LANE FOLLOWING MOBILE ROBOT

A project report submitted in partial fulfillment of

the requirements of the Degree of

Bachelor of Engineering

In

(Electronics Engineering)

By

CHINMAY BHOIR	**TU1F1819006**
MAHENDRA GUDLA	**TU1F1819009**
SUCHITA BOGA	**TU1S1920004**
SANDHYA YADAV	**TU1S1920008**

Under the guidance of

Mr. Vijaykumar Chaudhari

Department of Electronics Engineering

TERNA ENGINEERING COLLEGE, NAVI MUMBAI

University of Mumbai

May 2022

CERTIFICATE

This is to certify that

CHINMAY BHOIR	**TU1F1819006**
MAHENDRA GUDLA	**TU1F1819009**
SUCHITA BOGA	**TU1S1920004**
SANDHYA YADAV	**TU1S1920008**

Have successfully completed project titled

LANE FOLLOWING MOBILE ROBOT

In partial fulfillment of Degree Course in

ELECTRONICS ENGINEERING

2021-2022

UNDER THE GUIDANCE OF

Mr. Vijaykumar Chaudhari

Project Guide **Mr. Vijaykumar Chaudhari**	**Project Co-Ordinator** **Mrs. Renuka Chimankare**	**External Examiner**

Head of Department **Dr. B.G. Hogade**	**Principal** **Dr. L. K. Ragha**

Project report approval for BE

This project report entitled **"LANE FOLLOWING MOBILE ROBOT"**

by

CHINMAY BHOIR	**TU1F1819006**
MAHENDRA GUDLA	**TU1F1819009**
SUCHITA BOGA	**TU1S1920004**
SANDHYA YADAV	**TU1S1920008**

Is approved for the degree of **Electronics Engineering**

Examiners

1. ----------

2. ----------

Declaration

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not taken when needed.

CHINMAY BHOIR **MAHENDRA GUDLA**

SUCHITA BOGA **SANDHYA YADAV**

Date:

Place:

Acknowledgement

We would like to express our sincere gratitude towards our guide **Mr. Vijaykumar Chaudhari** for the help, guidance and encouragement, he provided during the BE Dissertation. This work would have not been possible without his valuable time, patience and motivation. We thank him for making our stint thoroughly pleasant and enriching. It was great learning and an honor being his student.

We are deeply indebted to **Dr. B. G. Hogade, Head of Electronics Engineering Department** and the entire team in the Electronics Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work.

We take the privilege to express our sincere thanks to **Dr. L.K Ragha,** our Principal & Project Co-Ordinator **Mrs. Renuka Chimankare** for providing the encouragement and support throughout our work.

Last but not the least we would like to thank all the helping hands which directly or indirectly helped us in our project.

Mr. Chinmay Bhoir
Mr. Mahendra Gudla
Ms. Suchita Boga
Ms. Sandhya Yadav

Abstract

Lane Following is one of the most important aspects of robotics. A Lane Following Mobile Robot is an autonomous robot which is able to follow a lane that is drawn on the surface consisting of a contrasting color. It is designed to move automatically and follow the lane. The robot uses Raspberry Pi Camera to identify the lane and thus assisting the robot to stay on the track. The robot is driven by DC gear motors to control the movement of the wheels. The Arduino Nano interface is used to perform and implement algorithms to control the speed of the motors and to control steering of the robot to travel along the lane. Computer vision and sensor fusion are parts and parcel of the auto driving. Based on the camera results the car will be moved. This project aims to implement the algorithm and control the movement of the robot by proper control parameters and thus achieving better performance. This ensures robust lane fitting even if the road plane changes and also detects obstacles in between the lane and takes action accordingly. It can be used for industrial automated equipment carriers, small household applications, tour guides in museums.

Keywords: *robotics, lane following, image processing, autonomous.*

TABLE OF CONTENT

i. Abstract	1
ii. List of Figures	3
iii. List of Tables	4
CHAPTER 1 INTRODUCTION	
1.1 Introduction	6
1.2 Background	7
1.3 Motivation	7
CHAPTER 2 LITERATURE REVIEW	
2.1 Literature Survey	8
CHAPTER 3 PROBLEM DEFINITION AND STATEMENT	
3.1 Problem Statement	12
3.2 Scope of the project	12
CHAPTER 4 METHODOLOGY	
4.1 Proposed System Overview	13
4.2 System Design	15
4.2.1 Block Diagram	15
4.2.2 Logical Flowchart	16
4.2.3 Circuit Diagram	19
4.3 System Requirements	20
4.3.1 Hardware requirements	20
4.3.1.1 Chassis	20
4.3.1.2 Raspberry Pi Model 3B+	21
4.3.1.3 Arduino Board	22
4.3.1.4 Raspberry Pi Camera	23
4.3.1.5 Motor driver	23
4.3.1.6 DC gear motor	24
4.3.1.7 Power bank	25
4.3.1.8 Battery	26
4.3.2 Software requirements	27
4.3.2.1 Raspberry Pi OS	27
4.3.2.2 Arduino IDE	27
CHAPTER 5 IMPLEMENTATION	
5.1 Implementation	28

CHAPTER 6	
6.1 Limitations	30
6.2 Future Scope	30
CHAPTER 7 RESULTS AND CONCLUSION	
7.1 Results	31
7.2 Conclusion	34
References	35
Component Datasheet reference	36
Appendix	37

List of Figures

Figure Number	Figure Name	Page Number
Figure 4.2.1	Block diagram	15
Figure 4.2.2	System flowchart	16
Figure 4.2.3	Image Processing flowchart	17
Figure 4.2.4	Motor controller flowchart	18
Figure 4.2.5	Circuit diagram	19
Figure 4.3.1	Robot chassis	20
Figure 4.3.2	Raspberry Pi model & pin configuration	21
Figure 4.3.3	Arduino Nano & pin configuration	23
Figure 4.3.4	Raspberry Pi Camera	23
Figure 4.3.5	L298N H-Bridge Motor Driver	24
Figure 4.3.6	DC Gear Motor	25
Figure 4.3.7	Power Bank	26
Figure 4.3.8	18650 Li-ion Battery	26
Figure 5.1.1	Front View of Prototype	28
Figure 5.1.2	Back View of Prototype	29
Figure 5.1.3	Design of Lane	29
Figure 7.1.1	Robot moving Forward Direction	31
Figure 7.1.2	Result showing Forward Direction	31
Figure 7.1.3	Robot moving Left Direction	32
Figure 7.1.4	Result showing Left Direction	32
Figure 7.1.5	Robot moving Right Direction	33
Figure 7.1.6	Result showing Right Direction	33
Figure 7.1.7	Robot stopped at Lane End	34

List of Tables

Table Number	Table Name	Page Number
2.1.1	Literature Survey	11
4.3.1	Raspberry Pi description	21
4.3.2	Arduino Nano description	22
4.3.3	Motor Driver description	24
4.3.4	DC Motor description	25
4.3.5	Power bank description	25
4.3.6	18650 Battery description	26

CHAPTER 1 INTRODUCTION

1.1 Introduction

Robotics is the branch of technology which deals with the construction, design, operation and application of robots as well as computer systems for their control, sensory feedback and information processing. Mobile robots are generally those robots which can move from place to place to perform desired and complex tasks, simple but time intensive, repetitive. There are different constructive variants of mobile robots, but lately with the mass development of microcomputers and embedded systems, it is possible to develop low-cost solutions. Depending on their destination, mobile robots may have different configurations, multiple sensors (infrared, ultrasonic, webcams, etc.) or different command and control algorithms, being locally or remotely supervised. This kind of robots have become a concrete reality and may pave the way for future system were computers take over the art of driving. Many existing algorithms like lane detection, obstacle detection is combined together to provide the necessary control to the robot. Detecting the lane and maneuvering the robot to stay on course, while constantly correcting wrong moves forms a simple yet effective system. Camera-based Lane detection is used to detect the lanes and position the robot in between the lanes properly and prevent the lane departure. Cameras provide us with a powerful and cheaper way of extracting information about our environment. Computer vision is at the core of perception algorithms and cameras are the closest equivalent technology to how humans perceive the world around them.

In this project, we have proposed a mobile robot platform which has a fixed four-wheel chassis. The platform was designed around the Raspberry Pi single board computer and uses Arduino Nano to control the movement of the chassis. It uses camera sensor and modules to determine the lane and follow the path. The path can be visible like a white line on a gray surface. Sensing a lane and maneuvering the robot to stay on course, while constantly correcting wrong moves using feedback from the sensor forms a simple yet effective system. To achieve autonomous navigation of mobile robot, perception of surroundings, driving path planning and robot control algorithm are key features. While lane detection and obstacles recognition using Raspberry Pi camera in real robot systems, it is hard to compute the distance between robot and obstacles or lane. The communication protocol between the components and the features implemented on the platform are presented.

1.2 Background

A Robot is any machine which is completely automatic, i.e. it starts on its own, decides its own way of work and stops on its own. It is actually a replica of human being, which has been designed to ease human burden. Robots can be fixed robots or mobile robots. Mobile Robots are robots with a mobile base which makes the robot move freely in the environment. One of the advanced mobile robots is the Lane Following Robot. It is basically a robot which follows a particular path or trajectory and decides its own course of action which interacts with obstacles.

As technology becomes increasingly important in today's world, it is invaluable to not onlylearn how to use technology, but also to understand how to create it.

- It gives visual grasp of math and science.
- It builds logical thinking.
- It brings out innovation and creativity.
- It enhances problem solving skills.

1.3 Motivation

Ants always travel in a line, following an invisible route in search of food, or back home. A robot that runs on line also. Which is a perfect or near the artificial nature. The purpose of this projectis to recreate in terms of machines what one sees around to solve a problem or fulfill a requirement.

The area will be benefitted from the project:

- Industrial automated equipment carriers.
- Entertainment and small household applications.
- Tour guides in museums and other similar applications.

CHAPTER 2 LITERATURE REVIEW

2.1 Literature Survey

1. PAPER 1

PAPER: Autonomous Self-Driving Car using Raspberry Pi Model

AUTHOR: Mr. Nihal A Shetty, Mr. Mohan K, Mr. Kaushik K

PUBLICATION: International Journal of Engineering Research & Technology (IJERT)

ISSN: 2278-0181, Volume 7, Issue 08

Self-driving cars are already being implemented in foreign countries however these cannot be implemented in India. Reason being these existing approaches uses GPS, Sensors. The problem with GPS is that these display roads on the map that might or might not exist and also these roads in India might not be a concrete road. To implement a self-driving vehicle which uses a pattern matching technique to overcome the problem. In this project, they planned of using a special pattern which will be deployed on the road. These patterns are a special pattern that is used for detection of the pathway and it detects the type of road. Hence using this technology, we can implement a self-driving car in India. Their prototype has used a modelled car which has a Raspberry Pi to process the captured images from the camera and send it remotely on remote computer process it and send back. Similarly, we have various sensors around the car to detect the surrounding obstacles. The camera will be able to capture specific pattern on the road. The pattern is like a pathway for the modelled car that makes it easier to drive on roads in India. Our prototype uses a hybrid combination of the existing technology as well as the newly implemented methodology of detecting special pattern marked on the road for providing better results.

Results: The problem of non-autonomous vehicles with the proposed system which reduces the human work of operating the vehicle. Furthermore, we also notice that the given system performance is much better than an average user. Since the performance is better and always consistence, hereby come to a conclusion that the proposed system can solve the basic human error that occurs. The working of the model where it is able to detect the special pattern provided and also is able to detect the obstacles in the surrounding.

