**Ain Shams University**
**Faculty of Computer & Information Sciences**
**Computer Systems Department**

# Self-Driving Car

## By

Ahmed Samy CSYS
Abd Al Rahman Ahmed CSYS
Karim El-Shazly CSYS
Ahmed Ashraf CSYS
Mohamed Hossam CSYS

## Under Supervision of

**Dr. Randa Mahmoud**
Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

**T.A. Fatma El-Wasify**
Computer Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

**June 2023**

# Acknowledgement

All praise and thanks to ALLAH, who provided me with the ability to complete this work. I hope to accept this work from me. I am grateful of *my parents* and *my family* who are always providing help and support throughout the whole years of study. I hope I can give that back to them.

I also offer my sincerest gratitude to my supervisors, Prof. Dr. Eman Shaaban, Dr. Randa Mohamed and T.A. Fatma El-Wasify who have supported me throughout my thesis with their patience, knowledge and experience. Finally, I would like to thank my friends and all people who gave me support and encouragement.

# Abstract

The advancement of autonomous vehicles has gained significant attention in recent years, with self-driving cars emerging as a prominent application. One of the crucial components of self-driving cars is the ability to accurately detect lanes and objects in real-time. This documentation presents an in-depth study of lane detection techniques using image processing and object detection using machine learning for self-driving cars.

Lane detection plays a vital role in autonomous driving, as it provides valuable information about road boundaries and enables the vehicle to maintain its position within the lane. Various image processing techniques, such as create region of interest, apply perspective transformation, threshold and canny edge detection, are explored to extract lane markings from the input video stream. These techniques are applied sequentially to obtain robust and accurate lane detection results.

In addition to lane detection, the identification and tracking of objects on the road are crucial for safe and efficient navigation. Machine learning algorithms, particularly object detection models like haar and YOLO, are employed to detect and classify objects such as pedestrians, vehicles, and traffic signs. These models are trained on annotated datasets, enabling them to recognize objects with high precision and recall.

The proposed lane detection and object detection techniques are implemented and evaluated on real-world driving scenarios using a self-driving car platform. Experimental results demonstrate the effectiveness of the methods in accurately detecting lanes and objects in real-time. The system achieves reliable lane tracking and provides timely warnings and interventions when encountering obstacles or potential hazards on the road.

# Table of Contents


**Acknowledgements** 1

**Abstract** 2

**List of Figures** 4

**Chapter 1: Introduction** 5

**1. Introduction** 6

**1.1 Motivation** 6

**1.2 Problem Definition** 7

**1.3 Objectives** 7

**1.4 Time plan** 8

**1.5 Documentation Outline** 9

**Chapter 2: Literature Review** 10

**2.1 Introduction** 11

**2.2 Theoretical Background** 11

**Chapter 3: System Architecture and Methods** 17

**3.1 System Architecture** 18

**3.2 Methods and procedures used** 19

**Chapter 4: System Implementation and Results** 27

**4.1 Dataset** 28

**4.2 Description of Software Tools Used** 28

**4.3 Setup Configuration (hardware)** 33

**4.4 Experimental and Results** 42

**Chapter 5: Run the Application** 48

**Chapter 6: Conclusion and Future Work** 55

**6.1 Conclusion** 56

**6.2 Future Work** 56

**References** 57

# List of Figures


Figure 1-1: Accidents reasons 7
Figure 1-4: Time plan (Gantt chart) 8
Figure 2-2-2-3: Human vision vs. Computer vision 13
Figure 2-2-2-4: Deep Learning Approach 14
Figure 2-2-2-3: Convolutional Network Approach 15
Figure 3-1 System architecture 18
Figure 3-2-1: Thresholding 20
Figure 3-2-1: Wrapping 21
Figure 3-2-1: Thresholding comparison 22
Figure 3-2-1: Canny edge detection 22
Figure 3-2-1: Image dilation 23
Figure 3-2-1: Siding window output 24
Figure 3-2-1: Approach output 24
Figure 3-2-2: Positive sample vs. Negative sample 26
Figure 4-3-1: Raspberry Pi 4 Board Layout 35
Figure 4-3-1: Raspberry Pi 4 GPIO Pinout 36
Figure 4-3-1: Raspberry Pi 4 GPIO headers 36
Figure 4-3-1: Power pins on Raspberry Pi 4 37
Figure 4-3-2: Arduino uno 38
Figure 4-3-3: Camera Module Connected to Raspberry Pi 39
Figure 4-3-3: Raspberry Pi Camera 39
Figure 4-3-4: L298n Motor driver module 40
Figure 4-3-5: 4 wheels 2 layers robot smart car chassis kit 41
Figure 4-3-6: Charger power bank 41
Figure 4-4-2: Lane Detection 45
Figure 4-4-3: Stop sign Detection 47
Figure 5-1: Run the Application 49
Figure 5-2: Go Forward(Car) 50
Figure 5-2: Go Forward(IDE) 50
Figure 5-3: Go Right(Car) 51
Figure 5-3: Go Right(IDE) 51
Figure 5-4: Go Left(Car) 52
Figure 5-4: Go Left(IDE) 52
Figure 5-5: The stop sign is far from the car 53
Figure 5-5: The stop sign is far from the car (IDE) 53
Figure 5-5: The stop sign near the car 54
Figure 5-5: The stop sign is far from the car (IDE) 54

# Chapter 1

# Introduction

# 1. Introduction

There are some problems that driving may cause, such as human error which may be caused by distracted driving or impaired driving that leads to accidents on the road. Traffic congestion, which is a major issue in many urban areas, resulting in wasted time, increased fuel consumption, and negative environmental impacts.

Self-driving cars can detect and respond to potential hazards more quickly and effectively than human drivers, reducing the number of accidents caused by human error with advanced sensors and machine learning algorithms.

Self-driving cars can also address accessibility issues by providing greater mobility to individuals with disabilities or those who are unable to drive due to age or health issues. They can enable greater independence and mobility for these individuals, improving their quality of life and reducing reliance on others for transportation.

## 1.1 Motivation

Our motivation behind the project is it aims to improve people's lives and save time. The project has several goals, including reducing the number of accidents on the road, eliminating traffic congestion, increasing highway capacity, enhancing human productivity, eliminating the hassle of hunting for parking spaces, improving mobility for children, the elderly, and the disabled, and raising speed limits. By reducing the number of accidents, the project aims to make the roads safer for everyone. Traffic congestion can be a major source of frustration and wasted time, so the project also aims to reduce or eliminate it. By increasing highway capacity, the project can make it easier for people to travel to and from their destinations quickly and efficiently. Enhanced human productivity is also a goal, as the project can help people get where they need to go more quickly, freeing up time for other activities. One major benefit of the project is that it eliminates the hassle of hunting for parking spaces. This can be a time-consuming and frustrating process, particularly in busy urban areas. By improving mobility for children, the elderly, and the disabled, the project can make it easier for these groups to get around and access the resources they need. Finally, by raising speed limits, the project can make travel faster and more efficient for everyone. Overall, these motives demonstrate the potential impact of the project on people's lives and the significant benefits it can bring.

## 1.2 Problem definition

Driving cars may look easy but for a group of old people and those with special needs, it's difficult and maybe impossible. Even drivers sometimes may get detriment from losing time while driving. Sometimes there is more important tasks to do. In addition, there's a high probability of human mistake too. According to National Highway Traffic Safety Administration (NHTSA), most accident happens because of these reasons shown in figure below:

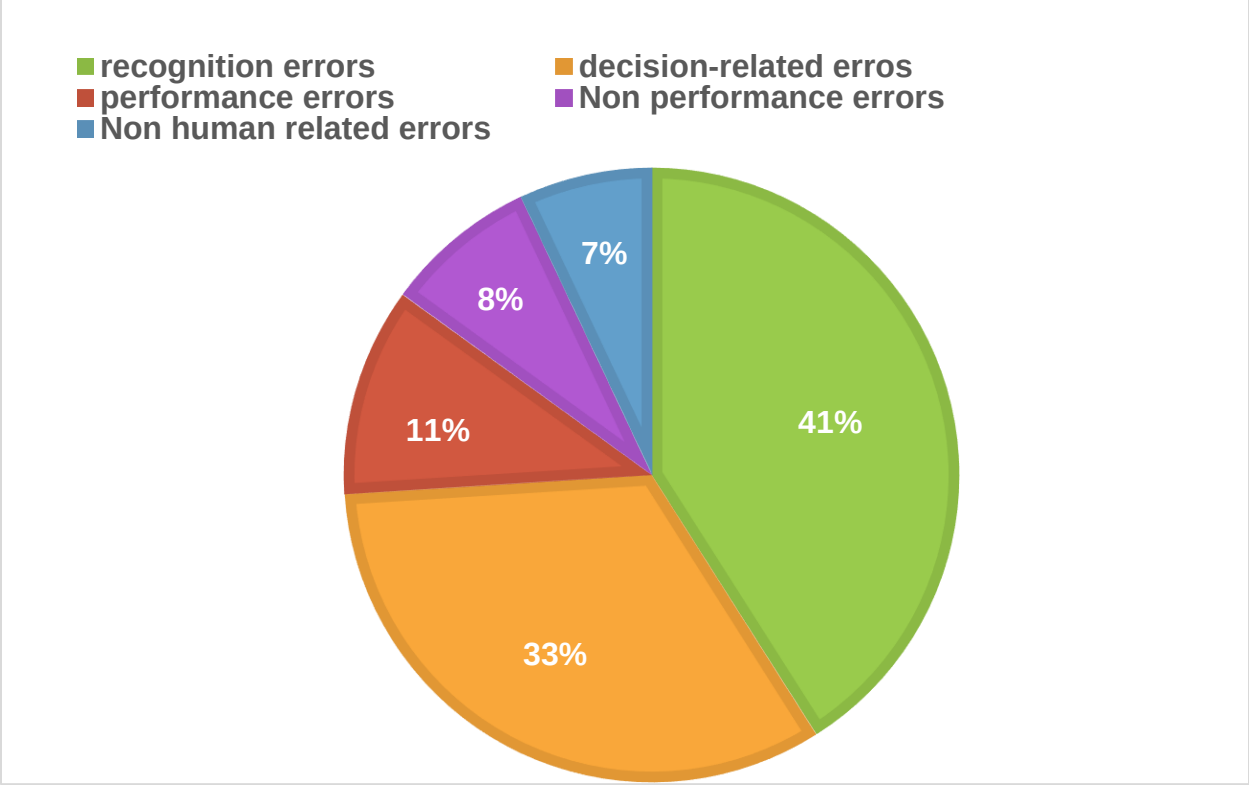


*Figure 1-2: Accidents' reasons according to NHTSA*

## 1.3 Objective

The objective of our project consists of 2 main functions:

I. Road detection, the car should drive in and can make turns itself which can be done using Image pre- processing, feature extraction and canny edge algorithm.

II. Objects detection on the road. For instance: Stop sign, Traffic light and obstacles. Using four methods positive samples, negative samples, neural network training and detection.

## 1.4 Time plan

From 2 October 2022 to 16 October 2022, we were collecting and looking for requirements we needed to create our project. Then from 17 October 2022 to 13 November 2022 we designed our project to have a full view of the project. After that we stepped up and started to implement the project hardware which done in the period from 20 November 2022 to 11 December 2022. We had a good break and then on 29 January 2023 to 26 February 2023 we started setup of the master device (Raspberry Pi) and solving the problem we got through. Meanwhile we started working on the image processing part on 7 February 2023 to 9 March 2023. On 20 February 2023 to 25 April we started one of the most difficult parts of the project, machine learning and object detection, which really took a lot of time of study and searching. All along with working with the past three parts we were testing and in enhancement of the project which were in the period from 5 February 2023 to 25 May 2023. As we create our project we started documenting from 1 December 2022 to 25 May 2023 as shown in the figure below.

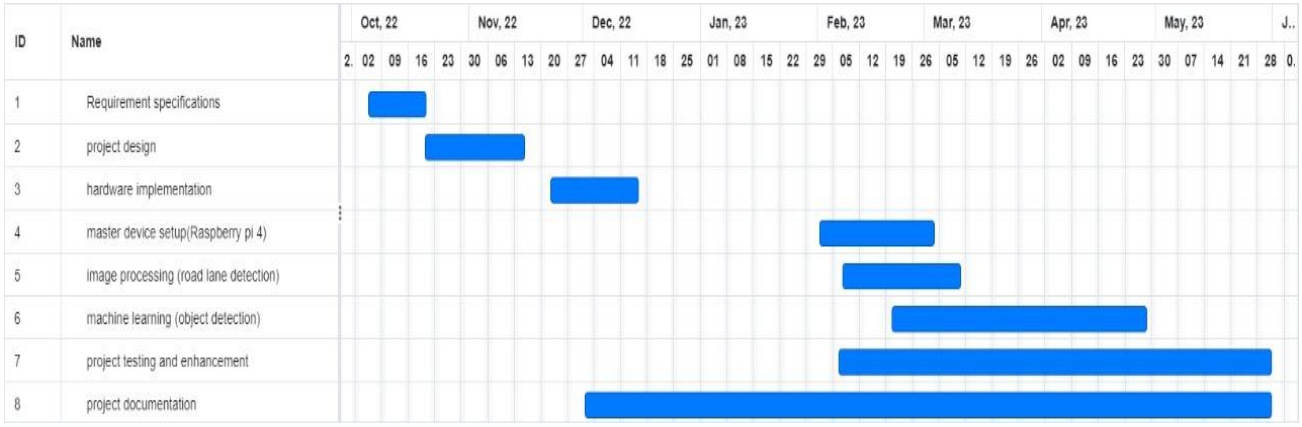


Figure 1-4: Time plan (Gantt chart)

## 1.5 Documentation outline

**Chapter 1:**

Introducing the project idea and motivation

**Chapter 2:**

Discuss the history of autonomy cars and a brief review about the topic and methods used in it.

