Internet Of Things

Public transport optimization using IOT sensors

Phase 5:Project Documentation

Introduction:

The main role of this project is optimization of public transport by IOT Technology .such that integrating iot devices like gps sensor and esp32 in all the public vehicle in the city and connect to internet through wifi. Code esp32 with python by importing necessary libraries for calculating latitude and longitude information from the gps sensor then the esp32 sensor made a Http post request to the server to store latitude and longitude information to the real time database(i.e.,Firebase) .By this real time information in our website the user can see live location of vehicle,this reduce the wait time at the bus station by knowing the nearest buses to a user, the real-time location of buses on the Google map to help passengers track buses in real-time, the arrival time of buses, and speed.

Project Objectives:

The **Public Transport Optimization** project aims to enhance public transportation services by integrating IoT sensors into public transportation vehicles. These sensors will be used to monitor ridership, track vehicle locations, and

predict arrival times. The ultimate goal is to provide real-time transit information to the public through a user-friendly web-based platform .

The project has the following objectives:

Provide passengers with real-time information about public transportation services, including vehicle locations and arrival times.

Develop algorithms to predict accurate arrival times based on real-time data and historical patterns.

Implement passenger counting mechanisms to monitor and report ridership on public transportation vehicles.

Utilize IoT technology to optimize public transportation schedules, reduce costs, and improve overall service quality .

To achieve these objectives, the project will design an IoT sensor system that includes GPS sensors, passenger counters, and additional sensors to collect environmental data such as temperature, humidity, and air quality. The sensors will be connected to a microcontroller that will collect, process, and manage the data locally within the vehicles. The system will ensure reliable connectivity through cellular networks, Wi-Fi, or other wireless technologies for real-time data transmission [1].

The project will also develop a web-based platform that displays the live locations of public transportation vehicles on a user-friendly map interface. Machine learning algorithms will be developed to predict arrival times accurately based on real-time data and historical patterns. Current ridership levels on each vehicle will be presented to help passengers make informed travel decisions. Relevant information on service disruptions, delays, and other announcements impacting passengers will also be provided.

IoT device setup:

The IOT sensors used in the project are esp32 with wifi module, gps sensor. Let us integrate this into public vehicle as follows

Connect the VCC pin of the GPS module to the ESP32 3.3V pin.

1. Connect the GPS ground pin to the ESP32 ground pin.
2. Connect the RX pin of the GPS module to the TX pin of the ESP32.
3. Connect your ESP32 to the computer through a USB cable.

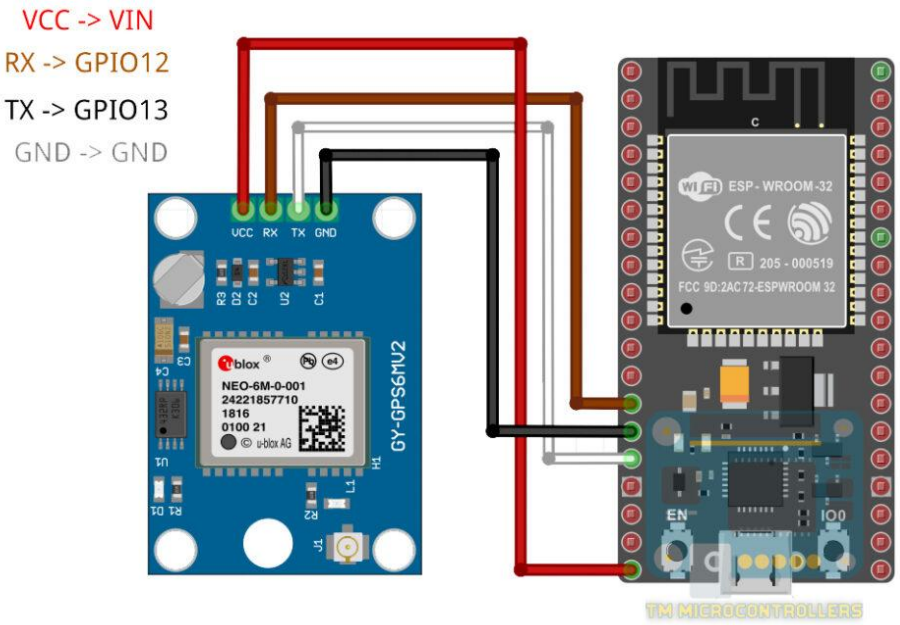
4. Program the ESP32 via Arduino IDE with python to calculate latitude and longitude information from gps data .
5. Integrate this setup to public vehicle.
6. Send the GPS data, read by the ESP32 from the GPS device, to an external web server to store it on the real time database(Firebase).

