ACCIDENT DETECTION SYSTEM

*-Sanidhya Kafle*

**Abstract**

Our IoT-based Accident Detection System aims to improve road safety by sending instant alerts to emergency contacts and hospitals when a crash occurs, reducing response time and saving lives. The system uses a ESP32 microcontroller to control all components. An impact sensor detects collisions and triggers a buzzer or LED. At the same time, a GPS module gets the crash location, and a ESP32 sends an e-mail with the coordinates.

Simultaneously, a GPS module collects the real-time location of the crash and a ESP32 module sends an email alert with the GPS coordinates to a predefined emergency contact. The entire system is powered by a buck converter connected to a dual Li-ion battery pack, ensuring stable voltage supply. The model car’s motion is handled by DC motors controlled via an L298N motor driver, though this is primarily for simulation purposes and not the accident detection function itself.

**Table of Contents**

[1. Introduction 1](#_Toc198149090)

[1.1 Current Scenario 1](#_Toc198149091)

[1.2 Problem Statement and Project as a Solution 2](#_Toc198149092)

[1.3 Aim and Objectives 2](#_Toc198149093)

[2. Background 3](#_Toc198149094)

[2.1 System Overview 3](#_Toc198149095)

[2.2 Design Diagram 3](#_Toc198149096)

[2.2.1 Circuit Diagram 3](#_Toc198149097)

[2.2.2 Block Diagram 4](#_Toc198149098)

[2.2.4 Flow Chart 4](#_Toc198149099)

[2.3 Requirement analysis 5](#_Toc198149100)

[2.3.1 Hardware Requirement 5](#_Toc198149101)

[2.3.2 Software Requirements 8](#_Toc198149102)

[3. Development 9](#_Toc198149103)

[3.1 Planning and Design 9](#_Toc198149104)

[3.2 Resource Collection 9](#_Toc198149105)

[3.3 System Development 10](#_Toc198149106)

[Phase 1: Motor and Chassis Setup: 10](#_Toc198149107)

[Phase 2: Motor Driver Installation (L298N) 10](#_Toc198149108)

[Phase 3: Microprocessor Connection: 11](#_Toc198149109)

[Phase 4: Power Circuit: 11](#_Toc198149110)

[Phase 5: Battery Installation: 12](#_Toc198149111)

[Phase 6: Connecting Jumper Wires: 12](#_Toc198149112)

[Phase 7: Buck Setup for another ESP32: 13](#_Toc198149113)

[Phase 8: Adding the GPS: 13](#_Toc198149114)

[Phase 9: Impact Switch with Buzzer: 14](#_Toc198149115)

[Phase 10: Uploading code to ESP32: 14](#_Toc198149116)

[4. Results and Findings 15](#_Toc198149117)

[4.1 Results 15](#_Toc198149118)

[4.2 Findings 15](#_Toc198149119)

[5. Future Work 18](#_Toc198149120)

[6. Conclusion 19](#_Toc198149121)

[References 20](#_Toc198149122)

[Appendix 24](#_Toc198149123)

**Table of Figures**

[Figure 1: Circuit diagram of Accident Detection System 3](#_Toc198148232)

[Figure 2: Block diagram for Accident detection system robotic car. 4](#_Toc198148233)

[Figure 3: Flowchart for accident detection system 4](#_Toc198148234)

[Figure 4: ESP32 5](#_Toc198148235)

[Figure 5: Impact Switch 5](#_Toc198148236)

[Figure 6: GPS Module 6](#_Toc198148237)

[Figure 7: DC buck converter 6](#_Toc198148238)

[Figure 8: L289N motor driver 7](#_Toc198148239)

[Figure 9: Lithium Ion battery 7](#_Toc198148240)

[Figure 10: Jumper Wires 7](#_Toc198148241)

[Figure 11: Connecting wires with DC motors 10](#_Toc198148242)

[Figure 12: Wires are plugged inside motor driver 10](#_Toc198148243)

[Figure 13: ESP32 connecting with motordriver 11](#_Toc198148244)

[Figure 14: Powering ESP32 with buck. 11](#_Toc198148245)

[Figure 15: Lithium-Ion battery is connected with switch 12](#_Toc198148246)

[Figure 16: Connecting wires from all the modules. 12](#_Toc198148247)

[Figure 17: Buck being setup for ESP32 13](#_Toc198148248)

[Figure 18: GPS is connected with buck and ESP32 13](#_Toc198148249)

[Figure 19: Final connection with impact switch and buzzer. 14](#_Toc198148250)

[Figure 20: Code is being uploaded through USB cable 14](#_Toc198148251)

[Figure 21: Arduino Uno with GSM module not functioning. 15](#_Toc198148252)

[Figure 22: ESP32 sending mail to recipients. 16](#_Toc198148253)

[Figure 23: GPS location of actual and sent coordinates 17](#_Toc198148254)

[Figure 24: ESP32 blinking after impact getting triggered. 17](#_Toc198148255)

**List of Tables**

[Table 1: GSM and Arduino replaced by ESP32 15](#_Toc198148545)

[Table 2: ESP32 sending mails. 16](#_Toc198148546)

[Table 3: Testing GPS is sending correct coordinates 16](#_Toc198148547)

[Table 4: Testing LED and Buzzers working or not 17](#_Toc198148548)

# 1. Introduction

The internet of things refers to a network that connects everyday objects like sensors, machines to the internet so they can exchange data with other devices. These devices can be found in homes, offices, or industries, and they work by collecting data and sending it over the internet (Puthur, 2019). When smart devices are connected and can talk to each other in real-time, they can quickly detect an accident and send a message to emergency services (Traca Logic, 2024). This helps reduce delays and can save lives. By enabling smart devices to communicate and act in real-time, IoT offers an opportunity to mitigate the consequences of road accidents through faster detection and notification (Puthur, 2019).

Road accidents are one of the major problems in Nepal, the rise in number of vehicles has led to more frequent accidents each year, with over two thousand fatalities annually and more than 12,000 deaths in the past five years (Chettri, 2024). However, emergency response systems have not kept pace many accident victims do not receive timely medical attention due to delays in reporting and dispatch. This **Accident Detection System** project aims to leverage IoT to bridge that gap in emergency response and reduce the time between a crash and the arrival of help (Kumar, 2024).

### 1.1 Current Scenario

According to a Nepal Police report, there are at least 75 road accidents every day in Nepal, and around seven people lose their lives in them (Kathmandu Post, 2025). Many of these deaths happen because medical help arrives too late. In busy cities or remote places, it can take a long time for someone to report an accident especially if the victims are badly hurt and can’t call for help themselves. With weak infrastructure and no automatic alert systems, accident victims often depend on strangers to inform the authorities. This delay can be deadly, as the chances of survival decrease the longer it takes to get medical care (Gupta, 2025). According to studies, reducing the accident response time even by a single minute can save about 6% of lives underscoring how critical quick reporting is to saving lives (Sund, 2018).

### 1.2 Problem Statement and Project as a Solution

Many accidents go unreported as there may be no witnesses or the victims themselves are not in the state to call for help. This delay in informing emergency services often end up being the reason behind slow medical response which increases the risk of fatalities (Sund, 2018). The lack of a quick and automated way to detect accidents and notify first responders is a significant gap in the current road safety scenario (Kumar, 2024).

The **Accident Detection System** project addresses this problem by using an IoT-enabled device installed in the vehicle that can automatically detect when a serious accident has occurred and instantly notify emergency responders. The system uses sensors to detect a collision impact and a communication module to send out an alert containing crucial information such as the crash location (GPS coordinates) and time. The notification can be sent via the internet communication platform used Gmail, which then alerts the nearest hospital or ambulance service. This IoT-based solution effectively acts as an automatic emergency caller for the vehicle. By removing the dependency on eyewitnesses, it ensures that even if an accident happens in an isolated area or during late hours, an alert for help will still be dispatched immediately.

### 1.3 Aim and Objectives

To reduce the time between the occurrence of a road accident and the emergency response by implementing an automated IoT-based accident detection and alerting system.

