PROJECT TITLE: PUBLIC TRANSPORT OPTIMIZATION

PHASE 5: PROJECT DOCUMENTATION AND SUBMISSION

OBJECTIVE:

The primary objective of this project is to create a smart transportation system that utilizes Arduino, GSM, and GPS technologies to enable efficient vehicle tracking and monitoring. The system aims to provide real-time location data and communication capabilities for vehicles, which can be applied in various scenarios, including fleet management, logistics, and personal vehicle tracking.

IOT SENSOR DEPLOYMENT:

- **WOKWI:**

Wokwi is a versatile online platform that allows you to design, simulate, and test electronic circuits in a virtual environment.

Website: (https://wokwi.com)

WIRING CONNECTIONS:

1. **Arduino to GPS Module:**
   - Connect GPS module TX (transmit) pin to Arduino RX (receive) pin.
   - Connect GPS module RX (receive) pin to Arduino TX (transmit) pin.
   - Connect the GPS module's VCC (power) and GND (ground) to the appropriate

     Arduino pins.

2. **Arduino to GSM Module:**
   - Connect GSM module TX (transmit) pin to Arduino RX (receive) pin.
   - Connect GSM module RX (receive) pin to Arduino TX (transmit) pin.
   - Connect the GSM module's VCC (power) and GND (ground) to the appropriate

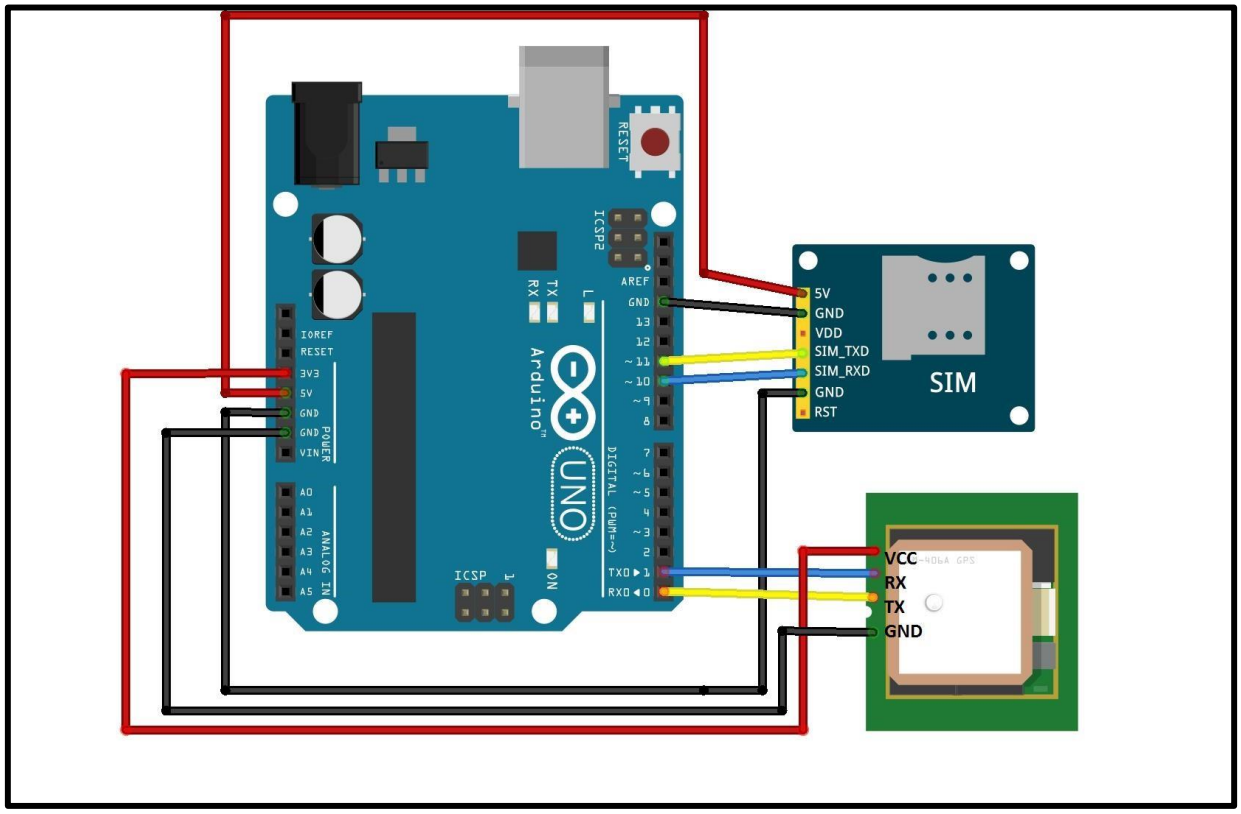
     Arduino pins.

