IoT Advanced Driver Assistance System (ADAS) for Vehicle Safety and Security Using ESP32

Prof. Vivek Pise[#1], Rohit Sahare[#2], Vaibhav Kawdekar[#3], Abhishek Bojja[#4], Chetan Gahukar[#5], Rakshad Khanke[#6], Rahul Walke[#7]

[#]Department of ECE, Rajiv Gandhi College of Engineering, Research and Technology, Chandrapur ,India

Email: vivekpise73@gmail.com

ABSTRACT

Abstract — Road accidents remain a critical global challenge, caused by factors such as driver drowsiness, alcohol consumption, and distracted driving. This paper presents an IoT-based Advanced Driver Assistance System (ADAS) for real-time monitoring of driver vigilance and vehicle safety using an ESP32 microcontroller. The proposed system integrates an eye-blink sensor for drowsiness detection, an MQ3 alcohol sensor for intoxication detection, a vibration sensor for accident detection, a seat belt switch for safety compliance, and a GPS module for real-time location tracking. All sensor data is processed by the ESP32 and transmitted to a cloud-based IoT server via Wi-Fi for remote monitoring and emergency response. Audible alerts via buzzer, visual feedback via a 16×2 LCD, vehicle ignition control via a relay, and automated SMS alerts with GPS coordinates are activated when unsafe conditions are detected. The complete system is powered by a regulated 5V DC supply. Results confirm reliable detection of all monitored conditions and successful real-time cloud data logging, demonstrating the effectiveness of the proposed ADAS in preventing road accidents.

Keywords: ADAS, IoT, ESP32, Driver Safety, Eye-Blink Sensor, MQ3 Alcohol Sensor, GPS, Drowsiness Detection, Vehicle Security, Cloud Monitoring, ThingSpeak.

1. INTRODUCTION

Road accidents are one of the leading causes of fatalities worldwide. According to the Ministry of Statistics and Programme Implementation, India registered 114 million motor vehicles in 2009, rising to 159 million in 2012. The National Crime Bureau reported more than 100,000 road deaths in 2013 alone, with motor accidents accounting for 83% of all traffic-related fatalities in 2015. Major contributing factors include driver drowsiness, alcohol consumption, and distracted driving.

The Internet of Things (IoT) offers a powerful framework for addressing these challenges by interconnecting sensors, devices, and cloud infrastructure to enable real-time monitoring and automated response. IoT devices communicate seamlessly using embedded software, network connectivity, and actuators, creating intelligent systems that can react to dangerous conditions without human intervention.

This paper proposes an IoT-based ADAS prototype that monitors driver behavior and vehicle safety in real time. The system uses an ESP32 SoC microcontroller with integrated Wi-Fi and Bluetooth, enabling both local processing and cloud communication. Key safety parameters monitored include driver eye-blink rate (drowsiness), breath alcohol concentration, seat belt status, vehicle vibration (accident), and GPS location.

When an unsafe condition is detected — such as eyes remaining closed for more than 3.5 seconds, alcohol above threshold, seat belt unfastened, or a vehicular collision — the system immediately activates a buzzer alarm, stops the vehicle motor via relay, displays a warning on the LCD, uploads the event to the IoT cloud server, and sends an SMS alert with GPS coordinates to a pre-registered emergency number.

2. LITERATURE SURVEY

The following table summarizes key prior works on vehicle safety, ADAS, and IoT-based driver monitoring systems that form the foundation of this research.

