Universidade Católica do Salvador

Bacharelado em Engenharia de Software

Sistema de Telemetria Veicular com ESP32, OBD-II/CAN, EMQX e Aplicação Web

Salvador

2026

Bryan Montgomery Hamilton
Clovis Suarez Dantas
Pedro Henrique Figueirêdo Vidigal
Pedro Vinícius Cunha dos Santos
Samuel Pereira dos Santos Santana

Sistema de Telemetria Veicular com ESP32, OBD-II/CAN, EMQX e Aplicação Web

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.
Orientador: Prof. Me. Marco Câmara

Universidade Católica do Salvador

Salvador
2026

Bryan Montgomery Hamilton

Clovis Suarez Dantas

Pedro Henrique Figueirêdo Vidigal

Pedro Vinícius Cunha dos Santos

Samuel Pereira dos Santos Santana

Sistema de Telemetria Veicular com ESP32, OBD-II/CAN, EMQX e Aplicação Web

Trabalho de Conclusão de Curso apresentado à Universidade Católica do Salvador como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Software.
Orientador: Prof. Me. Marco Câmara

Salvador, de junho de 2026.

Banca Examinadora:

Prof. Me. Marco Câmara
Universidade Católica do Salvador
Orientador

Prof. Me. Segundo professor
Universidade Católica do Salvador

Prof. Me. Terceiro professor
Universidade Católica do Salvador

Dedicamos este trabalho a todos que apoiaram a construção desta trajetória acadêmica e profissional, especialmente às nossas famílias, aos professores e aos colegas que contribuíram para a realização deste projeto.

AGRADECIMENTOS

Agradecemos a Deus, às nossas famílias, base formativa e apoio irrestrito à nossa trajetória de vida. Agradecemos também aos colegas e professores do Bacharelado em Engenharia de Software da Universidade Católica do Salvador, cujo apoio foi essencial para a consolidação deste protótipo.

“Você é meu emissário escolhido,
pois, em você, depositei toda a minha confiança.”
N. Sra. de Guadalupe

RESUMO

Este trabalho apresenta o desenvolvimento de um protótipo de baixo custo de telemetria veicular, voltado ao monitoramento básico de informações de um automóvel. O objetivo geral consiste em desenvolver uma solução capaz de coletar, transmitir, armazenar e exibir dados básicos de telemetria, permitindo acompanhamento remoto e consulta posterior dessas informações acessível por uma página Web. A proposta destina-se a estudantes, pesquisadores e interessados em monitoramento veicular e registro histórico de dados. Como aplicação prática, o sistema possibilita observar parâmetros básicos de funcionamento do veículo, manter histórico de uso e identificar condições relevantes de operação. Os resultados obtidos deste trabalho indicam a viabilidade técnica da solução desenvolvida, ao demonstrar a integração funcional entre módulo embarcado, comunicação de dados, processamento em servidor, armazenamento em banco de dados e interface web no contexto de telemetria veicular.

Palavras-chave: Telemetria veicular; monitoramento veicular; ESP32; OBD-II/CAN; sistema embarcado.

ABSTRACT

This work presents the development of a low-cost vehicle telemetry prototype, aimed at basic monitoring of vehicle information. The overall objective is to develop a solution capable of collecting, transmitting, storing, and displaying basic telemetry data, allowing remote monitoring and subsequent consultation of this information accessible through a web page. The proposal is intended for students, researchers, and those interested in vehicle monitoring and historical data recording. As a practical application, the system makes it possible to observe basic vehicle operating parameters, maintain usage history, and identify relevant operating conditions. The results obtained from this work indicate the technical feasibility of the developed solution, demonstrating the functional integration between the embedded module, data communication, server processing, database storage, and web interface in the context of vehicle telemetry.

Keywords: Vehicle telemetry; vehicle monitoring; ESP32; OBD-II/CAN; embedded system.

LISTA DE ILUSTRAÇÕES

Isto não faz mais sentido...

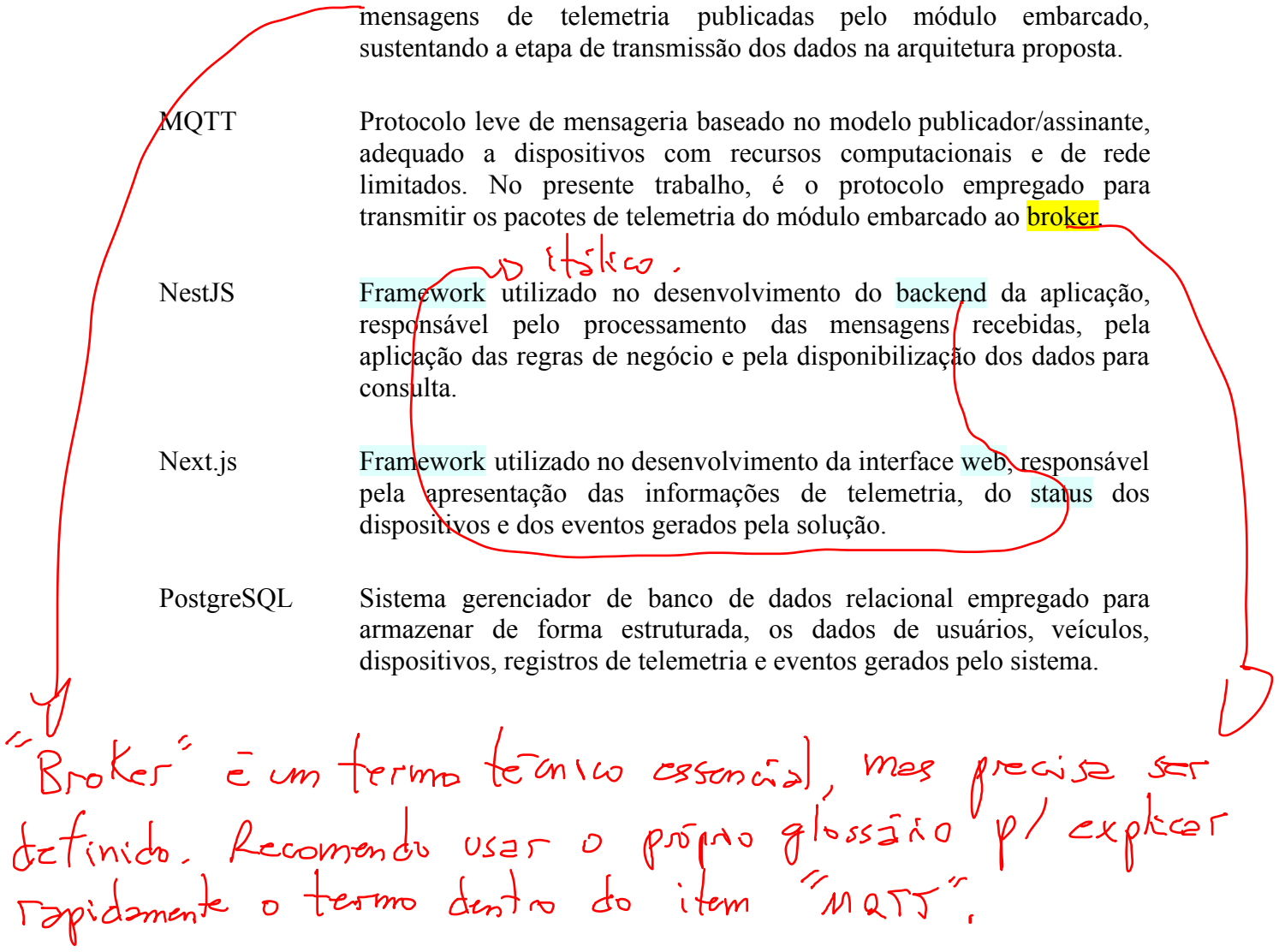
QUADROS

FIGURA

IMAGENS

GLOSSÁRIO DE TECNOLOGIAS

ESP32	Microcontrolador utilizado como núcleo do módulo embarcado do sistema. Sua adoção justifica-se pelo baixo custo, pela conectividade integrada e pela capacidade suficiente para realizar a leitura, a organização e o envio periódico dos dados de telemetria.
OBD-II/CAN	Conjunto de padrões e meios de comunicação utilizados para acesso a dados automotivos. Neste trabalho, esses conceitos fundamentam a obtenção dos parâmetros veiculares, representados no ambiente experimental por meio de simulador OBD/CAN, em conformidade com a lógica de aquisição prevista para o domínio automotivo.
EMQX	Broker MQTT utilizado no protótipo para receber e encaminhar as mensagens de telemetria publicadas pelo módulo embarcado, sustentando a etapa de transmissão dos dados na arquitetura proposta.
MQTT	Protocolo leve de mensageria baseado no modelo publicador/assinante, adequado a dispositivos com recursos computacionais e de rede limitados. No presente trabalho, é o protocolo empregado para transmitir os pacotes de telemetria do módulo embarcado ao broker.
NestJS	Framework utilizado no desenvolvimento do backend da aplicação, responsável pelo processamento das mensagens recebidas, pela aplicação das regras de negócio e pela disponibilização dos dados para consulta.
Next.js	Framework utilizado no desenvolvimento da interface web, responsável pela apresentação das informações de telemetria, do status dos dispositivos e dos eventos gerados pela solução.
PostgreSQL	Sistema gerenciador de banco de dados relacional empregado para armazenar de forma estruturada, os dados de usuários, veículos, dispositivos, registros de telemetria e eventos gerados pelo sistema.



SUMÁRIO

1. INTRODUÇÃO

A telemetria veicular consiste na obtenção, transmissão, armazenamento e na análise de dados relacionados ao funcionamento e ao uso do veículo. Essas informações podem ser empregadas em atividades de monitoramento, diagnóstico, registro histórico e identificação de eventos relevantes de condução. Com a ampliação do uso de sistemas eletrônicos embarcados e de soluções conectadas no setor automotivo, o tratamento desses dados passou a assumir maior relevância técnica e acadêmica.

Apesar desse avanço, o acesso a dados veiculares ainda apresenta limitações em contextos de experimentação e desenvolvimento acadêmico, seja pelo custo de soluções comerciais, seja pela baixa flexibilidade de plataformas proprietárias para fins didáticos. Nesse contexto, estabelece-se como problema de pesquisa a seguinte questão: como desenvolver um protótipo acadêmico de telemetria veicular capaz de coletar, transmitir, armazenar e disponibilizar dados básicos do veículo por meio de uma interface web?

Parte-se da hipótese de que é possível construir uma solução funcional de telemetria veicular a partir da integração entre hardware embarcado, comunicação com dados automotivos, mensageria, backend, banco de dados e interface web. Nessa perspectiva, a proposta assume características de uma caixa-preta veicular, uma vez que realiza o registro de informações e eventos para consulta e análise posteriores.

O objetivo geral deste trabalho é desenvolver um sistema de telemetria veicular capaz de integrar hardware embarcado, mensageria, backend, banco de dados e interface web, permitindo o acompanhamento de informações básicas do veículo. Para o alcance do objetivo, este estudo contempla a captura ou recepção de dados veiculares, o armazenamento local para contingência, a transmissão das informações, o processamento no backend, a armazenamento dos registros, a geração de eventos automáticos com base em regras de negócio e a disponibilização de uma aplicação web para consulta dos dados.

Este trabalho justifica-se, em primeiro lugar, por sua relevância prática, tendo em vista o potencial de aplicação de soluções de telemetria em monitoramento veicular, registro histórico e apoio à análise de ocorrências. Em segundo lugar, apresenta relevância acadêmica por articular conteúdos próprios da Engenharia de Software e da computação aplicada, tais

como análise de requisitos, modelagem de dados, integração entre sistemas, arquitetura de software, armazenamento de dados e desenvolvimento de interface web. Desse modo, o estudo contribui para materializar, em um protótipo funcional, conceitos que frequentemente são discutidos de forma isolada no percurso formativo.

Quanto ao escopo, a pesquisa limita-se ao desenvolvimento de um protótipo acadêmico e à validação funcional do fluxo de captura ou recepção, transmissão, processamento, armazenamento e visualização dos dados. Incluem-se nesse recorte o módulo embarcado, a comunicação com dados veiculares, o armazenamento local, a mensageria MQTT com broker EMQX, o backend, o banco de dados, a interface web e a geração de eventos automáticos. No cenário real previsto pela arquitetura, a obtenção dos dados ocorreria por OBD-II/CAN. No ambiente experimental adotado neste trabalho, entretanto, a validação foi realizada com simulador OBD/CAN, que representou a origem dos dados utilizados no fluxo testado.

Não fazem parte do escopo deste estudo a homologação automotiva, a certificação para uso comercial, a implantação em ambiente produtivo, os testes prolongados em frota real, a integração com aplicações móveis, o uso de geolocalização em tempo real, a aplicação de inteligência artificial e o desenvolvimento de produto final com encapsulamento industrial. Assim, a solução proposta deve ser compreendida como um protótipo funcional voltado à demonstração de viabilidade técnica em contexto acadêmico e experimental.

Portanto, o objeto de estudo concentra-se no sistema desenvolvido e na integração entre seus componentes, com ênfase na verificação de sua capacidade de realizar o ciclo de captura ou recepção, transmissão, processamento, armazenamento e visualização de dados veiculares.

2. FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica constitui o suporte conceitual do presente trabalho, ao reunir os referenciais necessários à compreensão do objeto de estudo e à sustentação das escolhas técnicas e metodológicas adotadas. Nessa perspectiva, esta seção não se limita à apresentação isolada de conceitos, mas busca estabelecer diálogo com a literatura especializada, de modo a evidenciar como os autores selecionados contribuem para a delimitação do problema de pesquisa, para a justificativa da proposta desenvolvida e para a interpretação dos resultados obtidos.

Assim, a revisão teórica foi estruturada a partir dos eixos temáticos centrais da pesquisa, compreendendo telemetria veicular, caixa-preta veicular, OBD-II, CAN Bus, sistemas embarcados, Internet das Coisas aplicada ao setor automotivo, mensageria, registro de dados, manutenção preditiva, validação de sistemas embarcados automotivos e metodologia científica.

acho que vale citar a sigla IoT, mais conhecida do que a tradução

2.1. Telemetria Veicular

A telemetria pode ser compreendida como o processo de coleta, transmissão, armazenamento e análise de dados obtidos a partir de um sistema monitorado. No contexto automotivo, esses dados podem incluir velocidade, rotação do motor, temperatura e outros parâmetros relacionados ao funcionamento do veículo e ao comportamento de condução.

Na literatura analisada, Silva (2025, p. 6) apresenta um sistema embarcado "modular e de baixo custo para aquisição, processamento e transmissão de dados veiculares", capaz de "coletar parâmetros do veículo, normalizá-los e publicá-los em redes locais ou remotas". Essa formulação é relevante porque amplia a noção de telemetria para além da simples leitura de informações, associando-a também ao tratamento e à disponibilização estruturada dos dados.

Em perspectiva mais ampla, Muthiya et al. (2023, p. 115) definem a Internet das Coisas como um método de obtenção de dados de dispositivos remotos para "automate and notify the user", evidenciando que a conectividade pode ser aplicada ao monitoramento e ao suporte ao usuário em diferentes contextos, inclusive no setor automotivo. No mesmo capítulo, os autores observam ainda que veículos equipados com IoT passam a "share important data" e a "monitor overall condition of the vehicle" (MUTHIYA et al., 2023, p.

Acho melhor traduzir.

117), o que reforça a ideia do automóvel como plataforma conectada de acompanhamento operacional.

De modo complementar, Ghosh et al. (2022, p. 328) destacam a necessidade de soluções "capable of managing, storing, processing, transmitting, receiving, and updating a large amount of data efficiently". Tal formulação reforça que a telemetria veicular, em ambientes conectados, envolve um ciclo completo de tratamento da informação, e não apenas sua captura.

Essa compreensão é central para o presente trabalho, uma vez que a proposta desenvolvida não se limita à leitura pontual de dados do veículo. No protótipo, as informações são obtidas, transmitidas, armazenadas e posteriormente disponibilizadas em interface web para consulta. Assim, a telemetria veicular, neste estudo, é entendida como um processo integrado de aquisição, comunicação, registro e uso da informação, em consonância com a literatura analisada.

2.2. Caixa Preta Veicular

A noção de caixa-preta veicular refere-se ao registro sistemático de informações relevantes sobre o funcionamento e o uso do veículo, com finalidade de consulta, rastreabilidade e interpretação posterior. Em aplicações automotivas, esse tipo de solução está associado à preservação de dados que possam subsidiar análise técnica, investigação de ocorrências e reconstrução de eventos.

Veitas e Delaere (2018) ampliam esse conceito ao deslocar a discussão da simples coleta para o problema mais abrangente do registro, do armazenamento e do gerenciamento de acesso aos dados veiculares. No resumo do trabalho, os autores definem o EDR/AD como um subsistema capaz de assegurar "the confidentiality, integrity and availability of data related to operation of a vehicle" (VEITAS; DELAERE, 2018, p. 1). Essa formulação é importante porque mostra que a ideia de caixa-preta não se limita ao ato de registrar dados, mas envolve também sua preservação, organização e acesso controlado.

Em linha mais aplicada, Vanitha et al. (2023, p. 2) afirmam que, após a coleta, "the second step is to save the data to an SD card module; and the third step is to upload the data to the cloud". Tal proposta articula armazenamento local e envio remoto das informações,

aproximando a caixa-preta veicular de arquiteturas conectadas e distribuídas.

