IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# Design And Development Of Advanced Driver Assistant System Using Raspberry Pi

[1]Amit Ajit Patil, [2] Sujay Dhananjay Kulkarni, [3] Rushikesh Somnath Mirajkar, [4]Aditya Madhuri Patil,

[5]Sudesh Rajaram Patil

Department of Mechanical Engineering, ADCET, Ashta-416302, India

**Abstract**: In this project, we present a comprehensive exploration into the design and development of an Advanced Driver Assistant System (ADAS) leveraging the capabilities of Raspberry Pi. The system is meticulously crafted to address critical aspects of driver safety and assistance through an array of advanced functionalities. Our approach integrates a Raspberry Pi as the central processing unit, enabling efficient and scalable computation for real-time image processing and analysis. A combination of cameras and sensors forms the sensory apparatus, capturing and interpreting the surrounding environment. Through sophisticated algorithms, the system offers features such as lane departure warnings, real-time collision detection, and adaptive cruise control.

The core objective is to augment driver safety by providing timely and accurate alerts, thereby reducing the risk of accidents. The implementation on the Raspberry Pi platform ensures a compact, cost-effective, and easily deployable solution. We delve into the intricacies of the image processing pipeline, sensor fusion techniques, and algorithmic intricacies that empower the ADAS to make informed decisions. This project not only contributes to the advancement of ADAS technology but also serves as a foundation for future developments in intelligent transportation systems. By embracing the versatility of Raspberry Pi, we present a robust and accessible solution poised to enhance overall road safety and redefine the driving experience.

## I. INTRODUCTION

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation. They generally consist of an ARM-based CPU, RAM, USB ports, GPIO (General Purpose Input/Output) pins, HDMI output, and networking capabilities. Raspberry Pi can be used.Some well-known models include the Raspberry Pi 3, Raspberry Pi 4, and the Raspberry Pi Zero. Computer vision is an interdisciplinary field of artificial intelligence and computer science that converts input from an image or video into a precise representation. It functions the same way as human eyesight does- verifying computers' ability to see, recognize, and analyze pictures. In computer vision and image processing, a feature is a piece of information about the content of an image.

ADAS stands for "Advanced Driver Assistance Systems." It refers to a set of technologies and features implemented in vehicles to enhance safety, improve driving experience, and assist drivers in various ways. ADAS technologies use sensors, cameras, radar and other advanced hardware to provide real-time information and automate certain tasks, ultimately reducing the risk of accidents and improving overall road safety. Key features and components of ADAS includes like Collision Avoidance Systems, Lane Departure Warning and Lane Keeping Assist, Adaptive Cruise Control, Automatic Emergency Braking, Blind Spot Monitoring, Traffic Sign Recognition, Traffic Sign Recognition etc.

## II. EXPERIMENTAL / ANALYTICAL / SIMULATION

Research: Understand existing ADAS technologies and identify features to implement. Hardware Selection: Choose sensors (e.g., cameras, ultrasonic sensors) compatible with Raspberry Pi for data collection.
Software Development: Use Python to develop algorithms for tasks like lane detection, object recognition, and collision avoidance.
Integration: Connect sensors to Raspberry Pi and develop software to process data and make decisions.
Testing: Conduct extensive testing to ensure the system's reliability and safety.
Iterate: Continuously refine the system based on testing feedback and real-world simulations.
Deployment: Deploy the ADAS system in a controlled environment for further testing and evaluation.
Remember, safety should always be the top priority, so thorough testing and validation are crucial at every stage of development.

## III. RESULTS AND DISCUSSION

Research: Understand existing ADAS technologies and identify features to implement.
Hardware Selection: Choose sensors (e.g., cameras, ultrasonic sensors) compatible with Raspberry Pi for data collection.
Software Development: Use Python to develop algorithms for tasks like lane detection, object recognition, and collision avoidance.
Integration: Connect sensors to Raspberry Pi and develop software to process data and make decisions.
Testing: Conduct extensive testing to ensure the system's reliability and safety.
Iterate: Continuously refine the system based on testing feedback and real-world simulations.
Deployment: Deploy the ADAS system in a controlled environment for further testing and evaluation.

**Results**:
Performance Evaluation: Present quantitative results on the performance of each component of the ADAS system, such as accuracy of lane detection, object recognition, and response time for collision avoidance.
Simulation Data: Provide data collected during simulated driving scenarios, including sensor readings, detected objects, and system responses.
Comparison: Compare the performance of your ADAS system with existing solutions or benchmarks, highlighting any improvements or limitations.

