



NFC-IET, Multan

BSAI-2k24

Project Report

CodingMoves-RC+ (RC Car using ESP8266)

Submitted by:

- Moavia Amir (2k24_BSAI_72)
- Hassan Khan(2k24_BSAI_48)
- Fatima Hassan (2k24_BSAI_07)
- JAVERIA BABAR (2K24_BSAI_14)

Department of Artificial Intelligence

NFC-IET, Multan

Submitted to: Engr. Romaisa Shamshad Khan

Date: 13/05/2025

Abstract

This project demonstrates the design and implementation of a Wi-Fi controlled RC car using the **ESP-12 (ESP8266)** module. A mobile phone interface built using HTML communicates with the ESP-12 over Wi-Fi, enabling users to control the car's direction (forward, backward, left, right) through button inputs. The system combines **IoT, embedded programming, and basic web development**, offering a strong educational model for real-world wireless automation and control.

Introduction

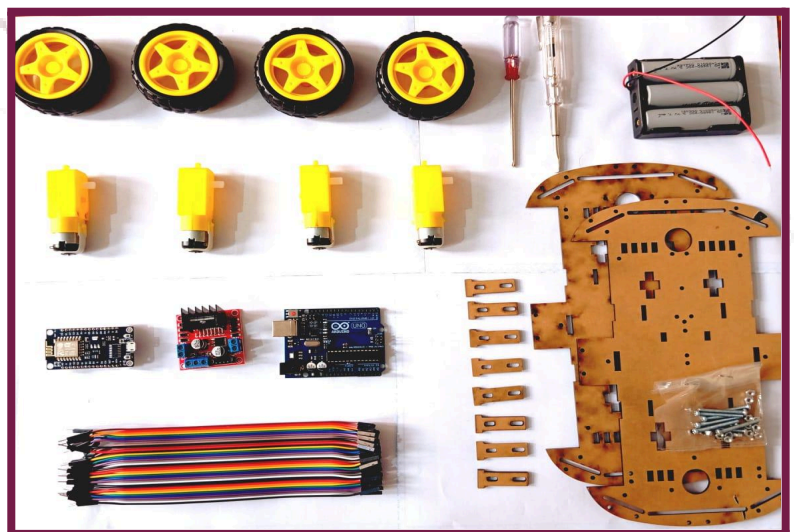
As automation technologies evolve, wireless control becomes central to smart systems. In this project, a prototype **Wi-Fi based RC car** is designed that allows wireless navigation via smartphone. The ESP-12 microcontroller hosts an internal web server, providing a custom-built HTML interface to the user. Button presses on this interface trigger specific GPIO pins, enabling precise control of the motor driver and movement. This project reflects fundamental IoT applications in remote robotics and embedded systems.

Objectives

- Design and implement an **IoT-enabled RC car** using the **ESP-12 module**.
- Develop a **mobile-responsive HTML interface** for user control.
- Establish **real-time wireless communication** between the device and mobile.
- Gain hands-on experience with **embedded systems, motor driving, and Wi-Fi networking**.

