Line Following Robot Report

Raspberry Pi & Propeller Integration

Prithvi Dev K V, Sarvesh Rathi, Roshan Balu TMB

May 2022

1. Hardware Used

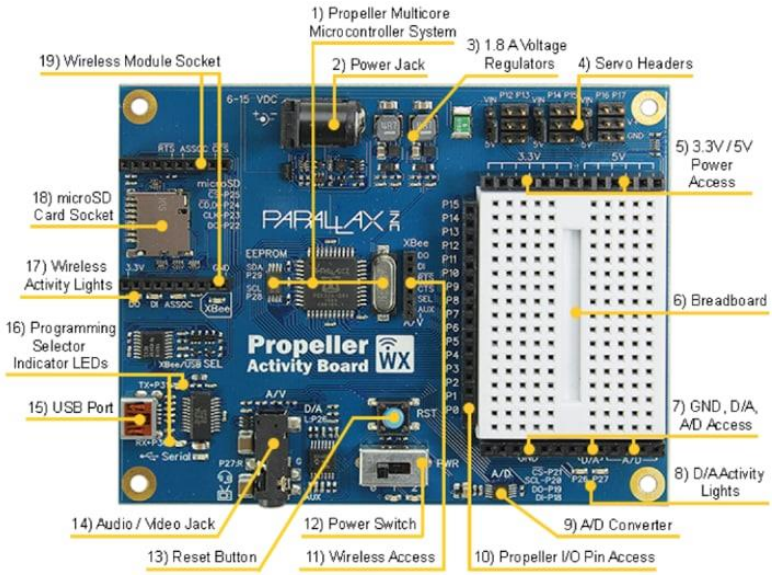
1.1 Raspberry Pi 4 Board

The Raspberry Pi 4 offers ground-breaking increases in processor speed, multimedia performance, memory, and connectivity compared to the prior-generation boards while retaining backward compatibility and similar power consumption. The Raspberry Pi 4 provides desktop performance comparable to entry-level x86 PC systems. The board's key features include a high-performance 64-bit quad-core processor, dual-display output via two Micro HDMI ports, up to 4K resolution, hardware video decoding at up to 4Kp60, up to 4GB of RAM, dual-band 2.4/5.0 GHz wireless LAN, Bluetooth 5.0, Gigabit Ethernet, USB 3.0, and PoE capability. Due to the higher power requirements, the Raspberry Pi 4 requires a 3.0A USB-C power supply. The standard HDMI port that was part of previous generation Raspberry Pi generation boards is replaced on the Raspberry Pi 4 by two Micro HDMI ports to provide dual monitor support.

1.2 Propeller Activity Board

Propeller Activity Board WX features a built-in 8-core Propeller P8X32A microcontroller, 64KB EEPROM, and a 5MHz crystal oscillator. The 8-core MCU is pre-wired to a host of popular peripherals for fast and fun experiments. Learn or teach the basics of electronics, programming, and robotics with this versatile board. The board has independent 1.8A, 3.3V,

and 5V switching regulators. This board also includes a mini TRRS audio/video (or audio/mic) jack, and selectable wireless socket/USB communication between a host computer and Propeller chip.



1.3 IR Sensor

An infrared (IR) sensor is an electronic device that measures and detects infrared radiation in its surrounding environment. There are two types of infrared sensors: active and passive. Active infrared sensors both emit and detect infrared radiation. Active IR sensors have two parts: a light-emitting diode (LED) and a receiver. When an object comes close to the sensor, the infrared light from the LED reflects off of the object and is detected by the receiver. Active IR sensors act as proximity sensors, and they are commonly used in obstacle detection systems (such as in robots).

1.4 HC-SR04 Ultrasonic Sensor

The HC-SR04 is a type of ultrasonic sensor which uses sonar to find out the distance of the object from the sensor. It provides an outstanding range of non-contact detection with high accuracy & stable readings. It includes two modules like ultrasonic transmitter & receiver. This sensor is used in a variety of applications like measurement of direction and speed, burglar alarms, medical, sonar, humidifiers, wireless charging, non-destructive testing, and

ultrasonography. The HC-SR04 Ultrasonic sensor comes with four pins namely the V_{cc} pin, Trigger pin, Echo pin, & Ground pin. This sensor is used to measure the accurate distance between the target and the sensor. This sensor mostly works on sound waves.