**Chapter 3:**

Discuss System Architecture and Methods with description.

**Chapter 4:**

Discuss about the System Implementation and Results

**Chapter 5:**

will be about Run the Application

**Chapter 6:**

Discuss Conclusion and Future Work

# Chapter 2

# Literature Review

## 2.1 Introduction:

Autonomous cars are equipped with advanced technology that allows them to operate without human intervention. They use a combination of sensors, cameras, and software to navigate roads, avoid obstacles, and make decisions based on their environment. The technology behind autonomous cars is constantly evolving, with new advancements being made in the areas of machine learning, artificial intelligence, and computer vision.

## 2.2 Machine Leaning

Machine learning (ML) is the study of algorithms and mathematical models that computer systems use to progressively improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in the applications of email filtering, detection of network intruders, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics. Machine learning tasks are classified into several broad categories.

In supervised learning, the algorithm builds a mathematical model of a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images with and without that object (the input), and each image would have a label (the output) designating whether it contained the object. In special cases, the input may be only partially available, or restricted to special feedback. In unsupervised learning, the algorithm builds a mathematical model of a set of data which contains only inputs and no desired outputs.

Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as in feature learning. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

## 2.2 Artificial Intelligence (AI)

Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence. In computer science, AI, sometimes called machine intelligence, is intelligence demonstrated by machines, in contrast to the natural intelligence displayed by humans and other animals. Computer science defines AI research as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals. More in detail, Kaplan and Haenlein define AI as “a system‟s ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation”. Colloquially, the term "artificial intelligence" is applied when a machine mimics "cognitive" functions that humans associate with other human minds, such as "learning" and "problem solving". The scope of AI is disputed: as machines become increasingly capable, tasks considered as requiring "intelligence" are often removed from the definition, a phenomenon known as the AI effect, leading to the quip in Tesler's Theorem, "AI is whatever hasn't been done yet." For instance, optical character recognition is frequently excluded from "artificial intelligence", having become a routine technology. Modern machine capabilities generally classified as AI include successfully understanding human speech, competing at the highest level in strategic game systems (such as chess and Go),autonomously operating cars, and intelligent routing in content delivery networks and military simulations.

## 2.2 Computer vision

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand. Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image. Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a

minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

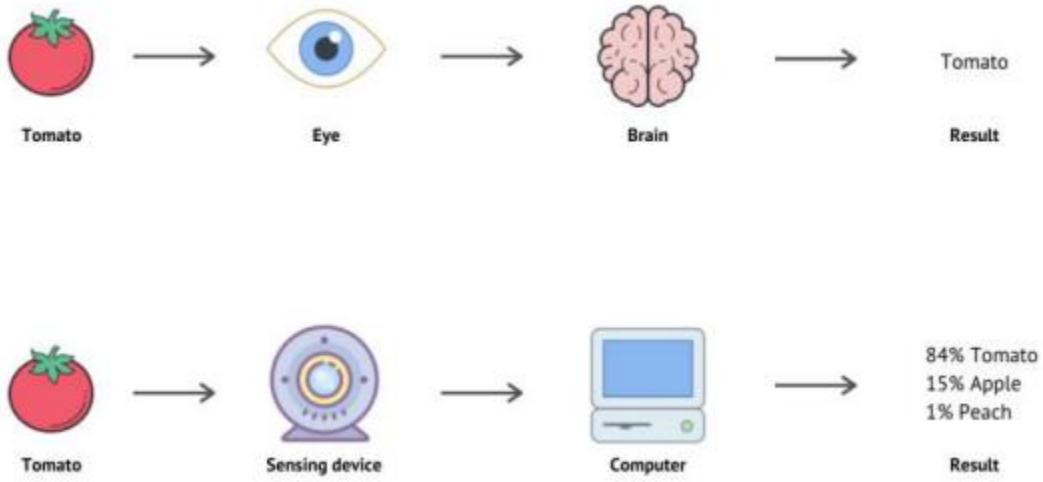


Figure 2-2-2-3: Human vision vs. Computer vision

## 2.2 Deep-Learning

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made. Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate. The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex.

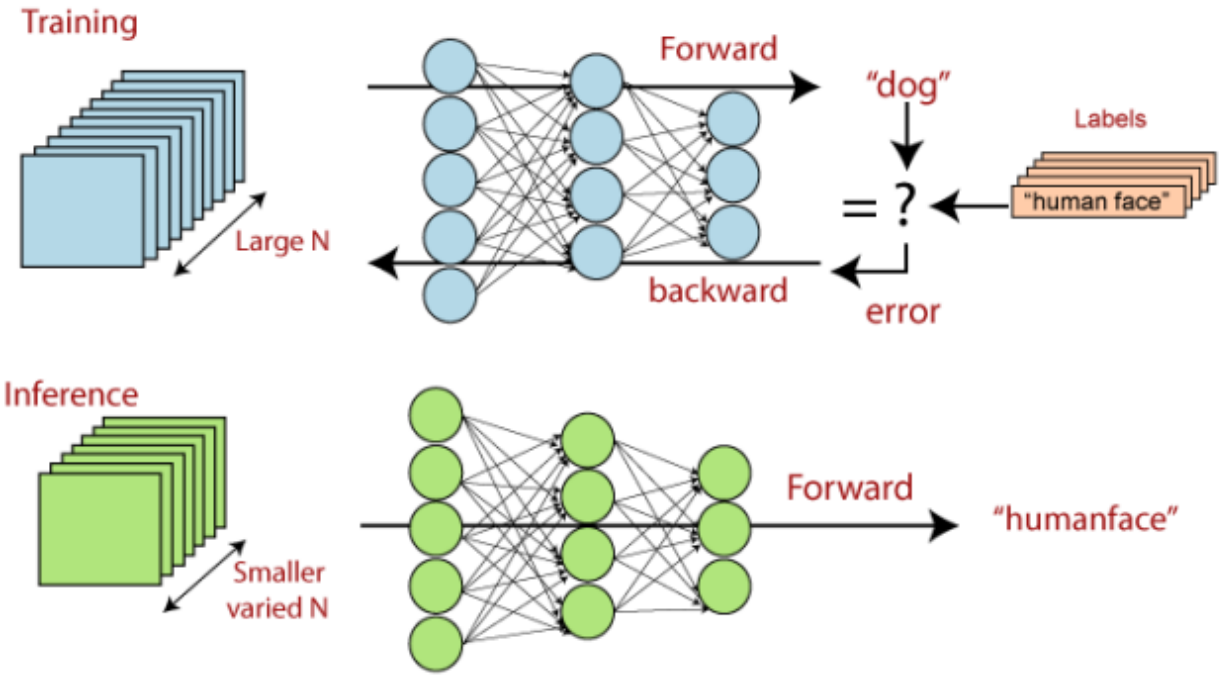


Figure 2-2-2-4: Deep Learning Approach

## 2.2 Convolutional neural networks (CNN)

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns. CNN is a mathematical construct that is typically composed of three types of layers (or building blocks): convolution, pooling, and fully connected layers. The first two, convolution and pooling layers, perform feature extraction, whereas the third, a fully connected layer, maps the extracted features into final output, such as classification. A convolution layer plays a key role in CNN, which is composed of a stack of mathematical operations, such as convolution, a specialized type of linear operation. In digital images, pixel values are stored in a two-dimensional (2D) grid, i.e., an array of numbers and a small grid of parameters called kernel, an optimizable feature extractor, is applied at each image position, which makes CNNs highly efficient for image processing, since a feature may occur anywhere in the image. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex. The process of optimizing parameters such as kernels is called training, which is performed so as to minimize the difference between outputs and round

truth labels through an optimization algorithm called backpropagation and gradient descent, among others.

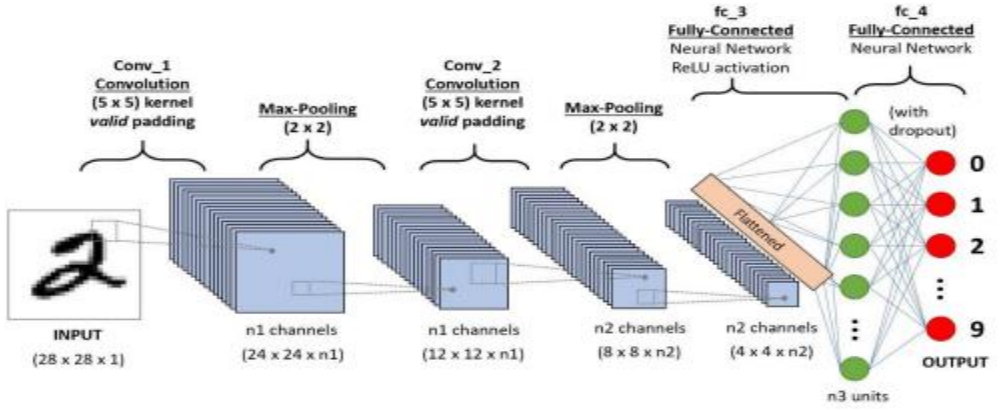


Figure 2-2-2-3: Convolutional Network Approach

## 2.2 Studies and work on the self-driving cars:

There have been numerous previous studies and works related to self-driving cars, which have explored various aspects of the technology, including safety, reliability, regulatory and legal issues, and societal impacts. Some of the key studies and works in this area are described below:

1. The National Highway Traffic Safety Administration (NHTSA) released a report in 2013 that outlined its preliminary findings on the safety of autonomous vehicles. The report concluded that self-driving cars have the potential to significantly reduce the number of accidents on the road, but that further research and testing are needed to ensure their safety.
2. A study by the Eno Center for Transportation found that autonomous cars could reduce traffic fatalities by up to 90% and save the US economy $447 billion per year. The study also highlighted the need for regulatory and legal frameworks to address the unique challenges presented by self-driving cars.
3. The RAND Corporation conducted a study on the societal impacts of autonomous cars, which found that the technology could have significant economic and environmental benefits, but could also lead to job displacement and increased urban sprawl.

4. A study by the University of Michigan found that hackers could potentially take control of autonomous cars' critical systems, such as brakes and steering. The study highlighted the need for robust cybersecurity measures to ensure the safety and security of self-driving cars.
5. The California Department of Motor Vehicles (DMV) released a set of regulations for autonomous vehicles in 2018. The regulations require companies testing self-driving cars to obtain a permit from the DMV, report any accidents, and comply with certain safety and cybersecurity requirements.

These studies and works represent only a small sample of the extensive research that has been conducted on self-driving cars. As the technology continues to evolve, it will be important to continue to study and evaluate its impacts and address any challenges that arise to ensure the safe and responsible deployment of autonomous vehicles.

# Chapter 3

# System Architecture and Methods

## 3-1 System architectures:

In our project we used a real time camera to:
system's input is of real time image, then the system extracts some features from the captured image:

- In lane detection we use canny edge detection.
- In stop sign detection we use convolutional neural networks (Haar Cascade).
- If the car detected a turn in the road, it should slow down and make a correct turn in the road.
- If the car detected a stop sign from over the road it should slow down and stop for few seconds, then go again.

As shown in figure below:

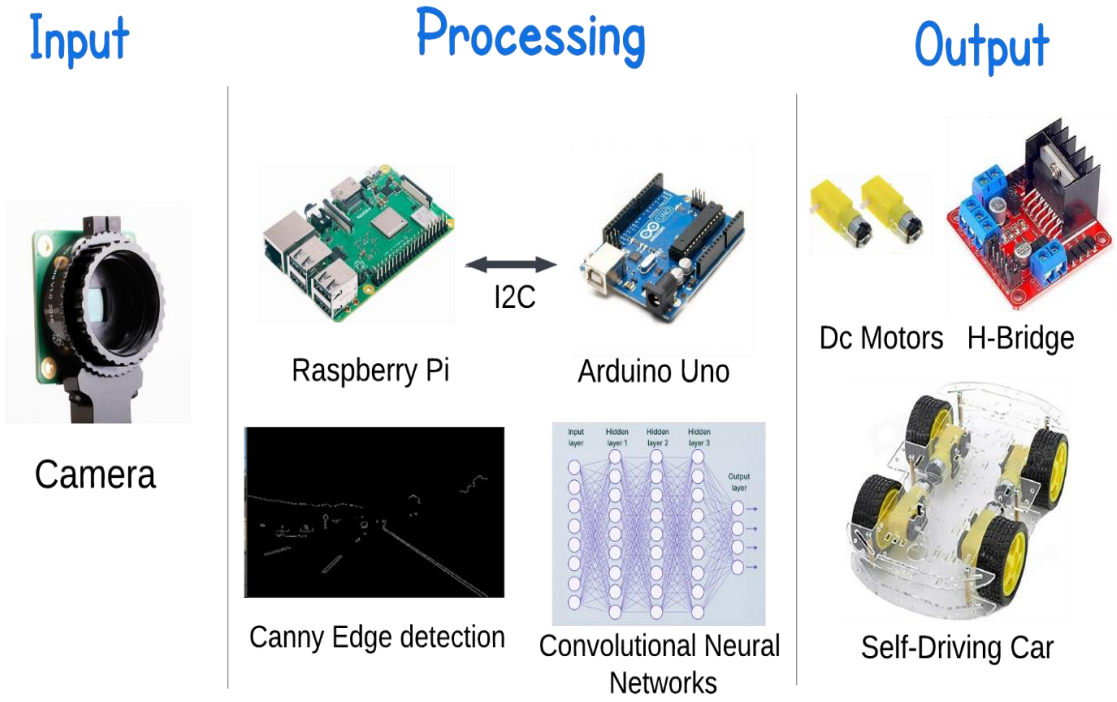


Figure 3-1 System architecture

## 3-2 Methods and procedures used:

To start talking about the methods in details we must divide it into 2 main topics:

- Methods for lane detecting

- Image pre-processing
- Feature extraction
- Model fitting

- Methods for Object detection

- Positive samples
- Negative samples
- Convolutional neural network training
- Detecting

### 3-2-1 Methods for Lane detecting:

We used an approach using only the edges of the path (road) are considered. The initial steps of Capturing the image, Warping and Thresholding. A few image pre-processing procedures are performed to get a clearer view of the path edges. The extracted edge geometry is then used to determine the curvature using sliding window algorithm. this approach is as follow:

## 1) Thresholding:

Since the approach is for the unmarked roads, Color thresholding is done in order to detect the color of the road instead of detecting the lane markings. The thresholding is not applied directly on the image obtained from video feed. Images obtained from video feed are in BGR (Blue Green Red) format, and are converted into HSV(Hue Saturation Value) [12] since it is better for object or color detection. HSV color space represents how colors behave under light. Mathematically HSV consists of three matrices for 'Hue', 'Saturation', and 'Value' with ranges lying between 0-179, 0-255 and 0-255 respectively. The main application for HSV is color based image segmentation which is used in both of the approaches for separating the road from its surroundings. Thresholding will convert the image into a binary image i.e the area where the color of the path is detected will be white and the rest of the part of that image will be kept black.


