CAN-ESP: Rede CAN de Baixo Custo para Veículos Elétricos

Danilo Moura Pereira[1,2], Roberto Rodrigues-Filho[3], Vinícius P. Gonçalves[1]
Hélio L. dos Santos[2], Roque M. P. Trindade[2], Rodolfo Meneguette[4]
André L. Marques Serrano[1], Geraldo P. Rocha Filho[2]

[1]Universidade de Brasília (UnB)
[2]Universidade Estadual do Sudoeste da Bahia (UESB)
[3]Universidade Federal de Santa Catarina (UFSC)
[4]Universidade de São Paulo (USP)

moura.danilo@aluno.unb.br, roberto.filho@ufsc.br, vpgvinicius@unb.br
{heliosantos, roquetrindade}@uesb.edu.br, meneguette@icmc.usp.br
andrelms@unb.br, geraldo.rocha@uesb.edu.br

Abstract. *Advancements in embedded electronics have heightened the demand for efficient, robust, and cost-effective communication networks in vehicles. The Controller Area Network (CAN) protocol has emerged as the dominant standard for communication among Electronic Control Units (ECUs), largely due to its reliability and real-time performance. However, state-of-the-art solutions face challenges such as high costs, limited flexibility for remote updates, and the absence of native wireless connectivity. Therefore, this paper introduces and evaluates CAN-ESP, a low-cost CAN network based on the ESP32 microcontroller and the TWAI protocol. The proposed solution not only supports efficient vehicular communication but also enables remote firmware updates (OTA). Our experiments demonstrate that CAN-ESP operates reliably, with a bus load ranging from 10% to 25%, a frame retransmission rate of 0.3 frames per million, and an average latency between 129 μs and 141 μs, thereby confirming its feasibility for electric vehicle applications, completely complying with ISO 11898.*

1. Introdução

No início dos anos 1980, uma parceria entre duas gigantes da indústria dos eletrônicos (Bosch e Intel) introduziu ao mercado o protocolo de comunicação CAN (Controller Area Network). O objetivo principal do projeto era fornecer suporte a aplicativos confiáveis em veículos [Lawrenz 2013]. Segundo a Bosch, CAN é um protocolo de comunicação serial que pode efetivamente suportar redes com controle distribuído, em tempo real e com alto nível segurança. [GmbH 1991]. Desde 1990, quando a Mercedes Benz se tornou o primeiro fabricante de automóveis a incorporar o CAN em um veículo de série (Classe S) para a integração da eletrônica da carroceria, esse protocolo tem sido amplamente adotado como uma tecnologia de comunicação robusta e confiável. Essa aplicação estende-se não apenas ao setor automotivo, mas também abrange áreas como automação industrial, aviônicos, robótica, indústria aeroespacial e aplicações militares, atestando sua versatilidade e eficácia em diversos contextos [Lawrenz 2013, Ribeiro Jr et al. 2023, Valentini et al. 2023]. Apesar de ter mais de 50 anos de existência e da introdução de diversas outras tecnologias para comunicação veicular, como LIN-Bus, FlexRay, ModBus, MOST, Ethernet, etc., o Barramento CAN (CAN

BUS) mantém sua ampla utilização em veículos de todos os tipos. A popularidade do protocolo CAN pode ser atribuída ao seu custo acessível e a outras características vantajosas, incluindo alta confiabilidade em ambientes suscetíveis a interferências e o sistema de arbitragem de barramento baseado em prioridade [Zuberi and Shin 1996].

Apesar das vantagens do protocolo CAN, algumas implementações apresentam desafios como custos elevados, complexidade na integração e ausência de suporte para atualização remota de firmware (Over-the-Air – OTA). Trabalhos anteriores, como os de [Li et al. 2010], [Desai et al. 2013] e [Ismail et al. 2015], demonstraram implementações funcionais de redes CAN, porém sem contemplar recursos essenciais, como suporte a OTA, adoção de microprocessadores com conectividade Wi-Fi integrada e conformidade total com o padrão ISO 11898-1. Além disso, estudos como os de [Salunkhe et al. 2016] e [Chikhale 2018] propuseram arquiteturas baseadas em plataformas de custo mais elevado, como Raspberry Pi, limitando sua viabilidade em projetos de baixo custo.

Diante desse cenário, este trabalho apresenta e valida a CAN-ESP, uma rede CAN de baixo custo baseada no System-on-Chip (SoC) ESP32. A solução oferece uma abordagem modular e escalável, alinhada às exigências da indústria automotiva, ao mesmo tempo em que possibilita a implementação de funcionalidades avançadas, como atualizações OTA (Over-The-Air) através de uma rede Wi-Fi mesh. A arquitetura desenvolvida adota um barramento CAN 2.0B (extended frame - 29 bits) operando a uma velocidade de 1 Mbps, utilizando o protocolo TWAI (Two-Wire Automotive Interface) nativo da ESP-IDF, garantindo conformidade com o padrão ISO 11898 assegurando a compatibilidade com sistemas CAN já consolidados no setor automotivo.

A utilização do ESP32 como núcleo da rede CAN-ESP abre novas possibilidades para aprimorar a eficiência e a flexibilidade na comunicação veicular. Diferentemente de implementações convencionais, que utilizam microcontroladores dedicados sem conectividade integrada, o ESP32 possibilita a fusão entre a rede CAN e infraestrutura sem fio (Wi-Fi e bluetooth), permitindo a adição de novos serviços à arquitetura, com a otimização da transmissão de dados sem necessidade de interfaces adicionais, agregando flexibilidade e adaptabilidade às novas demandas da indústria automotiva. Dessa forma, a CAN-ESP não apenas viabiliza uma solução de baixo custo e alto desempenho para veículos elétricos, como também estabelece um caminho promissor para futuras aplicações em mobilidade inteligente e infraestrutura veicular conectada.