2. **PAPER 2**

PAPER: A lane detection method based on 3D-LiDAR

AUTHOR: Yu-Fang Wang and Yi-Shueh Tsai

PUBLICATION: Automotive Research & Testing Center (ARTC), Changhua, Taiwan (R.O.C)

E-ISSN: 2018

Lane detection is crucial information for driving autonomy. To build a safe and robust lane detection system, 3D LiDAR based lane detection, capable of detecting all direction and working in all lighting condition, is an ideal sensor redundancy in addition to camera-based lane detection. The LiDAR can detect the intensity of road surface points, so the lane mark will appear as high intensity point segment in each scan layer. The regrouping of clusters into lanes is a challenging task due to the discontinuity of road lane, the variety of road direction and configuration, and the presence of other road marks beside lane marks. A method has been proposed to incorporate road edge detection to predict local road geometry model which assists mark-segment regrouping into lane and unrelated mark filtering. Experiments show the method is efficient, and can run in a real-time environment.

Results: In this study, the method is used by LiDAR sensor rather than camera which common on lane detection. According to the feature of road geometry, the slope, continuity and smoothness of the point clouds are considered to be analyzed to define the definite road surface and boundary. Since the road boundary is definite, the relationship between lane and boundary is easy to construct. The direction of lane is almost identical as the road boundary in ideal environment. Thus, the road surface searching by this method could define as the whole lane, and the central lane line could divide the lane into left part and right part.

3. **PAPER 3**

PAPER: Lane Detection Algorithm Using LRF for Autonomous Navigation of Mobile Robot

AUTHOR: Jong-Ho Han and Hyun-Woo Kim

PUBLICATION: Test & Evaluation Division, Korea

E-ISSN: Appl. Sci. 2021, 11, 6229. https://doi.org/10.3390/app11136229

This paper proposes a lane detection algorithm using a laser range finder (LRF) for the autonomous navigation of a mobile robot. The lane detection is a fundamental requirement for an automobile system that utilizes the external environment information of automobiles. Representative methods of lane recognition are vision-based and LRF-based systems. In the case of a vision-based system, the recognition of the environment of a three-dimensional space becomes excellent only in good conditions for capturing images. In this paper, a three-dimensional lane detection algorithm using LRF that is very robust against illumination is proposed. For the three-dimensional lane detection, the laser reflection difference between the asphalt and the lane according to color and distance has been utilized with the extraction of feature points.

Results: This paper proposed a real time lane detection algorithm using LRF (Laser Range Finder) for autonomous navigation of a mobile robot. There are some unexpected barriers, such as bad illumination, occlusions, and vibrations that the vision cannot be used for satisfying the fundamental requirement, and conventional lane detection has mainly been carried out by using vision-based methods, but such methods have a serious drawback of showing substantially diminished performance in driving environments where reliable vision-based information is not obtained, such as under conditions of dense fog. Therefore, in this study, we built a laser-scanned 3D road map and discerned and recognized lanes by using a feature point extraction algorithm using LRF calibration and amplification errors depending on the materials and colors of the asphalt and the lanes. The test results confirmed that the use of the proposed method could ensure safe driving under unfavorable road conditions such as fog, which could contribute to the R&D on autonomous driving technologies.

4. PAPER 4

PAPER: Vision-Based Adaptive Cruise Control Using Pattern Matching

AUTHOR: Ritesh Kanjee, Asheer K. Bachoo, Johnson Carroll

PUBLICATION: 2013 6th Robotics and Mechatronics Conference (RobMech) Durban, South Africa

E-ISSN: 978-1-4799-1518-7/13/$31.00 ©2013 IEEE

Adaptive Cruise Control (ACC) is a relatively new system designed to assist automobile drivers in maintaining a safe following distance. This paper proposes and validates a vision-based ACC system which uses a single camera to obtain the clearance distance between the preceding vehicle and the ACC vehicle. Pattern matching, with the aid of

lane detection, is used for vehicle detection. The vehicle and range detection algorithms are validated using real-world data, and then the resulting system performance is shown to be sufficient using a simulation of a basic vehicle model

Results: This study demonstrates that a vision-based range finding system can be used for adaptive cruise control systems, and that pattern matching is a reasonable method to explore for future system development. Future work can improve on multiple areas of this project. As with any such system, increased performance of the computing platform or of the nature of the algorithms would allow for more robust performance.

PAPER	AUTHOR	PUBLICATIONS	E-ISSN	SENSORS & BOARDS
Autonomous Self-Driving Car using Raspberry Pi Model	Mr. Nihal A Shetty, Mr. Mohan K, Mr. Kaushik K	International Journal of Engineering Research & Technology (IJERT)	2278-0181, Volume 7, Issue 08	Raspberry Pi, Arduino UNO & camera
A lane detection method based on 3D-LiDAR	Yu-Fang Wang and Yi-Shueh Tsai	Automotive Research & Testing Center (ARTC), Changhua, Taiwan (R.O.C)	2018	3D-LiDAR
Lane Detection Algorithm Using LRF for Autonomous Navigation of Mobile Robot	Jong-Ho Han and Hyun-Woo Kim	Test & Evaluation Division, Korea	Appl. Sci. 2021, 11, 6229. https://doi.org/10.3390/app11136229	3D map, laser range finder
Vision-Based Adaptive Cruise Control Using Pattern Matching	Ritesh Kanjee, Asheer K. Bachoo, Johnson Carroll	2013 6th Robotics and Mechatronics Conference (RobMech) Durban, South Africa	978-1-4799-1518-7/13/$31.00 ©2013 IEEE	PI Controller, Autonomous Vehicle, Single Camera

Table 2.1.1 Literature Survey

CHAPTER 3 PROBLEM DEFINITION & STATEMENT

3.1 Problem Statement

In the industries, carriers are required to carry products from one point to another point which areusually in separate buildings or blocks. Conventionally, carts or trucks were used with human drivers. Unreliability and inefficiency in this part of the assembly line formed the weakest link.

The task is to build an autonomous car that can basically control or change the course of directionwithout any human input. One of the use cases are in industries where certain goods are being placed in from one area to another and using this system, we can properly reduce manpower andtime delay.

3.2 Scope of the Project

This Project focuses at designing and building the lane following robot and implement obstacle detection. Therefore, this robot will cover the scopes as follows:

- Design lane following robot using Raspberry pi and Arduino Nano.
- Lane following algorithm is implemented using OpenCV Library.
- Raspberry Pi Camera is used for both Lane Detection and Obstacle Detection.

CHAPTER 4 METHODOLOGY

4.1 Proposed System Overview

Robotics is an interesting subject to discuss about and in this advanced world robots are becoming a part of our life. In this project, we have discussed about a robot which is capable of following a lane without the help of any external source. The Lane following Mobile Robot uses four DC motors to control rear and front wheels. It has Raspberry Pi Camera Sensor for detection of white lines. The proposed system detects the drivable region and road lines drawn on the floor from Camera data. The Path consists of a white line on a gray surface. The control system senses these lines and manoeuvre the robot to stay on course.

The proposed system consist of following terms:

4.1.1 Computer Vision

Computer vision is an interdisciplinary field, in which we are trying to describe the world that we see in one or more images and to reconstruct its properties, such as shape, illumination, and color distributions. In other words, computer vision provides us the understanding of the scene.

4.1.2 Digital Image Processing

Image processing is a type of signal processing, which has an image, series of images or video frames as the input, while the output is processed images, maybe in a different format. Image processing is often exploited in computer vision to manipulate digital images. Vision can be considered the most important part of human perception and it plays the same role in machine sensing. However, unlike human, machines "see" the world as digital images of 2D or 3D scenes produced by sensors. The 2D digital image, which, in many cases, represents a projection of a 3D scene, may be defined as a two-dimensional array of intensity samples called pixels. Pixel values typically represent gray levels, colors, opacities, etc. 2D digital image can be processed and manipulated by computer programs to produce useful information about the world. Several techniques in digital image processing are noise removal, image blur, image sharpening, object recognition, etc.

4.1.3 OpenCV Library

Most of the image processing techniques in this thesis, for example image reading, gray scaling, thresholding, etc., are realized by using Open-Source Computer Vision Library (OpenCV), which is the most popular open-source library in computer vision. OpenCV (Open-Source Computer Vision Library) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. OpenCV is written in optimized C/C++ language and has a modular structure.

Listed below are several examples of using OpenCV in image processing:

- **Gaussian Smoothing or Gaussian Blurring**: the most commonly used blurring method which blurs an image using a Gaussian filter.
- **Grayscaling:** converts an image from RGB color space to grayscale.
- **Thresholding:** transforms a grayscale image to a binary image.

4.1.4 Region of Interest

A Region of Interest (ROI) is a portion of an image that you want to filter or operate on in some way.

4.1.5 Canny Edge Detection

Canny edge detector is based on Gaussian blur and sobel masking method. Sobel edge masking is a kind of 1st order derivation operator, it can detect edge of every direction and normalization of every pixel data.

When implementing the project using camera with raspberry pi for image processing, we analyze the frame in seconds for finding the lane and there are few steps at first that are executed.

Step 1: Captures the image and then that image is converted from BGR to RGB as image with RGB format works best for analyzing as RGB image is simply a composite of three independent grayscale image.

Step 2: Region of Interest (ROI) is created for every image. As region of interest focuses on some portion of image. The Region of Interest covers most of lane portion of the frame.

Step 3: Another frame is created from that Region of Interest area of the frame to analyze the lane and the surroundings which basically called as perspective transformation.

Step 4: Created frame is converted to grayscale image using threshold operations where the gray portion of lane is given more threshold value and the noise is given less threshold value and as we see the lane clearly from the frame the distance between the lane center and right, left lane is determined. The distance is then stored and the data for Arduino to control the motors is evaluated using if-else conditions.

Step 5: The data from Raspberry pi is given to Arduino using four digital pins and by making a whole binary format from those pins for Arduino to control the motors. The binary format value is initialized to the variable and using the variable, left, right and stop functions are implemented.