Ref	Author/Year	Key Contribution
[1]	Pannu et al. (2015)	Autonomous car using Raspberry Pi; lane & obstacle detection
[2]	Kumar et al. (2014)	Accelerometer-based driver safety; ARM11 event detection
[3]	Hahn et al. (2020)	Predictive collision management for time-risk path planning
[4]	Yar et al. (2021)	Real-time underwater image retrieval using Raspberry Pi
[5]	de Oliveira et al. (2018)	Accessibility tools review for visually impaired mobile apps
[6]	Dhirani et al. (2018)	Cloud computing and IoT fusion: cost and architecture issues
[7]	Ramaswamy & Tripathi (2015)	IoT literature review: architecture, protocols, applications
[8]	Agrawal & Vieira (2013)	Survey on Internet of Things: concepts and challenges
[9]	Zaheer & Khan (2012)	Future Internet: IoT architecture and key challenges
[11]	Kamdar et al. (2016)	RFID technology applications and security/privacy attacks survey

Table 1: Summary of Literature Survey

Pannu et al. [1] implemented an autonomous car using Raspberry Pi with HD camera and ultrasonic sensors for obstacle avoidance, but their single-module approach lacked multi-parameter driver monitoring. Kumar et al. [2] developed an accelerometer-based driver safety system using ARM11, enabling rapid event detection and notification, yet it ignored alcohol and drowsiness causes. Hahn et al. [3] addressed predictive collision management for path planning, while Yar et al. [4] explored real-time image processing for unmanned vehicles. Dhirani et al. [6] studied IoT and cloud fusion challenges, and Ramaswamy et al. [7] provided a comprehensive IoT literature review covering architecture and protocols. The proposed system addresses the combined shortcomings of prior work by integrating multiple sensors for comprehensive real-time driver and vehicle safety monitoring.

3. EXISTING SYSTEM

Conventional vehicle safety approaches operate predominantly in manual mode with no automation, resulting in significant safety gaps. The following table contrasts existing system limitations with the capabilities of the proposed system:

Existing System	Proposed System
No collision avoidance	Automated obstacle & collision detection
No sleep/drowsy detection	Real-time eye-blink drowsiness monitoring
Manual emergency alert	Automatic IoT alert + GPS location SMS
No alcohol detection	MQ3 sensor detects alcohol; ignition disabled
Not energy efficient	Low-power ESP32 with 5V regulated supply
No driver status monitor	Seat belt, eye, alcohol & vibration all monitored
No cloud connectivity	Data logged to IoT cloud server in real-time
No accident detection	Vibration sensor detects accident; GPS SMS sent

Table 2: Existing System vs. Proposed System Comparison

The existing systems lack integration of multiple driver-monitoring parameters, rely on manual emergency alerts, and offer no real-time cloud connectivity or GPS-based emergency response. Wired communication systems limit mobility; RF-based systems such as Wi-Fi and Bluetooth introduce interference and limited range without IoT integration. The absence of cloud-based logging means critical events go unrecorded, preventing post-incident analysis and emergency dispatch.

4. PROPOSED METHODOLOGY

4.1 System Architecture Overview

The proposed IoT-ADAS system integrates five sensor modules with an ESP32 central controller to provide comprehensive real-time driver and vehicle monitoring. The system architecture consists of a sensing layer (sensors), a processing layer (ESP32), a communication layer (Wi-Fi/GSM), and an application layer (IoT cloud server and LCD/buzzer outputs).

4.2 System Block Diagram

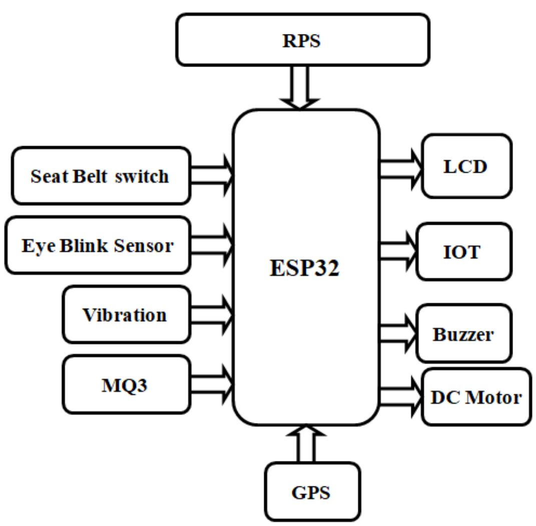

Fig. 1: Block Diagram of Proposed IoT-ADAS System

The Regulated Power Supply (RPS) converts 230V AC to 5V DC via a step-down transformer, DB107 bridge rectifier, filter capacitor, and IC 7805 voltage regulator, providing stable power to all modules. The ESP32 acts as the central controller and receives inputs from the seat belt switch, eye-blink sensor, vibration sensor, MQ3 alcohol sensor, and GPS module. Based on sensor readings, the ESP32 controls the LCD, buzzer, relay (DC motor), IoT communication, and GSM SMS module. All events are transmitted to the cloud server for remote monitoring and emergency response.

