SISTEMAS PROGRAMABLES

SEMESTRE ENERO – JUNIO 2024

DOCENTE: ROYCE RODRÍGUEZ

UNIDAD IV – PROGRAMACIÓN

DE MICROCONTROLADORES

EQUIPO 3

ROCÍO VANESA GARDEA HERNÁNDEZ 21550330

HÉCTOR ALEJANDRO RODRIGUEZ BARRÓN21550353

JOSÉ SEBASTIAN LÓPEZ IBARRA 21550362

ANDRÉS SAÉNZ OLIVAS 21550390

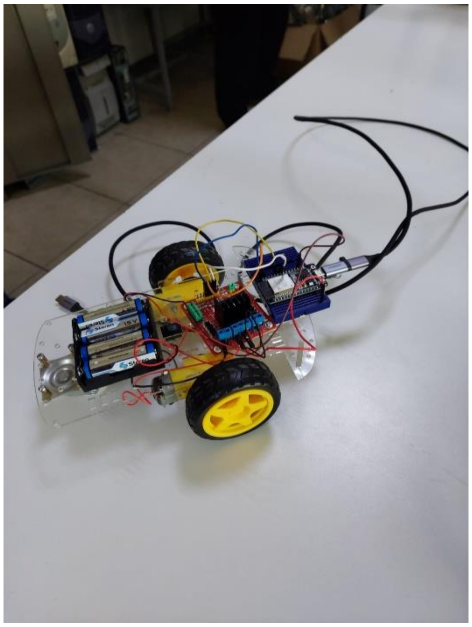
JORGE EDUARDO ESCOBAR BUGARINI 21550317

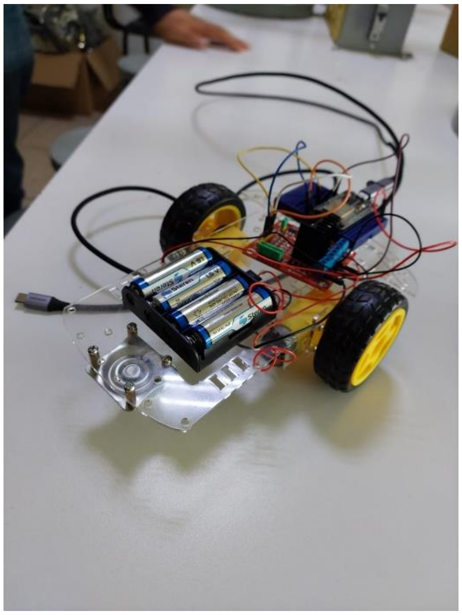
EDGAR GERARDO DELGADO CERRILLO 21550297

Contenido

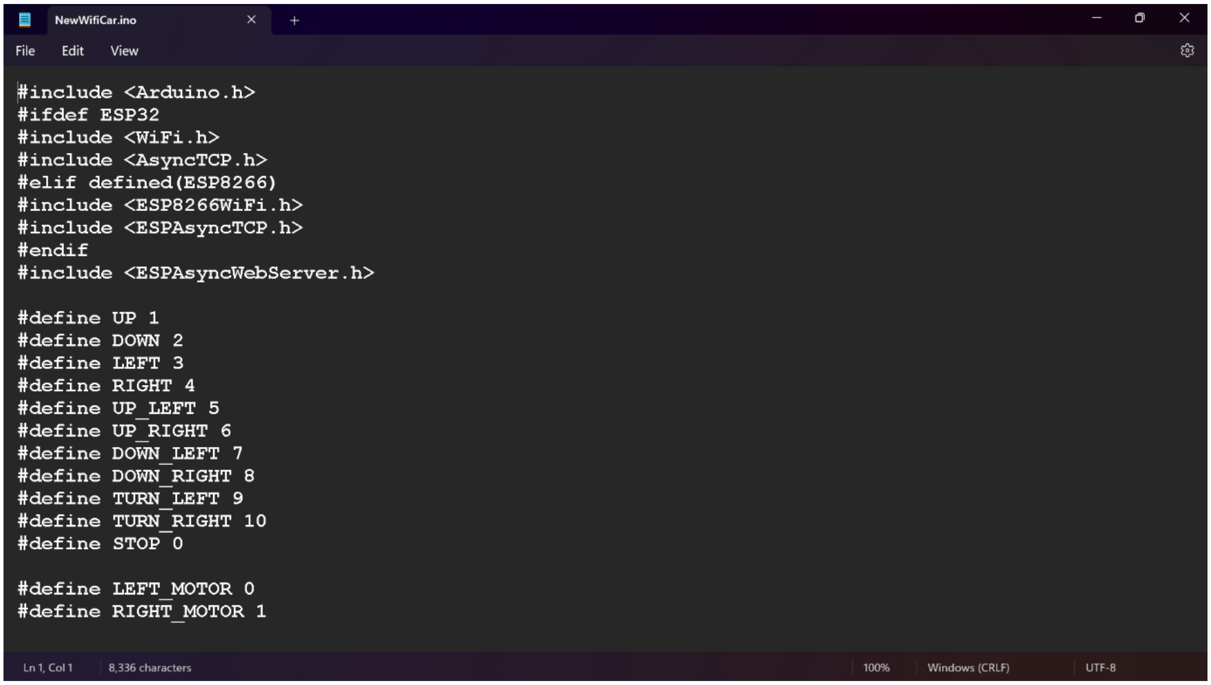
Carrito robot con Arduino

Cuenta con un microcontrolador Arduino SP32 el cual permite el control de los motores. Estos últimos están energizados mediante una batería de 9V y, el control del mismo es a través de un Smartphone vía Wi-Fi.

