AcciVision: IoT Powered Smart Accident Rescue and Injury Assessment System

Varsha M, M Sri Madhura

How to Cite this Article:
Madhura, M. S. & M, V. (2026). AcciVision: IoT Powered Smart Accident Rescue and Injury Assessment System. International Journal of Creative and Open Research in Engineering and Management, 02(05). https://doi.org/10.55041/ijcope.v2i5.165

https://doi.org/10.55041/ijcope.v2i5.165

ABSTRACT:

Death does not always occur at the point of impact in traffic accidents. It frequently occurs later, while you're waiting for assistance. No one made a call. No one took notice. It's too late by then. With this project, we attempted to solve that issue.
We developed AcciVision. It uses sensors installed in the car to record a collision as it occurs. It doesn't need to be reported. The location is sent directly to a cloud database, which selects the closest free ambulance and provides the driver with a route. All of this without a single button being pressed.

An image-based injury check was also added. When a person on the scene snaps a picture and uploads it, the system indicates which kind of hospital they should go to. A route is prepared for that as well. Testing revealed that the crash detection system functioned properly and that ambulances were dispatched in a matter of seconds. This approach improves response time and helps lives

KEYWORDS: Accident Detection, Internet of Things, Emergency Response, Ambulance Routing, Computer Vision, Firebase, GPS Tracking, Smart Healthcare, Automated Dispatch

1. INTRODUCTION

1.1 Background

Every year, a huge number of people die in traffic accidents worldwide. Many of those deaths were not inevitable. A quicker medical response would have altered the result. In emergency care, there is even a concept known as the "golden hour," which refers to the short time after a serious injury when treatment is most effective.

One main reason for the delay is the need for someone to report the accidents. If victim is unconscious, they cannot make a call. In some cases, no one may be nearby to notice it. Even after it is reported, coordination between ambulance service and hospitals takes time and the information is not complete. This delay can affect survival.

Today, several technologies such as sensors, cloud system and real-time data are available. However, most solution focus on only one part of the problem. Some system detect accidents, while other handle ambulance services or routing. In most cases, routing happens only after a manual report. Only a few systems combine all these features into a single solution.

1.2 Objectives

Our project's objective was to create an autonomous emergency response system in the event of a collision. Use the car's sensors to detect the collision, transmit the location to a server, locate the closest ambulance, obtain a route to the scene, examine the victim's photos to determine the extent of their injuries, select the appropriate hospital, and provide the driver with a route there as well. There is no need for calls or manual reporting; everything is done automatically.

2. LITERATURE REVIEW

We examined previous work in this field before developing AcciVision. The majority of the work that has been done on accident detection and emergency response focuses on a single aspect at a time.

Aslam et al. [1] proposed an IoT-based vehicle accident detection system using ESP32 and on-vehicle sensors to automatically detect crashes and transmit a visual situation report to a remote server. However, automatic ambulance assignment and route optimization are not addressed.

Ayesha and Chakravarthi [2] developed a smart ambulance system that detects accidents using vehicle-mounted IoT sensors and notifies the nearest ambulance with location data. While ambulance notification is handled, optimized route computation and injury-based hospital selection are not included.

Bochare et al. [3] presented an IoT-based accident detection and tracking system using GPS that transmits location to emergency contacts automatically upon crash detection. However, ambulance assignment and route optimization are not addressed.

Edna Elizabeth et al. [4] developed an accident detection and healthcare notification system using ESP32 that alerts nearby healthcare units on crash detection. Although notification is effective, optimal routing and injury-based hospital recommendation are not covered.

An IoT-based accident detection and automated emergency notification system was proposed in [5] that triggers alerts to emergency contacts using sensor data. However, ambulance dispatch optimization, routing, and injury-based assessment are not addressed.

Considering everything, there is good work being done, but it is dispersed. Detection, ambulance dispatch, routing, and injury analysis are not all covered by a single system. We've tried to consolidate all of that into AcciVision.

