UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

TIAGO PORSCH DOPKE

ESTRATÉGIAS PARA CONTORNAR AS LIMITAÇÕES DE LARGURA DE BANDA EM AQUISIÇÃO DE DADOS VEICULARES QUE UTILIZAM O PROTOCOLO XCP

JOINVILLE
2025

TIAGO PORSCH DOPKE

ESTRATÉGIAS PARA CONTORNAR AS LIMITAÇÕES DE LARGURA DE BANDA EM AQUISIÇÃO DE DADOS VEICULARES QUE UTILIZAM O PROTOCOLO XCP

Trabalho apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia Mecatrônica, no curso de Engenharia Mecatrônica, do Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Orientador: Dr. Anderson Wedderhoff Spengler

JOINVILLE
2025

Dedico esta monografia primeiramente a Deus, que sempre me amou, cuidou e guiou.

Aos meus pais, Gerson e Karin, por me incentivarem e me motivarem a chegar onde estou.

Aos membros do LISHA, pela ajuda e companheirismo de todos os dias.

À Luiza, por todo seu amor.

“Se podes?”, disse Jesus. “Tudo é possível àquele que crê”.

(Marcos 9:23)

RESUMO

A crescente complexidade eletrônica dos veículos modernos exige a aquisição de grandes volumes de dados em tempo real, um processo atualmente limitado pela restrita largura de banda do protocolo XCP sobre o barramento CAN clássico. Diante desse gargalo, o presente trabalho tem como objetivo analisar e comparar duas estratégias de otimização: a migração do transporte para o protocolo CAN FD e a implementação de troca dinâmica de listas de variáveis sobre o CAN tradicional. A metodologia baseou-se em uma abordagem experimental comparativa, utilizando uma ECU emulada para implementar, testar e validar o desempenho de ambas as técnicas em cenários controlados. Os resultados demonstraram que, embora a troca dinâmica seja tecnicamente viável, ela apresenta limitações práticas devido à complexidade de configuração e latência de 1.179 s na troca de contexto. Em contrapartida, o CAN FD obteve desempenho superior, permitindo a aquisição integral das variáveis com ampla margem de banda e sem impactos negativos no sistema. Conclui-se que a adoção do CAN FD representa a solução mais direta, robusta e escalável para atender às demandas da indústria automotiva.

Palavras-chave: aquisição de dados veiculares; CAN FD; XCP.

ABSTRACT

The growing electronic complexity of modern vehicles requires the acquisition of large volumes of data in real time, a process currently limited by the restricted bandwidth of the XCP protocol on the classic CAN bus. Given this bottleneck, this study aims to analyze and compare two optimization strategies: migration of transport to the CAN FD protocol and implementation of dynamic variable list switching over traditional CAN. The methodology was based on a comparative experimental approach, using an emulated ECU to implement, test, and validate the performance of both techniques in controlled scenarios. The results showed that, although dynamic switching is technically feasible, it has practical limitations due to the complexity of configuration and the 1.179 s latency in context switching. In contrast, CAN FD achieved superior performance, allowing full acquisition of variables with ample bandwidth and no negative impacts on the system. It is concluded that the adoption of CAN FD represents the most direct, robust, and scalable solution to meet the demands of the automotive industry.

Keywords: vehicle data acquisition; CAN FD; XCP.

SUMÁRIO

1. INTRODUÇÃO

A indústria automotiva contemporânea é marcada por uma crescente complexidade eletrônica, onde os veículos modernos podem ser considerados sistemas mecatrônicos sobre rodas, integrando dezenas ou até centenas de Unidades de Controle Eletrônico (ECUs). Essas unidades são responsáveis pelo gerenciamento de subsistemas críticos como injeção de combustível, transmissão, freios ABS, controle de estabilidade e sistemas avançados de assistência ao motorista. Nesse cenário, a capacidade de monitorar, validar e diagnosticar o comportamento desses sistemas em tempo real é fundamental para o ciclo de desenvolvimento, garantia de qualidade e manutenção do produto. A aquisição de dados veiculares emerge, portanto, como uma área de conhecimento indispensável na engenharia automotiva, fornecendo a base necessária para a análise de desempenho e a identificação de falhas.

O tema específico desta pesquisa se concentra nos protocolos utilizados para a aquisição de dados em ECUs, com ênfase na comunicação via rede CAN (*Controller Area Network*), o padrão para a comunicação intraveicular (Robert Bosch GmbH, 2022). Sobre esta camada de rede, opera o protocolo XCP (*Universal Measurement and Calibration Protocol*), normatizado pela ASAM (*Association for Standardisation of Automation and Measuring Systems*), que se consolidou como uma solução robusta para medição e calibração de parâmetros internos das ECUs. A implementação do XCP sobre o barramento CAN clássico (XCP on CAN) é amplamente difundida na indústria, permitindo que engenheiros e desenvolvedores acessem variáveis de software em tempo real (Patzer; Zaiser, 2016).

Contudo, a evolução dos sistemas embarcados impõe uma demanda cada vez maior por volume e frequência de dados, expondo uma limitação inerente à arquitetura do XCP sobre o CAN clássico (Esparza; Leichtfried; Gonzalez, 2017). O tamanho máximo da carga útil (*payload*) de uma mensagem CAN é de 8 bytes e sua taxa de transmissão nominal é limitada a 1 Mbit/s, o que restringe severamente a quantidade de variáveis que podem ser adquiridas simultaneamente sem causar saturação do barramento (Robert Bosch GmbH, 1991). Esta limitação representa um gargalo significativo em cenários de teste complexos, onde a correlação entre um grande número de parâmetros é essencial para um diagnóstico preciso. Diante deste obstáculo, surge o seguinte problema de pesquisa: como superar as limitações de largura de banda do protocolo XCP sobre CAN para viabilizar a aquisição de um grande volume de dados veiculares de forma eficiente e em tempo real?

A literatura aponta, fundamentalmente, para a evolução do hardware e da camada física da comunicação, com a migração para o protocolo CAN FD (*CAN with Flexible Data-Rate*). O CAN FD expande a carga útil da mensagem para até 64 bytes e permite taxas de transmissão na fase de dados de até 10 Mbit/s, representando um aumento substancial na capacidade de vazão de dados (Robert Bosch GmbH, 2012). Estudos recentes têm explorado os ganhos de desempenho dessa migração em diversas aplicações automotivas (Esparza; Leichtfried; Gonzalez, 2017; Marcon Zago; Freitas, 2018). Outra abordagem se baseia em uma otimização da camada de software e da estratégia de aquisição, por meio da a troca dinâmica de

listas de variáveis. Essa técnica permite que o sistema de medição altere, durante a execução, o conjunto de variáveis que está sendo monitorado, adaptando-se às necessidades momentâneas da análise sem interromper o processo.

Este trabalho foi desenvolvido no âmbito do projeto IASE (*Intelligent Acquisition and Analysis System for ECUs*), fruto da parceria entre o LISHA (Laboratório de Integração de *Software* e hardware) e a Renault do Brasil, com fomento da FUNDEP (Fundação de Desenvolvimento da Pesquisa) por meio do Programa Rota 2030. O referido projeto visa o desenvolvimento de um protótipo de baixo custo para a aquisição de variáveis em tempo real de ECUs automotivas (Bedretchuk *et al.*, 2023). No decorrer da pesquisa, emergiu a necessidade de coletar um volume de dados progressivamente maior do veículo de testes, tornando a abordagem até então utilizada insuficiente.

Para endereçar essa questão, apresenta-se neste trabalho a avaliação, de maneira isolada, das duas abordagens previamente descritas: a migração do protocolo de comunicação para o CAN FD e a implementação da troca dinâmica de variáveis a serem coletadas. Adicionalmente, a fim de facilitar o processo de validação, apresenta-se o desenvolvimento de uma ECU emulada, que permite a execução de testes em bancada sem a necessidade de um veículo real.

1.1. OBJETIVOS

Considerando a problemática identificada na introdução, que trata do limite inerente ao barramento CAN quanto à capacidade de transmissão de dados pelo protocolo XCP, propõe-se os seguintes objetivos:

1.1.1. Objetivo Geral

Analisar e comparar o desempenho de duas estratégias para a otimização da aquisição de dados veiculares via XCP: a migração da camada de transporte para CAN FD e a implementação de um sistema de troca dinâmica de listas de variáveis sobre o CAN clássico.

1.1.2. Objetivos Específicos

- Desenvolver uma ECU emulada para permitir os testes de bancada;
- Migrar a camada de transporte do sistema para o protocolo CAN FD;
- Projetar um sistema de troca dinâmica de variáveis adquiridas;
- Comparar as abordagens pela viabilidade e capacidade de aquisição de dados.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos teóricos que servem de alicerce para o desenvolvimento deste trabalho. A estrutura parte de uma visão geral sobre as redes de comunicação veiculares, aprofundando-se nas tecnologias de barramento CAN e sua evolução, o CAN FD. Em seguida, detalha-se o protocolo de aplicação XCP e integra-se esses conceitos para descrever o processo de aquisição de dados veiculares, contextualizando as estratégias de otimização que são o foco desta pesquisa. Por fim, também são detalhados sistemas comercias que exercem essas funções na presente indústria.

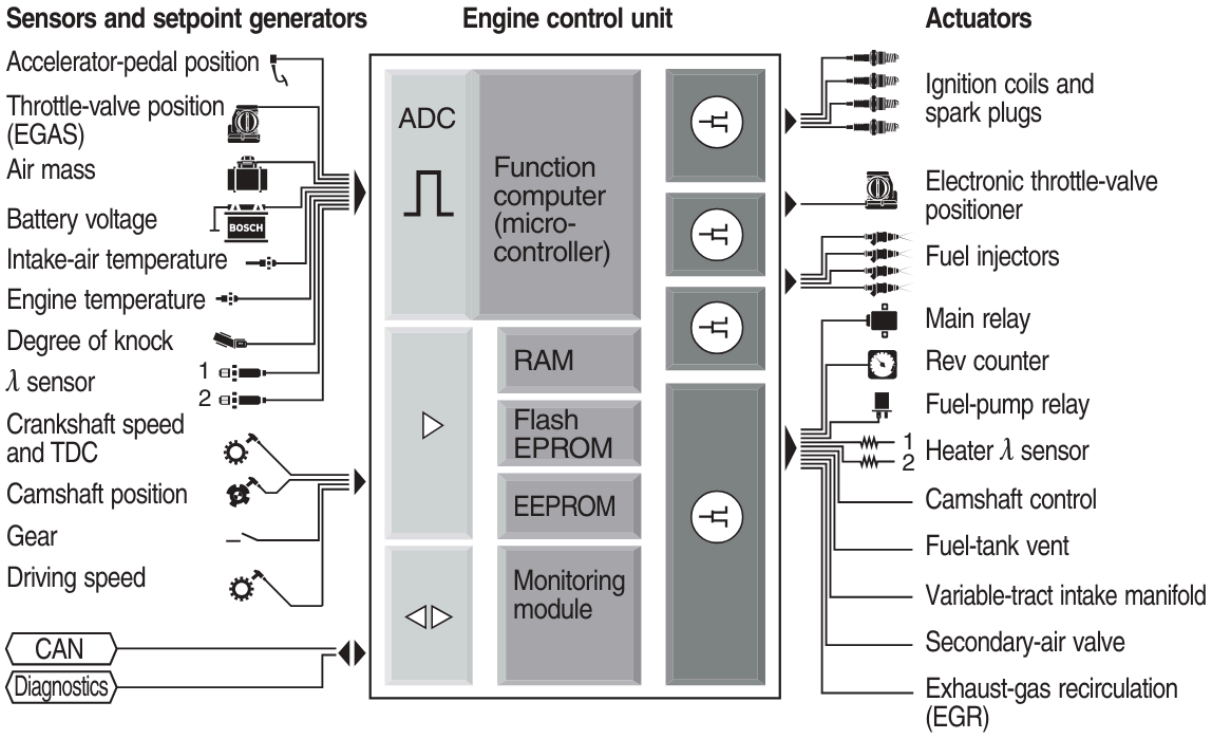
2.1. REDES DE COMUNICAÇÃO VEICULARES

A arquitetura eletrônica de um veículo moderno é um sistema distribuído complexo, composto por múltiplas ECUs. Para que essas unidades operem de forma coesa, é imperativo que troquem informações de maneira rápida e confiável. As redes de comunicação veicular são os sistemas nervosos que permitem essa troca de dados, substituindo a necessidade de complexos e pesados chicotes elétricos ponto a ponto (Robert Bosch GmbH, 2022).

2.1.1. Unidade de Controle Eletrônico

A ECU é um sistema embarcado essencial na arquitetura automotiva, projetado para processar dados, comunicar-se em rede e gerenciar sistemas do veículo a fim de otimizar sua operação (Gui; Siddiqui; Saqib, 2018). O componente central de uma ECU, como evidenciado na Figura 1, é o microcontrolador, que executa um software para interpretar sinais de entrada — provenientes de sensores ou de outras ECUs — e, a partir desse processamento, comanda atuadores (Robert Bosch GmbH, 2022). Para viabilizar essa arquitetura distribuída, as ECUs modernas utilizam múltiplas interfaces de comunicação, sendo as redes CAN, LIN (*Local Interconnect Network*) e FlexRay as prevalentes na indústria (Isermann, 2022). Além de sua função primária de controle em tempo real, a ECU desempenha um papel crucial como uma interface para o mundo externo, permitindo a conexão com ferramentas de diagnóstico e desenvolvimento. Essa capacidade de comunicação externa é fundamental para engenheiros e técnicos, sendo utilizada para a verificação de falhas, otimização de desempenho e, de forma central para este trabalho, para a aquisição de dados durante testes de rodagem e calibração de parâmetros do veículo (Robert Bosch GmbH, 2022).

Figura 1 – Componentes utilizados para o controle eletrônico de um motor



Fonte: Robert Bosch GmbH (2022).

2.1.2. Protocolos de Comunicação

Existem diversos padrões de redes veiculares, cada um projetado para uma área de aplicação específica, variando em custo, velocidade e tolerância a falhas. Entre os comumente utilizados, destacam-se:

- LIN: Um barramento de baixo custo e baixa velocidade, utilizado para conectar sensores e atuadores simples em aplicações de conforto, como vidros elétricos e controle de assentos (Robert Bosch GmbH, 2022);
- FlexRay: Um protocolo determinístico e de alta velocidade, projetado para sistemas críticos de segurança (*x-by-wire*) que exigem comunicação síncrona e tolerante a falhas (Robert Bosch GmbH, 2022);
- Ethernet Automotivo: Utilizado em aplicações que demandam altíssima largura de banda, como sistemas de infoentretenimento, câmeras de alta resolução e como *backbone* para conectar diferentes domínios da arquitetura veicular (Robert Bosch GmbH, 2022);
- CAN: Protocolo amplamente difundido na indústria automotiva, oferecendo um excelente balanço entre custo, robustez e desempenho para a maioria das aplicações de controle em tempo real (Robert Bosch GmbH, 2012).

