Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Automotiva

Modelagem e Análise Comparativa de Redes Automotivas CAN e CAN FD com uso do *Software Simulink* e do *Hadware* em Tempo Real da *Speedgoat*

Autor: Lucas Lock Martins
Orientador: Prof. PhD. Evandro Leonardo Silva Teixeira

Brasília, DF
2023

Lucas Lock Martins

Modelagem e Análise Comparativa de Redes Automotivas CAN e CAN FD com uso do *Software Simulink* e do *Hadware* em Tempo Real da *Speedgoat*

Monografia submetida ao curso de graduação em Engenharia Automotiva da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Automotiva .

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Prof. PhD. Evandro Leonardo Silva Teixeira

Brasília, DF

2023

Lucas Lock Martins

Modelagem e Análise Comparativa de Redes Automotivas CAN e CAN FD com uso do *Software Simulink* e do *Hadware* em Tempo Real da *Speedgoat*/ Lucas Lock Martins. – Brasília, DF, 2023-

85 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. PhD. Evandro Leonardo Silva Teixeira

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2023.

1. CAN. 2. CAN FD. I. Prof. PhD. Evandro Leonardo Silva Teixeira. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Modelagem e Análise Comparativa de Redes Automotivas CAN e CAN FD com uso do *Software Simulink* e do *Hadware* em Tempo Real da *Speedgoat*

CDU 02:141:005.6

Lucas Lock Martins

Modelagem e Análise Comparativa de Redes Automotivas CAN e CAN FD com uso do *Software Simulink* e do *Hadware* em Tempo Real da *Speedgoat*

Monografia submetida ao curso de graduação em Engenharia Automotiva da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Automotiva .

Trabalho aprovado. Brasília, DF, 18 de Dezembro de 2023:

Prof. PhD. Evandro Leonardo Silva Teixeira
Orientador

Prof. MSc. Rafael Rodrigues da Silva
Convidado 1

Prof. PhD Renato Vilela Lopes
Convidado 2

Brasília, DF
2023

Eu gostaria de agradecer à mim,
Agradecer à mim por acreditar em mim,
Agradecer à mim por ter realizado todo esse trabalho duro,
Agradecer à mim por não ter dias de denscanço,
Agradecer à mim por nunca desistir,
Agradecer à mim por tomar essas decisões,
Agradecer à mim por tantas noites na biblioteca central da UnB,
Agradecer à mim por nunca desistir dos meus sonhos,
E agradecer à mim por batalhar tanto que,
Hoje eu participo do desenvolvimento de um veículo.

Agradecimentos

Desejo expressar minha mais profunda gratidão aos meus pais, João Américo Pinheiro Martins e Luciana do Nascimento Lock Martins. Eles não apenas me ajudaram e me amaram incondicionalmente, como sempre me apoiaram em todas as minhas escolhas, oferecendo todo o suporte necessário para que eu pudesse perseguir meu sonho. Meus irmãos, Pedro Lock Martins e Bruna Lock Garrido, também estiveram ao meu lado, acompanhando-me em toda essa jornada, não poderia ter mais sorte de ter meus dois alicérces ao meu lado.

Quero dedicar um agradecimento especial à minha namorada, Luísa Sampaio Tavernard. Seu apoio foi constante, desde os tempos do cursinho, antes mesmo de ingressar na faculdade, à minha decisão de morar em outro estado para poder trabalhar na área automotiva até este momento crucial, no final desta etapa da minha vida acadêmica. Obrigado por toda a parceria nos dias e noites de estudos para enfim alcançarmos nossos objetivos.

Não posso deixar de mencionar as pessoas que escolhi para me acompanharem nessa incrível jornada chamada vida: meus amigos, que considero verdadeiros irmãos. Gustavo Giani Barbosa, um amigo de longa data que sempre me incentivou a seguir o meu sonho de me tornar um engenheiro e atuar na indústria automotiva, por todas as risadas, conversas e momentos de descontração que sempre fizeram super bem para meu psicológico. E meu grande amigo Carlos Yan de Souza Gomes, o qual fizemos amizade no meio de nossas graduações, mas após isso sempre esteve ao meu lado nas noites e madrugadas incansáveis de estudos na BCE, além das diversas conversas sobre vagas de emprego, ideias para *start-ups* e muitas vezes apenas como um amigo que está lá para ouvir e te aconselhar.

Aos meus colegas de faculdade que compartilharam comigo essa trajetória desafiadora: Lorrhan Lisboa, Douglas Evaristo, Felipe Miotto, Gustavo Borges, Yan Rodrigues, Gustavo Tavares. Muitos com matérias que compartilhamos ao decorrer do curso e outros pelo simples fato de afinidade que nos aproximou.

Expresso minha sincera gratidão ao meu orientador, Evandro Leonardo da Silva Teixeira, que compartilhou seu vasto conhecimento, não apenas em relação à tese, mas também ao longo da minha graduação em matérias do curso, desempenhando um papel fundamental na minha formação acadêmica.

Gostaria de finalizar agradecendo ao corpo docente do campus Faculdade do Gama por todo o conhecimento que me foi transferido desde meu primeiro dia na faculdade 08/08/2016 até o dia de minha defesa 18/12/2023. Ricardo Fragelli, Vanessa de Castro,

Diogo Garcia, Saleh Khalil, Yevsey Sobolevsky, Tatiane Evangelista, Glauceny Medeiros, Maria Alzira Nunes, Vinicius Rispoli, Osvaldo Kojiro, Lindomar Bomfim, Eneida Gonzalez, Felipe Duerno, Marcelino de Andrade, Maria del Pilar, Luís Filomeno, Suzana Ávila, Gabriela Possa, Emmanuel Pacheco, Rita de Cássia, Flávio Silva, Rejane Figueiredo, Roberto Baptista, Rodrigo Munoz, Felipe Storti, Josiane Aguiar, Edison Cueva, Alessandro Borges, Mateus Miranda, Fábio Alfaia, Carla Anflor, Jorge Cormane, Eberth Correa, André Murilo Pinto, Henrique de Moura, José Felipe Beaklini, Maura Shzu, Mario Benjamin, Armando Caldeira, Rhander Viana, Luciano Noleto, Guilherme Moreira, Eliane de Souza, Patrícia Elisângela, Fábio Lisboa, Sandra Luz, Rafael Silva e Renato Viela, à todos, gratidão! Sou muito grato pelos ensinamentos e perseverança em transmitir conhecimento, uma missão honrosa que merece reconhecimento, muito obrigado.

"A felicidade não é a ausência de conflitos, mas a habilidade de lidar com eles. Uma pessoa feliz não tem o melhor de tudo, ela torna tudo melhor."

(Autor desconhecido)

Resumo

Esta tese centra-se na modelagem e análise comparativa das redes automotivas CAN e CAN FD, utilizando o *software Simulink* e o *hardware Performance Real-Time Target Machine* da *Speedgoat*. A pesquisa visa determinar qual protocolo, CAN ou CAN FD, oferece melhor desempenho em redes veiculares. Inicialmente, foi conduzida uma revisão da literatura para abordar os principais parâmetros de desempenho. Posteriormente, duas redes automotivas foram modeladas no *Simulink*, proporcionando a base para dois estudos de caso. No primeiro estudo, com uma ECU transmissora e uma receptora, a rede CAN FD demonstrou vantagens significativas, apresentando 49,14% menos busload, 34,48% menos latência e 7,57% mais mensagens enviadas em comparação ao CAN. A segunda parte deste estudo, implementada no *hardware Speedgoat*, confirmou a superioridade do CAN FD, revelando 50% menos busload, 27,27% menos latência e 11,36% mais eficiência em relação ao CAN. O segundo estudo de caso, baseado em uma rede automotiva real, destacou ainda mais as vantagens do CAN FD, registrando 73,91% menos busload, 42,68% menos latência e uma notável diferença de 5184 mensagens a menos na rede em comparação ao CAN. A análise comparativa no *hardware* da *Speedgoat* reforçou essas descobertas. Conclui-se que o protocolo CAN FD é mais eficaz para redes automotivas que exigem alta taxa de transferência de dados, especialmente em cenários de transmissão de dados de sensores. Como sugestão para pesquisas futuras, propõe-se a análise do protocolo Ethernet e a exploração de redes híbridas entre CAN e CAN FD. Esses resultados contribuem significativamente para o entendimento e otimização das comunicações em redes automotivas.

Palavras-chaves: CAN, CAN FD, Rede Automotiva, Carregamento do Barramento, Speedgoat

Abstract

This thesis focuses on the modeling and comparative analysis of automotive networks CAN and CAN FD, using Simulink software and Speedgoat's real-time hardware. The research aims to determine which protocol, CAN or CAN FD, exhibits better performance in vehicular networks. Initially, a literature review was conducted to address key performance parameters. Subsequently, two automotive networks were modeled in Simulink, providing the basis for two case studies. In the first study, with a transmitting and receiving Electronic Control Unit (ECU), the CAN FD network demonstrated significant advantages, showing 49.14% less busload, 34.48% less latency, and 7.57% more messages sent compared to CAN. The second part of this study, implemented on Speedgoat hardware, confirmed the superiority of CAN FD, revealing 50% less busload, 27.27% less latency, and 11.36% more efficiency compared to CAN. The second case study, based on a real automotive network, further highlighted the advantages of CAN FD, recording 73.91% less busload, 42.68% less latency, and a notable difference of 5184 messages less in the network compared to CAN. Comparative analysis on Speedgoat hardware reinforced these findings. It is concluded that the CAN FD protocol is more effective for automotive networks requiring a high data transfer rate, especially in scenarios involving massive data transmission. As a suggestion for future research, the analysis of the Ethernet protocol and exploration of hybrid networks between CAN and CAN FD are proposed. These results significantly contribute to the understanding and optimization of communications in automotive networks.

Key-words: CAN, CAN FD, Automotive Network, Busload, Speedgoat

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Segundo Labayrade, Royere e Aubert (2005) 52% dos acidentes na Europa são causados pela colisão entre dois veículos ou de um veículo com algum obstáculo, sendo 90% deles gerados por falha humana. Sete anos depois, Milanés et al. (2012), através de dados da DGT (*Dirección General de Tráfico*) de 2009, informam que 44,3% dos acidentes terminaram em colisão traseira.

Em seguida, Wu et al. (2014) publicam que, segundo a NHTSA (*National Highway Traffic Safety Administration*), 79% dos acidentes que terminaram em colisão traseira foram causados por distração dos motoristas, que poderiam ser reduzidos em 60% caso eles prestassem atenção meio segundo antes. No mesmo artigo, pesquisas da *Benz Company* mostram que, se a iminência à colisão fosse detectada de 1 a 2 segundos antes, e fossem tomadas as medidas corretas, a maioria dos acidentes rodoviários poderiam ter sido evitados.

É evidente que a questão da segurança viária é um desafio global, com a maioria das tragédias ocorrendo nas áreas urbanas, como apontado por Nations (2020) e WHO (2022). No Brasil, os números de acidentes e óbitos no trânsito, conforme os dados do RENAEST (2023) de 2018 a 2021, demonstram a magnitude desse problema, indicados na Tabela 1. Esses acidentes não apenas causam mortes, mas também deixam um rastro de lesões debilitantes, afetando diversas pessoas anualmente.

Tabela 1 – Dados de acidentes no Brasil.

ANO	ACIDENTES	ÓBITOS
2018	646.113	23.946
2019	974.965	24.957
2020	892.758	23.763
2021	991.760	22.857

Fonte: Adaptado de RENAEST (2023)

No entanto, como resposta fundamental para mitigar ou até eliminar os acidentes no trânsito, surgem os sistemas e funções ADAS (*Advanced Driver Assistance System*). Tais funções foram desenvolvidas principalmente para atender a várias questões de segurança e conforto no setor automotivo. Seu objetivo é atuar na falha humana para reduzir o número de acidentes, ferimentos e fatalidades nas estradas. Esses sistemas não apenas ajudam a evitar falhas humanas no trânsito, mas também representam um investimento significativo em recursos financeiros e humanos para aprimorar a tecnologia e reduzir acidentes, tornando as estradas mais seguras para todos.

A história dessas funções, desde os primeiros marcos como o ABS (*Anti-Lock Bra-*

king System) e TCS (*Traction Control System*) na década de 1970 até os sistemas e funções ADAS avançados de hoje, tais como: AEB (*Autonomous Emergency Braking*), BSD (*Blind Spot Detection*), LDW (*Lane Departure Warning*), LKA (*Lane-Keeping Assistance*), entre outros, reflete um esforço contínuo para melhorar a segurança nas rodovias (SPICER et al., 2018). Com isso, pode-se notar que a evolução dos sistemas de controle e das funções ADAS estão em amplo desenvolvimento.

Para que as funções ADAS operem eficazmente, é preciso incorporá-las em um *hardware* dedicado que estabeleça comunicação com outros componentes eletrônicos do veículo. No cerne desses sistemas, as funções ADAS representam uma integração de *software* embarcado nas ECUs (*Eletronic Controll Units*), que estabelecem conexões com os sensores e atuadores. Essa complexa interação de componentes deu origem à necessidade de estabelecer sistemas eficazes que permitam a transferência de informações essenciais entre esses elementos. Nesse contexto, a troca de dados entre a ECU responsável pelos sensores e outras unidades de controle do veículo é de vital importância. Essa comunicação entre os componentes é facilitada por meio de um canal especializado, que, no contexto automotivo, é amplamente conhecido como barramento. Resumidamente, os sensores captam informações do ambiente ao redor do veículo e as transmitem para a ECU conectada a eles, que, por sua vez, utiliza o barramento para compartilhar essas informações com outras ECUs e atuadores. Vale ressaltar que os protocolos de comunicação já eram empregados para gerênciamento de outros subsistemas antes das funções ADAS, tais como motor, transmissão, bateria, etc (BOZDAL; SAMIE; JENNIONS, 2018).

Com isso, Robert Bosch GmbH desenvolveu o protocolo CAN (*Controller Area Network*) e lançou-o em 1986 (SALUNKHE; KAMBLE; JADHAV, 2016). O CAN demonstrou ser altamente vantajoso para o campo automotivo devido à sua alta imunidade a distúrbios elétricos, facilidade no cabeamento, capacidade de detectar erros e repará-los, além de baixa latência entre as mensagens no barramento (BOZDAL; SAMIE; JENNIONS, 2018).

No entanto, à medida que as funções ADAS se tornaram mais complexas e a comunicação entre as ECUs se intensificou, o desenvolvimento de outro protocolo de comunicação para enviar uma quantidade massiva de dados de forma mais rápida as informações se tornou atrativa. Para atender a essa demanda, a empresa Robert Bosch GmbH desenvolveu o protocolo CAN FD (*Controller Area Network with Flexible Data-rate*). O CAN FD mantém a essência do CAN original, mas introduz a capacidade de ajustar a largura de banda, no campo de dados, de acordo com as necessidades específicas de comunicação entre as ECUs (HARTWICH et al., 2012). Essa flexibilidade na largura de banda tornou-se essencial para lidar com a crescente complexidade dos sistemas automotivos e proporcionou melhorias significativas na comunicação entre as unidades de controle. A estrutura detalhada das mensagens será explicada mais afundo no Tópico 2.2.3.

1.1 Objetivos

O objetivo da tese é realizar uma análise comparativa dos protocolos de comunicação automotiva CAN e CAN FD, utilizando modelagem e experimentação prática com o equipamento *Performance Real-Time Target Machine* fornecido pela empresa *Speedgoat*, com o propósito de avaliar e determinar qual dos dois protocolos é mais eficiente para aplicações no setor automotivo, a partir de certas métricas de performance.

Como objetivos específicos tem-se:

- Modelar e analisar as redes automotivas CAN e CAN FD utilizando ferramentas de simulação, como o *Simulink*, para compreender suas características fundamentais.
- Identificar e justificar qual dos protocolos apresenta desempenho superior em termos de métricas específicas, incluindo *busload*, latência e quantidade de mensagens enviadas à rede.
- Planejar e executar dois experimentos práticos para avaliar o desempenho dos protocolos em cenários variados.
- Analisar os resultados obtidos nos experimentos e comparar as métricas de desempenho para determinar qual protocolo é melhor para os cenários abordados.
- Destacar o uso do equipamento da *Speedgoat* como um diferencial da pesquisa, ao fazer a iteração entre modelo e *hardware*.
- Contribuir para o avanço do conhecimento na área de comunicação automotiva, oferecendo uma avaliação abrangente e prática dos protocolos CAN e CAN FD.

1.2 Organização do trabalho

O presente trabalho está disposto em capítulos, em que, cada um possui um objetivo em específico, sendo eles:

- Capítulo 1: Nesse capítulo, encontram-se os objetivos gerais e específicos do trabalho além de, uma breve abordagem histórica do desenvolvimento de embarcados automotivos e dos protocolos de comunicação CAN e CAN-FD, que são usados para manter a comunicação entre as ECUs;
- Capítulo 2: Aqui se encontram uma revisão da bibliografia envolvendo as camadas físicas e de enlace, a sincronização de *bits*, as métricas de análise de desempenho para ambos os protocolos, além de um breve tópico sobre o *hardware Performance Real-Time Target Machine* da *Speedgoat* e o estado da arte para justificar as métricas

avaliadas nesta tese, além de demonstrar as pesquisas que estão sendo realizadas no campo de protocolos de comunicação CAN e CAN FD;

- Capítulo 3: No terceiro capítulo, apresenta-se a metodologia empregada nesta pesquisa, delineando o processo de modelagem das redes, sua integração no *hardware* e as características fundamentais dessas redes. Este segmento detalha minuciosamente a condução do experimento, proporcionando uma visão abrangente de como foi estruturado e implementado;
- Capítulo 4: Nesta seção, são apresentados os resultados obtidos para cada estudo de caso, seguidos por uma análise dos mesmos;
- Capítulo 5: Por fim, esse capítulo é a conclusão do trabalho e o fechamento do mesmo.

2 Fundamentação teórica

Este capítulo inicia com uma análise das características das redes automotivas. Em seguida, contém detalhes das especificações dos protocolos CAN e CAN FD, incluindo aspectos como camada física e de enlace, estrutura das mensagens e tipos de erros, além de uma breve explicação sobre sincronização de *bits*. Seguidamente será discutido as métricas de desempenho, que serão essenciais para explicar a metodologia adotada nos estudos de caso. Por fim, um panorama abrangente das tendências e pesquisas em andamento no campo dos protocolos de comunicação automotiva.

2.1 Redes automotivas

Com o crescimento da industria automotiva e a busca por melhorias de performance, conforto, segurança e mais tecnologia embarcada, a implementação de sistemas eletroeletrônicos complexos aumentou drasticamente, a ponto de alguns carros possuírem mais de 70 unidades de controle (ANDRADE et al., 2013).

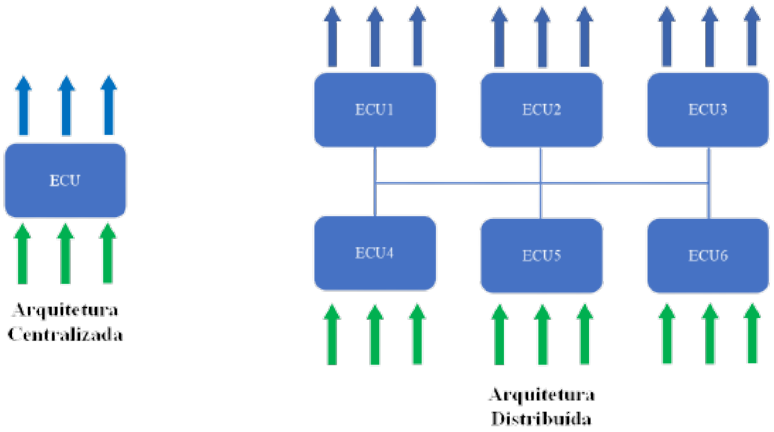
Como supracitado, para haver trocas de informações, entre as ECUs, deve haver um meio de comunicação, que pode ser realizada de forma centralizada ou descentralizada. A grande diferença entre elas se dá pela forma de comunicação, em que, na arquitetura descentralizada as unidades de controle se comunicam de forma independente, sem ter um ponto de controle. Já a centralizada há a necessidade de um *gateway* ou um controlador que administrará a comunicação entre as unidades de controle (DUARTE; FIGUEIREDO, 2023).

No início da década de 1990, as redes automotivas, em sua maioria, eram conectadas diretamente, porém essa arquitetura não se mostra tão eficiente quando comparada à descentralizada, devido ao acréscimo de peso, custo e complexidade, causados pela quantidade de canais de comunicação requeridos. Entretanto, a utilização de ambas as redes em conjunto se mostra mais eficiente, uma vez que, para sistemas de segurança crítica, a arquitetura centralizada se faz necessária, devido a confiabilidade e redundância, enquanto para sistemas menos críticos a descentralizada se mostra mais eficiente (NAVET et al., 2005). Na Figura 1 pode-se ver a diferença entre ambas as arquiteturas.

Além disso, a Figura 2 demonstra que para arquiteturas eletroeletrônicas centralizadas, há uma maior responsabilidade por parte das ECUs. Porém, a complexidade do sistema de controle da rede é menor, enquanto que para arquiteturas descentralizadas tem-se o inverso. Em um sistema de comunicação automotiva centralizado, há um controlador central que gerencia todas as comunicações entre as diferentes ECUs do veículo. Esse con-

trolador central atua como um intermediário, recebendo dados e sinais de várias unidades de controle e, em seguida, roteando-os e distribuindo-os para seus respectivos destinos. Isso resulta em um único ponto de controle e coordenação para todas as comunicações, facilitando a gestão e monitoramento do fluxo de dados. Por outro lado, em sistemas de comunicação descentralizados, a comunicação ocorre diretamente entre as ECUs, sem a necessidade de um controlador central. Cada unidade de controle se comunica independentemente com outras ECUs com base em protocolos de comunicação predefinidos. Isso leva a uma distribuição da responsabilidade de controle da rede, o que permite uma maior flexibilidade e escalabilidade (DUARTE; FIGUEIREDO, 2023).

Figura 1 – Arquitetura centralizada x distribuída.



Fonte: Duarte e Figueiredo (2023)

Figura 2 – Gráfico da responsabilidade das ECUs pela complexidade do sistema de acordo com a arquitetura eletreletrônica.

Fonte: Guimarães (2003)

Neste contexto, as redes de comunicação automotiva são diversas, incluindo protocolos como CAN, CAN FD, MOST (*Media Oriented Systems Transport*), LIN (*Local*

Interconnect Network), *Flexray*, *Ethernet*, entre outros. Cada protocolo possui características distintas que o tornam adequado para diferentes aplicações. O CAN é amplamente utilizado devido à sua confiabilidade e eficiência em aplicações críticas, enquanto o CAN FD oferece maior largura de banda para sistemas mais complexos. O MOST é conhecido por sua alta largura de banda em sistemas de entretenimento de alta qualidade. Já o LIN é um protocolo de baixa velocidade e custo para funções menos críticas, enquanto o *Flexray* é empregado em sistemas de segurança crítica. O *Ethernet*, com sua capacidade de lidar com grandes volumes de dados, está se tornando mais relevante em veículos modernos. Outra característica que os difere está na diversidade nos meios de comunicação, tais como, cabeamento ou *wireless*. Nesta monografia será trabalhado os protocolos CAN 2.0 A, CAN 2.0 B e CAN FD padrão e estendido, todos possuem o barramento cabeado que será melhor explicado no Tópico 2.2 a seguir.

2.2 Protocolo de comunicação CAN e CAN FD

2.2.1 Modelo OSI/ISO

Com o desenvolvimento dos protocolos CAN e CAN FD pela empresa Robert Bosch GmbH a norma ISO 11898, para altas velocidades de transmissão, e a ISO 11519, para baixas velocidades, foram criadas com o intuito de padronizá-los e documentá-los internacionalmente. Com isso, a ISO (*International Organization for Standardization*) criou o modelo OSI (*Open Systems Interconnection*) com sete camadas, em que somente duas são definidas, como pode ser visto na Tabela 2. As demais não são definidas pois são elaboradas de acordo com o projeto (SOUSA, 2002). Pode-se ver também, na Tabela 3, as diferentes documentações para o padrão ISO 11898. Após ambas as tabelas, é possível ver a definição de cada camada do modelo OSI/ISO.

