Wi-Fi controlled Rover

**Abstract**

We have made is a Wi-fi controlled remote controlled car. Our car will have a web-based control interface, so you can drive it with any computer, tablet, or phone. For this project we will be using the ESP32 Wi-Fi module. This microcontroller comes along with a Wi-Fi as well as a Bluetooth adapter. We are going to use this module as the basis for a simple remote-controlled car. We will make use of the ESP32 built-in Wi-Fi capabilities to program a web-based user interface that will allow us to drive the car. We can use this design as the heart of a more sophisticated robot, taking advantage of the many functions of the ESP32 to add sensors and output devices. This car is an excellent project, as the resulting product is interesting and fun to use and can be expanded and extended in many different ways. Plus, there are a lot of fundamental concepts like assembly, soldering, and programming that are involved in building the robot. The traditional remote-controlled cars that we have been using till date uses radio signals to transmit the signal from the car to the remote. Which pretty often results in weak signal in longer distances, where as in our project we have used a ESP32 Wi-Fi module, we will be able to use Wi-Fi as the mode of interface between the user and the car. Also, the range in our Wi-Fi based remote controlled car in pretty much, much higher as compared to any regular radio-controlled car.

**Introduction**

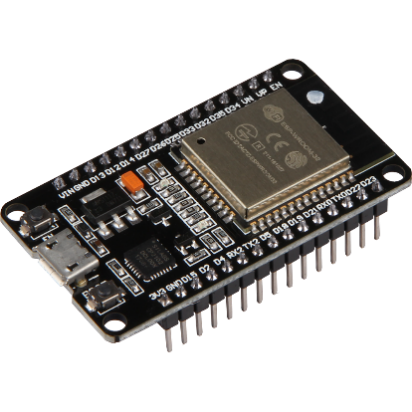
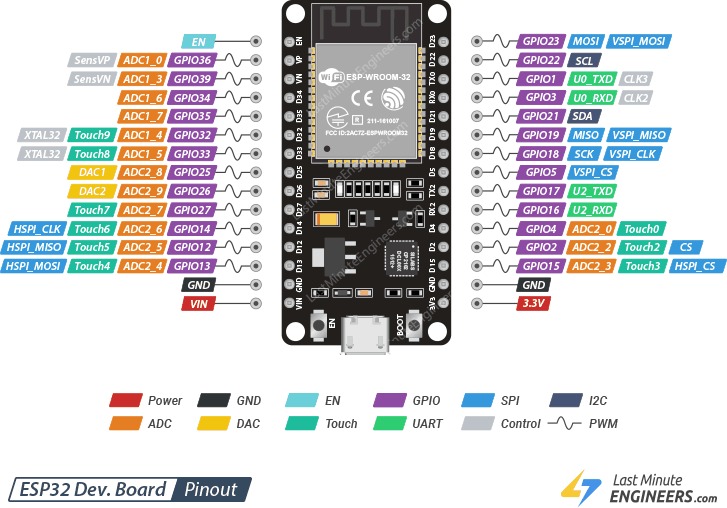
Driving remote-controlled cars is an entertaining hobby for both young and old alike. Although these toys may seem like modern inventions, they’ve actually been around for over half a century. Modern radio control systems offer nearly unlimited range, zero interference, and first-person vision piloting cameras, which allows you to control machines miles away, out of sight, with a remote. With the limitless nature of remote-control technology, it is amazing that smugglers, law enforcement, the military, and even pranksters are not exploiting this potential superpower to a greater degree.

The first known remote-controlled car was created in Italy by a company called Elettronica Giocattoli, or El-Gi, in 1966. It was a 1:12 scale model of the Ferrari 250LM and was later imported and sold to customers in the UK. By the early seventies, the market in RC cars had grown to include additional power sources to complement nitro fuel. Tamiya, a brand that is still well-known in toy car racing today, introduced their first electric on-road car in 1976. The model known as the Porsche 934 ushered in a wave of RC accessories, model kits, and RC racing paraphernalia. The 1980s brought about another surge in popularity for remote-controlled car racing. A lot of brands moved away from simple scale models of actual cars, and began to develop their own designs. High-performance cars also increased interest in organized racing events, and the 1:12 World Championship became a biennial event with up to 400 participants.