A contribuição desses autores é central para o presente trabalho porque desloca o foco da simples captura do dado para o problema mais amplo de registro, armazenamento e recuperação da informação. Essa perspectiva se relaciona diretamente com o protótipo desenvolvido, no qual os dados não são apenas obtidos, mas também armazenados localmente em cartão SD, transmitidos por rede e persistidos em banco de dados para consulta posterior. Assim, a ideia de caixa-preta adotada neste TCC corresponde a um registrador acadêmico de telemetria, voltado ao monitoramento e à preservação mínima do histórico operacional do veículo.

2.3. OBD-II

O OBD-II (On-Board Diagnostics II) consiste em uma interface padronizada de diagnóstico automotivo que permite acessar informações operacionais do veículo por meio de serviços e identificadores de parâmetros, conhecidos como PIDs. Entre os parâmetros comumente utilizados neste padrão estão velocidade, rotação do motor e temperatura, todos diretamente relacionados ao escopo deste trabalho.

Do ponto de vista normativo, a padronização do OBD-II encontra respaldo em documentos como a ISO 15031 e a SAE J1979, que definem a estrutura dos serviços de diagnóstico e dos parâmetros empregados na comunicação entre o veículo e o equipamento de leitura. Nesse contexto, a relevância do OBD-II reside em oferecer uma base técnica comum para a obtenção de dados veiculares, reduzindo a dependência de soluções proprietárias e favorecendo a interoperabilidade entre dispositivos e sistemas.

Essa padronização é central para o presente trabalho porque sustenta tecnicamente a escolha das variáveis observadas no protótipo. No sistema desenvolvido, a seleção dos dados veiculares não foi realizada de forma arbitrária, mas orientada por parâmetros reconhecidos no contexto do diagnóstico automotivo, o que confere maior consistência à proposta de telemetria implementada. No ambiente experimental adotado neste TCC, a leitura desses parâmetros foi representada por simulador OBD/CAN, com base em identificadores padronizados, preservando a aderência conceitual ao modelo de aquisição de dados previsto nas normas.

Desse modo, embora o estudo não tenha realizado validação em veículo real, o uso do simulador mostrou-se metodologicamente compatível com a proposta acadêmica de demonstrar o fluxo de captura ou recepção. Assim, o OBD-II não representa apenas uma interface de leitura, mas também o referencial técnico que delimita, de forma objetiva, quais informações são pertinentes ao monitoramento veicular realizado neste estudo.

2.4. CAN BUS

O CAN Bus (Controller Area Network) é um protocolo de comunicação amplamente utilizado na indústria automotiva para a troca de informações entre módulos eletrônicos do veículo. Entre suas principais características destacam-se a robustez, a confiabilidade e a capacidade de operar adequadamente em ambientes sujeitos a ruído elétrico, o que o torna apropriado para aplicações críticas de comunicação embarcada.

No contexto automotivo, o CAN Bus constitui uma das principais bases de integração entre sistemas eletrônicos, permitindo que diferentes módulos compartilhem informações de forma padronizada e eficiente. Por essa razão, sua presença na literatura e nas soluções veiculares contemporâneas o torna uma referência técnica relevante para trabalhos que envolvem aquisição, circulação e interpretação de dados automotivos.

No presente TCC, o CAN Bus não é tratado apenas como tecnologia acessória, mas como parte da sustentação conceitual do sistema proposto, uma vez que representa a lógica de comunicação por meio da qual o protótipo se aproxima do ecossistema técnico dos veículos modernos. A arquitetura concebida ~~não foi realizada com conexão direta a veículo real, mas com~~ uso de simulador OBD/CAN, ~~esse simulador foi empregado para~~ representar, em condições controladas, a origem dos dados veiculares utilizados no experimento, permitindo reproduzir a lógica de recepção e tratamento das informações no contexto acadêmico da pesquisa.

Em ambiente experimental adotado acerca da prática do fluxo neste estudo, considerou-se a possibilidade do uso do módulo TJA1050 conectado ao ESP32 como forma de aproximação ao contexto real, visando representar a interface esperada para comunicação com dados automotivos reais.

Assim, a contribuição deste trabalho não está na validação física da comunicação em

rede CAN automotiva real, mas na demonstração da coerência funcional do fluxo de captura ou recepção.

Dessa forma, a discussão sobre CAN Bus fundamenta a escolha da forma de acesso às informações veiculares no protótipo, ao mesmo tempo em que delimita corretamente o alcance experimental da pesquisa, restrito a ambiente simulado e controlado.

2.5. Sistemas Embarcados e ESP32

Sistemas embarcados são sistemas computacionais dedicados ao controle, monitoramento ou processamento de funções específicas em um dispositivo físico. No setor automotivo, esses sistemas assumem papel central em razão da crescente integração entre eletrônica, software e comunicação de dados, tornando-se parte essencial do funcionamento dos veículos contemporâneos.

Traduzir

Sonko et al. (2024, p. 96) afirmam que a indústria automotiva passou por uma "revolutionary transformation with the integration of embedded systems", destacando ainda impactos sobre "vehicle design, performance, and safety". Essa leitura é importante porque situa os sistemas embarcados como elementos estruturantes da evolução tecnológica do setor automotivo, e não apenas como componentes auxiliares.

Em contexto mais próximo ao presente estudo, Silva (2025, p. 6) apresenta o desenvolvimento de um sistema embarcado "modular e de baixo custo para aquisição, processamento e transmissão de dados veiculares", ressaltando que a solução permite "coletar parâmetros do veículo, normalizá-los e publicá-los em redes locais ou remotas". Tal contribuição aproxima-se diretamente da proposta deste TCC, que também busca materializar uma arquitetura funcional de telemetria em contexto acadêmico.

Isso não é fundamentação teórica

Nesse cenário, a escolha do ESP32 como núcleo embarcado do sistema se justifica por sua adequação ao escopo do trabalho. O dispositivo reúne baixo custo, capacidade de processamento compatível com a proposta, recursos de conectividade e facilidade de integração com módulos externos. No protótipo desenvolvido, o ESP32 é responsável pela leitura ou recepção dos dados, montagem dos pacotes de telemetria, publicação das mensagens, armazenamento local e controle básico de periféricos, constituindo a base material do fluxo de dados investigado. Assim, os sistemas embarcados, no presente trabalho,

não são apenas um suporte técnico, mas o elemento que viabiliza concretamente a coleta e o encaminhamento das informações tratadas pela solução proposta.

2.6. IOT Automotiva e Conectividade Veicular

A Internet das Coisas aplicada ao setor automotivo parte do princípio de que o veículo pode ser tratado como uma entidade conectada, capaz de coletar, transmitir e compartilhar dados em tempo real ou quase real. Nesse contexto, a conectividade amplia a capacidade de monitoramento, integração entre sensores e comunicação com plataformas digitais, permitindo que os dados do veículo sejam utilizados para acompanhamento operacional e apoio à tomada de decisão.

Muthiya et al. (2023, p. 115) apresenta a IoT como um meio de obter dados de dispositivos remotos para "automate and notify the user", com ênfase em segurança, monitoramento e prevenção de falhas. Os autores observam ainda que veículos conectados passam a compartilhar dados e a permitir o monitoramento de sua condição operacional (MUTHIYA et al., 2023, p. 117-118). Essa leitura mostra que a IoT automotiva não se limita à coleta dos dados, mas envolve sua circulação e uso em sistemas conectados.

Ammar et al. (2022, p. 1) reforçam essa perspectiva ao afirmar que o setor automotivo já incorpora "advanced IoT solutions". Em outra parte do trabalho, os autores discutem modalidades de comunicação veicular, como Vehicle-to-Vehicle (V2V) e Vehicle-to-Infrastructure (V2I), como parte do avanço da conectividade automotiva (AMMAR et al., 2022, p. 4). Essas aplicações evidenciam que a IoT no domínio veicular está associada não apenas à coleta interna de dados, mas também à interação do veículo com outros elementos do ambiente conectado.

Essas contribuições se relacionam diretamente com o presente trabalho, uma vez que a arquitetura proposta também transforma leituras veiculares em registros estruturados, transmitidos e disponibilizados para consulta posterior. A diferença está no escopo: enquanto Muthiya et al. (2023) e Ammar et al. (2022) discutem a IoT automotiva em uma perspectiva mais ampla, abrangendo múltiplas formas de conectividade e aplicações industriais, este TCC concentra-se em um caso delimitado de telemetria acadêmica, com foco em captura ou recepção, transmissão, armazenamento e visualização de dados do veículo.

2.7. Registro de Dados e Armazenamento da Informação Veicular

O registro de dados constitui elemento central em sistemas de telemetria e caixa-preta, uma vez que sem armazenamento não há histórico, rastreabilidade nem possibilidade de análise posterior. Em aplicações veiculares, o valor da informação não está apenas em sua obtenção imediata, mas também em sua preservação para consulta, interpretação e recuperação futura.

Veitas e Delaere (2018) evidenciam esse ponto ao sustentar que a discussão sobre sistemas veiculares dessa natureza envolve não apenas a coleta, mas também o registro, o armazenamento e o gerenciamento de acesso aos dados. No resumo do trabalho, os autores defendem que tais sistemas devem assegurar "the confidentiality, integrity and availability of data related to operation of a vehicle" (VEITAS; DELAERE, 2018, p. 1), ampliando a compreensão do registro veicular ao longo de todo o ciclo de vida da informação.

Traduzir

Em perspectiva aplicada, Vanitha et al. (2023, p. 2) descrevem uma arquitetura em que "the second step is to save the data to an SD card module; and the third step is to upload the data to the cloud". Esse trecho é particularmente relevante porque explicita a coexistência entre armazenamento local e transmissão remota como estratégia para preservar os dados e ampliar sua disponibilidade.

Essa compreensão relaciona-se diretamente com o presente trabalho, uma vez que o problema investigado não se limita ao acesso momentâneo à informação, mas à construção de um fluxo em que o dado possa ser mantido, recuperado e utilizado posteriormente. No protótipo desenvolvido, essa lógica se materializa na combinação entre armazenamento local em cartão SD, utilizado para contingência, e armazenamento remoto em banco de dados, responsável pela organização e recuperação estruturada dos registros. A diferença principal em relação aos autores analisados está no enfoque arquitetural adotado neste TCC, que articula mensageria, backend e confirmação de recebimento para compor o fluxo de armazenamento da informação veicular.

2.8. MQTT e Mensageria na Arquitetura Proposta

MQTT é um protocolo leve de mensageria baseado no modelo publicador/assinante, concebido para contextos com restrições de largura de banda e recursos computacionais.

Segundo a OASIS (2019, p. 1), MQTT é um “Client Server publish/subscribe messaging transport protocol”, sendo descrito como “light weight, open, simple, and designed to be easy to implement”. A especificação destaca ainda que o protocolo é adequado a contextos em que “a small code footprint is required and/or network bandwidth is at a premium” (OASIS, 2019, p. 1). Além disso, o padrão publicador/assinante favorece o “decoupling of applications” (OASIS, 2019, p. 2), isto é, o desacoplamento entre os elementos que produzem e os que consomem a informação.

Essa característica torna o MQTT particularmente adequado ao tipo de arquitetura proposta neste trabalho. No protótipo desenvolvido, o módulo embarcado pública os dados de telemetria em um broker MQTT, implementado com EMQX, sem depender de conexão síncrona direta com a aplicação final. Essa lógica aproxima-se também do trabalho de Silva (2025, p. 6), que descreve uma solução capaz de integrar comunicação embarcada e publicar dados em redes locais ou remotas, evidenciando a utilidade da mensageria para sistemas de telemetria veicular de baixo custo.

No presente TCC, a mensageria sustenta a adoção de uma arquitetura orientada a eventos. Esse tipo de arquitetura consiste em um modelo de integração no qual os componentes do sistema se comunicam por meio da publicação e do consumo de eventos ou mensagens, em vez de depender exclusivamente de chamadas diretas e síncronas entre si. No protótipo desenvolvido, essa lógica se materializa na publicação dos dados de telemetria em um broker MQTT, permitindo que o backend os processe posteriormente sem dependência de comunicação direta e imediata com o módulo embarcado.

Esse desenho favorece flexibilidade arquitetural, separação de responsabilidades e organização do fluxo de dados, além de apresentar maior coerência com o uso de dispositivos embarcados e com sistemas distribuídos. Assim, o MQTT não é empregado apenas como mecanismo técnico de envio, mas como fundamento da comunicação assíncrona e da arquitetura orientada a eventos adotadas pela solução desenvolvida.

2.9. Manutenção Preditiva, Monitoramento Inteligente e Indústria 4.0

A literatura sobre veículos conectados frequentemente relaciona a telemetria veicular à manutenção preditiva e ao monitoramento inteligente, inserindo essas aplicações no contexto mais amplo da Indústria 4.0. Nessa perspectiva, os dados obtidos do veículo deixam de servir

apenas ao acompanhamento imediato e passam a compor uma base informacional para análise, prevenção de falhas e apoio à decisão.

Dhall e Solanki (2017, p. 16) sustentam essa visão ao afirmar que os dados coletados por sensores veiculares podem ser enviados a aplicações de backend para finalidades de "analytical and decision making purposes". Esse entendimento evidencia que o valor da telemetria não reside apenas na visualização dos parâmetros coletados, mas também na capacidade de interpretar o comportamento operacional do veículo a partir desses registros.

Em complemento, Ghosh et al. (2022, p. 328) inserem a IoT automotiva no contexto da Indústria 4.0 e destacam a necessidade de soluções "capable of managing, storing, processing, transmitting, receiving, and updating a large amount of data efficiently". Tal formulação reforça a ideia de que o monitoramento inteligente depende de um ciclo completo de tratamento da informação, envolvendo captura, comunicação, processamento e utilização do dado em arquiteturas conectadas.

Essa perspectiva relaciona-se diretamente com o presente trabalho, uma vez que o protótipo desenvolvido já incorpora, em escala acadêmica, esse ciclo básico da informação: captura embarcada ou recepção, transmissão, processamento, armazenamento e consulta posterior. Embora a solução proposta não implemente manutenção preditiva em sentido pleno, ela estabelece uma base coerente para evoluções futuras orientadas pela análise de eventos, pelo histórico operacional e pela interpretação do comportamento do veículo ao longo do tempo.

2.10. Testes e Validação de Sistemas Embarcados Automotivos

A validação de sistemas embarcados automotivos constitui aspecto essencial para garantir a coerência funcional entre os componentes envolvidos, especialmente em ambientes marcados por alta complexidade, integração entre módulos e exigências de comportamento em tempo real. Em sistemas dessa natureza, testar não significa apenas verificar se um software executa corretamente uma função isolada, mas avaliar se o conjunto da arquitetura opera de forma consistente diante de requisitos funcionais e não funcionais.

Grossmann, Serbanescu e Schieferdecker (2008, p. 1) afirmam que "the problems of testing systems that, like automobiles, steadily increase in complexity and interconnectedness

Traduzir

are still not solved", chamando atenção para o desafio crescente da validação no domínio automotivo. Os autores observam ainda que a qualidade nesse setor permanece marcada por "high manual portions" e por uma "multiplicity of often proprietary test systems and test platforms" (GROSSMANN; SERBANESCU; SCHIEFERDECKER, 2008, p. 2), o que dificulta a integração, o reuso e a padronização dos testes.

No contexto deste trabalho, essa discussão é relevante porque o protótipo desenvolvido também envolve integração entre múltiplos elementos, como módulo embarcado, simulador OBD/CAN, broker MQTT, backend, banco de dados e interface web. Evidentemente, o trabalho não pretende resolver o problema da validação automotiva em escala industrial. Entretanto, diante desse cenário, adotou-se uma estratégia metodológica compatível com o escopo acadêmico, baseada na verificação funcional do fluxo ponta a ponta, na observação controlada do comportamento dos componentes e na documentação dos artefatos de arquitetura, regras de negócio e armazenamento dos dados.

Dessa forma, a resposta prática do trabalho ao problema apontado pelos autores não está na criação de uma infraestrutura padronizada de testes, mas na delimitação de um ambiente experimental reproduzível, no qual foi possível avaliar de maneira objetiva o recebimento, o processamento, e armazenamento, a confirmação e a visualização das informações. Assim, a validação realizada buscou demonstrar a coerência funcional da arquitetura proposta, em vez de reivindicar cobertura industrial completa dos desafios de teste em sistemas automotivos embarcados.

3. METODOLOGIA

Além de sustentar os conceitos técnicos do trabalho, a fundamentação teórica acima descrita também oferece base para o enquadramento metodológico da pesquisa. Gil (2002, p. 1) define pesquisa como o "procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos" e afirma que ela se desenvolve em um processo que vai "desde a adequada formulação do problema até a satisfatória apresentação dos resultados" (GIL, 2002, p. 1). Essa compreensão é relevante para o presente trabalho porque evidencia que o desenvolvimento do protótipo não pode ser tratado apenas como execução técnica, mas como investigação organizada a partir de problemas, objetivos, procedimentos e análise dos resultados.

Marconi e Lakatos (2003, p. 155-163), ao tratarem do planejamento da pesquisa, apresentam etapas como especificação de objetivos, formulação do problema, delimitação da pesquisa, seleção de métodos e técnicas e organização do instrumental de pesquisa. Essa contribuição é importante porque se relaciona diretamente com a forma como este trabalho foi organizado, desde a definição do problema até a documentação dos procedimentos e da validação do sistema desenvolvido.

