Real-Time Public Buses Tracking System Using GNSS, GPRS and Mobile Application

Hemant Kumar Mahatara[a], Dhiraj Chaurasiya[b]

[a]Department of Geomatics Engineering, Institute of Engineering Pashchimanchal Campus Pokhara, Nepal
[b]Department of Electronics and Computer Engineering, Institute of Engineering Pashchimanchal Campus Pokhara, Nepal

KEYWORDS

ESP-WROOM-32
GSM/GPRS
GPS
Smart Buses Tracking System(SBTS)
IoT
Vehicle Tracking
Mobile App Development
Server
SparkFun GPS Breakout
SAM-M8Q

ABSTRACT

An electronic device installed in a car that allows its owner or a third party to track the location of the vehicle is called a vehicle tracking system. Vehicles such as public buses are essential for traveling passengers from one place to another. Delays in the buses' arrival at the bus station can cause a number of problems for passengers. The Internet of Things (IoT) is a rapidly developing technology that links and integrates a range of devices to produce more useful data. This paper proposed to design a vehicle tracking system which is called Smart Buses Tracking System (SBTS) that works using GPS and GSM/GPRS modem is an android application controlled by ESP-WROOM-32, which would be the cheapest source of vehicle tracking. The device that is being used to track the signal of the satellite is SparkFun GPS Breakout - Chip Antenna, SAM-M8Q (Qwiic) is a high quality GPS board with equally impressive configuration options. The SAM-M8Q is a 72-channel GNSS receiver, meaning it can receive signals from the GPS, GLONASS, and Galileo constellations which increases precision and decreases lock time. The SAM-M8Q GPS Breakout can also be automatically detected, scanned, configured, and logged using the OpenLog Artemis datalogger system. The Tracking Unit, Server, and Android application are the three primary components of SBTS. The Tracking Unit is installed within a buses to detect the speed and location of the vehicle before uploading the data to the server over a GSM network. This system enables the passengers to observe and track the vehicle and find out about vehicle current position. This proposal has significant application for vehicle security, ambulances, municipality garbage truck, public buses tracking and private drivers.

1. INTRODUCTION

The delay of public buses is most common in Nepal due to improper rules, regulations, and the monitoring department. Due to the delay of the buses, the passengers face many problems. Most of the students, employees, tourists, and locals use the public bus as their main mode of transportation since the cost is most effective for everyone. But due to the delay in the arrival of buses at bus stops, people are now preferring their own transportation vehicles like bikes, scooters, and cars, which eventually increase the traffic on the road and consume more fuel, which eventually increases the use of fuel. Since the condition of the road in most places is narrower for two lanes, and due to the increase in traffic, it may cause accidents. More consumption of fossil fuels will eventually cause air and noise pollution. Hence, vehicle tracking is one way to reduce these problems by knowing the real-time location of the vehicles. This research introduces a vehicle tracking system using the Global Positioning System (GPS) for positioning, the General Packet Radio System (GPRS) for data transmission, and an Android application for location display.

Integrating tracking systems into probe vehicles capture traffic data in real-time, aiding traffic management and identifying traffic jams in a timelier manner. In the event of an emergency, these devices can automatically alert rescue personnel as to the position of a vehicle, reducing response times and potentially saving lives. By linking up with anti-theft systems, these systems add to security features by allowing for the early identification and recovery of stolen vehicles. Fleet management is also aided by the devices by allowing companies to keep track of their vehicles' whereabouts and statuses, increasing operational efficiencies. And public transportation systems can decrease passenger wait times by improving the efficiency of routes and schedules.

Our purposed system offers real-time tracking of public vehicles using a client-server model. Our client is an embedded device with a GPS/GPRS module to identify device location information that is periodically transmitted to a server. On the server, these data are stored in the database offered by the server. By using the Structured Query Language (SQL), we can easily obtain the data from the database with the help of the Application Programming Interface (API). An API, or application programming interface, is a collection of tools and protocols that facilitates communication between

Crossponding Author: Tel: +977-9822975366
Email Address: mahatara.hemant9@gmail.com

various software programs and the database. It provides for the smooth integration and communication between various systems or services by defining the processes and data formats that applications may use to request and exchange information. Now, as soon as the mobile application opens, it makes a call to an API to establish communication before sending data from the database to the mobile application. The vehicle's current location is now displayed by the application over the MapLibre basemap.

