VIT-AP UNIVERSITY

ENGINEERING CLINICS I

FINAL REPORT

ACCIDENT PREVENTION, DETECTION AND REPORTING SYSTEM USING ARDUINO

Team Members:

1. T. Maniram : 22BCE9641
2. B. Charitha : 22BCE9980
3. R. Veera Venkata Naga Rahul Reddy : 22BCE8855
4. K. Venkatesh : 22BCE9585
5. T. Hemanth Sai : 22BCE8693

Guided by: Dr. Bibhab Bandhu Majumdar

Abstract:

This report includes the development of accident detection system by integrating components such as Sim900A GSM module and Sim28ML GPS modules for real-time communication and precise GPS tracking and, an MQ-3 gas sensor for environmental monitoring, an accelerometer for impact detection, and a motor driver interfaced with an Arduino Mega for responsive control.

The interconnected system demonstrates a integrates of communication, sensing, and control technologies, enhancing emergency response mechanisms. The report provides insights into the integration process, challenges faced, and solutions implemented, during the development of this prototype.

CONTENTS

LIST OF FIGURES

INTRODUCTION:

In today's world, the use of vehicles has skyrocketed, resulting in an increase in traffic and subsequently, a rise in road accidents. These accidents not only cause damage to property but also result in the loss of human lives due to the lack of immediate preventive and safety measures. While complete accident prevention may be unattainable, we can take steps to reduce the repercussions.

User interaction is enabled through a Bluetooth module, allowing remote control of the vehicle via a Bluetooth-enabled device. Real-time feedback on the system's status is provided through an LCD display, enhancing user awareness. Additionally, GPS tracking offers precise location data during emergencies, aiding in immediate response by accurately pinpointing the vehicle's location.

Motivation:

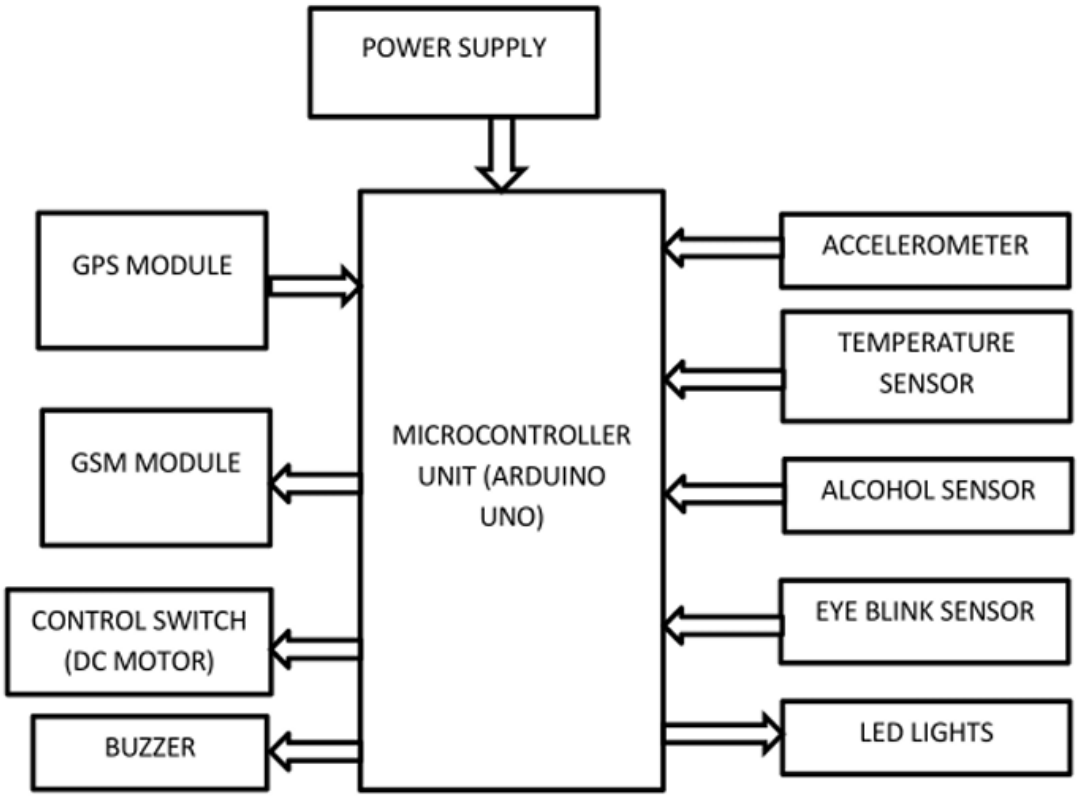
In today's world, the use of vehicles has become increasingly prevalent. While this has provided convenience and efficiency in transportation, it has also led to a rise in traffic and, subsequently, road accidents. Unfortunately, these accidents not only cause damage to property but also result in the loss of precious human lives due to the absence of immediate preventive and safety measures.

While it may not be possible to completely eradicate accidents, it is crucial to take necessary steps to reduce the impacts and prevent their occurrence. This is where embedded systems come into play. By leveraging advanced technologies, such as sensors and microcontroller units, we can develop a system that not only prevents accidents but also ensures prompt preventive measures. In this article, we will explore the components and functionality of such a system, highlighting its potential in mitigating accidents and protecting lives.

Problem Definition:

The accident system we will be discussing is designed to enhance road safety by utilizing various technological components. These include the eye blink sensor, temperature sensor, alcohol sensor, accelerometer, GPS module, GSM module, motor, buzzer, LED, and a central microcontroller unit. This comprehensive system takes advantage of the capabilities of each device to create a cohesive solution for accident prevention.

Methodology/Procedure:



1.1 Block diagram

This prototype primary goal is to detect ,prevent &to inform about the accident. We put together all of the hardware parts, including the Arduino UNO,MQ3 alcohol sensor, LCD display ,eye blink sensor ,buzzer ,accelerometer, HC05 Bluetooth Module, Sim900A GSM module ,Sim28ml GPS module and its antenna, as shown in the block diagram. The Arduino Mega is set up and configured properly. By connecting it to a laptop, we can power the Arduino Mega . The data is fed into the Arduino, which has a GPS modem interface.

