



Faculty of Engineering

كلية الهندسة



# *Smart Autonomous Electric Vehicle* (SAEV)

*Supervised by:*

*DR. Essam Nabil*

*Team Members:*

- ❶ *Norhan Mohammed Mohammed Negm*
- ❷ *Youstina Atef Salama Salieb*
- ❸ *Aya Basuony El-Garhy*
- ❹ *Kyrillos Alaa Samy Rezk*
- ❺ *Halim Nader Halim Habib*

# TABLE OF CONTENTS

|   |           |
|---|-----------|
| <b>CHAPTER 1 INTRODUCTION .....</b>                               | <b>1</b>  |
| 1.1 Background .....  | 1         |
| 1.2 Problem Statement .....                                       | 1         |
| 1.3 Proposed solution.....  | 1         |
| 1.4 A general overview of the project.....                        | 2         |
| <b>CHAPTER 2 SURVEY OF EXISTING METHODS AND SIMILAR SYSTEMS.3</b> |           |
| 2.1 Introduction.....   | 3         |
| 2.2 History of Autonomous Cars .....                              | 3         |
| 2.3 The 5 Levels of Autonomous Vehicles.....                      | 5         |
| 2.3.1 Level 0 – No Automation .....                               | 5         |
| 2.3.2 Level 1 - Driver Assistance.....                            | 6         |
| 2.3.3 Level 2 - Partial Automation.....                           | 6         |
| 2.3.4 Level 3 – Conditional Automation .....                      | 6         |
| 2.3.5 Level 4 - High Automation .....                             | 7         |
| 2.3.6 Level 5 – Full Automation.....                              | 8         |
| 2.4 Conclusion and Motivation.....                                | 9         |
| <b>CHAPTER 3 SYSTEM DESIGN.....</b>                               | <b>10</b> |
| 3.1 Introduction.....   | 10        |
| 3.2 Requirements and Specification Analysis .....                 | 10        |
| 3.3 System Architecture.....                                      | 10        |
| 3.3.1 Action Module .....   | 12        |

|  |  |           |
|--|--|-----------|
| 3.3.2  | Vision Module .....  | 13        |
| 3.3.3  | Control Module.....  | 14        |
| 3.4  | State Diagrams .....   | 15        |
| 3.5  | Class Diagrams .....   | 16        |
| 3.6  | Sequence Diagrams.....   | 17        |
| 3.7  | Conclusion .....   | 17        |
| <b>CHAPTER 4 IMPLEMENTATION/SIMULATION AND TESTING .....</b> |  | <b>18</b> |
| 4.1  | Introduction.....  | 18        |
| 4.2  | Implementation Tools (Hardware).....                           | 18        |
| 4.2.1  | Raspberry pi .....   | 20        |
| 4.2.2  | Camera .....   | 20        |
| 4.2.3  | Arduino uno.....   | 21        |
| 4.2.4  | Tesla Cyber Track.....   | 21        |
| 4.2.5  | DC Gear Motor .....  | 21        |
| 4.2.6  | Power Bank (10,000 mAh) .....                                  | 22        |
| 4.2.7  | Li-Ion Battery.....  | 22        |
| 4.3  | Implementation Tools (Software):.....                          | 22        |
| 4.3.1  | Raspbian Operating System.....                                 | 22        |
| 4.3.2  | C++ Programming Language (Main System Code) .....              | 23        |
| 4.3.3  | OpenCV library.....  | 23        |
| 4.3.4  | WiringPi library .....   | 24        |
| 4.3.5  | Genie IDE & Compiler .....                                     | 24        |
| 4.3.6  | Mu IDE & Compiler .....  | 24        |
| 4.3.7  | Python Programming Language (Micro:Bit Compass Read and Serial |           |

|  |           |
|--|-----------|
| Communication) .....   | 24        |
| 4.3.8 Cascade Training GUI software.....                     | 25        |
| 4.4 Implementation Summary.....                              | 25        |
| 4.4.1 Road Lanes Tracking .....                              | 25        |
| 4.4.2 U-Turn.....  | 31        |
| 4.4.3 Signs, Objects and Traffic Lights Recognition.....     | 34        |
| 4.5 Test Cases and Acceptance Criteria.....                  | 38        |
| 4.5.1 Test 1 ( Lane Detecting and Tracking) .....            | 38        |
| 4.5.2 Test 2 (U-Turn) .....                                  | 39        |
| 4.5.3 Test 3 (Stop Sign Detection and Function).....         | 41        |
| 4.5.4 Test 4 (Object Avoidance).....                         | 43        |
| 4.5.5 Test 5 (Traffic Light Detecting).....                  | 45        |
| 4.6 Conclusion .....   | 45        |
| <b>CHAPTER 5 CONCLUSION AND FUTURE WORK .....</b>            | <b>46</b> |
| 5.1 Conclusion .....   | 46        |
| 5.2 Future Work.....   | 47        |
| <b>APPENDIX A: IMPLEMENTATION DETAILS .....</b>              | <b>48</b> |
| <b>APPENDIX B: USER MANUAL.....</b>                          | <b>51</b> |
| <b>APPENDIX C: DEPLOYMENT AND CONFIGURATION MANUAL .....</b> | <b>55</b> |

## LIST OF FIGURES

|   |    |
|---|----|
| Figure 2.1: Magic Highway.....                | 4  |
| Figure 2.2: Levels of driving automation..... | 5  |
| Figure 2.3: Toyota Corolla .....              | 7  |
| Figure 2.4: Tesla Autopilot .....             | 7  |
| Figure 2.5: Audi A8 2019.....                 | 8  |
| Figure 2.6: Waymo Smart Car .....             | 8  |
| Figure 3.1: System central diagram.....       | 11 |
| Figure 3.2: Layer 2, Action module .....      | 12 |
| Figure 3.3: The Smart Car (Front View).....   | 13 |
| Figure 3.4: The Smart Car (Side View).....    | 13 |
| Figure 3.5: Vision Module Diagram .....       | 14 |
| Figure 3.6: GUI Draft.....                    | 15 |
| Figure 3.7: System State Diagram.....         | 15 |
| Figure 3.8: System class diagrams .....       | 16 |
| Figure 3.9: System Sequence Diagram .....     | 17 |
| Figure 4.1: Raspberry pi .....                | 20 |
| Figure 4.2: Region of Interest (RoI).....     | 25 |
| Figure 4.3: Unwrapped region.....             | 26 |
| Figure 4.4: Thresholding an image.....        | 27 |
| Figure 4.5: Histogram Algorithm Screen .....  | 28 |

|   |    |
|---|----|
| Figure 4.6: Center Line Creating.....                                 | 30 |
| Figure 4.7: Center Difference Calculating .....                       | 31 |
| Figure 4.8: Different states for road lanes analysis .....            | 31 |
| Figure 4.9: U-turn sign .....   | 33 |
| Figure 4.10: Cascade Training GUI .....                               | 34 |
| Figure 4.11: Object Detecting using an XML Cascade file .....         | 36 |
| Figure 4.12: Car in front of the stop sign .....                      | 37 |
| Figure 4.13: Stop sign side width in px.....                          | 37 |
| Figure 4.14: Measuring the distance between the car and the sign..... | 38 |
| Figure 4.15: Road Lanes Tracking Testing Result.....                  | 38 |
| Figure 4.16: U-Turn Result .....                                      | 39 |
| Figure 4.17: Stop Sign Detection Result.....                          | 42 |
| Figure 4.18: Car Avoiding Result .....                                | 44 |
| Figure 4.19: Traffic Light Testing.....                               | 45 |
| Figure 5.1: H-Bridge Architecture [5].....                            | 49 |
| Figure 5.2: H-Bridge Connections [16].....                            | 49 |
| Figure 5.3: Powering the Rpi ON.....                                  | 51 |
| Figure 5.4: IP Scanner .....  | 52 |
| Figure 5.5: MobaExterm Session .....                                  | 53 |
| Figure 5.6: VNC Session.....  | 53 |
| Figure 5.7: Raspbian Desktop .....                                    | 53 |

|                                       |    |
|---------------------------------------|----|
| Figure 5.8: System windows .....      | 54 |
| Figure 5.14: Turning the Motors ..... | 54 |

## LIST OF TABLES

|  |    |
|--|----|
| Table 2.1: SAE Levels of autonomy .....                | 9  |
| Table 3.1: Action Module Labels.....                   | 13 |
| Table 3.2: Vision Module Parts.....                    | 14 |
| Table 4.1: Hardware Tools.....                         | 18 |
| Table 4.2: Positive Sample For Different Objects ..... | 35 |
| Table 4.3: Negative Samples.....                       | 35 |
| Table 4.4: Measured Values for object detection.....   | 36 |
| Table 5.1: Motor Commands.....                         | 50 |
| Table 5.2: Hardware Wiring.....                        | 55 |



## **LIST OF SYMBOLS**

AI: Artificial Intelligence

Cv: Computer Vision

openCV: open source computer vision

DC: Direct Current

Rpi: Raspberry pi

L298N: Motor Driver Chip

GHz: Giga Hertz

Px : pixel

Rpm: Rotation per minute

V: Volt

H: hour

A: Amper

mA: milli Amper

mAh: milli Amper per hour

mm: millimeter

cm: centimeter

m: meter

in: inch

Rol: Region of interest

UART: Universal Asynchronous Receiver-Transmitter

# **1. INTRODUCTION**

## **1.1 Background**

According to the National Highway Traffic Safety Administration (NHTSA), “They lost 35,092 people in crashes on U.S. roadways during 2015”.

An analysis revealed that about 94% of those accidents were caused by human error, and the rest by the environment and mechanical failures.

Road traffic accidents, with an estimated 37,133 lives lost on U.S. roads in 2017, are a leading cause of death. So, we are facing big problem that we have to solve using technology and science after god’s will.

As result, using computer vision, artificial intelligence, and machine learning, we can build a system able to recognize objects, signs, and lines, and then make a safer self-driving car.

## **1.2 Problem Statement**

Basically, the world now depends on vehicles as a means of transportation under human leadership, which led to the occurrence of traffic accidents, a matter that all countries suffer from, the most important of which is the sudden factor, weather conditions and dangerous roads that humans cannot deal with. Therefore, to enhance transportation safety and reduce accidents, some companies have turned to Self-driving vehicles by using sensors to sense and driving safely.

The challenge is to make sure that the control system for an autonomous car ensures safety regardless of human unpredictability.

## **1.3 proposed solution**

In order to deal with the stated problem, the system needs to address the following requirements from a user perspective:

- A camera, to analyze the environment (road lane lines and the side signs)

- An algorithm to solve different tasks such as line tracking, maze solving, avoid obstacles, sign detection, and exploration.
- Artificial intelligence and machine learning system to solve the unpredictable situations and train the system.

## **1.4 general overview of the project**

The project aims to use a camera connected to a Raspberry Pi computer, Arduino uno, motor driver and dc motors. Using computer control to build a prototype that demonstrates the self-driving car using image processing.

## **2. SURVEY OF EXISTING METHODS AND SIMILAR SYSTEMS**

### **2.1 Introduction**

Self-driving vehicles are cars or trucks in which human drivers are never required to take control to operate the vehicle safely. Also known as autonomous or “driverless” cars, they combine sensors and software to control, navigate, and drive the vehicle.

Currently, there are no legally operating, fully-autonomous vehicles. There are, however, partially-autonomous vehicles—cars and trucks with varying amounts of self-automation, from conventional cars with brake and lane assistance to highly-independent, self-driving prototypes.

### **2.2 History of Autonomous Cars**

In GM’s 1939 exhibit, Norman Bel Geddes created the first self-driving car, which was an electric vehicle guided by radio-controlled electromagnetic fields generated with magnetized metal spikes embedded in the roadway. By 1958, General Motors had made this concept a reality. The car’s front end was embedded with sensors called pick-up coils that could detect the current flowing through a wire embedded in the road. The current could be manipulated to tell the vehicle to move the steering wheel left or right.

In 1977, the Japanese improved upon this idea, using a camera system that relayed data to a computer to process images of the road. However, this vehicle could only travel at speeds below 20 mph. The improvement came from the Germans a decade later in the form of the VaMoRs, a vehicle outfitted with cameras that could drive itself safely at 56 mph.

Figure 2.1 (represents an image that was published in “Life” magazine in 1956).

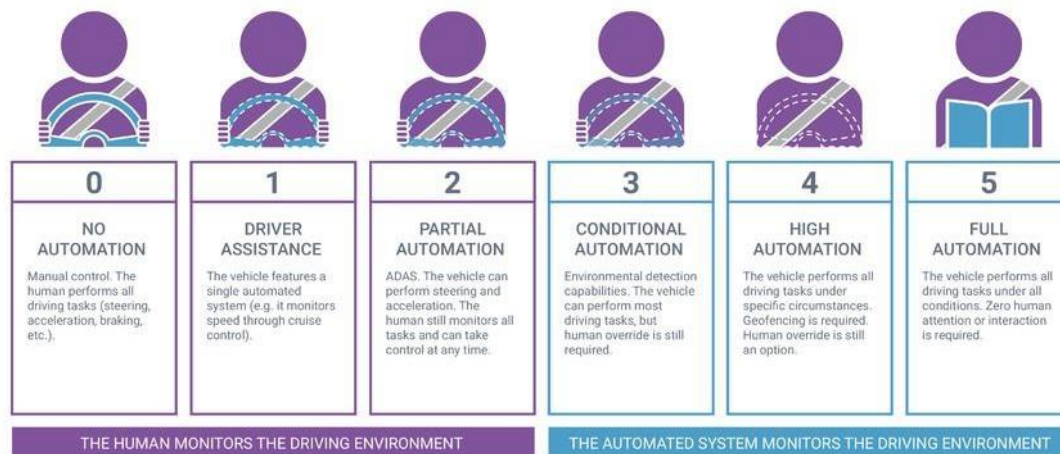


Figure 2.1: Magic Highway

## 2.3 The 5 Levels of Autonomous Vehicles

It's been over three decades since the famous 1982 TV series Knight Rider aired, showing the autonomous artificially intelligent automobile, K.I.T.T. What was once considered a far-fetched sci-fi character is well on its way to becoming a reality in the not-so-distant future. Although it may feel like autonomous vehicles are already among us with all the media coverage circling, our four-wheeled sidekicks are not quite as smart as K.I.T.T. just yet.

