AN IOT-BASED INTELLIGENT ACCIDENT DETECTION AND AUTOMATED EMERGENCY RESPONSE FRAMEWORK

Dr K RAJESH[1], P SPANDANA[2], B MAHESH[3]
ASSOCIATE PROFESSOR[1], ASSISTANT PROFESSOR [2&3]
DEPARTMENT OF ECE, SRI CHAITANYA INSTITUTE OF TECHNOLOGY AND RESEARCH, PONNEKAL, KHAMMAM, TELANGANA-507170

ABSTRACT
Road accidents are a major public safety concern, and delays in identifying accidents and notifying emergency services can increase the risk of severe injuries and loss of life. To address this challenge, this paper proposes an IoT-Enabled Automated Accident Detection and Reporting Platform that uses Internet of Things (IoT) technology to detect vehicle accidents and automatically transmit accident-related information to designated emergency contacts and response authorities. The proposed platform integrates sensors, a microcontroller, GPS, and wireless communication modules to continuously monitor vehicle conditions and identify abnormal events associated with accidents. When a potential accident is detected based on predefined sensor thresholds, the system processes the collected data, determines the vehicle's geographical location, and generates an accident alert containing relevant information such as GPS coordinates and incident status. The alert is transmitted through an available communication network to facilitate faster emergency notification and response. The platform aims to reduce manual intervention, improve accident reporting efficiency, and support timely assistance for accident victims. By combining real-time sensing, location tracking, and automated communication, the proposed system provides an effective IoT-based approach to intelligent vehicle safety and emergency management.

Keywords: Internet of Things (IoT), Automated Accident Detection, Accident Reporting, Vehicle Safety, GPS Tracking, Sensor-Based Monitoring, Microcontroller, Wireless Communication, Emergency Alert System, Real-Time Monitoring.

This is an open access article under the creative commons license https://creativecommons.org/licenses/by-nc-nd/4.0/
CC BY-NC-ND 4.0

1. INTRODUCTION

Road traffic accidents are a major public-safety challenge because the consequences of a crash depend not only on the collision itself but also on how quickly accurate information reaches emergency responders. The World Health Organization reports approximately 1.16 million road-traffic deaths annually worldwide and identifies timely post-crash care as an important part of reducing preventable harm [1]. In many situations, a victim may be unconscious, unable to use a phone, or located on an unfamiliar road. A system that automatically recognizes a possible accident and communicates its location can therefore support faster incident awareness.

The Internet of Things (IoT) provides a practical foundation for connecting vehicle-mounted sensors, embedded controllers, communication modules, cloud services, and emergency-contact applications. An in-vehicle unit can continuously monitor motion-related parameters such as acceleration, vibration, tilt, and orientation. When a pattern consistent with a crash is observed, the controller can verify the event, acquire geographic coordinates from a Global Positioning System (GPS) receiver, and transmit an alert through a cellular or internet connection.

This project, titled “IoT-Enabled Automated Accident Detection and Reporting Platform,” proposes an integrated platform with four primary functions: (i) sensing and detection of abnormal vehicle motion, (ii) event verification to reduce false alarms, (iii) GPS-based location acquisition, and (iv) automated reporting to registered contacts and a cloud dashboard. The platform is intended as an assistance and

notification system; it does not replace airbags, certified vehicle safety systems, emergency dispatch procedures, or human confirmation by authorized responders.

The central design consideration is balancing detection sensitivity with reliability. A very sensitive threshold may identify potholes, harsh braking, or vehicle handling as accidents. A conservative threshold may miss low-impact crashes. The proposed approach therefore combines multiple sensor observations over a short time window, applies configurable thresholds, and uses a cancellation interval where appropriate. The system stores an incident identifier, timestamp, coordinates, sensor readings, and notification status so that the event can be reviewed.

The report is organized as follows. Section 2 reviews relevant accident-detection and IoT communication research. Section 3 defines the problem and objectives. Sections 4 and 5 compare existing and proposed approaches and describe the architecture. Section 6 lists hardware and software requirements. Sections 7–9 explain detection, cloud communication, GPS tracking, and emergency reporting. Section 10 presents system design and communication protocols. Sections 11–13 describe the algorithm, implementation plan, and evaluation. Sections 14 and 15 discuss limitations, future work, and conclusions.

