Automatic Vehicle Accident Detection and Alerting Notification using IoT Technology with ESP32

1st Dhairya Senghani
Department of Electronics and Communication,
Nirma University,
Ahmedabad , India
21bec111@nirmauni.ac.in

2nd Shreya Nahta
Department of Electronics and Communication,
Nirma University,
Ahmedabad , India
21bec117@nirmauni.ac.in

Abstract

The massive population growth in developing nations has indirectly fueled the rapid motorization and widespread expansion of automobiles. This has made traffic accidents inevitable, which is contributing to the high number of fatalities and serious injuries. Because current technology have made vehicles much more intelligent and automated, they have had a major positive impact on reducing the number of accidents. This paper primarily discusses a system that was created using Internet of Things (IoT) technology and integrates an ESP32 microcontroller, an RFID card reading module for licence reading, an LCD display, an ultrasonic sensor for obstacle distance calculation, Accelerometer a GPS unit for latitude and longitude location, an SMTP server to detect accidents and send instant email notifications to emergency contacts and the registered email addresses in order to facilitate rescue efforts.

Index Terms

ESP32 Microcontroller, Internet of things(IoT), LCD Display, Ultrasonic Sensor, RFID Card Reader, GPS Unit, SMTP Server

I. Introduction

The number of accident cases is currently increasing at an incredibly fast pace. Taking into account the extraordinarily large number of accident instances that occur globally every minute, hour, and day. Accidents can range in severity from trivial to deadly, and they typically happen in an unanticipated way. Under certain conditions, an accident may result in the sad loss of life if information is not promptly communicated to the local healthcare facilities or if emergency services or first aid are not accessible.Considering all of these factors, it is imperative to create a system that can evaluate the situation and function well in order to close the gap in time between the occurrence of an accident and the provision of medical care. The main goal of this project is to use IoT technology to create a system that can prevent accidents by taking many factors into account. It also aims to identify and notify the appropriate emergency centres and interested parties in order to quickly address the problem. By integrating smart sensors with the ESP32 microprocessor that is housed inside the car, this may be done. These sensors activate when an accident occurs. The GPS module is used for notification purposes, locating the accident site and notifying surrounding hospitals and worried relatives appropriately. This project makes use of an open source programme called SMTP Server, which is primarily intended for Internet of Things applications where data may be sent via mail with ease [1].

II. Background

A. RFID

RFID is a technology that uses radio waves to identify and track tags attached to objects. The RFID module is used to scan RFID tags embedded in driver's licenses. When a driver's license with an RFID tag is brought near the RFID reader, the module reads the unique identifier stored in the tag. This identifier is then compared with a predefined RFID string in code to determine if the scanned license is valid or not. If the scanned RFID matches the predefined string, it indicates a successful scan, and the system proceeds with the rest of the operations. If the scanned RFID does not match the predefined string, it indicates an unsuccessful scan, and the system prompts the user to rescan the license. This makes sure that only the persons having license can drive.

B. ESP32

This paper explains about the system which uses IoT frame work integrated along with the sensors and a microcontroller ESP32 which are pre-programmed into the vehicle that can be activated at the time of an accident. This device is designed in such a manner that it alerts the respective authorities about an accident only if the passengers have been seriously injured. This device detects accidents on time and triggers the immediate notification.

C. GPS

The Global Positioning System (GPS) module, an electronic device that communicates with GPS satellites to provide geographical location data is utilized from where the signal is sent to the cloud. The signal indicates the severity of the accident and the GPS location. It does not require the user to transmit any data and it operates independently of any internet reception The notification is sent to registered mail and the GPS location can be accessed on SMTP by anyone with the login credentials. By obtaining the GPS coordinates, ambulance can reach the scene immediately.

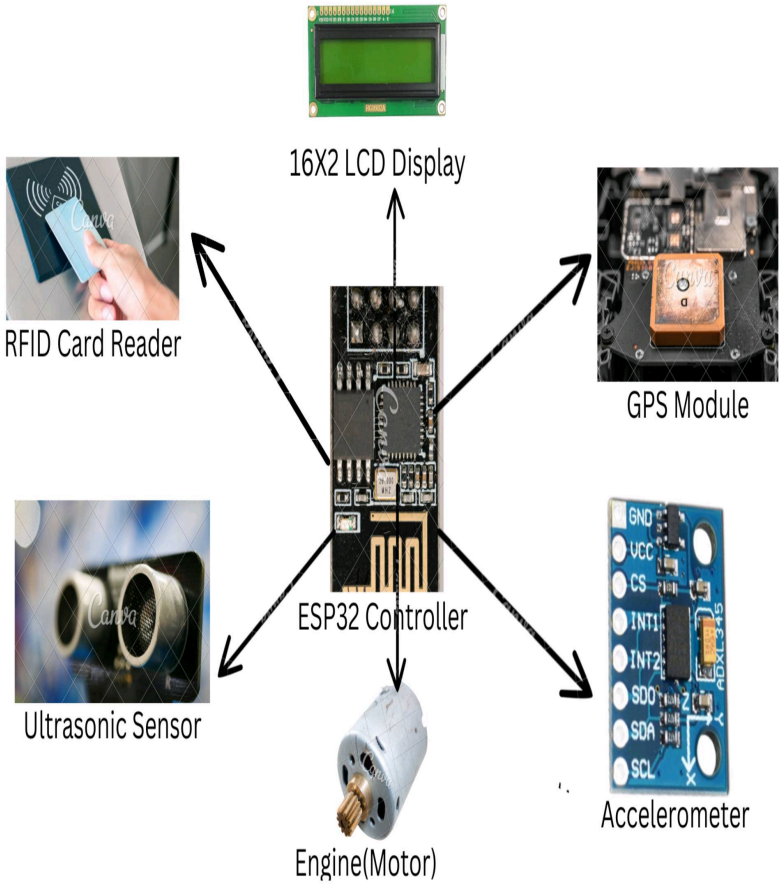

Fig. 1. Alert to Drive Slow

D. Accelerometer

The GY-61 is a small board based on the ADXL-335 Chip. Three sensors are at right angles to one another and measure the acceleration in each of three axes. It's used to detect sudden changes in acceleration, which could indicate an accident.

