UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Eduardo Vieira Costa Teixeira

Desenvolvimento de um sistema de projeção de Field of View dinâmico para calibração de sensores ultrassônicos

São Carlos

2023

Eduardo Vieira Costa Teixeira

Desenvolvimento de um sistema de projeção de Field of View dinâmico para calibração de sensores ultrassônicos

Monografia apresentada ao Curso de Engenharia Elétrica com Ênfase em Eletrônica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Marco Henrique Terra

São Carlos

2023

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Vieira, Eduardo Vieira Costa Teixeira
V658d Desenvolvimento de um sistema de projeção de Field of View dinâmico para calibração de sensores ultrassônicos / Eduardo Vieira Costa Teixeira Vieira; orientador Marco Henrique Terra Terra. São Carlos, 2023.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2023.

1. Advanced driver-assistance systems (ADAS). 2. Field of View (FoV). 3. Procolo XCP. 4. Protocolo ASAP3. 5. Protocolo COM. 6. Protocolo CAN. 7. Electronic Control Unit (ECU). 8. ISO 17386. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Eduardo Vieira Costa Teixeira

Título: "Desenvolvimento de um sistema de projeção de Field of View dinâmico para calibração de sensores ultrassônicos"

Trabalho de Conclusão de Curso defendido e aprovado em 30 / 11 / 2023,

com NOTA 9,0 (Nove, zero), pela Comissão Julgadora:

Prof. Titular Marco Henrique Terra - Orientador - SEL/EESC/USP

Prof. Associado Dennis Brandão - SEL/EESC/USP

Prof. Dr. Pedro de Oliveira Conceição Junior - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Professor Associado José Carlos de Melo Vieira Júnior

Este trabalho é dedicado aos meus pais, Elton e Leidemar, que se dedicaram plenamente para assegurar minha educação e estiveram ao meu lado em todos os momentos, viabilizando assim a concretização deste trabalho

AGRADECIMENTOS

Agradeço primeiramente a Deus, fonte inesgotável de sabedoria e força, por guiar meus passos nesta jornada acadêmica.

À minha família, em especial à minha mãe, por ser exemplo de determinação, sinceridade e honestidade; e ao meu pai, por ser meu pilar incansável de apoio. A minha gratidão transborda pelos seus sacrifícios feitos em prol do meu sucesso.

À minha amada namorada, Letícia, cujo amor e compreensão tornaram cada desafio mais leve.

Ao meu dedicado orientador, Marco, agradeço a orientação sábia e inspiradora que moldou meu trabalho.

À equipe EESC USP FORMULA SAE que me permitiu desenvolver o conhecimento que não está nos livros e me forneceu grandes amizades para vida, em especial ao meu colega e amigo Eduardo que me apoiou e continua me incentivando.

À equipe Robert BOSCH que me acolheu e ofereceu os materiais e a informação que foram a base desse trabalho, meu sincero reconhecimento pela oportunidade e aprendizado.

Aos meus amigos, Gaspar e Henrique por serem meus cúmplices de risos e superações, dentro e fora da universidade.

Ao meu amigo Matheus e todo o grupo Borracharia, o meu agradecimento por tornarem esta jornada inesquecível.

Este TCC é fruto de um esforço coletivo que celebra o poder da colaboração e do apoio.

Muito obrigado.

"O estudo, a busca da verdade e da beleza são domínios
em que nos é consentido sermos crianças por toda a vida."
Albert Einstein

RESUMO

VIEIRA, E. **Desenvolvimento de um sistema de projeção de Field of View dinâmico para calibração de sensores ultrasônicos**. 2023. 62p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

O uso de sistemas de monitoramento e alerta traseiros em veículos utilitários passa a ser lei conforme a Resolução 759:2018 do Conselho Nacional de Trânsito (CONTRAN), sendo obrigatórios a partir de 2025 para novos projetos de veículos produzidos ou importados e a partir de 2027 para todos os veículos produzidos. Nesse contexto, com o aumento de projetos para sensores ultrassônicos, esse trabalho visa atender aos requisitos de mercado com a entrega do documento requerido pela montadora, chamado de campo de visão, *Field of View* (FoV). Este trabalho abrangerá aspectos técnicos da instrumentação dos equipamentos de medição aos para-choque de interesse, aquisitando dados do sensor via protocolo CAN, XCP, ASAP3 e COM e gerando uma interface gráfica em tempo real do teste que é realizado segundo a ISO 17386, que trata sobre os procedimentos de testes para sistemas de auxílio a manobra em operações de baixa velocidade.

Palavras-chave: *Advanced driver-assistance systems* (ADAS). *Field of View* (FoV). Protocolo CAN. Procolo XCP. Protocolo ASAP3. Protocolo COM. *Park Assist Module* (PAM). *Electronic Control Unit* (ECU). ISO 17386. ISO16787.

ABSTRACT

VIEIRA, E. **Development of a projection system of dynamic Field of View for ultrasonic sensors calibration**. 2023. 62p. Monograph (Conclusion Course Paper) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2023.

The use of a monitoring and rear warning system in utility vehicles becomes law in accordance with 759:2018 Resolution from Conselho Nacional de Trânsito (CONTRAN), being mandatory from 2025 for new vehicle designs produced or imported and from 2027 for all vehicles produced. In this context, with the increase of projects for ultrasonic sensors, this work aims to meet the market requirements with the delivery of the document required by the OEM (Original Equipment Manufacturer), called by field of view (FoV). This work will cover the technical aspects of the instrumentation of measurement equipments to the interest bumpers, acquiring sensor data via CAN, XCP, ASAP3 and COM protocol and generating a graphical interface in real time for the test that is performed according to ISO 17386, which stablishes test procedures for maneuvering aid systems in low speed operations (MALSO).

Keywords: Advanced Driver-Assistance Systems (ADAS). Field of View (FoV). CAN Protocol. XCP Protocol. ASAP3 Protocol. COM Protocol. Park Assist Module (PAM). Electronic Control Unit (ECU). ISO 17386. ISO16787.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ADAS	*Advanced Driver-Assistance Systems*
ISO	*International Organization for Standardization*
IEL	Instituto Euvaldo Lodi
IPT	Instituto de Pesquisas Tecnológicas
CONTRAN	Conselho Nacional de Trânsito
OEM	*Original Equipment Manufacturer*
FoV	*Field of View*
CAN	*Controller Area Network*
ASAP3	*(Association for Standardization of Automation and Measuring Systems) MCD-2 Measurement and Calibration Protocol*
COM	*Component Object Model*
XCP	*Universal Measurement and Calibration Protocol*
ECU	*Electronic Control Unit*
TOF	*Time of Flight*
PAM	*Park Assist Module*
APS	*Asssisted parking systems*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
FIFO	*First in First Out*
API	*Application Programming Interface*
IDE	*Integrated Development Environment*

SUMÁRIO

1 INTRODUÇÃO

O termo carro autônomo vem sendo pauta não somente da mídia automotiva como também da sociedade mundial. O termo estrangeiro *advanced driver assistance systems* (ADAS), que é um sistema avançado de assistência ao motorista, é amplamente utilizado como uma ponte entre o carro estritamente controlado pelo motorista e o carro autônomo.

Com o advento das novas tecnologias na área de sensores e no aumento da capacidade computacional, observou-se um grande crescimento no desenvolvimento das tecnologias ADAS [Piao e McDonald 2008].

O presente trabalho abrange a área das tecnologias ADAS, mais especificamente em testes de sensores ultrassônicos, que será realizada presencialmente na empresa Robert Bosch LTDA, situada em Campinas - São Paulo. Este projeto foi financiado pelo Instituo Euvaldo Lodi (IEL), em parceira com o Instituto de Pesquisas Tecnológicas (IPT).

Em razão do projeto estar inserido em uma empresa privada, a tecnologia dos materiais patenteados pela BOSCH, utilizados para conclusão dele, não será divulgada para que não gere um vazamento de patentes, e o mesmo vale para qualquer dado que lese a empresa de qualquer forma.

1.1 Justificativa

Há vários motivos para o crescimento no desenvolvimento e na implementação das tecnologias ADAS, tais como: aumento da segurança veicular, fatores econômicos (trânsito eficiente, aumento do conforto na direção do automóvel que traz bons resultados nas vendas) e argumentos de cunho ambientais [Brookhuis, Waard e Janssen 2001].

A principal causa de acidentes em veículos automotores está na figura do motorista [Tigadi *et al.* 2016], cujas explicações residem no(a): consumo de bebidas alcóolicas, cansaço, desatenção, falta de preparo na direção, dentre outras. Para tanto, as tecnologias ADAS auxiliam o motorista no exercício da direção, com uma interface carro - motorista, mais conhecida pela sigla *Human-Machine Interface* (HMI), apropriada. Desse modo, essas tecnologias automatizam e auxiliam em processos de direção cansativos e repetitivos, aumentando a segurança [Tigadi *et al.* 2016].

De forma a seguir a tendência de mercado mundial, a resolução federal 759, de 2018 [Brasil. CONTRAN 2018], resolveu que:

> "Os veículos tipo automóvel, camioneta, utilitário e caminhonete, nacionais e importados, deverão ser dotados obrigatoriamente de um sistema de alerta traseiro,

> conforme estabelecido no Anexo I, e/ou sistema de monitoramento traseiro nos termos do Anexo II, desta Resolução."

Além disso, essa resolução definiu que será obrigatório o uso de um sistema de alerta traseiro a partir de 2025 para novos projetos de veículos produzidos e a partir de 2027 para todos os veículos em produção no Brasil.

Outrossim, levando em consideração o volume de projetos com sistemas de alerta traseiro que o Brasil se depara, este trabalho justifica-se sobre a óptica de melhorar o teste do campo de visão de sensores ultrassônicos, tecnicamente conhecido pela forma na língua inglesa: *field of view* (FoV).

O FoV (horizontal e vertical) é um documento enviado às montadoras que é usado para ratificar o projeto da instalação dos sensores ultrassônicos no veículo. Cada montadora possui um requisito quanto a área, distância de detecção e classificação do objeto pela unidade de controle eletrônica, do inglês *Electronic Control Unit* (ECU). As classificações normalmente são traduzidas nas formas visual e sonora, de modo que o condutor visualize a distância dos objetos em escalas de cores com o auxílio da HMI, e que ouça o aumento da frequência sonora enquanto o para-choque do veículo se aproxime do obstáculo.

1.2 Objetivos

Este trabalho tem como objetivo principal dinamizar e posteriormente automatizar, via *software*, o processo de geração do FoV, que atualmente é feito seguindo a ISO 17386, que será mais bem detalhado em 2.