The central processing unit, an Arduino Mega, serves as the brain of the system, managing inputs from various sensors and executing responses through actuators. Key sensors include an MQ-3 Alcohol Sensor, which prevents alcohol-impaired driving by restricting vehicle start-up if alcohol levels surpass a defined threshold. An eye-blink sensor continuously monitors driver fatigue, ensuring proactive interventions to maintain driver alertness during operation.

Collision detection is facilitated through an accelerometer, which measures acceleration and identifies sudden changes indicative of collisions or impacts. In the event of a collision, the system triggers safety measures, such as braking, and notifies emergency contacts through a GSM module, which also facilitates sending SMS alerts and making emergency calls.

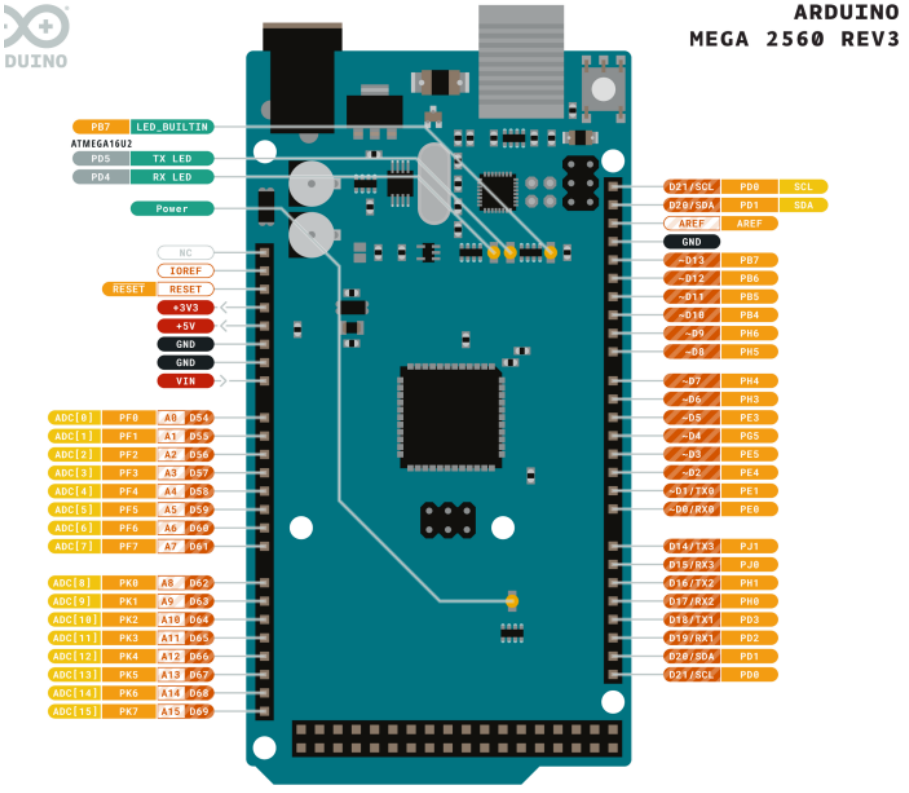
TOOLS REQUIRED

I. HARDWARE COMPONENTS :

- Arduino uno
- Chassis Kit
- Motor Driver(L289N)
- Bluetooth module(HC-05)
- GPS module(SIM28ML)
- GSM module (SIM900A)
- MQ-3 Alcohol sensor
- Accelerometer
- Eye Blink sensor
- LCD Display
- Jumper Wire

1. Arduino Mega:

Arduino Mega 2560 is an exemplary development board dedicated for building extensive applications as compared to other maker boards by Arduino. The board accommodates the ATmega2560 microcontroller, which operates at a frequency of 16 MHz. The board contains 54 digital input/output pins, 16 analog inputs, 4 UARTs (hardware serial ports), a USB connection, a power jack, an ICSP header, and a reset button.



1.2 Arduino Mega

The Mega board is considered superior in terms of SRAM memory space .It is well-suited for most of the Arduino shields. The bootloader is used to eliminate the exterior burner utilization .It is generally used to create complex projects due to its structure. It comes with more memory space, bigger size and more I/O pins .Speedy communication can be achieved since there is a reset button and 4 hardware serial port (USART).

There are three ways to power the board i.e. either through USB cable, or by using Vin pin of the board, or through Power jack. This board comes with resettable polyfuse that prevents the USB port of your computer from overheating in the presence of high current flowing through the board. This board comes with two voltage regulator i.e. 5V and 3.3V which provides the flexibility to regulate the voltage as per requirements.

ARDUINO SETUP :

1.Install the Arduino IDE (Integrated Development Environment)

2. Using a USB cable, connect the Arduino board to your computer.

3. Select the Correct Board(Arduino Mega) and Port to which the arduino is connected.

4. Write Your First Arduino Sketch and verify the output.

5.Upload the Sketch with External Components and verify whether the components are

working.

2. Chassis Kit:

The chassis kit used in the Smart Car Automation System project serves as the structural foundation for the vehicle, providing a robust and versatile platform to accommodate various components. Typically, these chassis kits are designed to be compatible with popular microcontrollers like Arduino and Raspberry Pi, making them suitable for embedded systems projects.

The chassis kit likely includes a solid framework made of durable materials such as aluminum or acrylic. The design often features mounting holes and slots strategically positioned to secure different components like motors, sensors, and the microcontroller board. This modularity allows for easy customization and integration of additional hardware elements.

The kit is likely equipped with multiple motor mounts, providing the necessary support for the DC motors used in the project for controlling the movement of the vehicle. Wheel mounting points are also incorporated to attach wheels securely to the motors. Additionally, the chassis may include a battery compartment or mounting space for the power supply to ensure convenient and organized placement of the energy source.