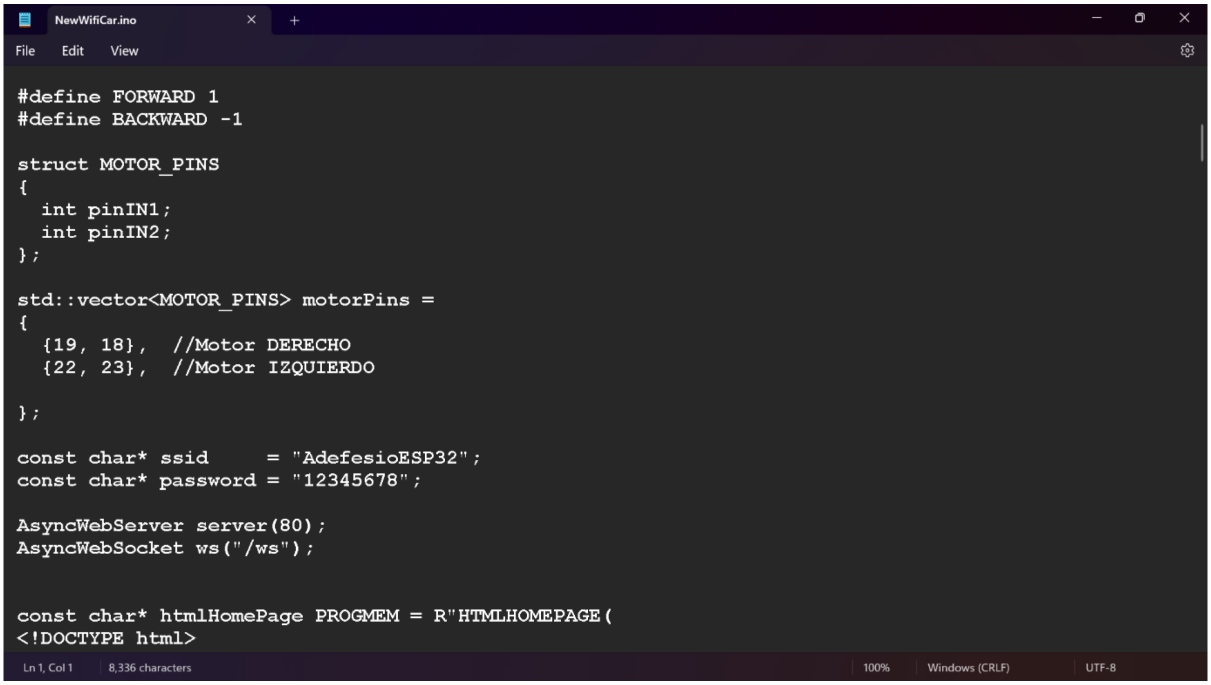




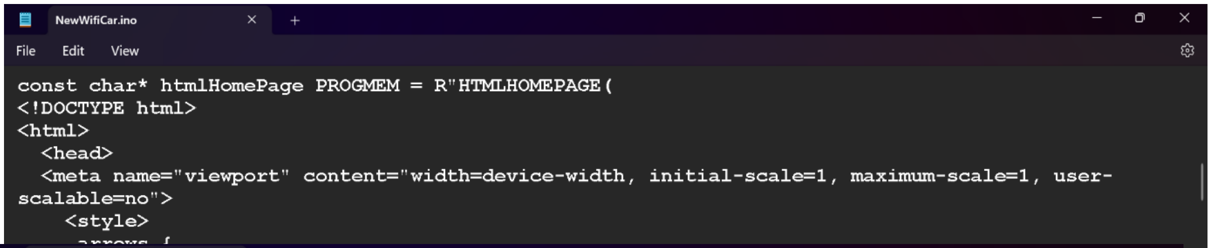
Código Arduino



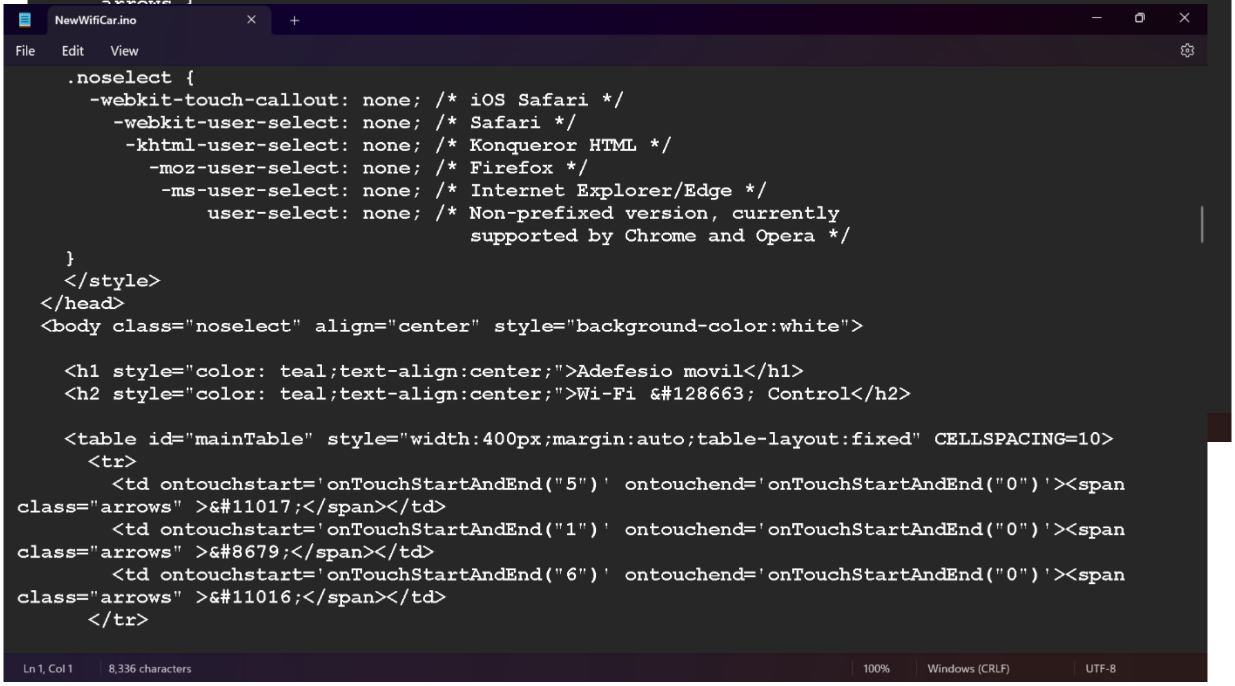
```
#include <Arduino.h>
#ifdef ESP32
#include <WiFi.h>
#include <AsyncTCP.h>
#elif defined(ESP8266)
#include <ESP8266WiFi.h>
#include <ESPAsyncTCP.h>
#endif
#include <ESPAsyncWebServer.h>

#define UP 1
#define DOWN 2
#define LEFT 3
#define RIGHT 4
#define UP_LEFT 5
#define UP_RIGHT 6
#define DOWN_LEFT 7
#define DOWN_RIGHT 8
#define TURN_LEFT 9
#define TURN_RIGHT 10
#define STOP 0

#define LEFT_MOTOR 0
#define RIGHT_MOTOR 1
```