Components Required

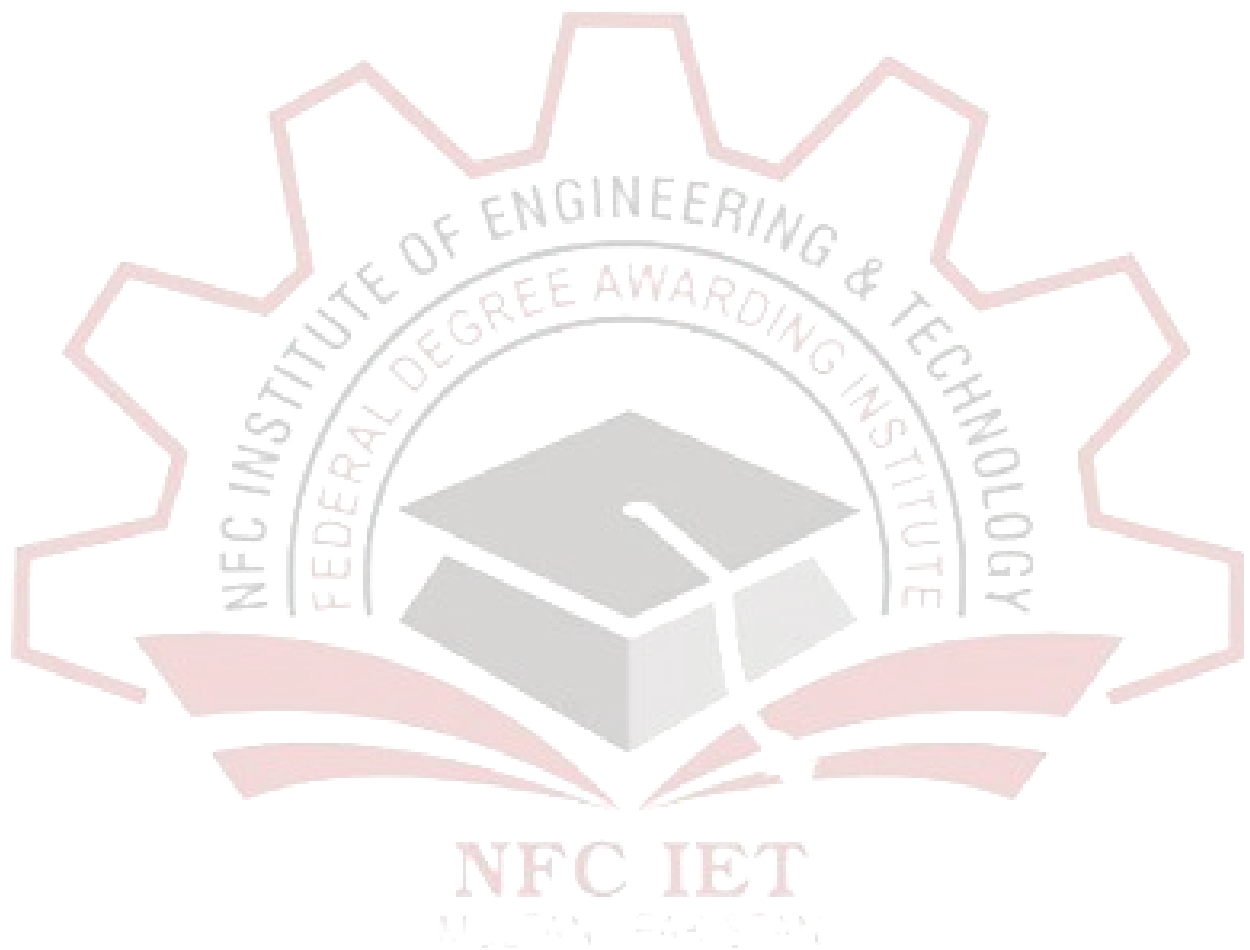
- ESP-12 (ESP8266) Wi-Fi Module
- L298N Motor Driver Module
- DC Motors with Wheels (x4)
- Lithium-ion Battery or Power Supply
- Jumper Wires
- Smartphone (with HTML interface loaded)
- Chassis for car body