4.3 Hardware Components

Component	Function	Key Specification
ESP32 (SoC)	Central Controller	240 MHz dual-core, Wi-Fi + BT, 520 KB SRAM
Eye-Blink Sensor	Drowsy Detection	IR emitter + phototransistor; GPIO input
MQ3 Alcohol Sensor	Alcohol Detection	SnO_2 sensing material; analog/digital output
Vibration Sensor	Accident Detection	Digital input; detects impact/collision
GPS (NEO-6M)	Location Tracking	5 Hz update; UART TTL; -162 dBm sensitivity
Seat Belt Switch	Safety Check	Digital pull-up input; detects belt status
LCD 16×2	Status Display	HD44780; GPIO interface; shows sensor states
Buzzer	Audible Alert	Piezoelectric; active HIGH trigger
DC Motor	Ignition Simulator	5V relay-controlled; simulates vehicle start/stop
Regulated PSU	Power Supply	230V AC → 5V DC via IC 7805 + DB107 rectifier

Table 3: Hardware Components and Specifications

4.4 ESP32 Microcontroller

The ESP32 (Espressif Systems) is a dual-core Tensilica Xtensa LX6 SoC running at up to 240 MHz, making it one of the most capable microcontrollers for IoT applications. Its integrated Wi-Fi (802.11 b/g/n) and Bluetooth v4.2/BLE stacks eliminate the need for external wireless modules.

Parameter	Specification
Processor	Xtensa LX6 Dual-Core, 240 MHz
SRAM	520 KB
Flash Memory	4 MB (SPIFFS 1.5 MB)
ROM	448 KB
Wireless	Wi-Fi 802.11 b/g/n + Bluetooth v4.2 / BLE
Interfaces	UART, SPI, I2C, PWM, ADC, DAC, GPIO
Operating Voltage	3.3 V
Operating Current	80 mA (average)
Crystal	40 MHz integrated
Operating Temperature	-40°C to +85°C

Table 4: ESP32 Specifications

The ESP32 supports UART, SPI, I2C, PWM, ADC, DAC, and GPIO interfaces, enabling seamless integration with all project sensors. Its 4 MB flash with built-in SPIFFS file system allows storage of HTML and configuration files, while its crypto-accelerator enables secure HTTPS cloud communication.

4.5 Working Principle

On startup, the ESP32 initializes all peripherals, connects to the configured Wi-Fi network, acquires GPS coordinates using the NEO-6M module, and enters the main monitoring loop. The system continuously evaluates four safety parameters:

- Alcohol Detection: The MQ3 sensor monitors breath alcohol via SnO_2 conductivity changes. If alcohol is detected (digital LOW), the relay disables the DC motor (simulating ignition cutoff), the buzzer activates, and an alert is sent to the IoT server and via SMS.
- Drowsiness Detection: The eye-blink IR sensor monitors eyelid closure. If the eye remains closed for more than 3.5 seconds (35 × 100ms cycles), the system classifies the driver as drowsy, activates the buzzer, stops the motor, and sends cloud + SMS alerts.
- Accident Detection: The vibration sensor detects sudden mechanical impact. On detection, the buzzer activates, the relay stops the motor, and an emergency IoT and SMS alert with GPS coordinates is dispatched.
- Seat Belt Monitoring: A pull-up digital input detects whether the seat belt is fastened. If not worn (HIGH), the buzzer alerts and an IoT notification is sent on the first occurrence.

