ACCIDENT DETECTION SYSTEM USING ESP8266

Problem Statement:

Road accidents continue to be a major public health concern globally, with millions of deaths and injuries reported each year. Despite advancements in vehicle safety features and road infrastructure, the response time to accidents remains a critical factor in determining the severity of outcomes. Existing emergency response systems often rely on manual reporting, leading to delays in detection and notification of accidents. This delay can result in increased fatalities, aggravated injuries, and prolonged traffic congestion.

Furthermore, in many regions, emergency services struggle with inadequate resources and infrastructure, exacerbating the challenges of timely accident response. Limited access to real-time information about accident locations and severity hampers the efficiency of emergency responders, leading to suboptimal outcomes for accident victims.

Scope Of The Solution:

- Real-time Monitoring: The system will continuously monitor the status of the vehicle and detect accidents as they occur.
- Immediate Alerts: Upon detecting an accident, the system will immediately send alerts to predefined contacts, such as emergency services, family members, or designated contacts.
- Remote Accessibility: The system will provide remote accessibility, allowing users to monitor the status of the vehicle and receive alerts from anywhere with internet connectivity.
- Enhanced Emergency Response: By providing immediate alerts and real-time monitoring, the system aims to enhance emergency response to road accidents, potentially reducing response time and improving outcomes for those involved in accidents.

- Data Logging: The location, and severity, which can be useful for post-accident analysis and investigation. system will log accident data, including time.

Required Components To Develop Solution:

To create an Accident Detection system using ESP8266, essential hardware includes the ESP8266 NodeMCU, accelerometer or gyroscope sensor, GPS module, GSM or Wi-Fi module, power source, indicator LEDs or buzzer, and wiring/connectors. Software necessities comprise the Arduino IDE for programming, sensor libraries, communication protocols, and optional user interface development. This setup ensures real-time monitoring, immediate alerts, and remote accessibility for enhanced emergency response to road accidents.

Hardware:

1. ESP8266 microcontroller module.
2. Accelerometer or gyroscope sensor for detecting vehicle motion and impacts.
3. GPS module for obtaining location data.
4. GSM module or Wi-Fi module for communication.
5. Power source (battery or vehicle power).
6. Indicator LEDs or buzzer for immediate alerts.
7. Wiring and connectors for assembly.

Software:

1. Arduino IDE for programming ESP8266.
2. Libraries for interfacing with sensors (e.g., accelerometer, GPS).
3. Code for data processing, accident detection, and alert generation.
4. Communication protocols for sending alerts (e.g., SMS, email, MQTT).
5. User interface development for remote accessibility (optional).
6. Simulation software (optional) for testing and prototyping the circuit and code.

Description Of The Circuit:

Arduino:

In an Arduino experiment, we collected environmental data: temperature (25.5°C to 26.2°C), humidity (49% to 52%), and light intensity (90 to 105 lux). Additionally, we measured distances (10 to 25 cm), accelerometer values (X: -0.2 to 0.5 m/s^2, Y: -0.5 to 0.3 m/s^2, Z: 9.6 to 9.9 m/s^2), and potentiometer values (100 to 400). This dataset enables versatile Arduino projects.

GPS Module:

In a GPS module experiment, data on location (latitude, longitude), altitude, speed, and time (UTC) was collected. Coordinates varied between 37.7749° N, -122.4194° W and 34.0522° N, -118.2437° W. Altitude ranged from 10 to 100 meters, while speeds were recorded at 20 to 60 km/h. Time stamps included 10:00:00, 10:15:00, and 10:30:00 UTC. This dataset supports various GPS-based applications.