The rest of this paper is organized as follows: We review related technology in Section II. The proposed system's design and implementation details are covered in Section III. It includes the design of hardware and software for devices used to identify and send the location of the vehicle to a remote server. It also goes over how the hardware was designed and how a web-based user interface was created. Results of system testing are covered in Section IV. The paper's conclusion and future work are presented in Section V.

2. BACKGROUND

A. GNSS Technology

Global Navigation Satellite System (GNSS) include constellations of Earth-orbiting satellites that broadcast their locations in space and time, of networks of ground control stations, and of receivers that calculates ground positions by trilateration. GNSS include two fully operational global systems, the United States' Global Positioning System (GPS), the Russian Federation's Global Navigation Satellite System (GLONASS), as well as the developing global and regional systems, namely Europe's Satellite Navigation System (GALILEO) and China's COMPASS/BeiDou, India's Regional Navigation Satellite System (IRNSS) and Japan's Quasi-Zenith Satellite System (QZSS). The satellite broadcasts two codes – the coarse acquisition (C/A) code, unique to the satellite, and the navigation data message.

In general GPS provides three types of measurements: Pseudorange, carrier phase, and Doppler. By pseudoranging, the GPS user measures an approximate distance between the GPS antenna and the satellite by correlation of a satellite transmitted code and a reference code created by the receiver. Four pseudorange observations are needed to resolve a GPS 3-D position. In practice there are often more than four satellites within view. A minimum of four satellite ranges are needed to resolve the clock-biases contained in both the satellite and the ground-based receiver. Thus, in solving for the X-Y-Z coordinates of a point, a fourth unknown (i.e. clock bias ~Δt) must also be included in the solution. The solution of the 3-D position of a point is simply the solution of four pseudorange observation equations containing four unknowns, i.e. X, Y, Z, and Δt.

A pseudorange observation is equal to the true range from the satellite to the user plus delays due to satellite receiver clock biases and other effects.

$$R = p^t + c\,(\Delta t) + d$$

Where

R = observed pseudorange

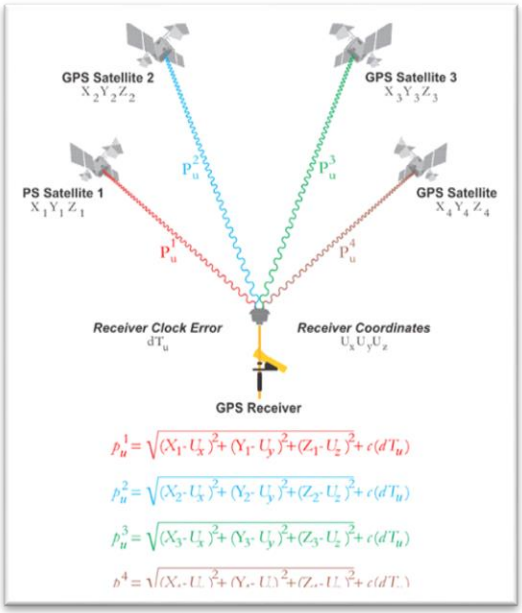

Fig 1: Working Principle of GNSS Technology

p^t = true range to satellite (unknown)

c = velocity of propagation

Δt = clock biases

d = propagation delays due to atmospheric conditions

SAM-M8Q GPS Breakout is chosen as the receiver of the GNSS signals which is responsible to accurately locate the points within the earth surface. The SAM-M8Q is a 72-channel GNSS receiver, meaning it can receive signals from the GPS, GLONASS, and Galileo constellations which increases precision and decreases lock time.

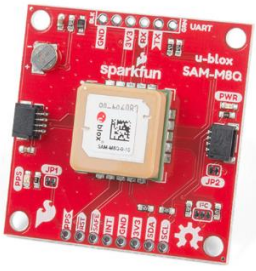

Fig 2: SparkFun GPS Breakout - Chip Antenna, SAM-M8Q (Qwiic)

B. GPRS Technology

General packet radio service (GPRS) is defined as a mobile communications standard that operates on 2G and 3G cellular

Crossponding Author: Tel: +977-9822975366
Email Address: mahatara.hemant9@gmail.com

networks to enable moderately high-speed data transfers using packet-based technologies GPRS Working. In essence, general packet radio service (GPRS) is a packet-switching technology that makes it possible to send data over mobile networks. Multimedia messaging services, internet connectivity, and other forms of data transmission are made use of this. GPRS cellphones, laptops, and portable devices with GPRS modems can all support GPRS. Up to 80 Kbps downstream bandwidths have been reported by subscribers. The second generation (2G) cellular network uses the global system for mobile communications (GSM) as its primary standard; GPRS is an enhanced version. Unlike GSM's short messaging service (GSM-SMS), which has a 160-byte message length limit, GPRS does not allow for this. While most networks run at about 35 kbps, GPRS has a theoretical maximum speed of 115 kbps. Unofficially, GPRS is sometimes referred to as 2.5G. It's a third-generation method of becoming accessible online.

