Information Technology

Report:

Real-time GPS Tracking System Using C Language

Presented by:

Vinit Patil(B24IT1040)
Sarthak Patil(B24IT1044)
Gaurav Jadhav(B24IT1054)
Shreyash Mandlapure(B24IT1130)

TABLE OF CONTENTS

ABSTRACT

This project focuses on developing a **Real-time GPS Location Tracking System** utilizing **C language** for embedded programming and **ESP8266 microcontroller** for wireless communication. The goal is to accurately track the user's location in real time, providing continuous updates of the **latitude** and **longitude** coordinates. The data is then transmitted over **Wi-Fi** to a server, and displayed on **Google Maps** through a URL. The system can be applied in diverse fields like **fleet management**, **asset tracking**, and **personal safety**. The project also explores integrating GPS decoding, wireless transmission, and web-based data display to offer a user-friendly interface for location tracking.

Key Features:

- **Real-time GPS tracking** through a **GPS module**.
- **ESP8266 microcontroller** for Wi-Fi connectivity.
- Transmission of GPS data to **Google Maps** via an internet connection.
- User-friendly display and easy access to location information on Google Maps.

Chapter 1 - Introduction

This project involves designing a **GPS tracking system** using the **ESP8266 microcontroller** and the **NEO 6M GPS module**. The aim is to create a system that can transmit **GPS coordinates** in real-time over a **Wi-Fi network**, enabling users to track their location on **Google Maps** through a URL generated by the ESP8266. This project is useful for applications that require **real-time tracking**, such as **vehicle** or **asset tracking**.

Project Objectives:

- To provide a real-time GPS tracking solution using **Embedded C** programming.
- To display the location on **Google Maps** by generating dynamic URLs with the latest coordinates.
- To implement **Wi-Fi communication** for seamless data transfer.

Applications:

- **Fleet management**: Tracking vehicles, transportation, and delivery systems.
- **Asset management**: Monitoring assets like equipment, machines, or containers.
- **Personal safety**: Tracking individuals in hazardous environments or for emergency purposes.

Problem Statement:

- Traditional tracking systems can be complex and expensive. This project provides an affordable solution with simple hardware and software, utilizing widely available components.

Chapter 2 – Analysis

In this project, the ESP8266 microcontroller is used to manage Wi-Fi connectivity and serve as the communication link between the GPS module and a remote client. The NEO 6M GPS Module provides latitude and longitude coordinates, which are encoded by the TinyGPS++ library. The GPS coordinates are then embedded into a Google Maps URL to visualize the location.

System Requirements

1. **Hardware**:
    - **ESP8266 Microcontroller**: A low-cost microcontroller with built-in Wi-Fi capability, allowing it to connect to the internet.
    - **NEO 6M GPS Module**
    - **GPS Module (ESIMP)**: This module communicates with the ESP8266 and provides data about the current geographical coordinates, including **latitude** and **longitude**.
2. **Software**:
    - **Arduino IDE**: An open-source software platform used for writing and uploading the code to the microcontroller. The libraries required are **TinyGPS++, ESP8266, Software**
    - **Embedded C**: The language used for programming the microcontroller.

    - **Wi-Fi Network**: Required for internet connectivity, enabling the data to be transmitted from the microcontroller to the web server.

Challenges

- **GPS Data Accuracy**: GPS modules are prone to interference, such as signal loss or reduced accuracy when used indoors or in areas with poor satellite visibility.
- **Wi-Fi Connectivity**: Ensuring stable internet connection between the ESP8266 and the server is crucial for real-time updates.
- **Data Formatting**: Converting the raw GPS data into a format that is understandable and usable by the web browser or client.

Key Features

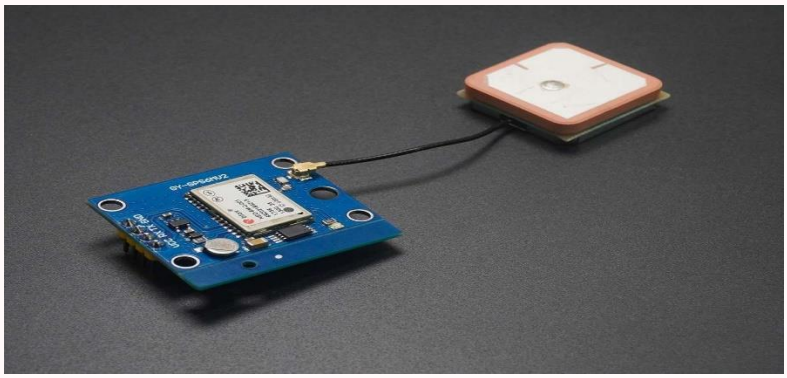
- **Real-time Location Tracking**: Continuous GPS data processing to provide accurate location.
- **Wi-Fi Communication**: Data transmission through Wi-Fi for easy access.
- **Google Maps Integration**: Provides a live URL displaying the exact location.

