Project Report

AI-Powered Driver Assistance System for Integrated Safety and Alertness Monitoring (ADAS)

Group Number: 01

Course: Microprocessor Interfacing-EE303

Submitted By:

Name	Roll Number
Paluri Sai Sasidhar **(Group Leader)**	2023BCS-045
Ramavath Chanti Balaji	2023BCS-054
Rathlavath Ramesh	2023BCS-055
Sure David	2023BCS-066
Yellamandala Sameer	2023BCS-073

Date of Submission: October 13, 2025

Submitted To:
Dr. Gaurav Kaushal
Department of Electrical Electronics Engineering

Table of Contents

1. Abstract

This project aims to develop a real-time, multi-modal Advanced Driver Assistance System built on an edge computing platform (Raspberry Pi 5) to actively mitigate the primary risks associated with driver fatigue, distraction, and unsafe driving behavior (rash driving, unmanaged turns).

---

2. Components Used

The system utilizes a combination of embedded hardware and specialized software libraries for high-performance, real-time image processing.

2.1. Hardware Components

Component	Function
Raspberry Pi 5	Main processing unit for concurrent execution of AI models (Drowsiness, Road Sign, and Behavioral analysis). Selected for its powerful BCM2712 CPU and optimized parallel processing capability (2-3$\times$ faster than previous models).
USB Webcams (x2)	Dual setup: One camera monitors the driver (Driver Monitoring System - DMS), and the second monitors the road view (Forward View Perception).
Speaker/Buzzer	Provides instant, non-visual **audio alerts** (using alarm.wav) for critical events like drowsiness or detected hazards.
Monitor/Display	Dashboard interface (to be built with **Pygame**) for visual overlays, real-time camera feeds, speed, and system status information.
Memory Card (16GB)	Dumping the model, code, etc required for the functioning of the raspberry pi 5.

2.2. Software Language and Libraries

Component	Role	Application in Project
Software Language	**Python 3.9+**	Primary implementation language for all components.
OpenCV (cv2)	Real-Time Computer Vision	Handles video streams, image preprocessing, and displaying visual output.
Dlib	Facial Landmark Detection	Provides the pre-trained model for accurate face and landmark extraction (68-points), crucial for EAR/MAR.
scipy.spatial.distance	Mathematical Utilities	Specifically used for the dist.euclidean function to calculate the distances between facial landmarks for EAR and MAR metrics.
imutils	Computer Vision Helpers	Utility functions like face_utils and helper functions for thread-safe video stream handling.
Keras	Deep Learning API (Model Building)	Used to **define, compile, and train** the Convolutional Neural Network (CNN) model for classifying the 43 classes in the **GTSRB dataset**.
NumPy	Scientific Computing	Essential for handling

		and manipulating the large array structures of image data and geometric points efficiently.
scikit-learn	Machine Learning Utilities	Used for preprocessing tasks like splitting datasets and one-hot encoding sign labels for training.
TensorFlow	Core Deep Learning Framework	The backend for Keras, used for model optimization and converting the final Road Sign model to **TensorFlow Lite** format.
TensorFlow Lite	Edge Inference Engine	Used for efficient, optimized deployment and execution of the Road Sign Recognition model on the Raspberry Pi 5.
Pygame	Graphical/Audio Toolkit	Used to create the dashboard interface and manage **audio alerts** (pygame.mixer) for immediate driver warnings.

---

3. Source of the Dataset

3.1. Drowsiness Detection

The drowsiness and face analysis models rely primarily on a geometric approach applied to facial landmarks, which requires minimal direct training data, but uses highly accurate pre-trained models.

- **Facial Landmark Model:** The project uses the **Dlib 68-point shape predictor model** (shape_predictor_68_face_landmarks.dat), which was trained on large-scale datasets such as the **iBUG -W dataset**.
- **Threshold Calibration:** Thresholds (EYE_AR_THRESH, MOUTH_AR_THRESH) are determined empirically and require testing against a custom dataset of drowsy and alert driver video clips to ensure local robustness.

3.2. Road Sign Recognition

- **Primary Dataset:** The traffic sign recognition module will be trained using standard, globally recognized datasets, such as the **German Traffic Sign Recognition Benchmark (GTSRB)** or the **Indian Traffic Sign Dataset (ITSD)**. These provide a large variety of illuminated, partially obscured, and rotated signs for robust model development.
- GTSRB - German Traffic Sign Recognition Benchmark

---

4. Base Paper / Base Available Model / Base of the Project

The project integrates several foundational Computer Vision concepts:

4.1. Driver Drowsiness Detection

- **Base Algorithm:** Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR).
- **Core Paper: "Real-Time Eye Blink Detection using Facial Landmarks"** by Soukupová and Čech, 2016. This paper established the EAR formula, which forms the mathematical basis for real-time micro-sleep detection.Real-Time Eye Blink Detection using Facial Landmarks
- **Base Code:** The project's current implementation (Facerecognition.py) is based on open-source Dlib and OpenCV implementations of the EAR/MAR algorithm.

The Dlib **shape_predictor_68_face_landmarks.dat** file is the essential pre-trained model for facial landmark extraction.

