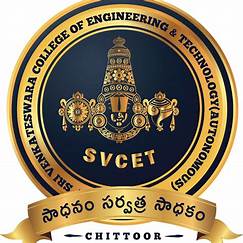
A black background with blue circle and text

AI-generated content may be incorrect.

**SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY**

**(AUTONOMOUS)**



[**Accident Detection System**](https://github.com/siri45704/Accident-Detection-System)

**23781A0447-D.Harish**

**23781A0448-D.Siri**

**23781A0449-D.VedaSri**

**23781A0450-D.Narasimha Reddy**

**23781A0451-D.Bhargava Ram**

**Problem Statement**

Road accidents claim countless lives annually, often due to delayed emergency response. An automated, real-time accident detection system can mitigate this by promptly alerting authorities and emergency contacts, potentially saving lives. The proposed system leverages the ESP8266 microcontroller, paired with a vibration sensor (e.g., SW-420 or MPU6050) to detect impacts indicative of accidents. Integrated with a GPS module (Neo-6M), it captures precise location data. Upon detecting an accident, the system triggers a buzzer for local alerts and sends notifications with GPS coordinates to an IoT platform (e.g., Blynk or ThingSpeak) via Wi-Fi. This ensures rapid communication with emergency services or predefined contacts. Designed for scalability, the system can be integrated into vehicles or two-wheelers, offering a cost-effective, reliable solution to enhance road safety by reducing response times and improving the chances of timely medical intervention.

**Scope of the Solution**

The Accident Detection System uses an ESP8266 to detect road accidents via a vibration sensor (SW-420 or MPU6050) and sends real-time alerts through Wi-Fi. A GPS module (Neo-6M) provides accurate location data, transmitted to emergency services or contacts via an IoT platform like Blynk or ThingSpeak. The system triggers local alerts with a buzzer and LEDs for status indication. It supports remote monitoring and is scalable for integration into vehicles or two-wheelers. The solution ensures rapid response, cost-effectiveness, and reliability, enhancing road safety by enabling timely emergency intervention.

**Required Components**

**Components of Hardware:**

* **ESP8266 (NodeMCU)**: Microcontroller with Wi-Fi for connectivity.
* **Vibration Sensor (SW-420 or MPU6050)**: Detects impact or shock.
* **GPS Module (Neo-6M)**: Provides location data.
* **Buzzer**: Local audible alert.
* **LEDs**: Status indicators (e.g., power, alert triggered).
* **Power Supply**: 3.3V/5V battery or USB.

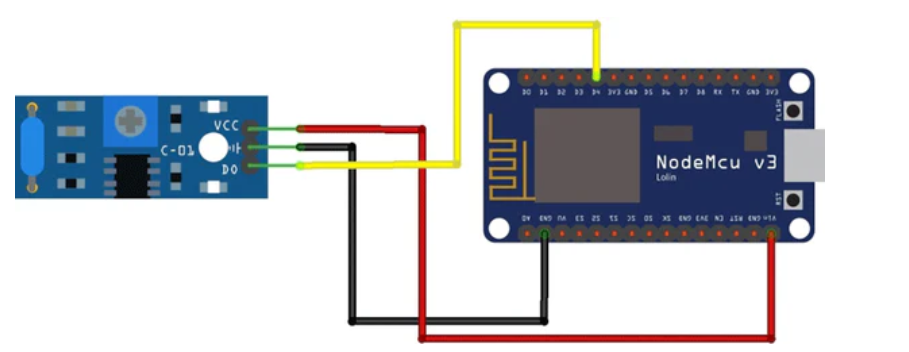
**Software & Tools:**

* **Arduino IDE**: For programming the ESP8266.
* **Fritzing/Tinkercad**: For circuit design and simulation.
* **GitHub**: For code hosting and documentation.
* **Blynk/ThingSpeak**: IoT platforms for remote monitoring  
    
  **Development Environment**
* IDE: Arduino IDE (v1.8.x or newer recommended)
* Board: ESP8266 NodeMCU
* Libraries Required:
* ESP8266WiFi.h
* BlynkSimpleEsp8266.h
* TinyGPS++.h
* SoftwareSerial.h

**Simulated Circuit**

**Circuit Design** :(**Fritzing)**

* **ESP8266 NodeMCU**: Central microcontroller.
* **SW-420 Vibration Sensor**: Connected to D2 (signal), 3.3V (VCC), GND.
* **Neo-6M GPS Module**: TX to D5, RX to D6, VCC to 3.3V, GND.
* **Buzzer**: Positive to D1, negative to GND.
* **LED**: Positive to D7 (with 220Ω resistor), negative to GND.
* **Power**: 3.3V from NodeMCU or external battery.



