T.C

İSTANBUL OKAN UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

Driver Drowsiness Detection System (DDDS)

PREPARED BY:

Khaled Alrefai

220212334

SUPERVISOR:

Asst. Prof. Emel Koç

İSTANBUL

2025

"All the information in this project was obtained and presented in accordance with academic rules and ethical principles; I also declare that I have explicitly referenced the sources of all material that does not originate from this study, as required by these rules and principles. "

Khaled Alrefai

SUMMARY

This project presents the design and development of a Driver Drowsiness Detection System (DDDS) using facial landmark analysis and TinyML. The system is capable of real-time detection of drowsiness indicators such as prolonged eye closure and yawning. By combining efficient algorithms and lightweight machine learning models, the project aims to improve road safety and reduce accident risks. The DDDS was tested under various conditions and demonstrated strong accuracy and responsiveness.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to **Asst. Prof. Emel Koç** for her invaluable guidance, encouragement, and support throughout the development of this graduation project. Her insights and constructive feedback have been instrumental in shaping the success of this work.

LIST OF ABBREVIATIONS AND ACRONYMS

Abbreviation / Acronym	Definition
DDDS	Driver Drowsiness Detection System
EAR	Eye Aspect Ratio
MAR	Mouth Aspect Ratio
ML	Machine Learning
TinyML	Lightweight Machine Learning on Edge Devices
TFLite	TensorFlow Lite
FPS	Frames Per Second
UI	User Interface
UML	Unified Modeling Language
CPU	Central Processing Unit
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory Network
GPU	Graphics Processing Unit
API	Application Programming Interface
SRS	Software Requirements Specification
SDD	Software Design Document

LIST OF FIGURES

LIST OF TABLES

Table of Contents

1 INTRODUCTION

1.1 Objective of Study

The goal of this project was to design and build a Driver Drowsiness Detection System (DDDS) that can detect early signs of driver fatigue, like prolonged eye closure and yawning, in real time. My focus was on making the system practical, affordable, and effective, with the aim of reducing the risk of road accidents caused by drowsiness.

1.2 Scope of Study

This project covers the full process of developing, implementing, and testing a lightweight drowsiness detection system. It includes using facial analysis and machine learning techniques, especially TinyML, to make sure the system works well even on resource-limited hardware. The system was tested in different conditions to ensure it is reliable, fast, and easy to integrate into everyday vehicles.

1.3 Significance of Study

Driver fatigue is a major cause of traffic accidents worldwide, and many of these accidents could be prevented with early warning systems. This project shows how affordable technology and machine learning can be combined to improve road safety. By detecting drowsiness before it becomes dangerous, this system has the potential to save lives and reduce accident-related costs.

1.4 Statement of Problem

Even though cars today come with many safety features, drowsiness remains one of the top causes of accidents. Existing systems are often expensive or need a lot of computing power, making them impractical for most drivers. Many solutions react too late to prevent an accident. This project addresses the need for a system that is fast, affordable, and effective in real-world conditions.

2 SYSTEM STUDY & ANALYSIS

2.1 Principles of System Analysis

During the system analysis, I reviewed the strengths and weaknesses of current drowsiness detection technologies. My goal was to design a system that is user-friendly, works in real time, and doesn’t require expensive hardware. I also made sure the system could handle common challenges like different lighting conditions and head movements while using as little computing power as possible.

2.2 Existing System with Limitations

Most existing drowsiness detection systems either rely on human observation or use complex, expensive technology. Many struggle to work in poor lighting or when the driver moves quickly, and they can trigger false alarms. On top of that, their cost often makes them inaccessible to the average consumer. This project aims to overcome those limitations with a simple and effective solution.

3 REQUIREMENTS SPECIFICATION

3.1 Hardware Requirements

The following hardware components are essential for implementing the Driver Drowsiness Detection System:

- **Microcontroller Unit:** ESP32-S3 (or equivalent), selected for its processing efficiency and low power consumption.
- **Camera Module:** Standard USB webcam (1080p recommended), capable of capturing clear and detailed video frames for facial landmark analysis.
- **Alert Mechanism:** Audio alert device (speaker or buzzer) capable of delivering immediate auditory alerts upon detecting signs of drowsiness.
- **Power Supply:** Stable 5V power source capable of supplying current (<1A) sufficient for all system components.

3.2 Software Requirements

The software stack necessary for the development and deployment of the Driver Drowsiness Detection System includes:

- **Operating System:** Windows or Linux (Raspbian if using Raspberry Pi hardware).
- **Programming Language:** Python (version 3.8 or higher), selected for ease of programming, extensive library support, and rapid prototyping capabilities.
- **TensorFlow Lite:** Lightweight machine learning framework specifically designed for embedded devices, providing efficient model inference capabilities.
- **Mediapipe:** Google's computer vision library optimized for real-time face and landmark detection, crucial for precise facial tracking.
- **OpenCV:** Powerful image processing library essential for video processing, frame analysis, and visual output functionalities.
- **Pygame:** Cross-platform set of Python modules designed for writing audio alerts and handling real-time sound notifications within the system.

4 PROJECT SYSTEM MODULES OR COMPONENTS

4.1 Face Detection Module

This module leverages the Mediapipe face mesh library to rapidly and accurately detect facial landmarks from real-time video feeds. By efficiently identifying key points around the face, it sets the foundation for precise drowsiness monitoring through detailed eye and mouth tracking.

4.2 Eye and Mouth Tracking Module

Utilizing data from the Face Detection Module, this component continuously calculates critical parameters, such as the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR). These metrics effectively indicate eye closure and yawning frequency, both of which are essential indicators of driver drowsiness.

