ADAPTIVE DRIVING ASSISTANCE SYSTEM

Submitted by

Aakash Patwa
(Enroll No.: IU1941091032)
Kritarth Chauhan
(Enroll No.: IU1841090006)
Vandit Patel
(Enroll No.: IU1841090013)

In partial fulfillment of the requirement for the degree of
Bachelor of Technology
in the
Department of Electronics and Communication Engineering

Under the Guidance of

Miloni Ganatra
Assistant Professor
Electronics and Communication Department
IITE,Indus University

INDUS INSTITUTE OF TECHNOLOGY AND ENGINEERING
INDUS UNIVERSITY

April, 2022

ADAPTIVE DRIVING ASSISTANCE SYSTEM

Submitted by

Aakash Patwa
(Enroll No.: IU1941091032)
Kritarth Chauhan
(Enroll No.: IU1841090006)
Vandit Patel
(Enroll No.: IU1841090013)

In partial fulfillment of the requirement for the degree of
Bachelor of Technology
in the
Department of Electronics and Communication Engineering

Under the Guidance of

Miloni Ganatra
Assistant Professor
Electronics and Communication Department
IITE,Indus University

INDUS INSTITUTE OF TECHNOLOGY AND ENGINEERING
INDUS UNIVERSITY

April, 2022

DECLARATION

By the B.Tech Student

I hereby declare that the Report of the Under Graduate Project Work entitled ***ADAPTIVE DRIVING ASSISTANCE SYSTEM*** which is being submitted to the **IITE, Indus University**, in partial fulfillment of the requirements for the award of the Degree of **Bachelor of Technology** in **Electronics and Communication Engineering** in the department of Electronics and Communication Engineering, *is a bonafide report of the work carried out by us.* The material contained in this Report has not been submitted to any University or Institution for the award of any degree.

Aakash Patwa
Department of Electronics and Communication Engineering

Kritarth Chauhan
Department of Electronics and Communication Engineering

Vandit Patel
Department of Electronics and Communication Engineering

Place: IITE, Indus University.
Date: 22/4/22

BONAFIDE CERTIFICATE

This is to certify that the U.G. project work report entitled **ADAPTIVE DRIVING ASSISTANCE SYSTEM** submitted by **Aakash Patwa**(Enrollment Number: IU1941091032) as the record of the work carried out by him, is accepted as the U.G. project work report submission in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **Electronics and Communication Engineering** in the Department of **Electronics and Communication Engineering of IITE, Indus University** during the academic year 2021-2022.

(Prof. Miloni Ganatra)
Project Guide

(Dr. Vrushank Shah)
HOD, EC

BONAFIDE CERTIFICATE

This is to certify that the U.G. project work report entitled **ADAPTIVE DRIVING ASSISTANCE SYSTEM** submitted by **Kritarth Chauhan**(Enrollment Number: IU1841090006) as the record of the work carried out by him, is accepted as the U.G. project work report submission in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **Electronics and Communication Engineering** in the Department of **Electronics and Communication Engineering of IITE, Indus University** during the academic year 2021-2022.

(Prof. Miloni Ganatra)
Project Guide

(Dr. Vrushank Shah)
HOD, EC

BONAFIDE CERTIFICATE

This is to certify that the U.G. project work report entitled **ADAPTIVE DRIVING ASSISTANCE SYSTEM** submitted by **Vandit Patel**(Enrollment Number: IU1841090013) as the record of the work carried out by him, is accepted as the U.G. project work report submission in partial fulfillment of the requirements for the award of degree of **Bachelor of Technology** in **Electronics and Communication Engineering** in the Department of **Electronics and Communication Engineering of IITE, Indus University** during the academic year 2021-2022.

(Prof. Miloni Ganatra)
Project Guide

(Dr. Vrushank Shah)
HOD, EC

ACKNOWLEDGEMENTS

We consider it a privilege to be associated with Indus Institute of Technology and Engineering, Ahmedabad in this academic endeavor. We owe special debt of gratitude to our Project Guide **Mrs. Miloni Ganatra, Assistant Professor, Department of Electronics and Communication**, Indus University, Ahmedabad for her constant support and guidance throughout the course of our work. It is only her cognizant efforts that our endeavors have seen light of the day.

Our deepest thanks to our Project Coordinator **Dr. Minesh Thaker, Assistant Professor, Department of Electronics and Communication**, Indus University, Ahmedabad for providing his invaluable inputs and guiding us with attention and care.

We also take the opportunity to acknowledge the contribution of **Dr. Vrushank Shah, Head Of Department of Electronics and Communication**, Indus University, Ahmedabad for his full support and assistance during the development of the project.

We are also thankful to all staff members of the department for their kind support at various stages of the work. It is beyond the reach of our language to express our sincere gratitude to our parents for their infinite inspiration and sacrifices towards the benefit of our academic career.

Aakash Patwa **Kritarth Chauhan** **Vandit Patel**

ABSTRACT

Great changes have taken place in automation and machine vision technology in recent years. Meanwhile, the demands for driving safety, efficiency, and intelligence have also increased significantly. Other factors, such as government regulations associated with safety increasing the prevalence of safe driving experience, can decrease traffic congestion. Adaptive Driver Assistance Systems are intelligent systems that reside inside the vehicle and assist the main driver in a variety of ways. These systems may be used to provide vital information about traffic, closure and blockage of roads ahead, congestion levels, suggested routes to avoid congestion etc. These systems may also be used to judge the fatigue and distraction of the human driver and thus make precautionary alerts or to assess the driving performance and make suggestions regarding the same. The greatest advantage of using the assistance systems is that they enable communication between different parts of vehicles and vehicle infrastructure systems. This enables exchange of information for better vision, planning and decision making of the vehicles. The project successfully demonstrates the functioning and working of automated headlights, lane correction, braking assistance, traffic signal identification, multiple object detection and maneuvering around obstacles, all these is achieved with the aid of combination of sensors and control units by establishing clear communication in the created network.

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Project Introduction

Vehicle safety can be classified into two types, Active safety and Passive safety. Passive safety relies on certain devices, such as safety belts, airbags, and bumpers, to protect passengers and reduce damage. However, passive safety cannot improve driving safety by itself because 93% of the traffic accidents are caused by the driver's lack of awareness of the danger. Consequently, active safety which is developed to sense and predict dangerous situations has been considered an important part of modern vehicles.

By exchanging data with other devices active safety modules can assist drivers in making decisions based on the overall traffic. If the risk is high and might cause a collision, the vehicle can warn the driver to correct the driving behavior, and in urgent cases, the active safety modules can take over the control of the vehicle to avoid a traffic accident. The latest active safety modules have achieved the identification of traffic signs by applying deep machine learning technology. As a result, a vehicle could recognize a traffic warning or limitation.

An adaptive Driving Assistance system is a group of electronic technologies that assist drivers in driving functions. Through a safe human-machine interface, it increases car and road safety. It uses automated technology, such as sensors and cameras, to detect nearby obstacles, and respond accordingly. Compared with traditional transportation, Adaptive Driving Assistance System is superior in ensuring passenger safety and improving driving control. [3] [4] [5]

1.2 Objectives

- ADAS employs a variety of sensors in conjunction with machine learning to aid drivers in avoiding errors in judgement, which in turn provides better passenger safety.
- It has a single driver support system that provides braking and steering assistance.
- The vehicle will come to a halt if an obstruction is detected.
- Headlights will be controlled based on the amount of outdoor light available, for better visibility for the driver.
- The turn indicators will be turned on, before changing lanes.
- It will be making decisions based on traffic signal status detection.

1.3 Related Works

O. Gietelink Et Al. in the paper "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simula- tions" presented the new VEHIL concept for testing ADASs, where a real in-telligent vehicle is operated in a HIL environment. VEHIL is suitable for various typesof ADASs: ACC, stop-and-go, FCW, pre-crash systems, blind spot systems, and fully au-tonomous vehicles. With test results for ACC and FCW it was demonstrated that VEHILhas an added value in several phases of the development process of an ADAS: sensor verification; rapid control prototyping; model validation; function level validation; fine-tuningof control algorithms; production sign-off tests; and preparation of test drives. [6]

H. Inoue in "Research into adas with driving intelligence for future innovation" suggested to reduce the amount of traffic accidents caused by elderly drivers, we focused on a proactive steering intervention system in a human-machine shared framework. To investigate the reaction tendencies of elderly drivers, we conducted an experiment using a driving simulator. [7]

M. Hasenj ager and H. Wersing in "Personalization in advanced driver assistance systems and autonomous vehicles: A review" assesses the current state of the art ofpersonalization in advanced driver assistance systems and inautonomous driving. The primary goal in the personalizationof ADAS is to improve the usability and hence the driveracceptance of the systems. This is especially important insafety relevant applications where alerts and their timingshould be adapted to the driver's skills and needs in order toprevent disuse of the system. [8]

T. Yoshida Et Al. in "Adaptive driver-assistance sys- tems" suggested that the current state of development regarding adaptive driver-assistance systems, this report explained the status of Hitachi Group's developments concerning these systems and then went on to summarize their future outlook. It is expected that adaptive driver-assistance systems will contribute to the reduction of traffic accidents; moreover, if the aging of society is taken into consideration, it is thought that such systems will become even more essential features of vehicles in the future. [9]

Chapter 2

Hardware

2.1 Lithium-Ion Power Source

Lithium ion batteries are rechargeable batteries that are characterized by very high power densities. Such batteries have become very commonplace: from everyday electronic products such as cell phones to electric vehicles.

They're generally much lighter than other types of rechargeable batteries of the same size. The electrodes of a lithium-ion battery are made of lightweight lithium and carbon. Lithium is also a highly reactive element, meaning that a lot of energy can be stored in its atomic bonds. This translates into a very high energy density for lithium-ion batteries. Here is a way to get a perspective on the energy density.

A typical lithium-ion battery can store 150 watt-hours of electricity in 1 kilogram of battery. A NiMH (nickel-metal hydride) battery pack can store perhaps 100 watt-hours per kilogram, although 60 to 70 watt-hours might be more typical. A lead-acid battery can store only 25 watt-hours per kilogram. Using lead-acid technology, it takes 6 kilograms to store the same amount of energy that a 1 kilogram lithium-ion battery can handle. That's a huge difference.

They hold their charge. A lithium-ion battery pack loses only about 5 percent of its charge per month, compared to a 20 percent loss per month for NiMH batteries. They have no memory effect, which means that you do not have to completely discharge them before recharging, as with some other battery chemistries. Lithium-ion batteries can handle hundreds of charge/discharge cycles.

2.2 DC Geared Motors

A Direct Current (DC) motor is a rotating electrical device that converts direct current, of electrical energy, into mechanical energy. An Inductor (coil) inside the DC motor produces a magnetic field that creates rotary motion as DC voltage is applied to its terminal. Inside the motor is an iron shaft, wrapped in a coil of wire. This shaft contains two fixed, North and South, magnets on both sides which causes both a repulsive and attractive force, in turn, producing torque.

A geared DC Motor has a gear assembly attached to the motor. The speed of the motor is counted in terms of rotations of the shaft per minute and is termed RPM. The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure.

This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction. This Insight will explore all the minor and major details that make the gear head and hence the working of geared DC motor.

Figure 2.1: DC Geared Motor

The main characteristics of these gear motors are miniature form factors, offering significant strength, torque, and other technical capabilities that these applications require. Their linear performance characteristics make them suitable for many applications requiring a controlled performance. Their linear performance characteristics make them suitable for many applications requiring a controlled performance.

2.3 Light Dependent Resistor

A photoresistor or light dependent resistor is an electronic component that is sensitive to light. When light falls upon it, then the resistance changes. Values of the resistance of the LDR may change over many orders of magnitude the value of the resistance falling as the level of light increases.

It is not uncommon for the values of resistance of an LDR or photoresistor to be several megohms in darkness and then to fall to a few hundred ohms in bright light.

LDRs are very different to other forms of resistor like the carbon film resistor, metal oxide film resistor, metal film resistor and the like that are widely used in other electronic designs. They are specifically designed for their light sensitivity and the change in resistance this causes.

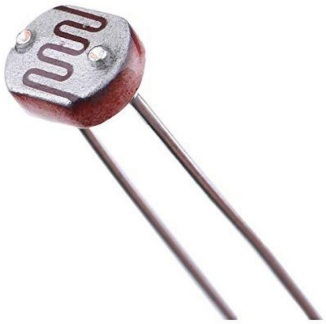


Figure 2.2: Light Dependent Resistor

Although other electronic components such as photodiodes or photo-transistor can also be used, LDRs or photo-resistors are a particularly convenient to use in many electronic circuit designs. They provide large change in resistance for changes in light level.

