

Review on IOT-Based Vehicle Accident Detection and Automated Insurance Notification System

Mrs. B. Uma

Assistant Professor, Dept. of CSE
Malnad College of Engineering
Hassan, Karnataka, India
uma@mce.ac.in

Aapthi G A

Dept. of CSE
Malnad College of Engineering
Hassan, Karnataka, India
aapthiraxidi@gmail.com

Abhishek H D

Dept. of CSE
Malnad College of Engineering
Hassan, Karnataka, India
lsabhi2005@gmail.com

Darshan Devappa Kivadannavar

Dept. of CSE
Malnad College of Engineering
Hassan, Karnataka, India
darshandk121@gmail.com

Deepika U H

Dept. of CSE
Malnad College of Engineering
Hassan, Karnataka, India
deepikadeepika4681@gmail.com

Abstract—Vehicle accidents claim over a million lives each year, yet the communication gap between an accident event and the notification of emergency services remains largely unsolved in practice. Manual reporting, reliance on bystanders, and paper-heavy insurance procedures together stretch the critical post-accident window well beyond what is medically acceptable. This paper describes an IoT-based accident detection and insurance notification system that addresses the notification gap at low hardware cost. The prototype uses an ESP32 microcontroller, an MPU6050 inertial sensor, a NEO-6M GPS receiver, and Google Firebase as the cloud backend. On detection of an impact exceeding a set acceleration threshold, the system logs GPS coordinates and pushes a structured record to Firebase, where emergency and insurance personnel can read it through web or mobile interfaces. Bench testing across 30 simulated impact events gave a detection rate of 96.7%, a mean location error of 2.8 m, and an average upload time of 1.4 s over a 4G hotspot. The paper also reviews five related works and positions the proposed design relative to their individual strengths and gaps.

Index Terms—IoT, ESP32, MPU6050, NEO-6M GPS, Firebase, accident detection, insurance automation, cloud computing, embedded systems, road safety

I. INTRODUCTION

India recorded 4.61 lakh road accidents in 2022 alone, killing over 1.68 lakh persons — roughly one death every three minutes [1]. Globally, the WHO puts the annual toll at 1.35 million fatalities. The numbers have remained stubbornly high despite improvements in vehicle design and road infrastructure, partly because the chain of events after an accident is still poorly automated.

When a collision occurs, two problems tend to compound each other. First, notification of emergency services usually depends on bystanders who happen to be present and able to call for help. Accidents on highways at night, in tunnels, or in sparsely populated areas may go unreported for ten to forty minutes — a window that is frequently fatal for head trauma and internal bleeding cases. Second, insurance claim processing in India follows a largely paper-based workflow:

the owner must obtain a police FIR, arrange a surveyor visit, gather medical receipts, and submit them manually, a process that routinely takes three to six weeks even for uncomplicated claims.

Embedded electronics and wireless connectivity have matured to the point where both problems can be addressed with a small, inexpensive in-vehicle module. A three-axis accelerometer can detect the force signature of a collision; a GPS receiver can pin the location; and a cellular or Wi-Fi link can push the data to a cloud database reachable by ambulance dispatchers and insurance back-ends simultaneously. The sensor costs involved — an ESP32 costs under \$5, an MPU6050 under \$2, and a NEO-6M under \$8 — put this kind of solution within reach for ordinary private vehicles, not just commercial fleets.

This paper makes the following contributions:

- A review of five recent systems covering sensor-based IoT detection, ML-enhanced detection, blockchain-based insurance automation, computer vision approaches, and basic GPS/GSM alerting, with an assessment of what each approach does and does not cover.
- A concrete hardware design using the ESP32, MPU6050, NEO-6M, and Firebase, with firmware details sufficient for replication.
- Quantitative bench results from 30 simulated impact events covering detection accuracy, GPS error, upload latency, and offline retry behaviour.
- A structured gap analysis comparing the proposed design against reviewed systems across cost, cloud connectivity, GPS use, and insurance notification.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes data collection. Section IV covers hardware components. Section V presents the system architecture. Section VI details the methodology. Section VII reports experimental results.

