Project report on:

CAR SAFETY SYSTEM USING ESP 32

Submitted by:

ZIG-Bees

Group Members:

1. Shashank Rao BT22ECI031
2. Utkarsh Ghore BT22ECI032

A report submitted for the partial fulfilment of the requirements of the course

ECE 206: Sensor and Transducers

Submission Date: 20/11/2023

Under the guidance of:

Dr. Mayank Thacker

Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर

Indian Institute of Information Technology, Nagpur

Table of Contents:

Chapter 1: Introduction

This project report explores the integration of multiple sensor technologies—specifically, a GPS module, gas sensor, and piezo sensor—with the ESP32 microcontroller. The ESP32, known for its versatility and integrated wireless capabilities, serves as the central processing unit to collect, process, and utilize data from these diverse sensors.

The objective is to create a comprehensive sensor integration system that enables the ESP32 to interface with the GPS module for location data, the gas sensor for environmental monitoring, and the piezo sensor for detecting physical disturbances. The report outlines the methodology for hardware setup, software implementation, and testing/validation of the sensor integration, showcasing the potential applications of sensor fusion in IoT and related fields.

Chapter 2: Components

NEO-6M GPS module

The NEO-6M GPS module by u-blox is a versatile, budget-friendly navigation receiver designed to receive signals from various satellite systems like GPS, GLONASS, Galileo, and BeiDou. It offers precise positioning data, typically within a few meters, through serial communication. This module is widely used in projects requiring accurate location information, such as vehicle tracking, drones, and IoT devices. Its low power consumption and ease of interfacing with microcontrollers like Arduino make it a popular choice among hobbyists and professionals for diverse navigation applications. Configuration using AT commands allows for customization, while clear sky visibility enhances its performance.

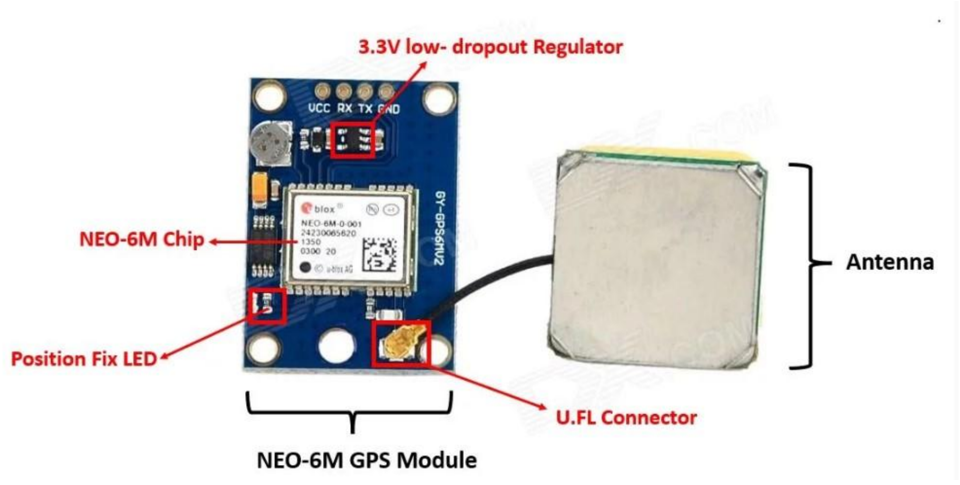


Fig – [1]   NEO-6M GPS Module

MQ2 Gas Sensor

The MQ-2 gas sensor is a small and affordable device widely used for detecting various gases in the air, particularly combustible gases like methane, propane, butane, and smoke. It operates based on the change in resistance of its sensitive material when exposed to different gases. The sensor provides analog output, where the level of gas concentration alters the voltage output. It is commonly used in gas leak detection systems, fire detection systems, and air quality monitoring devices due to its simplicity, cost-effectiveness, and ability to sense a range of gases. However, precise calibration and understanding its limitations are crucial for accurate gas detection.

Fig- [2] - MQ2 Gas Sensor

Piezoelectric vibration sensor

A piezoelectric vibration sensor is a device that converts mechanical vibrations or impacts into electrical signals. It consists of a piezoelectric material (like quartz or certain ceramics) sandwiched between electrodes. When subjected to mechanical stress or vibration, the piezoelectric material generates an electric charge proportional to the applied force or vibration intensity. This generated voltage or charge is then measured and interpreted as an indication of the presence, intensity, or frequency of the mechanical vibrations. Piezoelectric vibration sensors are commonly used in various applications such as industrial machinery monitoring, impact detection, seismic activity measurement, and musical instruments for their sensitivity, durability, and ability to convert mechanical energy into electrical signals.