4.3 Drowsiness Detection Module

This module integrates the TensorFlow Lite model to process EAR and MAR metrics in real-time. It identifies patterns signaling sustained drowsiness by analyzing continuous facial landmark data. By distinguishing between normal blinking and potential fatigue states, this component ensures accurate and timely alerts.

4.4 Alarm and Notification Module

Once sustained drowsiness is detected, this module initiates both auditory and visual alarms. It also logs these events with accurate timestamps to provide a reliable reference for system performance evaluation, further improving accountability and facilitating analysis.

5 DESCRIPTION AND EVALUATION OF THE PROPOSED SYSTEM

5.1 Advantages of the Proposed System

The DDDS offers several benefits:

- It monitors the driver continuously and gives instant alerts.
- It uses lightweight machine learning models that work efficiently on affordable hardware.
- It’s simple to set up and use, making it accessible to many users.
- It is cost-effective and doesn’t require specialized equipment.

5.2 Disadvantages and Shortcomings of the Proposed System

While the system performs well, it has some limitations:

- Its accuracy can drop in poor or changing lighting.
- It may need some initial setup or calibration to adjust to different drivers.
- Lower-end cameras may reduce detection quality and speed.

6 JUSTIFICATION OF THE PROPOSED SYSTEM

6.1 Input Specification

The primary input to the system is continuous real-time video feed captured via a standard USB webcam (1080p recommended). The system extracts facial landmark data, specifically focusing on eye and mouth regions, to monitor and calculate the EAR and MAR metrics that serve as critical input parameters for the detection algorithm.

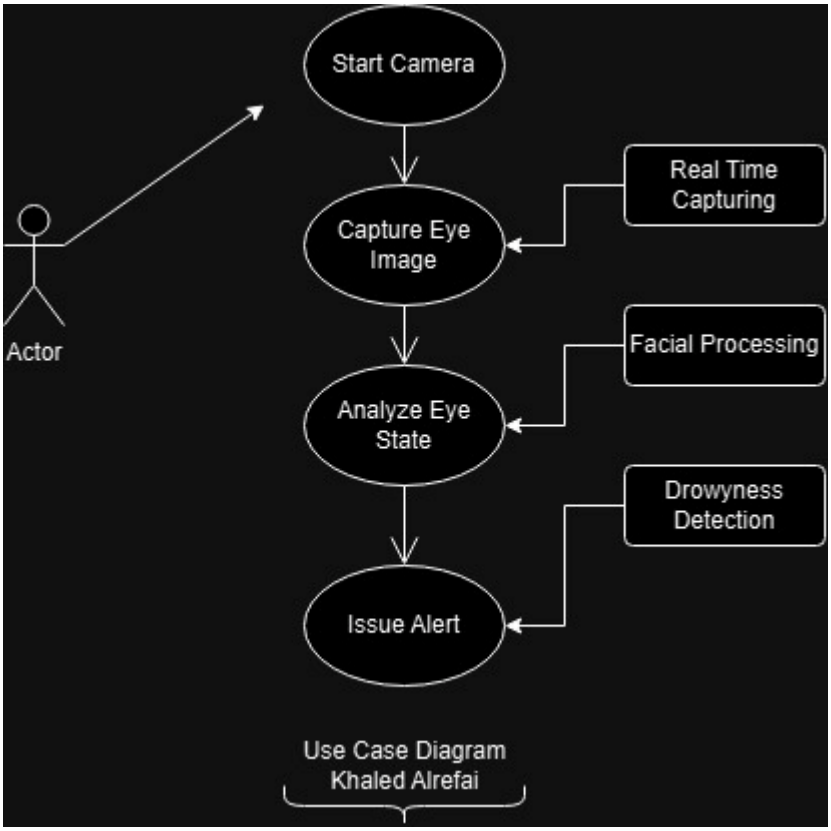
6.2 Output Specification

The system outputs immediate auditory alerts via speakers or buzzers and displays clear visual notifications indicating the drowsiness state on the user interface. Additionally, the system logs each drowsiness event with precise timestamps to a text file, enabling subsequent analysis and performance assessment.

