UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

YU ZHIPENG

DESENVOLVIMENTO DE UM DATALOGGER DA REDE CAN VEICULAR VIA RASPBERRY PI

CURITIBA

2024

YU ZHIPENG

DESENVOLVIMENTO DE UM DATALOGGER DA REDE CAN VEICULAR VIA RASPBERRY PI

DEVELOPMENT OF A VEHICULAR CAN NETWORK DATALOGGER VIA RASPBERRY PI

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientadora: Profª . Drª Rosângela Bach Rodrigues dos Santos

CURITIBA

2024

YU ZHIPENG

DESENVOLVIMENTO DE UM DATALOGGER DA REDE CAN VEICULAR VIA RASPBERRY PI

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do curso de Engenharia Elétrica da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 13 de Junho de 2024

Rosângela Bach Rodrigues dos Santos
Doutorado
Universidade Tecnológica Federal do Paraná

Elder Oroski
Doutorado
Universidade Tecnológica Federal do Paraná

Glauber Gomes de Oliveira Brante
Doutorado
Universidade Tecnológica Federal do Paraná

CURITIBA

2024

RESUMO

A rede CAN veicular juntamente com o protocolo CAN tornou possível a comunicação intersistêmica entre unidades de controle eletrônicos automotivas de uma maneira mais simultânea e organizada. Com o avanço das técnicas e tecnologias automotivas, os microcontroladores aumentam de número, assim aumentando o número de acionamentos e utilitários nos veículos. Neste projeto foi desenvolvido uma ferramenta de leitura da rede CAN veicular utilizando um Raspberry Pi, permitindo a leitura de dados obtidos em tempo real. Foi desenvolvida para essa ferramenta, uma interface para o usuário identificar os elementos obtidos através da leitura dos dados da rede CAN. Foi executada a simulação e teste desta ferramenta para a validação em caso real. Os resultados nos testes e simulações mostraram a possibilidade para a utilização real em validação de rotinas e baterias de testes.

Palavras-chave: rede CAN; raspberry pi; datalogger.

ABSTRACT

The vehicle CAN network together with the CAN protocol has made intersystem communication between automotive electronic control units possible in a more simultaneous and organized manner. With the advancement of automotive techniques and technologies, microcontrollers increase in number, thus increasing the number of drives and utilities in vehicles. In this project, a vehicle CAN network reading tool was developed using a Raspberry Pi, allowing the reading of data obtained in real time. An interface was developed for this tool for the user to identify the elements obtained by reading data from the CAN network. The simulation and testing of this tool was carried out for validation in a real case. The results in tests and simulations showed the possibility of real use in validating routines and test batteries.

Keywords: canbus ; raspberry pi; can logger.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Desde a invenção do primeiro automóvel moderno em 1886, um carro de motor monocilíndrico movido a gasolina, criado por Karl Benz (ECKERMANN, 2001), a humanidade não conteve esforços para aperfeiçoar a tecnologia automobilística. É normal atualmente serem encontrados carros que tenham centrais multimídias, vidros elétricos, sistemas de injeção automática, caixa de câmbio automático e diversos outros sistemas eletroeletrônicos. Funcionalidades que são essenciais para os funcionamentos dos carros, ou, um luxo a mais para o conforto dos consumidores que compram o veículo.

Nesta ideia de aumentar a quantidade de funções dos componentes eletroeletrônicos veiculares, a rede *Controller Area Network* (CAN) desenvolvida pela Bosch Gmbh, tem a funcionalidade de ser a rede de comunicação intersistêmica dos eletroeletrônicos veiculares (CAPELLI, 2010). Nesta rede, são conectados os microcontroladores que acionam as funções elétricas veiculares, como a função dos vidros elétricos.

Com a implementação da rede CAN, pode-se identificar e estudar como cada microcontrolador irá mandar e receber a sua mensagem de comunicação, possibilitando também, a coleta destes dados para analisar e interpretar possíveis anomalias e falhas intersistemas (ELETRONICS, 2022). Desta forma, segundo Guimarães (2007), a coleta de dados para o diagnóstico de falhas, para projetos que visam implementar mais funções elétricas, pode minimizar futuros custos não técnicos.

> Ferramentas de diagnose são fundamentais durante o desenvolvimento de novos veículos e sistemas eletrônicos, assim como durante a realização dos procedimentos de revisão e manutenção (GUIMARÃES, 2007, p. 227).

Com os microcontroladores veiculares, é possível a utilização de ferramentas que coletam os dados intersistêmicos veiculares daqueles dispositivos, em que, as mensagens e dados coletados da rede CAN mostram as condições atuais do veículo, tais como falhas e erros sistêmicos presentes.

No mesmo canal de leitura pode-se obter os erros históricos, que são como o nome diz, um registro histórico das falhas passadas, sendo elas uma base para a correção de falhas futuras (ELETRONICS, 2022).

Ao analisar os microcontroladores veiculares, interligados entre si com a rede CAN, é importante interpretar as mensagens trocadas entre eles. Existem tecnologias específicas para a coleta de dados veicular, como os *dataloggers*. Muitas delas já patenteadas por empresas e desenvolvidas em conjunto com *softwares* que interpretam estes dados (GUIMARÃES, 2007).