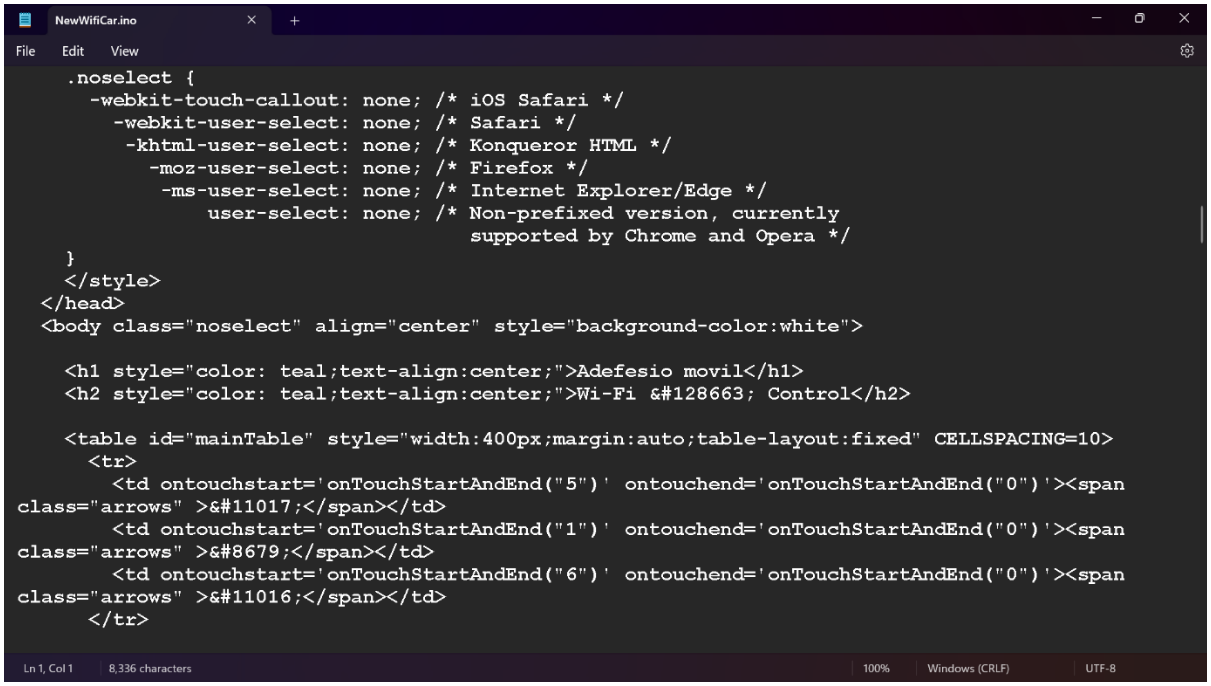
```
#define FORWARD 1
#define BACKWARD -1

struct MOTOR_PINS
{
  int pinIN1;
  int pinIN2;
};

std::vector<MOTOR_PINS> motorPins =
{
  {19, 18},  //Motor DERECHO
  {22, 23},  //Motor IZQUIERDO

};

const char* ssid     = "AdefesioESP32";
const char* password = "12345678";

AsyncWebServer server(80);
AsyncWebSocket ws("/ws");


const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
<!DOCTYPE html>
```



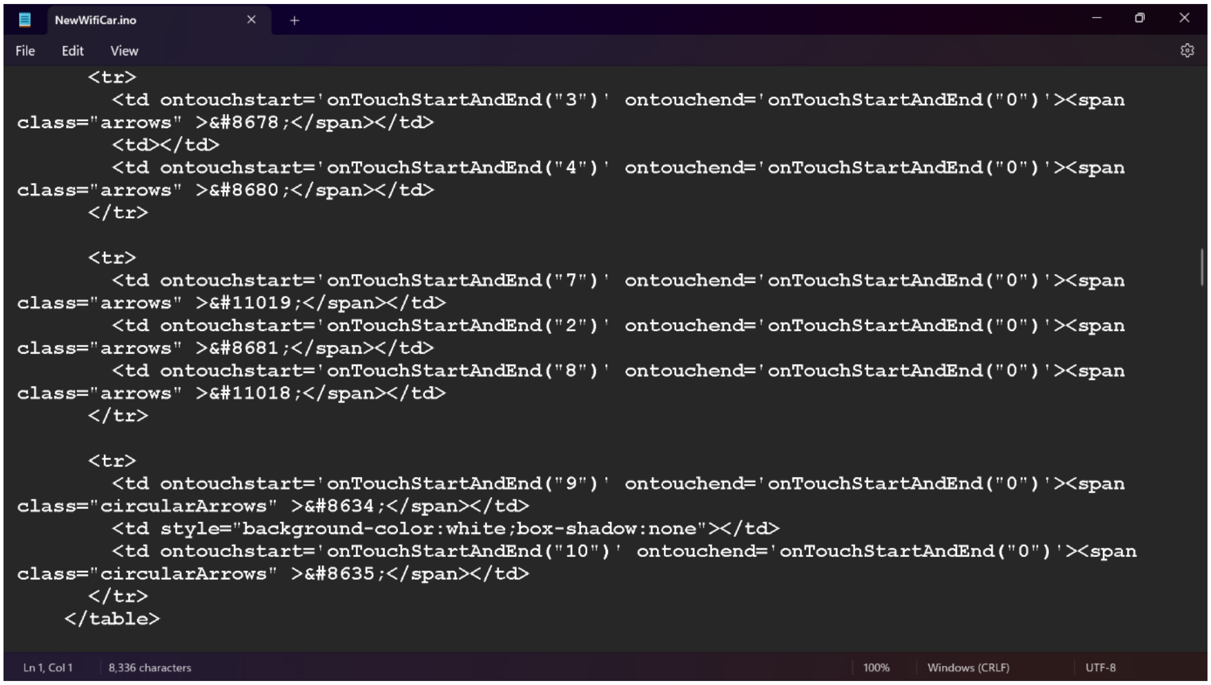
```
const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
<!DOCTYPE html>
<html>
  <head>
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no">
    <style>
    .arrows {
```