The following key objectives will be pursued to achieve the project’s aim:

* Develop a reliable accident detection mechanism.
* Establish real-time communication with emergency responders.
* Improve post-accident response to increase survival rates.

# 2. Background

### 2.1 System Overview

The system is a GPS-enabled, four-wheel drive robotic vehicle controlled by an ESP32 microcontroller. It uses an L298N motor driver module to control the four DC motors (two per channel), which are powered by two 18650 Li-ion batteries. The ESP32 receives location data from a GPS module and possibly other sensor inputs (such as a toggle switch and additional modules), allowing it to make decisions and send appropriate signals to the motor driver for navigation. The breadboard connects various components for logic-level voltage distribution and signal routing, creating a compact, wireless-capable, mobile robotic platform suitable for autonomous or semi-autonomous tasks.

### 2.2 Design Diagram

##### 2.2.1 Circuit Diagram

The circuit diagram of Accident Detection System was created in cirkitstudio ide which was really easy for beginner who are trying to create their own circuit diagram.

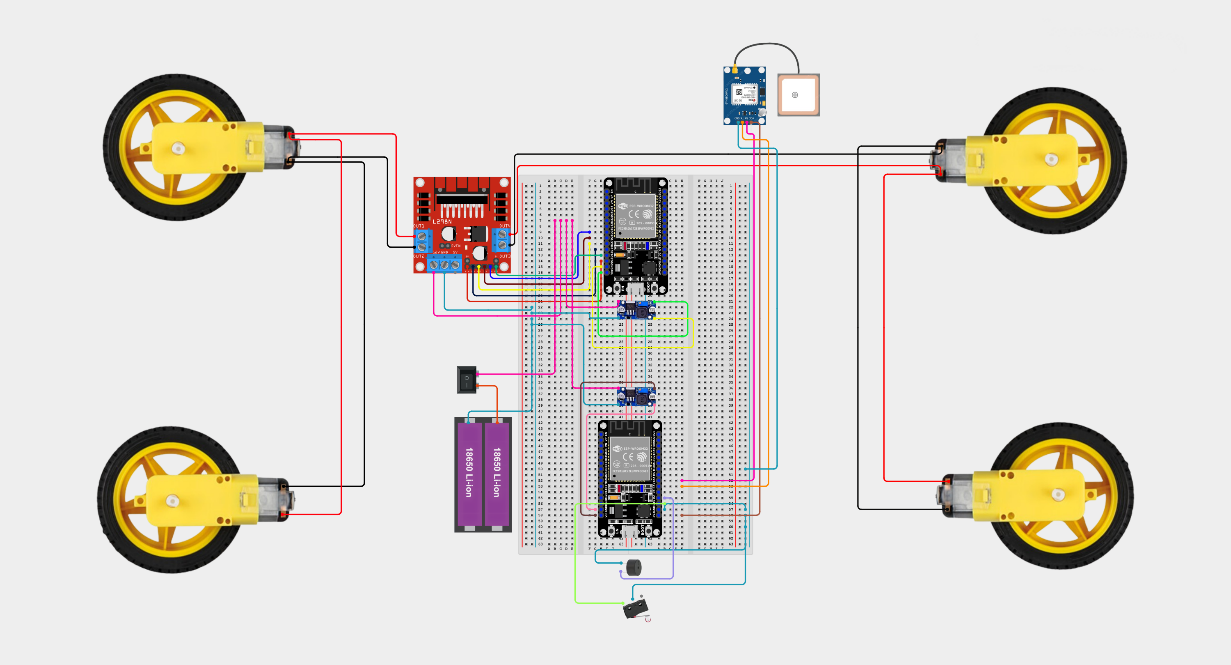


Figure 1: Circuit diagram of Accident Detection System

##### 2.2.2 Block Diagram

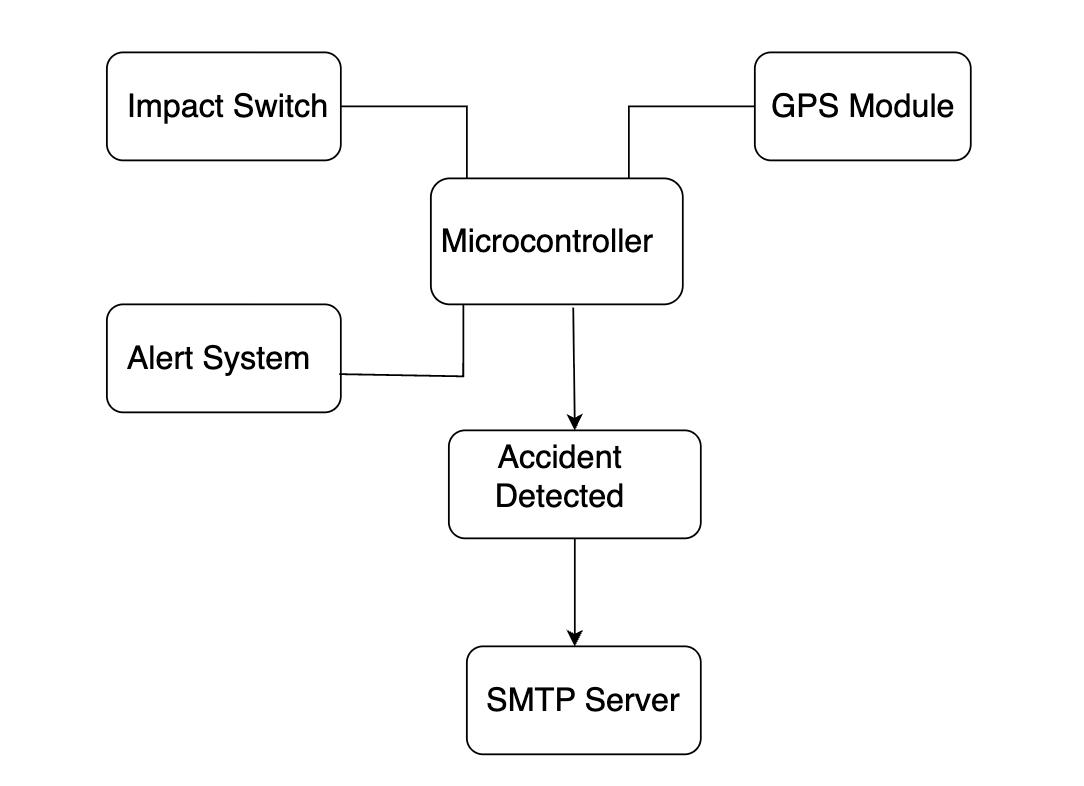


Figure 2: Block diagram for Accident detection system robotic car.

##### 2.2.4 Flow Chart

A diagram of a car

AI-generated content may be incorrect.

Figure 3: Flowchart for accident detection system

## 2.3 Requirement analysis

To build the Accident Detection System, we have listed all the things we need, including hardware parts, software tools, and important libraries. Below is a clear breakdown of each type of requirement we will use in the project:

#### 2.3.1 Hardware Requirement

* ESP32 Microcontroller: The ESP32 development board will serve as the brain of the system. It has built-in Wi-Fi connectivity and enough I/O pins to interface with sensors and modules. The microcontroller reads sensor inputs and controls the alert output and communication modules. The module has been used in this project for sending gps locations through internet (Nabto, 2024).



Figure 4: ESP32 (Cirkit Designer, 2025)

* Impact Switch (Bump Sensor or Accelerometer): A simple bump switch sensor (mounted on the front of the model car) will detect collisions by closing the circuit when pressed. This simulates an impact detection. (In an extended design, a 3-axis accelerometer like the MPU-6050 could be used to detect sudden acceleration changes from any direction, which would be more akin to real accident detection by measuring g-forces. (Circuits DIY, 2022)



Figure 5: Impact Switch (Cirkit Designer, 2025)

* GPS Module (u-blox NEO-7M): A GPS receiver module provides the geographical location coordinates of the vehicle (Kekre, 2020). When an accident is detected, the ESP32 will query this module for the current latitude and longitude to include in the alert message so that responders know exactly where the accident occurred.



