VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Name: Ananya Pillai

Course Name: Embedded Systems and IOT LAB

Course Code: BITE403P

<u>Topic</u>: ACCIDENT DETECTION SYSTEM

Abstract:

An IoT-based accident detection system is a cutting-edge solution that leverages the power of Internet of Things (IoT) technology to detect and respond to accidents or collisions in real-time. This system integrates smart sensors, communication modules, and cloud-based platforms to provide efficient and automated accident detection and reporting capabilities.

The primary objective of an IoT-based accident detection system is to enhance safety and minimize response times in emergency situations. The system utilizes a network of interconnected sensors, such as vibration sensor, strategically placed in vehicles, infrastructure, or personal devices. These sensors continuously monitor various environmental and motion-related parameters.

When an accident or collision occurs, the sensors capture and analyse the relevant data, such as impact forces, changes in motion patterns, or environmental conditions. The processed data is then transmitted to a central hub or cloud-based platform through wireless communication protocols, such as Wi-Fi, Bluetooth, or cellular networks.

In the cloud-based platform, advanced algorithms and machine learning techniques are employed to analyse the incoming data and determine the occurrence of an accident. Once an accident is detected, the system can trigger a range of automated actions, including generating alerts, notifying emergency services, and providing real-time updates to concerned individuals or authorities.

Moreover, an IoT-based accident detection system can also integrate with existing infrastructure or emergency response systems, allowing seamless coordination between various stakeholders. The system can provide accurate location information, route optimization for emergency vehicles, and even collect post-accident data for forensic analysis and insurance purposes.

Overall, an IoT-based accident detection system offers a sophisticated and proactive approach to accident prevention, detection, and response. By leveraging IoT technology, it enables faster emergency response, enhances safety measures, and potentially saves lives in critical situations.

Problem Definition:

The problem addressed by an accident detection system is the need for timely and accurate detection of accidents or collisions in various environments, such as vehicles, workplaces, or public spaces. Accidents can have severe consequences, including injuries, property damage, and loss of life. Therefore, it is crucial to have an effective system in place that can quickly identify accidents and initiate appropriate responses to minimize harm and provide timely assistance.

Traditional accident detection methods often rely on human observation or manual reporting, which can be slow, subjective, and prone to errors. Additionally, in certain situations, such as when the accident occurs in a remote or unattended location, immediate assistance may not be readily available.

The problem is to develop a reliable and automated accident detection system that utilizes advanced technologies, such as IoT, sensor networks, and data analytics, to accurately detect accidents in real-time. The system should be capable of continuously monitoring the environment, analysing sensor data, and promptly identifying the occurrence of accidents based on predefined criteria or abnormal patterns. It should also have the ability to communicate the detected accidents to relevant parties, such as emergency services or authorized personnel, to initiate appropriate actions or assistance.

The challenge lies in designing a robust and scalable system that can effectively handle different types of accidents, adapt to varying environmental conditions, and operate in diverse settings. The system should minimize false positives and false negatives, ensuring accurate accident detection while avoiding unnecessary alarms or notifications. Furthermore, considerations should be given to power efficiency, connectivity, data security, and integration with existing infrastructure or emergency response systems.

By addressing these challenges, an efficient accident detection system can significantly improve response times, reduce the impact of accidents, and enhance overall safety in various domains, ultimately saving lives and minimizing damage.

Objective:

The objective of an accident detection system is to develop a reliable, real-time, and automated solution that can accurately detect accidents or collisions in different environments. The system aims to achieve the following objectives:

Timely Accident Detection: The system should promptly identify the occurrence of accidents or collisions as soon as they happen, ensuring a quick response and minimizing the response time to provide necessary assistance and support.

Accuracy and Reliability: The system should accurately distinguish between actual accidents and normal environmental variations, minimizing false positives and false negatives to ensure reliable accident detection.

Real-time Monitoring: The system should continuously monitor the environment using sensors or data sources, enabling real-time detection and response to accidents.

Scalability and Adaptability: The system should be designed to handle various types of accidents in different settings, accommodating scalability and adaptability to suit different applications and environments.

