A

PROJECT REPORT

ON

"(SELF DRIVING CAR)"

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE AWARD OF

DIPLOMA

IN

ELECTRONICES & COMMUNICATION ENGINEERING

BY

SUBMITTED BY

PIYUSH RAJ (19GPTC4070019)

GUIDED BY

MS. AARTI NEEMA MAHAJAN

GALGOTIAS UNIVERSITY

Plot No.2, Sector 17-A Yamuna Expressway, Greater Noida, Gautam Buddha Nagar, Uttar Pradesh, India, Session (2021-2022)

GALGOTIAS UNIVERSITY, GREATER NOIDA.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

2021-22

CERTIFICATE

This is to Certify that the project report entitled "**SELF DRIVING CAR**" Was Successfully completed by Student of sixth semester Diploma in ELECTRONICS & COMMUNICATION ENGINEERING. Submitted by PIYUSH RAJ is accepted in partial fulfilment of the requirement for the award of Diploma in ELECTRONICS & COMMUNICATION ENGINEERING

Prof. Mohit Gaharwar	Ms. Aarti Neema Mahajan	Ms. Aarti Neema Mahajan
Principal	HOD	Project Guide

ACKNOWLEDGEMENT

This Major Project would not have been possible without the help of many people. Hereafter, I would like to thank them for their help, support, and/or advice they have given me while working on the project.

I would like to express my gratitude towards internal project guides **Prof. & HOD Aarti neeam mahajan** of Electronices & Communication engineering department for their constant guidance and supervision as well as for providing necessary information regarding the project.

Finally, I would like to thank my college, **Galgotias University** for this opportunity to work on this project.

It is noteworthy that this project has given me a fantastic experience which will prove beneficial in future when I will actually work for IT firms. The overall exposure to different technologies was really worth experiencing and the project was indeed a good tunnelling path which had darkness of difficulties in its middle but also a glowing light of success at its end.

ABSTRACT

In this contemporary era people are continuously trying to save time in their life. People are spending lots of time driving and travelling. They can save their time by using public transport but not every country has good facilities. So, Self-Driving Cars are a boon for humans in terms of safety and time saving. People can do work in autonomous vehicles or they can take a nap which helps to increase working efficiency. It is an arduous job to make independent machines because we don't know the driving environment and situation. Computer vision and sensor fusion are parts and parcel of the auto driving. Based on the sensors and camera results the car will be moved. Carla simulator used for testing purposes. It provides the best environment to check all components of a self-driving car. We made a car using DC motors, batteries and that operates using Raspberry pi, Arduino uno. Two neural networks are the backbone of this project. First network decides the move according to the track situation and the other one identifies objects on the scene. There were two ways, using which we could make this project; Vision and Deep learning and Lidar and Deep learning. There are many related to autonomous driving such as trolley problem, confusion in vision, but they can be avoided using good algorithms and the best equipment.

Table of Contents

CERTIFICATE …… **2**

ACKNOWLEDGEMENT ……. **3**

ABSTRACT ……. **4**

Chapter 1 : Introduction ……. **8**

Introduction …… **9**

History ……. **10**

Motivation ……. **11**

Objectives …… **12**

Project Modules …….. **12**

Softwares …… **13**

Spyder ……. **13**

VNC Viewer …… **13**

Lan Scan …… **14**

Etcher …… **14**

Hyper Label ……. **14**

Google Colab …….. **14**

Android studio …… **14**

Applications …… **15**

Advantages …….. **15**

Disadvantages …… **15**

Chapter 2 : System Analysis and Design ……. 16

Implementation Requirements …….. 17
Raspberry Pi 3B+: …… 17
Arduino Uno : ……. 18
L298N Motor Driver: ……. 19
Camera: …… 20
Dual Shaft BO Motor ……. 21
Lithium Battery 2200mh ……. 22
Libraries …… 23.25
Chapter 3 : UML Diagram …… 26
Use Case Diagram …….. 27
Data Flow Diagram …….. 29
CIRCUIT DIAGRAM …….. 30
Chapter 4 : Hardware based Implementation …… 31
Car Design …… 32
Parts of Car …… 32
Raspberry Pi Interfacing: …… 32
Car Photo …… 33
L298N Motor Driver …… 34
PWM – For controlling speed …… 35
For controlling rotation direction ……. 35
Detection System …….. 35
Lane Finding …… 36
The Pipeline ……. 36
Converting RGB image into HSL Image ……. 37

Chapter 5 : CODING …… 38

RASPBERRY PI CODING …….. 39,44

ARDUINO UNO CODING …….. 45,49

Result Analysis …… 50

Conclusion and Future Work …… 51

Chapter 1 : Introduction

Introduction

Smart cars have been a long-researched topic in the automobile industry. They are a fascinating piece of technology that is solely aimed at making the lives of people easier. The need to multitask opens countless avenues for smart cars to enter the market and dominate it essentially rendering manually driven cars obsolete. However, despite the overwhelming benefits of smart cars, there are certain issues that relate to this technology in terms of operation and safety. My paper aims to highlight some of the serious issues relating to smart cars and their difficulty in overcoming obstacles as well as investigate potential solutions to mitigate those issues. Obstacle avoidance is key in smart car technology because smart cars are the first leap toward driverless cars and thus require the confidence of the people who drive it. So far, the technology has gone ahead leaps and bounds and an average smart car is quite trustworthy but not in all circumstances. There exist certain unique scenarios where smart cars just do not possess the requirements to respond adequately in due time.

1.1 History

As the technology matures, personal and public transportation will be forever transformed. First experiment conducted in the USA, 1920. Carnegie Mellon researcher first published a research paper for Autonomous vehicle using the Neural Network. General Motors first built self-driving car in the year 1939 and fuel was electricity. Name of the crater was Norman Bel Geddes. Vehicle guided by radio- controlled electromagnetic fields generated with magnetized metal spikes embedded in the roadway. By the 1960s, enthusiasts of artificial intelligence (AI) on computers began dreaming of cars smart enough to navigate ordinary streets on their own. By 1958, General Motors had made their vision into a reality. The car's front end was embedded with sensors called pick-up coils that could detect the current flowing through a wire embedded in the road. The current could be manipulated to tell the vehicle to move the steering wheel left or right. Japanese people followed this work but they used a camera system that relayed data to a computer to process images of the road in 1977. However, this vehicle could only travel at speeds below 20 mph. Contribution of the Germans in the automobile industry is noticeable and here again they proved, a decade later in the form of the VaMoRs, a vehicle outfitted with cameras that could drive itself safely at 56 mph. As technology improved, so did self-driving vehicles' ability to detect and react to their environment. In 2002, DARPA announced the Urban Grand Challenge, a $1 million prize if a vehicle is able to navigate 142 miles through the Mojave Desert. In the year 2004, competition won by the Stanford team, under Sebastian Thrun observation. He became project leader in Google's Waymo project in 2009. For an automobile to be autonomous, it needs to be continuously aware of its surroundings—first, by perceiving (identifying and classifying information) and then acting on the information through the autonomous/computer control of the vehicle. Autonomous vehicles require safe, secure, and highly responsive solutions which need to be able to make split-second decisions based on a detailed understanding of the driving environment. The National Highway Traffic Safety Administration categorized driving in 6 levels based on automation level. By 2013, many big companies including BMW, Audi, Cruse, Ford, and Mercedes Benz were all working on their own self driving vehicle. There were many autonomous car fatalities in the past, many people lost their lives in it. Tesla first launched autopilot mode in cars. Nowadays, millions of dollars are invested in these projects throughout the world. If our cars took their own wheels, we could text, tweet, and even play driving games as we lived the lifestyle only a few chauffeured executives can afford today. But beyond that, we would have to change the ways we think about driving and transport. Tesla autopilot-2015 is the most significant example of the semi-autonomous vehicle and it succeeds on the road. Autopilot mode enabled hands-free control for highway and freeway driving.

