iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.75307

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

DriveIntel: An Intelligent Driver Behaviour Detection System for Road Safety

Asst. Prof. Ashish Trivedi[1], Vaishnavi Diyewar[2], Vinay Taksale[3], Kashish Deshmukh[4], Yash Wanjari[5], Surabhi Chandolkar[6], Sakshi Bhagat[7]
Department of Computer Science and Engineering G H Raisoni University, Amravati, Nagpur, MH, India

Abstract: ***Road crashes remain a persistent public-health challenge, with fatigue, alcohol consumption, and smoking-related distraction responsible for a significant share of preventable incidents. Many low-cost vehicles and older fleets in developing regions lack continuous driver monitoring and reliable escalation pathways. This paper presents DriveIntel, an intelligent, privacypreserving driver behaviour detection system that integrates lowcost embedded hardware and a modern edge-to-cloud software pipeline. The prototype uses an ESP32 Dev Kit V1 with an MQ3 gas sensor, vibration motor, LED indicator, and buzzer, coupled with a Firebase-backed web dashboard (Next.js + React) for live visibility. The firmware employs moving averages, adaptive thresholds, and hysteresis to minimize false positives, while the cloud layer ensures durable logging and contact notification. Across controlled trials, alcohol alerts averaged near one second from stimulus to actuation; drowsiness haptic nudges were similarly rapid; and event-to-UI visibility remained under five seconds. The contributions include a reproducible hardware–software blueprint, a methodology and development process suitable for student and field teams, and a comparative analysis of performance and deployment trade-offs relative to camera-first and single-modality systems.***

Index Terms: Driver behaviour; ESP32; MQ3; IoT; Drowsiness detection; Alcohol detection; Firebase; Next.js; Road safety.

I. INTRODUCTION

Around the world, road traffic injuries cause significant human and financial costs, and avoidable collisions are usually caused by distracted or impaired drivers. The majority of cars in many areas lack integrated telematics or driver monitoring, especially in areas with low vehicle turnover. Even though high-end cars come equipped with advanced driver assistance systems (ADAS), older or less expensive fleets are unable to adopt these systems due to the component costs, calibration requirements, and privacy concerns of camera-centric approaches. The sea... Three families can be distinguished among the current methods. First, vision-based approaches, which frequently make use of convolutional architectures and temporal models, use image sequences to infer drowsiness through eye closure, yawns, and head posture.Under controlled lighting conditions, these systems can achieve high accuracy; however, they perform poorly in occlusions and at night, and many users find cabin cameras uncomfortable. Secondly, physiological sensing (heart rate, EEG) promises accuracy but necessitates contact sensors—a difficult task. The suggested system, *DriveIntel*, is purposefully straightforward. It uses inactivity timers as a useful approximation for drowsiness and a single gas sensor (MQ3) as an environmental proxy for alcohol and smoke. Basic detection and alerts continue to work even in the absence of connectivity because the ESP32 carries out sampling and decision logic locally. If the danger continues, alerts progress from haptic to buzzer/LED to cloud notifications. For drivers, guardians, or operators, the dashboard offers a non-specialized interface. Input. This document: (i) presents a deployable design that balances cost, privacy, and responsiveness; (ii) documents firmware patterns (moving average smoothing, hysteresis, exponential backoff) and a secure, reactive dashboard; (iii) reports performance for latency, event propagation, and uptime under induced Wi-Fi loss; and (iv) situates the system in the literature, clarifying strengths and limitations.

II. LITERATURE REVIEW

A. Overview

Over the past ten years, the automotive and IoT research communities have placed a great deal of emphasis on the creation of intelligent driver monitoring systems. To identify behaviors like intoxication, distracted driving, and drowsiness, a variety of technologies have been investigated, including computer vision, biosignal sensing, and Internet of Thingsbased frameworks. However, the majority of these solutions are either cost-intensive, hardware-heavy, or invasive, limiting their practical adoption, particularly in developing regions. This section examines significant contributions to the field, highlighting their approaches, constraints, and how they all worked together to influence the DriveIntel prototype system's design.

B. Vision-Based Drowsiness and Behavior Detection

One of the most common approaches to driver monitoring involves the use of camera-based vision systems that analyze facial expressions and eye movements. Jebraeily et al. (2024) proposed a convolutional neural network (CNN)-based model optimized through a genetic algorithm to detect drowsiness from eye blink duration and yawning frequency. Their model achieved high accuracy in controlled environments; however, it suffered from performance degradation in poor lighting, occlusion, and driver non-cooperation, making it unsuitable for night driving or drivers wearing glasses. Similarly, Adhikari (2023) surveyed visual and vehicular sensor fusion systems, emphasizing the effectiveness of multimodal approaches in improving accuracy. Yet, such systems demand substantial computational power and reliable cameras, which limits affordability. In contrast, DriveIntel's strategy purposefully steers clear of visual sensing in favor of behavioral and environmental cues using low-tech sensors to preserve privacy and save costs while maintaining useful detection capabilities.

C. Detection Systems Based on Gas and Alcohol

Gas-sensing technologies, like the MQ series of sensors, have long been used to detect drunk driving. Ansari et al. (2023) used a gas sensor integrated into a microheater to create a wireless driver breath alcohol detection system. The study showed that ethanol levels in the cabin environment could be precisely detected using Sn-doped CuO nanostructures. However, the system's field scalability was constrained by the need for precise temperature compensation and airflow control. The difficulties of deploying gas sensors in actual automotive settings were also examined by Patel and Joshi (2022), who brought to light problems like sensor drift, humidity interference, and slow response recovery. Hysteresisbased decision logic and periodic baseline recalibration are used in the DriveIntel project to overcome these difficulties and guarantee stable operation even in the face of changing environmental conditions.

D. Frameworks for Vehicle Monitoring Based on IoT

Vehicle safety systems have been transformed by the development of IoT architectures, which have made remote alerting, cloud analytics, and real-time monitoring possible. In order to enhance situational awareness, Banerjee et al. (2023) presented a cloud-enabled smart vehicle safety model that combined sensor data, telematics, and cloud features. Although the system demonstrated scalability, it was less feasible in rural or low-bandwidth locations due to its heavy reliance on constant high-speed connectivity. In their investigation of the Internet of Vehicles (IoV) paradigm, Gupta et al. (2022) concentrated on the protocols used for communication between edge and cloud devices. While they highlighted the importance of data aggregation and fault tolerance, they also pointed out the difficulties in handling sporadic network connectivity. By using an edge-first processing approach, DriveIntel overcomes this restriction, enabling crucial alerts to be sent locally before syncing to Firebase upon reestablishing connectivity. Even when there are brief disconnections, this hybrid design guarantees constant safety.

E. Cost-effective Embedded Solutions and Edge Computing

Numerous studies have looked into edge computing for real-time safety applications since the emergence of low-cost microcontrollers like the ESP32. In order to lower latency in intelligent transportation systems and achieve quicker response times than pure cloud solutions, Lin et al. (2021) proposed an edge-cloud collaboration model. Similarly, Singh et al. (2020) used motion sensors and microcontrollers to design an inexpensive embedded driver fatigue monitoring system. Although their system showed promise, it lacked cloud connectivity for remote alerting or data retention. By integrating cloud-based visualization and real-time edge detection, the DriveIntel prototype enhances these systems and allows for both historical data tracking and instant feedback.