Figure 3-2-1: Thresholding

## 2) Warping:

The image obtained via the camera module has an angled perspective. The lane edges cannot be precisely located using this image as is. In order to fix this image warping concept is used. Image warping is used to get a birds eye / top view of the path by changing the perspective of the image. In Perspective Transformation, the perspective of an image or a video frame is changed in order to get certain insights in a better way. For warping a set of points is provided over the image. The values of these points will determine the Region of Interest (ROI). With these two sets of points the perspective of the image or a video

frame is transformed. It is done by passing these coordinates to OpenCV library functions. The resultant would be an image which would be cropped as per the ROI and having a bird's eye perspective. Warping is one of the most common preprocessing techniques used in lane detection systems as it provides an image upon which further calculations would be easier as compared to original image obtained through camera. The basic idea is to get a rectangle shape when the road is straight.

Figure 3-2-1: Wrapping

## 3) Gaussian Blurring:

Due to uneven road conditions, the edges of the thresholded image are too sharp. This may lead to improper edge extraction. Thus, the imperfections are smoothened out by applying a blur function to the image. Various methods are available like averaging, bilateral filtering, Gaussian blurring and median blurring. For the test case in consideration, Gaussian blurring provided the best balance in reducing noise and smoothening out the sharp edge imperfections. This algorithm scans the entire image pixels, followed by recalculating the pixel values. The intensity of this operation is adjusted by a kernel. The greater the kernel size, the greater the blur effect but at the same time can lead to decreased details. Gaussian blur mixes the values of the neighboring pixels depending on the distance to the reference pixel. The pixels right next to the reference pixel get more of the colors from the reference pixel while the pixels farther away get less of the colors. This can be viewed graphically wherein the values closer to 0 on the x-axis have higher magnitude on the y-axis which gradually decreases as we go further away from the Centre. Thus, the uneven edges get smoothened out as the pixels don't have a too sharp change in their values after blurring.

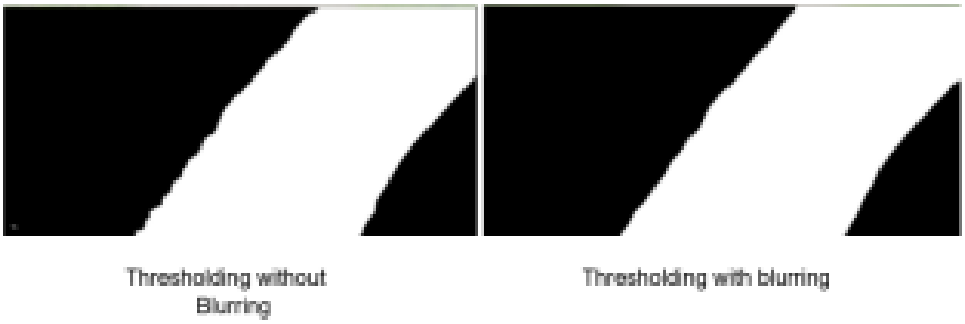


Figure 3-2-1: Thresholding comparison

## 4) Canny Edge Detection:

The next step is to extract the lane boundaries from the thresholded binary image. For straight lines, this can be achieved by Hough transform. However, the proposed test case needs to detect curved edges as well, so Canny edge detection algorithm is used. The Canny edge detector uses a multi-stage algorithm to detect edges in images by identifying large changes in gradients of the pixels. The steps involved are:

• The noise present in the image is reduced with a 5x5 Gaussian filter.
• The intensity gradient of the image is found by Sobel kernel which detects horizontal, vertical and diagonal edges.
• A scan of the complete image is performed followed by removing any unwanted pixels which may not represent the edge; this is also known as Non-maximum Suppression.
• The final stage i.e., Hysteresis Thresholding decides which all edges are really edges and which are not. For this, two threshold values - min and max - are used. Any edges having intensity gradient more than the maximum value are accepted as edges and those below minimum value are discarded. The remaining which lie between these two values are classified based on their connectivity.

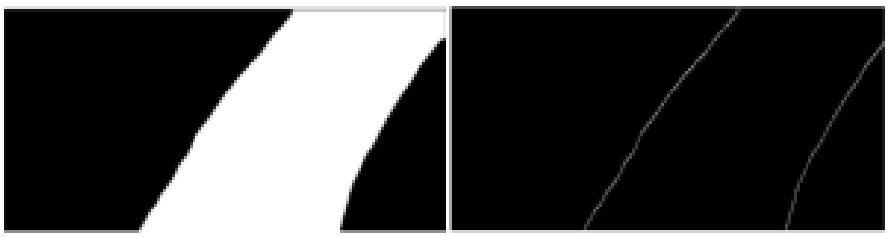

Figure 3-2-1: Canny edge detection

## 5) Image dilation:

The result of the canny detection function is an image with just the edge geometry of the path with the remaining image features removed. This needs to be passed to a searching algorithm for detection of this geometry. But before doing so image dilation is performed which makes the edges more clear and filled with less breaks. Dilation expands the image pixels and adds pixels to object boundaries.

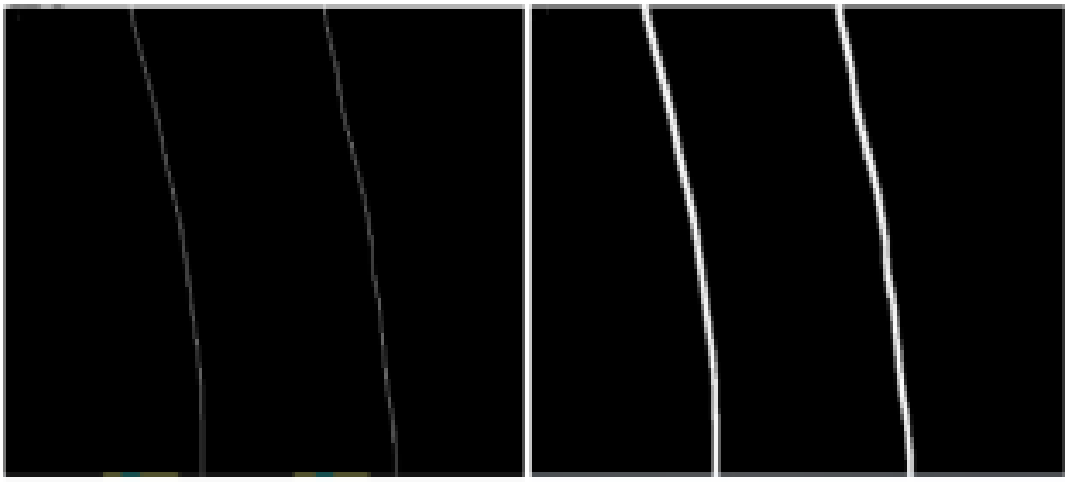

Figure 3-2-1: Image dilation

## 6) Sliding Window Algorithm:

The next step involves detecting the position where the edges are present in the Cannied image.

• The image is split into horizontal slices, and a search window i.e., a rectangular region of fixed width and height is slid across each slice finding areas of highest frequency.

• For this application, a window search takes a histogram of the lower half of the binary image, revealing the neighborhood about where the lane lines begin.

• Once this is found for both left and right edges, a fitting function is applied to get a best fit curve on the line. Window searching can be computationally expensive. To save time for subsequent frames, a margin search can be

performed. i.e., it only searches in a tight margin around the previously established lane line.

• The values of positions obtained for both left and right edges are stored and then subtracted with each other to find the curvature value.

• Similarly, the vehicle offset i.e., the deviation of the vehicle from the ideal center is obtained by calculating the difference between the ideal center and the edge position.

• It is more accurate than the 1st approach as using the edge geometry gives a precise value of the lane position.

• Other important measures such as vehicle offset can be measured.

• The downside of this method is if the edges are too uneven or obscured by vegetation the curve values may get disturbed.

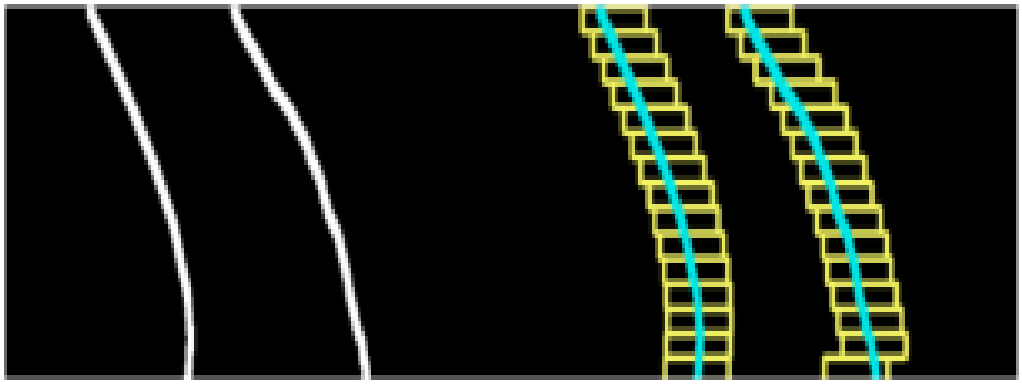

Figure 3-2-1: Siding window output


Figure 3-2-1: Approach output

## 3-2-2 Methods for Object detection:

In object detection methods we used haar cascade algorithm to detect stop signs. The Haar Cascade algorithm is a popular object detection technique used in computer vision. It was introduced by Viola and Jones in 2001 and has since become an essential tool in various applications such as face detection, pedestrian detection, and many others.

The Haar Cascade algorithm is based on the concept of Haar features, which are simple rectangular patterns that can be used to represent different parts of an object. The algorithm works by using a set of classifiers that are trained to recognize these Haar features.

These classifiers are combined in a cascade to form a strong detector that can identify the object of interest with high accuracy. The first step in the Haar Cascade algorithm is to extract the Haar features from the input image. This is done by sliding a rectangular window over the image and computing the difference between the sum of pixel values in the black and white regions of the window. This difference is then compared to a threshold value to determine whether or not the Haar feature is present in that region of the image.

Once the Haar features are extracted, the next step is to train the classifiers. This is done by using a set of positive and negative training images. The positive images contain the object of interest, while the negative images do not. The classifiers are trained to maximize the difference between the Haar feature responses in the positive and negative images.

After the classifiers are trained, they are combined in a cascade. The cascade consists of multiple stages, with each stage containing several classifiers. The output of each stage is used as input to the next stage. If the output of a stage is negative, the object is rejected, and the algorithm moves on to the next region of the image. If the output is positive, the algorithm continues to the next stage. The cascade is designed to reject as many non-object regions as possible to reduce the number of false positives.

The Haar Cascade algorithm has several advantages over other object detection techniques. It is fast, robust, and can detect objects in real-time. It is also relatively easy to train and can be adapted to detect different types of objects.

In conclusion, the Haar Cascade algorithm is a powerful tool for object detection in computer vision. It is based on the concept of Haar features and uses a set of classifiers combined in a cascade to detect objects with high accuracy. Its speed, robustness, and ease of use make it a popular choice for various applications.


Figure 3-2-2: Positive sample vs. Negative sample

# Chapter 4

# System Implementation and Results

## 4-1 Description of materials used (Dataset):

Our data is an image dataset taken by the Raspberry camera from different angles and views. The camera took image every 2 seconds so we can setup the angle and view. we created 2 parts of samples:

### 1- Positive samples:

50 images of the object we already need to detect (Stop sign).

### 2- Negative samples:

300 images of the surrounding environment the car is going to drive in.

## 4-2 Programs used:

### 1- Python:

Python is one of the most programming language used recently in various fields. Some of these fields is hardware and robotics, machine learning and deep learning and image processing.

Python provided us with modules like open cv which we used the most in our project for object detection and image processing.

### 2- Open CV:

OpenCV (Open Source Computer Vision Library) is an open-source library of computer vision and image processing functions that can be used in a variety of hardware projects, including those related to self-driving cars. Here are some of the advantages of using OpenCV in implementing hardware projects for self-driving cars:

1. Comprehensive Feature Set: OpenCV provides a comprehensive set of computer vision and image processing functions that can be used for various tasks, including object detection, image segmentation, and motion detection. These features are essential in self-driving cars, where the ability to analyze and interpret visual data is crucial.

