IoT and Mindsphere training conducted by TANSAM (Internship Project)

College name:	PANIMALR INSTITUTE OF TECHNOLOGY
Student's name:	1. SANDOSH KUMAR R
	2. UDHAYANIDHI M
	3. SIVANESHWARAN J
BATCH no:	8
Project no:	10
Project Title:	Track vehicles using an ESP32 GPS tracker for IoT-based vehicle tracking

Abstract:

- This project proposes a vehicle tracking system employing an ESP32 microcontroller integrated with a GPS module for IoT-based vehicle tracking. Leveraging the ESP32's capabilities and its compatibility with various IoT platforms, such as Blynk and AskSensors, the system accurately captures the vehicle's location in real-time, including latitude and longitude coordinates. These coordinates are wirelessly transmitted to the chosen IoT platform for visualization and analysis. This integration offers fleet managers efficient real-time monitoring, route optimization, and enhanced security features, ultimately contributing to improved transportation management efficiency and safety.

Application with real time used case:

1. Real-Time GPS Tracker with Adafruit Map:
2. IoT-based Vehicle Tracking System:
3. GPS Tracker Using ESP32 and A9G Module:
4. GPS Tracking over MQTT with AskSensors IoT Cloud:
5. Integration with Google Maps:

Objective:

The primary objectives of using an ESP32 GPS tracker for realtime vehicle tracking in IoT-based applications are:

- **Global Tracking:** To enable the tracking of vehicles from anywhere around the world, providing real-time location data in the form of longitude and latitude.
- **Theft Prevention:** To aid in the prevention of vehicle theft and assist in the recovery of stolen vehicles using real-time location tracking.
- **Monitoring and Management:** To allow user to monitor and manage the location of vehicles through mobile applications or web platforms, enhancing fleet management capabilities.
- **Data Accuracy:** To utilize the Global Navigation Satellite System (GNSS) Network for transmitting accurate location data received by the GPS receiver module.
- **User Conveinence:** To display GPS location coordinates on an OLED display connected with the ESP32 board and on mobile applications like the Blynk app for user convenience.

- **Integration with IoT Platforms:** To integrate with IoT platforms and cloud services for advanced monitoring, data analysis, and remote management of the vehicle tracking system.

These objectives contribute to the efficiency and effectiveness of vehicle tracking systms, making them invaluable tools for personal and commercial use.

Components required:

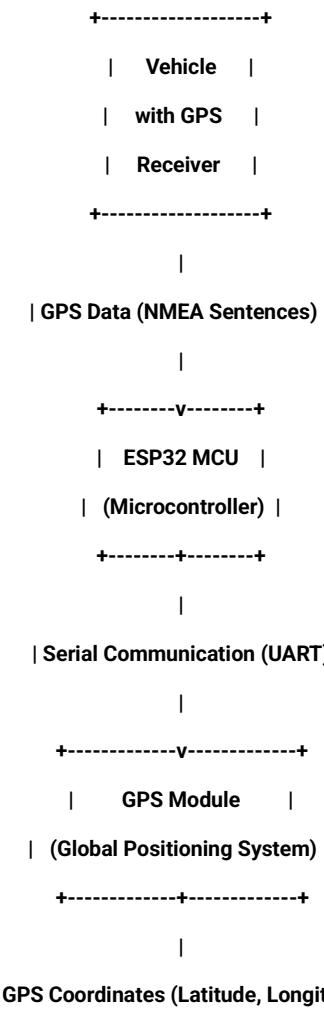
1. **ESP32 Development Board**: Acts as the central processing unit.
2. **GPS Module (e.g., NEO-6M)**: Captures the geographical coordinates.
3. **OLED Display Module**: Optionally displays the coordinates in real-time.
4. **Jumper Wires and Breadboard**: For connecting the components.