4.2 System Design

4.2.1 Block Diagram

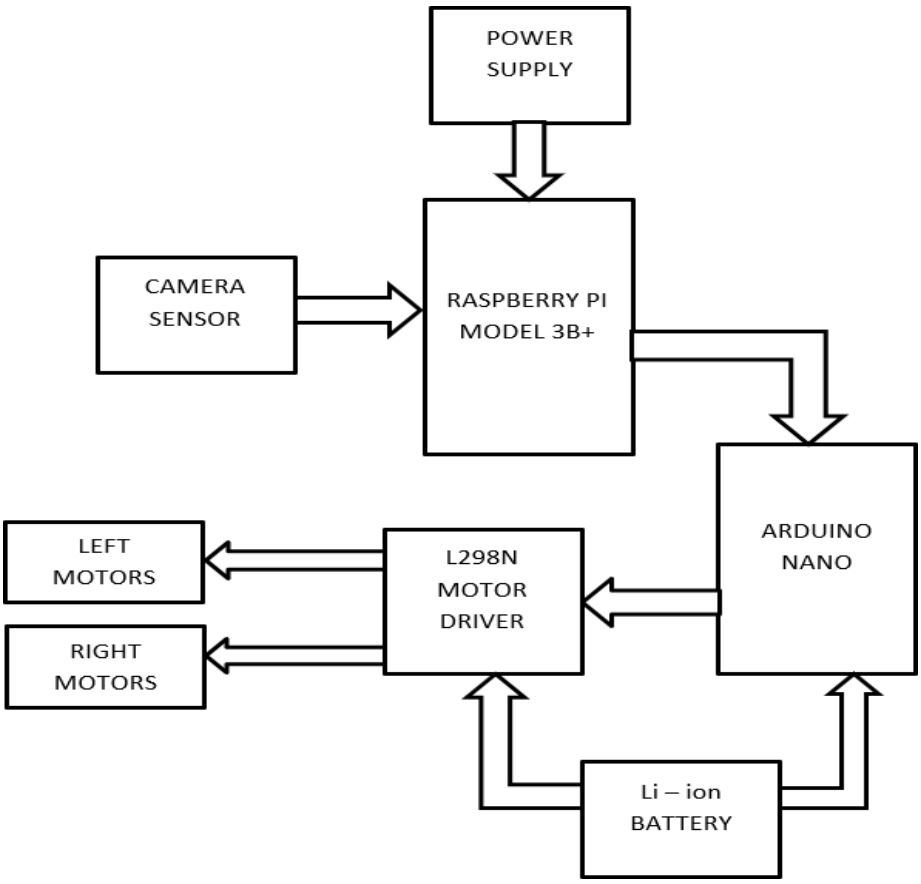

Figure 4.2.1 Block diagram

In the above block diagram, the camera sensor is used as the input device and DC gear motor as output device. Input device is connected to the Raspberry Pi and output device is connected tor Arduino Nano through Motor driver. Camera streams the video and send it to the Raspberry Pi. Raspberry Pi processes the data and transmits the processed data to Arduino Nano that further sends the information to the modelled robot, so it can operate based on the condition.

4.2.2 Logical Flowchart

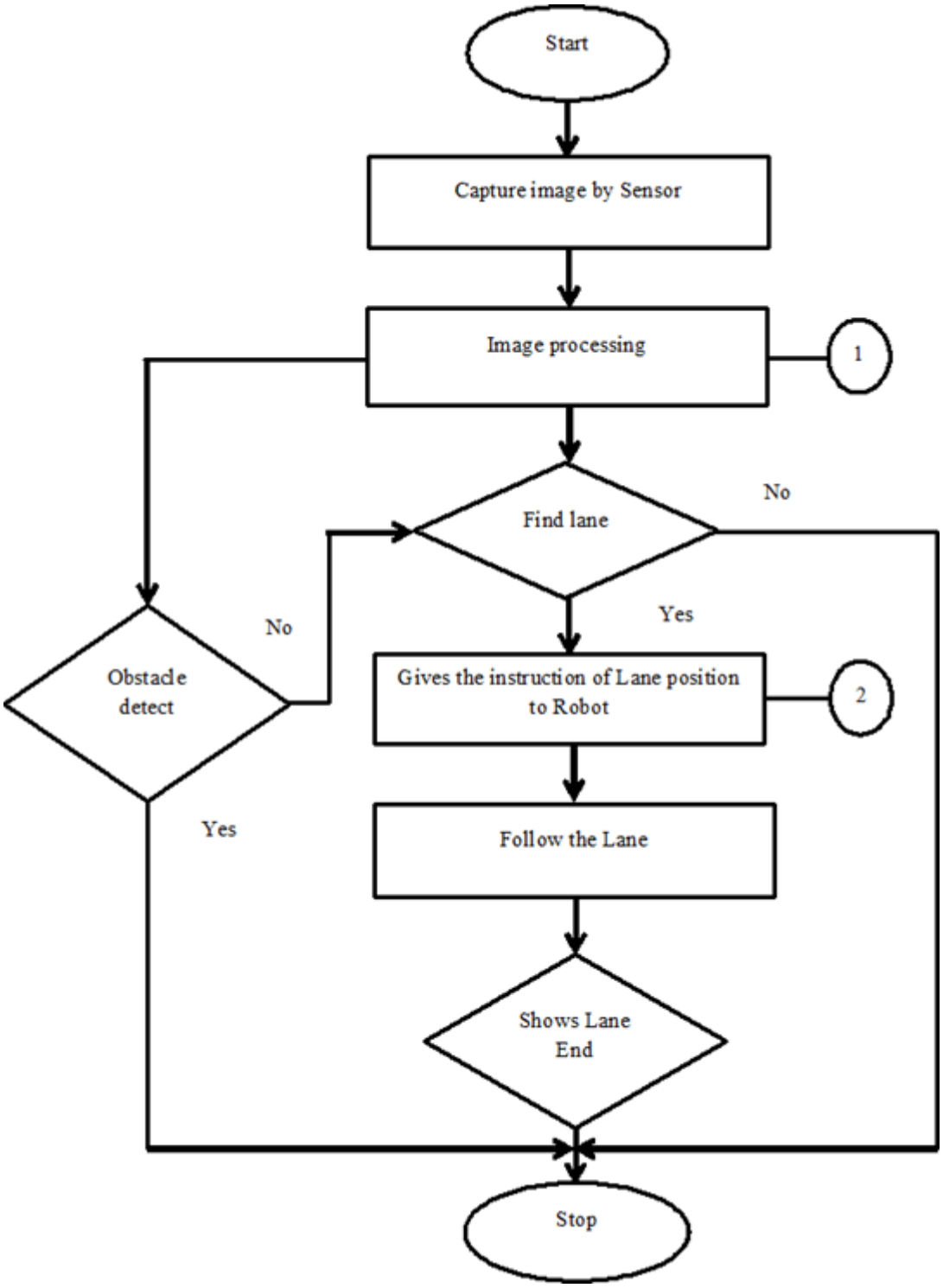

Figure 4.2.2 System flowchart

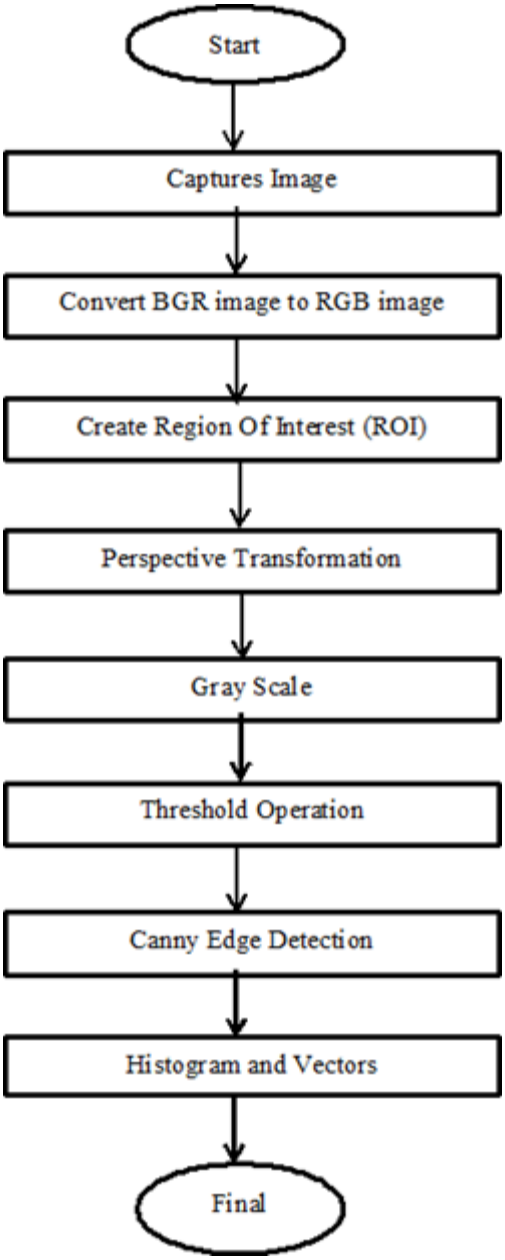

Figure 4.2.3 Image processing flowchart

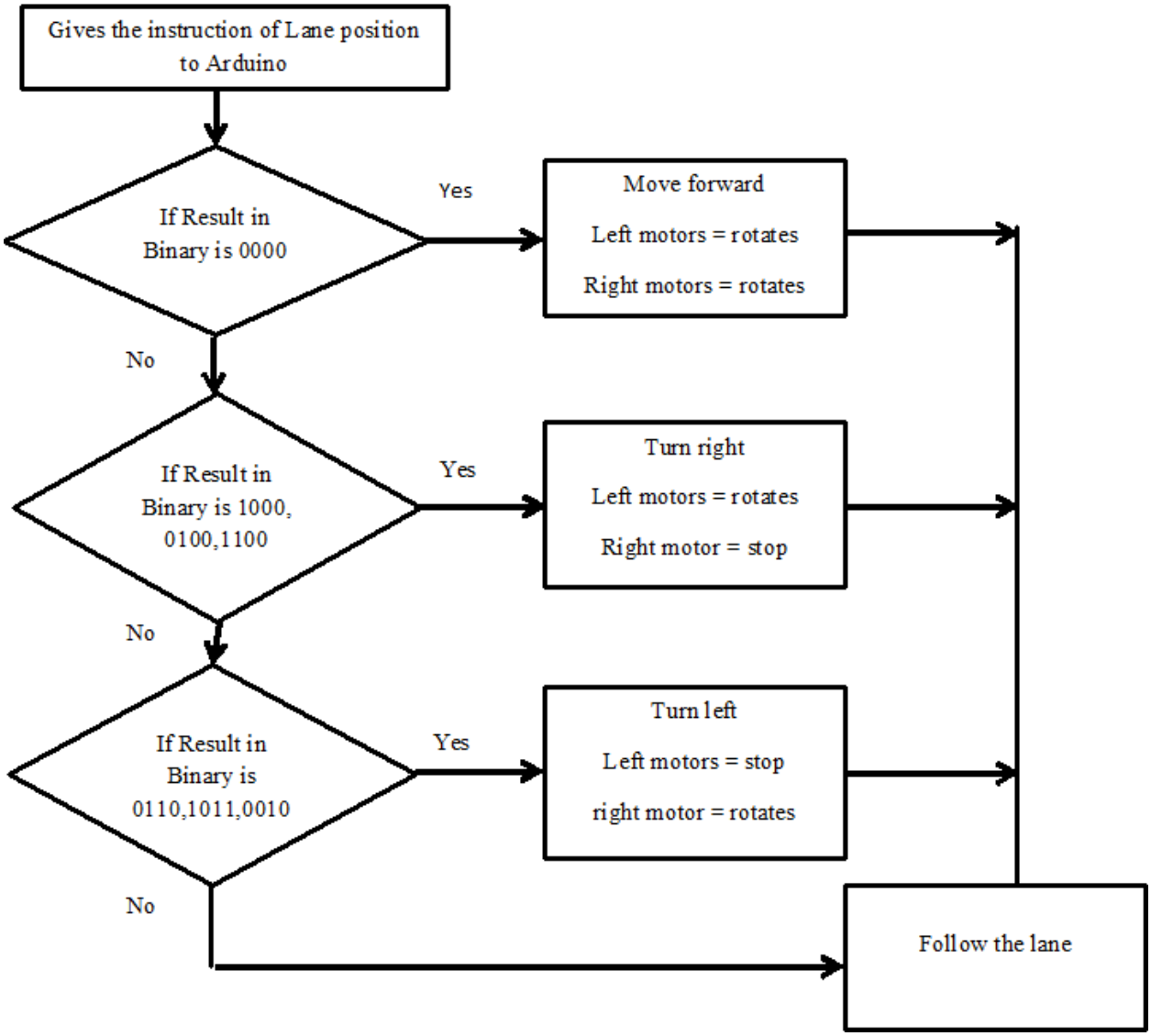

Figure 4.2.4 Motor controller flowchart

Explanation of Flow of Project

1. Initially, camera captures the image of the lane.
2. After the image processing takes place to find the edge of the lane and also detect the obstacles in between the lane
   a. It converts the image from BGR format to RGB format.
   b. Create region of interest.
   c. Find perspective view.
   d. Convert it into Gray scale and apply threshold operation and find edges of the lane.
3. After getting the lane position it gives the instructions to the robot accordingly.

a. If result value is 0000, robot moves forward.
b. If result value is 1000 or 0100 or 1100, robot will move to right side direction.
c. If result value is 0110 or 1011 or 0010, robot will turn to left side direction.