2. LITERATURE REVIEW

Research on automated accident reporting generally addresses two related challenges: identifying a crash reliably and communicating useful incident information without delay. Shaik et al. described a smart-car concept in which accelerometer and GPS information is sent to a cloud service, enabling subscribed recipients to receive the vehicle's location after an accident [2]. This approach illustrates the value of connecting on-board sensing with cloud-based notification.

Kumar, Acharya, and Lohani proposed an IoT-based vehicle accident detection and classification system using sensor fusion [3]. Their work considered vehicle-motion attributes including speed, absolute linear acceleration, altitude change, pitch, and roll. It compared Naïve Bayes, Gaussian mixture model, and decision-tree approaches for classifying collision, rollover, fall-off, and non-accident conditions. The study demonstrates that combining multiple motion features can provide richer information than relying on a single threshold. Its reported model performance belongs to that published study and should not be interpreted as a result of the present project.

Prabu and Namitha reviewed IoT-based automatic road accident detection systems and discussed sensing technologies, data-processing algorithms, communication protocols, and system architectures [4]. Their review highlights recurring challenges such as sensor accuracy, scalability, and real-time processing. These issues are directly relevant to a practical deployment: the device must operate under vibration, temperature variation, intermittent connectivity, and different vehicle types.

Other implementations combine vibration or tilt sensors with GPS and cellular communication. A 2022 IEEE conference paper describes a system using vibration, tilt, flame sensing, GPS, and network connectivity, with a delay period that allows a driver to cancel a false alert before reporting [5]. Such a cancellation mechanism is useful where a sensor may respond to a pothole or sudden maneuver. A 2024 IEEE conference paper also describes using accelerometer and eye-blink sensing with GPS and GSM to send location information to emergency contacts [6]. These designs show the range of possible sensor combinations, although the suitability of each sensor depends on the project's intended vehicle and detection objectives.

Recent work has also explored accident-severity classification and cloud dashboards. An IEEE conference paper from 2025 describes a prototype using GSM and Wi-Fi communication, sensor inputs, and a Random Forest classifier for severity estimation [7]. This demonstrates the potential of combining edge sensing with cloud visualization, but a reported accuracy from that paper cannot be transferred to a different hardware or dataset.

Across the literature, several common design patterns emerge: local sensing, embedded event processing, location acquisition, network transmission, and an emergency notification interface. The proposed platform follows this layered pattern while emphasizing sensor fusion, event confirmation, timestamped incident records, secure transmission, and explicit acknowledgement tracking. Its contribution is an integrated design and implementation workflow rather than a claim of a new medical or crash-certification algorithm.

3. PROBLEM STATEMENT AND RESEARCH OBJECTIVES

Problem Statement: In a road accident, emergency assistance may be delayed when occupants cannot call for help, when witnesses do not know the exact location, or when incident details are communicated inaccurately. Manual reporting also depends on human availability and awareness. Existing standalone alarms or single-sensor prototypes may generate false notifications due to road shocks, rough surfaces, abrupt braking, or device movement. There is a need for an IoT-enabled platform that monitors vehicle motion, verifies suspicious events, obtains geographic coordinates, and communicates a structured incident report to designated recipients.

Research Aim: To design and implement an embedded IoT platform that detects probable vehicle accidents and automatically reports event information and GPS location to registered emergency contacts and a cloud-based monitoring interface.

Specific Objectives:

1. Develop a sensor interface for acquiring vehicle-motion information using an accelerometer/gyroscope or suitable vibration and tilt sensors.
2. Implement a real-time event-detection routine using thresholding, time-window analysis, and multi-sensor confirmation.
3. Integrate a GPS receiver to obtain latitude, longitude, time, and positioning validity.
4. Establish wireless communication using a cellular modem, Wi-Fi, or an appropriate combination based on network availability.
5. Create a cloud data structure for storing incident identifiers, sensor observations, coordinates, timestamps, and notification status.
6. Provide an emergency-reporting mechanism that sends a concise alert and a map link to registered contacts.
7. Include a cancellation or confirmation period to reduce accidental reporting where system design permits.
8. Evaluate the prototype using controlled test cases and report measured detection, false-alarm, location, and communication metrics.

Research Questions:

- Can a combination of motion features reduce dependence on a single sensor threshold?
- How reliably can the system obtain and transmit GPS coordinates under open-sky and obstructed conditions?
- What is the end-to-end delay between a verified trigger and successful alert delivery?
- How frequently do non-accident events produce false triggers under the selected thresholds?

