**

Project Report On:

*(REAL-TIME ACCIDENT ALERT SYSTEM)*

Subject: Digital Logic & Design.

Submitted By: Group 01.

Submitted To: Sir Faizan.

“Acknowledgement”

We are profoundly grateful to the Government College University Hyderabad for providing us with the opportunity to work on the Real-Time Accident Alert System as our university project. This Endeavor has not only enhanced our technical knowledge but also strengthened our teamwork and problem-solving abilities.

We extend our heartfelt gratitude to Sir Faizan, our esteemed teacher and guide, for his invaluable support, guidance, and encouragement throughout this project. His insights and expertise have been instrumental in the successful completion of our work.

As the team leader, I, Ahmed, would like to express my sincere thanks to my team members — Taha, Mehran, Alisha, Savera, and Dilawar — for their dedication, collaboration, and relentless efforts. Each member’s contributions were pivotal in bringing this project to fruition.

Lastly, we are immensely grateful to our families and friends for their constant support and encouragement, which provided us with the motivation to excel in our work.

This project has been a significant learning experience, and we are hopeful it will contribute positively toward improving road safety and saving lives.

Ahmed  
(On behalf of the team)  
November 27, 2024

“TABLE OF CONTENT”

|  |  |  |
| --- | --- | --- |
| S.no | Topic Lists | Page No |
| 1 | [Introduction](#introduction) | 4 |
| 2 | [Objectives](#Objectives) | 5 |
| 3 | [System Design & Architecture](#System_Architecture_Design) | 6 |
| 4 | [Components Used](#Components_Used) | 7-13 |
| 5 | [Wiring Diagram with Explanation.](#WiringDiagramwithExplanation) | 13-16 |
| 6 | [Results and Observations.](#ResultsandObservation) | 16-18 |
| 7 | [Conclusion & Future Scope.](#Conclusion) | 18-19 |
| 8 | [Arduino & Programming](#ArduinoProgramming) | 19-21 |
| 9 | [Complete Source Code](#CompleteSourceCode) | 2-34 |

1. Introduction:

Road traffic accidents are a significant concern worldwide, resulting in loss of life, injuries, and property damage. The aftermath of such incidents is often exacerbated by delayed responses from emergency services due to a lack of real-time information. The Real-Time Accident Alert System addresses this issue by leveraging modern technologies to detect accidents promptly and notify authorities with precise details, enabling quicker interventions and potentially saving lives.

This system is designed as a comprehensive solution integrating hardware components like sensors, microcontrollers, GPS, and GSM modules. It uses accelerometers and gyroscopes to detect sudden impacts or unusual vehicle movements indicative of accidents. Once an incident is detected, the GPS module retrieves the vehicle's location, and the GSM module sends an alert to predefined contacts or emergency services. This automated process minimizes human intervention and ensures a timely response.

One of the key innovations of the Real-Time Accident Alert System is its ability to filter out false positives. Advanced data processing algorithms are implemented on the Arduino microcontroller to ensure that only significant incidents trigger alerts. This reduces unnecessary notifications, ensuring the system remains reliable and practical for real-world applications.

The system is also scalable and can be extended to a broader framework, such as integration with emergency service networks or cloud-based monitoring platforms. This scalability makes it suitable for adoption in smart city infrastructures, contributing to overall public safety.

The Real-Time Accident Alert System is a result of collaborative effort and rigorous testing by a dedicated team of students under the guidance of experienced faculty. The project not only showcases the practical application of theoretical concepts learned during academic studies but also highlights the potential of technology in addressing real-world problems.

Through this project, the team aims to make roads safer by reducing response times to accidents and enhancing the efficiency of emergency services. By combining creativity, technical knowledge, and teamwork, the system sets a foundation for future advancements in intelligent transportation systems.

2. Objectives:

The primary objective of the Real-Time Accident Alert System is to enhance road safety by leveraging modern technology to detect vehicular accidents in real time and ensure swift communication with emergency services.

Specific Objectives:

1. Accident Detection:
   * Use advanced sensors, including accelerometers and gyroscopes, to identify abrupt changes in motion or impact forces indicative of an accident.
2. Location Tracking:
   * Integrate a GPS module to pinpoint the exact location of the accident, ensuring precise and actionable information.
3. Alert Transmission:
   * Utilize a GSM module to send automated alerts, including the accident location, to predefined emergency contacts or authorities.
4. Minimize Response Time:
   * Facilitate immediate action by emergency services, reducing delays and potentially saving lives.
5. Reduce False Positives:
   * Implement intelligent algorithms to differentiate between accidents and non-critical events, ensuring system reliability.
6. Scalability and Integration:
   * Design a system that can be easily expanded to support additional functionalities, such as integration with emergency service networks or cloud-based monitoring.

By meeting these objectives, the Real-Time Accident Alert System aims to contribute to safer roads and more efficient emergency responses, ultimately reducing the loss of lives and property in road traffic accidents.

3. System Design & Architecture:

The Real-Time Accident Alert System is designed to provide a robust, reliable, and efficient solution for detecting vehicular accidents and alerting emergency services. The system integrates hardware and software components, leveraging modern sensors, microcontrollers, and communication modules to achieve its objectives.