1.5 Continuous Servo Motor

Continuous-rotation servos are servos that do not have a limited travel angle, instead, they can rotate continuously. They can be thought of as a motor and gearbox with servo input controls. In such servos, the input pulse results in a rotational speed, and the typical 1.5 ms center value is the stop position. A smaller value should turn the servo clockwise and a higher one counterclockwise.

1.6 Standard Servo Motor

The Standard Servo is ideal for robotics and basic movement projects. It can hold any position over a 180-degree range. The Standard Servo key features are that it can hold any position between 0 and 180 degrees, it has 38 oz-in torque at 6 V and weighs only 1.55 oz (44 g)

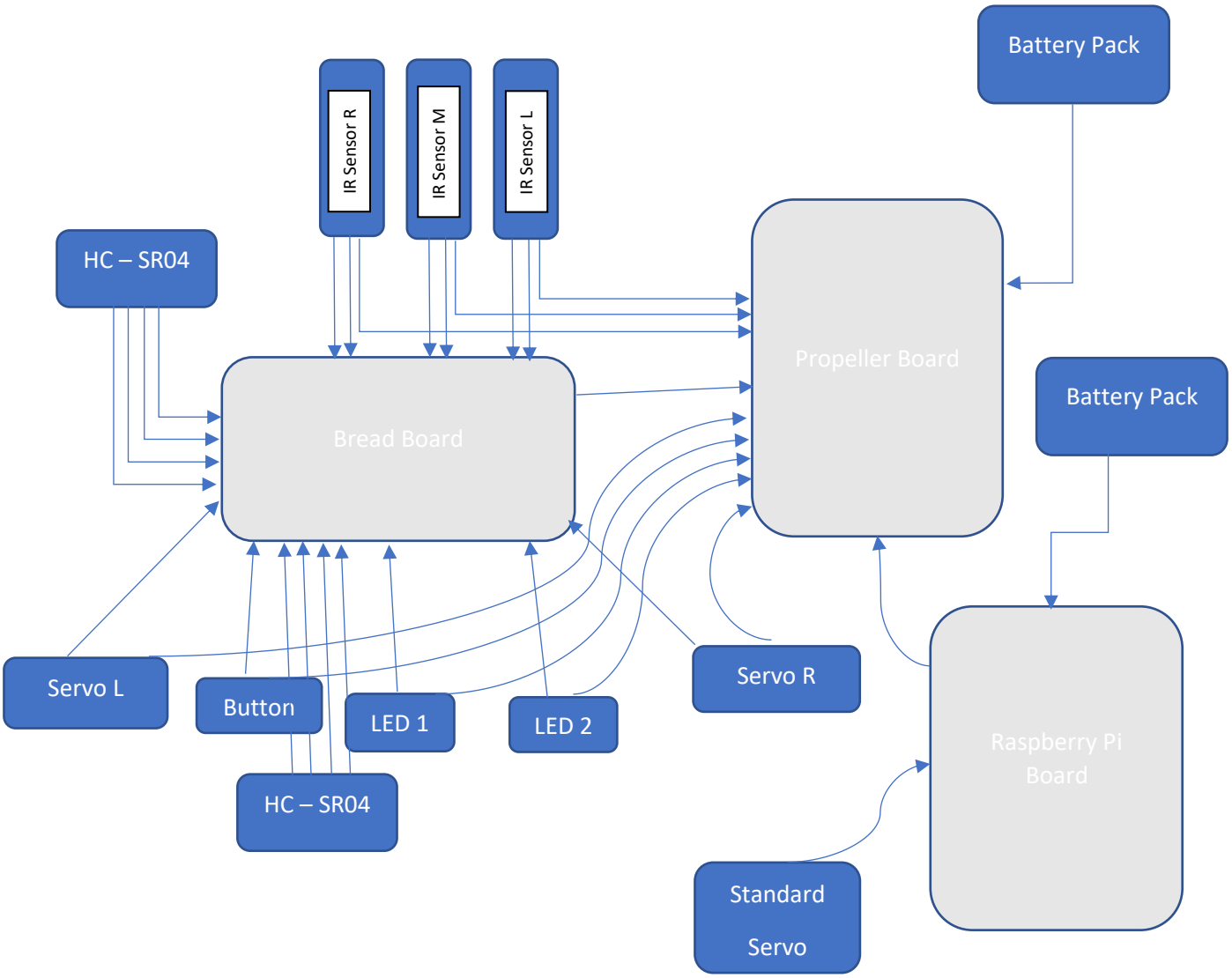
1.7 LED

A light-emitting diode (LED) is a semiconductor light source that emits light when current flows through it. Electrons in the semiconductor recombine with electron holes, releasing energy in the form of photons. The color of the light (corresponding to the energy of the photons) is determined by the energy required for electrons to cross the bandgap of the semiconductor. LEDs have many advantages, including lower power consumption, longer lifetime, improved physical robustness, smaller size, and faster switching.