2.4 L298N Motor Driver

This L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

The L298N Motor Driver module consists of an L298 Motor Driver IC, 78M05 Voltage Regulator, resistors, capacitor, Power LED, 5V jumper in an integrated circuit. 78M05 Voltage regulator will be enabled only when the jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator and the 5V pin can be used as an output pin to power the micro-controller. The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

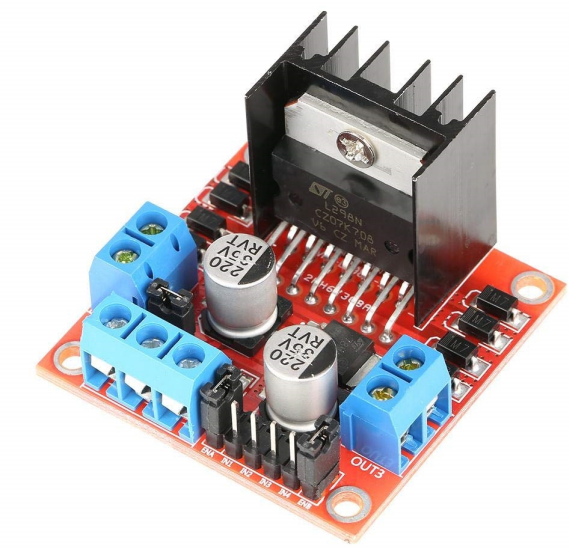


Figure 2.3: L298N Motor Driver

Features	Specifications
Driver Model	L298N 2A
Driver Chip	Double H Bridge L298N
Motor Supply Voltage (Maximum)	46V
Motor Supply Current (Maximum)	2A
Logic Voltage	5V
Driver Voltage	5-35V
Driver Current	2A
Logic Current	0-36 mA
Maximum Power (W)	25 W
Power	On LED indicator

Table 2.1: Technical Specifications of L298N

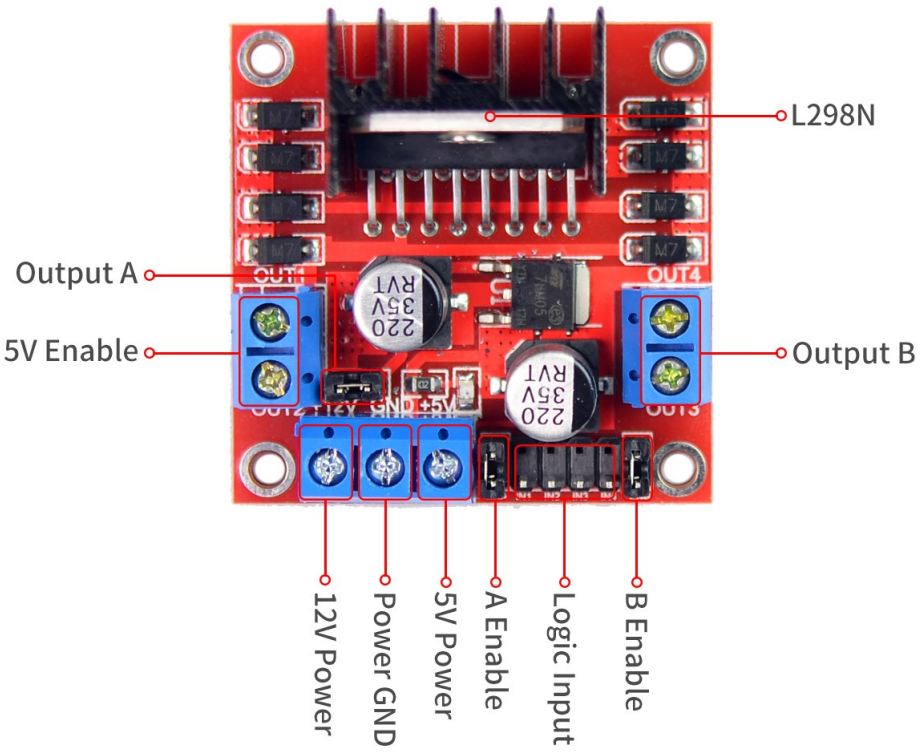


Figure 2.4: L298N Pin Configuration

Pin Name	Description
IN1 and IN2	Motor A input pins. Used to control the spinning direction of Motor A
IN3 and IN4	Motor B input pins. Used to control the spinning direction of Motor B
ENA	Enables PWM signal for Motor A
ENB	Enables PWM signal for Motor B
OUT1 and OUT2	Out pins of Motor A
OUT3 and OUT4	Out pins of Motor B
12V	12V input from DC power Source
5V	Supplies power for the switching logic circuitry inside L298N IC
GND	Ground pin

Table 2.2: Pin Configuration of L298N

The L298N motor controller follows the H-bridge configuration, which is handy when controlling the direction of rotation of a DC motor.

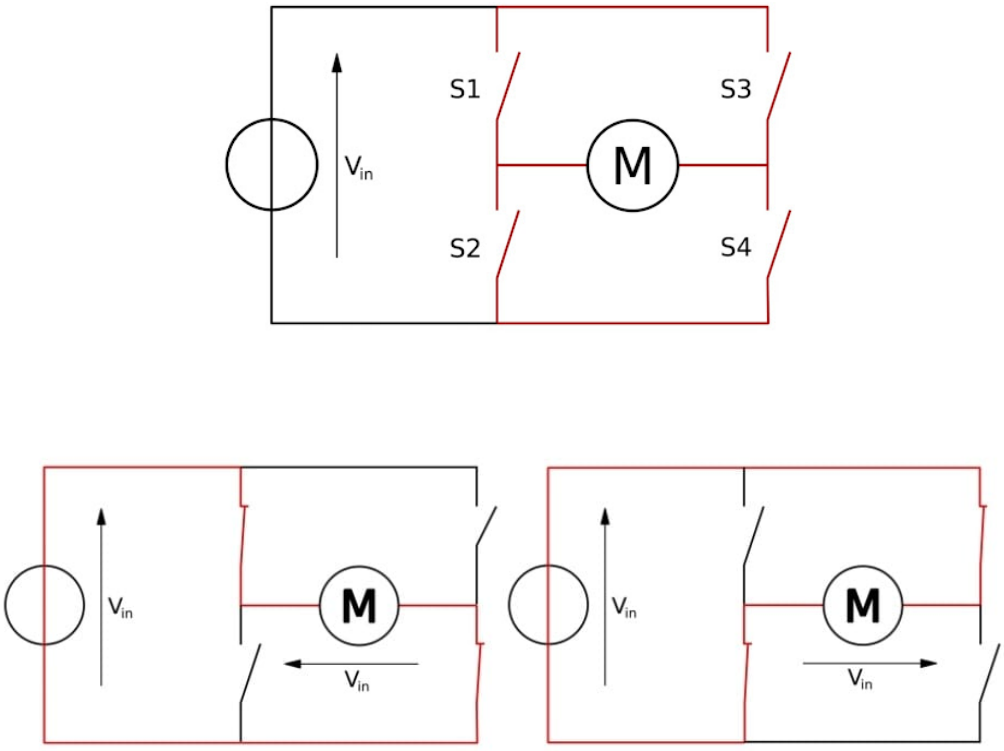


Figure 2.5: H-bridge Schematic

H-bridge Here, the motor rotates in the direction dictated by the switches. When S1 and S4 are on, the left motor terminal is more positive than the right terminal, and the motor rotates in a certain direction. On the other hand, when S2 and S3 are on, the right motor terminal is more positive than the left motor terminal, making the motor rotate in the other direction.

H-bridge left and right The other benefit of using an H-bridge is that you can provide a separate power supply to the motors. This is very significant especially when using an Arduino board where the 5V power source is simply not enough for two DC motors.

The motor driver IC drives two motors through two channels, A and B. For example, if a motor is using channel A, its terminals must be connected to pins Out 1 and Out 2. The Enable A pin must be high to turn on the motor. To drive a motor to a direction, say, clockwise, the pin Input 1 must be high while the pin Input 2 must be low. To drive the motor counter clockwise, the pin Input 1 is low while the pin Input 2 is high.

2.5 Arduino

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced with various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), and 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by a USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts.

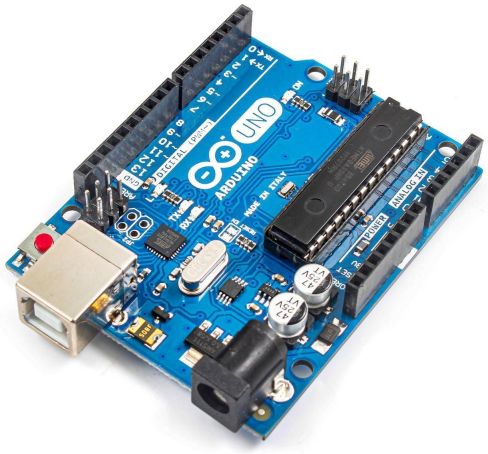


Figure 2.6: Arduino Uno

Table 2.3: Technical Specifications of Arduino

Microcontroller	ATmega328P - 8 bit AVR family microcontroller
Operating Voltage	5V
Recommended Input Voltage	7-12V
Input Voltage Limits	6-20V
Analog Input Pins	6 (A0 - A5)
Digital I/O Pins	14 (Out of which 6 provide PWM output)
DC current on I/O Pins	40 mA
DC current on 3.3V Pin	50 mA
Flash Memory	32 KB (0.5KB is used for Boot Loader)
SRAM	2 KB
EEPROM	1 KB
Frequency (Clock Speed)	16 MHz

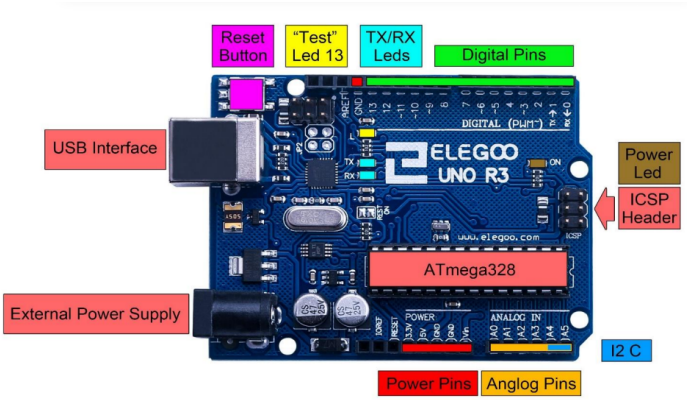


Figure 2.7: Arduino Uno Pin Configuration

Pin Category	**Pin Name**	**Details**
Power	Vin, 3.3V, 5V, GND	Vin: Input voltage to Arduino when using an external power source. 5V: Regulated Power Supply sued to power micro-controller and other components on the board. 3.3V: Maximum current drawn is 50 mA.
Reset	Reset	Resets the micro-controller
Analog Pins	A0-A5	Provides analog input in the range of 0-5V
Input/Output Pins	Digital Pins 0-13	Can be used as input or output pins.
Serial	0(Rx), 1(Tx)	Receive and transmit TTL serial data.
External Interupts	2,3	To trigger an interupt.
PWM	3, 5, 6, 9, 11	Provides 8-bit PWM output.
SPI	10 (SS), 11 (MOSI), 12 (MISO) and 13 (SCK)	Used for SPI communication.
Inbuilt LED	13	To turn on the inbuild LED.
TWI	A4 (SDA), A5 (SCA)	Used for TWI communication.
AREF	AREF	Provides reference voltage for input voltage.

Table 2.4: Pin Description of Arduino

2.6 RaspiCam

The Raspberry Pi Camera Module v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280 x 2464 pixel static images, and also supports 1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSi interface, designed especially for interfacing to cameras.

The board itself is tiny, at around 25mm x 23mm x 9mm. It also weighs just over 3g, making it perfect for mobile or other applications where size and weight are important.

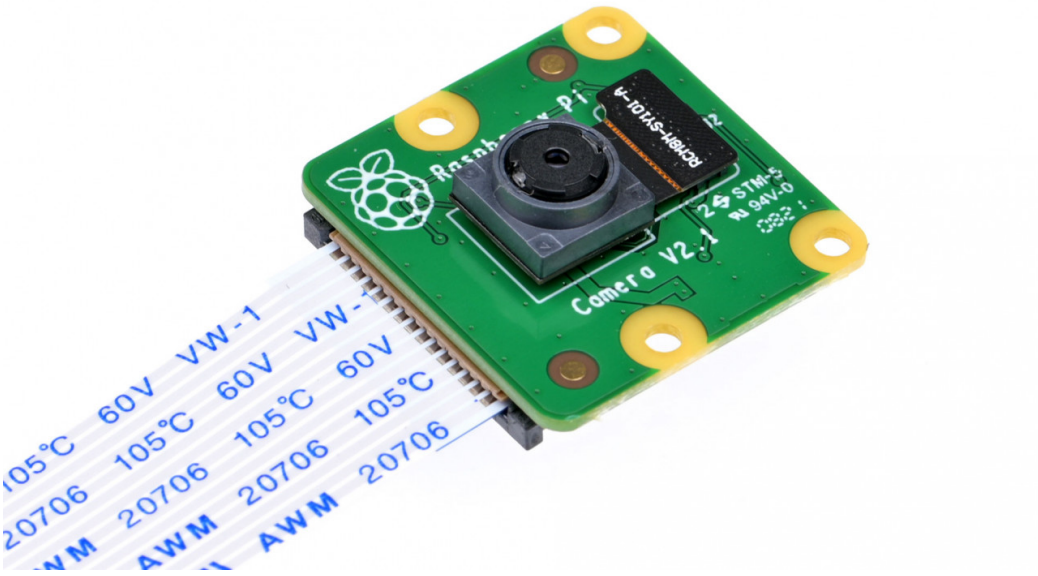


Figure 2.8: RaspiCam Module

Connecting the camera

The flex cable inserts into the connector situated between the Ethernet and HDMI ports, with the silver connectors facing the HDMI port. The flex cable connector should be opened by pulling the tabs on the top of the connector upwards then towards the Ethernet port. The flex cable should be inserted firmly into the connector, with care taken not to bend the flex at too acute an angle. The top part of the connector should then be pushed towards the HDMI connector and down, while the flex cable is held in place

Still Resolution	8 Megapixels
Video Modes	1080p30, 720p60 and 640 × 480p60/90
Linux Integration	V4L2 driver available
C programming API	OpenMAX IL and others available
Sensor	Sony IMX219
Sensor resolution	3280 × 2464 pixels
Sensor image area	3.68 x 2.76 mm (4.6 mm diagonal)
Pixel Size	1.12 µm x 1.12 µm
Optical size	1/4”
Depth of field	adjustable with supplied tool
Focal length	3.04 mm
Horizontal field of view	62.2 degrees
Vertical field of view	48.8 degrees
Focal ratio (F-Stop)	2.0

Table 2.5: Hardware Specification of RaspiCam

2.7 Raspberry Pi 4B

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with significantly enhanced CPU, GPU, and I/O performance; all within a similar form factor, power envelope, and cost as the previous generation Raspberry Pi 3B+. The Pi4B is available with either 1, 2, or 4 Gigabytes of LPDDR4 SDRAM.