1.3 Chassis Kit

The overall design of the chassis aims to strike a balance between sturdiness and flexibility, facilitating the attachment of sensors at specific locations for optimal functionality. Cable management features may also be included to keep the wiring organized, contributing to a neat and professional appearance.

3.MOTOR DRIVER(L298N):

The L298N Motor Driver is a key component in the Smart Car Automation System, playing a crucial role in controlling and driving the motors that propel the vehicle. The L298N is designed as a dual H-bridge, allowing it to control two motors independently. Each H-bridge can drive a motor in both directions (forward and reverse) and regulate its speed. The motor driver can handle relatively high currents, making it suitable for driving motors with varying power requirements. This is essential in applications where motors may draw significant current during operation.

The L298N includes built-in diodes that provide protection against back electromotive force (EMF), generated when the motor is turned off. These diodes prevent potential damage to the driver and other connected components. The L298N supports PWM, enabling precise control over the speed of the connected motors. By modulating the width of pulses, it can achieve varying levels of motor speed.

- It has 13 pins: 4 input pins ,4 output pins ,ENA,ENB,GND , 5v&9v pins.
- IN1& IN2 is used to control spinning direction of motor A.
- IN3& IN4 is used to control spinning direction of motor B.
- OUT1 & OUT 2 are output pins of motor A.
- OUT3 & OUT 4 are output pins of motor B.
- ENA will enables PWM(pulse width modulation) signal of motor A.
- ENB will enables PWM signal of motor B.

1.4 motor driver(L298N)

4.BLUETOOTH MODULE(HC-05):

The HC-05 Bluetooth module employed in the Smart Car Automation System plays a pivotal role in enabling wireless communication between the vehicle and external devices, such as smartphones or tablets. The HC-05 is a versatile and cost-effective Bluetooth module widely utilized in embedded systems and Internet of Things (IoT) applications. The HC-05 module operates based on the Bluetooth serial communication protocol (Bluetooth SPP - Serial Port Profile). This makes it compatible with various devices that support Bluetooth, allowing for seamless wireless data transfer.

The HC-05 supports pairing with other Bluetooth devices, ensuring a secure connection. This is essential for the project, especially when considering the remote control capability, where security is crucial to prevent unauthorized access. The module provides a reasonably sufficient communication range, typically up to 10 meters. This range is suitable for the project's application, where the user needs to have control within a close proximity to the vehicle.

- It has six pins: RXD ,TXD,VCC, GND ,LED ,Switch.
- RXD is used for receive serial data. Every serial data given to this pin will be broadcasted via Bluetooth.
- TXD is used to transmits Serial data .Everything received via Bluetooth will be given as output.
- VCC is connected to the power supply.
- LED indicates the status of the module.
- Blink once in 2 sec:Module has entered command mode.
- Repeated blinking :waiting for connection in data mode .

- Blink twice in 1 sec: connection successful in data mode.

1.5 Bluetooth module(HC-05)

5.GSM MODULE(SIM900A):

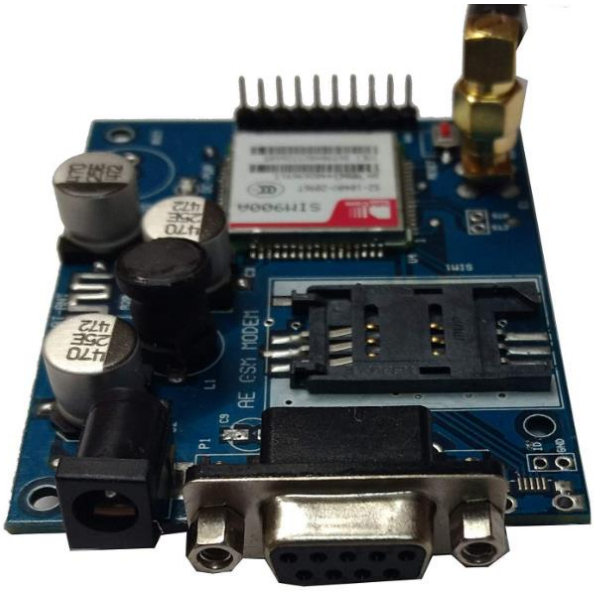
The SIM900A GSM module is a crucial component in the Smart Car Automation System, serving as the communication hub for transmitting data and facilitating interaction between the vehicle and external entities. The SIM900A is a GSM/GPRS (Global System for Mobile Communications/General Packet Radio Service) module designed for seamless wireless communication in embedded systems.

Key features of the SIM900A GSM module include its compact form factor, making it suitable for integration into space-constrained projects. It operates in the 900MHz frequency band, providing robust connectivity in regions where this frequency is supported. The module supports both GSM and GPRS communication standards, enabling voice and data transmission over mobile networks.

In the context of the Smart Car Automation System, the SIM900A GSM module plays a pivotal role in emergency response. In the event of a collision or another predefined emergency scenario, the module initiates communication by sending SMS alerts and making emergency calls to predefined contacts. This capability ensures that relevant parties are promptly notified, contributing to swift and effective responses during critical situations.

Moreover, the SIM900A module enables the system to provide real-time updates and alerts to the user, enhancing the overall functionality of the embedded system. Its reliable communication capabilities, compatibility with microcontrollers, and support for essential communication standards make it a suitable choice for applications demanding efficient and secure wireless communication.

- The 3VR pin should be connected to a 3.0V to 3.6V power supply for the SIM card.
- The GND pins should be connected to ground.
- The 3VT, 5VR, and 5VT pins should be connected to a 5.0V power supply for the module.
- The RX and TX pins should be connected to the serial receive data input and serial transmit data output pins of the microcontroller or other device.
- The DTR and RI pins are used for flow control and can be left unconnected if not needed.
- The SPK_P and SPK_N pins should be connected to a speaker.
- The MIC_P and MIC_N pins should be connected to a microphone.
- The PWRKEY pin can be connected to a push button to turn the module on and off.
- The P2 and P3 pins can be connected to push buttons for user input.
- The GPIO1 and GPIO2 pins are general-purpose input/output pins and can be used for various applications.