Tabela 2 – Modelo OSI/ISO.

Número da camada	Modelo OSI/ISO	Protocolo CAN
7	Aplicação	Especificada pelo usuário
6	Apresentação	Não definida
5	Sessão	Não definida
4	Transporte	Não definida
3	Rede	Não definida
2	Enlace	Definida pela ISO
1	Física	Definida pela ISO

Fonte: Adaptado de Paret (2007).

Tabela 3 – Descrição dos padrões ISO.

Padrão ISO	Descrição
11898-1	Especifica camada de dados
11898-2	Especifica alta velocidade de taxa de transmissão até 1 Mbps
11898-3	Especifica baixa velocidade de taxa de transmissão de 40 kbps até 125 kbps
11898-4	Especifica *time-triggered*
11898-5	Especifica camada física para taxa de transmissão até 1 Mbps e sistemas que requerem baixo consumo de energia
11898-6	Especifica camada física para taxa de transmissão até 1 Mbps e sistemas que requerem um mecanismo *Wake-up* configurável no *frame* do CAN

Fonte: Adaptado de Andrade (2014)

- **Camada Física**
  De acordo com Paret (2007), a camada física é subdividida em 3 sub-camadas, sendo PLS (*Physical Signalling*), PMD (*Physical Medium Attachment*) e MDI (*Medium-Dependent Interface*). Das três sub-camadas, somente o sinal físico (PLS) é especificado, as outras não, devido a possibilidade de otimizar o meio e a prioridade das mensagens.

- **Camada de Enalce**
  Já a camada de enlace é subdividida em 2 sub-camadas, sendo MAC (*Medium Access Control*) e LLC (*Logical Link Control*).

  – **MAC**
    Esta sub-camada é a parte principal da camada 2, tendo como função apresentar mensagens recebidas e aceitar a transmissão dos *frames* para a sub-camada LLC. Aqui também é realizada a diferenciação entre erro temporário e erro permanente.

  – **LLC**
    Apresenta a lógica da comunicação, uma vez definida não há a necessidade de alterar para nenhum protocolo.

- **Camada de Rede**
  A Camada de Rede assume o papel de rotear pacotes de dados entre redes e de-

cidir o caminho ótimo para a transmissão de dados, usando endereçamento lógico (endereços IP).

- **Camada de Transporte**
  A Camada de Transporte assegura a confiabilidade da comunicação de ponta a ponta e a integridade dos dados, segmentando e reagrupando dados, gerenciando o controle de fluxo e fornecendo correção de erros, juntamente com o endereçamento de portas por meio de números de porta.

- **Camada de Sessão**
  À medida que ascendemos à Camada de Sessão, sua função reside em estabelecer, gerenciar e encerrar sessões ou conexões entre aplicativos, supervisionando o controle de sessão e a sincronização.

- **Camada de Apresentação**
  A Camada de Apresentação concentra-se na tradução de dados, criptografia, compressão e formatação, garantindo que os dados sejam apresentados em um formato compreensível.

- **Camada de Aplicação**
  Finalmente, a Camada de Aplicação interage diretamente com as aplicações e fornece serviços de rede aos usuários finais, acomodando vários protocolos de nível de aplicação e fornecendo interfaces de usuário para acesso aos serviços de rede.

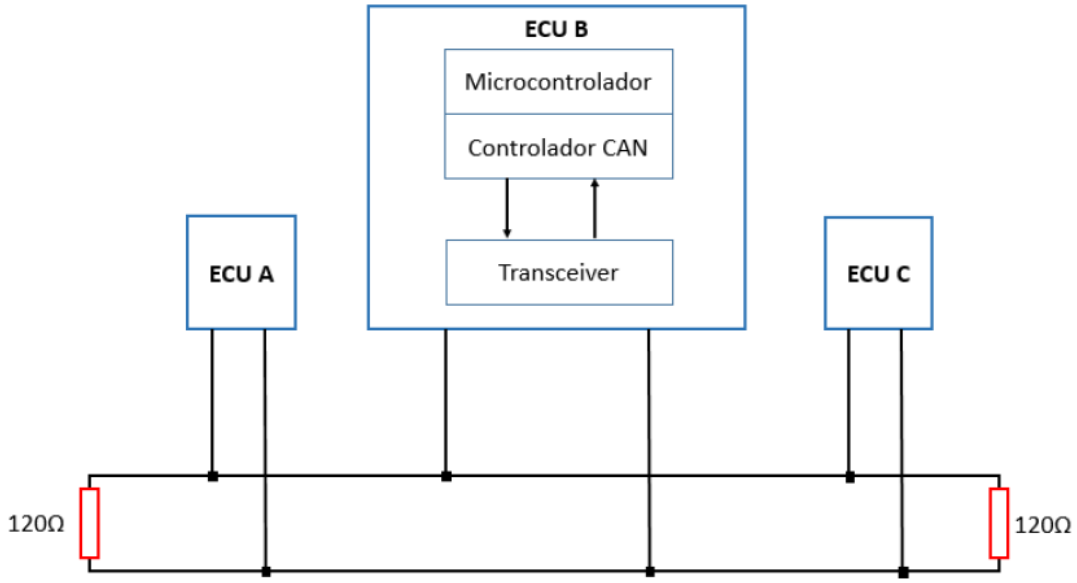
2.2.2 Camada física

No que diz respeito à camada física, tanto o CAN quanto o CAN FD compartilham o mesmo barramento, além disso, ambos os protocolos são compatíveis conforme estabelecido pela norma ISO 11898-1, o que possibilita uma rede híbrida entre CAN e CAN FD. Isso significa que as ECUs podem se comunicar entre si na mesma rede, desde que as mensagens enviadas pelo protocolo CAN FD não excedam 8 *bytes*, devido as limitações do protocolo CAN. Essa compatibilidade é fundamental, permitindo que sistemas que utilizam o protocolo CAN convencional e sistemas que adotam o CAN FD interajam de forma harmoniosa, facilitando a integração de diferentes partes de um sistema (ANDRADE, 2014). A representação de um barramento genérico pode ser exemplificada pela Figura 3, em que se tem três ECUs conectadas em paralelo por um par de cabos trançados, para reduzir interferências eletromagnéticas, suportar maiores distâncias, minimizar ruídos de sinal, ser economicamente viável e por ser um padrão amplamente aceito na indústria. Esses cabos garantem que a comunicações seja confiável em ambientes automotivos e industriais, onde a integridade dos dados é fundamental (PARET, 2007). Para

finalizar a descrição de um barramento genérico, tem-se um resistor de 120Ω em cada extremidade, para garantir a deflexão do sinal no barramento, inibindo erros ou falhas na leitura do mesmo (GUIMARÃES, 2007).

Como pode ser visto na Figura 3, a ECU possui ao menos um *transceiver*, um controlador CAN e um microcontrolador/microprocessador. Assim que uma mensagem é enviada ao barramento, o *transciever* converte o sinal de tensão da linha CAN/CAN FD para os níveis de tensão do controlador, que funciona como interface entre o comando e o barramento, no qual formata os dados recebidos para ser enviado à linha (BORTH, 2016).

Figura 3 – Barramento padrão de acordo com a ISO 11898-2.

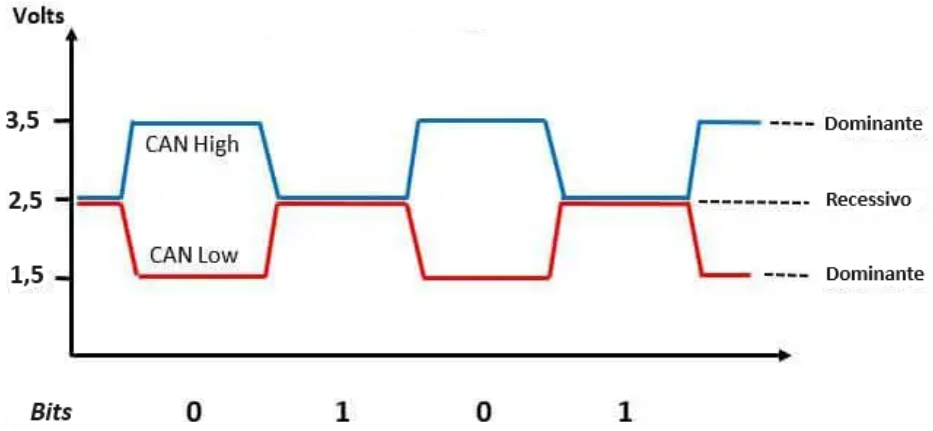


Fonte: Borth (2016)

Visto isso, tem-se as mensagens que percorrem o barramento, em que, ambos os protocolos possuem o mesmo nível lógico como citado no Tópico 2.2.1. Portanto, os *frames* são enviados ao barramento no formato de *bits* que operam em dois níveis lógicos, sendo eles, recessivo (*bit* lógico 1) ou dominante (*bit* lógico 0). Observa-se na Figura 4 que o *bit* recessivo possui a diferença de pontencial (DDP) igual a zero, ambos os cabos estão com tensão de 2,5 volts. Para o *bit* dominante, a DDP é de 2,0 volts, em que o cabo conhecido como *CAN HIGH*, demonstrado pela cor azul, possui a tensão de 3,5 volts e o cabo *CAN LOW*, na cor vermelha, possui a tensão de 1,5 volts (SILVA, 2015).

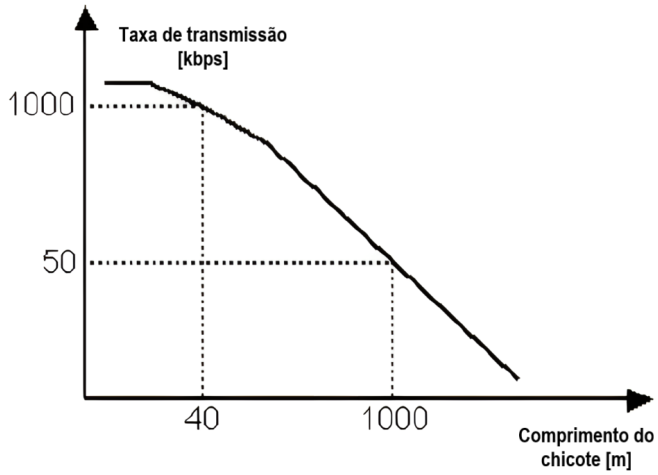
A ISO 11898 também especifica a taxa de transmissão de *bits* pelo comprimento do chicote. Para taxas de transmissão de 1 Mbps deve-se ter no máximo 40 metros de cabo. Para 40 kbps tem-se comprimento máximo de 1000 metros de chicote (BORTH, 2016). A relação pode ser vista na Figura 5.

Figura 4 – Níveis de tensão para os protocolos CAN e CAN FD.



Fonte: Adaptado de Silva (2019)

Figura 5 – Relação da taxa de transmissão pelo comprimento do chicote.



Fonte: Adaptado de Guimarães (2003)

2.2.3 Camada de enlace

A camada de enlace é responsável por encapsular e desencapsular as mensagens que transitam o barramento, definir qual mensagem irá ser transmitida em caso de dois ou mais *frames* serem enviados simultaneamente, além de detectar erros de transmissão e informar a origem do mesmo (SILVA, 2015). Com isso, serão abordados neste tópico os tipos de *frames*, o formato das mensagens de dados, os métodos de detecção de erros e os métodos de arbitragem para ambos os protocolos. Será abordado tópico a tópico iniciando pelo protocolo CAN e em sequência abordando o CAN FD para todos.

Tem-se que, as mensagens dos protocolos CAN e CAN FD são enviadas em formato de *frames*, em que há a leitura *bit* a *bit* por cada ECU, inclusive a própria transmissora. As mensagens podem ser de quatro tipos, segundo Silva (2015):

- ***Frame* de dados**
  A ECU está enviando uma mensagem ao barramento a respeito de alguma informação, como por exemplo a velocidade de rotação do motor;

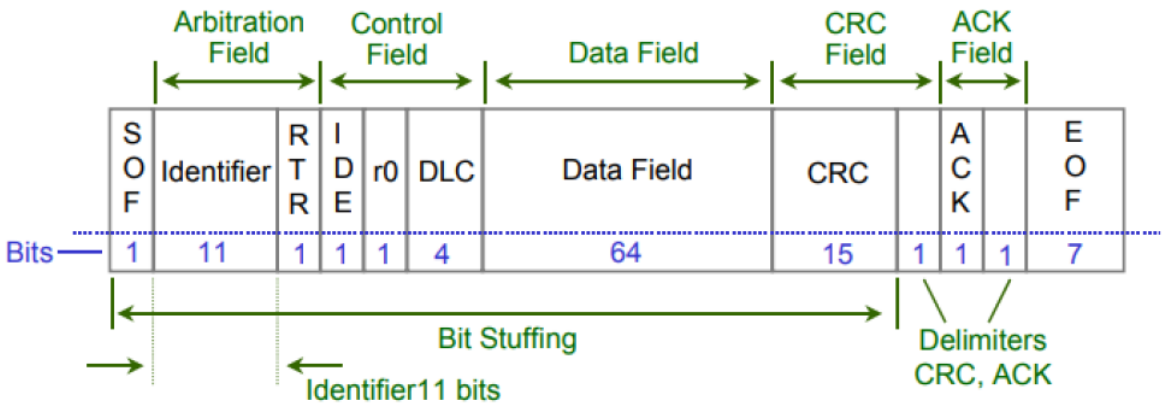
- ***Frame* de requisição**
  É a mensagem enviada ao barramento solicitando que uma informação seja enviada, por exemplo, a ECU do motor solicitando a posição do pedal do acelerador para saber quanto de gasolina deverá ser injetada;

- ***Frame* de erro**
  Quando um erro é detectado e a ECU envia uma mensagem de erro no barramento.

- ***Frame* de sobrecarga**
  Mensagem enviada ao barramento para que nenhum outro *frame* acesse-o, o que dá tempo para a ECU transmissora processar as informações recebidas.

Em sequência, tem-se a estrutura das mensagens. Na Figura 6 está demonstrado como é a estrutura do *frame* de dados para o protocolo CAN 2.0 A, a estrutura padrão do protocolo. A explicação de cada parte do *frame* está baseado em Paret (2007).

Figura 6 – Estrutura do frame de dados para o CAN.



Fonte: Duarte e Figueiredo (2023)

- **SOF – *Start Of Frame***
  O SOF contém 1 *bit* dominante o que indica o início da transmissão de uma mensagem ao barramento, caso o mesmo esteja inoperante no momento.

- ***Identifier***
  O campo *identifier* tem como função identificar quem está mandando a mensagem e para quem a mensagem é destinada, além de definir a prioridade. Possui 11 *bits*, em que, os sete primeiros *bits* não podem ser recessivos, pois indica o fim da transmissão do *frame*.

- **RTR – *Remote Transmition Request***
  O RTR indica se o *frame* é de dados, *bit* dominante, ou de requisição, *bit* recessivo.

- **IDE – *Identifier Extended Bit***
  Composto por 1 *bit*, serve para identificar qual o formato que será utilizado, sendo o *bit* dominante para o protocolo 2.0 A e recessivo para o 2.0B.

- **r0**
  Também possui 1 *bit* que, para o formato 2.0 A este *bit* é dominante, porém o mesmo é reservado para futuras melhorias.

- **DLC – *Data Length Code***
  Tem como finalidade indicar quantos *bits* serão enviados no campo de dados.

- ***Data field***
  Pode possui até 64 *bits* ou 8 *bytes*, em que carrega a mensagem a ser transmitida para o barramento com o MSB (*Most Significant Bit*) na frente.

- **CRC – *Cyclic Redundancy Check***
  Possui 15 *bits*, é uma parte da mensagem em que todas as receptoras devem ler para verificar se há erros de transmissão de mensagem.

- **CRC *delimiter***
  Somente 1 *bit* recessivo para indicar o término do CRC.

- **ACK *slot* – *Acknowledgement slot***
  Possui 1 *bit* que se transforma de recessivo para dominante, caso nenhum receptor envie uma mensagem de erro ao barramento, em outras palavras, se não houver erro de transmissão, o *bit* é dominante.

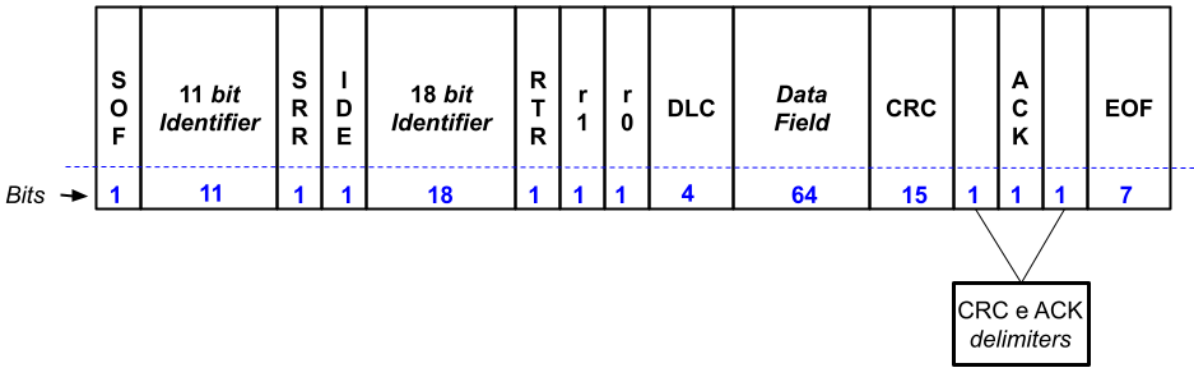
- **ACK *delimiter* – *Acknowledgement delimiter***
  Possui 1 *bit* recessivo para indicar o fim do *acknowledgement field*.

- **EOF – *End Of Frame***
  São 7 *bits* recessivos para indicar o fim da transmissão do *frame*.

Já o protocolo CAN 2.0 B possui uma pequena diferença em sua estrutura com relação ao CAN padrão, sendo ela o campo de arbitragem e de controle. A quantidade de *bits* que podem ser enviados na mensagem é aumentada devido a esta mudança. A diferença no formato do *frame* padrão para o estendido pode ser visto na Figura 7

Percebe-se que a diferença da Figura 6 para a Figura 7 se encontra nos *bits* "SRR", "IDE", "18 *bit identifier*" e "r1". As mudanças descritas abaixo para este protocolo estão de acordo com Paret (2007) e Bosch (1991):

Figura 7 – Estrutura do frame de dados para o CAN estendido.



Fonte: elaborado pelo autor

- **SRR – *Substitute Remote Request***
  Consiste em 1 *bit* recessivo para informar ao barramento que uma mensagem no formato estendido está sendo enviada. Em comparação com o protocolo CAN 2.0 A, que possuía nesta posição o "RTR", esse campo por ser preenchido com um *bit* recessivo sempre terá o *frame* do protocolo padrão lido, ao invés do estendido em caso de conflito de mensagens.

- **IDE – *Identifier Extended Bit***
  Aqui a definição é a mesma do supracitado no formato padrão, porém a sua posição muda, agora o mesmo se encontra no campo de arbitragem, ao invés do campo de controle.

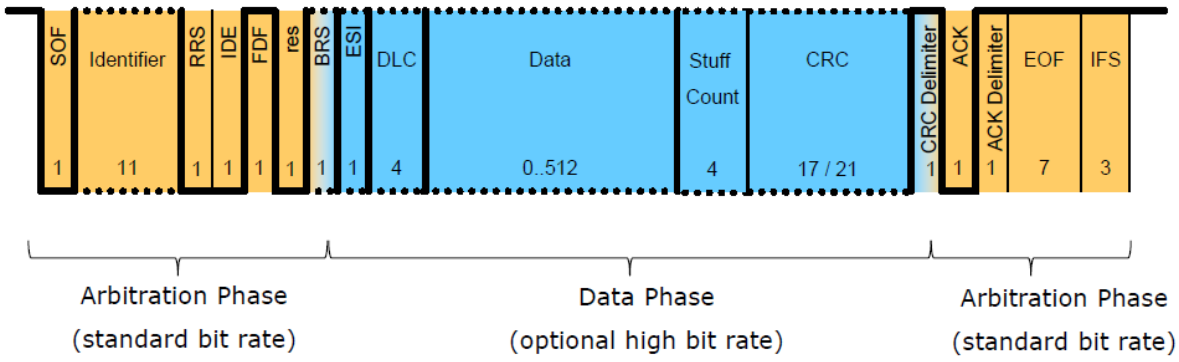
- **18 *bit identifier***
  Tem a mesma função do *identifier* citado acima, porém aqui pode possuir mais identificadores em um único *frame*.

- **r1**
  Possui a mesma definição do "r0" supracitado.

A grande diferença entre ambos os protocolos encontra-se no número de identificadores que podem receber a mensagem. Para o formato padrão, tem-se aproximadamente 2048 identificadores, enquanto no formato extendido pode-se obter mais que 536 milhões. Ademais, a estrutura do *frame* do protocolo CAN FD padrão pode ser visualizada na Figura 8. Pode-se reparar na similaridade entre ambos os *frames*, logo, será explicado somente os *bits* que não foram previamente elucidados no protocolo CAN. De acordo com Vector (2013), somente três *bits* estão a mais na estrutura da mensagem do protocolo CAN FD em comparação com o CAN, são eles:

- **FDF – *FD Format***
  Como ambos os protocolos podem trocar informações entre si, este *bit* está destinado

Figura 8 – Estrutura do frame de dados para o CAN FD padrão.



Fonte: Vector (2013)

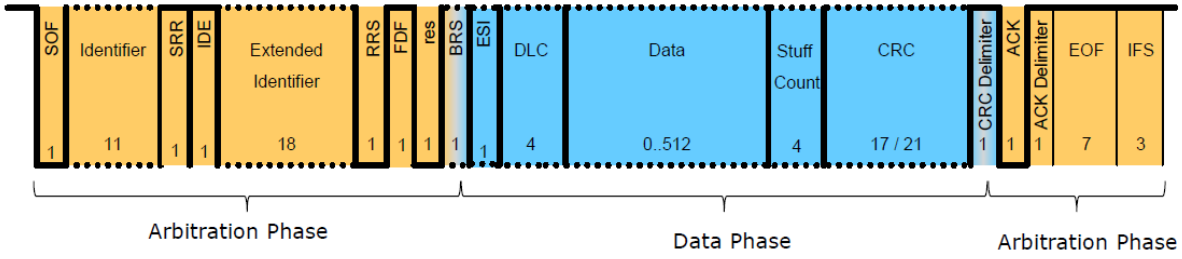
a diferenciar se a mensagem será do protocolo CAN (*bit* dominante) ou CAN FD (*bit* recessivo);

- **BRS – *Bit-Rate Switch***
  Esse *bit* tem como finalidade definir se o campo de dados será enviado na velocidade predefinida pelo *baud rate prescaler* ou se será enviado mais rápido, logo, *bit* dominante indica que a mensagem será enviada em velocidade constante e, no caso de *bit* recesivo há a alteração na taxa de transmissão do campo de dados;

- **ESI – *Error State Indicator***
  É o estado da ECU, se ela se encontra em erro passivo ou erro ativo. O *bus-off* não é sinalizado, pois uma vez neste estado, a ECU não transmite mais nenhuma mensagem, somente executa a leitura dos *frames* no barramento.

