IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

SELF DRIVING CAR USING RASPBERRY PI

[1]Shashank Agrawal, [2]Vineet Kumar, [3]Amar Kumar Dey

[1,2]Student, [3]Associate Professor

[1,2.,3]Department of Electronics And Telecommunication Engineering,

[1,2,3]Bhilai Institute of Technology, Durg, India

Abstract: Self-driving technology is becoming increasingly common and could revolutionize our transportation system in the near future. The success of autonomous vehicles largely depends on the development of reliable real-time detection systems for traffic control devices. This paper proposes a self-driving car prototype that utilizes various technologies such as image processing and machine learning, including algorithms for road lane detection, stop sign recognition, and traffic light recognition, controlled by a Raspberry Pi controller, with an Arduino UNO for additional processing, and an H-bridge to drive four DC motors for vehicle automation.

I. INTRODUCTION

Road accidents are a major problem worldwide, resulting in approximately 1.25 million deaths annually and an average of 3287 deaths per day. Additionally, between 20-50 million people suffer injuries and disabilities for life each year, according to the World Health Organization (WHO). It is found that 78% of road accidents are due to driver's fault or human error in driving and it is interesting to address this human issue with deployment of technology in the form of "Driverless Car".

The idea of self-driving cars has been talked about since the 1920s, but it wasn't until 1977 that Japan developed the first semi-automated car. A significant milestone in self-driving cars was achieved in the 1980s when a car with a speed limit of 31 kmph was created. Fast forward to 2013, Tesla began their autonomous car project and in 2014, they introduced the Tesla S with a semi-autopilot mode. Subsequent advancements in autopilot software were made in later models such as the Tesla X.

The researchers have developed a new AI model called SSD-MobileNet using CNN technology to identify and monitor real-life objects. The prototype has shown impressive performance in detecting and tracking trained objects, boasting a 99% accuracy rate with a confidence level of 98.2% [1]. This advancement aids in preventing accidents for robots. Additionally, a mathematical framework has been introduced to investigate the behavior of a two-wheeled self-balancing robot on flat and sloped surfaces [2]. A new system for detecting lanes using Raspberry Pi and Arduino technology has been developed, consisting of two main components: object detection and image processing. The camera on the prototype captures images, which are then analyzed to identify lane markings. The system also includes an obstacle detection feature that uses data from the images to generate commands. While this system has some limitations in accurately detecting lanes [3], it is able to recognize and respond to traffic signs and lights in real-time [4].

The paper is structured as follows: In Section II, the design of the self-driving car is explained using its circuit diagram. Section III discusses the subsystems of the self-driving car and their functions. Section IV covers the results and discussion, while Section V offers conclusions and suggestions for future work

II. ARCHITECTURE OF SELF-DRIVING CAR

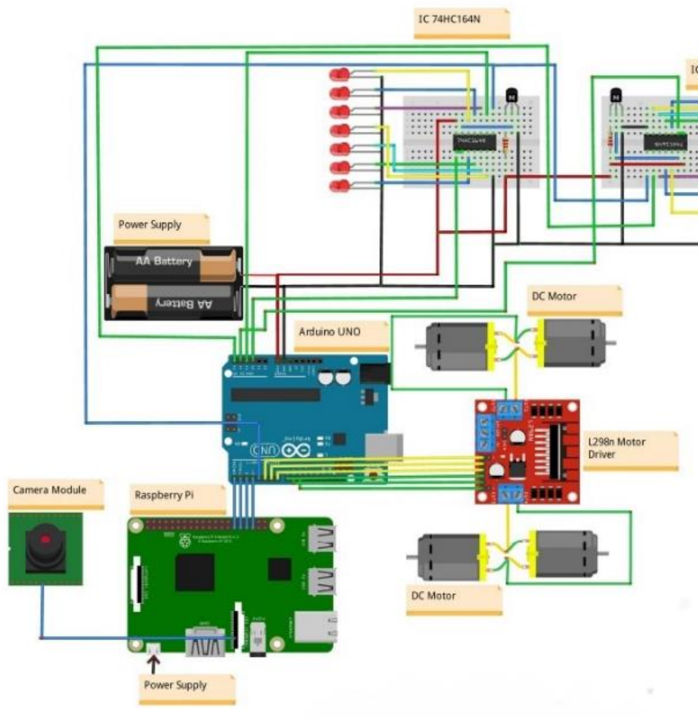


Fig. 1: Circuit Diagram of Self Driving Car

The circuit diagram of the self-driving car is shown in the Fig. 1. The components used in this project are enlisted below:

1. Raspberry Pi: The Raspberry Pi serves as the central control unit for the project, processing sensor data and running algorithms for autonomous navigation. It includes a CPU for computing, RAM for temporary storage, and an SD card for software and resources. GPIO pins allow communication with peripherals such as sensors and motors, while the camera module captures real-time images for lane detection and traffic analysis. USB ports provide connectivity with external devices, and networking capabilities allow communication with other systems. The Raspberry Pi, with its operating system managing resources, serves as a flexible platform for developing the self-driving car system.

