

IOT-Based Automatic Vehicle Accident Detection and Rescue System

Dr. Nahid Khan¹, Rohan Kishor Pawar², Sandeep Ashok Rathod³, Vaibhavi Manoj Uplenchwar⁴,
Tanushree Prakash Raghatate⁵, Shraddha Dnyaneshwar Kelzarkar⁶

^{1,2,3,4,5,6} Department of Electrical Engineering,

Tulsiramji Gaikwad Patil College of Engineering and Technology, Mohgaon, Nagpur, India

Guide Email: nahid.electrical@tgpcet.com

How to Cite this Article:

Khan, N., Pawar, R. K., Rathod, S. A., Uplenchwar, V. M., Raghatate, T. P. & Kelzarkar, S. D. (2026). IOT-Based Automatic Vehicle Accident Detection and Rescue System. International Journal of Creative and Open Research in Engineering and Management, <i>02</i>(9), 1-9. <https://doi.org/10.55041/ijcope.v2i9.198>

License:

This article is published under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

© The Author(s). Published by International Journal of Creative and Open Research in Engineering and Management.



<https://doi.org/10.55041/ijcope.v2i9.198>

Abstract—In a road accident, the minutes that follow the impact often matter as much as the impact itself. Victims on highways, at night, or in low-traffic stretches frequently wait for help simply because nobody has raised the alarm. This paper presents a low-cost, retrofittable system that detects a crash on its own, estimates how serious it is, and notifies an emergency contact and a fleet dashboard with the vehicle's location, without requiring any action from the driver. The design splits the workload across two edge platforms. An ESP32 microcontroller handles time-critical tasks: impact and rollover detection from an MPU6050 inertial sensor, flame and temperature monitoring with relay-driven suppression, a manual SOS input, GSM calling and SMS through a SIM800L module, and MQTT telemetry. A Raspberry Pi 4B running OpenCV performs the two visionheavy tasks: continuous driver drowsiness monitoring using the eye aspect ratio and PERCLOS, and post-impact severity estimation that fuses impact magnitude, rollover state, occupant motion and a lightweight CNN damage score into a Minor, Moderate or Severe tag. A dedicated ESP32-CAM uploads a photograph of the scene. A 10-second cancellation window suppresses false alarms, and severity analysis runs inside that window so that it adds no delay to the alert. The paper details the architecture, detection logic, interfaces, power and timing budgets, and the evaluation protocol for the prototype.

Index Terms—accident detection, Internet

I. Introduction

Road crashes kill about 1.19 million people worldwide every year [1]. India carries a heavy share of that burden: the Ministry of Road Transport and Highways recorded 4,61,312 accidents in 2022, which claimed 1,68,491 lives and injured 4,43,366 people, with 36.2% of the deaths occurring on national highways [2].

Many of these deaths are not caused by the impact alone. Emergency medicine has long recognised that the first hour after serious trauma, often called the “golden hour,” strongly influences survival [3]. When a crash happens on an empty highway at 2 a.m., or when the driver is unconscious, that hour can begin with nobody knowing that anything has happened.

Two separate problems therefore deserve attention. The first is response delay: someone has to notice the crash, find out where it happened, and call for help. The second is prevention: fatigue and momentary inattention cause a significant share of crashes in the first place. Most low-cost aftermarket devices address only one of these, usually vehicle tracking, and leave the rest to the driver.

This work tackles both ends of the problem in one retrofittable unit. The main contributions are:

- 1) a two-tier edge architecture in which an ESP32 performs all hard real-time sensing, actuation and communication while a Raspberry Pi 4B runs the computer-vision workload, so that neither task can starve the other;
- 2) an impact and rollover detector with a drivercancellable confirmation window, during which severity estimation, scene capture and GPS acquisition run in parallel;
- 3) a severity score that fuses sensor evidence with vision evidence, so that responders receive a triage hint rather than a bare alarm;
- 4) integration of drowsiness monitoring, post-crash fire response, a manual SOS input and a live fleet dashboard into the same alert pipeline.

