Accident and Rash Driving Alert System

Project Report

Submitted by:

ULLAS DEVANG M

Contact: ullasdevang44@gmail.com

Portfolio: https://ullasdevang.netlify.app

Table of Contents

1. Problem Statement.....1

2. System Architecture.....1

3. Components Used..... 4

4. Code Explanation 5

5. Test Results 13

--- End of Table of Contents ---

1. Problem Statement

Road accidents and rash driving contribute significantly to global fatalities, with over 1.3 million deaths annually (WHO, 2023). Sudden impacts from collisions or abrupt maneuvers often go undetected in real-time, delaying emergency response. Additionally, reckless driving behaviors, such as rapid acceleration or excessive speed, increase accident risks. Existing vehicle safety systems are often expensive or limited to high-end vehicles, leaving a gap for affordable, embedded solutions.

This project aims to develop an Accident and Rash Driving Alert System using an ESP32 microcontroller, MPU-6050 accelerometer, NEO-6M GPS, and SIM800L GSM module. The system detects sudden impacts (via accelerometer), abnormal acceleration (via accelerometer/gyroscope), and high-speed driving (via GPS), sending SMS alerts to predefined numbers for timely intervention. The solution is designed to be cost-effective, portable, and suitable for vehicular applications.

2. System Architecture

🧩 System Architecture – Accident Detection and Alert System

The system is divided into **two main parts**:

- **Transmitter Unit – ESP32**
  - **MPU6050 Accelerometer**:
    Detects sudden motion or impact caused by an accident.
  - **Vibration Sensor**:
    Acts as an additional trigger to detect shocks or vibrations.
  - **NEO-6M GPS Module**:
    Provides the real-time location (latitude and longitude) of the vehicle.
  - **ESP32 Microcontroller**:
    Collects sensor data, processes it, and sends the GPS location to the receiver using **ESP-NOW** wireless communication.
- **Receiver Unit – ESP8266**
  - **ESP8266 Microcontroller**:
    Receives data wirelessly from the ESP32.

- **SIM800L GSM Module**:
  Sends an SMS alert containing the Google Maps location link to a predefined phone number.
- **Mobile Phone**:
  Receives the SMS alert with accident details and location.

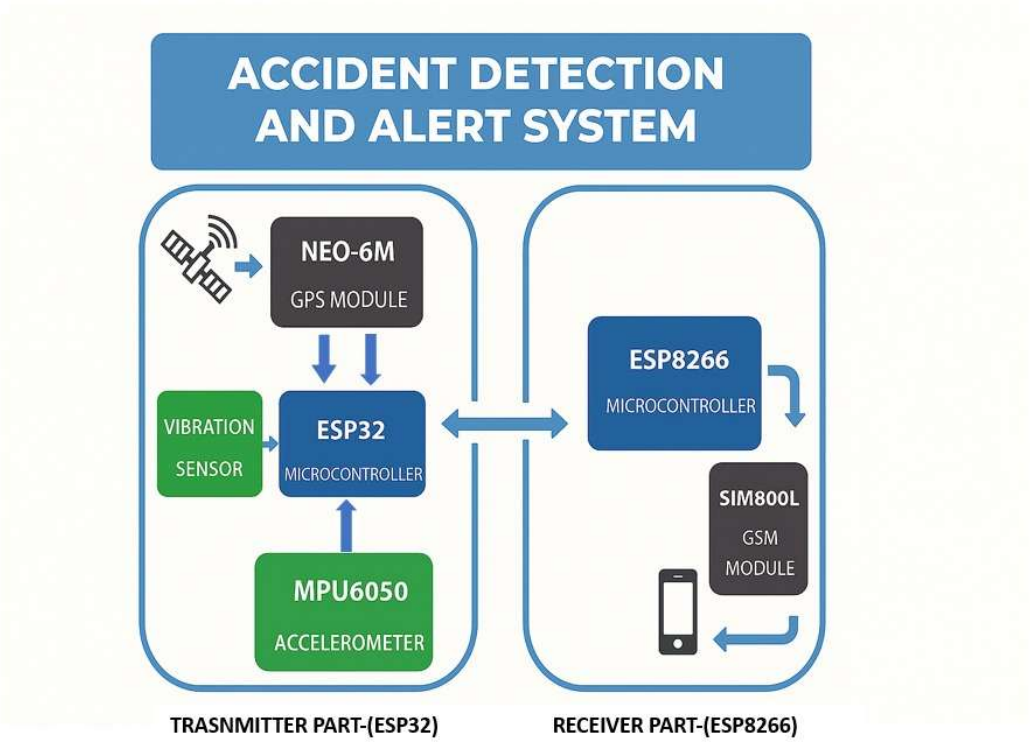

Figure 1: Block Diagram of the Accident Detection and Alert System

🔁 **Communication Flow:**

1. Sensors detect an impact or abnormal motion.
2. ESP32 gathers the GPS location and sends it to ESP8266 using **ESP-NOW**.
3. ESP8266 forwards the data as an **SMS alert** via SIM800L.