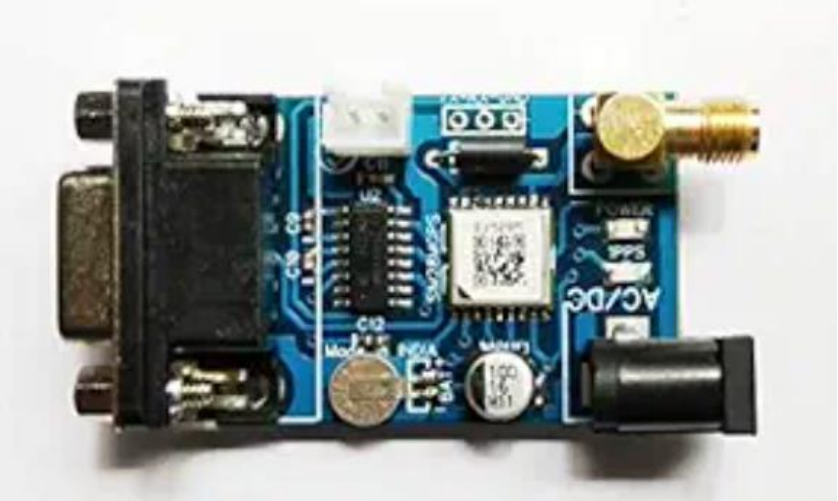
1.6 GSM module(SIM900A)

6.GPS MODULE(SIM28ML):

Smartphones, fleet management systems, military equipment, etc. all typically employ GPS receivers to track or determine location. A satellite-based system called the Global Positioning System (GPS) measures and calculates its position on Earth using satellites and ground stations. NAVSTAR GPS (Navigation System with Time and Ranging) is another name for GPS. In order for a GPS receiver to be accurate, it must receive data from at least four satellites. 8 Information is not sent by the GPS receiver to the satellites. This GPS receiver is utilized in numerous applications, including fleet management, cabs, and smartphones. A GPS receiver determines its precise location no matter where it is by using a constellation of satellites and ground stations. These GPS satellites send information signals to the receiver at radio frequency (1.1 to 1.5 GHz). A ground station or GPS module may calculate its position and time using the data it has received.

- The VCC pin should be connected to a 3.3V to 5.5V power supply.

- The GND pin should be connected to ground.
- The GPS_ANT pin should be connected to a GPS antenna.
- The TXD pin should be connected to the serial receive data input pin of the microcontroller or other device.
- The RXD pin should be connected to the serial transmit data output pin of the microcontroller or other device.
- The PPS pin can be connected to an interrupt pin on the microcontroller or other device to receive a pulse every second.
- The EN pin can be connected to a GPIO pin on the microcontroller or other device to enable or disable the module.
- The VBAT pin can be connected to a 3.0V to 3.6V backup battery to provide power to the module when the main power supply is disconnected.
- The RST pin can be connected to a GPIO pin on the microcontroller or other device to reset the module.



1.7 GPS module(SIM28ML)

7.MQ-3 Alcohol Sensor:

The MQ-3 alcohol sensor is a crucial component in the Smart Car Automation System designed to detect the presence of alcohol vapors in the surrounding environment. This sensor is based on the metal oxide semiconductor (MOS) principle, where the electrical conductivity of a metal oxide material changes when it comes into contact with a specific gas, in this case, alcohol.

The MQ-3 sensor consists of a sensing element made of a tin dioxide (SnO2) semiconductor, which exhibits a change in resistance when exposed to alcohol molecules. When alcohol vapor is present, it undergoes a chemical reaction on the surface of the sensing element, leading to a variation in its conductivity. This change in conductivity is then translated into an electrical signal that can be measured and interpreted by the microcontroller, in this case, the Arduino Mega used in the project.

The sensor provides an analog output that corresponds to the concentration of alcohol in the air. By calibrating the sensor and setting a predefined threshold, the system can determine whether the detected alcohol level is within acceptable limits. If the alcohol concentration exceeds the specified threshold, the system initiates preventive measures, such as restricting the vehicle's start-up to prevent alcohol-impaired driving.

- IT has four pins : VCC, GND, A0 ,D0 pins.
- VCC is connected to power supply.
- A0 pin is connected to analog pin and is used to measure the alcohol value.
- D0 pin is connected to digital output. It is generally high (1),If sensor detects alcohol then the value is changed to 0.

1.8 MQ-3 Alcohol sensor

8. Accelerometer(ADXL335):

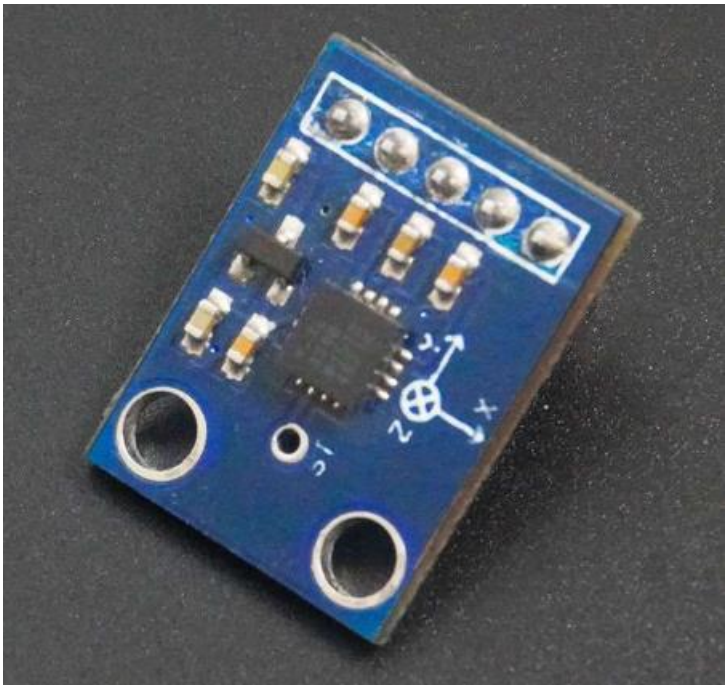
The ADXL335 is a Micro-Electro-Mechanical System (MEMS) accelerometer used in the project. As a key component in the embedded system, this accelerometer plays a crucial role in detecting and measuring acceleration forces in multiple directions. The ADXL335 is a small, low-power sensor capable of measuring static acceleration due to gravity and dynamic acceleration, which includes vibrations and movements.