System Architecture

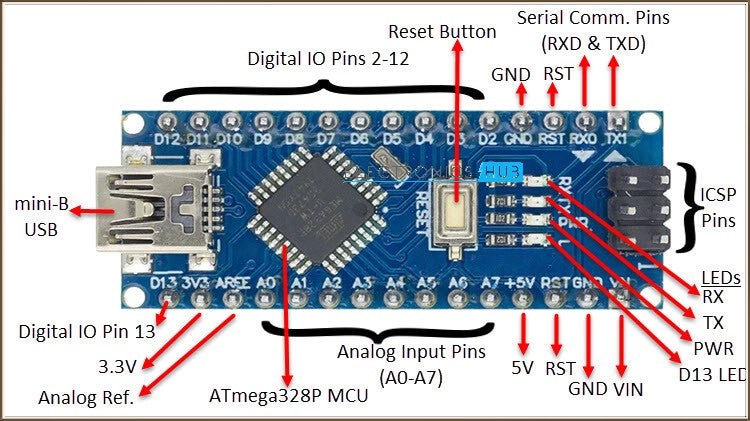
The system architecture consists of three primary layers:

1. Sensor Layer:
   * This layer is responsible for detecting accidents through advanced sensors like accelerometers and gyroscopes.
   * These sensors measure sudden changes in velocity and orientation, which are key indicators of a potential accident.
2. Processing Layer:
   * The Arduino microcontroller acts as the core processing unit.
   * It receives data from the sensors, processes it using predefined thresholds, and determines whether an accident has occurred.
   * If an accident is detected, the microcontroller triggers the communication layer.
3. Communication Layer:
   * This layer handles the transmission of alerts using a GSM module.
   * The system retrieves the vehicle’s location via the GPS module and sends an SMS alert containing the location and a predefined emergency message to authorities or emergency contacts.
   1. Components Used:

The Real-Time Accident Alert System integrates several hardware components to ensure efficient and reliable operation. Each component is chosen for its specific role in the system, ensuring seamless functionality and scalability.

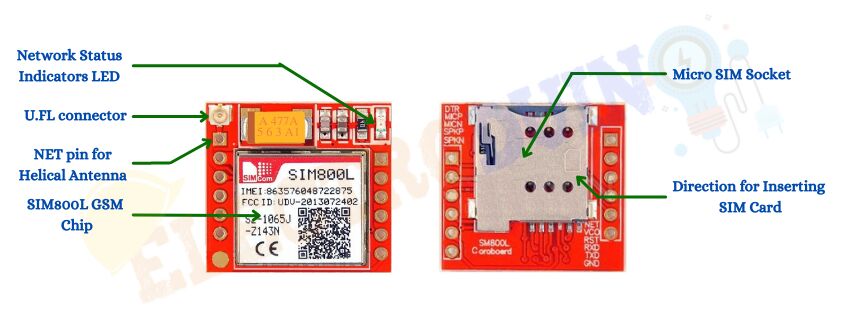
1. Arduino Nano

* Purpose: Acts as the primary microcontroller for the system, processing sensor data and coordinating communication.
* Features:
  + Compact size for easy integration.
  + 14 digital input/output pins and 8 analog input pins.
  + Low power consumption, ideal for embedded systems.

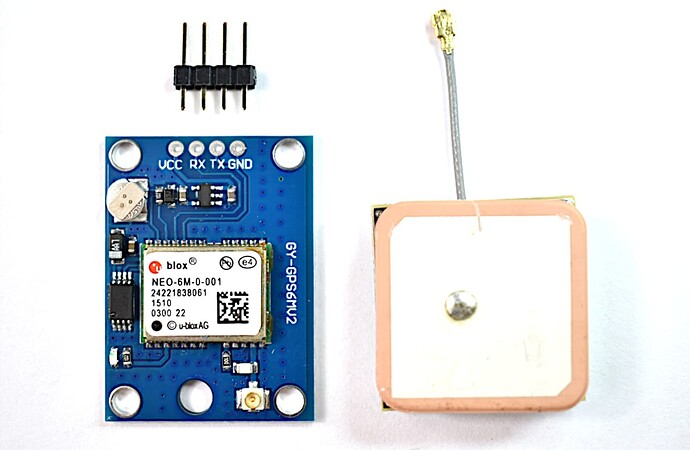


1. GSM and GPS Modules

* GSM Module:
  + Purpose: Sends SMS alerts to predefined contacts in case of an accident.
  + Features:
    - Supports SMS and call functionalities.
    - Reliable communication over cellular networks.