2. Cross-Platform Compatibility: OpenCV is cross-platform compatible, which means that it can be used on a variety of hardware platforms, including embedded systems and mobile devices. This makes it ideal for implementing hardware projects in self-driving cars, which require hardware and software to work together seamlessly.

3. Optimized Performance: OpenCV is optimized for performance, with many of its functions implemented using highly efficient algorithms and data structures. This means that it can perform complex image processing tasks quickly and efficiently, even on low-powered hardware platforms.

4. Large Community and Support: OpenCV has a large and active community of developers who contribute to its development and provide support. This means that there are many resources available for developers who are working on self-driving car projects using OpenCV.

5. Integration with Other Libraries and Frameworks: OpenCV can be easily integrated with other libraries and frameworks, such as TensorFlow and Keras, which are commonly used in machine learning and deep learning applications. This makes it easier to incorporate machine learning algorithms into self-driving car projects.

In summary, OpenCV is a powerful and versatile library that can be used to implement hardware projects for self-driving cars. Its comprehensive feature set, cross-platform compatibility, optimized performance, large community and support, and integration with other libraries and frameworks make it an ideal choice for developers who are working on self-driving car projects.

# 3- Thonny IDE:

Thonny is an integrated development environment (IDE) for Python that is designed to be easy to use and beginner-friendly, making it an ideal choice for implementing hardware projects in self-driving cars. Here are some of the advantages of using Thonny IDE in implementing hardware projects for self-driving cars:

1. Simple and User-Friendly Interface: Thonny has a simple and easy-to-use interface, which makes it easy for beginners to get started with Python programming. This is particularly useful for hardware projects in self-driving cars, where developers may not have a background in programming.
2. Built-in Debugger: Thonny comes with a built-in debugger that makes it easy to trace the flow of the program and identify errors. This is particularly important in hardware projects, where errors can be costly and difficult to diagnose.
3. Integration with Microcontrollers: Thonny can be easily integrated with microcontrollers, which are commonly used in self-driving car projects. This allows developers to program and control microcontrollers directly from the IDE, making it easier to implement hardware projects.
4. Code Completion and Error Highlighting: Thonny has a code completion feature that can help developers write code faster and more accurately. It also highlights errors in real-time, which can help developers catch mistakes before they become serious issues.
5. Cross-Platform Compatibility: Thonny is cross-platform compatible, which means that it can be used on a variety of operating systems, including Windows, macOS, and Linux. This makes it easier for developers to work on self-driving car projects using different hardware platforms.

In summary, Thonny IDE is a beginner-friendly and easy-to-use IDE that is ideal for implementing hardware projects in self-driving cars. Its simple interface, built-in debugger, integration with microcontrollers, code completion, error highlighting, and cross-platform compatibility make it an ideal choice for developers who are working on self-driving car projects and want to get started with Python programming.

# 4- Arduino IDE:

Arduino is an open-source electronics platform that is widely used for implementing hardware projects, including those related to self-driving cars. The Arduino IDE (Integrated Development Environment) is a software application that is used for programming Arduino boards and microcontrollers. Here are some of the advantages of using Arduino IDE in implementing hardware projects for self-driving cars:

1. Easy to Learn and Use: Arduino IDE is designed to be easy to learn and use, even for beginners who may not have a background in programming. It has a simple and intuitive interface, making it easy to write, compile, and upload code to Arduino boards.
2. Large Community and Libraries: Arduino has a large and active community of developers, which means that there are many resources and libraries available to help developers implement their projects. This is particularly useful in the context of self-driving cars, where there are many complex tasks that require specialized tools and libraries.
3. Cross-Platform Compatibility: Arduino IDE is cross-platform compatible, which means that it can be used on a variety of operating systems, including Windows, macOS, and Linux. This makes it easier for developers to work on self-driving car projects using different hardware platforms.
4. Integration with Sensors and Actuators: Arduino boards can be easily integrated with sensors and actuators, which are commonly used in self-driving car projects. This makes it easier to interface with hardware components and control their behavior.
5. Low-Cost and Open-Source: Arduino boards are relatively low-cost and open-source, which makes them accessible to a wide range of developers and enthusiasts. This also means that the hardware and software designs can be easily modified and customized to meet the specific needs of a self-driving car project.

In summary, Arduino IDE is a powerful and versatile platform for implementing hardware projects in self-driving cars. Its ease of use, large community and libraries, cross-platform compatibility, integration with sensors and actuators, and low-cost and open-source nature make it an ideal choice for developers who are working on self-driving car projects and want to interface with hardware components.

## 5- NoMachine:

NoMachine is a remote desktop software that can be used with Raspberry Pi, a low-cost, credit-card sized computer that is widely used for a variety of hardware projects. Here are some of the advantages of using NoMachine with Raspberry Pi:

1. Remote Access and Control: NoMachine allows users to remotely access and control their Raspberry Pi from anywhere in the world, which can be useful for testing and debugging. This is particularly important for Raspberry Pi projects that are located in remote or difficult-to-access locations.
2. Low Latency and High Performance: NoMachine is designed to provide low-latency and high-performance remote desktop connections, which means that users can control their Raspberry Pi in real-time without experiencing lag or delays. This is important for Raspberry Pi projects, where timing and responsiveness are critical.
3. Cross-Platform Compatibility: NoMachine is cross-platform compatible, which means that it can be used on a variety of operating systems, including Windows, macOS, and Linux. This makes it easier for developers to work on Raspberry Pi projects using different hardware platforms.
4. Security Features: NoMachine has several security features, including encryption, two-factor authentication, and the ability to restrict access to specific users or IP addresses. This helps to protect sensitive data and prevent unauthorized access to the Raspberry Pi.
5. Remote Collaboration: NoMachine allows multiple users to connect to and control a remote Raspberry Pi simultaneously, which can be useful for collaborative projects. This is particularly important for Raspberry Pi projects, which often involve teams of developers working together on complex tasks.

In summary, NoMachine is a powerful and versatile remote desktop software that can be used with Raspberry Pi to provide remote access and control capabilities. Its low-latency and high-performance connections, cross-platform compatibility, security features, and remote collaboration capabilities make it an ideal choice for developers who are working on Raspberry Pi projects and need to access and control their Raspberry Pi remotely.

## 4-3 Step-up Configuration (hardware):

## 1- Raspberry pi:

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education.

The Raspberry Pi launched in 2012, and there have been several iterations and variations released since then. The original Pi had a single-core 700MHz CPU and just 256MB RAM, and the latest model has a quad-core CPU clocking in at over 1.5GHz, and 4GB RAM.

The Raspberry Pi is a very cheap computer that runs Linux, but it also provides a set of GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IOT).

There have been many generations of the Raspberry Pi line: from Pi 1 to 4. There has generally been a Model A and a Model B of most generations. Model A has been a less expensive variant, and tends to have reduced RAM and fewer ports (such as USB and Ethernet).

Here's the line-up so far:

- Pi 1 Model B+ (2014)
- Pi 1 Model A+ (2014)
- Pi 2 Model B (2015)
- Pi Zero (2015)
- Pi 3 Model B (2016)
- Pi Zero W (2017)
- Pi 3 Model B+ (2018)
- Pi 3 Model A+ (2019)
- Pi 4 Model A (2019)
- Pi 4 Model B (2020)
- Pi 400 (2021)

## Raspberry Pi 4 Model B

This model has the latest high-performance Quad-Core 64-bit Broadcom 2711, Cortex A72 processor clocked at 1.5GHz speed.
This processor uses 20% less power and offers 90% greater performance than the previous model.
Raspberry Pi 4 model comes in three different variants of 2 GB, 4 GB, and 8 GB LPDDR4 SDRAM.
The other new features of the board are dual-display support up to 4k resolutions via a pair of micro-HDMI ports, hardware video decodes at up to 4Kp60, dual-channel 2.4/5.0GHz wireless LAN, true Gigabit Ethernet, two USB 3.0 ports and Bluetooth 5.0.

## Raspberry Pi 4 Specs:

- Broadcom BCM2711 chip consist of Quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 2GB, 4GB, and 8GB of LPDDR4 SDRAM (depending on the version of the board)
- Dual-channel 2.4/5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- Two USB 3.0 ports and two USB 2.0 ports.
- Raspberry Pi standard 40 pin GPIO header
- Two micro-HDMI ports (support up to 4kp60 resolution)
- 2-lane MIPI DSI display port
- 2-lane MIPI CSI camera port
- 4-pole stereo audio and composite video port
- 265 (4k@60 decode), H264 (1080@60 decode and 1080@30 encode)
- OpenGL ES 3.0 graphics
- Micro-SD card slot for loading operating system and data storage
- 1V/3A DC via USB-C connector
- Power over Ethernet (PoE) enabled (requires PoE HAT board)
- Operating temperature: 0 – 50oC

## Raspberry Pi 4 Board Layout:

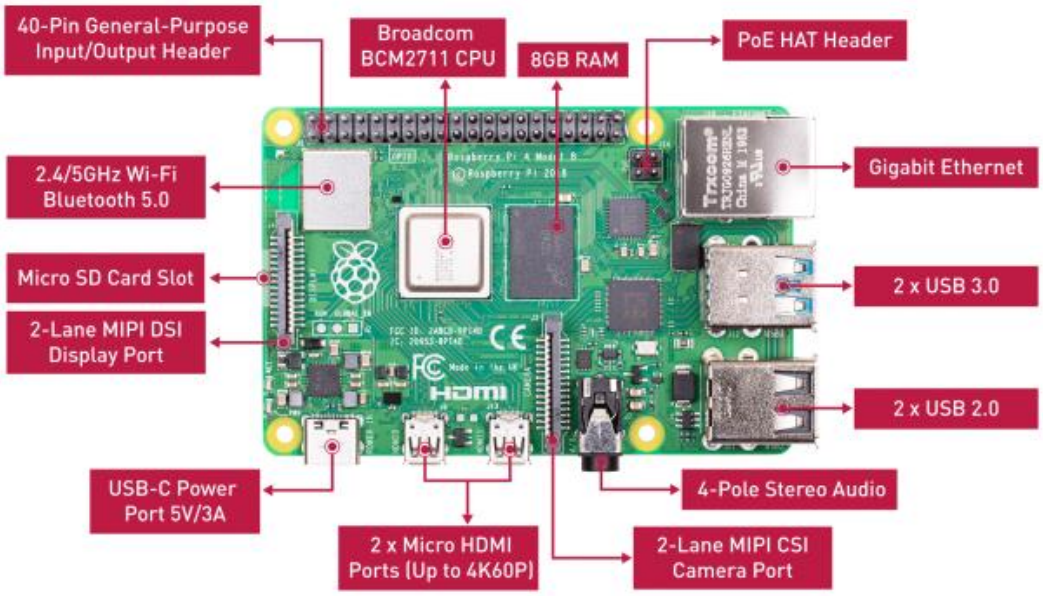


Figure 4-3-1: Raspberry Pi 4 Board Layout

- CPU: It consists of a Broadcom BCM2711 chip which contains a 1.5GHz 64-bit quad-core ARM Cortex-A72 processor (using an ARMv8-architecture core).
- GPU: Broadcom Video Core VI @ 500 MHz was released in 2009. It is capable of Blu-ray quality video playback, H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics.
- RAM: It comes with 2GB, 4GB, and 8GB (depends on different versions) variants of LPDDR4 SDRAM.
- USB port: It consists of two USB 3.0 and two USB 2.0 ports to connect it to an external keyboard, mouse, or other peripheral devices.
- USB power port: It consists of a 5.1V, 3A USB type-C power port.
- HDMI port: Two micro HDMI ports capable of supporting up to 4k@60HZ resolution.
- Ethernet Port: It comes with true Gigabit Ethernet capable of sending Ethernet frames at a rate of one gigabit per second (1 billion bits per second).
- Composite Video Output: Both the audio output socket and the video composite socket reside in a single 4-pole 3.5mm socket.
- SD card Slot: A micro-SD card slot is used for booting up the operating system and storage purposes.

# Raspberry Pi 4 GPIO Pinout:

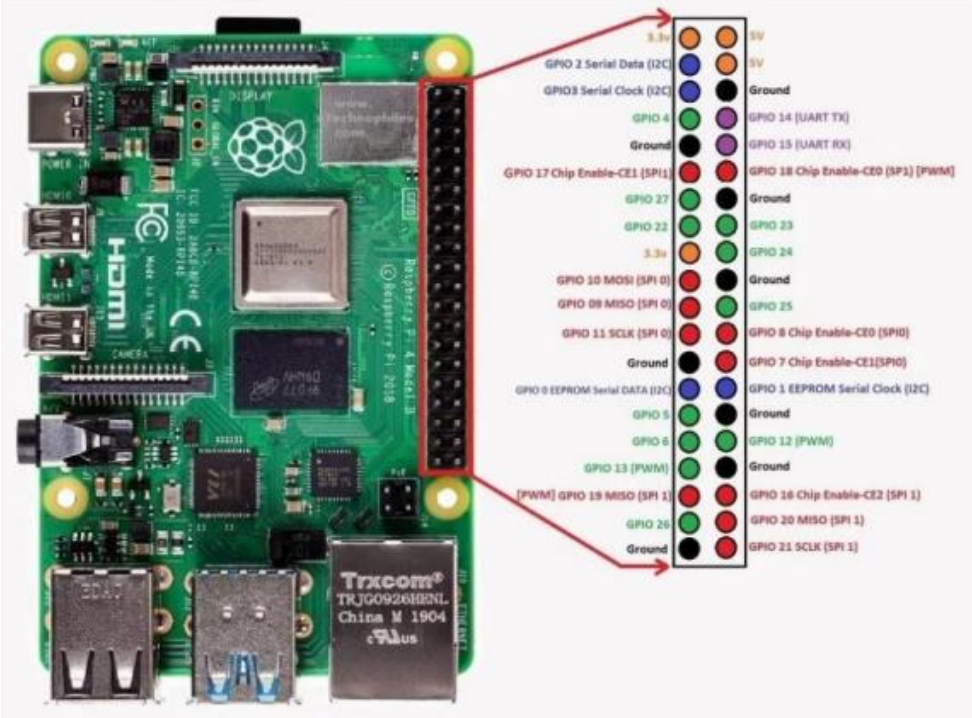


Figure 4-3-1: Raspberry Pi 4 GPIO Pinout

Raspberry Pi GPIO stands for General Purpose Input Output pins. These pins are used to connect the Raspberry pi board to external input/output peripheral devices.

