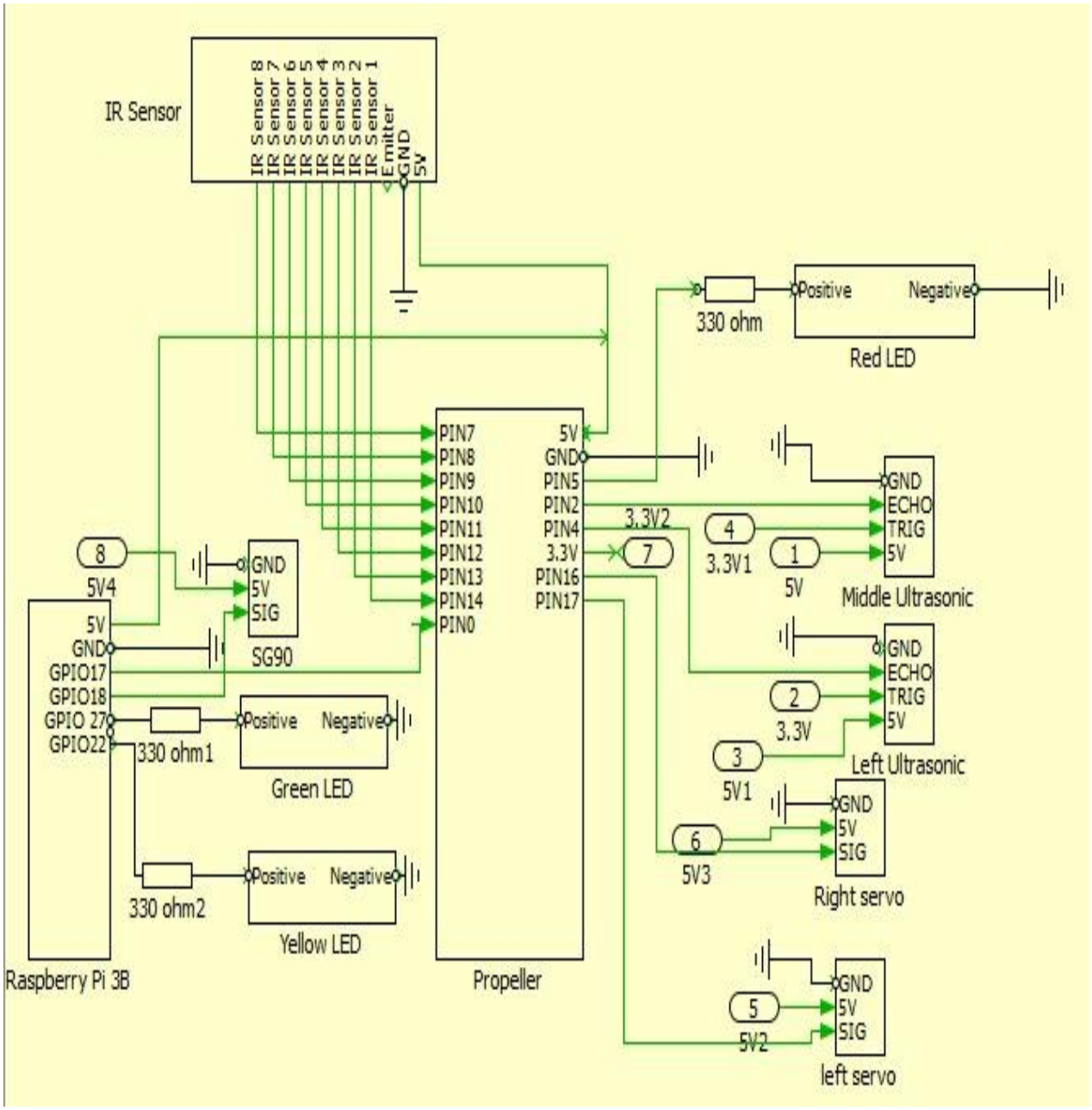
To detect an enemy/ Friendly we use a Raspberry pi Camera. Usually the pi camera was connected to the slot available the pi board but since our camera port was not working properly we used an USB camera in its place. The USB camera was connected to one of the USB port on the board. We have 2 two USB cameras in which we were using a Fish Eye Camera (Wide Vision Camera). But by using this camera we faced a small issue since the camera was a wide angled lens when was travelling in centre lane also it was able to detect tags in A lane and B lane which caused it to miss some objects while travelling in that lane. To overcome this issue the raspberry pi code was written in a way that it sees the Aruco tags available in only some part of the camera vision. This was done by using img.shape, np.array, and cv2.fillPoly functions in python cv2 library.

Obstruction Detection:

As stated in the objective that friendly block will be denoted by Aruco Tag numbered from 0 to 9 and the enemy block will be denoted by Aruco Tag numbered from 10 to 19. A python code was stored on the Raspberry Pi board and the Raspberry Pi camera was also connected to the Raspberry Pi via a USB. When a friendly block is detected the raspberry sends a high signal to the propeller board which makes the bot stop after which a Yellow LED on the raspberry pi board was used to indicate that a friendly is detected. If the bot detects an enemy on the way, the raspberry pi again sends a high signal to the propeller to make the bot stop after which then a blue LED glows

and then we move the servo to a 90-degree angle and then back to its home position back to 180 degrees which helps in knocking off the enemy. And also once an Aruco tag was detected it wouldn't get scanned again to avoid multiple knockings of the same enemy when the bot was travelling in the same path.