Figure 6: GPS Module (Cirkit Designer, 2025)

* Buck Converter: A buck converter is used in the project to efficiently step down a higher input voltage to a lower, stable voltage required by the ESP32 and GPS module. These components typically operate at 3.3V or 5V, and supplying a higher voltage directly could damage them. The buck converter ensures that the correct voltage is delivered without generating excess heat, making it more efficient than linear regulators. This helps maintain stable operation of the devices while protecting them from overvoltage (Arduino Forum, 2020).



Figure 7: DC buck converter (Cirkit Designer, 2025)

* L298N motor driver: The L298N motor driver is used in the project to control the direction and speed of the motors connected to the ESP32. Since the ESP32 cannot provide enough current to drive motors directly, the L298N acts as an interface between the microcontroller and the motors. It can handle higher voltages and currents, allowing it to power DC motors or stepper motors safely and efficiently. The module receives lowpower control signals from the ESP32 and uses them to switch a higher power supply to the motors, enabling forward, reverse, and speed control using PWM (Pulse Width Modulation). This makes the L298N essential for motor-driven applications in the project (Last Minute Engineers, 2022).

A close-up of a circuit board

AI-generated content may be incorrect.

Figure 8: L289N motor driver (Cirkit Designer, 2025)

* Lithium Ion: A lithium-ion battery is used in the project as a portable power source to supply electrical energy to the components such as the ESP32, GPS module, and motors. It is preferred due to its high energy density, lightweight design, and rechargeable nature, making it ideal for compact and mobile applications. The battery typically provides a voltage of 3.7V per cell and, which can be stepped down or regulated using a buck converter to match the voltage requirements of different modules. (U.S. Department of Energy, 2023).

A blue batteries with wires

AI-generated content may be incorrect.

Figure 9: Lithium-Ion battery (Cirkit Designer, 2025)

* Jumper Wires: Used to connect modules, sensors, and power lines on the breadboard. Essential for building the circuit and making flexible connections. It allows easy testing and rearrangement during development (Wiltronics, 2022).



Figure 10: Jumper Wires (Cirkit Designer, 2025)

#### 2.3.2 Software Requirements

* Arduino IDE: The open-source Arduino Integrated Development Environment will be used to program the ESP32 (which is Arduino-compatible via the ESP32 core). This IDE provides a code editor, compiler, and uploader to flash the microcontroller with the written program (Ubidots, 2022).
* Programming Language: In the project, HTML and JavaScript are used within the Arduino IDE to create a web-based user interface that interacts with the ESP32 over Wi-Fi. HTML is used to design the structure of the web page, while JavaScript adds interactivity and allows communication between the user and the ESP32, which is sending control commands to the motors or displaying GPS data in real time. (Random Nerd Tutorials, 2022).
* Circuit Designer: A circuit designer software is used in the project to create and visualize the electronic circuit layout before actual implementation. It helps in accurately placing and connecting components such as the ESP32, GPS module, L298N motor driver, impact switch, and other modules, ensuring proper wiring and avoiding mistakes. Using circuit design tools and reduces errors during hardware assembly. (hope, 2024).
* Microsoft Word: Microsoft Word is a widely used application for creating and managing documents, offering both simplicity and a broad range of features. It supports everything from casual writing to professional reports, making it suitable for various tasks. It has features like styles, headings, captions, tables of contents, and citation management enabled structured academic writing (geeksforgeeks, 2025).

# 3. Development

This section includes step-by-step process of the development of the project from planning to execution.

### 3.1 Planning and Design

In this initial phase, we considered a number of ideas, weighing their potential for viability, affordability, and potential impact on road safety. We finally decided on developing an IoTbased accident detection system in view of its potential to reduce emergency response time. This involved selection of appropriate hardware components such as impact sensors with a capability to recognize abrupt movement changes and a microcontroller with a capability to process sensor data. The organization of software was also decided, such as how data would be received (input), processed, and sent (output), and a means of emergency services warning. We distributed work among group members in assigning them responsibilities to choose hardware, design software, and design circuits. This collaboration enabled us to structure the planning phase and establish a well-defined roadmap for the next levels of development.

### 3.2 Resource Collection

Once we finished designing, we proceeded to gather all the necessary components. Ensuring that all components were compatible and available was crucial for the subsequent development stages. We gathered a variety of components from local suppliers and online vendors to ensure we had everything needed for successful implementation. The key elements were an Arduino Nano to act as a processing center for the system and an impact sensor to detect abrupt movement characteristic of an accident.

We also purchased a GPS module with advanced antenna to facilitate real-time location in such a way that an alarm will be triggered when an accident occurs. For completeness in our project, we also purchased a breadboard and jumper wires for easy assembly and testing of circuits. We opted to use a rechargeable lithium battery for the power source to introduce portability and reliability to the system. We selected each component carefully in terms of compatibility, performance and value to meet the unique needs of our project.

### 3.3 System Development

##### Phase 1: Motor and Chassis Setup:

Motors were fixed to the wheels and base. Wires were connected for power.

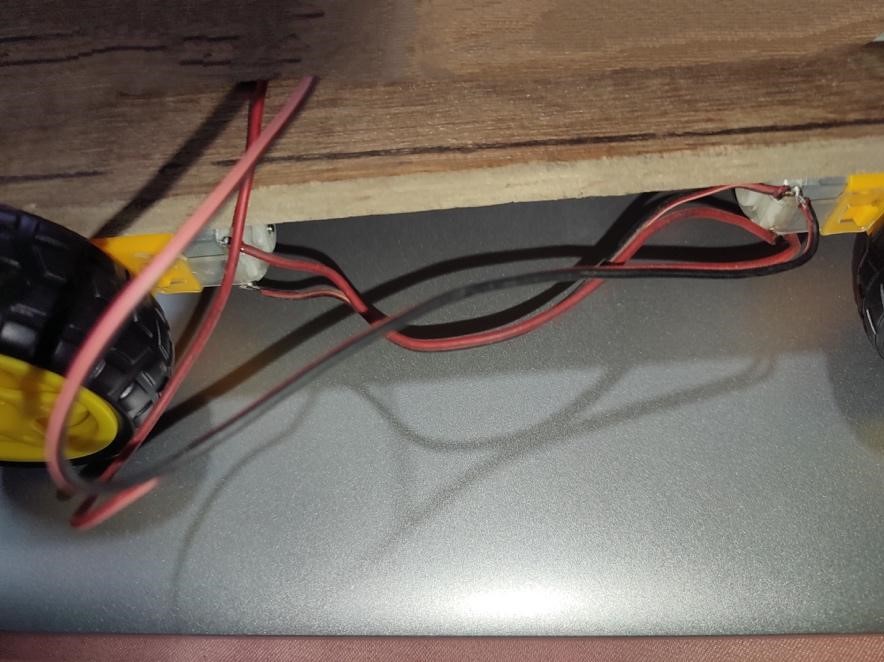


Figure 11: Connecting wires with DC motors

##### Phase 2: Motor Driver Installation (L298N)

This part controls motor movement. It was placed on the breadboard and wired to the motors where the lest side of motor is connected to OUT1 and OUT2 of motor driver and right side of motor is connected to OUT3 and OUT4 of motor driver.

