Table of Contents

List of Figures

List of Tables

Abstract

Autonomous cars are the future smart cars anticipated to be driver less, efficient and crash avoiding the ideal urban car of the future. To reach this goal automakers have started working in this area to realise the potential and solve the challenges currently in this area to reach the expected outcome. In this regard the first challenge would be to customise and imbibe existing technology in conventional vehicles to translate them to a near expected autonomous car. This transition of conventional vehicles into an autonomous vehicle by adopting and implementing different upcoming technologies and demonstration of various Technological and Security aspects of a fully autonomous vehicle is the goal of this project . This includes the objectives of autonomous vehicles and their implementation difficulties. Part of this Project also touches upon the existing standards for the same and compares the introduction of autonomous vehicles in the Indian market in comparison to other markets. Thereafter the acceptance approach in Indian market scenarios is discussed for autonomous vehicles.

This Prototype is a fully Autonomous robot model representation of a Self Driving Car with real-time obstacle evasion and human detection, It can detect and react to the changes in its environment and can act accordingly by using a well trained sets of Artificial Intelligence softwares which can seamlessly coordinate with each other and can react faster than human response time, this car will include AI algorithms from car usage behaviours to provide accurate and maximum battery performance.

The prototype will analyse the markings, signs and dimensions of the track to securely manoeuvre around an obstacle while following Road and Traffic rules and regulations closely, by just using camera and computer vision algorithms.

1. Introduction

A self-driving car (also known as an autonomous car or a driverless car) has no human input and can sense surrounding without any human interactions. Rushing around, trying to get errands done, thinking about the things to be bought from the nearest grocery store has become a part of our daily schedule. Driver error is one of the most common cause of traffic accidents, and with cell phones, in- car entertainment systems, more traffic and more complicated road systems, it isn't likely to go away.

With the number of accidents increasing day by day, it has become important to take over the human errors and help the mankind. All of this could come to an end with self-driving cars which just need to know the destination and then let the passengers continue with their work. This will avoid not only accidents but also bring a self-relief for minor day to day driving activities for small items.

Preamble: *This project presents a miniature model of self driving car for the obstacle avoidance using camera for lane detection by openCV and Tensor-flow's object detection model. Building a self driving car which can follow the lane, avoid the different obstacles present in the path and can also follow road safety rule. With the increasing number of vehicles traffic jams are also increasing this problem can be handled easily. It can even help people who don't know driving at some kind of emergency situations.*

1.1 Necessity of Self driving car:

With the growing needs of convenience, technology now tries to seek automation in every aspect possible. Also, with the growth in the number of accident in the recent years due to increased number of vehicles and some amount of carelessness of the drivers, it now seems necessary to seek automation in vehicles as well. Hence to achieve the merit above mentioned problems, we present an autonomously drive car which would eradicate human intervention in the field of driving. The car would drive itself from one place to the other on its own it would possess integrated features like lane-detection, obstacle-detection and traffic

sign detection. This features would help the car drive itself to the mentioned destination on the track properly, avid collisions, provide live streaming of the view in front of it with the help of camera mounted over the car and detect traffic signs and obey them accordingly so as to avoid accidents caused due to disobeying the traffic rules. This would ensure safer, easier, updated and more convenient mobility, hence providing out to be a revolutionary step in the field towards automation and mobility.

The main focus of the proposed miniature model of self driving car is that it can follow the lane and will also avoid the different obstacles present in the path and to demonstrate safety aspects of a self driving car compared to traditional electric vehicle. By making little bit of changes in the algorithms we can create a model which can even go to places where humans are not able to go or it is dangerous for humans to go. This can help us in doing surveys by collecting the data which we receive.

1.2. Significance of the project:

Miniature model is designed for path detection and object detection for autonomous purposes. The main significance of the proposed model is:

- By making little bit of changes in the algorithms we can create a model which can even go to places where humans are not able to go or it is dangerous for humans to go. This can help us in doing surveys by collecting the data which we receive.
- Building a self driving car which can follow the lane and will also avoid the different obstacles present in the path.With the increasing number of vehicles traffic jams are also increasing this problem can be handled easily. It can even help people who don’t know driving at some kind of emergency situations.
- Building a prototype that can follow current road safety guidelines and to also enlighten future traffic laws fine tuned for a self driving car future.

There is considerable uncertainty concerning autonomous vehicle development, benefits and costs, travel impacts, and consumer demand. Considerable progress is needed before autonomous vehicles can operate reliably in mixed urban traffic, heavy rain and snow, unpaved and unmapped roads, and where wireless access is unreliable. Years of testing and regulatory

approval will be required before they are commercially available in most jurisdictions. The first commercially available autonomous vehicles are likely to be expensive and limited in performance. They will introduce new costs and risks. These constraints will limit sales. Many motorists will be reluctant to pay thousands of extra dollars for vehicles that will sometimes be unable to reach a destination due to inclement weather or unmapped roads.

1.3 Literature Survey

Below is the tabulated study of papers or research available on this domain. The results are as follows:-

Sr. No.	Title	Authors	Accuracy	Technique used	Year of Publishing
1	Autonomous car using Raspberry PI and ML	Yasir A, Aiman Salim, Arya Dileep, Anjana S	76.25%	Machine Learning (Neural Networks)	2019
2	Obstacle avoidance with Ultrasonic sensor	Johann Borenstein, Yoran Koren	Not Mentioned	Open CV and python	2020
3	Design and Implementation of self driving car using Raspberry Pi		81.5%	Open CV and Python	2020
4	Autonomous car using Raspberry PI Model	Nihal A Shetty,Mohan K,Kaushik K	Not Mentioned	Artificial Neural Networks ,Open CV and Python	2020

TABLE- 1 LITERATURE SURVEY

2. Aim and Objective

2.1 Title

AI Self Driving Car

2.1. Aim

To develop a miniature prototype of self driving car that can detect lanes and avoid abstacles by using camera and computer vision.

2.2 Objective

Objective of this project is to demonstrate the working of a self driving car, which uses nothing but camera as input to guide through traffic and drive effortlessly on road while following all the road safety measure and rules.

2.3 Scope

Scope of this project is spread in various domains still all the goals converge to open specific goal of this project that is to make a car that can drive by itself, detect lanes of the road ,respond to various types of obstacles and follow traffic rules.by just by using a camera, to demonstrate the safety and also to talk about the future of driverless cars in various domains such as delivery, transport, terrain surveying etc. The scope of this project extends beyond this prototype, this is the power of computers this same model can be scaled to real self driving cars and it will still show promising results.

This report is divided in various phases each phase explaining software, hardware, construction, testing, calibration and demonstration in rich detail.

2.4. Methodology/Approach to attain each objective

- From the above block diagram , A light weight metal structure is used to build the base of the car model. The metal structure also acts as a platform for all the components such as microprocessor, battery, motor driver, and camera used in the car.

- Camera sensor takes the input image & perform the lane and object detection.

- Raspberry Pi 4B is used as the microprocessor. It performs all the AI and computer vision operations by using pertained Tensorflow object detection model, and openCV library for lane detection.
- Arduino Uno is connected as slave drive to RaspberryPi, It performs operations such as setting the speed for the motor and supplying power to the motor.
- Then, Motors receive the input from microprocessor and decides the rpm in accordance to that.
- After all the steps, car will move along the path.

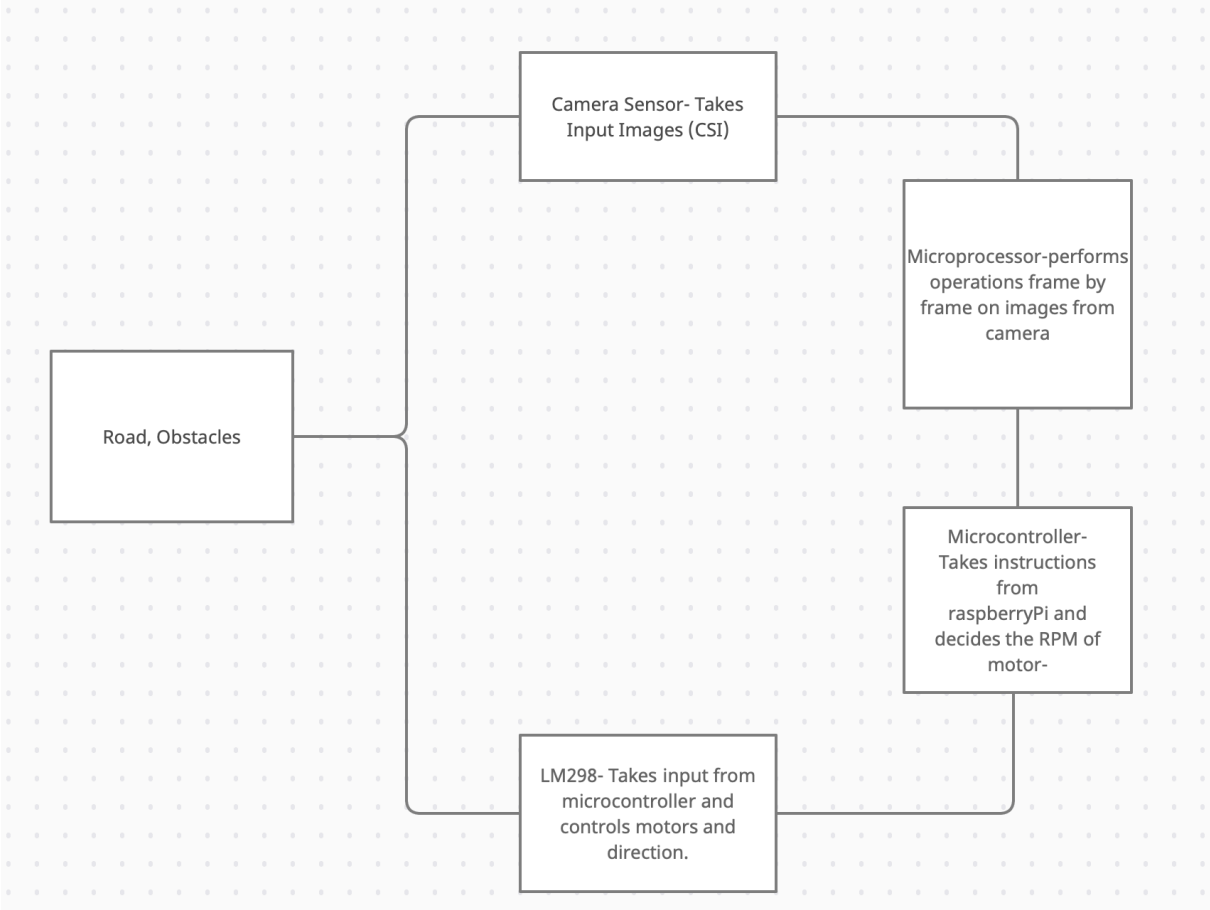


FIGURE 1 BLOCK DIAGRAM

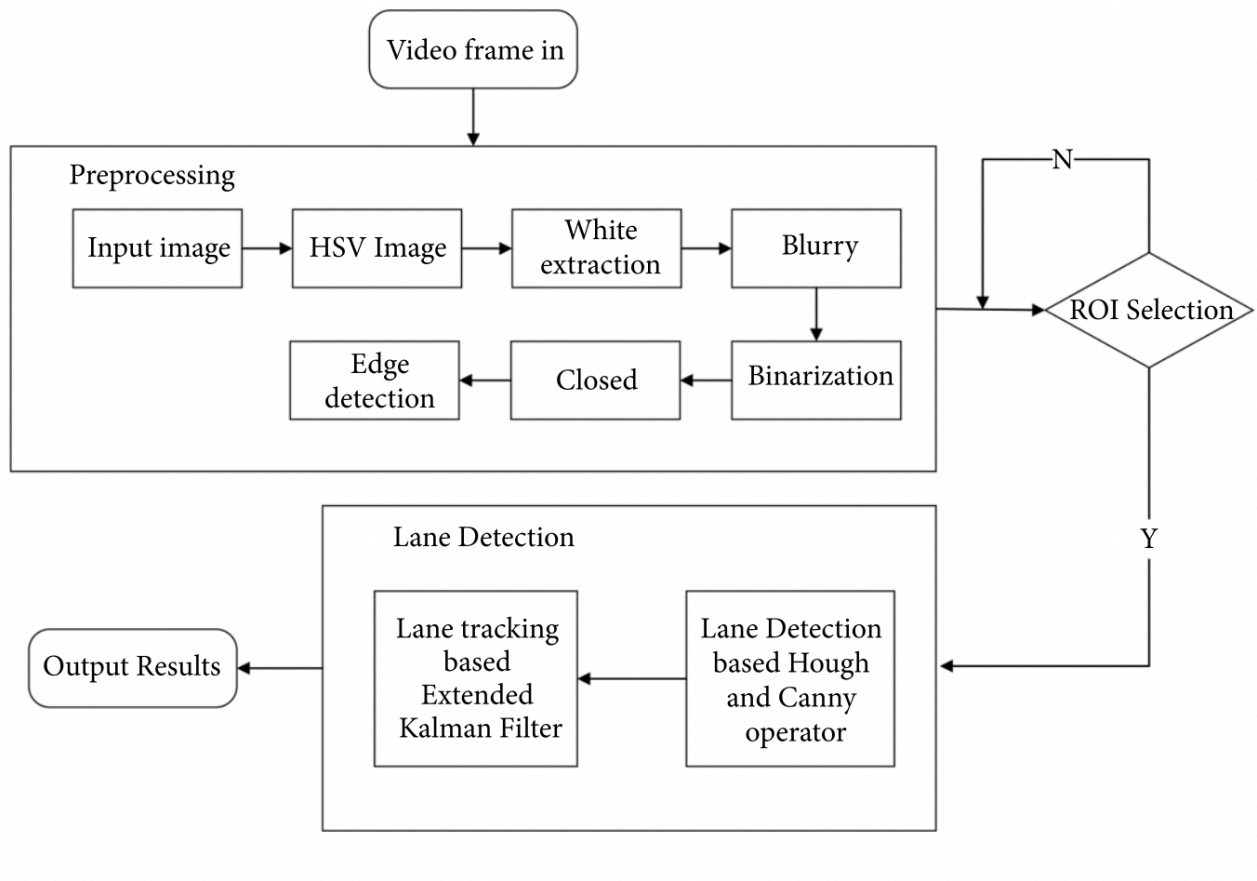


FIGURE 2 IMAGE DETECTION FLOW CHART

3. Background Theory

Our miniature model of self driving car is driven by using RaspberryPi, Arduino UNO, Raspberry Camera module, motor driver, motors.Here we will not discuss the hardware and software used in brief just will get to know the purpose or overview of all. We will discuss this more in development part.

3.1 Use of RaspberryPi 4b:

RaspberryPi is used as a microprocessor to handle complex AI and image/video processing algorithms it is the heart of this prototype, it uses linux kernel based on a Debian fork of Linux also known as raspberryPi OS. It is not the latest version as RaspberryPi camera module is no longer supported on latest release of the os.

3.2 Use of AI accelerator:

The Coral USB Accelerator adds an Edge TPU coprocessor to RaspberryPi. It includes a USB-C socket that can connect to a RaspberryPi to perform accelerated ML inferencing. The on-board Edge TPU is a small ASIC designed by Google that accelerates TensorFlow Lite models in a power efficient manner: it's capable of performing 4 trillion operations per second (4 TOPS), using 2 watts of power-that's 2 TOPS per watt. For example, one Edge TPU can execute state-of-the-art mobile vision models such as MobileNet v2 at almost 400 frames per second. This on-device ML processing reduces latency, increases data privacy, and removes the need for a constant internet connection.

ML accelerator	Google Edge TPU coprocessor: 4 TOPS (int8); 2 TOPS per watt
Connector	USB 3.0 Type-C* (data/power)
Dimensions	65 mm x 30 mm
Availability	Out Of Stock in India

TABLE- 2 TECHNICAL SPECIFICATIONS OF EDGE TPU

3.3 Use of Raspberry Camera module:

Raspberry Pi camera module is used to get the image in which operations are performed. It is a semi-wide angle n camera with NoIR filter so it can be very effecting in low light computer vision algorithms.

3.4 Use of Power bank:

2 OC3 certified Power banks are used for suppling power to Raspberry pi, and motors.

3.5 Background on use of Software:

We have downloaded the Rasbian Buster operating system for Raspberry Pi in which we installed the all the necessary libraries.

- Python 3.0, C-lang are open source software which has been used for all the coding, interfacing, calibrating, controlling the sensors and motors.
- Geany editor is used to build executable applications using c++.
- Microprocessor takes all the decision based on our algorithm for the proper movement of the car.

3.6 Reasons for choosing L298 Motor Driver and standalone RaspberryPi 4b

L298 Motor Driver

The L298 is a popular dual-H bridge Motor Driver IC. As the name suggests it is mainly used to drive motors. A single L298 IC is capable of running two DC motors at the same time; also the direction of these two motors can be controlled independently. So if you have motors which has operating voltage less than 36V and operating current less than 2A, which are to be controlled by digital circuits like Op-Amp, 555 timers, digital gates or even Microprocessor like Raspberry Pi and even in Microcontrollers like Arduino, PIC, ARM etc.