1.8 Button

A push-button (also spelled pushbutton) or simply button is a simple switch mechanism to control some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, to be easily depressed or pushed. Buttons are most often biased switches, although many un-biased buttons (due to their physical nature) still require a spring to return to their un-pushed state.

2. Schematic Diagram of Connections

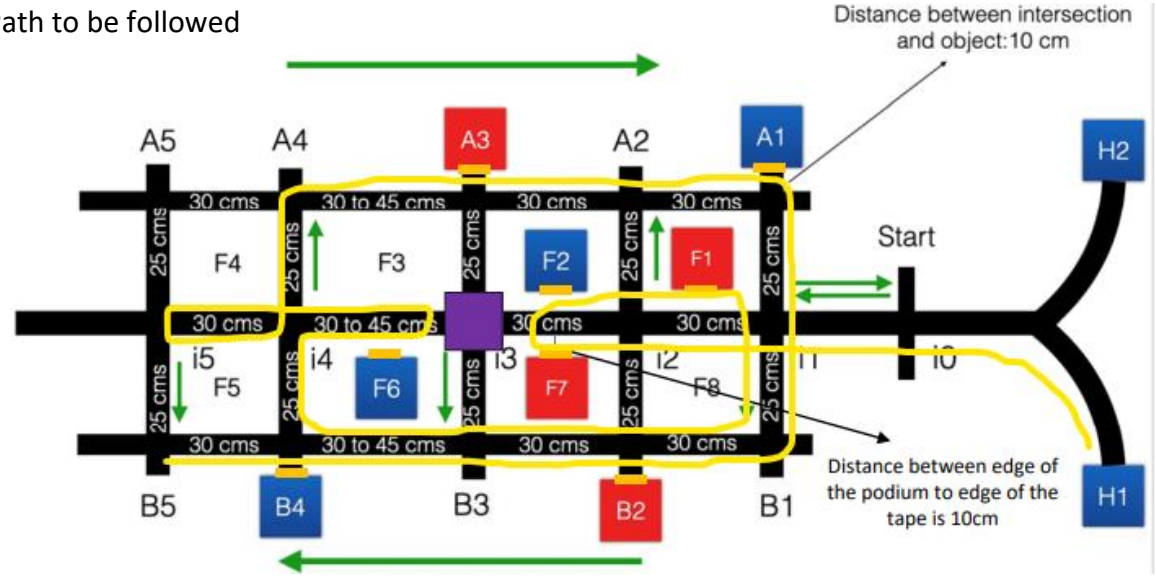


APPROACH

We have used the following approach for our bot and have implemented the following rules as well:

- The bot stops after detecting 8 objects
- The bot does not take an action while detecting the same Aruco tag the next time.
- We are using 6 X 6 Dictionionary type Aruco tags where any tags up to 10 are friendlies and above 10 are enemies