Scope and Assumptions: The prototype focuses on detection and notification, not autonomous vehicle control or direct dispatch authority. It assumes that the device is securely mounted, the power supply is stable, GPS and communication modules are correctly configured, and emergency contact details are maintained. A live emergency-service integration requires formal authorization, a verified interface, and compliance with applicable local procedures. For academic testing, controlled simulations should be performed safely without creating real crashes or sending false emergency calls.

4. EXISTING SYSTEM AND PROPOSED SYSTEM

Existing System: In a conventional manual reporting process, an occupant or witness recognizes an accident, locates a phone, contacts an emergency number, describes the incident, and provides a location verbally. This process may be impossible if occupants are unconscious or the location is unfamiliar. Smartphone-based emergency features can assist, but their availability and behavior vary by device, operating system, region, and user configuration.

A basic electronic accident-alert prototype often uses one vibration or impact sensor connected to a microcontroller. When the sensor output exceeds a fixed threshold, the controller sends an SMS containing a preconfigured message. Such a design is inexpensive and straightforward, but a single threshold may confuse potholes, speed breakers, dropped devices, or harsh braking with collisions. Some systems also lack event history, cloud monitoring, location-quality checks, delivery acknowledgement, and a method for cancelling false alarms.

Proposed System: The proposed platform uses a layered architecture. The sensing layer collects acceleration, angular velocity, vibration, and optional tilt or impact signals. The edge-processing layer filters noise, computes features, applies thresholds, and checks whether multiple indicators support a probable accident. The event-management layer assigns an incident ID and starts a short confirmation/cancellation timer. The location layer requests a fresh GPS fix and records the fix quality. The communication layer transmits the event through cellular data, SMS, or Wi-Fi depending on the available network. The cloud layer stores the event and updates a dashboard. The notification layer sends alerts to configured contacts and exposes the location through a map link.

Key Improvements:

- Multi-sensor evidence rather than a single vibration trigger.
- Local decision-making to reduce dependence on continuous cloud connectivity.
- GPS validity checks and timestamped location records.
- Alternative reporting paths, such as internet data and SMS fallback, where supported.
- Cloud incident history and notification status.
- Cancellation and acknowledgement controls to manage false alarms and confirm receipt.
- A modular design that allows sensor, controller, and cloud components to be replaced.

Comparison:

Manual reporting: human-dependent; location may be approximate; no automatic sensing.

Single-sensor GSM prototype: automatic alert; limited event discrimination; typically little cloud history.

Proposed IoT platform: sensor fusion, event verification, GPS, cloud records, notification tracking, and configurable communication.

The proposed system is an improvement in integration and workflow, but it must be validated against representative driving conditions. A block diagram or feature list alone does not prove detection accuracy, response-time improvement, or operational readiness.

5. SYSTEM ARCHITECTURE AND BLOCK DIAGRAM

The platform is divided into vehicle-side and remote-side subsystems. The vehicle-side unit performs continuous sensing and immediate event evaluation. The remote-side services receive, store, visualize, and distribute verified incident messages.

Block Diagram (logical representation):

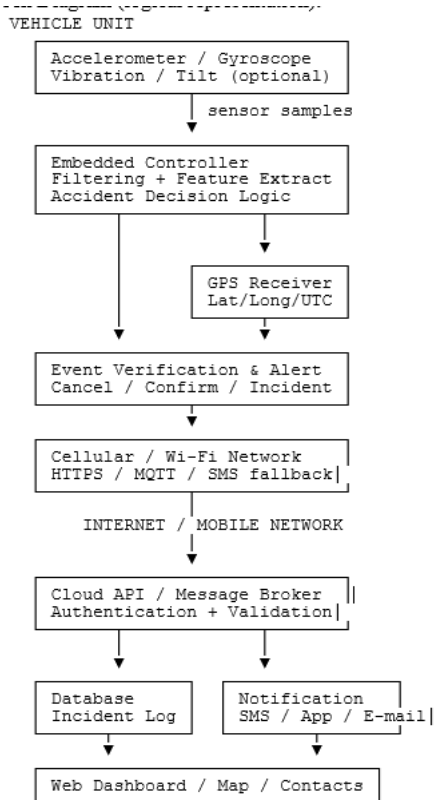


Figure 1. Logical architecture of the proposed IoT accident detection and reporting platform.

Functional Description: The sensor module supplies raw readings to the controller. The controller samples at a configured rate and applies calibration and filtering. Feature extraction may calculate resultant acceleration, peak magnitude, change in orientation, and duration above a threshold. A decision block combines these features to classify the current window as normal, suspicious, or probable accident. If the event is probable, the system requests a GPS fix and creates an incident packet. A communication module publishes the packet to the cloud API or broker. The backend validates the device identity, stores the record, and triggers notifications. A dashboard presents the incident state and map location to authorized users.