2. Arduino UNO: The Arduino Uno is the microcontroller for the project, receiving commands from the Raspberry Pi to manage the motors of the vehicle. It has digital and analog I/O pins for connecting to sensors and motor drivers, PWM pins for controlling motor speed, and UART communication for receiving navigation instructions. The Arduino Uno maintains stable operation with voltage regulation and has a reset button for troubleshooting. It effectively controls the vehicle's movements based on input from the Raspberry Pi for autonomous navigation.

3. L298N Motor Driver: The L298N motor driver is a key part of the self-driving car project, allowing for two-way control of DC motors. Its dual H-bridge setup, ability to handle high currents, PWM speed control, and diode protection make it perfect for efficient and dependable motor control during autonomous navigation. It works seamlessly with the Arduino Uno, allowing for precise control of the vehicle's movement to ensure smooth operation. In order to control all four motors at the same time using the L298N motor driver, a paired connection setup is used. This involves grouping the motors into pairs and connecting each pair to a single L298N motor driver module. By using a cross-connection arrangement, where one motor from each pair is connected to each output of the motor driver, all four motors can be controlled simultaneously. This setup ensures that the motors operate in sync, allowing the vehicle to move in a coordinated manner during autonomous navigation.

4. IC74HC164N SIPO: The IC74HC164N, also known as a Serial-In-Parallel-Out (SIPO) integrated circuit, plays a key role in converting serial input data into parallel output signals. In our self-driving car project, this IC is crucial for controlling the vehicle's lighting system. When the car needs to stop, signals from the Arduino Uno are processed in series by the IC74HC164N. This results in all LEDs turning on at once, giving the appearance of a brake light. Likewise, during turns, the IC lights up LEDs on the correct side one by one, serving as a sequential turn signal. This setup ensures that the vehicle's lighting system works together smoothly, improving safety and efficiency when the vehicle is navigating

III. SUB-SYSTEMS OF SELF-DRIVING CAR

3.1 Lane Detection System (LDS):

To achieve the main objective of lane detection, a self-driving car should be able to detect the tracks or lanes in the road. The Lane Detection System consists of a camera module mounted on top of the self-driving car which is connected to the Raspberry Pi controller to detect the position of the car relative to the white lines or the tracks on each end of the road. The process flow of Lane detection system is shown in Fig. 2.

In the proposed self-driving car, the lanes at each end of the road are designed which the self-driving car will follow by remaining at the centre of the road. When the camera is off, no commands will be sent to the Arduino which is acting as a slave for controlling the motors, the Arduino module has been programmed in such a way that according to the information or the data which the Raspberry Pi controller or the master of the system provides it will drive the motor accordingly. When the user starts the camera, the system starts capturing the images and creates a region of interest (ROI) in the captured images for the detection of the lanes, after that all the images captured are converted from RGB to GRAY scale image and then the pixels of the image are divided (for example :- a 100 pixels image is divided into 100 individual pixel i.e. it will have 1 image for every individual) such that we can detect the white lanes in the road as the pixels for the white lanes will be high and the pixels for rest of the image will be low, these whole image processing is done internally in the Raspberry Pi controller, now from these detected lanes the system creates an imaginary line which remains at the centre of the road (Lane Centre) and will be further used to keep the car at the centre of the lane.

After the lane detection process is completed using image processing the next step comes to keep the car at the centre of the lane, which is done by creating an imaginary line which will be interpreted by the system as the car, the system will compare this line with the lane centre to ensure the car remains in the centre of the road, prompting the car to move forward. Whenever the car deviates from the centre, either to the right or left, the Raspberry Pi controller sends the data accordingly to the Arduino. The Arduino then controls the motor power based on this data to keep the car centered. As a result, while moving, the car is always in a self-correcting mode to maintain its position in the centre of the road.