Section VIII compares reviewed systems. Section IX discusses limitations. Section X outlines future directions. Section XI concludes.

II. LITERATURE SURVEY

A. IoT-Enabled Vehicular Accident Detection and Notification System (2023)

Patel and Shah [2] built a vehicle-mounted IoT node using an accelerometer, a GPS receiver, and an ESP8266 Wi-Fi module. When measured acceleration crossed a fixed threshold, the device sent an alert packet with GPS coordinates to a remote monitoring centre. Field trials showed average alert delivery under five seconds from impact.

The work is notable for its clean, low-cost architecture, but its scope stops at the notification step. There is no cloud database backing the alerts, so historical records cannot be queried after an initial acknowledgement. More importantly, the system has no pathway to insurance processing — no claim is initiated, no documentation is generated, and no multi-party channel (hospital, police, insurer together) is established. In areas where the monitoring centre is unstaffed outside business hours, the alert may go unnoticed entirely.

B. Integrating AI, IoT and Blockchain for Real-Time Insurance Claim Automation (2024)

Joseph, Kumar, and Nair [3] took a more ambitious route, combining IoT sensors with an AI classifier and a blockchain smart contract. When a sensor event is classified as a genuine loss, the smart contract triggers an automatic payout without human review. The authors reported claim cycle times falling from days to under two hours.

The framework was originally built for crop insurance but its design maps onto vehicle insurance with minor adaptation. The blockchain layer is genuinely useful for fraud prevention — each accident record is hashed, timestamped, and immutable — but it brings non-trivial infrastructure requirements. Running even a private Ethereum or Hyperledger node demands server capacity and blockchain expertise that most regional insurance offices or vehicle-tracking startups do not have. AI model training requires a large labelled historical claims dataset, which is another resource many organisations lack. The system is technically impressive but practically out of reach for low-budget deployments.

C. Intelligent Accident Detection Using ML over IoT (2022)

Rathod, Tipale, and Jajulwar [4] tackled the false-alarm problem that plagues threshold-based detectors. A Support Vector Machine (SVM) was trained on MPU6050 readings labelled across four classes: normal driving, hard braking, speed bumps, and actual collisions. Running inference on embedded hardware, the trained model cut the false positive rate by about 34% compared to a fixed threshold while keeping detection sensitivity above 91%.

This is a meaningful improvement for roads with variable surface quality, where a fixed threshold can be triggered by potholes. The limitation lies in training data: the dataset was

collected on specific road types and vehicle categories. When the same model was evaluated on a different vehicle class or in a region with different road characteristics, performance dropped noticeably. Periodic retraining is required, which implies an ongoing data collection and annotation effort that is difficult to sustain without dedicated infrastructure.

D. Action Recognition for Accident Detection in Smart City Transport (2023)

The smart city research group [5] moved the detection responsibility from the vehicle to roadside infrastructure. CNN-based models processed surveillance camera streams and flagged accident scenes in real time, achieving 94.3% accuracy at 25 frames per second on a GPU server.

For urban expressways where cameras are already installed, this approach requires no per-vehicle hardware, which is a significant deployment advantage. The coverage problem is the main drawback: rural roads, which account for a disproportionate share of fatal accidents in India, have almost no camera infrastructure. Even in cities, gaps in camera coverage leave stretches of road unmonitored. The continuous GPU compute cost is also substantial — a deployment across even a medium-sized city network would require a sizeable GPU cluster running around the clock.

E. Vehicle Tracking and Accident Alert Using GPS and GSM (2023)

Singh [6] implemented a simple but practical system using a tilt sensor, SIM800 GSM module, and NEO-6M GPS. On detecting an abnormal tilt event, the device sent an SMS with GPS coordinates to a fixed list of numbers — family, hospital, police — using standard GSM text messaging.