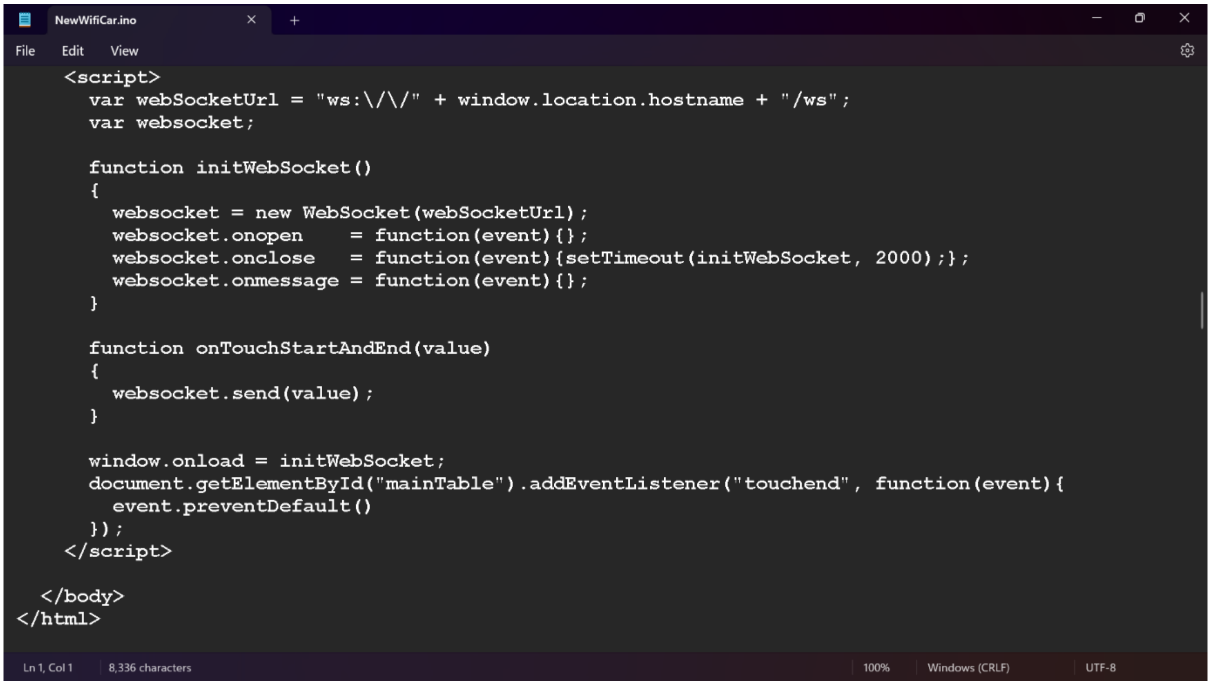
```
    .noselect {
      -webkit-touch-callout: none; /* iOS Safari */
        -webkit-user-select: none; /* Safari */
         -khtml-user-select: none; /* Konqueror HTML */
           -moz-user-select: none; /* Firefox */
            -ms-user-select: none; /* Internet Explorer/Edge */
                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;">Adefesio movil</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 ontouchstart='onTouchStartAndEnd("5")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#11017;</span></td>
        <td ontouchstart='onTouchStartAndEnd("1")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#8679;</span></td>
        <td ontouchstart='onTouchStartAndEnd("6")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#11016;</span></td>
      </tr>
```



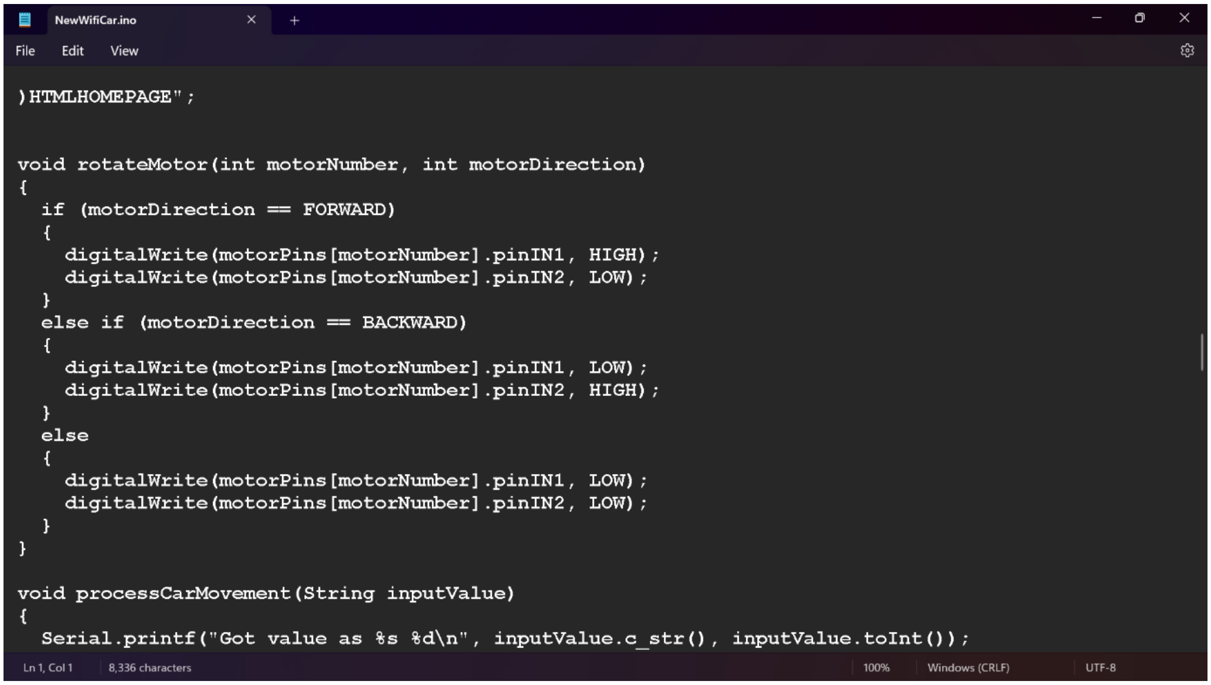
```
    .noselect {
      -webkit-touch-callout: none; /* iOS Safari */
        -webkit-user-select: none; /* Safari */
         -khtml-user-select: none; /* Konqueror HTML */
           -moz-user-select: none; /* Firefox */
            -ms-user-select: none; /* Internet Explorer/Edge */
                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;">Adefesio movil</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 ontouchstart='onTouchStartAndEnd("5")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#11017;</span></td>
        <td ontouchstart='onTouchStartAndEnd("1")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#8679;</span></td>
        <td ontouchstart='onTouchStartAndEnd("6")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#11016;</span></td>
      </tr>
```