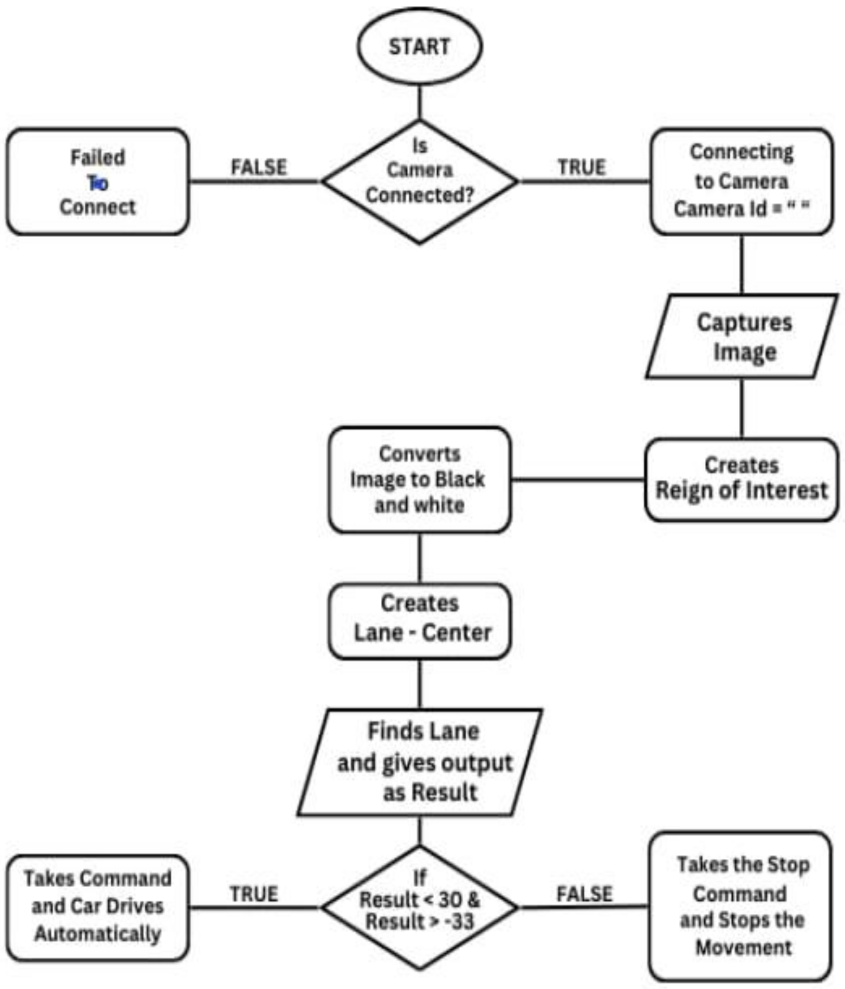
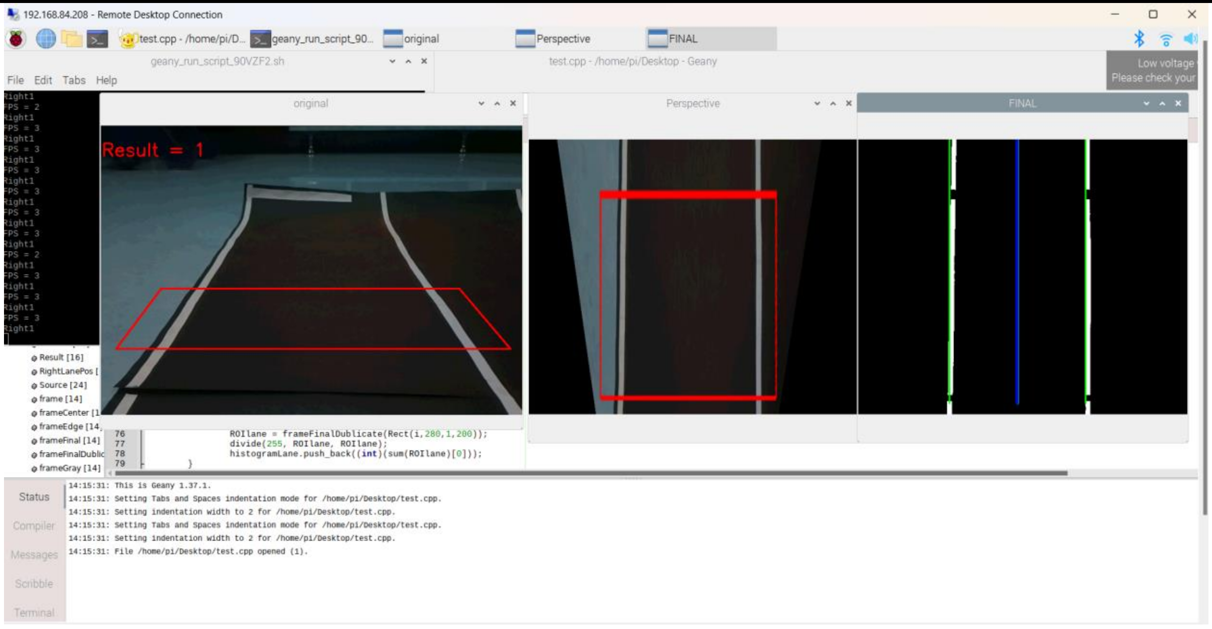

START
Is Camera Connected?
FALSE
Failed To Connect
TRUE
Connecting to Camera Camera Id = " "
Captures Image
Creates Reign of Interest
Converts Image to Black and white
Creates Lane - Center
Finds Lane and gives output as Result
If Result < 30 & Result > -33
TRUE
Takes Command and Car Drives Automatically
FALSE
Takes the Stop Command and Stops the Movement

Fig. 2: Process Flow of Lane Detection System



```
Right1
FPS = 2
Right1
FPS = 3
Right1
FPS = 3
Right1
FPS = 3
Right1
FPS = 3
Right1
FPS = 3
Right1
FPS = 3
Right1
FPS = 2
Right1
FPS = 3
Right1
FPS = 3
Right1
FPS = 3
Right1
```

```
            ROIlane = frameFinalDublicate(Rect(i,280,1,200));
            divide(255, ROIlane, ROIlane);
            histogramLane.push_back((int)(sum(ROIlane)[0]));
        }
```

```
14:15:31: This is Geany 1.37.1.
14:15:31: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
14:15:31: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
14:15:31: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
14:15:31: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
14:15:31: File /home/pi/Desktop/test.cpp opened (1).
```