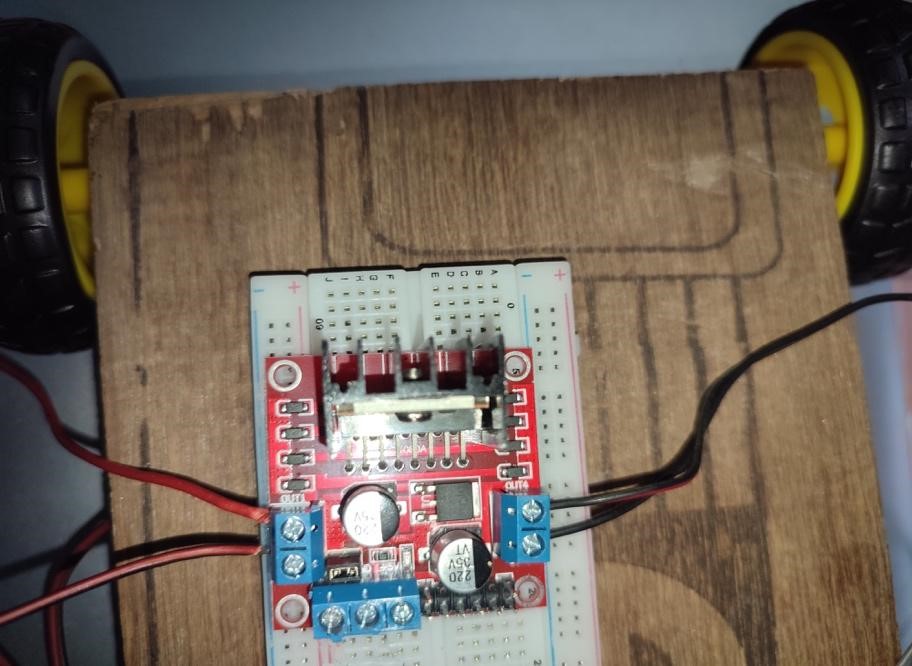


Figure 12: Wires are plugged inside motor driver

*.*

##### Phase 3: Microprocessor Connection:

The main controller (ESP32) was added and linked to the motor driver where motor driver pins of ESP32 are connected followingly: ENA to D13, IN1 to D14, IN2 to D27, IN3 to D26, IN4 to D25, ENB to D12.

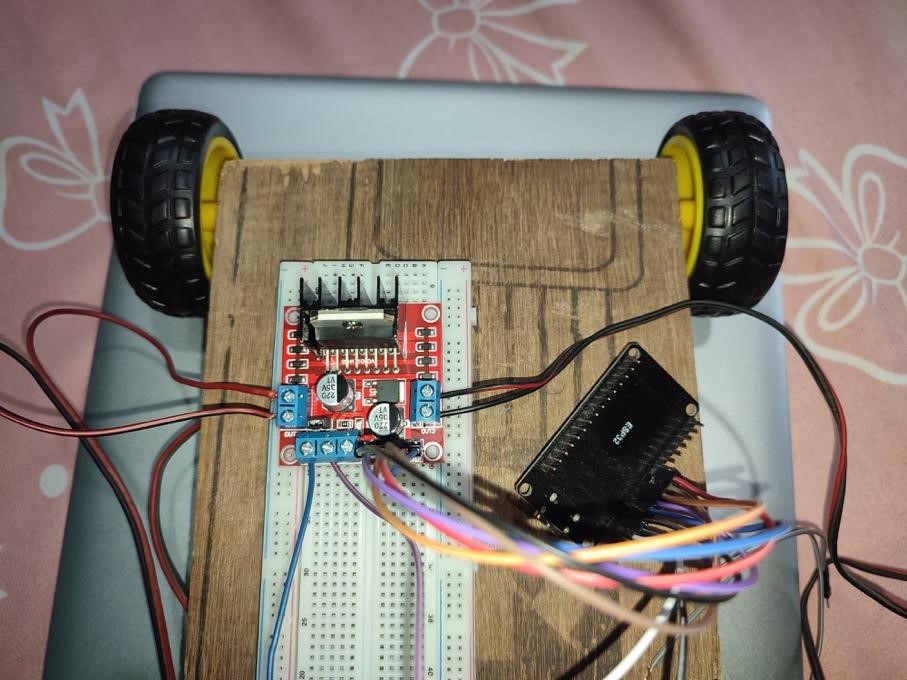


Figure 13: ESP32 connecting with motordriver

##### Phase 4: Power Circuit:

A buck converter was used to give the right voltage to each part where OUT+ of buck is connected to VIN and OUT- of buck is connected to GND.

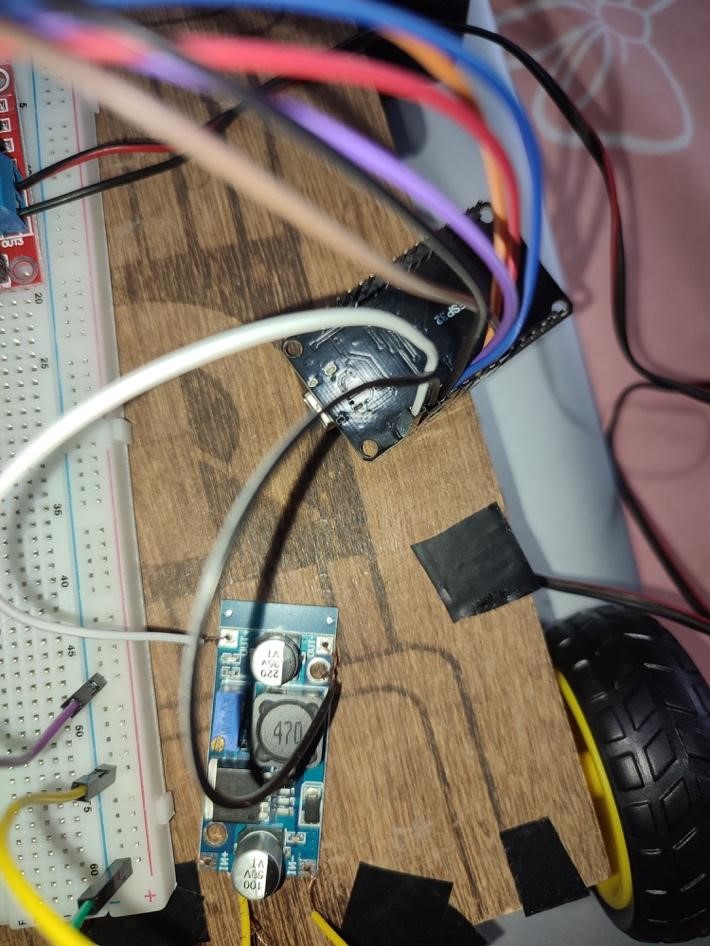


Figure 14: Powering ESP32 with buck.

##### Phase 5: Battery Installation:

Connecting Li-ion battery with switch. Two Li-ion batteries were added to power the where positive is connected to switch and negative to GND of whole system.

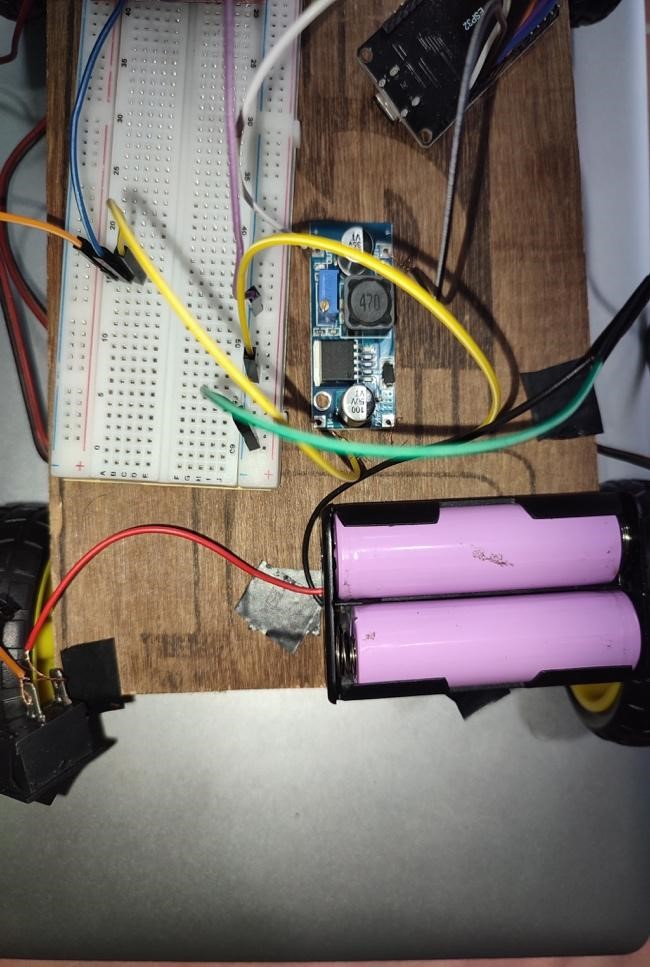


Figure 15: Lithium-Ion battery is connected with switch

##### Phase 6: Connecting Jumper Wires:

All major modules were connected using jumper wires for easy setup.