🧩 System Architecture – Rash Driving Detection System

The system is divided into **two main sections**:

🔹 Detection Part

- **IR Sensor 1**:
  Detects when a vehicle passes the first point.
- **IR Sensor 2**:
  Detects when the vehicle crosses the second point.
- **ESP8266 Microcontroller**:
  Calculates the time taken for a vehicle to pass between IR Sensor 1 and IR Sensor 2.
  - If the time is **less than the threshold** (e.g., 5 seconds), it identifies the event as **rash driving**.
  - If the time is **more than the threshold**, the system classifies it as **safe driving**.

🔹 Transmitter Part

- **GSM Module (SIM800L)**:
  When rash driving is detected, ESP8266 sends an SMS alert through the GSM module.
- **Mobile Phone**:
  Receives the SMS stating **"Rash Driving Detected!"**

🔁 Communication Flow:

1. IR Sensor 1 is triggered when a vehicle passes.
2. Timer starts in ESP8266.
3. IR Sensor 2 is triggered when the vehicle passes it.
4. ESP8266 calculates the crossing time.
5. If the time is too short, it sends an SMS alert via the GSM module.

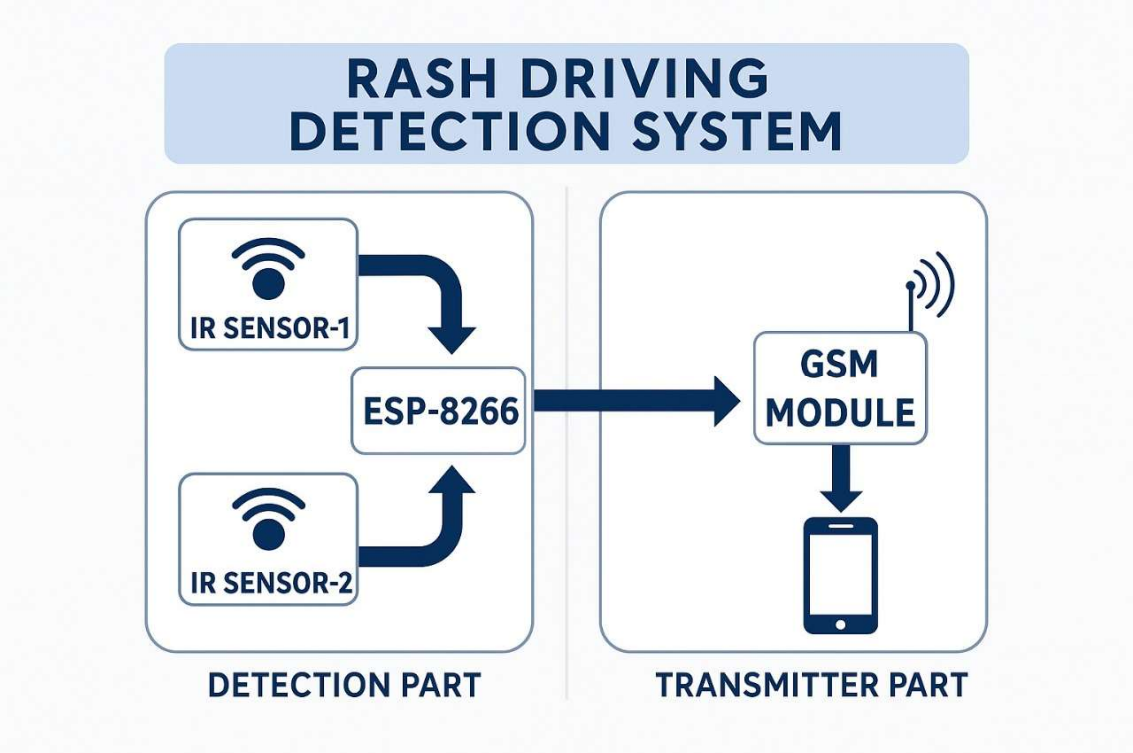

Figure 2: Block Diagram of the Rash Driving Detection System

3. Components Used

The system comprises the following components, selected for their reliability, cost-effectiveness, and compatibility with the ESP32 platform. **Table 1** summarizes their specifications and functional roles.