Para tanto, os aspectos físicos dos sensores ultrassônicos e suas aplicações serão analisadas posteriormente para que seja montado um ambiente para testes e validação do *software*.

Além disso alguns protocolos de comunicação com aplicação direta neste projeto serão estudados, tais como:

- Protocolo CAN, que é a Rede de Área do Controlador, traduzido do inglês *Controller Area Network*;
- Protocolo XCP, que é o protocolo de calibração e medição universal, vindo do inglês *Universal Measurement and Calibration Protocol*;
- Procolo ASAP3 (ASAM MCD-2 MC), que é o protocolo de Medição e Calibração da Associação para Padronização de Automação e Sistemas de Medição, vindo do inglês *Association for Standardization of Automation and Measuring Systems) MCD-2 Measurement and Calibration Protocol*;

- Protocolo COM (Modelo de Objeto de Componente, em português), desenvolvido pela *Microsoft* para facilitar a comunicação entre diferentes componentes de *software* em sistemas *Windows*.

Dessa forma, objetiva-se o desenvolvimento de um *software* que faça a medição da distância entre obstáculo e sensores ultrassônicos e represente os dados obtidos em uma interface gráfica amigável.

A interface gráfica será desenvolvida com o auxílio da biblioteca OpenCV, presente na linguagem de programação *Python*, e sua interação entre a ECU, será feita com o uso da linguagem embarcada C++ (utilizando o protocolo ASAP3 sobre o XCP) e também *Python* (utilizando o protocolo COM sobre o XCP).

O protocolo XCP, como dito anteriormente, será utilizado em ambas as linguagens, com maior foco em sua integração com o protocolo COM na linguagem *Python* e com ASAP3 na linguagem C++.

Como o presente trabalho tem como fomento a otimização dos testes de geração do FoV, a dinamização da utilização da ferramenta pelos engenheiros aplicadores é uma meta decisiva no desenvolvimento do *software* que interage com ele. Nesse sentido, visa-se atender alguns requisitos de performance para a ferramenta desenvolvida.

O primeiro requisito fundamental é a visualização do FoV sendo gerado em tempo real pelo engenheiro aplicador, de forma que este consiga se atentar para falhas e a correta execução do processo. Cabe ressaltar que sobre esse requisito todos os outros se apoiarão, mantendo esse como alicerce para os outros.

O segundo requisito é a escrita de um arquivo texto, no formato padrão aceito pelo *software* BOSCH (o qual é entregue à montadora e é o oficial). Esse arquivo de texto contém todos os vetores da medição em colunas com identificações próprias do projeto. Este é um dado de entrada para o *software* próprio da BOSCH que gera a imagem do FoV oficial.

O terceiro requisito baseia-se na otimização da própria geração, de modo que o engenheiro aplicador possa refazer uma medição ou outra, diminuindo o tempo de execução do teste caso identifique alguma zona morta e possa avaliar a causa disso.

Por fim, o quarto e último requisito baseia-se na usabilidade da ferramenta, salvando o estado da medição, ao não ser possível completá-la em determinado período, e podendo, assim, retomá-la posteriormente. Este requisito será um grande avanço para a geração do mesmo, pois além de ser um processo demorado, o mesmo, muitas vezes, é refeito em outro dia, o que acarreta retrabalho.

2 DESENVOLVIMENTO

2.1 Revisão bibliográfica

2.1.1 Sensores Ultrassônicos

Os sensores ultrassônicos passaram por bastante atualizações desde as versões mais rudimentares e analógicas até o "estado da arte"na era atual dos sensores digitais. Entretanto, o seu princípio físico do funcionamento de propagação e recepção de ondas sonoras (mecânicas) mantém-se inalterado.

Por se tratar de uma tecnologia baseada na propagação de ondas mecânicas em um meio físico determinado, pode ser encontrado uma sólida base teórica em [Kinsler *et al.* 2000].

O sensor ultrassônico mede a distância de um objeto através do tempo de deslocamento da onda, do inglês *Time of Flight* (TOF) [Park *et al.* 2008], que é o tempo entre o ínico da transmissão ultrassônica e a recepção do eco.

De acordo com [Cenkeramaddi *et al.* 2020], considerando t como o TOF , x a distância efetiva entre o sensor e o objeto a ser detectado, e c como a velocidade do som no ar a uma temperatura de T(°C):

$$\begin{aligned} x &= c \cdot \frac{t}{2} \\ c &= c' + 0,6 \cdot T \end{aligned} \tag{2.1}$$

No conjunto de equações 2.1, c' é uma constante de velocidade no som de 331 m/s. Em [Wei 2015], extraído de [Kinsler *et al.* 2000], é apresentada a equação que dá uma aproximação à constante c', a qual pode ser expressa por (mantendo a nomenclatura anterior) :

$$c' = \sqrt{\frac{\gamma \cdot P}{D}} \tag{2.2}$$

Na qual, assumido o comportamento da atmosfera ao de um gás perfeito, à temperatura de 0°C e à pressão atmosférica ao nível do mar, é obtido:

- $\gamma = 1,402$, que é a razão entre calores específicos do meio, sob pressão constante e sob volume constante;
- $P = 1,01325 \cdot 10^5 Pa$, que é a pressão do meio;
- $D = 1,293 kg/m^3$, que é a densidade do meio.

A partir disso uma estimativa de $c' = 331,5 m/s$ é obtida. De acordo com [Kinsler *et al.* 2000], a razão entre pressão e densidade, com baixa variação de temperatura, tem baixa

variação, uma vez que a variação da pressão é acompanhada pela densidade do meio. Contudo, como é bem sabido por fabricantes de sensores ultrassônicos, a velocidade do som em si é predominantemente função da temperatura do ambiente, como vimos pelo segundo termo da equação 2.1. De forma mais palpável, ao manter a razão entre pressão e densidade do meio constantes, tem-se a seguinte relação, mostrada em [Kinsler *et al.* 2000]:

$$c(T_k) = c'\sqrt{\frac{T_k}{273}} \tag{2.3}$$

Onde T_k é a temperatura em °K. Como sugerido em [Wei 2015], é possível traçar a tabela 1 de correspondência a partir das equações 2.1 e 2.3:

Tabela 1 – Velocidade no som no ar para diferentes temperaturas (°C) positivas

Temperatura em °C	Velocidade do som em m/s
35	352,1
30	349,2
25	346,3
20	343,4
15	340,4
10	337,5
5	334,5

Fonte: Extraída parcialmente de [Wei 2015]

De acordo com a tabela 1 e com a equação 2.3, observa-se que a velocidade do som é proporcional às temperaturas (°C) positivas. Dessa forma, a ECU que controla os sensores ultrassônicos, mais conhecida como *Park Assist Module* (PAM) ou módulo auxiliar de estacionamento, deve ser capaz de receber a mensagem de temperatura, provavelmente advindo da rede CAN por um termopar, e com base nisso incluir a variação da velocidade do som no cálculo do TOF.

Além da velocidade do som, há alguns outros fatores que devem ser levados em consideração quando se trabalha com sensores ultrassônicos, abordados em [Everett 1989]:

- FoV : O campo de detecção deve ser próprio para aplicação, levando em conta a quantidade de sensores que serão utilizados;
- Máxima distância de detecção: O recebimento da onda mecânica pelo piezoelétrico depende da potência do sinal, da forma do material na qual a onda se reflete, entre outros;
- Precisão e resolução do sensor: dependente da frequência de operação do sensor controlado por um microcontrolador, inserido na PAM;

- Faixa de frequência: A faixa de frequência utilizada pela onda sonora é impactada pelo ambiente, o que muda a percepção da dimensão dos objetos que serão detectados e consequentemente a resolução;

Além desses, Wei também cita os fatores que limitam a máxima capacidade de detecção destes sensores, que são: a característica da superfície do objeto (sua rugosidade, sua impedância acústica, e o ângulo relativo entre o emissor e objeto), o ruído presente no ambiente, o fenômeno de interferência cruzada entre a transmissão de mais de um sensor ultrassônico (também chamada por sua tradução em inglês "*crosstalk*") e o ângulo de abertura horizontal e vertical do sensor [Wei 2015].

Atualmente, observa-se que há um padrão de instalação de 3, 4 ou 6 canais de sensores por para-choque no mercado brasileiro e nesse âmbito o fenômeno de *crosstalk* pode induzir alguns erros de falso positivo na medição, ou seja, a detecção de objetos que na realidade não estão na região do veículo. Para solucionar esse problema Borestein e Koren propõem o método de comparação com tempos de espera (*delays*) alternantes [Borenstein e Koren 1995]. A capacidade da ECU armazenar sinais de ecos passados também possibilita com que a barreira física do *idle time* seja ultrapassada a fim de aumentar a performance do sistema, fenômeno que será mais bem definido a posteriori.

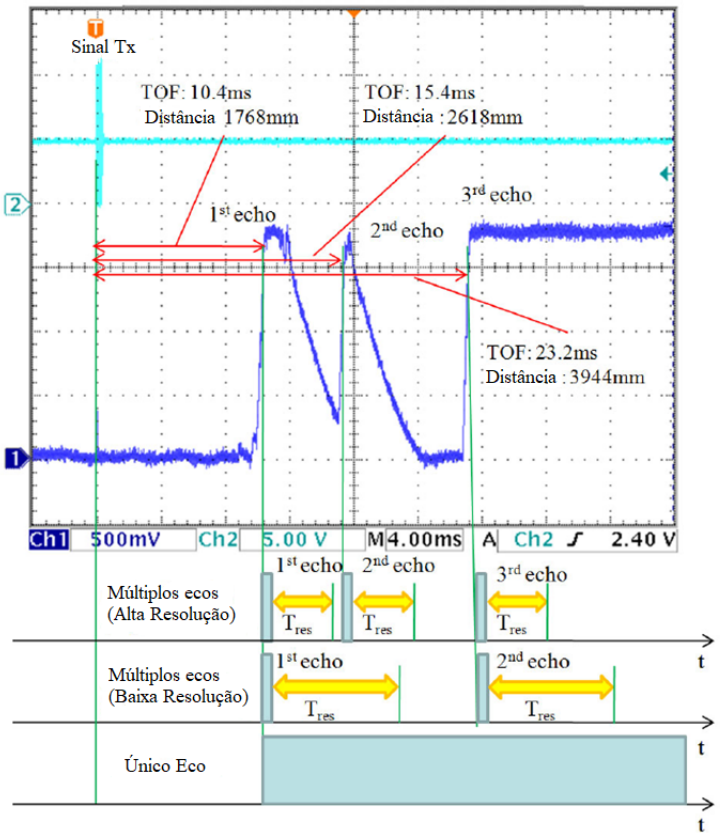
Em [Borenstein e Koren 1995] também é proposto o método de comparação de leituras consecutivas para cancelamento de ruído, e para isso, a ECU de assistência de estacionamento, PAM, deve ser capaz de gravar uma série de sinais sequenciais lidos pelos sensores. A partir disso é possível utilizar métodos computacionais para traçar padrões e identificar o nível do ruído, e no caso dos sensores implementados em veículos utilitários é possível cancelar os ecos que advém do chão, chamado pela indústria automotiva de *clutter level*.

