EV TRACKER

PROJECT REPORT

Submitted By

UMESH KANNA K B – ECE

PRANEET R - ECE

BALAPRIYAN S - ECE

HARIPRASATH N M – ECE

JAYAPRASATH S V – ECE

ISHVA S - ECE

CEG TECH FORUM

COLLEGE OF ENGINEERING GUINDY,
ANNA UNIVERSITY: CHENNAI 600025

FEBRUARY 2025

CEG Tech Forum

An ISO 9001:2015 certified organisation

BONAFIDE CERTIFICATE

Certified that this project report **EV TRACKER** is the bonafide work of **UMESH KANNA K B , BALAPRIYAN S, PRANEET R , HARIPRASATH N M, ISHVA S, JAYAPRASATH S V** who carried the project work under my supervision.

Christ Samuel H	Madeshwar R	Sujith Muralidharan
Student Director	Student Director	Student Director

Dr. V. Bhanumathi

Staff Coordinator

Associate Professor

Department of ECE

College of Engineering
Guindy

Anna University
Chennai–600025

ANNAUNIVERSITY
CHENNAI 600025

Degree:	Bachelor of Engineering
Organization:	CEG TECH FORUM
Name of the Project:	EV TRACKER
Name of the Students:	Umesh Kanna K B
	Praneet R
	Hariprasath N M
	Balapriyan S
	Jayaprasath S V
	Ishva S
Name and designation of Guide:	Dr. V. Bhanumathi
	Associate Professor,
	Department of Electronics and
	Communication Engineering
	College of Engineering Guindy,
	Anna University, Chennai-25.

ACKNOWLEDGEMENT

We extend our gratitude to Dr. V. Bhanumathi , Associate Professor, Department of Electronics and Communication Engineering, for her guidance and encouragement throughout this project. His expertise and insights were invaluable in shaping our approach and execution.

We sincerely thank CEG Tech Forum (CTF) for providing us with the platform, resources, and collaborative environment to undertake this project. The forum has been instrumental in fostering innovation and hands-on technical learning.

We take this opportunity to express gratitude for the valuable opportunity and support received during the course of our project titled "EV Tracking System Using ESP32, Ultrasonic Sensors, and ESP-NOW Communication".

This project was carried out independently as a self-driven initiative. We successfully integrated 13 ultrasonic sensors across two ESP32 boards, achieving wireless data communication using the ESP-NOW protocol. This setup demonstrated real-time distance monitoring and internal communication, which is crucial for EV tracking and obstacle detection applications.

Although we initially aimed to incorporate a SIM7672S 4G LTE module for transmitting GPS coordinates and HTTP requests to a backend server, we encountered hardware-level challenges and could not fully realize that part of the project within the expected timeframe. Nevertheless, this experience provided us with significant insights into embedded system integration, communication protocols, and real-world troubleshooting.

This project has been a valuable learning experience, enriching our knowledge of IoT systems, sensor networks, wireless communication, and embedded design. We are confident that the skills and lessons gained from this endeavor will be a strong foundation for our future engineering pursuits.

A special acknowledgment to the Student Directors of CTF – Christ Samuel, Sujith Muralidharan, and Madeshwar whose leadership and coordination ensured the smooth execution of this project. Their support and direction played a key role in overcoming challenges and refining our work.

TABLE OF CONTENTS

ABSTRACT

The rise of electric vehicles (EVs) has created a growing demand for intelligent tracking and safety systems that can operate efficiently in real time. This project, titled "EV Tracking System Using ESP32" aims to develop a cost-effective and reliable prototype for obstacle detection and intra-vehicle data transmission. The system leverages ultrasonic sensors for environmental awareness and utilizes the ESP32 microcontroller's ESP-NOW protocol for wireless communication between distributed nodes within the vehicle.

A total of 13 ultrasonic sensors were strategically interfaced with two ESP32 microcontrollers, with 7 sensors on one board and 6 on the other. The primary objective was to create a decentralized setup where both microcontrollers could collect, process, and share sensor data without reliance on external Wi-Fi or internet infrastructure. The ESP-NOW protocol, being a connectionless, low-power communication protocol developed by Espressif, proved ideal for fast, short-range data exchange in embedded IoT applications. This enabled a seamless and real-time flow of sensor data between nodes, paving the way for effective obstacle detection and monitoring.

The initial project scope also included the integration of a SIM7672S 4G LTE module for capturing GPS coordinates and sending the data to a remote server through HTTP POST requests. This was intended to enhance the tracking capabilities of the system by adding remote location reporting functionality. However, due to hardware integration challenges, signal issues, and limited documentation for this specific module, the GPS and internet-based data transmission component could not be successfully implemented within the timeframe. Despite extensive debugging and testing, the SIM7672S module failed to establish a stable connection with the ESP32 for consistent GPS data

acquisition and HTTP communication.

The success of the ultrasonic sensor network and ESP-NOW communication demonstrated the system's potential in creating a modular, scalable, and low-power tracking solution for EV applications. Moreover, the project's modular nature allows easy future upgrades, including successful integration of 4G or GPS modules, cloud communication, and advanced analytics.

