



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS**

**A Major Project Final Report
On
Mini Self-Driving Car**

Submitted By:

Arjun Poudel (THA075BEI008)

Ganesh Tamang (THA075BEI017)

Hemant Sharma (THA075BEI018)

Nabin Paneru (THA075BEI025)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2023



**TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
THAPATHALI CAMPUS**

**A Major Project Final Report
On
Mini Self-Driving Car**

Submitted By:

Arjun Poudel (THA075BEI008)
Ganesh Tamang (THA075BEI017)
Hemant Sharma (THA075BEI018)
Nabin Paneru (THA075BEI025)

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics,
Communication and Information Engineering

Under the Supervision of
Associate Professor Er. Shanta Maharjan

March, 2023

DECLARATION

We hereby declare that the report of the project entitled “**Mini Self-Driving Car**” which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Electronics, Communication and Information Engineering**, is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Arjun Poudel (Class Roll No: THA075BEI008)

Ganesh Tamang (Class Roll No: THA075BEI017)

Hemant Sharma (Class Roll No: THA075BEI018)

Nabin Paneru (Class Roll No: THA075BEI025)

Date: March, 2023

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a major project work entitled “**Mini Self- Driving Car**” submitted by **Arjun Poudel, Ganesh Tamang, Hemant Sharma** and **Nabin Paneru** in partial fulfillment for the award of Bachelor’s Degree in Electronics, Communication and Information Engineering. The Project was carried out under special supervision and within the time frame prescribed by the syllabus. We found the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor’s degree of Electronics, Communication and Information Engineering.

Project Supervisor

Associate Professor Er. Shanta Maharjan

Department of Electronics and Computer Engineering, Thapathali Campus

External Examiner

Project Co-ordinator

Mr. Umesh Kanta Ghimire

Department of Electronics and Computer Engineering, Thapathali Campus

Mr. Kiran Chandra Dahal

Head of the Department,

Department of Electronics and Computer Engineering, Thapathali Campus

March, 2023

COPYRIGHT

The author has agreed that the library, Department of Electronics and Computer Engineering, Thapathali Campus, may make this report freely available for inspection. Moreover, the author has agreed that the permission for extensive copying of this project work for scholarly purpose may be granted by the professor/lecturer, who supervised the project work recorded herein or, in their absence, by the head of the department. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, IOE, Thapathali Campus in any use of the material of this report. Copying of publication or other use of this report for financial gain without approval of the Department of Electronics and Computer Engineering, IOE, Thapathali Campus and author's written permission is prohibited.

Request for permission to copy or to make any use of the material in this project in whole or part should be addressed to department of Electronics and Computer Engineering, IOE, Thapathali Campus.

ACKNOWLEDGEMENT

To begin, we'd like to thank the Institute of Engineering (IoE) for including this Major Project on the Bachelors in Electronics, Communication and Information course syllabus.

We also want to express our gratitude to our department, Department of Electronics & Computer Engineering, Thapathali Campus, for providing us with outstanding project guidance and a great learning environment, as well as for letting us use this Major Project to demonstrate and advance our abilities.

We will surely gain technical and teamwork skills from this project experience, and it will also significantly help us achieve our goals in the future.

Arjun Poudel (Class Roll No: THA075BEI008)

Ganesh Tamang (Class Roll No: THA075BEI017)

Hemant Sharma (Class Roll No: THA075BEI018)

Nabin Paneru (Class Roll No: THA075BEI025)

ABSTRACT

Self-Driving cars are the currently emerging smart cars, with the potential of observing its environment to be driver less, efficient, and crash avoidant. The concept of our project was sparked by the recent boom in the automated vehicle industry. We have built a low- cost prototype of Self-driving car which will use camera to collect the data from the real world, and then rely on computer vision, machine learning, and a variety of other programs to interpret and react to that data. Our mini self-driving car is safe and efficient and follows lane path, calculate distance of the obstacle, avoid all the obstructions in its path, detect and follow the traffic light and stop sign using Raspberry Pi, DC Motors, Raspberry Pi camera and other hardware components for the algorithms of mapping, and obstacle detection. The entire system is capable of making accurate decisions.

Keywords: *Artificial Intelligence, Image Processing, Lane Detection, Machine Learning, Self-driving Cars, Traffic Lights.*

Table of Contents

DECLARATION.....	i
CERTIFICATE OF APPROVAL	ii
COPYRIGHT.....	iii
ACKNOWLEDGEMENT.....	iv
ABSTRACT	v
List of Figures.....	v
List of Tables	vii
List of Abbreviations	viii
1. INTRODUCTION.....	1
1.1 Background	1
1.2 Motivation.....	2
1.3 Problem Definition.....	3
1.4 Project Objectives	4
1.5 Project Scope and Applications.....	4
1.6 Report Organization.....	5
2. LITERATURE REVIEW	8
3. REQUIREMENT ANALYSIS.....	15
3.1 Hardware Requirements.....	15
3.1.1. Raspberry Pi 4B.....	15
3.1.2 Raspberry Pi Camera	16
3.1.3 DC Motor	16
3.1.4 L298N Motor Driver.....	17
3.1.5 Arduino Uno	18
3.1.6 Ultrasonic Sensor.....	19
3.1.7 Lithium-ion Battery.....	20
3.1.8 Buck Converter	20
3.2 Software Requirements	21
3.2.1 Visual Studio Code.....	21
3.2.2 Google Colab	22
3.2.3 PyTorch.....	22
3.2.4 Python	23
3.2.5 YOLOv5	23
3.3 Feasibility Analysis	24

3.3.1 Technical Feasibility.....	24
3.3.2 Economic Feasibility	24
3.3.3 Operational Feasibility.....	24
3.3.4 Safety Feasibility	24
4. SYSTEM ARCHITECTURE AND METHODOLOGY	25
4.1 System Block Diagram.....	25
4.1.1 Block Diagram of Multi Processing.....	25
4.1.2 Block Diagram of Procedural Processing.....	27
4.2 Flowchart	29
4.2.1 Flowchart for Arduino	29
4.2.2 Flowchart for Raspberry Pi	31
4.3 Working Algorithm.....	33
4.3.1 Yolo Algorithm	33
4.3.2 Residual Blocks	33
4.3.3 Bounding Box Regression	34
4.3.4 Intersection Over Union	35
4.3.5 Combination of these Three Techniques	36
4.4 Overall Hardware Architecture	37
5. IMPLEMENTATION DETAILS.....	40
5.1 Camera Calibration	40
5.1.1 Measuring Distortion.....	40
5.1.2 Finding Corners.....	41
5.1.3 Calibrating Camera	42
5.1.4 Distortion Correction.....	43
5.2 Perspective Transform	44
5.3 Lane Finding	45
5.4 Lane Curvature.....	47
5.5 Voltage Regulation.....	49
5.6 Procedural Processing in Raspberry Pi 4B.....	50
5.7 Power and Steering.....	51
5.8 Traffic light and Stop sign.....	52
5.9 Speed Limits	54
5.10 Obstacle Detection and Distance Calculation	56
5.11 Design of the Car.....	58
5.12 Operational Design Domain	59
6. RESULTS AND ANALYSIS	61

6.1 Software Results	61
6.2 Hardware Results	70
6.3 Challenges Faced and Possible Remedies.....	72
7. FUTURE ENHANCEMENTS	74
8. CONCLUSION	75
9. APPENDICES	76
Appendix A: GPIO Pinout Diagram.....	76
Appendix B: Hardware Specifications	78
Appendix C: Budget Estimation	80
References.....	81

List of Figures

Figure 3- 1: Raspberry Pi 4B	15
Figure 3- 2: Raspberry Pi Camera.....	16
Figure 3- 3: DC Motor	17
Figure 3- 4: L298N Motor Driver	18
Figure 3- 5: Arduino Uno.....	19
Figure 3- 6: Ultrasonic Sensor	19
Figure 3- 7: Lithium-ion Battery.....	20
Figure 3- 8: Buck Converter	21
Figure 4- 1: Multiprocessing in Raspberry Pi.....	25
Figure 4- 2: Procedural Processing in Raspberry Pi	27
Figure 4- 3: Flowchart of Operation in Arduino.....	29
Figure 4- 4: Flowchart of Operation in Raspberry Pi	31
Figure 4- 5: Division of Image into Grids.....	33
Figure 4- 6: Bounding Box Regression.....	34
Figure 4- 7: Intersection Over Union	35
Figure 4- 8: Image Detection using YOLO	36
Figure 4- 9: Hardware Architecture	37
Figure 5- 1: Chessboard and Distorted Chessboards	40
Figure 5- 2: Mapping Distorted Points to Undistorted Points	41
Figure 5- 3: Finding Corners.....	42
Figure 5- 4: Points in a Distorted and Undistorted Image	43
Figure 5- 5: Histogram Peaks for Lane Finding	46
Figure 5- 6: Sliding Window for Lane Finding	47
Figure 5- 7: Detect Lines and Determine Curvature.....	48
Figure 5- 8: Circuit Diagram For Buck Converter.....	49
Figure 5- 9: Stop Sign	53
Figure 5- 10: Traffic Lights	54
Figure 5- 11: Speed Limit 20	54
Figure 5- 12: Speed Limit 80	55
Figure 5- 13: Distance Calculation using Ultrasonic Sensor	56
Figure 5- 14: Model of Mini Self-Driving Car	58

Figure 6- 1: Finding Corners of a Chess Board	61
Figure 6- 2: Transformed Image	62
Figure 6- 3: Transformed S Channel and Gradient Thresholds	63
Figure 6- 4: Combined S Channel and Gradient Thresholds	64
Figure 6- 5: Stacked Thresholds	65
Figure 6- 6: Histogram of Transform S channel and Gradient Threshold	66
Figure 6- 7: Lane Detection using Sliding Window	66
Figure 6- 8: Crop Image of Lane Detected using Sliding Window	67
Figure 6- 9: Multiprocessing in Raspberry Pi	68
Figure 6- 10: Procedural Processing in Raspberry Pi	69
Figure 6- 11: Stop Sign Detection.....	70
Figure 6- 12: Car Following Lane.....	71
Figure 9- 1: Raspberry Pi 4B GPIO Pinout Diagram.....	76
Figure 9- 2: Arduino Uno GPIO Pinout Diagram.....	77

List of Tables

Table 9- 1: Raspberry Pi 4B Specifications	78
Table 9- 2: Arduino Uno Specifications	79
Table 9- 3: Budget Estimation	80

List of Abbreviations

2D	Two-dimensional
3D	Three-Dimensional
AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolution Neural Network
CPU	Central Processing Unit
CV	Computer Vision
GPS	Global Positioning System
GPU	Graphics Processing Unit
ML	Machine Learning
MP	Megapixel
RAM	Random-Access Memory
RGB	Red, Green and Blue
OS	Operating System

1. INTRODUCTION

1.1 Background

The vehicle industry has become a vital component of the global economy and has seen remarkable growth and development over the years. From the early days of steam-powered vehicles to the introduction of petrol and diesel engines, the industry has continuously evolved to meet the demands of consumers. Now, it seems that electric propulsion will be the future of this business, as the need for more sustainable and eco-friendly modes of transportation increases. The progress in vehicle production has allowed for the creation of faster and more convenient vehicles that can make our daily lives easier. However, with the increasing number of cars on the road, the frequency of accidents has also tragically increased. In our fast-paced world where vehicle usage has become a norm, it's imperative that safety measures and regulations are put in place to minimize the risks associated with driving. The vehicle industry will continue to play a crucial role in shaping our future, and it's exciting to see what new innovations and advancements will emerge in the years to come.

The transportation sector is undergoing a major transformation with the advent of self-driving cars, which have the potential to greatly reduce the number of accidents caused by human error. With their compact design, self-driving cars take up less space on the road, contributing to reduced traffic congestion and the likelihood of accidents. The human element has long been a significant factor in transportation, but technology has been continuously developed to assist people in becoming more efficient. This can be seen in features like autopilot in airplanes and cruise control in vehicles, which make decisions for the operator. The next big breakthrough in transportation is expected to be self-driving cars, which are expected to have a wide-ranging social impact, beyond just the technology itself. Over the next few years, the widespread adoption of self-driving cars could revolutionize the way people move from place to place, leading to a safer, more efficient, and more sustainable transportation system. The potential benefits of self-driving cars are immense, and they are expected to be a major disruptive force in the years to come.

1.2 Motivation

The self-driving technology has become one of the most hotly debated topics in the automotive industry, with numerous companies vying to incorporate autonomous driving capabilities into their production vehicles. The main driving force behind this project is the desire to enhance vehicle safety and efficiency, reduce the number of traffic accidents, free up people's time, and decrease carbon emissions. The reduction in traffic accidents is the most significant advantage of self-driving vehicles. Human error, such as distracted or drunk driving or poor decision making, is responsible for over 90% of all road accidents, which result in the loss of over 1.3 million lives annually. With self-driving cars, decisions are made by the vehicle, eliminating the risk of human error. The vehicles also have the ability to communicate with one another, reducing the likelihood of accidents caused by miscommunication. This technology has the potential to make our roads much safer, while also providing people with more time to focus on other tasks or activities. Additionally, the reduced carbon emissions will help to combat climate change and preserve the environment for future generations. In conclusion, the self-driving technology has the potential to revolutionize the transportation industry, and the potential benefits are immense. The widespread adoption of self-driving cars has the potential to lead to a safer, more efficient, and more sustainable transportation system, benefiting individuals and society as a whole.

The use of automobiles has long been a source of significant greenhouse gas emissions, contributing to climate change and posing a threat to the environment. Despite numerous efforts to address this problem, none of them have proven to be particularly effective. However, the advent of electric self-driving technology has the potential to solve all of these issues. By transforming our miniature model into a reality, self-driving cars can help to make our roads much safer and more efficient. With their ability to drive in a more guarded and organized manner, self-driving cars have the potential to increase productivity and make transportation safer for everyone. By reducing the number of traffic accidents and minimizing the risk of human error, self-driving cars are poised to revolutionize the transportation industry, leading to a safer, more efficient, and more sustainable future. The widespread adoption of electric self-driving technology has the potential to address many of the pressing issues facing the transportation sector today, making it an exciting and promising area of development.

1.3 Problem Definition

In today's fast-paced society, road accidents have become a major issue and are happening all over the world on a daily basis. These accidents are mainly caused by human error and carelessness, such as driving under the influence of drugs or alcohol, texting while driving, exceeding the speed limit, and not paying attention to the road. These types of reckless behavior not only put the lives of the drivers and passengers at risk, but also the lives of other innocent people on the road. Furthermore, automobiles have a significant impact on the environment and contribute significantly to pollution and greenhouse gas emissions. The excessive use of non-renewable resources, such as gasoline, for automobiles is causing a major energy crisis. This is because the demand for gasoline continues to increase as more and more cars are being used on the roads, which is leading to a depletion of the world's fossil fuel resources. Additionally, the vehicles themselves are not very efficient, and the transportation costs, maintenance requirements, and traffic congestion are all high. The traffic congestion, in particular, is causing severe problems in cities, as roads are becoming increasingly clogged with cars, leading to longer commutes and increased air pollution. The lack of public transportation options in many areas is also exacerbating the problem. As a result, it is imperative that we find sustainable and environmentally friendly alternatives to traditional automobile use.

All of the issues mentioned above are likely to be solved by the introduction of self-driving cars. With the advent of advanced technology, it is now possible to develop low-cost self-driving car prototypes that can greatly increase safety, convenience, reliability, and efficiency on the roads. In our project, we are proposing the development of a low-cost self-driving car and the car will be outfitted with a Raspberry Pi, camera, sensors, and DC motors to make appropriate decisions. The Raspberry Pi will act as the brain of the car and will be responsible for processing information from the sensors and cameras in real-time. The sensors and cameras will help the car to understand its surroundings and make decisions based on the data it receives. The DC motors will control the movement of the car and ensure that it follows the correct path. This combination of advanced technology will make it possible for the car to safely and efficiently avoid obstacles, lane following, detect stop sign and traffic lights and make decisions in real-time, making it much safer for all road users.

1.4 Project Objectives

The project's primary objectives are as follows:

- A mini self-driving car that can follow lane, detect and calculate the distance of the obstacles, follow stop sign and traffic lights in its pathway.
- Hands-on experience with autonomous vehicle control system.

1.5 Project Scope and Applications

The primary goal of developing a mini self-driving car is to create a small-scale autonomous vehicle capable of performing tasks such as obstacle avoidance and detection. This project aims to combine various aspects of engineering, such as electrical, mechanical, and software engineering, in order to create a functional prototype. This project's application is to demonstrate the capabilities of self-driving technology and to serve as an educational tool for students to learn about the development of autonomous systems. The mini self-driving car could also be used in the research and development of autonomous technologies, particularly in small-scale applications such as drones and robots.

The major scope and applications of our projects can be as follows:

- Distribution of the goods across different areas.
- Movement in highways or heavy traffic roads.
- Design and development of a small scale autonomous vehicle.
- Testing and validation of the self-driving functionality in various scenarios.
- Research and development in the fields of robotics, AI, and autonomous vehicles.
- Demonstration and showcase of technology for industries, institutions, and events.
- Testing and prototyping of new control algorithms and sensor configurations.

1.6 Report Organization

The introduction chapter provides an overview of the project report for a Mini-Self Driving Car using an embedded system and Raspberry Pi camera to follow the lane path, avoid obstacles, detect and follow traffic lights and stop signs. The background section discusses the necessity of autonomous vehicles, which can reduce human error and enhance road safety. The motivation section discusses the reasons for undertaking this project, such as the increasing market demand for autonomous vehicles. The section on problem definition outlines the obstacles that must be overcome, such as the complexity of real-world traffic scenarios. The section on project objectives describes the project's aims, such as achieving precise lane, stop sign, and traffic light following. Lastly, the project scope and applications section discuss the potential applications of the self-driving car, such as highway travel and goods distribution.

The literature review chapter provides a summary of the existing research on autonomous vehicles. It discusses the various technologies and methods used in autonomous vehicles, including machine learning, computer vision, and sensors. This review examines the history of self-driving cars, the current state of the art, and future directions for research. It also discusses the advantages and challenges of autonomous vehicles, such as enhancing road safety, reducing traffic congestion, and addressing ethical concerns. The review examines the strengths and weaknesses of different methods to identify the most suitable for the project. Additionally, the literature review helps identify gaps in the existing literature, where the project can provide novel contributions. Several studies and research papers that have contributed to the development of self-driving cars are cited and their findings and limitations are discussed.

The chapter on requirement analysis discusses the project's hardware and software requirements. The hardware requirements section lists the components necessary to construct the autonomous vehicle, such as the Raspberry Pi 4B, Raspberry Pi Camera, DC Motor, L298N Motor Driver, Arduino Uno, Ultrasonic Sensor, Lithium-ion Battery, and Buck Converter. This section describes each component and its function in great detail. The software requirements section lists the software tools and libraries necessary to develop the autonomous vehicle, such as Visual Studio Code, Google

Colab, PyTorch, and Python. The feasibility analysis section addresses the project's technical, economic, operational, and safety viability. It discusses the technical obstacles that must be overcome, such as accurate object detection in real-world scenarios. It also discusses the project's economic viability, including the cost of the components and the potential revenue from selling the self-driving vehicle. The section also discusses the operational and safety viability of the project, including the usability and safety of the autonomous vehicle.

The system architecture and methodology chapter discusses the autonomous vehicle system's overall architecture and the development methodology. The system block diagram section provides an overview of the autonomous vehicle system's various components, including the Raspberry Pi, camera, motor driver, and Arduino. In addition, the communication protocols between these components are discussed. The procedural processing and multiprocessing sections describe the various methods used to process the data generated by the self-driving car system, including the use of multiple threads or processes. The flowchart sections provide a visual representation of the various data processing stages in the autonomous vehicle system. The working algorithm section concludes with a discussion of the machine learning algorithms used to detect and classify objects, such as the YOLO algorithm, as well as the techniques used to improve its accuracy, including residual blocks and bounding box regression.

The implementation details chapter is the most extensive section of the report, describing the development and integration of various components of the mini self-driving car. In this chapter, the process of camera calibration is broken down in detail. You will learn how to measure distortion, locate corners, calibrate the camera, and correct distortion using the information provided in this chapter. Additionally, the chapter delves into the inner workings of the perspective transform and the lane finding. It offers a more in-depth explanation of how the system connects the Arduino with hardware and provides serial processing on the Raspberry Pi 4B. The report describes the operation of the power steering and the power brakes, as well as how the system recognizes stop signs, traffic lights, obstacles, and speed limits.

The chapter results and analysis will discuss the software and hardware outcomes of the mini self-driving car. The software outcomes will include the precision of the lane

following algorithm, the obstacle avoidance system, and the detection of traffic lights and stop signs. The hardware outcomes will include the motor and steering system's performance and the embedded system's dependability. In addition, the chapter will discuss the obstacles encountered during the development process and possible solutions. The analysis will assess the project's efficacy and limitations, as well as its potential impact on the future of autonomous driving technology.

The Future enhancement chapter discusses potential improvements to the mini autonomous vehicle. The chapter discusses potential enhancements to the hardware components, including the camera and motor drivers. The chapter also discusses potential software component enhancements, including the lane detection algorithm and the obstacle detection algorithm. The chapter also discusses about the introduction of more power-efficient components and more compact designs. The chapter concludes by emphasizing the addition of enhancing features, such as emergency braking and backup systems, to the mini self-driving car.

The conclusion chapter will summarize the project's key findings and discuss their implications. This chapter will discuss the significance of the mini self-driving car and its potential societal impact. In addition, the chapter will reflect on the project's achievements, limitations, and future directions. In addition, the conclusion will emphasize the significance of self-driving technology as a transformative force in the automotive industry as well as its potential social benefits.

The final chapter Appendices of the project report will provide technical information regarding the Mini Self-Driving Car project in great detail. Appendix A will contain a GPIO pinout diagram illustrating the layout of the project's pins. The specifications of the project's hardware are listed in Appendix B. This will include the model and specifications of the Raspberry Pi board and Arduino. The budget estimate for the project will be included in Annex C. This will include a breakdown of the hardware and other materials used for the project. This will be useful for readers who wish to replicate or modify the project for their own purposes, as they will be able to estimate the project's costs and plan accordingly. Also, it shows the schematic diagram of integration of whole system.

2. LITERATURE REVIEW

Self-driving cars, also known as autonomous or driverless cars, are a cutting-edge technology in the automotive industry that offers a new level of comfort and safety for drivers. Self-driving cars are software-based vehicles that run on their own, relying on complex algorithms and advanced sensors to navigate roads and respond to changing traffic conditions. These vehicles have the capability to take control of the driving process if they observe any signs of driver drowsiness or inattentiveness, providing an extra layer of safety for passengers. Self-driving cars are a game-changer in the automotive industry, providing a unique solution to many of the challenges that traditional human-driven cars face. From reducing the likelihood of accidents caused by human error to providing a more convenient and stress-free driving experience, self-driving cars are revolutionizing the way we think about transportation. With the continued development of self-driving car technology, the future of the automotive industry looks bright, and the possibilities for self-driving cars are nearly endless.

Nepal, unfortunately, has one of the highest death rates caused by road accidents in the world. The statistics provided by the government paint a grim picture, with an average of 2,500 people losing their lives every year in road accidents. These accidents not only result in a high number of fatalities but also lead to a large number of people being injured, with some even suffering from permanent disabilities. This high rate of road accidents in Nepal is a cause of great concern not just for the government but also for the families and communities affected by these accidents. The government and various organizations are taking active steps to address this issue and make the roads of the country safer. This is being achieved through a combination of improving road safety, raising awareness about the dangers of reckless driving and providing better medical facilities for accident victims. Despite the efforts being made, much more needs to be done in order to reduce the number of road accidents and make the roads of Nepal safer for all its citizens [1].