**Discussion**:
Effectiveness: Discuss how well the ADAS system performed in various driving scenarios and its effectiveness in enhancing driver safety.
Challenges: Address any challenges encountered during development, such as sensor limitations, algorithm complexities, or computational constraints.
Future Improvements: Propose potential enhancements to the system, such as incorporating additional sensors, optimizing algorithms, or integrating machine learning techniques for better decision-making.
Real-world Application: Consider the feasibility of deploying the ADAS system in real-world vehicles and discuss potential implications for road safety and vehicle automation.

## IV. EXPERIMENTAL SETUP:

Raspberry Pi Configuration: Describe the Raspberry Pi model used and any additional hardware components, such as cameras, ultrasonic sensors, and LIDAR.

Sensor Placement: Explain the placement and orientation of sensors on the vehicle or simulated environment to capture relevant data.

Software Framework: Outline the software framework used for development, including the programming language (e.g., Python), libraries (e.g., OpenCV for computer vision), and any custom code developed for data processing and decision-making.

Simulated Scenarios: Detail the simulated driving scenarios created for testing the ADAS system, such as highway driving, lane

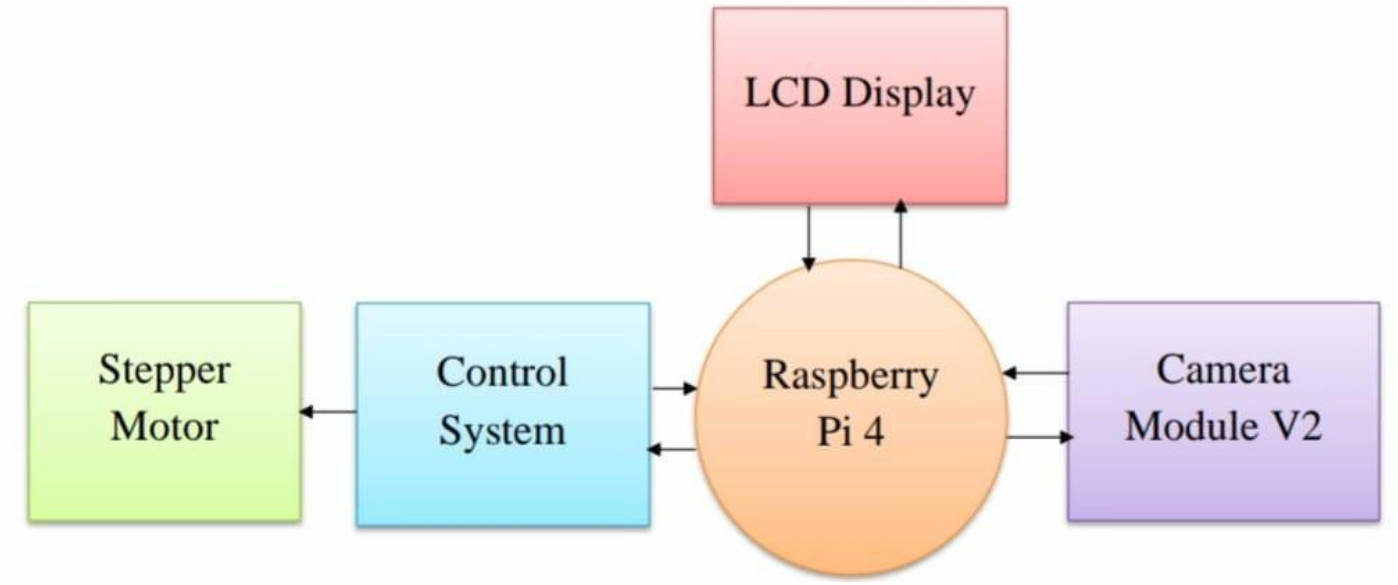
changes, pedestrian crossings, and obstacle avoidance.

Data Collection: Explain how data was collected during simulation runs, including sensor readings, image frames, and ground truth labels for evaluation.

Simulation Environment: Specify the software tools or simulators used to create the virtual environment for testing, such as CARLA, Unity, or custom-built simulations.

Validation Process: Briefly describe the validation process used to ensure the accuracy and reliability of the simulated results, such as comparing simulation outputs with real-world data or manual inspection of system behavior.

## V. Proposed experimental Setup




## VI. Efficiency and Productivity

Modular Design: Implement a modular design approach, breaking down the ADAS system into smaller, manageable components. This allows for easier debugging, testing, and future enhancements.