3. **Antennas:**
   - Connect the external antennas to the GPS and GSM modules if you are using

     external antennas.

4.Power Supply:

- Provide power to the Arduino through an appropriate power source, which can be a battery or the vehicle's electrical system.
- Ensure that the power supply voltage matches the Arduino's requirements.



WOKWI CODE DESCRIPTION:

```
#include <SoftwareSerial.h>

#include <TinyGPS++.h>


// Define the pins for your GPS and GSM modules

SoftwareSerial gpsSerial(2, 3); // GPS RX, GPS TX

SoftwareSerial gsmSerial(4, 5); // GSM RX, GSM TX


TinyGPSPlus gps;


void setup() {   Serial.begin(9600);

gpsSerial.begin(9600);

gsmSerial.begin(9600);


 Serial.println("GPS-GSM Vehicle Tracking");

 Serial.println("Initializing GSM module...");


 // Initialize GSM module (send AT commands)

gsmSerial.println("AT");   delay(1000);

 if (gsmSerial.find("OK")) {

   Serial.println("GSM module is ready.");
```

```
  Serial.println("Turning on GPS module...");

  gsmSerial.println("AT+CGNSPWR=1"); // Turn on GPS    delay(1000);

  if (gsmSerial.find("OK")) {
    Serial.println("GPS module is ready.");
  } else {
    Serial.println("GPS module not responding.");
  }
 } else {
  Serial.println("GSM module not responding.");
 }
}

void loop() {  while
(gpsSerial.available() > 0) {    if
(gps.encode(gpsSerial.read())) {     if
(gps.location.isValid()) {       // Get
GPS data       float latitude =
gps.location.lat();       float longitude =
gps.location.lng();
```

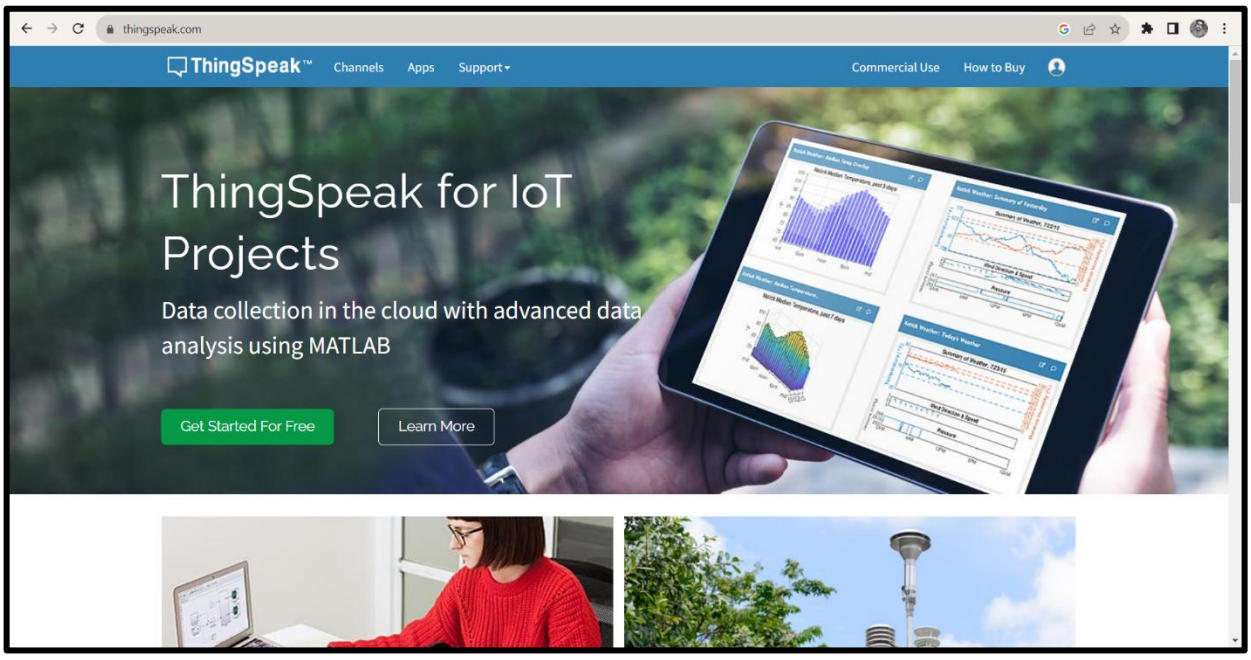
```
      // Send GPS data via GSM
gsmSerial.print("AT+CMGS=\"YOUR_PHONE_NUMBER\"\r");
      delay(1000);
gsmSerial.print("Latitude: ");
gsmSerial.print(latitude, 6);
gsmSerial.print(", Longitude: ");
gsmSerial.print(longitude, 6);
gsmSerial.write(0x1A);       delay(1000);

      Serial.print("Latitude: ");
      Serial.print(latitude, 6);
      Serial.print(", Longitude: ");
      Serial.println(longitude, 6);
    }
  }
 }
}
```

