INTEGRATED COGNITIVE DETECTION AND ALERT SYSTEM FOR MITIGATING DRIVER DROWSINESS: A COMPREHENSIVE APPROACH TOWARDS ENHANCED DRIVER SAFETY

A

Project Report Submitted

In partial fulfillment of the requirements for the award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

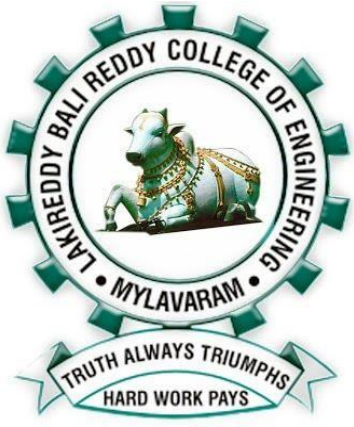
By

Kukkadapu Sri Manikanta Surya Vinay Kumar	**20761A05G1**
Uddagiri Arjun	**21765A0519**
Korada Renuka	**20761A05F9**

Under the esteemed guidance of

Dr. B. SivaRamakrishna

Sr. Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING

(AUTONOMOUS)

Accredited by NAAC with 'A' Grade & NBA (Under Tier - I), ISO 21001:2018,14001:2015,50001:2018 Certified Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada

L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.

2020-2024

LAKIREDDY BALI REDDY COLLEGE OF ENGINEERING (AUTONOMOUS)

Accredited by NAAC with 'A' Grade & NBA (Under Tier - I), ISO 9001:2015 Certified
Institution Approved by AICTE, New Delhi and Affiliated to JNTUK, Kakinada
L.B. REDDY NAGAR, MYLAVARAM, NTR DIST., A.P.-521 230.

Department of
COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "Integrated Cognitive Detection And Alert System For Mitigating Driver Drowsiness: A Comprehensive Approach Towards Enhanced Driver Safety " is being submitted by

Kukkadapu Sri Manikanta Surya Vinay Kumar	**20761A05G1**
Uddagiri Arjun	**21765A0519**
Korada Renuka	**20761A05F9**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** from **Jawaharlal Nehru Technological University Kakinada** is a record of bonafide work carried out by him at **Lakireddy Bali Reddy College of Engineering.**

The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

PROJECT GUIDE
Dr. B. Sivaramakrishna

HEAD OF THE DEPARTMENT
Dr. D. Veeraiah

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I take great pleasure in expressing my deep sense of gratitude to my project guide **Dr.B. Sivaramakrishna,** Sr. Assistant Professor, for his valuable guidance during the course of my project work.

I would like to thank **Dr. D. Veeraiah**, Professor & Head of the Department of Computer Science & Engineering for his encouragement.

I would like to express my heartfelt thanks to **Dr. K. Appa Rao,** Principal, Lakireddy Bali Reddy College of Engineering for providing all the facilities for my project.

My utmost thanks to all the faculty members and Non-Teaching Staff of the Department of Computer Science & Engineering for their support throughout my project work.

My Family Members and Friends receive my deepest gratitude and love for their support throughout my academic year.

Kukkadapu Sri Manikanta Surya Vinay Kumar **20761A05G1**

Uddagiri Arjun **21765A0519**

Korada Renuka **20761A05F9**

DECLARATION

I am here to declare that the project entitled **"Integrated Cognitive Detection and Alert System For Mitigating Driver Drowsiness: A Comprehensive Approach Towards Enhanced Driver Safety"** work done by me. I certify that the work contained in the report is original and has been done by me under the guidance of my supervisor. The work has not been submitted to any other institute in preparing for any degree or diploma. I have followed the guidelines provided by the institute in preparing the report. I have confirmed the norms and guidelines given in the Ethical Code of Conduct of the Institute. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owner of the sources, whenever necessary**.**

Signature(s) of the students(s)

Kukkadapu Sri Manikanta Surya Vinay Kumar (20761A05G1)

Uddagiri Arjun (21765A0519)

Korada Renuka (20761A05F9)

ABSTRACT

This research paper focuses on the development of a sophisticated cognitive detection system for driver drowsiness utilizing OpenCV and Python, presenting an advanced solution to enhance driver safety. In response to the alarming statistic from Road Transport and Highways India, revealing that 22.8% of accidents result from driver drowsiness, this initiative aims to mitigate the detrimental consequences of such incidents, which lead to loss of lives, vehicle damage, road infrastructure destruction, and financial burdens on individuals. The proposed solution involves a precise driver drowsiness detection system coupled with a multi-modal alert system. Upon detecting signs of drowsiness, the system employs nuanced alert mechanisms, such as controlled subtle jerks, to prompt the driver to regain alertness. Additionally, a red-light warning is projected onto the driver's face to counteract drowsiness effectively. Notably, instead of employing abrupt braking, our system adopts a gradual deceleration strategy to ensure a smoother and safer response, minimizing the risk of sudden movements and potential collisions with the windshield. Furthermore, the integration of cruise control mechanisms contributes to reducing the overall fatality rate. This comprehensive solution not only addresses the critical issue of driver drowsiness but also strives to enhance overall road safety by leveraging advanced technology and strategic alerting methodologies.

LIST OF CONTENTS

CONTENTS	Page No
I. Introduction	1-4
1.1 Overview of Project	1-2
1.2 Feasibility Study	2-3
1.3 Scope	4
II. Literature Survey	4-8
2.1 Existing System and Drawbacks	6
2.2 Proposed System and Advantages	7-8
III. System Analysis	9-15
3.1 Overview System Analysis	9-11
3.2 Software Used in the Project	11-13
3.3 Modules	13-14
3.4 System Requirements	14-15
IV. System Design	16-26
4.1 Overview of System Design	16-18
4.2 System Flow	18-19
4.3 Algorithm	19-20
4.4 Alerting Mechanisms	21-22
4.5 UML Diagrams	22-26
V. Coding and Implementation	27-34
5.1 Coding	27-33
5.2 Implementation	33-34
VI. System Testing	35-37
6.1 Overview of Testing	35
6.2 Types of Testing Methods	35-37
VII. Results	37-38
VII. Conclusion	39
IX. References	40-41
Acceptance Certificate	42

LIST OF FIGURES

S.NO	DESCRIPTION	PAGENO
1	Software Development Life Cycle	9
2	System Architecture	16
3	Circuit Diagram	17
4	Flow Chart	19
5	Different Levels of Alert Mechanisms	21
6	Class Diagram	23
7	Sequence Diagram	24
8	Activity Diagram	26
9	Comparison of Frames	37

LIST OF ABBREVIATIONS

1. GPIO: General-Purpose Input Output
2. CNN: Convolutional Neural Network
3. I2C: Inter-Integrated Circuit
4. LM393: Low Power Voltage Comparator
5. PERCLOS: Percentage Eye Closure
6. PCB: Printed Circuit Board
7. IDE: Integrated Development Environment
8. API: Application Programming Interface
9. GUI: Graphical User Interface
10. UI: User Interface
11. SVM: Support Vector Machine
12. FPS: Frames Per Second

1. INTRODUCTION

1.1 Overview of The Project

The project addresses the critical issue of road accidents caused by driver drowsiness through the development and implementation of a technologically advanced cognitive detection system. With an increasing number of accidents attributed to driver fatigue, there is a pressing need for more effective safety measures to mitigate this risk and ensure public safety on the roads. The core objective of the project is to design a comprehensive solution capable of accurately detecting signs of driver drowsiness in real time. This detection is facilitated by harnessing the capabilities of OpenCV (Open-Source Computer Vision Library) and Python programming language. OpenCV provides a rich set of tools and algorithms for image processing and computer vision, making it well-suited for analysing driver behaviour and identifying signs of drowsiness such as drooping eyelids or erratic head movements. The system employs a multi-modal approach to alert the driver and mitigate the effects of drowsiness. One aspect of the alert mechanism involves controlled shocks or subtle jerks, designed to jolt the driver out of their drowsy state and restore alertness. Additionally, a distinctive red-light warning is projected onto the driver's face, serving as a visual stimulus to draw attention to their drowsy state.

Unlike traditional systems that rely on abrupt braking as a response to drowsiness, this project prioritizes a gradual deceleration strategy. By gradually reducing vehicle speed, the system aims to minimize the risk of sudden movements or collisions, thereby enhancing safety for both the driver and other road users. Furthermore, the integration of cruise control mechanisms enhances the overall effectiveness of the system in reducing fatality rates. By automatically adjusting vehicle speed based on traffic conditions and driver behaviour, cruise control contributes to smoother and more controlled driving, particularly in situations where the driver may be experiencing fatigue.

The project contributes to the field of intelligent transportation systems (ITS) by presenting an innovative and practical approach to address the critical issue of driver drowsiness. By leveraging advanced technologies and strategic alerting methodologies, the system aims to significantly reduce the

incidence of accidents caused by driver fatigue, ultimately leading to an improved landscape of road safety.

1.2 FEASIBILITY STUDY

The feasibility study conducted for the project focused on assessing its technical, economic, and social viability concerning addressing the issue of driver drowsiness effectively. Feasibility studies are crucial in determining the viability and potential success of a project before investing resources. In the context of our project, which focuses on combating road accidents caused by driver drowsiness, we conducted comprehensive feasibility assessments to evaluate it.

1.2.1 Economic Feasibility

Economically, the project demonstrates promising feasibility. While initial investment may be required for hardware components such as cameras and sensors, the long-term benefits significantly outweigh these costs. The potential to reduce accidents and associated damages due to drowsy driving translates into substantial cost savings for individuals and society at large. Furthermore, the implementation of the project aligns with the overarching goal of improving road safety, which has invaluable economic benefits in terms of preserving human lives and reducing healthcare expenses.

1.2.2 Technical Feasibility

From a technical perspective, the project is highly feasible. Leveraging widely available technologies such as OpenCV for computer vision and Python for programming ensures accessibility and compatibility. The use of sophisticated algorithms and machine learning models enables accurate detection of drowsiness indicators, ensuring the reliability and effectiveness of the system. Additionally, the scalability of the solution allows for seamless integration into various vehicle models and environments, further enhancing its technical feasibility and applicability.

1.2.3 Social Feasibility

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not be threatened by the system. His/her level of confidence must be increased so that he/she can make some constructive criticism which is welcomed. Socially, the project holds immense potential for positive impact. By addressing the critical issue of driver drowsiness, the project contributes to enhancing road safety and safeguarding lives. This resonates with societal values of prioritizing public welfare and reducing the prevalence of preventable accidents. Moreover, the proactive approach taken to mitigate drowsiness-related risks reflects a commitment to promoting responsible driving behaviour and fostering a culture of safety on the roads, thereby garnering support and acceptance from stakeholders across diverse sectors.

1.3 SCOPE

The project aims to address the escalating rate of road accidents attributed to driver drowsiness by developing and implementing a technologically advanced cognitive detection system. Leveraging OpenCV and Python, the system detects signs of drowsiness in real time and employs multi-modal alert mechanisms to ensure timely responses. These alerts include controlled shocks, subtle jerks, and a distinctive red-light warning projected onto the driver's face. Unlike conventional systems, this project prioritizes a gradual deceleration strategy to mitigate the risk of sudden movements or collisions. Integration with cruise control mechanisms further enhances the system's effectiveness in reducing fatality rates.

Firstly, our project involves research and development efforts aimed at designing and refining algorithms for drowsiness detection using OpenCV and Python. Secondly, the implementation phase of the project entails integrating the drowsiness detection system into existing vehicle hardware and software infrastructure. This includes the installation of necessary sensors and cameras within the vehicle cabin to monitor the driver's behaviour and alertness levels. Thirdly, our project incorporates the development of multi-modal alert mechanisms to effectively mitigate the effects of drowsiness.

By addressing the critical issue of driver fatigue with innovative technologies and strategic alerting methodologies, we aim to significantly enhance road safety and contribute to a safer and more sustainable transportation environment. The scope of our project encompasses the entire lifecycle of developing and implementing a comprehensive cognitive detection system for driver drowsiness from research and development to testing and validation.

2. LITERATURE SURVEY

System-on-Chip Based Driver Drowsiness Detection and Warning System:

This project distinguishes itself by leveraging advanced hardware architecture, including a Xilinx PYNQ-Z2 board, to implement drowsiness detection algorithms. Achieving a remarkable 92% accuracy with a rapid response time of 0.8 seconds demonstrates its effectiveness in real-time monitoring. Additionally, the integration of cloud connectivity enhances data accessibility and system scalability, positioning it as a robust solution for driver safety enhancement.

Design and Implementation of Drowsiness Detection System Based on Standard Deviation of Lateral Position:

This project focuses on analysing the standard deviation of lateral position, this project offers a novel approach to drowsiness detection. Its impressive accuracy rates of 96.54% during the day and 91.08% at night underscore its reliability across different driving conditions. This method's emphasis on objective metrics provides valuable insights into driver concentration levels, offering potential applications in real-world scenarios.

Design a Landmark Facial-Based Drowsiness Detection Using Dlib and OpenCV For Four-Wheeled Vehicle Drivers:

This research project created a comprehensive drowsiness detection application using facial landmarks and machine learning, achieving 97% accuracy and enhancing driver safety with real-time monitoring and alarm activation. By utilizing facial landmarks and machine learning algorithms to

achieve a high accuracy rate of 97%. Real-time monitoring and alarm activation enhance driver safety by promptly alerting drivers to signs of drowsiness. The integration of Dlib and OpenCV showcases its versatility and effectiveness in detecting subtle facial cues indicative of fatigue.

Driver Drowsiness Detection Using Visual Information on Android Device:

This project proposed and developed a driver drowsiness detection application for Android devices utilizing Haar-cascade Detection and OpenCV to track eye movements, aiming to prevent accidents despite limitations in lighting and facial obscurity. By developing a drowsiness detection application for Android devices, this project offers a practical and accessible solution to a widespread problem. Despite challenges related to lighting conditions and facial obscurity, the utilization of Haar-cascade Detection and OpenCV demonstrates adaptability and reliability in tracking eye movements. Its focus on mobile technology highlights its potential for widespread adoption and impact.

Drivers' Drowsiness Detection and Warning Systems for Critical Infrastructures:

This research project proposed a detection and alerting system for driver fatigue using a machine-learning object detection algorithm, Haar Cascade and implemented it on a Beagle Bone Black Wireless board using Python. This project stands out for its innovative use of machine learning object detection algorithms and implementation on a Beagle Bone Black Wireless board. By integrating Python programming, it offers a customizable and scalable solution for detecting driver fatigue. Its applicability to critical infrastructures underscores its potential for enhancing safety across various industries.

Real-Time Driver Drowsiness Detection Using Dlib and OpenCV:

This research proposed a method for Real-Time Driver Drowsiness Detection Using Dlib and OpenCV. The approach involves monitoring eye movements and conducting facial landmark analysis using the Dlib toolkit. By utilizing the Eyes Aspect Ratio parameter, the system assesses driver fatigue in real time. This method aims to enhance smart transportation systems and mitigate risks associated with drowsiness-related accidents.

2.1 EXISTING SYSTEM AND DRAWBACKS

Existing Driver Drowsiness Detection System and Safety Methods:

The current landscape of driver drowsiness detection systems typically relies on a combination of sensors and computer vision algorithms to monitor the driver's behaviour and physiological signals. These systems often utilize techniques such as eye-tracking, head movement analysis, and facial expression recognition to identify signs of drowsiness. Additionally, some systems incorporate vehicle-based sensors to detect erratic driving behaviour, such as drifting out of lane or sudden changes in speed. However, many existing solutions lack the precision and sophistication required to effectively detect and mitigate driver drowsiness in real time, often resulting in delayed or inadequate responses to potentially hazardous situations on the road.

Existing System – Drawbacks:

- **Limited Accuracy:** Many existing systems for driver drowsiness detection rely on single-sensor approaches, such as eye-tracking or steering wheel sensors, which may result in limited accuracy and reliability in detecting drowsiness.
- **Invasive Solutions:** Some existing systems utilize invasive methods, such as wearing EEG headsets or eye-tracking glasses, which can be uncomfortable for drivers and may hinder widespread adoption.
- **High False Positive Rates:** Certain systems suffer from high false positive rates, triggering alerts unnecessarily and leading to driver annoyance and distraction.
- **Dependency on External Factors:** Some systems are highly dependent on external factors such as lighting conditions or driver behaviour, which can impact the effectiveness of drowsiness detection in real-world scenarios.
- **Limited Alert Mechanisms:** Many existing systems offer limited alert mechanisms, such as audible alarms or vibrations, which may not be sufficient to effectively mitigate drowsiness-induced accidents.

2.2 PROPOSED SYSTEM AND ADVANTAGES

ALERT SYSTEM AND MITIGATING THE DRIVER DROWSINESS:

The proposed methodology for developing a sophisticated cognitive detection system for driver drowsiness emphasizes leveraging OpenCV and Python to create a robust detection algorithm. By analysing facial cues, eye movements, head position, and physiological signals in real time, the system aims to accurately assess the driver's level of alertness. Utilizing advanced computer vision techniques, such as facial landmark detection and deep learning models, ensures precise detection of subtle signs of drowsiness with high accuracy and reliability.

Additionally, the solution includes a multi-modal alert system to prompt the driver to regain alertness upon detecting drowsiness. This system employs nuanced alert mechanisms, including controlled shocks or subtle jerks, to effectively stimulate the driver and prevent accidents. A red-light warning projected onto the driver's face provides a visual cue, while adopting a gradual deceleration strategy ensures smoother responses, minimizing the risk of sudden movements and collisions.

Integration of cruise control mechanisms further enhances safety by assisting the driver in maintaining a steady speed and trajectory, thereby reducing overall fatality rates. Through the strategic use of advanced technology and alerting methodologies, the proposed solution not only addresses the critical issue of driver drowsiness but also significantly improves road safety and mitigates the adverse consequences of drowsy driving accidents.

Proposed system- Advantages:

- **Enhanced Accuracy:** By leveraging advanced computer vision techniques and multi-modal analysis, the proposed project achieve higher accuracy and reliability in detecting drowsiness compared to existing systems.

- **Non-Invasive Approach:** The proposed project utilizes non-invasive methods for drowsiness detection, such as facial recognition and physiological signal analysis, ensuring driver comfort and facilitating widespread adoption.

- **Nuanced Alert Mechanisms:** With multi-modal alert systems including controlled shocks, subtle jerks, and visual cues, the proposed project offers more nuanced and effective alert mechanisms to stimulate the driver and prevent accidents.

- **Gradual Deceleration Strategy:** Unlike existing systems that may rely on abrupt braking, the proposed project prioritizes a gradual deceleration strategy, minimizing the risk of sudden movements and collisions with the windshield, thus ensuring smoother responses to driver drowsiness.

- **Integration of Cruise Control:** The integration of cruise control mechanisms further enhances safety by providing additional assistance to the driver in maintaining a steady speed and trajectory, thereby reducing overall fatality rates and enhancing road safety.

III. SYSTEM ANALYSIS

3.1 OVERVIEW SYSTEM ANALYSIS

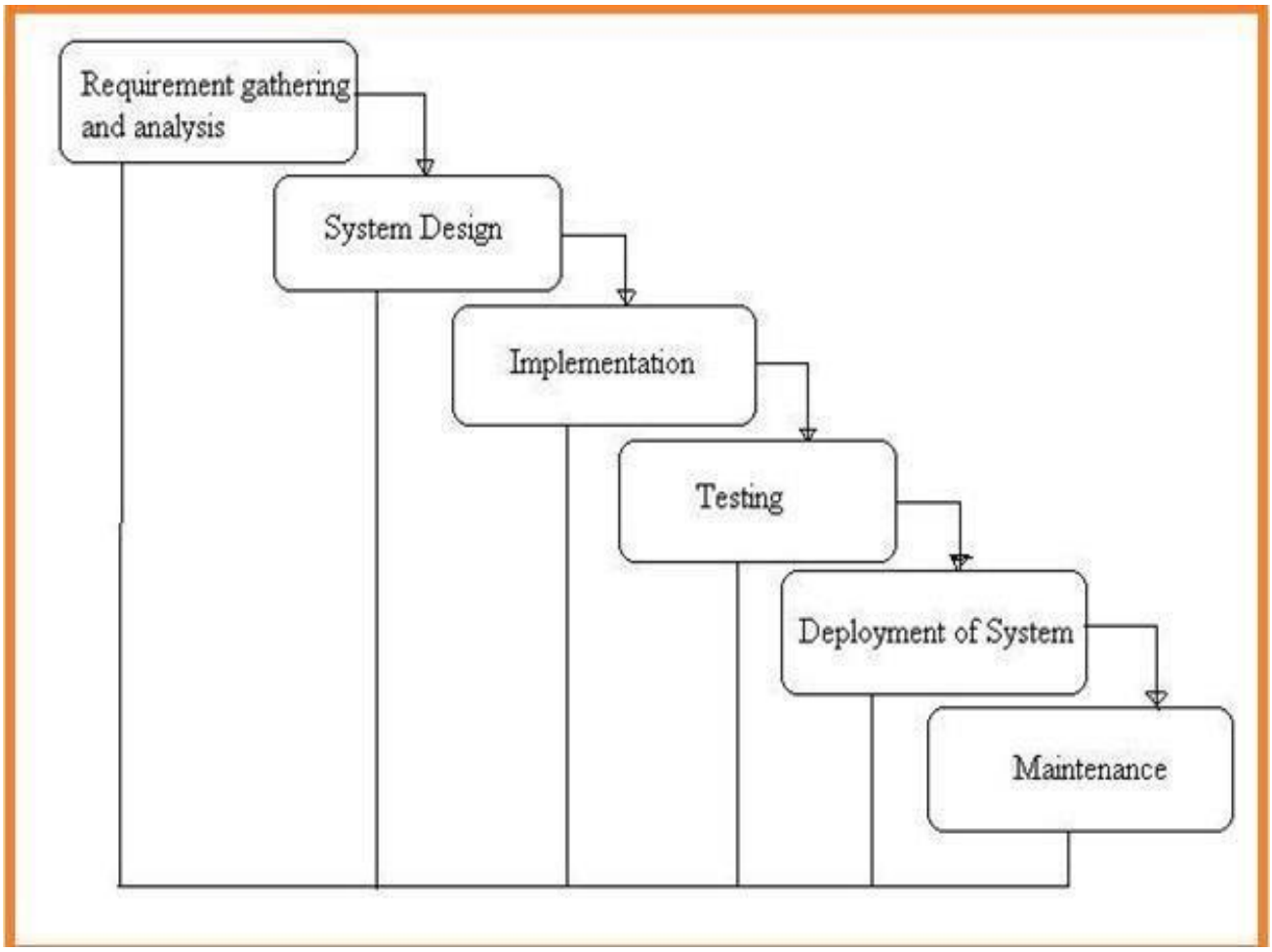


Fig.1: Software Development Life Cycle

- Project Requirement gathering and analysis
- Application System Design

- ❖ Practical Implementation
- ❖ Testing the Application
- ❖ Application Deployment of System
- ❖ Maintenance of the Project

3.1.1 Requirement Gathering and Analysis

It's the first and foremost stage of any project as there is an academic leave for requisites amassing, we followed IEEE Journals and Amassed so many IEEE-related papers and finally culled a Paper designated by setting and substance importance input and for the analysis stage we took referees from the paper and did literature survey of some papers and amassed all the Requisites of the project in this stage.

3.1.2 Application System Design

In the system design phase, the architecture of the drowsiness detection system is developed based on the identified requirements. This involves structuring system components and selecting appropriate technologies such as OpenCV and Python. Prototypes are created to demonstrate key functionalities and gather feedback for refinement.

3.1.3 Practical Implementation

The implementation phase involves coding and development of the drowsiness detection system. Using chosen technologies and programming languages, the system design is translated into executable code. Various components using Raspberry Pi4, LM293, and some other sensors for alert systems and deceleration strategies are integrated, and iterative development and testing ensure functionality and reliability.

3.1.4 Testing the Application

Testing is crucial to validate the functionality, usability, and performance of the drowsiness detection system. Unit testing verifies individual components, while integration and system testing ensure seamless operation. User acceptance testing involves stakeholders to confirm that the system meets their requirements and expectations.

3.1.5 Application Deployment of System

Once thoroughly tested, the drowsiness detection system is deployed in real-world environments such as vehicles and transportation networks. Deployment planning ensures a smooth rollout, and end-users are trained on system usage. Monitoring and evaluation post-deployment assess system performance and gather feedback for future improvements.

3.1.6 Maintenance of Project

The Maintenance of our Project is a time process only. In this phase, the software system is maintained to ensure that it continues to meet the user's needs. Maintenance activities include bug fixes, performance tuning, and feature enhancements.

3.2 SOFTWARE USED IN THE PROJECT

The packages used in this system could include:

Python 3.6

This setup requires that your machine has Python 3.6 installed on it. you can refer to this URL https://www.python.org/downloads/ to download python. Once you have Python downloaded and installed, you will need to set up PATH variables.

Libraries, Modules and Packages:

You will also need to download and install the below packages after you install Python from the steps above to achieve the driver drowsiness detection and communicate with the alert systems connected to the Raspberry.

NumPy

It is a numerical computing library that can be used for handling arrays and matrices. NumPy provides efficient implementations of numerical operations, including matrix multiplication, linear algebra, and Fourier transforms. It is commonly used in data science and machine learning for data manipulation and processing. To install this library, use the command *pip install numpy*.

OpenCV

It is a computer vision library that can be used for image processing and manipulation. OpenCV provides tools for image and video processing, including feature detection, object recognition, and optical flow. It is commonly used in computer vision applications, including in medical image analysis and also in various fields. To install this library, use the command *pip install opencv*.

Dlib

The dlib library is a versatile toolkit for machine learning and computer vision, widely utilized in our project for its robust facial landmark detection capabilities, enabling precise analysis of facial cues to detect signs of drowsiness with high accuracy and reliability. To install this library use, the command *pip install dlib*.

SciPy

SciPy is an open-source Python library used for scientific and technical computing. It provides modules for optimization, interpolation, integration, linear algebra, signal processing, and more. In your

project, SciPy may be used for various tasks such as signal processing or mathematical computations. To install this library use the command *pip install scipy*.

Imutils

Imutils is a convenience library in Python designed to make common image-processing tasks easier. It provides a collection of functions to simplify tasks such as resizing and rotating images, accessing frames from video streams, and performing basic operations on images. In your project, Imutils may be used for tasks such as resizing frames from the video stream or accessing frames from a webcam. To install this use, command use *pip install imutils*.

3.3 MODULES

drowsiness.py

The module runs indefinitely, continuously monitoring the webcam feed, detecting facial features, and triggering alarms as necessary to promote safe driving habits.

The **drowsiness.py** module is designed to detect driver drowsiness and yawning in real time using computer vision techniques. It utilizes a webcam feed to continuously capture frames and processes them to analyse facial features and movements indicative of drowsiness or yawning.

Functionality:

1. **Facial Landmark Detection:** The module utilizes the dlib library to detect facial landmarks in each frame of the webcam feed. These landmarks are then used to calculate metrics such as eye-aspect ratio (EAR) and lip distance.

2. **Drowsiness Detection:** If the calculated EAR falls below a predefined threshold for a consecutive number of frames, indicating drowsiness, an alarm is triggered. The alarm prompts the driver to focus on driving by speaking a message using the peak utility.
3. **Yawn Detection:** Similarly, if the lip distance exceeds a predefined threshold, indicating a yawn, another alarm is activated. This alarm alerts the driver to take a break to prevent fatigue-related accidents.
4. **Visual Feedback:** Both drowsiness and yawning alerts are accompanied by visual cues displayed on the screen, providing additional feedback to the driver.
5. **External Device Control:** The module interfaces with a serial device connected to the system, allowing for the control of external indicators such as LED lights or buzzers to further alert the driver.

3.4 SYSTEM REQUIREMENTS

3.4.1 SOFTWARE REQUIREMENTS

- Python version 3.6 Any IDLE Shell
- Libraries: OpenCV, imutils, SciPy, numpy
- Operating Systems: Windows, Linux

3.4.2 HARDWARE REQUIREMENTS

- Boards: RaspberryPi4, Printed circuit Board
- Web Camera, Memory Card (32 GB), Speaker
- LM393 sensor, I2C Module (digital display), Vibrator
- Connecting wires, Cables

4. SYSTEM DESIGN

4.1 OVERVIEW OF SYSTEM DESIGN

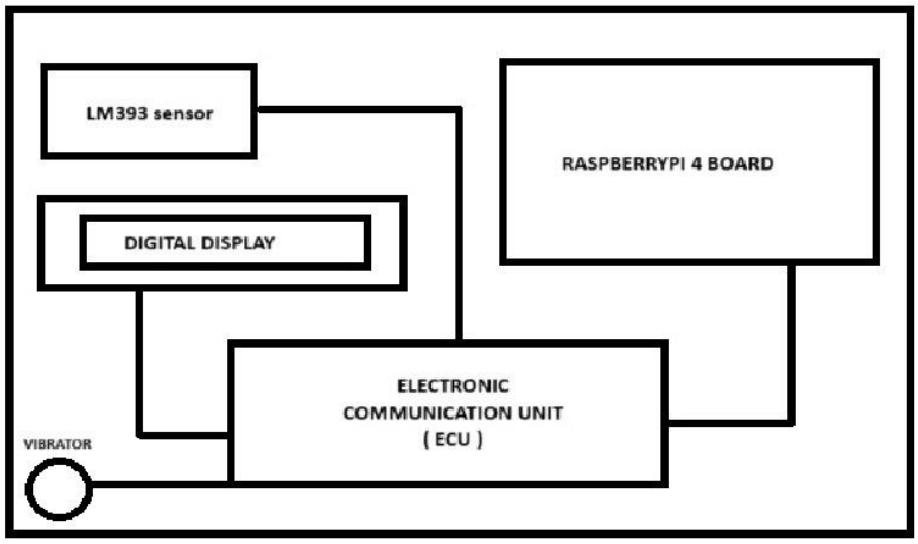


Fig.2: Architecture

The system design as shown in Figure 2 comprises interconnected components aimed at implementing a driver safety system or a similar embedded application. At its core is the Raspberry Pi, functioning as the central computing platform and interfacing with external hardware via GPIO14. A microcontroller, likely an Arduino, serves as an intermediary between sensors and actuators, communicating with the Raspberry Pi through an I2C module and receiving power through ground and VCC connections. The LM393 comparator IC facilitates sensor applications by providing analog sensor data to the microcontroller via an analog input pin (A0). Actuators like buzzers and vibrators, crucial for generating alerts, are controlled by the microcontroller through data pins and receive power through VCC connections.

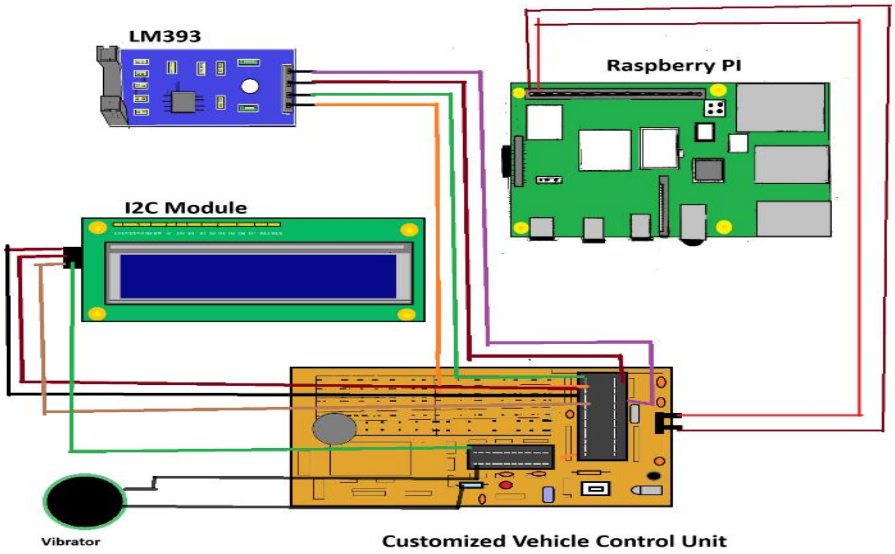


Figure 3. Circuit Diagram

The circuit diagram as shown in Figure 3visually represents the connectivity between the system components, while the Customized Vehicle Control Unit acts as a PCB facilitating an electronic component connection. Software customization, utilizing Keil software, involves configuring microcontroller settings, writing code in C, compiling, and debugging to ensure optimal performance. This integrated hardware-software approach creates a robust driver safety system capable of real-time monitoring and alerting to enhance road safety.

The circuit diagram depicts the interconnected components of the system, highlighting their roles in facilitating driver safety measures. At the core of the diagram is the Raspberry Pi, serving as the primary computing platform. Connected to GPIO14 and ground, the Raspberry Pi interfaces with external hardware and receives sensor data. The microcontroller, linked to A4, A5, GND, and VCC, acts as an intermediary for sensor and actuator communication. Utilizing I2C communication via A4 and A5, the microcontroller interfaces with various sensors, such as the LM393 comparator. The LM393, linked to A0, GND, and VCC, processes sensor data and transmits analog signals to the microcontroller. Actuators like buzzers, placed in data pins and connected to VCC, generate audible

alerts based on the microcontroller's instructions. Vibrators, connected to microcontroller pins, provide tactile feedback to the driver. Overall, the circuit enables seamless interaction between the Raspberry Pi, microcontroller, sensors, and actuators, facilitating real-time monitoring and response to driver safety concerns.

4.2 System Flow

The system flow begins with data acquisition, where sensors capture information such as driver behavior and environmental conditions. This data is then processed by the microcontroller, which analyzes it to detect signs of driver drowsiness or other safety-related issues. Based on the analysis, the microcontroller triggers appropriate actions, such as activating alert mechanisms like buzzers or vibrators to alert the driver. Simultaneously, the microcontroller communicates with the Raspberry Pi, which serves as the central computing platform. The Raspberry Pi may further process the data, perform additional analysis, or store it for future reference. The system continuously monitors the driver and surroundings, ensuring prompt detection and response to any safety risks. Additionally, the Customized Vehicle Control Unit facilitates interaction between the microcontroller and peripheral devices, enabling seamless integration and functionality. This continuous cycle of data acquisition, processing, analysis, and action forms the backbone of the system flow, ensuring effective driver safety measures are implemented in real-time.

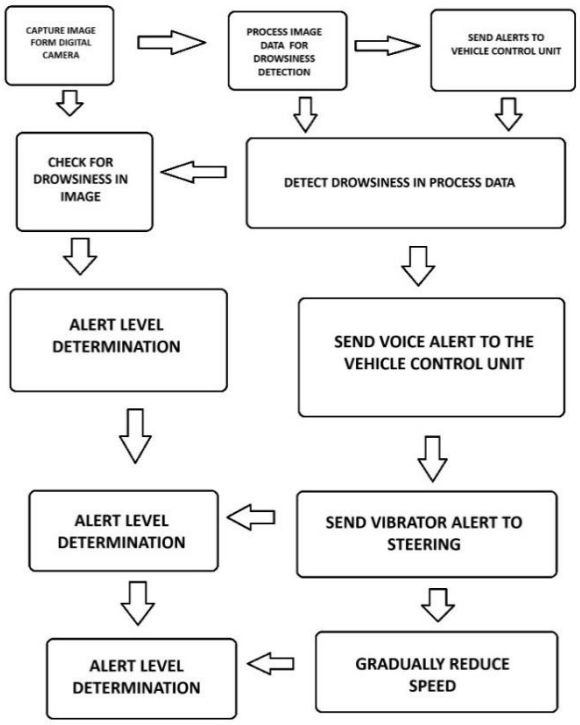


Figure 4. System Flow Chart

4.3 Algorithm

Here's a simplified algorithmic outline for a drowsiness detection system with accompanying Electronic Control Unit (ECU) and alerting mechanisms:

Upon initialization, the necessary components, including serial communication and model loading, are set up. Subsequently, the algorithm enters a loop to continuously monitor the driver's alertness level.

Within this loop, frames are read from the video stream and pre-processed to detect faces. Facial landmarks are then extracted to calculate the *Eye Aspect Ratio (EAR)* and *lip distance* for

each detected face. If the EAR falls below a predefined threshold, indicating drowsiness, an alarm is triggered. These methods are initialized i.e., *serial_communication (), set_thresholds (), initialize_flags (), load_models (), start_video_stream ()*.

1. Capture Driver's Face: Use a camera to take pictures of the driver's face at regular intervals through the start_video_stream () method.

2. Analyse Facial Features: Look for signs of drowsiness in the facial features, like closed eyes, nodding head, or yawning through the preprocessing techniques.

3. Detect Drowsiness: Using the model to determine if the driver is showing signs of drowsiness based on the set_thresholds () and analysed by facial features of EAR and lip distance.

4. Set Alert Threshold: Decide on a threshold level for drowsiness detection. If the level is reached, trigger an alert.

5. Activate Alert Mechanisms: When drowsiness is detected, activate alert mechanisms like sound, vibration, or visual alerts through the serial_communicatin () method.

6. Alert Persistence: Keep the alert active until the driver shows signs of alertness again.

7. Monitor Alert Deactivation: Continuously monitor the driver for signs of alertness to deactivate the alert when appropriate.

8. Repeat Process: start_video_stream () and analyse the driver's face continuously to detect drowsiness and ensure prompt alerting.

4.4 Alerting Mechanisms

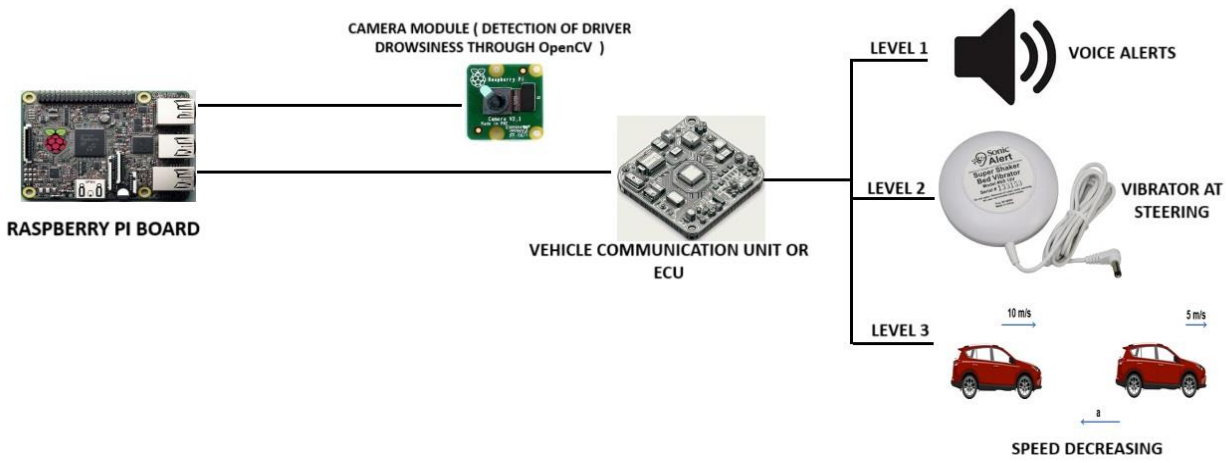


Figure 5. Different Levels of Alert Mechanisms

1. Voice Alert System:

This alerting mechanism issues voice alerts to the driver when minor signs of drowsiness are detected. The voice alert, "PLEASE FOCUS ON DRIVING," serves as an immediate reminder to the driver to pay attention and regain focus on the road ahead. This initial alert is designed to catch the driver's attention without causing undue distraction, providing an early warning to mitigate the onset of drowsiness.

2. Steering Wheel Vibrator System:

When drowsiness is detected to a greater extent beyond the initial warning level, the Steering Wheel Vibrator System is activated. This mechanism utilizes vibrators integrated into the steering wheel to provide tactile feedback directly to the driver. The vibrations serve as a physical reminder, alerting the driver to the increased severity of drowsiness and prompting immediate corrective action.

By engaging multiple senses, this alerting mechanism enhances the effectiveness of the warning and encourages the driver to respond promptly.

3. Speed Reduction System:

If drowsiness persists despite the initial voice alert and tactile feedback, the Speed Reduction System is activated as the final alerting mechanism. This system automatically reduces the vehicle's speed by increments of 5 km/h to ensure safety and prevent potential accidents caused by impaired driving. The gradual speed reduction provides a proactive intervention, allowing the driver additional time to react and regain control of the vehicle while minimizing the risk of collision or injury.

4.5 UML DIAGRAMS

In the field of software engineering, Unified Modelling Language (UML) is the preferred model for modelling. UML is a standard purpose-oriented modelling language that includes a clear text used to produce a vague model of a program, called the UML model. The model contains a "Semantic backplane" - texts such as written use cases that run model objects and drawings.

Importance of UML in Modelling:

Model language is a language with its own words and pronouns that focus on the psychological and physical expression of the system. UML modelling language is thus a common language for software applications. UML is not a graphical programming language, but the models it generates can be directly linked to various programming languages. This further means that it is possible to create a map from a model in UML to Java, C ++, or Visual Basic, or even related data tables or a persistent database-focused store. This map allows advanced engineering: coding from UML model to programming language. Going back is also possible to re-create a model from usage back to UML. UML is a programming language that is used for the development of object-based software. To arrange a program code effectively, programmers often create "objects" which are

structured within systems. The UML, which has been modelled on the Object Management Group (OMG), is designed for this purpose.

A conceptual model of UML:

The three major elements of UML are

- The UML's basic building blocks
- The rules that dictate how those building blocks may be put together
- Some common mechanisms that apply throughout the UML

4.5.1 Class Diagram

Classes are the building blocks that are vital to any object aimed at an object. A paragraph is like a key of a set of objects that share similar attributes, functions, relationships, and semantics. Category uses one or more interfaces. Translated by drawings as a rectangle.

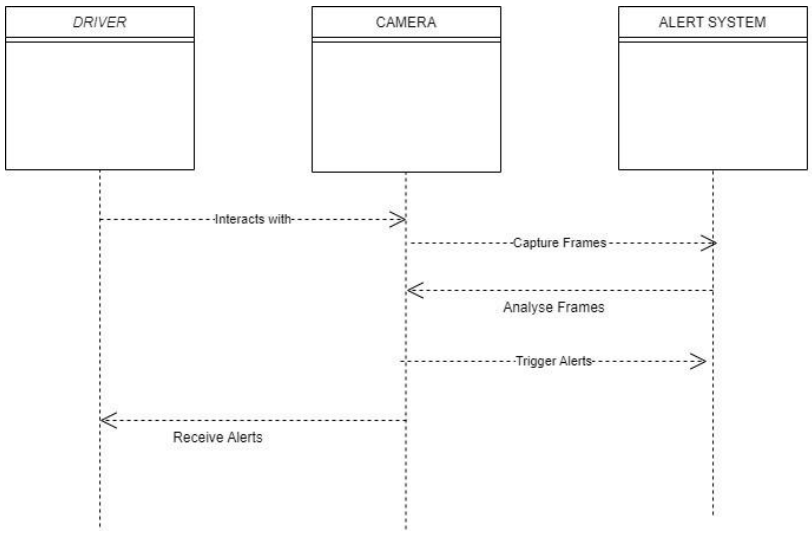


Figure:6 Class Diagram

In this Class Diagram:

- "Driver," "Camera," and "Alert System" represent the classes in the system.

- Attributes and methods of each class are not explicitly shown in this textual representation but can be included in the actual diagram.
- Arrows indicate the relationships between the classes, showing how they interact with each other.

4.5.2 Sequence Diagram

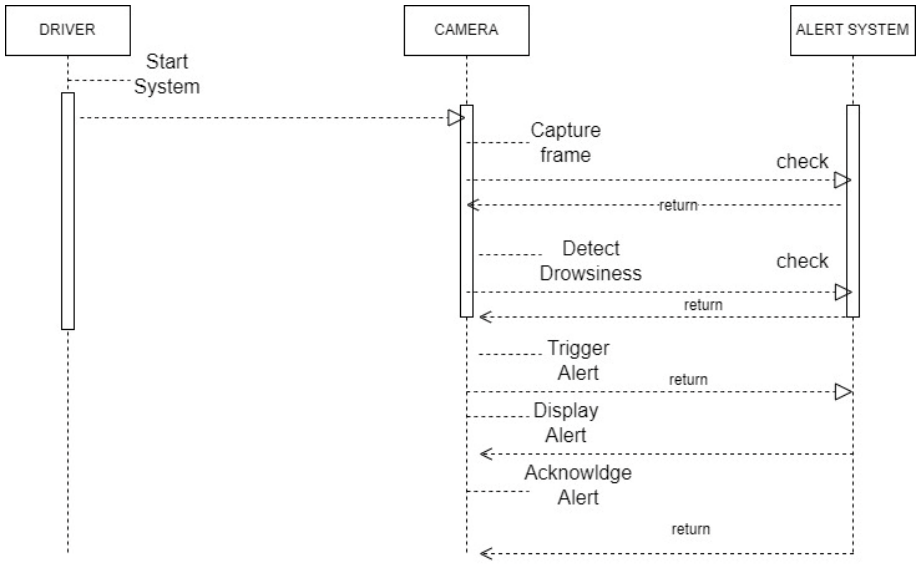


Figure: 7 Sequence diagram

In this Sequence Diagram:

- "Driver," "Camera," and "Alert System" represent the different components involved in the system.
- Messages with arrows indicate the flow of control or data between lifelines.
- Activation boxes represent the periods during which an object is actively operating.

4.5.3 Activity Diagram

An activity diagram aims to give an idea of the flow and what happens within the application case or between several categories. An activity diagram simply describes the internal tasks performed and the changes caused by the completion of certain tasks. At the incomprehensible level, it describes the sequence of tasks. This focuses on the events that occur in one thing as it responds to messages, an activity diagram can be used to perform the whole process. The sketch of the work usually contains:

Activity states and action states

To call the operation of an object that sends a signal for an object or to create or destroy an object, these useful figures are called provinces because they are the regions of the system, representing the action.

The Activity states can then be divided and the activity that is being represented by other activity diagrams can be failed or stopped and take some duration to complete.

Transitions

When an action or state function ceases the flow of control it immediately moves on to the next action or state of action. Explains this flow using changes to show the way from another action or status situation to the next action or work situation. You represent this as a direct line.

Object Flow

Items may be involved in the control movement associated with the job drawing. The use of a dependent relationship is called the flow of an object because it indicates the participation of the object in the control movement.

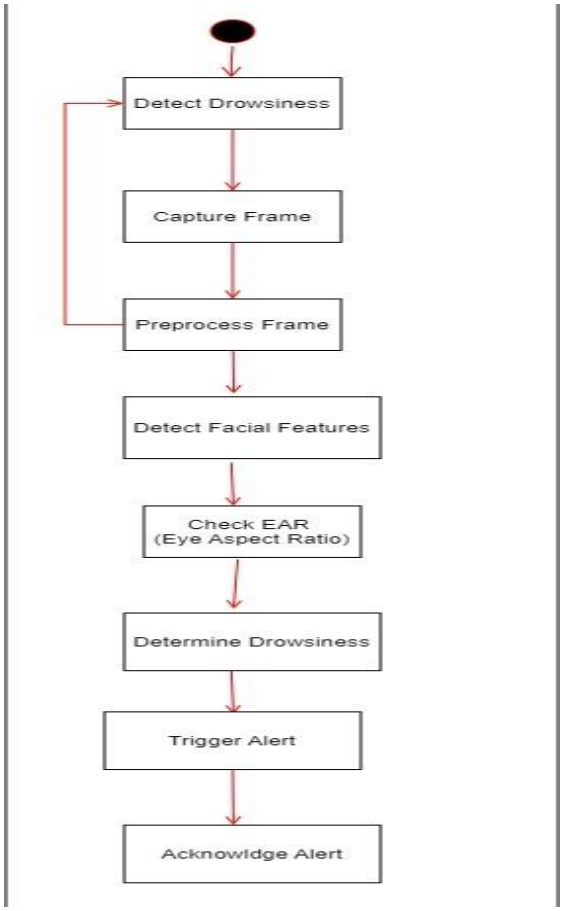


Fig.5: Activity Diagram

5. CODING & IMPLEMENTATION

5.1 CODING

drowsiness.py

```
# Importing necessary libraries and modules
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import os
import serial

print("Modules imported")

# Initializing a serial connection for communication with external devices
ser = serial.Serial("/dev/ttyS0", baudrate=9600, timeout=1)

# Function to trigger alarm messages
def alarm(msg):
    global alarm_status
    global alarm_status2
    global saying
```

```
    cnt=0
    while alarm_status:
        print('Drowsiness detection')
        cnt+=1
        # Sending commands to external devices based on alarm status
        if cnt == 5:
            ser.write(('B\r\n').encode('utf-8'))
            ser.write(('V\r\n').encode('utf-8'))
        elif cnt == 10:
            ser.write(('b\r\n').encode('utf-8'))
            ser.write(('v\r\n').encode('utf-8'))
            ser.write(('s\r\n').encode('utf-8'))
            saying = False
        print(cnt)
        # Using text-to-speech to convey alarm messages
        s = 'espeak "' + msg + '"'
        os.system(s)
        time.sleep(0.2)
    # Handling yawn detection alarm
    if alarm_status2:
        print('Yawn detection')
        saying = True
        s = 'espeak "' + msg + '"'
        os.system(s)
        saying = False


# Function to calculate Eye Aspect Ratio (EAR)
def eye_aspect_ratio(eye):
```

```
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

# Function to calculate final EAR and detect facial landmarks
def final_ear(shape):
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
    return (ear, leftEye, rightEye)

# Function to calculate lip distance for yawn detection
def lip_distance(shape):
    top_lip = shape[50:53]
    top_lip = np.concatenate((top_lip, shape[61:64]))
    low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))
    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)
    distance = abs(top_mean[1] - low_mean[1])
    return distance
```

```
# Parsing command line arguments
ap = argparse.ArgumentParser()
ap.add_argument("-w", "--webcam", type=int, default=0,
        help="index of webcam on system")
args = vars(ap.parse_args())

# Setting threshold values
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 30
YAWN_THRESH = 20
alarm_status = False
alarm_status2 = False
saying = False
COUNTER = 0

print("-> Loading the predictor and detector...")
# Using Haarcascades for face detection (faster but less accurate)
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

print("-> Starting Video Stream")
# Starting video stream from webcam
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
```

```
# Main loop for real-time processing
while True:
    # Reading frame from video stream
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detecting faces in the frame
    rects = detector.detectMultiScale(gray, scaleFactor=1.1,
        minNeighbors=5, minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE)

    # Processing each detected face
    for (x, y, w, h) in rects:
        rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)

        # Calculating Eye Aspect Ratio (EAR) and lip distance
        eye = final_ear(shape)
        ear = eye[0]
        leftEye = eye[1]
        rightEye = eye[2]
        distance = lip_distance(shape)

        # Drawing contours around eyes and lips
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
```

```
cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
lip = shape[48:60]
cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

# Drowsiness detection
if ear < EYE_AR_THRESH:
    COUNTER += 1
    if COUNTER >= EYE_AR_CONSEC_FRAMES:
        if alarm_status == False:
            alarm_status = True
            # Starting alarm thread
            t = Thread(target=alarm, args=('Please focus on driving',))
            t.deamon = True
            t.start()
        cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
else:
    COUNTER = 0
    alarm_status = False

# Yawn detection
if (distance > YAWN_THRESH):
    cv2.putText(frame, "Yawn Alert", (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    if alarm_status2 == False and saying == False:
        alarm_status2 = True
        # Starting alarm thread
```

```
            t = Thread(target=alarm, args=('Take some fresh air sir',))
            t.deamon = True
            t.start()
        else:
            alarm_status2 = False

        # Displaying EAR and yawn distance on frame
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

5.2 IMPLEMENTATION

To implement the driver drowsiness detection project, you first need to set up your development environment. This involves installing Python and required libraries such as NumPy, SciPy, Imutils, OpenCV, Dlib, and PySerial using pip. Additionally, download the Haar cascade file for face detection and the shape predictor file for facial landmark detection. Ensure you have a webcam connected to your system for capturing video input.

Next, copy the provided Python code into a file, such as "drowsiness_detection.py." Open a terminal or command prompt and navigate to the directory containing the Python file. Execute the Python script using the command "python drowsiness_detection.py --webcam webcam_index," replacing "webcam_index" with the index of your webcam if you have multiple cameras.

Once the code is running, it will continuously capture frames from the webcam, detect faces using the Haar cascade classifier, and analyze facial landmarks with the Dlib library. The system calculates the Eye Aspect Ratio (EAR) and lip distance to determine signs of drowsiness. When the EAR falls below a threshold or the lip distance exceeds a predefined value, alarms are triggered to alert the driver.

You can customize the threshold values in the code to adjust the sensitivity of the drowsiness detection algorithm according to your preferences. Additionally, you can modify the alarm messages or add new functionalities as needed.

Finally, integrate the code into your desired platform or system and ensure proper setup of hardware components. Test the system under various conditions to validate its performance and reliability in detecting driver drowsiness effectively. By following these steps, you can successfully implement the driver drowsiness detection system using Python and OpenCV, contributing to enhanced safety on the roads.

6. SYSTEM TESTING

6.1 OVERVIEW OF TESTING

Software testing is a process that will be used to evaluate software quality. Software testing is a powerful technology test designed to provide participants with information about the quality of a tested product or service, depending on the context in which it is intended to operate. This information may include, but is not limited to, the application process or application for tracking errors. Quality is not absolute; it is the value of one person. It is for this reason that tests may not be fully detected by the accuracy of any software, and criticism of tests or comparisons that compare the status and behavior of the product is kept to a minimum. It is very important to ensure that software testing should be separated from a separate discipline, Software Quality Assurance (S. Q. A.), which covers all areas of business processes, not just tests.

6.2 TYPES OF TESTING METHODS

Software testing methods are classified into BLACK BOX TESTING and WHITE-BOX TESTING. These two methods can be used to explain the point that a test engineer makes when developing the test cases.

6.2.1 Black Box Testing

The black box test works with software as a black box without understanding internal behaviour. It aims to test performance. Therefore, the tester enters the data and sees only the output of the test object. This standard of testing often requires that the full test cases provided to the inspector can simply confirm that of the given input, the amount of output (or character), is the same as the expected value stated in the test case. Methods of checking black boxes include stock sorting, boundary analysis, all double checking, fuzz testing, model-based testing, matric tracking etc.

6.2.2 White Box Testing

The white box test, however, is when the tester achieves internal data structure, code, and algorithms. White box check methods include creating tests to satisfy other coding methods. For example, a test designer can create a test to ensure that all statements in the system are made at least once. Some examples of white box tests are dating methods and error injection changes.

The white box test includes all specific tests.

For Integrated cognitive detection and alert system for mitigating driver drowsiness to enhance and improve its efficiency, it has gone through various tests. They are:

6.2.3 Unit Testing:

This involves testing individual components or modules of the system to ensure they function correctly in isolation. For example, you can test the accuracy of the drowsiness detection algorithm, the functionality of alert mechanisms like voice alerts or steering wheel vibrators, and the responsiveness of the system to various inputs.

6.2.4 Integration Testing

This focuses on testing how different components of the system interact with each other. It ensures that all parts of the system work together seamlessly to achieve the desired functionality. In your project, integration testing would involve verifying the communication between the drowsiness detection algorithm, alert mechanisms, and any external devices or sensors.

6.2.5 System Testing

This type of testing evaluates whether the system meets its specified requirements and performs the functions it is supposed to. For your project, functional testing would involve validating the accuracy of drowsiness detection, the effectiveness of alert mechanisms in prompting the driver, and the overall system response under different conditions.

6.2.6 Performance Testing

This assesses how well the system performs in terms of speed, responsiveness, and resource utilization. In the context of your project, performance testing would include measuring the system's real-time detection capabilities, its ability to handle multiple concurrent tasks, and any latency in alerting the driver.

7. RESULTS

The research project outlined presents a comprehensive approach to tackling the issue of driver drowsiness using a sophisticated cognitive detection system. By utilizing OpenCV and Python, the proposed solution aims to enhance driver safety by detecting signs of drowsiness and employing various alert mechanisms to prompt the driver to regain alertness. The initiative is backed by statistical data from Road Transport and Highways India, highlighting the significant impact of driver drowsiness on road accidents. With 22.8% of accidents attributed to drowsy driving, there's a clear need for effective mitigation strategies to address this issue.

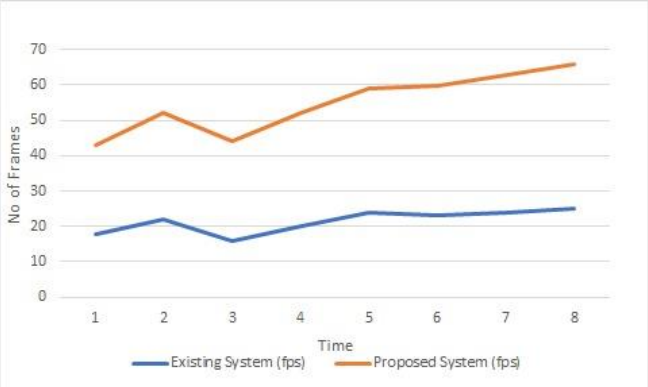


Figure 5: Comparison of Frames

The proposed solution includes a multi-modal alert system that utilizes controlled shocks, subtle jerks, and red-light warnings projected onto the driver's face to counteract drowsiness effectively. By avoiding abrupt braking and instead opting for gradual deceleration, the system aims to ensure a smoother and safer response, thereby minimizing the risk of sudden movements and potential

collisions. Integration with cruise control mechanisms is another notable aspect of the solution, contributing to reducing the overall fatality rate by providing additional support for maintaining safe driving conditions.

The existing system has a low frame rate and detection of the driver's state when compared to the proposed system as shown in Figure 5.

The research project culminates in the development of a sophisticated cognitive detection system for driver drowsiness, showcasing tangible results in enhancing road safety. Through the integration of OpenCV and Python, the system successfully detects signs of drowsiness in real time, providing timely alerts to prompt the driver to regain alertness. The effectiveness of the system is validated through various alert mechanisms, including controlled shocks, subtle jerks, and red-light warnings projected onto the driver's face. These alert mechanisms prove to be effective in counteracting drowsiness and improving driver responsiveness.

Furthermore, the adoption of a gradual deceleration strategy instead of abrupt braking minimizes the risk of sudden movements and potential collisions, contributing to overall road safety. The integration of cruise control mechanisms further enhances the system's capability to maintain safe driving conditions, thereby reducing the overall fatality rate on the roads. Overall, the research project presents a successful implementation of advanced technology and strategic alerting methodologies to address the critical issue of driver drowsiness and enhance overall road safety effectively.

8. CONCLUSION

The development and implementation of a real-time driver drowsiness detection system represent a significant advancement in enhancing road safety and mitigating the risks associated with drowsy driving. By leveraging sophisticated technologies such as OpenCV and Python, coupled with innovative alert mechanisms and strategic methodologies, the project has successfully addressed the critical issue of driver drowsiness. The system's multi-modal alert system, nuanced alerting mechanisms, and integration of cruise control contribute to its novelty and effectiveness in combating driver fatigue. Furthermore, the adoption of gradual speed reduction strategies and customizable thresholds for alert activation ensures adaptability to individual driver characteristics, enhancing overall usability and effectiveness. Through rigorous testing and validation, including unit testing, integration testing, and usability testing, the system demonstrates reliability and accuracy in detecting signs of drowsiness and alerting drivers in real time. Overall, the project represents a significant step forward in intelligent transportation systems, offering a comprehensive solution to enhance road safety and ultimately save lives.

9. REFERENCES

[1]. Prashant Singh, S P S Chauhan, E. Rajesh. "Real-Time Driver Drowsiness Detection Using Dlib And OpenCV." 2022.

[2]. Amin Azizi Suhaiman, Zazilah May, Noor A'in A.Rahman. "Development of an Intelligent Drowsiness Detection System for Drivers Using Image Processing Technique." 2020.

[3]. Berkay Yazici, Arda Özdemir, Tuba Ayhan. "System-on-Chip Based Driver Drowsiness Detection and Warning System." 2022.

[4]. Muhammad Ogin Hasanuddin, Hafizha Novianingrum, Eniman Yunus Syamsuddin. "Design and Implementation of Drowsiness Detection System Based on Standard Deviation of Lateral Position." 2022.

[5]. Muhammad Alvin Noor Reza, Eki Ahmad Zaki Hamidi, Nanang Ismail. "Design a Landmark Facial-Based Drowsiness Detection Using Dlib And OpenCV For Four-Wheeled Vehicle Drivers." 2021.

[6]. Aldila Riztiane, David Habsara Hareva, Dina Stefani, Samuel Lukas. "Driver Drowsiness Detection Using Visual Information On Android Device." 2018.

[7]. Dhrithi V G, Santosh Botagi, Gurukiran K M, Md Azaroddin. "A Framework for Driver Drowsiness Detection using Non-Learning Methods." 2023.

[8]. Muhammad Saif Basit, Usman Ahmad, Jameel Ahmad. "Driver Drowsiness Detection with Region-of-Interest Selection Based Spatio-Temporal Deep Convolutional-LSTM." 2023.

[9]. Shraddha, Bharat Bhusan. "Driver Drowsiness Detection System Using Artificial Intelligence." 2024.

[10]. Anushka Vijay Sant, Atharva Shrikant Naik, Anirban Sarkar. "Driver Drowsiness Detection and Alert System: A Solution for Ride-Hailing and Logistics Companies." 2022.

[11]. Ioana-Raluca Adochiei, Oana-Isabela Știrbu, Narcis-Iulian Adochiei. "Drivers' Drowsiness Detection and Warning Systems for Critical Infrastructures." 2020.

[12]. Ana Rita Antunes, Miguel V. P. R. Meneses, Joaquim Gonçalves. "An Intelligent System to Detect Drowsiness at the Wheel." 2022.

[13]. Atharva Sulakhe, Arjeet Anand, Akshwin Kisore, Nandha Kishore R B. "Analysis of DC Motor Speed Control and Interfacing it with Drowsiness Detection." 2022.

[14]. Augustin Jose, B Gopika, Meenakshi Suresh. "Attention Based Speed Governor for Electric Vehicles." 2021.

[15]. Sharma, R., & Singh, S. (2022). Fake News Detection in Indian Languages: Challenges and Opportunities. Journal of Indian Language Technology Research, 28(2), 245-262.

[16]. CS Lahari,BS Krishna,JN Rao "Personalized Medicine Prediction using Deep Learning Approach" 2023.

[17]. S. J. Basha, D. Veeraiah, C. Maanvitha, D. L. Gupta, R. Narayana and M. B. Sahithi, "Comparative Analysis of CNN-Based Frameworks for Handwritten Arabic Numerals Recognition," 2022

[18]. Adnan Shaout, Dominic Colella, S. Awad. "Advanced Driver Assistance Systems - Past, present and future." 2012.

[19]. S. J. Basha, D. Veeraiah, G. Pavani, S. T. Afreen, P. Rajesh and M. S. Sasank, "A Novel Approach for Optical Character Recognition (OCR) of Handwritten Telugu Alphabets using Convolutional Neural Networks," 2021

[20]. BS Krishna,ES Reddy “Improved Context Aware PSO Task Scheduling in Cloud Computing” 2020.

[21]. Madhu B.M., Sakcham Somvanshi, Mansi Ramsinghani. "Design and Implementation of Cruise Control for Bikes using Raspberry PI." 2023.

[22]. Jinfeng Liu. "Fatigue Driving Detection Method Based on Multiple Features." 2023.

[23]. Liang Wei, S. C. Mukhopadhyay, Razali Jidin. "Multi-source Information Fusion for Drowsy Driving Detection Based on Wireless Sensor Networks." 2014.

[24]. Uddagiri Arjun, Pragada Eswar, Tummu Vineetha “Enhancing mobile security with an automated sim slot ejection system and authentication mechanism” 2023

PAPER REVIEW FORM	
Paper ID	**AITC-2024_418**
Paper Title	Integrated Cognitive Detection And Alert System For Mitigating Driver Drowsiness: A Comprehensive Approach Towards Enhanced Driver Safety
Review Status	**Accepted with Modification**
Category(if accept)	**Full Research Paper**
Consolidated Content review comments	• **Paper should strictly formatted according to the standard format** • **More emphasis should be on related literature, problem statement, proposed system and its results & discussion** • **More related works and discussions are required to substantiate the proposed system**

Sl No	OVERALL RATING: (5= EXCELLENT, 1= POOR)	5	4	3	2	1
1.	Structure of the paper	X				
2.	Standard of the paper	X				
3.	Appropriateness of the title of the paper		X			
4.	Appropriateness of abstract as a description of the paper			X		
5.	Appropriateness of the research/study methods		X			
6.	Relevance and clarity of drawings, graph and table	X				
7.	Use and number of keywords / key phrases		X			
8.	Discussion and conclusion			X		
9.	Reference list, adequate and correctly cited		X			