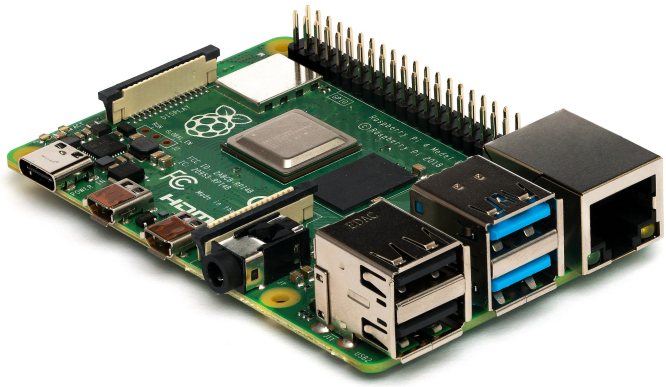


Figure 2.9: Raspberry Pi 4B

Features	
Hardware	• Quad core 64-bit ARM-Cortex A72 running at 1.5GHz • 1, 2 and 4 Gigabyte LPDDR4 RAM options • H.265 (HEVC) hardware decode (up to 4Kp60) • H.264 hardware decode (up to 1080p60) • VideoCore VI 3D Graphics • Supports dual HDMI display output up to 4Kp60

Interfaces	• 802.11 b/g/n/ac Wireless LAN • Bluetooth 5.0 with BLE • 1x SD Card • 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution • 2x USB2 ports • 2x USB2 ports • 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT) • 1x Raspberry Pi camera port (2-lane MIPI CSI) • 1x Raspberry Pi display port (2-lane MIPI DSI) • 28x user GPIO supporting various interface options
Software	• ARMv8 Instruction Set • Mature Linux software stack • Actively developed and maintained

Table 2.6: Features of Raspberry Pi 4B

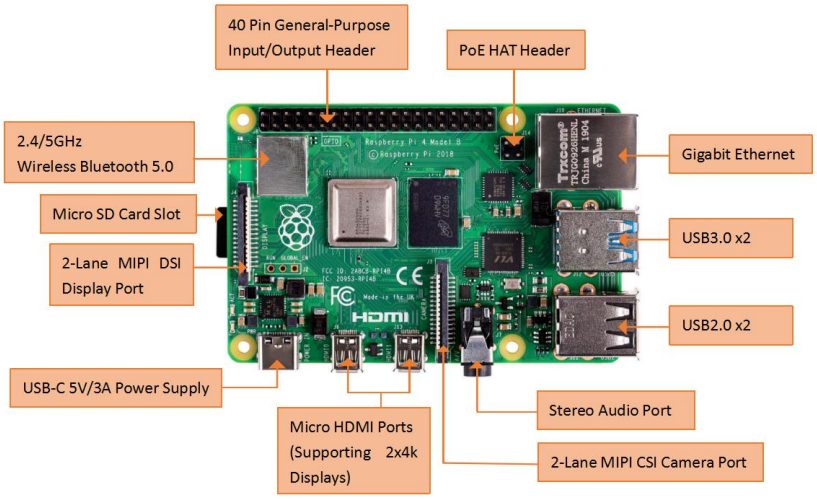


Figure 2.10: Raspberry Pi Pin Configuration

Peripherals	
GPIO Interface	The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.
Display Parallel Interface (DPI)	A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.
SD/SDIO Interface	The Pi4B has a dedicated SD card socket which suports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins
Camera and Display Interfaces	The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.
USB	The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.
HDMI	The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.
Audio and Composite (TV Out)	The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack. The analog audio output can drive 32 Ohm headphones directly

Table 2.7: Peripherals of Raspberry Pi 4B

Chapter 3

Key Hardware Implications

3.1 Power Distribution Circuit

The power distribution circuit which ensures that every circuit connected to it gets the power from the lithium-ion power source on-demand was designed based on the following 3 points:-

- Proper Voltage – The voltage provided by the lithium-ion power source should remain the same throughout the different connections provided for connecting different loads.
- Availability of Power on Demand – The DC power must be available to the connected circuits that they may require from time to time.
- Reliability – The variations in voltage at the connected terminals should be as low as possible so that the connected circuit functions without any power disruptions, which in turn provides reliability.

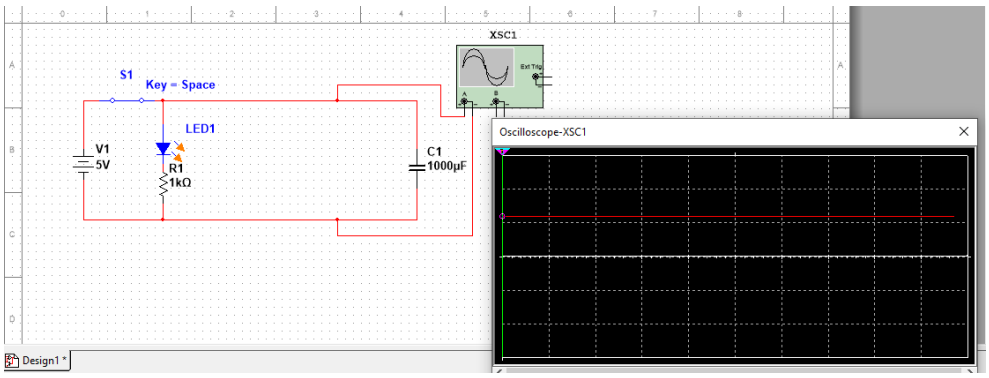


Figure 3.1: Power Distribution

3.2 Turn Indicators

A car's turn signal exists for a reason. Yet, many drivers hardly or never use their signals, which is a mistake.

Turn signals let other drivers know your intentions to turn or change lanes. If you don't signal those intentions, others naturally assume you will continue on as you are. If you don't use your signal, you could cause an accident with other drivers or pedestrians.

This is why we have used automatic turn indicators which turn ON, on sensing a turn with respect to the turning direction, in our project.

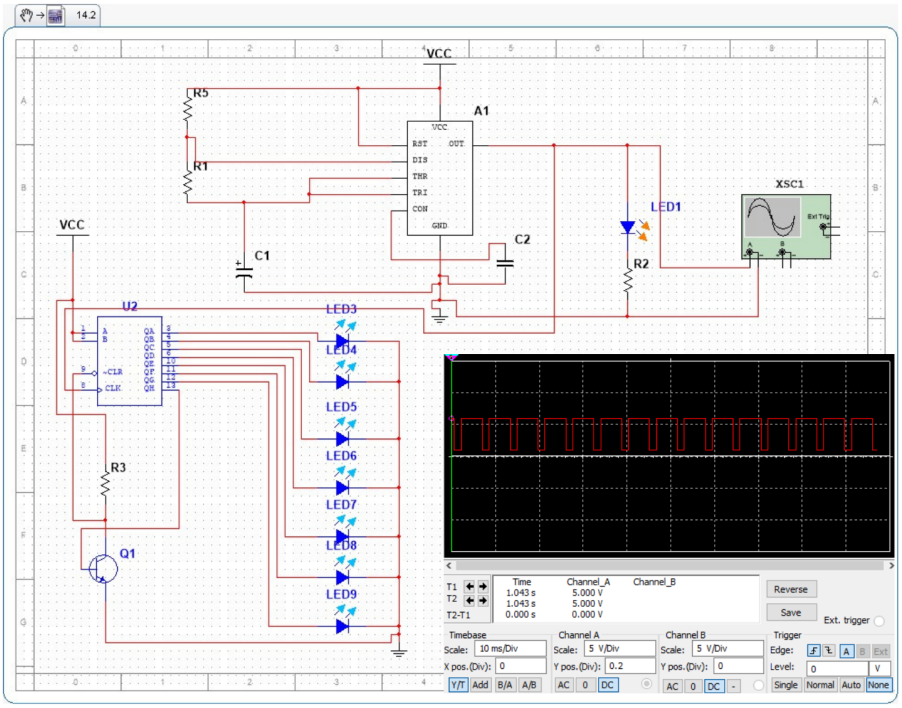


Figure 3.2: Turn Indicators Simulation

The above mentioned turn indicator circuit is designed by taking the output from a delay timer circuit which is made by using a 555 timer IC and connecting a 74164 IC which is basically a serial in parallel out shift register IC, to provide each LED with a delay while turning it ON which in turn provides us with visually appealing turn indicators.

Software Interface

```
const int LeftSignal = 4;
const int enableLeft = 11;
const int RightSignal = 12;
const int enableRight = 13;
void setup()
{
  Serial.begin(9600);
  pinMode(LeftSignal,OUTPUT);
  pinMode(enableLeft,OUTPUT);
  pinMode(RightSignal,OUTPUT);
  pinMode(enableRight,OUTPUT);
}
void loop()
{
  digitalWrite(LeftSignal,HIGH);
  digitalWrite(enableLeft,LOW);
  digitalWrite(RightSignal,LOW);   //Left Indicators
  digitalWrite(enableRight,LOW);
  delay(3000);
  digitalWrite(LeftSignal,LOW);
  digitalWrite(enableLeft,LOW);
  digitalWrite(RightSignal,HIGH);   //Right Indicators
  digitalWrite(enableRight,LOW);
  delay(3000);
  digitalWrite(LeftSignal,HIGH);
  digitalWrite(enableLeft,HIGH);
  digitalWrite(RightSignal,HIGH);   //Brake
  digitalWrite(enableRight,HIGH);
  delay(3000);
  digitalWrite(LeftSignal,LOW);
  digitalWrite(enableLeft,LOW);
  digitalWrite(RightSignal,LOW);   //Signal OFF
  digitalWrite(enableRight,LOW);
  delay(3000);
}
```

3.3 LDR Controlled Headlights

As per the circuit diagram, we have made a voltage divider circuit using LDR and a 100k resistor. The voltage divider output is fed to the analog pin of the Arduino. The analog pin senses the voltage and gives some analog value to Arduino. The analog value changes according to the resistance of LDR. So, as the light falls on the LDR the resistance of it get decreased, and hence the voltage value increase.

The intensity of light ↓ - Resistance↑ - Voltage at the analog pin↓ - Light turns ON

As per the Arduino code, if the analog value falls below 700 we consider it dark and the light turns ON. If the value comes above 700 we consider it bright and the light turns OFF.

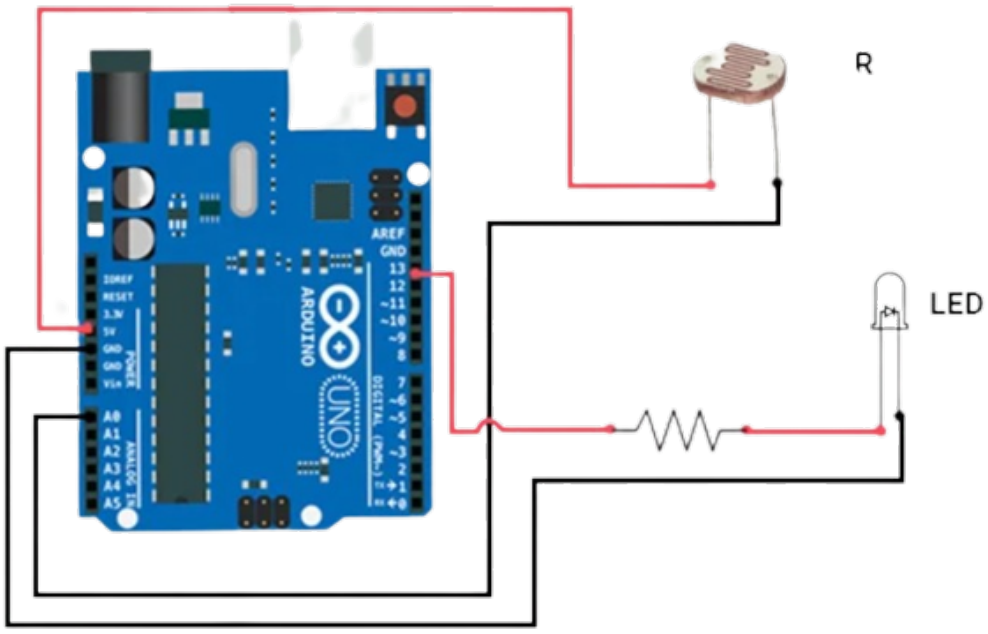


Figure 3.3: LDR Controlled Headlights Connections

Software Interface

```
int LED = 9;
int LDR = A0;
pinMode(LED, OUTPUT);
pinMode(LDR, INPUT);
int LDRValue = analogRead(LDR);

if (LDRValue <=700)
{
digitalWrite(LED, HIGH);
Serial.println("It's Dark Outside; Lights status: ON");
}
else
{
digitalWrite(LED, LOW);
digitalWrite(relay, LOW);
Serial.println("It's Bright Outside; Lights status: OFF");
}
```