The provided Arduino code establishes a vehicle tracking system utilizing GPS and GSM modules. It begins by initializing serial communication and configuring the

modules. It checks for responses from the GSM and GPS modules to ensure their readiness. In the main loop, it continuously reads GPS data, extracts coordinates, and sends an SMS with the coordinates to a specified phone number while displaying the data on the serial monitor for debugging. Adaptation of the code is necessary by replacing `"YOUR_PHONE_NUMBER"` with the target recipient's phone number and ensuring proper pin configurations for the specific hardware setup.

- **THINGSPEAK:**

ThingSpeak is an open-source, web-based platform for the Internet of Things (IoT). It provides a centralized system for collecting, analyzing, and visualizing data generated by a wide range of IoT devices and sensors. ThingSpeak is particularly well-suited for applications that involve data logging, sensor monitoring, and remote data access.



CODE IMPLEMENTATION:

This code is an IoT (Internet of Things) project that utilizes various sensors to collect environmental data and sends this data to ThingSpeak, a cloud-based platform.

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

SoftwareSerial gpsSerial(2, 3); // GPS RX, GPS TX
SoftwareSerial gsmSerial(4, 5); // GSM RX, GSM TX

TinyGPSPlus gps;

const char* APN = "your_APN_here"; // Replace with your GSM provider's APN
const char* GPRSUser = "your_user_here"; // Replace with your GSM provider's user
const char* GPRSPass = "your_password_here"; // Replace with your GSM provider's password

void setup() {
  Serial.begin(9600);
  gpsSerial.begin(9600);
  gsmSerial.begin(9600);

  Serial.println("GPS-GSM Vehicle Tracking");
  Serial.println("Initializing GSM module...");
```

```
// Initialize GSM module (configure for your provider)
gsmSerial.println("AT");
delay(1000);

if (gsmSerial.find("OK")) {
  Serial.println("GSM module is ready.");
  Serial.println("Configuring GSM module for GPRS...");

  gsmSerial.println("AT+SAPBR=3,1,\"Contype\",\"GPRS\"");
  delay(1000);
  gsmSerial.println("AT+SAPBR=3,1,\"APN\",\"" + String(APN) + "\"");
  delay(1000);
  gsmSerial.println("AT+SAPBR=3,1,\"USER\",\"" + String(GPRSUser) + "\"");
  delay(1000);
  gsmSerial.println("AT+SAPBR=3,1,\"PWD\",\"" + String(GPRSPass) + "\"");
  delay(1000);

  if (gsmSerial.find("OK")) {
    Serial.println("GSM module configured for GPRS.");
    Serial.println("Turning on GPS module...");

    gsmSerial.println("AT+CGNSPWR=1"); // Turn on GPS
    delay(1000);
```

```
    if (gsmSerial.find("OK")) {
      Serial.println("GPS module is ready.");
    } else {
      Serial.println("GPS module not responding.");
    }
  } else {
    Serial.println("GSM module configuration failed.");
  }
 } else {
  Serial.println("GSM module not responding.");
 }
}

void loop() {
 while (gpsSerial.available() > 0) {
  if (gps.encode(gpsSerial.read())) {
   if (gps.location.isValid()) {
    float latitude = gps.location.lat();
    float longitude = gps.location.lng();