Section II reviews related systems. Sections III–VII describe the architecture, hardware, detection algorithms, vision modules and communication layer. Section VIII analyses timing, Section IX sets out the evaluation protocol, Section X discusses limitations, and Section XI concludes.

II. Related Work

Automatic crash notification is already mandated in some markets. The European eCall regulation requires new type-approved passenger cars and light commercial vehicles to place an automatic call to the 112 emergency number after a serious crash and to transmit a minimum set of data that includes location, time and direction of travel [4]. In India, AIS-140

requires public transport vehicles to carry a vehicle location tracking device and emergency buttons that report to a backend control centre [5]. Both are valuable, but eCall is factory-fitted and depends on the vehicle's own crash sensors, while AIS-140 relies on a person pressing a button.

Smartphone-based approaches lower the cost further. WreckWatch [6] used the accelerometer and microphone of a phone to detect crashes and notify responders, but a phone's mounting, orientation and battery state are not under the system designer's control, and a phone may be thrown about during the very event it is meant to detect.

For drowsiness, the eye aspect ratio (EAR) proposed by Soukupová and Čech [7] offers a simple, training-free blink measure computed from six eye landmarks. Such landmarks can be obtained from the regression-tree face aligner of Kazemi and Sullivan [8] as implemented in dlib [9], or from MediaPipe Face Mesh [10], [11], which estimates 468 facial landmarks in real time on mobile-class hardware. PERCLOS, the proportion of time the eyes are closed, is a well-established alertness measure [12], and the YawDD dataset [13] provides in-car driver videos for evaluating yawning and fatigue detectors. For on-device image classification, MobileNetV2 [14] offers a favourable accuracy-to-compute trade-off and is widely deployed through TensorFlow Lite.

The system reported here borrows from each of these lines of work but combines them in a single unit aimed at private vehicles and small fleets that cannot rely on factory-fitted equipment. A feature-level comparison is given in Table IX (Section X).

III. System Architecture

Fig. 1 shows the overall architecture. Instead of a single controller doing everything, the system is organised into three tiers. The *real-time edge tier* consists of the main ESP32 and a separate ESP32-CAM. The ESP32 reads the inertial, flame and temperature sensors and the SOS button, drives the relay and buzzer, talks to the GSM and GPS modules, and publishes telemetry. The *vision tier* is a Raspberry Pi 4B with a driver-facing camera that runs OpenCV [15] and MediaPipe. The *cloud tier* is an MQTT broker [16], a backend database and a web dashboard.

The reasoning behind this split is practical. Crash detection needs deterministic sampling at 100 Hz and must never be delayed by a busy camera pipeline or a Linux process being rescheduled. Face-landmark tracking, on the other hand, needs far more memory and compute than a microcontroller offers. Keeping the two apart means that if the Pi hangs or reboots, the ESP32 can still detect a crash, place the call and send the SMS; only the severity tag and drowsiness flag are lost. The two boards exchange short text frames over a 115 200 baud UART link. The ESP32 sends a trigger with the peak acceleration and rollover flag, and the Pi replies with drowsiness status and, after an event, the severity result.

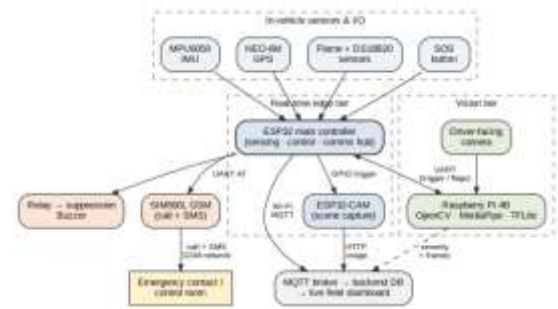


Fig. 1. Three-tier architecture of the proposed system. Solid arrows are data or control paths; the dashed path carries severity results and frames from the Raspberry Pi to the backend.