1.2Motivation

In a country like India, accident is a major problem which must be solved using new technologies rather than traditional ways.

Indian commuters travel approximately 35 km/day, out of that 75% commute using private vehicles. If the average speed is around 60 km/h then every person spends 30 minutes per day travelling. One crore people travel per day so people spend approximately 570 years in driving per day. This is too much time spent on just driving.

77.3% accidents caused because of Human error (2018). One-fourth of the Indian drivers use phones while driving which is dangerous for everybody (survey 2015).

1.3 Objectives

- Safety
- Save human time
- Follow Rules
- Better transportation service
- Less traffic
- To make life better

1.4 Project Modules

- LAN Detection
- Object detection
- Traffic sign detection and recognition
- Simulation
- Car Building
- Sensor fusion
- Path Planning
- Deployment on the Raspberry Pi

- **Softwares**

- Spyder
  Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It offers a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package. Beyond its many built-in features, its abilities can be extended even further via its plugin system and API. Furthermore, Spyder can also be used as a PyQt5 extension library, allowing developers to build upon its functionality and embed its components, such as the interactive console, in their own PyQt software. The reason behind using this IDE is variable explorer. Developers easily track variable values through the variable explorer window and find logical errors from the program. Name, Data type, Size and value of variable is mentioned in the window. File explorer is the next to the variable explorer and used to navigate files in the folders. Anyone can save the intermediate output of the program using I python. The I python console can be sent, exported and imported on different platforms easily. Profiler is used to find and eliminate bottlenecks to unchain your code's performance.

- VNC Viewer
  In computing, Virtual Network Computing (VNC) is a graphical desktop-sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical-screen updates back in the other direction, over a network. We use this to open Raspberry Pi Desktop View. VNC viewer can help to use remote machines using Public IP address. In cloud computing we can use instances using VNC software. Raspberry pi can be connected to a laptop using LAN or wireless. Wireless is a slow but better option because pi requires power supply and power supply may be at a distance. The common hotspot is the bottleneck in the connection. Problem of the VNC is that the IP address is required in connection and every time it gets changed.

- LanScan
  LanScan is a free, simple and efficient IPv4 network scanner that discovers all active devices on any subnet: the local one, or any public subnet that you configure. This software is only available for the MAC users. This software is used in Laptop and R-pi connection. LanScan identifies the IP address of the raspberry and that network address used in the VNC viewer.

- Etcher
  balenaEtcher (Commonly referred to as just 'Etcher') is a free and open-source utility used for writing image files such as .iso and .img files, as well as zipped folders onto storage media to create live SD cards and USB flash drives. Etcher is primarily used through a graphical user interface. Additionally, there is a command line interface available which is under active development. Future planned features include support for persistent storage allowing live SD card or USB flash drive to be used as a hard drive, as well as support for flashing multiple boot partitions to a single SD card or USB flash drive. [16] We have used Etcher for booting os for raspberry pi into memory card. First, format the SD card and then install etcher on the PC. After that, select the raspbian image and SD card in which you want to flash the OS. Finally, click on the “Flash” button and the flashing process will start. Etcher will show a progress bar and ETA while flashing the image. The process may take several minutes, depending on the size of the ISO file and the card speed.

- HyperLabel
  Remove ML project bottlenecks caused by time-consuming data labelling. HyperLabel is built for individual data scientists, as well as ML teams and even large enterprises. From desktop downloads, to

custom installations and managed services, HyperLabel gets you from zero-to- training easily, accurately, and at hyper speed. HyperLabel is used for creating annotated dataset. HyperLabel is used for creating annotated dataset. There's no need to upload image or video data to a HyperLabel cloud. With control via your desktop, simply import files from local drives or cloud storage. Individual developers, data scientists, and product owners, as well as large enterprise teams can flexibly manage projects of all sizes and complexity. Identify objects in image and video data with easy-to-select schema tools including rectangles, polygons, point, feature points, free text, select, and multi-select. A review panel makes it easier to manage labellers, track productivity, etc. Export formats include JSON, COCO, Pascal VOC, Create ML, and YOLO. We used this software for generating a dataset for doing transfer learning. We had to manually make bounding boxes and need to give it an annotation. Which generates a csv file which has annotation and bounding boxes also provides the same information in the json format and xml format. This software offers three different dataset formats.

- Google Colab

  Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with Zero configuration required Free access to GPUs Easy sharing. This platform gives free GPU and TPU. We trained yolo model here for traffic sign detection. This platform is very interactive for the user; developers can use unix commands such as "cd", "mkdir" and so on. These types of commands help to train models using cloud dataset. We stored a dataset on Google Drive and this drive was mounted in the Colab for training.

- Android Studio

  Android Studio is the IDE for android application. This software available in many different platforms such as Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development. Android Studio offers many features that enhance your productivity when building Android apps, such as: a flexible Gradle-based build system, a fast and feature-rich emulator, a unified environment where you can develop for all Android devices, apply Changes to push code and resource changes to your running app without restarting your app, code templates and GitHub integration to help you build common app features and import sample code, extensive testing tools and frameworks, lint tools to catch performance, usability, version compatibility, and other problems, C++ and NDK support, Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine. Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include: Android app modules, Library modules, Google App Engine modules. By default, Android Studio displays your project files in the Android project view. We prepared two applications in this software. First application captures a photo and sends it to the cloud for the object prediction and calculated results send back to the raspberry pi. This transmission takes place via the internet so no range restriction. Cheap cloud service changes public IP address every time when the instance is started so that application should be dynamic. To make applications in general the text field was used in the main page. Only one activity is there which runs continuously on the smartphone and sends images to the cloud. Text field take IPv4 from the user and set into a java socket program and make connections to the remote server. After the connection establishment camera sends images to the cloud. Second application is object detection using the tflite model. The Tflite model is specially made for the android application and it helps to solve many arduous problems in the android application. Tensorflow libraries are required in the app and those libraries download through gradle, just mention dependency in the file and gradle will do the rest. We can make applications in the eclipse also but Android studio is very handy software and has many more functionalities.

1.6Applications

- Street View for Google Maps
- Food delivery
- Transportation
- Goods delivery

1.7Advantages

- Safety; reduce accidents rate.
- Machines stick to the rules
- Speed - Save time
- Frictionless transport, no congestion
- New balances of public spaces; perhaps even road trains and intelligent routing to augment or replace public transport
- Improves health quality
- More Luxury

1.8Disadvantages

- Reduce job
- Expensive
- Hackers can change route
- Failure of sensor, vehicle can create chances of accident

Chapter 2 :
System Analysis and Design

System Analysis and Design

This project requires many prebuilt libraries and dedicated hardware for real time computation. All the used libraries and hardware's details mentioned in this section.