Chapter 3 - Design

1. System Block Diagram

This system's design encompasses the following components:

1. Hardware Setup:
   The NEO 6M GPS module is connected to the ESP8266 microcontroller via the SoftwareSerial library for dedicated GPS communication. The ESP8266 connects to the Wi-Fi network for transmitting the GPS data.

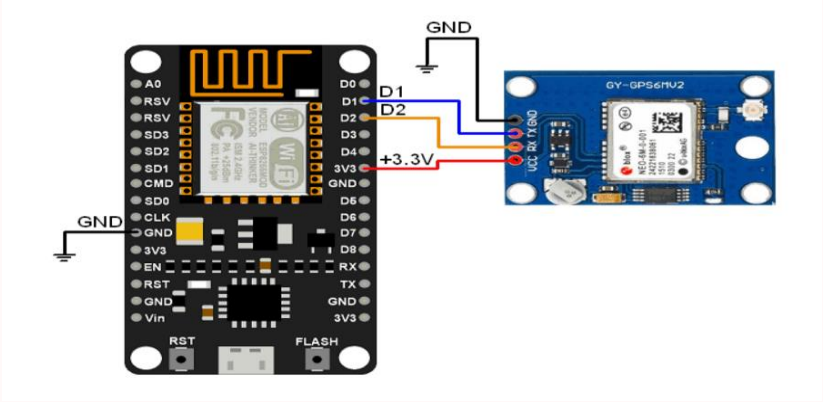


(GPS Module)

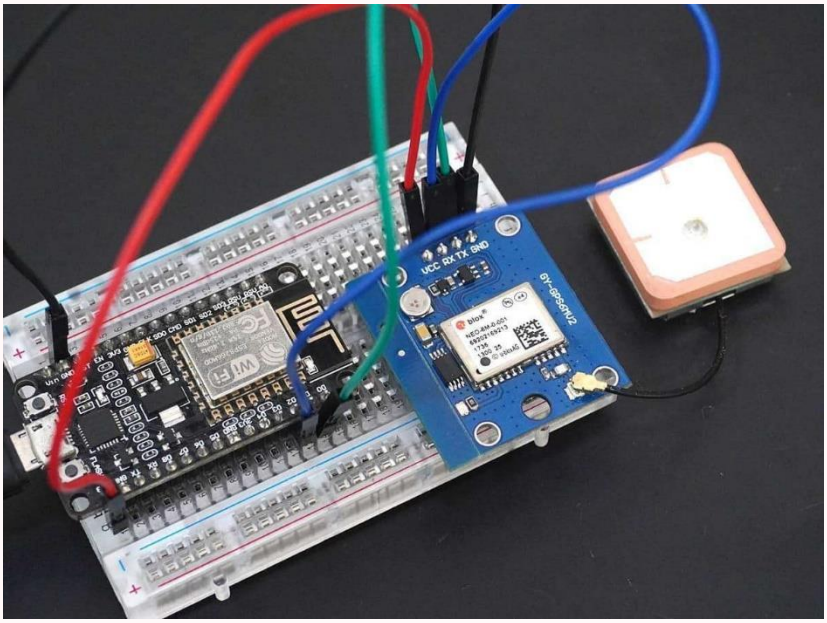
(ESP8266)

2. Data Flow:
   GPS data (latitude and longitude) is received by the ESP8266, encoded with TinyGPS++, and sent to a remote client using the WiFiServer class.

3. URL Generation:
   GPS coordinates are formatted as a Google Maps URL for easy visualization.



(Circuit Diagram)



(Wiring)