Here is a sample code that uses MicroPython to interface an ESP32 with a NEO-6M GPS module and obtain GPS parameters such as latitude, longitude, altitude, date, time, speed, satellites, etc. The code also shows how to store the obtained data on Firebase.

Code

```
# Import required libraries
import machine
import time
import network
import urequests as requests
from machine import UART

# Set up the UART interface for GPS module
uart = UART(2, baudrate=9600, tx=17, rx=16)

# Set up the Wi-Fi connection
ssid = 'your_wifi_ssid'
password = 'your_wifi_password'
station = network.WLAN(network.STA_IF)
station.active(True)
station.connect(ssid, password)

# Set up the Firebase database URL and authentication token
url =
'https://your_firebase_database_url.firebaseio.com/your_database_name.json'
auth_token = 'your_firebase_auth_token'

# Define a function to obtain GPS data from the NEO-6M module
def get_gps_data():
    while True:
        # Read the GPS data from the UART interface
        gps_data = uart.readline()
        if gps_data.startswith(b'$GPGGA'):
            # Parse the GPS data to obtain latitude and longitude
            gps_data_list = gps_data.split(b',')
            latitude = gps_data_list[2]
            longitude = gps_data_list[4]
```

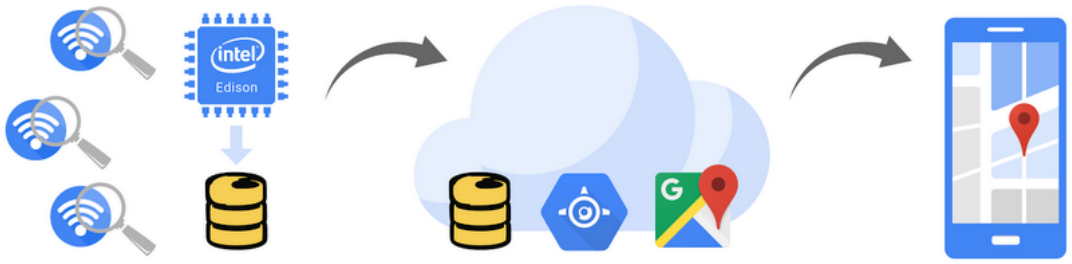
```
        # Return the latitude and longitude as strings
        return latitude.decode('utf-8'), longitude.decode('utf-8')

# Define a function to store the GPS data on Firebase
def store_gps_data_on_firebase(latitude, longitude):
    # Create a dictionary with the GPS data
    gps_data_dict = {'latitude': latitude, 'longitude': longitude}
    # Send a POST request to Firebase to store the GPS data
    response = requests.post(url + '?auth=' + auth_token, json=gps_data_dict)
    # Print the response from Firebase
    print(response.text)

# Obtain GPS data and store it on Firebase every 10 seconds
while True:
    # Wait for 10 seconds before obtaining and storing GPS data again
    time.sleep(10)
    # Obtain GPS data from the NEO-6M module
    latitude, longitude = get_gps_data()
    # Store the GPS data on Firebase
    store_gps_data_on_firebase(latitude, longitude)
```



Create a database in firebase for our project store the latitude and longitude data into it ,every 10 seconds this data will be updated by new data come from esp32 sensors integrated with each of the vehicle in the city.Next step is display the live location of vehicle the user need to see in the website.the location of vehicle ,arrival

time and the velocity of the vehicle will be accurate so that we go for loading and preprocessing the datasets stored in the firebase

.



Loading and Preprocessing of datasets:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.In our case we collect data from firebase and then preprocessing the location data to show accurate location of the vehicle to the user.