Four individuals, Dr. T. Manikandan, J. S. Jayashwanth, S. Harish, and K. N. Harshith Sivatej, developed a self-driving car that has the ability to move between two points on a map, detect and recognize obstacles, identify lanes, avoid collisions, and call for help in an emergency situation. This car is equipped with sensors that allow it to sense its

surroundings and navigate through traffic and other obstacles with minimal human input. The development of this car has the potential to bring about a revolution in transportation for people with disabilities and provide independence to blind people by allowing them to travel without assistance [2]. Utilizing hardware like a Raspberry Pi, LDR sensor, ultrasonic sensor, vibrating sensor, camera, GPS, motor driver, and DC motor, the project was successfully completed. The project overall required a lot of expensive hardware parts, and the system was somewhat complicated. In contrast to their project, ours uses an ultrasonic sensor, an Arduino, and a Raspberry Pi camera. Additionally, since our project won't have a specific destination and will initially only be operating in a small area, we are not using GPS. If possible, we may, however, add GPS in the future.

A team of six students from Bangladesh University of Engineering and Technology created a car that was capable of recognizing traffic signals and making the appropriate turn. To implement this system, the analyzer computer was wirelessly connected to the car's body, allowing it to analyze the video feed frame by frame. This cutting-edge technology was designed to make driving safer and more efficient by reducing the need for human intervention. The car's ability to recognize traffic signals and respond accordingly was a major breakthrough in the field of autonomous vehicles, and demonstrated the incredible potential for advanced technologies to improve the driving experience. This project was a testament to the innovation and creativity of the students involved, and showed that even students from a developing country could make a significant impact on the world of technology. The creation of this car capable of recognizing traffic signals was a major step forward in the development of self-driving cars, and had the potential to change the way that people think about transportation in the future [3]. A group of students has developed a self-driving car prototype that was designed with a combination of different components including an Arduino Uno, a Esp 8266, a Motor Driver L298 Dc motor, a Camera, a Traffic Light, and a Chassis. The primary purpose of the project was to create a self-driving car that would be able to navigate roads and traffic conditions on its own, without human intervention. The prototype designed by this group of students used the Arduino Uno as its central processing unit, while the Esp 8266 was used to provide wireless connectivity. The Motor Driver L298 Dc motor was used to control the movement of the car's wheels, while the Camera was responsible for capturing images of the environment. The Traffic

Light was used to detect traffic signals, and the Chassis was used to support the car's other components. This prototype was designed to be a self-contained system that could be operated without the need for an external computer. However, our project differs from this one in a couple of key areas. Firstly, we use an on-board Raspberry Pi instead of a separate computer to process images, which provides a more integrated and streamlined solution. Secondly, their prototype has fewer instructions than ours, which has more instructions and incorporates a machine learning algorithm for further self-sustaining improvements. The machine learning algorithm allows the self-driving car to make decisions based on previous experiences and continuously improve its performance over time. This results in a more advanced and intelligent self-driving car prototype that has the potential to revolutionize the field of autonomous vehicles.

Another approach to self-driving cars involves the use of machine learning algorithms to detect the feature sets of raw frames. The developer creates a dataset of images and their corresponding steering angles, and trains the model using this data. Based on the trained model, the steering angle is predicted by comparing each image with the path. This approach has some limitations, however. It can only make predictions for a path that has already been followed and trained, and cannot navigate new areas, making it less versatile. Additionally, the accuracy of this approach is highly dependent on the quality of the training data and the complexity of the feature sets used for prediction. Despite these limitations, this approach has gained popularity in the development of self-driving cars, and researchers are actively exploring ways to improve its accuracy and versatility. Some developers are using a method of lane detection that involves Hough transformation. In this method, the raw image is first converted into edges using cv2's built-in canny method, and lines are created using the collection of collinear points obtained through the Hough lines method. These lines are then drawn over the image. Based on the knowledge gained from this literature study, it can be concluded that a model plan can be created that uses the first method of image thresholding to achieve clear edge detection. This approach allows developers to effectively detect the lanes in an image and use this information to guide the self-driving car. The use of Hough transformation in lane detection is a promising method that has been shown to be effective in real-world applications. However, it also has some limitations, such as its dependence on the quality of the image and the accuracy of the edge detection. Despite these limitations, the development of this method continues to advance, and researchers

are exploring ways to improve its performance and make it more robust for use in self-driving cars [4].

A team of experts consisting of Pravel Kumar, Shivam Shandilya, and their friends was responsible for the development of a self-driving car using soft computing. The project aimed to create an autonomous car that was equipped with a sophisticated algorithmic "brain" and powerful sensors. To achieve this, the team leveraged the latest advancements in artificial intelligence and machine learning, with the main focus being on the incorporation of Deep Q-learning in the car's design. The first step in the development process was to create an environment and develop the car's brain using Deep Q-networks. The developed brain was then trained each time the program was executed, with the training occurring as the car performed various actions in the environment. This approach allowed the self-driving car to learn and adapt over time, becoming more efficient and capable of navigating complex environments and making decisions on its own. The ultimate goal of the project was to create a self-driving car that would be safe, reliable, and able to navigate various environments without human intervention. The innovative approach used in this project has the potential to be a game-changer in the field of autonomous vehicles and demonstrate the tremendous advancements that are being made in the development of self-driving car [5].

A convolutional neural network (CNN) was trained by a small group of NVIDIA Corporation employees to convert the unprocessed pixels from a single front-facing camera into steering commands. With the least amount of human training data, the system learned how to maneuver through traffic on highways, local roads with or without lane markings, and on local roads. The system automatically learned internal representations of the necessary processing steps, like spotting useful road features, with only the human steering angle as the training signal. To recognize things like road outlines, for example, they didn't specifically train it to do that. In contrast to an explicit decomposition of the issue, their end-to-end system simultaneously optimizes all processing steps, including lane marking detection, path planning, and control. They anticipated smaller systems and improved performance as a result of this. In order to maximize system performance, internal components will self-optimize rather than optimizing intermediate criteria selected by humans, such as lane detection, which will result in better performance. Such criteria are understandably selected for ease of

human interpretation, but this does not imply a guarantee of the best system performance. Smaller networks are made possible because the system learns to solve the issue in the fewest number of processing steps possible. At 30 frames per second, the system was in operation. More work was required to increase the network's robustness, discover ways to confirm its robustness, and enhance the visualization of the network's internal processing steps. This is an excellent example of self-driving car development [6].

The functioning of self-driving cars that use lane detection is based on the analysis of images captured by cameras installed on the vehicle. These cameras are used to identify the lanes by thresholding the image and calculating the direction of the lanes based on the presence of white pixels. The process starts by warping each frame of the image and then calculating the number of white pixels' column-wise. The direction of the lanes is then determined based on the column with the highest number of white pixels. While this method appears to be effective on the surface, it has some limitations that must be considered. Firstly, it can only be implemented in environments where the road is marked with white lanes, making it unsuitable for real-world road conditions where roads are often not clearly marked. The threshold values also have to be adjusted for each environment, which is not feasible on a commercial level, making it difficult to apply this method to real-world scenarios. Additionally, the technique is limited in its ability to handle different lighting and weather conditions, which can greatly affect the quality of the image and the accuracy of the lane detection process. Despite these challenges, the development of self-driving cars that use lane detection continues to advance, and researchers are actively working on developing more robust and flexible systems that can be used in a variety of environments and under different conditions [7].

According to Ginger Bonnie's "History debate on Autonomous car," the concept of self-driving cars has been around for quite some time. As early as 1939, General Motors proposed a self-driving model that was guided by radio-controlled electromagnetic fields generated by magnetized metal spikes embedded in the road. This design was later produced on a commercial level in 1958, with the car's front being equipped with coils that served as sensors. Fast forward to 1969, John McCarthy put forward his thoughts and proposed the idea of lane detection using cameras, which was a significant

step towards the development of autonomous vehicles. The history of autonomous vehicles is marked by the continuous advancement of technology, and today, the development of self-driving cars is one of the most exciting and rapidly growing fields in the automotive industry. Despite some challenges and concerns, the promise of autonomous vehicles is that they have the potential to revolutionize the way we travel, making our roads safer, reducing congestion, and improving the quality of life for people all over the world [8].

Similarly, a group of students developed the Tesla Model RC self-driving electric car, which is powered by a battery and has the ability to self-navigate. The main objective of the Tesla Model RC is to reach the destination set on an Android application using GPS navigation. The car receives data from sensors to understand its surroundings and detect obstacles in its path, allowing it to avoid them and safely reach its destination. The development of this self-driving car is a testament to the advancements being made in the field of autonomous vehicles, and it demonstrates the potential of using technology to create safer and more efficient modes of transportation. The Tesla Model RC is not only a highly innovative piece of technology, but it also represents the future of transportation and the growing importance of autonomous vehicles in our daily lives. Despite the remarkable progress that has been made in the development of self-driving cars, there is still much work to be done to perfect this technology and make it safe for widespread use. Nevertheless, the Tesla Model RC serves as a valuable contribution to the advancement of self-driving cars and highlights the potential for further progress in this field [9].

Ammar N. Abbas and Muhammad Asad Irshad proposed using machine learning algorithms to improve the edges of raw frames in image processing using canny edge detection. Their approach involved creating a dataset of image frames and corresponding steering angles, which they used to train a machine learning model. The model was then used to predict the steering angles by comparing each image frame to a previously followed path. However, this method has several limitations. Firstly, it can only be used on previously followed and trained paths, and cannot be applied to new areas. This means that every road that a driver may encounter in their daily life must be included in the training data, which can result in a significant loss of storage. Secondly, the process of comparing matrices of arrays to train the model is slow, and requires a

large amount of data to be collected. Due to these limitations, the authors believe that this method is inapplicable in practical situations [10].

C. Ma and M. Xie's research paper on lane detection outlines a method that involves thresholding an image and calculating the presence of white pixels to determine the direction of the lanes. In this method, each frame is warped, and the number of white pixels on the left and right sides are calculated. The side with the highest number of white pixels is determined by calculating the values of all pixels' column by column. If the side with the highest value has more white pixels, the vehicle turns in that direction. However, this technique is only practical in controlled environments, such as a path created using white paper or similar materials, and is not suitable for use on roads. This is because the threshold values must be adjusted for each environment, making it impractical for commercial production. [11]

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirements

3.1.1. Raspberry Pi 4B

The Raspberry Pi 4B is a powerful and versatile single-board computer that can be used for a wide range of applications, including building a mini self-driving car. With its high performance, the Raspberry Pi 4B is an ideal control unit for embedded systems and provides the necessary processing power to run the autonomous systems software. The device features two USB 3.0 ports and two USB 2.0 ports for connecting various peripheral devices, such as cameras and sensors, that are crucial for autonomous navigation. Additionally, the 40-pin GPIO header on the Raspberry Pi 4B is standard, making it easy to connect to other components and peripherals. The Raspberry Pi 4B uses a USB-C connector for power and runs on 5V DC, providing a stable and reliable source of power. It is capable of operating in a temperature range of 0 to 50 degrees Celsius, making it suitable for use in a wide range of environments. All these features make the Raspberry Pi 4B an ideal choice for building a mini self-driving car, providing a powerful and flexible platform to build and test autonomous systems.

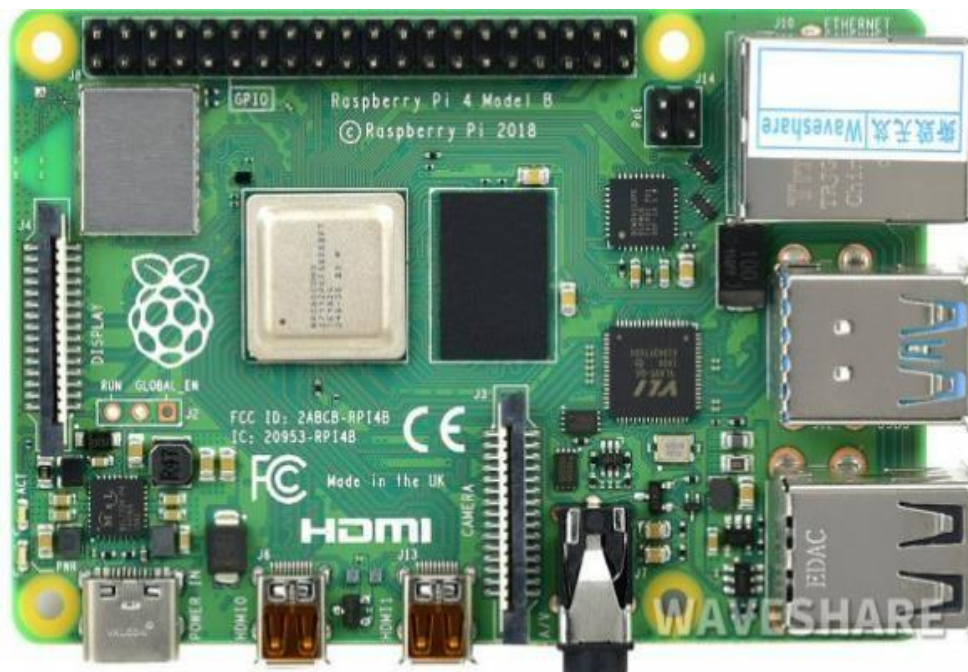


Figure 3- 1: Raspberry Pi 4B

3.1.2 Raspberry Pi Camera

The Raspberry Pi Camera Module v2 is a versatile and user-friendly camera board that can be easily attached to any Raspberry Pi board. The board is an official product from the Raspberry Pi Foundation and is designed to deliver high-quality images and videos. The camera module is equipped with an 8-megapixel Sony IMX219 image sensor that provides high-definition (HD) video and still images with excellent clarity and detail. The camera module features a fixed focus lens that has been carefully crafted to capture clear images even in low light conditions. With its small size and low power consumption, the Raspberry Pi Camera Module v2 is an ideal choice for a wide range of projects and applications.



Figure 3- 2: Raspberry Pi Camera

3.1.3 DC Motor

DC motors are commonly used for various applications due to their simple design and ease of control. They can be used for a wide range of applications, from small toys to large industrial machinery. DC motors are highly reliable, efficient, and durable, making them an excellent choice for use in demanding environments. These motors are composed of a stator, which is the stationary part of the motor, and a rotor, which is the rotating part. The stator contains the permanent magnets that create the magnetic field, while the rotor contains the windings that conduct current. When current flows through the rotor, it creates a magnetic field that interacts with the magnetic field from the stator, causing the rotor to rotate. The speed of a DC motor is determined by the voltage and

current applied to the rotor, and it can be easily controlled by changing the voltage or current. Overall, DC motors are a versatile and reliable solution for a wide range of applications.



Figure 3- 3: DC Motor

3.1.4 L298N Motor Driver

For controlling motors in miniature self-driving cars, the L298N motor driver is a well-liked option. This driver is perfect for high power motors used in small autonomous vehicles because it can control two DC motors and can handle high current. The L298N is flexible enough to be used with a variety of systems because it is designed with input logic compatibility with both 3.3V and 5V microcontrollers. Additionally, it has an integrated flyback diode that guards the driver against voltage spikes brought on by the inductive load of the motor. In order to keep the driver and motor safe even under conditions of high load, the driver also includes adjustable current protection and thermal protection. Furthermore, the L298N has an easy-to-use control interface, making it simple to integrate into your self-driving car project. Overall, the L298N motor driver is a dependable and versatile option for controlling among hobbyists and engineers alike.



Figure 3- 4: L298N Motor Driver

3.1.5 Arduino Uno

The Arduino Uno is a versatile and user-friendly microcontroller board that is popular among hobbyists, students, and professionals alike for building various electronics projects. It is based on the ATmega328P microcontroller and has a range of features that make it ideal for control and sensing applications. The Arduino Uno has 14 digital input/output pins that can be used for a variety of purposes, including reading the data from sensors and controlling the actuators of the system. In addition to digital inputs/outputs, it also has 6 analog inputs that can be used to read analog signals, such as those from sensors that measure temperature, light, and other physical parameters. The board is powered by a 16 MHz quartz crystal and can be connected to a computer through a USB connection for programming and data transfer. It also has a power jack that can be used to provide power to the board. In the context of a self-driving car project, the Arduino Uno can be used to send control commands to the car's actuators, such as its motor and steering mechanism, based on the data received from the sensors and decision-making algorithms.



Figure 3- 5: Arduino Uno

3.1.6 Ultrasonic Sensor

A device that uses high-frequency sound waves to measure the distance, velocity, or presence of objects is an ultrasonic sensor. They are widely used in mini self-driving cars for object detection and avoidance. The sensors are typically mounted on the front or rear of the car and emit high-frequency sound waves that bounce back when they encounter an obstacle. The time required for the sound waves to return is used to determine the distance between the vehicle and the obstacle. This information can then be used to control the car's speed and steering, allowing it to avoid colliding with the obstacle.



Figure 3- 6: Ultrasonic Sensor

3.1.7 Lithium-ion Battery

Lithium-ion batteries are a type of rechargeable battery that utilize lithium ions as the primary charge carrier. We are using these batteries (3 batteries with combined total of 12V) due to their high energy density, which allows them to store a large amount of energy in a small and lightweight package. They also have a long cycle life and low self-discharge rate, which makes them ideal for use in vehicles that require long-lasting power. These properties, combined with their relatively low cost, have made Li-ion batteries the most popular type of rechargeable battery for us.



Figure 3- 7: Lithium-ion Battery

3.1.8 Buck Converter

A buck converter, also referred to as a step-down converter, is an electronic circuit that converts a higher voltage to a lower voltage. It operates by turning on and off a series of transistors, which generates a series of pulses that are filtered to generate a smooth output voltage. Due to its high efficiency and small size, the buck converter is utilized in numerous applications, including power supplies, battery chargers, and LED drivers. In our miniature self-driving car, we use a buck converter to reduce the voltage output from the lithium-ion battery to 5V, the minimum input voltage required by the Raspberry Pi 4B. This ensures that the Raspberry Pi receives the correct power supply, allowing it to control the car's movements optimally.



Figure 3- 8: Buck Converter

3.2 Software Requirements

3.2.1 Visual Studio Code

Visual Studio Code is a code editor compatible with Windows, Linux, and macOS. Among the features are syntax highlighting for debugging, intelligent code completion, snippets, code refactoring, and embedded Git. In addition to customizing the theme, keyboard shortcuts, and preferences, users can install extensions that provide additional functionality. Visual studio code is source- code editor that supports Java, JavaScript, Go, Python, C++, and Fortan, among other programming languages. The majority of popular programming language have rudimentary support in visual studio code.

VS Code was used to write and edit the code that controls the movements and behavior of a miniature self-driving car. Python was used for the coding. In addition, VS Code was used to install and manage software dependencies, such as libraries and frameworks, that are required for the autonomous vehicle to function properly. This ensured that the development environment is correctly configured and that the developer has access to the necessary tools. In addition to these fundamental functions, VS Code offered a number of features that can aid in enhancing the development process for a small autonomous vehicle. For instance, VS Code includes powerful debugging tools that can be used to identify and fix code issues, as well as source control management features that can help keep track of code changes over time.

3.2.2 Google Colab

Google colab is an entirely cloud-based Jupyter notebook environment. It takes care of all your programs setup and configurations. Most significantly, it doesn't require any setup, and the notebooks you create can be changed by multiple team members at the same time, much like Google Docs pages. Many major machine learning libraries are supported by Colab, and can be loaded into your notebook with ease. As a programmer, you may use Google Colab to write and execute in python, make a list of all the code you have written that supports mathematical equations, import and export google drive notebooks, import/publish notebooks from GitHub and PyTorch, TensorFlow, Keras, and OpenCV can all be used together.

To use Google Colab for the implementation of our miniature self-driving car, we first created a new notebook in the Colab environment and then imported any required libraries or dependencies. This included computer vision and machine learning libraries, depending on the project's specific requirements. Python allowed us to write and test code for the self-driving car in the notebook environment once the necessary libraries were imported. Once the code is complete, we could save and export the notebook, then use it to control the autonomous vehicle in the real world. This required connecting the laptop to sensors, motors, and other hardware components used to control the movement and behavior of the vehicle.

3.2.3 PyTorch

PyTorch is a powerful and easy-to-use framework that allows developers and researchers to build and train complex neural networks for a wide range of applications such as computer vision, natural language processing, and robotics. PyTorch can be used to analyze real-time sensor data, such as camera feeds and make decisions based on the data in order to control a mini self-driving car. Using PyTorch, developers can construct and fine-tune neural networks to perform specific tasks, such as obstacle avoidance and lane detection, and then test them in a simulated or real-world setting. This enables the mini self-driving car to navigate the environment autonomously, making it a valuable instrument for education, research, and experimentation in the field of autonomous driving.

PyTorch was used in our implementation of a mini self-driving car to construct and train machine learning models that assisted the car in perceiving its environment and making decisions based on this perception. For instance, PyTorch was used to develop a computer vision model capable of detecting and classifying objects in the car's immediate environment, such as stop signs and traffic signs. PyTorch was also used to construct motion planning and control models, which are responsible for determining the car's trajectory and controlling its movements. These models can account for a number of variables, including the car's speed, direction, and the location of other objects in the environment.

3.2.4 Python

Python is an extremely popular programming language. It was developed by Guido van Rossum in 1991 and released at that time. It is utilized for server-side web development, software development, math, and system programming. Python is compatible with numerous platforms, including Windows, Mac, Linux, Raspberry Pi, etc. and is somewhat similar to English language. Prototyping can be very quick in python and a system can be built with fewer lines of codes compared to other programming languages.

Python was utilized in order to control the motion of and the course taken by the miniature self-driving car. Python libraries such as PySerial were utilized in order to interface with the car's various sensors, motors, and other pieces of hardware. Python libraries such as OpenCV were utilized in the process of putting computer vision algorithms into action for a variety of tasks, including object detection, lane detection, and traffic sign recognition.

3.2.5 YOLOv5

The object detection model known as YOLOv5 (You Only Look Once version 5) makes use of a deep neural network to determine the locations of specific objects within an image. It is an improvement over earlier versions of YOLO, both in terms of its accuracy and the speed at which it can draw conclusions. Because it only employs a single neural network to process an image from beginning to end, YOLOv5 is both quicker and more precise than competing object detection models.

In our mini self-driving car, YOLOv5 is capable of detecting objects in the environment of a mini self-driving car, such as obstacles and traffic lights. Because it uses YOLOv5, the car's microcontroller is able to quickly recognize objects and respond appropriately to steer the vehicle away from potential collisions or other hazards.

3.3 Feasibility Analysis

3.3.1 Technical Feasibility

The project uses resources which are affordable, and easy to implement in designs. The hardware used in our projects such as camera, Raspberry Pi, motors etc. are readily available. The picture taken by the camera will be processed by Raspberry Pi and data will be sent to servo motor which will decide the directions of the wheel. This entire process will use the concept of AI for image processing and related stuffs.

3.3.2 Economic Feasibility

We are designing our project in such a way that the system cost is as minimum as possible. Since, heavy equipment and technologies are not used, the project is cost effective. The hardware equipment used are affordable and software required are easily available. Thus, the project can be completed with minimal budget, thus making it economical.

3.3.3 Operational Feasibility

Our miniature car is operationally feasible, and we expect it to make outstanding decisions since we are confident in the hardware and software we have employed. Our proposed project has the potential to reduce the cost of a tiny self-driving automobile model while maintaining its quality. End users will be able to use our little self-driving automobiles with ease and efficiency once our project is successfully completed.