Fig 3: GPRS/GSM 800L Module

C. *Mobile Application Development using React Native*

React Native is an open-source framework developed by Facebook for building mobile applications using JavaScript and React. It enables programmers to create mobile applications that work on both the iOS and Android platforms by utilizing React, a well-liked JavaScript library for creating user interfaces. Developing cross-platform mobile apps with a single codebase is made possible by React Native, which saves time and effort when compared to developing apps specifically for each platform.

The ability to display the geospatial features found on the earth earth's surface on a mobile device is made possible by the integration of geospatial data with the mobile device. The app can display any geospatial data over a basemap, including hospital, road, house, bus station, bus route, and many other features. MapLibre is the source of the basemap used in this application. MapLibre is an open-source library for interactive maps on the web. It is a fork of the Mapbox GL JS library, which is a JavaScript library for rendering interactive maps.

Maps can be integrated into React Native applications with the help of the well-liked third-party library react-native-maps. Through the use of the MapView component, which is available for both iOS and Android, developers can incorporate interactive maps into their mobile applications. GeoJSON (Geographic JavaScript Object Notation) is an open standard format designed for representing geographical features and their attributes. It is a text-based, lightweight data interchange format that uses the JavaScript Object Notation (JSON) syntax to encode geographic data. GeoJSON is a widely used protocol that many GIS (Geographic Information System) software systems support for transferring spatial data between web servers and web clients.

3. DESIGN AND IMPLEMENTATION OF TRACKING SYSTEM

This paper proposes the design of an embedded system that uses the Global Positioning System (GPS) and the Global System for Mobile Communication (GSM) to track and position of any vehicle. Our real-time tracking management system is an open system built with readily available commodity hardware and free and open source software. Our system is composed of three components, a GPS Tracking Device, a server, database and mobile application as shown in Figure 1. The device's exact location is seamlessly provided by the GPS tracking device, and a microcontroller is used to format and control it. In this system we used ESP-WROOM-32 as the microcontroller.ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. When a vehicle is located in a remote area, its position (latitude and longitude) is sent via a GSM modem. The latitude and longitude, which indicate the position of the vehicle, are continuously provided by the GPS modem. The GPS modem outputs a wide range of parameters, but only the NMEA data that is received by the device. We used SAM-M8Q, a 72-channel GNSS receiver, for GPS tracking because it can receive signals from the Galileo, GLONASS, and GPS constellations, increasing precision and reducing lock time.

A. *Smart Bus Tracking System (SBTS)*

In our project we have developed the device called Smart Bus Tracking System (SBTS) which is imbedded with the SparkFun SAM-M8Q GNSS receiver which is a GNSS receiver and can receive 72-channel meaning it can receive signals from the GPS, GLONASS, and Galileo constellations which increases precision and decreases lock time. The device also contain the General packet radio service (GPRS) module along with the microcontroller. In this project we have used ESP-WROOM-32 microcontroller which is really powerful. The main purpose of the microcontroller is to control the signal coming out from the SAM-M8Q GNSS receiver since the GNSS receiver receives the satellite data in NMEA format. NMEA (National Marine Electronics Association) is a standard communication protocol used by marine and terrestrial navigation systems to enable devices from different manufacturers to communicate with each other. The NMEA data format is particularly common in the context of GPS (Global Positioning System) receivers and other navigation devices. One of the example of NMEA sentences is $GNGGA which is given below.

$GNGGA 123519, 2815.42533, N, 08358.61071, E, 1, 08, 0.9, 984.4, M,-39.9, M,*47

Where,

Crossponding Author: Tel: +977-9822975366
Email Address: mahatara.hemant9@gmail.com

GGA	Global Positioning System Fix Data
123519	Fix taken at 12:35:19 UTC
2815.42533,N	Latitude 28^0 15.42533' N
08358.61071,E	Longitude $83^0$58.61071'E
1	Fix quality
08	Number of satellite being tracked
0.9	HDOP
984.4,M	Altitude, meters, above mean sea level
-39.9,M	Height of geoid(MSL) above WGS84 ellipsoid
*47	The checksum data always being with the *