RaspberryPi 4b

RaspberryPi 4b with 8GB ram is used along with Samsung's 32GB class 10 high speed SD-card as a boot drive. Unfortunately other options which are far more capable in handling such tasks are either not for sale in India or out of stock in every location. Even the google edge TPU is unavailable due to ongoing silicon shortage across the globe. Some notable consideration instead of RaspberryPi 4B are given below:

Sr. No	Name	Reason for consideration	Reason of unavailability
1	RaspberryPi 3b+	Low Cost and Low power microprocessor.	Unsatisfactory performance.
2	Jetson Nano	Very High ML Performance	Unreasonable Price
3	Coral Dev board	Very High ML Performance, Low power	Less to no support for hardware and software
4	USB accelerator	Low Cost and very high performance	Unavailablity

TABLE-3 N NOTABLE HARDWARE CONSIDERATIONS

4. Hardware

This section gives a detailed description of hardware used and involves in the designing and development of the self driving car

4.1 Chassis

A Four wheel drive transparent acrylic chassis of such dimensions and this weight

FIGURE-2 CHASSIS

4.2 Motors

The 4 wheels of the chassis are connected to 4 separate geared DC motor with 1:48 gear ratio and Operating voltage range is 3-6v is mounted on the cassis, Motors of each half are connected in parallel 100RPM at 5v

FIGURE-3 DC GEARED MOTOR

4.3 Motor Driver

Motor drivers acts as an interface between the motors and the control circuits. Motor require high amount of current whereas the controller circuit works on low current signals. So the function of motor drivers is to take a low-current control signal and then turn it into a higher-current signal that can drive a motor

L298N board is a dual H-Bridge motor driver which can help you interface motors which draw upto 2 Ampere of current such as Gear motors. The L298 is an integrated monolithic circuit in a 15 pin Multi-watt packages. It is a high voltage, high current dual full-bridge driver de signed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the in put signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage. The Dual H-Bridge is an electronic circuit that switches the polarity of a voltage applied to a load.

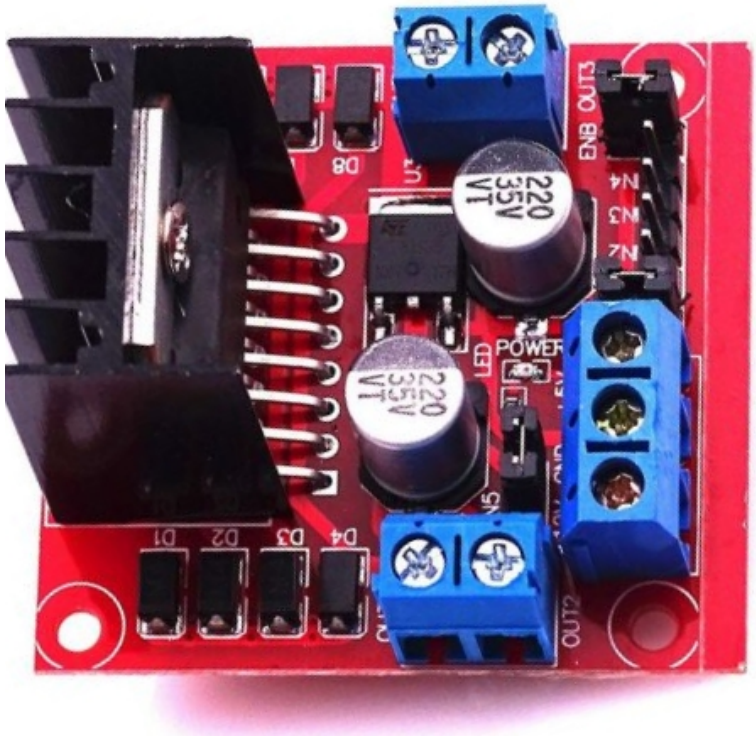
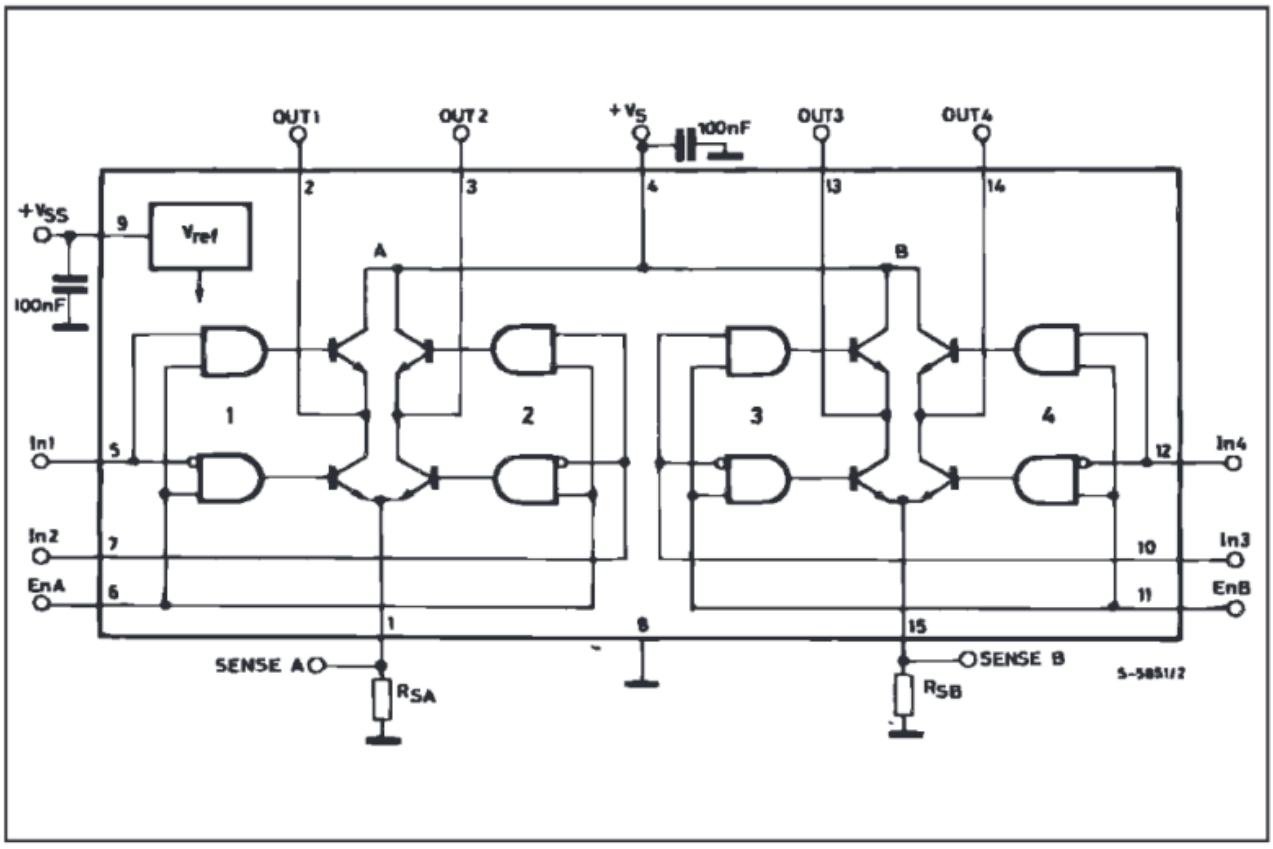


FIGURE- 4 L298 DUAL H-BRIDGE MOTOR CONTROLLER

The motor driver IC L298 is capable of driving 2 motors simultaneously. The rotation of the wheels is synchronized on the basis of the sides i.e. the left front and left back wheels rotate in sync and right front and right back- wheels rotate in sync. Thus the pair of motors on each side is given the same digital input from L298 at any moment. This helps the car in forward, backward movements when both side wheels rotate in same direction with same speed. The car turns when the left side wheels rotate in opposite direction to those in right.

The IC is fixed on the lower shelf with the help of two 0.5 inch screws. It is permanently connected to the motor wires and necessary jumper wires are drawn from L298 to connect to Arduino UNO, The left side motor connected to OUT 1 and OUT 2 and The right side motor connected to OUT 3 and OUT 4, ###**PIN TO ARDUINO###** are connected to the Digital IO pins of Arduino Uno via jumper wires.



Circuit Diagram of L298 Dual-H bridge motor controller

A H-bridge is an electronic circuit that switches the polarity of a voltage applied to a load. These circuits are often used in robotics and other applications to allow DC motors to run forwards or backwards.

ENA	IN1	IN2	Description
0	N/A	N/A	Motor is off
1	0	0	Motor is stopped (brakes)
1	0	1	Motor is on and turning backwards
1	1	0	Motor is on and turning forwards
1	1	1	Motor is stopped (brakes)

TABLE-4 LEFT MOTOR TRUTH TABLE

ENB	IN3	IN4	Description
0	N/A	N/A	Motor is off
1	0	0	Motor is stopped (brakes)
1	0	1	Motor is on and turning backwards
1	1	0	Motor is on and turning forwards
1	1	1	Motor is stopped (brakes)

TABLE-5 RIGHT MOTOR TRUTH TABLE

4.4 Arduino UNO R3

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/ output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller.

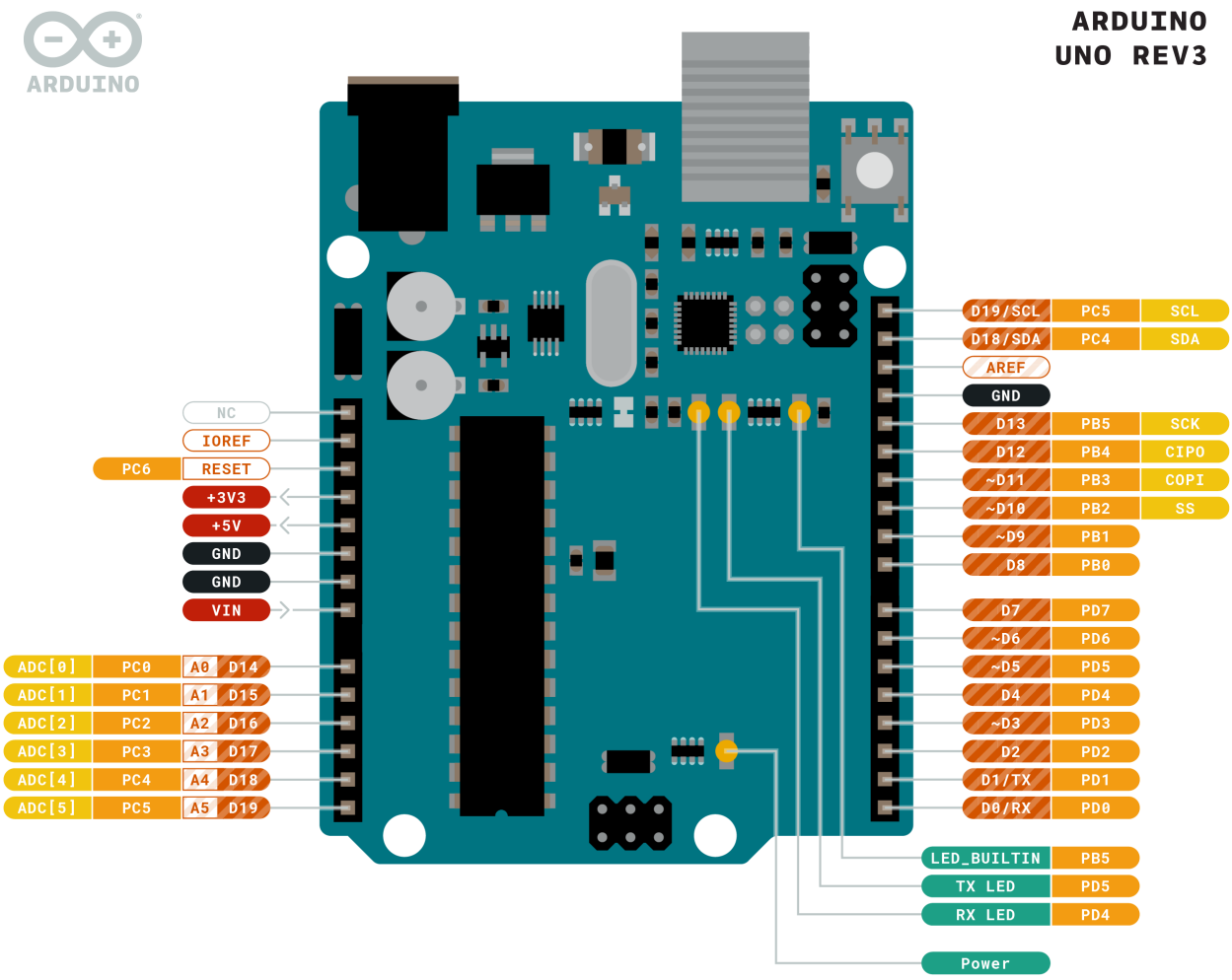


FIGURE -5 ARDUINO UNO PINOUT

The **ATmega328P** microcontroller have 32 KB Flash memory, 1KB EEPROM, 2 KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/ counters with compare modes, internal and external interrupts, serial programmable USART, a Two-Wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.

Due to presence of a voltage converter on Arduino UNO it can safely be powered with upto 12v by using Vcc pin of the board.

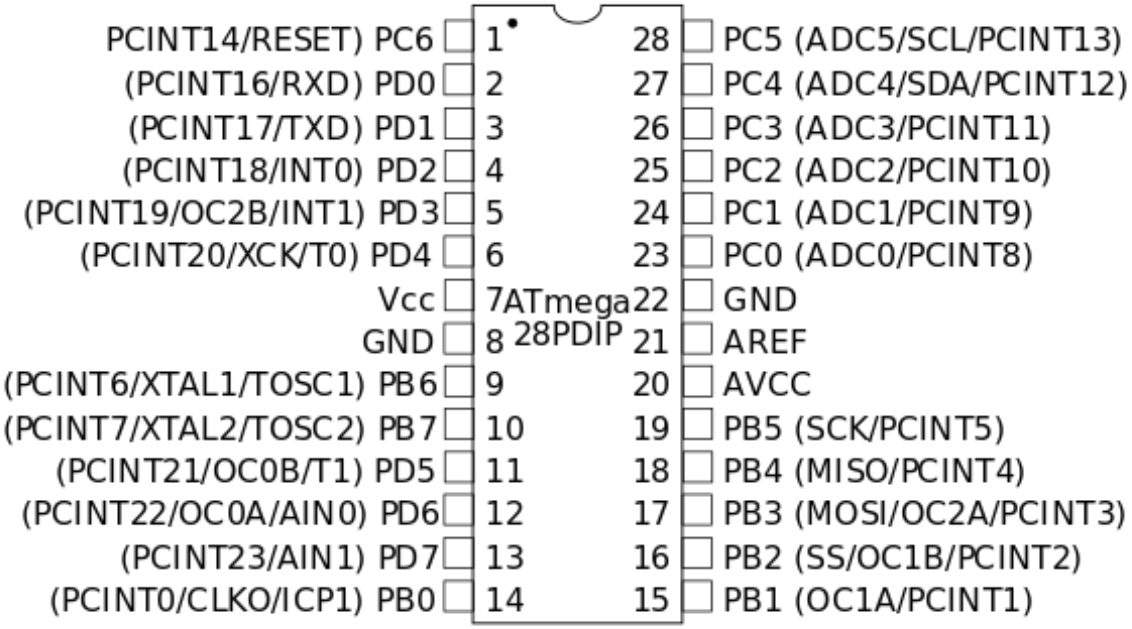


FIGURE -6 ATMEGA328P

It controls L298 motor driver by utilising its digital PWM (~) pins which helps control the speed of the motors easily.

It also controls the voltage of the power supply with pin 4

It acts as a slave device to master raspberry pi 4b over a 4bit parallel connection, It can also communicate serially over USB to RaspberryPi 4b.

4.5 Raspberry Pi

Raspberry Pi 4 Model-b 8GB, is a small single board computer. By connecting peripherals like Keyboard, mouse, display to the Raspberry Pi, it will act as a mini personal computer.

Raspberry Pi is popularly used for real time Image/Video Processing, IoT based applications and Robotics applications, although Raspberry Pi is slower than laptop or desktop but is still a computer which can provide all the expected features or abilities, at a low power consumption. Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide NOOBS OS for Raspberry Pi. We can install several Third-Party versions of OS like

Ubuntu, Arch-linux, RISC OS, Windows 10 IOT Core, etc. Raspbian OS is official Operating System available for free to use. This OS is efficiently optimised to use with Raspberry Pi. Raspbian have GUI which includes tools for Browsing, Python programming, office, games, etc. We should use SD card (minimum 8 GB recommended) to store the OS (operating System). Raspberry Pi is more than computer as it provides access to the on-chip hardware i.e. GPIOs for developing an application. By accessing GPIO, we can connect devices like LED, motors, sensors, etc and can control them too. It has ARM based Broadcom Processor SoC along with on-chip GPU (Graphics Processing Unit). It has ARM based Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz with *GB of LPDDR4 RAM.

FIGURE-7 RASPBERRYPI 4B

it is important to note that the performance of the new Raspberry Pi 4 Model B is equivalent to that of an entry-level x86 computer. Among the main features of this latest Raspberry Pi

computer, we can note: A high-performance 64-bit quad-core processor, Dual display support with resolutions up to 4K via a pair of micro-HDMI ports, Hardware video decoding up to 4Kp60sec. A connection to the dual-band wireless local area network 2.4/5.0 GHz. Bluetooth 5.0 / Gigabit Ethernet / USB 3.0 / PoE features (via a separate HAT PoE add-on module)

GPIO PIN

GPIO stands for General Purpose Input Output. The Raspberry Pi has two rows of GPIO pins, which are connections between the Raspberry Pi, and the real world. Output pins are like switches that the Raspberry Pi can turn on or off (like turning on/off a LED light). But it can also send a signal to another device.