Table 1: Comparative Analysis of Existing Accident Detection and Emergency Response Systems

Feature	Sarker [1]	Pathik [2]	Ali [3]	AcciVision
Auto Accident Detection	✓	✓	✓	✓
GPS Location Tracking	✓	✗	✓	✓
Ambulance Assignment	✗	✗	✗	✓
Route Optimization	✗	✗	✗	✓
Injury Analysis (CV)	✗	✗	✗	✓
Hospital Recommendation	✗	✗	✗	✓
Fully Automatic Pipeline	✗	✗	✗	✓

As presented in Table 1, existing systems in the literature address only isolated components of the emergency response pipeline. None of the reviewed works combine automatic accident detection, ambulance assignment, route optimization, computer vision based injury analysis, and hospital recommendation in a single integrated system. AcciVision addresses all of these together, making it the most complete end-to-end solution among the compared approaches.

3. METHODOLOGY

The foundation of AcciVision is the notion that the entire emergency response procedure ought to be automated. For the system to function, no one should need to do anything. The system should take over as soon as an accident occurs. This is made possible by the system's five interconnected components.

3.1 System Architecture

A tiny hardware unit with an ESP32 chip and an MPU6050 sensor is located inside the car. That sensor continuously monitors the movement of the car. Anything that appears to be an abnormal tilt or a sudden jerk is flagged. When that occurs, the data is sent to Firebase and the GPS coordinates are obtained.

A Python program, on the other hand, is constantly listening to that database. As soon as new accident data is received, it determines the best route, selects the closest ambulance from a list, and performs an injury analysis if the victim's photo is uploaded. The entire process takes place in a matter of seconds.

As illustrated in Figure 1, the end-to-end flow of AcciVision covers everything from hardware sensing to patient delivery.

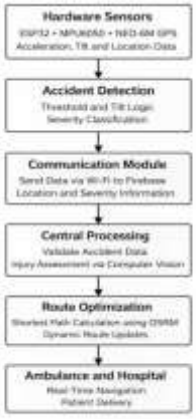

Figure 1: System Architecture of AcciVision showing the end-to-end flow from hardware sensor data collection to ambulance dispatch and patient delivery.

3.2 Accident Detection Module

There are two components to the MPU6050 sensor. One gauges the vehicle's speed in various directions. The other gauges the vehicle's rotation or tilting. These readings remain within a typical range when an automobile is driven normally. The readings abruptly increase during a collision, and the car may tilt at an odd angle.

To obtain a single figure that represents the total force of movement, we apply this formula:
$A = \sqrt{(ax^2 + ay^2 + az^2)}$

The individual readings along each axis are ax, ay, and az. The system flags A when it exceeds our predetermined threshold. At the same time, the gyroscope tilt value is examined. Before a crash is confirmed, both must go beyond their limits. Normal bumps and hard stops are prevented from setting off false alarms by that two condition check.

The system determines a severity level based on the force value after confirming a crash. Higher equates to worse. One record that is sent to Firebase contains that severity, the GPS coordinates, and a timestamp.

3.3 Communication and Cloud Data Management

The ESP32 transmits the accident record to Firebase via Wi-Fi. The primary reason we chose Firebase is that it doesn't require polling. The server side program is immediately informed when new data is received. No waiting, no double-checking.

Every record includes details such as time, location and severity. Firebase maintains an ambulance data including with current positions and availability status in addition to accident records. Storing all this information in one place helps avoid delays and remove the need for additional communication.

3.4 Ambulance Selection and Route Optimization

The Python program reads new accident data as soon as it is uploaded to Firebase. It then measures the distance between the accident location and available ambulance and assigns the nearest one.

We used OSRM for routing. It is an open-source program that uses actual road map data. After sending the coordinates, the system returns a suitable path, which is directly provided to the driver. After the patient is picked up, the system generates another route from the accident location to the recommended hospital. This route is automatically sent to the driver.

3.5 Computer Vision Based Injury Analysis

A team member captures a picture of the victim, and the system analyses it to detect signs of injury such as bleeding or visible damage. Based on analysis the system identifies the type of injury and suggests an appropriate hospital.

3.6 System Integration and Real Time Workflow

This section describes how the system works step by step. The sensors monitor the vehicle and when an accident is detected, the ESP32 send the data to firebase. The system reads the location, finds the nearest ambulance and generates a route using OSRM. The driver receives the route automatically. If an image is uploaded, the system analyses the injury and suggests a hospital. After pickup, another route to the hospital is provided. The process takes only a few seconds.