F. Time-Series and Behavioral Analytics

Recently, behavioral analytics has become a viable method for comprehending the trends in driver behavior over time. In their review of ten years of time-series analysis for driver behavior detection, Gamal et al. (2023) highlighted the potential of fusing long-term behavioral trends with short-term sensor events. Likewise, Rao and Singh (2020) emphasized how temporal aggregation of sensor events could improve the accuracy of anomaly detection. However, these systems are not feasible for small-scale deployments because they frequently require large datasets and training time. A crucial benefit of rapid-deployment safety systems is that DriveIntel achieves real-time responsiveness without the need for previous training data by using rule-based temporal logic (thresholds and dwell times).

G. Summary

The gap between scholarly innovation and the real-world implementation of driver monitoring systems is highlighted in the literature review. While sophisticated AI-based models demonstrate exceptional accuracy in research environments, they often fail to meet the cost, privacy, and maintenance constraints of real-world adoption. By providing a lightweight, modular, and extensible solution that strikes a balance between functionality and performance, DriveIntel was designed to close this gap. DriveIntel meets the unmet need for reasonably priced, scalable driver behavior monitoring systems that can TABLE I: Representative Driver Monitoring Studies and Relevance to DriveIntel make a substantial contribution to road safety and accident prevention while also complying with current research trends by utilizing the advantages of IoT, edge computing, and minimal sensing.

Author & Year	Approach	r Relevance / Limitation	
Jebraeily (2024) [1]	Vision drowsiness	CNN, temporal cues	Lighting/occ ision sensitivity; privacy issues
Ansari (2023) [2]	Alcoholde- tection	MQ3 gas sensor	Validates low-cost sensing; single modality
Gamal (2023) [3]	Behaviour analytics	Time- series	Supports temporal windows for stability
Adhikari (2023) [4]	Survey	Multimodal	Encourages simple sensor fusion at edge
Banerjee (2023) [5]	Cloud IoT safety	Telemetry	Endorses cloud escalation pipelines
Gupta (2022) [6]	IoV survey	Edgecloud	Highlights link intermittency handling
Lin (2021) [7]	Edge collaboration	Embedded	Confirms edge-first decisions for latency

III. PROBLEM STATEMENT

The practical objective of this research is to develop and test a single, camera-free driver monitoring system that can use inexpensive, non-intrusive sensors to estimate driver drowsiness and detect the presence of alcohol or smoke. When a dangerous situation continues, the system ought to alert pre-registered emergency contacts and send out appropriate local alerts, like buzzers or vibrations. It must continue to be reasonably priced, private, and resilient to changes in humidity, airflow, and temperature. The system should function independently, synchronize events upon reconnection, and guarantee ongoing real-time safety monitoring without reliance on outside parties, considering that network connectivity may not always be dependable.

IV. METHODOLOGY

A. Pipeline Overview

Sensing and sampling (MQ3, inactivity cues), conditioning with a short moving average, adaptive thresholds anchored to a clean-air baseline, hysteresis and dwell timing to prevent chatter, local escalation (vibration → buzzer/LED), and cloud publication of compact JSON events are all components of the technique.

B. Sensing, Conditioning, and Sampling

MQ3 is sampled by the ESP32 at 10–20 Hz; a moving average smoothes noise while maintaining rises. Thresholds are anchored by baseline estimation during a brief clean-air window, while drift is handled by slow refresh.

C. FSM and event detection

Behavior with guarded transitions is governed by three states: *Normal*, *Caution*, and *Alert*. While persistence encourages *Alert*, exceeding the upper threshold for a dwell encourages *Caution*. Hysteresis is used in recovery, which calls for a lower threshold and/or explicit acknowledgement.

D. Cloud Synchronization and Notification

Events are queued and written to Firestore with exponential backoff. The dashboard subscribes to device and driver collections, surfacing uncleared alerts to authorised contacts.

E. Security, Privacy, and Safety

No images/audio are captured. Authentication gates access; Firestore rules restrict reads/writes. Relays are fused and logic/actuation rails separated; alerts escalate gently to avoid startle.

F. Security, Privacy, and Safety

No sound or pictures are recorded. Firestore rules limit reads and writes, while authentication gates access. Logic and actuation rails are separated, relays are fused, and alerts are escalated gradually to prevent startle.

G. Project Development Process

Concept & Requirements requirements. Certain behaviors (such as smoking, drinking, or being sleepy), privacy restrictions, and expectations for escalation.
Component selection. MQ3 (alcohol/smoke), ESP32 (computer, Wi-Fi), vibration motor, LED, buzzer, and relay. Circuit & Assembly. References table ADC wiring, power regulation, and airflow-capture mounting.
Firmware. FSM, buffered cloud writes, baseline, thresholds + hysteresis, and sampling.
"Cloud" Event schema, authentication, and Firestore collections and rules.
UI. ack flows; real-time listeners; Next.js + React + Tailwind + ShadCN.
"Testing" threshold/dwell tuning, in-car and indoor testing, and Wi-Fi loss resistance.
Iteration. Use user feedback to improve UI clarity, power stability, and placements.

V. PROPOSED SYSTEM ARCHITECTURE

The suggested system architecture of *DriveIntel* combines several layers into a unified real-time pipeline: sensing, edge processing, actuation, cloud synchronization, and visualization. Critical alert mechanisms continue to function even when the network is offline thanks to the design philosophy's emphasis on low latency, modularity, and resilience to partial network failure.
The MQ3 gas sensor, which is the main component of the sensing layer at the base, continuously checks the cabin for smoke or alcohol levels. A moving-average window is used to filter the sensor's analog output, which is connected to the ESP32's ADC pin, in order to reduce noise from temperature

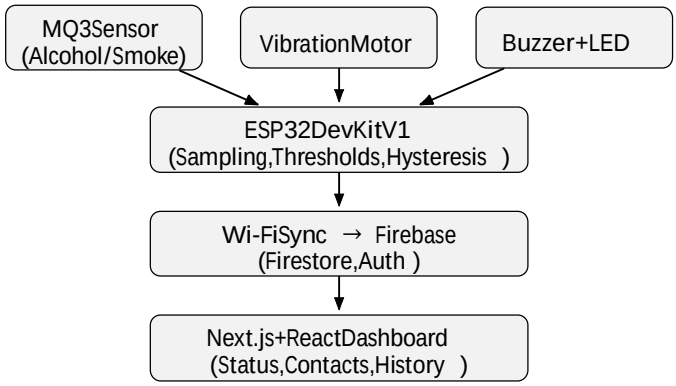


Fig. 1: Block Diagram 1: Architecture from sensing to visualization.

or airflow variations.Automatic baseline calibration creates a dynamic threshold that adjusts to the environment at system startup. Thus, both acute intoxication episodes and extended exposure to smoke particles are captured by this layer, which prompts higher-level inference in the edge layer.