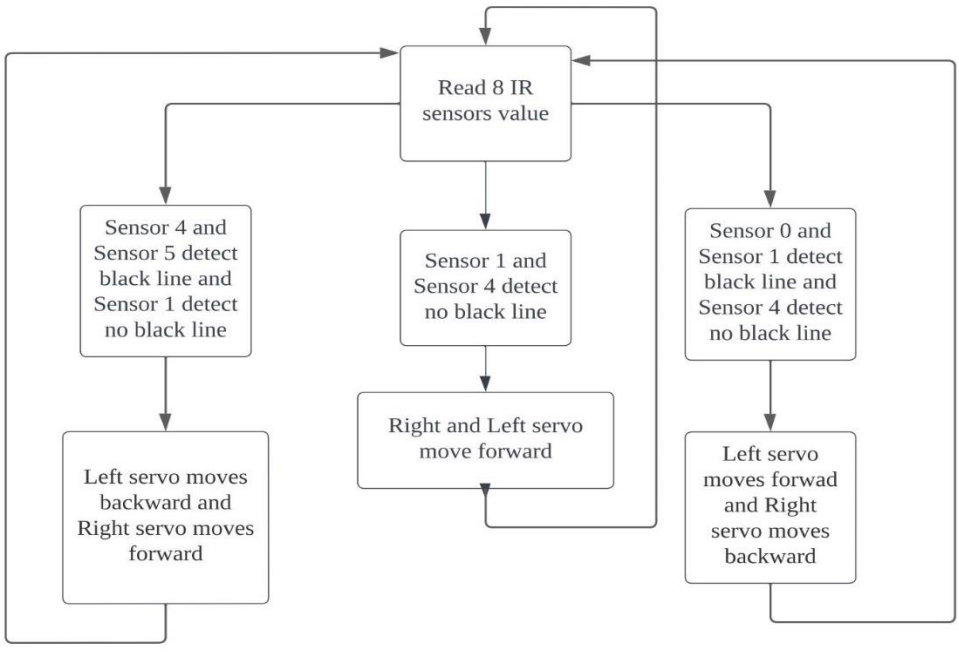
CIRCUIT DIAGRAM



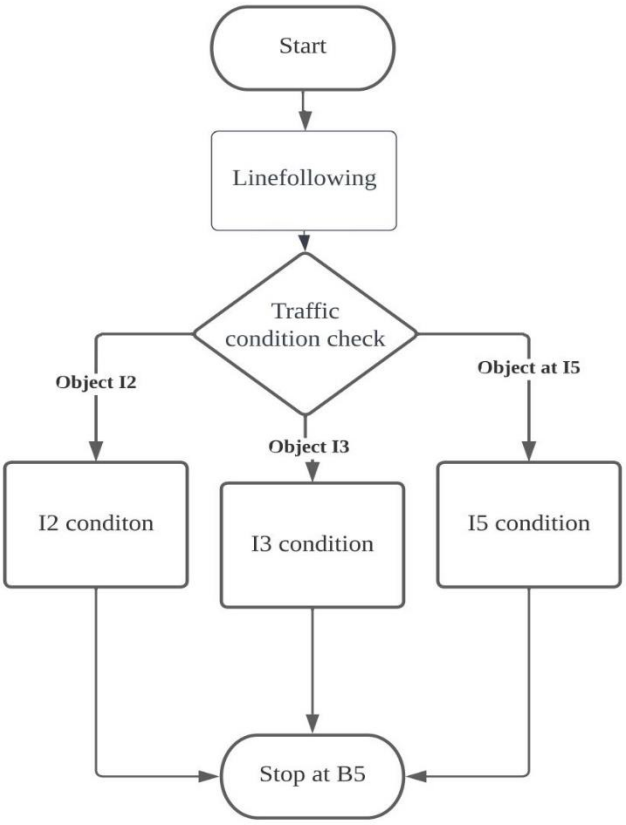
The electronic components are connected to one another in the most optimum way possible. Firstly, the pins on propeller activity board were connected. The 8 pins of the 8RC Reflectance sensor are directly interfaced to the Propeller Activity WX board from pin 7 to pin14, the emitter pin was directly connected to the 5V of the propeller activity board and ground pin was also connected to Gnd of propeller 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 16. 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. Now the connections on raspberry pi were made. The camera port of out pi board was not working so an USB camera was directly connected to one of the USB 2.0 Port and the knocking mechanism's servo signal pin was also connected to pi board on GPIO pin 23. Two LEDs were also connected to the pi board to signal when it detects an enemy/friendly. Also, a GPIO pin from raspberry pi was connected to propeller board for sending signal to propeller board to stop when the pi camera detects a Friendly/Enemy in its path. Since the propeller board was powering the knocking mechanism servo while the pi board was signaling it to rotate and also since pi board and propeller board were communicating with one another a common ground should be made for both the micro-controllers to work properly. The raspberry pi board was powered by using a portable power bank of 5V and 2.4Amp specifications by using the Micro-USB-B port on the raspberry pi board.