This model B consists of a 40-pin GPIO header. Out of these 40 pins, 26 pins are GPIO pins.


Figure 4-3-1: Raspberry Pi 4 GPIO headers

## Power Pins on Raspberry Pi 4:

The raspberry pi 4 model B board consists of two 5V pins, two 3V3 pins, and 7 ground pins (0V).


Figure 4-3-1: Power pins on Raspberry Pi 4

## Ways to program the Raspberry PI 4 Board:

You can control the Raspberry Pi 4 GPIO pins using many programming languages. Some of the popular languages along with learning material is given below:

- GPIO Programming using Python
- Programming GPIO with C/C++ using standard kernel interface via libgpiod
- Programming GPIO with C/C++ using 3rd party library pigpio
- GPIO Programming using Scratch 1.4
- GPIO Programming using Scratch 2
- GPIO Programming using Processing3

## 2- Arduino Uno:

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It includes 6 analog pins inputs, 14 digital pins.

Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.

You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

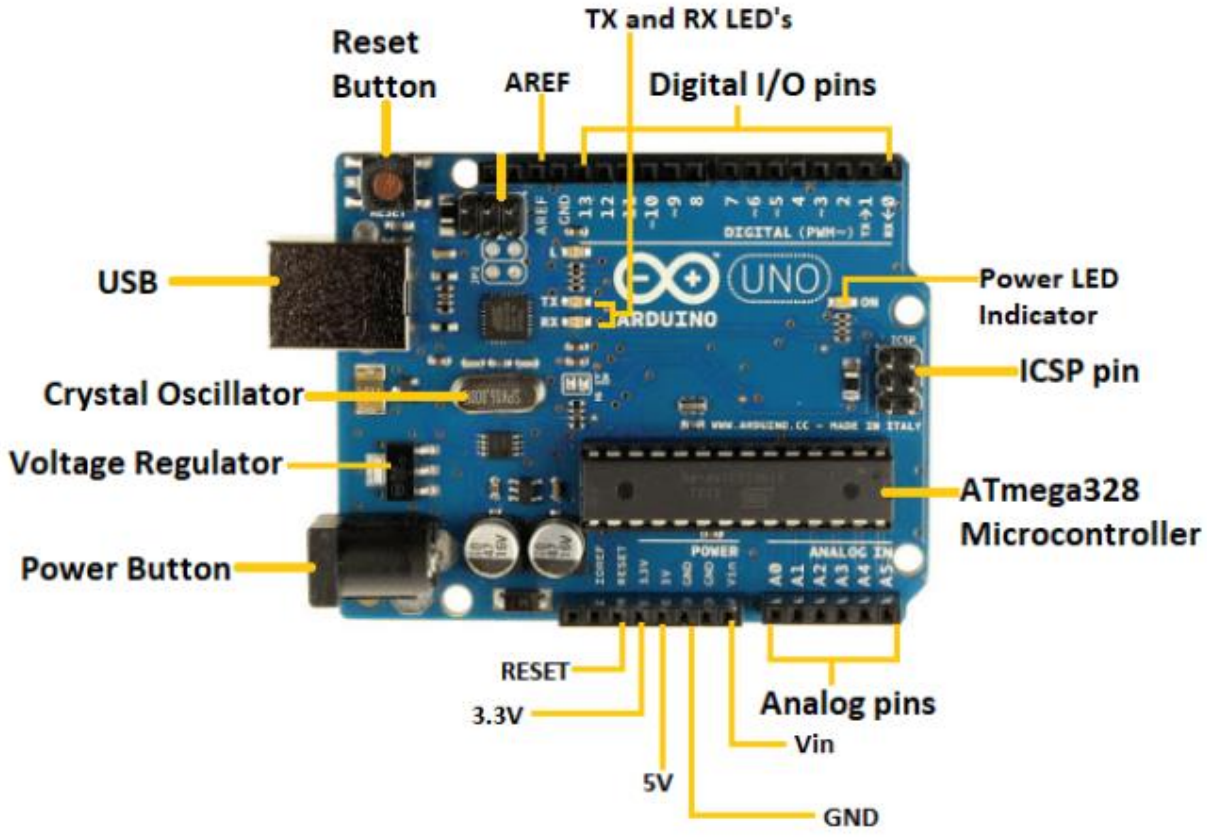


Figure 4-3-2: Arduino uno

- ATmega328 Microcontroller- It is a single chip Microcontroller of the Atmel family. The processor code inside it is of 8-bit. It combines Memory (SRAM, EEPROM, and Flash), Analog to Digital Converter, SPI serial ports, I/O lines, registers, timer, external and internal interrupts, and oscillator.
- ICSP pin - The In-Circuit Serial Programming pin allows the user to program using the firmware of the Arduino board.

- Power LED Indicator- the ON status of LED shows the power is activated. When the power is OFF, the LED will not light up.
- Digital I/O pins- The digital pins have the value HIGH or LOW. The pins numbered from D0 to D13 are digital pins.
- TX and RX LED's- the successful flow of data is represented by the lighting of these LED's.
- AREF- the Analog Reference (AREF) pin is used to feed a reference voltage to the Arduino UNO board from the external power supply.
- Reset button- It is used to add a Reset button to the connection.
- USB- It allows the board to connect to the computer. It is essential for the programming of the Arduino UNO board.
- Crystal Oscillator- the Crystal oscillator has a frequency of 16MHz, which makes the Arduino UNO a powerful board.
- Voltage Regulator- The voltage regulator converts the input voltage to 5V.
- GND- Ground pins. The ground pin acts as a pin with zero voltage.
- Vin- It is the input voltage.
- Analog Pins- The pins numbered from A0 to A5 are analog pins. The function of Analog pins is to read the analog sensor used in the connection. It can also act as GPIO (General Purpose Input Output) pins.

## 3- Raspberry pi camera:

Pi Camera module is a camera which can be used to take pictures and high-definition video.

Figure 4-3-3: Camera Module Connected to Raspberry Pi

Figure 4-3-3: Raspberry Pi Camera

## 4- L298n Motor driver Module:

This L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors.

This module consists of an L298 motor driver IC and a 78M05 5V regulator.

L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

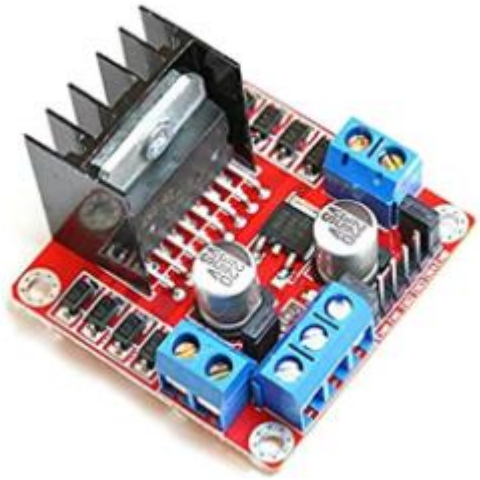

Figure 4-3-4: L298n Motor driver module

### Features & Specifications

- Driver Model: L298N 2A
- Driver Chip: Double H Bridge L298N
- Motor Supply Voltage (Maximum): 46V
- Motor Supply Current (Maximum): 2A
- Logic Voltage: 5V
- Driver Voltage: 5-35V
- Driver Current:2A
- Logical Current:0-36mA
- Maximum Power (W): 25W
- Current Sense for each motor
- Heatsink for better performance
- Power-On LED indicator

## 5- 4 wheels 2 layers robot smart car chassis kit:

Figure 4-3-5: 4 wheels 2 layers robot smart car chassis kit

## 6- Charger power bank:

Figure 4-3-6: Charger power bank

## 4.4 Experimental and results:

### 4.4.1 initializing the camera:

#### 1. Setting up the camera:

The “PiCamera” class is imported and initialized as “camera”.

The resolution of the camera is set to “length” x “width” pixels using “camera.resolution”.

Various camera settings like brightness, contrast, and saturation are configured using “camera.brightness”, “camera.contrast”, and “camera.saturation” respectively.

#### 2. Image capture:

The “image_capture” function is defined, which starts the camera preview, captures an image, and stops the preview.

The captured image is stored in the frame variable, which is not visible in the provided code snippet.

You may need to ensure that the frame variable is properly initialized and accessible.

#### 3. Calculating FPS:

The “calculate_fps” function is defined, which takes the start and end times as input and calculates the frames per second based on the elapsed time.

The elapsed time is obtained by subtracting the start time from the end time.

The function returns the FPS value.

## 4.4.2 lane detection :

### 1. Reading and resizing the input image:

The cv2.imread(frame) function is used to read the image from a file or capture device.

The cv2.resize function is used to resize the image to a specified width and length.

### 2. Converting color spaces:

The cv2.cvtColor function is used to convert the image from BGR (Blue-Green-Red) color space to RGB color space.

The cv2.cvtColor function is again used to convert the RGB image to grayscale (single-channel) image.

### 3. Drawing lines:

Four lines are drawn on the image using the cv2.line function. These lines connect specific points (pts) to form a rectangular shape.

### 4. Extracting a region of interest (ROI):

The cv2.boundingRect function is used to determine the bounding rectangle of the points (pts) defining the shape.

The region of interest (ROI) is extracted from the grayscale image based on the computed bounding rectangle.

### 5. Thresholding:

The grayscale image is thresholded using the cv2.threshold function to create a binary image.

The resulting thresholded image is resized to a specified width and length.

### 6. Edge detection:

The cv2.Canny function is used to detect edges in the ROI.

### 7. Further ROI operations:

The thresholded image ROI is extracted using the previously computed bounding rectangle.

The extracted ROI is resized to a specified width and length.

The edges and thresholded image ROI are merged using cv2.addWeighted.

The merged image is resized to a specified width and length.

## 8. Histogram calculation:

An empty list named histogram is created.

np.zeros(histogram) is used to create an array of zeros with the same shape as histogram.

histogram.resize(320) resizes the array to have a length of 320 (assuming histogram initially had a length of 0).

Two nested loops iterate over the pixels in the region of interest (resized_merged_image) from y=220 to y=319 (rows) and x=0 to x=319 (columns).

The intensity values of each pixel in the column are summed and stored in the corresponding index of the histogram array.

The resulting histogram is printed along with its length.

## 9. Finding left and right lane peaks:

Two variables, maxleft and leftlane, are initialized to 0.

A loop iterates from index 0 to 159 (left side of the histogram).

If the histogram value at the current index is greater than maxleft, maxleft is updated, and leftlane is set to the current index.

Similarly, maxright and rightlane are initialized to 0.

Another loop iterates from index 160 to 319 (right side of the histogram).

If the histogram value at the current index is greater than maxright, maxright is updated, and rightlane is set to the current index.

## 10. Drawing lines:

Two lines are drawn on the resized_roi image using cv2.line. One line is drawn from the top to the bottom at the leftlane index, and another line is drawn at the rightlane index.

These lines are meant to represent the detected left and right lanes.

## 11. Calculating the midpoint:

The midpoint between the leftlane and rightlane is calculated as ((rightlane - leftlane) / 2) + leftlane.

The midlineshow variable seems to represent a line drawn at the midpoint on the threshold image, but the thickness argument is not used correctly.

## 12. Calculating the result and displaying information:

A decision variable decisionvar is set to 160.

The difference between decisionvar and midlane is calculated and stored in result.

Several lines are drawn on the resized_roi image to represent the midlane, decision variable, and result.

The resized_roi image is displayed using cv2.imshow.

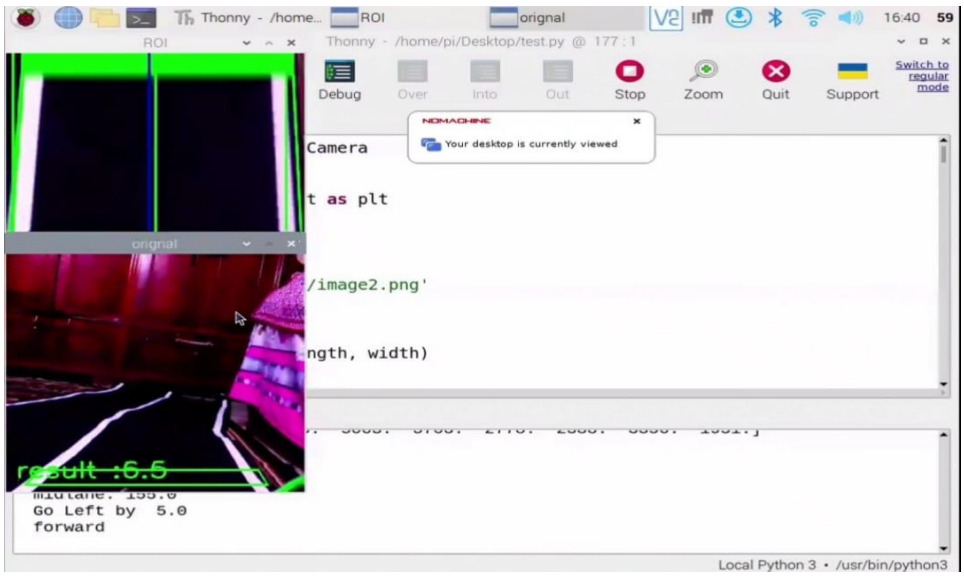


Figure 4-4-2: Lane Detection

## 4.4.3 object detection :

### 1. Extracting the ROI for stop sign detection:

The ROI is extracted from the image using the coordinates and dimensions obtained from cv2.boundingRect(pts_stop).