The design is inexpensive, easy to manufacture, and works wherever GSM coverage exists, which in India is over 95% of the country by geographic area. These are real advantages. The weaknesses are equally clear: SMS is not a database; there is no structured record that an insurance system can ingest. A tilt sensor is a blunt instrument compared to a three-axis IMU — bumpy roads, speed breakers, and parking on slopes can all produce false tilt events. And there is no mechanism for insurance notification or claim initiation.

III. DATASET AND DATA GENERATION

The proposed system does not use a pre-existing benchmark dataset. All evaluation data was generated from the physical prototype during controlled lab testing.

The MPU6050 produces acceleration readings at 100 Hz along three orthogonal axes. During testing, these readings were logged to a laptop over USB-serial alongside ground-truth labels (accident / non-accident) that were manually assigned for each trial. GPS fixes were logged from the NEO-6M simultaneously.

Two categories of test data were collected:

- 1) **Accident events (30 trials):** Simulated using a drop-weight rig that applied controlled impacts to the enclosure mounting the prototype. Four impact directions

were tested — frontal, lateral, rear, and rollover — at impact energies selected to produce peak resultant accelerations ranging from 1.9 g to 6.4 g. Each trial was logged with a timestamp, the raw accelerometer waveform, the peak resultant acceleration, and the system's detection decision.

- 2) **Normal driving events (60-minute session):** The prototype was mounted in a car driven on a route covering city roads, a stretch of highway, and a section of unpaved rural road. All detections during this session were counted as false positives.

The threshold $T_{acc} = 2.5$ g was derived from this data. Hard braking produced peak resultant accelerations up to 0.9 g; crossing a speed bump at 20 km/h produced up to 1.4 g; the lowest-energy simulated collision produced 2.3 g. The gap between 1.4 g (worst-case non-accident) and 2.3 g (mildest accident) gave a margin of 0.9 g from which 2.5 g was chosen as a conservative threshold. This choice resulted in one missed detection (the 2.3 g low-speed lateral impact) and zero false positives during the normal driving session.

Firebase upload records were logged with server-side timestamps, enabling upload latency to be computed as the difference between the Firebase write timestamp and the local detection timestamp maintained by the ESP32's internal clock, synchronized to GPS UTC.

IV. HARDWARE COMPONENTS

A. ESP32 Microcontroller

The ESP32 (Espressif Systems) is a dual-core Xtensa LX6 processor running at up to 240 MHz with 520 KB of on-chip SRAM and 4 MB of flash. Of particular relevance here is the on-chip 802.11 b/g/n Wi-Fi radio, which eliminates the need for an external wireless module and noticeably simplifies the circuit board. The device exposes I2C, SPI, UART, and ADC peripherals sufficient for the sensors used in this project. At around \$4–\$6 in single-unit pricing, it is among the lowest-cost options for embedded Wi-Fi development.

B. MPU6050 Six-Axis IMU

The MPU6050 (InvenSense / TDK) integrates a three-axis MEMS accelerometer and a three-axis MEMS gyroscope in a 4 mm × 4 mm QFN package. The accelerometer supports selectable full-scale ranges of ± 2 g, ± 4 g, ± 8 g, and ± 16 g at 16-bit resolution. For accident detection the ± 8 g range was selected, giving a resolution of approximately 0.24 mg per LSB. Communication with the ESP32 is over I2C at 400 kHz. Readings are sampled at 100 Hz in this implementation.

Collision events produce a distinctive impulse in the resultant acceleration magnitude:

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

When a_r exceeds the calibrated threshold T_{acc} , the firmware treats the event as a probable accident. Using the resultant magnitude rather than any single axis makes detection insensitive to sensor mounting orientation.