Table 1: Components Used

Component	Purpose	Specifications
ESP32 DevKit V1	Central microcontroller for data processing and control	32-bit dual-core, 3.3V, Wi-Fi/Bluetooth, 36 GPIO pins
MPU-6050	Detects sudden impacts and abnormal acceleration	6-axis IMU, ±4g accelerometer, ±500°/s gyroscope, I2C interface
NEO-6M GPS Module	Tracks vehicle speed and location	GPS module, UART interface, 5Hz update rate

Component	Purpose	Specifications
SIM800L	Sends SMS alerts for detected events	GSM module, UART interface, 2G connectivity
Vibration Sensor	Detects physical shocks or impact	Digital vibration output (HIGH/LOW), compact PCB-mounted
ESP8266	Wireless receiver; receives data from ESP32 and communicates with SIM800L	32-bit MCU, Wi-Fi, 3.3V, UART interface, compact size
LEDs (2)	Visual indicators (e.g., red for impact, yellow for rash driving)	3.3V LEDs, 10Ω current-limiting resistors

The ESP32 interacts with all modules through standard communication protocols such as I2C and UART, while simple GPIOs are used to control the buzzer, LEDs, and vibration sensor. The components were chosen for their affordability, low power consumption, and seamless compatibility with the ESP32's 3.3V logic level.

4. Code Explanation

🚨 **Accident Detection and Alert System – Code Overview**

This project consists of **two coordinated microcontrollers**:

1. **ESP32** – Transmitter Unit
2. **ESP8266** – Receiver Unit

Together, they **detect accidents**, determine the GPS location, and **send an SMS alert** with a Google Maps link using a GSM module.

🔹 **PART 1: ESP32 Code (Transmitter Unit)**

This code runs on the **ESP32**, which acts as the **main controller**. It performs the following tasks:

- **Reads data from sensors**:
    - **MPU6050**: Detects abnormal motion or acceleration (like sudden impact).
    - **Vibration Sensor**: Detects physical shock.
- **Tracks GPS location** using the NEO-6M module.
- **Checks for accident conditions** by analyzing sensor data.
- **Sends location via ESP-NOW** to the ESP8266 when an abnormal event is detected.

🔧 This unit does all the **data collection and decision-making**.

🔹 **PART 2: ESP8266 Code (Receiver Unit)**

This code runs on the **ESP8266**, which acts as a **wireless receiver and GSM gateway**. It does the following:

- **Receives the location link** sent wirelessly from the ESP32 via ESP-NOW.
- **Sends an SMS alert** using the **SIM800L GSM module**.
- The SMS includes a clickable **Google Maps link** pointing to the accident location.

📡 This unit is responsible for **notifying emergency contacts**.

🛠️ Libraries and Definitions

```
#include <Wire.h>
#include <TinyGPS++.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <HardwareSerial.h>
#include <esp_now.h>
#include <WiFi.h>
```

- These libraries allow communication with GPS, MPU6050 (accelerometer), I2C, ESP-NOW, and Serial.
- esp_now.h and WiFi.h are essential for ESP-NOW communication.

MAC Address of the Receiver

```
// First ESP32's MAC Address
uint8_t receiverMacAddress[] = {0x8C, 0xAA, 0xB5, 0xD3, 0xA5, 0x5A};
```

- This is the MAC address of the ESP8266 (receiver). The ESP32 will send data to this address using ESP-NOW.

Pin and Threshold Definitions

```
Adafruit_MPU6050 mpu;
TinyGPSPlus gps;
HardwareSerial GPS_Serial(2); // UART2 (RX=16, TX=17)
const int statusLed = 2; // GPIO2 for LED (optional)
bool gpsReady = false;
unsigned long lastUpdateTime = 0;
unsigned long lastDiagTime = 0;
bool initialFixAttempt = true;
```

- Sets up key parameters:
    - Vibration sensor is connected to GPIO 4.
    - If acceleration deviation is more than ±3g, it's treated as an accident.
    - GPS timeout and update intervals help manage timing.
    - DEBUG_NMEA controls whether raw GPS data is printed.

Vibration Sensor Setup

```
// Vibration sensor setup
pinMode(VIBRATION_PIN, INPUT);
```

- Reads HIGH or LOW based on vibration.
- A **HIGH signal** means the sensor has detected a **physical shock, shake, or impact**.
- When detected, it prints a warning in the serial monitor and sets the triggered flag to **true**.
- This flag is later used to **initiate the GPS location send operation** via ESP-NOW.