The extracted ROI is stored in the roi_stop variable.

The roi_stop is converted to grayscale using cv2.cvtColor with cv2.COLOR_BGR2GRAY.

### 2. Stop sign detection:

The cascade.detectMultiScale function is used to detect stop signs within the grayscale ROI.

The detected stop signs are stored in the stop_sign variable.

A loop iterates over the detected stop signs, and for each detection, a rectangle is drawn on the roi_stop using cv2.rectangle.

The variable d is computed based on the dimensions of the detected stop sign. It seems to be calculating a distance or some other value based on the width and height of the stop sign.

The variable d is converted to a string and stored in the text variable.

### 3. Displaying the result:

The text is added to the roi_stop image using cv2.putText to display the calculated distance or other information.

The roi_stop image, which includes the drawn rectangles and the text, is displayed using cv2.imshow with the window name "final stop".

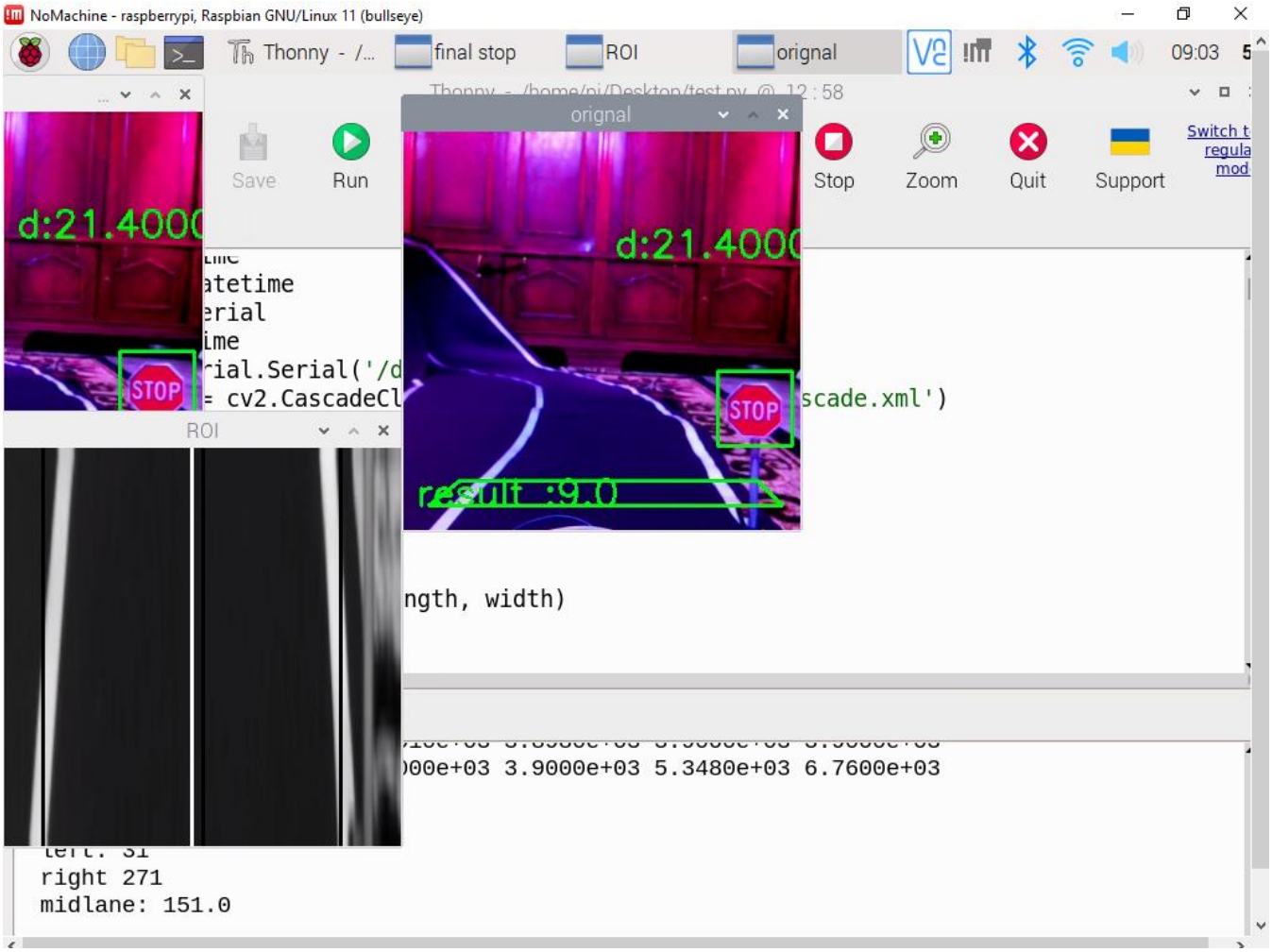


Figure 4-4-3: Stop sign Detection

# Chapter 5

# Run the Application

## 5.1 Run The Application :

To run the application in Thonny on Raspberry Pi, you can follow these steps:
Open Thonny: Go to the Raspberry Pi menu, navigate to Programming, and click on Thonny Python IDE. Thonny will open on your Raspberry Pi.
Create a new Python file: Click on "File" in the menu bar and select "New". A new Python file will open in the editor window.
Copy and paste the code: Copy the code you want to run and paste it into the new Python file in Thonny.
Save the file: Click on "File" in the menu bar and select "Save". Choose a location to save the file and give it a meaningful name with a .py extension (e.g., my_application.py).
Run the code: Click on the "Run" button (green play button) in the toolbar or press the F5 key to run the code. Thonny will execute the Python program.
Observe the output: If there is any output generated by the code (e.g., printed messages), it will appear in the "Shell" pane at the bottom of the Thonny window.

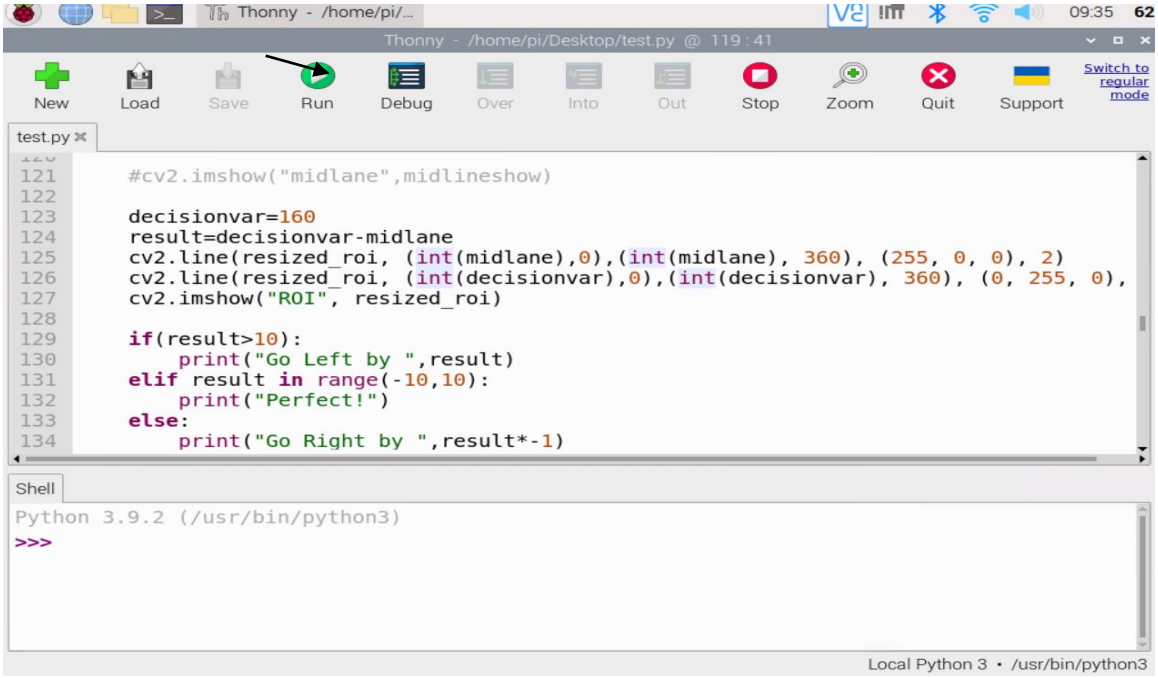


Figure 5-1: Run the Application

## 5.2 Go Forward

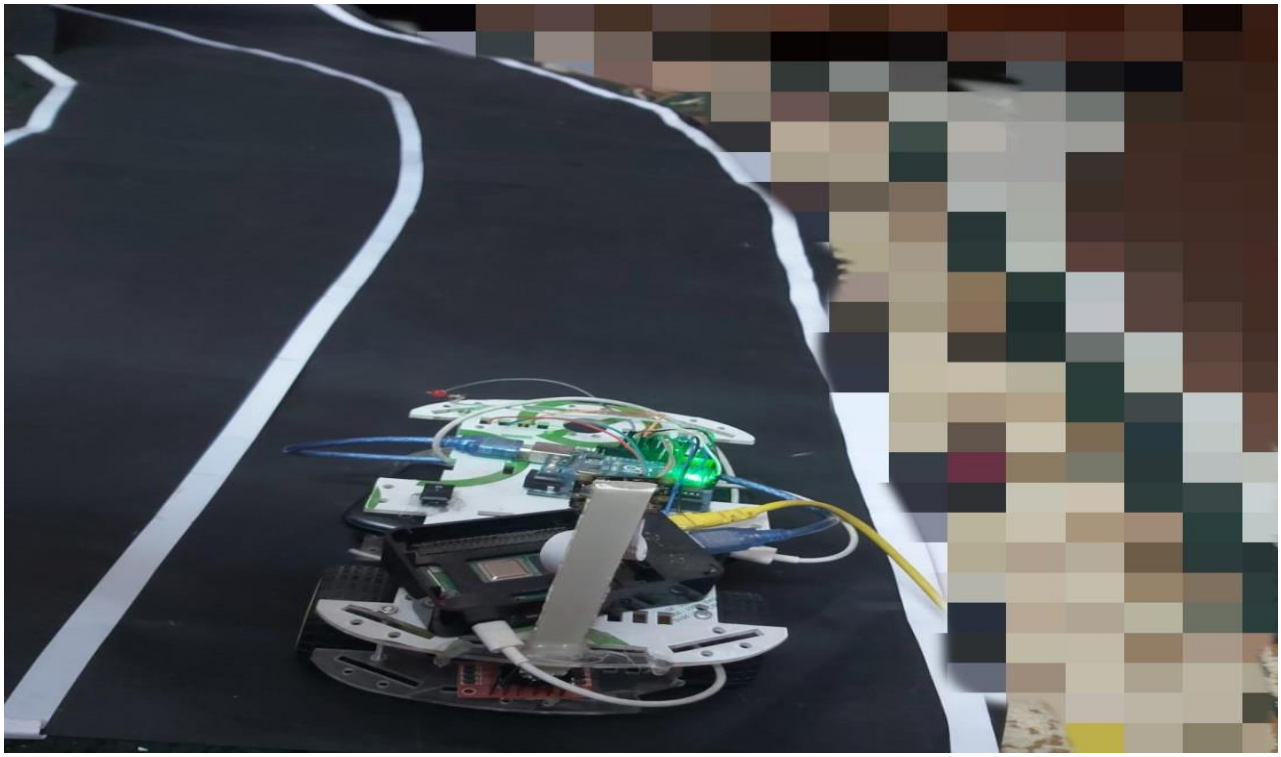

Figure 5-2: Go Forward(Car)