Seamless Integration: The system should seamlessly integrate with existing infrastructure, emergency response systems, or communication networks, enabling efficient coordination and communication between stakeholders.

Notification and Alerting: Once an accident is detected, the system should generate appropriate notifications or alerts to inform relevant parties, such as emergency services, authorized personnel, or individuals in the vicinity, facilitating timely assistance and intervention.

Data Analytics and Insights: The system should have the capability to collect and analyse accident data, providing valuable insights for post-accident analysis, preventive measures, and potential improvements in safety protocols.

Cost-effectiveness: The system should be developed with a consideration for cost-effectiveness, utilizing appropriate hardware, software, and communication technologies without compromising on performance and reliability.

By achieving these objectives, an accident detection system aims to enhance safety, minimize the impact of accidents, and improve emergency response.

- Early Accident Detection
- Reliable Accident Identification
- Notification to Emergency Services
- Timely Assistance
- Accurate Location Identification
- Post-Accident Support
- Data Collection for Analysis
- Reduction in Response Time

Components Used:

- ❖ Breadboard Small
- ❖ GPS Module
- ❖ Buzzer
- ❖ Vibration sensor
- ❖ Node MCU [ESP-8266]

Implementation:

An implementation for an accident detection system needs a detailed understanding of the specific requirements like hardware components, and technologies to be used. Implementations can vary based on the chosen sensors, communication protocols, and the platform used for data processing and analysis.

Hardware Setup:

- ❖ Choose appropriate sensors based on the application, such as vibration sensor, accelerometers, gyroscopes, proximity sensors, or cameras.
- ❖ Connect the sensors to the microcontroller or development board, ensuring proper wiring and compatibility.
- ❖ Include additional hardware components like power supply, wireless communication modules (e.g., Wi-Fi, Bluetooth, or cellular), and gps

module or any required peripherals.

Sensor Data Acquisition:

- ❖ Read data from the sensors using the microcontroller or development board's analog or digital input pins.
- ❖ Process the raw sensor data, applying filtering or calibration techniques if necessary.
- ❖ Convert analog sensor readings into digital values, if applicable.

Data Processing and Analysis:

- ❖ Implement algorithms to analyze sensor data for accident detection.
- ❖ Set thresholds or criteria to differentiate normal activities from potential accidents.

Real-Time Monitoring and Decision Making:

- ❖ Continuously monitor the sensor data in real-time.
- ❖ Apply the accident detection algorithm to determine if an accident has occurred.
- ❖ Make decisions based on the detection result, such as triggering alerts or notifications.

Communication and Alerting:

- ❖ Establish communication with the network or cloud platform.
- ❖ Send accident notifications or alerts to predefined recipients, such as emergency services, authorized personnel, or mobile applications.
- ❖ Utilize appropriate communication protocols, such as MQTT, HTTP, or custom APIs.

Testing, Validation, and Deployment:

- ❖ Conduct rigorous testing of the system under different scenarios to ensure accurate accident detection and reliable performance.
- ❖ Validate the system's effectiveness and responsiveness through simulations or real-world experiments.

❖ Deploy the system in the desired environment, considering factors like power supply, network connectivity, and environmental conditions.

Code:

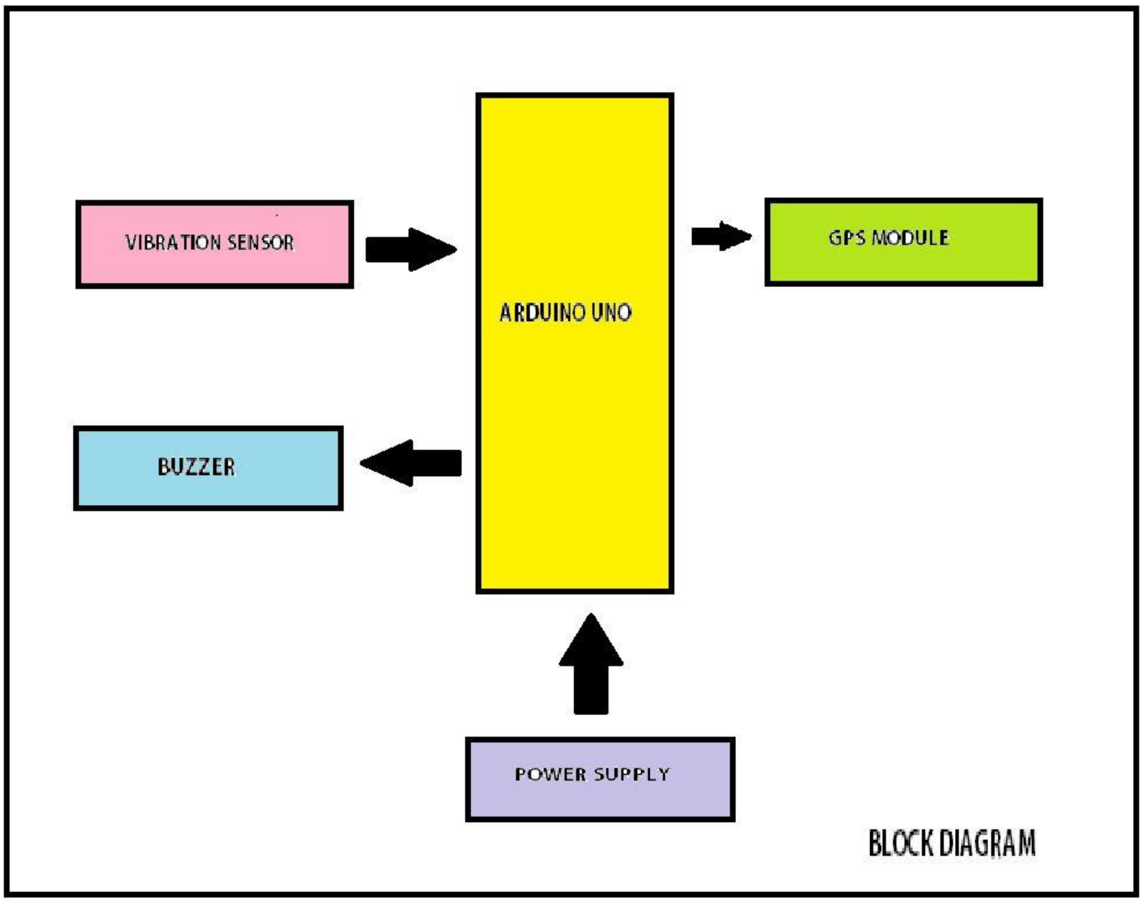
```
#include<TinyGPS++.h>
#include <SoftwareSerial.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <UrlEncode.h>


int temp= 0;
int vib_pin = D1;
int flag = 0;
int buzzer = D2;
int counter;

SoftwareSerial GPS_SoftSerial(D4, D3);

TinyGPSPlus gps;

const char* ssid = "Wifi enduku ra";
const char* password = "check123";

String phoneNumber = "+919700662454";
String apiKey = "5371501";

void sendMessage(String message)
{
  String url = "http://api.callmebot.com/whatsapp.php?phone=" +
phoneNumber + "&apikey=" + apiKey + "&text=" + urlEncode(message);
  WiFiClient client;
  HTTPClient http;
  http.begin(client, url);
  http.addHeader("Content-Type", "application/x-www-form-urlencoded");

  int httpResponseCode = http.POST(url);
```

```
 if (httpResponseCode == 200)
 {
   Serial.print("Message sent successfully");
 }
 else
 {
   Serial.println("Error sending the message");
   Serial.print("HTTP response code: ");
   Serial.println(httpResponseCode);
 }
 http.end();
}

void setup()
{
 pinMode(vib_pin,INPUT);
 pinMode(buzzer,OUTPUT);
 Serial.begin(9600);
 Serial.println("start");
 GPS_SoftSerial.begin(9600);
 Serial.println("end");
 WiFi.begin(ssid, password);
 Serial.print("Connecting to Wifi");
 while(WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 Serial.println("");
 Serial.print("Connected to WiFi network with IP Address: ");
 Serial.println(WiFi.localIP());
}

void loop()
{
 int val;
 val=digitalRead(vib_pin);
 if(digitalRead(vib_pin) == 1)
 {
   Serial.println("Collision Detected !! Fetching location...");
   digitalWrite(buzzer,HIGH);
```