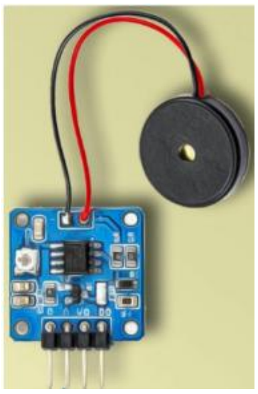


Fig. [3] - Piezoelectric Vibration Sensor

ESP32

ESP32 is a powerful,low-cost, and highly integrated microcontroller developed by Espressif Systems. It also has built-in Wi- Fi and Bluetooth connectivity, making vit idealfor IoT applications. TheWi-Fi connectivity supports both 2.4GHz and 5GHz frequency bands, while the Bluetooth connectivity supports classic Bluetooth and Bluetooth Low Energy (BLE) protocols [4]. The ESP32 microcontroller comes with a range of interfaces, including SPI, I2C, UART, I2S, PWM, and ADC. It also has a built-in hall sensor, temperature sensor, and touch sensor, making it suitable for a wide range of applications [4]. The ESP32 microcontroller supports several programming languages, including C, C++, and MicroPython. It also has an integrated development environment (IDE) called theESP-IDF, which provides a set ofsoftware development toolsfor developing applications.

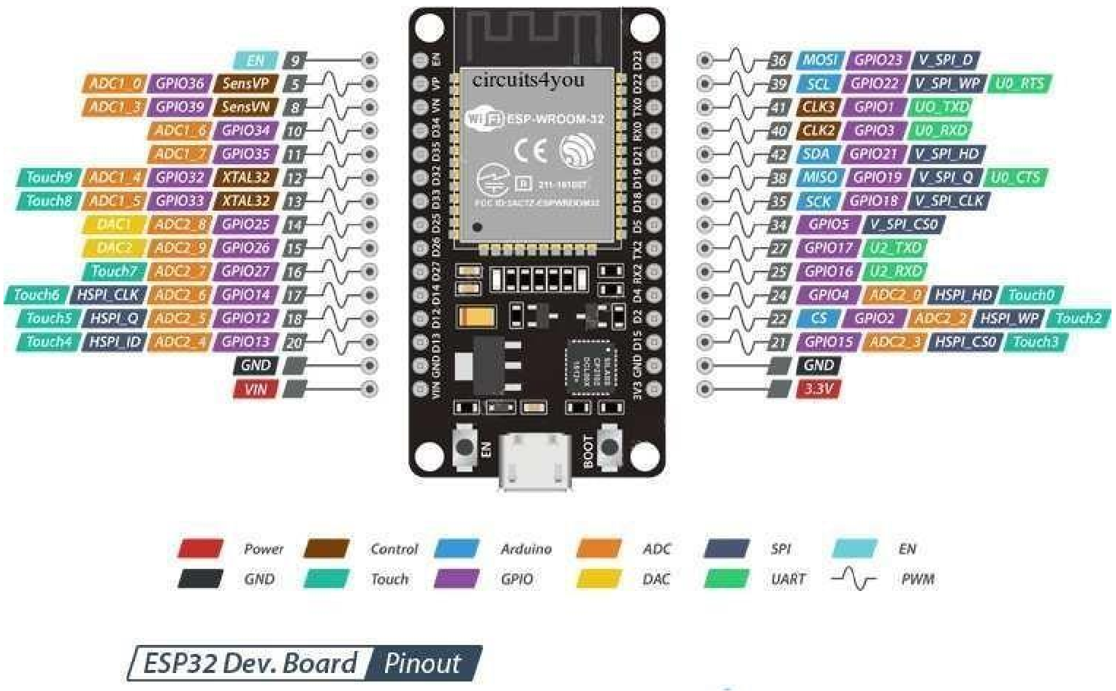


Fig - [4] ESP32 pinouts

Chapter 3: Working[3]

System Workflow:

Hardware Setup:

- In addition to the NEO-6M GPS sensor connected to the ESP32, integrate an MQ2 gas sensor and a piezoelectric vibration sensor.
- Connect the MQ2 gas sensor and piezoelectric vibration sensor to available GPIO pins on the ESP32.

Software Implementation Update:

- Extend the Arduino sketch to include functionality for reading data from the MQ2 gas sensor and piezoelectric vibration sensor.
- Utilize relevant libraries for the sensors to retrieve data regarding gas levels and vibration events.

