ROB-GY 6103
ADVANCED MECHATRONICS

PROPELLER AND RASPBERRY PI BASED LINE FOLLOWER ROBOT WITH TASKS

Project Report submitted to Professor Kapila in partial fulfillment of the requirements of the course

TEAM - 8

Arunagiri Adhithian R - ar7404

Nithya Muralidaran - nm3671

Rose Solow - rs7138

May 2022

NYU TANDON SCHOOL OF ENGINEERING

TABLE OF CONTENTS

ABSTRACT:

This project deals with the development of a robot to scour the streets of the city to find the enemies among civilians. The system is constructed using a Propeller Activity Board and a Raspberry Pi 4. This project deals with using computer vision as well as communication between two different microcontrollers.

1. INTRODUCTION:

This Propeller and Raspberry Pi-based line follower robot is constructed using two Adafruit 2349 reflective IR sensors for path following, one HC-SR04 ultrasonic sensor for object detection, two Parallax continuous servo motors for wheel motion, a Parallax standard servo for a knock off mechanism, a USB endoscope camera for tag detection and LEDs for indicating recognition of tags. The Propeller handles path navigation, obstacle avoidance, and the knock-off mechanism. The Raspberry Pi handles tag detection and enemy/friendly indication. The Raspberry Pi detects aruco tags using the Python OpenCV library. Communication between the microcontrollers is implemented using a pin-to-pin connection. Whenever the Raspberry Pi detects an enemy tag a high signal is sent to a monitoring pin on the Propeller which then initiates a knock-off routine.

2. OBJECTIVES:

The below-given details are the given objectives to be fulfilled while creating this project for detecting enemies amongst civilians.

- The robot starts from either one of the 2 home positions given.
- The robot must follow the path created (black tape 1”).
- The robot must detect an object - traffic congestion and maneuver around it.
- The robot must detect tags on the objects on the side of the road and indicate whether the object is a friend or an enemy.
- The robot must knock off detected enemies.

3.1 HARDWARE CONSTRUCTION:

The chassis of the robot is from the Parallax Boe-bot. The robot uses continuous servo motors as wheels for movement through the path. The robot is constructed of multiple layers of horizontal platforms. A breadboard and a Propeller Activity Board are mounted on the lower level of the Boe-bat chassis and another breadboard and a Raspberry Pi are mounted on a 3D printed platform above them. A 3D printed frontispiece is attached to the platform to fix the two IR sensors so that the two sensors face downward with a 5 mm gap from the ground surface for detection of reflection from the path with approximately 4 cm of dividing space between them.

Three LEDs for event indication are mounted on the top breadboard and an ultrasonic sensor for object detection is mounted on the breadboard on the lower level of the robot. Connecting wires are used for communication between the sensors, actuators, and in between the microcontrollers. The Propeller board uses 4 AA batteries that provide a cumulative voltage of 6V while the Raspberry Pi draws 5V from a portable charger mounted on top of the robot. A standard servo motor is mounted on the left side of the bot with a 3D printed arm which is used to knock off the enemies. A USB endoscope camera is used to read the tags and a mounting is created to hold the cable for the camera on the backside of the robot.

.

