UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE SISTEMAS ELETRÔNICOS

JOÃO PAULO BEDRETCHUK

PROPOSTA E DESENVOLVIMENTO DE UM HARDWARE EMBARCADO COM SUPORTE À COMPUTAÇÃO DE BORDA PARA AQUISIÇÃO, ANÁLISE E PROCESSAMENTO AVANÇADO DE DADOS VEICULARES

DISSERTAÇÃO DE MESTRADO

Joinville
2023

João Paulo Bedretchuk

PROPOSTA E DESENVOLVIMENTO DE UM HARDWARE EMBARCADO COM SUPORTE À COMPUTAÇÃO DE BORDA PARA AQUISIÇÃO, ANÁLISE E PROCESSAMENTO AVANÇADO DE DADOS VEICULARES

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Sistemas Eletrônicos da Universidade Federal de Santa Catarina para a obtenção do título de Mestre em Engenharia de Sistemas Eletrônicos.
Orientador: Prof. Anderson Wedderhoff Spengler, Dr.

Joinville
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Bedretchuk, João Paulo
Proposta e desenvolvimento de um hardware embarcado com suporte à computação de borda para aquisição, análise e processamento avançado de dados veiculares / João Paulo Bedretchuk ; orientador, Anderson Wedderhoff Spengler, 2023.
103 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Campus Joinville, Programa de Pós-Graduação em Engenharia de Sistemas Eletrônicos, Joinville, 2023.

Inclui referências.

1. Engenharia de Sistemas Eletrônicos. 2. Desenvolvimento de hardware. 3. Aquisição de dados veiculares. 4. Unidade de Controle Eletrônica. 5. Testes veiculares. I. Spengler, Anderson Wedderhoff. II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Engenharia de Sistemas Eletrônicos. III. Título.

João Paulo Bedretchuk

Proposta e Desenvolvimento de um Hardware Embarcado com Suporte à Computação de Borda para Aquisição, Análise e Processamento Avançado de Dados Veiculares

O presente trabalho em nível de Mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Giovani Gracioli, Dr.
Universidade Federal de Santa Catarina

Prof. Rodrigo Moreira Bacurau, Dr.
Universidade Estadual de Campinas

Prof. Anderson Wedderhoff Spengler, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de Mestre em Engenharia de Sistemas Eletrônicos.

Documento assinado digitalmente
Lucas Weihmann
Data: 01/09/2023 10:27:34-0300
CPF: ***.676.080-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Lucas Weihmann, Dr.
Coordenador do Programa

Documento assinado digitalmente
Anderson Wedderhoff Spengler
Data: 31/08/2023 14:24:26-0300
CPF: ***.275.659-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Anderson Wedderhoff Spengler, Dr.
Orientador

Joinville, 03 de Agosto de 2023.

AGRADECIMENTOS

Agradeço aos meus pais, Jessé e Roseli, que sempre fizeram todo o possível para me proporcionar as melhores oportunidades para o meu crescimento pessoal e profissional, por seu amor incondicional e por estarem sempre comigo, me apoiando e incentivando, e à Larissa, por estar ao meu lado em todos os momentos.

Agradeço aos professores Anderson Wedderhoff Spengler e Giovani Gracioli, por todo o suporte e orientação durante a pesquisa, ao Sergio Arribas Garcia e ao Thiago Nogiri Igarashi, pela parceria ao longo do desenvolvimento do projeto IASE e pela grande troca de conhecimentos, e aos demais amigos e colegas do Laboratório de Integração de Software/Hardware da UFSC que de alguma forma contribuíram para o desenvolvimento do trabalho.

Por fim, agradeço à Renault do Brasil pelo fornecimento do carro utilizado e por todo o suporte oferecido, e à Fundação de Desenvolvimento da Pesquisa (FUNDEP), Programa Rota 2030 - Linha V 27192.02.01/2020.09-0 pelo financiamento da pesquisa.

“Education is the most powerful weapon we can use to change the world.”
(Nelson Mandela, 2003)

RESUMO

O processo de desenvolvimento de veículos realizado pela indústria automotiva vem, ao longo dos anos, sofrendo diversas modificações no que diz respeito a fatores como implementação de novas tecnologias, atendimento a demandas específicas dos consumidores e a criação e alteração de normas regulamentadoras, tanto no sentido de padronização de processos industriais como de restrições relacionadas à emissão de gases provenientes de motores a combustão. Por consequência desses fatores, as fases de testes de protótipos veiculares têm se tornado cada vez mais importantes no processo de produção. As Unidades de Controle Eletrônicas (ECU) veiculares são parte essencial desses testes, pois são responsáveis por monitorar e controlar todos os dispositivos elétricos e eletrônicos presentes nos veículos. Nesse sentido, para que seja possível realizar adequadamente os processos de testagem dos protótipos, o uso de ferramentas capazes de adquirir dados e realizar a calibração de parâmetros dessas ECUs é indispensável. Apesar da existência de diversas soluções comerciais para esse fim, em geral, esses equipamentos se limitam a funcionalidades essenciais do processo de testagem e calibração. A fim de propor uma alternativa a esses equipamentos, o Sistema Inteligente de Aquisição e Análise de Dados de ECUs (IASE) foi concebido. O presente trabalho descreve o desenvolvimento do hardware de aquisição, análise e processamento avançado de dados veiculares, visando a implementação do IASE e outros sistemas. Esse hardware pretende habilitar a execução de aplicações que envolvam computação de borda, computação em nuvem, inteligência artificial, fusão de dados, e outras tecnologias de processamento de dados capazes de aprimorar a fase de testes, calibração e validação de protótipos veiculares. Dentre as principais características desse equipamento, destacam-se a capacidade de comunicação via barramento CAN com ECUs veiculares, a aquisição de dados de sensores integrados ou não aos veículos, o uso de redes Bluetooth para a comunicação com dispositivos móveis e o uso de tecnologia 4G para transferência de dados utilizando IoT. Ao longo do texto, são apresentados o desenvolvimento do hardware, com base nas especificações do IASE, a implementação de um sistema de detecção de knock noise para validar suas funcionalidades e os testes de desempenho do equipamento desenvolvido. Os testes realizados demonstraram que o hardware desenvolvido é capaz de atender a todas as funcionalidades do processo de testes e validação de veículos, com desempenho similar aos equipamentos comerciais. Além disso, possibilita a implementação de novas tecnologias (e.g. 4G, GPS e Bluetooth) e apresenta um custo de produção até 8 vezes menor do que o custo dos sistemas comerciais mais utilizados pelos fabricantes. Adicionalmente, o sistema apresenta os recursos necessários para o desenvolvimento de aplicações que envolvam computação de borda e computação em nuvem.

Palavras-chave: Hardware de aquisição de dados veiculares. Unidade de Controle Eletrônica (ECU). Sistema de testes de veículos automotivos. Pré-ignição

ABSTRACT

The vehicle development process carried out by the automotive industry has undergone several modifications over the years regarding factors such as the implementation of new technologies, meeting specific consumer demands, and the creation and alteration of regulatory standards, both in terms of industrial process standardization and restrictions related to emissions from combustion engines. As a consequence of these factors, the phases of vehicle prototype testing have become increasingly important in the production process. Vehicle Electronic Control Units (ECUs) are an essential part of these tests, as they are responsible for monitoring and controlling all electrical and electronic devices present in the vehicles. Therefore, in order to adequately perform prototype testing processes, the use of tools capable of acquiring data and calibrating parameters of these ECUs is indispensable. Despite the existence of many commercial solutions for this purpose, these equipment generally limit themselves to essential functionalities of the testing and calibration process. In order to propose an alternative to these equipment, the Intelligent Acquisition and Analysis System for ECUs (IASE) was conceived. This work describes the development of the hardware for acquisition, analysis, and advanced processing of vehicular data, aiming at the implementation of IASE and other systems. This hardware aims to enable the execution of applications involving edge computing, cloud computing, artificial intelligence, data fusion, and other data processing technologies capable of enhancing the testing, calibration, and validation phase of vehicle prototypes. Among the main features of this equipment, the ability to communicate via the CAN bus with vehicle ECUs, the acquisition of data from sensors integrated or not into the vehicles, the use of Bluetooth networks for communication with mobile devices, and the use of 4G technology for data transfer using IoT stand out. Throughout the text, the development of the hardware based on IASE specifications, the implementation of a knock noise detection system to validate its functionalities, and the performance tests of the developed equipment are presented. The conducted tests demonstrated that the developed hardware is capable of meeting all the functionalities of the vehicle testing and validation process, with performance similar to commercial equipment. Furthermore, it enables the implementation of new technologies (e.g. 4G, GPS, and Bluetooth) and has a production cost up to 8 times lower than the cost of the most widely used commercial systems by manufacturers. Additionally, the system provides the necessary resources for the development of applications involving edge computing and cloud computing.

Keywords: Vehicle data acquisition hardware. Electronic Control Unit (ECU). Test system for automotive vehicles. Knock noise

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

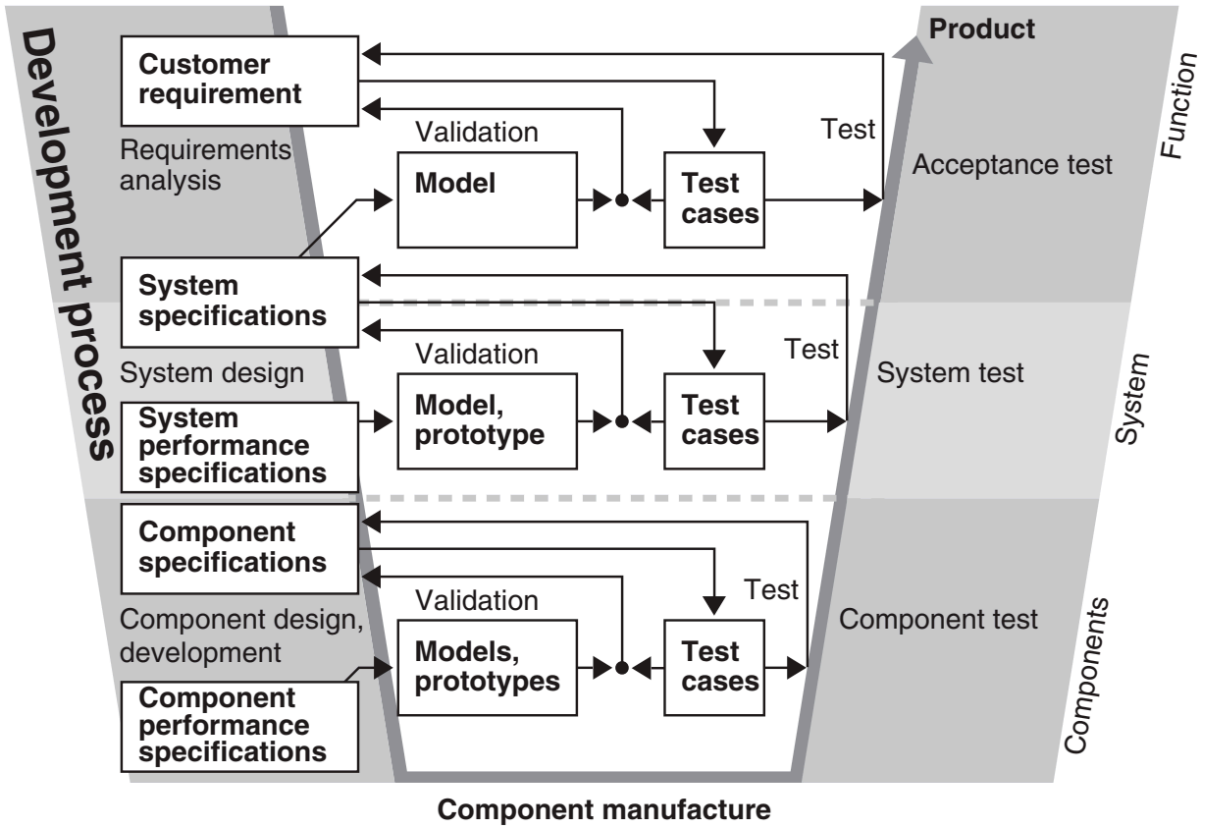
O processo de produção de veículos na indústria automobilística segue o modelo de produção em V, uma abordagem amplamente adotada para gerenciamento e organização da produção. Esse modelo consiste em um conjunto específico de fases de desenvolvimento, conforme ilustra a Figura 1.1 (DIETSCHE; KONRAD et al., 2022). O modelo em V proporciona uma estrutura eficiente para o fluxo de trabalho, garantindo a integração adequada das etapas, desde o projeto inicial até a fabricação final do veículo.

Dentre as principais fases do processo de produção de novos veículos, destacam-se: especificação de requisitos, desenvolvimento e simulação de sistemas, seleção e avaliação de componentes, validação e integração dos veículos (WEBER, 2009). Uma etapa fundamental nesse processo é a de testes e validação dos veículos, que ocorre na fase final do modelo de produção em V. Nessa etapa, o protótipo do novo veículo é submetido a testes em condições reais de funcionamento, enquanto uma ampla quantidade de informações é coletada das Unidades de Controle Eletrônico (ECUs). Esse procedimento é conhecido como teste de rodagem na indústria automobilística. Os dados obtidos durante os testes de rodagem fornecem todas as informações necessárias para verificar e validar o desempenho do trem de força, a calibração das ECUs, os níveis de emissões e outros parâmetros do veículo.

Devido à crescente demanda dos consumidores por veículos mais eficientes, seguros, potentes e com mais tecnologias embarcadas, além da intensificação das normas regulamentadoras relacionadas a emissões de CO2 e da complexidade dos novos sistemas mecânicos desenvolvidos para atender tais demandas, o desenvolvimento de veículos modernos tem exigido testes extensivos de verificação e validação, visando garantir o cumprimento de todos os requisitos, atingir a qualidade desejada e atender a expectativa dos consumidores (DU; STERPONE, 2016; ANDERT et al., 2016; GUSE et al., 2020; KÖTTER et al., 2018). Para a realização desses testes, grande parte dos fabricantes de veículos têm utilizado um conjunto de equipamentos e tecnologias (i.e, hardwares e softwares específicos para a aquisição de dados) fornecidos por empresas europeias e norte americanas. As principais empresas atuantes nesse ramo são: ETAS, Vector, ATI e Bosh. As soluções ofertadas por essas empresas incluem equipamentos de aquisição de dados para diferentes barramentos de comunicação veiculares, permitindo a captação e o armazenamento local das informações do veículo durante os testes de rodagem.

Apesar de atender a demanda atual dos fabricantes, os produtos ofertados nem sempre apresentam os recursos tecnológicos mais atuais e adequados para os testes veiculares da indústria automotiva brasileira. Equipamentos que possibilitem a integração do processo de captura de dados veiculares com tecnologias como computação

Figura 1.1 – Visão geral do modelo de produção em V.



Fonte: Dietsche, Konrad et al. (2022)

de borda, computação em nuvem, internet das coisas (IoT) e algoritmos de inteligência artificial (IA) ainda são escassos no mercado. Ademais, a grande maioria dos equipamentos ofertados atualmente ainda depende do uso de um computador pessoal ou *tablet* de alto desempenho para capturar e armazenar os dados coletados, permitindo o processamento dessas informações somente após o término do ciclo completo de um teste de rodagem. Poucas são as soluções existentes que possibilitam o processamento dos dados coletados em tempo real (VECTOR, 2023b), ou que utilizam tecnologias sem fio (VECTOR, 2021a) ou hardwares embarcados (ATI, 2023; VECTOR, 2023a) capazes de gerenciar todo o processo, e nenhuma solução atual é apresentada pelos principais fabricantes que englobe todas as características citadas.

Além das limitações apresentadas por grande parte das soluções comerciais atuais, o custo de aquisição de seus equipamentos é relativamente alto, podendo custar entre 5 mil e 20 mil dólares, sem considerar custos extras de manutenção, licenças de software e treinamentos. Fatores como esses podem limitar o processo de testagem e a produção de veículos e fazer com que o custo de desenvolvimento e o preço final dos carros se elevem.

A fim de evitar que os equipamentos e a tecnologia utilizados para o teste de

sistemas automotivos sejam fatores limitantes para a produção de novos veículos e prevenir o aumento de preços desses automóveis, varias estratégias têm sido propostas e aplicadas, como *hardware-in-the-loop* (HiL), *engine-in-the-loop* (EiL) e, mais recentemente, *X-in-the-loop* (XiL) (FAGHANI; ANDRIC; SJOBLOM, 2018; GUSE et al., 2020; KÖTTER et al., 2018). Essas técnicas utilizam tecnologias de simulação parcial do veículo, descartando a necessidade de colocar o automóvel em condições reais de rodagem antes de garantir que todos os equipamentos estejam funcionando corretamente. Apesar disso, os testes de calibração e validação em condições reais não podem ser dispensados devido à sua importância para assegurar a operacionalidade total, a calibração de todos os parâmetros, os níveis de emissão de CO2, e a segurança e eficiência do veículo.

Considerando a necessidade da realização de testes de rodagem para a calibração e validação dos veículos, bem como o custo dos equipamentos e tecnologias atuais disponíveis para esse fim, o Sistema Inteligente de Aquisição e Análise de Dados de ECUs (IASE) tem sido desenvolvido em um trabalho conjunto entre as equipes de pesquisa do Laboratório de Integração de Software e Hardware (LISHA) e a empresa francesa Renault, o qual é financiado pelo Programa Rota 2030, sendo este último gerenciado pela Fundação de Desenvolvimento da Pesquisa (FUNDEP). O sistema é projetado para extrair as informações necessárias dos sensores do veículo por meio de sua ECU em tempo real, processar os dados recebidos e enviá-los a um servidor de IoT, permitindo que as informações sejam recebidas pelos fabricantes quase instantaneamente. Após os dados do veículo serem recebidos pelo servidor, o sistema é capaz de processá-los aplicando algoritmos de IA para detectar anomalias e enviar todas as informações necessárias para a equipe de especialistas responsável pela análise do veículo.

Sendo parte essencial do projeto IASE, o hardware desenvolvido e apresentado neste trabalho habilita a implementação de tecnologias como IoT, IA, computação de borda e computação em nuvem simultaneamente com a captação de dados veiculares via barramentos de comunicação serial para ECUs ou sensores específicos, integrados ou não aos veículos de teste. Dessa maneira, o hardware deve possibilitar a integração de características que vão além do escopo do IASE, mantendo o baixo custo de produção do equipamento. Utilizando essas tecnologias, espera-se proporcionar aos fabricantes maior eficiência e agilidade no processo de testagem e validação de veículos, assim como melhores ferramentas para análises de condições específicas dos veículos e algoritmos inteligentes para detecção de falhas e automatização de tarefas.

Adicionalmente, para demonstrar a capacidade do hardware desenvolvido com relação a captação de dados diretamente de sensores específicos dos veículos, o presente trabalho expõe o desenvolvimento de um sistema de detecção de *knock noise*, baseado na aquisição de dados de um sensor piezoelétrico, presente na estrutura do

motor do veículo de testes. Esse sistema demonstra que o hardware é capaz de utilizar computação de borda para realizar a captura e o processamento de grandes quantidades de dados sensoriais paralelamente à execução do *firmware* do IASE.

Considerando as tecnologias disponíveis atualmente no mercado para a captação e processamento de dados veiculares, os requisitos e funcionalidades do IASE e a implementação de novas tecnologias que acelerem a testagem e validação de veículos, os objetivos do presente trabalho são expostos a seguir.

1.1 OBJETIVOS

Este trabalho tem como objetivo geral desenvolver um hardware para aquisição de dados de protótipos veiculares durante testes de rodagem, visando possibilitar a aquisição, o processamento e o *upload* de dados veiculares em tempo real para um servidor de IoT, e habilitar a implementação de aplicações de computação de borda e computação em nuvem que possam aprimorar essa fase do processo de desenvolvimento de veículos. Para alcançar o objetivo geral, os seguintes objetivos específicos são propostos:

1. Definir os requisitos de hardware com base nas funcionalidades básicas e complementares do sistema, tendo em vista a possível ampliação de suas aplicações.
2. Elaborar o projeto do hardware com base nos requisitos definidos, incluindo a seleção de componentes eletrônicos, o desenvolvimento do esquemático e da placa de circuito impresso.
3. Produzir o hardware projetado e conduzir testes funcionais abrangentes para garantir o seu correto funcionamento.
4. Implementar um sistema de detecção de *knock noise* para avaliação e demonstração da capacidade do hardware.
5. Realizar medições de desempenho e avaliar a capacidade do hardware desenvolvido, identificando os gargalos do sistema e seus diferencias com relação aos equipamentos comerciais.

1.2 ORGANIZAÇÃO DO TEXTO

O restante deste documento está organizado da seguinte forma: o Capítulo 2 apresenta uma revisão dos fundamentos teóricos e dos trabalhos relacionados ao sistema proposto. O Capítulo 3 expõe os procedimentos de teste, validação e calibração de protótipos veiculares. O Capítulo 4 descreve os materiais e métodos utilizados para o desenvolvimento do trabalho. No Capítulo 6 são apresentados os resultados e discussões e, por fim, no Capítulo 7 são descritas as conclusões.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais fundamentos teóricos necessários para o adequado desenvolvimento do trabalho. Inicialmente, aborda-se uma revisão bibliográfica a respeito da aplicação e funcionamento de ECUs veiculares, assim como os métodos de aquisição de dados utilizados para capturar informações desses dispositivos. Na sequência, são apresentados estudos referentes aos sistemas comerciais utilizados atualmente para a aquisição de dados. Por fim, são descritos os princípios e definições básicos do fenômeno de *knock noise* e apresentada uma revisão dos trabalhos relacionados.

2.1 UNIDADE DE CONTROLE ELETRÔNICA (ECU)

Ao longo dos anos, os projetos desenvolvidos pela indústria automobilística têm se tornado cada vez mais sofisticados e complexos. Diversos fatores contribuíram para isso, como a implementação de sistemas de assistência ao condutor (ADAS), o uso de novas variáveis de controle e o aprimoramento dos sistemas mecânicos dos veículos modernos, os exigentes requisitos de desempenho dos fornecedores, fabricantes e consumidores, e as restrições relacionadas ao consumo de combustível e emissões de CO2 (ISERMANN, 2022).

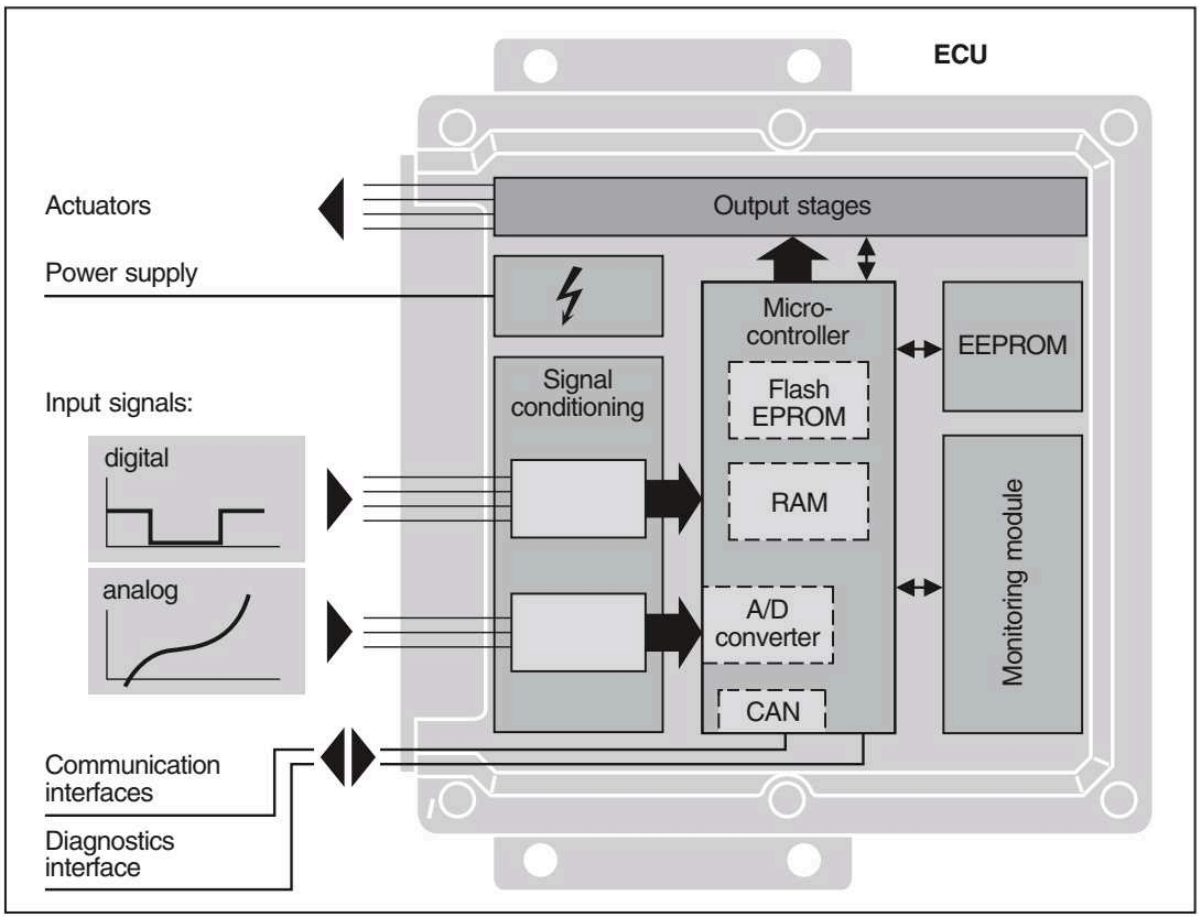
Para atender à contínua evolução dos novos projetos veiculares, os sistemas eletrônicos utilizados para o seu desenvolvimento também sofreram mudanças. Durante as três últimas décadas, a complexidade desses sistemas aumentou exponencialmente, permitindo aos fabricantes de veículos o desenvolvimento de avançados sistemas de controle automotivos. Atualmente, os sistemas eletrônicos veiculares (*E/E-Systems*) são compostos por uma infinidade de sensores, atuadores, chaves, botoeiras, redes de comunicação e principalmente Unidades de Controle Eletrônicas (ECUs) capazes processar e controlar as mais diversas funcionalidades de um veículo (PUPALA; SHUKLA, 2018).

Uma ECU pode ser definida como um dispositivo eletrônico embarcado desenvolvido para realizar o processamento de dados, comunicação e controle de sistemas automotivos a fim de otimizar o funcionamento dos veículos, automatizando o controle de seus sistemas mecânicos e elétricos (GUI; SIDDIQUI; SAQIB, 2018). De maneira geral, uma ECU utiliza informações provenientes de sensores, sinais externos, ou de outras ECUs para controlar atuadores presentes no veículo. Por exemplo, o sistema de frenagem ABS (*Anti-Lock Brake System*) de um veículo pode utilizar informações provenientes do Módulo de Controle do Trem de Força (PCM) para verificar se o controle de tração está funcionando corretamente (ALAM, 2018).

A Figura 2.1 apresenta a arquitetura básica de uma ECU e suas interfaces de comunicação, suprimento de energia e sinais de entrada e saída. O microcontrolador é

o componente principal da ECU, sendo responsável por realizar todo o processamento e controle do sistema de acordo com suas configurações e programação interna. Os sinais de entrada, podendo ser digitais ou analógicos, são tratados por circuitos de condicionamento de sinal e enviados para o microcontrolador que, por sua vez, envia sinais de controle para os atuadores. As interfaces de comunicação disponíveis em uma ECU podem variar de acordo com sua aplicação e tecnologias utilizadas pelo veículo. Em sistemas modernos, as ECUs normalmente utilizam interfaces de rede do tipo CAN (*Controller Area Network*), LIN (*Local Interconnect Network*) e FlexRay (ISERMANN, 2022).

Figura 2.1 – Arquitetura genérica de uma ECU e interface com sinais externos.



Fonte: Adaptado de Dietsche, Konrad et al. (2022)

Além das tarefas de controle e comunicação interna realizadas pelas ECUs, uma importante função desses dispositivos é a de realizar a troca de informações de diagnóstico com equipamentos externos de desenvolvimento e manutenção. Engenheiros e técnicos de desenvolvimento utilizam essa função para verificar e corrigir falhas no funcionamento dos veículos, assim como otimizar seu desempenho. A comunicação com dispositivos externos também é utilizada durante o processo de produção

de veículos para a aquisição de dados durante testes de rodagem e calibração de parâmetros do veículo (ETAS, 2022c).

2.2 MÉTODOS DE AQUISIÇÃO DE DADOS

A aquisição de dados e alteração de parâmetros de ECUs veiculares pode ser realizada pela rede CAN, utilizando protocolos como *CAN Calibration Protocol* (CCP), *Keyword Protocol 2000* (KWP2000), *Unified Diagnostic Services* (UDS), *On-Board Diagnostic* (OBD) e *Universal Measurement and Calibration Protocol* (XCP) (CIVELEK, 2012). Além desses métodos, a aquisição de dados e a alteração de parâmetros também podem ser realizadas diretamente nos barramentos de memória das ECUs, utilizando equipamentos com interface do tipo JTAG (*Joint Test Action Group*) (ETAS, 2022a).

Nas seções subsequentes abordaremos apenas os protocolos CAN, CCP, XCP e a interface JTAG, que são os métodos mais adequados para a aquisição de grandes quantidades de dados e são utilizados comumente por fabricantes de equipamentos destinados ao processo de testes e calibração de protótipos veiculares.

2.2.1 Controller Area Network

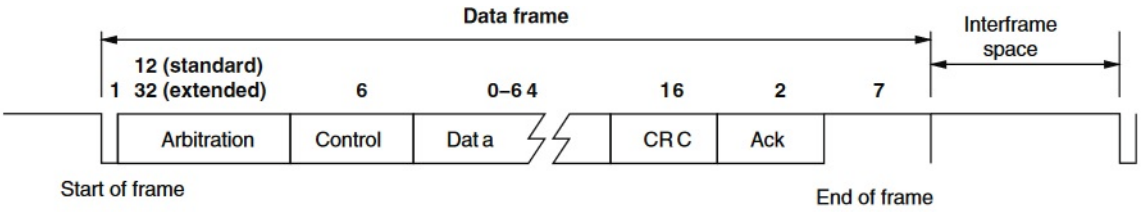
O barramento de comunicação serial *Controller Area Network* (CAN) foi criado na década de 1980, por Robert Bosch GmbH, e projetado para ser uma conexão serial robusta, simples e eficiente. Diferentemente de outras redes como a *Ethernet* e o *Universal Serial Bus* (USB), a rede CAN não é destinada a transferir grandes quantidades de dados, mas, sim, mensagens curtas em períodos de tempo determinados (i.e, requisitos de sistemas de tempo-real), o que faz com que ela seja adequada para a comunicação com sensores e outros dispositivos sensíveis a restrições de tempo (BI et al., 2022; JI; KO; HONG, 2021).

O protocolo utiliza quatro *frames* diferentes para o processo de comunicação: o *frame* de dados, o remoto, o de erro e o de sobrecarga (BI et al., 2022; GROZA; POPA; MURVAY, 2019). O *frame* de dados contém até 8 *bytes* de informação e seu conteúdo é definido por uma camada superior do protocolo. O padrão atual (CAN 2.0A e *Extended* CAN 2.0B) suporta taxas de transmissão de dados de até 1 Mbps (GROZA; POPA; MURVAY, 2019).

O formato do *frame* de dados do protocolo CAN 2.0b é apresentado na Figura 2.2, onde o tamanho dos campos é expresso em bits. Cada quadro começa com um único bit dominante, interrompendo o estado recessivo do barramento ocioso. Em seguida, o campo de identificação define tanto a prioridade da mensagem quanto o conteúdo de dados (identificação) do fluxo de mensagens. Na sequência são definidos: o campo de controle, contendo informações sobre o tipo de mensagem; o campo de

dados, contendo os dados reais a serem transmitidos; o *checksum*, usado para verificar a correção dos bits da mensagem; o *acknowledge* (ACK), usado para confirmar a recepção; o delimitador de fim (ED), usado para definir o término da mensagem; e o espaço ocioso (IS) ou bits interframe (IF), usados para separar um quadro do próximo (NATALE et al., 2012).

Figura 2.2 – Formato do *frame* de dados CAN 2.0b.