In summary, while the project did not fully realize its long-range communication objectives, it succeeded in implementing a robust intra-vehicle sensor network. The experience offered significant insights into embedded system design, real-time wireless communication, and practical limitations in integrating cellular IoT modules. The groundwork laid by this project opens avenues for further research and development toward a comprehensive smart EV tracking and safety system.

1. INTRODUCTION

1.1 Overview of EV Tracking Systems

Electric Vehicle (EV) tracking systems enable real-time monitoring of vehicle location and status using GPS, sensors, and wireless communication protocols. In campus settings, these systems help manage electric shuttles or transport carts by providing location and availability data. Wireless communication plays a crucial role in transmitting this data between devices and to central servers or user applications without the need for physical infrastructure.

1.2 Motivation for the Project

Campus EVs are widely used for student and staff transport, but often lack intelligent tracking and monitoring. Additionally, the absence of wireless communication limits real-time updates and efficient system coordination. This project is motivated by the need for a smart, wireless communication-based tracking system that is low-cost, scalable, and optimized for campus operations.

1.3 Problem Statement

Current campus vehicles operate without an integrated system to monitor their location or seating capacity in real-time. Moreover, most traditional tracking systems rely on mobile networks (GSM), which may be costly or unnecessary within a limited campus range. This project proposes a wireless communication-based solution using ESP32 and ESP-NOW, allowing peer-to-peer data transfer without relying on external internet or SIM-based connectivity.

1.4 Objectives of the Project

- Implement real-time EV tracking within a campus using GPS modules.
- Detect seat occupancy using ultrasonic sensors for better load monitoring.
- Enable wireless peer-to-peer data transmission using ESP-NOW between ESP32 modules.
- Upload sensor and location data to a remote server via Wi-Fi for user access.
- Develop a mobile or web interface to display EV availability and status.

1.5 Scope and Limitations

Scope:

- Designed specifically for tracking electric vehicles within campus environments.
- Utilizes wireless communication protocols (ESP-NOW and Wi-Fi) for data transfer.
- Provides real-time data to end-users for better decision-making and vehicle utilization.

Limitations:

- ESP-NOW is limited in range (~100m in open space).
- Wi-Fi-based data uploads depend on available infrastructure.

- GPS accuracy and mobile network may degrade under tree cover or in dense buildings.

2. LITERATURE REVIEW

2.1 Overview of Vehicle Tracking Technologies

Vehicle tracking technologies have evolved significantly over the past decade. Early systems relied on manual logging or radio-based transmission. Modern solutions utilize GPS (Global Positioning System), GSM (Global System for Mobile communication), and IoT (Internet of Things) platforms. These systems allow real-time vehicle monitoring, geofencing, speed control, and status tracking. Research shows that the integration of microcontrollers and wireless communication modules has greatly enhanced the functionality and reliability of tracking systems, especially in constrained environments like campuses.

2.2 Existing EV Tracking Solutions

Various commercial solutions exist for electric vehicle tracking, including systems developed by companies like Tesla, Ola Electric, and Mahindra Electric. These systems typically include onboard GPS modules, SIM-based data transmission, and cloud storage. However, such systems are cost-intensive and often designed for large-scale urban deployment. In contrast, studies on small-scale EV tracking (e.g., [S. Gupta et al., 2021]) demonstrate that microcontroller-based systems using Wi-Fi and Bluetooth can be more efficient and cost-effective for local environments like university campuses. Most of these studies also emphasize modularity, ease of deployment, and customization as critical factors in such systems.

2.3 Comparison of GPS, GSM, and IoT-Based Tracking Systems

Technology	**Pros**	**Cons**
GPS	Global coverage, accurate positioning	Susceptible to signal loss indoors or under foliage
GSM	Wide area coverage, SMS/Internet support	Requires SIM card, recurring cost, not ideal for campus-only systems
IoT (ESP-NOW, Wi-Fi, LoRa)	Low power, cost-efficient, peer-to-peer capable	Range-limited (except LoRa), network setup needed for Wi-Fi

Recent papers ([R. Kumar et al., 2020]) suggest that for confined areas such as college campuses, IoT-based protocols like ESP-NOW are preferable due to their simplicity, low power consumption, and the absence of dependency on telecom providers.

2.4 Role of Ultrasonic Sensors in Smart Vehicles

Ultrasonic sensors are widely used in automotive applications for proximity sensing, obstacle detection, and parking assistance. In intelligent transport systems, these sensors are utilized to monitor the internal status of vehicles, such as seat occupancy or luggage space.

Research ([P. Sharma et al., 2019]) highlights that ultrasonic sensors are affordable, easy to interface with microcontrollers, and effective in non-contact object detection, making them suitable for occupancy detection in campus EVs.

2.5 Advantages of ESP-NOW Communication

ESP-NOW is a proprietary communication protocol by Espressif that allows low-latency, peer-to-peer wireless communication between ESP32 devices without a router. Unlike Wi-Fi or Bluetooth, ESP-NOW uses a MAC-based addressing scheme, enabling fast and efficient device-to-device messaging.