The architecture separates time-critical detection from cloud-dependent functions. If internet access is lost, the controller can retain an event locally and attempt SMS transmission or retransmission when service returns, provided the hardware supports these functions. Every incident should have a unique identifier to avoid duplicate records when messages are retried.

6. HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

1. Embedded controller: ESP32, Arduino-compatible board, STM32, or another microcontroller with sufficient GPIO, serial interfaces, memory, and processing capacity. An ESP32 is suitable where Wi-Fi and Bluetooth are useful; a separate cellular modem may be needed for wide-area coverage.
2. Motion sensor: A 3-axis accelerometer or 6-axis IMU such as MPU6050-class hardware can provide acceleration and angular-rate data. Sensor selection must consider range, sampling rate, noise, mounting orientation, and calibration.
3. Optional impact/vibration sensor: A vibration switch or piezoelectric sensor can provide supplementary evidence but should not be treated as a complete accident detector.
4. GPS receiver: A GNSS module such as u-blox NEO-series or equivalent can provide latitude, longitude, speed, time, and fix status. Exact performance depends on antenna placement and satellite visibility.

5. Communication module: GSM/LTE modem for cellular data and SMS, or Wi-Fi where a suitable access point is available. Network bands and SIM compatibility must be checked for the deployment region.
6. Power supply: Vehicle DC input with regulated conversion, reverse-polarity protection, fuse, transient protection, and a backup battery if continued reporting after vehicle power loss is required.
7. Supporting components: enclosure, mounting bracket, antenna, wiring harness, status LEDs, buzzer, push button, and optional display.

Software Requirements:

- Firmware: Embedded C/C++ using Arduino IDE, PlatformIO, or the vendor SDK.
- Sensor libraries: I2C/SPI and IMU driver libraries; GPS parsing library such as TinyGPS++ or an equivalent NMEA parser.
- Network layer: HTTPS REST API or MQTT client library. MQTT is a lightweight publish/subscribe protocol standardized by OASIS and designed for constrained IoT environments [8].
- Backend: Python Flask/FastAPI, Node.js, or another API framework.
- Database: Firebase, PostgreSQL, MySQL, or a time-series/cloud database, selected according to deployment requirements.
- Dashboard: HTML/CSS/JavaScript or a web application framework with map visualization.
- Notification service: SMS gateway, email service, or mobile push notification provider.
- Development tools: serial monitor, IDE, API testing utility, version control, and a test log.

Interface Considerations: I2C is commonly used for an IMU; UART is commonly used for GPS and cellular modules; SPI may be used by other peripherals. Logic voltage levels must be checked before wiring. Cellular modems can draw high current bursts and should not be powered directly from a weak microcontroller regulator. The GPS antenna should be placed with a clear view of the sky where practical. The complete unit must be mounted securely so that its measured orientation remains consistent.

7. ACCIDENT DETECTION METHODOLOGY

The accident-detection process begins with continuous sampling of motion sensors. Let ax, ay, and az represent calibrated acceleration along three axes. The resultant acceleration magnitude is:

$$A(t) = \sqrt{(ax^2 + ay^2 + az^2)}$$

The system may calculate a dynamic acceleration component by removing the estimated gravity vector. Let ωx, ωy, and ωz denote angular-rate measurements. Additional features may include peak acceleration, acceleration change, angular-rate peak, orientation change, duration above threshold, and vehicle speed before and after the event.

Preprocessing: Raw sensor readings may contain electrical noise, vibration, and bias. The firmware should perform offset calibration, apply a low-pass or moving-average filter where appropriate, and reject invalid samples. Filtering must not be so aggressive that it suppresses a short collision impulse. The sampling frequency and filter parameters should be selected experimentally for the target vehicle.

Threshold and Fusion Logic: A first-stage trigger occurs when one or more features exceed configurable thresholds. A second-stage check evaluates the combination of features over a sliding time window. For example, a high acceleration peak together with a sudden angular-rate change and a persistent orientation change may provide stronger evidence than a vibration peak alone. The decision can be rule-based or replaced by a trained classifier if a representative labelled dataset is available.