Do ponto de vista classificatório, Gil (2002, p. 16-17) também oferece suporte para o enquadramento metodológico do estudo ao discutir modalidades como estudo de caso e pesquisa documental. Tal enquadramento ajusta-se ao trabalho desenvolvido, uma vez que a investigação está voltada à construção e à validação de uma solução prática, ao mesmo tempo em que utiliza artefatos, registros técnicos e documentação produzida ao longo do projeto como parte do processo analítico.

Essa base metodológica relaciona-se diretamente com o presente trabalho, pois a pesquisa foi conduzida a partir da formulação de um problema concreto, da definição de objetivos compatíveis com o escopo proposto e da observação técnica do comportamento do sistema implementado. Desse modo, a fundamentação teórica sustenta não apenas a compreensão conceitual do protótipo, mas também a forma pela qual ele é investigado, descrito e analisado. No contexto deste Trabalho de Conclusão de Curso, isso se traduz na adoção de uma metodologia compatível com o ambiente acadêmico, voltada à documentação do artefato desenvolvido e à verificação funcional de sua arquitetura em ambiente experimental controlado.

Do ponto de vista metodológico, o presente trabalho configura-se como pesquisa aplicada, uma vez que se volta ao desenvolvimento de uma solução prática para o problema investigado. Quanto aos objetivos, caracteriza-se como exploratória e descritiva, pois busca examinar a viabilidade da arquitetura proposta e, ao mesmo tempo, descrever os componentes, os fluxos e o funcionamento da solução desenvolvida. No que se refere aos procedimentos técnicos, a pesquisa compreende levantamento bibliográfico, análise documental, estudo de caso e atividade experimental. O levantamento bibliográfico sustenta a fundamentação conceitual do trabalho, enquanto a análise documental se apoia nos artefatos produzidos ao longo do projeto, tais como diagramas, modelagens, códigos e registros técnicos. O estudo de caso decorre da concentração da investigação em um objeto específico, representado pelo protótipo de telemetria veicular desenvolvido. Já o caráter experimental manifesta-se na implementação e na observação controlada do comportamento da solução em ambiente acadêmico. Quanto à abordagem do problema, a pesquisa possui natureza predominantemente qualitativa, tendo em vista que a análise se concentrou na observação técnica do funcionamento do sistema, da integração entre seus componentes e da coerência do fluxo implementado.

Os diversos autores pesquisados e a influência no trabalho estão listados abaixo em um quadro onde descrevemos por autor a contribuição teórica e a relação com o trabalho:

Quadro 1 - Síntese dos Autores e Influência no Trabalho

Autor(es)	Contribuição teórica	Relação com o trabalho
Ammar et al. (2022)	Aplicações práticas da IoT automotiva	Reforça a utilidade do protótipo para monitoramento, conectividade e apoio à decisão
Dhall e Solanki (2017)	Telemetria e backend para análise e decisão	Sustenta o papel analítico do backend no tratamento dos dados
Ghosh et al. (2022)	IoT automotiva na Indústria 4.0	Relaciona o trabalho ao ciclo de gestão, processamento e atualização do dado
Gil (2002)	Classificação e desenho da pesquisa	Sustenta o enquadramento da pesquisa como aplicada e exploratória
Grossmann, Serbanescu e Schieferdecker (2008)	Validação de sistemas embarcados automotivos	Fundamenta a estratégia de verificação funcional do fluxo ponta a ponta
Marconi e Lakatos (2003)	Estrutura metodológica do trabalho científico	Apoia a organização formal da pesquisa

Muthiya et al. (2023)	IoT aplicada ao setor automotivo	Sustenta a visão do veículo como plataforma conectada e orientada à notificação e monitoramento
Silva (2025)	Telemetria veicular em contexto acadêmico nacional	Reforça a viabilidade de um protótipo modular, de baixo custo e voltado à telemetria embarcada
Sonko et al. (2024)	Evolução dos sistemas embarcados automotivos	Sustenta o enquadramento do projeto como solução embarcada conectada no domínio automotivo
Vanitha et al. (2023)	Caixa-preta automotiva baseada em IoT	Apoia a integração entre armazenamento local e transmissão remota
Veitas e Delaere (2018)	Registro, armazenamento e acesso a dados veiculares	Fundamenta a noção de caixa-preta e a importância do ciclo de vida do dado

Fonte: Elaborado pelos autores (2026)

3.1. Visão da Pesquisa

A pesquisa desenvolvida teve como finalidade investigar a viabilidade de um protótipo acadêmico de telemetria veicular, estruturado a partir da integração entre módulo embarcado, comunicação com dados automotivos, mensageria MQTT com uso do broker EMQX, backend em NestJS, armazenamento em PostgreSQL com apoio do Prisma ORM e interface web. Do ponto de vista arquitetural, a solução foi organizada segundo uma lógica orientada a eventos, na qual a recepção e a transmissão dos dados ocorrem de forma desacoplada em relação às etapas posteriores de processamento, armazenamento e visualização. No cenário arquitetural previsto, a obtenção dos dados ocorreria por OBD-II/CAN em veículo real. No entanto, no ambiente experimental adotado neste trabalho, a origem dos dados foi representada por simulador OBD/CAN, permitindo a validação do fluxo funcional da solução em contexto acadêmico e controlado.

Quanto aos objetivos, o estudo caracteriza-se como exploratório e descritivo. Exploratório porque busca examinar, em contexto acadêmico, a aplicação integrada de tecnologias voltadas à telemetria veicular, possibilitando maior aproximação com o problema investigado. Descritivo porque apresenta, observa e registra as características da arquitetura proposta, bem como o comportamento funcional do sistema desenvolvido ao longo do fluxo de recepção, transmissão, processamento, armazenamento e visualização dos dados.

Quanto aos procedimentos técnicos, a pesquisa classifica-se como bibliográfica, documental e estudo de caso. Bibliográfica porque se fundamenta em livros, artigos científicos, trabalhos acadêmicos, normas e documentos técnicos relacionados à telemetria veicular, à *IoT* automotiva, aos sistemas embarcados, ao registro de dados, à mensageria e à validação de sistemas. Documental porque utiliza materiais produzidos no próprio desenvolvimento do trabalho, como levantamento de requisitos, diagramas, documentação da arquitetura, regras de negócio, artefatos de modelagem, estrutura do banco de dados e registros técnicos da implementação. Configura-se também como estudo de caso por concentrar a investigação em um objeto específico, representado pelo protótipo de telemetria veicular desenvolvido no âmbito deste Trabalho de Conclusão de Curso.

Quanto à abordagem do problema, a pesquisa possui natureza predominantemente qualitativa, pois está voltada à compreensão do funcionamento da solução proposta e à interpretação técnica do desempenho da arquitetura em ambiente controlado. Assim, a análise não se concentra em tratamento estatístico ou em generalizações quantitativas, mas na observação do comportamento do sistema e na verificação de sua capacidade de atender ao objetivo estabelecido.

3.2. Processo de Pesquisa

O processo de pesquisa foi conduzido em etapas sucessivas e interdependentes. Inicialmente, realizou-se a definição do tema e a delimitação do problema de pesquisa - como desenvolver um sistema de telemetria veicular composto por hardware embarcado, mensageria, backend, banco de dados e interface web, capaz de coletar, transmitir, armazenar e exibir dados básicos do veículo em um contexto acadêmico e de baixo custo?.

Em seguida, procedeu-se à revisão de literatura, com levantamento de referências teóricas e técnicas voltadas à compreensão dos conceitos de telemetria veicular, caixa-preta automotiva, OBD-II, *CAN Bus*, sistemas embarcados, *IoT* automotiva, mensageria, registro de dados e validação de sistemas.

A partir desse embasamento, formulou-se a hipótese de que é possível desenvolver, em contexto acadêmico e com tecnologias de baixo custo, um sistema de telemetria veicular funcional capaz de realizar a captura ou recepção, transmissão, processamento, armazenamento e visualização de dados básicos do veículo por meio da integração entre módulo embarcado, mensageria, *backend*, banco de dados relacional e interface *web*.

Na sequência, procedeu-se à definição metodológica e arquitetural da solução, estabelecendo-se os componentes, as tecnologias, os fluxos e os artefatos necessários ao desenvolvimento do sistema. Nessa etapa, adotou-se uma organização orientada a eventos para a comunicação entre os componentes, de modo que a origem dos dados permanecesse desacoplada das etapas de processamento, armazenamento e visualização.

A etapa seguinte consistiu na implementação do protótipo. Essa decisão arquitetural materializou-se no uso da mensageria *MQTT* como mecanismo de integração entre os componentes, permitindo que os dados publicados pelo módulo embarcado fossem posteriormente processados pelo *backend* sem dependência de comunicação síncrona direta.

Paralelamente, foram produzidos artefatos complementares do projeto, como levantamento de requisitos, diagramas, documentação da *API*, regras de negócio, modelagem de dados e descrição arquitetural, evidenciando a aplicação de conhecimentos de Engenharia de *S*oftware no desenvolvimento da solução.

O ambiente experimental foi composto por módulo embarcado baseado em *ESP32*, simulador *OBD/CAN*, armazenamento local em cartão *SD*, mensageria *MQTT* com uso do *broker EMQX*, backend desenvolvido em *NestJS*, banco de dados *PostgreSQL* com apoio do Prisma ORM e interface web para consulta das informações registradas.

Por fim, realizou-se a análise empírica e funcional dos resultados, verificando se a arquitetura implementada foi capaz de cumprir o fluxo proposto de recepção, transmissão, processamento, armazenamento e visualização dos dados, de modo a responder ao problema de pesquisa e subsidiar as conclusões do estudo.

4. EXPERIMENTO E COLETA DE DADOS

Esta seção descreve o experimento realizado e o processo de coleta de dados adotado para a validação do protótipo de telemetria veicular desenvolvido neste trabalho.

4.1. Descrição do Experimento

No cenário arquitetural previsto, a obtenção dos dados ocorreria por OBD-II/CAN em veículo real. No entanto, no ambiente experimental adotado neste trabalho, a interação com a lógica de aquisição de dados automotivos não ocorreu por conexão direta com automóvel, mas por meio de simulador OBD/CAN. Esse simulador foi empregado para representar, em condições controladas, a origem dos dados veiculares utilizados no experimento, permitindo reproduzir a lógica de recepção e tratamento das informações prevista para a arquitetura proposta.

Quadro 2 - Etapas do Desenvolvimento

Eixo	Evidência no projeto	Artefato correspondente
Análise de requisitos	Levantamento de requisitos funcionais, não funcionais e regras de negócio.	Documento de requisitos e regras de negócio.
Armazenamento de dados	Modelagem relacional com integridade e índices.	Schema Prisma, DER e dicionário de dados.
Integração e mensageria	Arquitetura orientada a eventos com MQTT e broker dedicado.	Firmware, EMQX e backend NestJS.
Interface homem-máquina	Painel web para consulta de status, telemetria e eventos.	Frontend Next.js.
Modelagem	Casos de uso, classes, sequência, atividades e DER.	Diagramas do projeto.
Projeto arquitetural	Separação entre gateway, broker, backend, banco e frontend.	Documento arquitetural, diagramas de componentes/arquitetura.
Validação	Testes funcionais de ponta a ponta em ambiente experimental controlado.	Seção metodológica e experimento.

Fonte: Elaborado pelos autores (2026)

Desse modo, a validação concentrou-se na coerência funcional do fluxo entre módulo embarcado, mensageria, backend, armazenamento e interface web, e não na homologação da solução em rede automotiva real. Assim, a rede CAN aparece neste experimento como referência técnica do domínio automotivo, representada por simulação compatível com o escopo acadêmico da pesquisa.

Repetição

O experimento verificou se a arquitetura proposta era capaz de operar de forma coerente e integrada, desde a recepção dos dados simulados até sua exibição ao usuário. Para isso, foram observadas variáveis como a recepção dos dados de telemetria pelo módulo embarcado, a reorganização e a publicação das mensagens em MQTT, o processamento pelo backend, a armazenamento no banco de dados, a geração de eventos automáticos e a exibição das informações pela interface web.

No que se refere à camada de armazenamento, adotou-se o PostgreSQL em conjunto com o Prisma ORM. A escolha do PostgreSQL relaciona-se à necessidade de armazenamento estruturado, integridade relacional e organização histórica dos registros de telemetria e eventos. A adoção do Prisma justificou-se por sua capacidade de acelerar o desenvolvimento da aplicação, facilitar a modelagem dos dados e simplificar a integração entre o banco de dados e o *backend*. Em contexto acadêmico, essa escolha mostrou-se adequada por reduzir a complexidade de implementação da camada de acesso a dados e permitir maior concentração no desenvolvimento e na validação da arquitetura proposta.

Para a validação do fluxo de dados automotivos, foi utilizado um simulador OBD/CAN, adotado como recurso compatível com o escopo acadêmico do estudo. No cenário experimental efetivamente realizado, o simulador publicou dados via MQTT, que foram recebidos pelo ESP32, reorganizados em pacotes de telemetria e republicados ao *broker* EMQX, seguindo posteriormente para processamento no *backend*, armazenamento no banco de dados e visualização pela interface web.

No que se refere à camada de *frontend*, a interface web foi definida com base nas necessidades funcionais identificadas ao longo do desenvolvimento do protótipo, especialmente aquelas relacionadas à consulta de telemetria, à visualização de eventos, ao acompanhamento do status dos dispositivos e à organização das informações de veículos e usuários. Desse modo, a construção das telas não partiu de processo formal de prototipação em ferramenta específica, mas da tradução direta dos requisitos do sistema em componentes

de navegação e apresentação compatíveis com o escopo acadêmico do trabalho. Reconhece-se, contudo, que a interface pode ser aprimorada em estudos futuros, especialmente quanto à usabilidade, à organização visual e à avaliação com usuários.

A execução do experimento ocorreu de forma sequencial, compreendendo as seguintes etapas: configuração do ambiente de execução; inicialização do simulador OBD/CAN, do módulo embarcado e dos serviços de software; publicação dos dados simulados; recepção e reorganização das informações pelo ESP32; envio dos pacotes de telemetria ao *broker* EMQX; processamento das mensagens no *backend*; armazenamento dos registros no banco de dados; e visualização dos dados e eventos na interface *web*. Em cada etapa, verificou-se se o comportamento observado correspondia ao fluxo arquitetural previsto no projeto.

4.2. Detalhamento Técnico do Protótipo

A solução foi estruturada como uma arquitetura distribuída de telemetria veicular, composta por módulo embarcado, mensageria MQTT, *backend*, banco de dados relacional e interface *web*. Em termos de funcionamento, o módulo embarcado atua como ponto de organização inicial das informações, a mensageria realiza o encaminhamento assíncrono dos dados, o *backend* processa os registros e aplica regras de negócio, o banco de dados mantém o histórico armazenado e a interface *web* disponibiliza a consulta das informações. No cenário de uso real previsto, a obtenção dos dados ocorreria a partir da porta OBD-II/CAN do veículo. No desenvolvimento deste trabalho, contudo, a entrada dos dados foi representada por simulador OBD/CAN, permitindo demonstrar a integração entre os componentes da solução em ambiente acadêmico controlado.

No *hardware* embarcado, o componente central adotado foi o ESP32 DevKit. Sua escolha se justifica pelo baixo custo, pela conectividade integrada, pela ampla disponibilidade e pela capacidade de executar satisfatoriamente as funções previstas no protótipo. Em comparação com alternativas de maior custo ou complexidade, o ESP32 mostrou-se mais adequado ao escopo acadêmico do trabalho, pois permitiu implementar uma solução funcional sem exigir uma plataforma computacional mais robusta. No sistema desenvolvido, o microcontrolador foi responsável por receber ou ler os dados, organizar os pacotes de telemetria, registrar informações localmente em cartão SD e publicá-las no *broker* MQTT.

Como recurso de contingência, foi utilizado um módulo de cartão SD interligado ao ESP32 por interface SPI. A adoção desse componente decorreu da necessidade de manter armazenamento local temporário dos pacotes em situações de indisponibilidade da comunicação remota. Em vez de soluções embarcadas mais complexas, optou-se pelo cartão SD por sua simplicidade de integração, baixo custo e compatibilidade com a proposta do protótipo. No esquema de ligação adotado, o módulo foi conectado ao ESP32 com as seguintes associações principais: GND com GND, VCC com VIN, MISO com D19, MOSI com D23, SCK com D18 e CS com D15.

Também foram incorporados ao protótipo dois LEDs, um vermelho e um verde, além de um botão tátil. Esses elementos foram utilizados como apoio à sinalização local e à depuração do firmware durante o desenvolvimento. Os LEDs foram conectados ao ESP32 com resistores de 220 Ω (ohms), enquanto o botão foi integrado com resistor de 10 Ω (ohms). Embora não componham o núcleo principal da solução, esses componentes auxiliaram a observação do comportamento do sistema embarcado e a confirmação de estados básicos de operação.

Para representar a interface prevista com o domínio automotivo, o projeto considerou um módulo *CAN/transceiver* TJA1050. Sua presença no protótipo se justifica pela necessidade de materializar, no nível do projeto eletrônico, a forma de comunicação esperada com dados automotivos baseados em OBD-II/CAN. No esquema utilizado, esse módulo foi conectado ao ESP32 por meio das ligações GND com GND, VCC com VIN, TX com D5 e RX com D4, além dos resistores indicados no desenho de interligação. Deve-se ressaltar, entretanto, que a validação prática deste trabalho não foi realizada com veículo real conectado à rede CAN, mas com simulador OBD/CAN. Assim, a interface representada pelo módulo expressa a arquitetura prevista para uso real, sem constituir, nesta etapa, validação em campo.