Este artigo esta estruturado da seguinte forma: a Seção 2 discute trabalhos relacionados, a Seção 3 detalha a solução proposta, a Seção 4 apresenta os testes de desempenho e a Seção 5 traz a conclusão e perspectivas futuras.

2. Trabalhos Relacionados

A implementação de redes CAN tem sido amplamente investigada na literatura, especialmente no contexto de aplicações automotivas e sistemas embarcados. Diversos estudos exploram soluções para aprimorar a comunicação entre módulos eletrônicos em veículos, destacando vantagens como a redução da complexidade da fiação e o aumento da confiabilidade da transmissão de dados [Lawrenz 2013]. No entanto, muitos desses trabalhos apresentam limitações que restringem sua aplicabilidade em veículos elétricos e em sistemas automotivos modernos, além de não atenderem a critérios de baixo custo. A Tabela 1 mostra um comparativo entre o CAN-ESP e outras soluções com o mesmo escopo.

Tabela 1. Características dos trabalhos relacionados

Trabalhos / Características	ISO 11898-1	Redes Distribuídas	Aplicada em EV	Atualização OTA	Wi-Fi & Bluetooth	Open Source	Baixo Custo	Métricas Quantitativas
Li et al., 2010	+	+	+	–	–	–	+	–
Desai et al., 2013	+	+	+	–	–	–	+	–
Ismail et al., 2014	–	+	–	–	–	–	+	–
Vijaya et al., 2015	+	+	–	–	–	–	+	–
Salunkhe et al., 2016	+	+	–	–	–	–	–	–
Wagh et al., 2017	+	+	–	–	–	–	+	–
Chikhale, 2018	–	+	–	–	–	–	–	–
Wang, 2021	–	+	+	–	–	–	+	–
Alzahrani et al., 2023	+	+	+	–	–	–	+	–
CAN-ESP (este trabalho)	+	+	+	+	+	+	++	+

Nota: '+' possui característica; '–' não possui característica; '++' o CAN-ESP possui um custo significativamente menor.

[Li et al. 2010] propõem um protocolo de camada de aplicação SAE J1939 voltado para a comunicação em redes CAN em veículos elétricos. O trabalho propõe uma arquitetura robusta para a troca de informações entre os módulos do veículo, mas não aborda aspectos essenciais, como a conformidade com o padrão ISO 11898 e a implementação de atualizações remotas de firmware (OTA). Além disso, a ausência de um microprocessador com conectividade Wi-Fi nativa restringe a possibilidade de integração com outras redes e dispositivos externos ao barramento, o que pode elevar os custos de manutenção e atualização do sistema.

[Desai et al. 2013] investigam o uso do protocolo CAN para melhorar a eficiência da comunicação em veículos inteligentes, enfatizando a escalabilidade do sistema e a facilidade de adição de novos módulos. Entretanto, o estudo não considera a aplicação em veículos elétricos nem a necessidade de conformidade com normas vigentes. Além disso, não há menção a algum mecanismo de atualização do software embarcado remotamente, o que compromete a manutenção e a escalabilidade do sistema, além de não apresentar uma solução de baixo custo.

[Ismail et al. 2015] apresentam um projeto de barramento CAN para veículos híbridos elétricos, utilizando microcontroladores ARM para otimizar o controle em tempo real. Apesar das contribuições para a integração de dispositivos, o estudo carece de detalhes sobre a conformidade com padrões estabelecidos pela indústria e não explora a possibilidade de atualizações remotas, dificultando sua adoção em automóveis modernos. Além disso, a utilização de hardware relativamente caro (se comparado ao ESP32) pode limitar sua adoção em aplicações que demandam baixo custo de implementação.

Outros trabalhos, como os de [Vijaya et al. 2015] e [Salunkhe et al. 2016], também exploram implementações baseadas em CAN, com ênfase na confiabilidade da comunicação e na integração de sensores. No entanto, essas soluções não contemplam a aplicação específica em veículos elétricos nem a possibilidade de atualização remota de firmware. Além disso, o custo dos componentes empregados, como o Raspberry Pi, pode limitar sua viabilidade em projetos de baixo custo, quando comparado a soluções baseadas no ESP32, cujo preço médio é de aproximadamente USD 5.

[Wagh et al. 2017] propõem um sistema de controle de velocidade veicular baseado em CAN, com a incorporação de sensores ultrassônicos para prevenção de colisões. Apesar de contribuir para a segurança veicular, o estudo não menciona a conformidade

com os padrões internacionais nem a possibilidade de implementação de atualizações remotas, limitando sua aplicabilidade em um contexto mais amplo. Além disso, o uso de hardware mais caro pode inviabilizar sua adoção em projetos comerciais.

[Chikhale 2018] revisa o protocolo CAN em aplicações automotivas, abordando tendências e desafios, mas carece de análise experimental ou validação prática, limitando-se a discussões teóricas. Não explora aspectos técnicos como latência e confiabilidade, o que dificulta a adoção em aplicações reais. Já [Wang 2021] discute um sistema de acionamento elétrico para veículos baseado em CAN, destacando a redução de cabeamento e robustez, mas não detalha conformidade com normas automotivas ou a possibilidade de atualizações remotas, além de não ser uma solução de baixo custo.

[Alzahrani et al. 2023] propõe a integração dos protocolos SAE J1939 e Modbus para comunicação entre o sistema de gerenciamento de bateria (BMS) e o controlador de carregador de um veículo elétrico, utilizando o Arduino UNO. A solução, de baixo custo, foi validada com módulos CAN (MCP2515) e Modbus (MAX485) e software PCAN View. No entanto, apresenta limitações como a falta de conformidade com o padrão ISO 11898, essencial para interoperabilidade e segurança, e não contempla atualizações remotas. Além disso, o uso do Arduino UNO restringe o desempenho em comparação a microprocessadores modernos, como o ESP32.