3.3.4 Safety Feasibility

Our self-driving car prototype is perfectly safe to operate on a single lane because it will be programmed to travel by avoiding all the obstacles in its route. Furthermore, our proposed model will have no negative impact on the environment or humanity. As a result, our project is safe to implement over time.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 System Block Diagram

4.1.1 Block Diagram of Multi Processing

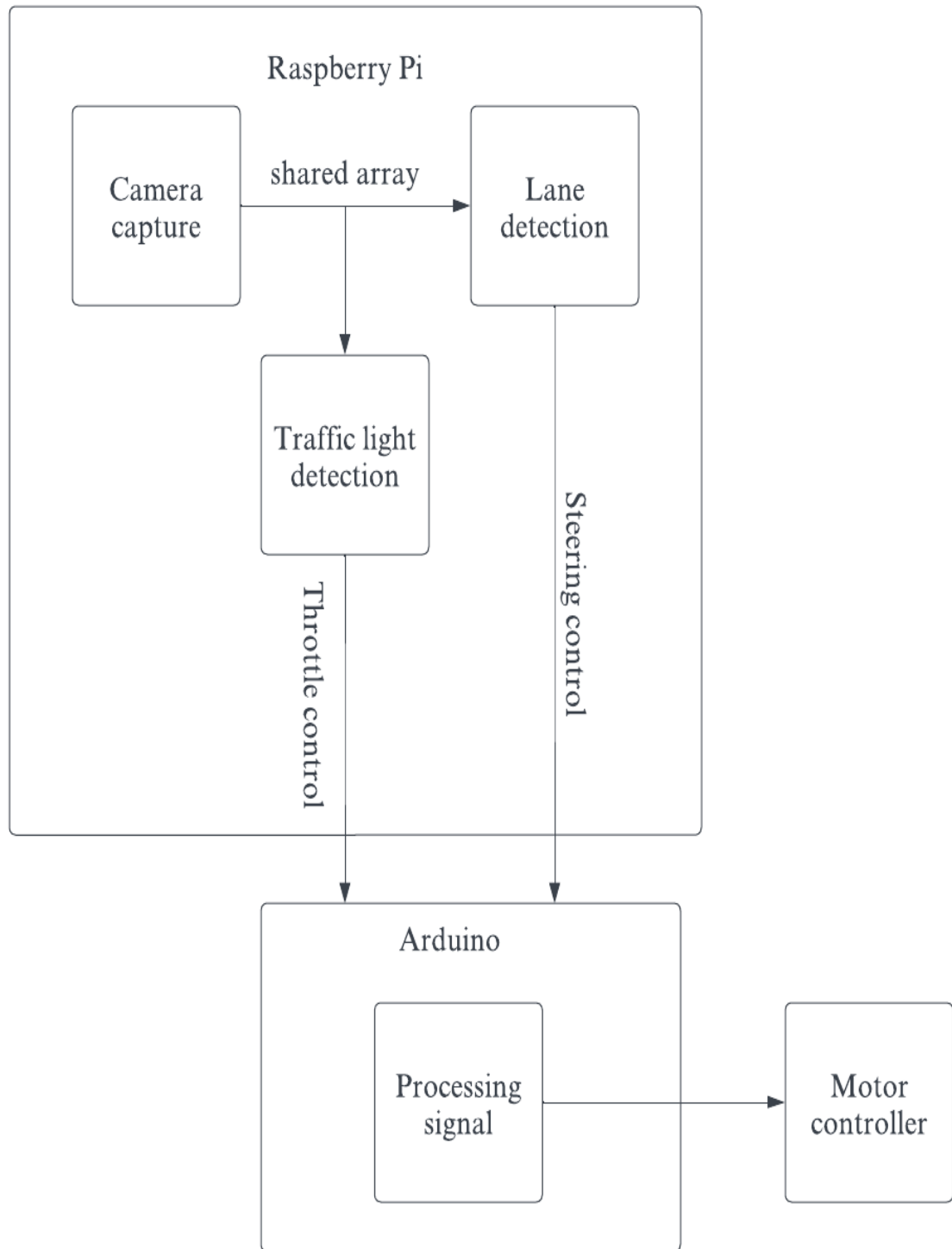


Figure 4- 1: Multiprocessing in Raspberry Pi

The figure 4-1 depicts the multiprocessing operation in our Raspberry Pi 4B including the role of a Raspberry Pi, a Arduino and Motor controller. Overall, it is clear that multiprocessing operation is done in Raspberry Pi and this, then is sent to Arduino which in turn send the processing signal to the Motor controller.

The process of self-driving car navigation involves various steps, starting with the capture of the image of the surrounding environment. The image is captured and processed simultaneously to detect the lanes on the road. This is done through a shared array that stores the image data and the lane information. Once the lanes are detected, the next step is to detect the traffic lights and distinguish between red, green and yellow signals. This information is crucial for the car to determine when to stop or go, and when to turn. The traffic light detection system sends throttle control and brake signals to the car, depending on the color of the traffic light.

In addition to traffic light detection, the lane detection system also sends steering control signals to the car, indicating which way the car should turn. These signals are received by the Arduino board, which acts as a central processor for the system. The Arduino processes the incoming signals from the traffic light and lane detection systems and sends the processed information to the Arduino controller. The Arduino controller is responsible for executing the commands from the Arduino and controlling the actuators of the car, such as its motor and steering mechanism. It uses the information from the traffic light detection system and the lane detection system to determine the speed of the car and the direction in which it should turn.

4.1.2 Block Diagram of Procedural Processing

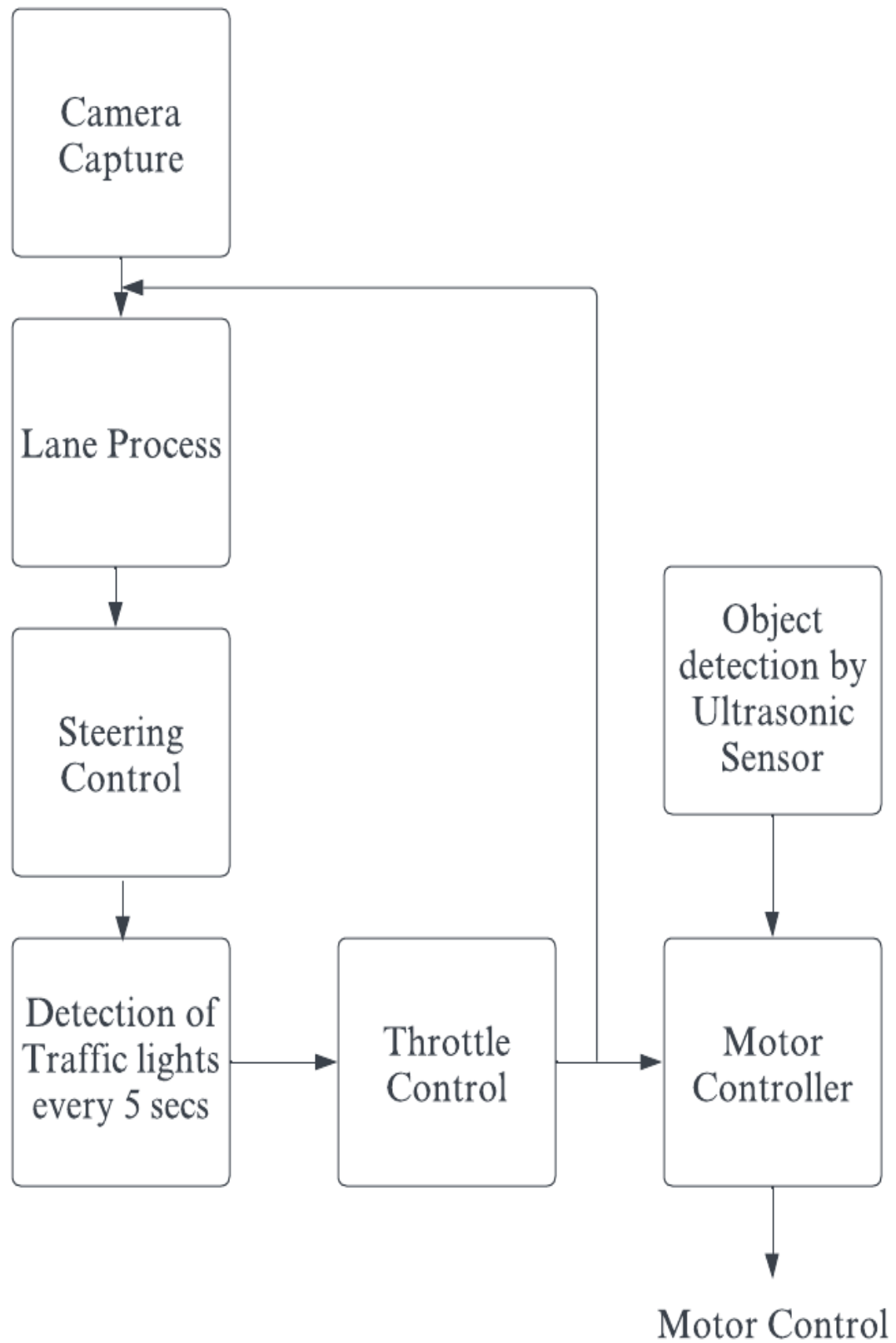


Figure 4- 2: Procedural Processing in Raspberry Pi

The figure 4-2 illustrates the block diagram of procedural programming in Raspberry Pi. Overall, it can be seen that procedural programming requires eight number of steps, beginning with the camera capture and ending with the motor control.

The Raspberry Pi Camera captures an image and then processes the lane. The processed lane information is then utilized to control the steering mechanism of the vehicle. After every five seconds, the system detects the traffic lights and controls the throttle accordingly. The motor is also controlled by the system, with the motor controller being responsible for controlling the speed and direction of the motor. Additionally, if the ultrasonic sensor detects an object near the vehicle, it sends a signal to the motor controller, which then takes the necessary actions to avoid the obstacle. The series of operations mentioned ensures the smooth and efficient functioning of the vehicle.

4.2 Flowchart

4.2.1 Flowchart for Arduino

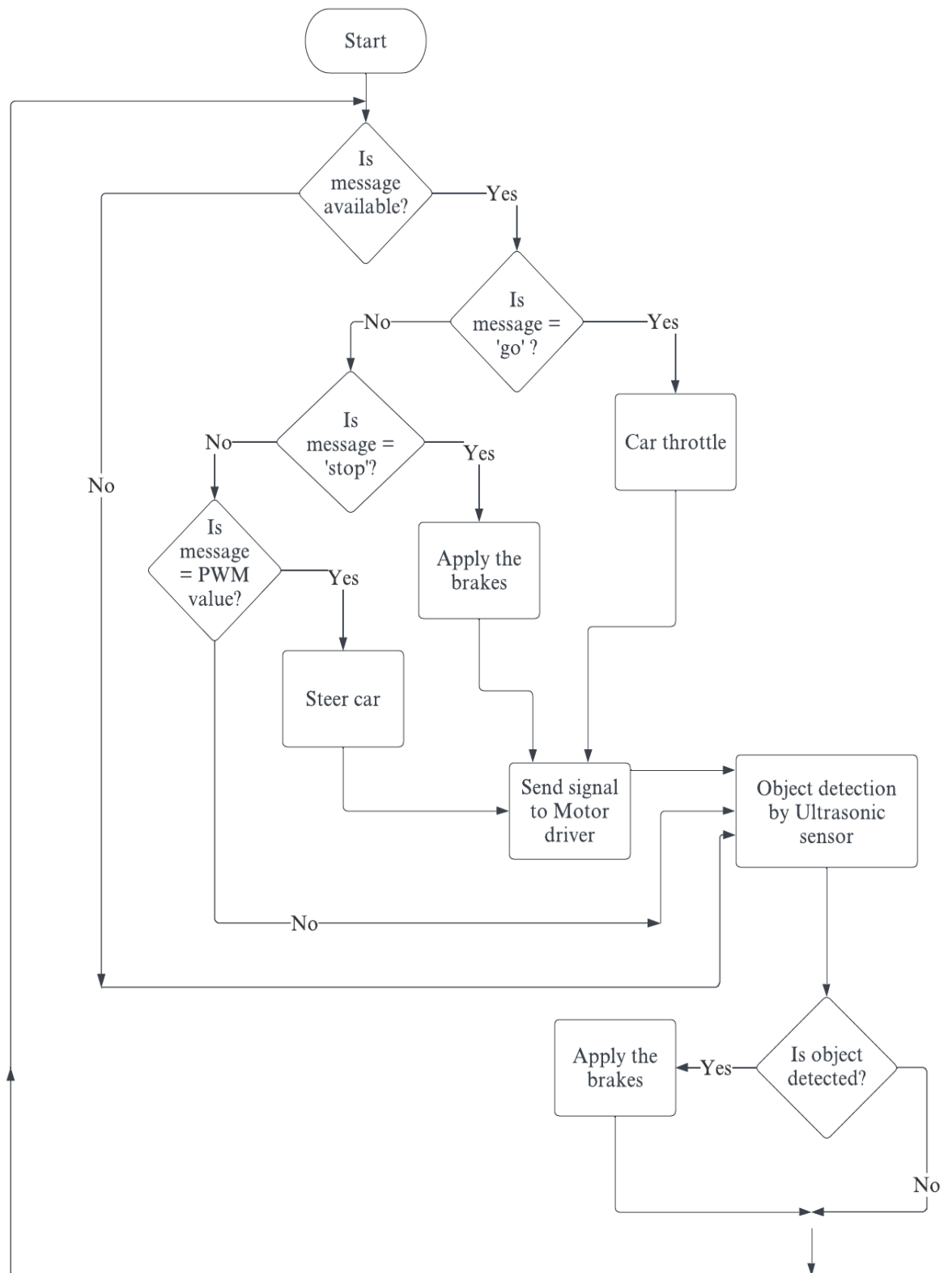


Figure 4- 3: Flowchart of Operation in Arduino

The flowchart of operation in Arduino, as depicted in figure 4-4, outlines the process involved in controlling the self-driving car. Overall, it is clear that the process of operation in Arduino is a cyclic process, and certain conditions need to be met in each process to go ahead in the flowchart.

The process begins with checking if a message is available in Arduino. If a message is available, the next step is to check if the message is "go." If the message is "go," the car is throttled. On the other hand, if the message is "no," the process moves to checking if the message is "stop." If the message is "stop," the car is braked, but if the message isn't "stop," the next condition checked is if the message contains a PWM value. If the message contains a PWM value, the car is then steered. The three signals from steering the car, braking the car, and throttling the car are all sent to the motor driver, which is then sent to object detection by an ultrasonic sensor. If the message does not contain a PWM value or if no message is available from the first testing condition, the process is sent to the object detection by ultrasonic sensor. After the object detection, if an object is detected, the car is braked, and the process initiates again from the beginning. If no object is detected, the process begins again from the start, ensuring that the self-driving car operates smoothly and efficiently. The figure 4-4 flowchart of operation in Arduino plays a crucial role in controlling and directing the actions of the self-driving car, ensuring its safe operation on the road.

4.2.2 Flowchart for Raspberry Pi

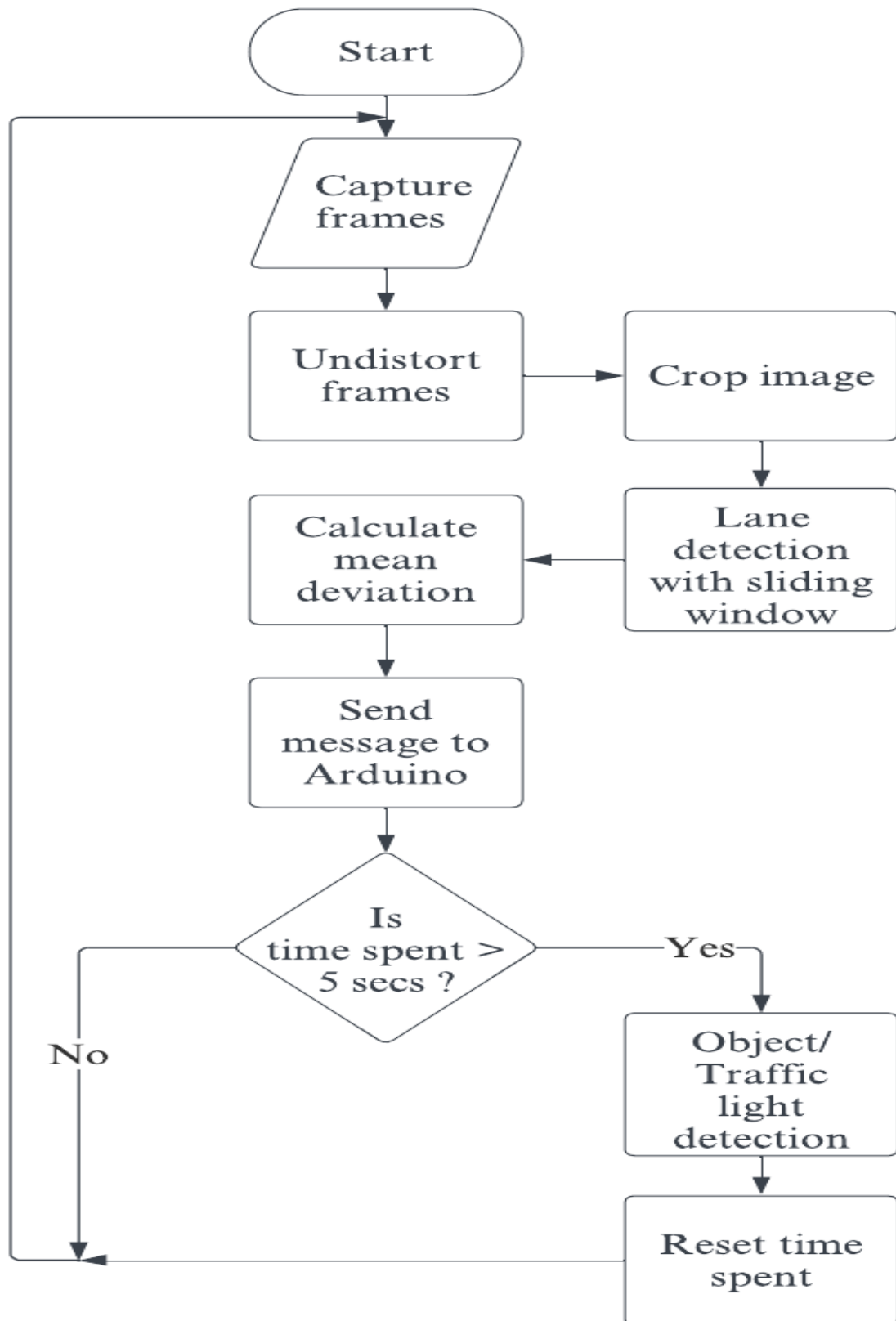


Figure 4- 4: Flowchart of Operation in Raspberry Pi

The flowchart of operation in Raspberry Pi, as depicted in figure 4-4, provides a clear visual representation of the process involved in creating a self-driving car. The flowchart outlines each step in the process, from capturing the frames using the Raspberry Pi camera, correcting the images, detecting the lane, processing the information, and performing object and traffic light detection. The process is designed to be a cyclic process that runs in a loop.

The journey towards creating a self-driving car is a multi-step process that involves capturing frames, correcting the images, detecting the lane, processing the information, and performing object and traffic light detection. It all begins with capturing the frames using a Raspberry Pi camera. The captured frames are distorted, and thus, need to be corrected through undistorting and cropping to meet specific requirements. The next step is to detect the lane by using the sliding window method and calculating the mean deviation. This information is then sent to an Arduino for further processing. Before continuing to the next stage, a check is performed to determine if the time spent is greater than or less than five seconds. If the time spent is greater than five seconds, then the process moves forward to perform object detection and traffic light detection. The time spent is then reset, and the process starts again from the beginning. However, if the time spent is less than five seconds, the process begins again from the initial stage, capturing the frames, and continuing through each step until it reaches the point where the time spent is greater than five seconds. The process of creating a self-driving car involves several intricate steps that require a high level of precision and accuracy to ensure that the car operates smoothly and safely on the road.

4.3 Working Algorithm

4.3.1 Yolo Algorithm

Yolo algorithm employs the following three procedures:

- Residual Blocks
- Bounding box regression
- Intersection Over Union

4.3.2 Residual Blocks

Grids are first created on the image. There is a $S \times S$ dimension for each grid. The grids created from an input image are displayed in the image below.

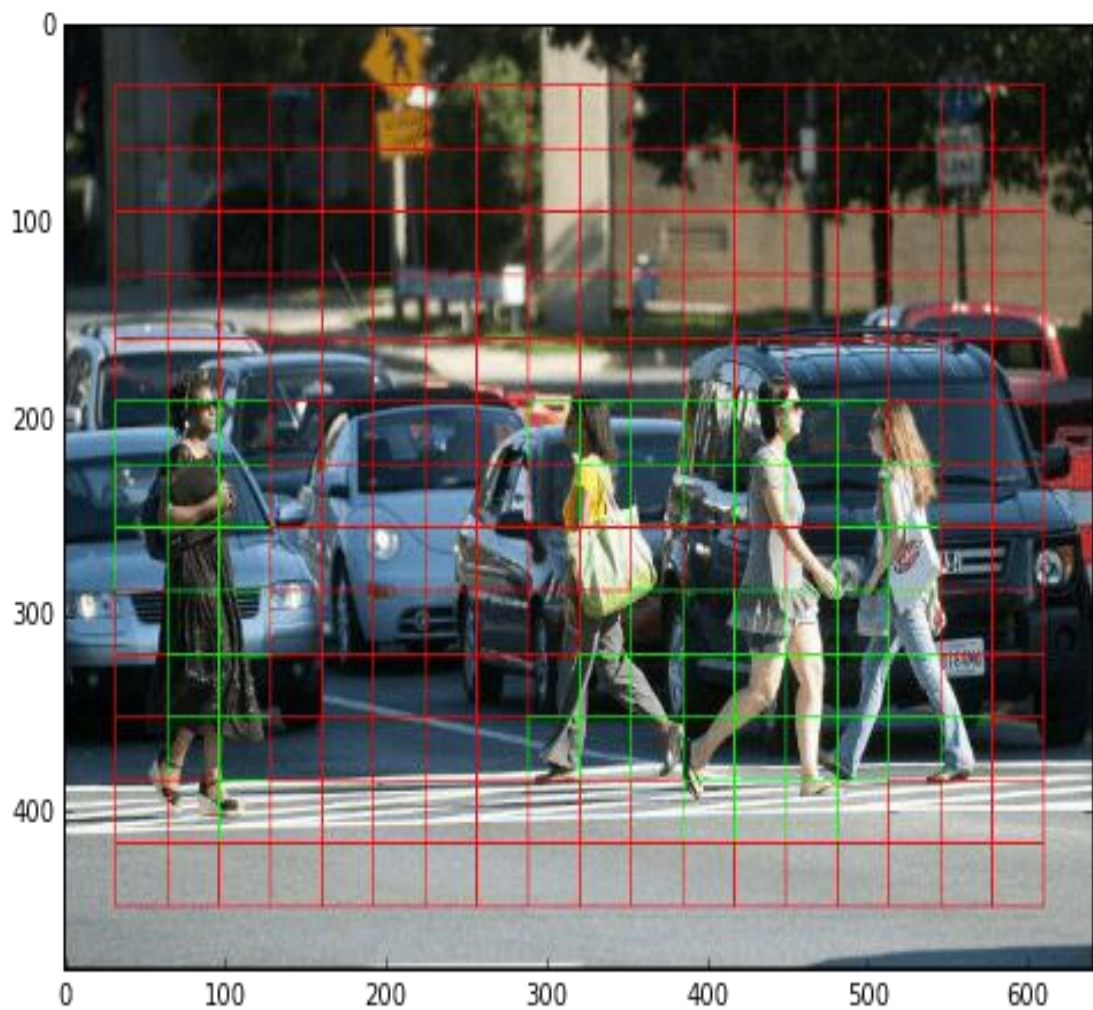


Figure 4- 5: Division of Image into Grids

The above picture is made up of many identically sized grid cells. Objects that appear within grid cells are detected by each grid cell. For example, if an object center appears within a particular grid cell, that cell will be in charge of detecting it.