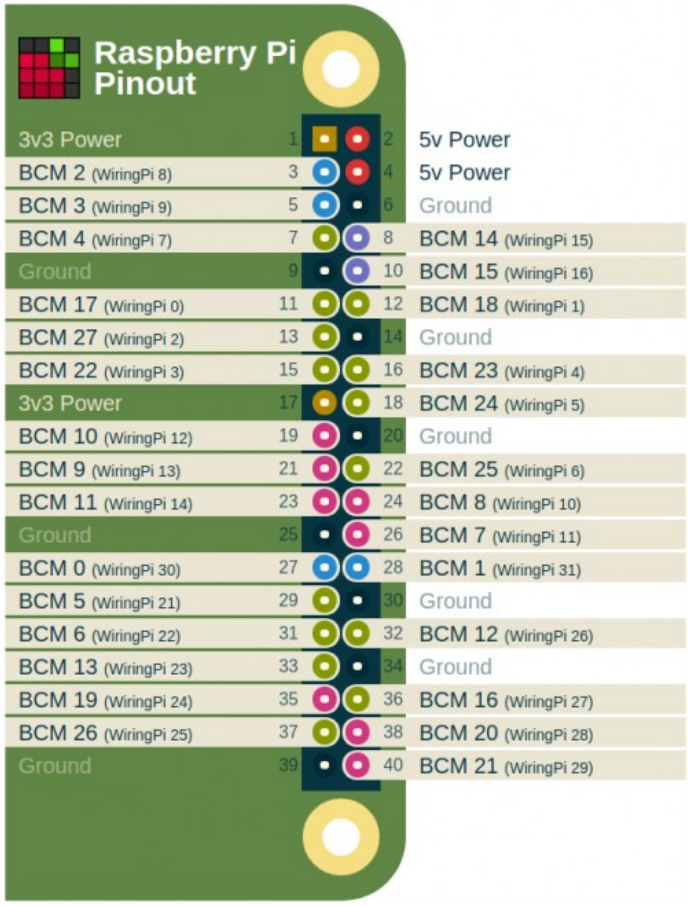


FIGURE - 8 GPIO PINOUT OF RASPBERRYPI

4.6 Camera

FIGURE -9 RASPBERRYPI 8MP NOIR CAMERA V2.1

RaspberryPi 8MP NoIR camera v2.1 is an 8-Megapixel native resolution high-quality Sony IMX219 image sensor, This camera is capable of 3280 x 2464 pixel static images and it can capture video at 1080p30, 720p60 and 640x480p90 resolutions, To use this camera raspbian os buster is required, No Infrared filter making it perfect for taking Infrared photographs or photographing objects in low light (twilight) conditions, 1.4 µm X 1.4 µm pixel with Omni-BSI technology for high performance.

This camera is connected to Camera Serial Interface (CSI) port of RaspberryPi 4B, after installing and initialising Raspicam library in geany editor the camera's arguments can be configured according to the environment conditions.

4.7 Power Supply

In order to transfer have sufficient power, two ground connected power banks are being used in this prototype, the power supplied to different components are as follows:

- Raspberry Pi
- Motor Power supply circuit

Power Bank 1 supplies power to RaspberryPi via USB Type-C interface, RaspberryPi requires 5V=3A to work optimally.

Power Bank 2 supplies power to Arduino UNO and Motor Controller, In-order to provide required power the power bank is triggered to supply power at QC3 standards i.e 12V=3A it is achieved with the help of custom made QC3 trigger circuit. The 12V is supplied to Vcc of Arduino UNO which can take voltage from anywhere in between 5V-19V

Power Bank 2 is also connected to buck Converter which steps-down the provided 12V to required amount while increasing total current that can be supplied to motor

Buck Converter is a simple and widely used voltage step-down device with high efficiency, the voltage can be changed by potentiometer, if voltage decreases current increases significantly, the output of buck converter is connected to 12V input LM298 motor controller.

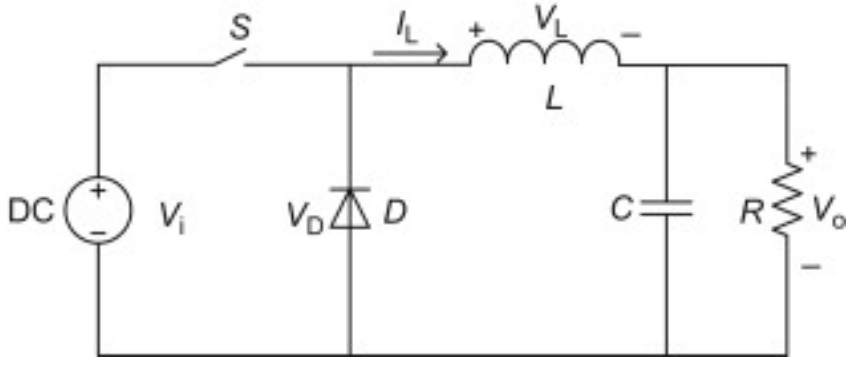


FIGURE -10 TYPICAL CIRCUIT OF A BUCK CONVERTER.

5. Assembly

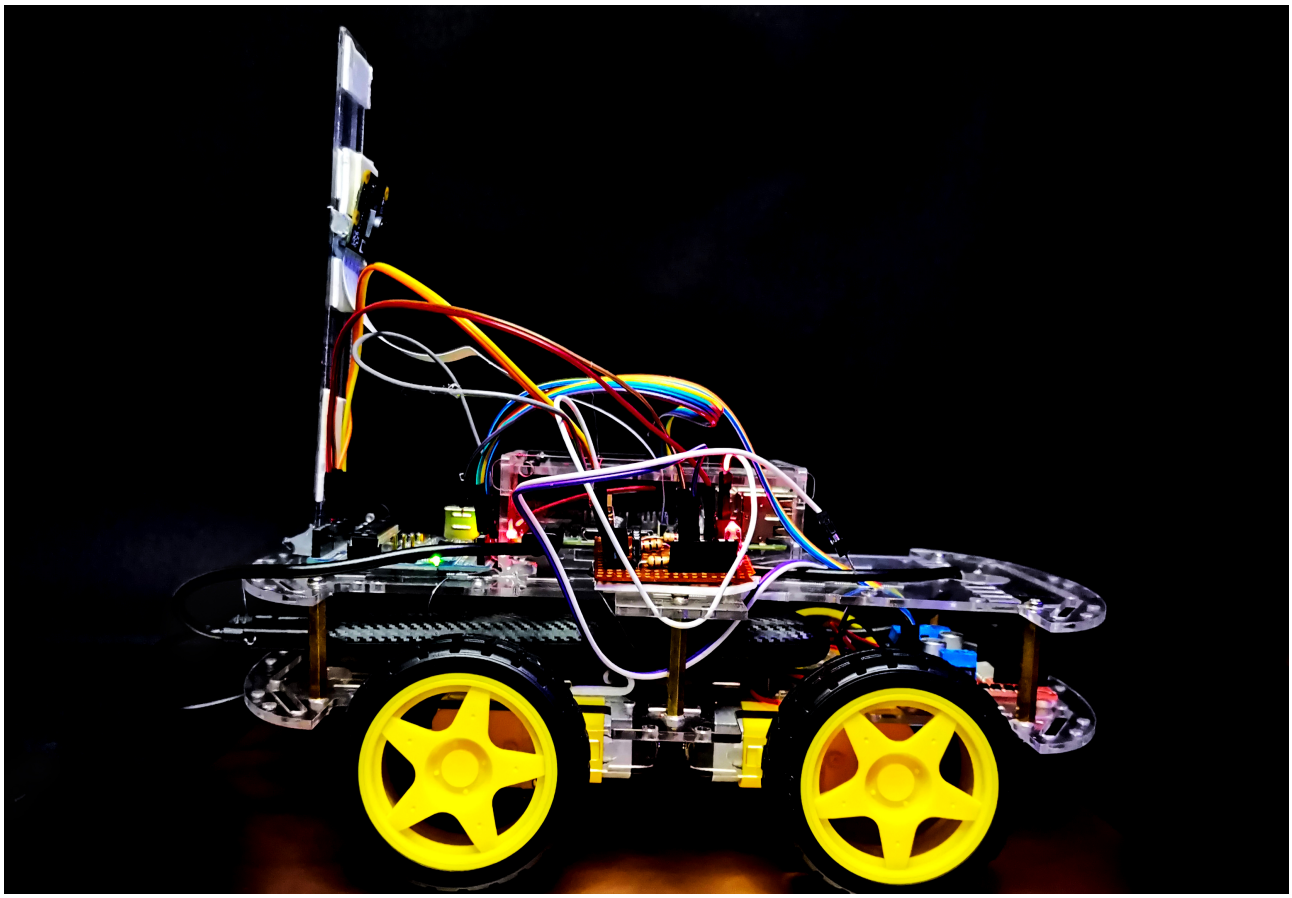


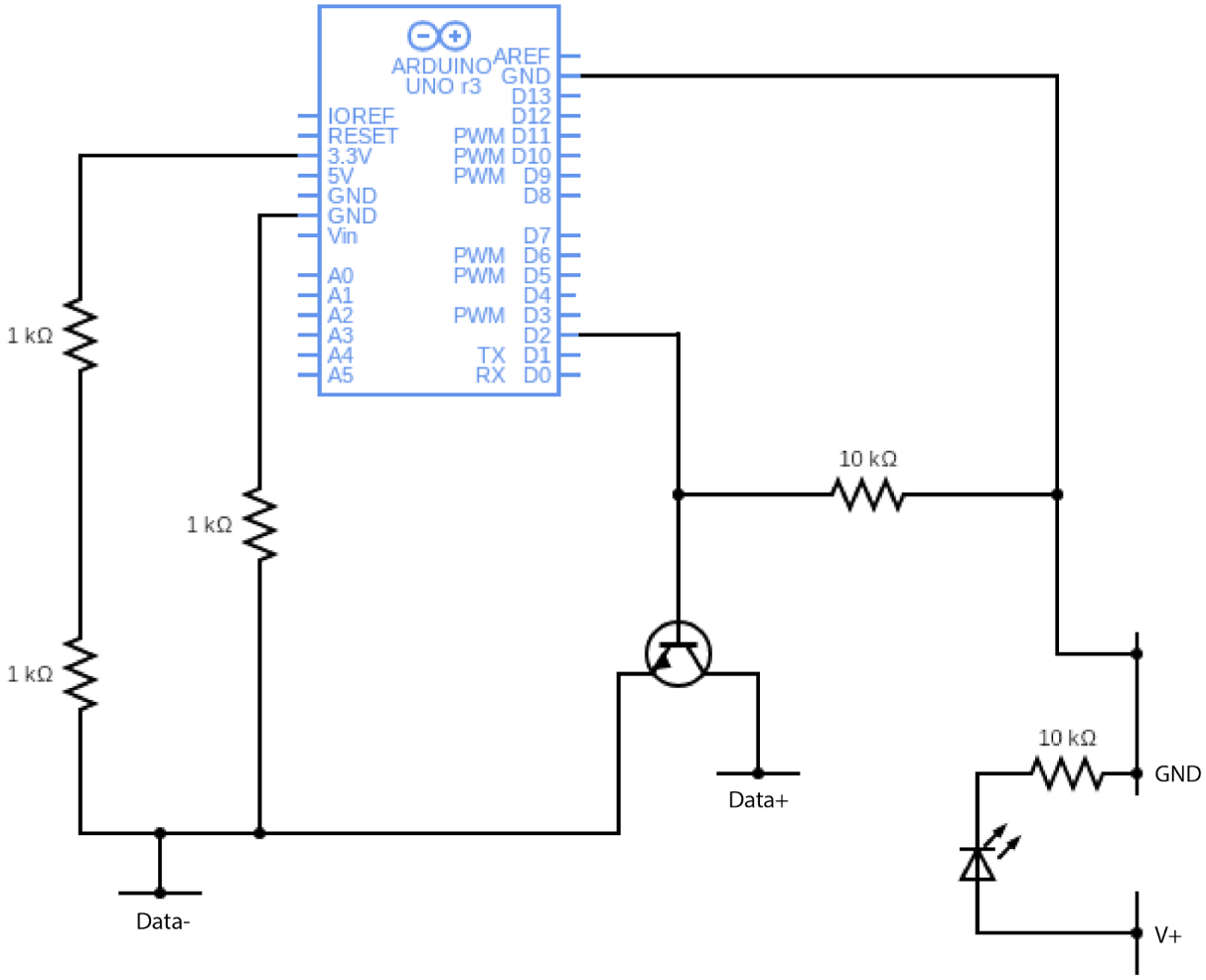
FIGURE- 11 ASSEMBLED PROTOTYPE SIDE-VIEW

The 4 wheels of the chassis are connected to 4 separate motors. The motor driver IC L298 is capable of driving 2 motors simultaneously. The rotation of the wheels is synchronized on the basis of the sides i.e. the left front and left back wheels rotate in sync and right front and right back- wheels rotate in sync. Thus the pair of motors on each side is given the samedigital input from L298 at any moment. This helps the car in forward, backward movements when both side wheels rotate in same direction with same speed. The car turns when the left side wheels rotate in opposite direction to those in right.

The chassis has two shelves over the wheels separated by 2 inch approx. The IC is fixed on the lower shelf with the help of two 0.5 inch screws. It is permanently connected to the motor wires and necessary jumper wires are drawn from L298 to connect to Arduino UNO. The

rest of the space on the lower shelf is taken by power bank 2 which provide the power to run the buck converter using 12V voltage trigger.

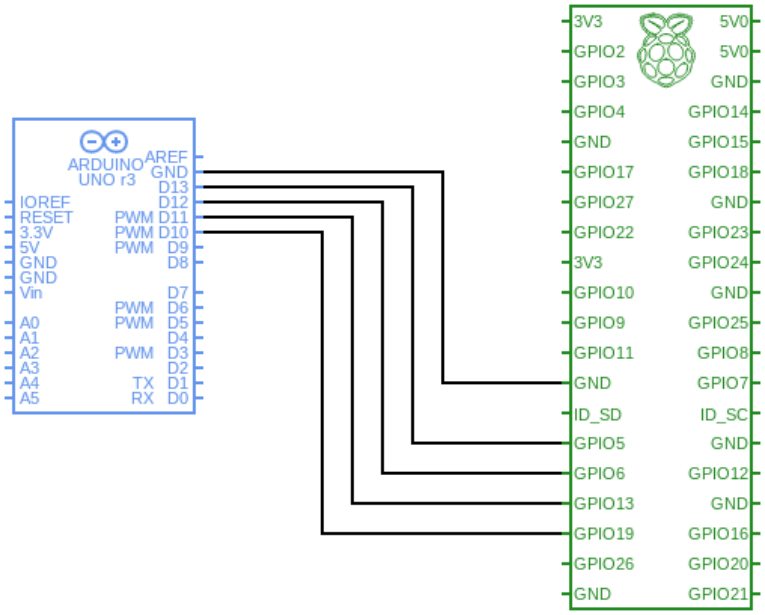
To control the motor connected to OUT1, OUT2, OUT3, OUT4 of motor driver, Left side motor's ENA, IN1, IN2 are connected to digital pins 5~, 6,7 of Arduino UNO where '~' indicates PWM pins, Right side motor's ENB, IN3, IN4 are connected to digital pins 3~,4,5 of Arduino UNO.



Circuit Diagram of OC3 trigger board

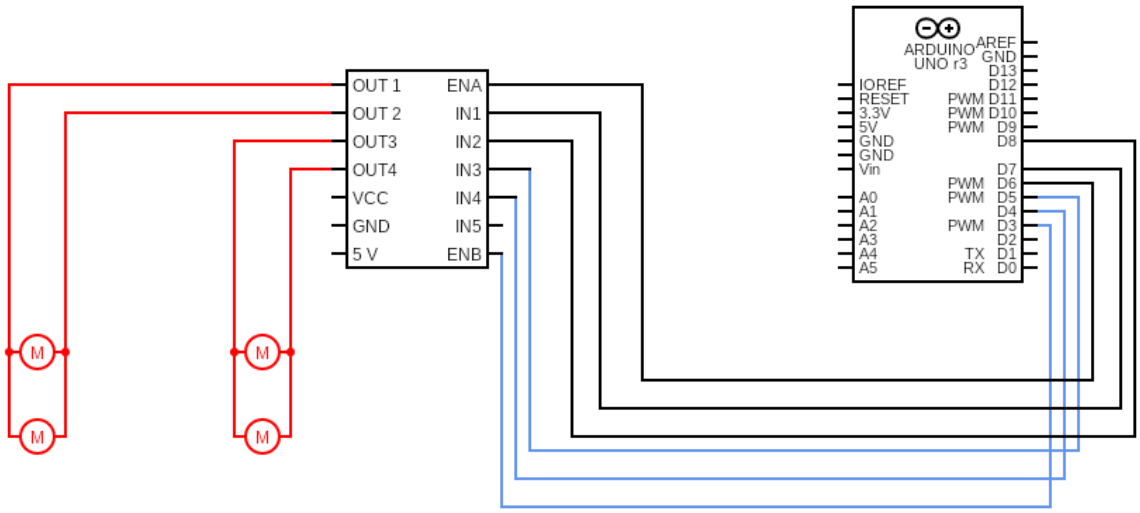
The raspberry pi case is glued on the top shelf along with Arduino UNO. The pi is fit in the case and two sick glued on top gives the support to the camera and power bank 1 is connect-

ed to it via USB-C cable, Power Management circuit is also glued and connected to power bank 2 its Vout and Ground bus is connected to Arduino UNO, LM298 motor driver, its 3.3v IN is connected to 3.3v out of Arduino UNO and gate of MOSFET is connected digit pin 2 of Arduino UNO.



Master-Slave Connection between raspberryPi and Arduino UNO

Arduino UNO is connected to the USB port in Raspberry Pi in order to communicate serially. RaspberryPi's wiringPi GPIO pins are unidirectional in parallel on pins 21-24 (where 21 is MSB and 24 is LSB) connected to digitalPin 13-10 set as input and pulled up to 5V. The complete connection of the Arduino UNO with motor controller L298 can be found in figure



.

Circuit Connection of Arduino UNO and L298 motor controller

5.1 Camera

Raspicam is connect to CSI camera port present behind ethernet RaspberryPi 4b via 15-pin ribbon cable.