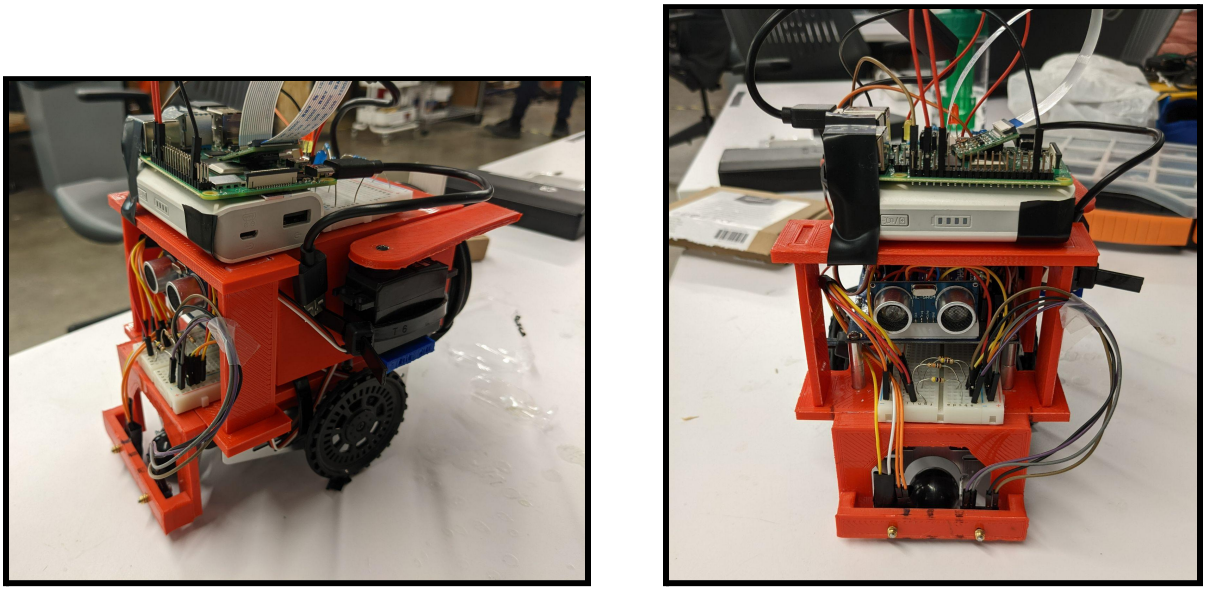


Fig 3.1 Autonomous path follower robot

3.2 MECHANICAL DESIGN

Using Autodesk Fusion 360, multiple parts were designed to create the framework for the robot. A base mounting board was 3D printed with four 7.5mm holes dedicated to the chassis standoffs to mount the Propeller board and lower breadboard. A top mounting board was 3D printed with three 3D printed supports to create an upper level for the robot to mount the Raspberry Pi and upper breadboard.

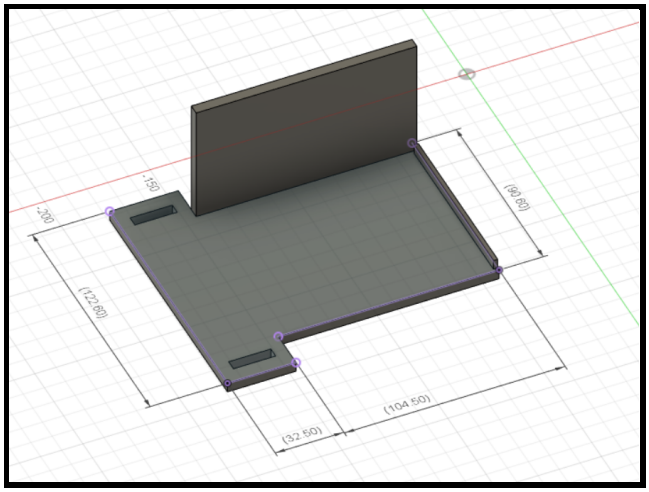


Fig 3.2 Top mounting board

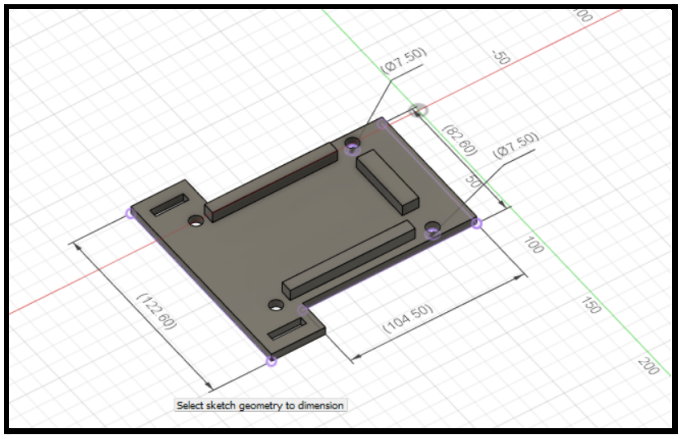
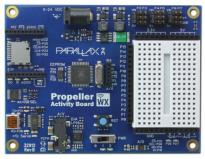
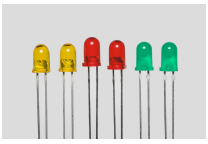


