Computer *Engineering*
Software Requirements Specification (SRS) Document

Drowsy Driver Detection System

Nov 03, 2024

Version 0.1

By: Khaled Alrefai

Supervisor: Prof. Emel Koç

Contents

Revisions:

Version	Primary Author(s)	comment	Date Completed
0.1	**Khaled Alrefai**	**initial draft.**	**Nov 03 2024**

Revision History Note:

I will visit Prof. Emel Koç on November 6, 2024 this week to discuss the initial draft and receive feedback for further improvements.

1. Problem Statement

Driving while drowsy is a critical road safety issue that leads to numerous accidents and fatalities each year. Factors such as sleep deprivation, long hours behind the wheel, and sedative effects of medications compromise driver alertness, putting lives at risk. Existing detection systems often rely on external connectivity or invasive data collection, making them impractical for widespread use. There is a need for an efficient, self-contained solution that preserves user privacy while offering reliable drowsiness detection.

2. Overview

2.1 Background

The increasing frequency of fatigue-related driving incidents has brought attention to the dangers posed by drowsy drivers. Studies have shown that fatigue can impair reaction times, reduce attentiveness, and affect decision-making capabilities, similar to driving under the influence of alcohol. Despite advances in automotive safety, the market lacks an accessible, private, and energy-efficient solution to monitor and alert drivers about drowsiness. Leveraging advancements in TinyML and microcontroller technology, we have an opportunity to develop a novel approach that addresses these challenges.

2.2 Overall Description

The Driver Drowsiness Detection System (DDDS) is designed as a compact, energy-efficient device that operates without internet connectivity. The system uses a camera to monitor the driver's eye activity, analyzing this data through TinyML models embedded on a microcontroller. Upon detecting signs of drowsiness, such as prolonged eye closure, the device triggers alerts (e.g., sound or light) to refocus the driver's attention. By processing all data locally, the DDDS ensures user privacy while maintaining high performance. The design emphasizes adaptability, allowing the system to function in various environments, from personal vehicles to commercial fleets.

3. Investigation & Analysis Methodology

3.1 System Investigation

To develop an effective Driver Drowsiness Detection System (DDDS), a comprehensive investigation of current technologies, existing solutions, and potential constraints is necessary. This involves researching and analyzing the effectiveness of existing drowsiness detection mechanisms, identifying gaps in these systems, and understanding the limitations imposed by hardware capabilities, energy consumption, and privacy concerns. We will explore the use of TinyML models on microcontrollers and test different configurations to achieve optimal performance. This investigation will also involve studying user safety standards and regulations that impact the design and functionality of the system.

3.2 Analysis Methodology

The analysis methodology for the DDDS project involves a systematic approach to ensure that the solution is both technically feasible and capable of meeting user needs efficiently.

3.2.1 Feasibility Study and Requirements Elicitation

A feasibility study will be conducted to evaluate the practicality of implementing a drowsiness detection system using TinyML on resource-constrained microcontrollers. This will include assessing the capabilities of available hardware, such as cameras and microcontrollers, to process data in real-time while remaining energy efficient. Requirements will be gathered through a combination of research, expert consultations, and a review of existing drowsiness detection technologies.

3.2.2 System Analysis and Requirements Specification

3.2.2.1 Problem Analysis

The problem will be analyzed in detail to understand how drowsiness impacts driver safety and to determine the specific conditions the system must detect (e.g., eye closure duration). This analysis will also include evaluating various machine learning algorithms and their suitability for deployment on microcontrollers.