E. Ultrasonic Sensor

An ultrasonic sensor is used to measure distance by emitting ultrasonic waves and calculating the time it takes for the waves to bounce back. It's used to detect objects in front of the vehicle that are so near, to alert the driver to slow down.

F. WiFi

Wi-Fi is a wireless networking technology that allows devices to connect to the internet and communicate with each other wirelessly. In this paper, it's used to connect the ESP32 board to a Wi-Fi network for internet access, which will help in sending mail.

G. LCD

LCDs are commonly used for visual output in electronic devices. In your code, it's used to display messages and status information, such as "Drive Slow", "WiFi connected" etc.

H. SMTP

SMTP is a protocol used for sending email messages over the internet. Here, it's used to send email alerts in case of an accident, with the specified location of the accident.

III. Proposed System

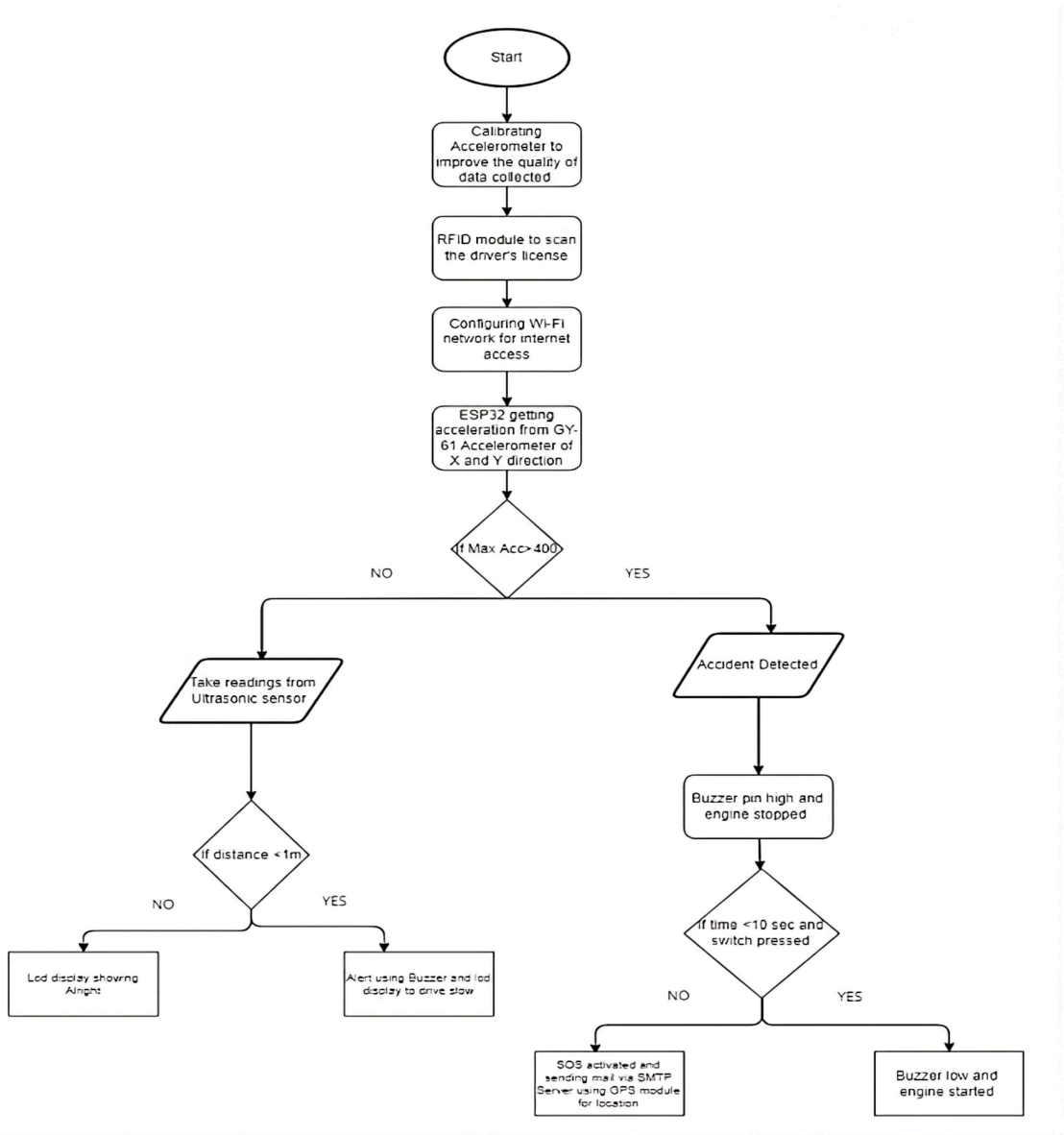

Fig. 2. Flowchart of the proposed system

The prototype system which has been developed detects the accident instantly and informs the concerned people and authorities. In this prototype model, ESP32 microcontroller is considered as the core of the overall system. It is basically considered as it is low power on-chip microcontroller which has in built Wi-Fi and Bluetooth module. In addition to this it is battery friendly [2]. The different sensors are interfaced to ESP32 which detects the various aspects of the vehicle functioning and drivers condition while driving in order to ensure maximum safety. The different cases considered in this prototype system are:

- If the vehicle gets bumped from any X or Y direction and there is sudden change in speed.
- The occurrence of any obstacles near the vehicle is determined then the system is designed to send the pre-notification or warning to drive safely.