3.7 Hardware Setup

The hardware setup consists of sensors and a microcontroller integrated into the vehicle. The system uses an ESP32 with an MPU6050 sensor to detect motion. A GPS module records the location at the time of the accident. When the sensor value crosses a set threshold, the ESP32 send the data to firebase using wi-fi. The setup is compact and can be easily fit inside any vehicle

As shown in Figure 2, the hardware connections used in the AcciVision prototype include the MPU6050 connected to the ESP32 using the I2C interface with (SDA on GPIO21, SCL on GPIO22) and the NEO-6M GPS module connected through UART using (GPIO16 and GPIO17). Both modules draw power from the ESP32's 3.3V and GND pins.

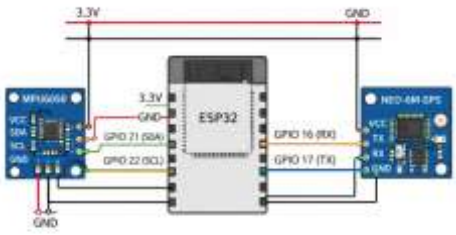

Figure 2: Shows the hardware wiring diagram of the AcciVision prototype. The ESP32 microcontroller is connected to the MPU6050 using I2C interface (GPIO21 - SDA, GPIO22 - SCL) and the NEO-6M GPS module is connected using UART (GPIO16 - RX, GPIO17 - TX).

4. RESULTS

In order to test the system, different accident scenarios were simulated. Three things were examined: how quickly the entire system responded from detection to dispatch, how well the ambulance routing functioned, and whether the accident detection produced the correct output.

4.1 Experimental Setup

The hardware setup used an ESP32 along with an MPU6050 sensor. The system was tested under different conditions to check how well it detects accidents. Both ambulance and accident details were stored in Firebase. A Python program handled the workflow and selected ambulances. OSRM was used for route calculation, and urban map data was used during testing.

As shown in Figure 3, all accident records created during testing were stored in the Firebase Realtime Database. Each record had an unique ID and also included ambulance details. This shows that communication between the ESP32 and Firebase worked properly during testing.

Figure 3: Displays the Firebase Realtime Database, which contains multiple accident records (ACC1 to ACC5) maintained throughout the system's testing.

4.2 Accident Detection Performance

The system was tested in four scenarios: normal driving, braking, collision and vehicle tilt. No false alerts were observed during normal driving and braking. In the collision and tilt cases, the system were detected correctly. This shows that the threshold-based detection works properly.

Table 2: Accident Detection Performance

Test Scenario	Sensor Reading	Detection Result
Normal Driving	Stable values	No accident
Hard Braking	Moderate change	No accident
Collision	High spike in reading	Accident detected
Vehicle Tilt	Unusual tilt angle	Accident detected

This table clearly shows how the system distinguishes between normal vehicle movement and actual accident conditions without giving false alerts.

The complete accident record for ACC5 is shown in figure 4. It includes details like pickup location with GPS coordinates, severity marked as moderate and injury type identified as general. The system also assigned Dr. Maibbar Hospital as the destination. The CV status is marked as processed.

Figure 4: Shows the detailed accident record in Firebase which includes injury type, hospital assignment, CV processing status, GPS coordinates, severity level, and final status with timestamp.

Figure 5 shows the system dashboard at the time an accident is detected. The system automatically identifies the nearest available ambulance as A1 and the accident location is displayed on the map. At this stage no manual input is required. The operator only needs to confirm the dispatch, after which of the process continues automatically.

Figure 5: Shows the dashboard with a new accident alert received for Zone 9 Teynampet. The nearest ambulance A1 is automatically selected and the accident location is marked on the map.

4.3 Ambulance Routing Performance

The routing was tested under three different scenarios by placing ambulances at different distances. OSRM was used to select the nearest ambulance and generate routes. In all cases, the system selects the nearest ambulance and the travel time was reasonable.

As shown in Table 3, the nearest ambulance was selected in all three cases and the travel time was reasonable.