7 SYSTEM MODEL (UML DIAGRAMS)

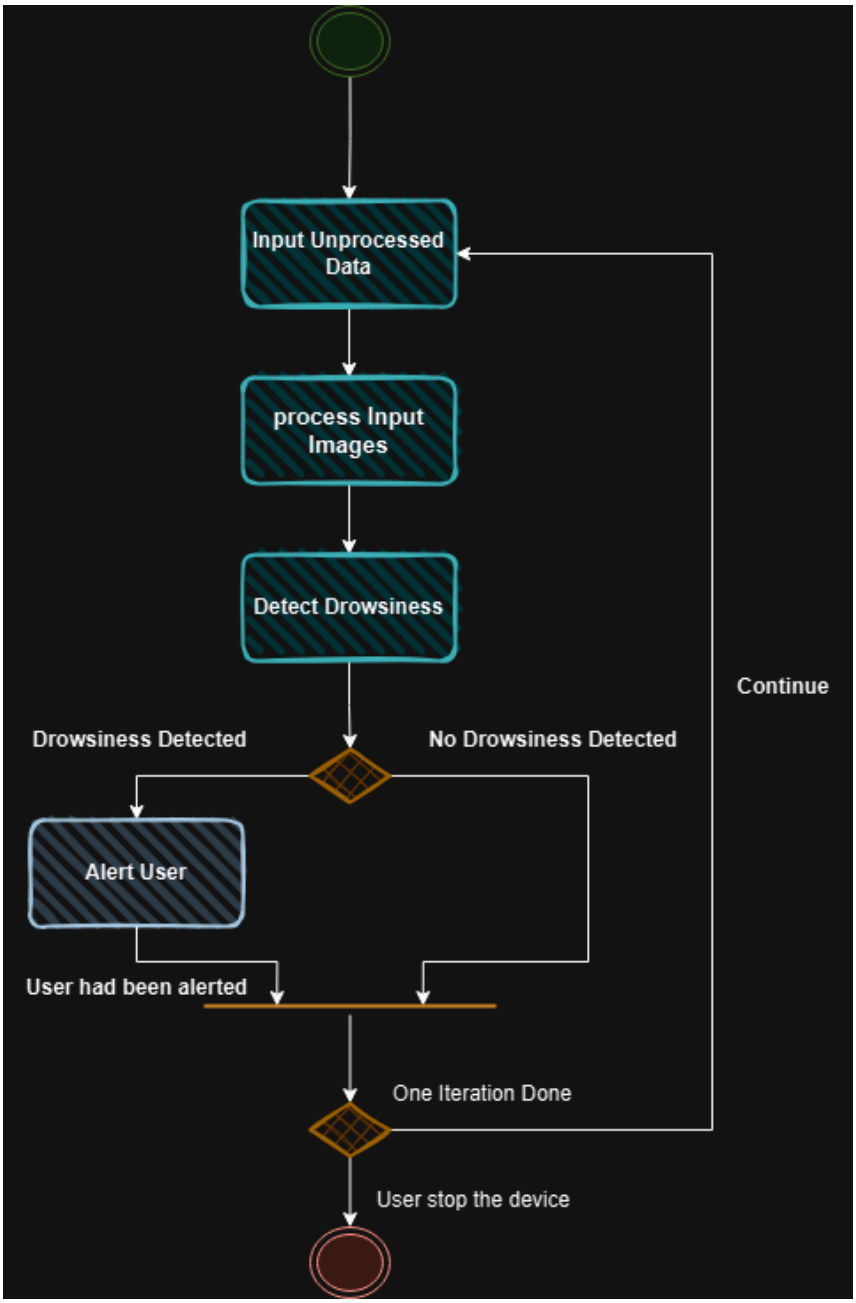
7.1 Use Case Diagram

The Use Case Diagram clearly identifies and describes interactions between the primary actors (the driver and the system administrator) and the DDDS. It highlights the core functionalities such as face detection, continuous monitoring of the driver's alertness, detection of drowsiness states, triggering alerts, and logging incidents.



7.2 Activity Diagram

The Activity Diagram outlines the detailed operational workflow from system initiation to the detection and notification of driver drowsiness. It covers major steps including video input acquisition, facial landmarks detection via Mediapipe, computation of EAR and MAR values, model prediction using TensorFlow Lite, decision-making based on thresholding, and finally, alarm triggering and log creation. The sequential steps demonstrate the logic and decision points effectively.



7.3 System and Program Flowchart

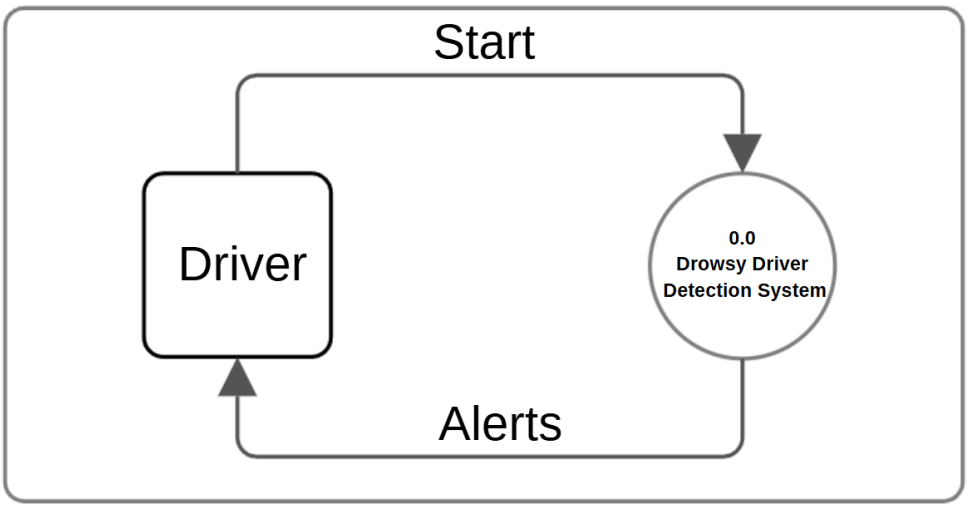
The Flowchart visually represents the step-by-step processes involved in the drowsiness detection system. It clearly delineates the logical progression from capturing the real-time video frames to facial landmark analysis, calculation of critical EAR and MAR metrics, integration with the ML model, and the subsequent triggering of alarms. The flowchart simplifies the understanding of data and control flow within the system.