A. Step I

When the engine starts up, the system must first scan for licences before allowing the engine to run. If this scan is not finished, the engine stays inert and cannot be started again until the licence has been correctly checked. This safety measure improves system security and guarantees compliance with licensing requirements. The system prioritises safety and compliance by enforcing this requirement, hence reducing possible dangers and unauthorised access.

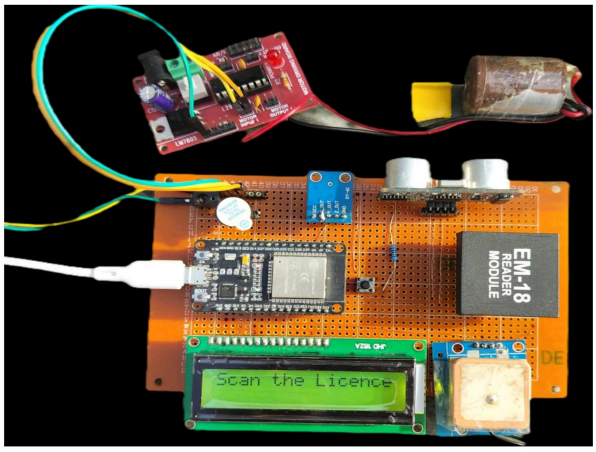

Fig. 3. Scan the License

B. Step II

Immediately after a licence scan is completed, the system shows a confirmation message that reads ”Licence Successful.” Once this validation is finished, the engine is quickly turned on and prepared to perform as intended. This smooth transition indicates that authorised access has been provided and all relevant licencing requirements have been satisfied, indicating that the system is ready to move forward with operational tasks. Users can interact with the engine with confidence as it starts up, knowing that their licence status has been verified.

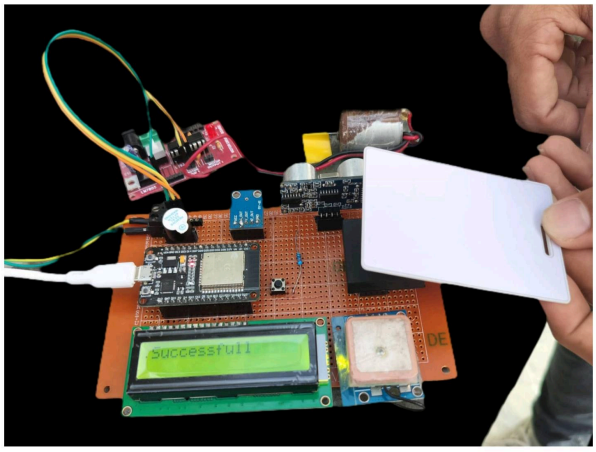

Fig. 4. License Scan Successful

C. Step III

Connecting to the ESP’s (Embedded Systems Platform) Wi-Fi network is the next step once the engine has been started and the licencing procedure has been finished. The device starts a scanning process to find Wi-Fi networks that are accessible in its immediate area. Through this scanning procedure, the system is able to pinpoint the precise network that is connected to the ESP, allowing for smooth device interaction and communication. The system’s functionality and possible uses are further expanded by connecting to the ESP and utilising its resources and capabilities. The system is ready to take advantage of all the capabilities and services made possible by this connectivity now that the Wi-Fi connection has been made and the ESP network has been located.

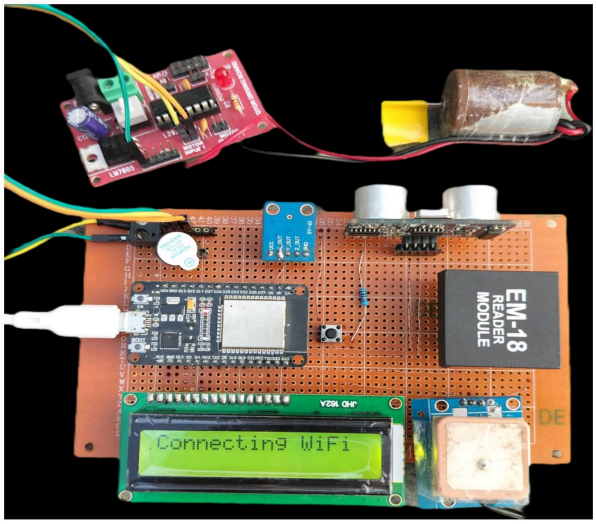

Fig. 5. Connecting to WiFi

D. Step IV

The system immediately shows a comforting message verifying the status after successfully connecting to the WiFi network: "WiFi connected." This message informs the user right away that their device is now connected to the specified network, allowing them to access data transfers, online services, and other functions that rely on the internet. Users can confidently continue with their tasks as the message gives a clear and straightforward signal, removing any doubt regarding the connectivity status. This smooth integration improves user experience by enabling effective interaction and communication in the digital world.