Fonte: Natale et al. (2012)

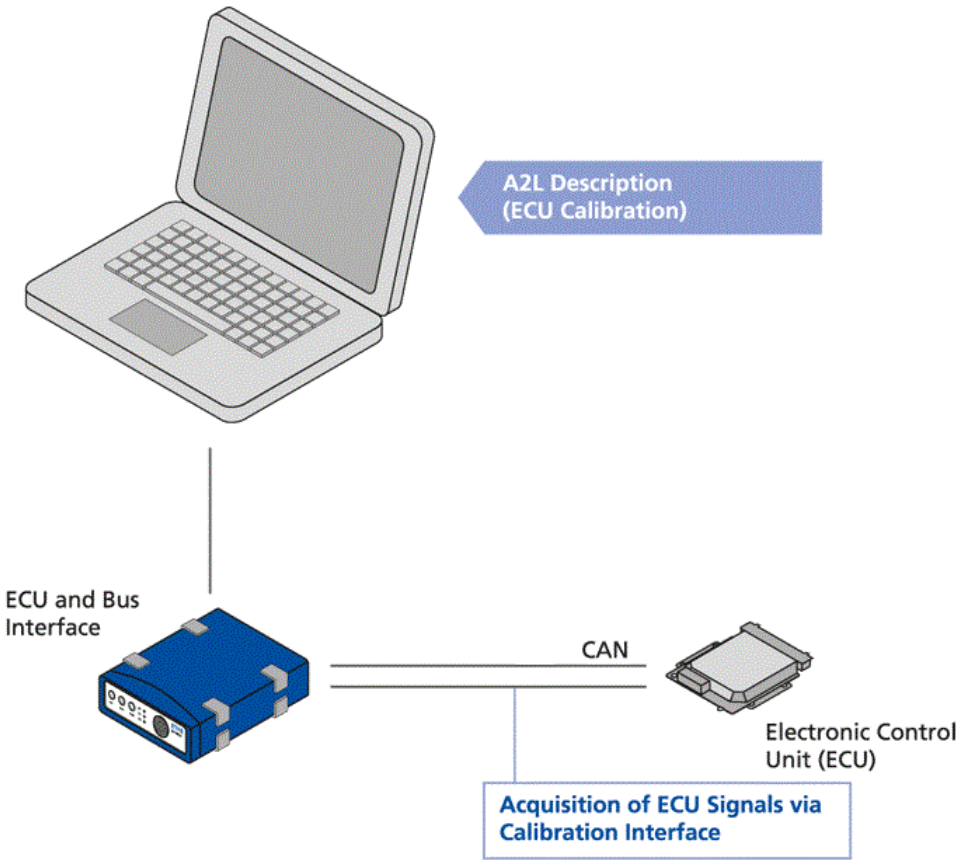
O uso do barramento CAN para a comunicação com ECUs veiculares é muito comum no setor industrial, tanto para extrair dados de diagnóstico quanto dados de testes para validação e calibração desses equipamentos. A Figura 2.3 mostra como a comunicação entre um computador pessoal e uma ECU veicular é realizada utilizando equipamentos comerciais de interface para acesso ao barramento CAN da ECU. Um módulo de interface é responsável por permitir a comunicação entre o barramento CAN e as portas USB ou *Ethernet* do computador. Dependendo das especificações do software utilizado no PC, do módulo de interface e das características da ECU, ambos o CCP e o XCP podem ser utilizados para a comunicação entre os dispositivos.

2.2.2 CAN Calibration Protocol

Enquanto o padrão CAN define as camadas físicas, a camada de *link* de dados e a camada de rede, de acordo com o modelo OSI, o CCP é um protocolo que define a camada de aplicação. Usando o padrão CAN 2.0B, o CCP determina como deve ocorrer a comunicação entre dispositivos, sendo utilizado para obter dados, transferir dados de memória e calibrar ECUs (DU; QI; ZHENG, 2011). Atualmente o protocolo é mantido pela ASAM (*Association for Standardization of Automation and Measuring Systems*).

A versão atual do CCP é a 2.1, lançada em 1999. Desde o lançamento dessa versão, um novo protocolo foi desenvolvido pela ASAM a fim de substituí-lo, o XCP. Atualmente o CCP é considerado obsoleto pela ASAM e o uso do protocolo XCP é recomendado para substituí-lo.

Figura 2.3 – Interface de comunicação entre um PC e uma ECU via barramento CAN.



Fonte: Adaptado de ETAS (2022f)

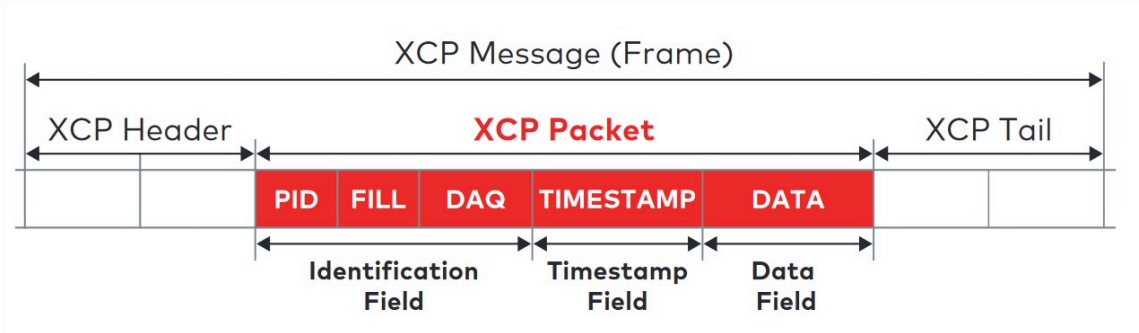
2.2.3 Universal Measurement and Calibration Protocol

A primeira versão do protocolo XCP foi lançada pela ASAM em 2003. Esse protocolo possui uma arquitetura do tipo mestre-escravo, suportando combinações de nós de rede do tipo um mestre para múltiplos escravos em um mesmo barramento. O grande diferencial do XCP é que sua camada de transporte não está vinculada unicamente ao protocolo CAN, podendo operar com diversos protocolos como *Ethernet*, *FlexRay*, USB, CAN e outros (PATZER; ZAISER, 2016a).

O XCP foi projetado para ser um protocolo eficiente e escalável, reduzindo o consumo de recursos dos dispositivos envolvidos como carga de processamento e consumo de memória (PATZER; ZAISER, 2016a). Conforme mostra a Figura 2.4, os dados são transportados em *frames* compostos por um cabeçalho, um pacote de dados e uma calda. O cabeçalho e a calda dependem do protocolo de transporte utilizado, enquanto o pacote de dados XCP possui *frames* muito similares aos do CCP, contendo um identificador, o *timestamp* e os campos de dados (PATZER; ZAISER, 2016b). O pacote de dados é dividido em informações de comandos e dados. Os comandos são utilizados para configurar, calibrar e controlar os dispositivos escravos, exigindo

uma resposta desses dispositivos. Os dados são transferidos utilizando objetos de transferência de dados (DTOs) e estruturas de lista de aquisição de dados (DAQ) (HE; SUN, 2009).

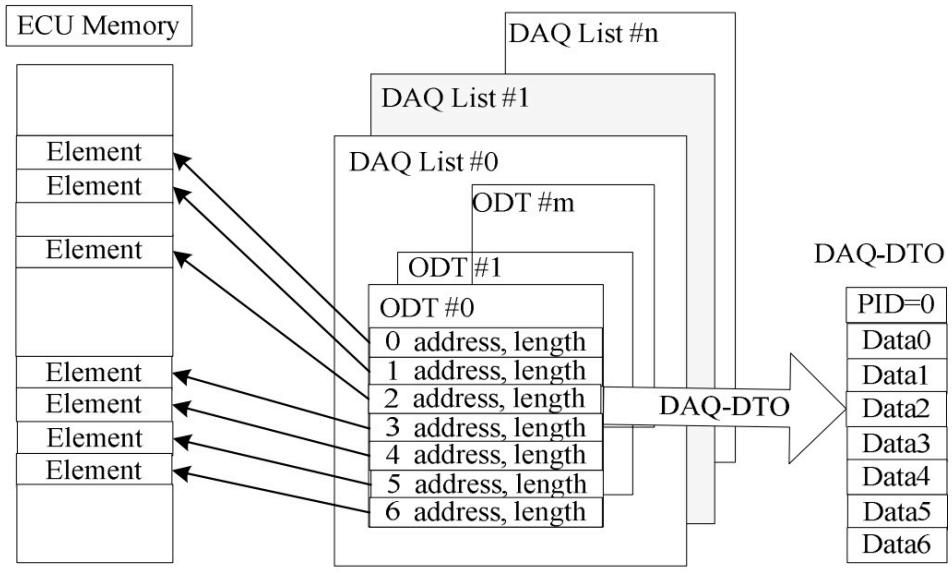
Figura 2.4 – Formato do *frame* de dados do protocolo XCP.



Fonte: Patzer e Zaiser (2016b)

Para realizar a aquisição de dados de um dispositivo escravo periodicamente, as listas de aquisição de dados são inicializadas com tabelas de descrição de objetos (ODTs) contendo os endereços das variáveis e seu comprimento. A partir dessa inicialização, o dispositivo escravo realiza o envio periódico dos DTOs para o dispositivo mestre com as informações que estão gravadas em seus respectivos endereços de memória (HE; SUN, 2009). A Figura 2.4 apresenta a organização da lista de ODTs relacionadas aos elementos da memória de uma ECU.

Figura 2.5 – Organização da lista de ODTs.



Fonte: He e Sun (2009)

2.2.4 JTAG Interface

A interface física desenvolvida pelo *Joint Test Action Group* (JTAG) foi criada em 1985 por diversos fabricantes para ser um padrão industrial que pudesse resolver os problemas de testagem de placas eletrônicas. Conhecida como interface JTAG, o padrão IEEE 1149.1 é uma interface física de hardware que permite a realização de testes de placas eletrônicas, bem como possibilita observar ou modificar o comportamento de circuitos durante sua operação (IEEE, 2012).

Utilizando sua capacidade de monitorar dados provenientes de circuitos eletrônicos, a interface JTAG pode ser aplicada para a extração de dados diretamente dos registradores de memória desses componentes. A velocidade de aquisição de dados por meio dessa interface é muito maior do que a permitida por protocolos de comunicação baseados em barramentos CAN, podendo operar em velocidades de dezenas de megabytes por segundo (DEVADZE et al., 2009). Por outro lado, a conexão da ECU com equipamentos externos por meio desse tipo de interface é muito mais complexa e a captura de dados exige o mapa dos seus registros de memória, que é fornecido somente para empresas que possuem acordos comerciais com as fabricantes de ECUs. Dessa forma, em uma etapa inicial de desenvolvimento, esse tipo de interface de comunicação se mostra inviável.

Alguns dos equipamentos de medição e calibração de ECUs disponíveis atualmente no mercado utilizam a interface JTAG para capturar dados e alterar parâmetros da ECU. A Figura 2.6 mostra um diagrama que representa o uso de um módulo com interface JTAG para a captura de dados e um segundo módulo para estabelecer a comunicação com um PC. Grande parte dos fabricantes utiliza esse esquema de ligação, variando apenas com relação aos softwares e protocolos utilizados.

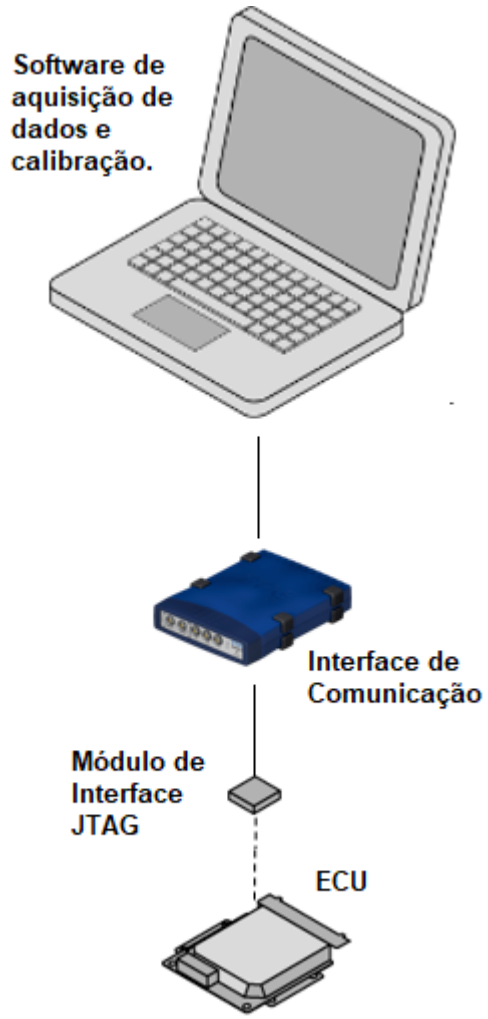
O tipo de módulo utilizado para a aquisição de dados de calibração de parâmetros veiculares via JTAG é diretamente dependente da arquitetura interna da ECU. Quando essa arquitetura possui um barramento de dados e endereços, o módulo pode utilizar uma interface paralela, espelhando o programa que está na memória da ECU em sua própria memória RAM. A Figura 2.7 apresenta um diagrama de blocos contendo a interface entre o módulo e uma ECU com essa arquitetura.

Os modelos de ECUs que não possuem barramento de dados e endereços precisam ser acessados de forma serial, utilizando um módulo de comunicação distinto. A Figura 2.8 mostra um diagrama de blocos para a interface entre a ECU e o módulo de comunicação.

2.3 SISTEMAS DE AQUISIÇÃO DE DADOS COMERCIAIS

Esta seção apresenta uma revisão dos principais sistemas comerciais utilizados na indústria automobilística para medição, calibração e diagnóstico de protótipos

Figura 2.6 – Diagrama de aplicação de módulos de aquisição de dados e calibração de parâmetros de ECUs por meio de interface JTAG.



Fonte: Adaptado de ETAS (2022e)

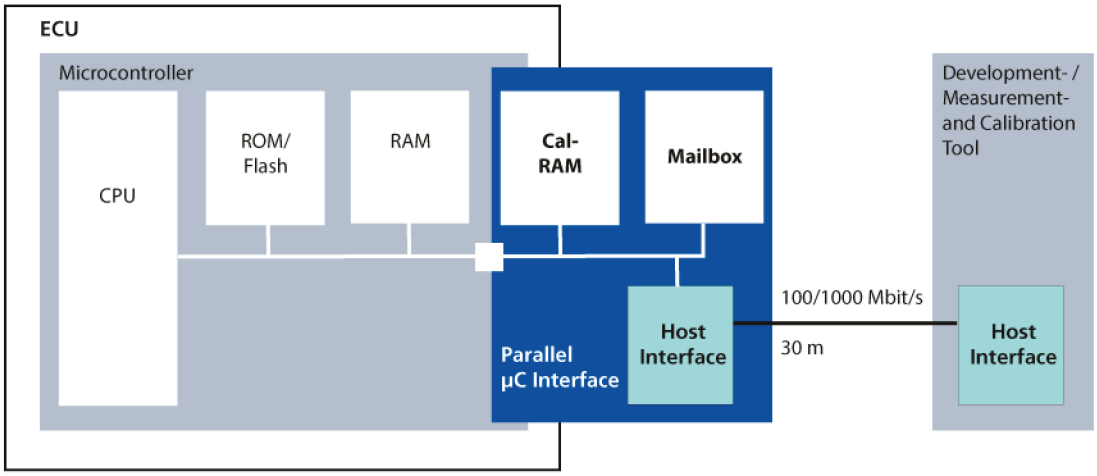
veiculares: Vector Informatik GmbH, Accurate Technologies Inc. (ATI) e ETAS GmbH.

2.3.1 ETAS

A ETAS é uma empresa subsidiária da multinacional alemã Robert Bosch GmbH que oferece equipamentos, serviços de engenharia, consultoria, treinamento e suporte para o desenvolvimento de sistemas embarcados para a indústria automotiva. Em seu portfólio, as duas principais famílias de equipamentos utilizados para a extração de dados e comunicação com as ECUs automotivas são a ETK e a ES. Esses equipamentos são capazes de operar com uma variedade de protocolos industriais como CCP e XCP ou acessar diretamente a memória das ECUs via interface JTAG (ETAS, 2021b; ETAS, 2021a).

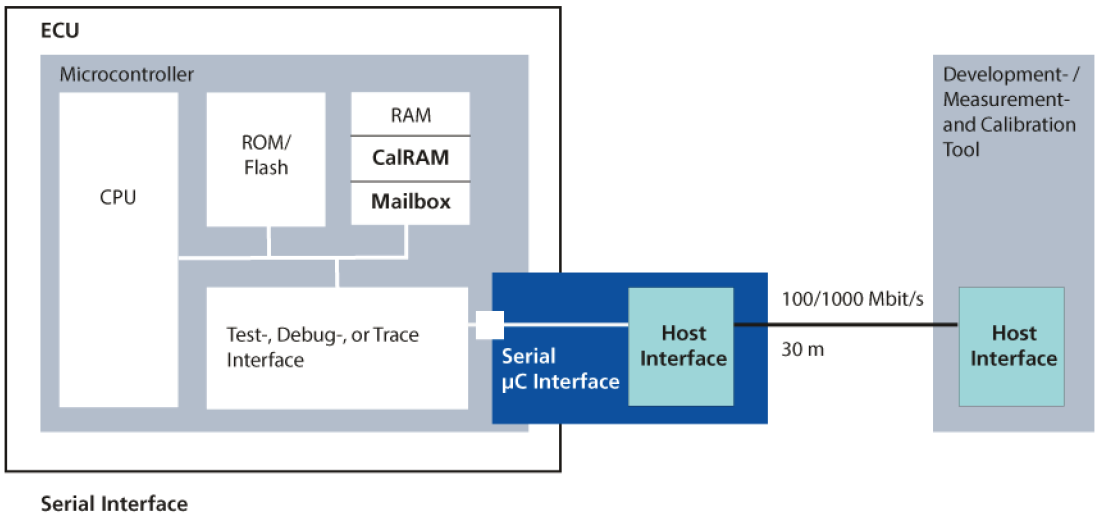
Com o objetivo de proporcionar uma interface homem-máquina, permitindo a

Figura 2.7 – Arquitetura de comunicação paralela entre ECU e módulo de interface JTAG.



Fonte: Adaptado de ETAS (2022d)

Figura 2.8 – Arquitetura de comunicação serial entre ECU e módulo de interface JTAG.



Fonte: Adaptado de ETAS (2022d)

calibração, monitoramento e gravação de dados das variáveis do veículo, o fabricante fornece soluções de software que podem ser instaladas em um computador conectado ao hardware. O software INCA é o produto desenvolvido pela ETAS para medição, calibração e diagnóstico de protótipos veiculares, tendo suporte para comunicação com ECUs que utilizam diferentes tipos de barramentos de comunicação como CAN, CAN FD, LIN, *FlexRay*, *K-Line*, J1939, e *Ethernet* (ETAS, 2022b).

2.3.2 Vector

O Grupo Vector é uma empresa focada na produção de ferramentas de engenharia, software e hardware para o desenvolvimento de sistemas embarcados para veículos, dispositivos médicos ou indústria 4.0. Os principais equipamentos oferecidos pela Vector para a aquisição de dados de ECUs são derivados das famílias VX1000 (VECTOR, 2021b) e *Vector Network Interface* (VECTOR, 2022b).

Os equipamentos da família VX1000 da Vector são projetados para extrair informações e alterar parâmetros diretamente da memória dos microcontroladores das ECUs, utilizando a interface JTAG (VECTOR, 2021b). O módulo de comunicação que é acoplado à ECU realiza a comunicação com um computador via protocolo XCP e conexão física *Ethernet*. A ferramenta de software CANape é o principal software desenvolvido e fornecido pela Vector para medição e calibração das ECUs, possuindo processos automáticos para gerenciamento de parâmetros e análise de dados. (VECTOR, 2022a).

Com base na mesma ferramenta de software, a família *Vector Network Interface* oferece suporte de comunicação para os protocolos CAN (FD), LIN, J1708, *Ethernet*, *Flexray*, 802.11p, MOST e outros (VECTOR, 2022b).

2.3.3 ATI

Há mais de 25 anos desenvolvendo soluções de hardware e software para o setor de medição, calibração e diagnóstico, a ATI oferece dois principais equipamentos para aquisição de dados e calibração de ECUs: as interfaces serial A8 (ATI, 2022a) e o A9 (ATI, 2022b). Uma grande limitação apresentada por esses equipamentos é a compatibilidade com apenas algumas das ECUs comerciais atuais, impossibilitando seu uso em diversos casos.

VISION é a solução de software desenvolvida pela empresa para operar em conjunto com essas interfaces, permitindo a aquisição e a análise de dados além da calibração de parâmetros de ECUs. O software permite também a comunicação do computador via protocolos CCP e XCP, sendo assim compatível com qualquer ECU, independentemente do tipo de módulo de conversão utilizado ou de seu fabricante (ATI, 2022c).

2.4 KNOCK NOISE

O fenômeno conhecido como *knock noise*, batida de motor ou ruído de pré-ignição é um evento de combustão anormal comum em motores de ignição por centelha. A pré-ignição, que ocorre no interior dos cilindros do motor, pode causar sérios danos ao processo de combustão e às emissões do escapamento, bem como reduzir o desempenho de potência e a economia de combustível, gerar ruído, diminuir o conforto

durante a condução do veículo e danificar componentes importantes do motor (BI; MA; WANG, 2019). Um exemplo atual diretamente relacionado a ocorrência do fenômeno de pré-ignição são as ocorrências de falhas nos motores de três cilindros 1.0, fabricados pela empresa General Motors (GM) e utilizados nos veículos modelo Onix Plus MY20. Além de sérios danos aos motor, essas falhas chegaram a provocar explosões e incêndios em alguns desses veículos (FELDMAN, 2019). Em nota oficial, a GM relatou que, por consequência de uma combinação de fatores, a falha estaria sendo gerada devido a um defeito na calibração do software que pode fazer com que ocorra uma condição de pré-ignição na câmara de combustão do motor (AUTOPAPO, 2019).

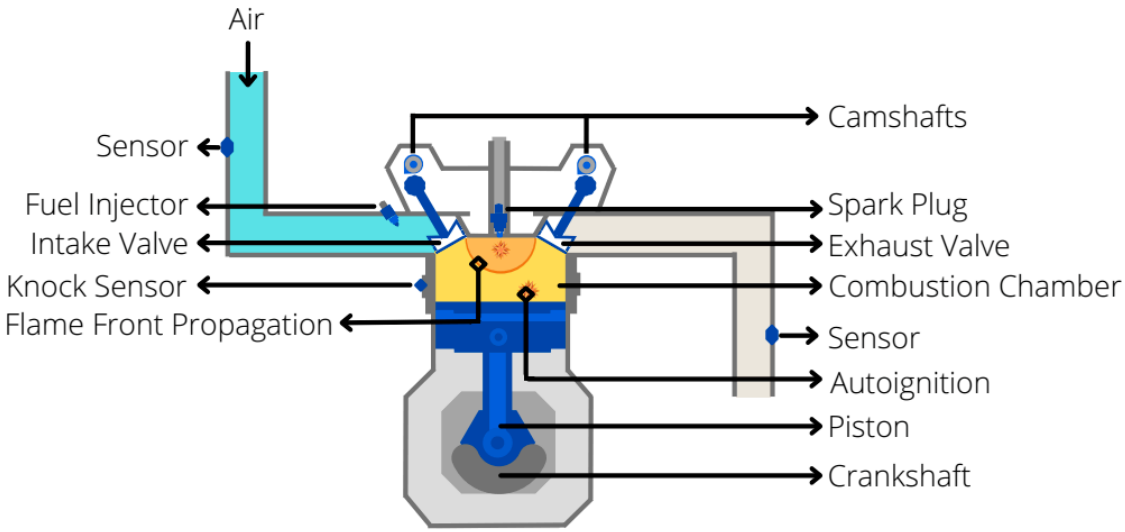
Em condições normais, um motor de ignição por centelha tem o processo de combustão iniciado pela vela de ignição, que permite que a mistura ar/combustível dentro da câmara de combustão seja queimada em uma explosão controlada. O fenômeno de *knock noise* é provocado devido a uma parcela dessa mistura não queimada no processo, chamada de gás final, e a condições específicas de temperatura e pressão dentro da câmara de combustão (LOUNICI et al., 2017). A pré-ignição, que ocorre antes da combustão provocada pela vela de ignição, cria uma onda de choque que é refletida dentro da câmara, causando um aumento na temperatura e pressão, gerando ressonâncias acústicas de alta intensidade características do ruído de detonação (DIETSCHE; KONRAD et al., 2022).

A batida gera uma onda de choque de alta frequência devido à colisão de duas chamas propagantes de combustão, como ilustra a Figura 2.9. A frequência aproximada dessa onda pode ser determinada através da Equação 2.1, onde f é a frequência de ressonância, c é a constante da velocidade do som, B é o coeficiente do modo de vibração e D é o diâmetro do cilindro (SHEN; ZHANG; SHEN, 2019; OFNER et al., 2022). Para motores automotivos, as frequências dos modos de vibração da batida geralmente estão dentro da faixa de 6 a 25 kHz (BI; MA; WANG, 2019; SHEN; ZHANG; SHEN, 2019).

$$f = \frac{c \cdot B}{\pi D} \tag{2.1}$$

Para evitar a ocorrência da batida, é essencial reduzir a pressão dentro do cilindro, o que pode ser realizado atrasando o ponto de ignição da mistura de combustível. No entanto, o atraso do ponto de ignição reduz o trabalho de expansão, responsável pelo movimento do pistão para baixo, resultando em uma menor eficiência do motor (DIETSCHE; KONRAD et al., 2022; MAHENDAR, 2021). Nesse sentido, para possibilitar o controle adequado da taxa de compressão variável do motor, é de grande importância que o atraso seja aplicado no momento necessário e da maneira correta. Para isso, a precisão e qualidade da informação referente à ocorrência do *knock noise* é imprescindível.

Figura 2.9 – Ocorrência de *knock noise* em um motor de ignição por centelha.



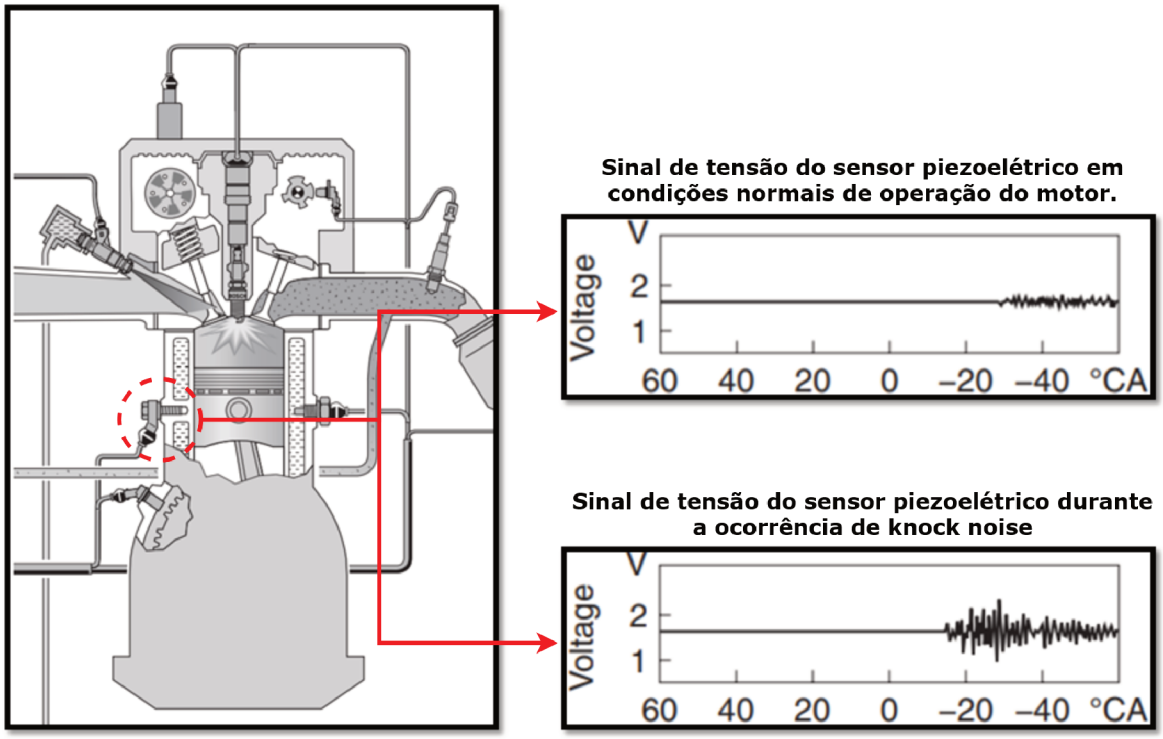
Fonte: Francis et al. (2022)

Existem diversos métodos que podem ser usados para detectar a ocorrência de *knock noise*. Esses métodos podem ser classificados em duas categorias amplas: diretos e indiretos. Os primeiros são baseados na medição direta e no estudo dos parâmetros internos do cilindro, que podem ser influenciados pela detonação. Por sua vez, os métodos indiretos são baseados na medição de variáveis como pressão sonora ou vibração do bloco do cilindro (ZHEN et al., 2012).

A detecção de detonação pode ser realizada com base em vários tipos de procedimentos, como análise da pressão dentro do cilindro, da vibração do bloco do cilindro, da corrente iônica, de radiação luminosa e emissões acústicas, de transferência de calor, de temperatura, entre outros (ZHEN et al., 2012). O método de análise da pressão dentro do cilindro utiliza um sensor instalado na câmara de combustão para medir a pressão durante a combustão. É um método altamente preciso para detectar a batida do motor, mas sua aplicação em larga escala é limitada devido à confiabilidade do sensor, à estrutura do motor e ao custo envolvido (BI; MA; WANG, 2019). Por outro lado, os métodos baseados na análise de vibração do bloco do motor utilizam sensores de fácil instalação, alta confiabilidade e baixo custo, motivos pelos quais são muito utilizados para motores de carros produzidos em massa (BI; MA; WANG, 2019; ZHEN et al., 2012).

O sensor piezoelétrico utilizado para detecção do sinal de vibração é instalado no bloco do motor, conforme mostra a Figura 2.10. Esse sensor é capaz de gerar um sinal elétrico de tensão que pode ser capturado e tratado, permitindo a detecção de ocorrências de *knock noise*. A Figura 2.11 apresenta o comportamento do sinal proveniente do sensor piezoelétrico em comparação com a curva de pressão caracte-

Figura 2.10 – Sensor de vibração piezoelétrico instalado na estrutura do motor e respectivos sinais de tensão gerados ao longo da posição do cilindro.



Fonte: Adaptado de Dietsche, Konrad et al. (2022)

rística da câmera de combustão em um ciclo normal de operação e em um ciclo com ocorrência de *knock noise*. Os sinais de pressão e tensão do piezoelétrico expostos estão filtrados para as frequências características do ruído de detonação.

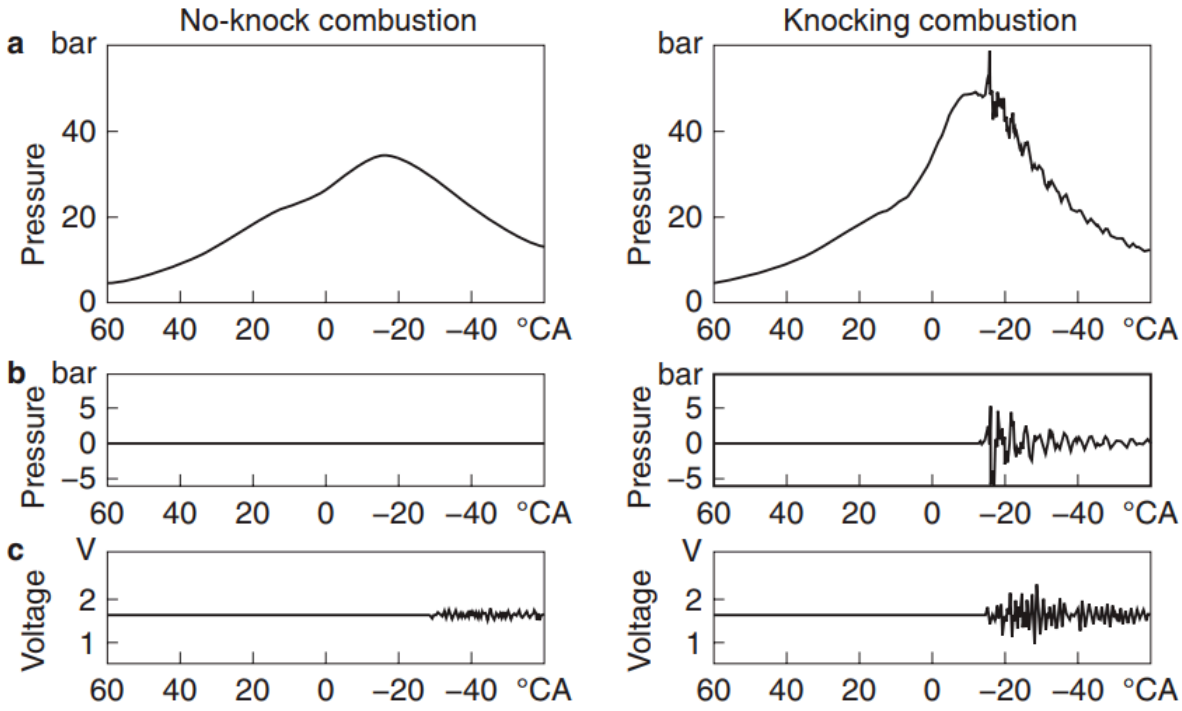
As ECUs veiculares atuais realizam a detecção da pré-ignição utilizando diversas variáveis envolvidas no processo de combustão e tem como principal fonte de informação os sensores piezoelétricos. Os veículos fabricados atualmente são equipados com esses sensores para possibilitar a identificação das ocorrências desse fenômeno e atuar preventivamente no sistema de ignição para evitar danos ao motor.

2.5 TRABALHOS RELACIONADOS

Ao longo dos últimos anos, diversos projetos de software e hardware têm sido desenvolvidos a fim de extrair dados, controlar ou calibrar parâmetros de ECUs veiculares. Diversos são os objetivos desses sistemas, como monitorar e calibrar ECUs veiculares (WANG et al., 2005), controlar fontes de energia e cargas auxiliares de veículos elétricos (AZZEH, 2007) e detectar excesso de velocidade em veículos inteligentes (KHAN et al., 2019), entre outros.

De maneira muito similar aos sistemas comerciais atuais utilizados para esse

Figura 2.11 – Curvas características de pressão na câmara de combustão e sinais correspondentes do sensor de *knock noise*. (a) Curva característica típica de pressão na câmara de combustão; (b) Sinal de pressão na câmara de combustão filtrado por um filtro passa-faixa; (c) Sinal do ruído transmitido pela estrutura e captado pelo sensor de *knock noise*.