Path to be followed



PSEUDOCODE

- ## Propeller Part

```
Initialize all header files
Initialise stacks for cogs
Initialize all servos, LEDs, Ultrasonic
Initialize all variables
Initialize Function Prototypes
Main Program
int main()
{
  low(12);
  while(1)
  {
    When switch is pressed)
    {
      while(1)
      {
        if(rot==0 && obj<8)
        {
          Take IR Reading
          line_follow();
        }
        /*
        else if((rot==0 && stop_flag==1 && obj<8) || (rot==1 &&
stop_flag==1 && obj<8))
        {
          high(12);//signal to pi
          forward_junc(50);
          servo_stop();
```

```
            low(12);//stopping signal
            pause(3000);//stopping the bot for 3 seconds
          }*/
          else if(rot==1 && obj<8)//U-turn procedure
          {
           cd = 1;
           rot = 0;
           uturn(1100);
          }
          else
          {
            For loop
            {
              BLINK 2 LEDs
            }
          }
        }
      }
    }
}

//Functions
void forward()
{
  Move forward
}
void forward_junc(int val)
{
  Move forwrad
}
void back()
{
  Move back
}
void back_junc(int a)
{
  Move back
}
void left(int a)
{
  Left turn
}
void right(int a)
{
  Right turn
}
void left_turn()
{
  Left turn
}
void right_turn()
{
  Right turn
}
void uturn(int val)
{
  U turn
}
```

```
void line_follow()
{
  if(Only MIDDLE IR is black)
  {
    forward();
  }
  else if(Only RIGHT IR is black)
  {
    right();
  }
  else if(Only RIGHT & MIDDLE are black)
  {
    right();
  }
  else if(Only LEFT & MIDDLE are black)
  {
    left();
  }
  else if(Only LEFT is black)
  {
    left();
  }
  else if(all are WHITE)
  {
    back();
  }
  else if(all are BLACK)
  {
    junction_maneuver();
  }
}
void just_line_follow()
{
  if(Only MIDDLE IR is black)
  {
    forward();
  }
  else if(Only RIGHT IR is black)
  {
    right();
  }
  else if(Only RIGHT & MIDDLE are black)
  {
    right();
  }
  else if(Only LEFT & MIDDLE are black)
  {
    left();
  }
  else if(Only LEFT is black)
  {
    left();
  }
  else if(all are WHITE)
  {
    back();
  }
```

```
  else if(all are BLACK)
  {
    if(sflag==0)
    {
      j_count=j_count+1;
      while(all are BLACK)
          {
            forward();
          }
    }
  }
}
void line_follow_2()
{
  if(Only MIDDLE IR is black)
  {
    forward();
  }
  else if(Only RIGHT IR is black)
  {
    right();
  }
  else if(Only RIGHT & MIDDLE are black)
  {
    right();
  }
  else if(Only LEFT & MIDDLE are black)
  {
    left();
  }
  else if(Only LEFT is black)
  {
    left();
  }
  else if(all are WHITE)
  {
    back();
  }
}
void led1_glow(void *par1)
{
  Glow LED
}
void led2_glow(void *par2)
{
  Glow LED
}

void check_obstacle(void *par3)
{
 while(rot==0)
  {
    if(cmDist1<=5)
    {
      rot = 1;
      if(junc<6)//To check between i5 & i3
      {
        turn_flag = 1;
```

```
      }
    }
  }
}
void object_detect(void *par4)
{
  while(obj<8)
  {
    if(cmDist2<15)
    {
      obj=obj+1;
    }
  }
}
void junction_maneuver()
{
  if(cd==0)
  {
    junc = junc+1;
    if(junc<3)
    {
      while(all IR's BLACK)
        {
          forward();
        }
    }
    else
    {
      if(junc==3)
      {
      while(all are BLACK)
        {
          forward();
        }
    }
  }
  else if(cd==1)
  {
    print("coming back \n");
    if(junc==3)
      {
        cd = 2;
        right_turn();
      }
    else
    {
      junc=junc-1;
      while(all are BLACK)
       {
        forward();
       }
    }
  }
  else
  {
    junc = junc+1;
   if(junc==4)
    {
```

```
  right_turn();
}
else if(junc==7)
{
  right_turn();
}
else if(junc==8)
{
  if(turn_flag==1)
  {
   right_turn();
   Check for obstacle
   while(detect==0)
   {
     just_line_follow();
   }
   uturn(1100);
   if(j_count==1)
   {
     while(j_count<4)
     {
       just_line_follow();
     }
   }
   else
   {
     while(j_count<2)
     {
       just_line_follow();
     }
   }
   uturn();
   //left();
   sflag = 1;
   while(Not all 3 IR black)
   {
     line_follow_2();
   }
   left_turn();
  }
  else
  {
   while(all 3 IR are black)
    {
      forward();
    }
  }
}
else if(junc==9)
{
  right_turn();
}
else if(junc==12)
{
  right_turn();
  if(obj==7)
  {
    Stop object detection");
```

```
      }
    }
    else if(junc==14)
    {
      right_turn();
    }
    else if(junc==18)
    {
      servo_stop();
      if(obj<8)
      {
        obj=8;
      }
    }
    else//TO cross normal junction
    {
      while(all are BLACK)
        {
          forward();
        }
    }
  }
}

void check_obs()
{
  while(detect==0)
  {
    if(local_Dist<=7)
    {
      detect = 1;
    }
  }
}
```