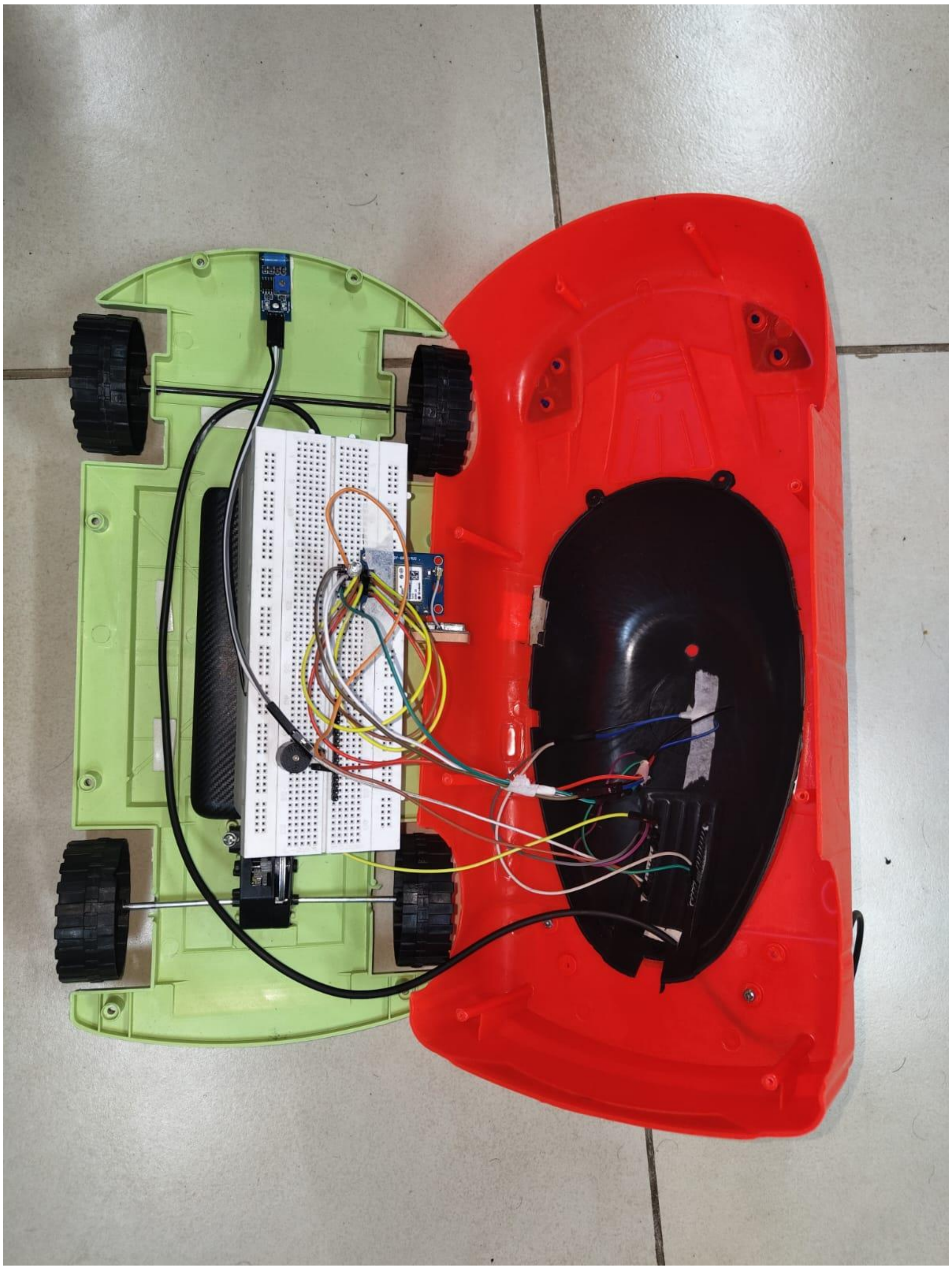
```
  while(flag == 0)
  {
    unsigned long start = millis();
    do
    {
      while (GPS_SoftSerial.available())
        gps.encode(GPS_SoftSerial.read());
    } while (millis() - start < 1000);

    double lat_val, lng_val, alt_m_val;
    uint8_t hr_val, min_val, sec_val;
    bool loc_valid, alt_valid, time_valid;
    lat_val = gps.location.lat();
    loc_valid = gps.location.isValid();
    lng_val = gps.location.lng();

    if (!loc_valid)
    {
      Serial.println("Location not identified yet...");
      delay(1000);
    }
    else
    {
      flag = 1;
      Serial.println("Accident detected !!");
      Serial.print("Latitude in Decimal Degrees : ");
      Serial.println(lat_val, 6);
      Serial.print("Longitude in Decimal Degrees : ");
      Serial.println(lng_val, 6);
      Serial.println("Press Reset button in under 10 seconds to cancel sending
message...");
      counter = 10;
      digitalWrite(buzzer,HIGH);
      while (counter > 0)
      {
        Serial.println(counter);
        delay(1000);
        counter = counter-1;
      }
      digitalWrite(buzzer,LOW);
```

```
        Serial.print("Sending message....");
        sendMessage("Accident Detected !! Location:
https://maps.google.com/?q="+String(lat_val,6)+"," + String(lng_val,6));
      }
    }
    flag = 0;
  }
}
```

