Kingdom of Saudi Arabia	المملكة العربية السعودية
Royal Commission at Yanbu	الهيئة الملكية بينبع
University College – Yanbu	الكلية الجامعية-ينبع
Yanbu Al-Sinaiyah	ينبع الصناعية

CSE Department

Introduction to Real-time Embedded Systems (CSE371)

SM172

Object Tracking Robot

(Using Raspberry Pi)

Supervised by:

Ms. Salma Hamoda

Student Name	Student ID
Nida Husain Mukhtar	3500903
Amal Ali Aleisa	
Tareef Ayaz Lodhi	3601168

Table of Contents

Abstract

The aim of this project is to create an object following robot that follows an object based on a specified color. The implementation is done using a Raspberry Pi, motor driver, Processing IDE, and a simple camera. It goes beyond previous ideas and results in a more advanced robot that can be useful in many applications such as surveillance.

1. Introduction

Nowadays surveillance has become crucial in protecting properties or even in identifying suspicious or wanted individuals. The major drawback or issue in today’s surveillance rests on the involvement of human operators which can easily be distracted. Therefore, we need a system which can autonomously monitor regions or objects continuously, making decisions while identifying unwanted or obnoxious things and responding accordingly. We can achieve this type of surveillance by building an object racking robot that uses technologies or concepts of computer vision which will achieve automated surveillance. To clarify, computer vision is a field of computer science that works on enabling computers to see, identify and process images to act the same way as the human vision does, and then provide appropriate output. Our overall plan starts by first finding the best way of implementing this idea. Then, finding the required software and hardware in the market. Finally, we can start by experimenting and implement the idea to produce a robot that can track a specific object depending on its color.

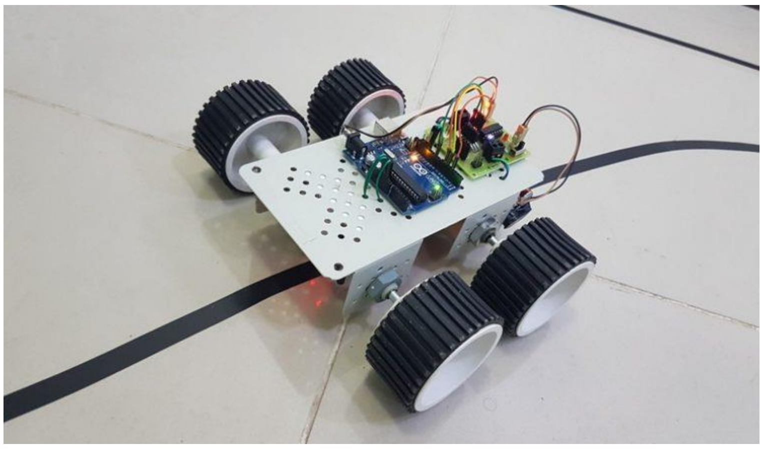
1.2 Literature Review

These are similar implementations of our projects:

1- Line Follower Robot

It is an automated guided vehicle which follows a line. Concept of working of line follower is related to light. We use here the behavior of light at black and white surface. When light fall on a white surface it is almost full reflected and in case of black surface light is completely absorbed. This behavior of light is used in building a line follower robot. In this arduino based line follower robot we have used IR Transmitters and IR receivers also called photo diodes. They are used for sending and receiving light. IR transmits infrared lights. When infrared rays falls on white surface, it’s reflected back and catched by photodiodes which generates some voltage changes. When IR light falls on a black surface, light is absorb by the black surface and no rays are reflected back, thus photo diode does not receive any light or rays.

Here in this arduino line follower robot when sensor senses white surface then arduino gets 1 as input and when senses black line arduino gets 0 as input.