Fonte: Adaptado de Dietsche, Konrad et al. (2022)

fim, o trabalho de Wang et al. (2005) apresenta o desenvolvimento de um sistema para medição e calibração de parâmetros de ECUs veiculares. Usando um computador pessoal (PC) e um cartão de interface CAN para PC da empresa Vector Informatik GmbH, o sistema proposto realiza a comunicação com a ECU veicular via CCP e permite a aquisição de dados e calibração de parâmetros com base no arquivo de mapeamento de links do software da ECU. Esse arquivo é gerado durante o desenvolvimento da ECU e contém a lista de dados com os endereços de todas as variáveis da memória *flash* do microcontrolador da ECU (WANG et al., 2005). Apesar de o trabalho de Wang et al. (2005) ter se mostrado eficiente para o monitoramento e calibração de parâmetros de ECUs, o sistema permanece dependente de um PC, utilizando um hardware comercial para a conexão entre o PC e o barramento CAN, o que consequentemente eleva seu custo de produção.

O trabalho apresentado por Dzambic et al. (2021), por sua vez, mostra o desenvolvimento de uma interface para medição e calibração de ECUs via XCP, utilizando uma placa eletrônica de desenvolvimento Aurix TC277. Caracterizado como um sistema de prototipagem rápida, o dispositivo é projetado para se comunicar com um PC

via barramento *Ethernet* e com uma ECU via barramento CAN, ambos em XCP. Como diferenciais, o hardware apresenta baixo custo em comparação com equipamentos comerciais similares e um sistema inteligente de *watchdog* implementado para permitir a detecção de falhas e possíveis implementações de algorítimos de tomada de decisão (DZAMBIC et al., 2021). Mesmo possuindo um hardware de menor custo, o trabalho desenvolvido por Dzambic et al. (2021) permanece limitado à necessidade de uso de um PC e de softwares comerciais para estabelecer a comunicação com a placa eletrônica. Além disso, o trabalho realizado é focado no desenvolvimento do software para a realização da comunicação entre o PC e a ECU, utilizando um hardware genérico na aplicação.

Em uma abordagem diferente, outros trabalhos foram desenvolvidos com base no sistema OBD ou em sua versão mais recente, o OBD-II (PALOMINO; CUTY; HUANACHIN, 2021; MALEKIAN et al., 2017; SMITH; MILLER, 2013), que é um sistema de monitoramento e reporte de dados de componentes relacionados a emissões e é padronizado pela Sociedade de Engenheiros Automotivos (SAE) (IQBAL et al., 2017; DZHELEKARSKI; ALEXIEV, 2005). O padrão OBD-II permite a conexão de equipamentos externos com a ECU por meio de diferentes protocolos como o SAE J1850-PWM, o ISO 9141-2, o ISO 15765 CAN, entre outros, sendo o mais rápido deles atualmente o protocolo CAN, com velocidade de comunicação de 500 kbits/s (SAE, 2018; IQBAL et al., 2017). A principal limitação apresentada para a aquisição de dados via padrão OBD-II é a de que somente uma limitada quantidade de variáveis pode ser acessada por meio desse protocolo (DZHELEKARSKI; ALEXIEV, 2005).

Em uma perspectiva geral, o acesso ao barramento CAN veicular por meio dos diferentes protocolos disponíveis permitiu o desenvolvimento de diversas aplicações de monitoramento e controle de variáveis e subsistemas específicos, como tacógrafos em veículos de carga (FERNANDES, 2016), velocidade, consumo de combustível, carga e torque do motor de veículos *off-road* (KHEIRALLA; ABUBAKR; GAMAL, 2019).

No que diz respeito a veículos inteligentes, o uso de recursos como localização GPS, redes GSM, 4G e 5G, e o paradigma da Internet das Coisas (IoT) tem sido realizado em trabalhos relacionados a sistemas de segurança inteligentes (SATHIYANARAYANAN; MAHENDRA; VASU, 2018), detecção de excesso de velocidade (KHAN et al., 2019), sistemas de monitoramento veicular (CHINONSO; ANAYO; ANIKWE, 2021), cidades inteligentes (MUSTAKIM, 2020) e outros (MALLIDI; VINEELA, 2018; FERRARI et al., 2020).

Um exemplo de monitoramento e análise de dados utilizando o sistema OBD-II, redes de internet móvel e o paradigma da IoT é apresentado por Hamid et al. (2019). Em seu trabalho intitulado *Smart Vehicle Monitoring and Analysis System (VMAS) with IOT Technology*, Hamid et al. (2019) mostra o desenvolvimento de um sistema capaz de extrair diversos parâmetros do motor e enviá-los para um servidor na nuvem.

Os dados, uma vez enviados ao servidor, podem ser facilmente acessados por uma interface de usuário. O hardware utilizado para o VMAS consiste basicamente em um *smartphone* com sistema operacional Android e uma placa de interface OBD-II com conexão *Bluetooth*. O *smartphone* realiza a requisição de dados ao conector OBD-II e, após recebê-los, envia-os para o servidor na nuvem. Por sua vez, os dados são visualizados por meio de uma interface de usuário (HAMID et al., 2019).

Outros trabalhos apresentam sistemas de captação de dados veiculares capazes de realizar a comunicação via OBD-II em conjunto com a aquisição de sinais de sensores específicos. O trabalho de Kumar e Ravi (2022), por exemplo, mostra o desenvolvimento de um hardware capaz de detectar vibrações de alta frequência utilizando acelerômetros distribuídos pelo veículo e simultaneamente adquirir informações provenientes da interface de comunicação OBD-II. Nesse sentido, o hardware apresentado por Simões e Mafort (2021) também é capaz de capturar dados da interface OBD em conjunto com informações provenientes de sensores inerciais, GPS e relógios de tempo real.

Trabalhos relacionados à detecção de *knock noise* em geral utilizam hardwares comerciais para a captura de dados e têm seu foco na análise e tratamento das informações obtidas. Os diversos métodos utilizados para a detecção da ocorrência do ruído de detonação incluem: o uso de ferramentas matemáticas como a *short-time Fourier transform* (AKIMOTO; KOMATSU; KURAUCHI, 2013), *wavelet-denoising*, *empirical mode decomposition* (EMD) (BI; MA; WANG, 2019), *variational mode decomposition* (VMD) (BI et al., 2019), e a aplicação de algoritmos de IA, como redes neurais convolucionais (OFNER et al., 2022), máquina de vetores de suporte (SVM), classificador de sequência, *autoencoder* e *isolated forest* (FRANCIS et al., 2022)

Considerando a revisão de literatura realizada, é possível notar que, apesar da existência de diversos trabalhos desenvolvidos para realizar a aquisição de dados veiculares, utilizando diferentes métodos de comunicação com as ECUs, nenhum deles apresenta uma solução similar ao hardware proposto no presente trabalho. Poucos sistemas abordam a unificação da aquisição de dados e o uso da tecnologia de IoT, e nenhum dos que utilizam essas tecnologias o faz a fim de coletar dados para teste de protótipos veiculares, limitando-se ao uso de interfaces de comunicação como o OBD-II, que não são aplicáveis para esse tipo de situação. A Tabela 2.1 mostra um comparativo das características dos principais trabalhos relacionados em relação ao hardware apresentado nesse trabalho.

Tabela 2.1 – Principais trabalhos relacionados e suas características em comparação com o hardware desenvolvido.

Características	Hardware Desenvolvido	Wang et al. (2005)	Dzambic et al. (2021)	Palomino, Cuty e Huanachin (2021)	Malekian et al. (2017)	Hamid et al. (2019)	Kumar e Ravi (2022)	Simões e Mafort (2021)	Chinonso, Anayo e Anikwe (2021)
OBD-II				X	X	X	X	X	
CCP	X	X							
XCP	X		X						
Protocolos de comunicação adicionais	X						X	X	X
Dependente de PC		X	X		X				
Hardware embarcado	X			X	X	X	X	X	X
Intergração com a nuvem	X					X			X
Comunicação 4G/LTE	X					X			X
Comunicação Wi-Fi	X				X	X			
Tenologia GPS	X				X		X	X	X
Interface Bluetooth	X					X			
Capacidade de armazenamento de dados	X	X	X	X	X		X	X	X
Capacidade de processamento de dados	X	X	X	X	X		X	X	X

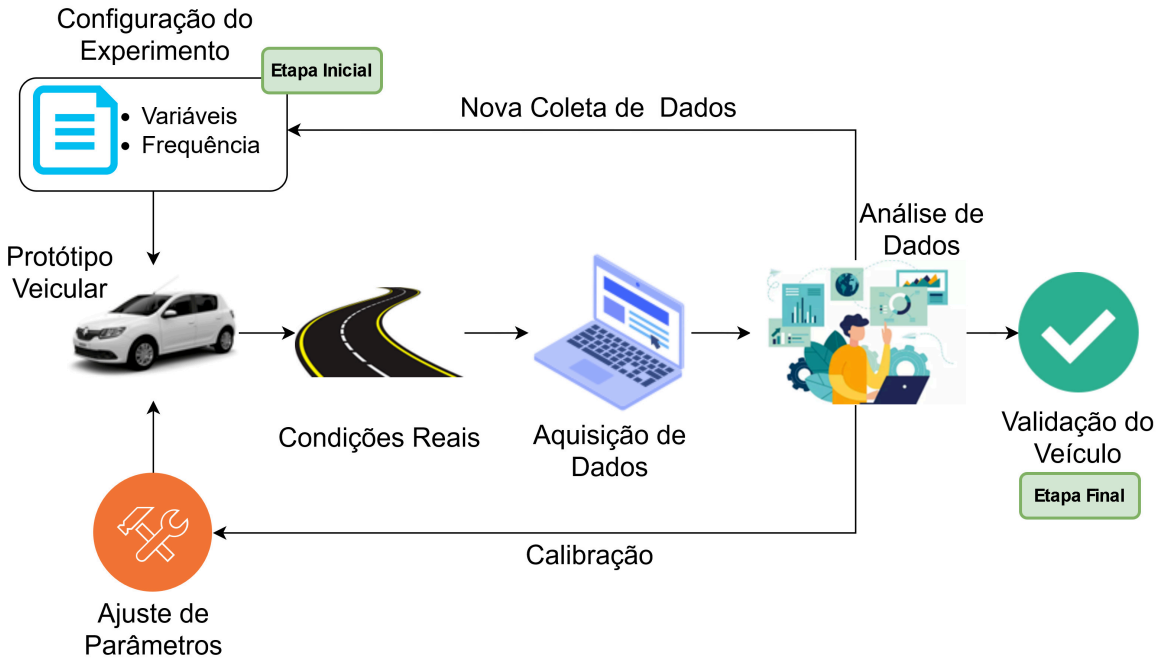
Fonte: Elaborada pelo autor.

3 TESTES, VALIDAÇÃO E CALIBRAÇÃO DE PROTÓTIPOS VEICULARES

O processo de testagem, validação e calibração de protótipos veiculares na indústria automotiva é uma parte essencial do desenvolvimento e produção de novos modelos de veículos. Sem que o novo modelo de veículo esteja totalmente testado, calibrado e atendendo as normas vigentes, sua produção em série não pode ser iniciada.

Cada fabricante pode apresentar procedimentos específicos distintos para a execução dos testes de rodagem de protótipos veiculares, porém, de maneira geral, todos eles seguem alguns passos essenciais para a adequada calibração e validação final do veículo. Esses passos serão apresentados a seguir, considerando o uso de equipamentos comerciais. Os testes a serem descritos consideram a aplicação de equipamentos que utilizam o barramento CAN para a comunicação com as ECUs. A Figura 3.1 apresenta as etapas do processo.

Figura 3.1 – Etapas do processo de testagem, calibração e validação de protótipos veiculares.



Fonte: Elaborada pelo autor.

Primeiramente, a equipe de especialistas responsável pelos testes do veículo precisa definir o conjunto de variáveis a serem monitoradas ao longo da rodagem do veículo. Durante a seleção dessas variáveis, dois parâmetros essenciais devem ser definidos, a variável em si e o seu respectivo período de aquisição, que irá determinar a frequência na qual a ECU enviará o valor dessa variável ao equipamento. O conjunto de informações provenientes dessa seleção de variáveis e períodos de aquisição é

denominado experimento. Cada experimento apresenta uma quantidade máxima de variáveis que podem ser configuradas em função do barramento de comunicação utilizado, do tipo de variáveis selecionadas e de sua frequência de aquisição.

O experimento, por sua vez, é carregado no equipamento de medição instalado no veículo. Com o sistema corretamente instalado e o experimento carregado, o motorista deve ligar o sistema e o veículo, iniciar a execução do experimento e dirigir o automóvel nas condições predefinidas pela equipe de especialistas. Após conduzir o veículo ao longo do percurso preestabelecido, o motorista deve encerrar a execução do experimento.

Finalizada a primeira etapa de testes, os especialistas devem capturar os dados adquiridos pelo equipamento e avaliar os valores obtidos durante o teste. Caso necessário, a execução de novas etapas de rodagem podem ser solicitadas, utilizando experimentos iguais ou modificados, a fim de adquirir novas informações para a correta avaliação do funcionamento do veículo. Além disso, ajustes nos parâmetros das ECUs podem ser solicitados para corrigir a calibração do protótipo. Os ajustes de parâmetro são, em geral, realizados pelo mesmo equipamento de medição utilizado, por meio do envio de dados pelo barramento CAN.

Após a realização das calibrações necessárias, ou simplesmente após a substituição do experimento a ser realizado, uma nova etapa de rodagem e monitoramento de variáveis é executada. A quantidade de testes de rodagem executada depende de diversos fatores, como procedimentos internos de fabricação, problemas apresentados durante a execução do experimento, necessidade de medir variáveis específicas, ajustes de calibração, entre outros.

Por fim, após toda a sequência de testes de rodagem ser realizada e os dados adquiridos serem avaliados, o veículo pode ser validado pela equipe de especialistas.

4 MATERIAIS E MÉTODOS

Este capítulo descreve as etapas do trabalho e os materiais utilizados para o seu desenvolvimento. Inicialmente, o processo de levantamento de requisitos é apresentado. Na sequência, a seleção dos principais componentes do hardware é descrita, indicando sua relação com os requisitos funcionais e não funcionais do sistema. Por fim, as etapas de desenvolvimento do hardware são apresentadas e os procedimentos de teste e validação do hardware são definidos.

Os testes de desenvolvimento e implementação dos sistemas foram realizados utilizando um veículo do modelo Renault Sandero, ano 2019, fornecido pela empresa Renault S.A. e equipado com uma ECU com barramento de comunicação CAN disponível no protocolo CCP. Além disso, foram realizados testes em bancada com uma ECU também fornecida pela empresa Renault S.A. capaz de realizar a comunicação pelo protocolo XCP em barramento CAN.

4.1 LEVANTAMENTO DE REQUISITOS

O levantamento de requisitos do hardware tem como base os requisitos funcionais do projeto IASE, que é um sistema desenvolvido para realizar a aquisição de dados e a calibração de parâmetros de ECUs automotivas durante a fase de testes, calibração e validação de protótipos veiculares. Além disso, o sistema foi projetado para operar com o paradigma de IoT e utilizar algoritmos de Inteligência Artificial a fim de realizar a detecção de anomalias no funcionamento de *powertrains*, indicando possíveis ajustes de calibração ou informando falhas críticas de operação.

Tendo como base requisitos do IASE, o levantamento de requisitos do hardware foi realizado considerando o uso e tecnologias que permitissem a maximização da eficiência e confiabilidade do sistema.

4.1.1 Requisitos do IASE

Os requisitos do IASE foram definidos pela equipe de pesquisa do LISHA em conjunto com a equipe de especialistas da empresa Renault. A fim de especificar cada funcionalidade e requisito do sistema, reuniões recorrentes foram realizadas entre as equipes.

O ponto de partida para a determinação dos requisitos do projeto são os processos de teste, calibração e validação de veículos realizados atualmente pela empresa Renault. Com base nos procedimentos e equipamentos comerciais utilizados, foi possível determinar as funcionalidades essenciais do sistema, como: capacidade de comunicação via barramento CAN, capacidade de armazenamento e processamento de dados e interfaces de configuração e controle de experimentos.

Partindo das especificações básicas do projeto, a implementação de outras tecnologias foi debatida, ponderando a viabilidade de cada proposta. Considerando a disponibilidade de um servidor de IoT e o conhecimento específico da equipe do LISHA com relação à utilização desse tipo de tecnologia, um dos principais requisitos definidos para o projeto foi a implementação da transmissão dos dados coletados para o servidor de IoT através da rede 4G. Aproveitando a disponibilidade dos dados na nuvem, outro requisito especificado foi a utilização de algoritmos de IA para automatizar processos de detecção de falha e geração de relatórios.

Conforme apresenta a Figura 4.1, os principais requisitos do projeto IASE são:

1. **Aquisição de dados via barramento CAN (CCP/XCP)**: o sistema deve ser capaz de realizar a comunicação com a ECU via barramento CAN, utilizando protocolos de comunicação CCP e XCP, a fim de capturar os dados veiculares dos diferentes modelos de ECU.

2. **Calibração de ECUs via barramento CAN (CCP/XCP)**: o sistema deve ser capaz de realizar a comunicação com a ECU via barramento CAN, utilizando protocolos de comunicação CCP e XCP, a fim de possibilitar o ajuste de parâmetros de calibração do veículo para os diferentes modelos de ECU.

3. **Upload de dados baseado no paradigma de IoT**: utilizando tecnologias da rede de telefonia móvel, como a rede 4G, o sistema deve ser capaz de enviar dados padronizados para o servidor de IoT, garantindo a segurança dessas informações e o armazenamento e o gerenciamento eficiente dos dados coletados.

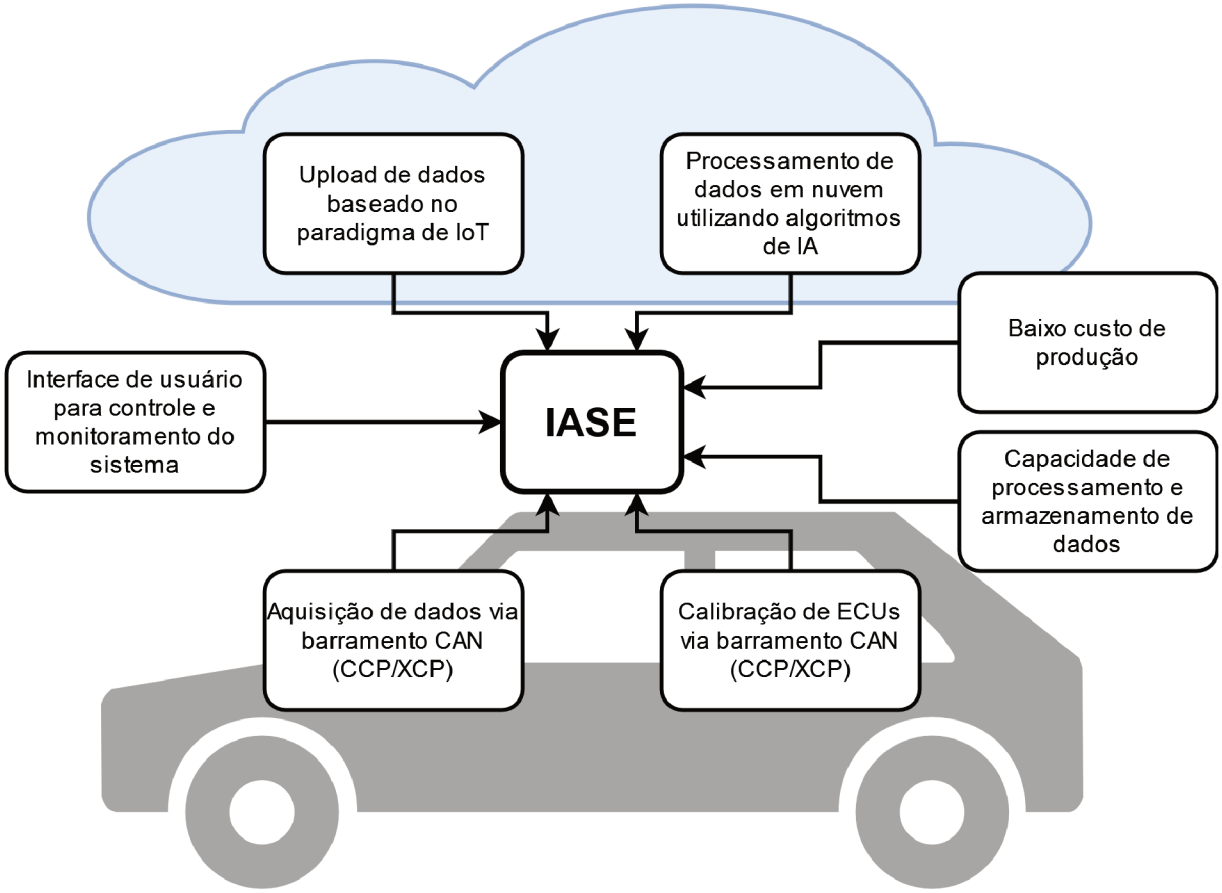
4. **Processamento de dados em nuvem utilizando algoritmos de IA**: os dados coletados e enviados ao servidor devem ser processados por algoritmos de IA, automatizando o processo de análise preliminar de dados, a detecção de falhas ou desvios de comportamento em tempo real, entre diversas outras aplicações possíveis.

5. **Capacidade de processamento e armazenamento de dados**: o sistema deve ser capaz de processar, preparar e enviar os dados coletados em tempo real e ter capacidade suficiente de memória para armazenar dados temporários e permanentes dos testes em execução.

6. **Interface de usuário para controle e monitoramento do sistema**: para realizar o controle e monitoramento da captura de dados durantes os testes veiculares, o sistema deve disponibilizar uma interface de usuário intuitiva que permita ao motorista ou ao especialista responsável verificar o status do equipamento e iniciar ou parar a coleta de dados.

7. **Baixo custo de produção**: os materiais e serviços necessários para o desenvolvimento e produção do projeto devem ser selecionados visando o baixo custo do sistema.

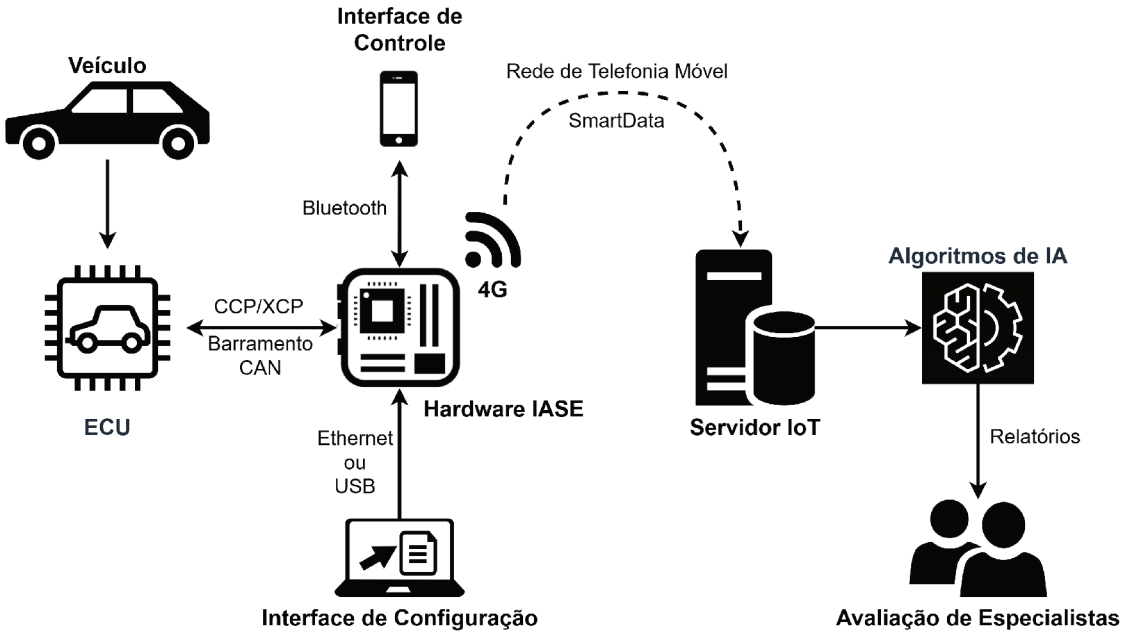
Figura 4.1 – Requisitos principais do IASE.



Fonte: Elaborada pelo autor.

Tendo como base os requisitos do IASE, uma visão geral do fluxo de dados do projeto é representado pelo diagrama de blocos na Figura 4.2. A ECU, conectada aos principais sensores do *powertrain* do veículo fornece dados tratados e não tratados periodicamente ao hardware IASE utilizando o barramento CAN e protocolos CCP ou XCP. Por sua vez, o hardware com o *firmware* implementado nele é configurado por meio da interface de configuração (via *Ethernet* ou USB) para executar o experimento definido pelos especialistas. A interface de controle utiliza a tecnologia *Bluetooth* para realizar o controle de execução do sistema por meio de um aplicativo desenvolvido para *smartphones*. Durante a execução dos testes, os dados coletados são enviados ao servidor de IoT via rede 4G e processados por algoritmos de IA que, por sua vez, fornecem relatórios para avaliação de especialistas da empresa fabricante.

Figura 4.2 – Visão geral do fluxo de dados do IASE.



Fonte: Elaborada pelo autor.

4.1.2 Requisitos de Hardware

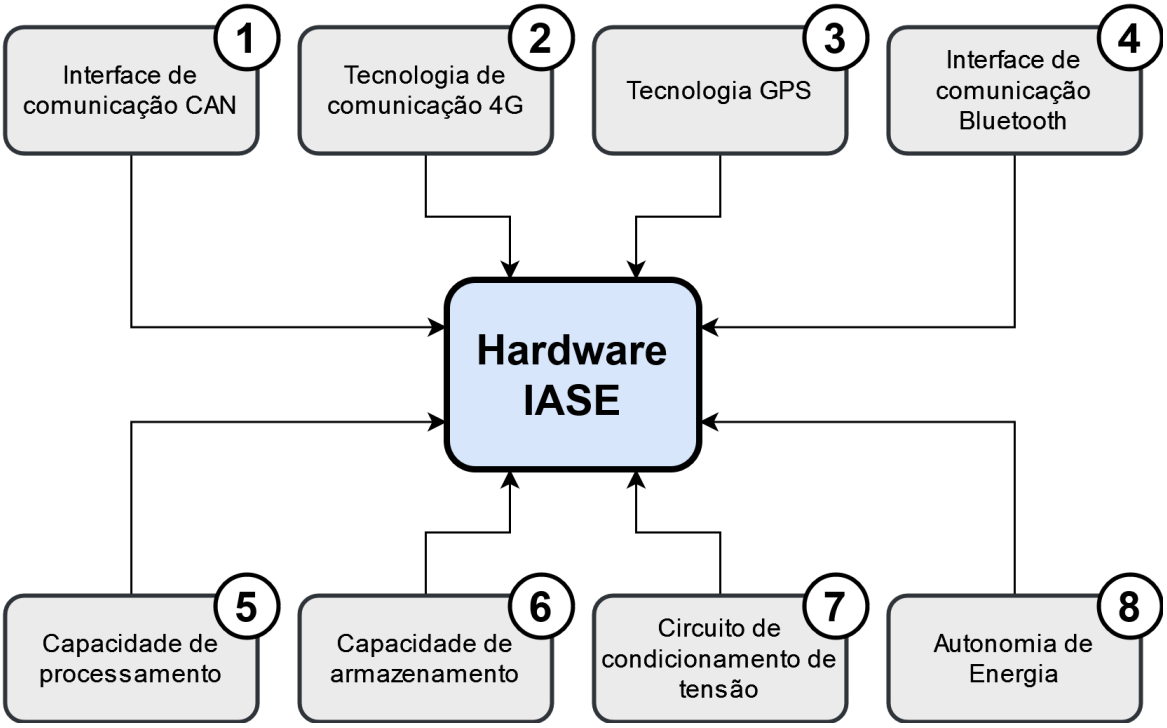
Conforme apresentado na seção anterior, a definição dos requisitos de projeto do IASE foi realizada de acordo com as necessidades básicas da indústria automobilística e na intenção de aplicar tecnologias de IoT e IA para proporcionar maior agilidade, confiabilidade e eficiência ao processo de validação e calibração de protótipos veiculares. Adicionalmente, pretende-se utilizar o hardware para viabilizar a implementação de tecnologias como computação de borda e captura de dados de sensores específicos integrados ou não aos veículos. A seguir, são listados os requisitos de hardware (Figura 4.3) necessários para atender os requisitos do sistema:

1. Possuir interface de comunicação CAN que possibilite estabelecer comunicação com ECUs veiculares utilizando protocolos CCP ou XCP. Essa interface é responsável por permitir que o sistema realize a aquisição de dados e a calibração de parâmetros do veículo, conforme requisitos 1 e 2 do IASE.

2. Ser capaz de realizar a transmissão de dados via rede 4G em tempo real, seguindo os padrões de formatação de IoT (requisitos 3 e 4 do IASE).

3. Obter dados de localização do veículo via Sistema de Posicionamento Global (GPS). Essas informações devem ser utilizadas para compor os pacotes de dados padronizados enviados ao servidor de IoT (requisito 3 do IASE).

Figura 4.3 – Requisitos de hardware.



Fonte: Elaborada pelo autor.

4. Possuir interface de comunicação *Bluetooth* para o controle e o monitoramento de execução do sistema (requisito 6 do IASE).

5. Ter capacidade de processamento suficiente para gerenciar todo o fluxo de dados proveniente do veículo e demais periféricos, evitando a perda de dados do sistema e garantindo o envio de informações em tempo real (requisito 5 do IASE). Especificações mínimas estimadas: 2 núcleos e *clock* de 1.0 GHz.

6. Realizar o armazenamento dos dados adquiridos sempre que a falta de conexão com o servidor não permita o envio de dados, provendo capacidade de memória suficiente para que não ocorram perdas de informação (requisito 5 do IASE). Valor mínimo estimado para o armazenamento de dados: 8 GB.

7. Ser capaz de utilizar a bateria veicular para o fornecimento de energia a todos os componentes de hardware empregando circuitos de condicionamento de tensão.

8. Possuir fonte de energia própria (i.e., bateria) garantindo o funcionamento do sistema durante possíveis interrupções da energia proveniente da bateria do veículo.

4.2 PROJETO E DESENVOLVIMENTO DO HARDWARE

A concepção e implementação do projeto, assim como o desenvolvimento do hardware, foram conduzidos com base nos requisitos funcionais previamente listados, com o objetivo de garantir a operacionalidade do sistema e mitigar potenciais gargalos indesejados.

Ao longo do trabalho foram produzidas três diferentes versões do hardware a fim de corrigir erros de projeto e problemas no esquemático e na estrutura da PCB, assim como incluir novas funcionalidades de controle ao sistema. Cada uma das versões passou por procedimentos similares de revisão, testes e validação.

Nesta seção, são descritos o processo de seleção dos principais componentes, a elaboração do projeto e a produção do hardware, com foco na última versão da placa eletrônica desenvolvida.

4.2.1 Seleção de Componentes

Embora determinados parâmetros do sistema tenham sido preestabelecidos durante o levantamento de requisitos, algumas informações essenciais para a seleção de componentes não estavam disponíveis durante as fases de produção do hardware, fundamentalmente devido à fase de desenvolvimento dos referidos sistemas. Essas informações incluíam a capacidade de processamento e memória que seria exigida pelo *firmware*, bem como a carga de dados a ser processada e transmitida por meio da rede 4G. Dessa forma, alguns dos componentes do hardware foram sobredimensionados, visando garantir a operabilidade do sistema.

Considerando os requisitos de hardware, o primeiro componente selecionado para a construção da placa eletrônica foi o dispositivo principal de processamento de dados. A capacidade de processamento e de memória foram estimadas com base no fluxo máximo de dados que poderiam ser extraídos da ECU via barramento CAN. A seleção desse componente foi baseada nas seguintes características:

- Interface de comunicação CAN (requisito 1).
- Processador multi-core com velocidade de *clock* mínima de 1GHz (requisito 5).
- Memória RAM de ao menos 2 GB DDR3 (requisito 6).
- Memória flash de no mínimo 8 GB (requisito 6).
- Múltiplas interfaces de comunicação UART, USB e SPI para permitir a comunicação com os demais componentes do circuito.
- Possibilitar a utilização de um sistema operacional com *kernel* Linux para facilitar a implementação do *firmware*.

Com base nas características listadas, o dispositivo de processamento selecionado foi a placa eletrônica computacional modelo FZ3, fabricada pela empresa Shenzhen Myir Technology. Essa placa possui um Zynq UltraScale+ MPSoC (*Multiprocessor System-on-a-Chip*) composto por uma unidade lógica programável (*Kintex Ultrascale+* FPGA) e um subsistema de processamento (*ARM quad-core Cortex-A53 e dual-core Cortex-R5*) operando a uma frequência máxima de 1.5 GHz. A placa utiliza uma memória DDR4 SDRAM de 4 GB e tem diversas interfaces e recursos periféricos, como interface SD/MMC, USB 2.0 e 3.0, portas *Ethernet*, QSPI (*Quad Serial Peripheral Interface*), e UART (*Universal Asynchronous Receiver/Transmitter*).