2.1 Implementation Requirements

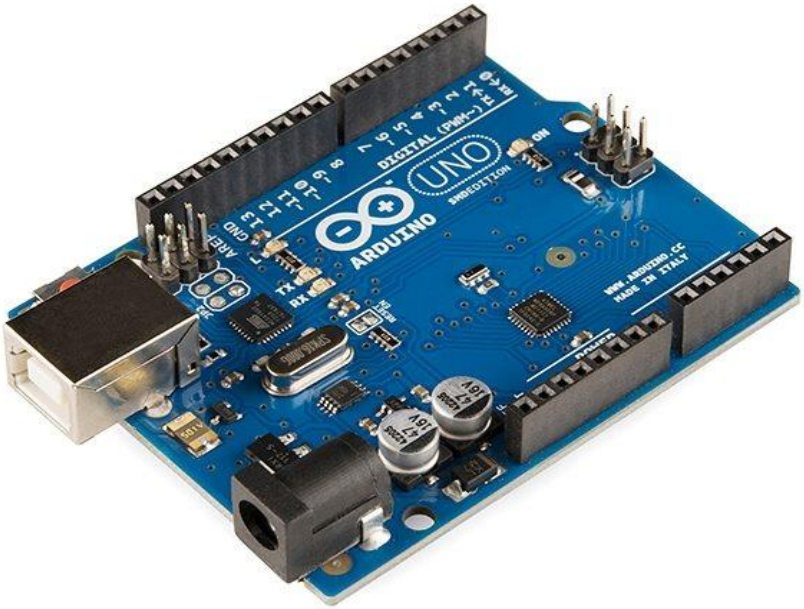
2.1.1 Hardwares

1) Raspberry Pi 3B+:

The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated, selling outside its target market for uses such as robotics. It now is widely used even in research projects, such as for weather monitoring because of its low cost and portability. It does not include peripherals (such as keyboards and mice) or cases.[12] 1.4GHz 64-bit quad-core processor, dual-band wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and Power-over-Ethernet support. The Raspberry Pi is a very cheap computer that runs Linux, and it provides a set of General Purpose Input/Output) pins that allow you to control electronic components for physical computing and explore the Internet of Things. Raspberry pi controls the car through the motor driver controller. Python scripts are deployed in the pi for the car driving. This device is small and easy to set up so it is perfect to install in the toy Car. This device plays a major role in this project because it is interfaced with the camera and each frame is sent to the server. Python is preinstalled in the OS itself, therefore image transmission can be done effectively and efficiently. The python scripts and detection model sent using VNC viewer.

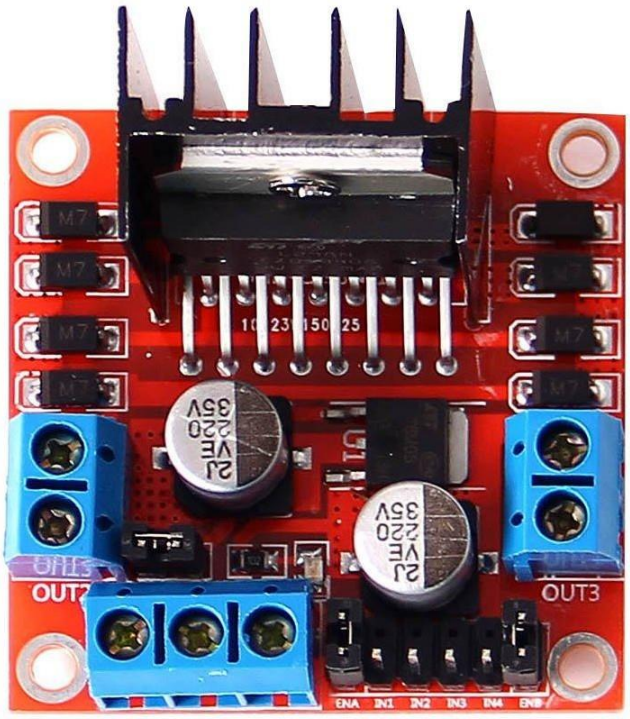
2) Arduino Uno :

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.



3) L298N Motor Driver:

The L298N Motor Driver module consists of an L298 Motor Driver IC, 78M05 Voltage Regulator, resistors, capacitor, Power LED, 5V jumper in an integrated circuit. There are 4 input pins and output pins available. This circuits requires 12V input from DC power supply. This motor driver controls car speed and direction according to raspberry pi's instruction.



5) **Camera:**

This is the eye of the car. Camera is the paramount part of the autonomous car. Computer vision is based on this device. Camera interfaced with the Raspberry pi and continuously captured images. These images feed to the neural network and identify objects and traffic signs in it.

6) **Dual Shaft BO Motor** :

The 100 RPM Dual Shaft BO Motor Plastic Gear Motor – BO series straight motor gives good torque and rpm at lower operating voltages, which is the biggest advantage of these motors. Small shaft with matching wheels gives an optimized design for your application or robot.

7) **Lithium Battery 2200mh:**

A lithium-ion battery is a type of rechargeable battery composed of cells in which lithium ions move from the negative electrode through an electrolyte to the positive electrode during discharge and back when charging. Li-ion cells use an intercalated lithium compound as the material at the positive electrode and typically graphite at the negative electrode. Li-ion batteries have a high energy density, no memory effect (other than LFP cells) and low self-discharge. Cells can be manufactured to prioritize either energy or power density. They can however be a safety hazard since they contain flammable electrolytes and if damaged or incorrectly charged can lead to explosions and fires.

Libraries

1) Python 3.6.4:

Python was not named for a snake even though we use the snake motif all the time. Python is the most popular programming language used by software engineers, analysts, data scientists, and machine learning engineers alike. This is an open source programming language with certain powerful libraries for designing neural networks and performing image processing.

2) Image AI 2.0.3 Library:

Image AI is a project developed by Moses Olafenwa and John Olafenwa , the Deep Quest AI team. Image AI is an easy to use Computer Vision Python library that empowers developers to easily integrate state-of-the-art Artificial Intelligence features into their new and existing applications and systems. Image AI provides an API to detect, locate and identify 80 most common objects in everyday life in a picture using pre-trained models that were trained on the COCO Dataset. The model implementations provided include Retina Net, YOLOv3 and TinyYOLOv3. Image AI provides very powerful yet easy to use classes to perform Image Recognition tasks. You can perform all of these state-of-the-art computer vision tasks with python code that ranges between just 5 lines to 12 lines

3) TensorFlow 2.0.0:

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. Easily train and deploy models in the cloud, on-pram, in the browser, or on-device no matter what language you use. A simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster. Machine learning researchers use the low-level APIs to create and explore

new machine learning algorithms. In this class, you will use a high-level API named tf. keras to define and train machine learning models and to make predictions. tf.keras is the TensorFlow variant of the open-source Keras API.

4) Pillow:

Pillow is a fork of PIL (Python Image Library), started and maintained by Alex Clark and Contributors. It was based on the PIL code, and then evolved to a better, modern and friendlier version of PIL. It adds support for opening, manipulating, and saving many different image file formats.

5) Keras:

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides. Built on top of TensorFlow 2.0, Keras is an industry-strength framework that can scale to large clusters of GPUs or an entire TPU pod. It's not only possible; it's easy.

6) Matplotlib:

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

7) PyGame :

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. This library is used in the car driving simulator. Udamy simulator requires this library.

8) Open-CV:

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license. This library is used in simulation and also in image transformation. Opencv displays images so that users can see all deleted objects, Lan and Traffic signs. Opencv also helped in non max suppression after the model detection.

9) Carla:

CARLA is an open-source simulator for autonomous driving research. CARLA has been developed from the ground up to support development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites and environmental conditions

10) Timeit:

This library is associated with time related functions in Python. In real time application time plays a very crucial role and it is the responsibility of the developer to check delay and computation time in the project. Default_timer function returns the current time at the start of execution and at the end of the program. Using those value delays can be computed and we can decide whether this model works or not.

11) OS:

OS is a very popular and frequent use library in the python environment. We can communicate with the os using this library. This library is used in many models for loading data from the current working directory. Path setting is difficult in the Colab but using this library there are only a few lines of code.

12) tkinter:

Tkinter is a GUI toolkit for python developers. This name comes from Tk Interface. Like swing in Java, tkinter supports all major concepts such as frame, panel. It provides various components and event listeners for event triggers. Windows, Mac OS, Linux support this library. This package is used in displaying images after applying detection.

