Propeller Project

Yash Patel (yp2285)
Anais Roche (aer523)
Rushi Shah (rs7236)

Hardware Setup

Two continuous servo motors are used for maneuvering the Boebot rover. Values at 1400 indicate no rotation, incrementing will cause the motor to rotate in one direction and decrementing will cause the motor to rotate in the opposite direction. A standard servo motor is used to rotate the PING sensor where the input value is the degree of rotation scaled by 10. (0 is 0, 90 is 900, and 180 is 1800).

The PING ultrasonic sensor is mounted on the standard servo at the front of the bot and is initialized to look for objects 35 centimeters when detecting obstacles, and 15 centimeters when detecting objects/drop off points. There is a 5 centimeter margin from the actual distance to ensure the robot detects the object.

QTR-HD-15A Reflectance Sensor Array is used for line detection. It is an array of 15 IR LED and phototransistor pairs and four of the 15 pairs are selected to be used to detect lines. Values above 1 indicate a black line is detected while values smaller than 1 indicate no line (on a white background). The servo actuation is determined based on the readings of the two inner sensors while the outer two sensors determine intersection detection.

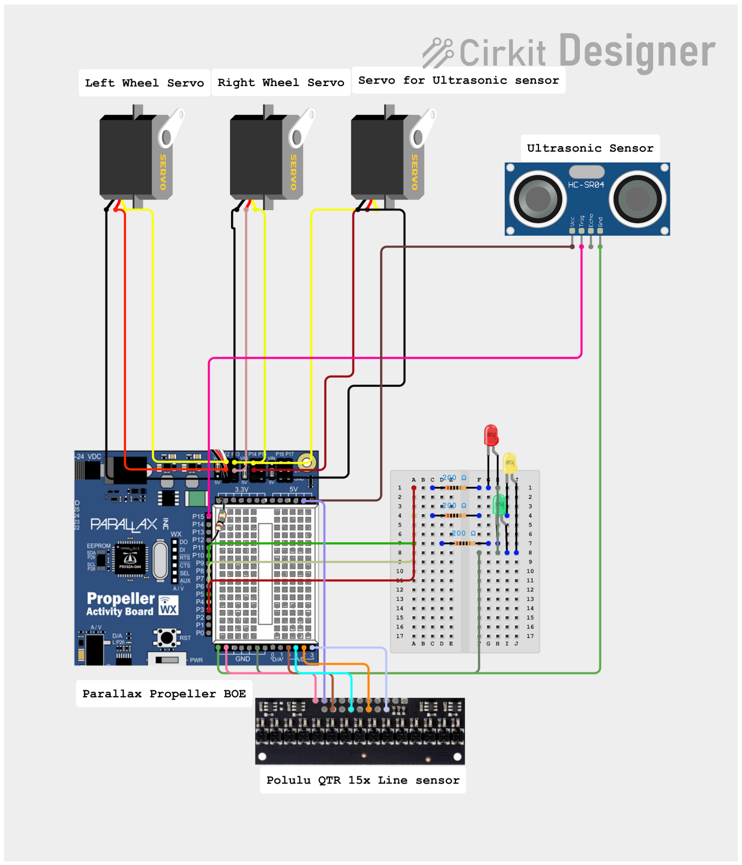

Figure 1: Circuit Diagram

Line tracking using QTR sensors

The read_line_sensor() function initializes the four pins and assigns the values to a vector array v. For efficiency, only 4 sensors are used to detect and follow the line instead of the full array of sensors. This vector v is returned at the end of the function. When the inner two sensors detect values greater than 1, indicating the black line, the bot will move forward. If the inner left sensor detects a value greater than 2, but the inner right sensor detects a value less than one, the bot will make a corrective motion to the right. If the inner right function detects a value greater than 2 and the inner left sensor detects a value less than one, the bot will make a corrective motion to the left. Only when all four sensors detect values greater than 2 will an intersection counter be incremented and the cog that monitors this value will blink the red LED to indicate it has reached an intersection.