Devido à sua predominância e relevância direta para este trabalho, o protocolo CAN e sua evolução, CAN FD, serão detalhados a seguir.

2.2. CONTROLLER AREA NETWORK (CAN)

Desenvolvido pela Robert Bosch GmbH na década de 1980 e posteriormente padronizado pela ISO 11898, o protocolo CAN foi projetado para ser um barramento serial robusto e confiável para aplicações em tempo real. Sua popularidade deriva da sua capacidade de operar em ambientes com alto ruído eletromagnético, como o de um veículo, e de seu eficiente mecanismo de arbitragem (Robert Bosch GmbH, 1991).

2.2.1. Camada Física e Topologia

O meio físico do protocolo CAN é constituído por um par de fios trançados, responsáveis pela transmissão diferencial dos sinais (CAN *High* e CAN *Low*). Essa configuração confere ao sistema elevada imunidade a ruídos eletromagnéticos devido ao princípio da rejeição de modo comum: o trançamento garante que qualquer interferência externa induza a mesma tensão em ambos os condutores simultaneamente. Como o transceptor interpreta a informação lógica baseando-se na diferença de potencial entre os dois fios, a interferência – presente igualmente em ambos – é anulada matematicamente na subtração do sinal (Robert Bosch GmbH, 1991).

Em relação à topologia, o barramento linear deve ser terminado em suas extremidades por resistores de 120 Ω. Esses componentes realizam o casamento de impedância com o cabo utilizado, cuja impedância característica possui o mesmo valor. O efeito físico provocado por essa terminação é a absorção da energia do sinal elétrico quando este atinge o final da linha, dissipando-a e evitando que a onda reflita e retorne pelo barramento. Sem isso, o fenômeno de reflexão causaria distorções e oscilações capazes de corromper a interpretação dos bits pelos nós da rede (Robert Bosch GmbH, 1991).

2.2.2. Estrutura do Quadro e Arbitragem

O CAN é um protocolo orientado a mensagens, não a nós. Isso significa que as mensagens não possuem endereço de destino, mas sim um identificador (ID) que define seu conteúdo e sua prioridade. Todos os nós na rede recebem a mensagem, mas apenas processam se o ID coincidir com o seu critério de recepção (Robert Bosch GmbH, 1991).

O mecanismo de acesso ao meio do protocolo CAN utiliza um processo denominado arbitragem bit a bit não destrutiva, que se baseia na natureza elétrica do barramento operando em uma configuração lógica *and* (e). Neste sistema, definem-se dois estados lógicos: o estado dominante (representando o bit 0 e fisicamente forçando uma diferença de potencial no barramento) e o estado recessivo (representando o bit 1, que ocorre quando o barramento está em repouso ou flutuando). Como os nomes sugerem, se um nó transmite um bit dominante e outro transmite um bit recessivo simultaneamente, o estado físico resultante no barramento será dominante (Robert Bosch GmbH, 1991).

Quando múltiplos nós iniciam a transmissão ao mesmo tempo, eles enviam seus identificadores começando pelo bit mais significativo. Durante essa fase, cada nó monitora

o barramento enquanto transmite. Se um nó transmite um bit recessivo, mas lê um estado dominante no barramento, ele detecta um conflito e reconhece que outro nó com um ID de maior prioridade (contendo um 0 naquela posição) está transmitindo. Nesse momento, o nó perdedor interrompe imediatamente sua transmissão e passa a atuar apenas como receptor.

Para fins de exemplificação, na Tabela 1 analisa-se uma rede CAN com identificadores de 11 bits composta por dois nós, cujos IDs são 15 (representação binária 00000001111) e 16 (representação binária 00000010000). Na hipótese de a transmissão ser iniciada simultaneamente por ambos, o bit de início e os seis primeiros zeros dos respectivos identificadores são enviados ao barramento sem que ocorra decisão de arbitragem nesta etapa inicial. Contudo, no momento em que o bit 4 do identificador é transmitido, o nó 16 envia o nível lógico 1 (recessivo), ao passo que o nó 15 envia o nível lógico 0 (dominante). Nesse instante, ao monitorar o barramento, o nó 16 detecta a divergência entre o sinal enviado e o nível lógico 0 presente na linha. Consequentemente, a colisão é identificada e a perda da arbitragem é reconhecida pelo hardware. A transmissão do nó 16 é imediatamente interrompida, o que permite ao nó 15 prosseguir com o envio de seus dados sem perda de informações. Demonstra-se, assim, que o dispositivo detentor do menor identificador vence a arbitragem, possuindo a maior prioridade no sistema.

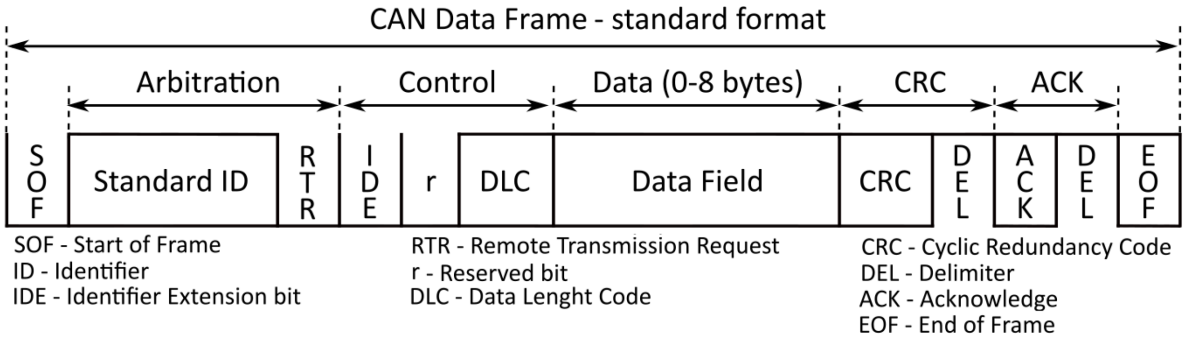
Tabela 1 – Exemplo de colisão de mensagem e arbitragem

	Bit de	**Bits de ID**											**Restante**
	início	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**do quadro**
Nó 15	0	0	0	0	0	0	0	0	1	1	1	1	...
Nó 16	0	0	0	0	0	0	0	1	Parou a transmissão				
CAN data	0	0	0	0	0	0	0	0	1	1	1	1	...

Fonte: Autor (2025).

A estrutura de um quadro CAN clássico, conforme a norma ISO 11898-1 e detalhado na Figura 2, é composta por vários campos, mas o principal para esta pesquisa é o Campo de Dados (*Data Field*), que possui um tamanho máximo de 8 bytes. Essa limitação, juntamente com a taxa de transmissão máxima de 1 Mbit/s, é a principal fonte do gargalo de comunicação que busca-se solucionar (Leen; Heffernan, 2002).

Figura 2 – Visão geral do formato padrão do quadro CAN



Fonte: Popa; Berdich; Groza (2023).

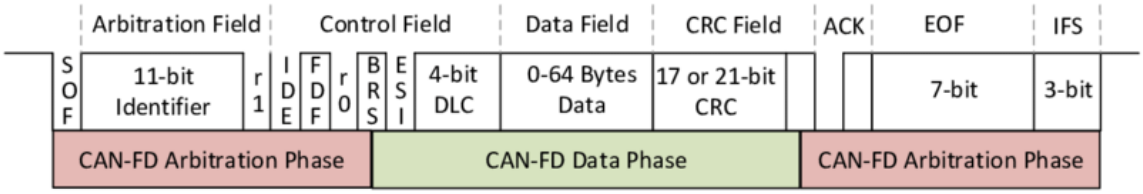
2.3. CONTROLLER AREA NETWORK FLEXIBLE DATA-RATE (CAN FD)

Com o aumento exponencial do número de ECUs e da complexidade do software embarcado, a largura de banda proporcionada pelo CAN clássico tornou-se insuficiente. Para atender a essa nova demanda sem a necessidade de uma mudança disruptiva de protocolo, a Bosch desenvolveu o CAN FD, padronizado em 2012 (Robert Bosch GmbH, 2012).

O CAN FD mantém os princípios fundamentais do CAN clássico, como o mecanismo de arbitragem, mas introduz duas melhorias cruciais que resolvem diretamente os gargalos de comunicação identificados:

- Aumento da Carga Útil: Na Figura 3 é possível observar que o tamanho do campo de dados foi expandido de 8 bytes para até 64 bytes. Isso permite que mais informações sejam enviadas em um único quadro, reduzindo o *overhead* (cabeçalho do protocolo) e aumentando a eficiência da transmissão (Robert Bosch GmbH, 2012);
- Taxa de Dados Flexível (*Flexible Data-Rate*): O quadro CAN FD é dividido em duas fases de transmissão. A primeira, a fase de arbitragem, opera na taxa de bits nominal do CAN (até 1 Mbit/s), mantendo a compatibilidade com a arbitragem tradicional e respeitando o limite do barramento clássico. A segunda, a fase de dados, pode operar a uma taxa de bits mais elevada (tipicamente 2 ou 5 Mbit/s, com suporte para taxas ainda maiores) (Robert Bosch GmbH, 2012).

Figura 3 – Visão geral do formato padrão do quadro CAN FD



Fonte: Joshi *et al.* (2017).

A combinação de uma carga útil maior com uma transmissão mais rápida na fase de dados resulta em um aumento significativo da vazão de dados da rede, tornando o CAN FD

uma solução natural para a evolução das arquiteturas eletrônicas que atingiram o limite do CAN clássico (Di Natale *et al.*, 2012).

Além dessas duas melhorias principais, a estrutura do quadro CAN FD (Figura 3) introduz outras mudanças técnicas significativas em relação ao CAN clássico (Figura 2) para gerenciar o novo formato:

- Campo de Controle Modificado: O Campo de Controle é expandido para incluir bits específicos que gerenciam a nova funcionalidade. Sobressai-se o BRS (*Bit Rate Switch*), que é o bit que efetivamente comanda a mudança da taxa de bits nominal (usada na fase de arbitragem) para a taxa de bits mais elevada (usada na fase de dados) (Robert Bosch GmbH, 2012);
- Codificação do DLC (*Data Length Code*): Embora o campo DLC mantenha 4 bits, sua codificação é alterada. No CAN clássico, valores de DLC de 0 a 8 representavam de 0 a 8 bytes de dados. No CAN FD, essa codificação é expandida para permitir a especificação de cargas úteis maiores, como 12, 16, 20, 24, 32, 48 ou até 64 bytes (Robert Bosch GmbH, 2012);
- CRC (*Cyclic Redundancy Code*) Aprimorado: Com o aumento da carga útil de 8 para 64 bytes, a probabilidade de erros não detectados aumentaria se o mesmo CRC fosse mantido. Por isso, o CAN FD adota um polinômio de CRC mais robusto e um campo de CRC maior, que pode ser de 17 ou 21 bits , em oposição ao CRC de 15 bits (mais o delimitador) do CAN clássico (Robert Bosch GmbH, 2012).

Essas mudanças estruturais demonstram que o CAN FD não é apenas uma versão mais rápida do CAN, mas uma revisão do protocolo de enlace projetada para garantir a robustez e a eficiência com cargas de dados significativamente maiores (Robert Bosch GmbH, 2012).

2.4. UNIVERSAL MEASUREMENT AND CALIBRATION PROTOCOL (XCP)

Enquanto o CAN e o CAN FD operam nas camadas física e de enlace do modelo OSI, o XCP é um protocolo da camada de aplicação. Padronizado pela ASAM, o XCP foi projetado para ser uma interface universal e flexível para acesso à memória interna de ECUs (Schuermans *et al.*, 2003).

Ele opera em um modelo mestre-escravo, onde uma ferramenta de medição e calibração (o mestre, executando em um PC) se comunica com a ECU (o escravo). Suas principais funcionalidades são:

- Medição (*Data Acquisition*): Leitura síncrona de variáveis da ECU (Schuermans *et al.*, 2003);
- Calibração (*Calibration*): Escrita de parâmetros na memória da ECU (Schuermans *et al.*, 2003);
- Reprogramação (*Flashing*): Atualização do software da ECU (Schuermans *et al.*, 2003).

Uma característica fundamental do XCP é sua independência da camada de transporte. Ele foi projetado para rodar sobre diferentes barramentos, como CAN, CAN FD,

Ethernet, *FlexRay* e USB. Quando operando sobre CAN (XCP *on* CAN), as mensagens do protocolo XCP são encapsuladas dentro dos quadros CAN (Schuermans *et al.*, 2003).

O processo de aquisição de dados é o conjunto de etapas que permite a um engenheiro visualizar e registrar o comportamento de variáveis internas de uma ECU em tempo real. Ele se baseia na interação entre a ferramenta mestre (o software de medição) e a ECU escrava, mediada pelo protocolo XCP (Schuermans *et al.*, 2003). O funcionamento desse ecossistema depende de arquivos, componentes e pacotes de dados específicos, detalhados a seguir.

2.4.1. A2L

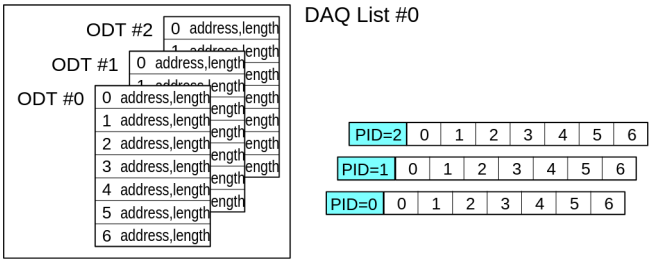
O arquivo A2L, cujo nome deriva do padrão ASAM MCD-2 MC, é a base de todo o processo de medição e calibração. Trata-se de um arquivo de texto formatado que funciona como um dicionário e mapa de memória da ECU. Ele é gerado durante a compilação do software embarcado e contém todas as informações necessárias para que a ferramenta mestre possa interpretar os dados da ECU. Sem o A2L, a ferramenta receberia apenas bytes brutos, sem significado. As informações contidas em um arquivo A2L incluem (ASAM e.V., 2018):

- Descrições de variáveis que podem ser lidas, incluindo nome, endereço na memória e tipo de dado (e.g. inteiro de 16 bits, ponto flutuante de 32 bits);
- Descrições de parâmetros que podem ser lidos e escritos (calibrados);
- Fórmulas de conversão para transformar os valores brutos lidos da memória em unidades de engenharia compreensíveis (e.g. converter um valor de 0-1023 de um sensor para uma temperatura em graus Celsius);
- Informações do protocolo subjacente de comunicação que será utilizado (CAN, CAN FD, LIN, etc).