Since such types of message is difficult to transmit through the SIM800L module so firstly microcontroller handle the satellite data and convert the data into transformable format. The SAM-M8Q had library which convert the NMEA message into latitude, longitude, speed, DOP, Number of satellite, elevation etc. After that this data is then format into JSON format. JSON, which stands for JavaScript Object Notation, is a lightweight data interchange format. It is easy for humans to read and write, and it is also easy for machines to parse and generate. JSON data is represented as key-value pairs in a hierarchical structure. JSON is commonly used for data exchange between a server and a web application, as well as for configuration files and other data storage purposes. Here is an example of JSON containing the location data provided by the device.


```
{
  "data": [
    {
      "id": "1319",
      "lat": "28.257350",
      "lng": "83.976493",
      "created_date": "2024-01-30 20:02:45"
    }
  ]
}
```


B. Database Preparation

It is crucial to develop a database which is used to store the data which is sent by the tracking device. The purpose of the database is to store, retrieve, and review and analyses the data. After formatting, the data is then sent to the database through the GSM/GPRS 800L module which is capable of transmitting the data wirelessly. The GSM 800L is imbedded with Subscriber Identity Module (SIM) which uses the local network to transmit the data. The SIM800L GSM/GPRS module is a miniature GSM modem that can be used in a variety of IoT projects. We can use this module to do almost anything a normal cell phone can do, such as sending SMS messages, making phone calls, connecting to the Internet via GPRS, and much more.

To make an HTTP request using a GSM/GPRS module such as the GSM800L, the process involves several key steps. Initially, the module must be powered up and initialized. Subsequently, a GPRS connection is established to enable mobile data communication. The configuration for the HTTP request follows, with parameters like the server URL, port, and any necessary headers or authentication information being set. The actual HTTP request is then composed, specifying the method (e.g., GET or POST), headers, and, if applicable, a request body. The request is sent using AT commands or a module-specific communication protocol. Following the transmission, the device awaits the HTTP response from the server. Upon receipt, relevant information is extracted and processed. Optionally, the GPRS connection can be terminated or resources released. This entire sequence allows devices equipped with GSM/GPRS capabilities, like the GSM800L, to interact with web servers, making it particularly useful for applications involving data exchange with remote servers or fetching information from the internet. Now in order to store the data we created a local server with the use of XAMPP. XAMPP, which stands for Cross-Platform, Apache, MySQL, PHP, and Perl, is a free platform that allows developers to test their code locally on their own computers. This platform provides the experience of having our own mini web server at home, compatible with both Windows (WAMP) and Linux (LAMP) environments. It is a safe space to experiment and perfect code before it goes live. With the use of GPRS/GSM 800L and XAMPP the data is then seamlessly stored in the local server database. Now in order to publically accessible the data stored in the database we generate an API.

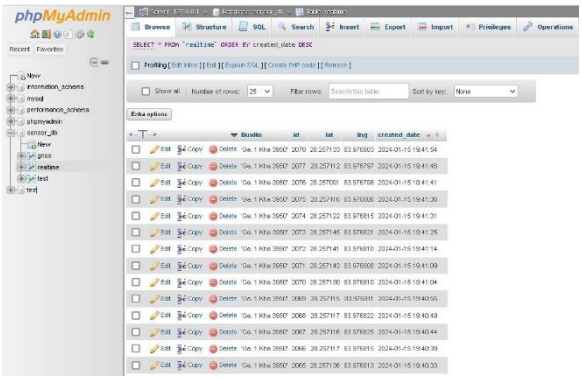

Fig 4: Database Preparation using XAMPP

C. APIs Generation

In the MeroBus project, the implementation of an Application Programming Interface (API) plays a pivotal role in overcoming the communication barrier between the mobile app and the underlying database. As users enter their source and destination within the app, the API acts as an intermediary, facilitating seamless communication with the database to retrieve and deliver pertinent information. Specifically, when a user inputs source and destination details, the API takes these parameters and queries the database for the corresponding bus fare data. Once retrieved, the API efficiently

Crossponding Author: Tel: +977-9822975366

Email Address: mahatara.hemant9@gmail.com

delivers this information back to the app, enabling users to access real-time bus fare details for both the general public and students. The significance of the API lies in its ability to serve as a bridge, allowing different software components to communicate and share data. In the context of MeroBus, the API streamlines the interaction between the user interface of the app and the database housing crucial fare information. Without this intermediary layer, direct communication between the app and the database would be challenging and potentially insecure. Moreover, the API encapsulates the complexity of the database queries, presenting a simplified and standardized interface for the mobile app to request and receive information. This abstraction not only enhances security but also promotes modular design, facilitating future updates or modifications to the database structure without directly impacting the app's functionality.