Implementation

As one of the tasks was to be able to detect and maneuver around obstacles accordingly, it was important for the robot to be able to differentiate between an obstacle and an object. As the obstacle was only situated at either i2, i3, or i5, the code was separated into these three routines. The first function run through one cog is our navigate() function which executes the line follow function and counts the number of intersections encountered after the home intersection as it traverses the center lane. It runs the ping sensor as well at every intersection detected to search for the obstacle. The bot will stop one intersection away from the detected obstacle and use a yellow LED to indicate the obstacle has been found. Depending on the number of intersections counted until this instance, the next routine will be selected: 2 intersections indicate an object at i2 and the i2_routine() will be selected, 3 intersections indicate i3 and the i3_routine() will be selected, and 5 intersections indicate i5 and the i5_routine() will be selected.

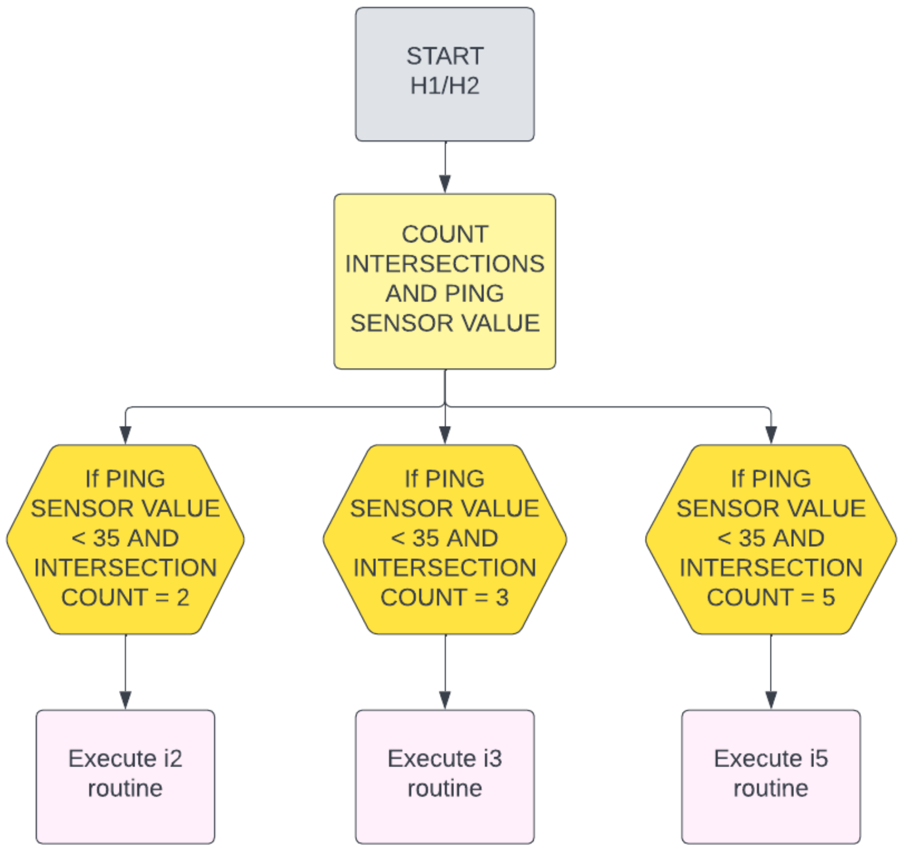
START
H1/H2
COUNT INTERSECTIONS AND PING SENSOR VALUE
If PING SENSOR VALUE < 35 AND INTERSECTION COUNT = 2
If PING SENSOR VALUE < 35 AND INTERSECTION COUNT = 3
If PING SENSOR VALUE < 35 AND INTERSECTION COUNT = 5
Execute i2 routine
Execute i3 routine
Execute i5 routine

Figure 2: Navigate Routine

The i2 routine causes the robot to take an immediate left at the first intersection after encountering the obstacle at i2. It will execute a subroutine for a single block encounter b1_counter_check(). The ping sensor remains face forward and tracks to see if there is an object detected at the next intersection. At this stage, obstacles are no longer considered as it has been detected and evaded. If an object is detected, a counter will increment by one. When this incremented value reaches two the robot will stop and pause for 10 seconds indicating that all objects have been found. A second cog will be monitoring this incremented value. When it is increased upon object detection, this second cog will activate the green LED to indicate an object has been found.

After the b1 intersection has been reached, the bot will turn right and start the b1tob4_counter_check() function. The servo attached to the ping sensor will rotate to the 180 degree position facing left. The line follower function will continue to run, and an intersection counter will also be running. A third cog is monitoring this incremented value and running a function to blink the red LED every time an intersection is detected. If another object is detected, the object counter will be incremented and the bot should stop, indicating both objects have been found.