Chapter 4

Project Methodology

4.1 Block Diagram

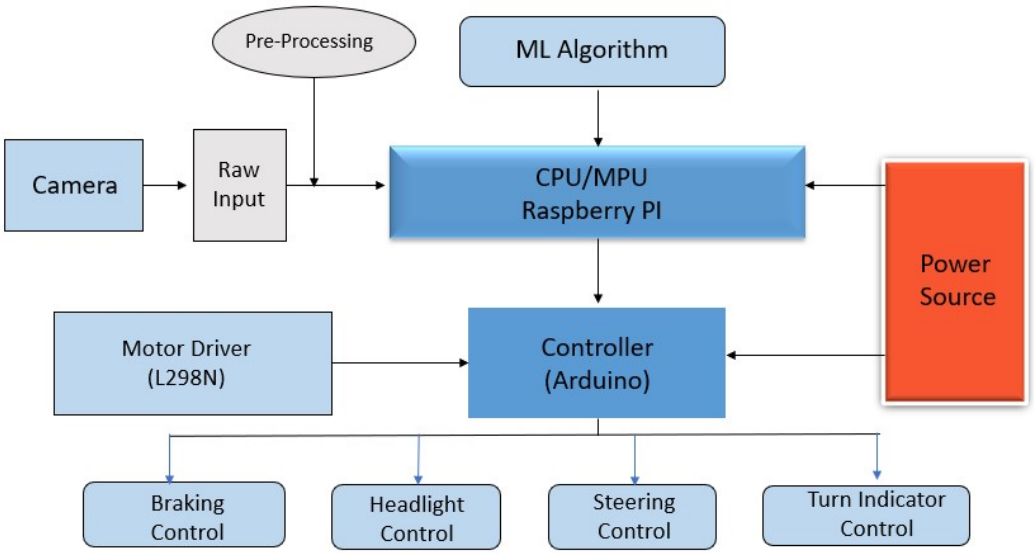


Figure 4.1: Basic Block Diagram

4.2 Block Diagram Description

Power Source: 5V DC Supply, to power up the devices connected with it.

Camera: The camera acts up as the primary image sensor, which provides raw image/video to the raspberry pi.

Pre-processing: Pre-Processing is performed on the image/video from the camera to fine-tune it and to provide as clear an image/video as possible to the raspberry pi for better extraction of data.

CPU/MCU Raspberry pi: This block can be referred to as the mind of the whole project as all the processing and controlling is done by this device, may it be image processing, video processing, or passing on the commands to the connected controller. It acts as a master device in our project.

Machine learning algorithm: An algorithm to detect objects like a car, a stop sign or even a traffic signal, is integrated with raspberry pi for training purposes. This data is further implied for triggering the slave device (Arduino) for the driving the motors with respect to the processed input provided by the raspberry pi.

Arduino: It simply acts as slave device. The Arduino Uno here, is configured to communicate with the raspberry pi and take up the controlling commands and to execute them, in line with motor driver controls, led headlights, and turn indicators/tail lights.

Motor Driver: The motor driver, connected with the Arduino is to provide steering and braking controls as required or demanded by the Raspberry pi, via Arduino.

Chapter 5

Master and Slave Device Communication

5.1 Integration

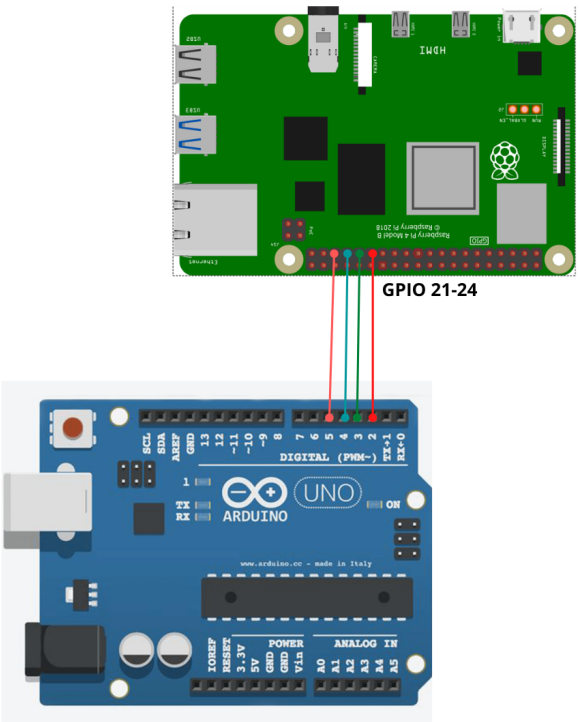


Figure 5.1: Communication Connections

5.2 Software Interface

5.2.1 Arduino IDE Code

```
const int D0 = 2;
const int D1 = 3;
const int D2 = 4;
const int D3 = 5;

int a,b,c,d,data;

void setup() {

Serial.begin(9600);
pinMode(D0, INPUT_PULLUP);
pinMode(D1, INPUT_PULLUP);
pinMode(D2, INPUT_PULLUP);
pinMode(D3, INPUT_PULLUP);
}

void loop()
{
   a = digitalRead(D0);
   b = digitalRead(D1);
   c = digitalRead(D2);
   d = digitalRead(D3);
   Serial.print("a= ");
   Serial.println(a);
   Serial.print("b= ");
   Serial.println(b);
   Serial.print("c= ");
   Serial.println(c);
   Serial.print("d= ");
   Serial.println(d);
   data = 8*d+4*c+2*b+a;
}
```

5.2.2 Open CV Code

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

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

RaspiCam_Cv Camera;

stringstream ss;

 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
    Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,500 ) );
    Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );
    Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,50 ) );
    Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );
    Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,50 ) );
    Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
    Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,100));

}

void Capture()
{
Camera.grab();
    Camera.retrieve( frame);
    cvtColor(frame, frame, COLOR_BGR2RGB);
}

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();

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

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

    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;
}
```

5.3 Results

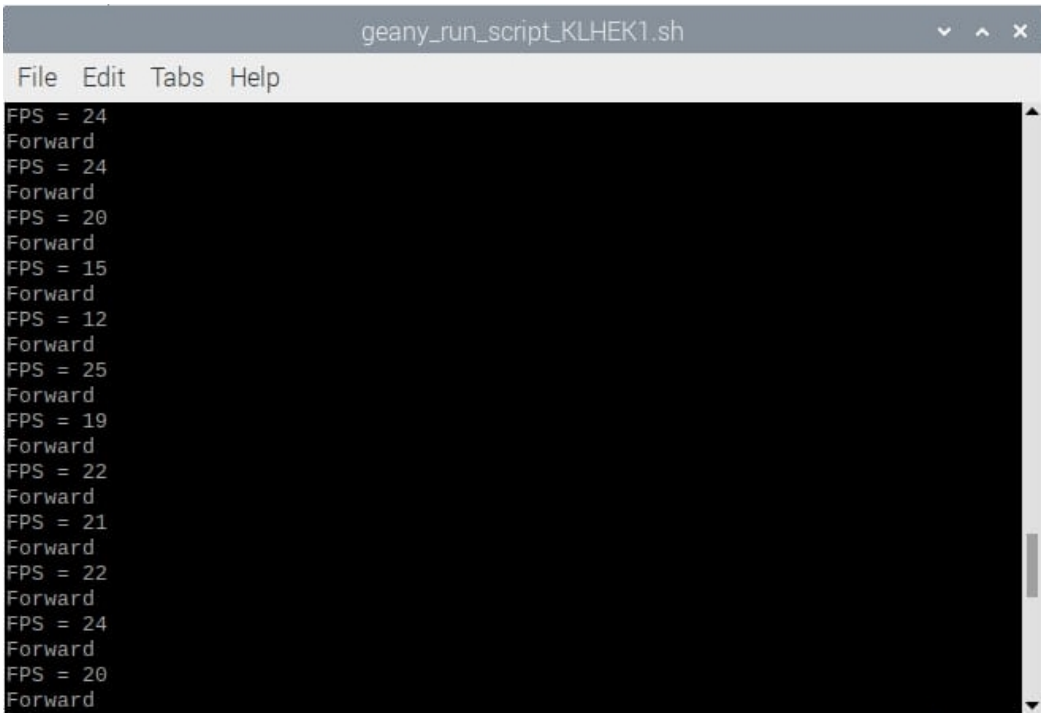


Figure 5.2: Raspberry Pi Terminal Window

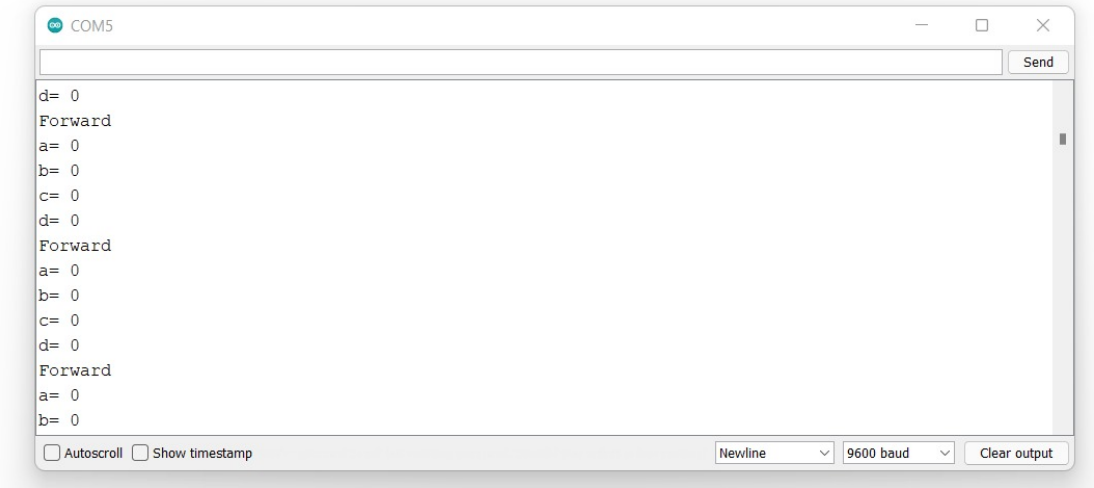


Figure 5.3: Arduino Serial Monitor

Chapter 6

Arduiuno Uno and L298N Motor Driver Communication

6.1 Integration

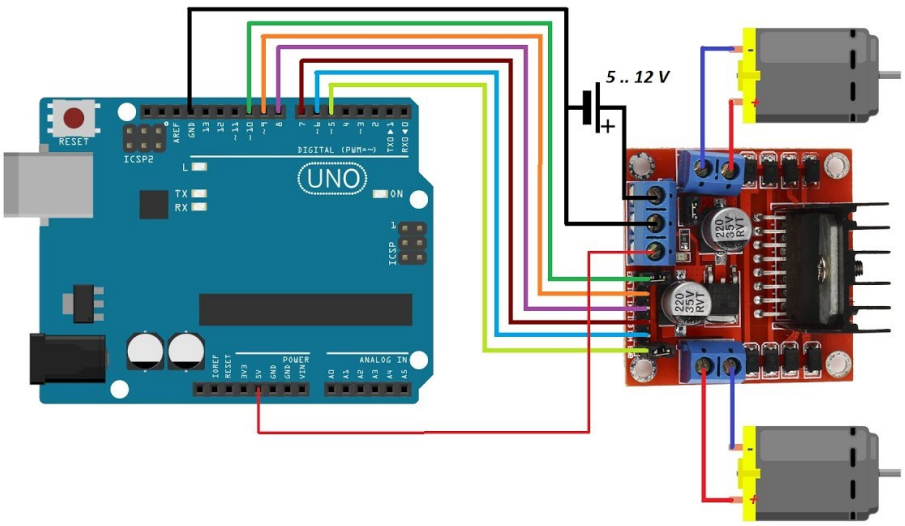


Figure 6.1: Arduino and L298N Connections

6.2 Software Interface

```
const int EnableL = 11;
const int HighL = 9;        // LEFT SIDE MOTOR
const int LowL =10;
```

```
const int EnableR = 6;
const int HighR = 7;        //RIGHT SIDE MOTOR
const int LowR =8;

void setup() {

Serial.begin(9600);
pinMode(EnableL, OUTPUT);
pinMode(HighL, OUTPUT);
pinMode(LowL, OUTPUT);