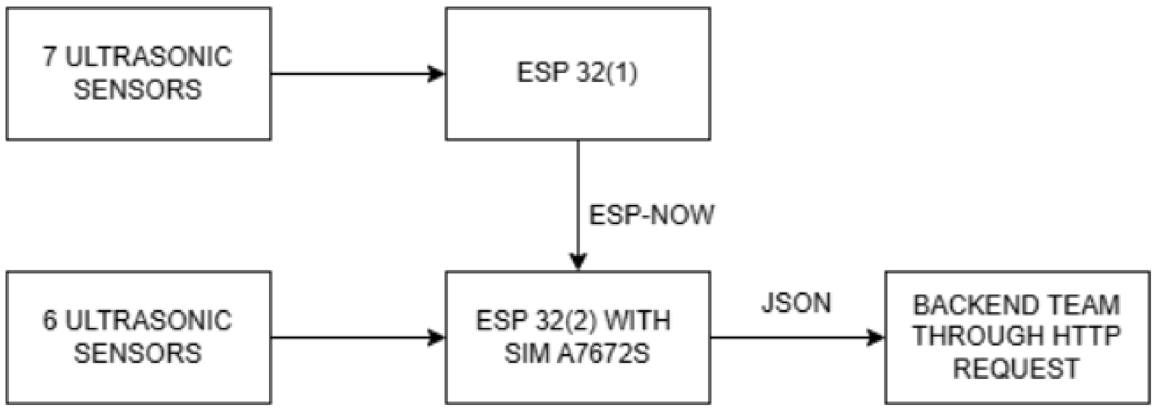
Advantages include:

- No need for an access point or internet.
- Low power consumption – ideal for battery-powered devices.
- Supports communication with up to 20 peers.
- Excellent for real-time sensor data sharing and mesh-like networks.

Studies ([L. Zhang et al., 2021]) show that ESP-NOW can be integrated into campus-scale IoT applications, ensuring seamless data flow between devices while maintaining low latency and power efficiency.

3. SYSTEM DESIGN AND ARCHITECTURE

3.1 Block Diagram of the EV tracker system:



3.2 Hardware Component Selection:

The list of components that we select for our project are:

- Ultrasonic Sensors
- ESP-32
- SIM A7672s

Ultrasonic sensor:

An ultrasonic sensor is a device that uses sound waves to measure the distance between itself and an object. It emits high-frequency sound waves (typically around 40 kHz) and measures the time taken for the echo to return after bouncing off an object.

Working Principle

1. Transmission**:** The sensor’s transmitter emits ultrasonic pulses.
2. Reflection**:** These pulses hit an object and reflect back.
3. Reception**:** The receiver detects the echo.
4. Distance Calculation**:** The sensor calculates distance using the

formula:

Distance = (speed of sound x Time delay) / 2

Ultrasonic sensors are used in this project to detect human presence in an EV seat by measuring height differences.

Non-Contact Sensing – Unlike pressure or touch-based sensors, ultrasonic sensors detect objects without physical contact, ensuring durability.

Height Measurement – They accurately measure the height of an object (passenger) above the seat, distinguishing between an occupied and unoccupied seat.

Environmental Adaptability – They work well in various lighting conditions, unlike infrared sensors, which can be affected by sunlight.

Cost-Effective & Easy to Interface – Ultrasonic sensors (like the HC-SR04) are affordable and easily integrated with microcontrollers like ESP32.

ESP -32:

The ESP32 is an advanced microcontroller designed by Espressif Systems, featuring Wi-Fi, Bluetooth, and BLE connectivity, making it ideal for IoT, embedded systems, and sensor-based applications. It is widely used in projects that require wireless communication, real-time data processing, and low power consumption.

Detailed Features & Capabilities

1. Powerful Processing Unit

- Dual-Core Xtensa LX6 CPU – Can run separate tasks on two cores simultaneously.

- Clock Speed**:** Up to 240 MHz, making it fast for real-time applications.
- SRAM: Up to 520 KB for quick data handling.

2. Built-in Connectivity

- Wi-Fi (802.11 b/g/n, 2.4 GHz): Connects to the internet or local networks for data transmission.
- Bluetooth Classic & BLE (v4.2): Useful for short-range communication and energy-efficient applications.
- ESP-NOW Protocol: Low-latency, direct communication between ESP32 modules without a Wi-Fi network.

3. Peripheral Support for Sensor Integration

- GPIO (General-Purpose Input/Output): Up to 34 pins for interfacing sensors and actuators.
- ADC (Analog-to-Digital Converter): Reads analog sensor values (e.g., force sensors, temperature sensors).
- DAC (Digital-to-Analog Converter): Generates analog signals for controlling external devices.
- PWM (Pulse Width Modulation): Controls LEDs, motors, and other actuators.
- I2C/SPI/UART: Interfaces with various digital sensors and communication modules.

4. Low Power Consumption

- Supports Deep Sleep Mode, reducing power usage to a few microamperes (µA).

- Ideal for battery-powered applications, like EV monitoring systems.

Why ESP32 for the EV Sensor System?

1. Wireless Communication:
    - ESP-NOW enables fast, peer-to-peer data exchange between multiple ESP32 modules in the EV.
    - Wi-Fi or SIM7672S can be used for cloud or backend communication.
2. Real-time Sensor Processing: Captures and processes ultrasonic and force sensor data efficiently.
3. More GPIO Pins for Multiple Sensors:
    - The ESP32 has up to 34 GPIO pins, allowing it to connect multiple sensors.
    - This makes it ideal for a sensor-rich EV application, where multiple inputs need to be processed simultaneously.
4. Low Power Mode: Helps conserve battery life in an electric vehicle by supporting Deep Sleep Mode with ultra-low power consumption.
5. Multiple Interfaces: Supports I2C, SPI, UART, ADC, and PWM, making it easy to integrate additional components.