A simple rule-based decision can be represented as:

ACCIDENT = (Apeak > TA) AND ((ωpeak > Tω) OR (Δorientation > Tθ))

This expression is illustrative only. Thresholds TA, Tω, and Tθ must be calibrated for the sensor, mounting location, and vehicle type. A production system should use a more carefully validated rule or model and should explicitly represent uncertainty.

False-Alarm Control: When the suspicious state is reached, the system may sound a buzzer and start a cancellation interval. If the driver presses the cancel button within the configured period, the event is marked as cancelled and may be logged for analysis. If no cancellation occurs, the event is promoted to a confirmed alert. Cancellation must not be the only safety mechanism because an injured or unconscious driver may be unable to respond.

Detection States:

NORMAL: sensor values remain within normal operating patterns.

SUSPECTED: one or more motion features cross initial thresholds.

VERIFYING: multi-feature check and cancellation timer are active.

CONFIRMED: event is accepted for emergency reporting.

REPORTED: alert has been transmitted and acknowledged by the backend or gateway.

FAILED/RETRY: transmission failed; the system retains the event and retries according to policy.

A critical safety distinction is that "confirmed" means the device's algorithm accepted the event; it does not prove that a medically significant injury occurred. The notification should use wording such as "possible vehicle accident detected" unless a validated severity assessment is available.

8. IoT-BASED DATA TRANSMISSION AND CLOUD INTEGRATION

After local verification, the controller constructs an incident message. A recommended data payload contains: device_id, incident_id, event_timestamp, latitude, longitude, GPS_fix_status, speed (if available), acceleration_peak, angular_rate_peak, event_class, battery_voltage, firmware_version, and message_sequence_number. Personal information should be minimized; only information required for emergency response should be transmitted.

Communication Options: HTTPS REST is appropriate for request/response submission to a backend API. MQTT is useful when devices publish events to a broker and multiple services subscribe to incident topics. OASIS defines MQTT as a lightweight client-server publish/subscribe messaging transport for IoT and constrained environments [8]. The project should select a protocol based on bandwidth, latency, power, reliability, and backend support. SMS can serve as a secondary path where packet-data service is unavailable, but SMS delivery and timing are not guaranteed.

Cloud Workflow:

1. The device authenticates to the API or broker using a device credential.
2. The incident payload is transmitted over an encrypted connection.
3. The backend validates required fields, device identity, timestamp, and coordinate format.
4. The event is written to a database with a unique incident ID.
5. The notification service sends an alert to configured recipients.
6. The backend updates delivery and acknowledgement states.
7. The dashboard displays incident location, time, status, and available sensor metadata.

Reliability: Each event should be assigned a unique ID and sequence number. If the device does not receive an acknowledgement, it can retry using the same incident ID. The backend should treat repeated submissions as idempotent so that retries do not create multiple emergency incidents. Local flash storage can retain unsent events. Retry intervals should use bounded backoff to avoid excessive network traffic.

Security and Privacy: Use TLS for network transport, unique device credentials, access control for dashboards, and secure storage of contact information. Avoid hard-coded public credentials in firmware. Cloud APIs should validate payload size and types, limit request rates, log access, and protect against

replay or unauthorized device registration. Location information is sensitive; access should be limited to authorized contacts and responders, and retention periods should be defined.

Cloud integration does not guarantee that emergency services are automatically dispatched. The prototype can notify configured contacts or an authorized monitoring service. Direct integration with public emergency systems requires approved interfaces, operational agreements, and compliance with local regulations.

9. GPS-BASED LOCATION TRACKING AND EMERGENCY REPORTING

The GPS/GNSS module receives satellite signals and estimates the device position. A typical receiver outputs NMEA sentences containing position, time, speed, and fix information. The firmware parses these sentences and checks whether the receiver has a valid fix before using the coordinates. A location record should include latitude, longitude, timestamp, fix validity, and, where available, estimated accuracy or dilution-of-precision information.

Location Acquisition: Under open-sky conditions, a receiver may obtain a position more quickly and consistently than in tunnels, dense urban streets, parking structures, or areas with obstructed antenna visibility. The system should not assume that the most recently read coordinates are current. It should store the age of the GPS fix and mark stale or unavailable coordinates clearly. If a valid fix is not available at the accident time, the platform can transmit the last known location with its timestamp and indicate that it is not a fresh fix.

Emergency Message: A concise alert can contain vehicle/device identifier, event time, coordinates, map URL, and a note that the event is algorithmically detected. Example message:

"Possible vehicle accident detected. Device: VEH-001. Time: [UTC timestamp]. Location: [latitude], [longitude]. Map: https://maps.google.com/?q=[latitude],[longitude]. Please verify and contact emergency services if required."