Embora os trabalhos relacionados discutam aspectos funcionais e estruturais, como conformidade com padrões, arquitetura distribuída e baixo custo, nenhum apresenta um conjunto completo de métricas quantitativas. Este estudo se diferencia ao oferecer uma avaliação experimental detalhada do CAN-ESP, com medições de latência, carga do barramento, taxa de retransmissão e colisões, possibilitando uma análise objetiva do desempenho. Para superar as limitações anteriores, o CAN-ESP integra o ESP32, um SoC de baixo custo com Wi-Fi e Bluetooth, adotando o padrão ISO 11898 para compatibilidade automotiva e implementando atualizações OTA. Além de reduzir custos ao substituir hardware mais caro por um microcontrolador eficiente, a solução mantém a confiabilidade e a performance da comunicação veicular, disponibilizando seu código-fonte como open source para fomentar a inovação no setor.

3. CAN-ESP: Solução Proposta

3.1. Visão Geral da Arquitetura

O CAN-ESP [1] implementa uma rede CAN de baixo custo para aplicações em veículos elétricos, utilizando o SoC ESP32 WROOM-32 e o protocolo TWAI , nativo da arquitetura ESP. A solução adota os padrões ISO 11898 [ISO 2015], garantindo compatibilidade com redes CAN convencionais. Além disso, o sistema incorpora suporte a atualizações remotas de firmware (OTA), proporcionando maior flexibilidade e facilidade de manutenção.

A arquitetura proposta do CAN-ESP (Figura 1) segue o modelo distribuído da norma ISO 11898, permitindo a comunicação direta entre múltiplas Unidades de Controle Eletrônico (ECUs) sem a necessidade de um controlador central. Ainda, o CAN-ESP adota um barramento CAN (versão 2.0B) de alta velocidade, operando a 1 Mbps.

[1] Endereço do repositório: https://github.com/danilo-moura-pereira/CAN-ESP

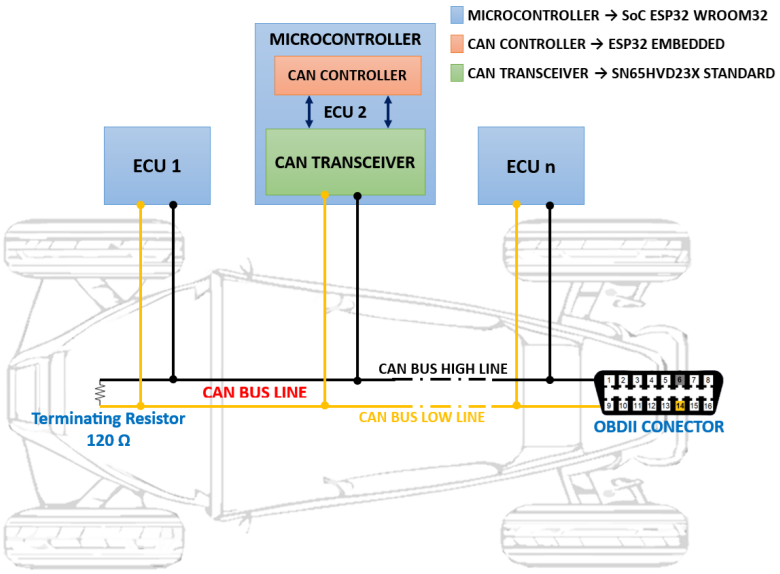

Figura 1. Arquitetura Distribuída Proposta

3.2. Componentes de Hardware

O hardware foi projetado utilizando componentes de baixo custo e amplamente disponíveis, garantindo a viabilidade econômica e a reprodutibilidade da solução. O barramento conecta cinco ECUs: *(i)* Controle do Motor (gerencia os motores elétricos); *(ii)* Controle da Aceleração (processa comandos do pedal); *(iii)* Controle do Freio (gerencia o sistema de frenagem, incluindo regeneração); *(iv)* Controle da Direção (atua sobre o motor da direção elétrica); e *(v)* Monitoramento (exibe informações no display TFT LCD, registra parâmetros da rede CAN e distribui atualizações via OTA através de rede Wi-Fi mesh). A Figura 2 apresenta a arquitetura e a integração dos componentes com o barramento.

A lista dos principais componentes utilizados na solução inclui os seguintes itens:

- **ESP32 WROOM-32**: System-on-Chip (SoC) equipado com um microprocessador Xtensa® Dual-Core 32-bit LX6, com clock de até 240 MHz, 520 KB de RAM e 4 MB de memória flash. Possui controlador CAN 2.0B integrado.
- **SN65HVD230**: Transceptor CAN responsável pela comunicação entre o microcontrolador e o barramento.
- **Cabo UTP CAT-5 (24 AWG)**: Utilizado para a conexão física do barramento CAN, interligando as ECUs.
- **Resistores de terminação de 120Ω**: Instalados nas extremidades do barramento CAN para reduzir reflexões de sinal e garantir a integridade da comunicação, conforme especificado pela norma [ISO 2015].

Para tornar funcional a integração das ECUs à rede, foram utilizados outros componentes de hardware e software, sensores e atuadores:

- **Motores elétricos**: Dois motores brushless de 8 polegadas, 36V e 350W, comumente utilizados em hoverboards.
- **Controladoras de motores**: Duas controladoras brushless de 36V, 18A e 350W, responsáveis pelo acionamento dos motores elétricos.