4. According to the above instructions robot will start following the lane.
5. If obstacle is detected within the region of interest, then the robot will stop.
6. Robot will stop when the camera detects the end of the lane.

4.2.3 Circuit Diagram

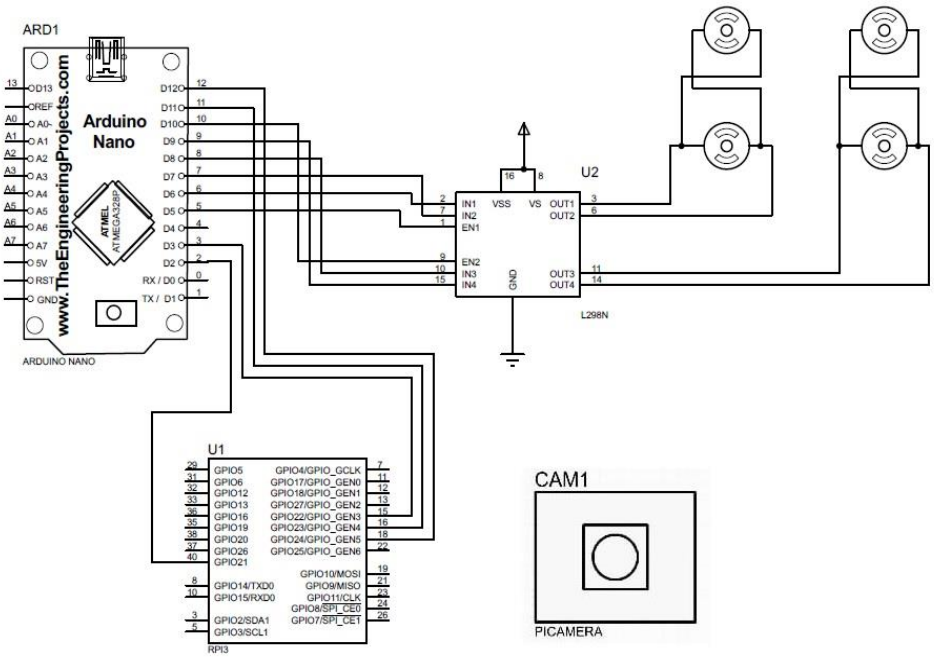

Figure 4.2.5 Circuit diagram

Circuit diagram description

- The above circuit diagram consist of Raspberry Pi camera, Raspberry pi (model 3B+), Arduino Nano, Motor driver and DC motor.
- This system is powered by Power Bank and Battery.
- Raspberry Pi camera is connected to camera module part of raspberry pi to capture image.
- GPIO pins 21, 22, 23 and 24 of Raspberry Pi are connected to the digital pins D2, D3, D11, and D12 of Arduino Nano to send the processed data. The GPIO pins are set as output pins and digital pins of microcontroller are set as input pins.
- EN1, IN1, IN2, IN3, IN4, EN2 pins of Motor driver L298N are connected to the controller pins D5, D6, D7, D8, D9, D10 Arduino Nano for receiving the required instructions from the controller.
- Connect two DC motor in parallel as output. Left side motors are connected to Out1 and Out2 and Right side motors are connected to Out3 and Out4 of Motor Drivers tom operate according to the instructions.

4.3 System Requirements

4.3.1 Hardware Requirements

4.3.1.1 Chassis

It is the base frame of the robot on which the mechanical as well as electronic components are mounted. The chassis is made of acrylic sheet. It is also referred as PMMA sheet. PMMA (Polymethyl Methacrylate) or Acrylic is also known as Acrylic Glass or Plexiglass.

Figure 4.3.1 Robot Chassis

4.3.1.2 Raspberry Pi Model 3 B+

Raspberry pi is a small size computer which have its Raspbian OS. Raspberry Pi is the most important module in this system. The Raspberry Pi is used for image processing to detect the lane using OpenCV Library. It able to do multiple function simultaneously.

Specifications:

	Description
CPU	Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
RAM	1GB LPDDR2 SDRAM
GPIO Header	Extended 40-pin
Wireless	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
Ports	4 USB 2.0 , Wired Ethernet up to 330 Mbps
Camera Port	CSI camera port for connecting a Raspberry Pi camera
Board Dimensions	85 x 56 mm
Multimedia	1080p30
Power Input	5V/2.5A DC
Storage	Micro SD port for loading operating system and storing data
Output	4-pole stereo output and composite video port

Table 4.3.1 Raspberry Pi description

GPIO#	NAME			NAME	GPIO#
	3.3 VDC Power	1	2	5.0 VDC Power	
8	GPIO 8 SDA1 (I2C)	3	4	5.0 VDC Power	
9	GPIO 9 SCL1 (I2C)	5	6	Ground	
7	GPIO 7 GPCLK0	7	8	GPIO 15 TxD (UART)	15
	Ground	9	10	GPIO 16 RxD (UART)	16
0	GPIO 0	11	12	GPIO 1 PCM_CLK/PWM0	1
2	GPIO 2	13	14	Ground	
3	GPIO 3	15	16	GPIO 4	4
	3.3 VDC Power	17	18	GPIO 5	5
12	GPIO 12 MOSI (SPI)	19	20	Ground	
13	GPIO 13 MISO (SPI)	21	22	GPIO 6	6
14	GPIO 14 SCLK (SPI)	23	24	GPIO 10 CE0 (SPI)	10
	Ground	25	26	GPIO 11 CE1 (SPI)	11
30	SDA0 (I2C ID EEPROM)	27	28	SCL0 (I2C ID EEPROM)	31
21	GPIO 21 GPCLK1	29	30	Ground	
22	GPIO 22 GPCLK2	31	32	GPIO 26 PWM0	26
23	GPIO 23 PWM1	33	34	Ground	
24	GPIO 24 PCM_FS/PWM1	35	36	GPIO 27	27
25	GPIO 25	37	38	GPIO 28 PCM_DIN	28
	Ground	39	40	GPIO 29 PCM_DOUT	29

Figure 4.3.2 Raspberry Pi & Pin configuration

4.3.1.3 Arduino Board

Arduino Nano is a single board microcontroller kit for building digital devices and interactive objects in the physical and digital world. It allows to program in C language. Arduino Nano consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. It has internal analog to digital convertor. In this system, Arduino Nano is used to give instructions to motor driver and DC motors. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

Specifications:

	Description
Microcontroller	ATMEGA328P
Architecture	AVR
Operating Voltage	5V
Flash Memory	32 KB of which 2KB used by Bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog in Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm

Table 4.3.2 Arduino Nano description

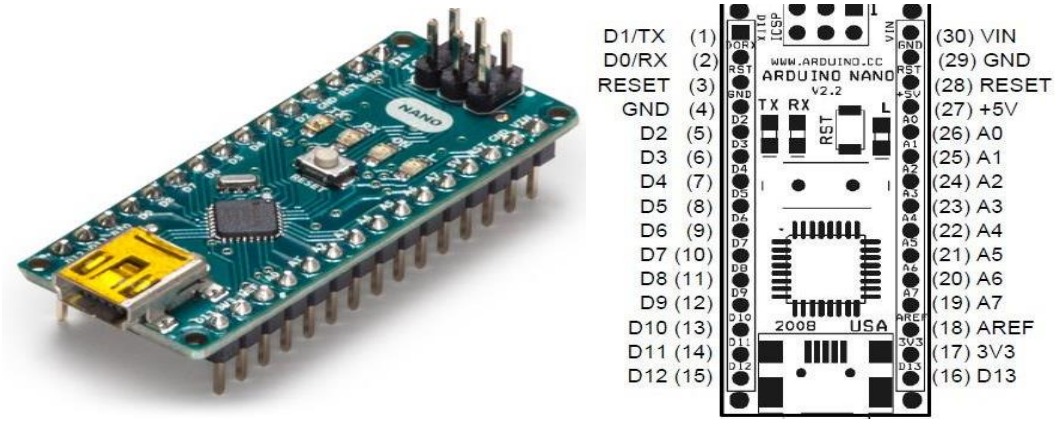

Figure 4.3.3 Arduino Nano & pin configuration

4.3.1.4 Raspberry Pi Camera

The Raspberry Pi 5MP camera board is a custom designed camera board that is equipped with a flexible ribbon cable, making it compatible with Raspberry Pi boards. The camera board is integrated with a fixed lens that has a resolution of 5 megapixels. This camera board can capture moments with a resolution of 2592 x 1944 pixels and can record high quality videos that supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 format. The compact size and the light weight features of the Raspberry Pi camera board makes it compatible and feasible to use as a hidden camera or cameras for Pi phones.

Figure 4.3.4 Raspberry Pi Camera

4.3.1.5 Motor Driver

The L298N is a dual-channel H-Bridge motor driver capable of driving a pair of DC motors. The module can drive DC motors that have voltages between 5 and 35V, with a peak current up to 2A. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

Specifications:

	Description
Driver Module	L298N 2A
Driver Chip	Double H Bridge L298N
Motor Supply Voltage (Maximum)	46V
Motor Supply Current (Maximum)	2A
Logic Voltage	5V
Driver Voltage	5-33V
Driver Current	2A
Logical Current	0-36mA
Maximum Power (W)	25W

Table 4.3.3 Motor Driver Description

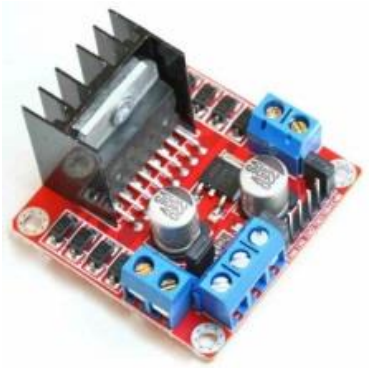

Figure 4.3.5 L298N H-Bridge Motor Driver

4.3.1.6 DC Gear Motor

A DC motor with gear box attached to the shaft, which is mechanically commutated electric motor powered from direct current (DC). Generally used in DIY projects, Battery operated toys, Radio controlled vehicles, Robotic projects etc.

Specifications:

- Low density, lightweight, low inertia.
- Capability to absorb shock and vibration as a result of elastic compliance.
- Ability to operate with minimum or no lubrication

	Description
Motor Supply Voltage (Maximum)	46V
Operating voltage (VDC)	3~12
Shaft length (mm)	8.5
Shaft diameter(mm)	5.5 (doubled type)
No load current	40 – 180mA
Speed (after reduction)	150RPM
Torque	2.5 Kg/cm

Table 4.3.4 DC Motor description

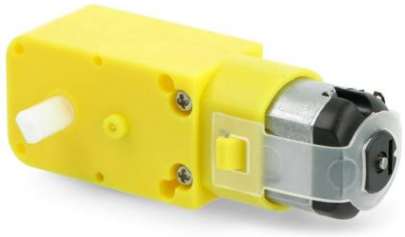

Figure 4.3.6 DC Gear Motor

4.3.1.7 Power Bank

Power bank consist of 20000mAh power comes with inbuilt voltage regulator. It has 3 Output ports. Power bank is used to power up Raspberry Pi due to its output current.