Com a obtenção dos dados dos modos de falhas, a identificação da origem da anomalia será mais fácil, tornando mais eficiente o trabalho dos pilotos de testes. Além do que, com os dados, projetos automobilísticos futuros terão uma base para contornar as falhas. Quanto mais simples o entendimento dos dados, mais objetivo será para o desenvolver ou projetar futuros microcontroladores, assim como a simplicidade nos testes veiculares.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo deste trabalho é desenvolver uma ferramenta de coleta de dados de mensagem da rede CAN veicular utilizando Raspberry Pi.

1.1.2 Objetivos específicos

São indicados os seguintes objetivos específicos deste TCC:

a) Desenvolver o circuito de coleta de dados da rede CAN, utilizando o MCP2515 para a interpretação de dados tendo o Raspberry Pi, como o microcontrolador;
b) Programar por meio do sistema operacional do Raspberry Pi, uma interface homem máquina (IHM);
c) Utilizar um *Liquid Crystal Display* (LCD) conectado ao Raspberry Pi para iniciar e finalizar a coleta de dados de mensagens da rede CAN.

1.2 Justificativa

Este trabalho de conclusão de curso é uma continuação de um projeto iniciado durante o estágio que visa o desenvolvimento de uma ferramenta alternativa para a coleta de dados veiculares.

A rede CAN é comumente atrelada à indústria automobilística, seja em projetos ou pesquisas ao desenvolver e ampliar microcontroladores veiculares. Com o desenvolvimento de um *datalogger* da rede CAN veicular via Raspberry Pi é possível coletar dados transmitidos pelos sensores, atuadores e pelas Unidades de Controle Eletrônicas (UCE), tendo um baixo custo em relação aos *dataloggers* do mercado.

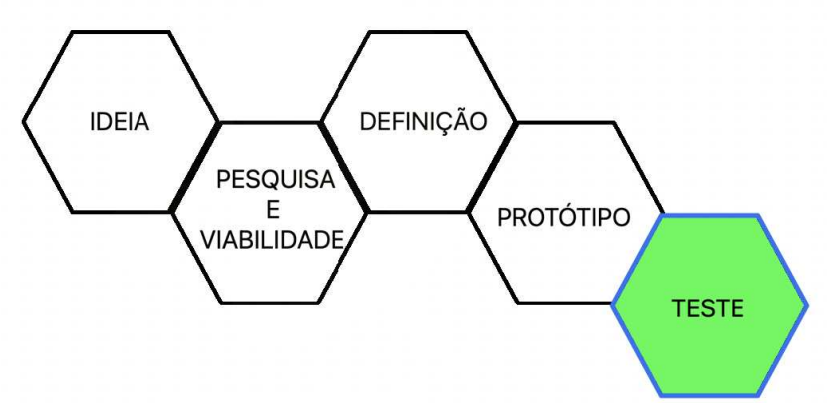
1.3 Metodologia

Para este trabalho de conclusão de curso foi adotada a metodologia ágil e a metodologia de inovação em gestão de projetos, elas são direcionadas ao desenvolvimento de produtos novos com a certeza de sua finalidade.

Os métodos ágeis permitem o desenvolvimento de cada ciclo do projeto até a sua fase de maturação, assim, os processos subsequentes não são afetados pela etapa anterior (ADJEI, 2009). Nesta metodologia, são aplicados testes de validação alinhadas as opiniões e *feedback* dos participantes do projeto para a conclusão.

Já a metodologia de inovação elabora as fases desde a idealização do processo até a prototipagem do projeto (ADJEI, 2009). Na Figura 1 é aplicado o primeiro esboço do método da inovação para a definição de cada fase do projeto, desde o surgimento da ideia até a elaboração do protótipo, assim no fim, para concluir o projeto, é utilizado o método ágil para a realização dos testes de validação.

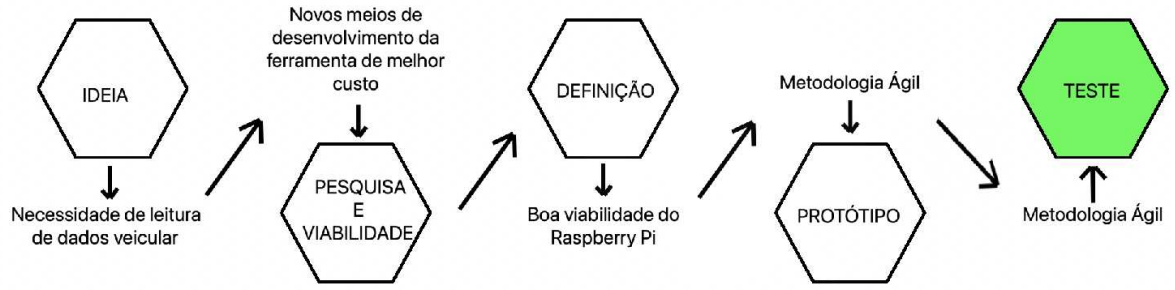
Figura 1 - As fases do *thinking* do projeto.



Fonte: Autoria própria (2023).

Com as fases bem definidas, a Figura 2 mostra o fluxograma das etapas do projeto associado às fases desenvolvidas com a implementação do método ágil.

