UNIVERSIDADE FEDERAL DO PARÁ
CAMPUS UNIVERSITÁRIO DE TUCURUÍ
FACULDADE DE ENGENHARIA DE COMPUTAÇÃO

WILLIAMS DAVID ESTUMANO DUARTE

SISTEMA DE TELEMETRIA VEICULAR BASEADO EM ESP32 E PLATAFORMA WEB FULL STACK INTEGRADA

TUCURUÍ/PA
2026

WILLIAMS DAVID ESTUMANO DUARTE

SISTEMA DE TELEMETRIA VEICULAR BASEADO EM ESP32 E PLATAFORMA WEB FULL STACK INTEGRADA

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a obtenção de grau de Bacharel em Faculdade de Engenharia de Computação, pela Universidade Federal do Pará.

Orientador: Prof. Dr. Daniel da Conceição Pinheiro
Universidade Federal do Pará

TUCURUÍ/PA
2026

WILLIAMS DAVID ESTUMANO DUARTE

SISTEMA DE TELEMETRIA VEICULAR BASEADO EM ESP32 E PLATAFORMA WEB FULL STACK INTEGRADA

Trabalho de Conclusão de Curso, apresentado como requisito parcial para a obtenção de grau de Bacharel em Faculdade de Engenharia de Computação, pela Universidade Federal do Pará.

DATA DE APROVAÇÃO: 06/02/2026

CONCEITO: Excelente

Prof. Dr. Daniel da Conceição Pinheiro
Orientador: – FECOMP/UFPA

Prof. Dr. Marco José de Sousa Rocha
Membro – FECOMP/UFPA

Prof. Dr. Otávio Noura Teixeira
Membro – FECOMP/UFPA

TUCURUÍ/PA
2026

Aos meus pais, que me incentivaram de diversas maneiras durante toda a minha vida para que eu sempre avançasse em meus estudos.

AGRADECIMENTOS

Agradeço a Deus por sempre estar comigo em todos os momentos e às pessoas ao meu redor, como professores, familiares e amigos, especialmente aos meus pais, por sempre me incentivarem a continuar meus estudos, concluir o bacharelado e avançar ainda mais nos níveis acadêmicos. Sem eles, nada disso seria possível.

"A sabedoria é o conhecimento das coisas divinas e humanas, ordenadas ao bem."
(São Tomás de Aquino, *Summa Theologica*, I–II, q.57)

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema completo de telemetria veicular, composto por um dispositivo físico embarcado e uma plataforma web de gerenciamento. O sistema tem como objetivo monitorar o comportamento de motoristas e veículos em tempo real, registrando trajetos, velocidades e ocorrências de infrações, além de emitir alertas sonoros em caso de excesso de velocidade. O dispositivo foi desenvolvido com o microcontrolador ESP32 em conjunto com o ESP8266 e diversos módulos de comunicação e sensoriamento, enquanto a plataforma foi construída utilizando React, Node.js e banco de dados PostgreSQL. O sistema proposto visa oferecer uma solução eficiente para empresas que buscam aumentar a segurança de suas frotas e otimizar o controle de trajetos e condutas de motoristas.

Palavras-chave: ESP32. Telemetria. IoT. Geofencing. Node.js. React. Segurança Veicular.

ABSTRACT

This work presents the development of a complete vehicle telemetry system composed of an embedded device and a comprehensive web management platform. The system aims to monitor driver and vehicle behavior in real time, recording routes, speeds, and instances of speeding, while providing audible alerts when speed limits are exceeded. The embedded device was developed using the ESP32 microcontroller in conjunction with the ESP8266 and several communication and sensing modules, whereas the web platform was built using React, Node.js, and a PostgreSQL database. The proposed system provides an efficient solution for companies seeking to enhance fleet safety and optimize the monitoring of routes and driver conduct.

Keywords: ESP32. Telemetry. IoT. Geofencing. Node.js. React. Vehicle Safety.

LISTA DE FIGURAS

Figura 1.1 – Arquitetura geral do sistema embarcado e suas camadas de integração. 16
Figura 3.1 – Excessos de velocidade após a implantação do sistema de telemetria. . 23
Figura 4.1 – Esquema geral de ligação dos controladores e módulos no simulador. . . 27
Figura 4.2 – Fluxo geral de funcionamento do dispositivo embarcado. 29
Figura 4.3 – Lógica de atualização de cercas e motoristas. 31
Figura 4.4 – Conexão física entre os microcontroladores. 33
Figura 4.5 – Representação visual do método Ray casting 36
Figura 4.6 – Dispositivo ELM327 e sua interface OBD-II em veículos. 38
Figura 4.7 – Diagrama entidade-relacionamento do banco de dados. (Elaborado em *dbdiagram.io*, 2025.) 43
Figura 4.8 – Tela inicial da plataforma online de monitoramento desenvolvida em *React*. 45
Figura 4.9 – Outras telas da plataforma online de monitoramento desenvolvida em *React*. 46
Figura 4.10–Etapas do processo de fabricação da placa de circuito impresso. 48
Figura 4.11–Caixa do aparelho desenvolvida para o encapsulamento do dispositivo. 49
Figura 5.1 – Um dos percursos registrados durante os testes, exibido na plataforma. 52
Figura 5.2 – Exemplos de percursos de teste visualizados na plataforma online desenvolvida. 53

LISTA DE TABELAS

Tabela 5.1 – Custos aproximados dos componentes da versão atual do protótipo . . 54
Tabela 5.2 – Custos aproximados dos componentes da versão planejada do protótipo 55
Tabela 5.3 – Estimativa de custos mensais de hospedagem da infraestrutura de software 55

LISTA DE SIGLAS

API	*Application Programming Interface*
ASCII	*American Standard Code for Information Interchange*
BLE	*Bluetooth Low Energy*
CAN	*Controller Area Network*
DOM	*Document Object Model*
ELM327	*Módulo interpretador de protocolos OBD-II desenvolvido pela ELM Electronics*
ECU	*Electronic Control Unit*
ESP	*Espressif Systems* (fabricante dos microcontroladores ESP32 e ESP8266)
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
I2C	*Inter-Integrated Circuit*
ID	*Identificador*
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
LCD	*Liquid Crystal Display*
NPM	*Node Package Manager*
OBD-II	*On-Board Diagnostics II (Sistema de Diagnóstico Embarcado de Segunda Geração)*
PCB	*Printed Circuit Board (Placa de Circuito Impresso)*
PSRAM	*Pseudo Static RAM*
RAM	*Random Access Memory*
REST	*Representational State Transfer*

RFID	*Radio-Frequency Identification*
RPM	*Rotações por Minuto*
SD	*Secure Digital*
SPI	*Serial Peripheral Interface*
SQL	*Structured Query Language*
UART	*Universal Asynchronous Receiver-Transmitter*

SUMÁRIO

1 INTRODUÇÃO 14

2 OBJETIVOS 17
2.1 Objetivo geral 17
2.2 Objetivos específicos 17

3 FUNDAMENTAÇÃO TEÓRICA 18
3.1 Sistemas embarcados e IoT 18
3.2 Microcontroladores ESP32 e ESP8266 18
3.3 Comunicação e armazenamento de dados 19
3.4 Módulos e sensores utilizados 20
3.5 API e arquitetura de software 21
3.6 Plataforma de Monitoramento 21
3.7 React e Leaflet 22
3.8 Geocercas e controle de velocidade 22

4 METODOLOGIA 24
4.1 Requisitos do sistema 24
4.2 Arquitetura de Hardware 26
4.2.1 Integração Física dos Controladores e Módulos 26
4.2.2 Fluxo Geral de Processamento 28
4.2.3 Lógica de Armazenamento e Atualização de Dados Recebidos da API . 30
4.2.4 Lógica de Comunicação entre ESP32 e ESP8266 32
4.2.5 Lógica de Autenticação de Motoristas 34
4.2.6 Registro de Viagens e Aquisição de Localização 35
4.2.7 Aquisição de Dados do Veículo 37
4.2.8 Situações de Alerta 39
4.3 Arquitetura de Software 40
4.3.1 Modelagem e Estrutura do Banco de Dados 40
4.3.2 API do sistema 43
4.3.3 Plataforma de Monitoramento e Interface do Usuário 44
4.4 Encapsulamento 46
4.4.1 Placa de Circuito Impresso 47
4.4.2 Caixa do Aparelho 49

5 RESULTADOS 50
5.1 Funcionamento do Dispositivo Embarcado 50

5.2 Integração com a Plataforma Online 51
5.3 Cenários de uso testados 52
5.4 Custos de Produção 54
5.4.1 Versão Atual do Protótipo 54
5.4.2 Segunda Versão Planejada 54
5.4.3 Hospedagem 55

6 POSSÍVEIS MELHORIAS E TRABALHOS FUTUROS 56

7 CONCLUSÃO 59

REFERÊNCIAS 60

1 INTRODUÇÃO

Recursos tecnológicos aplicados ao setor de transporte têm auxiliado trabalhadores e contribuído para o aumento da segurança nas atividades relacionadas à operação de veículos. A integração entre sensores, microcontroladores e sistemas em nuvem permite desenvolver soluções capazes de coletar, processar e transmitir informações em tempo real sobre o comportamento de veículos e motoristas.

Segundo GOMES (2022), a utilização da telemetria e de sistemas de monitoramento em tempo real amplia a visibilidade das operações, favorecendo o controle e o acompanhamento contínuo por parte dos responsáveis. Esse tipo de sistema possibilita a análise de indicadores operacionais em tempo real, permitindo a identificação antecipada de problemas e a tomada de decisões mais eficientes, o que contribui para o aumento da segurança, redução de custos e melhoria da qualidade e da lucratividade dos processos.

Esse cenário reforça a importância de soluções integradas de telemetria que não apenas coletem dados, mas que também possibilitem análise inteligente e ação preventiva em tempo real.

Neste contexto, o presente trabalho propõe o desenvolvimento de um sistema de telemetria veicular baseado em um microcontrolador com conectividade sem fio, capaz de coletar dados de velocidade e localização do veículo, bem como realizar a identificação de motoristas por meio de tecnologia de identificação por radiofrequência (RFID). As informações coletadas são processadas localmente e enviadas a uma aplicação servidora hospedada em nuvem, permitindo o acompanhamento remoto e o armazenamento de relatórios.

O sistema proposto é baseado no microcontrolador ESP32, da Espressif Systems, amplamente empregado em soluções de Internet das Coisas por integrar conectividade sem fio e recursos de processamento adequados a sistemas embarcados (Espressif Systems, 2023).

A solução utiliza recursos típicos de aplicações de Internet das Coisas (IoT), integrando módulos de posicionamento global (GPS) e comunicação sem fio, além do uso de cercas virtuais para a delimitação de áreas com restrições de velocidade.

O projeto tem como objetivo principal demonstrar a viabilidade de uma solução de baixo custo e alta integração para o monitoramento inteligente de veículos, utilizando componentes amplamente disponíveis e de fácil implementação. Diferentemente das soluções comerciais existentes, que em geral não disponibilizam informações técnicas detalhadas sobre sua arquitetura, funcionamento interno e custos de implementação, este trabalho propõe uma abordagem transparente e reprodutível, voltada à compreensão e à construção do sistema.

A relevância deste trabalho está na apresentação de um estudo prático que

exemplifica como tecnologias de Internet das Coisas e sistemas embarcados podem ser empregadas na construção de um sistema de telemetria veicular, servindo como referência técnica e de custos para pesquisas futuras, projetos acadêmicos e aplicações experimentais.

O sistema desenvolvido neste trabalho é composto por três camadas principais:

(i) Sistema embarcado, implementado a partir de um dispositivo baseado no microcontrolador ESP32, em cooperação com um microcontrolador ESP8266, responsáveis pelo processamento e comunicação do sistema (Espressif Systems, 2023; Espressif Systems, 2020). O sistema integra módulos auxiliares, incluindo um leitor por radiofrequência para identificação de motoristas (NXP Semiconductors, 2012), um módulo de posicionamento global (GPS) para obtenção de dados de localização (u-blox AG, 2010) e um módulo ELM327, desenvolvido pela ELM Electronics, utilizado para a aquisição da velocidade do veículo (ELM Electronics, 2010). Complementarmente, o sistema utiliza um módulo adaptador para cartão Micro SD, destinado ao armazenamento temporário de dados (MICRO..., 2019), um módulo de display de cristal líquido (LCD) com interface I2C para exibição de informações ao usuário (EONE Electronics, 2018; NXP Semiconductors, 2015) e um transdutor piezoelétrico para a emissão de alertas sonoros locais (Murata Manufacturing Co., Ltd., 2016).

(ii) Aplicação servidora, desenvolvida em Node.js, ambiente de execução JavaScript no lado do servidor, responsável por receber, validar e persistir os dados enviados pelos dispositivos embarcados (Node.js Foundation, 2025);

(iii) Plataforma de gerenciamento, implementada como uma aplicação web utilizando a biblioteca React, voltada à construção de interfaces baseadas em componentes, responsável pela visualização de rotas em mapas, filtragem de registros, gerenciamento de cercas virtuais e seus respectivos limites de velocidade, além da exibição de alertas e do rastreamento em tempo real dos veículos (Meta Platforms, Inc., 2025).

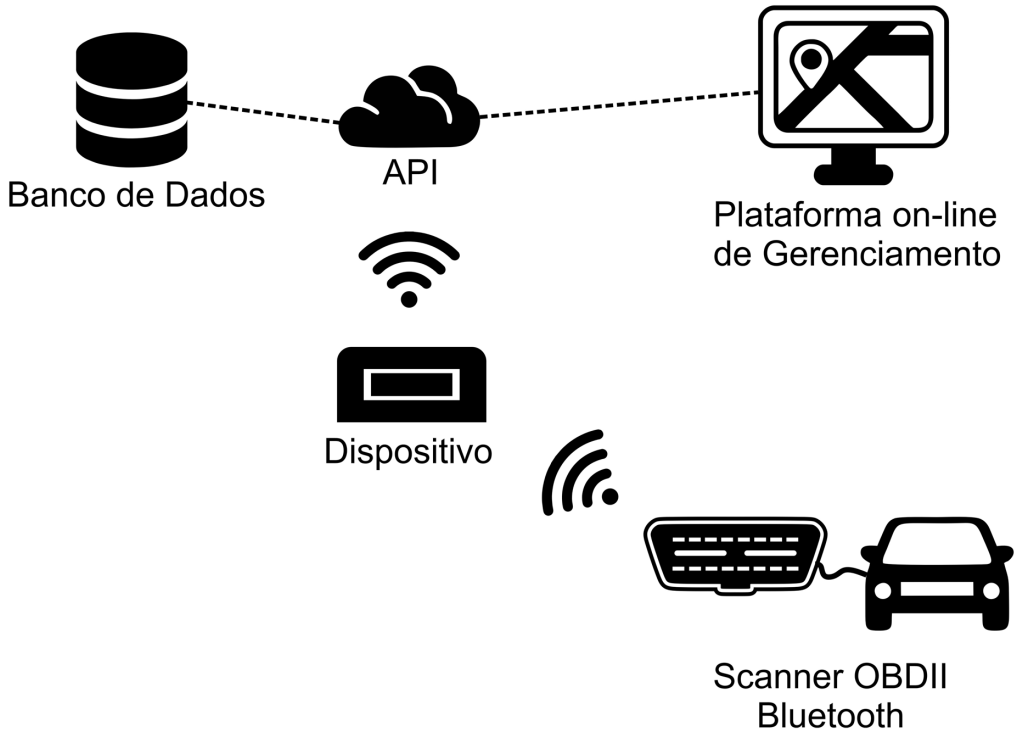


Figura 1.1 – Arquitetura geral do sistema embarcado e suas camadas de integração.

Fonte: Elaboração própria.

O dispositivo embarcado registra cada viagem, aplica regras locais de segurança (comparação de velocidade com limites definidos nas cercas virtuais, emissão de alertas sonoros e registro de ocorrências) e sincroniza os dados com a plataforma assim que a conectividade é restabelecida, garantindo redundância e reduzindo perda de dados. Essa arquitetura visa fornecer uma solução integrada, de baixo custo e resiliente, para monitoramento e aumento da segurança em frotas comerciais.

2 OBJETIVOS

2.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um sistema completo de telemetria veicular baseado em tecnologias embarcadas e conectividade IoT, capaz de coletar, armazenar e transmitir dados de forma confiável, visando o monitoramento e o aumento da segurança de motoristas e frotas. Além disso, busca-se que o sistema apresente baixo custo, seja de fácil instalação, manutenção, fabricação e utilização, tornando-o uma solução viável e acessível para diferentes contextos de aplicação.

2.2 Objetivos específicos

Os objetivos específicos incluem:

- Projetar e implementar um sistema embarcado utilizando ESP32 e ESP8266 para aquisição, processamento e transmissão de dados veiculares;
- Integrar sensores e módulos periféricos, como GPS, RFID, cartão SD, display I2C e ELM327 via Bluetooth, assegurando compatibilidade e desempenho eficiente;
- Desenvolver uma API em Node.js para recepção, armazenamento e gerenciamento dos dados de telemetria, garantindo confiabilidade e segurança na comunicação;
- Criar uma plataforma web interativa em React para visualização das rotas, alertas, cercas geográficas e informações dos motoristas em tempo real;
- Implementar lógica de alerta e limitação de velocidade com base em condições de chuva, localização e identificação do motorista, visando promover condução segura e responsável;
- Garantir redundância no armazenamento dos dados em casos de falha de conexão com a internet, evitando perdas de informações de rota e eventos;
- Avaliar a viabilidade do sistema quanto ao custo, facilidade de instalação, manutenção, fabricação e uso, de modo a favorecer sua aplicação prática em diferentes cenários.