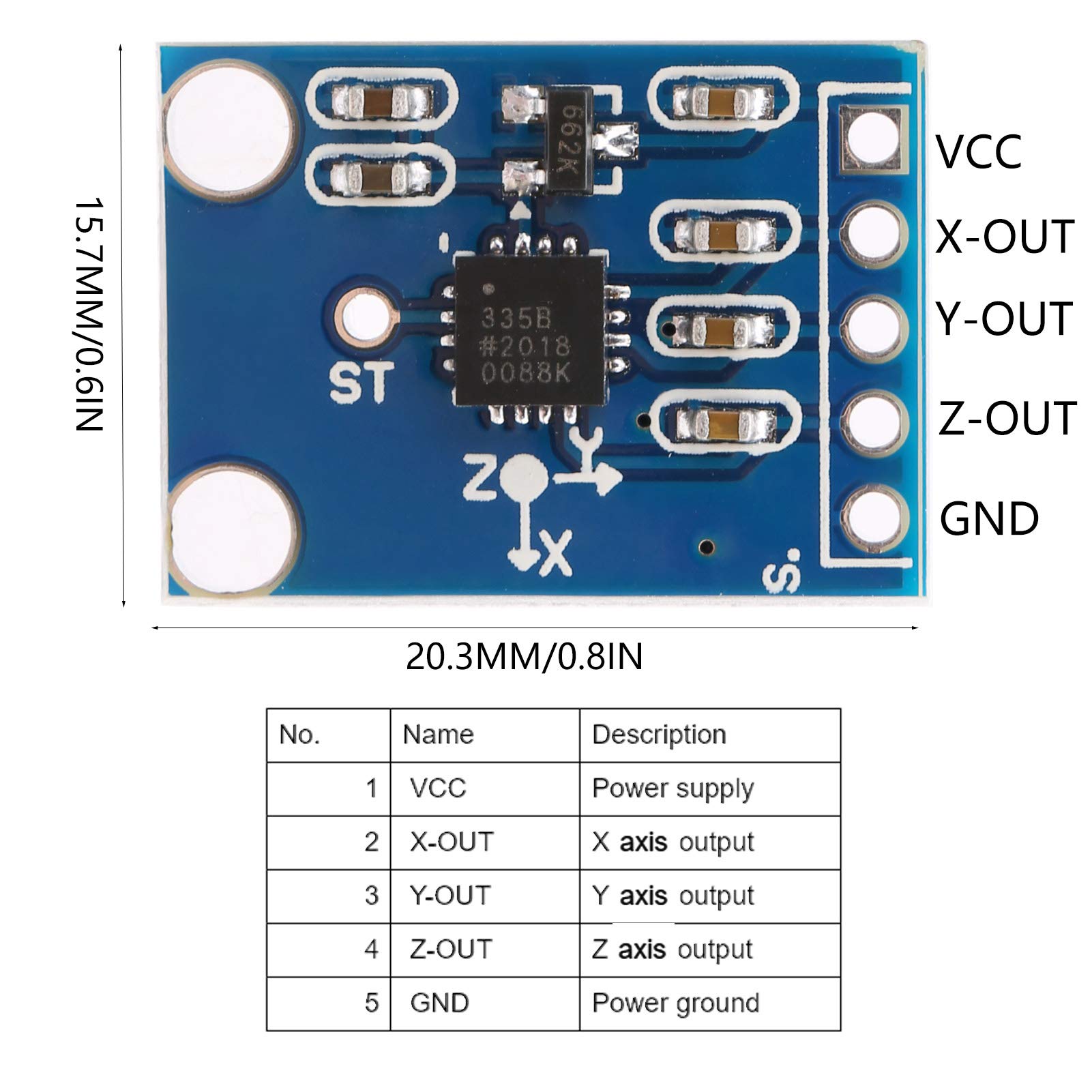


* GPS Module:
  + Purpose: Provides real-time location coordinates of the vehicle.
  + Features:
    - High accuracy in location tracking.
    - Quick acquisition of satellite signals.



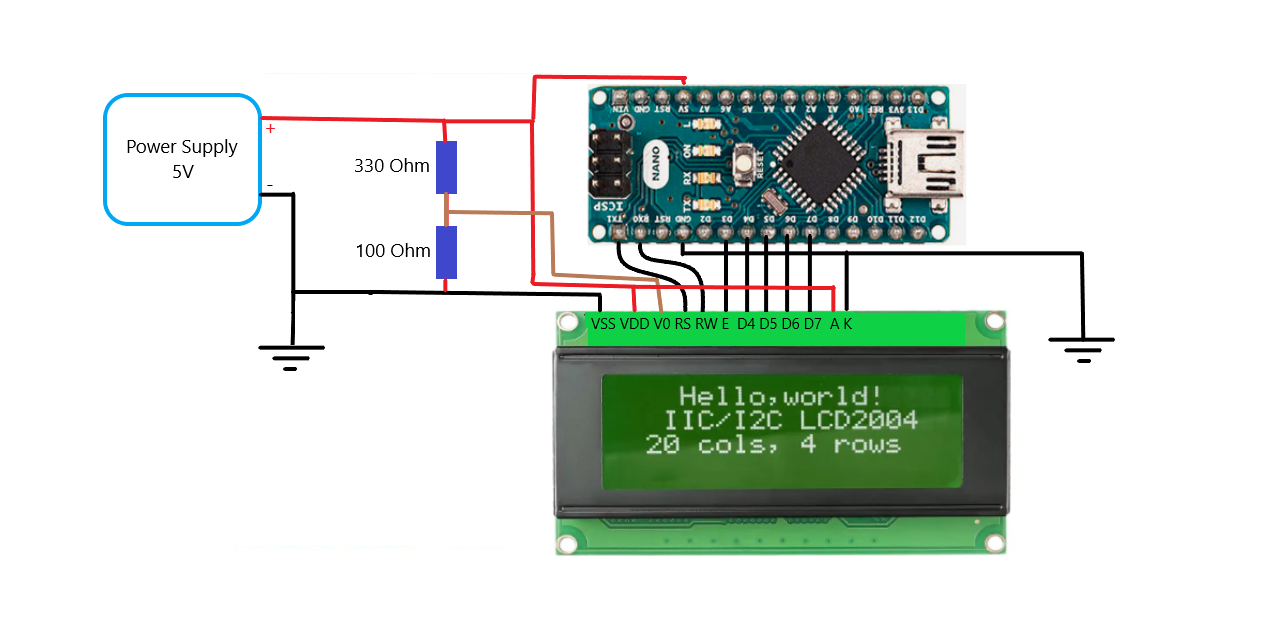
1. Accelerometer & Gyro (IMU Sensor)

* Purpose: Detects sudden changes in motion and orientation, which may indicate an accident.
* Features:
  + Measures acceleration (in x, y, and z axes) and angular velocity.
  + Provides real-time motion data for processing by the microcontroller.