Specifications:

	Description
Input Voltage	5V
Output Voltage	5V
Output Current	3A
Weight	0.5 Kg

Table 4.3.5 Power bank description

Figure 4.3.7 Power Bank

4.3.1.8 Battery

2 cells of 18650 Li - ion battery is used to power Arduino nano, motor driver and DC Motors.

Specifications:

	Description
Voltage	3.7V x 2
Capacity	2500 mAh x 2
Output Current	2.5 A x 2
Battery Type	Lithium Ion
Weight	35 gm
Size	50 mm * 34mm * 4mm

Table 4.3.6 18650 Battery description

Figure 4.3.8 18650 Li-ion Battery

4.3.2 Software Specifications

4.3.2.1 Raspberry Pi OS

Raspberry Pi OS (formerly Raspbian) is a Debian-based operating system for Raspberry Pi. Since 2013, it has been officially provided by the Raspberry Pi Foundation as the primary operating system for the Raspberry Pi family of compact single-board computers.

4.3.2.2 Arduino IDE

The Arduino Integrated Development Environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in functions from C and C++. It is used to write and upload programs to Arduino compatible boards, but also, with the help of third-party cores, other vendor development boards. A range of Arduino modules available including Arduino Uno, Arduino Mega, Arduino Nano, Arduino Leonardo, Arduino Micro and many more. Each of them contains a microcontroller on the board that is actually programmed and accepts the information in the form of code. The main code, also known as a sketch, created on the IDE platform will ultimately generate a Hex File which is then transferred and uploaded in the controller on the board. The IDE environment mainly contains two basic parts: Editor and Compiler where former is used for writing the required code and later is used for compiling and uploading the code into the given Arduino Module. This environment supports both C and C++ languages.

Features of Arduino IDE

1. Sketching editing tools
2. Burn bootloader
3. Libraries
4. Sketches Management
5. Serial monitor
6. Sharing
7. Programming function
8. Auto formatting
9. Board Selection & Management
10. Port menu
11. Project Documentation
12. Sketchbook
13. Sketch Archive
14. Sketches Management
15. Fix Encoding & Reload
16. User Preferences

CHAPTER 5 IMPLEMENTATION

5.1 Implementation

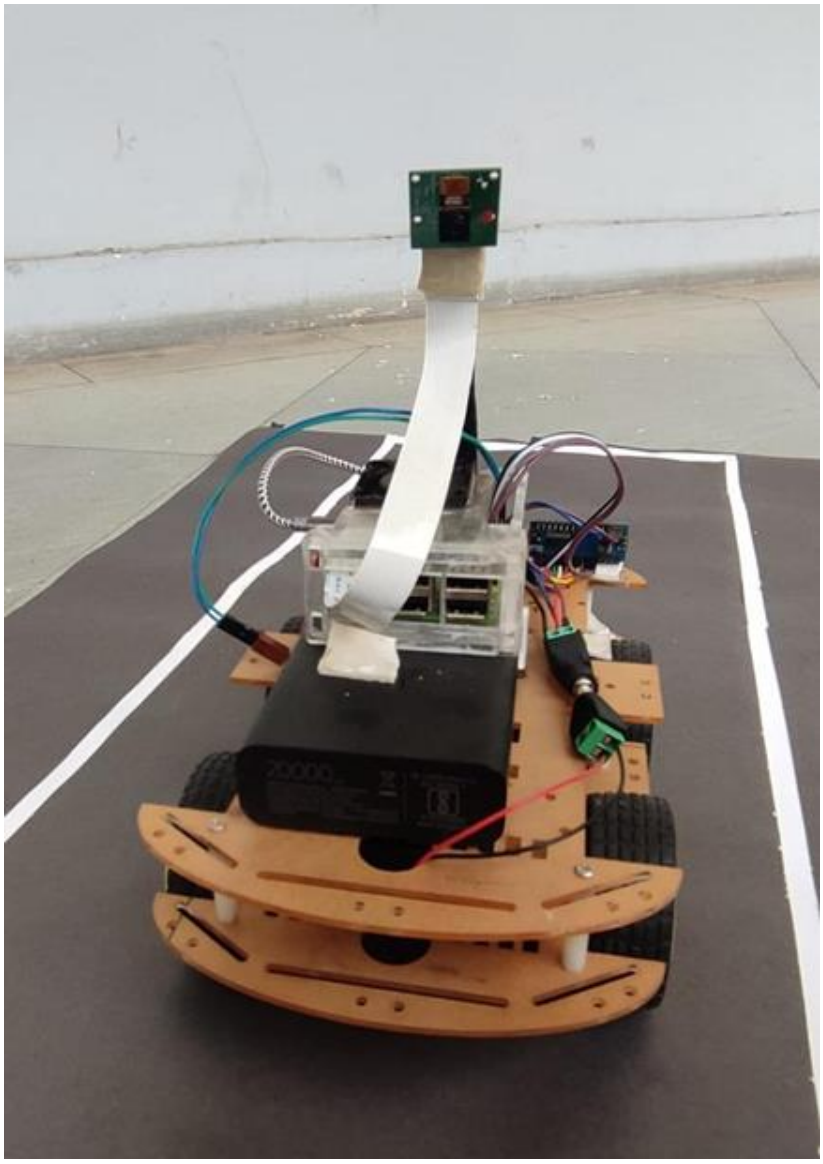

Figure 5.1.1 Front View of Prototype

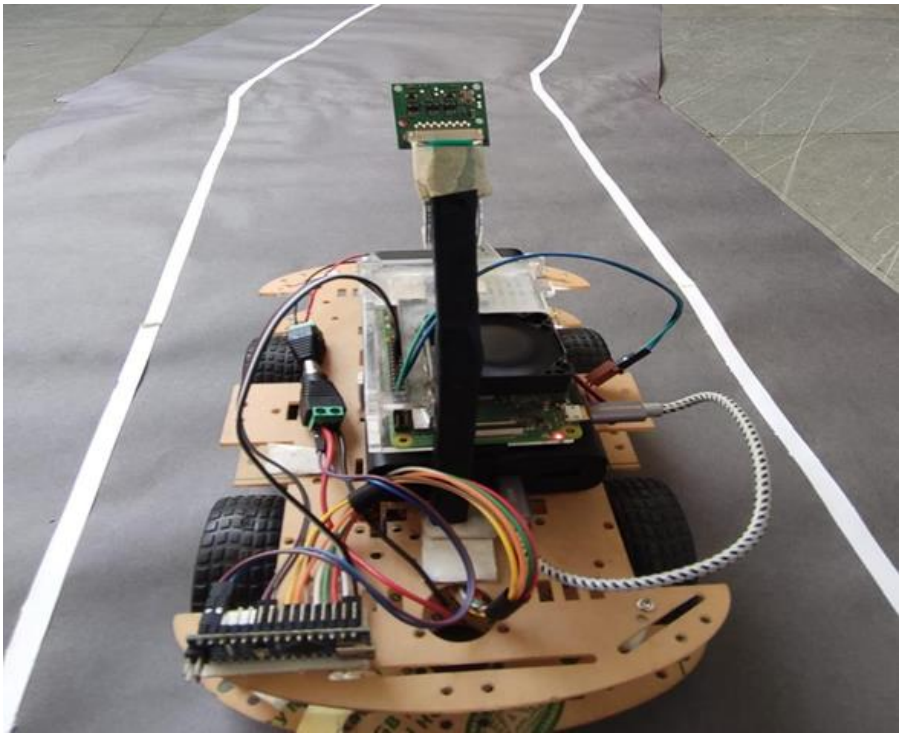

Figure 5.1.2 Back View of Prototype

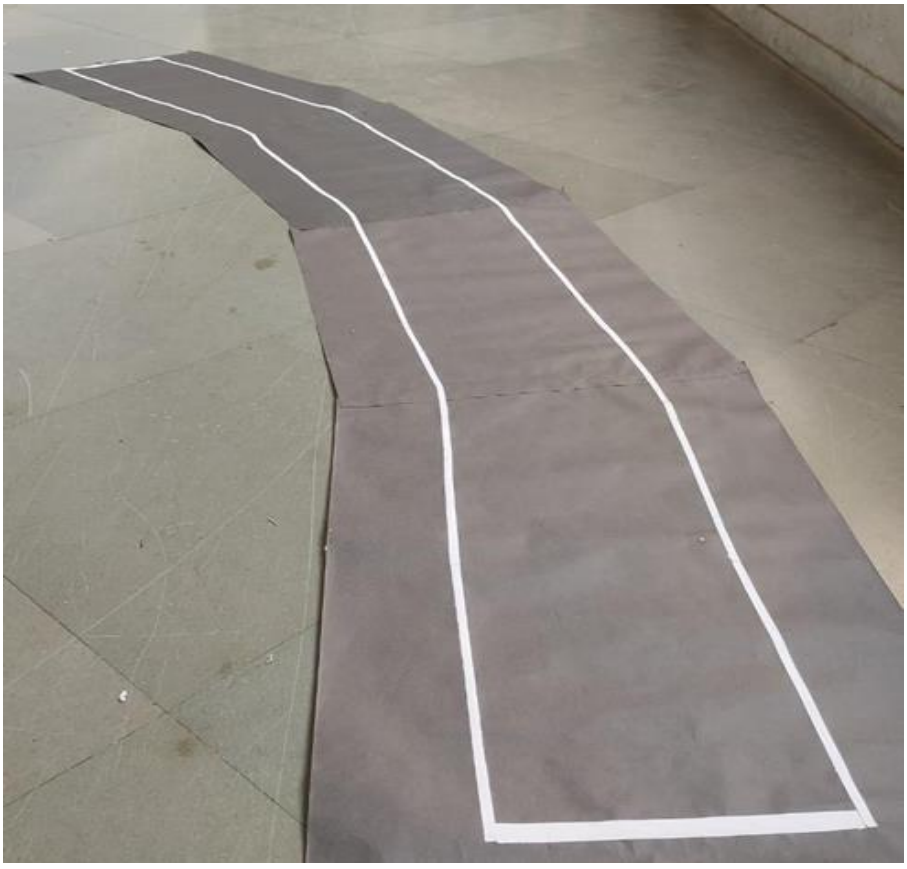

Figure 5.1.3 Design of Lane

CHAPTER 6 LIMITATIONS AND FUTURE SCOPE

6.1 Limitations

1. The approach is costly for small scale industries.
2. The robot calibration would affect due to worst case scenario of weather change.
3. We would need a high pixel range camera for better calibration.
4. At slope areas robot wouldn't be working efficiently.

6.2 Future scope

The major advantages of this system when compared to other Lane detection algorithms is that this system is more reliable and efficient to detect lane and maneuver the robot using Arduino Nano.

- Software control of the lane type (dark or light) for automatic detection possible.
- With few changes this system can also be implemented in drones.
- It can go in hazardous environment where humans cannot go.
- It can be used in hospitals and industries for specific applications.
- It can also be used in home appliances.
- By using 2 cameras we can create a 3D image of the environment.
- It can also be used in military applications example spying.
- By using different cameras, we can change the applications example night vision, thermal, etc.
- Assertive navigation system for wheel chair and blind people.