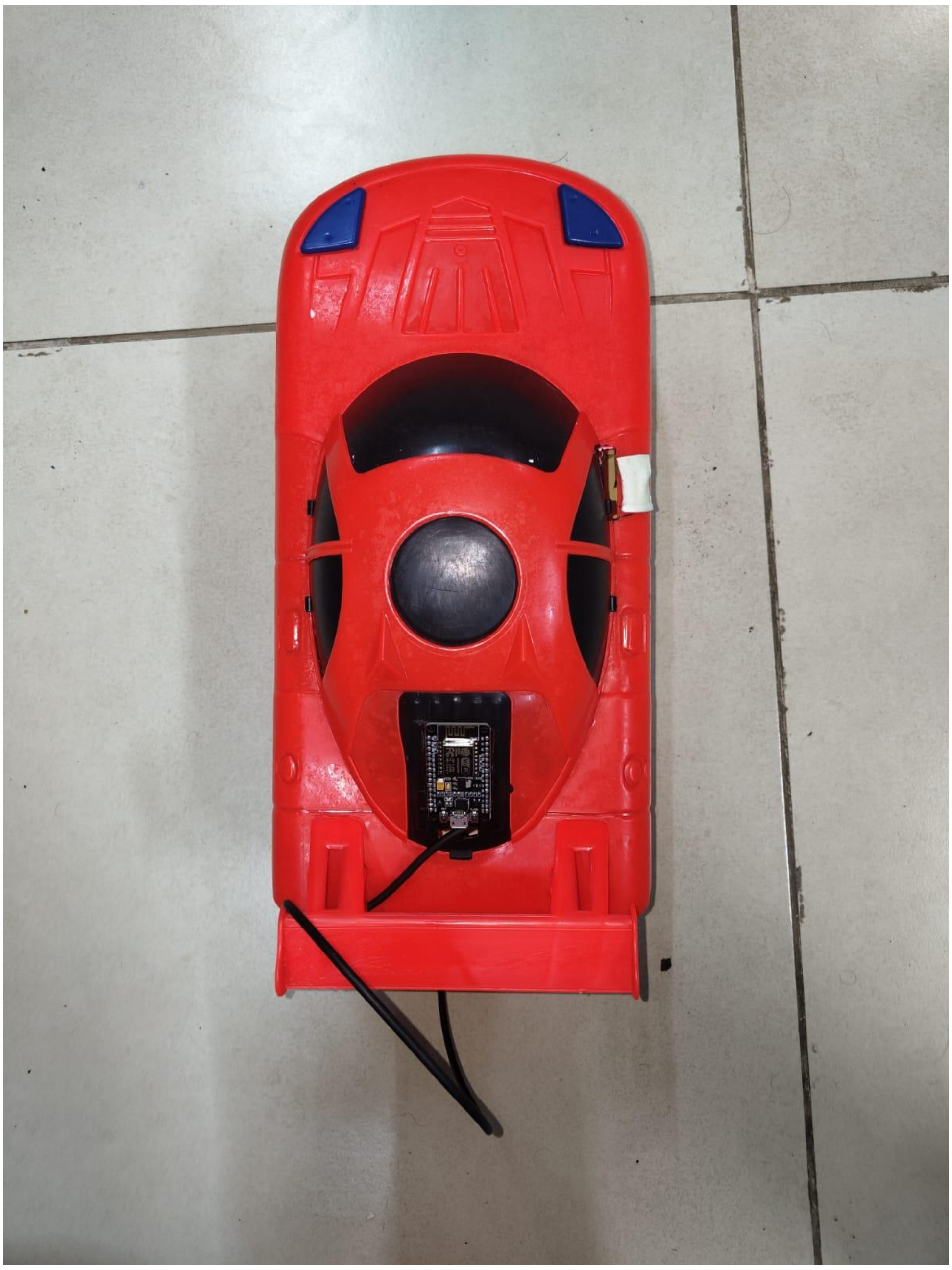
Block Diagram:

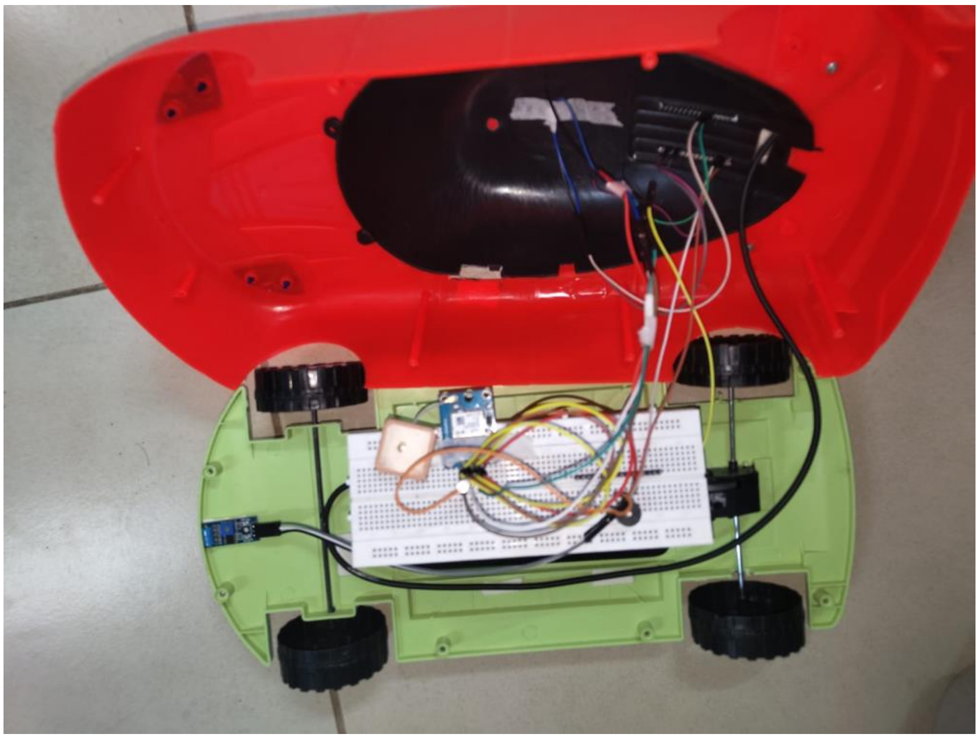