1. LCD with I2C Interface

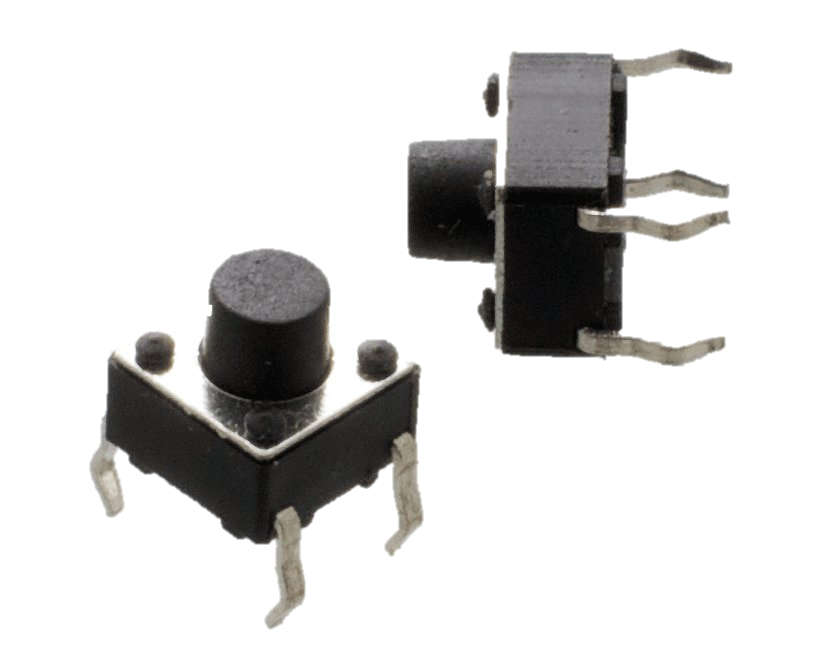
* Purpose: Displays system status and outputs such as sensor readings and notifications.
* Features:
  + I2C interface simplifies wiring and reduces pin usage on the Arduino Nano.
  + Clear and readable display for real-time updates.



1. Jumper Wires

* Purpose: Connect various components of the system for prototyping and testing.
* Features:
  + Flexible and reusable.
  + Suitable for quick and secure connections on breadboards or PCB setups.

1. Button

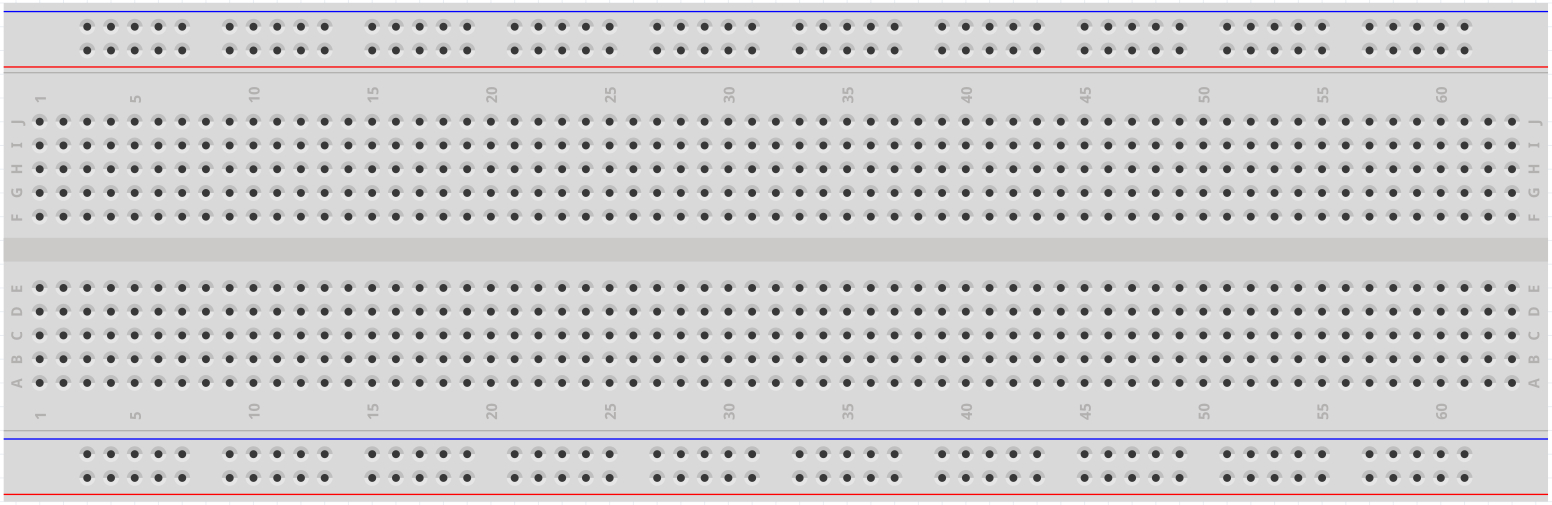
* Purpose: Serves as a manual trigger or reset control for the system.
* Features:
  + Simple input mechanism for user interaction.
  + Durable and reliable for repetitive use.

1. Power Supply (3.7V)

* Purpose: Powers the entire system, ensuring uninterrupted operation.
* Features:
  + Compact and portable.
  + Provides stable voltage to support all components, including the Arduino Nano and modules.

1. Breadboard

* Purpose: Provides a platform for connecting components during prototyping and testing.
* Features:
  + Solderless connections for easy assembly and modifications.
  + Organized layout to connect sensors, modules, and microcontrollers seamlessly.
  + Enables secure integration of jumper wires and power supply for stable performance.