3 Fundamentação Teórica

3.1 Sistemas embarcados e IoT

Sistemas embarcados são dispositivos de processamento dedicados a tarefas específicas, integrando sensores, atuadores e mecanismos de comunicação para operar de forma autônoma ou semi-autônoma. Eles são projetados para eficiência, baixo consumo de energia, confiabilidade e facilidade de manutenção, permitindo aplicações práticas em diversos setores da indústria e transporte (CUGNASCA, 2020).

A IoT conecta esses sistemas a redes, possibilitando a coleta, transmissão e processamento de dados em tempo real. Com a redução do tamanho e do custo de microcontroladores e sensores, tornou-se viável incorporar capacidades computacionais a objetos cotidianos, transformando-os em dispositivos inteligentes capazes de interagir com usuários e sistemas remotos.

- **Dispositivos inteligentes:** hardware embarcado equipado com sensores e atuadores que coletam informações do ambiente ou do usuário e transmitem dados para uma aplicação central.
- **Plataforma de processamento:** software ou serviços na nuvem que recebem, armazenam e processam os dados, podendo utilizar técnicas de análise avançada, aprendizado de máquina ou inteligência artificial para fornecer insights e respostas automáticas.
- **Interface de usuário:** aplicações móveis ou web que permitem monitoramento, configuração e controle dos dispositivos, oferecendo visualizações de dados e relatórios em tempo real.

A Internet das Coisas tem um amplo impacto na vida humana e no trabalho. Ela permite que as máquinas façam mais trabalho pesado, assumam tarefas tediosas e tornem a vida mais saudável, produtiva e confortável (Amazon Web Services, 2025).

No contexto deste trabalho, a integração de sistemas embarcados com IoT possibilita a criação de uma plataforma de telemetria veicular eficiente, de baixo custo e de fácil instalação, capaz de monitorar motoristas e veículos, alertar sobre violações de velocidade e armazenar dados de forma confiável mesmo em situações de desconexão temporária da rede.

3.2 Microcontroladores ESP32 e ESP8266

Os microcontroladores são dispositivos eletrônicos que integram, em um único chip, unidade de processamento, memória e interfaces de comunicação, permitindo o controle de sensores, atuadores e outros módulos de forma autônoma. Devido a essas características, são amplamente empregados em sistemas embarcados e aplicações de Internet das Coisas

(IoT). Entre os microcontroladores mais utilizados nesse contexto destacam-se o ESP32 e o ESP8266, ambos desenvolvidos pela Espressif Systems (Espressif Systems, 2023; Espressif Systems, 2020).

O **ESP32** é um microcontrolador de 32 bits que oferece maior capacidade de processamento e recursos avançados de comunicação, sendo adequado para aplicações que exigem gerenciamento simultâneo de múltiplos módulos e processamento mais complexo. O dispositivo integra conectividade sem fio e diversas interfaces de comunicação, o que facilita sua integração com sensores e sistemas externos, além de permitir a execução de aplicações em tempo real (Espressif Systems, 2023).

O **ESP8266**, por sua vez, apresenta uma arquitetura mais simples e recursos mais limitados quando comparado ao ESP32. Ainda assim, destaca-se por sua eficiência na comunicação sem fio, sendo amplamente utilizado em aplicações que demandam conectividade com redes Wi-Fi. Essas características tornam o ESP8266 adequado como módulo auxiliar em sistemas embarcados voltados à transmissão de dados para aplicações servidoras ou serviços em nuvem (Espressif Systems, 2020).

O uso combinado do ESP32 e do ESP8266 permite a distribuição das responsabilidades do sistema embarcado, de modo que o ESP32 atue como controlador central, responsável pelo processamento dos dados e gerenciamento dos dispositivos, enquanto o ESP8266 pode ser dedicado às tarefas de comunicação. Essa abordagem contribui para maior eficiência, organização do sistema e confiabilidade na transmissão das informações.

Embora compartilhem características comuns, como suporte a interfaces de entrada e saída programáveis e protocolos de comunicação amplamente utilizados, os microcontroladores diferem em capacidade de processamento, quantidade de memória e recursos adicionais. Essas diferenças tornam cada dispositivo mais adequado a funções específicas dentro de um sistema embarcado integrado, possibilitando uma arquitetura mais flexível e eficiente.

3.3 Comunicação e armazenamento de dados

Em sistemas embarcados e IoT, a comunicação eficiente entre os dispositivos e os servidores é fundamental para garantir o correto processamento e monitoramento das informações coletadas. Essa comunicação geralmente é realizada utilizando protocolos baseados em redes IP, como Hypertext Transfer Protocol (HTTP) ou Hypertext Transfer Protocol Secure (HTTPS), que permitem a troca de dados de forma padronizada e segura (Amazon Web Services, 2023).

O formato de dados mais utilizado nesses casos é o JSON (JavaScript Object Notation), por ser leve, de fácil leitura e compatível com diversas linguagens de programação e plataformas. Esse formato permite que os dados sejam estruturados de maneira clara, facilitando sua transmissão e posterior interpretação pelos sistemas receptores.

A persistência local dos dados, por meio de armazenamento em memória não volátil, como cartões SD ou memória flash, é uma prática importante em sistemas embarcados. Ela garante que as informações coletadas não sejam perdidas em caso de falhas temporárias de comunicação ou indisponibilidade da rede. Dessa forma, os dados podem ser transmitidos posteriormente, mantendo a integridade e a confiabilidade do sistema (Cloudian, 2023).

Além disso, a redundância nos processos de comunicação e armazenamento aumenta a robustez do sistema, reduzindo o risco de perda de informações críticas e permitindo que operações de monitoramento e controle sejam realizadas de forma contínua, mesmo diante de falhas parciais na infraestrutura de network.

3.4 Módulos e sensores utilizados

Para compor o sistema de telemetria proposto, foram integrados diversos módulos eletrônicos responsáveis pela coleta, armazenamento e exibição das informações veiculares. Esses módulos são baseados em circuitos integrados amplamente utilizados em sistemas embarcados. A seguir, são descritos os principais módulos empregados no projeto:

- **Módulo RFID RC522**: módulo de identificação por radiofrequência baseado no circuito integrado MFRC522, desenvolvido pela NXP Semiconductors. Esse módulo é utilizado para a leitura de tags RFID, sendo amplamente empregado em aplicações de controle de acesso, identificação de objetos e autenticação de usuários (NXP Semiconductors, 2012).
- **Módulo adaptador para cartão Micro SD**: módulo utilizado para a interface entre cartões de memória Micro SD e microcontroladores, possibilitando o armazenamento local e persistente de dados em memória não volátil. Esse tipo de módulo é amplamente empregado em sistemas embarcados para registro temporário de informações, especialmente em cenários de indisponibilidade de conexão com a rede. O módulo permite a leitura e escrita de dados de forma eficiente, contribuindo para a confiabilidade e integridade do sistema (MICRO. . . , 2019).
- **Módulo GPS NEO-6M V2**: módulo de posicionamento global baseado no receptor GPS NEO-6, desenvolvido pela u-blox AG. Esse módulo permite a obtenção de dados como latitude, longitude, altitude e velocidade (u-blox AG, 2010).
- **Display LCD com interface I2C**: módulo de exibição baseado em um display de cristal líquido do tipo 16×2, utilizado para a apresentação de informações ao usuário. O módulo é composto por um controlador dedicado integrado ao display, amplamente empregado em sistemas embarcados, e utiliza um expansor de portas com interface I2C, permitindo a comunicação com o microcontrolador por meio de um número reduzido de pinos. Essa abordagem é comum em aplicações de monitoramento e controle, nas quais a visualização de dados em tempo real é necessária (EONE Electronics, 2018; NXP Semiconductors, 2015).

3.5 API e arquitetura de software

As APIs desempenham um papel essencial na integração entre sistemas distribuídos, permitindo a comunicação entre dispositivos embarcados, servidores e aplicações de monitoramento. Entre os estilos arquiteturais mais utilizados está o *Representational State Transfer* (REST), que define um conjunto de princípios para o desenvolvimento de serviços web baseados em recursos acessíveis por meio dos métodos do protocolo HTTP, como *GET*, *POST*, *PUT* e *DELETE*. Esse modelo promove a simplicidade, escalabilidade e interoperabilidade entre diferentes plataformas e linguagens (IBM, 2024).

O ambiente de execução Node.js tem se destacado no desenvolvimento de APIs modernas devido à sua arquitetura assíncrona e orientada a eventos, baseada no motor V8 do Google Chrome. Essa característica permite lidar com múltiplas requisições simultâneas de forma eficiente, tornando-o especialmente adequado para aplicações em tempo real, como sistemas de telemetria e monitoramento remoto (Node.js Foundation, 2025). Além disso, o vasto ecossistema de bibliotecas disponíveis no Node Package Manager (NPM) facilita a criação de soluções personalizadas e modulares (npm, Inc., 2025).

Para o armazenamento e gerenciamento de dados, o banco de dados PostgreSQL é amplamente utilizado por oferecer robustez, conformidade com o padrão Structured Query Language (SQL) e suporte a estruturas complexas de dados. Sua arquitetura relacional permite o uso de chaves estrangeiras, índices e transações, assegurando integridade e consistência. O PostgreSQL também se destaca por sua extensibilidade e pela capacidade de lidar com grandes volumes de dados, características essenciais em sistemas que dependem de registros contínuos e históricos, como aplicações de telemetria veicular (PostgreSQL Global Development Group, 2024).

Essa arquitetura de software complementa a estrutura física do sistema embarcado, garantindo uma comunicação confiável e contínua entre o veículo e a plataforma de monitoramento.

3.6 Plataforma de Monitoramento

Os sistemas de telemetria veicular modernos integram múltiplos componentes de hardware e software para coleta, processamento e visualização de dados em tempo real. Esses sistemas permitem acompanhar parâmetros de operação do veículo, como velocidade, posição geográfica, comportamento do motorista e condições ambientais, fornecendo informações úteis para gestão e segurança.

A infraestrutura tecnológica, composta por microcontroladores, sensores, redes de comunicação e plataformas em nuvem, possibilita o envio contínuo de dados a servidores remotos, onde são processados e disponibilizados por meio de interfaces gráficas. Essa integração entre camadas físicas e digitais caracteriza o ecossistema de Internet das Coisas (IoT) aplicado ao contexto automotivo. Conforme discutido por SILVA e SANTOS (2024),

esse tipo de infraestrutura permite não apenas o monitoramento do comportamento dos veículos, mas também a identificação de anomalias e padrões associados a riscos operacionais, contribuindo diretamente para a prevenção de acidentes e para a otimização da segurança.

Plataformas de monitoramento bem projetadas devem oferecer funcionalidades como rastreamento em tempo real, análise de rotas, emissão de alertas automáticos e armazenamento histórico dos dados coletados. Essas características são essenciais para promover a segurança operacional, reduzir custos de manutenção e aprimorar a gestão de frotas.

3.7 React e Leaflet

No contexto das plataformas web modernas, frameworks como o *React* têm se destacado por oferecerem alto desempenho, modularidade e escalabilidade no desenvolvimento de interfaces interativas (Meta Platforms, Inc., 2025). Criado pelo Facebook, o React utiliza um modelo baseado em componentes e um Document Object Model (DOM) virtual, o que reduz o custo computacional das atualizações de interface e garante maior fluidez na visualização de dados em tempo real. Essa característica o torna especialmente adequado para aplicações de monitoramento veicular, em que a responsividade e a atualização contínua das informações são fatores críticos.

Para a representação geográfica das informações, bibliotecas como o *Leaflet* são amplamente utilizadas devido à sua leveza, precisão e compatibilidade com diferentes provedores de mapas (Leaflet Contributors, 2026). O Leaflet permite a renderização eficiente de camadas vetoriais, marcadores e polígonos, possibilitando a visualização de rotas, áreas de cobertura e cercas geográficas de forma intuitiva.

Assim, a combinação de tecnologias como React e Leaflet contribui para o desenvolvimento de plataformas de monitoramento robustas e seguras, capazes de oferecer visualizações dinâmicas, intuitivas e confiáveis dos dados provenientes dos dispositivos embarcados.

3.8 Geocercas e controle de velocidade

As geocercas atuam como delimitadores virtuais que permitem o controle dinâmico de velocidade e o monitoramento de eventos em áreas específicas. Quando integradas a sistemas de telemetria, tornam-se ferramentas eficazes para a prevenção de acidentes e a promoção de uma direção mais consciente.

O uso de geocercas é amplamente adotado em soluções comerciais de telemetria veicular. Empresas especializadas no gerenciamento de frotas utilizam cercas virtuais para delimitar áreas operacionais, controlar velocidades máximas, registrar eventos de entrada e saída de regiões específicas e apoiar a tomada de decisão por gestores de frota. Essas

funcionalidades são empregadas tanto para aumentar a segurança quanto para otimizar a eficiência operacional dos veículos.

De acordo com a Geotab (Geotab Inc., 2024), as geocercas permitem a configuração de regras automáticas associadas à localização, possibilitando o envio de alertas em tempo real e a geração de relatórios baseados no comportamento dos motoristas em áreas previamente definidas. Essa abordagem é conceitualmente semelhante à adotada no sistema proposto neste trabalho, que utiliza geocercas para definir limites de velocidade dinâmicos e emitir alertas conforme as condições detectadas.

A correlação entre o monitoramento contínuo e a redução de infrações é observada em estudos recentes. Um exemplo é o caso apresentado por SILVA e SANTOS (2024), no qual um sistema de telemetria foi implantado em uma empresa de grande porte do setor de mineração, localizada em Congonhas do Campo, Minas Gerais. Os resultados obtidos indicaram uma redução significativa na frequência de excessos de velocidade ao longo do período de monitoramento, evidenciando o impacto positivo da utilização de dados telemétricos como ferramenta de apoio à conscientização dos motoristas e à atuação dos gestores.

A Figura 3.1 apresenta a evolução do número de registros de excesso de velocidade ao longo dos meses analisados, evidenciando uma tendência de redução progressiva após a implantação do sistema de telemetria, o que indica o impacto positivo do monitoramento no comportamento dos motoristas.

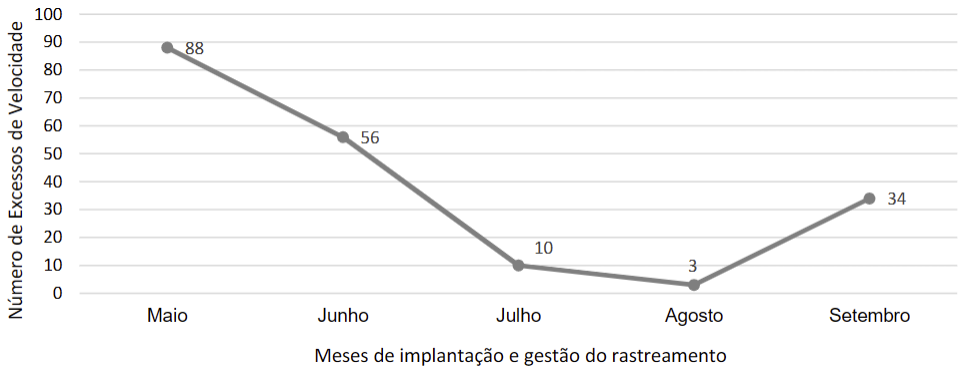


Figura 3.1 – Excessos de velocidade após a implantação do sistema de telemetria.

Fonte: SILVA e SANTOS (2024).

Esse conceito fundamenta o funcionamento do sistema proposto neste trabalho, que utiliza geocercas para definir limites de velocidade adaptáveis e emitir alertas automáticos conforme as condições detectadas.

4 METODOLOGIA

O objetivo deste capítulo é apresentar de forma estruturada a concepção e a implementação do protótipo desenvolvido, abrangendo tanto os aspectos de hardware, responsáveis pela coleta, processamento e armazenamento local dos dados, quanto os componentes de software, que realizam o gerenciamento, a análise e a visualização das informações em ambiente web.

A metodologia adotada baseia-se em uma arquitetura distribuída, organizada em quatro camadas principais: requisitos do sistema, dispositivo embarcado, API e plataforma de monitoramento. A camada de requisitos define as necessidades funcionais e operacionais que orientam o desenvolvimento do sistema como um todo. O dispositivo embarcado é responsável pela coleta, tratamento inicial e armazenamento local dos dados veiculares; a API atua como intermediária na validação, comunicação e persistência das informações em um banco de dados; e a plataforma web fornece uma interface interativa para visualização, análise e acompanhamento em tempo real dos dados coletados.

4.1 Requisitos do sistema

Os requisitos do sistema, tanto de hardware quanto de software, foram definidos a partir da observação de um sistema de telemetria veicular real, atualmente em operação em uma empresa do setor de mineração, cuja identificação não será divulgada por motivos de confidencialidade. Uma empresa terceirizada, contratada por essa organização, autorizou a análise do funcionamento do sistema existente, possibilitando a identificação das principais necessidades operacionais e funcionais do dispositivo em uso.

Com base nessas observações, foram levantados os requisitos essenciais para a concepção do sistema proposto neste trabalho. As tecnologias empregadas foram selecionadas de modo a viabilizar uma solução de construção simples, custo relativamente reduzido e fácil replicação. O microcontrolador ESP32, por exemplo, foi escolhido por sua ampla disponibilidade no mercado, integração de recursos de conectividade e facilidade de desenvolvimento, permitindo a implementação de sistemas embarcados com nível moderado de complexidade.

As funcionalidades do sistema também foram definidas com o objetivo de manter a simplicidade. A partir do sistema real anteriormente mencionado, foram selecionadas apenas as funções consideradas essenciais, com a intenção de demonstrar a viabilidade de replicação da solução proposta, permitindo futuras expansões e melhorias mediante novos investimentos.