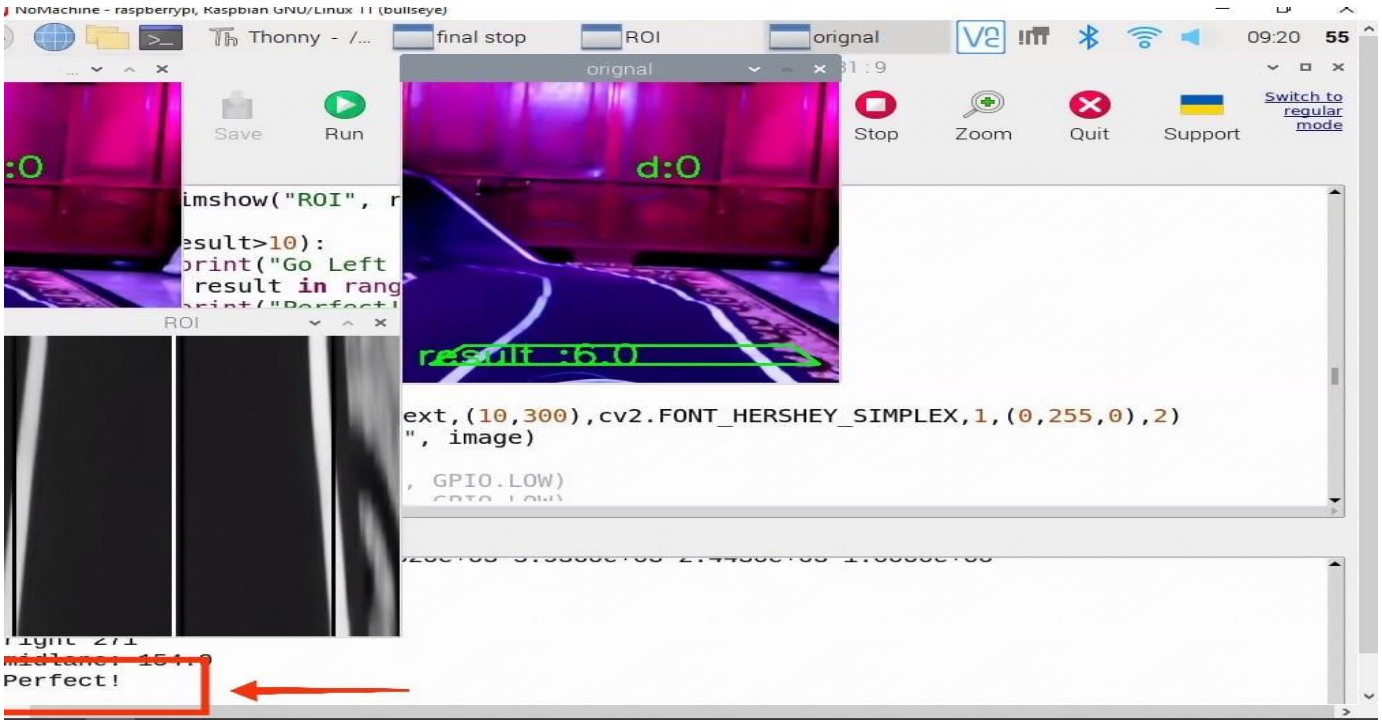


Figure 5-2: Go Forward (IDE)

# 5.3 Go Right

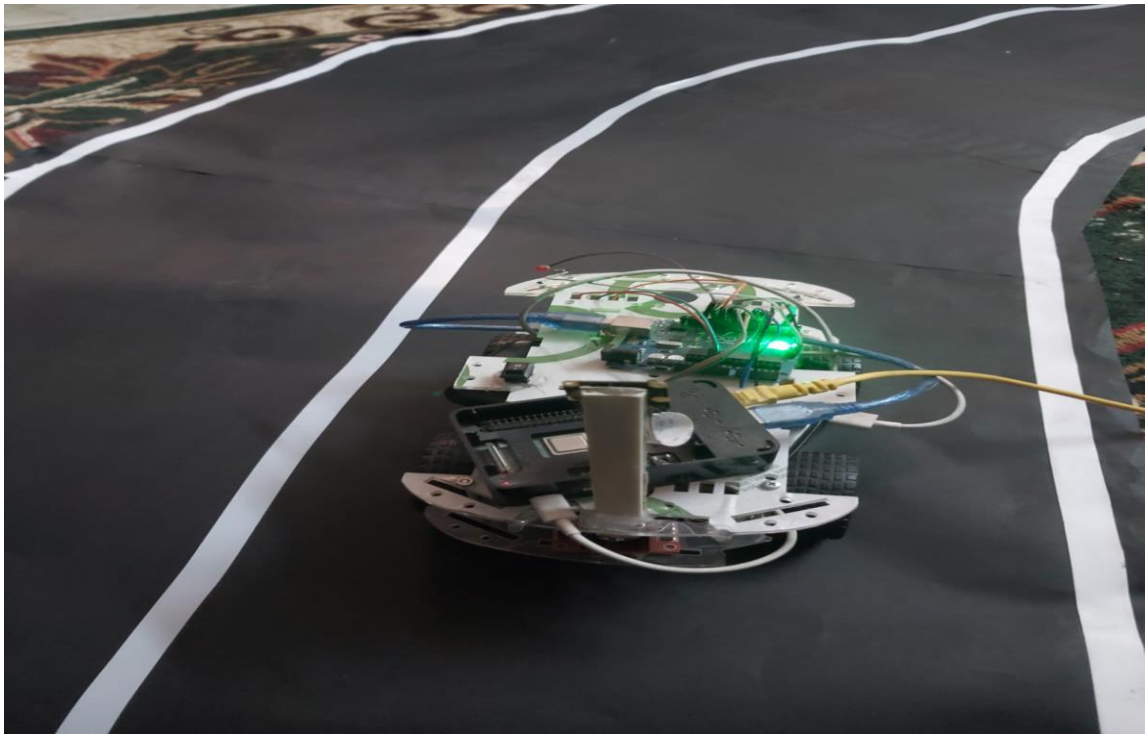

Figure 5-3: Go Right (Car)

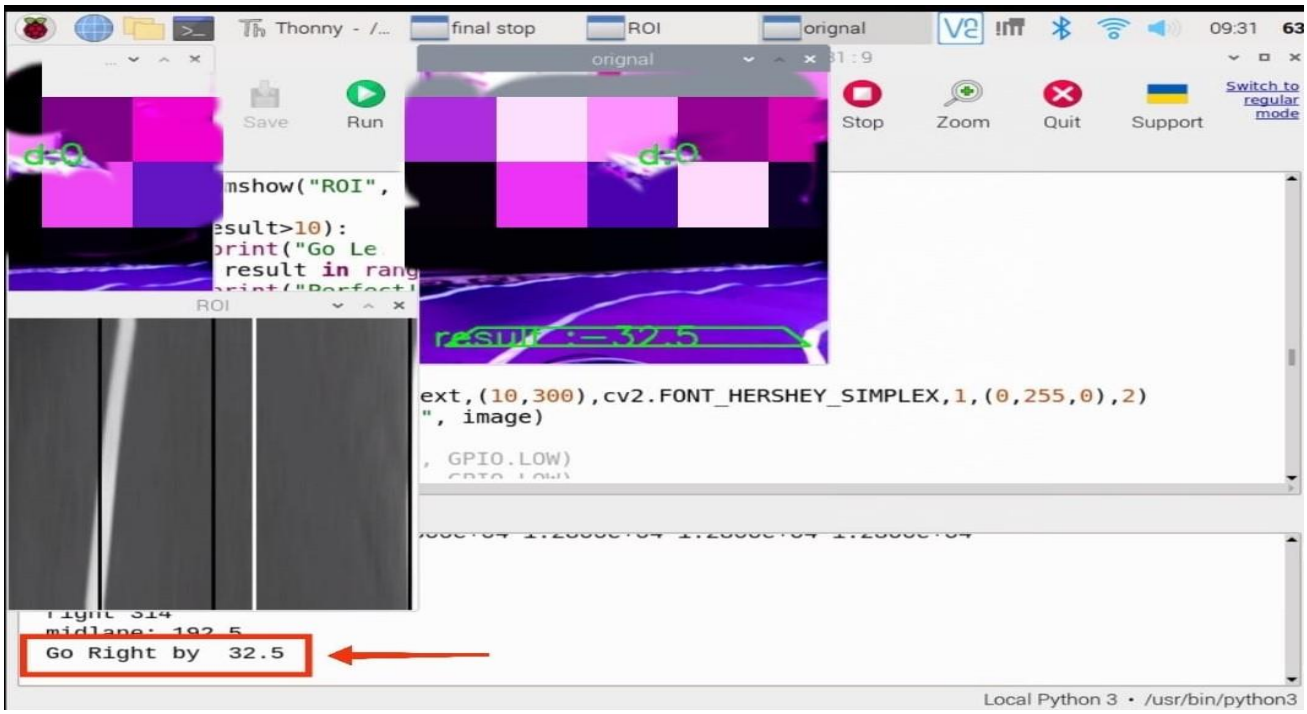


Figure 5-3: Go Right(IDE)

# 5.4 Go Left

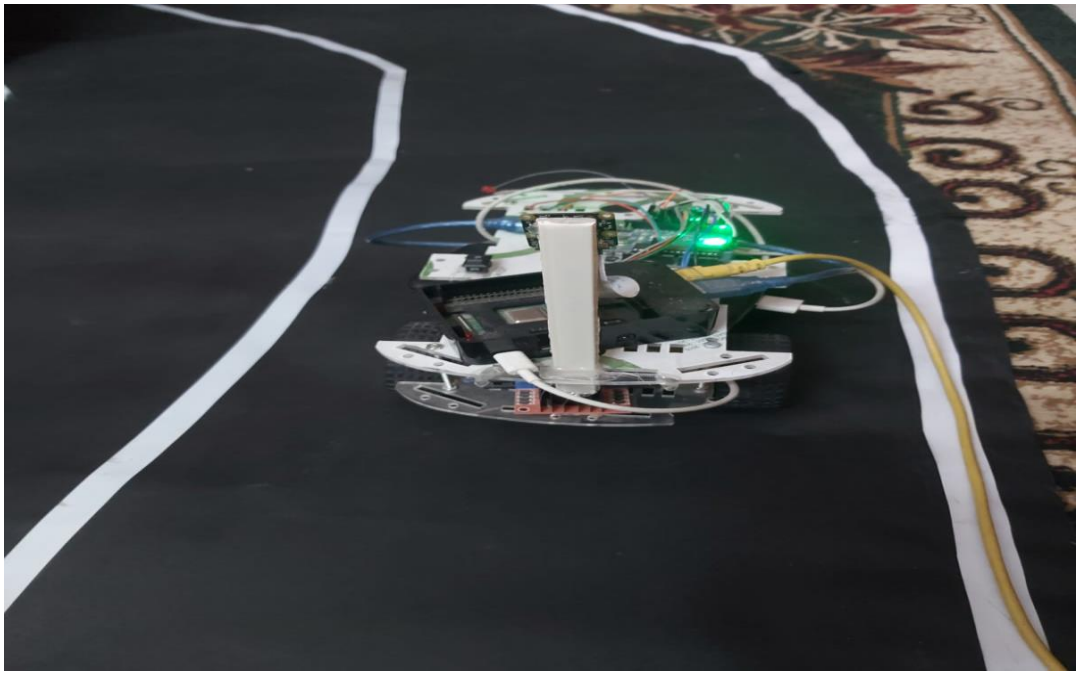

Figure 5-4: Go Left(Car)

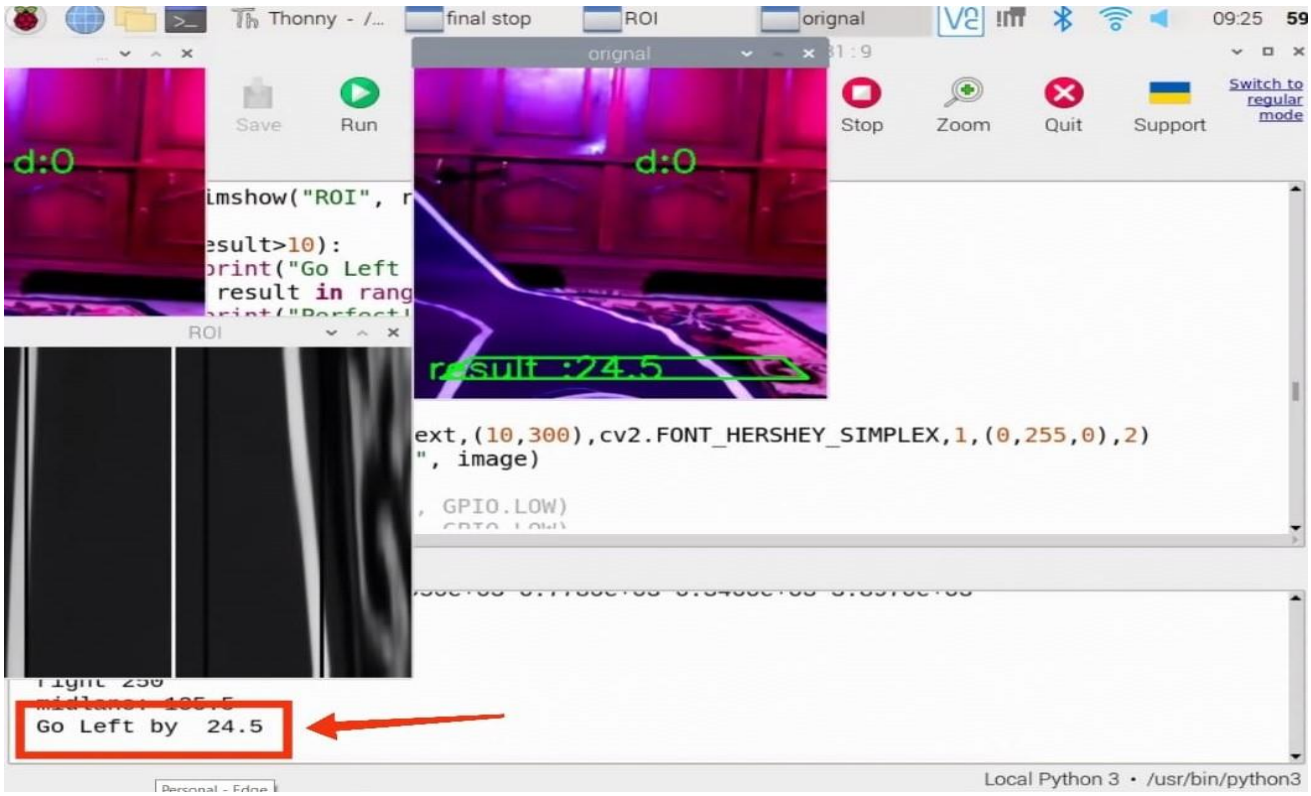


Figure 5-4: Go Left (IDE)