The ADXL335 is capable of measuring acceleration along three orthogonal axes: X, Y, and Z. This tri-axial capability provides comprehensive information about the vehicle's movement in three-dimensional space .The sensor has user-selectable sensitivity levels, allowing for customization based on the specific requirements of the application. It can measure accelerations in the range of ±3 g, where "g" represents the acceleration due to gravity.

The ADXL335 provides analog voltage outputs proportional to the acceleration along each axis. These analog outputs can be easily interfaced with microcontrollers for further

processing .Designed for low-power applications, the ADXL335 is energy-efficient, making it suitable for embedded systems where power consumption is a critical consideration. The ADXL335 is widely used in applications such as motion detection, tilt sensing, and impact detection. In the smart car automation system, it serves as a collision detection sensor, identifying sudden changes in acceleration that may indicate a collision or impact.

- It has five pins VCC, X_OUT, Y_OUT, Z_OUT, GND
- VCC**:** Power supply pin i.e. connect 5V here.
- X_OUT: X axis analog output**.**
- Y_OUT: Y axis analog output.
- Z_OUT: Z axis analog output.
- GND: Ground pin i.e. connect ground here.



1.9 Accelerometer(ADXL335)

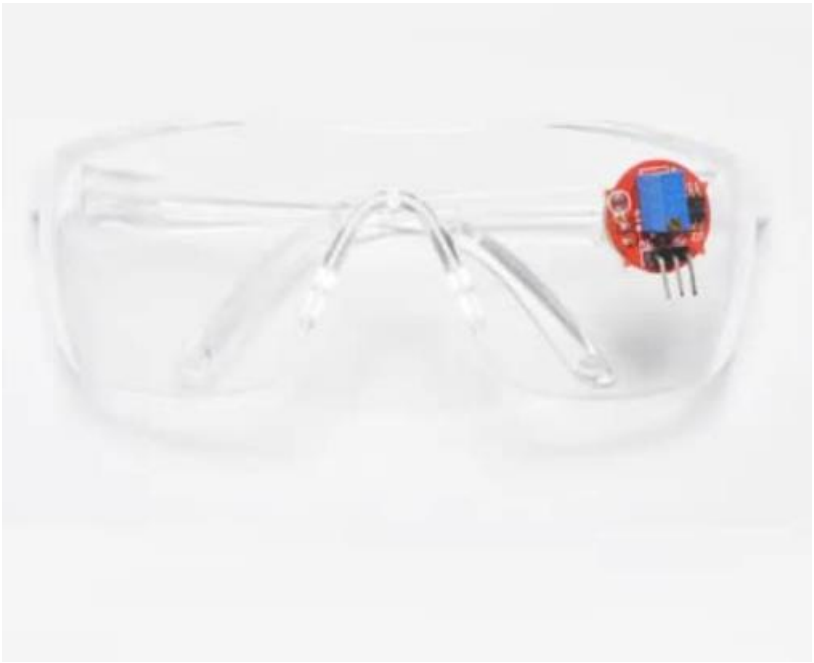
9.Eye Blink Sensor:

The eye blink sensor utilized in the Smart Car Automation System is a critical component designed to monitor and detect the blink patterns of the driver. This sensor is instrumental in assessing the driver's level of alertness and preventing potential accidents caused by fatigue or drowsiness.The eye blink sensor typically employs infrared (IR) light to illuminate the driver's eyes and a photodetector to capture the reflected infrared light. The sensor works on

the principle that the amount of reflected IR light varies with the opening and closing of the eyes. When the eyes are open, more IR light is reflected, and when they blink or close, the reflected IR light decreases.

The sensor's output is processed by the system to determine the duration and frequency of the eye blinks. Sudden and prolonged closures of the eyes, indicating drowsiness or fatigue, trigger the system to activate safety measures, such as alerting the driver or initiating an emergency stop .In the Smart Car Automation System, the eye blink sensor plays a pivotal role in real-time monitoring of the driver's condition. By continuously analyzing eye blink patterns, the system can detect signs of driver drowsiness or distraction and respond promptly to ensure a safe driving environment. This proactive approach to driver monitoring enhances overall road safety and reduces the risk of accidents due to driver fatigue.

- IT has 3 pins: VCC , GND , OP.
- VCC is connected to power supply.
- OP is connected to Digital output. IT has an IR sensor.



1.10 Eye Blink sensor

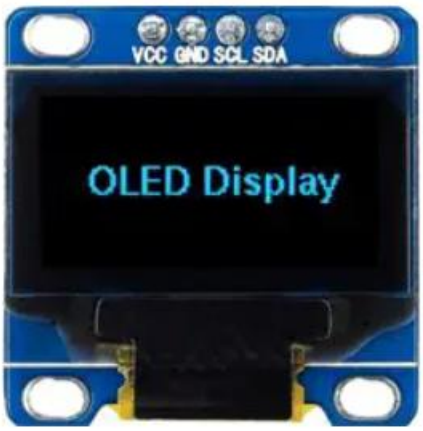
10. OLED DISPLAY :

The organic light-emitting diode (OLED), also known as organic electroluminescent (organic EL) diode, is a light-emitting diode (LED) in which the emissive electroluminescent layer is film of organic compound that emits light in response to an electric current. This organic layer is situated between two electrodes; typically, at least one of these electrodes is

transparent. OLEDs are used to create digital displays in devices such as television screens, computer monitors, and portable systems such as smartphones and handheld game consoles.