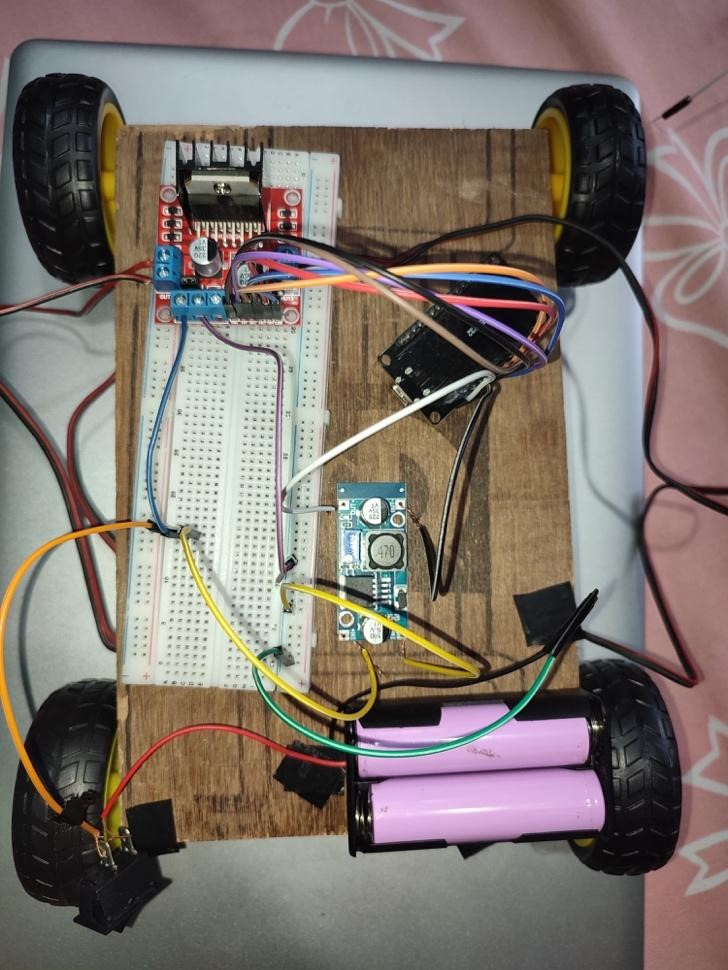


Figure 16: Connecting wires from all the modules.

##### Phase 7: Buck Setup for another ESP32:

The buck converter is being setup for second ESP32 as backup.

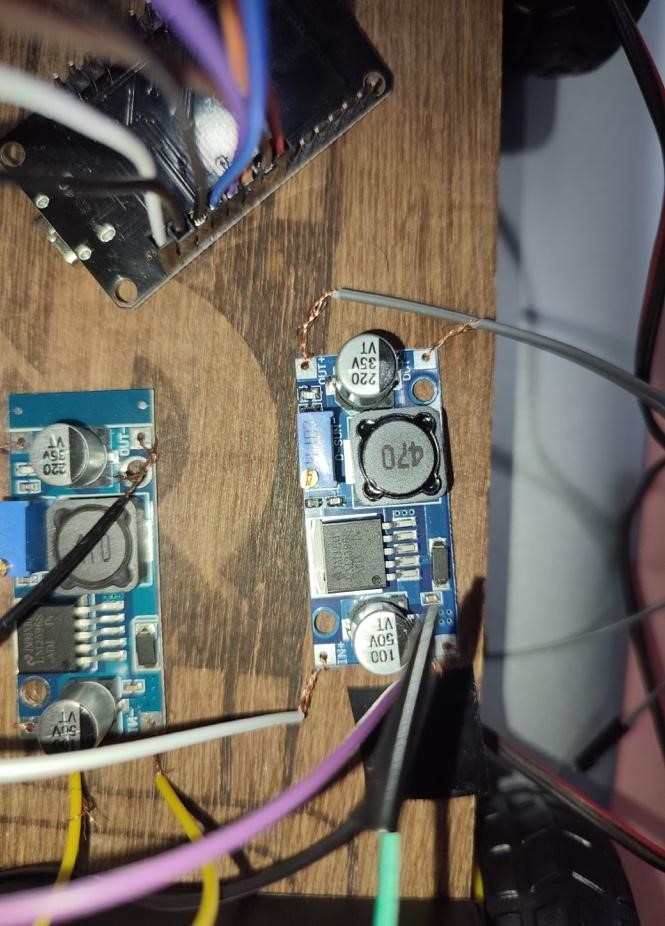


Figure 17: Buck being setup for ESP32

##### Phase 8: Adding the GPS:

The gps is now connected with another ESP32 with buck for location tracing where pins of GPS module are connected with ESP32 pins: VCC to VIN, GND to GND, RX2 to TXD and TX2 to RXD.

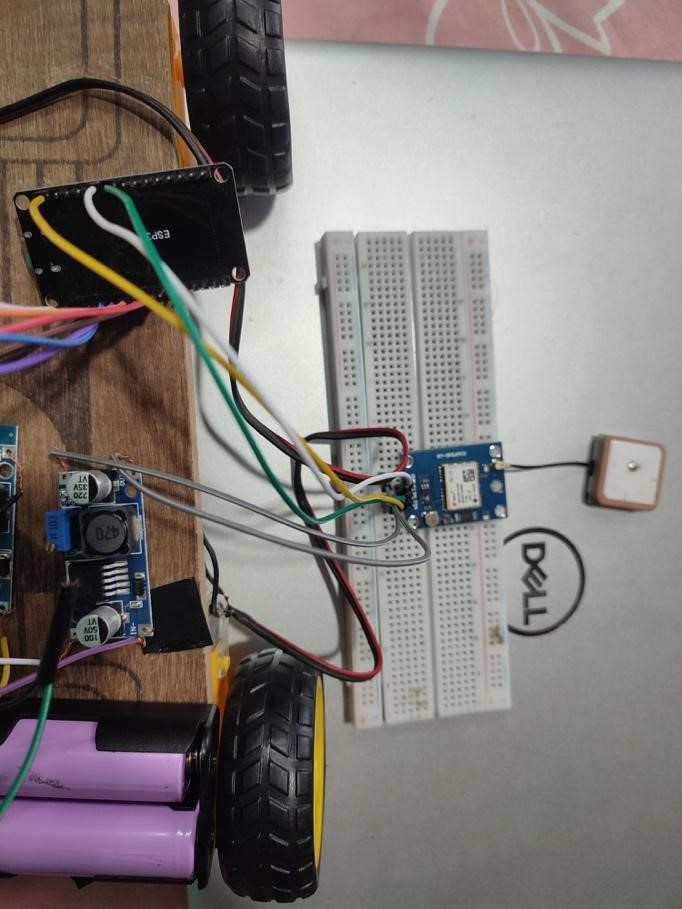


Figure 18: GPS is connected with buck and ESP32

##### Phase 9: Impact Switch with Buzzer:

Connecting impact switch and buzzer with Esp32 where impact switch 1st pin is connected to D15 of ESP32 and 2nd pin to GND and buzzer’s positive pin is connected to D2 and other one to GND.