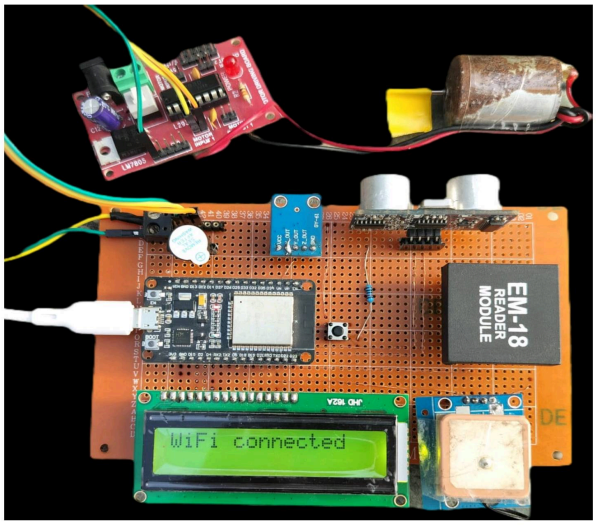

Fig. 6. Wifi Connected Successfully

E. Step V

Now that the engine has started and all systems are primed, the traveller is prepared to set out. When licence verification, WiFi connectivity, and engine starting are all completed, the action is signalled. The engine's readiness signals the start of any journey—be it a drive, a chore, or an adventure. Equipped with confidence on the functionality and compliance of the system, the user can go towards their desired destination with assurance.

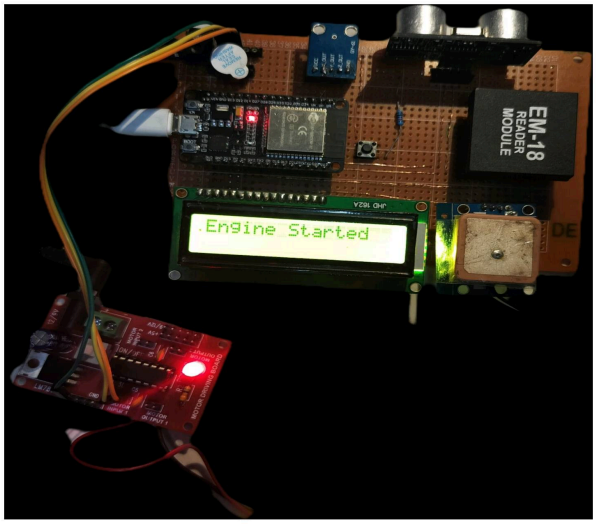

Fig. 7. Engine(Motor) Strated

F. Step VI

As an essential safety precaution, the ultrasonic sensor carefully searches the area for any potential impediments that could be dangerous if overlooked. When the system notices one of these dangers, it immediately sounds a buzzer alarm, alerting the driver to the impending risk. Concurrently, a warning sign that reads ”Drive Slow” appears, acting as a visual cue to be cautious and slow down appropriately. The driver can take proactive steps to reduce the likelihood of an accident by being instantly alerted of any upcoming risks thanks to this synchronised reaction mechanism. The technology improves road safety by integrating visual and audio inputs, creating a responsive and alert driving environment.

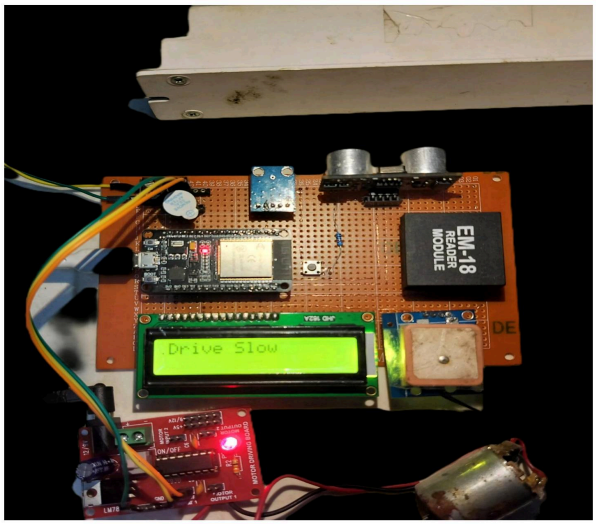

Fig. 8. Alert to Drive Slow

G. Step VII

The system reacts by showing a reassuring message, usually ”Alright,” indicating that the immediate threat has been avoided once the object leaves the ultrasonic sensor’s danger range. Simultaneously, the buzzer that was previously activated is turned off, so stopping its warning sound. This coordinated response guarantees that the motorist gets unambiguous information about how the hazard has been resolved, enabling them to confidently resume regular driving circumstances. The ”Alright” message provides reassurance that everything is safe once more, giving the driver and passengers a feeling of stability and confidence [3].

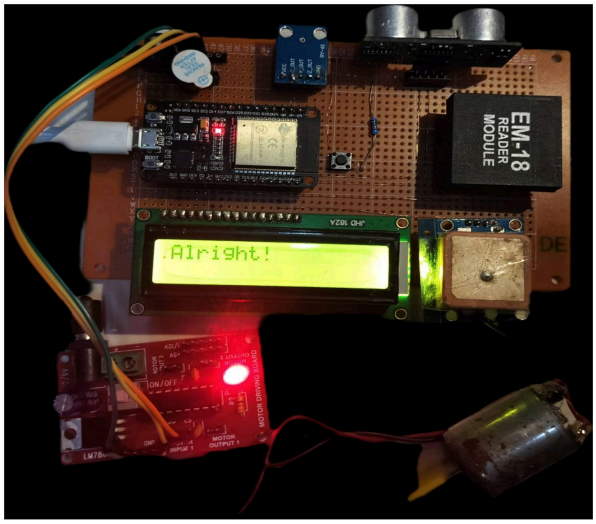

Fig. 9. Safe to Go

H. Step VIII

The accelerometer quickly identifies any abrupt acceleration that exceeds the pre-established threshold for typical conditions when there is a bump along the X or Y axis, regardless of direction. As soon as it detects something, the system prompts the driver to ”Push the Button if Safe.” Concurrently, the buzzer sounds a characteristic ringing sound, alerting people to the alert right away. The driver is prompted by this proactive response mechanism to evaluate the situation and, if it is judged safe, to act by pressing the appropriate button. The technology enables quick decision-making by giving audio cues and explicit directions, enabling the driver to prioritise road safety while successfully responding to unforeseen circumstances.