A seleção dos demais componentes do hardware se baseou primariamente nas interfaces de comunicação necessárias. Dessa forma, o dispositivo escolhido para realizar a interface *Bluetooth* do sistema (requisito 4) foi o microcontrolador modelo ESP32-WROOM-32E da empresa Espressif Systems. Esse dispositivo é um microcontrolador Wi-Fi+BT+BLE (*Wi-fi*, *Bluetooth* e *Bluetooth Low Energy*) que possui 2 núcleos e frequência de *clock* ajustável de, no máximo, 240 MHz.

Para a habilitar o *upload* de dados via 4G (requisito 2) e a captação da localização do veículo (requisito 3), o módulo LTE EC25 Mini PCIe, fabricado pela empresa Quectel, foi escolhido. Esse modem apresenta taxas de transferência de 150 Mbps para *download* e 50 Mbps para *upload* de dados, suportando as tecnologias LTE, WCDMA, e GNSS. Além disso, para garantir melhor qualidade de sinal, duas antenas externa são utilizadas, uma para o GPS, modelo MEA-GPS-GG, da empresa Maxtena, e outra para o sinal 4G/LTE, modelo GSA.8841.A.105111, da empresa Taoglas. O acesso à rede 4G é feito utilizando um cartão SIM de uma operadora local.

O circuito de condicionamento de tensão (requisito 7) foi dimensionado considerando reguladores de tensão lineares e resistores de alta potência destinados à dissipação de calor, evitando o superaquecimento dos reguladores. Para proporcionar autonomia de energia (requisito 8) ao hardware, uma bateria de íon-lítio de 12 V e capacidade de 2200 mAh foi utilizada em conjunto com um circuito de gerenciamento de baterias (BMS).

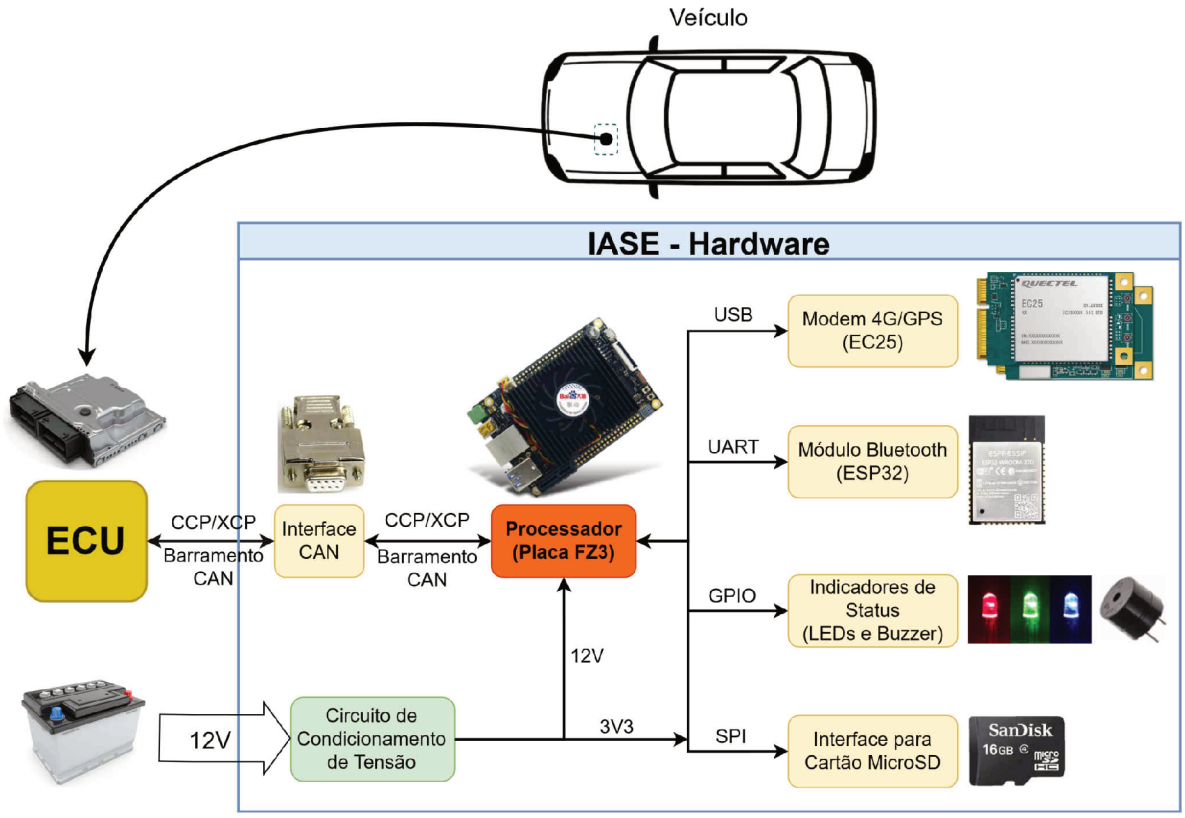
Além dos elementos citados, um conjunto de LEDs foi empregado para sinalizar o status do sistema diretamente na placa. Os demais componentes selecionados para o circuito são secundários e servem apenas para garantir o correto funcionamento e configuração dos elementos principais do hardware.

4.2.2 Arquitetura do Hardware

Com base nos componentes selecionados, a arquitetura do hardware é apresentada na Figura 4.4. A interface CAN, que realiza a conexão física entre a ECU e o processador, é composta por adaptadores do tipo DB9 e nenhum condicionamento de sinal é necessário. O processador (placa FZ3) é responsável por estabelecer a

comunicação e processar todas as informações provenientes dos diferentes módulos do sistema.

Figura 4.4 – Arquitetura do hardware.



Fonte: Elaborada pelo autor.

O módulo *Bluetooth* é responsável por gerenciar a comunicação com o *smartphone* para o controle de execução do sistema e funcionalidades de calibração e identificação de falhas. O módulo realiza a interface de comunicação entre o barramento UART do processador e a rede *Bluetooth* para conexão com o *smartphone*.

O modem 4G/GPS provê os dados de geolocalização e a conexão com o servidor de IoT para o sistema, comunicando-se com o processador por meio do barramento USB.

Um cartão de memória é responsável por armazenar o sistema operacional da placa, o *firmware* e os dados coletados, se necessário. Os LEDs e o *buzzer* permitem informar ao usuário o status do sistema. Por fim, os reguladores de tensão garantem a correta distribuição de energia para os componentes do hardware.

A Tabela 4.1 relaciona os componentes e circuitos do hardware com cada um dos requisitos elencados na Seção 4.1.2.

Tabela 4.1 – Relação entre requisitos e partes do hardware.

nº	Requisito	Parte do Hardware
1	Interface de comunicação CAN	Placa FZ3 e conector DB9
2	Tecnologia de comunicação 4G	Módulo EC25-AU e antenas
3	Tecnologia GPS	Módulo EC25-AU e antenas
4	Interface de comunicação Bluetooth	Módulo ESP32
5	Capacidade de processamento	Placa FZ3
6	Capacidade de armazenamento	Placa FZ3
7	Circuito de condicionamento de tensão	Reguladores de tensão
8	Autonomia de energia	Bateria íon-lítio

Fonte: Elaborada pelo autor.

4.2.3 Esquemático do Circuito e Layout da PCB

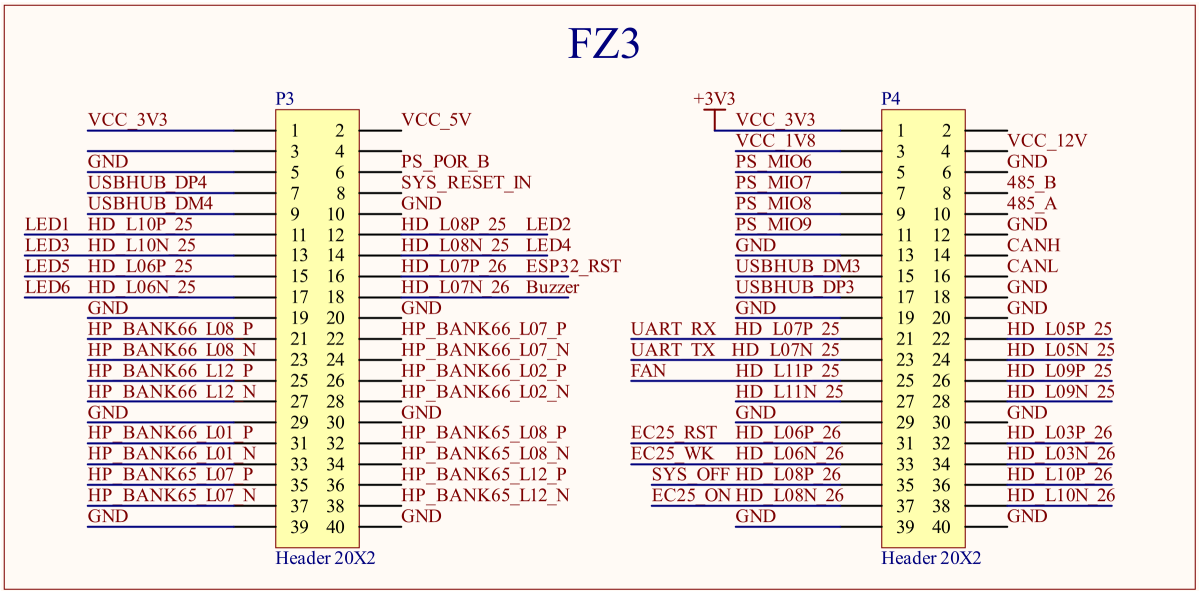
O desenvolvimento do diagrama esquemático e da estrutura física da PCB foi realizado utilizando um software de desenvolvimento para placas de circuito impresso. Nesta seção são apresentadas as partes principais do circuito.

O hardware/circuito pode ser dividido em 11 partes relevantes e praticamente independentes:

1. Interface da Placa FZ3.
2. Módulo EC25 e Conector do Cartão SIM.
3. Módulo ESP32.
4. Display de LEDs.
5. Buzzer.
6. Fontes de Energia e Reguladores.
7. Conectores CAN e SPI.
8. Controle de Hardware ON/OFF.
9. Interface Externa.
10. Circuito de Sinal Externo.
11. Controle da ventoinha.

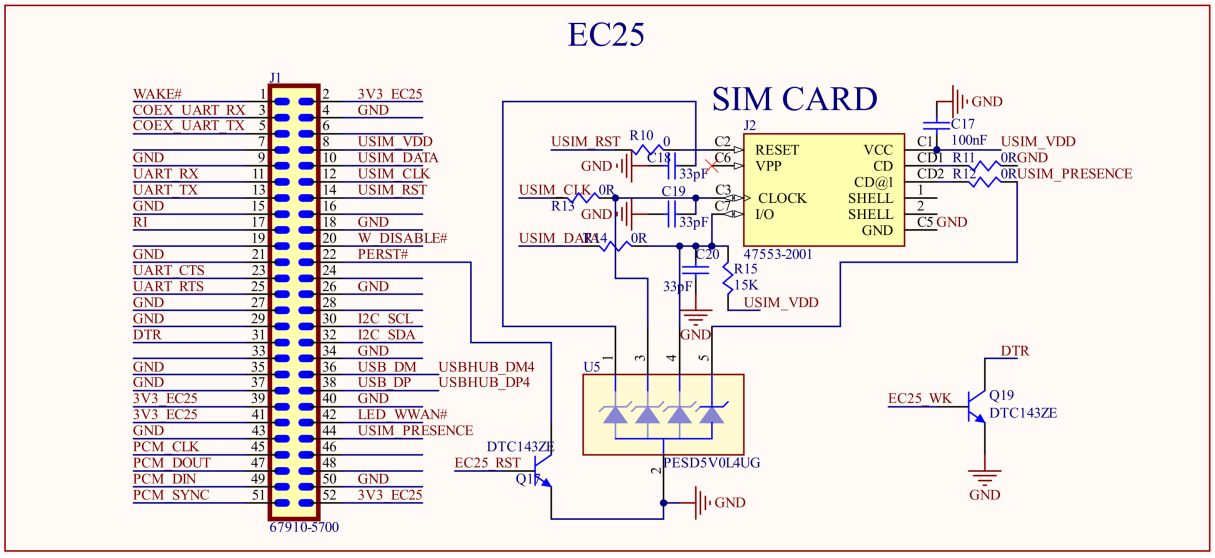
A placa FZ3 é capaz de estabelecer a interface com o sistema por meio de seus barramentos de entradas e saídas digitais, ligados fisicamente a dois conectores de 2x20 pinos. A Figura 4.5 apresenta as conexões utilizadas na GPIO da placa FZ3, projetadas para possibilitar a comunicação com a ECU do veículo e com os demais

Figura 4.5 – Diagrama esquemático do hardware de aquisição de dados - interface com a placa FZ3.



Fonte: Elaborada pelo autor.

Figura 4.6 – Diagrama esquemático do hardware de aquisição de dados - EC25 e conector do cartão SIM.



Fonte: Elaborada pelo autor.

módulos do circuito. A Tabela 4.2 resume as conexões da placa FZ3 com as demais partes do circuito.

O esquemático do módulo EC25 e do conector do cartão SIM é apresentado na Figura 4.6. Enquanto o conector do cartão SIM possui apenas conexões com a

Tabela 4.2 – Relação entre pinos da placa FZ3, sinais e partes do circuito.

Pinos	Sinal	Parte do Circuito
P3.7	USBHUB_DP4	EC25
P3.9	USBHUB_DM4	EC25
P3.11	HD_L10P_25/LED 1	Display de LEDs
P3.12	HD_L8P_25/LED 2	Display de LEDs
P3.13	HD_L10N_25/ LED 3	Display de LEDs
P3.14	HD_L8N_25/ LED 4	Display de LEDs
P3.15	HD_L06P_25/LED 5	Display de LEDs
P3.16	HD_L06P_25/ESP32_RST	ESP32
P3.17	HD_L06N_25/LED 6	Display de LEDs
P3.18	HD_L07N_26/Buzzer	Buzzer
P4.2	VCC_12V	Fontes de Energia
P4.5	PS_MIO6	Interface externa
P4.7	PS_MIO7	Interface externa
P4.9	PS_MIO8/ESP32_RXD	ESP32
P4.11	PS_MIO9/ESP32_TXD	ESP32
P4.14	CANH	Conector CAN
P4.16	CANL	Conector CAN
P4.21	UART_RX	EC25
P4.23	UART_TX	EC25
P4.25	HD_L11P_25/FAN	Ventoinha
P4.31	EC25_RST	EC25
P4.33	EC25_WK	EC25
P4.35	SYS_OFF	Controle ON/OFF
P4.37	EC25_ON	EC25

Fonte: Elaborada pelo autor.

interface *Mini PCIExpress* do módulo EC25, o modem está conectado aos barramentos de comunicação da placa FZ3 (USB e UART) e ao circuito de condicionamento de tensão.

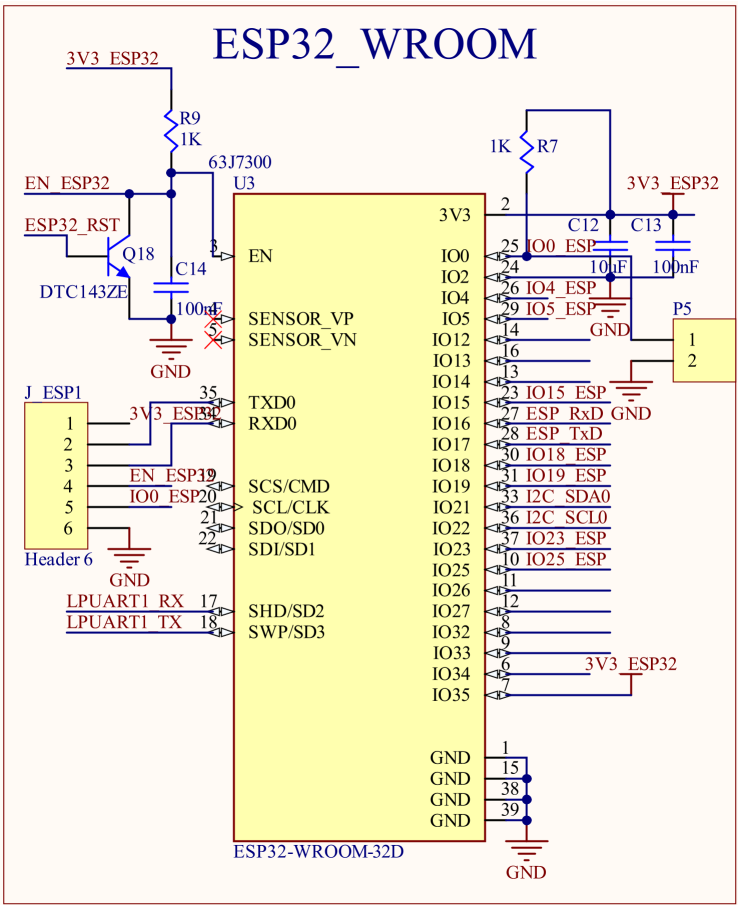
A configuração do circuito para o ESP32 (Figura 4.7) apresenta uma interface externa para gravação do microcontrolador, comunicação UART com a placa FZ3 e conexão com o circuito de condicionamento de tensão do hardware.

O circuito de condicionamento de energia (Figura 4.8) se destina a fornecer tensões de 12 V para a placa FZ3 e 3,3 V para o modem EC25, para o módulo ESP32 e para os demais componentes do circuito. Três reguladores são utilizados para ajustar a tensão proveniente das baterias e dois resistores de alta potência auxiliam na dissipação de calor do circuito.

4.2.4 Produção, Montagem e Testes de Bancada

A produção das Placas de Circuito Impresso (PCBs) foi conduzida por um fabricante contratado, enquanto a montagem dos componentes foi executada pela equipe

Figura 4.7 – Diagrama esquemático do hardware de aquisição de dados - ESP32.



Fonte: Elaborada pelo autor.

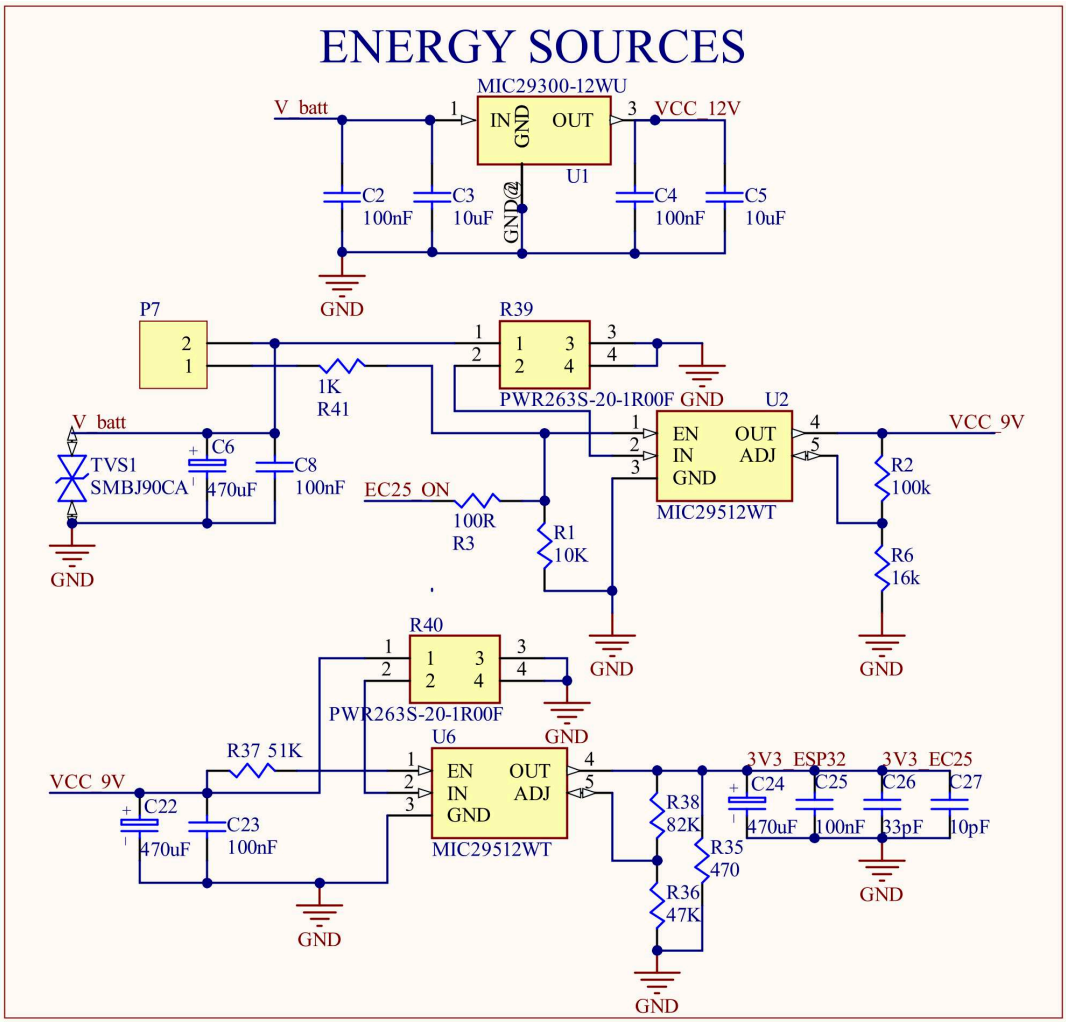
responsável pelo projeto.

Com o propósito de verificar a disposição correta das conexões nas PCBs e assegurar o fornecimento adequado de tensão para todos os elementos do hardware, testes de bancada foram empregados. Inicialmente foram realizadas inspeções visuais para identificar possíveis problemas na placa. Utilizando uma fonte chaveada com diferentes níveis de tensão e um multímetro, testes de continuidade e verificação da distribuição adequada dos sinais de tensão na placa foram conduzidos antes da incorporação dos componentes principais do circuito, com o intuito de mitigar possíveis danos a esses componentes. Por fim, testes de comunicação entre os principais elementos do circuito foram efetuados.

A Figura 4.9 mostra a primeira versão da placa após a montagem dos componentes. Após os testes de bancada realizados e testes de integração com o *firmware*, os seguintes problemas foram identificados:

- Inversão entre os pinos do coletor e do emissor do transistor DTC143ZE devido a utilização da versão incorreta do *footprint* durante o projeto da PCB.

Figura 4.8 – Diagrama esquemático do hardware de aquisição de dados - circuito de condicionamento de tensão.

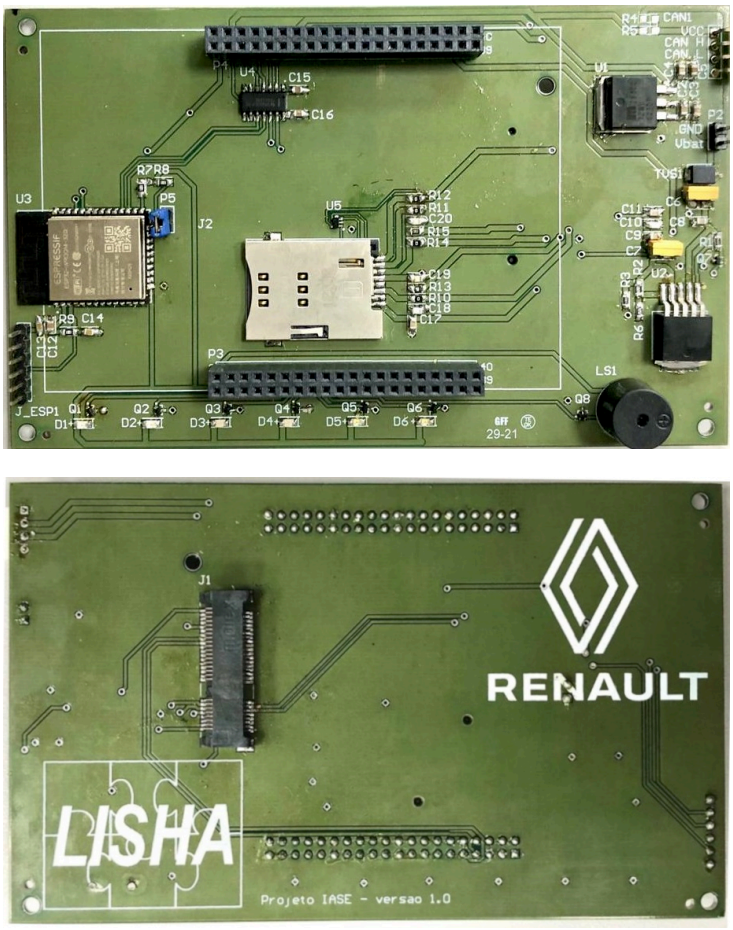


Fonte: Elaborada pelo autor.

- Falha na comunicação da placa FZ3 com o módulo ESP32 devido a informação incorreta presente no mapa de identificação dos pinos da FZ3.
- Posição inadequada do conector da EC25 e falta do suporte de fixação da *Mini PCIExpress*.
- Distância entre os conectores da placa FZ3 com 1 mm de diferença.

A segunda versão da PCB (Figura 4.10) foi produzida para corrigir as falhas identificadas na primeira e garantir a disponibilidade de todas as funcionalidades de hardware para o sistema. Após os testes de bancada, testes de integração da placa com o *firmware* e testes funcionais no veículo, os seguintes problemas, modificações e melhorias necessárias foram identificados:

Figura 4.9 – Primeira versão da PCB.



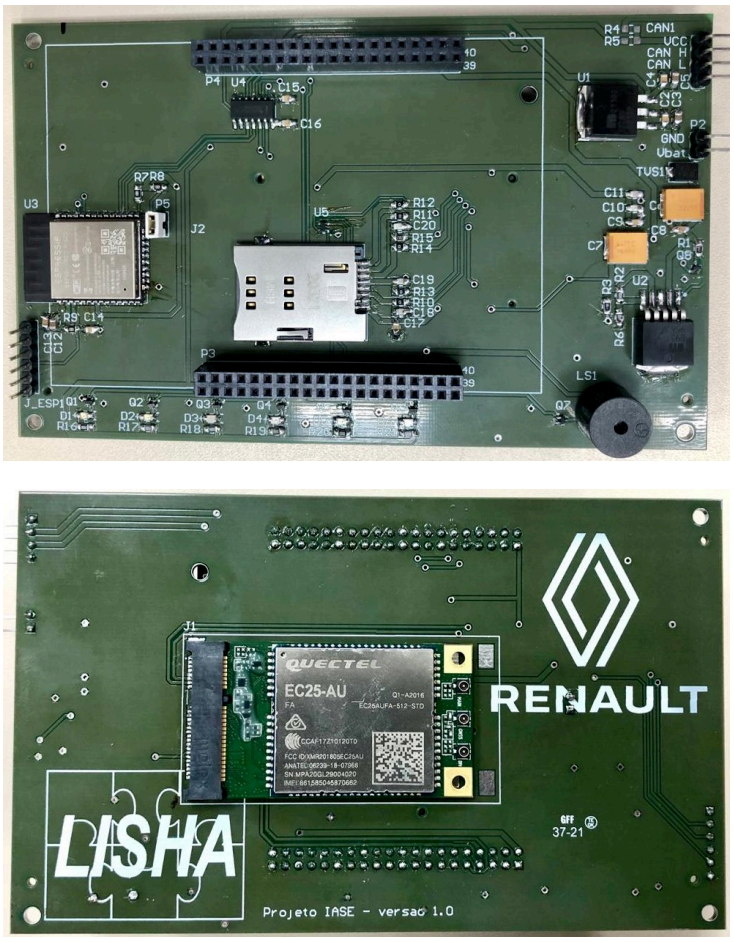
Fonte: Do autor.

- Aquecimento dos reguladores de tensão do circuito de condicionamento de tensão durante os testes de integração do sistema.
- Aquecimento do módulo EC25 durante os testes de integração do sistema.
- Impossibilidade de reiniciar os módulos EC25 e ESP32 individualmente via *firmware* em caso de falha.
- Necessidade de substituição dos conectores por conectores de padrão industrial.
- Necessidade de alteração da posição do conector do cartão SIM para facilitar sua substituição.

Além das alterações necessárias para corrigir os problemas identificados na placa, a implementação de novas características foram solicitadas para atender a demandas específicas do processo de testagem e validação dos veículos:

- Adição de uma conexão externa para ligar a placa.

Figura 4.10 – Segunda versão da PCB.

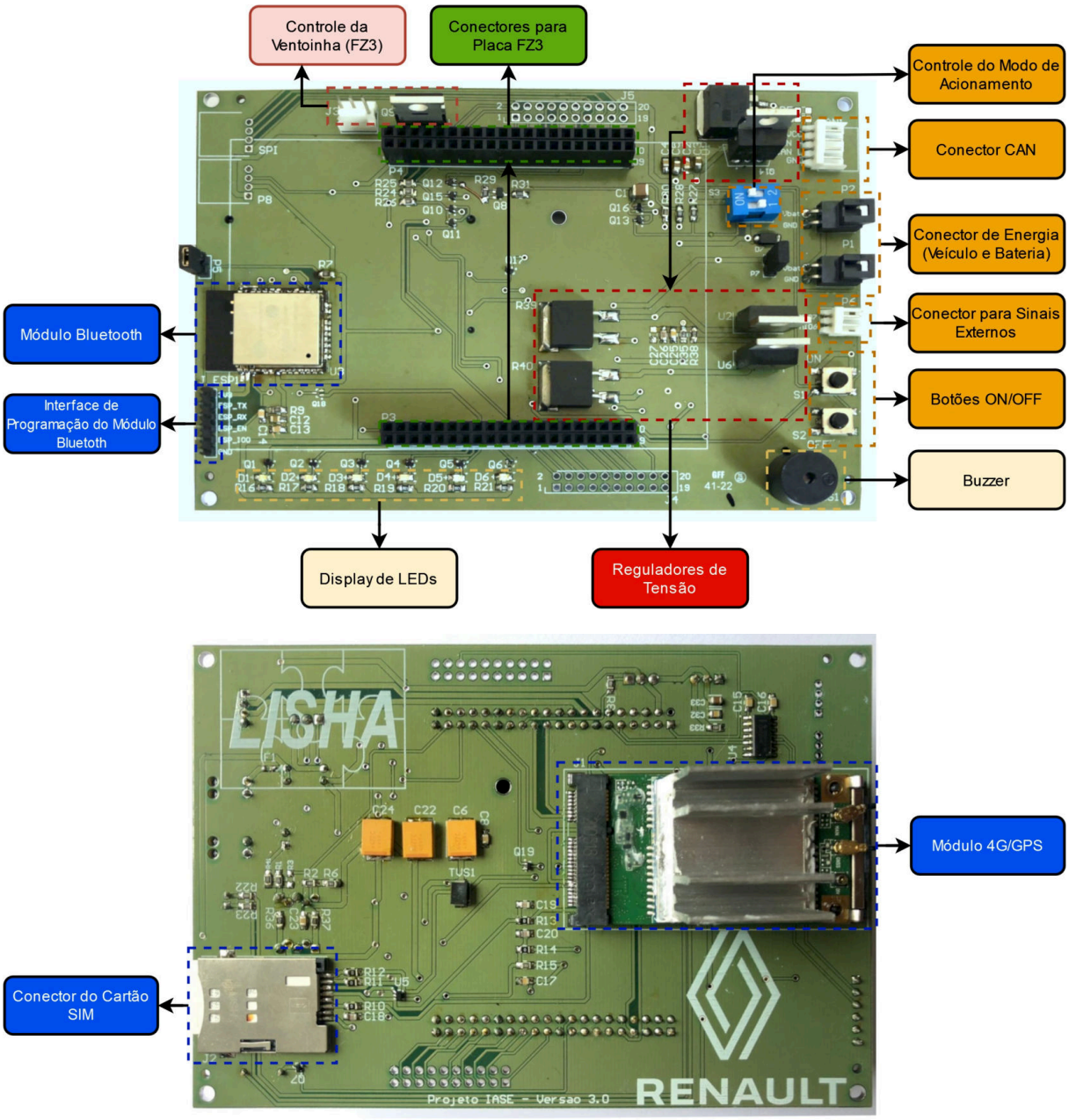


Fonte: Do autor.

- Adição de uma conexão externa para indicar o fim do experimento.
- Implementação de um novo circuito de controle de acionamento (*ON/OFF*) do hardware utilizando sinais internos ou externos.
- Implementação de um circuito de controle para a ventoinha da placa FZ3.
- Adição de novos pontos de conexão com as entradas e saídas da placa FZ3.

A partir das modificações e melhorias listadas, a terceira versão da PCB foi desenvolvida. A Figura 4.11 mostra a versão final da placa, destacando as principais partes do circuito. Uma caixa de acrílico foi confeccionada para permitir a fixação e proteção adequada do hardware e conectores. A Figura 4.12 mostra o hardware instalado na caixa e os respectivos elementos do conjunto. Na Figura 4.13 são apresentados os conectores de energia, comunicação e controle do hardware e na Figura 4.14 são apresentadas as antenas e conectores utilizados para captação dos sinais de GPS e 4G.

Figura 4.11 – Terceira versão da PCB.