Fig. 3.3 Base mounting board

4. METHODOLOGY

4.1 Hardware List of Electronics

Count	Parts	Type/Function	Specification	Image
1x	Raspberry Pi 4	Single board computer	Model B	

1x	Propeller Activity Board	Project board for Propeller microcontroller	Development Board	
1x	Ultrasonic sensor	Detects the distance of an object using ultrasonic sound waves	HC-SR04	
2x	Reflective Infrared IR optical sensor	Detects reflected IR radiation emitted by IR emitter for following black line path	Adafruit 2349	
1x	USB Endoscope camera	Takes images continuously to detect tags	720p HD camera	
2x	Continuous servo motor	Provides bi-directional continuous 360-degree rotation for wheels	Bidirectional continuous rotation with 0 to 50 RPM	
1x	Standard servo motor	Provides 180-degree rotation for knock-off mechanism	180-degree rotation with 0 to 50 RPM	
3x	LED	Blinks to indicate intersection, friendlies, and enemies	1x blue LED 1x red LED 1x yellow LED	

11x	Resistors	To limit the current through the microcontroller and associated circuits	7x 220 OHM, 2x 470 OHM, 2x 10 kOHM	
2x	Capacitors	To measure the IR sensors using an RC circuit	2x 0.1 µF	

4.2 Final Circuit

The figure below shows the final functional circuit diagram of the system.

