TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE

KALIMATI, KATHMANDU

Minor Project Final report on

SELF DRIVING CAR WITH ACCIDENT DETECTION

BY

ANUSHA PANT- KAT077BEI001

PRAJWAL DAHAL- KAT077BEI010

SHAILESH LAMA- KAT077BEI018

SWARAJ K.C.-KAT077BEI021

TO

DEPARTMENT OF ELECTRONICS, COMMUNICATION AND INFORMATION ENGINEERING

KATHMANDU, NEPAL

FALGUN, 2080

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
KATHMANDU ENGINEERING COLLEGE
Kalimati, Kathmandu
Department of Electronics, Communication and Information Engineering

CERTIFICATE

The undersigned certified that they have read and recommended to the Department of Electronics, Communication and Information Engineering a minor project work entitled "Self-Driving Car with Accident Detection" submitted by Anusha Pant (77001), Prajwal Dahal (77010), Shailesh Lama (77018) and Swaraj K.C (77021) in partial fulfillment of the requirements for the degree of bachelor of Engineering.

Er. Sarina Barahi
(Project Supervisor)
(Lecturer)
Department of Electronics,
Communication and Information
Engineering
(Kathmandu Engineering College)

Er. Anmol Ratna Bajracharya
(Project Supervisor)
(Associate Professor)
Department of Electronics,
Communication and Information
Engineering
(Kathmandu Engineering College)

Er. Sujan Shrestha
(Project Coordinator)
(Associate Professor)
Department of Electronics,
Communication and Information
Engineering
(Kathmandu Engineering College)

Er. Rajan Lama
(Head of Department)
(Associate Professor)
Department of Electronics,
Communication and Information
Engineering
(Kathmandu Engineering College)

Dr. Nanda B. Adhikari
(External Examiner)
(Associate Professor)
Department of Electronics
And Computer Engineering
(Pulchowk Campus)

ACKNOWLEDGEMENT

We would like to sincerely thank our supervisors **Associate Professor Er. Anmol Ratna Bajracharya** and **Lecturer Er. Sarina Barahi** for their unlisted encouragement and moreover for their timely support and guidance.

We owe our profound gratitude to our project coordinator **Associate Professor Er. Sujan Shrestha**, who took keen interest in our project and guided us all along, provided the necessary information for developing a good system.

We would like to express our deep gratitude to our Head of the Department of Electronics, Communication and Information Engineering **Associate Professor Er. Rajan Lama** for providing us with the opportunity to perform this project.

We are thankful and fortunate to get constant encouragement, support, constructive criticism, and guidance from all teaching staff of the Department of Electronics, Communication and Information Engineering.

ABSTRACT

This report presents the results of the development of a self-driving car with accident detection which aims to demonstrate a self-automated vehicle that recognizes traffic signs, follows traffic rules and sends signal to the phone number linked through GPS/GSM module in the case of an accident.

The fundamental components of a self-driving car with accident detection include perception, decision-making and control systems. The major goal of this project is to build and represent a prototype of a fully autonomous car that employs computer vision to detect lanes, traffic signs and accidents without human intervention using limited computing capacity. The project contains a system represented by a Raspberry Pi which serves as the image processing and machine learning unit. The detection of various traffic signs and objects is be done through Haar Cascade Classifier algorithm and the detection of lanes is done through Canny Edge Detection algorithm. Overall, the system is able to detect the lane and respond to changes in lane direction, detect traffic signs and follow the traffic rules accordingly as well as the system will detect if any accident occurs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

AI	Artificial Intelligence
API	Application Programming Interface
AV	Autonomous Vehicle
CNN	Convolution Neural Network
CV	Computer Vision
DC	Direct Current
GPS	Global Positioning System
GSM	Global System for Mobile Communication
Hz	Hertz
RNN	Recurrent Neural Network
SMS	Short Message Service
WHO	World Health Organization

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND THEORY

Self-driving cars, [1] also known as AV, are vehicles capable of navigating and operating without human intervention. They use a combination of advanced technologies, including sensors, computer vision, AI, and machine learning algorithms, to perceive their surroundings and make real-time decisions. By a true self driving car, we mean a car that can be essentially driven in any manner same as a human driving a car. For resolving the unsafe driving self-driving cars are introduced.

Self-driving cars were equipped with complex electromechanical parts and required a lot of requirement specification [2]. Just like the first self-driving model that was proposed by General Motors that was guided by radio controlled electromagnetic field. To drive such a car, you need to rebuild the roads that should contain magnetized metallic spikes. Then John McCarthy an inventor who invented the modern autonomous vehicles that is based on the artificial intelligence technology. John McCarthy is a researcher who described the idea of neural network and how a vehicle can capture the raw image from the road and control steering in real time environment. At present, more and more enterprises and scientific research institutions have invested in this field [3]. Google, Tesla, Apple, Nissan, Audi, General Motors, BMW etc. have participated in the research and development of self-driving cars. The development of self-driving cars has its roots in the field of robotics and AI research, which has been ongoing for several decades. However, recent advancements in technology, particularly in the areas of sensors and computing power have accelerated progress in this field.

Overall, self-driving cars represent a transformative technology that has the potential to reshape the future of transportation [4]. With continued advancements in technology, regulations, and public acceptance, self-driving cars can bring about safer, more efficient, and inclusive mobility solutions.

1.2 PROBLEM STATEMENT

According to WHO [5] Every year the lives of approximately 1.3 million people are cut short as a result of a road traffic crash. Between 20 and 50 million more people suffer non-fatal injuries, often resulting in long term disabilities. 90% of this accident happens due to human errors. One of the main challenges in the existing system of self-driving cars is the lack of infrastructure and regulations. Also, there are unpredictable traffic patterns that change in day-to-day life which causes many accidents. So, self-driving car was introduced to resolve such accidents as it never gets tired, distracted, impulsive or gets influenced by alcohol or drugs. It also enhances human productivity as they will be freed from tasks of managing the progress of the vehicle and has access of doing any productive activity.

1.3 OBJECTIVES

The main objectives of our project are:

- To demonstrate a self-automated vehicle that recognizes traffic signs and follows traffic rules.
- To detect accidents and send SMS with the current location of vehicle to the rescue team.

1.4 APPLICATION AND SCOPE OF THE PROJECT

Self-driving cars with accident detection also known as autonomous vehicles (AVs), have the potential to revolutionize transportation in numerous ways. Self-driving cars can be used as part of ride-sharing platforms, such as pathao, in-drive, to provide convenient, on-demand transportation. Self-driving cars can be used for the efficient and cost-effective delivery of goods. Companies like Amazon are exploring the use of autonomous vehicles for last-mile delivery, where packages can be transported directly from warehouses to customers' homes without the need for human drivers. Our project can also provide efficient and reliable public transportation services i.e., Self-driving technology can help optimize routes, reduce congestion, and improve accessibility, particularly in urban areas. Self-driving cars have the potential to greatly improve mobility for elderly and disabled individuals who may face challenges using traditional transportation options. One of the primary advantages of self-driving cars is their potential to significantly reduce accidents

caused by human error. We can also get an information about the accidents through the GPS and GSM modules.

The scope of this project is to help in reducing road accidents caused due to the human error. It helps in gaining productivity in work by utilizing time to do other tasks. This project can be also used to deliver goods in the future which will be cheaper as no human resource is needed. On proper implementation of the project, it can also improve the traffic conditions and scenarios.

1.5 ORGANIZATION OF PROJECT REPORT

Chapter 1 is the introductory part of this project report. It deals with the background, objectives, and application of the project. It also contains scope and problem statement of the project. **Chapter 2** deals with the literature review that describes the past works that were undertaken related to this project and also the components that were used in the past. **Chapter 3** deals with the conceptual design and outline of the project. **Chapter 4** talks about the working methods of various components used in the project and it also talks about the total working of the system through various algorithms and flowcharts. And, finally the **Chapter 5** includes the result and analysis portion of this report.

CHAPTER 2: LITERATURE REVIEW

In the current landscape of autonomous vehicle research, our project focuses on the development of a self-driving car with a specialized emphasis on accident detection capabilities.

This initiative draws inspiration from various studies within the field, with a notable reference to the work by [6], they introduce an autonomous car prototype where Raspberry Pi serves as the central processing unit. This innovative system seamlessly integrates OpenCV and machine learning technologies for autonomous navigation, eliminating the need for human intervention. The Raspberry Pi, connected to a Pi camera module, facilitates real-time video streaming to a local host monitor, forming the basis for intricate detections like pedestrians, vehicles, and road signs. Subsequent commands are sent serially to an Arduino unit, orchestrating autonomous car movement. Equipped with functionalities like traffic signal detection, vehicle detection, and road sign recognition, the Raspberry Pi ensures cautious and timely navigation. Implementation involves programming in C++, incorporating methods such as Gaussian Blur, Canny Edge Detection, and Region of Interest. The physical assembly encompasses robot body construction, motor soldering, and precise installation of components like Raspberry Pi, Arduino Uno, and motor driver. This work not only draws from established methodologies but also lays the groundwork for the ambitious integration of diverse technologies, advancing self-driving car capabilities.

In the insightful paper by [7], the intricacies of self-driving cars are explored, with a specific focus on color-based road detection through image processing techniques. Emphasizing the pivotal role of feature extraction, the study details how processing occurs on each video frame, treating it as a sequence captured at minimal instants. It underscores the significance of interpreting colored pixels in a two-dimensional mesh for effective frame manipulation. The paper delves into developers' strategies, highlighting the use of Hough transformation for lane detection. Through this method, raw images are transformed into edges using the Canny method, and collinear points contribute to the creation of lines representing detected lanes. This concise review encapsulates the nuanced approach to self-driving car navigation, underscoring the importance of image processing in shaping autonomous vehicle technologies.

Before canny edge detection to improve edges, another approach proposed by [8] is by applying machine learning algorithms that detects the feature sets of raw frames to create dataset of image and steering angle and train data using that dataset. On the basis of trained model, the values of steering are predicted by comparing each image with the path. This method can only do prediction for a path that is already followed and trained and cannot run on new areas. We need to put in every single road to train model that we need to use in our daily life because it can only predict the steering angle according to image data. Also, it makes it slower as it has to compare matrices of arrays to train a model. And also, it takes in a lot of data to train.