13) Numpy:

Python language has Numpy for N-Dimension array processing. High level of mathematical operations can be performed using Numpy. Many functions are useful in array and matrix manipulation. Many mathematical operations are written very efficiently in this part of the bundle. It is very easy to use because of the high level of abstractions.

14) Socket:

This is a low level networking interface in python programming. This package is used to connect two nodes via the internet or LAN. Socket programming helps to send images from smartphones to servers using TCP protocol. This library also communicates with the android socket. Port number is also important in this data transmission. First step is to bind the IP Address and port number. Then listen from the other side.

15) base64:

This python library is generally useful for cryptanalysis. This is an encoding technique in data transmission. We encrypted the image using this technology and sent data to the cloud decrypted using the same.

16) Glob:

This is the UNIX path finding system. This python module is used in the Car Simulation. This function uses regular expressions to read files on the local storage.

17) Pandas:

Pandas is the most popular among Data Analysts. This package helps to load a dataset in the python program for the model training. In Udacity model training this library was used because all the steering value, throttle value, image names stored in the CSV file. This tool handles dataframe very efficiently and supports many complex operations.

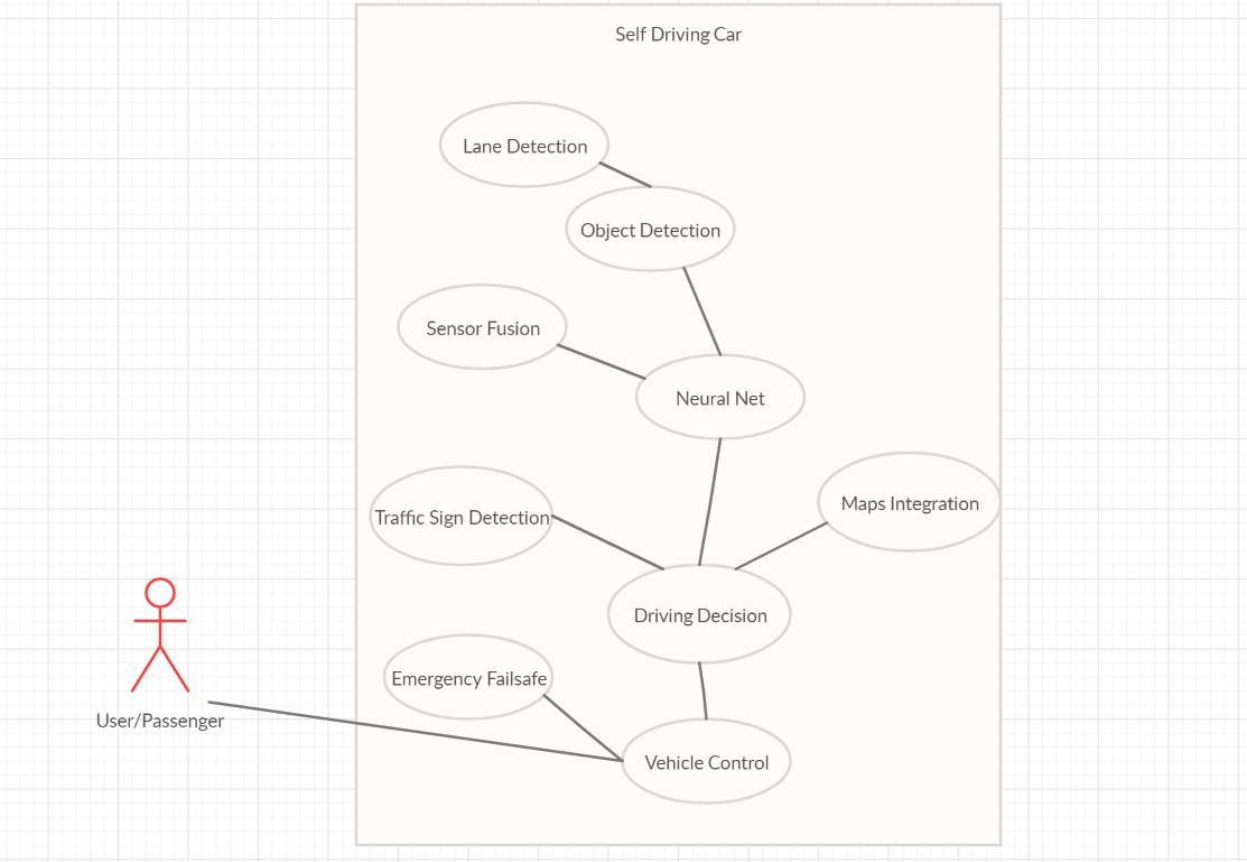
18) Sklearn:

Every machine learning engineer is familiar with this library. This is a free package for the python developers. Sklearn used in the splitting data for training and testing of model. This package also has many Regression, Classification, Clustering algorithms such as support vector machine, gradient boosting.

UML Diagram

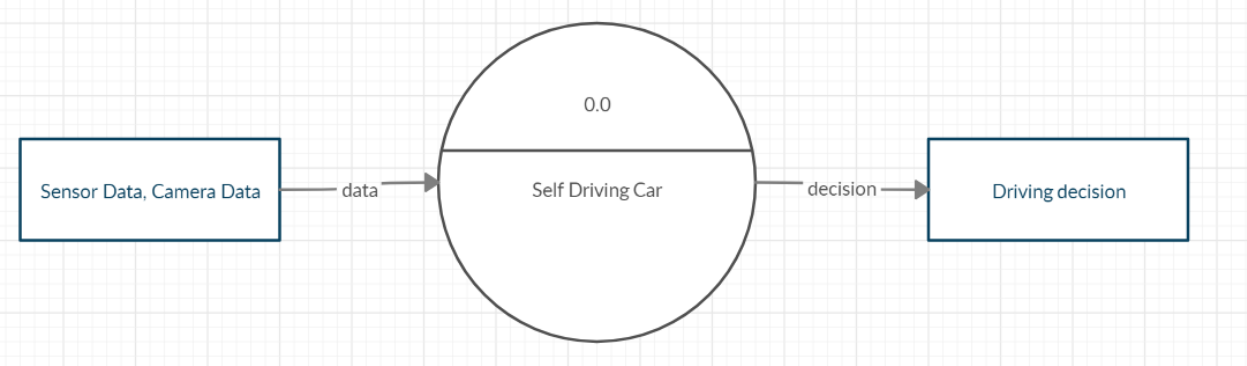
Unified Modeling Language is popular in the market because it is easy to understand. This is part of software engineering. Developer gets better idea about the system.