De forma geral, definiu-se que o sistema deveria atender aos seguintes requisitos funcionais básicos:

- **Autenticação de motoristas**: o sistema deve ser capaz de identificar cada motorista que conduziu o veículo durante o período de operação, associando os registros coletados ao respectivo condutor.
- **Registro de percursos**: o sistema deve registrar os trajetos percorridos pelo veículo, incluindo informações de localização geográfica, datas, horários e motoristas envolvidos.
- **Controle de velocidade em tempo real**: a partir da coleta da velocidade diretamente do veículo, o sistema deve ser capaz de aplicar limites de velocidade com base em geocercas previamente definidas, além de registrar ocorrências de ultrapassagem desses limites durante o funcionamento do sistema.

Além das funcionalidades, foram definidos requisitos específicos relacionados ao dispositivo embarcado, visando garantir confiabilidade e robustez durante a operação:

- **Funcionamento autônomo**: o dispositivo deve operar com o mínimo possível de dependências externas, permitindo seu funcionamento mesmo na ausência temporária de conectividade com a internet. Dessa forma, durante o desenvolvimento do sistema, foram previstas lógicas que garantem a continuidade da operação em modo offline.
- **Garantia de persistência dos dados**: o dispositivo deve assegurar que os dados coletados sejam armazenados de forma confiável e transmitidos ao banco de dados em momento oportuno. Para isso, são necessárias estratégias que evitem a corrupção das informações e permitam o registro local dos dados, com envio automático posterior quando a comunicação for restabelecida.
- **Proteção contra interferência física**: a concepção física do dispositivo deve dificultar, tanto quanto possível, intervenções manuais que possam comprometer seu funcionamento. Isso inclui restringir o acesso direto à fonte de alimentação e evitar a exposição dos módulos eletrônicos ao usuário ou a terceiros com acesso ao veículo.

Para a plataforma online de gerenciamento e monitoramento, foram definidos os seguintes requisitos:

- **Listagem e reconstrução de registros**: todos os registros armazenados no banco de dados durante a operação do sistema devem poder ser listados e reconstruídos sobre o mapa. Além disso, a plataforma deve permitir a aplicação de filtros conforme a necessidade do operador, como período de tempo, motorista envolvido ou veículo monitorado.
- **Gerenciamento de geocercas**: a plataforma deve permitir a criação, edição e exclusão de cercas geográficas, bem como o controle e a atualização de seus respectivos limites de velocidade.
- **Rastreamento de veículos**: deve ser possível visualizar, em tempo real, os veículos em operação sobre o mapa, permitindo o acompanhamento do deslocamento e do trajeto que está sendo realizado em um determinado momento.
- **Visualização de alertas**: a plataforma deve disponibilizar uma seção específica para

a visualização de alertas de excesso de velocidade, permitindo tanto a listagem dos eventos quanto sua reconstrução geográfica no mapa, fornecendo feedback visual ao operador do sistema.

Os requisitos definidos para o sistema, tanto no nível do dispositivo embarcado quanto da plataforma de gerenciamento, orientaram diretamente as decisões relacionadas à arquitetura de hardware e software adotadas neste trabalho. A partir dessas necessidades, foi projetada uma solução capaz de atender aos requisitos funcionais e operacionais levantados, priorizando simplicidade, confiabilidade e viabilidade de implementação.

4.2 Arquitetura de Hardware

Esta seção apresenta a composição física do sistema embarcado desenvolvido, detalhando a integração entre os controladores, sensores e módulos de comunicação. São descritos os mecanismos de autenticação de motoristas, registro de viagens, gerenciamento de geocercas e estratégias de armazenamento e atualização de dados.

4.2.1 Integração Física dos Controladores e Módulos

Apesar de o protótipo empregar dois microcontroladores, a lógica principal do sistema está centralizada no *ESP32*. Este dispositivo é responsável pelo gerenciamento das funções de todos os módulos conectados, incluindo o *ESP8266*, que atua de forma auxiliar. Dessa forma, as interligações físicas e o controle dos periféricos concentram-se no *ESP32*, que coordena a comunicação entre os componentes e garante o funcionamento integrado do sistema.

O esquema apresentado na Figura 4.1 foi elaborado no software Fritzing, utilizado para a representação de circuitos em ambientes de prototipagem eletrônica. O diagrama tem como objetivo ilustrar de forma simplificada a interconexão entre os principais módulos e controladores do sistema, conforme descrito a seguir:

- ESP32: microcontrolador central do protótipo, responsável por gerenciar todas as funções dos módulos conectados.
- ESP8266: microcontrolador auxiliar, encarregado exclusivamente das funções de rede, como o recebimento dos dados de cercas e motoristas diretamente da API, além do envio das informações registradas de viagem quando requisitado.
- RC522: módulo RFID destinado à autenticação dos motoristas; a aproximação de um cartão RFID é capaz de iniciar ou finalizar uma viagem.
- GPS NEO6-M: responsável pela coleta de informações de localização via sinal de GPS e pelo repasse dos dados ao ESP32, permitindo o registro preciso do percurso.
- 4MD36: módulo adaptador para cartão SD, cuja função inclui não apenas o armazenamento local dos dados antes do envio à API, mas também o registro de informações

sobre motoristas e cercas, possibilitando a operação do sistema mesmo sem conexão à rede.

- LCD I2C: exibe ao usuário o limite de velocidade definido em comparação com a velocidade atual, além de outras informações relevantes, como erros de operação, estados do processamento e identificação do motorista durante a autenticação via RFID.
- Buzzer piezoelétrico: dispositivo responsável por emitir sinais sonoros auxiliares ao LCD, tendo como principal função alertar o motorista ao ultrapassar o limite de velocidade, além de indicar outros estados do sistema para aprimorar a experiência do usuário.

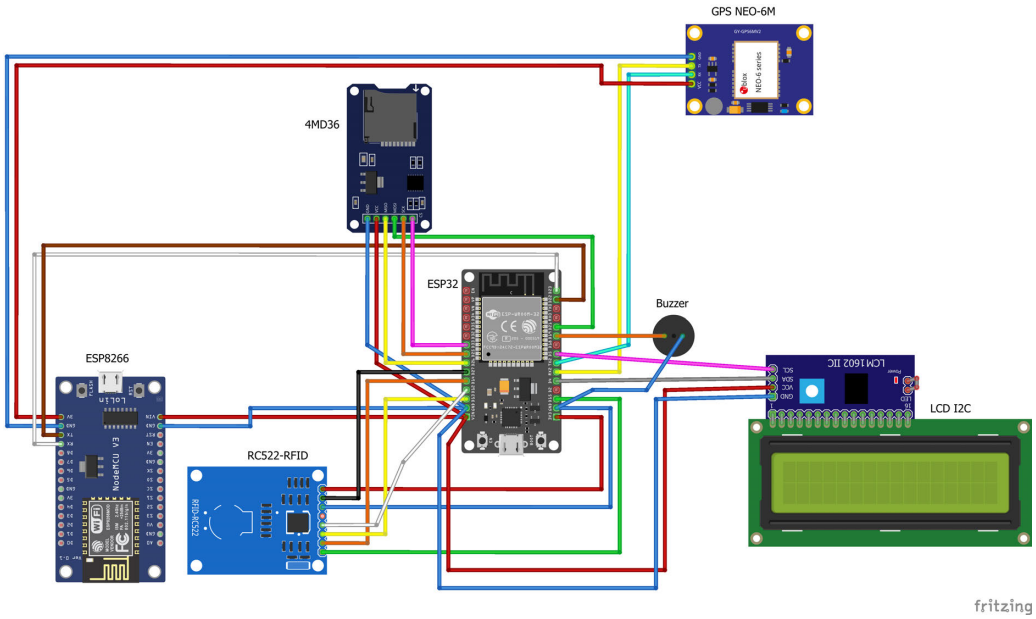


Figura 4.1 – Esquema geral de ligação dos controladores e módulos no simulador.

Fonte: Elaboração própria (Fritzing).

Observa-se que, conforme mencionado anteriormente, tanto o ESP32 quanto o ESP8266 possuem conectividade Wi-Fi integrada. Essa característica torna a versão atual do protótipo dependente de uma rede Wi-Fi externa para realizar a comunicação com a API, tanto para envio quanto para recebimento de dados. Contudo, essa dependência não é adequada para um cenário real de aplicação, uma vez que não é comum que veículos disponham de conexão Wi-Fi própria. Portanto, para uma implementação prática, o sistema deve ser capaz de prover sua própria conectividade de rede.

Uma alternativa viável consiste na utilização de módulos de comunicação celular que permitam o uso de chips de rede móvel. Entre as opções disponíveis, destaca-se o módulo SIM7600E (SIMCom Wireless Solutions Co., Ltd., 2016), compatível com redes 4G LTE e com suporte adicional a GNSS, permitindo comunicação de dados em redes mais atuais. Entretanto, esse módulo apresenta baixa disponibilidade no mercado nacional, sendo normalmente necessária sua importação.

Outra alternativa amplamente encontrada no Brasil é o módulo GSM/GPRS SIM800L (SIMCom Wireless Solutions, 2015). Esse módulo opera exclusivamente em redes 2G, o que pode representar uma limitação futura, considerando as discussões e o processo gradual de desligamento das redes 2G e 3G no território brasileiro (Lumini Tracker, 2025).

Diante desse cenário, optou-se por considerar, para versões futuras do protótipo, a adoção direta de um módulo compatível com redes 4G, de modo a garantir maior longevidade tecnológica ao sistema e evitar possíveis problemas decorrentes da descontinuação de infraestruturas legadas.

4.2.2 Fluxo Geral de Processamento

Esta seção apresenta, de forma simplificada, o fluxo de funcionamento do dispositivo desenvolvido. O objetivo é fornecer uma visão geral do processamento interno, ressaltando que a descrição a seguir não aborda as lógicas detalhadas nem todos os trechos de código, mas apenas os principais eventos e comportamentos do sistema.

A seguir, são descritas as etapas que compõem o fluxo geral de processamento do dispositivo:

1. Ao ser energizado, o sistema inicia a sequência de inicialização dos módulos. Todas as bibliotecas, objetos e periféricos são configurados, garantindo o funcionamento integrado dos componentes. Caso a inicialização ocorra sem falhas, o dispositivo estará pronto para iniciar os registros; se algum erro for detectado, uma indicação visual é exibida no display LCD. Embora esse trecho não seja dedicado exclusivamente à detecção de erros, ele é fundamental para a verificação de integridade inicial do sistema.
2. Em seguida, ocorre a atualização dos dados de cercas e motoristas (caso haja conexão disponível). As informações de motoristas são utilizadas na lógica de autenticação, realizando o cruzamento entre o cartão RFID lido e os registros armazenados, enquanto os dados de cercas permitem comparar as coordenadas do veículo com as coordenadas das cercas previamente definidas na plataforma. Esses dados são armazenados como objetos JSON no cartão SD e são essenciais, pois possibilitam a operação autônoma do dispositivo mesmo na ausência de rede, garantindo maior rapidez e confiabilidade nas verificações.
3. Como a atualização é executada em segundo plano, o dispositivo torna-se imediatamente operacional após a inicialização. Para iniciar uma viagem, o motorista aproxima o cartão RFID, que é autenticado pelo sistema. Caso o motorista esteja devidamente cadastrado na base de dados da telemetria, o registro da viagem é iniciado automaticamente.
4. O registro de viagem é realizado em intervalos configuráveis, podendo variar conforme as preferências da empresa. A cada nova viagem, um objeto JSON é criado no cartão SD, armazenando informações como velocidade, localização, motorista e veículo. Em

cada ciclo de coleta, novos registros são adicionados a esse arquivo.

5. Após cada registro, o sistema tenta enviar os dados para a API. Foram implementadas lógicas de segurança e mecanismos de backup destinados a preservar a integridade das informações armazenadas no cartão SD e evitar qualquer tipo de perda de dados. Esses mecanismos também contemplam situações em que não há conexão com a internet, condição considerada comum em cenários reais, e serão detalhados em seções posteriores.
6. Quando o leitor RFID detecta a aproximação do cartão pela segunda vez, se for autenticado, o dispositivo realiza um registro final imediato, independente do intervalo configurado, e tenta enviar esses dados à API. Esse processo marca o encerramento da viagem.
7. Com o encerramento do ciclo de registros, o dispositivo retorna ao estado inicial, permanecendo pronto para o início de uma nova viagem.

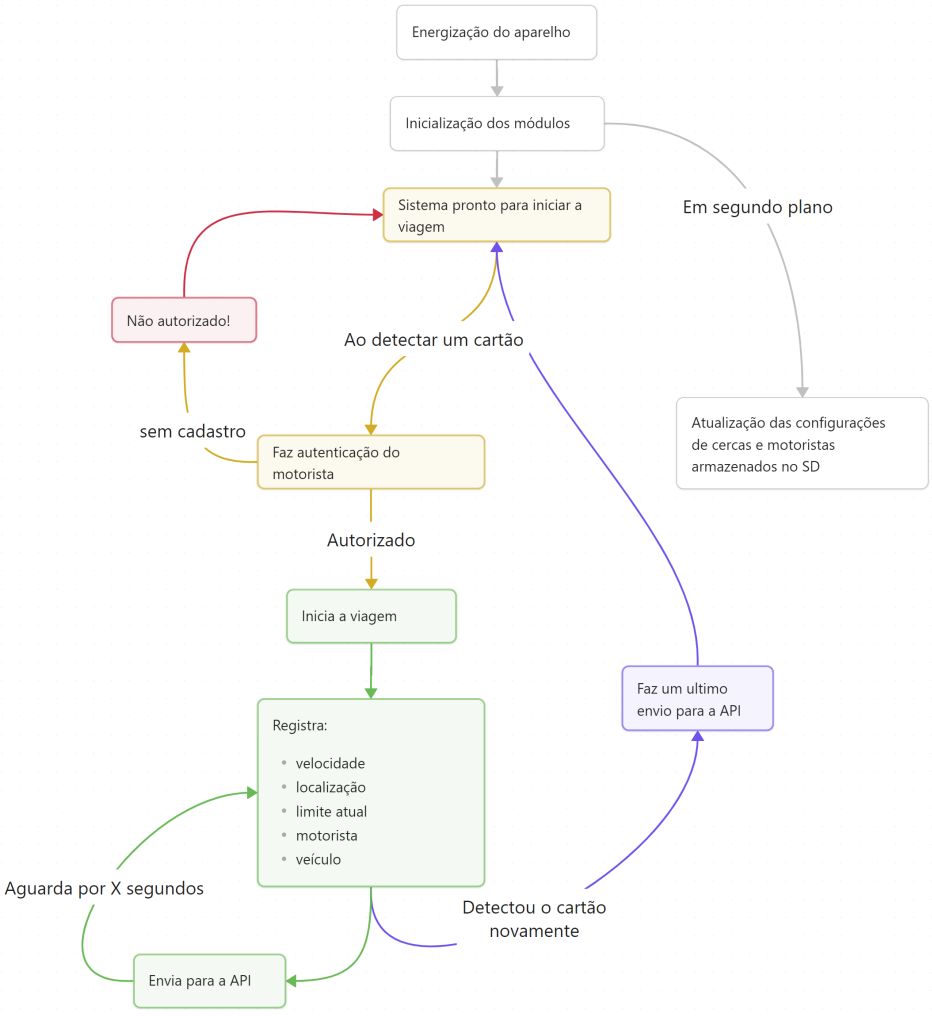


Figura 4.2 – Fluxo geral de funcionamento do dispositivo embarcado.

Fonte: Elaboração própria.

4.2.3 Lógica de Armazenamento e Atualização de Dados Recebidos da API

As listas de motoristas cadastrados no sistema e as cercas criadas no mapa permanecem sempre disponíveis para uso offline, sendo armazenadas no cartão SD em formato *JSON*. Esses arquivos desempenham papel fundamental no funcionamento do sistema, pois a lista de motoristas é empregada no processo de autenticação, e a lista de cercas é utilizada para comparar as coordenadas do veículo com as delimitações poligonais previamente desenhadas no mapa.

As listas são atualizadas automaticamente a cada inicialização do sistema e também em intervalos periódicos configuráveis. Na versão atual do protótipo, o intervalo está definido em cinco minutos. A cada atualização, o ESP32, responsável pelo gerenciamento dos dados armazenados no cartão SD, solicita ao ESP8266 a atualização das informações. Essa comunicação ocorre por meio de comandos específicos, sendo o primeiro utilizado para verificar o status da conexão Wi-Fi e o segundo para requisitar os novos dados de motoristas e cercas.

Caso o ESP8266 não possua conexão com a rede, o ESP32 cancela imediatamente o processo de atualização, evitando consumo desnecessário de recursos de processamento. Caso haja conexão ativa, o ESP8266 executa a requisição à API, que fornece rotas específicas para o envio de dados resumidos, otimizando tanto o tempo de resposta quanto o processamento pelos microcontroladores. Durante esse período, o ESP32 permanece em estado de espera, utilizando tarefas assíncronas, conceito detalhado em seções posteriores. Assim que o ESP8266 recebe os dados da API, estes são transmitidos diretamente ao ESP32.

Devido às limitações de memória de ambos os microcontroladores, foi implementada uma lógica de transmissão baseada em fluxo (*streaming*). Nessa abordagem, em vez de carregar todo o arquivo JSON na memória RAM do ESP8266, o fluxo de dados identifica os delimitadores da estrutura (como chaves e colchetes) e envia cada segmento individualmente ao ESP32. Após o envio, o segmento é descartado, liberando memória para a leitura e o envio do próximo trecho até a conclusão do arquivo.

O ESP32, por sua vez, recebe os segmentos e aplica o mesmo princípio para gravá-los no cartão SD. Assim que a transmissão é iniciada, o ESP32 cria um arquivo temporário e, à medida que cada segmento é recebido, ele é imediatamente escrito no armazenamento, montando o JSON progressivamente, como um quebra-cabeças. Essa abordagem, embora eficiente, pode introduzir riscos de inconsistência, uma vez que o arquivo completo não é processado em memória. Para mitigar esse risco, os dados são inicialmente gravados em arquivos temporários, evitando a substituição imediata dos arquivos utilizados pelo sistema.