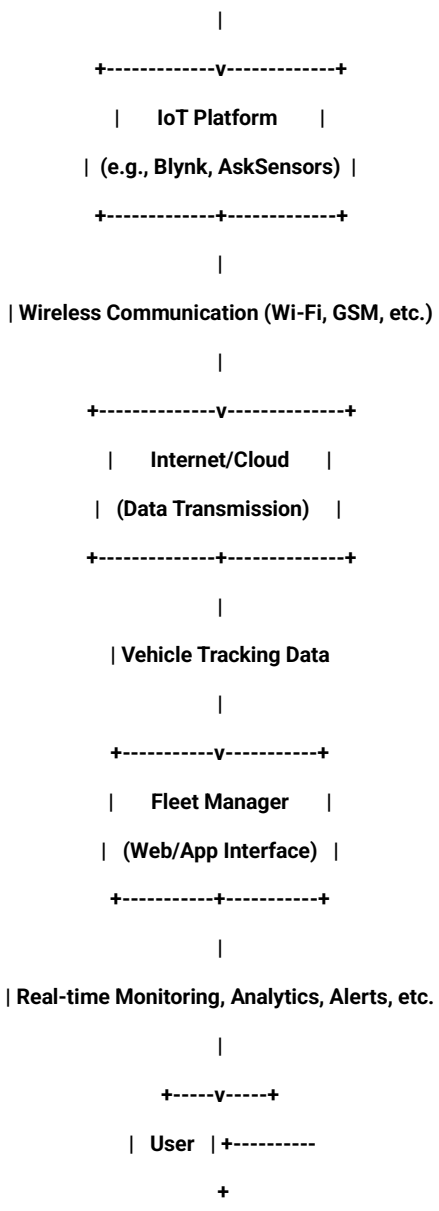
Software used:

- Arduino IDE: This is the integrated development environment where you will write and upload your code to the ESP32 board.
- Blynk App: An IoT platform that allows you to monitor and track the vehicle's location from anywhere.

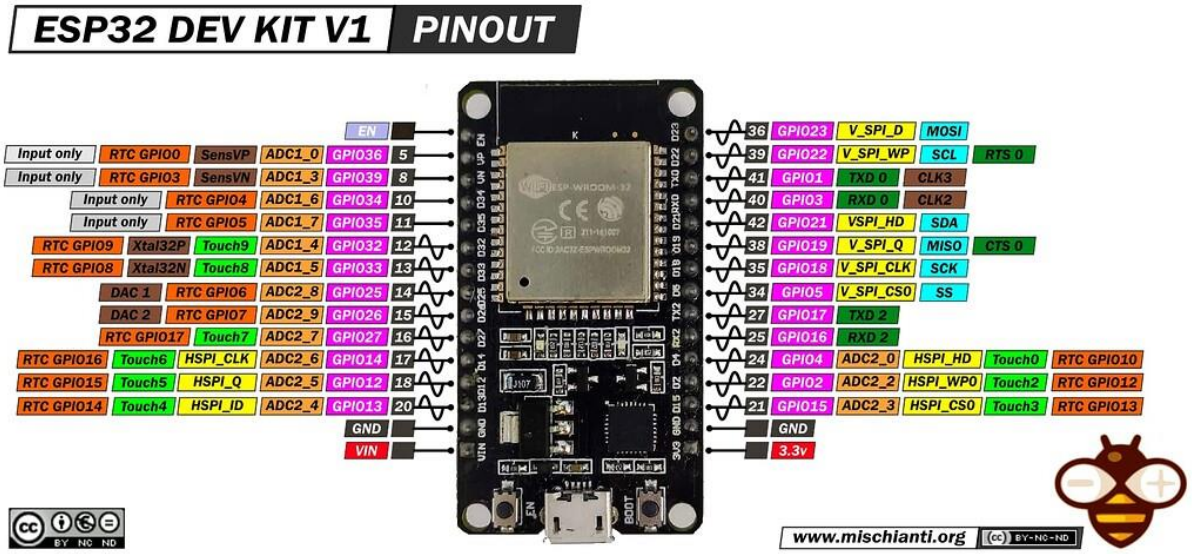
- AskSensors IoT Cloud: This platform can be used to publish latitude and longitude positions to the cloud in real-time over MQTT protocol.