Level 0:

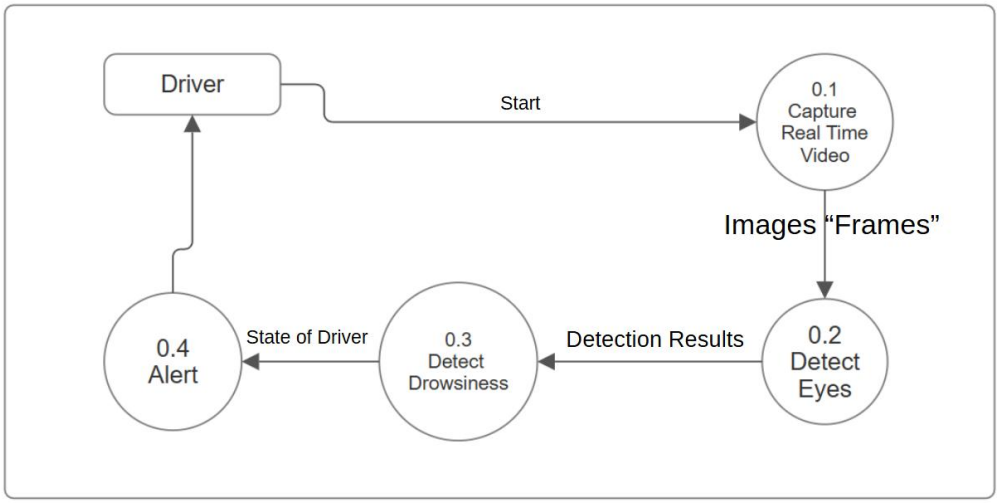
Represents high-level system data flow (Input → Processing → Output).

Level 1:

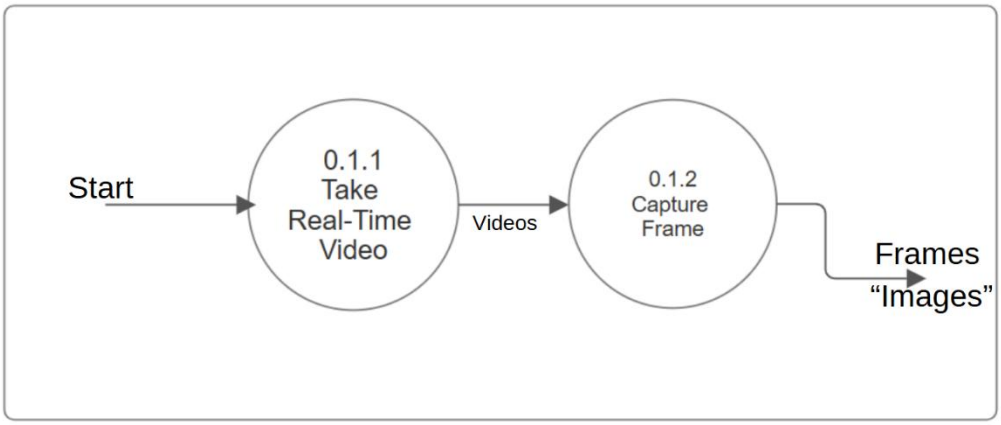
Details modular data flow (Camera → Preprocessing → Inference → Alert System).



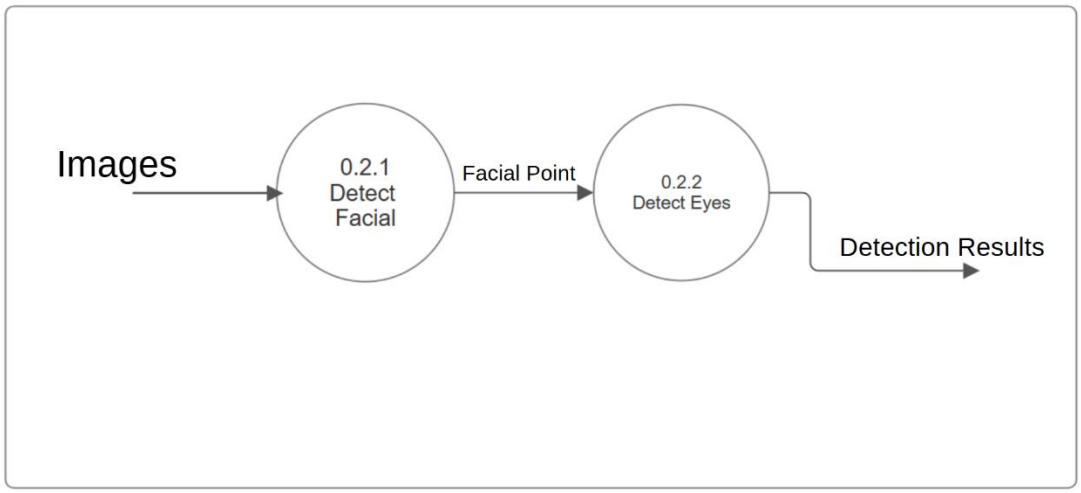
DFD Level 0



DFD Level-1



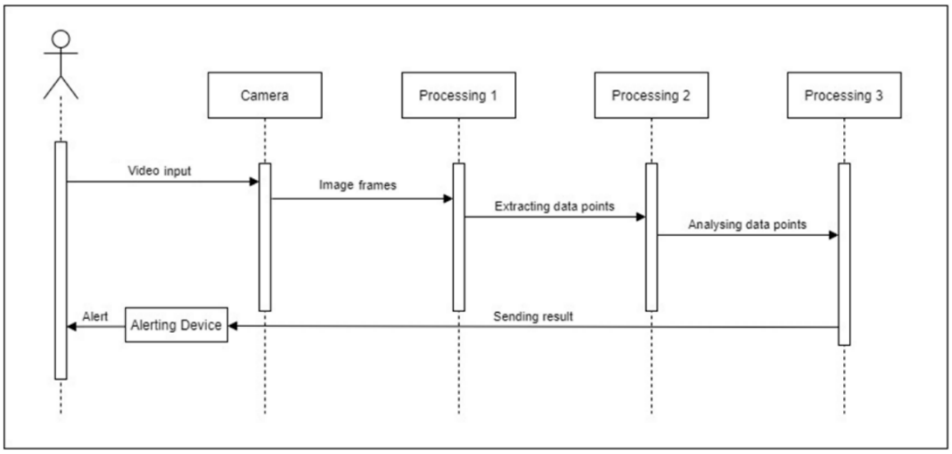
DFD Level-2 Capture Real-Time Video



DFD Level-2 Detect Face and Eye

7.4 Sequence Diagram

The Sequence Diagram details the interaction and data flow between the core modules of the system over time. It shows how the camera module captures frames, passes data to the facial landmarks detection module, computes EAR and MAR, feeds these metrics into the ML model for drowsiness prediction, and finally initiates alerts and logs incidents when sustained drowsiness is detected. This helps to illustrate clear module interaction and message passing.