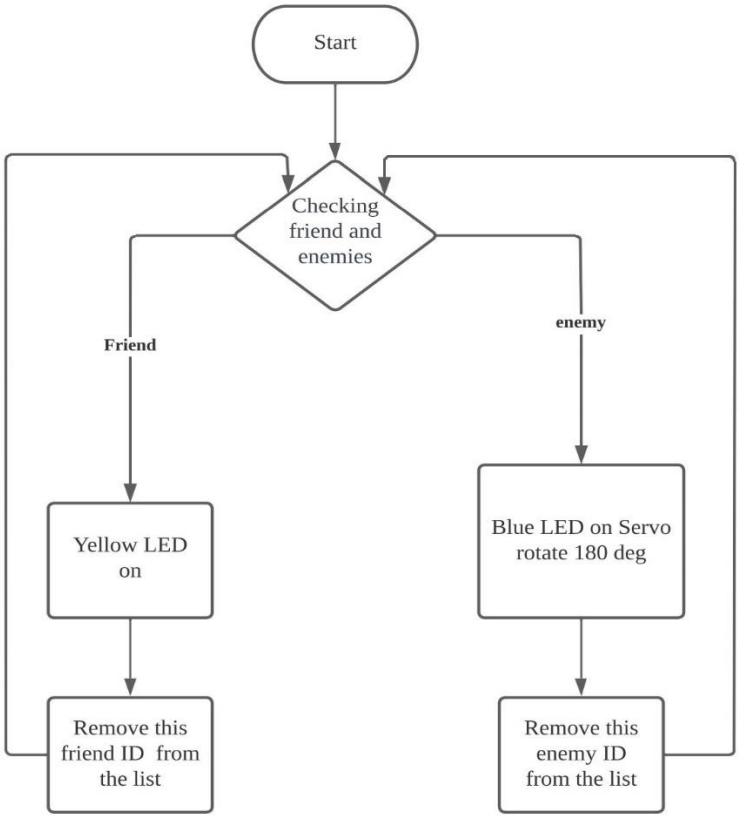
BLOCK DIAGRAM



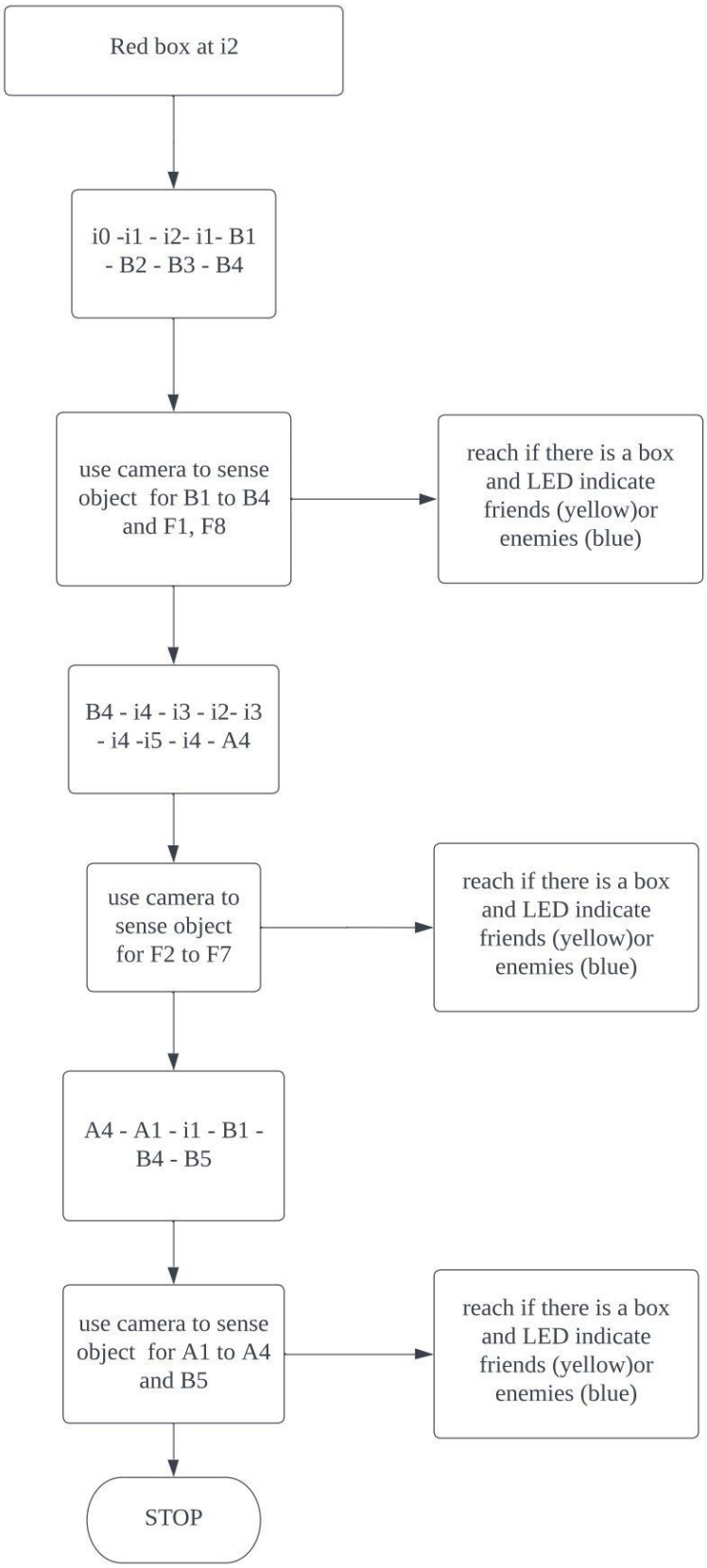
