INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

TÉCNICO EM ELETROELETRÔNICA

EMANUEL JOSÉ FRAZÃO GARCIA

GUSTAVO VIEIRA DAS NEVES MUNIZ

JOÃO PAULO CALEGARE

KAIQUE LEONARDO RODRIGUES

LEONARDO HENRIQUE CAMILO PATRÃO

RICARDO GABRIEL DE CAMPOS RODRIGUES

ROBÔ DE EXPLORAÇÃO

SISTEMA ROBÓTICO DE EXPLORAÇÃO E SUAS APLICAÇÕES TECNOLOGICAS

SOROCABA

2025

EMANUEL JOSÉ FRAZÃO GARCIA

GUSTAVO VIEIRA DAS NEVES MUNIZ

JOÃO PAULO CALEGARE

KAIQUE LEONARDO RODRIGUES

LEONARDO HENRIQUE CAMILO PATRÃO

RICARDO GABRIEL DE CAMPOS RODRIGUES

ROBÔ DE EXPLORAÇÃO

SISTEMA ROBÓTICO DE EXPLORAÇÃO E SUAS APLICAÇÕES TECNOLOGICAS

Trabalho apresentado ao IFSP - Câmpus Sorocaba como requisito para conclusão do Ensino Médio integrado ao curso Técnico em Eletroeletrônica.

Orientadores: Prof. Marcelo Custodio Cardozo

Prof. Sidnei de Oliveira Nascimento

SOROCABA

2025

RESUMO

O presente trabalho trata do desenvolvimento de um projeto denominado Robô de Exploração, que tem como objetivo possibilitar o monitoramento de áreas remotas por meio do controle à distância. O projeto utiliza componentes eletrônicos acessíveis e de baixo custo, como o microcontrolador ESP32, que possui conectividade Wi-Fi e Bluetooth integradas, permitindo o controle remoto por meio de um smartphone, via interface web. Além disso, o robô conta com o módulo de câmera ESP32-CAM e sensores de distância, que viabilizam a captação de imagens e a detecção de obstáculos durante o deslocamento. A proposta busca aliar simplicidade e funcionalidade para aplicações em ambientes de difícil acesso ou que oferecem risco à presença humana. O embasamento teórico foi construído com base em materiais técnicos e projetos similares disponíveis na literatura e em repositórios online. A metodologia utilizada inclui pesquisa aplicada, de abordagem qualitativa e natureza exploratória, com foco no desenvolvimento e validação prática do protótipo. Os resultados obtidos demonstram que o robô é funcional, sendo capaz de se deslocar e transmitir imagens em tempo real, atendendo aos objetivos propostos.

Palavras-chaves: Robótica; ESP32; Controle remoto; Automação; Monitoramento remoto.

ABSTRACT

This work addresses the development of a project called *Exploration Robot*, which aims to enable the monitoring of remote areas through remote control. The project uses accessible and low-cost electronic components, such as the ESP32 microcontroller, which has integrated Wi-Fi and Bluetooth connectivity, allowing remote control via a smartphone through a web interface. In addition, the robot is equipped with the ESP32-CAM camera module and distance sensors, which enable image capture and obstacle detection during movement. The proposal seeks to combine simplicity and functionality for applications in hard-to-reach environments or those that pose a risk to human presence. The theoretical foundation was built based on technical materials and similar projects available in the literature and online repositories. The methodology used includes applied research with a qualitative and exploratory approach, focusing on the development and practical validation of the prototype. The results obtained demonstrate that the robot is functional, capable of moving and transmitting images in real time, meeting the proposed objectives.

Keywords: Robotics; ESP32; Remote control; Automation; Remote monitoring.

SUMÁRIO

1 INTRODUÇÃO

A exploração de ambientes de difícil acesso, como cavernas, florestas densas ou áreas com terrenos inóspitos, tem se tornado uma tarefa desafiadora e perigosa para seres humanos. Com o avanço da tecnologia, robôs têm se mostrado soluções eficazes para esses desafios, sendo capazes de acessar e explorar locais que seriam inacessíveis ou arriscados para pessoas. Este projeto visa o desenvolvimento de um robô de exploração de baixo custo, projetado para operar em ambientes restritos e de difícil navegação, utilizando tecnologias acessíveis e eficientes.

O robô proposto será equipado com sensores avançados e um módulo de câmera ESP-32 CAM, que permitirá uma visão ampla e detalhada da área explorada. A câmera e os sensores fornecerão dados em tempo real, possibilitando a análise do ambiente de forma precisa e segura. Com isso, será possível realizar a exploração de locais como cavernas e florestas fechadas, onde a visibilidade é limitada e a segurança das equipes humanas é comprometida. A interação com o robô ocorrerá de maneira remota, por meio de um módulo Wi-Fi junto com um arduino, que possibilitará o controle do robô via site, tornando o sistema altamente acessível e intuitivo para o usuário.

Além disso, o projeto destaca-se pela escolha de componentes de baixo custo, o que torna a solução mais acessível financeiramente. A utilização de materiais e tecnologias econômicas, como o módulo ESP-32 CAM, oferece um equilíbrio entre desempenho e custo, permitindo que o robô seja uma alternativa viável para diversas aplicações, desde projetos educacionais até operações de exploração em locais de difícil acesso. Esse fator torna o robô desenvolvido neste projeto mais acessível do que as opções comerciais disponíveis no mercado, muitas das quais envolvem altos custos de aquisição e manutenção.

A proposta deste trabalho teve como base o projeto publicado no canal Hash Include Electronics (2022), que apresenta a construção de um robô controlado por Wi-Fi com ESP32. A partir dessa base, este projeto propôs adaptações, modificações no código e incrementos na estrutura, a fim de adequá-lo à finalidade de exploração remota em ambientes desafiadores.

Este trabalho está estruturado em capítulos que abordam, respectivamente, a fundamentação teórica sobre tecnologias aplicadas à robótica de exploração, a metodologia adotada para o desenvolvimento do robô, a construção do protótipo, os testes realizados e os resultados obtidos, seguidos das considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, apresenta-se o embasamento técnico que orienta o desenvolvimento do robô de exploração. São abordados os principais conceitos relacionados à robótica móvel, ao funcionamento do microcontrolador Arduino, à comunicação via Wi-Fi e à utilização de interfaces web para controle remoto.

2.1 Robótica Móvel

A robótica móvel estuda robôs que possuem capacidade de locomoção e são utilizados para atuar em ambientes variados. Seu uso é comum em áreas como resgate, mineração, exploração espacial e patrulhamento remoto.

2.2 Microcontrolador Arduino

O Arduino é um microcontrolador com um modulo Wi-Fi e Bluetooth. Possui diversos pinos, suporte a PWM e recursos de baixo consumo, sendo ideal para projetos IoT e robótica.

2.3 Comunicação Wi-Fi e WebSocket

A comunicação via WebSocket permite que o robô se conecte a um navegador web em tempo real, recebendo comandos de movimentação e enviando imagens captadas pela câmera.

2.4 Interface HTML

O controle do robô ocorre por meio de botões HTML interativos, que enviam comandos ao servidor hospedado no ESP32-CAM.

3 DESENVOLVIMENTO

3.1 FUNCIONAIS

3.1.1 Hardware

- **ESP32-CAM**: responsável por capturar imagens e transmitir via Wi-Fi.
- **Arduino**: realiza o processamento dos dados e controle dos motores.
- **Ponte H (L298N)**: utilizada para controlar o sentido e a velocidade dos motores DC.
- **Motores DC**: proporcionam a locomoção do robô.
- **Servo motor**: opcional, pode ser usado para movimentar a câmera ou sensor.
- **Dispositivo móvel ou computador**: usado para acessar a interface web e controlar o robô remotamente via Wi-Fi.

3.1.2 Software

- Arduino IDE: ambiente de programação usado no desenvolvimento do código.
- Bibliotecas utilizadas:
  - ESPAsyncWebServer.h: permite criar um servidor web assíncrono.
  - AsyncTCP.h: utilizada para comunicação assíncrona com ESP32.
  - ESP32Servo.h: permite controlar servos com precisão.
  - WiFi.h: para criação da rede Wi-Fi local.
  - Arduino.h: biblioteca base do Arduino.

3.1.3 Arquitetura da solução

A arquitetura do projeto foi organizada nas seguintes etapas:

- **Planejamento da Missão:** definição do ambiente a ser explorado e dos dados a serem coletados com a câmera.
- **Navegação Remota:** o robô é controlado via Wi-Fi por meio de uma interface em site, acessível por celular ou computador.
- **Coleta de Dados:** a câmera do ESP32-CAM captura imagens do ambiente em tempo real.
- **Análise Visual:** o operador humano visualiza os dados no site e avalia os obstáculos ou pontos de interesse.
- **Tomada de Decisões:** com base nas imagens recebidas, o operador controla os motores para movimentar o robô com segurança.

3.2 TECNOLOGIAS EMPREGADAS

3.2.1 Hardware

- ESP32-CAM;
- Módulo Arduino;
- Modulo Wi-fi e Bluetooth;
- Ponte H (L298N);

- Motores DC;
- Servo motor;
- Fonte de alimentação (bateria ou power bank);
- Celular ou computador com navegador web.

3.2.2 Linguagens e Ambientes

- **Linguagem:** C++;
- **Ambiente:** Arduino IDE;
- **Comunicação:** Wi-Fi com WebSocket;
- **Interface:** HTML + JavaScript no código embarcado.

3.3 DIAGRAMA DE BLOCOS FUNCIONAL

ESP32-CAM: capta imagens e envia via Wi-Fi para o navegador do operador.

Arduino: recebe comandos de movimento do site e aciona os motores via ponte H.

Ponte H: ativa os motores com controle de direção e velocidade.

Interface Web: controla o robô via botões de setas e barra de velocidade.

Dispositivo Remoto: envia comandos e recebe vídeo/imagens.

Podemos observar esse diagrama em forma de imagem no anexo F da página 23 deste documento.

3.4 DIAGRAMA ELETRÔNICO

O diagrama eletrônico do projeto apresenta a interligação entre o ESP32-CAM, a ponte H (L298N), os motores DC e a alimentação elétrica. A conexão com o dispositivo controlador é realizada via Wi-Fi. Os pinos do Arduino estão configurados para controle dos motores e leitura de sensores, além do envio de imagens por meio do módulo de câmera embarcado.

Podemos observar isso no anexo B na página 21 deste documento.

3.5 LÓGICA DE PROGAMAÇÃO

A lógica do sistema foi desenvolvida utilizando a linguagem C++ na plataforma Arduino IDE. O código é responsável por configurar o Wi-Fi, controlar os motores via ponte H, receber comandos pela interface web e transmitir imagens da câmera.

3.5.1 Criação da Rede Wi-Fi e Servidor Web

```
const char* ssid     = "MyWiFiCar";

const char* password = "12345678";

WiFi.softAP(ssid, password);

IPAddress IP = WiFi.softAPIP();

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

Serial.println(IP);
```

```
server.on("/", HTTP_GET, handleRoot);
server.onNotFound(handleNotFound);
server.begin();
```

Esse trecho cria uma rede Wi-Fi local, permitindo que um dispositivo móvel acesse o site hospedado pelo ESP32-CAM, por onde será feito o controle do robô.

3.5.2 Controle de Direção com WebSocket

```
void moveCar(int inputValue) {
 switch(inputValue)  {
   case UP:
    rotateMotor(RIGHT_MOTOR, FORWARD);
    rotateMotor(LEFT_MOTOR, FORWARD);
    break;
   case DOWN:
    rotateMotor(RIGHT_MOTOR, BACKWARD);
    rotateMotor(LEFT_MOTOR, BACKWARD);
    break;
   case LEFT:
    rotateMotor(RIGHT_MOTOR, FORWARD);
    rotateMotor(LEFT_MOTOR, BACKWARD);
    break;
   case RIGHT:
    rotateMotor(RIGHT_MOTOR, BACKWARD);
    rotateMotor(LEFT_MOTOR, FORWARD);
    break;
   case STOP:
   default:
    rotateMotor(RIGHT_MOTOR, STOP);
    rotateMotor(LEFT_MOTOR, STOP);
    break;  }}
```

Essa função interpreta os comandos recebidos do site e movimenta o robô conforme a direção solicitada. Ela traduz os comandos recebidos via WebSocket em ações físicas dos motores, controlando a movimentação do robô para frente, traz, esquerda e direita.

3.5.3 Controle da Velocidade (PWM)

```
const int PWMFreq = 1000;

const int PWMResolution = 8;

const int PWMSpeedChannel = 4;

ledcSetup(PWMSpeedChannel, PWMFreq, PWMResolution);

ledcAttachPin(motorPins[i].pinEn, PWMSpeedChannel);
```

Essas linhas configuram o canal PWM para controlar a velocidade dos motores com base no valor enviado pelo controle web.

3.5.4 Sensor Ultrassônico – Medição de Distância

O sensor ultrassônico é responsável por medir a distância entre o robô e obstáculos à sua frente. Ele funciona emitindo um pulso de ultrassom por meio do pino trig, e medindo o tempo que o eco leva para retornar ao pino echo. Com base nesse tempo, o microcontrolador calcula a distância utilizando a fórmula:

$$\text{Distância (cm)} = \frac{\text{Duração do pulso} \times \text{Velocidade do som}}{2}$$

No código, esse cálculo é realizado da seguinte forma:

```
digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distanceCm = duration * SOUND_SPEED / 2;
```

O valor calculado é então exibido no monitor serial, possibilitando a análise da distância em tempo real:

```
Serial.print("Distância (cm): ");

Serial.println(distanceCm);
```

Essa funcionalidade permite que o robô detecte obstáculos durante o deslocamento, servindo de base para futuras implementações de autonomia, como parada automática ou desvio de obstáculos.

3.5.5 Controle de LED PWM – Indicador Visual de Atividade

Além do controle dos motores, o projeto implementa um canal PWM (Modulação por Largura de Pulso) dedicado a um LED. Esse LED funciona como indicador visual da velocidade e do estado de operação do robô.

O brilho do LED varia proporcionalmente ao valor de PWM aplicado aos motores, permitindo que o operador visualize a intensidade de movimento do robô em tempo real.

A configuração e uso do LED PWM são realizados conforme o trecho a seguir:

```
const int ledPWMChannel = 5;

const int ledPin = 25;

ledcSetup(ledPWMChannel, ledFreq, ledResolution);

ledcAttachPin(ledPin, ledPWMChannel);

ledcWrite(ledPWMChannel, valueInt);
```

Essa implementação contribui para o monitoramento do sistema, funcionando como um feedback visual simples e eficiente do controle de velocidade e funcionamento geral do robô.

3.5.6 Interface Web (HTML)

Este elemento representa um botão de controle da interface HTML que, ao ser acionado, envia comandos via WebSocket com um par chave-valor (MoveCar,1 ou MoveCar,0). Esses comandos são lidos pela função moveCar (Seção 3.5.2), que por sua vez, aciona os motores.

3.5.7 Código-fonte Completo

O código-fonte do projeto teve como inspiração o material publicado no canal Hash Include Electronics (2022) e em seu repositório no GitHub, que foi adaptado e ampliado para atender aos objetivos desta pesquisa.

Devido à extensão do código-fonte utilizado na construção do robô explorador, optou-se por disponibilizá-lo em um repositório público no GitHub. Essa medida visa manter a organização e a clareza do presente trabalho, ao mesmo tempo em que facilita o acesso ao código completo para fins de estudo, testes e reuso.

O repositório contém:

- Código-fonte completo e comentado;
- Interface HTML utilizada no controle remoto;
- Diagrama eletrônico em imagem;

- Instruções para replicar o projeto.

O código-fonte completo do projeto foi desenvolvido em grupo, está disponível no repositório público no GitHub: < https://github.com/JoaoCalegare/Rob-explorador-com-ESP32 >.

Alternativamente, o código-fonte também pode ser consultado no Anexo A página 17 deste trabalho.

4 CONCLUSÃO

Nosso trabalho tem como objetivo o desenvolvimento de um robô de exploração de baixo custo, voltado para o monitoramento de áreas remotas e de difícil acesso. Utilizamos componentes acessíveis, como o microcontrolador ESP32-CAM e ESP 32 com conectividade Wi-Fi e Bluetooth, e conseguimos montar um sistema funcional, capaz de se locomover e transmitir imagens em tempo real para um smartphone ou computador, por meio de uma interface web.

Para isso, usamos o ESP32-CAM, uma ponte H (L298N) e motores DC para a locomoção. Toda a lógica de programação é desenvolvida em C++ na plataforma Arduino IDE, onde configuramos a rede Wi-Fi para permitir o controle remoto via aplicativo e enviar para uma interface web, criada com HTML, permite enviar comandos diretamente ao robô, que aciona os motores e transmite as imagens captadas pela câmera.

Acreditamos que o grande diferencial do projeto está no uso de componentes simples e de baixo custo, o que torna a solução viável para diferentes aplicações — principalmente em ambientes educacionais e na exploração de locais perigosos, onde a presença humana oferece risco. O projeto é uma adaptação de uma base publicada pelo canal Hash Include Electronics, e mostra como conseguimos unir praticidade, funcionalidade e acessibilidade.

Estamos satisfeitos com os resultados que alcançamos. Sentimos que esse projeto representa uma conquista importante para o grupo, pois conseguimos colocar em prática conceitos teóricos, resolver problemas reais e entregar uma solução funcional. Para nós, é gratificante ver que com criatividade e dedicação conseguimos desenvolver um robô eficiente e acessível.

REFERÊNCIA

1. **HASH INCLUDE ELECTRONICS.** *WiFi Robot Tank | ESP32 + Smartphone Controlled*. YouTube, 24 abr. 2022. Disponível em: https://www.youtube.com/watch?v=XCp0qFipG2o&ab_channel=hashincludeelectronics. Acesso em: 11 jul. 2025.

2. **HASH INCLUDE ELECTRONICS.** *WiFi Tank Control – Código-fonte.* GitHub, 2022. Disponível em: https://github.com/un0038998/WiFiTank. Acesso em: 11 jul. 2025.

3. **HASH INCLUDE ELECTRONICS.** *Robotic Arm Car using ESP32 and PS3 Controller.* YouTube, 29 mar. 2022. Disponível em: https://youtu.be/Q3pCpVepCxk?si=vhtUXB88f1R0YL06. Acesso em: 25 set. 2025

4. **CALEGARE, João Paulo.** *Robô explorador com ESP32.* GitHub, 2025. Repositório do projeto em grupo. Disponível em: https://github.com/JoaoCalegare/Rob-explorador-com-ESP32. Acesso em: 16 out. 2025.

ANEXOS

ANEXO A – CÓDIGO-FONTE COMPLETO DO ROBÔ EXPLORADOR

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <iostream>
#include <sstream>
// -------- SENSOR ULTRASSÔNICO --------
const int trigPin = 32;
const int echoPin = 33;
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701
long duration;
float distanceCm;
float distanceInch;
// -------- MOTOR CONTROL --------
struct MOTOR_PINS {
  int pinEn;
  int pinIN1;
  int pinIN2;
};
std::vector<MOTOR_PINS> motorPins = {
  {22, 26, 27},  // RIGHT_MOTOR
  {23, 14, 12}   // LEFT_MOTOR
};
#define UP 1
#define DOWN 2
#define LEFT 3
```

```
#define RIGHT 4
#define STOP 0
#define RIGHT_MOTOR 0
#define LEFT_MOTOR 1
#define FORWARD 1
#define BACKWARD -1
const int PWMFreq = 1000;
const int PWMResolution = 8;
const int PWMSpeedChannel = 4;
void rotateMotor(int motorNumber, int motorDirection) {
 digitalWrite(motorPins[motorNumber].pinIN1, motorDirection == FORWARD ? HIGH : LOW);
 digitalWrite(motorPins[motorNumber].pinIN2, motorDirection == BACKWARD ? HIGH : LOW);
 if (motorDirection == STOP) {
  digitalWrite(motorPins[motorNumber].pinIN1, LOW);
  digitalWrite(motorPins[motorNumber].pinIN2, LOW);  }}
void moveCar(int inputValue) {
 switch (inputValue) {
  case UP: rotateMotor(RIGHT_MOTOR, FORWARD); rotateMotor(LEFT_MOTOR, FORWARD); break;
  case DOWN: rotateMotor(RIGHT_MOTOR, BACKWARD); rotateMotor(LEFT_MOTOR, BACKWARD); break;
  case LEFT: rotateMotor(RIGHT_MOTOR, FORWARD); rotateMotor(LEFT_MOTOR, BACKWARD); break;
  case RIGHT: rotateMotor(RIGHT_MOTOR, BACKWARD); rotateMotor(LEFT_MOTOR, FORWARD); break;
  case STOP:
  default:
   rotateMotor(RIGHT_MOTOR, STOP); rotateMotor(LEFT_MOTOR, STOP); break;
 }}
```

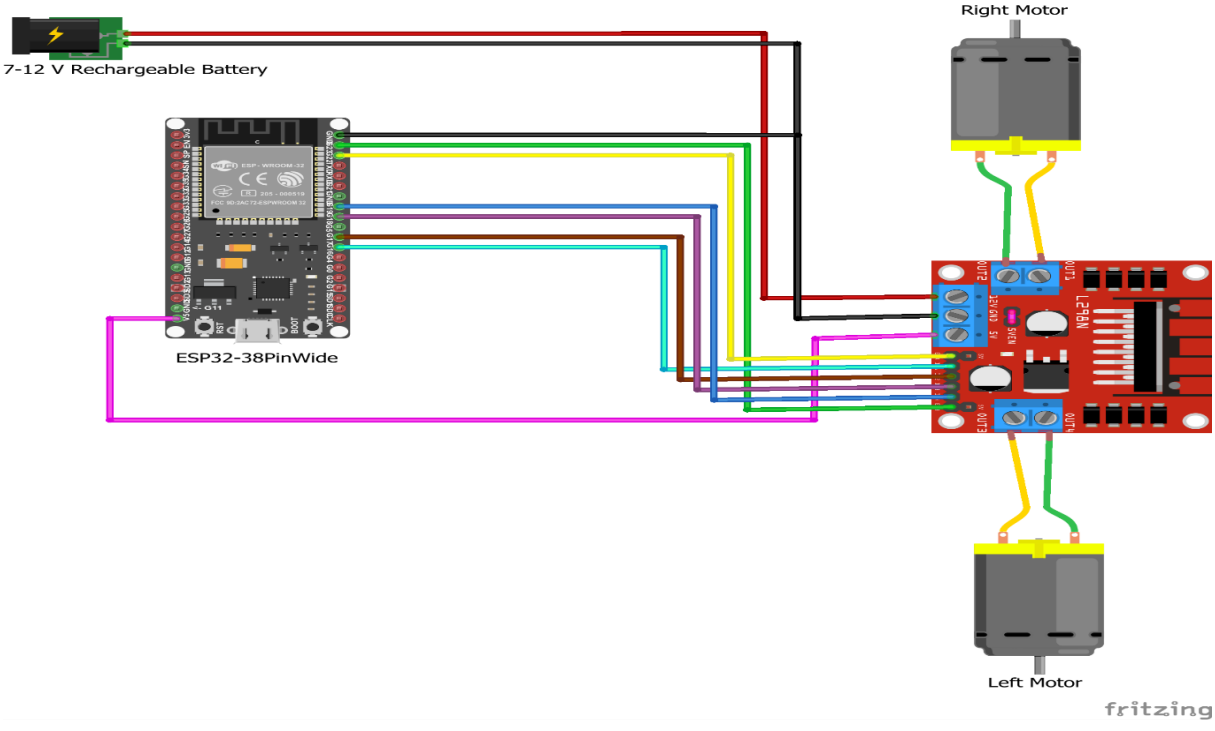
```
// -------- PWM LED (SEPARADO DO MOTOR) --------
const int ledPWMChannel = 5;
const int ledPin = 25;  // Pino separado para LED PWM
const int ledFreq = 5000;
const int ledResolution = 8;
// -------- WiFi + WebSocket --------
const char* ssid = "MyWiFiCar";
const char* password = "12345678";
AsyncWebServer server(80);
AsyncWebSocket wsCarInput("/CarInput");
const char* htmlHomePage PROGMEM = R"HTMLHOMEPAGE(
// ... [HTML completo do controle remoto aqui, idêntico ao seu código original] ...
)HTMLHOMEPAGE";
void handleRoot(AsyncWebServerRequest *request) {
  request->send_P(200, "text/html", htmlHomePage);
}
void handleNotFound(AsyncWebServerRequest *request) {
  request->send(404, "text/plain", "File Not Found");}
void onCarInputWebSocketEvent(AsyncWebSocket *server, AsyncWebSocketClient *client,
AwsEventType type,
                      void *arg, uint8_t *data, size_t len) {
  if (type == WS_EVT_DATA) {
    AwsFrameInfo info = (AwsFrameInfo)arg;
    if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT){
      std::string myData((char*)data, len);
      std::istringstream ss(myData);
      std::string key, value;
      std::getline(ss, key, ',');
```

```
      std::getline(ss, value, ',');
      int valueInt = atoi(value.c_str());
      if (key == "MoveCar") moveCar(valueInt);
      else if (key == "Speed") {
        ledcWrite(PWMSpeedChannel, valueInt);  // Motor speed
        ledcWrite(ledPWMChannel, valueInt);    // LED brightness   }    }
  } else if (type == WS_EVT_DISCONNECT) {
    moveCar(STOP);  }}
void setUpPinModes() {
  ledcSetup(PWMSpeedChannel, PWMFreq, PWMResolution);
  ledcSetup(ledPWMChannel, ledFreq, ledResolution);
  ledcAttachPin(motorPins[0].pinEn, PWMSpeedChannel);
  ledcAttachPin(motorPins[1].pinEn, PWMSpeedChannel);
  ledcAttachPin(ledPin, ledPWMChannel);
  for (auto m : motorPins) {
    pinMode(m.pinIN1, OUTPUT);
    pinMode(m.pinIN2, OUTPUT);  }
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  moveCar(STOP);}
void setup() {
  Serial.begin(115200);
  setUpPinModes();
  WiFi.softAP(ssid, password);
  Serial.print("AP IP address: ");
  Serial.println(WiFi.softAPIP());
  server.on("/", HTTP_GET, handleRoot);
  server.onNotFound(handleNotFound);
  wsCarInput.onEvent(onCarInputWebSocketEvent);
```

```
  server.addHandler(&wsCarInput);
  server.begin();}
void loop() {
  wsCarInput.cleanupClients();
  // SENSOR ULTRASSÔNICO
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distanceCm = duration * SOUND_SPEED / 2;
  distanceInch = distanceCm * CM_TO_INCH;
  Serial.print("Distância (cm): ");
  Serial.println(distanceCm);
  delay(1000);  // Pode ajustar para leitura mais frequente
}
```

Fonte: Adaptado, criado pelo canal HASH INCLUDE ELECTRONICS.

ANEXO B – DIAGRAMA ELETRÔNICO

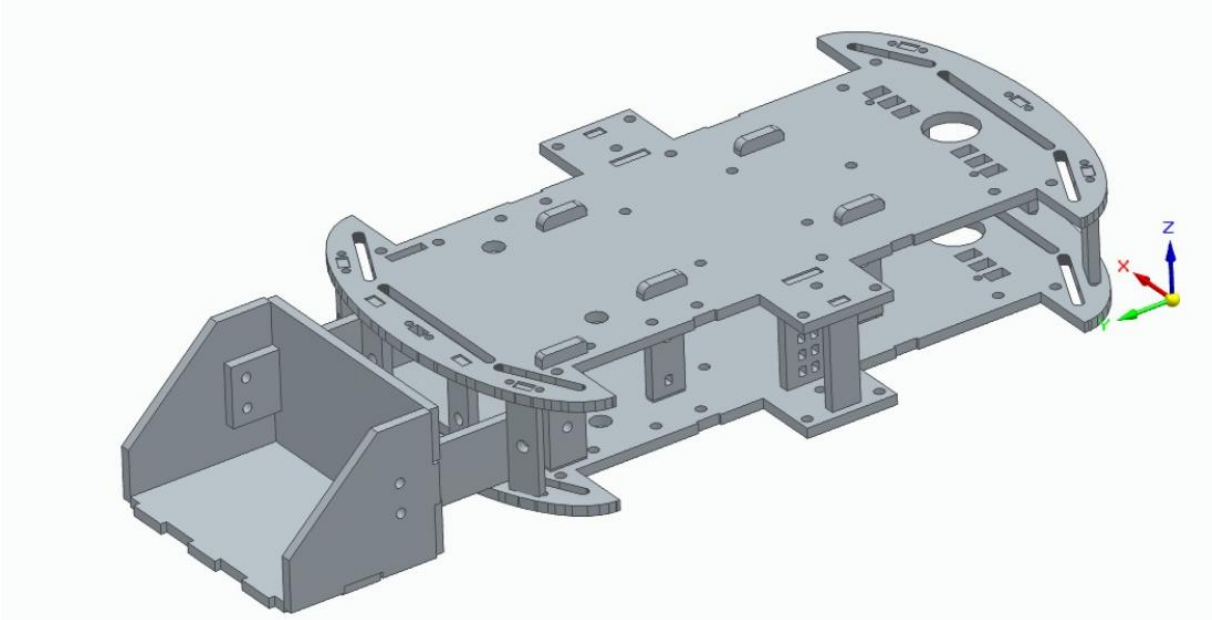


Fonte: Adaptado, criado pelo canal HASH INCLUDE ELECTRONICS.

ANEXO C – INTERFASE WEB (HTML)

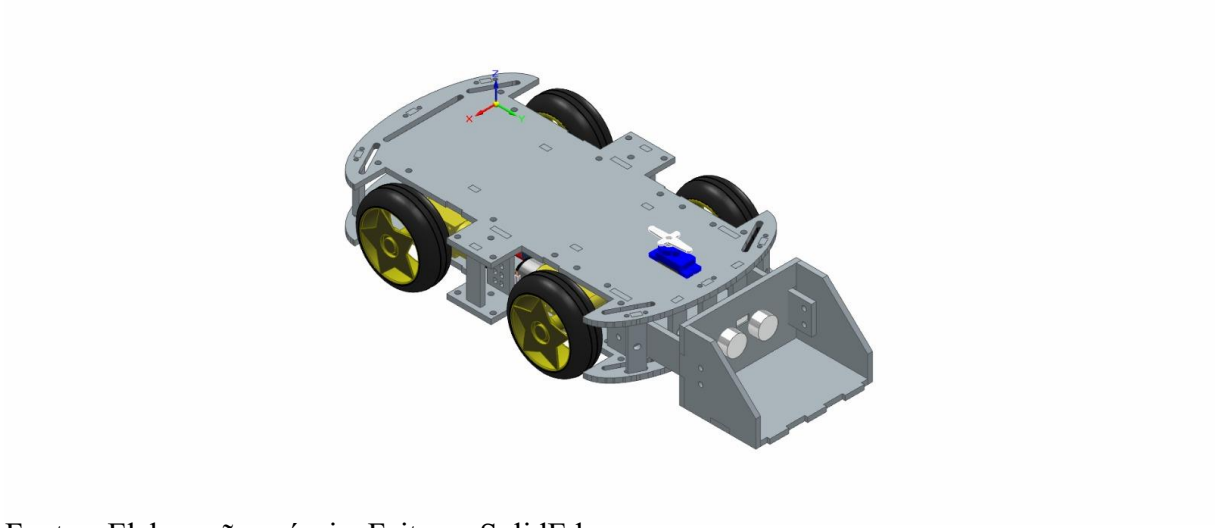
Fonte: Adaptado, criado pelo canal HASH INCLUDE ELECTRONICS.

ANEXO D – PROTÓTIPO CARCAÇA



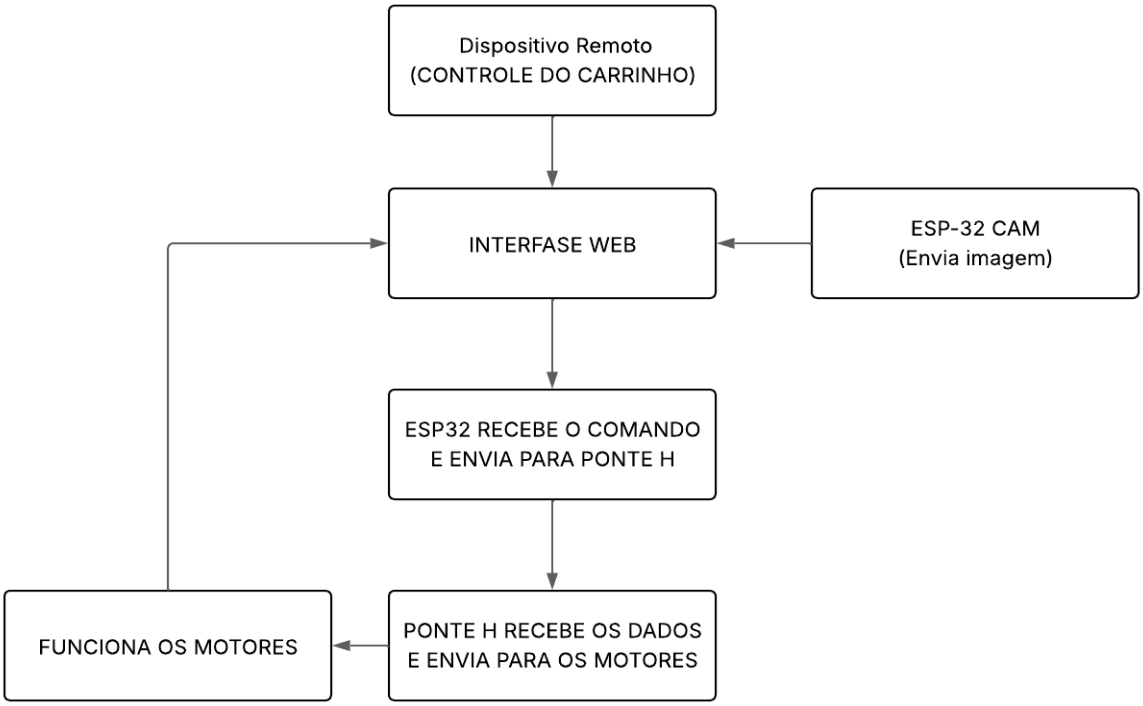
Fontes: Elaboração própria, Feita no SolidEdge.

ANEXO E – PROTOTIPO



Fontes: Elaboração própria, Feita no SolidEdge.

ANEXO F - DIAGRAMA DE BLOCOS FUNCIONAL



Fontes: Elaboração própria feito no Lucid.app.