4.2. Road Side Signal Assistant

- **Base Architecture:** A light-weight Convolutional Neural Network (CNN) architecture, such as **MobileNet** or **EfficientNet**, adapted for **TensorFlow Lite**.
- Traffic Signs Classification
- **Purpose:** To ensure high frames-per-second (FPS) processing speed suitable for real-time embedding on the Raspberry Pi 5.

---

5. Novelty

This project goes beyond a standard single-feature drowsiness detection system by integrating three key novelties:

1. **Multi-Modal Drowsiness Detection with Behavioral Analysis:**
   - The current implementation combines three crucial metrics: **Eye Aspect Ratio (EAR)** for micro-sleep detection, **Mouth Aspect Ratio (MAR)** for yawning, and **Head Nod Detection** (implemented by measuring vertical distance changes of the nose tip/facial center in Facerecognition.py). This combined approach significantly improves reliability and reduces false positives/negatives compared to EAR alone.
2. **Edge Computing for Parallel Real-Time ADAS:**
   - The system is specifically designed for the **Raspberry Pi 5**, utilizing its enhanced processing power for a **dual-camera, parallel processing pipeline**. This allows the simultaneous execution of two complex tasks (Driver Monitoring + Forward View Perception) without performance degradation, which is a major challenge for edge devices.
3. **Integrated Dashboard for Cognitive Fusion and Decision Support:**
   - The system introduces a centralized dashboard that acts as a cognitive fusion center. It goes beyond displaying isolated data streams by synthesizing real-time outputs from both the driver drowsiness system (EAR, MAR, Head Nod) and the forward-view sign detection system. This integrated display provides the driver with a holistic view of their state and the immediate road environment, presenting clear, context-aware decisions.

---

6. Methodology

The project follows a phased development and integration roadmap:

Phase 1: Hardware and Software Setup(Completed/Progress)

- **Setup:** Configure the Raspberry Pi 5 with Raspberry Pi OS and install core Python libraries (OpenCV, Dlib, imutils, pygame, TensorFlow Lite dependencies).
- **Peripherals:** Connect and validate both USB webcams and the speaker system for input/output testing.

Phase 2: Drowsiness Detection Development(Completed)

- **Implementation:** Implement the eye_aspect_ratio and mouth_aspect_ratio functions using Dlib facial landmarks.
- **Behavioral Logic:** Develop and fine-tune the combined logic for EAR, MAR, and Head Nod counting to accurately trigger the alarm (Facerecognition.py).
- **Testing:** Test the **alert system** (playing alarm.wav using pygame.mixer) and visually confirm real-time detection performance on the Pi.

Phase 3: Road Sign and Behavioural Module Development(Progress)

- **Model Training:** Collect and preprocess traffic sign data (GTSRB/ITSD). Train a MobileNet classification model using TensorFlow and convert it to a **TensorFlow Lite model** for edge inference.
- **Forward View Perception:** Integrate the Road View camera feed and run the TFLite model to detect and classify road signs (e.g., Speed Limit, Sharp Curve Ahead).
- **Dynamic Data Integration:** Implement algorithms to detect "rash driving" (uncontrolled steering inputs) based on estimated motion or connected IMU/vehicle data.

Phase 4: Parallel Processing and Final Integration

- **Concurrency:** Design a robust multi-threading or multi-processing architecture in Python to run the Driver Monitoring (DMS) and Road Perception (Forward View) streams simultaneously on the Raspberry Pi 5.
- **Dashboard:** Develop the final Pygame-based dashboard to display dual-camera feeds, visual warnings, and system status information.
- **Safety Output Logic:** Finalize the logic that determines when the "Stop Car" safety protocol is triggered, combining inputs from Drowsiness, Road Sign compliance, and Rash Driving detection.

---

7. Progress Till Now

We have made significant progress in establishing the foundational Computer Vision pipeline, specifically for the internal driver monitoring functions.

- **Core Drowsiness Detection is Operational:** The primary code (Facerecognition.py) is complete and runs the real-time Drowsiness Detection logic, encompassing:
    - Real-time facial and 68-point landmark detection using Dlib.
    - Calculation of **Eye Aspect Ratio (EAR)** for blink/micro-sleep detection.
    - Calculation of **Mouth Aspect Ratio (MAR)** for yawning detection.
    - Implementation of **Head Nod detection** logic.
- **Alarm System Integration:** The system successfully uses pygame.mixer to play the **alarm.wav** file continuously when the combined drowsiness criteria are met.
- **Initial Setup Complete:** The foundational libraries (OpenCV, Dlib, Pygame) and the core Python environment on a host machine (simulating the Pi environment) have been successfully configured and tested.

Next Intermediate Steps

The focus shifts to integrating the external perception and control features:

1. Benchmarking and optimizing the current Drowsiness module performance on the Raspberry Pi 5.
2. A supervised learning technique called online learning will be used to make the model up to date with respect to the dataset.
3. Beginning **Phase 3** development: Collecting, training, and converting the first **TensorFlow Lite model** for Road Sign Recognition.
4. The two separate models of drowsiness detection and road sign detection will be integrated into one final model using multithreading which allows the final model to perform much more efficiently and accurately.