IR sensor Detection and working



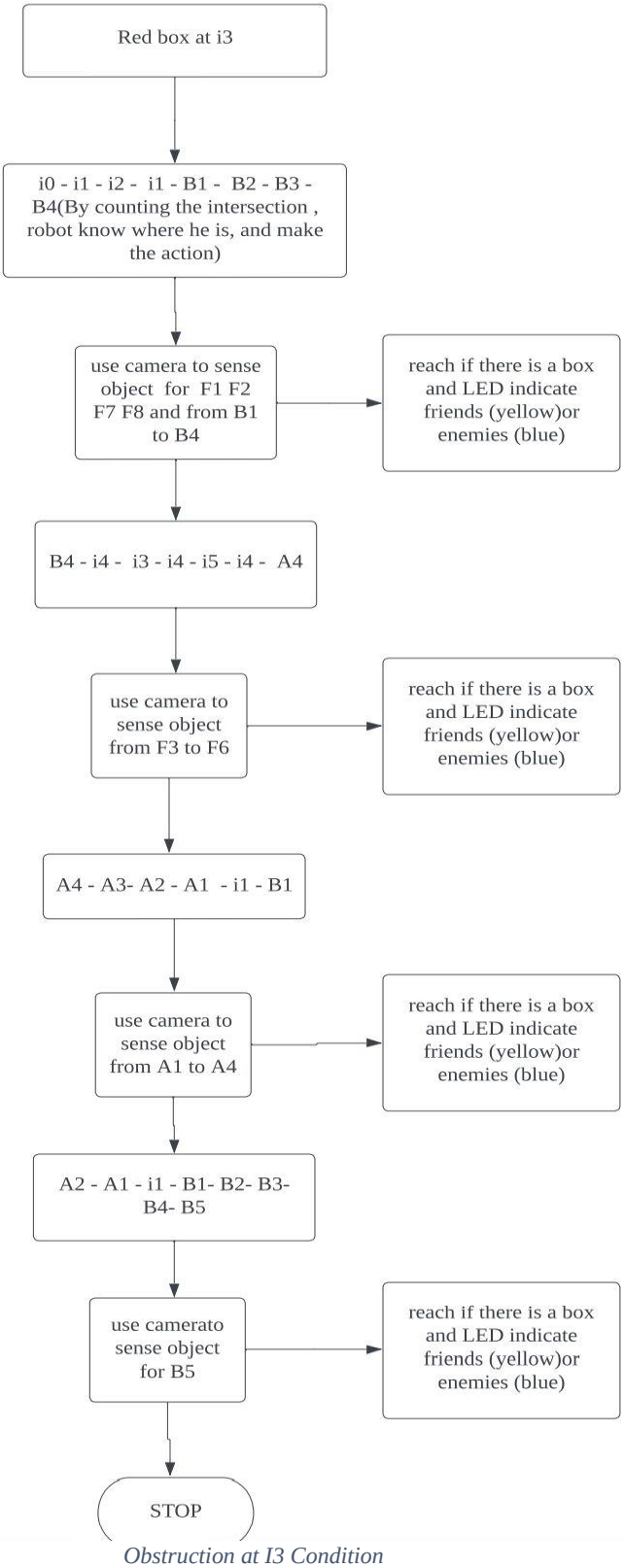
Obstruction Detection and Condition Check