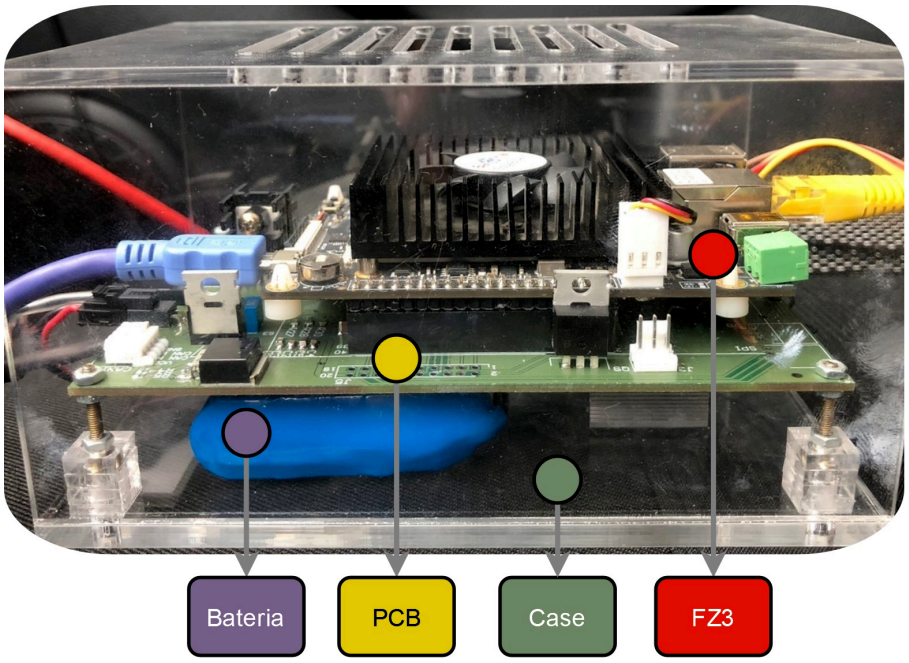


Fonte: Elaborada pelo autor.

4.3 CONFIGURAÇÃO DO HARDWARE, SISTEMA OPERACIONAL E FIRMWARE DO IASE

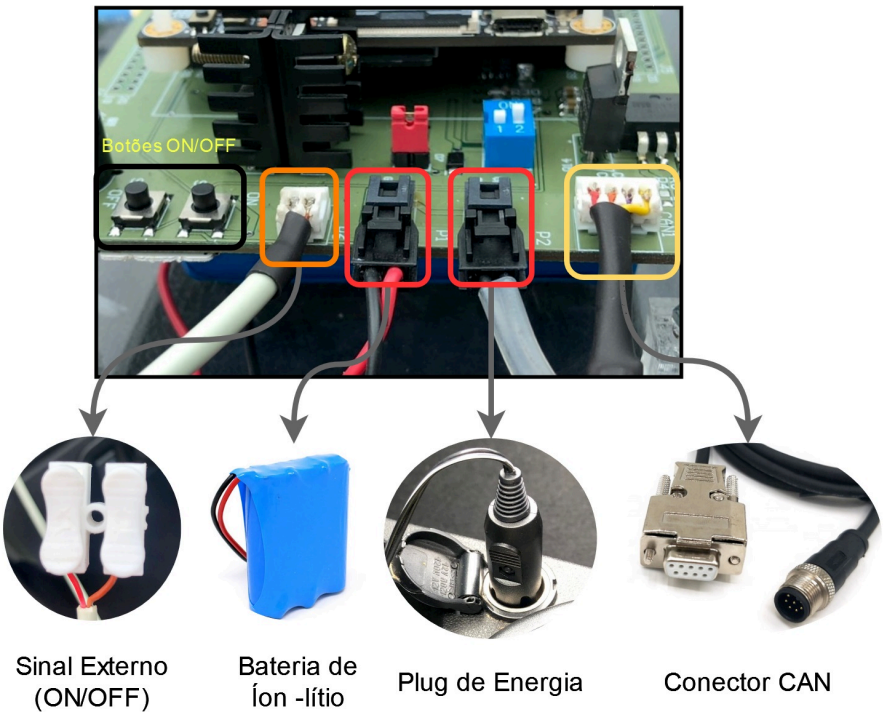
A configuração do hardware se refere à configuração do MPSoC Zynq UltraScale+, presente na placa de processamento FZ3, através do software Vivado Design Suite, que é um conjunto de software produzido pela empresa Xilinx para síntese e análise de projetos de linguagem de descrição de hardware (AMD, 2023). Para possibilitar a comunicação com todos os elementos do hardware, portas de comunicação UART

Figura 4.12 – Hardware completo instalado na caixa de acrílico.



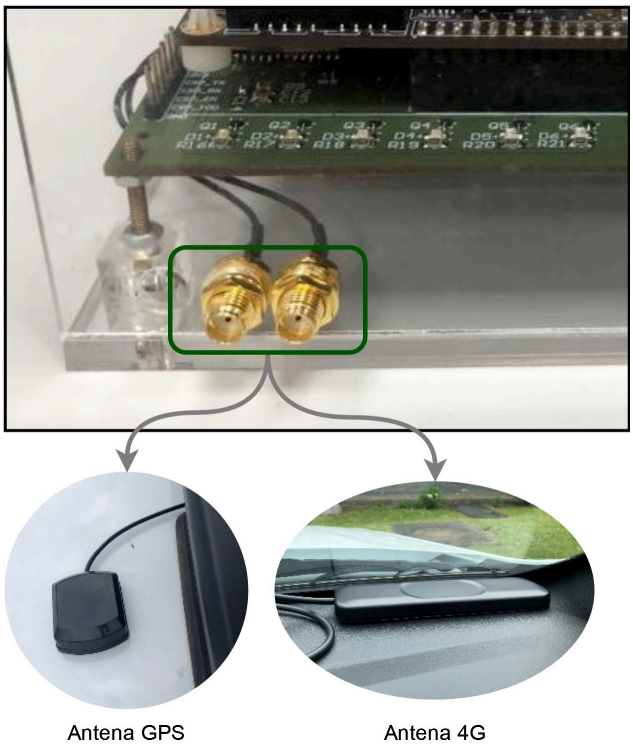
Fonte: Elaborada pelo autor.

Figura 4.13 – Conectores da PCB e respectivas funções.



Fonte: Elaborada pelo autor.

Figura 4.14 – Conectores e antenas utilizados para captação dos sinais de GPS e 4G.



Fonte: Elaborada pelo autor.

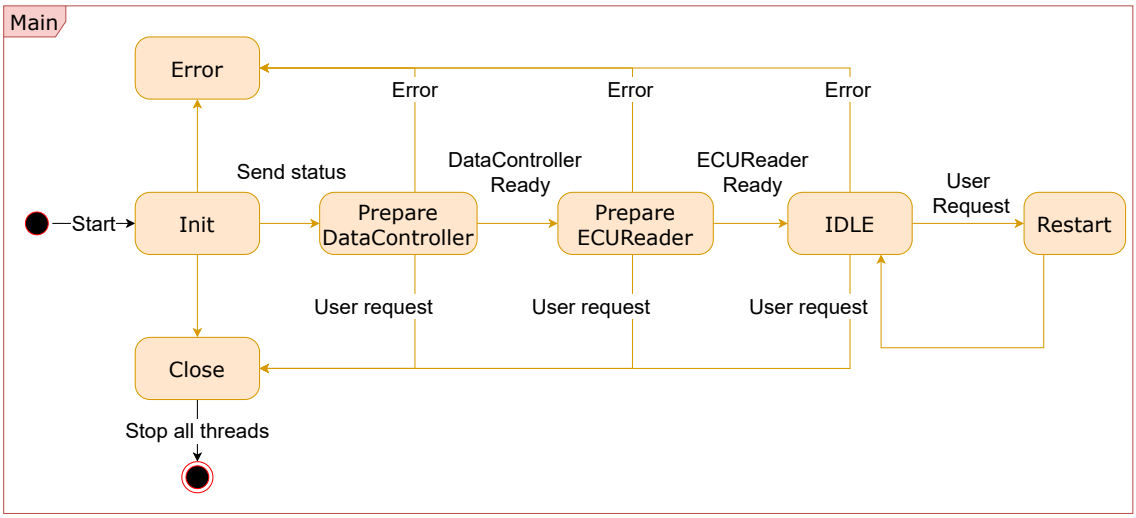
e USB foram habilitadas e direcionadas para os pinos adequados da FZ3 através da FPGA do MPSoC. Da mesma forma, para permitir o acionamento dos LEDs e buzzer do hardware desenvolvido, assim como o recebimento de interrupções externas, as GPIOs da placa foram configuradas de acordo com o diagrama esquemático da placa de processamento e do hardware desenvolvido. Detalhes sobre o processo de configuração do MPSoC Zynq UltraScale+ podem ser verificados em Xilinx (2021).

A placa FZ3 suporta o Linux Board Support Package da Xilinx, chamado Petalinux, que constrói e gera automaticamente um *kernel* Linux, incluindo os *drivers* de dispositivo necessários. A versão do sistema operacional utilizada para o IASE foi a versão 2020.1 do Petalinux, que é baseada no *kernel* Linux 5.4.0-xilinx-v2020.1. A compilação do sistema operacional foi realizada utilizando o pacote de ferramentas de software do Petalinux e o arquivo de configuração do hardware gerado pelo Vivado para implementação das modificações do MPSoC (HARTFIEL, 2021).

O *firmware* IASE gerencia todas as funcionalidades executadas no hardware. Ele é programado em C++ e controla a interface com a ECU, o servidor de comunicação e a interface do usuário. Uma máquina de estados gerencia a lógica do *firmware* dependendo do estado do sistema. A Figura 4.15 ilustra o diagrama de estados UML da máquina de estados principal do *firmware* IASE. Os retângulos representam os estados, enquanto as setas com os nomes dos eventos representam as transições entre

os estados. Após iniciar o *firmware*, todas as classes (i.e., DataController, ECUReader, etc.) são inicializadas e verificadas. Em seguida, o *firmware* permanece no modo IDLE, lendo e escrevendo na ECU até que o usuário solicite um novo ciclo ou finalize o processo (BEDRETCHUK et al., 2023). Informações detalhadas sobre o *firmware* desenvolvido para o IASE podem ser encontradas em García et al. (2022) e Igarashi (2022).

Figura 4.15 – Máquina de estados principal do *firmware* do IASE.



Fonte: Bedretchuk et al. (2023).

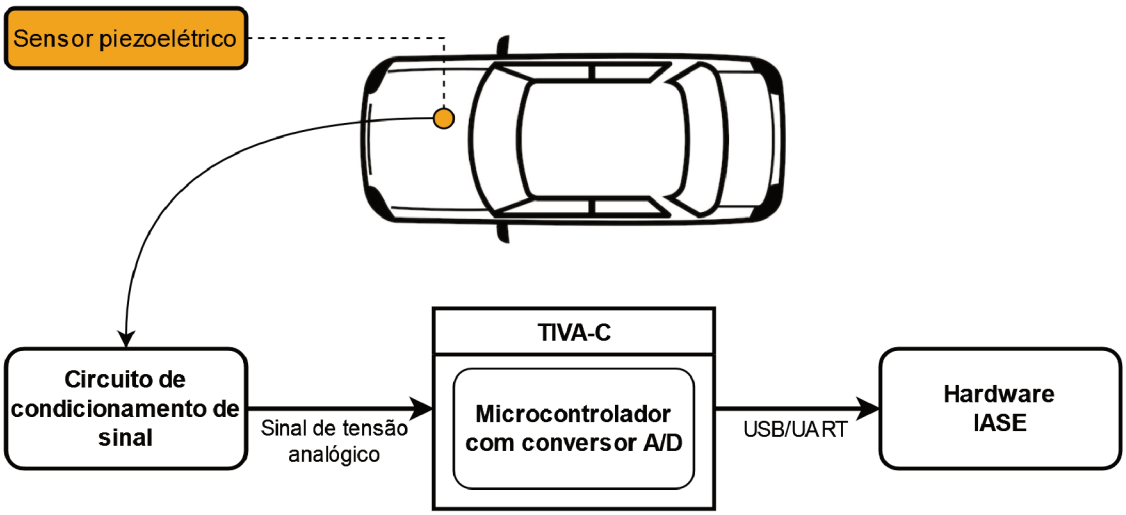
4.4 SISTEMA DE DETECÇÃO DE KNOCK NOISE

Para realizar a detecção do ruído de pré-ignição, o sistema desenvolvido utiliza o sensor piezoelétrico presente na carcaça do motor do protótipo veicular. O sinal proveniente desse sensor é o mesmo utilizado pela ECU do veículo para realizar a detecção de *knock noise* e atuar no controle de ignição do motor a fim de reduzir as ocorrências desse fenômeno.

Com o propósito de efetuar a aquisição e a análise de dados provenientes do sensor piezoelétrico, para a detecção de ocorrências de *knock noise* durante sua operação, um circuito suplementar foi concebido e implementado. A Figura 4.16 ilustra o diagrama de blocos correspondente ao sistema empregado para identificar as ocorrências de *knock noise*.

Devido à indisponibilidade de informações específicas sobre o sensor piezoelétrico instalado no veículo, medições iniciais foram realizadas para identificar as características do sinal gerado por ele. O sensor produz variações de tensão que oscilam entre -3 V e 3 V de acordo com os ruídos que ocorrem no *powertrain* do veículo.

Figura 4.16 – Diagrama de blocos referente ao sistema de captação de dados do sensor piezoelétrico.



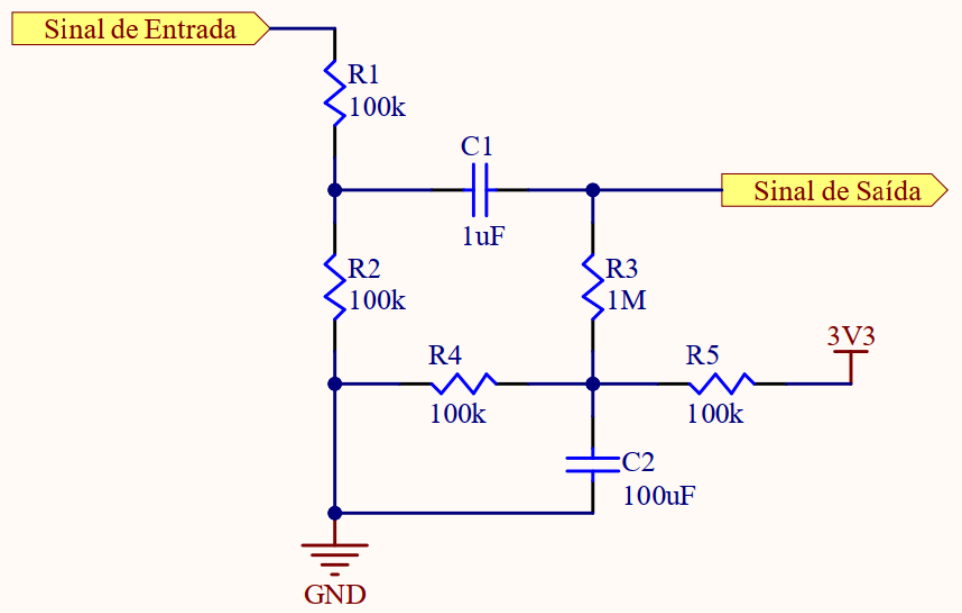
Fonte: Elaborada pelo autor.

Conforme apresenta a Seção 2.4, os modos de vibração relacionados ao *knock noise* estão dentro da faixa de 6 a 25 kHz. Para que seja possível mapear esses sinais digitalmente, de acordo com o Teorema de Nyquist, a taxa de amostragem do sinal deve ser de, no mínimo, 50 kHz. Dessa maneira, o microcontrolador selecionado para realizar a conversão analógica-digital (AD) do sinal foi o Arm Cortex-M4F, presente no *LaunchPad* EK-TM4C123GXL (TIVA-C), da Texas Instruments, que possui um ADC de 12 bits e frequência de amostragem máxima de 1 Msps. Apesar da baixa resolução do conversor, não são esperadas perdas significativas de informação, visto que os picos de intensidade a serem monitorados apresentam ampla oscilação. Os valores lidos pelo microcontrolador são transmitidos diretamente para o hardware IASE, de modo que o *LaunchPad* atua apenas como um conversor AD para o sistema. A utilização desse microcontrolador intermediário se deve a indisponibilidade de um conversor AD no hardware desenvolvido que seja capaz de capturar os dados na frequência necessária para a detecção da ocorrência do fenômeno de pré-ignição.

Sabendo que a faixa de tensão de operação da entrada analógica do microcontrolador escolhido tem seus limites entre 0 V e 3,3 V, um circuito de condicionamento de sinal é necessário para adaptar o sinal do piezoelétrico à faixa de tensão aceita pelo conversor AD do microcontrolador. A Figura 4.17 apresenta o esquemático desse circuito, que altera as variações de tensão da faixa de -3 V a 3 V para 0 V a 3 V.

Uma consequência da utilização desse circuito de condicionamento é a atenuação do sinal em determinadas faixas de frequência, em decorrência do uso de capacitores para a alteração dos níveis de tensão. Para evitar a atenuação do sinal na faixa de frequência de interesse para o sistema de detecção de *knock noise*, os

Figura 4.17 – Circuito de condicionamento para o sinal do sensor piezoelétrico.



Fonte: Elaborada pelo autor.

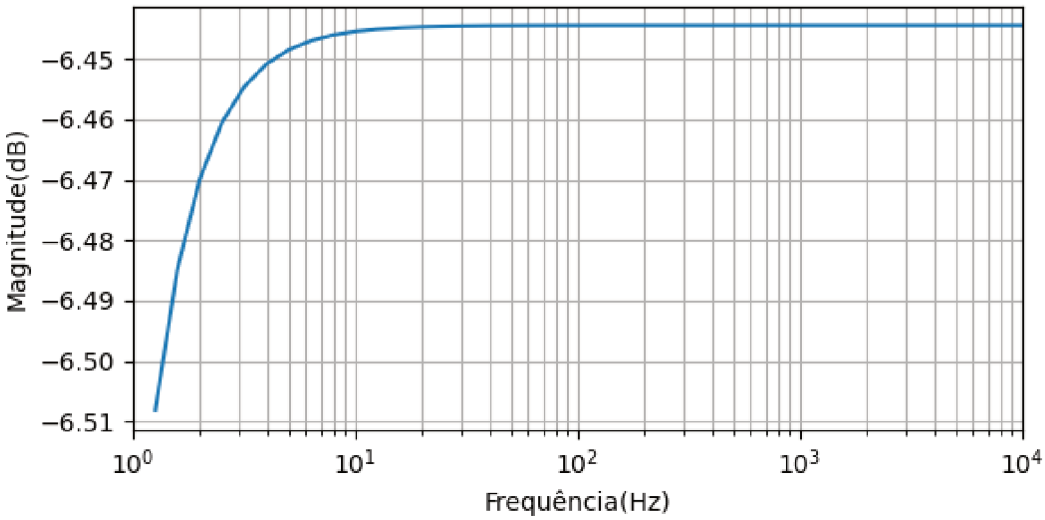
capacitores foram selecionados considerando a curva de resposta em frequência do circuito. A Figura 4.18 apresenta a curva de atenuação, em decibéis, em função da frequência do sinal de entrada. O nível mínimo de atenuação, em torno de -6.445 dB, é provocado intencionalmente pelos resistores de divisão de tensão conectados ao sinal de entrada. Atenuações mais elevadas são percebidas apenas na faixa de 1 a 10 Hz, não provocando alterações na faixa de frequência de interesse.

Com o hardware de aquisição de dados montado, testes iniciais de aquisição do sinal analógico do sensor foram realizados a fim de verificar a qualidade do sinal digital resultante. A melhor qualidade do sinal digital foi obtida utilizando amostras resultantes de uma média simples de 4 amostras reais do sinal capturadas a uma frequência aproximada de 460 kHz. Como consequência, a frequência de aquisição das amostras resultantes foi fixada em aproximadamente 115 kHz, atendendo a demanda mínima de 50 kHz para a detecção da ocorrência de *knock noise* no motor.

A frequência fundamental de ocorrência do *knock noise* foi determinada utilizando a Equação 2.1, considerando o modo de vibração fundamental, igual a 1,841, o diâmetro dos cilindros do veículo de testes, igual a 71 mm, e a velocidade do som de 966 m/s, para uma câmara de combustão com temperatura máxima de 2500 K (OFNER et al., 2022). O valor de frequência obtido foi de aproximadamente 7.973 Hz.

Utilizando a frequência fundamental, um filtro FIR do tipo passa-faixa foi projetado para filtrar o sinal proveniente do piezoelétrico em tempo real. A faixa de frequência permitida pelo filtro foi configurada de 7173 Hz a 8773 Hz, com um *ripple* de 5 dB,

Figura 4.18 – Resposta em frequência do circuito de condicionamento de sinal do sensor piezoelétrico.



Fonte: Elaborada pelo autor.

enquanto as faixas de frequência abaixo de 6373 Hz e acima de 9573 Hz foram configuradas para sofrer atenuação de -20 dB. Ajustes dos parâmetros do filtro foram realizados empiricamente, visando permitir a melhor qualidade de filtragem do sinal e evitar o excesso de coeficientes.

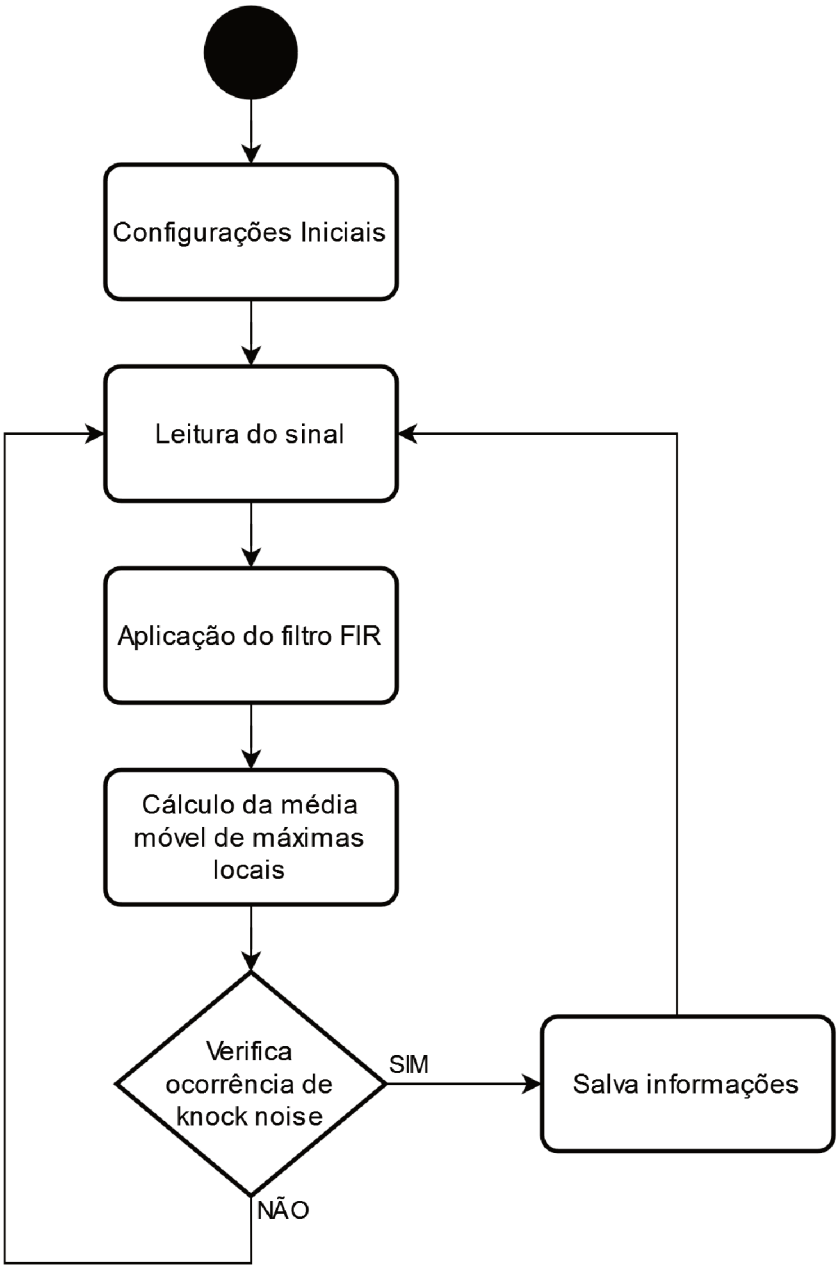
Para realizar a detecção de ocorrências de *knock noise*, o sistema deve ser capaz de detectar desvios anormais na amplitude do sinal de vibração captado e filtrado. Com esse objetivo, uma média móvel dos máximos locais do sinal filtrado foi construída para servir de referência aos desvios de amplitude característicos do ruído de detonação. A partir dessa média, um fator de multiplicação é aplicado para determinar o limite para detecção de ocorrências de *knock noise*. Os valores utilizados para cada um dos parâmetros do algoritmo de detecção foram definidos empiricamente a partir de dados reais obtidos do sensor piezoelétrico durante testes de rodagem do veículo.

A Figura 4.19 mostra um fluxograma contendo as etapas principais da aplicação desenvolvida para identificar as ocorrências de *knock noise* a partir do sinal do sensor piezoelétrico do veículo. Primeiramente são realizadas configurações iniciais relacionadas à inicialização de variáveis, configuração da porta de comunicação e abertura de arquivos para armazenamento de dados. Na sequência a leitura de uma amostra do sinal é feita através da comunicação serial USB. A aplicação do filtro FIR calcula o valor resultante do sinal considerando o histórico de dados do sinal. A média móvel de

máximos locais é determinada a partir de um conjunto de valores máximos detectados a cada N amostras recebidas, desprezando valores que estão acima do limite de detecção de *knock noise*. Com base nesse limite, a ocorrência de *knock noise* é verificada. Sempre que uma ocorrência de *knock noise* é detectada, o valor da intensidade do sinal e o momento de ocorrência são salvos em um arquivo CSV.

A aplicação de detecção do ruído de pré-ignição foi construída em linguagem C++ e o código fonte desenvolvido é apresentado no Apêndice A.

Figura 4.19 – Fluxograma de processos para detecção de *knock noise*.



Fonte: Elaborada pelo autor.

4.5 PROCEDIMENTOS DE TESTE E VALIDAÇÃO DO HARDWARE

Com o objetivo de avaliar a integridade operacional e realizar uma análise quantitativa do desempenho do dispositivo de hardware concebido, testes de funcionalidade e desempenho do sistema foram conduzidos.

Para os testes relacionados ao IASE, que envolvem a coleta de dados provenientes da ECU veicular, os ensaios conduzidos utilizaram um experimento que explorou o número máximo de variáveis suportadas pelo barramento CAN, visando analisar sua capacidade de processamento e *upload* de dados em tempo real para o servidor de IoT.

A capacidade de processamento e transferência de dados foi avaliada determinando-se a quantidade de *bytes* por segundo transferida pela rede 4G para o servidor. Um código específico foi implementado no *firmware* para monitorar o *timestamp* de envio de cada novo pacote de dados. A partir dos dados coletados, foi possível calcular a quantidade de *bytes* em cada pacote de dados e determinar a velocidade de processamento e *upload* de dados do sistema.

A utilização de CPU e memória RAM foi monitorada com a execução de um programa paralelo ao *firmware* do sistema. O programa desenvolvido é capaz de registrar a cada segundo a porcentagem de utilização da CPU e da memória RAM do hardware desenvolvido, enquanto o *firmware* executa a captação, o processamento e o envio de dados. Essas informações foram coletadas durante a execução do *firmware* do IASE e do programa de monitoramento de *knock noise*, a fim de avaliar o desempenho do hardware durante a execução individual e simultânea dos algoritmos.

O consumo de energia do hardware foi verificado utilizando um resistor *shunt* (e.g. 0,1 Ohms) em série com o equipamento e um osciloscópio monitorando a tensão sobre ambos os terminais desse resistor. Com o sistema operando, a oscilação de tensão sobre o resistor foi gravada, permitindo a análise e o cálculo da potência consumida pelo equipamento.

4.6 PROCEDIMENTOS DE TESTE E ANÁLISE DO SISTEMA DE MONITORAMENTO DE KNOCK NOISE

A avaliação do desempenho do sistema de detecção de *knock noise* desenvolvido foi realizada comparando os resultados obtidos com as informações provenientes da ECU veicular referente à ocorrência do ruído de detonação, ou seja, utilizando como parâmetro a detecção do fenômeno de pré-ignição realizada pela ECU. Para isso, tanto o *firmware* de aquisição de dados da ECU quanto o algoritmo de captura e análise do sinal do sensor piezoelétrico foram executados simultaneamente.

Para permitir a análise dos dados provenientes do sensor piezoelétrico, coletas de dados foram realizadas armazenando os valores de cada amostra de sinal. Com os

conjuntos de dados coletados, gráficos do sinal original, de seu espectro de frequências e do sinal filtrado foram gerados para a realização de análises visuais. Utilizando como parâmetro as ocorrências do ruído de detonação indicadas pela ECU do veículo, diferentes filtros foram aplicados ao sinal original para avaliar o comportamento do sinal resultante. Apesar da identificação de pequenas oscilações relacionadas às características dos filtros, a frequência mais adequada para a detecção da ocorrência de *knock noise* foi de 7.973 Hz, conforme especificado na Seção 4.4.

O desempenho do sistema foi determinado comparando a quantidade de ocorrências detectadas através do sinal do sensor piezoelétrico e a quantidade informada pela ECU do veículo. O momento em que as ocorrências do ruído foram detectadas também foi considerado para estimar a assertividade do algoritmo.

5 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados do trabalho, incluindo o desempenho do hardware desenvolvido, os testes de detecção de *knock noise* e a comparação das características do hardware em relação aos sistemas comerciais.

5.1 DESEMPENHO DO HARDWARE

O desempenho do hardware foi avaliado considerando as duas principais aplicações expostas nesse trabalho. Nessa seção são apresentados os resultados relacionados à aquisição de dados da ECU veicular, à detecção de *knock noise* através do sensor piezoelétrico e, por fim, à operação de ambos os sistemas simultaneamente.

5.1.1 Desempenho do Hardware para o IASE

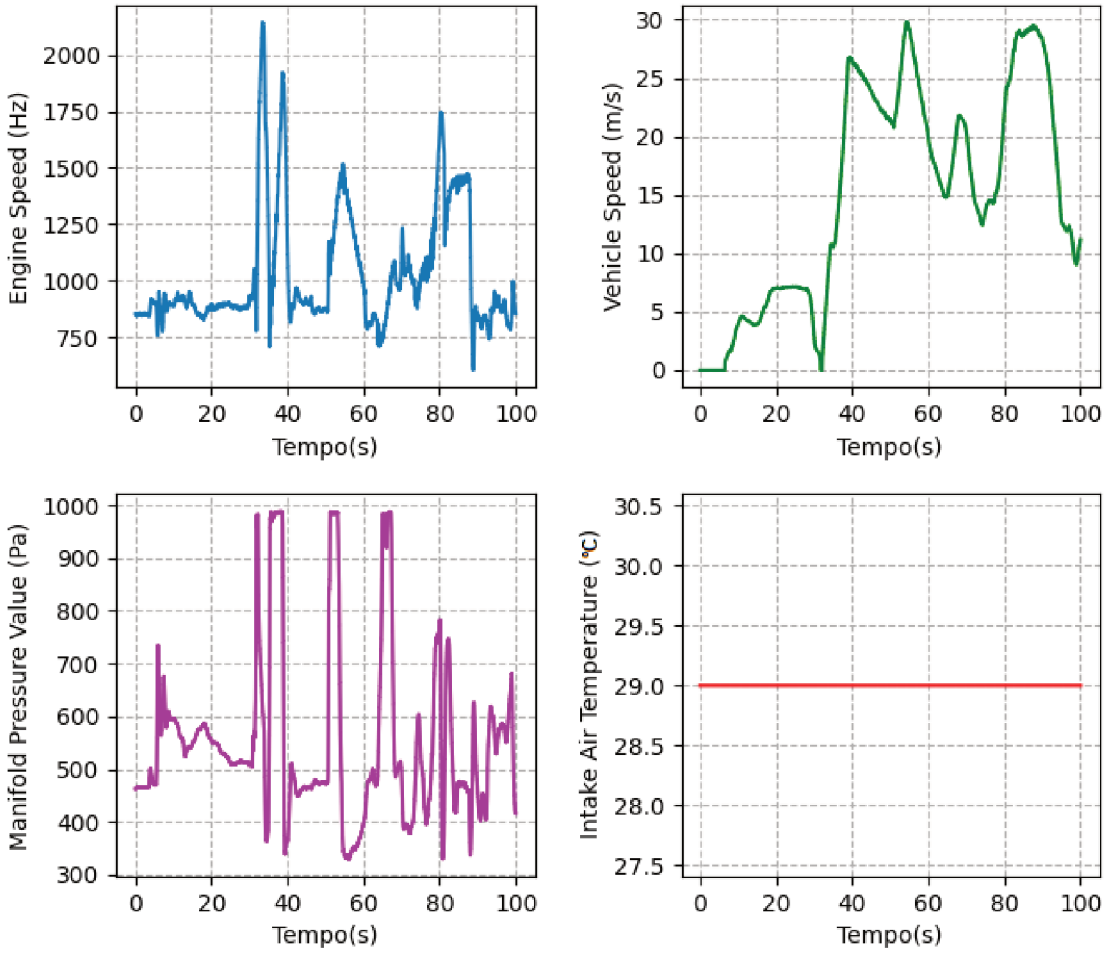
A partir da implementação do *firmware* e da programação e configuração dos módulos do hardware desenvolvido, testes iniciais de captação e envio de dados foram realizados a fim de comprovar o correto funcionamento do sistema. Utilizando o carro de testes em conjunto com o equipamento de aquisição de dados (conectado à ECU) e um experimento configurado, a coleta, processamento e envio de dados ao servidor de IoT pôde ser realizada. A Figura 5.1 apresenta quatro gráficos de diferentes variáveis contendo os dados extraídos do servidor instantes após o sistema ser iniciado, comprovando o adequado funcionamento do hardware.