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

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

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

6.3 Graphical Representation

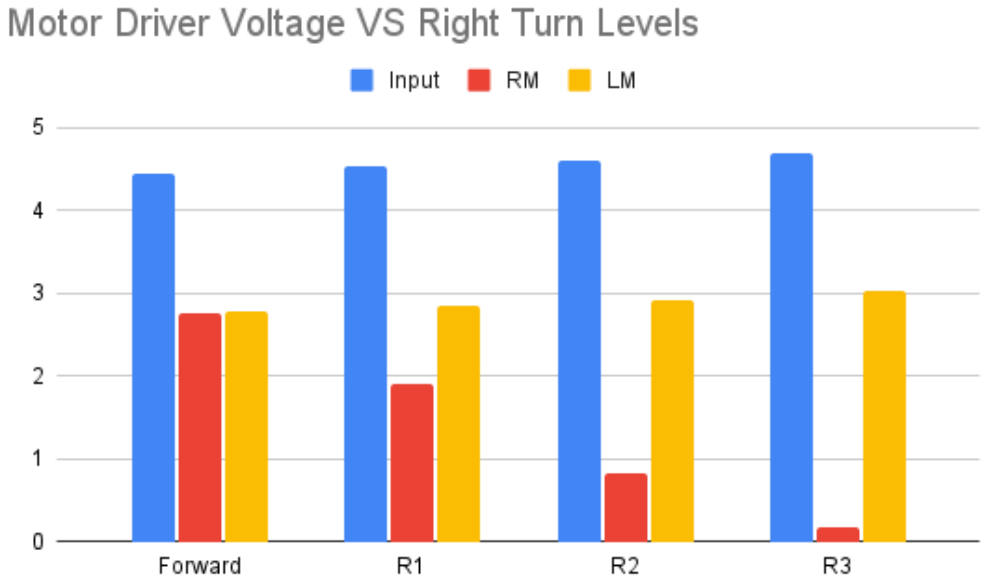


Figure 6.2: Variation in Motor Driver Voltages VS Right Turn Levels

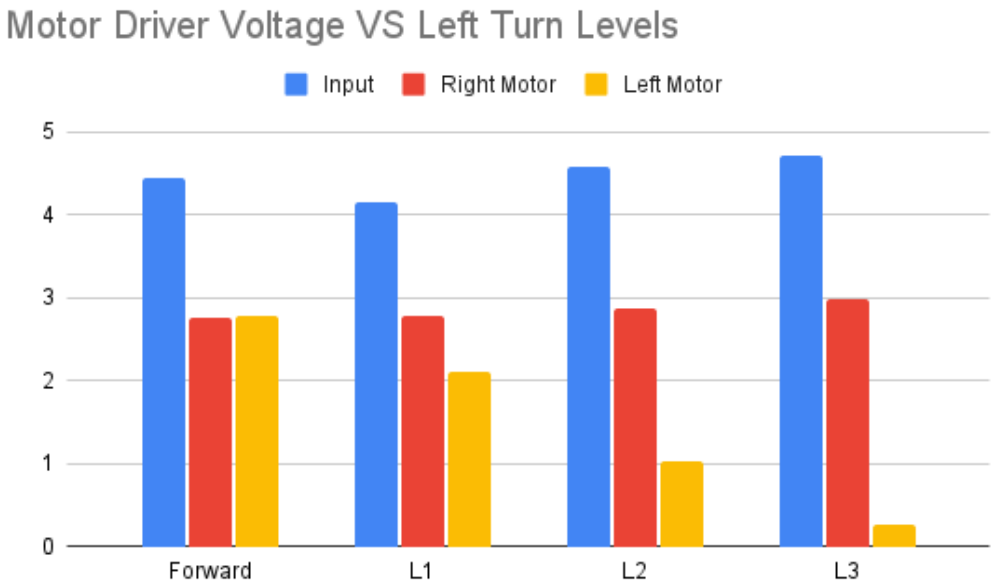


Figure 6.3: Variation in Motor Driver Voltages VS Left Turn Levels

Chapter 7

RaspiCam Interface with Raspberry Pi

7.1 Integration

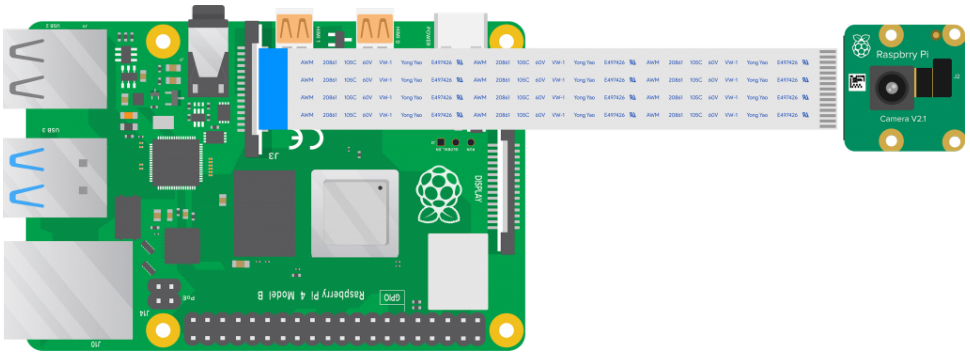


Figure 7.1: RaspiCam Connection

7.2 Defining Region Of Interest

7.2.1 Introduction

A region of interest is a sample within a data set identified for a particular purpose. The ROI defines the borders of an object under consideration.

In our project, the ROI is set for the car to detect the upcoming lane and the surroundings for processing and decision-making for steering control and braking

assistance.

7.2.2 Parameters

Region Of Interest

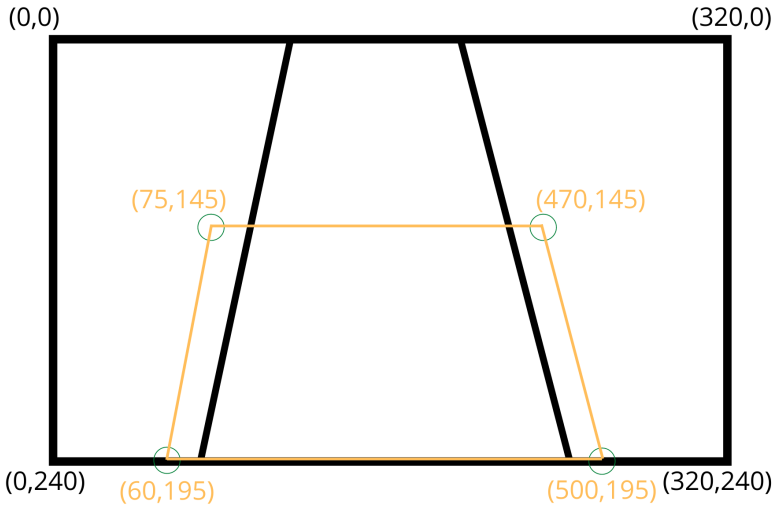


Figure 7.2: Design sketch (ROI)

Bird Eye View

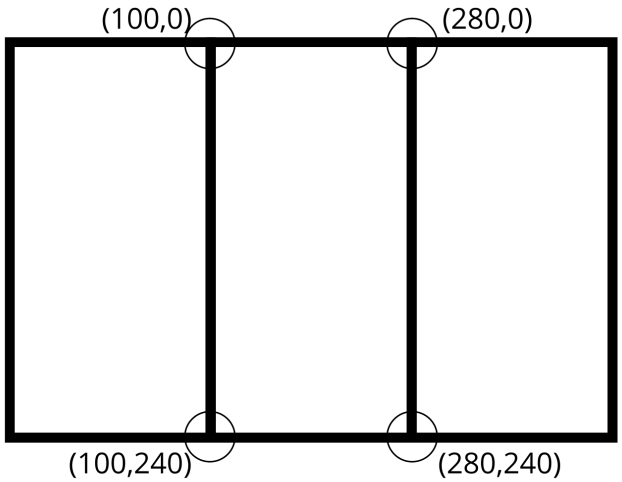


Figure 7.3: Design Sketch (Bird Eye View)

Software Interface

```
Point2f Source[] = {Point2f(75,145),Point2f(470,145),
Point2f(60,195), Point2f(500,195)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),
Point2f(100,240), Point2f(280,240)};

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, 185, 240, frameThresh);
Canny(frameGray,frameEdge, 450, 200, 3, false);
add(frameThresh, frameEdge, frameFinal);
cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);
//used in histrogram function only

}
```

Result

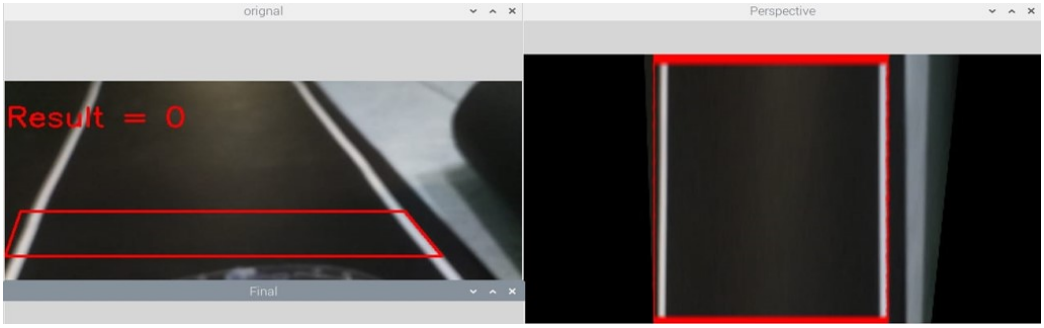


Figure 7.4: Real-time ROI and Bird Eye View

7.2.3 Canny Edge Detection [1]

Introduction

Canny edge detection is a image processing method used to detect edges in an image while suppressing noise. The main steps are as follows:

Step 1 - Grayscale Conversion

Convert the image to grayscale.

Step 2 - Gaussian Blur

Perform a Gaussian blur on the image. The blur removes some of the noise before further processing the image. A sigma of 1.4 is used in this example and was determined through trial and error.

Step 3 - Determine the Intensity Gradients

The gradients can be determined by using a Sobel filter where A is the image. An edge occurs when the color of an image changes, hence the intensity of the pixel changes as well.

$$\text{G\_x} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \text{A, G\_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \text{A}$$

Taking the derivatives will output:

Then, calculate the magnitude and angle of the directional gradients:

$$—G—=\sqrt{{G_x}^2 + {G_y}^2}$$

$$\angle G = arctan(G\_y/G\_x)$$

The magnitude of the image results in the following output:

Step 4 - Non Maximum Suppression

The image magnitude produced results in thick edges. Ideally, the final image should have thin edges. Thus, we must perform non maximum suppression to thin out the edges.

Non maximum suppression works by finding the pixel with the maximum value in an edge. In the above image, it occurs when pixel q has an intensity that is larger than both p and r where pixels p and r are the pixels in the gradient direction of q. If this condition is true, then we keep the pixel, otherwise we set the pixel to zero (make it a black pixel).

Non maximum suppression can be achieved by interpolating the pixels for greater accuracy:

$$r=\alpha b + (1 - \alpha)a$$

The result of this is:

Step 5 - Double Thresholding

We notice that the result from non maximum suppression is not perfect, some edges may not actually be edges and there is some noise in the image. Double thresholding takes care of this. It sets two thresholds, a high and a low threshold. In my algorithm, I normalized all the values such that they will only range from 0 to 1. Pixels with a high value are most likely to be edges. For example, you might choose the high threshold to be 0.7, this means that all pixels with a value larger than 0.7 will be a strong edge. You might also choose a low threshold of 0.3, this means that all pixels less than it is not an edge and you would set it to 0. The values in between 0.3 and 0.7 would be weak edges, in other words, we do not know if these are actual edges or not edges at all. Step 6 will explain how we can determine which weak edge is an actual edge.

This threshold is different per image so I had to vary the values. In my implementation I found it helpful to choose a threshold ratio instead of a specific value and multiple that by the max pixel value in the image. As for the low threshold, I chose a low threshold ratio and multiplied it by the high threshold value:

highThreshold = max(max(im))*highThresholdRatio;

lowThreshold = highThreshold*lowThresholdRatio;

Step 6 - Edge Tracking by Hysteresis

Now that we have determined what the strong edges and weak edges are, we need to determine which weak edges are actual edges. To do this, we perform an edge tracking algorithm. Weak edges that are connected to strong edges will be actual/real edges. Weak edges that are not connected to strong edges will be removed. To speed up this process, my algorithm keeps track of the weak and strong edges that way I can recursively iterate through the strong edges and see if there are connected weak edges instead of having to iterate through every pixel in the image.