OLED is fundamentally different from LED which is based on a p-n diode structure. In LEDs doping is used to create p- and n- regions by changing the conductivity of the host semiconductor. OLEDs do not employ a p-n structure. Doping of OLEDs is used to increase radiative efficiency by direct modification of the quantum-mechanical optical recombination rate. Doping is additionally used to determine the wavelength of photon emission.
The *organic light-emitting diode* (OLED) display that we'll use in this tutorial is the SSD1306 model: a monocolor, 0.96-inch display with 128×64 pixels

- It has four pins VCC,GND,SCL,SDA.
- The VCC pin is connected to the power supply.
- The GND pin is connected to the common ground of the Arduino board.
- A serial clock pin(SCL) is used to control pulses at a regular interval.
- A serial data Pin (SDA) is used to transfer data between two devices.



1.11 OLED Display

11.JUMPER WIRES :

Simply said, jumper wires are wires with connector pins at either end that can be used to connect two places without soldering. With breadboards and other prototype tools, jumper wires are frequently used to make it simple to change a circuit as required. Fairly easy. Jumper wires are actually among the most elementary things there are. Jumpers are typically small metal connections that are used to open or close circuit components. They have two or more connecting points that control a circuit board for an electrical system. They are responsible for setting up the motherboard and other computer devices. They are employed to connect two circuit locations without the usage of soldering.

Jumper wires come in three versions:

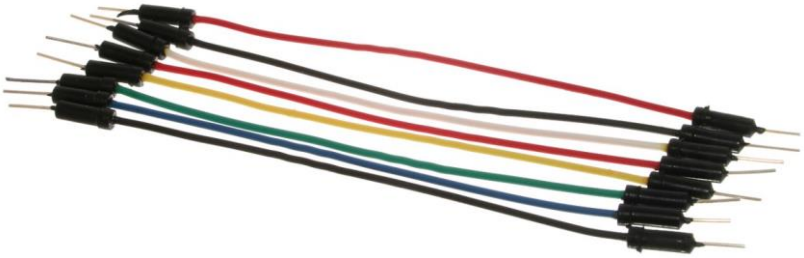
• Male-to-male jumper

• Male-to-female jumper

• Female-to-female jumper

And two types of head shapes: square head and round head

The wire terminus distinguishes each one from the other. While female ends do not have projecting pin yet may also plug into objects, male ends do. Also known as a plug, a male connection contains a solid pin for center conduction. The center conductor of a 11female connector, often known as a jack, has a hole cut out of it to receive the male pin. The most typical and often utilized jumper cables are male to male. For instance, a male-to-male cable is required to connect two ports on a breadboard. The wires are able to be twisted and coiled in any pattern due to the flexible stranded cores. Furthermore, it is simple to thread into any breadboard thanks to the reinforced male tips on either end.



1.12 Jumper wires

I. SOFTWARE COMPONENTS :

- Arduino IDE

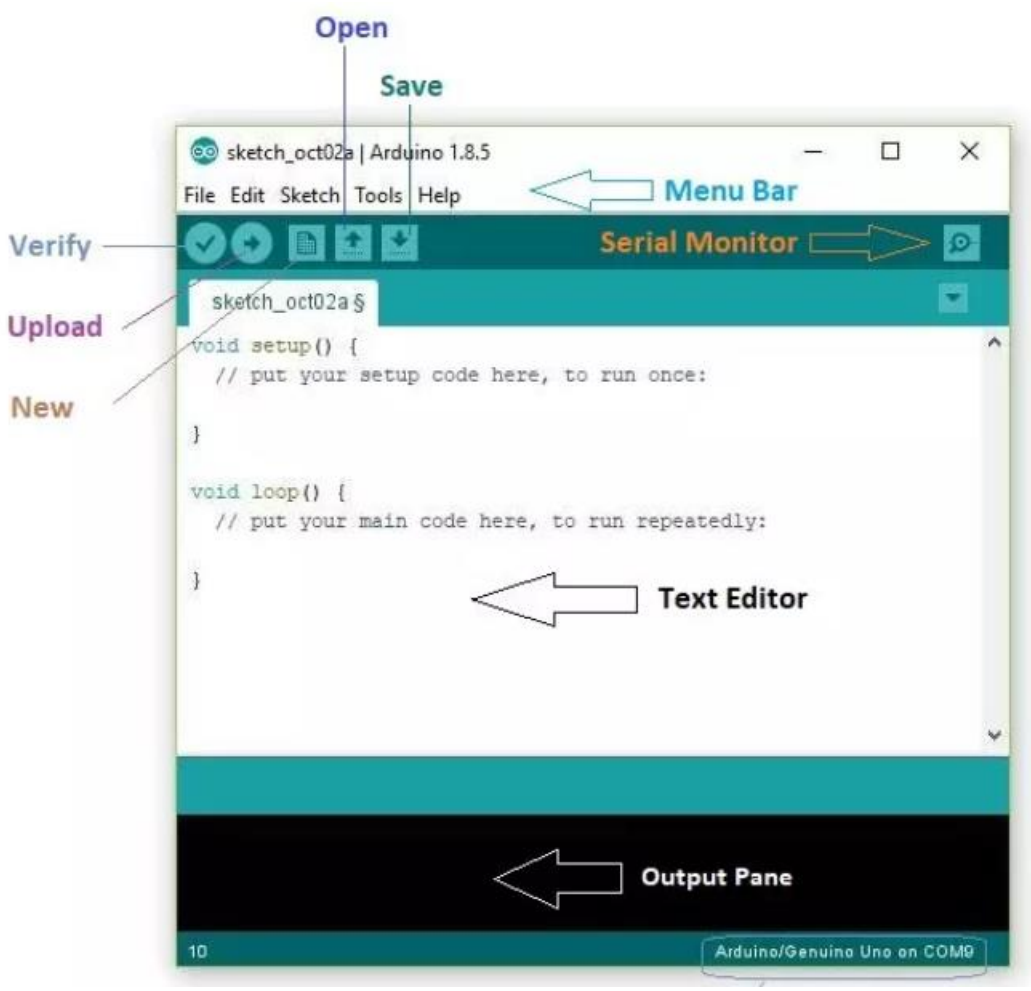
Arduino IDE:

The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as **Windows, Mac OS X, and Linux**. It supports the programming languages C and C++. Here, IDE stands for **Integrated Development Environment**. The program or code written in the Arduino IDE is often called as sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.'