With the increase in power and reach of radio control technology, RCs took another step in their development. 4WD models and rock crawlers became a lot more popular, as better control and the increase in life of batteries made off-roading models prominent once more. Nowadays, although the hobby isn’t what it once was, there are still plenty of enthusiasts to drive it forward. With modern materials, more powerful engines, better batteries, and versatile control methods, the RC’s potential for entertainment is greater than ever. Recent developments include motion and gesture controls. Replacing the use of the traditional joystick, RC cars can now be driven by simply moving your hands in the direction you want it to go. Paired with the ability of 4 wheel-drive RCs to go off-roading, you can take the hobby to new heights.

Nowadays, wireless controlled car is commonly found as toy-grade radio controlled car with the focus on reducing production costs and also for hobby grade radio controlled which can be customized. The 2.4 GHz frequency radio has been used in the hobby-grade R/C Cars. Wi-Fi technology [JGP08] has been implemented in wireless controlled car with high data rate (54Mbit/s +), but also high-power consumption. It is used when you need to connect directly to the internet, such as an internet-of-things (IoTs) device, and have an external power source. Generally, the system is divided into three parts [DJ10]: the car-end, PC-receiver-end, and PC-control-end. In this part, we are going to discuss the required materials used for this project, those are for hardware and software specifications.

A. **Hardware Specifications:** There are some hardware materials needed to fulfil this project, like ESP32 Module as transmitter and receiver, Motor driver IC, DC motors and Power supply, chassis, and some other related equipment.

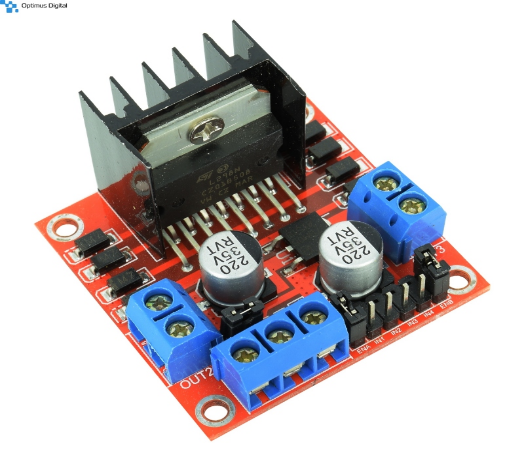
A.1. **NodeMCU:** NodeMCU is the development board containing the microcontroller ESP32 or ESP8266. In our project, we are using ESP32 microcontroller. ESP is the abbreviation of the Chinese company Espressif. ESP32 is a newer version of ESP8266 which has a faster dual core processor. A faster Wi-Fi and also Bluetooth support. It has also 10 touch sensors that utilise Halls Effect. In our project, this board basically acts as the server for the control system. It creates a Wi-Fi hotspot zone for the car and allows user to connect to it with the appropriate password. Then, it gives access to the HTML file coded in the firmware. The HTML creates a webpage for the control systems and has buttons for controlling the car. The Buttons send a specific value to the server which allows us to read that value from the microcontroller, which acts as the server here, and with the specific value we perform the actions.

A.2. **L298N Motor Driver IC:** The L298N is an IC (integrated circuit) that is designed to allow DC motor to drive on both directions. It has 15 pins just as any other 15-pin IC. This motor driver module consists of two main key components, these are L298N motor driver IC and a 78M05 5V regulator.

L298N motor driver IC

L298N is a high voltage, high current dual full-bridge motor driver IC. It accepts standard TTL logic levels (Control Logic) and controls inductive loads such as relays, solenoids, DC and Stepper motors. This is a 15 pin IC. According to the L298N datasheet, its operating voltage is +5 to +46V, and the maximum current allowed to draw through each output 3A. This IC has two enable inputs, these are provided to enable or disable the device independently of the input signals.