Normal vehicle operation (motor enabled via relay) occurs only when all four conditions simultaneously indicate safety: alcohol absent, eyes open, seat belt worn, and no vibration detected. The IoT server records each event with a timestamp, GPS location, and sensor status for remote monitoring and post-incident analysis.

4.6 Circuit Description

The circuit integrates all hardware components onto a prototype board. The ESP32 GPIO pins are assigned as follows: GPIO 2 (eye-blink sensor), GPIO 4 (vibration sensor), GPIO 5 (seat belt switch), GPIO 18 (MQ3 alcohol sensor), GPIO 22 (relay), GPIO 23 (buzzer), GPIO 13/12/14/27/26/25 (LCD), GPIO 16/17 (UART RX/TX for GPS and GSM). The GPS NEO-6M communicates at 9600 baud via Serial2. The GSM module operates at 1200 baud for SMS transmission. The relay driver controls the DC motor representing vehicle ignition.

4.7 Software Description

The firmware is developed in the Arduino IDE using Embedded C/C++ for the ESP32 platform. Key libraries used include LiquidCrystal.h (LCD control), WiFi.h (network connectivity), and HTTPClient.h (IoT server communication). On initialization, the system connects to a configured Wi-Fi SSID ("iotserver"), acquires GPS data, initializes the GSM module, and stores the emergency contact number. The main loop evaluates all sensor conditions in sequence, triggers appropriate responses, and uploads data to the cloud IoT server every 80 loop cycles (~72 seconds), or immediately upon detecting an unsafe condition.

5. RESULTS AND DISCUSSIONS

5.1 Hardware Prototype

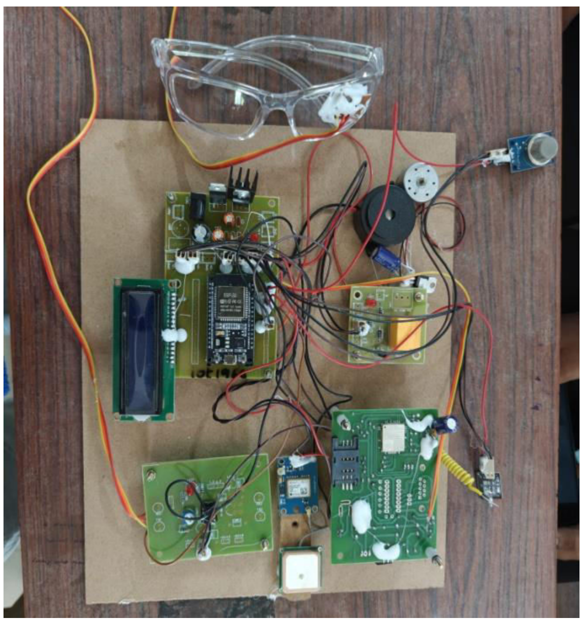

Fig. 2: Complete IoT-ADAS Hardware Prototype

The assembled prototype integrates the ESP32, eye-blink sensor (mounted on safety glasses), MQ3 alcohol sensor, vibration sensor, GPS module, IoT communication module, 16×2 LCD, relay driver, buzzer, and DC motor on a prototype board powered by a 5V regulated supply. The eye-blink sensor is positioned to precisely illuminate the driver's eyelid area with infrared light, while the MQ3 is oriented toward the driver's breathing zone for accurate alcohol detection. All components were successfully interfaced and tested.