Figure 2.2 lists the levels of driving automation.



**Figure 2.2: Levels of driving automation**

### 2.3.1 Level 0 – No Automation

This describes your everyday car. No bells and whistles. Just your ordinary cruise control to help with long-distance driving and minimize the risk of a speeding ticket from a lead foot. Almost all cars today will offer Level 0 autonomous technology.

### **2.3.2 Level 1 - Driver Assistance**

Here we can find your adaptive cruise control and lane-keep assist to help with driving fatigue. Adaptive cruise control will keep a safe distance between you and the car ahead of you by using radars and/or cameras to automatically apply the brake when traffic slows, and resume speed when traffic clears. Lane will help nudge you back into the lane should you veer off a bit. These systems will assist drivers but still require the driver to be in control. You can find Level 1 autonomy in almost all cars today, including the 2018 Toyota Corolla (Toyota Safety Sense<sup>1</sup>, Figure 2.3) and the 2018 Nissan Sentra (Intelligent Cruise Control).

### **2.3.3 Level 2 - Partial Automation**

This is where it gets a bit more interesting. Although the driver must have hands on the wheel and be ready to take control at any given moment, level 2 automation can assist in controlling speed and steering. It will help with stop-and-go traffic by maintaining the distance between you and the vehicle in front of you, while also providing steering assist by centering the car within the lane. These features are a godsend for commuters! Tesla Autopilot (Figure 2.4), Volvo Pilot Assist, Audi Traffic Jam Assist are some examples of Level 2 autonomous capabilities.

### **2.3.4 Level 3 – Conditional Automation**

Here's where we start blurring the line between present technology and technology that's soon to come. Level 3 autonomous vehicles are capable of driving themselves, but only under ideal conditions and with limitations, such as limited-access divided highways at a certain speed. Although hands are off the wheel, drivers are still required behind the wheel. A human driver is still required to

take over should road conditions fall below ideal.

The next generation, 2019 Audi A8 (Figure 2.5), is expected to be the first to market a level 3 autonomous driving system<sup>3</sup>.



**Figure 2.3: Toyota Corolla**



**Figure 2.4: Tesla Autopilot**

### **2.3.5 Level 4 - High Automation**

Level 4 autonomous vehicles can drive themselves without human interactions (besides entering your destination) but will be restricted to known use cases.

We're not too far from seeing driverless vehicles out on public roads. Though regulations constrict its availability, Waymo (Figure 2.6) has developed and is in the process of testing Level 4 vehicles capable of driving themselves in most environments and road conditions. If there were no regulations or legal obstacles, you'd likely see more level 4 vehicles on the road today!





**Figure 2.5: Audi A8 2019**



**Figure 2.6: Waymo Smart Car**

### **2.3.6 Level 5 – Full Automation**

Super Pursuit Mode! At Level 5 autonomy, we arrive at actual driverless cars. Level 5 capable vehicles should be able to monitor and maneuver through all road conditions and require no human interventions, eliminating the need for a steering wheel and pedals.

**(Table 2.1 compares the levels according to SAE)**

**Table 2.1: SAE Levels of autonomy**

|  | SAE<br>LEVEL 0  | SAE<br>LEVEL 1  | SAE<br>LEVEL 2  | SAE<br>LEVEL 3   | SAE<br>LEVEL 4  | SAE<br>LEVEL 5  |
|--|---|---|---|--|---|---|
| What does the human in the driver's seat have to do? | You <u>are</u> driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering   |   |   | You <u>are not</u> driving when these automated driving features are engaged – even if you are seated in “the driver’s seat” |   |   |
|  | You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety                      |   |   | When the feature requests, you must drive  | These automated driving features will not require you to take over driving  |   |
|  | These are driver support features   |   |   | These are automated driving features   |   |   |
| What do these features do?                           | These features are limited to providing warnings and momentary assistance   | These features provide steering OR brake/acceleration support to the driver                           | These features provide steering AND brake/acceleration support to the driver  | These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met    | This feature can drive the vehicle under all conditions   |   |
| Example Features                                     | <ul style="list-style-type: none"><li>• automatic emergency braking</li><li>• blind spot warning</li><li>• lane departure warning</li></ul> | <ul style="list-style-type: none"><li>• lane centering OR</li><li>• adaptive cruise control</li></ul> | <ul style="list-style-type: none"><li>• lane centering AND</li><li>• adaptive cruise control at the same time</li></ul> | <ul style="list-style-type: none"><li>• traffic jam chauffeur</li></ul>  | <ul style="list-style-type: none"><li>• local driverless taxi</li><li>• pedals/steering wheel may or may not be installed</li></ul> | <ul style="list-style-type: none"><li>• same as level 4, but feature can drive everywhere in all conditions</li></ul> |

## 2.4 Conclusion and Motivation

It does not take much imagination to extend this concept beyond self-driving cars. Principally any robot or robot-like system—especially those whose decisions can impact human safety—could benefit from a similar classification system.

This project aims to build a smart car that is completely capable of self-driving in every situation. Using image processing is the primary key to build such a car. The method used will be an opening for level 5 of driving, which is fully automated, and there is no need for any human interaction.

### 3. SYSTEM DESIGN

#### 3.1 Introduction

The system is an experimental prototype, and then all hardware used are experimental and cannot be used in real applications. However, the concept and functionality of the hardware are almost the same as the hardware used in the real environment.

#### 3.2 Requirements and Specification Analysis

As mentioned before, this autonomous car can path a route surrounded by white lanes and recognize signs, traffic light, or objects and interact with them.

For the realization of this project, fourth primary devices are used; the first one is the Raspberry pi. The second one is the raspberry pi camera v2. The third device will be the motor driver connected to 4 dc motors and the fourth is Arduino uno.

These hardware devices will work together in this way. The camera will capture the image of the scene, trace the route using image processing, and the raspberry pi program will make the decision and will send signals to Arduino uno to send commands to the motor driver to move, turn, and stop the car.

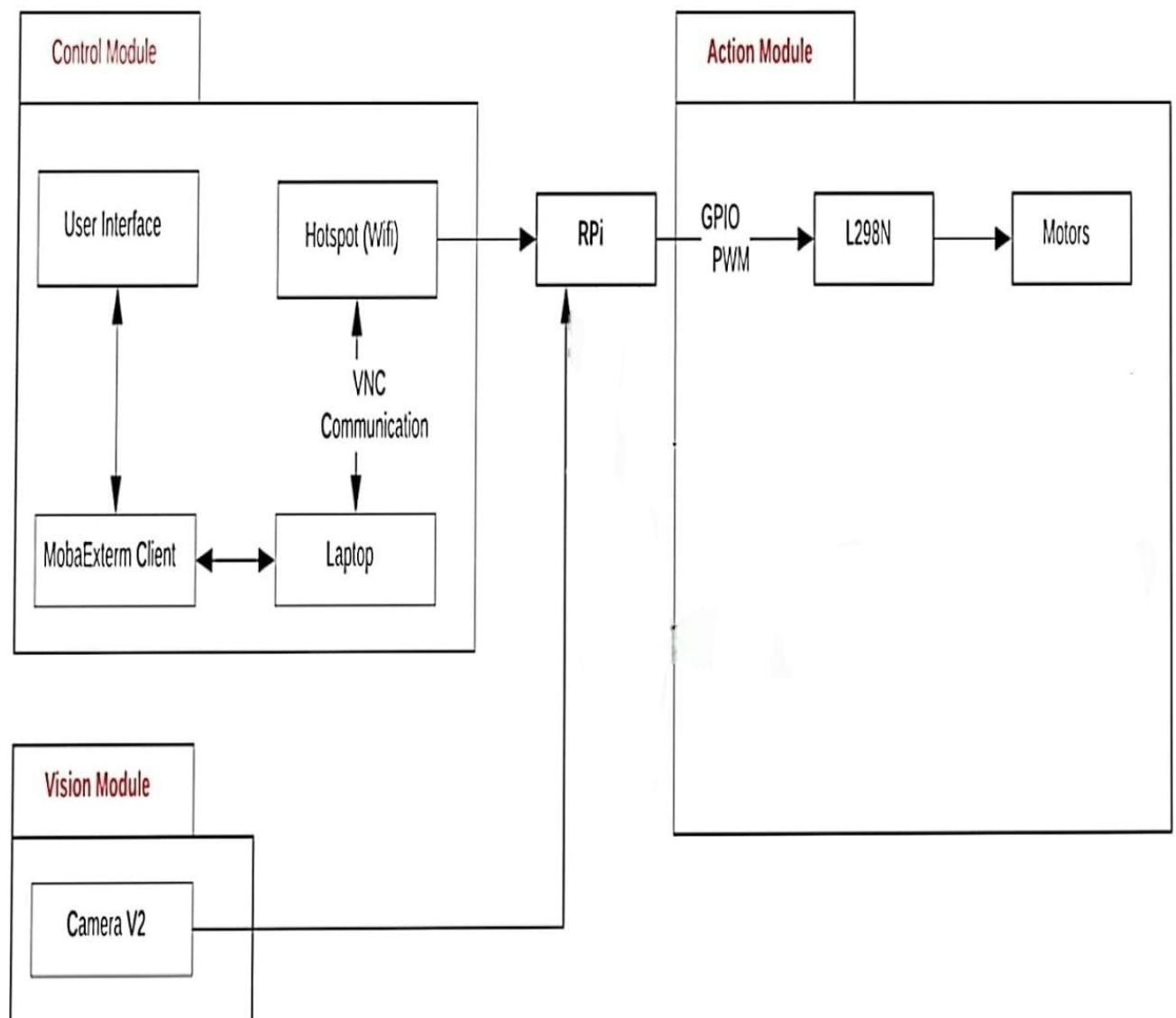
#### 3.3 System Architecture

To implement this scheme, we purpose the creation of three software modules (see Figure3.1):

1. **Action Module:** This module will be responsible for managing the control motors using the motor driver.

2. **Vision and computing module:** This module will be responsible for capturing the scene, analyzing it by detecting the obstacles using image processing, and calculate the path.

**3. User Control module:** This module will be responsible for performing the complete control of the system. It will be achieved by remote control using the VNC server.



**Figure 3.1: System central diagram**

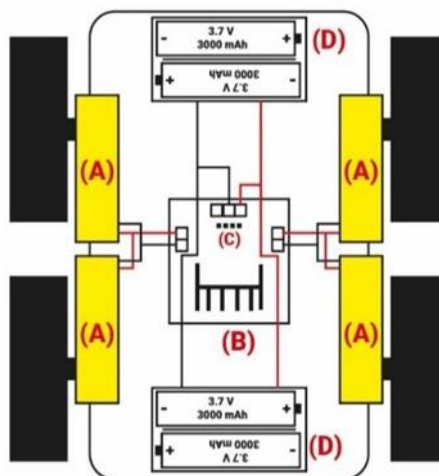
### 3.3.1 Action Module

This module is the responsible one for giving movement to the autonomous robot. It is an embedded hardware system. It comprises a Tesla cyber track. It takes the orders from the raspberry pi. Both sides can go forward and backward independently, and the robot can rotate about its origin. This technique allows the robot to make a piecewise linear path.

Moving the car depends on the power applied to each side left and right.

See First The motor should be calibrated to move forward when the power applied on both sides is the same. Some mechanical modifications may be used in addition to software modifications (making a side faster than the other) to make the robot move straight precisely.

Figure 3.2 represents the top view of the action module layer, and Table 3.1: Action Module Labels of it.



**Figure 3.2: Layer 2, Action module**

**Table 3.1: Action Module Labels**

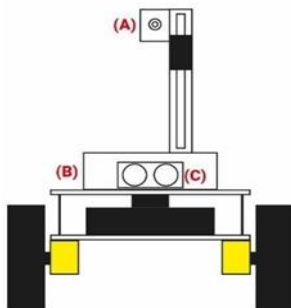
|     |                                     |
|-----|-------------------------------------|
| (A) | DC gear motor with a wheel          |
| (B) | H-Bridge L298N                      |
| (C) | Control pins to be connected to RPi |
| (D) | Battery Box                         |

### 3.3.2 Vision Module:

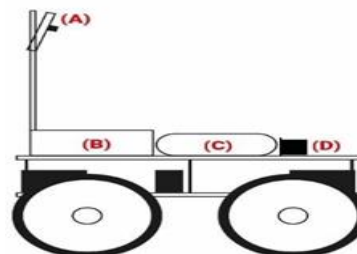
This module is completely embedded in the Raspberry pi system. It is responsible for interpreting the scene using an image, detecting the obstacles, and generating paths (see Figure3.3, Figure 3.4, and Table 3.2: Vision Module Parts Table 3.2 for labels reference).

This module can also be decomposed into a few sub-modules (Figure 3.5):

- **Image Acquisition:** This will capture the image of the scene.
- **Lane lines detection:** Will be responsible for detecting the white markers on the street.
- **Object Detection:** Will be responsible for detecting the obstacles, signs, and trafficlighs.