(i)Get the dataset:

Firstly get the dataset from the firebase database to preprocess it,

1.Connect to Firebase:

First, you need to establish a connection to your Firebase database. You can use the pyrebase package in Python for this.

```
import pyrebase

config = {
  "apiKey": "your-api-key",
  "authDomain": "your-auth-domain",
  "databaseURL": "your-database-url",
  "storageBucket": "your-storage-bucket"
```

```
}

firebase = pyrebase.initialize_app(config)
db = firebase.database()
```

2.Load the data:

Fetch the data from the Firebase database.

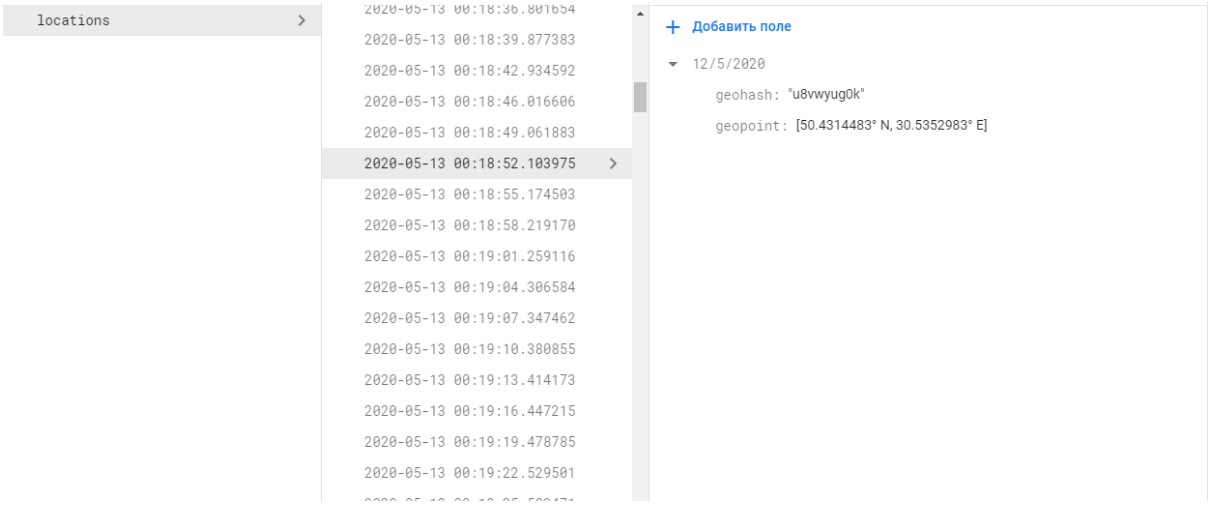
```
data = db.child("your-database-child").get().val()
```

3.Convert to DataFrame:

Convert the data into a pandas DataFrame for easier manipulation.

```
import pandas as pd

df = pd.DataFrame(data)
```

in our web application .



This is the dataset of single node which store latitude and longitude as geopoint every four seconds it will update it.

(ii)Importiing Libraries:

Install the geopy library, which is a popular package for geocoding and reverse geocoding in Python. Geocoding is the process of converting addresses into geographical coordinates, and reverse geocoding is the opposite. You can install geopy using the command in the terminal

```
pip install geopy
```

Install the lat-lon-parser library, which is a package for parsing lat-long coordinates in various formats, and for converting between lat-long formats (e.g. decimal degrees to degrees-minutes-seconds). You can install lat-lon-parser using the command

```
pip install lat-lon-parser
```

Now you need to import the modules from these libraries in your Python script. For example, you can use the following lines of code to import the Nominatim geocoder from geopy and the parse function from lat-lon-parser:

```
from geopy.geocoders import Nominatim
from lat_lon_parser import parse
```

(iii)Preprocess dataset:

Inspect the data:Check the first few rows of your data to understand its structure. Also, check if there are any missing values in the latitude and longitude columns.

```
print(df.head())
print(df.isnull().sum())
```

Handling missing values:Preprocess the dataset is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.
There are mainly two ways to handle missing data, which are:
(i)By deleting a particular row.