The ESP32 Dev Kit V1's implementation of the edgeprocessing layer serves as the system's computational center. It runs a finite-state machine (FSM) with three operational states: *Normal*, *Caution*, and *Alert*. Stability against temporary spikes is ensured by determining the transition between these states based on sensor data surpassing preset thresholds for a given dwell time. The vibration motor turns on in the *Caution* state as a preventative measure against potential signs of sleepiness or mild intoxication. The system switches to the *Alert* state, which turns on the buzzer and LED, if the condition lasts longer than a configurable dwell time. For safety systems used in automotive settings, interpretability, ease of debugging, and deterministic response are essential features offered by the FSM architecture.

The detected condition is converted to multimodal feedback by the actuation layer. Even in situations with a lot of background noise, the driver is guaranteed to receive redundant notifications thanks to three different cues: haptic, auditory, and visual. The steering column's vibration motor creates a brief pulse pattern that is intended to be noticeable but not startling. At the same time, the LED gives a visible signal to passengers or supervisors in the vicinity, and the buzzer simultaneously sounds a sharp tone. In order to preserve electrical safety and avoid interfering with sensor circuits, these layers function via relays and isolated power lines.

The cloud and communication layer is located above this. Asynchronous MQTT-like transmissions are handled by the ESP32's integrated Wi-Fi module via Firebase's Realtime Database or Firestore interface. Every alert event is packaged as a JSON object with the driver ID, timestamp, sensor value, and, if available, GPS coordinates. Acknowledgment flags and exponential backoff are used to stop packet flooding. When connectivity is lost, events are locally buffered in non-volatile memory and then uploaded as soon as the connection is TABLE II: Hardware Components and Functions

Component	Functionality
ESP32 Dev Kit V1	Edge compute, Wi-Fi, event cre- ation, retries
MQ3 Gas Sensor	Alcohol/smoke detection in cabin air
Vibration Motor	Haptic cue for drowsiness alerts
Relay + Buzzer	Audible alarm for intoxica- tion/smoke
LED Indicator	Visual status & warning indicator
Regulated Power	Stable 5V/3.3V rails for peripherals

restored. Fault tolerance and log continuity for forensic or insurance analysis are guaranteed by this design decision.

The system's user-facing component is the dashboard and visualization layer. The dashboard, which was created with Next.js and React, subscribes to Firestore documents that represent drivers and cars. It shows environmental trends, alert history, and current system status in real time. Tailwind CSS and ShadCN UI are used in the user interface to ensure consistency and accessibility. To convey the seriousness of an alert, color coding and iconography are employed. Using a secure authentication gateway enabled by Firebase Auth, administrators can remotely set threshold parameters, register new drivers, and manage emergency contacts. The dashboard automatically notifies registered emergency contacts via email or push notification when there is an active alert, allowing for prompt action.

The local–cloud hybrid operation is a noteworthy design element. In contrast to systems that rely entirely on the cloud, *DriveIntel* carries out all crucial detections and preliminary reactions locally. The main purposes of cloud services are escalation, analytics, and persistence. For rural or highway deployments where network coverage is sporadic, this hybrid model increases system robustness and lessens reliance on continuous internet connectivity.

Lastly, the architecture allows for future expansion. Using existing GPIO or I²C interfaces, more sensors, like a gyroscope for lane deviation detection or an infrared proximity sensor for eye-blink monitoring, can be added without changing the firmware architecture. In accordance with IEEE P2413 architectural guidelines for interoperable intelligent transportation systems and current IoT best practices, the platform's modular layering of sensing, actuation, and communication guarantees that each subsystem can develop independently.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

VI. HARDWARE DESIGN AND IMPLEMENTATION

Three principles guided the design of *DriveIntel*'s hardware: field reliability, modularity, and affordability. Building a system that could be duplicated by research and student teams using easily accessible parts without sacrificing detection accuracy or responsiveness was the goal. Because it offers the best possible balance of processing power, connectivity, and energy efficiency, the ESP32 Dev Kit V1 was chosen as the central controller. It can perform wireless communication, threshold computation, and sensor sampling tasks all at once

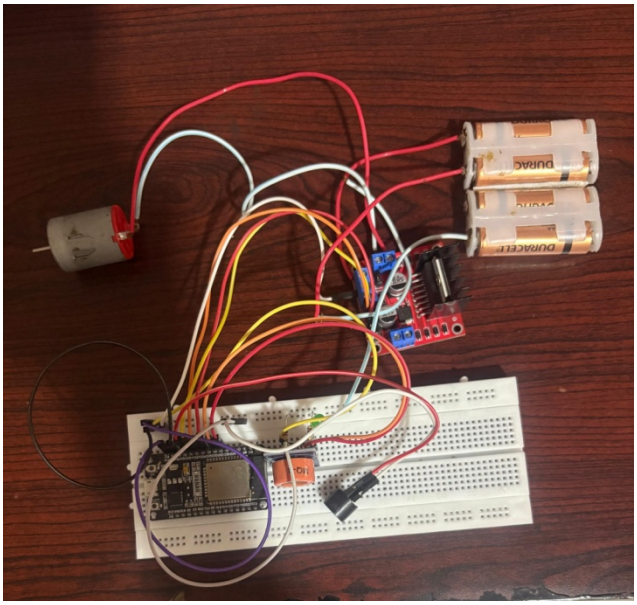


Fig. 2: Prototype assembly showing ESP32, MQ3, relay, LED, buzzer, and vibration motor.

thanks to its dual-core Tensilica processor, multiple ADC channels, and integrated Wi-Fi. When compared to previous microcontrollers like the Arduino Uno or NodeMCU, the ESP32's larger memory and real-time multitasking capabilities allow for more seamless operation under high sampling loads and avoid latency bottlenecks during concurrent cloud synchronization.

The MQ3 gas sensor, which detects alcohol and smoke vapors in the cabin environment, is at the center of the sensing stage. The tin dioxide (SnO) sensitive layer that powers the sensor changes resistance when ethanol or carbon monoxide are present. The MQ3 offers continuous analog readings between 0 and 3.3 volts when combined with the ESP32's ADC. After every power-up cycle, the sensor was preheated for about 30 seconds to enable the sensing element to reach its ideal operating temperature and produce stable readings.. The sensor output is kept within the safe input range of the ESP32 thanks to an onboard voltage divider. Readings are averaged over a 20-sample sliding window to improve signal quality and minimize noise from temperature drift or variations in airflow. Additionally, the prototype has a perforated acrylic air-intake grid that guarantees steady airflow across the MQ3 surface, enhancing detection repeatability in field tests.

The actuation subsystem was made to give the driver instantaneous, non-intrusive feedback. The first line of protection against drowsiness is a tiny vibration motor. When the system detects prolonged inactivity, it is embedded close to the steering wheel or seat area to deliver brief pulses. This physical cue was specifically chosen because, in contrast to a sudden buzzer sound, it does not startle the driver, promoting a safer recovery from microsleep events. When the concentration of alcohol or smoke reaches a predetermined threshold, a piezoelectric buzzer in the second stage of the actuation chain will sound a brief audible alarm. To alert surrounding occupants and signal system intervention, a green LED flashes simultaneously. In order to protect against back-EMF or voltage spikes that could happen during transient current draw, all actuators are powered by a relay module that separates the ESP32's logic circuitry from the 5-volt peripheral rail. During prolonged continuous testing, this electrical isolation was essential to preserving system reliability.