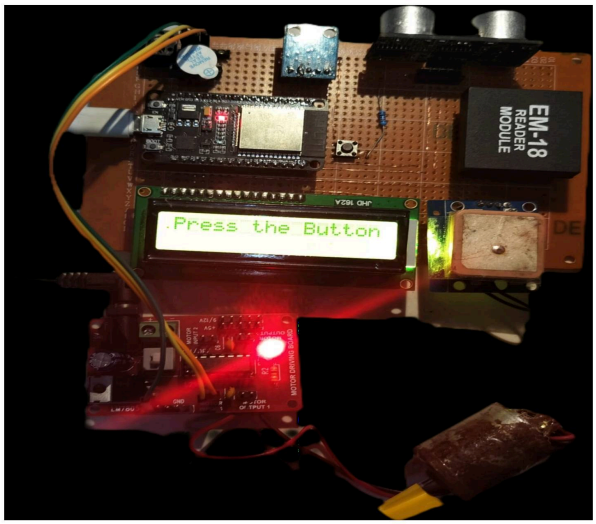

Fig. 10. Press the Button if in Good Condition

I. Step IX

The SOS GPS trackers are automatically activated by the system if the designated button is not hit for ten seconds after the first alert. These trackers pinpoint the user’s exact location at any given time by using satellite positioning systems. They also constantly track the user’s location and status in real time, making it possible to provide help right away if necessary. This seamless GPS integration is a vital safety precaution that gives users and their loved ones peace of mind by facilitating quick reaction and aid in an emergency [4].

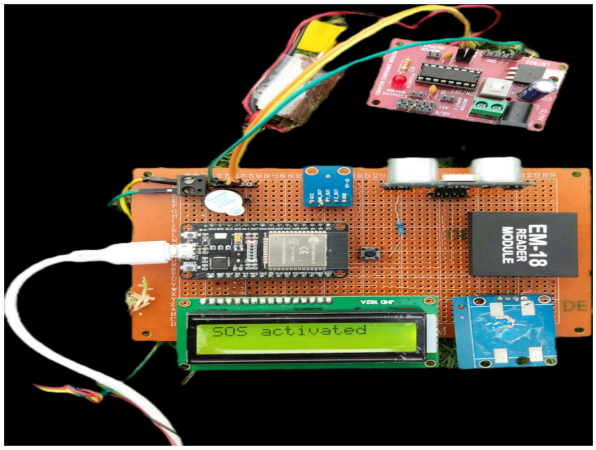

Fig. 11. SOS Activated

J. Step X

Should the user need assistance, an email alert is sent out right away via the SMTP server with all the important information, including the user's location at the accident scene. Responders can easily get the precise coordinates by using the Google Maps link that conveniently contains this location information. At the same time, emergency services are immediately notified, allowing for quick action and the delivery of assistance to the user's location. The user's total safety and security is improved by the smooth integration of email notifications, GPS position sharing through Google Maps, and real-time coordination with emergency services, which guarantees prompt aid and response in an emergency.

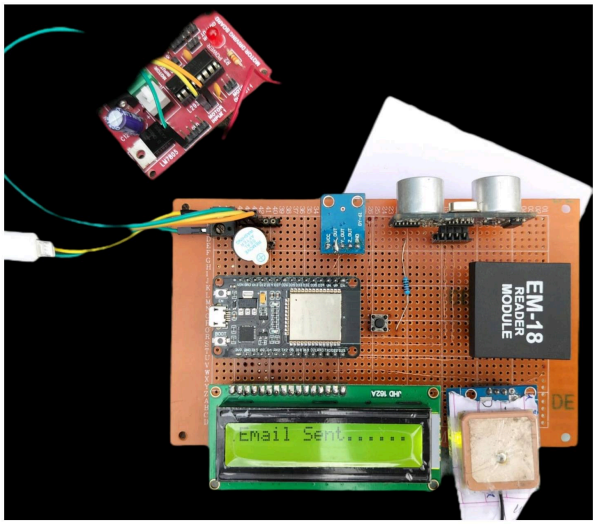

Fig. 12. Email Sent

The email that was sent after the accident is shown in Fig. 12 and includes important details about what happened. The email contains information on the accident, including the time and place, along with a link that allows the user to view their exact location on Google Maps. With the help of this thorough notification, responders will be fully prepared to provide a prompt and efficient reaction. Furthermore, the email is a vital line of contact for arranging emergency services and giving the person in need of aid the help they need.

When the message link is opened using Google Maps, the view in Fig. 13 shows the latitude and longitude coordinates of the accident location. Responders are provided with an accurate and efficient way to navigate to the accident scene thanks to this graphic representation, which shows the actual location of the incident. The user's coordinates are automatically converted into a map display by utilising Google Maps, which offers contextual information to support emergency response activities. By integrating geographic data, rescue operations become more successful and help may be quickly provided to the user who needs it.