SIM A7672s:

The SIM7672S is a 4G LTE Cat-1 module that enables cellular-based communication, making it suitable for IoT applications like EV tracking and data transmission. It supports GPS, HTTP, MQTT, and cloud connectivity, allowing remote monitoring and control.

Key Features & Capabilities

1. Cellular Network Connectivity
   - Supports 4G LTE Cat-1, 3G, and 2G fallback for reliable communication.
   - Works in various network conditions, making it ideal for EVs moving across different areas.
2. GPS for Vehicle Tracking
   - Built-in GNSS (GPS, BeiDou, GLONASS, Galileo) for real-time location tracking.
   - Provides latitude and longitude, allowing EV tracking over a backend server.
3. Data Transmission & Protocols
   - Supports HTTP, MQTT, FTP, and TCP/IP, making it ideal for cloud-based applications.
   - Can send sensor data (ultrasonic, force sensor readings) to a remote server or mobile app.
4. More GPIO Pins for External Sensors & Peripherals
   - The SIM7672S module includes multiple GPIOs to interface with external sensors, relays, or actuators.
   - This allows direct sensor integration without needing a separate microcontroller in some cases.
5. Low Power Consumption for EV Applications
   - Supports low-power sleep modes, ensuring efficient energy usage in an EV system.

6. Compatible with ESP32 & Other Microcontrollers

- Communicates via UART (AT commands) with ESP32 or other MCUs.
- Can work alongside ESP-NOW or LoRa WAN for hybrid data transmission.

Why SIM7672S for the EV TRACKER System?

Reliable Communication Anywhere:

- Unlike Wi-Fi, cellular networks provide broader coverage, ensuring continuous data transmission.

Real-time GPS Tracking:

- Allows location-based monitoring for EV fleet tracking or security purposes.

More GPIO Pins for Multiple Sensors & Devices:

- Enables direct connection of additional sensors or actuators, reducing hardware complexity.

Multiple Protocols for Cloud Integration:

- Supports HTTP, MQTT, and TCP/IP, making it easy to send data to a backend server or mobile application.

Low Power for Energy Efficiency:

- Helps extend EV battery life by optimizing power consumption.

3.3 Software And Communication Protocols:

The Arduino Integrated Development Environment (IDE) is an open-source software used for writing, compiling, and uploading code to microcontrollers like ESP32, Arduino boards, and other compatible devices. It provides a simple yet powerful interface for embedded system development, making it popular among students, hobbyists, and professionals.

Key Features of Arduino IDE

User-Friendly Interface:

- Simple code editor, compiler, and uploader in a single platform.

Cross-Platform Support:

- Works on Windows, macOS, and Linux.

Extensive Library Support:

- Provides built-in and external libraries for sensors, communication modules (Wi-Fi, Bluetooth, LoRa), and peripherals.

Serial Monitor & Debugging Tools:

- Helps in real-time debugging by displaying sensor values and logs.

Board Manager & Compatibility:

- Supports ESP32, Arduino, STM32, and more with easy installation via the Board Manager.

COMMUNICATION PROTOCOLS:

Our EV sensor system uses multiple communication protocols for sensor data transmission, cloud connectivity, and inter-device communication.

1. ESP-NOW (ESP32-to-ESP32 Communication)

- Why? Fast and low-power peer-to-peer data exchange.
- How? One ESP32 collects all sensor data and transmits it to another ESP32 for processing.
- Advantages:
  - Low latency.
  - No need for Wi-Fi or the internet.
  - Power-efficient.

2. HTTP (For Sending Data to Backend Server - SIM7672S to Cloud)

- Why? Used to send sensor data (e.g., GPS, ultrasonic, force sensor readings) to a web server/cloud backend.
- How? SIM7672S connects to the internet and sends data using an HTTP POST request in JSON format.
- Advantages:
  - Compatible with cloud databases like Firebase, AWS, and custom APIs.
  - Can be accessed from anywhere.

SYSTEM WORKFLOW:

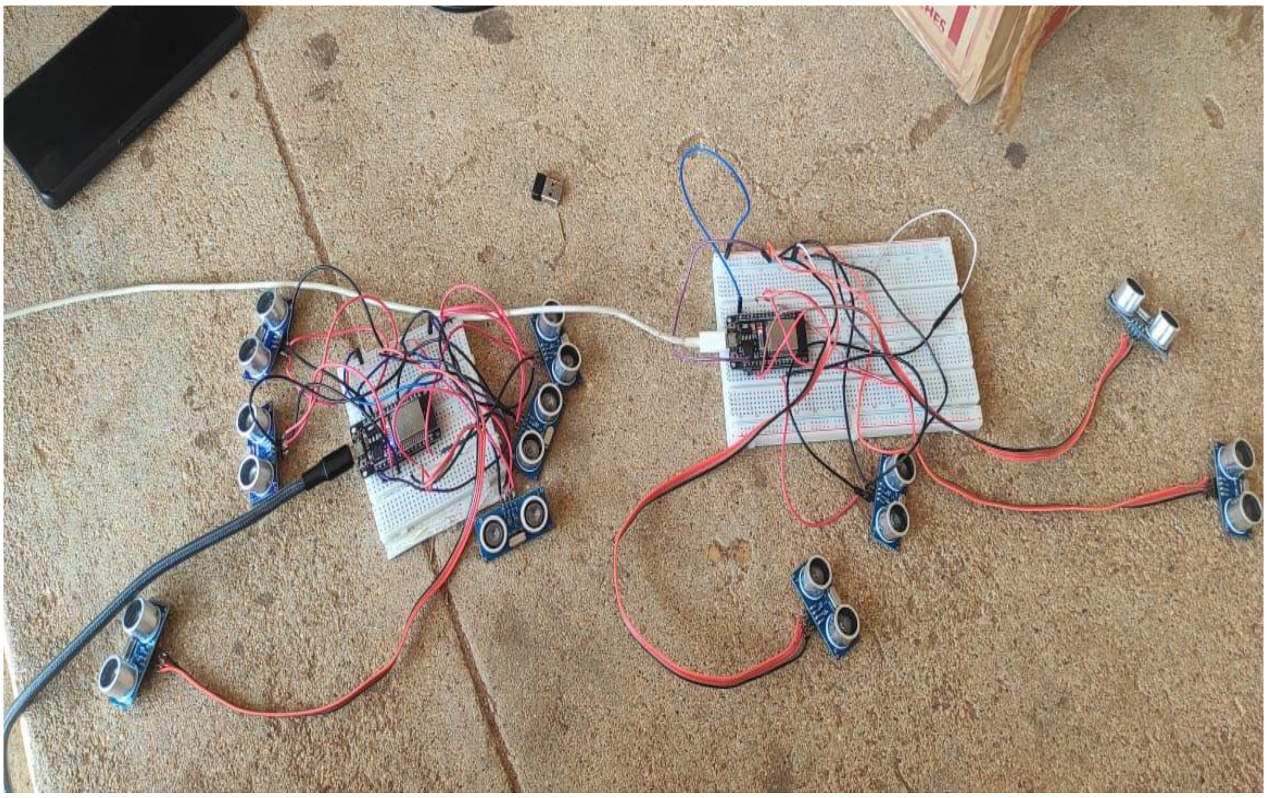
Step-by-Step System Workflow

1. Sensor Data Collection (ESP32-1 & ESP32-2)

  - Multiple ultrasonic sensors are connected to ESP32-1 and ESP32-2.
  - These sensors measure distances and detect the human presence in the seats.
  - ESP32-1 processes the sensor data, filtering noise if needed.
2. Wireless Data Transfer (ESP32-1 to ESP32-2 via ESP-NOW)
  - ESP-NOW enables fast, low-latency communication between ESP32-1 and ESP32-2.
  - ESP32-1 transmits the processed ultrasonic sensor data to ESP32-2.
  - ESP32-2 acts as a data hub, collecting sensor data from ESP32-1 and its own sensors, preparing the data for transmission.
3. GPS Data Retrieval (ESP32-2 via SIM7672S)
  - ESP32-2 sends AT commands to the SIM7672S module to obtain the GPS coordinates of the EV.
  - The latitude and longitude are retrieved and stored along with the ultrasonic sensor data.
4. Cellular Data Transmission (ESP32-2 to Backend via SIM7672S)
  - ESP32-2 sends the processed sensor and GPS data to the SIM7672S module via UART communication.
  - SIM7672S connects to the 4G LTE network and sends the data to a predefined URL via an HTTP POST request in

JSON format.

- The backend team receives the data from this URL and processes it for further analysis.



3.4 Component Connecions:

The hardware connections in this project are as follows:

- Ultrasonic Sensor (HC-SR04) → ESP32
  - VCC → VIN (5V)
  - GND → GND
  - TRIG → GPIO 5
  - ECHO → GPIO 18 (via voltage divider to 3.3V
- SIM7672S Module → ESP32 (UART Communication)
  - TX → GPIO 16 (RX2)
  - RX → GPIO 17 (TX2)
  - VCC → 5V

    - GND → GND
- Power Considerations:
    - Ultrasonic sensors & ESP32 → Powered via VIN (5V) from ESP32
    - SIM7672S → Requires higher current; may need a separate power source

4.COMMUNICATION PROTOCOLS AND DATA TRANSMISSION

4.1 Software Implementation

Microcontroller Firmware for ESP32

The firmware for ESP32 in the **EV Tracker Project** is developed using Arduino IDE. The code implementation includes:

- Initialization of **ESP-NOW communication** for seamless data transfer.
- Configuration of **13 ultrasonic sensors** to detect seat occupancy.
- Implementation of **GPS module integration** for real-time vehicle tracking.
- Debugging and logging mechanisms to monitor data transmission and processing.

Sensor Data Processing Algorithm

- The **sender ESP32** triggers ultrasonic sensors and measures the echo time.
- Using the speed factor (0.0343 cm/µs), the sensor calculates the **distance to an object**.
- With the help of this, the data is processed and stored in an array and then, structured into a message packet.
- The **receiver ESP32** validates the data and integrates it with GPS coordinates.
- The final processed data is structured for real-time tracking.