TABLE I
MAJOR HARDWARE COMPONENTS

Component	Function	Key specification
ESP32 DevKit (main)	Sensing, control, comms hub	Dual-core Xtensa LX6, 240 MHz, Wi-Fi/BT [17]
MPU6050	Impact and rollover detection	3-axis accel ± 16 g, 3-axis gyro, I ² C [18]
SIM800L	Emergency call and SMS	Quad-band GSM/GPRS, 3.4–4.4 V [19]
NEO-6M	Location for alerts	2.5 m CEP, 1 Hz default, UART [20]
IR flame module + DS18B20	Fire detection	760–1100 nm IR; -55 to $+125$ °C
Relay + suppression actuator	Fire response	5 V coil, contact rated for actuator load
Push button, buzzer	Manual SOS, local alerts	Active-low input; transistor-driven
ESP32-CAM (AI-Thinker)	Scene photo capture	OV2640, 2 MP, PSRAM
Raspberry Pi 4B	Vision processing	Quad Cortex-A72, 4 GB RAM [21]
Pi Camera / USB webcam	Driver-facing feed	640×480 at 15–30 fps used

IV. Hardware Design

A. Interfacing

Table II lists the pin assignment of the main ESP32. The IMU sits on the I²C bus and its data-ready interrupt paces the sampling loop. The ESP32 has three hardware UARTs: UART2 is given to the GSM modem, UART1 is remapped to GPIO32/33 for the GPS receiver, and UART0 is used for the Raspberry Pi link, with serial debug logging disabled in the deployed firmware. The flame sensor and SOS button use GPIO34 and GPIO35, which are input-only pins without internal pull-ups, so a 10 k Ω external pull-up is fitted on the button line. Both the ESP32 and the Pi use 3.3 V logic, so their UART link needs no level shifting.

The SIM800L deserves particular care. Its UART operates at 2.8 V, so a resistor divider is placed on the ESP32 TX line feeding the modem. More importantly, the modem draws current bursts of up to 2 A during GSM transmission [19]. Powering it from the ESP32's 3.3 V regulator is a common cause of brown-out resets exactly when a call is being placed. The module is therefore fed from its own buck converter set to 4.0 V, with a 1000 μ F low-ESR capacitor close to its supply pins.

TABLE III
ESTIMATED POWER BUDGET

Load	Rail (V)	Typ. (W)	Peak (W)
Raspberry Pi 4B + camera	5.0	5.50	7.50
ESP32 DevKit	3.3	0.33	0.79
ESP32-CAM	5.0	0.90	1.55
SIM800L (idle / TX burst)	4.0	0.08	8.00
NEO-6M	3.3	0.15	0.22
MPU6050, flame, DS18B20	3.3	0.07	0.07
Buzzer, relay coil	5.0	0.00	0.51
Total (load side)		7.02	18.64
12 V input ($\eta = 0.85$)	12	8.26	21.93

Values are typical figures from datasheets [17]–[21] and commonly reported measurements; they are to be confirmed on the bench.

V. Detection and Control Logic

A. Impact Detection

The MPU6050 is configured for the ± 16 g range so that genuine impacts do not saturate the sensor too early, with its digital low-pass filter set to reject engine vibration. At every 10 ms sample the ESP32 computes the resultant acceleration

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2} \quad (1)$$

which is independent of how the unit is mounted. A candidate impact is declared when $|a|$ exceeds a threshold T_a for at least three consecutive samples (30 ms). Requiring consecutive samples removes single sample spikes caused by potholes and door slams. The initial value of T_a is 4 g, well above the 0.8–1 g seen during hard braking, and it is tuned during the field trials described in Section IX. The peak value g_{peak} over the event is recorded for the severity stage.