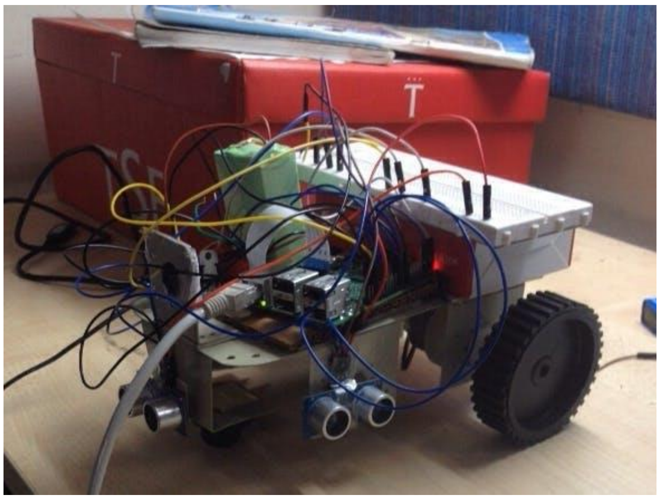


2- Ball Tracking Robot

Features:

a- It is a robotic car that follows a specific ball colored red.
b- It uses Raspberry Pi 3 as a microcontroller which controls its ultrasonic sensors, camera module and motors.
c- It is implemented using Python programming language.
d- It implements computer vision concept.
e- It uses OpenCV library in Python IDE.

This robot will stay still until it detects a red ball in front of it. It will do so by using the OpernCV library that implements computer vision. It will be constantly capturing footages using its camera module and comparing the image to an existing image already saved in its code. When it detects the ball, it will constantly move towards it.



2. Methodology

2.1 Project Description

Ball Tracking Robot using Processing IDE

Basically, this robot will first detect the color of the ball only when the operator introduces the ball to it using the camera module. When the operator clicks on the ball appearing in the live feed, the chosen color will be selected. It will then break down the footage into images then into pixels. By comparing each pixel to the saved color, it will detect when the required ball is in front of it. Then, it will constantly follow that ball until it is out of its sight.

Working:

The first things needed are the latest Raspbian and Processing ARM installed on the Raspberry Pi. Next, a camera, motor driver, and motors are connected to the Raspberry Pi. This circuit is then powered using a power source such as a battery.

Hardware:

To drive the motors, four pins (A, B, A, B) are used on the motor driver. They are connected to GPIO 14, 4, 17, and 18 on the Raspberry Pi. The motors are connected to the motor driver L293D and the motor driver is powered by the power source. All of these components are placed on a chassis of the object following robot.

Program:

The project aims to track an object. This is done by identifying the object using its color. Since videos are frames of pictures, each picture is split into pixels. Each pixel color is compared to the color of the object. If they match, then the object is tracked. This information is used to identify the position of the object (specifically the pixel color) on the screen. If the position is on the far left, the robot is moved to the right. If the position is far right the robot is moved to the left. This is done so that the position of the pixel says in the center of the screen. This means that the object is being tracked accurately and stays in the middle.

Program Steps:

a. The Hardware I/O and Glvideo libraries are imported. The first one is to access the GPIO pins of the PI from the processing environment. The second one is used to access the camera.
b. The output pins to control the motor and get the video from the camera are initialized. The camera window size is decided.
c. The video frame is split into pixels each with a value of red, green, and blue.
d. To decide which color to follow, the user can click on the color in the frame and is saved in a variable called trackColour. The track color and the current color are compared repeatedly.
e. When the value of dist is zero, the match is exact. If the value of dist is less than a specified value then it is assumed that the track color has been found. The location of that pixel is found and stored in closestx and closesty variables to get the location.
f. An ellipse around the found color means that the color needed has been found.
g. The position of the closestx and closesty are compared and motors are adjusted to make sure the color stays in the middle of the screen. GPIO pins are used to send signals to the motor driver which then moves the motors accordingly.

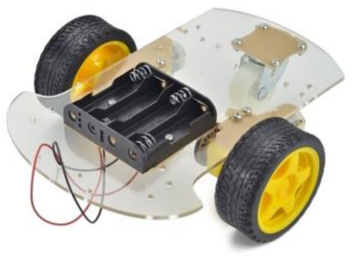
2.2 Hardware and Software Requirements

Hardware:

a. Raspberry Pi

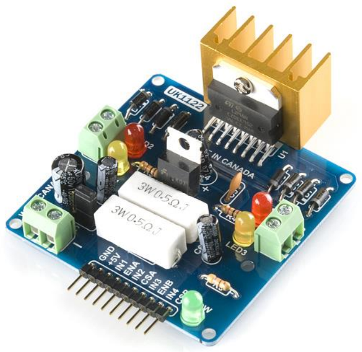
b. Camera

c. Robot Chassis



d. Gear motors with wheel

e. L293D motor driver



f. Power bank or any other portable power source
g. Monitor
h. Keyboard
i. Mouse

Software:

a. Processing ARM software
b. Raspbian

2.3 Timeline

Week 8-9: Setting up the software and hardware.