Após o término da gravação, o ESP32 executa uma verificação de integridade dos dados. Considerando as restrições de memória, é utilizado um método de contagem de delimitadores: o sistema verifica se todas as chaves e colchetes de abertura possuem

correspondentes de fechamento. Embora essa técnica não garanta integridade total, ela reduz significativamente a probabilidade de utilização de dados corrompidos, visto que falhas no fluxo de transmissão dificilmente manteriam uma estrutura JSON sintaticamente válida.

Caso ocorra qualquer falha durante o processo, sendo a perda de conexão com a rede a mais comum, a atualização é imediatamente interrompida, sem uma segunda tentativa. Isso porque o sistema sempre irá garantir novas tentativas, seja na próxima inicialização do dispositivo ou na execução do ciclo periódico de atualização. Dessa forma, se ocorrer uma falha na atualização durante a inicialização, os arquivos de cercas e motoristas anteriormente utilizados permanecem válidos, e uma nova tentativa ocorrerá após o intervalo configurado.

O diagrama a seguir ilustra, de forma visual, o fluxo de processamento realizado a cada atualização das listas de cercas e motoristas.

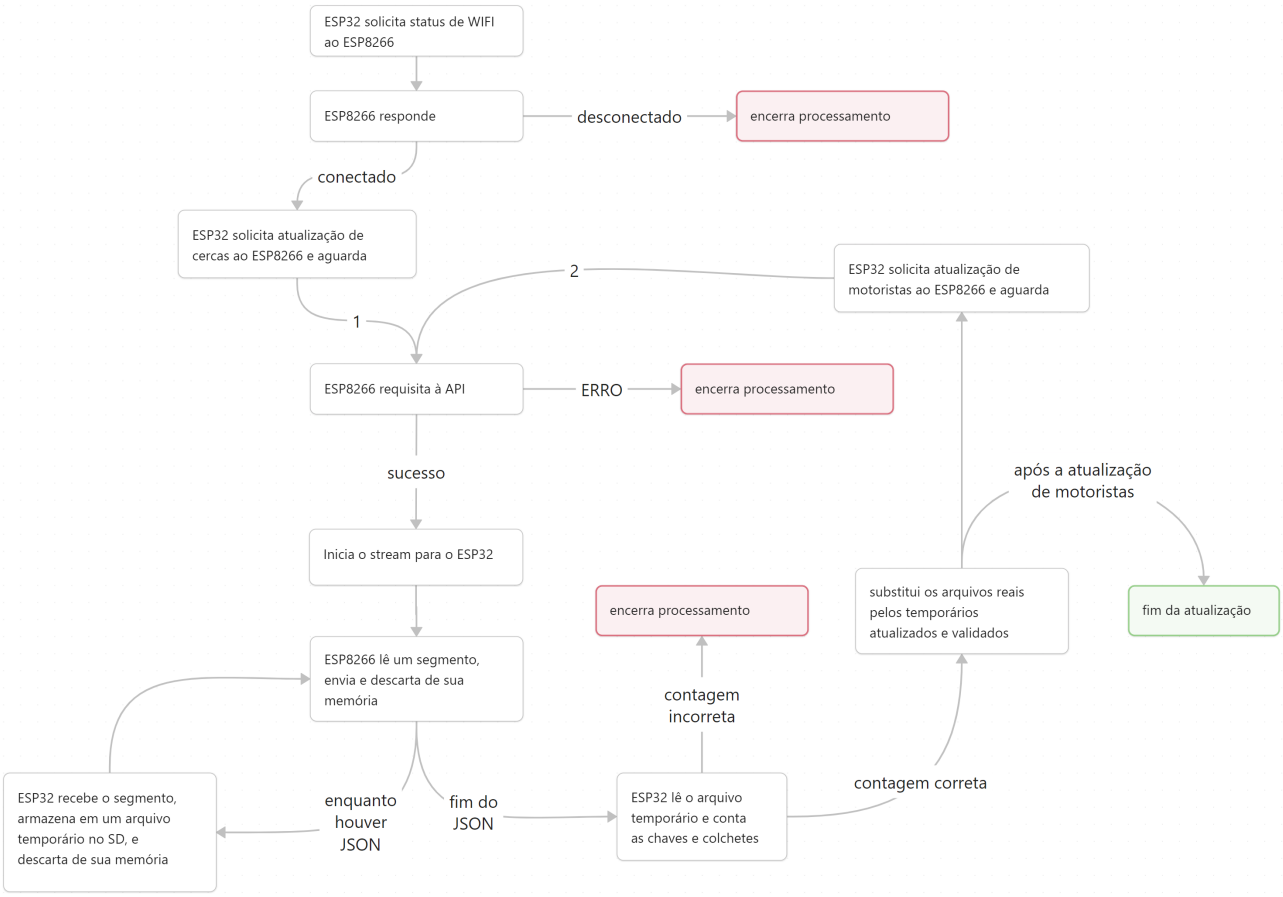


Figura 4.3 – Lógica de atualização de cercas e motoristas.

Fonte: autoria própria.

4.2.4 Lógica de Comunicação entre ESP32 e ESP8266

Durante o desenvolvimento do protótipo, foi utilizado inicialmente apenas o ESP32, que apresentou desempenho consistente nas etapas preliminares. Entretanto, à medida que novas bibliotecas e módulos foram incorporados ao sistema, observou-se um estouro de memória já durante a fase de inicialização, impossibilitando o funcionamento completo do código. Diante dessa limitação, tornou-se necessária a inclusão de um segundo microcontrolador para redistribuir as tarefas e reduzir a carga sobre o ESP32. O microcontrolador escolhido foi o ESP8266, cuja seleção se deu, a princípio, por ser o componente disponível no momento do desenvolvimento. Apesar disso, ele demonstrou excelente desempenho como controlador secundário, operando de forma integrada ao controlador principal.

A decisão de transferir a conexão Wi-Fi para o ESP8266 foi motivada pelo consumo de memória da biblioteca responsável por essa funcionalidade, que comprometia parte significativa da RAM (Random Access Memory) do ESP32. Essa redistribuição permitiu otimizar os recursos do sistema, viabilizando a adição das lógicas pendentes sem a necessidade de modificações estruturais extensas no código original, preservando sua consistência e acelerando o processo de desenvolvimento.

Alternativamente, essa limitação poderia ser contornada por meio da utilização de uma variante do ESP32 equipada com PSRAM (Pseudo Static RAM), que consiste em um chip adicional de memória integrado à placa do microcontrolador (Espressif Systems, 2025). Essa configuração permitiria centralizar todas as funcionalidades do sistema em um único microcontrolador, incluindo as operações de conectividade Wi-Fi. No entanto, essa alternativa não foi adotada nesta etapa do projeto devido ao custo mais elevado dessa variação e à disponibilidade prévia de um segundo microcontrolador, optando-se, portanto, pela solução adotada nesta versão inicial do sistema.

A Figura 4.4 apresenta o esquema de conexão física entre os dois microcontroladores, destacando que o ESP8266 é alimentado pelo pino de 5 V do ESP32.

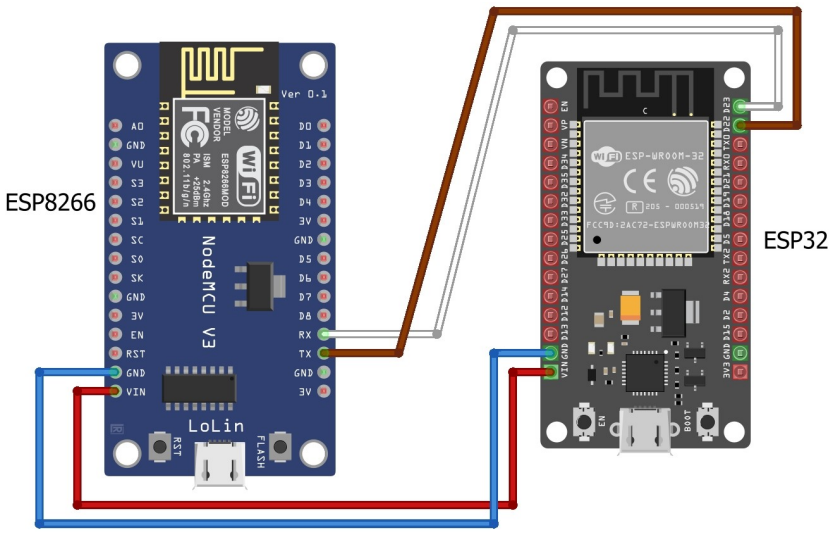


Figura 4.4 – Conexão física entre os microcontroladores.

Fonte: Elaboração própria (Fritzing).

A comunicação entre os dois microcontroladores é realizada por meio da interface UART, onde o ESP32 envia comandos textuais ao ESP8266, que interpreta o conteúdo recebido e executa a função correspondente. Dependendo da natureza do comando, o ESP32 pode ou não aguardar a conclusão do processamento antes de prosseguir. Os principais comandos implementados são os seguintes:

`GET_WIFI`: solicita ao ESP8266 o status de conexão Wi-Fi, retornando uma resposta positiva ou negativa conforme o estado atual da rede.

`GET_CERCAS`: solicita que o ESP8266 realize a requisição das cercas geográficas à API e retorne o resultado via stream (a lógica completa é descrita na subseção 4.2.3).

`GET_MOTORISTAS`: solicita a obtenção da lista de motoristas por meio da API, com o retorno igualmente enviado via stream.

`POST_VIAGEM`: solicita o envio dos dados de viagem ao servidor. Nesse caso, o ESP32 transmite o conteúdo via stream ao ESP8266, que realiza a requisição e informa se o envio foi bem-sucedido. O ESP32 decide, conforme o caso, se deve aguardar a resposta antes de prosseguir.

A arquitetura dos códigos de ambos os microcontroladores foi estruturada de modo a permitir a expansão modular, possibilitando a adição de novos comandos com relativa facilidade.

Uma limitação identificada foi o fato de o ESP8266 dispor de apenas uma porta UART, enquanto o código do ESP32 executa tarefas assíncronas que demandam comunicação de rede. Para contornar essa restrição, foi desenvolvida uma lógica de operações em fila, na qual uma nova tarefa assíncrona no ESP32 se encarrega de processar sequencialmente cada um dos quatro comandos. Essa tarefa recebe as solicitações em uma fila e as transmite ao ESP8266 de forma ordenada, garantindo o uso eficiente do único barramento UART disponível.

4.2.5 Lógica de Autenticação de Motoristas

Optou-se pela utilização da tecnologia RFID para a identificação dos motoristas no sistema proposto em função de dois fatores principais. O primeiro está relacionado às alternativas tecnológicas disponíveis. Métodos como reconhecimento facial exigiriam a aplicação de técnicas de visão computacional, implicando a necessidade de hardware mais robusto e maior complexidade de desenvolvimento. Outra possibilidade seria a autenticação por meio de dispositivos móveis, como smartphones, porém essa abordagem poderia comprometer a praticidade do sistema, aumentando o esforço exigido do motorista durante a operação do veículo.

O segundo fator refere-se à padronização e à aderência a práticas já consolidadas no ambiente corporativo. Em empresas de médio e grande porte, é comum que os funcionários possuam cartões de proximidade para controle de acesso e registro de atividades. Conforme observado durante a análise de um sistema real em operação em uma frota de veículos do setor de mineração, o processo de autenticação dos motoristas era realizado exclusivamente por meio de RFID, sendo a única interação necessária a aproximação do cartão ao leitor.

Esse comportamento foi adotado como referência neste trabalho, uma vez que se trata de um método com o qual os motoristas já estão familiarizados, além de ser rápido, intuitivo e de fácil aceitação no ambiente operacional. Ademais, a tecnologia RFID apresenta boa viabilidade técnica e econômica, tornando-se adequada aos objetivos do sistema proposto.

Uma tarefa no código do ESP32 permanece em execução contínua em segundo plano, aguardando a detecção de um cartão RFID. Quando a leitura ocorre, o ID (Identificador) do cartão é imediatamente comparado com a lista de motoristas armazenada no cartão SD. Caso o ID seja encontrado, o sistema inicia uma nova viagem, sinalizada por um aviso sonoro emitido pelo dispositivo. Em seguida, o nome do motorista correspondente é exibido no display, acompanhado por uma mensagem de boas-vindas. Essa etapa, embora pareça apenas informativa, tem como principal objetivo fornecer um retorno visual e auditivo ao usuário, confirmando qual motorista foi autenticado com sucesso.

Se o veículo começar a se mover antes que o motorista realize a autenticação por meio do cartão RFID, o dispositivo detectará esse deslocamento e ativará o modo de alerta, insistindo para que o motorista efetue a leitura do cartão. O sistema estabelece um tempo limite para essa autenticação, que, na versão atual do protótipo, é de quinze segundos. Caso o motorista ignore o alerta e continue a condução sem autenticação, o dispositivo iniciará o registro da viagem utilizando o perfil padrão de motorista, identificado como "Desconhecido". Dessa forma, mesmo sem a autenticação, todas as informações referentes ao percurso, ao veículo e ao momento da ocorrência (data e hora) serão devidamente registradas, permitindo que o operador identifique esses eventos posteriormente e realize a devida verificação.

Ao final do trajeto, o motorista deve novamente aproximar o cartão RFID do

leitor, sinalizando ao sistema o encerramento da viagem. Nesse momento, os registros coletados são enviados à API em segundo plano, enquanto, em primeiro plano, o sistema permanece pronto para iniciar uma nova viagem imediatamente. Detalhes adicionais sobre essa lógica de registro e transmissão de viagens são apresentados em seção específica deste trabalho.

Outro cenário possível ocorre quando, durante uma viagem em andamento, um segundo motorista aproxima o seu cartão RFID sem que o anterior tenha finalizado o trajeto. Nessa situação, o sistema encerra automaticamente a viagem anterior, realiza o envio dos dados para a API em segundo plano e, em seguida, inicia uma nova viagem em nome do motorista recém-autenticado.

Por fim, há o caso em que o dispositivo é desenergizado antes que uma viagem seja devidamente encerrada. Ao ser religado, o sistema é capaz de identificar registros de viagens não finalizadas armazenados no cartão SD e enviá-los imediatamente à API. Essa lógica assegura que os dados de telemetria sejam sempre preservados e eventualmente transmitidos ao banco de dados, minimizando a possibilidade de perdas e garantindo a consistência das informações coletadas.

4.2.6 Registro de Viagens e Aquisição de Localização

Após a autenticação do motorista, o dispositivo inicia o processo de registro das viagens. Esse processo ocorre periodicamente, em um intervalo configurável no código, definido nesta versão do protótipo como quinze segundos.

No início de cada registro, o sistema cria um arquivo no formato JSON no cartão SD, dentro de uma pasta específica destinada às viagens em andamento. Nesse arquivo, são inseridas as informações preliminares no cabeçalho do JSON, como o identificador do motorista envolvido, o identificador do veículo, um identificador auxiliar da viagem e uma lista inicialmente vazia destinada aos registros de posição. O identificador auxiliar é gerado de forma aleatória e única pelo microcontrolador. Embora não seja utilizado como chave primária no banco de dados, ele é essencial para a lógica de comunicação com a API, permitindo que o sistema identifique, a partir desse valor, se a viagem em questão já está registrada ou se trata de um novo percurso iniciado (esse processo será detalhado a seguir).

A cada ciclo de registro, o sistema obtém as coordenadas fornecidas pelo módulo GPS e as compara com as coordenadas definidas nas cercas geográficas cadastradas no sistema. As cercas geográficas são compostas por pontos que, quando conectados, formam polígonos. A verificação da posição do veículo é realizada com base no método de ray casting, que consiste em traçar uma linha horizontal imaginária a partir do ponto definido pelas coordenadas atuais do veículo e contabilizar quantas vezes essa linha cruza as arestas do polígono. Se o número de interseções for par, o veículo está fora da cerca; se for ímpar, considera-se que o veículo está dentro da área delimitada. Embora essa explicação

represente uma simplificação conceitual, na prática, o cálculo é inteiramente matemático e executado de forma eficiente pelo microcontrolador.

A Figura 4.5 ilustra de forma simplificada o funcionamento do método de verificação de cercas utilizado no sistema.

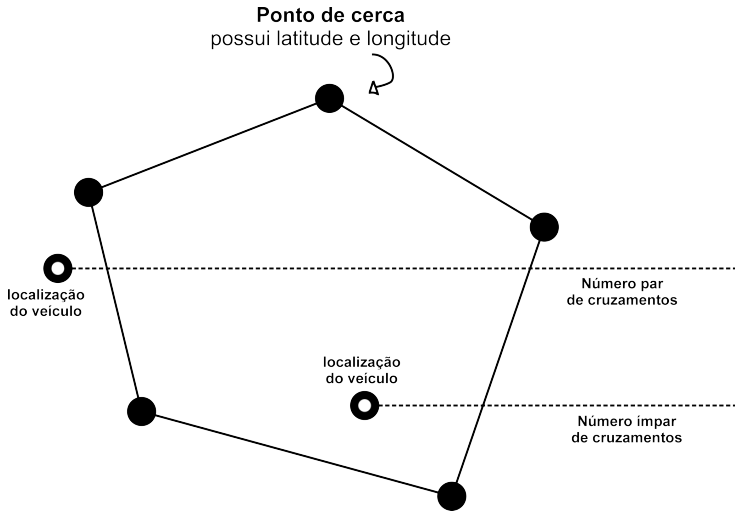


Figura 4.5 – Representação visual do método Ray casting

Fonte: Autoria própria.

Essa verificação é feita repetidamente em todas as cercas, mesmo depois de detectar que o veículo já está dentro de uma, isso porque no sistema é permitido que uma cerca seja desenhada dentro de outra. Nesse caso, durante a verificação, caso ocorra esse acúmulo de limites, o menor limite será considerado.

