Remote Control of Vehicle Indicators

1. Objective of the Project

The primary objective of this project is to design and demonstrate a **wireless remote control system** capable of operating vehicle indicators (Left and Right) using a **mobile application** and an **ESP32 microcontroller**.

This project focuses on:

- True remote actuation (not just monitoring)
- Deterministic and reliable command execution
- Automotive aligned control behavior

Scope note: This document strictly covers remote control of indicators only. It does not include diagnostics (DTCs), immobilizer logic, tilt detection, or other vehicle features.

2. Problem Statement

Traditional indicator control relies on:

- Manual switches
- Physical wiring
- Direct human interaction

For prototyping, testing, and IoT based automotive concepts, this approach has limitations:

- Difficult to test remotely
- Limited flexibility
- Not suitable for modern connected vehicle demonstrations

Hence, there is a need for a **wireless remote control mechanism** that allows indicators to be operated through a mobile device without physical switches.

3. Why Remote Control for Indicators

Remote controlled indicators are useful for:

- Automotive feature prototyping
- Embedded system demonstrations
- IoT based vehicle control concepts
- Reducing manual wiring complexity during testing

Benefits

- Wireless operation using Wi Fi
- Easy control via mobile application
- Flexible and scalable architecture
- Ideal for learning automotive and embedded concepts

4. Finalized Approach

4.1 Why ESP32 + Mobile Application

The ESP32 was selected because it:

- Has built in Wi Fi
- Supports Access Point (AP) mode
- Is widely used in automotive and IoT prototyping
- Allows direct mobile to device communication

The mobile application acts as a **remote controller**, sending commands to the ESP32 over Wi Fi to control the indicators.

5. System Overview

The system consists of three main components:

- **Mobile Application** – Sends control commands
- **ESP32 Microcontroller** – Receives commands and controls outputs
- **Indicators** – Left and Right indicator lamps / LEDs

6. Components Used

6.1 Hardware Components

- **ESP32 Development Board**
  Acts as the main controller with built in Wi Fi
- **Left Indicator (LED / Lamp)**
  Represents left turn indicator
- **Right Indicator (LED / Lamp)**
  Represents right turn indicator
- **Relay Module / Transistor Driver**
  Used when controlling 12V automotive indicators
- **Power Supply**
  5V for ESP32, 12V for indicators (if required)
- **Connecting Wires**

6.2 Software Components

- Arduino IDE – Programming ESP32
- Android Studio – Mobile application development
- ESP32 Wi Fi libraries

7. Role of ESP32

The ESP32 acts as the **control unit** of the system.

Functions of ESP32

- Creates a Wi Fi network (AP mode)
- Accepts connections from mobile phone
- Receives control commands
- Controls GPIO pins
- Switches indicators ON or OFF

8. Role of Mobile Application

The mobile application acts as a **remote controller interface**.

Functions of the Mobile App

- Provides buttons for indicator control
- Sends commands over Wi Fi
- Communicates directly with ESP32

Mobile App Control Buttons

Button	**Function**
Left ON	Turns Left Indicator ON
Left OFF	Turns Left Indicator OFF
Right ON	Turns Right Indicator ON
Right OFF	Turns Right Indicator OFF
Hazard ON	Both Left and Right Indicators ON

Hazard OFF	Both Left and Right Indicators OFF

9. Wi Fi Communication Design

ESP32 as Access Point (AP Mode)

In this project, ESP32 is configured in **Wi Fi Access Point mode**.

- ESP32 creates its own Wi Fi network
- Mobile phone connects directly to ESP32
- No internet connection required
- No external Wi Fi router needed

Communication Flow

Mobile App → ESP32 Wi-Fi → Indicators

Example Command

IP address:192.168.4.1/left_on
port:8888

10. Working Principle

1. **ESP32 powers ON and connects to the Wi-Fi network**
   The ESP32 connects to the available Wi-Fi network in **Station (STA) mode**.
2. **ESP32 displays IP address and TCP port number on Serial Monitor**
   After successful connection, the ESP32 prints its **IP address and TCP port number** on the Serial Monitor.

3. **Mobile phone connects to the same Wi-Fi network**

Both the mobile phone and ESP32 are connected to the **same Wi-Fi network**.

4. **User enters ESP32 IP address and port number in the mobile application**
   The user manually enters the **IP address and port number** obtained from the Serial Monitor into the Android app.
5. **User presses a control button in the mobile app**
   The app provides buttons to control indicators such as **Left, Right, and OFF**.
6. **Command is sent using communication over Wi-Fi**
   The mobile application establishes a **connection** with the ESP32 and sends control commands.
7. **ESP32 receives and processes the data**
   The ESP32 listens on the specified port, receives incoming data, and parses the command.
8. **Corresponding GPIO pin is activated**
   Based on the received command, the ESP32 sets the respective GPIO pin **HIGH or LOW**.
9. **Indicator turns ON or OFF**
   The connected indicator turns **ON or OFF** according to the command.

12. Challenges Faced

12.1 Wi-Fi Connectivity

- Ensuring a stable and reliable Wi-Fi connection between the mobile device and the ESP32 for uninterrupted communication.

12.2 Android Studio Environment

- Android Studio requires a **high-performance system setup**, including **more than 8 GB RAM**, to ensure smooth development, build, and emulator operation. Insufficient memory may lead to slow performance or build failures.

12.3 Command Reliability

- Ensured deterministic and reliable mapping of received commands to the corresponding GPIO actions on the ESP32.

13. Key Learnings

- Gained hands-on experience in establishing **reliable Wi-Fi communication** between an ESP32 and an Android mobile application using the same network.

- Learned how to implement **communication** using IP address and port number for real-time device control.

- Understood the importance of **stable network connectivity** for consistent data transfer and system reliability.

- Acquired practical knowledge of **Android Studio development**, including awareness of **system requirements such as more than 8 GB RAM** for smooth application development.

- Learned to ensure **deterministic command handling**, where each received command is accurately mapped to a specific GPIO action on the ESP32.
- Improved understanding of **embedded system–mobile application integration** and real-time hardware control.

14. Current Status of the Project

- Left and Right indicators successfully controlled
- Mobile application communication verified
- ESP32 Wi Fi AP mode stable
- Project objectives achieved

15. Conclusion

This project successfully demonstrates a **remote indicator control system** using ESP32 and a mobile application. By leveraging Wi Fi communication and a clear control architecture, the system provides reliable and scalable remote operation aligned with automotive prototyping principles. This approach serves as a strong foundation for future connected vehicle and ECU level control developments.