4.3.3 Bounding Box Regression

The following attributes are present in every image bounding box:

Width(b_w), Height (b_h), Class (c) and Bounding box center(b_x, b_y).

The image below depicts a bounded box. The bounding box is indicated by a yellow border.

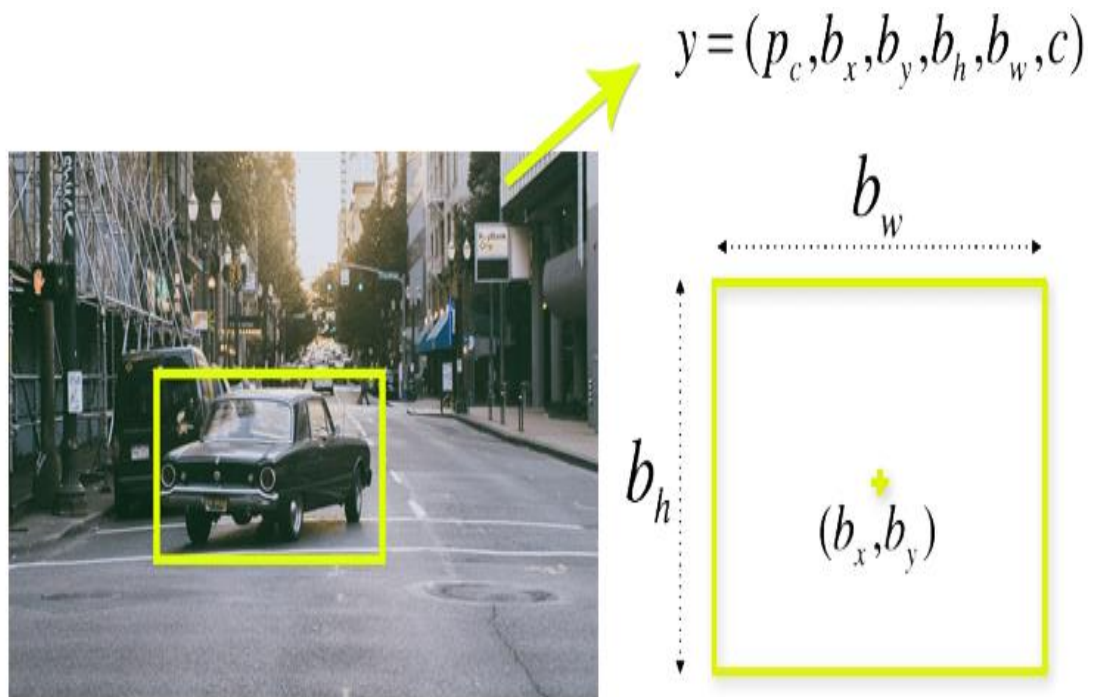


Figure 4- 6: Bounding Box Regression

YOLO utilizes a single bounding box regression to ascertain the height, width, center, and class of an object. The probability of an object appearing within the bounding box is depicted in the preceding diagram.

4.3.4 Intersection Over Union

The object detection phenomenon known as intersection over union, or box overlapping, describes this phenomenon (IOU). In order to properly enclose the items, YOLO uses IOU to build an output box.

Each grid cell is in charge of producing the anticipated bounding boxes and their confidence scores. The IOU is equal to 1 in cases where the projected and actual bounding boxes coincide. This strategy gets rid of bounding boxes that aren't the same as the actual box.

The following diagram depicts the straightforward operation of IOU:

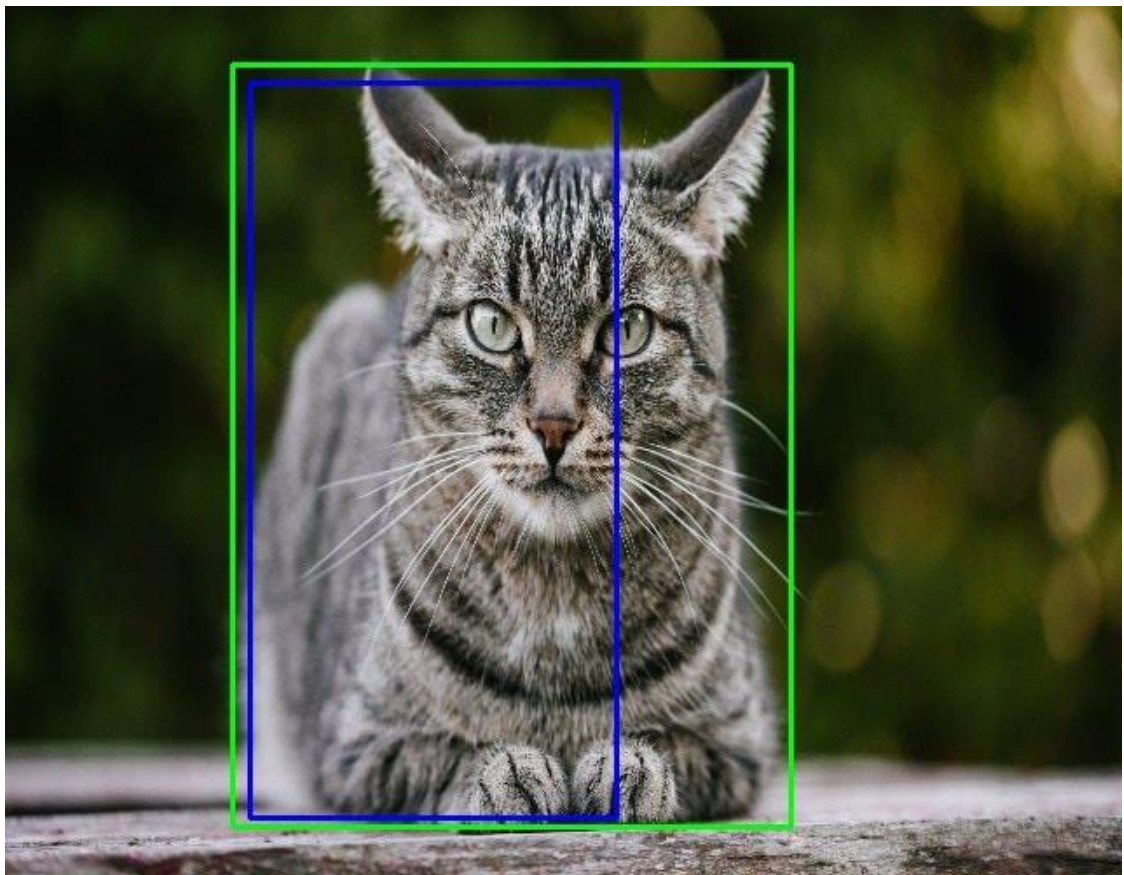


Figure 4- 7: Intersection Over Union

In the picture above, two bounding boxes can be seen, one in green and the other in blue. As opposed to the blue box, which is a prediction, the green box represents the actual box. The two bounding boxes are balanced thanks to the YOLO algorithm.

4.3.5 Combination of these Three Techniques

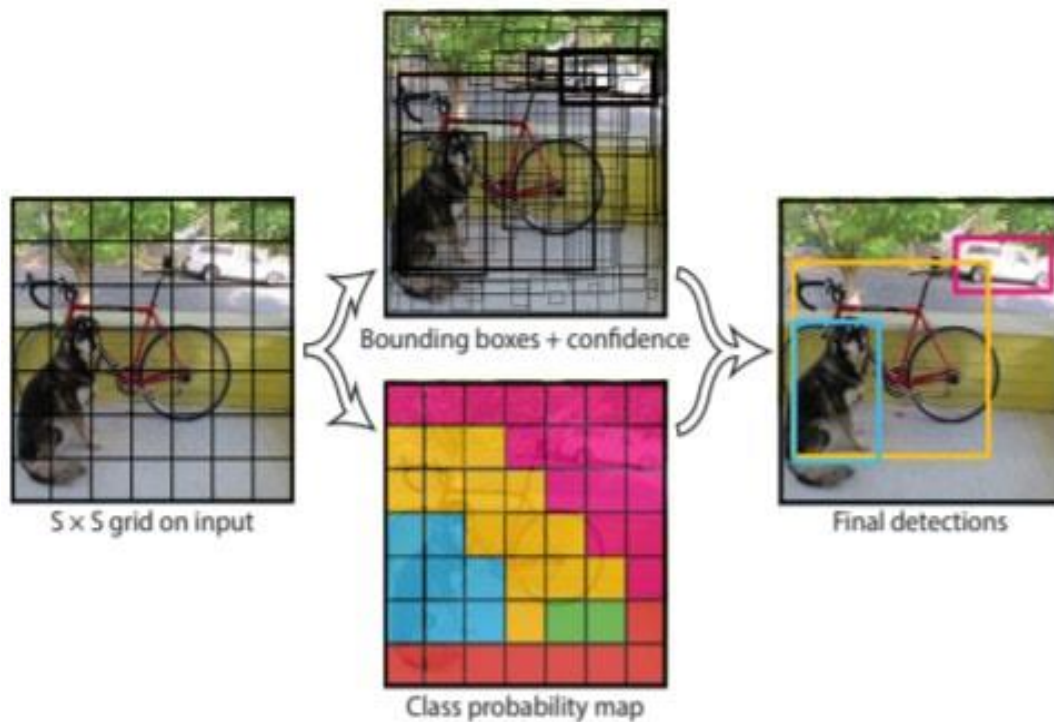


Figure 4- 8: Image Detection using YOLO

First, the picture is cut up into individual grid cells. Along with confidence ratings, B bounding boxes are predicted to exist inside of each grid cell. The cells make predictions regarding the class probability in order to identify the category that each object belongs to. At least three distinct categories of things can be represented by the following examples: a canine, a canine companion, and a car. In order to make all of the predictions at the same time, a single convolutional neural network is utilized.

It has been ensured by IOU that the projected bounding boxes are the same size as the actual boxes that the objects occupy. This phenomenon eliminates bounding boxes that aren't required for the object or that don't correspond to the characteristics of the object (like height and width). The final detections will be composed of unique bounding boxes that are tailored to fit the objects in question very precisely.

For example, the yellow bounding box encloses the bicycle, while the pink bounding box encircles the car. Both of these bounding boxes are shown below. The dog has been brought to the reader's attention by using the blue bounding box.

4.4 Overall Hardware Architecture

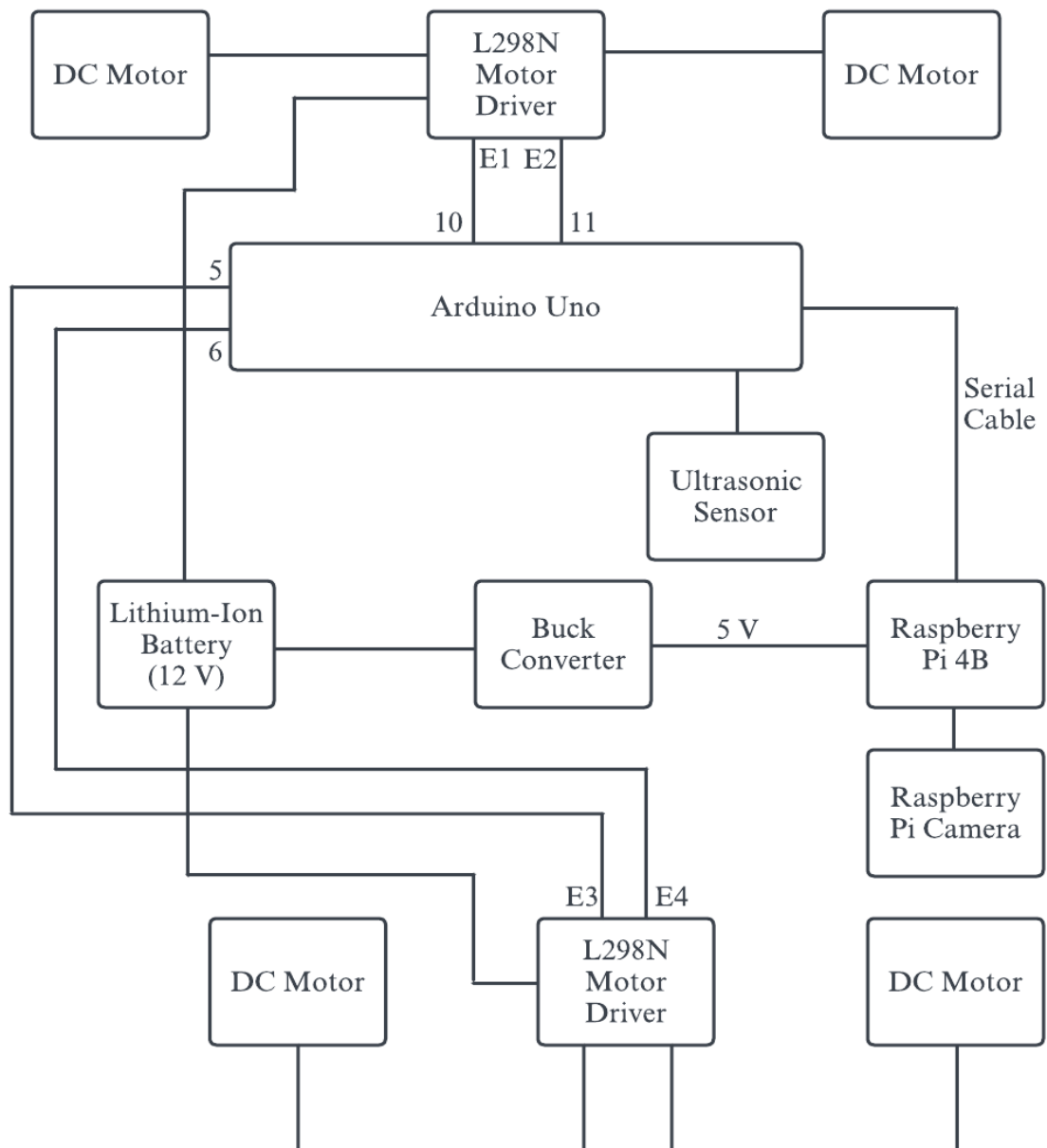


Figure 4- 9: Hardware Architecture

The above diagram depicts that the mini self-driving car's hardware architecture is comprised of multiple components that collaborate to ensure the vehicle operates smoothly and autonomously. Each component is essential to the functionality of the automobile, and the design and implementation of the architecture are crucial.

The Arduino Uno microcontroller is the first component of the mini self-driving car's hardware architecture. It is responsible for controlling the movement and operation of the automobile. The microcontroller communicates with the L298N motor drivers, which control the DC motors' speed and direction. The Arduino Uno's Pins 10 and 11 are connected to the first L298N motor driver, whereas Pins 5 and 6 are connected to the second L298N motor driver. The responsibility of each motor driver is to control the speed and direction of two DC motors.

The L298N motor driver is a crucial component of the Mini self-driving car's hardware architecture. It is responsible for controlling the speed and direction of the DC motors, which is crucial for the car's movement. The motor driver receives signals from the Arduino Uno microcontroller, which controls the movement of the vehicle by sending signals to the motor driver.

The primary source of power for the Mini self-driving car is the DC motors. They are powered by the 12-volt lithium-ion battery, which also powers the motor drivers. The use of DC motors ensures that the vehicle has sufficient power to move and operate efficiently.

The Buck convertor is another essential component of the Mini self-driving car's hardware architecture. It converts the battery's 12-volt power supply into a 5-volt power supply used to power the Raspberry Pi 4B microcontroller. It is the responsibility of the Raspberry Pi 4B microcontroller to process data from the car's various sensors and cameras.

The Raspberry Pi camera is another vital component of our mini self-driving car's hardware architecture. It captures images and videos of the vehicle's surroundings, which is necessary for navigation. The Raspberry Pi 4B microcontroller processes data from the vehicle's camera to make operational decisions.

The Raspberry Pi 4B microcontroller also powers the Arduino Uno microcontroller via a serial cable. This ensures that both microcontrollers are cooperating to control the movement and operations of the vehicle. Communication between the microcontrollers is essential for the autonomous movement and decision-making of the vehicle.

The ultrasonic sensor is another component of the Mini self-driving car's hardware architecture. It detects obstacles in the car's path and sends signals to the Arduino Uno microcontroller to redirect or stop the car. Utilizing an ultrasonic sensor enables the vehicle to navigate autonomously without human intervention.

To conclude, our mini self-driving car's hardware architecture is a complex system that requires proper design and implementation. The car relies heavily on microcontrollers, sensors, cameras, and power sources for its success. The design and implementation of the hardware architecture must be performed meticulously to ensure that the car functions effectively and efficiently. The Mini self-driving car is an intriguing project that necessitates originality, attention to detail, and ingenuity for success.

5. IMPLEMENTATION DETAILS

5.1 Camera Calibration

5.1.1 Measuring Distortion

When a camera captures a 2D image of a 3D object in the actual world, there is an imperfect transition that results in image distortion. These 3D items apparent size and shape are really altered by distortion. To measure distortion, we took a picture of the known shapes. Here we took chess board because its regular and high contrast pattern makes it easy to detect.

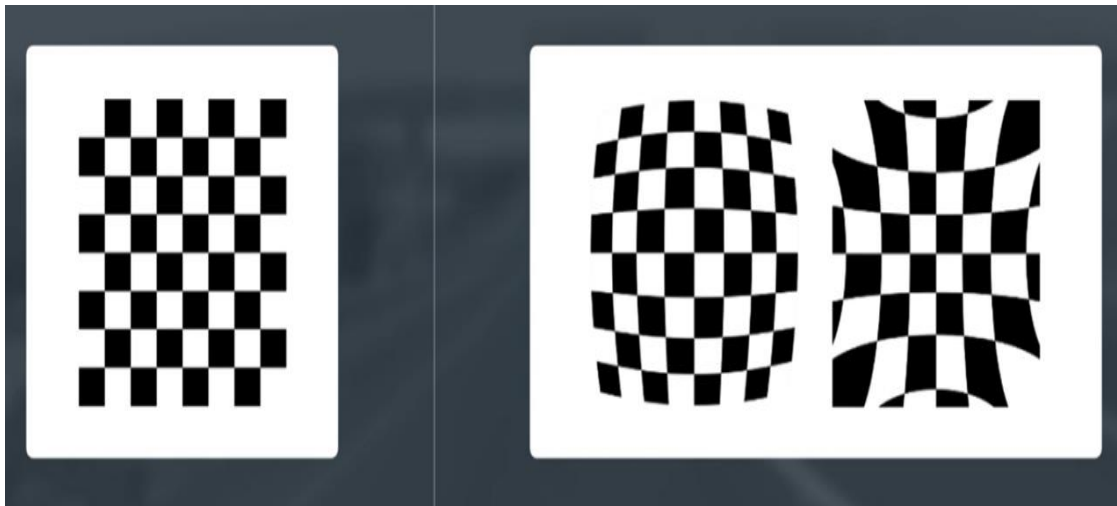


Figure 5- 1: Chessboard and Distorted Chessboards

We used our camera to take multiple picture of chess board against the flat surface, then we were able to measure any distortion by looking at the difference which means the apparent shape of size of the squares in these images as opposed to the shape and size they actually are. Then we used this information to calibrate our camera. We created a transform that mapped undistorted points into distorted points and finally undistorted any images.

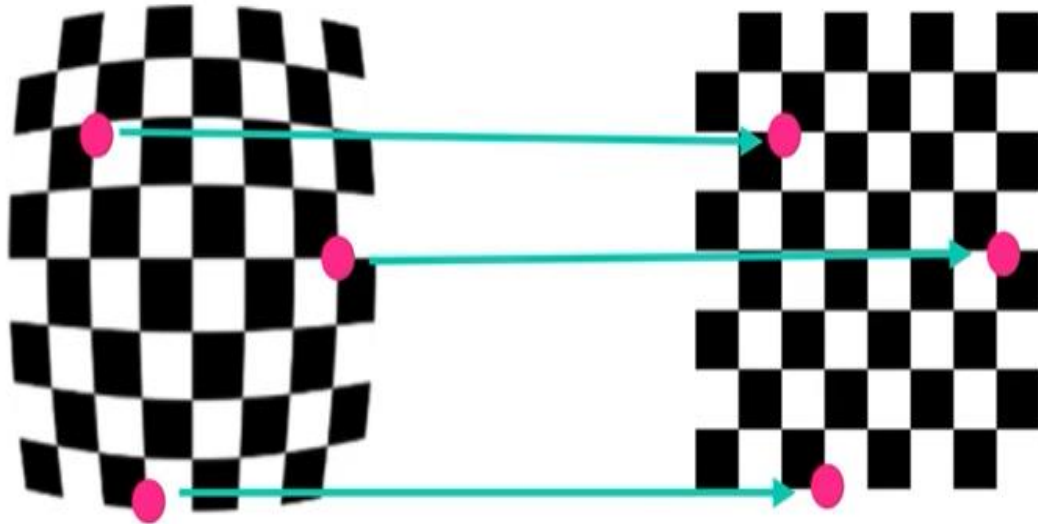


Figure 5- 2: Mapping Distorted Points to Undistorted Points

5.1.2 Finding Corners

The OpenCV library provides two powerful functions, `findChessboardCorners()` and `drawChessboardCorners()`, which are commonly used to perform tasks related to camera calibration. These functions are specifically designed to locate and draw corners in images that feature a chess board pattern. To use these functions effectively, it's important to accurately determine the number of corners in each row and column of the chess board. To do this, the number of corners in any given row should be counted and stored in the variable `nx`. Similarly, the number of corners in a given column should be tallied and stored in the variable `ny`. These values are then used as inputs for the `findChessboardCorners()` and `drawChessboardCorners()` functions, allowing them to automatically locate and draw the corners in the image. By utilizing the `findChessboardCorners()` and `drawChessboardCorners()` functions, along with a carefully counted `nx` and `ny`, it's possible to quickly and easily perform camera calibration and other related tasks.

If we use these two functions in a sample image, the outcome will look like this:

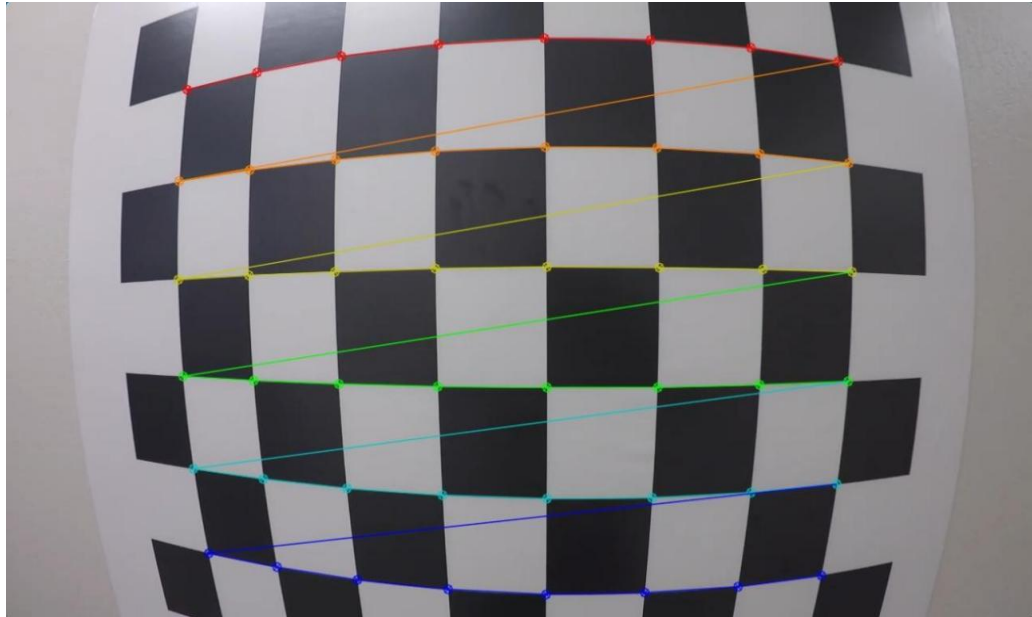


Figure 5- 3: Finding Corners

5.1.3 Calibrating Camera

Just the image's height and width are passed as the image's shape to the `calibrateCamera` function. These values can be obtained, for example, by accessing the grayscale image shape array `gray.shape[0:-1]`. This provides the image's height and width in pixels, such as (1280, 960).