COG 0 – Main Function

COG 1 – Red LED

COG 2 – Green LED

COG 3 – Obstacle Detection Ultrasonic

COG 4 – Object Detection Ultrasonic

- Raspberry Pi Part

Import the OpenCV library for computer vision

Initialise Servo pin

```
Load the dictionary that was used to generate the markers.
Initialize the detector parameters using default values
Initialize the camera and grab a reference to the raw camera capture
Set camera resolution and parameters
Initialse servo.min()
Define servo_hit():
Create mrks_list
while True:#Program keeps running forever
   try:
      # loop that runs the program forever
       Get input from propeller
      if getting GPIO input
         Run Marker Detection)
         Creates an "img" var that takes in a camera frame
         Convert to grayscale
         Detect aruco tags within the frame
          Draw box around aruco marker within camera frame
         If a tag is found...
         if markerIds is not None:
            For every tag in the array of detected tags..
            if detected_mkrs < 11:
               do nothing
            else:
               if previously detected marker is again detected
                  do nothing
               else:
                  servo_hit()
                  append to mrks_list
                  camera.capture("img_recog2.jpg")
```

markerIds[0] = 0

else:

print("Waiting for input")

When everything done, release the capture

CODE

- Propeller Part

```
#include "simpletools.h"
#include "servo.h"
#include "ping.h"

//stacks for cogs
unsigned int stack1[40+25];
unsigned int stack2[45+25];
unsigned int stack3[45+25];
unsigned int stack4[45+25];
unsigned int stack5[45+25];

static volatile int led1,led2,ping1,ping2;

//constants
#define servo1 13 //Left
#define servo2 15 //Right
#define led1 1 //Red
#define led2 2 //Green
#define ping1 6//Center ultrasonic
#define ping2 4//Left Ultrasonic

int junc = 0;
int cmDist1;
int cmDist2;
int local_Dist;
int cd = 0;
int rot = 0;
int cog1,cog2,cog3,cog4,cog5;
int ir1, ir2, ir3;
int j_count = 0;
int turn_flag = 0;
int detect = 0;
int sflag = 0;
//int stop_flag = 0;
```

```
int obj = 0;

//Function Prototypes
void line_follow();
void forward();
void forward_junc(int a);
void back_junc(int a);
void back();
void left(int a);
void right(int a);
void junction_maneuver();
void led1_glow(void *par1);
void led2_glow(void *par2);
void check_obstacle(void *par3);
void left_turn();
void right_turn();
void uturn(int val);
void object_detect(void *par4);
void just_line_follow();
void check_obs(void *par5);
void line_follow_2();

//Main Program
int main()
{
  low(12);
  while(1)
  {
    int c = input(0);
    pause(50);
    if(c==1)
    {
      while(1)
      {
        if(rot==0 && obj<8)
        {
          ir1 = input(10);//Right
          ir2 = input(5);//Middle
          ir3 = input(11);//Left
          line_follow();
         }
         /*
         else if((rot==0 && stop_flag==1 && obj<8) || (rot==1 &&
stop_flag==1 && obj<8))
         {
           print("Stop_flag==1 \n");
           high(12);//signal to pi
           //pause(500);
           forward_junc(50);
           servo_stop();
           low(12);//stopping signal
           pause(3000);//stopping the bot for 3 seconds
           stop_flag=0;
         }*/
         else if(rot==1 && obj<8)//U-turn procedure
         {
          print("Distance detected just before taking a uturn:
%d\n",cmDist1);
```

```
          print("Rotation Var pos1: %d\n",rot);
          back_junc(100);
          cd = 1;
          rot = 0;
          pause(100);
          print("Rotating...... \n");
          uturn(1100);
          back_junc(250);
          print("Rotation Var pos2: %d\n",rot);
         }
         else
         {
           print("Done!!");
           forward_junc(300);
           pause(50);
           servo_stop();
           //cogstop(cog3);
           for(int i=0;i<5;i++)
           {
             high(led1);
             high(led2);
             pause(500);
             low(led1);
             low(led2);
             pause(500);
           }
           pause(10000);
         }
       }
     }
   }
}

//Functions
void forward()
{
  servo_speed(servo1, 20);
  servo_speed(servo2, -25);
}
void forward_junc(int val)
{
  pause(100);
  servo_set(servo1,1590);
  servo_set(servo2,1490);
  pause(val);
}
void back()
{
  servo_speed(servo1,-30);
  servo_speed(servo2,30);
}
void back_junc(int a)