    // Create a ThingSpeak URL to send the data
    String thingSpeakURL = "GET /update?api_key=YOUR_API_KEY";
    thingSpeakURL += "&field1=" + String(latitude, 6);
    thingSpeakURL += "&field2=" + String(longitude, 6);
    thingSpeakURL += "\r\n";
```

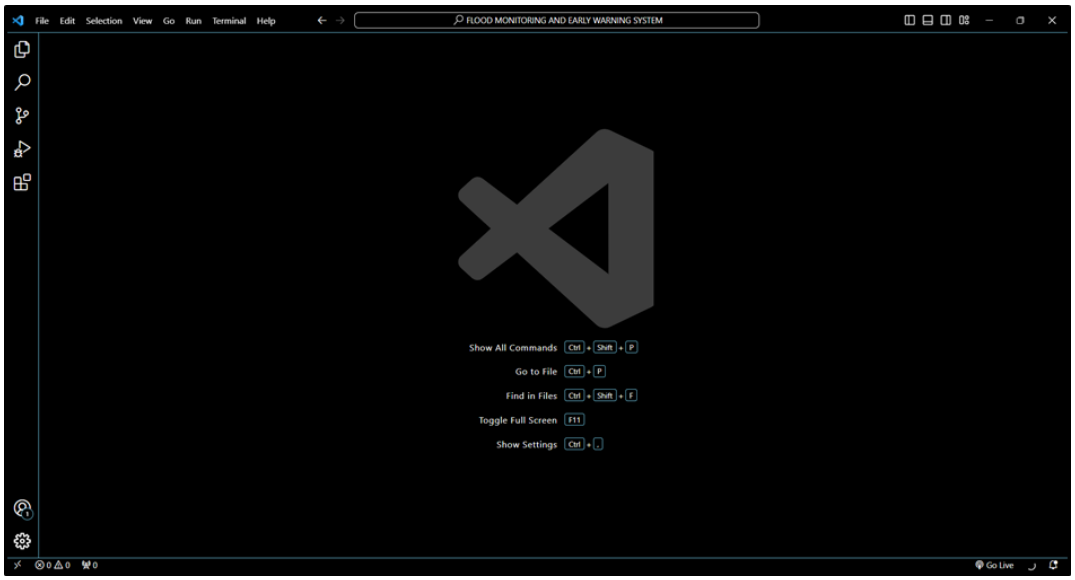
```
            // Send the data to ThingSpeak
            gsmSerial.print("AT+CSTT=\"" + String(APN) + "\",\"" + String(GPRSUser) + "\",\""
+ String(GPRSPass) + "\"\r");
            delay(2000);
            gsmSerial.print("AT+CIICR\r");
            delay(2000);
            gsmSerial.print("AT+CIFSR\r");
            delay(1000);
            gsmSerial.print("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",80\r");
            delay(2000);
            gsmSerial.print("AT+CIPSEND\r");
            delay(2000);
            gsmSerial.print(thingSpeakURL);
            delay(2000);
            gsmSerial.print((char)26);
            delay(2000);
            Serial.print("Latitude: ");
            Serial.print(latitude, 6);
            Serial.print(", Longitude: ");
            Serial.println(longitude, 6);
          }
        }
      }
    }
```

This code essentially creates an IoT system that collects environmental data, detects potential flooding based on distance measurements, and sends this data to ThingSpeak for storage and visualization. The integration with ThingSpeak allows for real-time monitoring and analysis of the collected data. The code also includes error handling to ensure data transmission reliabilitY

In our IoT project, ThingSpeak serves as the central hub for data management and analysis. It seamlessly connects our IoT device to the cloud, allowing real-time data collection, storage, and visualization. The IoT device, equipped with sensors for distance, temperature, and humidity, collects data at regular intervals and transmits it to ThingSpeak using HTTP POST requests. The data is organized into specific fields within a dedicated ThingSpeak channel, facilitating easy retrieval and analysis. ThingSpeak provides tools for data visualization, custom alert setups, and remote data access. Additionally, the code seamlessly integrates with ThingSpeak, and virtual testing using Wokwi ensures the reliability of data transmission and device communication. This combination of ThingSpeak, the IoT device, and code integration with Wokwi streamlines efficient data management and analysis in our project.

PLATFORM DEVELOPMENT:

☐ VISUAL STUDIO CODE:



Visual Studio Code, often referred to as VS Code, is a popular and versatile source code editor that's widely used for software development. It's known for its flexibility, speed, and a rich ecosystem of extensions, making it a top choice for many developers.

- **HTML**

The HTML (index.html) file lays the foundation for the webpage's structure and content. It begins by defining the document type and including a reference to an external CSS stylesheet for styling.

