**A**

**Project Report**

**On**

# Self Driving Car

*Submitted in the fulfilment of the requirement for the award of degree of*

Bachelor of Technology

In

**COMPUTER ENGINEERING**

Submitted by

Manasi Pramod Gharat (2230331245503)

Priyanka Naresh Gothal (2230331245511)

*Under the guidance of*

Prof. Shweta N. Tembe


DEPARTMENT OF COMPUTER ENGINEERING

**DR.BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY**

Lonere-402103, Tal. Mangaon, Dist.Raigad(MS) INDIA

2024-2025

# Acknowledgement

We would like to express our sincere gratitude to all those who contributed to the successful completion of this project.

First and foremost, We are deeply thankful to **Prof. Shweta N. Tembe**, our project guide, for her valuable guidance, encouragement, and constant support throughout this journey.

Her insightful suggestions and expertise were instrumental in shaping the project and overcoming challenges. We extend our heartfelt thanks to the **Department Of Computer Engineering** and its faculty for providing us with the necessary resources and a supportive learning environment.

We are grateful to our peers and friends who provided constructive feedback and shared ideas, making this project more impactful. Finally, We express our deepest gratitude to our family for their unwavering support, patience, and motivation, which kept our focused and determined throughout the process.

Submitted By:

**MANASI PRAMOD GHARAT (2230331245503)**

**PRIYANKA NARESH GOTHAL (2230331245511)**

# Certificate


The report entitled **Self Driving Car** submitted by **Manasi Pramod Gharat (2230331245503)** and **Priyanka Naresh Gothal (2230331245511)** is approved for the partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Engineering**.

**Prof. Shweta N. Tembe**
Guide
Department Of Computer Engineering

**Dr. A. W. Kiwelekar**
Head
Department Of Computer Engineering

**Examiners:**

1. _____ (Name: _____)

2. _____ (Name: _____)

**Place:** Dr. Babasaheb Ambedkar Technological University, Lonere

**Date:** _____

# Abstract

**Self-driving cars**, frequently referred to as autonomous vehicles, are transforming the future of transportation. Road safety has been a longstanding concern, with over 1.3 million lives lost globally each year due to traffic accidents—many of which are avoidable. Increasing traffic congestion has significantly extended travel times, negatively impacting both individual productivity and environmental health. However, advancements in machine learning and artificial intelligence, combined with the growing capabilities of modern computing, have made it possible to harness these technologies in the development of autonomous vehicles.

This project focuses on the design and development of an autonomous vehicle system that leverages advanced technologies to enable self-driving capabilities. Autonomous vehicles use various techniques to perceive their surroundings, including environmental sensors and computer vision. The proposed system employs a combination of sensors and cameras to collect data, which is then processed using a machine learning model for object detection, tracking, and navigation.

The model used in this project is implemented using the Caffe framework. One file defines the model architecture layer by layer, including input, convolution, ReLU, and pooling operations, and is written in text format. Another file contains the trained weights and learned parameters of the model, such as weights and biases, and is stored in binary format. The Self-Driving Car Simulator is used to collect training data and evaluate the system's performance. The control system processes sensor input and accurately identifies objects and vehicles on the road to enable safe and efficient navigation.

In conclusion, this project addresses both road safety and traffic congestion through a comprehensive system combining **Deep Learning, IoT, and Machine Learning**, demonstrating the potential of autonomous vehicles to reshape modern transportation. The system's scalability, adaptability to changing road conditions, and ability to optimize routes make it a promising solution for the future of sustainable urban mobility.

# Contents

# List of Figures

# Project Synopsis

The increasing demand for safer, smarter, and more sustainable transportation has led to the rapid evolution of autonomous vehicle technology. Self-driving cars, also known as autonomous vehicles, are at the forefront of this revolution, offering solutions to some of the most critical problems faced in modern transportation namely, road safety, traffic congestion, human error, and pollution. This project focuses on designing and implementing a functional prototype of a self-driving car that can operate in a controlled environment with minimal or no human intervention.

The goal of the project is to demonstrate how intelligent technologies such as computer vision, machine learning, and the Internet of Things (IoT) can be integrated into an affordable and scalable autonomous driving system. The prototype is built using easily accessible components, including a Raspberry Pi (which serves as the processing unit), Arduino Uno (used for motor control), and a camera module (for visual input and data acquisition). The car is designed to perceive its surroundings, analyze the visual data in real time, and make autonomous decisions regarding navigation and obstacle avoidance.

At the core of the system is the Raspberry Pi, which handles the computational tasks involved in object detection, lane recognition, and decision-making using machine learning algorithms. The visual data captured by the camera module is processed using OpenCV and a trained Machine Learning Model. Based on the processed data, the system identifies the suitable response and transmits commands to the Arduino, which then manages the motors to control steering, acceleration, or braking of the car.. This hardware-software integration allows the vehicle to drive autonomously in an indoor track setup, making decisions based on live visual input.

This project goes beyond technical implementation by also examining the wider societal and environmental implications of autonomous vehicles One of the major advantages of self-driving cars is the potential to significantly reduce accidents caused by human errors such as distraction, fatigue, and impaired driving. Moreover, autonomous vehicles have the potential to improve traffic efficiency, lower fuel usage, and decrease greenhouse gas emissions. These aspects align with the goals of sustainable urban development and smart city initiatives.This project serves as a strong foundation for future research and development in the field of autonomous vehicles. It provides students with hands-on experience in combining embedded systems, artificial intelligence, robotics, and software engineering.

The prototype developed as part of this work can serve not only as an educational tool butalso as a stepping stone toward building more advanced self-driving systems. In conclusion,the Self-Driving Car project showcases the practical application of cutting-edge technologies to address real-world problems and demonstrates the potential of autonomous systems to shape the future of transportation.

# CHAPTER 1

# Introduction

The increasing reliance on transportation in modern society has amplified the demand for innovative solutions to enhance road safety, reduce traffic congestion, and promote sustainable mobility. Among these innovations, self-driving cars have emerged as a transformative technology that integrates the power of IoT (Internet of Things) and Machine Learning to address some of the most pressing challenges in transportation.

Every year, over 1.3 million people lose their lives in road accidents, with human error being a leading cause. Mistakes such as distracted driving, fatigue, and reckless behavior contribute significantly to these alarming statistics. Autonomous vehicles, designed to operate without human intervention, promise to mitigate these issues by leveraging advanced sensory systems and intelligent algorithms. These vehicles can perceive their environment, process real-time data, and make informed decisions, ensuring safer and more efficient transportation systems..

A self-driving car is equipped with an array of IoT-enabled sensors, such as cameras, ardiuno, raspberry pi , to collect and interpret environmental data. Machine learning algorithms further enhance these capabilities by enabling the vehicle to recognize patterns, predict outcomes, and respond to dynamic road conditions. Key functionalities include object detection to identify obstacles, path planning to determine optimal routes, and control systems to execute precise driving maneuvers.

This project aims to explore the integration of IoT and machine learning in building an autonomous vehicle prototype. By tackling issues like human error, traffic congestion, and environmental degradation, this project aims to showcase how self-driving technology can transform the future of transportation.. The solution also emphasizes real-time data processing, sensor fusion, and adaptive decision-making to create a reliable, intelligent trans- portation system.

In this report, we delve into the details of the self-driving car project, covering its objectives, benefits, challenges, implementation, and outcomes. This study emphasizes the role of technology in building safer, more intelligent, and environmentally friendly urban transportation systems, while addressing the challenges posed by today's complex road infrastructures.

# CHAPTER 2

# Objectives

The main aim of this project is to design and develop a working prototype of an autonomous self-driving vehicle that functions using real-time data and artificial intelligence. It involves the integration of multiple technologies, including hardware components like the Raspberry Pi, Arduino Uno, and camera modules, along with software tools such as Python, OpenCV, and Machine Learning Model to interpret and react to environmental inputs in real time.

A primary goal is to enable the vehicle to effectively sense its environment using vision-based sensors and analyze that information for decision-making. The system must detect key road elements, such as lanes and obstacles, and based on this information, control the vehicle's actions steering, acceleration, or braking as required.

Another essential objective is to showcase real-time functionality using affordable, easily available components. Unlike commercial autonomous vehicle systems, this prototype is developed with cost-effective resources, making it ideal for academic or experimental use. Utilizing the Raspberry Pi as the core processing unit provides flexibility, a compact design, and practical computing power within a limited budget.

Furthermore, this project aims to demonstrate seamless integration between hardware and software. Machine learning models must effectively interact with motor control mechanisms through the Arduino, ensuring that image-based decisions are accurately executed in the physical environment.

From an educational perspective, the project offers students hands-on experience with a multidisciplinary system that blends AI, IoT, robotics, and embedded technologies. It encourages the development of problem-solving skills, sparks innovation, and builds real-world engineering capabilities.

In addition, the project supports the broader vision of safer and smarter transportation systems. It explores how autonomous vehicle technologies can reduce human errors, lower accident rates, improve traffic efficiency, and minimize fuel usage and environmental impact.

Finally, the design is intended to be scalable, allowing for future enhancements such as GPS-based navigation for outdoor environments, cloud integration for remote access and data storage, and Vehicle-to-Vehicle (V2V) communication to support cooperative driving scenarios.

# Scope

The development of a self-driving car system follows a systematic approach that integrates advanced technologies such as computer vision, machine learning (ML), and the Internet of Things (IoT). The process begins with a detailed analysis of system requirements, including the selection of appropriate hardware like camera modules, radar sensors, and high-definition cameras, along with the necessary software tools for model development and simulation.

Large datasets are collected from both real-world driving scenarios and simulation environments. These datasets encompass a range of factors, including diverse road conditions, traffic behavior patterns, and environmental influences. The collected data is then preprocessed to remove noise and highlight critical features such as object contours, lane markings, and traffic indicators.

Machine Learning Model are utilized for detecting and tracking objects, while predictive models help anticipate the actions of surrounding vehicles and pedestrians. To enhance environmental awareness, sensor fusion techniques are employed to integrate data from multiple inputs such as cameras and radar into a unified representation of the surroundings.