In essence, the API serves as a facilitator for data exchange, ensuring the seamless retrieval of bus fare details for users. Its role in our project is paramount, contributing to the overall effectiveness of MeroBus by providing users with accurate and timely fare information, thereby enhancing their ability to plan and optimize their commutes in the Pokhara Valley.

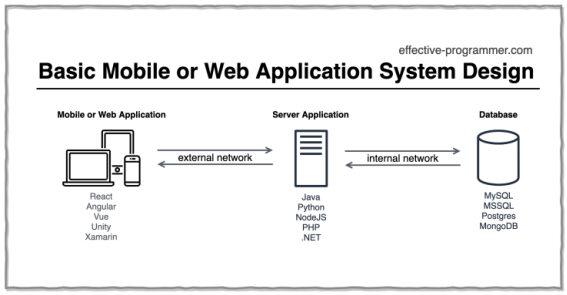

Fig 5: APIs Development for Mero Bus

So the device continuously update the latest location data with the database using GPRS/GSM 800L module and the API that is generated is used to interact with the database and the mobile application. The volume of data in the database is excessive because it is seamlessly uploaded to the server four times every second. We employed Structured Query Language (SQL) to solve this issue. We first sorted the data by ID so that the most recent information always appears at the top of the table, then we deleted every table except the last 10 locations, which remained in the database. In order for the SQL code to only keep the most recent 10 data when the application calls an API, it must first provide the most recent position data before deleting all other data.

So when we call an API,

http://192.168.254.16/gnss/test_data.php?action=retrieve

Then the SQL coded inside the test_data.php assign the current location using this code,

retrieveSql = "SELECT * FROM realtime ORDER BY id DESC LIMIT 1";

Subsequently, an additional SQL query is executed, eliminating all the data except for the most recent 10 position data.

deleteSql = "DELETE FROM realtime WHERE id < (SELECT id FROM realtime ORDER BY id DESC LIMIT 1 OFFSET 9)";

4. HARDWARE DESIGN

In order to achieve the real time tracking of the public bus we have developed a prototype Smart Bus Tracking Device (SBTD). This device is imbedded with SAM-M8Q which receives GNSS signal and gives the precise location of the point. Another device that is attached with this is GPRS/GSM 800L module which is capable not only for messaging and phone calls but it is also capable to do http request to the server. With the help of the cellular network and a SIM card inserted into the device it can communicate with the server and do http request which helps to send the position data through internet and store the position data into the database. Let's breakdown the structure into the two parts-connection between the SAM-M8Q and the ESP-WROOM-32 and the connection between the GPRS/GSM 800L module with the ESP-WROOM-32.

A. Connection between SAM-M8Q and ESP-WROOM-32

The SAM-M8Q module, produced by u-blox, is a Global Navigation Satellite System (GNSS) receiver commonly integrated into various applications requiring accurate positioning and navigation. The module operates within a specified voltage range, usually around 2.7V to 3.6V, and is designed for low power consumption, making it suitable for battery-powered applications. The SAM-M8Q typically communicates with a microcontroller or a host system using an I2C communication. I2C is a two-wire serial communication protocol using a serial data line (SDA) and a serial clock line (SCL). The protocol supports multiple target devices on a communication bus and can also support multiple controllers that send and receive commands and data. This allows the module to send and receive commands and data from the host system.

In order to check and verify the I2C connection we have used Arduino IDE software. The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. In order to check the connection, inside the Arduino IDE software,

```
#include <Wire.h>

void setup() {

  Wire.begin();

  Serial.begin(9600);

}

void loop() {

  // Request data from SAM-M8Q

  Wire.beginTransmission(SAM_M8Q_I2C_ADDRESS);
```

Crossponding Author: Tel: +977-9822975366

Email Address: mahatara.hemant9@gmail.com

```
    Wire.write(REGISTER_TO_READ);

    Wire.endTransmission();

     // Read data from SAM-M8Q

    Wire.requestFrom(SAM_M8Q_I2C_ADDRESS,
NUM_BYTES_TO_READ);

    while (Wire.available()) {

       int data = Wire.read();

       // Process the data as needed

    }

    delay(1000); // Adjust as needed based on the SAM-M8Q update
rate

}
```

The architecture of the connection between the SAM-M8Q and the ESP-WROOM-32 is shown below.