The first two values in the color image shape array can be obtained by using the `img.shape[1:-1]` function to retrieve the image shape directly from the color image. This code snippet requests the shape array's first two values only, then reverses them. It should be noted that because we are dealing with a grayscale image in this instance rather than a color image, where there are three dimensions (height, width, and depth), this step is not required.

Use the first two values of a color image shape or the entire grayscale image shape. This is due to the fact that a color image's overall shape will also include a third value, or the number of color channels, in addition to the image's height and width. If you try to pass these three values into the `calibrateCamera` function, you'll encounter an error. For instance, a color image's shape array might read (960, 1280, 3), where the first two

values correspond to the image's pixel width and height, and the third value represents the image's three color channels.

5.1.4 Distortion Correction

Real cameras employ curved lenses to create an image, and around the edges of these lenses, light rays frequently based either too much or too little. As a result, lines and objects appear to be more or less curved than they actually are, distorting the margins of image. The most typical kind of distortion is referred to as radial distortion.

Tangential distortions is another type of distortion. When the lens of a camera is not completely parallel to the imaging plane, which contains the camera film or sensor, this happens. This gives the appearance that a picture is slanted, making some objects look closer or farther away than they actually are.

To adjust for radial distortion, k_1 , k_2 , and k_3 coefficients are required. One can use a correlation formula to fix the look of radially deformed points in an image. In the shown equations, (x,y) is a distorted image point. To undistort these points, OpenCV calculates r , which is a known distance between a point in a undistorted image, $(x_{corrected}, y_{corrected})$ and the center of the image distortion, which is the center of that image (x_c, y_c) is sometime called distortion center. These points are shown below:

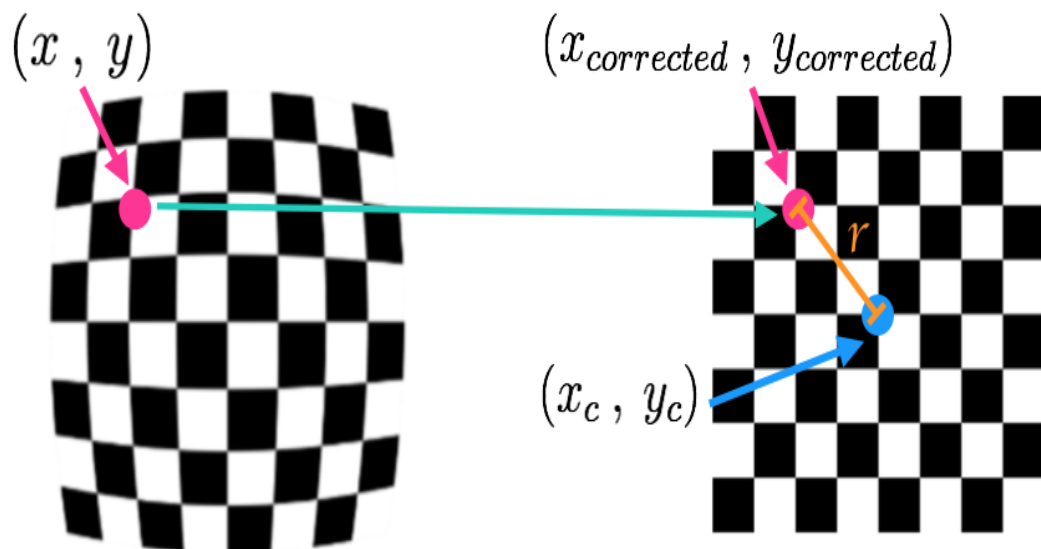


Figure 5- 4: Points in a Distorted and Undistorted Image

A single point in a distorted image is represented by (x,y) and (x_{corrected}, y_{corrected}) represents where that point will appear in the undistorted(corrected) image. Radial distortion correction can be done as follows:

$$x_{\text{distorted}} = x_{\text{ideal}} (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{\text{distorted}} = y_{\text{ideal}} (1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

There are two coefficients that accounts for tangential distortion: p₁ and p₂, and this distortion can be corrected by using a different correlation formula as follows:

$$x_{\text{corrected}} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$

$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

To obtain image point and object point, use chessboard images. Next, computer the calibration and distortion using the OpenCV functions cv2.callibrateCamera() and cv2.undistort().

5.2 Perspective Transform

When objects in three dimensions are projected onto a flat picture plane, the process is called a perspective transformation. This makes distant objects appear smaller than nearby objects. .A perspective transform maps the points in a given image to new image points with a different perspective. The perspective transforms you'll be most interested in is a bird's-eye view transform that allows us to see a lane from above; this will come in handy later on when calculating the lane curvature. A perspective transform can be used for a variety of different viewpoints in addition to creating a bird's eye view representation of an image.

Now we perform perspective transform by taking the camera matrix mtx and distortion coefficient dist as follows:

- Undistort the image using cv2.undistort() with mtx and dist
- Convert to grayscale

- Find the chessboard corners
- Draw corners
- Establish four source points
- Specify four locations where travel will end
- Get the transformation matrix M with `cv2.getPerspectiveTransform()`
- To create a top-down perspective, use the M operator with `cv2.warpPerspective()`

By following the above steps we can perform the perspective transform which is a crucial part in a mini- self driving car.

5.3 Lane Finding

After performing calibration, thresholding, and a perspective transform on a road image, you should have a binary image with distinct lane lines. However, you must still specify which pixels are part of the lines and which belong to the left line and which to the right line. One possible solution is to plot a histogram of where the binary activation occur across the image.

Take the histogram of this region along the columns

```
histogram = np.sum(img[img.shape[0]//2:,:], axis=0)
```

Then, identify peaks and use peak positions as a starting points.

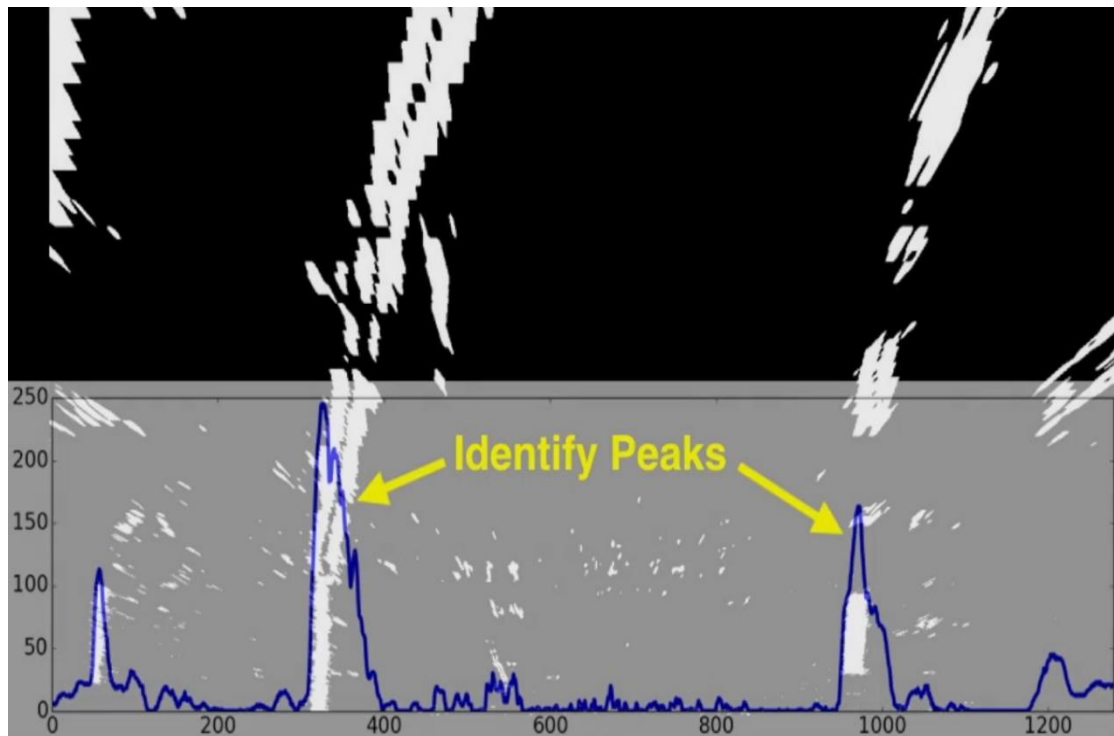


Figure 5- 5: Histogram Peaks for Lane Finding

We are adding up the pixel values along each column in the image with this histogram. Because pixels in our threshold binary image are either 0 or 1, the two most prominent peaks in this histogram will be good indicators of the x-position of the lane lines' bases. That can serve as a starting point for where to look for the lines. From there, we can use a sliding window centered on the line centers to locate and follow the lines all the way to the top of the frame.

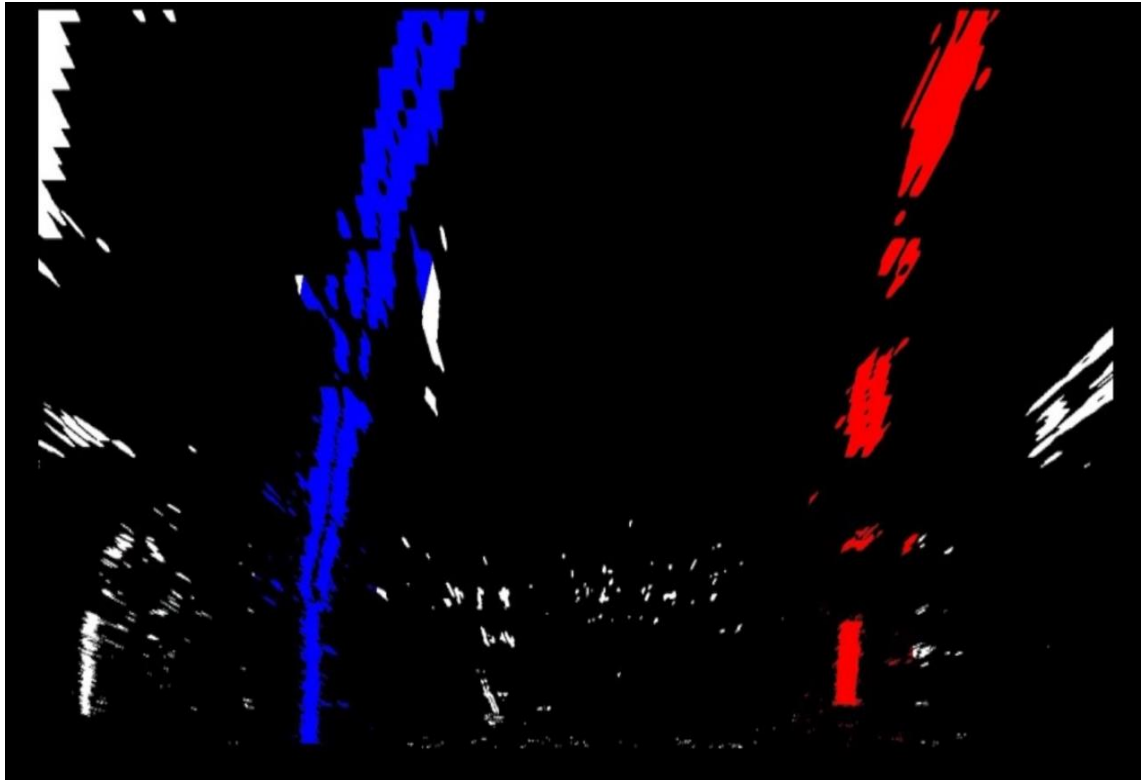


Figure 5- 6: Sliding Window for Lane Finding

5.4 Lane Curvature

The proper steering angle, whether to turn left or right, must be communicated to self-driving cars. If you are aware of the car's speed, dynamics, and degree of lane curvature, you can calculate this angle.

A 2nd degree polynomial can be fitted to a lane line to determine its curvature, and from this you can quickly deduce useful information.

$f(y) = Ay^2 + By + C$, where A, B, and C are coefficients, can be used to fit a line to a nearly vertical lane line. Because the lane lines in the warped image are near vertical and may have the same x value for more than one y value, you're fitting for $f(y)$ rather than $f(x)$.

The radius of curvature of the function $x = f(y)$ at any point x is given as follows:

$$\text{Radius of curvature} = ([1 + ((dy)/(dx))^2]^{3/2}) / (| (d^2y)/(dx^2) |).$$

The first and second derivatives of the above second order polynomial are:

$$f'(y) = dx / dy = 2Ay + B$$

$$f''(y) = d^2(x) / d^2(y) = 2A$$

So, our equation of radius of curvature becomes:

$$\text{Radius of curvature} = (1 + (2Ay + B)^2)^{3/2} / |2A|$$

Since the y values in your image rise from top to bottom, if we wanted to measure the radius of curvature closest to your vehicle, for instance, we could evaluate the formula above at the y value that corresponds to the bottom of your image, or in Python, at `yvalue = image.shape[0]`.

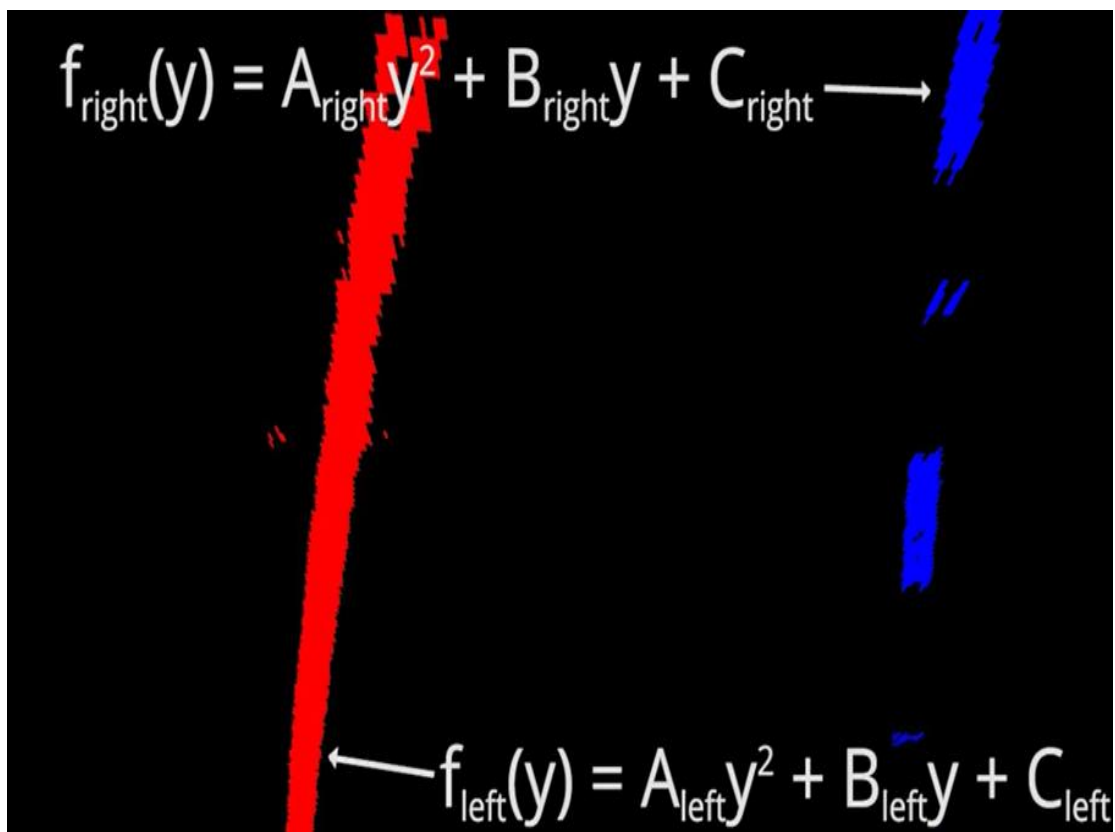


Figure 5- 7: Detect Lines and Determine Curvature

5.5 Voltage Regulation

We are using buck convertor in our project, which is a type of DC-DC converter that is used to step down a DC voltage level to a lower level.

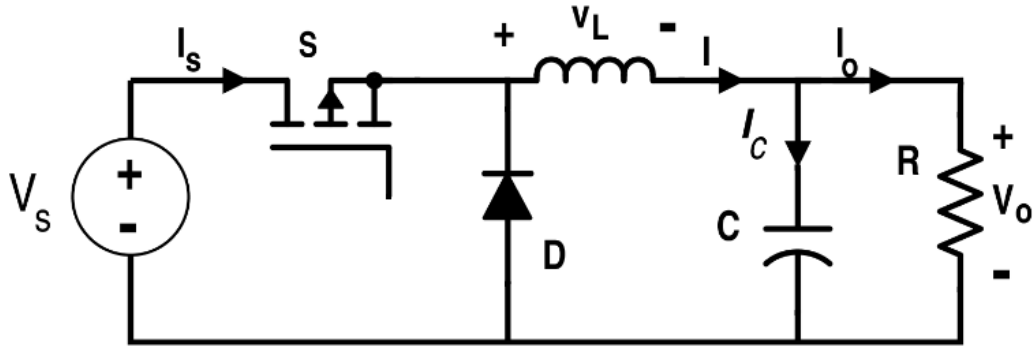


Figure 5- 8: Circuit Diagram For Buck Converter

The basic working principle of a buck converter is to switch an input voltage on and off at a high frequency using a switching element such as a MOSFET. The output voltage is then filtered to remove any unwanted high-frequency components, resulting in a smooth DC voltage level that is lower than the input voltage. The output voltage level is determined by the duty cycle of the switching element, which is the ratio of the on-time to the total switching period.

The Raspberry Pi 4B requires a 5V power supply, which is not compatible with the 12V output of the Li-ion battery used in the mini self-driving car. Therefore, a buck converter is used to step down the voltage level from 12V to 5V. The output of the buck converter is then connected to the power input of the Raspberry Pi 4B. The Raspberry Pi 4B also has a 5V output pin, which can be used to power other components in the mini self-driving car. For example, the Arduino Uno, which is another microcontroller used in our mini self-driving car, can be powered from the 5V output pin of the Raspberry Pi 4B.

Overall, the buck converter plays a crucial role in the mini self-driving car by providing efficient power conversion and ensuring that all the components are powered correctly. Without the buck converter, it would be difficult to power the Raspberry Pi 4B and other components from the 12V Li-ion battery used in the mini self-driving car.

5.6 Procedural Processing in Raspberry Pi 4B

The Raspberry Pi 4B supports multiprocessing using multiple cores. This allows you to run multiple processes simultaneously, improving performance and efficiency. Python's multiprocessing module provides a way to create and manage multiple processes in your Python programs. You can use the module to run independent tasks in parallel, taking advantage of the multiple cores available on the Raspberry Pi 4B.

However, lane detection and model object detection took longer than 10 seconds on the Raspberry Pi when we used multiprocessing, which was a significant issue for our project. Using serial processing rather than multiprocessing can be a practical solution to this issue.

We conducted some research to determine why multiprocessing in the Raspberry Pi is slower, and we discovered the following explanations:

- **Overhead of Task Switching:** Multiprocessing requires the operating system to switch between different processes, which incurs overhead and reduces performance. The overhead of task switching can be especially pronounced on the Raspberry Pi due to its limited hardware resources.
- **Memory Bandwidth Constraints:** The Raspberry Pi has a limited amount of memory bandwidth, which can be a bottleneck when using multiprocessing. When multiple processes are running simultaneously, they may compete for memory bandwidth, which can slow down overall performance.
- **Power Management:** The Raspberry Pi has a low-power architecture that is designed to conserve energy. This can result in slower performance when using multiprocessing, as the CPU may switch to a lower power state when not in use.

Overall, the most common causes of a slow Raspberry Pi are the hardware, operating system, and installed applications. To make a slow Raspberry Pi run faster, we can remove unused apps and services. Use a better SD card, don't forget to cool it down, get a proper power supply, or even experiment with overclocking.

Procedural processing is a method of running tasks one after another in a sequential manner, without parallel processing. In the Raspberry Pi 4B, you can run serial processing using a single core, without utilizing the other cores. This method is typically used when the tasks are dependent on each other, or when parallel processing would not improve performance. Serial processing can be implemented in Python using a simple loop or by using the queue module, which allows you to create a queue of tasks to be processed one after another. The queue module is part of Python's standard library, and it provides a way to manage and organize serial processing in a programmatic manner.

Thus, following the application of procedural processing, lane detection required about 0.04 seconds and model object detection required about 1.9 seconds, both of which are still too long but are still superior to the outcomes of multiprocessing. For this reason, in the Raspberry Pi 4B, we favor procedural processing over multiprocessing.

5.7 Power and Steering

A lithium-ion battery serves as the primary source of power for our miniature self-driving car, which receives its power from a battery and is programmed to travel at a predetermined speed. Due to the fact that they have a high cycle life and a low rate of self-discharge, they are ideal for use in vehicles that require power for an extended period of time. These qualities, in addition to the relatively affordable cost of Li-ion batteries, have contributed to their status as the most popular type of rechargeable battery in the United States. The electric motors that are responsible for turning the wheels of the vehicle are powered by electricity supplied by the battery.

The speed of the motors in a mini self-driving car is a critical aspect that plays a significant role in its movement and maneuverability. This speed is controlled by a microcontroller, which receives input from a camera and processes it to make decisions about how the car should move. The microcontroller is the central processing unit of the car, and it is responsible for making decisions based on the information it receives from various sensors. In the case of the motors, the microcontroller sends control signals to the Electronic Speed Controller (ESC), which adjusts the voltage supplied to the motors. The ESC acts as a bridge between the microcontroller and the motors,

converting the control signals from the microcontroller into changes in the voltage supplied to the motors. This voltage adjustment directly affects the speed of the motors, allowing the microcontroller to control their speed. In our mini self-driving car, we are supplying power to the rear wheels, which play a crucial role in the car's movement and maneuverability. By utilizing a microcontroller to control the speed of the motors and an ESC to regulate the voltage supplied to the motors, we can ensure that the car moves smoothly and efficiently, responding to input from the camera and other sensors to navigate through its environment.

It is essential for a self-driving car of any size to have the capacity to steer, but this is especially true for miniature versions of self-driving vehicles. A direct current (DC) motor that is connected to the automobile's front wheels is used in these vehicles to control the steering system. The microcontroller, which is responsible for determining the desired position of the wheels, sends a control signal that drives the DC motor. This signal causes the wheels to move in the desired direction. This control signal establishes the inclination at which the DC motor rotates the wheels, thereby enabling the vehicle to turn in the desired direction. The direct current (DC) motor is built to be extremely accurate. This ensures that the wheels are positioned precisely as desired, which enables the car to turn with a high degree of precision. Because of the precise positioning capabilities of the DC motor, the mini self-driving car is able to easily navigate through confined spaces and around obstacles, which ensures that it can travel in any environment in a manner that is both safe and effective.