In the innovative proposal by [9], the authors address a critical issue in road safety—the delay in medical assistance following accidents, a leading cause of fatalities. Their proposed system leverages GPS, GSM, accelerometer, and Arduino to proactively mitigate this challenge. The system is designed to promptly notify authorities, including the nearest hospital and police headquarters, along with notifying family and friends in the event of an accident. Detection of changes in accelerometer readings serves as the trigger, prompting the system to send a Google Maps link via the GPS module and Arduino. Notably, the Arduino sets a flag bit upon identifying an accident, continuously monitoring for abrupt deviations from threshold values. The effective sensitivity of measuring instrument detectors is dynamically adjusted during an accident, unless a crash is detected. Once an accident is identified, the GSM module is activated, sending a pre-stored SMS to the manually saved emergency contact of the accident victim. This forward-thinking system not only addresses a significant road safety concern but also showcases the integration of multiple technologies to provide timely and automated emergency response in the critical aftermath of accidents.

CHAPTER 3: RELATED THEORY

3.1 HARDWARE SPECIFICATIONS:

For the implementation of any system, hardware and software components are the essential parts. The integral parts of the hardware that we will use in our minor project are: -

1. Raspberry Pi 4
2. Arduino UNO
3. Raspicam V2
4. L298N motor driver
5. Gear Motor
6. Sim 900A GSM module
7. Neo-6M GPS module
8. MPU-6050 accelerometer and gyroscope

RASPBERRY PI 4

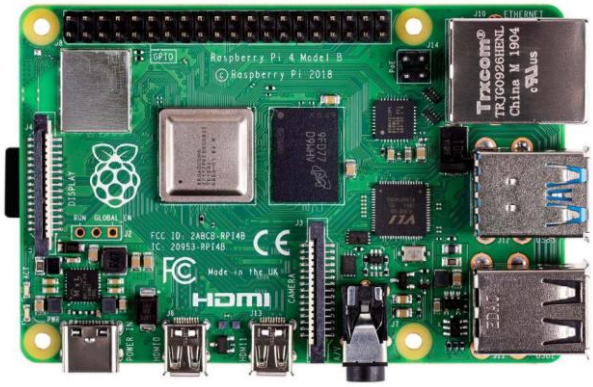

Figure 3.1 Raspberry pi 4 [10]

The Raspberry Pi 4 signifies a substantial evolution in single-board computing. Driven by the BCM2711 SoC, it incorporates a quad-core ARM Cortex-A72 processor with speeds of up to 1.5 GHz, representing a notable performance enhancement from its predecessors. Available with 2GB, 4GB, or 8GB of LPDDR4 RAM, the Raspberry Pi 4 facilitates efficient multitasking and supports a broader spectrum of applications. Notable improvements in connectivity include Gigabit Ethernet, dual-band Wi-Fi, and Bluetooth

5.0 capabilities. The introduction of USB 3.0 ports further extends peripheral options. In keeping with its tradition, the Raspberry Pi 4 provides 40 GPIO pins, allowing seamless interfacing with electronic components for a wide range of project possibilities. Embraced across diverse domains, including home automation, robotics, IoT applications, and education, the Raspberry Pi 4 stands out as a versatile and powerful single-board computer.

ARDUINO UNO

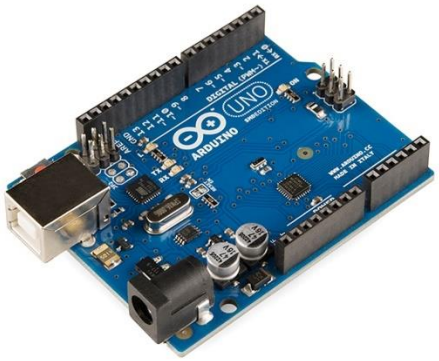

Figure 3.2 Arduino UNO [11]

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 (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

RASPICAM V2

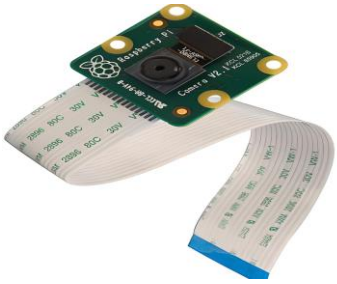

Figure 3.3 Raspicam V2 [27]

The Camera Module 2 can be used to take high-definition video, as well as stills photographs. It is easy to use for beginners, but has plenty to offer. The specifications are as follows:

- Improved resolution- 8-megapixel native resolution sensor-capable of 3280*2464-pixel static images.
- Supports 1080p30,720p60 and 640*480p90 video
- Size 25mm*23mm*9mm
- Weight just over 3g
- Connects to the Raspberry Pi board via a short ribbon cable (supplied)
- Uses the Sony IMX219PQ image sensor- high-speed video imaging and high sensitivity
- 1.4m*1.4m pixel with Omni BSI technology for high performance (high sensitivity, low crosstalk)

L298N MOTOR DRIVER

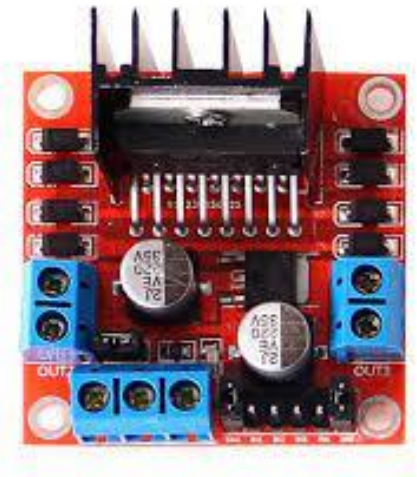

Figure 3.4 L298N Motor Driver [12]

The 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 specifications are as follows:

- Driver chip: Double H Bridge L298N
- Motor Supply Voltage and current (maximum):46v,2A
- Maximum Power(W):25W
- Power-On LED indicator

GEAR MOTOR

Figure 3.5 Gear Motor [12]

A gear motor, also called a geared motor, is a combination of a gear system or gear box and an electric motor. The specifications are as follows:

- Magnet types: Ferrite magnets
- Operating Voltage:3V-9V
- Motor Voltage:12V
- RPM:100RPM or 150RPM

SIM 900 GSM MODULE

Figure 3.6 SIM 900 GSM Module [13]

SIM900 GSM Module is the module for GPRS/GSM communication. It is common with Arduino and microcontroller in most of embedded application. The module offers GPRS/GSM technology for communication with the uses of a mobile sim. It uses a 900 and 1800MHz frequency band and allows users to receive/send mobile calls and SMS. The keypad and display interface allow the developers to make the customize application with it. Furthermore, it also has modes, command mode and data mode. In every country the GPRS/GSM and different protocols/frequencies to operate. Command mode helps the developers to change the default setting according to their requirements.

NEO 6M GPS MODULE

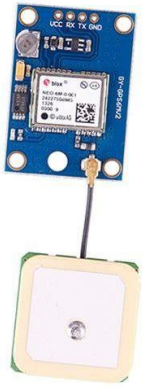

Figure 3.7 NEO 6M GPS Module [14]

[14]Neo 6M GPS module is a highly sensitive and low power module offering 56 channels for precise position which updates at a rate of 10Hz. The module is a comprehensive GPS solution based on the Neo 6M technology. It seamlessly integrates with Arduino to provide accurate GPS data.

MPU 6050 ACCELEROMETER AND GYROSCOPE

Figure 3.8 MPU 6050 Accelerometer and Gyroscope [15]

MPU-6050 6-DoF Accelerometer and Gyroscope Sensor is a triple-axis accelerometer and gyroscope combination. Inside the breakout is two sensors, one is a 3-axis accelerometer, which can tell the user which direction is down towards the earth (by measuring gravity).

3.2 SOFTWARE SPECIFICATIONS:

The integral parts the software that we will use in our minor project are: -

1. Arduino IDE
2. Raspbian OS
3. Cascade trainer GUI
4. Open CV with Python

ARDUINO IDE:

Arduino IDE is used to provide the backend functionality to the system, a communication is established between Arduino module and other components. Arduino IDE uses C language as the programming language to interface between user.

RASPBIAN OS:

Raspberry Pi OS is a free, open-source Debian Linux-based operating system engineered for use on Pi boards. Additionally, several ARM-based single-board computers also run Raspberry Pi OS. The first version, then known as Raspbian, debuted in 2013, and from 2015 onwards the Raspberry Pi Foundation offered it as an officially-sanctioned Pi distro.

CASCADE TRAINER GUI:

Cascade classifiers are trained using several positive (with faces or objects) images and arbitrary negative (without faces or objects) images. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is a program which simplifies the training process of the cascade classifiers. Not only training but also testing and improving can also be easily done with this program. To start the training, we need to create a folder for your classifier and create two folders inside it. One should be “p” (for positive images) and the other should be “n” (for negative images). Positive image samples are the images of the object we want to train your classifier and detect. For example, if we want to train and detect cars then we need to have many car images which are the positive samples and negative images are the surrounding environment except cars.

Open CV with Python:

Open CV basically is a library of programming functions mainly dedicated to computer vision or we can see image processing. This library is cross platform that means it can be implemented for multiple different platforms. Most importantly Open CV also supports machine learning. Open CV module can be used to train the cascade classifier. Various applications are facial recognition, gesture recognition, motion understanding, object identification/ detection, deep neural network, etc. This project will be done in open cv in python. Python, a high-level, general-purpose programming language developed in the late 1980s, offers many features and capabilities suitable for applications such as web development, scientific computing, data analysis, artificial intelligence, and automation. With an extensive standard library and support for numerous third-party libraries and frameworks, Python facilitates web development, scientific computing, machine learning and natural language processing. Python's interpreted nature enables rapid development and testing, making it highly accessible. It is available on multiple platforms, including Windows, macOS, and Linux, ensuring high portability.

