"Autonomous Robotic Vehicle using Raspberry Pi"

Capstone project report

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology

In

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING (AI & CYBERNETICS)

Submitted to

VIT BHOPAL UNIVERSITY (M. P.)

Submitted by

SACHIN – 21BAC10036
ASHISH BARPETE - 21BAC10037
KAJAL – 21BAC10039

Under the Supervision of

DR. SOUMITRA K. NAYAK

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING
VIT BHOPAL UNIVERSITY

BHOPAL (M.P.)-466114

December – 2024

VIT BHOPAL UNIVERSITY, BHOPAL

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING

DECLARATION

I hereby declare that the Dissertation entitled "Autonomous Robotic Vehicle Using Raspberry Pi" is my own work conducted under the supervision of Dr. Soumitra K Nayak, Assistant professor, School of Electrical and Electronics Engineering (SEEE). at VIT Bhopal University, Bhopal.

I further declare that to the best of my knowledge this report does not contain any part of work that has been submitted for the award of any degree either in this university or in other university / Deemed University without proper citation.

Sachin(21BAC10036)

Ashish Barpete(21BAC10037)

Kajal(21BAC10039)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date: 12 Dec 2024

Dr. Soumitra K. Nayak
Assistant Professor, SEEE

Digital Signature of Guide

VIT BHOPAL UNIVERSITY, BHOPAL

SCHOOL OF ELECTRICAL & ELECTRONICS ENGINEERING

CERTIFICATE

This is to certify that the work embodied in this Capstone project report entitled **"AUTONOMOUS ROBOTIC VEHICLE USING RASPBERRY PI"** has been satisfactorily completed by **Mr. SACHIN** (**21BAC10036)**, **Mr. ASHISH BARPETE (21BAC10037), Ms. KAJAL (21BAC10039)** in the School of Electrical & Electronics Engineering at VIT Bhopal University, Bhopal. This work is a bonafide piece of work, carried out under my/our guidance in the School of Electrical & Electronics Engineering for the partial fulfilment of the degree of Bachelor of Technology.

PROJECT GUIDE

Dr. Soumitra K. Nayak,
Assistant Professor,
School of Electrical and Electronics
Engineering (SEEE)

Forwarded by

Dr. Soumitra K. Nayak
Program Chair,
B-Tech ECE (AI & Cybernetics)
School of Electrical and Electronics
Engineering (SEEE)

Approved by

Dr. M Suresh
Associate Professor & Dean
School of Electrical and Electronics
Engineering (SEEE)

Acknowledgement

In the first place I would like to record my gratitude to "***Dr. Soumitra K. Nayak***" for his supervision, advice, and guidance from the very early stage of this thesis work as well as giving me extraordinary experiences throughout the work. Above all and the most needed, he provided me unflinching encouragement and support in various ways. His words have always inspired me to work in an efficient and comprehensive way. I would like to thank him for his constant encouragement that enabled me to grow as a person. His presence has definitely improved me as a human being.

I express my gratitude towards **"Dr M Suresh"**, Associate Professor & Dean, School of Electrical and Electronics Engineering Department, VIT Bhopal University, for providing me all the help and permitted me to work in the laboratory with no time limits.

I shall ever remain indebted to **"Dr. Senthil Kumar Arumugam",** Vice Chancellor, VIT Bhopal University, for providing me institutional and administrative facilities during my project work at VIT.

I express my gratitude towards **"Mr. Praveen Soni"**, Lab Assistant, School of Electrical and Electronics Engineering Department, VIT Bhopal University, for providing me all the help and permitted me to work in the laboratory in the weekends as well and providing all the necessary components during this work.

Sachin (21BAC10036)

Ashish Barpete(21BAC10037)

Kajal (21BAC10039)

Abstract

This project focuses on the development of an *autonomous robotic vehicle* powered by Raspberry Pi, designed to navigate a predefined route in a controlled environment as part of its initial implementation phase. The vehicle integrates advanced vision-based algorithms, sensor systems, and control mechanisms to ensure reliable and safe navigation. The camera module acts as the primary sensor for *lane detection*, where raw image data undergoes pre-processing through *Gaussian Blur* to minimize noise and thresholding to enhance visual contrast. The processed data is analysed using the *Canny Edge Detection* algorithm, which identifies prominent edges, and the *Hough Transform*, which detects linear patterns corresponding to lane boundaries. These algorithms enable precise lane adherence even under varying environmental conditions. For obstacle detection and avoidance, *ultrasonic sensors* such as *HC-SR04* are employed to measure distances by emitting ultrasonic waves and calculating the time delay between their transmission and echo reception. This real-time data is processed by the *Raspberry Pi,* which uses decision-making algorithms to dynamically adjust the vehicle's trajectory and prevent collisions. The movement of the vehicle is controlled through an *L298N motor driver* module, which regulates the speed and direction of the DC motors via pulse-width modulation *(PWM)* signals. The motors are calibrated for stable and precise motion, with feedback mechanisms ensuring accurate wheel alignment. Power is supplied through a portable battery pack, ensuring continuous operation of all components, including the Raspberry Pi, sensors, and actuators. The implementation phase involves integrating and calibrating hardware components to achieve seamless data acquisition, processing, and actuation. Rigorous testing is conducted under controlled scenarios, with the vision system evaluated for *lane detection* accuracy and the obstacle detection subsystem validated through dynamic object placement. Observations highlight the system's capability to maintain lane adherence, respond dynamically to obstacles, and ensure smooth and stable navigation, addressing key challenges in autonomous navigation systems. This phase demonstrates the practical application of computer vision, sensor fusion, and real-time processing technologies in autonomous vehicle design, laying the foundation for reliable and scalable autonomous systems. By showcasing the successful integration and operation of essential components, this project emphasizes the feasibility of using low-cost and efficient hardware for *autonomous robotic navigation in controlled environments*, setting the stage for further advancements in automation technologies.

List of Figures

List of Symbols & Abbreviations

A	**Ampere**
ARM	**Advanced RISC Machine**
BO	**Battery operated motor**
CSI	**Camera Serial Interface**
DC	**Direct current**
GB	**Giga-Byte**
GPIO	**General Purpose Input Output**
RAM	**Random Access Memory**
ROI	**Region of interest**
V	**Volt**
W	**Watt**

Table of Contents

Chapter 1

1.1 Introduction

The growing demand for automation and precision in transportation and monitoring systems has driven significant advancements in autonomous vehicle technologies. This project focuses on designing and developing an *autonomous robotic vehicle* powered by a *Raspberry Pi* to navigate a predefined route in a controlled environment, addressing foundational aspects such as lane detection, real-time obstacle avoidance, and stable motion. The system employs a camera module as its primary sensor, capturing real-time images of the environment for vision-based navigation. These images undergo processing using advanced computer vision algorithms, where Gaussian Blur minimizes noise and enhances the clarity of critical features, *Canny Edge Detection* extracts prominent edges, and the *Hough Transform* detects lane lines to enable the vehicle to identify and adhere to lane boundaries reliably, even under variable lighting and environmental conditions. To ensure collision-free operation, the system integrates ultrasonic sensors that emit ultrasonic waves and measure the time taken for their echoes to return, enabling accurate distance measurement from obstacles. The Raspberry Pi processes this sensor data in real time, executing decision-making algorithms to dynamically adjust the vehicle's trajectory, ensuring safe navigation while maintaining its course. The vehicle's propulsion is driven by DC motors controlled through an L298N motor driver module, which regulates speed and direction using pulse-width modulation (PWM) signals, offering precise control over acceleration and turning. A portable battery powers the entire system, providing uninterrupted energy to the Raspberry Pi, sensors, and actuators. The implementation process involves integrating these components into a cohesive system, carefully calibrating hardware and software to ensure seamless interaction. Rigorous testing in controlled conditions validates the system's performance, demonstrating its ability to accurately detect and follow lanes, respond dynamically to obstacles, and maintain stable motion. By combining computer vision, sensor fusion, and real-time processing, this project underscores the potential of autonomous robotic systems in addressing critical challenges in automation. It highlights the viability of such technologies for real-world applications in fields like logistics, surveillance, and smart transportation, offering significant advancements in safety, efficiency, and operational precision. This project highlights the transformative potential of autonomous robotic vehicles, leveraging advancements in AI, sensor technology, and embedded systems to address practical challenges in logistics, surveillance, and smart cities.

1.2 Motivation of the study

The motivation for this project arises from the increasing demand for innovative and efficient solutions to address challenges in transportation, surveillance, and automation. In modern times, industries seek systems capable of minimizing human intervention while enhancing safety, precision, and reliability in complex environments. *Autonomous robotic vehicles* have emerged as a revolutionary technology, offering the potential to perform repetitive, hazardous, and labor-intensive tasks with exceptional accuracy. By automating such tasks, these vehicles not only improve efficiency but also significantly reduce human exposure to risks in unsafe or extreme conditions.

A real-world example of this can be seen in autonomous delivery robots, which are increasingly used in logistics to transport goods within warehouses and even for last-mile delivery in urban areas. These robots navigate autonomously, reducing dependency on human labor and ensuring faster, cost-effective operations. Similarly, self-driving cars are transforming the transportation sector, promising reduced road accidents by eliminating human errors and optimizing traffic flow through intelligent systems. Another practical application lies in autonomous agricultural robots, which can plow fields, plant seeds, and monitor crop health, addressing labor shortages and enhancing productivity in farming.

This project also aims to harness affordable and versatile technologies such as the Raspberry Pi, combined with advanced *vision-based algorithms* and sensor integration, to build an efficient and scalable autonomous navigation system. The study focuses on addressing foundational challenges like lane detection, obstacle avoidance, and stable motion in controlled environments, while paving the way for future applications in fields like smart transportation, logistics, and surveillance. By developing a robust framework, this project contributes to the broader goal of reducing human risk, optimizing resource utilization, and fostering innovation in the rapidly growing domain of autonomous systems.