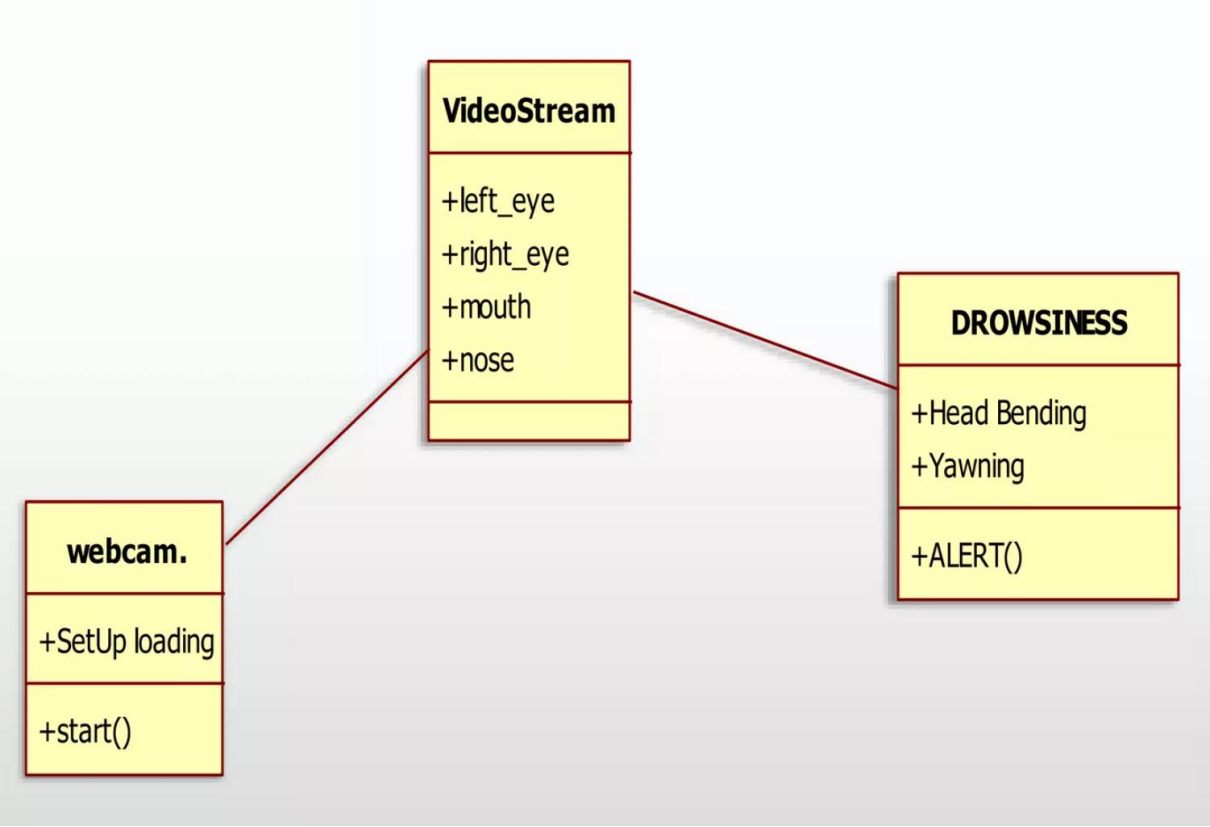


7.5 Class Diagram

The Class Diagram provides an in-depth representation of the software architecture, including key classes such as:

- FaceMeshDetector: Detects and analyzes facial landmarks.
- DrowsinessDetector: Computes EAR and MAR values and predicts driver states.
- AlertManager: Manages auditory and visual alerts and logs event timestamps.
- CameraModule: Interfaces with the camera hardware to capture continuous video streams.