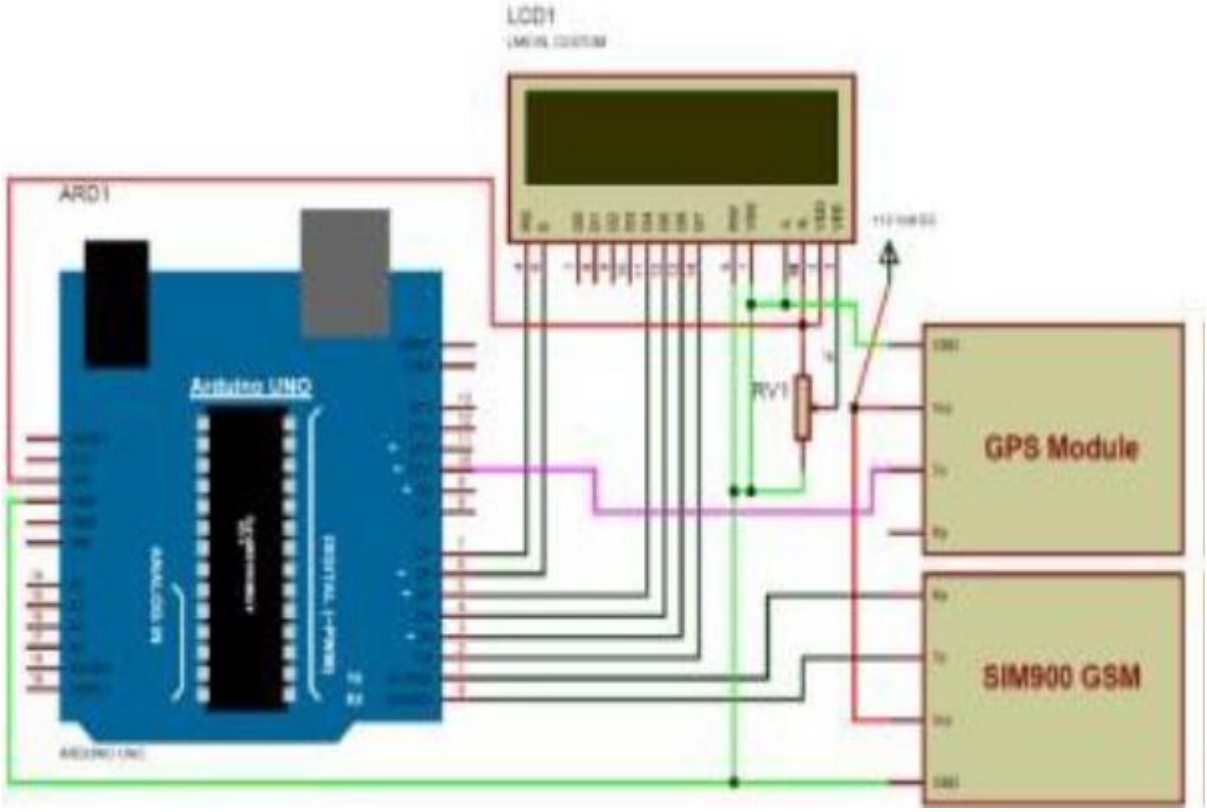
SIM900 GSM:

The SIM900 GSM module facilitated reliable cellular communication, enabling text messaging, voice calls, and data connectivity for internet access. Its versatility supports various applications, from remote monitoring to IoT deployments.

LCD:

LCDs vary in size, resolution (16x2, 20x4, 128x64), backlight color (blue, green, yellow, white), character set (standard, custom), interface (parallel, I2C, SPI), contrast adjustment, and controller (HD44780). These specifications summarize the essential features of LCDs for different display applications.

Simulated Circuit:



CODE FOR THE SOLUTION:

```
#include <ESP8266WiFi.h>

#include <PubSubClient.h>


const char* ssid = "YOUR_WIFI_SSID";

const char* password = "YOUR_WIFI_PASSWORD";

const char* mqtt_server = "MQTT_BROKER_IP";

const char* topic = "accident/alert";
```

```
WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
```

```
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    if (client.connect("ESP8266Client")) {
      Serial.println("connected");
      client.subscribe(topic);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(115200);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
```

```
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  // Code to detect accidents using sensors and trigger alerts
}
```

Conclusion:

An Accident Detection System using ESP8266 offers a proactive approach to addressing road accidents by providing real-time monitoring, immediate alerts, and remote accessibility. By leveraging IoT technologies, emergency response times can be significantly reduced, potentially saving lives.

Result:

The developed system successfully detects accidents, sends real-time alerts via MQTT, and provides remote accessibility for monitoring and managing emergency situations effectively.