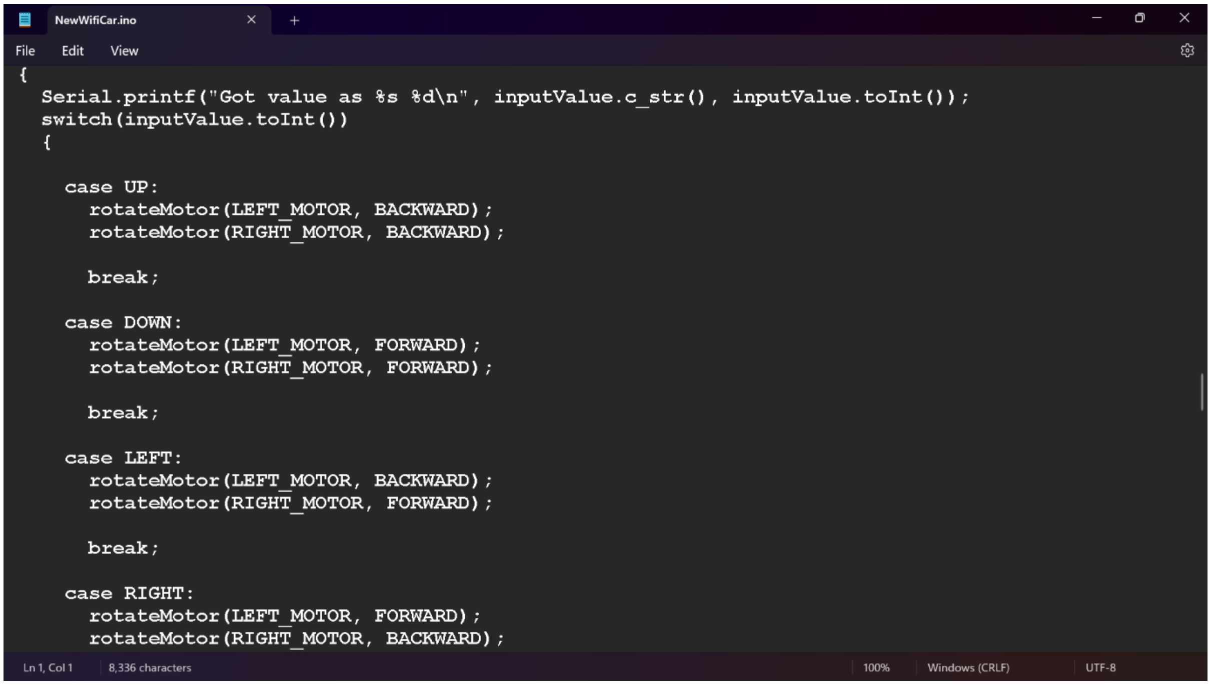
```
      <tr>
        <td ontouchstart='onTouchStartAndEnd("3")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#8678;</span></td>
        <td></td>
        <td ontouchstart='onTouchStartAndEnd("4")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#8680;</span></td>
      </tr>

      <tr>
        <td ontouchstart='onTouchStartAndEnd("7")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#11019;</span></td>
        <td ontouchstart='onTouchStartAndEnd("2")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#8681;</span></td>
        <td ontouchstart='onTouchStartAndEnd("8")' ontouchend='onTouchStartAndEnd("0")'><span
class="arrows" >&#11018;</span></td>
      </tr>

      <tr>
        <td ontouchstart='onTouchStartAndEnd("9")' ontouchend='onTouchStartAndEnd("0")'><span
class="circularArrows" >&#8634;</span></td>
        <td style="background-color:white;box-shadow:none"></td>
        <td ontouchstart='onTouchStartAndEnd("10")' ontouchend='onTouchStartAndEnd("0")'><span
class="circularArrows" >&#8635;</span></td>
      </tr>
    </table>
```