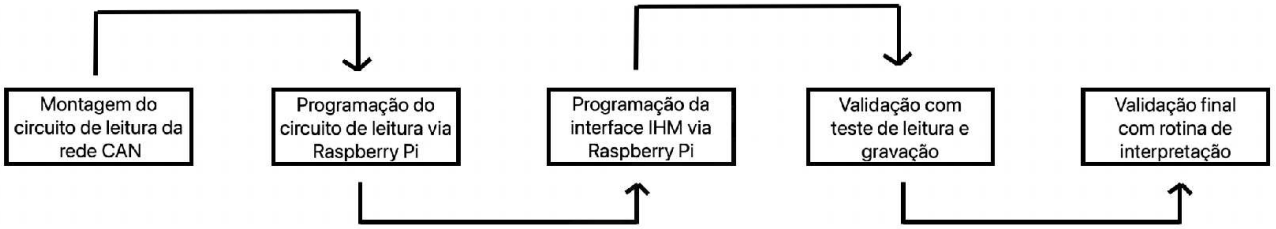
Figura 2 - Fluxograma do projeto.



Fonte: Autoria própria (2023).

Uma vez definidas as etapas do projeto, é mostrado na Figura 3 o fluxograma do desenvolvimento do protótipo.

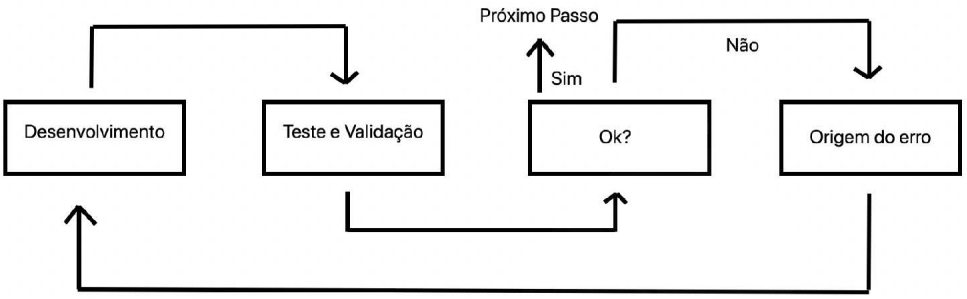
Figura 3 - Fluxograma de prototipagem.



Fonte: Autoria própria (2023).

Alinhado à prototipagem do projeto, na Figura 4 é mostrado o fluxograma do desenvolvimento a partir dos testes.

Figura 4 - Fluxograma de teste.



Fonte: Autoria própria (2023).

Com isso, tem-se a metodologia ágil e de inovação, na qual o progresso do projeto é alcançado depois da conclusão das etapas a partir dos resultados de teste e desenvolvimento, permitindo que cada etapa do projeto seja testada e validada.

1.4 Materiais

Para este projeto, foi realizada a montagem do circuito da leitura de dados, com o MCP2515 e o Raspberry Pi. Estão integrados ao circuito, os componentes interativos para a coleta de dados e o LCD. Estes dispositivos são explicados nos itens 2.4, 2.6, 2.7, respectivamente. Já na programação do IHM, foi utilizado o Visual Studio Code, um editor de código fonte utilizando a linguagem Python.

Os testes foram realizados na Plataforma de Integração Eletroeletrônica (PIE), mostrado na Figura 5. Essas PIEs são as bancadas de teste eletrônicas veiculares utilizadas pelas montadoras.

Figura 5 - Plataforma de Integração Eletroeletrônica (PIE).

Fonte: Autoria própria (2023).

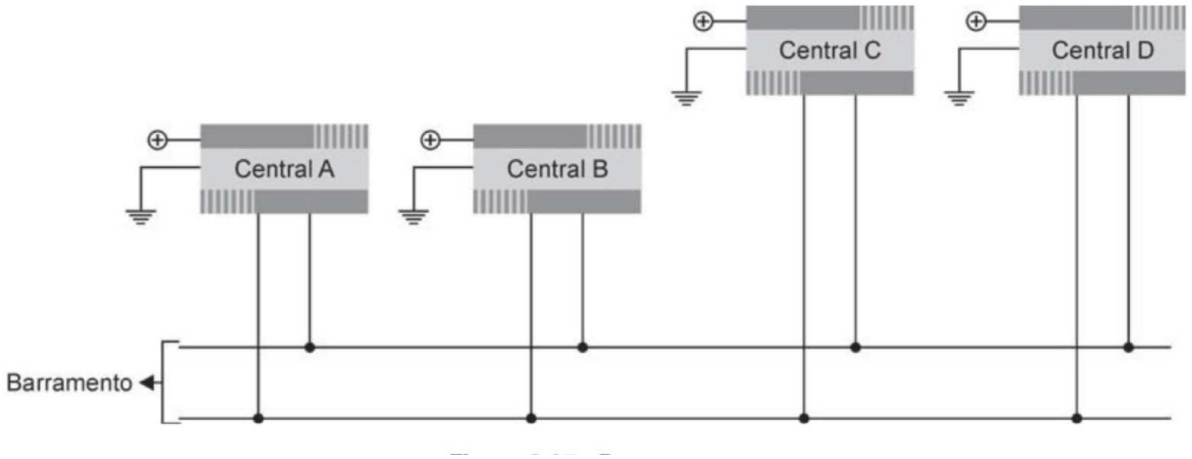
2 FUNDAMENTAÇÃO TEÓRICA

É abordada nesta seção as bases teórica utilizada para o TCC, utilizando as referências em artigos e livros a respeito do assunto.