Use Case Diagram

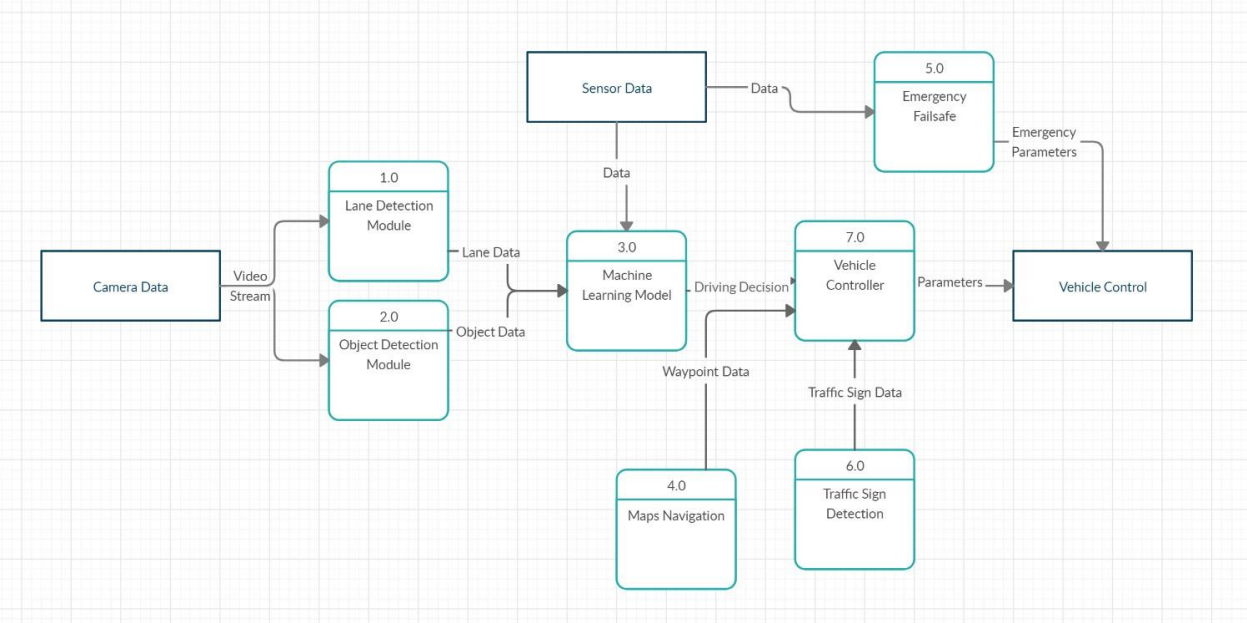


Data Flow Diagram

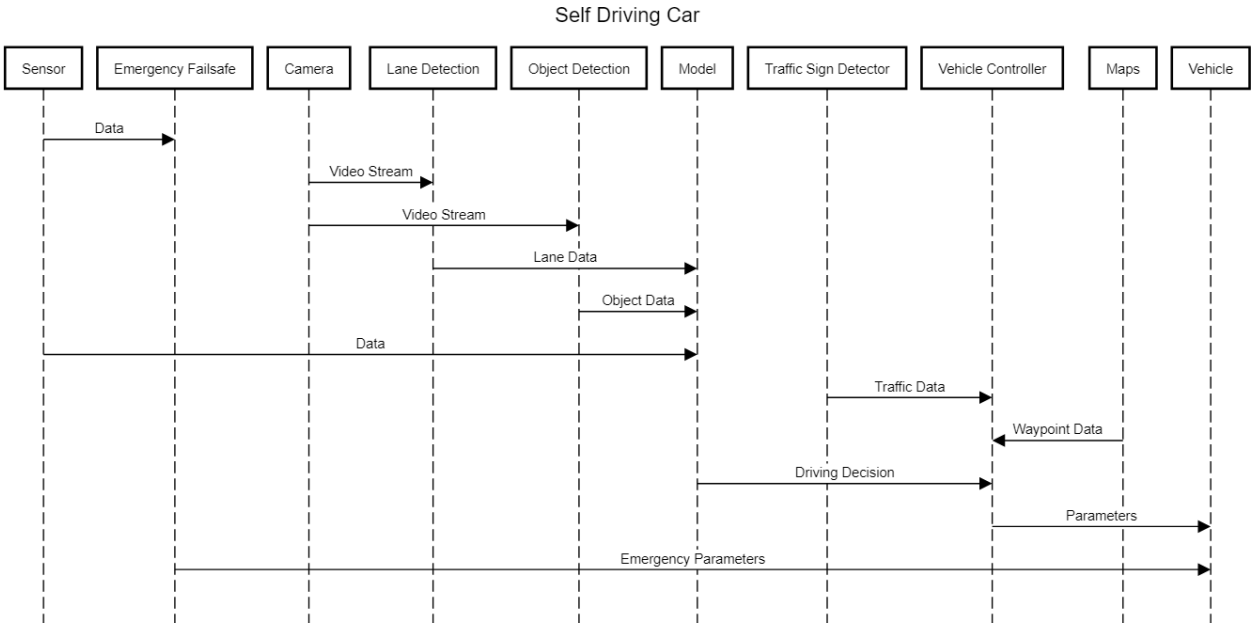
Context Level (level-0) DFD



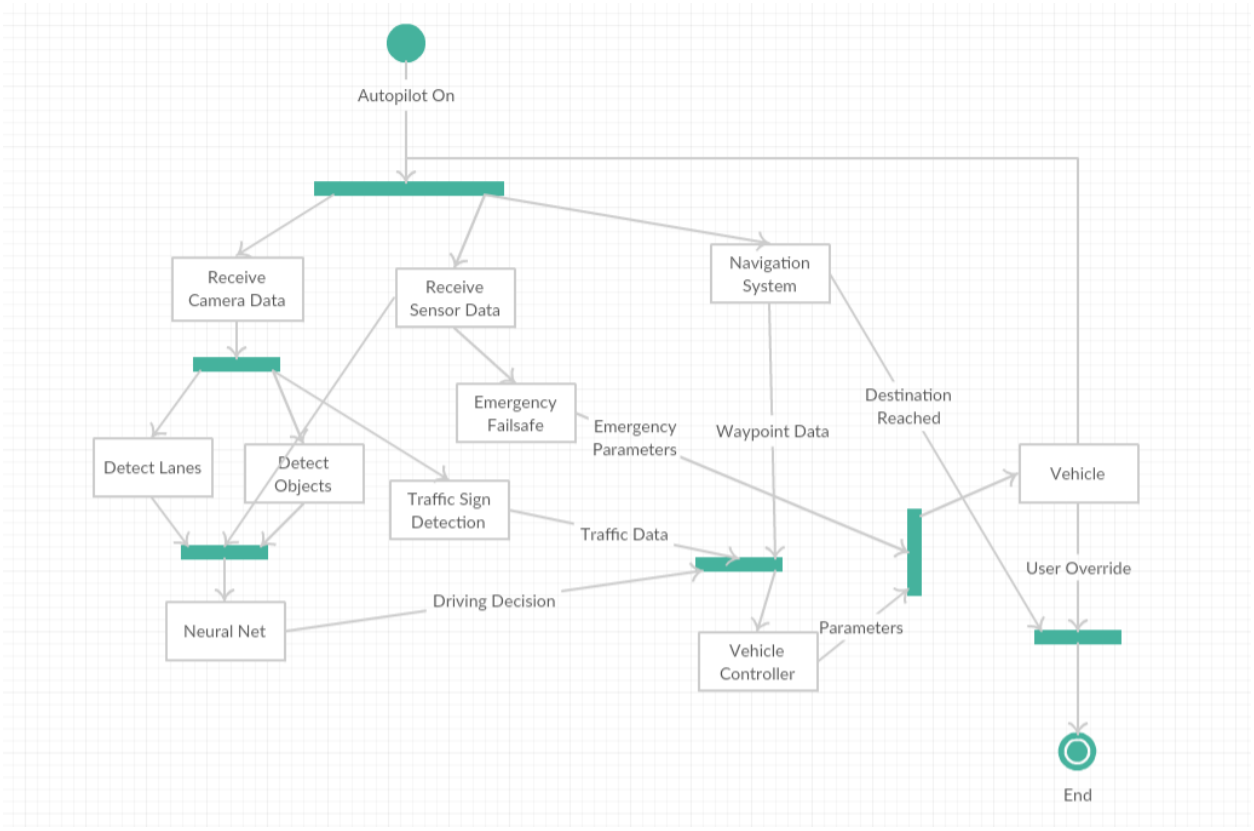
Level-1 DFD



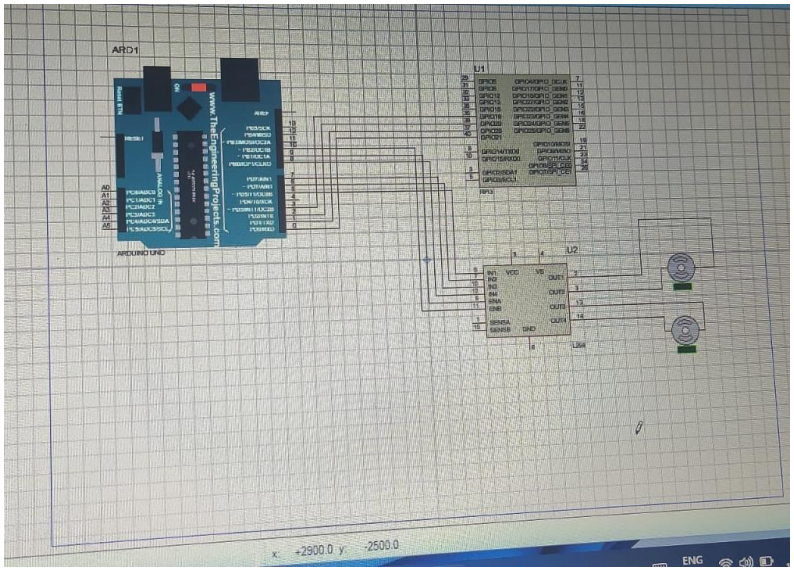
Sequence Diagram



State Diagram



CIRCUIT DIAGRAM



Chapter 4 :
Hardware based Implementation

Hardware based Implementation

Car Design

Initially a remote-control car was tried in this project but limited range and less control changed the plan. We assembled a car component and made a car using a motor driver. Ultrasonic sensor mounted on the top of front wheels.

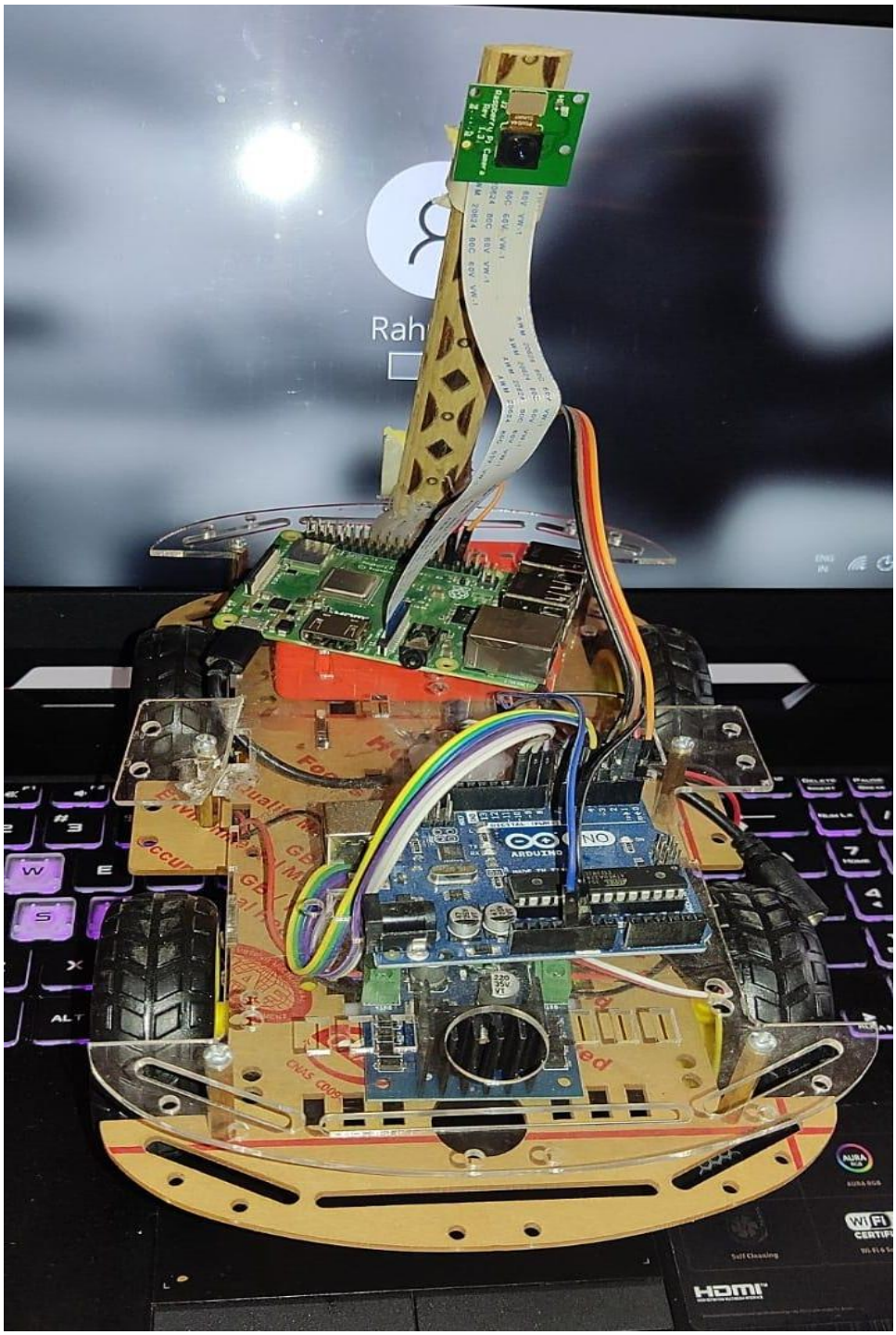
Parts of Car

- L298 Motor Driver
- 12V Battery
- Raspberry Pi 3B+
- 2 300 rpm gear motors
- 2 150 rpm gear motors
- 4 Wheels
- pi Camera
- Selfie Stick (mounting camera)
- Arduino uno
- Jumper wires

Raspberry Pi Interfacing:

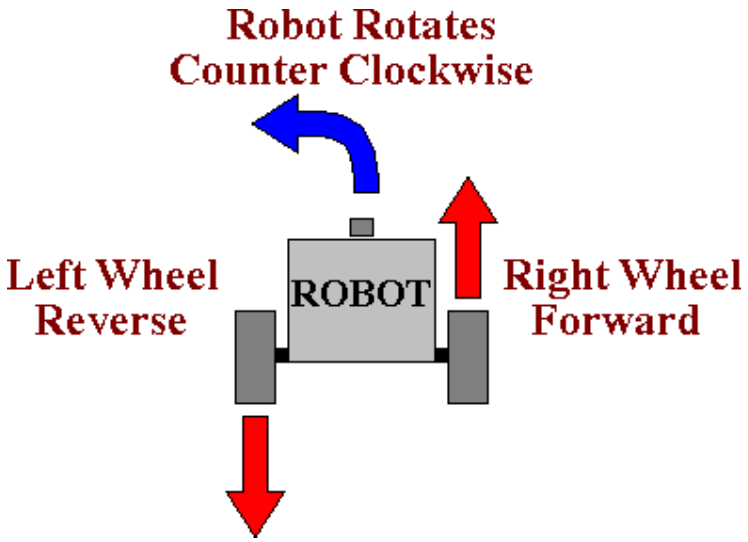
Raspberry pi 3B+ is used in track monitoring while car driving. I have worked with this hardware, so I know how to deal with it. Firstly, I downloaded Raspbian Buster with desktop and etched it into the USB. Raspberry pi takes more time to boot from USB compared to SD card, but USB provides more storage space. This OS does not have a VNC server preinstalled, so I installed and enabled it manually. After this camera module activated and tested on raspberry pi. All images required to examine for object detection and lane detection. These actions need good computation power, so I sent the captured photos to the laptop wirelessly. For transferring photos, the VLC library was used, and this library sends images using HTTP. All streamed images have been received by the VLC software at the remote desktop. This process had more lag in streaming, so I implemented another method. Motion library provides the same facilities with high frame rate (100) which means less legging. We deployed the model into a raspberry pi3 B+ module with 32GB memory card which has 1 GB RAM. But we didn’t get expected outcomes. Due to TensorFlow RAM required is more than 1GB so our process got killed. And we weren't able to use pi3 for implementation. So, we thought of implementing it on Lan Server.