Machine learning models are initially trained and validated in simulated scenarios to improve their performance before being tested in controlled real-world environments. Autonomous control systems are then implemented to manage key driving functions, including acceleration, braking, and steering, based on the processed sensor data.

Finally, the system undergoes continuous performance monitoring and refinement to enhance its accuracy, adaptability, and safety. This ensures that the vehicle can reliably operate under complex and dynamic road conditions.

# Purpose

The primary purpose of this study is to explore the feasibility of building an intelligent transportation system using widely accessible technologies and to investigate how artificial intelligence and IoT can be applied to solve real-world challenges in the domain of mobility. With the rising number of vehicles on roads and increasing incidents of accidents due to human error, there is an urgent need for smarter and safer driving solutions. This project seeks to be a step in that direction.

By designing and developing a self-driving car prototype, the project provides a practical understanding of how technologies like machine learning, computer vision, and embedded systems can be integrated into a real-world application. It aims to demonstrate that even with limited resources, it is possible to create a functional model that mimics the core behavior of a fully autonomous vehicle.

The project is also aimed at enhancing academic learning and research in the field of autonomous systems. It helps students and developers get hands-on experience with the challenges of processing real-time data, training machine learning models, and implementing control algorithms that can handle dynamic, real-world conditions. Through this, it promotes innovation, interdisciplinary learning, and critical thinking.

Another key purpose is to promote sustainable and efficient transportation systems. Autonomous vehicles can contribute to reducing fuel consumption and carbon emissions by optimizing route planning and minimizing idle time. By eliminating aggressive and erratic driving behaviors, they can help reduce wear and tear on vehicles and infrastructure.

In the context of society, the project aims to highlight how autonomous driving technology can provide mobility solutions to people with disabilities or the elderly, who may otherwise find driving difficult or unsafe. It also envisions a future where self-driving systems reduce the need for human intervention in transportation, thereby reducing fatigue-related and distraction-related accidents.

The broader purpose is to bridge the gap between theoretical knowledge and practical implementation. Through this project, learners and developers not only understand the underlying principles of autonomous systems but also gain insight into how such systems are built, tested, and refined in the real world. The project serves as a stepping stone for more complex applications, such as vehicle-to-vehicle communication, smart traffic systems, and AI-based mobility-as-a-service platforms.

In conclusion, this study aims to inspire further research, build technical skills, and encourage the adoption of intelligent technologies for real-world challenges in transportation, safety, and urban development

# CHAPTER 3

# System Requirements Specification

This Chapter describes about the requirements. It specifies the hardware and software requirements that are in order to run the system properly. The Hardware Requirement Specification is explained in details.

## 3.1 Hardware Requirements

- Arduino Uno
- Raspberry Pi
- Motors
- Camera Module
- Power Bank
- Motor Driver
- Battery

### 3.1.1 Arduino Uno

The Arduino UNO is a commonly used and foundational board in the Arduino lineup. The term ”UNO” means ”one” in Italian and was chosen to mark the first release of the Arduino software. It was also the first USB-compatible board introduced by Arduino. Recognized for its versatility and reliability, the Arduino UNO is widely used in various electronic and automation projects. It was developed by Arduino.cc and is built around the ATmega328P microcontroller.

Compared to other Arduino boards like the Arduino Mega, the UNO is easier to use, making it ideal for beginners and educational purposes. The board features a combination of digital and analog input/output (I/O) pins, support for shields, and several onboard circuits. Specifically, it includes 14 digital I/O pins, 6 analog input pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header.

The Arduino UNO is programmed using the Arduino IDE (Integrated Development Environment), which is available for both online and offline use. Among the 20 available pins, 6 support Pulse Width Modulation (PWM), 6 are analog inputs, and the remaining

8 are digital I/O pins. The board is equipped with a 16 MHz crystal oscillator, which provides accurate timing for operations.

Certain variants of the Arduino UNO are equipped with built-in Wi-Fi functionality, utilizing the ESP8266 module alongside the ATmega328P microcontroller.The input voltage range of the board is between 7V and 20V. It can automatically switch between drawing power from an external power source or via the USB connection.

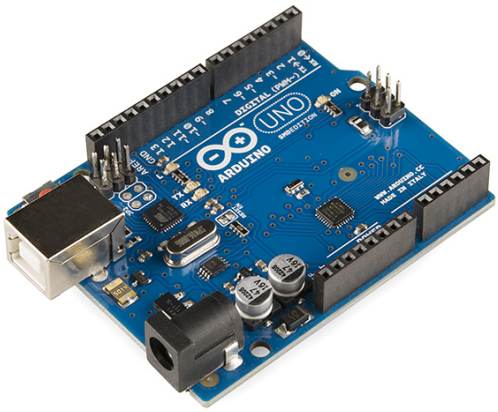


Figure 3.1: Arduino Uno

### 3.1.2 Raspberry Pi

The Raspberry Pi is an affordable, small-sized computer, comparable in size to a debit card. It connects to a TV or computer monitor and can be used with a regular mouse and keyboard. Similar to a conventional PC, it includes its own processor, memory, and graphics support.Despite its compact size, it is capable of handling everyday computing tasks such as browsing, coding, and media playback efficiently. It runs on its own operating system, Raspberry Pi OS, which is a customized version of Linux. The Raspberry Pi supports web browsing, HD video streaming, document editing, and even light gaming, offering functionality similar to that of a standard desktop.

It is also capable of interacting with external devices, making it ideal for digital maker projects such as music synthesizers, weather monitoring systems, motion-activated cameras, and more. Enthusiasts and learners around the world use it to explore programming and gain a deeper understanding of computer hardware and software.

While the Raspberry Pi doesn't include built-in storage, it supports microSD cards for installing and running operating systems like Raspberry Pi OS or Ubuntu Mate. With built-in Bluetooth, Ethernet, and Wi-Fi, it can easily connect to the internet and transfer files wirelessly. However, it's important to note that while some parts of the Raspberry Pi ecosystem are open for public use, the core hardware design and certain software elements are not open-source.

Figure 3.2: Raspberry Pi

### 3.1.3 Motors

Electric motors are electromechanical devices that facilitate the conversion of electrical input into physical movement. This is typically achieved through the interaction between the motor's magnetic field and electric current flowing through coil windings, generating torque on the shaft. In contrast, an electric generator works in reverse by converting mechanical energy into electrical power, even though both machines are structurally similar.

Electric motors can operate using either direct current (DC), sourced from batteries or rectifiers, or alternating current (AC), typically supplied by inverters, power grids, or generators. They are categorized based on various factors such as the type of electrical input, structural design, functional application, and the kind of motion they produce. Motors may be designed with brushes or be brushless, and can run on single-phase, two-phase, or three-phase systems. Additionally, they may use axial or radial magnetic flux, and their cooling can be managed through either air-based or liquid-based systems.

Widely used in industrial settings, standard motors power machinery and systems efficiently. Some of the largest motors, delivering outputs exceeding 100 megawatts, are used in large-scale operations like ship propulsion, water pumping stations, and pipelines. Since electric motors generate torque or linear motion to drive external components, they are considered actuators. While most are designed for continuous movement, devices like solenoids also convert electrical energy into motion, though only over a limited range.

Figure 3.3: Motors

### 3.1.4 Camera Module

The Raspberry Pi Camera Module is a compact yet powerful component designed specifically for Raspberry Pi boards, offering high-resolution photo and video capture. It features an 8-megapixel Sony IMX219 image sensor, capable of delivering still images at resolutions up to 3280 × 2464 pixels. For video, it supports 1080p at 30 frames per second, 720p at 60 fps, and 640x480p at 90 fps. The module interfaces with the Raspberry Pi using the Camera Serial Interface (CSI), which allows for fast data transfer and high-quality image streaming.

The camera includes a fixed-focus lens, although it can be swapped for specialized lenses like wide-angle or macro, depending on project requirements. Operating on 5V and drawing about 250mA of current, it is ideal for portable or battery-powered applications. Its small size measuring just 25mm by 24mm makes it easy to embed in a wide range of projects, including surveillance systems, robotics, computer vision, and time-lapse photography.

Compatible with Raspberry Pi OS, the module can be programmed using languages such as Python and C++, with support from image processing libraries like OpenCV. For more advanced imaging, the Raspberry Pi High-Quality Camera version is also available, which features a 12.3 MP Sony IMX477 sensor and supports interchangeable lenses for improved flexibility.

Typical use cases include object recognition, motion tracking, and live video streaming in applications such as IoT systems, home security, and automation projects. Due to its affordability, ease of use, and robust performance, the Raspberry Pi Camera Module remains a top choice for hobbyists, developers, and educators integrating vision into their designs.

Figure 3.4: Camera Module

### 3.1.5 Power Bank

A power bank is a portable charging device equipped with internal batteries, designed to recharge various battery-powered electronics such as smartphones, tablets, and other USB Type-C compatible devices. Some models also support wireless charging and can

power USB-C-based accessories like portable speakers, LED lights, handheld fans, and camera battery chargers. Typically, power banks provide power outputs ranging from 30W to 200W, depending on their capacity and design.

Well-designed power banks require efficient power management systems to ensure reliability, user safety, and a seamless charging experience. Users often prefer compact devices that deliver fast charging, high energy efficiency, and clear indicators for remaining battery level and other vital information.

From a safety standpoint, power banks must include necessary protection features to prevent hazardous conditions during operation. Although many power management integrated circuits (ICs) come with built-in protections, they depend on precise monitoring of voltage, current, and temperature to function correctly and avoid potential risks.


Figure 3.5: Power Bank

### 3.1.6 Motor Driver

A motor driver is a crucial electronic device that serves as an interface between an electric motor and a control unit, such as a microcontroller. It regulates the flow of electrical power to control the motor's speed, direction, and torque with precision. This ensures efficient motor operation across various fields, including robotics, automated systems, and electric transportation.

Motor drivers play a critical role in controlling motors, especially where direct control from a microcontroller is not feasible due to power limitations. These devices are commonly used in projects ranging from simple DIY systems to complex industrial automation setups.

One widely used example is the L298N motor driver module, which is particularly popular in embedded and robotic systems, such as small-scale self-driving car prototypes. It serves as a reliable interface between low-power controllers like the Raspberry Pi and

high-power DC motors that require more current than the Raspberry Pi's GPIO pins can handle directly.