O limite de aquisição de dados do hardware está relacionado ao barramento CAN e ao protocolo (CCP ou XCP) utilizado pela ECU para a comunicação com os equipamentos de teste e, portanto, a capacidade máxima de aquisição de dados do sistema IASE deve ser igual à capacidade dos demais equipamentos comerciais que utilizam os mesmos barramentos de comunicação. No intuito de avaliar essa capacidade de aquisição de dados do equipamento, um experimento contendo o número máximo de variáveis com os menores períodos de aquisição possíveis foi realizado. O experimento gerado para os testes tem como base o mesmo cálculo de utilização do barramento de comunicação dos sistemas comerciais. A configuração do experimento é apresentada na Tabela 5.1.

Utilizando o experimento exibido na Tabela 5.1, o sistema demonstrou capacidade suficiente para capturar, processar e enviar os dados provenientes da ECU do veículo para o servidor de IoT. Dessa forma, foi possível identificar que o gargalo do sistema de aquisição de dados está diretamente relacionado ao barramento de comunicação utilizado, que é definido pelo tipo de ECU instalada no protótipo veicular.

Durante a execução dos testes, o tempo total de aquisição, processamento e *upload* de dados foi monitorado. A Figura 5.2 apresenta a quantidade cumulativa de

Figura 5.1 – Variáveis capturadas do servidor durante os testes de rodagem do veículo utilizando o IASE.



Fonte: Elaborada pelo autor

Tabela 5.1 – Detalhes da configuração das variáveis do experimento.

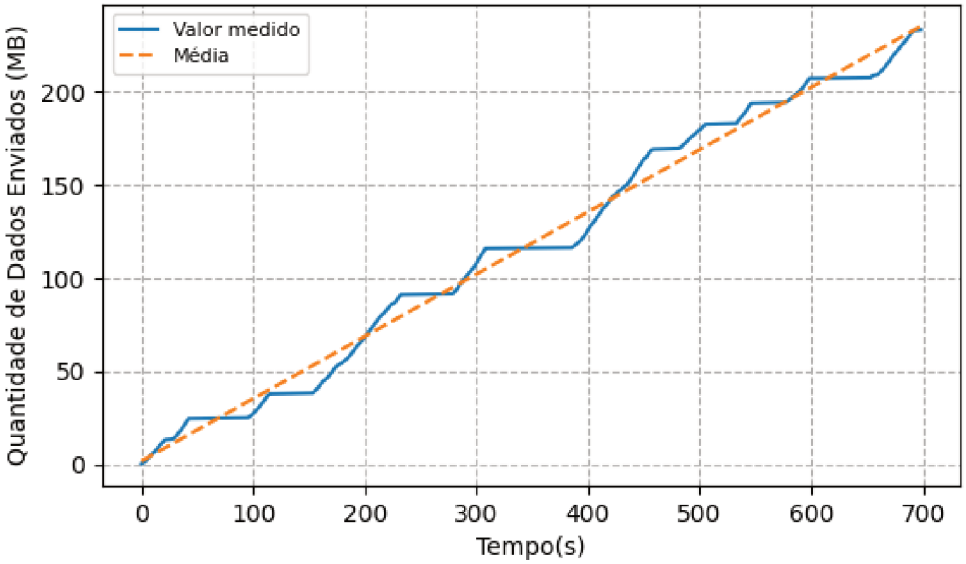
Período de Aquisição	Número de Variáveis
4 ms	28
5 ms	24
10 ms	60
100 ms	54
Sincronizada com o cilindro 1	23
Sincronizada com o cilindro 2	23
Sincronizada com o cilindro 3	23
Sincronizada com o cilindro 4	23

Fonte: Elaborada pelo autor.

dados (eixo y), em megabytes, enviados ao servidor ao longo do tempo (eixo x). A velocidade média com a qual os dados são recebidos no servidor é de aproximadamente

330 kB/s. Nenhum atraso cumulativo relacionado ao processamento e envio de dados foi identificado.

Figura 5.2 – Quantidade de bytes recebidos, processados e enviados pelo sistema ao longo do tempo.



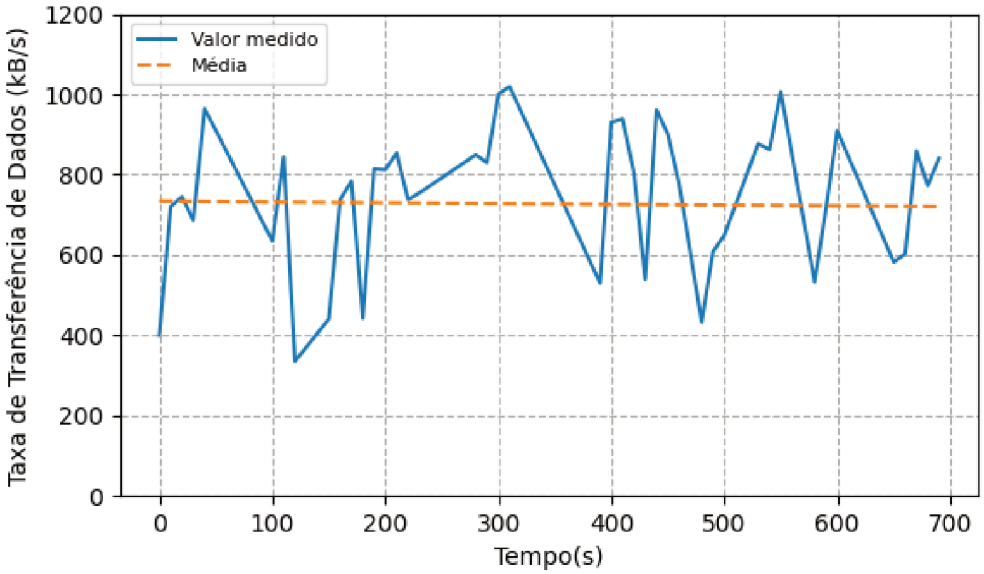
Fonte: Elaborada pelo autor

A taxa de transferência de dados realizada por meio do modem 4G foi monitorada isoladamente, uma vez que essa variável está relacionada diretamente com a qualidade do sinal da operadora utilizada no local onde o sistema está operando. A Figura 5.3 mostra a velocidade de transferência de dados em quilobytes por segundo ao longo do tempo. A taxa média de transferência calculada foi de 726 kB/s. Esse resultado indica que, quando a qualidade do sinal é satisfatória, o módulo 4G é capaz de realizar o *upload* de todos os dados coletados e processados pelo sistema para um experimento com o máximo de variáveis possível, visto que a taxa média de transferência é mais de 2 vezes maior que o valor da velocidade média de aquisição, processamento e *upload* de dados.

A capacidade de processamento e a memória RAM do sistema também se mostram suficientes para realizar todo o processamento de dados proveniente da ECU e demais módulos do hardware. Testes utilizando a capacidade máxima do barramento CAN indicam que o sistema é capaz de receber, processar e enviar todas as informações necessárias para o servidor sem indicar acúmulo de dados não processados.

Por meio de testes realizados para verificar o desempenho do hardware, a utilização dos núcleos de processamento (CPU) da FZ3 foi monitorada e é apresentada na Figura 5.4. Conforme é possível observar, a utilização da CPU se mantém predominantemente entre 5% e 10%, apresentando picos recorrentes de aproximadamente 28% de uso da capacidade máxima de processamento do sistema. Considerando que

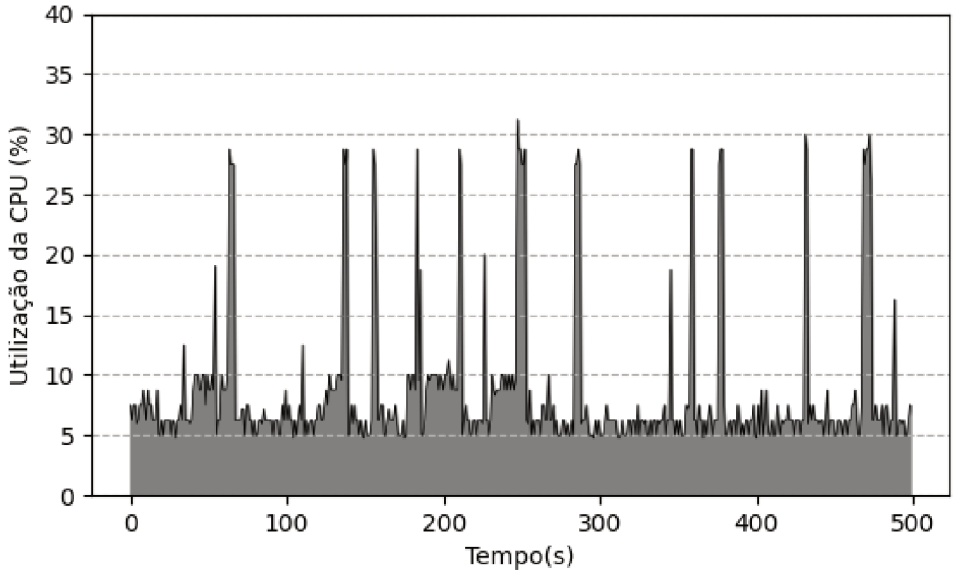
Figura 5.3 – Velocidade média de transferência de dados ao longo do tempo por meio do modem 4G.



Fonte: Elaborada pelo autor

o SoC possui quatro núcleos com frequência máxima de *clock* de 1,5 GHz, pode-se perceber uma considerável ociosidade da CPU.

Figura 5.4 – Utilização da CPU durante a execução do *firmware* do IASE ao longo do tempo.

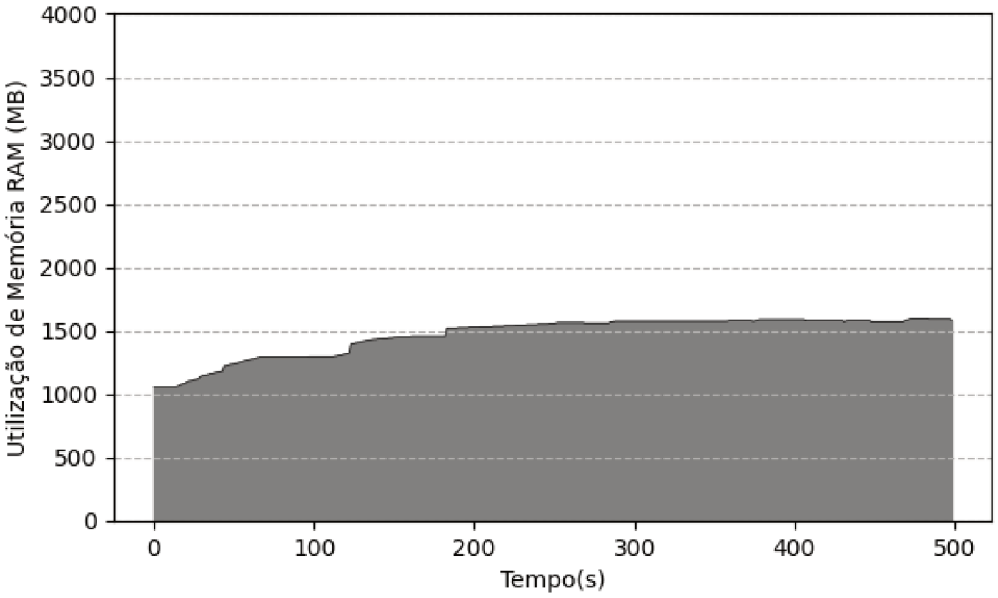


Fonte: Elaborada pelo autor

Paralelamente ao monitoramento de uso da CPU, o consumo de memória RAM do sistema também foi medido. A Figura 5.5 apresenta o gráfico referente às medições

de consumo de memória em *gigabytes* realizadas ao longo do tempo. O crescente aumento no consumo de memória no início do gráfico se deve ao acúmulo de dados provenientes da ECU nos *buffers* do *firmware*, antes do início do envio de dados para o servidor. Na medida que os pacotes de dados passam a ser enviados, o consumo de memória se estabiliza próximo a 1,6 GB, que equivale a aproximadamente 40% da memória RAM disponível.

Figura 5.5 – Utilização da memória RAM durante a execução do *firmware* do IASE ao longo do tempo.



Fonte: Elaborada pelo autor

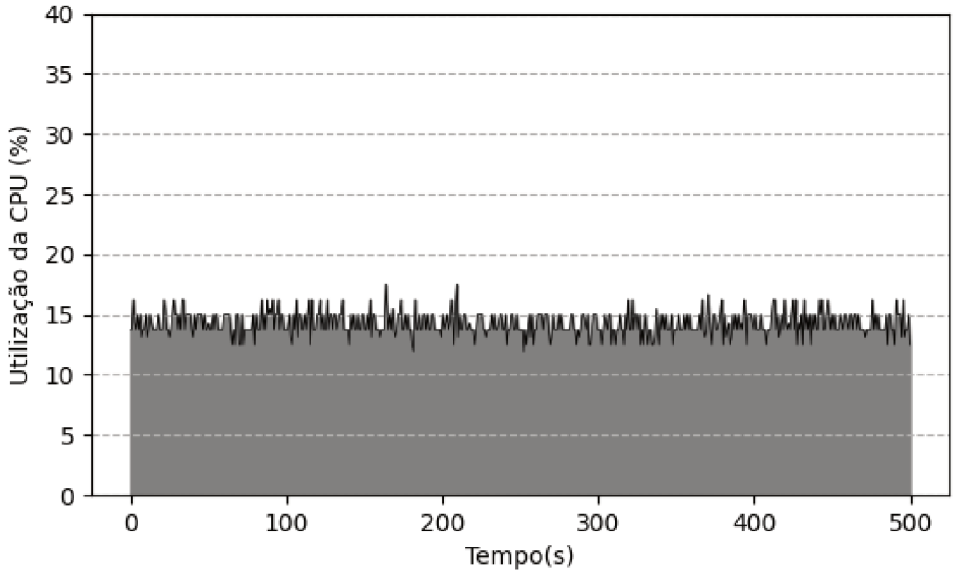
5.1.2 Desempenho do Hardware para Detecção de Knock Noise

Durante a execução do sistema de detecção de *knock noise* as informações referentes à utilização de CPU e memória RAM do hardware foram capturadas. Nessa seção são apresentados os resultados obtidos com essas medições.

Com relação à utilização da CPU, a Figura 5.6 apresenta a capacidade de processamento necessária para a execução do algoritmo de detecção do ruído de detonação ao longo do tempo. A média de utilização é de aproximadamente 14,24% e a carga máxima é de 17,5%.

A utilização da memória RAM apresentou valores muito próximos de zero, essencialmente devido à característica do programa de processamento de dados, que realiza o processamento de cada dado de entrada individualmente e armazena apenas o resultado final da análise diretamente na memória flash do hardware.

Figura 5.6 – Utilização da CPU durante a execução do programa de detecção de *knock noise* ao longo do tempo.



Fonte: Elaborada pelo autor

5.1.3 Desempenho Geral do Hardware

Com o objetivo de avaliar o desempenho do hardware durante a execução de ambos os sistemas de aquisição de dados simultaneamente e validar sua capacidade de captação e processamento de dados, testes foram realizados com a capacidade máxima de aquisição de dados dos barramentos de comunicação. Nesse sentido, o experimento configurado para a aquisição de dados via CAN foi o mesmo apresentado na Seção 5.1.1, Tabela 5.1.

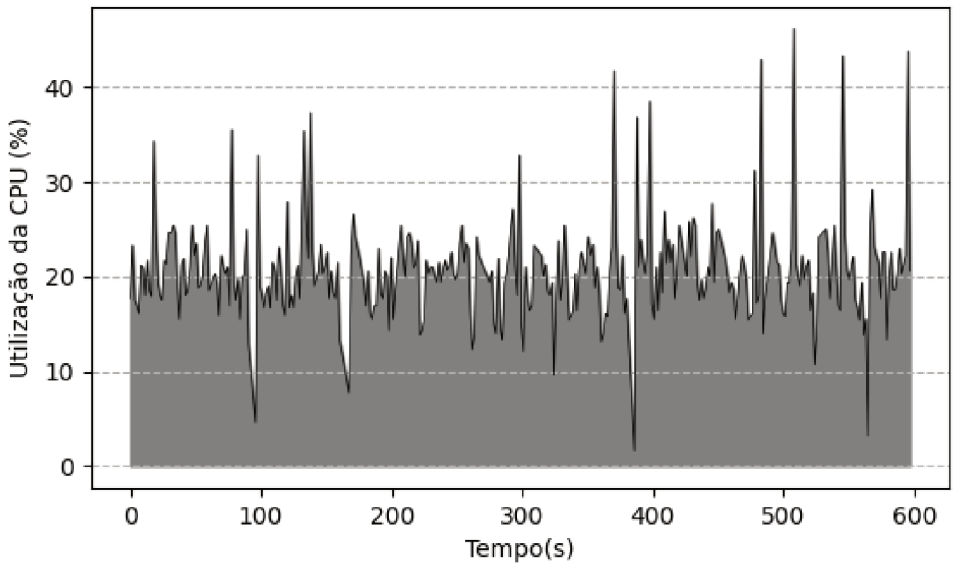
A Figura 5.7 ilustra a utilização da CPU durante a execução de ambos os sistemas de aquisição e processamento de dados, considerando a carga do sistema operacional. O valor médio do consumo da CPU ao longo do período monitorado é de 20,4% e o pico máximo de utilização é de 46,2%.

De forma similar, a utilização de memória RAM, ilustrada na Figura 5.8, indica o valor total do consumo da memória. A média de utilização total de memória é de 1723 MB enquanto o valor máximo durante o período monitorado é de 1758 MB.

5.1.4 Discussão de Resultados

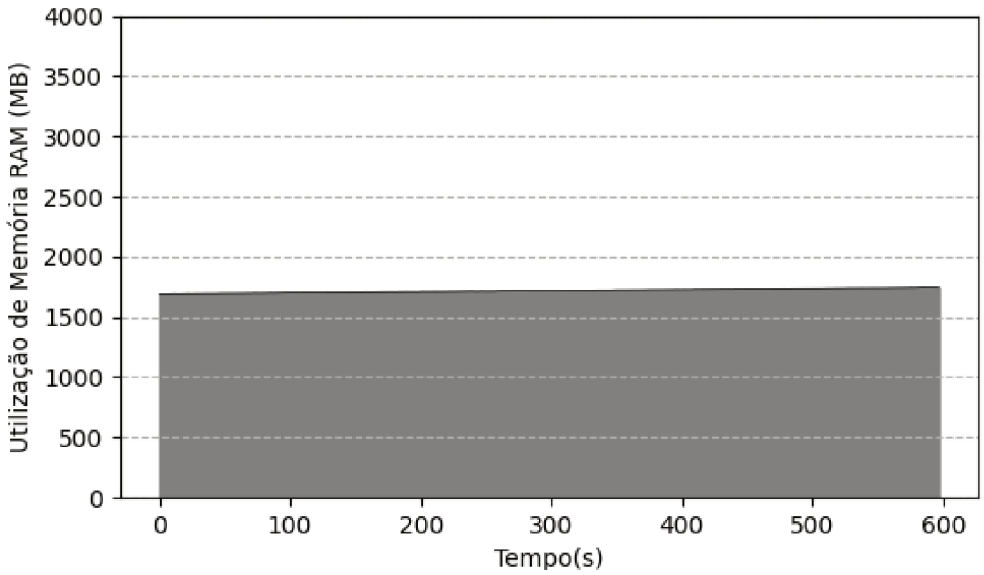
Considerando os resultados apresentados, o hardware demonstrou um desempenho adequado durante os testes de desempenho, sem proporcionar limitações para a operacionalidade do IASE ou do sistema de detecção de *knock noise*. Nesse sentido, é possível observar que o hardware desenvolvido é capaz de garantir a execução adequada de ambos os sistemas.

Figura 5.7 – Utilização da CPU durante a execução do IASE e do sistema de detecção de *knock noise* ao longo do tempo.



Fonte: Elaborada pelo autor

Figura 5.8 – Utilização da memória RAM durante a execução do IASE e do sistema de detecção de *knock noise* ao longo do tempo.



Fonte: Elaborada pelo autor

Para o IASE, o processo de captação de dados realizado através do barramento CAN atinge a taxa máxima de aquisição de dados permitida pelo protocolo definido pela ECU de testes. Em termos de processamento, o hardware demonstra recursos suficientes para garantir a execução do *firmware* sem atrasos ou acúmulo de dados

na memória. Por sua vez, o envio de dados ao servidor é realizado a uma taxa duas vezes maior que a necessária para o envio dos dados captados e processados. Em resumo, o hardware desenvolvido é capaz de atender os requisitos do IASE de forma satisfatória, com sobras consideráveis em termos de processamento, armazenamento e *upload* de dados ao servidor.

Os resultados obtidos durante a execução do sistema de detecção de *knock noise* indicam que o hardware possibilita a implementação de aplicações que demandem diferentes barramentos de aquisição de dados e maior capacidade de processamento. Adicionalmente, o hardware é capaz de executar ambos os sistemas simultaneamente sem apresentar sobrecarga em termos de processamento ou memória RAM.

Considerando essas características, é possível afirmar que o hardware desenvolvido habilita o desenvolvimento de diversas aplicações relacionadas à aquisição, ao processamento e à análise e armazenamento de dados provenientes de sensores veiculares, seja através da ECU ou diretamente de sensores específicos. Além disso, o cruzamento dos dados capturados pode permitir a aplicação de algoritmos de fusão de dados e IA, assim como o envio das informações resultantes dessas aplicações pode ser realizado ao servidor de IoT para armazenamento ou para uma segunda etapa de processamento e geração de relatórios.

Em geral, o hardware pode ser utilizado para uma extensa gama de aplicações, permitindo o desenvolvimento de análises específicas dos sistemas veiculares durante a etapa de testes e validação dos veículos. Por outro lado, o hardware pode ser otimizado para a aquisição de dados da ECU veicular, reduzindo seu custo de desenvolvimento e produção.

5.2 DETECÇÃO DE KNOCK NOISE

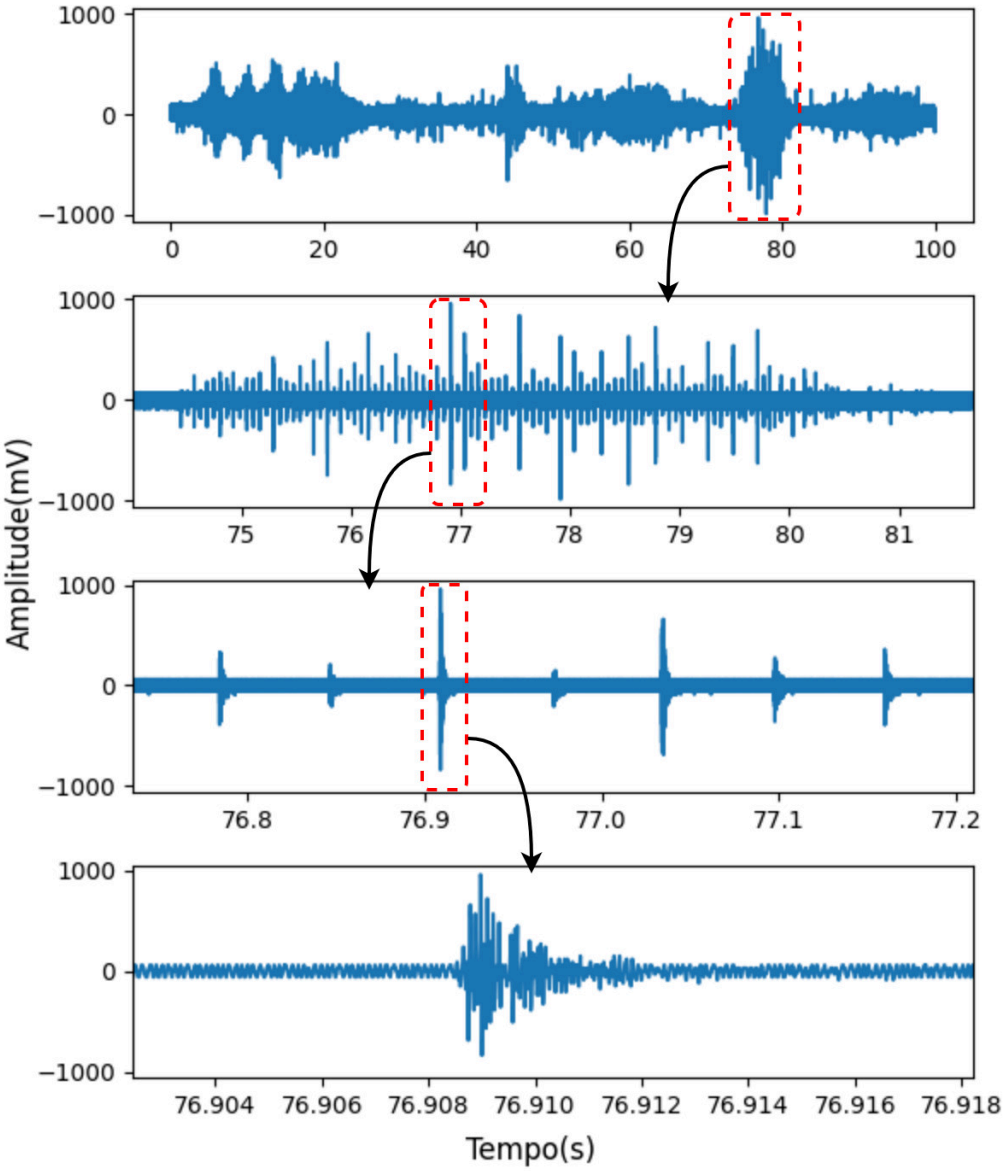
Nesta seção, são apresentados os testes realizados com o sistema de detecção de *knock noise*, demonstrando os dados originais captados do sensor piezoelétrico e os resultados de cada etapa do processo de identificação de ocorrências do ruído de detonação. Para mapear o desempenho do sistema, os resultados são comparados aos dados provenientes da ECU do veículo. Por fim, uma breve discussão sobre os resultados é apresentada.

5.2.1 Testes de Detecção de Knock Noise

A Figura 5.9 ilustra o sinal capturado pelo sensor piezoelétrico, utilizando uma frequência de amostragem de 115 kHz pelo hardware de detecção de *knock noise*. A imagem demonstra uma sequência de aproximações que descrevem o comportamento do sinal ao longo do tempo. É possível observar a presença de picos de tensão gerados

pelo sensor piezoelétrico, os quais podem ou não indicar a ocorrência do ruído de detonação, dependendo da frequência de oscilação do sinal.

Figura 5.9 – Sinal do sensor piezoelétrico do veículo com sucessivas aproximações temporais.



Fonte: Elaborada pelo autor

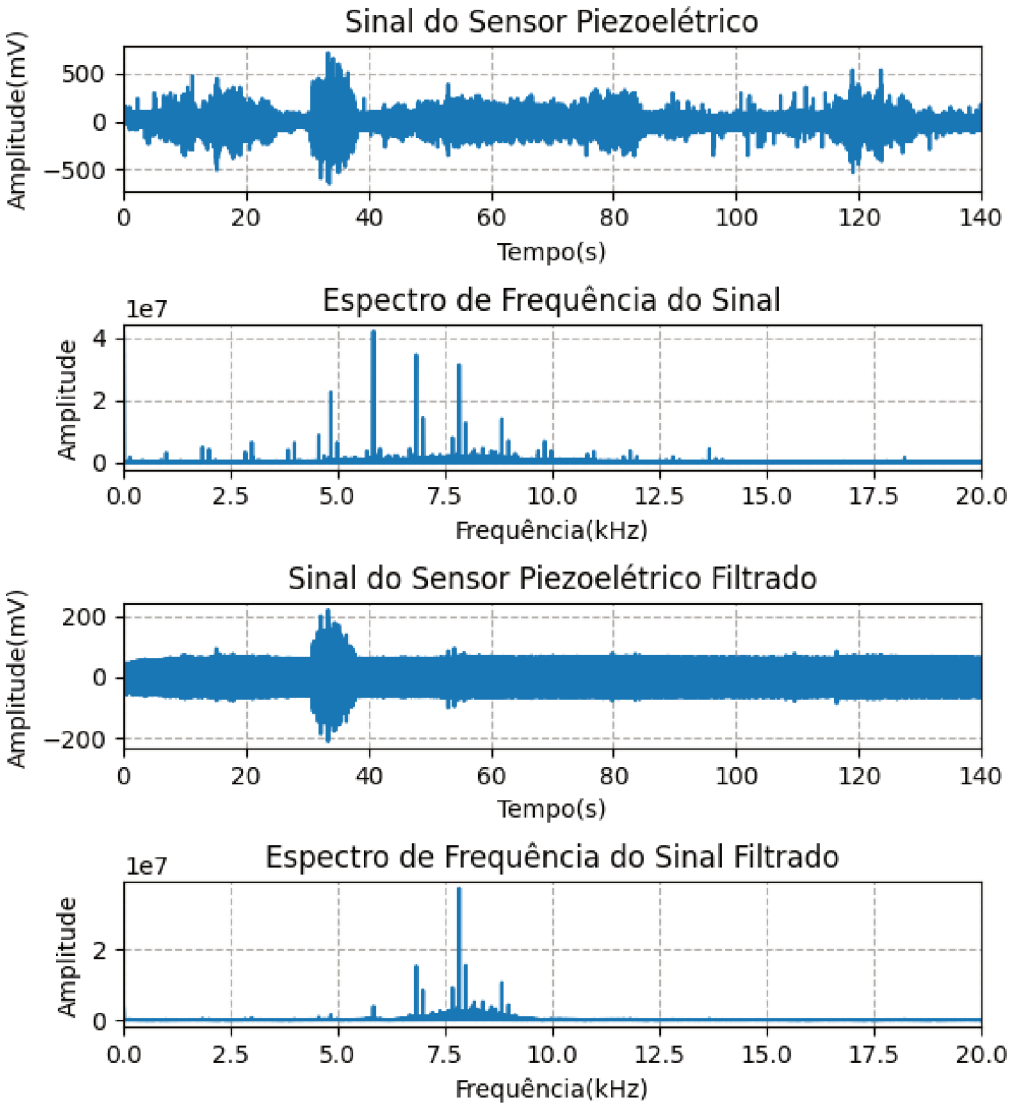
Apesar de o sinal capturado apresentar picos de tensão em diversos instantes, a maioria desses picos está relacionada a ruídos mecânicos de movimentação e vibração da estrutura do veículo. Além disso, ruídos característicos do funcionamento normal do *powertrain* também são captados e indicados pelo sensor.

Considerando a teoria exposta na Seção 2.4, que discute as propriedades do sinal de *knock noise* captado pelo sensor piezoelétrico, sabe-se que as frequências dos modos de vibração do ruído devem estar dentro da faixa de 6 a 25 kHz. Além

disso, com base nas informações apresentadas na Seção 4.4, determinou-se que a frequência fundamental do ruído de detonação para o motor do veículo de testes utilizado é de aproximadamente 7.973 kHz.

Com base nessas considerações, a Figura 5.10 apresenta o sinal original registrado pelo sensor piezoelétrico durante um período de 140 segundos de teste, juntamente com seu respectivo espectro de frequência. A análise do espectro revela a presença de vários picos de amplitude, que indicam frequências predominantes no sinal. Notavelmente, um desses picos ocorre muito próximo à frequência fundamental do ruído de detonação, sugerindo que possivelmente representa as ocorrências desse fenômeno.

Figura 5.10 – Sinal do sensor piezoelétrico original e filtrado com seus respectivos espectros de frequência.



Fonte: Elaborada pelo autor

Além disso, a Figura 5.10 exibe o sinal resultante da aplicação de um filtro FIR sobre o sinal original do sensor piezoelétrico, conforme os parâmetros estabelecidos na Seção 4.4. O sinal filtrado demonstra uma redução significativa na quantidade de picos de tensão. Os picos restantes, teoricamente, devem estar diretamente relacionados à ocorrência do ruído de detonação. Para ilustrar o efeito do filtro sobre o espectro de frequência original do sinal, o último gráfico na Figura 5.10 apresenta o espectro de frequência do sinal resultante após a aplicação do filtro.

Tendo como base um sinal filtrado para a frequência fundamental do ruído de detonação, a Figura 5.11 ilustra o processo de detecção das ocorrências de *knock noise*. A média móvel, destacada em vermelho, serve como parâmetro para a detecção de desvios do sinal. Picos de tensão que apresentam desvios maiores que 25% acima da média móvel são contabilizados como ocorrências do ruído de detonação. No gráfico, esses desvios são destacados em verde. As sucessivas aproximações permitem visualizar de forma mais adequada cada momento de detecção do ruído.

O último gráfico da Figura 5.11 indica a contagem cumulativa de cada uma das ocorrências detectadas ao longo do tempo. Nesse conjunto de dados, 76 ocorrências foram identificadas, sendo que 73 delas ocorreram sequencialmente em um curto período de tempo e são expostas no terceiro gráfico da Figura 5.11.