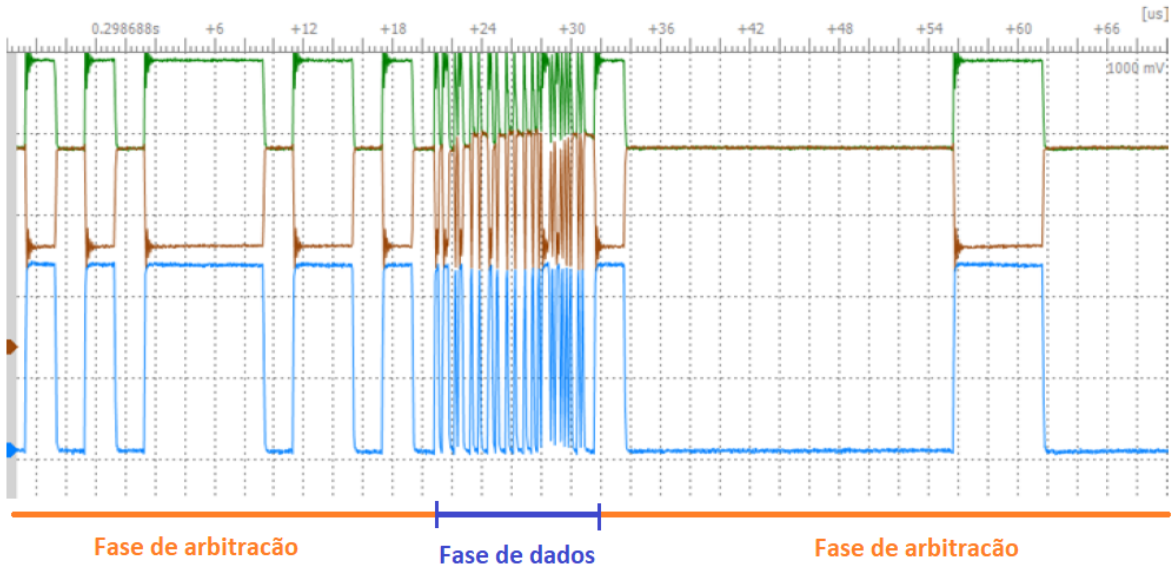
Em seguida, tem-se na Figura 9 a estrutura do *frame* do CAN FD estendido. Todos os *bits slots* foram previamente elucidados. Segundo a Vector (2013), a parte em laranja (*arbitration phase*) nas Figuras 8 e 9 possuem a mesma taxa de transmissão que o protocolo CAN, ou seja, está de acordo com o *baud rate prescaler* da rede, enquanto na parte azul (*data phase*), a velocidade de transmissão pode variar para até 5 Mbps. A Figura 10 demonstra a diferença na taxa de transmissão do campo de dados com relação ao de arbitragem. Nota-se que a parte sinalizada de azul possui a frequência maior que das partes marcadas de laranja, isso ocorre pois o protocolo CAN FD possui um *bit slot* em seu *frame* que define se o campo de dados será diferente do campo de arbitração, o "BRS". Logo abaixo tem-se a Figura 11 que representa, de forma mais intuitiva, o que acontece com o comprimento do *frame* (na unidade do tempo) para o protocolo CAN FD.

Figura 9 – Estrutura do frame de dados para o CAN FD estendido.



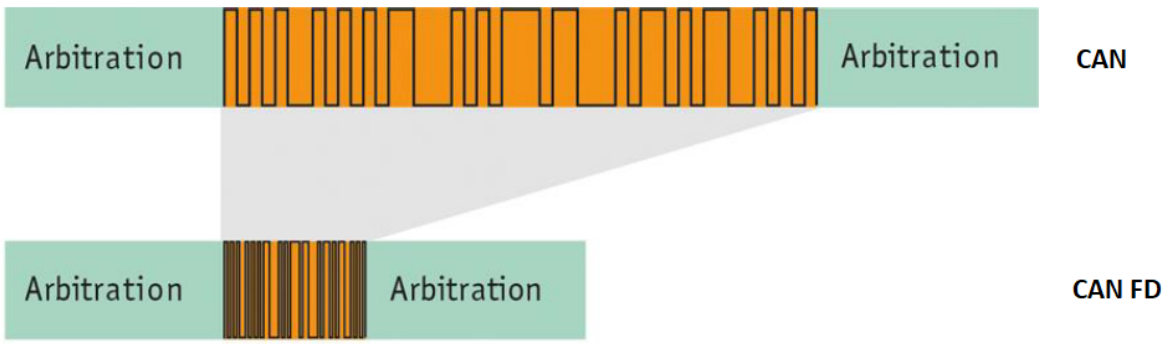
Fonte: Vector (2013)

Figura 10 – Velocidade de transmissão do campo de dados.



Fonte: Adaptado de Vector (2013)

Figura 11 – Comparativo da velocidade de transmissão do campo de dados de ambos os protocolos.



Fonte: Adaptado de Vector (2013)

Por fim, tem-se os métodos de detecção de erros de ambos os protocolos. O CAN possui cinco tipos diferentes de erros. Na camada física possuem dois, *bit errors* e *stuffing*

errors, os outros três erros são apresentados na camada de mensagens que são ACK *errors*, CRC *errors* e *frame check message*. As definições estão de acordo com Paret (2007).

- ***Bit Errors***
  A medida que o transmissor envia uma mensagem ao barramento, o mesmo faz a leitura do sinal. Com isso, se há a leitura de algum *bit* que apresenta divergência do que se espera ter enviado, um *frame* de erro é enviado ao barramento para identificar a falha na transmissão;

- ***Stuffing Errors***
  O *bit stuffing* está presente nos campos de SOF, *arbitration*, *control*, *data* e CRC. O erro é detectado quando é enviado ao barramento mais de cinco *bits* dominantes ou recessivos consecutivos;

- **ACK *Errors***
  Se não há mudança de *bit* recessivo para dominante no ACK *slot* a ECU transmissora detecta o erro e envia um *frame* de erro para o barramento;

- **CRC *Errors***
  O cálculo do CRC envolve a escolha de um polinômio gerador, a anexação de *bits* de preenchimento, a divisão binária dos dados, a obtenção do resto dessa divisão como o valor CRC e a anexação desse valor aos dados para detecção de erros durante a transmissão ou armazenamento, com isso, se houver divergência entre as ECUs uma mensagem de erro é enviada ao barramento;

- ***Frame Check Messages***
  Os campos SOF, RTR, IDE, DLC, EOF e os delimitadores possuem seus *bits* fixos, de acordo com a espeficação do protocolo a ser utilizado (2.0 A ou 2.0 B), ou seja, o valor deles sempre serão os mesmos. Posto isso, se algum desses valores for lido de forma diferente pelo receptor, o mesmo envia a mensagem de erro para o barramento.

Já a detecção de erros para o protocolo CAN FD obtiveram melhoras com relação ao CAN, devido a maior complexidade do protocolo. Com isso, os métodos para se detectar um erro estão descritos abaixo de acordo com Mutter e Hartwich (2015).

- ***Bit Monitoring***
  Ocorre a verificação *bit* a *bit* para conferir se o que foi enviado é o mesmo que foi lido, afinal a ECU lê todas as suas mensagens enviadas. Com isso, se houve algo divergente um *frame* de erro é enviado. A diferença é que, no protocolo CAN FD, a ECU transmissora pode fazer a aferição da mensagem após o envio do campo de dados.

- ***Frame Format Check***
  A detecção ocorre nos *bits* que possuem formatos definidos, como os delimitadores, r0, EDL, IDE e r1.

- ***Cyclic Redundancy Check***
  Com o princípio de funcionamento muito semelhante ao do CAN, porém o polinômio utilizado é diferente, devido a inclusão dos *dynamics stuff bits* antes do campo CRC. Para *frames* de até 16 *bytes* de dados usa-se um polinômio de 17 *bits*. Para valores até 64 *bytes* de dados, o polinômio é de 21 *bits*.

- ***Acknowledgement***
  Se o *bit* no ACK não for dominante, quer dizer que a mensagem recebida não está de acordo com a transmitida, logo, esse *bit* permanece recessivo ao invés de mudar para dominante.

- ***Stuff Rule Check***
  O erro é detectado quando mais de 5 *bits* são enviados de mesmo valor lógico. Porém, o CAN FD faz a inserção de *stuffs bits* fixados no início do campo CRC a cada 4 *bits*, garantindo que não ocorra esse erro. Caso não haja o *stuff bit* fixado após 4 *bits* há também a sinalização do erro.

- ***Stuff Count Check***
  A quantidade de *stuffs bits* é contada tanto no transmissor quanto nos receptores, caso o valor verificado entre ambas não for igual, é considerado um erro de CRC e a receptora considera o *frame* como inválido.

- ***Interaction Between Error Detection and Error Signaling***
  Devido a possibilidade de ter a aferição da mensagem após o campo de dados, a detecção de um erro de CRC só será sinalizado após a leitura do delimitador ACK.

Por fim, tem-se os métodos de arbitração ao barramento para ambos os protocolos. O método CSMA/CA (*Carrier Sensor Multiple Access / Collision Avoidance*) é utilizado, tanto para o protocolo CAN quanto para o CAN FD. Com isso, há os *bits* dominantes e recessivos, no qual, com o envio de múltiplos *frames*, o *bit* dominante sobreporá, devido a diferença de potencial, o *bit* recessivo no barramento. Desta forma é definida a ordem de prioridade das mensagens que serão acessadas no *bus*, como pode ser visto na Figra 12 (PARET, 2007).

Figura 12 – Método de arbitragem.

ECU na rede	ID da ECU	Número do ID convertido para binário											
1	200	0	0	0	0	1	1	0	0	1	0	0	0
2	250	0	0	0	0	1	1	1	1	1	0	1	0
3	300	0	0	0	1	0	0	1	0	1	1	0	0
4	350	0	0	0	1	0	1	0	1	1	1	1	0
5	400	0	0	0	1	1	0	0	1	0	0	0	0
Barramento	200	0	0	0	0	1	1	0	0	1	0	0	0

Fonte: elaborado pelo autor

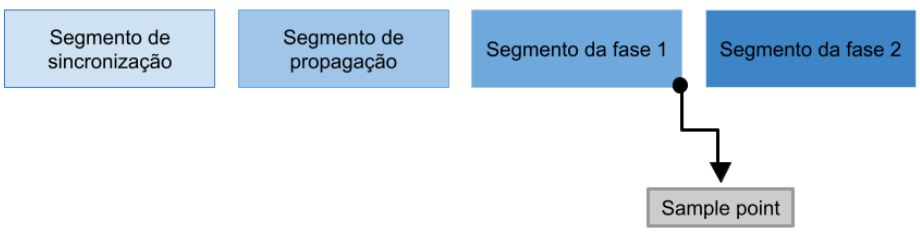
Tem-se na Figura 12 cinco ECUs denominadas por números de 1 a 5 e que possuem seus respectivos números ID. Percebe-se que a mensagem que vai acessar o barramento será a que possuir o menor número ID. Com isso, a ECU 1, que possui seu ID igual a 200 obteve êxito no envio de sua mensagem, enquanto as demais unidades de controle esperarão a rede entrar em *bus-off*, em que não há mensagens sendo transmitidas, para continuar o envio de seus respectivos *frames* (BORTH, 2016).

2.3 Sincronização de *bits*

A sincronização de *bits* tem papel fundamental para os protocolos CAN e CAN FD para compensar a diferença de tempo entre a transmissão e o recebimento das mensagens, ou seja, a ECU transmissora envia o *frame*, porém, a ECU receptora não fará a leitura do mesmo instântaneamente, levará um certo tempo para que a mensagem percorra o barrameto, que varia de acordo com o *baud rate prescaler* (BRP). Com isso, o tempo de sincronização é fundamental para que não haja a leitura incorreta da mensagem enviada (PARET, 2007).

Posto isso, ambos os protocolos possuem um *clock*, derivado do *clock* do micro-controlador, na qual faz a sincronização através de quatro etapas, sendo o segmento de sincronização, segmento de propagação, fase do segmento 1 e fase do segmento 2. Entre as fases de segmentos 1 e 2, há o *sample point*, ponto de leitura do *bit* pela ECU receptora, a Figura 13 descreve o que foi supracitado.

Figura 13 – Tempo de sincronização de *bit*.



Fonte: elaborado pelo autor

Sendo assim, cada etapa possui um determinado tq (*Time Quantum*). Segundo Andrade (2014) a quantidade de *time quantum* em cada etapa e a finalidade de cada uma estão descritas abaixo:

- **Segmento de sincronização**:
  Esta etapa é responsável por realizar a sincronização dos nós. Como o próprio nome indica, sintoniza a entrada na rede com o *baud rate prescaler*. Aqui há no máximo 1 tq para ambos os protocolos.

- **Segmento de propagação**:
  Possui a finalidade de compensar todos os atrasos do meio físico da rede utilizando até 8 *time quantuns* para o CAN e 32 tq's para o CAN FD.

- **Segmento da fase 1**:
  Este segmento tem a finalidade de se ajustar, entre 1 a 8 tq's para o CAN e até 32 *time quantuns* para o CAN FD, para que ocorra a leitura correta do *bit* enviado.

- **Segmento da fase 2**:
  Possui a mesma característica da fase 1, porém aqui pode ocorrer o alargamento ou encurtamento da mesma, quem define é a fase anterior. Caso a fase 1 seja alongada, a fase 2 será encurtada, no caso do encurtamento da fase 1, há o alongamento da fase 2. A quantidade de *time quantuns* é a mesma da fase 1 para ambos os protocolos.

- **Sample point**:
  É neste ponto que ocorre a leitura do *bit* por parte da ECU receptora.

Dessarte do supracitado, há a necessidade de expor a forma de se encontrar o *time quantum*, porém, para tal há a realização de alguns passos. Inicialmente, necessita-se do tempo que 1 *bit* gasta para percorrer a rede automotiva, descrita pela Equação 2.1

$$C_{bit} = \frac{1}{T_{tx}} \tag{2.1}$$

Em que:

$\mathbf{C_{bit}}$ é o tempo que 1 *bit* leva para ser transmitido;

$\mathbf{T_{tx}}$ é a taxa de transmissão da rede;

Com o tempo de 1 *bit*, precisa-se agora do tempo de 1 tq. Haja visto a Equação 2.2 exemplifica este tempo.

$$T_{tq} = \frac{[2\,(BRP + 1)]}{F_{osc}} \tag{2.2}$$

Na qual:

BRP é o *baud rate prescaler* que é o *clock* definido para o protocolo de comunicação.

$\mathbf{F_{osc}}$ é a frequência do oscilador.

Com as Equações 2.1 e 2.2 a Equação 2.3 pode ser escrita, representando a quantidade de *time quantuns* necessária para a sincronização do *bit*.

$$N_{tq} = \frac{C_{bit}}{T_{tq}} \tag{2.3}$$

Tem-se que:

$\mathbf{N_{tq}}$ é a quantidade de tq's.

2.4 Métricas de performance

Os parâmetros de performance têm como finalidade avaliar o desempenho da rede automotiva, ou seja, mensurar quão efetiva a comunicação é. Com isso, o tempo de transmissão da mensagem, o *busload*, o *Bit Error Rate* (BER), a Latência e o *troughput* são os parâmetros avaliados nesse projeto que serão explicados a seguir. Inicialmente, serão abordados os tempos de transmissão de ambos os protocolos. Este cálculo é o mesmo para o CAN e o CAN FD. O termo que os diferenciará é o tempo do *frame* na rede C_m, portanto, as variáveis com subíndice "m", "j", "k" e "*bit*" são para ambos os protocolos. Os termos que possuírem "CAN" e "CAN FD" como subíndice são específicos para o CAN e o CAN FD, respectivamente. Com isso, o tempo de transmissão da mensagem se trata do período levado para transmitir um *frame*, logo, os autores Tindell e Burns (1994) definiram que para se calcular, no pior caso, o tempo de transmissão dos protocolos CAN e CAN FD, faz-se uso da Equação 2.4

$$R_m = J_m + w_m + C_m \tag{2.4}$$

Na Equação 2.4 tem-se que:

$\mathbf{R_m}$ é o tempo total de transmissão da mensagem;

$\mathbf{J_m}$ é o *jitter* da mensagem, que significa a variação estatística no tempo de atraso de uma mensagem na rede;

$\mathbf{w_m}$ é o tempo entre ser adicionada à fila de transmissão e ser transmitida, no pior caso;

$\mathbf{C_m}$ é o tempo em que uma mensagem percorre na rede, para ambos os protocolos;

De acordo com Silva (2015) o *jitter* (J_m) é a diferença entre o tempo esperado do tempo real, determinado empiricamente, e normalmente, utiliza-se o valor de 0,1 milisegundo. Segundo Tindell e Burns (1994) o tempo de espera na fila de prioridades (w_m) pode ser representado pela Equação 2.5

$$w_m = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{2.5}$$

Em que:

$\mathbf{B_m}$ é o tempo de bloqueio da mensagem no pior caso;

$\mathbf{hp(m)}$ é o grupo de mensagens com prioridade superior ao *frame* "m";

$\mathbf{J_j}$ é o *jitter* de uma mensagem;

$\tau_{\mathbf{bit}}$ é o tempo levado para transmitir um *bit*, *bit time*;

$\mathbf{T_j}$ é o período de uma determinada mensagem;

$\mathbf{C_j}$ é o tempo que uma mensagem percorre na rede, para ambos os protocolos;

Porém, a Equação 2.5 tem o tempo de espera (w_m) dos dois lados da equação. Para resolver esse problema utiliza-se a relação de recorrência, demostrada na Equação 2.6, apresentada por Tindell e Burns (1994).

$$w_m^{n+1} = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{w_m^n + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{2.6}$$

Em seguida, Tindell e Burns (1994) determinaram o tempo de bloqueio de uma mensagem (B_m) através da Equação 2.7.

$$B_m = \underset{\forall k \in lp(m)}{max}(C_k) \tag{2.7}$$

Aqui o termo $lp(m)$ faz referência ao grupo de mensagens com prioridade inferior a mensagem "m". Por fim, o tempo na qual a mensagem percorre o barramento CAN pode ser descrito pela Equação 2.8, na qual foi adaptada de Andrade (2014).

$$C_{CAN} = \left(\frac{b_{Stuffing} + 8B_D}{5} + b_{Stuffing} + b_{ACK} + 8B_D \right) \tau_{bit} \tag{2.8}$$

Em que:

$\mathbf{C_{CAN}}$ é o tempo que uma mensagem percorre a rede CAN;

$\mathbf{b}_{Stuffing}$ é a quantidade de *bits* que possui no *frame* até o último *bit* do CRC;

$\mathbf{B_D}$ é a quantidade de *bytes* no campo de dados, está acompanhada pelo multiplicador 8 para fazer a conversão para *bits*;

$\mathbf{b_{ACK}}$ é a quantidade de *bits* após o último *bit* do CRC;

$\tau_{\mathbf{bit}}$ é o tempo em que 1 *bit* percorre a rede;

Dessarte, a Equação 2.8 representa o tempo em que a mensagem levará para percorrer a rede, tanto para o CAN 2.0 A quanto 2.0 B. Vale ressaltar que, a parte da equação que é dividia por 5 representa a quantidade de *bit stuffing* daquela mensagem. Além disso, Punnekkat, Hansson e Norstrom (2000) abreviaram a Equação 2.8, de forma a torná-la mais enxuta, como pode ser visto na Equação 2.9

$$C_{CAN} = \left(\frac{T + 8B_D - 1}{4} + O + 8B_D\right)\tau_{bit} \tag{2.9}$$

Na qual, segundo Godoy (2007), as variáveis T e O possuem os valores de acordo com a Tabela 4.

Tabela 4 – Valor das variáveis do protocolo CAN.

Variáveis da equação	Valor das variáveis para cada tipo de mensagem	
	2.0 A	2.0 B
T	34	54
O	47	67

Fonte: Adaptado de Godoy (2007)

Já o protocolo CAN FD, por possuir dois tipos de taxa de transmissão em uma mensagem, como supracitado no Tópico 2.2.3, deve-se realizar a soma de ambas as partes, rápida e devagar. Com o intuito de obter o tempo gasto pelo *frame* para percorrer o barramento, como descrito na Equação 2.10, formulada por Andrade (2014).

$$C_{CAN\,FD} = C_{Fast} + C_{Slow} \tag{2.10}$$

No qual:

$\mathbf{C_{CAN\,FD}}$ é o tempo que uma mensagem percorre a rede CAN FD;

$\mathbf{T_{Fast}}$ é o tempo dos *bits* mais rápidos;

$\mathbf{T_{Slow}}$ é o tempo dos *bits* mais lentos.

O tempo dos *bits* lentos pode ser descrito pelas Equação 2.11 que foi adaptada de Andrade (2014).

$$C_{Slow} = \frac{1,2b_{AD} + b_{DD}}{T_{Frame\,slow}} \tag{2.11}$$

Na Equação 2.11 o C_{Slow} indica o tempo que a mensagem lenta percorre o barramento, ou seja, a parte que é transmitida no mesmo tempo do *clock*, b_{AD} é a quantidade de *bits* antes do campo de dados, b_{DD} são os *bits* após o campo de dados e o $T_{Frame\,slow}$ é o BRP da parte lenta. O fator multiplicador de 1,2 está presente para contabilizar o *bit stuffing* que pode ser gerado na parte anterior ao campo de dados.

Tabela 5 – Valor das variáveis da parte devagar do protocolo CAN FD.

Variáveis da equação	Valor das variáveis para cada tipo de mensagem	
	CAN FD padrão	CAN FD estendido
$\mathbf{b_{AD}}$	17	36
$\mathbf{b_{DD}}$	13	13

Fonte: elaborado pelo autor

Para a parte rápida, tem-se duas formas de calcular, uma para o campo de dados menores que 16 *bytes*, de acordo com a Equação 2.12 e outra para o campo de dados maior ou igual que 16 *bytes* na Equação 2.13. Ambas foram adaptadas de Andrade (2014).

$$C_{fast\,<\,16} = \left[1,2\left(8B_D + b_{DS}\right) + CRC_{17} + 5\right]\tau_{bit} \tag{2.12}$$

$$C_{fast\,\geq\,16} = \left[1,2\left(8B_D + b_{DS}\right) + CRC_{21} + 6\right]\tau_{bit} \tag{2.13}$$

Na qual:

$\mathbf{b_{DS}}$ é a quantidade de *bits* sujeitos a *bit stuffing* no campo de dados;

$\mathbf{CRC_{17}}$ é a quantidade de *bits* no campo CRC. Soma-se 5 para adicionar o *bit stuffing*;

$\mathbf{CRC_{21}}$ é a quantidade de *bits* no campo CRC. Soma-se 6 para adicionar o *bit stuffing*;

Para as Equações 2.12 e 2.13 tem-se a Tabela 6 que descreve os valores a serem utilizados nas variáveis dispostas.

Tabela 6 – Valor das variáveis da parte rápida do protocolo CAN FD.

Variáveis da equação	Valor das variáveis para cada tipo de mensagem	
	CAN FD padrão	CAN FD estendido
$\mathbf{b_{DS}}$	6	6
$\mathbf{CRC_{17}}$	17	-
$\mathbf{CRC_{21}}$	-	21

Fonte: elaborado pelo autor

Após o esclarecimento do tempo de transmissão, tem-se que o *busload*, como supracitado, representa o tanto de tempo que o barramento é ocupado, o que é de extrema

relevância, tendo em vista que, um barramento com baixo carregamento indica que mais *bits* podem transitar na rede sem afetar a performance da mesma. Já o contrário indica uma sobrecarga no barramento, uma saturação da rede o que reduz o desempenho da mesma (GODOY, 2007). Para se calcular o *busload* para o protocolo CAN, necessita-se da Equação 2.9 demonstrada pelos autores Tindell e Burns (1994). Portanto, para se obter o *busload*, tem-se a razão entre o somatório do tempo de todos os *frames* que percorrem o barramento pelo tempo de amostragem de cada *frame*, como mostra a Equação 2.14, adaptada de Godoy (2007)

$$U_{CAN} = \frac{\sum_{i=1}^{N} C_{CAN,i}}{T_i} * 100\% \tag{2.14}$$

Entretanto, o protocolo CAN FD não pode fazer uso da mesma equação que o CAN utiliza, com isso, Andrade (2014) desenvolveu, em sua tese, uma equação que foi validada com testes e simulações utilizando o *software* de medição CANoe. Portanto, na Equação 2.15 tem-se o *busload* do protocolo CAN FD.

$$U_{CAN\ FD} = \frac{\sum_{i=1}^{n} C_{CAN\ \ FD,i}}{T_i} * 100\% \tag{2.15}$$

Em que:

$\mathbf{T_i}$ é o tempo da mensagem do CAN/CAN FD;

Para as métricas BER, latência e *throughput* tem-se o mesmo método de cálculo para os dois protocolos. O BER, indica a resiliência do protocolo ao erro pois, é a razão entre a quantidade de *bits* com erro pelo número total de *bits* enviados corretamente, elucidado pela Equação 2.16. Em outras palavras, o BER indica a taxa de erro do protocolo, logo, quanto menor esse valor, mais resiliente ao erro é o protocolo.

$$BER = \frac{b_{Error}}{b_{Total}} \tag{2.16}$$

Em seguida, a latência é o tempo medido desde o envio até o recebimento da mensagem. esta métrica de performance indica, qualitativamente, o desempenho do protocolo em tempo real, uma vez que uma latência muito alta pode acarretar no não recebimento do *frame*. A latência está representada pela Equação 2.17

$$L = T_{Rx} - T_{Tx} - T_{ciclo} \tag{2.17}$$

Em que L é a latência, T_{Tx} é o tempo medido na transmissão da mensagem, T_{Rx} é o tempo medido na recepção da mensagem e T_{ciclo} é o período de amostragem definido.