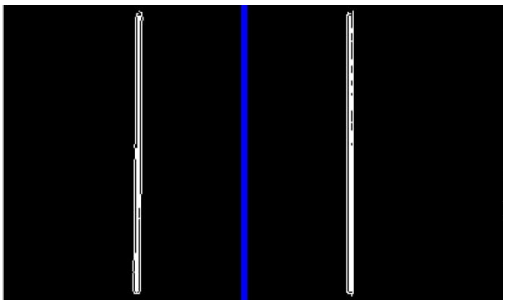
Step 7 - Cleaning Up

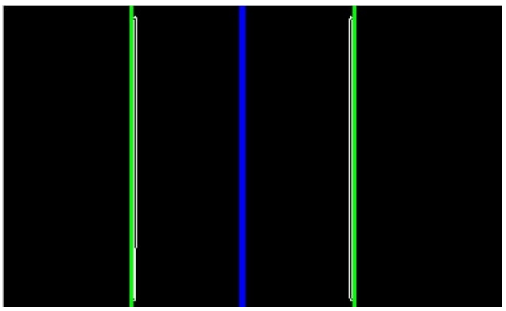
Finally, we will iterate through the remaining weak edges and set them to zero resulting in the final processed image:

Implementation Algorithm

```
Canny(frameGray,frameEdge, 450, 200, 3, false);
```

Output





7.2.4 Software Interface

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

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

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

RaspiCam_Cv Camera;

stringstream ss;


vector<int> histrogramLane;

Point2f Source[] = {Point2f(75,145),Point2f(470,145),
Point2f(60,195), Point2f(500,195)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),
Point2f(100,240),
Point2f(280,240)};


 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
    Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,500 ) );
    Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );
    Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,50 ) );
    Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );
```

```
    Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,50 ) );
    Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
    Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,100));

}

void Capture()
{
Camera.grab();
    Camera.retrieve( frame);
    cvtColor(frame, frame, COLOR_BGR2RGB);
}

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


Matrix = getPerspectiveTransform(Source, Destination);
warpPerspective(frame, framePers, Matrix, Size(400,240));
}

void Threshold()
{
cvtColor(framePers, frameGray, COLOR_RGB2GRAY);
inRange(frameGray, 200, 255, frameThresh);
Canny(frameGray,frameEdge, 450, 200, 3, false);
add(frameThresh, frameEdge, frameFinal);
cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);
//used in histrogram function only

}
```

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

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

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

    vector<int>:: iterator RightPtr;
    RightPtr = max_element(histrogramLane.begin()
    +250, histrogramLane.end());
    RightLanePos = distance(histrogramLane.begin(), RightPtr);

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

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

    line(frameFinal, Point2f(laneCenter,0), Point2f(laneCenter,
```

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

    Result = laneCenter-frameCenter;
}


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

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();

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

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

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

    namedWindow("Final", WINDOW_KEEPRATIO);
    moveWindow("Final", 1280, 100);
    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;

}
```

7.2.5 Result

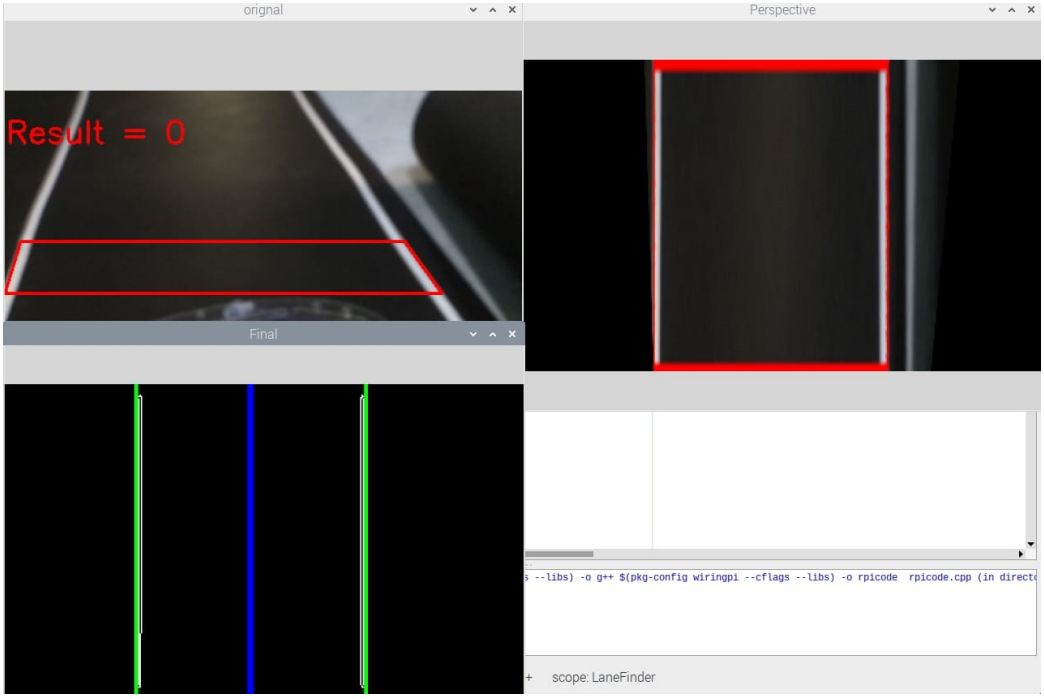


Figure 7.5: Lane Identification Final Output

7.2.6 Graphical Representation

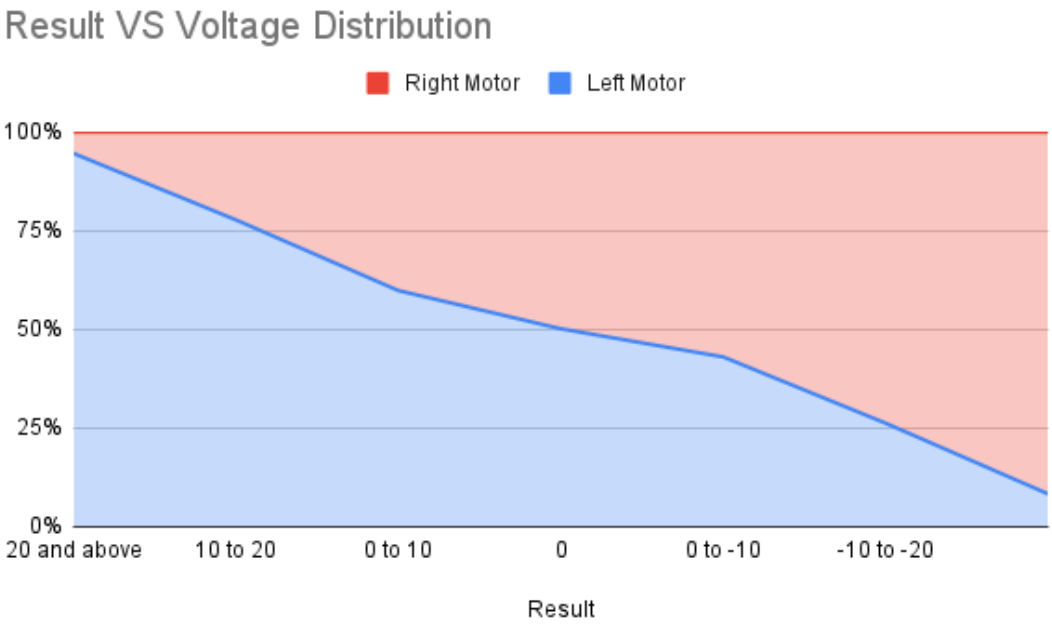


Figure 7.6: Change in Result VS Percentage of voltage drawn by each motor

Chapter 8

Implementation of Machine Learning for Image Processing

8.1 Cascade Training [2]

8.1.1 Introduction

Computer vision is how computers automate tasks that mimic the human response to visual information. Image features such as points, edges, or patterns are used to identify an object in an image. A cascade classifier uses these visual cues as features to determine if an object is in the image, such as a face.

Cascading is a particular case of ensemble learning based on the concatenation of several classifiers, using all information collected from the output from a given classifier as additional information for the next classifier in the cascade. Unlike voting or stacking ensembles (multi-expert systems), cascading is a multistage method.

Cascading classifiers are trained with positive sample views of a particular object and arbitrary negative images of the same size. After the classifier is trained it can be applied to a region of an image and detect the object in question. To search for the object in the entire frame, the search window can be moved across the image and check every location for the classifier. This process is most commonly used in image processing for object detection and tracking, primarily facial detection, and recognition.

8.1.2 Preparation of the Training Data

For training a boosted cascade of weak classifiers we need a set of positive samples (containing actual objects you want to detect) and a set of negative images (containing everything you do not want to detect). The set of negative samples must be prepared manually, whereas a set of positive samples is created using the opencv_createsamples application.

Negative Samples

Negative samples are taken from arbitrary images, not containing objects you want to detect. These negative images, from which the samples are generated, should be listed in a special negative image file containing one image path per line. Negative samples and sample images are also called background samples or background images, and can be used interchangeably.

Images may be of different sizes, however, each image should be equal to or larger than the desired training window size because these images are used to sub sample a given negative image into several image samples having this training window size.

Positive Samples

The boosting process uses positive samples to define what the model should actually look for when trying to find your objects of interest. Two ways of generating a positive sample data-set.

1. We can generate a bunch of positives from a single positive object image.
2. We can supply all the positives ourselves and only use the tool to cut them out, and resize them.

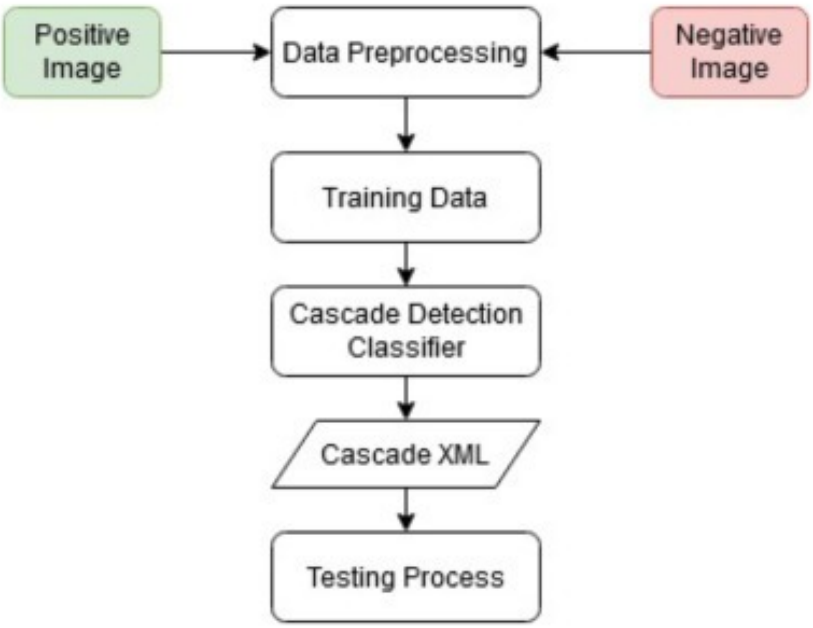


Figure 8.1: Cascade Training Flow

8.1.3 Implementation of Object Detection

Software Interface

Arduino Uno Code

```
void Object()
{

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);            //stop
  SignalOFF();
  delay(1000);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);        //left
  analogWrite(EnableL, 250);
  analogWrite(EnableR, 250);
```

```
LeftIndicator();
delay(500);

analogWrite(EnableL, 0);
analogWrite(EnableR, 0);            //stop
SignalOFF();
delay(200);

digitalWrite(HighL, LOW);
digitalWrite(LowL, HIGH);           //forward
digitalWrite(HighR, LOW);
digitalWrite(LowR, HIGH);
analogWrite(EnableL, 255);
analogWrite(EnableR, 255);
SignalOFF();
delay(1000);

analogWrite(EnableL, 0);           //stop
analogWrite(EnableR, 0);
SignalOFF();
delay(200);

digitalWrite(HighL, LOW);
digitalWrite(LowL, HIGH);
digitalWrite(HighR, HIGH);         //right
digitalWrite(LowR, LOW);
analogWrite(EnableL, 255);
analogWrite(EnableR, 255);
RightIndicator();
delay(500);

analogWrite(EnableL, 0);               //stop
analogWrite(EnableR, 0);
SignalOFF();
delay(1000);
```

```
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);       // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  SignalOFF();
  delay(500);

   i  = i+1;
}
```

Open CV Code

```
void Object_detection()
{
    if(!Object_Cascade.load("//home//pi//Desktop//MACHINE
    LEARNING//Object_cascade.xml"))
    {
printf("Unable to open Object cascade file");
    }

    RoI_Object = frame_Object(Rect(100,50,200,190));
    cvtColor(RoI_Object, gray_Object, COLOR_RGB2GRAY);
    equalizeHist(gray_Object, gray_Object);
    Object_Cascade.detectMultiScale(gray_Object, Object);

    for(int i=0; i<Object.size(); i++)
    {
Point P1(Object[i].x, Object[i].y);
Point P2(Object[i].x + Object[i].width, Object[i].y +
Object[i].height);

rectangle(RoI_Object, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Object, "Object", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
```

```
dist_Object = (-0.48)*(P2.x-P1.x) + 56.6;

        ss.str(" ");
        ss.clear();
        ss<<"D = "<<dist_Object<<"cm";
        putText(RoI_Object, ss.str(), Point2f(1,130), 0,1,
        Scalar(0,0,255), 2);