B. Rollover Detection

A vehicle lying on its side produces no dramatic acceleration once it has come to rest, so a separate check is needed. The tilt angle between the unit's vertical axis and gravity is

$$\psi = \cos^{-1}(a_z / |a|) \quad (2)$$

and is evaluated only when $|a|$ is close to 1 g, i.e. when the vehicle is not accelerating. A rollover is flagged if ψ exceeds 60° continuously for 2 s. Calibration at installation records the resting orientation so that ψ is measured relative to the vehicle, not the enclosure. *C. Fire Detection and Suppression*

IR flame modules respond to sunlight and hot exhaust components, and absolute temperature near an engine varies widely with ambient conditions. Neither sensor is trusted on its own. A fire is declared when the flame output stays active for 2 s and the DS18B20 shows either an absolute temperature above a calibrated limit or a rise faster than 5°C per 10 s. The ESP32 then energises the suppression relay, opens a second relay that isolates auxiliary loads, and escalates the event through the alert pipeline. The prototype uses a small 12 V pump as a stand-in actuator. Water is not a suitable agent for fuel or electrical fires, so a deployable unit should trigger a certified engine compartment extinguisher, such as a dry-chemical or aerosol type, through the same relay output.

D. Confirmation Window and State Machine

Fig. 2 shows the main control flow and Fig. 3 the corresponding state machine. When an impact or rollover is detected, the ESP32 enters the CANDIDATE state and sounds a distinct buzzer pattern for 10 s. A conscious driver who has only hit a speed breaker hard can cancel the alarm with a 3 s long press of the SOS button. If nobody cancels, the system moves to ALERT. SOS and FIRE events skip the window entirely, because a person pressing SOS has already confirmed the emergency and a fire leaves no time to wait. Once in ALERT, SOS or FIRE, the unit keeps reporting until it is reset from the dashboard, so a dazed occupant cannot accidentally silence a genuine alert.



Fig. 2. Main control flowchart of the ESP32 firmware. Connector A return to the sensing loop.

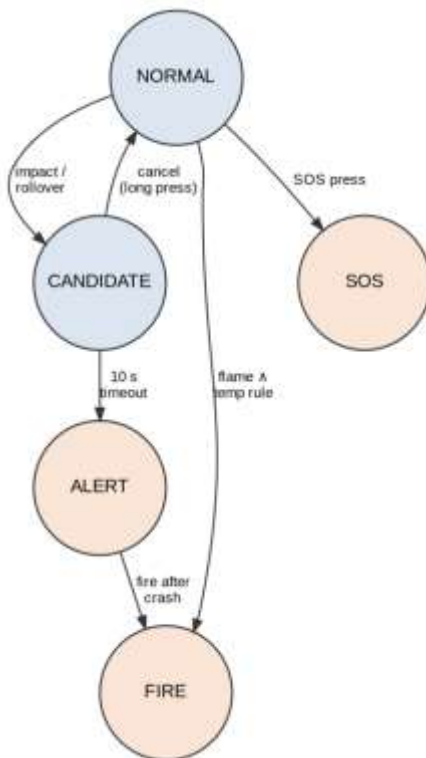


Fig. 3. Event state machine. ALERT, SOS and FIRE are latched states that return to NORMAL only on a remote reset from the dashboard.

to Media Pipe Face Mesh [10]. From the six landmarks around each eye (Fig. 4) the eye aspect ratio is computed as [7]

$$EAR = (||p_2 - p_6|| + ||p_3 - p_5||) / (2||p_1 - p_4||) \quad (3)$$

The EAR stays roughly constant while the eye is open and falls towards zero as it closes. With Media Pipe indexing, the landmark sets used are {33, 160, 158, 133, 153, 144} and {362, 385, 387, 263, 373, 380}, and the two eyes are averaged. Yawning is measured with a mouth aspect ratio built from the innerlip landmarks 13, 14, 78 and 308:

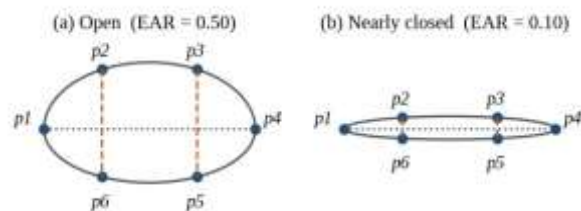


Fig. 4. Eye landmarks p_1 – p_6 used in (3). The EAR values shown are computed from the drawn geometry and illustrate how the ratio collapses as the eye closes.