A black color heat sink is attached to the L298N IC of the module. A heat sink is a passive heat exchanger that transfers the heat generated by an electronic or a mechanical device to a fluid medium, often air or a liquid coolant.

78M05 5V Regulator

The module has an on-board 78M05 5V Voltage regulator. This Voltage regulator will be performed only when the 5V Enable jumper is placed. When the power supply is less than or equal to 12V, then the internal circuitry will be powered by the voltage regulator, and the 5V pin can be used as an output pin to power the microcontroller or other circuitry (sensor).

The jumper should not be placed when the power supply is greater than 12V and separate 5V should be given through 5V terminal to power the internal circuitry.

There are 4 input pins for L298N IN1, IN2, IN3 and IN4. There are two L298N Motor Driver used for this project, therefore 4 of the input pins from each connects to the esp32 module. The remaining pins connects the dc motors which runs the wheels.

A.3. **DC Motors and Chassis:** DC motor has voltage supply from 6 to 7V. There are four plastic gear box motors with wheels attached to it. Two 3.37V batteries are used to power the motors, and a power bank is used to power the ESP32. Since the battery case gives a voltage of 6 to 7V and the ESP32 supports a source of around 5V, a power bank of 5V output is used in order to keep the microcontroller safe from electrical faults. The material for the chassis is wood with four wheels for steering and driving. The motors are attached to it with strong glue in order to keep them steady.

B. **Software Specifications:** We are using a Linux laptop and using Arduino IDE. Arduino IDE is a software developed by Arduino. This software is free and can be availed at their website. Arduino IDE allows us to upload the firmware intro the Microprocessor which has an Arduino core. This has multiple microcontroller support including the whole Arduino Series and also the ESP series. We have also used the “Wi-Fi” library and also “AsyncTCP” and also “ESPAsyncWebServer”. These libraries help us use the Wi-Fi capabilities of the chip and also use the Server mode for the ESP.

C. **Working Mechanism:**

Now we are going to explain about how the project was implemented. First step is programming into the ESP32 and L298D IC. The second step will be making the controller. Here, there is no physical controller being used because this project utilizes HTML page to control the car. The logic is, first we connect to the access point made by ESP32 and then we navigate to the HTML page that function as controller, and in the HTML page there are some navigation buttons that can be pressed. After pressing the navigation button, it will send a command into the ESP32 which in turn will interact with L298D motor driver IC and will determine how the motor will act.

Below we have the code which has been uploaded to the ESP32 Wi-Fi module:

#include <Arduino.h>

#ifdef ESP32

#include <WiFi.h>

#include <AsyncTCP.h>

#elif defined(ESP8266)

#include <ESP8266WiFi.h>

#include <ESPAsyncTCP.h>

#endif

#include <ESPAsyncWebServer.h>

#define UP 2

#define DOWN 1

#define LEFT 4

#define RIGHT 3

#define STOP 0

#define FRONT\_RIGHT\_MOTOR 0

#define BACK\_RIGHT\_MOTOR 1

#define FRONT\_LEFT\_MOTOR 2

#define BACK\_LEFT\_MOTOR 3

#define FORWARD 1

#define BACKWARD -1

struct MOTOR\_PINS

{

int pinIN1;

int pinIN2;

};

std::vector<MOTOR\_PINS> motorPins =

{

{13, 12}, //FRONT\_RIGHT\_MOTOR

{14, 27}, //BACK\_RIGHT\_MOTOR

{26, 25}, //FRONT\_LEFT\_MOTOR

{33, 32}, //BACK\_LEFT\_MOTOR

};

const char\* ssid = "MyWiFiCar";

const char\* password = "12345678";

AsyncWebServer server(80);

AsyncWebSocket ws("/ws");

const char\* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no">

<style>

.arrows {

font-size:70px;

color:red;

}

.circularArrows {

font-size:80px;

color:blue;