Chapter 4 - Implementation

The code implementation uses the ESP8266, NEO 6M GPS module, and TinyGPS++ library to retrieve, parse, and transmit location data. WiFi connectivity is established using the ESP8266's inbuilt capability, and location data is served over a WiFi network.

```
#include <SoftwareSerial.h>
#include <TinyGPS++.h>
#include <ESP8266WiFi.h>

#define RX_PIN 4  // GPS TX to ESP8266 RX (GPIO 4)
#define TX_PIN 5  // ESP8266 TX to GPS RX (GPIO 3, usually not used)

// Wi-Fi credentials
const char* ssid = "GoJo";        // Replace with your Wi-Fi SSID
const char* password = "gojo@123"; // Replace with your Wi-Fi password

// Create a SoftwareSerial object
SoftwareSerial GPS_SoftSerial(RX_PIN, TX_PIN);
TinyGPSPlus gps;
WiFiServer server(80); // Create a web server on port 80

void setup() {
    Serial.begin(115200);           // Start Serial Monitor
       // Start SoftwareSerial for GPS
    WiFi.begin(ssid, password);     // Connect to Wi-Fi

    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }

    Serial.println("Connected to WiFi");
    server.begin();                 // Start the server
    Serial.println("Server started");
    GPS_SoftSerial.begin(9600);
}

void loop() {
    while (GPS_SoftSerial.available()) {
        gps.encode(GPS_SoftSerial.read());
        if (gps.location.isUpdated()) {
            double latitude = gps.location.lat();
            double longitude = gps.location.lng();
            String url = "http://maps.google.com/?q=" + String(latitude,6) + "," + String(longitude,6);
            Serial.println(url); // Print URL to Serial Monitor

            // Handle client connections
            WiFiClient client = server.available();
            if (client) {
                Serial.println("New client connected");
                String response = "HTTP/1.1 302 Found\r\n";
                response += "Location: " + url + "\r\n";
                response += "Connection: close\r\n";
                response += "\r\n";
                client.print(response);
                delay(100); // Give the client time to receive the data
                client.stop(); // Close the connection
            }
        }
    }
}
```

Detailed Code Explanation:

1. **Libraries Used:**
    - **SoftwareSerial.h: Enables GPS data transfer on RX and TX pins.**
    - **TinyGPS++.h: Decodes GPS data, extracting latitude and longitude coordinates.**
    - **ESP8266WiFi.h: Manages Wi-Fi connection for data transmission.**
2. **Data Processing:**
    - **The gps.encode() function reads and decodes GPS data, updating coordinates in real time.**
3. **URL Generation:**
    - **Formats latitude and longitude into a Google Maps URL for easy location visualization.**
4. **Client Communication:**
    - **A WiFiClient object handles incoming client connections. When connected, the system provides the current location via the Google Maps URL.**

Algorithm:

S1) START

S2) CONNECT TO WIFI NETWROK

S3) CONNECTED TO WIFI NETWORK

S4) GET LATITUDE AND LONGITUDE FROM GPS MODULE

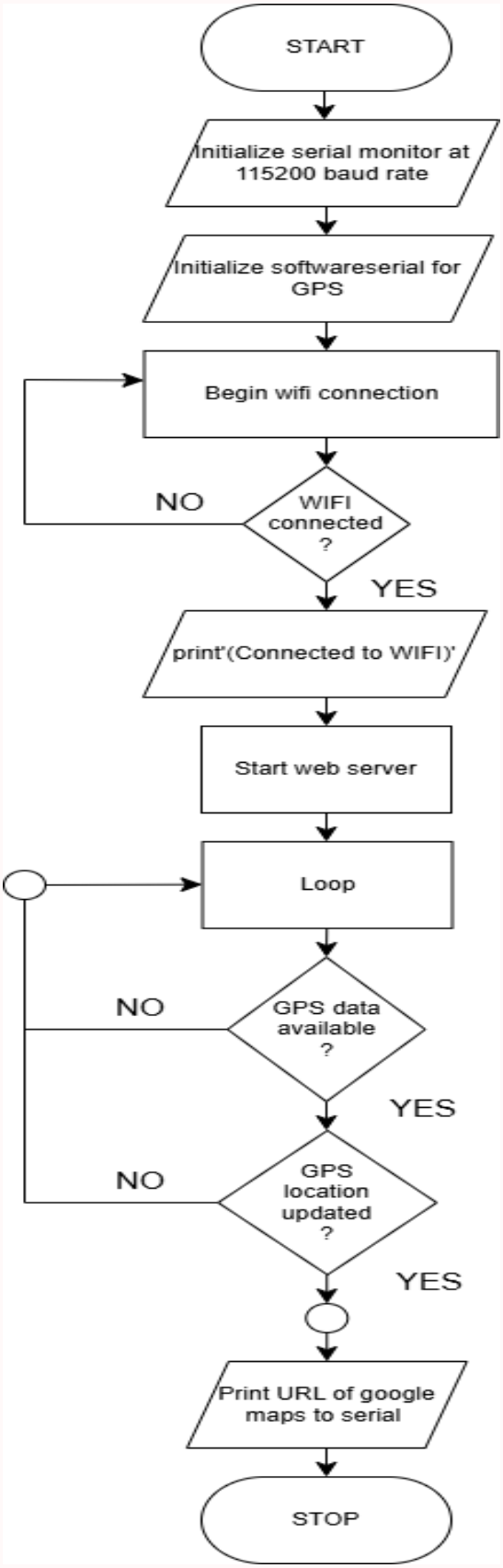
S5) INPUT THE DATA TO MICROCONTROLLER

S6) GENERATE A URL LINK FOR GOOGLE MAPS USING THE LATITUDE AND LONGITUDE

S7) PASTE THE LINK ON THE BROWSER

S8) END

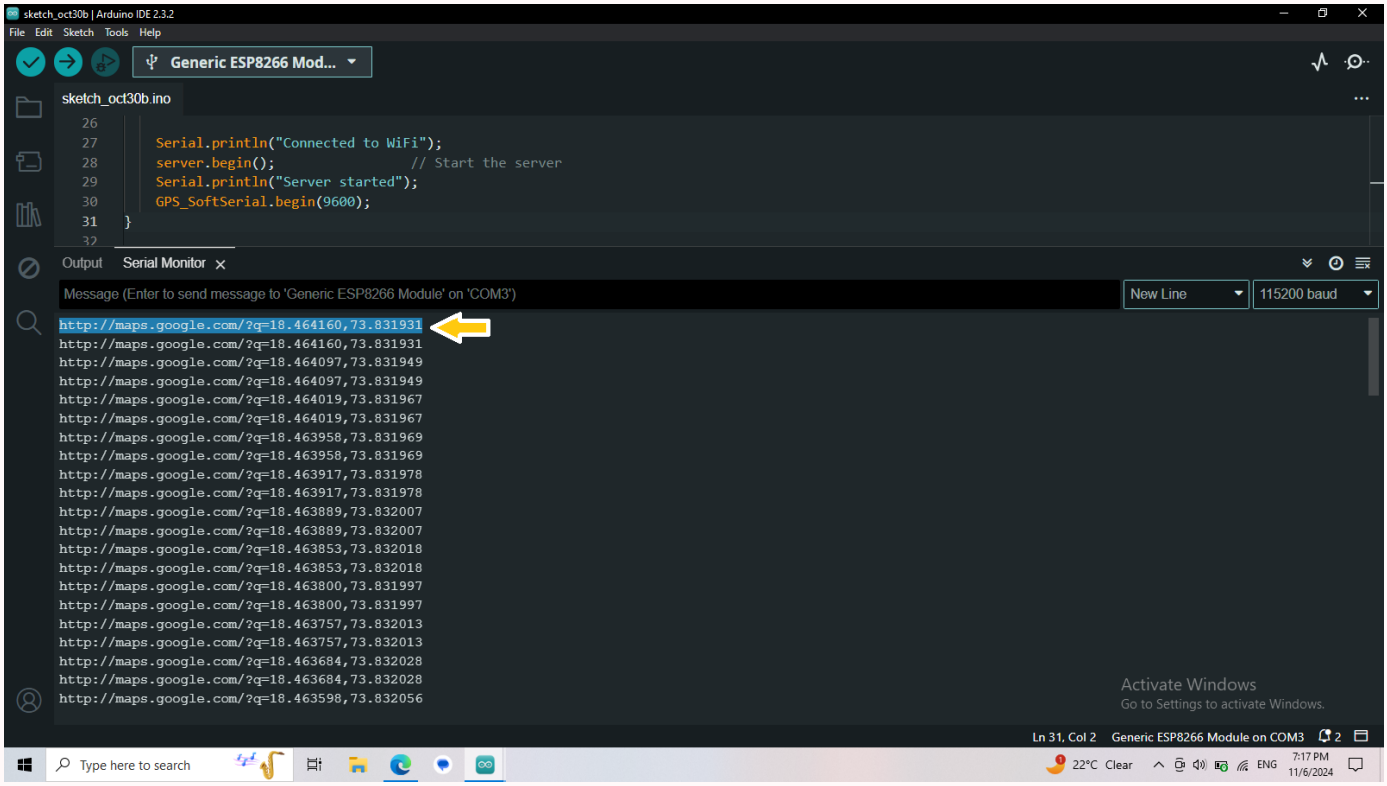
Flowchart


START
Initialize serial monitor at 115200 baud rate
Initialize softwareserial for GPS
Begin wifi connection
NO
WIFI connected ?
YES
print'(Connected to WIFI)'
Start web server
Loop
NO
GPS data available ?
YES
NO
GPS location updated ?
YES
Print URL of google maps to serial
STOP