FIGURE-12 CAMERA CONNECTION TO CSI PORT OF RASPBERRYPI4.1 RASPBERRY PIE SETUP

The camera is then Mounted on top of chassis with the help of hot glue, it is mounted at the backside of the car and calibrated in order to get perfect bird's eve perspective for lane detection algorithm.

5.2 Setting Up RaspberryPi

1. Go to the RaspberryPi official webpage by the link below:
2. https://www.raspberrypi.com/software/
3. Scroll down till you find the 'Download' button below.
4. Click the 'Download' button for your Computer's Operating System.
5. Install the RaspberryPi imager on your Computer.

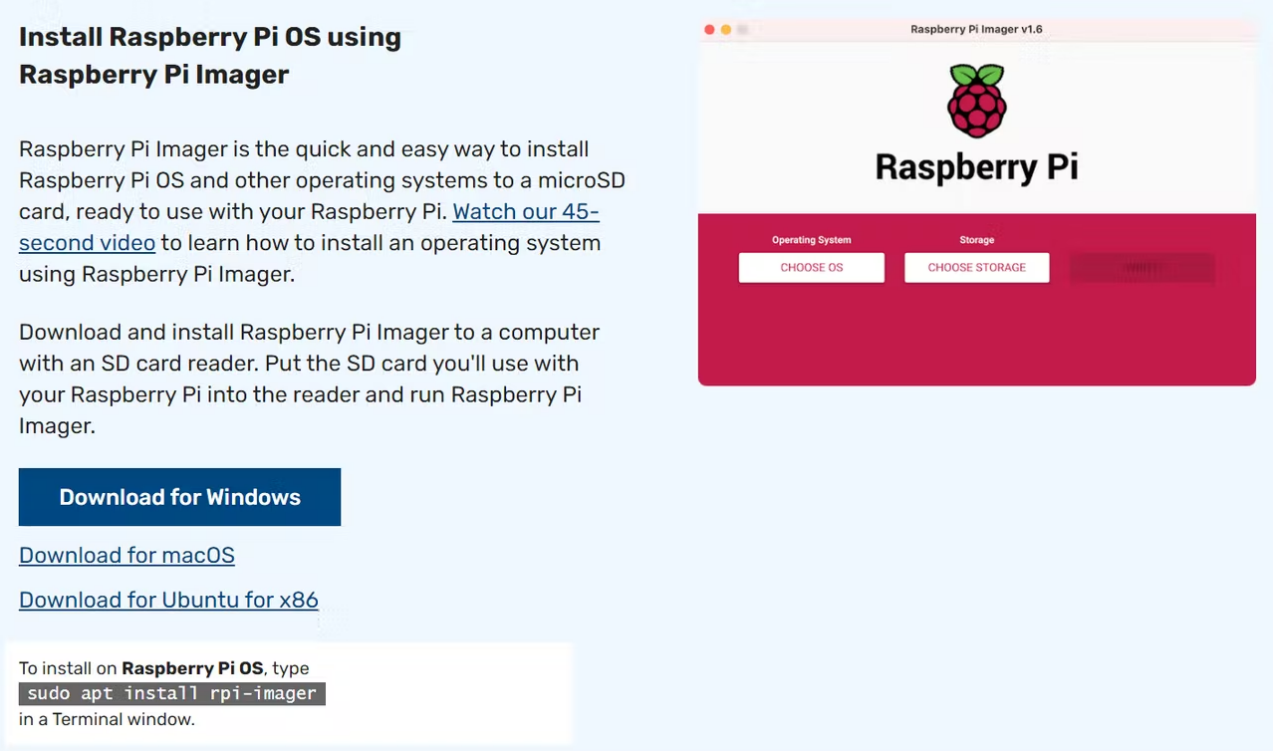


FIGURE-13 INSTALLATION WINDOW

Write the OS into your SD Card:-

1. Connect your microSD card to your computer.
2. Open RaspberryPii installer, and Click 'Choose OS'.
3. Click on 'Raspberry Pi OS (32-bit)'/The one on the top.
4. Click 'Choose Storage' and select your SD Card reader.
5. DO NOT click "Write" yet.

SSH and Wi-Fi Enabling:-

1. Press Ctrl + Shift + X to bring up the Advanced options menu.
2. Tick Enable SSH tick boxes in the menu.
3. Give your RaspberryPi a password for SSH.

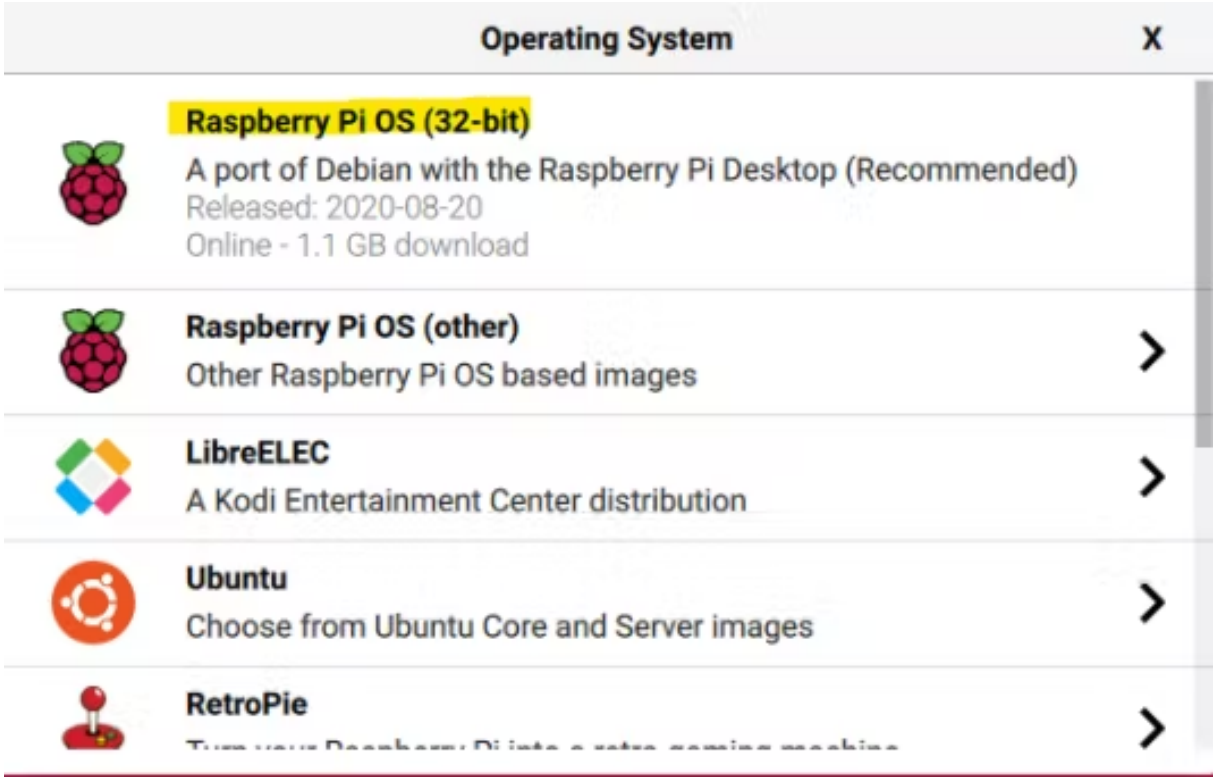


FIGURE- 14 OPERATING SYSTEM

Install N-Map on your computer:-

1. Go to the following link, download and install N-map on your computer:
2. https://nmap.org/download.html

Install VNC Viewer on your computer:-

1. Go to the following link, download and install VNC Viewer on your computer:
2. https://www.realvnc.com/en/connect/download/viewer/
3. Make sure you install 'VNC Viewer', and NOT 'VNC Server'.

Insert your microSD card into your RaspberryPi:-

1. After verifying the OS, go ahead and get your microSD card from your SD card Adapter, and plug it into your RaspberryPi.
2. Then plug in you're RaspberryPi into its power source.

Find your RaspberryPi IP Address:-

1. I use 'Command Prompt', but you can use terminals such as 'Windows Power-shell'.
2. Setting > Network and Internet > Wi-Fi > Properties > IPv4
3. In IPv4, you will find the IP address of your Computer.
4. In your terminal: nmap -sn xxx.xxx.x.0/24
5. Use '0/24' instead of the last number after the last period (decimal point) of your IP, This will run through all the IP Addresses in your router, to find the RaspberryPi.
6. You will find your RaspberryPi's IP address, if you didn't find it, repeat it after a minute till you find the IP, listed next to an item, 'RaspberryPi'.

Get into you Pi

1. After that use your terminal to get into your RaspberryPi's terminal.
2. Type on your terminal:
3. ssh pi@'IP'
4. Use the RaspberryPi's IP that you got, instead of 'IP' (xxx.xxx.x.14 is the number that I got, your number can be different.) .
5. Then you will find a prompt asking for your password.
6. Type the password that you entered in the RaspberryPi Imager for SSH, to get into your Pi.

View desktop of RaspberryPi

1. Launch VNC Viewer on your computer, type your RaspberryPi's IP:
2. Type in pi for the username and,
3. Type in your password, it should be raspberry .
4. But if the password doesn't work, type in your SSH password.
5. You should now see your Desktop:
6. It is completely normal if you had an error message now, I got it too!
7. You might be thinking 'I did all of this for an error!'. But fixing it is relatively simple.
8. Get onto the RaspberryPi terminal using step 8, Then type:
9. sudo raspi-config
10. Go to 'Display options' and hit Enter:
11. Go to 'VNC Resolution', and select and confirm '1280x720' (I don't know why, but that resolution seems to work for everyone):

12. Go to 'Finish', enter it and it will ask you to reboot. Press Enter, when the pink selector is on 'Yes'.
13. Now the connection will be interrupted because of the reboot.
14. Now, try to do step 11.
15. If it didn't work, it's completely normal. I didn't get the desktop too.
16. Redo step 9, to go to the RaspberryPi terminal, if it doesn't work, repeat after a minute.
17. On the RaspberryPi terminal, type:
18. sudo apt-get install lx session
19. After LXSession is installed, type:
20. sudo reboo

5.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognise faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognise scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people

of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

Installing OpenCV on RaspberryPi Python3

OpenCV can be installed by python package manager PIP by following commands:

```
sudo apt-get update
sudo apt-get upgrade
pip3 install —upgrade pip
Pip3 install opencv-contrib-python
```

Install dependencies

```
sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev
sudo apt-get install build-essential cmake pkg-config
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install libatlas-base-dev gfortran
```

Downloading OpenCV libraries

```
mkdir opencv

cd ~/opencv

git clone https://github.com/opencv/opencv.git

git clone https://github.com/opencv/opencv_contrib.git
```

Building Using CMAKE

```
mkdir build

cd build

cmake -D CMAKE_BUILD_TYPE=Release -DCMAKE_EXTRA_MODULES_PATH=../
opencv_contrib/modules/ -D CMAKE_INSTALL_PREFIX=/usr/local ../opencv
-DBUILD_opencv_java=OFF -DBUILD_opencv_python2=OFF
-DBUILD_opencv_python3=OFF

make -j4

sudo make install
```

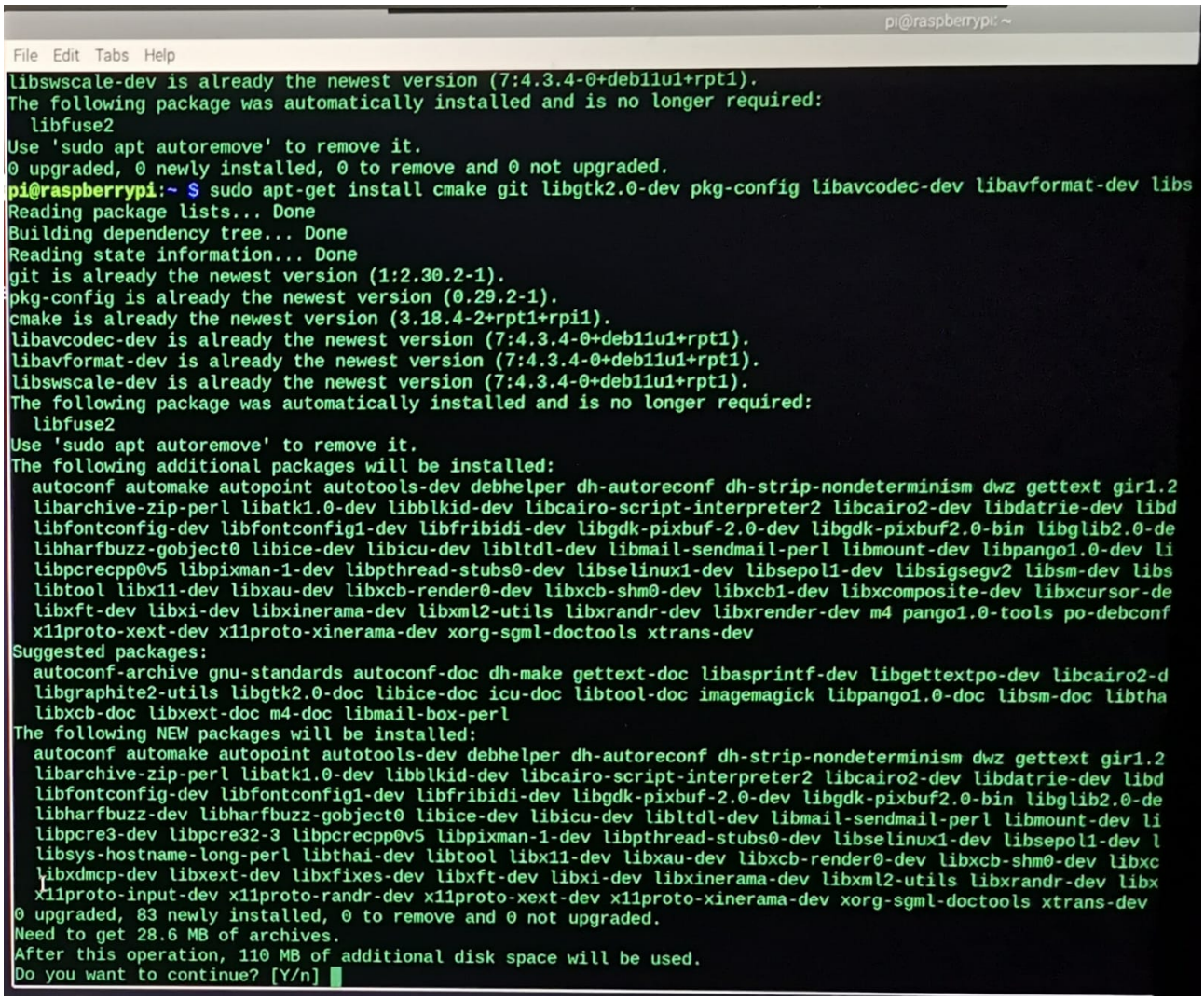


FIGURE -15A BUILDING OPENCV WITH CMAKE

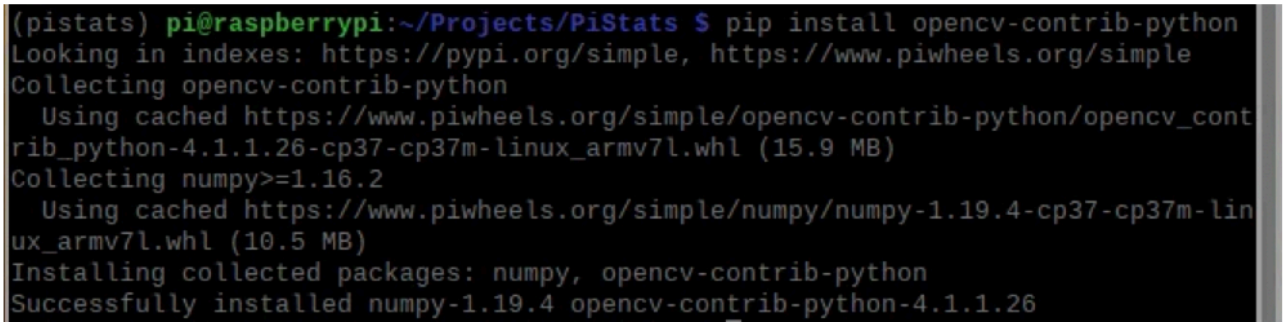


FIGURE- 15B INSTALLING OPENCV USING PIP

6. Lane Detection

Lane Detection algorithm helps car navigate through the lanes of road, It is achieved by OpenCV library.