If the number of rows with missing values is small, you might opt to remove these rows.

```
df = df.dropna(subset=['latitude', 'longitude'])
```

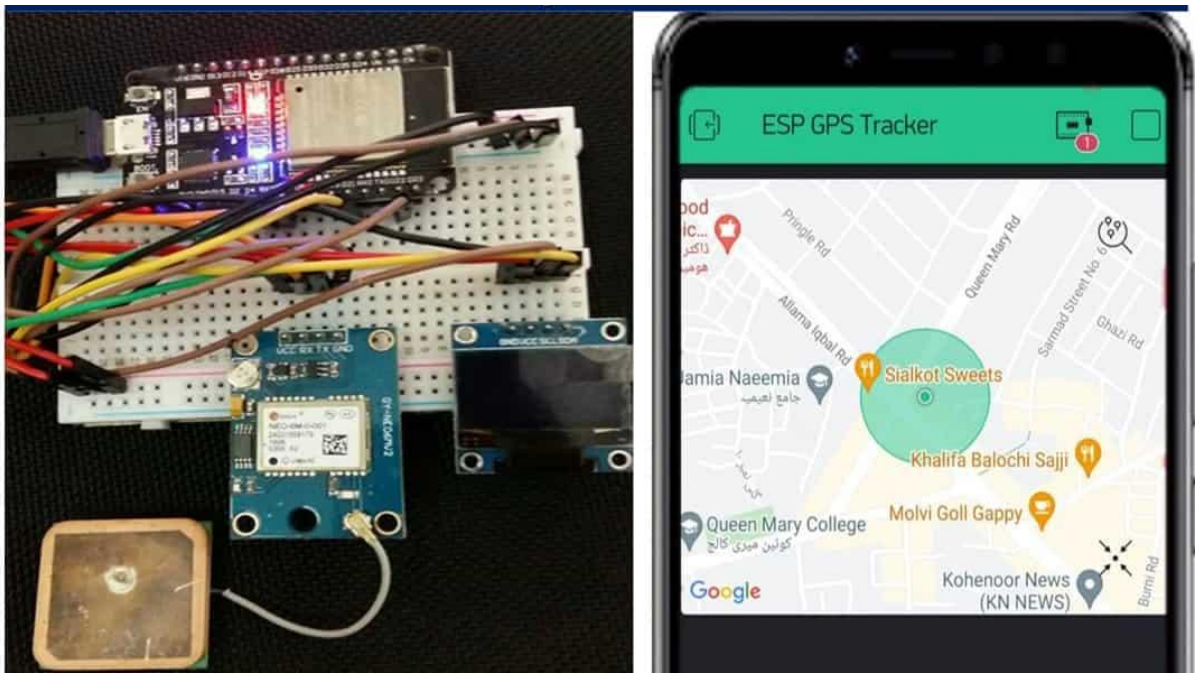
(ii)By calaculating the missing data with the help of present data.

If you can't afford to lose data, you might decide to fill the missing values with some value. This could be the mean or median of the available values, or some other strategy.

```
df['latitude'].fillna(df['latitude'].mean(), inplace=True)
df['longitude'].fillna(df['longitude'].mean(), inplace=True)
```

The missing of data occurs due to the break of internet connection of particular node in very small duration .so that maintain unbreakable of internet is necessary.

Now with this data we can track the live location of the vehicle in our web application.



Platform development:

Now, we developing the real-time transit information platform by using web technologies (e.g.,HTML, CSS, javascript) to create a platform that display real time transit information by retrieving latitude and longitude information stored in the firebase then display the vehicle live location that the user want to see.steps include are in the following:

(i)Create a simple HTML page with search engine to search particular vehicle location and then style it using CSS.

(ii)Create google map api key in google cloud platform .

(iii)Integrate Google map javascript api into the website using google map api key.

(iv)Retrieving latitude and longitude data of the particular vehicle from the Firebase,So that the real time location of the vehicle is displayed on the website.