# 5.5 object detection

## 1. The stop sign is far from the car

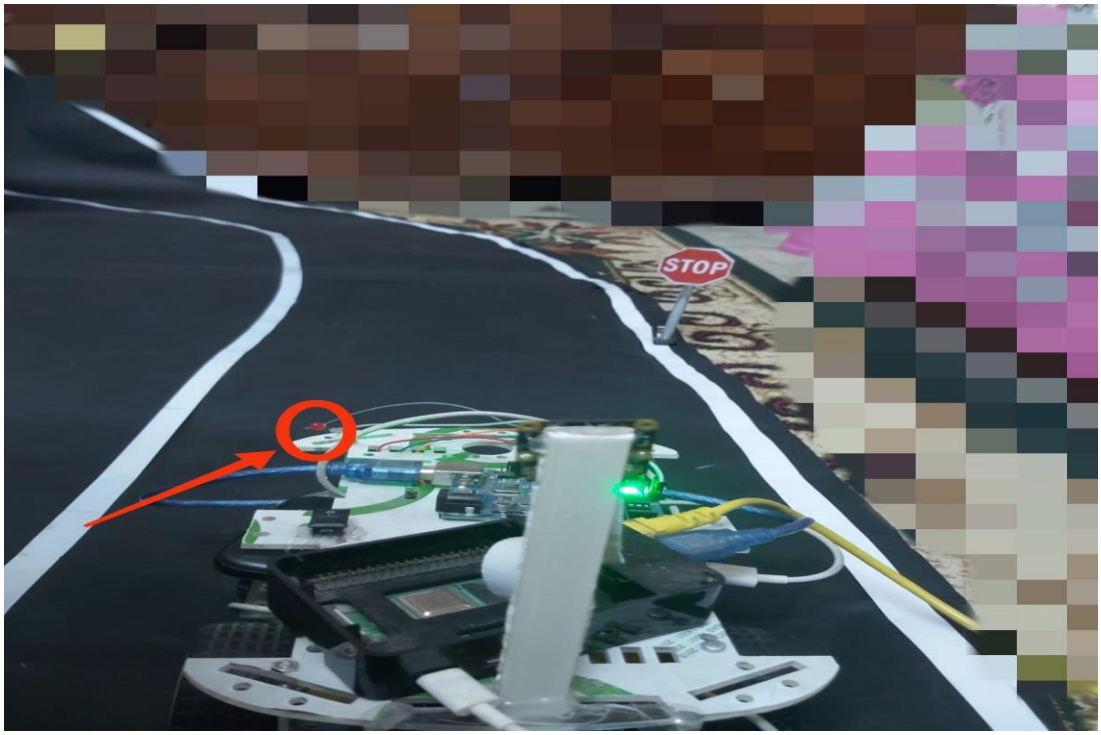


Figure 5-5: The stop sign is far from the car

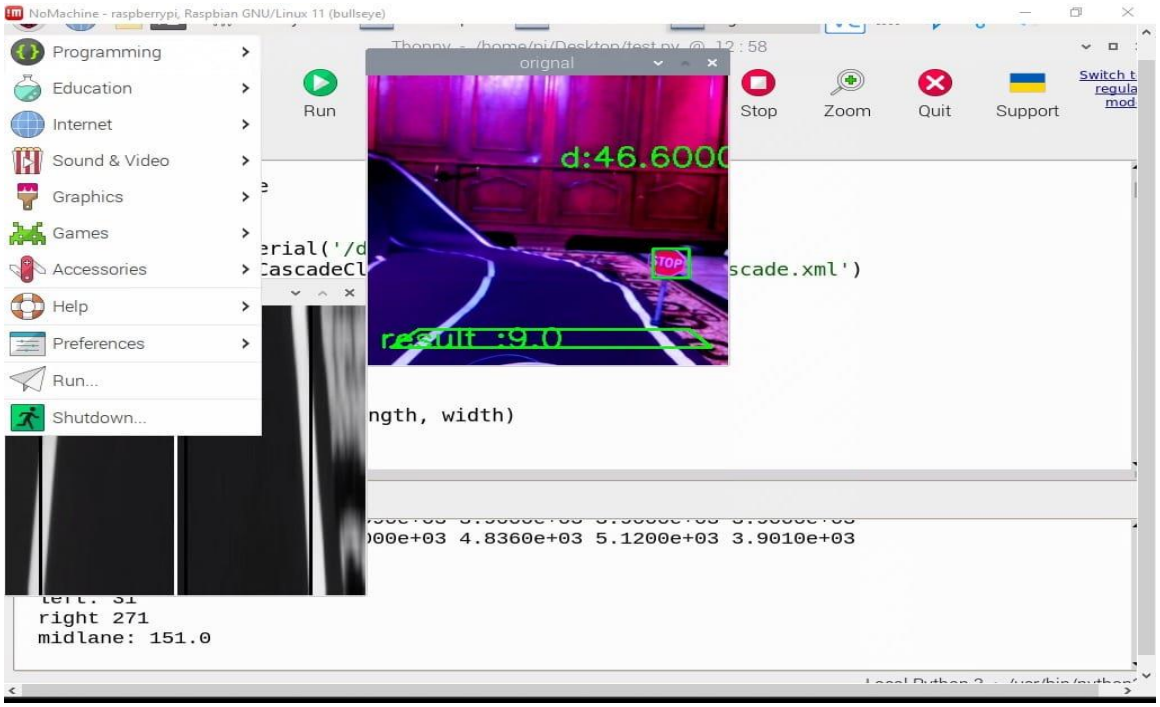


Figure 5-5: The stop sign is far from the car (IDE)

## 2. Stop sign near the car :

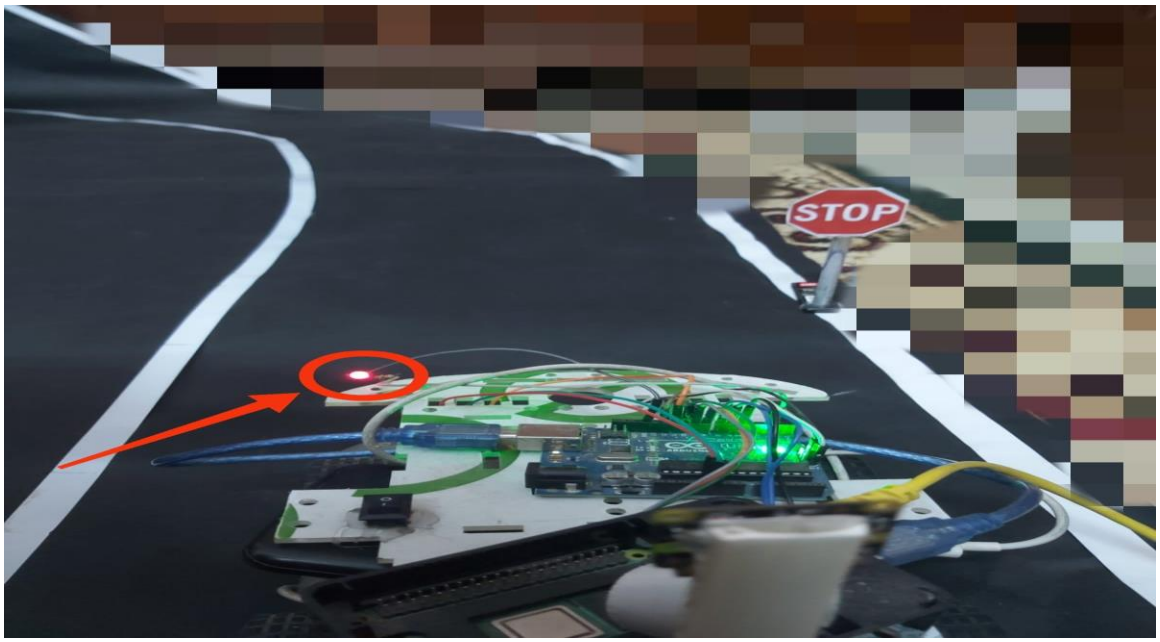


Figure 5-5: The stop sign near the car

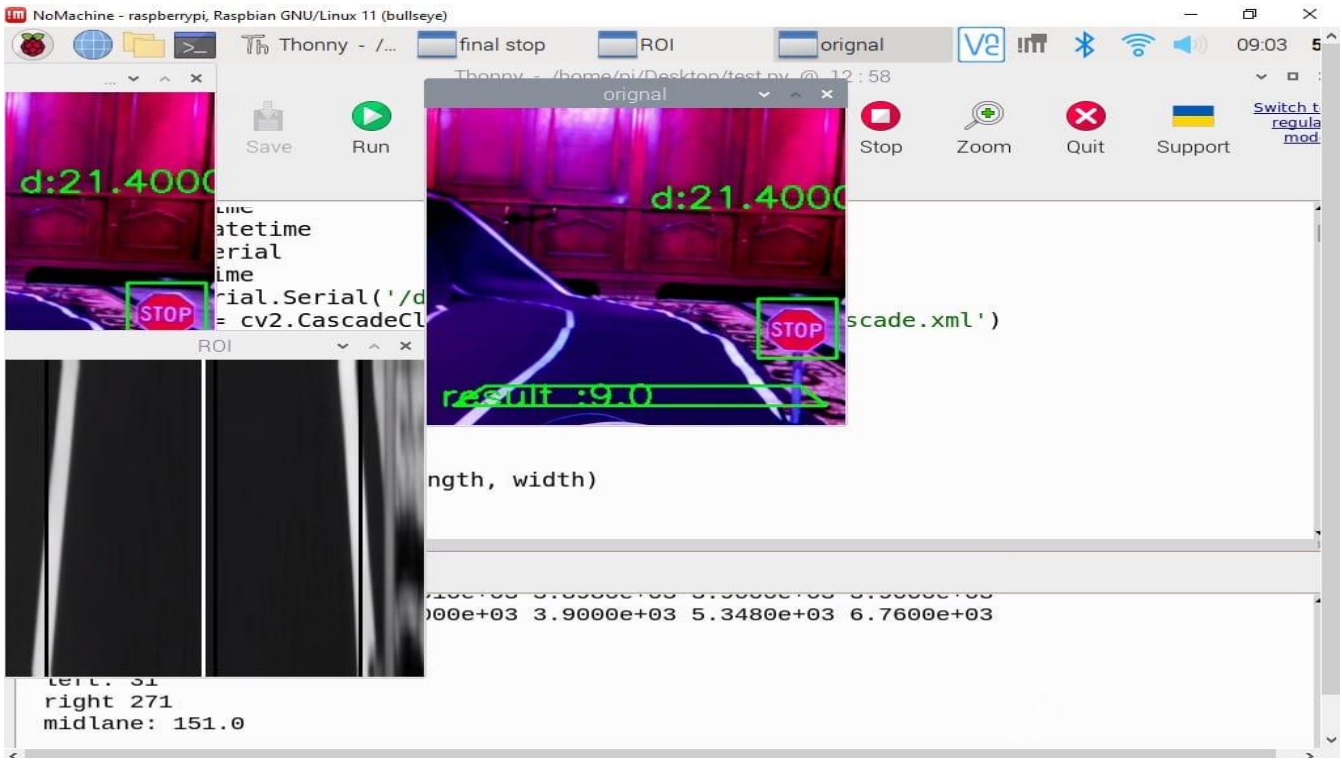


Figure 5-5: The stop sign near the car (IDE)

# Chapter 6

# Conclusion and Future Work

## 6-1 Conclusion:

In this project we used computer vision, deep-Learning, machine-Learning to develop self-driving car project that was directed for solving problem faced during driving to prevent accidents happening from human error and eases driving for those with special needs.

These technologies provided us with some modules which helped us in getting our project done. In our project we used image processing techniques such as canny edge algorithm, and machine learning technique such as haar cascade.

The result in the documentation shows that our result is that our car model can drive and go forward itself. It can understand the curve on the road and behave accordingly and take decisions for turning left or right. In-addition the self-driving system can detect stop signs and react accordingly.

## 6-2 Future work:

The future work planned for our project can be composed of three ideas:
1- driving backward which can done by setup another camera looking back in the car.
2- Automatic parking which when search a little bit found that it can

# REFERENCES

# REFERENCES:


[1] Vighnesh, Ganesh , Hritish , Gaurav (2021). Lane Detection Techniques using Image Processing. Department of Electronics Engineering, Ramrao Adik Institute of Technology, Mumbai University, India. ITM Web of Conferences 40, 03011.

[2] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1, I-511.

[3] R. Ltd, "Raspberry Pi OS – Raspberry Pi", Raspberry Pi, 2022. [Online]. Available: https://www.raspberrypi.com/software/ .

Last retrived on: 4 / 7 / 2023.

[4] "Which Raspberry Pi should you choose for your project?", Opensource.com, 2022. [Online]. Available: https://opensource.com/life/16/10/which-raspberry-pi-should-you-choose-your-project .

Last retrived on: 4 / 7 / 2023.

[5] https://components101.com/modules/l293n-motor-driver-module .

Last retrived on: 4 / 7 / 2023.

[6] https://www.tecmint.com/nomachine-an-advanced-remote-desktop-access-tool/ .

Last retrived on: 4 / 7 / 2023.

[7] Etechnophiles.com. 2022. [online] Available at: Raspberry Pi 4 GPIO Pinout, Specs, Schematic (Detailed board layout) (etechnophiles.com)

Last retrived on: 4 / 7 / 2023.

[8] www.javatpoint.com. 2022. Arduino UNO - JavaTpoint. [online] Available at: https://www.javatpoint.com/arduino-uno

Last retrived on: 4 / 7 / 2023.

[9] https://www.researchgate.net/publication/326672984_Static_Loop_Parallelization_Decision_Using_Template_Metaprogramming . Last retrived on: 4 / 7 / 2023.

[10] https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/self-parking-car.htm . Last retrived on: 4 / 7 / 2023.