CHAPTER 7 RESULTS AND CONCLUSION

7.1 Results

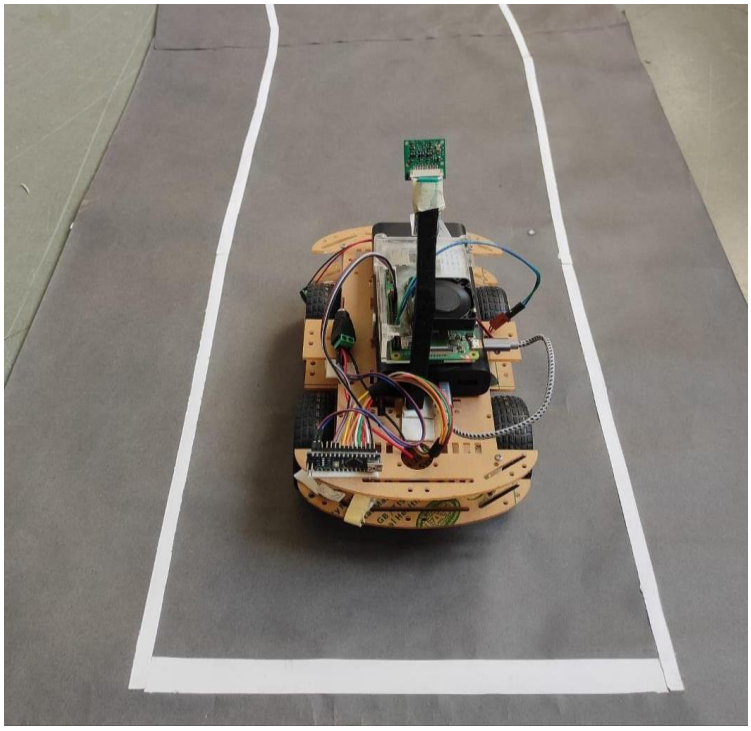

Figure 7.1.1 Robot moving Forward Direction

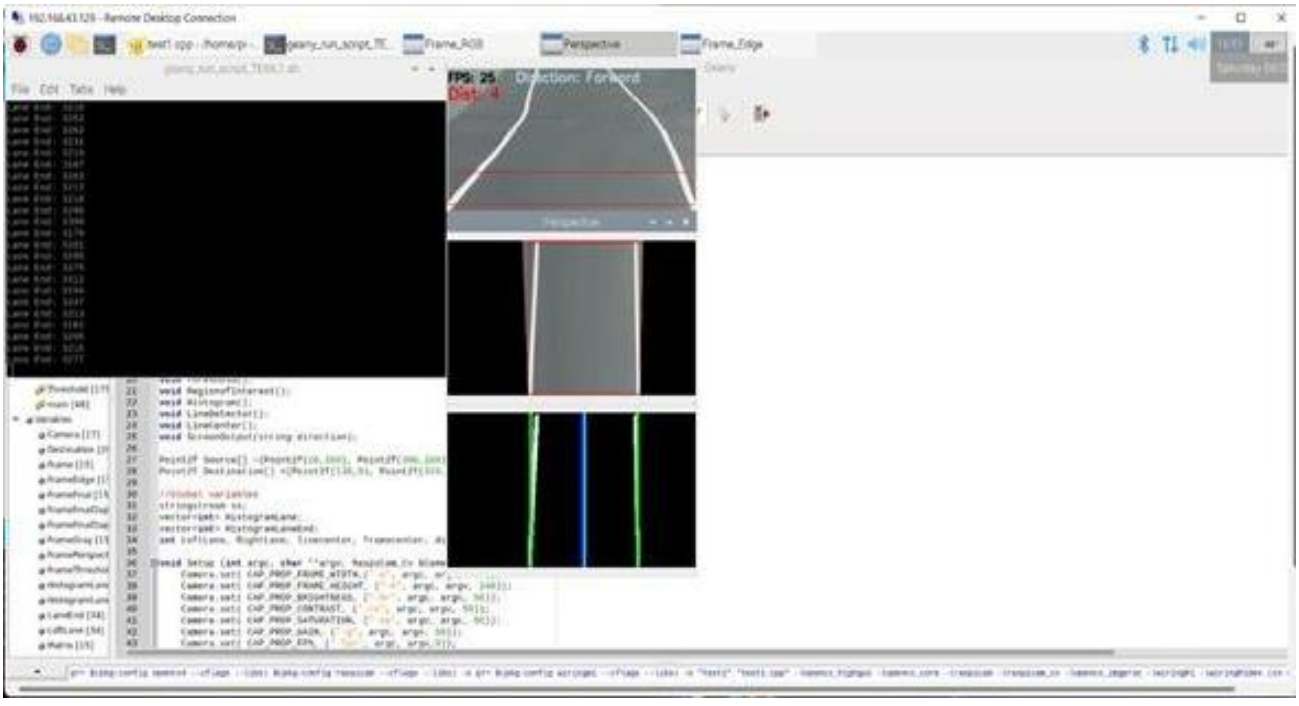

Figure 7.1.2 Result showing Forward Direction

The above figures 7.1.1 and 7.1.2 shows the robot is moving forward direction and camera is able to detect the forward lane.

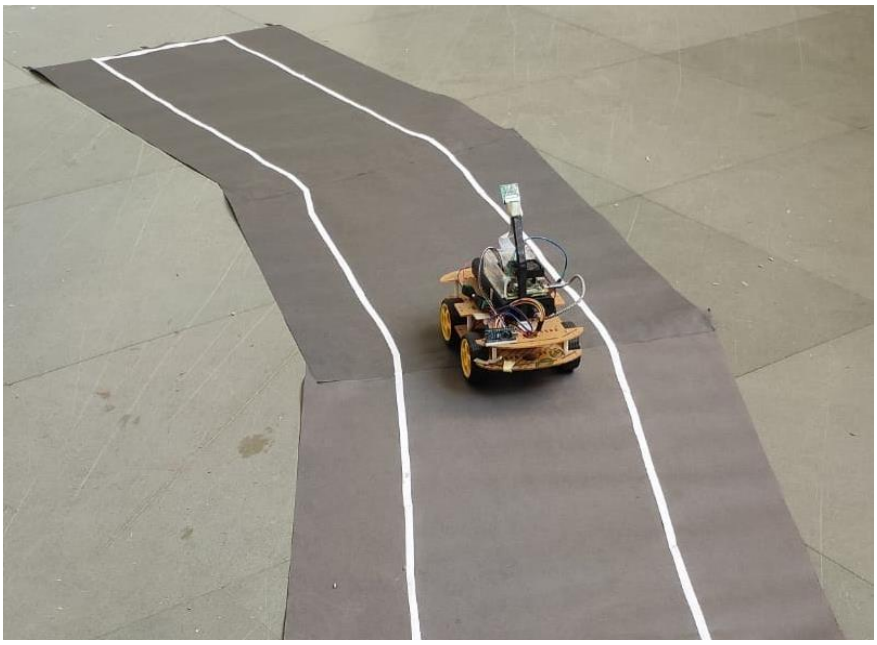

Figure 7.1.3 Robot moving Left Direction

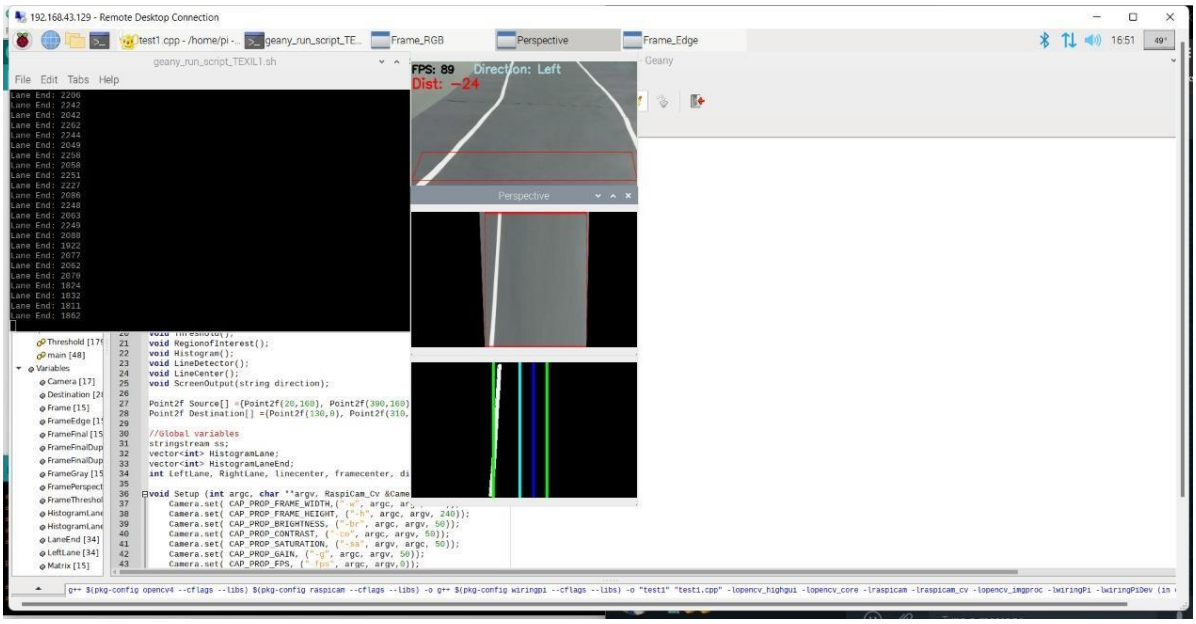

Figure 7.1.4 Result showing Left Direction

The above figures 7.1.3 and 7.1.4 shows the robot is moving left direction and camera is able to detect the left lane.