The Upload button compiles and runs our code written on the screen. It further uploads the code to the connected board. Before uploading the sketch, we need to make sure that the correct board and ports are selected .We also need a USB connection to connect the board and the computer. Once all the above measures are done, click on the Upload button present on the toolbar .The latest Arduino boards can be reset automatically before beginning with

Upload. In the older boards, we need to press the Reset button present on it. As soon as the uploading is done successfully, we can notice the blink of the Tx and Rx LED .If the uploading is failed, it will display the message in the error window.

The Open button is used to open the already created file. The selected file will be opened in the current window. The save button is used to save the current sketch or code. The New button is used to create a new sketch or opens a new window. The Verify button is used to check the compilation error of the sketch or the written code. The serial monitor button is present on the right corner of the toolbar. It opens the serial monitor.



1.13 Arduino IDE

CODE : The Code for this prototype is :

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include<SoftwareSerial.h>
#include <AltSoftSerial.h>
#include <TinyGPS++.h>
#include <math.h>

#define in1 3 //L298n Motor Driver pins.
#define in2 4
#define in3 5
#define in4 6
#define LED 2
#define eyeBlinkPin 7   // Pin where the eye blink sensor is connected
#define buzzerPin 8     // Pin for the buzzer

bool eyesClosed = false;
unsigned long eyesClosedStartTime = 0;
unsigned long eyesClosedDuration = 0;

//SoftwareSerial mySerial(2, 3); // RX, TX for GSM module
//SoftwareSerial gps(10, 11);    // RX, TX for GPS module

//-----------------------------------------------------------------
//emergency phone number with country code
const String EMERGENCY_PHONE = "+91XXXXXXXXXX";
//-----------------------------------------------------------------
//GSM Module RX pin to Arduino 11
//GSM Module TX pin to Arduino 10
#define rxPin 10
#define txPin 11
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
//-----------------------------------------------------------------

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2 rows


// Define pins for the alcohol sensor (MQ-3)
const int MQ3 = A0;
bool isAlcoholDetected = false;

int command; //Int to store app command state.
int Speed = 204; // 0 - 255.
int Speedsec;
int buttonState = 0;
int lastButtonState = 0;
int Turnradius = 0; //Set the radius of a turn, 0 - 255 Note:the robot will malfunction if this is higher tha
int brakeTime = 45;
int brkonoff = 1; //1 for the electronic braking system, 0 for normal.

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

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 = 70;
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() {
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(LED, OUTPUT); //Set the LED pin.
  // Initialize the pins for the eye sensor and buzzer
  pinMode(eyeBlinkPin, INPUT);
  pinMode(buzzerPin, OUTPUT);
    // Initialize sensors, modules, and pins
  pinMode(MQ3, INPUT);

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

  //-------------------------------------------------------------
  //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(16, 2);  // Assuming 16 columns and 2 rows

  // 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);
 //------------------------------------------------------------

 //lcd.init();
 //lcd.backlight();
 //lcd.begin(16, 2);
```

```
  //Serial.begin(9600);  //Set the baud rate to your Bluetooth module.
  //mySerial.begin(9600);
  //gps.begin(9600);


  // ... (other pin mode initializations)
  // ... (other setup code)

}

/**********************************************************************************************
 * loop() function
 *********************************************************************************************/

void loop() {

  // Check for alcohol detection
  isAlcoholDetected = checkAlcohol();

  if (isAlcoholDetected) {
    handleAlcoholDetection();
  }

  // ... (other parts of the loop for temperature, eye blink, etc.)

  // ... (Bluetooth car control code)

  if (Serial.available() > 0) {
    command = Serial.read();
    Stop(); //Initialize with motors stoped.
    switch (command) {
      case 'F':
        forward();
        break;
```

```
      case 'B':
        back();
        break;
      case 'L':
        left();
        break;
      case 'R':
        right();
        break;
      case 'G':
        forwardleft();
        break;
      case 'I':
        forwardright();
        break;
      case 'H':
        backleft();
        break;
      case 'J':
        backright();
        break;
      case '0':
        Speed = 100;
        break;
      case '1':
        Speed = 140;
        break;
      case '2':
        Speed = 153;
        break;
      case '3':
        Speed = 165;
        break;
      case '4':
        Speed = 178;
        break;
      case '5':
        Speed = 191;
        break;
```

```
    case '6':
      Speed = 204;
      break;
    case '7':
      Speed = 216;
      break;
    case '8':
      Speed = 229;
      break;
    case '9':
      Speed = 242;
      break;
    case 'q':
      Speed = 255;
      break;
  }
  Speedsec = Turnradius;
  if (brkonoff == 1) {
    brakeOn();
  } else {
    brakeOff();
  }
}

//-----------------------------------------EYE BLINK--------------------------------------

//Now, I'm Wriring the code for the Eye blink sensor

 int eyeBlinkState = digitalRead(eyeBlinkPin);

if (eyeBlinkState == HIGH) {
  // Eyes are open
  eyesClosed = false;
} else {
  // Eyes are closed
  if (!eyesClosed) {
    eyesClosedStartTime = millis();  // Record the time when the eyes first close
    eyesClosed = true;
  }
  eyesClosedDuration = millis() - eyesClosedStartTime;
```

```
  if (eyesClosedDuration > 4000) {
    // Eyes have been closed for more than 4 seconds
    stopCar();  // Implement your car stopping function
    activateBuzzer();
  }
}

// Other code for handling Bluetooth commands and motor control

//-------------------------------------------------------------
//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());
  }
  //------------------------------------------------------------