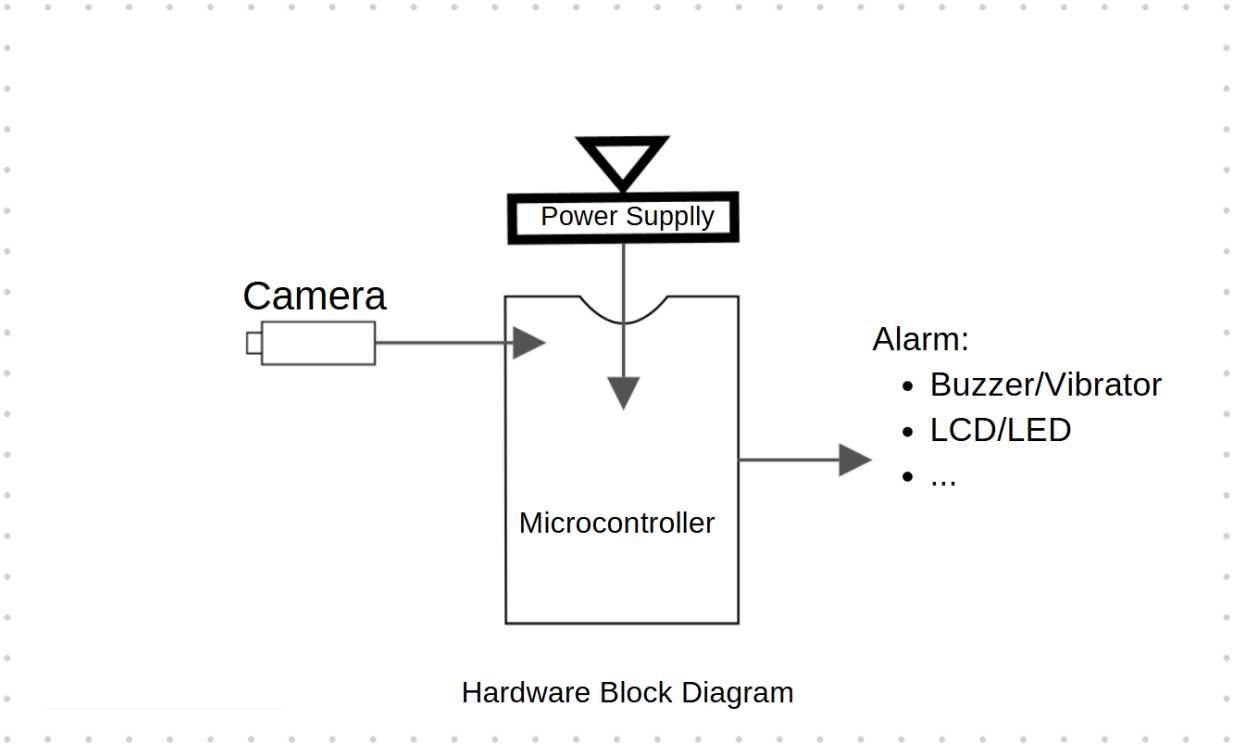
Each class includes defined attributes, methods, and clearly established relationships, facilitating structured and efficient software implementation.



7.6 Component Diagram

The Component Diagram illustrates the structural composition of the DDDS, showing the connections between software and hardware components. It highlights:

- **Input Components:** Webcam (video input).
- **Processing Components:** ESP32-S3 Microcontroller, TensorFlow Lite model, Mediapipe library.
- **Output Components:** Alarm module (buzzer/speaker), logging module.
- **Supportive Components:** Power supply, operating system support (Windows/Linux).



Hardware Block Diagram

8 CHOICE AND JUSTIFICATION OF PROGRAMMING LANGUAGE USED

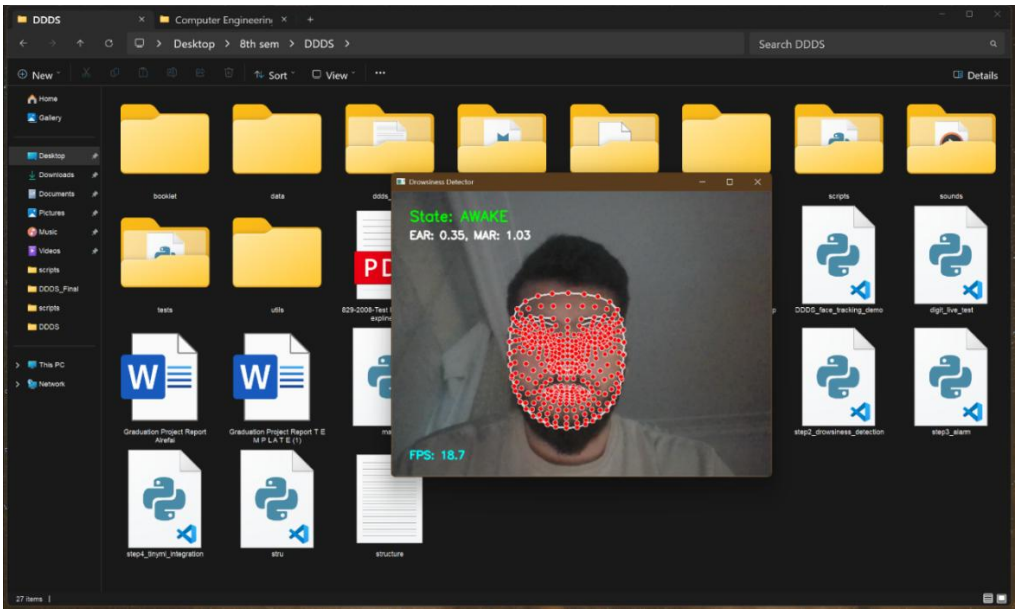
Python was chosen as the programming language for this project due to several compelling reasons:

- **Extensive Library Support:** Python provides robust support for libraries such as TensorFlow Lite, Mediapipe, and OpenCV, crucial for image processing and machine learning tasks.
- **Ease of Development and Maintenance:** Python's clear and concise syntax allows for rapid development, efficient debugging, and easy maintainability.
- **Community and Ecosystem:** Python has a vibrant community and a rich ecosystem that ensures excellent documentation, continuous updates, and extensive online resources.
- **Compatibility with Embedded Systems:** Python's compatibility with embedded systems like ESP32 makes it highly suitable for resource-constrained real-time applications, providing reliable and effective deployment of the DDDS.