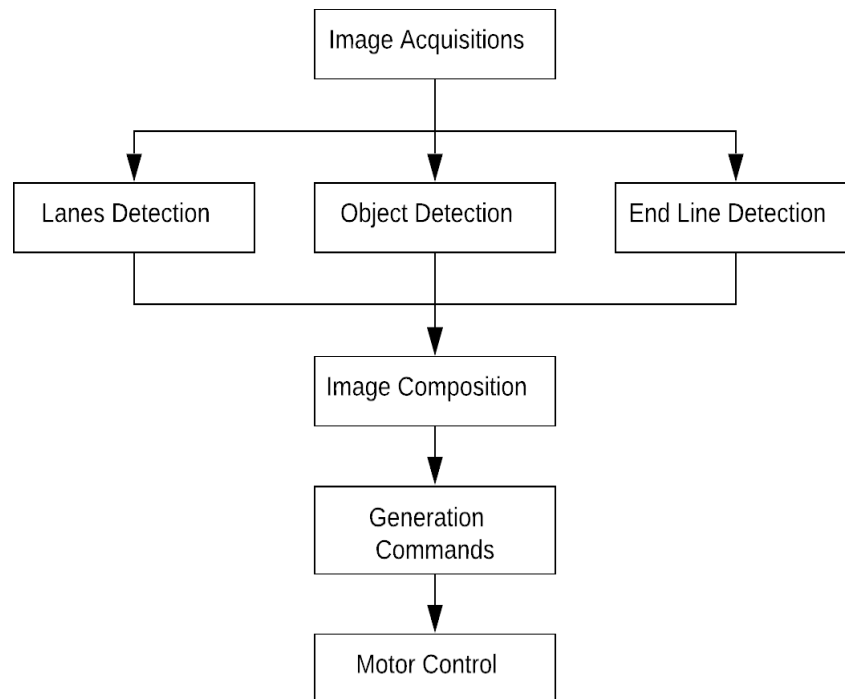
**Figure 3.3: The Smart Car (Front View)**



**Figure 3.4: The Smart Car (Side View)**

**Table 3.2: Vision Module Parts**

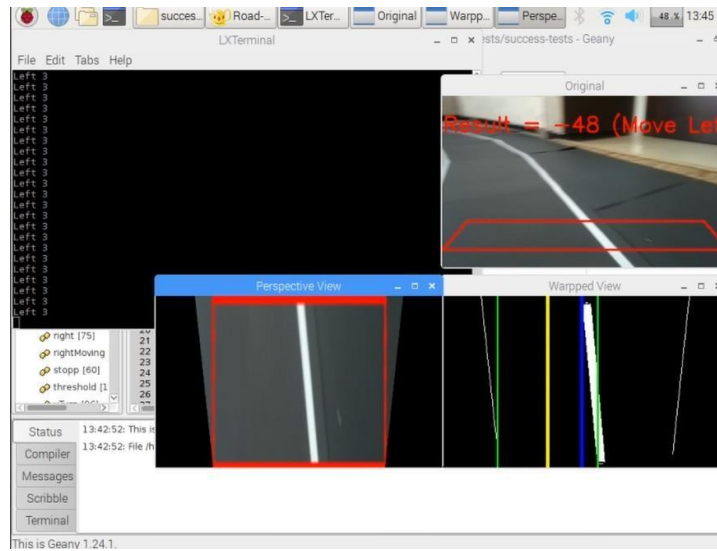
|     |                        |
|-----|------------------------|
| (A) | Raspberry Pi Camera V2 |
| (B) | Raspberry Pi           |
| (C) | Power Bank             |



**Figure 3.5: Vision Module Diagram**

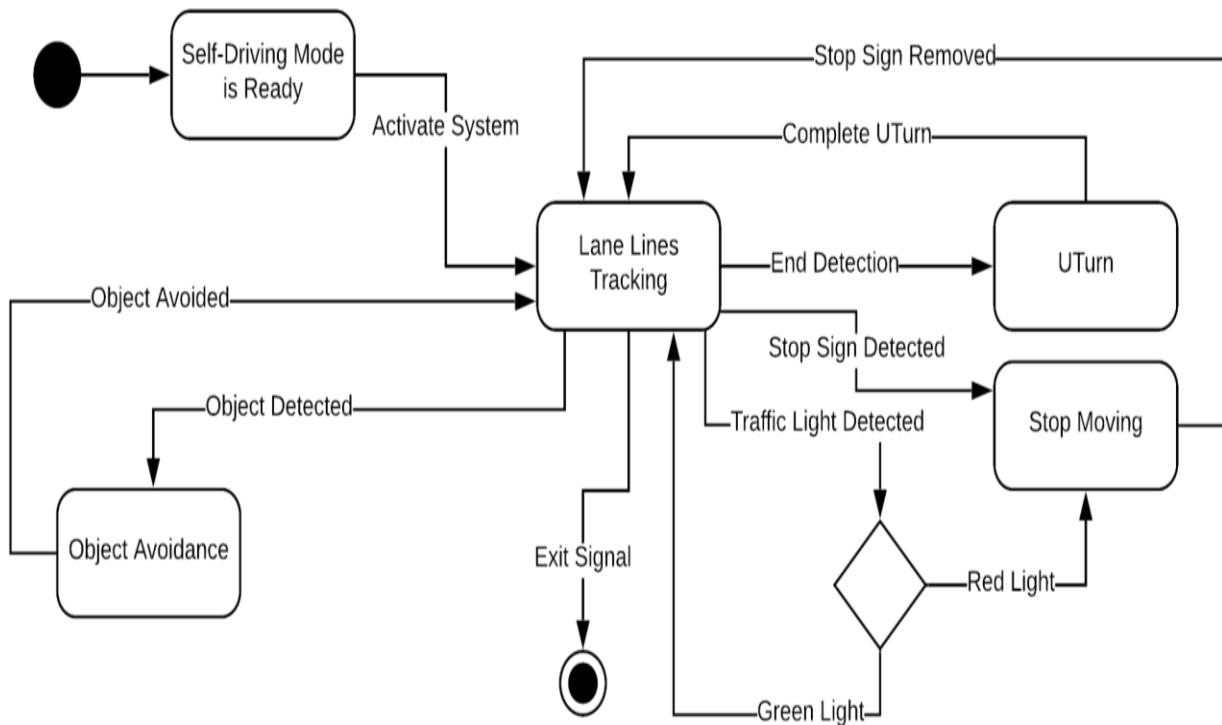
### 3.3.1 Control Module

To be able to control the entire system and to see what the raspberry pi is watching, a GUI is used to act as a client to access the different tasks and analyze the car movement. Figure 3.6 represents the initial windows for the user to analyze the overall system of the car.



**Figure 3.6: GUI Draft**

### 3.4 State Diagrams



**Figure 3.7: System State Diagram**



### 3.5 Class Diagrams

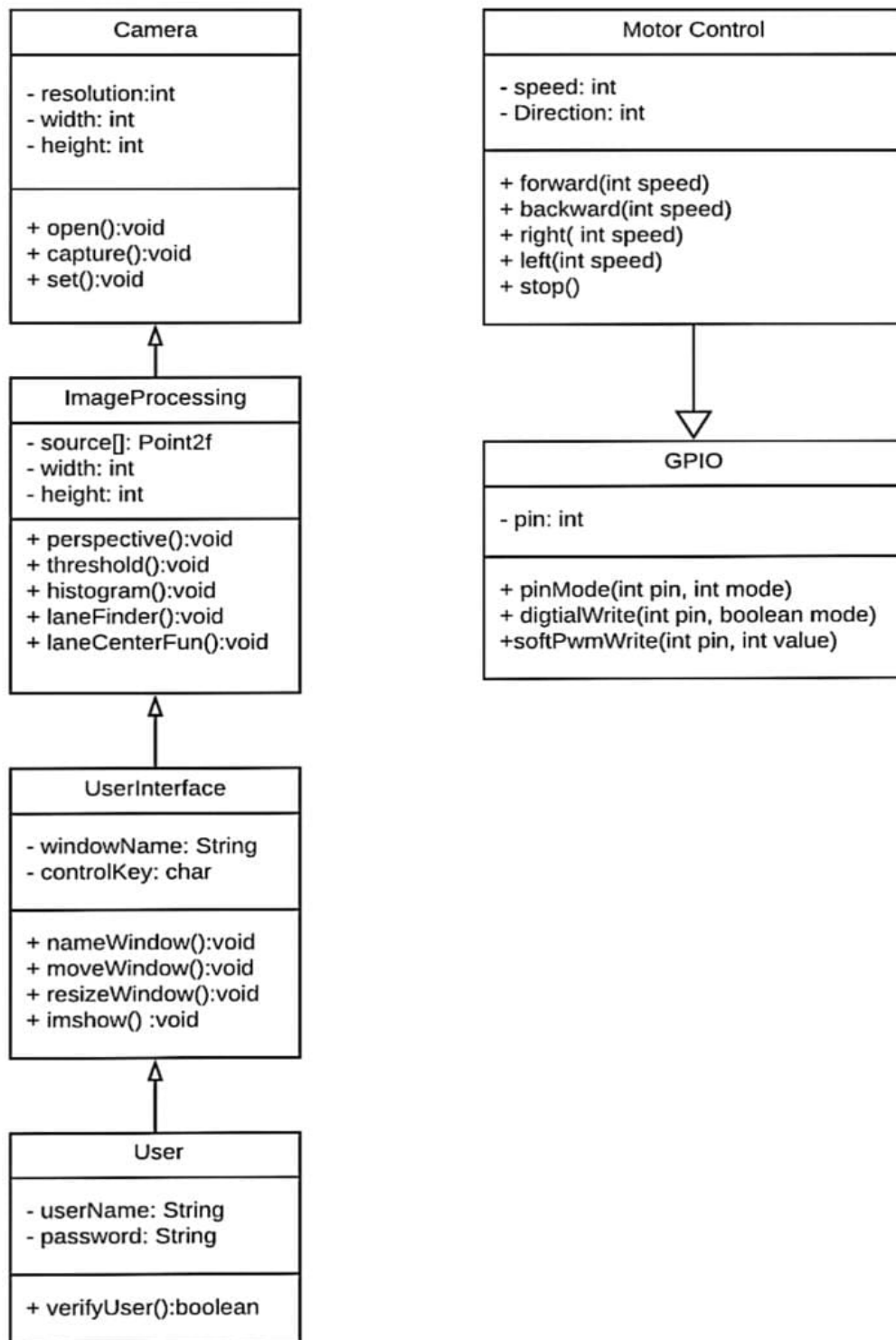


Figure 3.8: System class diagrams

### 3.6 Sequence Diagrams

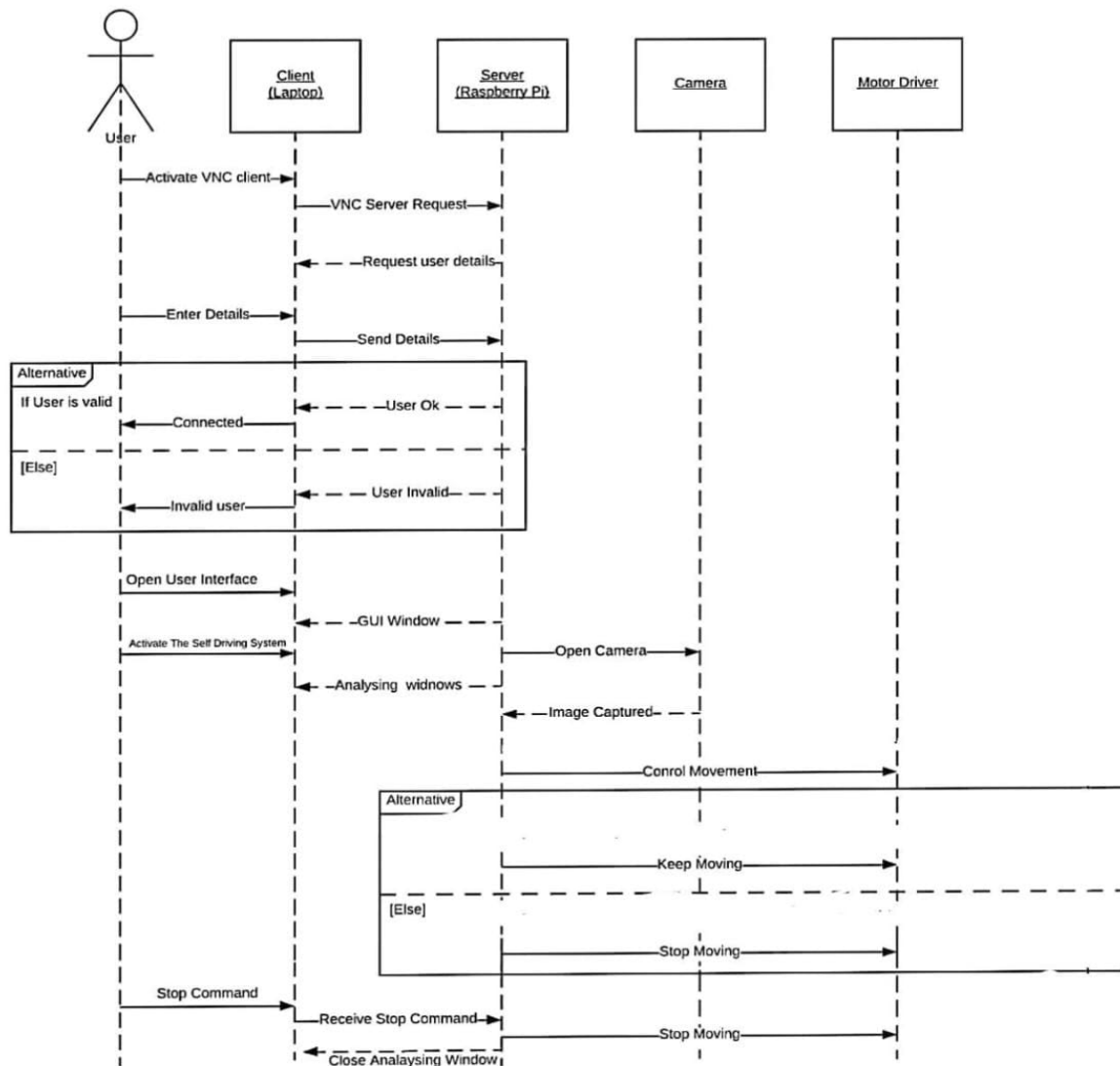


Figure 3.9: System Sequence Diagram

### 3.7 Conclusion

By listing the requirements for the project, analyzing and discovering the different components, the idea almost completed. In this chapter, the system architecture was designed completely. Moreover, the system functionality was stated and clarified step by step through class, sequence, and state machine UML diagrams.