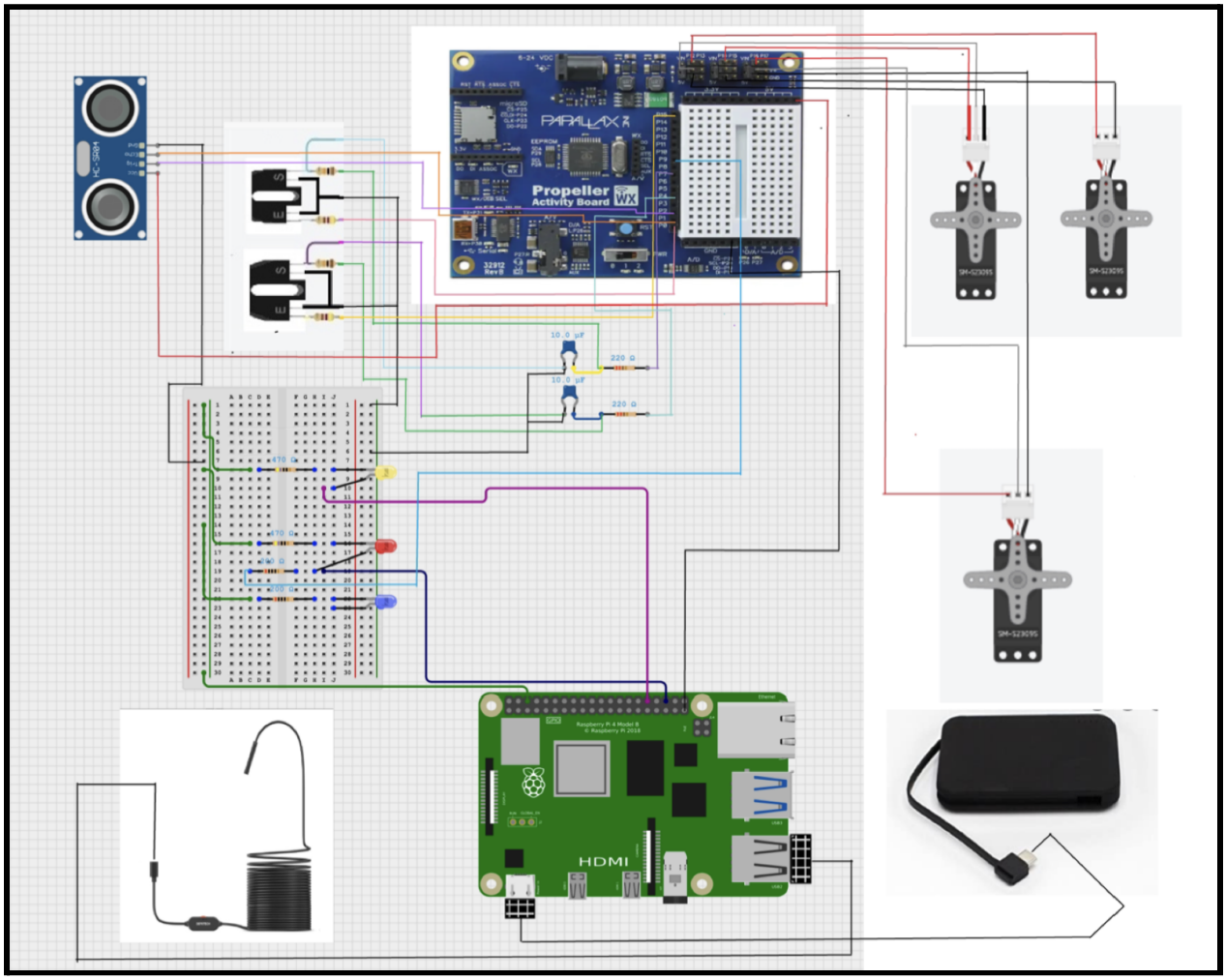


Fig. 4.1 Circuit diagram of the Robot

5. PSEUDOCODE

1. Turn on left and right IR emitters and read left and right IR sensors.

2. Follow line based on IR sensor measurements.

    2a. If the left sensor reads bright then the left wheel rotates otherwise the left wheel stops.

    2b. If the right sensor reads bright then the right wheel rotates otherwise the right wheel stops.

3. If the left sensor reads dark and the right sensor reads dark then at an intersection.

    3a. Move both wheels forward to clear intersection while blinking blue LED to indicate intersection detected.

4. On Raspberry Pi continuously reads frames from the camera and detects the aruco tags in each frame.

    4a. If a friendly aruco tag is found, turn on the yellow LED to indicate friendly detected.

    4b. If an enemy aruco tag is found, turn on the red LED to indicate enemy detected and send a signal to Propeller to run a routine to knock off the enemy.

5. If the Propellor reads enemy detections from the pin signal from Raspberry Pi then stop in front of the location and swing the attack arm to knock off the enemy.

6. Drive down the main road.

7. While driving down the main road search for a blocking object.

    7a. Send ultrasonic ping and read ultrasonic sensor echo.

    7b. If the measured distance is less than 6cm then the blocking object is detected.

8. If a blocking object is detected, turn around and return to intersection I1.

9. Drive down B road and turn right at the B4 intersection.

10. Drive down the main road until the blocking object is detected then turn around and return to intersection I4.

11. Drive down A road and turn right at the A1 intersection.

12. After fully navigating the map head toward location B5 by driving down B road (B5 is the last location to check as it is a dead zone after arriving since traffic rules are violated when trying to leave B5).

13. Stop at location B5.

6. CONCLUSION

The project presents a Propeller and Raspberry Pi-based automated path follower which detects intersections and objects without stopping. We have indicated every event by blinking the blue LED for intersection detections, blinking the red LED for enemy detections and yellow LED for friend detections.

We have achieved all the objectives we have mentioned above.

Click to view the Video of a Successful Demo

7. APPENDIX

7.1 Commented Propeller Code

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

//pins
int rightWheel = 14;
int leftWheel = 12;
int attackServo = 16;
int echo = 1;
int trigger = 2;
int leftLED = 15;
int rightLED = 0;
int leftSensor = 7;
int rightSensor = 4;
int led = 5;
int enemy = 9;

int leftLimit = 2300; //limit to determine whether bright or dark
int rightLimit = 1300; //limit to determine whether bright or dark
int leftVal = 800; //left sensor reading
int rightVal = 800; //right sensor reading
long duration = 1; //duration of sound wave travel
int distance = 1; //distance measurement
int numIntersection = -2; //number of intersections detected
```

```
int numLeft = 0; //number of left turns
int numRight = 0; //number of right turns
int detectingObjects = 1; //detect blocking object only when going down main road
int leftSpeed = 50;
int rightSpeed = -12;
int flag = 0;

void measure_IR();
void follow_line();
void keep_following();
void measure_distance();
void check_friends();
void destroy_enemies();
void turn_right();
void turn_left();
void turn_around();

int main()
{
  //servo_speed(leftWheel,80); //fast
  //servo_speed(rightWheel,-40); //fast