1.3 Objective of the work

To enable the autonomous robotic vehicle to navigate a controlled environment safely, ensuring precise obstacle detection and collision avoidance for enhanced safety. To achieve reliable lane detection and adherence using advanced computer vision techniques, ensuring the vehicle follows a predefined path with minimal deviations.

Exploring Challenging Terrains: To demonstrate the vehicle's ability to navigate complex controlled environments, overcoming obstacles and maintaining stable motion in diverse scenarios.To optimize the integration and functionality of hardware components like Raspberry Pi, camera modules, ultrasonic sensors, and motor driver modules for real-time processing and efficient navigation.

Real-Time Decision Making: To incorporate advanced algorithms for real-time data processing and decision-making, enabling the vehicle to adapt dynamically to changes in the environment.

Chapter 2

2.1 Literature Review

[1] **Gurjashan Singh Pannu et al. [2015]: Design and Implementation of Autonomous Car using Raspberry Pi** This research focuses on developing an autonomous car prototype using Raspberry Pi, integrating a Pi camera and ultrasonic sensors for data collection and processing, and combining feature-based and model-based lane detection methods supported by OpenCV to enable autonomous navigation on marked and unmarked roads. Key techniques include lane detection through the Hough Transform and feature-based approaches to identify road edges, ultrasonic sensors for obstacle detection and safe path determination, and the use of HSV thresholds to extract road features under varying lighting conditions. A Raspberry Pi Model B serves as the processing unit, with a servo motor and motor drivers for control. The study provides foundational insights into monocular vision for autonomous navigation, addressing challenges such as noise and inconsistent road markings, which are crucial for developing reliable autonomous systems. However, gaps include limited scalability for larger-scale or higher-speed applications, a lack of integration with machine learning for adaptive performance improvements, and challenges in processing efficiency when handling dynamic road conditions.

This study designs a lane departure warning system using image processing to detect and alert drivers about lane deviations, leveraging Canny Edge Detection and Hough Transform to identify road edges and calculate lane departure metrics. Key techniques include lane detection through ROI selection to reduce processing load and the Hough Transform for robust identification of straight and curved roads, and edge detection using Canny Edge Detection enhanced by background subtraction techniques for feature extraction. The hardware comprises a Raspberry Pi 2 as the processing hub and a USB camera for video capture, employing simplified algorithms for real-time processing with low computational overhead. The study highlights cost-effective methods for implementing lane detection on embedded systems, emphasizing computational efficiency for resource-constrained applications. However, gaps include challenges in detecting lane edges under complex environmental conditions like heavy traffic or inconsistent lighting, limited adaptability of simple algorithms to diverse road scenarios, and a lack of integration with additional features such as obstacle avoidance or traffic sign recognition

Chapter 3

3.1 Problem formulation

The problem formulation for this project arises from the growing demand for autonomous systems capable of executing predefined tasks with high precision, safety, and reliability. Traditional human-operated systems encounter limitations in performing repetitive, hazardous, or time-sensitive tasks, especially in controlled environments such as warehouses, research facilities, and industrial zones. Current autonomous solutions often lack scalability and cost-effectiveness, which restricts their deployment for specialized applications. Furthermore, designing systems that can respond in real time to environmental changes, such as detecting obstacles or identifying pathways accurately, presents a significant challenge. To address these issues, this project aims to develop an autonomous robotic vehicle utilizing a Raspberry Pi as the central processing unit to integrate sensor data and execute control algorithms. The vehicle incorporates a camera module to capture real-time images, processed through advanced computer vision techniques such as Gaussian Blur, Canny Edge Detection, and Hough Transform to detect lanes and guide navigation. Ultrasonic sensors are used for obstacle detection, enabling dynamic path adjustments to avoid collisions. Motion control is achieved through DC motors operated via an L298N motor driver module, ensuring precise control over speed and direction using pulse-width modulation (PWM). A portable battery provides uninterrupted power supply, enhancing the system's reliability during operation. The problem also emphasizes the importance of a modular design that can be extended to handle more complex tasks in future phases, including dynamic path planning and autonomous exploration. By addressing foundational challenges in autonomous navigation, this project seeks to create a scalable, efficient, and cost-effective solution for industries requiring automation in areas such as logistics, surveillance, and environmental monitoring, bridging the gap between advanced technology and practical applications.

3.2 Proposed methodology

Proposed Methodology for Autonomous Robotic Vehicle using Raspberry Pi
The development of the autonomous robotic vehicle begins with configuring the Raspberry Pi as the central processing unit. The first step is to set up the Raspberry Pi environment by installing the necessary libraries and software tools, such as OpenCV for computer vision and GPIO control libraries for hardware interfacing. Next, the camera module is attached and

configured to capture real-time images of the environment, ensuring proper resolution and frame rates for effective processing. The captured images are processed using advanced computer vision techniques, including Gaussian Blur to reduce noise, Canny Edge Detection for edge detection, and Hough Transform to identify lane lines. These processed outputs are used to guide the vehicle along the predefined path. The next stage involves connecting ultrasonic sensors to the Raspberry Pi to enable obstacle detection and collision avoidance. These sensors continuously emit ultrasonic waves and analyse their echoes to measure the distance between the vehicle and surrounding objects. The data from the ultrasonic sensors is fed to the Raspberry Pi, which executes decision-making algorithms to adjust the vehicle's trajectory dynamically and avoid obstacles while adhering to its designated route. For motion control, the Raspberry Pi is connected to an L298N motor driver module, which regulates the speed and direction of the vehicle using pulse-width modulation (PWM). Four BO motors are connected to the L298N module to facilitate smooth and precise movement. A portable DC power supply, typically ranging from 12-20V, powers the entire system, including the Raspberry Pi, sensors, and motors. The hardware setup is calibrated to ensure stable and reliable operation under varying conditions. Once the hardware is configured, a Python-based control program is deployed on the Raspberry Pi to integrate the sensor data and implement the navigation algorithms. The robot is then tested in a controlled environment to validate its

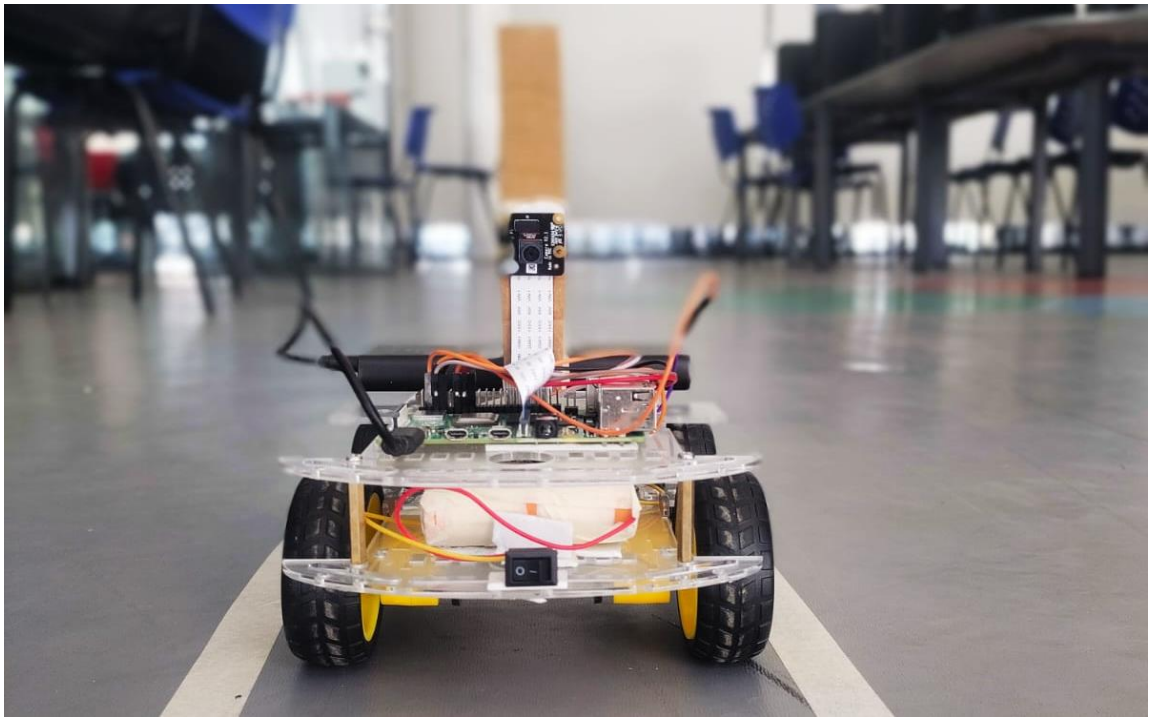

Fig 3.1 Autonomous Lane detection car model

performance. During these tests, observations are recorded to refine the system's lane detection accuracy, obstacle avoidance capabilities, and motion stability. The final setup ensures that the vehicle can autonomously navigate predefined paths, respond dynamically to environmental changes, and maintain stable operation, laying the groundwork for future extensions like dynamic path planning and autonomous exploration.

3.2.1 Components

I. Raspberry - Pi:

The Raspberry Pi 4 Model B is a powerful, versatile single-board computer developed by the Raspberry Pi Foundation, designed to cater to a wide range of applications, from DIY projects to advanced systems. It is powered by the Broadcom BCM2711 SoC, featuring a 1.5 GHz quad-core ARM Cortex-A72 processor that offers substantial improvements in performance compared to earlier models. The Raspberry Pi 4 is available in three RAM variants: 2GB, 4GB, and 8GB of LPDDR4-3200 SDRAM, providing options suitable for both lightweight and resource-intensive tasks. It includes storage via a microSD card slot, which supports UHS-1 cards for faster read and write speeds. The device has dual micro HDMI ports, capable of driving two 4K displays simultaneously, and offers USB 3.0, USB 2.0, Gigabit Ethernet, and Bluetooth 5.0 for seamless connectivity. Additionally, it has a dedicated camera interface (CSI-2) and display interface (DSI) for integration with cameras and touchscreen displays, while the 40 GPIO pins enable users to connect and control sensors, motors, and other peripherals. The Raspberry Pi 4 Model B also supports Power-over-Ethernet (PoE) via an optional hat and features high-definition audio output through the 3.5mm jack or HDMI. This comprehensive feature set makes it a strong contender for applications in robotics, IoT, digital signage, and home automation.