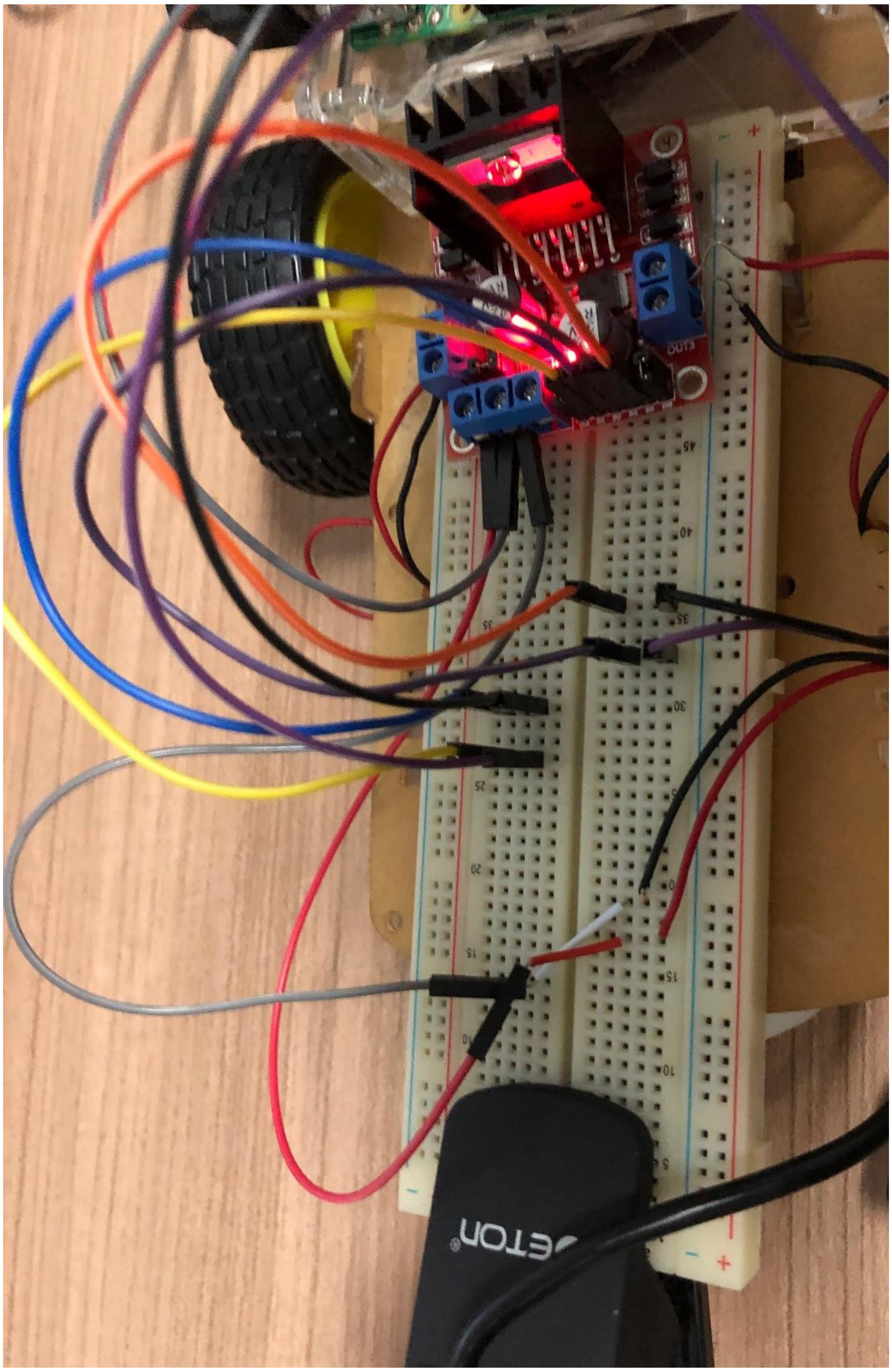
Week 10-11: Coding.