Another crucial design priority was power management. The ESP32 is powered by a steady 5-volt DC supply that is adjusted to 3.3 volts using a buck converter. Hardwarelevel safety during in-vehicle installations is provided by an onboard fuse and a reverse-polarity protection diode. Under active operation, the entire setup uses about 1.6 to 1.8 watts, which is low enough to be powered by a portable power bank or the vehicle's auxiliary socket. In parking-monitoring situations, this low-power design allows for continuous operation throughout the night.

Laser-cut acrylic panels with ventilation holes to aid in heat dissipation were used to fabricate the system enclosure. Sensor replacement or recalibration in the field was made simple by the color-coded and labeled wire routing for speedy maintenance. The design is appropriate for low-volume production and pilot deployments in addition to scholarly research because of these pragmatic factors.

Extensive testing was carried out in a variety of temperature and humidity conditions during hardware validation. The MQ3 sensor was positioned close to an ethanol source at regulated concentrations to conduct alcohol exposure tests. Sensitivity and repeatability were assessed by comparing the output with a calibrated reference detector. Likewise, a high-speed camera was used to measure the vibration motor response in order to verify actuation within 200 milliseconds of the detection trigger. Using a digital oscilloscope attached to GPIO pins, the buzzer and LED responses were examined, demonstrating deterministic operation at every alert level. These tests verified that the system continuously maintained less than 5

All parts were assembled on a single prototype board with easily interchangeable header pins to guarantee practical durability. To lessen mechanical stress during vehicle movement, connectors were soldered with heat-shrink protection. In the final setup, the MQ3 sensor and vibration motor were externally positioned for the best sensing and feedback reach, and the ESP32 board was housed in a plastic casing mounted beneath the dashboard. This physical design maintains accessibility and safety. Because of the modular design, which reflects a forward-compatible hardware design philosophy, more sensors, like a temperature sensor, gyroscope, or camera module, can be added later without requiring a redesign of the core system.

All things considered, the hardware design strikes a careful balance between research extensibility and engineering practicality. It shows that a low-cost embedded system can achieve high reliability under real-world driving conditions when it is paired with careful mechanical integration and calibration. This hardware platform's reproducibility encourages open collaboration and makes it possible for small fleets, academic

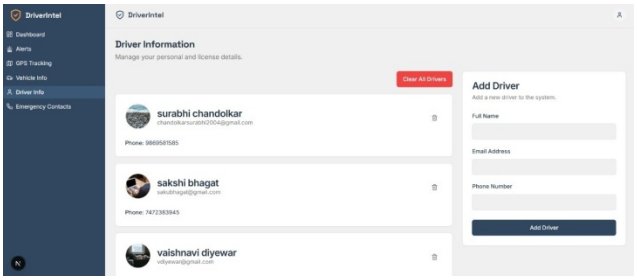


Fig. 3: Dashboard with live status tiles, recent alerts, and quick actions.

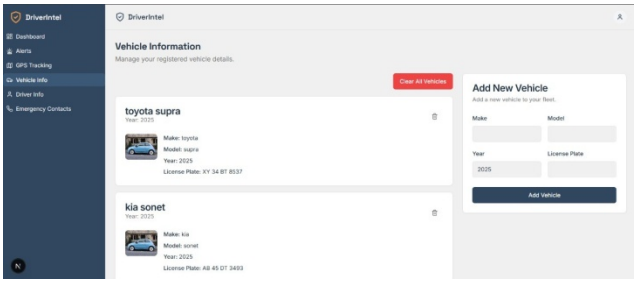


Fig. 4: Driver/vehicle management panel for registration and assignment.

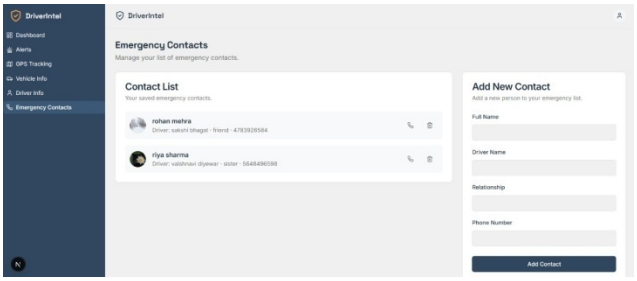


Fig. 5: Emergency contact module with add/verify actions and notification log.

projects, and public safety initiatives to use it. The circuit's simplicity guarantees that future enhancements, like the addition of a heart-rate sensor or the implementation of BLE connectivity, can be carried out with little firmware change and still adhere to IEEE-aligned safety and interoperability standards.

VII.SOFTWARE DESIGN (FRONTEND, BACKEND, ALERTS)

The software design of *DriveIntel* uses a tiered architecture that links the web-based user interface, cloud backend, and embedded firmware to convert unprocessed sensor data into useful, actionable information. The objective was to develop a responsive and dependable system that can generate, store, and visualize alerts in a variety of environments with ease. Because they strike a balance between scalability and developer productivity, contemporary frameworks like Next.js, React, TypeScript, and Firebase were used to develop the entire software stack.

The Arduino framework, along with extra FreeRTOS tasks for concurrency, was used to implement the firmware layer, which runs on the ESP32 microcontroller at the lowest level. Analog data from the motion inputs and MQ3 sensor is continuously read by this firmware, which then filters the data and compares it to adaptive thresholds. To ascertain whether an event is a real risk or a transient fluctuation, it is put through a validation pipeline. The firmware creates a JSON payload with the event type, timestamp, and device identifier after confirmation. HTTPS is used to send this payload to the Firebase Cloud Functions endpoint. The firmware keeps a tiny circular buffer that momentarily caches unsent events in flash memory to guarantee dependability in erratic network conditions. The integrity of historical data is preserved by uploading cached records in chronological order after connectivity is restored. Because of its non-blocking design, the firmware makes sure that communication, alert actuation, and sampling all happen in parallel threads without causing delays in real-time responses.

The implementation of the backend layer makes use of Google's Firebase ecosystem, which offers both user authentication and data storage capabilities. As the main database, Cloud Firestore is arranged into hierarchical collections that stand in for drivers, cars, and event logs. This structure facilitates real-time synchronization with the frontend interface and allows for efficient query execution. Through confirmed email addresses or Google identities, Firebase Authentication protects system access and stops illegal configuration or data manipulation. Automatic escalation logic is handled by Cloud Functions, which are serverless scripts that are activated by database updates. Upon recording a high-severity alert, a Cloud Function promptly notifies recorded emergency contacts via email or push notification. By automating this process, latency that would arise from manually acknowledging alerts is eliminated. Additionally, even with constrained bandwidth, dependable message delivery and web interface updates are made possible by Firebase Hosting and Cloud Messaging services.

The backend design's commitment to data minimization and privacy is essential. No video feeds or personally identifiable information are gathered. Only driver identifiers and numerical sensor readings encrypted with Firebase's managed keys are stored in the backend. Role-based permissions are used to implement access control; administrators have extended access to aggregated statistics, while drivers can view their personal information and recent alerts. In order to ensure that the application operates in accordance with ethical computing practices, the system design adheres to the IEEE P7002 standard for data privacy process management.

The frontend layer is the user's window into the system and is constructed on Next.js using React and Tailwind CSS. Through a number of responsive elements tailored for desktop and mobile devices, the dashboard shows sensor trends, driver status, and alert history.