Fig 3.2 Raspberry Pi Model 4B

The Raspberry Pi 4 Model B is a robust computing platform that combines versatility and power for various use cases, especially for projects requiring advanced processing capabilities. Its 1.5 GHz quad-core ARM Cortex-A72 CPU, supported by 2GB, 4GB, or 8GB of LPDDR4 RAM, ensures excellent performance in tasks such as multitasking, media streaming, gaming, and more. The device supports dual 4K video output via its micro-HDMI ports, allowing for high-definition visual output suitable for digital signage or multimedia systems. Its USB 3.0 and USB 2.0 ports, along with Gigabit Ethernet, Wi-Fi 802.11ac, and Bluetooth 5.0, enable fast data transfer and reliable connectivity, making it ideal for networking and IoT applications. The Pi 4 also features the ability to handle high-quality audio output through both HDMI and the 3.5mm jack, and with the 40 GPIO pins, it supports integration with a wide range of sensors, actuators, and other hardware components. For image and video processing, the board supports the CSI-2 camera interface, allowing users to add cameras for projects like machine vision and surveillance. The power consumption of the Raspberry Pi 4 ranges from 2.7W at idle to 7.6W under full load, ensuring it remains energy-efficient while offering superior processing power. This extensive functionality, along with its small form factor and affordable price, makes the Raspberry Pi 4 Model B an ideal choice for everything from educational projects to professional applications.

II. Thonny IDE:

Thonny is an integrated development environment (IDE) designed specifically for beginners and educational purposes, primarily focused on Python programming. It was created to provide a simple, user-friendly interface that helps new programmers learn coding concepts without overwhelming them with complex features. Thonny comes with a built-in Python interpreter, eliminating the need for separate installations or configurations, making it particularly suited for beginners. The IDE offers a clean and intuitive interface, where users can write, run, and debug Python code in a straightforward manner. Key features include an interactive shell, easy-to-use debugger, variable viewer, and a simple code editor with syntax highlighting. Thonny also supports virtual environments, allowing users to manage different Python projects and dependencies independently. Its lightweight nature and minimalistic design make it an excellent choice for teaching and learning programming, especially for those new to coding and Python.

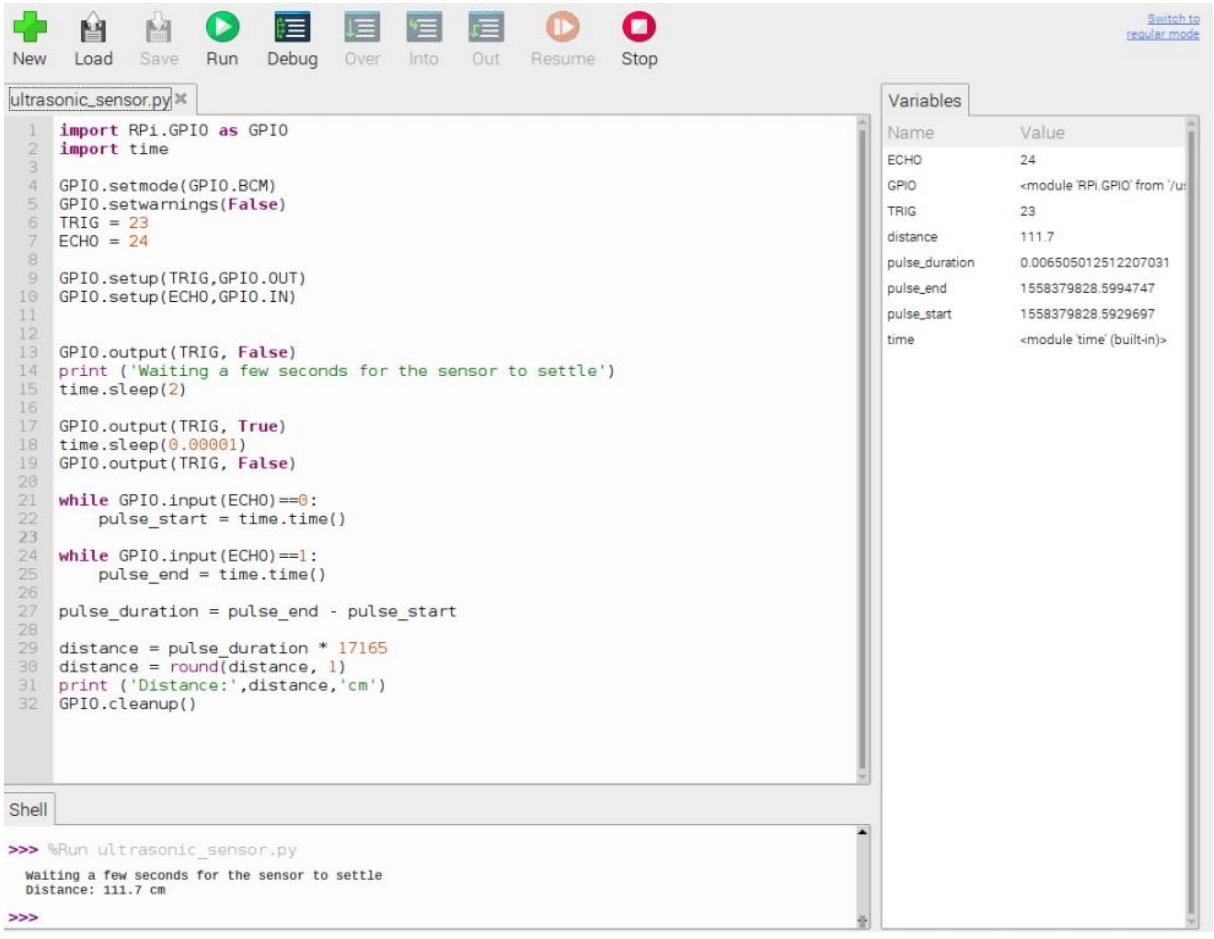

Fig 3.3 Thonny IDE (Python)

Thonny IDE has become a go-to platform for beginners and educational institutions due to its simplicity and ease of use. The IDE provides an integrated Python environment with a built-in interpreter, which simplifies the setup process for new users, ensuring that they can start coding without the hassle of configuration. Its user-friendly interface includes an interactive shell and a basic code editor with essential features like syntax highlighting, auto-completion, and error checking. One of the standout features is its debugger, which allows beginners to step through their code line by line, helping them understand how their programs work. Thonny also includes a variable view that shows the current state of all variables, making it easier to track data flow in real-time. For those working with different projects, Thonny supports the use of virtual environments, enabling users to maintain project-specific dependencies. Its lightweight nature, combined with Python support, makes it an ideal tool for students, educators, and anyone looking to start their journey into Python programming without unnecessary distractions or complexities.

III. Raspberry Pi Camera module 2:

The Raspberry Pi Camera Module 2 is an official camera accessory designed for the Raspberry Pi series of single-board computers. It features an 8-megapixel Sony IMX219 sensor, which is capable of capturing high-quality images and video. The module connects to the Raspberry Pi via the dedicated Camera Serial Interface (CSI) port, ensuring high-speed data transfer between the camera and the Raspberry Pi. The Camera Module 2 is compatible with all Raspberry Pi models that have a CSI port, providing a straightforward solution for image and video capture projects. Its compact size, ease of use, and high performance make it ideal for various applications, such as surveillance systems, robotics, photography, and DIY projects. With support for both still images and HD video (1080p at 30fps), the Raspberry Pi Camera Module 2 is a versatile tool for capturing visual data in a wide range of environments.

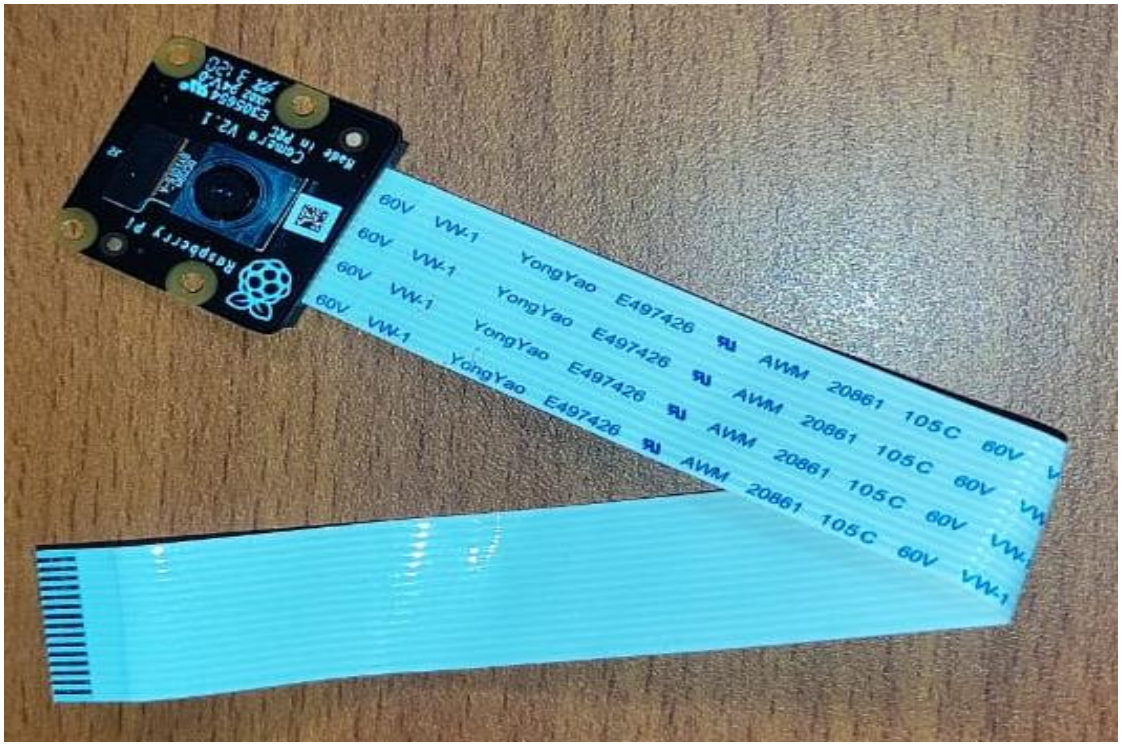

Fig 3.4 Pi Camera module 2

The Raspberry Pi Camera Module 2 is a versatile and powerful accessory that enhances the capabilities of Raspberry Pi projects by enabling high-quality image and video capture. Equipped with an 8-megapixel Sony IMX219 sensor, the camera offers sharp and clear images, as well as the ability to record full HD video at 1080p resolution at 30 frames per second. This makes it an excellent choice for a wide range of applications, from surveillance and security systems to computer vision tasks and robotics. The module is easy to integrate with Raspberry Pi, connecting directly via the Camera Serial Interface (CSI) port for fast data transfer. It supports both still image capture and video streaming, providing flexibility for different use cases. The camera also includes features such as auto-exposure and automatic white balance, which help ensure that the captured visuals are of high quality under varying lighting conditions. The compact design and powerful performance make the Raspberry Pi Camera Module 2 a valuable addition to any Raspberry Pi-based project, especially in scenarios where visual data is critical.