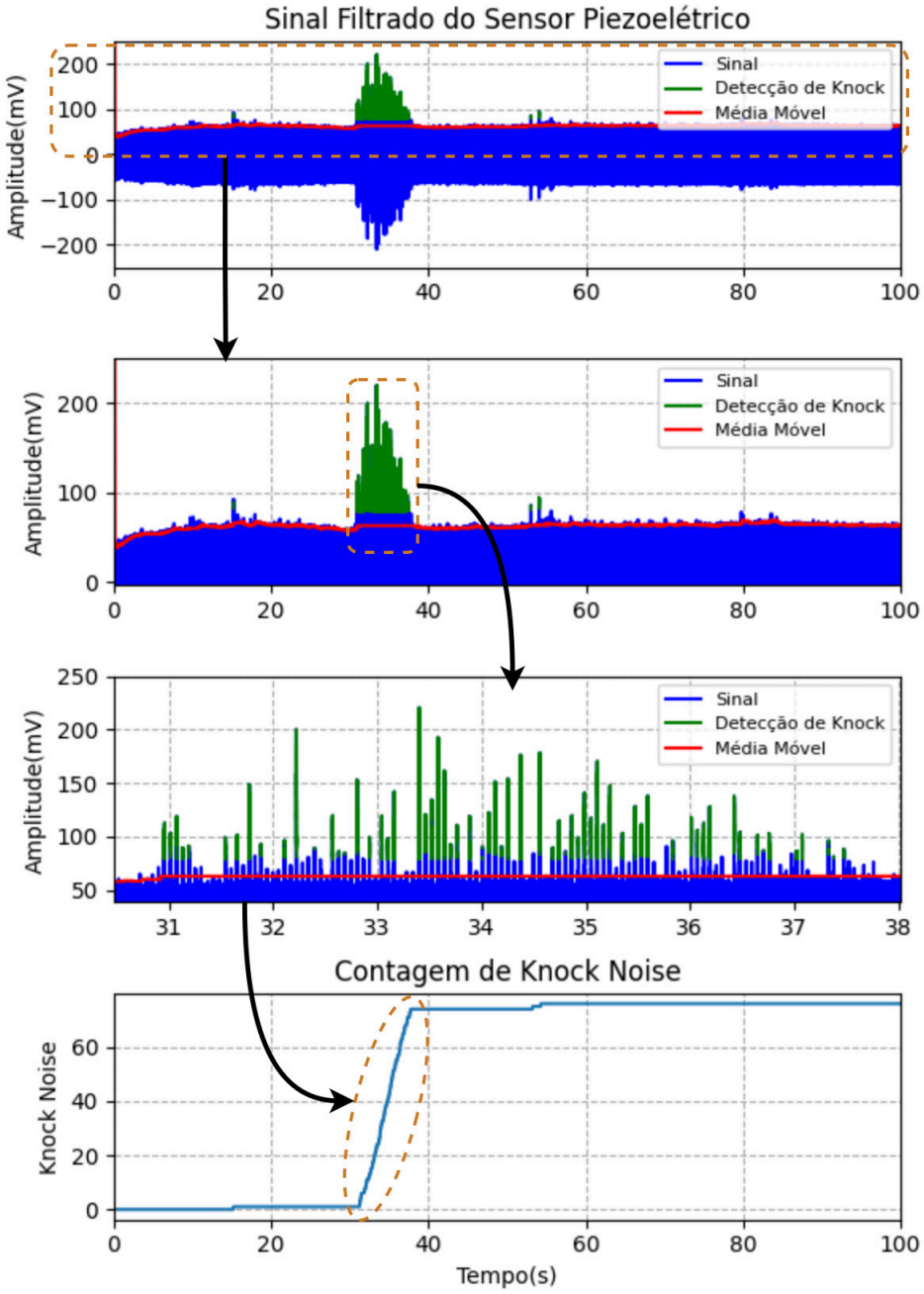
A fim de mapear a assertividade do sistema e avaliar o comportamento do sinal capturado e tratado, a Figura 5.12 apresenta, para um período de testes de 100 segundos, o sinal original de tensão capturado do sensor piezoelétrico, o sinal filtrado para a frequência fundamental do ruído, a contagem de *knock noise* realizada pelo sistema e a contagem de *knock noise* realizada pela ECU do veículo. A Tabela 5.2 apresenta a contagem de *knock noise* em cada um dos momentos destacados no gráfico. É possível notar que, durante esse período de testes, o sistema de detecção desenvolvido identificou mais ocorrências do ruído que a ECU, o que pode indicar que o sistema possui maior sensibilidade a ruídos que não são provenientes da ocorrência real de *knock noise* no motor do veículo. A porcentagem total de erros e acertos apresentada na Tabela 5.2 é determinada considerando médias ponderadas de cada período de ocorrência do ruído nos conjuntos de dados coletados.

Tabela 5.2 – Quantidade de ocorrências de *knock noise* para cada período da Figura 5.12.

Período	**Knock Noise (Piezoelétrico)**	**Knock Noise (ECU)**	**Erros (%)**	**Acertos (%)**
1	1	2	50%	50%
2	73	63	16%	84%
3	2	0	100%	0%
Total	76	65	17%	83%

Fonte: Elaborada pelo autor.

Figura 5.11 – Resultado do processo de detecção de *knock noise* a partir do sinal filtrado do sensor piezoelétrico.

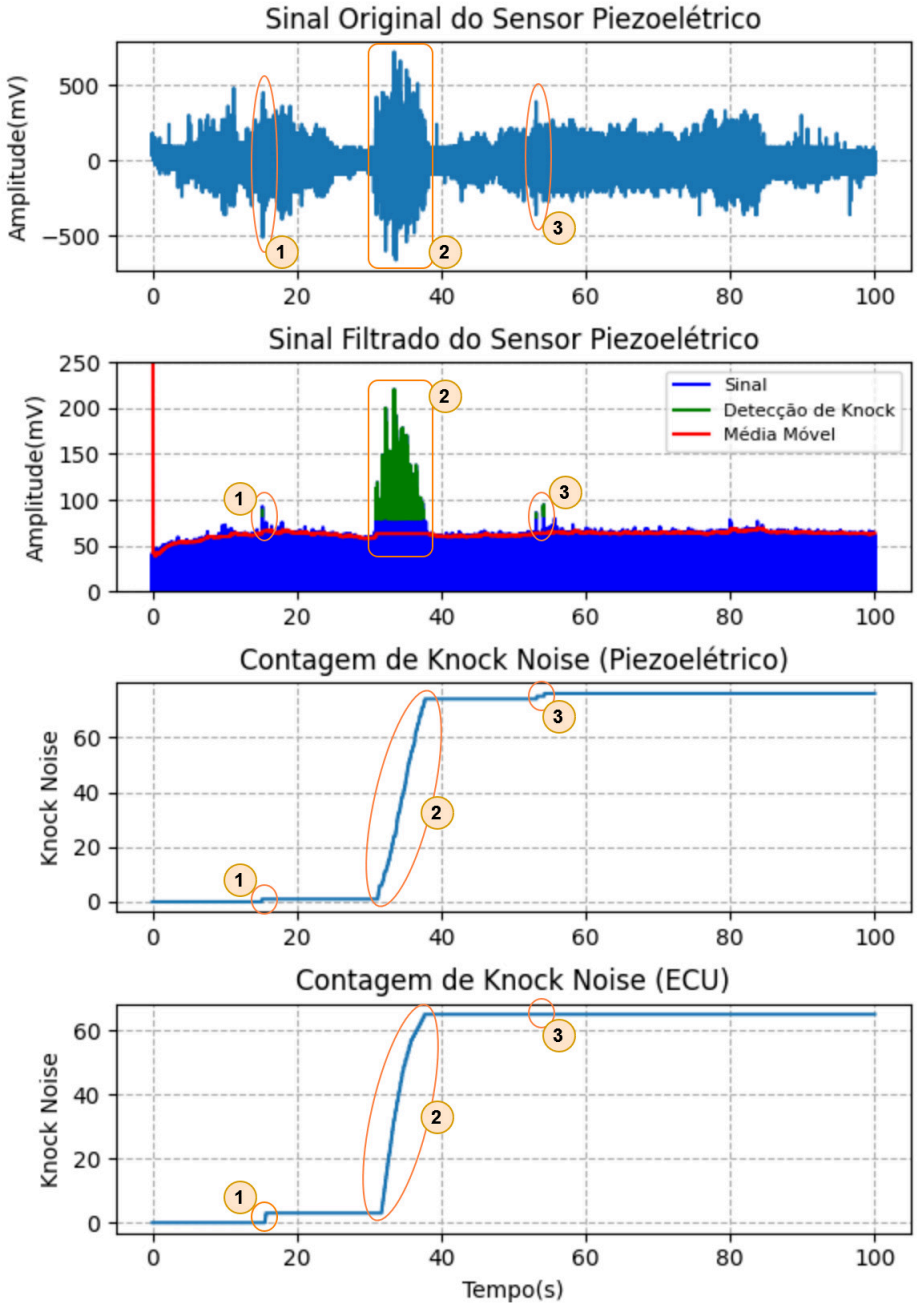


Fonte: Elaborada pelo autor

Um segundo conjunto de dados comparando as ocorrências de *knock noise* detectadas pelo sistema desenvolvido e pela ECU é ilustrado na Figura 5.13. Dessa vez, é possível notar que a quantidade de ocorrências identificadas pela ECU é maior que a quantidade de ocorrências detectadas através dos dados do sensor piezoelétrico. A Tabela 5.3 resume as ocorrências do ruído para cada período destacado nos gráficos.

Com base nos diversos testes realizados para avaliar o desempenho do sistema

Figura 5.12 – Detecção de *knock noise* a partir do sinal do sensor piezoelétrico e detecção realizada pela ECU.

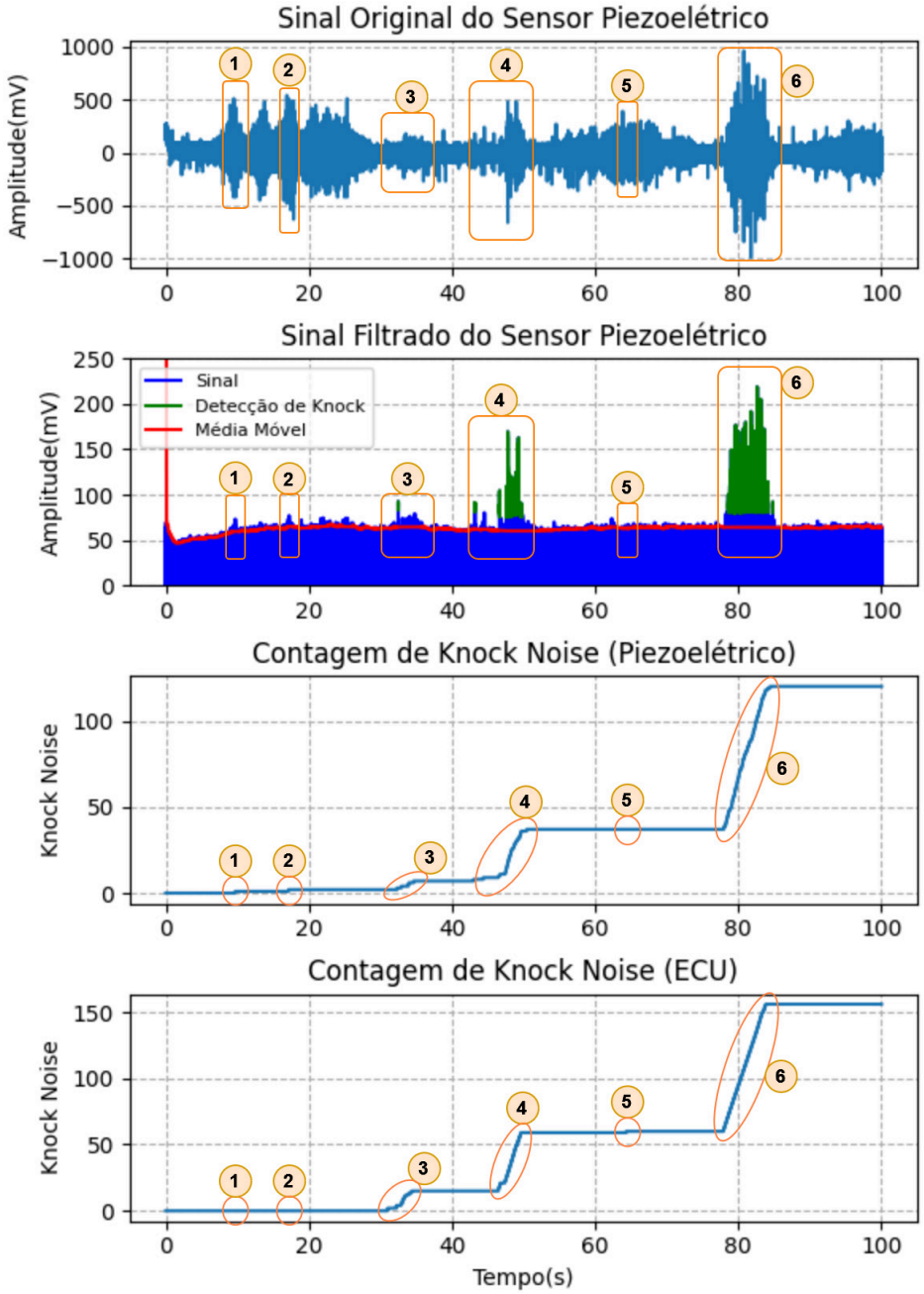


Fonte: Elaborada pelo autor

de detecção de *knock noise* e considerando médias ponderadas de cada período de ocorrência do ruído nos conjuntos de dados coletados, o valor médio de acertos obtido foi de aproximadamente 67,5%.

O tempo de resposta do sistema pode ser estimado em torno de 8,7 μs, estando diretamente relacionado à frequência de aquisição de dados do sensor e considerando que o tempo de processamento de cada elemento do sinal é desprezível em relação

Figura 5.13 – Detecção de *knock noise* a partir do sinal do sensor piezoelétrico e detecção realizada pela ECU.



Fonte: Elaborada pelo autor

ao tempo de aquisição.

5.2.2 Discussão de Resultados

A detecção de *knock noise* realizada pelas ECUs veiculares é crucial para garantir a eficiência e a durabilidade do motor. Esses métodos de detecção são bem estabelecidos e amplamente utilizados na indústria automotiva há décadas. Embora

Tabela 5.3 – Quantidade de ocorrências de *knock noise* para cada período da Figura 5.13.

Período	Knock Noise (Piezoelétrico)	Knock Noise (ECU)	Erros (%)	Acertos (%)
1	1	0	100%	0%
2	1	0	100%	0%
3	5	15	67%	33%
4	30	44	32%	68%
5	0	1	100%	0%
6	83	96	14%	86%
Total	120	156	24,4%	75,6%

Fonte: Elaborada pelo autor.

nem todos os parâmetros e sinais usados pelo algoritmo de detecção da ECU sejam conhecidos, sabe-se que além do sinal do sensor piezoelétrico, outros fatores são considerados, como mapas de detonação, temperatura do ar de admissão, temperatura do motor, pressão do coletor de admissão, rotação do motor, entre outros. Assim sendo, o sistema desenvolvido não tem a intenção de substituir os métodos de detecção atuais, mas sim permitir a aplicação de novas metodologias e algoritmos que possam proporcionar maior precisão e menor tempo de resposta na detecção do ruído de detonação.

Durante o desenvolvimento e implementação do sistema, outros fatores capazes de prejudicar seu desempenho e assertividade foram identificados e devem ser considerados para uma avaliação adequada do equipamento. Primeiramente, os dados utilizados em todo o processo de desenvolvimento foram coletados de um protótipo veicular em movimento, ou seja, não foi possível produzir um sinal de *knock noise* isolado de vibrações mecânicas para o ajuste e calibração dos parâmetros do sistema. Ao forçar a ocorrência do ruído de detonação durante a movimentação do veículo, uma quantidade ainda maior de ruídos mecânicos é produzida, podendo gerar sinais falsos do ruído. Além disso, as informações provenientes da ECU são capturadas a cada 4 ms, impedindo uma comparação mais precisa do momento de ocorrência do ruído e impossibilitando um ajuste mais adequado dos parâmetros de detecção.

O algoritmo utilizado para o processamento dos dados do sinal também apresenta limitações com relação ao seu desempenho. Conforme descrito na Seção 2.5, algoritmos mais avançados podem ser aplicados para aumentar a precisão do sistema. Métodos de análise do sinal utilizando IA e informações adicionais do veículo podem possibilitar resultados tão adequados quanto os indicados pela ECU. Informações sobre o ciclo dos cilindros podem permitir o isolamento de períodos nos quais o *knock noise* não ocorre, evitando indicações falsas em grande parte do período de aquisição dos dados.

Considerando a proposta do presente trabalho, o sistema desenvolvido teve como principal objetivo validar a capacidade do hardware para capturar e processar as informações veiculares. Nesse sentido, a implementação de algoritmos mais avançados e a modificação das condições de testagem foi evitada, mantendo o escopo do trabalho e evitando a adição de maior complexidade ao sistema.

Apesar das limitações apresentadas, o sistema de detecção de *knock noise* foi capaz de atingir um desempenho de 75,6%, demonstrando a capacidade de identificar as ocorrências do ruído. O tempo de resposta do sistema, de 8,7μs, permite ao hardware identificar as anomalias com uma velocidade 400 vezes superior à velocidade de aquisição de dados via ECU, com períodos de aquisição de 4 ms.

De maneira geral, o sistema desenvolvido é capaz de comprovar a capacidade do hardware para realizar a aquisição e o processamento de dados de diversos sensores do veículo através de suas interfaces de comunicação. Adicionalmente, é possível afirmar que o hardware habilita a computação de borda para diversas aplicações relacionadas à aquisição e ao processamento dos dados veiculares, incluindo algoritmos de IA e de fusão de dados.

5.3 RESULTADOS COMPLEMENTARES

Além dos resultados obtidos e demonstrados nas seções anteriores, o hardware desenvolvido tem possibilitado a pesquisa e desenvolvimento de novos trabalhos relacionados a aquisição, processamento, análise e armazenamento de dados veiculares. Essa seção apresenta brevemente alguns desses trabalhos e seus respectivos resultados.

O *firmware* embarcado desenvolvido para o IASE foi o primeiro trabalho habilitado pelo hardware, possibilitando a aquisição, processamento e *upload* dos dados veiculares ao servidor. Nesse sentido, os trabalhos de hardware e software são complementares e permitiram a obtenção de resultados específicos nas respectivas áreas de pesquisa. Os trabalhos apresentados por García et al. (2022) e Igarashi (2022) descrevem as etapas de desenvolvimento do *firmware*, os testes realizados e os resultados alcançados. Dentre os principais resultados estão a indicação da leitura adequada dos dados provenientes da ECU e o respectivo processamento e *upload* padronizado dos dados baseado no paradigma de IoT.

Utilizando os dados veiculares capturados e transmitidos pelo hardware e *firmware* do IASE, os trabalhos de Silva, Gracioli e Araujo (2022) e Silva (2022) buscaram identificar uma relação entre a ocorrência do fenômeno de pré-ignição e as diversas variáveis que podem ser monitoradas através da ECU do veículo. Para isso, diferentes técnicas de seleção de atributos foram aplicadas, visando obter conjuntos de variáveis que tenham relação direta ou indireta com a ocorrência de *knock noise*. Essas técnicas consistem na implementação de algoritmos de aprendizado de máquina que

tem por objetivo construir o menor subconjunto de dados possível que represente todos os atributos vitais dos dados de entrada. Os resultados desses trabalhos demonstram que a seleção de características pode reduzir o número de atributos selecionados em um classificador de falhas em 55% (de 9 para 5), com um aumento na precisão de detecção de 2%.

Por sua vez, os trabalhos apresentados por Francis et al. (2022) e Pierozan (2022) empregam técnicas de inteligência artificial para detectar o fenômeno de pré-ignição em motores a combustão. A pesquisa foi realizada utilizando cinco diferentes técnicas de aprendizado de máquina: classificador de extração de características; AutoEncoder Denso e Convolucional; SVM; e floresta isolada. Um conjunto de 19 variáveis provenientes da ECU do veículo foram monitoradas e utilizadas para as análises e o melhor resultado obtido pelos algoritmos foi através do classificador de extração de características, que alcançou uma precisão de 81%.

Outro trabalho, habilitado pelo IASE, que utiliza os dados provenientes da ECU veicular como entrada para algoritmos de aprendizado de máquina, é apresentado por Canal (2023). Nessa pesquisa são utilizados diferentes modelos de aprendizado de máquina para a construção de três análises inteligentes: detecção da falha de ignição, classificação do perfil de condução do motorista e predição do consumo de combustível. Em comparação com trabalhos relacionados, os algoritmos aplicados apresentaram resultados similares ou superiores, com a diferença de serem obtidos por meio de algoritmos menos custosos computacionalmente, com dados exclusivos da ECU e em tempo real. Uma análise específica da aplicação do aprendizado de máquina para determinação do perfil de condução do motorista pode ser verificada em Canal, Riffel e Gracioli (2023).

5.4 COMPARAÇÃO COM SISTEMAS COMERCIAIS

Com o objetivo de avaliar as diferenças e similaridades entre os equipamentos comerciais de aquisição de dados de protótipos veiculares e o hardware desenvolvido, esta seção se dedica a comparar suas características, desempenho e custos. Considerando que o foco principal do hardware é habilitar a aquisição de dados da ECU veicular, o comparativo é realizado com sistemas de aquisição de dados comerciais que utilizam os mesmos protocolos e interfaces de comunicação. Nesse sentido, são considerados os produtos e soluções dos fabricantes Vector, ATI e ETAS, que utilizam o barramento CAN e protocolos CCP ou XCP para a aquisição dos dados veiculares.

Para facilitar a descrição, a comparação de características relacionadas ao sistema de aquisição de dados da ECU utiliza como referência o IASE, considerando que todas as funcionalidades desse sistema são habilitadas pelo hardware desenvolvido.

5.4.1 Características

Em relação aos métodos de aquisição, tanto o IASE quanto os sistemas comerciais utilizam o mesmo para se comunicar com a ECU, definir as variáveis do experimento e receber os dados solicitados. Além disso, os protocolos e a velocidade de comunicação são os mesmos, portanto, os equipamentos não diferem nesse recurso específico.

No que diz respeito à arquitetura do sistema, os equipamentos comerciais são dependentes de um computador pessoal para a aquisição, processamento e armazenamento de dados, configuração dos experimentos, visualização dos dados e interface com o usuário, possuindo uma estrutura centralizada que utiliza apenas um dispositivo de interface para permitir a comunicação entre a ECU e o PC. O IASE, por outro lado, foi projetado para interagir com outros dispositivos e plataformas. As etapas de aquisição e processamento são feitas pelo hardware desenvolvido, os dados são armazenados no servidor de IoT, a visualização dos dados pode ser realizada em tempo real por aplicativos web ou softwares dedicados, e algoritmos de IA podem ser executados na nuvem, gerando alertas caso alguma anomalia seja encontrada.

A capacidade do sistema de enviar, processar e disponibilizar os dados coletados em tempo real durante a rodagem dos veículos é um grande diferencial do IASE. Essa característica permite que a equipe de engenheiros e especialistas do fabricante seja rapidamente informada caso anomalias sejam identificadas. Em situações como essa, o problema pode ser analisado e a rodagem do veículo pode ser interrompida, reiniciada ou uma nova configuração de experimento pode ser feita para fornecer mais informações relacionadas à condição específica. Esse recurso permite que o fabricante melhore a eficiência da fase de testes dos veículos.

Em contraste, a maior parte das soluções comerciais só permite o *upload* dos dados adquiridos quando o experimento é concluído. Nesse caso, se um experimento tiver erros, o procedimento do motorista estiver incorreto, o veículo apresentar alguma falha crítica ou algumas variáveis estiverem fora dos limites especificados e o experimento precisar ser reiniciado, reconfigurado ou interrompido, isso só poderá ser verificado depois que o motorista concluir o teste de rodagem e o arquivo do experimento for carregado no servidor e processado.

Outra funcionalidade do IASE é a interface de comunicação *Bluetooth*, que possibilita a utilização de um *smartphone* para o controle e visualização do status do sistema. Adicionalmente, o *display* de LEDs e o *buzzer* garantem informações básicas referentes à execução do experimento.

Além do processo de aquisição de dados, tanto os sistemas comerciais quanto o IASE são capazes de realizar a calibração de parâmetros da ECU. Esse recurso permite que o sistema seja usado em fases mais precoces do desenvolvimento do veículo através de ajustes nos parâmetros da ECU, a fim de garantir o adequado

funcionamento do veículo e melhorar sua eficiência.

Um dos grandes diferenciais do hardware desenvolvido é a possibilidade de implementação de sistemas de aquisição e processamento de dados diretamente dos sensores do veículo ou de sensores adicionais instalados para o monitoramento de variáveis específicas. Essa capacidade é comprovada pela implementação do sistema de detecção de *knock noise*, que permite a captação de dados externos através de barramentos de comunicação específicos.

Por fim, a disponibilidade computacional do hardware permite que ele seja utilizado para realizar computação de borda para diversas aplicações. Nesse sentido, os dados coletados da ECU ou diretamente de sensores do veículo podem ser processados ou pré-processados localmente pelo hardware, acelerando os processos de análise e resposta do sistema. Os principais equipamentos comerciais atualmente disponíveis não apresentam essa capacidade de processamento e integração.

5.4.2 Desempenho

A velocidade de processamento e aquisição do hardware desenvolvido são limitadas pelo método de comunicação com a ECU. O barramento CAN, usado para implementar essa comunicação, permite taxas de transmissão de até 1 Mbit/s (ISO 11898-2). Como consequência dessa limitação, o número de variáveis que podem ser adquiridas da ECU também é restrito. Portanto, a quantidade de variáveis que podem ser configuradas em um experimento varia de acordo com seu tamanho e frequência de aquisição. Assim, em termos de velocidade de aquisição, tanto os sistemas comerciais como o hardware desenvolvido apresentam o mesmo desempenho, pois ambos se comunicam com a ECU usando os mesmos métodos e têm as mesmas limitações de velocidade de comunicação.

Considerando que tanto o IASE quanto os sistemas comerciais recebem dados da ECU por meio de protocolos digitais, a possível divergência na precisão só pode ser atribuída a possíveis divergências nos métodos de conversão ou erros na transmissão de dados. Com o objetivo de verificar a assertividade da aquisição e processamento de dados, o valor resultante de diversas variáveis do veículo foi comparado. Nesse critério, não foram encontradas diferenças entre os dois sistemas.

5.4.3 Custos

Os custos relacionados à aquisição de materiais, à fabricação e montagem da PCB e à confecção da caixa de acrílico para proteção do hardware são listados na Tabela 5.4. Devido à dificuldade de obter informações relacionadas ao custo de equipamentos comerciais, o comparativo foi realizado apenas com o equipamento da empresa ETAS, mais especificamente o módulo USB ETAS ES581 CAN Bus (ETAS,

2011), juntamente com o software ETAS INCA (ETAS, 2020). Os valores relacionados à aquisição do hardware e da licença de uso do software são listados na Tabela 5.5.

Considerando apenas o hardware, a solução desenvolvida é cerca de 5,6 vezes mais barata do que o INCA, comprovando sua viabilidade. Incluindo a licença anual necessária para usar o software INCA, a diferença de custo aumenta para mais de 8 vezes.

Tabela 5.4 – Custos com materiais e mão de obra para a fabricação do hardware.

Item	Quantidade	Valor Unitário	Total
PCB (Fabricação)	1	R$ 30.00	R$ 30.00
PCB (Montagem)	1	R$ 70.00	R$ 70.00
FZ3 (Processador)	1	R$ 2,375.21	R$ 2,375.21
EC25 (Modem 4G LTE/GPS)	1	R$ 412.62	R$ 412.62
ESP32 (Módulo Bluetooth)	1	R$ 23.01	R$ 23.01
Resistores	39	R$ 1.06	R$ 41.26
Capacitores	25	R$ 2.38	R$ 59.51
Reguladores de Tensão	3	R$ 23.81	R$ 71.42
Conectores	15	R$ 13.23	R$ 198.38
LEDs	6	R$ 1.59	R$ 9.52
Transistors	19	R$ 2.65	R$ 50.26
Case	1	R$ 60.00	R$ 60.00
Outros	1	R$ 26.45	R$ 26.45
Custo Total			**R$ 3,427.63**

Fonte: Elaborada pelo autor.

Tabela 5.5 – Custos de aquisição do hardware e licença de uso do software de um dos sistemas de aquisição de dados da empresa ETAS.

Item	Valor Total
Conector ES581	R$ 5,000.00
Computador	R$ 14,500.00
Licença (anual)	R$ 8,000.00
Custo Total	**R$ 27,500.00**

Fonte: Elaborada pelo autor.

6 CONCLUSÃO

Com base nos objetivos apresentados na Seção 1.1, este trabalho descreve o desenvolvimento de um hardware de aquisição de dados para protótipos veiculares capaz de atender as demandas do processo de testes e validação de novos veículos realizado pela indústria automotiva. Nesse sentido, o hardware é capaz de atender a todas as funcionalidades do processo, permitir a implementação de novas tecnologias e apresentar baixo custo de produção em relação aos sistemas comerciais mais utilizados pelos fabricantes.

Os requisitos funcionais e não funcionais do hardware foram definidos a partir dos requisitos funcionais estabelecidos para o IASE e considerando a implementação de tecnologias de aquisição e processamento de dados adicionais, tendo em vista a possível ampliação de suas aplicações. Os testes de integração do *firmware* do IASE com o hardware desenvolvido demonstraram que os requisitos de hardware foram estabelecidos adequadamente, permitindo que todas as funcionalidades especificadas para o sistema sejam atendidas. Adicionalmente, os testes do sistema de detecção de *knock noise* indicaram a capacidade do hardware de habilitar a implementação de outras aplicações paralelamente ao IASE.

Os componentes selecionados para a elaboração do projeto apresentaram um desempenho adequado para o correto funcionamento do sistema. A placa de processamento (cartão FZ3) se mostrou capaz de gerenciar as diversas interfaces de comunicação do hardware e garantir o processamento dos dados coletados do veículo sem atrasos ou sobrecarga. O módulo ESP32 permitiu a comunicação do hardware via interface *Bluetooth* com dispositivos externos para o controle e monitoramento de execução do IASE. O modem EC25-AU apresentou capacidade suficiente para a transmissão dos dados coletados e para a captura das informações de GPS. Por fim, os demais componentes utilizados para o desenvolvimento do hardware proporcionaram a adequada interconexão e energização dos módulos.

A partir da execução dos testes de desempenho do hardware foi possível avaliar o funcionamento do processo de coleta, processamento e envio de dados ao servidor de IoT. Utilizando a capacidade máxima do barramento CAN, os resultados de medição do sistema comprovaram que o hardware é capaz de processar e atualizar todos os dados adquiridos para o servidor a uma taxa de 330 kB/s. Adicionalmente, o equipamento se demonstrou capaz de realizar a coleta de dados diretamente dos sensores do veículo e executar aplicações específicas como a detecção de *knock noise* em paralelo com o *firmware* de aquisição de dados da ECU.

A baixa utilização da CPU e da memória RAM do hardware desenvolvido e o considerável desempenho do sistema de detecção de *knock noise* demonstram a possibilidade de implementação de novas aplicações ao sistema baseadas em computação

de borda. Algoritmos de fusão de dados podem ser utilizados para gerar informações mais precisas sobre determinados fenômenos e ferramentas de IA podem ser aplicadas para identificar padrões e anomalias durante a execução dos testes de rodagem do veículo.

Em comparação com as soluções comerciais que utilizam os mesmos métodos para aquisição de dados das ECUs veiculares, o IASE apresenta resultados similares de desempenho, enquanto seu custo, comparado a um dos equipamentos da empresa ETAS, é até 8 vezes menor, indicando a viabilidade comercial do equipamento.

A capacidade do IASE de adquirir, processar e enviar os dados em tempo real mostrou ser uma diferença importante em comparação com as soluções comerciais que geralmente permitem a análise dos dados apenas após a conclusão de todo o processo de teste. Esse recurso pode ser um ativo importante se algo inesperado acontecer durante os testes em estradas, permitindo que os especialistas do fabricante analisem o problema e tomem medidas imediatas, se necessário, reiniciando, interrompendo ou reconfigurando o experimento.

6.1 TRABALHOS FUTUROS

O hardware desenvolvido habilita a implementação de uma gama muito grande de aplicações relacionadas à coleta, ao processamento e à análise de dados veiculares. Nesse sentido, sem realizar novas alterações de hardware, novos projetos podem ser realizados utilizando suas funcionalidades. Uma possibilidade é a implementação de algoritmos de IA e fusão de dados que podem auxiliar diversas análises nos sistemas de protótipos veiculares, incluindo um diagnóstico mais preciso com relação à detecção de *knock noise*.

Em relação ao hardware, diversas melhorias podem ser realizadas visando a adição de funcionalidades ou melhorias de desempenho. A seguir são listados alguns itens específicos:

- Substituição do modem 4G por um modem 5G para elevar a taxa de transmissão de dados para o servidor de IoT, acelerando o processo de computação em nuvem dos dados veiculares.
- Melhorias no circuito de condicionamento de tensão podem ser realizadas para evitar a alta dissipação de energia através dos reguladores de tensão lineares.
- O design da placa pode ser melhorado, visando disponibilizar maior conectividade com dispositivos externos.
- As funcionalidades da placa FZ3 e do módulo ESP32 podem ser melhor exploradas, possibilitando maior velocidade de aquisição de dados de sensores veiculares, seja via barramento de comunicação ou sinais analógicos.

- O hardware pode ser otimizado para a redução de custos de produção com base nos dados de processamento e armazenamento de dados relatados no presente trabalho.

Além dessas melhorias, equipamentos adicionais podem ser projetados para operar em conjunto com o hardware desenvolvido, amplificando suas funcionalidades. Conversores analógicos digitais podem ser utilizados para capturar dados em alta frequência de sinais analógicos de sensores veiculares. Outros barramentos de dados podem ser acessados através de conversores que permitam a adequação dos níveis de tensão necessários ao hardware. Sensores adicionais podem ser instalados nos veículos para monitorar variáveis específicas e enviar seus dados através do módulo 4G.