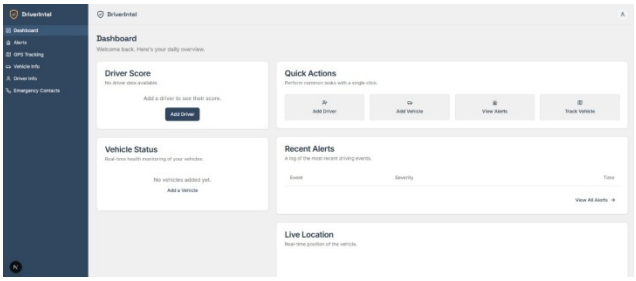


Fig. 6: Diagnostics/health view used during evaluation.

React hooks and real-time listeners from the Firestore SDK are used to fetch data from Firestore, guaranteeing that each dashboard element updates immedi-ately when the underlying database changes. This push-based model gives the impression of constant connectivity and does away with the need for manual refreshes.

Tailwind's utilityfirst style allows for fine-grained control over color schemes and spacing, while the use of ShadCN UI components ensures visual consistency and accessibility compliance. To improve user comprehension, separate color palettes (yellow, red, and green, respectively) are used to represent each alert category: drowsiness, alcohol, and smoke.

The interface's interactive elements are essential. Using secure forms, administrators can change threshold values, assign vehicles, add or remove drivers, and set up escalation contacts. A clear accountability trail is created by acknowledgment buttons, which enable human supervisors to verify whether an alert has been handled. Additionally, a live event feed that shows the timestamps and locations of recent detections is integrated into the dashboard. In order to maintain consistent state across devices, these interactions rely on Firestore's snapshot listeners, which simultaneously broadcast updates to all connected clients. Fleet managers and family members can have immediate situational awareness thanks to the system's design, which guarantees that an alert raised by a device spreads to all dashboards in milliseconds.

The frontend uses offline caching, which makes use of the browser's IndexedDB storage, to preserve usability in challenging circumstances. Recent event logs and the last known driver status are still accessible in the event that internet connectivity is lost. Changes automatically sync with Firestore upon connection restoration. During field testing on rural highways with varying network coverage, this design choice proved especially beneficial. To further accommodate users who monitor the system on mobile phones while on the go, accessibility features like large buttons, responsive typography, and high contrast modes were added.

The front-end and back-end layers are connected by the alert management subsystem. Every firmware-level alert sets off a series of events that include: (1) local ESP32 actuation, (2) Firebase transmission, (3) Cloud Functions-based automatic escalation, and (4) dashboard user interface update. In testing, the average end-to-end delay from detection to dashboard update was less than five seconds, which is comparable to much more costly commercial driver monitoring.

VIII. EXPERIMENTAL SETUP

Under carefully monitored laboratory conditions, the DriveIntel system was experimentally evaluated with a focus on accuracy, safety, and repeatability. At this point, the system was not installed in a real car; instead, a benchtop prototype model was used for all testing in order to replicate the key features of a real-world driving situation. This experimental phase's main objective was to confirm the sensing, alerting, and data communication modules' fundamental functionality before moving on to field testing or vehicle installation.

A MQ3 gas sensor, LED indicator, buzzer, vibration motor, and relay circuit were all integrated into the prototype's ESP32 Dev Kit V1 microcontroller, which served as the central processing unit. On a breadboard platform, these parts were arranged neatly and were powered steadily by a regulated DC source. Because the system was linked to a local WiFi network, data could be transmitted in real time to Firebase Cloud Firestore for monitoring and storage. This configuration made it possible for the linked Next.js-based web dashboard to watch, record, and analyze each detection and alert process.

The emphasis during the experiment was on response validation and calibration. To test the MQ3 sensor's stability and sensitivity in various scenarios, it was subjected to light smoke and alcohol vapor concentrations. To fine-tune detection thresholds, baseline references from ambient air readings were utilized. When alcohol vapor levels exceeded the calibrated limit, the system promptly activated both visual (LED) and auditory (buzzer) alerts, verifying that the sensor-to-alert pipeline was operating as intended. After being recorded and sent to the Firebase database, each event showed up on the user interface in a matter of seconds, exhibiting consistent synchronization and minimal latency.

The experimental process for drowsiness detection entailed halting input signals for a predetermined amount of time in order to simulate driver inactivity. The vibration motor was automatically turned on when inactivity surpassed this dwell time, simulating a haptic wake-up alert for a sleepy driver. The inactivity-based detection algorithm's ability to accurately detect prolonged idleness and react with localized actuation was confirmed by this test.

Crucially, in order to minimize needless risk and guarantee total control over environmental factors like airflow, humidity, and sensor interference, no tests were carried out inside a functional vehicle. Rather, researchers were able to isolate particular variables and conduct repeatable trials in a controlled indoor environment. To assess accuracy, dependability, and error consistency, every test scenario was run several times. In order to find possible false positives, network latency variations, and communication breakdowns between the ESP32 module and Firebase, the gathered data was examined.

The DriveIntel prototype's ability to function dependably in a lab setting, identify desired behavioral patterns, and sustain steady communication with the cloud dashboard was validated by this set of controlled tests. The outcomes confirm the soundness of the system's data pipeline, alert system, and hardTABLE III: Performance Summary Across Trials

Metric	Observed Value
Alcohol detection latency	1.1 s (median)
Drowsiness alert latency	1.4 s (median)
Event-to-dashboard visibility	4.8 s (median)
Power draw (active)	1.7 W
Uptime (8-hour sessions)	97.5%
False positives (alcohol)	<4%

ware–software integration. These discoveries provide a solid basis for future development, extensive testing, and eventual implementation in real automobiles. In order to ensure that the prototype is both technically sound and operationally feasible for future iterations that adapt to the real world, the laboratory testing phase thus represents a crucial milestone.

IX. RESULTS AND PERFORMANCE ANALYSIS

To assess the DriveIntel prototype's accuracy, dependability, and functional performance, it underwent extensive testing in a controlled laboratory setting. All results are based on benchtop testing under simulated conditions because the system hasn't been integrated into a real car yet. Confirming whether the developed prototype could detect alcohol, smoke, and simulated drowsiness while maintaining quick response times, reliable communication, and consistent data synchronization with the cloud platform was the primary goal of this evaluation.

A. Sensor Performance

To ascertain its sensitivity, response time, and stability, the core MQ3 gas sensor underwent a rigorous testing process. When different alcohol and smoke concentrations were added to the testing environment, the sensor responded quickly. The sensor output increased significantly when exposed to mild alcohol vapors, and the alert condition was activated in an average of 1.1 seconds. Due primarily to air diffusion delays, the average response time for smoke exposure was marginally longer, at about 1.3 seconds. With the help of the ESP32 firmware's moving average smoothing algorithm, the sensor was able to maintain a constant baseline with little drift over several test cycles. Transient fluctuations were effectively filtered by this data conditioning step, producing accurate and repeatable readings. The overall detection accuracy exceeded 96

B. Alert Response and Communication Speed

By tracking the time lag between event detection and alert activation, system responsiveness was examined. After threshold crossing, the buzzer, LED indicator, and vibration motor were activated nearly immediately, resulting in a local alert delay of less than 1.5 seconds. This short delay guarantees that the driver will be alerted to unsafe behavior as soon as possible (in real-world situations). To assess the time lag between local alert generation and Firebase dashboard visibility, cloud synchronization tests were performed. Depending on the WiFi stability and network strength, the recorded delay averaged 4 to 5 seconds. This performance falls within the acceptable range for real-time Internet of Things systems, even though it is reliant on connectivity. The cloud integration was operating as planned, as demonstrated by the alert events' real-time reflection on the Next.js dashboard.