The map URL is generated from the received coordinates. The message should avoid unsupported statements about injury severity. A notification record should capture the intended recipient, provider response, transmission timestamp, and acknowledgement state. If a recipient does not acknowledge, the system may notify additional contacts according to a preconfigured escalation policy.

Location Tracking and Privacy: Continuous tracking is not always necessary for an accident alert. A privacy-preserving design can transmit periodic low-frequency health/status messages and send detailed location only after a verified incident or explicit user request. Users should be informed about what is collected, when it is transmitted, who can view it, and how long it is retained.

Emergency Reporting in India: A prototype should not send test messages to public emergency numbers. For demonstrations, use consenting test contacts and clearly label test events. Any integration with public emergency response infrastructure must be undertaken through the appropriate authorized channels. The system should be treated as an assistive reporting platform, not a substitute for calling the applicable emergency number when a person is able to do so.

10. SYSTEM DESIGN AND COMMUNICATION PROTOCOLS

The system design follows a modular embedded-to-cloud model. The firmware is divided into sensor acquisition, signal conditioning, feature extraction, accident-state management, GPS handling, communication, and diagnostics modules. This modular structure improves maintainability and allows the sensing hardware or cloud provider to be changed without rewriting the complete application.

Sensor Interface: An IMU commonly communicates over I2C or SPI. The firmware should verify the sensor identity, configure measurement ranges, read samples at a fixed interval, and detect missing or out-of-range values. GPS and cellular modules commonly use UART. Serial buffers should be sized to prevent data loss when messages arrive in bursts.

Data Format: JSON is suitable for readable event payloads. Example:

{

"device_id": "VEH-001",
"incident_id": "INC-000123",
"timestamp_utc": "YYYY-MM-DDThh:mm:ssZ",
"latitude": 17.000000,
"longitude": 78.000000,
"gps_fix": true,
"event_type": "possible_accident",
"status": "confirmed"
}

The coordinates above are placeholders and are not a real incident location.

Protocol Selection: MQTT supports publish/subscribe communication and is useful when an event must be delivered to several backend consumers, such as a database writer, alert service, and dashboard updater [8]. HTTPS is useful for simple API-based event submission and can be easier to integrate with conventional web applications. The project may use either protocol; the implementation should document the chosen one rather than claiming both were implemented if only one is used.

Topic/API Design: A sample MQTT topic may be `vehicle/VEH-001/incident`; a REST endpoint may be `POST /api/v1/incidents`. Device credentials should be scoped to the device's own topic or API permission. Acknowledgement messages should include the incident ID and processing status.

Power and Fault Handling: The controller should monitor supply voltage where possible. Brownout recovery should not erase a pending incident. Watchdog timers can recover from firmware hangs. On startup, the device should check sensor connectivity, GPS availability, network registration, and storage state. Fault conditions should be indicated through status LEDs or diagnostic logs.

The design should define normal sampling rate, detection-window length, cancellation interval, GPS timeout, transmission timeout, retry count, and event-retention capacity. These are configurable engineering parameters; they must be selected and validated during implementation rather than copied as universal constants.

11. ALGORITHM AND SYSTEM WORKFLOW

Algorithm 1: Automated Accident Detection and Reporting

Input: Motion sensor samples, GPS data, device configuration, network status.

Output: Incident record and emergency notification.

Step 1: Initialize the microcontroller, sensor interfaces, GPS receiver, communication modem, and local event storage.

Step 2: Verify that required sensors respond. Record a diagnostic fault if a required component is unavailable.

Step 3: Continuously acquire accelerometer and gyroscope samples at the configured sampling rate.

Step 4: Calibrate offsets and filter high-frequency noise using the selected signal-processing method.

Step 5: Calculate motion features such as resultant acceleration, peak acceleration, angular-rate magnitude, and orientation change.

Step 6: Compare the current feature window with configured thresholds.

Step 7: If the evidence is insufficient, remain in NORMAL state and continue sampling.

Step 8: If suspicious evidence is found, enter SUSPECTED/VERIFYING state and evaluate the multi-sensor decision rule.

Step 9: Activate the buzzer or indicator and start the cancellation timer, if included in the design.

Step 10: If the event is cancelled, store a cancelled-event record and return to monitoring.

Step 11: If the event is not cancelled and verification criteria are met, create a unique incident ID and timestamp.