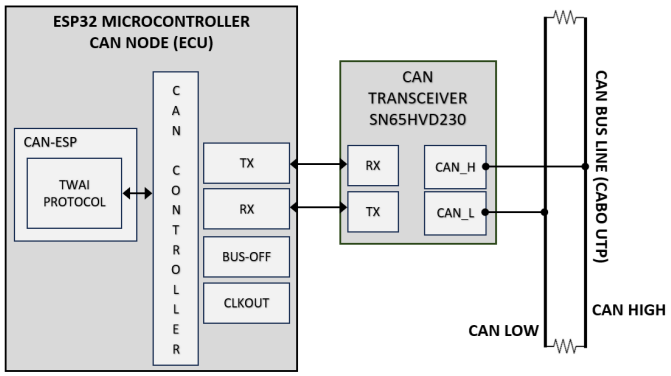

Figura 2. Conexão de uma ECU CAN-ESP ao barramento CAN

- **Acelerador**: Pedal eletrônico original do Volkswagen UP, modelo 2020.
- **Sensor de freio eletrônico**: Sensor Hall 49E-1378G acoplado a um ímã de neodímio (8mm x 4mm) para detecção da ativação do freio.
- **Motor da direção elétrica**: Motor universal de vidro elétrico, 12V, adaptado para o controle assistido da direção.
- **Velocímetro**: Captação do sinal de rotação disponível nas controladoras dos motores elétricos para medição da velocidade.
- **Display TFT LCD 1.8" SPI**: Tela compacta para exibição de informações do sistema em tempo real.
- **Placa CANABLE V2.0 nano USB**: Interface para conexão de um PC à rede CAN, permitindo a coleta e análise de dados via software Cangaroo (versão 0.2.3).
- **Analisador lógico USB SCM 24 MHz, 8 canais**: Dispositivo para diagnóstico e avaliação do barramento CAN, utilizado em conjunto com o software Logic 2 (versão 2.4.14, demo mode) da Saleae Inc.

3.3. Modelo de Comunicação

O modelo de comunicação segue as características inerentes ao protocolo CAN, garantindo que todas as ECUs possam transmitir e receber mensagens conforme suas configurações de filtragem e prioridade. O sistema é baseado na pilha de comunicação TWAI (implementação do protocolo CAN 2.0B no ESP32), nativa da ESP-IDF (Espressif IoT Development Framework) e compatível com o padrão ISO 11898 ([Espressif Systems 2025b]).

O protocolo implementa arbitragem por prioridade, garantindo que mensagens críticas, como comandos de frenagem, aceleração e direção, tenham precedência sobre dados menos urgentes, como informações de diagnóstico. Além disso, emprega mecanismos avançados de detecção e correção de erros, incluindo CRC (Cyclic Redundancy Check), bit monitoring, bit stuffing, detecção de erros por ACK (Acknowledgment Check) e retransmissão automática, assegurando a confiabilidade da comunicação. A Figura 3 ilustra o processo de envio e recepção de mensagens no barramento, onde o nó A transmite uma mensagem, recebida pelos nós B, C e D. Destes, apenas o nó D aceita a mensagem, enquanto os nós B e C a recusam.

A biblioteca *can_esp_lib* simplifica a comunicação CAN sobre o driver TWAI do ESP32, permitindo configuração dinâmica de parâmetros como taxa de transmissão, pinos, timeouts, filtros e modo de operação. Durante a inicialização, o driver é configurado

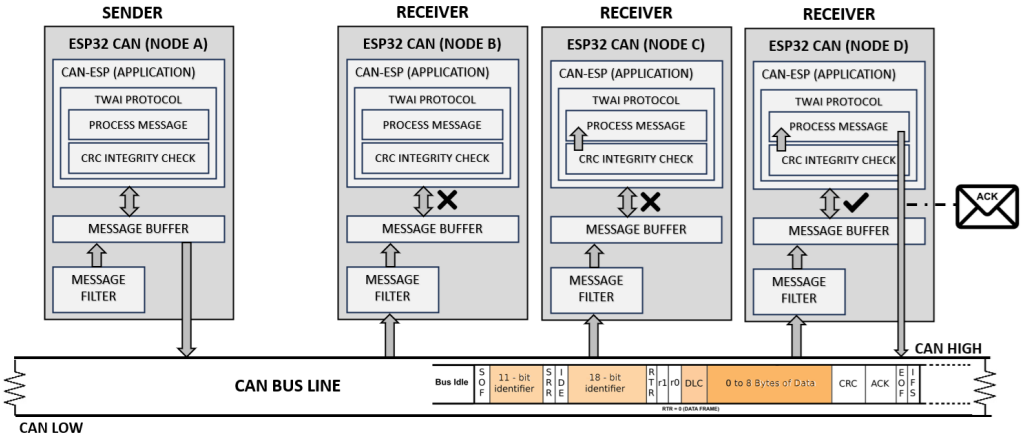

Figura 3. Transmissão/recepção de dados no barramento CAN

e uma fila é criada para envio assíncrono de mensagens. A recepção pode ocorrer via polling ou tarefa bloqueante, acionando um callback para processamento imediato. Ademais, a biblioteca oferece suporte à codificação de identificadores CAN e diagnóstico do barramento, garantindo robustez e escalabilidade para aplicações veiculares.

3.4. Atualização Remota OTA

A atualização de firmware OTA (Over-the-Air) foi projetada para permitir atualizações remotas das ECUs sem necessidade de intervenção física. Para isso, foi adotada uma solução baseada no protocolo ESP-WIFI-MESH (rede mesh sem fio para dispositivos ESP32) ([Espressif Systems 2025a]) e MQTT (Message Queuing Telemetry Transport). A ECU de Monitoramento, conectada à internet via módulo Wi-Fi embarcado, atua como root node da rede e verifica periodicamente a disponibilidade de novas versões de firmware em um servidor MQTT. Caso uma atualização esteja disponível, o firmware é baixado em pedaços (chunks) que serão recombinados para reconstruir o arquivo original no nó de destino. O firmware é finalmente distribuído via protocolo ESP-WIFI-MESH para as demais ECUs, garantindo uma transição segura entre versões, facilitando e flexibilizando o trabalho de manutenção do sistema.