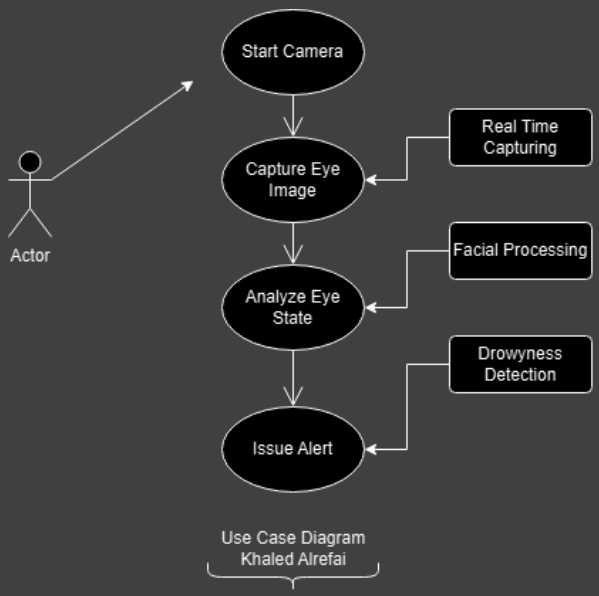
3.2.2.2 Scope and Limitations

The scope of the project includes developing a device capable of real-time drowsiness detection that operates entirely offline. Limitations may include processing power constraints of microcontrollers, variability in lighting conditions, and ensuring the device remains compact and energy-efficient.

3.2.3 Object-Oriented Design Using UML

The system will be modeled using UML (Unified Modeling Language) to ensure a clear and structured design. This includes:

- Use Case Diagrams: Depicting the interactions between the driver (user) and the system components (e.g., camera, alert mechanisms).
- Sequence Diagrams: Illustrating the data flow from capturing images to issuing alerts.



3.2.4 Prototyping

A prototype of the DDDS will be developed to validate the design and ensure the system meets performance expectations. This prototype will be tested in controlled environments to simulate various driving conditions and drowsiness scenarios.

4. Constraints

4.1 Scalability

The system must be designed to handle real-time processing with a minimal footprint, but scalability in terms of hardware integration (e.g., different types of microcontrollers or sensors) should be considered for future enhancements.

4.2 Data and Function Mapping

Efficient data and function mapping will be crucial to ensure that image data is processed quickly and accurately within the limited resources of the microcontroller.

4.3 Proprietary Hardware and Software

The project may be constrained by the availability of proprietary components, such as specialized camera modules or specific microcontroller models. Where possible, open-source alternatives will be explored.

4.4 Real-Time Updates

The system must prioritize real-time processing to detect and alert the driver immediately. Batch processing is not an option due to the critical nature of drowsiness detection.

4.5 Project Schedule

The project will be developed within a structured timeline to ensure key milestones, such as prototyping, testing, and final deployment, are met. The six-month timeframe will be used to implement and refine the system, allowing for thorough testing and iterations.

5. Operational Requirements

5.1 User Assistance and Help Features

The DDDS should include a user-friendly setup process with clear instructions. In case of troubleshooting, a basic guide will be provided to assist users with common issues, such as configuring the device or understanding alert signals.

5.2 Technical Support and Maintenance

The device should be designed for easy maintenance, with minimal technical intervention required. In the event of technical failures, instructions for basic maintenance or component replacement will be made available, although the design will minimize the need for frequent support.

5.3 Device Configuration and Customization

The system will feature configuration options that allow drivers to customize alert settings, such as sound volume or light intensity. These features will be accessible through a simple interface, either on the device itself or via a connected app, if applicable.

5.4 System Interface Independence

The DDDS will operate as a standalone device, independent of any external systems or network connectivity. It will not require an interface with external databases or servers, ensuring consistent functionality in offline environments.

5.5 Hardware Reliability and Data Protection

The device will be designed with reliable hardware components that minimize the risk of failure. Routine checks and self-diagnostics will be integrated to ensure consistent performance. Since the system operates without data collection, there will be no need for data backups, but power management will be optimized to prevent failures.

5.6 Event Logging and System Diagnostics

An internal audit trail will be maintained for diagnostic purposes, logging system events such as detection alerts and system errors. This information will be used to improve performance and troubleshoot issues if they arise.

6. Functional Requirements

6.1 Drowsiness Detection and Alert System

The core functionality of the DDDS involves detecting signs of driver drowsiness and issuing alerts. Each requirement outlines a key component of this system.