Table 3: Ambulance Routing Performance

Scenario	Distance to Accident	Ambulance Sent	Travel Time
Scenario 1	2.1 km	A1	4 minutes
Scenario 2	3.5 km	A2	6 minutes
Scenario 3	1.8 km	A1	3 minutes

This table shows the distance and travel time for each scenarios confirming that the system consistently selected the nearest available ambulance and routing logic works consistently.

Table 4: AcciVision End-to-End System Response Time Breakdown

Step	Time Taken
Crash detection to Firebase upload	~1 second
Firebase to ambulance assignment	~1 second
OSRM route calculation	~2 seconds
CV injury analysis	~3 seconds
Total end-to-end response time	~7 seconds

Table 4 shows the time taken at each stage of the system. The total response time from crash detection to ambulance dispatch is around 7 seconds. Each step adds only a small delay, which shows that the system can respond quickly in emergency situations. Since quick response is important during the 'golden hour', this approach reduces delays that usually occur to manual reporting.

Figure 6 shows how the route is generated from ambulance to the accident location. Once the ambulance is assigned, the system calculates the shortest path and displays it on the map in real time. The estimated pickup time was 6 minutes and 28 seconds. When the ambulance reaches the location, the status is updated, showing that the routing works correctly.

Figure 6: Route from ambulance A1 to the accident location in Zone 9 Teynampet, with estimated arrival time.

Figure 7 shows the result after uploading the injury image. The system identifies the injury as general with moderate severity and automatically assigns Dr. Maibbar Hospital. The hospital location is retrieved from the map data, and system updates the status to routing, which start the next route calculation.

Figure 7: Output after injury analysis showing GENERAL injury(moderate), assigned hospital, and updated system status.

4.4 Overall System Response

The system response was measured to see how quickly an ambulance could be assigned after detecting an accident. It includes the time taken for accident detection, transfer to firebase, processing by the python program and route generation. Each step add only a small delay, so the total response time remains low. The system reacts quickly without depending on manual reporting. This reduces delay and helps emergency services to respond faster.

Figure 8 shows the final state of the dispatch for accident ACC5. The system completed the full process, including pickup and hospital routing. The map shows the route from the accident location to Dr. Maibbar Hospital, and the status is marked as Completed. This shows that the system handled the entire process without manual intervention at any step.

Figure 8: Hospital route from the accident location with final dispatch status marked as completed.

Figure 9 shows the time taken at each stage of the system. The CV-based injury analysis takes the most time around(3 seconds) because of image processing, while detection and ambulance assignment are faster. The total response time of about 7 seconds is much faster compared to manual reporting, which usually takes several minutes.

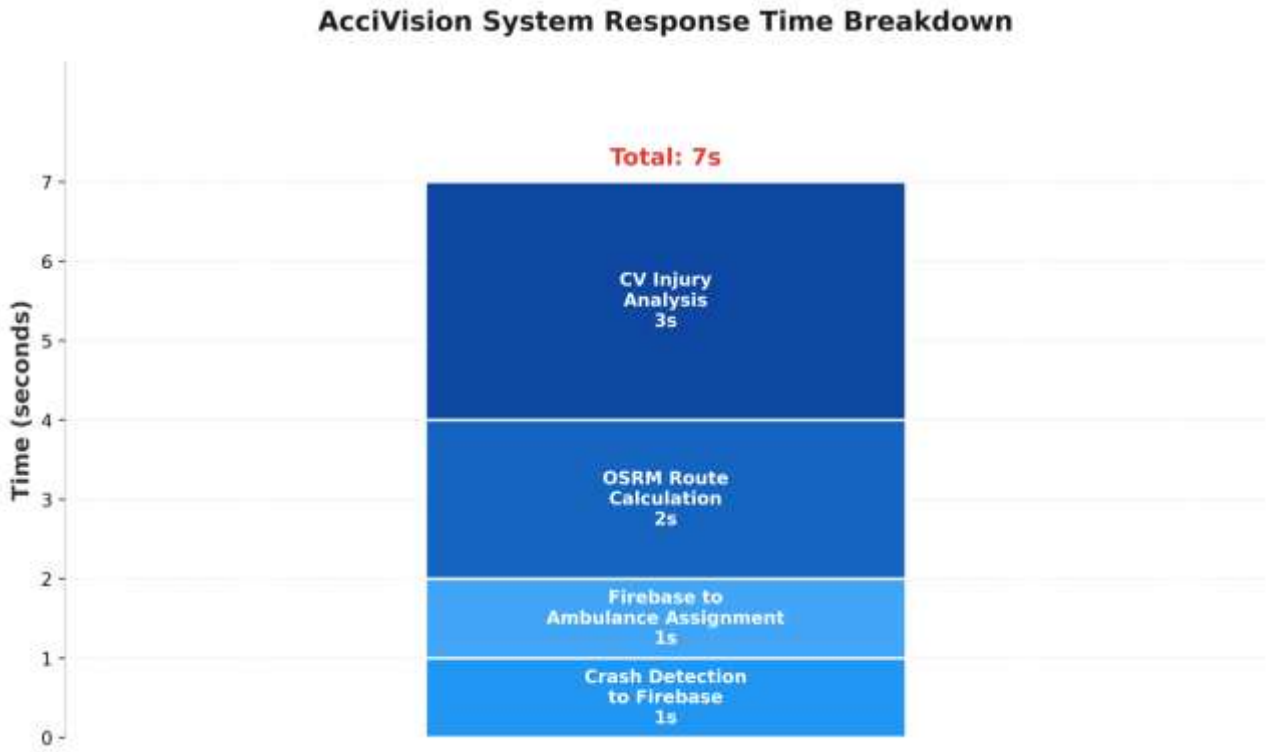

Figure 9 presents the response time from Table 4 in a visual form, of how each stage contributes to the total time.

5. CONCLUSION

AcciVision began with a simple query: Why must someone report an accident before assistance can be provided? We designed a system based on the notion that it shouldn't operate in that manner. The crash is detected by sensors, the coordination is handled by the cloud, the navigation is handled by routing, and a vision module assists in selecting the appropriate hospital. No one has to start any of this.

Our tests showed that the detection performed well in a variety of situations. During regular driving, there were no false alarms, and collisions were detected when they should have been. Routing always went to the closest ambulance. Every time, the entire process was completed in a matter of seconds.

We believe AcciVision presents a compelling case for developing emergency systems that don't require human intervention. Eliminating the reporting step completely could mean the difference between life and death in circumstances where every minute counts.

6. FUTURE WORK

We would like to make some improvements in the future. The system needs to be tested in real vehicles under actual road conditions. While lab testing gives a basic idea, real-world driving is more unpredictable and complex ,so further testing is necessary

The routing system can be improved by including real-time traffic information. Currently, it selects routes based only on map information, which may not always provide the fastest path. It is unaware of the state of the traffic. In practice, those routes would be far more helpful with the addition of a live traffic feed, particularly in urban areas during peak hours.

The injury analysis module can be improved by training it with a larger set of medical images. This can improve the accuracy of injury detection and make the suggestions better.

Additionally, the current prototype uses Wi-Fi for cloud communication, which restricts deployment to network-covered areas. Future iterations would be able to operate in remote areas by incorporating a GSM module.

7. REFERENCES

[1] S. Aslam, S. Islam, N. Nigar, S. A. Ajagbe, and M. O. Adigun, "An IoT-Based Automatic Vehicle Accident Detection and Visual Situation Reporting System," *Journal of Advanced Transportation*, Wiley, vol. 2024, Article ID 4719669, 14 pages, 2024. DOI: https://pdfs.semanticscholar.org/42cd/4ef3f4a9db2e2408ac5b0dcb2c78103a9319.pdf

[2] A. Ayesha and K. Chakravarthi, "Smart Ambulances for IoT Based Accident Detection, Tracking and Response," *Journal of Computer Science*, vol. 19, no. 6, pp. 677–685, 2023. DOI: https://doi.org/10.3844/jcssp.2023.677.685

[3] S. P. Bochare, A. G. Feran, S. S. Naswale, P. S. Satao, and N. Y. Mhaisagar, "IoT Based Vehicle Accident Detection and Tracking System Using GPS," *International Research Journal of Engineering and Technology (IRJET)*, vol. 11, no. 4, pp. 494–498, 2024. Link: https://www.irjet.net/archives/V11/i4/IRJET-V11I479.pdf