C. NEO-6M GPS Receiver

The NEO-6M (u-blox) is a 72-channel GPS receiver with a specified position accuracy of 2.5 m CEP under open-sky conditions. It communicates over UART at 9600 baud, emitting NMEA 0183 sentences including GPRMC and GPGLA. Cold-start time-to-first-fix averages around 27 seconds; subsequent hot-start fixes typically take under one second when ephemeris data is cached in the module's battery-backed RAM. The module ships with a ceramic patch antenna that is adequate for roof or dashboard mounting in a vehicle.

D. Firebase Realtime Database

Firebase Realtime Database (Google LLC) is a cloud-hosted NoSQL store that represents data as a JSON tree and distributes updates to connected clients via WebSocket within tens to hundreds of milliseconds. For this project, accident records are written as JSON nodes; web and mobile clients subscribe to the relevant database path and receive push notifications as soon as a new record appears. Firebase handles authentication, TLS encryption, and horizontal scaling, none of which require any configuration on the ESP32 side beyond supplying a valid API key.

E. Power Supply and Backup

Under normal operation the circuit draws power from the vehicle's 12 V DC supply through an LM2596-based step-down converter regulated to 3.3 V. To handle the scenario where a severe collision interrupts the vehicle's electrical supply before the Firebase upload completes, a 3.7 V / 1000 mAh LiPo cell is connected through a TP4056 charge controller and a power-path switch. In a power-loss event, the switch transfers the load to the LiPo within a few milliseconds, giving the ESP32 enough reserve time to complete the upload.

V. SYSTEM ARCHITECTURE

The system is organized into four functional layers stacked from hardware to end-user application.

Sensing layer. The MPU6050 and NEO-6M run continuously. The accelerometer produces I2C readings at 100 Hz; the GPS receiver outputs NMEA sentences over UART approximately once per second when a fix is active.

Processing layer. The ESP32 firmware, written in C++ using the Arduino framework, polls the MPU6050 on every 10 ms interrupt, computes a_r from Equation (1), and checks the result against T_{acc} . Normal operation is a tight polling loop with a current draw of approximately 160 mA. When a detection event occurs, firmware execution branches to the GPS query and upload routines.

Communication layer. After a detection, the ESP32 opens a Wi-Fi connection (or uses an existing one) and transmits the accident payload to Firebase over HTTPS using the Firebase ESP Client library [10]. If Wi-Fi is unavailable, the payload is serialized to SPIFFS flash for later retransmission. A background task checks connectivity every 30 seconds and retransmits any pending SPIFFS payloads.

Application layer. The Firebase database acts as the central broker. A web dashboard, built with JavaScript using the Firebase SDK, shows incoming accident records on an embedded map. An insurance portal reads the same data through a REST interface. Email and push-notification triggers can be configured through Firebase Cloud Messaging at no additional infrastructure cost.

VI. METHODOLOGY

A. Sensor Initialization

On power-up, the ESP32 writes 0×00 to MPU6050 register $0 \times 6B$ to exit sleep mode. It then configures the accelerometer full-scale range to ± 8 g (register $0 \times 1C$, value 0×10) and sets the digital low-pass filter to a 10 Hz cutoff (register $0 \times 1A$, value 0×05) to suppress vibration noise above the frequency range relevant to collisions. The GPS task is started as a parallel FreeRTOS task; it begins parsing NMEA sentences immediately and caches the most recent valid fix.

B. Accident Detection Loop

The main detection loop runs at 100 Hz driven by a hardware timer interrupt. On each tick, raw 16-bit accelerometer counts are read from registers $0 \times 3B$ – 0×40 over I2C, converted to g using the ± 8 g scale factor (1 LSB = $1/4096$ g), and a_r is computed. If $a_r \geq T_{acc} = 2.5$ g, the accident handler is called and a 10-second lockout counter is set to prevent duplicate alerts from the same event (vehicles typically oscillate for several seconds after a hard impact). The gyroscope output is logged in parallel but not used for the detection decision in the current firmware version; it is reserved for future severity classification.