6.1.1 Eye Image Capture

The system must continuously capture images of the driver's eyes using a built-in camera, ensuring minimal latency in data acquisition.

6.1.2 Eye State Analysis

Captured images will be analyzed in real time using TinyML algorithms to determine if the driver's eyes are closed for an extended period, indicating drowsiness.

6.1.3 Alert Mechanisms

Upon detecting signs of drowsiness, the system will trigger an alert through sound, light, or both. The alert settings can be configured by the user to suit their preferences and needs.

7. Input Requirements

7.1 Environmental and Usage Conditions

The system will accept and process environmental inputs, such as lighting conditions and driver positioning, to optimize detection accuracy. This will be managed through internal calibration mechanisms.

7.2 Configuration Inputs

Users will be able to set configuration options, such as alert volume, light intensity, and detection sensitivity, through a simple user interface. These inputs will ensure the device operates according to the driver's needs.

8. Process Requirements

8.1 Data Processing Flow

The DDDS must efficiently process data captured by the camera to ensure real-time detection and alert generation. The data flow includes capturing images, running machine learning algorithms to analyze eye states, and triggering alerts if drowsiness is detected.

1. **Image Acquisition**: The camera continuously captures images of the driver's eyes.
2. **Data Preprocessing**: Basic preprocessing, such as image normalization and filtering, is applied to prepare the data for analysis.
3. **Eye State Analysis**: TinyML algorithms analyze the images to detect whether the driver's eyes are open or closed.
4. **Alert Generation**: If prolonged eye closure is detected, the system immediately issues an alert.

8.2 Performance Metrics

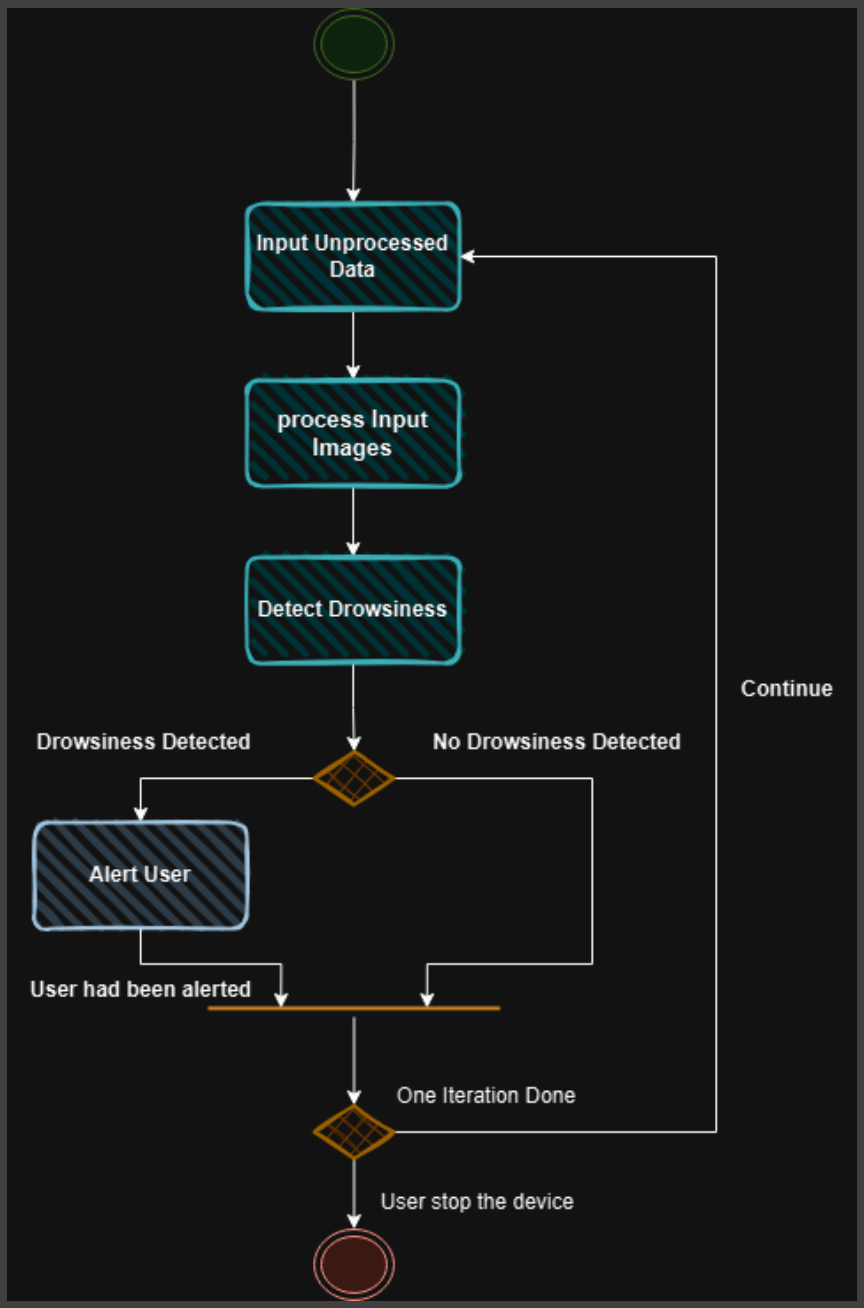
The system must meet specific performance requirements to ensure effectiveness:

- **Latency**: The detection and alert system should respond within few milliseconds of detecting drowsiness.
- **Accuracy**: The eye state analysis must achieve a detection accuracy of at least 85-95% under various lighting conditions.
- **Power Efficiency**: The device should operate continuously for some hours on a single charge.

8.3 Activity Diagram

The Activity Diagram illustrates the flow of actions performed by the DDDS, from image capture to alert generation. This visual representation will clarify how different system components work together in a sequence of operations.

- Start: The system is powered on and initializes components.
- Image Capture: The camera captures images at regular intervals.
- Preprocessing: The images are prepared for analysis.
- Drowsiness Detection: The ML model processes the images to determine if the driver is drowsy.
- If Drowsy: Proceed to alert generation.
- If Not Drowsy: Return to image capture.
- Alert Generation: An alert is issued based on the user's configured settings.
- End: The cycle repeats or stops if the system is powered off.



9. Constraints

9.1 Hardware Limitations

The system's performance is constrained by the processing power and memory of the microcontroller. TinyML models must be optimized to run efficiently within these limits.

9.2 Environmental Factors

Variations in lighting, driver position, and facial features may impact the accuracy of detection. The device must be calibrated to handle a range of conditions, but extreme cases may still pose challenges.

9.3 Privacy and Security

The system must operate without internet connectivity and without storing or transmitting personal data. This constraint ensures that user privacy is maintained at all times.

10. System Interfaces

10.1 User Interface (UI)

The device will feature a minimalistic user interface that allows for easy configuration of settings, such as alert type and volume. The UI may include simple buttons or a small display for feedback.

10.2 External Interfaces

The DDDS will interface with:

Power Supply: USB or car power adapter for charging.
Optional External Sensors: The system could support additional sensors, such as temperature or heart rate monitors, to enhance detection accuracy in future versions.

11. System Reliability and Maintenance

11.1 Reliability

The device must be designed to operate reliably over long periods. Key reliability features include:

Self-Diagnostics: The system will perform regular self-checks to ensure proper functioning.
Error Handling: If a component fails, the system will alert the user and attempt to reset.

11.2 Maintenance

The DDDS will require minimal maintenance. Users may need to occasionally clean the camera lens or update firmware, which can be done using simple, offline methods.

12. Deployment Requirements

12.1 Deployment Environment

The DDDS must be easy to deploy in various vehicles, from personal cars to commercial fleets. It should be designed for easy mounting and setup, requiring minimal installation effort.

12.2 User Training and Setup

A simple user guide will be provided to assist with installation and configuration. Training videos or quick-start guides may be available to ensure that users can set up the device effortlessly.