Após essa verificação, as coordenadas do veículo, sua velocidade, data, hora e os limites definidos são montados em um objeto que será adicionado à lista dentro do JSON de viagens.

O código a seguir é um exemplo de estrutura de viagem que seria criada e armazenada:


```
{
    "viagem_id":3846856777,
    "motorista_id":9,
    "veiculo_id":2,
    "registros":[
        {"timestamp":"2025-10-08T20:01:39Z","lat":-3.757600,"lng":-49.683205,
        "vel":0.00,"chuva":false,"lim_seco":60,"lim_chuva":50},
        {"timestamp":"2025-10-08T20:02:13Z","lat":-3.754320,"lng":-49.685539,
        "vel":0.00,"chuva":false,"lim_seco":60,"lim_chuva":50},
        {"timestamp":"2025-10-08T20:02:31Z","lat":-3.751728,"lng":-49.687336,
        "vel":0.00,"chuva":false,"lim_seco":60,"lim_chuva":50},
```

Percebe-se que o JSON apresentado não contém a chave e o colchete de fechamento, pois essa estrutura foi planejada para que o microcontrolador apenas adicione novos blocos ao final do arquivo a cada registro. Essa abordagem simplifica o processamento e a recuperação das viagens, conforme será detalhado a seguir.

A cada novo registro inserido no arquivo JSON, o sistema também solicita o envio desses dados à API. Esse procedimento ocorre por meio da criação de um arquivo temporário, no qual são adicionados os delimitadores finais. Em seguida, o ESP8266 é instruído a realizar o envio desse arquivo, transmitindo-o via stream para a API (lógica descrita na Subseção 4.2.3). O JSON completo é então recebido pela rota da API, que identifica se a viagem já está registrada no banco de dados com base no identificador gerado pelo microcontrolador. Dessa forma, o sistema consegue determinar quais pontos do trajeto já foram armazenados e quais ainda precisam ser inseridos, realizando uma atualização simples no caso de viagens já existentes. Esse processo garante a integridade dos dados, mesmo em situações de falha de conexão, pois assegura que todos os registros serão eventualmente transmitidos ao servidor, seja durante os ciclos normais de envio ou durante o processo de recuperação na inicialização do dispositivo.

Ao término da viagem, o sistema adiciona o fechamento adequado diretamente no arquivo principal e solicita o envio final à API. Nessa etapa, o ESP32 aguarda uma resposta do servidor. Caso a confirmação seja positiva, o arquivo é movido para a pasta de viagens concluídas; caso contrário, ele permanece na pasta de viagens pendentes. Essa organização permite que o dispositivo verifique facilmente se há viagens não enviadas, reintentando automaticamente o envio e evitando perda de dados. A pasta de viagens pendentes é inspecionada tanto durante a inicialização do sistema quanto a cada tentativa de envio de uma nova viagem em andamento.

Observação importante: conforme mencionado na introdução deste artigo, embora existam trechos de código que indicam uma funcionalidade de detecção de chuva, essa lógica ainda não foi implementada na versão atual do sistema. No entanto, esses fragmentos foram mantidos para facilitar sua integração em versões futuras.

4.2.7 Aquisição de Dados do Veículo

A aquisição dos dados do veículo é realizada por meio do barramento CAN (Controller Area Network), utilizando o scanner ELM327, conectado à porta OBD-II do automóvel. Esse dispositivo interpreta as mensagens trafegadas no barramento e as transmite ao ESP32 via comunicação Bluetooth Classic, possibilitando a leitura de parâmetros em tempo real, como a velocidade, importante para a lógica do código.

Uma alternativa à aquisição da velocidade por meio do ELM327 seria a utilização do próprio módulo de GPS já presente no dispositivo. Essa abordagem eliminaria a necessidade do scanner OBD-II, simplificando a arquitetura do sistema. Entretanto, a velocidade estimada por GPS apresenta limitações relacionadas à precisão e à taxa de

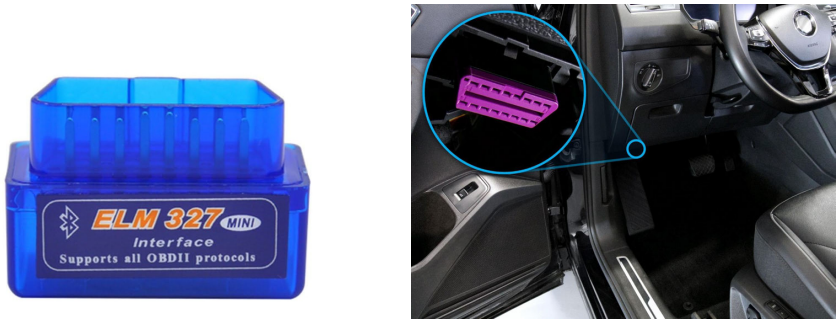
atualização, podendo sofrer atrasos e variações decorrentes do processamento interno e das condições de recepção do sinal. Além disso, em determinadas regiões, especialmente em ambientes fechados ou com obstruções físicas, a disponibilidade do sinal de satélite pode ser comprometida, impactando a confiabilidade da medição.

O uso do ELM327, por sua vez, permite a obtenção direta da velocidade informada pela unidade de controle eletrônico (ECU) do veículo por meio do barramento CAN, proporcionando maior precisão e menor latência na atualização dos dados. Dessa forma, pequenas variações na velocidade podem ser detectadas praticamente em tempo real, o que é essencial para a lógica de controle e geração de alertas do sistema.

A escolha da versão do ELM327 com comunicação via Bluetooth ocorreu em função de sua ampla disponibilidade no mercado e da simplicidade de integração com o ESP32. As alternativas disponíveis incluem versões com interface USB e Wi-Fi. A utilização da versão USB demandaria a implementação de módulos adicionais e rotinas específicas para interpretação e gerenciamento da comunicação serial, aumentando a complexidade do sistema. Nesse contexto, a interface Bluetooth mostrou-se mais adequada para a proposta inicial do protótipo.

Quanto à versão com comunicação via Wi-Fi, sua adoção poderia representar uma alternativa viável em versões futuras do projeto, uma vez que a rede criada pelo próprio ELM327 tende a oferecer maior estabilidade em comparação ao Bluetooth, que pode sofrer interferências ambientais. Contudo, na arquitetura atual do sistema, o ESP32 já utiliza sua interface Wi-Fi para comunicação com a API remota. Como o microcontrolador não pode manter simultaneamente duas conexões Wi-Fi distintas em modo estação, a utilização do ELM327 Wi-Fi tornaria a arquitetura inviável na configuração adotada, justificando a escolha da versão Bluetooth nesta etapa do desenvolvimento.

Figura 4.6 – Dispositivo ELM327 e sua interface OBD-II em veículos.



(a) Dispositivo ELM327 (imagem ilustrativa).

(b) Porta OBD-II e sua localização típica em veículos.

Fonte: Geotab. *Como funciona a leitura de dados OBD: mitos, verdades e gestão de frotas*. Disponível em: <https://www.geotab.com/pt-br/blog/como-funciona-leitura-obd-mitos-verdades/>. Acesso em: 17 out. 2025.

O barramento CAN funciona como uma rede interna de comunicação no veículo.

As ECUs (Electronic Control Units — unidades de controle eletrônico) trocam informações através dessa rede, enviando dados como velocidade, rotação do motor, temperatura, pressão e outros parâmetros. Parte dessas informações segue um padrão comum entre veículos, enquanto outras variam conforme o modelo e a marca. A porta OBD-II está conectada diretamente a essa rede, centralizando os dados. O módulo ELM327 é capaz de receber, interpretar e transmitir essas informações para dispositivos externos por meio de Bluetooth Classic.

A comunicação entre o ELM327 e o ESP32 utiliza o formato ASCII, um sistema de codificação criado para representar caracteres em dispositivos digitais, padronizado para a interface OBD-II. O microcontrolador envia comandos como `010D` para a interface OBD-II, que responde com algo como `410D1E`. Essa resposta é interpretada pelo microcontrolador, permitindo extrair o valor desejado. Com o envio contínuo desses comandos, é possível atualizar as leituras em tempo real.

Na versão atual do sistema, apenas o parâmetro de velocidade está sendo extraído, sendo utilizado para o registro de percurso, verificação de limites de velocidade e detecção de movimento não autorizado (saída sem autenticação).

4.2.8 Situações de Alerta

Há duas situações em que o sistema entra em modo de alerta: quando o veículo é movimentado sem autenticação do motorista, e quando ocorre ultrapassagem do limite de velocidade definido em alguma cerca geográfica do sistema.

O primeiro caso foi detalhado na Subseção 4.2.6, mas, recapitulando, quando o sistema não possui uma viagem em andamento e detecta movimento do veículo a partir da velocidade recebida do módulo ELM327, ele entra em modo de alerta de saída não autorizada. Nessa condição, o dispositivo emite um sinal sonoro insistente solicitando a autenticação do motorista. Caso a autenticação não seja realizada dentro do tempo configurado, o sistema realiza o registro mesmo assim, coletando todos os dados normalmente, porém vinculando o registro a um motorista padrão identificado como “Desconhecido”. O objetivo dessa lógica é evidenciar ao operador da plataforma que ocorreu uma saída não autorizada e, ao mesmo tempo, disponibilizar todas as informações possíveis, como veículo envolvido, momento da ocorrência e trajeto percorrido, para permitir a investigação e identificação do responsável.

O segundo caso ocorre quando o motorista está em uma viagem regular, mas excede o limite de velocidade definido em uma das cercas. Nessa situação, o dispositivo pausa temporariamente os registros e emite um sinal sonoro contínuo para alertar o motorista a reduzir a velocidade. Se o motorista corrigir a velocidade dentro do tempo configurado, o sistema retoma os registros normalmente, sem contabilizar o evento no banco de dados. Caso contrário, além de permitir o registro, o intervalo entre amostragens é reduzido, proporcionando maior detalhamento do percurso durante o evento. Assim, o

sistema não cria uma lógica especial para o excesso de velocidade; ele apenas suspende o registro enquanto o alerta está ativo e ajusta a frequência de amostragem se o motorista persistir na infração, garantindo fidelidade dos dados e coerência no histórico de telemetria.

4.3 Arquitetura de Software

Esta seção apresenta a arquitetura de software do sistema, abordando a estrutura do banco de dados, o funcionamento da API e os principais requisitos da plataforma online.

4.3.1 Modelagem e Estrutura do Banco de Dados

A definição de uma estrutura de banco de dados consistente é fundamental para o correto funcionamento da plataforma online e para a organização adequada dos registros enviados pelo dispositivo. A modelagem adotada foi projetada para armazenar as informações de forma genérica e extensível, permitindo a adição de novas funcionalidades à plataforma sem a necessidade de modificações significativas na base de dados.

O banco de dados foi hospedado na plataforma *Neon.tech* (NEON, 2025), que fornece instâncias de *PostgreSQL* em ambiente em nuvem com gerenciamento automatizado de escalabilidade, controle de acesso e segurança. Essa escolha se deu principalmente pela compatibilidade com a API desenvolvida, escrita em *Node.js*, que se comunica com o banco por meio de conexões seguras *SSL*. Além disso, a plataforma oferece suporte a ambientes de desenvolvimento e produção independentes, o que facilita o processo de atualização e manutenção da aplicação sem interrupção dos serviços. Dessa forma, a solução garante o armazenamento confiável dos registros enviados pelos dispositivos e o acesso eficiente aos dados pela plataforma online.

A versão atual do banco de dados é composta por nove tabelas. Seus *IDs* são incrementais, ou seja, gerados automaticamente pelo próprio sistema de gerenciamento do banco de dados. A seguir, são descritas as tabelas e seus respectivos atributos (exceto os identificadores automáticos):

Veículos: responsável por armazenar os dados dos veículos cadastrados. Atualmente, guarda o identificador do veículo (placa, chassi ou identificador único da empresa), modelo e status. O campo *status* indica o estado atual do veículo, por exemplo, “parado” ou “em movimento”, sendo útil para a lógica de exibição da plataforma online.

Motoristas: contém informações dos motoristas cadastrados. Nesta versão, considerando que poderia haver outro banco com dados completos de funcionários, armazena apenas o nome e o código RFID de cada motorista.

Viagens: representa o ponto central dos registros. Todas as informações coletadas pelo dispositivo são organizadas nesta tabela, que armazena:

- ID do motorista envolvido;
- ID do veículo;

- ID de referência, que não serve como chave primária nesta tabela, mas é usado pela API para detectar se uma viagem já foi armazenada (ver subseção 4.2.6);
- *timestamp* de início e fim da viagem;
- coordenadas de origem (latitude e longitude);
- coordenadas de destino;
- chuva detectada: valor booleano que indica se houve chuva durante a viagem.

Registros: representa cada ponto das viagens. A partir da sequência desses pontos, o sistema é capaz de reconstruir o percurso realizado pelo veículo e compreender as condições de operação em cada instante. Esta tabela armazena:

- ID da viagem associada ao registro (chave estrangeira da tabela *viagens*, não confundir com o ID de referência);
- ID do veículo envolvido;
- *timestamp* (data e hora do registro);
- coordenadas pontuais (latitude e longitude), que permitem a reconstrução do percurso;
- velocidade, recebida pelo módulo ELM327 diretamente do barramento CAN e enviada à API;
- variável booleana de chuva, que indica se havia precipitação no momento do registro.

Observa-se que a tabela *registros* não armazena explicitamente a cerca ou o limite de velocidade vigente no momento, porém, a partir das coordenadas, da velocidade e da variável de chuva, o sistema pode reconstruir essas informações com precisão.

Alertas: assim como a tabela *viagens* reúne múltiplos registros para reconstruir um percurso, a tabela *alertas* agrega registros do mesmo tipo para representar ocorrências de infrações. Por exemplo, se forem registrados cinco pontos consecutivos em que o motorista ultrapassou o limite de velocidade em uma área específica, um alerta é criado e esses cinco registros são associados a ele. Na lógica atual do sistema embarcado, um alerta só contém múltiplos registros quando eles ocorrem em sequência. Essa tabela armazena:

- ID da viagem associada;
- ID do veículo associado;
- *timestamp* correspondente ao momento da infração;
- tipo de ocorrência (campo ainda não utilizado, reservado para futuras expansões da plataforma);
- descrição (também não utilizada na versão atual).

Nota-se que o ID do motorista não é armazenado nesta tabela, pois essa informação já está presente na tabela *viagens*, à qual o alerta está vinculado. O mesmo ocorre com o ID do veículo, de modo que sua presença nesta tabela representa um erro de análise de estrutura durante o desenvolvimento. Embora esse equívoco não cause prejuízos ao funcionamento normal do sistema, ele indica que a plataforma como um todo necessita de uma revisão mais aprofundada.

Registros_alertas: tabela intermediária responsável por associar registros a alertas. Diferente da relação entre *viagens* e *alertas*, que é de um para muitos, esta tabela permite uma associação flexível: múltiplos registros podem estar relacionados a um alerta, e múltiplos alertas podem estar vinculados a um mesmo registro. Essa estrutura será essencial em versões futuras do sistema que preveem diferentes tipos de alertas, como alertas de entrada em área restrita. Assim, esta tabela armazena apenas:

- ID do alerta;
- ID do registro associado.

Cercas: são desenhadas como polígonos sobre o mapa, onde cada vértice do polígono representa um ponto com coordenadas geográficas (latitude e longitude). A tabela *cercas* armazena as informações essenciais para representar e configurar esses polígonos na plataforma:

- nome da cerca;
- cor exibida sobre o mapa;
- tipo de cerca, que na versão atual é implementada apenas como limitador de velocidade, mas a estrutura já prevê outros tipos, como áreas restritas;
- velocidade máxima permitida;
- velocidade máxima em condições chuvosas;
- camada, que foi implementada para facilitar a organização e exibição das cercas na plataforma online. Atualmente há apenas um nível de organização.

Pontos_cerca: representam os vértices que compõem cada polígono. Essa tabela é responsável por armazenar os pontos que, quando conectados na ordem correta, formam o contorno da cerca no mapa. Ela armazena:

- ID da cerca à qual o ponto pertence;
- latitude e longitude;
- ordem, que indica a sequência correta dos pontos para a reconstrução do polígono no mapa.

Camadas: são utilizadas apenas para a organização visual e hierárquica das cercas na plataforma online. Elas permitem agrupar cercas de acordo com critérios definidos pelo usuário, facilitando sua exibição e gerenciamento. Essa tabela armazena:

- ID das cercas associadas;
- nome da camada.

A Figura 4.7 apresenta o diagrama entidade-relacionamento do banco de dados, ilustrando a estrutura lógica adotada e as relações entre as tabelas descritas anteriormente.