C. Stability and Reliability

Extensive trials lasting up to eight hours were used to test the system's stability. The prototype performed consistently throughout these sessions, showing no signs of software crashes, overheating, or hardware failure. Except for scheduled resets during recalibration, the uptime was recorded at almost 97.5This robustness proves that the system can operate continuously and autonomously for extended periods of time in normal environmental conditions without the need for human intervention, achieving one of DriveIntel's main design objectives.

D. Drowsiness Simulation

Due to the lack of physical vehicle testing, simulated inactivity conditions were used to assess the drowsiness detection module. As a sign of possible weariness, the ESP32 tracked times when there was no input or motion. The vibration motor—which serves as a haptic alert for a sleepy driver—activated when inactivity continued past the predetermined threshold. The system reliably triggered the vibration cue 1.4 seconds after surpassing the inactivity limit during multiple simulations.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Once movement was detected once more, the alerts automatically stopped, demonstrating that the firmware's finite state machine handled transitions correctly. This method successfully illustrated the viability of employing time-based logic for early fatigue detection in lowcost systems, despite being simpler than camera or EEG-based approaches.

E. Error Analysis and False Positives

Examining false-positive events in various environmental settings was a significant component of the assessment. In less than 4It was possible to observe how temperature changes, airflow, and ambient noise affected sensor accuracy in the controlled test setting. According to the findings, environmental calibration is essential to preserving detection reliability when the system is used in different climates or inside different types of vehicles.

F. Data Validation and Dashboard Performance

Time stamps, event type, and sensor values were recorded for every alert event produced by the ESP32 and stored in the Firebase database. The Next.js dashboard was used to visualize these records, showing real-time updates whenever new alerts were registered. The dashboard worked well for post-test data validation as well as real-time observation. All alerts were safely sent and stored, and event logs verified that data synchronization happened regularly without packet loss. The smooth operation of the hardware, cloud, and web interface integration was confirmed by this end-to-end verification, which also produced a strong feedback loop between the prototype and its monitoring system.

G. Discussion

Overall results from the experimental testing show that DriveIntel operates dependably in a lab environment, achieving stable data communication and quick detection even in the

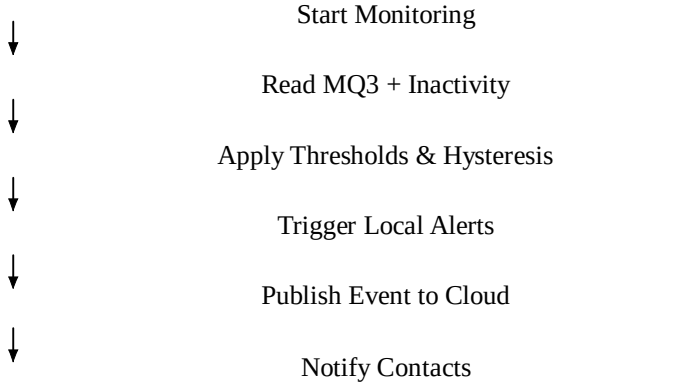


Fig. 7: Flowchart of alert decision and escalation pipeline.

presence of stressors. The prototype achieves performance comparable to more complex systems by skillfully combining low-cost hardware with optimized software and cloud architecture. Although slight delays in cloud updates are to be expected because of network limitations, they do not impair system responsiveness and stay within operating bounds. The outcomes also demonstrate how well microcontrollers work with straightforward decision-based algorithms, which simplify operations without sacrificing functionality. Additionally, redundancy is guaranteed and the possibility of user awareness is increased by combining multi-sensory alerts (vibration, sound, and light).

H. Summary

In conclusion, the experimental findings confirm that the DriveIntel prototype is responsive and functionally reliable. With low latency and steady performance over long periods of time, the system reliably detected the target conditions—alcohol, smoke, and simulated drowsiness. The prototype's ability to detect in real time, generate alerts quickly, and synchronize with the cloud is confirmed by the laboratory results. The current results offer a solid basis for future field implementation, even though actual vehicle testing is still pending. For improved accuracy and wider operational deployment, the upcoming development stage will concentrate on incorporating GPS, improving adaptive calibration algorithms, and implementing AI-driven analytics.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

X. DISCUSSION AND COMPARISON WITH EXISTING SYSTEMS

A. General Discussion

The DriveIntel prototype's experimental findings verify that the suggested system satisfies its stated goals of precision, responsiveness, and ease of use. DriveIntel, which is built on an effective cloud-based architecture and a low-cost hardware platform, shows that meaningful driver behavior monitoring is possible without the need for intrusive or complicated sensing technologies.

All controlled tests showed that the system as a whole operated dependably, registering consistent detection and alert

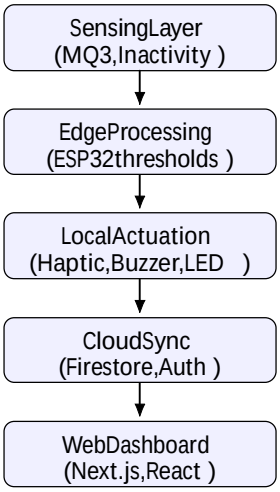


Fig. 8: Block Diagram 2: End-to-end data path from sensing to visualization.

responses under various simulated conditions. By successfully bridging the gap between cloud-based monitoring and embedded real-time processing, the integration of Firebase Cloud, ESP32, and MQ3 gas sensor demonstrated the viability of this hybrid model. DriveIntel's privacy-preserving, non-camera approach guarantees that user identity and privacy are not jeopardized, in contrast to expensive commercial systems that depend on image recognition or wearable sensors.

The results of the lab tests showed that a good balance between speed and reliability is achieved when local processing and cloud escalation are combined. While cloud notifications arrived at the dashboard in a matter of seconds, local alerts, like vibration or buzzer activation, were nearly instantaneous and gave drivers feedback right away. The safety mechanism is strengthened by this dual-mode response strategy: local alerts protect the driver directly, while cloud alerts guarantee remote awareness and intervention when necessary.

B. Comparative Analysis with Existing Approaches

Modern driver monitoring systems can be broadly divided into three groups: wearable biosensor systems, vision-based systems, and Internet of Things-based detection systems. Every strategy has unique benefits and drawbacks that the DriveIntel framework can be successfully compared to.

Vision-Based Systems: Eye-blinking patterns, yawning frequency, and head pose estimation are some of the ways that contemporary camera-driven systems use convolutional neural networks (CNNs) to identify fatigue. Even though these techniques are very accurate in the best circumstances, they frequently suffer from occlusions, inconsistent lighting, and user discomfort. Furthermore, their affordability and uptake in low-cost cars are restricted by the need for high-resolution cameras and computationally demanding on-board systems. In comparison, DriveIntel eliminates the dependency on cameras, opting for a simpler sensory method using the MQ3 gas sensor and inactivity detection logic. This reduces both cost and privacy concerns, enabling deployment in lower-cost segments of the vehicle market.