IV. L298N Motor Driver:

The Raspberry Pi Camera Module 2 is a versatile and powerful accessory that enhances the capabilities of Raspberry Pi projects by enabling high-quality image and video capture.

Equipped with an 8-megapixel Sony IMX219 sensor, the camera offers sharp and clear images, as well as the ability to record full HD video at 1080p resolution at 30 frames per second. This makes it an excellent choice for a wide range of applications, from surveillance and security systems to computer vision tasks and robotics. The module is easy to integrate with Raspberry Pi, connecting directly via the Camera Serial Interface (CSI) port for fast data transfer. It supports both still image capture and video streaming, providing flexibility for different use cases. The camera also includes features such as auto-exposure and automatic white balance, which help ensure that the captured visuals are of high quality under varying lighting conditions. The compact design and powerful performance make the Raspberry Pi Camera Module 2 a valuable addition to any Raspberry Pi-based project, especially in scenarios where visual data is critical.

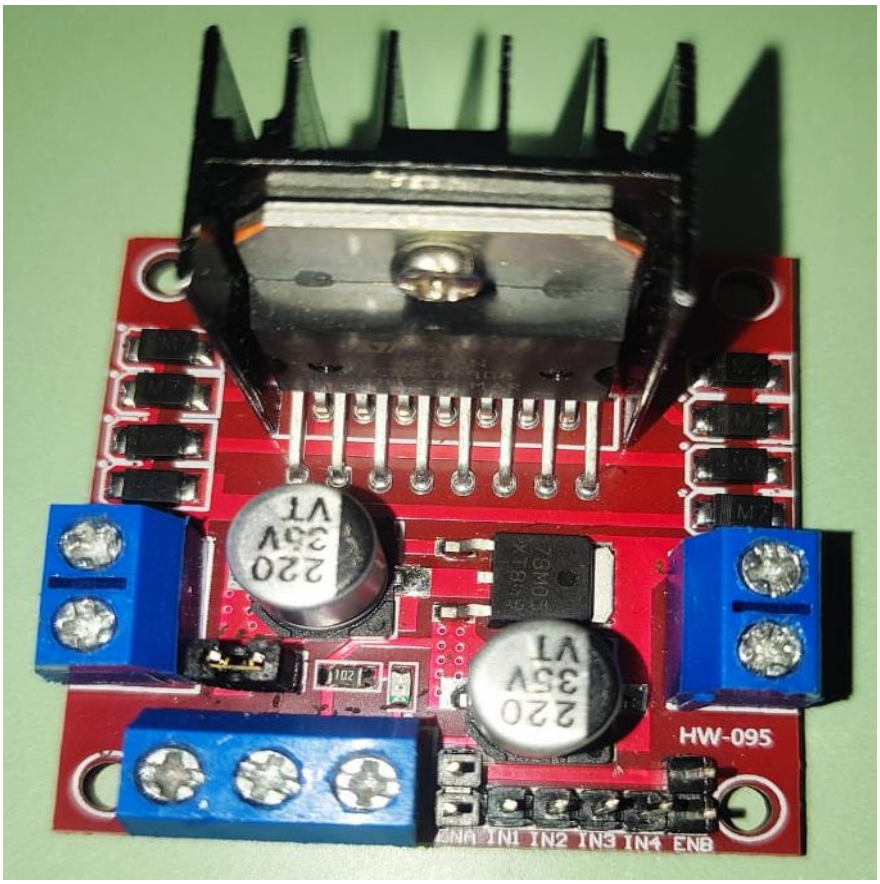

Fig 3.5 L298N Motor Driver

The L298N Motor Driver is a popular and widely used dual H-Bridge motor driver IC, designed for driving motors in various robotics and automation applications. It is capable of controlling the direction and speed of DC motors, stepper motors, and other inductive loads. The L298N

is equipped with two H-Bridge circuits, allowing it to control two DC motors simultaneously, or one stepper motor, by providing bi-directional control and enabling variable speed operation. It operates with a voltage range of 4.5V to 46V and can supply a continuous current of up to 2A per motor, with peak currents reaching up to 3A. The module is designed to interface easily with microcontrollers such as Arduino, Raspberry Pi, and other development platforms. The L298N also includes built-in thermal shutdown and overload protection, making it a reliable and robust solution for motor control applications.

V.Ultrasonic sensor:

The HC-SR04 ultrasonic sensor is a popular and cost-effective sensor used for measuring distance and detecting obstacles in various robotic and automation projects. It operates based on ultrasonic waves, emitting a high-frequency sound pulse and measuring the time it takes for the sound to bounce back after hitting an object. The sensor consists of two main components: the transmitter (which emits the sound pulse) and the receiver (which detects the reflected sound). The HC-SR04 operates with a voltage range of 5V and is capable of measuring distances ranging from 2 cm to 4 meters with an accuracy of about 3mm. It is commonly used in robotics for obstacle avoidance, level sensing, and proximity detection. The sensor communicates with microcontrollers like Arduino and Raspberry Pi via two pins: the trigger pin (to emit the ultrasonic pulse) and the echo pin (to receive the reflected pulse).

Fig 3.6 HC-SR04 Ultrasonic sensor

The HC-SR04 ultrasonic sensor is widely used in robotics and automation for measuring distances and detecting obstacles by emitting and receiving ultrasonic sound waves. It has a

voltage requirement of 5V and provides accurate distance measurements ranging from 2 cm to 4 meters, with an accuracy of around 3mm. The sensor consists of two main components: the ultrasonic transmitter, which emits a pulse, and the receiver, which detects the reflected pulse. The HC-SR04 operates by sending a pulse through the trigger pin and receiving the reflected pulse through the echo pin, allowing the microcontroller (such as Arduino or Raspberry Pi) to calculate the distance based on the time difference between the emission and reception of the pulse. This sensor is ideal for applications in obstacle avoidance, object detection, and range-finding in various projects, especially in robotics. Its ease of use, affordability, and reliable performance make it a go-to solution for many robotic systems requiring distance measurement and proximity detection.

VI. Lithium Ion Battery:

The 3.7V lithium-ion (Li-ion) battery is a rechargeable power source commonly used in portable electronics and robotics. It is known for its high energy density, lightweight design, and long lifespan. These batteries typically offer a stable 3.7V output and are available in various capacities, making them ideal for mobile systems and low-power applications. They are often charged using circuits like the TP4056 module, ensuring efficient and safe charging.

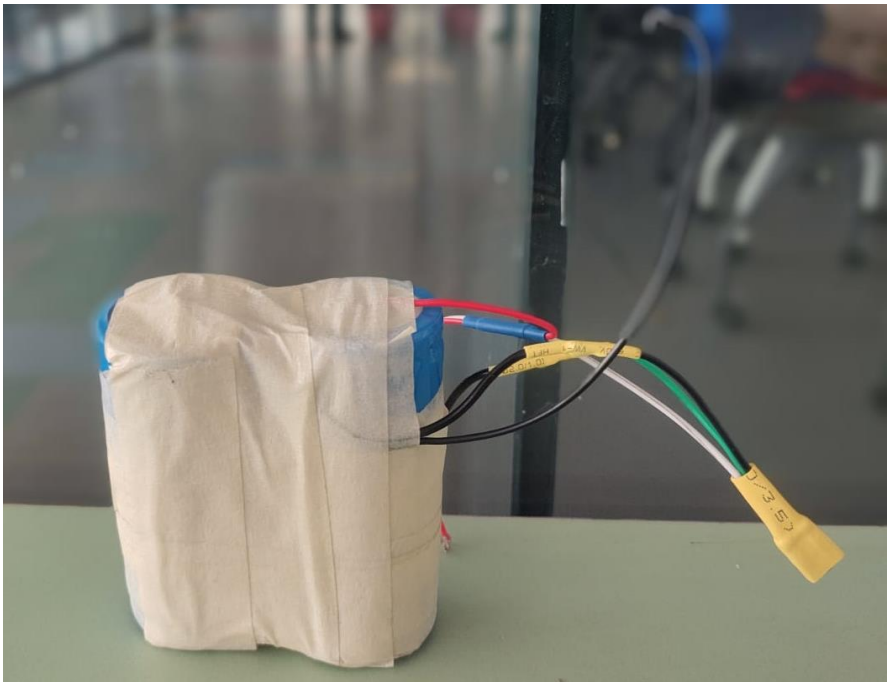

Fig 3.7 3.7-V Lithium Ion Battery

The 3.7V lithium-ion (Li-ion) battery is a lightweight and rechargeable power source, ideal for robotics and portable electronics. It offers a stable voltage output and comes in various capacities, providing flexibility for different power requirements. These batteries are charged using modules like TP4056 and are known for their long lifespan and energy efficiency, making them perfect for mobile applications. Proper battery management is essential for safety and longevity.