## 4. IMPLEMENTATION/SIMULATION AND TESTING

### 4.1 Introduction

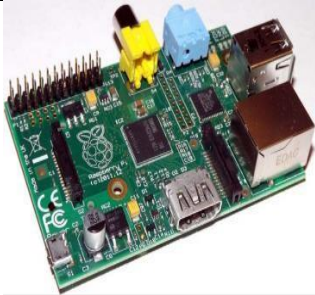

Using the open-source computational hardware and software, we can build a complete system that uses the image processing algorithm to solve complicated tasks such as lane tracking and signs detecting.

This chapter explains the hardware and software used to implement the whole project.

### 4.2 Implementation Tools (Hardware)

The system requires computing ability with camera vision and motor driving. This section explains the hardware used to satisfy the system requirements. The Table 4.1 lists the hardware used in the project.

**Table 4.1: Hardware Tools**

|   |   |
|---|---|
|  |  |
| <b>Raspberry Pi Model B</b>   | <b>Raspberry Pi Camera</b>  |



**Motor wheel**



**Dc gear motor with encoder**



**L298N Motor Driver [5]**



**Power bank**



**Lithium-Ion battery**



**Arduino uno**



**Tesla cyber track**

### 4.2.1 Raspberry pi

The Raspberry Pi is a single-board computer created by the Raspberry Pi Foundation; a charity formed with the primary purpose of reintroducing low-level computer skills to children in the UK. The aim was to rekindle the microcomputer revolution from the 1980s, which produced a whole generation of skilled programmers [6]. Figure 4.1 shows the raspberry pi 3B+ specifications.

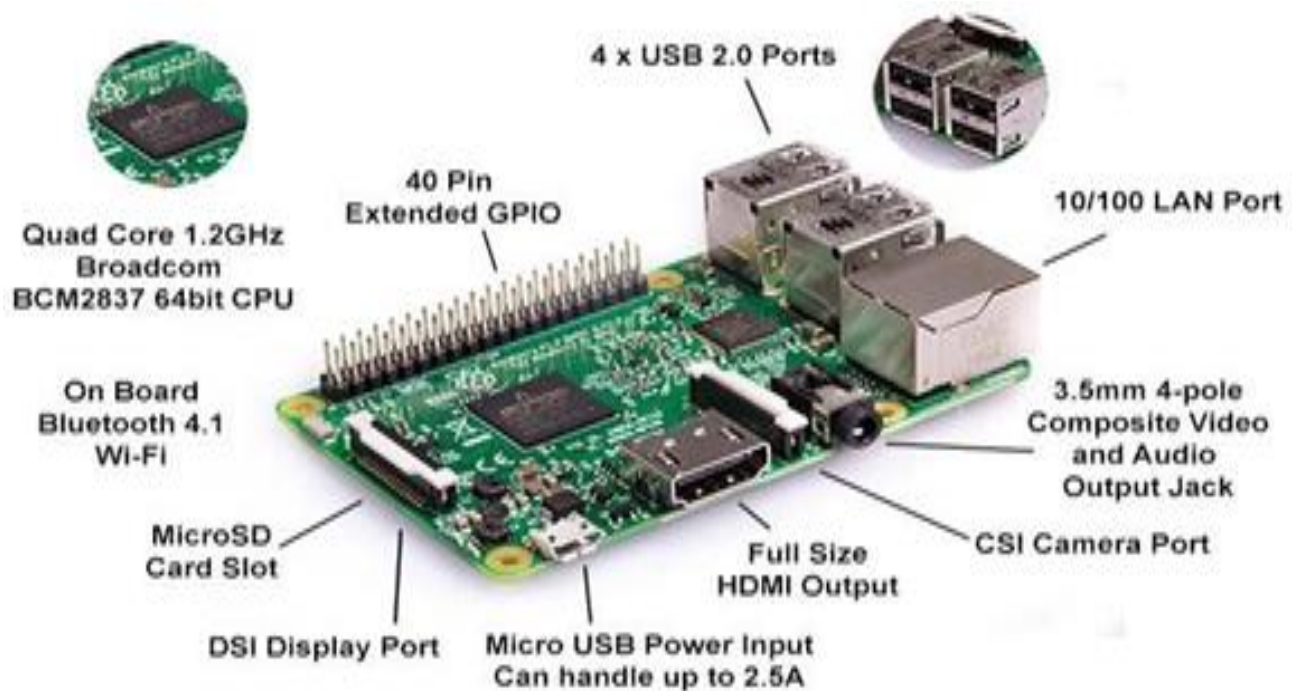


Figure 4.1: Raspberry pi

### 4.2.2 Camera

The Raspberry Pi Camera v2 is a high quality 8-megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464-pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSI interface, designed especially for interfacing with cameras.

### **4.2.3 Arduino uno**

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller (MCU) and developed by Arduino.cc and initially released in 2010. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by a USB cable or a barrel connector that accepts voltages between 7 and 20 volts, such as a rectangular 9-volt battery.

### **4.2.4 Tesla cyber truck**

We use 3d printing technology to print the frame of tesla cyber truck model. We use plastic material to minimize cost and at same time it performs required function.

### **4.2.5 DC Gear Motor with encoder**

(25GA370 DC 12V Micro Gear Reduction Motor 25mm Extended Shaft Gear Motor)

#### **Specifications:**

- Brand: Generic
- Material: Metal
- Voltage: DC 12V
- Power: 3W
- Speed: 200RPM
- Stall Torque: 4.5Kg/cm
- Dimensions:
  - Output Shaft Dia.: 4mm
  - Output Shaft Length: 25mm

- Motor: 50 x 24mm (without output shaft)

#### **4.2.6 Power Bank (10,000 mAh)**

USB power banks are capable of powering a Raspberry Pi and a camera module since they usually have an output voltage of 5 V. The power bank used is 10,000 mAh.

According to actual Raspberry Pi power measurements, the Model 3B consumes about 520 mA when shooting 1080p video. The actual runtime should be somewhere between 7 h and 14 h, depending on the setup.

#### **4.2.7 Li-Ion Battery**

A Lithium-polymer (Li-Po) is quite an old technology that you can find in your old, bar phones, or laptops. These batteries have a similar structure like Li-ion batteries but are made of a gel-like (Silicon-Graphene) material, which is quite light in weight. Due to its light and flexible characteristics, these batteries are used in laptops and most of the high-capacity power banks.

This type of battery is used for the motors powering. Two batteries (LiPo 3.7V, 3000 mAh) are connected serially to deliver about 7V for each motor side.

### **4.3 Implementation Tools (Software):**

The system uses many programming languages and methodologies. Each programmable hardware works in a different environment. To combine all the hardware together, the system uses the following software.

#### **4.3.1 Raspbian Operating System**

The operating system of the car is Raspbian. It is the Raspberry pi Foundation's official supported operating system. It is a Linux Distribution. Rather than a brand-new OS, Raspbian is a modified version of the popular Debian Squeeze Wheezy distro.

### 4.3.2 C++ Programming Language (Main System Code)

C++ is a high-level object-oriented programming language that helps programmers write fast, portable programs. C++ provides rich library support in the form of Standard Template Library (STL).

### 4.3.3 OpenCV library

OpenCV (Open-Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language, it is a library used for Image Processing. It is mainly used to do all the operations related to Images.

#### 4.3.3.1 *What it can do:*

1. Read and Write Images.
2. Detection of faces and their features.
3. Detection of shapes like Circle, rectangle, etc. in an image.  
E.g., Detection of a coin in images.
4. Text recognition in images, Reading Number Plates.
5. Modifying image quality and colors, e.g., Instagram, Cam Scanner.
6. Developing Augmented reality apps.

#### 4.3.3.2 *Which Language it supports:*

1. C++
2. Android SDK
3. Java
4. Python



#### **4.3.4 Wiring Pi library**

Wiring Pi is a PIN-based GPIO access library written in C for the BCM2835, BCM2836, and BCM2837 SoC devices used in all Raspberry Pi. Versions. It was released under the GNULGPLv3 license and is usable from C, C++, and RTB (BASIC), as well as many other languages with suitable wrappers. It was designed to be familiar to people who have used the Arduino “wiring” system<sup>1</sup> and is intended for use by experienced C/C++ programmers. It is not a newbie learning tool.

#### **4.3.5 Genie IDE & Compiler**

Genie is a powerful, stable, and light weight programmer's text editor that provides tons of useful features without bogging down the workflow. It runs on Linux, Windows and macOS is translated into over 40 languages and has built-in support for more than 50 programming languages.

#### **4.3.6 Mu IDE & Compiler**

The Micro Bit board can be programmed using Mu IDE. Mu is a simple code editor for beginner programmers based on extensive feedback from teachers and learners. Mu is for anyone who wants to use a simple "no-frills" editor.

#### **4.3.7 Python Programming Language (Micro Bit Compass Read and Serial Communication)**

Python is an interpreted, high-level, general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

### 4.3.8 Cascade Training GUI software

Cascade Trainer GUI is a program that can be used to train, test and improve cascade classifier models. It uses a graphical interface to set the parameters and make it easy to use OpenCV tools for training and testing classifiers.

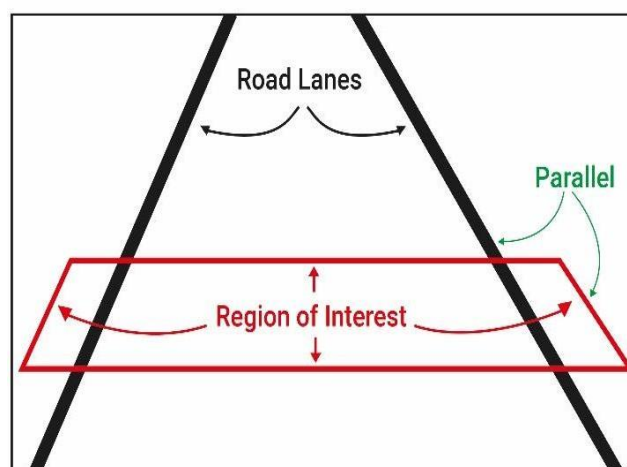
## 4.4 Implementation Summary

After capturing the environment images using the camera and the Rpi, the images are processed to solve different problems. The videos captured was cropped to 400 \* 240 screen to optimize the processing performance.

### 4.4.1 Road Lanes Tracking

#### 4.4.1.1 *Creating the region of interest:*

In order to follow the track, the system creates a region of interest to process it. This region follows the road lanes shape concerning the camera position. Figure 4.2 shows the RoI to be processed. The whole frame is the images captured and black bold lines are the road lines. This region should be drawn and calibrated on the screen, and the edges should be parallel to the road lanes lines.



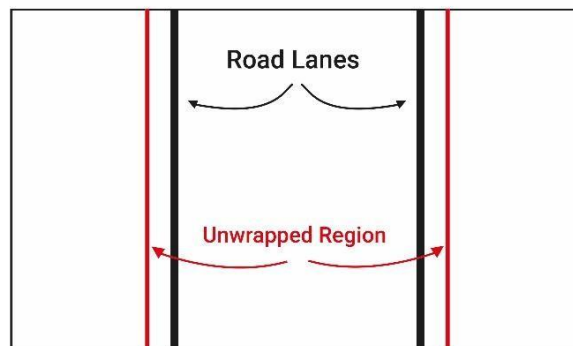
**Figure 4.2: Region of Interest (RoI)**

Code:

```
void Capture()
{
    Camera.grab();
    Camera.retrieve( frame);
    cvtColor(frame, frame_Stop, COLOR_BGR2RGB);
    cvtColor(frame, frame_Object, COLOR_BGR2RGB);
    cvtColor(frame, frame_Uturn, COLOR_BGR2RGB);
    cvtColor(frame, frame, COLOR_BGR2RGB);    }
```

#### 4.4.1.2 Unwrap the RoI:

After creating the RoI, this region is unwrapped into a rectangle shape to see the road lines from the top. Figure 4.3 shows the result of unwrapping the RoI.



**Figure 4.3: Unwrapped region**

Code:

```
void Perspective()
{
    line(frame,Source[0], Source[1], Scalar(0,0,255), 2);
    line(frame,Source[1], Source[3], Scalar(0,0,255), 2);
    line(frame,Source[3], Source[2], Scalar(0,0,255), 2);
    line(frame,Source[2], Source[0], Scalar(0,0,255), 2);
}
```

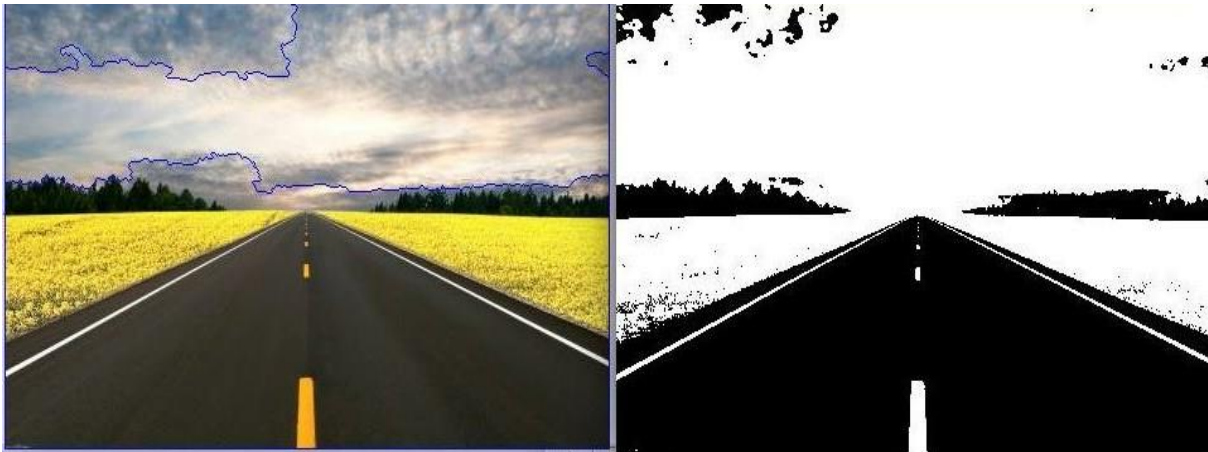
```

Matrix = getPerspectiveTransform(Source, Destination);
warpPerspective(frame, framePers, Matrix, Size(400,240));
}

```

#### 4.4.1.3 Thresholding Colors:

For precise movement, the unwrapped images should be transformed into black and white images. This can be done by converting the image into the gray color system and then converting the maximum intensity into white color and the others into black (See Figure 4.4). By Thresholding, the analysis is much more comfortable and Finer.