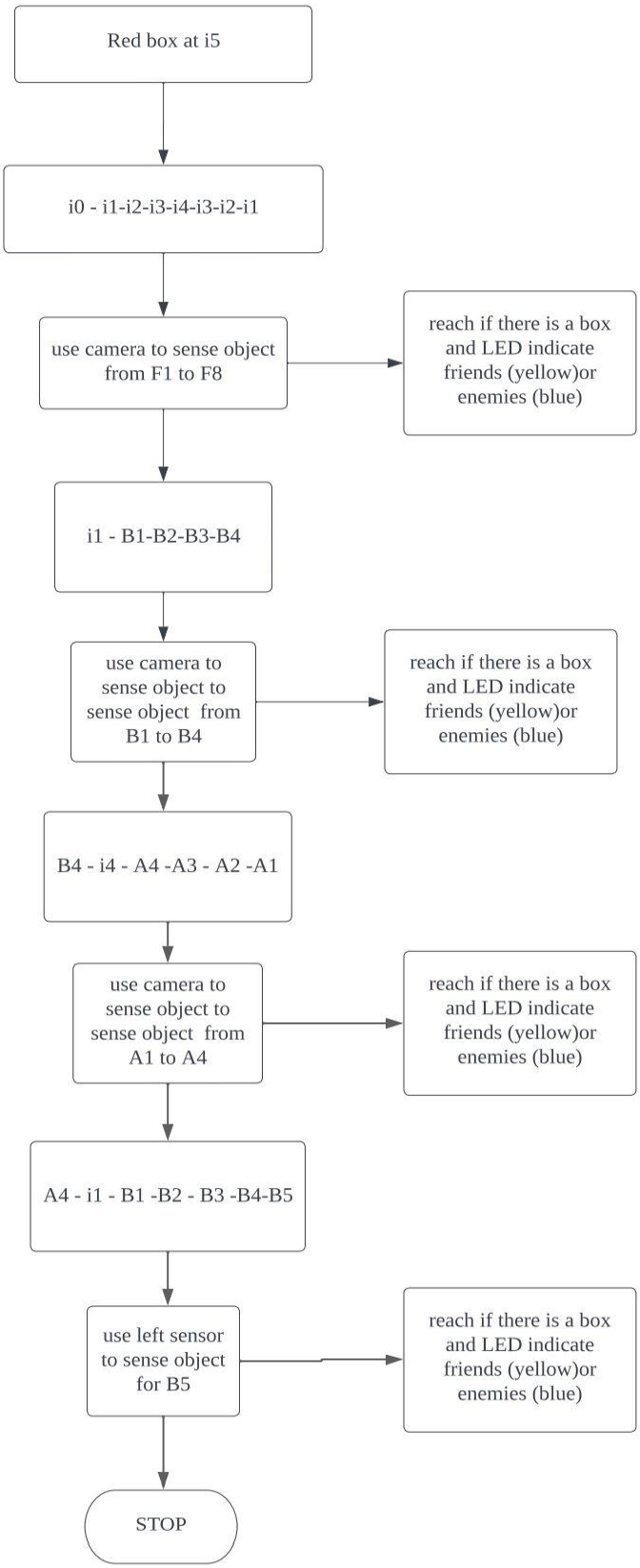
Detecting Friendly/Enemy



Obstruction at I2 Condition



Obstruction at I3 Condition



Obstruction at I5 Condition

DISCUSSION AND CONCLUSION

The robot achieves the given objective i.e., using two micro-controllers one is raspberry pi and other was Propeller activity board. Next the robot starts from one of the safe houses and always follows line (Black tape). Once the robot reaches the start position the robot always 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. Also, while avoiding this traffic congestion and moving around it also detects the Friendlies/Enemies present in either A1 to A5 or B1 to B5 or F1 to F8. It detects these with the use of camera by detecting the Aruco tag placed on the front of them on the podium. If it detects a friendly it signals with yellow LED light and when it detects an enemy, it signals by using the blue LED. If it detects an enemy, it knocks it off by using the knocking mechanism.

The Robot achieves all the given tasks and objectives. The project uses two micro-controllers propeller activity board and raspberry pi. Coming to Propeller board, 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. Coming to Raspberry pi, the prominent problems were powering the raspberry pi, camera angle and communication between propeller and pi. First powering the pi board was a problem because it requires at least 5V 3Amp but most of the commercial power-banks available in the market gives only 2.4Amps maximum we could use Li-Po but the capacity of it was very less and also if we use a power-bank it also adds additional weight to the robot. Secondly the camera angles the camera should be placed at an angle to detect the tags on podium efficiently and it should also be placed in a way that the field of view was good enough. And also, the raspberry pi camera was delicate and gets damaged if not handled properly. Our Raspberry pi camera was damaged, and we were forced to use an USB camera which made things a little complicated due to power issues, magnifications, and camera size. Thirdly the communication between propeller and pi was somewhat difficult and as both clock speeds were different, and they should be able to communicate properly so that they execute the tasks properly.

Overall, 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 an In-house delivery bot in large office spaces to replace the man force deployed to transport various documents in the workspace. Also, by using the Tag detection function in the bot it can also be used in Warehouse automation where it can detect a particular pellet and

transfer the contents properly. Finally, by using a proper power source and calibrating the sensors used in the Robot properly, the Robot can efficiently perform the given tasks in the known environments.

APPENDIX

Code:

The programming of the Propeller Activity WX board was done using simple IDE. Libraries like Servo, ping was used to interface the continuous servo motors and ultrasonic sensors. And the code of raspberry pi was done in Python 3.7 and uploaded to raspberry pi