**Vibration Sensor**

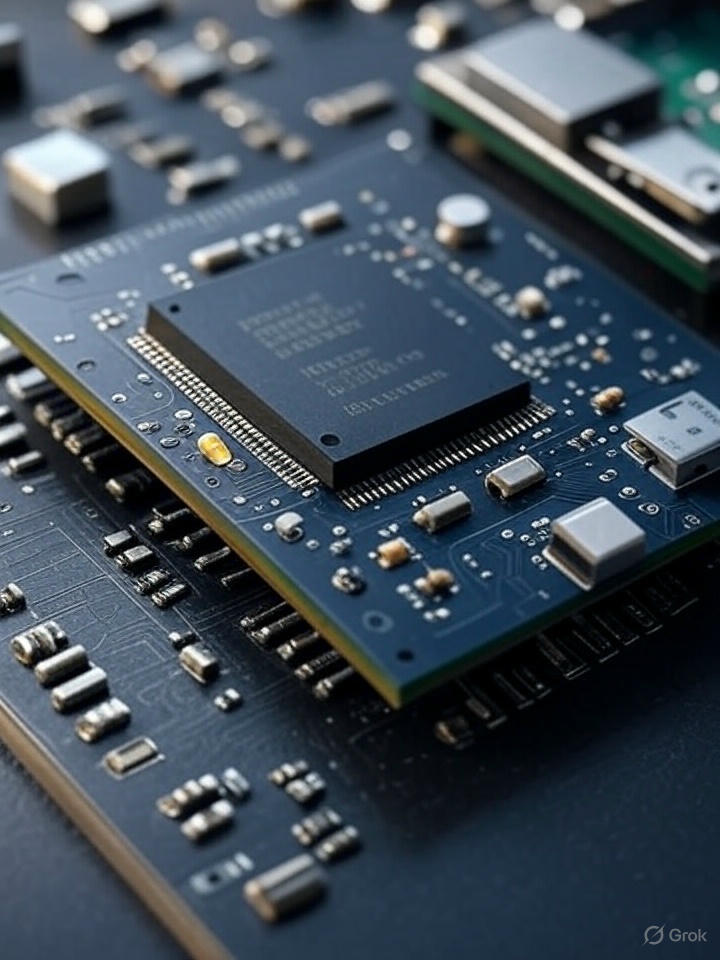
**A blue circuit board with red lights

AI-generated content may be incorrect.**

**Implementation using ESP8266**

The ESP8266 is a low-cost Wi-Fi microchip with full TCP/IP stack and microcontroller capabilities, widely used in IoT applications. Its implementation is ideal for projects requiring wireless connectivity, such as smart home automation, sensor networks, and remote data monitoring. Using the ESP8266, developers can collect data from various sensors and transmit it wirelessly to a server or cloud platform for real-time monitoring and analysis. It supports programming with the Arduino IDE, making it accessible for beginners and efficient for professionals. The chip includes GPIO pins for interfacing with external devices like LEDs, relays, and sensors. To implement a project using the ESP8266, the chip is first programmed via a USB-to-Serial converter or a NodeMCU development board. Once connected to a Wi-Fi network, it can send or receive data through HTTP, MQTT, or other protocols. Power efficiency and a compact design make it suitable for battery-powered applications. Overall, the ESP8266 enables cost-effective and scalable wireless solutions, transforming traditional electronic systems into intelligent, connected devices. Its widespread support, community resources, and ease of integration make it a cornerstone of modern IoT development.

**There are the Image of the ESP8266**

 A close-up of a computer chip

AI-generated content may be incorrect.

**Demo Video**

**Demo video is available in GitHub in https://github.com/siri45704/Accident-Detection-System**

**Gerber File**

**PCB Design**

* **Tool**: Use EasyEDA (easyeda.com) or Fritzing for PCB design.
* **Components**:
  + ESP8266 NodeMCU footprint.
  + SW-420 sensor (3-pin: VCC, GND, signal).
  + Neo-6M GPS module (4-pin: VCC, GND, TX, RX).
  + Buzzer (2-pin).
  + LED with 220Ω resistor.
  + Voltage regulator (e.g., AMS1117-3.3 for 3.3V stability).
* **Connections**: Map pins as in the circuit (D2 for SW-420, D5/D6 for GPS, D1 for buzzer, D7 for LED).
* **Power**: Include 3.3V regulator and connectors for battery/USB.

**Code:**

#include <ESP8266WiFi.h>

#include <SoftwareSerial.h>

#include <TinyGPS++.h>

#include <BlynkSimpleEsp8266.h>

char auth[] = "Your\_Blynk\_Token"; // Replace with Blynk auth token

char ssid[] = "Your\_WiFi\_SSID";   // Replace with Wi-Fi SSID

char pass[] = "Your\_WiFi\_Password"; // Replace with Wi-Fi password

SoftwareSerial gpsSerial(D5, D6); // GPS TX to D5, RX to D6

TinyGPSPlus gps;

int vibSensor = D2; // Vibration sensor pin

int buzzer = D1;    // Buzzer pin

int led = D7;       // LED pin

void setup() {

  Serial.begin(9600);

  gpsSerial.begin(9600);

  pinMode(vibSensor, INPUT);

  pinMode(buzzer, OUTPUT);

  pinMode(led, OUTPUT);

  Blynk.begin(auth, ssid, pass);

}

void loop() {

  Blynk.run();

  while (gpsSerial.available()) {

    gps.encode(gpsSerial.read());

  }

  int shock = digitalRead(vibSensor);

  if (shock == HIGH) {

    digitalWrite(buzzer, HIGH);

    digitalWrite(led, HIGH);

    if (gps.location.isValid()) {

      String msg = "Accident Detected! Location: ";

      msg += "Lat: " + String(gps.location.lat(), 6);

      msg += ", Lon: " + String(gps.location.lng(), 6);

      Blynk.logEvent("accident\_alert", msg);

      Serial.println(msg);

    } else {

      Blynk.logEvent("accident\_alert", "Accident Detected! GPS data unavailable.");

      Serial.println("GPS data unavailable.");

    }

    delay(10000); // Alert for 10 seconds

    digitalWrite(buzzer, LOW);

    digitalWrite(led, LOW);

  }

}

**Project Github links**

https://github.com/siri45704/Accident-Detection-System