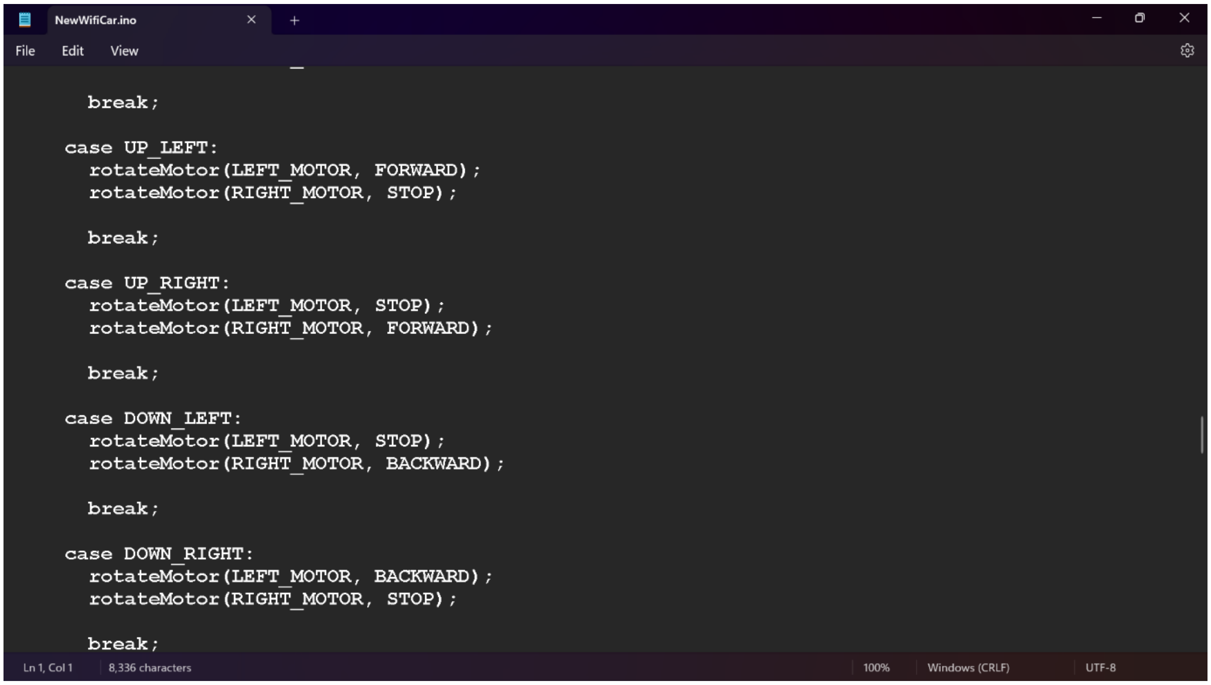
```
    <script>
      var webSocketUrl = "ws:\/\/" + window.location.hostname + "/ws";
      var websocket;

      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>
```



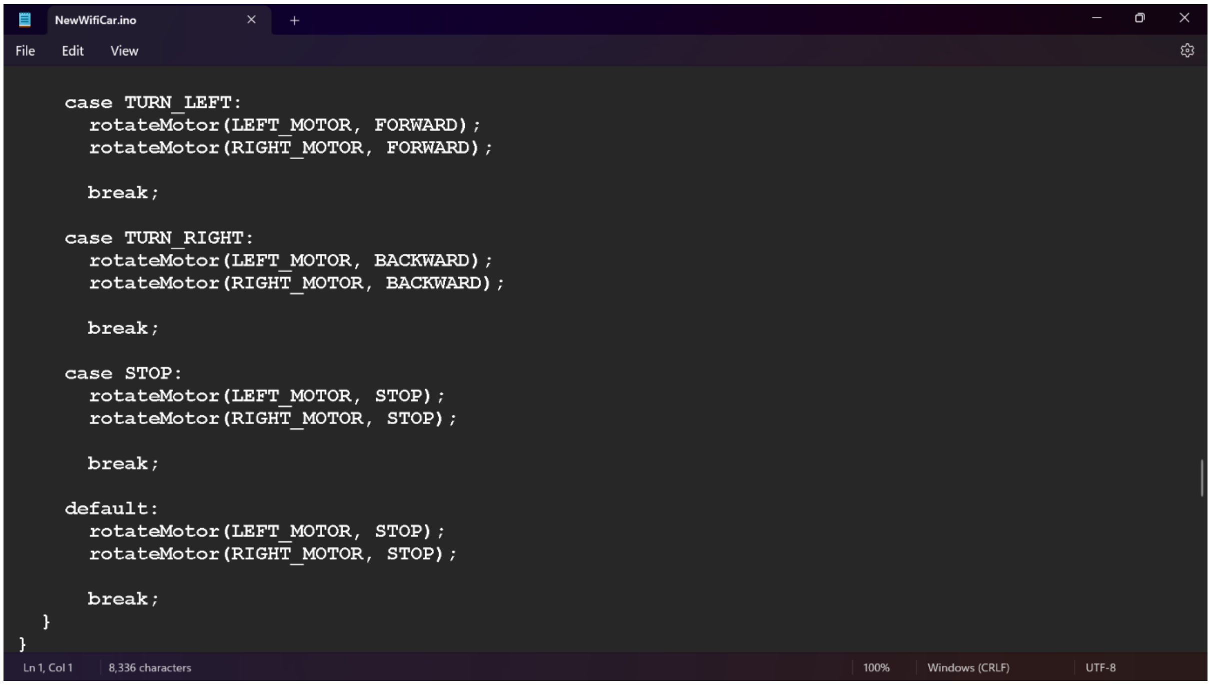
```
)HTMLHOMEPAGE";


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());
```