Por fim, o *throughput* é a razão entre o total de dados transmitidos pelo tempo medido, ou seja, é capaz de mensurar a razão da quantidade de dados transmitidos com

sucesso em um determinado período. O que representa a eficiência da rede em lidar com tráfego de *frames*. A Equação 2.18 representa o *throughput*.

$$\sigma = \frac{D_T}{T_T} \tag{2.18}$$

Na qual σ representa o *throughput*, D_T a quantidade total de dados transmitidos e T_T o tempo total mensurado.

2.5 Speedgoat Performance Real-Time Target Machine

A simulação realizada em um *hardware* em tempo real é fundamental em uma variedade de aplicações, desde o controle de máquinas industriais até testes de veículos autônomos e desenvolvimento de sistemas de automação. Para atender a essas demandas, a *Speedgoat* oferece uma solução de *hardware* de alto desempenho conhecida como *Performance Real-Time Target Machine*, representado na Figura 14. Neste tópico, exploraremos as características e benefícios dessa máquina, destacando como ela capacita a execução de simulações em tempo real de alta complexidade.

Figura 14 – Equipamento da *Speedgoat Performance Real-Time Target Machine*.

Fonte: Dedicated System (2023)

A *Performance Real-Time Target Machine* da *Speedgoat* é projetada para fornecer desempenho de elite. Equipada com um poderoso processador Intel Core i7 4.2 GHz 4 core, 32 GB de memória RAM e 1 TB de memória no *drive* principal, esta máquina oferece a capacidade de executar simulações altamente complexas e intensivas em termos de computação, mantendo uma alta taxa de atualização. Isso é fundamental para simulações

onde a precisão temporal é crítica. Esta máquina oferece uma variedade de opções de E/S (entrada/saída), incluindo portas *Ethernet Gigabit*, entradas e saídas analógicas e digitais, que pode ser personalizado na compra do equipamento. Isso a torna adequada para aplicações que exigem comunicação com uma ampla gama de dispositivos externos, como sensores, atuadores e sistemas de controle.

Além disso, ela é altamente modular e personalizável. Ela oferece slots PCIe que permitem a expansão com placas adicionais para atender às necessidades específicas do projeto. Essa flexibilidade permite que a máquina seja construída de acordo com as demandas de sua aplicação, podendo variar o processador, quantidade de memória RAM, etc. A alta escalabilidade oferece a capacidade de adicionar recursos à medida que seu projeto evolui, garantindo que você possa atender a futuras necessidades sem ter que substituir todo o sistema.

A *Speedgoat* é conhecida por sua dedicação à precisão e confiabilidade. Ela é cuidadosamente projetada para garantir que as simulações executadas nela sejam altamente precisas e previsíveis, atendendo aos rigorosos padrões de controle em tempo real. Em resumo, a *Performance Real-Time Target Machine* da *Speedgoat* é uma escolha de *hardware* de alto desempenho para aplicações de simulação em tempo real e controle que demandam precisão, desempenho e flexibilidade. Sua capacidade de execução de simulações complexas e comunicação eficaz a torna uma escolha valiosa para uma ampla gama de setores e aplicações.

2.6 Estado da arte

2.6.1 Análise de desempenho do protocolo CAN para aplicações agrícolas

Inicialmente, Godoy (2007) desenvolveu em sua tese de mestrado uma ferramenta de análise de performance de rede CAN para aplicações agrícolas. Para tal, fez-se uso do *software* LabVIEW 7.0 para gerar um arquivo executável que pudesse ser realizado o *download* em qualquer computador. Para a validação da ferramenta, o autor desenvolveu uma arquitetura eletroeletrônica utilizando o microcontrolador PIC18F258 como controlador CAN, o *transceiver* MCP2551 como transmissor, entre outros dispositivos eletrônicos para a criação da rede CAN. Após a validação, foi realizado a análise da performance da rede CAN de um robô agrícola através dos parâmetros de taxa de utilização do barramento, taxa de utilização de mensagens, taxa de eficiência do barramento e tempo de transmissão. Como resultado, o autor desenvolveu uma ferramenta de análise de rede CAN para auxiliar no desenvolvimento de outras redes de acordo com a norma ISO 11783.

2.6.2 Análise de desempenho do protocolo CAN FD utilizando o CANoe

Em sequência, o artigo apresentado pelos autores Cheon e Jeon (2013) teve como objetivo avaliar a performance da rede CAN FD através do *software* CANoe da empresa *Vector GmbH*. Para tal, foram criadas as bases de dados de ambos os protocolos através da ferramenta CANdb++ para haver a troca de mais de 40 mensagens entre 4 ECUs. Com isso, foi comparado o carregamento do barramento e o tempo de resposta no pior cenário para ambos os protocolos.

2.6.3 Desenvolvimento do cálculo de *busload* para o CAN FD

Neste estudo, o autor Andrade (2014) propõe uma contribuição significativa ao desenvolver uma equação matemática para calcular o *busload* em redes CAN FD. O cerne da pesquisa reside na comparação do *busload* e da latência entre redes CAN e CAN FD, explorando dois estudos de caso distintos. No primeiro estudo, o autor utiliza sua fórmula para calcular o *busload* da rede, posteriormente comparando esses resultados com os valores medidos pelo *software* CANoe. Este processo é conduzido em três cenários específicos, são 6 ECUs que ao todo transmitem 24, 31 e 39 mensagens, respectivamente. A comparação direta entre a abordagem teórica e as medições práticas no CANoe oferece uma validação robusta da equação proposta. No segundo estudo de caso, o foco da análise é estendido para avaliar o *busload* e a latência de ambas as redes em um contexto de sistema de controle de malha fechada para um motor de corrente contínua.

2.6.4 Análise de desempenho do protocolo CAN para veículos pesados

Outro estudo, realizado por Silva (2015), teve como objetivo obter dados reais de três veículos pesados diferentes e analisá-los em três estudos de caso, simulando a comunicação entre as ECUs em ambiente virtual. No caso em questão, utilizou-se o *software* *BusMaster* para realizar a simulação da comunicação entre ECUs via protocolo CAN. O primeiro estudo de caso consistiu na análise da comunicação entre 4 ECUs, no estudo de caso dois foram 10 ECUs enquanto no estudo de caso 3 foi modelado a rede em ambiente virtual para comparar os resultados com dados reais do veículo. Com isso, métricas de performance, tais como *busload*, tempo de transmissão da mensagem e a quantidade de *frames* enviados ao barramento foram avaliadas para os 3 casos.

2.6.5 Análise comparativa de desempenho dos protocolos CAN e CAN FD para veículos pesados

Em 2016, Borth (2016), em sua tese de mestrado realiza 5 estudos de caso para comparar a performance das redes CAN e CAN FD. Os cinco estudos de caso são compostos por 5 redes automotivas, sendo duas redes CAN conectadas por um *gateway*, duas

redes CAN unificadas, duas redes CAN FD unificadas com campo de dados de no máximo 8 *bytes*, uma rede CAN e uma rede CAN FD conectadas por um *gateway* e por fim duas redes CAN FD unificadas enviando mais de 8 *bytes* no campo de dados. Com isso, o autor avalia o *busload* e a latência das mensagens sícronas e assíncronas.

2.6.6 Análise comparativa de desempenho dos protocolos CAN e CAN FD para aplicações agrícolas

Os autores Zago e Freitas (2017) realizaram um estudo comparativo a respeito do protocolo CAN e CAN FD, em que, foi simulado no *software* CANoe uma rede agrícola contendo 3 ECUS e 1 terminal virtual. Para tal, foram simuladas 3 redes, sendo uma CAN com campo de dados de 8 *bytes*, a segunda sendo CAN FD com campo de dados de 8 *bytes* também, e por fim a rede CAN FD com 64 *bytes* no campo de dados, para avaliar a performance das redes e avaliar em qual cenário cada uma desempenha melhor, para tal, os autores avaliam o *busload*, a quantidade de mensagens enviadas e o tempo de transmissão no pior cenário para os 3 casos.

2.6.7 Análise comparativa de desempenho dos protocolos CAN e CAN FD para aplicações gerais

Os autores Xie et al. (2017) também realizaram um estudo comparativo entre os protocolos CAN e CAN FD, porém, foram realizados 4 casos, iniciando pelo protocolo CAN com 500 kbps de taxa de transmissão, em sequência três experimentos com o protocolo CAN FD com taxa de transmissão de 500 kbps no campo de *acknowledgement*, o que os diferencia é a taxa de trasmissão do campo de dados que são de 1 Mbps, 2 Mbps e 5 Mbps, respectivamente. Para todos foram realizadas 14 experimentos, alterando o comprimento do campo de dados, sendo 1/2/3/4/5/6/7/8/12/16/20/32/48/64 *bytes*. Com isso, foi avaliado o *busload*, *message overhead* e o tempo de transmissão no pior cenário.

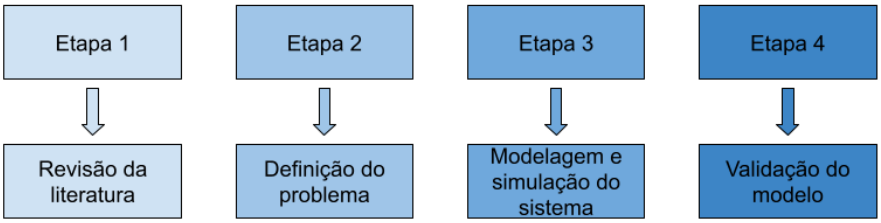
2.6.8 Análise comparativa de desempenho de uma rede mista CAN/CAN FD utilizando *gateway* para aplicações automotivas

Ainda no ano de 2017, foi proposto no estudo realizado pelos autores An e Jeon (2017) a transferência das mensagens de 64 *bytes* pela rede CAN FD à 2,5 Mbps para o *gateway* que armazena os dados e envia para a rede CAN 8 mensagens com 8 *bytes* à 500 kbps. Para isso, foi utilizado o *hardware* TC234 EVB como *gateway*, o equipamento VN5610 da Vector para mensurar e o *software* CANoe para simular as redes. Com isso, foram analisados o *busload* e a quantidade de mensagens enviadas à rede.

3 Metodologia

Neste capítulo serão elucidados os métodos e ferramentas utilizados para a realização da modelagem e simulação da rede automotiva de ambos os protocolos. Com isso, as 4 etapas para a realização deste estudo estão descritas abaixo, pela Figura 15 e melhor explicitadas nos tópicos subsequentes.

Figura 15 – Etapas para a realização da metodologia.



Fonte: elaborado pelo autor

3.1 Revisão da literatura

Na revisão da literatura foi realizado um estudo a respeito dos protocolos de comunicação automotiva utilizados na indústria. Com isso, os protocolos CAN e CAN FD foram selecionados com o intuito de realizar um estudo exploratório comparativo entre ambos, para aferir a motivação da empresa Robert Bosch GmbH em desenvolver o protocolo CAN FD sendo que o CAN já estava bastante difundido na indústria. Além de que no Laboratório de Sistemas e Controle da Universidade de Brasília há a licença do *software Matlab/Simulink* para a realização da modelagem, simulação e medições, além de possuir o equipamento *Performance Real-Time Target Machine* da empresa *Speedgoat* para embarcar o *software* no *hardware* e realizar as simulações e medições em tempo real.

Com a escolha dos protocolos a serem estudados, foi realizada a busca por artigos e referências relacionadas ao tema, com isso, foi realizado o estudo da arte para compreender o que têm sido desenvolvido na última década, além das métricas de performance utilizadas para avaliar cada protocolo individualmente. Para tal, foram levantados tópicos a respeito das redes automotivas e suas características, o modelo OSI/ISO, a camada física e a camada de enlace dos protocolos. Para completar o estudo, uma breve abordagem sobre sicronização de *bits*, fator muito importante para aplicações em tempo real, métricas de performance que podem ser utilizadas para comparar o CAN com o CAN FD e uma breve abordagem a respeito do equipamento *Performance Real-Time Target Machine* da *Speedgoat*.

3.2 Definição do problema

O tema central desta pesquisa é comparar os protocolos de comunicação CAN e CAN FD. Ambos têm desempenhado um papel fundamental na indústria automotiva, permitindo a comunicação eficiente entre os diversos componentes eletroeletrônicos dos veículos. No entanto, uma questão que se coloca é por que o protocolo CAN ainda é amplamente empregado, mesmo com a presença do protocolo CAN FD, que oferece uma maior largura de banda e a capacidade de transmitir uma quantidade significativamente maior de dados em uma única mensagem. Essa questão é de grande relevância em um contexto em que veículos automotivos estão se tornando cada vez mais tecnológicos e com mais mensagens transitando no barramento, exigindo uma alta troca de informações e comunicações precisas e eficientes entre as diversas unidades de controle em tempo real.

A motivação por trás deste estudo reside na necessidade de aprofundar o entendimento dos protocolos de comunicação no setor automotivo, com isso, o foco da pesquisa está em compreender as vantagens e desvantagens dos protocolos CAN e CAN FD em aplicações automotivas específicas. A pesquisa parte da hipótese inicial de que o protocolo CAN FD pode apresentar um desempenho superior em métricas como *busload*, quantidade de mensagens enviadas e latência em comparação com o protocolo CAN tradicional.

A importância desta pesquisa para a contribuição do conhecimento na área de comunicação automotiva, bem como na sua relevância para a indústria, onde a escolha do protocolo de comunicação pode afetar significativamente o desempenho e a eficiência dos sistemas de controle em veículos automotivos modernos, se dá pelo fato de ser realizado a modelagem da rede em ambiente virtual para em sequência embarcar este modelo no *hardware* da *Speedgoat*. Essa pesquisa seguirá os padrões da indústria ao realizar simulações em ambiente virtual antes de embarcar o sistema em *hardware*, garantindo uma compreensão abrangente do comportamento da rede automotiva. O *software Simulink* e o equipamento *Performance Real-Time Target Machine* da *Speedgoat* foram selecionados por possuírem as ferramentas necessárias para analizar ambos os protocolos, além da compatibilidade mútua entre sí. Assim, esta monografia busca fornecer informações valiosas para profissionais da indústria automotiva, pesquisadores e acadêmicos interessados no aprimoramento das comunicações em sistemas automotivos, contribuindo para a evolução da tecnologia nesse campo crucial.

3.3 Modelagem e simulação do sistema

Nesta etapa, foram modeladas e simuladas as redes automotivas. Foram realizados dois estudos de caso, em que ambos utilizaram o *software Simulink* para realizar a modelagem da arquitetura eletroeletrônica e o *hardware Performance Real-Time Target Machine*, da empresa *Speedgoat*, para embarcar o modelo, simulá-lo e mensurá-lo em tempo real. A

descrição da metodologia de cada estudo de caso será dividida em duas partes, sendo a primera referente as simulações realizadas em ambiente virtual (*Simulink*) e a segunda a respeito da simulação com o modelo embarcado no *hardware* em tempo real da *Speedgoat*, para ambos os protocolos.

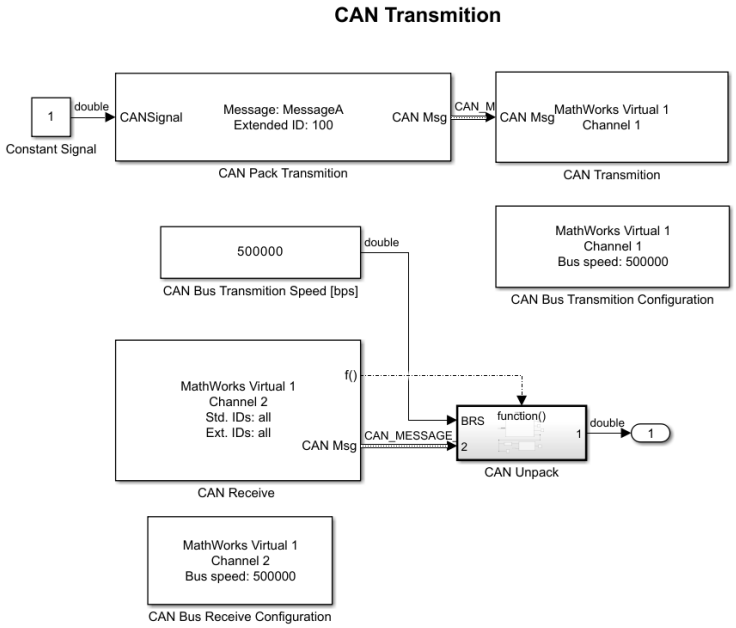
3.3.1 Estudo de caso 1

Para o estudo de caso 1 foi realizado a comunicação entre duas ECUs virtuais, sendo uma transmissora e uma receptora. A primeira parte do experimento foi modelar a rede no *software Simulink* e simulá-la em ambiente virtual. A segunda parte do experimento foi modelada a mesma rede no *Simulink*, porém utilizou-se dos blocos disponíveis pela biblioteca *Simulink Real-Time: Speedgoat I/O Blockset*, para em seguida, embarcar a rede modelada no *hardware Performance Real-Time Target Machine* da *Speedgoat* para que a máquina realiza-se as simulações e as medições dos parâmetros de desempenho da rede em tempo real.

Para a primeira parte do experimento foram realizadas simulações para os protocolos CAN e CAN FD. Para tal utilizou-se a biblioteca *Vehicle Network Toolbox* (VNT) para fazer a construção da arquitetura eletroeletrônica. O fluxo de informações parte do envio de um sinal constante, no valor de 1, para o bloco *CAN Pack*, que tem como função encapsular esse sinal no campo de dados do *frame*. Em seguida, o dado encapsulado é transmitido para o bloco *CAN Transmition*, o qual transmite a mensagem, com os dados, do canal 1 (transmissor) para o canal 2 (receptor) à taxa de 500 kbps, que foi configurado pelo bloco *CAN Configuration* para a transmissão e recepção, como pode ser visto pela Figura 16. Após a transmissão da mensagem, o bloco *CAN Receive* recebe o *frame* e transmite os dados, encapsulados, para o bloco *Function-Call Subsytem*, na qual, dentro do bloco está inserido o *CAN Unpack*.

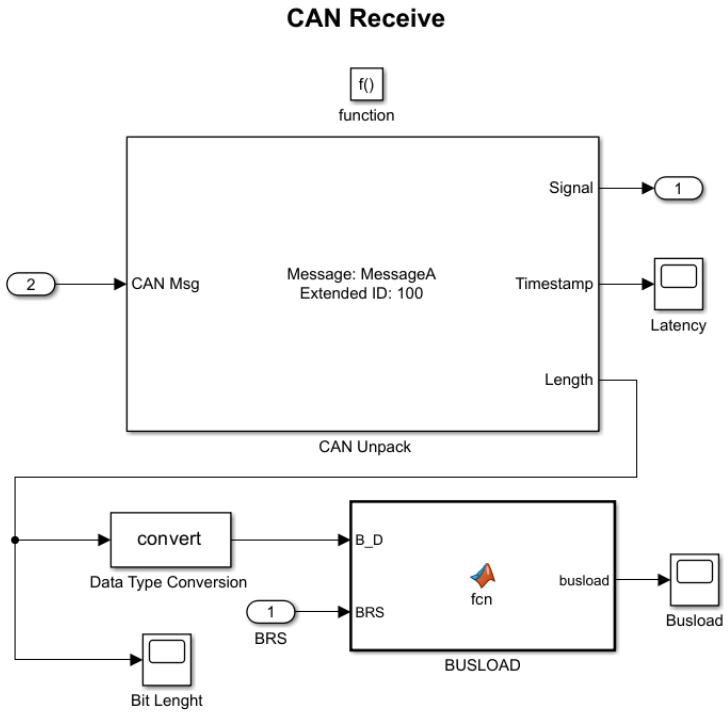
Na Figura 17 pode-se observar que os dados encapsulados são representados pela entrada denominada por "2", a qual transmite para o bloco *CAN Unpack* que desencapsula os dados e dispõe como saída os parâmetros necessários para que as análises sejam realizadas através do *Scope*. Em sequência, tem-se a saída *Lenght*, que é entrada do *Convert*, para converter o comprimento do campo de dados de *8uint* para *double* para realizar os cálculos de *busload*. Após o bloco *Convert*, tem-se que esse valor é um *input* do bloco *Matlab Function-Subsystem*, este bloco tem como função calcular o *busload* da rede através da Equação 2.14 para, em seguida, realizar a leitura dos dados pelo *Scope*. Como para as duas partes do estudo de caso 1 tem-se a mesma arquitetura (simulada em ambiente virtual e embarcada no *hardware*), tem-se que as características da mensagem que percorre o barramento e a configuração da rede também são as mesmas, com isso, logo abaixo encontram-se a configuração da rede CAN, na Tabela 7 e a característica da mensagem que percorreu a rede, de acordo com a Tabela 8.

Figura 16 – Arquitetura eletroeletrônica do *transmiter* da rede CAN para o estudo de caso 1 no *Simulink*.



Fonte: elaborado pelo autor

Figura 17 – Arquitetura eletroeletrônica do *receiver* da rede CAN para o estudo de caso 1 no *Simulink*.



Fonte: elaborado pelo autor

Tabela 7 – Parâmetros de configuração da rede CAN para o estudo de caso 1 no *Simulink*.

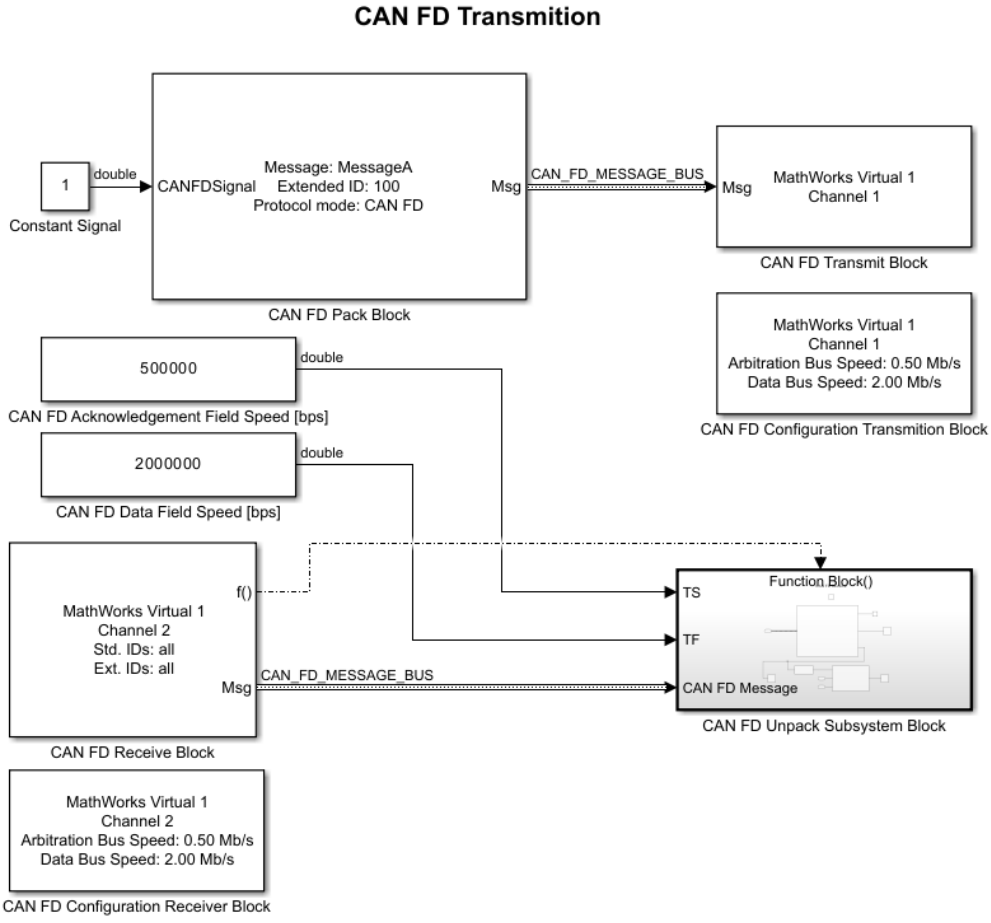
Tipo de mensagem	Velocidade de transmissão	Tempo de simulação
Periódica	500 kbps	2 segundos

Tabela 8 – Definição do *frame* da rede CAN para o estudo de caso 1 no *Simulink*.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
100	8 bytes	5 milisegundos	400 mensagens	Estendido	Constante

Após concluir as simulações do protocolo CAN, a etapa seguinte consistiu na realização de simulações do protocolo CAN FD. A modelagem da rede foi realizada da mesma forma do protocolo CAN, porém, os blocos selecionados foram os do protocolo CAN FD. Desta forma, o fluxo de dados segue a mesma estrutura do CAN convencional, tem-se um sinal constante que é encapsulado no bloco *CAN FD Pack* e transmitido para o bloco *CAN FD Tranmition*, o qual envia a mensagem para o bloco *CAN FD Receiver*, de acordo com a Figura 18.