A fim de visualização, uma amostra de um sinal ultrassônico obtido através de um osciloscópio, retirada de [Park *et al.* 2008], ilustra bem a situação da forma como uma ECU lê um ou múltiplos ecos de um mesmo objeto (função da geometria), a depender de um limite de ativação, atribuído pela PAM, conhecido na literatura como *threshold*.

No segundo canal de medição do osciloscópio na figura 1, é possível observar a representação de um piezoelétrico transmitindo uma onda sonora, e com base na mesma, é possível observar a distância mínima (referenciado em 1) de detecção do ultrassom com base no tempo que o piezo para de ser influenciado pela primeira transmissão, conhecido como *idle time*. Ademais, o TOF é observado com base na diferença entre a recepção e a transmissão do sinal, delimitando a distância do primeiro eco. Outro fator que pode ser observado na figura 1, é a capacidade de se detectar múltiplos ecos a partir de uma única transmissão, e em função disso a PAM consegue atribuir padrões e diferenciar geometrias dos objetos.

Com base nas informações aquisitadas na bibliografia é possível destacar os principais parâmetros dos sensores ultrassônicos utilizados em trabalhos científicos (em média, pois há

Figura 1 – *Padrão de Eco*



Fonte: Adaptado de [Park *et al.* 2008]

novos implementações que conseguem contornar o alcance de detecção mínimo) no quadro 1:

Quadro 1 – Parâmetros principais dos sensores ultrassônicos

Alcance nominal de sensibilidade do sensor	**Frequência nominal de operação do sensor**	**Protocolos de comunicação e plataforma compatível**	**Formato da saída de dados**
20 cm (min) e de 2 a 10 m (max)	40 a 70 kHz	I2C, RS232, USB, UART em uma ECU com um microprocessador embarcado	Tensão analógica, Largura de pulso, saída digital

Fonte: Adaptado de [Cenkeramaddi *et al.* 2020]

2.1.2 Aplicação de sistemas equipados com sensores ultrassônicos

Com o sistema ultrassônico bem definido, com suas características intrínsecas e extrínsecas, resta discorrer sobre suas aplicações ao tema da mobilidade. Este sistema encontrou-se mais adequado dentro das funções de assistência de estacionamento, em um contexto de baixas velocidades, onde não há alta influência do efeito *Doppler*.

De acordo com Wang et al., esses sensores são amplamente utilizados no caso de estacionamento paralelo, ou seja, a vaga é paralela ao sentido da via ou do estacionamento [Wang *et al.* 2014]. O seu uso não se restringe somente à detecção do espaço de vaga paralela e o seu armazenamento na memória, como também pode ser utilizada para abortar a manobra caso algum objeto, previamente não detectado, se posicione como um obstáculo para ela.

É necessário salientar que neste âmbito há duas vertentes: o sistema de estacionamento autônomo e o assistido. O primeiro refere-se a todo o processo ser controlado automaticamente pelo próprio veículo, desde a identificação da vaga, o posicionamento inicial, o cálculo da trajetória, a direção do volante, o engate da marcha ré, e os pedais de acelerador e freio. Por outro lado, o sistema assistido, também chamado na literatura por *Assisted parking system* (APS), não assume o lugar do motorista, apenas tem a função de auxiliar no controle lateral do veículo (direção), como bem definido pela ISO 16787 [Standardization 2017].

A montadora Citröen desenvolveu um sistema de auxílio de estacionamento no veículo C3 em Sina no ano de 2005, de acordo com [Wang *et al.* 2014]. A velocidade deveria ser menor de 25km/h e se o espaço detectado pelos sensores ultrassônicos não fosse o "suficiente"(de acordo com sua própria lógica), haveria um sistema de alerta para o motorista.

Wang et al. também relata sobre o sistema "Park4U"instalado em um carro sedan de série LS da montadora Toyota-Lexus, que utilizava a direção elétrica, na qual o volante realizou a direção automática enquanto o motorista apenas deveria engrenar a marcha ré [Wang *et al.* 2014]. Como dito anteriormente, o sistema de detecção ultrassônico continua ativo para que colisões sejam evitadas.

Mais especificamente na parte de estacionamento autônomo, [Chen, Hsu e Yao 2012] descrevem o modelo das três operações fundamentais no estacionamento: direção, aceleração e frenagem - com o auxílio de sensores ultrassônicos, bem como a aplicação da função de frenagem de emergência na operação do estacionamento.

Embora sua aplicação esteja amplamente ligada à assistência em sistemas de estacionamento, a mesma não se restringe somente a esses. Esses sensores estão ligados também a diversos ramos da indústria, e fazem diversas medições. Como relatado em [Hauptmann, Hoppe e Püttmer 2002] , alguns fenômenos físicos dos sensores, como : fase, frequência, amplitude - são utilizados para medidas de densidade, viscosidade, escoamento de líquidos, propriedade dos materiais - podem ser utilizados em processos fundamentais na indústria e no trabalho de qualidade do produto.

2.1.3 Protocolo CAN

O protocolo CAN foi criado a partido da mudança de paradigmas na indústria automotiva na década de 1980, na qual houve uma crescente necessidade do aumento da comunicação entre as unidades eletrônicas de controle para melhorar a performance do veículo automotor [Vector].

Nesse sentido, com limitadores, tais como: excesso de cabos e troca de dados restringida a poucos módulos - a empresa Robert Bosch idealizou e implementou a comunicação CAN que permitiu o atendimento dos requisitos de diminuição da complexidade dos barramentos, da simplificação do planejamento e instalação, da redução de peso e do espaço ocupado pelos barramentos.

O protocolo CAN é padronizado desde 1994, de acordo com [Vector], e é descrito por 4 documentos:

- ISO 11898-1, que descreve o protocolo CAN;
- ISO 11898-2, que descreve a camada física de alta velocidade da rede;
- ISO 11898-3, que descreve a camada física de baixa velocidade da rede;
- ISO 11898-4, que descreve a comunicação por tempo de disparo.

De acordo com [Standardization 2015], o protocolo CAN ocupa as duas primeiras camadas do modelo ISO/OSI (camada física e de dados). A rede é caracterizada por nós que se ligam ao barramento, o qual é formado por um par de fios trançado (*High* e *Low*). A razão disso é a redução do campo magnético e da suscetibilidade aos seus efeitos, os quais degradam os sinais que trafegam no barramento. É definido que a máxima taxa de dados é de 1Mbit/s e a rede deve ter no máximo 40 metros de comprimento para que não haja perdas de dados.

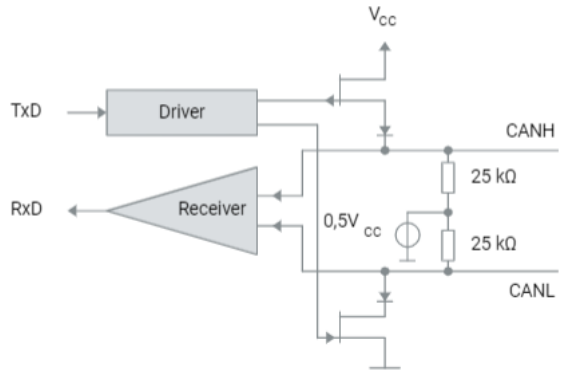
O fenômeno de reflexão ocorre quando um sinal transmitido no meio tem grande parte de sua potência refletida de volta à origem por razão de imperfeições no cabeamento e na divergência de impedâncias [Natale *et al.* 2012]. Para isso, a [Standardization 2015] determina algumas configurações, sendo a mais usual o posicionamento dos resistores de 120 Ω (impedância da rede estimada) nas terminações do barramento. Outra estratégia, é o posicionamento em série de resistores de 60 Ω em cada terminação, e no nó medial a ligação de um capacitor, cuja capacitância varia de 10nF a 100nF [Natale *et al.* 2012], conectado diretamente à referência. Essa configuração atua como um filtro passa-baixa, o qual aumenta a imunidade à ruídos internos e externos de alta frequência [Vector].

A transmissão dos dados é baseada em diferencial de tensão de forma que elimine os efeitos negativos da interferência supracitada, trazida da tensão induzida dos motores, sistemas de ignição e dos interruptores ao longo do sistema elétrico do veículo.

A interface entre o controlador e o barramento é realizada pelo *transceiver*, ilustrado na imagem 2. Este é uma ponte entre a saída SPI e o protocolo CAN. Da imagem pode ser extraída o princípio de funcionamento do *transceiver*; quando o sinal Tx está em nível alto, habilita o seu *driver* e o mesmo deixa os transístores na região de corte, e tanto *CAN High* e *CAN Low* têm diferença de potencial nula. Quando o sinal Tx está em nível baixo deixa os transístores na

região de saturação e de acordo com a [Standardization 2015] a tensão diferencial deve ser de 2V.

Figura 2 – *Esquemático do Transceiver*



Fonte: Retirada de [Vector]

Esse protocolo, assim como todos os outros tem um *data frame* específico, em português quadro de dados, que basicamente é expresso por um: identificador ou ID, o conteúdo da mensagem, a codificação da mensagem (chamado de CRC), e o reconhecimento da mesma. O protocolo CAN tem algumas particularidas que cabem ser ressaltadas, como o seu princípio de acesso, no qual todos os nós da rede têm o direito de acessá-la sem requerer permissão [Vector] de quem está enviando.

Pelo motivo desse protocolo ser baseado em eventos, a possibilidade de dois ou mais nós tentarem transmitir dados ao mais tempo é resolvida com o método de portadora sensitiva de múltiplo acesso para evitar colisões, cuja nomenclatura é CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*). Este método assegura que os nós que tentem acessar a rede esperem até que o barramento esteja disponível.

O critério de priorização das mensagens que serão enviadas está baseado no identificador delas. O método CSMA/CA, que é baseado em *bitwise arbitration*, estabelece que as mensagens com maior prioridade, definidas pelo seu identificador, são enviadas na sequência decrescente. A prioridade segue a lei que determina que o menor identificador absoluto tem a maior prioridade dentre os outros do que com o maior, ou seja, em um identificador de 4 bits, o $(1010)_2$ tem maior prioridade sobre o $(1011)_2$.