Na camada de comunicação, foi adotado o protocolo MQTT com uso do *broker* EMQX. Essa escolha foi orientada pela necessidade de desacoplar o módulo embarcado das etapas posteriores de processamento e persistência, favorecendo uma arquitetura orientada a eventos e mais compatível com dispositivos de recursos limitados. Em comparação com envio síncrono direto ao backend, a mensageria MQTT mostrou-se mais adequada ao protótipo por simplificar o encaminhamento assíncrono dos dados e ampliar a flexibilidade do fluxo de tratamento das mensagens. O EMQX foi selecionado por sua aderência ao protocolo, facilidade de configuração e adequação ao contexto acadêmico da solução.

Na implementação adotada, o simulador OBD/CAN publica dados representativos do domínio automotivo em MQTT. O ESP32 recebe essas mensagens, reorganiza seu conteúdo em estrutura compatível com a telemetria do sistema, registra localmente os pacotes quando necessário e os republica ao *broker* EMQX. A partir desse ponto, os dados seguem para o *backend*, que realiza o processamento lógico da aplicação. O trecho de código a seguir exemplifica a lógica implementada no firmware para recepção, organização e publicação da telemetria.

Quadro 3 -Código 1 trecho do *firmware* responsável pela recepção dos dados simulados e publicação da telemetria via MQTT

```
#include <ArduinoJson.h>
#include <PubSubClient.h>
#include <WiFi.h>
const char *DEVICE_NAME = "ESP_READER";
const char *MQTT_BROKER = "BROKER_IP";
const int MQTT_PORT = 1884;
const char *MQTT_USERNAME = "USUARIO_MQTT";
const char *MQTT_PASSWORD = "SENHA_MQTT";
const char *MQTT_INPUT_TOPIC = "simulador/obd";
const char *MQTT_OUTPUT_TOPIC = "telemetria/veicular";
WiFiClient espClient;
PubSubClient mqttClient(espClient);
int currentRpm = 0;
int currentVelocity = 0;
float currentTemp = 0.0f;
void publishTelemetry(int rpm, int velocity, float temp) {
  if (!mqttClient.connected()) {
    Serial.println("[MQTT OUT] Cannot publish. MQTT disconnected.");
    return;
  }
  StaticJsonDocument<256> doc;
  doc["pacoteId"] = String(DEVICE_NAME) + "-" + String(millis());
  doc["codigoDispositivo"] = DEVICE_NAME;
  doc["timestamp"] = "";
  doc["velocidadeObd"] = velocity;
  doc["rpm"] = rpm;
  doc["temperaturaMotor"] = temp;
  char payload[256];
  size_t payloadSize = serializeJson(doc, payload, sizeof(payload));
  if (payloadSize == 0) {
    Serial.println("[JSON ERROR] Failed to serialize output payload.");
    return;
  }
  bool sent = mqttClient.publish(MQTT_OUTPUT_TOPIC, payload);
```

```
  if (sent) {
    Serial.print("[MQTT FLOW] ");
    Serial.print(MQTT_INPUT_TOPIC);
    Serial.print(" -> ");
    Serial.print(MQTT_OUTPUT_TOPIC);
    Serial.print(" | ");
    Serial.println(payload);
  } else {
    Serial.print("[MQTT OUT] Publish FAILED. State: ");
    Serial.println(mqttClient.state());
  }
}
void handleMqttMessage(char *topic, byte *payload, unsigned int length) {
  if (strcmp(topic, MQTT_INPUT_TOPIC) != 0) {
    return;
  }
  StaticJsonDocument<256> doc;
  DeserializationError error = deserializeJson(doc, payload, length);
  if (error) {
    Serial.print("[JSON ERROR] Invalid simulator payload: ");
    Serial.println(error.c_str());
    return;
  }
  currentRpm = doc["rpm"] | currentRpm;
  currentVelocity = doc["velocidadeObd"] | doc["velocity"] | currentVelocity;
  currentTemp = doc["temperaturaMotor"] | doc["motor_temp"] | currentTemp;
  publishTelemetry(currentRpm, currentVelocity, currentTemp);
}
void setup() {
  mqttClient.setServer(MQTT_BROKER, MQTT_PORT);
  mqttClient.setCallback(handleMqttMessage);
}
void loop() {
  if (mqttClient.connected()) {
    mqttClient.loop();
  }
}
```

Fonte: Elaborado pelos autores (2026)

O Código 1 apresenta a lógica implementada no firmware do ESP32 para recepção dos dados simulados, organização da telemetria e publicação das mensagens via MQTT. Inicialmente, o dispositivo mantém a configuração dos tópicos de entrada e saída e define as variáveis que armazenam os valores correntes de rotação, velocidade e temperatura.

Quando uma mensagem é recebida no tópico de entrada, o firmware desserializa o conteúdo em formato JSON, extrai os campos relevantes e atualiza os valores da telemetria em memória. Em seguida, esses dados são reorganizados em uma nova estrutura, contendo identificador do pacote, código do dispositivo, *timestamp* e parâmetros veiculares, e então publicados no tópico de saída destinado ao restante da arquitetura.

Esse comportamento evidencia o papel do ESP32 como elemento intermediário entre a origem dos dados e a infraestrutura de mensageria do sistema, realizando a adaptação do conteúdo recebido antes de sua transmissão ao *backend*.

Na camada de processamento, foi utilizado um *backend* desenvolvido com NestJS. A escolha desse *framework* se justificou por sua organização modular, pela separação clara entre responsabilidades da aplicação e pela facilidade de integração com a camada de persistência. Em vez de uma implementação menos estruturada, o uso do NestJS permitiu organizar de forma mais consistente a recepção da telemetria, a aplicação das regras de negócio, a geração de eventos automáticos e a disponibilização dos dados para consulta. O *backend*, portanto, não apenas armazena os dados recebidos, mas interpreta as informações e identifica condições relevantes definidas na lógica do sistema.

Quadro 4 - Código 2 Trecho do *backend* responsável pelo processamento da telemetria e geração de eventos

```
import { Injectable, NotFoundException } from '@nestjs/common';
import { PrismaService } from '../prisma/prisma.service';
import { SeveridadeEvento, StatusSincronizacao, TipoEvento } from '@prisma/client';
import { CreateRegistroTelemetriaDto } from './dto/create-registro-telemetria.dto';
@Injectable()
export class TelemetriaService {
  constructor(private readonly prisma: PrismaService) {}
  async create(dto: CreateRegistroTelemetriaDto) {
    const dispositivo = await this.prisma.dispositivo.findUnique({
      where: { codigoDispositivo: dto.codigoDispositivo },
      include: { veiculo: { include: { configuracao: true } } },
    });
    if (!dispositivo) {
      throw new NotFoundException('Codigo do dispositivo nao encontrado.');
    }
    const registro = await this.prisma.registroTelemetria.create({
      data: {
        pacoteId: dto.pacoteId,
        dispositivoId: dispositivo.id,
```

```
      veiculoId: dispositivo.veiculoId,
      timestamp: new Date(dto.timestamp),
      velocidadeObd: dto.velocidadeObd,
      rpm: dto.rpm,
      temperaturaMotor: dto.temperaturaMotor,
    },
  });
  await this.prisma.dispositivo.update({
    where: { id: dispositivo.id },
    data: {
      statusSincronizacao: StatusSincronizacao.SINCRONIZADO,
      ultimaSincronizacao: new Date(),
    },
  });
  const configuracaoAplicada =
    dispositivo.veiculo.configuracao ??
    (await this.prisma.configuracaoVeiculo.create({
      data: { veiculoId: dispositivo.veiculoId },
    }));
   const eventosGerados = await this.gerarEventosDerivados({
    registro,
    dispositivoId: dispositivo.id,
    veiculoId: dispositivo.veiculoId,
    configuracao: configuracaoAplicada,
  });
  return { registro, eventosGerados, configuracaoAplicada };
 }
 private async gerarEventosDerivados(params) {
  const { registro, dispositivoId, veiculoId, configuracao } = params;
  const eventosParaCriar = [];
  const velocidadeAtual = registro.velocidadeObd ?? null;
  if (velocidadeAtual !== null && velocidadeAtual > configuracao.limiteVelocidade) {
    eventosParaCriar.push({
      tipo: TipoEvento.EXCESSO_VELOCIDADE,
            descricao: `Velocidade registrada de ${velocidadeAtual} km/h acima do limite de
${configuracao.limiteVelocidade} km/h.`,
      severidade: SeveridadeEvento.MEDIA,
    });
  }
   const registroAnterior = await this.prisma.registroTelemetria.findFirst({
    where: {
      dispositivoId,
      timestamp: { lt: registro.timestamp },
      id: { not: registro.id },
    },
    orderBy: { timestamp: 'desc' },
  });
```

```
    if (registroAnterior && velocidadeAtual !== null) {
      const velocidadeAnterior = registroAnterior.velocidadeObd ?? null;
      const diferencaSegundos =
        (registro.timestamp.getTime() - registroAnterior.timestamp.getTime()) / 1000;
      if (velocidadeAnterior !== null && diferencaSegundos > 0) {
        const variacao = velocidadeAtual - velocidadeAnterior;
        if (variacao <= -configuracao.limiteFrenagemBrusca && diferencaSegundos <= 2) {
          eventosParaCriar.push({
            tipo: TipoEvento.FRENAGEM_BRUSCA,
              descricao: `Reducao de ${Math.abs(variacao)} km/h em ${diferencaSegundos.toFixed(1)}
segundos.`,
            severidade: SeveridadeEvento.ALTA,
          });
        }
         if (variacao >= configuracao.limiteAceleracaoBrusca && diferencaSegundos <= 3) {
          eventosParaCriar.push({
            tipo: TipoEvento.ACELERACAO_BRUSCA,
            descricao: `Aumento de ${variacao} km/h em ${diferencaSegundos.toFixed(1)} segundos.`,
            severidade: SeveridadeEvento.MEDIA,
          });
        }
      }
    }
    if (eventosParaCriar.length === 0) {
      return [];
    }
    await this.prisma.eventoVeicular.createMany({
      data: eventosParaCriar.map((evento) => ({
        veiculoId,
        dispositivoId,
        registroTelemetriaId: registro.id,
        tipo: evento.tipo,
        descricao: evento.descricao,
        severidade: evento.severidade,
        timestamp: registro.timestamp,
      })),
    });
    return this.prisma.eventoVeicular.findMany({
      where: { registroTelemetriaId: registro.id },
      orderBy: { createdAt: 'asc' },
    });
  }
  }
```

Fonte: Elaborado pelos autores (2026)

O Código 2 exemplifica o comportamento da camada de *backend* responsável pelo processamento da telemetria recebida, pelo armazenamento dos registros e pela geração de

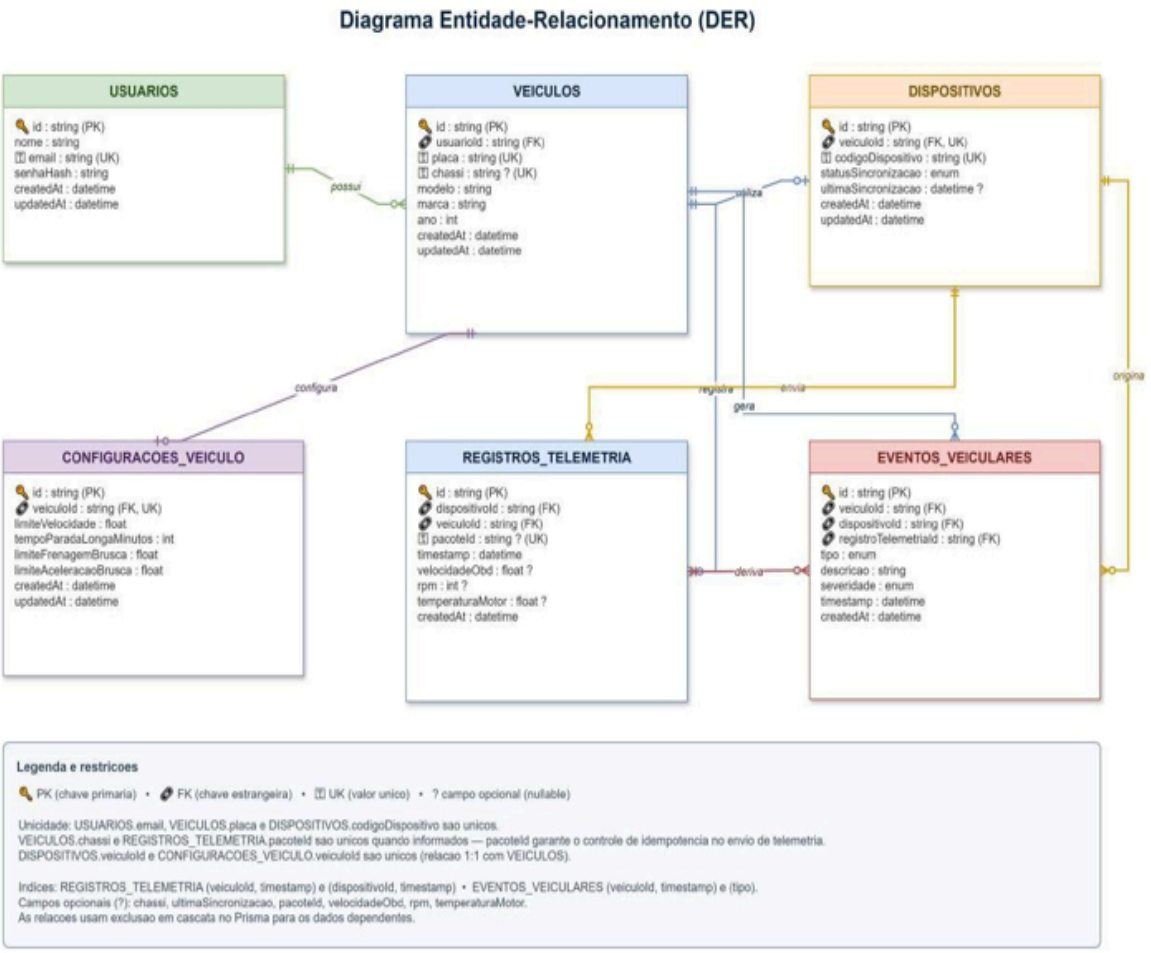
eventos automáticos. No trecho apresentado, o serviço localiza inicialmente o dispositivo associado ao código recebido, validando sua existência e cria um novo registro de telemetria com os dados enviados pelo módulo embarcado. Na sequência, atualiza o status de sincronização do dispositivo e recupera a configuração aplicável ao veículo. A partir dessas informações, o *backend* executa a lógica de derivação de eventos, verificando condições como excesso de velocidade, frenagem brusca e aceleração brusca com base nos parâmetros de configuração e na comparação com registros anteriores. Desse modo, o código evidencia que o *backend* não atua apenas como repositório de dados, mas como camada de interpretação lógica da telemetria, transformando leituras brutas em informações mais úteis para análise do comportamento veicular.

Para o armazenamento dos dados, adotou-se o PostgreSQL em conjunto com o Prisma ORM. A escolha do PostgreSQL justificou-se pela necessidade de armazenamento relacional estruturado, integridade entre entidades e manutenção consistente do histórico de usuários, veículos, dispositivos, registros de telemetria e eventos gerados pelo sistema. O Prisma ORM, por sua vez, foi utilizado como camada intermediária entre o *backend* e o banco de dados, contribuindo para simplificar a modelagem da base, padronizar o acesso aos dados e reduzir a complexidade de implementação da persistência.

No contexto acadêmico do trabalho, essa combinação mostrou-se adequada por favorecer maior organização da camada de dados e acelerar o desenvolvimento da solução, sem comprometer a clareza estrutural da modelagem adotada.

A Figura 1 apresenta o diagrama entidade-relacionamento da base de dados desenvolvida, evidenciando as principais tabelas, campos e relacionamentos que sustentam a persistência das informações do sistema. Em seguida, o Código 3 apresenta um trecho do schema Prisma correspondente à modelagem implementada, e o Quadro 6 sintetiza, em formato textual, a estrutura principal do banco de dados.