The L298N module is built around a dual H-Bridge motor driver IC, capable of independently controlling two DC motors or a single stepper motor. It supports input voltages between 5V and 35V and delivers up to 2A per channel, making it suitable for medium-powered motors commonly used in robotics.

Additional features include built-in heat sinks for thermal protection and an onboard 5V voltage regulator, useful for powering logic-level components such as sensors. The dual H-Bridge setup allows for full bidirectional control of each motor, enabling complex movements such as forward and reverse driving, turning, and accurate steering—key functionalities in autonomous vehicle applications.

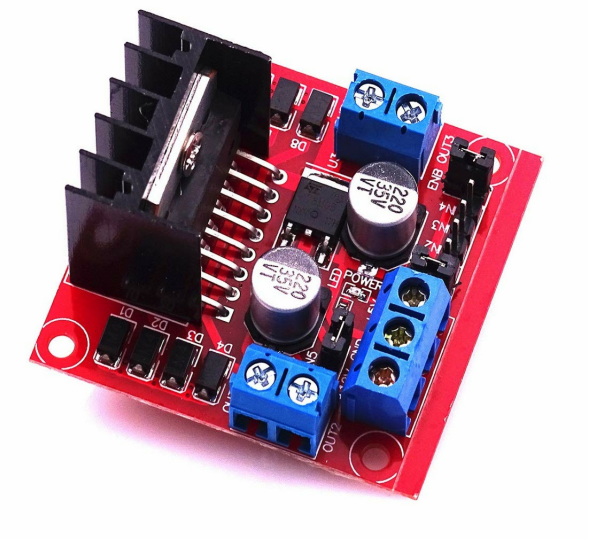


Figure 3.6: Motor Driver

### 3.1.7 Batteries

batteries play a critical role by serving as the primary power source for essential components such as the Raspberry Pi and motor drivers. A battery is an electrochemical system that converts stored chemical energy into usable electrical power on demand.These devices may consist of one or multiple electrochemical cells connected to provide a stable power output.

For compact embedded systems such as the Raspberry Pi, batteries provide essential portability and continuous functionality, particularly in the absence of a direct power source. Based on the project requirements, one can use either primary (single-use) or

secondary (rechargeable) batteries. However, in most prototype and development scenarios, rechargeable batteries are preferred due to their ability to undergo multiple charge-discharge cycles.

Rechargeable batteries operate through reversible chemical reactions. During discharge, they generate electricity by converting reactants into products. When recharged, the process reverses—electrical energy converts the products back into their original form. This cycle makes them ideal for projects that demand repeated and efficient energy use.

batteries in a self-driving car setup are vital for powering core electronics, ensuring system reliability, and supporting mobility and real-time processing without dependency on external power sources.


Figure 3.7: Batteries

## 3.2 Software Requirements

- IDE (Integrated Development Environment)
  - Arduino IDE
  - Raspberry Pi Imager
  - TigerVNC
- Operating System
  - Raspberry Pi OS
- Libraries
  - OpenCV (Open Source Computer Vision Library)
- Programming Languages
  - Python

### 3.2.1 IDE (Integrated Development Environment)

**1. Arduino IDE**

Arduino serves as an open-source electronic development platform featuring intuitive hardware and software, making it accessible to users of all skill levels.Arduino boards can take input from various sources such as sensors, buttons, or even online data—and respond by controlling outputs like motors, LEDs, or web-based actions. These tasks are programmed using the Arduino language (derived from Wiring) and developed within the Arduino Integrated Development Environment (IDE), which is based on Processing.

Over time, Arduino has become the foundation for countless projects, ranging from simple DIY applications to sophisticated scientific tools. It has cultivated a global community of developers, including students, hobbyists, engineers, artists, and programmers. This community has contributed a vast amount of open-source knowledge and resources, making it easier for newcomers to learn and experiment.

Originally developed at the Ivrea Interaction Design Institute, Arduino was created as a rapid prototyping tool for students with little or no experience in electronics or coding. As its popularity grew, the Arduino platform evolved to address diverse needs, expanding from basic 8-bit boards to advanced models supporting IoT, wearable tech, 3D printing, and embedded systems.

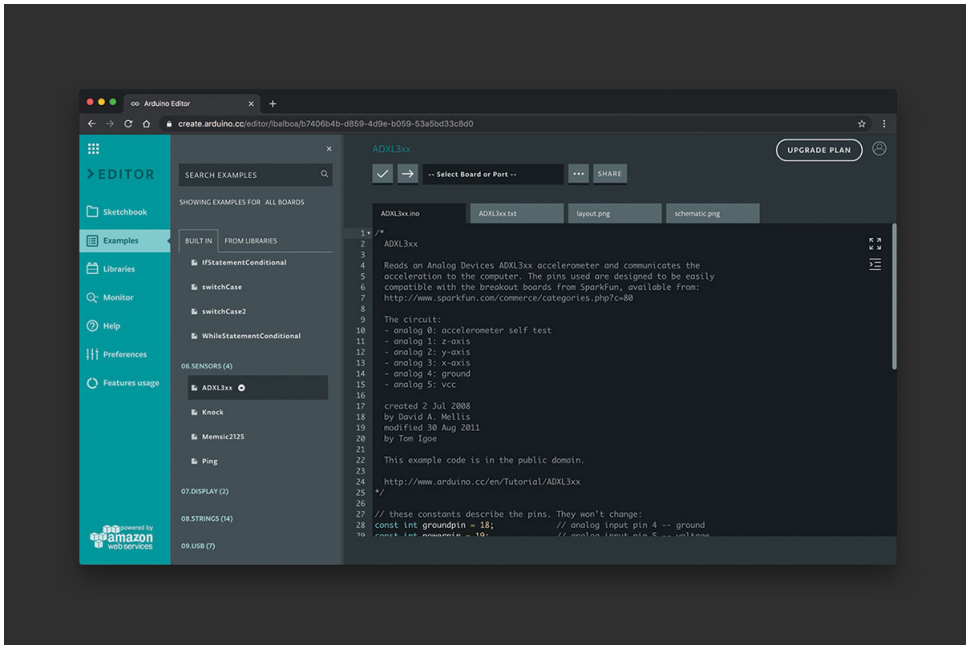


Figure 3.8: Arduino IDE

### 2. Raspberry Pi Imager

Raspberry Pi Imager is the official, open-source application created by the Raspberry Pi Foundation to simplify the process of installing operating systems onto SD cards or USB storage devices for use with Raspberry Pi computers. Its main objective is to offer a quick, user-friendly, and dependable way for users—whether beginners or experienced developers—to prepare bootable media for their Raspberry Pi devices.

Previously, setting up an OS for a Raspberry Pi required several manual steps: downloading the system image, decompressing it if needed, formatting the SD card, and using external tools such as balenaEtcher or Win32 Disk Imager to write the image. Raspberry Pi Imager consolidates all these actions into a single, intuitive interface, significantly cutting down setup time and minimizing user error.

The tool provides a curated selection of officially supported operating systems, including Raspberry Pi OS (formerly Raspbian), Ubuntu, LibreELEC (for media centers), and other third-party distributions tailored for specific purposes. Users also have the flexibility to flash custom operating system images by supplying their own .img or .iso files, making the software suitable for a wide variety of projects.

One of Raspberry Pi Imager's key advantages is its streamlined graphical user interface (GUI). Once launched, users simply select an operating system from the list, choose a destination storage device (like an SD card or USB drive), and begin the flashing process. The software handles formatting, writing, and verifying the image to ensure that the system will boot correctly on the Raspberry Pi.

In addition, Raspberry Pi Imager includes advanced configuration settings accessible through a keyboard shortcut or menu option before flashing. These settings allow users

to preconfigure essentials such as enabling SSH, entering Wi-Fi credentials, choosing locale and keyboard layout, and assigning a custom hostname. This functionality is particularly useful for headless setups, where the Raspberry Pi operates without a connected display or input devices.

The software is available across major platforms, including Windows, macOS, and Linux, making it widely accessible. As an open-source project, it also invites contributions from developers and the global community, encouraging ongoing improvements and feature expansions.