2.4.2. Object Description Tables (ODTs) e Data Acquisition Lists (DAQs)

Enquanto o arquivo A2L serve como referência externa para a ferramenta mestre, a operacionalização da aquisição de dados internamente na ECU ocorre por meio das Listas DAQ. Este conceito de alto nível organiza a medição permitindo que, em vez de solicitar cada variável individualmente, a ferramenta mestre instrua a ECU a criar grupos de variáveis monitoradas ciclicamente. Para estruturar esses dados na memória volátil, cada Lista DAQ é composta por uma ou mais ODTs. A ODT é a estrutura que define exatamente como a carga útil da mensagem deve ser construída, e é composta de um byte identificador (PID) e N bytes de dados, à depender da capacidade de carga útil do protocolo físico utilizado. Conforme ilustra a Figura 4, essa hierarquia conecta a Lista DAQ aos seus fragmentos de dados (ODTs) e a um evento de tempo (gatilho) que determina o momento da amostragem (ASAM e.V., 2018).

Figura 4 – Organização de uma lista de aquisição de dados

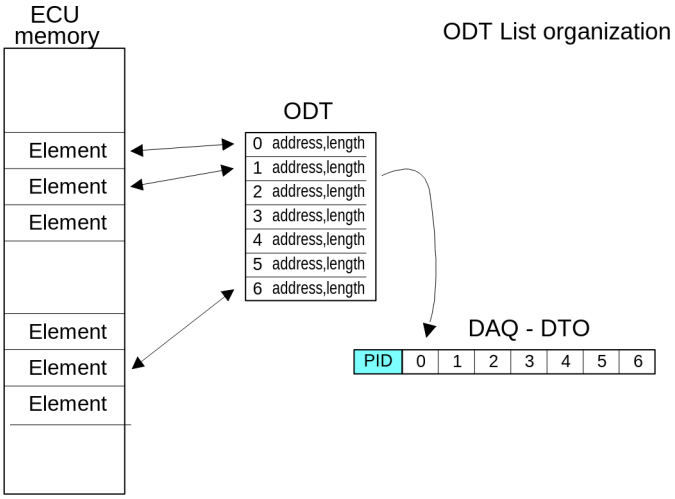


Fonte: Schuermans *et al.* (2003).

Para cada lista DAQ, existe também um número máximo de ODTs definido pelo arquivo A2L. Essas tabelas são configuradas dinamicamente pela ferramenta mestre através de comandos XCP no início da medição (ASAM e.V., 2018).

Como observado na Figura 5, cada ODT é, essencialmente, uma lista de apontadores. Cada entrada na tabela, chamada de ODT *entry*, especifica o endereço de memória de uma variável e o número de bytes a serem lidos a partir daquele endereço. Quando o evento de medição da DAQ ocorre (por exemplo, a cada 10 ms), o *driver* XCP na ECU percorre a ODT, coleta os dados de cada endereço de memória apontado e os concatena na ordem especificada para formar a carga útil do pacote que será enviado (Schuermans *et al.*, 2003).

Figura 5 – Organização de uma lista de ODTs



Fonte: Schuermans *et al.* (2003).

Uma vez ativada, a ECU se encarrega de executar o ciclo de medição daquela lista de forma autônoma e periódica. Múltiplas listas DAQ podem ser configuradas para rodar com diferentes taxas de atualização (e.g. uma lista de 10 ms para sinais rápidos como rotação do motor, e uma de 100 ms para sinais lentos como temperatura do óleo), permitindo uma gestão flexível dos recursos de medição (Schuermans *et al.*, 2003).

Outra característica importante desse sistema de coleta de variáveis é que ele pode ser configurado e reconfigurado sem que haja a necessidade de reiniciar nenhuma das partes, de tal maneira que torna viável a ideia de realizar a troca dinâmica dos conteúdos dessas listas de aquisição em uma tentativa de superar o limite imposto pelo protocolo CAN.

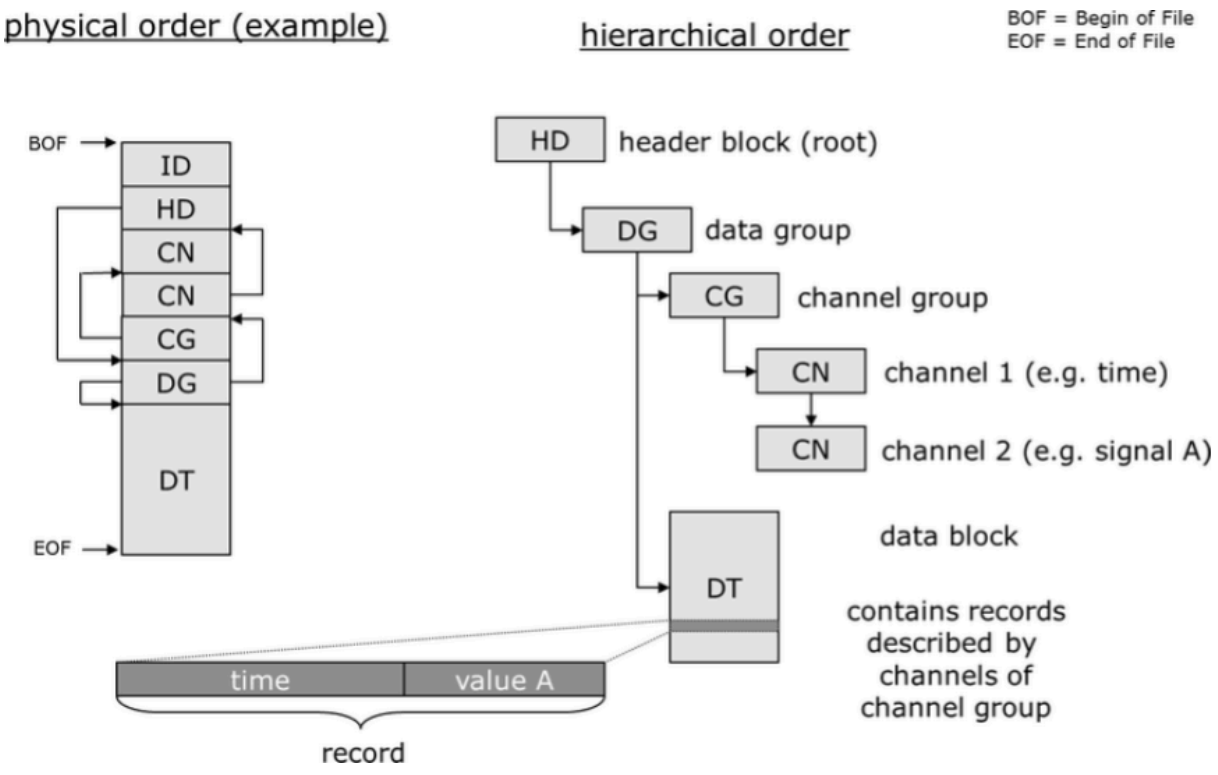
2.5. MEASUREMENT DATA FORMAT (MDF)

Measurement Data Format (MDF) é um formato de arquivo binário padronizado pela ASAM, que pode ser utilizado para aquisição, troca, e análise de dados mensurados. O MDF é bem estabelecido na indústria automotiva, e traz consigo um padrão para troca de dados entre aplicações na indústria automotiva, descrição compacta de dados, e acesso rápido às informações gerais do arquivo independentemente do tamanho dele (ASAM e.V., 2014). Este formato é utilizado extensivamente no projeto IASE para armazenamento dos dados adquiridos.

O formato de arquivo consiste em diferentes blocos de dados com funcionalidades específicas, conectados para formar o arquivo final, como visto na Figura 6 (ASAM e.V., 2014). Para o escopo desse trabalho, utilizou-se apenas um subconjunto de todas as funcionalidades que o MDF proporciona, das quais são importantes relatar:

- *CG*: O *Channel Group* é isomorfo a uma lista DAQ, no sentido de que contém um sumário de variáveis, das quais compartilham um mesmo período de aquisição;
- *CN*: O *Channel* é equivalente a uma variável e traz informações sobre como ela está salva no arquivo.

Figura 6 – Visão geral da estrutura de um arquivo MDF



Fonte: ASAM e.V. (2014).

2.6. SISTEMAS DE AQUISIÇÃO COMERCIAIS

A instrumentação necessária para aplicar os conceitos teóricos de aquisição de dados veiculares é fornecida por um mercado de ferramentas altamente especializadas. A operacionalização de protocolos como o XCP sobre barramentos CAN e CAN FD é viabilizada por

plataformas comerciais que integram hardware de interface e ambientes de software. Dentre os líderes deste setor, destacam-se as soluções oferecidas pela ETAS GmbH, Vector Informatik GmbH e Accurate Technologies Inc. (ATI), cujas abordagens são exploradas a seguir.

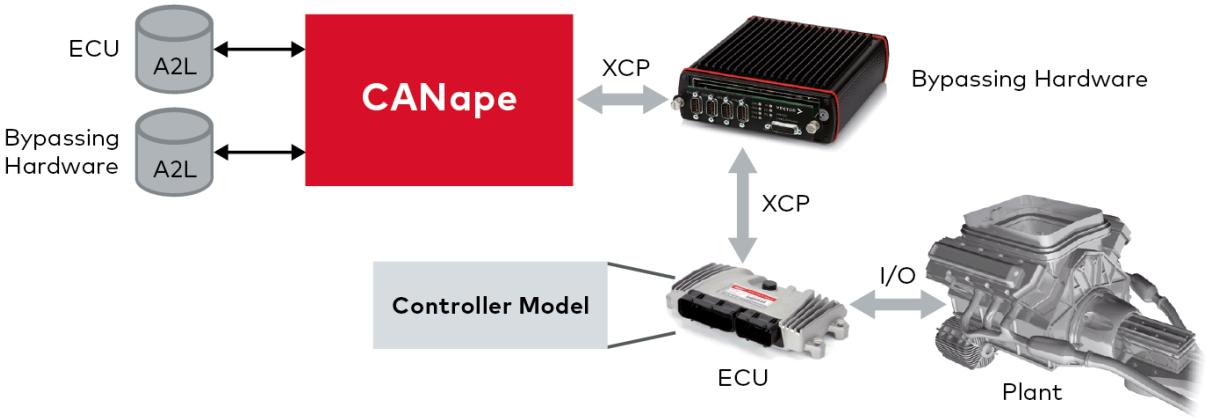
2.6.1. ETAS

Proveniente do grupo Robert Bosch GmbH, a ETAS especializa-se em fornecer um ecossistema integrado para o desenvolvimento de sistemas embarcados. Sua plataforma permite que os engenheiros acessem e manipulem dados internos das ECUs através de interfaces de hardware, como as das famílias ETK e ES. Tais equipamentos estabelecem o elo de comunicação com a unidade de controle, sendo capazes de operar tanto via protocolos padrão da indústria, como XCP, quanto por acesso direto à memória do processador através de interfaces de depuração JTAG (ETAS GmbH, 2021). O ambiente de software que centraliza as operações é o INCA, que desempenha o papel de mestre na comunicação XCP. A partir dele, o engenheiro pode configurar medições, calibrar parâmetros e registrar dados, aproveitando a ampla compatibilidade do software com diversos barramentos, notadamente o CAN e o CAN FD, cruciais para o escopo desta pesquisa (ETAS GmbH, 2020).

2.6.2. Vector

A Vector Informatik oferece soluções que se adaptam a diferentes necessidades de aquisição de dados, segmentadas em duas metodologias principais, apesar de compartilharem uma estrutura base similar, como apresentado na Figura 7. A primeira, e predominante, utiliza as Interfaces de Rede Vector para se conectar aos barramentos do veículo, permitindo a comunicação e o monitoramento de redes como CAN (FD), LIN e FlexRay (Vector Informatik GmbH, 2025a). A segunda metodologia, voltada para máxima performance, emprega a família de hardware VX1000, que se conecta diretamente na interface de depuração da ECU para uma extração de dados de baixa latência, utilizando XCP sobre Ethernet para a transmissão ao computador. A plataforma de software que unifica ambas as abordagens é o CANape. Este software atua como a aplicação mestre para as tarefas de medição e calibração, sendo capaz de gerenciar e analisar os dados provenientes tanto de uma interface de barramento padrão quanto do sistema de acesso direto VX1000 (Vector Informatik GmbH, 2025b).

Figura 7 – Visão geral da arquitetura de aquisição da Vector



Fonte: Vector Informatik GmbH (2025b).

2.6.3. Accurate Technologies Inc.

A Accurate Technologies Inc. (ATI) foca sua estratégia na flexibilidade de seu ambiente de software, o VISION. Esta plataforma foi concebida para ser uma solução robusta para aquisição, análise e calibração de dados, tendo como um de seus pilares o suporte universal aos protocolos de aplicação CCP e XCP. Essa conformidade com o padrão permite que o VISION atue como uma aplicação mestre versátil, capaz de se comunicar com qualquer ECU que implemente esses protocolos (Accurate Technologies Inc., 2023). Embora a ATI ofereça suas próprias interfaces de hardware para conexão com as ECUs, como os módulos A8 (Accurate Technologies Inc., 2025a) e A9 (Accurate Technologies Inc., 2025b), a arquitetura do software VISION garante sua interoperabilidade em um ecossistema de hardware mais amplo, reforçando seu caráter agnóstico em relação ao fabricante (Accurate Technologies Inc., 2023).

2.7. TRABALHOS RELACIONADOS

Diversos estudos proporcionam um entendimento importante do CAN FD e seus principais objetivos. Robert Bosch GmbH (2012) introduziu o protocolo, detalhando as melhorias chaves: o aumento da largura de banda máxima e o aumento do tamanho máximo do pacote, tudo enquanto mantém um alto grau de compatibilidade com o hardware CAN existente. Isto foi motivado pela necessidade de suportar ECUs mais complexas e aplicações com mais intensidade de dados, como evidenciado por De Andrade *et al.* (2018) e Dhanush; Ananthakrinshna (2024), que enquadram suas análises na necessidade do CAN FD para superar as limitações do CAN clássico.

Um tema prominente na literatura é a análise quantitativa e experimental de performance do CAN FD contra seu predecessor. De Andrade *et al.* (2018) trazem tanto avaliações experimentais quanto analíticas, concluindo que o CAN FD oferece largura de banda e carga útil significativamente maiores. Similarmente, Dhanush; Ananthakrinshna (2024) conduziram um estudo comparativo focando nas melhorias a nível de protocolo, resultando em um aumento de volume de dados e diminuição de latência. Xu; Cheng; Ruan (2020) confirmaram

esses resultados por meio de análises experimentais, demonstrando que o CAN FD pode alcançar uma largura de banda de até dez vezes a do CAN clássico.

Além de focar apenas na largura de banda e volume de dados, pesquisadores têm focado em análises de performance em tempo real, críticas para redes veiculares. Marcon Zago; Freitas (2018) realizaram um estudo de simulação quantitativo no contexto de veículo agrônomos. Os resultados mostram que a migração do CAN para o CAN FD reduziu significativamente a carga do barramento, tempo de resposta de mensagem, e o *jitter* do sinal, métricas cruciais para sistemas de tempo real. Berisa *et al.* (2023) estenderam essa linha de pesquisa com uma análise comparativa de temporizações, onde afirmaram que o CAN FD supera o CAN clássico para cargas úteis de até 64 bytes.