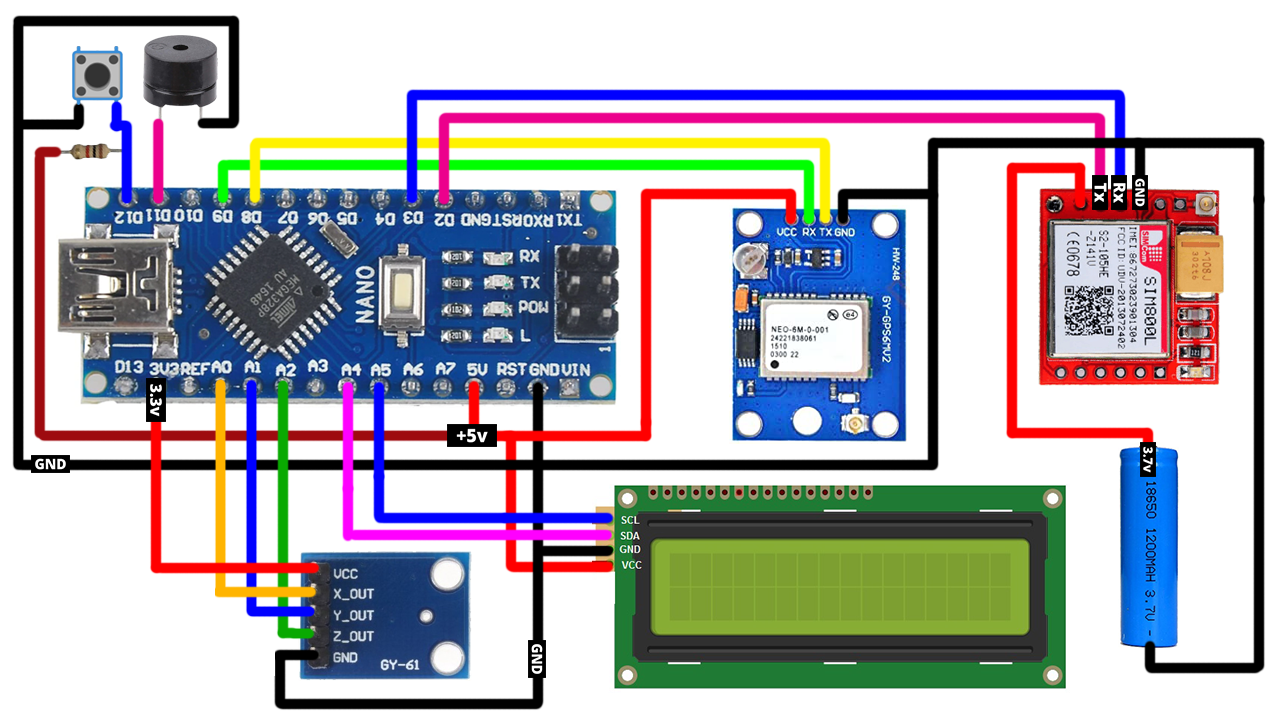


Integration of Components

* The Arduino Nano acts as the central processing unit, connecting to the accelerometer and gyroscope for data input, and to the GSM and GPS modules for communication.
* The LCD with I2C displays status messages and real-time data, while jumper wires connect all components to the power supply and facilitate data flow.
* The button is used for manual intervention, such as resetting the system or triggering test operations.

By combining these components, the Real-Time Accident Alert System achieves a compact, efficient, and scalable design suitable for real-world applications.

5. Wiring Diagram with Explanation:



1. Arduino Nano

* Power:
  + Connected to a 5V power supply for powering the Arduino Nano and its peripherals.
  + Ground (GND) is shared across all components to ensure a common reference voltage.
* I/O Pins:
  + Digital and Analog pins are used to interface with sensors and module.

2. GSM and GPS Modules

* GSM Module:
  + VCC: Connected to 5V on the Arduino Nano.
  + GND: Connected to GND of the Arduino Nano.
  + TX (Transmitter): Connected to RX (Receiver) pin on the Arduino Nano.
  + RX (Receiver): Connected to TX (Transmitter) pin on the Arduino Nano.
* GPS Module:
  + VCC: Connected to 3.3V or 5V, depending on module requirements.
  + GND: Connected to GND.
  + TX: Connected to RX pin on the Arduino Nano.
  + RX: Connected to TX pin on the

3. Accelerometer and Gyroscope (IMU Sensor)

* Power:
  + VCC: Connected to 3.3V on the Arduino Nano.
  + GND: Connected to GND.
* Data Pins:
  + X\_OUT, Y\_OUT, Z\_OUT: Connected to analog pins on the Arduino Nano to measure motion data.

4. LCD with I2C Interface

* Power:
  + VCC: Connected to 5V on the Arduino Nano.
  + GND: Connected to GND.
* Data Pins:
  + SCL (Serial Clock Line): Connected to the A5 pin on the Arduino Nano.
  + SDA (Serial Data Line): Connected to the A4 pin on the Arduino Nano.

5. Button

* Purpose: Provides manual input for resetting or testing the system.
* Connection:
  + One terminal connected to a digital pin on the Arduino Nano.
  + Other terminal connected to GND with a pull-down resistor.

6. Power Supply (2.5V to 5V)

* Provides stable power to the entire circuit.
* The battery connects directly to the VIN and GND pins of the Arduino Nano.

Breadboard Integration

* Used to prototype the connections in an organized manner.
* Ensures the components are securely connected without soldering.

6. Results and Observation:

The Real-Time Accident Alert System was successfully implemented and tested under different scenarios to evaluate its functionality and performance. The results and observations are as follows:

1. System Performance

* The system accurately detects sudden changes in motion using the Accelerometer and Gyroscope (IMU Sensor).
* When an accident or abnormal movement is detected, the system sends an SMS alert using the GSM Module to predefined emergency contacts.
* The GPS Module provides precise location coordinates, which are included in the alert message to assist responders in locating the accident site quickly.