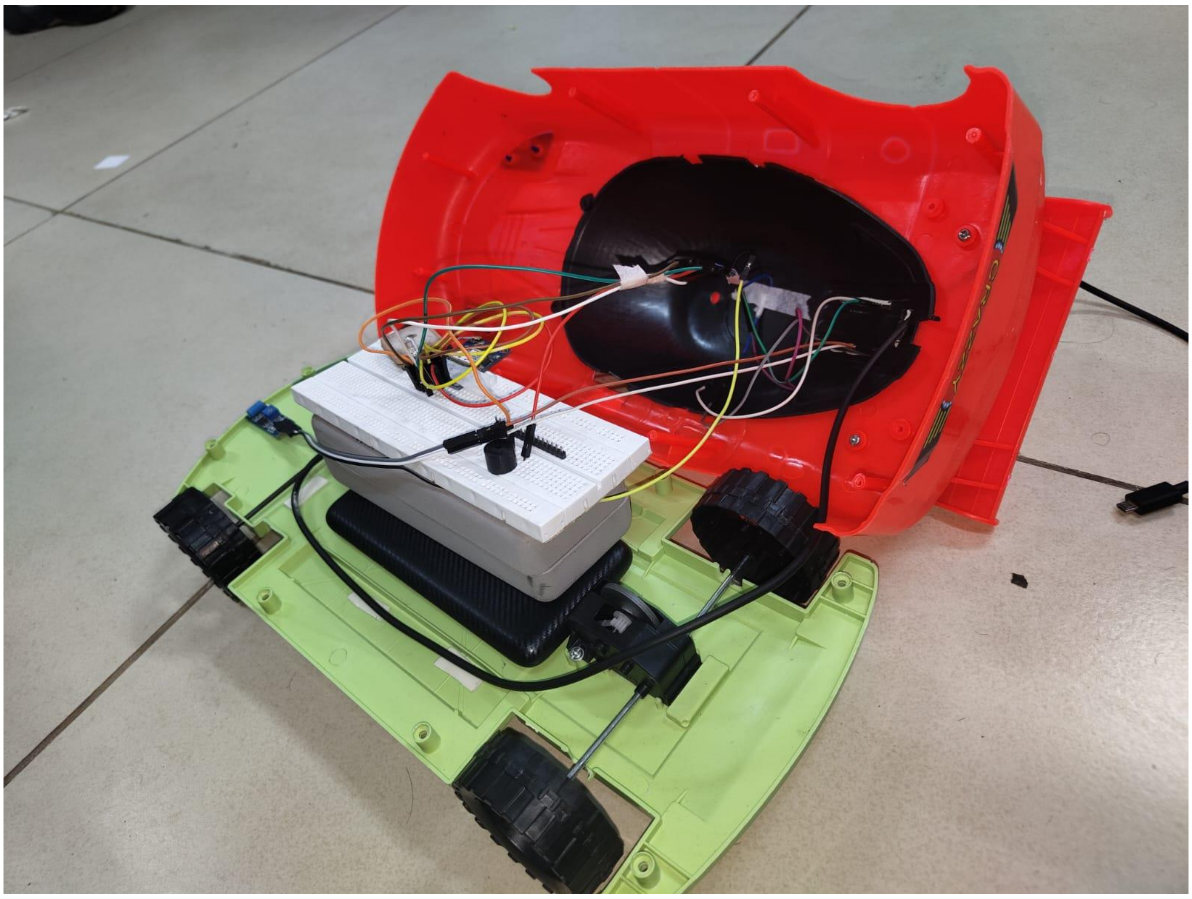