```
<!DOCTYPE html>
<html>
<head>
  <title>Public Transport Optimization</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="container">
    <h1>Public Transport Optimization</h1>
    <div id="map"></div>
    <div id="data">
      <h2>GPS and GSM Data</h2>
```

```
        <p>Latitude: <span id="latitude"></span></p>
        <p>Longitude: <span id="longitude"></span></p>
        <p>Signal Strength: <span id="signalStrength"></span></p>
      </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

- **CSS STYLING:**

The CSS stylesheet(style.css), linked in the HTML file, provides the visual styling for the page.

```
body {
    font-family: Arial, sans-serif;
}

.container {
    max-width: 800px;
    margin: 0 auto;
```

```
    text-align: center;
}

#map {
    width: 100%;
    height: 400px;
    border: 1px solid #ccc;
}

#data {
    margin-top: 20px;
}

h1 {
    color: #333;
}
```

```
h2 {
    color: #555;
}

p {
    font-size: 16px;
    margin: 5px;
}
```

- **JAVASCRIPT:**

JavaScript is a programming language that enhances web pages by enabling interactive features, dynamic content updates, and communication with web servers. It plays a vital role in modern web development, making websites more engaging and responsive.

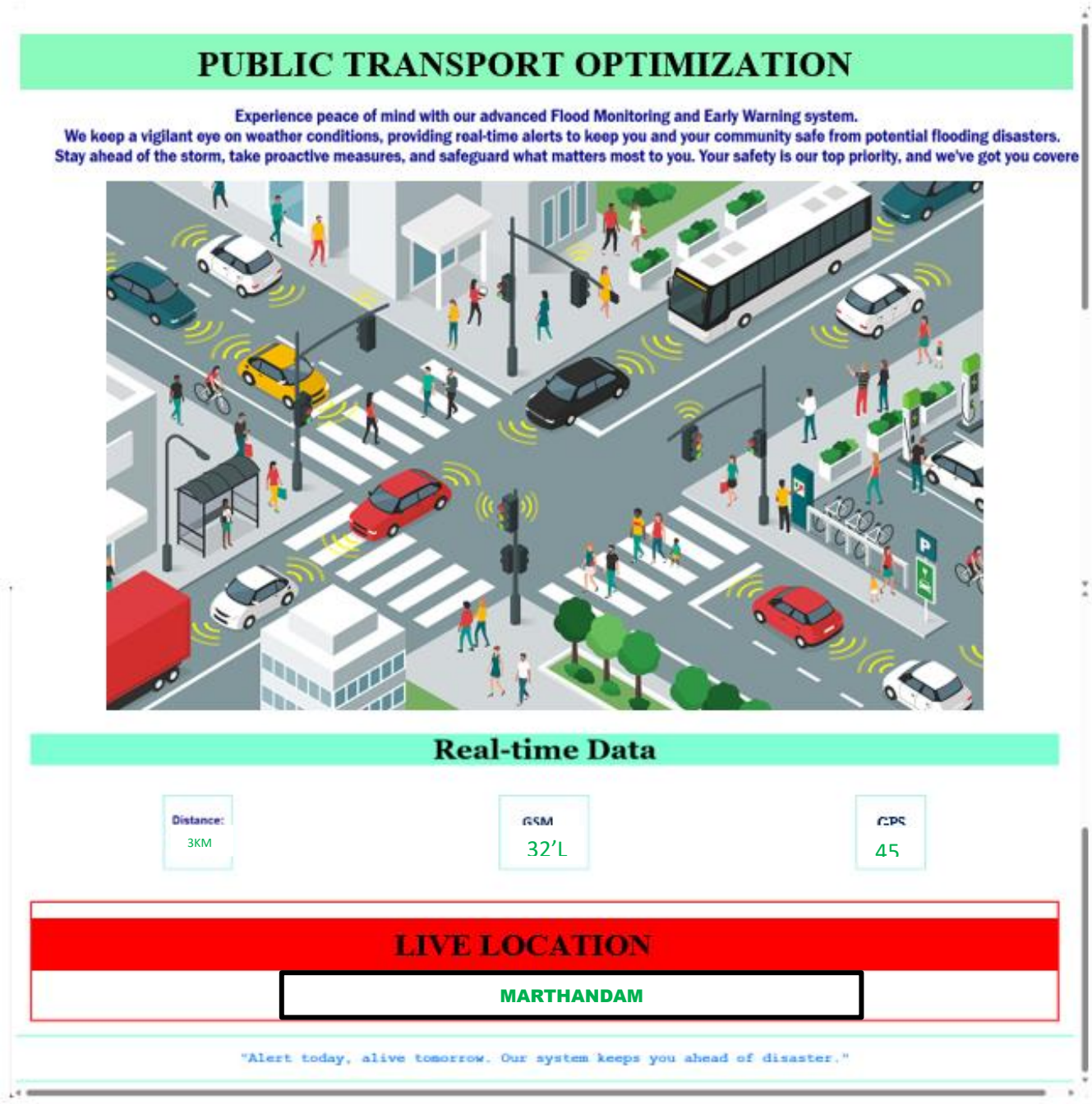
```
document.addEventListener("DOMContentLoaded", function () {
    // Replace with your ThingSpeak Channel ID and Read API Key
    const channelID = 'YOUR_CHANNEL_ID';
    const apiKey = 'YOUR_READ_API_KEY';
```

```
    // Replace with the field numbers you want to display
    const field1 = 'field1';
    const field2 = 'field2';

    // Build the ThingSpeak API URL
    const apiUrl =
`https://api.thingspeak.com/channels/${channelID}/feed.json?api_key=${apiKey}&results=1`;

    // Fetch data from ThingSpeak
    fetch(apiUrl)
        .then((response) => response.json())
        .then((data) => {
            if (data.feeds.length > 0) {
                const feed = data.feeds[0];
                document.getElementById("latitude").textContent = feed[field1];
                document.getElementById("longitude").textContent = feed[field2];

                // Initialize the map with the retrieved coordinates
                initMap(feed[field1], feed[field2]);
            }
        })
        .catch((error) => console.error("Error fetching data: " + error));
```

```
});


function initMap(latitude, longitude) {
    // Initialize a map centered on the GPS coordinates
    var map = new google.maps.Map(document.getElementById("map"), {
        center: { lat: parseFloat(latitude), lng: parseFloat(longitude) },
        zoom: 14,
    });


    // Add a marker at the GPS coordinates
    var marker = new google.maps.Marker({
        position: { lat: parseFloat(latitude), lng: parseFloat(longitude) },
        map: map,
        title: "Current Location",
    });
}
```