2. Observations during Testing

a. Detection Accuracy

* The system effectively differentiates between regular movements (e.g., bumps or small vibrations) and significant impacts (e.g., collisions).
* By fine-tuning the threshold values for motion detection, false positives were minimized during real-world testing.

b. GPS Location Tracking

* The GPS Module delivered accurate location data within a margin of error of approximately 5-10 meters, depending on environmental factors.
* In open areas, GPS signals were stable and reliable, but in urban environments (e.g., near tall buildings), minor delays in signal acquisition were observed.

c. GSM Communication

* The GSM Module successfully transmitted SMS alerts in less than 5 seconds under normal network conditions.
* Delays of up to 10 seconds were experienced in areas with weak network coverage.

d. LCD Display

* The LCD with I2C interface provided real-time feedback, displaying the system status, GPS coordinates, and alerts.
* The display was clear and responsive, making it easy to monitor system activity.

3. Power Consumption

* The system demonstrated efficient power usage with the Arduino Nano and peripheral components consuming minimal power.
* Using a 2.5V to 5V power supply, the system operated for over 4 hours on a fully charged battery during testing.
* Observations indicated that the GSM and GPS modules were the primary power-consuming components.

4. Scalability and Portability

* The compact design using Arduino Nano and a breadboard allowed the system to be portable and lightweight.
* The modular nature of the design enables future scalability, such as adding more sensors or integrating additional communication protocols (e.g., Wi-Fi or Bluetooth).

Results Summary:

| Parameter | Result/Observation |
| --- | --- |
| Accident Detection | Successfully detected sudden impacts and transmitted alerts effectively. |
| GPS Accuracy | Precise location from Anywhere in the Country. |
| SMS Alert Speed | Average SMS transmission time: 5-10 seconds. |
| LCD Display | Clear and responsive, displaying system status in real-time. |
| Power Efficiency | Operated for over 4 hours on a fully charged battery. |
| Portability | Compact, lightweight, and easy to transport. |

1. Conclusion:

* The Real-Time Accident Alert System is a comprehensive and innovative solution designed to enhance road safety and provide immediate assistance in the event of an accident. Through the integration of advanced components such as the Arduino Nano, GSM and GPS modules, Accelerometer and Gyroscope, and an LCD display, the system effectively detects accidents, pinpoints the location, and notifies emergency contacts in real-time.
* The project successfully met its objectives, demonstrating reliability, accuracy, and practicality during testing. The system efficiently detected significant impacts, minimized false positives, and provided accurate location data with minimal delays. The use of a modular design ensures that the system is scalable, portable, and suitable for a wide range of applications, such as vehicles, construction sites, and other safety-critical environments.
* This project highlights the potential of combining embedded systems and real-time communication technologies to address critical societal challenges. By providing rapid emergency notifications and precise location data, the Real-Time Accident Alert System has the potential to save lives and reduce response times in accident scenarios.

Future Scope:

* The system can be further enhanced by:
* Integrating additional sensors for more detailed accident analysis.
* Implementing smartphone app compatibility for user-friendly monitoring.
* Optimizing power consumption for prolonged operation.
* Adding features like voice alerts and advanced communication protocols (e.g., Wi-Fi or Bluetooth).
* In conclusion, this project serves as a strong foundation for the development of safety-oriented technologies and demonstrates the practical application of embedded systems and IoT in addressing real-world problems. It is a significant contribution to the field of smart safety systems and paves the way for future advancements in road safety technologies.

8. Arduino & Programming:

Arduino is an open-source electronics platform that simplifies the process of designing and building interactive systems. It combines a microcontroller-based hardware board with a user-friendly Integrated Development Environment (IDE) for programming. Due to its versatility and ease of use, Arduino has become one of the most popular platforms for prototyping and developing a wide range of applications.

Applications of Arduino

Arduino is used across various domains, including:

1. Embedded Systems and Automation:
   * Home automation systems for controlling appliances like lights, fans, and security devices.
   * Industrial automation projects, such as temperature and humidity monitoring.
2. Robotics:
   * Arduino is extensively used in robotic systems to control actuators, sensors, and communication modules.
   * Examples include line-following robots, robotic arms, and drones.
3. IoT (Internet of Things):
   * Arduino serves as a backbone for IoT projects, allowing sensors to communicate with cloud platforms.
   * Applications include smart agriculture, connected weather stations, and healthcare monitoring systems.
4. Education and Research:
   * Arduino is an excellent tool for teaching programming and electronics in schools and universities.
   * It is used in research projects for developing innovative solutions in fields like renewable energy and biomedical engineering.
5. Creative Projects:
   * Artists and designers use Arduino for interactive installations, such as light and sound displays.
   * DIY enthusiasts create projects like automatic pet feeders and smart alarms.

Programming Arduino:

Arduino programming is based on a simplified version of C/C++, making it accessible to beginners while still powerful for advanced users. The Arduino IDE provides a seamless environment for writing, compiling, and uploading code to Arduino boards.

Structure of Arduino Code

An Arduino sketch (program) consists of two primary functions:

1. setup():
   * This function runs once when the program starts.
   * It is used to initialize variables, pin modes, and libraries.

Example:

void setup() { pinMode(13, OUTPUT); // Sets pin 13 as an output }

1. loop():
   * This function runs continuously after setup().
   * It contains the core logic that keeps the program running.

Example:

void loop() { digitalWrite(13, HIGH); // Turn on LED delay(1000); // Wait for 1 second digitalWrite(13, LOW); // Turn off LED delay(1000); // Wait for 1 second }

Libraries in Arduino

Arduino provides a wide range of libraries to simplify complex tasks. For example:

* Servo.h: For controlling servo motors.
* Wire.h: For I2C communication.
* SoftwareSerial.h: For additional serial communication ports.

Uploading Code to Arduino

1. Write the program in the Arduino IDE.
2. Connect the Arduino board to a computer using a USB cable.
3. Select the correct board and port in the IDE.
4. Click the "Upload" button to transfer the code to the board.