5.8 Traffic light and Stop sign

To implement a traffic light detection and stop sign system in a mini self-driving car, we followed the following steps:



Figure 5- 9: Stop Sign

- Mount a camera on the car: A camera is mounted on the car to capture images of the traffic lights or stop sign.
- Image processing: Image processing algorithms are applied on the captured images to detect the presence and location of the traffic lights or stop signs. This can be done using techniques such as color detection, and object detection.
- State identification: Once the traffic light is detected in the image, the state of the traffic light (red or green) is identified using color detection algorithms. In our project we are not implementing only red and green lights for simplicity and we didn't find yellow light that significant as this is just the prototype of the car. In our project once the stop sign is detected in the image, the stop sign recognition algorithms are applied to confirm that the detected object is indeed a stop sign.
- Integration with control system: The information obtained from the traffic light or stop sign detection system is then integrated with the self-driving car's control system to control the car's movements based on the state of the traffic lights or stop signs.

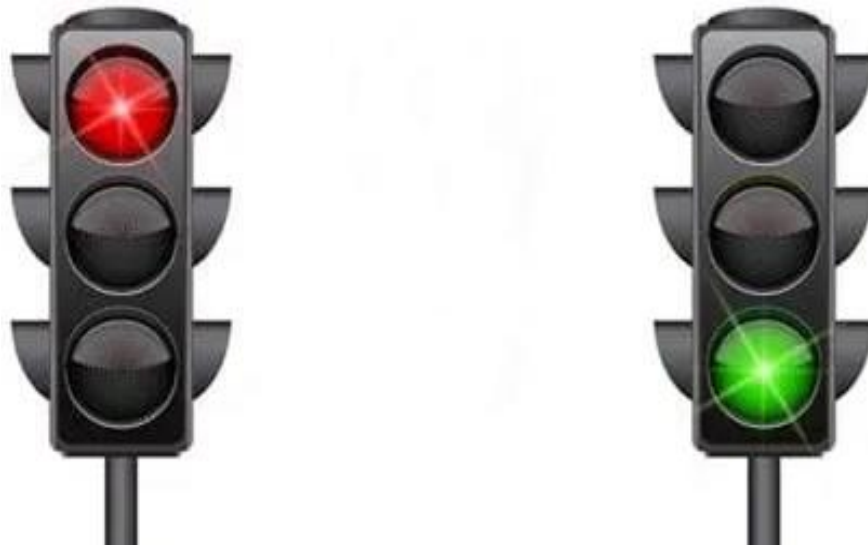


Figure 5- 10: Traffic Lights

5.9 Speed Limits

To implement a speed limit in a mini-self-driving car, we use a combination of hardware and software components in the following ways:



Figure 5- 11: Speed Limit 20



Figure 5- 12: Speed Limit 80

The first picture illustrates the upper limit of our vehicle, while the second picture illustrates the lower limit of our vehicle. Our vehicle will not exceed the maximum speed limit, and if it comes close to doing so, it will automatically begin to reduce its speed on its own.

The first thing that we are going to do is count the number of times that the wheel turns in one minute (rpm). In most circumstances, we start off with a speed that is approximately 120 revolutions per minute. Following this step, the rate at which the wheel is rotating is transformed into an actual speed value. A microcontroller based on a Raspberry Pi is used for the processing of the speed data, and an Arduino UNO is used for the management of the motor. Both of these devices are part of our system. The data on speed that is sent from the sensor is read by a program that we have written, and then that data is compared to the maximum speed limit that is permitted. When the new speed limit goes into effect, it will be 80 miles per hour, which is significantly faster than the alternative choice of 20 miles per hour. Following the reading of the limit sign by the camera, the algorithm can then instruct the motor to reduce speed if it determines that the vehicle is traveling at a speed that is higher than the limit. To successfully implement a speed limit in a vehicle that is capable of performing limited levels of autonomous driving, one must have knowledge of both hardware interfacing and software programming. This is necessary in order to achieve the desired level of success.

5.10 Obstacle Detection and Distance Calculation

To implement an obstacle detection system in our mini self-driving car, the following steps is followed:

- **Sensor setup:** Obstacle detection is performed using various sensors such as ultrasonic sensor and camera. WE have chosen the appropriate sensor based on the requirements and integrate it with the self-driving car.
- **Data collection:** Collect data from the sensors to obtain information about the surrounding environment of the car, including the distance and position of any obstacles.
- **Data processing:** Apply data processing algorithms to the collected data to identify and isolate obstacles. This involves techniques such as clustering or thresholding.
- **Obstacle avoidance:** Based on the information about the obstacles, the self-driving car can make decisions about how to avoid or navigate around them. This information can be used to control the car's movements and ensure its safety.

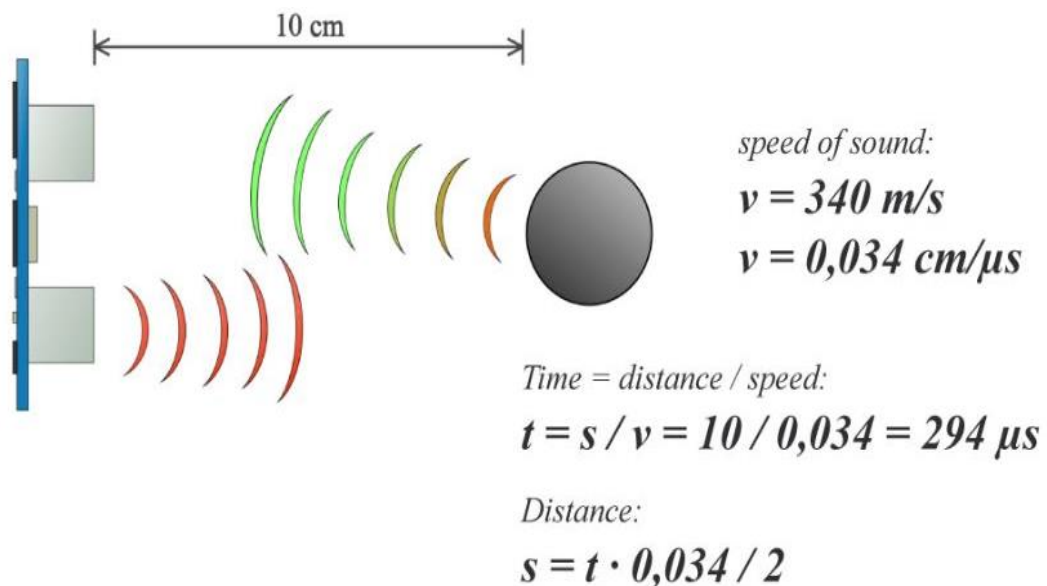


Figure 5- 13:Distance Calculation using Ultrasonic Sensor

Our mini self-driving car is able to calculate the distance of the obstacle using the ultrasonic sensor. We have used the ultrasonic sensor whose range is from 2 centimeters

to 400 centimeters. In order to function, ultrasonic sensors send out sound waves at a high frequency, which then reflect off of the target object and are reflected back to the sensor. Taking into account the amount of time it takes for the sound waves to return, the sensor is able to compute the distance that separates it from the obstacle.

The speed of sound in air, which is approximately 343 meters per second when the temperature is at room temperature, is used as the basis for the calculation of distance. The ultrasonic sensor starts by sending out a sound wave, and then it waits for it to come back to it after it has been reflected off of the obstruction. After determining the amount of time necessary for the sound wave to complete this round-trip, the following formula is used to determine the distance to the obstruction:

$$\text{distance} = (\text{speed of sound} \times \text{time taken}) / 2$$

The travel time of the sound wave from the sensor to the obstacle and back to the sensor is accounted for by the factor of 2 in the formula.

Let's say that the time it takes for the echo to return after the ultrasonic sensor has emitted a sound wave is 5 milliseconds. At a temperature of approximately 343 meters per second, the speed of sound in air is approximately 343 meters per second.

$$\text{distance} = (343 \text{ m/s} \times 0.005 \text{ s}) / 2 \quad \text{distance} = 85.75 \text{ centimeters}$$

As a direct result of this, the distance to the obstruction is approximately 85.75 centimeters. For the purpose of this calculation, we will assume that the sensor is not hampered in any way, either by obstructions or reflections, which could cause it to produce inaccurate results. According to the findings of our research, ultrasonic sensors have a few drawbacks, the most notable of which are their susceptibility to interference from other objects and their susceptibility to background noise, both of which can have an effect on the precision and range of the sensors.

5.11 Design of the Car

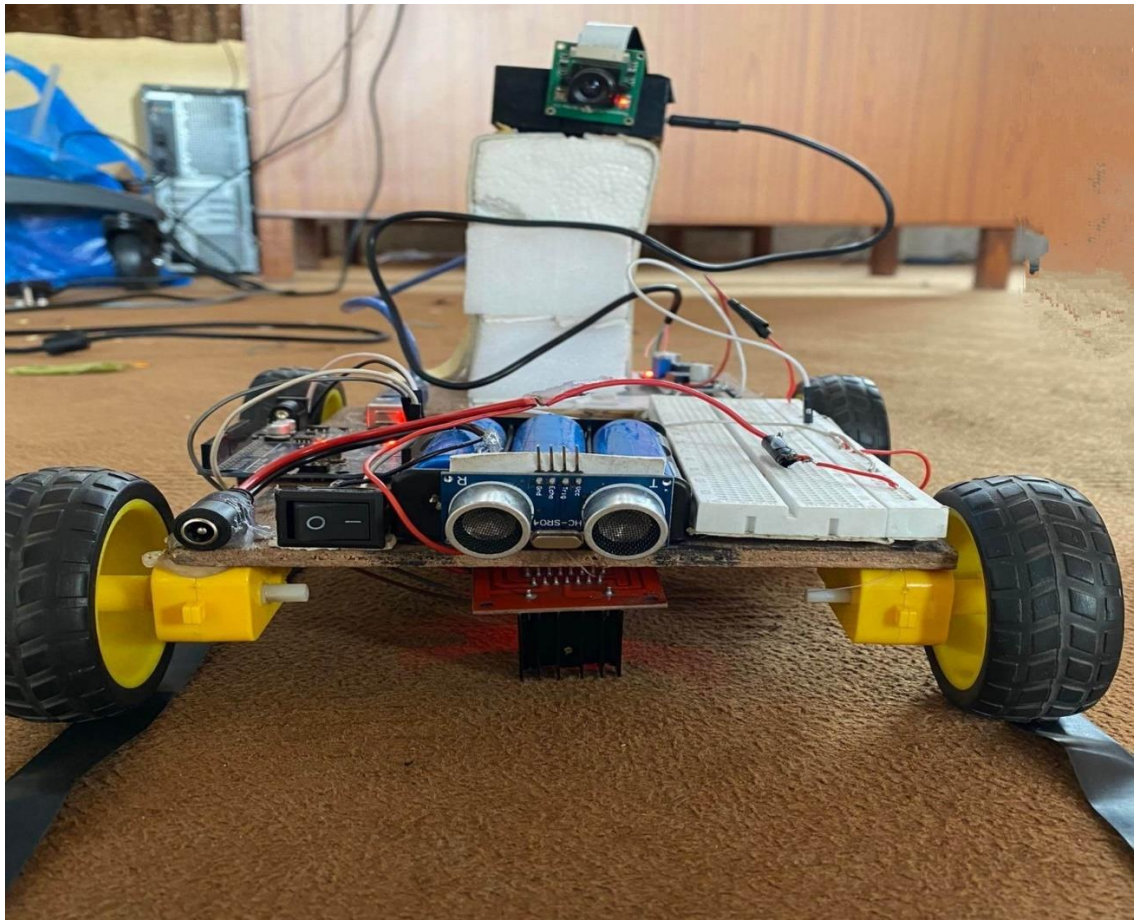


Figure 5- 14: Model of Mini Self-Driving Car

As it determines how the vehicle will move and how it will interact with its surroundings, the design of a mini self-driving car is extremely important to the implementation of the technology. It is imperative that careful consideration be given to the design in order to guarantee that it is not only practical but also secure. When developing the blueprints for a small autonomous vehicle, it is essential to take into account the components' weight, dimensions, and overall shape.

The selection of materials for a mini self-driving car's construction is crucial. We have used plywood to make chassis of our mini self-driving car because it is lightweight, durable, and easy to manipulate. Also, it is readily accessible and relatively inexpensive compared to materials like metal or carbon fiber. The car's wheels are made of a durable material rubber, for optimal road traction.

The measurements of the autonomous vehicle's dimensions are also very important. It is imperative that the vehicle be compact enough to negotiate narrow roads and sharp turns. Our miniature vehicle measures approximately 25 centimeters long and 10 centimeters wide. We were concerned about its maneuverability, safety, and potential impact on the environment, so we reduced its size.

The vehicle's weight is another vital design consideration. To reduce energy consumption, increase range, and improve performance, the vehicle must be lightweight. Additionally, it must be heavy enough to provide road stability. Our mini self-driving car is around 5 kg which includes the mass of plywood, battery, raspberry pi, Arduino, motor, motor driver, wheels, raspberry pi camera, ultrasonic sensors, buck convertor and wire. The vehicle's mass should be roughly optimized to strike a balance between above factors and ensure that it is both safe and fuel-efficient.

For reasons having to do with efficiency, stability, and aerodynamics, we decided to give our little car a rectangular form. The camera is mounted on a piece of foam so that the user has a more comprehensive view of the track, and all of the motors and motor drivers have been mounted to the chassis of the vehicle so that it does not appear to be cluttered. On top of that, the buck converter, the battery, and any other pieces of hardware are placed on the plywood in order to ensure that the weight of the car is distributed evenly.

5.12 Operational Design Domain

In the context of autonomous vehicles, the term "Operational Design Domain" (ODD) refers to the range of scenarios in which a vehicle of this type is designed to carry out its functions in a manner that is both efficient and risk-free. The ODD basically delineates the limits of what a self-driving car is designed and programmed to handle. Any conditions that fall outside of those boundaries are considered "out of domain," and they may render the car unable to navigate or operate in a safe manner. It is essential to define the ODD for our project involving a miniature self-driving car in order to ensure that users are aware of the car's capabilities as well as its limitations. When determining the ODD for your mini self-driving car, we kept following key considerations in our mind which are as follows:

- **Type and Conditions of the Roads:** The ODD for your mini autonomous vehicle could specify that the vehicle is designed to operate on well-marked roads with distinct lane markings and minimal traffic. This means that the vehicle should be able to stay in its lane and not deviate from its path. The vehicle is able to operate in good lighting conditions under which the vehicle is designed to operate.
- **Environmental Factors:** Environmental factors can have a substantial impact on the performance of your mini self-driving car, so it is essential to specify what the vehicle can and cannot handle. Our car is designed to operate within a specific temperature range which is approximately 0 and 50 degrees Celsius to protect the hardware components.
- **Speed Limits:** The maximum speed at which a vehicle can be driven safely is a crucial factor for the ODD. We have specified the maximum permissible speed limit for the vehicle which is around 10 centimeters per second.
- **Obstacle Detection and Avoidance:** To ensure safe operation of car, our mini self-driving car is able to detect and avoid obstacles in its path. The ODD have specified the types of obstacles that a vehicle can detect and avoid, such as stationary obstacles. The vehicle should be able to navigate these obstacles safely and avoid collisions.
- **Traffic Light and Stop Sign Detection:** To ensure safe navigation, your mini autonomous vehicle should also be able to detect and respond to traffic lights and stop signs. The ODD have specified the detection range and illumination conditions under which a vehicle is designed to detect and respond to traffic lights and stop signs. For instance, our car is programmed to detect and respond to traffic lights and stop signs within 1 meter and during daylight hours.

6. RESULTS AND ANALYSIS

6.1 Software Results

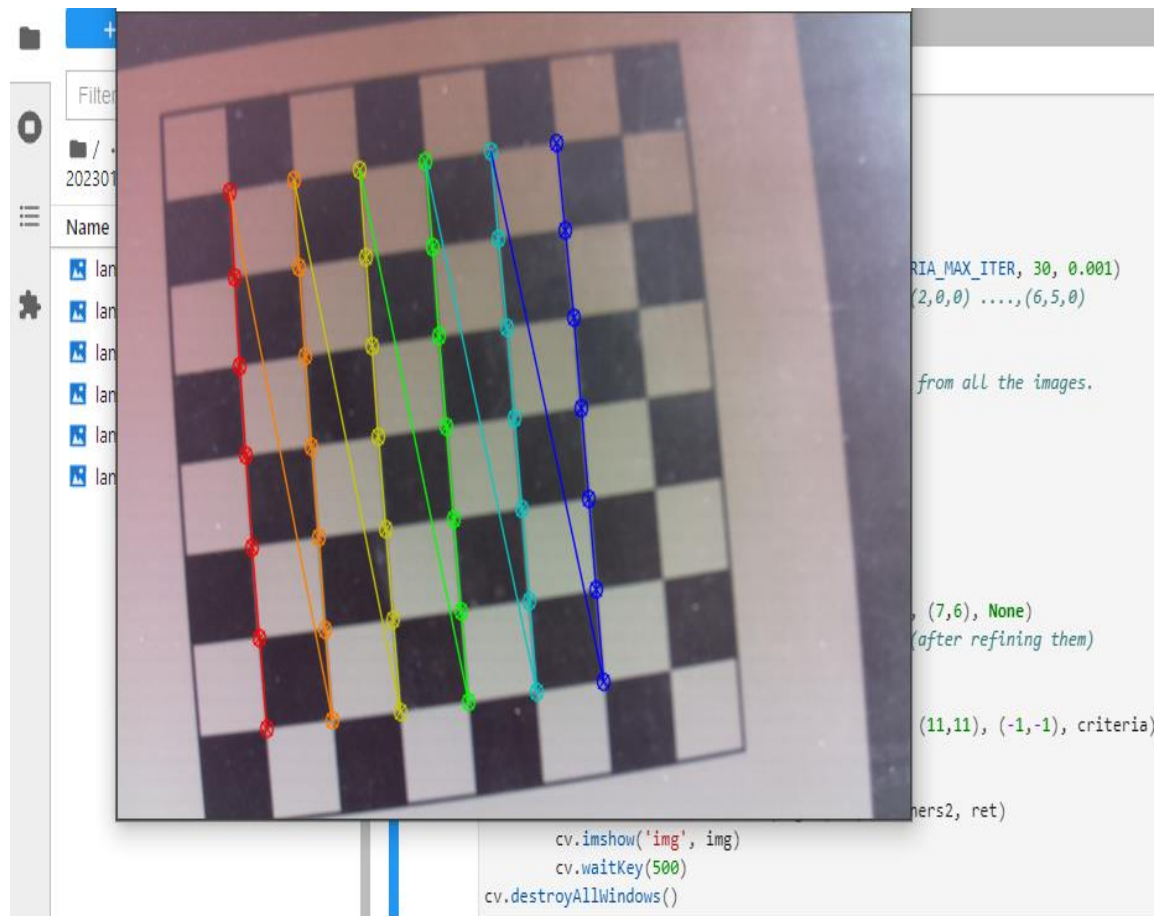


Figure 6- 1: Finding Corners of a Chess Board

The camera feed was analyzed to determine the chessboard's corners, which were then used to map the car's position and orientation. The implementation involved processing the camera feed with the OpenCV library. First, a series of image processing techniques were used to identify the chessboard's corners. This included thresholding the image to convert it to a binary image, using edge detection algorithms to detect the edges, and the Hough transform to identify lines.

Once the lines were identified, their intersections were calculated to determine the chessboard's corners. The algorithm was designed to be resilient to variations in lighting and camera position, and it was tested under a variety of conditions to ensure its accuracy.

The results demonstrated that the algorithm accurately identified the corners of the chessboard in a variety of conditions, including lighting and camera position changes. The corners were identified with a high degree of precision, enabling the car to map its position and orientation on the chessboard with precision.

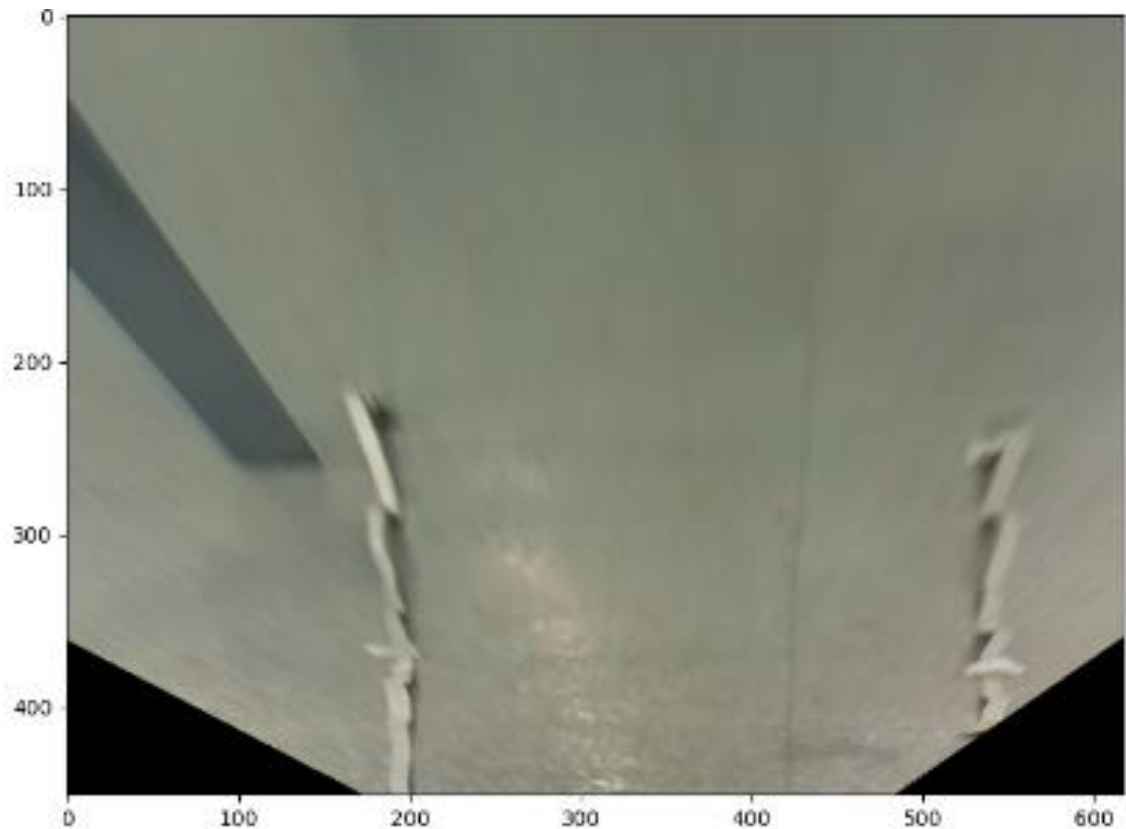


Figure 6- 2: Transformed Image

During the implementation of a miniature self-driving car, one of the tasks was to convert the camera image to a top-down view using the OpenCV library to facilitate processing and analysis. Image transformation techniques were applied to the camera feed in order to obtain a bird's-eye view of the car's surroundings. The results demonstrated that the transformation was successful in providing a top-down view of the scene, which made the camera feed easier to process and analyze. The transformed image accurately represented the car's surroundings, allowing for precise detection and avoidance of obstacles.