C. GPS Fix Retrieval

The accident handler reads the cached GPS fix immediately. If the cached fix is older than five seconds (possible if the vehicle has been stationary or is in a GPS-denied area), the handler waits up to 30 seconds for a fresh fix. Failing that, it marks the GPS fields with the last valid coordinates and a staleness flag before proceeding. The UTC timestamp embedded in the GPS fix is used as the authoritative accident time rather than the ESP32's local clock, which drifts by a few seconds per day without synchronization.

D. Firebase Upload

A valid Wi-Fi connection is checked. If connected, the handler constructs the following JSON payload:

```
{
  "vehicleID"      : "KA-13-AB-1234",
  "accidentDetected": true,
  "latitude"       : 13.0067,
  "longitude"      : 76.0996,
  "timestamp"      : "2026-05-23T08:34:12Z",
  "accelPeak_g"    : 3.82,
  "gpsStaleness"   : false
}
```

This object is written to the Firebase path `/accidents/{vehicleID}/{timestamp}` via an authenticated HTTPS PUT. The Firebase ESP Client library handles OAuth2 token refresh automatically. If the Wi-Fi association is lost at the moment of upload, the payload is written to a SPIFFS file named with the local millisecond counter and queued for retransmission.

E. Notification to Emergency and Insurance Parties

Firebase's WebSocket-based push mechanism delivers the new record to all active listeners typically within 300 ms of the server write. The web dashboard renders the GPS coordinates on a Leaflet map, displays the peak acceleration value as a severity indicator, and logs the vehicle ID for cross-referencing with insurance records. Emergency dispatch operators using the same dashboard can copy GPS coordinates directly into navigation software. The system design allows any number of independent clients — hospitals, police control rooms, insurance servers — to subscribe simultaneously without any additional server-side configuration.

F. Offline Retry Behaviour

When the ESP32 regains Wi-Fi after an outage, the SPIFFS queue is flushed in chronological order. Each payload is uploaded and then deleted from flash on confirmation of a successful HTTP 200 response from Firebase. In all offline retry tests conducted during evaluation, 100% of buffered payloads were successfully delivered after connectivity was restored.

VII. EXPERIMENTAL RESULTS

Table I summarizes the results from the 30-event test campaign and the 60-minute normal-driving false-positive trial.

TABLE I
SYSTEM PERFORMANCE UNDER SIMULATED ACCIDENT CONDITIONS

Metric	Result
Total simulated impact events	30
Correctly detected	29 / 30
Detection accuracy	96.7%
False positives (60 min normal driving)	0
Mean GPS coordinate error	2.8 m
Mean upload latency (Wi-Fi connected)	1.4 s
Maximum upload latency observed	3.1 s
SPIFFS offline retry success rate	100%

The single missed detection was a low-energy lateral impact at roughly 15 km/h that produced a peak a_r of 2.3 g, sitting 0.2 g below T_{acc} . A lower threshold would catch such events but at the cost of false positives on rough surfaces; the trade-off is discussed further in Section IX.

GPS errors averaged 2.8 m, marginally above the 2.5 m CEP quoted in the NEO-6M datasheet. The difference is consistent with multipath effects during testing near a two-storey building. All 30 recorded positions fell on the correct road segment when plotted on a map, which is the practically relevant criterion for emergency dispatch.

The 1.4 s average upload latency covers all time from the moment a_r crosses T_{acc} to the moment Firebase acknowledges the write. Of this, approximately 0.8 s is GPS fix confirmation and JSON serialization; the remaining 0.6 s is network round-trip and Firebase processing. The 3.1 s maximum occurred during a test in which the GPS retry path was exercised (the module had just cold-started). All latencies are well within the 5 s threshold considered acceptable for emergency alerting in related literature [2].

VIII. COMPARISON OF REVIEWED SYSTEMS

Table II maps each reviewed system against the proposed design across six attributes.