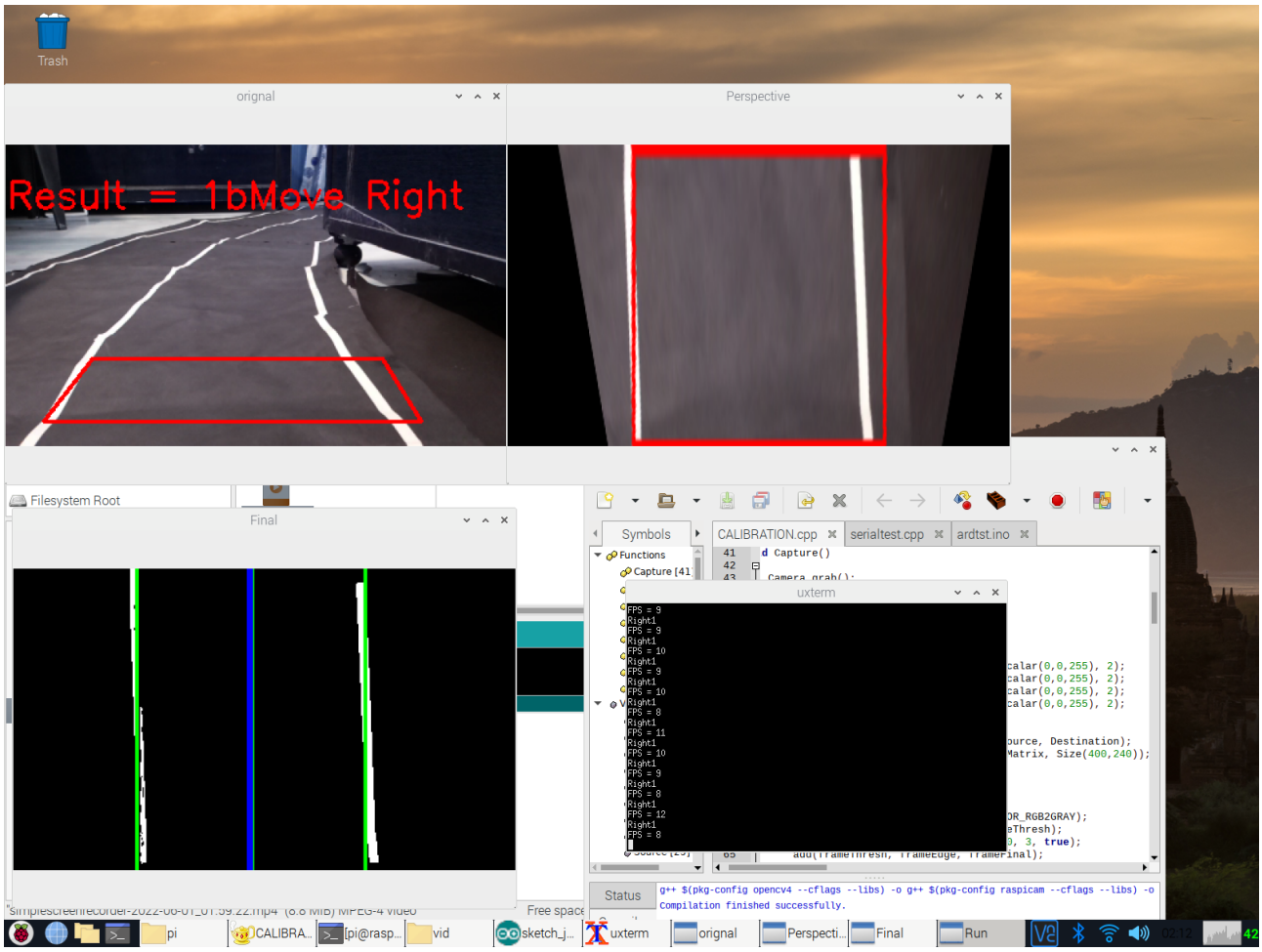


FIGURE -16 LANE DETECTION ALGORITHM

Capturing and decoding video file: We will capture the video using VideoCapture object and after the capturing has been initialized every video frame is decoded (i.e. converting into a sequence of images).

Grayscale conversion of image: The video frames are in RGB format, RGB is converted to grayscale because processing a single channel image is faster than processing a three-channel colored image.

Region of Interest: This step is to take into account only the region covered by the road lane. A mask is created here, which is of the same dimension as our road image. Furthermore, bitwise AND operation is performed between each pixel of our canny image and this mask. It ultimately masks the canny image and shows the region of interest traced by the polygonal contour of the mask.

Reduce noise: Noise can create false edges, therefore before going further, it's imperative to perform image smoothening. Gaussian filter is used to perform this process.

Canny Edge Detector: It computes gradient in all directions of our blurred image and traces the edges with large changes in intensity. For more explanation please go through this article: Canny Edge Detector

Hough Line Transform: The Hough Line Transform is a transform used to detect straight lines. The Probabilistic Hough Line Transform is used here, which gives output as the extremes of the detected lines

6.1 Canny Edge Detection

When it comes to image classification, the human eye has the incredible ability to process an image in a couple of milliseconds, and to determine what it is about (label). It is so amazing that it can do it whether it is a drawing or a picture. The idea today is to build an algorithm that can sketch the edges of any object present on a picture, using the Canny edge detection algorithm.First of all, let's describe what is the Canny Edge Detector:-

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a computational theory of edge detection explaining why the technique works.

The Canny edge detection algorithm is composed of 5 steps:-

1. Noise reduction;

2. Gradient calculation;

3. Non-maximum suppression;

4. Double threshold;

5. Edge Tracking by Hysteresis.

6.2 Noise Reduction

Since the mathematics involved behind the scene are mainly based on derivatives (cf. Step 2: Gradient calculation), edge detection results are highly sensitive to image noise.

One way to get rid of the noise on the image, is by applying Gaussian blur to smooth it. To do so, image convolution technique is applied with a Gaussian Kernel (3x3, 5x5, 7x7 etc…). The kernel size depends on the expected blurring effect. Basically, the smallest the kernel, the less visible is the blur. In our example, we will use a 5 by 5 Gaussian kernel.

The equation for a Gaussian filter kernel of size (2k+1)×(2k+1) is given by:

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \leq i, j \leq (2k+1)$$

6.2 Gradient Calculation

The Gradient calculation step detects the edge intensity and direction by calculating the gradient of the image using edge detection operators. Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y) When the image is smoothed, the derivatives Ix and Iy w.r.t. x and y are calculated. It can be implemented by convolving I with Sobel kernels Kx and Ky, respectively:-

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

Sobel filters for both direction (horizontal and vertical). Then, the magnitude G and the slope θ of the gradient are calculated as follow:-

$$|G| = \sqrt{I_x^2 + I_y^2},$$
$$\theta(x, y) = arctan\left(\frac{I_y}{I_x}\right)$$

The result is almost the expected one, but we can see that some of the edges are thick and others are thin. Non-Max Suppression step will help us mitigate the thick ones.

Moreover, the gradient intensity level is between 0 and 255 which is not uniform. The edges on the final result should have the same intensity (i-e. white pixel = 255).

6.4 Non-Maximum Suppression

Ideally, the final image should have thin edges. Thus, we must perform non-maximum suppression to thin out the edges. The principle is simple: the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.

Let's take an easy example:-

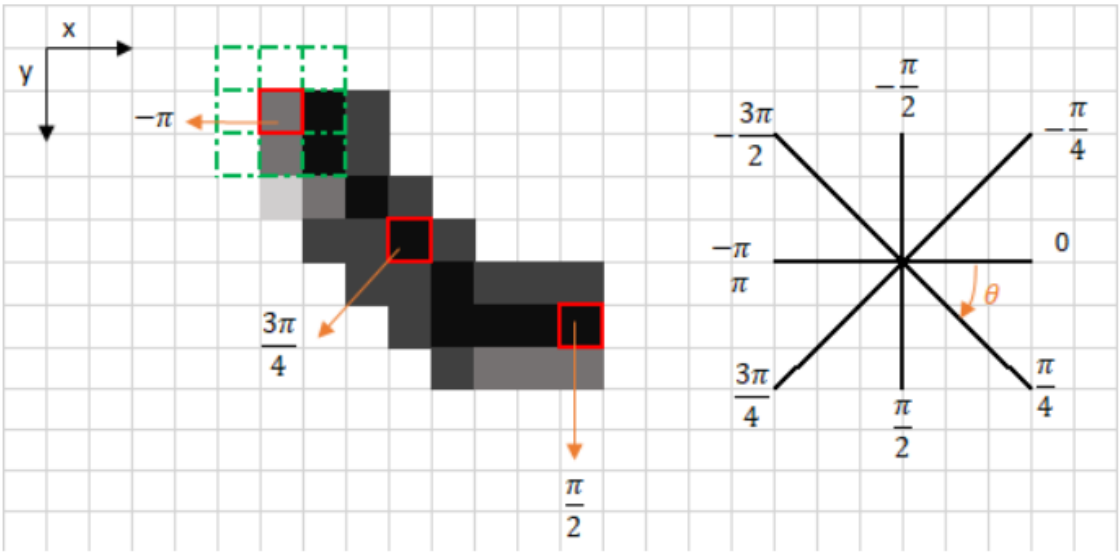


FIGURE- 17 NON MAXIMUM SUPPRESSION

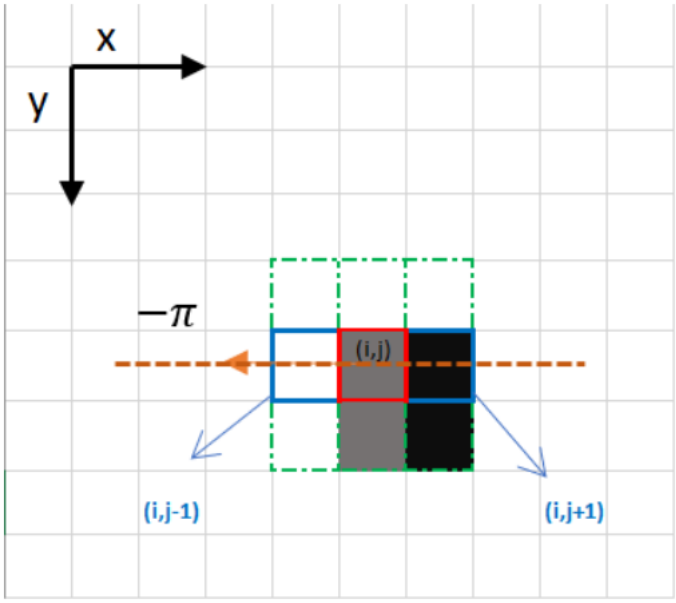


FIGURE- 18 FOCUS ON THE UPPER LEFT CORNER RED BOX PIXEL

The edge direction is the orange dotted line (horizontal from left to right). The purpose of the algorithm is to check if the pixels on the same direction are more or less intense than the ones being processed. In the example above, the pixel (i, j) is being processed, and the pixels on the same direction are highlighted in blue (i, j-1) and (i, j+1). If one those two pixels are more intense than the one being processed, then only the more intense one is kept. Pixel (i, j-1) seems to be more intense, because it is white (value of 255). Hence, the intensity value of the current pixel (i, j) is set to 0. If there are no pixels in the edge direction having more intense values, then the value of the current pixel is kept.

Let's now focus on another example:-

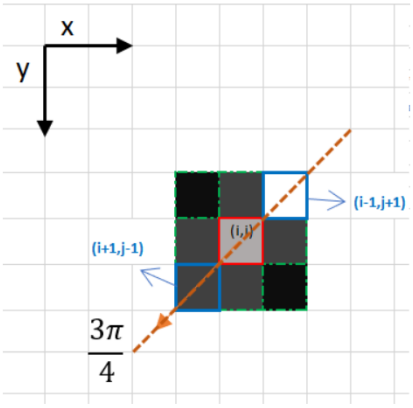


FIGURE-19 EDGE DETECTION

In this case the direction is the orange dotted diagonal line. Therefore, the most intense pixel in this direction is the pixel (i-1, j+1).

Let's sum this up. Each pixel has 2 main criteria (edge direction in radians, and pixel intensity (between 0–255)). Based on these inputs the non-max-suppression steps are:-

1. Create a matrix initialized to 0 of the same size of the original gradient intensity matrix;
2. Identify the edge direction based on the angle value from the angle matrix;
3. Check if the pixel in the same direction has a higher intensity than the pixel that is currently processed;
4. Return the image processed with the non-max suppression algorithm.

The result is the same image with thinner edges. We can however still notice some variation regarding the edges' intensity: some pixels seem to be brighter than others, and we will try to cover this shortcoming with the two final steps.

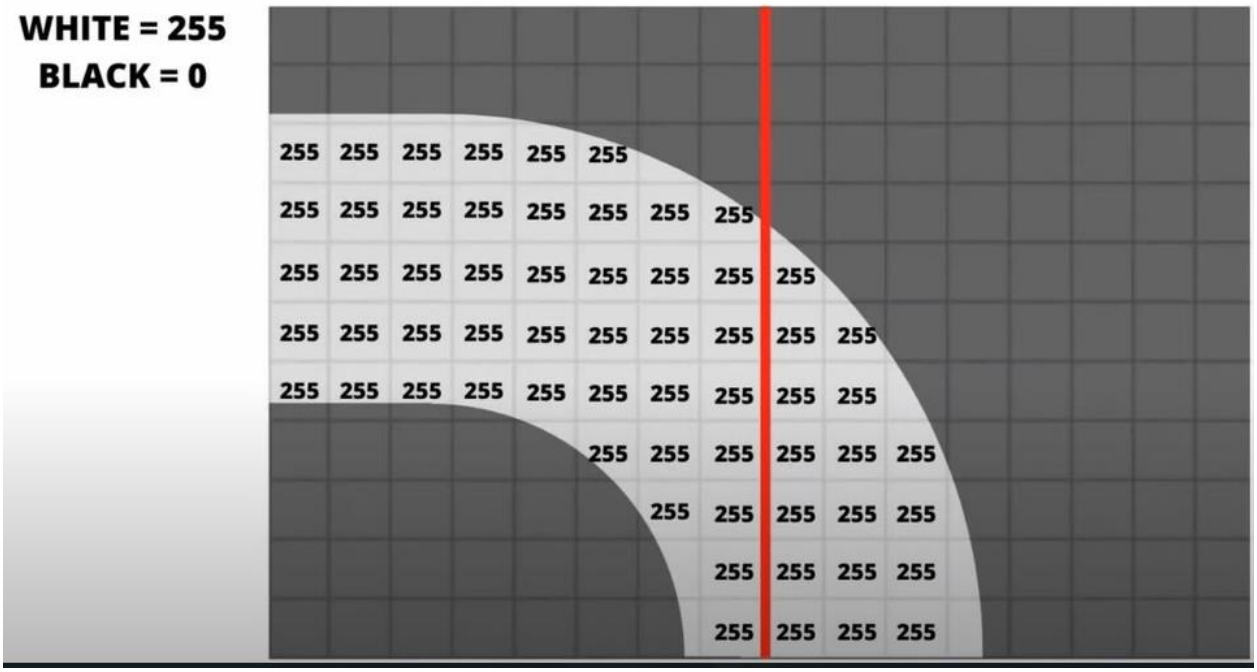


FIGURE- 20 PIXEL SUMMATION

6.5 Double threshold

The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant:-

1. Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
2. Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
3. Other pixels are considered as non-relevant for the edge.
4. High threshold is used to identify the strong pixels (intensity higher than the high threshold)
5. Low threshold is used to identify the non-relevant pixels (intensity lower than the low threshold)
6. All pixels having intensity between both thresholds are flagged as weak and the Hysteresis mechanism (next step) will help us identify the ones that could be considered as strong and the ones that are considered as non-relevant.

6.6 Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below:-

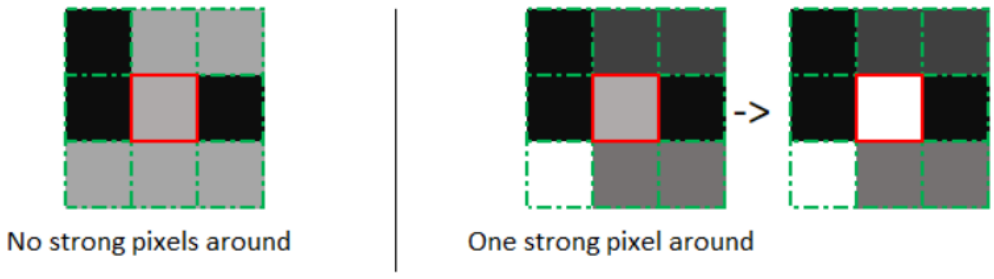


FIGURE-21A EDGE TRACKING

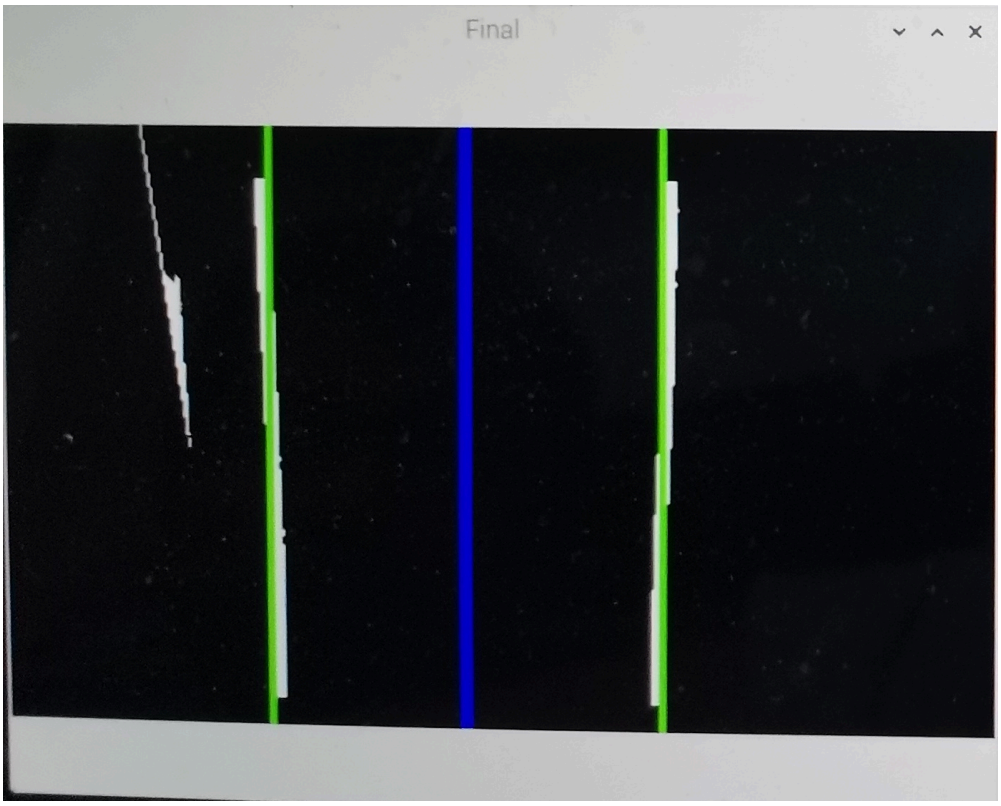


FIGURE-21B EDGE TRACKING LIVE FEED

FIGURE- 22A INPUT

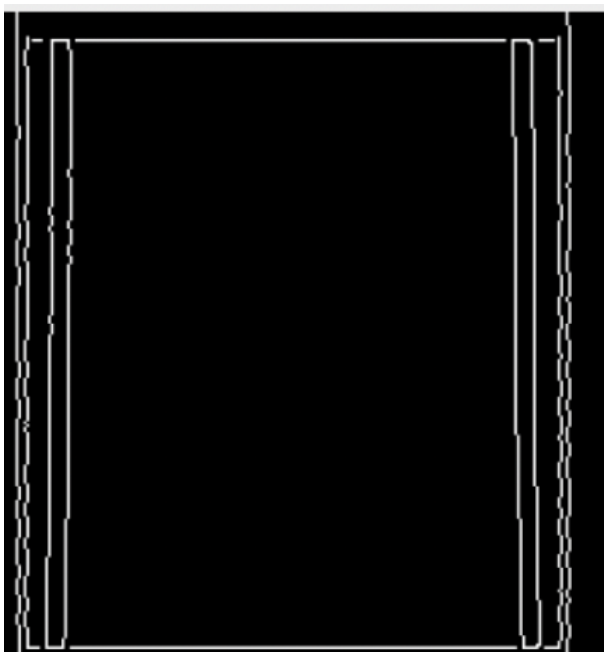


FIGURE- 22B OUTPUT

7. Object Detection

Image classification is one of the many exciting applications of convolutional neural networks. Aside from simple image classification, there are plenty of fascinating problems in computer vision, with object detection being one of the most interesting. YOLO ("You Only Look Once") is an effective real-time object recognition algorithm.