[4] N. Edna Elizabeth et al., "Automatic Vehicle Accident Detection and Healthcare Unit Notification Using IoT Technology with ESP32," *International Journal of Innovative Research in Multidisciplinary Physical Sciences (IJIRMPS)*, vol. 10, no. 4, 2022. Link: https://www.ijirmps.org/research-paper.php?id=1572

[5] Anonymous et al., "IoT-Based Vehicle Accident Detection and Automated Emergency Notification System," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 15, no. 1, 2026. Link: https://ijarcce.com/wp-content/uploads/2026/01/IJARCCE.2026.15109-IoT.pdf

[6] Anonymous et al., "Sustained Approach for Accident Detection and Rescue Alerting System," *E3S Conferences*, vol. 2024, no. 01035, 2024. Link: https://www.e3s-conferences.org/articles/e3sconf/pdf/2024/37/e3sconf_icftest2024_01035.pdf

[7] R. Natarajan et al., "IoT Based Smart Emergency Response System (SERS) for Monitoring Vehicle, Home and Health Status," *Discover Internet of Things*, Springer, vol. 4, no. 73, 2024. DOI: https://doi.org/10.1007/s43926-024-00073-6

[8] Anonymous et al., "An IoT-Based Emergency Detection, Alerting, and Rescue System (Smart-ResQ)," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 7, no. 6, 2025. Link: https://www.ijfmr.com/papers/2025/6/60839.pdf

[9] J. Jovanovic et al., "A Novel Internet of Things-Enabled Accident Detection and Reporting System for Smart City Environments," *Sensors*, MDPI/PMC, vol. 19, no. 2064, 2019. Link: https://pmc.ncbi.nlm.nih.gov/articles/PMC6540187/

[10] D. Luxen and C. Vetter, "Real-Time Routing with OpenStreetMap Data," in *Proc. 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, IL, USA, 2011, pp. 513–516. Link: https://doi.org/10.1145/2093973.2094062

[11] T. Giraud, "osrm: Interface Between R and the OpenStreetMap-Based Routing Service OSRM," *Journal of Open Source Software*, vol. 7, no. 78, p. 4574, 2022. DOI: https://doi.org/10.21105/joss.04574

[12] Anonymous et al., "Ambulance Routing Optimization Based on Emergency Medical Service Availability," *Discover Artificial Intelligence*, Springer, 2025. DOI: https://doi.org/10.1007/s44163-025-00352-3

[13] M. H. Tusar et al., "AI-Powered Image-Based Assessment of Pressure Injuries Using YOLOv8 Models," *Advances in Wound Care*, Mary Ann Liebert, 2025. DOI: https://doi.org/10.1089/wound.2024.0245

[14] J. Terven, D. Cordova-Esparza, and J. A. Romero-Gonzalez, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, MDPI, vol. 5, no. 4, pp. 1680–1716, 2023. DOI: https://doi.org/10.3390/make5040083

[15] Anonymous et al., "Developing Real-Time IoT-Based Public Safety Alert and Emergency Response Systems," *Scientific Reports*, Nature, 2025. Link: https://www.nature.com/articles/s41598-025-13465-7

[16] Anonymous et al., "Cloud-Enabled IoT System for Real-Time Environmental Monitoring and Remote Device Control Using Firebase," *arXiv preprint*, 2026. Link: https://arxiv.org/html/2601.17414v1

[17] P. O. Ayeni and O. C. Adesoba, "IoT-Based Home Control System Using NodeMCU and Firebase," *Journal of Edge Computing*, vol. 4, no. 1, pp. 17–34, 2025. Link: https://acnsci.org/journal/index.php/jec/article/view/814

[18] Anonymous et al., "IoT-Based Elderly Health Monitoring System Using Firebase Cloud Computing," *PMC / Healthcare*, 2025. Link: https://pmc.ncbi.nlm.nih.gov/articles/PMC11873372/

[19] S. H. Li et al., "JustIoT Internet of Things Based on the Firebase Real-Time Database," in *Proc. IEEE International Conference on Applied System Innovation (ICASI)*, 2018, pp. 1–4. DOI: https://ieeexplore.ieee.org/document/8353979

[20] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint*, arXiv:1804.02767, 2018. Link: https://arxiv.org/abs/1804.02767