Driver Fatigue Detection And Warning Application

BOSCH CodeRace Challenge Round 2

Team Jolibee

July, 2025

Table of Contents

1 Introduction

1.1 Purpose

This software design document describes the architecture and system design of the Driver Fatigue Detection and Warning Application, a solution developed to enhance road safety by detecting early signs of driver fatigue using steering wheel angle (SWA) and yaw rate (YR)—two types of data already available in most modern vehicles. By delivering early and clear warnings when signs of fatigue are detected, the system encourages drivers to take timely actions—such as pulling over to rest—before accidents occur. This proactive strategy is intended to help prevent avoidable collisions, reduce property damage, lower insurance claims, and ultimately save lives.

1.2 Scope

This document describes the implementation details of the Driver Fatigue Detection and Warning Application (DFDWA). DFDWA will consist of five major components: Data Acquisition, Signal Processing, Fatigue Detection Algorithm and Warning System. The current implementation is designed specifically for cars. However, future enhancements will focus on expanding compatibility with a broader range of car models, including trucks, with the goal of providing tailored and robust safety solutions that address the specific fatigue-related risks associated with different vehicle types.

1.3 Definitions and Acronyms

Acronym	**Meaning**
DFDWA	Driver Fatigue Detection and Warning Application
YA	Yaw angle
SWA	Steering wheel angle

1.4 References

- Arduino Official Website: `https://www.arduino.cc`
- Real Python - Arduino with Python: `https://realpython.com/arduino-python/`

2 System Overview

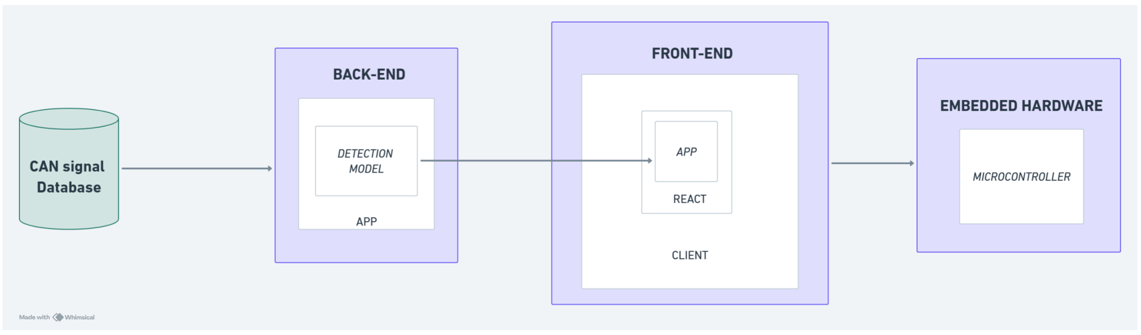


Figure 1

Figure 1 illustrates the architectural structure designed for the development of the CAN signal-based detection system.

The system consists of four main components: the CAN signal database, back-end processing, web-based frontend interface, and embedded hardware. The main objective is to analyze CAN data to detect anomalies or critical conditions in real time.

The backend component receives and processes data from the CAN signal database. It runs the detection model, analyzing CAN message patterns and behaviors to identify irregularities. When a critical event is detected, the system sends control commands to the hardware to activate warning devices such as a buzzer.

The frontend, built with React, runs in the user's browser and serves as the interface for visualizing real-time CAN data, detection results, and alerts. It communicates with the backend to display relevant indicators in a user-friendly manner.

The embedded hardware, equipped with a microcontroller, receives control commands from the software and triggers the warning system, such as activating a buzzer, to alert the driver.

The modular design of this system ensures flexibility and scalability, enabling future enhancements such as integrating advanced detection models, supporting mobile platforms, or expanding hardware capabilities.