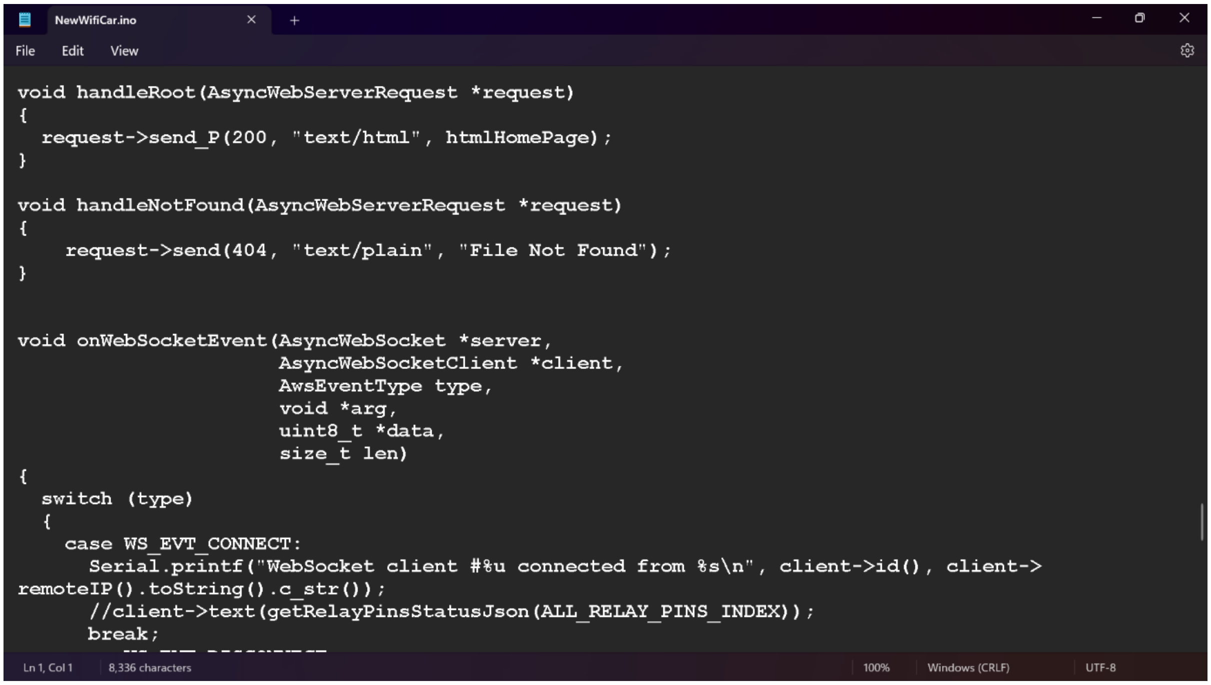
```
{
  Serial.printf("Got value as %s %d\n", inputValue.c_str(), inputValue.toInt());
  switch(inputValue.toInt())
  {

    case UP:
      rotateMotor(LEFT_MOTOR, BACKWARD);
      rotateMotor(RIGHT_MOTOR, BACKWARD);

      break;

    case DOWN:
      rotateMotor(LEFT_MOTOR, FORWARD);
      rotateMotor(RIGHT_MOTOR, FORWARD);

      break;

    case LEFT:
      rotateMotor(LEFT_MOTOR, BACKWARD);
      rotateMotor(RIGHT_MOTOR, FORWARD);

      break;

    case RIGHT:
      rotateMotor(LEFT_MOTOR, FORWARD);
      rotateMotor(RIGHT_MOTOR, BACKWARD);
```



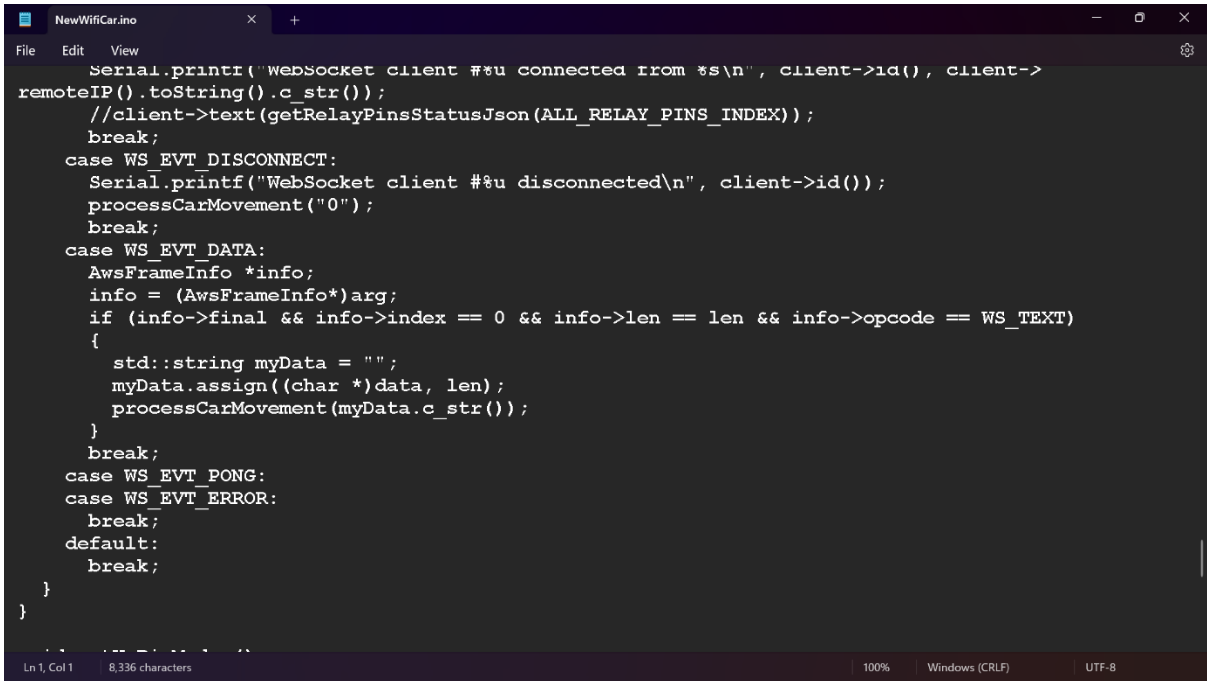
```
      break;

    case UP_LEFT:
      rotateMotor(LEFT_MOTOR, FORWARD);
      rotateMotor(RIGHT_MOTOR, STOP);

      break;

    case UP_RIGHT:
      rotateMotor(LEFT_MOTOR, STOP);
      rotateMotor(RIGHT_MOTOR, FORWARD);

      break;

    case DOWN_LEFT:
      rotateMotor(LEFT_MOTOR, STOP);
      rotateMotor(RIGHT_MOTOR, BACKWARD);

      break;

    case DOWN_RIGHT:
      rotateMotor(LEFT_MOTOR, BACKWARD);
      rotateMotor(RIGHT_MOTOR, STOP);

      break;
```