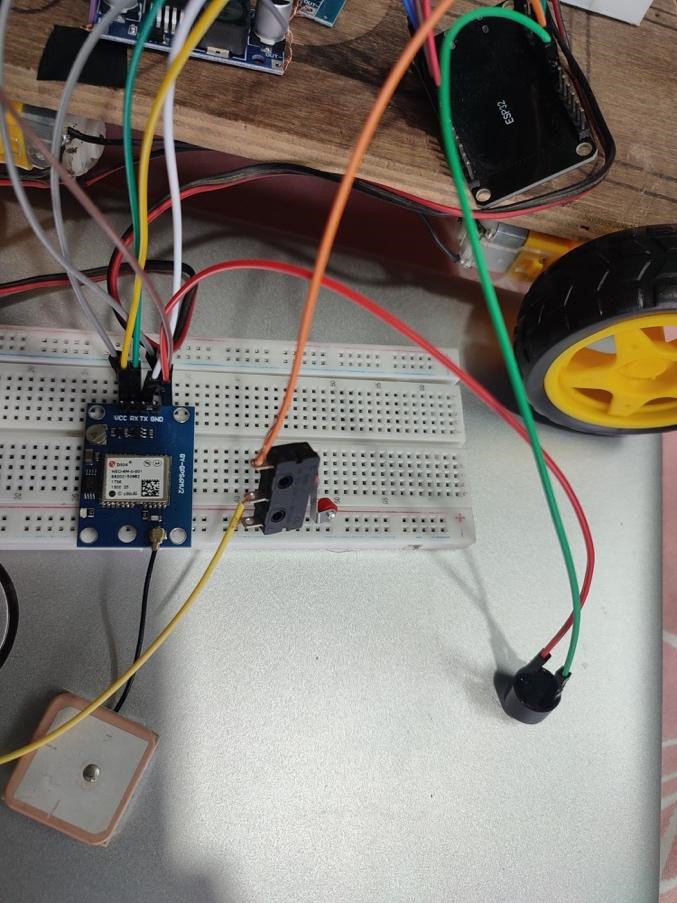


Figure 19: Final connection with impact switch and buzzer.

##### Phase 10: Uploading code to ESP32:

In this final process the code is being uploaded in ESP32 for controlling car and sending mail through internet.

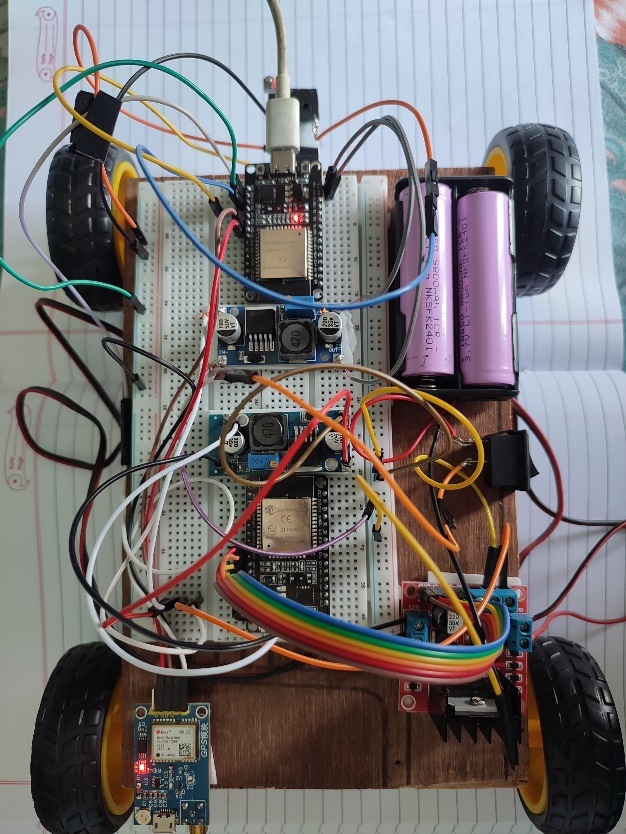


Figure 20: Code is being uploaded through USB cable

# 4. Results and Findings

### 4.1 Results

The project was successfully completed and resulted in the design and development of an IoTbased accident detection system that addresses the very crucial issue of quick emergency response in the event of an accident. The system is capable of automatically identifying accidents and notify emergency services with the help of communication and sensor technology.

Upon detection of the accident, the system immediately sent alert messages to the emergency contacts. This reduces the response time considerably, and in the event of accidents, it is crucial in ensuring life-saving. The findings indicate that this IoT accident detection system has the potential to contribute significantly in road safety and emergency response systems.

### 4.2 Findings

In this section, different test cases are mentioned to clarify successful result.

**Test 1:** Evaluation of GSM Module and Transition to ESP32 Wi-Fi Based Alert System.

|  |  |
| --- | --- |
| **Component Mentioned in Proposal** | Arduino Uno + GSM Module |
| Issue Faced During Testing | Arduino Uno with GSM module failed to send SMS alerts during impact detection testing due to GSM module's 2G limitation. |
| Modification Done | Replaced Arduino Uno and GSM module with ESP32 Wi-Fi-enabled microcontroller. Configured ESP32 to send GPS location via Gmail SMTP service. |
| Reason for Modification | GSM module supported only 2G networks, which have limited coverage. Using ESP32 allows Wi-Fi-based email alerting, ensuring reliable message delivery for the project. |

Table 1: GSM and Arduino replaced by ESP32

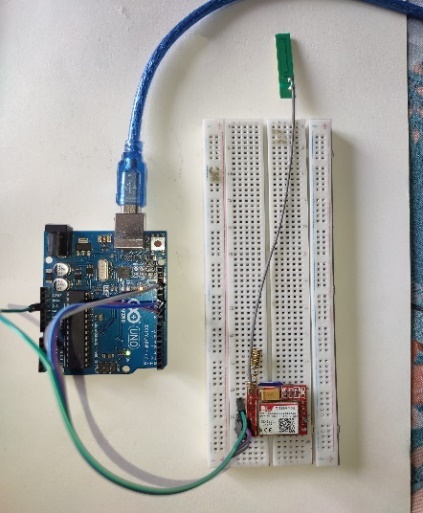


Figure 21: Arduino Uno with GSM module not functioning.

**Test 2:** ESP32 sending mail.

|  |  |
| --- | --- |
| Objective | To verify that beeps esp32 sends the alert message in mail through Gmail. |
| Action | The impact switch was triggered and buzzer beeps and esp32 sends the mail of recipient. |
| Expected Result | The impact switch gets triggered, and mail is sent. |
| Actual Result | As soon as the impact switch was pressed, the esp32 sends the mail to receiver within few seconds. |
| Conclusion | The test was successful. |

Table 2: ESP32 sending mails.

A screenshot of a phone

AI-generated content may be incorrect.

Figure 22: ESP32 sending mail to recipients.

**Test 3:** GPS is sending accurate location.

|  |  |
| --- | --- |
| Objective | To verify that the GPS module captures the location and sends the accurate location. |
| Action | A test impact was simulated by manually pressing the impact switch. |
| Expected Result | The GPS coordinates and sends the location to a predefined email address using SMTP with accurate coordinates. |
| Actual Result | The location was successfully received with accurate coordinates in gmail. |
| Conclusion | The test was successful. |