Communication between ESP32 (Master-Slave Configuration)

- The **sender ESP32** measures seat occupancy and transmits data via ESP-NOW.
- The **receiver ESP32** acts as a master node, integrating GPS and seat occupancy data.
- The system implements a retransmission mechanism with delays between the two Esp32's ensuring code and data reliability and reducing data loss.
- The processed data is formatted and stored for further analysis.

4.2 HTTP Request Handling and Data Forwarding to Backend

- The **receiver ESP32** sends processed data to a remote server via an HTTP POST request.

- The data is formatted in **JSON format**, which allows structured transmission of seat occupancy and GPS information.
- An **HTTPClient** library is used in the ESP32 firmware to initiate the request.
- The request includes necessary headers, such as Content-Type: application/json.
- The ESP32 utilizes **WiFiClientSecure** to establish a secure HTTPS connection with the backend server.
- The **JSON payload** includes the device identifier, timestamp, and an array of sensor readings.
- After sending the data, the ESP32 handles the **server response**, checking HTTP status codes to confirm successful data reception.
- If the request fails, appropriate error handling mechanisms are implemented to retry or log failures for debugging.
- The system continuously attempts to reconnect if the Wi-Fi connection is lost, ensuring reliable data transmission.

4.3 Challenges faced and Solutions:

Ultrasonic Sensor Interference:

- **Problem:** Multiple ultrasonic sensors operating simultaneously caused signal interference, leading to inaccurate readings.

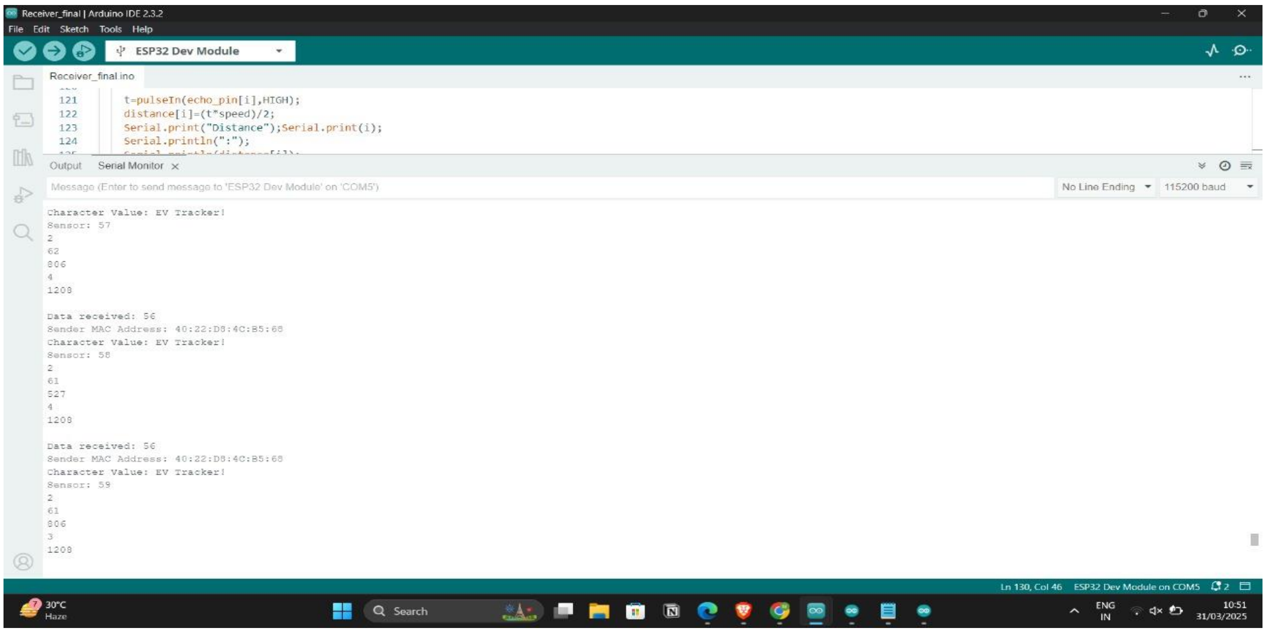
- **Solution:** Adjusted trigger timings and optimized sensor placement to minimize cross-talk, improving detection accuracy.

ESP-NOW Packet Loss:

- **Problem:** Data packets were occasionally lost during wireless transmission, leading to incomplete data reception.
- **Solution:** Implemented a retransmission mechanism with controlled delays and acknowledgment-based communication to ensure data integrity.