Car Photo



L298N Motor Driver

The differential drive is a two-wheeled drive system with independent actuators for each wheel. The name refers to the fact that the motion vector of the robot is the sum of the independent wheel motions. The drive wheels are usually placed on each side of the robot and toward the front.



Two motors at both sides are connected in parallel, so motor controller consider it as single unit. The Difference between voltages supplied to motors at both sides makes car turn to a certain radius either right or left.

Controlling Car Driving

There are following control pins on L298N:

- Speed Control- ENA, ENB
- Direction Control- IN1, IN2, IN3, IN4

In order to have a complete control over DC motor, we have to control its speed and rotation direction. This can be achieved by combining these two techniques.

PWM – For controlling speed
H-Bridge – For controlling rotation direction

PWM – For controlling speed

The speed of a DC motor can be controlled by varying its input voltage. A common technique for doing this is to use PWM (Pulse Width Modulation)

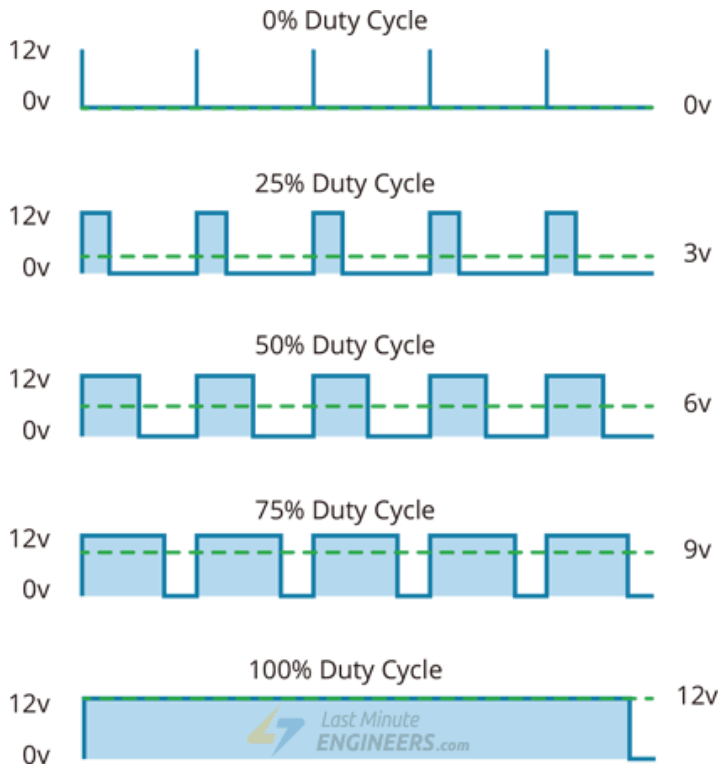
PWM is a technique where average value of the input voltage is adjusted by sending a series of ON- OFF pulses.

Our motor driver's ENA and ENB port is connected via RPi's PWM port which can be used to control speeds using driver program.

The average voltage is proportional to the width of the pulses known as Duty Cycle.

The higher the duty cycle, the greater the average voltage being applied to the dc motor (High Speed) and the lower the duty cycle, the less the average voltage being applied to the dc motor (Low Speed).

Below image illustrates PWM technique with various duty cycles and average voltages.



For controlling rotation direction

The DC motor's spinning direction can be controlled by changing polarity of its input voltage. We can simply change polarity by output ports Raspberry Pi which is connected IN port of motor driver.

Detection System

Detection models is too big for cheap computing units so we had to cut it down for better performance. There are two ways; first one is train model using less layers and another is pruning model. Here we had first pruned the model and then trimmed in following way:

Lane Finding

Identifying lane lines on the road is a pretty common task performed by all human drivers to ensure their vehicles are inside lane constraints when driving, so as to make sure smoother traffic and to avoid chances of collisions with other objects due to lane misalignment.

Similarly, it is a most crucial task for an autonomous vehicle to perform. Eventually recognizing lane markings on roads is possible using state of the art computer vision techniques.

The Pipeline

In this section we will discuss each step of our pipeline in detail.

Converting RGB image into HSL Image

HSL colour space is defined as hue, saturation and lightness, it helps us to get whiteness of each pixel as it enables us to differentiate between day and night images. It also helps us to deal with shadows of objects on the lane lines.

Isolating White and Yellow from HSL Image

As you can see in this image lane lines are in white and yellow in colour, so we have to get rid of other colours in the image.