Figura 1 - Diagrama Entidade-Relacionamento (DER)



Fonte: Elaborado pelos autores (2026)

Quadro 5 - Código 3 Trecho do schema Prisma com as principais entidades do sistema

```
enum StatusSincronizacao {
  SINCRONIZADO
  NAO_SINCRONIZADO
}
 enum SeveridadeEvento {
  BAIXA
  MEDIA
  ALTA
}
enum TipoEvento {
  FRENAGEM_BRUSCA
  ACELERACAO_BRUSCA
  EXCESSO_VELOCIDADE
  PARADA_PROLONGADA
}
```

```
model Veiculo {
  id          String              @id @default(uuid())
  usuarioId   String
  placa       String              @unique
  chassi      String?             @unique
  modelo      String
  marca       String
  ano         Int
  dispositivo  Dispositivo?
  telemetrias  RegistroTelemetria[]
  eventos      EventoVeicular[]
  configuracao ConfiguracaoVeiculo?
  @@index([usuarioId])
  @@map("veiculos")
}
model Dispositivo {
  id                String             @id @default(uuid())
  veiculoId         String             @unique
  codigoDispositivo String             @unique
  statusSincronizacao StatusSincronizacao  @default(NAO_SINCRONIZADO)
  ultimaSincronizacao DateTime?
  registrosTelemetria RegistroTelemetria[]
  eventos           EventoVeicular[]
  veiculo Veiculo @relation(fields: [veiculoId], references: [id], onDelete: Cascade)
  @@map("dispositivos")
}
model RegistroTelemetria {
  id              String          @id @default(uuid())
  pacoteId        String?         @unique
  dispositivoId   String
  veiculoId       String
  timestamp       DateTime
  velocidadeObd   Float?
  rpm             Int?
  temperaturaMotor Float?
  eventos         EventoVeicular[]
  dispositivo Dispositivo @relation(fields: [dispositivoId], references: [id], onDelete: Cascade)
  veiculo     Veiculo     @relation(fields: [veiculoId], references: [id], onDelete: Cascade)
  @@index([veiculoId, timestamp])
  @@index([dispositivoId, timestamp])
  @@map("registros_telemetria")
}
model EventoVeicular {
  id                String           @id @default(uuid())
  veiculoId         String
  dispositivoId     String
  registroTelemetriaId String
```

```
  tipo              TipoEvento
  descricao         String
  severidade        SeveridadeEvento
  timestamp         DateTime
   veiculo          Veiculo          @relation(fields: [veiculoId], references: [id], onDelete: Cascade)
   dispositivo      Dispositivo      @relation(fields: [dispositivoId], references: [id], onDelete:
Cascade)
    registroTelemetria RegistroTelemetria @relation(fields: [registroTelemetriaId], references: [id],
onDelete: Cascade)
  @@index([veiculoId, timestamp])
  @@index([tipo])
  @@map("eventos_veiculares")
}
model ConfiguracaoVeiculo {
  id                  String   @id @default(uuid())
  veiculoId           String   @unique
  limiteVelocidade    Float    @default(80)
  tempoParadaLongaMinutos Int     @default(10)
  limiteFrenagemBrusca   Float    @default(20)
  limiteAceleracaoBrusca  Float    @default(25)
  veiculo Veiculo @relation(fields: [veiculoId], references: [id], onDelete: Cascade)
  @@map("configuracoes_veiculo")
}
```

Fonte: Elaborado pelos autores (2026)

O Código 3 apresenta um trecho do schema Prisma utilizado na modelagem da camada de armazenamento da aplicação. Nele, observam-se as principais entidades da solução, como veículo, dispositivo, registro de telemetria, evento veicular e configuração do veículo, além dos enumeradores que representam estados de sincronização, níveis de severidade e tipos de eventos.

A estrutura define relações entre essas entidades, como o vínculo entre dispositivo e veículo, entre registros de telemetria e seus respectivos veículos e dispositivos, e entre eventos e os registros que lhes deram origem. Também são estabelecidas restrições de unicidade e índices que favorecem a integridade relacional e a eficiência das consultas. Esse trecho demonstra como o Prisma ORM foi empregado para representar formalmente a organização da base de dados, servindo como elo entre a modelagem conceitual e a implementação efetiva da persistência no PostgreSQL.

Quadro 6 - Síntese Textual do Banco de Dados

Itens	Campos principais	Relacionamentos centrais
Usuários	id, nome, email, senhaHash	1:N com veículos.
Veículos	id, usuarioId, placa, chassi, modelo, marca, ano	N:1 com usuários; 1:1 com dispositivos; 1:N com registros telemetria e eventos veiculares; 1:1 com configurações veículo.
Dispositivos	id, veiculoId, codigoDispositivo, statusSincronizacao	1:1 com veículos; 1:N com registros telemetria e eventos veiculares.
Registros telemetria	id, pacoteId, dispositivoId, veiculoId, timestamp, velocidadeObd, rpm, temperaturaMotor	N:1 com veículos e dispositivos; 1:N com eventos veiculares.
Eventos veiculares	id, veiculoId, dispositivoId, registroTelemetriaId, tipo, descricao, severidade, timestamp	N:1 com veículos, dispositivos e registros telemetria.
Configurações veículo	id, veiculoId, limiteVelocidade, tempoParadaLongaMinutos, limiteFrenagemBrusca, limiteAceleracaoBrusca	1:1 com veículos.

Fonte: Elaborado pelos autores (2026)

Na camada de *Fontend*, foi desenvolvida uma interface *web* com Next.js. A adoção desse *framework* se justifica pela organização em componentes, facilidade de integração com a API e adequação à construção de telas voltadas à consulta da telemetria e à visualização de eventos. A interface foi definida a partir das necessidades funcionais identificadas no desenvolvimento do sistema, especialmente acompanhamento de registros, status dos dispositivos e informações associadas a veículos e usuários.

Quadro 7 - Código 4 Trecho do *frontend* responsável pela consulta e exibição da telemetria

```
// src/lib/api.ts
export function fetchVehicleTelemetry(baseUrl: string, veiculoId: string) {
  return authenticatedRequest<TelemetryRecord[]>(
    baseUrl,
    `/telemetria/veiculo/${veiculoId}`
  );
}
// app/historico/page.tsx
"use client";
```

```
import { useEffect, useState } from "react";
import {
  fetchVehicleTelemetry,
  fetchVehicles,
  getStoredBaseUrl
} from "@/src/lib/api";
import type { TelemetryRecord, VehicleRecord } from "@/src/types/telemetry";
export default function HistoricoPage() {
  const [vehicles, setVehicles] = useState<VehicleRecord[]>([]);
  const [vehicleId, setVehicleId] = useState("");
  const [records, setRecords] = useState<TelemetryRecord[]>([]);
  const [error, setError] = useState<string | null>(null);
  useEffect(() => {
    async function loadVehicles() {
      const result = await fetchVehicles(getStoredBaseUrl());
      if (result.ok) {
        setVehicles(result.data);
        const firstId = result.data[0]?.id || "";
        setVehicleId(firstId);
        if (firstId) {
          await loadTelemetry(firstId);
        }
      } else {
        setError(result.message);
      }
    }

    void loadVehicles();
  }, []);
  async function loadTelemetry(nextVehicleId: string) {
    const result = await fetchVehicleTelemetry(getStoredBaseUrl(), nextVehicleId);
    if (result.ok) {
      setRecords(result.data);
      setError(null);
    } else {
      setError(result.message);
      setRecords([]);
    }
  }
  return (
    <section>
      <select
        value={vehicleId}
        onChange={(event) => void loadTelemetry(event.target.value)}
      >
        <option value="">Selecione um veículo</option>
        {vehicles.map((vehicle) => (
```

```
        <option key={vehicle.id} value={vehicle.id}>
          {vehicle.placa} - {vehicle.marca} {vehicle.modelo}
        </option>
      ))}
    </select>
    {error ? <p>{error}</p> : null}
    <table>
      <thead>
        <tr>
          <th>Data</th>
          <th>Velocidade</th>
          <th>RPM</th>
          <th>Temperatura</th>
          <th>Eventos</th>
        </tr>
      </thead>
      <tbody>
        {records.map((record) => (
          <tr key={record.id}>
            <td>{record.timestamp}</td>
            <td>{record.velocidadeObd ?? "--"} km/h</td>
            <td>{record.rpm ?? "--"}</td>
            <td>{record.temperaturaMotor ?? "--"} C</td>
            <td>{record.eventos?.length ?? 0}</td>
          </tr>
        ))}
      </tbody>
    </table>
  </section>
 );
}
```

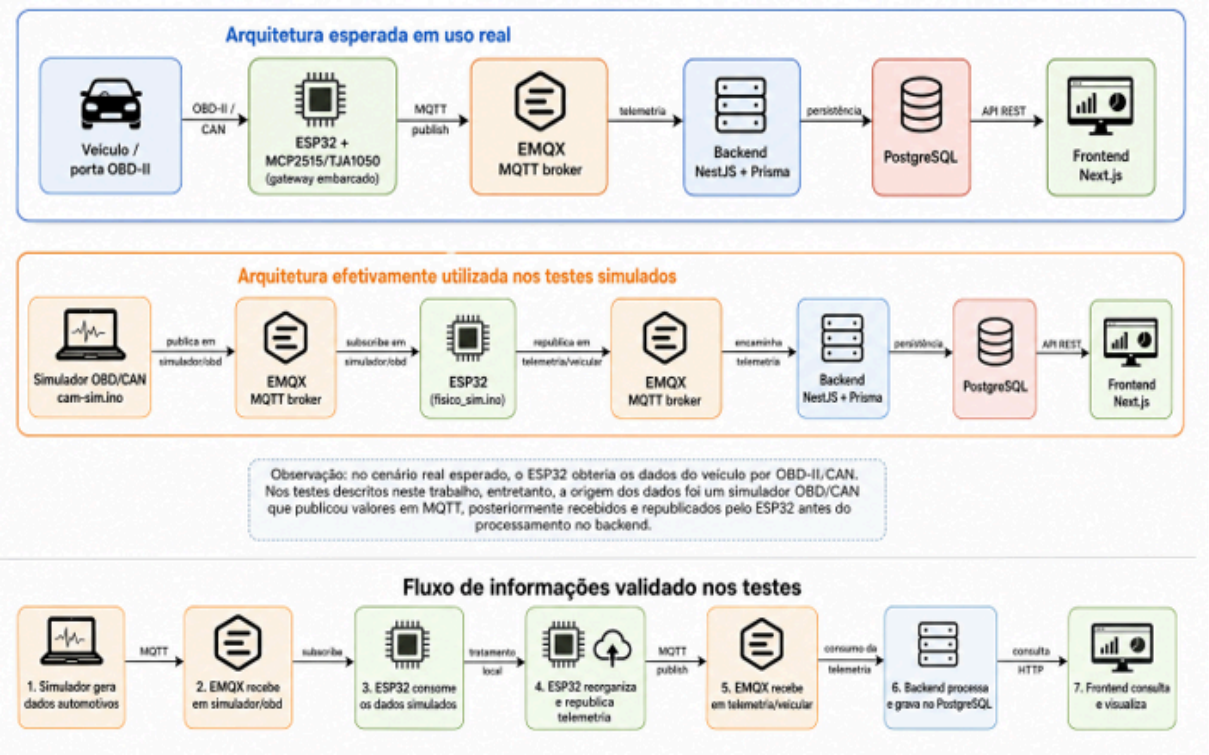
Fonte: Elaborado pelos autores (2026)

O Código 4 ilustra a lógica do *frontend* desenvolvida em Next.js para consulta e exibição da telemetria por veículo. Inicialmente, a aplicação recupera a lista de veículos disponíveis por meio da API e define o veículo selecionado para consulta. A partir dessa seleção, executa uma nova requisição ao *backend* para obter os registros de telemetria associados ao veículo escolhido. Os resultados retornados são então armazenados no estado da interface e apresentados em tabela, com colunas correspondentes à data da leitura, velocidade, rotação do motor, temperatura e quantidade de eventos vinculados. Esse comportamento mostra que a camada de apresentação foi estruturada para transformar os dados persistidos no *backend* em informação legível ao usuário, viabilizando a consulta histórica das leituras registradas pelo sistema.

As imagens seguintes registram o protótipo físico e etapas representativas do fluxo implementado, como publicação do simulador, recepção e republicação pelo ESP32, processamento no *backend* e visualização das informações no *frontend.*

A Figura 2 apresenta a arquitetura geral da solução, distinguindo a configuração prevista para uso real, da configuração efetivamente adotada no desenvolvimento do trabalho.

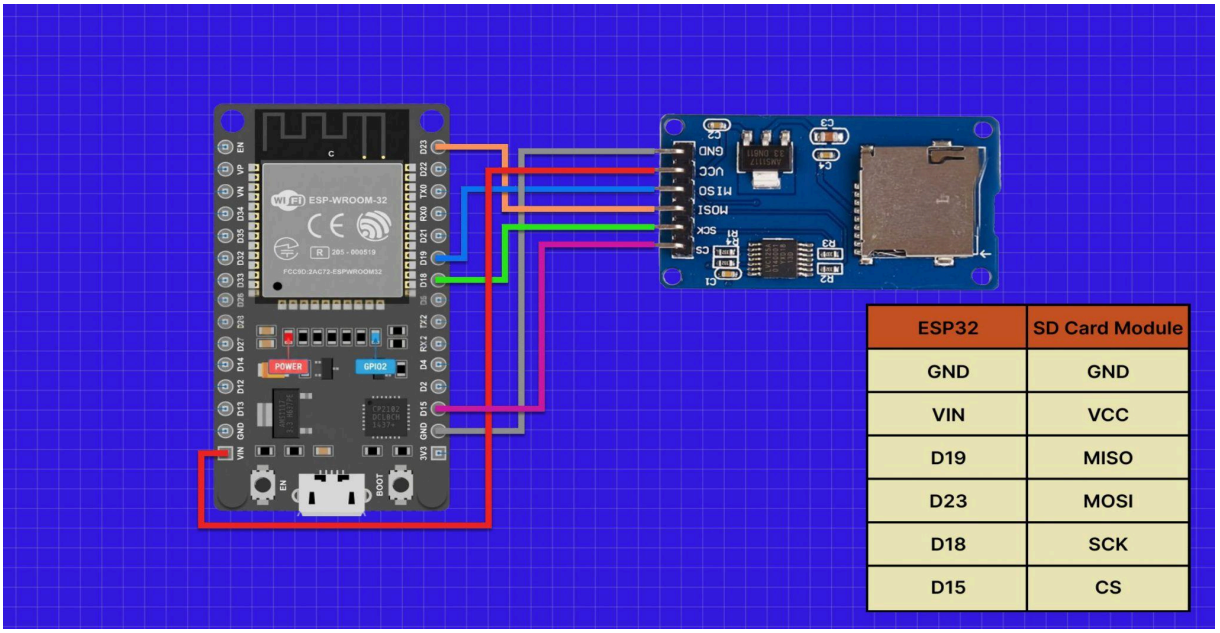
Figura 2 - Arquitetura esperada e arquitetura simulada do protótipo



Fonte: Elaborado pelos autores (2026)

As Figuras 3, 4 e 5 mostram, respectivamente, a ligação do módulo SD, dos LEDs e botão, e do módulo CAN/transceiver ao ESP32.

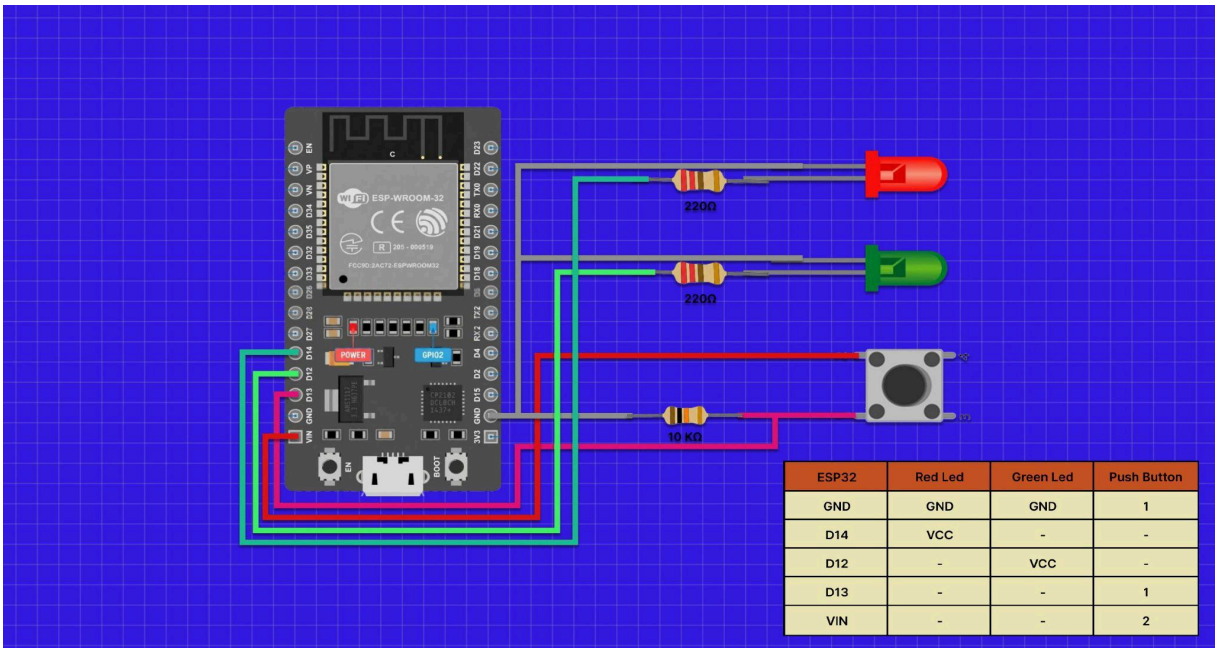
Figura 3 - Ligação do módulo SD ao ESP32



Fonte: Elaborado pelos autores (2026)

A Figura 3 apresenta a ligação do módulo de cartão SD ao ESP32 por interface SPI, evidenciando os pinos utilizados para alimentação e troca de dados. Nesse arranjo, o módulo foi conectado ao microcontrolador por meio das linhas GND, VCC, MISO, MOSI, SCK e CS, permitindo o armazenamento local dos pacotes de telemetria para fins de armazenamento e preservação temporária das informações.