Code Reusability: Maximize code reusability by creating reusable functions, classes, and modules. This reduces development time and ensures consistency across different parts of the system.

Parallel Development: Assign tasks to multiple team members to work on different components simultaneously, accelerating the development process.

Version Control: Use version control systems like Git to track changes, collaborate with team members, and revert to previous versions if needed.

Automated Testing: Implement automated testing frameworks to quickly identify bugs and regressions, ensuring the reliability and stability of the system.

Continuous Integration/Continuous Deployment (CI/CD): Set up CI/CD pipelines to automate the build, testing, and deployment processes, streamlining the development workflow and reducing manual errors.

Documentation: Maintain comprehensive documentation for code, APIs, and system architecture to facilitate knowledge sharing, onboarding new team members, and troubleshooting.

Performance Optimization: Continuously optimize algorithms and code for better performance and resource utilization, considering factors like speed, memory usage, and power consumption..

## VII. Experimental Model

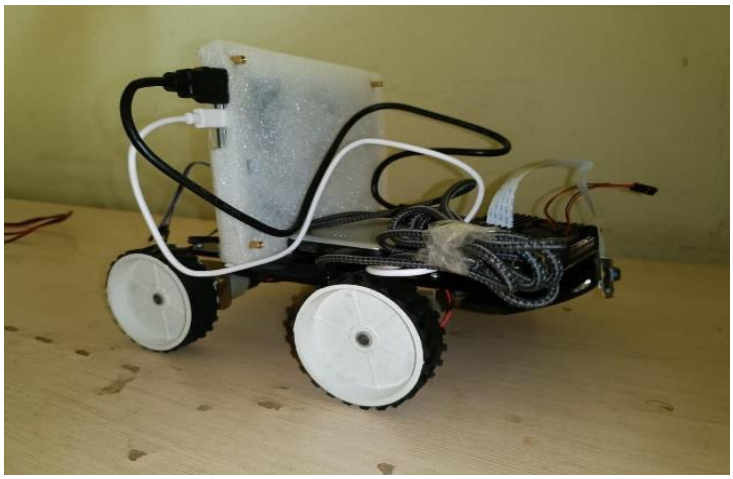


Fig.- Experimental model

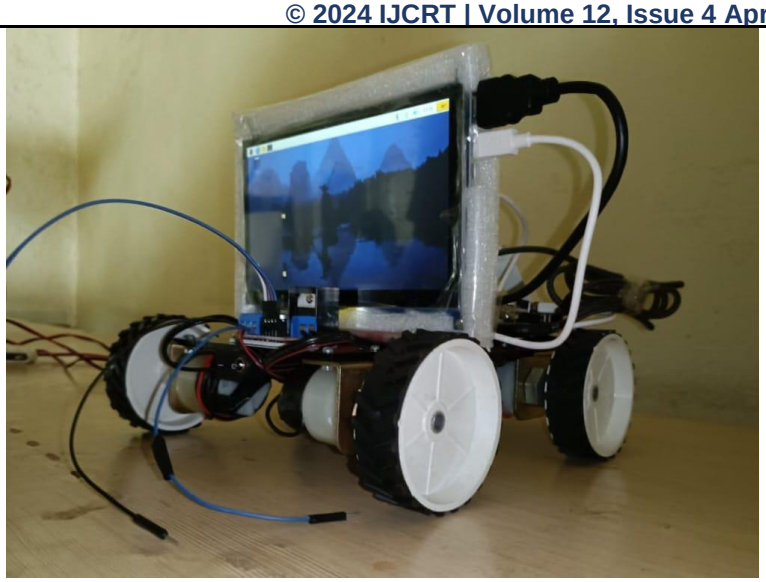


Fig.- General view of model

## VIII. **Lane Detection Program Code**

```
 import cv2
import numpy as np
import utlis
curveList = []
avgVal=10
def getLaneCurve(img,display=2):
imgCopy = img.copy()
imgResult = img.copy()
#### STEP 1
imgThres = utlis.thresholding(img)
#### STEP 2
hT, wT, c = img.shape
points = utlis.valTrackbars()
imgWarp = utlis.warpImg(imgThres,points,wT,hT)
   imgWarpPoints = utlis.drawPoints(imgCopy,points)
#### STEP 3
middlePoint,imgHist = utlis.getHistogram(imgWarp,display=True,minPer=0.5,region=4)
curveAveragePoint, imgHist = utlis.getHistogram(imgWarp, display=True, minPer=0.9)
curveRaw = curveAveragePoint - middlePoint
#### SETP 4
curveList.append(curveRaw)
if len(curveList)>avgVal:
curveList.pop(0)
curve = int(sum(curveList)/len(curveList))
#### STEP 5
if display != 0:
imgInvWarp = utlis.warpImg(imgWarp, points, wT, hT, inv=True)
imgInvWarp = cv2.cvtColor(imgInvWarp, cv2.COLOR_GRAY2BGR)
imgInvWarp[0:hT // 3, 0:wT] = 0, 0, 0
imgLaneColor = np.zeros_like(img)
imgLaneColor[:] = 0, 255, 0
imgLaneColor = cv2.bitwise_and(imgInvWarp, imgLaneColor)
imgResult = cv2.addWeighted(imgResult, 1, imgLaneColor, 1, 0)
midY = 450
cv2.putText(imgResult, str(curve), (wT // 2 - 80, 85), cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 255), 3)
   cv2.line(imgResult, (wT // 2, midY), (wT // 2 + (curve * 3), midY), (255, 0, 255),
5) cv2.line(imgResult, ((wT // 2 + (curve * 3)), midY - 25), (wT // 2 + (curve * 3), midY + 25), (0, 255, 0), 5)
for x in range(-30, 30):
w = wT // 20
cv2.line(imgResult, (w * x + int(curve // 50), midY - 10),
(w * x + int(curve // 50), midY + 10), (0, 0, 255), 2)
```

```
#fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer);
#cv2.putText(imgResult, 'FPS ' + str(int(fps)), (20, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (230, 50, 50), 3);
if display == 2:
imgStacked = utlis.stackImages(0.7, ([img, imgWarpPoints, imgWarp],
[imgHist, imgLaneColor, imgResult]))
cv2.imshow('ImageStack', imgStacked)
elif display == 1:
   cv2.imshow('Resutlt', imgResult)
```

## IX. Summary :

An Advanced Driver Assistance System (ADAS) is an integrated automotive technology designed to enhance vehicle safety by employing sensors, cameras, and machine learning algorithms to assist drivers in various aspects of driving. ADAS encompasses features such as adaptive cruise control, lane departure warning, collision avoidance, and parking assistance. By continuously monitoring the vehicle's surroundings and analyzing real-time data, ADAS aims to provide timely alerts, automate certain driving tasks, and ultimately reduce the risk of accidents. This technology represents a crucial step toward semi-autonomous and autonomous driving, contributing to improved road safety and increased driver convenience.

## X. Conclusions :

The development of an advanced driver assistance system (ADAS) using Raspberry Pi offers significant potential for enhancing road safety and driving experience. Through the integration of sensors, image processing algorithms, and real-time data analysis, this system can provide features such as lane departure warning, collision avoidance, and adaptive cruise control. Additionally, leveraging the versatility and affordability of Raspberry Pi enables scalability and customization for various vehicle types and driving environments. As technology advances, continuous refinement and integration of new functionalities will be essential to ensure optimal performance and compliance with evolving safety standards. Ultimately, the design and development of this ADAS represent a crucial step towards realizing safer and more efficient transportation systems.

## XI. References :

1. Joost de Winter, Jim Hoogmoed ,Jork Stapel,Dimitra Dodou,Pavlo Bazilinskyy, "Predicting perceived risk of traffic scenes using computer vision",Transportation Research Part F: Psychology and Behaviour 93 (2023) 235–247.
2. Brian L. DeCost, Elizabeth A. Holm, "A computer vision approach for automated analysis and classification of microstructural image data", Computational Materials Science 110 (2015) 126–133.
3. Ioannis Mavromatis, Aleksandar Stanoev, Pietro Carnelli,Yichao Jin, Mahesh Sooriyabandara, Aftab Khan, "A dataset of images of public streetlights with operational monitoring using computer vision techniques",Data in Brief 45 (2022) 108658.
4. Matthew Sands, Jinki Kim, "A low-cost and open-source measurement system to determine the Young's and shear moduli and Poisson's ratio of soft materials using a Raspberry Pi camera module and 3D printed parts",HardwareX 13 (2023) e00386.
5. Harry C. Wright, Duncan D. Cameron, Anthony J. Ryan, "FoamPi: An open-source raspberry Pi based apparatus for monitoring polyurethane foam reactions",HardwareX 12 (2022) e00365.