}

td {

background-color:black;

border-radius:25%;

box-shadow: 5px 5px #888888;

}

td:active {

transform: translate(5px,5px);

box-shadow: none;

}

.noselect {

-webkit-touch-callout: none; /\* iOS Safari \*/

-webkit-user-select: none; /\* Safari \*/

-khtml-user-select: none; /\* Konqueror HTML \*/

-moz-user-select: none; /\* Firefox \*/

-ms-user-select: none; /\* Internet Explorer/Edge \*/

user-select: none; /\* Non-prefixed version, currently

supported by Chrome and Opera \*/

}

</style>

</head>

<body class="noselect" align="center" style="background-color:white">

<h1 style="color: teal;text-align:center;">Hash Include Electronics</h1>

<h2 style="color: teal;text-align:center;">Wi-Fi &#128663; Control</h2>

<table id="mainTable" style="width:400px;margin:auto;table-layout:fixed" CELLSPACING=10>

<tr>

<td></td>

<td ontouchstart='onTouchStartAndEnd("1")' ontouchend='onTouchStartAndEnd("0")'><span class="arrows" >&#8679;</span></td>

<td></td>

</tr>

<tr>

<td ontouchstart='onTouchStartAndEnd("3")' ontouchend='onTouchStartAndEnd("0")'><span class="arrows" >&#8678;</span></td>

<td></td>

<td ontouchstart='onTouchStartAndEnd("4")' ontouchend='onTouchStartAndEnd("0")'><span class="arrows" >&#8680;</span></td>

</tr>

<tr>

<td></td>

<td ontouchstart='onTouchStartAndEnd("2")' ontouchend='onTouchStartAndEnd("0")'><span class="arrows" >&#8681;</span></td>

<td></td>

</tr>

</table>

<script>

var webSocketUrl = "ws:\/\/" + window.location.hostname + "/ws";

var websocket;

function initWebSocket()

{

websocket = new WebSocket(webSocketUrl);

websocket.onopen = function(event){};

websocket.onclose = function(event){setTimeout(initWebSocket, 2000);};

websocket.onmessage = function(event){};

}

function onTouchStartAndEnd(value)

{

websocket.send(value);

}

window.onload = initWebSocket;

document.getElementById("mainTable").addEventListener("touchend", function(event){

event.preventDefault()

});

</script>

</body>

</html>

)HTMLHOMEPAGE";

void rotateMotor(int motorNumber, int motorDirection)

{

if (motorDirection == FORWARD)

{

digitalWrite(motorPins[motorNumber].pinIN1, HIGH);

digitalWrite(motorPins[motorNumber].pinIN2, LOW);

}

else if (motorDirection == BACKWARD)

{

digitalWrite(motorPins[motorNumber].pinIN1, LOW);

digitalWrite(motorPins[motorNumber].pinIN2, HIGH);

}

else

{

digitalWrite(motorPins[motorNumber].pinIN1, LOW);

digitalWrite(motorPins[motorNumber].pinIN2, LOW);

}

}

void processCarMovement(String inputValue)

{

Serial.printf("Got value as %s %d\n", inputValue.c\_str(), inputValue.toInt());

switch(inputValue.toInt())

{

case UP:

rotateMotor(FRONT\_RIGHT\_MOTOR, FORWARD);

rotateMotor(BACK\_RIGHT\_MOTOR, FORWARD);

rotateMotor(FRONT\_LEFT\_MOTOR, FORWARD);

rotateMotor(BACK\_LEFT\_MOTOR, FORWARD);

break;

case DOWN:

rotateMotor(FRONT\_RIGHT\_MOTOR, BACKWARD);

rotateMotor(BACK\_RIGHT\_MOTOR, BACKWARD);

rotateMotor(FRONT\_LEFT\_MOTOR, BACKWARD);

rotateMotor(BACK\_LEFT\_MOTOR, BACKWARD);

break;

case LEFT:

rotateMotor(FRONT\_RIGHT\_MOTOR, FORWARD);

rotateMotor(BACK\_RIGHT\_MOTOR, FORWARD);

rotateMotor(FRONT\_LEFT\_MOTOR, BACKWARD);

rotateMotor(BACK\_LEFT\_MOTOR, BACKWARD);

break;

case RIGHT:

rotateMotor(FRONT\_RIGHT\_MOTOR, BACKWARD);

rotateMotor(BACK\_RIGHT\_MOTOR, BACKWARD);

rotateMotor(FRONT\_LEFT\_MOTOR, FORWARD);

rotateMotor(BACK\_LEFT\_MOTOR, FORWARD);

break;

default:

rotateMotor(FRONT\_RIGHT\_MOTOR, STOP);

rotateMotor(BACK\_RIGHT\_MOTOR, STOP);

rotateMotor(FRONT\_LEFT\_MOTOR, STOP);

rotateMotor(BACK\_LEFT\_MOTOR, STOP);

break;

}

}

void handleRoot(AsyncWebServerRequest \*request)

{

request->send\_P(200, "text/html", htmlHomePage);

}

void handleNotFound(AsyncWebServerRequest \*request)

{

request->send(404, "text/plain", "File Not Found");

}

void onWebSocketEvent(AsyncWebSocket \*server,

AsyncWebSocketClient \*client,

AwsEventType type,

void \*arg,

uint8\_t \*data,

size\_t len)

{

switch (type)

{

case WS\_EVT\_CONNECT:

Serial.printf("WebSocket client #%u connected from %s\n", client->id(), client->remoteIP().toString().c\_str());

//client->text(getRelayPinsStatusJson(ALL\_RELAY\_PINS\_INDEX));

break;

case WS\_EVT\_DISCONNECT:

Serial.printf("WebSocket client #%u disconnected\n", client->id());

processCarMovement("0");

break;

case WS\_EVT\_DATA:

AwsFrameInfo \*info;

info = (AwsFrameInfo\*)arg;

if (info->final && info->index == 0 && info->len == len && info->opcode == WS\_TEXT)

{

std::string myData = "";

myData.assign((char \*)data, len);

processCarMovement(myData.c\_str());

}

break;

case WS\_EVT\_PONG:

case WS\_EVT\_ERROR:

break;

default:

break;

}

}

void setUpPinModes()

{

for (int i = 0; i < motorPins.size(); i++)

{

pinMode(motorPins[i].pinIN1, OUTPUT);

pinMode(motorPins[i].pinIN2, OUTPUT);

rotateMotor(i, STOP);

}

}

void setup(void)

{

setUpPinModes();

Serial.begin(115200);

WiFi.softAP(ssid, password);

IPAddress IP = WiFi.softAPIP();

Serial.print("AP IP address: ");

Serial.println(IP);

server.on("/", HTTP\_GET, handleRoot);

server.onNotFound(handleNotFound);

ws.onEvent(onWebSocketEvent);

server.addHandler(&ws);

server.begin();

Serial.println("HTTP server started");

}

void loop()

{

ws.cleanupClients();