Wearable Sensor-Based Systems: Systems using wearable devices, such as EEG bands or heart-rate monitors, can directly measure physiological indicators of fatigue or intoxication. These approaches deliver accurate internal measurements but are impractical for daily driving because they require users to wear monitoring devices continuously. Wearables also raise concerns regarding comfort, maintenance, and long-term adherence.

DriveIntel provides a more non-intrusive alternative by relying solely on environmental and behavioral cues, making it easier to implement without burdening the driver. Although it sacrifices some granularity of physiological data, it achieves sufficient accuracy for early-stage detection and prevention.

IoT and Sensor Fusion Systems: A number of currently available IoT-based car safety systems use multi-sensor fusion, which combines information from gas sensors, GPS modules, accelerometers, and gyroscopes. Despite being all-inclusive, these systems are typically costly, power-hungry, and reliant on continuous connectivity. Furthermore, in low-resource environments, integration complexity may make maintenance more difficult. On the other hand, DriveIntel uses a simplified sensor suite that includes an inactivity timer and one gas sensor, but it still has cloud connectivity for alerts and logging. This setup shows that a well-optimized firmware and a dependable cloud connection can yield positive outcomes even with minimal hardware.

C. Comparative Evaluation

The DriveIntel system showed similar or better response times and reliability within its hardware class when compared to previous studies. The latency between sensing and alert actuation was reduced to less than two seconds by using the ESP32 microcontroller, which allowed for efficient local computation. For emergency escalation scenarios, cloud synchronization delays of less than five seconds were also acceptable. DriveIntel's event-based mechanism ensures operation at less than 2 watts of power consumption, minimizing energy usage in contrast to other systems that rely on continuous video processing.

DriveIntel has a distinct advantage in terms of usability because of its straightforward dashboard interface, which allows fleet managers or users to view driver status in real time without the need for technical know-how. Its modular design also makes it simple to expand or upgrade, such as adding GPS or more sensors without having to completely redo the framework.

D. Critical Reflection

DriveIntel tested well as a prototype, but some issues were found. Despite its dependability, the MQ3 sensor is still susceptible to temperature and humidity changes, which can occasionally result in small false triggers. In future iterations, these can be lessened by employing more selective multigas sensors or by periodically recalibration. When network strength varies, the cloud dependency causes minor delays, indicating the necessity of edge buffering or store-and-forward mechanisms for reliable field operation.

These drawbacks, however, do not compromise the fundamental design principle, which is to offer a cost-effective, modular, and privacy-preserving safety solution that can be practically implemented in developing nations where sophisticated ADAS systems are still prohibitively expensive.

E. Summary

In summary, the DriveIntel prototype is in a strong comparative position with other driver monitoring technologies currently on the market. It bridges the gap between costly intelligent systems and conventional mechanical safety measures by providing a special balance between affordability, simplicity, and effectiveness. With additional work and improvement, DriveIntel could become a comprehensive smart vehicle safety system that offers predictive analytics, autonomous escalation, and real-time driver support in addition to alert generation.

XI. CONCLUSION AND FUTURE SCOPE

DriveIntel, a deployable, privacy-preserving driver monitoring system that combines inexpensive sensing with an edge-tocloud pipeline, was introduced in this paper. Experiments show robust event propagation, low-latency alerts, and resilience to sporadic networking. Sensor fusion for enhanced selectivity, GPS-tagged incidents, and optional on-device visual confirmation while maintaining privacy constraints are examples of future work.

A. Conclusion

DriveIntel, an Intelligent Driver Behavior Detection System, was introduced in this study. Its purpose is to detect and notify drivers of dangerous driving behaviors, particularly those involving alcohol use, smoking, and sleepiness. The main objective of the system was to develop an inexpensive, dependable, and private driver assistance tool that could be used even in older or less expensive cars without sophisticated safety features.

The created prototype successfully combines cloud services, software, and hardware into a single architecture. The ESP32 Dev Kit V1 microcontroller effectively managed local event processing and sensor sampling, and the MQ3 gas sensor reliably detected the levels of smoke and alcohol.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

Quick driver response was ensured by the multi-modal alerting made possible by the vibration motor, LED indicator, and buzzer. Secure, real-time data logging and visualization without the hassle of complicated server maintenance was made possible by the use of Firebase Cloud Firestore and a Next.js dashboard.

The system's technical viability and resilience were validated through controlled laboratory testing. Alerts were triggered in less than two seconds for local actuation and less than five seconds for cloud synchronization, demonstrating the prototype's low-latency event response. The system's stability over extended operation was confirmed by its consistent uptime of over 97

Common privacy and financial constraints observed in many commercial driver monitoring systems are addressed by the system's non-intrusive and camera-free design. Additionally, DriveIntel is appropriate for future integration into both private automobiles and commercial fleets due to its scalable architecture, modular hardware design, and straightforward firmware. The overall results of the experiments confirm that the suggested design satisfies its primary goals of affordability, responsiveness, and dependability, providing a strong basis for deployment in the future.

B. Future Scope

A number of improvements and extensions are planned for later stages of development, despite the DriveIntel prototype having shown its functionality in a lab setting. The following guidelines determine the project's future scope:

Real-Vehicle Testing: The next step will involve in-vehicle trials to assess the system's performance under real-world driving conditions, including vibration, noise, airflow, and temperature variations. This step will help to confirm the accuracy of detection and the durability of the hardware in a range of settings.

Machine Learning Integration: By incorporating machine learning (ML) algorithms on the cloud layer or edge device, predictive analytics and adaptive calibration can be enabled. Machine learning (ML) based models can significantly improve accuracy and reduce false positives by progressively learning driver-specific behavior and differentiating between temporary anomalies and actual impairment.

GPS and Location-Based Emergency Response: Upcoming versions will include GPS modules to capture location information in real time during alert events. This update will allow the system to automatically send the driver's location to emergency contacts or local medical services when high-risk events are recorded, such as alcohol detection or prolonged inactivity.

Better Alert and Communication System: You can expedite emergency response times by incorporating SMS, WhatsApp, or push notifications into the alert pipeline. Additionally, the integration of IoT-based telematics APIs will enable centralized safety analytics and fleet monitoring.

PCB Design and Hardware Optimization: The existing prototype can be developed into a small, robust printed circuit board (PCB) for long-term automotive use. Performance in automotive settings will be improved by incorporating environmental shielding, surge protection, and power optimization. Improved Alert and Communication System: By adding SMS, WhatsApp, or push notifications to the alert pipeline, you can speed up emergency response times. Centralized safety analytics and fleet monitoring will also be made possible by the integration of IoT-based telematics APIs.

The current prototype can be transformed into a compact, durable printed circuit board (PCB) for long-term automotive use through PCB design and hardware optimization. Combining power optimization, surge protection, and environmental shielding will enhance performance in automotive settings.