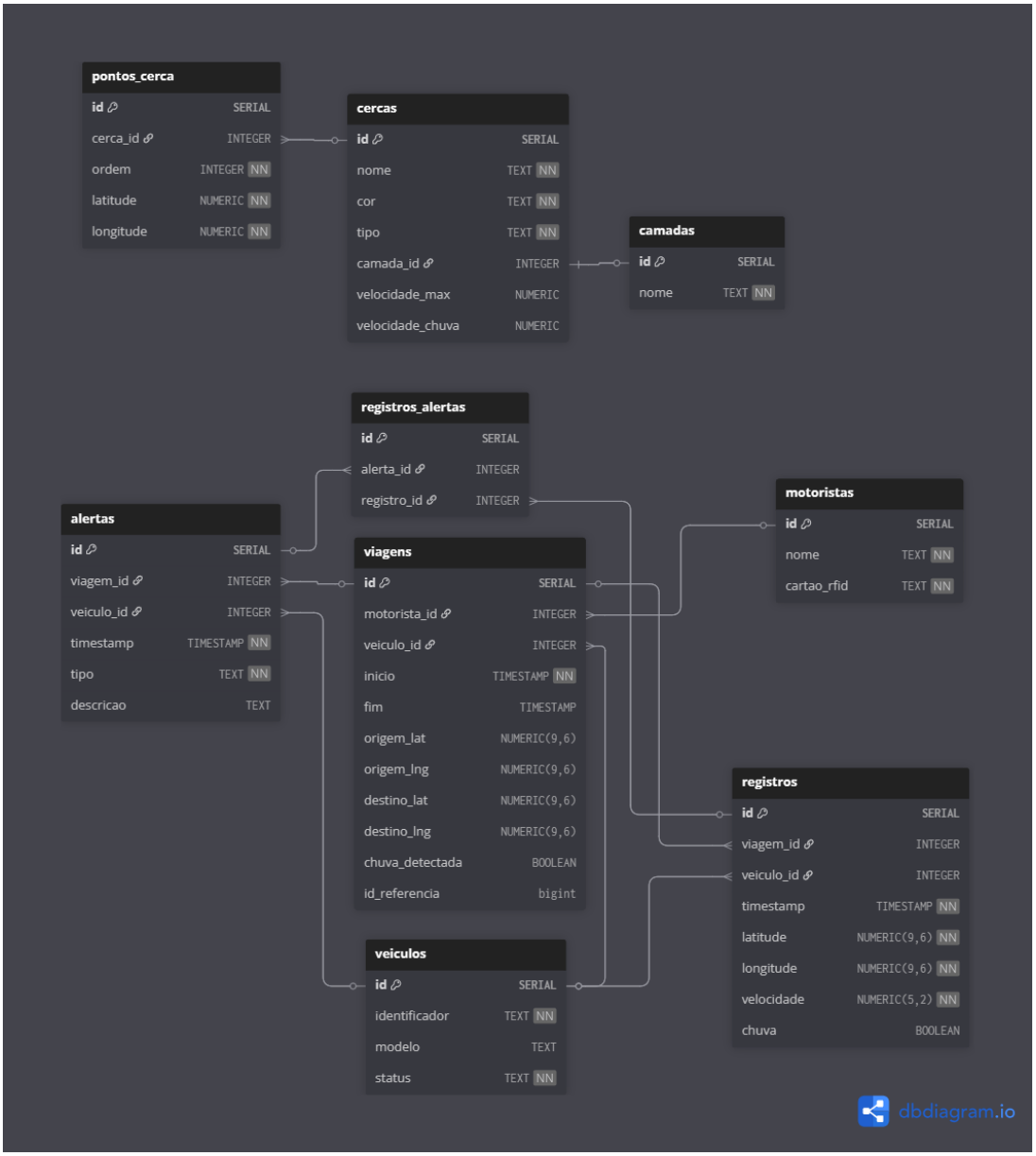


Figura 4.7 – Diagrama entidade-relacionamento do banco de dados. (Elaborado em *dbdiagram.io*, 2025.)

4.3.2 API do sistema

A API do sistema foi desenvolvida em *Node.js* com o objetivo de permitir a comunicação entre o dispositivo embarcado, o banco de dados e a plataforma online. Sua estrutura segue o estilo *RESTful*, ainda que com algumas adaptações práticas feitas durante o desenvolvimento. A API implementa os métodos HTTP básicos (*GET*, *POST*, *PUT* e *DELETE*), mas, na versão atual, apenas os métodos *GET* e *POST* são utilizados ativamente.

O método *GET* é empregado tanto pela plataforma online quanto pelo dispositivo embarcado. Na plataforma, ele fornece dados como motoristas, veículos, viagens, cercas e registros de alertas. Já no dispositivo embarcado, o *GET* é utilizado para resgatar as cercas

geográficas cadastradas no sistema, empregadas na verificação dos limites de velocidade com base nas coordenadas atuais, e também para obter a lista de motoristas, permitindo o processo de autenticação via RFID. Foram criadas variações desse método para otimizar o desempenho e reduzir o volume de dados transmitidos, retornando apenas as informações essenciais para cada funcionalidade da interface e do dispositivo. Um exemplo dessa abordagem é o conjunto de rotas voltadas ao resgate de motoristas cadastrados no sistema: a rota `/motoristas`, que retorna informações completas, e a rota `/motoristas/limpo`, que fornece apenas os dados essenciais.

O método *POST*, por sua vez, é utilizado pelo dispositivo embarcado para o envio de registros coletados durante as viagens. Esse processo envolve uma lógica de verificação que determina se a viagem enviada já está cadastrada no sistema. Caso exista, os novos registros são vinculados à viagem correspondente; caso contrário, uma nova viagem é criada automaticamente (mais detalhes na subseção 4.2.6). Essa abordagem garante a integridade e continuidade dos dados, mesmo em situações em que o dispositivo opere de forma intermitente ou sem conexão estável com a internet.

Embora a API já suporte métodos adicionais (*PUT* e *DELETE*) para futuras expansões, como edição e exclusão de registros diretamente pela plataforma, essas funcionalidades ainda não foram implementadas na versão atual.

4.3.3 Plataforma de Monitoramento e Interface do Usuário

A plataforma online foi desenvolvida em *React*, com o objetivo de oferecer uma interface interativa e responsiva para visualização e gerenciamento dos dados coletados pelo sistema embarcado. Ela atua como o principal meio de interação do usuário com a solução, permitindo o acompanhamento em tempo real das viagens, veículos, motoristas, alertas e cercas geográficas cadastradas.

A plataforma foi projetada para garantir uma navegação intuitiva, priorizando a clareza das informações e o desempenho na renderização dos elementos geográficos sobre o mapa. A versão atual é composta pelas seguintes telas principais:

Início: exibe os veículos no mapa em tempo real. Essa tela contém apenas o mapa, incluindo as cercas geográficas e os percursos. Ao clicar em um veículo, o usuário visualiza um resumo de seus status e pode, por meio de um botão no *popup*, acessar diretamente a tela de registros do motorista correspondente. Há também um seletor que permite alterar o tipo de visualização do mapa (como relevo, satélite ou ruas). Essa funcionalidade está presente em todos os mapas da plataforma.

Cercas: apresenta o mapa com ferramentas de edição, permitindo criar, editar e excluir cercas existentes. Na lateral esquerda, há um painel que exibe as camadas e as cercas cadastradas no sistema, facilitando a organização e visualização sem depender exclusivamente do mapa.

Motoristas: reúne todos os motoristas cadastrados no sistema. Nessa tela, é possível pesquisar motoristas, visualizar sua última localização no mapa, acessar registros e alertas associados, e até reconstruir percursos previamente registrados.

Registros: exibe todos os registros de forma não agrupada, o que facilita a pesquisa detalhada por dados específicos, como modelo do veículo, nome do motorista, data e hora exatos ou intervalos de tempo. Cada registro pode ser reconstruído visualmente no mapa.

Alertas: segue a mesma lógica da tela de registros, mas é dedicada exclusivamente aos alertas. Além das funções de pesquisa e reconstrução no mapa, há uma funcionalidade adicional que exibe todos os alertas simultaneamente, fornecendo um feedback visual sobre as regiões com maior incidência de infrações.

As Figuras 4.8 e 4.9 apresentam diferentes telas da plataforma online de monitoramento desenvolvida em *React*.

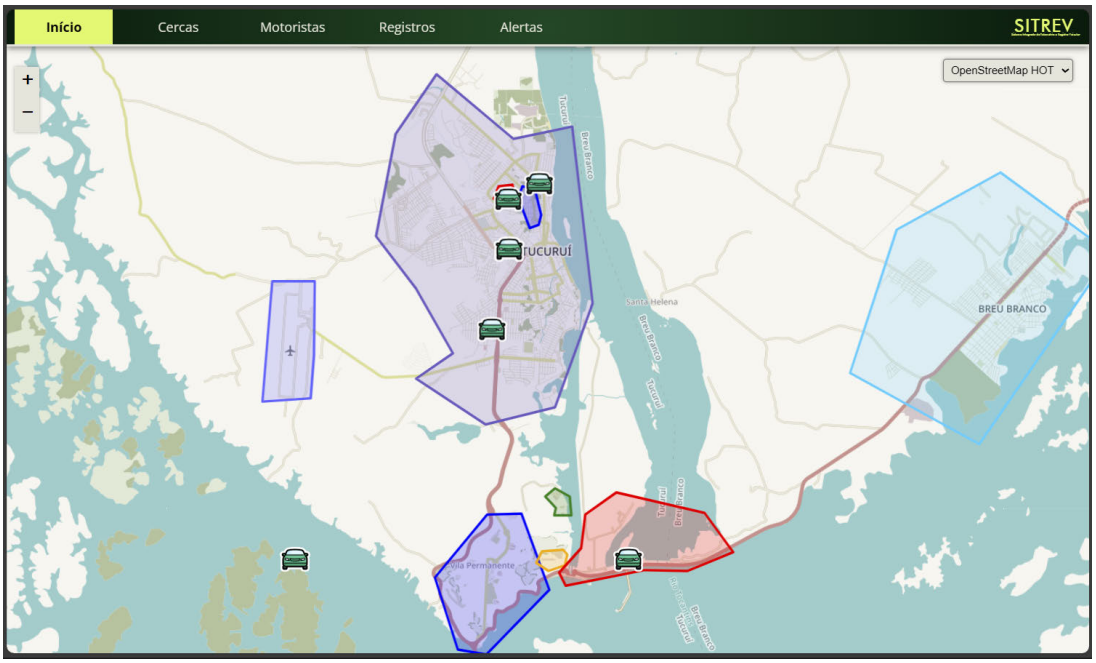
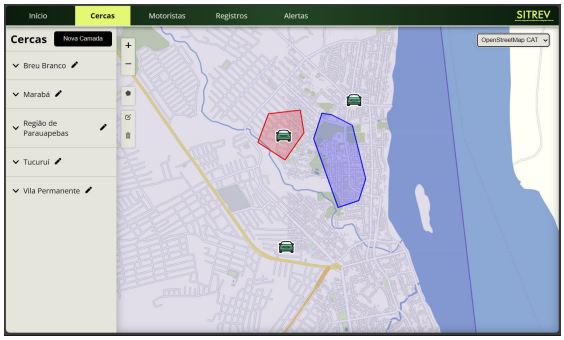


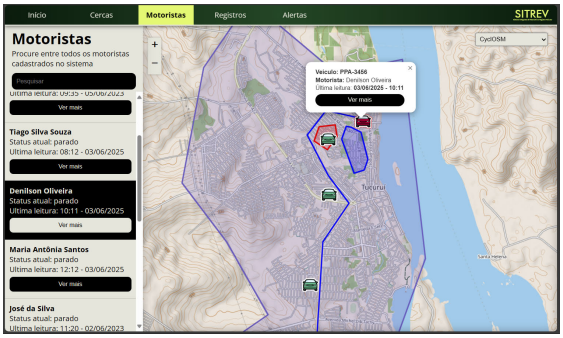
Figura 4.8 – Tela inicial da plataforma online de monitoramento desenvolvida em *React*.

Fonte: Autoria própria (2025).

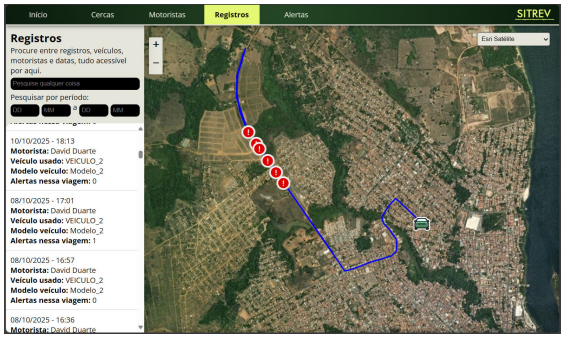
Figura 4.9 – Outras telas da plataforma online de monitoramento desenvolvida em *React*.



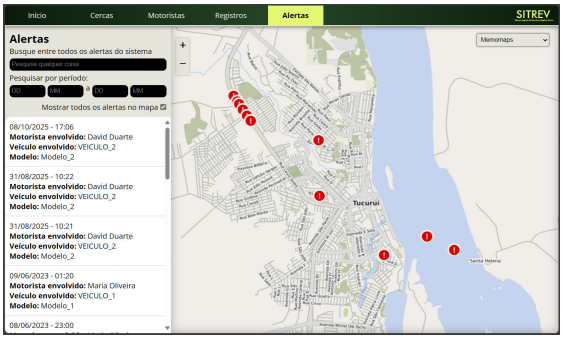
(a) Tela *Cercas*.



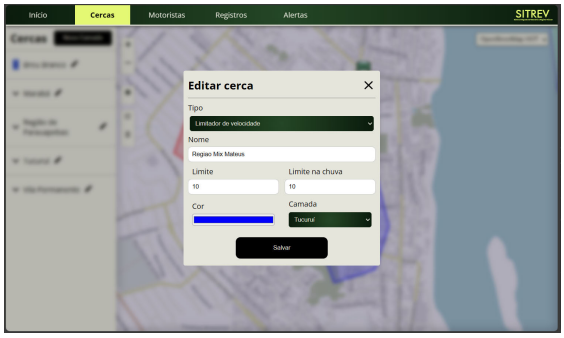
(b) Tela *Motoristas*.



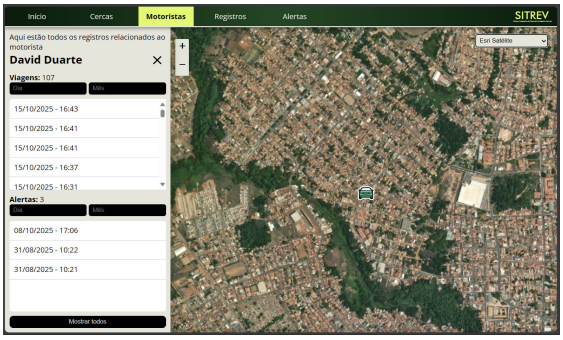
(c) Tela *Registros*.



(d) Tela *Alertas*.



(e) Tela de *Edição de Cerca*.



(f) Tela de *Registro Individual*.

Fonte: Autoria própria (2025).

4.4 Encapsulamento

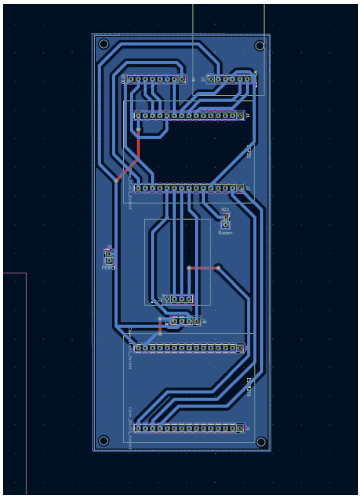
Esta seção descreve os aspectos relacionados ao encapsulamento do sistema desenvolvido, abordando tanto o processo de fabricação da placa de circuito impresso quanto as decisões técnicas adotadas para garantir modularidade, manutenção e futuras expansões do projeto.

4.4.1 Placa de Circuito Impresso

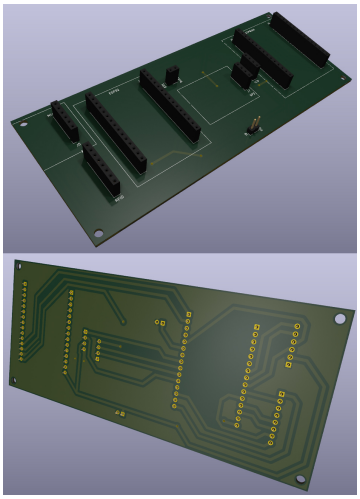
A placa de circuito impresso (PCB, do inglês *Printed Circuit Board*) foi fabricada a partir de uma placa folheada de cobre, utilizando o processo de corrosão química com percloreto de ferro. Após a fabricação, a placa apresentou dimensões aproximadas de 15 cm por 7 cm. O procedimento adotado seguiu as etapas descritas a seguir:

1. **Modelagem do circuito em software:** foi utilizado o software KiCad para a modelagem da placa, definindo as conexões entre os componentes, o roteamento das trilhas e o planejamento do layout final esperado.
2. **Impressão do circuito:** o esquema desenvolvido foi impresso utilizando uma impressora a laser. O objetivo dessa etapa foi possibilitar a transferência das partículas termoplásticas do toner para a placa de cobre, de modo que as trilhas ficassem protegidas durante o processo de corrosão química.
3. **Transferência de toner:** as partículas termoplásticas foram transferidas para a placa folheada de cobre com o auxílio de uma prensa térmica, originalmente destinada à estamparia de camisetas, garantindo boa aderência do toner à superfície metálica.
4. **Corrosão química:** utilizou-se percloreto de ferro para remover o cobre exposto entre as trilhas, preservando apenas as regiões protegidas pelo toner e resultando no layout definitivo da placa.
5. **Furação:** os furos necessários para a fixação e conexão dos componentes eletrônicos foram realizados com o auxílio de um perfurador específico para PCBs.
6. **Soldagem:** optou-se pela soldagem de barras de pinos fêmea na placa, permitindo o encaixe dos componentes sem soldagem direta. Essa decisão foi tomada com o objetivo de facilitar a reutilização dos componentes em versões futuras do projeto, além de possibilitar a montagem e desmontagem do dispositivo sempre que necessário.

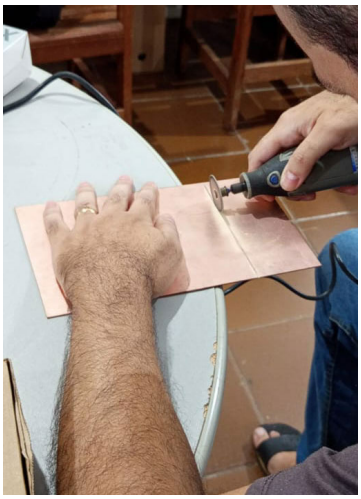
Com o objetivo de ilustrar as principais etapas do processo descrito, a Figura 4.10 apresenta registros visuais da fabricação da placa de circuito impresso, desde a modelagem até a soldagem dos componentes.



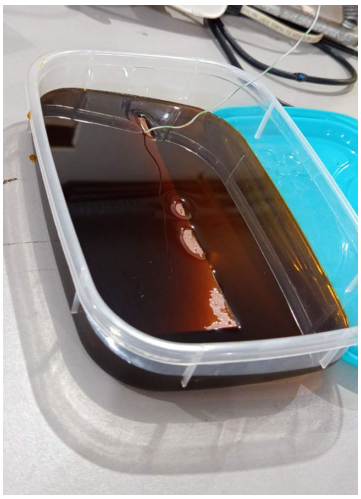
(a) Captura de tela da modelagem da PCB no software KiCad.



(b) Visualização tridimensional da PCB no ambiente de software.

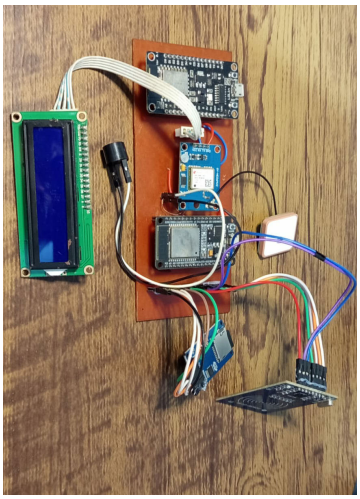


(c) Processo de corte da placa folheada em cobre.



(d) Processo de corrosão química do cobre excedente.

(e) PCB após o processo de corrosão.



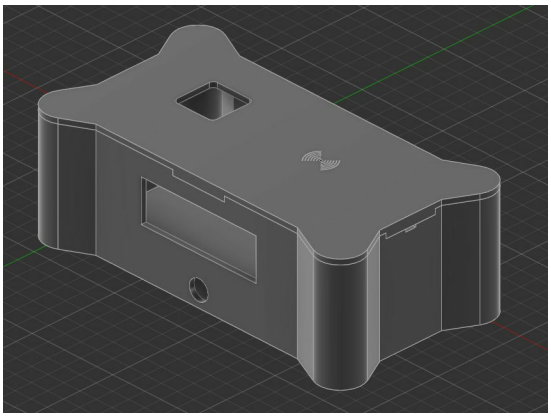
(f) Componentes montados após a soldagem das barras de pinos.

Figura 4.10 – Etapas do processo de fabricação da placa de circuito impresso.

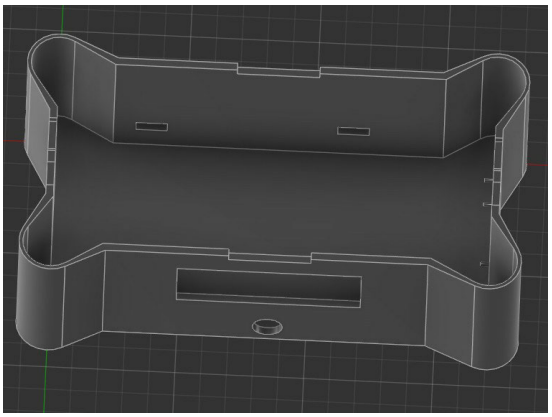
4.4.2 Caixa do Aparelho

Após a fabricação da placa de circuito impresso, a caixa do aparelho foi modelada tendo como base as dimensões e características físicas da PCB já produzida. O projeto da caixa foi desenvolvido no software Fusion 360, permitindo a definição precisa do encaixe dos componentes, pontos de fixação e aberturas necessárias para conexões.

A caixa foi posteriormente fabricada por meio de impressão 3D, resultando em um encapsulamento adequado para o dispositivo. A Figura 4.11 apresenta registros visuais da caixa finalizada, evidenciando o resultado do processo de modelagem e fabricação.



(a) Captura de tela do modelo da caixa no software Fusion 360.



(b) Outra captura do modelo da caixa no software.

(c) PCB encaixada na caixa já impressa.

(d) Protótipo montado e fechado na caixa.

Figura 4.11 – Caixa do aparelho desenvolvida para o encapsulamento do dispositivo.

5 RESULTADOS

Esta seção apresenta os principais resultados obtidos com o desenvolvimento e a integração do sistema de telemetria veicular proposto, abrangendo tanto o funcionamento do dispositivo embarcado quanto sua comunicação com a plataforma online. Os resultados demonstram o cumprimento dos objetivos definidos e a viabilidade prática da solução.

5.1 Funcionamento do Dispositivo Embarcado

O protótipo final foi montado utilizando o microcontrolador *ESP32*, integrando os módulos de GPS, leitor RFID, cartão SD, display LCD e buzzer para os sinais sonoros. Também foi utilizado um módulo *ESP8266* como unidade auxiliar, responsável pela conexão com a rede e pela comunicação com a API. Após a autenticação do motorista por meio do cartão RFID, o sistema inicia automaticamente o registro da viagem, coletando informações de localização e velocidade, além de comparar, durante todo o trajeto, a velocidade do veículo com o limite definido na cerca geográfica.

Todos os módulos foram testados individualmente antes e durante o desenvolvimento do protótipo. O processo de construção ocorreu de forma incremental, com a adição gradual de módulos e funcionalidades, o que facilitou o desenvolvimento e os testes de integração. Durante essas etapas, foi desenvolvida a lógica de atuação conjunta dos módulos e comprovada a compatibilidade entre todos eles em funcionamento simultâneo.

Ao finalizar cada versão do dispositivo, foram realizados testes progressivos, inicialmente sem o uso de um veículo, conforme a viabilidade de cada etapa. Os testes de limite de velocidade foram conduzidos com a adição provisória de um potenciômetro ao circuito, o qual, quando configurado adequadamente, pôde ser interpretado como a velocidade recebida em tempo real do veículo.

A conexão com o módulo ELM327, mesmo sem um veículo presente, foi testada por meio da alimentação do dispositivo em seus pinos específicos utilizando uma fonte de 12 V. Dessa forma, foi possível validar a comunicação via *Bluetooth*, ainda que sem o retorno de dados válidos, o que foi essencial para resolver problemas de emparelhamento enfrentados anteriormente.

Também foram realizados testes em um veículo parado, conectando o ELM327 à entrada OBD-II para resgatar os valores de velocidade e rotação (RPM) em tempo real, confirmando o funcionamento adequado da comunicação. Por fim, os testes de verificação das cercas geográficas em vigor foram realizados alterando as configurações diretamente na plataforma online, dispensando a necessidade de deslocamento físico para validar se a localização estava sendo detectada corretamente e se o limite correspondente era aplicado de forma precisa.

Os testes práticos foram realizado em um VW Polo 2024. O dispositivo foi instalado sobre o painel e alimentado por uma entrada USB do próprio veículo. O scanner OBD-II foi conectado à porta de diagnóstico e a conexão Wi-Fi foi estabelecida por meio do roteamento de internet móvel de um aparelho celular. As cercas geográficas já estavam previamente registradas no banco de dados.

Inicialmente, foram realizados testes de registro em condições normais, respeitando os limites de velocidade definidos na plataforma. Em seguida, foi testado o caso de ultrapassagem desses limites, simulando um cenário de alerta.

De modo geral, o dispositivo cumpriu seus objetivos, registrando com precisão cada ponto do percurso, desde a autenticação do motorista até a finalização do trajeto ao detectar a aproximação do cartão RFID de testes, cujo ID já estava cadastrado no banco de dados. A velocidade do veículo foi corretamente obtida do scanner ELM327 via Bluetooth Classic e registrada no percurso. O dispositivo identificou a cerca ativa na região e aplicou o limite de velocidade correspondente ao motorista.

5.2 Integração com a Plataforma Online

A plataforma online, desenvolvida em *React*, demonstrou pleno funcionamento na recepção e exibição dos dados transmitidos pelo dispositivo. Cada viagem registrada pôde ser visualizada no mapa com seu respectivo trajeto, motorista, velocidade e alertas de limite excedido.

Além disso, o sistema mostrou-se estável na reconstrução de percursos armazenados e na atualização dinâmica das informações em tempo real. O mecanismo de sincronização entre o dispositivo e a API garantiu a integridade dos dados mesmo em cenários de perda temporária de conexão.

Por fim, a funcionalidade de edição de cercas também se mostrou eficaz, apresentando efeitos imediatos nas alterações aplicadas e refletindo corretamente as atualizações tanto no banco de dados quanto nos dados armazenados no cartão SD do dispositivo embarcado.

A Figura 5.1 apresenta um dos percursos registrados durante os testes práticos com o veículo em movimento, evidenciando o pleno funcionamento de todos os componentes do sistema: o registro realizado pelo dispositivo, o envio e processamento pela API, o armazenamento no banco de dados e a exibição correta na plataforma online.

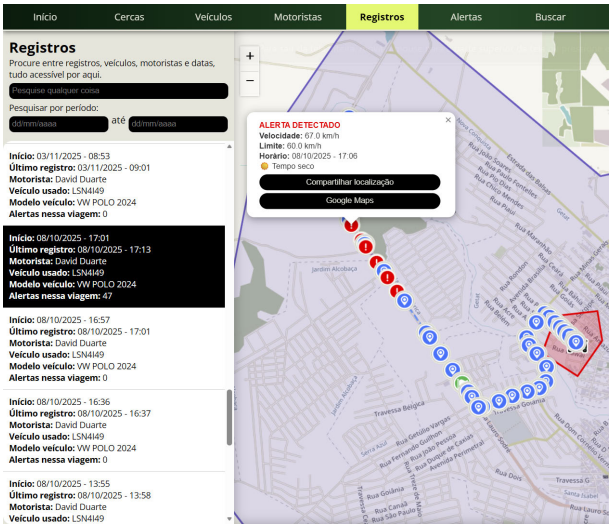


Figura 5.1 – Um dos percursos registrados durante os testes, exibido na plataforma.

Fonte: Autoria própria (2025).

5.3 Cenários de uso testados

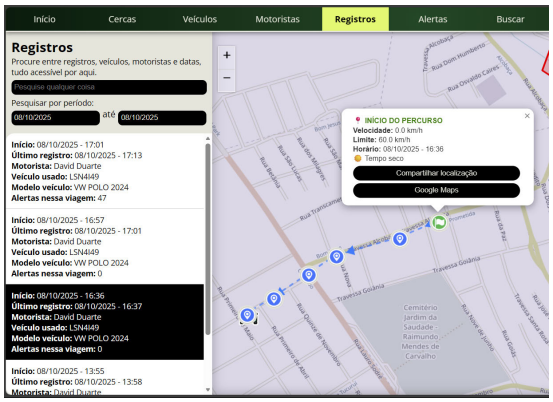
Os testes do sistema foram realizados em dois momentos distintos. O primeiro ocorreu ainda com o protótipo montado em protoboard, permitindo ajustes finais de hardware e software. O segundo foi conduzido após o encapsulamento, com os componentes já fixados na placa de circuito impresso e acomodados na caixa do dispositivo, simulando uma condição mais próxima da aplicação real.

Devido à limitação de conectividade com a internet, discutida anteriormente, o acesso à rede foi fornecido por meio de internet móvel compartilhada via roteamento Wi-Fi de um dispositivo celular.

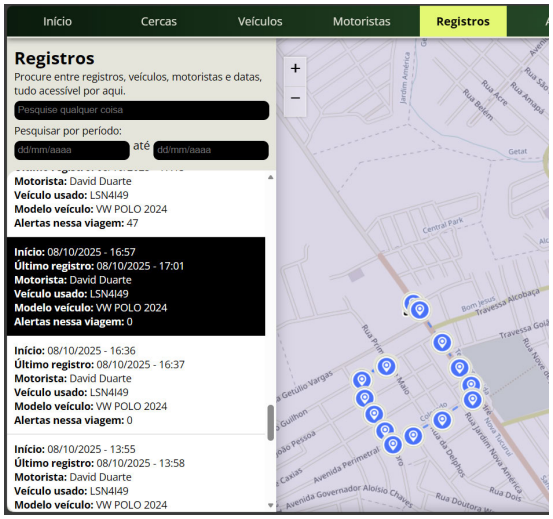
Durante os testes, o sistema registrou corretamente os percursos do veículo e transmitiu os dados à API, possibilitando sua visualização na plataforma desenvolvida. Também foram avaliados cenários de ultrapassagem de limite de velocidade definido por cercas geográficas, sendo possível identificar com precisão os pontos de infração, bem como as informações associadas, como data e horário.

Foram ainda testadas situações como falhas temporárias de conexão, indisponibilidade da API, uso do veículo sem autenticação prévia por RFID e desligamento do dispositivo antes do encerramento da viagem. Em todos os casos, o sistema apresentou o comportamento esperado, assegurando o armazenamento local dos dados e a sincronização automática quando a conexão foi restabelecida.

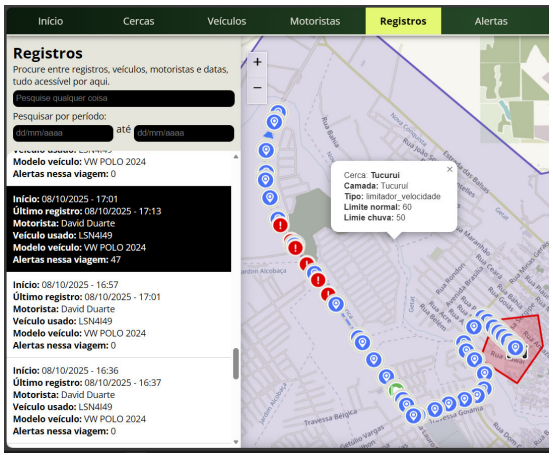
A seguir, são apresentadas capturas de tela de registros realizados durante esses testes, conforme visualizados na plataforma online.



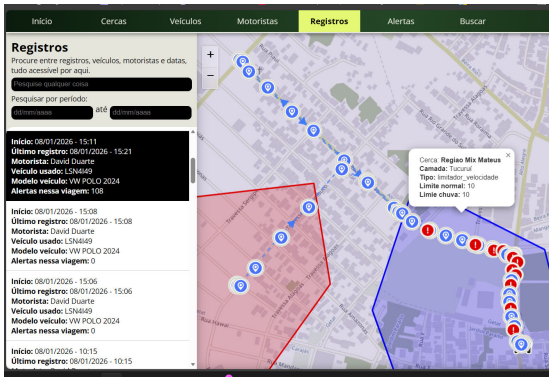
(a) Registro de percurso realizado em cenário real de uso.



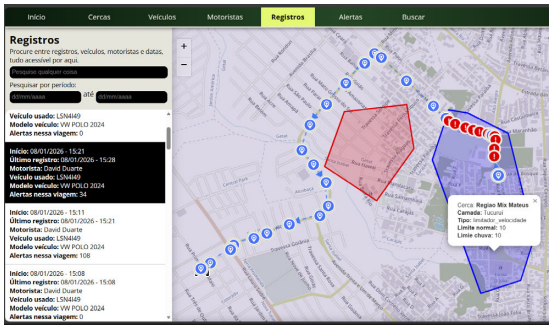
(b) Exemplo de registro inicial com percurso de curta duração.



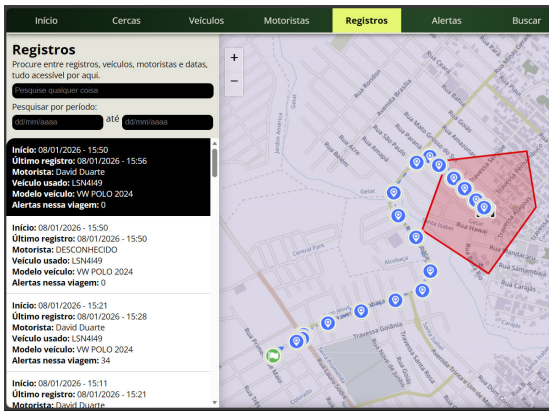
(c) Registro de percurso mais extenso com identificação de ultrapassagem do limite de velocidade.



(d) Registro coletado após o encapsulamento do dispositivo, com cerca geográfica configurada para limite de 10 km/h.



(e) Registro de percurso realizado após o encapsulamento do sistema.



(f) Registro final de percurso coletado durante os testes em condições reais.

Figura 5.2 – Exemplos de percursos de teste visualizados na plataforma online desenvolvida.

5.4 Custos de Produção

Os custos de produção do dispositivo foram organizados em três categorias distintas: a primeira refere-se à versão atual do protótipo desenvolvido, a segunda corresponde a uma versão futura planejada, com melhorias em termos de conectividade e autonomia, e a terceira diz respeito aos custos de funcionamento da plataforma online associada ao sistema, incluindo backend, frontend e banco de dados.

Ressalta-se que os valores apresentados são aproximados, uma vez que podem variar de acordo com o fornecedor, região e período de aquisição dos componentes. Para esta análise, foram considerados preços médios de mercado, evitando-se tanto os valores mais baixos quanto os mais elevados, com o objetivo de obter uma estimativa mais realista.

5.4.1 Versão Atual do Protótipo

A Tabela 5.1 apresenta os custos estimados dos principais componentes utilizados na versão atual do protótipo desenvolvido.

Tabela 5.1 – Custos aproximados dos componentes da versão atual do protótipo

Componente	**Valor aproximado (R$)**
ESP32	49,00
ESP8266	39,00
Módulo 4MD36 (adaptador para cartão SD)	28,00
Cartão SD 32 GB	30,00
Leitor RFID RC522	20,00
Display LCD 16×2 com módulo I2C	28,00
Módulo GPS NEO-6M v2	53,00
ELM327 (scanner OBD2 via Bluetooth)	30,00
Impressão 3D da caixa do protótipo	50,00
Placa de fenolite cobreada (PCB)	30,00
Total aproximado	**357,00**

Não foram incluídos nesta estimativa os custos de componentes auxiliares, como cabos de conexão, barras de pinos e materiais consumíveis de soldagem. Além disso, esta versão do dispositivo depende de uma conexão Wi-Fi para seu funcionamento, o que implica na necessidade de disponibilidade desse recurso no veículo onde o sistema é instalado.