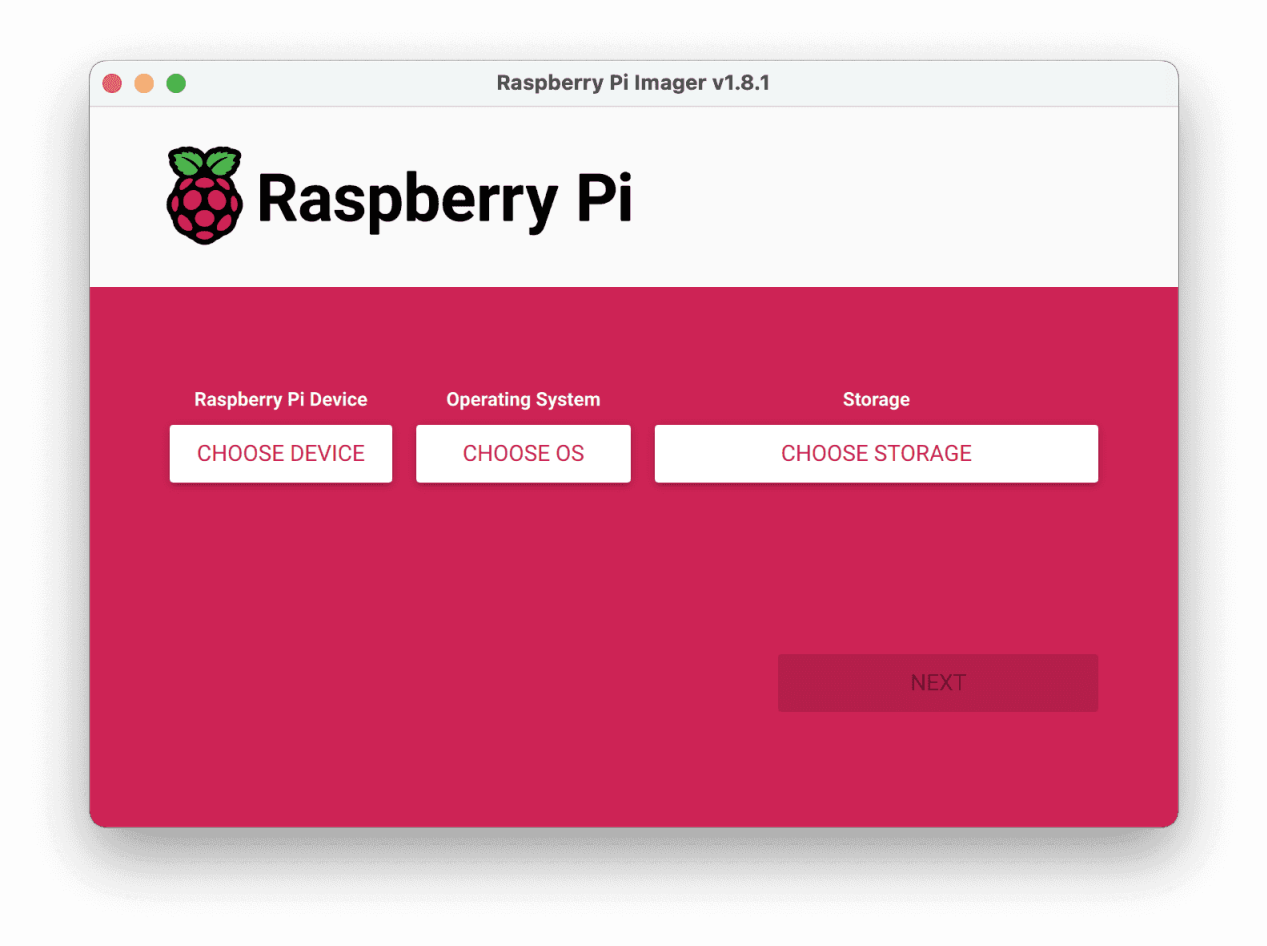


Figure 3.9: Raspberry Pi Imager

### 3. TigerVNC

TigerVNC is an open-source software project that implements Virtual Network Computing (VNC) technology, enabling users to remotely access and control graphical desktops over a network. TigerVNC is a high-performance, platform-independent solution that builds on the original VNC protocol with enhancements for modern usage, focusing on speed, responsiveness, and secure connections.

**Key Features of TigerVNC**

- **High Performance Remote Access:** TigerVNC offers fast and responsive remote desktop experiences by efficiently transmitting screen updates and input events.
- **Cross-Platform Support:** Compatible with all major operating systems such as Windows, macOS, and Linux, ensuring flexibility across various user environments.
- **Secure Connections:** Supports TLS encryption and UNIX authentication methods to secure remote sessions and protect against unauthorized access.
- **Open Source:** TigerVNC is actively developed under an open-source license (GPL), making it free to use and customizable for both individuals and organizations.
- **Virtual Session Support (Xvnc):** Allows creation of virtual desktop sessions on Linux servers, useful for headless systems and remote development environments.
- **Compatibility with VNC Standards:** Maintains compatibility with other VNC clients and servers, while introducing performance and security improvements.
- **No Cloud Dependency:** Unlike some other VNC providers, TigerVNC uses direct client-server connections without relying on cloud services, which enhances privacy and reduces latency.

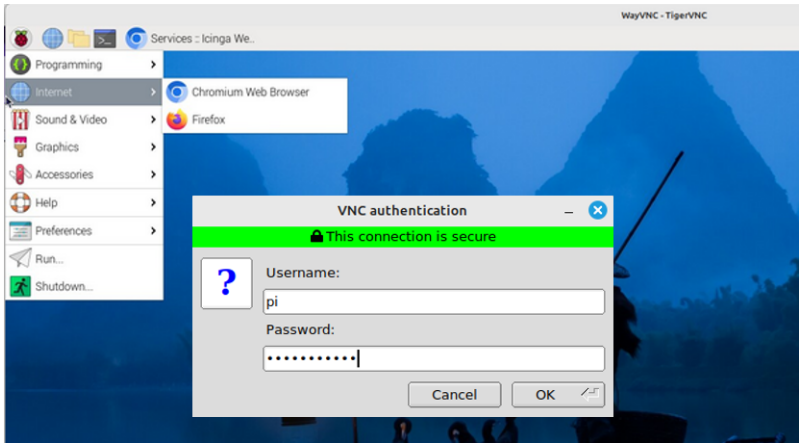


Figure 3.10: TigerVNC

### 3.2.2 Operating System

#### 1. Raspberry Pi OS

Raspberry Pi OS is a lightweight, open-source operating system built on the Debian Linux foundation, created specifically for Raspberry Pi boards. While it's tailored for Raspberry Pi devices, it's also capable of running on various other systems that use ARM architecture. The operating system was first known as Raspbian, a community project independently developed by Mike Thompson and Peter Green, with its earliest version launched on July 15, 2012.

During the early stages of Raspberry Pi development, there was no official OS available. To address this gap, the Raspberry Pi Foundation collaborated with the Raspbian project and began releasing it as their official operating system starting in September 2013. By 2015, it was established as the go-to OS for Raspberry Pi devices.
In May 2020, the Foundation introduced a test version of a 64-bit operating system. Unlike the 32-bit release that was based on Raspbian, the new version used packages from Debian GNU/Linux. To avoid confusion between the two, the Foundation rebranded both the new 64-bit OS and the existing 32-bit version under the unified name Raspberry Pi OS.

Eventually, the stable 64-bit edition was officially released on February 2, 2022, offering enhanced support for modern applications and hardware that require better memory handling and performance.

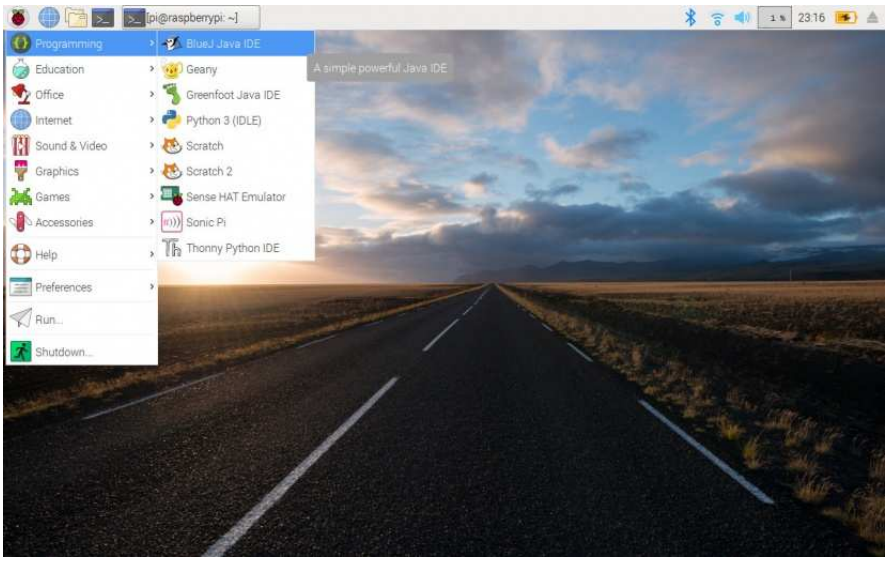


Figure 3.11: Raspberry Pi OS

### 3.2.3 Libraries

**1. OpenCV (Open Source Computer Vision Library)**

OpenCV, which stands for Open Source Computer Vision Library, is a widely-used software toolkit that supports real-time computer vision and machine learning applications. Initially developed by Intel and now maintained by OpenCV.org, this library enables efficient image and video analysis across multiple platforms, including Windows, Linux, macOS, Android, and iOS.
Although it is primarily written in C and C++, OpenCV also offers support for other programming languages such as Python and Java. It features a vast collection of optimized algorithms that assist in performing complex tasks like object detection, motion tracking, facial recognition, and visual data analysis.

**Key Features of OpenCV**

- **Image Processing:** Includes operations like filtering, transformations, thresholding, and histogram equalization.
- **Object Detection:** Real-time detection using Haar cascades, HOG features, and deep learning-based approaches.
- **Facial Recognition:** Offers face detection and recognition capabilities.
- **Video Analysis:** Tools for motion detection, object tracking, and background subtraction.
- **Machine Learning:** Comes with a module supporting SVMs, decision trees, k-NN, and other learning algorithms.
- **Cross-Platform Support:** OpenCV is designed to run across a variety of operating systems, such as Windows, macOS, Linux, Android, and iOS, ensuring wide compatibility.
- **Hardware Acceleration:** Supports CUDA, OpenCL, and Intel IPP for faster computations.

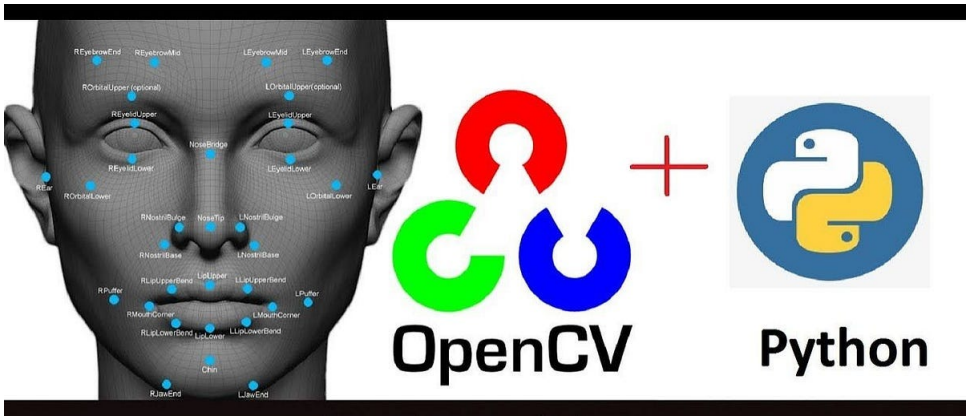


Figure 3.12: OpenCV

### 3.2.4 Programming Languages

#### 1. Python

Python is a key programming language in the development of self-driving car systems due to its ease of use, adaptability, and vast collection of libraries specifically designed for artificial intelligence, robotics, and sensor-based technologies. Its simplicity allows developers to prototype quickly, while its versatility makes it suitable for building robust, real-world applications. Python is extensively used in both academic projects and the automotive industry for implementing modules related to perception, path planning, and vehicle control. Applications of Python in Autonomous Driving

**Applications of Python in Autonomous Driving**

- **Computer Vision:** Libraries such as `OpenCV` are used for image and video processing tasks like lane detection, object recognition, and traffic sign classification.
- **Path Planning and Control:** Python is used to implement algorithms for route planning and vehicle control such as PID controllers and trajectory generation.
- **Simulation and Testing:** Simulation tools like `CARLA`, `Gazebo`, and `AirSim` provide Python APIs for realistic testing of autonomous driving systems.
- **Data Analysis:** Python uses tools like `Pandas`, `Matplotlib`, and `Seaborn` to process and visualize sensor data and vehicle logs effectively.
- **Embedded Systems:** Python is commonly used in Raspberry Pi-based autonomous car projects to interface with motors, sensors, and cameras.