CHAPTER 4: METHODOLOGY

4.1 SYSTEM BLOCK DIAGRAM FOR SELF DRIVING CAR

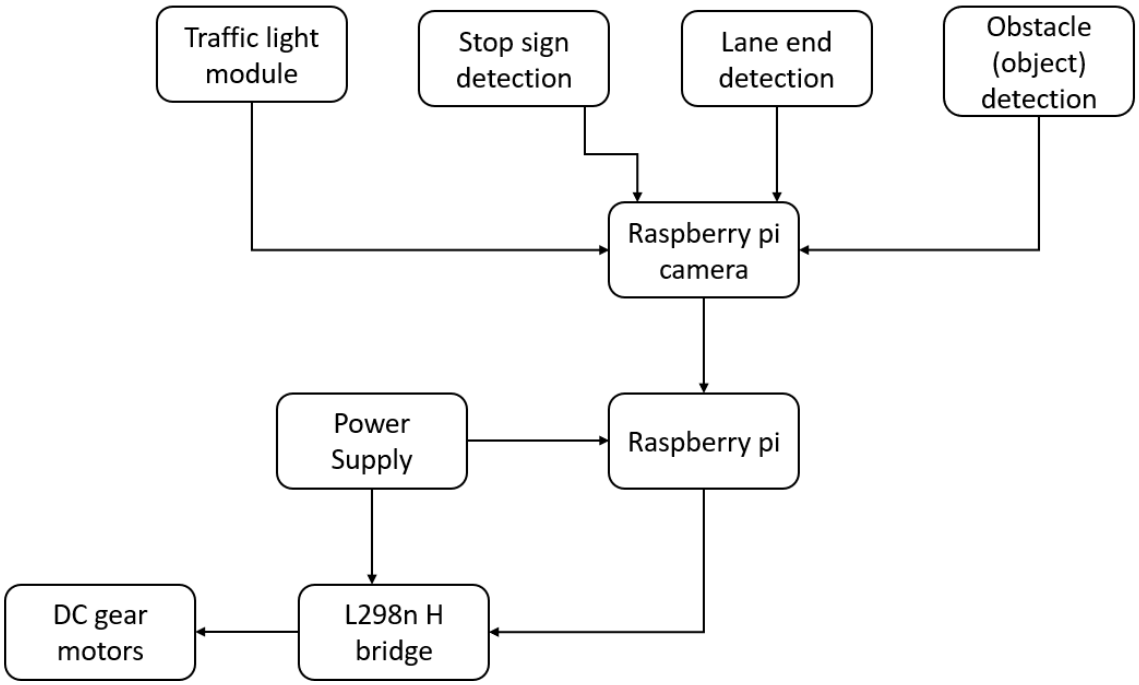

Figure 4.1 Block Diagram for Self-Driving Car

The figure 4.1 is the block diagram of self-driving car. It consists of raspberry pi camera, raspberry pi, lipo battery, l298n H-bridge and DC gear motors. Raspberry Pi camera is used to detect the instruction for traffic light module, stop sign detection, lane end detection and obstacle. This instruction is passed to Raspberry Pi which is powered by a power supply, which then send instructions to the motor driver. Through those instructions the gear motors move respectively.

4.2 SYSTEM BLOCK DIAGRAM FOR ACCIDENT DETECTION

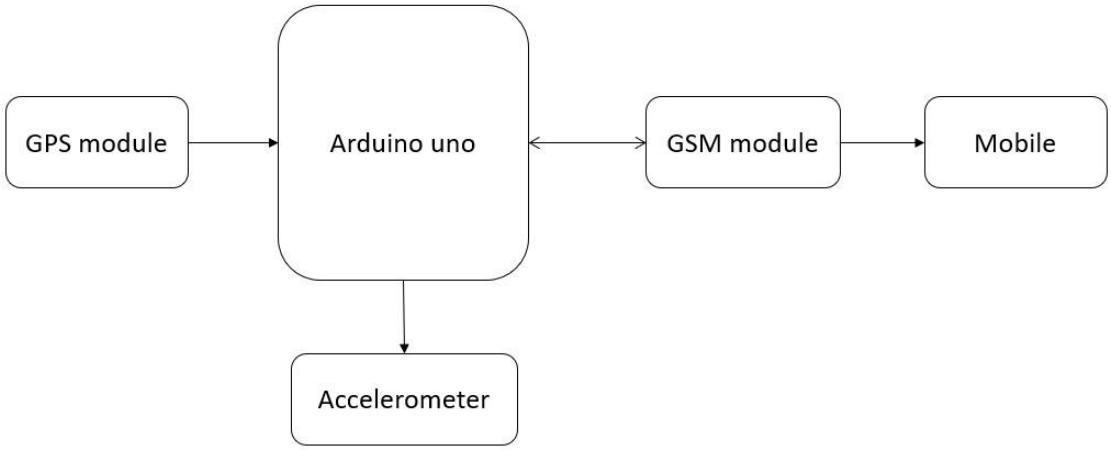

Figure 4.2 Block diagram for accident detection

The accelerometer/gyroscopic sensor measure's the proper calibration of car. If an accident occurs then GPS (a satellite-based navigation system) provides location and time. This information is sent to the Arduino Uno which is connected with GSM module for wireless data link with mobile phone.