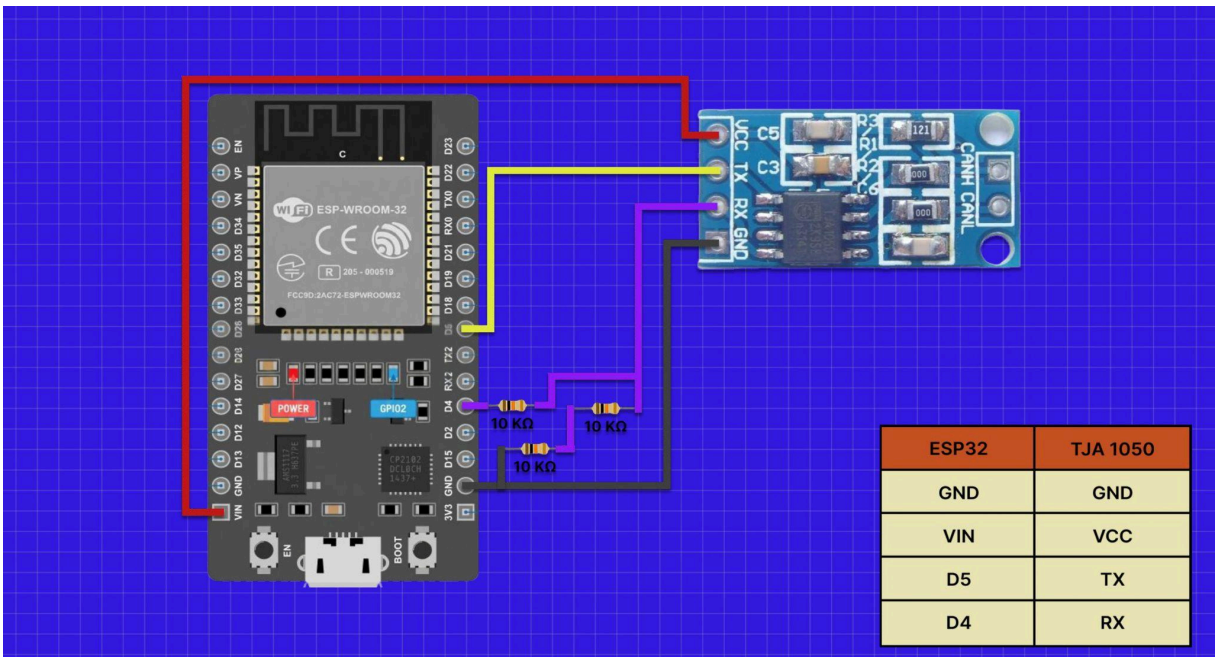
Figura 4 - Ligação dos LEDs e botão ao ESP32



Fonte: Elaborado pelos autores (2026)

A Figura 4 apresenta a ligação dos LEDs indicadores e do botão tátil ao ESP32, incluindo os resistores empregados no circuito. Esses componentes foram utilizados como apoio à sinalização local do funcionamento do protótipo e à depuração do firmware, permitindo observar estados básicos de operação e interações simples durante os testes de desenvolvimento.

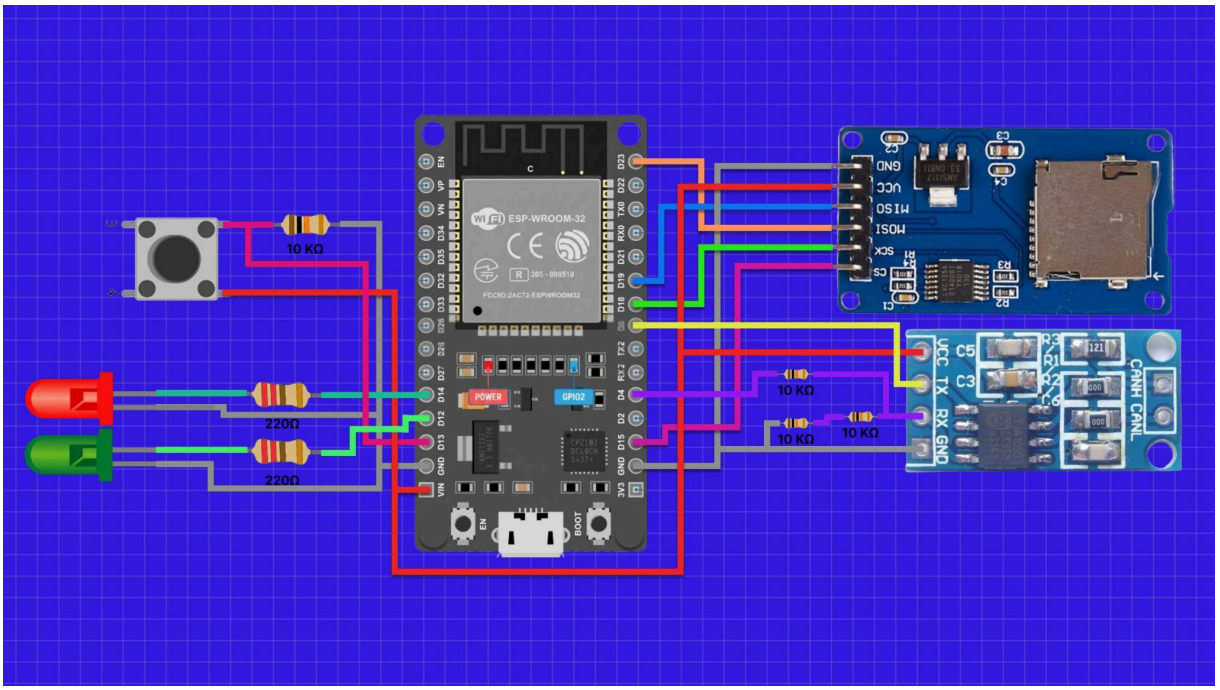
Figura 5 - Ligação do módulo CAN/transceiver ao ESP32



Fonte: Elaborado pelos autores (2026)

A Figura 5 apresenta a ligação prevista entre o ESP32 e o módulo *CAN/transceiver*, representando a interface eletrônica pensada para aproximação com o domínio automotivo. O esquema evidencia os pinos de alimentação e comunicação utilizados entre os módulos, constituindo a base física prevista para integração com dados automotivos em cenário de uso real, embora a validação do fluxo neste trabalho tenha sido realizada em ambiente simulado.

Figura 6 - Visão consolidada das interligações do protótipo embarcado

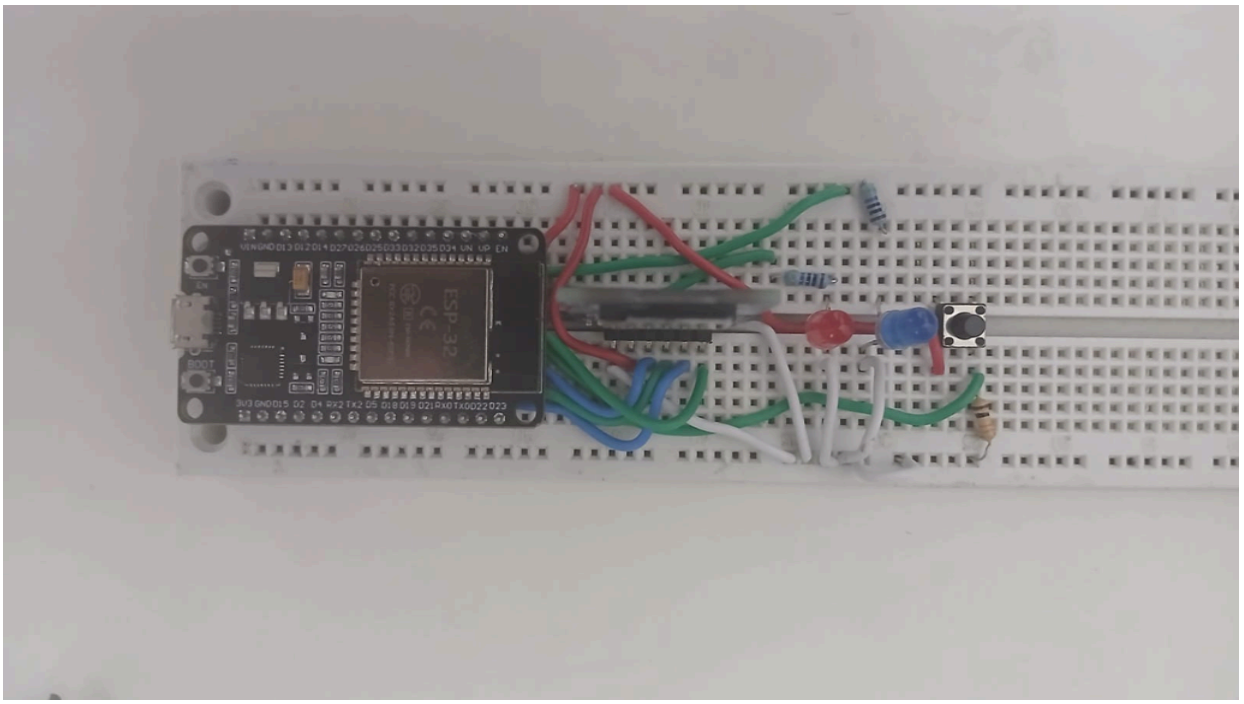


Fonte: Elaborado pelos autores (2026)

A Figura 6 reúne, em um único esquemático, a visão consolidada das interligações do protótipo embarcado, integrando o módulo de cartão SD, os LEDs, o botão tátil e o módulo *CAN/transceiver* ao ESP32. Essa representação permite visualizar, de forma sintética, a composição física do módulo embarcado da solução e a organização dos principais elementos de hardware utilizados no desenvolvimento do protótipo.

As imagens a seguir são fotos que configuram-se em evidência do funcionamento do protótipo desenvolvido e da efetivação prática da pesquisa, abrangendo tanto a montagem física do módulo embarcado quanto a validação do fluxo de dados entre simulador, módulo embarcado, mensageria, *backend* e *frontend*.

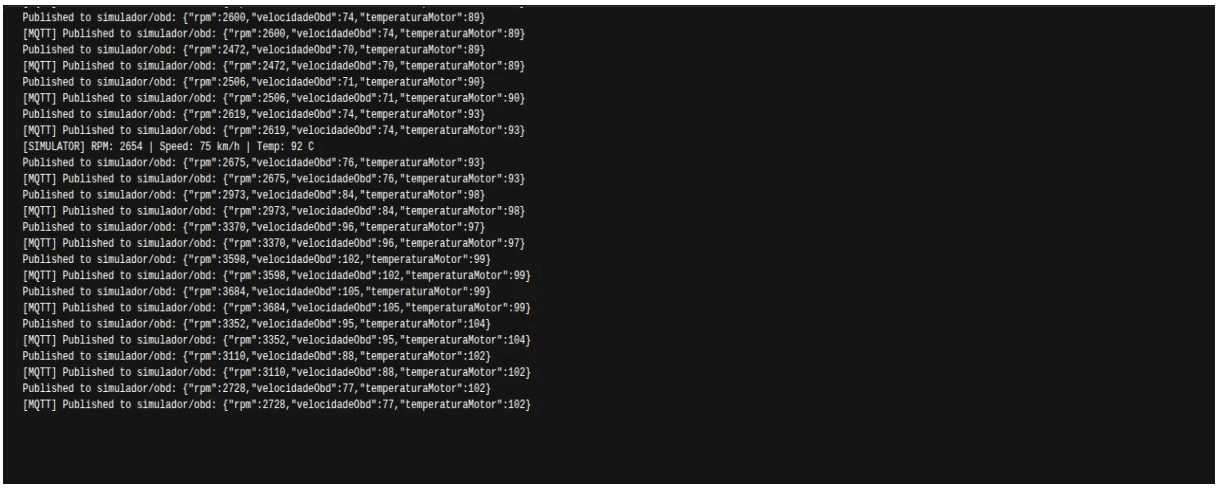
Imagem 1 - Protótipo físico montado



Fonte: Autoria própria (2026)

A Imagem 1 apresenta o protótipo físico montado, evidenciando a organização dos componentes embarcados utilizados na solução. A fotografia registra a materialização da arquitetura de *hardware* descrita anteriormente, especialmente no que se refere ao ESP32, ao módulo de cartão SD e aos elementos auxiliares de operação. Ressalta-se, contudo, que a interface com módulo *CAN/transceiver*, embora prevista nos esquemas apresentados anteriormente como parte da arquitetura esperada para o domínio automotivo, não foi utilizada na validação prática realizada neste trabalho, a qual ocorreu em ambiente simulado.

Imagem 2 – Simulador OBD/CAN em execução publicando dados

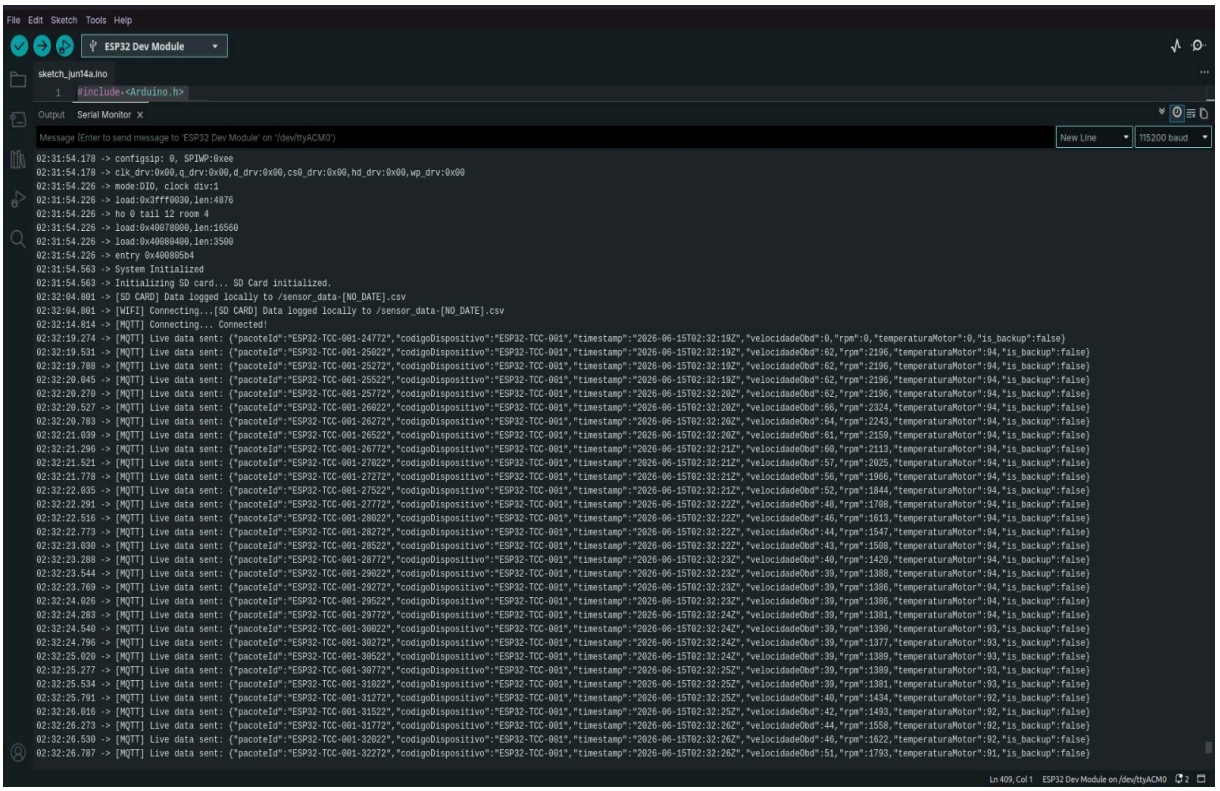


Fonte: Autoria própria (2026)

A Imagem 2 apresenta o simulador OBD/CAN em execução, responsável por representar, em ambiente controlado, a origem dos dados automotivos utilizados no trabalho. Nessa etapa, o simulador publica informações em MQTT, viabilizando a entrada dos dados no fluxo adotado para a demonstração funcional da solução.

De forma simplificada, o simulador foi utilizado para emular uma fonte de dados automotivos, organizando as leituras em mensagens no formato JSON e publicando esse conteúdo em um tópico MQTT previamente definido. Para isso, o programa mantém valores simulados de parâmetros como rotação do motor, velocidade e temperatura, atualizando-os periodicamente e convertendo-os em uma estrutura textual compatível com o fluxo de telemetria do sistema. Assim, essas informações passam a ser disponibilizadas de maneira estruturada ao ESP32, reproduzindo em ambiente controlado a etapa inicial de alimentação da arquitetura sem exigir conexão direta com um automóvel real.