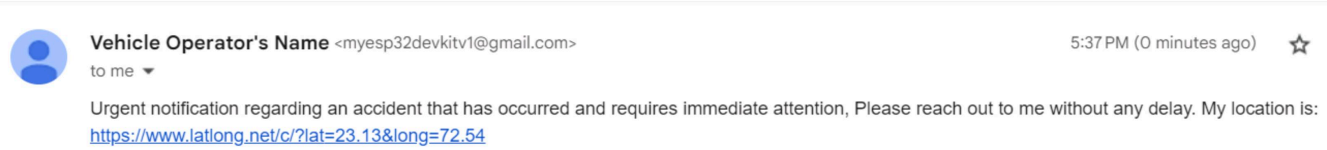

Fig. 13. Notification of mail

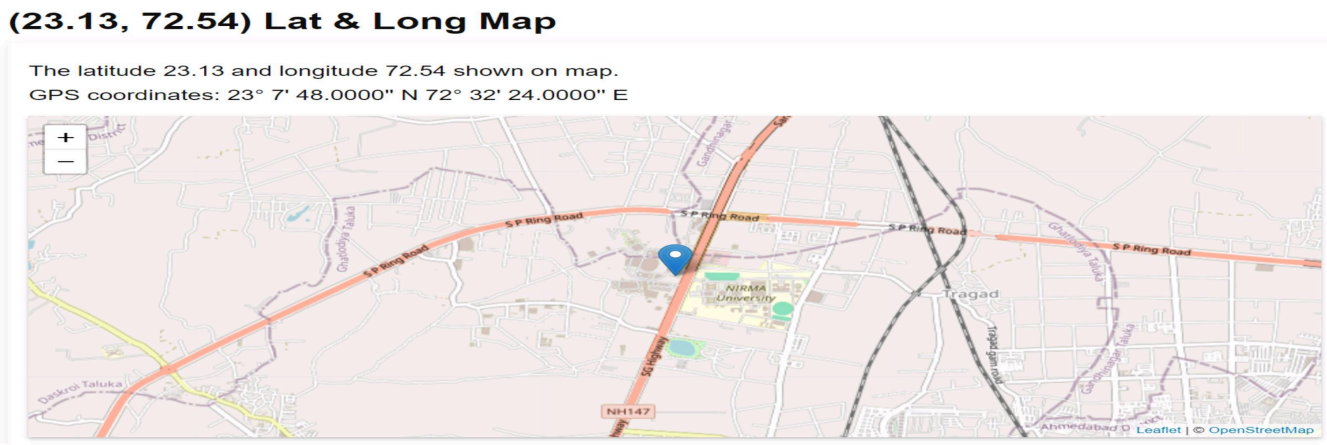

Fig. 14. Location on Map

Hence in the proposed system, Ultrasonic sensor is used for the detection of any sort of obstacles that might come in contact with the vehicle. Accelerometer GY-61 is incorporated for the accurate measurement of acceleration. The acceleration is measured in three axes which are X, Y and Z directions. Positive and negative acceleration are measured by X and Y axes respectively. Along with these sensors the GPS module is incorporated to detect the vehicle location and for sending the notifications to the related people and authorities respectively. The SMTP Server allows to send emails to registered id whenever accident is detected [5].

IV. Conclusion

Every day, tremendous advancements take place all throughout the world. One of the newest and most promising technologies is IOT. A system built with IOT technology can produce results in a way that is more organised and logical. There is an enormous increase in the number of traffic accidents. When an accident happens, the person's life is the most important consideration. The terrifying threat of fatalities often exists due to both a lack of assistance and a delay in learning about the event. To combat these circumstances, IOT-based technology assists in quickly identifying the accident scene and pinpointing the vehicle location. It then promptly notifies surrounding medical facilities and concerned family members of the emergency to prevent and monitor accidents in real life, a more effective system such to this one may be incorporated into automobiles during the production process.

References

[1] Jnana, K.P., Narayan, S. and Gatade, S., 2022. Automatic Vehicle Accident Detection and Healthcare Unit Notification using IoT Technology with ESP32. *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, *10*(4), pp.13-19.

[2] Veeraragaavan, P., Ali, S.A.D., Subaiya, K.A., Vaigundam, V.H. and Mani, M.J., 2020. Accident Alert System And Intimation For Ambulance And Hospital Using Lora. In *international journal of engineering research & technology (ijert) eclectic*.

[3] Gatade, S., Kulkarni, S.V. and Samanvitha, N., 2022, October. Automated Vehicle Accident Detection and Healthcare Unit Alerting Using IoT. In *2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon)* (pp. 1-5). IEEE.

[4] Bhingare, Dipti, et al. "EMERGENCY ALERT SYSTEM FOR VEHICLE USING IOT."

[5] Mansor, Muhammad Naufal, et al. "Application System Development of Accident Prevention and Safety Assistance using IoT Application." Journal of Advanced Research in Applied Sciences and Engineering Technology 31.3 (2023): 265-281.

APPENDIX:

Code:

```
#include <LiquidCrystal.h>    //For 16x2 LCD display
#include <TinyGPS++.h>        //To decode the GPS module data
#include <WiFi.h>             //For WiFi conectivity
#include <ESP_Mail_Client.h>  //For E-Mail SMTP server

//Pin Configuration
#define LCD_RS 15  // Register select pin
#define LCD_EN 2   // Enable pin
#define LCD_D4 4   // Data pin 4
#define LCD_D5 5   // Data pin 5
#define LCD_D6 18  // Data pin 6
#define LCD_D7 19  // Data pin 7
#define Ultrasonic_Echo 21
#define Ultrasonic_Trigger 22
#define Engine_pin 13
#define Buzzer_pin 12
#define Switch 23
#define Accl_X_pin A0
#define Accl_Y_pin A3
// GPS_Rx TxD
// GPS_Tx RxD
// RFID_Rx Tx2
// RFID_Tx Rx2
#define my_RFID_String "10004AF701AC"  //Your RFID code
#define Accl_Cal_Sample 50            //Number of samples for Calibration
#define Max_Accl 500                  //Accalarometer threshold value for accident detection

//-----------For WiFi----------------------//
#define WIFI_SSID "Shreya"          //WiFi name
#define WIFI_PASSWORD "shreya17"  //WiFi password

//---------------For Email-------------------//
```

```
// The smtp host name e.g. smtp.gmail.com for GMail or smtp.office365.com for Outlook or smtp.mail.yahoo.com
#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465

// The sign in credentials
#define AUTHOR_EMAIL "myesp32devkitv1@gmail.com"  //Sende E-Mail address
#define AUTHOR_PASSWORD "qano dkcy munu ikdh"     //Sender application key

// Recipient's email
#define RECIPIENT_EMAIL "dhairyasenghani03@gmail.com"  //Receiver e-Mail address

TinyGPSPlus gps;  // the TinyGPS++ object

SMTPSession smtp;  // Declare the global used SMTPSession object for SMTP transport

LiquidCrystal lcd(LCD_RS, LCD_EN, LCD_D4, LCD_D5, LCD_D6, LCD_D7);  //16x2 LCD oin attache ti library

long Accl_X_Offset;  //X-axis offset value of accarometer
long Accl_Y_Offset;  //Y-axis offset value of accarometer
int Accl_X_value;    //X-axis value of accarometer
int Accl_Y_value;    //Y-axis value of accarometer
unsigned long Accident_time;
double GPS_Latitude;    //Latitude value form GPS at the time of accident
double GPS_Longitude;   //Latitude value form GPS at the time of accident
int Buzzer_status = 1;  //Store the current status of buzzer to prevent the LCD from flickering
void setup() {
  pinMode(Ultrasonic_Echo, INPUT);
  // pinMode(Accl_X_pin, INPUT);
  // pinMode(Accl_Y_pin, INPUT);
  pinMode(Switch, INPUT);
  pinMode(Engine_pin, OUTPUT);
  // pinMode(LCD_RS, OUTPUT);
  // pinMode(LCD_EN, OUTPUT);
  // pinMode(LCD_D4, OUTPUT);
  // pinMode(LCD_D5, OUTPUT);
```

```
  // pinMode(LCD_D6, OUTPUT);
  // pinMode(LCD_D7, OUTPUT);
  pinMode(Ultrasonic_Trigger, OUTPUT);
  pinMode(Buzzer_pin, OUTPUT);

  Serial.begin(9600);   //For GPS
  Serial2.begin(9600);  //For RFID

  // for Accelarometer GY-61
  Accelerometer_Calibration();  //Calculate Accelarometer Offset

  lcd.print("Welcome to ES");
  delay(3000);

  Scan_Licence();  //Scan and validate the licence
  WIFI_Config();   //Initialize and Connect WiFi


  lcd.clear();
  lcd.print("Engine Started");
  digitalWrite(Engine_pin, HIGH);
  delay(3000);
}


void loop() {
  Check_Acclerometer();  //Check Acclerometer Reading and take action if required
  Check_Distance();      //Check distance and take action if required
}



void Scan_Licence() {
  while (Serial2.available() == 0) {
scan_again:
```

```
    lcd.clear();
    delay(500);
    lcd.print("Scan the Licence");
    delay(1000);
  }
  lcd.clear();
  if (Serial2.readString() == my_RFID_String) {
    lcd.print("Successfull");
    delay(2000);
  } else {
    lcd.print("Unsuccessfull");
    delay(2000);
    goto scan_again;
  }
}



void Accelerometer_Calibration() {
  Accl_X_Offset = 0;
  Accl_Y_Offset = 0;
  for (int n = 0; n < Accl_Cal_Sample; n++) {
    Accl_X_Offset = Accl_X_Offset + analogRead(Accl_X_pin);
    Accl_Y_Offset = Accl_Y_Offset + analogRead(Accl_Y_pin);
  }
  Accl_X_Offset = Accl_X_Offset / Accl_Cal_Sample;
  Accl_Y_Offset = Accl_Y_Offset / Accl_Cal_Sample;
}



void Check_Acclerometer() {
  Accl_X_value = analogRead(Accl_X_pin);
  Accl_Y_value = analogRead(Accl_Y_pin);
  // Serial.print("   X=");
  // Serial.print(Accl_X_value - Accl_X_Offset);
```

```
  // Serial.print("   Y=");
  // Serial.println(Accl_Y_value - Accl_Y_Offset);

  if ((Accl_X_value - Accl_X_Offset) > Max_Accl) {
    Accident_Detected();
  } else if ((Accl_X_value - Accl_X_Offset) < -Max_Accl) {
    Accident_Detected();
  } else if ((Accl_Y_value - Accl_Y_Offset) > Max_Accl) {
    Accident_Detected();
  } else if ((Accl_Y_value - Accl_Y_Offset) < -Max_Accl) {
    Accident_Detected();
  }
}

void Check_Distance() {
  digitalWrite(Ultrasonic_Trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(Ultrasonic_Trigger, LOW);
  if (pulseIn(Ultrasonic_Echo, HIGH) < 3000) {
    if (Buzzer_status == 0) {
      digitalWrite(Buzzer_pin, HIGH);
      Buzzer_status = 1;
      lcd.clear();
      lcd.print("Drive Slow");
    }
  } else if (Buzzer_status == 1) {
    digitalWrite(Buzzer_pin, LOW);
    Buzzer_status = 0;
    lcd.clear();
    lcd.print("Alright!");
  }
}

void Accident_Detected() {
  digitalWrite(Engine_pin, LOW);
```

```
  digitalWrite(Buzzer_pin, HIGH);
  lcd.clear();
  Accident_time = millis();
  while ((millis() - Accident_time) < 10000) {
    if (digitalRead(Switch) == 0) {
      lcd.print("Press the Button");
      delay(1000);
      lcd.clear();