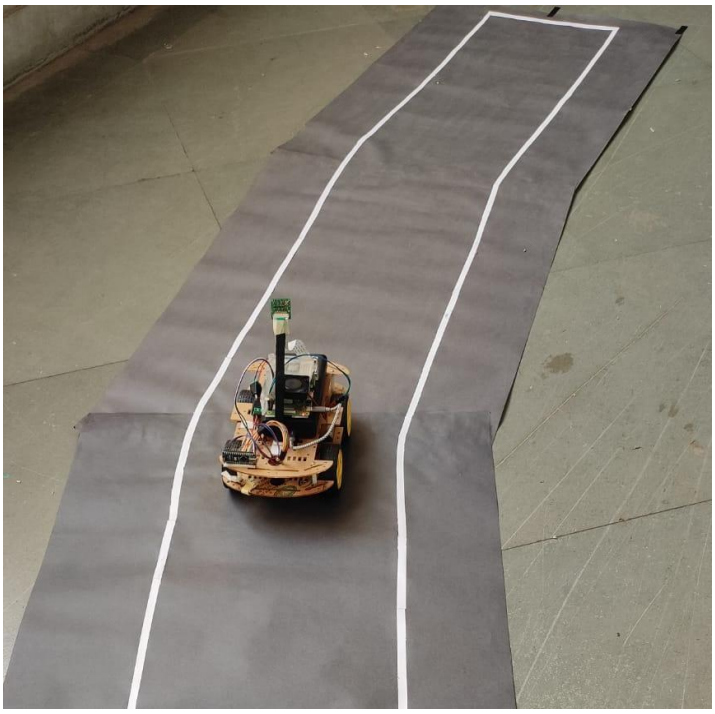

Figure 7.1.5 Robot moving Right Direction

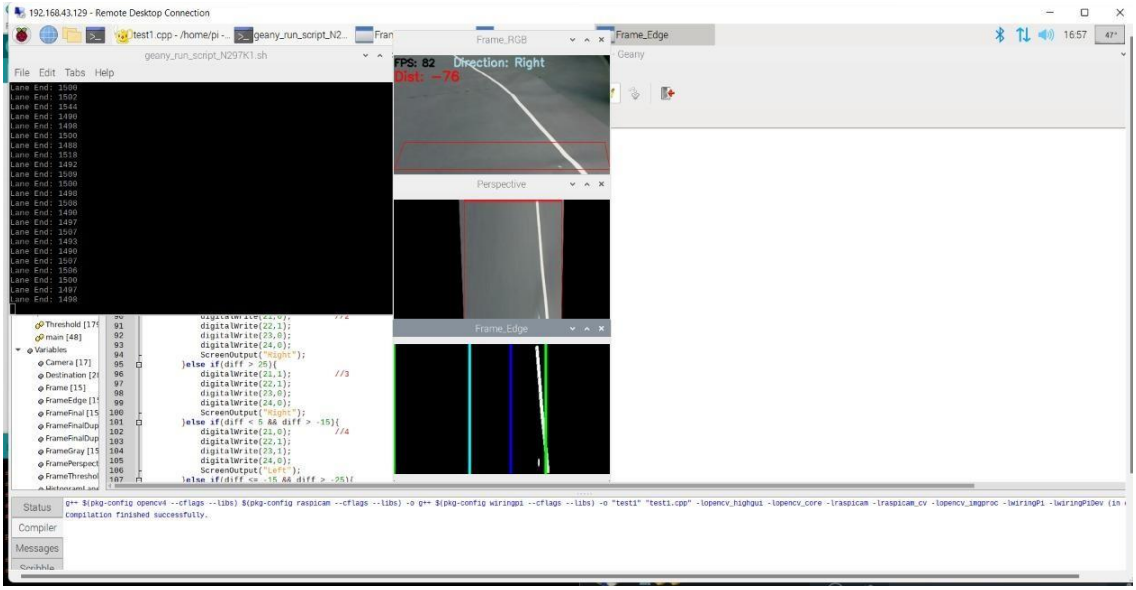

Figure 7.1.6 Result showing Right Direction

The above figures 7.1.1 and 7.1.2 shows the robot is moving right direction and camera is able to detect the right lane.

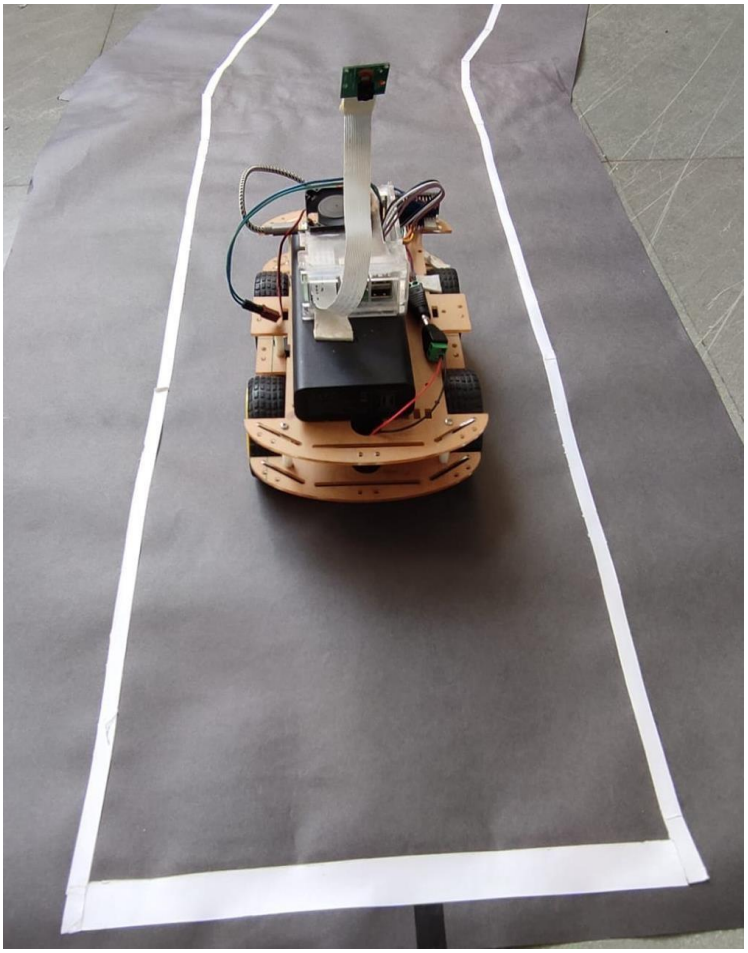

Figure 7.1.7 Robot stopped at Lane End

The above figure 7.1.7 shows that when the end of the lane is detected in the region of interest the robot stops.

7.2 Conclusion

The Lane Following Robot is automobile system that has ability to recognize its path, move and change the robot's position toward the lane in the best way to remain in track. This project report presents a camera sensor-based lane following robot design which always directs along the white line on gray surface. The robot is able to detect its path in case it is out of path. The robot also detects obstacles coming between the path and responds accordingly. Thus the "LANE FOLLOWING MOBILE ROBOT" has been designed and tested successfully. The Lane following Mobile Robot project challenged the group to cooperate, communicate, and expand understanding of electronics, mechanical systems, and their integration with programming.

References

[1] Nihal A Shetty, Mohan k and Kaushik k , Autonomous Self-Driving Car using Raspberry Pi Model, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by, www.ijert.orgRTESIT - 2019 Conference Proceedings

[2] Yu-Fang Wang and Yi-Shueh Tsai, A lane detection method based on 3D-LiDAR, Automotive Research & Testing Center (ARTC), Changhua, Taiwan (R.O.C) ISSN: 2018

[3] Han, J.-H.; Kim, H.-W,Lane Detection Algorithm Using LRF for Autonomous Navigation of Mobile Robot. Appl. Sci. 2021, 11, 6229. https://doi.org/ 10.3390/app11136229

[4] Kanjee, Ritesh; Bachoo, Asheer K.; Carroll, Johnson (2013). [IEEE 2013 6th Robotics and Mechatronics Conference (RobMech) - Durban, South Africa (2013.10.30-2013.10.31)] 2013 6th Robotics and Mechatronics Conference (RobMech) - Vision-based Adaptive Cruise Control using pattern matching., (), 93–98. doi:10.1109/robomech.2013.6685498

[5] R.Muthalagu, A.Bolimera and V.kalaichelvi ,Lane detection technique based on perspective transformation and histogram analysis for self-driving cars, Computer and electrical engineering, https://doi.org/10.1016/j.compeleceng.2020.106653

COMPONENTS DATASHEET REFERENCE

[1] Raspberry Pi 3 model B+
https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/

[2] Arduino Nano
https://www.arduino.cc/en/uploads/Main/ArduinoNanoManual23.pdf

[3] Raspberry Pi 5MP Camera Module with Cable
https://docs.rs-online.com/2888/0900766b8127db0a.pdf

[4] L298N Motor Driver Module
https://components101.com/modules/l293n-motor-driver-module

[5] Power bank
https://www.amazon.in/dp/B08HV83HL3/ref=cm_sw_r_apan_i_JQ5ZA490TYMK339W7H61

[6] 18650 Battery
https://datasheetspdf.com/mobile/1408721/TENERGY/18650/1

[7] DC Motor
https://zbotic.in/product/duel-shaft-bo-motor-150-rpm-with-wheel-black-and-yellow/