A estrutura do software foi desenvolvida de forma modular, permitindo que as ECUs compartilhem uma biblioteca comum para comunicação CAN responsável pela integração ao protocolo TWAI, para troca de mensagens no barramento. Esse modelo de implementação simplifica a manutenção do sistema e permite a incorporação de novas funcionalidades sem comprometer a comunicação entre as ECUs existentes.

3.5. Confiabilidade da Solução

A confiabilidade da rede CAN-ESP é garantida pelos mecanismos avançados de detecção e correção de erros presentes no protocolo TWAI já citados e através do mecanismo de Bus-Off (estado de falha onde um nó CAN desativa sua transmissão devido a erros excessivos) também presente no protocolo, isolando automaticamente ECUs com falhas persistentes para preservar a integridade da rede [Espressif Systems 2025b].

Dessa forma, o CAN-ESP apresenta-se como uma solução robusta, tecnicamente viável, de baixo custo e escalável, adequada para aplicações em veículos elétricos e sistemas embarcados distribuídos, inclusive, comercialmente.

4. Avaliação de Desempenho

A avaliação de desempenho do CAN-ESP foi conduzida em um ambiente controlado, simulando um ciclo operacional típico de um veículo elétrico. O ambiente experimental incluiu cinco ECUs, cada uma desempenhando funções específicas dentro da arquitetura distribuída proposta. O transceptor SN65HVD230 foi utilizado na conversão dos sinais elétricos típicos da rede CAN (Figura 4) e na transmissão dos quadros (frames) de dados para a controladora CAN presente no ESP32. O cabeamento do barramento foi feito utilizando cabos UTP CAT-5 (24 AWG), com resistores de terminação de 120 Ω posicionados nas extremidades para minimizar reflexões de sinal, conforme estipula a norma ISO.

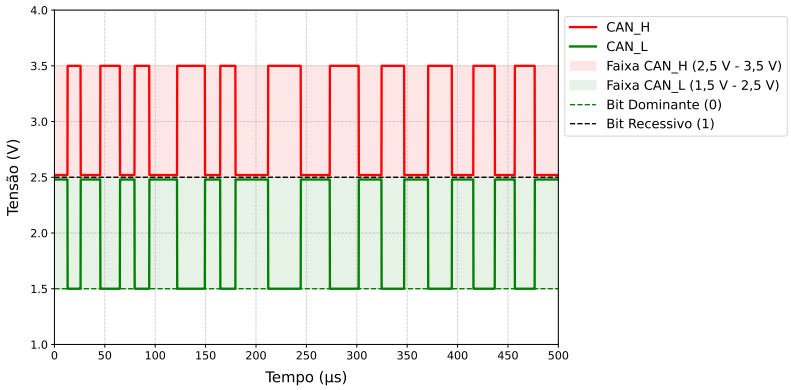

Figura 4. Faixa de tensão da rede CAN

Com relação ao monitoramento dos dados, este foi realizado através da placa CANABLE V2.0 nano, utilizando o software Cangaroo (versão 0.2.3), e do analisador lógico USB SCM 24 MHz de 8 canais, utilizando o software Logic 2 (versão 2.4.14 demo mode), permitindo a captura e análise detalhada dos quadros de comunicação.

Durante os experimentos, foram trocados 3.000.000 de quadros de dados entre as ECUs, abrangendo o quadro geral de mensagens descrito na Tabela 2.

4.1. Carga do Barramento

A carga do barramento (*bus load*) foi monitorada durante a transmissão de 3.000.000 de mensagens, com resultados variando entre 10% e 25%. Isso indicou uma utilização eficiente da rede, com margem para expansão sem comprometer a taxa de transmissão. A carga foi calculada pela Equação 1, onde T_{frame} é o tempo total de transmissão dos quadros no período de observação e T_{total} é o tempo total de operação do barramento.

$$\text{Bus Load}(\%) = \frac{\sum T_{\text{frame}}}{T_{\text{total}}} \times 100 \tag{1}$$

Os experimentos mostraram que a taxa de ocupação não ultrapassou 25%, permitindo a adição de novos nós sem risco de saturação.

4.2. Latência da Comunicação (Event Time Stamping)

A latência média do sistema, calculada entre a transmissão e recepção das mensagens, variou de 129 μs a 141 μs, com um tempo teórico mínimo de 128 μs para a transmissão

Tabela 2. Sinais e períodos de atualização

Categoria / Sinal	ID	Período de Atualização
Controle do Motor Elétrico		
Torque Requerido	0x001	50 ms
Velocidade do Motor	0x002	100 ms
Temperatura do Motor	0x003	200 ms
Status do Motor	0x004	1 s
Controle da Aceleração		
Posição do Acelerador	0x101	50 ms
Demanda de Potência	0x102	50 ms
Controle do Freio		
Pressão de Freio	0x201	100 ms
Frenagem Regenerativa	0x202	100 ms
Controle da Direção		
Ângulo do Volante	0x301	20 ms
Torque da Direção Assistida	0x302	20 ms
Status do Motor de Direção	0x303	200 ms
Controle da Velocidade do Veículo		
Velocidade do Veículo	0x501	100 ms
Demanda de Velocidade	0x502	100 ms
Diagnóstico (via OBD-II)		
Erros de Sistema	0x601	Evento sob demanda
Leitura de Parâmetros	0x602	Evento sob demanda
Status de Comunicação	0x603	1 s