Finalmente, questões práticas de migrações e considerações de performance no mundo real foram levantadas. Klebe-Klingemann; Webermann (2021) enfatizam que a transição pode ser relativamente simples, muitas vezes aumentando o volume de dados em até oito vezes sem necessitar de mudanças no cabeamento graças à compatibilidade do quadro do CAN FD. Porém, outros estudos alertam que os máximos teóricos nem sempre traduzem diretamente para performance no mundo real. Sinha; Saurabh (2017) notam que enquanto maiores larguras de banda e cargas úteis melhoram o volume de dados, a performance real, a nível de sistema, depende de outros fatores da rede; uma observação compartilhada por Xu; Cheng; Ruan (2020) na questão de topologia de rede.

Em resumo, a literatura existente forma um forte consenso de que o CAN FD apresenta uma significante e necessária evolução do CAN clássico, trazendo mensuráveis melhoras na largura de banda, capacidade da carga útil, aumento do volume de dados e performance em tempo real. A apresentação deste trabalho foca, primariamente, em aumentar o volume de dados por meio dessa melhora a nível de protocolo.

Porém, uma abordagem alternativa ou complementar para contornar o problema da ocupação da rede – adaptar quais os dados são adquiridos e transmitidos de maneira dinâmica – não foi identificada entre os trabalhos correlatos analisados na revisão bibliográfica. Esta dissertação busca também abordar essa lacuna, propondo e avaliando uma nova estratégia para alternar dinamicamente a aquisição de variáveis para otimizar o rendimento da rede.

3. METODOLOGIA

Com base na problemática apresentada, detalha-se neste capítulo a metodologia adotada para atingir o objetivo geral deste trabalho. A abordagem está estruturada em três etapas principais, alinhadas aos objetivos específicos: (i) desenvolvimento de uma ECU emulada para testes de bancada; (ii) migração para o protocolo CAN FD no aparato de testes; e (iii) projeto de um sistema de aquisição em que o conjunto de variáveis adquiridas muda durante a execução.

Estando este trabalho inserido no âmbito do projeto IASE, e sendo a principal motivação do trabalho o limite encontrado durante a utilização do protótipo de aquisição para testes na Renault do Brasil, utilizou-se o hardware e o firmware desenvolvidos dentro do projeto para fins de testes e validações das técnicas aqui apresentadas. O conjunto tem como objetivo adquirir as variáveis da ECU de um veículo de testes proporcionado pela empresa, e foi projetado para utilizar o protocolo XCP sobre CAN (Bedretchuk *et al.*, 2023).

Como fundamentado previamente, o protocolo XCP opera em um modelo mestre-escravo e proporciona uma interface à memória interna da ECU (Schuermans *et al.*, 2003). Em uma visão geral, o trabalho do firmware é ler um arquivo de configuração que determina quais variáveis devem ser adquiridas (Figura 8), configurar as ODTs e listas DAQ corretamente com as informações de cada variável, começar o processo de aquisição e então armazenar todos os dados recebidos, e para isso é necessário uma ECU de bancada ou um veículo de testes (Bedretchuk *et al.*, 2023). Como o presente trabalho consiste em viabilizar maneiras de contornar o limite encontrado nesse processo de aquisição, se tornou necessário o desenvolvimento de uma ECU emulada que também suportasse a implementação dos conceitos apresentados.

Figura 8 – Trecho do arquivo de configuração de variáveis

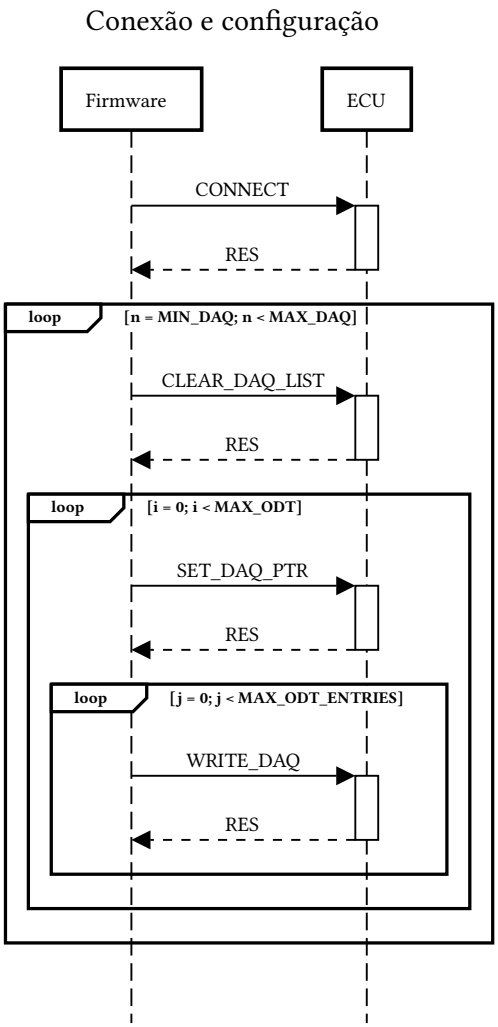
Fonte: Autor (2025).

3.1. EMULADOR DE ECU

Para desenvolver um emulador de ECU, primeiro é necessário entender como funciona o processo de comunicação entre o firmware e a ECU real. No XCP, este processo é padronizado e todas as informações estão disponíveis no documento *"XCP - The Universal Measurement and Calibration Protocol Family, Part 1: Overview"*. Uma visão geral do processo de configuração pode ser observada na Figura 9: o firmware inicializa o canal de comunicação por meio da mensagem *CONNECT* e em seguida configura os endereços de memória. É notável que durante todo esse processo, o trabalho da ECU é apenas responder que os comandos foram efetuados com sucesso (Schuermans *et al.*, 2003).

O procedimento de configuração inicia-se com a reinicialização das estruturas de dados preexistentes, executada pelo comando *CLEAR_DAQ_LIST*. Subsequentemente, o ponteiro de configuração é posicionado no início de cada ODT (*SET_DAQ_PTR*). A etapa final consiste na alocação dos endereços de memória das variáveis de interesse (*WRITE_DAQ*), preenchendo as entradas da tabela sequencialmente até o limite definido pelo parâmetro *MAX_ODT_ENTRIES*, que varia conforme a capacidade de carga do protocolo físico utilizado.

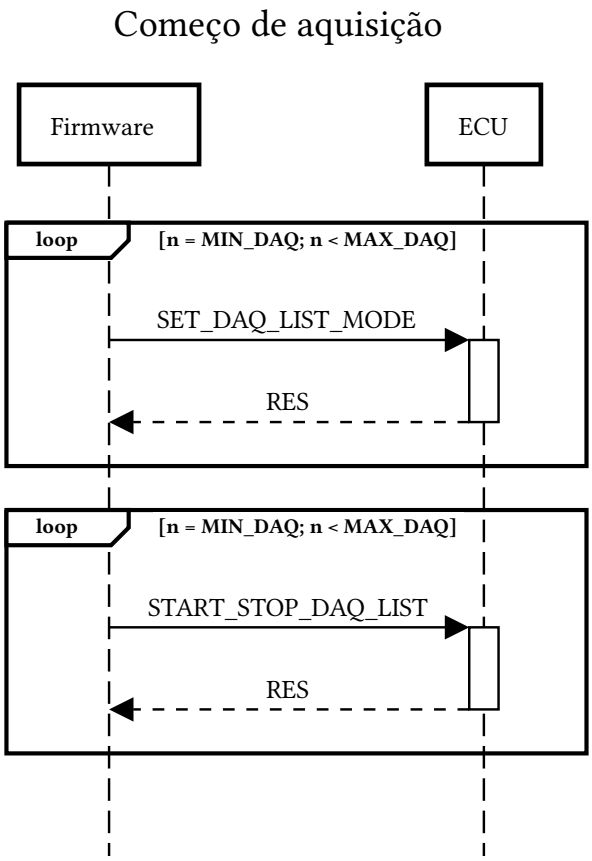
Figura 9 – Sequência de conexão e configuração pelo XCP



Fonte: Autor (2025).

Caso a configuração (variáveis e endereços de memória) seja conhecida tanto para o firmware quanto para a ECU emulada, é possível projetar o emulador ignorando os comandos iniciais, e apenas identificar o momento em que for emitida uma mensagem de *START_STOP_DAQ_LIST* (Figura 10), quando deve então começar a enviar dados para serem adquiridos pelo firmware. Essa abordagem simplifica significativamente o processo de desenvolvimento deste aparato de testes.

Figura 10 – Sequência de início de aquisição pelo XCP

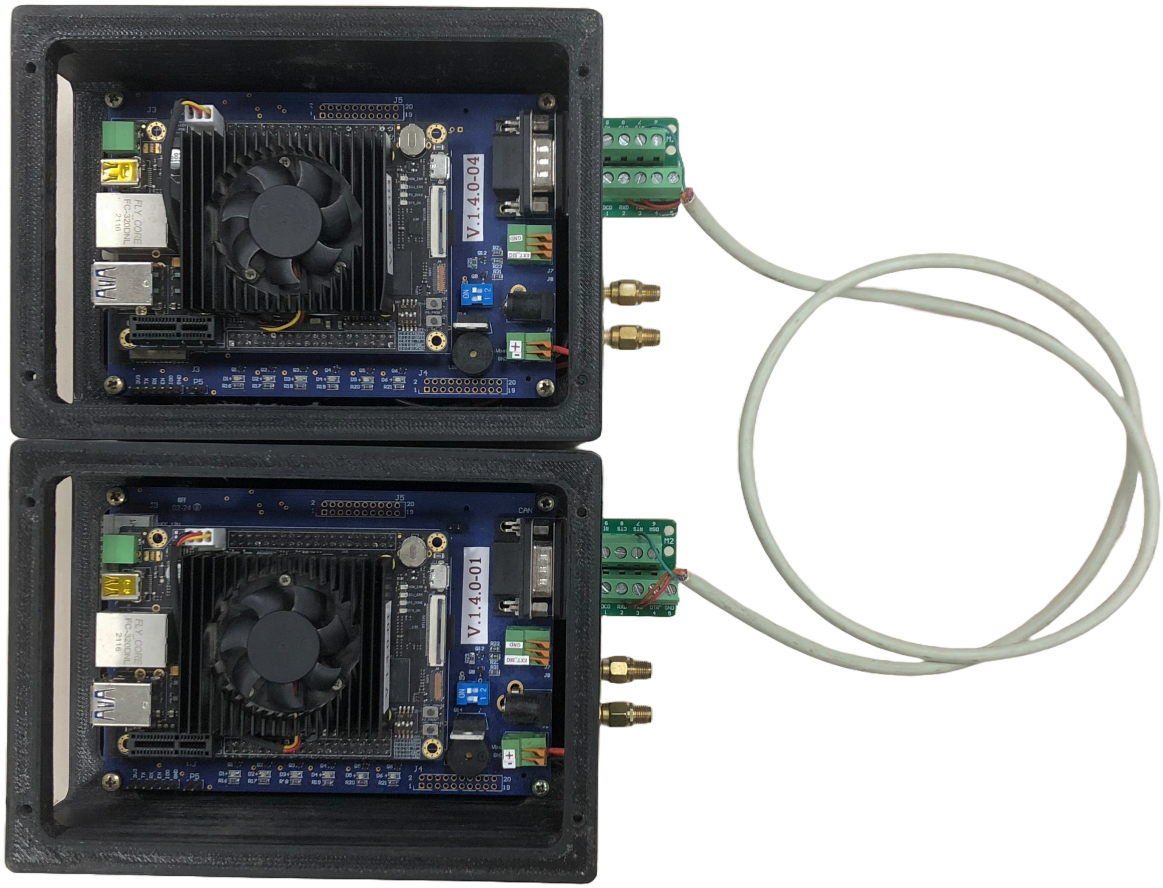


Fonte: Autor (2025).

Para a ECU emulada enviar dados veiculares no barramento, realizou-se o processo contrário à aquisição de dados. Enquanto na aquisição os dados são lidos do barramento e guardados em um arquivo MDF com seu devido *timestamp*, na emulação esses dados são lidos do arquivo MDF e enviados ao barramento no instante indicado por esse *timestamp*. É importante que tanto o firmware quanto a ECU emulada tenham exatamente a mesma configuração de execução, pois a ordem em que os dados são configurados e enviados é importante para a integridade da comunicação.

A fim de validar esse sistema de testes, o barramento CAN de ambas as placas foi conectado (Figura 11), e utilizando um arquivo MDF proveniente de uma rodagem real feita anteriormente no projeto, foi executada uma emulação. Para que o sistema seja caracterizado como funcional, é necessário que o MDF adquirido contenha os mesmos dados que o MDF fonte, com tempos similares.

Figura 11 – Configuração do hardware para o teste de validação



Fonte: Autor (2025).

Uma vez tendo esse aparato desenvolvido, é possível realizar qualquer tipo de modificação, seja de hardware ou de firmware para buscar abordagens que contornem o problema aqui proposto.

3.1.1. Teste de referência para comparação

Antes de implementar as abordagens teorizadas numa tentativa de resolver a problemática, é importante realizar um teste que proporcione uma referência base para calcular as melhorias obtidas com as tentativas. Para isso, será utilizado um experimento real fornecido pela Renault do Brasil que contém 1338 variáveis, que a empresa utiliza no dia a dia para testar e calibrar os carros de produção.

Tanto o firmware quanto o emulador foram configurados para ignorar as variáveis que excedem a capacidade do protocolo CAN, a fim de simular corretamente como seria adquirido o experimento na realidade com essa limitação.

As duas principais diferenças entre o CAN clássico e o CAN FD são a carga útil máxima por mensagem (de 8 para 64 bytes) e a *bitrate* máxima (de 1 para 8 Mbit/s). Com essas informações, é possível primeiro calcular os limites teóricos do CAN e do CAN FD para produzir uma estimativa dos ganhos esperados.

3.2.1. Limites teóricos

No protocolo XCP, a ECU envia os valores das variáveis por meio de ODTs. O tamanho de uma ODT é igual ao do quadro do protocolo que está sendo utilizado. No CAN clássico, esse tamanho é de até 8 bytes, enquanto no CAN FD pode variar entre 1 e 64 bytes. O primeiro byte representa o número identificador da ODT, que pode ser entre 0 e 254 (inclusivo), e os demais bytes são dedicados ao transporte dos dados. O tamanho escolhido impacta diretamente na quantidade máxima de variáveis que podem ser adquiridas em um mesmo experimento. Com essas informações, é possível calcular o limite teórico de aquisição de dados sobre o CAN:

$$255\ \text{ODTs} \times 7\frac{\text{bytes}}{\text{ODT}} = 1785\ \text{bytes} \tag{1}$$

E também sobre o CAN FD:

$$255\ \text{ODTs} \times 63\frac{\text{bytes}}{\text{ODT}} = 16065\ \text{bytes} \tag{2}$$

Note que uma atualização para o CAN FD traz um aumento teórico da capacidade de aquisição de nove vezes.