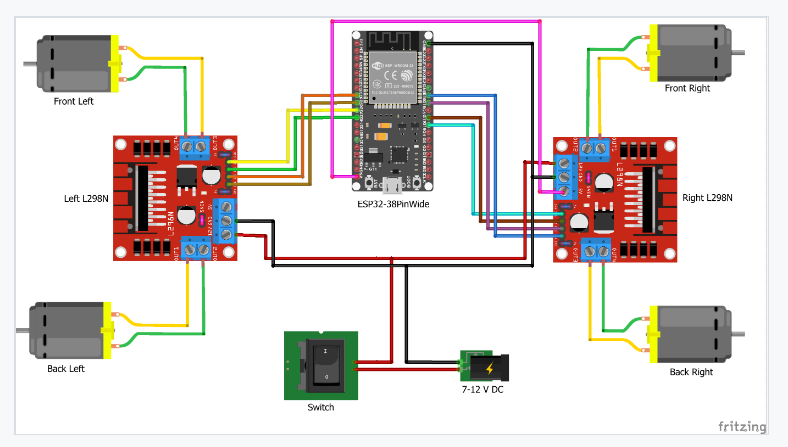
}

**Results and Discussions**

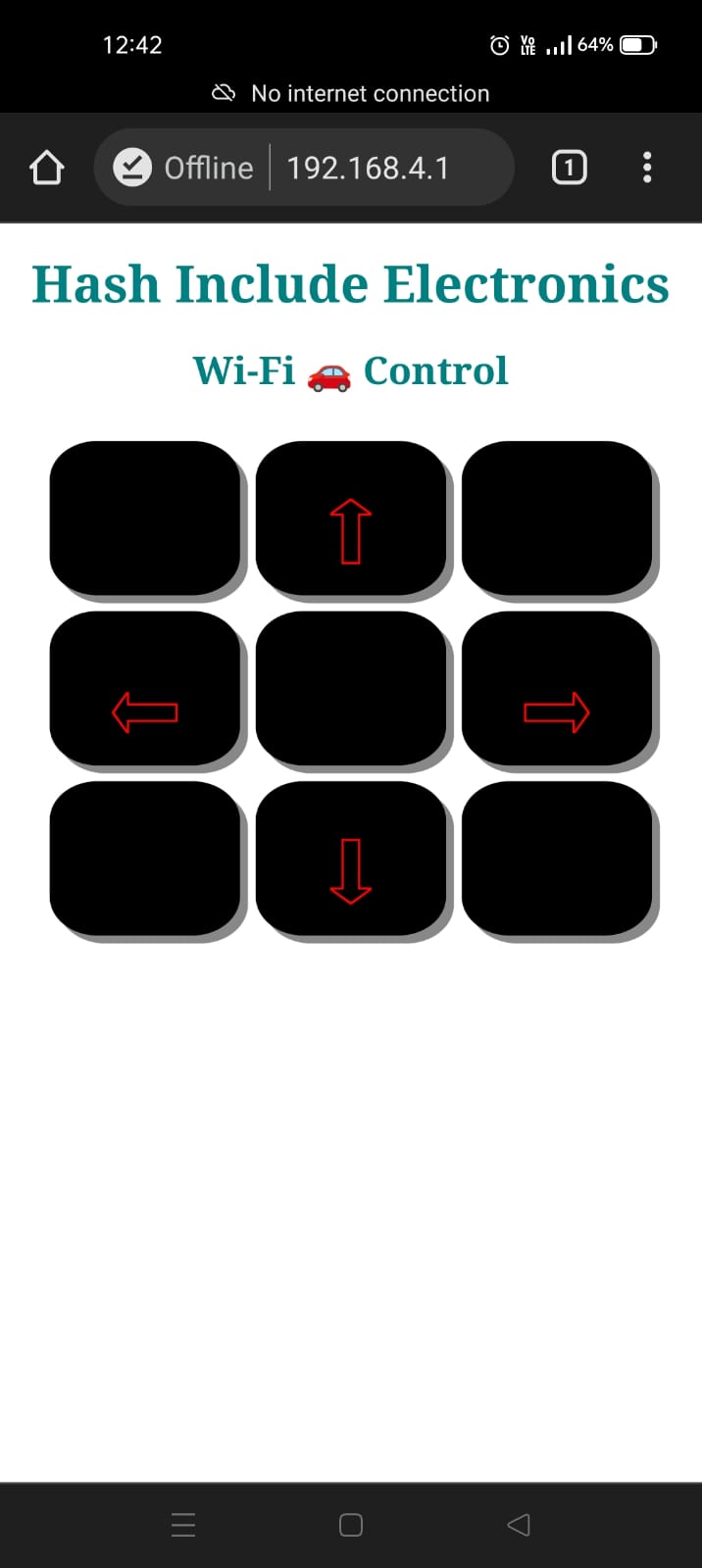
Before showing the experimental results and start to discuss it, here we explain first uploading the code to ESP32 then the connection of the Computer with ESP32, and Controller-Motor Connection.