Week 11-12: Building Circuit.

Week 13-14: Testing the system.

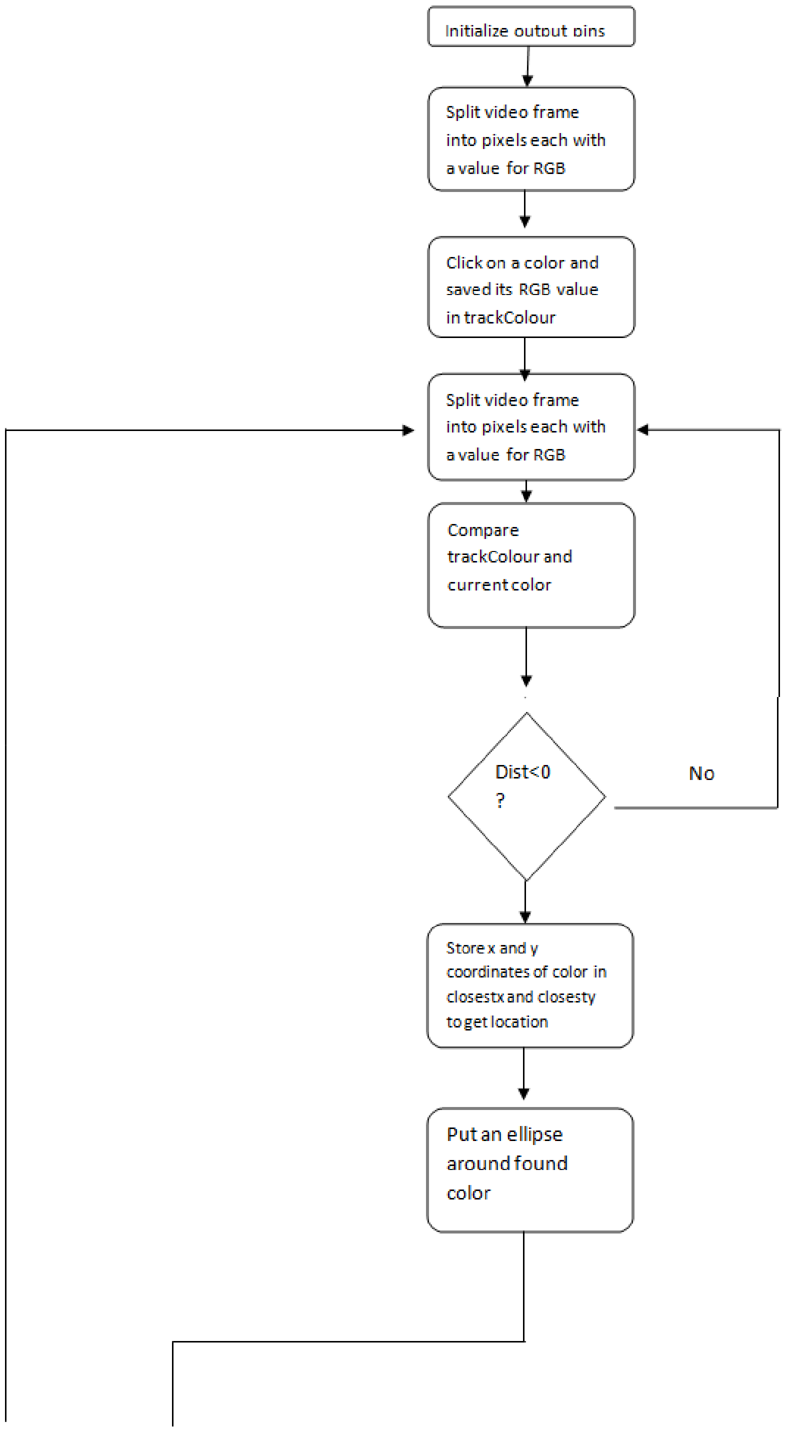
2.4 Block Diagram

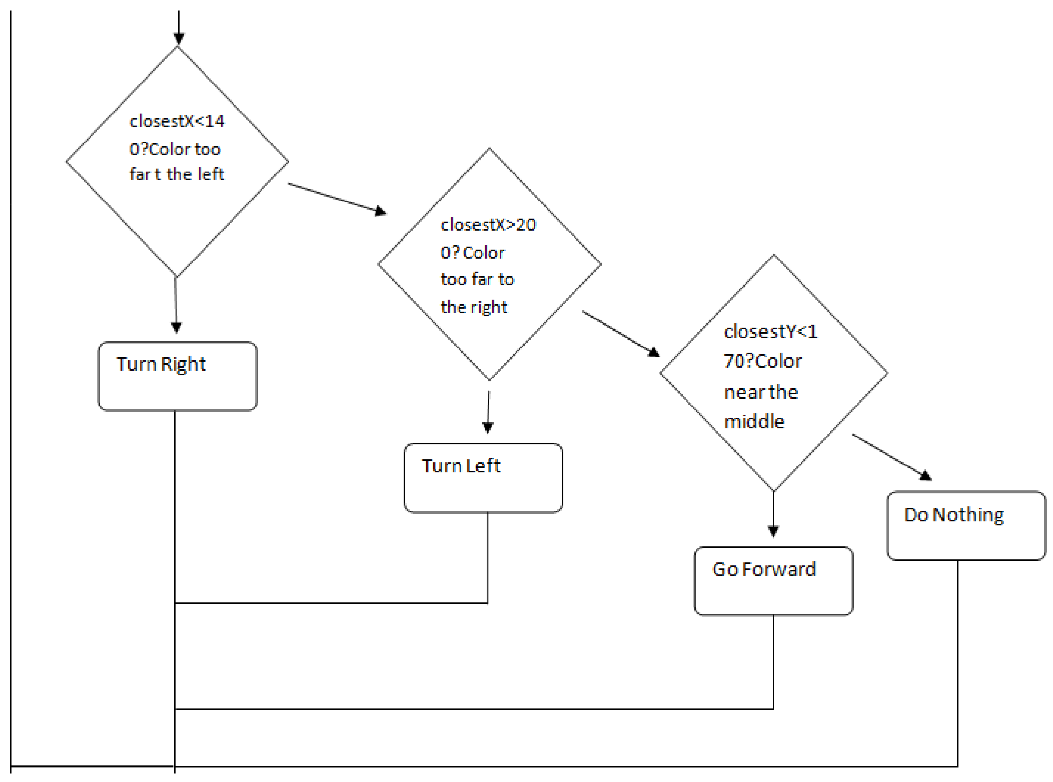
2.5 Circuit Diagram



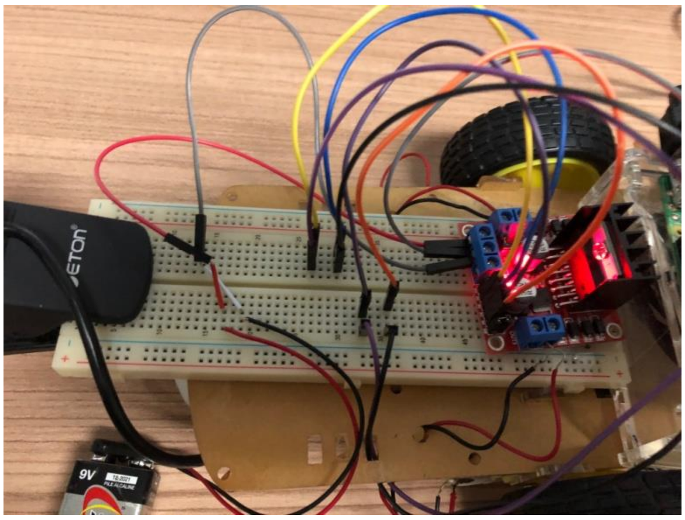
2.6 Implementation

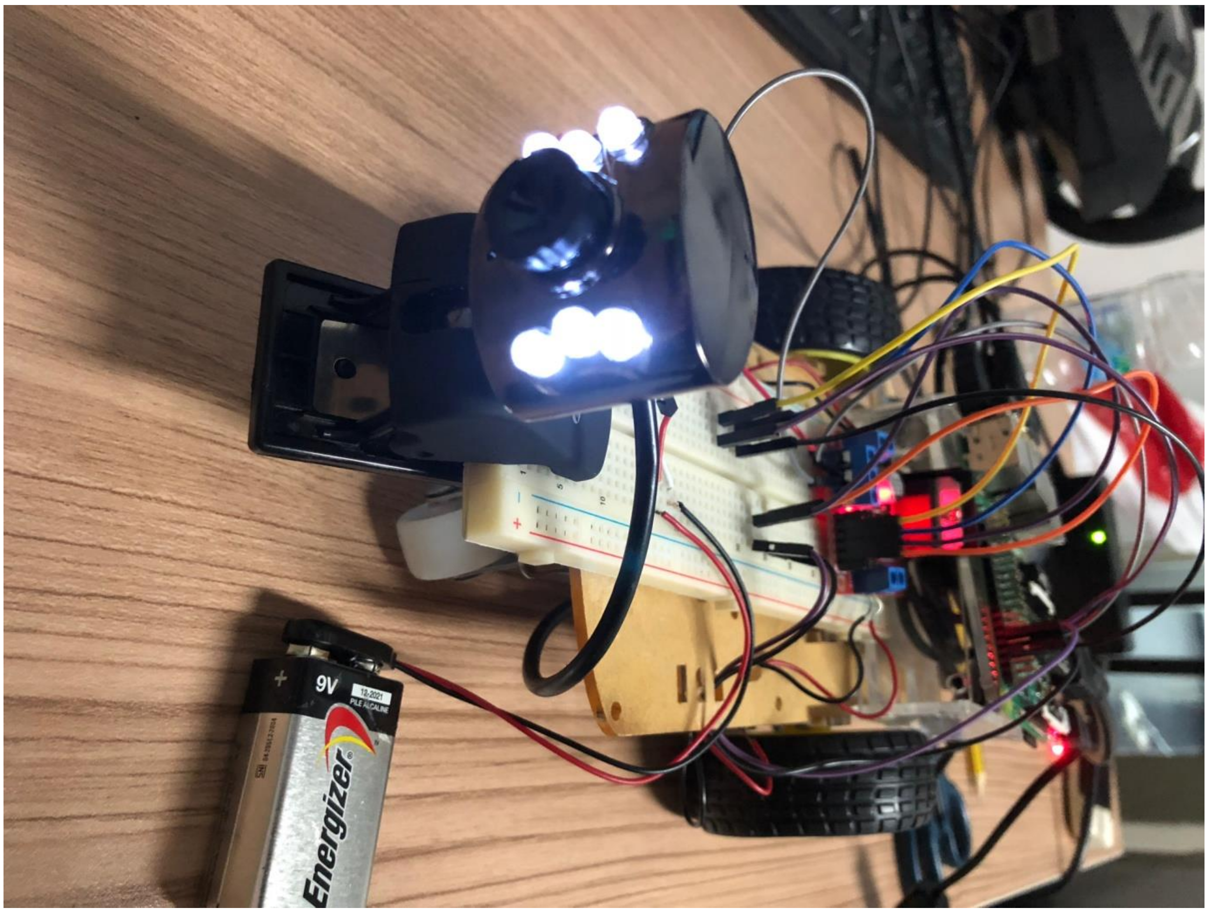
Flowchart


Initialize output pins
Split video frame into pixels each with a value for RGB
Click on a color and saved its RGB value in trackColour
Split video frame into pixels each with a value for RGB
Compare trackColour and current color
Dist<0 ?
No
Store x and y coordinates of color in closestx and closesty to get location
Put an ellipse around found color


closestX<140?Color too far t the left
Turn Right
closestX>200? Color too far to the right
Turn Left
closestY<170?Color near the middle
Go Forward
Do Nothing