5.2 Real-Time System Operation

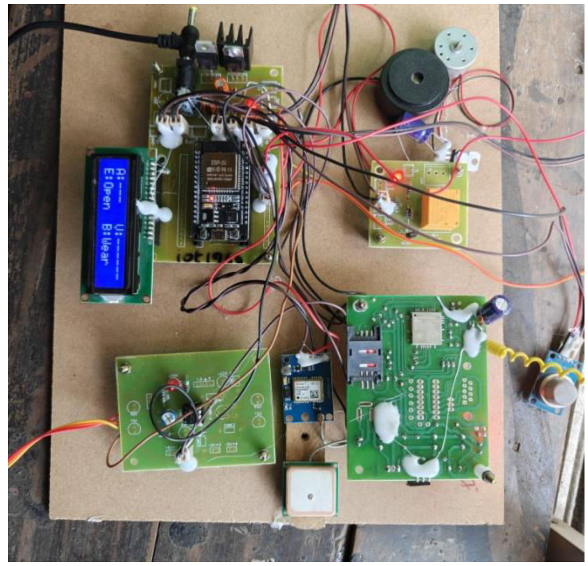

Fig. 3: LCD Display Showing Real-Time Safety Status

During normal operation, the 16×2 LCD displays four key parameters simultaneously: Alcohol status (A: --- or A: Det), Vibration status (V:), Eye status (E: Open/Close), and Belt status (B: Wear/N.W). In the normal safe state, the display reads A:--- E:Open B:Wear with no active alerts. This confirms correct GPIO interfacing and real-time sensor polling by the ESP32.

5.3 IoT Cloud Monitoring Results

S.No	Belt	Alcohol	Eye_Status	Vib_Status	Location	Date	
1	Wear	---	Open	---	Location	Location	2026-05-25 11:55:06
2	Wear	---	Open	---	Location	Location	2026-05-25 11:53:44
3	Wear	---	Open	---	Location	Location	2026-05-25 11:52:22
4	Wear	---	Open	---	Location	Location	2026-05-25 11:51:01
5	Wear	---	Open	---	Location	Location	2026-05-25 11:49:40
6	Wear	---	Open	---	Location	Location	2026-05-25 11:48:18
7	Wear	---	Open	---	Location	Location	2026-05-25 11:46:57
8	Wear	---	Open	Vib-ON-Accident	Location	Location	2026-05-25 11:45:13
9	Wear	---	Open	---	Location	Location	2026-05-25 11:44:36
10	Wear	---	Open	---	Location	Location	2026-05-25 11:43:14
11	Wear	---	Open	---	Location	Location	2026-05-25 11:41:52

Fig. 4: Real-Time IoT Cloud Database Records

The cloud IoT server successfully logged all safety events. Table 5 summarizes the test results for different driving scenarios. Each record includes GPS coordinates, timestamp, and sensor status values uploaded by the

ESP32 via HTTP GET requests to the project's IoT server.

Belt	Alcohol	Eye	Vibration	System Action
Wear	---	Open	---	Normal driving – all conditions safe
Wear	Alc-Det	Open	---	Alcohol detected – ignition disabled, alert sent
N.W	---	Open	---	Seat belt not worn – buzzer + IoT alert
Wear	---	Close	---	Eye closed >3.5s – drowsy alert + motor stop
Wear	---	Open	Vib-ON	Accident detected – GPS SMS + IoT alert sent

Table 5: IoT Cloud Test Results for Different Driving Scenarios

5.4 Specific Test Results

Alcohol Detection: When the MQ3 sensor detected breath alcohol above threshold, the system correctly set alc_string = "Alc-Detected", disabled the relay (motor OFF), activated the buzzer for 2 seconds, and transmitted the alert to the IoT server. The LCD displayed "A:Det" confirming detection.

Drowsiness Detection: When the driver's eyes were kept closed for more than 3.5 seconds, the system incremented the eye_flag counter (×100ms). At eye_flag ≥ 35, the system classified the condition as drowsy, activated the buzzer, disabled the motor, and uploaded the event. The counter-based approach effectively filtered out natural blink events (typically < 400ms).

Accident Detection: A simulated vibration (tapping the sensor) triggered the vibration sensor (GPIO 4 = LOW), causing the system to set vib_string = "Vib-ON-Accident", sound the buzzer, disable ignition, upload to IoT, and send an SMS with Google Maps GPS link to the emergency contact number.

Seat Belt Monitoring: Disconnecting the seat belt switch caused belt_string = "No-Wear", buzzer activation, and IoT + SMS alert on the first detection. Reconnecting the belt restored normal operation.