2.1 Protocolo CAN

A rede CAN é o principal meio de comunicação entre as unidades de comando dentro de um automóvel. São as redes de comunicação eletroeletrônicas entre as Unidades de Controle Eletrônicas (UCE). O número de UCEs difere para cada projeto automobilístico e utilizando o protocolo CAN, o número de cabeamento que liga cada UCE no sistema é reduzido significativamente (ELETRONICS, 2022). Na Figura 6 é mostrado como é a disposição física que cada UCE tem com a rede CAN.

Figura 6 - Circuito da rede CAN.



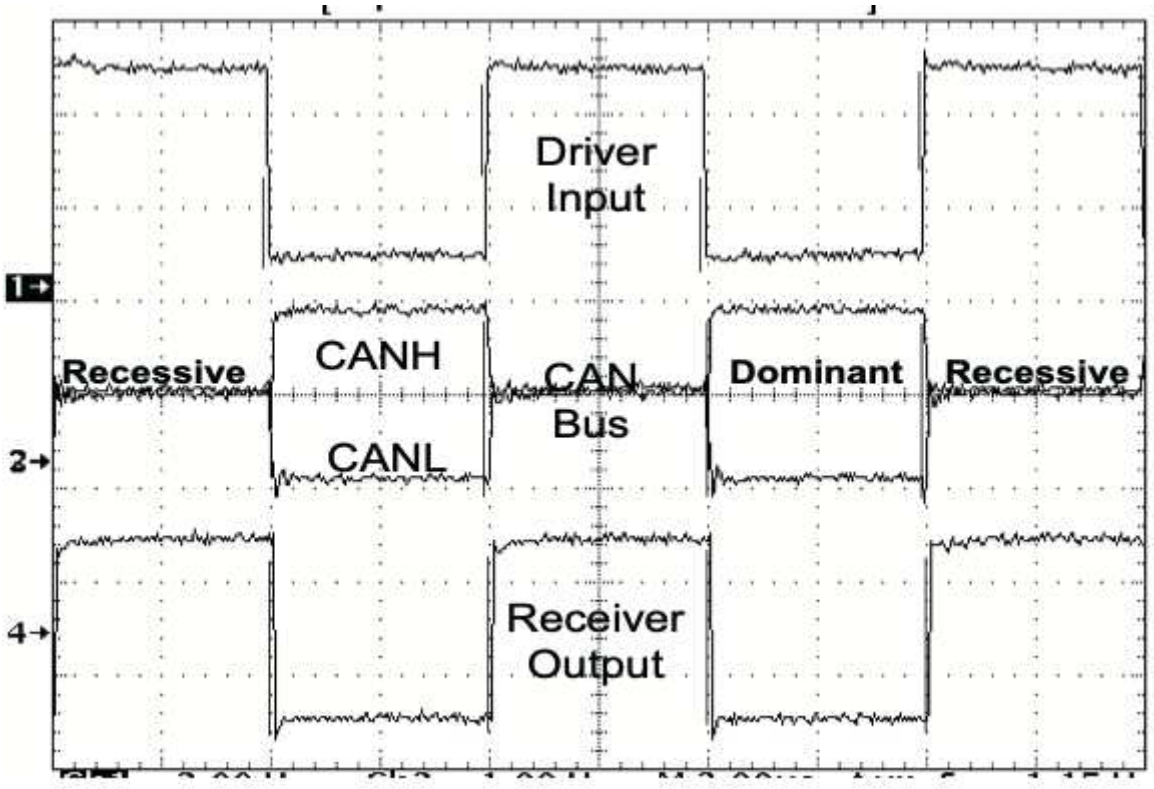
Fonte: Capelli (2010).

Pode-se observar na Figura 6 que cada UCE é ligada à rede CAN em paralelo ao barramento CAN, de modo que cada central envia suas mensagens ao mesmo tempo para o barramento e, ao mesmo tempo, módulos destas centrais recebem e interpretam essas mensagens simultaneamente (CAPELLI, 2010).

2.2 Composição de mensagem CAN

Como é mostrado na Figura 7, o barramento CAN possui ondas sequenciais de mesmo período com picos opostos, denominados de CAN *high* e CAN *low* (CORRIGAN, 2002). Quando os picos estão alinhados em seus zeros de amplitude, tem-se um bit recessivo e quando os picos estão alinhados em suas amplitudes máximas, têm-se os bits dominantes (ELETRONICS, 2022).

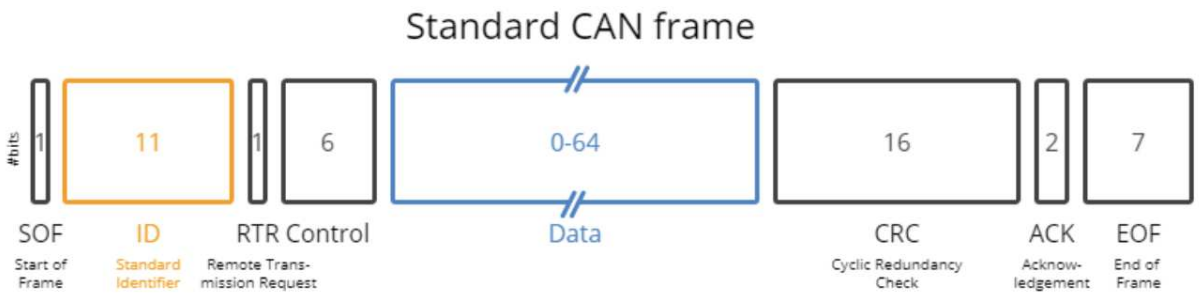
Figura 7 - CAN high e CAN low.