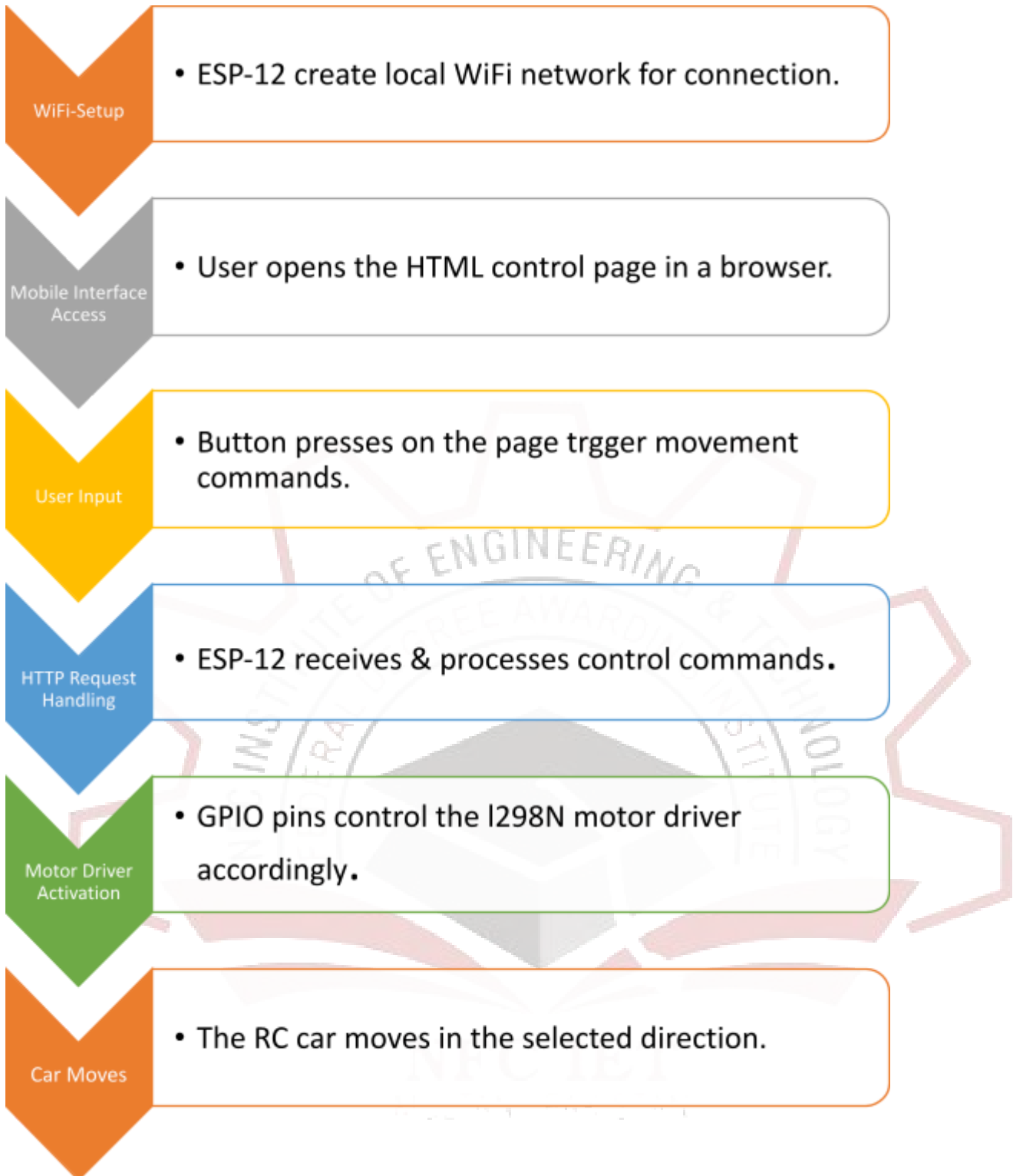


Working Principle

The ESP-12 creates a local Wi-Fi network and hosts a **web server** that serves a **steering interface (HTML page)**. When a user connects via mobile, they can interact with buttons labeled for car movement. These button presses send **HTTP GET requests** to the ESP-12, which triggers relevant **GPIO pins** to control the **L298N Motor Driver**. The driver then powers the motors in the corresponding direction. This design provides a **wireless, real-time solution** for embedded hardware control using web technology.