3 System Architecture

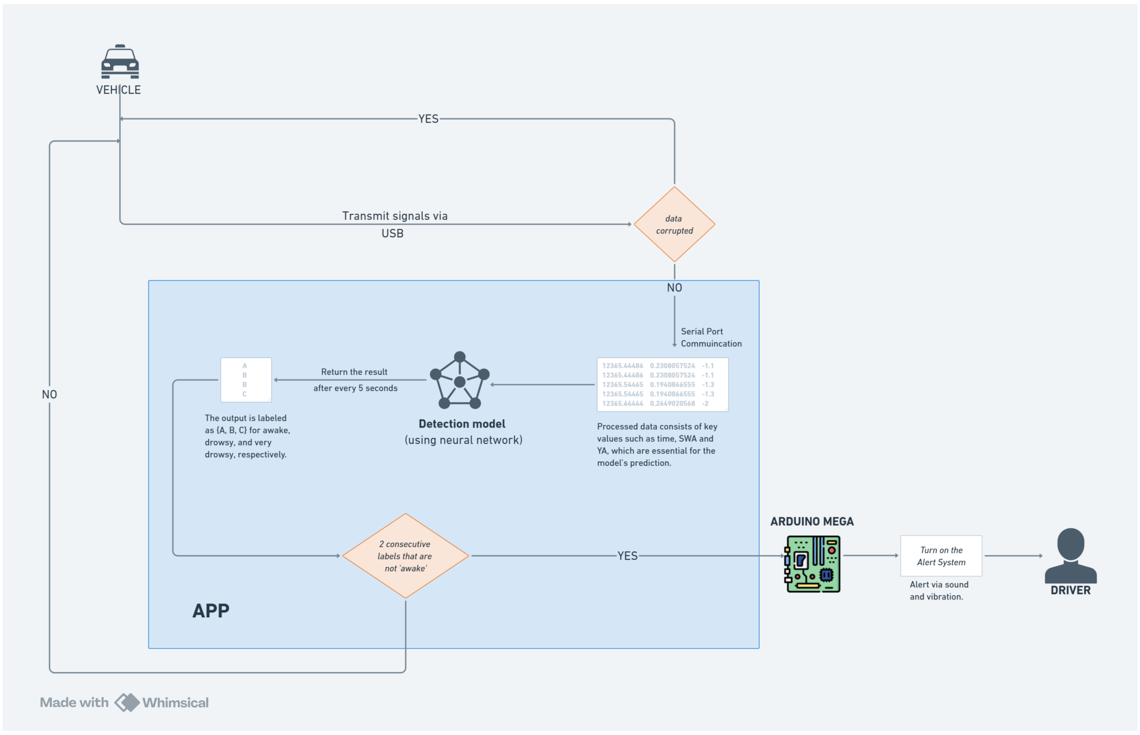

VEHICLE
YES
Transmit signals via USB
data corrupted
NO
NO
Serial Port Commuincation
12365.44486 0.2308057524 -1.1
12365.44486 0.2308057524 -1.1
12365.54465 0.1940866555 -1.3
12365.54465 0.1940866555 -1.3
12365.64444 0.2649020568 -2
Processed data consists of key values such as time, SWA and YA, which are essential for the model's prediction.
Detection model
(using neural network)
Return the result after every 5 seconds
A
B
B
C
The output is labeled as {A, B, C} for awake, drowsy, and very drowsy, respectively.
2 consecutive labels that are not 'awake'
YES
APP
ARDUINO MEGA
Turn on the Alert System
Alert via sound and vibration.
DRIVER
Made with Whimsical

Figure 2

Figure 2 above shows a top-down description of how the drowsiness detection system is expected to work and how components interact with one another. The detection model, implemented using a neural network, continuously receives vehicle data such as time, SWA, and YA through serial port communication. It processes the data and produces a classification label (A, B, or C) every 5 seconds, representing the driver's state: awake, drowsy, or very drowsy, respectively.

The application logic within the app continuously monitors these outputs. If two consecutive predictions are not labeled as "awake," the app sends a command to the Arduino to trigger the alert system. Finally, the Arduino alert module activates the warning mechanism (sound and vibration), which directly notifies the driver to regain attention. This process ensures the loop of sensing, detecting, and alerting is continuously maintained in real time.