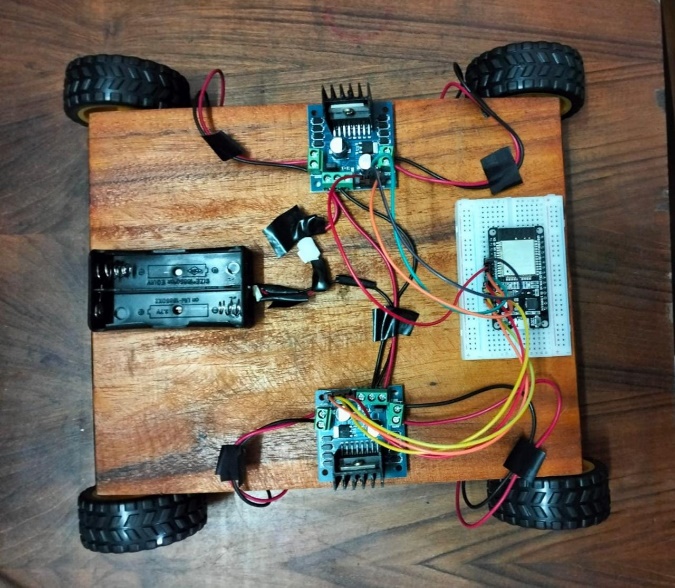
1. **Uploading the code to ESP32:** ESP32 connection will be tested using Arduino software. Plug in the USB power supply into the computer. Next, open the Arduino software, using the software’s serial monitor. If there is a connection between Arduino and ESP32, the serial monitor will send a sequence of message. The blue LED in ESP32 will also be turned on.
2. **Computer with ESP32 connection:** After uploading the code to ESP32, the next step is testing the connection between computer and ESP32. This task is simply done by connecting to the access point that ESP32 has made. The SSID is “MyWiFiCar”. After connecting into the access point, this step is done.

1. **Controller and Motor Connection:** After testing connection from computer and ESP32, now the project can be continued to setting up the controller and motor connection. Open the controller’s HTML page, and click on the buttons shown in the HTML page. Make sure that the computer is still connected to the ESP32 access point, and the testing phase will begin. There are four directions that needed to be tested. Forward, backward, left, and right.

****

User may know that the connection is successful by seeing the directions of the DC motors. If Straight button is pushed, both DC motors will turn in the same way. If Right button is pushed, only the right DC motor will turn. If Left button is pushed, only the left DC motor will turn. If Reverse button is pushed, both DC motors will turn in the same way but in opposite direction from the Straight and if no button is pushed, no DC motors will turn.

1. **Result:** The Wi-Fi controlled car runs smoothly with the HTML webpage we have created interacts well with the car and the signals are being transmitted efficiently. The microcontroller chip gets a heated by using for a prolonged period of time.



The car as of now

HTML page as in a mobile

**ACKNOWLEDGEMENT**

We would like to express our special thanks of gratitude to our teacher, Professor Apurba Paul and all the members of department of Computing who gave us the golden opportunity to do this wonderful project on the topic "Wifi\_Controlled\_Rover", which also helped us in during a lot of research and we all came to know about so many new things. His constant guidance and willingness to share his vast knowledge made us understand this project and helped in finalising this project within the limited time frame. We are really thankful to him.

**CERTIFICATE**

This is to certify that Swaraj Lahiri , Udayan Nath , Tushar Ghorui , Tina Das and Utsho Dutta of B.Tech 1st year of Department of Information Technology, Computer Science Engineering and Electrical Engineering of JIS college of Engineering has done the project entitled Wifi\_Controlled\_Rover, under the mandatory project course of B.Tech 2nd semester Examination 2022 under my supervision.

………………………

Prof. Apurba Paul

Department of Computing