Table 3: Testing GPS is sending correct coordinates

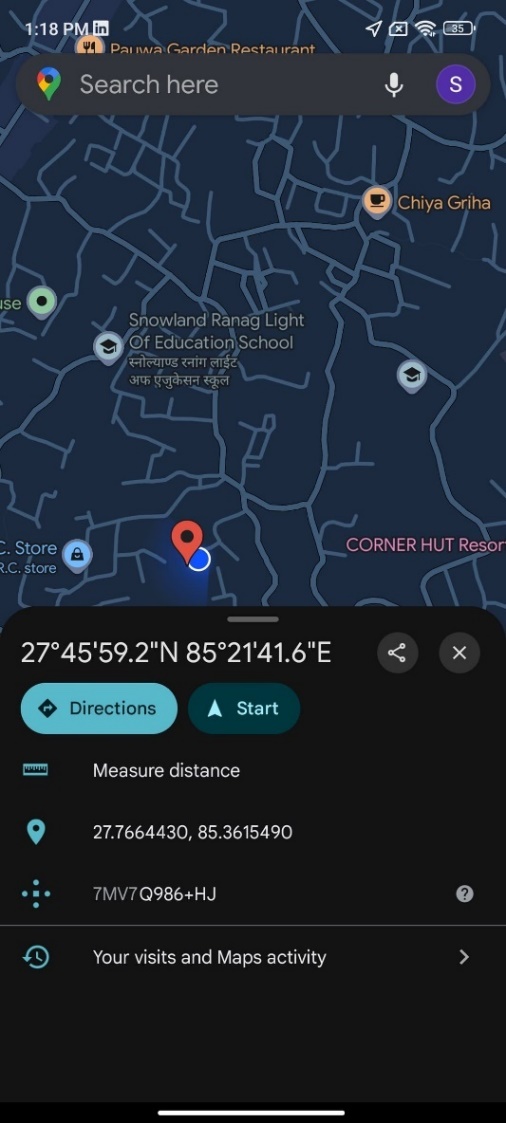


Figure 23: GPS location of actual and sent coordinates

**Test 4:** Visual and Audio Alert (LED & Buzzer Activation)

|  |  |
| --- | --- |
| Objective | To confirm that the LED light on ESP32 and buzzer activate to indicate a collision. |
| Action | A collision was simulated to trigger the impact switch. |
| Expected Result | Both LED and buzzer should activate for a few seconds. |
| Actual Result | LED turned ON in ESP32 and buzzer sounded for 5 seconds after impact detection. |
| Conclusion | The test was successful. |

Table 4: Testing LED and Buzzers working or not

A circuit board with wires

AI-generated content may be incorrect.

Figure 24: ESP32 blinking after impact getting triggered.

# 5. Future Work

The IoT-based Accident Detection System successfully demonstrates the ability to detect crashes and send emergency alerts, there are several ways this project can be improved and expanded in the future:

* **Integration with Emergency Services:** Future versions could directly notify nearby hospitals, police stations, or ambulance services through a centralized server or cloud platform, ensuring professional assistance without manual intervention.
* **Use of Advanced Sensors:** Incorporating a 3-axis accelerometer and gyroscope would allow for more accurate crash detection, distinguishing between minor bumps and serious accidents.
* **Mobile Application Interface:** A dedicated mobile app could be developed for vehicle owners or family members to receive accident alerts, view real-time GPS locations, and access the vehicle’s status remotely.
* **Cloud-Based Data Logging:** Storing incident data on the cloud can help in analyzing accident trends and improving road safety through big data analytics.
* **Multi-Vehicle Communication:** In the long term, enabling vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication can help in preventing accidents altogether by alerting nearby vehicles of sudden stops or hazards.

# 6. Conclusion

This IoT-based Accident Detection System is a helpful solution in getting medical help after road accidents. In countries like Nepal, many accidents are not reported on time, especially in remote areas or at night. This often happens because the victims are too hurt to call for help, and there may be no one around to report it. Our project solves this by using simple electronic parts like a ESP32 (a small computer), sensors, GPS module. These parts work together to detect a crash, find the location, and send a message to emergency contacts right away.

This means that even if no one is there to see the accident, the system can still send an alert for help. This can save valuable time and possibly lives, as getting medical help quickly makes a big difference. With this project, we have shown how technology can be used in real life to make roads safer and help people faster in emergencies. It also shows how IoT devices can be used not just in homes or industries, but also to solve serious problems in our daily lives. This system could help reduce deaths caused by road accidents.

# References

Ankita, w. S. S. S. S. A. U., 2023. *irjmets.* [Online] Available at: https://www.irjmets.com/uploadedfiles/paper//issue\_4\_april\_2023/37434/final/fin\_irjmets16830

24590.pdf#:~:text=According%20to%20the%20statistics%2C%20reducing,To [Accessed 29 April 2025].

geeksforgeeks., 2025. *geeksforgeeks..* [Online]

Available at: https://www.geeksforgeeks.org/introduction-to-microsoft-word/ [Accessed 30 April 2025]. hope, c., 2024. *Computer hope.* [Online]

Available at: https://www.computerhope.com/jargon/d/drawio.htm [Accessed 30 April 2025].

irjmets, 2023. *irjmets.* [Online] Available at: https://www.irjmets.com/uploadedfiles/paper//issue\_4\_april\_2023/37434/final/fin\_irjmets16830

24590.pdf#:~:text=According%20to%20the%20statistics%2C%20reducing,To [Accessed 30 April 2025].

Kathmandupost, 2025. *TheKathmandupost.* [Online]

Available at: https://kathmandupost.com/national/2025/01/26/nepal-sees-at-least-75-roadaccidents-on-average-daily-reportshows#:~:text=Nepal%20witnesses%20at%20least%2075,to%20a%20Nepal%20Police%20repo rt

[Accessed 30 April 2025].

Kekre, A., 2020. *medium.* [Online]

Available at: https://medium.com/@kekreaditya/how-does-a-gps-module-work-81b6892f300a [Accessed 1 May 2025].

Arduino Forum, 2020. *What are the critical considerations when selecting between buck regulators or voltage regulators?.* [Online]   
Available at: https://forum.arduino.cc/t/what-are-the-critical-considerations-when-selecting-between-buck-regulators-or-voltage-regulators/873492

Chettri, K. R., 2024. *The Annapurna Express.* [Online]   
Available at: https://theannapurnaexpress.com/story/50315/  
[Accessed 07 May 20205].

Circuits DIY, 2022. *YL-99 Impact Switch Crash Collision Sensor.* [Online]   
Available at: https://www.circuits-diy.com/crash-collision-sensor/  
[Accessed 12 May 2025].

Cirkit Designer, 2025. *Design, Simulate and Prototype Circuits With Ease.* [Online]   
Available at: https://www.cirkitstudio.com/  
[Accessed 12 May 2025].

Gupta, D. L., 2025. *Saving Lives in Seconds: Understanding the Vital Significance of Emergency Response Time.* [Online]   
Available at: https://www.medanta.org/patient-education-blog/saving-lives-in-seconds-understanding-the-vital-significance-of-emergency-response-time  
[Accessed 07 May 2025].

International Journal of Research Publication and Reviews, 2022. *Automatic Vehicle Accident Detection and Rescue System.* [Online]   
Available at: https://ijrpr.com/uploads/V4ISSUE4/IJRPR11838.pdf

Kathmandu Post, 2025. *kathmandu post national.* [Online]   
Available at: https://kathmandupost.com/national/2025/01/26/nepal-sees-at-least-75-road-accidents-on-average-daily-report-shows

Kumar, A., 2024. *IoT Based Automatic Vehicle Accident Detection And Rescue System.* [Online]   
Available at: https://www.hashstudioz.com/blog/iot-based-automatic-vehicle-accident-detection-and-rescue-system/  
[Accessed 07 May 2025].

Last Minute Engineers, 2022. *In-Depth: Interface L298N DC Motor Driver Module with Arduino.* [Online]   
Available at: https://lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial/

Nabto, 2024. *Nabto.* [Online]   
Available at: https://www.nabto.com/guide-to-iot-esp-32/  
[Accessed 11 May 2025].

Puthur, J. A., 2019. IoT based accident detection and prevention system with. *International Journal of Advance Research, Ideas and Innovations in Technology,* 5(2), p. 3.

Random Nerd Tutorials, 2022. *ESP32 Web Server - Arduino IDE.* [Online]   
Available at: https://randomnerdtutorials.com/esp32-web-server-arduino-ide/

Sund, B., 2018. *Effect of response times on survival from,* Örebro: Swedish Business School, Örebro University, Sweden .

Traca Logic, 2024. *The Need for IoT in Emergency Services.* [Online]   
Available at: https://www.tracalogic.com/emergency-services/  
[Accessed 07 May 2025].