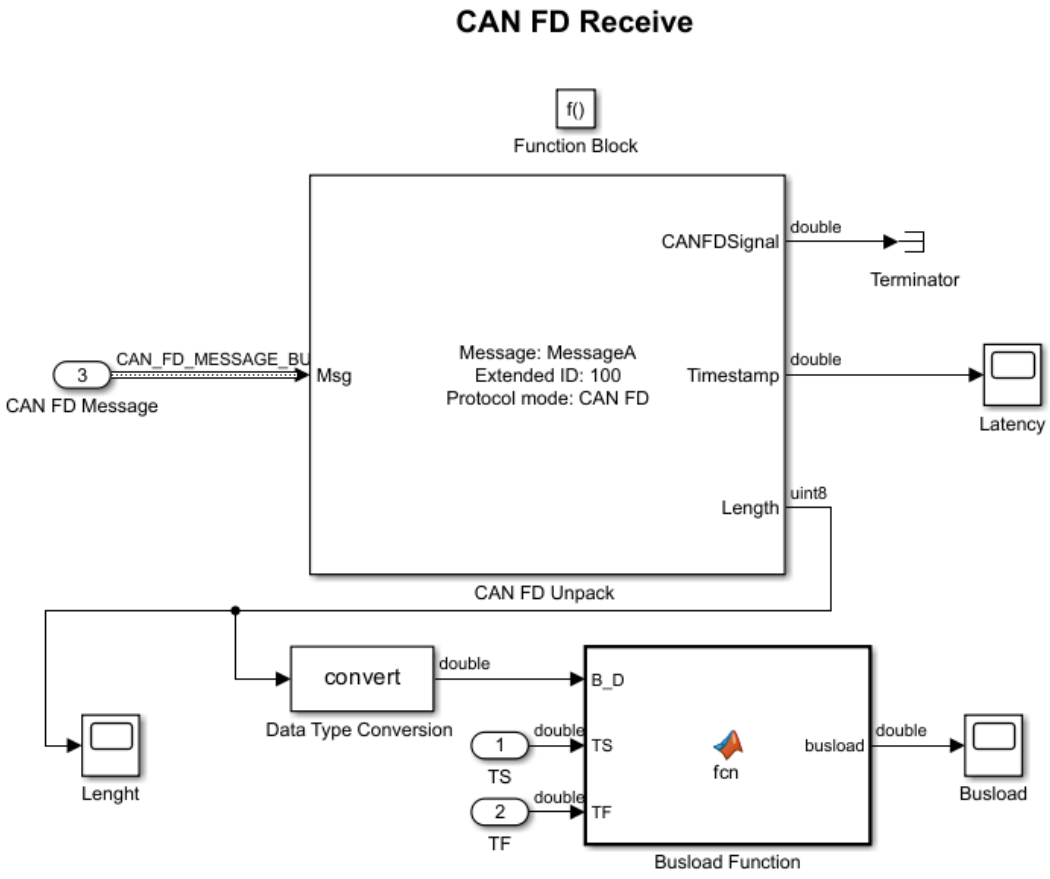
Figura 18 – Arquitetura eletroeletrônica do *transmiter* da rede CAN FD para o estudo de caso 1 no *Simulink*.



Fonte: elaborado pelo autor

Em sequência, o dado encapsulado é transmitido para o bloco *Function-Call Subsystem*, na qual, dentro do bloco está inserido o *CAN FD Unpack* que desencapsula os dados e transmite para o *Scope* realizar as medições para as análises. Como para o protocolo CAN, o *Lenght* também é convertido em *double* para ser utilizado como entrada do bloco *Matlab Function-Subsystem* para a realização dos cálculos de *busload*. Tem-se também *input* da taxa de transmissão dos campos de *acknowledgement* e de dados neste bloco, representados pelos blocos denominados por "TS" e "TF", respectivamente. Com isso, o carregamento da rede é calculado de acordo com a Equação 2.15 que é apresentada pelo *Scope*, de acordo com a Figura 19. Abaixo encontra-se a configuração da rede na Tabela 9 e as características do *frame* enviado na simulação do CAN FD na Tabela 10 para ambas as partes do estudo de caso 1, assim como na rede CAN, citado anteriormente.

Figura 19 – Arquitetura eletroeletrônica do *receiver* da rede CAN FD para o estudo de caso 1 no *Simulink*.



Fonte: elaborado pelo autor

Tabela 9 – Parâmetros de configuração da rede CAN FD para o estudo de caso 1 no *Simulink*.

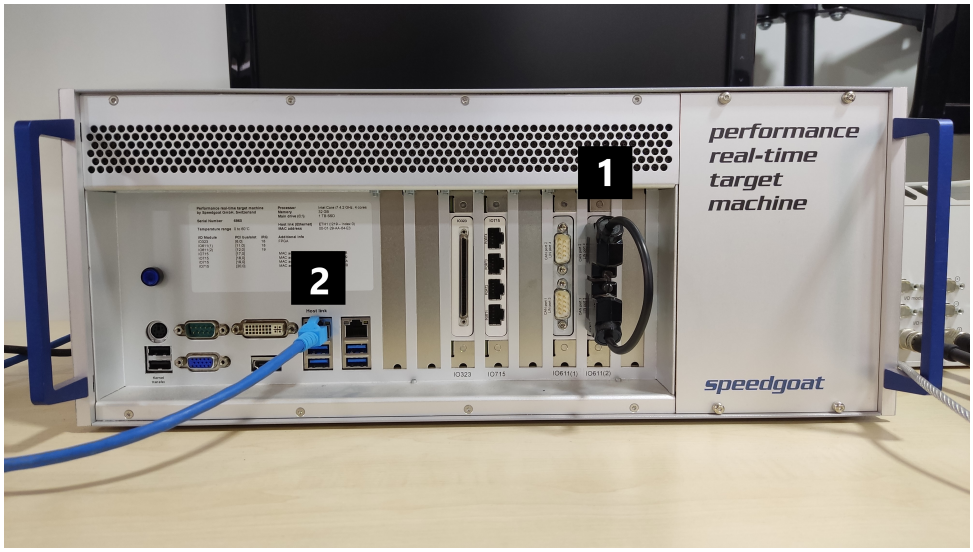
Tipo de mensagem	Velocidade de transmissão do *Acknoledgement Field*	Velocidade de transmissão do *Data Field*	Tempo de simulação
Periódica	500 kbps	2000 kbps	2 segundos

Tabela 10 – Definição do *frame* da rede CAN FD para o estudo de caso 1 no *Simulink*.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
100	8 bytes	5 milisegundos	400 mensagens	Estendido	Constante

Para a segunda parte do estudo de caso 1, tanto o protocolo CAN quanto o CAN FD fazem uso dos mesmo blocos, o que os diferencia são os blocos de *CAN/CAN FD Pack*, *CAN/CAN FD Unpack* e a configuração utilizada no *Block Parameters: CAN & LIN Setup*, que são encontrados na biblioteca *Simulink Real-Time: Speedgoat I/O Blockset*, disponibilizado somente com a posse da licença. Inicialmente modelou-se a rede no *Simulink* para embarcar o modelo no *hardware Performance Real-Time Target Machine* da *Speedgoat*, na qual, a ECU transmissora foi embarcada no módulo IO611(2) na porta 1 enquanto a ECU receptora está no módulo IO611(2) na porta 2, perceptível na Figura 20, que demonstra a conexão entre as duas portas serial, sinalizado pelo número "1". Nesta mesma figura é possível aferir uma conexão *ethernet*, na qual é responsável por exibir as informações adquiridas pela *Performance Real-Time Target Machine* no *Simulink*. O intuito deste estudo de caso está em desenvolver habilidades para simulação em tempo real e fazer a integração de *software* com *hardware* para comparar, qualitativamente, os protocolos mencionados, além de validá-los com os resultados obtidos através da simulação no *Simulink*.

Figura 20 – Conexões na *Performance* da *Speedgoat* utilizado no estudo de caso 2.

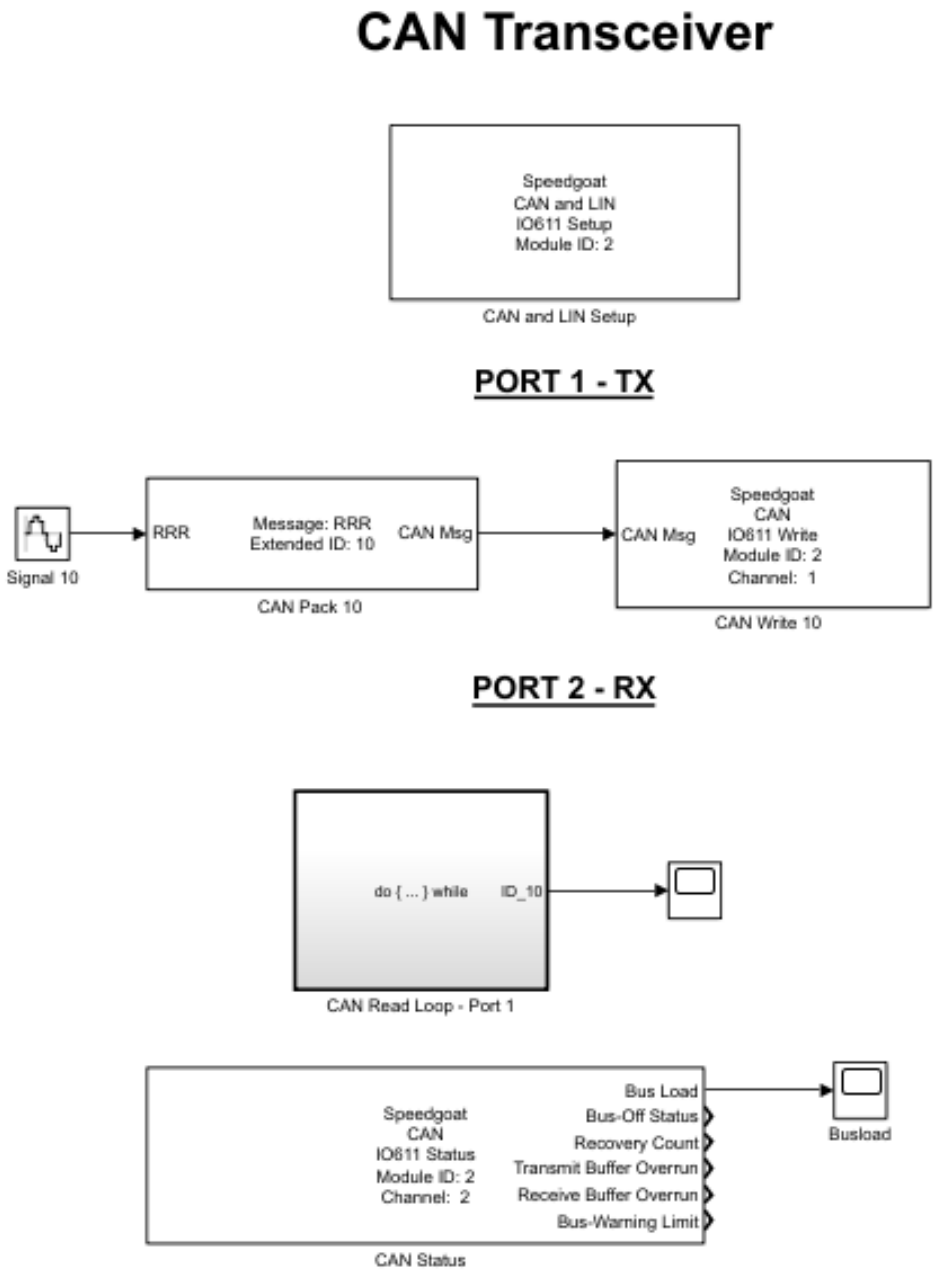


Fonte: elaborado pelo autor

Com isso, o fluxo de dados parte de um sinal senoidal para o bloco *CAN Pack*, que encapsula os dados do sinal e o transmite para o bloco *CAN Write*, que envia a mensagem

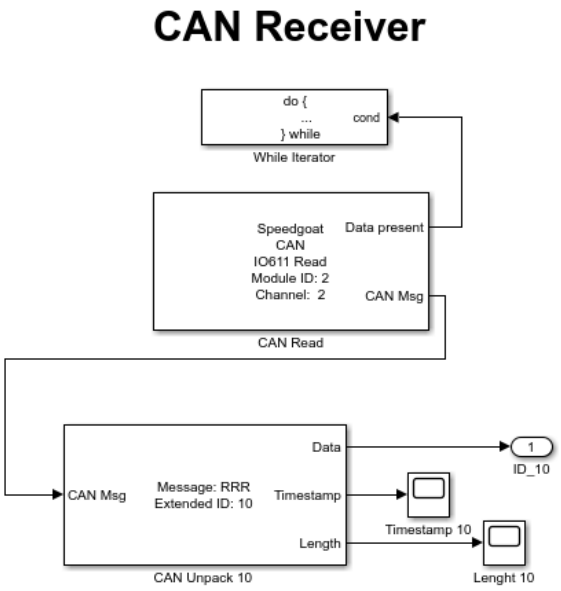
para o bloco *CAN Read Loop*, a cada 5 milisegundos, como pode ser visto na Figura 21. Nesta mesma figura, há também o bloco *CAN Status*, no qual há a medição do *busload* da rede. Dentro do bloco *CAN Read Loop* há a leitura do sinal, através do *CAN Read* que é entrada do bloco *CAN Unpack*. Após o desencapsulamento da mensagem, há a leitura do comprimento do campo de dados e a medição da latência, de acordo com a Figura 22.

Figura 21 – Arquitetura eletroeletrônica do *transmiter* da rede CAN para o estudo de caso 1 embarcada no *hardware*.



Fonte: elaborado pelo autor

Figura 22 – Arquitetura eletroeletrônica do *receiver* da rede CAN para o estudo de caso 1 embarcada no *hardware*.



Fonte: elaborado pelo autor

Para ambos os protocolos, a disposição dos blocos utilizados é a mesma, o que os diferencia são os blocos de *Pack*, *Unpack* e o *setup* selecionado no bloco *Block Parameters: CAN & LIN*, sendo "CAN (HS)" para o protocolo CAN e o "CAN-FD" para o protocolo CAN FD. Como mencionado anteriormente, tem-se na Tabela 11 os parâmetros de configuração da rede CAN embarcada, na Tabela 12 a definição do *frame* que transita na rede durante a simulação CAN em tempo real, na Tabela 13 os parâmetros de configuração da rede CAN FD embarcada e na Tabela 14 a definição da mensagem que percorre o barramento da rede CAN FD em *hardware Performance Real-Time Target Machine* da *Speedgoat*.

Tabela 11 – Parâmetros de configuração da rede CAN para o estudo de caso 1 embarcada.

Tipo de mensagem	Velocidade de transmissão	Tempo de simulação
Periódica	500 kbps	2 segundos

Tabela 12 – Definição do *frame* da rede CAN para o estudo de caso 1 embarcado.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
10	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal

Assim como pro CAN, a rede CAN FD possui um sinal senoidal que é encapsulado pelo bloco *CAN FD Pack*, para em seguida ser enviado para o bloco *CAN FD Write*, o qual envia a mensagem com os dados para o bloco *CAN Read Loop - Port 1*, de acordo com a Figura 23. Ainda nesta figura é possível vizualizar o bloco *CAN Status*, o qual faz a

Tabela 13 – Parâmetros de configuração da rede CAN FD para o estudo de caso 1 embarcado.

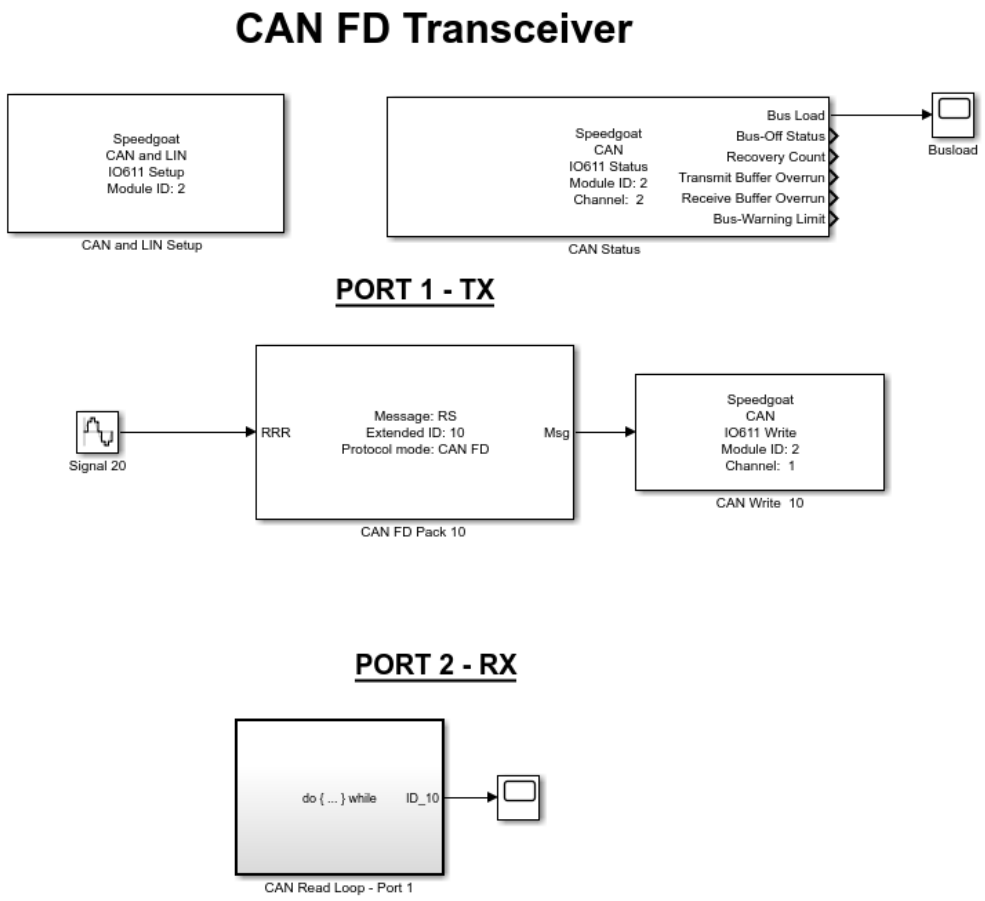
Tipo de mensagem	Velocidade de transmissão do *Acknoledgement Field*	Velocidade de transmissão do *Data Field*	Tempo de simulação
Periódica	500 kbps	2000 kbps	2 segundos

Tabela 14 – Definição do *frame* da rede CAN FD para o estudo de caso 1 embarcado.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
10	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal

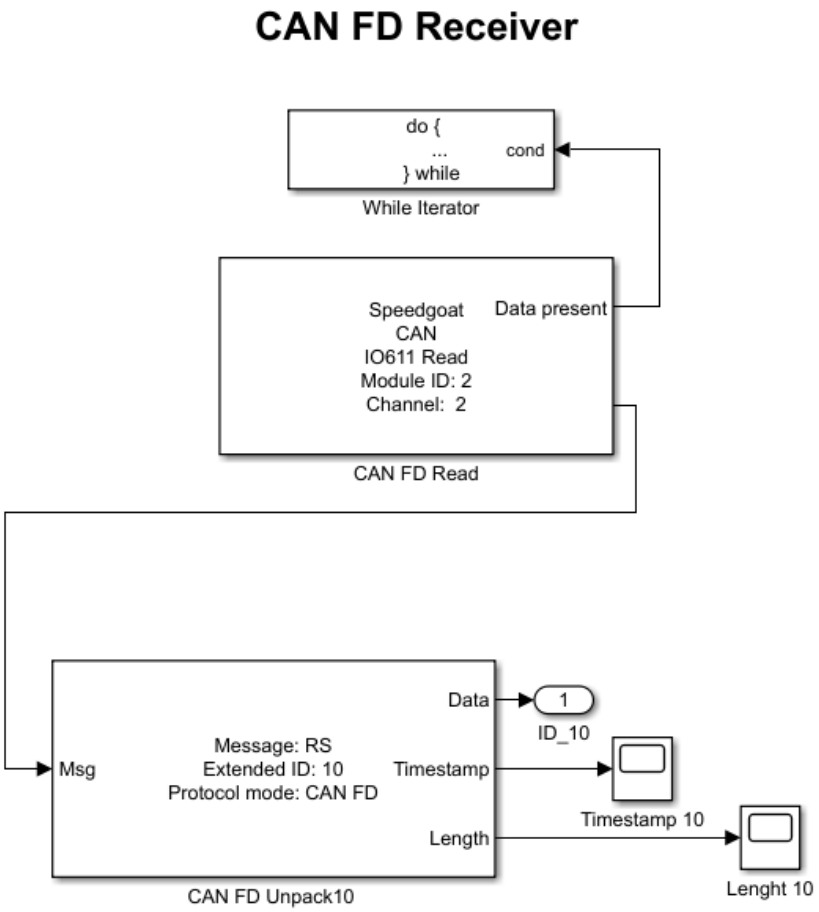
medição do *busload*. Em sequência, o *frame* enviado é recebido pelo bloco *CAN FD Read*, que envia o dado encapsulado para o bloco *CAN FD Unpack*, que disponibiliza os dados para o *Scope* realizar a leitura dos sinais, de acordo com a Figura 24 logo abaixo.

Figura 23 – Arquitetura eletroeletrônica do *transmiter* da rede CAN FD para o estudo de caso 1 embarcada no *hardware*.



Fonte: elaborado pelo autor

Figura 24 – Arquitetura eletroeletrônica do *receiver* da rede CAN FD para o estudo de caso 1 embarcada no *hardware*.