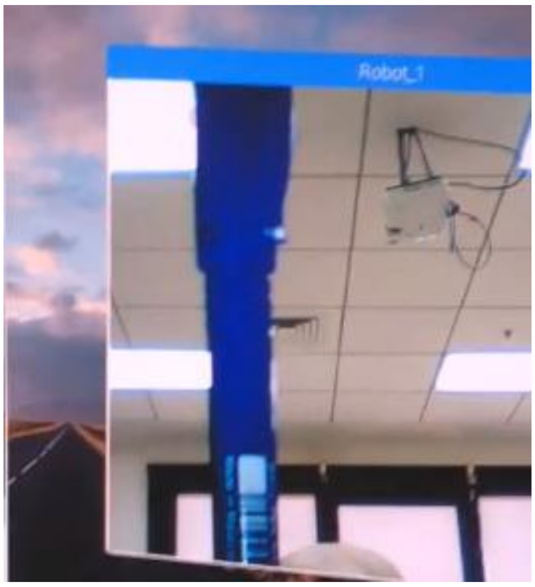
Final System





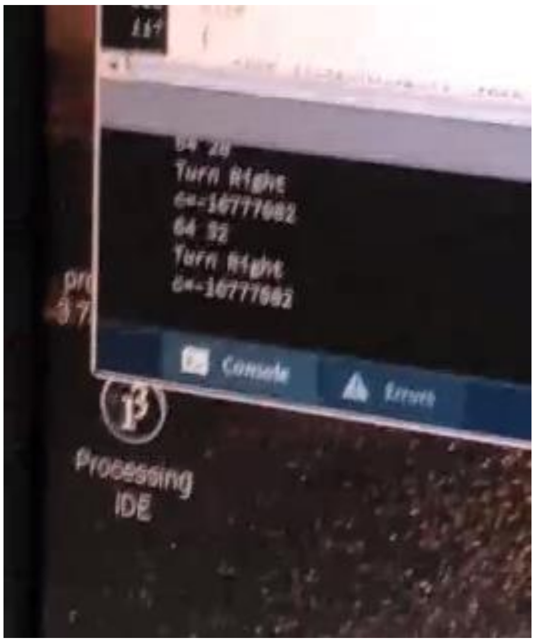
3. Experiment

The system was tested several times. The most important feature to test was the color detection and tracking. Different colored objects were placed in front of the system and colors were clicked on. Every time the color from the pixel was stored and the system followed the object with that certain color.



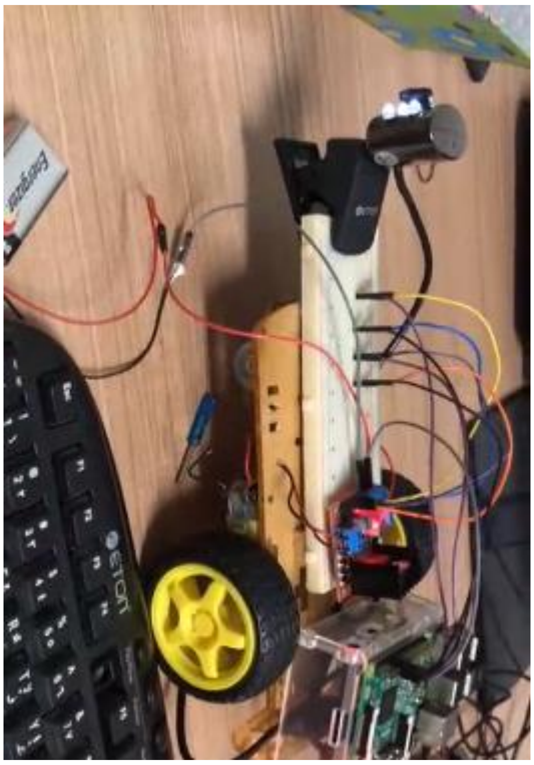
For instance, a blue area on this object was clicked on to follow.