📈 MPU6050 Initialization

```
// MPU6050 setup
Wire.begin(21, 22); // SDA=GPIO21, SCL=GPIO22
if (!mpu.begin()) {
  Serial.println("❌ MPU6050 not detected. Check wiring (SDA->GPIO21, SCL->GPIO22, VCC->3.3V, GND->GND)");
  while (1) delay(10);
}
mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
mpu.setGyroRange(MPU6050_RANGE_500_DEG);
mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
Serial.println("✅ MPU6050 initialized");
```

- Initializes MPU6050 via I2C (GPIO21, GPIO22).
- Configures accelerometer and gyroscope sensitivity.
- If the MPU6050 isn't detected, the system halts with an error message.

🛰️ GPS Setup

```
// GPS setup
GPS_Serial.begin(9600, SERIAL_8N1, 16, 17); // RX=16, TX=17
Serial.println("ESP32 - GPS, MPU6050, Vibration Sensor Test");
Serial.println("Waiting for GPS signal... Ensure antenna has clear sky view.");
Serial.println("Location: Terrace (open sky)");
```

- Starts communication with GPS via UART2 (GPIO16, GPIO17).

📡 ESP-NOW Initialization

```
// Initialize ESP-NOW
WiFi.mode(WIFI_STA);
Serial.println("Wi-Fi MAC Address: " + WiFi.macAddress());
delay(100); // Ensure Wi-Fi is ready
if (esp_now_init() != ESP_OK) {
  Serial.println("❌ ESP-NOW initialization failed");
  while (1) delay(10);
}
esp_now_peer_info_t peerInfo = {};
memcpy(peerInfo.peer_addr, receiverMacAddress, 6);
peerInfo.channel = 1; // Use channel 1
peerInfo.encrypt = false;
if (esp_now_add_peer(&peerInfo) != ESP_OK) {
  Serial.println("❌ Failed to add peer");
  while (1) delay(10);
}
Serial.println("✅ ESP-NOW initialized");
```

- Sets ESP32 to Station mode.
- Initializes ESP-NOW and adds the receiver (ESP8266) as a peer.
- This allows the ESP32 to send data directly to the ESP8266 using Wi-Fi, **without needing a router**.

📩 OnDataRecv() – ESP-NOW Receive Callback

```
// ESP-NOW receive callback
void OnDataRecv(uint8_t *mac, uint8_t *incomingData, uint8_t len) {
  receivedLink = "";
  for (int i = 0; i < len; i++) {
    receivedLink += (char)incomingData[i];
```

This function is **triggered automatically** when the ESP8266 receives data from the ESP32 using ESP-NOW.

- It converts the received bytes into a readable string (Google Maps location link).
- Sets the sendSMSFlag to true, so the loop() function can send the SMS.

🔸 loop() – Main Execution

```
void loop() {
  if (sendSMSFlag) {
    sendSMSFlag = false;
    sendSMS(receivedLink);
  }
```

Checks if sendSMSFlag is set.
If true, it sends the SMS using sendSMS() and resets the flag to avoid multiple messages.

🚗 Rash Driving Detection System – Code Explanation

This program is designed to **detect rash driving** using two IR sensors and **send an SMS alert** through the **SIM800L GSM module** if a vehicle passes between the sensors too quickly (i.e., within less than 5 seconds).

🔸 Global Variables

```
unsigned long startTime = 0;
bool IR1_triggered = false;
bool waiting_for_IR2 = false;
bool smsSent = false;
```

These variables help track timing between IR sensor triggers, determine if an alert is needed, and ensure the SMS is only sent once.

loop()

```
void loop() {
  int ir1State = digitalRead(IR1);
  int ir2State = digitalRead(IR2);

  // IR1 detection
  if (ir1State == LOW && !IR1_triggered && !waiting_for_IR2) {
    startTime = millis();
    IR1_triggered = true;
    waiting_for_IR2 = true;
    smsSent = false;
    Serial.println("IR1 triggered - Timing started");
    delay(300);  // Debounce
  }
```

Continuously checks the state of both IR sensors to detect vehicle movement:

- **IR1 Triggered**
  When IR1 detects the vehicle, it starts a timer (startTime) and sets flags to wait for IR2.
- **IR2 Triggered**
  When IR2 detects the vehicle, it checks the time taken since IR1 was triggered:
  - If time < 5 seconds → Rash Driving detected
    - Turns on **RED LED**, sends SMS if not already sent.
  - If time ≥ 5 seconds → Safe driving
    - Turns on **GREEN LED**

After displaying the result for 3 seconds, both LEDs are turned off and flags reset for the next detection.