Object detection is commonly associated with self-driving cars where systems blend computer vision, LIDAR, and other technologies to generate a multidimensional representation of the road with all its participants.

To explore the concept of object detection it's useful, to begin with, image classification. Image classification goes through levels of incremental complexity.

7.1 Image classification

aims at assigning an image to one of a number of different categories (e.g. car, dog, cat, human, etc.), essentially answering the question "What is in this picture?". One image has only one category assigned to it.

7.2 Object localisation

allows us to locate our object in the image, so our question changes to "What is it, and where it is?".

7.3 Object detection

provides the tools for doing just that – finding all the objects in an image and drawing the so-called bounding boxes around them.

In a real-life scenario, we need to go beyond locating just one object but multiple objects in one image. For example, a self-driving car has to find the location of other cars, traffic lights, signs, and humans and take appropriate action based on this information.

7.4 The YOLO algorithm

The YOLO algorithm works by dividing the image into N grids, each having an equal dimensional region of S*S. Each of these N grids is responsible for the detection and localisation

of the object it contains. Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and probability of the object being present in the cell.

This process greatly lowers the computation as both detection and recognition are handled by cells from the image, but It brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions. YOLO makes use of Non Maximal Suppression to deal with this issue.

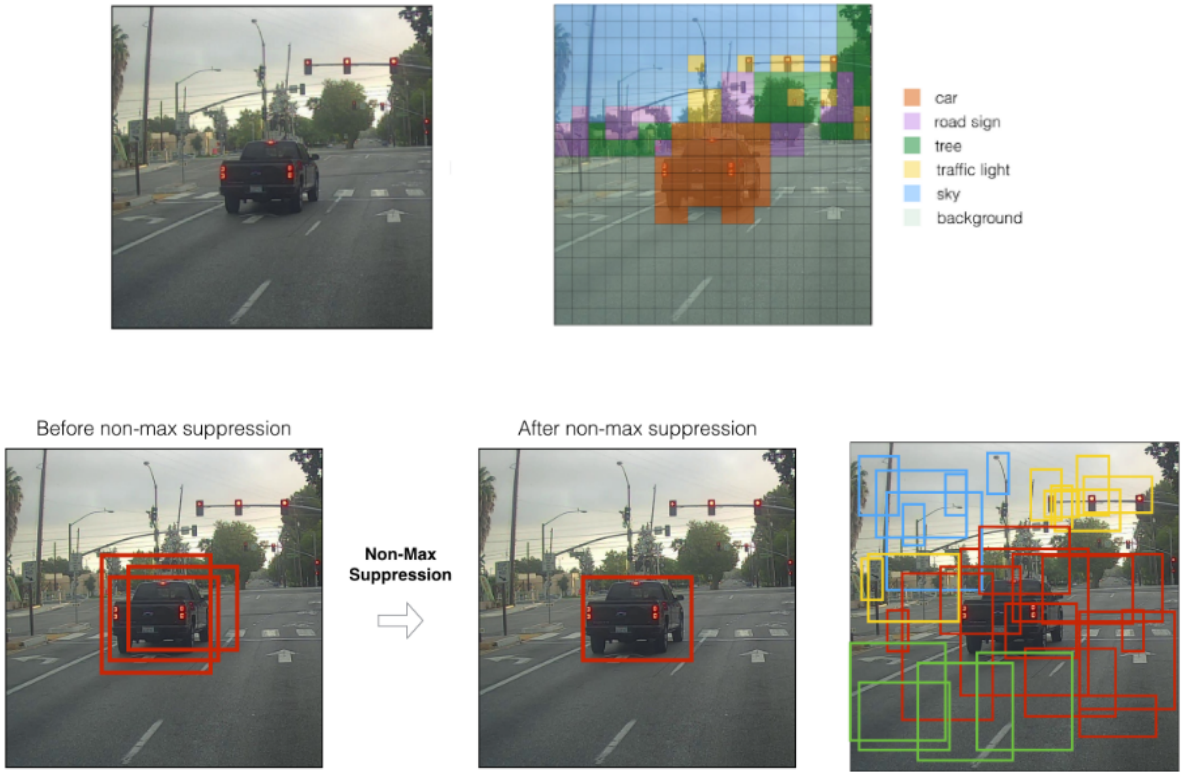


FIGURE- 23 PROCEDURE OF OBJECT DETECTION

In Non Maximal Suppression, YOLO suppresses all bounding boxes that have lower probability scores. YOLO achieves this by first looking at the probability scores associated with each decision and taking the largest one. Following this, it suppresses the bounding boxes having the largest Intersection over Union with the current high probability bounding box. This step is repeated till the final bounding boxes are obtained.

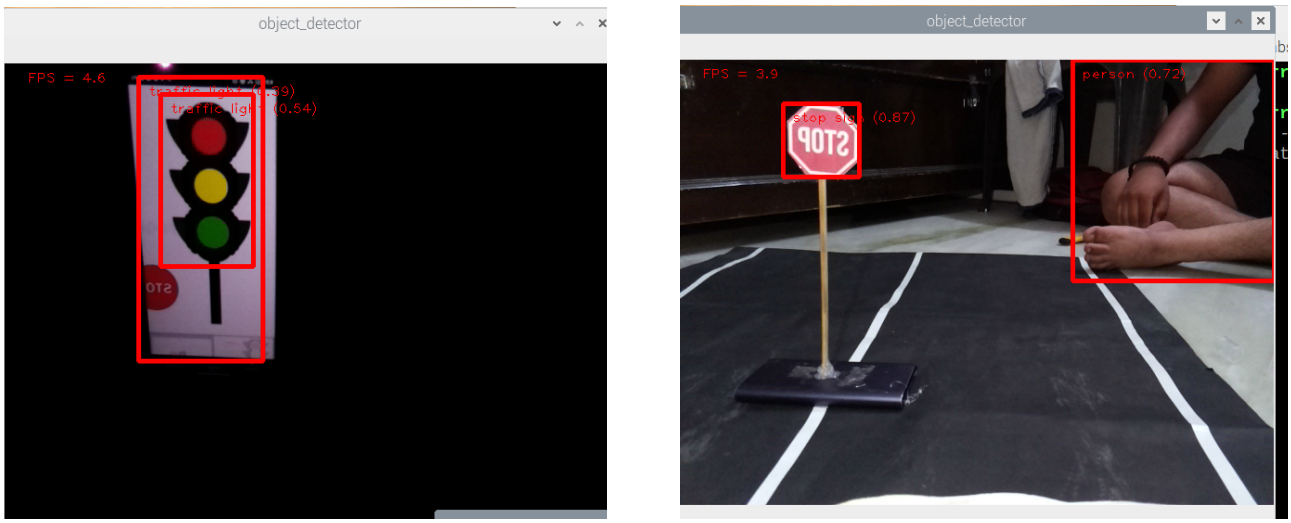


FIGURE -24 DETECTION OF A TRAFFIC LIGHT AND STOP SIGN.

7.5 YOLO Architecture

YOLO is a convolution neural network. It consists of a total of 24 convolutional layers and followed by 2 fully connected layers. Each layer has its own importance and the layers are separated by their functionality. Figure-25 shows the architecture of object detection model using convolutional neural networks.

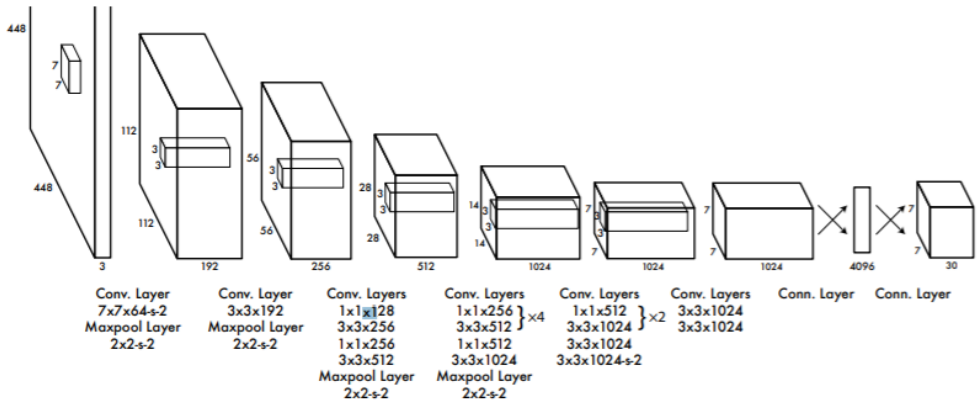


FIGURE - 25 ARCHITECTURE OF OBJECT DETECTION

The First 20 convolutional layers followed by an average pooling layer and a fully connected layer is pre-trained on the ImageNet dataset which is a 1000-class classification dataset. The pre training for classification is performed on the dataset with the image resolution of 224 x 224×3. The layers comprise 3×3 convolutional layers and1x1 reduction layers. For object detection, in the end, the last 4 convolutional layers followed by 2 fully connected layers are added to train the network. Object detection requires more precise detail hence the resolution of the dataset is increased to 448 x 448 Then the final layer predicts the class probabilities and bounding boxes. All the other convolutional layers use leaky ReLU activation whereas the final layer uses a linear activation. The input is of 448 x 448 image and the output is the class prediction of the detected object enclosed in the bounding box.

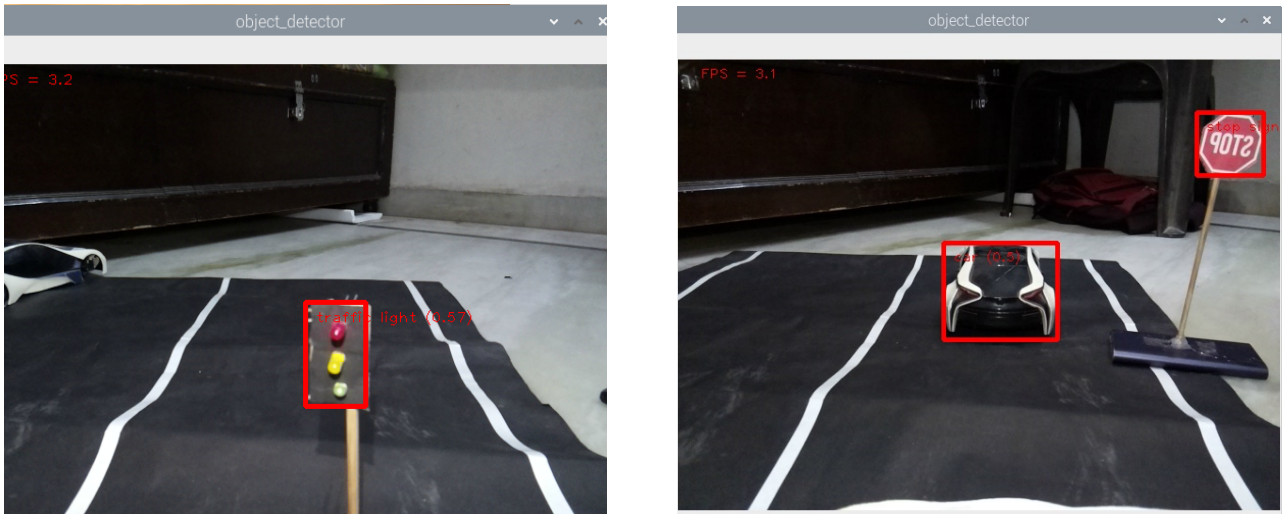


FIGURE-26 DETECTION OF OBSTACLES (CAR), TRAFFIC SIGNS

7.6 Training the object detection model with Tensor-flow

TensorFlow is a software library most famous for its flexibility and ease of use in neural networks. You can find a lot of examples online from image classification to object detection, but many of them are based on TensorFlow 1.x. It is a big change from TensorFlow 1.0 to 2.0 with a tighter Keras integration, where the focus is more on higher level APIs. Many methods have been depreciated (or you may use tf.compat.v1. Model construction becomes a lot easier and default parameters in each model already work very well for general use. With all the benefits, it still provides flexibility should you need to change the parameters.

It this article, I will use TensorFlow 2.0 (more specifically, Keras in TensorFlow) to classify traffic signs. The dataset is available many places on the web, but I will use this one hosted on Kaggle.

The data package includes folders of Train, Test and a test.csv. There are a meta.csv and a Meta folder to show the standard image for each traffic sign. There is also a signname.csv for mapping a label to its description. Train folder contains 43 sub-folders whose names are the labels of the images in them. For example, all the images in folder 0 has a class label of 0 and so on. The images are of different sizes ranging from 20x20 to 70x70, and all have 3 channels: RGB.

So the first thing we have to do is to resize all the images to 32x32x3 and read them into a numpy array as training features. At the same time, we have created another numpy array with labels of each image, which is from the fold name where the image loaded from.

```
import cv2
import glob
import pickle
import numpy as np
import pandas as pd

# function to read and resize images, get labels and store
them into np array
def get_image_label_resize(label, filelist, dim = (32,
32), dataset = 'Train'):
    x = np.array([cv2.resize(cv2.imread(fname), dim, in-
terpolation = cv2.INTER_AREA) for fname in filelist])
    y = np.array([label] * len(filelist))
#print('{} examples loaded for label
{}'.format(x.shape[0], label))
    return (x, y)

# data for label 0. I store them in parent level so that
they won't be uploaded to github
filelist = glob.glob('../Train/'+'0'+'/*.png')
trainx, trainy = get_image_label_resize(0, glob.glob('../
Train/'+str(0)+'/*.png'))
```

```
# go through all others labels and store images into np ar-
ray
for label in range(1, 43):
    filelist = glob.glob('../Train/'+str(label)+'/*.png')
    x, y = get_image_label_resize(label, filelist)
    trainx = np.concatenate((trainx ,x))
    trainy = np.concatenate((trainy ,y))

# get path for test images
testfile = pd.read_csv('Test.csv')['Path'].apply(lambda x:
'../' + x).tolist()

X_test = np.array([cv2.resize(cv2.imread(fname), (32, 32),
interpolation = cv2.INTER_AREA) for fname in testfile])
y_test = np.array(pd.read_csv('Test.csv')['ClassId'])
```

We need to do the same for testing images. However the labels for testing images are stored as `ClassId` in `test.csv` with paths of that image. So we used pandas to read the `csv` file, load the image from path and assign the corresponding ClassId.