  //servo_speed(leftWheel,35); //slow
  //servo_speed(rightWheel,-10); //slow

  //servo_speed(leftWheel,22); //stop
  //servo_speed(rightWheel,13); //stop

  /*
  while(1){
    follow_line();
    if (rightVal > rightLimit && leftVal > leftLimit) {
       //pause(100);
       //numIntersection = numIntersection + 1;
       servo_speed(leftWheel,70);
       servo_speed(rightWheel,-30);
       high(led);
       pause(400);
       low(led);
     }
  }
  */

  //preset attack arm
  servo_angle(attackServo,1700);
  pause(2000);
```

```
//move down main road while detecting front objects to find blocked intersection
while (detectingObjects == 1){
  //if rpi sending enemy signal then knock off detected enemy
  if (input(enemy) == 1){
    destroy_enemies();
  }
  else {
    measure_distance();
    follow_line();
    //if intersection detected
    if (rightVal > rightLimit && leftVal > leftLimit) {
      //pause(100);
      numIntersection = numIntersection + 1;
      //move off intersection while blinking intersection indicator led
      servo_speed(leftWheel,70);
      servo_speed(rightWheel,-30);
      high(led);
      pause(400);
      low(led);

      //curved line "intersection"
      if (numIntersection == -1){
        if (numRight > numLeft) {
          servo_speed(leftWheel,22); //stop
        }
        if (numRight < numLeft) {
          servo_speed(rightWheel,13); //stop
        }
        pause(300);
      }
    }
  }

  if (distance < 6){
    //blocking object detected
    detectingObjects = 0;
    pause(300);
    turn_around();
    pause(300);
  }
}

//blocking object at i2
//run navigation routine
if (numIntersection == 1){
```

```
while (numIntersection <= 22){
 if (input(enemy) == 1){
  destroy_enemies();
 }
 else   {
  follow_line();
  if (rightVal > rightLimit && leftVal > leftLimit) {
   //pause(100);
   numIntersection = numIntersection + 1;
   servo_speed(leftWheel,70); //fast
   servo_speed(rightWheel,-30); //fast
   high(led);
   pause(400);
   low(led);

   //i1
   if (numIntersection == 2){
    keep_following();
    turn_right();
   }
   //b1
   if (numIntersection == 3){
    keep_following();
    turn_right();
   }
   //b4
   if (numIntersection == 6) {
    keep_following();
    if (input(enemy) == 1){
     destroy_enemies();
    }
    keep_following();
    if (input(enemy) == 1){
     destroy_enemies();
    }
    turn_right();
   }
   //i4
   if (numIntersection == 7) {
    keep_following();
    turn_right();
   }
   //i3
   if (numIntersection == 8) {
    measure_distance();
    while (distance > 4){
```

```
      follow_line();
      measure_distance();
     }
     pause(100);
     turn_around();
     pause(100);
    }
    //i5
    if (numIntersection == 11) {
     turn_around();
    }
    //i4
    if (numIntersection == 12) {
     keep_following();
     turn_left();
    }
    //a4
    if (numIntersection == 13) {
     keep_following();
     turn_right();
    }
    //a1
    if (numIntersection == 16) {
     keep_following();
     turn_right();
    }
    //b1
    if (numIntersection == 18) {
     keep_following();
     turn_right();
    }
    //b5
    if (numIntersection == 22) {
     keep_following();
     if (input(enemy) == 1){
      destroy_enemies();
     }
     keep_following();
     while(1){
      servo_speed(leftWheel,22); //stop
      servo_speed(rightWheel,13); //stop
     }
    }
   }
  }
}
```

```
}

//blocking object at i3
//run navigation routine
if (numIntersection == 2){
  while (numIntersection <= 22){
    if (input(enemy) == 1){
      destroy_enemies();
    }
    else   {
      follow_line();
      if (rightVal > rightLimit && leftVal > leftLimit) {
        //pause(100);
        numIntersection = numIntersection + 1;
        servo_speed(leftWheel,70); //fast
        servo_speed(rightWheel,-30); //fast
        high(led);
        pause(400);
        low(led);