$$MAR = ||m_{13} - m_{14}|| / ||m_{78} - m_{308}|| \quad (4)$$

A single long blink is not drowsiness, so two time-based rules are used (Fig. 5). An *acute* alarm fires when EAR stays below 0.21 for 1.5 s. A *gradual* alarm uses PERCLOS [12] over a sliding 60 s window,

$$\text{PERCLOS} = N_{\text{closed}} / N_{\text{total}} \quad (5)$$

and fires when it exceeds 0.15, or when three or more yawns (MAR above 0.6 for over 1.5 s) occur within 5 min. The Pi sounds its own buzzer at once and sends a DROWSY flag to the ESP32, which includes it in the next telemetry message. If no face is found for 3 s, a separate distraction flag is raised. The thresholds given here are starting values; eye shape varies between people, and a short calibration of the open-eye EAR when the driver starts the trip allows a per-driver threshold of about 75% of that baseline.

VI. Vision Modules on the Raspberry Pi

A. Driver Drowsiness Monitoring

The driver-facing camera streams 640×480 frames to the Pi, which applies histogram equalisation to reduce the effect of changing cabin light and passes each frame

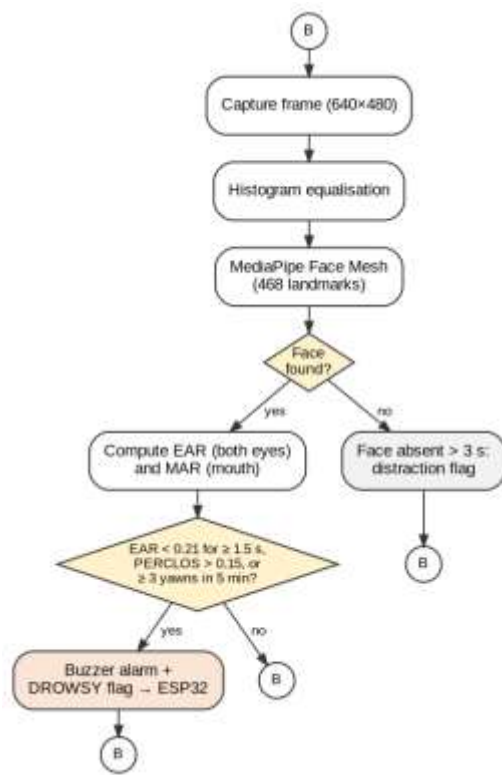


Fig. 5. Per-frame drowsiness detection pipeline on the Raspberry Pi 4B. Connector B returns to frame capture.

B. Accident Severity Estimation

Severity estimation starts the moment the ESP32 sends its trigger, which is also the start of the 10 s confirmation window. It fuses four normalised features (Fig. 6). The first is $x_1 = \min(g_{\text{peak}}/16, 1)$, the impact magnitude relative to the sensor's full scale. The second is the rollover flag $x_2 \in \{0, 1\}$. The third is an occupant-stillness measure $x_3 = 1 - M$, where M is the normalised mean absolute frame difference of the cabin view over the window: $M = (1 / K) \sum_k \text{mean}|I_k - I_{k-1}| / 255$ (6)

scaled so that normal occupant movement gives M close to 1. Prolonged stillness after a hard impact is one of the more worrying signs a responder could be told about. The fourth, x_4 ,

is the probability of visible vehicle damage produced by a MobileNetV2 classifier [14], fine-tuned by transfer learning on vehicle-damage images and run as a quantised TensorFlow Lite model on frames from the forward view. The score is

$$S = 0.40x_1 + 0.20x_2 + 0.15x_3 + 0.25x_4 \quad (7)$$

and is mapped to a tag using Table IV. The weights are initial heuristic choices that give the most trust to the physical measurement; they are intended to be re-fitted once labelled test data are available. Two overrides apply regardless of S : a rollover or a subsequent fire always yields SEVERE. The analysis is advisory. It never suppresses an alert, it only tells the receiver how urgent the event appears.