VII. Servo motor:

The SG90 is a small and lightweight servo motor commonly used in hobbyist robotics and automation projects. Operating on a voltage range of 4.8V to 6V, it provides precise control over angular movements, with a typical range of 0° to 180°. The motor includes a built-in control circuit and feedback mechanism, allowing it to rotate to specific positions as per the input PWM signal. The SG90 is ideal for applications that require compact size and high precision, such as controlling robotic arms, steering mechanisms, or camera gimbals in small-scale projects.

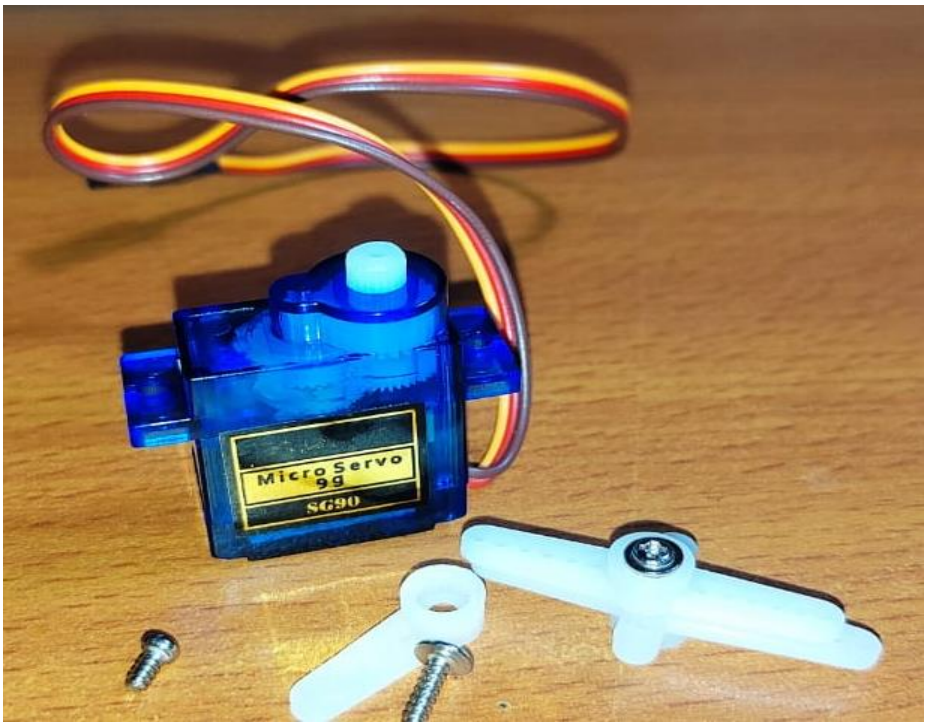

Fig 3.8 SG 90 Servo motor

The SG90 is a miniature servo motor often used in hobby robotics and various automation tasks. It operates within a voltage range of 4.8V to 6V and offers precise angular control with

a typical range of 0° to 180°. The motor is equipped with a built-in feedback control circuit, enabling it to respond accurately to PWM signals for specific rotational movements. Known for its compact size, efficiency, and reliability, the SG90 is suitable for applications like robotic arms, small vehicle steering, or camera controls where precise movement and size constraints are crucial.

VIII. Car Chassis

The 4-wheel car chassis is a versatile and robust platform used in robotics and autonomous vehicle projects. Typically made of durable materials like plastic or metal, it features four wheels for stability and mobility. The chassis design accommodates various components such as motors, motor drivers, and sensors, providing a solid foundation for building robots or vehicles with precise control over movement.

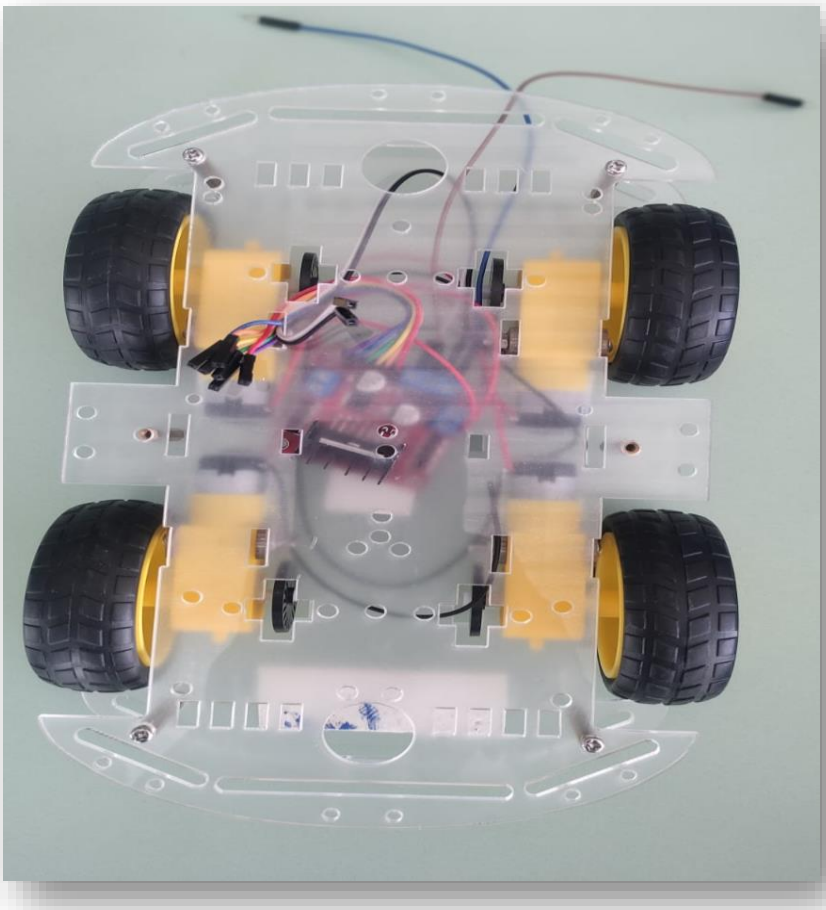

Fig 3.9 Four-wheel Car Chassis

The 4-wheel car chassis is a reliable base for building autonomous or robotic vehicles. Constructed from sturdy materials like plastic or metal, it includes four wheels to ensure stability and efficient movement. It is designed to easily mount motors, sensors, and other components, making it ideal for creating customizable robotic systems with excellent manoeuvrability and control.

3.3 Implementation

3.3.1 Lane detection car model

The implementation of the Lane Detection Car Model began with integrating various hardware components and developing the core algorithm to process images in real-time for lane detection. The Raspberry Pi 4-B served as the central controller, running the necessary algorithms for image processing and motor control. The Pi camera was attached to the Raspberry Pi to capture live video footage of the road, which is then fed into the image processing pipeline. The captured frames are processed using several computer vision techniques. Initially, the frames undergo Gaussian blur to reduce noise and smooth the image, which helps in better edge detection. After that, thresholding is applied to create a binary image, where lane markings are emphasized by setting all pixels above a certain intensity level to white and the rest to black. Following this, the Hough Transform algorithm was used to detect straight lines in the image, which represent the lane boundaries. To enhance efficiency, a Region of Interest (ROI) was defined to focus on the bottom half of the image, where the lanes are typically located, reducing unnecessary computation for irrelevant parts of the frame.