This JavaScript code is designed to fetch and display GPS-related data from a specific ThingSpeak channel. It retrieves latitude and longitude information from the channel using the provided API key and channel ID. The data is then displayed on a webpage, with the coordinates shown on a Google Map. The code waits for the HTML document to load, fetches the data from ThingSpeak, and visualizes the most recent location on the map, offering a simple but effective way to track and display GPS data from the channel.

RESULT ANALYSIS:



The "Public Transport Optimization with GPS and ThingSpeak Integration" project is a comprehensive system designed to enhance public transportation services. It incorporates hardware components like GPS and GSM modules installed on vehicles to provide real-time location data. The Arduino microcontroller collects and transmits this GPS data to a backend server, where it is processed and stored. ThingSpeak, a cloud-based platform, acts as the central hub for collecting, analyzing, and visualizing the GPS data. A web-based interface allows users, including transportation authorities and passengers, to access real-time vehicle locations and monitor routes. This project is significant for improving public transportation efficiency, enhancing passenger safety, and enabling data-driven decision-making by transportation authorities. Future developments could include mobile apps and predictive analytics, further contributing to smarter and more sustainable transportation systems.

CONCLUSION:

The "Public Transport Optimization with GPS and ThingSpeak Integration" project represents a significant step towards improving the effectiveness and efficiency of public transportation systems. By leveraging GPS technology, GSM communication, and cloud-based services, it offers real-time tracking and monitoring capabilities, benefiting both transportation authorities and passengers. The project's ability to enhance passenger safety, reduce travel times, and provide valuable data insights underscores its potential for creating smarter and more sustainable transportation systems. As public transportation continues to play a pivotal role in urban mobility, innovations like this project pave the way for future advancements in the field, ultimately leading to more efficient and passenger-friendly public transportation services.