Advantages of Arduino Programming

* Ease of Use: Arduino’s programming environment is simple and intuitive.
* Extensive Community Support: The large Arduino community provides tutorials, examples, and libraries.
* Cost-Effectiveness: Arduino boards are affordable and widely available.
* Compatibility: Supports integration with a variety of sensors, modules, and actuators.

In the Real-Time Accident Alert System, Arduino plays a crucial role in processing sensor data, making decisions based on predefined thresholds, and coordinating actions like sending alerts via communication modules. Its versatility and reliability make it an ideal choice for this project.

9. Complete Source Code.

#include<LiquidCrystal\_I2C.h>

#include <AltSoftSerial.h>

#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#include <math.h>

#include<Wire.h>

//must add i2c lcd address use i2c-scanner.ino file

LiquidCrystal\_I2C lcd(0x27, 16, 2);

//--------------------------------------------------------------

//emergency phone number with country code

const String EMERGENCY\_PHONE = "+923198194328";

//--------------------------------------------------------------

//GSM Module RX pin to Arduino 3

//GSM Module TX pin to Arduino 2

#define rxPin 2

#define txPin 3

SoftwareSerial sim800(rxPin,txPin);

//--------------------------------------------------------------

//GPS Module RX pin to Arduino 9

//GPS Module TX pin to Arduino 8

AltSoftSerial neogps;

TinyGPSPlus gps;

//--------------------------------------------------------------

String sms\_status,sender\_number,received\_date,msg;

String latitude, longitude;

//--------------------------------------------------------------

#define BUZZER 12

#define BUTTON 11

//--------------------------------------------------------------

#define xPin A1

#define yPin A2

#define zPin A3

//--------------------------------------------------------------

byte updateflag;

int xaxis = 0, yaxis = 0, zaxis = 0;

int deltx = 0, delty = 0, deltz = 0;

int vibration = 2, devibrate = 75;

int magnitude = 0;

int sensitivity = 20;

double angle;

boolean impact\_detected = false;

//Used to run impact routine every 2mS.

unsigned long time1;

unsigned long impact\_time;

unsigned long alert\_delay = 30000; //30 seconds

//--------------------------------------------------------------

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* setup() function

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void setup()

{

  //--------------------------------------------------------------

  //Serial.println("Arduino serial initialize");

  Serial.begin(9600);

  //--------------------------------------------------------------

  //Serial.println("SIM800L serial initialize");

  sim800.begin(9600);

  //--------------------------------------------------------------

  //Serial.println("NEO6M serial initialize");

  neogps.begin(9600);

  //--------------------------------------------------------------

  pinMode(BUZZER, OUTPUT);

  pinMode(BUTTON, INPUT\_PULLUP);

  //--------------------------------------------------------------

  //initialize lcd screen

  lcd.begin();

  // turn on the backlight

  lcd.backlight();

  lcd.clear();

  //--------------------------------------------------------------

  sms\_status = "";

  sender\_number="";

  received\_date="";

  msg="";

  //--------------------------------------------------------------

  sim800.println("AT"); //Check GSM Module

  delay(1000);

  //SendAT("AT", "OK", 2000); //Check GSM Module

  sim800.println("ATE1"); //Echo ON

  delay(1000);

  //SendAT("ATE1", "OK", 2000); //Echo ON

  sim800.println("AT+CPIN?"); //Check SIM ready

  delay(1000);

  //SendAT("AT+CPIN?", "READY", 2000); //Check SIM ready

  sim800.println("AT+CMGF=1"); //SMS text mode

  delay(1000);

  //SendAT("AT+CMGF=1", "OK", 2000); //SMS text mode

  sim800.println("AT+CNMI=1,1,0,0,0"); /// Decides how newly arrived SMS should be handled

  delay(1000);

  //SendAT("AT+CNMI=1,1,0,0,0", "OK", 2000); //set sms received format

  //AT +CNMI = 2,1,0,0,0 - AT +CNMI = 2,2,0,0,0 (both are same)

  //--------------------------------------------------------------

  time1 = micros();

  //Serial.print("time1 = "); Serial.println(time1);

  //--------------------------------------------------------------

  //read calibrated values. otherwise false impact will trigger

  //when you reset your Arduino. (By pressing reset button)

  xaxis = analogRead(xPin);

  yaxis = analogRead(yPin);

  zaxis = analogRead(zPin);

  //--------------------------------------------------------------

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* loop() function

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop()

{

  //--------------------------------------------------------------

  //call impact routine every 2mS

  if (micros() - time1 > 1999) Impact();

  //--------------------------------------------------------------

  if(updateflag > 0)

  {

    updateflag=0;

    Serial.println("Impact detected!!");

    Serial.print("Magnitude:"); Serial.println(magnitude);

    getGps();

    digitalWrite(BUZZER, HIGH);

    impact\_detected = true;

    impact\_time = millis();

    lcd.clear();

    lcd.setCursor(0,0); //col=0 row=0

    lcd.print("Crash Detected");

    lcd.setCursor(0,1); //col=0 row=1

    lcd.print("Magnitude:"+String(magnitude));

  }

  //--------------------------------------------------------------

  if(impact\_detected == true)

  {

    if(millis() - impact\_time >= alert\_delay) {

      digitalWrite(BUZZER, LOW);

      makeCall();

      delay(1000);

      sendAlert();

      impact\_detected = false;

      impact\_time = 0;

    }

  }

  if(digitalRead(BUTTON) == LOW){

    delay(200);

    digitalWrite(BUZZER, LOW);

    impact\_detected = false;

    impact\_time = 0;