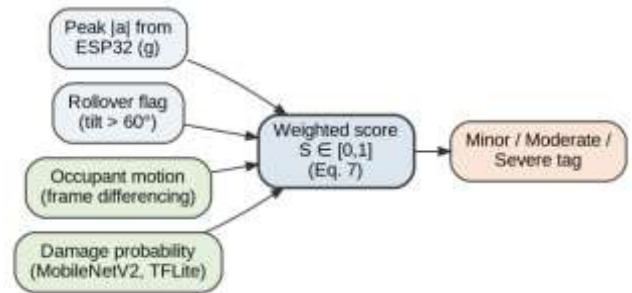


Fig. 6. Severity fusion: sensor features (blue) and vision features (green) are combined into a single score S .

TABLE IV
SEVERITY MAPPING AND RESPONSE

Tag	Condition	Response
Minor	$S < 0.35$	SMS + dashboard alert
Moderate	$0.35 \leq S < 0.65$	SMS + call to contact
Severe	$S \geq 0.65$, rollover or fire	Call control room first, then SMS to all contacts

VII. Communication and Fleet Dashboard

A. GSM Alerts

On entering ALERT, SOS or FIRE, the ESP32 sends the SMS first and then places the voice call, because an SMS

survives a weak or intermittent signal better than a call does. A typical message reads:

ACCIDENT ALERT: MH31AB1234, Sev: SEVERE,
21.14580,79.08820
<https://maps.google.com/?q=21.14580,79.08820> at
14:32:05 23/09/26

At 124 characters, all from the GSM 7-bit default alphabet, it fits in a single 160-character SMS segment, so it cannot be split and partially delivered. If no GPS fix is available, the last valid fix and its age are sent instead.

B. Telemetry and Dashboard

Each vehicle publishes to the topics in Table V. Periodic telemetry uses QoS 0, since a lost position update is replaced five seconds later. Events use QoS 1 so that the broker acknowledges delivery. A retained last-will message marks a vehicle offline if its connection drops. Wi-Fi comes from an in-vehicle hotspot; when it is unavailable, the SIM800L's GPRS link serves as a fallback path for event messages. The ESP32-CAM uploads its JPEG over HTTP and the returned URL is attached to the event. The dashboard plots every vehicle on a map, highlights those in an alert state, and keeps an incident log with severity tags and images for later review. Table VI lists the software stack.

TABLE V
MQTT TOPIC STRUCTURE

Topic	QoS	Content
fleet/{id}/telemetry	0	Every 5 s: time, lat, lon, speed, state, DROWSY flag, RSSI
fleet/{id}/event	1	Type (ACCIDENT, SOS, FIRE), severity, g_{peak} , lat, lon, image URL
fleet/{id}/status	1, retained	Online / offline (last will)
fleet/{id}/cmd	1	Dashboard → vehicle: reset, configuration

TABLE VI
SOFTWARE AND TOOLS

Tool / library	Purpose
Arduino core for ESP32, PlatformIO (C++)	ESP32 and ESP32-CAM firmware
Python 3, OpenCV [15]	Frame capture and preprocessing
MediaPipe Face Mesh [10], [11]	Facial landmarks for EAR and MAR
TensorFlow Lite, MobileNetV2 [14]	Damage classification
Mosquitto MQTT broker [16]	Telemetry and event transport
Node.js / Firebase, Google Maps API	Backend, storage, live map dashboard

VIII. Timing Analysis

Table VII breaks down the expected time from impact to the SMS leaving the vehicle. The dominant term is the deliberate 10 s confirmation window. Because photo capture, severity analysis and GPS refresh all happen *inside* that window, they do not add to the total. The only steps that follow it are the SMS and the MQTT publish. The design target for impact-to-SMS is therefore about 15 s, a small fraction of the golden hour.