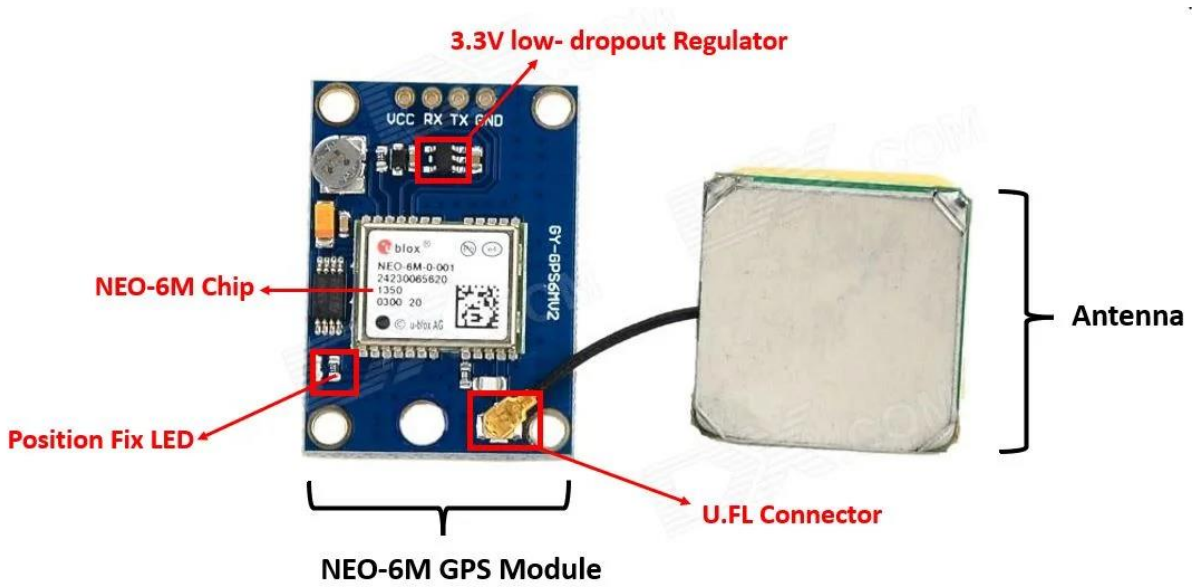
Block Diagram:

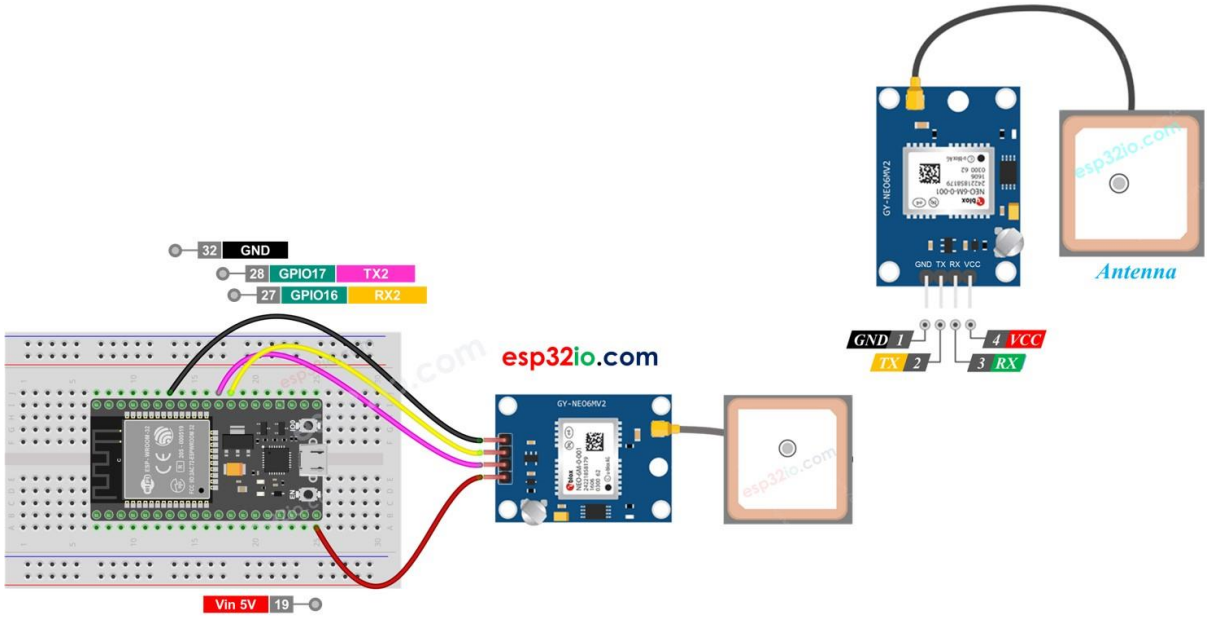


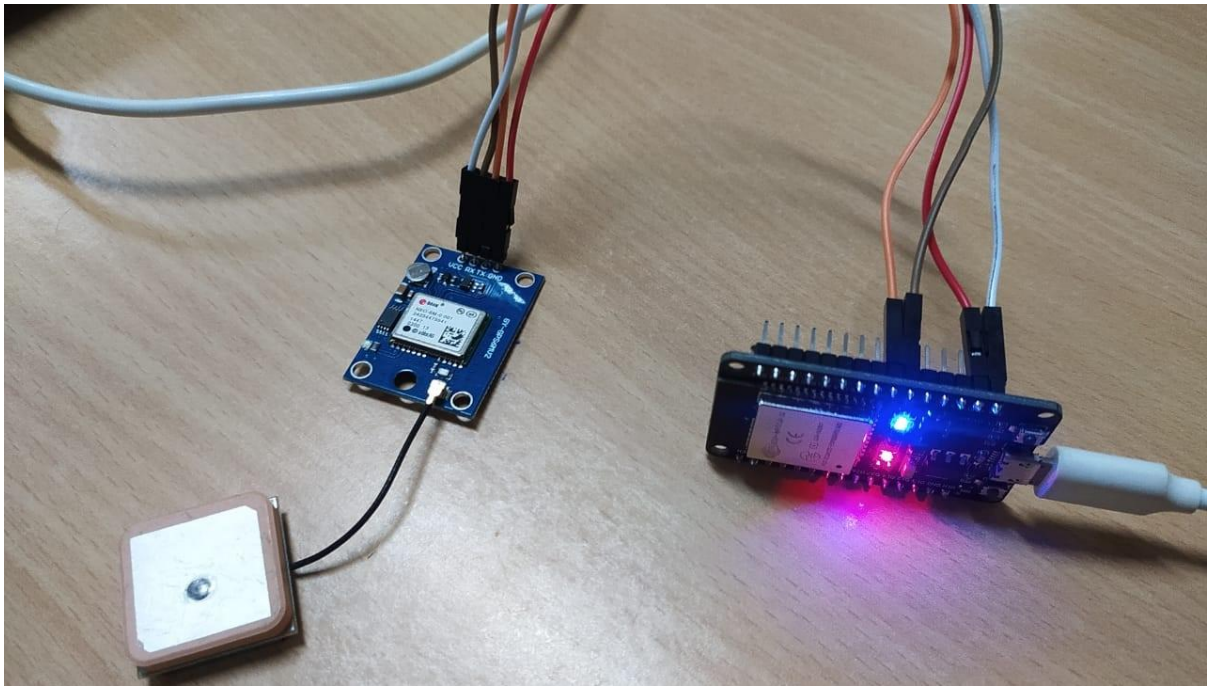

|
+----------v----------+
| IoT Platform |
| (e.g., Blynk, AskSensors) |
+----------+----------+
|
| Wireless Communication (Wi-Fi, GSM, etc.)
|
+----------v----------+
| Internet/Cloud |
| (Data Transmission) |
+----------+----------+
|
| Vehicle Tracking Data
|
+----------v----------+
| Fleet Manager |
| (Web/App Interface) |
+----------+----------+
|
| Real-time Monitoring, Analytics, Alerts, etc.
|
+-----v-----+
| User | +----------
+

Connection view:






32 GND
28 GPIO17 TX2
27 GPIO16 RX2
Vin 5V 19
esp32io.com
GY-NEO6MV2
GND TX RX VCC
GND 1
TX 2
4 VCC
3 RX
Antenna

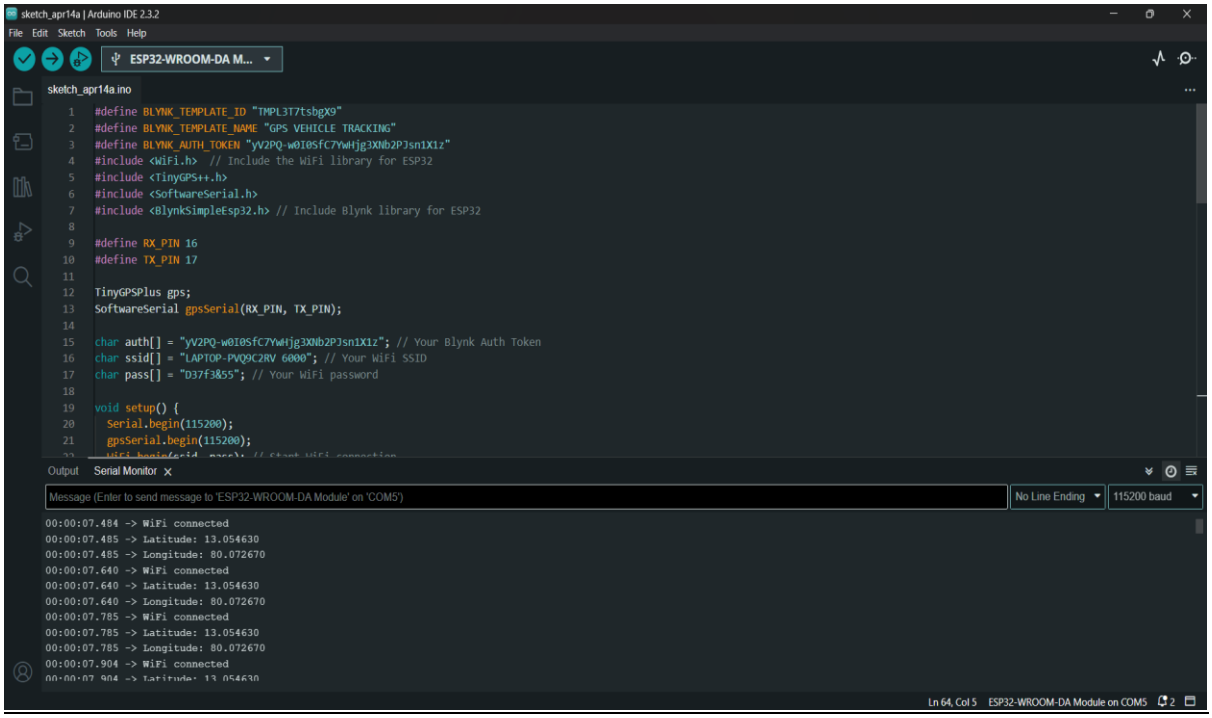


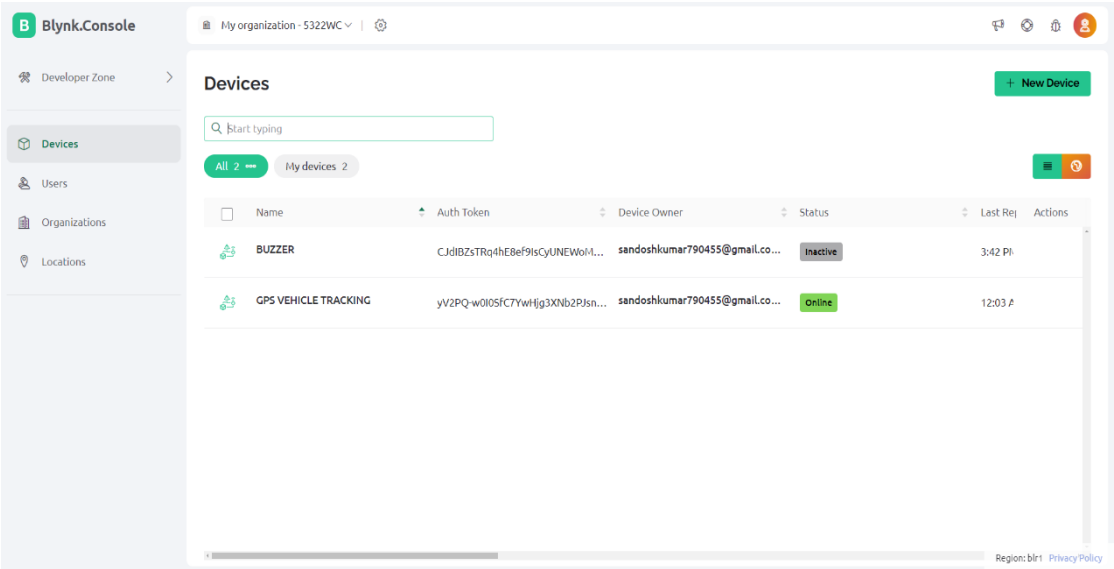
Code:

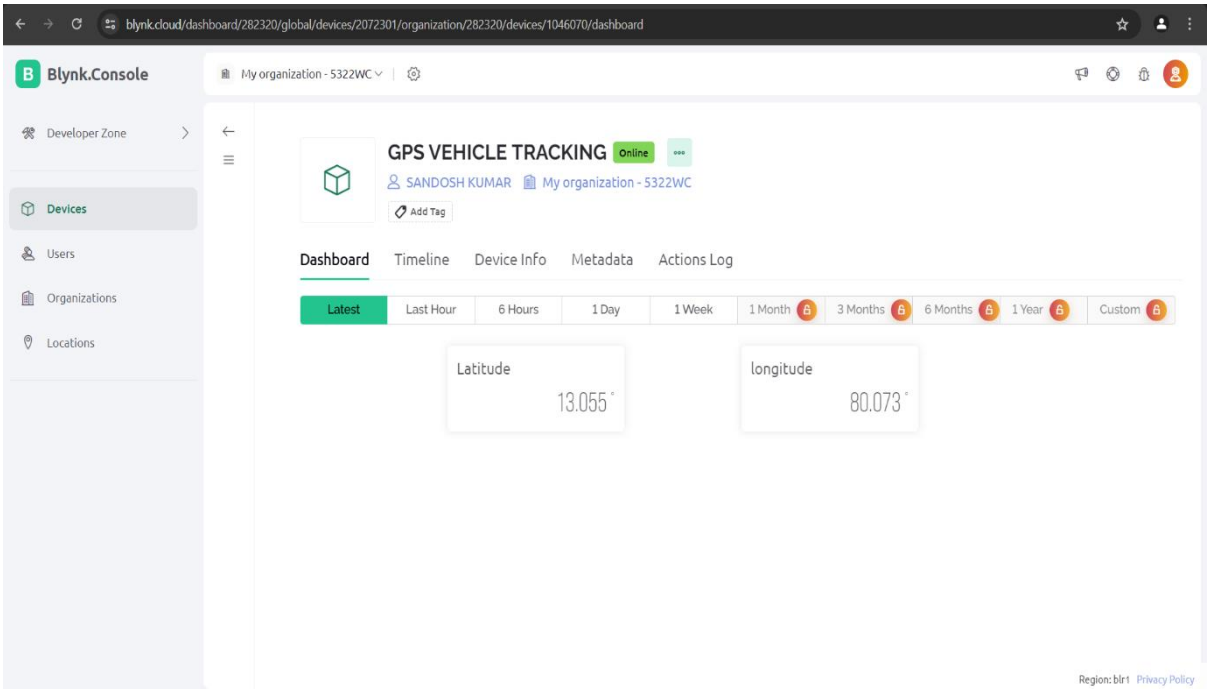
```
#define BLYNK_TEMPLATE_ID "TMPL3T7tsbgX9"
#define BLYNK_TEMPLATE_NAME "GPS VEHICLE TRACKING"
#define BLYNK_AUTH_TOKEN "yV2PQ-w0I0SfC7YwHjg3XNb2PJsn1X1z"
#include <WiFi.h>  // Include the WiFi library for ESP32
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
#include <BlynkSimpleEsp32.h> // Include Blynk library for ESP32

#define RX_PIN 16
#define TX_PIN 17

TinyGPSPlus gps;
SoftwareSerial gpsSerial(RX_PIN, TX_PIN);

char auth[] = "yV2PQ-w0I0SfC7YwHjg3XNb2PJsn1X1z"; // Your Blynk Auth Token
char ssid[] = "LAPTOP-PVQ9C2RV 6000"; // Your WiFi SSID
char pass[] = "D37f3&55"; // Your WiFi password

void setup() {
  Serial.begin(115200);
  gpsSerial.begin(115200);
  WiFi.begin(ssid, pass); // Start WiFi connection
  Blynk.begin(auth, ssid, pass); // Connect to Blynk
}

void checkWiFiAndReconnectIfNeeded() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Connecting to WiFi...");
    WiFi.begin(ssid, pass);
  }

  if (!Blynk.connected()) {
    Serial.println("Lost connection to Blynk server. Reconnecting...");
    Blynk.connect();
  }
}

void loop() {
  Blynk.run();
  checkWiFiAndReconnectIfNeeded(); // Check WiFi and Blynk connection

  if (WiFi.status() == WL_CONNECTED) {
    Serial.println("WiFi connected");
  } else {
    Serial.println("WiFi not connected");
```