4 Component Design

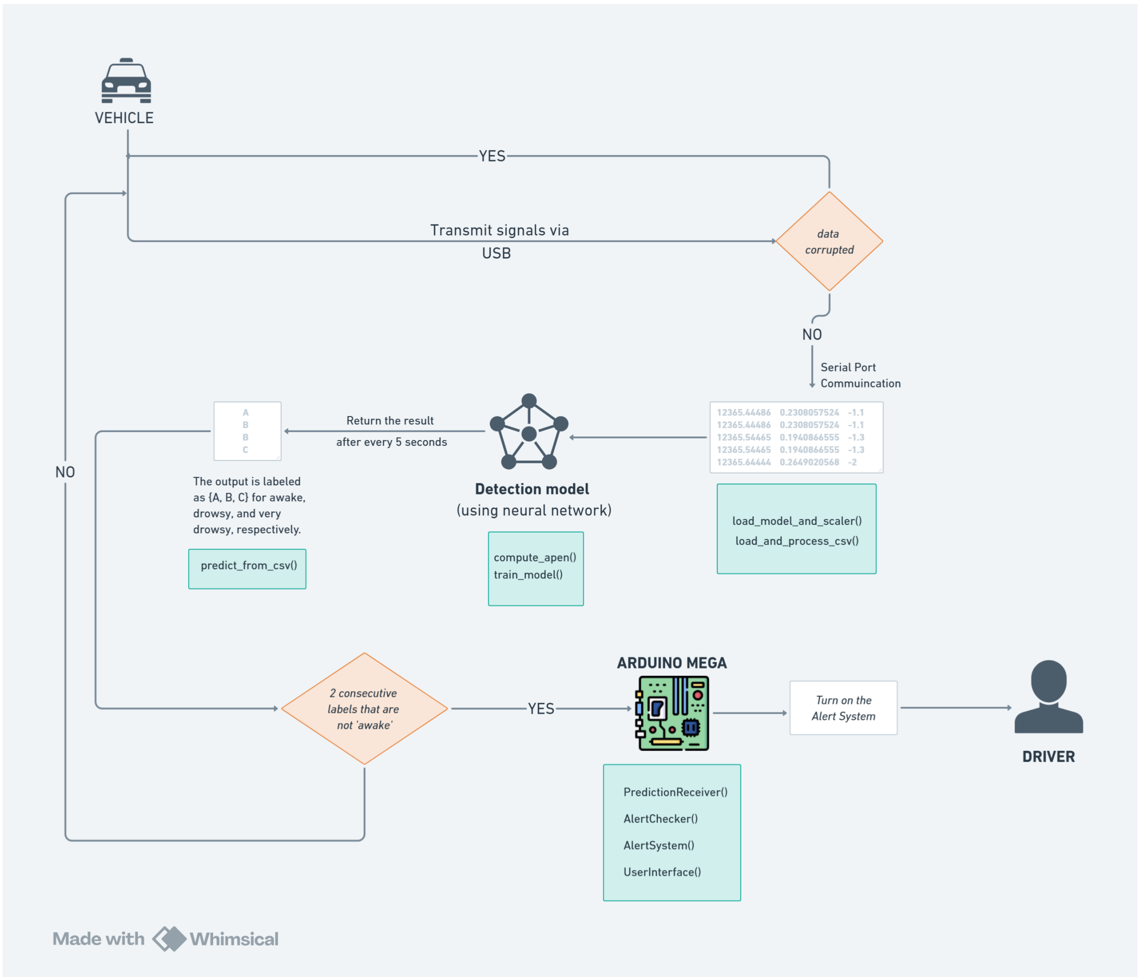
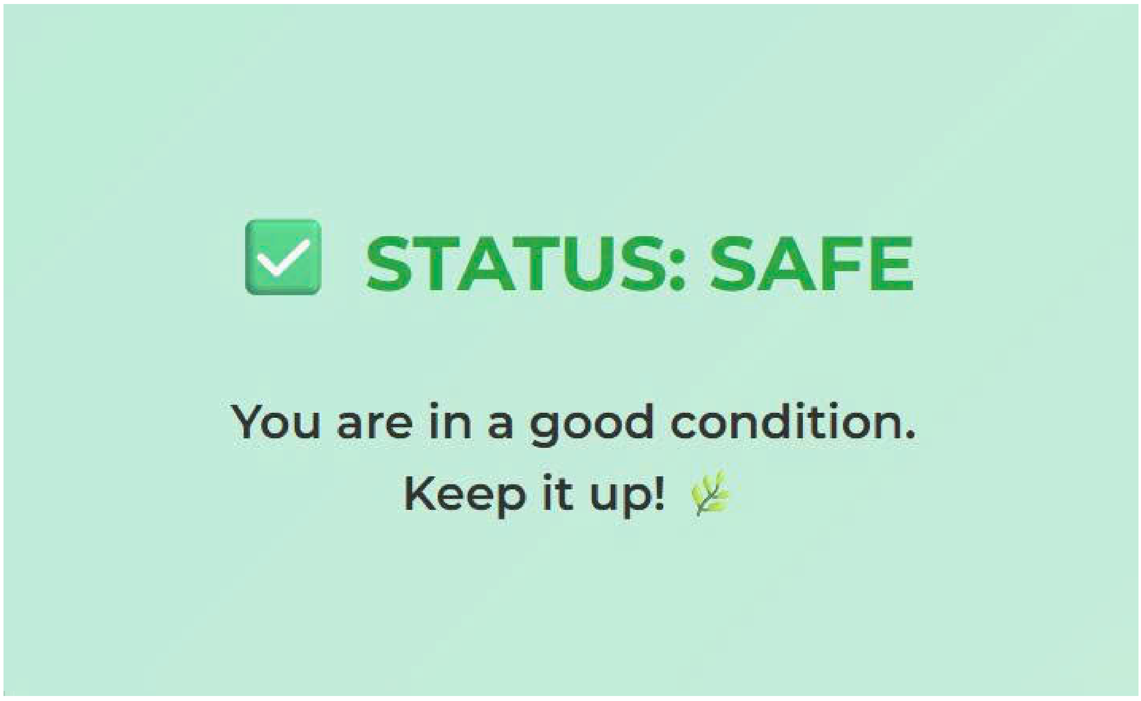


Figure 3

The drowsiness detection system is composed of three main components: the detection model, the alert management logic, and the microcontroller module.