As listas DAQ comumente encontradas no decorrer do projeto são: 5 ms, 10 ms, 100 ms, Seg1, Seg2 e Seg3; sendo as primeiras periódicas e as últimas sincronizadas com o movimento dos pistões do motor, de tal forma que quando uma revolução é completa, o evento dispara. Considerando o CAN FD, se as 255 ODTs forem divididas de forma aproximadamente uniforme entre essas listas DAQ (separando 43, 43, 43, 42, 42, e 42 ODTs respectivamente), e o pior cenário, em que o motor opera no limitador de rotação for assumido (7500 rpm ou 125 Hz), é possível calcular a taxa de transmissão necessária para adquirir todos os dados. Para essa conta também deve-se levar em conta o byte identificador de cada ODT, para um total de 64 bytes por ODT.

$$\begin{aligned}(43\ \text{ODTs} \times (200\ \text{Hz} + 100\ \text{Hz} + 10\ \text{Hz}) + 3 \times 42\ \text{ODTs} \times 125\ \text{Hz}) \\ \times 64\frac{\text{bytes}}{\text{ODT}} = 14.889\ \text{Mbit/s}\end{aligned} \tag{3}$$

Porém a maior taxa de transmissão que o CAN FD atinge é de 8 Mbit/s, então com esse perfil de aquisição não é possível alcançar o limite teórico de 16065 bytes. Por outro lado, se todas as 255 ODTs forem alocadas para a lista DAQ de 100 ms, é perfeitamente possível alcançar o teto teórico com sobra.

$$255 \text{ ODTs} \times 64 \frac{\text{bytes}}{\text{ODT}} \times 10 \text{ Hz} = 1.306 \text{ Mbit/s} \tag{4}$$

Isso exemplifica que na prática, o limite vai depender do perfil de configuração do experimento à ser rodado; se as variáveis estiverem concentradas nas listas DAQ menos frequentes, será possível adquirir mais variáveis, e vice-versa.

Apesar de serem 255 ODTs no total, todos os A2Ls utilizados no âmbito do projeto mostram outro fator limitante: cada lista DAQ (que representa uma determinada frequência de amostragem) possui como parâmetro de configuração um valor chamado `MAX_ODT`, que dita o teto da quantidade de ODTs que podem ser alocadas para aquela lista. Em todos os casos tratados, os limites eram como na Tabela 2.

Tabela 2 – Limite de ODTs para cada lista DAQ

Lista DAQ	**MAX_ODT**
5 ms	24
10 ms	24
100 ms	24
Seg1	18
Seg2	18
Seg3	18

Fonte: Autor (2025).

Caso fossem mantidos esses limites, a Tabela 3 demonstra o efeito do tamanho do quadro na capacidade de aquisição pela Equação 5.

Tabela 3 – Capacidade de aquisição levando em conta quadros de 8, 16, 32 e 64 bytes

Lista DAQ	**8 bytes**	**16 bytes**	**32 bytes**	**64 bytes**
5 ms	168 bytes	360 bytes	744 bytes	1512 bytes
10 ms	168 bytes	360 bytes	744 bytes	1512 bytes
100 ms	168 bytes	360 bytes	744 bytes	1512 bytes
Seg1	126 bytes	270 bytes	558 bytes	1134 bytes
Seg2	126 bytes	270 bytes	558 bytes	1134 bytes
Seg2	126 bytes	270 bytes	558 bytes	1134 bytes
Total	**882 bytes**	**1890 bytes**	**3906 bytes**	**7938 bytes**

Fonte: Autor (2025).

$$<\text{capacidade}> = <\text{MAX\_ODT}> \times (<\text{tamanho do quadro}> - 1) \quad (5)$$

A banda necessária para cada uma dessas configurações é calculada pela soma das ocupações de banda de cada lista, dada pela Equação 6.

$$<\text{banda}> = <\text{frequência}> \times <\text{capacidade}> \quad (6)$$

Por exemplo, com um quadro de 64 bytes, a lista de 10 ms necessita de uma banda de $100\ \text{Hz} \times 1512\ \text{bytes} = 1.2096\ \text{Mbit/s}$. Lembrando que a frequência considerada para as listas sincronizadas com os cilindros do motor é de 125 Hz, que corresponde ao pior caso de 7500 rpm. Dessa maneira, a Tabela 4 mostra a largura de banda necessária para adquirir todos os dados para cada configuração no caso de um experimento cheio.

Tabela 4 – Largura de banda necessária para adquirir um experimento cheio para cada tamanho de quadro teorizado

Tamanho do quadro	**Largura de banda**
8 bytes	0.795 Mbit/s
16 bytes	1.816 Mbit/s
32 bytes	3.633 Mbit/s
64 bytes	7.265 Mbit/s

Fonte: Autor (2025).

Todas as alternativas de tamanho de quadro são viáveis dentro do limite de largura de banda do CAN FD, que é de 8 Mbit/s.

O experimento em uso na Renault do Brasil, utilizado como referência, não é um experimento balanceado. Ao converter cada variável para seu tamanho em bytes e somar os resultados, agrupando por listas DAQ, é construída a Tabela 5.

Tabela 5 – Distribuição de variáveis do experimento de referência

Lista DAQ	**Tamanho**
5 ms	76 bytes
10 ms	1137 bytes
100 ms	1465 bytes
Seg1	4 bytes

Fonte: Autor (2025).

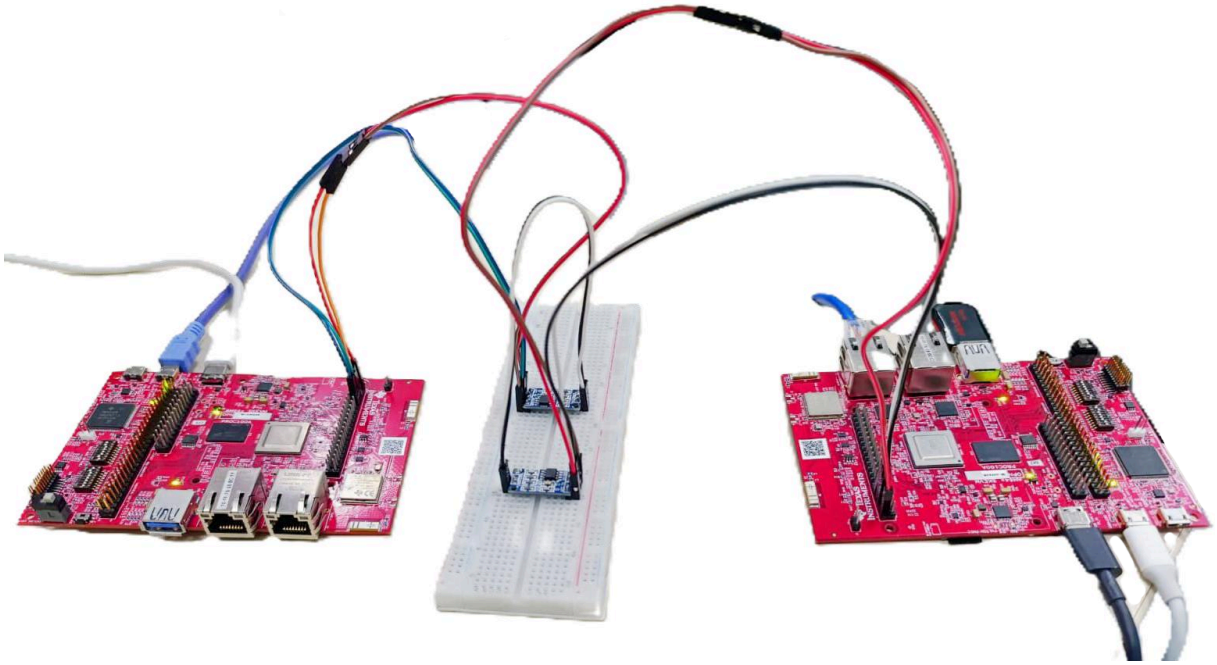
Nota-se a ausência das listas Seg2 e Seg3, e observa-se a maneira como a maioria das variáveis estão concentradas nas listas de 10 e 100 ms. Caso seja utilizado um quadro de 64 bytes, é teoricamente possível adquirir todas essas variáveis nessa exata configuração, e utilizando uma largura de banda de apenas 1.152 Mbit/s, de tal maneira que permite um crescimento futuro desse experimento de até 6.3 vezes.

3.2.2. Implementação

Tendo uma referência e a teoria do que é esperado, foi dada continuidade na implementação do CAN FD. O primeiro passo se deve a uma necessidade de hardware: a placa utilizada no IASE foi projetada para o CAN, logo não possui a capacidade de trabalhar com o CAN FD (Bedretchuk *et al.*, 2023). Ainda aproveitando o âmbito do projeto, foram utilizadas as placas de desenvolvimento do processador que será utilizado na próxima versão: um conjunto de *SK-AM64B Evaluation Boards*.

A configuração de teste pode ser observada na Figura 12, a qual expõe duas placas de desenvolvimento conectadas a dois transceptores CAN TJA1050. Ressalta-se que, embora este componente tenha atendido aos requisitos funcionais deste protótipo de bancada, sua aplicação em um ambiente industrial ou veicular exigiria a substituição por transceptores com suporte nativo e certificado para o CAN FD, a fim de assegurar a integridade do sinal sob as exigências elétricas e de temporização reais dessa tecnologia.

Figura 12 – Configuração do teste com o CAN FD



Fonte: Autor (2025).

O firmware foi adaptado para o novo *upgrade* de protocolo, onde foi necessário lidar com um tamanho de mensagem variável nas funções de envio (Listagem 1) e recebimento de mensagens XCP. A abstração proporcionada pelo *Linux* se encarrega das mudanças no baixo nível; desta maneira, não foi necessário realizar modificações significativas nessa camada. Outra modificação ocorreu na lógica de aquisição de variáveis, que passou a processar não apenas 8 bytes por vez, mas sim um número variável de até 64 bytes. A arquitetura modular do código desenvolvido previamente contribuiu para a facilidade de implementação, já que em diversas áreas a única mudança consiste em trocar o tamanho de um vetor de 8 para 64.

Listagem 1 – Código de envio pertinente ao CAN (acima) e CAN FD (abaixo)

```
auto XCP::send(const std::array<uint8_t, CAN_MAX_DLEN>& message) -> bool {
  struct can_frame can_frame;
  can_frame.can_id = 0x7a0;
  can_frame.can_dlc = CAN_MAX_DLEN;
  std::fill(can_frame.data, can_frame.data + CAN_MAX_DLEN, 0);
  std::copy(message.begin(), message.end(), can_frame.data);
  std::unique_lock lock(can_mutex);
  if (write(can_socket, &can_frame, sizeof(can_frame)) < 0)
    return false;
  return true;
}

auto XCP::send(const std::vector<uint8_t>& message) -> bool {
  assert(message.size() < CANFD_MAX_DLEN);
  struct canfd_frame canfd_frame;
  canfd_frame.can_id = 0x7a0;
  canfd_frame.len = message.size();
  std::copy(message.begin(), message.end(), canfd_frame.data);
  std::unique_lock lock(can_mutex);
  if (write(can_socket, &can_frame, sizeof(can_frame)) < 0)
    return false;
  return true;
}
```

Fonte: Autor (2025).

Nota-se que a mudança no firmware o permite trabalhar com mensagens CAN de um número arbitrário de bytes (contanto que esteja dentro do limite de 64 bytes proveniente do CAN FD). O emulador foi configurado para utilizar ao máximo esse novo limite, minimizando o *overhead* de cada mensagem. A Listagem 2 mostra a lógica de construção das ODTs, onde `XCP::message_length()` retorna 64 no caso do CAN FD. Ao mesmo tempo, o barramento foi configurado para operar em 8 Mbit/s na fase de dados; dessa maneira os benefícios do novo protocolo podem ser aproveitados ao máximo.

Listagem 2 – Código de construção das ODTs por parte do emulador

```
static auto append(std::vector<uint8_t>& buffer, uint8_t& pid, const uint8_t*
data_pointer, size_t data_size) -> void {
  auto remaining = XCP::message_length() - buffer.size();
  if (remaining < data_size) {
    buffer.insert(buffer.end(), data_pointer, data_pointer + remaining);
    XCP::send(buffer);
    buffer = { ++pid };
    buffer.insert(buffer.end(), data_pointer + remaining, data_pointer +
data_size);
  } else {
    buffer.insert(buffer.end(), data_pointer, data_pointer + data_size);
  }
}
```

Fonte: Autor (2025).

Uma vez desenvolvido e adaptado tanto o hardware quanto o firmware, o teste para essa abordagem consiste no mesmo procedimento realizado para obter os dados de referência. Novamente, carregou-se o experimento proveniente da Renault do Brasil com 1338 variáveis realizou-se a emulação, salvando os resultados em um arquivo MDF.

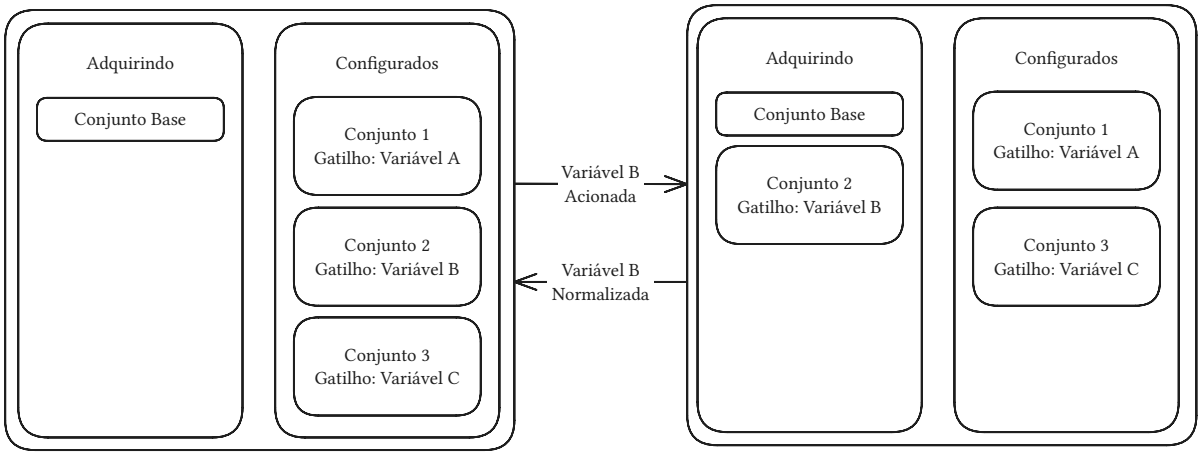
3.3. TROCA DINÂMICA DE VARIÁVEIS

A estratégia de troca dinâmica de variáveis apresenta-se como uma solução de firmware para a otimização da largura de banda, prescindindo de quaisquer alterações na camada física ou no hardware do barramento CAN tradicional. O princípio fundamental desta abordagem consiste na reconfiguração em tempo real das listas DAQs baseada em eventos condicionais. Em vez de saturar o barramento transmitindo todas as variáveis continuamente, o sistema prioriza o contexto momentâneo da operação.