TABLE II
COMPARATIVE SUMMARY OF REVIEWED SYSTEMS

System	Detection	Cloud	GPS	Insur.	Low Cost
Patel & Shah [2]	Accel+WiFi	Part.	Yes	No	Yes
Joseph et al. [3]	AI+BC	Yes	Yes	Yes	No
Rathod et al. [4]	SVM/IoT	Part.	Yes	No	Yes
Smart City [5]	CNN video	No	No	No	No
Singh [6]	Tilt+GSM	No	Yes	No	Yes
Proposed	Accel+FB	Yes	Yes	Yes	Yes

Several patterns are worth noting. A persistent cloud backend — one that retains searchable historical records rather than just routing a one-time alert — is absent or only partially present in three of the five reviewed systems. Without it, data cannot be audited after the fact, which is a basic requirement for insurance use. Insurance notification itself appears in only one reviewed system [3], and that system achieves it through a blockchain smart contract whose infrastructure requirements put it out of reach for small-scale deployment. The proposed system reaches the same functional outcome through Firebase, which is available as a managed cloud service requiring no server administration. The camera-based system [5] achieves the highest raw detection accuracy but is the only reviewed system with no GPS integration and no affordable deployment path outside camera-dense urban corridors.

IX. CHALLENGES AND LIMITATIONS

Fixed threshold sensitivity. The 2.5 g threshold was derived from tests on one vehicle type on specific road surfaces. Vehicles with stiffer suspension, heavier mass, or different mounting positions for the sensor will have different acceleration profiles for equivalent collisions. Unpaved roads in poor condition can produce sustained vibrations that approach or briefly exceed the threshold during normal driving, particularly for motorcycles or light commercial vehicles.

GPS coverage gaps. The NEO-6M loses fix in tunnels, multi-storey car parks, and dense urban canyons. The fallback of transmitting the last known position introduces a location error that grows with the distance and speed of travel since the last valid fix. In a worst-case scenario — a vehicle that entered a long tunnel minutes before an accident — the reported location could be off by several kilometres.

Wi-Fi coverage dependency. The system's primary communication channel requires a Wi-Fi access point, either roadside or from the occupant's mobile phone acting as a hotspot. In rural areas, this is not reliably available. The SPIFFS retry mechanism defers the alert rather than eliminating the gap. Adding a SIM7600 or similar 4G module as a fallback would resolve this but increases cost by roughly \$15–\$20.

Power interruption. A very severe collision may rupture the vehicle's 12 V battery or blow the fuse before the LiPo backup can complete the upload. At 1.4 s average upload latency, the LiPo capacity is more than sufficient in most cases, but the edge case cannot be eliminated entirely without a dedicated supercapacitor buffer and guaranteed power-fail detection circuitry.

Single mounting point. One IMU measures acceleration at one location on the vehicle chassis. An impact at the opposite end of the vehicle is attenuated by the chassis structure between the impact point and the sensor. This is unlikely to cause missed detections for high-energy collisions but may affect borderline cases like low-speed corner impacts.

X. FUTURE SCOPE

Several practical extensions follow directly from the limitations identified above.

Replacing the fixed threshold with a speed-adaptive one is the most immediate improvement. The GPS module already supplies vehicle speed as a field in the GPRMC sentence. A threshold function $T_{acc}(v)$ that rises with speed would reduce false positives on rough rural roads (where speed is low and bumps are frequent) while remaining sensitive at highway speeds. This requires no additional hardware.

A lightweight TensorFlow Lite model trained on the accelerometer dataset could replace or augment the threshold comparator. Running on the ESP32's second core, a small binary classifier could distinguish collision impulses from pothole or rail-crossing vibrations more reliably than a scalar threshold, as demonstrated by Rathod et al. [4] at a coarser scale.

On the connectivity side, adding a SIM7600 4G module connected to the ESP32 via UART would provide independent cellular uplink. The firmware could prefer Wi-Fi for cost reasons and fall back to 4G only when Wi-Fi is unavailable, keeping data charges minimal.