Figure 3.13: Python Programming

# CHAPTER 4

# System Design

## 4.1 Block Diagrams

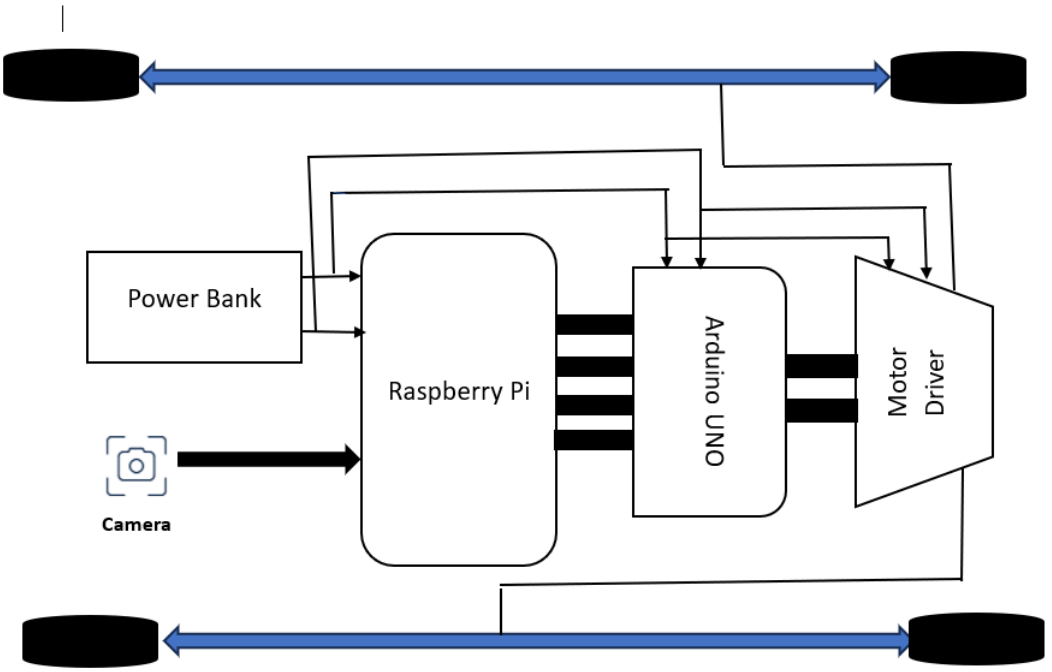


Figure 4.1: Block Diagram of Self Driving Car

The diagram shows an connection:

- Camera:

  Connected to the Raspberry Pi, it is likely used for capturing visual input, such as lane detection, object recognition, or navigation.

- Raspberry Pi:

  Acts as the main processing unit. It receives power from the power bank. It processes the data from the camera and communicates with the Arduino UNO for motor control.

- Arduino UNO:

  Functions as a microcontroller for controlling the motors. It interfaces with the motor driver to send control signals.

- Motor Driver:

  The motor driver interprets control signals sent by the Arduino UNO and adjusts the motors accordingly. It enables wheel movement by managing speed and direction, as represented by the arrows near the wheels.

- Power Bank:

  Supplies power to the Raspberry Pi, Arduino UNO, and possibly other components.

- Motors/Wheels:

  Controlled by the motor driver, enabling the movement of the system (robot or vehicle).

**Functionality:**

- Power Supply:

  The power bank ensures a steady power supply to the Raspberry Pi, Arduino UNO, and other components.

- Processing and Decision-Making:

  The Raspberry Pi processes visual data from the camera (e.g., detecting objects, following a path). It sends appropriate commands to the Arduino UNO.

- Motor Control:

  The Arduino UNO processes commands from the Raspberry Pi and converts them into control signals for the motor driver. The motor driver then modulates the wheel movements, enabling the vehicle to move and steer as needed.

- Real-Time Feedback:

  The system operates in a feedback loop where camera data influences motor adjustments for autonomous movement.

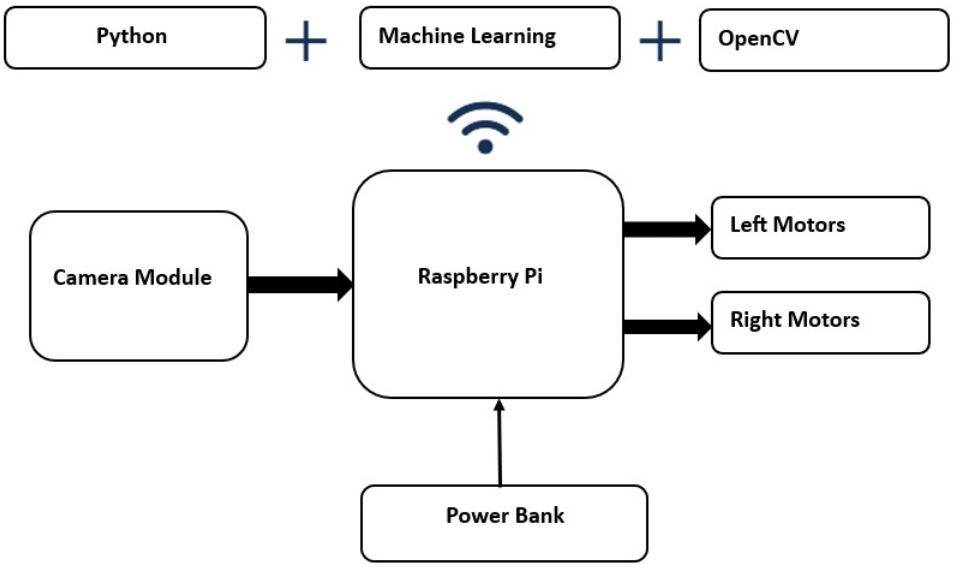


Figure 4.2: Block Diagram of Raspberry Pi

## 1. Programming Framework

- **Python + Machine Learning + OpenCV**: The system is programmed using Python as the core language. Machine learning is employed for decision-making, pattern recognition, or control tasks, while OpenCV (Open Source Computer Vision Library) is utilized for image and video analysis.

## 2. Hardware Components

- **Camera Module**: Captures live images or video of the surrounding environment and transmits the visual data to the Raspberry Pi for analysis.
- **Raspberry Pi**: Functions as the brain of the system. It processes input from the camera using machine learning algorithms and OpenCV to make autonomous navigation decisions.
- **Left and Right Motors**: Operated based on instructions from the Raspberry Pi, these motors control the movement and turning of the robot or vehicle.
- **Power Bank**: Serves as the power source for the Raspberry Pi and potentially other hardware, ensuring uninterrupted system operation.

## 3. Wireless Communication

- The wireless symbol above the Raspberry Pi suggests that the system might support remote control or monitoring over Wi-Fi or another wireless protocol. This could be used for live data transmission, debugging, or manual overrides.

## 4. Working Flow

- The **camera module** captures input (images or videos) and sends it to the **Raspberry Pi**.
- The **Raspberry Pi** handles input data by running algorithms written in Python. It utilizes tools such as **Machine Learning** and **OpenCV** to perform critical tasks like detecting objects, following predefined paths, and avoiding obstacles during navigation.
- Based on the analysis, the Raspberry Pi sends control signals to the **left and right motors** to adjust movement.
- The system is powered by a **power bank**, ensuring portability and consistent operation.

## 5. Applications

This setup is suitable for projects like autonomous navigation, line-following robots, or AI-based image recognition systems.