      delay(1000);
    } else {
      goto Switch_pressed;
    }
  }
  Send_Aleart();
Switch_pressed:
  {
    lcd.print("Alright!");
    digitalWrite(Engine_pin, HIGH);
    digitalWrite(Buzzer_pin, LOW);
  }
}

void Send_Aleart() {
  digitalWrite(Buzzer_pin, LOW);
  lcd.clear();
  lcd.write("SOS activated");
  // delay(2000);
  Get_Location();
  Send_EMAIL();
  while (1) {
  }
}

void Get_Location() {
  while (1) {
```

```
    if (Serial.available() > 0) {
      if (gps.encode(Serial.read())) {
        if (gps.location.isValid()) {
          GPS_Latitude = gps.location.lat();
          GPS_Longitude = gps.location.lng();
          Serial.print(F("- latitude: "));
          Serial.println(gps.location.lat());

          Serial.print(F("- longitude: "));
          Serial.println(gps.location.lng());
        } else {
          lcd.clear();
          lcd.print("Location Invalid");
          Serial.println(F("- location: INVALID"));
        }
        break;
      }
    }
  }
}



// Callback function to get the Email sending status
void smtpCallback(SMTP_Status status) {
  // Print the current status
  Serial.println(status.info());

  // Print the sending result
  if (status.success()) {
    // ESP_MAIL_PRINTF used in the examples is for format printing via debug Serial port
    // that works for all supported Arduino platform SDKs e.g. AVR, SAMD, ESP32 and ESP8266.
    // In ESP8266 and ESP32, you can use Serial.printf directly.

    Serial.println("----------------");
    ESP_MAIL_PRINTF("Message sent success: %d\n", status.completedCount());
```

```
    ESP_MAIL_PRINTF("Message sent failed: %d\n", status.failedCount());
    Serial.println("----------------\n");

    for (size_t i = 0; i < smtp.sendingResult.size(); i++) {
      // Get the result item
      SMTP_Result result = smtp.sendingResult.getItem(i);

      // In case, ESP32, ESP8266 and SAMD device, the timestamp get from result.timestamp should be valid if
      // your device time was synched with NTP server.
      // Other devices may show invalid timestamp as the device time was not set i.e. it will show Jan 1, 1970.
      // You can call smtp.setSystemTime(xxx) to set device time manually. Where xxx is timestamp (seconds since Jan 1, 1970)

      ESP_MAIL_PRINTF("Message No: %d\n", i + 1);
      ESP_MAIL_PRINTF("Status: %s\n", result.completed ? "success" : "failed");  // my interest
      ESP_MAIL_PRINTF("Date/Time: %s\n", MailClient.Time.getDateTimeString(result.timestamp, "%B %d, %Y %H:%M:%S").c_str());
      ESP_MAIL_PRINTF("Recipient: %s\n", result.recipients.c_str());
      ESP_MAIL_PRINTF("Subject: %s\n", result.subject.c_str());
    }
    Serial.println("----------------\n");

    // You need to clear sending result as the memory usage will grow up.
    smtp.sendingResult.clear();
  }
}



void WIFI_Config() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  lcd.clear();
  lcd.print("Connecting WiFi");
  Serial.print("Connecting WiFi");
  delay(4000);
  while (WiFi.status() != WL_CONNECTED && digitalRead(Switch) == 0) {
    Serial.print(".");
```

```
    delay(300);
  }
  lcd.clear();
  lcd.print("WiFi connected");
  delay(4000);
  // Serial.println();
  // Serial.print("Connected with IP: ");
  // Serial.println(WiFi.localIP());
  // Serial.println();
}



void Send_EMAIL() {
  //  Set the network reconnection option
  MailClient.networkReconnect(true);

  // Enable the debug via Serial port
  //  * 0 for no debugging
  //  * 1 for basic level debugging
  //  *
  //  * Debug port can be changed via ESP_MAIL_DEFAULT_DEBUG_PORT in ESP_Mail_FS.h

  smtp.debug(1);

  // Set the callback function to get the sending results
  smtp.callback(smtpCallback);

  // Declare the Session_Config for user defined session credentials
  Session_Config config;

  // Set the session config
  config.server.host_name = SMTP_HOST;
  config.server.port = SMTP_PORT;
  config.login.email = AUTHOR_EMAIL;
```

```
config.login.password = AUTHOR_PASSWORD;
config.login.user_domain = "";

// Set the NTP config time
// For times east of the Prime Meridian use 0-12
// For times west of the Prime Meridian add 12 to the offset.
// Ex. American/Denver GMT would be -6. 6 + 12 = 18
// See https://en.wikipedia.org/wiki/Time_zone for a list of the GMT/UTC timezone offsets

config.time.ntp_server = F("pool.ntp.org,time.nist.gov");
config.time.gmt_offset = 3;
config.time.day_light_offset = 0;

// Declare the message class
SMTP_Message message;

// Set the message headers
message.sender.name = F("Vehicle Operator's Name");
message.sender.email = AUTHOR_EMAIL;
message.subject = F("Urgent Accident Detection Alert!");
message.addRecipient(("State Authority"), RECIPIENT_EMAIL);
//Urgent Accident Detection Alert!

//Send HTML message
// String htmlMsg = "<div style=\"color:#2f4468;\"><h1>Hello World!</h1><p>- Sent from ESP board</p></div>";
// message.html.content = htmlMsg.c_str();
// message.html.content = htmlMsg.c_str();
// message.text.charSet = "us-ascii";
// message.html.transfer_encoding = Content_Transfer_Encoding::enc_7bit;

//Send raw text message
 String textMsg = "Urgent notification regarding an accident that has occurred and requires immediate attention, Please reach out to me without any delay. My location is: https://www.latlong.net/c/?lat=" + String(GPS_Latitude) + "&long=" + String(GPS_Longitude);

message.text.content = textMsg.c_str();
```