U.S. Department of Energy, 2023. *How Lithium-ion Batteries Work.* [Online]   
Available at: https://www.energy.gov/energysaver/articles/how-lithium-ion-batteries-work

Ubidots, 2022. *Program the ESP8266 with the Arduino IDE in 3 simple steps.* [Online]   
Available at: https://help.ubidots.com/en/articles/928408-program-the-esp8266-with-the-arduino-ide-in-3-simple-steps

Wiltronics, 2022. *What Are Jumper Wires: Know by Colour, Types and Uses.* [Online]   
Available at: https://www.wiltronics.com.au/wiltronics-knowledge-base/what-are-jumper-wires/

# Appendix

**The source code of Car running through ESP32 is:**

#include <WiFi.h>

#include <WebServer.h>

// WiFi credentials

const char\* ssid = "ESP32\_Car";

const char\* password = "12345678";

// Motor Pins

#define IN1 27

#define IN2 14

#define ENA 12 // Enable Pin for Motor A

#define IN3 26

#define IN4 25

#define ENB 13 // Enable Pin for Motor B

WebServer server(80);

// ===== Movement Functions =====

void moveForward() {

digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);

digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);

digitalWrite(ENA, HIGH);

digitalWrite(ENB, HIGH);

}

void moveBackward() {

digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);

digitalWrite(ENA, HIGH);

digitalWrite(ENB, HIGH);

}

void moveLeft() {

digitalWrite(IN1, HIGH); digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW); digitalWrite(IN4, HIGH);

digitalWrite(ENA, HIGH);

digitalWrite(ENB, HIGH);

}

void moveRight() {

digitalWrite(IN1, LOW); digitalWrite(IN2, HIGH);

digitalWrite(IN3, HIGH); digitalWrite(IN4, LOW);

digitalWrite(ENA, HIGH);

digitalWrite(ENB, HIGH);

}

void stopCar() {

digitalWrite(IN1, LOW); digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW); digitalWrite(IN4, LOW);

digitalWrite(ENA, LOW);

digitalWrite(ENB, LOW);

}

// ===== HTML Page =====

const char\* htmlPage = R"rawliteral(

<!DOCTYPE html><html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>ESP32 Car Control</title>

<style>

body { text-align: center; font-family: Arial; margin-top: 20px; }

button {

padding: 12px 24px;

font-size: 16px;

margin: 5px;

border-radius: 5px;

border: none;

background: #4CAF50;

color: white;

cursor: pointer;

}

button:active { background: #45a049; }

.control-panel { max-width: 400px; margin: 0 auto; }

</style>

</head>

<body>

<div class="control-panel">

<h2>ESP32 Car Controller</h2>

<button onclick="controlCar('forward')"> Forward</button><br><br>

<button onclick="controlCar('left')">Left</button>

<button onclick="controlCar('stop')">Stop</button>

<button onclick="controlCar('right')">Right</button><br><br>

<button onclick="controlCar('backward')">Backward</button>

</div>

<script>

function controlCar(direction) {

fetch(`/${direction}`)

.catch(err => console.log('Error:', err));

}

</script>

</body>

</html>

)rawliteral";

// ===== Setup =====

void setup() {

Serial.begin(115200);

// Motor direction pins

pinMode(IN1, OUTPUT); pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT); pinMode(IN4, OUTPUT);

pinMode(ENA, OUTPUT); pinMode(ENB, OUTPUT);

// Start WiFi

WiFi.softAP(ssid, password);

Serial.print("AP IP address: ");

Serial.println(WiFi.softAPIP());

// Handle Web Requests

server.on("/", []() {

server.send(200, "text/html", htmlPage);

});

server.on("/forward", []() {

moveForward();

server.send(200, "text/html", htmlPage);

});

server.on("/backward", []() {

moveBackward();

server.send(200, "text/html", htmlPage);

});

server.on("/left", []() {

moveLeft();

server.send(200, "text/html", htmlPage);

});

server.on("/right", []() {

moveRight();

server.send(200, "text/html", htmlPage);

});

server.on("/stop", []() {

stopCar();

server.send(200, "text/html", htmlPage);

});

server.begin();

}

// ===== Loop =====

void loop() {

server.handleClient();

}

**The source code of sending mail through SMTP using Gmail is:**

#include <WiFi.h>

#include <ESP\_Mail\_Client.h>

#include <TinyGPSPlus.h>

#include <HardwareSerial.h>

// WiFi

const char\* ssid = "Wifi\_SSID";

const char\* password = "Wifi Password";

// SMTP

#define SMTP\_HOST "smtp.gmail.com"

#define SMTP\_PORT 465

#define AUTHOR\_EMAIL "s37129912@gmail.com"

#define AUTHOR\_PASSWORD "vede yidv yesj vrym"

#define RECIPIENT\_EMAIL "sanidhya9998@gmail.com"

// GPIO

#define IMPACT\_SWITCH\_PIN 15

#define BUZZER\_PIN 2

// Timing

unsigned long lastImpactTime = 0;

const unsigned long debounceDelay = 500;

const unsigned long cooldownPeriod = 10000;

SMTPSession smtp;

// GPS

TinyGPSPlus gps;

HardwareSerial gpsSerial(2); // Use UART2

#define GPS\_RX 16

#define GPS\_TX 17

void smtpCallback(SMTP\_Status status) {

Serial.println(status.info());

}

void setup() {

Serial.begin(115200);

gpsSerial.begin(9600, SERIAL\_8N1, GPS\_RX, GPS\_TX);

pinMode(IMPACT\_SWITCH\_PIN, INPUT\_PULLUP);

pinMode(BUZZER\_PIN, OUTPUT);

digitalWrite(BUZZER\_PIN, LOW);

WiFi.begin(ssid, password);

Serial.print("Connecting to WiFi");

while (WiFi.status() != WL\_CONNECTED) {

Serial.print(".");

delay(500);

}

Serial.println("\nWiFi connected!");

configTime(0, 0, "pool.ntp.org", "time.nist.gov");

Serial.print("Waiting for NTP time sync");

while (time(nullptr) < 100000) {

Serial.print(".");

delay(500);

}

Serial.println("\nTime synchronized");

smtp.callback(smtpCallback);

}

void loop() {

unsigned long currentTime = millis();

// Read GPS data

while (gpsSerial.available()) {

gps.encode(gpsSerial.read());

}

// If impact switch pressed and cooldown passed

if (digitalRead(IMPACT\_SWITCH\_PIN) == LOW && (currentTime - lastImpactTime > cooldownPeriod)) {

lastImpactTime = currentTime;

Serial.println("Impact Detected!");

// Beep buzzer

digitalWrite(BUZZER\_PIN, HIGH);

delay(1000);

digitalWrite(BUZZER\_PIN, LOW);

// Wait for valid GPS data

if (gps.location.isValid()) {

float latitude = gps.location.lat();

float longitude = gps.location.lng();

Serial.printf("GPS: %.6f, %.6f\n", latitude, longitude);

// Send email

ESP\_Mail\_Session session;

session.server.host\_name = SMTP\_HOST;

session.server.port = SMTP\_PORT;

session.login.email = AUTHOR\_EMAIL;

session.login.password = AUTHOR\_PASSWORD;

session.login.user\_domain = "";

SMTP\_Message message;

message.sender.name = "ESP32 GPS Alert";

message.sender.email = AUTHOR\_EMAIL;

message.subject = "🚨 Accident Detected!";

message.addRecipient("Recipient", RECIPIENT\_EMAIL);

String body = "An accident has been detected!\n\n";

body += "📍 Location:\n";

body += "Latitude: " + String(latitude, 6) + "\n";

body += "Longitude: " + String(longitude, 6) + "\n";

body += "\n📡 Google Maps: https://maps.google.com/?q=" + String(latitude, 6) + "," + String(longitude, 6);

message.text.content = body;

message.text.charSet = "utf-8";

message.text.transfer\_encoding = Content\_Transfer\_Encoding::enc\_7bit;

message.priority = esp\_mail\_smtp\_priority::esp\_mail\_smtp\_priority\_high;

if (!smtp.connect(&session)) {

Serial.println("SMTP connection failed");

Serial.println(smtp.errorReason());

return;

}

if (!MailClient.sendMail(&smtp, &message)) {

Serial.println("Error sending Email");

Serial.println(smtp.errorReason());

} else {

Serial.println("✅ Email with GPS sent successfully");

}

smtp.closeSession();

} else {

Serial.println("⚠️ Waiting for valid GPS signal...");

}

}

delay(debounceDelay);

}