5.4.2 Segunda Versão Planejada

Em função das limitações relacionadas à dependência de conexão Wi-Fi, foi planejada, embora não implementada, uma segunda versão hipotética do protótipo. Essa versão propõe poucas modificações estruturais, com o objetivo principal de permitir o funcionamento autônomo do dispositivo, além de incorporar pequenas melhorias em relação à versão atual.

A principal alteração consiste na utilização de um módulo ESP32 integrado a um módulo de comunicação móvel 4G A7670, que também oferece suporte a GPS, reduzindo a necessidade de módulos externos. A Tabela 5.2 apresenta os custos estimados dessa versão.

Tabela 5.2 – Custos aproximados dos componentes da versão planejada do protótipo

Componente	**Valor aproximado (R$)**
ESP32 LILYGO com A7670 4G e GPS	300,00
Módulo 4MD36 (adaptador para cartão SD)	28,00
Cartão SD 32 GB	30,00
Leitor RFID RC522	20,00
Display OLED	30,00
ELM327 (scanner OBD2 via Bluetooth)	30,00
Impressão 3D da caixa do protótipo	50,00
Placa de fenolite cobreada (PCB)	30,00
Total aproximado	**518,00**

Assim como na versão atual, não foram considerados os custos de componentes auxiliares. Ressalta-se que os valores estimados para esta nova versão podem variar em uma implementação real, visto que o módulo ESP32 integrado ao A7670 ainda não foi testado no projeto. Além disso, pode haver a necessidade de um microcontrolador auxiliar adicional, o que impactaria diretamente o custo final. As melhorias previstas estão descritas no Capítulo 6.

5.4.3 Hospedagem

Os custos de hospedagem da infraestrutura de software não podem ser definidos com exatidão, pois dependem de variáveis como volume de dados armazenados, número de requisições, processamento, largura de banda e transferência de dados. Esses fatores envolvem tanto custos fixos quanto variáveis.

Para esta estimativa, foram considerados valores praticados por plataformas amplamente utilizadas, como o Render (hospedagem da API) e o Neon.tech (banco de dados relacional), que são as mesmas empregadas na versão atual do projeto, ainda que em planos gratuitos. Os valores apresentados consideram planos pagos, uma vez que as versões gratuitas possuem limitações que não atendem a um cenário de produção contínua.

A Tabela 5.3 apresenta a estimativa mensal de hospedagem.

Tabela 5.3 – Estimativa de custos mensais de hospedagem da infraestrutura de software

Serviço	**Valor (US$)**	**Valor aproximado (R$)**
Hospedagem da API	19,00 + variáveis	100,00 + custos de computação
Hospedagem do banco de dados	10,00 a 30,00	50,00 a 160,00

6 POSSÍVEIS MELHORIAS E TRABALHOS FUTUROS

Apesar do resultado satisfatório obtido com o desenvolvimento deste protótipo, ele ainda não está pronto para uma aplicação imediata. Diversas melhorias podem ser implementadas no projeto como um todo para viabilizar seu uso em um cenário real, tanto no software, que abrange a plataforma online, a API e o banco de dados, quanto no hardware.

Substituição dos microcontroladores. A mais evidente das melhorias é a substituição dos microcontroladores utilizados. O conjunto formado pelo ESP32 e ESP8266 apresentou bom desempenho, porém ainda impõe limitações que não puderam ser completamente superadas. A adoção de um Raspberry Pi poderia não apenas solucionar as restrições de memória e processamento enfrentadas, mas também simplificar a lógica do código, reduzindo a necessidade de otimizações rigorosas de desempenho.

Além disso, o uso do Raspberry Pi tornaria o dispositivo mais seguro contra perdas de dados e mais eficiente na sincronização com o banco de dados, já que todas as funções seriam centralizadas em um único controlador. Suas capacidades superiores também abririam espaço para a implementação de lógicas mais robustas e novas funcionalidades, como a detecção de fadiga do motorista por meio de algoritmos de visão computacional, pensada para uma versão futura e mais avançada do projeto.

Detecção de chuva. Outra funcionalidade pendente que se mostrou relevante é a detecção de chuva. Sua implementação não foi possível na versão atual devido à limitação de tempo de desenvolvimento, uma vez que essa funcionalidade exigiria uma pesquisa mais aprofundada e uma série de testes experimentais. Assim, optou-se por adiar sua inclusão para versões futuras do sistema.

Entretanto, algumas abordagens viáveis foram consideradas, como a detecção do acionamento do limpador de para-brisa por meio de sensores ópticos, como sensores infravermelhos ou de distância a laser, a leitura direta do sinal elétrico proveniente da rede do veículo abaixo do painel, o uso de sensores de contato para detecção de água, e até a aplicação de algoritmos de visão computacional capazes de identificar gotas de chuva no para-brisa.

Conectividade com rede móvel. Uma melhoria extremamente importante que deve ser implementada antes do uso em cenários reais é a conexão com a rede móvel. A versão atual utiliza conexão Wi-Fi para acessar a internet, o que funciona bem em testes controlados, mas se torna inviável em movimento, já que o veículo não poderia fornecer uma conexão Wi-Fi própria.

Foram analisados módulos de comunicação móvel baseados em chips SIM convencionais, porém não foi possível implementá-los nesta versão devido a limitações orçamentárias. Os únicos módulos acessíveis no mercado nacional operam em redes 2G, que já se encontram

obsoletas. Os módulos ideais, com suporte a 4G ou superior, estão disponíveis apenas por importação, o que eleva significativamente o custo devido às taxas alfandegárias, tornando inviável sua aquisição neste primeiro desenvolvimento.

Alimentação do circuito. A alimentação do circuito também requer melhorias. O ideal seria que o motorista não pudesse interferir no funcionamento do dispositivo, mas a versão atual é alimentada via cabo USB, o que permitiria que o motorista simplesmente desligasse o aparelho, interrompendo o rastreamento. Já existem propostas para corrigir esse problema, como ligar o dispositivo diretamente à bateria do veículo ou, em uma versão futura, alimentá-lo pela porta OBD-II. Nesse último caso, o scanner ELM327 sem fio seria substituído por uma versão com fio, permitindo aproveitar a alimentação de 12 V da própria entrada OBD.

Comunicação com o barramento CAN (ELM327). Em relação ao ELM327, foi identificada uma possível instabilidade na transmissão dos dados do barramento CAN. A conexão Bluetooth pode gerar ruídos nos dados recebidos, dificultando a interpretação correta pelo ESP32. A substituição por uma versão com fio poderia eliminar completamente essa instabilidade, especialmente se o dispositivo passar a ser alimentado pela OBD-II. Outra alternativa é o uso de um modelo via Wi-Fi, que oferece uma comunicação mais estável; contudo, esse método não seria viável na versão atual, pois o ESP32 teria de se conectar simultaneamente à rede do scanner e à rede Wi-Fi da API, o que causaria conflitos. Em contrapartida, na futura versão baseada no Raspberry Pi 5 com conexão móvel, essa abordagem se tornaria perfeitamente viável.

Encapsulamento e qualidade construtiva. No que diz respeito ao encapsulamento do projeto, uma melhoria considerada essencial é a produção de uma placa de circuito impresso (PCB). A versão atual do dispositivo é funcional, porém ainda distante de um padrão profissional de construção. Atualmente, existem diversos serviços especializados que permitem a fabricação de PCBs em pequenos lotes, geralmente a partir de 5 ou 10 unidades, possibilitando a evolução do protótipo sem grandes custos iniciais. Para versões futuras do sistema, torna-se necessária uma maior qualidade construtiva do dispositivo, sendo a encomenda de PCBs uma etapa recomendada para garantir melhor organização dos componentes, redução de falhas mecânicas e maior confiabilidade do conjunto.

Em relação à integração dos componentes na PCB, é necessário um maior rigor técnico na montagem. Recomenda-se a substituição de barras de pinos por soldagem direta dos componentes à placa, sempre que possível, bem como a substituição de jumpers por cabos mais resistentes, como conectores do tipo JST plug. Essas melhorias contribuem significativamente para a durabilidade do protótipo, especialmente considerando que o dispositivo seria fixado no painel de um veículo, estando sujeito a vibrações e variações de temperatura.

Melhorias na arquitetura de software. Em relação à parte de software, é necessária uma revisão geral na estrutura do banco de dados e na arquitetura da API. Durante o

desenvolvimento, tornaram-se evidentes alguns problemas decorrentes do planejamento inicial, exigindo ajustes no processo. Acredita-se que, com uma reestruturação completa, outros pontos de melhoria possam ser identificados, resultando em um backend mais otimizado e estável.

Expansões do front-end. No que diz respeito ao front-end, diversas melhorias também podem ser implementadas. A mais evidente é a inclusão de um sistema de autenticação, uma vez que a versão atual não diferencia usuários nem gerencia configurações específicas para cada um. O front-end foi desenvolvido com foco em demonstrar como os dados poderiam ser exibidos e manipulados, evidenciando a eficiência da comunicação com o dispositivo embarcado.

Além disso, há potencial para expansão das funcionalidades da plataforma, como geração de relatórios, exportação e importação de configurações, criação de uma seção exclusiva de registros por veículo e consulta de registros por localização, entre outras melhorias.

7 CONCLUSÃO

O desenvolvimento deste trabalho possibilitou a criação de um protótipo funcional de um sistema embarcado voltado à coleta e transmissão de dados veiculares, integrado a uma plataforma online para visualização e gerenciamento das informações. Ao longo do projeto, foram abordados aspectos tanto de hardware quanto de software, contemplando desde a construção física do dispositivo até o desenvolvimento da API, do banco de dados e da interface web.

Os objetivos propostos foram alcançados, uma vez que o sistema desenvolvido foi capaz de coletar dados do veículo, processá-los de forma local e enviá-los para uma plataforma remota, demonstrando a viabilidade da arquitetura adotada. A integração entre o dispositivo embarcado e o backend permitiu validar a comunicação entre os diferentes componentes do sistema, bem como a organização e o armazenamento dos dados em ambiente de nuvem.

Durante o desenvolvimento, algumas limitações técnicas e práticas foram identificadas, especialmente relacionadas à conectividade, à capacidade de processamento dos microcontroladores utilizados e à qualidade construtiva do protótipo. Essas limitações, no entanto, não comprometem o objetivo principal do trabalho, mas evidenciam pontos importantes a serem aprimorados em versões futuras do sistema.

Dessa forma, o projeto apresentado serve como uma base sólida para evoluções posteriores, tanto no aprimoramento do hardware quanto na expansão das funcionalidades do software. As melhorias e trabalhos futuros discutidos ao longo do trabalho indicam caminhos viáveis para tornar o sistema mais robusto, confiável e adequado para aplicações em cenários reais, reforçando o potencial do projeto como uma solução de monitoramento veicular integrada.

REFERÊNCIAS

Amazon Web Services. **Qual é a diferença entre HTTP e HTTPS?** 2023. Acesso em: fev. 2026. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-https-and-http/>. Citado na página 19.

Amazon Web Services. **O que é a Internet das Coisas (IoT)? [What is the Internet of Things (IoT)?]**. 2025. Disponível em: <https://aws.amazon.com/pt/what-is/iot/>. Citado na página 18.

Cloudian. **IoT Storage: Approaches, Technologies, and Key Challenges**. 2023. Acesso em: fev. 2026. Disponível em: <https://cloudian.com/guides/ai-infrastructure/iot-storage-approaches-technologies-and-4-key-challenges/>. Citado na página 20.

CUGNASCA, C. E. **Aprofundando os conceitos de sistemas embarcados – Parte 1**. 2020. Disponível em: <https://integra.univesp.br/courses/2710/pages/texto-base-aprofundando-os-conceitos-de-sistemas-embarcados-parte-1-%7C-carlos-eduardo-cugnasca>. Citado na página 18.

ELM Electronics. **ELM327 OBD-II Interpreter Datasheet**. [S.l.], 2010. Documento obtido via AllDataSheet. Acesso em: 09 jan. 2026. Disponível em: <https://www.alldatasheet.com/datasheet-pdf/view/197403/ELM/ELM327.html>. Citado na página 15.

EONE Electronics. **EONE-1602A1 LCD Module Datasheet**. [S.l.], 2018. Acesso em: 10 fev. 2026. Disponível em: <https://blog.eletrogate.com/wp-content/uploads/2018/04/EONE-1602a1.pdf>. Citado 2 vezes nas páginas 15 e 20.

Espressif Systems. **ESP8266EX Datasheet**. [S.l.], 2020. Acessado em: 12 out. 2025. Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Citado 2 vezes nas páginas 15 e 19.

Espressif Systems. **ESP32 Series Datasheet**. [S.l.], 2023. Acesso em: 12 out. 2025. Disponível em: <https://documentation.espressif.com/esp32_datasheet_en.pdf>. Citado 3 vezes nas páginas 14, 15 e 19.

Espressif Systems. **ESP32 External RAM (PSRAM) API Guide**. [S.l.], 2025. Acesso em: 13 fev. 2026. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-guides/external-ram.html>. Citado na página 32.

Geotab Inc. **Cercas virtuais na gestão de frotas**. 2024. <https://www.geotab.com/pt-br/blog/cercas-virtuais-gestao-de-frotas/>. Acesso em: 12 fev. 2026. Citado na página 23.

GOMES, M. G. **Telemetria Veicular**. Ouro Preto: [s.n.], 2022. Monografia (Graduação em Engenharia de Controle e Automação). Disponível em: <http://www.monografias.ufop.br/handle/35400000/4331>. Citado na página 14.

IBM. **O que são APIs REST**. 2024. Disponível em: <https://www.ibm.com/br-pt/think/topics/rest-apis>. Citado na página 21.

Leaflet Contributors. **Leaflet — JavaScript Library for Interactive Maps**. 2026. <https://leafletjs.com/reference.html>. Acesso em: 12 fev. 2026. Citado na página 22.

Lumini Tracker. **Desligamento das redes 2G e 3G no Brasil: panorama, etapas e impactos**. 2025. Disponível em: <https://www.luminitracker.com.br/post/desligamento-das-redes-2g-e-3g-no-brasil-panorama-etapas-e-impactos>. Citado na página 28.

Meta Platforms, Inc. **React Documentation: API Reference**. 2025. Disponível em: <https://react.dev/reference/react>. Citado 2 vezes nas páginas 15 e 22.

MICRO SD Card Module Datasheet. [S.l.], 2019. Documento técnico de módulo genérico. Acesso em: 10 fev. 2026. Disponível em: <https://cdn.awsli.com.br/945/945993/arquivos/Datasheet-MicroSD-Module.pdf>. Citado 2 vezes nas páginas 15 e 20.

Murata Manufacturing Co., Ltd. **Piezoelectric Sound Component PKM22EPPH4001-BO Datasheet**. [S.l.], 2016. Acesso em: 13 fev. 2026. Disponível em: <https://www.arduino.cc/documents/datasheets/PIEZO-PKM22EPPH4001-BO.pdf>. Citado na página 15.

NEON. **Neon.tech: PostgreSQL Serverless Database Hosting**. 2025. Acesso em: 17 out. 2025. Disponível em: <https://neon.tech>. Citado na página 40.

Node.js Foundation. **Node.js Documentation: API Reference**. 2025. Disponível em: <https://nodejs.org/docs/latest/api/>. Citado 2 vezes nas páginas 15 e 21.

npm, Inc. **npm Documentation**. 2025. Disponível em: <https://docs.npmjs.com/>. Citado na página 21.

NXP Semiconductors. **MFRC522 Datasheet**. [S.l.], 2012. Acesso em: 09 jan. 2026. Disponível em: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>. Citado 2 vezes nas páginas 15 e 20.

NXP Semiconductors. **PCF8574 — Remote 8-bit I/O Expander for I2C-bus**. [S.l.], 2015. Acesso em: 10 fev. 2026. Disponível em: <https://blog.eletrogate.com/wp-content/uploads/2018/04/PCF8574-NXP.pdf>. Citado 2 vezes nas páginas 15 e 20.

PostgreSQL Global Development Group. **PostgreSQL 18 Documentation**. [S.l.], 2024. Acesso em: 12 fev. 2026. Disponível em: <https://www.postgresql.org/files/documentation/pdf/18/postgresql-18-A4.pdf>. Citado na página 21.

SILVA, G. M. H.; SANTOS, R. A. O. Telemetria como ferramenta iot na gestão da segurança de frotas veiculares. **Revista Aracê**, São José dos Pinhais, v. 6, n. 3, p. 7692–7709, 2024. Disponível em: <https://doi.org/10.56238/arev6n3-201>. Citado 2 vezes nas páginas 21 e 23.

SIMCom Wireless Solutions. **SIM800L Quad-Band GSM/GPRS Module Datasheet**. [S.l.], 2015. Acesso em: 14 fev. 2026. Disponível em: <https://www.makerhero.com/img/files/download/Datasheet_SIM800L.pdf>. Citado na página 28.

SIMCom Wireless Solutions Co., Ltd. **SIM7600CE_SIM7600C_Hardware Design**. [S.l.], 2016. Versão 1.01, 27 de julho de 2016. Disponível em: <https://www.rhydolabz.com/documents/30/Hardware-Design.pdf>. Acesso em: 14 fev. 2026. Citado na página 27.

u-blox AG. **NEO-6 GPS Module Datasheet**. [S.l.], 2010. Acesso em: 09 jan. 2025. Disponível em: <https://content.u-blox.com/sites/default/files/products/documents/NEO-6_DataSheet_%28GPS.G6-HW-09005%29.pdf>. Citado 2 vezes nas páginas 15 e 20.