4.3 ALGORITHM AND FLOWCHART

4.3.1 Algorithm for lane detection and following system:

Step 1: Start

Step 2: Initiate Raspberry Pi

Step 3: Pi camera captures image and sends to Raspberry Pi

Step 4: Raspberry Pi sends signal Motor driver turns Motor on

Step 5: If camera detects lane go to step 6

Step 6: Move forward

Step 7: if lane ends go to step 8

Step 8: Move to the index specified and go to step 5

Step 9: Stop

4.3.2 Flowchart for lane detection and following system:

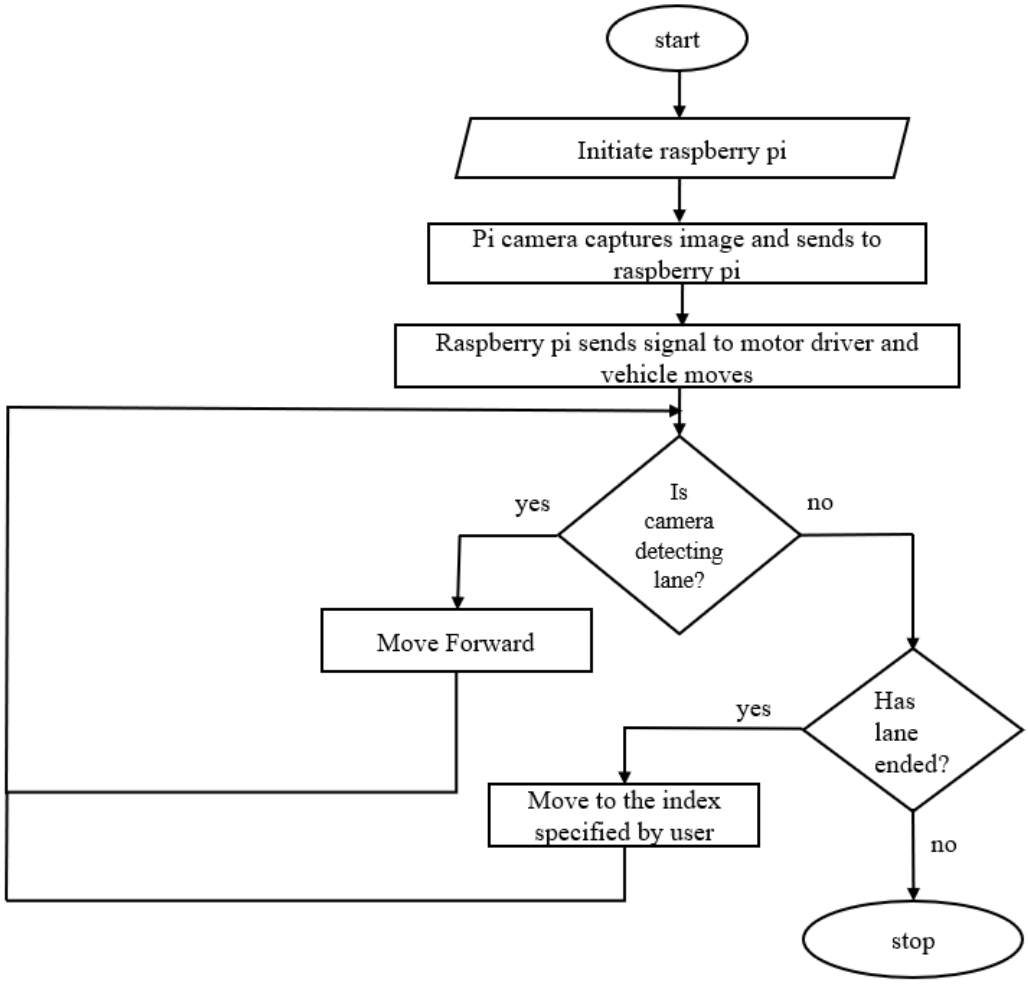

Figure 4.3.2 Flow chart for lane detection and following system

4.3.3 Algorithm for accident detection

Step 1: Start

Step 2: Detect the accident

Step 3: If panic button is on go to step 4 else step 7

Step 4: Send victim details

Step 5: Track the location of the victim

Step 6: Send message to rescue team

Step 7: Abort request to send SMS

Step 8: Stop

4.3.4 Flowchart for accident detection:

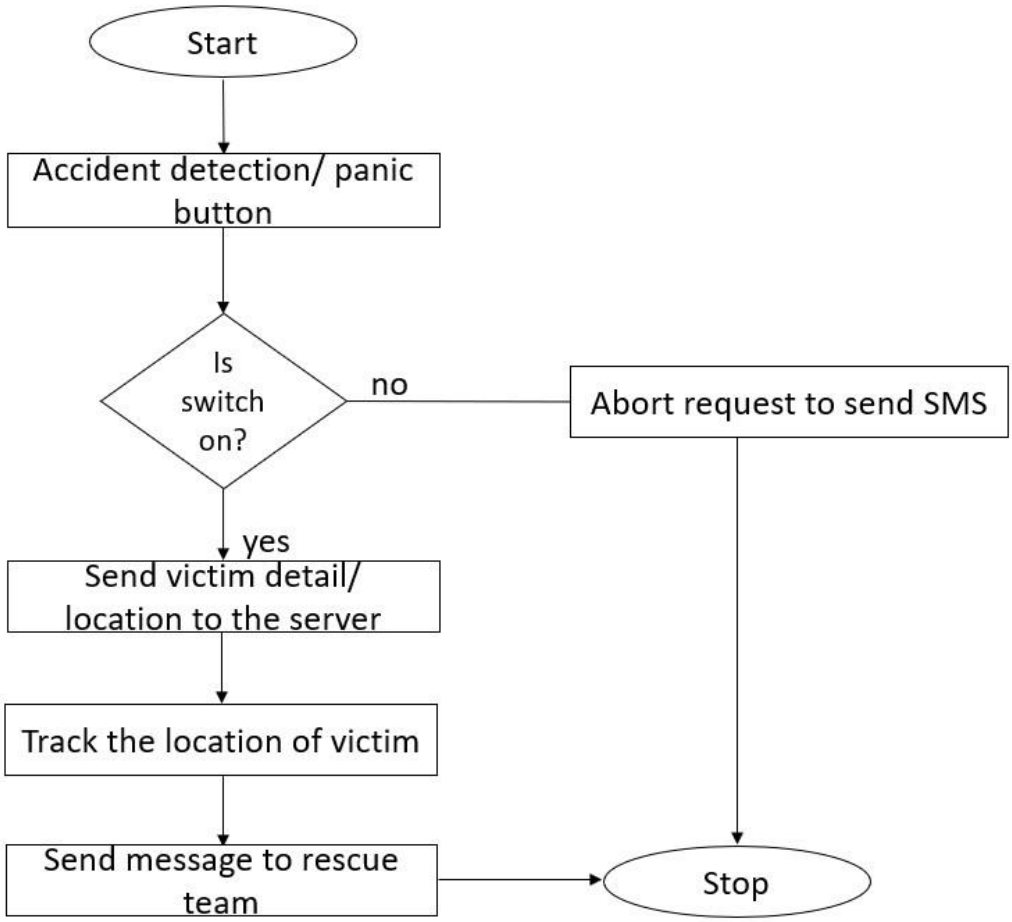

Figure 4.3.4 Flowchart for accident detection

4.3.5 Algorithm for traffic sign and object detection

Step 1: Start

Step 2: Read the positive and negative samples

Step 3: Train the module through neural network

Step 4: Detection of objects through image classification and training

Step 5: Send signals back to the model

Step 6: Stop

4.3.6 Flowchart for traffic sign and object detection

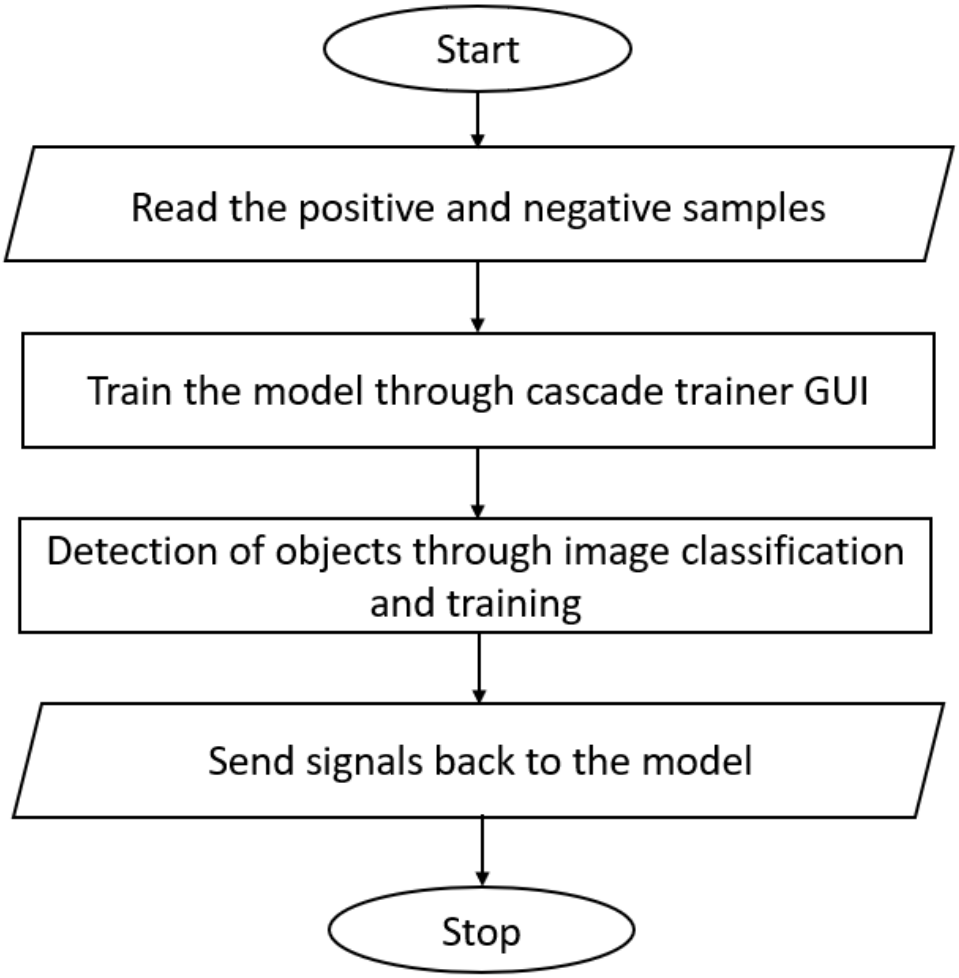

Figure 4.3.6 Flowchart for traffic sign and object detection

4.4 HAAR CASCADE CLASSIFIER ALGORITHM

[16] Haar Cascade is a feature-based object detection algorithm to detect objects from images. A cascade function is trained on lots of positive and negative images for detection. The algorithm does not require extensive computation and can run in real-time.

The algorithm can be explained in four stages:

- Calculating Haar Features
- Creating Integral Images
- Using Adaboost
- Implementing Cascading Classifiers

It is important to remember that this algorithm requires a lot of positive images of faces and negative images of non-faces to train the classifier, similar to other machine learning models.

4.4.1 CALCULATING HAAR FEATURES

The first step is to collect the Haar features. A Haar feature is essentially calculations that are performed on adjacent rectangular regions at a specific location in a detection window. The calculation involves summing the pixel intensities in each region and calculating the differences between the sums. Here are some examples of Haar features below.

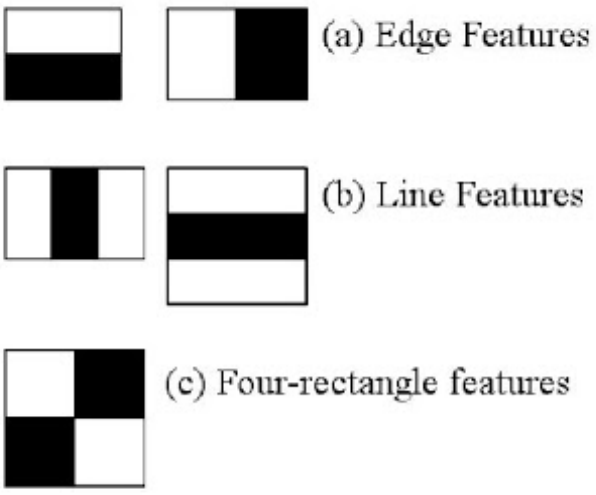

Figure4.4.1: Types of Haar features. *[16]*

These features can be difficult to determine for a large image. This is where integral images come into play because the number of operations is reduced using the integral image.

4.4.2 INTEGRAL IMAGES

Without going into too much of the mathematics behind it, integral images essentially speed up the calculation of these Haar features. Instead of computing at every pixel, it instead creates sub-rectangles and creates array references for each of those sub-rectangles. These are then used to compute the Haar features.

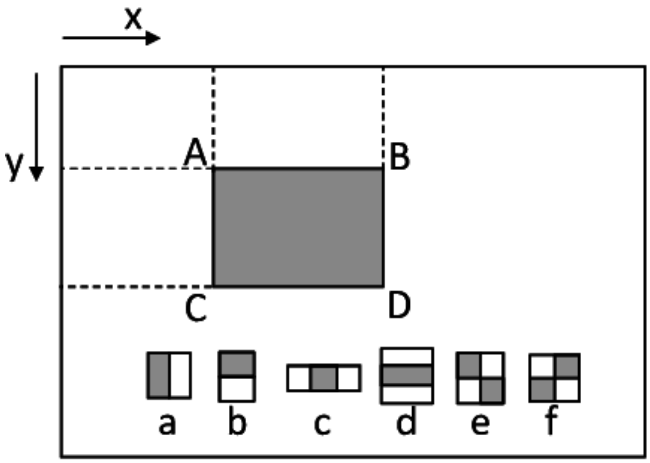

Figure4.4.2: Illustration for how an integral image works. *[16]*

It is important to note that nearly all of the Haar features will be irrelevant when doing object detection, because the only features that are important are those of the object. However, how do we determine the best features that represent an object from the hundreds of thousands of Haar features? This is where Adaboost comes into play.

4.4.3ADABOOST TRAINING

Adaboost essentially chooses the best features and trains the classifiers to use them. It uses a combination of "weak classifiers" to create a "strong classifier" that the algorithm can use to detect objects.

Weak learners are created by moving a window over the input image, and computing Haar features for each subsection of the image. This difference is compared to a learned threshold that separates non-objects from objects. Because these are "weak classifiers," a large number of Haar features is needed for accuracy to form a strong classifier.

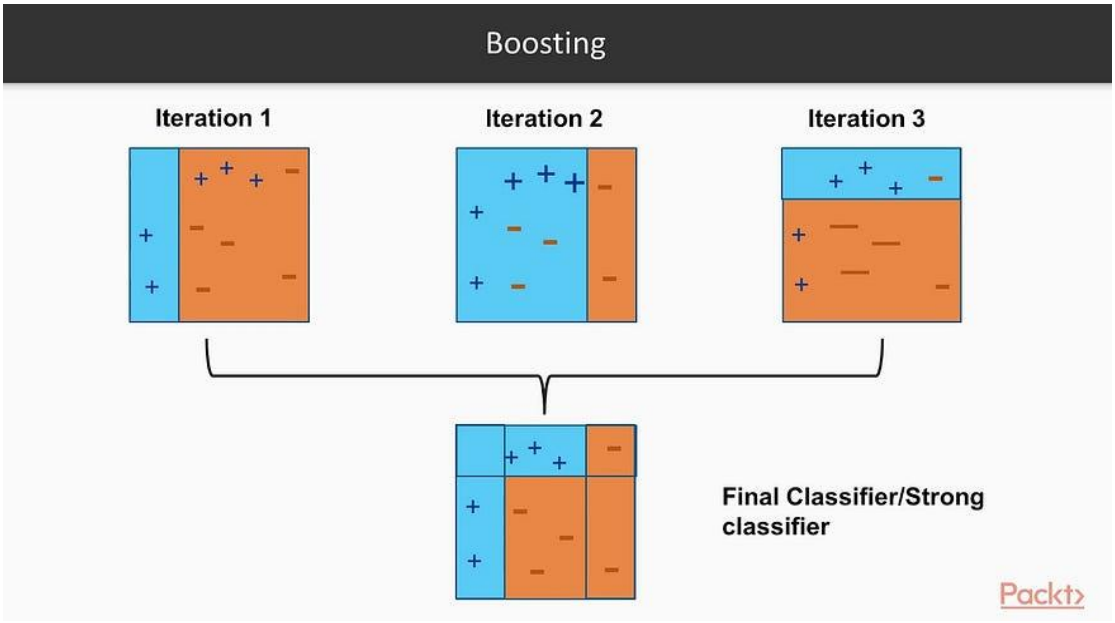

Figure 4.4.3: Representation of a boosting algorithm. *[16]*

The last step combines these weak learners into a strong learner using cascading classifiers.