Por fim, a implementação de um sistema de captura de dados da ECU veicular via JTAG pode ser desenvolvido com base no hardware atual caso as informações necessárias referentes ao endereçamento das variáveis e design do circuito da unidade de processamento estejam disponíveis.

REFERÊNCIAS

AKIMOTO, K.; KOMATSU, H.; KURAUCHI, A. Development of pattern recognition knock detection system using short-time fourier transform. *IFAC Proceedings Volumes*, Elsevier, v. 46, n. 21, p. 366–371, 2013.

ALAM, M. S. U. *Securing vehicle Electronic Control Unit (ECU) communications and stored data*. Tese (Doutorado) — Queen's University (Canada), 2018.

AMD. *Vivado Design Suite User Guide*. v2023.1. [S.l.], 2023.

ANDERT, J. et al. Virtual shaft: Synchronized motion control for real time testing of automotive powertrains. *Control Engineering Practice*, Elsevier Ltd, v. 56, p. 101–110, 11 2016. ISSN 09670661.

ATI. *A8 High-Speed Serial ECU Interface*. 2022. Disponível em: <https://www.accuratetechnologies.com/PDFs/A8%20Serial%20Interface%20EN.pdf>. Acesso em: 11 de jul. de 2022.

ATI. *A9 High-Speed Serial ECU Interface*. 2022. Disponível em: <https://www.accuratetechnologies.com/PDFs/A9%20Serial%20Interface%20EN.pdf>. Acesso em: 11 de jul. de 2022.

ATI. *Beyond Automotive*: Measurement, calibration and diagnostics portfolio overview. 2022. Disponível em: <https://www.accuratetechnologies.com/PDFs/ATI%20MCD%20Overview%20191119_LowRes.pdf>. Acesso em: 11 de jul. de 2022.

ATI. *DLX Compact Datalogger*: Manual. [S.l.], 2023. Disponível em: <https://www.accuratetechnologies.com/PDFs/DLX%20Datalogger%20EN.pdf>.

AUTOPAPO. *Fogo nos Onix Plus é decorrente de quebra do motor*. 2019. Disponível em: <https://autopapo.uol.com.br/noticia/fogo-nos-onix-plus-motor/>.

AZZEH, A. R. Can control system for an electric vehicle. University of Canterbury. Electrical and Computer Engineering, 2007.

BEDRETCHUK, J. P. et al. Low-cost data acquisition system for automotive electronic control units. *Sensors*, v. 23, n. 4, 2023. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/23/4/2319>.

BI, F. et al. Knock detection based on the optimized variational mode decomposition. *Measurement*, Elsevier, v. 140, p. 1–13, 2019.

BI, F.; MA, T.; WANG, X. Development of a novel knock characteristic detection method for gasoline engines based on wavelet-denoising and emd decomposition. *Mechanical Systems and Signal Processing*, Elsevier, v. 117, p. 517–536, 2019.

BI, Z. et al. Bit-level automotive controller area network message reverse framework based on linear regression. *Sensors*, v. 22, n. 3, 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/3/981>.

CANAL, R. *Machine Learning Applied in Automotive ECUs for Real-time Engine Behavior Analisys*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2023.

CANAL, R.; RIFFEL, F. K.; GRACIOLI, G. Driving profile analysis using machine learning techniques and ecu data. In: IEEE. *2023 IEEE 32th International Symposium on Industrial Electronics (ISIE)*. [S.l.], 2023.

CHINONSO, E. A.; ANAYO, O. H.; ANIKWE, C. V. Vehicle monitoring system based on iot, using 4g/lte. *International Journal of Engineering and Management Research*, v. 11, 2021.

CIVELEK, U. *A Software tool for vehicle calibration, diagnosis and test via controller area network*. Dissertação (Mestrado) — Middle East Technical University, 2012.

DEVADZE, S. et al. Fast extended test access via jtag and fpgas. In: IEEE. *2009 International Test Conference*. [S.l.], 2009. p. 1–7.

DIETSCHE, K.-H.; KONRAD, R. et al. *Automotive Handbook 11th Edition*. [S.l.]: Robert Bosch GmbH, 2022. 1620 p. ISBN 9781119911906.

DU, B.; STERPONE, L. An fpga-based testing platform for the validation of automotive powertrain ecu. In: *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. [S.l.: s.n.], 2016. p. 1–7.

DU, X.; QI, B.; ZHENG, M. Calibration system for battery management system based on can calibration protocol. In: IEEE. *2011 6th IEEE Conference on Industrial Electronics and Applications*. [S.l.], 2011. p. 2310–2312.

DZAMBIC, M. et al. A rapid prototyping system, intelligent watchdog and gateway tool for automotive applications. In: IEEE. *2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C)*. [S.l.], 2021. p. 149–154.

DZHELEKARSKI, P.; ALEXIEV, D. Initializing communication to vehicle obdii system. *ELECTRONICS*, v. 5, p. 21, 2005.

ETAS. *ES581.3 CAN Bus Interface USB Module User's Guide 2*. [S.l.], 2011. Available at <https://www.etas.com/download-center-files/products_ES500/ES581.3_UG_EN.pdf>.

ETAS. *INCA-MCE V7.3 User's Guide*. [S.l.], 2020. Available at <https://www.etas.com/download-center-files/products_INCA_Software_Products/INCA-MCE_V7.3_Users_Guide_EN.pdf>.

ETAS. *ETAS ES922.1 CAN FD Module (2 Channels)*: User guide. [S.l.], 2021. Disponível em: <https://www.etas.com/download-center-files/products_ES900/etas-es922-user-guide-r04-en-20220221.pdf>.

ETAS. *ETAS ETK-S4.2 Emulator Probe for Infineon TriCore TC17xx*: User guide. [S.l.], 2021. Disponível em: <https://www.etas.com/download-center-files/products_ETK/etas-etk-s4.2-user-guide-r08-en-202107.pdf>.

ETAS. *ETAS BR-XETK-S2.0 Emulator Probe for JDP MPC57xx Microprocessor Family*: User guide. [S.l.], 2022.

ETAS. *ETAS INCA V7.4*: Getting started. [S.l.], 2022. Disponível em: <https://www.etas.com/download-center-files/products_INCA_Software_Products/etas-inca-v7.4-getting-started-en-202203.pdf>.

ETAS. *ETAS ODX-LINK V7.4*: User guide. [S.l.], 2022. Disponível em: <https://www.etas.com/download-center-files/products_INCA_Software_Products/etas-inca-odx-link-v7.4-users-guide-r02-en-202206.pdf>.

ETAS. *ETK / FETK / XETK – ECU Interfaces*. 2022. Disponível em: <https://www.etas.com/en/products/etk_fetk_xetk_ecu_interfaces.php>. Acesso em: 20 de jul. de 2022.

ETAS. *ETK-S20.1A – ECU Interface*. 2022. Disponível em: <https://www.etas.com/en/products/etks20.php>. Acesso em: 20 de jul. de 2022.

ETAS. *On and Off board Diagnostics*. 2022. Disponível em: <https://www.etas.com/en/applications/applications_onboard_offboard_diagnostics.php>. Acesso em: 20 de jul. de 2022.

FAGHANI, E.; ANDRIC, J.; SJOBLOM, J. *Toward an effective virtual powertrain calibration system*. [S.l.], 2018.

FELDMAN, B. *Onix: por quê o motor explode e o carro pega fogo?* 2019. Disponível em: <https://autopapo.uol.com.br/noticia/onix-por-que-carro-pega-fogo/>.

FERNANDES, J. N. O. A real-time embedded system for monitoring of cargo vehicles, using controller area network (can). *IEEE Latin America Transactions*, v. 14, n. 3, p. 1086–1092, 2016.

FERRARI, P. et al. On the use of lorawan for the internet of intelligent vehicles in smart city scenarios. In: *2020 IEEE Sensors Applications Symposium (SAS)*. [S.l.: s.n.], 2020. p. 1–6.

FRANCIS, L. T. et al. Data-driven anomaly detection of engine knock based on automotive ecu. In: *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*. [S.l.: s.n.], 2022. p. 1–8.

GARCÍA, S. A. et al. Desenvolvimento de software embarcado para aquisição e calibração de ecus automotivas. In: *Anais do XXIX Simpósio Internacional de Engenharia Automotiva*. [s.n.], 2022. v. 9, n. 2, p. 50 – 60. ISSN 2357-7592. Disponível em: <https://doi.org/10.5151/simea2022-pap10>.

GROZA, B.; POPA, L.; MURVAY, P.-S. Tricks—time triggered covert key sharing for controller area networks. *IEEE Access*, v. 7, p. 104294–104307, 2019.

GUI, Y.; SIDDIQUI, A. S.; SAQIB, F. Hardware based root of trust for electronic control units. In: IEEE. *SoutheastCon 2018*. [S.l.], 2018. p. 1–7.

GUSE, D. et al. Next level of testing - extended frontloading through latency-optimized eil test benches. *MTZ worldwide*, v. 81, p. 44–49, 10 2020. ISSN 2192-9114.

HAMID, A. H. F. A. et al. Smart vehicle monitoring and analysis system with iot technology. *International Journal of Integrated Engineering*, v. 11, n. 4, 2019.

HARTFIEL, J. *PetaLinux*. 2021. Disponível em: <https://wiki.trenz-electronic.de/display/PD/PetaLinux>.

HE, Y.; SUN, X. An xcp based distributed calibration system. In: SPRINGER. *Advances in Software Engineering: International Conference on Advanced Software Engineering and Its Applications, ASEA 2009 Held as Part of the Future Generation Information Technology Conference, FGIT 2009, Jeju Island, Korea, December 10-12, 2009. Proceedings*. [S.l.], 2009. p. 9–15.

IEEE. Ieee standard for boundary-scan-based stimulus of interconnections to passive and/or active components. *IEEE Std 1149.8.1-2012*, p. 1–95, 2012.

IGARASHI, T. N. *Desenvolvimento de sistema de aquisição e calibração para ECUs*. 62 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal de Santa Catarina, Joinville, SC, 2022.

IQBAL, M. N. et al. Diagnostic tool and remote online diagnostic system for euro standard vehicles. In: *2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC)*. [S.l.: s.n.], 2017. p. 415–419.

ISERMANN, R. *Automotive control: modeling and control of vehicles*. [S.l.]: Springer, 2022.

JI, C.; KO, T.; HONG, M. Ca-cre: Classification algorithm-based controller area network payload format reverse-engineering method. *Electronics*, v. 10, n. 19, 2021. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/10/19/2442>.

KHAN, M. A. et al. Development & implementation of smart vehicle over speeding detector using iot. *Advances in Science, Technology and Engineering Systems*, v. 4, n. 2, p. 170–175, 2019.

KHEIRALLA, A.; ABUBAKR, O. A.; GAMAL, M. A. Reliable low cost can-logger for off-road vehicle engineering education and research. In: AMERICAN SOCIETY OF AGRICULTURAL AND BIOLOGICAL ENGINEERS. *2019 ASABE Annual International Meeting*. [S.l.], 2019. p. 1.

KÖTTER, M. et al. Powertrain calibration based on x-in-theloop: Virtualization in the vehicle development process. In: SPRINGER. *18. Internationales Stuttgarter Symposium*. [S.l.], 2018. p. 1187–1201.

KUMAR, K. R.; RAVI, H. Automotive vibration datalogger. In: IEEE. *2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT)*. [S.l.], 2022. p. 1–4.

LOUNICI, M. et al. Knock characterization and development of a new knock indicator for dual-fuel engines. *Energy*, v. 141, p. 2351–2361, 2017. ISSN 0360-5442. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0360544217319965>.

MAHENDAR, S. K. *Mitigating Knock in Heavy Duty Spark Ignition Engines: Experiments and simulations of diluted ethanol and methanol combustion*. Tese (Doutorado) — KTH Royal Institute of Technology, 2021.

MALEKIAN, R. et al. Design and implementation of a wireless obd ii fleet management system. *IEEE Sensors Journal*, v. 17, n. 4, p. 1154–1164, 2017.

MALLIDI, S. K. R.; VINEELA, V. Iot based smart vehicle monitoring system. *Int. J. Adv. Res. Comput. Sci*, v. 9, n. 2, p. 738–741, 2018.

MUSTAKIM, H. U. 5g vehicular network for smart vehicles in smart city: A review. *Journal of Computer, Electronic, and Telecommunication*, v. 1, n. 1, 2020.

NATALE, M. D. et al. *Understanding and using the controller area network communication protocol: theory and practice*. [S.l.]: Springer Science & Business Media, 2012.

OFNER, A. B. et al. Knock detection in combustion engine time series using a theory-guided 1-d convolutional neural network approach. *IEEE/ASME Transactions on Mechatronics*, IEEE, v. 27, n. 5, p. 4101–4111, 2022.

PALOMINO, J.; CUTY, E.; HUANACHIN, A. Development of a can bus datalogger for recording sensor data from an internal combustion ecu. In: *2021 IEEE International Workshop of Electronics, Control, Measurement, Signals and their application to Mechatronics (ECMSM)*. [S.l.: s.n.], 2021. p. 1–4.

PATZER, A.; ZAISER, R. Xcp–the standard protocol for ecu development. *Vector Informatik Gmbh: Stuttgart, Germany*, 2016.

PATZER, A.; ZAISER, R. *XCP – The Standard Protocol for ECU Development*: Manual. [S.l.], 2016. Disponível em: <https://cdn.vector.com/cms/content/application-areas/ecu-calibration/xcp/XCP_ReferenceBook_V3.0_EN.pdf>.

PIEROZAN, V. E. *Inteligência Artificial Aplicada para Detecção de Knock em Motores Automotivos*. 54 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal de Santa Catarina, Joinville, SC, 2022.

PUPALA, R.; SHUKLA, J. Review paper on vehicle diagnosis with electronic control unit. *International Journal of Innovative Science and Research Technology*, v. 3, n. 2, p. 117–123, 2018.

SAE. *OBD-II Scan Tool*. [S.l.], 2018.

SATHIYANARAYANAN, M.; MAHENDRA, S.; VASU, R. B. Smart security system for vehicles using internet of things (iot). In: *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*. [S.l.: s.n.], 2018. p. 430–435.

SHEN, X.; ZHANG, Y.; SHEN, T. Cylinder pressure resonant frequency cyclic estimation-based knock intensity metric in combustion engines. *Applied Thermal Engineering*, Elsevier, v. 158, p. 113756, 2019.

SILVA, M. E. da. *Seleção de Atributos em Aprendizado de Máquina para Identrificação de Falhas em Motores de Combustão Interna*. 72 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal de Santa Catarina, Joinville, SC, 2022.

SILVA, M. E. R. da; GRACIOLI, G.; ARAUJO, G. M. de. Feature selection in machine learning for knocking noise detection. In: IEEE. *2022 XII Brazilian Symposium on Computing Systems Engineering (SBESC)*. [S.l.], 2022. p. 1–8.

SIMÕES, D. T.; MAFORT, L. S. S. Automotive event logger based on the esp32 platform. 2021.

SMITH, K.; MILLER, J. Obdii data logger design for large-scale deployments. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. [S.l.: s.n.], 2013. p. 670–674.

VECTOR. *VH4110 IoT Enabler*: Manual. [S.l.], 2021. Disponível em: <https://cdn.vector.com/cms/content/products/VH4110/docs/VH4110_IoT_Enabler_Manual_EN.pdf>.

VECTOR. *VX1000 Manual*. [S.l.], 2021. Disponível em: <https://cdn.vector.com/cms/content/products/vx1000/Docs/VX1000_Manual.pdf>.

VECTOR. *CANape*: Measuring, calibrating and logging of ecus and adas sensors. 2022. Disponível em: <https://www.vector.com/br/pt/produtos/products-a-z/software/canape/#c210350>. Acesso em: 11 de jul. de 2022.

VECTOR. *Interfaces to Interconnect Your PC*. 2022. Disponível em: <https://www.vector.com/br/pt/produtos/products-a-z/hardware/network-interfaces>. Acesso em: 11 de jul. de 2022.

VECTOR. *GL5450*: Manual. [S.l.], 2023. Disponível em: <https://cdn.vector.com/cms/content/products/gl_logger/Docs/GL5450_Fact_Sheet_EN.pdf>.

VECTOR. *MICROSAR Connect - Data Collector*. 2023. Disponível em: <https://www.vector.com/br/pt/products/products-a-z/embedded-components/microsar-connect/vehicle-data-collector/#>.

WANG, J. et al. Development of a new calibration and monitoring system for in-vehicle electronic control units based on controller area network calibration protocol. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, Sage Publications Sage UK: London, England, v. 219, n. 12, p. 1381–1389, 2005.

WEBER, J. *Automotive development processes: Processes for successful customer oriented vehicle development*. [S.l.]: Springer Berlin Heidelberg, 2009. 1-312 p. ISBN 9783642012525.

XILINX. *Zynq UltraScale+ MPSoC*. 2021. Disponível em: <https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/444006775/Zynq+UltraScale+MPSoC>.

ZHEN, X. et al. The engine knock analysis–an overview. *Applied Energy*, Elsevier, v. 92, p. 628–636, 2012.

APÊNDICE A – APLICAÇÃO EM LINGUAGEM C++ PARA DETECÇÃO DE KNOCK NOISE

Este Apêndice apresenta o código fonte da aplicação desenvolvida para detecção de *knock noise*.

- **Programa Principal (*main.cpp*):**

```
1  #include <unistd.h>
2  #include <iostream>
3  #include <chrono>
4  #include <fstream>
5  #include <sstream>
6  #include <cstdlib>
7  #include <signal.h>
8  #include <string>
9  #include "libs/FIRFilter.cpp"
10 #include "libs/serial.cpp"
11 #include "libs/knock_detection.cpp"
12
13 using namespace std;
14 static bool run=0;
15
16 // Exit function (ctr+c)
17 void signal_callback_handler(int signum) {
18    cout << "Caught signal " << signum << endl;
19    // Terminate program
20    run=1;
21    // Exit(signum);
22    return;
23 }
24
25 int main() {
26     // Initialize variables
27     int i=0, j=0, k=0, adcValue = 0, knock_counter=0;
28     float avrg;
29     // Initialize exit function
30     signal(SIGINT, signal_callback_handler);
31
32     auto t1 = std::chrono::duration_cast< std::chrono::
       microseconds >(std::chrono::system_clock::now().
       time_since_epoch()).count();
```

```
33      auto t2 = std::chrono::duration_cast< std::chrono::
         microseconds >(std::chrono::system_clock::now().
         time_since_epoch()).count();
34      auto t3 = std::chrono::duration_cast< std::chrono::
         microseconds >(std::chrono::system_clock::now().
         time_since_epoch()).count();
35
36      // Create and initialize knock count file
37      std::ofstream knock_data;
38      knock_data.open("piezo_outputs/knock_count.csv");
39      knock_data << "value,fir_value,knock_counter,timestamp,
         average,\n";
40
41      // Create knock_count and filter objects
42      knock_verifier knock_count;
43      FIRFilter filter;
44      FIRFilter_Init(& filter);
45
46      // Create and initialize UART communication
47      UART uart_tiva;
48      if(uart_tiva.InitUart()==-1){
49          std::cout << "UART Init ERROR" << "\n";
50          return -1;
51      }
52
53      auto ti = std::chrono::duration_cast< std::chrono::
         microseconds >(std::chrono::system_clock::now().
         time_since_epoch()).count();
54
55      while(run==0){
56          i++;k++;
57          // Get uart value
58          adcValue = uart_tiva.GetBuffer();
59          // Error counter
60          if(adcValue < 0){j++;}
61          // Signal offset
62          adcValue -= 103;
63
64          t2=t1;
65          t1 = std::chrono::duration_cast< std::chrono::
             microseconds >(std::chrono::system_clock::now().
             time_since_epoch()).count();
```

```
66
67          // Display variables each second or if a 1ms delay occurs
68          if ((t1>t3)||(t1-t2)>1000){
69              std::cout << "Piezo_Signal: " << adcValue << "
                 timeStamp: " << t1 << "  Period: " << (t1-t2)
70              << "  Errors: " << j << "  knock occurences: " <<
                 knock_counter << std::endl;
71              t3 = t1+1000000;
72          }
73          // Apply the FIR filter to adcValue
74          FIRFilter_Update(&filter, adcValue);
75
76          // Calculate local maxima average
77          avrg = knock_count.lm_average(filter.out);
78
79          // Detect knock occurrence, print, and save the data
80          if (knock_count.knock_check(filter.out)){
81              knock_counter++;
82              knock_data << adcValue << "," << filter.out << "," <<
                  knock_counter <<"," << t1 << "," << avrg << ","<< "\
                 n";
83              std::cout << "filter.out: " << filter.out << "
                 knock_counter: " << knock_counter << "  avrg: " <<
                 avrg << "  timestamp: " << t1 << endl;
84          }
85          // Save data each 1000 samples if knock is not detected
86          else if(k>=1000){
87              k=0;
88              knock_data << signal << "," << filter.out << "," <<
                 knock_counter <<"," << t1 << "," << avrg << ","<< "\n
                 ";
89          }
90      }
91      auto tf = std::chrono::duration_cast< std::chrono::
         microseconds >(std::chrono::system_clock::now().
         time_since_epoch()).count();
92
93      // Print final experiment information
94      std::cout << "total time: " << (tf-ti) << std::endl;
95      std::cout << "Errors: " << j << std::endl;
96      std::cout << "knock occurences: " << knock_counter << endl;
97
```

```
98      // Close uart and knock file.
99      uart_tiva.CloseUart();
100     knock_data.close();
101     return 0;
102 }
```

- **Classe de Configuração da Porta de Comunicação Serial (*serial.cpp*):**

```
1  #include <fcntl.h>
2  #include <termios.h>
3  #include <unistd.h>
4  #include <iostream>
5  #include <cstring>
6  #include <sys/ioctl.h>
7  #include <queue>
8  #include <ctime>
9  #include <errno.h>
10 #include <vector>
11 #include <thread>
12 #include <chrono>
13
14 #define DATA_SIZE 1
15
16 using namespace std::this_thread; // sleep_for, sleep_until
17 using namespace std::chrono;      // nanoseconds, system_clock,
    seconds
18
19 class UART
20 {
21 public:
22     int serial_port;
23     struct termios tty;
24     std::string port_ = "/dev/ttyACM0";
25     int adcValue = 0;
26     uint8_t buffer[DATA_SIZE];
27     uint8_t buffer_send[1];
28     int i=0;
29
30     int InitUart()
31     {
32         std::cout << "Entering: InitUart()\n";
33         serial_port = open(port_.c_str(), O_RDWR | O_NOCTTY);
```

```
34          if (serial_port < 0)
35          {
36              std::cout << "Error opening serial port, error " <<
                 errno << " " << strerror(errno) << "\n";
37              std::cout << "Leaving: InitUart()\n";
38              return -1;
39          }
40
41          if (tcgetattr(serial_port, &tty) != 0)
42          {
43              std::cout << "Error on tcgetattr(), error " << errno
                 << " " << strerror(errno) << "\n";
44              std::cout << "Leaving: InitUart()\n";
45              return -1;
46          }
47
48          tty.c_cflag &= ~PARENB;         // Clear parity bit
49          tty.c_cflag &= ~CSTOPB;         // One stop bit
50          tty.c_cflag |= CS8;             // 8 bits per byte
51          tty.c_cflag &= ~CRTSCTS;        // Disable hardware flow
             control
52          tty.c_cflag |= CREAD | CLOCAL; // Turn on READ & ignore
             ctrl lines (CLOCAL = 1)
53          tty.c_lflag &= ~ICANON;         // Disable canonical mode
54
55          tty.c_lflag &= ~ECHO;   // Disable echo
56          tty.c_lflag &= ~ECHOE;  // Disable erasure
57          tty.c_lflag &= ~ECHONL; // Disable new-line echo
58          tty.c_lflag &= ~ISIG;   // Disable interpretation of INTR
             , QUIT and SUSP
59
60          tty.c_iflag &= ~(IXON | IXOFF | IXANY);
                                                // Turn off software
              flow ctrl
61          tty.c_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP |
             INLCR | IGNCR | ICRNL); // Disable any special handling
             of received bytes
62
63          tty.c_cc[VTIME] = 1; // Timeout after read() or after
             receiving first bytes
64          tty.c_cc[VMIN] = 1;   // Number of bytes to be received
65
```

```
66          cfsetispeed(&tty, B2000000);
67          cfsetospeed(&tty, B2000000);
68
69          if (tcsetattr(serial_port, TCSANOW, &tty) != 0)
70          {
71              std::cout << "Error on tcsetattr(), error " << errno
                 << " " << strerror(errno) << "\n";
72              std::cout << "Leaving: InitUart()\n";
73              return -1;
74          }
75
76          tcflush(serial_port, TCIOFLUSH);
77          sleep_for(nanoseconds(1000000));
78          tcflush(serial_port, TCIOFLUSH);
79
80          std::cout << "Leaving: InitUart()\n";
81          return 1;
82      }
83
84      int CloseUart()
85      {
86          std::cout << "Entering: CloseUart()\n";
87          write(serial_port, buffer_send, 1);
88          close(serial_port);
89          std::cout << "Leaving: CloseUart()\n";
90          return 0;
91      }
92
93      int GetBuffer()
94      {
95          if(i==0){
96              i++;
97              buffer_send[0]=255;
98              write(serial_port, buffer_send, 1);
99          }
100
101         if(read(serial_port, buffer, DATA_SIZE)>=0){
102             adcValue = buffer[0];
103             return adcValue;
104         }
105         else{
106             write(serial_port, buffer_send, 1);
```

```
107                 tcflush(serial_port, TCIOFLUSH);
108                 sleep_for(nanoseconds(10000));
109                 i=0;
110                 return -1;
111             }
112             return -3;
113         }
114 };
```

- **Classe de Aplicação do Filtro FIR (*FIRFilter.cpp*):**

```
1  /*
2  Adaptado de: github.com/pms67/LittleBrain-STM32F4-Sensorboard/
     blob/master/Firmware/Core/Src/FIRFilter.c
3  */
4  #include "FIRFilter.h"
5
6  static float FIR_IMPULSE_RESPONSE[FIR_FILTER_LENGTH] = {
7    0.03134118558806809,
8    -0.02411960938220963,
9    -0.01692751370140494 3,
10   -0.013803318994651852,
11   -0.014164217646035922,
12   -0.016033537865188206,
13   -0.01657295497519033,
14   -0.013259990996602137,
15   -0.005257026156015486,
16   0.005983480208821029,
17   0.017230935502882315,
18   0.024820826483776884,
19   0.02615928291226718,
20   0.02070751961175557,
21   0.010067610390319393,
22   -0.0027735701317648438,
23   -0.014616571032428315,
24   -0.023071308844593186,
25   -0.027011198421952997,
26   -0.026395461633057823,
27   -0.021747492217989037,
28   -0.013718317222135493,
29   -0.0030217885205670878,
30   0.009285481061891919,
```

```
    0.02147727041601375,
    0.03114189121653475,
    0.035661197832957695,
    0.03307390787249496,
    0.022964667012633534,
    0.0069478800619738375,
    -0.011543233954055399,
    -0.028158764103553773,
    -0.03891252586715132,
    -0.041352932420183626,
    -0.03518627637072678,
    -0.022165286727993033,
    -0.00537690026653829,
    0.011710910426193977,
    0.026057012579977563,
    0.03548427058965797,
    0.03875707378986779,
    0.03548427058965797,
    0.026057012579977563,
    0.011710910426193977,
    -0.00537690026653829,
    -0.022165286727993033,
    -0.03518627637072678,
    -0.041352932420183626,
    -0.03891252586715132,
    -0.028158764103553773,
    -0.011543233954055399,
    0.0069478800619738375,
    0.022964667012633534,
    0.03307390787249496,
    0.035661197832957695,
    0.03114189121653475,
    0.02147727041601375,
    0.009285481061891919,
    -0.0030217885205670878,
    -0.013718317222135493,
    -0.021747492217989037,
    -0.026395461633057823,
    -0.027011198421952997,
    -0.023071308844593186,
    -0.014616571032428315,
    -0.002773570131764843 8,
```

```
    0.010067610390319393,
    0.02070751961175557,
    0.02615928291226718,
    0.024820826483776884,
    0.017230935502882315,
    0.005983480208821029,
    -0.005257026156015486,
    -0.013259990996602137,
    -0.01657295497519033,
    -0.016033537865188206,
    -0.014164217646035922,
    -0.013803318994651852,
    -0.016927513701404943,
    -0.02411960938220963,
    0.03134118558806809
};

void FIRFilter_Init(FIRFilter *fir) {
        /* Clear filter buffer */
        for (uint8_t n = 0; n < FIR_FILTER_LENGTH; n++) {
                fir->buf[n] = 0.0f;
        }
        /* Reset buffer index */
        fir->bufIndex = 0;

        /* Clear filter output */
        fir->out = 0.0f;
}

float FIRFilter_Update(FIRFilter *fir, float inp) {
        /* Store latest sample in buffer */
        fir->buf[fir->bufIndex] = inp;

        /* Increment buffer index and wrap around if necessary */
        fir->bufIndex++;

        if (fir->bufIndex == FIR_FILTER_LENGTH) {
                fir->bufIndex = 0;
        }
        /* Compute new output sample (via convolution) */
        fir->out = 0.0f;
        uint8_t sumIndex = fir->bufIndex;
```

```
115
116         for (uint8_t n = 0; n < FIR_FILTER_LENGTH; n++) {
117                 /* Decrement index and wrap if necessary */
118                 if (sumIndex > 0) {
119                         sumIndex--;
120                 } else {
121                         sumIndex = FIR_FILTER_LENGTH - 1;
122                 }
123                 /* Multiply impulse response with shifted input
                     sample and add to output */
124                 fir->out += FIR_IMPULSE_RESPONSE[n] * fir->buf[
                     sumIndex];
125         }
126         /* Return filtered output */
127         return fir->out;
128
129 }
```

- **Classe de Detecção de Knock Noise (*knock_detection.cpp*):**

```
1  #include <cstdio>
2  #include <iostream>
3  #include <cstdlib>
4  #include <fstream>
5  #include <string>
6  #include <bits/stdc++.h>
7
8  using namespace std;
9
10 #define LOCAL_MAXIMA_FACTOR 1.70
11 #define LOCAL_MAXIMA_DELAY 50000
12 #define LOCAL_MAXIMA_SAMPLES 10000
13 #define AVERAGE_LENGTH 10
14 #define AVERAGE_WEIGHT 2
15 #define KNOCK_THRESHOLD 1.75
16 #define KNOCK_DELAY 500
17
18 class knock_verifier
19 {
20 public:
21     float l_max, lm_avrg=10;
22     bool knock;
```

```
23      float lm_average(float signal){
24          l_max = local_maxima(signal, lm_avrg);
25          lm_avrg = local_maxima_avrg(l_max);
26          return lm_avrg;
27      }
28      bool knock_check(float signal){
29          knock = knock_count(signal, lm_avrg);
30          return knock;
31      }
32
33  private:
34      int i=0, j=LOCAL_MAXIMA_DELAY, k=0, l=0, m=0;
35      float l_max_mem=0, l_max_mem_=0, l_max_sum=0, l_max_avrg=10;
36      float l_max_array[AVERAGE_LENGTH]={};
37
38      // Local maxima function
39      float local_maxima(float signal, float l_max_avrg_){
40          i++; j++;
41          if(signal >= LOCAL_MAXIMA_FACTOR*l_max_avrg_){
42              j=0;
43          }
44          if(j+1==LOCAL_MAXIMA_DELAY){
45              i=0;
46          }
47          if(signal>l_max_mem_ && j>=LOCAL_MAXIMA_DELAY){
48              l_max_mem = signal;
49              l_max_mem_ = signal;
50              j=LOCAL_MAXIMA_DELAY;
51          }
52          if(i>=LOCAL_MAXIMA_SAMPLES){
53              l_max_mem_=0;
54          }
55          return l_max_mem;
56      };
57
58      // Local maxima average function
59      float local_maxima_avrg(float l_max){
60          if(i>=LOCAL_MAXIMA_SAMPLES && j>=LOCAL_MAXIMA_DELAY){
61              i=0;
62              if(k>=AVERAGE_LENGTH){
63                  k=0;
64              }
```

```
65              l_max_array[k] = l_max;
66              if((k+1)<AVERAGE_LENGTH){
67                  l_max_sum = l_max_sum - l_max_array[k+1] +
                     l_max_array[k];
68              }
69              else{
70                  l_max_sum = l_max_sum - l_max_array[0] +
                     l_max_array[k];
71              }
72              if((k+1)>=AVERAGE_LENGTH || l == 1){
73                  l_max_avrg = (l_max_sum + (AVERAGE_WEIGHT*
                     l_max_array[k]))/ (AVERAGE_LENGTH-1+
                     AVERAGE_WEIGHT);
74                  l = 1;
75              }
76              else{
77                  l_max_avrg = (l_max_sum + (AVERAGE_WEIGHT*
                     l_max_array[k]))/ (k+1+AVERAGE_WEIGHT);
78              }
79              k++;
80          }
81          return l_max_avrg;
82      }
83      // Knock verifier
84      bool knock_count(float signal, float l_max_avrg__){
85          if (signal > (KNOCK_THRESHOLD*l_max_avrg__) && m>
             KNOCK_DELAY){
86              m=0;
87              return 1;
88          }
89          else{
90              m++;
91              return 0;
92          }
93      }
94  };
```