Propeller Code:

```
/*
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 followline2();
void readdata();
void check();
void check2();
void stop();
int location_check();
void IRsense();
```

```
void reach();
void reach2();
void reach3();
void turnright2();
void turnright3();
// 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;
int objtag = 0;
int traffictag = 0;
int EEEE = 0;
int checkflag = 0;
int turnflag = 0;


void camcheck()
{
while(1){
pause(100);
EEEE = input(0);
pause(100);
//printf("flagdebug =%d\n",EEEE);
if (EEEE == 1){
turnflag = 1;
}
}
}
```

```
void stop()
{
servo_disable(16);
servo_disable(17);
pause(2000);
}
void IRsense(){
while(1)
{
//read values from all the 8 sensors
high(7);
high(8);
high(9);
high(10);
high(11);
high(12);
high(13);
high(14);

pause(100);
time1=rc_time(7,1);
time2=rc_time(8,1);
time3=rc_time(9,1);
time4=rc_time(10,1);
time5=rc_time(11,1);
time6=rc_time(12,1);
time7=rc_time(13,1);
time8=rc_time(14,1);
}
}



void reach()
{
servo_speed(16,-40);
servo_speed(17,-50);
pause(3100);
if (EEEE == 1)
{
stop();
//turnflag == 0;
}
}



void reach2()
{
servo_speed(16,-20);
```

```
servo_speed(17,-50);
pause(900);
}



void reach3()
{
servo_speed(16,20);
servo_speed(17,50);
pause(2200);//servo_speed(16,-40);
//servo_speed(17,-50);
//pause(1400);
}

void turnright3()
{
check();
servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
//tunrright90 deg
servo_speed(16,20);
servo_speed(17,50);
pause(1050);
//servo_speed(16,-20);
//servo_speed(17,-50);
//pause(900);
}

void turnright()
{
check();
servo_speed(16, -50);
servo_speed(17, 20);
pause(500);
//tunrright90 deg
servo_speed(16,20);
servo_speed(17,50);
pause(950);
//servo_speed(16,-20);
//servo_speed(17,-50);
//pause(900);
}

void turnright2()
{
check();
servo_speed(16, -50);
servo_speed(17, 20);
```

```
pause(500);
//tunrright90 deg
servo_speed(16,20);
servo_speed(17,50);
pause(1100);
//servo_speed(16,-20);
//servo_speed(17,-50);
//pause(900);
}



void turnleft()
{
check();
servo_speed(16, -50);
servo_speed(17, 20);
pause(700);
servo_speed(16,-40);
servo_speed(17,-50);
pause(1450);
}



void joint_check(){//check if the interseciton point found
while(1){
while(checkflag == 0)
{
//print("%csensor2 = %d, sensor3 = %d%, sensor6 = %d%, sensor7 = %d%c", HOME,
sensor2,
//sensor3,sensor6,sensor7, CLREOL);
pause(100);
//intersection point found condition
if ( time3 > 930&& time4 > 800&& time5 >930 && time6 > 930 && time7 > 930 && time8
> 930)
{
//count


//servo_speed(11 ++;
//count3 = count1 - 3;
//6, -40);
//servo_speed(17, 20);
if (count1>=1)
{
count1 ++;
joint_flag = 1;
pause(600);
high(5);
```

```
pause(600);
low(5);
}
//avoid first interscetion which is the curve one
else if (count1 == 0 )
{
count1 ++;
joint_flag = 1;
pause(3000);

}
if(count1 == 4 && traffictag == 1 ) //i2 condition
{
redbox_flag2 = I2_STATE;
}
else if(count1 == 5 && traffictag == 1) //i3 condition
{
redbox_flag2 = I3_STATE;
}
else if(count1 == 7 && traffictag == 1) //i5 condition
{
redbox_flag2 = I5_STATE;
}
else{
redbox_flag2 = 1; //meets no conditions
}
}
else
{ joint_flag = 0;
low(6);
}
}
pause(1);
continue;
}
}



void check()
{
if(EEEE == 1 ) //if left side of the sensor sense the friend or enemies and no traffic at front
{
checkflag = 1;
servo_disable(17);
servo_disable(16);
pause(4000);
checkflag = 0;
}
```

```
else if( cmDist3 < 3 && traffictag == 0) //if only left side of the sensor sense the friend or enemies and traffic at the front
{
//servo_speed(17, -50);
//servo_speed(16, 20);
//pause(300);
reach();
servo_speed(17, -50);
servo_speed(16, 20);
pause(800);
traffictag = 1;
}

else if( cmDist3 < 3 && traffictag == 2) //if only left side of the sensor sense the friend or enemies and traffic at the front
{

reach();
servo_speed(17, -50);
servo_speed(16, 20);
pause(800);
traffictag = 1;
}

else
{
pause(1);
}
}




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, -50);
servo_speed(17, 20);
check();
}
else if (((time3 > 400 && time2 > 400)|| (time2 >400)) && (time7 < 400) )
{ // sees line on right side, is making a right turn
servo_speed(16, -20);
servo_speed(17, 20);
//check();
```

```
}
else if (((time6 > 400 && time7 > 400)||(time7>400)) && ( time2 < 400||
time1<400) )
{ // sees line on left side, is making a left turn
servo_speed(16, -40);
servo_speed(17, 0);
//check();
}
else
{ // both are seeing line, goes straight until light is seen
servo_speed(16, -50);
servo_speed(17, 20);
check();
}
}
void followline2(){

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, -50);
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);
//check();
}
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);
//check();
}
else
{ // both are seeing line, goes straight until light is seen
servo_speed(16, -50);
servo_speed(17, 20);
}
}



void readdata()//
{
while(1)
// Repeat indefinitely
{
```

```
//rightsensor =cmDist1, leftsensor= cmDist2, middlesensor = cmDist3
pause(100);
//cmDist1 = ping_cm(4);
cmDist2 = ping_cm(4);
pause(100);
cmDist3 = ping_cm(2); // Get cm distance from Ping)))
pause(100);



while(cmDist3 < 10 )
{
//location object detected
traffictag == 1;
break;
}
// Wait 1/5 second
}
}



void i2_state(){
while(1){
//when no tag is detected
while(checkflag == 0){
if ( count1 == 4 && joint_flag == 1)
{
turnright2();
followline();
continue;
}
else if (count1 == 5 && joint_flag == 1 && cmDist3 < 20)
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
servo_speed(16,20);
servo_speed(17,50);
pause(1000);
stop();

followline();
continue;
}
else if (count1 == 5 && joint_flag == 1 && cmDist3 > 20)
{//turn right
turnright();
followline();
continue;
```

```
}
else if (count1 == 8 && joint_flag == 1 && cmDist2 < 15)
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(1000);
stop();
servo_speed(16,20);
servo_speed(17,50);
pause(1100);
followline();