    }

}
```

Results

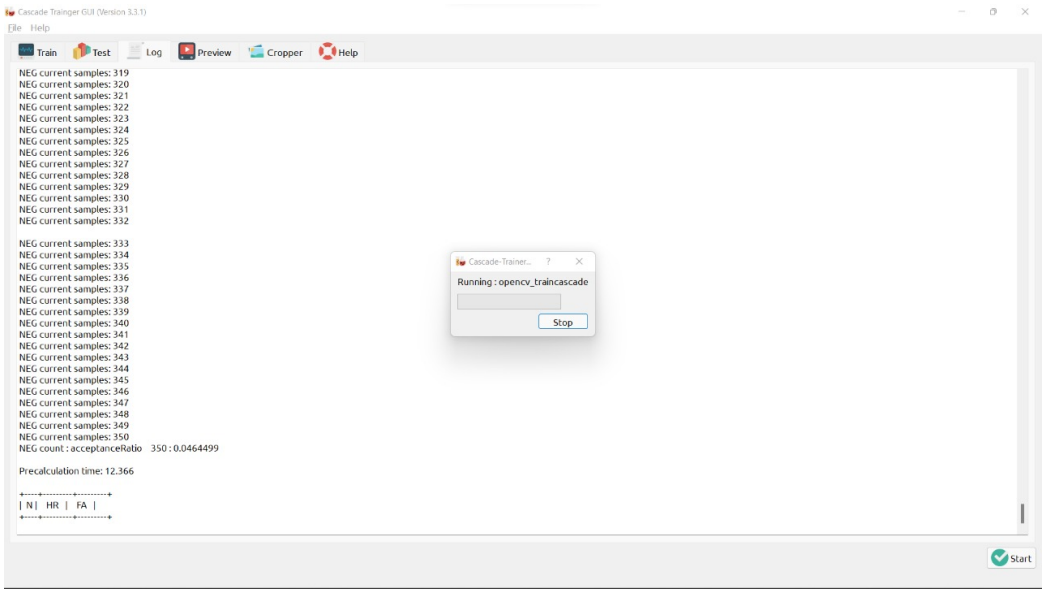


Figure 8.2: Multi Stage Training

Figure 8.3: Virtual Detection of Stop Sign

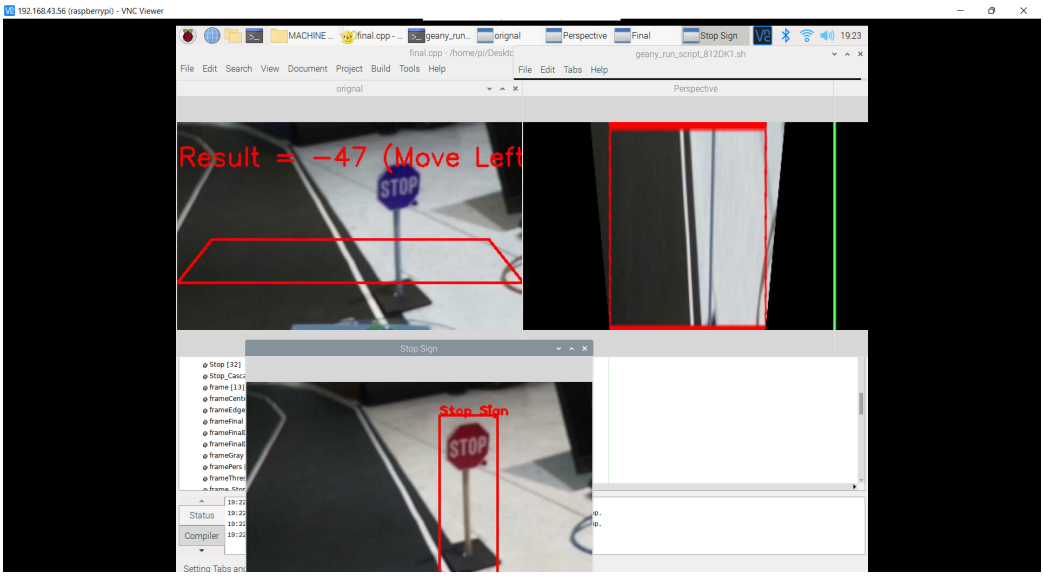


Figure 8.4: Real-time Detection of Stop Sign

8.2 Traffic Signal Identification

8.2.1 Software Interface

Arduino Uno Code

```
    analogWrite(EnableL, 0);
    analogWrite(EnableR, 0);
    delay(2000);
```

Open CV Code

```
void Traffic_detection()
{
    if(!Traffic_Cascade.load("//home//pi//Desktop//MACHINE
    LEARNING//Trafficc_cascade.xml"))
    {
printf("Unable to open traffic cascade file");
    }

    RoI_Traffic = frame_Traffic(Rect(200,0,200,140));
    cvtColor(RoI_Traffic, gray_Traffic, COLOR_RGB2GRAY);
    equalizeHist(gray_Traffic, gray_Traffic);
    Traffic_Cascade.detectMultiScale(gray_Traffic, Traffic);

    for(int i=0; i<Traffic.size(); i++)
    {
Point P1(Traffic[i].x, Traffic[i].y);
Point P2(Traffic[i].x + Traffic[i].width, Traffic[i].y +
Traffic[i].height);

rectangle(RoI_Traffic, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Traffic, "Traffic Light", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
dist_Traffic = (-1.07)*(P2.x-P1.x) + 102.597;

       ss.str(" ");
```

```
        ss.clear();
        ss<<"D = "<<P2.x-P1.x<<"cm";
        putText(RoI_Traffic, ss.str(), Point2f(1,130), 0,1,
        Scalar(0,0,255), 2);
    }
}
```

Results

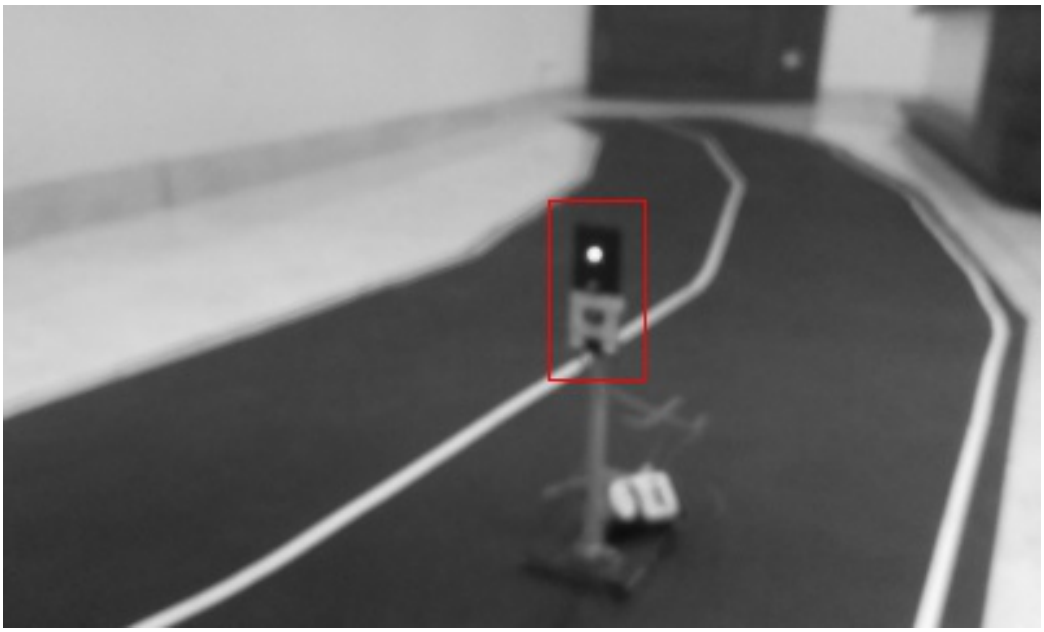


Figure 8.5: Virtual Detection of Traffic Light

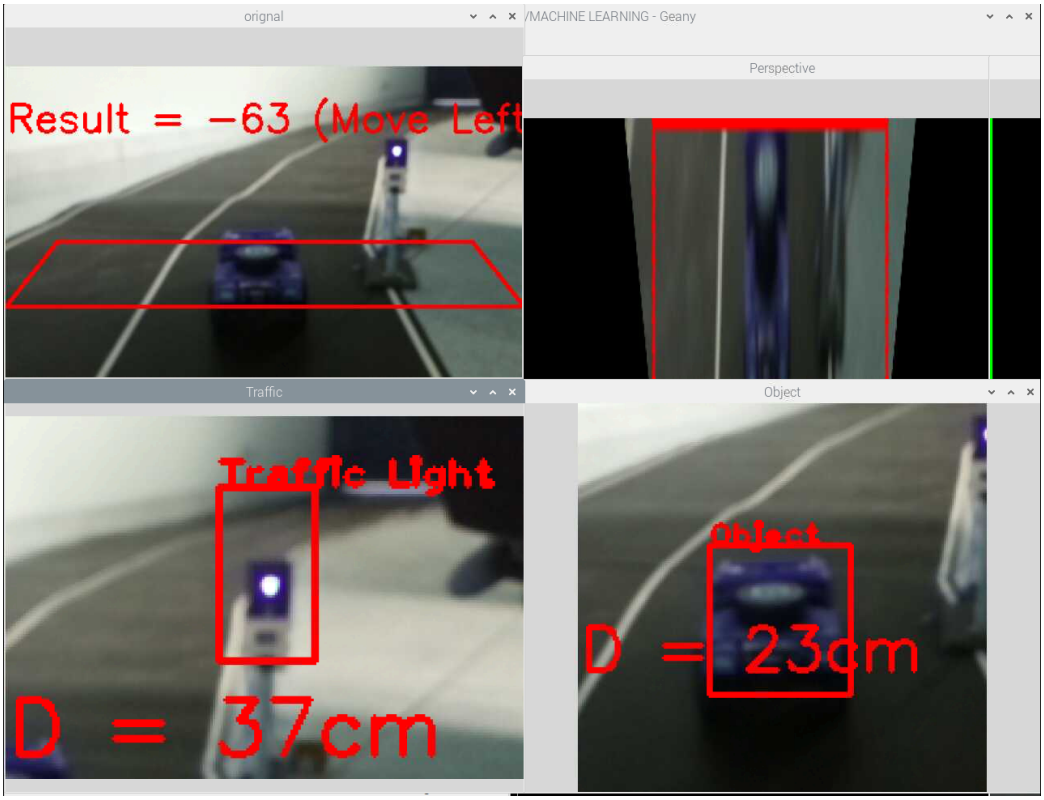


Figure 8.6: Real-Time Detection of Traffic Light

Chapter 9

Final Code

9.1 Arduino Uno

```
const int LeftSignal = 4;
const int enableLeft = 11;
const int RightSignal = 12;
const int enableRight = 13;
int i =0;
unsigned long int j =0;

const int EnableL = 11;
const int HighL = 9;       // LEFT SIDE MOTOR
const int LowL =10;

const int EnableR = 6;
const int HighR = 7;       //RIGHT SIDE MOTOR
const int LowR =8;

const int D0 = 2;       //Raspberry pin 21    LSB
const int D1 = 3;       //Raspberry pin 22
const int D2 = 4;       //Raspberry pin 23
const int D3 = 5;       //Raspberry pin 24    MSB

int sensorPin = A0;
int LED = A1;
```

```
int a,b,c,d,data;


void setup() {

pinMode(EnableL, OUTPUT);
pinMode(HighL, OUTPUT);
pinMode(LowL, OUTPUT);


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

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

pinMode(LeftSignal,OUTPUT);
pinMode(enableLeft,OUTPUT);
pinMode(RightSignal,OUTPUT);
pinMode(enableRight,OUTPUT);

pinMode(sensorPin, INPUT);
pinMode(LED, OUTPUT);

}

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,255);

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

}


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

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

}

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

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

```
}

void Left1()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,160);

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

}

void Left2()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,90);

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

}


void Left3()
{
  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL,50);

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

```
}

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

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,160);  //200

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

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,90);   //160

}

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

  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableR,50);   //100

}
```

```
void UTurn()
{
  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(400);

  analogWrite(EnableL, 250);
  analogWrite(EnableR, 250);    //forward
  SignalOFF();
  delay(1000);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(400);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);   //   left
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  LeftIndicator();
  delay(700);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(400);