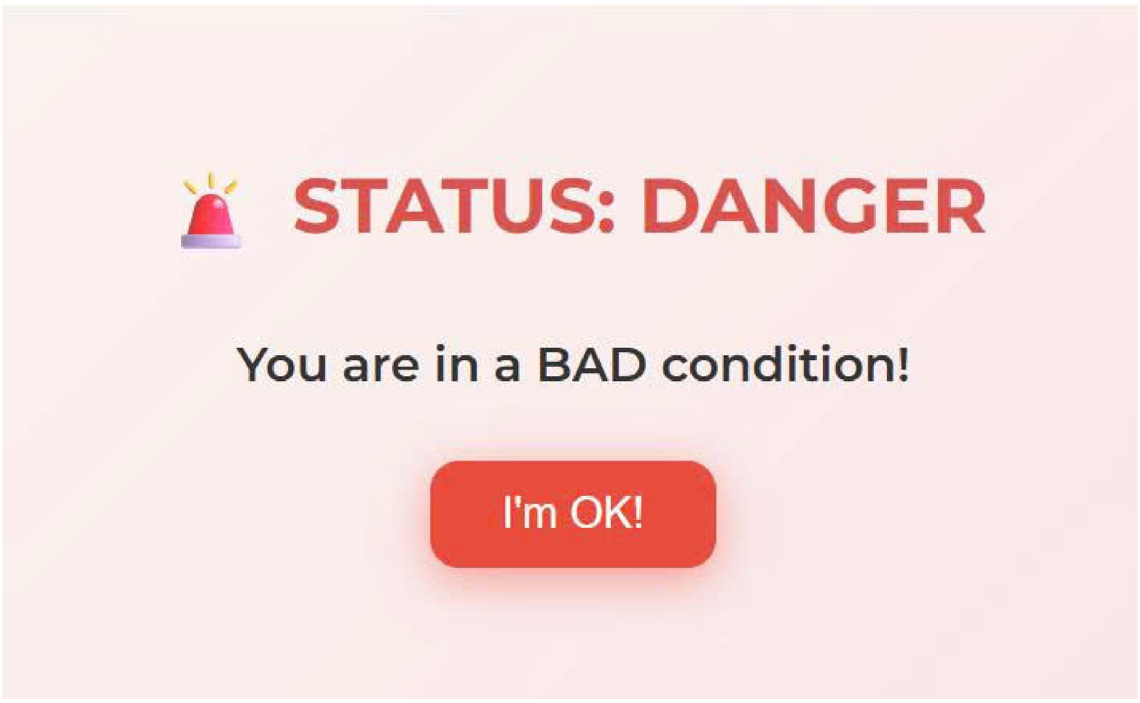
The detection model is implemented using a lightweight neural network that continuously receives processed driving data through serial communication. Internally, it includes modules for preprocessing, entropy calculation, model training, and real-time prediction (`load_model_and_scaler()`, `predict_from_csv()`). It outputs a driver status label every 5 seconds: awake (a), drowsy (b), or very drowsy (c). The alert logic monitors the prediction stream and triggers a response if two consecutive non-awake labels are detected. This logic is implemented within the application environment and is responsible for initiating the warning process.

5 Human Interface Design


STATUS: SAFE

You are in a good condition.
Keep it up!

Figure 4: SAFE Screen


STATUS: DANGER

You are in a BAD condition!