Fonte: Corrigan (2002).

Os bits recessivos formam os uns e os bits dominantes formam os zeros das mensagens binárias da rede CAN. O sequenciamento das mensagens binárias formam os hexadecimais que são a composição das mensagens das UCEs no protocolo CAN. Estas mensagens são denominadas de CAN *frames* (ELETRONICS, 2022), como é mostrada na Figura 8.

Figura 8 - CAN *frame*.



Fonte: Eletronics (2022).

Dentro do protocolo CAN é possível encontrar dois formatos de mensagem que dependem do tamanho de bits, tem-se a CAN 2.0A com o identificador de 11 bits e a CAN 2.0B, com o identificador de 29 bits, no caso da Figura 8, tem-se uma composição de mensagem de CAN 2.0A, notado pelo ID de 11 bits (ELETRONICS, 2022). Os significados das siglas da Figura 8 são:

1) SOF (*Start of Frame*): é a parte inicial da mensagem, indicará em qual rede CAN está a mensagem. Nos automóveis, normalmente tem-se duas redes CAN interligadas entre si, esta parte da mensagem descreve para qual rede CAN irá a mensagem. Esta parte da mensagem terá 1 bit;

2) ID (*Identification*): é a parte da mensagem responsável pela identificação, ou seja, caracteriza a natureza do pacote de dados enviados. Esta parte da mensagem pode ter até 11 bits;

3) RTR (*Remote Transmission Request*): é a parte da mensagem que indica quando uma rede CAN repassa mensagens enviadas de uma UCE para outra. Isso ocorre pelo fato de que, para o acionamento de um determinado mecanismo, uma UCE precisa da resposta de dois ou mais diferentes UCEs para o acionamento do mecanismo. Esta parte da mensagem tem 1 bit;

4) *Control*: esta é a parte da mensagem que indica o tamanho da mensagem ou tamanho de toda a estrutura de dados enviada, conhecido como *data*. Esta parte da mensagem pode ter até 6 bits;

5) *Data*: esta é a parte da mensagem que possui as informações da mensagem em si, que contém as informações que realmente são enviadas de uma UCE para outra. Esta parte da mensagem pode ter até 64 bits;

6) CRC (*Cyclic Redundancy Check*): é a parte da mensagem que verifica a integridade da mensagem recebida. Esta parte da mensagem pode ter até 16 bits;

7) ACK (*Acknowledgement*): é a parte da mensagem que indicará quando um microcontrolador reconheceu e recebeu a mensagem corretamente, complemento ao CRC. Esta parte da mensagem pode ter até 2 bits;

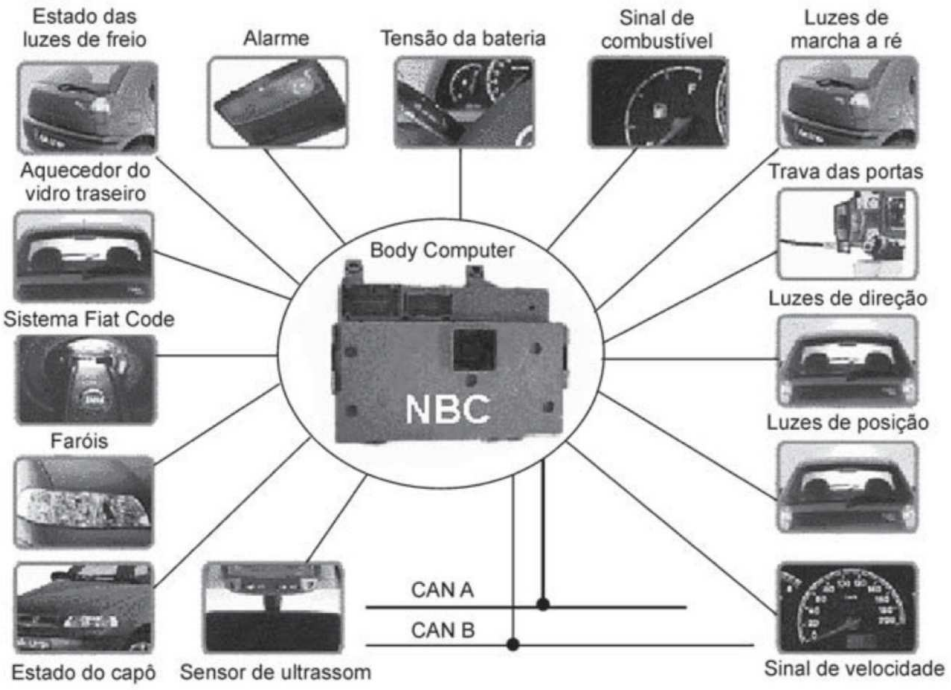
8) EOF (*End of Frame*): parte da mensagem que indica o fim de toda a *frame* de mensagem. Esta parte da mensagem pode ter até 7 bits;

2.3 Unidade de Controle Eletrônica

Uma UCE automotiva é um microcontrolador com um programa gravado em sua memória para desempenhar as respectivas funções projetadas (GUIMARÃES, 2007).