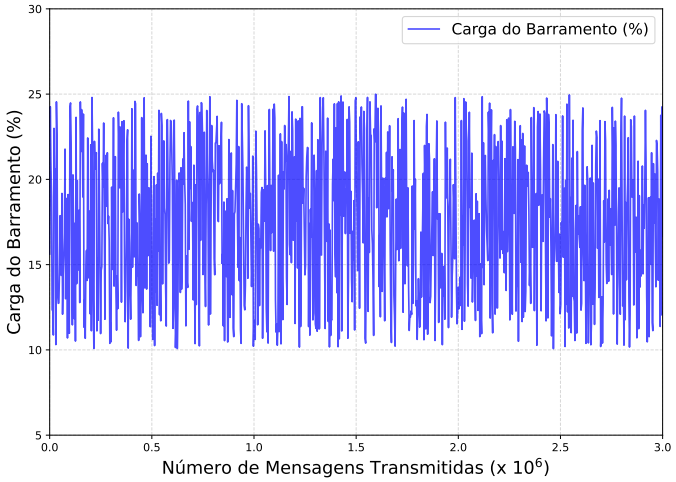

Figura 5. Variação da Carga do Barramento CAN (Bus Load)

de um quadro. A variação de aproximadamente 9,3% está dentro dos padrões aceitáveis para redes CAN de alta velocidade (1 Mbps), como mostrado na Figura 5.

A latência foi calculada pela Equação 2, onde N_{bits} é o número total de bits do quadro (128 bits para o formato estendido de 29 bits) e a taxa de transmissão de 1 Mbps.

$$T_{\text{frame}} = \frac{N_{\text{bits}}}{\text{Taxa de transmissão (bps)}} \tag{2}$$

A sincronização dos módulos foi mantida, garantindo tempo de resposta adequado para comandos críticos como frenagem e aceleração.

4.3. Taxa de Retransmissão de Quadros (Retransmission Rate)

A análise de retransmissão de quadros avaliou a robustez do protocolo TWAI na detecção e recuperação de erros. Durante a transmissão de 3.000.000 de mensagens, registrou-se apenas 0,3 retransmissões a cada 1.000.000 de quadros, resultando em uma taxa de erro inferior a 0,00003%, o que demonstra a eficácia dos mecanismos de erro e retransmissão do protocolo TWAI (Figura 6).

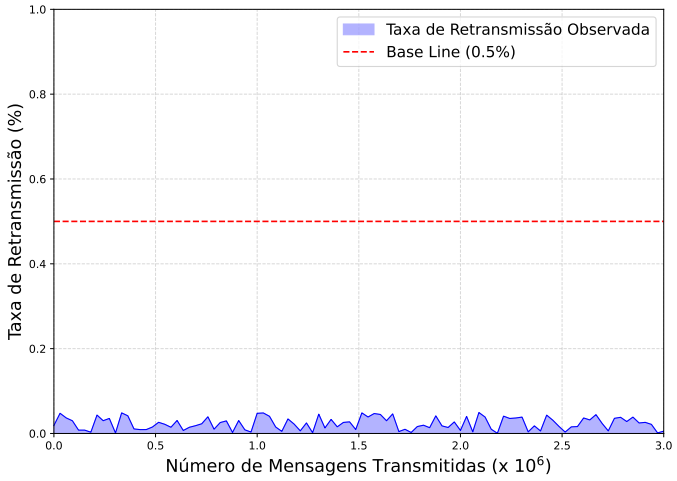

Figura 6. Taxa de Retransmissão de Quadros (Retransmission Rate)

A taxa de retransmissão foi calculada pela Equação 3, confirmando a alta imunidade do barramento CAN-ESP a interferências e sua adequação a aplicações automotivas.

$$\text{Retransmission Rate}(\%) = \frac{\text{Quadros retransmitidos}}{\text{Total de quadros enviados}} \times 100 \tag{3}$$

4.4. Contadores de Erro (Error Counters) e Integridade da Comunicação

A integridade da comunicação foi verificada pelos contadores de erro das ECUs, que registraram falhas na comunicação. Nos experimentos, não houve estados críticos como Bus-Off, assegurando a estabilidade da rede.

O protocolo TWAI implementa um mecanismo de confinamento de falha, no qual um nó que gera erros persistentes é automaticamente excluído do barramento. Para isso, cada nó mantém dois contadores internos: o Contador de Erros de Transmissão (*Transmit Error Counter - TEC*) e o Contador de Erros de Recepção (*Receive Error Counter - REC*), que são ajustados dinamicamente conforme o sucesso ou falha na transmissão e recepção de mensagens [Espressif Systems 2025a]. Com base nesses valores, o estado do nó é classificado em três categorias (Figura 7): Erro Ativo (*Error Active*), quando $TEC, REC < 128$, indicando operação normal da ECU; Erro Passivo (*Error Passive*), quando $128 \leq TEC$ / $REC < 256$, sinalizando degradação parcial da comunicação; e Bus-Off, quando $TEC \geq 256$, indicando que o nó foi desconectado do barramento.

Nenhuma das ECUs atingiu o estado Bus-Off, evidenciando alta confiabilidade na comunicação e robustez do protocolo.

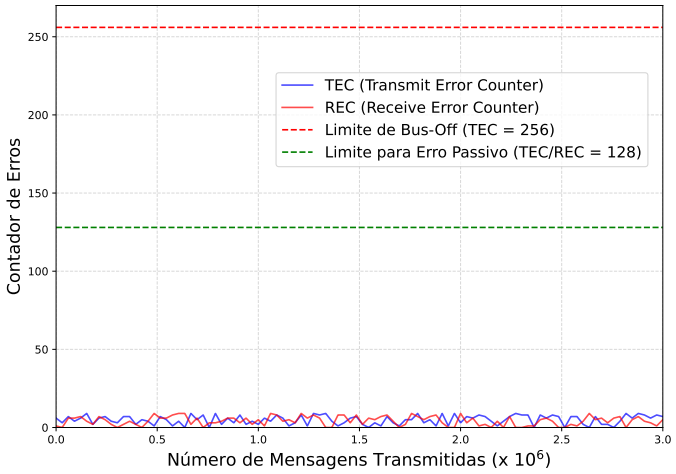

Figura 7. Contadores de erro (Error Counters)

4.5. Taxa de Colisões (Message Collision Rate)

A taxa de colisões foi monitorada para avaliar o mecanismo de arbitragem CSMA/CR do protocolo. Durante os experimentos, a taxa de colisão permaneceu abaixo de 1%, indicando eficiência na priorização de mensagens (Figura 8).