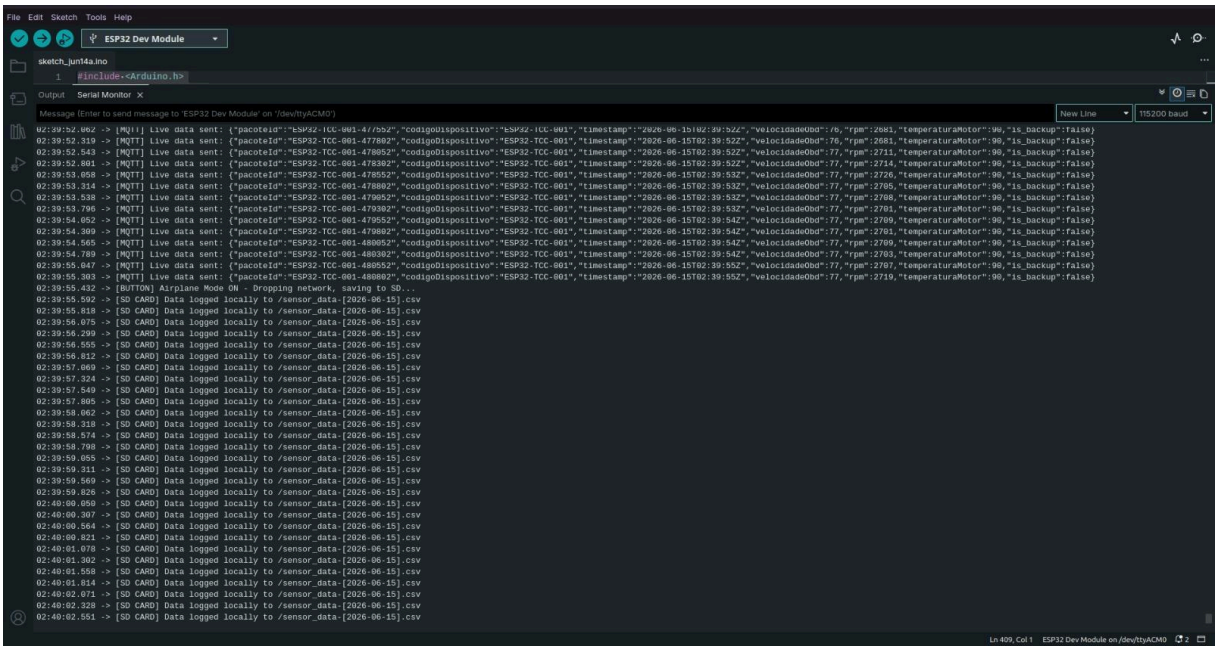
Imagem 3 – ESP32 Etapa de recepção



Fonte: Autoria própria (2026)

A Imagem 3 evidencia o funcionamento do ESP32 na etapa de recepção dos dados simulados. O registro mostra o comportamento do módulo ao estabelecer conexão com a rede, conectar-se ao *broker* MQTT e iniciar o tratamento das mensagens recebidas, constituindo o primeiro estágio embarcado do fluxo de telemetria implementado.

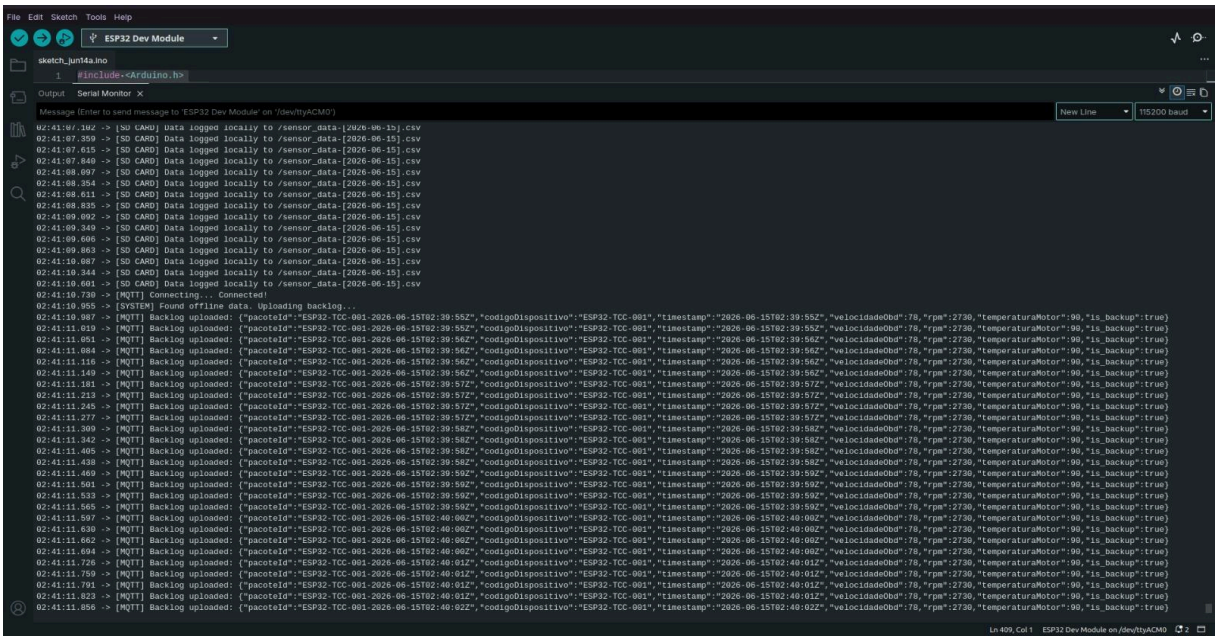
Imagem 4 – ESP32 Etapa de gravação



Fonte: Autoria própria (2026)

A Imagem 4 apresenta a evidência do armazenamento local dos dados no cartão SD. Esse comportamento corresponde ao mecanismo de contingência adotado no protótipo, no qual os pacotes são registrados localmente quando necessário, preservando data e horário das leituras para posterior sincronização.

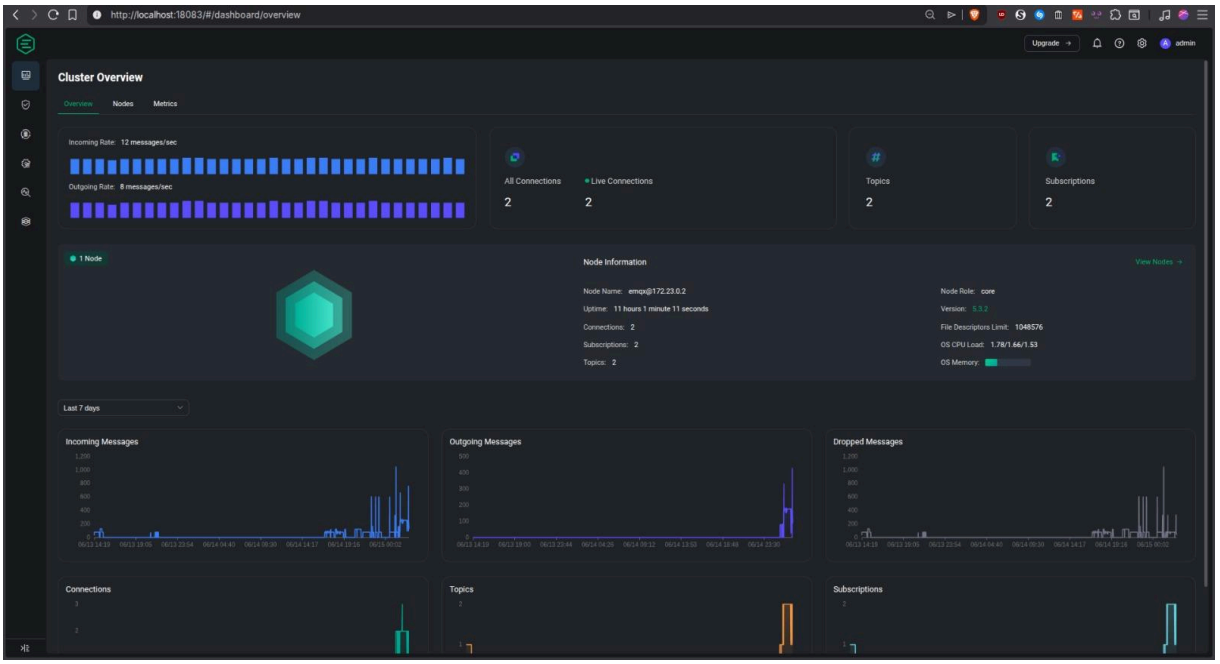
Imagem 5 – ESP32 fluxo de republicação dos dados



Fonte: Autoria própria (2026)

A Imagem 5 evidencia a etapa de republicação dos dados pelo ESP32. Nesse momento, após o restabelecimento da conectividade, o módulo envia ao *broker* MQTT os registros anteriormente armazenados localmente e, em seguida, retoma o envio em tempo real das novas leituras recebidas no sistema.

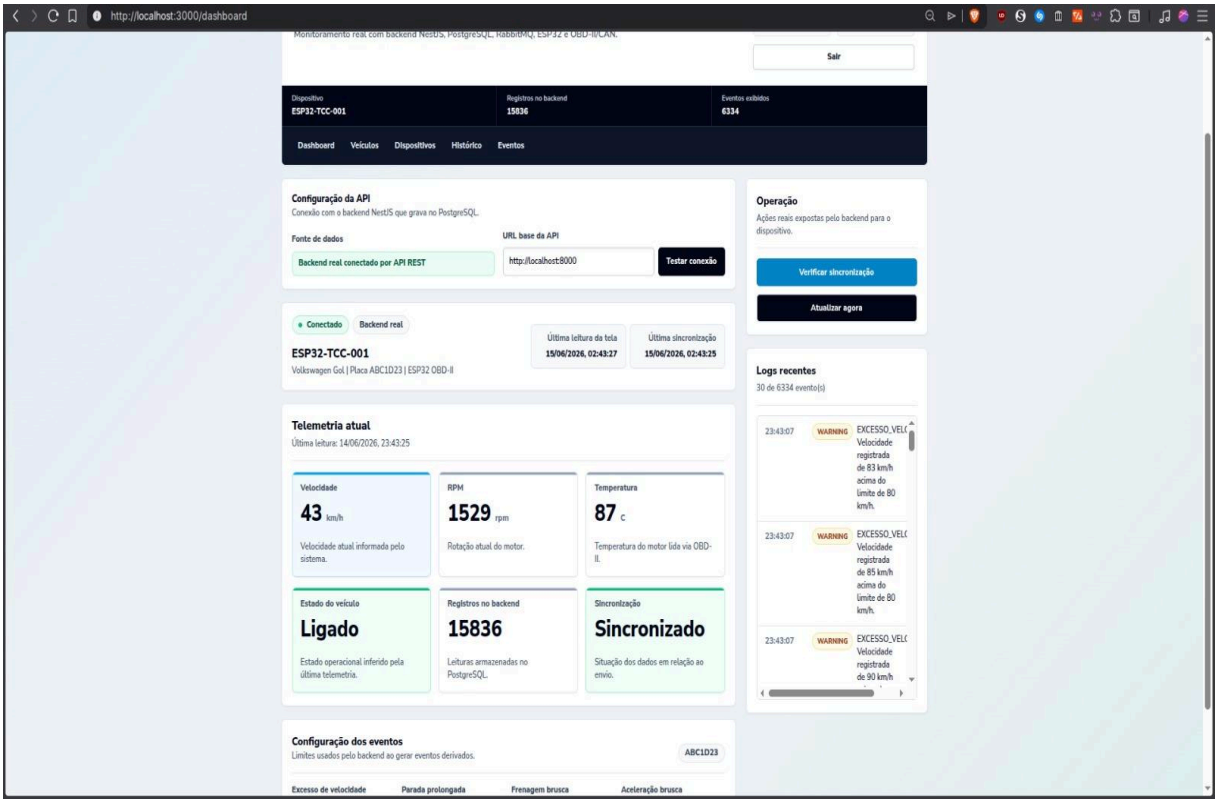
Imagem 6 – Recepção e processamento no *backend*



Fonte: Autoria própria (2026)

A Imagem 6 apresenta evidências da recepção e do encaminhamento dos dados na infraestrutura de mensageria e processamento. O registro mostra o acompanhamento dos tópicos no ambiente do *broker*, demonstrando a circulação das mensagens que sustentam o processamento posterior no *backend.*

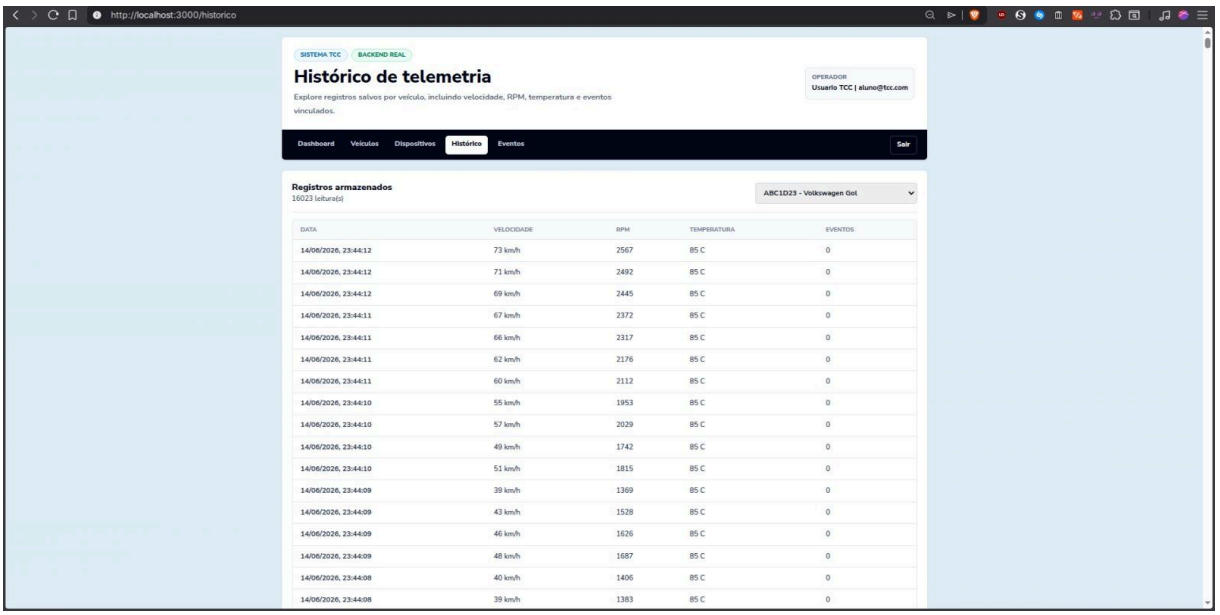
Imagem 7 – Visualização da telemetria e dos eventos no *frontend*



Fonte: Autoria própria (2026)

A Imagem 7 apresenta a interface *web* em funcionamento, exibindo informações de telemetria e eventos processados pela aplicação. Essa evidência demonstra a integração entre *frontend e backend,* permitindo visualizar, em ambiente de uso, os dados já tratados e disponibilizados ao usuário.

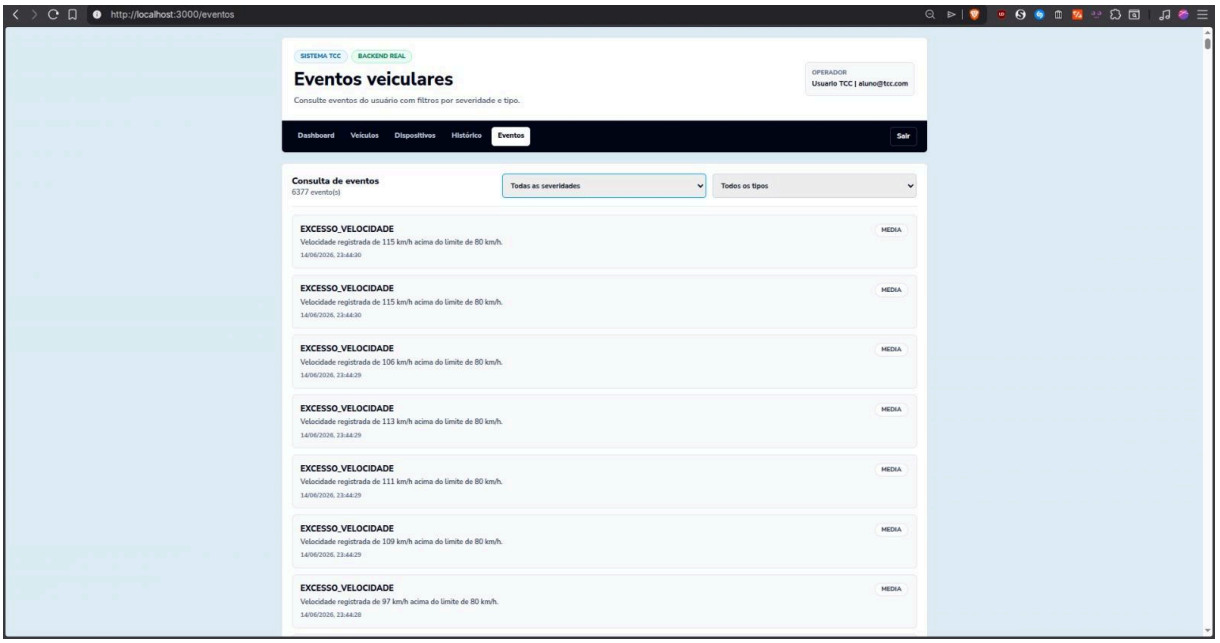
Imagem 8 – Histórico de Telemetria



Fonte: Autoria própria (2026)

A Imagem 8 apresenta a tela de histórico de telemetria, na qual os registros armazenados podem ser consultados de forma detalhada. Essa funcionalidade evidencia a persistência dos dados e a possibilidade de acompanhamento retrospectivo das leituras recebidas pelo sistema.

Imagem 9 – Consulta de eventos registrados



Fonte: Autoria própria (2026)

A Imagem 9 apresenta a tela de consulta de eventos registrados, permitindo visualizar ocorrências derivadas automaticamente a partir das regras de negócio implementadas no *backend*. Essa evidência demonstra que a solução não apenas armazena telemetria, mas também transforma leituras em informações interpretáveis para acompanhamento operacional.

Em síntese, os componentes e tecnologias empregados foram selecionados com base em compatibilidade com o objetivo do trabalho, viabilidade de implementação em contexto acadêmico e equilíbrio entre simplicidade, custo e capacidade funcional. As evidências apresentadas ao longo desta seção demonstram que a solução foi efetivamente construída como um sistema integrado, articulando *hardware* embarcado, mensageria, processamento, armazenamento e visualização. Desse modo, o tópico cumpre a função de caracterizar tecnicamente a arquitetura desenvolvida e de documentar sua implementação, servindo como base para a compreensão da validação funcional apresentada na seção específica de experimento e coleta de dados.