São responsáveis pela leitura das entradas e saídas de dados, gerenciando o funcionamento dos respectivos protocolos de comunicação intersistêmica (GUIMARÃES, 2007). Como é mostrado na Figura 9, esses módulos podem ser os diversos componentes eletrônicos do veículo.

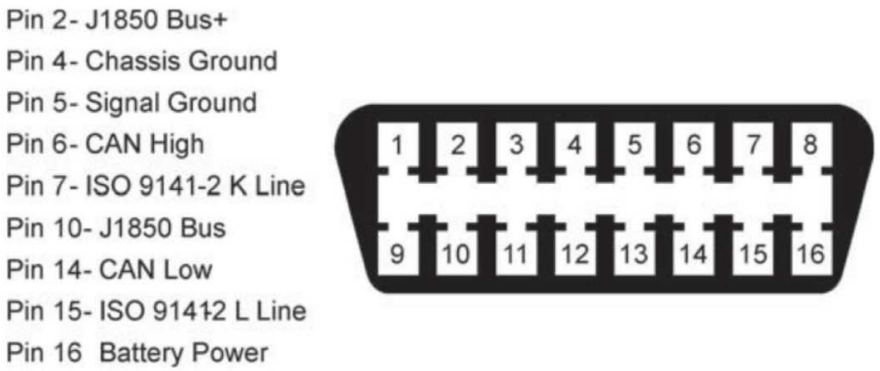
Figura 9 - UCEs de um automóvel.



Fonte: Capelli (2010).

Em paralelo aos UCEs, conectadas ao barramento da rede CAN, tem-se o conector *On-Board Diagnostic* (OBD) padrão II. É neste conector que pode coletar os dados da rede CAN. Na Figura 10 tem-se a imagem do conector OBD II e sua pinagem.

Figura 10 - Conector OBD-II.



Fonte: Capelli (2010).

É a partir do conector OBD-II que é feita a coleta de dados do projeto deste TCC. Os pinos utilizados são 6 e 14, CAN *high* e *low,* respectivamente.

2.4 MCP2515

O transceptor MCP2515 é o responsável pela interpretação dos dados da rede CAN para a formação das mensagens hexadecimais de leitura de dados. Os dados do protocolo CAN são simplificados através do MCP2515 graças aos componentes do microcontrolador que são: o módulo de leitura CAN, registradores e os controladores de lógica (MICROCHIP, 2018).

O protocolo de operação de leitura do transceptor é o *Serial Peripheral Interface* (SPI), usado para a comunicação de um ou mais dispositivos controlados por um microcontrolador mestre (IDEALI, 2021). Na Figura 11, tem-se a ilustração do transceptor.

Figura 11 - MCP2515.

Fonte: Autoria própria (2023).

Os canais do MCP2515 na Figura 11, são:

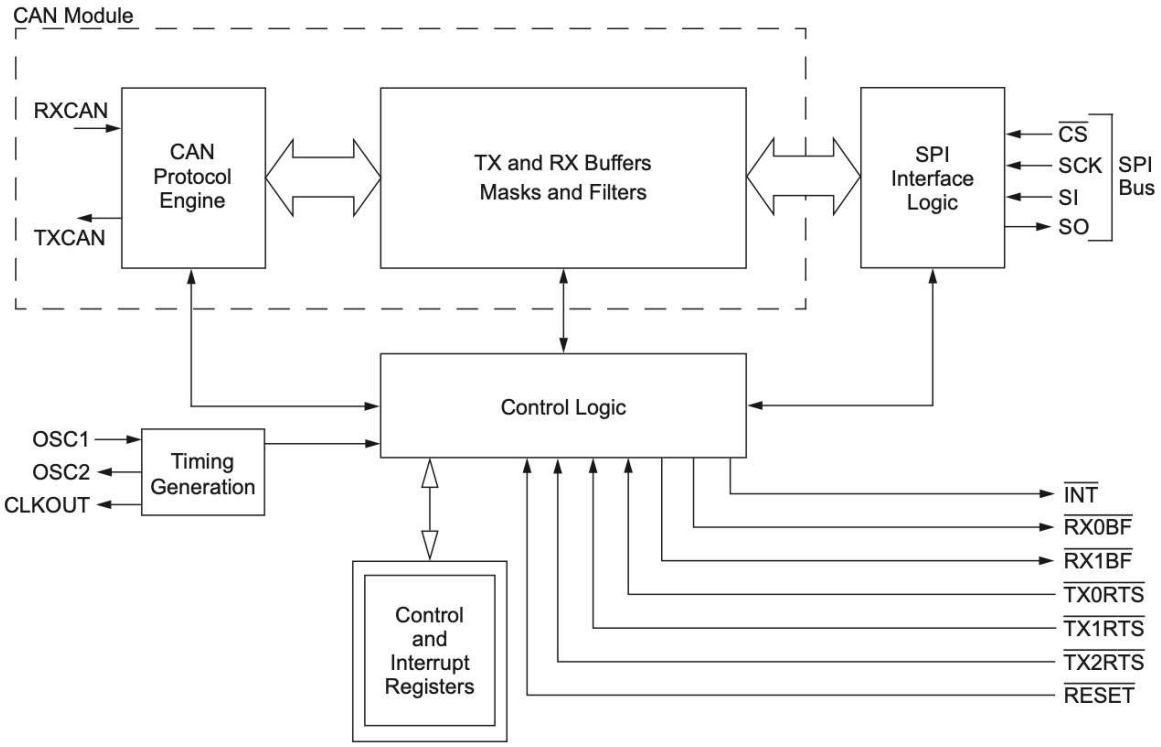
1. CAN H e CAN L: São respectivamente o CAN *high* e CAN *low*, conexão direta com o conector OBD-II;
2. VCC e GND: Pinos de alimentação do MCP2515;
3. INT (*interrupt*): Canal onde as interrupções são habilitadas para a interpretação das formas de onda da rede CAN;
4. SCK (*Serial Clock*): Sincroniza o controlador mestre com escravo por meio do INT e do *clock*;
5. SI (*Master OUT Slave IN*): Pino de recepção dos dados SPI;
6. SO (*Master IN Slave OUT*): Pino de transmissão de dados SPI;
7. CS (*Slave Select*): Prioriza qual controlador escravo recebe os dados.