4.4.4 IMPLEMENTING CASCADING CLASSIFIERS

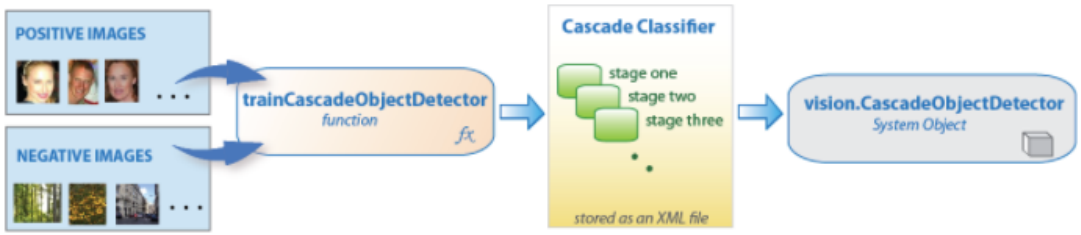

Figure 4.4.4: A flowchart of cascade classifiers *[16]*

The cascade classifier is made up of a series of stages, where each stage is a collection of weak learners. Weak learners are trained using boosting, which allows for a highly accurate classifier from the mean prediction of all weak learners.

Based on this prediction, the classifier either decides to indicate an object was found (positive) or move on to the next region (negative). Stages are designed to reject negative samples as fast as possible, because a majority of the windows do not contain anything of interest.

It is important to maximize a low false negative rate, because classifying an object as a non-object will severely impair your object detection algorithm.

4.5 CANNY EDGE DETECTION ALGORITHM

[17]Canny edge detection is a technique to extract useful structural information from different vision objects and dramatically reduce the amount of data to be processed. It has been widely applied in various computer vision systems. Canny has found that the requirements for the application of edge detection on diverse vision systems are relatively similar. Thus, an edge detection solution to address these requirements can be implemented in a wide range of situations. The general criteria for edge detection include:

1. Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
2. The edge point detected from the operator should accurately localize on the center of the edge.
3. A given edge in the image should only be marked once, and where possible, image noise should not create false edges.

To satisfy these requirements Canny used the calculus of variations – a technique which finds the function which optimizes a given functional. The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian.

Among the edge detection methods developed so far, Canny edge detection algorithm is one of the most strictly defined methods that provides good and reliable detection. Owing to its optimality to meet with the three criteria for edge detection and the simplicity of process for implementation, it became one of the most popular algorithms for edge detection.

Processes for Canny edge detection:

The process of Canny edge detection algorithm can be broken down to five different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

4.5.1 Gaussian filter

Since all edge detection results are easily affected by the noise in the image, it is essential to filter out the noise to prevent false detection caused by it. To smooth the image, a Gaussian filter kernel is convolved with the image. This step will slightly smooth the image to reduce the effects of obvious noise on the edge detector. The equation for a Gaussian filter kernel of size $(2k+1)\times(2k+1)$ is given by:

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

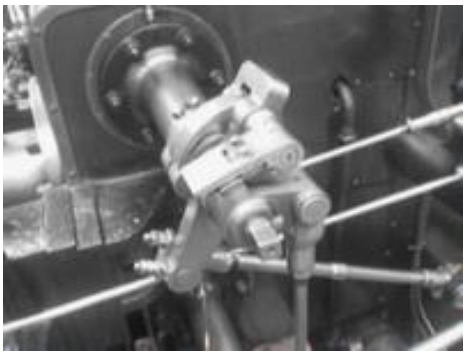

Figure 4.5.1 Image after 5*5 Gaussian mask has been passed across each pixel *[17]*

4.5.2 Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (*Gx*) and vertical direction (*Gy*). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

4.5.3 Gradient magnitude thresholding or lower bound cut-off suppression

Minimum cut-off suppression of gradient magnitudes, or lower bound thresholding, is an edge thinning technique. Lower bound cut-off suppression is applied to find the locations with the sharpest change of intensity value. The algorithm for each pixel in the gradient image is:

1. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
2. If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (e.g., a pixel that is pointing in the y-direction will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

In some implementations, the algorithm categorizes the continuous gradient directions into a small set of discrete directions, and then moves a 3x3 filter over the output of the previous step (that is, the edge strength and gradient directions). At every pixel, it suppresses the edge strength of the center pixel (by setting its value to 0) if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction. For example,

- if the rounded gradient angle is 0° (i.e. the edge is in the north–south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the east and west directions,

- if the rounded gradient angle is 90° (i.e. the edge is in the east–west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north and south directions,
- if the rounded gradient angle is 135° (i.e. the edge is in the northeast–southwest direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north-west and south-east directions,
- if the rounded gradient angle is 45° (i.e. the edge is in the northwest–southeast direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes at pixels in the north-east and south-west directions.

In more accurate implementations, linear interpolation is used between the two neighboring pixels that straddle the gradient direction. For example, if the gradient angle is between 89° and 180°, interpolation between gradients at the north and north-east pixels will give one interpolated value, and interpolation between the south and south-west pixels will give the other. The gradient magnitude at the central pixel must be greater than both of these for it to be marked as an edge.

4.5.4 Double threshold

After application of non-maximum suppression, remaining edge pixels provide a more accurate representation of real edges in an image. However, some edge pixels remain that are caused by noise and color variation. To account for these spurious responses, it is essential to filter out edge pixels with a weak gradient value and preserve edge pixels with a high gradient value. This is accomplished by selecting high and low threshold values. If an edge pixel’s gradient value is higher than the high threshold value, it is marked as a strong edge pixel. If an edge pixel’s gradient value is smaller than the high threshold value and larger than the low threshold value, it is marked as a weak edge pixel. If an edge pixel's gradient value is smaller than the low threshold value, it will be suppressed. The two threshold values are empirically determined and their definition will depend on the content of a given input image.

4.5.5 Hysteresis Thresholding

[18]This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so

discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

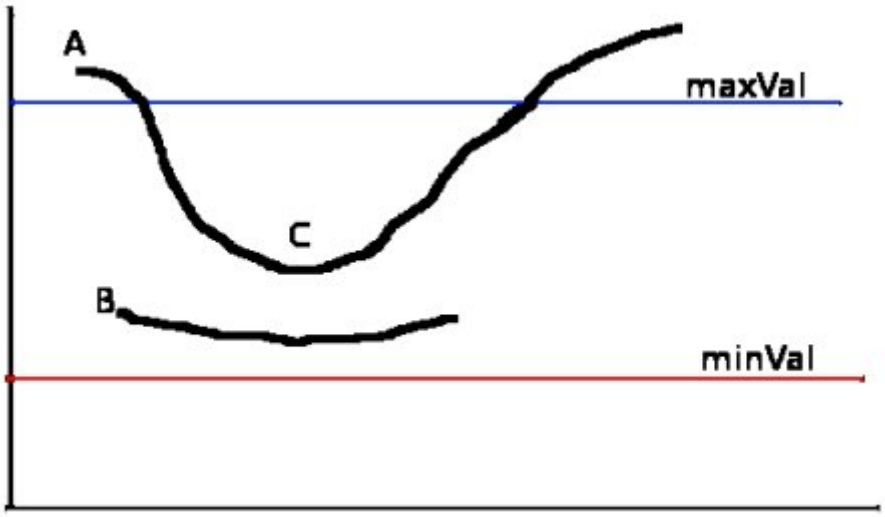

Figure 4.5.5 Hysteresis Thresholding *[18]*

The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So, it is very important that we have to select minVal and maxVal accordingly to get the correct result. This stage also removes small pixels noises on the assumption that edges are long lines. So, what we finally get is strong edges in the image.

4.5.6 Walkthrough of the algorithm

This section will show the progression of an image through each of the five steps.

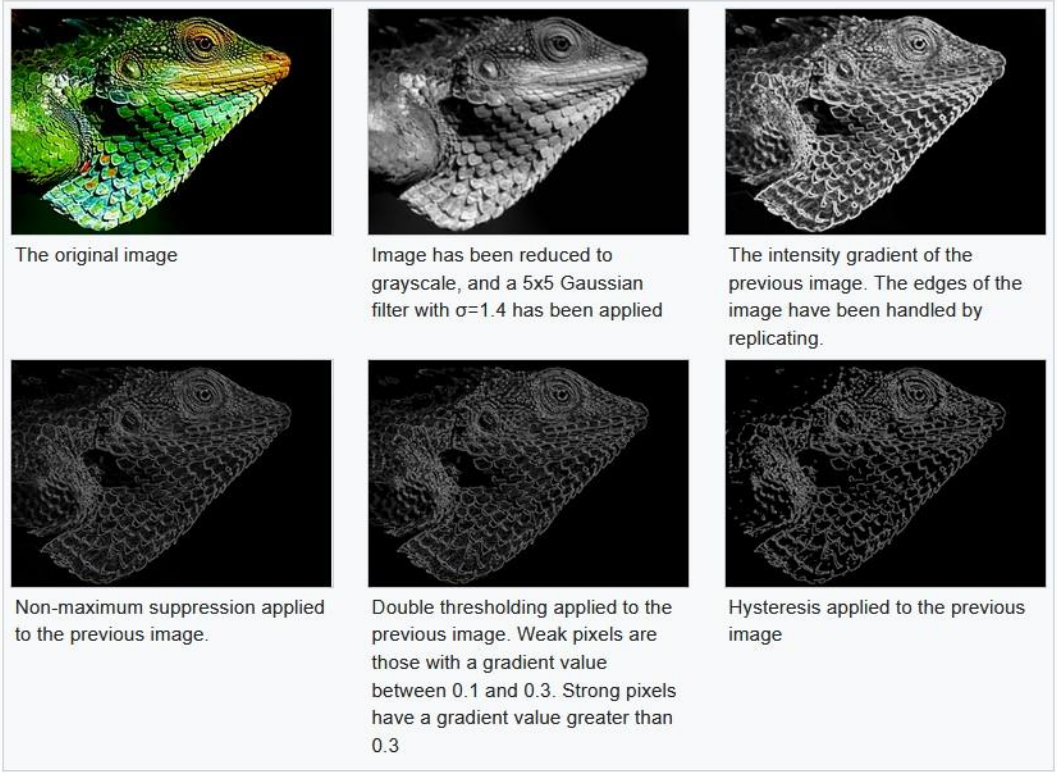

Figure 4.5.6 Results after Canny Edge Detection is applied *[17]*

CHAPTER 5: RESULTS AND ANALYSIS

5.1 RESULTS:

The goal of this project was to create a “Self-driving car with Accident Detection” using, raspberry pi and if any accident occurs then a message is sent to the rescue team through GPS/GSM modules.