Circuit:

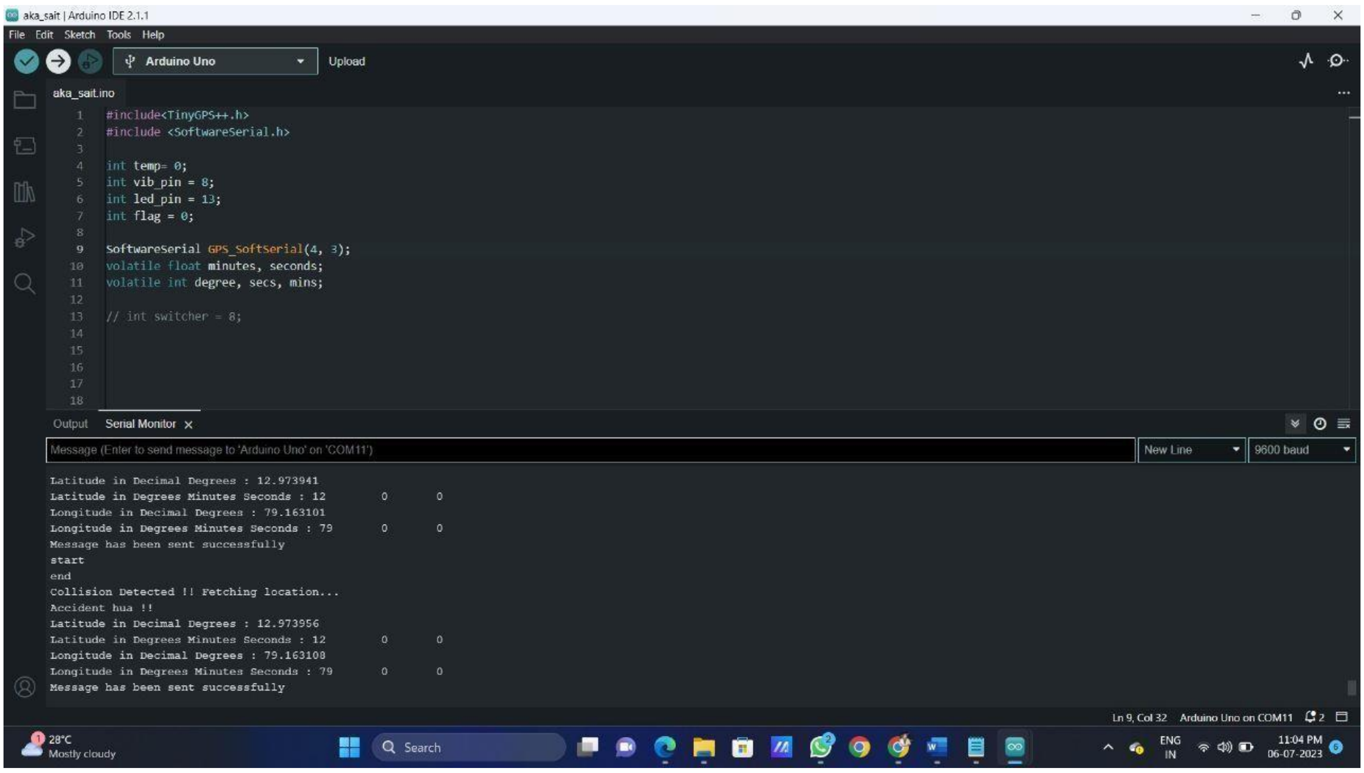


Model Pic:



Conclusion:


aka_sait | Arduino IDE 2.1.1
File Edit Sketch Tools Help
Arduino Uno
Upload
aka_sait.ino
Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'COM11')
New Line
9600 baud
Ln 9, Col 32 Arduino Uno on COM11
28°C
Mostly cloudy
Search
ENG IN
11:04 PM
06-07-2023

```
#include<TinyGPS++.h>
#include <SoftwareSerial.h>

int temp= 0;
int vib_pin = 8;
int led_pin = 13;
int flag = 0;

SoftwareSerial GPS_SoftSerial(4, 3);
volatile float minutes, seconds;
volatile int degree, secs, mins;

// int switcher = 8;
```

```
Latitude in Decimal Degrees : 12.973941
Latitude in Degrees Minutes Seconds : 12	0	0
Longitude in Decimal Degrees : 79.163101
Longitude in Degrees Minutes Seconds : 79	0	0
Message has been sent successfully
start
end
Collision Detected !! Fetching location...
Accident hua !!
Latitude in Decimal Degrees : 12.973956
Latitude in Degrees Minutes Seconds : 12	0	0
Longitude in Decimal Degrees : 79.163108
Longitude in Degrees Minutes Seconds : 79	0	0
Message has been sent successfully
```

Thank you !