ESP32 Setup Update:

- Modify the code to accommodate the setup and configuration of the MQ2 gas sensor and the piezoelectric vibration sensor alongside the GPS sensor.
- Implement logic to handle data retrieval from these sensors.

Main Loop Execution Update:

- Alongside continuous GPS data retrieval, include routines to collect data from the MQ2 gas sensor and piezoelectric vibration sensor within the main loop.
- Format the collected data from all sensors into a publishable format (e.g., CSV or JSON) for MQTT publication.

MQTT Publication Update:

- Enhance the MQTT publication mechanism to include data from the newly integrated sensors (MQ2 gas sensor and piezoelectric vibration sensor).
- Publish the formatted data containing information from all sensors to separate designated MQTT topics on Adafruit IO.

Error Handling and Connectivity Check Update:

- Extend error handling mechanisms to manage issues related to the newly integrated sensors, including reconnecting in case of sensor disconnections or data transmission problems.

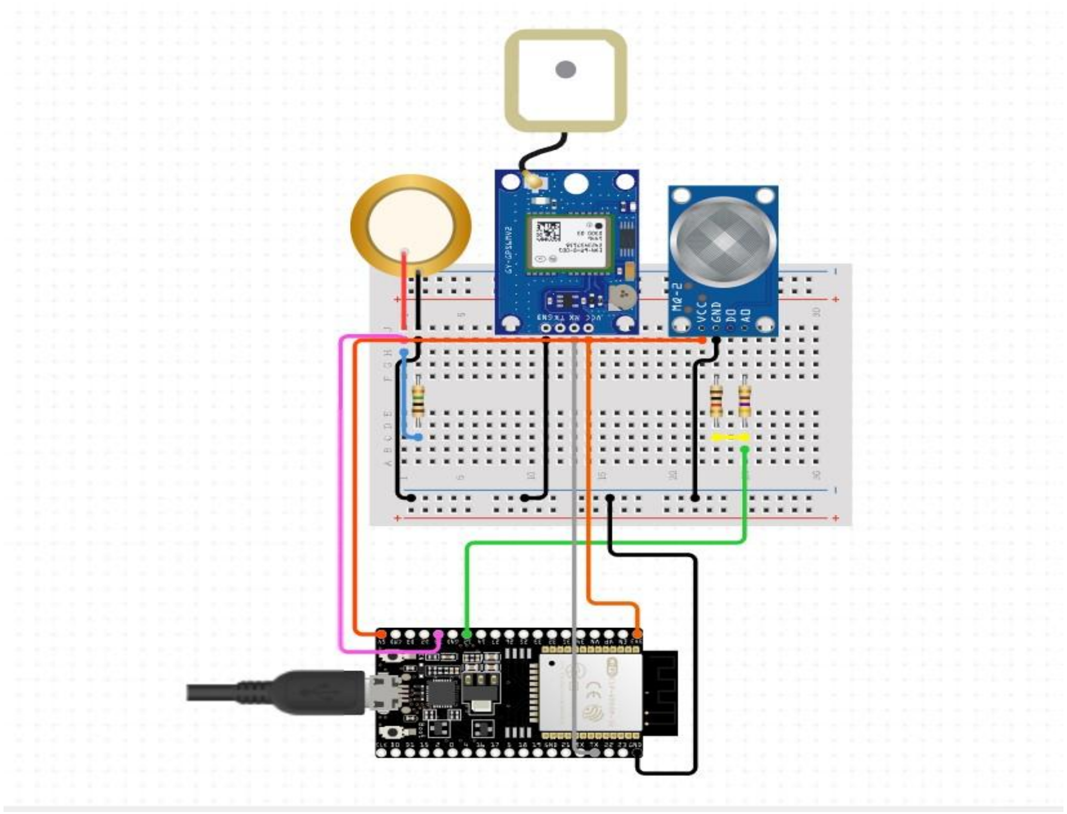
Overall System Behavior Update:

- The ESP32 continuously reads data from the GPS sensor, MQ2 gas sensor, and piezoelectric vibration sensor.