}
/*********************************************************************************************
 * END OF VOID LOOP
 *********************************************************************************************/

void forward() {
  analogWrite(in1, Speed);
  analogWrite(in3, Speed);
}

 void back() {
   analogWrite(in2, Speed);
   analogWrite(in4, Speed);
 }

 void left() {
   analogWrite(in3, Speed);
   analogWrite(in2, Speed);
 }

 void right() {
   analogWrite(in4, Speed);
   analogWrite(in1, Speed);
 }
 void forwardleft() {
   analogWrite(in1, Speedsec);
   analogWrite(in3, Speed);
 }
 void forwardright() {
   analogWrite(in1, Speed);
   analogWrite(in3, Speedsec);
 }
 void backright() {
   analogWrite(in2, Speed);
   analogWrite(in4, Speedsec);
 }
 void backleft() {
   analogWrite(in2, Speedsec);
   analogWrite(in4, Speed);
 }

 void Stop() {
   analogWrite(in1, 0);
   analogWrite(in2, 0);
   analogWrite(in3, 0);
   analogWrite(in4, 0);
 }
```

```
void brakeOn() {
  //Here's the future use: an electronic braking system!
  // read the pushbutton input pin:
  buttonState = command;
  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == 'S') {
      if (lastButtonState != buttonState) {
        digitalWrite(in1, HIGH);
        digitalWrite(in2, HIGH);
        digitalWrite(in3, HIGH);
        digitalWrite(in4, HIGH);
        delay(brakeTime);
        Stop();
      }
    }
    // save the current state as the last state,
    //for next time through the loop
    lastButtonState = buttonState;
  }
}
void brakeOff() {

}
void stopCar() {
  // Implement your car stopping function
  analogWrite(in1, 0);
  analogWrite(in2, 0);
  analogWrite(in3, 0);
  analogWrite(in4, 0);
}

void activateBuzzer() {
  digitalWrite(buzzerPin, HIGH);  // Turn on the buzzer
  delay(1000);  // Buzzer sounds for 1 second
  digitalWrite(buzzerPin, LOW);   // Turn off the buzzer
}
```

```
bool checkAlcohol() {
  // Read alcohol sensor value
  int alcoholValue = analogRead(MQ3);

  // You may need to adjust the threshold value based on your sensor and environment
  int alcoholThreshold = 500; // Adjust this value accordingly

  // Check if alcohol level is above the threshold
  return alcoholValue > alcoholThreshold;
}

void handleAlcoholDetection() {
  lcd.clear();
  lcd.print("Alcohol Detected");
  lcd.setCursor(0, 1);
  lcd.print("Stopping the car");

  // Implement your car stopping function here
  stopCar();

  // ... (send SMS, GPS location, etc.)
  sendAlert();

  delay(5000); // Display the message for 5 seconds, adjust as needed

  lcd.clear();
  lcd.print("System Ready");
}

// ... (other functions for motor control, GPS, SMS, etc.)
/*****************************************************************************
 * 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));
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  if (magnitude >= sensitivity) //impact detected
  {
    updateflag=1;
    // reset anti-vibration counter
    vibration = devibrate;
  }
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  else
  {
    //if (magnitude > 15)
      //Serial.println(magnitude);
    //reset magnitude of impact to 0
    magnitude=0;
  }
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
}



/*******************************************************************************************
 * 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();
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  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"
  }
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
}


/*****************************************************************************************
 * 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;
}
```

Results:

The automobile automatically stops when the alcohol sensor (MQ-3) detects alcohol .The alert buzzer will begin to sound, along with an alert notification display and a message sent to the number provided in the code. The communication includes the address of the place where the With the aid of the GPS module and GSM, the car halted. If the individual felt sleepy and lethargic the eye blink sensor is going to pick it up. A buzzer will sound if it senses something, and if the driver doesn't press the appropriate button, the automobile will automatically stop; if not, it will continue to operate as usual. The accelerometer of the vehicle detects collisions and issues a warning if one occurs message is delivered to the number with assistance .

Conclusion:

Finally, on performing all the required procedure, we were able to implement Our project on "Accident Prevention, Detection and Reporting System"

In this 21's century, with the continuous advancement in science and technology, more emphasis is given for vehicle safety. With the increase in number of vehicle, of road accident is also increasing day by day, so it is our duty to control it.

Mostly the accident takes place because of drunk drivers, drowsiness while driving and over heating of engine causing fire. Implementation of this project will help to decrease the accident caused because of above reason. The system is automatic, low cost and power efficient which makes it easy to install in vehicle. Unfortunately, if accident happens to take place, the system detects it and with the help of GPS exact location can be pointed and informed to emergency unit using GSM module. This helps to save many lives by informing rescuing agent in time. Over all, this system is very affordable, targets common people and easily implemented in all types of vehicles.

Future scope:

With the completion of this project there are certain enhancement that can be done

- Along with the stopping of vehicle after a driver is found drunk, he can be fined automatically by connecting central system with traffic control room.
- Scientifically proved music which keeps people awake, can be played time to time in driver's cabin.
- CCTV camera can be installed in the driver's cabin and can be controlled from central room of travel agencies, this can prevent accident considerably.
- Vehicle unit can be connected with central server to find contact no. of ambulance and police control room.
- Vehicle if obtains over speed, the accelerometer can determine it and automatically complain to traffic by which driver can be fined. This will limit the speed of vehicle and can prevent from accident too.
- Surveillance using A.I. cameras and solving hit and run cases.

REFERENCES :

- https://www.mdpi.com/2071-1050/14/1/210
- https://ieeexplore.ieee.org/document/7151519
- https://ijeebs.com/wp-content/uploads/2020/06/IJEEBS-2015-V2-I1-004.pdf
- https://youtu.be/47ocmY1CCD8?si=JNAqkFm_-sEBwv7z
- https://youtu.be/WNezQ0maD9I?si=UqN_cUnAVdBPhmda
- https://github.com/imamhossain94/accident-detection-prevention-and-an-emergency-solution
- https://projecthub.arduino.cc/