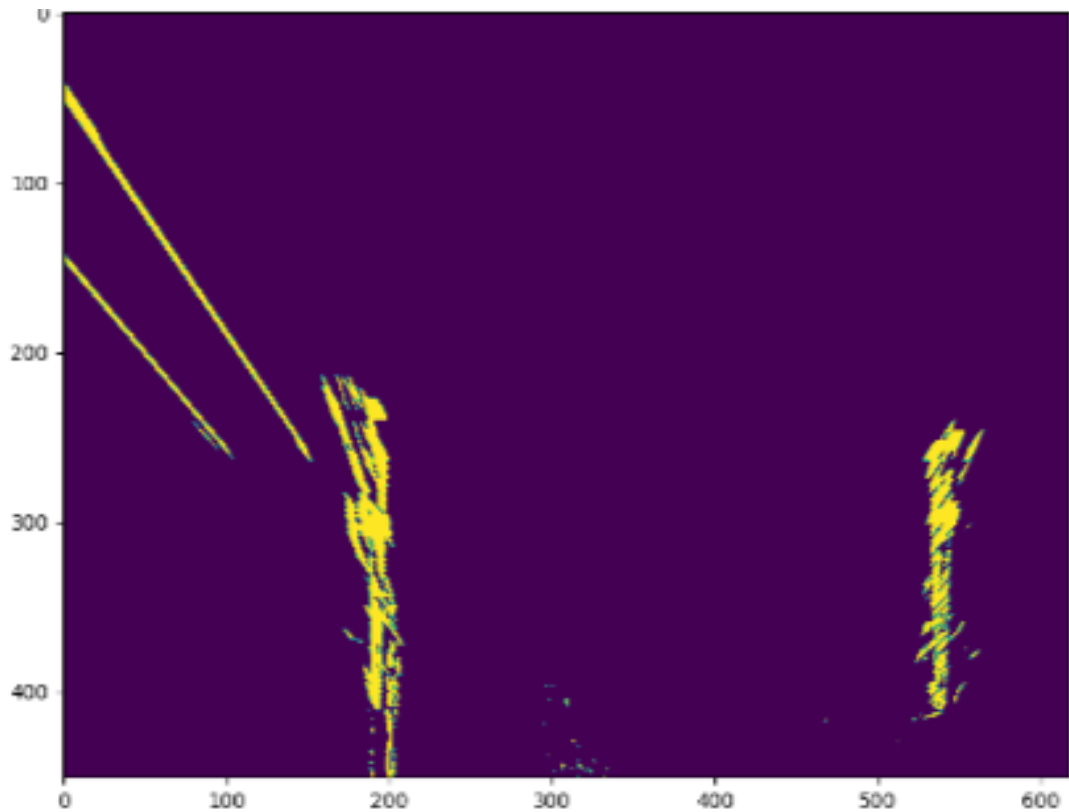


Figure 6- 3: Transformed S Channel and Gradient Thresholds

We used transformed S channel and gradient thresholds to the camera feed to detect lane lines. The process involved analyzing the transformed image to identify lane lines, which would then be used to guide the car's steering. The results demonstrated that the transformed S channel and gradient thresholds were effective at accurately detecting lane lines. The lane lines were identified with a high degree of certainty, allowing for precise steering guidance of the autonomous minicar.

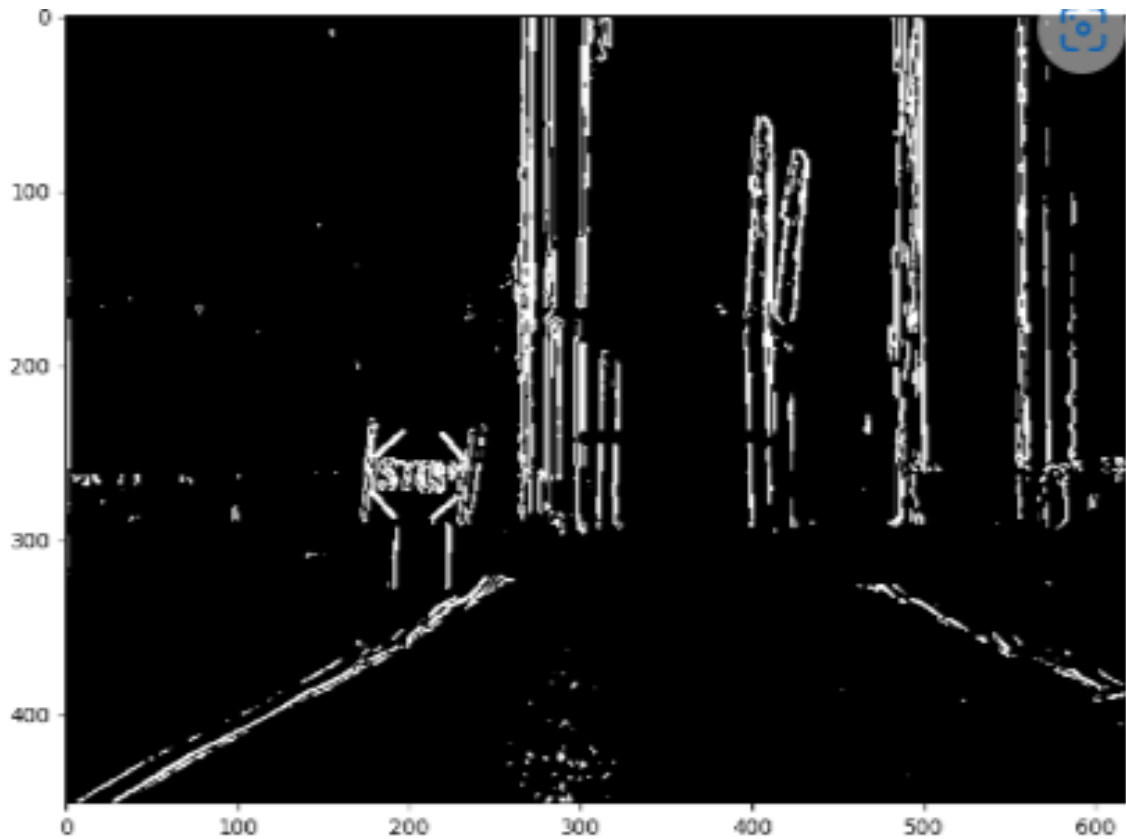


Figure 6- 4: Combined S Channel and Gradient Thresholds

By combining the S channel from the HLS color space with gradient thresholds in the x and y directions, the vehicle was able to detect lane lines with greater precision and robustness than when using either threshold alone. The S channel was particularly effective at detecting white and yellow lane lines, while gradient thresholds assisted in the detection of image edges and gradients.

The combined threshold approach was also effective in a variety of lighting conditions, including low light and shadows, which are frequently problematic for lane detection. The analysis of the results revealed that the combined threshold approach decreased the number of false positives and improved the accuracy of lane detection, resulting in more accurate and dependable navigation of the vehicle on the road.

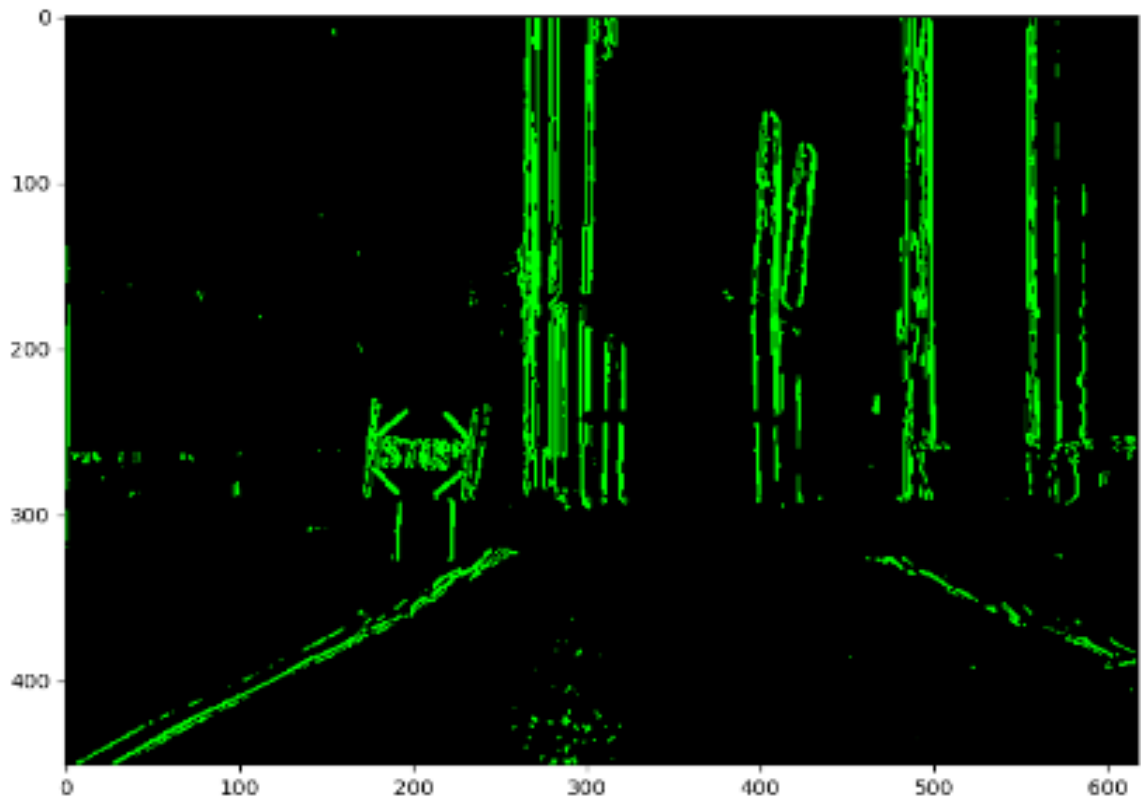


Figure 6- 5: Stacked Thresholds

The results and analysis of using stacked threshold in a miniature self-driving vehicle indicated promising improvements in the vehicle's ability to detect and avoid obstacles. Using multiple thresholding techniques, such as color thresholding and edge detection, the car was able to identify and track objects in its environment with greater precision.

The stacked threshold approach enabled a more thorough analysis of the image data captured by the vehicle's sensors, resulting in a more reliable obstacle detection system. This enhanced the vehicle's safety and dependability, making it more suitable for real-world applications.

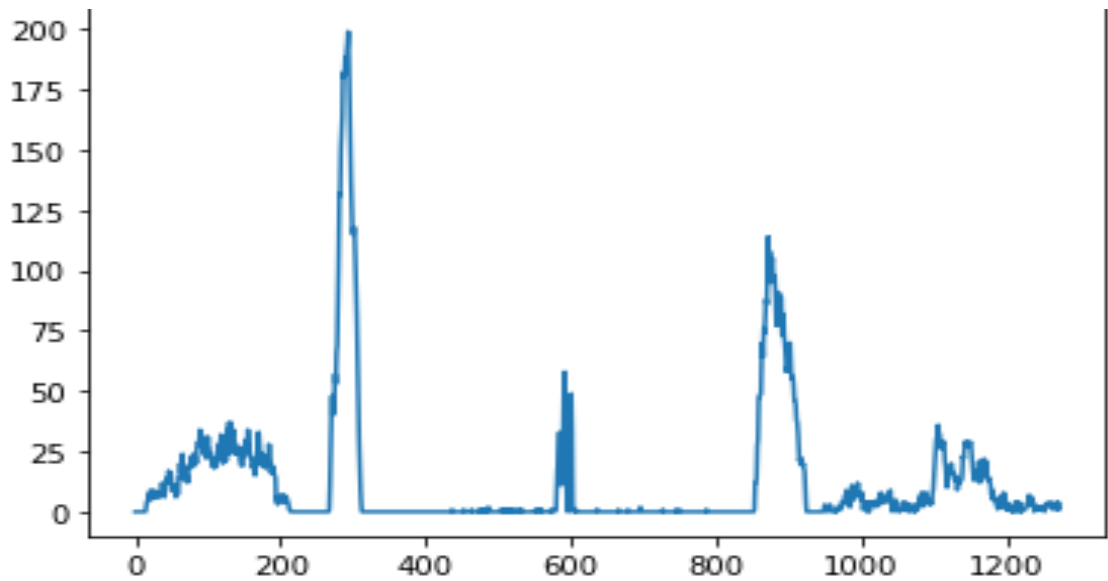


Figure 6- 6: Histogram of Transform S channel and Gradient Threshold

The histogram of transform S channel enabled the vehicle to isolate lane lines based on their color characteristics, while the gradient threshold helped to identify corresponding edges and gradients. From the preceding histogram, the two peaks at approximately 300 and 900 on the x-axis represent the two lanes for the vehicle's path.

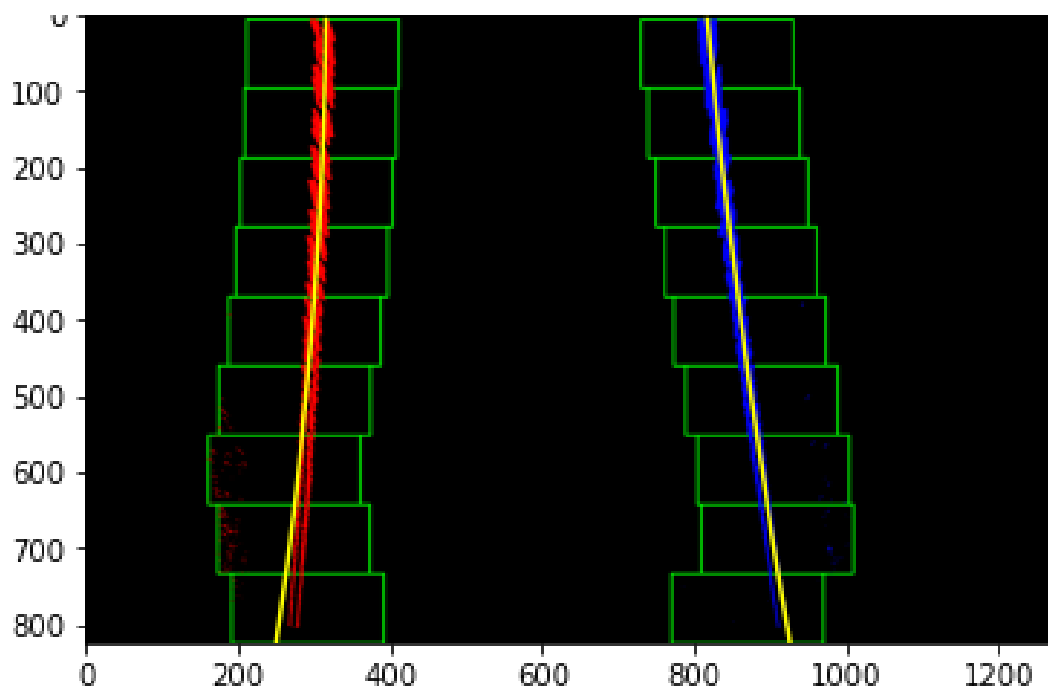


Figure 6- 7: Lane Detection using Sliding Window

From the two peaks of the histogram, the car's lane is created using a sliding window. The sliding window method enabled the vehicle to isolate the lane lines and fit a polynomial curve to their positions, resulting in a more precise and dependable lane detection system.

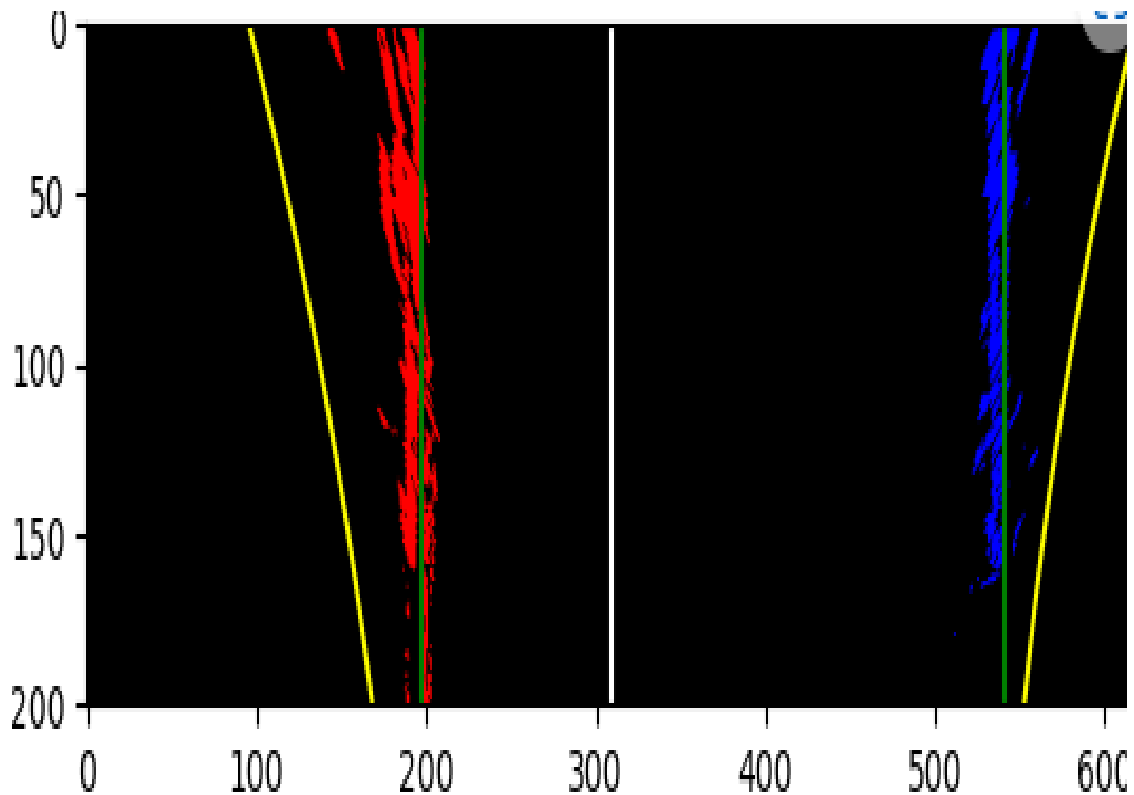


Figure 6- 8: Crop Image of Lane Detected using Sliding Window

The results and analysis of using a cropped image of a lane detected by a sliding window in a mini self-driving car demonstrated significant improvements in its ability to precisely follow the lane and navigate more efficiently. By cropping the image around the detected lane, the vehicle was able to reduce the amount of image data it needed to process, resulting in faster and more efficient navigation.

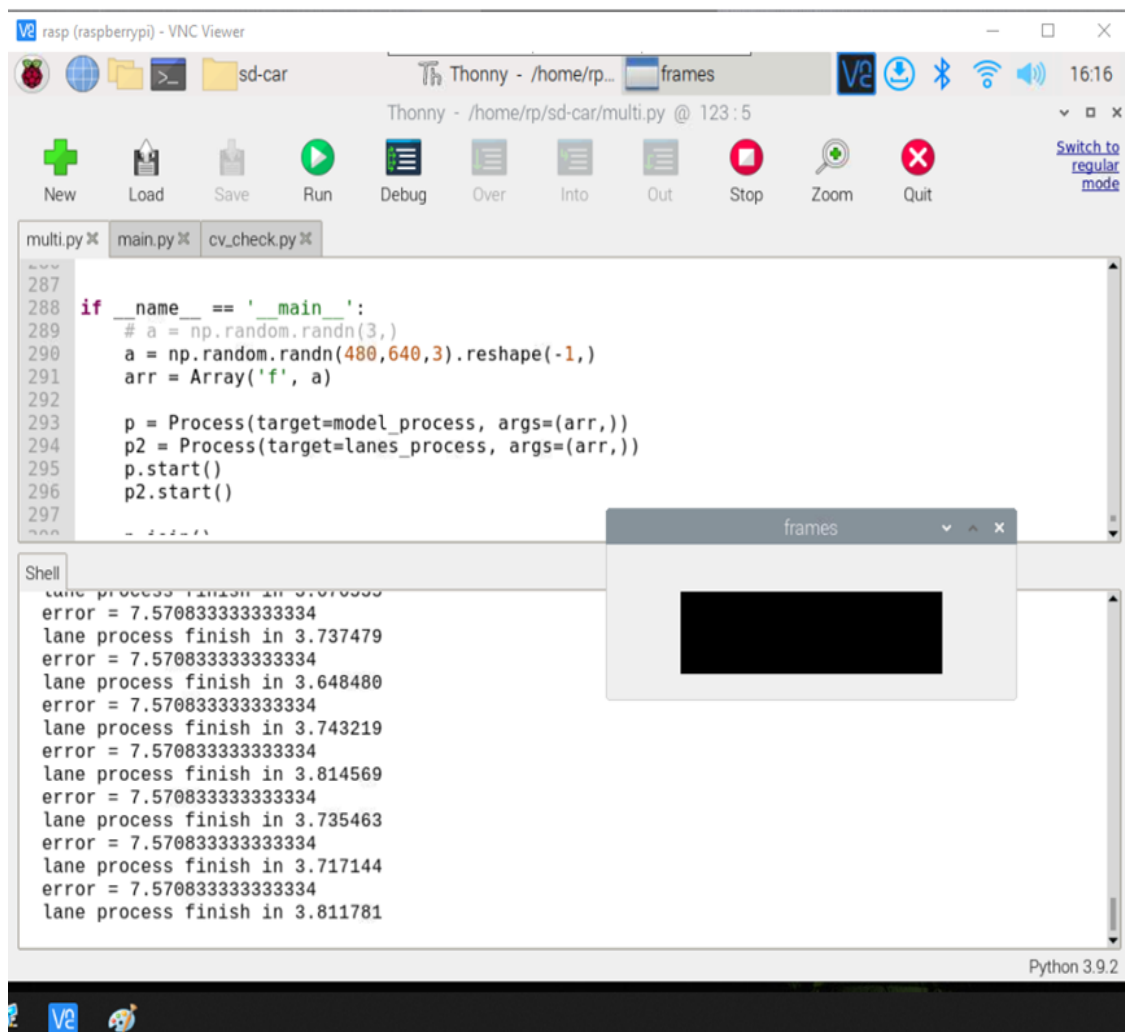
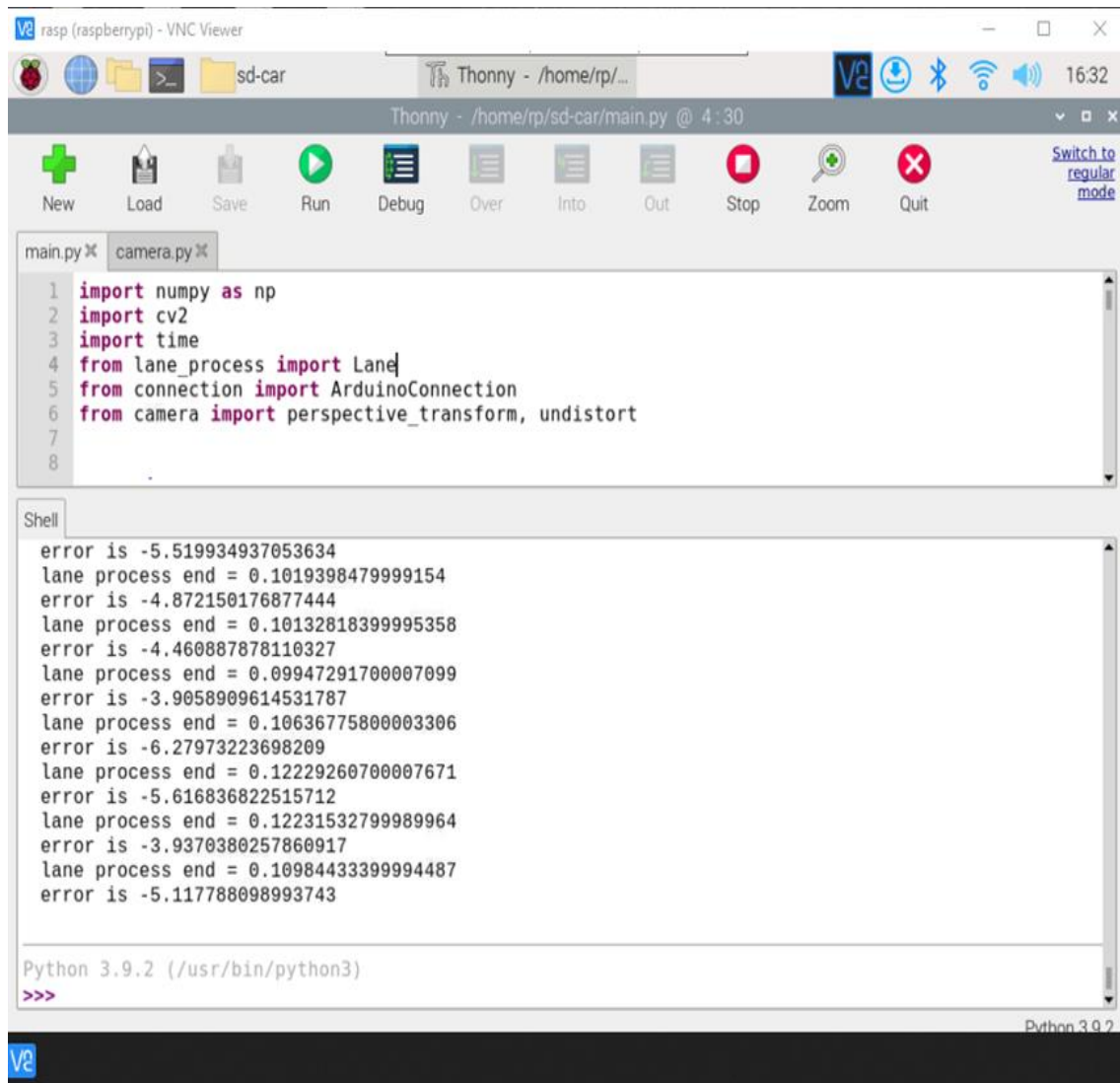


Figure 6- 9: Multiprocessing in Raspberry Pi

When we utilized multiprocessing on the Raspberry Pi, both lane detection and the detection of model objects took more than ten seconds to complete. This was a major obstacle for the project that we were working on. As a direct consequence of this, we decided against utilizing multiprocessing and instead opted to make use of procedural processing instead.



```
1 import numpy as np
2 import cv2
3 import time
4 from lane_process import Lane
5 from connection import ArduinoConnection
6 from camera import perspective_transform, undistort
7
8
```

```
error is -5.519934937053634
lane process end = 0.1019398479999154
error is -4.872150176877444
lane process end = 0.10132818399995358
error is -4.460887078110327
lane process end = 0.09947291700007099
error is -3.9058909614531787
lane process end = 0.10636775800003306
error is -6.27973223698209
lane process end = 0.12229260700007671
error is -5.616836822515712
lane process end = 0.12231532799989964
error is -3.9370380257860917
lane process end = 0.10984433399994487
error is -5.117788098993743
```

```
Python 3.9.2 (/usr/bin/python3)
>>>
```

Figure 6- 10: Procedural Processing in Raspberry Pi

We were able to accomplish lane detection in approximately 0.04 seconds using procedural processing in Raspberry Pi, and object model detection in approximately 1.9 seconds. These times are significantly better than the results we achieved using multiprocessing in Raspberry Pi.