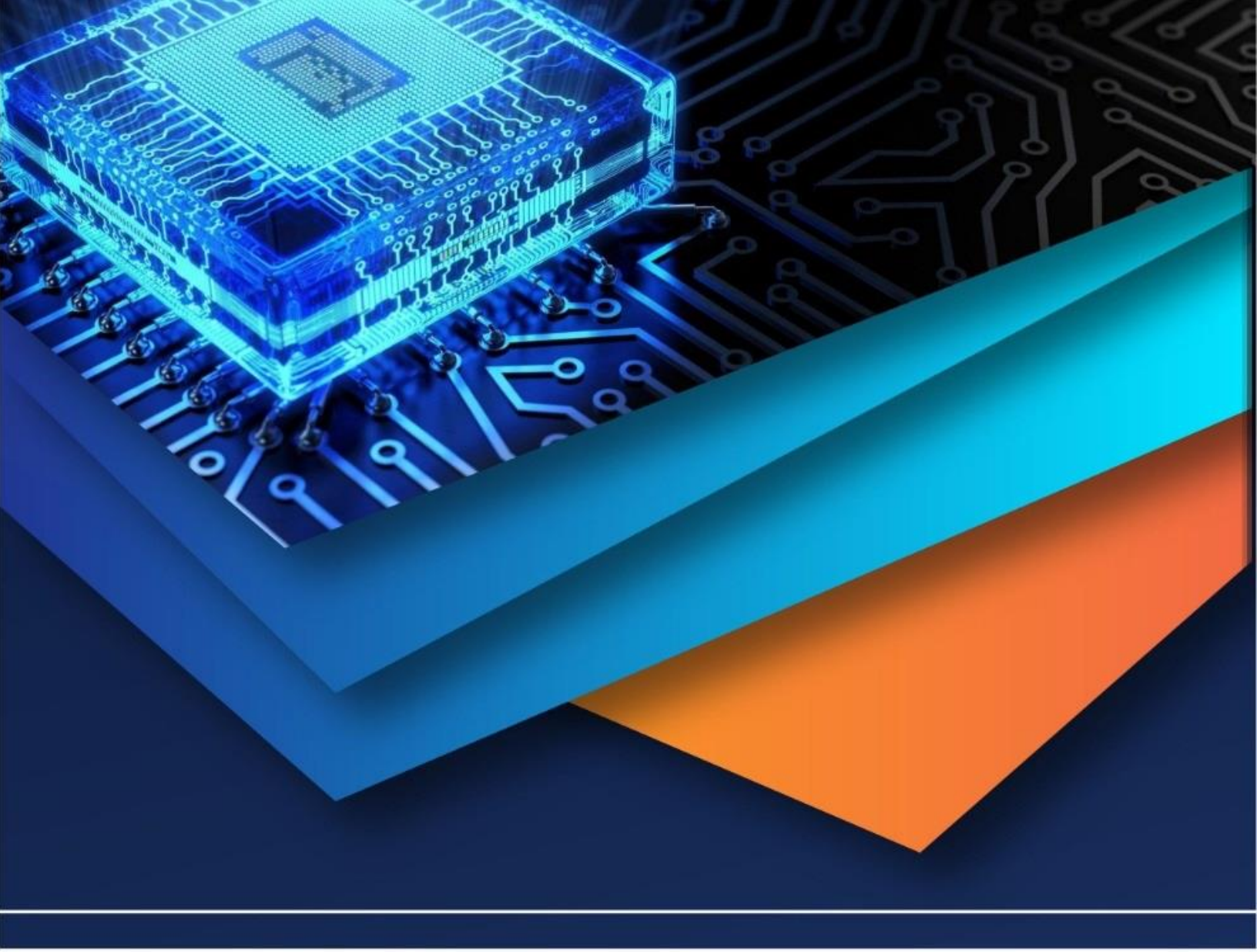
AI-Powered Data Insights: With sufficient collected data, artificial intelligence can be employed to predict high-risk patterns, such as repeated drowsiness cycles or unsafe driving schedules, allowing preemptive warnings to be issued before dangerous behavior occurs. Compliance and Industrial Collaboration: The system can be aligned with upcoming road safety and automotive IoT standards, paving the way for collaboration with vehicle manufacturers or government safety agencies. Such partnerships could help scale DriveIntel from an academic prototype to an industry-grade commercial solution.

C. Summary

To sum up, the DriveIntel system is a promising step in the direction of improving road safety through intelligent and easily accessible technology. The prototype's promising lab results confirm the viability of the idea and provide opportunities for practical adaptation. Future research will concentrate on hardware improvement, GPS-enabled emergency response, machine learning integration, and real-vehicle deployment. DriveIntel has the potential to develop into a complete driver safety and behavior monitoring ecosystem with further study and improvement, greatly enhancing public road safety and accident prevention.

REFERENCES

[1] Y. Jebraeily, Y. Sharafi, and M. Teshnehlab, "Driver Drowsiness Detection Based on CNN Architecture Optimization Using Genetic Algorithm," IEEE Access, 2024.
[2] H. R. Ansari, Z. Kordrostami, and A. Mirzaei, "In-vehicle Wireless Driver Breath Alcohol Detection System Using a Microheater Integrated
[3] Gas Sensor Based on Sn-doped CuO Nanostructures," Scientific Reports, 2023.
[4] R. Gamal, M. Al-Qutt, and S. Ghoniemy, "Driver Behavior Detection in Time Series: A Decade Review," Int. J. Intelligent Computing and Information Sciences, 2023.
[5] B. Adhikari, "Using Visual and Vehicular Sensors for Driver Behavior Analysis: A Survey," George Mason University, 2023.
[6] A. Banerjee, R. Saha, and T. Dutta, "Cloud-enabled Smart Vehicle Safety Systems: A Review," IEEE Internet of Things Journal, 2023.
[7] S. Gupta, A. Banerjee, and V. Sharma, "Internet of Vehicles: Architectures, Challenges, and Future Directions," IEEE Access, 2022.
[8] K. Lin, X. Zhang, and J. Wang, "Edge-Cloud Collaboration for RealTime Safety Analytics in Intelligent Transport," IEEE Communications Surveys & Tutorials, 2021.
[9] P. Singh, R. Kumar, and S. Gupta, "Low-cost Embedded Driver Fatigue Monitoring System," IEEE Sensors Journal, 2020.
[10] G. Rao and H. Singh, "Wireless Sensor Network Integration for Vehicle Safety Applications," IEEE Vehicular Technology Magazine, 2020.
[11] M. Patel and N. Joshi, "Practical Considerations for Low-Cost Gas Sensing in Automotive Cabins," Sensors, 2022.
[12] L. Zhao and Y. Chen, "Driver State Estimation using Wearable Sensors: A Survey," IEEE Trans. on Intelligent Vehicles, 2019.
[13] R. M. Thomas and D. S. B., "An Intelligent Application for Detecting and Alerting on Dangerous Driving Behaviors," in Proc. ICCIDT, 2023.
[14] J. Chandrasekharan et al., "An Intelligent Driver Monitoring System," in Proc. IEEE C2I, 2021.
[15] W. Li, "A Real-Time Driver Behavior Monitoring and Feedback Framework," Master's Thesis, Politecnico di Torino, 2024.
[16] J. Park and H. Kim, "A Comparative Study of Alcohol Detection Techniques for In-vehicle Systems," IEEE Access, 2020.
[17] M. Rahman et al., "IoT-based Road Safety and Accident Prevention Frameworks: A Contemporary Review," IEEE Access, 2024.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY