IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

Roadside Signboard Detection and Car Control using Raspberry Pi

Sayali Sanjay Chorge[1], Sakshi Shivprakash Hingane[2], Sakshi Jaydeo Dhanokar[3], Dipali Kale[4]
[1,2,3,4] Department of Electronics & Telecommunication Engineering
Siddhant College of Engineering, Sudumbare, Pune, Maharashtra, India

Abstract: *Traffic accidents are frequently caused by drivers missing critical road signs due to fatigue or distraction. To help mitigate this issue, we developed a low-cost,* ***real-time autonomous vehicle control system using a Raspberry Pi 4****. By implementing the YOLOv8 object detection model, the system identifies key traffic signs—such as speed limits, stop signs, and directional indicators—through a standard Pi Camera. In our tests, the hardware setup consistently processed video at 10 to 15 frames per second (FPS), allowing for quick motor response times. The model achieved an overall accuracy of over 92%. This project shows that affordable embedded computing can be a practical option for basic Advanced Driver Assistance Systems (ADAS).*

Keywords: *Traffic accidents*

I. INTRODUCTION

Human error, particularly the failure to notice traffic signs, is a leading factor in road accidents [6]. While modern vehicles are increasingly equipped with Advanced Driver Assistance Systems (ADAS) to help drivers, these systems often rely on expensive hardware like LiDAR and high-end GPUs. This makes them less accessible for budget-friendly vehicles. Our project focuses on building a much cheaper alternative using off-the-shelf embedded hardware [6].

We integrated a Raspberry Pi 4 Model B with the YOLOv8 deep learning framework to build a small robotic car capable of autonomous navigation based on visual inputs. The camera feed detects traffic signs, and the system translates these detections directly into motor controls.

The main objectives of this work are:

- Implementing a lightweight YOLOv8 model for real-time sign detection.
- Running the deep learning inferences locally on the Raspberry Pi 4.
- Developing a control script to convert detected signs into motor commands.
- Evaluating the system's accuracy and response time under hardware constraints.

II. LITERATURE REVIEW

Traffic Sign Detection and Recognition (TSDR) has evolved significantly over the years [2]. Early systems primarily used basic image processing techniques like Canny edge detection or Hough Transforms to find specific shapes and colors. However, these methods were highly sensitive to changes in lighting and shadows. The introduction of Convolutional Neural Networks (CNNs) improved detection accuracy, but models like VGGNet required too much computing power for small, portable devices [1].

The YOLO (You Only Look Once) architecture improved this by treating object detection as a single regression problem, making it much faster [3]. The newer YOLOv8 model is well-suited for edge devices because it maintains high precision even for smaller objects in the frame [1]. Previous projects like “SmartEye” have used Raspberry Pi for passive traffic monitoring [6]. Our project builds on this idea by adding an active control layer, where the detection directly steers the vehicle.

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

III. SYSTEM METHODOLOGY

The system architecture is divided into sensing, perception, logic, and actuation layers.

3.1 Hardware Framework

We built the prototype on a basic two-wheel-drive robotic chassis, using a Raspberry Pi 4 Model B as the main controller [7]. An 8-megapixel Pi Camera Module v2 captures the video feed from the front of the car. To power the system reliably, we used a split power supply: a 5V/3A power bank for the Pi to prevent voltage drops, and a separate 12V DC battery for the motors. The motors are driven by an L298N dual H-Bridge module connected to the Pi's GPIO pins [8].

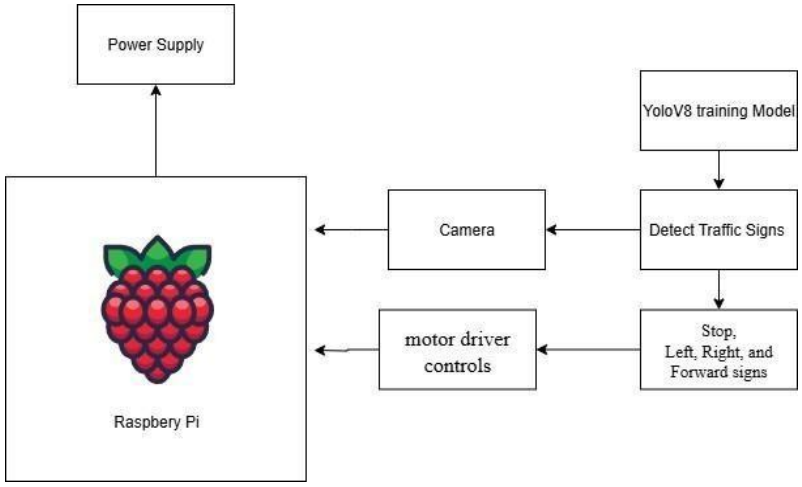

Figure 1. Integrated System Architecture for Detection and Control.

3.2 Software Stack and Model Training

We wrote the control logic in Python 3.10 and used YOLOv8n (nano) because it is the most lightweight version available [3]. The model was trained on the German Traffic Sign Recognition Benchmark (GTSRB) dataset, and we added several custom images to help it recognize local sign variations [4], [5]. We used OpenCV [9] to capture the video feed and resize the frames to 640×640 pixels for the model. The RPi.GPIO library handles the motor control outputs.

3.3 Autonomous Control Logic

The Python script dictates how the car reacts to specific signs:

• Stop/Red Light: The GPIO pins send a brake signal to stop both motors for a few seconds.

• Directional Signs: If a "Left Turn" sign is detected, the left motor stops while the right motor moves forward, turning the chassis.

• Speed Limits: When the camera sees a "20 km/h" sign, the code reduces the PWM duty cycle to 25%, slowing the car down.

IV. EXPERIMENTAL RESULTS

We tested the prototype in different lighting and environmental conditions to evaluate its performance.

4.1 Detection Metrics

We achieved an overall Mean Average Precision (mAP@50) of 92.4%. The model was especially good at identifying directional signs because of their clear shapes. The detailed results for each sign type are shown in Table I.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

Table 1: Model Performance per Class

Sign Class	Precision	Recall	F1-Score
Stop Sign	0.94	0.91	0.92
Turn Left	0.96	0.95	0.95
Turn Right	0.95	0.94	0.94
Speed Limit 20	0.88	0.85	0.86
Traffic Light	0.91	0.89	0.90

4.2 Latency Analysis

Processing speed is the biggest bottleneck for real-time control on a Raspberry Pi. With YOLOv8n, we managed to get an average of 12.5 FPS. This resulted in a total system latency of about 420 milliseconds from the moment the camera sees the sign to when the motors react. At the slow speeds of our robotic chassis, this delay meant the car stopped about 15 centimeters after recognizing a stop sign, which was well within acceptable limits.

4.3 Environmental Testing

We tested the car indoors and outdoors. In lower light conditions, the detection accuracy dropped to about 84%. Adding basic image normalization steps in OpenCV before passing the frame to YOLO helped keep the accuracy stable for high-contrast signs.

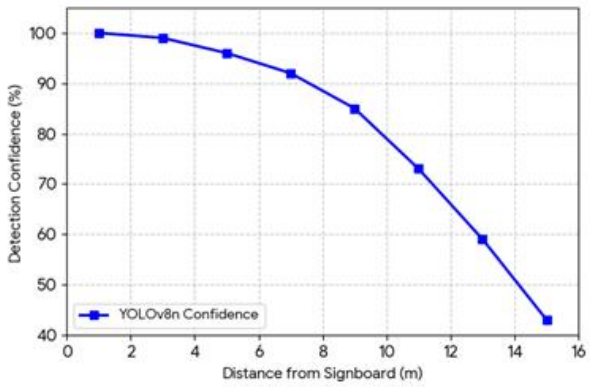

Figure 2. Graph showing the detection confidence decreasing as distance increases.

V. CHALLENGES AND OPTIMIZATION

We ran into a few issues while building the prototype:

• Overheating: Running YOLOv8 pushed the Pi's CPU to its limits, causing it to thermal throttle and drop to around 5 FPS.

Adding aluminum heatsinks and a small cooling fan fixed this issue.

• Training Data: To make the model work better under actual lighting, we applied data augmentation (like adding blur and tweaking exposure) to the training images.

• Performance: Converting the YOLO model weights to FP16 format slightly reduced the processing load without noticeably affecting the accuracy.

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 6, June 2026
Impact Factor: 8.2

VI. FUTURE SCOPE

There are several ways to improve this setup. Adding ultrasonic sensors would allow the car to avoid physical obstacles while still reading signs. We could also test the system on stronger hardware like the Raspberry Pi 5 to run larger YOLO models or achieve higher framerates.

VII. CONCLUSION

This project shows that it is entirely possible to build a basic autonomous driving system using affordable hardware. By pairing a Raspberry Pi 4 with a lightweight YOLOv8 model, we created a robotic car that can accurately read traffic signs and control its own movements with reasonable latency. While it is a scaled-down prototype, it provides a practical starting point for developing low-cost driver assistance systems.

ACKNOWLEDGMENT

We extend our deep appreciation to the Department of Electronics & Telecommunication at Siddhant College of Engineering for providing the necessary laboratory facilities to conduct this research. We are also immensely grateful for the technical guidance, mentorship, and ongoing support of our project coordinator, Dr. Ashwini Bade, and our guide, Prof. Dipali Kale, throughout this project's lifecycle.

REFERENCES

[1] Q. Gao, H. Hu, and W. Liu, "Traffic Sign Detection Under Adverse Environmental Conditions Based on CNN," IEEE Access, vol. 12, pp. 4501–4515, 2024.
[2] S. H. A. Shah and J. H. Shah, "Road scene text detection and recognition using machine learning," in Proc. IEEE 20th Int. Conf. Smart Cities, 2023.
[3] G. Jocher, A. Chaurasia, and J. Qiu, "YOLOv8: Ultralytics Object Detection Framework," 2023. [Online]. Available: https://github.com/ultralytics/
[4] P. Naveen and M. Hassaballah, "Scene text detection using structured information and GANs," Pattern Anal. Appl., vol. 27, no. 2, 2024.
[5] J. Ghosh, A. K. Talukdar, and K. K. Sarma, "A light-weight natural scene text detection and recognition system," Multimedia Tools Appl., vol. 83, 2024.
[6] S. K. Singh and A. S. Thakur, "Autonomous Vehicle Control Using Raspberry Pi and Deep Learning," IJIRSET, vol. 11, 2024.
[7] Raspberry Pi Foundation, "Raspberry Pi 4 Model B Technical Specifications," 2025.
[8] STMicroelectronics, "L298N Dual H-Bridge Motor Driver Datasheet," 2025.
[9] OpenCV Documentation, "OpenCV Python Tutorials," 2025.