Sparkfun SAM-M8Q	ESP-WROOM-32
3V3	3V3
GND	GND
SDA	D21
SCL	D22

The connection diagram between the devices is shown below. In order to continuously operate the microcontroller, we have to supply a power supply of 5 volts. Since the battery we have is 7.4V, we have to first drop the voltage to 5V before we can send it to the microcontroller. For this, we have used a voltage drop-down regulator connected to the capacitor. The voltage regulator is essential for stabilizing the output voltage when lowering the voltage from 7.4V to 5V to power an ESP32 microcontroller. Since the ESP32 normally needs a certain voltage to function properly, this conversion is essential. Furthermore, a capacitor is frequently added at the regulator's output to improve stability by decreasing noise and smoothing out voltage fluctuations.

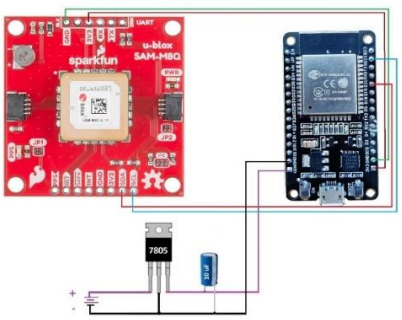

Figure 6: Schematic Diagram between GPS receiver and ESP-32

By assisting as a buffer, the capacitor reduces the impact of abrupt changes in current demand and promotes a more steady DC output. The ESP32 is then given the filtered and regulated 5V output, which satisfies its power needs.

B. Connection between GPRS/GSM 800L with microcontroller

Typically, serial communication using the UART (Universal Asynchronous Receiver-Transmitter) communication protocol is required to connect an ESP32 to a GSM module such as the SIM800L. The link between an ESP32 and the GSM SIM800L module is broken down as follows:

- *Hardware connections*

Connect the TX (Transmit) pin of the SIM800L module to the RX (Receive) pin of the ESP32, and vice versa. Because the SIM800L module lacks an on-board voltage regulator, we must select a power supply capable of supplying the SIM800L module within its 3.4V to 4.4V range (preferably 4.0V). Also the power supply must be capable of supplying at least 2A of surge current; otherwise, the chip will repeatedly reset. We use a 2A-rated DC-DC buck converter, such as the LM2596, with the output voltage set to 3.5V. These are far superior to linear voltage regulator modules in terms of efficiency.

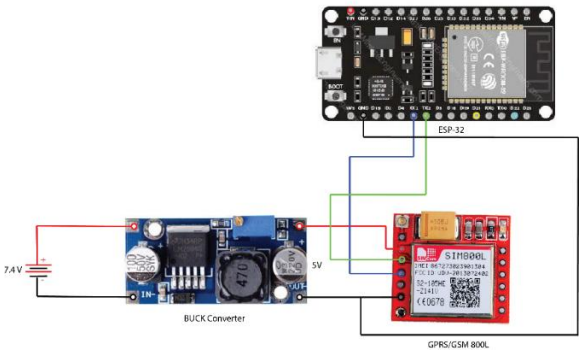

Figure 7: Schematic Diagram between GPRS/GSM and ESP-32

- *Serial Communication Configuration*

In the ESP32 firmware code, configure the serial communication for the UART interface. We use the Arduino IDE and the SoftwareSerial library.

```
#include <SoftwareSerial.h>

SoftwareSerial gsmSerial(RX_PIN, TX_PIN); // Define
SoftwareSerial object

void setup() {

   Serial.begin(115200); // Initialize serial for debugging

   gsmSerial.begin(9600); // Initialize GSM module serial
communication

}

void loop() {
```

Crossponding Author: Tel: +977-9822975366

Email Address: mahatara.hemant9@gmail.com

```
    // Your GSM communication code here
}
```

- *AT Commands*

Conventionally, AT commands are used to communicate with the GSM module. We use the established serial connection to transmit AT commands from the ESP32 to the SIM800L module. The GSM module can be used to send SMS, make calls, and manage network-related tasks thanks to AT commands.

```
void sendATCommand(String command) {
    gsmSerial.println(command);
    delay(500);
}
void setup() {
    // Initialization code
    sendATCommand("AT"); // Example AT command
}
void loop() {// Your main code}
```

The overall schematic diagram of the device is shown below.