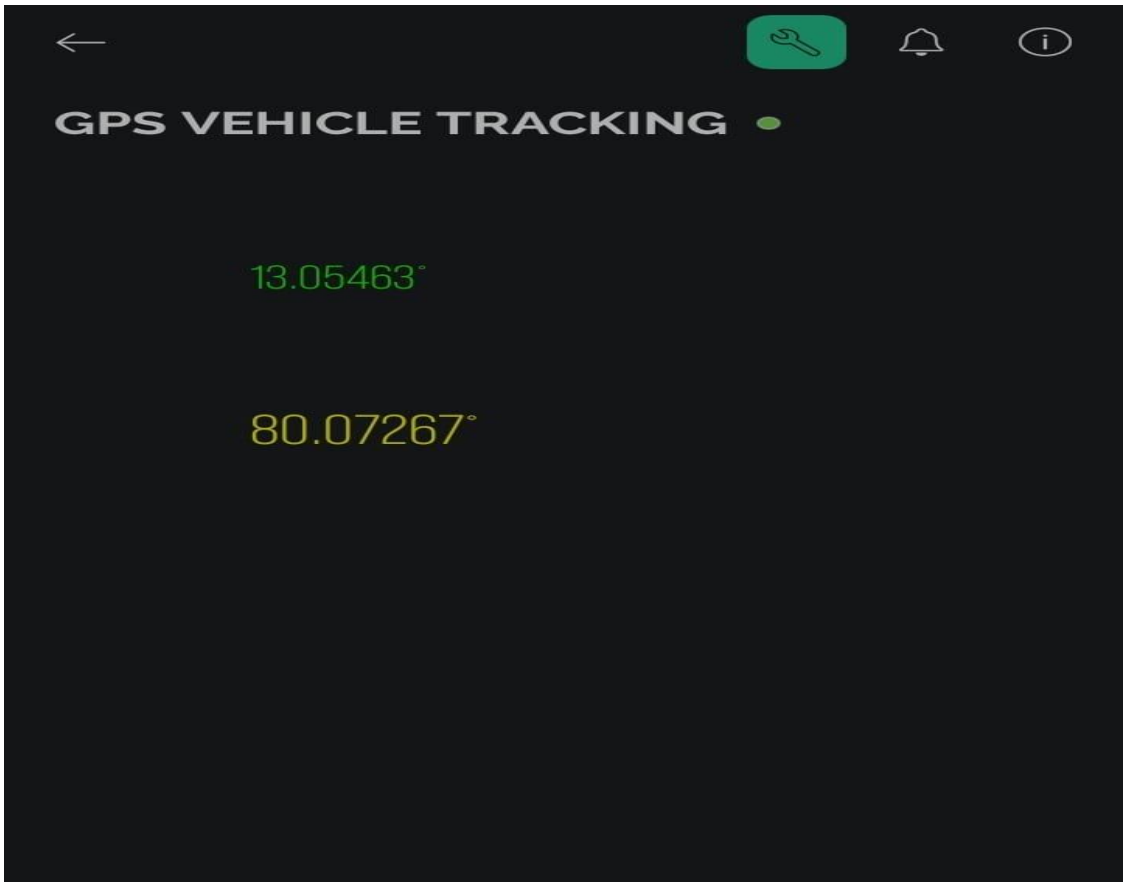
```
  }

  bool gpsDataReceived = false;
  bool gpsDataValid = false;

  while (gpsSerial.available() > 0) {
    gpsDataReceived = true;
    char c = gpsSerial.read();
    Serial.write(c);  // Print raw GPS data to help with debugging
    if (gps.encode(c)) {
      if (gps.location.isValid()) {
        gpsDataValid = true;
        float latitude = gps.location.lat();
        float longitude = gps.location.lng();

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

        // Send GPS data to Blynk app
        Blynk.virtualWrite(V0, latitude);
        Blynk.virtualWrite(V1, longitude);
      }
    }
  }

  if (gpsDataReceived && !gpsDataValid) {
    Serial.println("No GPS detected");
  } else if (!gpsDataReceived) {
    Serial.println("No GPS data received");
  }

  delay(1000); // Delay for 1 second to reduce the loop execution frequency
}
```