Figure 5: DANGER Screen

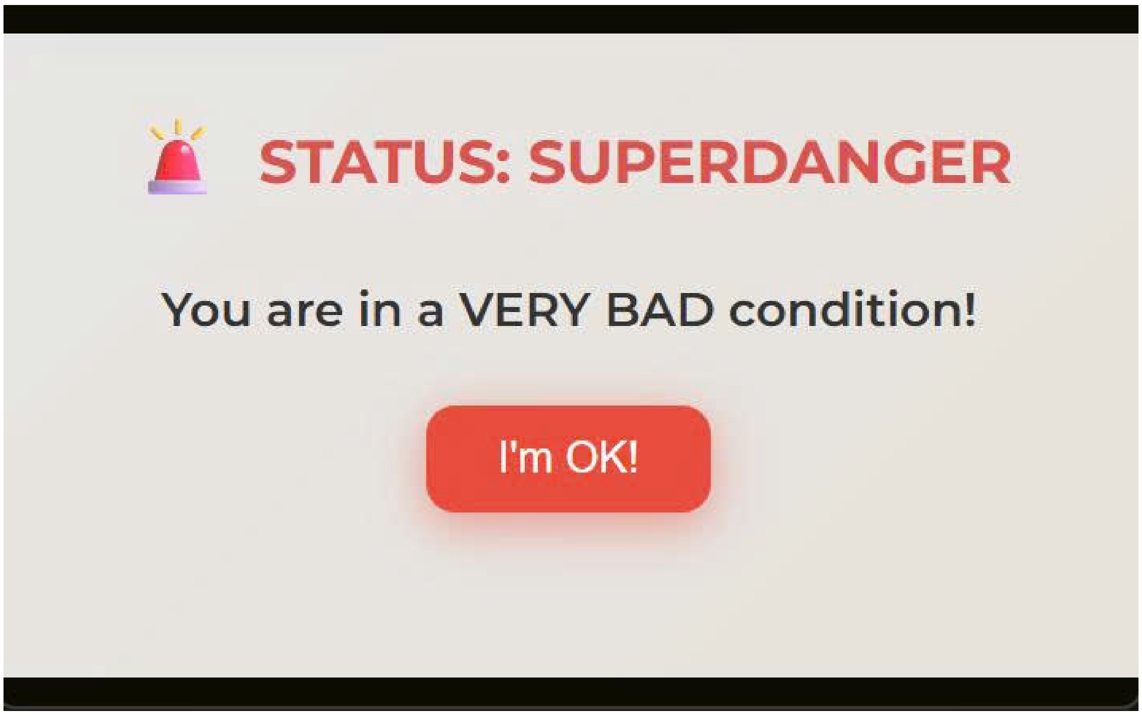


Figure 6: SUPERDANGER Screen

The application interface is designed to be clear, responsive, and distraction-free, ensuring safe use during vehicle operation. It continuously monitors the driver's condition and updates the screen status based on predictions from the fatigue detection model .

Upon launching, the application displays the SAFE screen (Figure 4), indicating that the driver is in good condition. This screen remains visible as long as the model predicts an awake state.

If the system detects two consecutive predictions indicating a risk, the interface automatically switches to the DANGER screen (Figure 5). This screen prominently displays a warning message and includes a confirmation button ("I'm OK!") that the driver must press to acknowledge the alert. Once confirmed, the interface returns to the SAFE screen.

If the condition worsens and the model detects a severe risk level, the application switches to the SUPERDANGER screen (Figure 6). Similar to the DANGER state, the driver must press the "I'm OK!" button to acknowledge the alert, after which the system returns to the SAFE screen.

After confirmation, the alert system (buzzer or vibration) is deactivated, and the application resumes monitoring the driver, repeating the same process for subsequent predictions.

6 Testcase

6.1 Test 1: Safe Situation

Description: In this scenario, the driver remains fully alert and attentive throughout the monitoring period. The primary objective of this test is to verify that the system does not generate unnecessary alerts when the driver's condition is normal and there is no indication of drowsiness or fatigue.