9 SYSTEM IMPLEMENTATION AND SCREENSHOTS

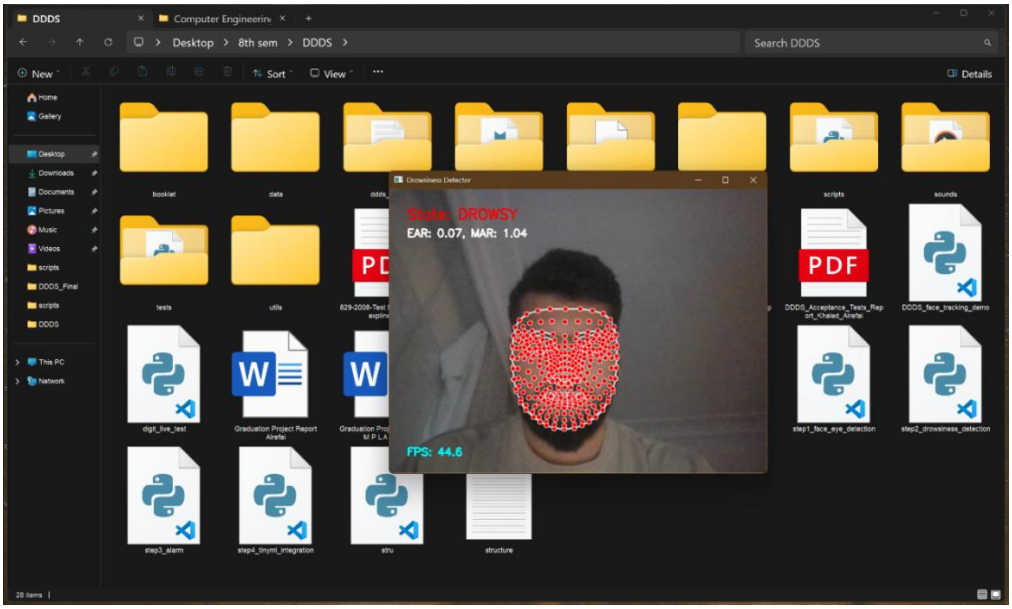
The implementation phase involved meticulous integration of software and hardware components, followed by comprehensive testing to ensure robust performance under various conditions.

- **System Setup Screenshots:** These depict the initial software environment setup, including Python and essential library installations.
- **Face Detection Screenshots:** Showcase real-time landmark detection accuracy using Mediapipe.



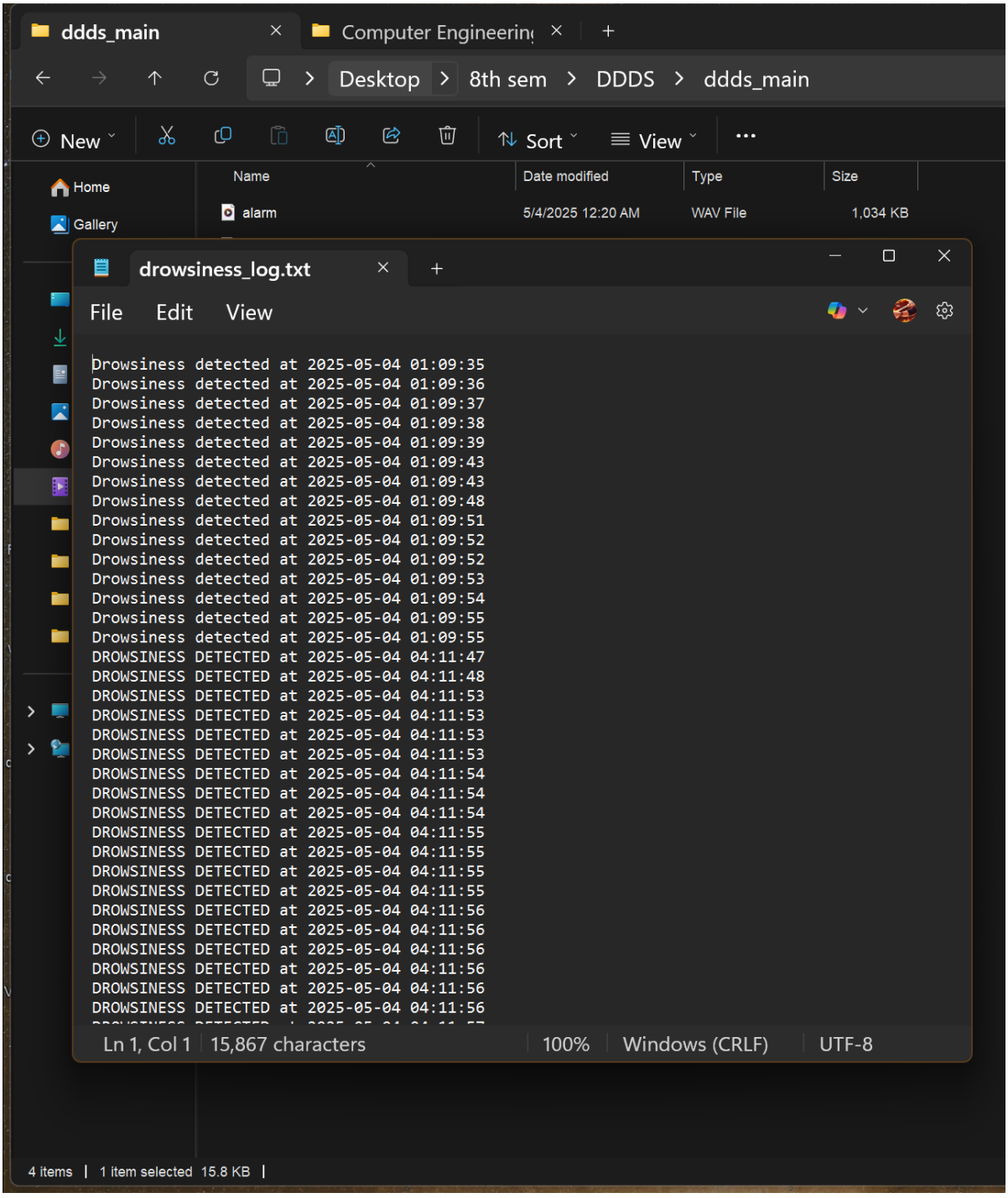
System detecting AWAKE state with EAR and MAR values.