Fonte: elaborado pelo autor

3.3.2 Estudo de caso 2

No estudo de caso 2, realizou-se uma investigação detalhada sobre os protocolos CAN e CAN FD em um veículo de luxo, visando identificar situações específicas para cada protocolo, ou seja, avaliar nesse veículo para qual finalidade o protocolo CAN é utilizado e qual o CAN FD é empregado. Os resultados destacaram o uso preferencial do protocolo CAN FD na transmissão de dados de sensores, como radares e ultrassônicos, enquanto o protocolo CAN é adotado para funções de conforto e iluminação do veículo tais como a posição do assento, teto solar, luz de freio, farol, tomada de indução entre outros. A modelagem e simulação no *Simulink*, seguidas pela implementação prática no *hardware Performance Real-Time Target Machine* da *Speedgoat*, permitiram a avaliação de parâmetros essenciais, como *busload*, latência e quantidade de mensagens no barramento,

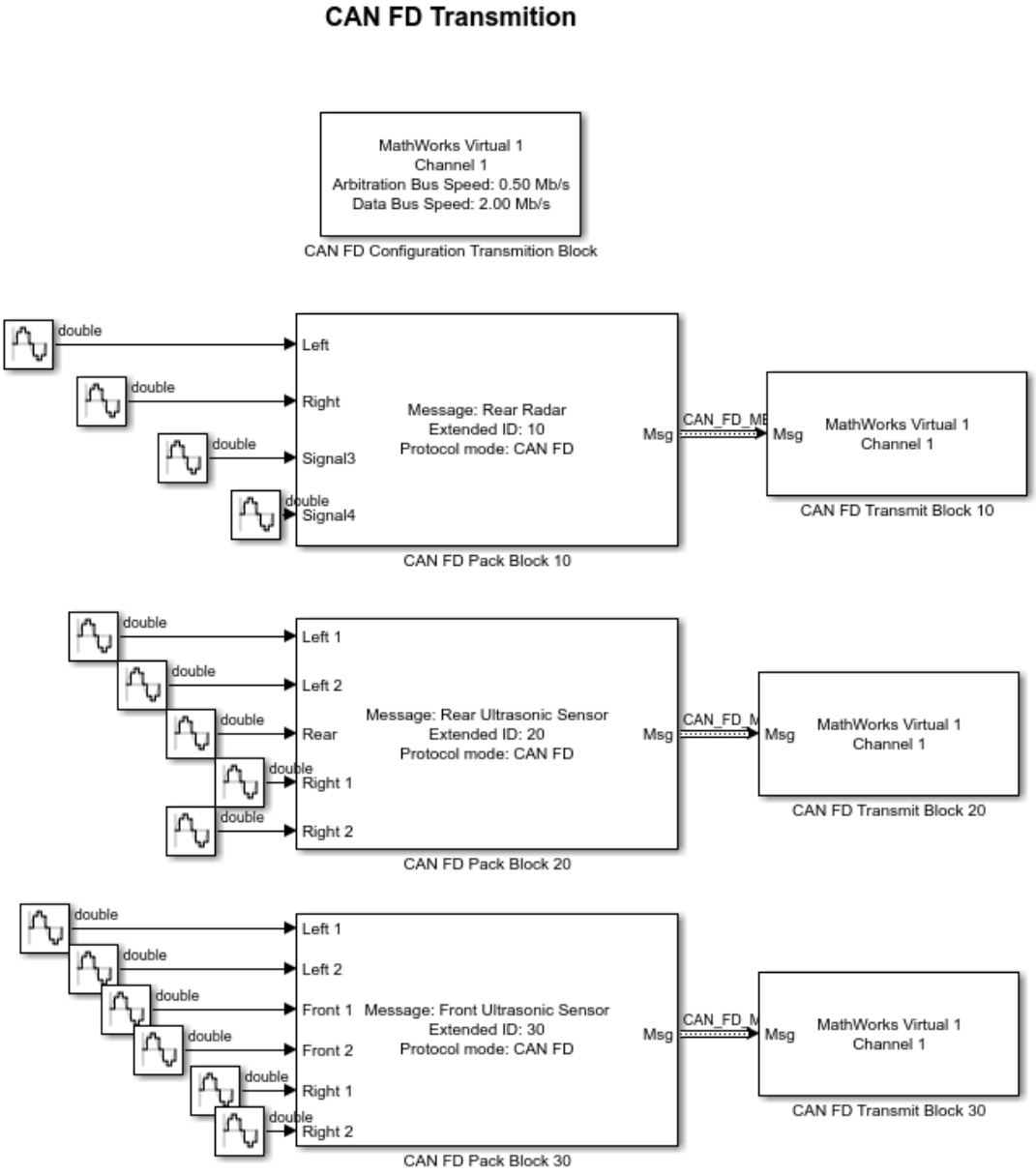
proporcionando uma compreensão abrangente do comportamento dos protocolos.

Com isso, foi modelada uma rede baseada na rede CAN FD dos sensores de um veículo real de luxo de uma montadora. A análise do desempenho desses protocolos envolveu a modelagem e simulação da rede equivalente à dos sensores do veículo real no *software Simulink*, seguida pela implementação prática no *hardware* da *Speedgoat* para ambos os protocolos. Essa abordagem proporcionou uma avaliação abrangente de parâmetros essenciais, como *busload*, latência e quantidade de mensagens no barramento, contribuindo para uma compreensão mais profunda do comportamento de cada protocolo nos contextos de aplicação selecionados para o estudo de caso 2.

Como ambas as redes foram baseadas em uma rede CAN FD, será abordado a modelagem do CAN FD antes da do protocolo CAN, ao contrário do estudo de caso 1, com isso, utilizou-se a VNT para modelar a rede. Para tal, a arquitetura eletroeletrônica opera com taxa de transmissão do campo de *acknowledgement* em 500 kbps e do campo de dados à 2000 kbps, definido pelo bloco *Block Parameters: CAN & LIN*. A rede é disposta de 15 sensores, sendo 4 radares e 11 ultrasônicos, além de 3 ECUs, sendo uma para os radares, uma para os ultrasônicos laterais, e a última para os ultrasônicos dianteiros e traseiros. Os radares enviam sinais senoidais para o campo de dados em um período de 5 milisegundos com 8 *bytes* de dados, já os ultrasônicos enviam sinais senoidais a cada 1 milisegundo com 4 *bytes*. Com isso, os sinais são encapsulados no campo de dados pelo bloco *CAN FD Pack* que enviam o dado encapsulado para o bloco *CAN FD Transmit*, de acordo com a Figura 25. Como o protocolo CAN FD possui maior capacidade de transferência de dados que o CAN, a ECU dos radares envia 4 sinais senoidais, em que, cada um possui 8 *bytes*, logo, a mesma transmite 32 *bytes* no campo de dados, enquanto a ECU dos sensores dianteiros e traseiros são 5 sinais e a dos laterias são 6 sinais senoidais, cada uma com 4 *bytes*, totalizando 20 e 24 *bytes*, respectivamente.

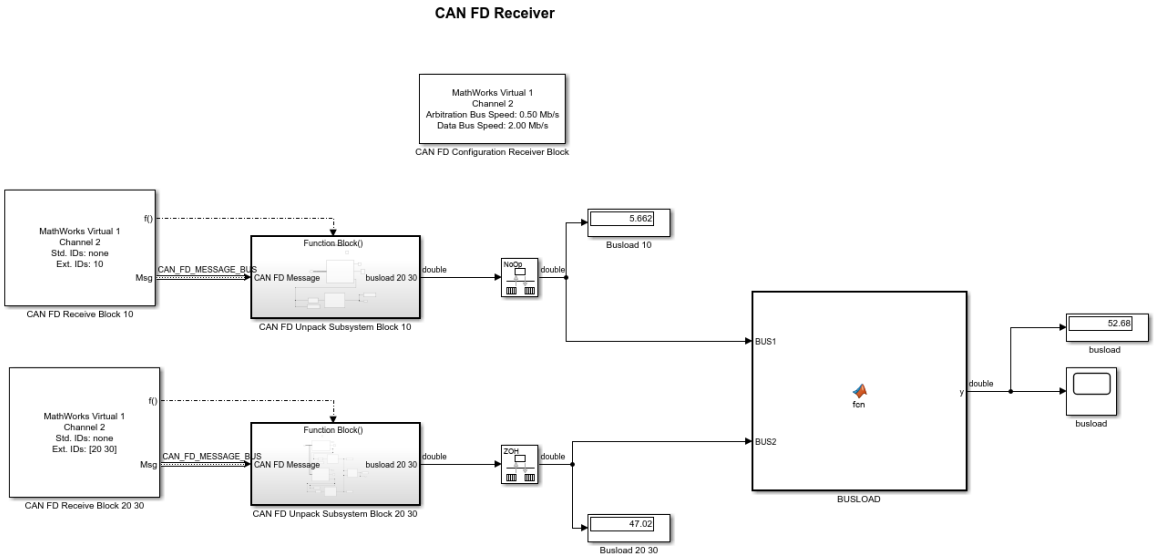
Como as ECUs possuem diferentes períodos de amostragem, a ECU de sensores radares, identificada pelo ID 10 envia os *frames* para a ECU que está com o mesmo período de amostragem, enquanto as demais ECUs enviam sua mensagem para a outra ECU que possui ciclos de envio de 1 milisegundos. Após a trasmissão da mensagem, o bloco *CAN FD Receive* recebe o *frame* e envia os dados encapsulados para o bloco *CAN FD Unpack Subsystem*, como pode ser visto na Figura 26, o qual, dentro dele, há o desencapsulamento dos dados que são lidos pelo *Scope*, de acordo com as Figuras 27 e 28. Assim como no estudo de caso 1, há a necessidade de converter o comprimento do campo de dados para *double* para que sejam realizados os cálculos de *busload*.

Figura 25 – Arquitetura eletreletrônica do *transmiter* da rede CAN FD para o estudo de caso 2 no *Simulink*.



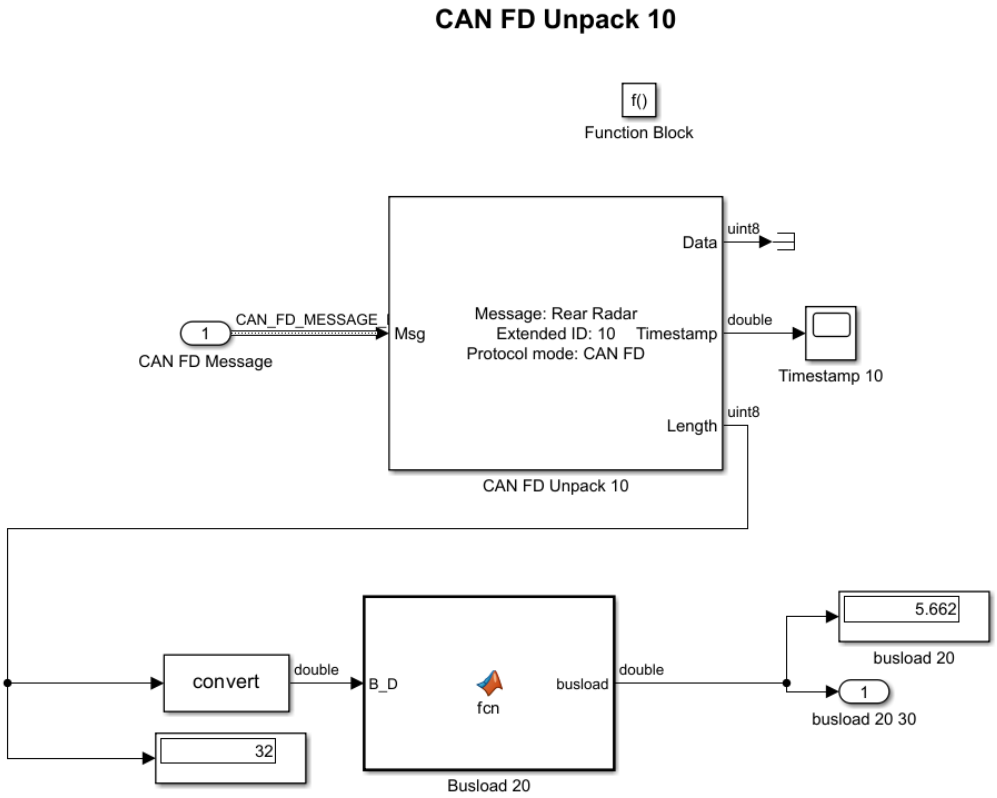
Fonte: elaborado pelo autor

Figura 26 – Arquitetura eletroeletrônica do *receiver* da rede CAN FD para o estudo de caso 2 no *Simulink*.



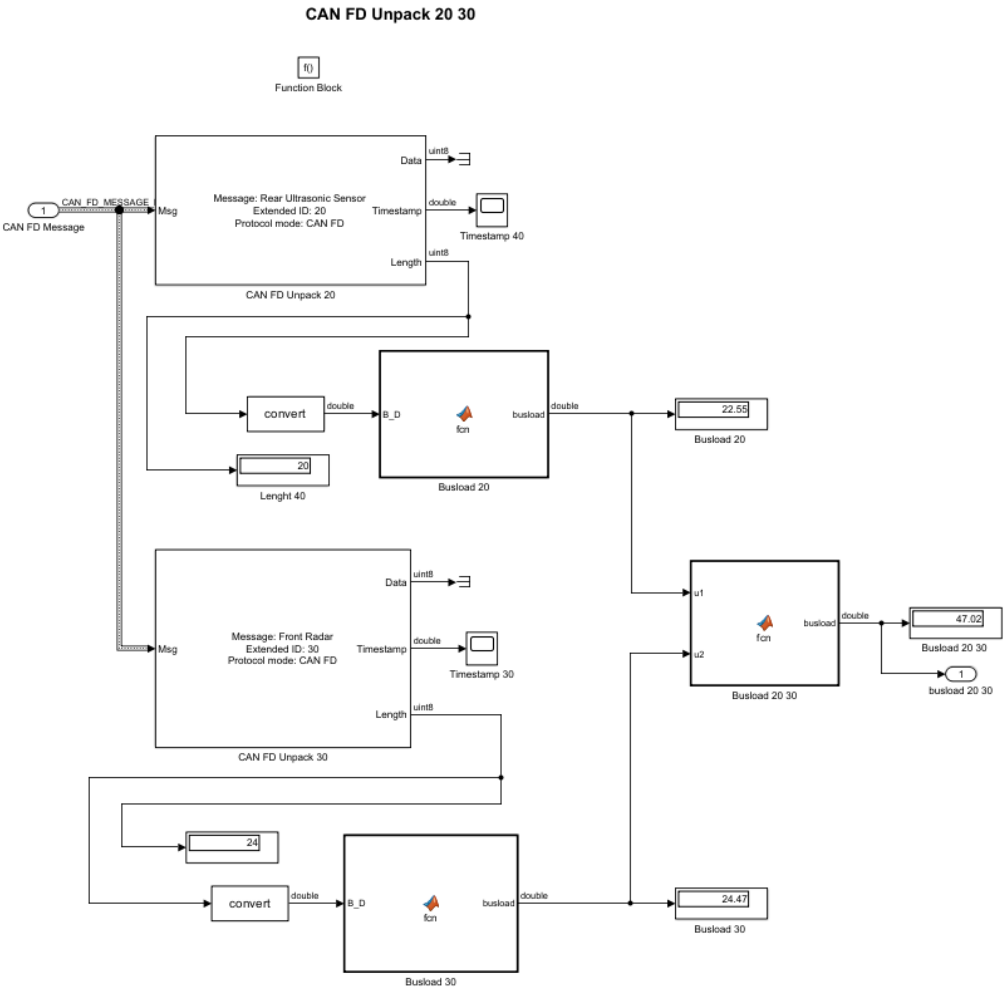
Fonte: elaborado pelo autor

Figura 27 – Arquitetura eletroeletrônica do *receiver* da rede CAN FD para o estudo de caso 2 no *Simulink* para os sensores de radar.



Fonte: elaborado pelo autor

Figura 28 – Arquitetura eletroeletrônica do *transmiter* da rede CAN FD para o estudo de caso 2 no *Simulink* para os sensores ultrasônicos.



Fonte: elaborado pelo autor

Após calcular o carregamento do barramento para cada período de amostragem, esse valor é enviado para o bloco *Rate Transition* para definir o tempo de amostragem de ambos em 5 milisegundos para que o *busload* total da rede fosse calculado. Tem-se na Tabela 15 os parâmetros de configuração da rede CAN FD e na Tabela 16 as definições dos *frames* que percorreram a rede durante a simulação.

Tabela 15 – Parâmetros de configuração da rede CAN FD para o estudo de caso 2 no *Simulink*.

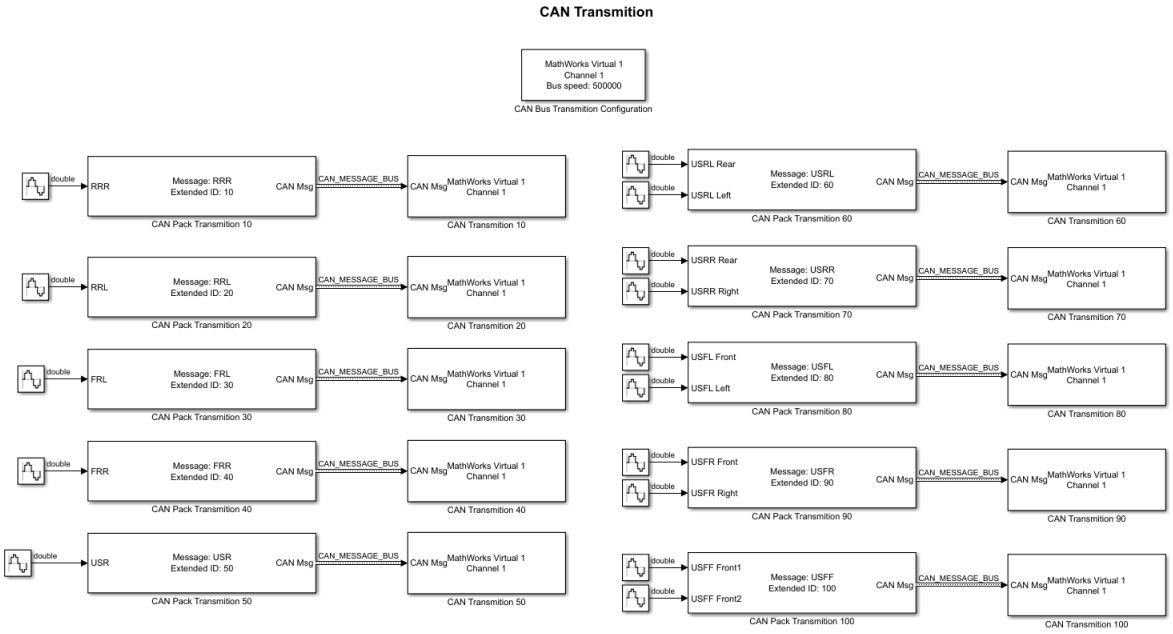
Tipo de mensagem	**Velocidade de transmissão do *Acknoledgement Field***	**Velocidade de transmissão do *Data Field***	**Tempo de simulação**
Periódica	500 kbps	2000 kbps	2 segundos

Tabela 16 – Definição do *frame* da rede CAN FD para o estudo de caso 2 no *Simulink*.

ID	**DLC**	**Período de amostragem**	**Número teórico de mensagens enviadas**	**Formato do frame**	**Tipo de sinal**
10	32 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
20	20 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
30	24 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal

Já a rede CAN, por possuir limitações no comprimento do campo de dados (máximo de 8 *bytes*) não foi possível seguir o *layout* modelado no CAN FD, com isso, foi adaptado para que todos os *frames* fosse enviados à rede. Portanto, os radares, por possuírem 8 *bytes* de dados, foram enviados individualmente por suas respectivas ECUs, já os ultrasônicos, por possuírem 4 *bytes* de dados, são enviados 2 sinais por cada ECU, exceto o sinal "USR", pois são 11 sensores, logo um único sinal do de sensor ultrasônico tinha que ser enviado por uma ECU individualmente, vale ressaltar que os períodos de amostragem são os mesmos que os utilizados no CAN FD. Para modelar a rede, utilizou-se da VNT, assim como no estudo de caso 1, com isso, a rede foi configurada para transmitir à uma taxa de 500 kbps de acordo com o bloco *CAN Bus Transmition Configuration*. Para efetuar a simulação, tem-se 15 sinais senoidais que são encapsulados pelo bloco *CAN Pack* que transmitem os dados encapsulados para o bloco *CAN Transmition* para enviar os *frames*, como descrito na Figura 29 e, na sequência, o bloco *CAN Receive* recebe as mensagens e envia os dados encapsulados para o bloco *Function-Call Subsystem*.

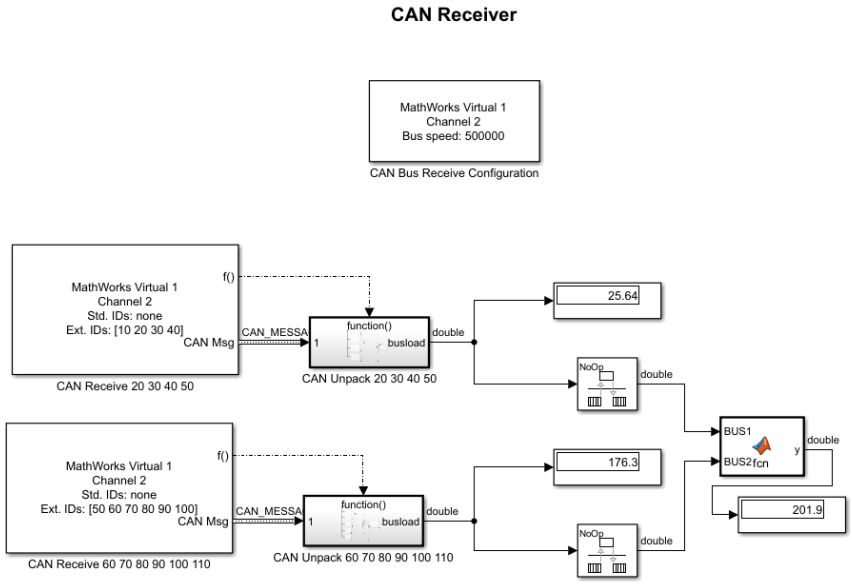
Figura 29 – Arquitetura eletreletrônica do *transmiter* da rede CAN para o estudo de caso 2 no *Simulink*.



Fonte: elaborado pelo autor

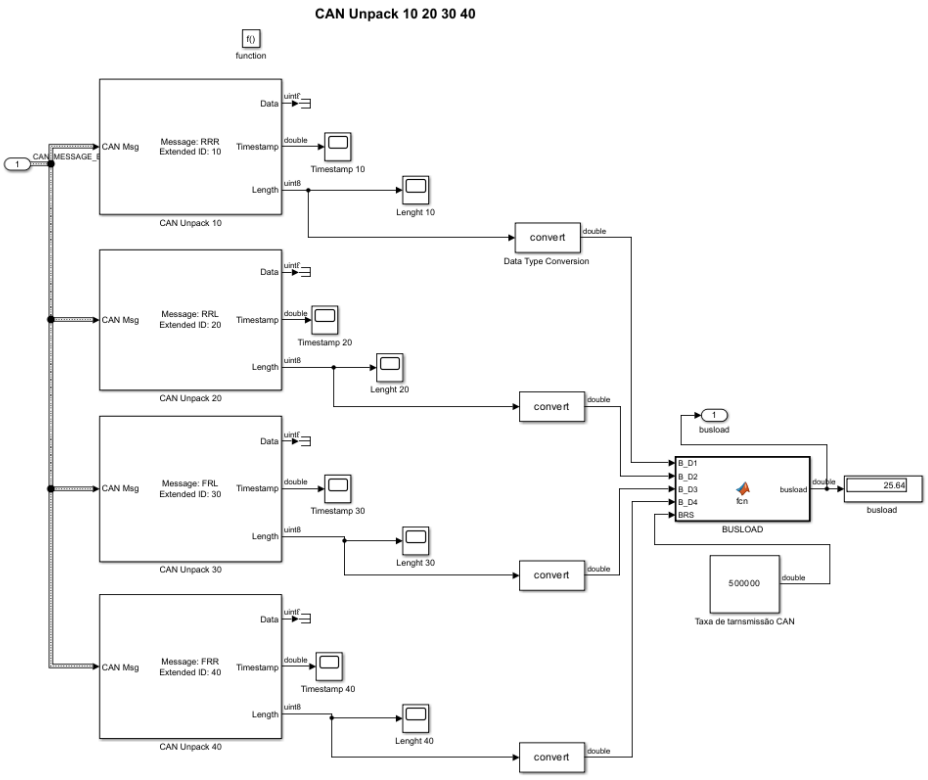
Dentro deste bloco, estão localizados os blocos *CAN Unpack*, os quais desencapsulam os dados que são mensurados pelo *Scope*, de acordo com as Figuras 31 e 32. O comprimento é convertido de *8uint* para *double* para que seja efetuado o cálculo do carregamento da rede para em seguida, o valor calculado ser entrada do bloco *Rate Transition* para que ambos possuam o tempo de amostragem de 5 milisegundos para, por fim, ser entrada do bloco *Matlab Function-Subsystem* para se obter o *busload* da rede completa, como pode ser visto na Figura 30. Pode-se aferir nas Tabelas 17 e 18 a configuração da rede e a configuração dos *frames* enviados por cada ECU, respectivamente.