continue;
}
else if (count1 == 8 && joint_flag == 1 && cmDist2 > 15)
{//turn right
turnright();
followline();
continue;
}
else if (count1 == 9 && joint_flag == 1)

{//turn right
traffictag = 2;
turnright();
followline();
continue;
}
else if (count1 == 13 && joint_flag == 1)
{// turn left

reach();
servo_speed(17, -50);
servo_speed(16, 20);
pause(800);
followline();
continue;
}

else if (count1 == 14 && joint_flag == 1 )
{//turn right
turnleft();
followline();
continue;
}
else if (count1 == 15 && joint_flag == 1 && cmDist3 < 20)
{//turn right

servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
```

```
servo_speed(16,20);
servo_speed(17,50);
pause(1000);
stop();
followline();
continue;
}
else if (count1 == 15 && joint_flag == 1 && cmDist3 > 20 )
{//turn right

turnright();
followline();
continue;
}
else if (count1 == 18 && joint_flag == 1 && cmDist2 < 15 )
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(1000);
stop();
servo_speed(16,20);
servo_speed(17,50);
pause(1200);
followline();
continue;
}
else if (count1 == 18 && joint_flag == 1 && cmDist2 > 15 )
{//turn right
turnright();
followline();
continue;
}
else if (count1 == 20 && joint_flag == 1 )
{//turn right
turnright();
followline();
continue;
}
else if (count1 == 24 )
{
servo_disable(16);
servo_disable(17);
pause(10000);
}
else
{followline();
}
}
// when tag is detected, it will stop for 2 seconds
servo_disable(16);
```

```
servo_disable(17);
pause(2000);
continue;
}
}



void i3_state(){ //
//int count4;
while(1){
//count4 = count1 - 3;
if (count1 == 6 && joint_flag == 1)
{//turn right
turnright();
followline();
continue;
}


else if (count1 == 7 && joint_flag == 1 && cmDist3 < 20 )
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
servo_speed(16,20);
servo_speed(17,50);
pause(1000);
stop();
followline();
continue;
}

else if (count1 == 7 && joint_flag == 1 && cmDist3 > 20)
{//turn right
turnright();
followline();
continue;
}


else if (count1 == 10 && joint_flag == 1 && cmDist2 < 15)


{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(1000);
stop();
servo_speed(16,20);
```

```
servo_speed(17,50);
pause(1200);
followline();
continue;
}

else if (count1 == 10 && joint_flag == 1 && cmDist2 > 15)


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

else if (count1 == 11 && joint_flag == 1)
{// turn left
traffictag = 0;
turnright();
followline();
continue;
}
else if (count1 == 13 && joint_flag == 1)
{// turn left
reach();
servo_speed(17, -50);
servo_speed(16, 20);
pause(800);

followline();
continue;
}
else if (count1 == 14 && joint_flag == 1 )
{//turn right
turnleft();
followline();
continue;
}
else if (count1 == 15 && joint_flag == 1 && cmDist3 < 20 )
{//turn right

servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
servo_speed(16,20);
servo_speed(17,50);
pause(1000);
stop();
followline();
continue;
```

```
}

else if (count1 == 15 && joint_flag == 1 && cmDist3 > 20)
{//turn right

turnright();
followline();
continue;
}

else if (count1 == 18 && joint_flag == 1 && cmDist2 < 15)
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(1000);
stop();
servo_speed(16,20);
servo_speed(17,50);
pause(1200);
followline();
continue;
}

else if (count1 == 18 && joint_flag == 1 && cmDist2 > 15)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 20 && joint_flag == 1 )
{//turn right
turnright();
followline();
continue;
}
else if (count1 == 24 )
{
servo_disable(16);
servo_disable(17);
pause(10000);
continue;
}
else
{followline();
continue;
}
}
}
```

```
void i5_state() { //
//int count4;
while(1){
//count4 = count1 - 3;
if ( count1 == 10 && joint_flag == 1)
{
turnright();
followline();
continue;
}
else if (count1 == 11 && joint_flag == 1 && cmDist3 < 20)
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
servo_speed(16,20);
servo_speed(17,50);
pause(1000);
stop();
followline();
continue;
}

else if (count1 == 11 && joint_flag == 1 && cmDist3 > 20)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 14 && joint_flag == 1 && cmDist2 > 15)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 14 && joint_flag == 1 && cmDist2 < 15)
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(1000);
stop();
servo_speed(16,20);
servo_speed(17,50);
pause(1100);
followline();
continue;
```

```
}
//else if (count1 == 11 && joint_flag == 1 && cmDist2 > 30)
//{//turn right
//check2();
//turnright();
//followline();
//continue;
//}
else if (count1 == 16 && joint_flag == 1 && cmDist3 < 20)
{// turn left
servo_speed(16, -50);
servo_speed(17, 20);
pause(800);
servo_speed(16,20);
servo_speed(17,50);
pause(1000);
stop();
followline();
continue;
}
else if (count1 == 16 && joint_flag == 1 && cmDist3 > 20)
{// turn left
turnright();
followline();
continue;
}

else if (count1 == 19 && joint_flag == 1 && cmDist2 > 15)
{//turn right
turnright();
followline();
continue;
}

else if (count1 == 19 && joint_flag == 1 && cmDist2 < 15)
{//turn right
servo_speed(16, -50);
servo_speed(17, 20);
pause(1000);
stop();
servo_speed(16,20);
servo_speed(17,50);
pause(1100);
followline();
continue;
}

else if (count1 == 21 && joint_flag == 1)
{// turn left
turnright3();
```

```
followline();
continue;
}


else if (count1 == 25 )
{
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, 256);
//cog4 = cogstart((void*)readdata, NULL, stack4, sizeof(stack4));
cog_run(camcheck,256);
//cog6 = cogstart((void*)redbox_check, NULL, stack6, sizeof(stack6));
pause(500);


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, -50);
servo_speed(17, 20);
check();
}
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);
//check();

}
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);
//check();
}



else
{ // both are seeing line, goes straight until light is seen
servo_speed(16, -50);
servo_speed(17, 20);
check();
}

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



}
}
```