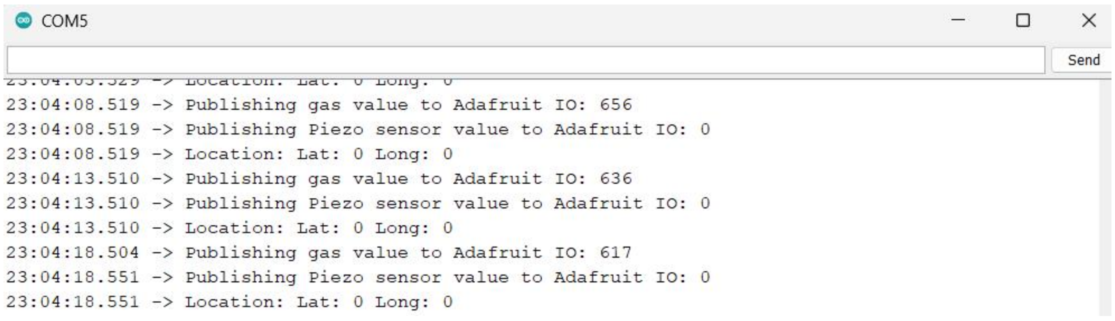
- It securely transmits all collected sensor data (GPS location, gas levels, vibration events) to Adafruit IO via MQTT over Wi-Fi.
- Users can visualize and manage all sensor data in real-time through Adafruit IO's dashboard interface, allowing monitoring and analysis of multiple data streams simultaneously.

This system architecture incorporates the functionality of the MQ2 gas sensor and piezoelectric vibration sensor alongside the existing GPS sensor, providing a comprehensive platform for real-time tracking and visualization of multiple data streams.
These data streams are used to build a device that can be used to enhance car safety. The GPS data helps keep a track of the location of the car. Meanwhile, the gas sensor checks for any possible leaks or presence of other harmful gases and the piezoelectric vibration sensor can detect car crash. This creates a safety device for crashes and gas leaks with the facility of live GPS location. Thus when such an emergency arrives, Emergency response units and families of the affected can be informed along with the live location.

Circuit diagram[7]

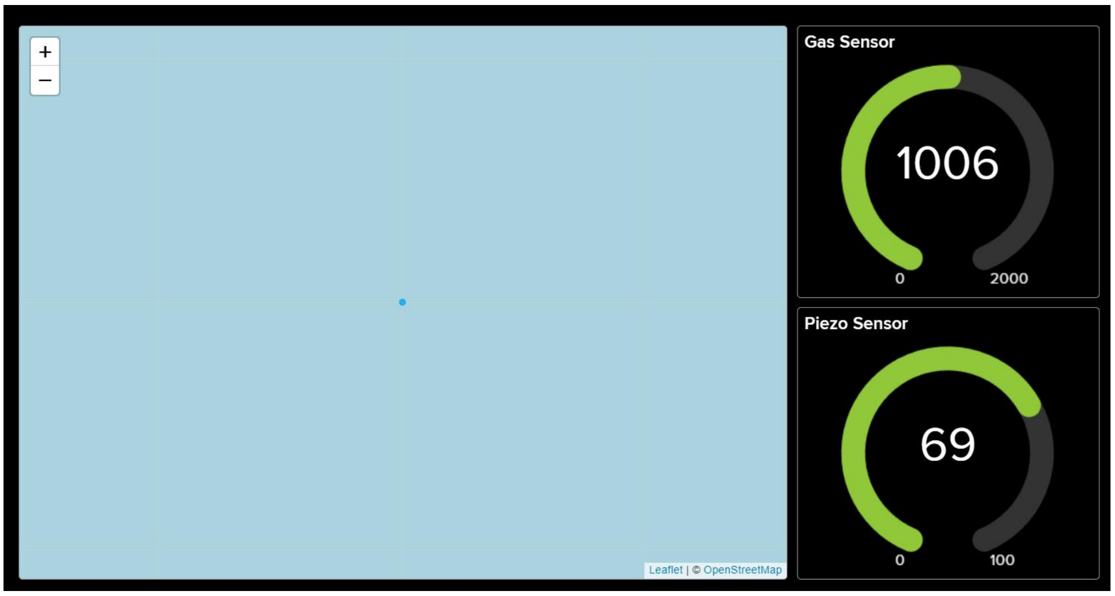


Output:



```
23:04:08.519 -> Publishing gas value to Adafruit IO: 656
23:04:08.519 -> Publishing Piezo sensor value to Adafruit IO: 0
23:04:08.519 -> Location: Lat: 0 Long: 0
23:04:13.510 -> Publishing gas value to Adafruit IO: 636
23:04:13.510 -> Publishing Piezo sensor value to Adafruit IO: 0
23:04:13.510 -> Location: Lat: 0 Long: 0
23:04:18.504 -> Publishing gas value to Adafruit IO: 617
23:04:18.551 -> Publishing Piezo sensor value to Adafruit IO: 0
23:04:18.551 -> Location: Lat: 0 Long: 0
```