**Figure 4.4: Thresholding an image**

Code:

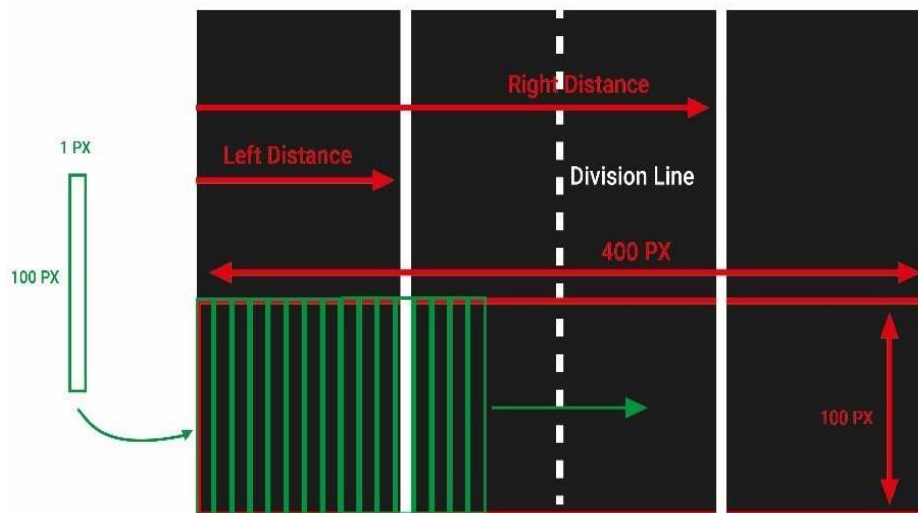
```

void Threshold()
{
    cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
    inRange(frameGray,150,255, frameThresh);
    Canny(frameGray,frameEdge, 900, 900, 3, false);
    add(frameThresh, frameEdge, frameFinal);
    cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
    cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR); //used in
histogram function only
    cvtColor(frameFinal, frameFinalDuplicate1, COLOR_RGB2BGR); //used in
histogram function only
}

```

#### 4.4.1.4 Calculating the distances

After getting the black and white image, the system calculates the distance between the left side and each road line (left and right). The calculating starts by dividing the unwrapped thresholded image into a rectangle of the screen width (400 px) and 100 px height starting from the bottom left screen point. Then it creates a small rectangle (100 px \* 1 px) among the whole rectangle, which means 400 small rectangles will be created. After this, the system calculates the white intensity of each small rectangle and pushes it into an array. The black intensity is 0 and the white is 255. To calculate the distance for each line, the screen is divided into two halves (Left and Right). The distance between the left line and the screen left edge is the number of the array element of the highest Intensity number. The same method is used for the right line. Figure 4.5 shows the histogram rectangles, the division line, distances, and the processing rectangle.



**Figure 4.5: Histogram Algorithm Screen**

Code:

```
void Histogram()
{
    histogramLane.resize(400);
    histogramLane.clear();
}
```

```

for(int i=0; i<400; i++)    //frame.size().width = 400
{
    ROILane = frameFinalDuplicate(Rect(i,140,1,100));
    divide(255, ROILane, ROILane);
    histogramLane.push_back((int)(sum(ROILane)[0]));
}
}

```

#### 4.4.1.5 Creating the centerline:

Now the left and right distance are calculated. The center distance is the half distance between the two lines. It can be calculated as follow:

$$centerDistance = \frac{rightDistance - leftDistance}{2} + leftDistance$$

The center distance is used to draw a vertical line, as shown in Figure 4.7.

After creating the centerline, we calculate the difference between the screen half (200 px in our case) and the centerline. In order to keep the car in the center, the difference should be zero (see Figure 4.8).

Code:

```

void LaneCenter()
{
    laneCenter = (RightLanePos-LeftLanePos)/2 +LeftLanePos;
    frameCenter = 192;

    line(frameFinal,    Point2f(laneCenter,0),    Point2f(laneCenter,240),
    Scalar(0,255,0), 3);
    line(frameFinal,    Point2f(frameCenter,0),    Point2f(frameCenter,240),
    Scalar(255,0,0), 3);
}

```

```
Result = laneCenter-frameCenter;  
}
```

#### *1.1.1.1 Movement Functions:*

The center differential is used to adjust the movement of the car.

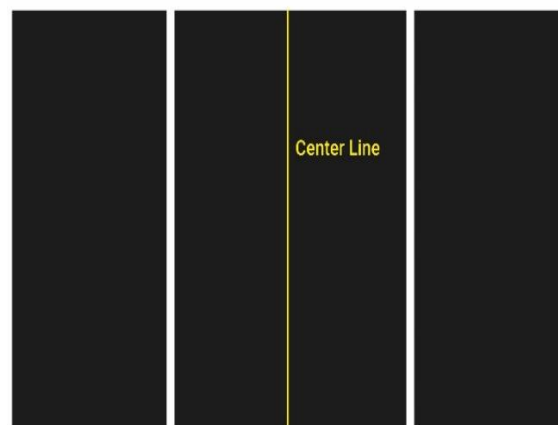
If the difference is higher than  $-5$  px and less than  $5$  px, the car should move straight.

If the difference is positive, the car moves right, and as the difference increase, the moving should be sharper. If the difference is negative, the car moves left, and the speed of moving increases with respect to the difference also.

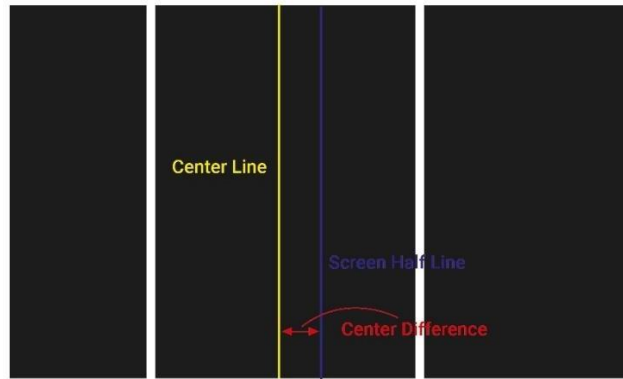
Figure 4.6 shows different states for the car and the feedback.

For more details about the movement functions, review Appendix A and the code in the

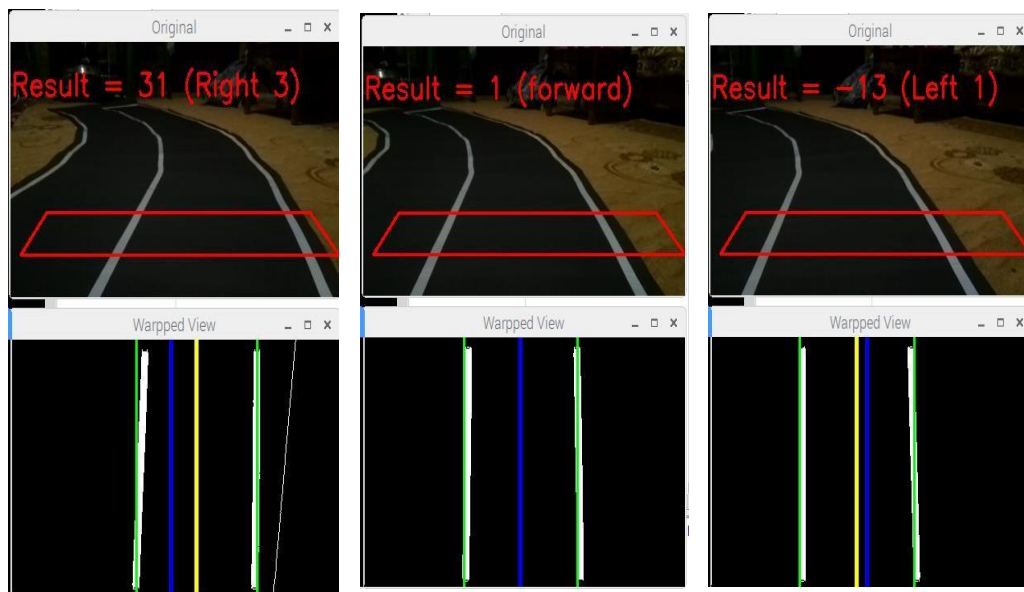
Attached DVD.



**Figure 4.6: Center Line Creating**



**Figure 4.7: Center Difference Calculating**



**Figure 4.8: Different states for road lanes analysis**

#### 4.4.2 U-Turn

The U-Turn system follows the same method for all shapes and objects. It depends on a cascade training file. This file is created by using a Machine learning training software (Cascade Training GUI).

Code:



```

void Uturn_detection()
{
    if(!Uturn_Cascade.load("/home/pi/Downloads/uturn+.xml"))
    {
        printf("Unable to open stop cascade file");
    }

    RoI_Uturn = frame_Uturn(Rect(0,0,300,240));
    cvtColor(RoI_Uturn, gray_Uturn, COLOR_RGB2GRAY);
    equalizeHist(gray_Uturn, gray_Uturn);
    Uturn_Cascade.detectMultiScale(gray_Uturn, Uturn);
    for(int i=0; i<Uturn.size(); i++)
    {
        Point P1(Uturn[i].x, Uturn[i].y);
        Point P2(Uturn[i].x + Uturn[i].width, Uturn[i].y + Uturn[i].height);
        rectangle(RoI_Uturn, P1, P2, Scalar(0, 0, 255), 2);
        putText(RoI_Uturn, "U-turn", P1, FONT_HERSHEY_PLAIN, 1, Scalar(0, 0, 255, 255), 2);
        dist_Uturn = (-0.866)*(P2.x-P1.x) + 74.2;

        ss.str(" ");
        ss.clear();
        ss<<"D = "<<dist_Uturn<<" CM ";
        putText(RoI_Uturn, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);
    } }

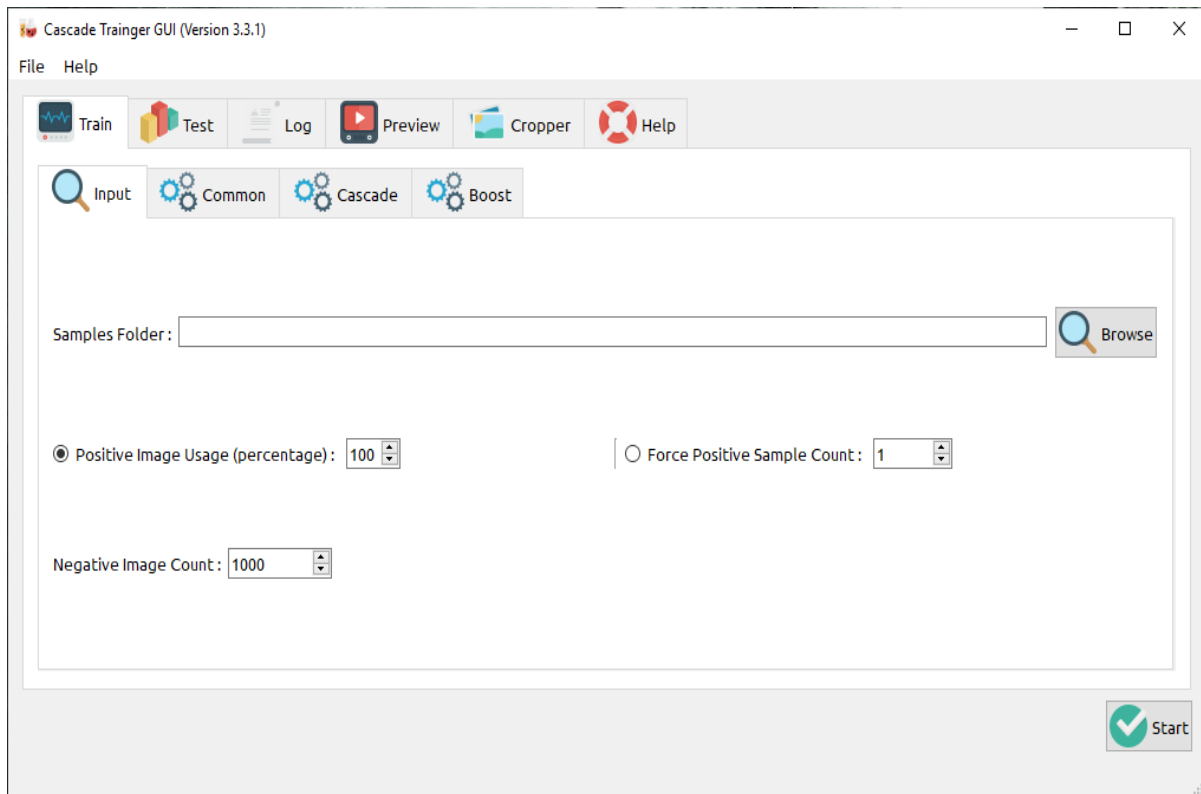
```



**Figure 4.9: U-turn sign**

### 4.4.3 Signs, Objects and Traffic Lights Recognition

The object recognition system follows the same method for all shapes and objects. It depends on a cascade training file. This file is created by using a Machine learning training software (Cascade Training GUI), as shown in Figure 4.10.






**Figure 4.10: Cascade Training GUI**

For each object, the following steps are used:

#### *4.4.3.1 Positive samples capturing*

For each object, 40 images at least are captured from a different angle and cropped (See Table 4.2).






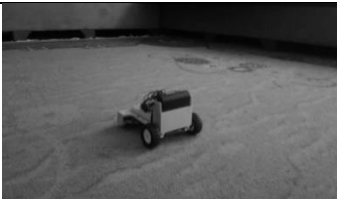
**Table 4.2: Positive Sample For Different Objects**

|  |  |  |
|--|--|--|
|  <p>Car Positive Sample</p> |  <p>Stop Sign Positive Sample</p> |  <p>Traffic Light Positive Sample</p> |
|--|--|--|

#### 4.4.3.2 Negative samples capturing

Three hundred images at least are captured for everything except the aimed object such as the road from different angles, other objects, and similar but different objects (See Table 4.3).