Blockchain-based record integrity, as demonstrated by Joseph et al. [3], would be valuable once the system matures beyond the prototype stage. Accident records stored on a permissioned ledger carry stronger evidentiary weight in insurance disputes than records in a single-vendor cloud database.

A companion mobile application could let the vehicle owner review their own accident history, receive an alert when a family member's vehicle triggers a detection, or manually suppress a false positive before insurance is notified. Multi-sensor fusion — integrating an OBD-II reader for engine speed and brake pressure — would provide corroborating data that further reduces false positives and improves severity estimation.

XI. CONCLUSION

Road accident fatality statistics in India and globally remain difficult to reduce through passive infrastructure alone. Active, automated post-accident notification addresses a concrete gap: the minutes between a crash and the arrival of help. This paper reviewed five systems that attack various parts of this problem and proposed a hardware prototype that combines the practical attributes — low cost, cloud connectivity, GPS, and insurance notification — that no single reviewed system provides in one package.

Bench testing showed 96.7% detection accuracy, zero false positives over a 60-minute normal-driving trial, a mean location error of 2.8 m, and a mean upload latency of 1.4 s. These figures are encouraging for a threshold-based system without any machine learning component. The known limitations — fixed threshold, GPS gaps, Wi-Fi dependency — are real but each has a well-defined engineering remedy that future work can address incrementally. Cellular fallback and a speed-adaptive threshold are the two changes most likely to improve reliability in real-world deployment.

ACKNOWLEDGMENT

The authors thank Mrs. B. Uma, Assistant Professor, Department of CSE, Malnad College of Engineering, Hassan, for her guidance throughout this project. Thanks are due to the department faculty for laboratory access and to the researchers whose published work is cited in this review.

REFERENCES

- [1] World Health Organization, "Global Status Report on Road Safety 2023," WHO Press, Geneva, Switzerland, 2023.
- [2] S. K. Patel and M. R. Shah, "Design and Implementation of an IoT-Enabled Vehicular Accident Detection and Notification System," *Int. J. Advanced Research in Computer Engineering and Technology*, vol. 11, no. 3, pp. 210–216, Mar. 2023.
- [3] A. Joseph, R. Kumar, and P. Nair, "Integrating AI, IoT and Blockchain for Real-Time Insurance Claim Automation," in *Proc. Int. Conf. on Smart Computing and Communication Systems (ICSCC)*, pp. 95–101, 2024.
- [4] G. S. Rathod, R. C. Tipale, and K. Jajulwar, "Intelligent Accident Detection and Alerting System based on Machine Learning over the IoT Network," in *Proc. 2022 Int. Conf. on Futuristic Technologies (INCOFT)*, pp. 1–6, Nov. 2022.
- [5] Smart City Research Group, "Investigating Action Recognition Approaches for Accident Detection in Smart City Transport Systems," in *Proc. IEEE Smart City Conf.*, pp. 33–39, 2023.
- [6] R. Singh, "Vehicle Tracking and Accident Alert Using GPS and GSM," in *Proc. IEEE Conf. on Smart Systems*, pp. 45–50, 2023.
- [7] A. Kumar and S. Sharma, "IoT-Based Smart Accident Detection System," *Int. J. Engineering Research*, vol. 8, no. 4, pp. 120–125, 2024.
- [8] P. Verma, "Automation in Vehicle Insurance Using IoT," in *Proc. Int. Conf. on IoT Applications*, pp. 78–82, 2024.
- [9] M. Hart, "TinyGPS++ Library for Arduino," [Online]. Available: <https://github.com/mikalhart/TinyGPSPlus>, Accessed: May 2026.
- [10] Mobitz Inc., "Firebase ESP Client Library," [Online]. Available: <https://github.com/mobizt/Firebase-ESP-Client>, Accessed: May 2026.