Methodology





Coding

- EPS8266 Code in C++

// By Coding Moves - ESP8266 WiFi RC Car

```
#include <ESP8266WiFi.h>
```

```
#include <WiFiClient.h>
```

```
#include <ESP8266WebServer.h>
```

```
#define ENA 14
#define ENB 12
#define IN_1 15
#define IN_2 13
#define IN_3 2
#define IN_4 0

const char* ssid = "Coding Moves RC+";
const char* password = "codingmoves123";

ESP8266WebServer server(80);

String command;

int speedCar = 800, speed_Coeff = 3;

void setup() {
    pinMode(ENA, OUTPUT); pinMode(ENB, OUTPUT); pinMode(IN_1, OUTPUT);
    pinMode(IN_2, OUTPUT); pinMode(IN_3, OUTPUT); pinMode(IN_4, OUTPUT);
    Serial.begin(115200);
    WiFi.mode(WIFI_AP); WiFi.softAP(ssid, password);
    Serial.print("AP IP: "); Serial.println(WiFi.softAPIP());
    server.on("/", HTTP_handleRoot); server.onNotFound(HTTP_handleRoot); server.begin();
}

void goAhead() { digitalWrite(IN_1,0); digitalWrite(IN_2,1); analogWrite(ENA,speedCar);
digitalWrite(IN_3,0); digitalWrite(IN_4,1); analogWrite(ENB,speedCar); }

void goBack() { digitalWrite(IN_1,1); digitalWrite(IN_2,0); analogWrite(ENA,speedCar);
digitalWrite(IN_3,1); digitalWrite(IN_4,0); analogWrite(ENB,speedCar); }

void goRight() { digitalWrite(IN_1,0); digitalWrite(IN_2,1); analogWrite(ENA,speedCar);
digitalWrite(IN_3,1); digitalWrite(IN_4,0); analogWrite(ENB,speedCar); }

void goLeft() { digitalWrite(IN_1,1); digitalWrite(IN_2,0); analogWrite(ENA,speedCar);
digitalWrite(IN_3,0); digitalWrite(IN_4,1); analogWrite(ENB,speedCar); }

void goAheadRight() { digitalWrite(IN_1,0); digitalWrite(IN_2,1);
analogWrite(ENA,speedCar/speed_Coeff); digitalWrite(IN_3,0); digitalWrite(IN_4,1);
analogWrite(ENB,speedCar); }

void goAheadLeft() { digitalWrite(IN_1,0); digitalWrite(IN_2,1); analogWrite(ENA,speedCar);
digitalWrite(IN_3,0); digitalWrite(IN_4,1); analogWrite(ENB,speedCar/speed_Coeff); }

void goBackRight() { digitalWrite(IN_1,1); digitalWrite(IN_2,0);
analogWrite(ENA,speedCar/speed_Coeff); digitalWrite(IN_3,1); digitalWrite(IN_4,0);
analogWrite(ENB,speedCar); }

void goBackLeft() { digitalWrite(IN_1,1); digitalWrite(IN_2,0); analogWrite(ENA,speedCar);
digitalWrite(IN_3,1); digitalWrite(IN_4,0); analogWrite(ENB,speedCar/speed_Coeff); }
```

```
void stopRobot() { digitalWrite(IN_1,0); digitalWrite(IN_2,0); analogWrite(ENA,0);
digitalWrite(IN_3,0); digitalWrite(IN_4,0); analogWrite(ENB,0); }

void loop() {
  server.handleClient(); command = server.arg("State");
  if(command=="F") goAhead(); else if(command=="B") goBack();
  else if(command=="L") goLeft(); else if(command=="R") goRight();
  else if(command=="I") goAheadRight(); else if(command=="G") goAheadLeft();
  else if(command=="J") goBackRight(); else if(command=="H") goBackLeft();
  else if(command=="S") stopRobot();
  else if(command=="0") speedCar=400; else if(command=="1") speedCar=470;
  else if(command=="2") speedCar=540; else if(command=="3") speedCar=610;
  else if(command=="4") speedCar=680; else if(command=="5") speedCar=750;
  else if(command=="6") speedCar=820; else if(command=="7") speedCar=890;
  else if(command=="8") speedCar=960; else if(command=="9") speedCar=1023;
}

void HTTP_handleRoot() {
  if(server.hasArg("State")) Serial.println(server.arg("State"));
  server.send(200, "text/html", ""); delay(1);
}
```

Software & Tools

- Arduino IDE (ESP8266 Support via Board Manager)
- HTML/CSS for web interface
- Serial Monitor (for testing inputs)
- Mobile Web Browser for control interface

Results and Observations

- Car responded correctly to all directional inputs (F, B, L, R).

- Smooth control observed at various speeds (0–9 scale PWM).
- Minimal delay in signal transmission over Wi-Fi.
- Stable battery supply maintained continuous movement.

Challenges Faced

- Power management: ESP-12 resets with motor spikes.
- HTML interface optimization for mobile resolution.
- Adjusting delay and PWM values for synchronized movement.

Learning Outcomes

- Built a full-stack embedded system using **hardware + HTML + embedded C++**.
- Understood **Wi-Fi-based HTTP request handling on ESP8266**.
- Gained insights into **motor driver control and power distribution**.

Future Enhancements

- Add obstacle detection (Ultrasonic Sensor).
- Replace HTML with a dedicated Android/iOS App.
- Add video streaming using ESP32-CAM for live camera feed.
- Implement voice control through AI-based voice commands.

References

- YouTube Video (Basic Concept): (https://youtu.be/gU-CZP2nIwQ?si=nWQ2JBPl1zxwM_1u)
- ESP8266 (ESP-12) Documentation: (<https://docs.espressif.com/projects/esp8266/en/latest>)
- Arduino IDE with ESP8266: (<https://arduino-esp8266.readthedocs.io/>)



NFC IET
MULTAN, PAKISTAN

www.graficsea.com