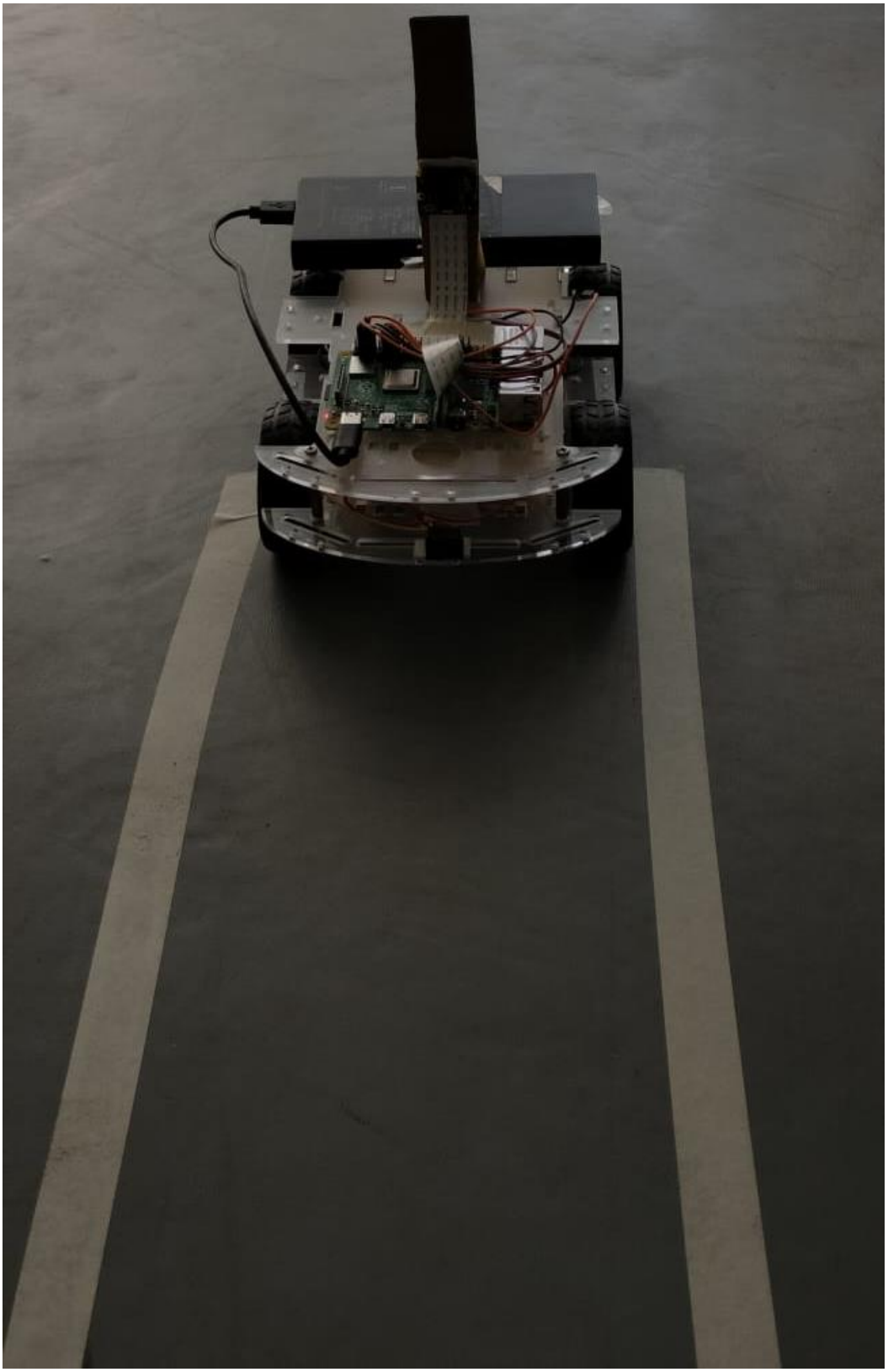

Fig 3.10 Lane detection car model

On the hardware side, the L298N motor driver was used to control the movement of the car. The motor driver was connected to the Raspberry Pi via GPIO pins to manage the four DC motors of the car. Using Pulse Width Modulation (PWM**)**, the Raspberry Pi could control the speed of the motors, while GPIO pins managed the direction of the motors (forward, backward, left, and right). This allowed the car to follow the detected lane by adjusting its position accordingly. The car's power system was supported by a 3.7V lithium-ion battery, which powered both the Raspberry Pi and the motors, ensuring the vehicle could operate autonomously for extended periods. In testing, various adjustments were made to fine-tune the motor speeds and lane detection parameters, optimizing the car's ability to stay within the lane and make turns when necessary. The system was tested in different scenarios, such as straight paths and curves, with feedback used to enhance motor synchronization and lane detection accuracy. The integration of these components enabled the vehicle to successfully follow lane boundaries autonomously under controlled conditions, marking a key milestone in the project's development.

3.3.2 Block diagram

This block diagram illustrates the control system of an autonomous robotic vehicle using a Raspberry Pi. The system begins with a power supply that provides energy to all components. A Raspberry Pi Camera captures real-time visuals of the environment, which are sent to the Raspberry Pi for processing. The Raspberry Pi runs a lane and edge detection algorithm to analyse the camera feed and determine the appropriate path for the vehicle. Based on the algorithm's output, control signals are sent to a motor driver, which regulates the DC gear motor's operation. The motor then drives the vehicle's wheels, enabling it to navigate autonomously while following lanes and avoiding edges.

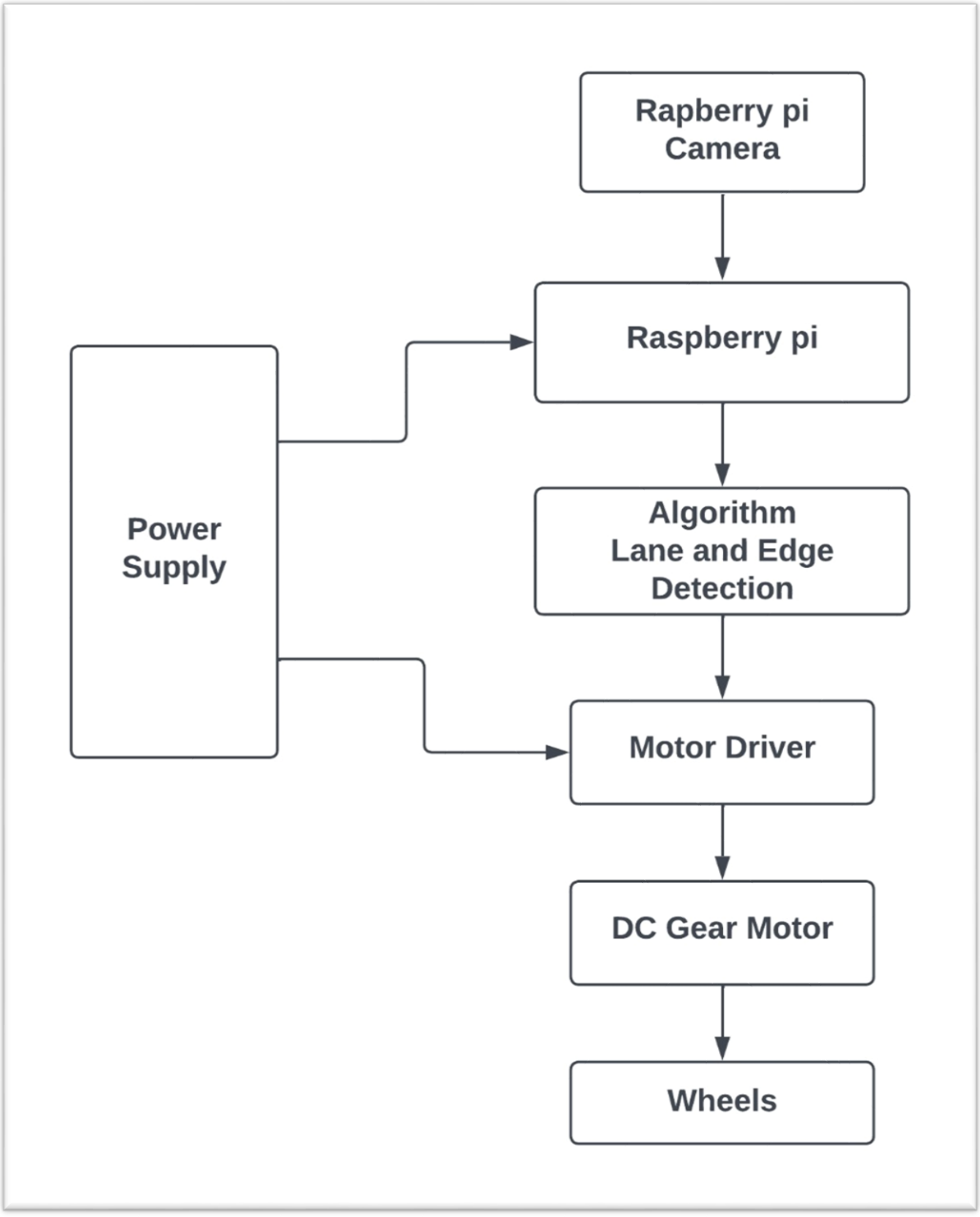

Fig 3.11 Block Diagram of model

3.3.3 Flow Chart

This flowchart illustrates the workflow of an autonomous robotic vehicle designed to detect lanes and avoid obstacles using a Raspberry Pi. The process begins with the system initialization, where all components, including sensors, camera, and algorithms, are prepared for operation. After initialization, the Raspberry Pi activates the camera to capture real-time video of the environment. The system checks if real-time lane detection is active; if not, it loops back to ensure activation. Once detection is active, the Canny Edge Detection algorithm is initiated to identify lane markings and edges in the captured video feed. This serves as the basis for guiding the vehicle along the correct path.

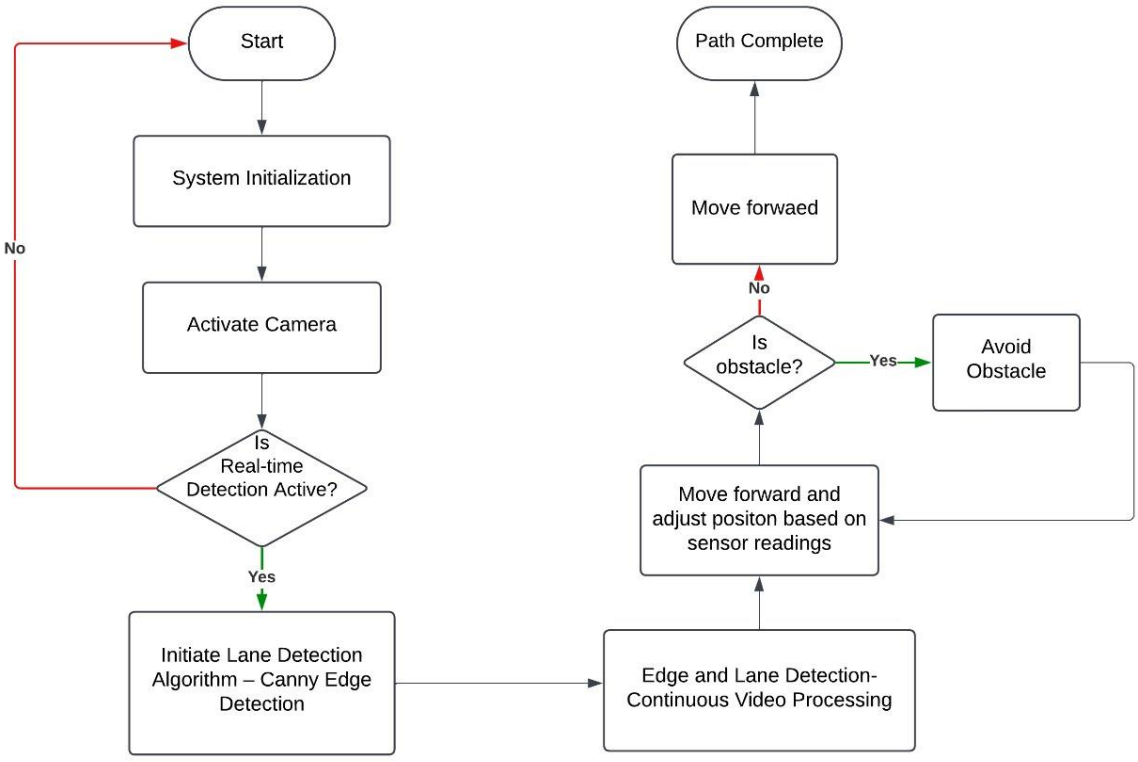

Fig 3.12 Flow chart of the work done

The system continuously processes video input to monitor lanes and surroundings. As the vehicle moves forward, sensors detect obstacles in its path. If an obstacle is present, the system triggers an obstacle avoidance manoeuvre, ensuring the vehicle bypasses the object safely. Following obstacle avoidance or confirmation of a clear path, the vehicle adjusts its position based on sensor readings to maintain alignment with the lane. This feedback loop ensures the vehicle continues moving forward safely and effectively until the designated path is completed.