If no object is detected at B4, the robot will rotate right and the ping sensor will return to the 90 degree position. The b4_a4_counter_check() function will be run. Similar to the b1_counter_check(), as the bot traverses i4 (the only intersection guaranteed not to have an obstacle), the ping sensor will detect if there is an object at A4. At the intersection, if no object has been detected when the bot arrives, then it will rotate right and repeat b1_b4_counter_check() for A4-A1. Based on the direction of the one-way streets on the map, there is no legal route to A5 so the case will be disregarded.

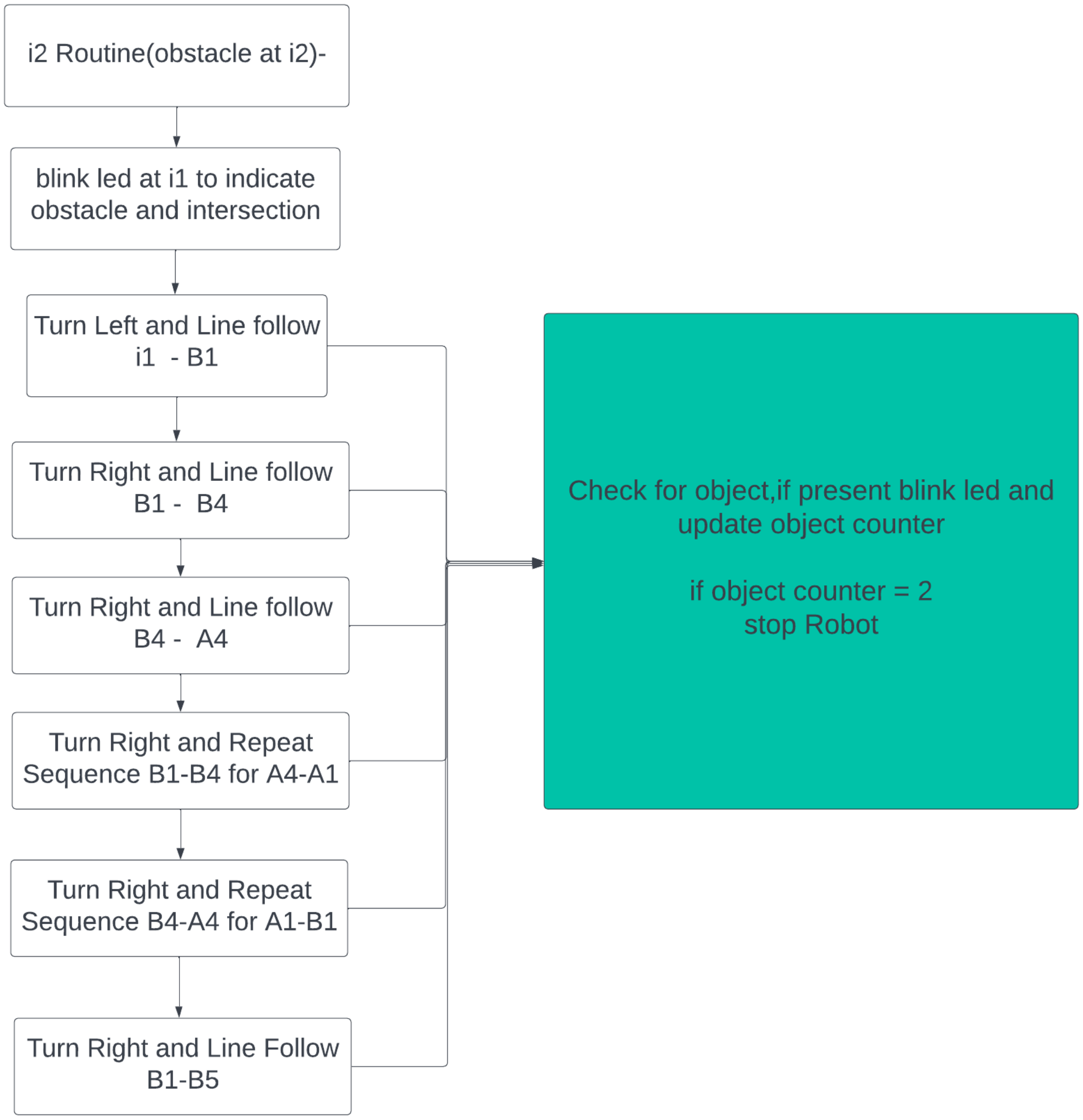
i2 Routine(obstacle at i2)-
blink led at i1 to indicate obstacle and intersection
Turn Left and Line follow i1 - B1
Turn Right and Line follow B1 - B4
Turn Right and Line follow B4 - A4
Turn Right and Repeat Sequence B1-B4 for A4-A1
Turn Right and Repeat Sequence B4-A4 for A1-B1
Turn Right and Line Follow B1-B5
Check for object,if present blink led and update object counter
if object counter = 2 stop Robot