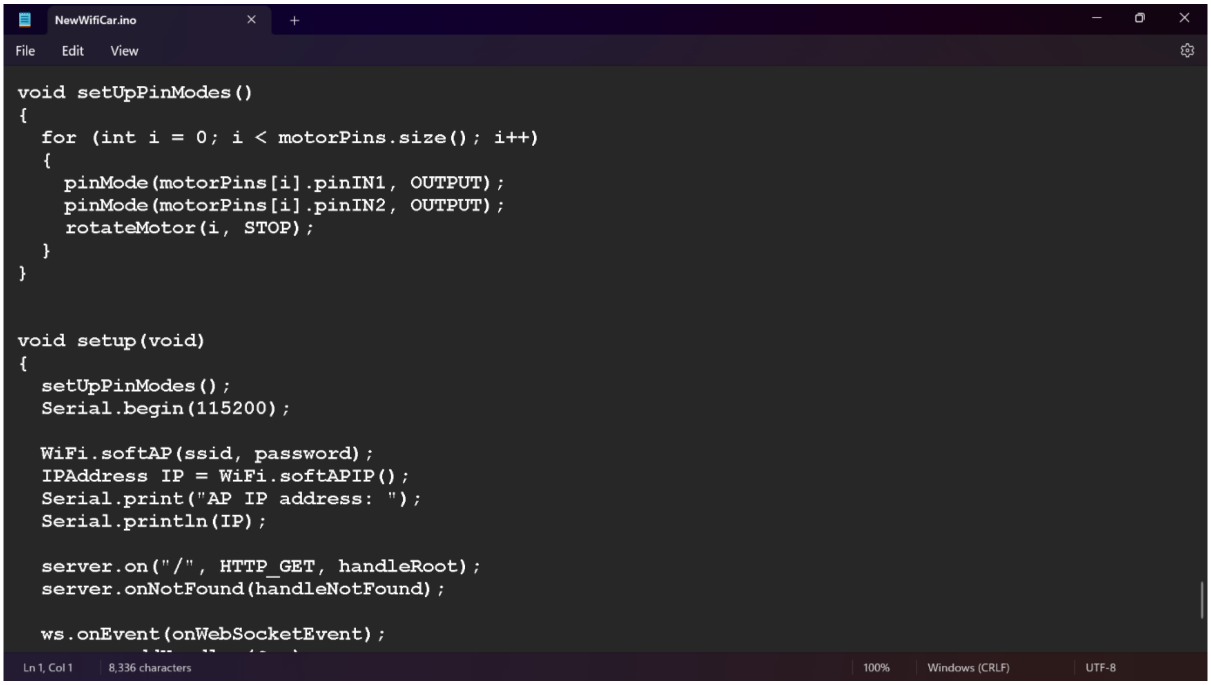
```
    case TURN_LEFT:
      rotateMotor(LEFT_MOTOR, FORWARD);
      rotateMotor(RIGHT_MOTOR, FORWARD);

      break;

    case TURN_RIGHT:
      rotateMotor(LEFT_MOTOR, BACKWARD);
      rotateMotor(RIGHT_MOTOR, BACKWARD);

      break;

    case STOP:
      rotateMotor(LEFT_MOTOR, STOP);
      rotateMotor(RIGHT_MOTOR, STOP);

      break;

    default:
      rotateMotor(LEFT_MOTOR, STOP);
      rotateMotor(RIGHT_MOTOR, STOP);

      break;
  }
}
```



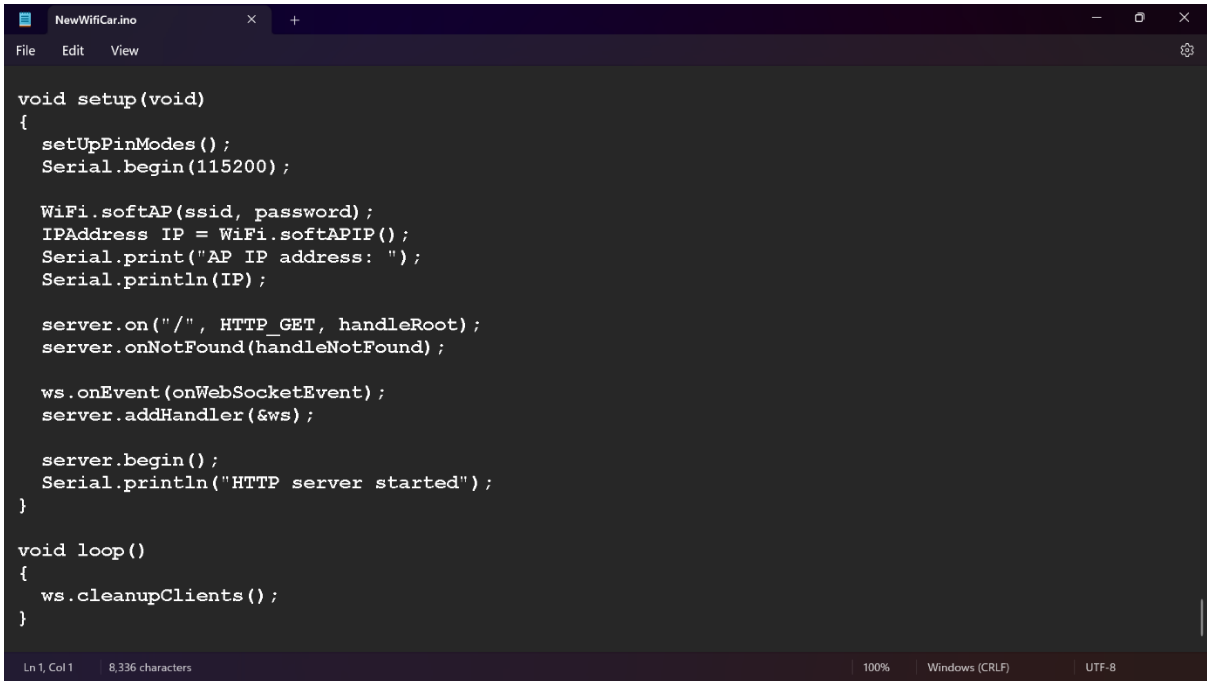
```
void handleRoot(AsyncWebServerRequest *request)
{
  request->send_P(200, "text/html", htmlHomePage);
}

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 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);
```



```
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();
}
```