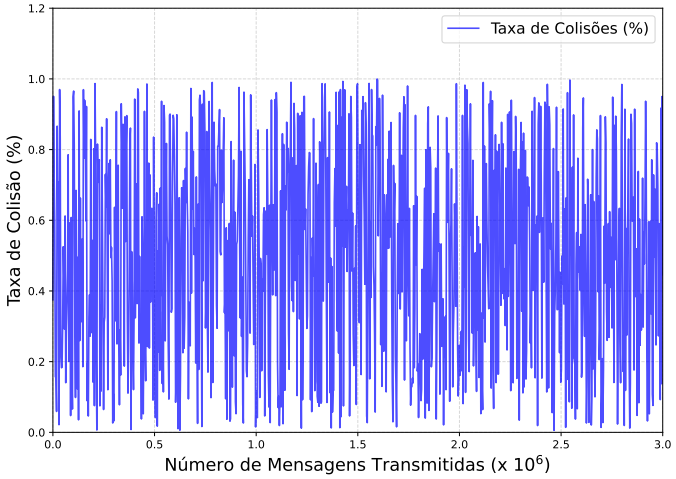

Figura 8. Taxa de Colisões (Message Collision Rate)

A taxa de colisão foi calculada pela Equação 4, confirmando que o protocolo TWAI garantiu prioridade a mensagens críticas, como controle de freio, direção e aceleração.

$$\text{Collision Rate}(\%) = \frac{\text{Mensagens em disputa}}{\text{Total de mensagens enviadas}} \times 100 \tag{4}$$

4.6. Tempo de Resposta do Sistema (System Response Time)

O tempo de resposta do sistema é um parâmetro crítico para avaliar o desempenho de uma rede CAN em aplicações automotivas. Ele pode ser definido como a soma do tempo

necessário para a transmissão do quadro no barramento, o tempo de propagação do sinal e o tempo de processamento da ECU receptora antes de gerar uma resposta.

O tempo de resposta do sistema foi analisado para verificar a capacidade da rede CAN-ESP em lidar com variações na carga operacional. A análise demonstrou que a comunicação entre ECUs ocorreu de maneira síncrona e eficiente, sem degradação significativa no tempo de resposta mesmo sob carga elevada (Figura 9).

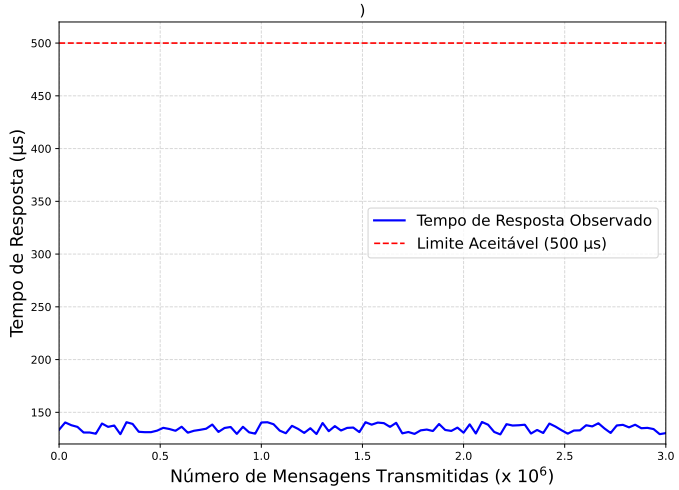

Figura 9. Tempo de Resposta do Sistema (System Response Time)

A resposta do sistema foi modelada conforme a Equação 5, em que T_{frame} representa o tempo de transmissão do quadro, $T_{\text{processing}}$ corresponde ao tempo de processamento na ECU receptora, e $T_{\text{propagation}}$ refere-se ao tempo de propagação no barramento.

$$T_{\text{response}} = T_{\text{frame}} + T_{\text{processing}} + T_{\text{propagation}} \tag{5}$$

O tempo de transmissão de um quadro CAN depende do tamanho do quadro e da taxa de transmissão do barramento, sendo calculado por:

$$T_{\text{transmissão}} = \frac{\text{Tamanho do quadro (bits)}}{\text{Taxa de transmissão (bps)}} \tag{6}$$

No sistema CAN-ESP, cada quadro transmitido utiliza o formato *Extended Frame* (ID de 29 bits) com um payload de 8 bytes. O tamanho total do quadro é de aproximadamente 128 bits, e a taxa de transmissão adotada é de 1 Mbps:

$$T_{\text{transmissão}} = \frac{128}{1.000.000} = 128\ \mu\text{s} \tag{7}$$

O tempo de propagação do sinal no barramento é influenciado pelo comprimento da rede e pela velocidade de propagação da onda eletromagnética no cabo. A relação é dada por:

$$T_{\text{propagação}} = \frac{\text{Comprimento do barramento}(m)}{\text{Velocidade de propagação (m/s)}} \tag{8}$$

Para um barramento de 10 metros, utilizando um cabo UTP CAT-5 com velocidade de propagação típica de aproximadamente 5 ns/m, tem-se:

$$T_{\text{propagação}} = 10 \times 5 \text{ ns} = 50 \text{ ns} = 0.05 \text{ µs} \tag{9}$$

O tempo de processamento da ECU varia conforme a arquitetura do microcontrolador. No caso do ESP32 WROOM-32, temos: Clock da CPU de 240 MHz, tempo médio por instrução de aproximadamente 4 ns, e cerca de 5.000 ciclos para processar uma mensagem CAN e gerar uma resposta.