GPS Connectivity Issues:

- **Problem:** The **Neo-6M GPS module** had long satellite acquisition times, sometimes taking hours to connect.
- **Solution:** Testing in open areas reduced connection time to seconds, but the module's limitations made it unsuitable for EV tracking. Upgrading to **Neo-M8N or Neo-M9N** was suggested for improved performance.



```
t=pulseIn(echo_pin[i],HIGH);
distance[i]=(t*speed)/2;
Serial.print("Distance");Serial.print(i);
Serial.println(":");
```

```
Character Value: EV Tracker!
Sensor: 57
2
62
806
4
1208

Data received: 56
Sender MAC Address: 40:22:D8:4C:B5:68
Character Value: EV Tracker!
Sensor: 58
2
61
527
4
1208

Data received: 56
Sender MAC Address: 40:22:D8:4C:B5:68
Character Value: EV Tracker!
Sensor: 59
2
61
806
3
1208
```

WORKING CODE:

Sender Code:

```
// Include Libraries
#include <esp_now.h>
#include <WiFi.h>
const int trig_pin[]={18,2,13,24,19,32};
const int echo_pin[]={5,4,12,35,21,33};
int n=sizeof(trig_pin);
float distance[6];
float t;
float speed=0.0343;
// Variables for test data
int int_value;
float float_value;
```

```
bool bool_value = true;
// MAC Address of responder - edit as required
uint8_t broadcastAddress[] = {0xC8, 0x2E, 0x18, 0x8D, 0xC1,
0x98};
// Define a data structure
typedef struct struct_message {
  char a[32];
  int u[6];
} struct_message;
// Create a structured object
struct_message myData;
// Peer info
esp_now_peer_info_t peerInfo;
// Callback function called when data is sent
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t
status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Delivery
Success" : "Delivery Fail");
}
void setup() {
 // Set up Serial Monitor
  Serial.begin(115200);
  for(int i=0;i<6;i++){
    pinMode(trig_pin[i],OUTPUT);
    pinMode(echo_pin[i],INPUT);
    digitalWrite(trig_pin[i],LOW);
  }
  // Set ESP32 as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  // Initilize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
  // Register the send callback
  esp_now_register_send_cb(OnDataSent);
  // Register peer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;
  // Add peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }
}
void loop() {

  // Create test data
  for(int i=0;i<1;i++){
   digitalWrite(trig_pin[i],LOW);
   delayMicroseconds(2);
   digitalWrite(trig_pin[i],HIGH);
   delayMicroseconds(10);
   t=pulseIn(echo_pin[i],HIGH);
   distance[i]=(t*speed)/2;
   myData.u[i]=distance[i];
   Serial.println("Distance");
   Serial.println(distance[i]);
  }
  // Format structured data
  strcpy(myData.a, "EV Tracker!");
  Serial.print("Character Value: ");
  Serial.println(myData.a);
  Serial.print("Sensor: ");
  for(int i=0;i<6;i++){
  Serial.println(myData.u[i]);
  }
  Serial.println();
  // Send message via ESP-NOW
  esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *)
&myData, sizeof(myData));
  if (result == ESP_OK) {
   Serial.println("Sending confirmed");
  }
  else {
    Serial.println("Sending error");
  }
  delay(500);
}
```

Receiver Code :

```
// C-type esp32
#include <esp_now.h>
#include <WiFi.h>
#include<HTTPClient.h>
#include<ArduinoJson.h>
const int trig_pin[]={18,2,13,25,22,32};
const int echo_pin[]={5,4,12,26,23,35};
const char* ssid="OnePlus 11R 5G";
const char* password="ymd112365";
const char* server ="server_url";
int n=sizeof(trig_pin);
int distance[13];
float t;
float speed=0.0343;
bool dataReceived=false;
// Define a data structure
typedef struct struct_message {
  char a[32];
  int u[6];
} struct_message;
// Create a structured object
struct_message myData;
// Callback function executed when data is received
void OnDataRecv(const esp_now_recv_info* recv_info, const uint8_t
*incomingData, int len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Data received: ");
  Serial.println(len);
  // Print the MAC address of the sender
  Serial.print("Sender MAC Address: ");
  for (int i = 0; i < 6; i++) {
    Serial.print(recv_info->des_addr[i], HEX); // peer_addr contains
the MAC address
    if (i < 5) {
      Serial.print(":");
    }
  }
  Serial.println();
  // Print the received data
  Serial.print("Character Value: ");
  Serial.println(myData.a);
  Serial.print("Sensor: ");
  for(int i=0;i<6;i++){
  Serial.println(myData.u[i]);
  }
  Serial.println();
  dataReceived=true;
  Serial.println("Sensor Data");
    for(int i=0;i<6;i++){
    digitalWrite(trig_pin[i],LOW);
    delayMicroseconds(2);
    digitalWrite(trig_pin[i],HIGH);
    delayMicroseconds(10);
    t=pulseIn(echo_pin[i],HIGH);
    distance[i]=(t*speed)/2;
    Serial.print("Distance");Serial.print(i);
    Serial.println(":");
    Serial.println(distance[i]);
    delay(100);
  }
  for(int i=6;i<12;i++){
    distance[i]=myData.u[i-6];
    Serial.print("Distance");Serial.print(i);
    Serial.println(":");
    Serial.println(distance[i]);
    delay(100);
  }
}
void send_data_to_server(){
  HTTPClient http;
  http.begin(server);
  http.addHeader("Content-Type","applicatiob/json");
  StaticJsonDocument<200>doc; //200 bytes size of data
  JsonArray ultrasonicArray =
doc.createNestedArray("Ultrasonic_Sensor");
  for(int i=0;i<13;i++){
    ultrasonicArray.add(myData.u[i]);
  }
  String json;
  serializeJson(doc,json);
  int httpResponseCode=http.POST(json);
  Serial.println("HTTP Response :" + String(httpResponseCode));
  String response=http.getString();
  Serial.println("Response:"+ response);
  http.end();
}
void setup() {
  Serial.begin(115200);
 // Set ESP32 as a Wi-Fi Station
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid,password);
  if(WiFi.status()!= WL_CONNECTED){
    delay(1000);
    Serial.print(".");
  }
  // Initialize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }
   for(int i=0;i<6;i++){
    pinMode(trig_pin[i],OUTPUT);
    pinMode(echo_pin[i],INPUT);
    digitalWrite(trig_pin[i],LOW);
  }  // Register the callback function
  esp_now_register_recv_cb(OnDataRecv);
}
void loop() {
  if(dataReceived){
  send_data_to_server();
  delay(1000);
  dataReceived=false;
  }
}
```