Quadro 8 - Decisões de projeto e justificativas

Elemento	Alternativa considerada	Justificativa da escolha
ESP32	Raspberry Pi e outros SBCs	Menor custo e consumo, conectividade integrada e capacidade suficiente para leitura serial, empacotamento e publicação da telemetria sem sistema operacional completo.
EMQX	Envio HTTP direto ou broker mais simples	Permite receber as publicações MQTT do gateway com baixo acoplamento e sustenta, de forma simples, a etapa de transmissão da telemetria no protótipo.
PostgreSQL	Banco não relacional ou armazenamento local apenas	Oferece integridade relacional, consultas históricas, índices e boa aderência ao Prisma para o vínculo entre usuários, veículos, dispositivos, telemetria e eventos.
NestJS	Servidor ad hoc	Favorece modularização, DTOs, validação, separação entre controllers e services e documentação mais clara do backend.
Next.js	Interface estática simples	Facilita a composição da interface homem-máquina em componentes reutilizáveis e a demonstração do painel web no contexto do TCC.
Prisma ORM	Acesso direto ao banco com consultas SQL ou implementação manual da camada de armazenamento	A adoção do Prisma ORM justificou-se por sua capacidade de acelerar o desenvolvimento, facilitar a modelagem dos dados e simplificar a integração entre o banco PostgreSQL e o backend em NestJS, reduzindo a complexidade da camada de acesso a dados no contexto acadêmico do trabalho.

Fonte: Elaborado pelos autores (2026)

4.3. Processos de Coleta de Dados

A coleta de dados foi realizada em ambiente acadêmico e experimental, a partir da observação técnica do funcionamento do protótipo durante sua execução. Diferentemente de pesquisas baseadas em questionários, entrevistas ou levantamento de percepção, os dados deste estudo foram obtidos diretamente da operação da solução desenvolvida. Para isso, consideraram-se as mensagens publicadas pelo simulador OBD/CAN, os registros organizados pelo módulo embarcado, as mensagens trafegadas no *broker* MQTT, os dados

armazenados no banco de dados e os resultados exibidos na interface *web*.

No que se refere à origem e à padronização dos dados observados, o Quadro 8 apresenta os PIDs OBD-II empregados no protótipo, bem como sua correspondência com a base normativa utilizada. Observa-se que os parâmetros selecionados velocidade do veículo, rotação do motor e temperatura do motor foram escolhidos por sua relevância para a composição de uma telemetria básica e por sua aderência às especificações da ISO 15031-5 e da SAE J1979. Além de sustentarem a legitimidade técnica da coleta, esses parâmetros foram suficientes para demonstrar, no contexto do trabalho, a captura de dados veiculares padronizados e sua utilização no processamento das regras implementadas.

Quadro 9 - PIDs OBD-II utilizados e relação com a padronização

PID	**Parâmetro**	**Cálculo /resposta**	**Base normativa**	**Uso no protótipo**
0x0C	Rotação do motor (RPM)	((A*256)+B)/ 4	ISO 15031-5 / SAE J1979	Suporta monitoramento do regime do motor e composição dos eventos.
0x0D	Velocidade do veículo	A	ISO 15031-5 / SAE J1979	É o principal parâmetro usado para excesso de velocidade, aceleração, frenagem e parada.
0x05	Temperatura do motor	A-40	ISO 15031-5 / SAE J1979	Compõe a telemetria básica e demonstra a coleta de dados operacional padronizado.

Fonte: Elaborado pelos autores (2026)

Em complemento, o Quadro 10 sintetiza os eventos automáticos e suas respectivas regras de geração no *backend*. Esse quadro é relevante para a compreensão da coleta porque explicita de que maneira os dados brutos recebidos foram posteriormente interpretados pelo sistema. Nota-se que os eventos não decorrem de inserção manual, mas da aplicação de regras objetivas sobre os parâmetros monitorados, especialmente velocidade e variação de velocidade em intervalos de tempo definidos. Dessa forma, o quadro evidencia que a coleta não se restringiu ao recebimento e armazenamento de leituras, mas abrangeu também a observação do comportamento derivado dessas informações no processamento automático da solução.

Quadro 10- Relação de eventos automáticos e respectivas regras

Evento	Regra implementada	Parâmetro padrão	Severidade
EXCESSO VELOCIDADE	Gerado quando velocidadeObd > limiteVelocidade.	80 km/h	MÉDIA
FRENAGEM BRUSCA	Gerado quando a variação de velocidade é menor ou igual a limite FrenagemBrusca em até 2 s.	20 km/h	ALTA
ACELERAÇÃO BRUSCA	Gerado quando a variação de velocidade é maior ou igual a limite Aceleração Brusca em até 3 s.	25 km/h	MÉDIA
PARADA PROLONGADA	Gerado quando o veículo permanece com velocidade zero por tempo maior ou igual a tempo ParadaLongaMinutos.	10 min	BAIXA

Fonte: Elaborado pelos autores (2026)

O processo de coleta ocorreu durante a fase de testes integrados do protótipo, quando foram monitoradas as etapas do fluxo funcional de ponta a ponta. No cenário experimental efetivamente realizado, o simulador publicou dados via MQTT, que foram recebidos pelo ESP32, reorganizados em pacotes de telemetria e reenviados ao *broker* EMQX para posterior processamento no *backend*, armazenamento em banco de dados e visualização na aplicação *web*. Assim, os dados brutos observados incluíram os pacotes transmitidos, os registros armazenados localmente para contingência, as mensagens processadas pelo *backend*, os eventos gerados automaticamente com base nas regras de negócio e os resultados apresentados na interface web.

O Quadro 11 apresenta, de forma sintética, a sequência das principais etapas de execução do trabalho. Sua inclusão nesta seção contribui para situar temporalmente o processo de construção e observação da solução, evidenciando que a coleta de dados não ocorreu de forma isolada, mas como parte de um encadeamento que envolveu revisão bibliográfica, definição arquitetural, modelagem, implementação, integração e testes. Nesse sentido, o cronograma ajuda a contextualizar a coleta dentro do percurso metodológico efetivamente seguido no desenvolvimento do protótipo.

Quadro 11 - Cronograma sintético de execução

Etapa	**Período aproximado**	**Síntese da atividade**
Revisão bibliográfica e delimitação do tema	mar. 2026	Levantamento do problema, objetivos e referências iniciais.
Levantamento de requisitos e definição arquitetural	mar. 2026 - abr. 2026	Definição dos casos de uso, requisitos e arquitetura geral.
Modelagem de dados e regras de negócio	abr. 2026	Elaboração do DER, entidades, relacionamentos e eventos automáticos.
Implementação do backend e armazenamento	abr. 2026 - maio 2026	Construção da API NestJS, Prisma e *Postgre*SQL.
Implementação da interface web	maio 2026	Desenvolvimento do painel de consulta e status.
Implementação do gateway e do simulador	maio 2026 - jun. 2026	Codificação do fluxo embarcado, leitura de PIDs e contingência em SD.
Integração e testes funcionais	jun. 2026	Validação ponta a ponta com *broker*, *backend*, banco e *frontend*.
Redação e consolidação do TCC	jun. 2026	Ajustes textuais, documentação técnica e preparação para a banca.

Fonte: Elaborado pelos autores (2026)

Após a coleta, as informações observadas foram organizadas e analisadas de acordo com sua função no sistema, o que permitiu verificar a consistência entre os dados recebidos, os registros persistidos e os resultados apresentados pela solução. Desse modo, a coleta de dados assumiu caráter técnico e funcional, voltado à verificação da coerência do fluxo implementado e à observação do comportamento integrado da arquitetura desenvolvida.

4.4. Resultados Obtidos

Os resultados obtidos indicam que o protótipo apresentou comportamento coerente com a proposta definida para o estudo, demonstrando a viabilidade técnica da arquitetura implementada em ambiente acadêmico e experimental. A execução dos testes permitiu verificar o funcionamento integrado do fluxo de captura/recepção, transmissão, processamento, armazenamento e visualização dos dados veiculares básicos considerados no

escopo da pesquisa.

No cenário experimental adotado, observou-se que o simulador OBD/CAN publicou dados representativos do domínio automotivo, os quais foram recebidos pelo módulo embarcado em ESP32, reorganizados em pacotes de telemetria e reenviados ao *broker* MQTT. Em seguida, esses dados foram processados no *backend*, persistidos em banco de dados relacional e disponibilizados na interface *web* para consulta posterior. Tal resultado demonstra que a solução desenvolvida foi capaz de sustentar, de forma funcional, o fluxo principal previsto para a arquitetura proposta.

Não vi isso. Está registrado?

Também se verificou que o sistema conseguiu manter registro histórico das leituras recebidas e gerar eventos automáticos a partir das regras de negócio estabelecidas, especialmente nos casos relacionados à velocidade e às variações de condução simuladas. Esse aspecto é relevante porque evidencia que o protótipo não apenas transporta dados entre componentes, mas também os interpreta de modo compatível com a lógica operacional definida no trabalho.

Outro resultado importante diz respeito ao uso do armazenamento local em cartão SD como mecanismo de contingência. No contexto da solução desenvolvida, essa estratégia reforça a confiabilidade básica do fluxo ao prever a preservação temporária dos dados quando necessário, permitindo sua posterior reintegração ao sistema. Ainda que a validação tenha ocorrido em ambiente controlado, essa característica contribui para aproximar o protótipo de situações práticas de operação.

Dessa forma, os resultados obtidos permitem afirmar que a solução implementada atendeu ao recorte estabelecido para a pesquisa, ao demonstrar, de maneira objetiva, a integração funcional entre módulo embarcado, mensageria, *backend*, armazenamento e interface *web*. Assim, as evidências observadas no capítulo experimental oferecem base suficiente para, nas conclusões do trabalho, confirmar a viabilidade técnica do protótipo desenvolvido e a coerência da arquitetura proposta.

5. CONCLUSÕES

Os resultados obtidos permitem concluir que a proposta desenvolvida apresentou viabilidade técnica no contexto acadêmico em que foi realizada. A arquitetura implementada demonstrou funcionamento coerente entre seus componentes, permitindo estruturar um fluxo integrado de captura/recepção, transmissão, processamento, armazenamento e visualização de dados veiculares básicos. Desse modo, considera-se que o objetivo da pesquisa foi alcançado, na medida em que o protótipo conseguiu materializar uma solução funcional de telemetria veicular com características de caixa-preta, compatível com o recorte e com as condições estabelecidas para o estudo.

Também se conclui que a hipótese formulada foi confirmada. Os resultados evidenciaram que foi possível desenvolver, em contexto acadêmico, uma solução de baixo custo capaz de organizar o ciclo de tratamento dos dados automotivos por meio da integração entre módulo embarcado, mensageria MQTT, *backend*, banco de dados e interface *web*. A solução mostrou-se apta a manter histórico de registros, apoiar a consulta posterior das informações e produzir eventos automáticos a partir de regras previamente definidas, o que reforça sua utilidade como protótipo de monitoramento e registro operacional.

No que se refere ao alcance experimental do estudo, é importante registrar que a validação não foi realizada com uso efetivo do módulo CAN em veículo real. Embora a arquitetura prevista contemple essa possibilidade, os testes práticos ocorreram com simulador OBD/CAN, empregado como recurso de representação controlada da origem dos dados automotivos. Ainda assim, esse procedimento se mostrou metodologicamente compatível com a proposta do trabalho, pois o simulador adotado reproduziu a lógica de parâmetros padronizados do domínio OBD-II, com uso de identificadores e respostas compatíveis com a estrutura prevista em normas como a ISO 15031-5 e a SAE J1979, especialmente para velocidade, rotação do motor e temperatura do motor. Assim, embora não se trate de validação automotiva em campo, o ambiente experimental preservou aderência conceitual ao modelo de aquisição de dados considerado pela pesquisa.

Diante disso, conclui-se que o trabalho atende ao escopo proposto e oferece contribuição acadêmica e prática ao demonstrar a viabilidade de integração entre diferentes componentes de hardware e software no contexto da telemetria veicular. Mais do que uma

solução isolada, o protótipo desenvolvido evidencia a possibilidade de articulação entre sistemas embarcados, mensageria, persistência de dados e interface *web* em uma aplicação coerente com demandas contemporâneas de monitoramento e registro técnico. As limitações do estudo não invalidam essa contribuição, mas delimitam seu alcance e indicam a necessidade de continuidade em investigações futuras.

5.1. Trabalhos Futuros

Como limitações do estudo, destaca-se que a pesquisa permaneceu restrita ao ambiente acadêmico e experimental, não abrangendo testes prolongados em frota real, homologação automotiva, geolocalização, integração com aplicações móveis ou recursos avançados de análise preditiva. Também não foi realizada, nesta etapa, avaliação com usuários ou especialistas por meio de instrumentos de pesquisa. Tais limitações delimitam o alcance do trabalho, mas não comprometem sua contribuição como demonstração de viabilidade técnica.

Como perspectiva de continuidade, recomenda-se a ampliação do protótipo para contextos de uso mais próximos de cenários reais, com testes prolongados em veículos e observação do comportamento do sistema em condições operacionais menos controladas. Mostra-se pertinente, ainda, a inclusão de novas funcionalidades, como geolocalização, geração de relatórios, monitoramento em tempo real e mecanismos mais robustos de autenticação e segurança.

No plano metodológico, recomenda-se a realização de pesquisa por formulários - questões semi-estruturadas, com usuários, avaliadores ou especialistas da área, com o objetivo de analisar percepções sobre utilidade, clareza, aplicabilidade e usabilidade da solução desenvolvida. A adoção da pesquisa pode complementar a validação funcional já realizada, agregando uma dimensão avaliativa centrada na percepção dos participantes.

Além disso, a solução pode evoluir por meio da ampliação de sua capacidade analítica, com incorporação de tratamento de séries históricas, identificação de padrões de uso e recursos voltados à manutenção preditiva e à análise inteligente de eventos. Esses avanços podem aprofundar o potencial do sistema como ferramenta de monitoramento e apoio à decisão.

REFERÊNCIAS

AMMAR, Mohd et al. *Significant applications of smart materials and Internet of Things (IoT) in the automotive industry*. *Materials Today: Proceedings*, v. 68, n. 2, 2022. DOI: 10.1016/j.matpr.2022.07.180.

DHALL, R.; SOLANKI, V. K. *An IoT based predictive connected car maintenance approach*. *International Journal of Interactive Multimedia and Artificial Intelligence*, v. 4, n. 3, p. 16-27, 2017.

GHOSH, Raj Krishan et al. *Intelligent IoT for automotive industry 4.0: challenges, opportunities, and future trends*. In: GHOSH, Uttam et al. *Intelligent Internet of Things for Healthcare and Industry*. Cham: Springer, 2022. p. 327-352. DOI: 10.1007/978-3-030-81473-1_16.

GIL, Antonio Carlos. *Como elaborar projetos de pesquisa*. 4. ed. São Paulo: Atlas, 2002.

GROSSMANN, Jurgen; SERBANESCU, Diana; SCHIEFERDECKER, Ina Kathrin. *Test specification and test methodology for embedded systems in automobiles*. [S. l.: s. n.], 2008. Disponível em: https://www.researchgate.net/publication/228538349_Test_Specification_and_Test_Methodology_for_Embedded_Systems_in_Automobiles. Acesso em: 30 mar. 2026.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO 15031-5:2015. Road vehicles: communication between vehicle and external equipment for emissions-related diagnostics. Part 5: emissions-related diagnostic services*. Geneva: ISO, 2015.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. *Fundamentos de metodologia científica*. 5. ed. São Paulo: Atlas, 2003.

MUTHIYA, Solomon Jenoris et al. *Application of Internet of Things (IoT) in the automotive industry*. In: RAJASEKAR, R.; MOGANAPRIYA, C.; HARIKRISHNA KUMAR, M.; SATHISH KUMAR, P. (ed.). *Integration of Mechanical and Manufacturing Engineering with IoT: a digital transformation*. Hoboken: Wiley, 2023. p. 115-139.

OASIS. *MQTT version 5.0*. [S. l.]: OASIS Standard, 2019. Disponível em: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf. Acesso em: 9 jun. 2026.

SAE INTERNATIONAL. *J1979: E/E diagnostic test modes*. Warrendale: SAE International, 2014.

SILVA, Antonio Vitor da. *Desenvolvimento de sistema embarcado para telemetria veicular*. 2025. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) - Universidade Federal de Pernambuco, Recife, 2025. Disponível em: https://repositorio.ufpe.br/handle/123456789/67373. Acesso em: 9 jun. 2026.

SONKO, Sedat et al. *The evolution of embedded systems in automotive industry: a global review*. *World Journal of Advanced Research and Reviews*, v. 21, n. 2, p. 96-104, 2024. DOI: 10.30574/wjarr.2024.21.2.0420.

VANITHA, M. et al. *A smart IoT based black-box system for automobiles*. *Journal of Physics: Conference Series*, v. 2484, n. 1, p. 012052, 2023.

VEITAS, V. K.; DELAERE, S. *In-vehicle data recording, storage and access management in autonomous vehicles*. arXiv:1806.03243 [cs.CY], 2018. Disponível em: https://arxiv.org/abs/1806.03243. Acesso em: 28 mar. 2026.