Chapter – 4

4.1 Results & Analysis

Real-Time-Lane-detection

The system demonstrated the ability to detect lane boundaries in real-time using a robust computer vision pipeline. Techniques like Gaussian blur, thresholding, and the Hough Transform were implemented effectively, ensuring the identification of white lane boundaries even in the presence of minor environmental challenges such as shadows or uneven lighting. This real-time capability was critical for maintaining smooth and continuous navigation along the defined path.

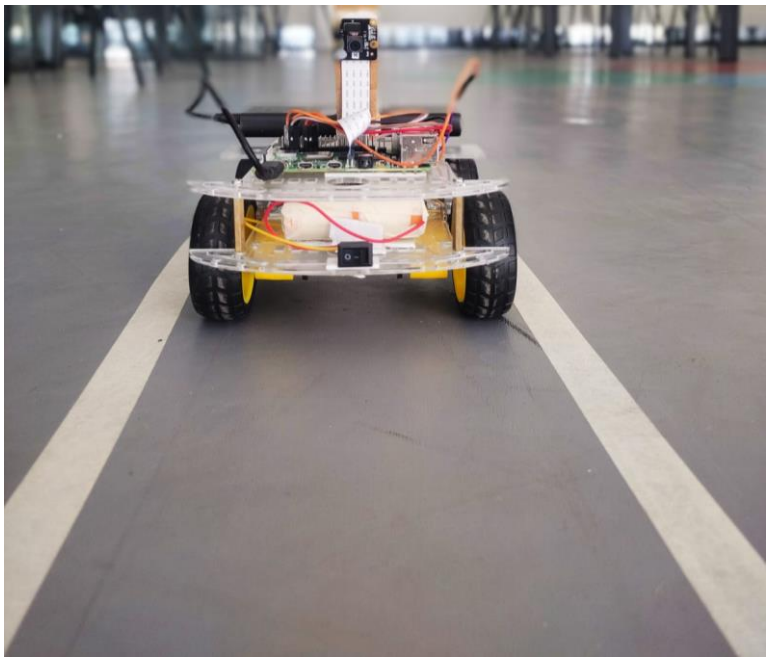

Fig 4.1 Real time lane detection

Autonomous-Navigation:

The car achieved autonomous movement by interpreting lane detection outputs and translating them into precise motor commands. The vehicle successfully demonstrated forward motion and smooth directional adjustments, such as left and right turns, based on the detected lane boundaries. This capability validated the seamless integration of the vision algorithm with motor control logic, enabling reliable lane-following behaviour without manual intervention.

Hardware-Integration-Success:

All hardware components were successfully integrated to work cohesively in the system. The Raspberry Pi 4 served as the primary computational unit for processing the camera feed and executing the lane detection algorithm. The Pi-camera provided high-resolution, real-time video input essential for accurate lane detection. The L298N motor driver ensured precise control over the DC motors by regulating speed and direction through PWM signals and GPIO inputs, while the servo motor contributed to fine directional control.

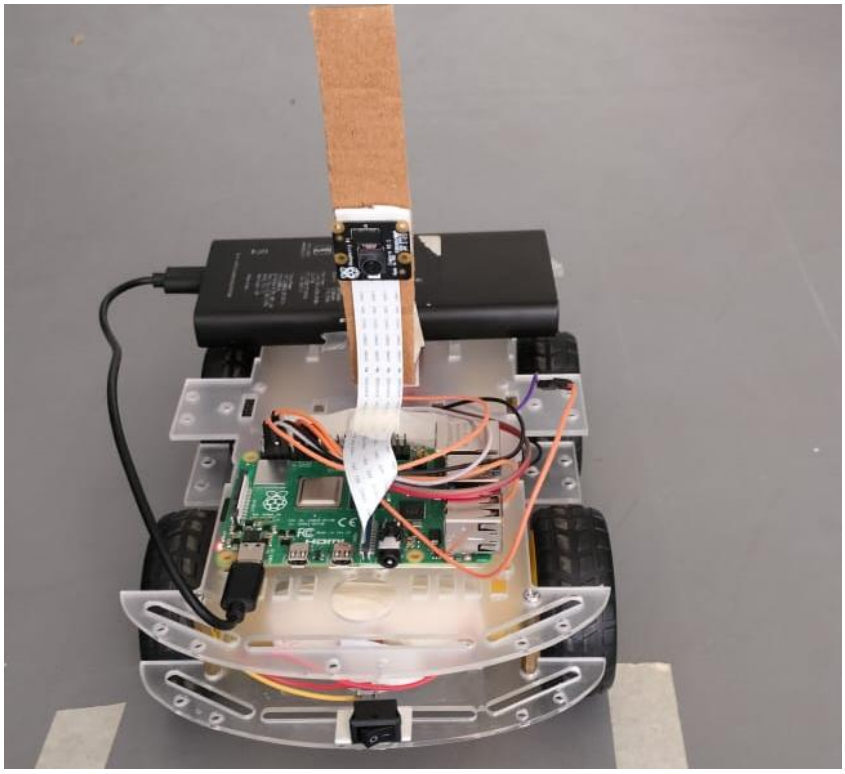

Fig 4.2 Hardware Integration

Stable-Performance:

The power system, powered by 3.7V lithium-ion batteries, provided consistent energy to all components, including the Raspberry Pi, motor driver, and motors, ensuring uninterrupted operation during testing. This stability was crucial for maintaining the reliability of the system, allowing the car to perform effectively over extended durations without fluctuations in performance.

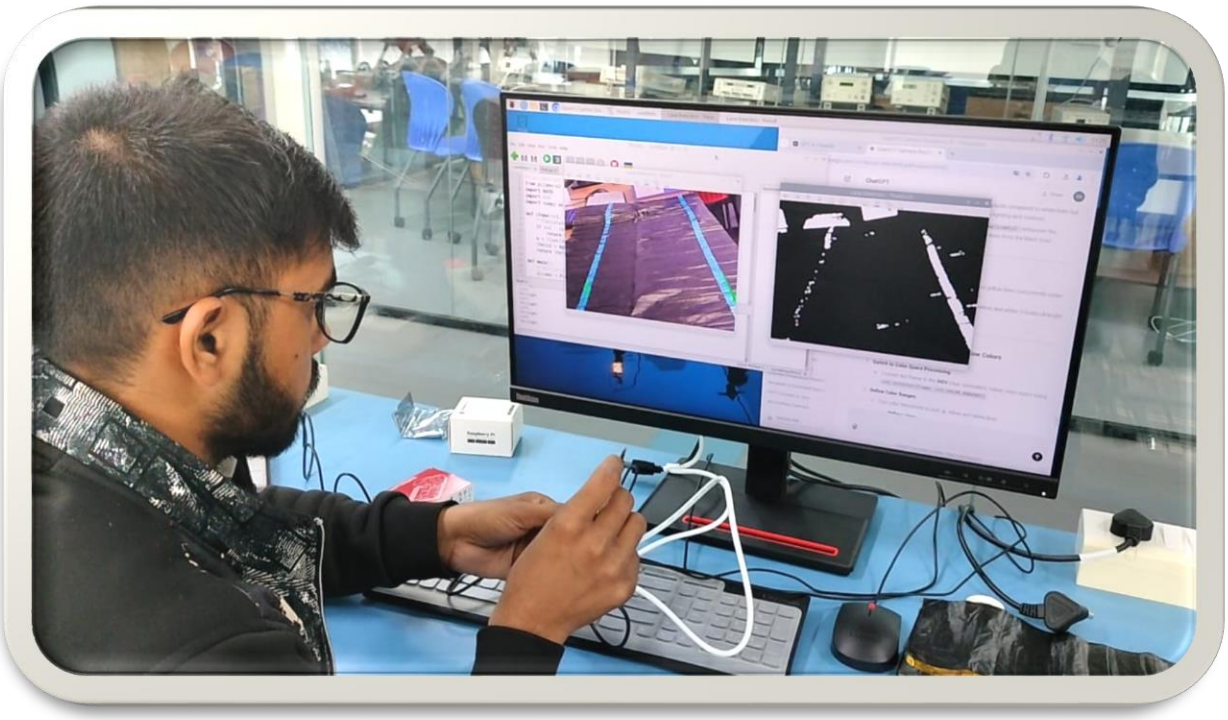

Fig 4.3 progress &results

Controlled-Environment-Validation:

In controlled testing environments, the model successfully navigated straight paths and gentle curves while maintaining its position within the lane boundaries. The system was able to process and respond to lane detection outputs quickly, ensuring minimal deviations from the intended path. This validation demonstrated the potential of the car to perform well in environments with defined lanes and consistent road conditions.

Chapter – 5

5.1 Conclusion

The Lane Detection Car Model represents a significant step toward autonomous navigation, successfully integrating computer vision algorithms and motor control mechanisms. It demonstrated real-time lane detection, precise motor synchronization, and reliable navigation under controlled conditions. The project effectively showcased the role of components like the Raspberry Pi 4, Pi-camera, L298N motor driver, and other hardware in achieving seamless functionality. While the system performed well in structured environments, it highlighted the need for further advancements to address complex real-world scenarios.

5.2 Future Stages

The future stages of this project will focus on enhancing the lane detection algorithm with machine learning for better adaptability to diverse road conditions and curve predictions. Feedback mechanisms such as encoders and PID controllers will be integrated to achieve precise speed and directional control. The system will be refined to handle environmental challenges like glare, shadows, and broken lane markings, ensuring robustness in real-world scenarios. Additionally, scalability will be addressed by incorporating features such as dynamic path planning, multi-lane navigation, and modular extensions for advanced functionalities in subsequent phases.

References

[1] Mohammad Ariyanto, Ismoyo Haryanto, Joga Dharma Setiawan, M. Munadi, M. Sri Radityo, *"Real-Time Image Processing Method* Using Raspberry Pi for a Car Model", *International Conference on Electric Vehicular Technology* (ICEVT), pp. 46-51, 2019.