Figure 3: i2 Routine

If there is no object detected at A1, the robot will turn right and run b4_a4_countercheck() again to return to B1. This indicates that an object is at B5 and the robot will run b1_b5_countercheck() where it will continue to count intersections until it reaches B5 and stop once the final object has been detected.

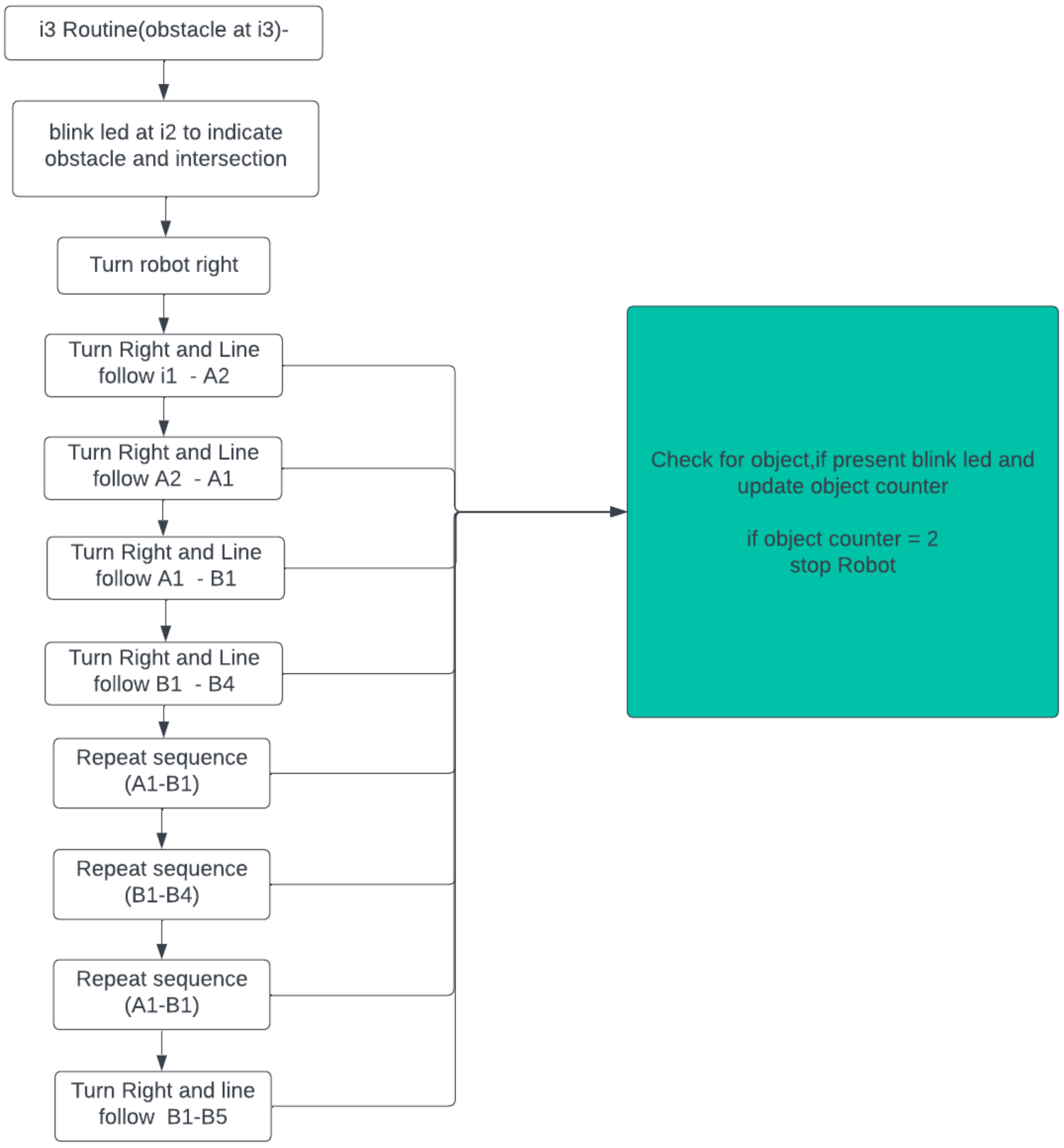
i3 Routine(obstacle at i3)-
blink led at i2 to indicate obstacle and intersection
Turn robot right
Turn Right and Line follow i1 - A2
Turn Right and Line follow A2 - A1
Turn Right and Line follow A1 - B1
Turn Right and Line follow B1 - B4
Repeat sequence (A1-B1)
Repeat sequence (B1-B4)
Repeat sequence (A1-B1)
Turn Right and line follow B1-B5
Check for object,if present blink led and update object counter
if object counter = 2 stop Robot