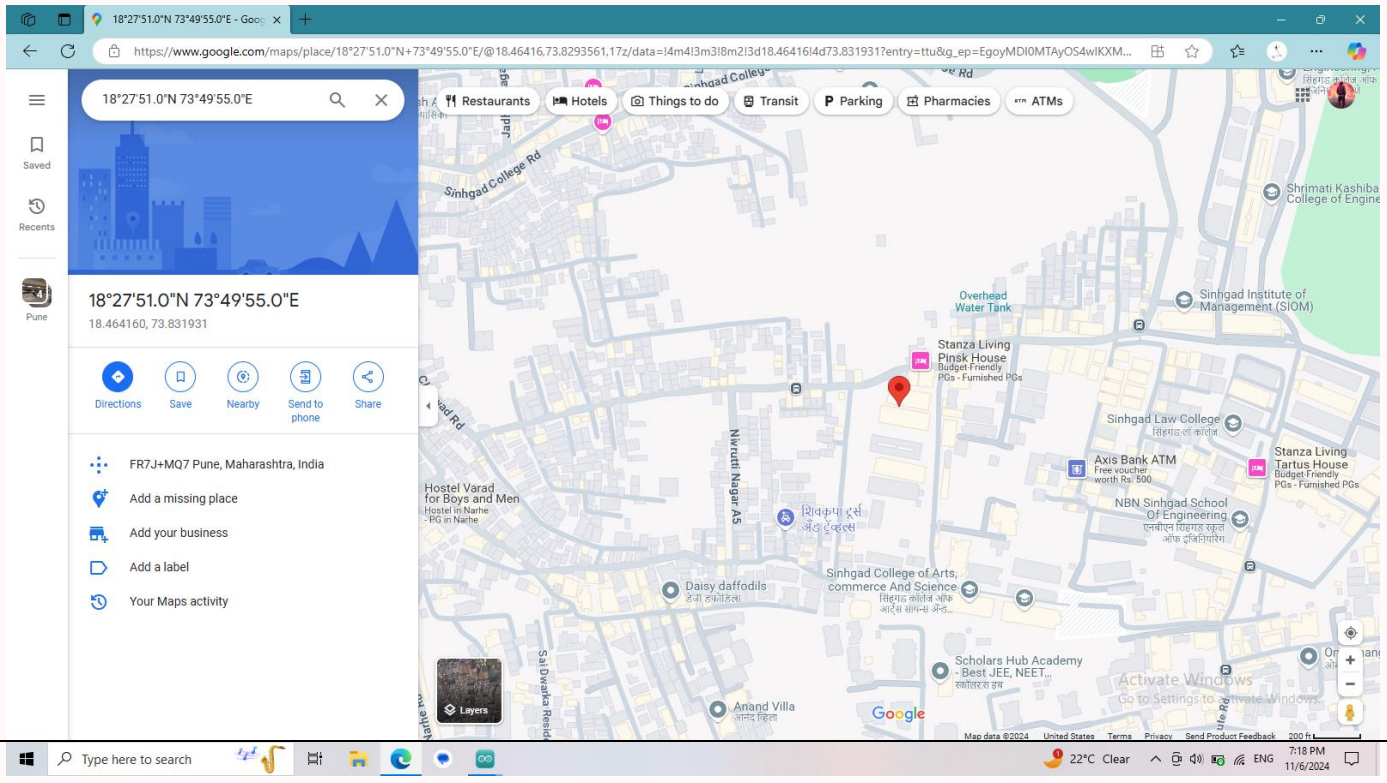
Chapter 5 – Images

(Screenshot for link)

Link: http://maps.google.com/?q=18.464160,73.831931



Screenshot of location(output)



7) Chapter 6 - Conclusion

In conclusion, the **Real-time GPS Tracking System** successfully implements a low-cost and efficient solution for location tracking using the **ESP8266 microcontroller** and **GPS module**. The system provides real-time GPS coordinates and generates a Google Maps URL for displaying the location, which can be accessed by any web browser. This project demonstrates the effective use of **Embedded C** for embedded systems and highlights the advantages of using widely available components for building cost-effective tracking solutions.

Future Enhancements:

- **Power Optimization**: Improve the energy efficiency of the GPS and Wi-Fi modules for longer battery life in portable applications.
- **Geofencing**: Implement geofencing capabilities to send notifications when the tracked device enters or leaves predefined areas.