[2] K.Vinothini, Dr.S.Jayanthy, "Road Sign Recognition System for Autonomous Vehicle using Raspberry Pi", *5th International Conference on Advanced Computing & Communication Systems*, pp. 78-83, 2019.

[3] Bianca-Cerasela-Zelia Blaga, Mihai-Adrian Deac, Rami Watheq Yaseen Aldoori, Mihai Negru, Radu Dnescu, "Miniature Autonomous Vehicle Development on Raspberry Pi", In *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing,* pp. 229-236, 2018.

[4] Vladimir A. Kulyukin, Vikas Reddy

Sudini, "Real Time Vision-Based Lane Detection on Raspberry Pi with 1D Haar Wavelet Spikes ", In *Proceedings of the International MultiConference of Engineers and Computer Scientists, 2017*

[5] Pravin T. Mandlik, Prof A. B. Deshmukh, "Raspberry-Pi Based Real Time Lane Departure Warning System using Image Processing", *International Journal of Engineering Research & Technology*, *5*(6), pp.762-766, 2016

[6] M. Malathi and R. Geetha, "*A Real Time Image Processing Based Fire Safety* Intensive Automatic Assistance System using Raspberry Pi," Int. *J. Mod. Trends Sci. Technol*., pp. 34–38, 2016.

[7] Vikas Reddy Sudini, *Real-time Vision-Based Lane Detection* with 1D Haar Wavelet Transform on Raspberry Pi. Master Thesis, Utah State University, 2017.

[8] Prasanth, M., K. S. Roshini, T. Pujitha, C. Sai Thanusha, C. Sai Mahesh, M. Purushottam Rao, and P. Rajesh, "*Design and Implementation of Smart Parking System Based on Raspberry Pi Advanced Microcontroller System*," Journal of Interdisciplinary Cycle Research, vol. XII, no. VI, pp. 960-965, 2020.

[9] Khanna, Abhirup, and Rishi Anand, "IoT based smart parking system," In 2016 *International Conference on Internet of Things and Applications* (IOTA), pp. 266-270, 2016.

[10] Oh, C. S., & Yoon, J. M. (2019, February). Hardware acceleration technology for deep-learning in autonomous vehicles. In *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)* (pp. 1-3). IEEE.

APPENDIX-I

Code for the process

Capturing video frame:

```
from picamera2 import Picamera2
import cv2
# Initialize Picamera2
picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"size": (640, 480)}))
picam2.start()
while True:
    frame = picam2.capture_array()
    if frame is None:
        print("Error: No frame captured!")
        break
    cv2.imshow("Camera Feed", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
picam2.stop()
cv2.destroyAllWindows()
```

Code for motor control:

```
import RPi.GPIO as GPIO
from time import sleep
# GPIO Setup
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
# Motor Class Definition
class Motor:
    def _init_(self, Ena, In1, In2):
        self.Ena = Ena
        self.In1 = In1
        self.In2 = In2
        # Set up GPIO pins
        GPIO.setup(self.Ena, GPIO.OUT)
        GPIO.setup(self.In1, GPIO.OUT)
        GPIO.setup(self.In2, GPIO.OUT)
        # Initialize PWM on Enable pin
        self.pwm = GPIO.PWM(self.Ena, 100)  # 100 Hz frequency
        self.pwm.start(0)  # Start with 0% duty cycle
    def move_forward(self, speed=100, duration=0):
        """Move motor forward at the specified speed and duration."""
        self.pwm.ChangeDutyCycle(speed)
        GPIO.output(self.In1, GPIO.HIGH)
        GPIO.output(self.In2, GPIO.LOW)
        sleep(duration)
    def move_backward(self, speed=100, duration=0):
        """Move motor backward at the specified speed and duration."""
        self.pwm.ChangeDutyCycle(speed)
        GPIO.output(self.In1, GPIO.LOW)
        GPIO.output(self.In2, GPIO.HIGH)
        sleep(duration)
```

```
    def stop(self, duration=0):
        """Stop the motor and wait for the specified duration."""
        self.pwm.ChangeDutyCycle(0)
        sleep(duration)
# Motor Initialization
motor1 = Motor(2, 3, 4)  # Replace pins as per your wiring
# Main Program Loop
try:
    while True:
        motor1.move_forward(30, 2)  # Move forward at 30% speed for 2 seconds
        motor1.stop(1)              # Stop for 1 second
        motor1.move_backward(100, 2)  # Move backward at 100% speed for 2 seconds
        motor1.stop(1)              # Stop for 1 second
        # Uncomment the below block for gradual speed control
        '''
        # Gradually increase speed from 20% to 100%
        for speed in range(20, 101):
            motor1.move_forward(speed, 0.05)
             print(f"Speed: {speed}%")
        # Gradually decrease speed from 100% to 20%
        for speed in range(100, 19, -1):
            motor1.move_forward(speed, 0.05)
            print(f"Speed: {speed}%")
        motor1.stop(5)  # Stop for 5 seconds
        '''
except KeyboardInterrupt:
    # Cleanup GPIO on program exit
```

```
    print("Program stopped by user.")
    GPIO.cleanup()
```

Code for Overall process

```
#!/usr/bin/env python3
import RPi.GPIO as GPIO
from picamera2 import Picamera2
import cv2
import numpy as np
import math
# Motor GPIO pins
ENA = 2  # Motor A speed (PWM)
IN1 = 3  # Motor A direction
IN2 = 4  # Motor A direction
ENB = 17 # Motor B speed (PWM)
IN3 = 22 # Motor B direction
IN4 = 27 # Motor B direction
def setup_gpio():
    """Set up GPIO pins for motor control."""
    GPIO.setmode(GPIO.BCM)
    GPIO.setup([ENA, IN1, IN2, ENB, IN3, IN4], GPIO.OUT)
    # Initialize PWM for speed control
    global pwm_a, pwm_b
    pwm_a = GPIO.PWM(ENA, 100)  # Frequency: 100 Hz
    pwm_b = GPIO.PWM(ENB, 100)
    pwm_a.start(0)  # Start with 0% duty cycle (stopped)
    pwm_b.start(0)
```

```
def stop_motors():
    """Stop both motors."""
    pwm_a.ChangeDutyCycle(0)
    pwm_b.ChangeDutyCycle(0)
    GPIO.output([IN1, IN2, IN3, IN4], False)
def move_forward(speed=50):
    """Move forward."""
    pwm_a.ChangeDutyCycle(speed)
    pwm_b.ChangeDutyCycle(speed)
    GPIO.output(IN1, True)
    GPIO.output(IN2, False)
    GPIO.output(IN3, True)
    GPIO.output(IN4, False)
def turn_left(speed=50):
    """Turn left."""
    pwm_a.ChangeDutyCycle(speed // 2)  # Reduce speed for turning
    pwm_b.ChangeDutyCycle(speed)
    GPIO.output(IN1, True)
    GPIO.output(IN2, False)
    GPIO.output(IN3, True)
    GPIO.output(IN4, False)
def turn_right(speed=50):
    """Turn right."""
    pwm_a.ChangeDutyCycle(speed)
    pwm_b.ChangeDutyCycle(speed // 2)  # Reduce speed for turning
    GPIO.output(IN1, True)
    GPIO.output(IN2, False)
```

```
    GPIO.output(IN3, True)
    GPIO.output(IN4, False)
def slope(vx1, vx2, vy1, vy2):
    """Calculate the slope of a line and its angle in degrees."""
    if vx2 - vx1 == 0:  # Avoid division by zero
        return 90 if vy2 > vy1 else -90
    m = float(vy2 - vy1) / float(vx2 - vx1)
    theta = math.atan(m)
    return theta * (180 / np.pi)
def main():
    setup_gpio()
    picam2 = Picamera2()
    picam2.configure(picam2.create_video_configuration(main={"size": (640, 480)}))
    picam2.start()
    print("Lane detection and autonomous driving started. Press 'q' to quit.")
    try:
        while True:
            frame = picam2.capture_array()
            frame = cv2.resize(frame, (640, 480))
            gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
            # Histogram equalization to enhance contrast
            equ = cv2.equalizeHist(gray)
            # Apply Gaussian Blur to reduce noise
            blur = cv2.GaussianBlur(equ, (5, 5), 0)
            # Threshold to isolate bright lane boundaries
            _, thresh = cv2.threshold(blur, 240, 255, cv2.THRESH_BINARY)
            # Focus on Region of Interest (ROI) for lane detection
```

```
roi = thresh[250:480, :]  # Bottom half of the frame
# Detect lines using Hough Transform
lines = cv2.HoughLinesP(
    roi, 1, np.pi / 180, threshold=25, minLineLength=10, maxLineGap=40
)
# Initialize lane detection flags
left_lane_detected = right_lane_detected = False
if lines is not None:
    for line in lines:
        for x1, y1, x2, y2 in line:
            x1 += 0  # Adjust for ROI offset
            x2 += 0
            y1 += 250
            y2 += 250
            angle = slope(x1, x2, y1, y2)
            if -80 <= angle <= -30:  # Right lane
                right_lane_detected = True
                cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
            elif 30 <= angle <= 80:  # Left lane
                left_lane_detected = True
                cv2.line(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
# Make decisions based on lane detection
if left_lane_detected and right_lane_detected:
    print("Moving Forward")
    move_forward()
elif left_lane_detected:
    print("Turning Left")
```

```
            turn_left()
        elif right_lane_detected:
            print("Turning Right")
            turn_right()
        else:
            print("Stopping")
            stop_motors()
        # Display the processed frame
        cv2.imshow("Lane Detection - Threshold", roi)
        cv2.imshow("Lane Detection - Result", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    except KeyboardInterrupt:
        print("Exiting program.")
    finally:
        stop_motors()
        pwm_a.stop()
        pwm_b.stop()
        GPIO.cleanup()
        cv2.destroyAllWindows()
        picam2.stop()
if _name_ == "_main_":
    main()
```