Serial Monitor Output



[2]Adafruit IO dashboard output

Chapter 4: Code

```
#include <WiFi.h>

#include <WiFiAP.h>

#include <WiFiClient.h>

#include <WiFiGeneric.h>

#include <WiFiMulti.h>

#include <WiFiSTA.h>

#include <WiFiScan.h>

#include <WiFiServer.h>

#include <WiFiType.h>

#include <WiFiUdp.h>


#include <Wire.h>

#include <TinyGPS++.h>

#include <Adafruit_MQTT.h>

#include <Adafruit_MQTT_Client.h>


#define GAS_SENSOR_PIN 32

#define BUZZER_PIN 2

#define Piezo_SENSOR_PIN 33  // Pin to which the Piezo sensor is connected


TinyGPSPlus gps;


bool gasDetected = false;

#define WLAN_SSID       "Acer"

#define WLAN_PASS       "a123456b"
```

```
// Adafruit IO Setup
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "flamerider12"
#define AIO_KEY         "aio_ZbDC53ENPhEIwPrRxTZlT3f90V5r"

WiFiClient client;
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);

// Adafruit IO Feeds
Adafruit_MQTT_Publish gasFeed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/feeds/gas");
Adafruit_MQTT_Publish PiezoFeed = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/piezo");
void MQTT_connect();

void setup() {
  Serial.begin(115200);
  pinMode(GAS_SENSOR_PIN, INPUT);
  pinMode(BUZZER_PIN, OUTPUT);
  pinMode(Piezo_SENSOR_PIN, INPUT);
  Serial.println(); Serial.println();
  Serial.print("Connecting to ");
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
```

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

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

void loop() {
 // Read gas sensor value
 MQTT_connect();
 int gasValue = analogRead(GAS_SENSOR_PIN);
 publishGasValue(gasValue);
 int PiezoValue = analogRead(Piezo_SENSOR_PIN);
  // Publish Piezo sensor value to Adafruit IO
 publishPiezoValue(PiezoValue);
  if(gasValue>2000){
   buzzerAlert();
  }

  readGPSLocation();
  printGPSLocation();

  delay(5000);
```

```
//  float latitude = gps.location.lat();
//   float longitude = gps.location.lng();
//   Serial.print("Latitude:  ");
//   Serial.println(latitude, 6);
//   Serial.print("Longitude: ");
//   Serial.println(longitude, 6);
}

void buzzerAlert() {
  digitalWrite(BUZZER_PIN, HIGH); // Turn the buzzer on
  delay(500); // Buzz for 1 seconds
  digitalWrite(BUZZER_PIN, LOW); // Turn the buzzer off
}

bool readGPSLocation() {
  while (Serial.available() > 0) {
    if (gps.encode(Serial.read())) {
      return true;
    }
  }
  return false;
}

void printGPSLocation() {
  Serial.print("Location: ");
  if (gps.location.isValid()) {
    Serial.print("Lat: ");
    Serial.print(gps.location.lat(), 6);
    Serial.print(", Long: ");
    Serial.print(gps.location.lng(), 6);
  } else {
    Serial.print("Lat: 0 ");
    Serial.print("Long: 0");
  }
  Serial.println();
}

void publishGasValue(int value) {
  Serial.print("Publishing gas value to Adafruit IO: ");
  Serial.println(value);
  gasFeed.publish(value);
}

void publishPiezoValue(int value) {
  Serial.print("Publishing Piezo sensor value to Adafruit IO: ");
  Serial.println(value);
  PiezoFeed.publish(value);
}

void MQTT_connect() {
  int8_t ret;
  // Stop if already connected.
  if (mqtt.connected()) {
    return;
  }

  Serial.print("Connecting to MQTT... ");

  uint8_t retries = 3;
  while ((ret = mqtt.connect()) != 0) { // connect will return 0 for connected
       Serial.println(mqtt.connectErrorString(ret));
       Serial.println("Retrying MQTT connection in 5 seconds...");
       mqtt.disconnect();
       delay(5000);  // wait 5 seconds
       retries--;
       if (retries == 0) {
         // basically die and wait for WDT to reset me
         while (1);
       }
  }
  Serial.println("MQTT Connected!");
}
```

Chapter 5: References

[1]https://microcontrollerslab.com/neo-6m-gps-module-arduino-tutorial/
[2]https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/
[3]https://en.wikipedia.org/wiki/Piezoelectric_sensor
[4]https://io.adafruit.com
[5]https://chat.openai
[6]https://unsplash.com/s
[7]https://www.circuito.io/