        //i1
        if (numIntersection == 4){
          keep_following();
          turn_right();
        }
        //b1
        if (numIntersection == 5){
          keep_following();
          turn_right();
        }
        //b4
        if (numIntersection == 8) {
          keep_following();
          if (input(enemy) == 1){
            destroy_enemies();
          }
          keep_following();
          if (input(enemy) == 1){
            destroy_enemies();
          }
          turn_right();
        }
        //i4
        if (numIntersection == 9) {
          keep_following();
          turn_right();
```

```
  measure_distance();
  while (distance > 4){
    follow_line();
    measure_distance();
  }
  pause(100);
  turn_around();
  pause(100);
}
//i5
if (numIntersection == 11) {
  pause(100);
  turn_around();
  pause(100);
}
//i4
if (numIntersection == 12) {
  keep_following();
  turn_left();
}
//a4
if (numIntersection == 13) {
  keep_following();
  turn_right();
}
//a1
if (numIntersection == 16) {
  keep_following();
  turn_right();
}
//b1
if (numIntersection == 18) {
  keep_following();
  turn_right();
}
//b5
if (numIntersection == 22) {
  keep_following();
  if (input(enemy) == 1){
    destroy_enemies();
  }
  keep_following();
  while(1){
    servo_speed(leftWheel,22); //stop
    servo_speed(rightWheel,13); //stop
  }
```

```
      }
     }
    }
  }
}

//blocking object at i5
//run navigation routine
if (numIntersection == 4){
  while (numIntersection <= 23){
    if (input(enemy) == 1){
      destroy_enemies();
    }
    else   {
      follow_line();
      if (rightVal > rightLimit && leftVal > leftLimit) {
        //pause(100);
        numIntersection = numIntersection + 1;
        servo_speed(leftWheel,70); //fast
        servo_speed(rightWheel,-30); //fast
        high(led);
        pause(400);
        low(led);

        //i1
        if (numIntersection == 8){
          keep_following();
          turn_right();
        }
        //b1
        if (numIntersection == 9){
          keep_following();
          turn_right();
        }
        //b4
        if (numIntersection == 12) {
          keep_following();
          if (input(enemy) == 1){
            destroy_enemies();
          }
          keep_following();
          if (input(enemy) == 1){
            destroy_enemies();
          }
          turn_right();
        }
```

```
        //a4
        if (numIntersection == 14) {
          keep_following();
          turn_right();
        }
        //a1
        if (numIntersection == 17) {
          keep_following();
          turn_right();
        }
        //b1
        if (numIntersection == 19) {
          keep_following();
          turn_right();
        }
        //b5
        if (numIntersection == 23) {
          keep_following();
          if (input(enemy) == 1){
            destroy_enemies();
          }
          keep_following();
          while(1){
            servo_speed(leftWheel,22); //stop
            servo_speed(rightWheel,13); //stop
          }
        }
      }
    }
  }
}

void measure_IR(){
  //turn on IR LEDs
  high(leftLED); high(rightLED);

  //charge capacitors
  high(rightSensor); high(leftSensor);
  pause(100);
  //read in reflectance value based on rc_time dissipation
  leftVal = rc_time(leftSensor,1);
  rightVal = rc_time(rightSensor,1);

  print("right sensor: %d \n",rightVal);
  print("left sensor: %d \n",leftVal);
```

```
  //turn off IR LEDs
  low(leftLED); low(rightLED);
  pause(100);
}

void follow_line(){
  measure_IR();

  //turn wheels based on IR readings
  if (leftVal <= leftLimit) {
    servo_speed(leftWheel,leftSpeed);
  }
  else{
    servo_speed(leftWheel,22);
    numRight = numRight + 1;
  }

  if (rightVal <= rightLimit) {
    servo_speed(rightWheel,rightSpeed);
  }
  else{
    servo_speed(rightWheel,13);
    numLeft = numLeft + 1;
  }
}

void keep_following(){
  for (int i=1;i<=3;i++){
    follow_line();
  }
}

void measure_distance(){
  //clear trigger then send ultrasonic chirp
  low(trigger);
  pause(.2);
  high(trigger);
  pause(.1);
  low(trigger);