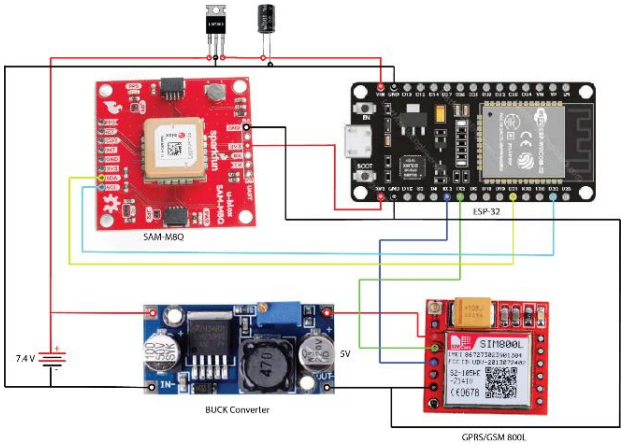

Figure 8: Schematic Diagram of Smart Bus Tracking System (SBTC)

5. RESULT AND DISCUSSION

Improving the effectiveness and user experience of the Pokhara Metropolitan City bus tracking system is the main goal of our project. The Android application seeks to give users up-to-date information about bus locations by integrating crucial features like precise bus numbers, accurate timings, and GPS tracking for accurate location. By enabling users to arrange their itineraries, calculate travel durations, and retrieve fare information, this feature ultimately decreases wait times and boosts user satisfaction. The addition of these features improves the efficiency and productivity of the public transportation system in addition to streamlining it. In the future, there exist ample opportunities for improvements that could further increase the usefulness of our project. A potential future direction is the development of an all-encompassing car monitoring system that makes use of GPS modules that are equipped with fast processors. The scope and impact of our project could be expanded by incorporating different forms of transportation, such as cars, jeeps, and taxis, into this system in addition to buses. In keeping with the overall strategy, we can also investigate incorporating a bus ticketing system into the application. This feature, which includes location-based fare calculations and convenient payment options through third-party applications like Eswea and Khalti, could make digital ticket purchases easier.

Moreover, the project can be expanded to serve individual car owners by providing a hardware component that allows for personal vehicle tracking, which can be helpful in theft situations and have potential uses for law enforcement. By extending the project to include private travel companies, it may be possible to give them a powerful tool for tracking their bus fleet and increasing operational effectiveness. By taking these potential improvements into account, our project not only meets Pokhara's immediate needs for public transportation but also establishes the framework for a flexible and scalable system with wider applications in the fields of security and transportation.

A. Android based Mobile App Development

In the current demo phase of the MeroBus mobile app, significant strides have been made towards enhancing the user experience and functionality of public transportation in the Pokhara Valley. The integration of a Low-Cost High-Accuracy GNSS Receiver System based on QZSS MADOCA Signal is underway, representing a pivotal advancement that will bring real-time bus location tracking to the fingertips of users. The app presently displays the user's current location and provides a comprehensive overview of the entire bus route network within the Pokhara Valley, setting the stage for a more informed and efficient commuting experience.

The upcoming integration of the GNSS Receiver System signifies a critical milestone for MeroBus, as it will enable users to track the real-time location of buses operating in the valley. This functionality aligns with the app's overarching goal of offering users dynamic and accurate information to optimize their travel plans. Moreover, the API integration within the app, allowing users to access detailed bus fare information between any two stations on a specific route, is a testament to MeroBus commitment to transparency and user convenience. The inclusion of fare details for students further underscores the app's dedication to catering to diverse user demographics. As MeroBus progresses from its demo phase, these features collectively contribute to creating a robust and user-centric platform that has the potential to significantly improve the public transportation landscape in Pokhara Valley.

Crossponding Author: Tel: +977-9822975366
Email Address: mahatara.hemant9@gmail.com