From the training set, We have to randomly spit 20% as validation set for use during the process of model training. The model accuracy of training and validation will give us information about under-fitting or overfitting.

```
#convert the images to grayscale
X_train_gry = np.sum(X_train/3, axis=3, keepdims=True)
X_validation_gry = np.sum(X_validation/3, axis=3, keep-
dims=True)
X_test_gry = np.sum(X_test/3, axis=3, keepdims=True)
```

```
# shuffle training data and split them into training and
validation
indices = np.random.permutation(trainx.shape[0])
# 20% to val
split_idx = int(trainx.shape[0]*0.8)
train_idx, val_idx = indices[:split_idx],
indices[split_idx:]
X_train, X_validation = trainx[train_idx,:],
trainx[val_idx,:]
y_train, y_validation = trainy[train_idx], trainy[val_idx]


# get overall stat of the whole dataset
n_train = X_train.shape[0]
n_validation = X_validation.shape[0]
n_test = X_test.shape[0]
image_shape = X_train[0].shape
n_classes = len(np.unique(y_train))
print("There are {} training examples ".format(n_train))
print("There are {} validation examples".format(n_valida-
tion))
print("There are {} testing examples".format(n_test))
print("Image data shape is {}".format(image_shape))
print("There are {} classes".format(n_classes))


There are 31367 training examples
There are 7842 validation examples
There are 12630 testing examples
Image data shape is (32, 32, 3)
There are 43 classes
```

Next, We have converted images to grayscale and normalised each pixels. Normalisation makes model to converge more quickly.

```
# Normalize data
X_train_normalized_gry = (X_train_gry-128)/128
X_validation_normalized_gry = (X_validation_gry-128)/128
X_test_normalized_gry = (X_test_gry-128)/128

import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-colorblind')

# descriptions for each label
sign = pd.read_csv('signnames.csv')

# pick an image, display the original and the normalized
gray image
index = np.random.randint(0, n_train)
fig, ax = plt.subplots(1,2)
ax[0].set_title('original ' + sign.loc[sign['ClassId'] ==y_-
train[index], 'SignName'].values[0])
ax[0].imshow(cv2.cvtColor(X_train[index],
cv2.COLOR_BGR2RGB))

ax[1].set_title('norm_gry ' + sign.loc[sign['ClassId'] ==y_-
train[index], 'SignName'].values[0])
ax[1].imshow(X_train_normalized_gry[index].squeeze(),
cmap='gray')

<matplotlib.image.AxesImage at 0x1340cbc50>
```

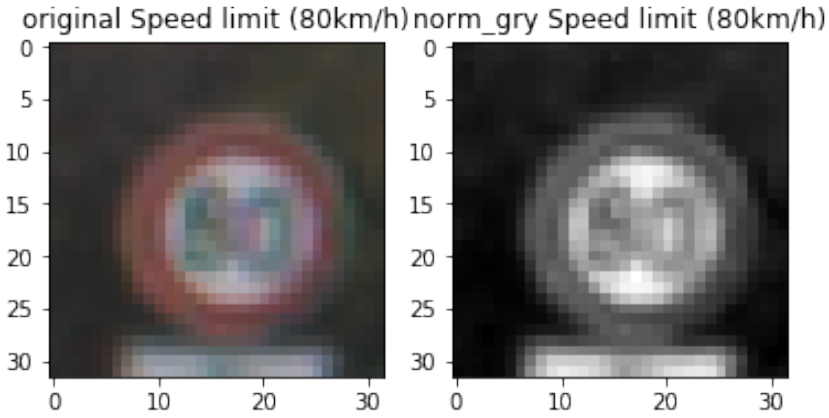


Figure-27A comparison of original and grayscale image

```
# update the train, val and test data with normalized gray images
X_train = X_train_normalized_gry
X_validation = X_validation_normalized_gry
X_test = X_test_normalized_gry
```

Here is a comparison between an RGB and grayscale image. The grayscale image still retains its features and can be recognised but with much smaller size.

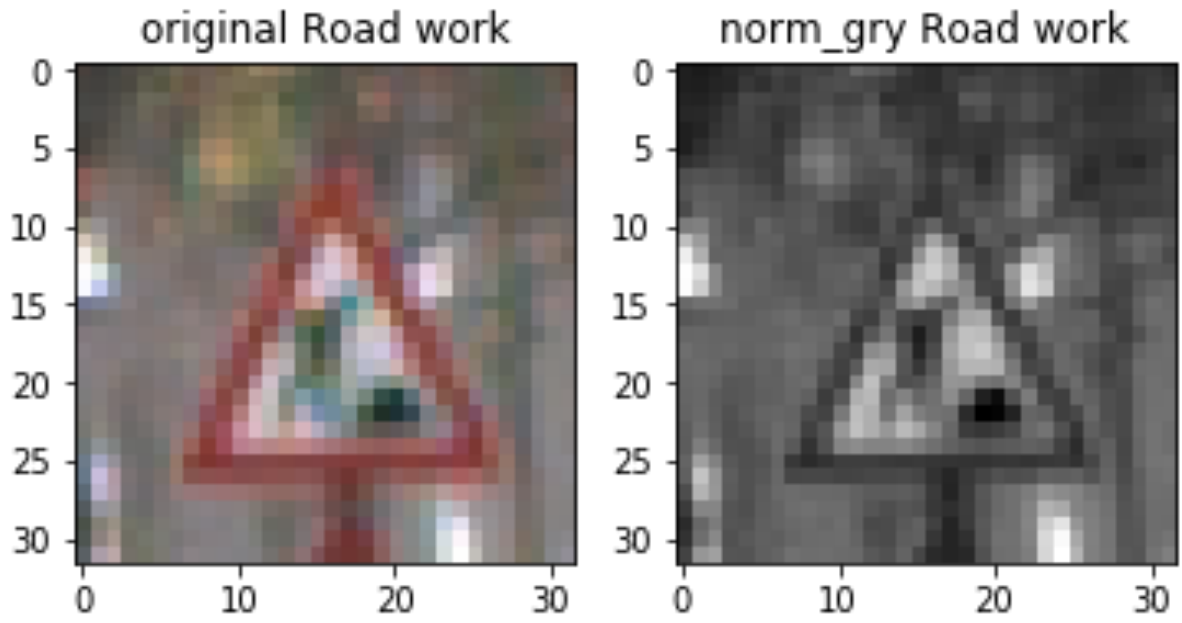


FIGURE-27B COMPARISON OF ORIGINAL AND GRAYSCALE IMAGE

6.1 Model construction

We have used the famous LeNet published in 1998 by Yann LeCun et al. The input shape is 32x32x1. First convolution layer will have a depth of 6, a filter size of (5, 5), and a stride of (1, 1). Valid padding is used (i.e. no padding). Therefore the width (or height) of this layer is 32–5+1 = 28, i.e. the shape is 28x28x6. The activation of this layer is relu.

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

model = models.Sequential()
```

```
# Conv 32x32x1 => 28x28x6.
model.add(layers.Conv2D(filters = 6, kernel_size = (5, 5),
strides=(1, 1), padding='valid',
activation='relu', data_format = 'channels_last', in-
put_shape = (32, 32, 1)))
# Maxpool 28x28x6 => 14x14x6
model.add(layers.MaxPooling2D((2, 2)))
# Conv 14x14x6 => 10x10x16
model.add(layers.Conv2D(16, (5, 5), activation='relu'))
# Maxpool 10x10x16 => 5x5x16
model.add(layers.MaxPooling2D((2, 2)))
# Flatten 5x5x16 => 400
model.add(layers.Flatten())
# Fully connected 400 => 120
model.add(layers.Dense(120, activation='relu'))
# Fully connected 120 => 84
model.add(layers.Dense(84, activation='relu'))
# Dropout
model.add(layers.Dropout(0.2))
# Fully connected, output layer 84 => 43
model.add(layers.Dense(43, activation='softmax'))
```

Following the first convolution layer is a max polling layer. It effectively downsizes the data by only selecting the max value pixel for adjacent pixels. LeNet uses a (2, 2) kernel size. The default stride is the same as kernel, which means there is no overlap between the group of pixels the max is selected from. Now the shape of output becomes 14x14x6.

Next LeNet has a second convolution layer with depth of 16, filter size of (5, 5) and relu activation function, followed by a max pooling layer. The width (or height) of output is now (14–5+1)/2 = 5, i.e. the shape is 5x5x16.

The data is then flattened before the fully connected layers. The shape of output is 5x5x16 = 400. This followed by 2 fully connected layers of size 120 and 84, with relu as activation function for both. A dropout layer is added to reduce overfitting. And finally a fully connected layer with size of 43 (the no. of classes). Softmax is used to return the probabilities of each class.

```
model.summary()


Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 6)         156
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 14, 14, 6)         0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 10, 10, 16)        2416
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_17 (Dense)             (None, 120)               48120
_________________________________________________________________
dense_18 (Dense)             (None, 84)                10164
_________________________________________________________________
dropout_5 (Dropout)          (None, 84)                0
_________________________________________________________________
dense_19 (Dense)             (None, 43)                3655
=================================================================
Total params: 64,511
Trainable params: 64,511
Non-trainable params: 0
_________________________________________________________________
```

6.2 Model training and evaluation

Training is very straightforward with Keras. We only need to specify optimizer, loss function and validation metric. Within 10 epochs, the accuracy of both training and validation is

above 0.97. With a dropout layer, there is no apparent overfitting. On the other hand, increasing training will only yield minimum improvement, so I stopped only after 10 epochs.

```
# specify optimizer, loss function and metric
model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy',
 metrics=['accuracy'])
# training batch_size=128, epochs=10
conv = model.fit(X_train, y_train, batch_size=128,
epochs=10,
 validation_data=(X_validation, y_validation))
```

```
Train on 31367 samples, validate on 7842 samples
Epoch 1/10
31367/31367 [==============================] - 8s 255us/sam-
ple - loss: 2.2325 - accuracy: 0.4007 - val_loss: 0.8845 -
val_accuracy: 0.7619
Epoch 2/10
31367/31367 [==============================] - 8s 241us/sam-
ple - loss: 0.7074 - accuracy: 0.7961 - val_loss: 0.4006 -
val_accuracy: 0.8986
Epoch 3/10
31367/31367 [==============================] - 8s 242us/sam-
ple - loss: 0.3948 - accuracy: 0.8884 - val_loss: 0.2567 -
val_accuracy: 0.9320
Epoch 4/10
31367/31367 [==============================] - 8s 243us/sam-
ple - loss: 0.2703 - accuracy: 0.9242 - val_loss: 0.1827 -
val_accuracy: 0.9547
Epoch 5/10
31367/31367 [==============================] - 8s 241us/sam-
ple - loss: 0.2034 - accuracy: 0.9419 - val_loss: 0.1470 -
val_accuracy: 0.9647
Epoch 6/10
31367/31367 [==============================] - 8s 240us/sam-
ple - loss: 0.1582 - accuracy: 0.9550 - val_loss: 0.1352 -
val_accuracy: 0.9651
```

We can also plot the model performance on training and validation with each epoch. Indeed, the model appears to be quite generalised and not overfitting the training data.

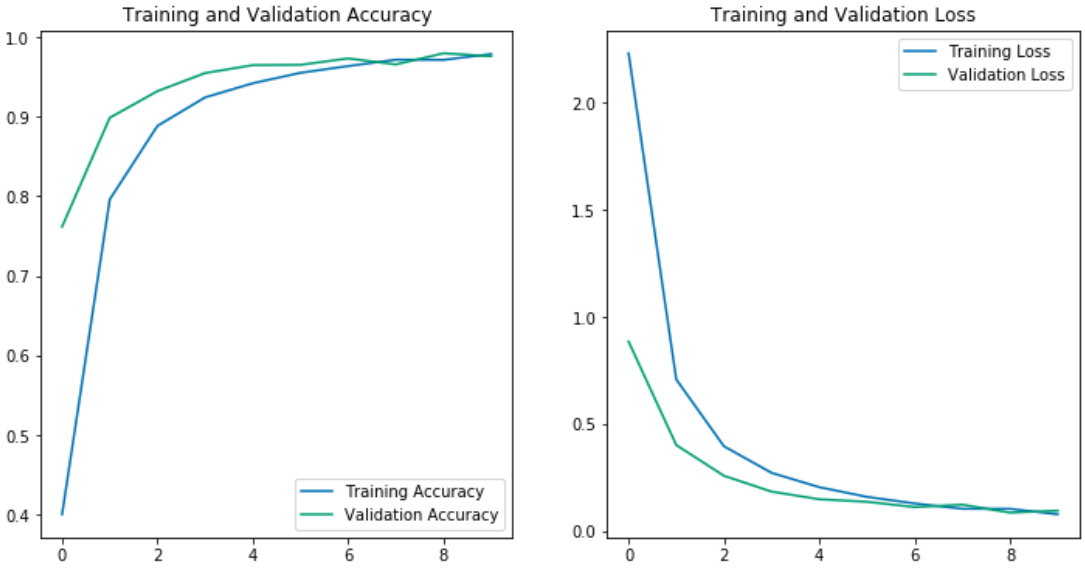


FIGURE- 27C CCURACY AND LOSS OF TRAINING AND VALIDATION FOR EACH EPOCH

Finally, the model is used to predict the labels of test set. The accuracy is about 0.925.

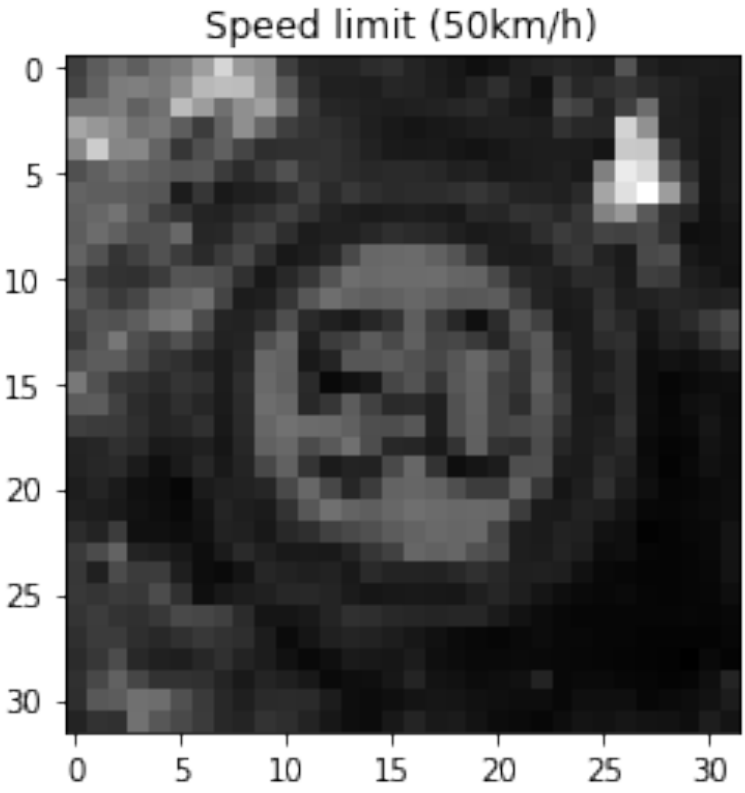


FIGURE-28 LABELLED OUTPUT

```
model.evaluate(x=X_test, y=y_test)
```

```
12630/12630 [==============================] - 1s 82us/sam-
ple - loss: 0.6084 - accuracy: 0.9250
[0.6084245350228081, 0.9250198]
```