  //read in echo and calculate measured distance
  duration = pulse_in(echo, 1);
  distance = duration * 0.034 / 2;
  print("distance: %d \n",distance);
}
```

```
void destroy_enemies(){
  //knock off enemy
  servo_speed(leftWheel,22); //stop
  servo_speed(rightWheel,13); //stop
  pause(2000);
  //swing attackServo arm
  servo_angle(attackServo,10);
  pause(1000);
  servo_angle(attackServo,1700);
  pause(1000);
}

void turn_right(){
  servo_speed(leftWheel,70);
  servo_speed(rightWheel,70);
  pause(850);
  servo_speed(leftWheel,22); //stop
  servo_speed(rightWheel,13); //stop
}

void turn_left(){
  servo_speed(leftWheel,-40);
  servo_speed(rightWheel,-40);
  pause(800);
  servo_speed(leftWheel,22); //stop
  servo_speed(rightWheel,13); //stop
}

void turn_around(){
  servo_speed(leftWheel,-40);
  servo_speed(rightWheel,-40);
  pause(1600);
  servo_speed(leftWheel,22); //stop
  servo_speed(rightWheel,13); //stop
}
```

7.2 Commented Raspberry Pi Code

```
import cv2
import picamera
import time
import RPi.GPIO as GPIO

enemy_pin = 16
friend_pin = 12
```

```
#capture set number of frames on set camera
def run(frames=100,cam_num=0):
        #set up pins
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(enemy_pin,GPIO.OUT)
        GPIO.setup(friend_pin,GPIO.OUT)
        GPIO.output(enemy_pin,0)
        GPIO.output(friend_pin,0)

        #set up aruco parameters
        arucoDict = cv2.aruco.Dictionary_get(cv2.aruco.DICT_6X6_250)
        arucoParams = cv2.aruco.DetectorParameters_create()

        #create camera object
        cam = cv2.VideoCapture(cam_num)

        tags = []
        for i in range(frames):
                #capture frame, convert to gray, show frame
                ret, image = cam.read()
                image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
                cv2.imshow('Imagetest',image)
                #wait for 50ms or keystroke
                k = cv2.waitKey(20)
                #if keystroke closes window then break loop
                if k != -1 :
                        cam.release()
                        cv2.destroyAllWindows()
                        print(tags)
                        break
                #detect aruco tags in frame
                (corners,ids,rejected) = cv2.aruco.detectMarkers(image,arucoDict,
                        parameters=arucoParams)
                #if a tag is detected then check if enemy or friend
                if ids is not None:
                        vec = corners[0][0][0] - corners[0][0][1]
                        d = vec[0]**2 + vec[1]**2
                        if d > 10000:
                                if ids[0][0] >= 10 and ids[0][0] not in tags:
                                        tags.append(ids[0][0])
                                        print('enemy')
                                        print(ids[0][0])
                                        #enemy indicator led on
                                        #also sends pin signal to propeller to
                                        #execute knock off enemies routine on propeller
```

```
                                        GPIO.output(enemy_pin,1)
                                        time.sleep(1)
                                        GPIO.output(enemy_pin,0)
                                if ids[0][0] < 10 and ids[0][0] not in tags:
                                        tags.append(ids[0][0])
                                        print('friendly')
                                        #friend indicator led on
                                        GPIO.output(friend_pin,1)
                                        time.sleep(1)
                                        GPIO.output(friend_pin,0)
                        else:
                                print('too far')
                else:
                        print('no tags')
                        GPIO.output(friend_pin,0)
                        GPIO.output(enemy_pin,0)

        #clean up all created objects
        #cv2.imwrite('/home/pi/testimage.jpg', image)
        cv2.destroyAllWindows()
        cam.release()
        cv2.destroyAllWindows()
        GPIO.output(friend_pin,0)
        GPIO.output(enemy_pin,0)
        GPIO.cleanup()
        print(tags)

#check indicator leds work properly
def check_leds():
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(enemy_pin,GPIO.OUT)
        GPIO.setup(friend_pin,GPIO.OUT)

        GPIO.output(enemy_pin,1)
        GPIO.output(friend_pin,1)
        time.sleep(1)
        GPIO.output(enemy_pin,0)
        GPIO.output(friend_pin,0)

        GPIO.cleanup()
```