Step 12: Request a fresh GPS fix. If unavailable before timeout, attach the last known coordinates with their age and validity status.
Step 13: Construct the incident payload and store it locally before transmission.
Step 14: Transmit the payload using the configured HTTPS or MQTT connection. Use SMS fallback only if implemented and configured.
Step 15: Wait for backend acknowledgement. On success, mark the event as REPORTED; otherwise retain it and retry according to the retry policy.
Step 16: Trigger notifications to authorized contacts and update the cloud dashboard.
Step 17: Continue monitoring and retain diagnostic information for later evaluation.
Workflow Explanation: The algorithm separates event detection from event reporting. This is important because network availability should not determine whether the local controller recognizes a probable crash. Similarly, GPS acquisition may take additional time, so the controller should preserve the event timestamp and sensor features before waiting for coordinates. The cloud backend should distinguish an event received by the server from an alert delivered to a recipient; these are separate stages.
State Transition: NORMAL → SUSPECTED → VERIFYING → CONFIRMED → REPORTED. A cancelled event transitions from VERIFYING to CANCELLED. A communication failure transitions from CONFIRMED to RETRY_PENDING and later to REPORTED or DELIVERY_FAILED. These states support traceability and help identify where delays occur.
The algorithm is a reference workflow. A final implementation must replace generic thresholds and timeouts with the actual values used in firmware and document the corresponding sensor configuration, code version, and test procedure.

12. IMPLEMENTATION AND EXPERIMENTAL ANALYSIS

Implementation Plan: First, assemble the controller, IMU, GPS receiver, communication modem, and regulated power supply. Confirm voltage compatibility and serial/I2C wiring before connecting the device to a vehicle. Second, test each sensor independently using a serial monitor. Third, implement data acquisition and calibration. Fourth, develop the event-detection routine and verify it using safe bench tests and controlled motion tests. Fifth, integrate GPS parsing and confirm that coordinates are marked valid only when the receiver reports a valid fix. Sixth, connect the device to the cloud endpoint and test event submission, acknowledgement, database storage, and notifications. Finally, perform end-to-end tests using consenting test recipients.
Experimental Design: Evaluation should use labelled scenarios, not uncontrolled road accidents. Possible scenarios include normal stationary operation, normal driving vibration, speed-breaker or pothole-like shocks in a controlled test rig, sudden braking simulation, sharp turns, device handling, and controlled impact simulation using a safe fixture. Each trial should be labelled as accident-like or non-accident for analysis. No test should expose people to crash risk.
Metrics:
- Detection rate / recall = TP / (TP + FN)
- Precision = TP / (TP + FP)
- F1-score = 2 × Precision × Recall / (Precision + Recall)
- False-alarm rate = FP / (number of non-accident test windows or trips)
- Event-to-cloud latency = cloud receipt time − verified event time
- Notification latency = recipient-provider submission time − verified event time
- GPS error = distance between reported position and a surveyed reference position
- Delivery success rate = acknowledged alerts / total attempted alerts

Here TP denotes correctly detected accident-like trials, FN denotes missed accident-like trials, FP denotes false accident detections, and TN denotes correctly rejected non-accident trials. The

denominator and unit for false-alarm rate must be stated clearly (per trip, per hour, or per test window) so comparisons are meaningful.

Experimental Results Table (populate only with measured values):

Metric | Measurement
Number of accident-like trials | Not supplied
Number of non-accident trials | Not supplied
True positives / false negatives | Not supplied
False positives / true negatives | Not supplied
Precision / recall / F1-score | Not supplied
Mean event-to-cloud latency | Not supplied
Mean notification latency | Not supplied
Mean GPS position error | Not supplied
Alert delivery success rate | Not supplied

Analysis: Results should be reported across repeated trials and, where possible, different vehicle types, mounting positions, network conditions, and GPS environments. Mean values should be accompanied by standard deviation or percentile values. A confusion matrix should be included for classification. Latency should be measured with synchronized timestamps. A screenshot of a dashboard can demonstrate that a user interface exists, but it cannot by itself establish detection accuracy or delivery reliability. Since no project-specific logs or measured test results were supplied for this report, no numerical performance claim is made.

13. RESULTS AND DISCUSSION

The expected functional outcome of the proposed platform is an end-to-end incident workflow: motion data are acquired, a probable accident is identified, GPS coordinates are attached, an incident record is transmitted, and a notification is presented to authorized contacts. Successful demonstration should show each stage separately, including sensor readings, state transitions, GPS validity, server receipt, database persistence, and notification acknowledgement.