6.2 Hardware Results



Figure 6- 11: Stop Sign Detection

Our mini self-driving car was able to detect stop signs and respond appropriately by decelerating and stopping in front of the sign. Using computer vision algorithms, the vehicle was able to identify the shape and color of stop signs in its environment. While the system was able to correctly identify the stop sign, it was found to be relatively slow due to the time required for the camera to process the image. To address this issue, a more potent camera and microcontroller could be implemented to improve the system's accuracy and response time.

The car was additionally equipped with a traffic light detection system. The car was able to recognize the colors of traffic lights and respond accordingly thanks to computer vision algorithms. The system was highly accurate at detecting and responding to red and green traffic lights. However, the accuracy of the system was found to be dependent on the image quality captured by the camera. Therefore, the implementation of more potent hardware components could enhance the traffic light detection system's precision and responsiveness.

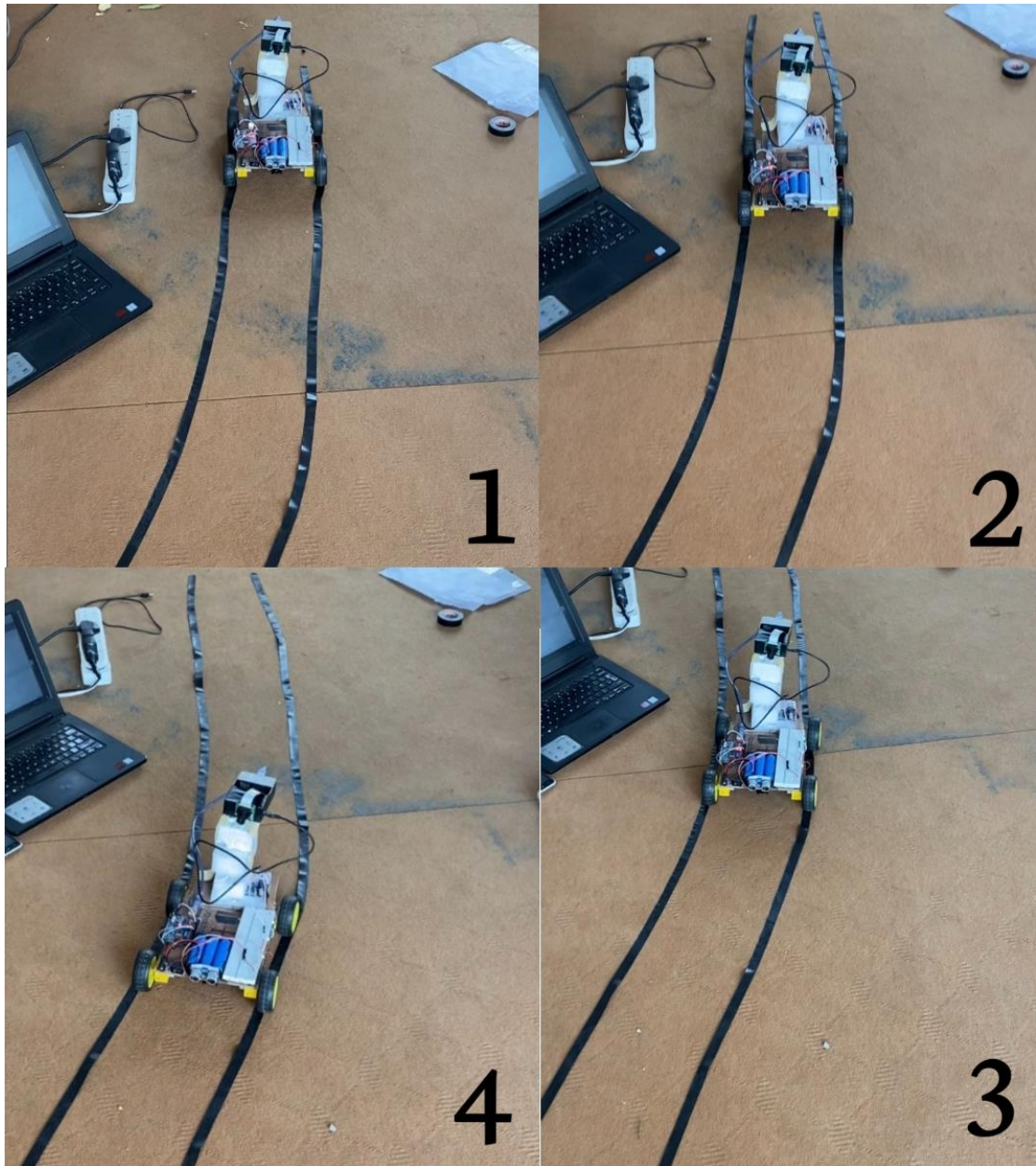


Figure 6- 12: Car Following Lane

The above image depicts the car's movement along the lane. These four images were extracted from our video of a mini self-driving car to demonstrate how precisely our vehicle follows the lane. Our car was able to follow the lane successfully by detecting the lane markings. A camera mounted on the top of our car captured images of the road ahead, which were then processed by an image processing algorithm to identify the lines. To stay on course, the car adjusted its steering angle based on the difference between its actual position and its desired position within the lane. On average, the distance between the center of the car and the center of the lane was less than 3

centimeters, indicating the vehicle's precision. The successful demonstration of our mini self-driving car demonstrates the potential of autonomous driving technology, but there are still obstacles to overcome before it can be implemented.

6.3 Challenges Faced and Possible Remedies

We encountered a number of issues while developing a mini-self-driving car, but we were able to successfully resolve all of them. We discovered the following as a result of our project work:

- Differential current, shorted amateur winding, and shorted field winding are three important parameters that are to be closely monitored when using DC motors. These parameters are crucial indicators of the performance and health of the motor, and can provide valuable insight into its functioning. By observing these factors, we are able to identify any potential issues or faults with the motor, allowing for prompt corrective action to be taken. This proactive approach helps to ensure that the DC motor continues to operate efficiently and effectively, and can help to extend its lifespan and prevent costly downtime or repairs.
- Despite the use of a buck converter to reduce the voltage, a voltage deficiency may still be observed in a Raspberry Pi. This means that the device is not receiving the expected voltage, and may experience performance issues or shutdowns as a result. It is important to closely monitor the voltage being supplied to a device like the Raspberry Pi, and to take appropriate corrective action if a voltage deficiency is observed.
- Despite providing the same voltage and current to different DC motors, different torque can be produced. This can result in a car slipping from its lane, causing a potential safety hazard. The torque produced by a DC motor is directly proportional to the current flowing through it and the strength of the magnetic field generated by the winding. As such, even if the same voltage and current are supplied to different DC motors, differences in their construction or design may result in differences in the torque produced. This highlights the importance of carefully selecting the appropriate DC motor for a given application and of closely monitoring its performance to ensure that it is functioning as expected.

- The Raspberry Pi, being a single-board computer, may not have the necessary resources to run YOLO efficiently, leading to difficulties during the installation process. This can include issues with compatibility, system requirements, and software dependencies. To overcome these challenges, it is necessary to carefully research and follow detailed installation instructions, or to use alternative object detection systems that are better suited to the Raspberry Pi's limited resources.
- Calibration is a critical aspect of camera system particularly in a mini-self driving car. If the camera is not calibrated correctly, it will result in significant problems with lane detection and following. This can lead to incorrect or unreliable lane recognition, which can cause the autonomous vehicle to veer off course or make unsafe driving decisions. To ensure that the camera is functioning properly, it is important to regularly calibrate it and check for any issues that may arise.
- When attempting to connect to a Raspberry Pi that is connected in our car, we were firstly unable to establish an SSH connection. This issue was preventing us from remotely accessing the device and control its driving functions. We found that the issue was due to various reasons such as network configuration, firewall settings, etc.
- Integrating various components, such as sensors, cameras, and actuators is a challenging task. So, we recommend to start with a smaller, simpler model, and gradually add complexity after becoming familiar with the system.
- Using at least two cameras is highly recommended when building a mini self-driving car, especially if it needs to travel in a larger area. A single Raspberry Pi camera has a narrow range of view and may not be able to capture objects that are far away or in peripheral vision. By using multiple cameras, the coverage area can be increased, the accuracy of object detection and recognition can be improved, and the system can become more robust by providing redundant information. Having two or more cameras also allows for stereo vision, which can be used to estimate depth and improve obstacle avoidance.

7. FUTURE ENHANCEMENTS

Future enhancements for our mini self-driving cars could include:

- Improved sensors: Enhanced sensors such as LIDAR, cameras, and radar could provide more accurate data for better decision making and driving performance.
- Advanced AI algorithms: The use of imitation learning algorithms could further improve the car's ability to detect and respond to changing road conditions.
- More power-efficient components: Continued development of more energy-efficient components, such as electric motors and batteries, could extend the car's range and reduce dependence on charging infrastructure.
- More compact design: Our mini self-driving car could become even smaller and more lightweight, making them more accessible for a wider range of users.
- Improved safety features: Enhanced safety features, such as obstacle detection, automatic emergency braking, and backup systems.

8. CONCLUSION

In conclusion, our mini self-driving car project has been a successful demonstration of the capabilities of autonomous vehicle technology. The implementation of a speed limit system, obstacle detection and its distance calculation, traffic sign and stop sign follow and lane following in the car has shown that it is possible to create a small and affordable self-driving car that is capable of taking decisions itself. The project has also highlighted some of the challenges and limitations that still need to be overcome in order to make self-driving cars more widespread and accessible. Despite the limitations, the results of this project have been encouraging and have shown that it is possible to develop a mini self-driving car that can be used for various purposes, such as educational and research purposes. This project has also demonstrated the potential for future advancements in the field of autonomous vehicles, including the development of more advanced control systems, improved sensors, and increased computational power.

9. APPENDICES

Appendix A: GPIO Pinout Diagram

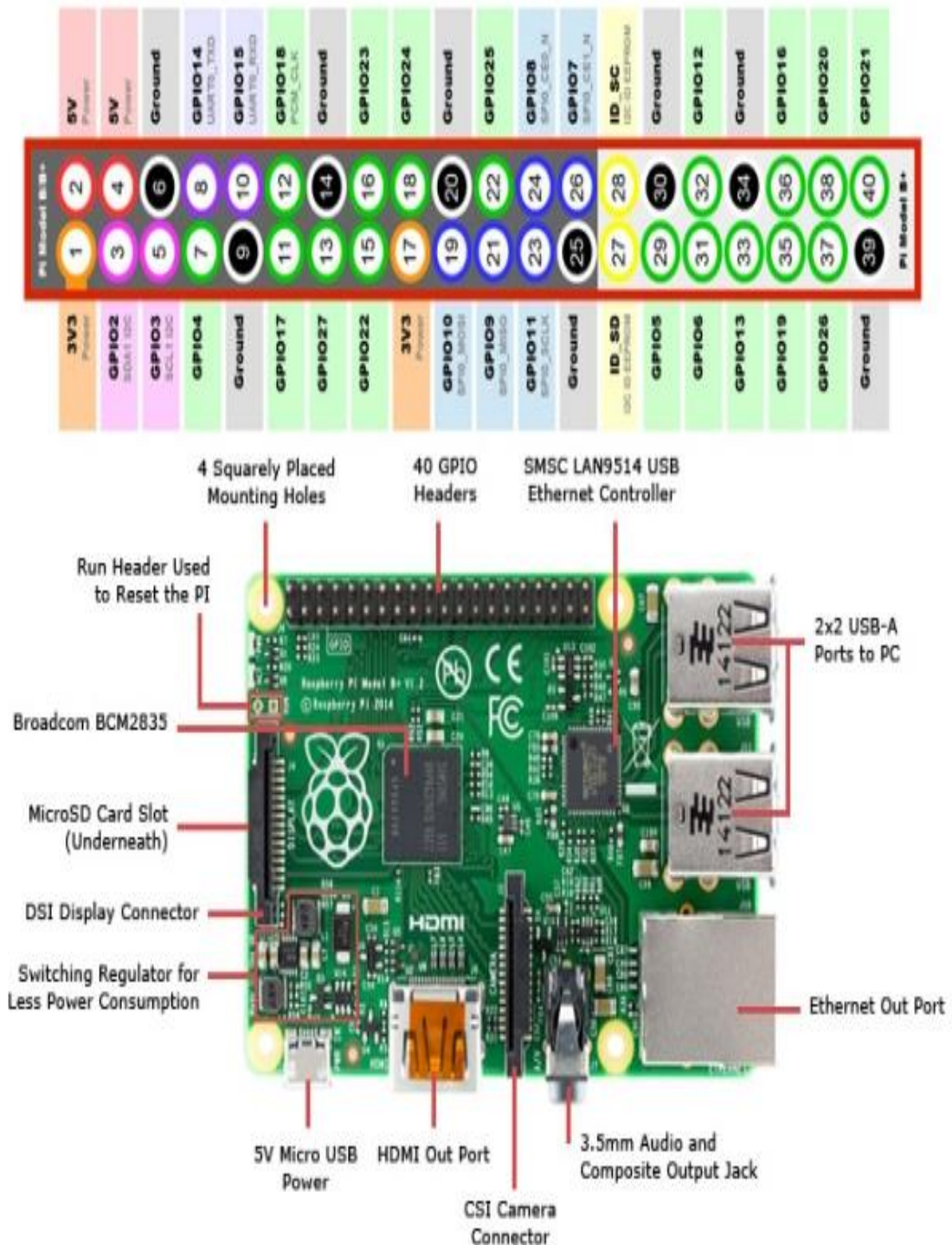


Figure 9- 1: Raspberry Pi 4B GPIO Pinout Diagram

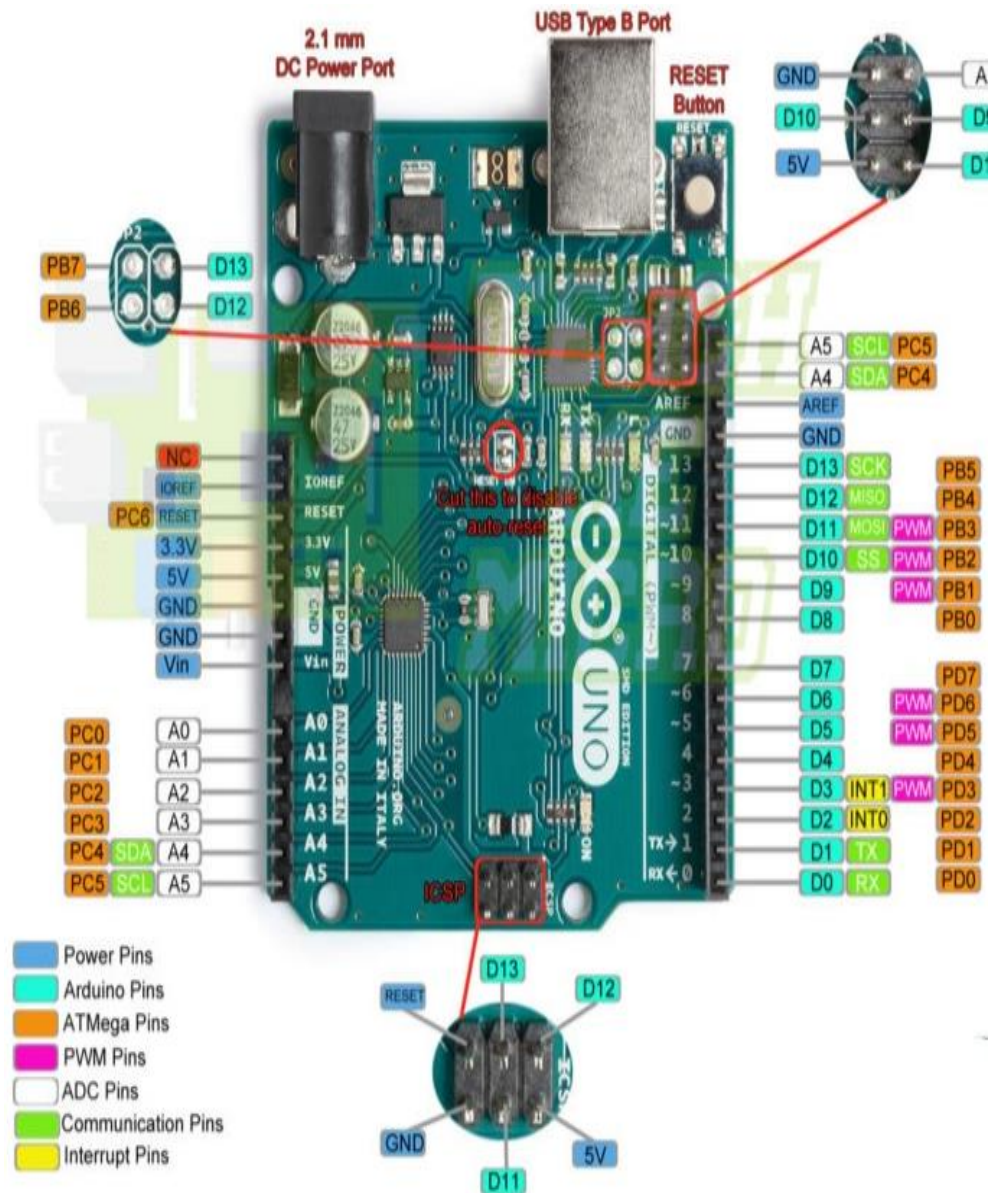


Figure 9- 2: Arduino Uno GPIO Pinout Diagram

Appendix B: Hardware Specifications

Table 9- 1: Raspberry Pi 4B Specifications

CPU	Quad core Cortex-A72 (64-bit) @ 1.5GHz
GPU	H264 (1080p60 decode, 1080p30 encode) OpenGL ES 3.0 graphics, H.265 (4kp60 decode)
RAM	1GB, 2GB, 4GB.
Operating Voltage Range	5V with 3A minimum
GPIO PORTS	28 I/O Pins
LAN	Available
PoE	Enable
WIFI	Available
Bluetooth	5.0
SD Card	Available
HDMI	2- Port with 4k Display (mini-HDMI)
PWR Exp Header	Not Available
Power Source	DC Power Jack, mini USB-C Port
Expansion Connectors	40 Pins (SPI, I ² C, LCD, UART, PWM, SDIO)
USB	2×2.0, 2×3.0
Camera	CSI

Table 9- 2: Arduino Uno Specifications

Microcontroller	ATmega328
Clock Speed	16MHz
Operating Voltage	5V
Maximum supply Voltage (not recommended)	20V
Supply Voltage (recommended)	7-12V
Analog Input Pins	6
Digital Input/Output Pins	14
DC Current per Input/Output Pin	40mA
DC Current in 3.3V Pin	50mA
SRAM	2KB
EEPROM	1KB
Flash Memory	32KB of which 0.5KB used by boot loader

Appendix C: Budget Estimation

Table 9- 3: Budget Estimation

Hardware	Quantity	Price(Rs)
Raspberry Pi 4B	1	20000.00
Raspberry Pi Camera	1	3000.00
DC Motor	4	500.00
L292D Motor Driver	2	500.00
Arduino Uno	1	1100.00
Ultrasonic Sensor	1	500.00
Lithium-ion battery	3	1000.00
Buck Convertor	1	400.00
18650 Battery	4	1000.00
Bread board	2	500.00
Miscellaneous		3000.00
Total		31,500.00

References

- [1] "Road Accidents," The Himalayan Times, Kathmandu, 2021.
- [2] M. Thiagarajan, "Self Driving Car," International Journal of Psychosocial Rehabilitation, 2020.
- [3] Yashi Fatha Abed, H M Ashiqul M Islam, Anowarul Azim, Md. Abeed Hasan, "Automated Self Driving Vehicle," 2021.
- [4] Y. W. G. Deng, "Double Lane Line Edge Detection Method Based on Constraint Conditions Hough Transform," Wuxi, 2018.
- [5] Praval Kumar, Shivam Shandilya, Tushar Sachan and Mr. Nizam Uddin Khan, "Self-driving Car Using Soft Computing," 2019.
- [6] K. Zieba, "End to End Learning for Self-Driving Cars," 2016.
- [7] Muhammad Muteeb Armaghan, Haroon Atiq, Muhammad Ali, Muhammad Tayyab, "Autonomous vehicle prototype," 2021.
- [8] B. Gringer, "History of the Autonomous Car," 2021.
- [9] Salvatore Nicosia, Nickolas Schiffer, Danny Nuch, Ayesha Siddiqua, Andrew Kwon, "S20: Tesla Model RC," 2020.
- [10] Ammar N. Abbas, Muhammad Asad Irshad, "Trajectory Control for Autonomous Vehicles," Giza, 2021.
- [11] C. Ma , M. Xie, " A Method for Lane Detection Based on Color Clustering," 2010.