Operacionalmente, define-se um conjunto de monitoramento padrão (base), onde apenas variáveis macroscópicas do veículo são adquiridas. No instante em que uma dessas variáveis exibe um comportamento anômalo ou atinge um limiar crítico, o sistema dispara um gatilho que adiciona à aquisição outro conjunto de variáveis para serem monitoradas. Neste novo contexto, o sistema passa a não só os dados macroscópicos, mas também os dados detalhados específicos do subsistema afetado, permitindo uma análise granular sem comprometer a banda durante o funcionamento normal.

A Figura 13 mostra uma visão genérica desse sistema, configurando 3 conjuntos de variáveis (além do conjunto base), cada um com sua variável de gatilho (que deve ser adquirida pelo conjunto base). Mostra também como um conjunto de variáveis passa a ser adquirido caso sua variável de gatilho seja acionada, e como ele deixa de ser adquirido caso sua variável de gatilho normalize. Cabe ressaltar que caso um número elevado de variáveis de gatilho sejam acionadas simultaneamente, esta configuração pode não ser capaz de adquirir todos os dados definidos no projeto.

Figura 13 – Conjuntos adquiridos e configurados de variáveis e a causa da inclusão/exclusão de conjuntos adquiridos



Fonte: Autor (2025).

Em uma visão prática, a Figura 14 mostra a estrutura de um arquivo de configuração para essa abordagem, que é diferente do arquivo mostrado na Figura 8 na parte de `measurements`, que representa as variáveis a serem adquiridas. A estrutura agora comporta um *array* de mapas, onde cada mapa tem um *array* de variáveis e uma variável de gatilho (*trigger*). Caso a variável de gatilho ultrapasse seu limite mínimo ou máximo (limites que estão nela definidos), ela é considerada acionada e o seu conjunto é selecionado para aquisição.

Figura 14 – Trecho do arquivo de configuração de variáveis para um experimento com troca dinâmica

```
source:
  0:            { name: "Time 100ms", event_fixed: 2, first_pid: 48, … }
  1:            { name: "Seg 1", event_fixed: 4, first_pid: 72, … }
  2:            { name: "Time 5ms", event_fixed: 10, first_pid: 0, … }
  3:            { name: "Seg 2", event_fixed: 6, first_pid: 90, … }
  4:            { name: "Seg 3", event_fixed: 8, first_pid: 108, … }
  5:            { name: "Time 10ms", event_fixed: 0, first_pid: 24, … }
measurements:
  0:
    trigger:    ""
    variables:  (240)[ {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, … ]
  1:
    trigger:    "Vxx_n"
    variables:
      0:        { name: "Vxx_lbup_lean_rich_jug", datatype: "FLOAT32_IEEE", ecu_address: 1342250296, … }
      1:        { name: "Vxx_lbup_lean_rich_jug_add", datatype: "FLOAT32_IEEE", ecu_address: 1342250344, … }
      2:        { name: "Vxx_lbup_pas_ups_rich_diff", datatype: "FLOAT32_IEEE", ecu_address: 1342250192, … }
```

Fonte: Autor (2025).

O aparato de teste é o mesmo de validação da ECU emulada (Figura 11), visto que não há necessidade de mudança de hardware. Apesar de simples no mundo físico, essa abordagem se torna menos direta na implementação em firmware e na etapa de seleção das variáveis a serem adquiridas. No que diz respeito à seleção de variáveis, optou-se por utilizar um conjunto

de sinais comumente empregados em testes automotivos, ainda que não necessariamente configurados de forma ideal para esta abordagem. Essa escolha deve-se ao fato de que o objetivo desta seção é demonstrar a viabilidade e o funcionamento do método, e não propor uma estratégia definitiva de seleção de variáveis para aplicação em campo.

No firmware, foi necessário implementar a seguinte lógica: durante a aquisição de dados, o firmware deve constantemente conferir o valor das variáveis definidas como gatilhos, e caso a condição predeterminada para ativar aquele gatilho seja atingida, acontece a troca de contexto. A troca de contexto consiste em parar a aquisição atual, reconfigurar a ECU para o novo conjunto de variáveis, e então começar uma nova rodada de aquisição.

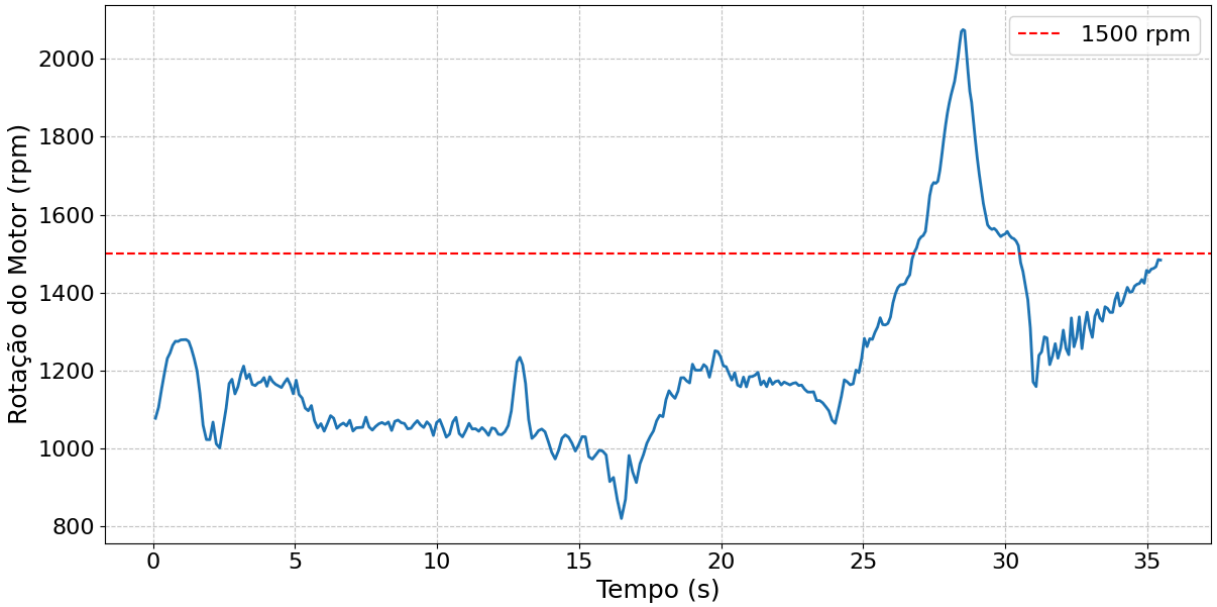
O teste configurado para validar essa abordagem tem como gatilho o valor da variável `Vxx_n` – que representa a velocidade de rotação do motor – quando ultrapassa 1500 rpm. A Listagem 3 mostra a configuração das variáveis, com o foco no gatilho, e a Figura 15 traz o gráfico do valor dessa variável no tempo a partir do MDF fonte, que é usado como base para a ECU emulada enviar os seus dados. Tal figura também demarca o valor de corte, evidenciando que haverá duas trocas de contexto, logo, o resultado deverá ser dividido em três arquivos de dados diferentes.

Listagem 3 – Configuração do experimento de teste

```
// [...]
"measurements": [
  {
    "trigger": "",
    "variables": [
      // [...]
      {
        "name": "Vxx_n",
        // [...]
        "upper_limit": 1500
      },
      // [...]
    ]
  },
  {
    "trigger": "Vxx_n",
    "variables": [
      // [...]
    ]
  }
],
// [...]
```

Fonte: Autor (2025).

Figura 15 – Valor da variável `Vxx_n` no tempo, com uma régua horizontal em 1500 rpm



Fonte: Autor (2025).

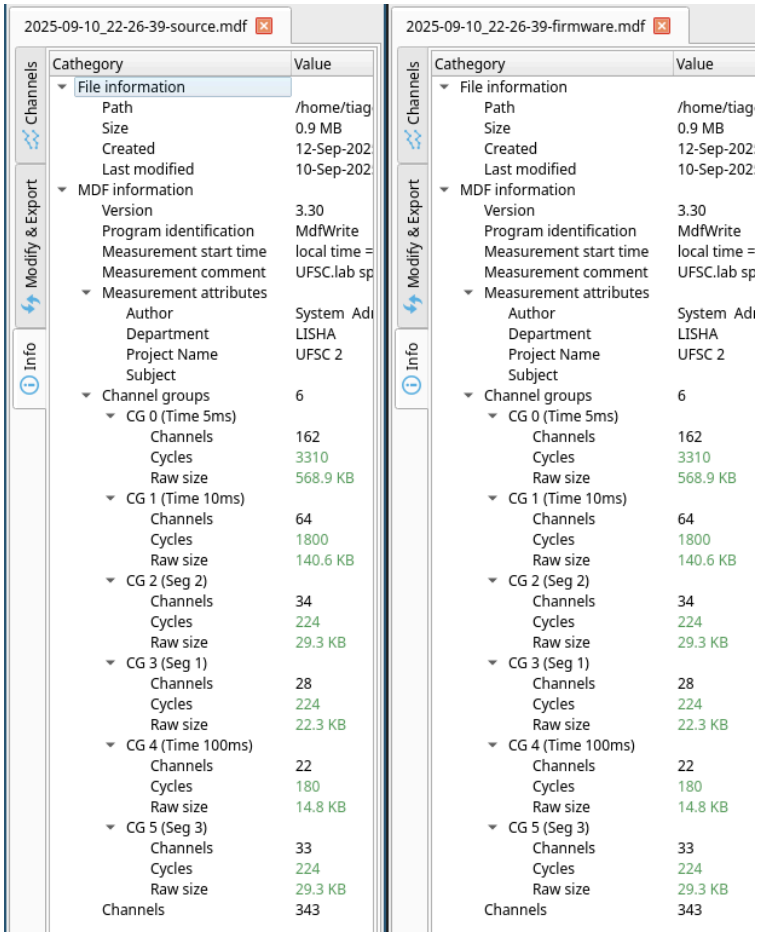
4. RESULTADOS

Este capítulo apresenta e discute os dados obtidos a partir dos cenários experimentais definidos na metodologia. A análise concentra-se na comparação de desempenho entre o sistema de referência (CAN clássico), a implementação do protocolo CAN FD e a estratégia de alocação dinâmica de variáveis.

4.1. EMULADOR DE ECU

No contexto do teste de validação da ECU emulada, o arquivo MDF das aquisições do firmware foi comparado com o arquivo fonte, tanto em estrutura quanto em quantidade de dados, por meio do número de ciclos e tamanho em bytes de cada grupo de dados, utilizando o programa *asammdf*, como observado na Figura 16. Observa-se a quantidade de canais e o número de ciclos, que representam a quantidade de variáveis adquiridas, e a quantidade de amostras, respectivamente. Os números são iguais, de tal maneira a confirmar que o sistema de testes funciona e é capaz de emular uma rodagem tal qual um veículo de testes.

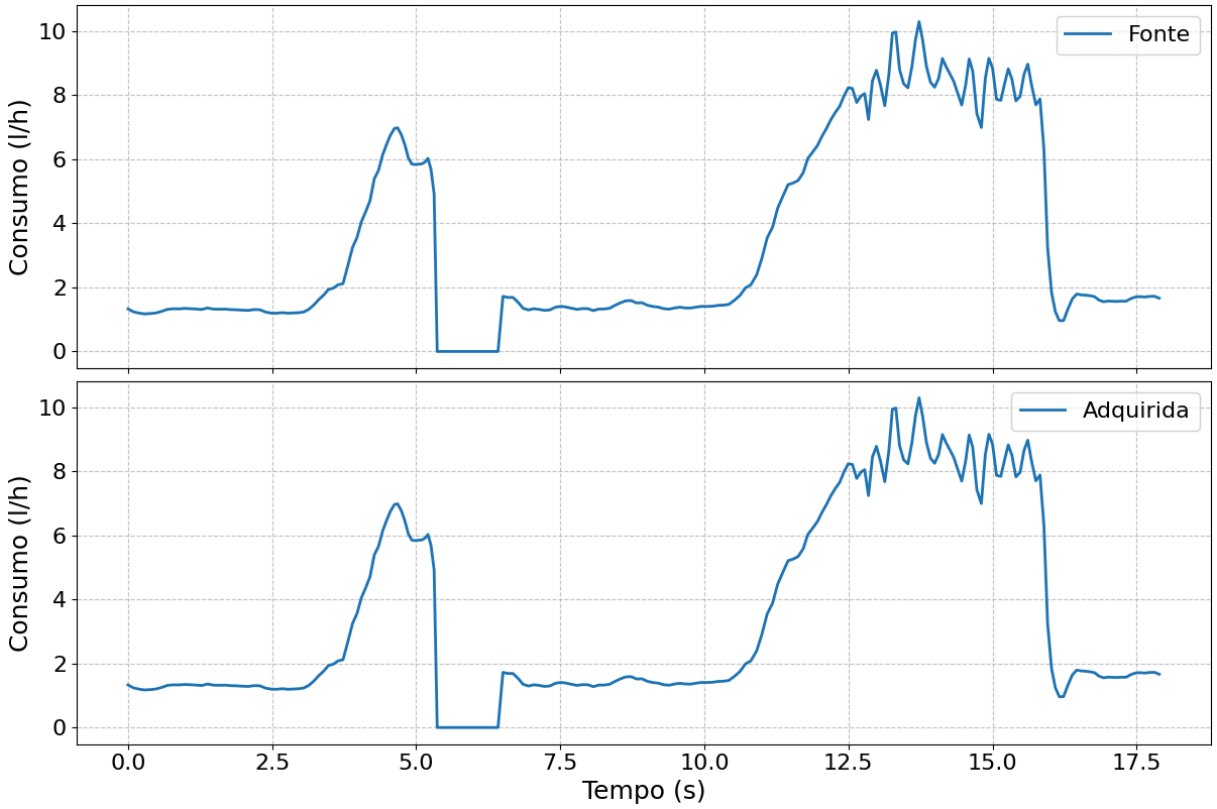
Figura 16 – Comparação das informações do MDF fonte (esquerda) com o adquirido (direita) no teste da ECU emulada



Fonte: Autor (2025).