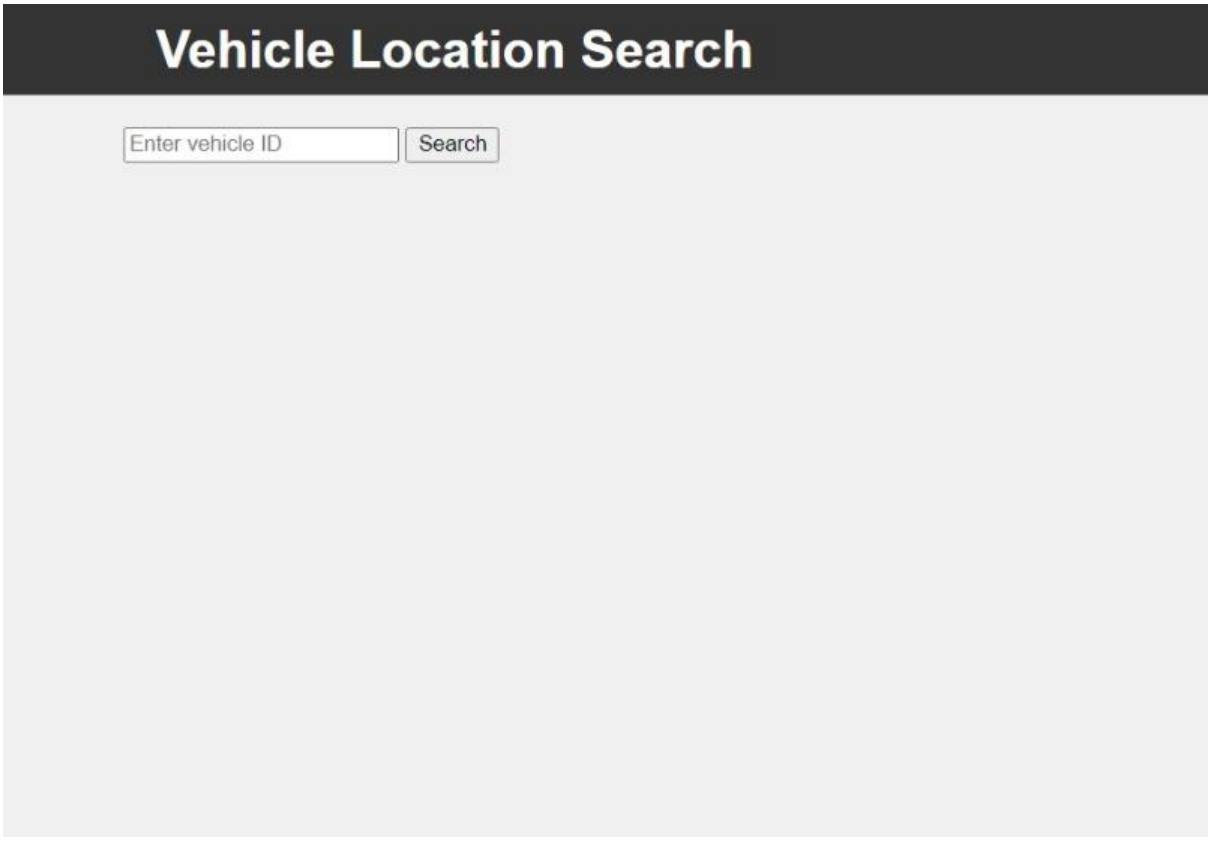
Create a simple HTML page with search engine to search particular vehicle location and then style it using CSS:

Let’s create an HTML page with a search engine to search for a particular vehicle location.
The below creates a webpage with a search form. When the form is submitted, it prevents the page from reloading (which is the default behaviour of forms) and retrieves the value from the input field.

CODE:

```
<!DOCTYPE html>
<html>
<head>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 0;
         padding: 0;
         background-color: #f0f0f0;
      }
      .container {
         width: 80%;
         margin: 0 auto;
      }
      header {
         background-color: #333;
         color: white;
         padding: 10px 0;
      }
      header h1 {
         margin: 0;
         padding: 0 20px;
      }
      #search-form {
         margin-top: 20px;
      }
   </style>
</head>
<body>
   <header>
      <div class="container">
         <h1>Vehicle Location Search</h1>
      </div>
   </header>
   <div class="container">
      <form id="search-form">
         <input type="text" id="search-input" placeholder="Enter vehicle ID">
```

```
        <button type="submit">Search</button>
      </form>
   </div>

   <script>
      document.getElementById('search-form').addEventListener('submit',
function(event) {
         event.preventDefault();
         var vehicleId = document.getElementById('search-input').value;
         // TODO: Perform search for vehicle location
      });
   </script>
</body>
</html>
```



(ii)Create google map api key in google cloud platform .