Figura 30 – Arquitetura eletroeletrônica do *receiver* da rede CAN para o estudo de caso 2 no *Simulink*.



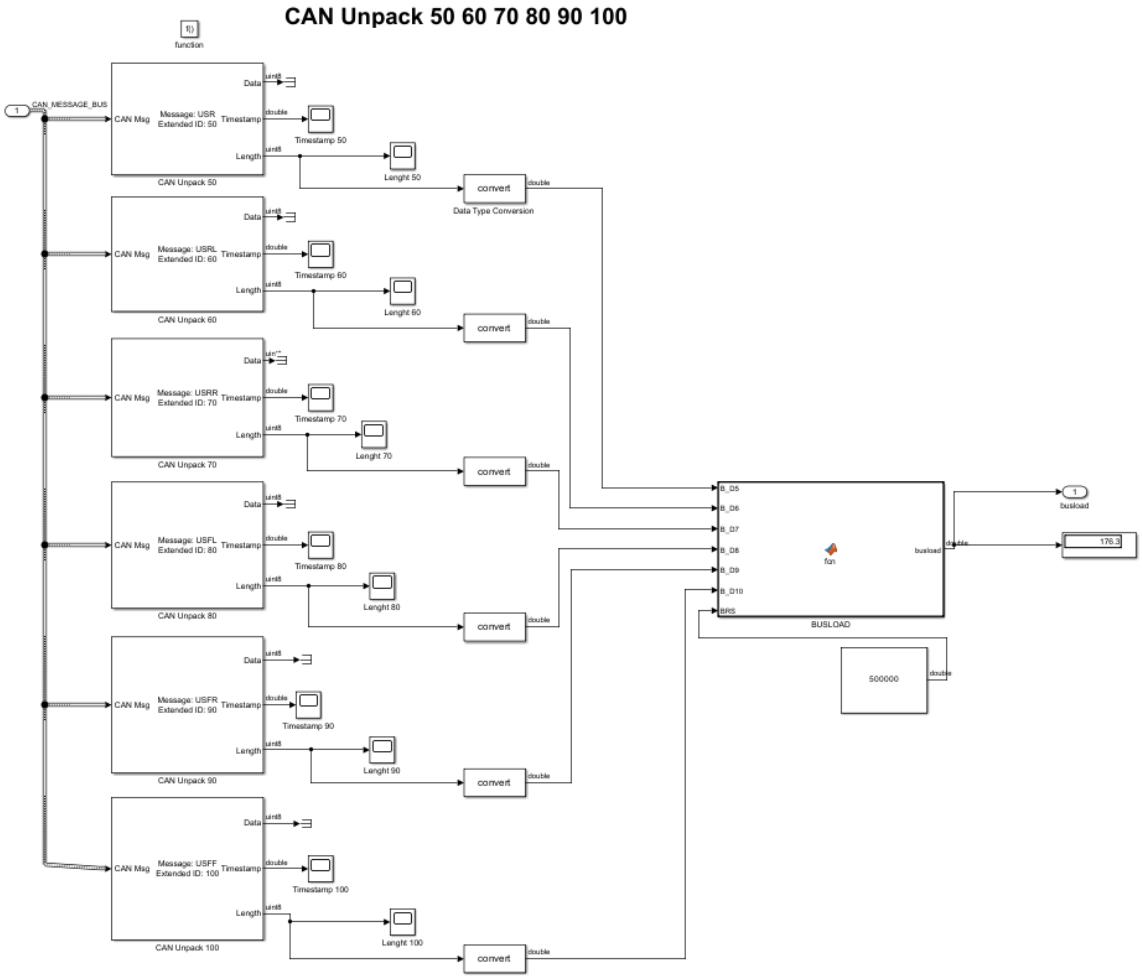
Fonte: elaborado pelo autor

Figura 31 – Arquitetura eletroeletrônica do *receiver* da rede CAN para o estudo de caso 2 no *Simulink* para os sensores de radar.



Fonte: elaborado pelo autor

Figura 32 – Arquitetura eletroeletrônica do *transmiter* da rede CAN para o estudo de caso 2 no *Simulink* para os sensores ultrasônicos.



Fonte: elaborado pelo autor

Tabela 17 – Parâmetros de configuração da rede CAN para o estudo de caso 2 no *Simulink*.

Tipo de mensagem	Velocidade de transmissão do *Acknoledgement Field*	Velocidade de transmissão do *Data Field*	Tempo de simulação
Periódica	500 kbps	2000 kbps	2 segundos

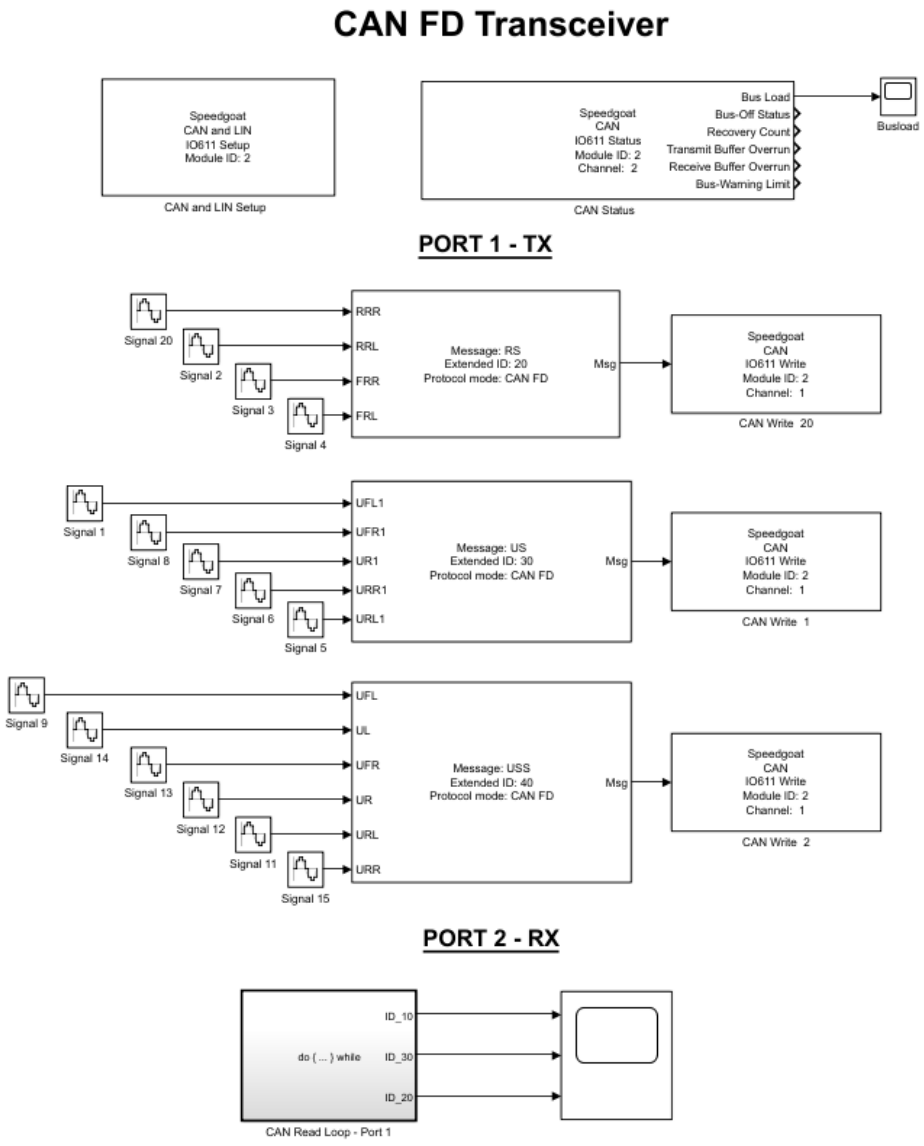
Tabela 18 – Definição do *frame* da rede CAN para o estudo de caso 2 no *Simulink*.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
10	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
20	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
30	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
40	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
50	4 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
60	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
70	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
80	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
90	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
100	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal

Após as simulações no *Simulink*, os modelos foram recriados, utilizando os blocos da biblioteca *Simulink Real-Time: Speedgoat I/O Blockset*, disponibilizados somente com a posse da licença. Para o CAN FD foi definido, pelo bloco *Block Parameters: CAN & LIN*

Setup, que a rede opera à 500 kbps no campo de *ackowledgement* e a taxa de transmissão do campo de dados é de 2000 kbps, com isso, tem-se os 15 sinais senoidais sendo entrada dos blocos *CAN FD Pack*, que encapsulam os dados nos *frames*, para que em seguida os dados encapsulados sejam enviados para os blocos *CAN FD Write*, a cada 5 milisegundos pela ECU "RS" e a 1 milisegundos pelas ECUs "US" e "USS", por fim, as mensagens são enviadas para os blocos *CAN FD Receiver*, de acordo com a Figura 33.

Figura 33 – Arquitetura eletroeletrônica do *transmiter* da rede CAN FD para o estudo de caso 2 embarcado.

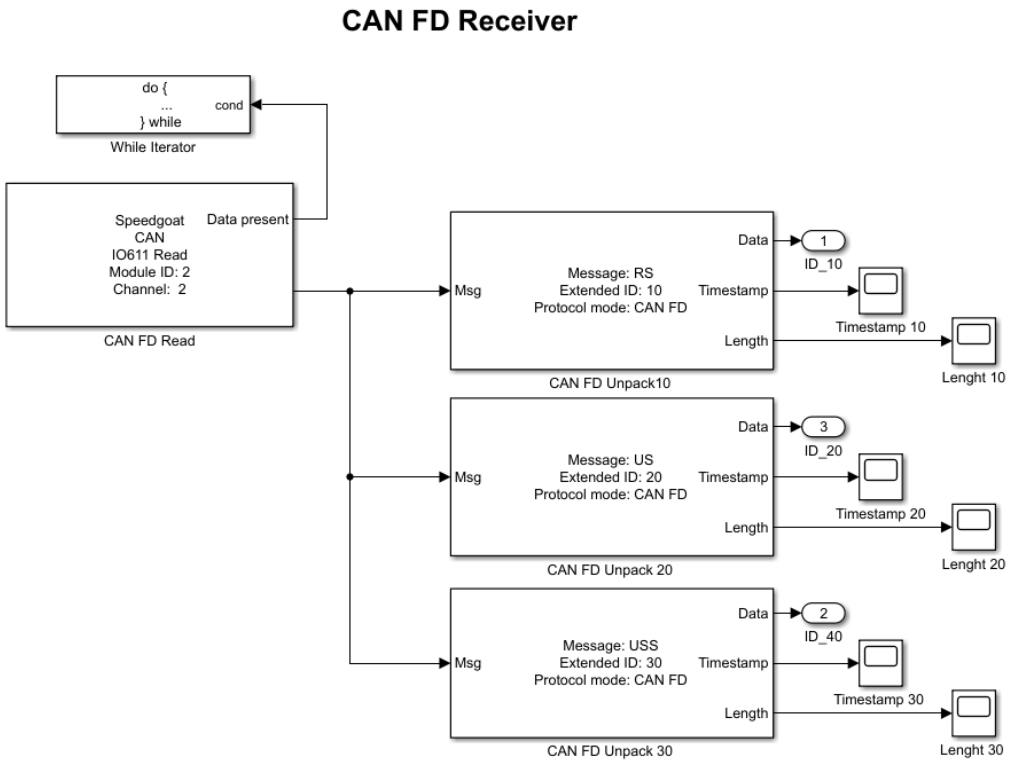


Fonte: elaborado pelo autor

Assim como na primeira parte deste estudo de caso, por conta do período de amostragem definido, cada ECU receptora recebe a mensagem da ECU transmissora que possuir o mesmo *sample time*. Logo, os dados encapsulados são então enviados para os blocos *CAN FD Unpack*, que se encontram dentro do bloco *CAN Read Loop - Port 1*, para desencapsular os dados e enviar para os *Scope* realizarem a leitura da latência, como

pode-se aferir na Figura 34. O comprimento é medido para aferir que todos os *bytes* foram enviados com sucesso. Ainda na Figura 33, tem-se o bloco *CAN Status* que mede o *busload* e projeta os valores no *Scope*. Pode-se aferir nas Tabelas 19 e 20 a configuração da rede e a configuração dos *frames* enviados por cada ECU, respectivamente.

Figura 34 – Arquitetura eletroeletrônica do *receiver* da rede CAN FD para o estudo de caso 2 embarcado.



Fonte: elaborado pelo autor

Tabela 19 – Parâmetros de configuração da rede CAN FD para o estudo de caso 2 embarcado.

Tipo de mensagem	Velocidade de transmissão do *Acknoledgement Field*	Velocidade de transmissão do *Data Field*	Tempo de simulação
Periódica	500 kbps	2000 kbps	2 segundos

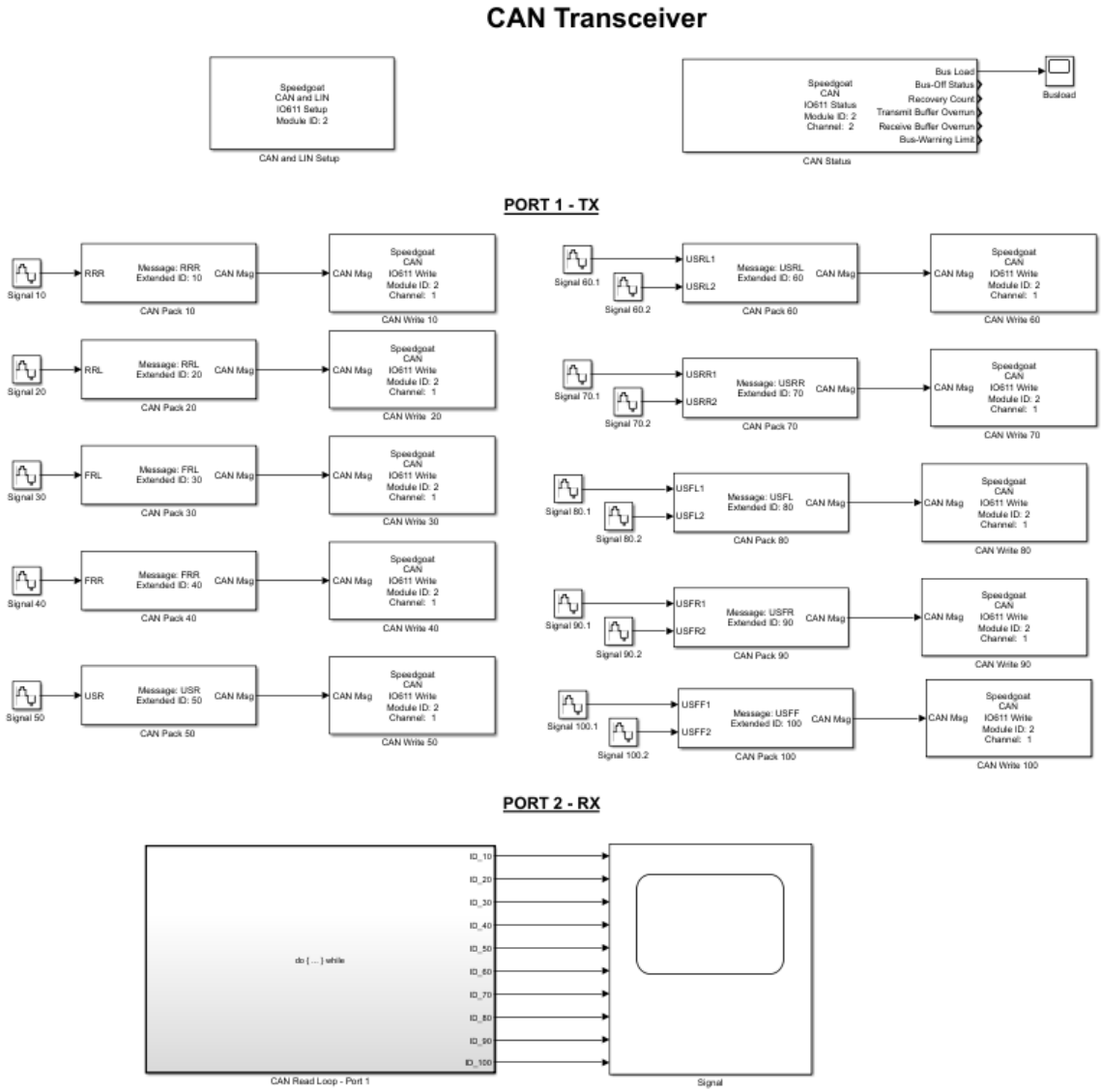
Tabela 20 – Definição do *frame* da rede CAN FD para o estudo de caso 2 embarcado.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
10	32 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
20	20 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
30	24 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal

Para o protocolo CAN, fez-se uso da mesma biblioteca e de alguns blocos da simulação do CAN FD, pois são compatíveis. Já os blocos *CAN Pack* e *CAN Unpack* foram substituídos pelos específicos do protocolo CAN. O bloco *Block Parameters: CAN & LIN*

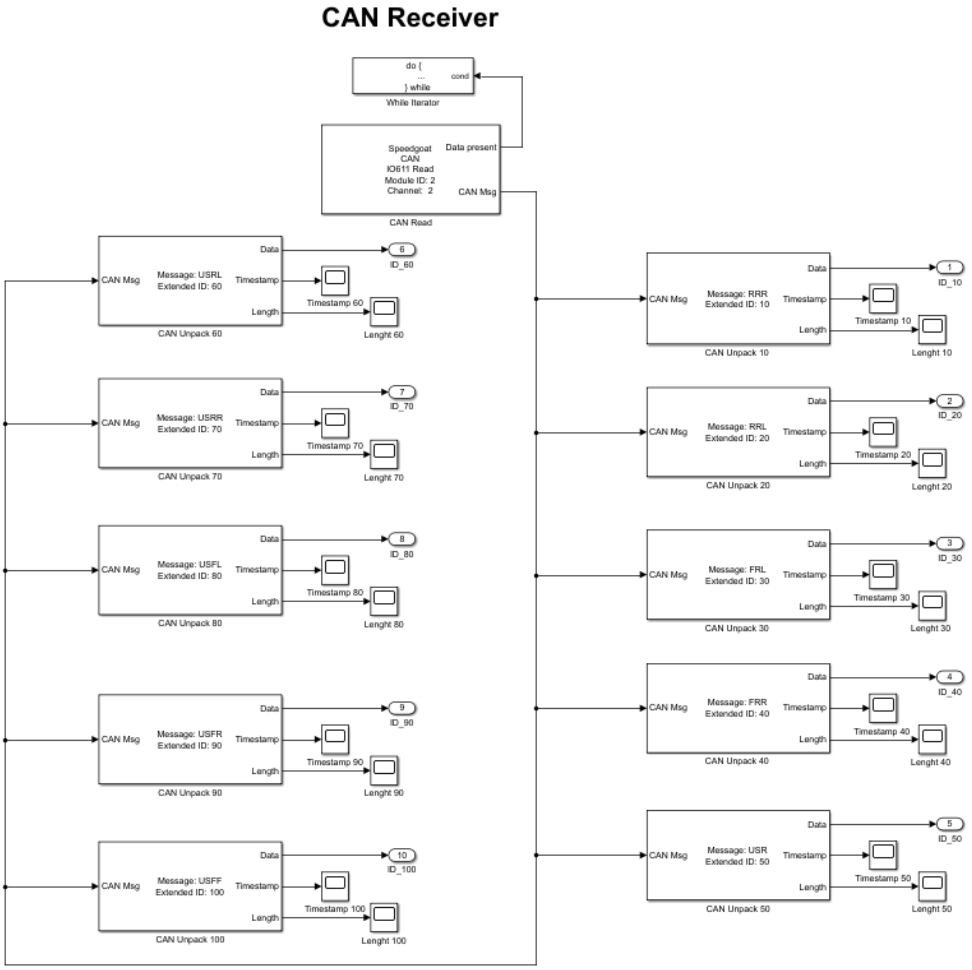
Setup foi configurado para operar à taxa de transmissão de 500 kbps para simulação, com isso, os sinais são encapsulados pelo bloco *CAN Pack*, que envia os dados encapsulados para o bloco *CAN Write*. As ECUs com os IDs 10, 20, 30 e 40 enviam os dados a cada 5 milisegundos, já as demais ECUs enviam a cada 1 milisegundos para o bloco *CAN Read*, que se encontra dentro do bloco *CAN Read Loop - Port 1*, como pode ser visto na Figura 35. Com isso, na Figura 36 pode-se ver que os dados são então enviados para os blocos *CAN Unpack*, que desencapsulam os sinais para a realização da medição da latência pelo *Scope*. Pode-se aferir nas Tabelas 21 e 22 a configuração da rede e a configuração dos *frames* enviados por cada ECU, respectivamente.

Figura 35 – Arquitetura eletroeletrônica do *transmiter* da rede CAN para o estudo de caso 2 embarcado.



Fonte: elaborado pelo autor

Figura 36 – Arquitetura eletroeletrônica do *receiver* da rede CAN para o estudo de caso 2 embarcado.



Fonte: elaborado pelo autor

Tabela 21 – Parâmetros de configuração da rede CAN para o estudo de caso 2 embarcado.

Tipo de mensagem	Velocidade de transmissão do *Acknoledgement Field*	Velocidade de transmissão do *Data Field*	Tempo de simulação
Periódica	500 kbps	2000 kbps	2 segundos

Tabela 22 – Definição do *frame* da rede CAN para o estudo de caso 2 embarcado.

ID	DLC	Período de amostragem	Número teórico de mensagens enviadas	Formato do frame	Tipo de sinal
10	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
20	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
30	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
40	8 bytes	5 milisegundos	400 mensagens	Estendido	Senoidal
50	4 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
60	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
70	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
80	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
90	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal
100	8 bytes	1 milisegundos	2000 mensagens	Estendido	Senoidal

3.4 Validação do modelo

A validação do modelo desempenha um papel crucial em garantir a confiabilidade e a credibilidade dos resultados obtidos neste estudo comparativo entre os protocolos CAN e CAN FD. Essa validação foi realizada por meio da comparação entre os resultados simulados em ambiente virtual, *Simulink*, e aqueles obtidos pelo modelo embarcado no *hardware Performance Real-Time Target Machine* fornecido pela empresa *Speedgoat*, possibilitando uma validação abrangente que abordou tanto as simulações computacionais quanto os testes em *hardware*. A comparação entre os resultados da simulação e os dados reais desse equipamento permitiu verificar a concordância entre o modelo e o comportamento real da rede automotiva, validando, assim, a precisão da rede em refletir o desempenho simulado dos protocolos CAN e CAN FD.

4 Resultados e análises

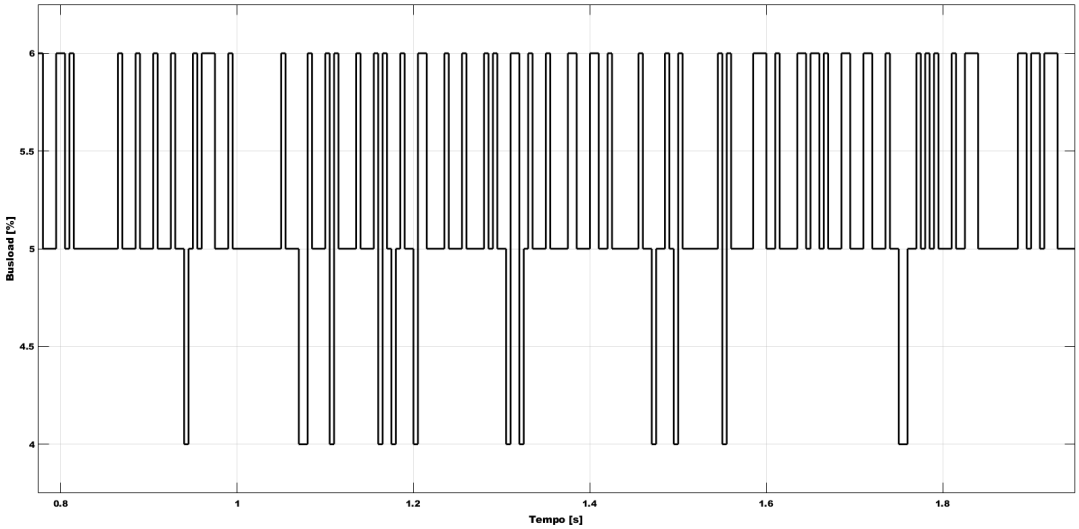
Neste capítulo, será apresentado de forma clara e concisa os resultados provenientes das simulações realizadas. Essa exposição será conduzida por meio da análise de estudos de caso, nos quais discorre-se sobre os resultados obtidos. Cada estudo de caso será apresentado em duas partes, sendo a primeira abordando as simulações em ambiente virtual e a segunda parte será relacionada a simulação embarcada em *hardware*.