Além de conferir a quantidade de canais e ciclos, também foi necessário verificar se os dados adquiridos têm o valor esperado e estão formatados corretamente. Para isso, a Figura 17 traz dois gráficos de uma das variáveis do experimento: na esquerda a variável original, e na direita a variável adquirida. Uma análise ponto a ponto mostra que os dados são idênticos, ou seja, pode-se concluir que o experimento foi executado com sucesso.

Figura 17 – Comparação dos dados de uma variável do MDF fonte com os adquiridos no teste da ECU emulada



Fonte: Autor (2025).

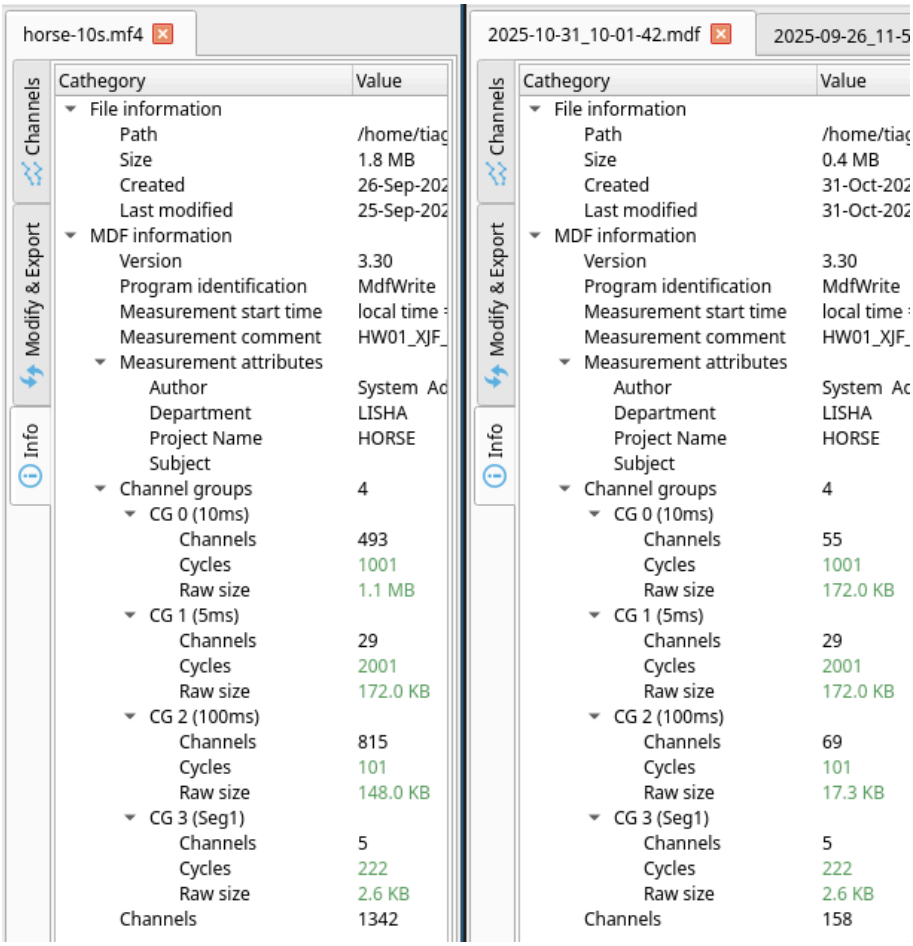
4.1.1. Teste de referência para comparação

A avaliação da eficácia da proposta parte, preliminarmente, da quantificação das limitações impostas pelo sistema legado. Estabelecendo-se uma linha de base sobre o barramento clássico, buscou-se mensurar o gargalo existente antes da aplicação da melhoria.

Nesse contexto, no teste de referência evidencia-se que centenas de variáveis não foram adquiridas devido à saturação da largura de banda do protocolo CAN: enquanto o experimento original demandava 1338 variáveis (1342 canais menos 4 canais de tempo), essa configuração foi capaz de adquirir apenas 154. Nota-se que o volume de variáveis adquiridas é inferior até mesmo ao obtido no teste inicial do emulador. Esse fenômeno justifica-se pela distribuição e tipologia dos dados: o teste anterior contava com variáveis de menor porte (1-2 bytes) distribuídas homogeneamente, enquanto o experimento industrial utiliza majoritariamente variáveis de 4 bytes, concentradas nas listas de alta frequência (10 ms e 100 ms).

A Figura 18 apresenta a tela de informações de ambos os arquivos MDF – o original e o resultante do experimento de referência – na qual se comprova que a quantidade de variáveis efetivamente gravadas foi significativamente menor do que a solicitada. Apesar da perda de variáveis, a contagem de ciclos manteve-se síncrona, indicando que, para o subconjunto de dados que conseguiu ser alocado na banda, não houve descarte de quadros.

Figura 18 – Comparação do MDF fonte (esquerda) com o adquirido (direita) no experimento de referência

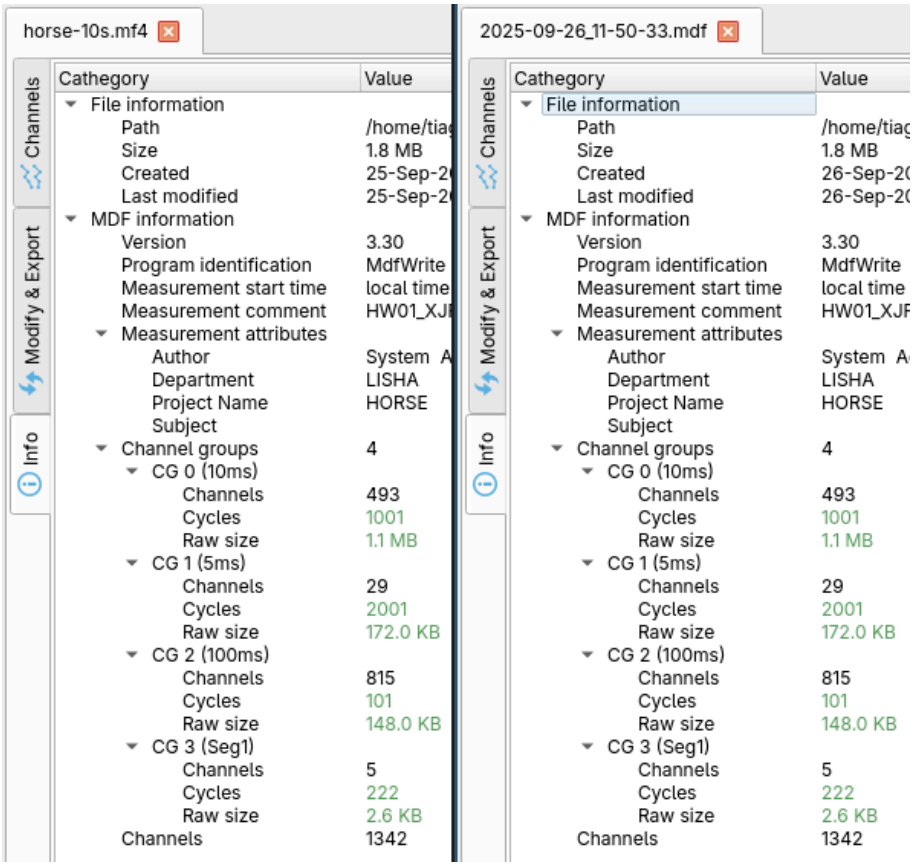


Fonte: Autor (2025).

4.2. CAN FD

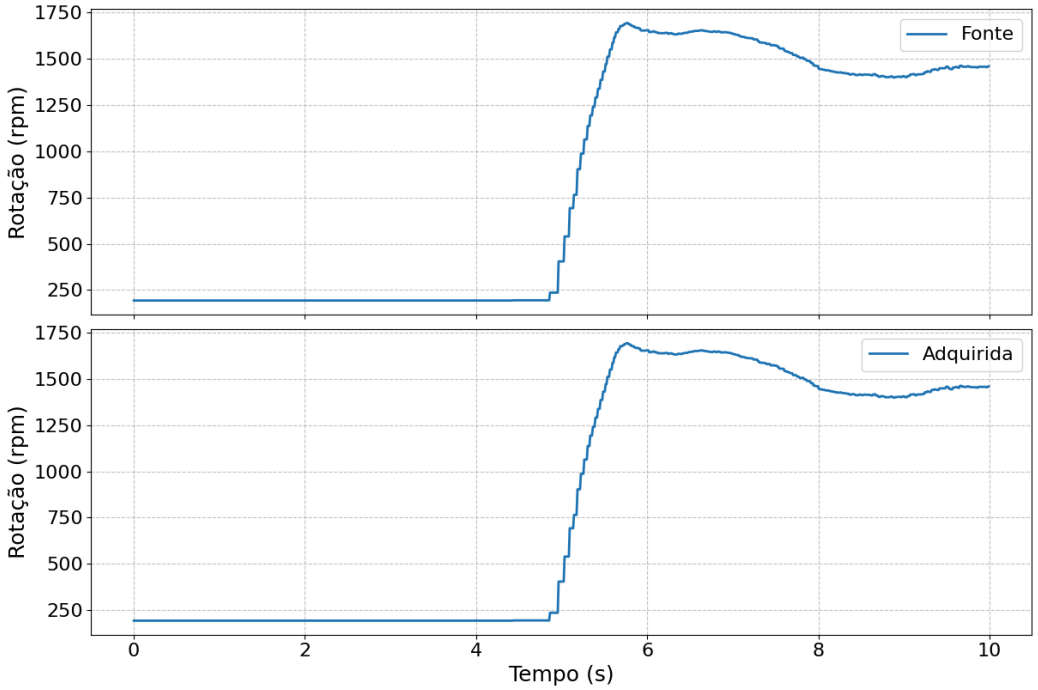
Com a integração do protocolo CAN FD, o procedimento experimental foi replicado mantendo-se os mesmos parâmetros de solicitação do teste de referência. Após a execução, a comparação dos arquivos MDF resultantes (Figura 19) demonstrou a superação do gargalo anterior. Nesta configuração, a totalidade das variáveis foi adquirida com êxito. Simultaneamente, a integridade dos dados foi preservada, visto que a quantidade de ciclos permaneceu idêntica à fonte. A Figura 20 ilustra a comparação gráfica da variável `Vxx_n`, onde observa-se a fidelidade tanto na morfologia do sinal quanto na magnitude dos valores, sendo essa observação confirmada por uma análise ponto a ponto.

Figura 19 – Comparação do MDF fonte (esquerda) com o adquirido (direita) durante o teste do CAN FD



Fonte: Autor (2025).

Figura 20 – Comparação da variável `Vxx_n` fonte com a adquirida durante o teste do CAN FD



Fonte: Autor (2025).

Para complementar a análise lógica do arquivo MDF, realizou-se a inspeção da camada física através da captura de sinais com osciloscópio. A Figura 21 ilustra um dos quadros transferidos durante o processo. Embora o instrumento utilizado não disponha de decodificação nativa para o protocolo CAN FD, a análise visual do envelope do sinal (Figura 22) permite constatar o aumento significativo no tamanho da carga útil (*payload*) e a alteração na taxa de transmissão (*bit rate*) durante a fase de dados, características distintivas em relação ao formato do CAN clássico.

Figura 21 – Captura do osciloscópio no teste do CAN FD

Fonte: Autor (2025).

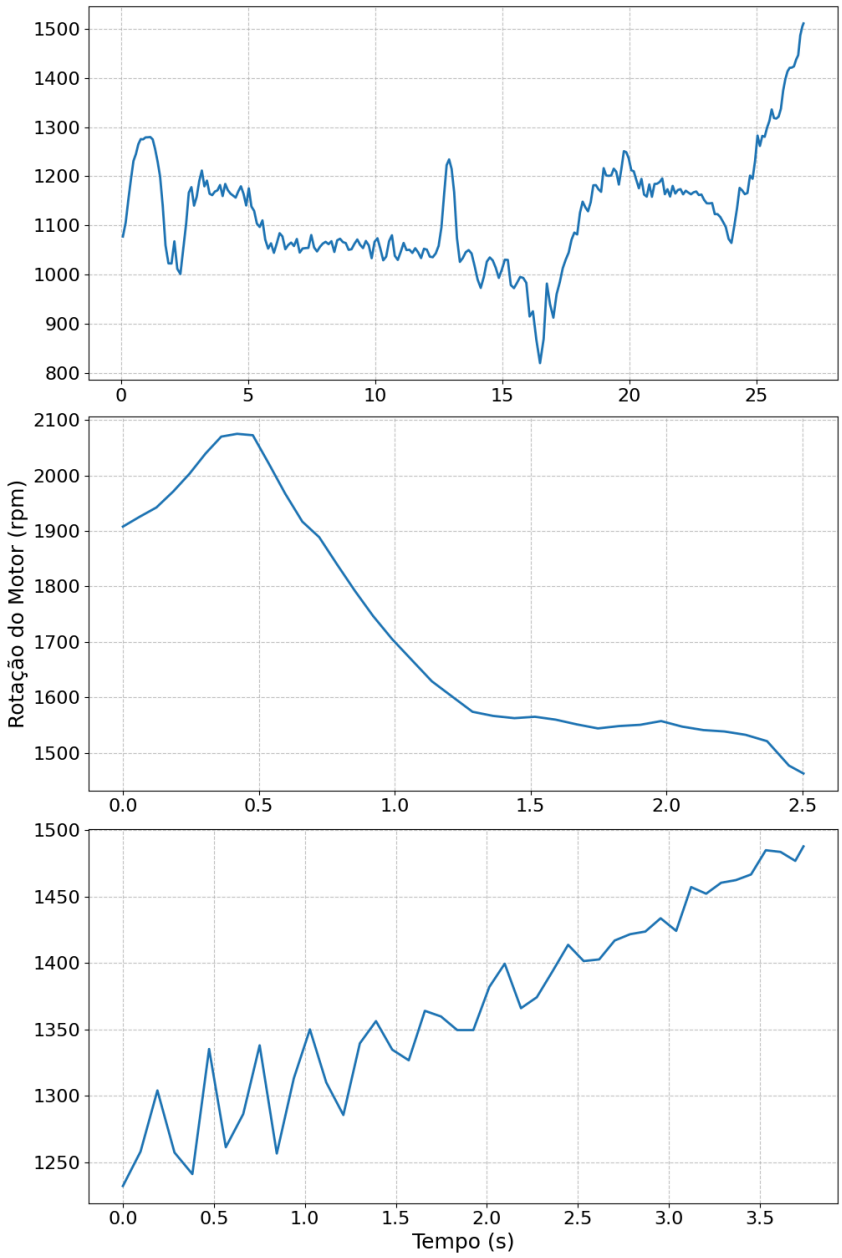
Figura 22 – Captura do osciloscópio de uma mensagem do CAN clássico

Fonte: Autor (2025).