  }

  //--------------------------------------------------------------

  while(sim800.available()){

    parseData(sim800.readString());

  }

  //--------------------------------------------------------------

  while(Serial.available())  {

    sim800.println(Serial.readString());

  }

  //--------------------------------------------------------------

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* Impact() function

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Impact()

{

  //--------------------------------------------------------------

  time1 = micros(); // resets time value

  //--------------------------------------------------------------

  int oldx = xaxis; //store previous axis readings for comparison

  int oldy = yaxis;

  int oldz = zaxis;

  xaxis = analogRead(xPin);

  yaxis = analogRead(yPin);

  zaxis = analogRead(zPin);

  //--------------------------------------------------------------

  //loop counter prevents false triggering. Vibration resets if there is an impact. Don't detect new changes until that "time" has passed.

  vibration--;

  //Serial.print("Vibration = "); Serial.println(vibration);

  if(vibration < 0) vibration = 0;

  //Serial.println("Vibration Reset!");

  if(vibration > 0) return;

  //--------------------------------------------------------------

  deltx = xaxis - oldx;

  delty = yaxis - oldy;

  deltz = zaxis - oldz;

  //Magnitude to calculate force of impact.

  magnitude = sqrt(sq(deltx) + sq(delty) + sq(deltz));

  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

  if (magnitude >= sensitivity) //impact detected

  {

    updateflag=1;

    // reset anti-vibration counter

    vibration = devibrate;

  }

  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

  else

  {

    //if (magnitude > 15)

      //Serial.println(magnitude);

    //reset magnitude of impact to 0

    magnitude=0;

  }

  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* parseData() function

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void parseData(String buff){

  Serial.println(buff);

  unsigned int len, index;

  //--------------------------------------------------------------

  //Remove sent "AT Command" from the response string.

  index = buff.indexOf("\r");

  buff.remove(0, index+2);

  buff.trim();

  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

  if(buff != "OK"){

    //--------------------------------------------------------------

    index = buff.indexOf(":");

    String cmd = buff.substring(0, index);

    cmd.trim();

    buff.remove(0, index+2);

    //Serial.println(buff);

    //--------------------------------------------------------------

    if(cmd == "+CMTI"){

      //get newly arrived memory location and store it in temp

      //temp = 4

      index = buff.indexOf(",");

      String temp = buff.substring(index+1, buff.length());

      temp = "AT+CMGR=" + temp + "\r";

      //AT+CMGR=4 i.e. get message stored at memory location 4

      sim800.println(temp);

    }

    //--------------------------------------------------------------

    else if(cmd == "+CMGR"){

      //extractSms(buff);

      //Serial.println(buff.indexOf(EMERGENCY\_PHONE));

      if(buff.indexOf(EMERGENCY\_PHONE) > 1){

        buff.toLowerCase();

        //Serial.println(buff.indexOf("get gps"));

        if(buff.indexOf("get gps") > 1){

          getGps();

          String sms\_data;

          sms\_data = "GPS Location Data\r";

          sms\_data += "http://maps.google.com/maps?q=loc:";

          sms\_data += latitude + "," + longitude;

          sendSms(sms\_data);

        }

      }

    }

    //--------------------------------------------------------------

  }

  else{

  //The result of AT Command is "OK"

  }

  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* getGps() Function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void getGps()

{

  // Can take up to 60 seconds

  boolean newData = false;

  for (unsigned long start = millis(); millis() - start < 2000;){

    while (neogps.available()){

      if (gps.encode(neogps.read())){

        newData = true;

        break;

      }

    }

  }

  if (newData) //If newData is true

  {

    latitude = String(gps.location.lat(), 6);

    longitude = String(gps.location.lng(), 6);

    newData = false;

  }

  else {

    Serial.println("No GPS data is available");

    latitude = "";

    longitude = "";

  }

  Serial.print("Latitude= "); Serial.println(latitude);

  Serial.print("Lngitude= "); Serial.println(longitude);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* sendAlert() function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sendAlert()

{

  String sms\_data;

  sms\_data = "Accident Alert!!\r";

  sms\_data += "http://maps.google.com/maps?q=loc:";

  sms\_data += latitude + "," + longitude;

  sendSms(sms\_data);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* makeCall() function

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void makeCall()

{

  Serial.println("calling....");

  sim800.println("ATD"+EMERGENCY\_PHONE+";");

  delay(20000); //20 sec delay

  sim800.println("ATH");

  delay(1000); //1 sec delay

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* sendSms() function

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

 void sendSms(String text)

{

  //return;

  sim800.print("AT+CMGF=1\r");

  delay(1000);

  sim800.print("AT+CMGS=\""+EMERGENCY\_PHONE+"\"\r");

  delay(1000);

  sim800.print(text);

  delay(100);

  sim800.write(0x1A); //ascii code for ctrl-26 //sim800.println((char)26); //ascii code for ctrl-26

  delay(1000);

  Serial.println("SMS Sent Successfully.");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* SendAT() function

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

boolean SendAT(String at\_command, String expected\_answer, unsigned int timeout){

    uint8\_t x=0;

    boolean answer=0;

    String response;

    unsigned long previous;

    //Clean the input buffer

    while( sim800.available() > 0) sim800.read();

    sim800.println(at\_command);

    x = 0;

    previous = millis();

    //this loop waits for the answer with time out

    do{

        //if there are data in the UART input buffer, reads it and checks for the asnwer

        if(sim800.available() != 0){

            response += sim800.read();

            x++;

            // check if the desired answer (OK) is in the response of the module

            if(response.indexOf(expected\_answer) > 0){

                answer = 1;

                break;

            }

        }

    }while((answer == 0) && ((millis() - previous) < timeout));

  Serial.println(response);

  return answer;

}