5. PERFORMANCE ANALYSIS & RESULTS

This section presents a comprehensive evaluation of the system’s performance, focusing on hardware testing, communication reliability, sensor accuracy, and challenges encountered during development. Alternative solutions and workarounds for identified issues are also discussed.

5.1 Hardware Testing Methodology

Objective

- To validate the robustness and reliability of the hardware components under real-world operating conditions.
- Testing Approach Functional Testing: Each module (ESP32, ultrasonic sensors, 4G LTE module) was tested individually before system integration.
- Environmental Testing: Performed in varying conditions (indoor/outdoor) to evaluate resilience.
- Findings ESP32: Operated reliably with minimal overheating, even under continuous data transmission.
- Power Consumption: The system drew 5V during active operation when transmitting data.
- Failures Observed: Occasional Wi-Fi instability in high-interference environments.

ESP-NOW Performance Evaluation

Objective

To assess the reliability, latency, and range of ESP-NOW communication between ESP32 nodes.

Testing Parameters Range: Tested in open-field and obstructed (indoor) environments.

Packet Loss: Measured at different distances (1m to 50m). Latency: Time taken for a message to transmit and receive acknowledgment.

Results Analysis

Optimal Range: Best performance within 10m, with near-instantaneous communication.

Limitations: Significant packet loss beyond 30m, making it unsuitable for long-range applications without repeaters.

Ultrasonic Sensor Data Accuracy Analysis

Objective :

To evaluate the precision and reliability of ultrasonic sensors in distance measurement.

Testing Setup: Compared sensor readings against a laser distance measurer as ground truth. Tested at varying distances (10cm to 400cm) and angles (0° to 30° tilt).

Accuracy: Best within 100cm (error < 2%), deteriorates slightly beyond 200cm.

Environmental Impact: Humidity and wind caused minor fluctuations (±3cm). (Recommendations: Use averaging filters to reduce noise. Avoid extreme angles (>15°) for reliable readings)

5.2 Failed GPS Tracking with 4G LTE SIM7670S

Objective

To integrate GPS tracking using the SIM7670S module for real-time location updates.

5.3 Challenges Encountered

1. GPS Lock Failure: Module frequently failed to acquire satellite signals in urban areas.
2. High Power Consumption: Draining battery quickly (~180mA in GPS mode).
3. Data Transmission Issues: Intermittent 4G connectivity led to incomplete location updates.

Root Cause Analysis

Urban Canyon Effect: Tall buildings obstructed GPS signals.

Antenna Limitations: Weak internal antenna in the SIM7670S.

Network Latency: Delays in 4G data transmission increased location inaccuracy.

6. FUTURE ENHANCEMENTS

6.1 Battery optimization :

The whole setup required 12V 7Ah external battery when implemented independently in the EV. This will give the unwanted power hittings with respect to EV. While power distribution wiring can be done using BUCK CONVERTER 12-5V which will ensure safe and secure consumption for this system. The battery can be rechargeable so that weekly once it can be recharged for perfect functioning.

6.2 Integration with Mobile Application

A dedicated mobile application can be developed to display real-time EV data, including location, route, and seat availability. It can also feature notifications when a vehicle is near, improving user convenience and system accessibility.

6.3 Real-Time Server Alerts for Maintenance

Include a feature where sensor or GPS anomalies (like signal loss, low battery, or frequent seat sensor errors) automatically trigger alerts to the maintenance team via email or server notifications. This ensures proactive troubleshooting and increases system reliability.

6.4 Solar-Powered ESP32 Nodes

To increase energy efficiency and support sustainability, ESP32 devices and sensors can be powered using small solar panels. This reduces dependency on external power sources and simplifies installation within the campus.

7. CONCLUSION

This project presents a working prototype of a campus electric vehicle (EV) tracking system that combines GPS, ultrasonic sensors, ESP32 microcontrollers, and ESP-NOW wireless communication. The system is designed to monitor EV location and seat occupancy in real-time, offering an efficient and low-cost solution tailored for campus environments.

While full deployment has not yet been implemented, the developed model successfully demonstrates the feasibility and functionality of the proposed system. It showcases how real-time tracking and wireless data transmission can be achieved without relying on GSM or internet-based solutions, making it ideal for limited-range applications like college campuses.

The prototype lays a strong foundation for future expansion and practical implementation, with enhancements such as mobile app integration and solar-powered modules proposed for further development.