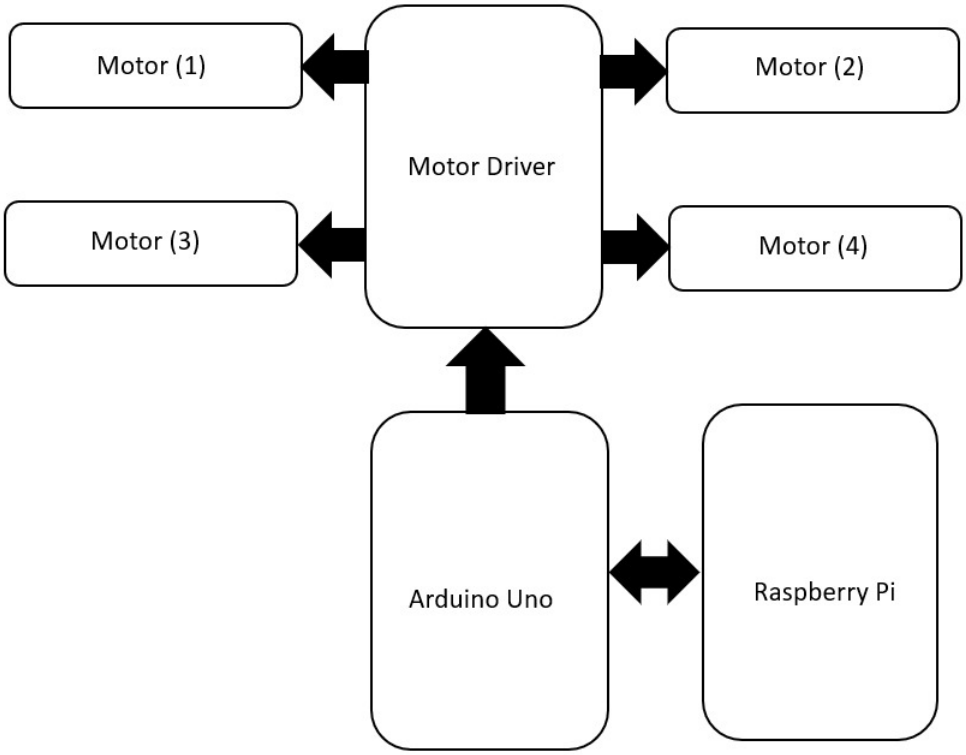


Figure 4.3: Block Diagram of Arduino Uno

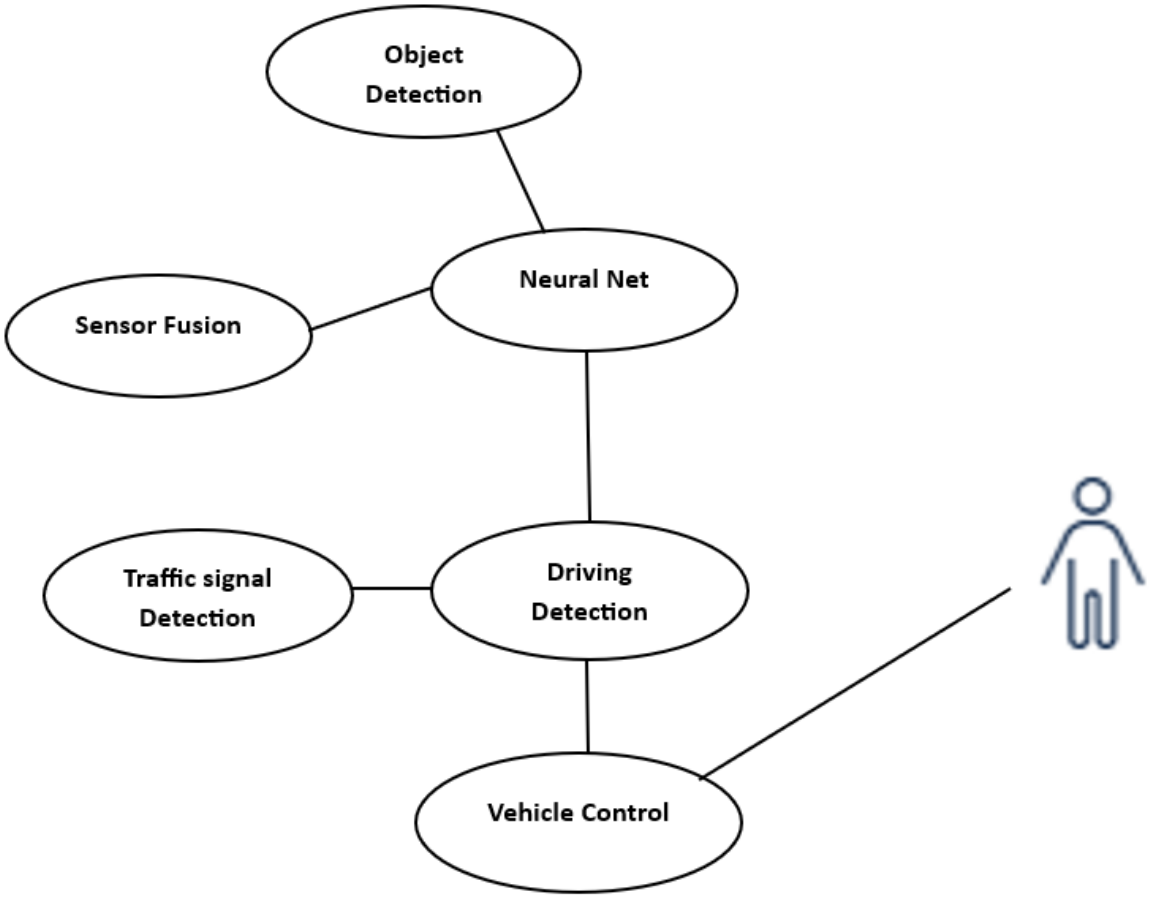


Figure 4.4: Activity Diagram

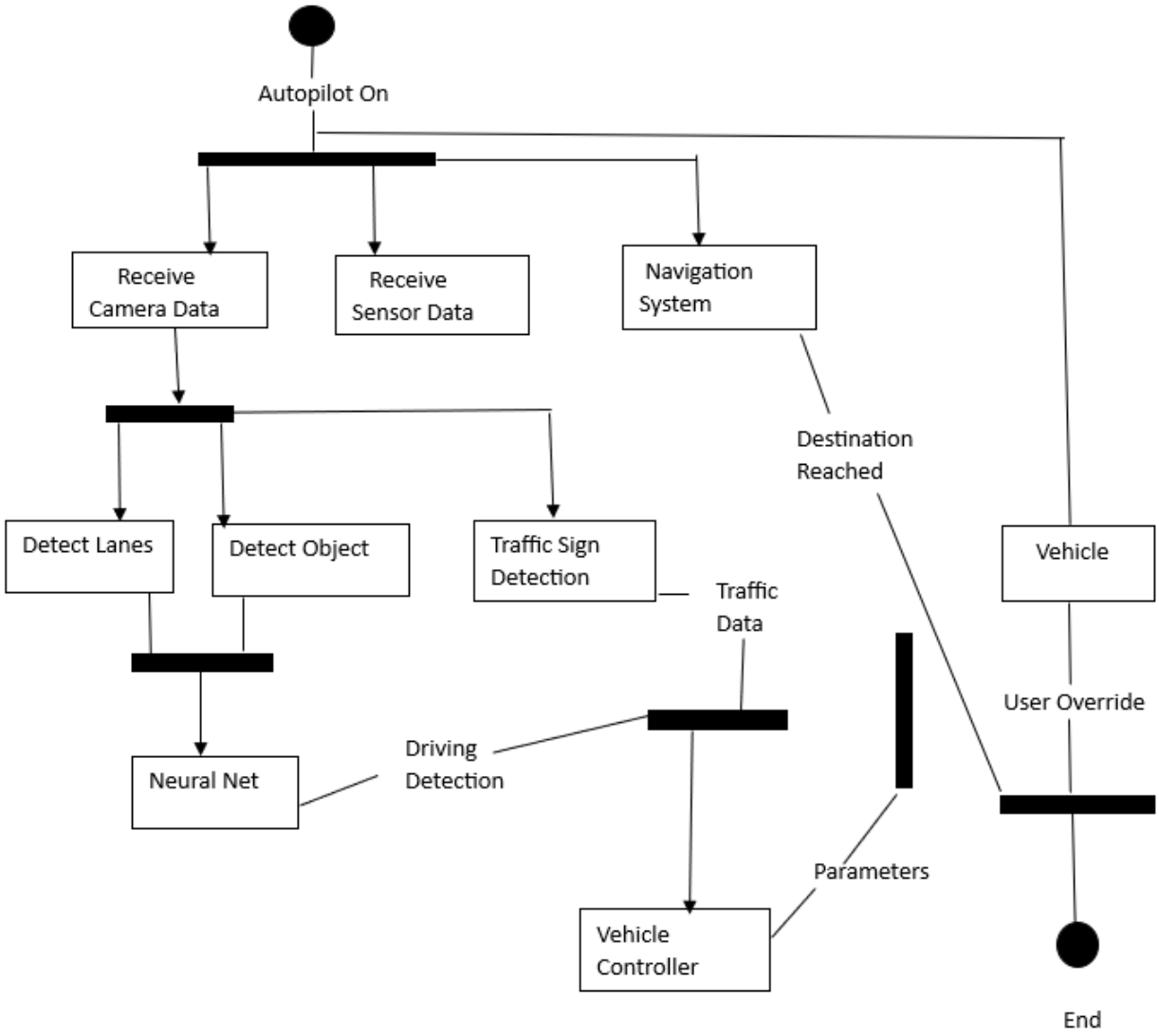


Figure 4.5: State Diagram

# CHAPTER 5

# System Flow

## 5.1 System Flow

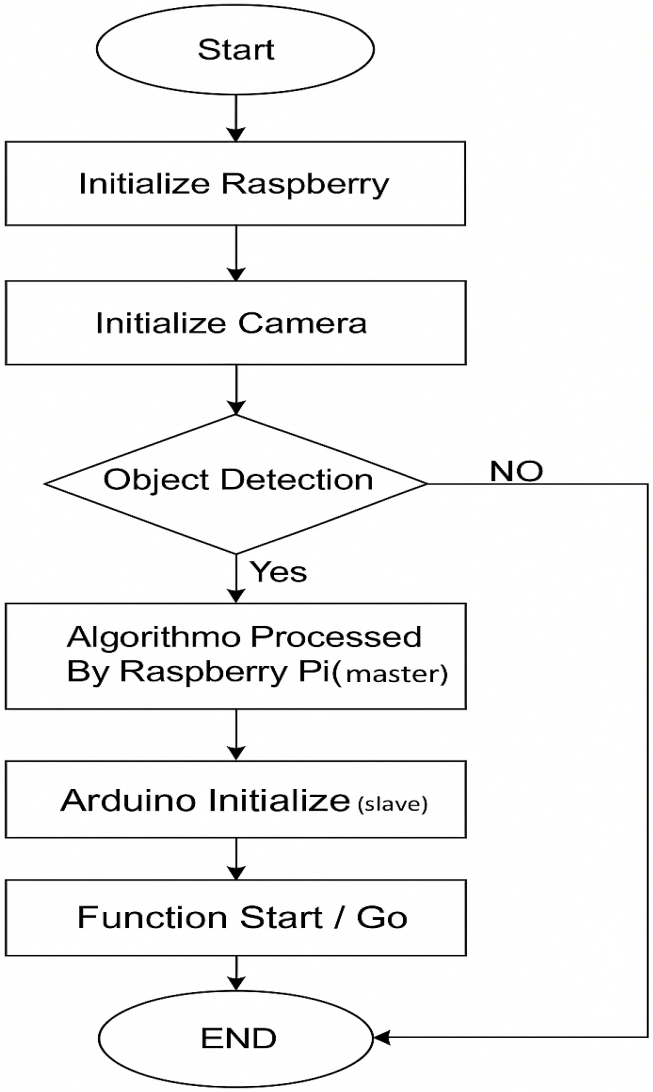


Figure 5.1: Flowchart of Self Driving Car

# CHAPTER 6

# Implementation

## 6.1 Circuit Diagram

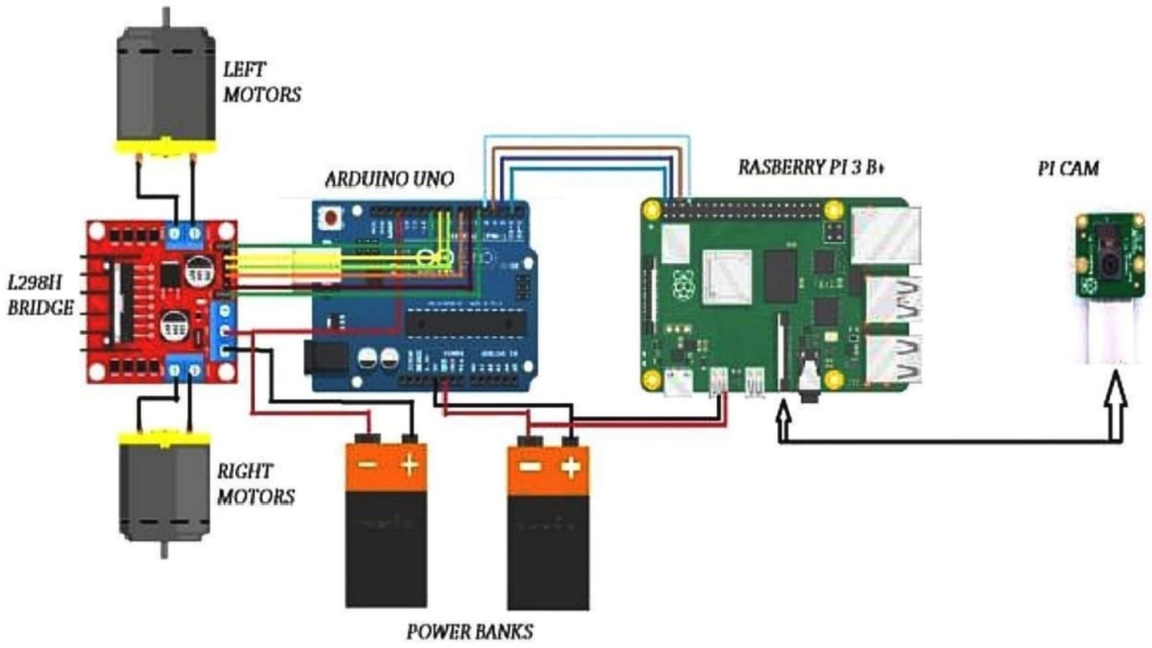


Figure 6.1: Circuit Diagram

# CHAPTER 7

# Result

## 7.1 Prototype of Self Driving Car

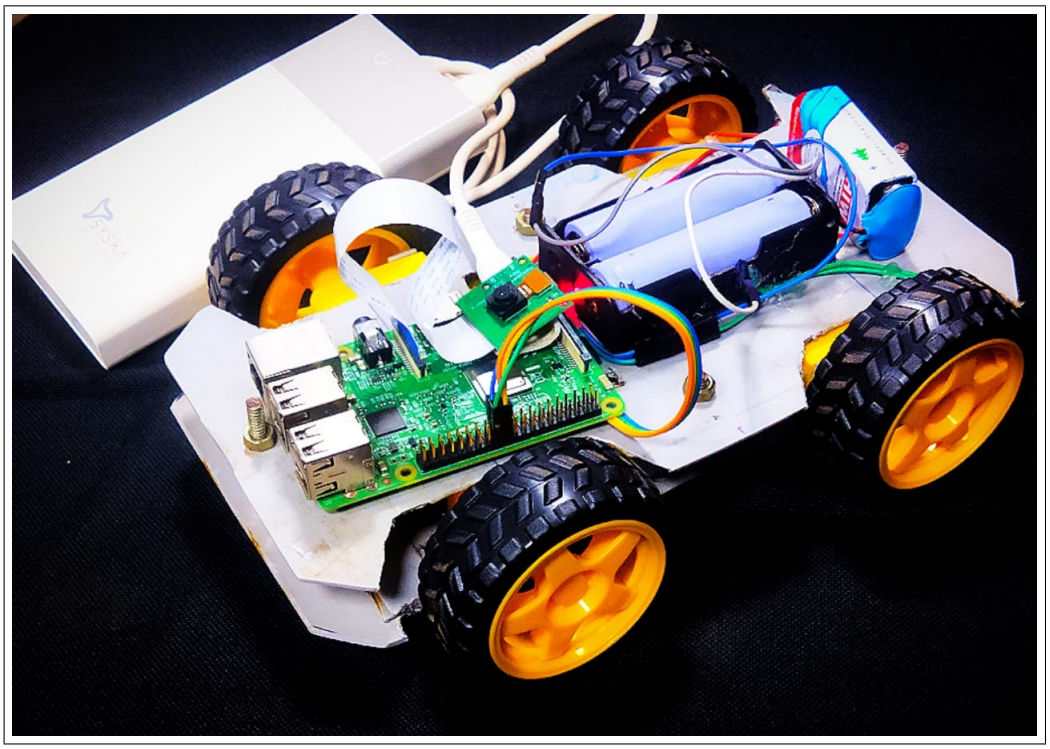