The program ran correctly. It detected the color and therefore the object. Since the object is towards the far left, it indicated that the system must turn to the right to keep the object in the middle.



```
64 28
Turn Right
c=-16777082
64 32
Turn Right
c=-16777082
```

Once it the program was tested, the actual hardware was tested next. The motors were powered and attached to the motor driver; which was then attached to the Raspberry Pi. This was done to see whether or not the actual robot moves properly. This type of testing focused on the motors and the motor driver.



The object was tracked and the robot moved just as expected.

4. Conclusion

In conclusion, an object tracking robot was successfully built and implemented. This robot makes use of color to detect the object and adjust its position. This design can be used for surveillance, entertainment, etc. It can further be modified to include other features.

5. References

https://circuitdigest.com/microcontroller-projects/line-follower-robot-using-arduino

https://circuitdigest.com/microcontroller-projects/raspberry-pi-ball-tracking-robot-using-processing

https://www.hackster.io/junejarohan/ball-tracking-robot-7a9865

6. Appendix:

7. Code

```
import processing.io.*;
import gohai.glvideo.*;
GLCapture video;

color trackColor;

void setup() {
  size(320, 240, P2D);

  video = new GLCapture(this);

  video.start();

  trackColor = color(255, 0, 0);

  GPIO.pinMode(4, GPIO.OUTPUT);
  GPIO.pinMode(14, GPIO.OUTPUT);
  GPIO.pinMode(17, GPIO.OUTPUT);
  GPIO.pinMode(18, GPIO.OUTPUT);
}

void draw() {
   background(0);
  if (video.available()) {
    video.read();
  }
  video.loadPixels();
  image(video, 0, 0);

  float worldRecord = 500;
  int closestX = 0;
  int closestY = 0;

  // Begin loop to walk through every pixel
  for (int x = 0; x < video.width; x ++ ) {
    for (int y = 0; y < video.height; y ++ ) {
      int loc = x + y*video.width;
      // What is current color
      color currentColor = video.pixels[loc];
      float r1 = red(currentColor);
      float g1 = green(currentColor);
      float b1 = blue(currentColor);
      float r2 = red(trackColor);
      float g2 = green(trackColor);
      float b2 = blue(trackColor);

      // Using euclidean distance to
compare colors
      float d = dist(r1, g1, b1, r2, g2, b2); //
We are using the dist( ) function to
compare the current color with the color
we are tracking.

      // If current color is more similar to
tracked color than
      // closest color, save current location
and current difference
      if (d < worldRecord) {
        worldRecord = d;
        closestX = x;
        closestY = y;} }  }

  if (worldRecord < 10) {
    // Draw a circle at the tracked pixel
    fill(trackColor);
    strokeWeight(4.0);
    stroke(0);
    ellipse(closestX, closestY, 16, 16);
    println(closestX,closestY);

    if (closestX<140)
    {
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.HIGH);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18, GPIO.LOW);
     delay(10);
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.HIGH);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18, GPIO.HIGH);

     println("Turn Right");
    }
    else if (closestX>200)
    {
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.LOW);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18, GPIO.HIGH);
     delay(10);
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.HIGH);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18, GPIO.HIGH);
     println("Turn Left");
    }
    else if (closestY<170)
    {
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.LOW);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18, GPIO.LOW);
     delay(10);
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.HIGH);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18, GPIO.HIGH);
     println("Go Frwd"); }
    else
     {
     GPIO.digitalWrite(4, GPIO.HIGH);
     GPIO.digitalWrite(14, GPIO.HIGH);
     GPIO.digitalWrite(17, GPIO.HIGH);
     GPIO.digitalWrite(18,
GPIO.HIGH); } }  }
```

```
  else
  {
    GPIO.digitalWrite(4, GPIO.HIGH);
    GPIO.digitalWrite(14, GPIO.HIGH);
    GPIO.digitalWrite(17, GPIO.HIGH);
    GPIO.digitalWrite(18, GPIO.HIGH);
  }
}
```

```
void mousePressed() {
  // Save color where the mouse is clicked
in trackColor variable
  int loc = mouseX + mouseY*video.width;
  trackColor = video.pixels[loc];
}
```