The API key is a unique identifier that authenticates requests associated with your project for usage and billing purposes. You must have at least one API key associated with your project.

To create a Google Maps API key in the Google Cloud Platform, you can follow these steps

1. Go to the Google Maps Platform > Credentials page
2. On the Credentials page, click Create credentials > API key

3. The API key created dialog displays your newly created API key
4. Click Close. The new API key is listed on the Credentials page under API keys.

Remember to restrict the API key before using it in production. Restricting your API keys adds security to your application by ensuring they are only used for appropriate services and APIs.

(iii)Integrate Google map javascript api into the website using google map api key.

Let's see how to load the Maps JavaScript API. There are three ways to do this:

- Use Dynamic Library Import (Recommended)
- Use the NPM js-api-loader package
- Use the legacy script loading tag

Use Dynamic Library Import

This method is highly recommended because it is the very simple:

Add the bootstrap loader code directly to your JavaScript code.

To load libraries at runtime, use the await operator to call importLibrary() from within an async function, as shown in the following code example:

```
let map;

async function initMap() {
  const { Map } = await google.maps.importLibrary("maps");

  map = new Map(document.getElementById("map"), {
    center: { lat: -34.397, lng: 150.644 },
    zoom: 8,
  });
}

initMap();
```

Required parameters

- key: Your API key. The Maps JavaScript API will not load unless a valid API key is specified.

Optional parameters

- v: The version of the Maps JavaScript API to load.
- libraries: A comma-separated list of additional Maps JavaScript API libraries to load. Specifying a fixed set of libraries is not generally recommended, but is available for developers who wish to finely tune the caching behavior on their website.
- language: The language to use. This affects the names of controls, copyright notices, driving directions, and control labels, and the responses to service requests. See the list of supported languages.
- region: The region code to use. This alters the map's behavior based on a given country or territory.
- solutionChannel: Google Maps Platform provides many types of sample code to help you get up and running quickly. To track adoption of our more complex code samples and improve solution quality, Google includes the solutionChannel query parameter in API calls in our sample code.
- authReferrerPolicy: Maps JS customers can configure HTTP Referrer Restrictions in the Cloud Console to limit which URLs are allowed to use a particular API Key. By default, these restrictions can be configured to allow only certain paths to use an API Key. If any URL on the same domain or origin may use the API Key, you can set authReferrerPolicy: "origin" to limit the amount of data sent when authorizing requests from the Maps JavaScript API. When this parameter is specified and HTTP Referrer Restrictions are enabled on Cloud Console, Maps JavaScript API will only be able to load if there is an HTTP Referrer Restriction that matches the current website's domain without a path specified.

(iv)Retrieving latitude and longitude data of the particular vehicle from the Firebase,So that the real time location of the vehicle is displayed on the website.

To retrieve the data, add a listener to the DatabaseReference of the vehicle. Here's an example:

CODE:

```
DatabaseReference locationRef = db.child("location"); // replace "location" with the id of the vehicle
locationRef.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot dataSnapshot) {
        double latitude = dataSnapshot.child("lat").getValue(Double.class);
        double longitude = dataSnapshot.child("lng").getValue(Double.class);
        // Now you have the latitude and longitude
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // Handle possible errors.
```

```
  }
});
```

This code retrieves the latitude and longitude whenever they change

This data is given to the html page to display the real time transit information of the vehicle.Now successfully track the real time transit location of the bus by using iot sensors and display it in our website.

Conclusion:

To conclude, the project of public transport optimization using ESP32 and GPS IoT sensor has been successfully implemented. The system provides real-time location data of the vehicle on a web platform by retrieving the location data stored in Firebase by the ESP32 sensor. The system is designed to be low-cost, easy-to-use, and efficient. It consists of a GPS module and an ESP32 microcontroller with Wi-Fi capabilities that are installed inside the buses. The GPS module is used to obtain real-time bus location, speed, and direction data. This data is then sent to the ESP32 microcontroller, which stores it in Firebase's real-time database. The web platform retrieves this data from Firebase and displays it in real-time on a map. The system can be used to monitor public transport vehicles from anywhere in the world. It can also be used for component monitoring, vehicle analysis, and fleet management.

Real time location tracking