4.1 Estudo de caso 1

Iniciando pelo protocolo CAN, utilizou-se da Equação 2.14 dentro do bloco *Matlab Function* para realizar os cálculos de *busload*, o qual, considerando *bit stuffing*, a rede obteve um valor de 6,41%. Para aferir a latência, foi gerado uma planilha com todos os valores obtidos pelo *Scope*, através do próprio *software Matlab/Simulink*. De acordo com os valores obtidos, o maior valor em que a ECU receptora aguarda pela mensagem é de 0,079 segundos, ou seja, 29 milisegundos de latência no pior cenário, considerando ciclos de 5 milisegundos. Além disso, teve-se 370 mensagens enviadas à rede, um valor bem próximo do esperado de 400 mensagens, o que era de se esperar, para uma simulação de 2 segundos com período de amostragem de 5 milisegundos, com isso, sendo transmitido 64 *bytes* de dados por mensagem, tem-se que o *throughput* foi de 11,84 kbps. Já o protocolo CAN FD, utilizou-se da Equação 2.15 para calcular o *busload* da rede, que obteve o valor de 3,26%, quase a metade do valor do protocolo CAN. O período de envio da mensagem no pior cenário foi de 0,069 segundos, ou seja, uma latência de 19 milisegundos, quase 35% a menos que o protocolo CAN. Por fim, houveram 398 mensagens enviadas ao barramento, devido ao baixo *busload* resultando em 12,74 kbps de *throughput*.

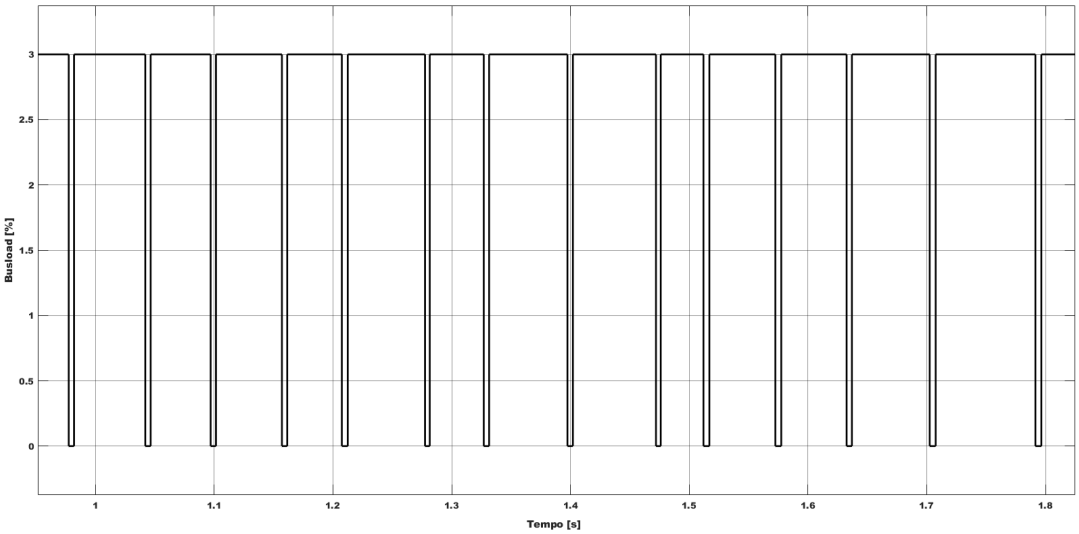
Após os resultados das simulações no *Simulink*, obteve-se os resultados a respeito das simulações embarcadas no *hardware* em tempo real, logo, o protocolo CAN obteve o valor de 6% de *busload* para o pior cenário, correspondente ao simulado no *Simulink*, como pode ser aferido pela Figura 37. A latência foi mínima, no valor de 44 microsegundos, obteve-se 195 mensagens enviadas à rede de 220 esperada, uma vez que a simulação começou aos 0,78 segundos e finalizou aos 1,98 segundos e o *throughput* de 10,40 kbps. Para o protocolo CAN FD, obteve-se o *busload* de 3% no pior caso, também correspondente ao *Simulink*, como pode ser visto na Figura 38. A latência foi de 32 microsegundos e a quantidade de mensagens foi de 206 mensagens para a simulação que iniciou-se aos 0,95 segundos e finalizou com 1,85 segundos, ou seja, todas as mensagens foram enviadas. O *throughput* foi de 14.65 kbps. Vale lembrar que a latência medida é a do pior cenário. Ambos os protocolos para as duas simulações não tiveram erro na transmissão dos *bits*.

Figura 37 – *Busload* da rede CAN do estudo de caso 1 embarcada.



Fonte: Autoral

Figura 38 – *Busload* da rede CAN FD do estudo de caso 1 embarcada.



Fonte: Autoral

Com isso, na Tabela 23 é possível realizar uma análise comparativa entre ambos os protocolos, no qual, para a simulação no *Simulink* o protocolo CAN FD teve 49,14% a menos de *busload*, 34,48% a menos de latência, para o pior cenário, 7,60% a mais de *throughput* e 7,57% a mais de mensagens enviadas ao barramento que o protocolo CAN. Para a simulação embarcada no *hardware Performance Real-Time Target Machine* o protocolo CAN FD obteve 50,00% a menos de *busload*, 27,27% a menos de latência, 40,87% a mais de *throughput* e 11,36% a mais de mensagens enviadas à rede que o protocolo CAN. Como esperado, o protocolo CAN FD performou melhor que o CAN, devido à mudança no *Baud Rate Prescaler* para o envio do campo de dados mais rápido. Para este estudo de caso, o BER foi 0.

Tabela 23 – Definição do *frame* do estudo de caso 2 para CAN embarcado.

HW/SW	Protocolo	***Busload***	Latência	Quantidade de mensagens	*Throughput*
Simulink	CAN	6,41%	29 ms	370/400	11,84 kbps
	CAN FD	3,26%	19 ms	398/400	12,74 kbps
Performance	CAN	6,00%	0,044 ms	195/220	10,40 kbps
	CAN FD	3,00%	0,032 ms	206/206	14,65 kbps

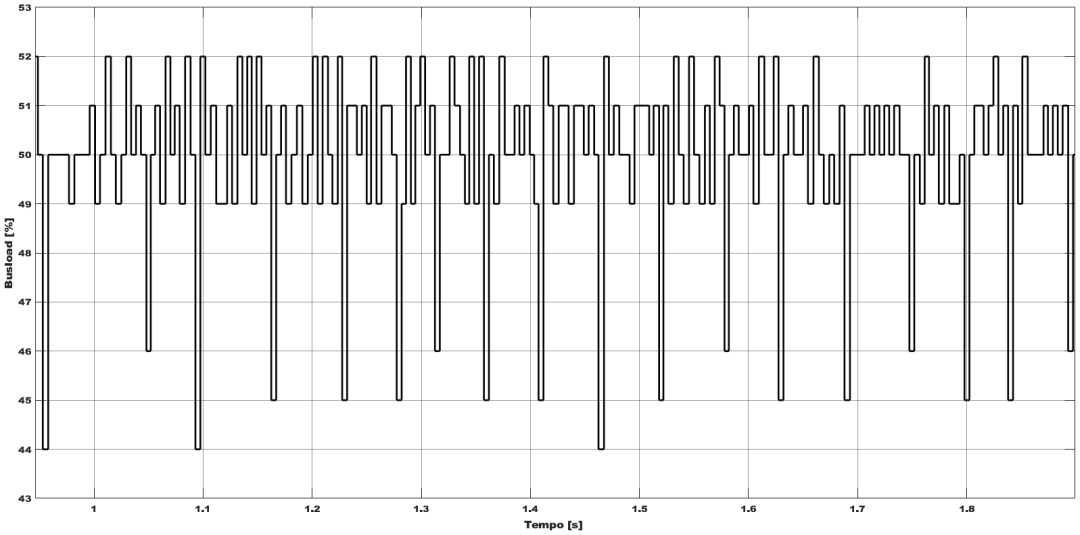
4.2 Estudo de caso 2

Como o estudo de caso 2 foi modelado com base na rede CAN FD de um veículo de luxo, será iniciado este tópico por ela. Com isso, da mesma forma que para o estudo de caso 1, utilizou-se da Equação 2.15 para calcular o valor do *busload* que foi de 52,68%. A latência foi aferida pelos dados da planilha gerada pelo *Scope*, e o maior tempo de espera para o pior cenário foi de 0,502 segundos, totalizando 501 milisegundos de latência, com isso, pode-se aferir também a quantidade de mensagens enviadas na rede, no caso em questão foram 8392 mensagens de 8400 no total, logo 8 mensagens não foram enviadas, totalizando 402,4 kbps de *throughput*. Para o protocolo CAN teve-se que o *busload* foi de 201,9%, aproximadamente quatro vezes maior que do protocolo CAN, a latência foi de 0,875 segundos para o pior cenário, ou seja, 874 milisegundos foi a maior latência. A quantidade de mensagens enviadas à rede foram 13576 de 13600, um total de 24 mensagens perdidas, totalizando 402,4 kbps de *throughput*. Para a segunda parte, que teve o modelo embarcado no *hardware Performance Real-Time Target Machine*, o protocolo CAN FD teve o *busload* de 52% devido ao *bit stuffing*, como pode ser aferido na Figura 39. A latência foi de 0,010 segundos no pior cenário, logo um tempo de espera de 9 milisegundo e a quantidade de mensagens foram 4044 de 4200 mensagens, devido ao período simulado pela máquina ter sido de aproximadamente 1 segundo, logo o *throughput* foi de 776,5 kbps. Para o protocolo CAN, o *busload* foi de 99%, justificável pela simulação ter alcançado valores acima de 200%, de acordo com a Figura 40. A latência do teste foi de 0,069 segundos no pior cenário, ou seja, 68 milisegundos de *delay* para todas as mensagens, o que acarretou no envio de 4126 mensagens de um total de 6800, totalizando 635,3 kbps.

Com isso, através da Tabela 24, é possível comparar ambos os protocolos. Para a rede modelada, na qual foi inspirada em uma rede CAN FD, pode-se observar que a mesma está sendo bastante utilizada, mais que metade da sua capacidade total, entretanto, ao reproduzí-la para o protocolo CAN, a rede não suporta a quantidade de dados transitando pelo barramento. Logo, comparando os dados, o CAN FD simulado no *Simulink* possui 73,91% a menos de *busload*, 42,68% a menos de latência no pior cenário e 5184 mensagens a menos na rede que o protocolo CAN, porém o *throughput* é o mesmo para ambos, assim como o BER. Vale ressaltar que para a mesma quantidade de *bytes* enviados, a rede CAN FD envia 38,19% menos mensagens que o CAN. Já a simulação embarcada no *hardware*

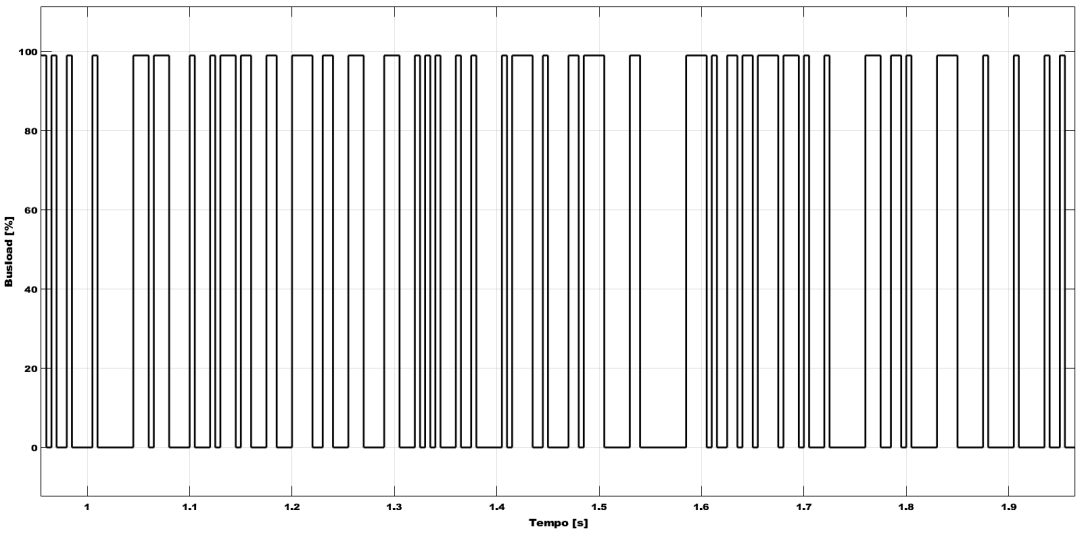
o CAN FD obteve 47,47% a menos de *busload*, 86,76% a menos de latência, 22,23% a mais de *throughput* e, com relação a quantidade de mensagens enviadas, o CAN FD teve 96,29% dos *frames* enviados, enquanto o protocolo CAN somente 60,71% das mensagens foram recebidas, além de que o protocolo CAN FD enviou 2,03% a menos de mensagens que o CAN, para o mesmo período de tempo, demonstrando sua eficiência na transmissão massiva de dados. O BER para ambos permaneceu em 0.

Figura 39 – *Busload* da rede CAN FD do estudo de caso 2 embarcado.



Fonte: Autoral

Figura 40 – *Busload* da rede CAN do estudo de caso 2 embarcado.



Fonte: Autoral

Tabela 24 – Definição do *frame* do estudo de caso 2 para CAN embarcado.

HW/SW	**Protocolo**	***Busload***	**Latência**	**Quantidade de mensagens**	*Throughput*
Simulink	CAN	201,9%	874 ms	13576/13600	402,4 kbps
	CAN FD	52,68%	501 ms	8392/8400	402,4 kbps
Performance	CAN	99,00%	68 ms	4128/6800	635,3 kbps
	CAN FD	52,00%	9 ms	4044/4200	776,5 kbps

5 Conclusão

A presente pesquisa teve como objetivo realizar uma modelagem e análise comparativa das redes automotivas CAN e CAN FD utilizando o *software Simulink* e o *hardware Performance Real-Time Target Machine* da *Speedgoat*. A principal indagação que orientou este estudo foi qual protocolo apresentaria a melhor performance em uma rede de veículo com quantidades massivas de dados. Os resultados obtidos por meio de dois estudos de caso revelaram *insights* valiosos sobre as características e eficiências dos protocolos estudados.

No primeiro estudo de caso, que envolveu uma ECU transmissora e uma ECU receptora, demonstrou os benefícios do protocolo CAN FD. Na simulação em ambiente virtual, observou-se uma redução de 49,14% no busload, 34,48% na latência, 7,60% a mais de *throughput* e 28 mensagens a mais do que o protocolo CAN. Ao ser implementado no *hardware*, o CAN FD manteve sua superioridade em relação ao CAN, exibindo uma redução de 50,00% no busload, uma diminuição de 27,27% na latência, 40,87% a mais de *throughput* e uma taxa de recebimento de mensagens 12,82% maior do que o CAN.

O segundo estudo de caso, baseado em uma rede automotiva real, solidificou ainda mais os benefícios do protocolo CAN FD. Na simulação em ambiente virtual, a rede CAN FD apresentou uma redução notável de 73,91% no busload, 42,68% na latência e uma eficiência impressionante ao enviar 5184 mensagens a menos do que o protocolo CAN para a mesma quantidade de bytes. Ao implementar no *hardware*, o CAN FD continuou a demonstrar superioridade com relação ao CAN, com uma redução de 47,47% no busload, redução de 86,76% na latência, aumento de 22,23% no *throughput* e a quantidade de mensagens recebidas sendo 35,58% maior em relação ao CAN. Vale ressaltar que o CAN FD enviou mais dados em menos mensagens ao barramento que o protocolo CAN.

Estes resultados, em conformidade com a literatura e respaldados por equações teóricas, confirmam a eficácia do CAN FD em ambientes automotivos, especialmente durante transmissões massivas de dados. As métricas de desempenho avaliadas neste estudo estão alinhadas com as pesquisas referenciadas no tópico "estado da arte", validando assim a contribuição desta tese para o campo. O uso de uma rede modelada com base na de um veículo de luxo real fortalece a conclusão de que o CAN FD é ideal para a conexão de sensores, enquanto o protocolo CAN é mais adequado para funções relacionadas ao conforto e redes não cíclicas, sendo também uma escolha eficaz para funções com *trigger* em sistemas automotivos modernos. Esta constatação ajuda a explicar por que o protocolo CAN continua a ser empregado nos dias de hoje. Como sugestão para pesquisas futuras, recomenda-se a análise da mesma rede ou de uma mais complexa, implementando uma

rede mista entre CAN e CAN FD. Isso permitiria avaliar se uma configuração híbrida apresentaria melhor desempenho do que uma rede exclusivamente CAN FD.

Referências

AN, H. S.; JEON, J. W. Analysis of can fd to can message routing method for can fd and can gateway. In: IEEE. *2017 17th International Conference on Control, Automation and Systems (ICCAS)*. [S.l.], 2017. p. 528–533. Citado na página 49.

ANDRADE, R. d. *Sistemas de comunicação CAN FD: modelamento por software e análise temporal.* Tese (Doutorado) — Universidade de São Paulo, 2014. Citado 8 vezes nas páginas 28, 29, 40, 42, 43, 44, 45 e 48.

ANDRADE, R. de et al. A didactic platform to study of can fd bus. In: IEEE. *2013 IEEE Global Engineering Education Conference (EDUCON)*. [S.l.], 2013. p. 318–323. Citado na página 25.

BORTH, T. F. Analisando os impactos do uso do protocolo can fd em aplicações automotivas: estudo de caso. 2016. Citado 3 vezes nas páginas 30, 39 e 48.

BOSCH, C. Can specifications. *Robert Bosch GmbH, Postfach*, v. 50, 1991. Citado na página 33.

BOZDAL, M.; SAMIE, M.; JENNIONS, I. A survey on can bus protocol: Attacks, challenges, and potential solutions. In: *2018 International Conference on Computing, Electronics Communications Engineering (iCCECE)*. [S.l.: s.n.], 2018. p. 201–205. Citado na página 22.

CHEON, B.; JEON, J. W. The can fd network performance analysis using the canoe. In: *IEEE ISR 2013*. [S.l.: s.n.], 2013. p. 1–5. Citado na página 48.

Dedicated System. *Speedgoat Performance Real-Time Target Machine*. 2023. Disponível em: <https://dedicatedsystems.com.au/products/performance-real-time-target-machine/>. Acesso em: 23 de outubro 2023. Citado na página 46.

DUARTE, J. V. C.; FIGUEIREDO, P. H. L. Desenvolvimento de arquitetura eletroeletrônica para veículo de pequena escala. 2023. Citado 3 vezes nas páginas 25, 26 e 32.

GODOY, E. P. *Desenvolvimento de uma ferramenta de análise de desempenho de redes CAN (Controller Area Network) para aplicações em sistemas agrícolas.* Tese (Doutorado) — Universidade de São Paulo, 2007. Citado 3 vezes nas páginas 43, 45 e 47.

GUIMARÃES, A. d. A. *Análise da norma ISO11783 e sua utilização na implementação do barramento do implemento de um monitor de semeadora.* Tese (Doutorado) — Universidade de São Paulo, 2003. Citado 2 vezes nas páginas 26 e 31.

GUIMARÃES, A. de A. *Eletrônica embarcada automotiva*. [S.l.]: Érica, 2007. Citado na página 30.

HARTWICH, F. et al. Can with flexible data-rate. In: CITESEER. *Proc. iCC*. [S.l.], 2012. p. 1–9. Citado na página 22.

LABAYRADE, R.; ROYERE, C.; AUBERT, D. A collision mitigation system using laser scanner and stereovision fusion and its assessment. In: IEEE. *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* [S.l.], 2005. p. 441–446. Citado na página 21.

MILANÉS, V. et al. A fuzzy aid rear-end collision warning/avoidance system. *Expert Systems with Applications*, Elsevier, v. 39, n. 10, p. 9097–9107, 2012. Citado na página 21.

MUTTER, A.; HARTWICH, F. Advantages of can fd error detection mechanisms compared to classical can. *CAN in Automation iCC*, 2015. Citado na página 37.

NATIONS, U. Resolution adopted by the general assembly on 31 august 2020. New York, 2020. Disponível em: <https://digitallibrary.un.org/record/3879711/files/A_RES_74_299-EN.pdf>. Citado na página 21.

NAVET, N. et al. Trends in automotive communication systems. *Proceedings of the IEEE*, v. 93, n. 6, p. 1204–1223, 2005. Citado na página 25.

PARET, D. *Multiplexed networks for embedded systems: CAN, LIN, flexray, safe-by-wire...* [S.l.]: John Wiley & Sons, 2007. Citado 8 vezes nas páginas 27, 28, 29, 32, 33, 37, 38 e 39.

PUNNEKKAT, S.; HANSSON, H.; NORSTROM, C. Response time analysis under errors for can. In: IEEE. *Proceedings Sixth IEEE Real-Time Technology and Applications Symposium. RTAS 2000.* [S.l.], 2000. p. 258–265. Citado na página 43.

RENAEST. *Panorama dos acidentes*. 2023. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/arquivos-senatran/docs/renaest>. Citado na página 21.

SALUNKHE, A. A.; KAMBLE, P. P.; JADHAV, R. Design and implementation of can bus protocol for monitoring vehicle parameters. In: *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT).* [S.l.: s.n.], 2016. p. 301–304. Citado na página 22.

SILVA, I. L. da. *Barramento CAN entre Arduinos Uno*. 2019. Acessado em: 26 de dezembro de 2023. Disponível em: <https://embarcados.com.br/barramento-can-entre-arduinos-uno/>. Citado na página 31.

SILVA, R. R. d. Modelagem e análise de redes automotivas em ambiente virtual. 2015. Citado 4 vezes nas páginas 30, 31, 42 e 48.

SOUSA, R. V. d. *CAN (Controller Area Netwoork): uma abordagem para automação e controle na área agrícola*. Tese (Doutorado) — Universidade de São Paulo, 2002. Citado na página 27.

SPICER, R. et al. Field effectiveness evaluation of advanced driver assistance systems. *Traffic injury prevention*, Taylor & Francis, v. 19, n. sup2, p. S91–S95, 2018. Citado na página 22.

TINDELL, K.; BURNS, A. Guaranteed message latencies for distributed safety-critical hard real-time control networks. *Dept. of Computer Science, University of York*, Citeseer, 1994. Citado 3 vezes nas páginas 41, 42 e 45.

VECTOR. *CAN FD Introduction*. 2013. Urlhttps://cdn.vector.com/cms/content/know-how/can/Slides/CAN$_F$D$_I$ntroduction$_E$N.pdf. Citado 3 vezes nas páginas 34, 35 e 36.

WHO. *World Health Organization - Road traffic injuries*. 2022. Disponível em: <https://www.who.int/en/news-room/fact-sheets/detail/road-traffic-injuries>. Citado na página 21.

WU, C. et al. A method of vehicle motion prediction and collision risk assessment with a simulated vehicular cyber physical system. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 47, p. 179–191, 2014. Citado na página 21.

XIE, Y. et al. Comparison between can and can fd: A quantified approach. In: IEEE. *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. [S.l.], 2017. p. 1399–1403. Citado na página 49.

ZAGO, G. M.; FREITAS, E. P. de. A quantitative performance study on can and can fd vehicular networks. *IEEE Transactions on Industrial Electronics*, IEEE, v. 65, n. 5, p. 4413–4422, 2017. Citado na página 49.