Figure 4: i3 Routine

Similarly, for the i3 routine the robot will stop one intersection away from the detected obstacle and make a legal right turn to A2 to evade the obstacle. The i3 routine will also run the functions b1_countercheck(), b1_b4_countercheck(), to the i4 intersection and b4_a4_countercheck() in a similar fashion and follow the b1_b5_countercheck() if the second object is not detected previously. The robot will loop through A2 and A1 again as it will follow the A4-A1 trajectory to make sure all spaces have been searched prior to the b5 countercheck routine.

The i5 routine is similar however it does not require the second loop through A2 and A1 like the i3 routine and will simply run b1_b4_countercheck() for A4 to A1 and B1 to B4 before continuing forward to B5 for the final item.

Appendix: Project Code

```
#include "simpletools.h"
#include "abdrive.h"                          // abdrive library
#include "servo.h"
#include "ping.h"                             // Include ping header
#include "adcDCpropab.h"



float * read_line_sensor();
void  line_follow();
void intersection_led_blink(void *par);
void object_detection(void *par);
void bot_forward(int speed_value);
void navigate(void *par1);
void bot_stop();
void i2_routine();
void i3_routine();
void i5_routine();
volatile int count_intersect = 0;
volatile int count_object = 0;
int count_intersect_b1b4 = 0;
unsigned int stack1[40+25];
unsigned int stack2[40+25];
unsigned int stack3[40+25];
int cmDist;
float v[4];
//float v_l[4];
volatile int detection = 0;
volatile int intersect = 0;
int main()                                    // Main function
{
   //adc_init(21,20,19,18);
   cogstart((void*)navigate, NULL, stack2, sizeof(stack2));
   //static int cog_id = cog_start(intersection_led_blink,NULL, 128);
   cogstart((void*)intersection_led_blink,NULL, stack1,sizeof(stack1));
   cogstart((void*)object_detection,NULL, stack3,sizeof(stack3));
}

// read line sensor values interms of
voltage---------------------------------------------------
float * read_line_sensor()
{
      adc_init(21,20,19,18);
      //float v0,v1,v2,v3;
      v[0] = adc_volts(0);
      v[1] = adc_volts(1);
      v[2] = adc_volts(2);
      v[3] = adc_volts(3);
      //print("A/D2 = %f V%c\n", v0, CLREOL);     // Display volts
      //print("A/D3 = %f V%c\n", v1, CLREOL);
      //print("A/D3 = %f V%c\n", v2, CLREOL);
      //print("A/D3 = %f V%c\n", v3, CLREOL);
      return v;
}

//line
follower------------------------------------------------------------------
--------------------
void  line_follow()
{
    float * v;
    v = read_line_sensor();
    //print("A/D3 = %f V%c\n", v[2], CLREOL);

    if (v[1] > 1 && v[2] > 1)
    {
        bot_forward(50);
    }
    else if (v[1]>2 && v[2]<1)
    {
        servo_set(12,1440);  //slight_right
        servo_set(13,1400);
    }
    else if(v[2]>2 && v[1]<1)
    {
      servo_set(12,1400); //slight_left
      servo_set(13,1360);
     }
}

// Intersection LED Blink
Function------------------------------------------------------------------
--------
void intersection_led_blink(void *par)
{
    while(1)
   {
        while(intersect)
        {
            high(3);
            pause(200);
            low(3);
            pause(200);
        }
    }

}

// Object detection LED Blink
Function---------------------------------------------------------------------------
--------
void object_detection(void *par)
{

  while (1)
  {
    while(detection)
    {
        high(5);
        pause(200);
        low(5);
        pause(200);
    }

   }
}



//obstacle LED blink
---------------------------------------------------------------------------
-------
void obstacle_led_blink()
{