Fig. 3: Implementation of LDS in Raspberry Pi controller

3.2 Traffic-Light and Stop Sign Detection System (TLSSDS):

A traffic-light and stop sign detection system is a vital technology in development of autonomous driving car and systems like advanced driver assistance systems (ADAS). In the proposed self-driving car, we will be using machine learning (ML) technology to accurately identify stop signs in various driving environments. The process flow of TLSSDS with LDS is shown in Fig. 4.
A traffic-light and stop sign detection system using ML works by processing and analyzing data of the images captured by the camera mounted on the car, the system utilizes a XML file, which stores trained data from positive and negative samples of stop sign and traffic light.

Positive samples: These are the instances that the model learn to identify as the target or the instance which is to be detected. For example, in a model designed to identify spam emails, the spam emails are the positive samples.

Negative samples: These are the instances that the model learn to identify as not belonging to the target or the instance which are not to be detected. For example, in a model designed to identify spam emails, the legit emails are the negative samples.

The detection of stop sign and traffic light takes place in several steps: -

1. Data Collection: A wide range of images featuring stop signs and traffic light which is to be detected (red light) are captured from different angles, lighting conditions, and backgrounds to ensure that system is exposed to different real-time scenarios, these images will be considered as the positive samples. Similarly, the negative samples are also captured in different angles, lighting conditions. In our proposed work the negative samples are the designed road which the car is intended to follow by lane detection.

2. Preprocessing: Once collected, the images are pre-processed. This involve resizing, enhancing contrast, cropping and to convert the image from RGB to GRAY-SCALE image to highlight the stop signs and the traffic lights. This step makes the data more uniform and easier for the system to analyse and utilize for training.

3. Model Training: The processed images are then used to train a detection model. The training helps the model to learn what stop signs and traffic light looks like from different angles and in different lighting condition. At last, the trained model is saved into a file in XML format, which will be further used in our project to help the car detect and respond to the models in real time, as it drives.

4. Model Testing: To ensure that the trained data is reliable, the trained model undergoes a test with new images it hasn't seen before during the training. This step is crucial as it helps to ensure that the model can accurately identify the models in different real-world conditions.
5. Deployment: The trained model is then incorporated within our system and integrated to work simultaneously with the lane detection system. While the system is busy detecting the lanes by processing the live feed from the camera module, it also ensures that the stop sign and traffic light is detected and provide necessary data to be used further in the design.
This integration n of the stop sign and traffic light detection and the lane detection ensures that our system is capable of handling multiple tasks at once, like staying in the centre of the lane while also detecting the required models.

6. Real-Time Detection: As the car drives, performing multiple tasks of lane detection, stop sign detection and traffic light detection at once, it also performs another task which is the most important part of this whole system, i.e. calculating the distance between the car and the models which is to be detected, the system uses the linear equation given below to calculate the distance:

$$\mathbf{y = mx + c}$$

- x represents the change in x-axis, calculated as **P2.x – P1.x**, where P2 indicates the car's position and P1 indicates the position of the model to be detected.
- m represents the rate at which y changes with respect to x.
- y is distance to be calculated between car and the model.

To calculate the distance between them, we first manually calculate the distance by placing the models at two different distances (15 cm and 30 cm) and recording the corresponding value of x in pixels format which is displayed at the user's display, by using these measurements, we can determine the coefficients m and c in the equation.

Once the system can calculate the distance between the car and the model, any model which is detected within a 45 cm range gives a response. Based on this response, the Raspberry Pi controller then provides the information to the Arduino, which then execute a programmed task based on the data received from the Raspberry Pi controller. If a stop sign is detected, the car stops for few seconds and resume the normal operation (Fig. 5). Similarly, if a red light is detected at a traffic light, the car stops there and wait until the signal goes green before proceeding (Fig. 6 & Fig. 7). This multi-tasking capability of our self-driving car ensures that the car responds appropriately to the traffic rules, while enhancing safety.