The self-driving car was made of a thin sun board making it light portable and easy to move and the components were easily adjusted in the chassis. The use of sun board in the structure of the vehicle provides versatility and ease of fabrication. It can be easily cut, shaped, and assembled, allowing for the creation of intricate designs. The material is also cost-effective, making it an attractive option for prototyping and experimental vehicle structures.

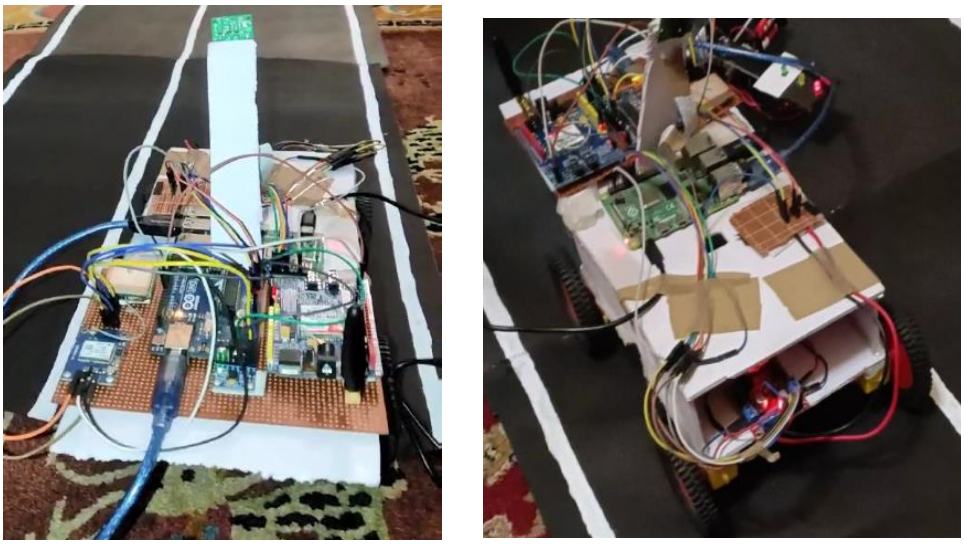

Figure 5.1.1 Hardware Structure of vehicle

Then, the implementation of lane detection and following system was done, which was done through the Canny Edge Detection algorithm. Canny edge detection, renowned for its accuracy in identifying edges within images, is employed to precisely detect lane boundaries on the road. This system leverages the algorithm's ability to highlight edges with significant intensity changes, allowing the identification of distinct lane markings. In real-time, the vehicle's onboard processing unit utilizes this information to establish a clear understanding of the road's structure. By continuously tracking the detected lane edges, the vehicle is equipped to autonomously follow the designated path. This technology, combining computer vision with the Canny edge detection algorithm, contributes to the development of advanced driver-assistance systems, offering a reliable solution for

automated lane detection and following in various driving scenarios. The output observed after implementing the lane detection and following system is:

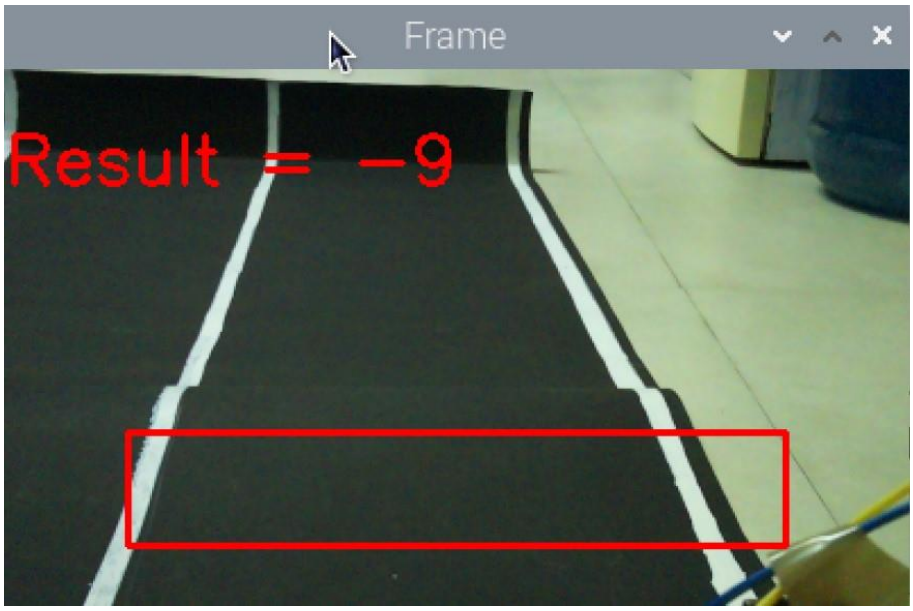

Figure 5.1.2 Initial Frame Captured

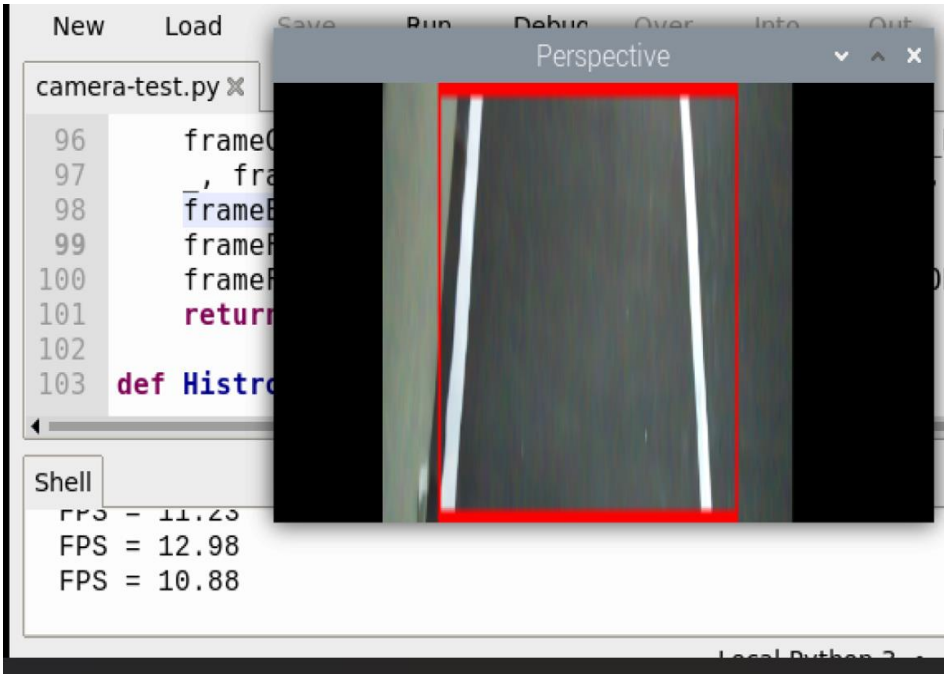

Figure 5.1.3 Perspective image

Figure 5.1.4 Final Image after implementing Canny

After this implementation of Canny Edge detection algorithm, the lane was properly followed by the vehicle and the vehicle will stop when a lane end is detected which is a white strip across the whole lane which is shown below:

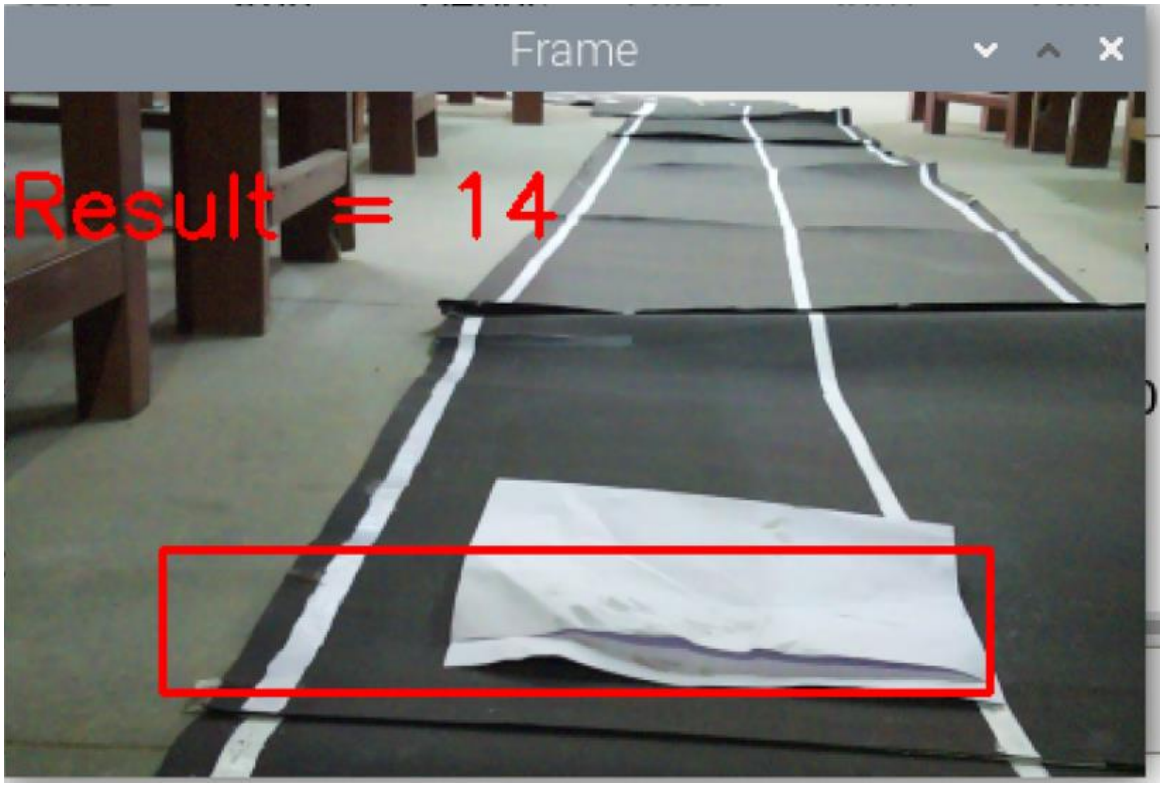

Figure 5.1.5 Lane End detection in original frame

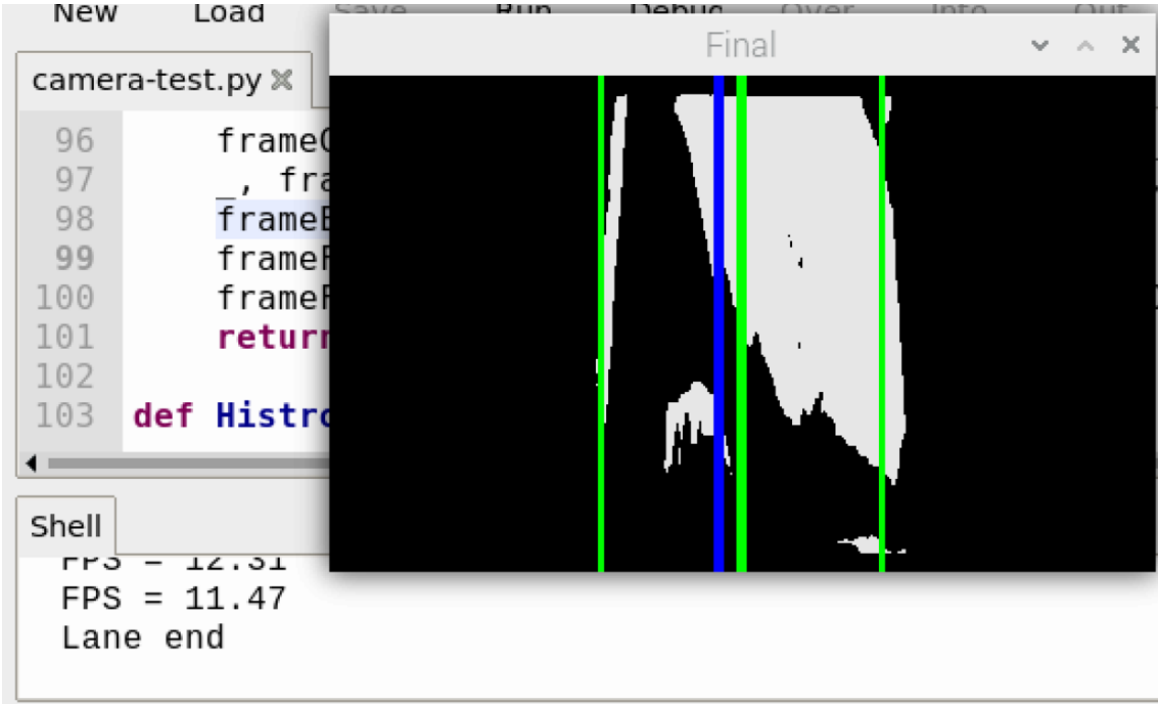

Figure 5.1.6 Lane End Detection on final frame

Then, the detection of object, stop sign, and traffic sign were done in order to make the vehicle autonomous, and these detections were done through the Haar Cascade Algorithm. At first the stop signs and traffic signs models were implemented which are:

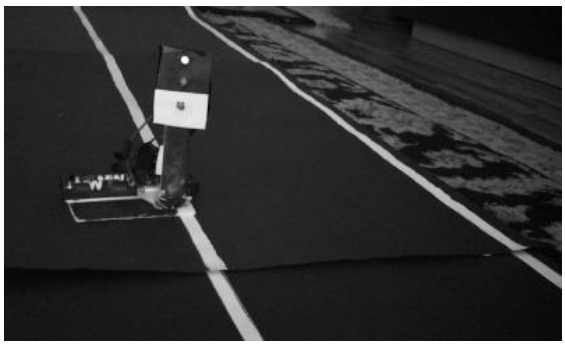

Figure 5.1.7 Traffic Sign Model

Figure 5.1.8 Stop Sign Model

Figure 5.1.9 Object Model

Then these all were detected using Haar Cascade Algorithm where the Cascade trainer GUI creates a 'XML' file and these were implemented in the vehicle through which all these were detected in real time.

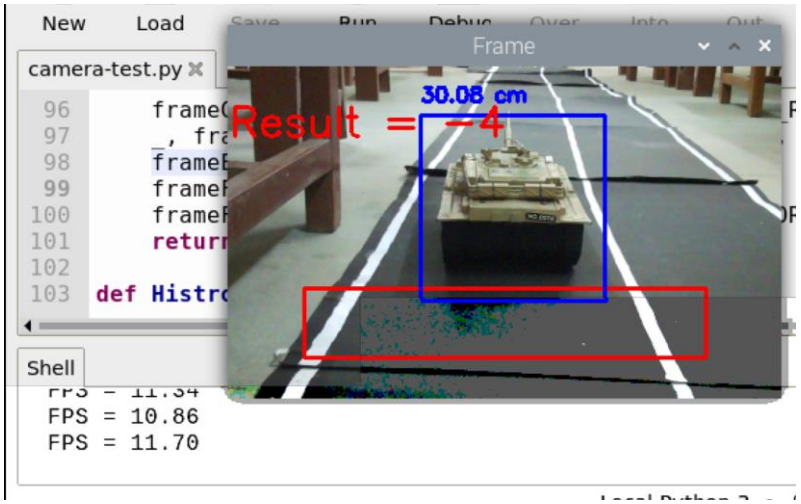

Figure 5.1.10 Object Detected when it is far

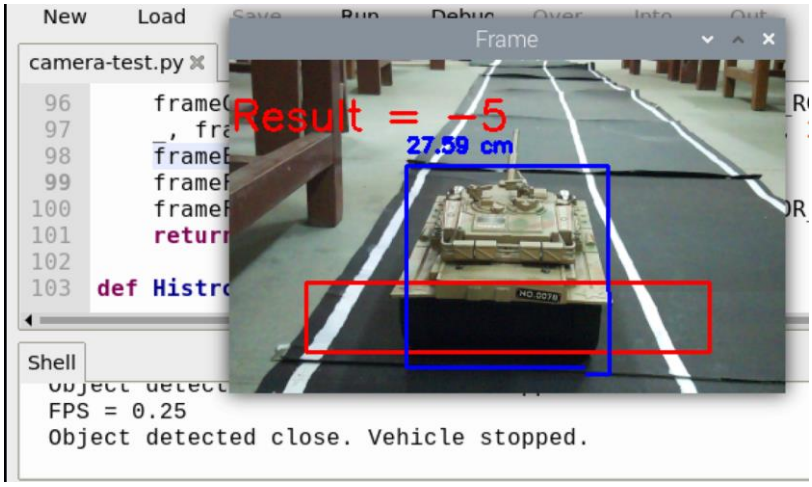

Figure 5.1.11 Object Detected near and stopped

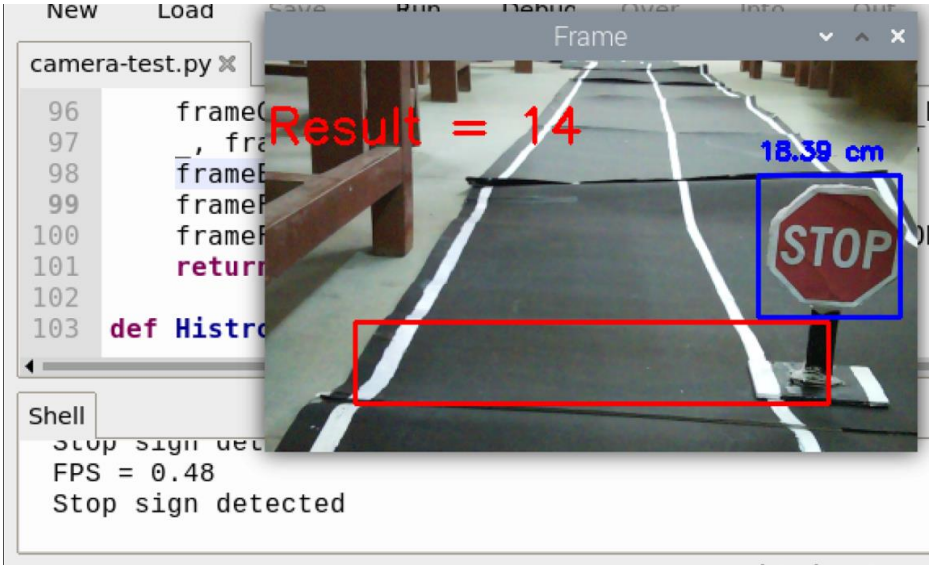

Figure 5.1.12 Detection of Stop Sign

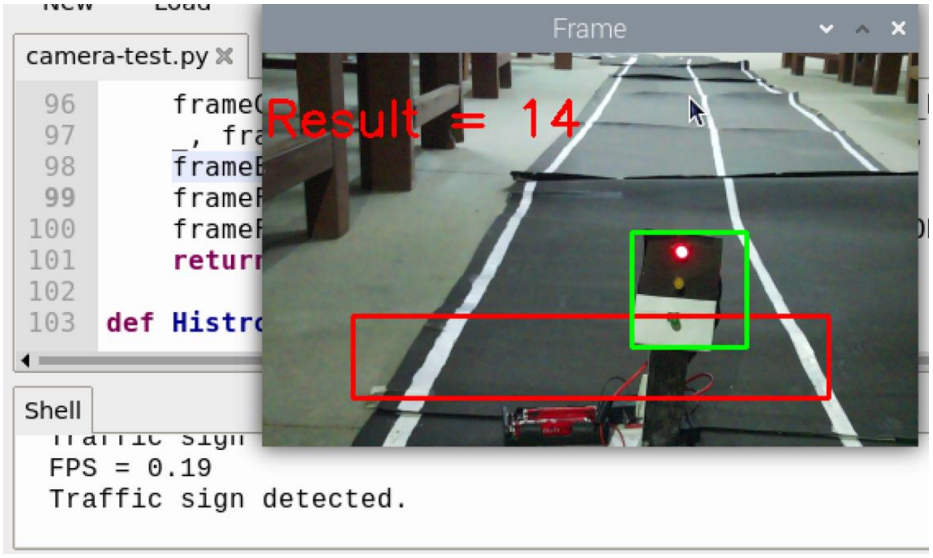

Figure 5.1.13 Detection of traffic sign when light is red

Figure 5.1.14 No Detection when traffic light is green

Finally, after all the implementation of detection algorithms of the self-driving car, the accident detection was implemented in Arduino Uno and it was connected to the GPIO pin of raspberry pi and the Arduino Uno will send an SMS to the rescue team if an accident has occurred and also the vehicle will stop completely.