RGB Image

HSL Image

Chapter 5 : CODING

RASPBERRY PI CODING

```
#include <opencv2/opencv.hpp>
#include <raspicam_cv.h>
#include <iostream>
#include <chrono>
#include <ctime>
#include <wiringPi.h>

using namespace std;
using namespace cv;
using namespace raspicam;

Mat frame, Matrix, framePers, frameGray, frameThresh, frameEdge, frameFinal,
frameFinalDuplicate;
Mat ROILane;
int LeftLanePos,RightLanePos, frameCenter, laneCenter,Result;

RaspiCam_Cv Camera;

stringstream ss;


vector<int> histrogramlane;

Point2f Source[] = {Point2f(40,145),Point2f(360,145),Point2f(10,195),Point2f(390,195)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),Point2f(100,240), Point2f(280,240)};


 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
    Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,400 ) );
    Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );
    Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,50 ) );
    Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );
    Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,50 ) );
    Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
    Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,100));

}

void capture()
{
   Camera.grab();
   Camera.retrieve( frame);
   cvtColor(frame, frame, COLOR_BGR2RGB);
```

```
}

void Perspective()
{
  line(frame,Source[0], Source[1], Scalar(0,0,255), 2);
  line(frame,Source[1], Source[3], Scalar(0,0,255), 2);
  line(frame,Source[3], Source[2], Scalar(0,0,255), 2);
  line(frame,Source[2], Source[0], Scalar(0,0,255), 2);


  Matrix = getPerspectiveTransform(Source, Destination);
  warpPerspective(frame, framePers, Matrix, Size(360,240));
}

void Threshold()
{
   cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
   inRange(frameGray, 200, 255, frameThresh);
   Canny(frameGray,frameEdge, 900, 900, 3, false);
   add(frameThresh, frameEdge, frameFinal);
   cvtColor(frameFinal, frameEdge, COLOR_GRAY2RGB);
   cvtColor(frameFinal, frameFinalDuplicate,COLOR_RGB2BGR);  //used in histrogram
function only

}

void Histrogram()
{
   histrogramlane.resize(400);
   histrogramlane.clear();

   for(int i=0; i<400; i++)      //frame,size(),width =400
   {
      ROILane = frameFinalDuplicate(Rect(i,140,1,100));
      divide(255, ROILane, ROILane);
      histrogramlane.push_back((int)(sum(ROILane)[0]));
   }
}

void LaneFinder()
{
   vector<int>:: iterator LeftPtr;
   LeftPtr = max_element(histrogramLane.begin(), histrogramLane.begin() +150);
   LeftLanePos = distance(histrogramLane.begin(), LeftPtr);

   vector<int>:: iterator RightPtr;
```

```
   RightPtr = max_element(histrogramLane.begin(), +250, histrogramLane.end());
   RightLanePos = distance(histrogramLane.begin(), RightPtr);

   line(frameFinal, Point2f(LeftLanePos, 0), Point2f(LeftLanePos, 240), Scalar(0, 255,0), 2);
   line(frameFinal, Point2f(RightLanePos, 0), Point2f(RightLanePos, 240), Scalar(0, 255,0), 2);
}

void LaneCenter()
{
   laneCenter = (RightLanePos-LeftLanePos)/2 +LeftLanePos;
   frameCenter = 188;

   line(frameFinal, Point2f(laneCenter, 0), Point2f(laneCenter, 240), Scalar(0, 255,0), 3);
   line(frameFinal, Point2f(frameCenter, 0), Point2f(frameCenter, 240), Scalar(0, 255,0), 3);

   Result = laneCenter-frameCenter;
}


int main(int argc,char **argv)
{

        wiringPiSetup();
   pinMode(21, OUTPUT);
   pinMode(22, OUTPUT);
   pinMode(23, OUTPUT);
   pinMode(24, OUTPUT);

        Setup(argc, argv, Camera);
        cout<<"Connecting to camera"<<endl;
        if (!Camera.open())
        {

        cout<<"Failed to Connect"<<endl;
    }

    cout<<"Camera Id = "<<Camera.getId()<<endl;


   while(1)
   {
   auto start = std::chrono::system_clock::now();

   capture();
   Perspective();
   Threshold();
```

```
Histrogram();
LaneFinder();
LaneCenter();

if (Result ==0)
{
   digitalWrite(21, 0);
   digitalWrite(21, 0);   //decimal = 0
   digitalWrite(21, 0);
   digitalWrite(21, 0);
   cout<<"Forward"<<endl;
}


else if (Result >0 && Result <10)
{
   digitalWrite(21, 1);
   digitalWrite(22, 0);  //decimal = 1
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout<<"Right1"<<endl;
}

else if (Result >=10 && Result <20)
{
   digitalWrite(21, 0);
   digitalWrite(22, 1);  //decimal = 2
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout<<"Right1"<<endl;
}

else if (Result >20)
{
   digitalWrite(21, 1);
   digitalWrite(22, 1);  //decimal = 3
   digitalWrite(23, 0);
   digitalWrite(24, 0);
   cout<<"Right1"<<endl;
}

else if (Result <0 && Result >-10)
{
   digitalWrite(21, 0);
   digitalWrite(22, 0);  //decimal = 4
   digitalWrite(23, 1);
```

```
   digitalWrite(24, 0);
   cout<<"Left1"<<endl;
}

else if (Result <=-10 && Result >-20)
{
   digitalWrite(21, 1);
   digitalWrite(22, 0);  //decimal = 5
   digitalWrite(23, 1);
   digitalWrite(24, 0);
   cout<<"Left2"<<endl;
}

else if (Result <-20)
{
   digitalWrite(21, 0);
   digitalWrite(22, 1);  //decimal = 6
   digitalWrite(23, 1);
   digitalWrite(24, 0);
   cout<<"Left3"<<endl;
}




ss.str(" ");
ss.clear();
ss<<"Result ="<<Result;
putText(frame, ss.str(), Point2f(1,50), 0,1, Scalar(0,0,255), 2);

namedWindow("orignal", WINDOW_KEEPRATIO);
moveWindow("orignal", 0, 100);
resizeWindow("orignal",640, 480);
imshow("orignal",frame);

namedWindow("Perspective", WINDOW_KEEPRATIO);
moveWindow("Perspective", 640, 100);
resizeWindow("Perspective",640, 480);
imshow("Perspective",frame);

namedWindow("Final", WINDOW_KEEPRATIO);
moveWindow("Final", 1280, 100);
resizeWindow("Final",640, 480);
imshow("Final",frame);
```

```
    waitKey(1);
    auto end = std::chrono::system_clock::now();
    std::chrono::duration<double> elapsed_seconds = end-start;

    float t = elapsed_seconds.count();
    int FPS = 1/t;
    cout<<"FPS = "<<FPS<<endl;

  }


    return 0;

}
```

ARDUINO UNO CODING

```
const int EnableL= 5;
const int HighL = 6;     //Left side Motors
const int LowL = 7;

const int EnableR= 10;
const int HighR = 8;     //Right side Motors
const int LowR = 9;

const int D0 = 0;       //Raspberry pin 21  LSB
const int D1 = 1;       //Raspberry pin 22
const int D2 = 2;       //Raspberry pin 23
const int D3 = 3;       //Raspberry pin 24  MSB

int a,b,c,d,data;

void Data()
{
  a = digitalRead(D0);
  b = digitalRead(D1);
  c = digitalRead(D2);
  d= digitalRead(D3);

  data = 8*d+4*c+2*b+a;
}


void setup()
{
 pinMode(EnableL, OUTPUT);
 pinMode(HighL, OUTPUT);
 pinMode(LowL, OUTPUT);

 pinMode(EnableR, OUTPUT);
 pinMode(HighR, OUTPUT);
 pinMode(LowR, OUTPUT);

 pinMode(D0, INPUT_PULLUP);
 pinMode(D1, INPUT_PULLUP);
 pinMode(D2, INPUT_PULLUP);
 pinMode(D3, INPUT_PULLUP);

}

void Forward()
```

```
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);
}


void Backward()
{
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnableR, 255);

}

void Stop()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 0);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 0);
}

void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 200);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);

}
```

```
void Left2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 160);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);

}

void Left3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 100);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 255);

}

void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 200);

}

void Right2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 100);
```

```
}

void Right3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL,HIGH);
  analogWrite(EnableL, 255);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR, 100);

}


void loop()
{
  Data();
  if(data==0)
   {
     Forward();
   }

  else if(data==1)
   {
    Right1();
   }

  else if(data==2)
   {
    Right2();
   }

  else if(data==3)
   {
    Right3();
   }

  else if(data==4)
   {
    Left1();
   }

  else if(data==5)
   {
    Left2();
```

```
  }

 else if(data==6)
  {
   Left3();
  }

  else if (data>6)
  {
   Stop();
  }


}
```

Result Analysis

Currently on the software side (simulation), almost all modules are implemented. Foreign object avoidance and lane following are working. The car will follow the lane lines and avoid any other vehicles / pedestrians and any other obstacles. It will also slow down on junctions, and overtake other vehicles, and execute emergency stops when other options are not available. On the hardware side, some issues are occurring due to insufficient compute power. The car is following lanes but some issues are occurring due to limitations of the hardware. The car will execute an emergency stop when the sensor detects any obstacles using the sensor. The backbone of this project is the LAN server because raspberry pi has insufficient computation power. Car drives automatically very well after training using a high-resolution dataset. Car stops whenever it finds a STOP sign or RED signal and if the sensor detects any object it will break and avoid collision.

Conclusion and Future Work

Conclusion:

This system is work perfect in the simulated environment like Carla but face some difficulties in the new environment. All the sensor work according to requirement and stop where it needed. This system will forcefully stop at the some unfamiliar situations like image resolutions is not cleared or person on the way. The Car fatalities will not occurred because of working algorithm. Path prediction is done by Kalman Filter and based on that further path is decided. Light and brightness of the surrounding is affect driving because of Camera. So to get good performance high quality Camera is required. Computation power is also limited therefore delay can easily noticed. This can be solved using powerful mobile computation devices such as Jetson, but it is costly.

Follow-up Activities :

- Improve vehicle hardware
- Improve Vehicle Controller
- Deploy more sensors such as LIDAR, RADAR which enables us in tracking down objects precisely.
- Audio sounds
    - Car sound
    - Break sound
    - collision sound
    - Horn sound
- Predict move of Objects.
- Vehicle to vehicle communication using edge computation (M2M)

THANK YOU