Checkpoint # 4 Report

1. Purpose:

The objective of this checkpoint is to empower the robot to locate a particular object or a designated signal. In this phase, the robot is programmed to identify a beacon emitting two distinct IR signals using its IR sensors. With this capability, the mobile robot can navigate towards the targeted beacon.

2. Description of Design:

For the sake of ease in testing and debugging, this checkpoint exclusively focuses on implementing the Arduino board. It is essential to clarify that this phase does not involve any communication with the Raspberry Pi using ROS. The Arduino code is meticulously crafted to directly detect IR signals from the beacon by connecting to Arduino pins through digital input.

Initially, the robot executes the same actions as in the previous checkpoint, engaging in tasks such as locating the light ball and avoiding obstacles through touch and light sensors. Upon the activation of the bottom touch sensor, the robot transitions into a new mode dedicated to locating the beacon with the IR sensor.

During this process, once the touch sensor on the base is no longer pressed, indicating the absence of the light ball, the mode reverts to the initial objective of finding the ball. This adaptive approach equips the robot with the capability to identify specific objects and navigate towards predefined targets.

Arduino:

(1) Checkpint4.ino

```
// include
#include<ros.h>
#include<std_msgs/Int32.h>

// pins
#define EnA 5
#define EnB 6
#define In1 8
#define In2 9
#define In3 10
#define In4 11
#define In5 7  // ball
#define In6 12 // left
#define In7 13 // right
#define Light A4 // light
#define Beacon 3 // IR

// IR and beacon
int door_frequency_1 = 600;
int door_frequency_2 = 1500;
```

```
float rate;

// ros
ros::NodeHandle nh;
std_msgs::Int32 light;
std_msgs::Int32 IR;
ros::Publisher IR_publisher("IR_feedback", &IR);
ros::Publisher light_publisher("light_feedback", &light);

// touch sensor and light sensor data
int catch_ball = 0;
int right_collision, left_collision = 0;
int light_data;

// IR
int count_0 = 0;
int count_1 = 1;
int cnt = 0;

//robot control function
void move_robot(int right,int left)
{
    if(right>0)
    {
        digitalWrite(In1, LOW);
        digitalWrite(In2, HIGH);
    }
    else
    {
        digitalWrite(In1, HIGH);
        digitalWrite(In2, LOW);
        right *= -1;
    }
    if(left>0)
    {
        digitalWrite(In3, LOW);
        digitalWrite(In4, HIGH);
    }
    else
    {
        digitalWrite(In3, HIGH);
        digitalWrite(In4, LOW);
        left *= -1;
    }
    analogWrite(EnA, right);
    analogWrite(EnB, left);
}
```

```
void foward()
{
    move_robot(155,146);
}

void backward()
{
    move_robot(-105,-100);
}

void right()
{
    move_robot(100,135);
}

void left()
{
    move_robot(135,100);
}

void left_spin()
{
    move_robot(125,-120);
}

void right_spin()
{
    move_robot(-125,120);
}

void robot_stop()
{
    move_robot(0,0);
}

void collision()
{
    int sec = 0;
    while(sec <= 10)
    {
        backward();
        sec++;
        delay(100);
    }
}

void robot_pause()
{
```