**Table 4.3: Negative Samples**

|   |   |   |
|---|---|---|
|  |  |  |
|  |  |  |

#### 4.4.3.3 Training

The “Cascade Training GUI” software (Attached to the DVD) is used to create a training cascade file as an XML file.

#### 4.4.3.4 Detection

The generated XML file for each object is placed in the software folder and loaded to the system program using the cascade functions in the OpenCV library (See Figure 4.11).



**Figure 4.11: Object Detecting using an XML Cascade file**

#### 4.4.3.5 Measure the Object distance

After detecting the object, the system draws a rectangle around it. The rectangle is used to measure the distance as follows.

1. Put the object at a distance where the system can detect it (See Figure 4.13).
2. Consider the rectangle horizontal side width in PXs (See Figure 4.14).
3. Measure the distance between the car and the object (Figure 4.15).
4. Repeat the previous steps for two times ( Table 4.4).

**Table 4.4: Measured Values for object detection**

| Distance          | Side Width Px  |
|-------------------|----------------|
| <i>Distance 1</i> | <i>Width 1</i> |
| <i>Distance 2</i> | <i>Width 2</i> |
| <i>Distance 3</i> | <i>Width 3</i> |

5. The distance equation can be created using a linear equation

$$distance = m(sideWidthPx) + c$$

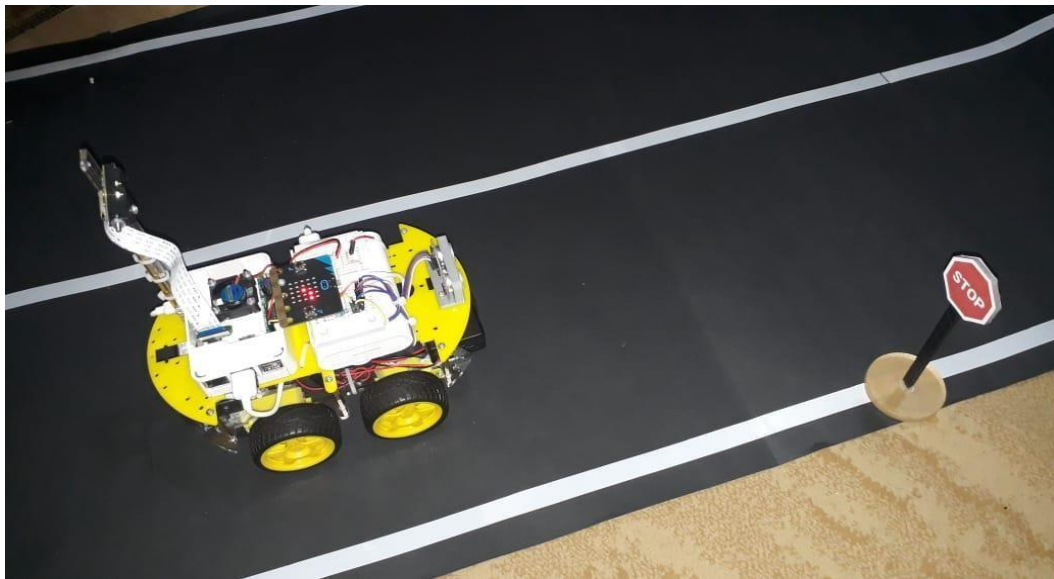
The m represents the slope and c is the intersection with the y-axis.

6. m & c can be calculated using the result in filled ( Table 4.4) using two-equation and two unknowns solving methods.

$$distance1 = m(width1) + c$$

$$distance2 = m(width2) + c$$

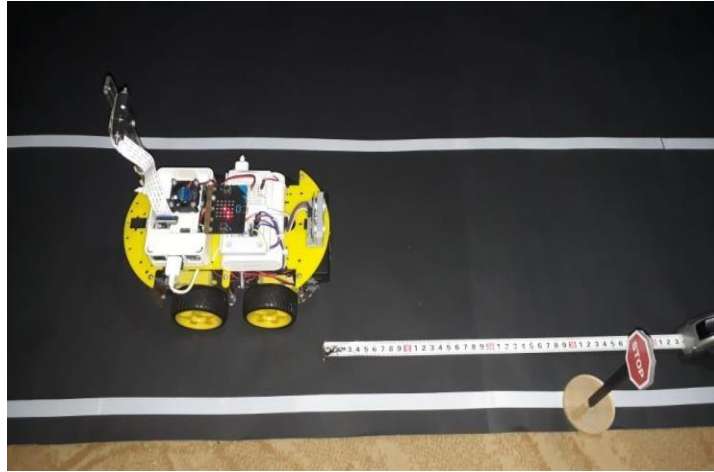
The third inputs are used to verify the results.



**Figure 4.12: Car in front of the stop sign**



**Figure 4.13: Stop sign side width in px**

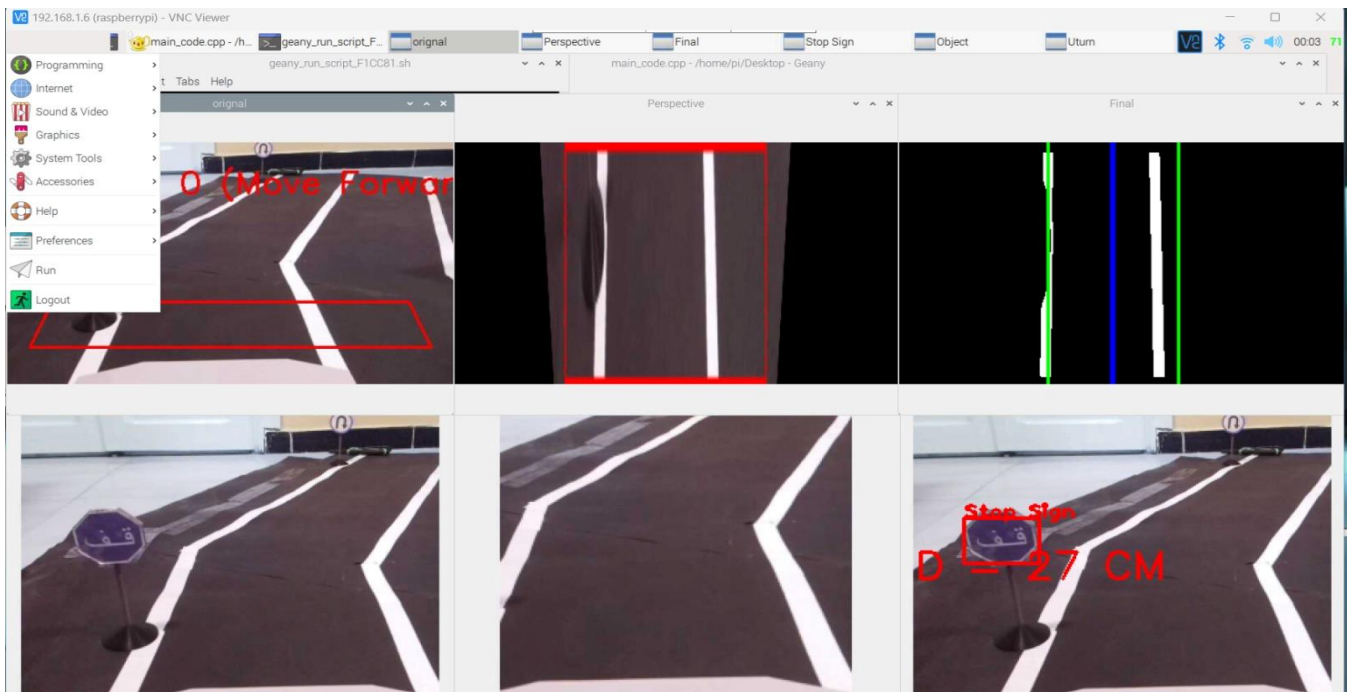


**Figure 4.14: Measuring the distance between the car and the sign**

## 4.5 Test Cases and Acceptance Criteria

### 4.5.1 Test 1 (Lane Detecting and Tracking)

In this test, the car should follow the road curve and stay in the middle. After making some calibration for the speeds and the threshold image values, the car was able to follow the road with its sharp curves (See Figure 4.15).



**Figure 4.15: Road lanes Tracking Testing Result**



### 4.5.2 Test 2 (U Turn)

After detecting the U-turn sign at a distance less than 30 cm, the car should stop for two seconds and then opposite its direction (return).







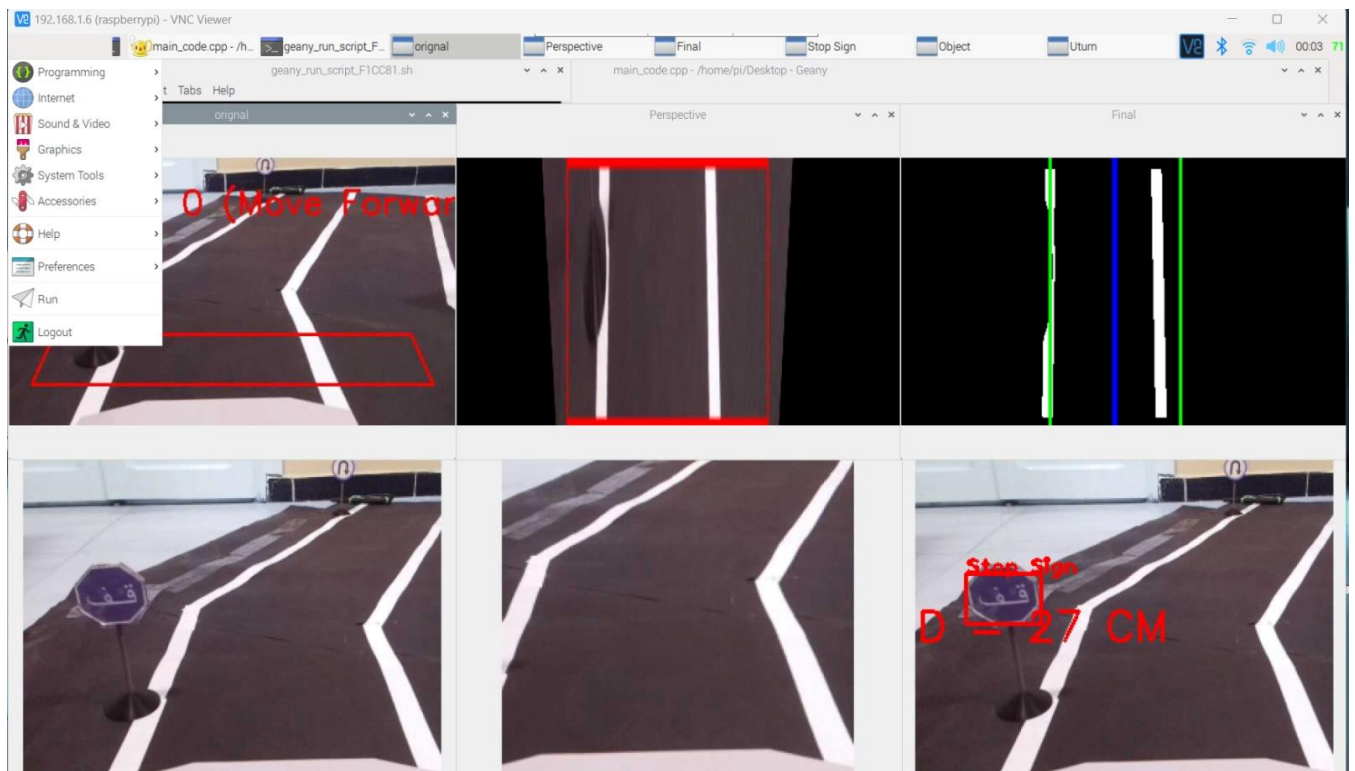
**Figure 4.16: U-Turn Result**

### 4.5.3 Test 3 (Stop Sign Detection and Function)

After detecting the stop sign at a distance less than 30 cm, the car should stop for two seconds and then move again. Figure 4.17 shows the stop sign detection window.