4.3. TROCA DINÂMICA DE VARIÁVEIS

Para validar a eficácia da estratégia de seleção dinâmica em um cenário transiente, analisou-se o comportamento do sistema durante variações de regime do motor, utilizando a rotação (`Vxx_n`) como gatilho de transição. A Figura 23 ilustra a resposta do sistema às variações de rotação: (i) operação abaixo de 1500 rpm; (ii) excursão acima do limiar de 1500 rpm; (iii) retorno à faixa inferior a 1500 rpm. O resultado demonstra que a ferramenta identificou corretamente os eventos de transição exatos. Isso gerou a segmentação automática dos dados em três arquivos independentes (como se fossem rodagens distintas), confirmando que o sistema está apto a realizar a troca de contexto no momento preciso em que a condição lógica é satisfeita.

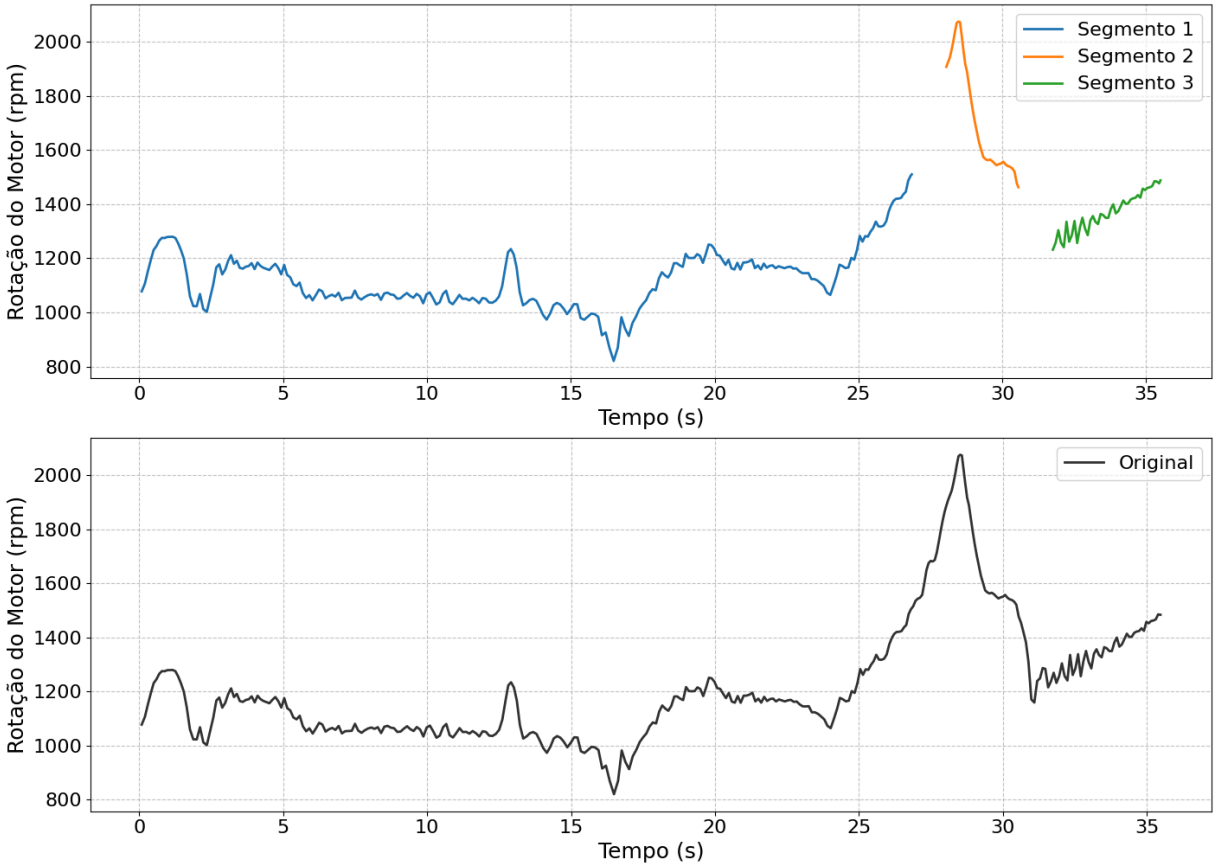
Figura 23 – Valores da variável adquirida `Vxx_n` no tempo, para os três segmentos de aquisição



Fonte: Autor (2025).

Quando concatenados, observa-se na Figura 24 que os dados adquiridos são similares aos dados usados como fonte para o teste, salvo um trecho que é perdido a cada troca de contexto. A figura também mostra a comparação com os dados originais, onde é exemplificado de forma gráfica como essa troca influencia na aquisição de dados, acarretando na perda de dados durante a reconfiguração da ECU para a aquisição de novas variáveis.

Figura 24 – Valores da variável adquirida `Vxx_n` concatenados em um único gráfico e comparados com os dados originais



Fonte: Autor (2025).

Ao observar a saída de texto do firmware, pode-se calcular o tempo gasto em cada troca de contexto, uma métrica importante para analisar a viabilidade do método. A Listagem 4 traz as mensagens do programa, com algumas omissões sinalizadas por `[...]`. Aproximadamente no tempo 11:14:14.907 houve a primeira detecção de um valor acima do gatilho, e aos 11:14:16.086 houve a próxima aquisição dessa mesma variável; logo o tempo de troca de contexto foi de 1.179 s.

Listagem 4 – Saída de texto do firmware durante a troca de contexto

```
[11:14:12.099991] [INFO] [IASE::IASE()]: Running
[11:14:13.099996] [INFO] [IASE::IASE()]: Running
[11:14:14.100004] [INFO] [IASE::IASE()]: Running
[11:14:14.906888] [INFO] [std::optional<IASE::Error> IASE::acquire()]:
Alarmed: Vxx_n (1504.25)
[11:14:14.950752] [INFO] [std::optional<IASE::Error> IASE::acquire()]:
Alarmed: Vxx_n (1510.317138671875)
[11:14:15.100007] [WARN] [IASE::IASE()]: Triggered "Vxx_n"
[...]
[11:14:16.003192] [TRCE] [static bool XCP::start_stop_synch(uint8_t)]: mode=0
[...]
[11:14:16.014964] [TRCE] [static bool XCP::start_stop_synch(uint8_t)]: mode=0
[11:14:16.015004] [TRCE] [static bool XCP::clear_daq_list(uint16_t)]: daq=2
[11:14:16.021613] [TRCE] [static bool XCP::set_daq_list_mode(uint8_t,
uint16_t, uint16_t, uint8_t)]: mode=0
[...]
[11:14:16.050433] [TRCE] [static bool XCP::start_stop_synch(uint8_t)]: mode=1
[11:14:16.054114] [INFO] [IASE::IASE()]: Running
[11:14:16.054122] [TRCE] [std::optional<IASE::Error> IASE::acquire()]:
[11:14:16.085917] [INFO] [std::optional<IASE::Error> IASE::acquire()]:
Alarmed: Vxx_n (1515.25)
[11:14:16.100013] [INFO] [IASE::IASE()]: Running
```

Fonte: Autor (2025).

Este teste serviu como prova de conceito da abordagem, e mostra que é possível trocar as variáveis que estão sendo adquiridas dinamicamente durante uma rodagem, tendo como critério de troca o valor de uma variável de gatilho. Isso permite contornar o limite de largura de banda presente no CAN original sem alteração de hardware ou calibração. No entanto, traz consigo uma dificuldade na elaboração do conjunto de aquisição, perda de dados durante a execução, e a maior limitação é que caso muitos gatilhos sejam acionados simultaneamente, pode não ser possível adquirir todas as variáveis de interesse ao mesmo tempo.

5. CONCLUSÃO

O presente trabalho apresenta a proposição de duas abordagens alternativas para superar as limitações de largura de banda na aquisição de dados via XCP: (i) o *upgrade* para o protocolo CAN FD e (ii) uma configuração dinâmica de variáveis. Para testar essas abordagens, foi necessário primeiramente desenvolver um emulador de ECU.

O objetivo associado ao desenvolvimento de uma ECU emulada foi alcançado de maneira satisfatória, uma vez que o sistema desenvolvido é capaz de emular uma rodagem de um veículo real, sendo necessário apenas um arquivo de dados proveniente de tal rodagem. Esse sistema de emulação permite a realização de testes de maneira fácil, repetível e reproduzível e foi essencial para a elaboração do presente trabalho.

Os testes pertinentes ao objetivo relacionado ao CAN FD revelaram-se um sucesso absoluto: foi possível testar com um experimento utilizado na indústria e ter todas as suas variáveis adquiridas com ampla margem de banda disponível; e, ao mesmo tempo, não houve desvantagem percebida em relação ao CAN original. Essa abordagem requer alterações mínimas de hardware, calibração e firmware e pode proporcionar uma largura de banda de até dez vezes mais (Xu; Cheng; Ruan, 2020).

Por outro lado, apesar dos testes utilizando a abordagem de troca dinâmica também demonstrarem-se bem-sucedidos e não requererem mudança em questões de hardware e calibração, esse conceito trouxe um problema que – aos olhos da indústria – é crucial para a decisão: a necessidade de um projeto de engenharia para definir os conjuntos de aquisição e seus gatilhos. Além de outras desvantagens, como:

- Perda de dados durante, em média, 1.179 s entre as trocas de contexto;
- Caso haja um problema que necessite de muitas variáveis para ser diagnosticado, ou vários problemas devem ser diagnosticado simultaneamente, a limitação pode ser facilmente atingida e o problema original deste trabalho retorna.

Em conclusão, apesar de ambas as abordagens se mostrarem tecnicamente resolutivas para a problemática proposta, a atualização para o CAN FD representa o caminho mais direto, historicamente coerente e fácil de ser implementado.

Com relação à trabalhos futuros, o próximo passo direto consiste na reprogamação de uma ECU para utilizar o novo tamanho de mensagem e a nova taxa de transmissão proporcionados pelo CAN FD, seguida da realização de testes em um veículo real. Outro caminho interessante é analisar o sucessor do CAN FD, o CAN XL. O CAN XL oferece taxas de dados mais altas, de até 20 Mbit/s, e campos de dados maiores (até 2048 bytes) em comparação com o CAN FD, que suporta taxas de dados de até 8 Mbit/s e um máximo de 64 bytes por mensagem (Robert Bosch GmbH, 2023).

REFERÊNCIAS

ACCURATE TECHNOLOGIES INC. **Beyond Automotive Measurement, Calibration & Diagnostics Portfolio Overview**. *[S.l.: S.n.]*.

ACCURATE TECHNOLOGIES INC. **A8 Serial ECU Interface Module**. *[S.l.: S.n.]*. Disponível em: <https://www.accuratetechnologies.com/>.

ACCURATE TECHNOLOGIES INC. **A9 Serial ECU Interface Module**. *[S.l.: S.n.]*. Disponível em: <https://www.accuratetechnologies.com/>.

ASAM E.V. **Format Specification: MDF Format**. , 11 mar. 2014.

ASAM E.V. **ASAM MCD-2 MC (ASAP2 / A2L): Data Model for ECU Measurement and Calibration**. , 30 jan. 2018.

BEDRETCHUK, João Paulo *et al.* Low-Cost Data Acquisition System for Automotive Electronic Control Units. **Sensors**, v. 23, n. 4, 2023.

BERISA, Aldin *et al.* Comparative Evaluation of Various Generations of Controller Area Network Based on Timing Analysis. *In:* 2023.

DE ANDRADE, Ricardo *et al.* Analytical and Experimental Performance Evaluations of CAN-FD Bus. **IEEE Access**, v. 6, , p. 21287–21295, 2018.

DHANUSH, M. S. ; ANANTHAKRINSHNA, T. Enhancing In-Vehicle Networks with CAN--FD: A study of Protocol Improvements over Classical CAN. *In:* 2024.

DI NATALE, Marco *et al.* **Understanding and Using the Controller Area Network Communication Protocol: Theory and Practice**. 1. ed. *[S.l.]:* Springer New York, NY, 2012.

ESPARZA, Orlando; LEICHTFRIED, Wilhelm; GONZALEZ, Fernando G. Transitioning applications from CAN 2 . 0 to CAN FD. *In:* 2017. Disponível em: <https://api.semanticscholar.org/CorpusID:145038212>

ETAS GMBH. **INCA V7.2 Service Pack 17 Release Notes**. Stuttgart, jun. 2020.

ETAS GMBH. **ETAS ES922.1 CAN FD Module (2 Channels) User Guide**. Stuttgart, mar. 2021.

GUI, Yutian; SIDDIQUI, Ali Shuja; SAQIB, Fareena. Hardware Based Root of Trust for Electronic Control Units. *In:* 2018.

ISERMANN, Rolf. **Automotive Control: Modeling and Control of Vehicles**. 1. ed. *[S.l.]:* Springer Berlin, Heidelberg, 2022.

JOSHI, Prachi *et al.* The Multi-Domain Frame Packing Problem in CAN-FD. *In:* 2017.

KLEBE-KLINGEMANN, Renate; WEBERMANN, Hauke. Migration from Classical CAN to CAN FD. **CAN Newsletter**, n. 1, p. 27–29, 2021.

LEEN, Gabriel; HEFFERNAN, D. Expanding automotive electronic systems. **Computer**, v. 35, p. 88–93, 2002.

MARCON ZAGO, Guilherme; FREITAS, Edison Pignaton de. A Quantitative Performance Study on CAN and CAN FD Vehicular Networks. **IEEE Transactions on Industrial Electronics**, v. 65, n. 5, p. 4413–4422, 2018.

PATZER, Andreas; ZAISER, Rainer. **XCP – The Standard Protocol for ECU Development**. , 2016.

POPA, Lucian; BERDICH, Adriana; GROZA, Bogdan. CarTwin—Development of a Digital Twin for a Real-World In-Vehicle CAN Network. **Applied Sciences**, v. 13, n. 1, 2023.

ROBERT BOSCH GMBH. **CAN Specification**. , 1991.

ROBERT BOSCH GMBH. **CAN with Flexible Data-Rate**. , 2012.

ROBERT BOSCH GMBH. **Automotive Handbook**. *[S.l.]:* Wiley, 2022.

ROBERT BOSCH GMBH. **CAN XL – The next step in CAN evolution**. *[S.l.]:* Automotive Electronics, 2023.

SCHUERMANS, Roel *et al.* **XCP - The Universal Measurement and Calibration Protocol Family, Part 1: Overview**. *[S.l.: S.n.]*.

SINHA, Amit Kumar; SAURABH, Shubham. CAN FD: Performance reality. *In:* 2017.

VECTOR INFORMATIK GMBH. **VN15xx - High Performance for CAN (FD) and LIN via PCIe**. , 2025.

VECTOR INFORMATIK GMBH. **VX1000Manual**. Stuttgart, 2025.

XU, Ye; CHENG, Ximing; RUAN, Xusong. Experiment Analysis of CAN-FD Factors. **IOP Conference Series: Earth and Environmental Science**, v. 619, n. 1, p. 12063, dez. 2020.