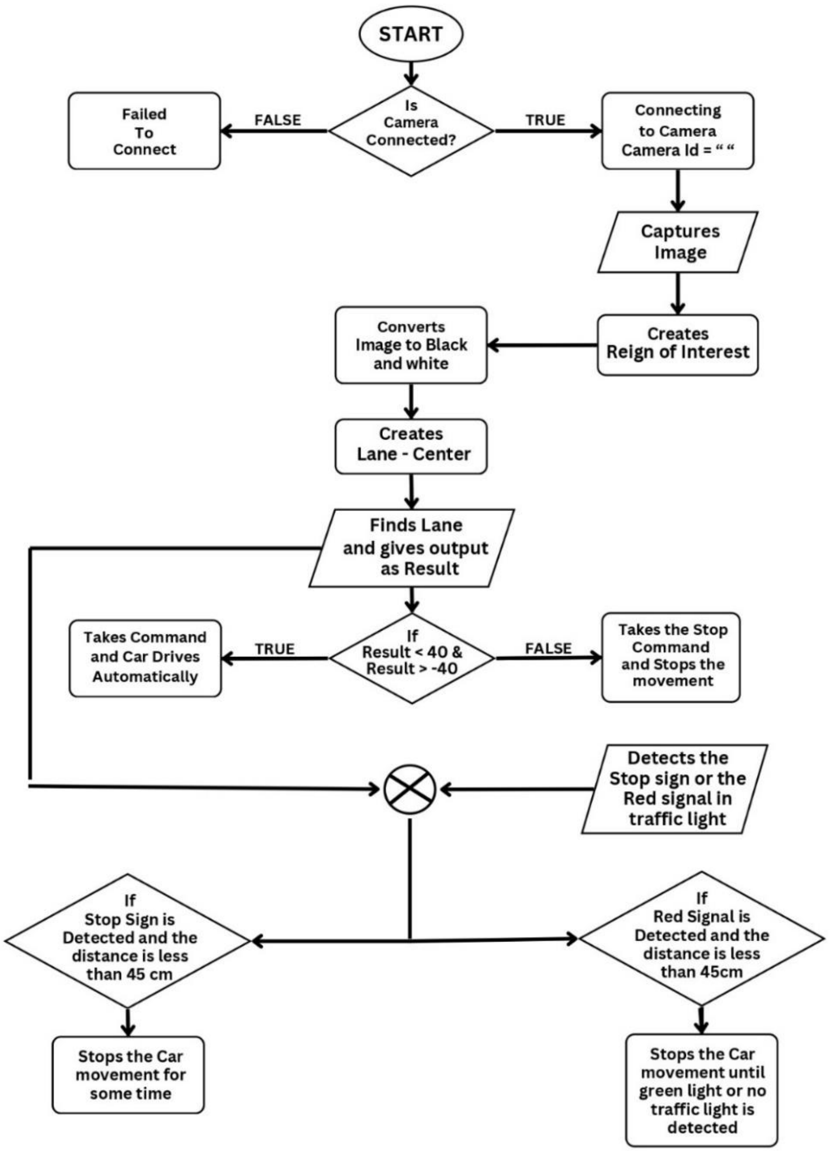
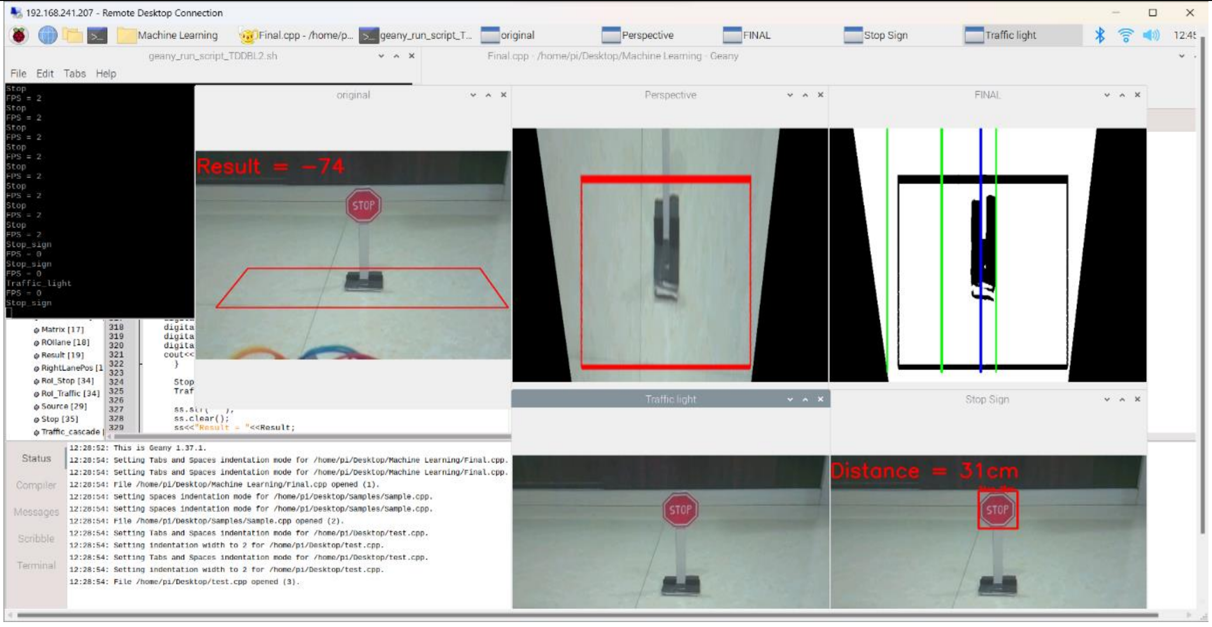