Figure 7.1: Prototype of Self-Driving Car

## 7.2 Images Capturing using Camera Module

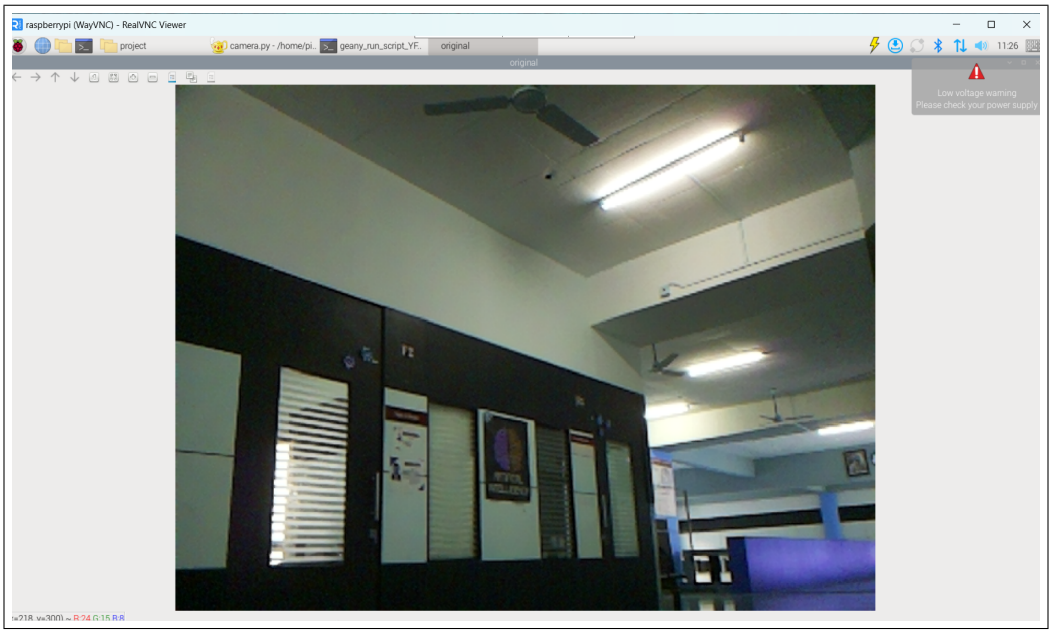


Figure 7.2: Capturing images using Camera Module on Raspberry pi (I)


Figure 7.3: Capturing images using Camera Module on Raspberry pi (II)

## 7.3 Object Detection using Raspberry Pi

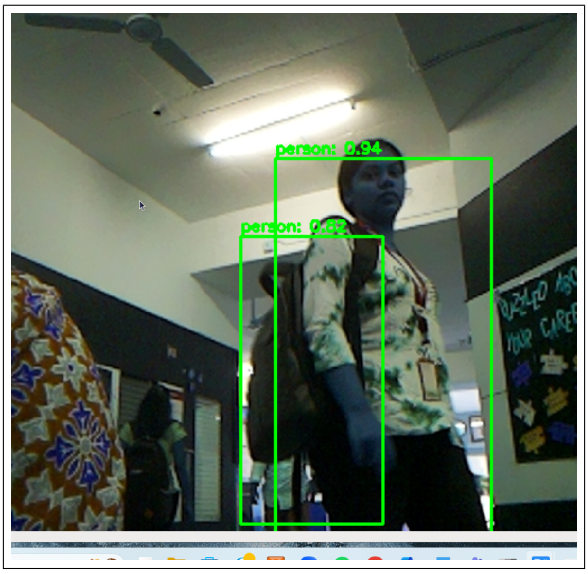


Figure 7.4: Object Detection using Raspberry Pi (I)

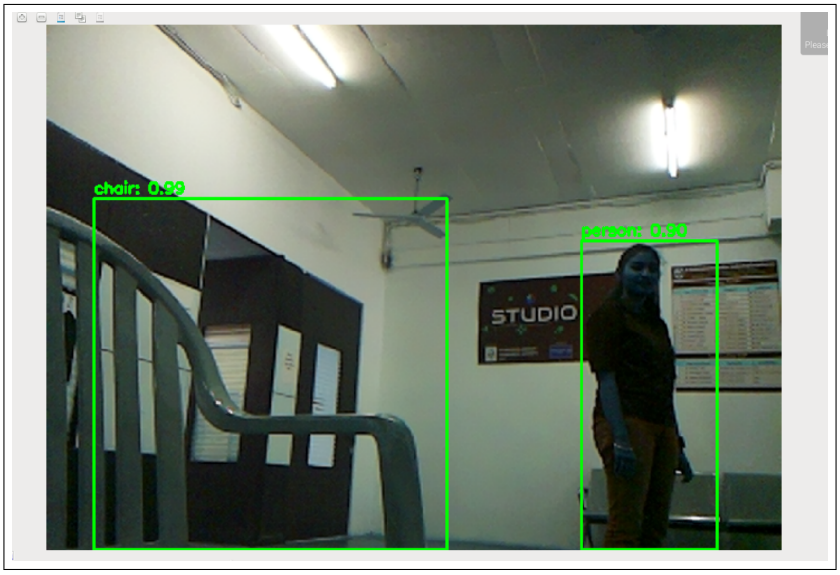


Figure 7.5: Object Detection using Raspberry Pi (II)