A seguir outros protocolos serão estudados, utilizados nesse projeto, os quais tem uma maior camada de aplicação, para calibração e medição de dados internos das memórias das ECU's. Entretanto é necessário salientar que o protocolo CAN é de suma importância para essa aplicação, uma vez que a PAM deve receber várias mensagens via barramento CAN, para que esta funcione tanto em bancada quanto *on-site*. A instrumentação requerida para pleno funcionamento da bancada de teste será melhor definida em 2.2.2.

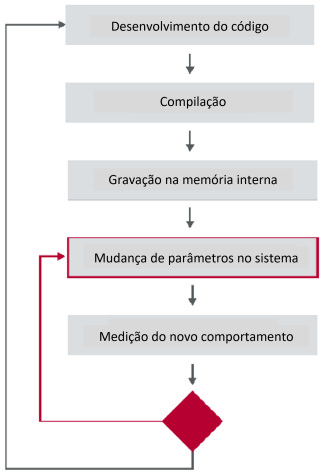
2.1.4 Protocolo XCP

O protocolo XCP, abreviação de *Universal Measurement and Calibration Protocol*, traduzido por protocolo de calibração e medição universal, é um padrão de comunicação que possibilita escrita e leitura às memórias internas de sistemas embarcados [Patzer 2022].

A implementação deste protocolo está pautada na comunicação mestre-escravo, onde o mestre é a ferramenta de medição e calibração, normalmente um *software* instalado em um computador. A parte do escravo é constituída basicamente dos sistemas embarcados de interesse (ECU's).

Essa comunicação se difere do processo convencional de desenvolvimento de *software* de aplicação, em que o desenvolvimento e a mudança de parâmetros ou variáveis vêm antecipadamente ao processo de compilação e gravação na memória do microcontrolador da ECU. Nesse caso, o protocolo XCP permite que a mudança desses parâmetros (calibração) seja feita de forma *online*, sem que haja necessidade de recompilar o código novamente e gravá-lo na memória interna da ECU; esse fluxo de trabalho está mostrado na figura 3.

Figura 3 – *Processo de desenvolvimento de aplicação XCP para sistemas embarcados*



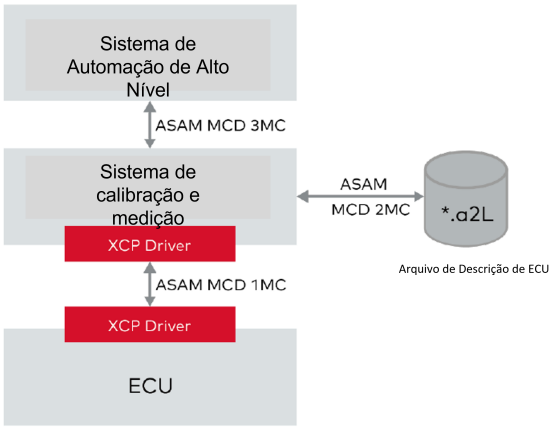
Fonte: Adaptado de [Patzer 2022]

Em linhas gerais, a figura 3 exemplifica bem o desenvolvimento de *software* e aplicação dos parâmetros para gravação nas ECU's utilizando o protocolo XCP no mercado automotivo. Essa ferramenta torna-se de suma importância para o engenheiro aplicador, uma vez que a calibração de parâmetros dentro de cada aplicação pode ser iterativa de forma exaustiva.

A arquitetura da interface XCP presente no sistema introduzido acima é exemplificado pela figura 4, e é necessário salientar que a mesma exemplifica o modelo utilizado no presente trabalho e que será mais bem definido no item 2.2.

A primeira interface, ASAM MCD 1MC, é caracterizada pela camada física entre o microcontrolador da ECU e o *hardware* de calibração e medição VX1000, cuja ligação será

Figura 4 – *Modelo de interface ASAM*



Fonte: Adaptado de [Patzer 2022]

detalhada em 2.2.2. A conexão com o microcontrolador da ECU é via interface JTAG (*Joint Test Action Group*), que é utilizado como depurador de *hardware*, o que simplifica o teste e depuração de placas de circuito impresso [Lu *et al.* 2018]. De outro lado, é utilizado na parte da VX1000, um cabo POD, (*Plug on Device*), traduzido para conexão ao aparelho, que tem um interfaceamento próprio ao JTAG.

A segunda interface, ASAM MCD 2MC, explicita o modelo de endereçamento do protocolo XCP, e nesse projeto será utilizado o *software* VECTOR CANape para esse fim. Como o XCP é baseado em acesso à endereços, o acesso à escrita e à leitura de objetos na memória são baseados nos mesmos [Patzer 2022]. Para que o usuário não trabalhe com objetos referenciados a endereçamentos e sim com nomes simbólicos, o arquivo .a2l, retratado na figura 4, faz a relação entre nome do objeto e seu endereçamento.

A terceira interface, ASAM MCD 3MC, que é a interface de automação, é utilizada para conectar um sistema terceiro à interface de medição e calibração. Nesse sentido, a ferramenta de medição e calibração da VECTOR GmbH oferece documentação dessa comunicação no seu menu de ajuda. No presente projeto optou-se por explorar as comunicações ASAP3 e COM.

2.1.5 Protocolo ASAP3

Toda informação contida nesse tópico é associada ao documento adjunto à instalação do *software* (CANape API *Help*). Esse material de apoio está disponível somente após a instalação do programa que faz a interface XCP e o arquivo .a2l, nesse caso o *software* VECTOR CANape.

De modo geral há duas formas de se utilizar o protocolo ASAP3; uma delas é utilizando o CANape e aplicação cliente na mesma máquina e a outra é utilizando uma rede sobre o protocolo TCP/IP, ou seja, a aplicação cliente e o CANape são executadas em duas máquinas distintas. Neste projeto, ambas aplicações são executadas na mesma máquina, sem a necessidade

de estabelecer uma conexão remota.

Em suma, esse protocolo expõe as funções do CANape (medição e calibração de módulos embarcados) para outras aplicações, de modo que essas possam avaliar e processá-las para outros fins. Para o projeto será utilizada apenas a parte de medição com funções de aquisição de dados.

Para criar uma aplicação, nesse caso, a aquisição de dados, é necessário que se insira a biblioteca CANapApi.lib no projeto de *software* (ambiente de desenvolvimento C++). Além disso o arquivo canapapi.h deve estar no arquivo fonte do código e a CANapAPI.dll deve ser salva no mesmo diretório aberto do programa cliente.

Para a confecção do código em C++ há a necessidade de executar as sequências de: início e fim, medição e de aquisição de dados, cujos objetos e instruções serão abordados a seguir.

Na sequência de início será chamada uma função própria, definida na biblioteca referida, que retorna um objeto *Handle*. Este objeto é necessário em praticamente todas as outras funções de comunicação. Nessa função se inicia o CANape e configura algumas propriedades, como o tamanho do *buffer* FIFO, *First In First Out* (Primeiro na entrada e primeiro na saída), e do tempo de resposta da aplicação para que a comunicação seja abortada, conhecido como *time out*. Além dessa primeira função, também há outras que fazem seleção do dispositivo (normalmente a ECU XCP própria) e seu arquivo de descrição, chamados de módulo (*device*); e por fim se encerra a aplicação com uma função de saída, que é chamada pela CANapAPI.dll, a qual fecha o CANape.

É importante ressaltar que cada tarefa da ECU tem seu *buffer* FIFO, e sua taxa de amostragem é definida pelo arquivo .a2l. O espaço dedicado na memória do FIFO é dividido por uma matriz bidimensional, onde uma dimensão é o tamanho do FIFO, e a outra é o tamanho da amostra, de modo que o espaço na memória seja: espaço na memória = tamanho do FIFO x tamanho da amostra. Dessa forma, se uma determinada tarefa tem uma taxa de amostragem de 10 milissegundos, e o tamanho do FIFO é de 180 bits, significa que o *buffer* é capaz armazenar dados de medição de até 1,8 segundos. Além disso o FIFO também conta com uma propriedade de exclusão dos valores excedam sua memória, portanto o engenheiro deve se atentar para incluir o número de sinais suficientes e que o período analisado não seja maior que a memória (em bits) do FIFO multiplicado pela amostragem desse sinal.

A sequência de medição é uma parte de configuração que contém a aquisição de dados. Em primeiro plano, ela avalia as tarefas das ECU's (informação como taxa de transmissão, identificação e modo de dados). É necessário salientar que para facilitar a leitura do código de uma ECU, os eventos de medição são divididos em tarefas, nas quais vários sinais são processados, e cada uma possui seu *polling*, mecanismo de checagem, próprio. Em seguida se apaga os canais de medição anteriores e se cria uma lista de canais de aquisição (um para cada tarefa). Logo após essas instruções, o *loop* da aquisição de dados é iniciado.

Por fim, a primeira função da sequência de aquisição de dados checa o estado do *buffer* FIFO; caso o tamanho do FIFO seja insuficiente ou a taxa da aplicação cliente seja muito baixa

alguns dados são perdidos. Após isso, outra função retorna as amostras contidas no *buffer* FIFO com a identificação da tarefa correspondente. Por fim, se realiza um *loop* com o número de iterações iguais ao tamanho de amostras contidas no *buffer*, e uma função retorna os valores contidos dentro de cada amostra.

Em linhas gerais essa é a rotina básica, e pode ser feita sobre a camada física *Ethernet*, de forma que CANape e aplicação cliente se comuniquem, que é o caso do projeto em questão.

2.1.6 Protocolo COM

O protocolo COM por sua vez tem a mesma finalidade que o protocolo ASAP3; fazer a interface entre a automação da medição e/ou calibração e um dispositivo (*software*) que faz a comunicação XCP com a ECU. Entretanto, por ser um protocolo abrangente dentro do sistema operacional Windows, e por ter a possibilidade de ser utilizado em linguagens interpretadas, como o *Python*, têm funções e métodos mais bem definidos, o que simplifica sua implementação para o programador.

A inicialização dessa comunicação com o *software* CANape, é similar ao do ASAP3, indicando o diretório do arquivo de configuração base, o tempo de espera de resposta (*time-out*), o modo, a dimensão do *buffer* FIFO e da dimensão amostral de cada FIFO. Entretanto, essa inicialização é feita de maneira mais aperfeiçoada com métodos e funções da biblioteca win32com.client.

Semelhante ao protocolo 2.1.5, a parte de aquisição de dados, ou melhor, a medição, é feita levando em consideração o dispositivo XCP em questão, seu arquivo .a2l, e fazendo a varredura da quantidade de FIFO's e dos seus canais de aquisição de dados (amostras), ou seja, duas varreduras.

Por fim, o programa é encerrado ao utilizar métodos da biblioteca explicitada, o que faz com que o código seja mais compacto do que aquele utilizado em 2.1.5. Outrossim, pela programabilidade do protocolo COM via *Python* e por ser a mesma linguagem utilizada para a interface gráfica, discutida em 2.2.3.1, mostra-se mais atraente para o teste da medição em comparação com o protocolo ASAP3.

2.2 Materiais e Métodos

2.2.1 Requisitos de performance e procedimento de testes horizontais traseiros

A resolução 759 do CONTRAN determina os requisitos de performance e de testes para a adoção dos sistemas traseiros de alerta automotivo [Brasil. CONTRAN 2018]. Em função da mesma ser baseada na [Standardization 2018], e o padrão da empresa, na qual este projeto está sendo desenvolvido, segui-la por ser mais abrangente e específica, abordarei as duas.

A resolução [Brasil. CONTRAN 2018] delimita tanto os requisitos para sistemas de alerta e detecção traseira (sistemas de ultrassom) quanto requisitos para sistemas de monitoramento

traseiro (sistemas com auxílio de vídeo). O presente trabalho tem foco apenas na primeiro item, que trata do sistema que tem a finalidade apenas de alertar o condutor veículo acerca da aproximação de um objeto que colida com o para-choque.

Em [Standardization 2018] é definido o critério de ativação desse sistema e ele está condicionado a alguns itens, como: a marcha ré engatada, velocidade menor que a ativação ($v_{on} \geq 0,5m/s$), ou outra marcha engatada com velocidade maior (v_{off}) que a velocidade de desativação com distância percorrida maior que um limiar x_{off}. O quadro 2 mostra o critério de ativação e desativação do sistema de alerta, instruído às montadoras, onde "o"indica opcional, "+"indica ativo e -"indica inativo.

Quadro 2 – Critério de ativação e desativação do sistema

Área de monitoramento	**Marcha ré engatada**	**Outra marcha engatada**	**Outra marcha engatada**
		$v < v_{on}$	$v \geq v_{off}$ ou $x > x_{off}$
Dianteira	o	+	-
Lateral Dianteira	o	+	-
Traseira	+	o	-
Lateral Traseira	+	o	-

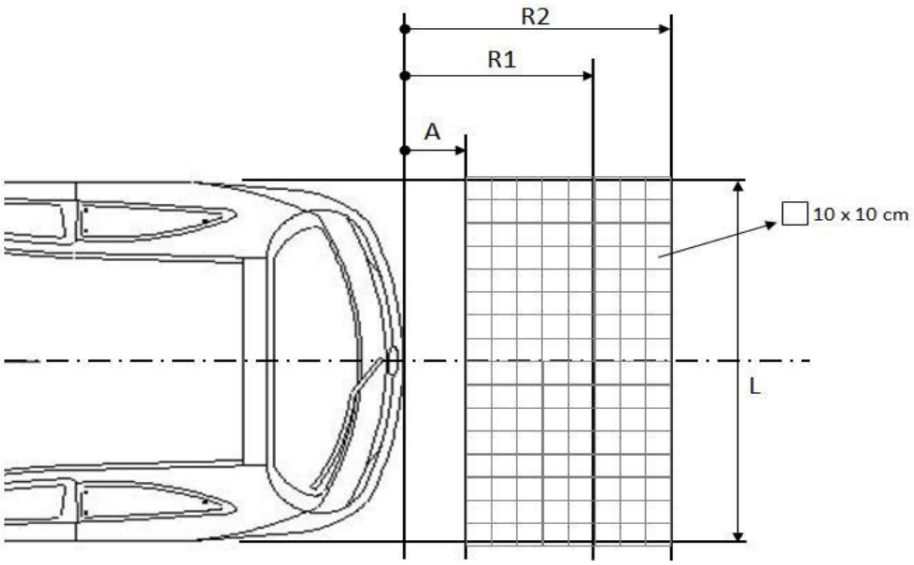
Fonte: Adaptado de [Standardization 2018]

É necessário salientar que esses padrões (v_{on}, v_{off}, $xoff$) são definidos pela montadora e que o quadro 2 é um requisito mínimo a ser seguido. Além disso em veículos com transmissão automática o sistema pode ser desativado quando o motorista pressiona o pedal de freio ou câmbio está na posição P.

Nesse ínterim, a resolução [Brasil. CONTRAN 2018] resolve alguns critérios acerca da taxa de cobertura da zona de monitoramento traseira, que é item deste projeto. Como mostrado na figura 5 e quantificado na tabela 2, a distância "A", a 20 centímetros da extremidade traseira não deve ser considerada como meta de detecção pelas limitações físicas dos sensores ultrassônicos, em função do *idle time*, caracterizado em 2.1.1.

Como mostrado na figura 5, uma planificação no chão (tapete) é dividida em quadrados de dimensões 10 x 10 centímetros, e sua dimensão deve ser de no mínimo 80 centímetros de comprimento e largura igual à calculada do carro (normalmente a distância de uma extremidade do retrovisor direito ao esquerdo). Dentro de cada quadrado é posicionado um elemento obstáculo (cilindro ou *isopole*) com 75 milímetros de diâmetro, altura de 1 metro e oco. A partir dessa operação o teste é feito observando as medições de eco dos sensores ultrassônicos, de forma que os requisitos de detecção sejam cumpridos para cada área de interesse: A1 (delimitada por $(R1 - A) \cdot L$ e A2 (delimitada por $(R2 - A) \cdot L$).

Figura 5 – *Determinação da taxa de cobertura da zona de monitoramento horizontal traseira (fora de escala)*



Fonte: Retirado de [Brasil. CONTRAN 2018]

Tabela 2 – Tabela das zonas de monitoramento e distâncias

Limites das zonas de monitoramento	Distâncias (m)
R1	0,6
R2	1,0
A	0,2

Fonte: Extraída de [Brasil. CONTRAN 2018]

De acordo com [Standardization 2018], a razão mínima de detecção do processo de geração do FoV deve ser de 90% para a área A1 e de 87% para a área A2. Similar em parte, a resolução [Brasil. CONTRAN 2018] define que ambas as regiões devem ter uma razão de detecção de 87 % por parte do sistema ultrassônico.

Por fim, é necessário salientar que o ambiente de testes deve ser isolado acusticamente de forma que não haja nenhuma interferência de ecos oriundos de fontes externas, nem reflexões indevidas em outros objetos, exceto o *isopole*. Além disso a superfície sobre a qual o teste é feito, onde o tapete está posicionado, deve ser seca, lisa e plana de forma que não haja nenhuma reflexão ultrassônica indesejada.

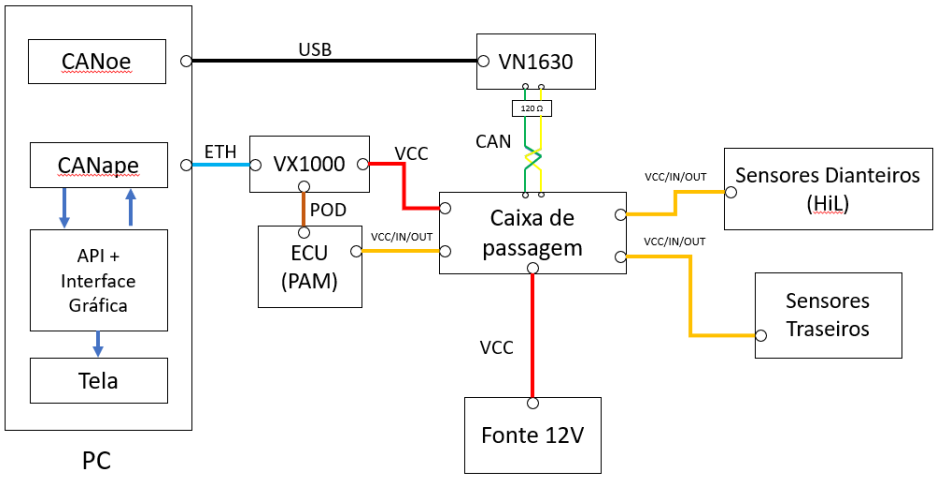
2.2.2 Montagem da aplicação

A montagem do ambiente de aplicação do projeto consistiu em:

- Sensoriamento de interesse, com 4 sensores ultrassônicos posicionados com o piezoelétrico faceado à superfície do para-choque (2 pares simétricos em relação ao centro do carro);
- Sensoriamento de 4 sensores ultrassônicos simulados em *Hardware in the loop* próprio, por se tratar de uma ECU de 8 canais;
- Fonte de tensão e caixa de passagem, para maior organização do cabeamento e modularidade dos testes, com passagem do chicote de alimentação e de sinais;
- ECU (PAM) de interesse do projeto;
- *Hardwares* VECTOR de medição e simulação de sinais;
- Máquina pessoal com os *softwares* e *drivers* específicos para comunicação com os aparelhos de medição e com o *software cliente*.

A figura 6 mostra o esquemático e a ligação física dos módulos entre si, com ligações caracterizadas por cor, ligações físicas conectadas entre terminais, e ligações entre *software* representadas por setas.

Figura 6 – *Esquemático da montagem do experimento*



Ligação entre *Hardware* e *Software*, realizado pelo autor

De forma geral, a montagem em geral está focada em gerar sinais necessários para o pleno funcionamento da ECU e ler os sinais internos de interesse para o projeto (eco direto e cross eco).

Para tal, uma fonte de 12V é conectada à caixa de passagem e a partir dela é alimentado os módulos: ECU, VX1000, e o *Hardware in the loop*.

Como dito anteriormente, para um projeto de 8 canais, e como a região de interesse nesse experimento é a traseira, os 4 canais dianteiros devem ser simulados por um *Hardware in the*

loop próprio que tem o chicote de ligação idêntico ao traseiro (alimentação, sinais de entrada e sinais de saída).

Os sensores traseiros devem ser posicionados da melhor maneira possível para realização do teste, com a peça firme ao *holder*, com um anel de silicone circunscrito ao piezoelétrico de forma que isole acusticamente o sensor do para-choque. O chicote de ligação, assim como o do *Hardware in the loop*, é conectado à caixa de passagem.