Pode-se observar na Figura 12 os componentes internos do MCP2515. Os canais RXCAN e TXCAN são responsáveis por captar e transmitir as mensagens da rede CAN enquanto os *buffers* e *filters* adaptam os sinais entre a rede CAN e SPI, tendo a conversão de CAN para SPI e vice-versa, permitindo assim a leitura e transmissão de dados ao utilizar o Raspberry Pi. E essa conversão é dada às conexões CS, SCK, SI e SO, descritos na Figura 11.

Essa transmissão de dados SPI é funcional graças ao controlador lógico presente no MCP2515, pois nele são conectados registradores e interruptores que são responsáveis pelo processo de conversão de dados CAN (MICROCHIP, 2018). Como foi visto na seção de composição de mensagem CAN, a natureza de sua composição é o alinhamento das ondas sequenciais de CAN *low* e CAN *high*, assim

para a leitura direta das mensagens CAN pelo Raspberry Pi, é necessário uma programação que alinhe as ondas sequenciais em seus tempos corretos, pois qualquer desalinhamento não se tem a formação da mensagem CAN. Logo, ao utilizar o MCP2515, não será necessária essa programação.

Figura 12 - MCP2515 Componentes.



Fonte: Microchip (2018).

Essa limitação do Raspberry Pi, explicado no Item 2.6 é devido as entradas em uma única pinagem, o que dificulta o processo de obtenção de dados da rede CAN já que a mensagem CAN possui duas componentes, a *high* e a *low*, para a formação de um bit.

A função do transceptor MCP2515 é a junção das ondas de CAN *high* e *low* a partir de seus interruptores e registradores, habilitando a leitura de dados via Raspberry Pi.

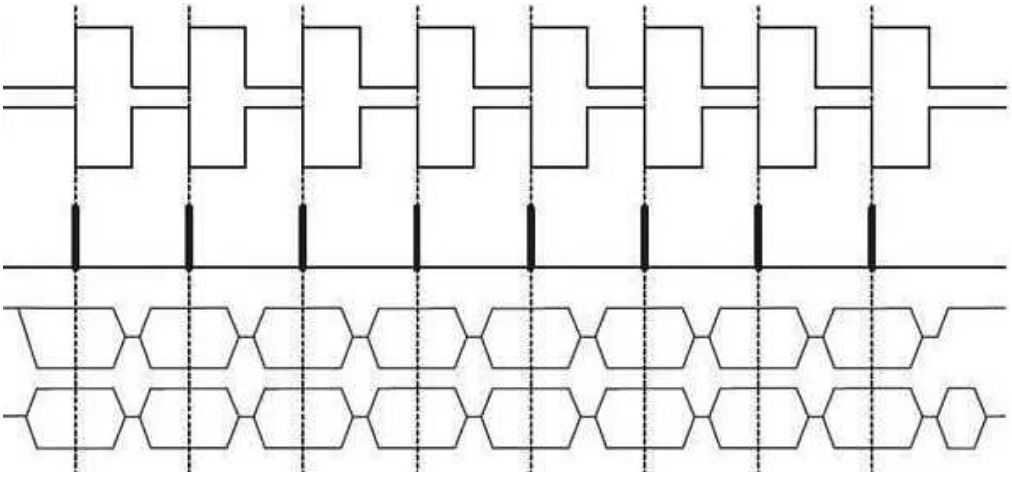
2.5 Protocolo SPI

O protocolo *Serial Peripheral Interface* (SPI) é o meio de comunicação do transceptor MCP2515 e este protocolo funciona de forma serial e síncrona, conhecida como Mestre-Escravo (DHAKER, 2023). Diferente da rede CAN, ou protocolo CAN, em que a comunicação é do tipo Mestre-Mestre, na qual os microcontroladores enviam e recebem dados simultaneamente ao barramento CAN.

Na comunicação SPI, o gerador de sinal Mestre é responsável pelo sincronismo da comunicação, enquanto a parte escrava recebe os sinais de comunicação externa obedecendo os padrões de sincronismo definidos pela parte Mestre (DHAKER, 2023). Os pinos de conexão SI e SO na Figura 11, os pinos 5 e 6 da descrição são os responsáveis pelo recebimento de dados Mestre-Escravo.

As ondas de forma quadrada no topo da Figura 13 são os dados recebidos dos microcontroladores da rede CAN. A partir do MCP2515 e do protocolo SPI, o gerador de frequência Mestre converte os dados CAN em SPI, que são as ondas hexagonais na parte inferior da Figura 13.