## 7.4 Stop Sign Detection

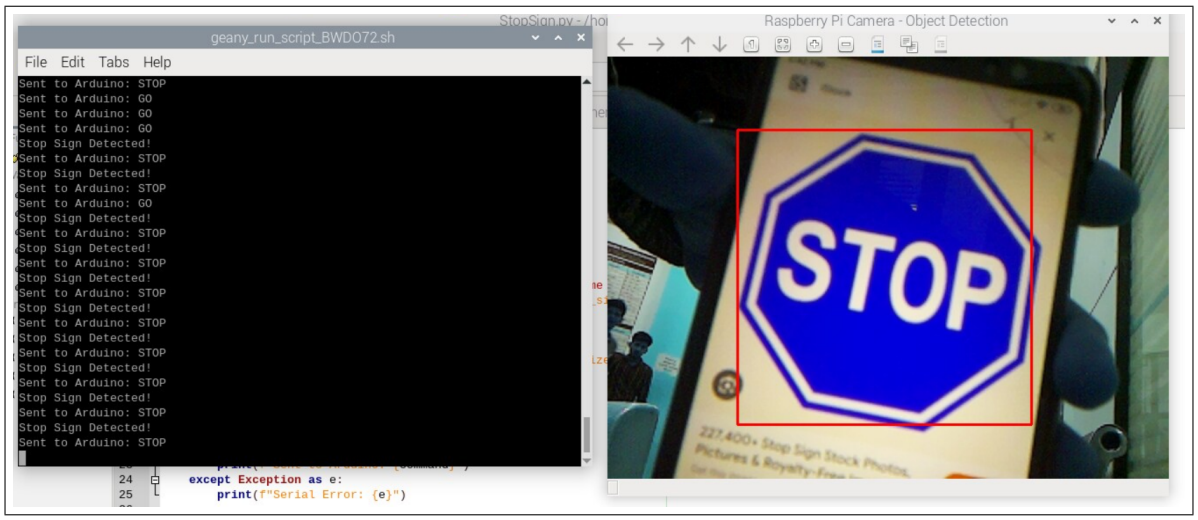


Figure 7.6: Stop Sign Detection

## 7.5 Object Detection using Arduino and Raspberry Pi

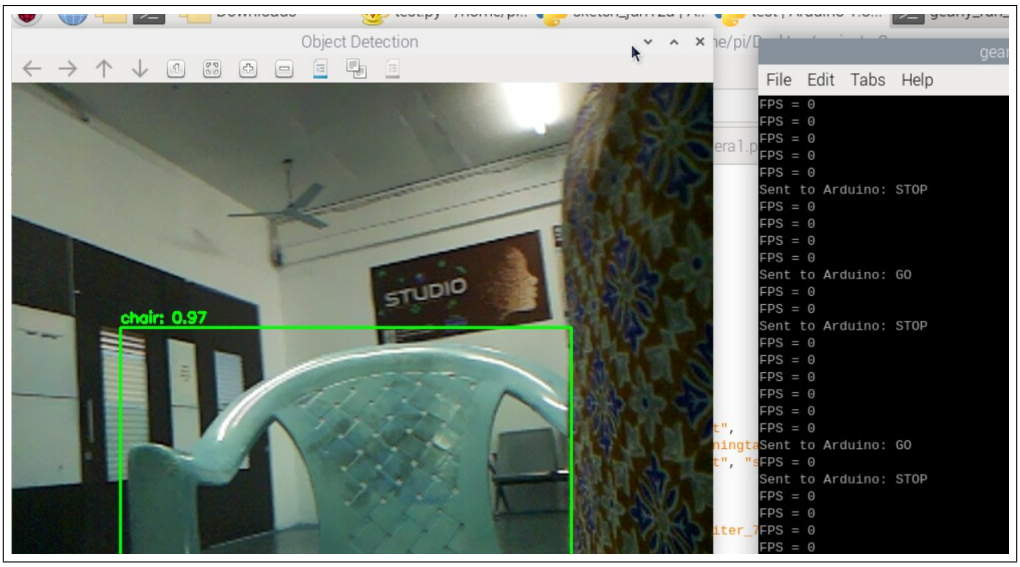


Figure 7.7: Object Detection (I)

# Advantages

1. **Increased Safety**:
   - Self-driving vehicles are designed to reduce the risks caused by human error, which accounts for a large number of road accidents. With the help of advanced technologies such as sensors, artificial intelligence, and cameras, these vehicles can recognize and respond to potential hazards more swiftly and accurately than human drivers.

2. **Reduced Traffic Congestion**:
   - Self-driving vehicles communicate with one another to better coordinate traffic flow. By maintaining consistent speeds, minimizing sudden stops, and adhering strictly to traffic rules, they help reduce congestion and ensure smoother, more efficient travel.

3. **Enhanced Mobility for Disabled and Elderly**:
   - For people with limited mobility due to age or physical disabilities, autonomous cars offer an opportunity for increased freedom and improved access to transportation, thereby enhancing their daily lives.

4. **Empowerment**:
   - By ensuring a secure and dependable means of transport, self-driving cars give individuals—especially those who may feel at risk—more freedom to travel independently and with peace of mind.

5. **Cost Reduction in Transportation**:
   - Over time, the adoption of autonomous vehicles can lead to financial savings by cutting down the need for professional drivers, lowering insurance rates, and improving vehicle durability through consistent and optimized driving behavior.

6. **Environmental Impact**:
   - When integrated with electric vehicle technology, autonomous systems can contribute to a cleaner environment by reducing harmful emissions and decreasing reliance on fossil fuels.

# Conclusion

In summary, the design and execution of this self-driving car project represent a significant advancement in the field of intelligent transportation systems. By integrating cutting-edge technologies such as computer vision, machine learning, and the Internet of Things (IoT), the project directly tackles critical issues related to road safety, traffic efficiency, and environmental sustainability. The system leverages Machine Learning Model for real-time object recognition, tracking, and navigation, enabling rapid and accurate decision-making. The combination of hardware components—including sensors, cameras, Arduino Uno, and Raspberry Pi—demonstrates a strong emphasis on building a complete and intelligent perception system. This seamless interaction between hardware and software enhances the system's reliability in varied scenarios and allows it to adapt to dynamic conditions, such as traffic congestion or unpredictable weather. The development of a Self-Driving Car Simulator further strengthens the project, providing a controlled environment for algorithm testing, model training, and performance evaluation. This bridge between simulation and real-world implementation ensures robustness and reduces risks during deployment. Overall, the project highlights the transformative potential of autonomous vehicles in shaping future transportation. It supports global initiatives for smarter urban infrastructure, improved road safety, and reduced environmental footprint. With its scalability, adaptability, and focus on sustainable mobility, this self-driving car prototype offers a promising step toward a more intelligent, safer, and eco-friendly transportation future.

# Future Scope

- **Improved Safety Systems:** -Future advancements in AI algorithms, sensor technology, and predictive analytics will further enhance the ability of self-driving cars to identify and respond to road hazards, reducing accident rates to near-zero levels. Integration of real-time data from connected vehicles and infrastructure can make the system even more robust.
- **Vehicle-to-Everything (V2X) Communication:** - V2X enables real-time interaction between autonomous vehicles and external systems such as traffic lights, road infrastructure, and surrounding vehicles. This communication helps vehicles make informed decisions, enhances coordination on the road, and contributes to smoother and more efficient traffic flow.
- **Advanced Machine Learning Models:** - Implementing cutting-edge AI techniques like deep learning and reinforcement learning enhances the decision-making abilities of autonomous vehicles. These advanced models enable the system to manage complex driving situations, including dense traffic, unpredictable weather, and changing road conditions more effectively.
- **Personalized User Experience:** - Future self-driving cars can focus on providing personalized experiences for passengers, such as customized entertainment systems, automated navigation preferences, and health monitoring during travel.
- **Environmentally Sustainable Solutions:** - Combining autonomous driving systems with electric or solar-powered vehicles promotes a greener transportation ecosystem. This integration helps lower carbon emissions and supports efforts to mitigate climate change.
- **Cost Optimization:** - Over time, as technology becomes more affordable and accessible, self-driving systems can be incorporated into mid-range and low-cost vehicles, making autonomous transportation available to a wider population.

# References


- Goodall, N. J. (2014). **Machine Ethics and Automated Vehicles**. In Road Vehicle Automation (pp. 93-102). Springer Vieweg, Berlin, Heidelberg.
- Gkartzonikas, C., Gkritza, K. (2019). **The impact of autonomous vehicles on public transport: A review of the literature.** Transportation Research Part C: Emerging Technologies, 101, 83-92.
- J. Christian Gerdes, Mike P. McKinley, and David C. T. Nguyen (2022). **Autonomous Vehicles: Opportunities, Strategies, and Impacts**. Elsevier.
- T.Do, M. Duong, Q. Dang and M. Le, "Real-Time SelfDriving Car Navigation Using Deep Neural Network," 2018 4th International Conference on Green Technology and Sustainable Development (GTSD), 2018, pp. 7-12, doi: 10.1109/GTSD.2018.8595590
- H. Wu, Y. Liu and J. Rong, "A Lane Detection International Conference on Information Engineering and Computer Science, 2009, pp. 1-4, doi: 10.1109/ICIECS.2009.5365404.Approach for Driver Assistance," 2009
- Jonathan Hui, "Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3" [Online]. Available: [Online]. Available: "https://medium.com/ @jonathan$_hui/real-timeobject-detection-with-yolo-yolov2-28b1b93e2088$"
- Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi "You Only Look Once: Unified, Real-Time Object Detection" 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)