Em vias de simular as mensagens CAN do restante do carro (e.g velocidade, sinais da HMI, sinais do câmbio do veículo) requeridas pela ECU, conectada à caixa de passagem está o aparelho VN1610A da Vector GmbH, que se comunica através do protocolo CAN (*High* e *Low*) e um resistor de 120 Ω é posicionado para suprimir o fenômeno da reflexão. A mesma está conectada à máquina de aplicação por um cabo USB, cujo *driver* é utilizado pelo *software* CANoe, onde está simulado as mensagens CAN (identificadores, campo de dados, CRC e o restante do *frame*), os nós na rede e arquivo .dbc, que mapeia a rede.

Acoplado à ECU (PAM) por meio de um conector JTAG (direto no microcontrolador) está o aparelho VX1000 da Vector GmbH, conectado a ele através do cabo POD. Este aparelho está conectado ao computador por meio do protocolo Ethernet, cujo IP é sincronizado ao software CANape, patenteado pela Vector GmbH, que mapeia todos os sinais da ECU por meio do arquivo a2l. A partir desse *software* é possível criar janelas, gráficos em tempo real, numéricos a partir desses sinais.

2.2.3 API entre *software* próprio e ECU

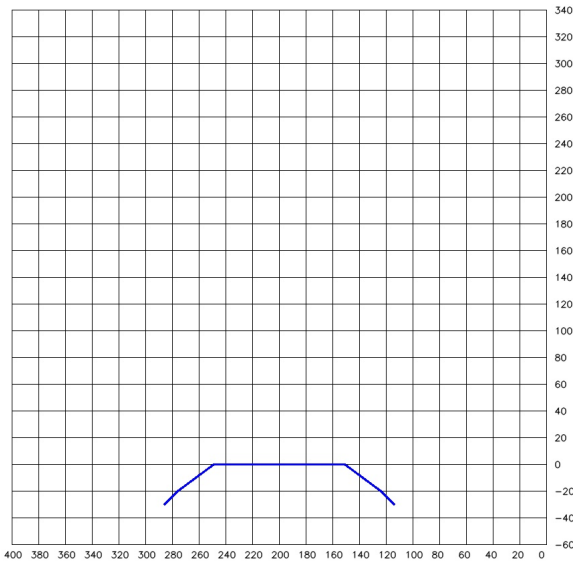
As API's, ou interface de aplicação programável (do inglês *Application Programming Interface*, utilizadas foram a ASAP3 e COM, discorridas em 2.1.5 e 2.1.6. Cada qual sua rotina de início, aquisição de dados e finalização têm o mesmo princípio de atribuição de sinais às tarefas da ECU por meio do FIFO *buffer*.

2.2.3.1 Interface gráfica no OpenCV via *Python*

Alinhado aos objetivos deste trabalho, o requisito principal é a identificação do FoV dos sensores do para-choque pelo engenheiro aplicador, seja o teste realizado de forma manual ou de forma automática. Para tal, uma interface gráfica deve ser gerada para facilitar a visualização da geração do FoV pelo engenheiro e a aprovação ou não do resultado final, para o envio do documento à montadora.

Como dito anteriormente, uma interface gráfica foi gerada para que a medição fosse mais amigável para o usuário. O propósito inicial do projeto consistiu em projetar a imagem 7 em um tapete com dimensões correspondentes à mesma, 400 por 340 centímetros. É necessário salientar que todos os *grids*, e legendas foram confeccionadas a partir do uso da biblioteca OpenCV ou cv2, disponível em ambiente *python*.

Figura 7 – *Interface em python que simula tapete quadriculado e para-choque sobre uma vista vertcial*



Fonte: Retirado da aplicação deste projeto

Ademais, o contorno, traçado em azul, representa o para-choque do veículo, cujos sensores serão analisados posteriormente, e a representação na imagem tem como entrada as posições geométricas dos sensores no para-choque, que são realizadas pelo usuário a partir do CAD fornecido pela montadora. A identificação das posições é feita a partir de um documento de estudo de instalação próprio da empresa, e esses dados são inseridos via teclado pelo o usuário do *software* desenvolvido por este presente projeto.

Adiante, essa configuração dos *grids*, das legendas, e lugar geométrico do parachoque, servirá de gabarito para a projeção, na qual haverá o processo de varredura dos dados de distância dos sensores, dentro de cada quadrado, que representa 10 x 10 cm no tapete posicionado no laboratório, onde o *isopole* será posicionado.

2.3 Resultados

Os resultados deste projeto têm maior cunho determinístico em detrimento do estatístico. O projeto é o desenvolvimento de uma ferramenta capaz de auxiliar um teste fundamental no âmbito de sensores ultrassônicos. Ao longo do ano foram realizadas algumas versões, dentre elas pode-se destacar:

- A versão off-line, ou seja, apenas uma interface gráfica que permite a visualização de um teste que já foi realizado;

- A versão on-line em C++;

- A versão on-line em python.

Cada versão teve uma melhoria em relação à anterior e foi fundamental para a construção da final. O ponto partida foi concebido através de uma dor na dificuldade de visualizar a geração do FoV em tempo real e a possibilidade de alterar medições não bem sucedidas no ponto de vista do engenheiro aplicador.

2.3.1 Interface gráfica

A primeira versão atingiu o requisito de ilustrar o tapete real graficamente com os *grids*, mostrado na figura 7, o parachoque (com base em 10 pontos mapeados em CAD), e uma pré visualização de um eco direto ou um *cross echo* já mensurado anteriormente. A ilustração gráfica foi realizada com a utilização da biblioteca Open-CV, disponível em *Python*, usando métodos de linha, texto e pontos, considerando a imagem ou objeto como vetor tridimensional (altura, largura e canal de cor RGB) de resolução ajustável, onde cada canal de color é multiplicado por 255, ou seja, 8 bits.

Sobre o objeto imagem, todas as operações de pintura dos quadrados (10 x 10), utilizando o método retângulo, são realizadas considerando um quadrado (6x6), o que facilita a visualização gráfica por parte do engenheiro aplicador, em função do comprimento e largura dos *grids*. O resultado é mostrado na imagem 8, que é a combinação entre a interface gráfica desenvolvida e a parte de aquisição de dados dos sensores ultrassônicos via protocolo XCP, seja por meio de ASAP3 ou COM. Nessa imagem foi considerado o dado de eco direto do sensor externo esquerdo (sensor 2).

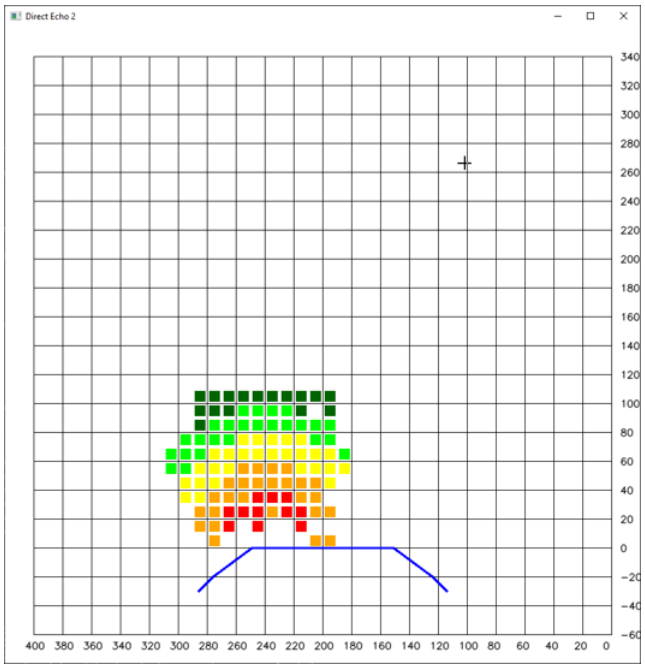
É necessário dizer que o usuário tem a possibilidade de escolher o sinal de interesse (eco direto ou *cross* eco) e o sensor que se analisará, entretanto, a ECU também entrega sinais que visualizam todo o FoV gerado pelo somatório desses ecos ao longo do para-choque, que pode ser preterido pelo usuário.

2.3.2 Geração on-line do FoV em C++

A última versão do *software* de geração do FoV em C++ obteve um grande avanço em relação à realização do teste em si. Anteriormente havia a necessidade do engenheiro verificar os dados de eco por meio do CANape e ou do CANoe e inserir os dados em uma planilha. Nesse âmbito, foi criado um *software* que faz uma interação com o usuário por meio de uma janela do Windows e pede os dados do projeto como entrada e dá comandos ao engenheiro de aplicação sobre a movimentação do *isopole*, seguindo os requisitos 2.2.1.

O processamento do código envolve obter as entradas de confirmação da movimentação do *isopole* e obter os dados de eco por meio do protocolo de comunicação ASAP3. Esses dados, retirados do *buffer* FIFO, são armazenados em uma matriz com colunas dependentes dos números de sinais de eco, e linhas dependentes do comprimento de interesse do FoV. A matriz é

Figura 8 – *Geração do FoV Dinâmico através do software de aplicação*



Fonte: Retirado da aplicação deste projeto, resultante de 2.3.2 e 2.3.3

subsequentemente transcrita em um arquivo de texto, com um cabeçalho em função dos dados inseridos pelo engenheiro, e os dados contendo a matriz de eco. A partir disso, esse arquivo é lido pela interface gráfica desenvolvida anteriormente e o usuário escolhe qual padrão de eco quer analisar.

Não obstante, a última versão faz o processamento do sinais via comunicação ASAP3 e a saída visual serem realizados de forma paralela, ou seja, além do processo descrito no último parágrafo, houve pela primeira vez a visualização on-line (em tempo real) da geração do FoV (ilustrada na figura 8) utilizando a chamada da função desenvolvida em Python paralelamente à aplicação C++.

É necessário salientar que o ambiente necessário para compilar o código é o Microsoft Visual Studio C++ 2008 e há arquivos próprios, como o CANapeAPI.h, e algumas configurações próprias da IDE (*Integrated Development Environment*). A necessidade da recompilação do executável reside na integração com a parte gráfica, o que envolve a chamada da função e do ambiente em Pyhton.

2.3.3 Geração on-line do FoV em Python

Inicialmente, a geração on-line do FoV em Python baseou-se principalmente nas funções já implementadas em C++ com algumas diferenças. Primeiramente, para executar o programa (com extensão .py) é necessário abrir um arquivo com extensão .bat que chama o a aplicação e o local de instalação do python e instala as bibliotecas necessárias a partir de um arquivo .yml.

Ademais, nessa versão escolheu-se o protocolo de comunicação COM (ASAM 3 MCD) para retirar os dados de eco mapeados no arquivo .a2l, que é feito de forma mais simplificada, sem necessidade de configurações adicionais, somente uma biblioteca próprio. A causa da simplificação reside na ampla utilização desse protocolo em diversas aplicações dentro do sistema Windows. Há um demonstrativo de código base disponível em anexo A.

Nessa versão, como não há a necessidade de se trabalhar com duas aplicações simultâneas, a matriz de eco, que é construída pelo programa, alimenta diretamente a interface gráfica sem a necessidade de escrever dinamicamente um arquivo de texto, diminuindo o esforço de processamento.

Em linhas gerais, o modo de operação dessa versão em Python é similar ao do C++, entretanto a aplicação recebe os dados do usuário por meio do terminal Windows e além disso o arquivo de texto é escrito somente no final do teste, que é posteriormente utilizado como entrada para um *software* de visualização da Bosch.

2.4 Discussão

Como dito anteriormente, cada fase do projeto buscou atender aos requisitos de visualização da geração do FoV em tempo real, cada qual com seu grau de avanço em relação ao anterior, permitindo assim mais assertividade na execução do teste, na análise de defeitos de forma antecipada e na diminuição do retrabalho.

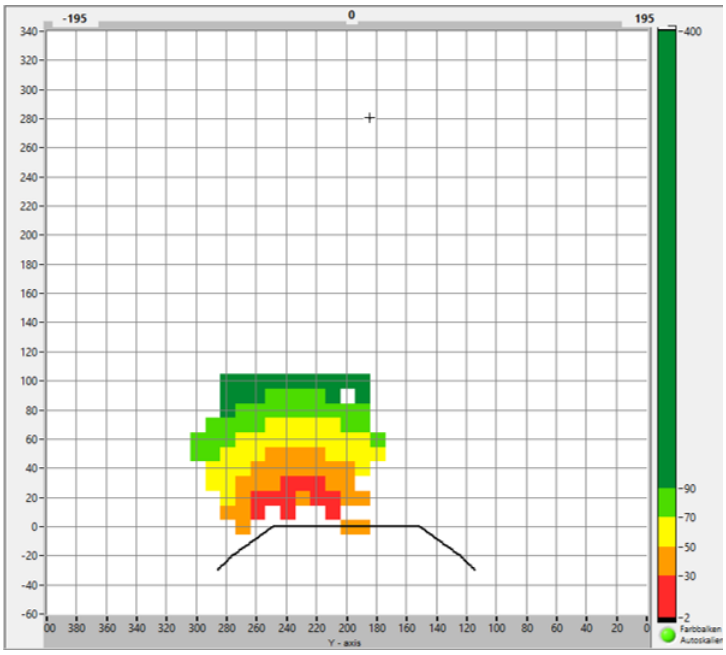
Ao analisar a imagem gerada por um *software* BOSCH na figura 9 (mesma medição da anterior) é possível perceber alguns avanços da interface gráfica gerada pelo *software* desse projeto (figura 8) em relação a anterior.

A interface gráfica delimita melhor a região dos quadrados coloridos (dados de detecção de eco dos sensores) e os discretiza melhor na figura (quadrados 6x6 ao invés de 10x10), o que permite visualizar e analisar cada sinal mais facilmente. Outrossim, como pode ser visto na figura 9, a mesma tem detecções em cima da linha do para-choque, que é um problema de *offset*, o qual foi corrigido na figura 8 e também foi atualizado o *offset* no sentido horizontal para facilitar essa identificação.

A implementação em *Python* buscou manter a fidedignidade do projeto em possibilitar a escolha do intervalo de distância das zonas (cores do FoV) por meio da entrada via teclado pelo usuário, pois cada montadora tem sua peculiaridade quanto à escolha do intervalo das zonas de detecção.

Em segunda análise é possível identificar o poder da ferramenta *online*, ao se identificar detecção de segunda zona (laranja) em regiões que deveriam sem de primeira zona (vermelha) na imagem 8. Na ferramenta *offline*, o aplicador só perceberia a leitura incongruente após finalizar o teste, que dura cerca de 4 horas. Na ferramenta *online* implementada, o engenheiro aplicador consegue identificar essa leitura indesejada em sua primeira aparição e tentar diagnosticar qual

Figura 9 – *Geração do FoV a partir de um arquivo de texto através de um software Bosch*



Fonte: Retirado de um teste em um veículo

sua causa raiz. Nesse caso em específico, a instalação do sensor no para-choque induzia o efeito de *back propagation*, que é o aprisionamento das ondas trasmitidas na própria estrutura do parachoque (placas por exemplo) e retorno posterior, com um *TOF* condizente à uma distância de segunda zona. Esse efeito é indesejado em vista que indica de forma permanente um obstáculo presente em segunda zona, ou seja, traz alertas falsos-positivos para o sistema.

Além disso, no terminal de interação do usuário com o programa, além da instrução para movimentar o *isopole* sobre o tapete, o programa retorna a última medição de todos os sinais de interesse, o que possibilita uma melhor análise do valor, que ratifica a cor da zona de detecção ilustrada na interface gráfica.

De outro lado, destaca-se o avanço do ambiente de programação e da mudança do protocolo ASAP3 para o protocolo COM. Esse comparativo é mais bem ilustrado pela tabela:

Tabela 3 – Tabela de comparação entre implementações em C++ (2.3.2) e em *Python* (2.3.3)

	Anterior	**Atual**
Linguagem	C++	Python
Ambiente	Microsoft C++ 2008	VSCode
Adaptabilidade	Baixa	Alta
Amigabilidade ao usuário	Baixa	Alta
Visulização online	Nenhuma	Tempo real
Retomada da medição	Não possível	Possível

Fonte: Desenvolvida pelo autor

A mudança da linguagem C++ juntamente com seu ambiente de desenvolvimento (IDE) para Python e o ambiente VSCode diminuem a complexidade para o desenvolvedor, uma vez que é uma linguagem mais conhecida pelos desenvolvedores e por seu ambiente ser mais adaptável, tanto em formatação quanto na disponibilidade de suporte.

A adaptabilidade é crucial para um bom funcionamento do projeto, por razão da linguagem *Python* ser uma linguagem interpretada, o aumento da facilidade para se programar possibilita que mais pessoas possam entender o código e partir dele fazer melhorias e alterações que julgarem necessário no futuro. Além disso, as bibliotecas que eram necessárias no protocolo ASAP3 (programado com C++) são mais facilmente importadas na versão COM (programado em Python), com o auxílio de um ambiente virtual próprio para o projeto.

A amigabilidade com o usuário é fundamental para uma boa execução do teste, e para isso foi implementado itens de robustez ao código, como o uso dos comandos *try-except* para prevenir que erros de entrada induzam a finalização do *software*. Além disso, como explicitado em 2.3.3, um arquivo .bat que chama a aplicação do código na Prompt de comandos também foi bolado de forma que facilite a execução pelo engenheiro aplicador.

Por fim cumpriu-se a visualização on-line em ambos os protocolos (ASAP3 e COM) que trouxe todos os benefícios mencionados como: rápida análise de sinais indesejados, diminuição do retrabalho, aumento da assertividade do teste - além de possibilitar retomar medições anteriores incompletas.

3 CONCLUSÃO

A implementação de resoluções como a 759 do CONTRAN [Brasil. CONTRAN 2018] estimulam e direcionam o desenvolvimento de novas tecnologias e aumentam o conhecimento de suas funcionalidades e seus critérios de performance. Para tal, o teste do FoV propicia à montadora a garantia to atendimento ou não dos requisitos estabelecidos pela mesma.

Portanto, este projeto justificou-se sobre a óptica de melhorar a assertividade e diminuir o retrabalho do engenheiro aplicador na execução deste teste. Além de possibilitar a visualização em tempo real, também possibilitou ao engenheiro salvar o estado do teste, o que possibilita a análise mais cuidadosa dos sensores ultrassônicos pelo engenheiro.

Ademais, os protocolos estudados foram alicerce desta aplicação, cada qual com sua função no sistema, e pode-se concluir que a versão final (protocolo COM) mostrou-se superior à anterior (ASAP3), na qual era necessário incluir diversas bibliotecas no ambiente de compilação. Além disso, sua programação em *Python* facilita a visualização para um futuro programador e também dinamiza a importação das bibliotecas inerentes ao projeto. A interface gráfica, confeccionada utilizando-se a biblioteca OpenCV, foi determinante para a visualização dos dados, de forma que possibilitou uma adequação de resolução à tela de projeção e possibilitou a discretização dos ecos.

Nesse sentido, ao considerar o projeto como realidade na aplicação de engenharia cotidiana, a escolha de uma IDE (VSCode) amigável e difundida e de uma linguagem interpretada possibilita o desenvolvimento de novas ferramentas para o *software*, de forma que aumente a colaboração e possa ser aplicado em outras áreas, tais como: câmera dianteira, câmera de ré, radares e lidares. Os sinais de medição aquisitados via protocolo COM sobre o protocolo XCP podem ser implementados em qualquer ECU XCP que possua os arquivos de descrição de memória (.a2l), e a partir disso o desenvolvedor da aplicação poderá utilizá-los para quaisquer fins de interesse, ilustrando em gráficos, tabelas e diagramas.

Sob uma óptica futura, e considerando-se uma nova geração de sensores ultrassônicos que necessitam de movimento ao encontro do obstáculo, pode-se vislumbrar uma implementação adicional ao *software* desenvolvido, de forma que um veículo remoto envie sinais COM de posição e movimento, e este compute os sinais de eco recebidos pelo para-choque. Dessa forma, os dados de eco serão mais fidedignos às situações atuais, em que o para-choque movimenta-se em direção ao obstáculo e vice e versa. Dessa forma o projeto pode ser continuado e seu aprimoramento pode ser conduzido tanto por mim quanto por outra pessoa que tenha familiaridade com *software*, e assim a ferramenta continuará se atualizando ao longo tempo, cumprindo com os requisitos que será submetida.

REFERÊNCIAS

BORENSTEIN, J.; KOREN, Y. Error eliminating rapid ultrasonic firing for mobile robot obstacle avoidance. **IEEE Transactions on Robotics and automation**, IEEE, v. 11, n. 1, p. 132–138, 1995.

BRASIL. CONTRAN. Resolução contran nº 759, de 20 de dezembro de 2018. **Diário Oficial da União**, Brasília, 20 de dezembro de 2018.

BROOKHUIS, K. A.; WAARD, D. D.; JANSSEN, W. H. Behavioural impacts of advanced driver assistance systems–an overview. **European Journal of Transport and Infrastructure Research**, v. 1, n. 3, 2001.