Code:

```
void Stop_detection()
{
    if(!Stop_Cascade.load("/home/pi/Downloads/New_stop_Home100.xml"))
    {
        printf("Unable to open stop cascade file");
    }
    RoI_Stop = frame_Stop(Rect(0,0,300,240));
    cvtColor(RoI_Stop, gray_Stop, COLOR_RGB2GRAY);
    equalizeHist(gray_Stop, gray_Stop);
    Stop_Cascade.detectMultiScale(gray_Stop, Stop);
    for(int i=0; i<Stop.size(); i++)
    {
        Point P1(Stop[i].x, Stop[i].y);
        Point P2(Stop[i].x + Stop[i].width, Stop[i].y + Stop[i].height);
        rectangle(RoI_Stop, P1, P2, Scalar(0, 0, 255), 2);
        putText(RoI_Stop, "Stop Sign", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
        dist_Stop = (-0.866)*(P2.x-P1.x) + 74.2;
        ss.str(" ");
        ss.clear();
        ss<<"D = "<<dist_Stop<<" CM ";
        putText(RoI_Stop, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);} }
```



**Figure 4.17: Stop Sign Detection Result**

#### 4.5.4 Test 4 (Object Avoidance)

When detecting a car, the self-driving car should pass it without any collision. After calibrating the stop distance between the car and the object and adjust the rotation degrees. The self-driving car was able to pass the car object without any collision (See Figure 4.18)

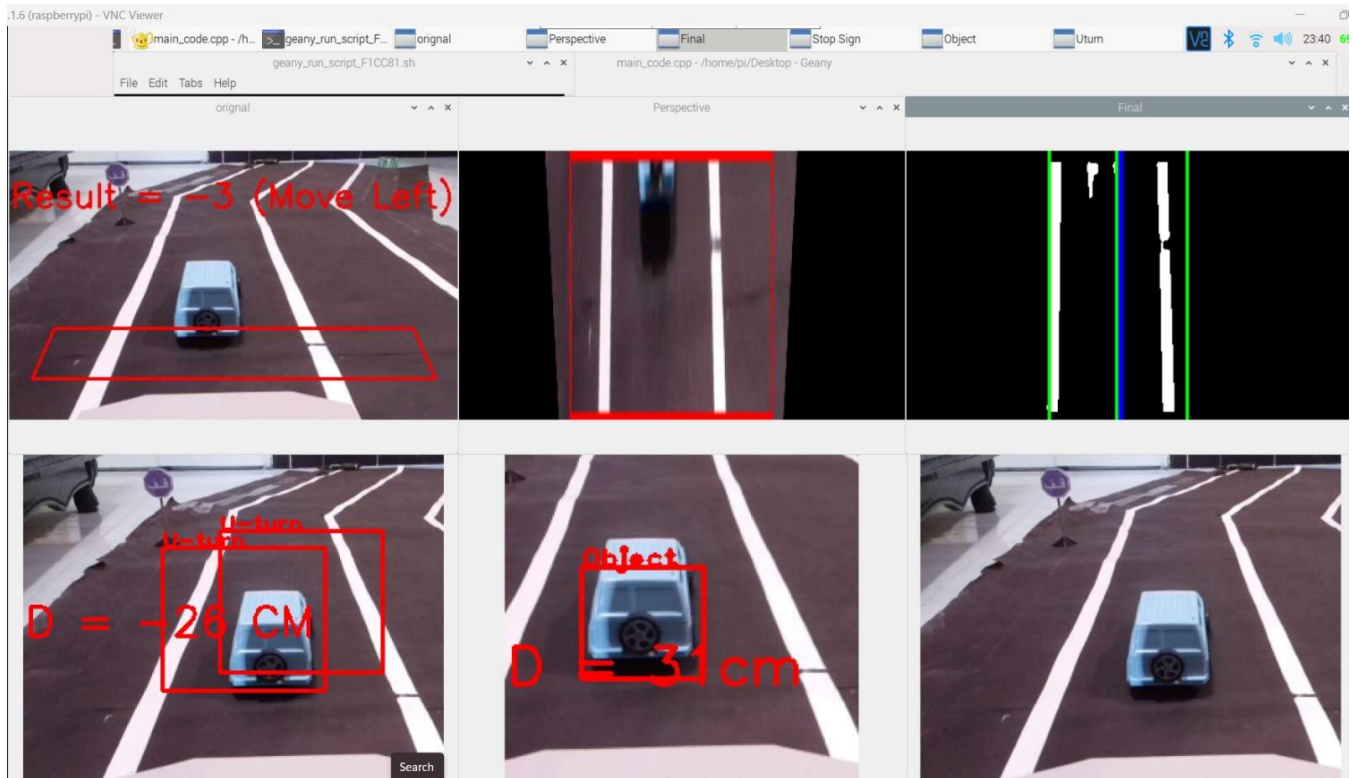
Code:

```
void Object_detection()
{
    if(!Object_Cascade.load("/home/pi/Downloads/geshcar20more.xml"))
    {
        printf("Unable to open Object cascade file");
    }
    RoI_Object = frame_Object(Rect(100,50,200,190));
    cvtColor(RoI_Object, gray_Object, COLOR_RGB2GRAY);
    equalizeHist(gray_Object, gray_Object);
    Object_Cascade.detectMultiScale(gray_Object, Object);
    for(int i=0; i<Object.size(); i++)
    {
        Point P1(Object[i].x, Object[i].y);
        Point P2(Object[i].x + Object[i].width, Object[i].y + Object[i].height);
        rectangle(RoI_Object, P1, P2, Scalar(0, 0, 255), 2);
        putText(RoI_Object, "Object", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
        dist_Object = (-0.411)*(P2.x-P1.x) + 59.411;
        ss.str(" ");
        ss.clear();
        ss<<"D = "<< dist_Object<<"cm";
```

```

    putText(RoI_Object, ss.str(), Point2f(1,130), 0,1, Scalar(0,0,255), 2);
}

```

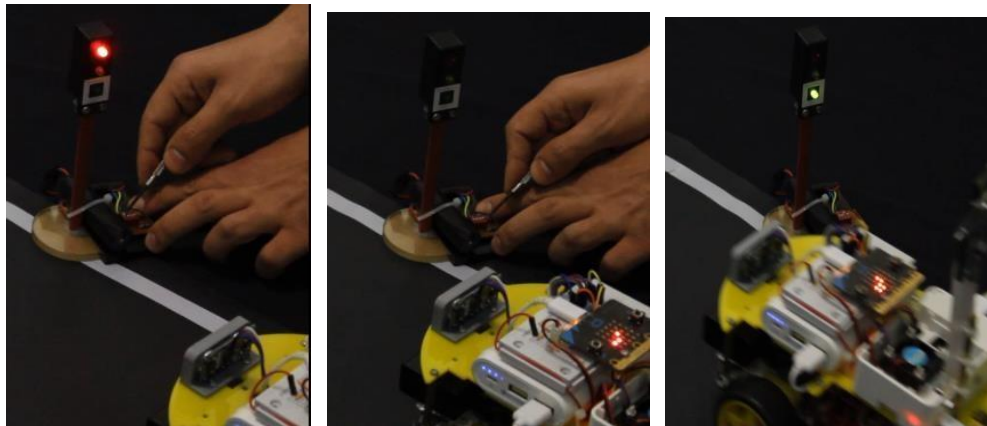


**Figure 4.18: Car Avoiding Result**

#### 4.5.5 Test 5 (Traffic Light Detecting)

The car should distinguish the red and green lights and act according to them.

After testing, the car stopped successfully when the red light is ON or when the all lights are turned OFF, and it moved when the green light is turned ON (See Figure 4.19)



**Figure 4.19: Traffic Light Testing**

#### 4.6 Conclusion

Using the computer system. The system was able to track the road, detect the signs, and avoid obstacles. The power of the computer vision combined with the c++ programming language allowed us to build a completely self-driving car prototype.

## **5. CONCLUSION AND FUTURE WORK**

### **5.1 Conclusion**

After working in the field of smart cars, robot, computer vision, sensors, and machine learning for months and being in direct contact with these concepts, we discovered the following:

Firstly, from our point of view, we can say that computer vision is an excellent application for the field of autonomous robots. With current tools (rising of computing power, big cameras resolution...) applying computer vision and machine learning in real-time does not mean an obstacle when used on autonomous systems.

Secondly, working with computer vision needs a controlled environment. It can become inconvenient when working in real-world situations. The light varies its position making shadows behind the obstacles and objects, and they can be complicated to identify. There are a lot of uncontrolled physical conditions that make the probability of failing in the project very high. This project has been done in controlled conditions (white background, fixed light, black objects, and markers). I can say that the result is successful, and the system is functional in all cases. Thanks to the appearance of new technologies and platforms such as fast wireless network technology and the OpenCV platform, which helped us to achieve the project goal.

To conclude, we can say that this technology is getting closer to the customer every day, and the applications where the autonomous robot can be applied are increasing in all the fields.

## 5.2 Future Work

This system may be improved by adding some sensors such as Lidar, which make the driving even more safer.

The system may use wheels' encoders to control the car speed and to increase the reliability of the car movement.

Further modifications may be added to the user interface to make it easier, responsive, and user-friendly.

The car may achieve other tasks such as exploring people, animals, and objects and doing a search for custom objects. Besides, the system can be switched between manual and autonomous systems for those who love manual driving and drifting!

The current system can only deal with objects it knows (After training). The system in the future can be linked to the internet and use the online platform to know more objects and get more training.

Some sensors may be added to control the system in different environments. For example, a light intensity sensor may be used to control the camera threshold.

In future work, more situations can be considered to deal with, especially in fog environment, storm, rain and cloudy weather.

All these concepts should be applied to the future self-driving car to make it safer, faster, more reliable, and smarter!



## **APPENDIX A:**

### **IMPLEMENTATION DETAILS**

#### **A.1 Raspberry pi setup**

To install and setup of the raspberry pi operating system, follow the steps listed in the following link <https://projects.raspberrypi.org/en/projects/noobs-install>.

#### **A.2 Connecting Raspberry Pi to Personal Computer**

Connecting the raspberry pi to the personal computer can be using an Ethernet cable or using a wifi connection.

For more details see <https://maker.pro/raspberry-pi/tutorial/how-to-connect-a-raspberry-pi-to-a-laptop-display> and <https://www.electronicwings.com/raspberry-pi/access-raspberry-pi-on-laptop-using-wi-fi>

#### **A.3 OpenCV and Wiring Pi setup**

The commands and files used to install OpenCV and Raspicam libraries are attached to the DVD.

The wiring Pi library can be installed using the steps in the following links. <http://wiringpi.com/download-and-install/>

#### **A.4 Motor Driver H-Bridge L298N**

For direction control, the current flow through the motor should be reversed, and the most common method of doing that is by using an H-Bridge (Figure 5.1). An H-Bridge circuit contains four switching elements, transistors or MOSFETs, with the motor at the center forming an H-like configuration. By activating two particular switches at the same time, we can change the direction of the current flow, thus change the rotation direction of the motor.

The L298N is a dual H-Bridge motor driver that allows speed and direction control of two DC motors at the same time. The module can drive DC motors that have voltages between 5 and 35V, with peak current up to 2A.

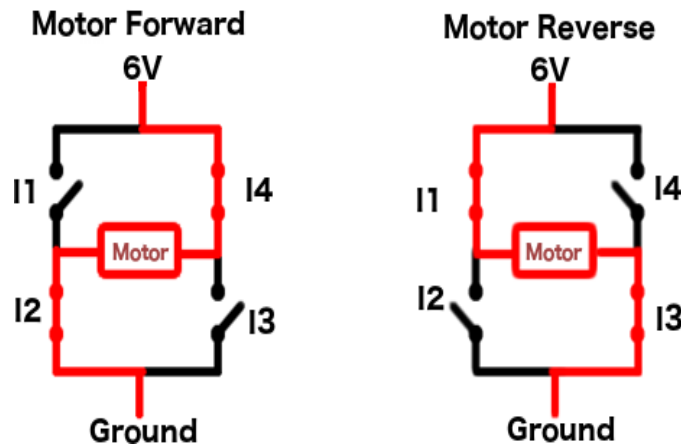


Figure 5.1: H-Bridge Architecture [5]

The module has two screw terminal blocks for the motor A and B and another screw terminal block for the Ground pin, the VCC for motor and a 5V pin which can either be an input or output. PWM1 and PWM2 control the motor one. PWM3 and PWM4 control the motor2. The difference of the PWM signal between the two pins controls the direction, speed, and the stop of the motor. The following circuit (Figure 5.2) shows the connection of the H- Bridge with the microcontroller.

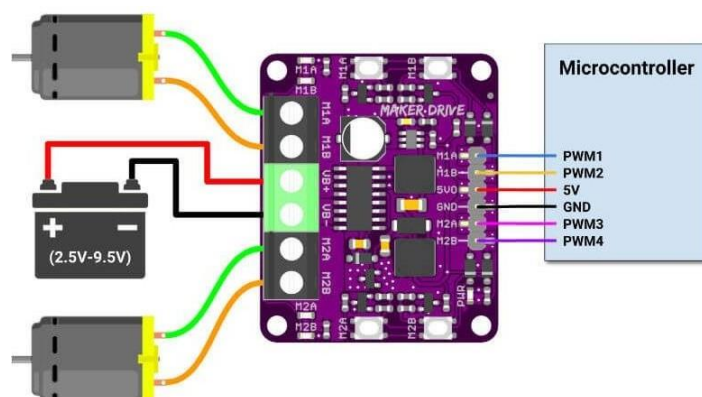
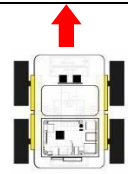
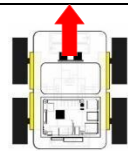
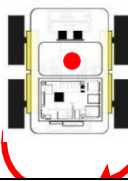
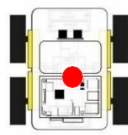
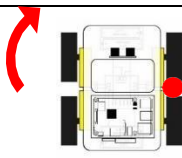
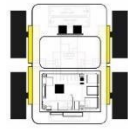


Figure 5.2: H-Bridge Connections [16]

## A.5 Car movement methods

In order to make the car moves in all directions at different speeds, the system controls the power in each side (left and right) as shown in the following (Table 5.1: Motor Commands)

**Table 5.1: Motor Commands**

| Motor Commands                                      | Resulting Movement  |
|---|---|
| Left Power = 100%<br>Right Power = 100%             |  A top-down diagram of a car with a red arrow pointing straight up from its front, indicating forward movement.                      |
| Left Power = 50%<br>Right Power = 50%               |  A top-down diagram of a car with a red arrow pointing straight up from its front, indicating forward movement at a reduced speed.  |
| Left Power = 0 %<br>Right Power = 0 %               |  A top-down diagram of a car with a red dot in the center and two curved red arrows forming a circle around it, indicating a spin. |
| Left Power = 100 %<br>Right Power = 100 % (Reverse) |  A top-down diagram of a car with a red dot in the center, indicating backward movement.   |
| Left Power = 100 %<br>Right Power = 0 %             |  A top-down diagram of a car with a red dot on the right side and a curved red arrow pointing to the left, indicating a left turn.  |
| Left Power = 100 %<br>Right Power = 50 %            |  A top-down diagram of a car with a red dot to its right, indicating a combination of forward movement and a right turn.           |

## APPENDIX B:

### USER MANUAL

To operate the system, follow the next steps:

#### **B.1 Turning ON the system and connect the raspberry pi with the laptop**

1- Make sure to download and install the following programs on the personal

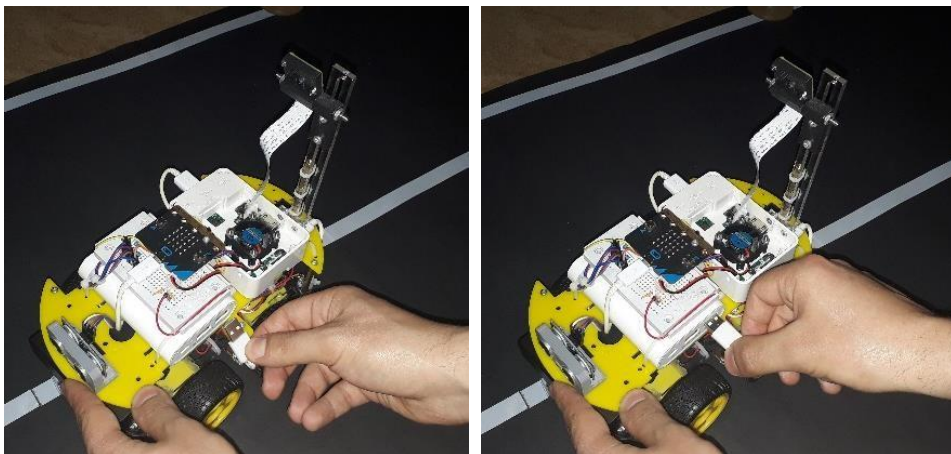
laptop <https://mobaxterm.mobatek.net/download-home-edition.html>

[https://www.advanced-ip-scanner.com/download/Advanced\\_IP\\_Scanner\\_2.5.3850.exe](https://www.advanced-ip-scanner.com/download/Advanced_IP_Scanner_2.5.3850.exe)

1. Make a new hotspot on your smartphone as follows:

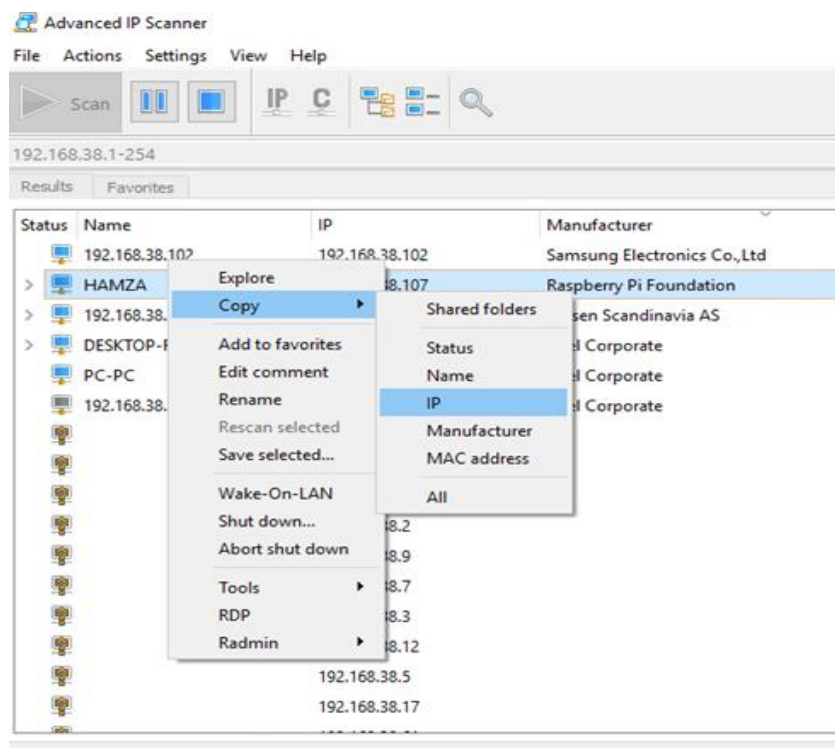
- SSID Name: “ ^ \_ \* “
- Password: “ 4444hhhh “

2. Turn On the raspberry pi by connecting the power bank USB cable as shown in

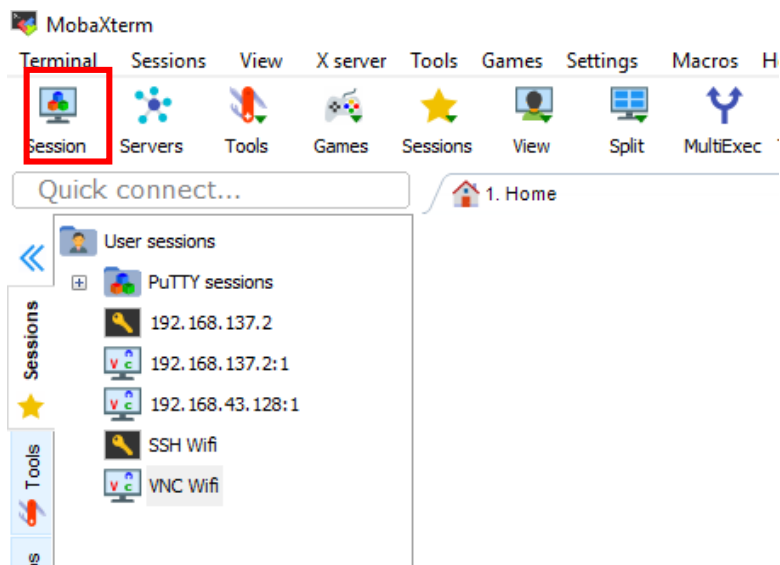


**Figure 5.3: Powering the Rpi ON**

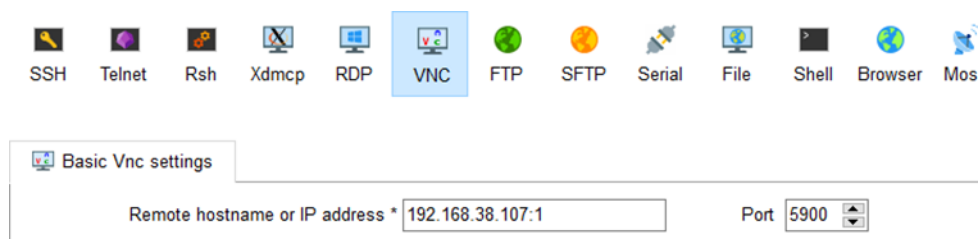
3. Connect the Laptop to the “ ^\_ \* “ Network.
4. Open the “Advance IP Scanner” and press “IP” then “scan” buttons.
5. Copy the “HAMZA” IP (Figure 5.4)
6. Open “MobaXterm” program and click on Session Button ( Figure 5.5)
7. Choose the VNC tab and paste the IP followed by “ :1 “ ( Figure 5.6 ) and click okay
8. The system will ask for a password. The VNC password is “ 12345678”Figure 5.7 shows the Desktop of the Raspbian system.



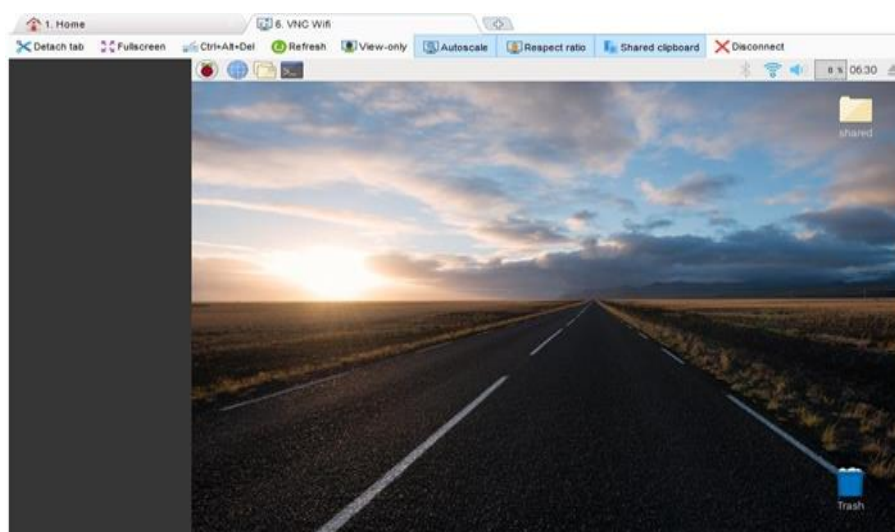
**Figure 5.4: IP Scanner**



**Figure 5.5: MobaExterm Session**



**Figure 5.6: VNC Session**



**Figure 5.7: Raspbian Desktop**

## B.2 Operating the system program

1. Open the folder location on the Desktop “ shared/Smart-Car-Program “ or open the location “ /home/pi/Desktop/shared/Smart-Car-Program”
2. Double click the output file “sdc” and click “Execute in Terminal”
3. If the system is operated well, the system windows appear, as shown in Figure5.8.

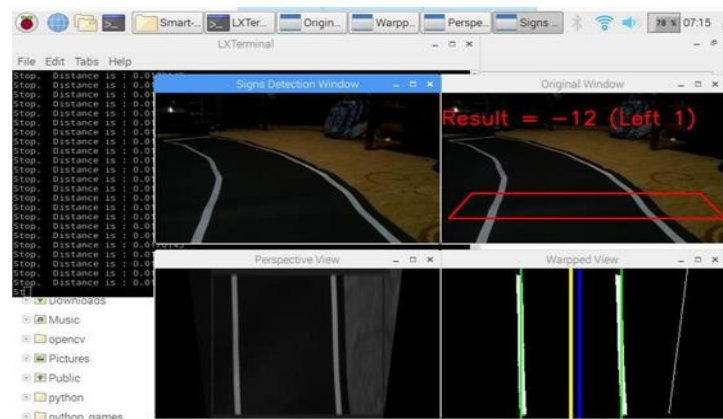


Figure 5.8: System windows

## B.1 Operating the Motors

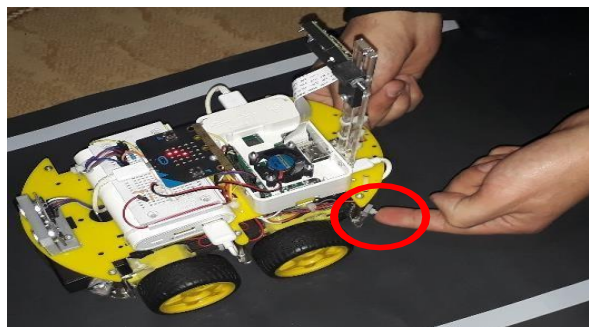


Figure 5.9: Turning the Motors

### B.1 Close the system

To close the system, turn off the driver of the motor using the switch (See Figure 5.9) and close the terminal window

## APPENDIX C:

### DEPLOYMENT AND CONFIGURATION MANUAL

#### C.1 Hardware Connection

Table 5.2 shows the connection pins for the motor driver.

**Table 5.2: Hardware Wiring**

| Pin Description | Physical Number | BCM Number | Wiring Pi Number | Wire Color |
|-----------------|-----------------|------------|------------------|------------|
| IN1             | 35              | 19         | 24               | White      |
| IN2             | 37              | 26         | 25               | Yellow     |
| IN3             | 31              | 6          | 22               | Black      |
| IN4             | 33              | 13         | 23               | Red        |
| Trig            | 11              | 17         | 0                | Violet     |
| Echo            | 13              | 27         | 2                | Blue       |
| GND             | GND             | GND        | GND              | Yellow     |



## REFERENCES

- [1] J. Widom, "Stanford Engineering," March 2017. [Online]. Available:  
  
<https://engineering.stanford.edu>.
- [2] "How Self-Driving Cars Work," [Online]. Available:  
  
<http://www.rapidrepairautocenter.com/how-self-driving-cars-work/>.
- [3] B. Gringer, "TitleMax," [Online]. Available:  
  
<https://www.titlemax.com/resources/history-of-the-autonomous-car/>.
- [4] "TrueCAR Adviser," 05 March 2018. [Online]. Available:  
  
<https://www.truecar.com/blog/5-levels-autonomous-vehicles/>.
- [5] Dejan, "Arduino DC Motor Control Tutorial – L298N | PWM | H-Bridge," 2019. [Online].  
  
Available:  
  
<https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/>.
- [6] T. Cox, Raspberry Pi Cookbook for python programmers, Birmingham B3  
  
2PB, UK.: Packt Publishing Ltd., 2014.
- [7] "Better Understanding Of Batteries – Li-Ion Vs. Li-Po," [Online].  
  
Available: <https://www.reliancedigital.in/solutionbox/better-understanding-of-batteries-li-ion-vs-li-po/>.

- [8] Osoyoo, "Micro bit lesson — Using the Ultrasonic Module," 2018. [Online]. Available: <https://osoyoo.com/zh/2018/09/18/micro-bit-lesson-using-the-ultrasonic-module/>.
- [9] "What is a micro:bit?," 25 May 2018. [Online]. Available: <https://support.microbit.org/support/solutions/articles/19000013983-what-is-a-micro-bit->.
- [10] "What is Raspbian?," 2013. [Online]. Available: <https://raspberrypi.stackexchange.com/questions/1217/what-is-raspbian>.
- [11] R. Galgali, "What is openCV?," 27 February 2016. [Online]. Available: [https://www.quora.com/What-is-openCV?awc=15748\\_1578431320\\_31223a707d866ecc0b91e5548aec9483&uiiv=6&txvt=8&source=awin&medium=ad&campaign=uad\\_mkt\\_en\\_acq\\_us\\_awin&set=awin&pub\\_id=101248](https://www.quora.com/What-is-openCV?awc=15748_1578431320_31223a707d866ecc0b91e5548aec9483&uiiv=6&txvt=8&source=awin&medium=ad&campaign=uad_mkt_en_acq_us_awin&set=awin&pub_id=101248).
- [12] "About WiringPi," [Online]. Available: <http://wiringpi.com/>.
- [13] N. Tollervey, 24 November 2019. [Online]. Available: <https://github.com/mu-editor/mu>.
- [14] "Python Programming Language," [Online]. Available: [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)).

- [15] A. Ahmadi, "Cascade Training GUI," [Online]. Available: <http://amin-ahmadi.com/cascade-trainer-gui/>.
- [16] "Simple H-Bridge DC Motor Driver," [Online]. Available: <https://robotbits.co.uk/product/cytron-maker-drive-simple-h-bridge-dc-motor-driver-2-5-9-5volt-1-5amp-max/>.