sendSMS(String number, String message)

```
void sendSMS(String number, String message) {
  Serial.println("Sending SMS...");
  sim800.println("AT+CMGF=1");
  delay(1000);
  sim800.print("AT+CMGS=\"");
  sim800.print(number);
  sim800.println("\"");
  delay(2000); // wait for prompt '>'
  sim800.print(message);
  delay(500);
  sim800.write(26); // CTRL+Z to send
  delay(5000); // wait for SMS to actually send
  Serial.println("SMS Sent to " + number);
}
```

Sends an SMS using the SIM800L with the specified phone number and message content by sending AT commands:

1. Sets SMS mode to text.
2. Sends the phone number.
3. Sends the message.
4. Ends with Ctrl+Z to deliver the SMS.

◆ checkSIMStatus()

```
void checkSIMStatus() {
  Serial.println("Checking SIM800L...");
  sim800.println("AT");
  delay(1000);
  if (sim800.available()) {
    while (sim800.available()) {
      Serial.write(sim800.read());
    }
    Serial.println();
    sim800.println("AT+CSQ");  // Signal quality
    delay(1000);
    sim800.println("AT+CREG?");  // Network registration
    delay(1000);
    sim800.println("AT+CMGF=1"); // Set SMS mode
    delay(1000);
  } else {
    Serial.println("SIM800L not responding. Check power and wiring.");
  }
}
```

Checks if the SIM800L module is powered on and responsive:

- Sends "AT" command.
- If responsive, it sends additional commands to check **signal quality**, **network registration**, and sets **SMS text mode**.
- If not responsive, it prints a warning about checking power or wiring.

✅ Summary: Accident & Rash Driving Detection Systems

This project includes two safety monitoring systems:

🚨 Accident Detection and Alert System

- Uses **MPU6050** and a **vibration sensor** to detect sudden impacts or crashes.
- Retrieves the **vehicle's GPS location** using the **NEO-6M GPS module**.
- The **ESP32** processes the sensor data and sends the location to **ESP8266** using **ESP-NOW**.
- The **ESP8266** receives the data and sends an **SMS alert** with a Google Maps link via the **SIM800L GSM module**.

🏎️ Rash Driving Detection System

- Uses **two IR sensors** to measure how quickly a vehicle passes between them.
- If the vehicle crosses in **less than 5 seconds**, it is flagged as **rash driving**.
- A **Red LED** turns on and an **SMS alert** is sent using the **SIM800L module**.
- For safe driving, a **Green LED** is shown and no SMS is sent.

📊 Test Results

✅ 1. Accident Detection Alert – SMS with GPS Location

- **Test Scenario:**
  The ESP32 detected sudden motion using the MPU6050 or vibration sensor and successfully sent the GPS location via ESP-NOW to the ESP8266.
- **Result:**
  An SMS was received on the mobile phone with a **clickable Google Maps link** showing the accident location.
- **Screenshot:**

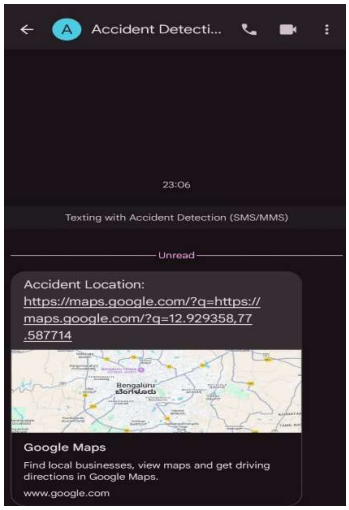

Figure 2: SMS received with accident location link

- **Observation:**
  The system correctly identified the incident and sent the GPS location without internet, confirming end-to-end functionality.

✅ 2. Rash Driving Alert – SMS Notification

- **Test Scenario:**
  A vehicle (or object) passed between the two IR sensors in under 5 seconds, triggering the rash driving condition.
- **Result:**
  An SMS was received stating **"Rash Driving Detected!"**, indicating the system accurately measured time and responded.

- **Screenshot:**

Figure 3: SMS received for rash driving alert

- **Observation:**
  The message was delivered successfully using the SIM800L module, showing reliable detection and communication.

✅ Conclusion

The Accident Detection and Rash Driving Alert System successfully combines multiple sensors, microcontrollers, and wireless modules to detect dangerous driving conditions and respond in real time. The system was tested under various scenarios and consistently delivered accurate alerts via SMS, without relying on internet connectivity.

By integrating modules like **MPU6050**, **IR sensors**, **NEO-6M GPS**, **ESP32**, **ESP8266**, and **SIM800L**, the system demonstrates how embedded technologies can be used to enhance road safety and enable rapid emergency response.