TABLE VII
DESIGN LATENCY BUDGET (IMPACT TO ALERT)

Stage	Target	Runs in
Impact detection (3 samples)	30 ms	Series
Confirmation window	10 s	Series
ESP32-CAM capture + upload	≤ 3 s	Parallel
Severity analysis on Pi	≤ 8 s	Parallel
SMS submission (AT+CMGS)	≤ 5 s	Series
MQTT event publish	≤ 1 s	Parallel with SMS
Impact → SMS sent	≈ 15 s	

Targets, not measured values. Network delivery time after submission depends on the operator.

IX. Experimental Setup and Evaluation Protocol

Crash testing a real vehicle is neither safe nor affordable at this scale, so the evaluation is staged. Impact detection is tested by mounting the enclosure on a weighted trolley and rolling it into a padded barrier from controlled heights, which gives repeatable impacts. False-alarm behaviour is tested the other way round: the unit is driven over a mixed urban and highway route containing speed breakers, potholes and hard braking, where any transition to ALERT counts as a failure. The drowsiness module is evaluated on the YawDD videos [13] and on live volunteers who simulate prolonged eye closure and yawning under day and night cabin lighting. Table VIII lists the tests and their pass criteria. The results column is completed from the prototype trials.

TABLE VIII
TEST PLAN AND RESULTS

ID	Test	Pass criterion	Result
T1	Trolley impact, 20 trials	20/20 reach CANDIDATE	to be recorded
T2	Road route with speed breakers, potholes, hard braking	No ALERT state entered	to be recorded
T3	Rollover, enclosure turned 90°	Flag within 2.5 s	to be recorded
T4	SOS press to SMS received	≤ 10 s, correct location	to be recorded
T5	Impact to SMS received	≤ 20 s	to be recorded
T6	Flame + heat source at 30 cm	Relay on within 3 s	to be recorded

T7	Drowsiness on YawDD and volunteers	Report precision, recall	to be recorded
T8	Severity on labelled images	Report confusion matrix	to be recorded
T9	Dashboard update latency	≤ 6 s per vehicle	to be recorded

X. Discussion and Limitations

Table IX compares the proposed unit with the two regulatory systems discussed in Section II. Its advantage is breadth: detection, triage, evidence, prevention and fleet visibility in one retrofittable device. It is not a substitute for certified in-vehicle safety systems, and several limitations should be stated plainly.

TABLE IX
FEATURE COMPARISON

Feature	eCall [4]	AIS-140 [5]	Proposed
Automatic crash detection	Yes	No	Yes
Manual SOS	Yes	Yes	Yes
Location to responders	Yes	Yes	Yes
Feature	eCall [4]	AIS-140 [5]	Proposed
Severity hint	No	No	Yes
Scene image	No	No	Yes
Driver drowsiness	No	No	Yes
Fire detection/response	No	No	Yes
Fleet dashboard	No	Yes	Yes
Deployment	Factoryfitted	Mandated fit	Retrofit

Network dependence. The SIM800L is a 2G-only module. 2G coverage in India is shrinking and is not offered by every operator, so a production design should use a 4G LTE Cat-1 module with integrated GNSS. The firmware's AT-command layer is written to make that swap straightforward.

Sensing range. Real crash pulses can exceed the MPU6050's ± 16 g range. Saturation does not prevent detection, since the threshold is far lower, but it caps x_1 and so compresses the severity scale at the top end. A ± 200 g accelerometer would remove this limit.

Night-time vision. A standard camera sees little in a dark cabin, which is exactly when drowsiness is most likely. A NoIR camera with 850 nm IR illumination is the obvious remedy. Sunglasses and heavy head rotation also degrade landmark tracking.

Severity model. The weights in (7) are not yet learned from data, and public vehicle-damage images differ from what an in-car camera sees. The severity tag should therefore be treated as a triage hint until it has been validated on representative footage.