START
Is Camera Connected?
FALSE
Failed To Connect
TRUE
Connecting to Camera Camera Id = " "
Captures Image
Creates Reign of Interest
Converts Image to Black and white
Creates Lane - Center
Finds Lane and gives output as Result
If Result < 40 & Result > -40
TRUE
Takes Command and Car Drives Automatically
FALSE
Takes the Stop Command and Stops the movement
Detects the Stop sign or the Red signal in traffic light
If Stop Sign is Detected and the distance is less than 45 cm
Stops the Car movement for some time
If Red Signal is Detected and the distance is less than 45cm
Stops the Car movement until green light or no traffic light is detected

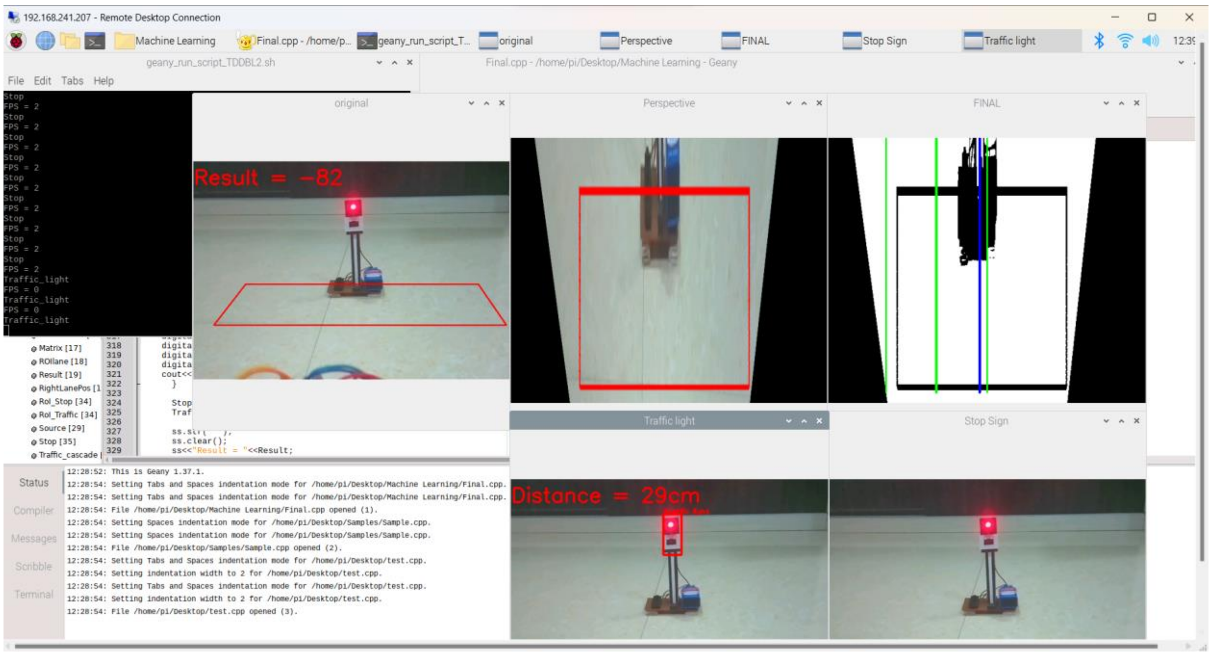
Fig. 4: Process flow of Stop Sign and Traffic Light Detection System with Lane Detection System



```
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop_sign
FPS = 0
Stop_sign
FPS = 0
Traffic_light
FPS = 0
Stop_sign
```

```
12:28:52: This is Geany 1.37.1.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/Machine Learning/Final.cpp.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/Machine Learning/Final.cpp.
12:28:54: File /home/pi/Desktop/Machine Learning/Final.cpp opened (1).
12:28:54: Setting Spaces indentation mode for /home/pi/Desktop/Samples/Sample.cpp.
12:28:54: Setting Spaces indentation mode for /home/pi/Desktop/Samples/Sample.cpp.
12:28:54: File /home/pi/Desktop/Samples/Sample.cpp opened (2).
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
12:28:54: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
12:28:54: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
12:28:54: File /home/pi/Desktop/test.cpp opened (3).
```

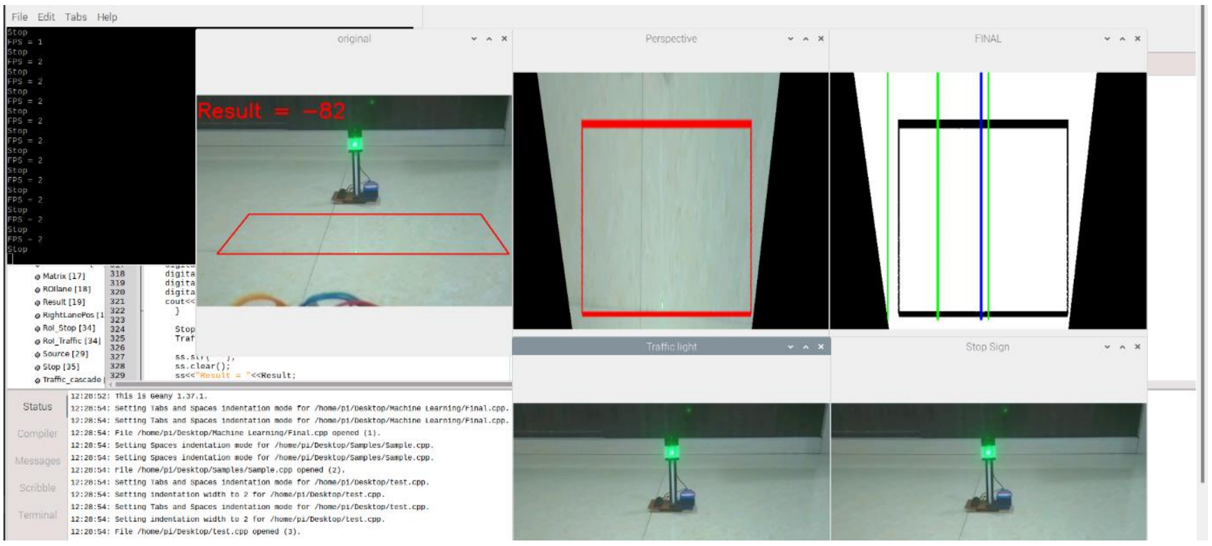
Fig. 5: Stop Sign Detection with Lane Detection System in Raspberry Pi controller



```
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Traffic_light
FPS = 0
Traffic_light
FPS = 0
Traffic_light
```

```
12:28:52: This is Geany 1.37.1.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/Machine Learning/Final.cpp.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/Machine Learning/Final.cpp.
12:28:54: File /home/pi/Desktop/Machine Learning/Final.cpp opened (1).
12:28:54: Setting Spaces indentation mode for /home/pi/Desktop/Samples/Sample.cpp.
12:28:54: Setting Spaces indentation mode for /home/pi/Desktop/Samples/Sample.cpp.
12:28:54: File /home/pi/Desktop/Samples/Sample.cpp opened (2).
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
12:28:54: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
12:28:54: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
12:28:54: File /home/pi/Desktop/test.cpp opened (3).
```

Fig. 6: Traffic Light Detection (Red Light) with Lane Detection System in Raspberry Pi controller



```
Stop
FPS = 1
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
FPS = 2
Stop
```

```
12:28:52: This is Geany 1.37.1.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/Machine Learning/Final.cpp.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/Machine Learning/Final.cpp.
12:28:54: File /home/pi/Desktop/Machine Learning/Final.cpp opened (1).
12:28:54: Setting Spaces indentation mode for /home/pi/Desktop/Samples/Sample.cpp.
12:28:54: Setting Spaces indentation mode for /home/pi/Desktop/Samples/Sample.cpp.
12:28:54: File /home/pi/Desktop/Samples/Sample.cpp opened (2).
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
12:28:54: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
12:28:54: Setting Tabs and Spaces indentation mode for /home/pi/Desktop/test.cpp.
12:28:54: Setting indentation width to 2 for /home/pi/Desktop/test.cpp.
12:28:54: File /home/pi/Desktop/test.cpp opened (3).
```

Fig. 7: No detection of Green Light and the normal operation continues

IV. RESULT & DISCUSSION

A prototype of self-driving car is developed in this project and is shown in Fig. 8. Our self-driving car uses advanced image processing techniques to accurately detect the lanes and ensures that it can drive safely under various conditions, the technique is highly effective for lane detection but in rainy or foggy weather the efficiency of the system reduces. In these cases, incorporating new sensors like RADAR or LIDAR could enhance the detection accuracy of the system. This system also features detection of stop sign and traffic light detection, the system is trained with wide range of images from different angles and different conditions. Thus, the system provides a high level of accuracy in detecting stop sign and traffic lights, ensuring effectiveness even when these signs are partially visible or at a distance. Integrating these techniques in our system enables the car to make real-time decisions essential for safe driving.

It is demonstrated that the system accurately follows the lane and while moving forward, ensures the detection of the stop sign and traffic-light detection, the response of the system to detection is as follows: for stop sign, the car stops it's movement for few seconds and after that resumes the normal operation, for red signal in the traffic light, the car stops its movement until the light is red, when the red signal turns green the car then starts moving displaying a very good accuracy and performance.