Raspberry Pi Code:

```
import cv2
import cv2.aruco as aruco
import RPi.GPIO as GPIO
import time
import numpy as np

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(23, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)
GPIO.setup(22, GPIO.OUT)


cap = cv2.VideoCapture(0)

def Aruco(img, markersize = 6, total = 250, draw = True):

    key = getattr(aruco, f'DICT_{markersize}X{markersize}_{total}')
    Dictonary = aruco.Dictionary_get(key)
    Para = aruco.DetectorParameters_create()
    bbox, ids,_ = aruco.detectMarkers(img, Dictonary, parameters = Para)
    print(ids)

    if draw :

        aruco.drawDetectedMarkers(img, bbox)

    return bbox, ids

def Servomove(Servoangle):
    pulsew = Servoangle*9.72 + 500
    pulsewidth = pulsew/1000000
    for indx in range(0,2):
        GPIO.output(23,1)
        time.sleep(pulsewidth)
        GPIO.output(23,0)
        time.sleep(0.1)

def roi(img):
    height = img.shape[0]
    height = height + 100
    width = img.shape[1]
    grey_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
    #ROI_up = np.array([[(117, 10), (117, 200), (510, 200), (510, 10)]], dtype = np.int32)
    ROI = np.array([[(130,height),(130,180),(510,180),(510,height)]], dtype = np.int32)
    black = np.zeros_like(grey_img)
    roi = cv2.fillPoly(black, ROI, 255)
    roi_img = cv2.bitwise_and(grey_img,roi)
    return roi_img

count69 = 0
count0 = 0
count1 = 0
count2 = 0
count3 = 0
count4 = 0
count5 = 0
count6 = 0
count7 = 0
count8 = 0
count9 = 0
count10 = 0
count11 = 0
count12 = 0
count13 = 0
count14 = 0
count15 = 0
count16 = 0
count17 = 0
count18 = 0
count19 = 0
count = count10 + count11 + count12 + count13 + count14 + count15 + count16 + count17 +
count18 + count19
x = 10
y = 180

while True:
    GPIO.output(17,0)
    _, img = cap.read()
    roi_img = roi(img)
    bbox, ids = Aruco(roi_img)
    tatti = int(ids or 69)

    if count69 == 0:
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
        count69 = count69 + 1
```

```
if tatti == 10:
    count10 = count10 + 1
    if count10 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
elif tatti ==
    count11 = count11 + 1
    if count11 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
elif tatti == 12:
    count12 = count12 + 1
    if count12 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
elif tatti == 13:
    count13 = count13 + 1
    if count13 == 1:
        GPIO.output(17,1)
        time.sleep(0.5)
        GPIO.output(27,1)
        GPIO.output(17,0)
        time.sleep(0.5)
```

```
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
    elif tatti == 14:
       count14 = count14 + 1
       if count14 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
    elif tatti == 15:
       count15 = count15 + 1
       if count15 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)

    elif tatti == 16:
       count16 = count16 + 1
       if count16 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
    elif tatti == 17:
       count17 = count17 + 1
```

```
    if count17 == 1:
        GPIO.output(17,1)
        time.sleep(0.3)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
elif tatti == 18:
    count18 = count18 + 1
    if count18 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
elif tatti == 19:
    count19 = count19 + 1
    if count19 == 1:
        GPIO.output(17,1)
        time.sleep(0.3)
        GPIO.output(17,0)
        GPIO.output(27,1)
        time.sleep(0.5)
        GPIO.output(27,0)
        Servomove(x)
        time.sleep(0.2)
        Servomove(y)
        time.sleep(0.2)
elif tatti == 0:
    count0 = count0 + 1
    if count0 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 1:
```

```
    count1 = count1 + 1
    if count1 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 2:
    count2 = count2 + 1
    if count2 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 3:
    count3 = count3 + 1
    if count3 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 4:
    count4 = count4 + 1
    if count4 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 5:
    count5 = count5 + 1
    if count5 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 6:
    count6 = count6 + 1
    if count6 == 1:
```

```
        GPIO.output(17,1)
        time.sleep(0.3)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 7:
    count7 = count7 + 1
    if count7 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 8:
    count8 = count8 + 1
    if count8 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)
elif tatti == 9:
    count9 = count9 + 1
    if count9 == 1:
        GPIO.output(17,1)
        time.sleep(2)
        GPIO.output(17,0)
        GPIO.output(22,1)
        time.sleep(0.5)
        GPIO.output(22,0)



if cv2.waitKey(1) == 113:
    break
cv2.imshow("Live Feed", roi_img)
```