    high(4);
    //pause(50);
}

// robot forward
motion---------------------------------------------------------------------
-----
void bot_forward(int speed_value)
{
  int left_motor_speed, right_motor_speed;
  left_motor_speed = 1400 + speed_value;
  right_motor_speed = 1400 - speed_value;
  servo_set(12, left_motor_speed);
  servo_set(13,right_motor_speed);
}

void bot_left()
{
  float * v;
  v = read_line_sensor();
  while(v[1]>2)
  {
    float * v;
    v = read_line_sensor();
    //printf("%d \n", v[1]);
    servo_set(12,1400);
    servo_set(13,1350);
  }

  while(v[1]<2)
  {
    float * v;
    v = read_line_sensor();
    //printf("low= %d \n", v[1]);
    servo_set(12,1400);
    servo_set(13,1350);
  }

}

void bot_right()
{
  float * v;
  v = read_line_sensor();
  while(v[2]>2)
  {
    float * v;
    v = read_line_sensor();
    servo_set(12,1450);
    servo_set(13,1400);
  }
  while(v[2]<2)
  {
    float * v;
    v = read_line_sensor();
    servo_set(12,1450);
    servo_set(13,1400);
  }
}

void bot_stop()
{
  servo_set(12,1400);
  servo_set(13,1400);
}
```

```
int b1_counter_check()
{
   while(1)
   {
      do{
        line_follow();
        }while(v[0]<1 && v[3]<1);

      count_intersect = count_intersect + 1;
      float * v;
      v = read_line_sensor();
      if (v[0]>2 && v[3]>2)
      {
        //int count_object = object_count();
        intersect  =  1;
        int cmDist = ping_cm(15);
        if (cmDist < 15)
        {
           detection = 1;
           count_object = count_object +1;
           if(count_object ==2){
              bot_stop();
              pause(10000);
          }
         }
        break;
      }
      //bot_forward(50);
      //pause(500);
    }

    return count_intersect;
}

int b1_b4_counter_check()
{
   count_intersect = 0;
   while(1)
   {
      do{
        line_follow();
        }while(v[0]<1 && v[3]<1);
      intersect =  1;
      int cmDist = ping_cm(15);
      if (cmDist < 15)
      {
           detection = 1;
           count_object = count_object +1;
          if(count_object ==2){
             bot_stop();
             pause(10000);
           }
        }
      count_intersect = count_intersect + 1;

      float * v;
      v = read_line_sensor();
      //printf("%d \n",count_intersect);
      if (v[0]>2 && v[3]>2 && count_intersect==3)
      {
        int cmDist = ping_cm(15);
        if (cmDist < 15)
        {
          detection = 1;
          count_object = count_object +1;
          if(count_object ==2){
             bot_stop();
             pause(10000);
           }
         }
        break;
      }
      bot_forward(50);
      pause(500);
      intersect = 0;
      detection =0;
    }
    return count_intersect;
}

int b4_a4_counter_check()
{
   count_intersect = 0;
   while(1)
   {
      do{
        line_follow();
        }while(v[0]<1 && v[3]<1);
      intersect =1;
      int cmDist = ping_cm(15);
      count_intersect = count_intersect + 1;
      float * v;
      v = read_line_sensor();
      if (v[0]>2 && v[3]>2 && count_intersect==2)
      {
        //count_object = object_count();
        int cmDist = ping_cm(15);
        if (cmDist < 15)
        {
           detection = 1;
           count_object = count_object +1;
           if(count_object ==2){
              bot_stop();
              pause(10000);
           }
         }
        break;
      }
      bot_forward(50);
      pause(500);
      intersect = 0;
    }

    return count_intersect;
}

int b1_b5_counter_check()
{
   count_intersect = 0;
   while(1)
   {
      do{
        line_follow();
        }while(v[0]<1 && v[3]<1);
      intersect  = 1;
      count_intersect = count_intersect + 1;

      float * v;