{
  pause(100);
  servo_set(servo1,1400);
  servo_set(servo2,1590);
  pause(a);
}
```

```
void left(int a)
{
  servo_speed(servo1,a);
  servo_speed(servo2,a+5);
  pause(100);
}
void right(int a)
{
  servo_speed(servo1,-a);
  servo_speed(servo2,-a-5);
  pause(100);
}
void left_turn()
{
  pause(100);
  servo_set(servo1,1400);
  servo_set(servo2,1400);
  pause(600);
}
void right_turn()
{
  pause(100);
  servo_set(servo1,1600);
  servo_set(servo2,1600);
  pause(300);
}
void uturn(int val)
{
  pause(100);
  servo_set(servo1,1400);
  servo_set(servo2,1400);
  pause(val);
}

void line_follow()
{
  if(ir1 == 0 && ir2 == 1 && ir3 == 0)
  {
    forward();
  }
  else if(ir1==1 && ir2==0 && ir3==0)
  {
    right(10);
  }
  else if(ir1 == 1 && ir2 == 1 && ir3 == 0)
  {
    right(20);
  }
  else if(ir1 == 0 && ir2 == 1 && ir3 == 1)
  {
    left(10);
  }
  else if(ir1 == 0 && ir2 == 0 && ir3 == 1)
  {
    left(20);
  }
  else if(ir1 == 0 && ir2 == 0 && ir3 == 0)
  {
```

```
    back();
  }
  else if(ir1 == 1 && ir2 == 1 && ir3 == 1)
  {
    junction_maneuver();
  }
}
void just_line_follow()
{
  if(ir1 == 0 && ir2 == 1 && ir3 == 0)
  {
    forward();
  }
  else if(ir1==1 && ir2==0 && ir3==0)
  {
    right(20);
  }
  else if(ir1 == 1 && ir2 == 1 && ir3 == 0)
  {
    right(20);
  }
  else if(ir1 == 0 && ir2 == 1 && ir3 == 1)
  {
    left(20);
  }
  else if(ir1 == 0 && ir2 == 0 && ir3 == 1)
  {
    left(20);
  }
  else if(ir1 == 0 && ir2 == 0 && ir3 == 0)
  {
    back();
  }
  else if(ir1 == 1 && ir2 == 1 && ir3 == 1)
  {
    if(sflag==0)
    {
      j_count=j_count+1;
      print("Junction number = %d \n",junc);
      print("J count = %d \n",j_count);
      cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      while(ir1==1 && ir2==1 && ir3==1)
          {
            forward();
            ir1 = input(10);
            ir2 = input(5);
            ir3 = input(11);
          }
    }
  }
}
void line_follow_2()
{
  if(ir1 == 0 && ir2 == 1 && ir3 == 0)
  {
    forward();
  }
  else if(ir1==1 && ir2==0 && ir3==0)
```

```
  {
    right(20);
  }
  else if(ir1 == 1 && ir2 == 1 && ir3 == 0)
  {
    right(20);
  }
  else if(ir1 == 0 && ir2 == 1 && ir3 == 1)
  {
    left(20);
  }
  else if(ir1 == 0 && ir2 == 0 && ir3 == 1)
  {
    left(20);
  }
  else if(ir1 == 0 && ir2 == 0 && ir3 == 0)
  {
    back();
  }
}
void led1_glow(void *par1)
{
  high(led1);
  pause(500);
  low(led1);
  pause(500);
  cogstop(cog1);
}
void led2_glow(void *par2)
{
  high(led2);
  pause(500);
  low(led2);
  pause(500);
  cogstop(cog2);
}

void check_obstacle(void *par3)
{
 while(rot==0)
  {
    cmDist1 = ping_cm(ping1);
    pause(200);
    if(cmDist1<=5)
    {
      rot = 1;
      //print("Object Detected on periphery");
      if(junc<6)//To check between i5 & i3
      {
        turn_flag = 1;
      }
      cog2 = cogstart((void*)led2_glow,NULL,stack2,sizeof(stack2));
    }
  }
  pause(200);
  cogstop(cog4);
}
void object_detect(void *par4)
```

```
{
  while(obj<8)
  {
    cmDist2 = ping_cm(ping2);
    pause(100);
    if(cmDist2<15)
    {
      cog2 = cogstart((void*)led2_glow,NULL,stack2,sizeof(stack2));
      pause(1000);
      //stop_flag = 1;
      obj=obj+1;
      pause(2000);//Add 6 seconds for stop flag
    }
  }
  cogstop(cog3);
}
void junction_maneuver()
{
  if(cd==0)
  {
    junc = junc+1;
    print("going straight \n");
    if(junc<3)
    {
      //cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      print("Junction number = %d \n",junc);
      while(ir1==1 && ir2==1 && ir3==1)
        {
          forward();
          ir1 = input(10);
          ir2 = input(5);
          ir3 = input(11);
        }
    }
    else
    {
      cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      //pause(50);
      if(junc==3)
      {
        print("Starting the obstacle detection!\n");
        cog4 =
cogstart((void*)check_obstacle,NULL,stack4,sizeof(stack4));
        print("Starting object detection also!! \n");
        cog3 =
cogstart((void*)object_detect,NULL,stack3,sizeof(stack3));
      }
      print("Junction number = %d \n",junc);
      print("Checking for obstacle \n");
      print("Distance detected: %d\n",cmDist1);
      while(ir1==1 && ir2==1 && ir3==1)
        {
          forward();
          ir1 = input(10);
          ir2 = input(5);
          ir3 = input(11);
        }
    }
```

```
  }
  else if(cd==1)
  {
    print("coming back \n");
    if(junc==3)