The main engineering trade-off is detection sensitivity versus false alarms. A threshold that is too low may react to ordinary road conditions; a threshold that is too high may miss less severe impacts. Multi-sensor fusion, time-window checks, and a cancellation mechanism can reduce some false triggers, but they require empirical calibration. Network latency and GPS acquisition time should also be analysed separately because a fast detection decision does not guarantee a fast notification.

A complete discussion should compare measured values with the objectives and explain failed or delayed cases. Until actual test logs are available, the system should be described as a proposed or implemented prototype according to the work completed, not as a clinically validated or road-certified product.

14. LIMITATIONS AND FUTURE SCOPE

Limitations include dependence on sensor mounting and calibration, variation among vehicle structures, possible false alarms from rough roads, GPS unavailability in tunnels or covered locations, cellular network outages, power interruption, and delays in third-party SMS or cloud services. A device-level event classification does not establish injury severity. The system also requires careful handling of location privacy, device authentication, contact data, and false emergency notifications.

Future work may include collecting a larger and diverse labelled dataset; training and validating machine-learning models for collision and rollover classification; adaptive thresholds for different vehicle categories; edge inference with model updates; redundant GNSS and cellular connectivity; backup power; tamper detection; and integration with authorized emergency-response platforms. A production design should undergo environmental, electromagnetic compatibility, automotive power-

transient, cybersecurity, and road-safety testing. Human-factors testing should assess whether alarms and cancellation controls are understandable and usable under stress.

15. CONCLUSION

This report presented the design of an IoT-enabled automated accident detection and reporting platform. The proposed architecture integrates vehicle-motion sensing, embedded event verification, GPS location acquisition, wireless communication, cloud storage, and emergency notifications. The layered design separates local accident recognition from remote reporting and supports incident identifiers, retry handling, acknowledgement tracking, and a cloud dashboard. Multi-sensor evidence and a verification interval are intended to reduce false alerts compared with a simple single-threshold trigger.

The platform's practical value depends on rigorous validation. Detection performance, false-alarm rate, location accuracy, and end-to-end notification delay must be measured using safe, repeatable test scenarios. No experimental values are claimed in this report because project-specific test logs were not provided. With calibrated hardware, secure communication, verified cloud services, and authorized emergency workflows, the proposed platform can serve as a foundation for further research and development in connected vehicle safety.

16. REFERENCES

[1] World Health Organization, "Road traffic injuries," Fact sheet, Jul. 20, 2026. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries/

[2] A. Shaik, N. Bowen, J. Bole, G. Kunzi, D. Bruce, A. Abdelgawad, and K. Yelamarthi, "Smart Car: An IoT Based Accident Detection System," in Proc. 2018 IEEE Global Conf. Internet of Things (GCIoT), Alexandria, Egypt, Dec. 2018, doi: 10.1109/GCIoT.2018.8620131.

[3] N. Kumar, D. Acharya, and D. Lohani, "An IoT-based vehicle accident detection and classification system using sensor fusion," IEEE Internet Things J., vol. 8, no. 2, pp. 869–880, Jan. 2021, doi: 10.1109/JIOT.2020.3008896.

[4] C. R. Prabu and K. Namitha, "IoT-based automatic road accident detection systems: A review," in Proc. 2024 4th Int. Conf. Soft Computing for Security Applications (ICSCSA), Salem, India, Sep. 2024, pp. 716–721, doi: 10.1109/ICSCSA64454.2024.00122.

[5] "Internet of Things (IoT) Enabled Automobile Accident Detection and Reporting System," in Proc. 2022 5th Information Technology for Education and Development (ITED), Nov. 2022, doi: 10.1109/ITED56637.2022.10051224.

[6] "Vehicle Accident Detection and Locating Using GSM and GPS," in Proc. 2024 Int. Conf. Communication, Computing and Internet of Things (IC3IoT), Chennai, India, Apr. 2024, doi: 10.1109/IC3IoT60841.2024.10550407.

[7] "Design of an IoT-based Vehicle Accident Severity Detection System with Artificial Intelligence," in Proc. 2025 IEEE Middle East and North Africa Communications Conf. (MENACOMM), Byblos, Lebanon, Feb. 2025, doi: 10.1109/MENACOMM62946.2025.10910997.

[8] A. Banks, E. Briggs, K. Borgendale, and R. Gupta, Eds., "MQTT Version 5.0," OASIS Standard, Mar. 7, 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html