      v = read_line_sensor();
      if (v[0]>2 && v[3]>2 && count_intersect==4)
      {
        int cmDist = ping_cm(15);
        if (cmDist < 15)
        {
           detection = 1;
           count_object = count_object +1;
           if(count_object ==2){
              bot_stop();
              pause(10000);
           }
         }
        break;
      }
      bot_forward(50);
      pause(500);
      intersect = 0;
    }
    return count_intersect;
}

void navigate(void *par1)
{
  servo_angle(14,900);
  int cmDist = ping_cm(15);
  float * v;
  v = read_line_sensor();
  while(1)
   {
      do{
        line_follow();
        float * v;
        v = read_line_sensor();
        if (v[0]>1  &&  v[1]>1 &&  v[2]>1  && v[3]>1)
        {
            intersect = 1;
            break;
        }
        }while(1);
      //intersect = 0;
      int cmDist = ping_cm(15);
      count_intersect = count_intersect + 1;
      float * v;
      v = read_line_sensor();

      if (cmDist < 35 && v[0]>2 && v[3]>2)
      {
        obstacle_led_blink();
        intersect  = 1;
        break;
      }
      bot_forward(50);
      pause(500);
      intersect=0;
    }
  intersect = 0;
  if (count_intersect == 2)
   {
      count_intersect = 0;
      i2_routine();
   }

  else if (count_intersect == 3)
  {
    count_intersect = 0;
    i3_routine();
   }

  else if (count_intersect ==  5)
  {
      count_intersect = 0;
      i5_routine();
  }
}

void i2_routine()
{
  bot_left();
  low(4);
  int count_intersect = b1_counter_check();  // to  B1
  intersect =  0;
  detection = 0;
  bot_right();
  //count_intersect = 0;
  servo_angle(14, 1800);
  count_intersect = b1_b4_counter_check(); // B1  to  B4
  bot_right();
  intersect = 0;
  detection = 0;
  servo_angle(14, 900);
  count_intersect = b4_a4_counter_check();  // B4 to A4
  bot_right();
  intersect = 0;
  detection = 0;
  servo_angle(14, 1800);
  count_intersect = b1_b4_counter_check();    //A4 to A1
  bot_right();
  intersect = 0;
  detection = 0;
  servo_angle(14, 900);
  count_intersect = b4_a4_counter_check();   //A1 to B1
  bot_right();
  intersect = 0;
  detection = 0;
  servo_angle(14,1800);
  count_intersect  =  b1_b5_counter_check(); // B1to B5
  intersect =0;
  detection = 0;
  bot_stop();
  pause(2000);
}
```

```
void i3_routine()
{
  bot_right();
  low(4);
  count_intersect = b1_counter_check(); //i2 to  a2
  bot_right();
  intersect = 0;
  detection =  0;
  servo_angle(14, 1800);
  count_intersect = b1_counter_check(); // a2 to a1
  bot_right();
  intersect = 0;
  detection =  0;
  count_object = 0;
  servo_angle(14, 900);
  count_intersect = b4_a4_counter_check(); //a1  to b1
  bot_right();
  intersect = 0;
  detection =  0;
  servo_angle(14, 1800);
  count_intersect = b1_b4_counter_check();
  bot_right();
  intersect = 0;
  detection =  0;
  servo_angle(14, 900);
  count_intersect = b4_a4_counter_check();
  bot_right();
  intersect = 0;
  detection =  0;
  servo_angle(14,  1800);
  count_intersect = b1_b4_counter_check();
  intersect = 0;
  detection =  0;
  count_intersect  =  b1_b5_counter_check(); // B1to B5
  intersect = 0;
  detection =  0;
  bot_stop();

}

void i5_routine()
{
    bot_right();
    low(4);
    count_intersect  = b1_counter_check();
    bot_right();
    intersect = 0;
    detection =  0;
    servo_angle(14,1800);
    count_intersect = b1_b4_counter_check();    //A4 to A1
    bot_right();
    intersect = 0;
    detection =  0;
    servo_angle(14, 900);
    count_intersect = b4_a4_counter_check();   //A1 to B1
    bot_right();
    intersect = 0;
    detection =  0;
    servo_angle(14,  1800);
    count_intersect = b1_b4_counter_check();
    bot_stop();
    intersect = 0;
    detection =  0;
}
```