Dessa forma, o tempo total de processamento pode ser estimado como:

$$T_{\text{processamento}} = 5.000 \times 4 \text{ ns} = 20 \text{ µs} \tag{10}$$

Somando os tempos calculados anteriormente:

$$T_{\text{resposta}} = 128 \text{ µs} + 0.05 \text{ µs} + 20 \text{ µs} \approx 148 \text{ µs} \tag{11}$$

O tempo de resposta do CAN-ESP está bem abaixo do limite crítico de 500 µs normalmente adotado em sistemas automotivos baseados em CAN [GmbH 1991]. Além disso, considerando os valores medidos experimentalmente, a latência observada variou entre 129 µs e 141 µs, resultando em uma variação de aproximadamente 9,3% em relação ao tempo mínimo teórico esperado. Os resultados obtidos confirmam que o CAN-ESP atende aos requisitos de tempo real para aplicações automotivas, garantindo comunicação confiável e resposta rápida entre as ECUs do sistema.

A análise de desempenho demonstrou que a rede CAN-ESP apresentou altos níveis de confiabilidade, baixa latência, mínima taxa de retransmissões e capacidade de resposta adequada para aplicações veiculares. O protocolo TWAI garantiu uma comunicação robusta, assegurando a prioridade de mensagens críticas e mantendo a integridade da comunicação.

5. Conclusão

Este trabalho apresentou o CAN-ESP, uma arquitetura de baixo custo para redes veiculares baseada no protocolo CAN e implementada com o ESP32, garantindo confiabilidade e eficiência conforme a ISO 11898. Os experimentos demonstraram operação estável, com carga de barramento entre 10% e 25%, taxa mínima de retransmissão e latência entre 129 μs e 141 μs, assegurando desempenho adequado para aplicações automotivas. O CAN-ESP se destaca pela aderência a padrões internacionais, suporte a atualizações remotas (OTA), arquitetura modular e possibilidade de integração escalável de novas ECUs. Ademais, a disponibilização do código-fonte como open source fomenta a inovação e favorece a adoção da solução tanto em contextos acadêmicos quanto industriais. Como trabalhos futuros, propõe-se a incorporação de mecanismos de segurança em nível de protocolo, testes em veículos reais e a avaliação do desempenho da solução em cenários urbanos com múltiplos veículos interconectados.

Referências

Alzahrani, A., Wangikar, S. M., Indragandhi, V., Singh, R. R., and Subramaniyaswamy, V. (2023). Design and implementation of sae j1939 and modbus communication protocols for electric vehicle. *Machines*, 11(201).

Chikhale, S. N. (2018). Automobile design and implementation of can bus protocol - a review. *IJRDO-Journal of Electrical And Electronics Engineering*, 4(1).

Desai, M. et al. (2013). Controller area network for intelligent vehicular systems. In *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pages 1–6. IEEE.

Espressif Systems (2025a). ESP-WIFI-MESH. Acesso em: 10 de fevereiro de 2025.

Espressif Systems (2025b). Two-Wire Automotive Interface (TWAI). Acesso em: 05 de fevereiro de 2025.

GmbH, R. B. (1991). *CAN Specification 2.0 Part B*. Robert Bosch GmbH, Stuttgart.

Ismail, K., Muharam, A., Amin, and Kaleg, S. (2015). Design of can bus for research applications purpose hybrid electric vehicle using arm microcontroller. *Energy Procedia*, 68:288–296.

ISO (2015). ISO 11898- Road vehicles – Controller area network (CAN). Available at: `https://www.iso.org/standard/63648.html`.

Lawrenz, W., editor (2013). *CAN System Engineering: From Theory to Practical Applications*. Springer London, London.

Li, R., Wu, J., and Wang, H. (2010). Design method of can bus network communication structure for electric vehicle. In *IFOST 2010 Proceedings*, page 4.

Ribeiro Jr, A., da Costa, J. B., Rocha Filho, G. P., Villas, L. A., Guidoni, D. L., Sampaio, S., and Meneguette, R. I. (2023). Harmonic: Shapley values in market games for resource allocation in vehicular clouds. *Ad Hoc Networks*, 149:103224.

Salunkhe, A. A., Kamble, P. P., and Jadhav, R. (2016). Design and implementation of can bus protocol for monitoring vehicle parameters. In *IEEE International Conference On Recent Trends In Electronics Information Communication Technology*, pages 1–5, India.

Valentini, E. P., Rocha Filho, G. P., De Grande, R. E., Ranieri, C. M., Júnior, L. A. P., and Meneguette, R. I. (2023). A novel mechanism for misbehavior detection in vehicular networks. *IEEE Access*, 11:68113–68126.

Vijaya, D., Kokane, R., and Kalyankar, S. B. (2015). Implementation of the can bus in the vehicle based on arm 7. *International Journal of Research in Engineering and Technology*, 4(1):29–31.

Wagh, P. A., Pawar, R. R., and Nalbalwar, D. S. (2017). Vehicle speed control and safety prototype using controller area network. In *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)*, Lonere, Raigad, Maharashtra, India.

Wang, Y. (2021). Design of electric drive system of electric vehicle based on can bus. *Journal of Physics: Conference Series*, 1982:012131.

Zuberi, K. and Shin, K. (1996). Real-time decentralized control with can. In *Proceedings 1996 IEEE Conference on Emerging Technologies and Factory Automation. ETFA '96*, volume 1, pages 93–99 vol.1.