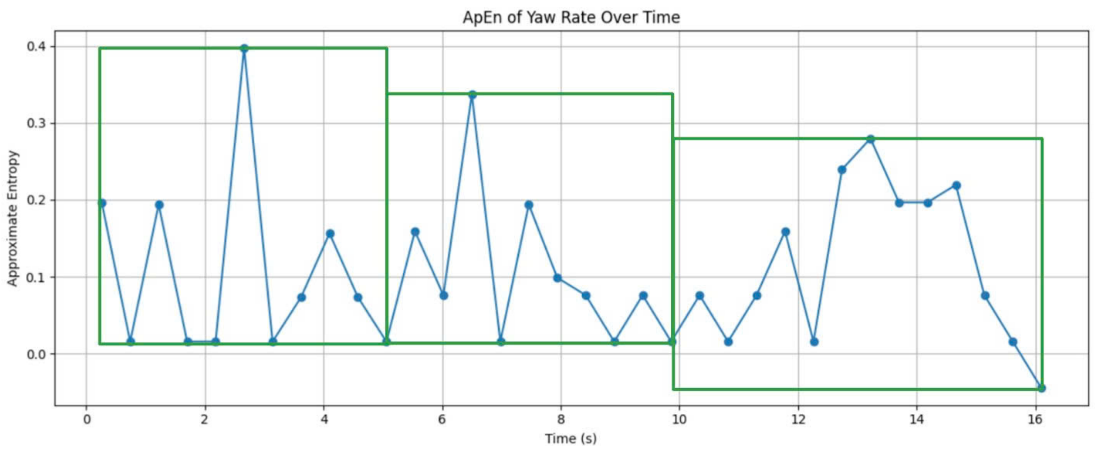


Figure 7

Analysis: As shown in Figure 7, the Approximate Entropy (ApEn) values remain relatively stable within a normal range throughout the observation window. These values correspond to steering behavior that is typical of an alert driver.

The detection model continuously processes incoming CAN signals and assigns a status label every 5 seconds. During this test, the model consistently outputs Label A (Awake), indicating that the driver is in a safe state. Because no consecutive non-awake labels are detected, the system does not initiate the warning process, and no alert (visual, sound, or vibration) is triggered.

Expected Result: The program should remain inactive during the entire test duration, confirming that the alert logic functions correctly by avoiding false alarms when the driver is attentive.

Outcome: The system behaves as expected, maintaining the **SAFE** screen and refraining from generating alerts.

6.2 Test 2: Cautionary Situation

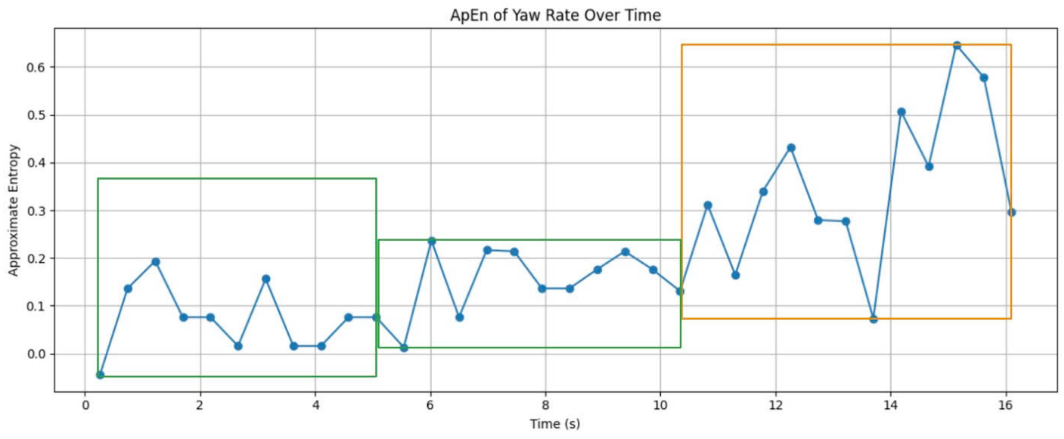


Figure 8

Analysis: As shown in Figure 8, the Approximate Entropy (ApEn) values initially remain within a stable, low range (highlighted by green rectangles), which corresponds to normal, controlled steering behavior typical of an alert driver.

Around the 11-second mark, the entropy values start to increase and become more erratic (highlighted by the orange rectangle). This pattern indicates a decrease in steering control consistency, suggesting the driver is entering a slightly drowsy state. The detection model, which evaluates CAN signal patterns and assigns a status label every 5 seconds, reflects this change by transitioning from Label A (Awake) to Label B (Drowsy).

When two consecutive Label B segments are detected, the system's internal logic classifies the situation as cautionary and automatically activates the alert mechanism. This includes changes in the graphical interface (e.g., switching from SAFE to WARNING) and potentially audible or visual cues depending on implementation.

Expected Result: The system should recognize the sustained transition to a drowsy state and trigger the corresponding cautionary alert.

Outcome: The system correctly identified the change in driver behavior and issued the alert after detecting consecutive Label B segments. The interface transitioned from the SAFE screen to the CAUTION/ALERT state, confirming that the drowsiness detection and alert mechanisms are working as designed.

6.3 Test 3: Warning Situation

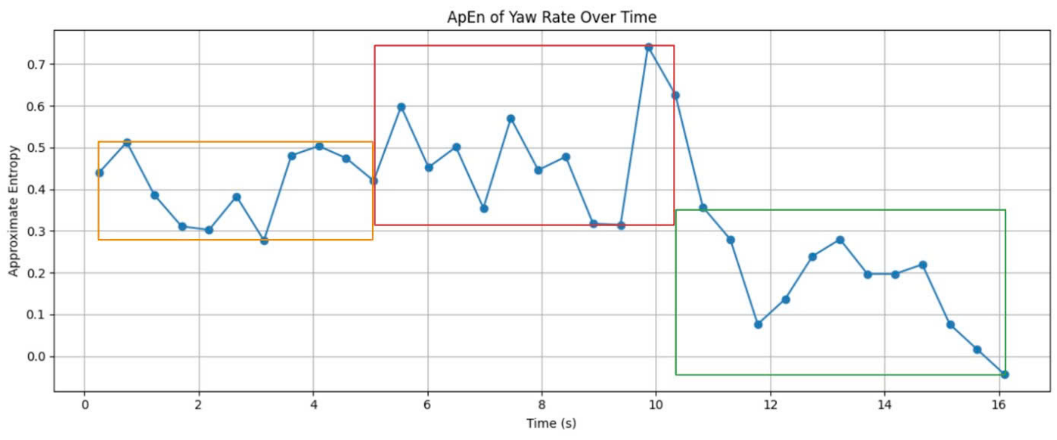


Figure 9

Analysis: As illustrated in Figure 9, the Approximate Entropy (ApEn) initially shows medium-level fluctuations (highlighted in orange), consistent with Label B (Slightly Drowsy). Over time, the values increase both in amplitude and irregularity (red box), indicating the driver is becoming more drowsy and entering Label C (Very Drowsy). Eventually, the entropy values begin to stabilize again but at a lower range (green box), possibly due to reduced steering input or slower responses, which also supports the drowsiness hypothesis.

Upon detecting a transition from state B to state C with sustained high entropy values, the system initiates a stronger alert protocol. This may involve flashing red warning messages, louder alarm tones, or more aggressive tactile alerts (e.g., vibration or seat feedback), depending on the system implementation.

Expected Result: The system should transition from a cautionary to a warning state and issue a high-level alert when sustained Label C values are detected.

Outcome: The program successfully transitioned from Label B to Label C and activated the appropriate warning measures. The interface changed to a red WARNING screen and simulated an urgent alert, verifying that the system can recognize and respond to serious driver drowsiness.

6.4 Test 4: Prolonged Drowsiness Situation

Description: This test case evaluates the system's performance when the driver remains in a slightly drowsy state (Label B) over an extended period. The goal is to verify whether the system can detect a prolonged cautionary state and maintain persistent alerts until the driver's condition improves.

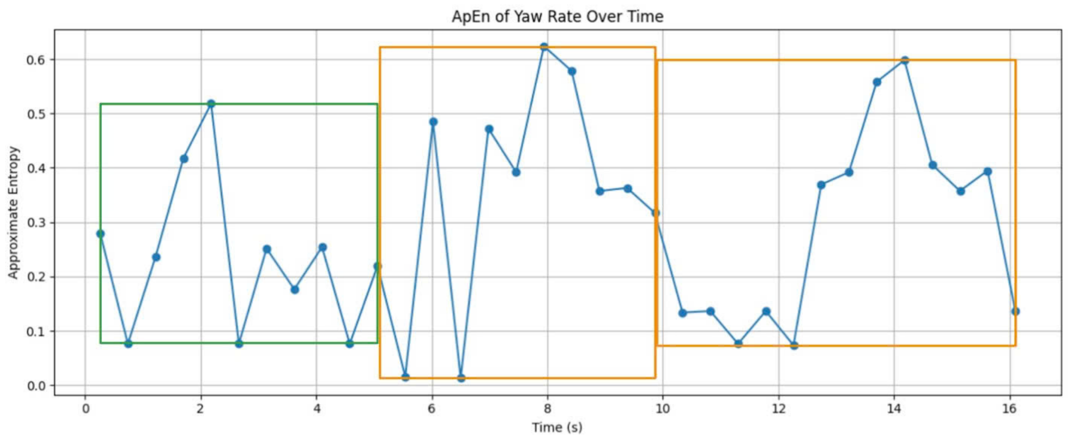


Figure 10

Analysis: Figure 10 shows the Approximate Entropy (ApEn) of the yaw rate over a 16-second interval. Initially, the entropy values (green box) are relatively low and stable, indicating an alert driver (Label A). Around the 5-second mark, entropy begins to increase in amplitude and frequency, entering a range consistent with Label B (Slightly Drowsy). This elevated state is maintained for several consecutive windows (highlighted by the orange rectangles).

Unlike Test 2 where the system quickly returned to a safe state, here the signal continues to oscillate within the cautionary range without returning to baseline. The system recognizes this sustained Label B pattern and continues to display a warning state on the interface. No transition to Label C is detected, but the alert remains active due to the persistent drowsiness level.

Expected Result: The system should detect the prolonged presence of Label B, avoid transitioning back to the SAFE screen prematurely, and maintain an active alert (visual or audio) until the driver's state improves.

Outcome: The program correctly identified the prolonged cautionary state and sustained the alert accordingly. No false recovery to the safe state was triggered during this extended period of drowsiness, confirming the reliability of the system in handling long-term fatigue detection.

6.5 Test 5: Critical Situation

Description: This test simulates a critical condition in which the driver's state rapidly declines from fully alert to very drowsy within a short period. The aim is to evaluate whether the system can correctly recognize this accelerated transition and escalate the alert level in a timely and accurate manner.

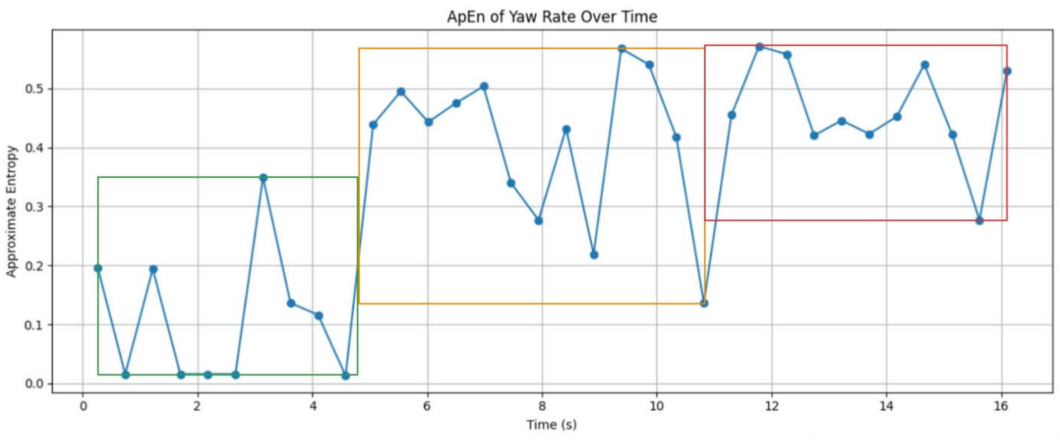


Figure 11

Analysis: As illustrated in Figure 11, the Approximate Entropy (ApEn) of the yaw rate shows three distinct phases. In the initial phase, the ApEn values remain low and consistent, indicating that the driver is in an alert state. This corresponds to Label A, and no alert is issued. However, as time progresses, the values begin to rise, entering a medium range. This change reflects slightly unstable steering behavior and results in the assignment of Label B, prompting the system to issue a cautionary alert.

The situation escalates further when the entropy values spike to higher levels and fluctuate more irregularly. This pattern, categorized as Label C, is interpreted by the system as a sign of serious drowsiness. At this stage, the program transitions from a cautionary state to a warning state, triggering a high-level alert.

Expected Result: The system should detect the progression from Label A to Label B, and subsequently from B to C. It is expected to issue no alert during the initial safe phase, activate a cautionary alert during the slightly drowsy period, and escalate to a strong warning as the entropy becomes critically high.

Outcome: The system functioned as intended. It accurately identified the critical shift in driver behavior, responded appropriately to each stage of drowsiness, and issued alerts at the correct intensity. This confirms that the alert mechanism can handle rapid transitions and provide timely warnings to enhance driver safety.

7 System Requirements

7.1 Minimum Hardware Requirements

- **Processor:** Intel Core i5 (8th generation or newer) or equivalent.
- **RAM:** 8 GB.
- **Disk Space:** At least 3 GB of free storage for application files, CAN data, and model files.

- **Communication Port:** At least one USB or Bluetooth port to connect to the microcontroller (e.g., Arduino).

7.2 Minimum Software Requirements

- **Operating System:** Windows 10 or newer (64-bit).
- **Python Environment:** Python 3.9+ with libraries including `numpy`, `pandas`, `scikit-learn`, and `serial`.
- **Web Framework:** Node.js and npm.
- **IDE (Recommended):** Visual Studio Code, PyCharm, Arduino IDE.

8 Innovation

An important innovation of DFDWA is its ability to collaborate with Advanced Driver Assistance Systems (ADAS) during critical fatigue. The system may activate lane-keeping, reduce speed, or suggest rest stops. In future versions, it could initiate autonomous pull-over if supported. This transforms DFDWA from a passive alert tool into an active safety mechanism, enhancing real-time intervention and aligning with modern semi-autonomous driving trends.