CENKERAMADDI, L. R. *et al.* A survey on sensors for autonomous systems. *In*: IEEE. **2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)**. [*S.l.: s.n.*], 2020. p. 1182–1187.

CHEN, C. H.; HSU, C. W.; YAO, C. C. A novel design for full automatic parking system. *In*: IEEE. **2012 12th International Conference on ITS Telecommunications**. [*S.l.: s.n.*], 2012. p. 175–179.

EVERETT, C. H. Survey of collision avoidance and ranging sensors for mobile robots. **Robotics and Autonomous Systems**, v. 5, n. 1, p. 5–67, 1989. ISSN 0921-8890. Disponível em: https://www.sciencedirect.com/science/article/pii/0921889089900419.

HAUPTMANN, P.; HOPPE, N.; PÜTTMER, A. Application of ultrasonic sensors in the process industry. **Measurement Science and Technology**, IOP Publishing, v. 13, n. 8, p. R73, 2002.

KINSLER, L. E. *et al.* **Fundamentals of acoustics**. [*S.l.: s.n.*]: John wiley & sons, 2000.

LU, W. *et al.* A general fault injection method based on jtag. *In*: **2018 Prognostics and System Health Management Conference (PHM-Chongqing)**. [*S.l.: s.n.*], 2018. p. 604–608.

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

PARK, W.-J. *et al.* Parking space detection using ultrasonic sensor in parking assistance system. *In*: IEEE. **2008 IEEE intelligent vehicles symposium**. [*S.l.: s.n.*], 2008. p. 1039–1044.

PATZER, R. Z. A. **The Standard Protocol for the Embedded Development**: Basics and areas of application. [*S.l.: s.n.*], 2022. Status 2022. Disponível em: https://cdn.vector.com/cms/content/application-areas/ecu-calibration/xcp/XCP_Book_V1.5_EN.pdf. Acesso em: 15 set. 2023.

PIAO, J.; MCDONALD, M. Advanced driver assistance systems from autonomous to cooperative approach. **Transport reviews**, Taylor & Francis, v. 28, n. 5, p. 659–684, 2008.

STANDARDIZATION, I. O. for. **ISO 11898: Road vehicles - Controller Area Network (CAN)**. [*S.l.*], 2015.

STANDARDIZATION, I. O. for. **ISO 16787: Intelligent transport systems - Assisted parking system (APS) - Performance requirements and test procedures**. [*S.l.*], 2017.

STANDARDIZATION, I. O. for. **ISO 17386: Transport information and control systems - Manoeuvring Aids for Low Speed Operation (MALSO) - Performance requirements and test procedures**. [*S.l.*], 2018.

TIGADI, A. *et al.* Advanced driver assistance systems. **International Journal of Engineering Research and General Science**, v. 4, n. 3, p. 151–158, 2016.

Vector. **Vector E-Learning**. Accessed on September 13, 2023. Disponível em: https://elearning.vector.com/mod/page/view.php?id=333.

WANG, W. *et al.* Automatic parking of vehicles: A review of literatures. **International Journal of Automotive Technology**, Springer, v. 15, p. 967–978, 2014.

WEI, D. C. M. **Método de desvio de obstáculos aplicado em veículo autônomo.** 2015. Tese (Doutorado) — Universidade de São Paulo, 2015.

ANEXOS

ANEXO A – CÓDIGO DE AQUISIÇÃO DE DADOS COM (ASAM MCD 3) VIA PROTOCOLO XCP

Código distribuído pela VECTOR Gmbh. que caracteriza as duas funções de aquisição de dados via protocolo XCP de uma ECU.

Listing A.1 – Código de aquisição de dados COM (ASAM MCD 3) via protocolo XCP

```
# This example was implemented using Python v3.7 (x64)
# Python method used to access CANape COM Interface
# is win32com.client.Dispatch()

import win32com.client  # This is mandatory
import winreg  # Access to winreg
import platform  # 32Bit or 64Bit
import time  # time.sleep(x)


def getCnpExamplePath():
    # Get CANape installation path from the registry
    hKey = winreg.OpenKey(
        winreg.HKEY_LOCAL_MACHINE,
        "SOFTWARE\\VECTOR\\CANape",
        0,
        winreg.KEY_READ | 0x100,
    )
    basePath = winreg.QueryValueEx(hKey, "Path")[0]
    winreg.CloseKey(hKey)
    # Read registry key for the examples from vector.ini
    regKeyExamples = 0
    vectorini = open(basePath + r"vcustom.ini", "r")
    for line in vectorini.readlines():
        if "CANapeOnlineExamplesRegKey" in line:
            regKeyExamples = (
                line.split("=")[1]
                .replace("HKEY_LOCAL_MACHINE\\", "")
                .replace("\n", "")
                .replace("Path", "")
            )
```

```
            break
    # Get the path to the CANape online examples
    hKey = winreg.OpenKey(
        winreg.HKEY_LOCAL_MACHINE,
        regKeyExamples,
        0,
        winreg.KEY_READ | 0x100,
    )
    basePath = winreg.QueryValueEx(hKey, "Path")[0]
    if basePath.endswith("\\"):
        basePath = basePath[0 : len(basePath) - 1]
    winreg.CloseKey(hKey)
    return basePath


# platform things
if platform.architecture()[0] == "32bit":
    print("Python x32 environment is used")
else:
    print("Python x64 environment is used")

# Initialize the CANape COM Interface
canape = win32com.client.Dispatch("CANape.Application")

# Prepare the values to open the project
cnpWorkingDir = getCnpExamplePath() + r"\XCPDemo"
debugMode = 1
COMTimeout = 100000
clearDevList = 0
modalMode = 0
# Fifo and sample size have to be set before CANape is loaded
canape.Measurement.FifoSize = 2048
canape.Measurement.SampleSize = 1024

# Open the project given
canape.Open2(
    cnpWorkingDir,
    debugMode,
    COMTimeout,
```

```
    clearDevList,
    modalMode,
    1,
)

# Wait for XCPsim simulator to be ready
time.sleep(3)

# Init. an (existing) device
myXCPsim = canape.Devices.Add(
    "XCPsim", "XCPsim.a2l", "XCP", 1
)

# Get and print out DAQ events supported by the ECU
print("-----------------------------------------------")
print(
    "{:<20}{:^2}{:^5}{:^2}{:^}".format(
        "Description", "|", "ID", "|", "SamplingTime"
    )
)
print("-----------------------------------------------")
for singleTask in myXCPsim.Tasks:
    print(
        "{:<20}{:^7d}{:^14d}".format(
            singleTask.Name,
            singleTask.ID,
            singleTask.SamplingTime,
        )
    )
print("-----------------------------------------------")

# Clear measurement list of all modules
for dev in canape.Devices:
    for task in dev.Tasks:
        if task.Channels.Count > 0:
            task.Channels.Clear()

# Configure measurement list for the XCP device
measObjects_XCPsim = ["channel1", "channel2", "channel3"]
```

```
# -> First select and get the DAQ event to be used
taskSel = myXCPsim.Tasks("100ms")
# -> Finally insert the signals into the measurement list
for signal in measObjects_XCPsim:
    taskSel.Channels.Add(signal)
    # Define the process to save the signal into the measurement file
    taskSel.Channels(signal).Save2MDF = True

# Check the contents of the current measurement list (of all devices)
print("================")
print("MEASUREMENT LIST")
print("================")
print("-----------------------------------------")
print(
    "{:<10}{:^5}{:^10}{:^5}{:^}".format(
        "Device", "ObjectName", "taskId", "rate", "SaveFlag"
    )
)
print("-----------------------------------------")
for dev in canape.Devices:
    measSigsForDev_empty = True
    for task in dev.Tasks:
        if task.Channels.Count > 0:
            measSigsForDev_empty = False
            for channel in task.Channels:
                print(
                    "{:<}{:<}{:<2}{:^12}{:^8d}{:^8d}{:^5d}".format(
                        "[",
                        dev.Name,
                        "]",
                        channel.Name,
                        task.ID,
                        task.SamplingTime,
                        channel.Save2MDF,
                    )
                )
    if measSigsForDev_empty:
        print(
            "{:<}{:<}{:<2}{:<2}".format(
```

```
                "[", dev.Name, "]", "No signals configured"
            )
        )

# Start the measurement
canape.Measurement.Start()

# Wait until a measurement value has arrived in the Fifo
while taskSel.FifoLevel == 0:
    continue

# Fetch measurement values
print("-------------------------------------")
print("---------Measurement VALUES----------")
print("-------------------------------------")
for a in range(0, 100):
    # (1) ITask.CurrentValues()
    #     Using CurrentValues means to get a snapshot of the very last
    # (actual) entry of the Fifo.
    #     It won't change the contents of the Fifo. The Fifo would work
    # automatically (to first delete the oldest entry and then to let a
    oCurrentValue = taskSel.CurrentValues()
    txt = (
        "[FifoLvl: "
        + str(taskSel.FifoLevel)
        + ", TS: "
        + str(oCurrentValue[1])
        + "] [DEC] @"
        + str(taskSel.SamplingTime)
        + "ms: "
    )
    for a in range(0, taskSel.Channels.Count):
        txt = txt + "(" + str(oCurrentValue[0][a]) + ")"
    print(txt)
    # Task.NextSample()
    # Using NextSample means to fetch/take out an entry of the Fifo.
    # Every call of NextSample() reduces the number of entries inside o
    # -> Too many calls of NextSample() at a specific short time would
    # a COM expection "Fifo doesn't contain any value".
```

```
188     # -> On the other hand, the Fifo would overflow and the COM exception "
189     # detected" would be thrown out IF NextSample() is not called
190     #    with a proper time interval
191     try:
192         oNextSample = taskSel.NextSample()
193         if oNextSample:
194             txt = (
195                 "[FifoLvl: "
196                 + str(taskSel.FifoLevel)
197                 + ", TS: "
198                 + str(oNextSample[1])
199                 + "] [DEC] @"
200                 + str(taskSel.SamplingTime)
201                 + "ms: "
202             )
203             for a in range(0, taskSel.Channels.Count):
204                 txt = (
205                     txt + "(" + str(oNextSample[0][a]) + ")"
206                 )
207             print(txt)
208     except Exception as inst:
209         print(inst)
210     time.sleep(float(taskSel.SamplingTime) / 1000)
211 print("-------------------------------------")
212 
213 # Stop the measurement
214 canape.Measurement.Stop()
215 
216 # Exit the COM connection and close CANape
217 print("Closing CANape..")
218 canape.Quit()
```