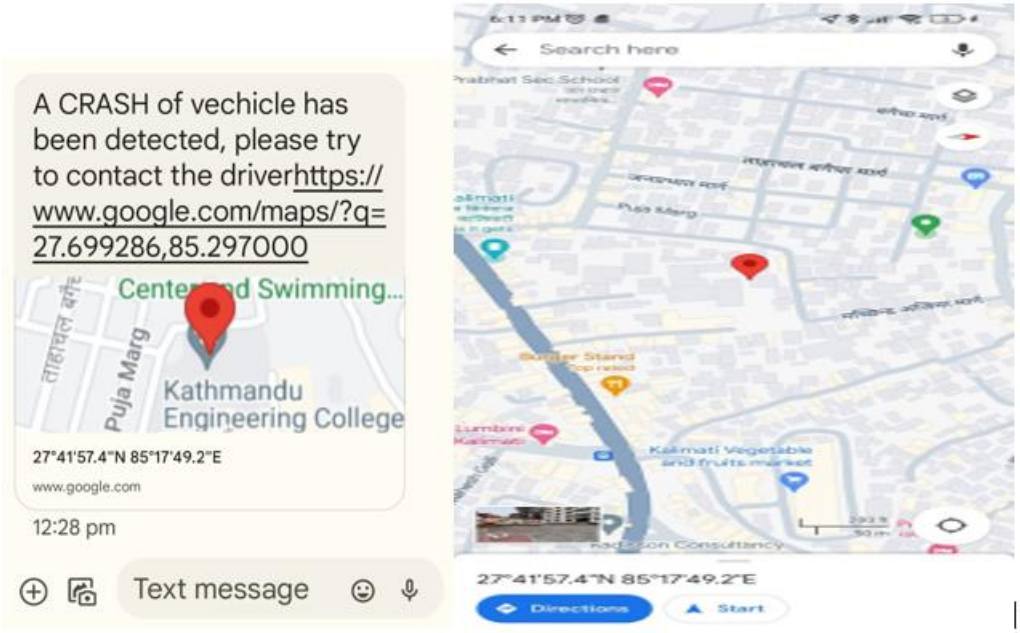

Figure 5.1.15 Accident Detection message in the rescue number

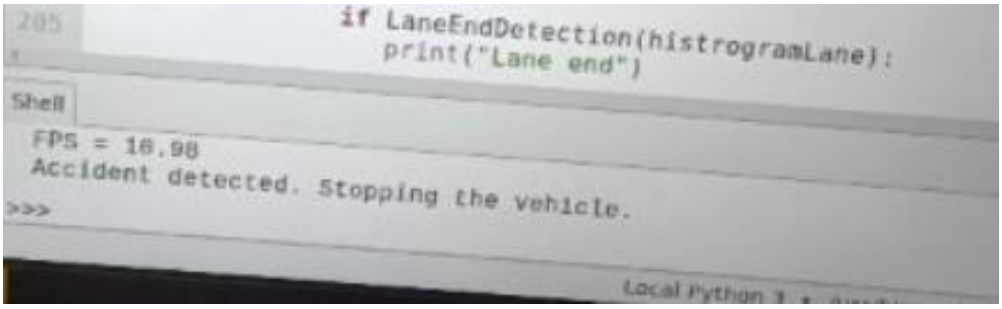

Figure 5.1.16 Accident Detection message in raspberry pi

5.2 PROBLEMS ENCOUNTERED:

We faced a lot of problems and challenges in our project. Some of them are:

5.2.1 Power Supply for GSM module:

The power supply for the GSM module is enough when giving from the 5v/2A DC adapter but when giving it from the power bank or the lipo battery the GSM module could not connect to the network.

5.2.2 Instability of Camera:

Due to the vibrations on the camera module during the movement of vehicle, the raspberry pi could not detect the various modules like stop sign, traffic light, and objects properly.

5.3 FUTURE ENHANCEMENTS:

5.3.1 Real-time Mapping and Localization:

Implementing real-time mapping and localization features to improve the system's understanding of its position on the road, contributing to more accurate lane following, especially in complex urban environments.

5.3.2 Using GPS Co-ordinates:

Using the proper GPS co-ordinates, the vehicle could move in the appropriate locations as wanted by the user.

5.3.3 Lane Change Assistance:

Extend the system to include lane change assistance features, allowing the vehicle to safely navigate and signal lane changes based on the surrounding traffic conditions.

5.3.4 Machine Learning for Decision Making:

Incorporate machine learning algorithms to enable the system to learn from patterns and behaviors over time, enhancing its ability to predict and respond to different driving scenarios more effectively.

5.3.5 Advanced Lane Recognition Algorithms:

Implement more sophisticated lane recognition algorithms, such as Hough Transform or Deep Learning-based models, to enhance the system's accuracy and robustness in diverse road conditions.

REFERENCES

[1] B. Lutkevich, "techtarget," techtarget enterprise, january 2023. [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/driverless-car. [Accessed 25 May 2023].

[2] H. M. M. Armaghan, " Self driving car without machine learning," IRJET, Faisalabad, 2021.

[3] W. Cao, J. Ni, Y. Chen, Y. Chen, J. Zhu and D. Ali, "Theories and Applications for Self-Driving Cars Based on Deep Learning Methods," *Applied Sciences*, 2020.

[4] J. Deichmann, . E. Ebel and . K. Heineke, "Autonomous driving's future: Convenient and connected," mckinsy and company, 2023.

[5] "World Health Organization," WHO, 20 June 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries. [Accessed 1 June 2023].

[6] S. Anjana, D. Arya, S. Aiman and Y. A, "Autonomous Car using Raspberry PI and Ml," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 9, no. 2, p. 5, 2020.

[7] S. A. Rodríguez, V. Frémont and B. Wang, "Color-based Road Detection and its Evaluation on the KITTI Road," in *IEEE Intelligent Vehicles Symposium Proceedings*, 2014.

[8] M. A. Irshad, A. . N. Abbas and H. H. Ammar, "Experimental Analysis of Trajectory Control Using Computer Vision and Artificial Intelligence for Autonomous Vehicles," Giza, 2021.

[9] N. Bansode, K. Kunal, S. Yede and R. Rajvardhan, "Automatic Messaging System for Vehicle Tracking and Accident Detection," in *International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020.

[10] B. Nutwall, "Opensource.com," Opensource.com. , January 2021. [Online]. Available: https://opensource.com/resources/raspberry-pi. [Accessed 10 June 2023].

[11] Arduino, "Arduino Documentation," 2023. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3. [Accessed 10 June 2023].

[12] Components101, "Components101," 13 April 2021. [Online]. Available: https://components101.com/modules/l293n-motor-driver-module. [Accessed 10 June 2023].

[13] T. Malik, "Microcontrollers Lab," May 2020. [Online]. Available: ttps://microcontrollerslab.com/sim900a-gsm-module-pinout-examples-applications-datasheet/. [Accessed 10 June 2023].

[14] M. Chowhan, "Robocraze," 2023. [Online]. Available: https://robocraze.com/products/neo-6m-gps-module. [Accessed 10 June 2023].

[15] Cytron Technologies Singapore, "Cytron," 2023. [Online]. Available: https://sg.cytron.io/p-gy-521-mpu6050-6dof-accelerometer-plus-gyro. [Accessed 20 June 2023].

[16] A. Mittle, "Haar Cascades, Explained," Analytic Vidhya, Delhi, 2020.

[17] Wikimedia Foundation, "Wikipedia," Mediawiki, 25 July 2023. [Online]. Available: https://en.wikipedia.org/wiki/Canny_edge_detector#. [Accessed 31 July 2023].

[18] Open CV, "Open CV," 25 July 2023. [Online]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html. [Accessed 25 July 2023].

[19] P. Zende, A. Shelar, V. Vangate and M. Mali, "VEHICLE ACCIDENT DETECTION AND ALERT SYSTEM," IJARIIE-ISSN, pune, 2023.

[20] K. Dinakaran, A. S. Sagayara, S. Kabilesh, T. Mani, A. Anandkumar and G. Chandrasekaran, "Advanced lane detection technique for structural highway based on computer vision algorithm," meterialstoday, Gokul, 2020.

[21] K. Muhammad, ullah, J. Lloret and J. Del Sel, "Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions," researchgate, Islamabad, 2020.

[22] C. K. Gomathy, "ACCIDENT DETECTION AND ALERT SYSTEM," Reasearchgate, maharastra, 2022.

[23] D. Zukowski, K. Darji, A. R. Qureshi, V. Bhosale and H. Dubey, "Autonomous car using ultrasonic sensor, Arduino controller," International Journal of Advance Research, Ideas and Innovations in Technology, Mumbai, 2021.

[24] G. Bonnie, "titlemax," titlemax, 2014. [Online]. Available: https://www.titlemax.com/resources/history-of-the-autonomous-car/. [Accessed 1 june 2023].

[25] M. Liu and D. Grana, "Accelerating geostatistical seismic inversion using TensorFlow: A," *Computers & Geosciences* , vol. 178, p. 9, 2018.

[26] B. Zhao, J. Feng, X. Wu and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, p. 25, 2017.

[27] R.P. ltd, "Raspberry Pi," raspberry pi, 2023. [Online]. Available: https://www.raspberrypi.com/products/camera-module-v2/. [Accessed 10 june 2023].