      {
        cd = 2;
        cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
        print("Junction number = %d \n",junc);
        forward_junc(200);
        pause(100);
        right_turn();
        print("At the start junction, Turning right");
      }
    else
    {
      junc=junc-1;
      print("Returning after detecting an obstruction \n");
      print("Junction number = %d \n",junc);
      cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      while(ir1==1 && ir2==1 && ir3==1)
       {
        forward();
        ir1 = input(10);
        ir2 = input(5);
        ir3 = input(11);
       }
    }
  }
  else
  {
    junc = junc+1;
    //print("approaching junction 4 \n");
    print("Junction number = %d \n",junc);
   if(junc==4)
    {
      cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      pause(50);
      //cog3 =
cogstart((void*)object_detect,NULL,stack3,sizeof(stack3));
      forward_junc(200);
      pause(100);
      right_turn();
      //print("Junction number = %d \n",junc);
    }
    else if(junc==7)
    {
      cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      //pause(500);
      servo_stop();
      pause(300);
      forward_junc(300);
      pause(100);
      right_turn();
      //print("Junction number = %d \n",junc);
    }
    else if(junc==8)
    {
```

```
print("At junction 8 \n");
cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
if(turn_flag==1)
{
 print("Taking right at junction 8 \n");
 forward_junc(400);
 pause(100);
 right_turn();
 //pause(50);
 //int detect = 0;
 cog5 = cogstart((void*)check_obs,NULL,stack5,sizeof(stack5));
 print("Checking for obstacle... \n");
 while(detect==0)
 {
   ir1 = input(10);
   ir2 = input(5);
   ir3 = input(11);
   just_line_follow();
 }
 print("Uturn taken, Going to i5... \n");
 uturn(1100);
 back_junc(100);
 //pause(4000);
 print("Returning to junction 8 \n");
 if(j_count==1)
 {
   print("Obstacle at i2\n");
   while(j_count<4)
   {
     ir1 = input(10);
     ir2 = input(5);
     ir3 = input(11);
     just_line_follow();
   }
 }
 else
 {
   print("Obstacle at i3\n");
   while(j_count<2)
   {
     ir1 = input(10);
     ir2 = input(5);
     ir3 = input(11);
     just_line_follow();
   }
 }
 print("i5 reached..\n");
 forward_junc(200);
 uturn(1200);
 back_junc(100);
 //left();
 sflag = 1;
 while(!(ir1==1 && ir2==1 && ir3==1))
 {
   print("Searching for i4\n");
   ir1 = input(10);
   ir2 = input(5);
   ir3 = input(11);
```

```
     line_follow_2();
    }
    print("Turning Left\n");
    forward_junc(400);
    pause(100);
    left_turn();
   }
   else
   {
    print("Normal Junction");
    //cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
    while(ir1==1 && ir2==1 && ir3==1)
     {
       forward();
       ir1 = input(10);
       ir2 = input(5);
       ir3 = input(11);
     }
   }
}
else if(junc==9)
{
  cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
  forward_junc(200);
  pause(100);
  right_turn();
  //print("Junction number = %d \n",junc);
}
else if(junc==12)
{
  cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
  forward_junc(200);
  pause(100);
  right_turn();
  if(obj==7)
  {
    print("Stopping object detection");
    cogstop(cog3);
  }
  //print("Junction number = %d \n",junc);
}
else if(junc==14)
{
  cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
  forward_junc(200);
  pause(100);
  right_turn();
  //print("Junction number = %d \n",junc);
}
else if(junc==18)
{
  cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
  print("Starting object detection again!! \n");
  cog3 = cogstart((void*)object_detect,NULL,stack3,sizeof(stack3));
  forward_junc(200);
  //pause(1000);
  //cogstop(cog3);
  servo_stop();
```

```
      if(obj<8)
      {
        obj=8;
      }
      //pause(20000);
      //right_turn();
      //print("Junction number = %d \n",junc);
    }
    else//TO cross normal junction
    {
      cog1 = cogstart((void*)led1_glow,NULL,stack1,sizeof(stack1));
      while(ir1==1 && ir2==1 && ir3==1)
        {
          forward();
          ir1 = input(10);
          ir2 = input(5);
          ir3 = input(11);
        }
    }
  }
}

void check_obs(void *par5)
{
  while(detect==0)
  {
    local_Dist = ping_cm(ping1);
    pause(200);
    if(local_Dist<=7)
    {
      detect = 1;
      //cog2 = cogstart((void*)led2_glow,NULL,stack2,sizeof(stack2));
      pause(100);
    }
  }
  cogstop(cog5);
}
```