Privacy. A driver-facing camera records people. Frames are processed on the Pi and discarded; only event-time images leave the vehicle, and this policy should be communicated to drivers in any fleet deployment.

XI. Conclusion and Future Work

This paper has presented an accident detection and rescue system that separates hard real-time work on an ESP32 from vision work on a Raspberry Pi 4B. The division keeps the life-critical path of detect, confirm, call and message independent of the more fragile vision software, while still delivering a severity hint, a scene photograph and continuous drowsiness monitoring. A cancellable confirmation window reduces false alarms, and running all slow analysis inside that window keeps the design target for impact-to-SMS near 15 s. Future work will complete the field trials in Table VIII, learn the severity weights from labelled data, move to a 4G module, and add seat-belt status, alcohol sensing and direct integration with the national 112 emergency response system.

Acknowledgment

The authors thank the Department of Electrical Engineering, Tulsiramji Gaikwad Patil College of Engineering and Technology, Nagpur, for the guidance and laboratory facilities provided during this work.

References

- [1] World Health Organization, *Global Status Report on Road Safety 2023*. Geneva, Switzerland: WHO, 2023.
- [2] Transport Research Wing, Ministry of Road Transport and Highways, Government of India, "Road accidents in India— 2022," New Delhi, India, 2023.
- [3] R. A. Cowley, "A total emergency medical system for the State of Maryland," *Maryland State Med. J.*, vol. 24, no. 7, pp. 37–45, 1975.
- [4] Regulation (EU) 2015/758 of the European Parliament and of the Council concerning type-approval requirements for the deployment of the eCall in-vehicle system based on the 112 service, *Off. J. Eur. Union*, L 123, May 2015.
- [5] Automotive Research Association of India, "Intelligent transportation systems (ITS)—Requirements for public transport vehicle operation," AIS-140, 2016.
- [6] J. White, C. Thompson, H. Turner, B. Dougherty, and D. C. Schmidt, "WreckWatch: Automatic traffic accident detection and notification with smartphones," *Mobile Netw. Appl.*, vol. 16, no. 3, pp. 285–303, 2011.
- [7] T. Soukupová and J. Čech, "Real-time eye blink detection using facial landmarks," in *Proc. 21st Comput. Vis. Winter Workshop*, Rimske Toplice, Slovenia, 2016.
- [8] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2014, pp. 1867–1874.
- [9] D. E. King, "Dlib-ml: A machine learning toolkit," *J. Mach. Learn. Res.*, vol. 10, pp. 1755–1758, 2009.
- [10] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile GPUs," arXiv:1907.06724, 2019.
- [11] C. Lugaresi *et al.*, "MediaPipe: A framework for building perception pipelines," arXiv:1906.08172, 2019.
- [12] D. F. Dinges and R. Grace, "PERCLOS: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance," Federal Highway Admin., Tech. Brief FHWA-MCRT-98-006, 1998.
- [13] S. Abtahi, M. Omidyeganeh, S. Shirmohammadi, and B. Hariri, "YawDD: A yawning detection dataset," in *Proc. 5th ACM Multimedia Syst. Conf.*, 2014, pp. 24–28.

- [14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.C. Chen, “MobileNetV2: Inverted residuals and linear bottlenecks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.
- [15] G. Bradski, “The OpenCV library,” *Dr. Dobbs’s J. Softw. Tools*, 2000.
- [16] OASIS, “MQTT version 3.1.1,” OASIS Standard, Oct. 2014.
- [17] Espressif Systems, “ESP32 series datasheet,” Shanghai, China.
- [18] InvenSense Inc., “MPU-6000 and MPU-6050 product specification,” Rev. 3.4, 2013.
- [19] SIMCom Wireless Solutions, “SIM800L hardware design,” Shanghai, China.
- [20] u-blox AG, “NEO-6 u-blox 6 GPS modules data sheet,” Thalwil, Switzerland.
- [21] Raspberry Pi Ltd., “Raspberry Pi 4 Model B datasheet,” Cambridge, U.K.