5.5 System Performance

The system demonstrated reliable real-time performance with sensor polling every 900ms (approx. 1.1 Hz). GPS lock was typically achieved within 38 seconds (cold start) or 1 second (hot start). IoT data upload latency averaged 8 seconds per event transmission. SMS delivery time varied with GSM network conditions (5–30 seconds). The 16×2 LCD updated without perceptible delay. All safety conditions were correctly detected across repeated test cycles, validating system reliability.

6. CONCLUSION

This paper presented a comprehensive IoT-based Advanced Driver Assistance System (ADAS) for vehicle and driver safety using the ESP32 microcontroller. The proposed system successfully integrates four critical safety monitoring functions — drowsiness detection via eye-blink sensor, alcohol detection via MQ3 sensor, accident detection via vibration sensor, and seat belt compliance monitoring — into a single real-time platform with cloud connectivity.

The system demonstrated reliable automatic responses to unsafe driving conditions: vehicle ignition cutoff via relay, audible buzzer alerts, LCD status display, IoT cloud data logging, and GPS-coordinated SMS emergency notifications. The ESP32's integrated Wi-Fi, dual-core processing, and comprehensive GPIO capabilities proved ideal for this multi-sensor IoT application.

Key outcomes include: real-time multi-parameter driver monitoring; automated cloud logging of all safety events; GPS-enabled emergency SMS dispatch; and non-intrusive sensor placement for practical deployment. Future enhancements may include image processing-based drowsiness detection for higher accuracy, mobile application integration for real-time dashboard access, machine learning-based behavioral pattern recognition, and vehicle-to-vehicle (V2V) communication for cooperative collision avoidance. The proposed system provides a cost-effective, scalable solution that can significantly reduce road accident fatalities.

. REFERENCES

[1] G. S. Pannu, M. D. Ansari, and P. Gupta, "Design and implementation of autonomous car using Raspberry Pi," Int. J. Computer Applications, vol. 113, no. 9, 2015.

[2] V. N. Kumar, V. S. Reddy, and L. P. Sree, "Design and development of accelerometer based system for driver safety," Int. J. Science, Engineering and Technology Research (IJSETR), vol. 3, p. 12, 2014.

[3] C. Hahn, S. Feld, and H. Schroter, "Predictive collision management for time and risk dependent path planning," in Proc. 28th Int. Conf. Advances in Geographic Information Systems (SIGSPATIAL '20), pp. 405–408, 2020.

[4] G. N. A. H. Yar, A.-B. Noor-ul Hassan, and H. Siddiqui, "Real-time shallow water image retrieval and enhancement for low-cost unmanned underwater vehicle using Raspberry Pi," in Proc. 36th Annual ACM Symp. Applied Computing (SAC '21), pp. 1891–1899, 2021.

[5] A. F. B. A. de Oliveira and L. V. L. Filgueiras, "Developer assistance tools for creating native mobile applications accessible to visually impaired people: A systematic review," in Proc. 17th Brazilian Symp. Human Factors in Computing Systems (IHC 2018), 2018.

[6] L. L. Dhirani, T. Newe, E. Lewis, and S. Nizamani, "Cloud computing and Internet of Things fusion: Cost issues," Proc. Int. Conf. Sensor Technologies (ICST), pp. 1–6, 2018.

[7] S. Ramaswamy and R. Tripathi, "Internet of Things (IoT): A literature review," J. Computer Communications, vol. 3, pp. 164–173, 2015.

[8] S. Agrawal and D. Vieira, "A survey on Internet of Things," pp. 78–96, 2013.

[9] R. Zaheer and S. Khan, "Future Internet: The Internet of Things architecture, possible applications and key challenges," pp. 257–260, 2012.

[10] H. K. Singh, "IoT Basics," 2018. [Online]. Available: https://www.slideshare.net/hiteshkumarsingh10/iot basics-86993434

[11] M. N. Kamdar, V. Sharma, and S. Nayak, "A survey paper on RFID technology, its applications and classification of security/privacy attacks and solutions," IRACST – Int. J. Computer Science and Information Technology & Security, vol. 4, 2016.