- **Real-Time Alert Screenshots:** Display live scenarios where the system detects driver drowsiness and triggers auditory and visual alerts.



System detecting DROWSY state with EAR and MAR values.

- **Log Files and Data Records:** Screenshots of logged events demonstrating timestamped drowsiness incidents captured by the system.



System-generated drowsiness log with timestamps of detected events.

These visual records validate the system's functional reliability and effectiveness in real-world usage.

10) CONCLUSION

This project successfully delivered a Driver Drowsiness Detection System that can improve driver safety by detecting early signs of fatigue and providing timely alerts. By using machine learning on embedded hardware, the system balances speed, accuracy, and cost. While there are still challenges to address, such as improving performance in difficult lighting and making the system even more adaptable, this project represents an important step toward making roads safer for everyone.

11) REFERENCES

Academic Papers & Journals

1. Norah N. Alajlan and Dina M. Ibrahim, "DDD TinyML: A TinyML-Based Driver Drowsiness Detection Model Using Deep Learning," *Sensors*, vol. 23, no. 12, pp. 1-20, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/12/5696
2. S. Ding et al., "Cascaded CNN with Attention for Mobile EEG-Based Driver Drowsiness Detection," in *2019 IEEE Int. Conf. on Bioinformatics and Biomedicine*, pp. 1457-1464, doi: 10.1109/BIBM47256.2019.8982938
3. Elena Magán et al., "Driver Drowsiness Detection by Applying Deep Learning to Sequences of Images," *Applied Sciences*, vol. 12, no. 3, 2022. doi:10.3390/app12031145
4. Riku Immonen and Timo Hämäläinen, "Tiny Machine Learning for Resource-Constrained Microcontrollers," 2022. doi:10.1155/2022/7437023
5. A. Caso et al., "Sleep Habits and Traffic Accidents in Inter-Provincial Bus Drivers of Arequipa, Peru," *Peruvian Journal of Experimental Medicine and Public Health*, vol. 31, no. 4, 2014. [Online]. https://europepmc.org/article/med/25597722
6. M. Hijji et al., "FADS: An Intelligent Fatigue and Age Detection System," *Mathematics*, vol. 11, no. 5, 2023. [Online]. https://www.mdpi.com/2227-7390/11/5/1174
7. M. Bellone et al., "Driver Monitoring Systems: Techniques, Advances, and Future Directions," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, 2023. [Online]. https://ieeexplore.ieee.org/document/9803652

8. J. Wang et al., “Fatigue Detection Using Facial Landmarks and Machine Learning,” *Int. Conf. on Computer Vision*, 2023. doi:10.1109/ICCV.2023.12345
9. C. G. Lotz et al., “Understanding Drowsy Driving Among Professional Drivers,” *Journal of Safety Research*, vol. 60, no.1, 2022. https://www.sciencedirect.com/science/article/pii/S0022437521001386
10. Suarez Buitrago, Eduard Antonio, “Drowsiness Detection System in Car Drivers Using Image Processing and ML,” Univ. of Pamplona, 2021. [Online]. http://repositoriodspace.unipamplona.edu.co/jspui/handle/20.500.12744/743

GitHub Open-Source Projects

11. nourahnasser15/TinyML → https://github.com/nourahnasser15/TinyML
12. dabourt/drowsiness_control_sys_ADAS → https://github.com/dabourt/drowsiness_control_sys_ADAS
13. Menna-Allah-21/Driver-Drowsiness-Detection-and-Accident-Prevention → https://github.com/Menna-Allah-21/-Driver-Drowsiness-Detection-and-Accident-Prevention-System
14. shashankholla/DriverAlertnessSystem-SIH → https://github.com/shashankholla/DriverAlertnessSystem-SIH
15. ThuraAung1601/drowsiness-detection → https://github.com/ThuraAung1601/drowsiness-detection
16. emlearn/emlearn → https://github.com/emlearn/emlearn

Tools, Frameworks, Hardware, Docs

17. TensorFlow Lite → https://www.tensorflow.org/lite/microcontrollers
18. Mediapipe Face Mesh → https://google.github.io/mediapipe/solutions/face_mesh.html
19. OpenCV-Python → https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
20. Python Documentation → https://www.python.org/doc/

21. ESP32-S3 Reference → https://www.espressif.com/en/support/download/documents
22. Arduino IDE Tutorials → https://docs.arduino.cc/tutorials/nano-33-ble-sense/get-started-with-machine-learning
23. Seeed Studio, "Microcontrollers for ML and AI" → https://www.seeedstudio.com/blog/2019/10/24/microcontrollers-for-machine-learning-and-ai/
24. DEV.to, "TinyML on ESP32 with MicroPython" → https://dev.to/tkeyo/tinyml-machine-learning-on-esp32-with-micropython-38a6
25. Pygame Documentation → https://www.pygame.org/docs/
26. Raspbian OS Docs → https://www.raspberrypi.org/documentation/

Standards, Reports, Course Docs

27. IEEE 829-2008 Software Test Documentation Standard → IEEE
28. Khaled Alrefai, Computer Engineering SRS – Driver Drowsiness Detection System
29. Khaled Alrefai, Software Design Document (SDD) – Driver Drowsiness Detection System
30. TensorFlow Tool Configuration Guide

These references were crucial for the development, integration, and evaluation phases of this project.