- Raspberryry Pi Code

```
# import the OpenCV library for computer vision
import cv2
import RPi.GPIO as GPIO
from gpiozero import Servo
from time import sleep
from picamera.array import PiRGBArray
from picamera import PiCamera

servo = Servo(23,minimum_pulse_width=1.05,maximum_pulse_width=2.95)

comm_in = 26
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(comm_in,GPIO.IN)

# Load the dictionary that was used to generate the markers.
# There's different aruco marker dictionaries, this code uses 6x6
dictionary = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)

# Initialize the detector parameters using default values
parameters = cv2.aruco.DetectorParameters_create()

parameters.minMarkerPerimeterRate = 0.5


# initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
#camera.framerate = 32
#rawCapture = PiRGBArray(camera, size=(640, 480))
servo.min()
def servo_hit():
    servo.max()
    sleep(0.5)
    servo.min()
    sleep(0.5)

mrks_list = []
sleep(1)
while True:#Program keeps running forever
    try:
        # loop that runs the program forever
        #comm_status = GPIO.input(comm_in)
        if GPIO.input(comm_in):

            print("Running Marker Detection!!")
            # creates an "img" var that takes in a camera frame
            camera.capture("img.jpg")
            img = cv2.imread("img.jpg")

            # Convert to grayscale
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

            # detect aruco tags within the frame
            markerCorners, markerIds, rejectedCandidates = cv2.aruco.detectMar
kers(gray, dictionary, parameters=parameters)
            # draw box around aruco marker within camera frame
            #img = cv2.aruco.drawDetectedMarkers(img, markerCorners, markerIds
)
            # if a tag is found...
```

```
            if markerIds is not None:
                # for every tag in the array of detected tags..
                print(markerIds[0])
                detected_mkrs = int(markerIds[0])

                if detected_mkrs < 11:
                    print("Good guy!")
                    camera.capture("img_recog1.jpg")
                    markerIds = [None]
                    #run = int(input("Enter 1 to continue 0 to stop: "))
                else:
                    print("Bad Guys!!")
                    if (detected_mkrs in mrks_list):
                        markerIds[0] = 0
                        sleep(2)
                    else:
                        servo_hit()
                        mrks_list.append(detected_mkrs)
                        print(mrks_list)
                        camera.capture("img_recog2.jpg")
                        markerIds[0] = 0
                    #run = int(input("Enter 1 to continue 0 to stop: "))
                #GPIO.output(comm_out,1)
                sleep(3)
        else:
            print("Waiting for input")
            sleep(0.5)

    except KeyboardInterrupt:
        camera.close()
        cv2.destroyAllWindows()
        GPIO.cleanup()

# When everything done, release the capture
print("Program is Stopping!!")
```