```
index = np.random.randint(0, n_test)
im = X_test[index]
fig, ax = plt.subplots()
ax.set_title(sign.loc[sign['ClassId'] ==np.argmax(model.pre-
dict(np.array([im]))), 'SignName'].values[0])
ax.imshow(im.squeeze(), cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x149dfdbd0>
```

8. Safety Aspects and Regulations of self driving car

8.1 ARE SELF DRIVING ARE SAFE?

Self-driving cars are coming and there is a lot of excitement around the new technology. But, before they can be deployed on a larger scale, they need to be reliable and safe. Would you trust self-driving cars to take you where you want to go?

Many experts say that self-driving cars can be trained to be safer than human drivers. With the sensors and cameras monitoring and guiding, these cars can not only sense their environment but also can anticipate what's coming up ahead, which humans are not capable of. They may one day make the world a safer place by eliminating human error and reducing the number of car crashes.But, we should go through a long process before self-driving cars reach end consumers.

The scenarios for convenience and quality-of-life improvements are limitless. The elderly and the physically disabled would have independence. If your kids were at summer camp and forgot their bathing suits and toothbrushes, the car could bring them the missing items. You could even send your dog to a veterinary appointment.

But the real promise of autonomous cars is the potential for dramatically lowering CO2 emissions. In a recent study, experts identified three trends that, if adopted concurrently, would unleash the full potential of autonomous cars: vehicle automation, vehicle electrification, and ride sharing. By 2050, these "three revolutions in urban transportation" could:

- Reduce traffic congestion (30% fewer vehicles on the road)
- Cut transportation costs by 40% (in terms of vehicles, fuel, and infrastructure)
- Improve walkability and liveability.
- Free up parking lots for other uses (schools, parks, community centre)
- Reduce urban CO2 emissions by 80% worldwide.

8.2 Safety

The safe and responsible introduction of self-driving vehicles. The report recommends introducing a new Automated Vehicles Act, to regulate vehicles that can drive themselves. The proposed Act would do the following:

• Draw a clear distinction between features which just assist drivers, such as adaptive cruise control, and those that are self-driving;

• Introduce a new system of legal accountability once a vehicle is authorised by a regulatory agency as having self-driving features, and a self-driving feature is engaged, including that the person in the driving seat would no longer be a driver but a 'user-in-charge' and responsibility when then rest with the Authorised Self-Driving Entity (ASDE).

• Mandate the accessibility of data to understand fault and liability following a collision.

8.3 The new system of legal accountability for self-driving cars

There are then two categories of self-driving car. First, the 'user in charge' (UIC) which is where a self-driving vehicle is driving, but there is a person in the driving seat, and the vehicle can pass back control to the driver. Second, the 'no user in charge' (NUIC) situation where the vehicle is entirely self-driving, there is no human driver, and only passengers. The report recommends that every NUIC vehicle should be overseen by a licensed NUIC operator, with responsibilities for dealing with incidents and (in most cases) for insuring and maintaining the vehicle

The distinction between aids to assist drivers and genuine 'self-driving' The report recommends a clear distinction be made between automated features that merely assist human drives, for example adaptive cruise control, with those that take over entirely. Then, for example, there would be no change to the legal landscape where a driver crashes while using (non-adaptive) cruise control. Cruise control allows the driver to set the car to maintain a set speed without having to keep the accelerator depressed. It can help with fatigue on long journeys. However, it is still the driver's responsibility to be in control of the vehicle. If a queue of traffic appears on the horizon, it is the driver's job to deactivate the cruise control

(by switching it all or depressing the brake pedal) and bring the car to a safe stop at the end of the traffic queue. If the driver instead ploughs into the back of the queue and causes an accident it is his fault notwithstanding he was using a driver aid. Having the cruise control on was no different to having a foot on the pedal: the driver still remains in control and responsible for keeping watch.

Where, however, the car is genuinely 'self-driving' the position is radically different. The car is not simply 'aiding' the driver, with the driver keeping watch and remaining in control: the car is instead doing the 'thinking' and controlling of all key inputs that, before self-driving, were down to the driver. If a car driving itself makes a mistake, such as by swerving left to avoid an imagined hazard on the right, and thereby crashes into another car on the left when there was in fact no hazard on the right, it would in lay terms seem difficult to point the finger of blame at the driver. The machine had made the mistake and not the driver.

The distinction made by the Law Commissions would seem to be a sensible one based in factual reality: the only legal landscape still applies to human drivers, a different one may be needed for self-driving machines.

8.4 DATA IN SELF DRIVING CARS

Autonomous Vehicles will generate a huge amount of data on location, surroundings, route and systems. The report also recommends that this data will be needed in order to understand fault and liability following a collision and must be accessible. Potentially such data may assist in the event of collisions in being a more precise determining factor than the evidence of drivers or witnesses, which can often be incorrect. With that in mind, fleet staff may need to include data analysts or cybersecurity analysts ensuring data is protected from hackers. Fleet managers face the challenge of ensuring their software systems are both state-of-the-art and constantly updated to mitigate the potential for hacking.

9. Code

9.1 Lane Detection Code

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

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

Mat frame, Matrix, framePers, frameGray, frameThresh, frame-
Edge, frameFinal, frameFinalDuplicate, frameFinalDuplicate1,
ROILane, ROILaneEnd;
int LeftLanePos, RightLanePos, frameCenter, laneCenter, Result,
laneEnd;

RaspiCam_Cv Camera;
stringstream ss;

vector<int> histrogramLane;
vector<int> histrogramLaneEnd;

Point2f Source[] =
{Point2f(69,170),Point2f(300,170),Point2f(30,220),
Point2f(331,220)};
Point2f Destination[] =
{Point2f(100,0),Point2f(300,0),Point2f(100,240),
Point2f(300,240)};

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,46 ) );
    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,0));
}
```

```
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(400,240));
}
void Threshold()
{
     cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
     inRange(frameGray, 105 , 120, frameThresh);
     Canny(frameGray,frameEdge, 900, 1000, 3, true);
     add(frameThresh, frameEdge, frameFinal);
     cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
     cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);
//used in histrogram function only
     cvtColor(frameFinal, frameFinalDuplicate1, COLOR_RGB2BGR);
//used in histrogram function only
}

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

    for(int i=0; i<frame.size().width; i++)       //
frame.size().width = 400
    {
     ROILane = frameFinalDuplicate(Rect(i,100,1,90));
     divide(255, ROILane, ROILane);

     histrogramLane.push_back((int)(sum(ROILane)[0]));
    }

 histrogramLaneEnd.resize(400);
 histrogramLaneEnd.clear();
     for (int i = 0; i <frame.size().width; i++)
     {
          ROILaneEnd = frameFinalDuplicate1(Rect(i, 0, 1, 240));
          divide(255, ROILaneEnd, ROILaneEnd);
```

```
          histrogramLaneEnd.push_back((int)(sum(ROILaneEnd)
[0]));
     }
        laneEnd = sum(histrogramLaneEnd)[0];
        cout<<"Lane END = "<<laneEnd<<endl;
}

void LaneFinder()
{
    vector<int>:: iterator LeftPtr;
    LeftPtr = max_element(histrogramLane.begin()+90, histro-
gramLane.begin() + 190);
    LeftLanePos = distance(histrogramLane.begin(), LeftPtr);

    vector<int>:: iterator RightPtr;
    RightPtr = max_element(histrogramLane.begin() +210, histro-
gramLane.end()-80);
    RightLanePos = distance(histrogramLane.begin(), RightPtr);
    for (int i = 210; i<400 ; i++){
     if (histrogramLane[i] > 0){
     RightPtr = max_element(histrogramLane.begin()+i, histro-
gramLane.begin()+i+10);
     break;
     }}
 line(frameFinal, Point2f(LeftLanePos, 0), Point2f(LeftLanePos,
240), Scalar(0, 255,0), 2);
 line(frameFinal, Point2f(RightLanePos, 0), Point2f(RightLane-
Pos, 240), Scalar(0,255,0), 2);
}
void LaneCenter()
{
    laneCenter = (RightLanePos-LeftLanePos)/2 +LeftLanePos;
    frameCenter = 202;
     line(frameFinal, Point2f(laneCenter,0), Point2f(laneCen-
ter,240), Scalar(0,255,0), 3);
    line(frameFinal, Point2f(frameCenter,0), Point2f(frameCen-
ter,240), Scalar(255,0,0), 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 (laneEnd > 3000)
  {
       digitalWrite(21, 1);
   digitalWrite(22, 1);    //decimal = 7
   digitalWrite(23, 1);
   digitalWrite(24, 0);
   cout<<"Lane End"<<endl;
  }
  if (Result == 0)
  {
   digitalWrite(21, 0);
   digitalWrite(22, 0);    //decimal = 0
   digitalWrite(23, 0);
   digitalWrite(24, 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<<"Right2"<<endl;
  }
      else if (Result >20 && Result < 50)
  {
   digitalWrite(21, 1);
```

```
 digitalWrite(22, 1);    //decimal = 3
 digitalWrite(23, 0);
 digitalWrite(24, 0);
 cout<<"Right3"<<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 && Result>-70)
{
 digitalWrite(21, 0);
 digitalWrite(22, 1);    //decimal = 6
 digitalWrite(23, 1);
 digitalWrite(24, 0);
 cout<<"Left3"<<endl;
}
 else if (Result <-70 )
{
 digitalWrite(21, 1);
 digitalWrite(22, 0);    //decimal = 9
 digitalWrite(23, 0);
 digitalWrite(24, 1);
 cout<<"Left4"<<endl;
}
 else if (Result > 50 )
{
 digitalWrite(21, 0);
 digitalWrite(22, 0);    //decimal = 8
 digitalWrite(23, 0);
 digitalWrite(24, 1);
 cout<<"Right4"<<endl;
}
if (laneEnd > 4000)
 {
   ss.str(" ");
   ss.clear();
```

```
       ss<<" Lane End";
       putText(frame, ss.str(), Point2f(1,50), 0,1,
Scalar(255,0,0), 2);
        }
     else if (Result == 0)
    {
       ss.str(" ");
       ss.clear();
       ss<<"Result = "<<Result<<" Move Forward";
       putText(frame, ss.str(), Point2f(1,50), 0,1,
Scalar(0,0,255), 2);
     }

    else if (Result > 0)
    {
       ss.str(" ");
       ss.clear();
       ss<<"Result = "<<Result<<"bMove Right";
       putText(frame, ss.str(), Point2f(1,50), 0,1,
Scalar(0,0,255), 2);
        }
     else if (Result < 0)
    {
       ss.str(" ");
       ss.clear();
       ss<<"Result = "<<Result<<" Move Left";
       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", framePers);

   namedWindow("Gray", WINDOW_KEEPRATIO);
   moveWindow("Gray", 640, 640 );
    resizeWindow("Gray", 640,480);
    imshow("Gray", ROILane);

    namedWindow("Final", WINDOW_KEEPRATIO);
    moveWindow("Final", 10, 640);
    resizeWindow("Final", 640, 480);
    imshow("Final", frameFinal);
```

```
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;
 }
```

9.1 Object Detection

```
"""Main script to run the object detection routine."""
import argparse
import sys
import time

import cv2
from tflite_support.task import core
from tflite_support.task import processor
from tflite_support.task import vision
import utils
def run(model: str, camera_id: int, width: int, height: int,
num_threads: int,
        enable_edgetpu: bool) -> None:

  """
  Arguments:
    model: Name of the TFLite object detection model.
    camera_id: The camera id to be passed to OpenCV.
    width: The width of the frame captured from the camera.
    height: The height of the frame captured from the camera.
    num_threads: The number of CPU threads to run the model.
    enable_edgetpu: True/False whether the model is a EdgeTPU
model.
  """

 # Variables to calculate FPS
  counter, fps = 0, 0
  start_time = time.time()

  # Start capturing video input from the camera
  cap = cv2.VideoCapture(camera_id)
  cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
  cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)

  # Visualization parameters
  row_size = 20  # pixels
  left_margin = 24  # pixels
  text_color = (0, 0, 255)  # red
  font_size = 1
  font_thickness = 1
  fps_avg_frame_count = 10

  # Initialize the object detection model
  base_options = core.BaseOptions(
      file_name=model, use_coral=enable_edgetpu,
num_threads=num_threads)
  detection_options = processor.DetectionOptions(
      max_results=3, score_threshold=0.3)
```

```
  options = vision.ObjectDetectorOptions(
      base_options=base_options, detection_options=detec-
tion_options)
  detector = vision.ObjectDetector.create_from_options(options)

  # Continuously capture images from the camera and run infer-
ence
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      sys.exit(
          'ERROR: Unable to read from webcam. Please verify
your webcam settings.'
      )

    counter += 1
    image = cv2.flip(image, 1)

    # Convert the image from BGR to RGB as required by the
TFLite model.
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Create a TensorImage object from the RGB image.
    input_tensor = vision.TensorImage.create_from_array(rgb_im-
age)

    # Run object detection estimation using the model.
    detection_result = detector.detect(input_tensor)

    # Draw keypoints and edges on input image
    image = utils.visualize(image, detection_result)

    # Calculate the FPS
    if counter % fps_avg_frame_count == 0:
      end_time = time.time()
      fps = fps_avg_frame_count / (end_time - start_time)
      start_time = time.time()

    # Show the FPS
    fps_text = 'FPS = {:.1f}'.format(fps)
    text_location = (left_margin, row_size)
    cv2.putText(image, fps_text, text_location, cv2.FONT_HER-
SHEY_PLAIN,
                font_size, text_color, font_thickness)

    # Stop the program if the ESC key is pressed.
    if cv2.waitKey(1) == 27:
      break
    cv2.imshow('object_detector', image)
```

```
  cap.release()
  cv2.destroyAllWindows()

def main():
  parser = argparse.ArgumentParser(
      formatter_class=argparse.ArgumentDefaultsHelpFormatter)
  parser.add_argument(
      '--model',
      help='Path of the object detection model.',
      required=False,
      default='efficientdet_lite0.tflite')
  parser.add_argument(
      '--cameraId', help='Id of camera.', required=False,
type=int, default=0)
  parser.add_argument(
      '--frameWidth',
      help='Width of frame to capture from camera.',
      required=False,
      type=int,
      default=640)
  parser.add_argument(
      '--frameHeight',
      help='Height of frame to capture from camera.',
      required=False,
      type=int,
      default=480)
  parser.add_argument(
      '--numThreads',
      help='Number of CPU threads to run the model.',
      required=False,
      type=int,
      default=4)
  parser.add_argument(
      '--enableEdgeTPU',
      help='Whether to run the model on EdgeTPU.',
      action='store_true',
      required=False,
      default=False)
  args = parser.parse_args()

run(args.model, int(args.cameraId), args.frameWidth,
args.frameHeight,
int(args.numThreads), bool(args.enableEdgeTPU))

#Run Code
if __name__ == '__main__':
  main()
```

9.3 Arduino UNO

```
const int EnableL = 6;
const int HighL = 8;       // LEFT SIDE MOTOR
const int LowL =7;

const int EnableR = 3;
const int HighR = 4;       //RIGHT SIDE MOTOR
const int LowR =5;

const int D0 = 10;       //Raspberry pin 21    LSB
const int D1 = 11;       //Raspberry pin 22
const int D2 = 12;       //Raspberry pin 23
const int D3 = 13;       //Raspberry pin 24    MSB

int a,b,c,d,data;


void setup() {

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

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

void Forward()
{
  digitalWrite(HighL, LOW);
```

```
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,100);

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

}

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

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

}

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,70);

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

}

void Left2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,50);
```

```
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,120);

}

void Left3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,30);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,150);

}

void Left4()
{
  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  analogWrite(EnableL,50);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,130);

}

void Right1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,100);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,70);

}

void Right2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,120);

  digitalWrite(HighR, LOW);
```

```
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,50);

}

void Right3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,150);

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,30);

}

void Right4()
{
  delay(300);
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,120);

  digitalWrite(HighR, HIGH);
  digitalWrite(LowR, LOW);
  analogWrite(EnableR,50);

}

void UTurn()
{
  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(200);

  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);    //forward
  delay(1250);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(200);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);   //   left
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
```

```
  delay(400);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(200);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);   // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  delay(1100);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(200);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);    //left
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  delay(400);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  delay(800);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 100);
  analogWrite(EnableR, 100);
  delay(500);
}

void loop()
{
  Data();
  if(data==0)
   {
     delay(300);
     Forward();
   }

  else if(data==1)
   {
```

```
    delay(300);
     Right1();
   }

  else if(data==2)
   {delay(300);
     Right2();
   }

  else if(data==3)
   {delay(300);
     Right3();
   }

  else if(data==4)
   {delay(300);
     Left1();
   }

  else if(data==5)
   {delay(300);
     Left2();
   }

  else if(data==6)
   {delay(300);
     Left3();
   }

  else if (data==7)
   {
     Stop();
   }

  else if (data>10)
   {
     Stop();
   }
  else if (data==8)
   {
     Right4();
   }
  else if (data==9)
   {
     Left4();
   }

}
```

10. Conclusion and Future scope

10.1. Conclusion

In this report, a method to make a self driving robot car is presented. The different hardware components and their assembly are clearly described. A novel method to determine the uneven, marked or unmarked road edges is explained in details relying upon OpenCV. Using object detection, the collisions with obstacles is avoided and lane detection is performed by using camera module (open CV). The algorithm mentioned in the paper has been successfully implemented on a small autonomous car. We addressed the problem of non-autonomous vehicles with the proposed system which reduces the human work of operating the vehicle. Furthermore, we also notice that the given system performance is much better than an average user. Since the performance is better and always consistence, we hereby come to a conclusion that the proposed system can solve the basic human error that occurs.

The autonomous car would surely prove out to be a boom in the automation industry and would be preferred over many traditional techniques. They could be used for patrolling and capturing the images of the offender. As they won't require any drivers, the accidents caused by the carelessness of the goods carrier vehicles would be reduced and would ensure better logistic flow. Buses for public transport would be more regulated due to minimal errors. Hence, due to greater autonomous nature and efficiency, an autonomous car of this nature can be practical and is highly beneficial for better regulation in the goods and people mover's section.

10.2 Future Scope

- Future work that can be added to this project may be the development of a web app. Here the user can operate when the vehicle encounters two pathways to reach common destination the user can interact though a web app. Also, the user can get suggestion of nearby places to visit also though this app using ESP32.

- To enhance the performance and ensure practicality of the car, the efficiency and processor speed need to be raised. A camera of better resolution would also be required as the scenes keep changing rapidly in the real world. Also the speed of the car should decrease gradually so that the passengers aren't hurt and the goods aren't damaged.

- As in our model power consumption is high ,so we need motors with low rpm. The very simple formula is Power = Torque (load) x rpm. So if your load is 1000 N-m and the speed you want is 48 rpm, power would be 48,000 units; if load is 10 N-m and the speed required is 1,760 rpm, power would be 17,600 units. OK lets say you want to move your load of 1000 N-m at just 0.1 rpm then your power requirement would be of 100 units. Of course the selection of a motor goes far beyond motor HP, voltage.So it is better if you have power at lower RPMs rather than at higher RPMs. More power at lesser RPM indicate better efficiency of model.

- If we are talking about future enhancement then apart from automatic parking, GPS can be added to the system for avoiding traffic congestion and time can be saved. Emergency notifying systems can be implemented in case the passenger is not well or suffering from sudden illness.

- For further improvement in our model, we should use Lithium ion batteries for Automated Guided Vehicles because efficiency increases upto ~95%.

11. Reference

1. Anand Y, Rahul Ajithkumar Autonomous Car with Swarm Intelligence, 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT),2019.

2. Raj Shirolkar and Rohan Datar In International Journal of Engineering Research & Technology (IJERT) on May-2019 Self-Driving Autonomous Car using Raspberry Pi.

3. Aditya Kumar Jain Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino, 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018).

4. Shruti P Path Planning for Autonomous Car, 2019 Amrita Vishwa Vidyapeetham India.

5. Qudsia Menom Self-Driving and Driver Relaxing Vehicle, 2016 Mehraan University of Engineering & Technology, Jamshoro Sindh.

6. Naohiro Nakamoto Development of an Open-source Educational and Research platform for Autonomous, 2019 Graduate School of Robotics &design Osaka institute Of Technology Osaka, Japan.

7. Duc Lich Luu, Ciprian Lupu, Doinita Chirita Design and Development of Smart Cars Model for Autonomous Vehicles in a Platooning, 15th International Conference on Engineering of Modern Electric Systems (EMES),2019.

8. Paul George Parakkal, Sajith Variyar V V GPS Based Navigation System for Autonomous Car, Centre for Computational Engineering and Networking (CEN),2017Amrita School of Engineering, Coimbatore.

9. Syed Riaz un-Nabi Jafri, Syed Minhaj un-Nabi Jafri, Syed Zeeshan Shakeel Intelligent Navigation of Unmanned Land Vehicle by using GPS & One ABS Sensor, Proceedings of the 4th International Conference on Autonomous Robots and Agents, Feb 10-12, 2019, Wellington, New Zealand.

10. Myeon-gyun Cho A study on the obstacle recognition for autonomous driving RC cars using lidar and thermal infrared camera, School of Information and Communication,2019 Semyung University, Korea.

11. Bhaskar Barua A self-driving car implementation using computer vision for detection and navigation, Information technology,2019 Sardar Patel institute of technology Mumbai, India.

12. Moongu Jeon Multiple objects Tracking using Radar for Autonomous Driving, School of Electrical Engineering and Computer Science,2020Gwangju Institute of Science and Technology, Korea.

13. https://www.tensorflow.org/learn