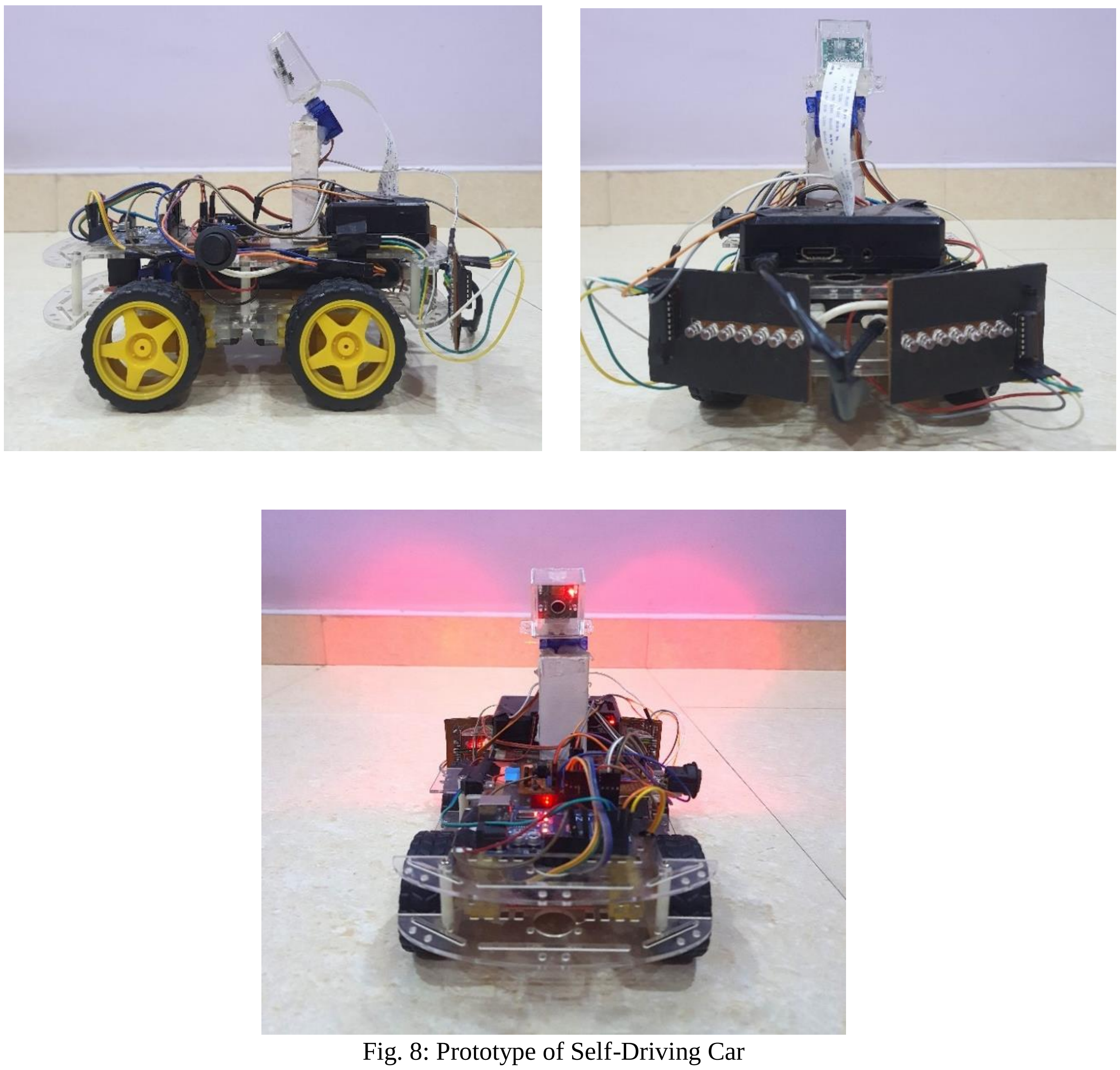


Fig. 8: Prototype of Self-Driving Car

V. CONCLUSIONS & FUTURE SCOPE

5.1 CONCLUSIONS

A low-cost prototype of a Self-Driving Car model is designed developed and all functionalities are successfully demonstrated. The car is able to follow lane efficiently using the programmed algorithms and the traffic colours are detected and decisions are made by the car using image processing techniques to follow real-time traffic rules. Car is able to respond to the given instructions precisely and is able to detect stop signs and respond accordingly.

5.2 FUTURE SCOPE

- The lane detection algorithms can be further optimized to increase the efficiency of the system.
- Features such as GPS can be added so that the live location of the car can be accessed in real time for safety purposes.
- Accident alert system can be added to the car so that it contacts the emergency services and sends the last location if the car meets with an accident.

REFERENCES

[1] G. Chandan, A. Jain, H. Jain and Mohana, "Real Time Object Detection and Tracking Using Deep Learning and OpenCV," *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2018, pp. 1305-1308.

[2] Pannaga R.M. and B.P. Harish, “Modelling and implementation of twowheel self- balance robot,” International Journal of Electrical, Electronics and Computer Science Engineering, Vol. 4, Issue 6, pp. 33- 40, December 2017.

[3] Tiple, A. Hemant, T. Hemant, Gurav, and S. Hanumant, “Prototype of Autonomous Car Using Raspberry Pi,” *International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)*, vol. 5, 2018, pp. 39-43.

[4] T. Moura, A. Valente, A. Sousa and V. Filipe, "Traffic Sign Recognition for Autonomous Driving Robot," *2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Espinho, Portugal, 2014, pp. 303-308.