APPENDIX

Raspberry Pi Program

```
#include <opencv2/opencv.hpp>
#include <raspicam_cv.h>
#include <iostream>
#include <chrono>
#include <ctime>
#include <vector>
#include <wiringPi.h>
#include <string.h>
using namespace std;
using namespace cv;
using namespace raspicam;
// Creates the frames
Mat Frame, FramePerspective, FrameGray, Matrix, FrameThreshold, FrameEdge, FrameFinal;
Mat RnLane, FrameFinalDuplicate, FrameFinalDuplicate0, RnLaneEnd;
RaspiCam_Cv Camera;
void Capture();
void Threshold();
void RegionofInterest();
void Histogram();
void LineDetector();
void LineCenter();
void ScreenOutput(string direction);
Point2f Source[] = {Point2f(80, 160), Point2f(300, 160), Point2f(40, 210), Point2f(340, 210)};
Point2f Destination[] = {Point2f(130, 0), Point2f(310, 0), Point2f(130, 240), Point2f(310, 240)};
// Global variables
stringstream ss;
vector<int> HistogramLane;
vector<int> HistogramLaneEnd;
int LeftLane, RightLane, linecenter, framecenter, diff, LaneEnd;
void Setup(int argc, char **argv, RaspiCam_Cv &Camera) {
   Camera.set(CAP_PROP_FRAME_WIDTH, ("-w", argc, argv, 400));
   Camera.set(CAP_PROP_FRAME_HEIGHT, ("-h", argc, argv, 240));
   Camera.set(CAP_PROP_BRIGHTNESS, ("-br", argc, argv, 50));
```

```
    Camera.set(CAP_PROP_CONTRAST, ("-co", argc, argv, 50));
    Camera.set(CAP_PROP_SATURATION, ("-sa", argv, argc, 50));
    Camera.set(CAP_PROP_GAIN, ("-g", argc, argv, 50));
    Camera.set(CAP_PROP_FPS, ("-fps", argc, argv, 0));
}
int main(int argc, char **argv){
    wiringPiSetup();
    pinMode(21, OUTPUT);
    pinMode(22, OUTPUT);
    pinMode(23, OUTPUT);
    pinMode(24, OUTPUT);
    Setup(argc, argv, Camera);
    cout << "Connection to the Camera " << endl;
    if (!Camera.open()){
        cout << "Failed to connect to camera " << endl;
        return -1;
    }
    cout << "Camera ID= " << Camera.getId() << endl;
    while (1){
        auto start = std::chrono::system_clock::now();
        Capture();
        waitKey(1);
        auto end = std::chrono::system_clock::now();
        std::chrono::duration<double> elapsed_seconds = end - start;
        float t = elapsed_seconds.count();
        int FPS = 1 / t;
        if (diff > -5 && diff < 5){
            digitalWrite(21, 0); // 0
            digitalWrite(22, 0);
            digitalWrite(23, 0);
            digitalWrite(24, 0);
            ScreenOutput("Forward");
        }
        else if (diff > 5 && diff < 15){
            digitalWrite(21, 1); // 1
            digitalWrite(22, 0);
```

```
    digitalWrite(23, 0);
    digitalWrite(24, 0);
    ScreenOutput("Right");
}
else if (diff >= 15 && diff < 25){
    digitalWrite(21, 0); // 2
    digitalWrite(22, 1);
    digitalWrite(23, 0);
    digitalWrite(24, 0);
    ScreenOutput("Right");
}
else if (diff > 25){
    digitalWrite(21, 1); // 3
    digitalWrite(22, 1);
    digitalWrite(23, 0);
    digitalWrite(24, 0);
    ScreenOutput("Right");
}
else if (diff < 5 && diff > -15){
    digitalWrite(21, 0); // 4
    digitalWrite(22, 1);
    digitalWrite(23, 1);
    digitalWrite(24, 0);
    ScreenOutput("Left");
}
else if (diff <= -15 && diff > -25){
    digitalWrite(21, 1); // 5
    digitalWrite(22, 0);
    digitalWrite(23, 1);
    digitalWrite(24, 0);
    ScreenOutput("Left");
}
else if (diff < -25){
    digitalWrite(21, 0); // 6
    digitalWrite(22, 1);
    digitalWrite(23, 1);
```

```
            digitalWrite(24, 0);
            ScreenOutput("Left");
        }
        RegionofInterest();
        Threshold();
        Histogram();
        LineDetector();
        LineCenter();
        ss.str(" ");
        ss.clear();
        ss << "FPS: " << FPS;
        putText(Frame, ss.str(), Point2f(1, 20), 0, 0.6, Scalar(0, 0, 0), 2);
        ss.str(" ");
        ss.clear();
        ss << "Dist: " << diff;
        putText(Frame, ss.str(), Point2f(1, 47), 0, 0.8, Scalar(0, 0, 255), 2);
        namedWindow("Frame_RGB", WINDOW_KEEPRATIO);
        moveWindow("Frame_RGB", 640, 10);
        resizeWindow("Frame_RGB", 360, 240);
        imshow("Frame_RGB", Frame);
        namedWindow("Perspective", WINDOW_KEEPRATIO);
        moveWindow("Perspective", 640, 250);
        resizeWindow("Perspective", 360, 240);
        imshow("Perspective", FramePerspective);
        namedWindow("Frame_Edge", WINDOW_KEEPRATIO);
        moveWindow("Frame_Edge", 640, 490);
        resizeWindow("Frame_Edge", 360, 240);
        imshow("Frame_Edge", FrameFinal);
    }
    return 0;
}
void RegionofInterest(){
    line(Frame, Source[0], Source[1], Scalar(0, 0, 255), 1);
    line(Frame, Source[1], Source[3], Scalar(0, 0, 255), 1);
    line(Frame, Source[3], Source[2], Scalar(0, 0, 255), 1);
    line(Frame, Source[2], Source[0], Scalar(0, 0, 255), 1);
```

```
    Matrix = getPerspectiveTransform(Source, Destination); // Perspective transformation of Region of
interest
    warpPerspective(Frame, FramePerspective, Matrix, Size(400, 240));
}
void Capture(){
    Camera.grab();
    Camera.retrieve(Frame);
    cvtColor(Frame, Frame, COLOR_BGR2RGB);
}
void Threshold(){
    cvtColor(FramePerspective, FrameGray, COLOR_RGB2GRAY);
    inRange(FrameGray, 180, 255, FrameThreshold);
    inRange(FrameGray, 160, 255, FrameGray);
    Canny(FrameGray, FrameEdge, 150, 400, 3, false);
    add(FrameThreshold, FrameEdge, FrameFinal);
    cvtColor(FrameFinal, FrameFinal, COLOR_GRAY2RGB);
    cvtColor(FrameFinal, FrameFinalDuplicate, COLOR_RGB2GRAY);  // for histogram only
    cvtColor(FrameFinal, FrameFinalDuplicate0, COLOR_RGB2GRAY); // for histogram only
}
void Histogram(){
    HistogramLane.resize(400);
    HistogramLane.clear();
    for (size_t i{0}; i < Frame.size().width; ++i){
        RnLane = FrameFinalDuplicate(Rect(i, 140, 1, 100));
        divide(255, RnLane, RnLane);
        HistogramLane.push_back((int)(sum(RnLane)[0]));
    }
    HistogramLaneEnd.resize(400);
    HistogramLaneEnd.clear();
    for (size_t i{0}; i < Frame.size().width; ++i){
        RnLaneEnd = FrameFinalDuplicate0(Rect(i, 0, 1, 240));
        divide(255, RnLaneEnd, RnLaneEnd);
        HistogramLaneEnd.push_back((int)(sum(RnLaneEnd)[0]));
    }
    LaneEnd = sum(HistogramLaneEnd)[0];
    cout << "Lane End: " << LaneEnd << endl;
```

```
}
void LineCenter(){
   linecenter = (RightLane - LeftLane) / 2 + LeftLane;
   framecenter = 216;
   line(FrameFinal, Point2f(linecenter, 0), Point2f(linecenter, 240), Scalar(255, 255, 0), 3);
   line(FrameFinal, Point2f(framecenter, 0), Point2f(framecenter, 240), Scalar(255, 0, 0), 3);
   diff = linecenter - framecenter;
}
void LineDetector(){
   vector<int>::iterator LeftPtr;
   LeftPtr = max_element(HistogramLane.begin(), HistogramLane.begin() + 200);
   LeftLane = distance(HistogramLane.begin(), LeftPtr);
   vector<int>::iterator RightPtr;
   RightPtr = max_element(HistogramLane.begin() + 240, HistogramLane.end());
   RightLane = distance(HistogramLane.begin(), RightPtr);
   line(FrameFinal, Point2f(LeftLane, 0), Point2f(LeftLane, 240), Scalar(0, 255, 0), 3);
   line(FrameFinal, Point2f(RightLane, 0), Point2f(RightLane, 240), Scalar(0, 255, 0), 3);
}
void ScreenOutput(string direction){
   ss.str(" ");
   ss.clear();
   ss << "Diretion: " << direction;
   putText(Frame, ss.str(), Point2f(110, 20), 0, 0.7, Scalar(230, 216, 173), 2);
}
```

Arduino Program

```
const int EnL = 5;                          //Left side of Motors
const int HighL = 6;
const int LowL = 7;
const int EnR = 10;                         //Right side of Motors
const int HighR = 8;
const int LowR = 9;
const int D0 = 0; // Raspberry Pi pin 21   //Raspberry Pi connection
const int D1 = 1; // Raspberry Pi pin 22
const int D2 = 2; // Raspberry Pi pin 23
const int D3 = 3; // Raspberry Pi pin 24
```

```
int a,b,c,d,binary;
void setup() {
  pinMode(EnL, OUTPUT);
  pinMode(HighL, OUTPUT);
  pinMode(LowL, OUTPUT);
  pinMode(EnR, OUTPUT);
  pinMode(HighR, OUTPUT);
  pinMode(LowR, OUTPUT);
  pinMode(D0, INPUT_PULLUP);
  pinMode(D1, INPUT_PULLUP);
  pinMode(D2, INPUT_PULLUP);
  pinMode(D3, INPUT_PULLUP);
}
void Data(){
  a = digitalRead(D0);
  b = digitalRead(D1);
  c = digitalRead(D2);
  d = digitalRead(D3);
  binary = 8*d + 4*c + 2*b + a;
  }
void Backward(){                    //Backward
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnL, 250);
  analogWrite(EnR, 250);
}
void Forward(){                     //Forward
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnL, 250);
  analogWrite(EnR, 250);
```

```
}
void Right(){                          //Right
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnL,230);
  analogWrite(EnR, 0);
}
void Left(){                           //Left
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnL, 0);
  analogWrite(EnR,230);
}
void Stop(){
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnL, 0);
  analogWrite(EnR, 0);
}
void Right_soft(){
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnR,255);
  analogWrite(EnL,255);
  delay(1000);
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);
```

```
 digitalWrite(LowR, LOW);
 analogWrite(EnR,255);
 analogWrite(EnL,255);
 delay(200);
}
void Right_medium(){
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 digitalWrite(HighR, LOW);
 digitalWrite(LowR, HIGH);
 analogWrite(EnR,170);
 analogWrite(EnL,255);
}
void Left_soft(){
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 digitalWrite(HighR, LOW);
 digitalWrite(LowR, HIGH);
 analogWrite(EnR,255);
 analogWrite(EnL,255);
 delay(1000);
 digitalWrite(HighL, HIGH);
 digitalWrite(LowL, LOW);
 digitalWrite(HighR, LOW);
 digitalWrite(LowR, HIGH);
 analogWrite(EnR,255);
 analogWrite(EnL,255);
 delay(200);
}
void Left_medium(){
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 digitalWrite(HighR, LOW);
 digitalWrite(LowR, HIGH);
 analogWrite(EnR,255);
 analogWrite(EnL,170);
```

```
}
void Object(){
 analogWrite(EnL, 0);
 analogWrite(EnR, 0);
 delay(400);
 digitalWrite(HighL, HIGH);
 digitalWrite(LowL, LOW);
 digitalWrite(HighR, LOW);    //left
 digitalWrite(LowR, HIGH);
 analogWrite(EnL, 255);
 analogWrite(EnR,255);
 delay(1100);
 analogWrite(EnL, 0);
 analogWrite(EnR, 0);
 delay(400);
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 digitalWrite(HighR, LOW);   //Forward
 digitalWrite(LowR, HIGH);
 analogWrite(EnL, 250);
 analogWrite(EnR, 250);
 delay(1000);
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 digitalWrite(HighR, HIGH);    //Right
 digitalWrite(LowR, LOW);
 analogWrite(EnL, 255);
 analogWrite(EnR,255);
 delay(1000);
 analogWrite(EnL, 0);
 analogWrite(EnR, 0);
 delay(400);
 digitalWrite(HighL, LOW);
 digitalWrite(LowL, HIGH);
 digitalWrite(HighR, LOW);   //Forward
 digitalWrite(LowR, HIGH);
```

```
analogWrite(EnL, 250);
analogWrite(EnR, 250);
delay(1400);
analogWrite(EnL, 0);
analogWrite(EnR, 0);
delay(400);
digitalWrite(HighL, LOW);
digitalWrite(LowL, HIGH);
digitalWrite(HighR, HIGH);    //Right
digitalWrite(LowR, LOW);
analogWrite(EnL, 255);
analogWrite(EnR,255);
delay(1400);
analogWrite(EnL, 0);
analogWrite(EnR, 0);
delay(400);
digitalWrite(HighL, LOW);
digitalWrite(LowL, HIGH);
digitalWrite(HighR, LOW);   //Forward
digitalWrite(LowR, HIGH);
analogWrite(EnL, 250);
analogWrite(EnR, 250);
delay(800);
analogWrite(EnL, 0);
analogWrite(EnR, 0);
delay(400);
digitalWrite(HighL, HIGH);
digitalWrite(LowL, LOW);
digitalWrite(HighR, LOW);    //left
digitalWrite(LowR, HIGH);
analogWrite(EnL, 255);
analogWrite(EnR,255);
delay(1000);
digitalWrite(HighL, LOW);
digitalWrite(LowL, HIGH);
digitalWrite(HighR, LOW);   //Forward
```

```
  digitalWrite(LowR, HIGH);
  analogWrite(EnL, 255);
  analogWrite(EnR, 255);
  delay(200);
  }
  void StopS(){
     analogWrite(EnL, 0);
     analogWrite(EnR, 0);
     delay(4500);
     analogWrite(EnL, 250);
     analogWrite(EnR, 250);
     delay(1000);
     }
void loop() {
  Data();
  if (binary == 0){
    Forward();
    }
   else if(binary == 1){
    Right_medium();
    }
   else if(binary == 2){
     Right();
    }
   else if(binary == 3){
    Right();
    }
  else if(binary == 4){
    Left_medium();
    }
   else if(binary == 5){
    Left();
    }
  else if(binary == 6){
    Left();
    }
```

```
    else if(binary == 8){
     StopS();
    }
    else if(binary == 9){
    Object();
    }
    else if(binary > 9){
    Stop();
    }
}
```