Screenshots:






blynk.cloud/dashboard/282320/global/devices/2072301/organization/282320/devices/1046070/dashboard
Blynk.Console
My organization - 5322WC
Developer Zone
Devices
Users
Organizations
Locations
GPS VEHICLE TRACKING
Online
SANDOSH KUMAR
My organization - 5322WC
Add Tag
Dashboard
Timeline
Device Info
Metadata
Actions Log
Latest
Last Hour
6 Hours
1 Day
1 Week
1 Month
3 Months
6 Months
1 Year
Custom
Latitude
13.055°
longitude
80.073°
Region: blr1
Privacy Policy


GPS VEHICLE TRACKING
13.05463°
80.07267°

Improvisation:

Certainly! Let's delve into a more detailed improvisation on how you can create a robust IoT-based vehicle tracking system using an ESP32 GPS tracker:

1. **Advanced Hardware Setup:**
    - Instead of just a basic ESP32 development board, consider using a custom-designed PCB (Printed Circuit Board) for the tracker unit. This can integrate additional features like voltage regulation, power management, and multiple communication interfaces (WiFi, Bluetooth, LoRa, etc.).
    - Opt for a high-precision GPS module with support for multiple satellite systems (GPS, GLONASS, Galileo, etc.) to ensure accurate positioning even in challenging environments.

2. **Enhanced Software Development:**
    - Implement advanced algorithms for data processing directly on the ESP32 to reduce the load on the IoT platform. This could include predictive analytics for estimating future vehicle locations based on historical data.
    - Utilize machine learning techniques for anomaly detection, such as detecting unauthorized vehicle usage or abnormal driving behavior.
    - Implement firmware-over-the-air (FOTA) updates capability to remotely update the ESP32 firmware,

ensuring that the tracker units can receive new features or security patches without manual intervention.

3. **Integration with Advanced IoT Platform:**
    - Select an IoT platform with built-in support for edge computing to offload some processing tasks from the ESP32 devices. This can reduce latency and improve scalability.
    - Utilize blockchain technology for securely storing and sharing vehicle tracking data, ensuring data integrity and transparency, which is crucial for applications like supply chain logistics or vehicle rental services.
    - Integrate with third-party APIs for additional functionality, such as weather data to optimize route planning or vehicle maintenance services for proactive maintenance scheduling.

4. **Data Visualization and User Experience:**
    - Develop a responsive and intuitive web-based dashboard with real-time updates for tracking multiple vehicles simultaneously. Include features like customizable alerts, live traffic updates, and geofencing capabilities.
    - Create a mobile application for on-the-go access to vehicle tracking information, enabling fleet managers or vehicle owners to monitor their assets from anywhere.
    - Implement a comprehensive reporting system with customizable reports for analyzing vehicle performance, fuel efficiency, driver behavior, and more.

5. **Security and Privacy Measures:**

   - Implement end-to-end encryption for data transmission between the ESP32 devices and the IoT platform to protect sensitive information from unauthorized access.
   - Utilize secure authentication mechanisms, such as OAuth or JWT (JSON Web Tokens), to ensure that only authorized users can access the tracking system.
   - Adhere to data privacy regulations such as GDPR (General

Data Protection Regulation) or CCPA (California Consumer Privacy Act) by providing users with control over their data and obtaining explicit consent for data collection and processing.

6. **Scalability and Reliability:**

   - Design the system with scalability in mind, allowing it to handle a large number of vehicles and users without compromising performance.
   - Implement redundancy and failover mechanisms to ensure uninterrupted operation in case of network failures or hardware malfunctions.
   - Monitor system health and performance metrics in real-time, leveraging tools like Prometheus and Grafana for proactive maintenance and troubleshooting.

By incorporating these advanced features and techniques, you can create a cutting-edge IoT-based vehicle tracking system that not only provides real-time location data but also offers advanced analytics, enhanced security, and a seamless user experience.

Conclusion:

- utilizing an ESP32 GPS tracker for IoT-based vehicle tracking offers a robust and cost-effective solution for real-time location monitoring. This system leverages the capabilities of the ESP32 microcontroller in conjunction with GPS technology to provide precise location data, which can be transmitted over the internet to a central server or database. By integrating these technologies, users can achieve enhanced visibility and control over vehicle movements, significantly improving operational efficiencies and security measures. This IoT-based tracking system is not only versatile but also scalable, making it suitable for a wide range of applications from personal vehicle monitoring to fleet management in commercial settings.