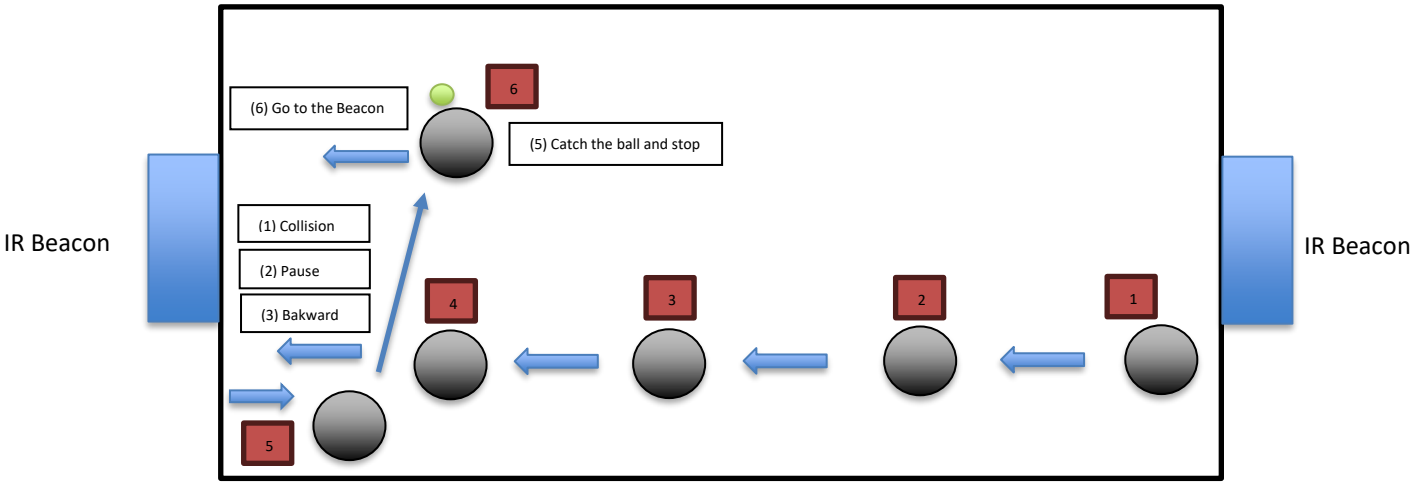
```
    int sec = 0;
    while(sec <= 10)
    {
        robot_stop();
        sec++;
        delay(100);
    }
}

// light sensor and find light ball
bool find_ball()
{
    int dist;
    catch_ball = digitalRead(In5);
    left_collision = digitalRead(In6);
    right_collision = digitalRead(In7);
    light_data = analogRead(Light);
    if(!left_collision || !right_collision)
    {
        backward();
        delay(300);
        scan();
    }
    else
    {
        foward();
    }
    if(!catch_ball)
    {
        return true;
    }
    return false;
}

void scan()
{
    int index;
    int light_max=1000;
    int range=30;
    for(int i=0;i< 60; i++)
    {
        light_data=analogRead(Light);
        if (light_data < light_max)
        {
            light_max=light_data;
          index=i;
       }
       right_spin();
```

```
        delay(50);
        robot_stop();
        delay(10);
    }
    robot_stop();
    delay(1000);

    for(int j=0;j<index;j++)
    {
        light_data = analogRead(Light);
        if((light_data <= (light_max + range)) && ( light_data >= (light_max -
range)))
        {
            robot_stop();
        }
        else
        {
            right_spin();
            delay(50);
            robot_stop();
            delay(10);
        }
    }
}

// beacon and IR sensor
void find_beacon(float rate)
{
    left_spin();
    delay(50);
    if (rate>=20)
    {
        foward();
        delay(5000);
        //stop_moving();
        //delay(5000);
    }
}

float read_ir()
{
    int val;
    float rate;
    cnt++;
    val = digitalRead(Beacon);
     if(val) count_1++;
     if(!val) count_0++;
     rate = count_0*100/(count_0+count_1);
```

```
    return rate;
}

void setup()
{
    //ros
    nh.initNode();
    nh.advertise(IR_publisher);
    nh.advertise(light_publisher);
    Serial.begin(57600);

    //pin
    pinMode(EnA, OUTPUT);
    pinMode(EnB, OUTPUT);
    pinMode(In1, OUTPUT);
    pinMode(In2, OUTPUT);
    pinMode(In3, OUTPUT);
    pinMode(In4, OUTPUT);
    pinMode(In5, INPUT);
    pinMode(In6, INPUT);
    pinMode(In7, INPUT);
    pinMode(Light ,INPUT);
    pinMode(Beacon ,INPUT);
    Serial.begin(115200);
}

void loop()
{
    int ir_msg;
    bool start = true;
    bool ball = false;
    bool find_door = false;
    cnt++;
    ir_msg = read_ir();
    if(cnt % 50 == 0)
    {
        if(start)
        {
            ball = find_ball();
            if(ball)
            {
                find_beacon(ir_msg);
            }
        }
    }
    if(cnt%100==0)
    {
        cnt = 0;
```

```
        count_0 = 0;
        count_1 = 0;
    }
    delay(1);
}
```

3. Result:



4. Discussion

The primary goal of this checkpoint is to enable the robot to locate a specific object or signal, specifically a beacon emitting two distinct IR signals. The implementation involves programming the robot to identify the beacon using its IR sensors. This capability empowers the mobile robot to navigate towards the targeted beacon.

This checkpoint focuses on the implementation of the Arduino board for testing and debugging convenience. Notably, there is no communication with the Raspberry Pi using ROS in this phase. The Arduino code is carefully designed to directly detect IR signals from the beacon by connecting to Arduino pins through digital input.

The code includes functions for controlling the movement of the robot, such as forward(), backward(), right(), left(), left_spin(), right_spin(), robot_stop(), collision(), and robot_pause(). These functions are crucial for the robot's mobility and interaction with its environment.

The find_ball() function utilizes touch and light sensors to locate the light ball. The robot adjusts its movement based on sensor inputs and can revert to searching for the ball if it is not detected.The scan() function is responsible for scanning the environment using the light sensor to identify the direction of the light ball. It helps the robot orient itself towards the target.

Functions like find_beacon(), read_ir(), and setup() are dedicated to handling the beacon and IR sensors. The read_ir() function calculates the rate of IR signals, and find_beacon() initiates specific movements based on the calculated rate.

The robot's adaptive behavior is notable, as it switches between finding the light ball and locating the beacon based on inputs from touch sensors. This adaptive approach enhances the robot's versatility in different scenarios.

The provided functions for robot movement, sensor handling, and mode switching collectively contribute to the robot's capability to navigate, detect, and adapt to different situations. However, thorough testing and validation will be crucial to ensure the robustness of the overall system in real-world scenarios.