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

```
  analogWrite(EnableR, 255);
  SignalOFF();
  delay(900);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(400);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);    //left
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  LeftIndicator();
  delay(700);


  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(1000);



  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);
  digitalWrite(LowL, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  SignalOFF();
  delay(300);
}
```

```
void Object()
{

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);            //stop
  SignalOFF();
  delay(1000);

  digitalWrite(HighL, HIGH);
  digitalWrite(LowL, LOW);
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);        //left
  analogWrite(EnableL, 250);
  analogWrite(EnableR, 250);
  LeftIndicator();
  delay(500);

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);            //stop
  SignalOFF();
  delay(200);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);           //forward
  digitalWrite(HighR, LOW);
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  SignalOFF();
  delay(1000);

  analogWrite(EnableL, 0);           //stop
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(200);

  digitalWrite(HighL, LOW);
```

```
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);         //right
  digitalWrite(LowR, LOW);
  analogWrite(EnableL, 255);
  analogWrite(EnableR, 255);
  RightIndicator();
  delay(500);

  analogWrite(EnableL, 0);               //stop
  analogWrite(EnableR, 0);
  SignalOFF();
  delay(1000);


  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);       // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  SignalOFF();
  delay(500);

   i  = i+1;
}

void Lane_Change()
{

  analogWrite(EnableL, 0);
  analogWrite(EnableR, 0);           //stop
  SignalOFF();
  delay(1000);

  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, HIGH);
```

```
digitalWrite(LowR, LOW);        //Right
analogWrite(EnableL, 250);
analogWrite(EnableR, 250);
RightIndicator();
delay(500);

analogWrite(EnableL, 0);
analogWrite(EnableR, 0);            //stop
SignalOFF();
delay(200);

digitalWrite(HighL, LOW);
digitalWrite(LowL, HIGH);           //forward
digitalWrite(HighR, LOW);
digitalWrite(LowR, HIGH);
analogWrite(EnableL, 255);
analogWrite(EnableR, 255);
SignalOFF();
delay(800);

analogWrite(EnableL, 0);           //stop
analogWrite(EnableR, 0);
SignalOFF();
delay(200);

digitalWrite(HighL, HIGH);
digitalWrite(LowL, LOW);
digitalWrite(HighR, LOW);         //LEFT
digitalWrite(LowR, HIGH);
analogWrite(EnableL, 255);
analogWrite(EnableR, 255);
LeftIndicator();
delay(500);

analogWrite(EnableL, 0);               //stop
analogWrite(EnableR, 0);
SignalOFF();
```

```
  delay(1000);


  digitalWrite(HighL, LOW);
  digitalWrite(LowL, HIGH);
  digitalWrite(HighR, LOW);       // forward
  digitalWrite(LowR, HIGH);
  analogWrite(EnableL, 150);
  analogWrite(EnableR, 150);
  SignalOFF();
  delay(500);

}

void LeftIndicator()
{
  digitalWrite(LeftSignal,HIGH);
  digitalWrite(enableLeft,LOW);
  digitalWrite(RightSignal,LOW);
  digitalWrite(enableRight,LOW);
}

void RightIndicator()
{
  digitalWrite(LeftSignal,LOW);
  digitalWrite(enableLeft,LOW);
  digitalWrite(RightSignal,HIGH);
  digitalWrite(enableRight,LOW);
}

void Brake()
{
  digitalWrite(LeftSignal,HIGH);
  digitalWrite(enableLeft,HIGH);
  digitalWrite(RightSignal,HIGH);
  digitalWrite(enableRight,HIGH);
}
```

```
void SignalOFF()
{
  digitalWrite(LeftSignal,LOW);
  digitalWrite(enableLeft,LOW);
  digitalWrite(RightSignal,LOW);
  digitalWrite(enableRight,LOW);
}

void Headlight()
{
  int sensorValue = analogRead(sensorPin);
  if( sensorValue <= 700 )
  {
    analogWrite(LED, 255);
    Serial.println("It's Dark Outside; Lights status: ON");
}
  else
{
    analogWrite(LED, 0);
    Serial.println("It's Bright Outside; Lights status: OFF");
  }
}


void loop()
{
    if (j > 25000)
    {
      Lane_Change();
      i = 0;
      j = 0;
    }

  Data();
  Headlight();
  if(data==0)
```

```
 {
   Forward();
   SignalOFF();
   if (i>0)
   {
    j = j+1;
   }
 }

else if(data==1)
 {
   Right1();
   SignalOFF();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==2)
 {
   Right2();
   SignalOFF();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==3)
 {
   Right3();
   SignalOFF();
        if (i>0)
   {
    j = j+1;
   }
```

```
 }

else if(data==4)
 {
   Left1();
   SignalOFF();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==5)
 {
   Left2();
   SignalOFF();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==6)
 {
   Left3();
   SignalOFF();
        if (i>0)
   {
    j = j+1;
   }
 }

else if(data==7)
 {
   UTurn();
 }
```

```
  else if (data==8)
   {
      analogWrite(EnableL, 0);
      analogWrite(EnableR, 0);
      delay(4000);

      analogWrite(EnableL, 150);
      analogWrite(EnableR, 150);
      delay(1000);
   }

      else if(data==9)
   {
     Object();
   }

         else if(data==10)
   {
      analogWrite(EnableL, 0);
      analogWrite(EnableR, 0);
      Brake();
      delay(2000);
   }


     else if(data>10)
   {
     Stop();
    }
}
```

9.2 Raspberry Pi

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

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


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

RaspiCam_Cv Camera;

stringstream ss;


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

Point2f Source[] = {Point2f(75,145),Point2f(470,145),
Point2f(60,195), Point2f(500,195)};
Point2f Destination[] = {Point2f(100,0),Point2f(280,0),
Point2f(100,240), Point2f(280,240)};



CascadeClassifier Stop_Cascade, Object_Cascade, Traffic_Cascade;
Mat frame_Stop, RoI_Stop, gray_Stop, frame_Object, RoI_Object,
```

```
gray_Object, frame_Traffic, RoI_Traffic, gray_Traffic;
vector<Rect> Stop, Object, Traffic;
int dist_Stop, dist_Object, dist_Traffic;

 void Setup ( int argc,char **argv, RaspiCam_Cv &Camera )
  {
    Camera.set ( CAP_PROP_FRAME_WIDTH,  ( "-w",argc,argv,500 ) );
    Camera.set ( CAP_PROP_FRAME_HEIGHT,  ( "-h",argc,argv,240 ) );
    Camera.set ( CAP_PROP_BRIGHTNESS, ( "-br",argc,argv,50 ) );
    Camera.set ( CAP_PROP_CONTRAST ,( "-co",argc,argv,50 ) );
    Camera.set ( CAP_PROP_SATURATION,  ( "-sa",argc,argv,50 ) );
    Camera.set ( CAP_PROP_GAIN,  ( "-g",argc,argv ,50 ) );
    Camera.set ( CAP_PROP_FPS,  ( "-fps",argc,argv,0));

}

void Capture()
{
    Camera.grab();
    Camera.retrieve( frame);
    cvtColor(frame, frame_Stop, COLOR_BGR2RGB);
    cvtColor(frame, frame_Object, COLOR_BGR2RGB);
    cvtColor(frame, frame_Traffic, COLOR_BGR2RGB);
    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, 230, 255, frameThresh);
Canny(frameGray,frameEdge, 450, 200, 3, false);
add(frameThresh, frameEdge, frameFinal);
cvtColor(frameFinal, frameFinal, COLOR_GRAY2RGB);
cvtColor(frameFinal, frameFinalDuplicate, COLOR_RGB2BGR);
cvtColor(frameFinal, frameFinalDuplicate1, COLOR_RGB2BGR);

}

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

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

histrogramLaneEnd.resize(400);
        histrogramLaneEnd.clear();
for (int i = 0; i < 400; 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(),
    histrogramLane.begin() + 150);
    LeftLanePos = distance(histrogramLane.begin(), LeftPtr);

    vector<int>:: iterator RightPtr;
    RightPtr = max_element(histrogramLane.begin() +250,
    histrogramLane.end());
    RightLanePos = distance(histrogramLane.begin(), RightPtr);

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

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

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

    Result = laneCenter-frameCenter;
}


void Stop_detection()
{
```

```
    if(!Stop_Cascade.load("//home//pi//Desktop//MACHINE LEARNING//
    Stop_cascade.xml"))
    {
printf("Unable to open stop cascade file");
    }

    RoI_Stop = frame_Stop(Rect(200,0,200,140));
    cvtColor(RoI_Stop, gray_Stop, COLOR_RGB2GRAY);
    equalizeHist(gray_Stop, gray_Stop);
    Stop_Cascade.detectMultiScale(gray_Stop, Stop);

    for(int i=0; i<Stop.size(); i++)
    {
Point P1(Stop[i].x, Stop[i].y);
Point P2(Stop[i].x + Stop[i].width, Stop[i].y + Stop[i].height);

rectangle(RoI_Stop, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Stop, "Stop Sign", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
dist_Stop = (-1.07)*(P2.x-P1.x) + 102.597;

       ss.str(" ");
       ss.clear();
       ss<<"D = "<<dist_Stop<<"cm";
       putText(RoI_Stop, ss.str(), Point2f(1,130), 0,1,
       Scalar(0,0,255), 2);

    }

}

void Traffic_detection()
{
    if(!Traffic_Cascade.load("//home//pi//Desktop//MACHINE LEARNING//
    Trafficc_cascade.xml"))
    {
printf("Unable to open traffic cascade file");
```

```
    }

    RoI_Traffic = frame_Traffic(Rect(200,0,200,140));
    cvtColor(RoI_Traffic, gray_Traffic, COLOR_RGB2GRAY);
    equalizeHist(gray_Traffic, gray_Traffic);
    Traffic_Cascade.detectMultiScale(gray_Traffic, Traffic);

    for(int i=0; i<Traffic.size(); i++)
    {
Point P1(Traffic[i].x, Traffic[i].y);
Point P2(Traffic[i].x + Traffic[i].width, Traffic[i].y +
Traffic[i].height);

rectangle(RoI_Traffic, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Traffic, "Traffic Light", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
dist_Traffic = (-1.07)*(P2.x-P1.x) + 102.597;

       ss.str(" ");
       ss.clear();
       ss<<"D = "<<P2.x-P1.x<<"cm";
       putText(RoI_Traffic, ss.str(), Point2f(1,130), 0,1,
       Scalar(0,0,255), 2);

    }

}




void Object_detection()
{
    if(!Object_Cascade.load("//home//pi//Desktop//MACHINE LEARNING//
    Object_cascade.xml"))
    {
printf("Unable to open Object cascade file");
```

```
    }

    RoI_Object = frame_Object(Rect(100,50,200,190));
    cvtColor(RoI_Object, gray_Object, COLOR_RGB2GRAY);
    equalizeHist(gray_Object, gray_Object);
    Object_Cascade.detectMultiScale(gray_Object, Object);

    for(int i=0; i<Object.size(); i++)
    {
Point P1(Object[i].x, Object[i].y);
Point P2(Object[i].x + Object[i].width, Object[i].y + Object[i].height);

rectangle(RoI_Object, P1, P2, Scalar(0, 0, 255), 2);
putText(RoI_Object, "Object", P1, FONT_HERSHEY_PLAIN, 1,
Scalar(0, 0, 255, 255), 2);
dist_Object = (-0.48)*(P2.x-P1.x) + 56.6;

       ss.str(" ");
       ss.clear();
       ss<<"D = "<<dist_Object<<"cm";
       putText(RoI_Object, ss.str(), Point2f(1,130), 0,1,
       Scalar(0,0,255), 2);

    }

}


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();
    Stop_detection();
    Object_detection();
    Traffic_detection();

    if (dist_Stop > 5 && dist_Stop < 20)
    {
digitalWrite(21, 0);
digitalWrite(22, 0);    //decimal = 8
digitalWrite(23, 0);
digitalWrite(24, 1);
cout<<"Stop Sign"<<endl;
dist_Stop = 0;

goto Stop_Sign;
    }
```

```
        if (dist_Object > 5 && dist_Object < 30)
    {
digitalWrite(21, 1);
digitalWrite(22, 0);    //decimal = 9
digitalWrite(23, 0);
digitalWrite(24, 1);
cout<<"Object"<<endl;
dist_Object = 0;

goto Object;
    }

     if (dist_Traffic > 5 && dist_Traffic < 20)
    {
digitalWrite(21, 0);
digitalWrite(22, 1);    //decimal = 10
digitalWrite(23, 0);
digitalWrite(24, 1);
cout<<"Traffic Light"<<endl;
dist_Traffic = 0;

goto Traffic;
    }



    if (laneEnd > 4500)
    {
        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)
    {
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)
    {
digitalWrite(21, 0);
digitalWrite(22, 1);    //decimal = 6
digitalWrite(23, 1);
digitalWrite(24, 0);
cout<<"Left3"<<endl;
    }

    Stop_Sign:
    Object:
    Traffic:


   if (laneEnd > 4500)
    {
       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<<" (Move 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("Final", WINDOW_KEEPRATIO);
    moveWindow("Final", 1280, 100);
    resizeWindow("Final", 640, 480);
    imshow("Final", frameFinal);

    namedWindow("Stop Sign", WINDOW_KEEPRATIO);
    moveWindow("Stop Sign", 1280, 580);
    resizeWindow("Stop Sign", 640, 480);
    imshow("Stop Sign", RoI_Stop);

    namedWindow("Object", WINDOW_KEEPRATIO);
    moveWindow("Object", 640, 580);
    resizeWindow("Object", 640, 480);
    imshow("Object", RoI_Object);

    namedWindow("Traffic", WINDOW_KEEPRATIO);
    moveWindow("Traffic", 0, 580);
    resizeWindow("Traffic", 640, 480);
    imshow("Traffic", RoI_Traffic);



    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;
}
```

Conclusion

Adaptive Driving Assistance System, is the term used to describe the growing number of safety functions designed to improve driver, passenger and pedestrian safety by reducing both the severity and overall number of motor vehicle accidents. By the means of developing this project, we are able to achieve major key features like braking control, headlight control using LDR, steering control using machine learning and automatic turn indicators. The prototype developed by us, is successfully able to manoeuvre on the track made, which demonstrates all the above stated features with an overall 70-75% accuracy.

Bibliography

[1] J. Liang, "Canny edge detection," *Retrieved*, vol. 6, p. 15, 2019.

[2] A. Ahmadi, "Cascade trainer gui," 2016.

[3] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[4] M. Wagner, D. Zöbel, and A. Meroth, "An adaptive software and systems architecture for driver assistance systems based on service orientation," *International Journal of Machine Learning and Computing*, vol. 1, no. 4, pp. 359–365, 2011.

[5] A. Shaout, D. Colella, and S. Awad, "Advanced driver assistance systems-past, present and future," in *2011 Seventh International Computer Engineering Conference (ICENCO'2011)*. IEEE, 2011, pp. 72–82.

[6] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 569–590, 2006.

[7] H. Inoue, "Research into adas with driving intelligence for future innovation," in *2014 IEEE International Electron Devices Meeting*. IEEE, 2014, pp. 1–3.

[8] M. Hasenjäger and H. Wersing, "Personalization in advanced driver assistance systems and autonomous vehicles: A review," in *2017 ieee 20th international conference on intelligent transportation systems (itsc)*. IEEE, 2017, pp. 1–7.

[9] T. Yoshida, H. Kuroda, and T. Nishigaito, "Adaptive driver-assistance systems," *Hitachi Review*, vol. 53, no. 4, p. 213, 2004.