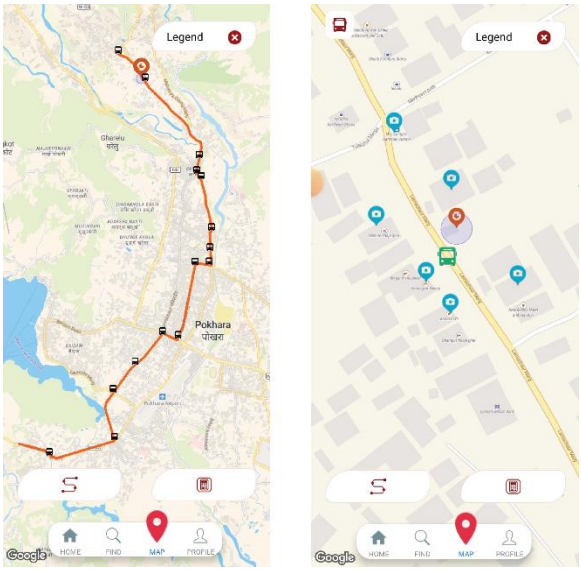

Figure 9: Mero Bus App Demo

B. Smart Bus Tracking System (SBTS)

In the beautiful Pokhara Valley of Nepal, the Smart Bus Tracking System (SBTS) is a full of features and advanced tool for tracking public buses in real time. The SAM-M8Q GNSS receiver for precise positioning, a buck converter for effective voltage regulation, a GPRS/GSM module for smooth communication, and a dedicated battery for continuous power supply are all integrated into this advanced system. Real-time monitoring of bus movements by passengers and transit authorities is made possible by the precise location tracking provided by the SAM-M8Q GNSS receiver. By optimizing the power supply, the buck converter increases energy efficiency and increases the device's useful life. Continuous communication is made possible by the GPRS/GSM module, which also allows the device to send bus location data to a central server. The integration of a reliable battery ensures continuous functionality, especially in areas with intermittent power supply. The Smart Bus Tracking System thus represents a robust and efficient solution for enhancing public transportation management, providing stakeholders with valuable insights into bus movements for improved service planning and passenger experience in the Pokhara Valley.

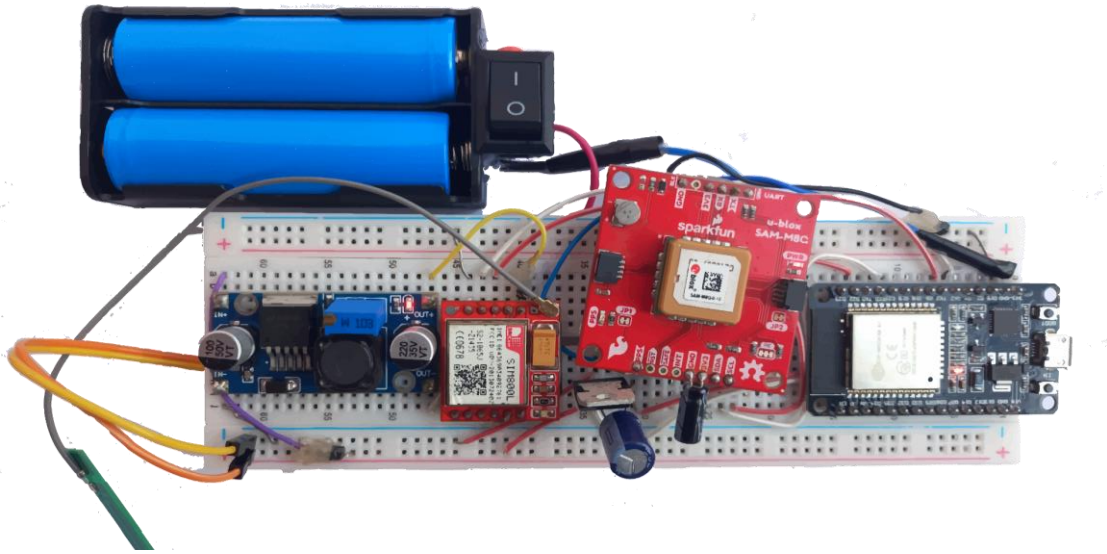

Figure 10: Smart Bus Tracking System (SBTS)

REFERANCES

- Maurya, K., Singh, M., & Jain, N. (2012). Real time vehicle tracking system using GSM and GPS technology-an anti-theft tracking system. *International Journal of Electronics and Computer Science Engineering*, *1*(3), 1103-1107.
- Zhu, H., Li, M., Zhu, Y., & Ni, L. M. (2008). Hero: Online real-time vehicle tracking. *IEEE Transactions on Parallel and distributed Systems*, *20*(5), 740-752.
- Khin, J. M. M., & Oo, N. N. (2018). Real-time vehicle tracking system using Arduino, GPS, GSM and web-based technologies. *International Journal of Science and Engineering Applications*, *7*(11,433-436).
- Chadil, N., Russameesawang, A., & Keeratiwintakorn, P. (2008, May). Real-time tracking management system using GPS, GPRS and Google earth. In *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology* (Vol. 1, pp. 393-396). IEEE.

Crossponding Author: Tel: +977-9822975366

Email Address: mahatara.hemant9@gmail.com