Figura 13 - Relação entre as ondas CAN e SPI.



Fonte: Dhaker (2023).

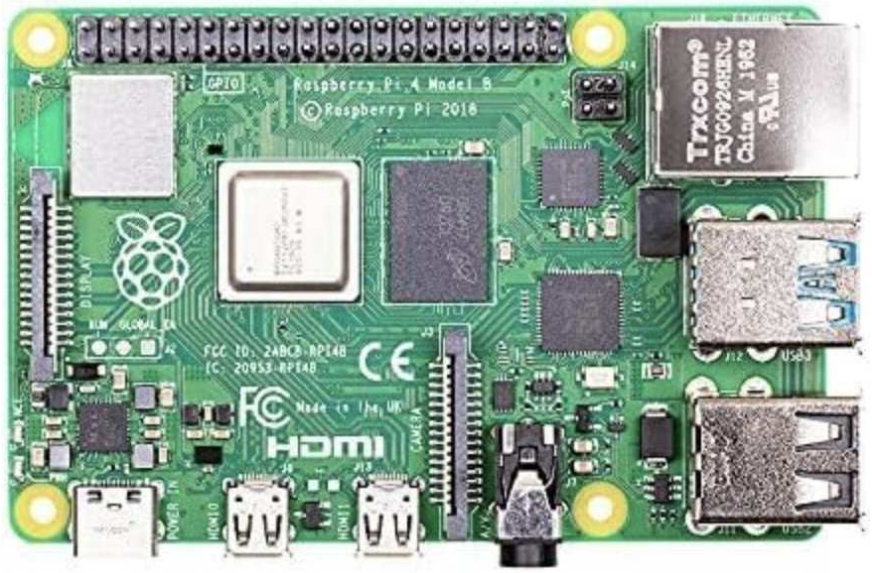
Essa conversão é possível a partir dos *timers* e *clocks* da própria arquitetura eletrônica do MCP2515. Pode-se verificar que no começo da conversão de dados, assim como o fim da conversão, as formas de onda SPI ou estão fora de sincronia, ou ainda não estão formadas. Isso é devido à própria natureza do protocolo, na qual são pelos pontos iniciais e finais pode-se encontrar o começo da sincronização dos dados série com os dados CAN.

A partir dos dados SPI é possível a obtenção dos binários representada na Figura 13 pelo gráfico de picos, localizado no meio da Figura. O Raspberry Pi lê esses binários para a formação das mensagens hexadecimais de rede CAN.

2.6 Raspberry Pi

O Raspberry Pi é um mini computador, de circuito integrado de placa única. Dentre seus componentes é ressaltado a presença de: processador, memória *Random Access Memory* (RAM), placa de vídeo integrada, entradas *Universal Serial Bus* (USB), *High-Definition Multimedia Interface* (HDMI) e as *General Purpose Input Output* (GPIO) (RASPBERRY, 2022). Na Figura 14, tem-se a ilustração do Raspberry Pi modelo 4b.

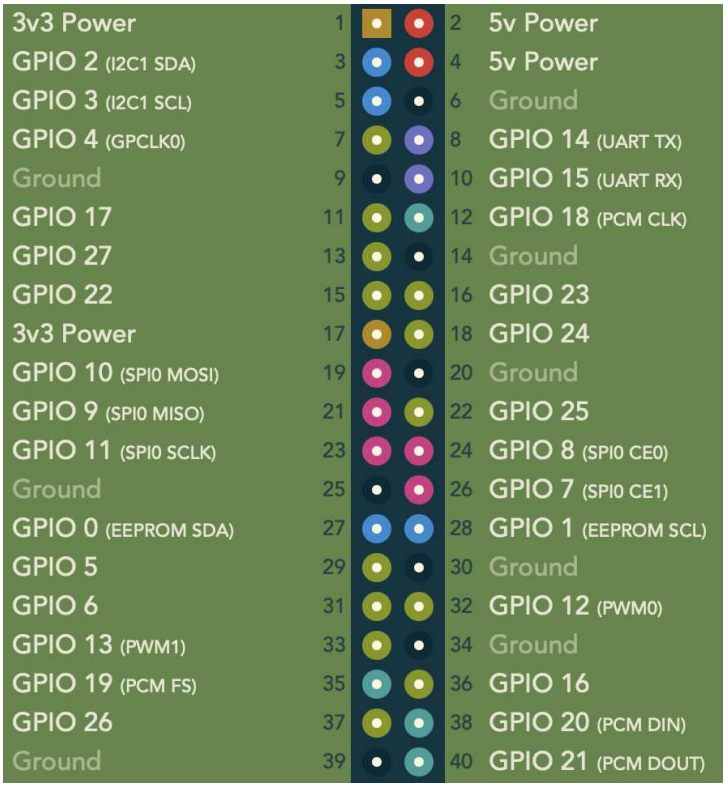
Figura 14 - Raspberry Pi model 4b.



Fonte: Raspberry (2023).

O Raspberry Pi possui o sistema operacional Raspbian, uma distribuição GNU/Linux de 32 bits Debian. Na Figura 15 é observada a pinagem presente no microcontrolador e como foi descrito no item 2.4, as suas GPIO's são dedicadas para uma entrada e uma saída por pino.

Figura 15 - Pinagem do Raspberry Pi model 4b.



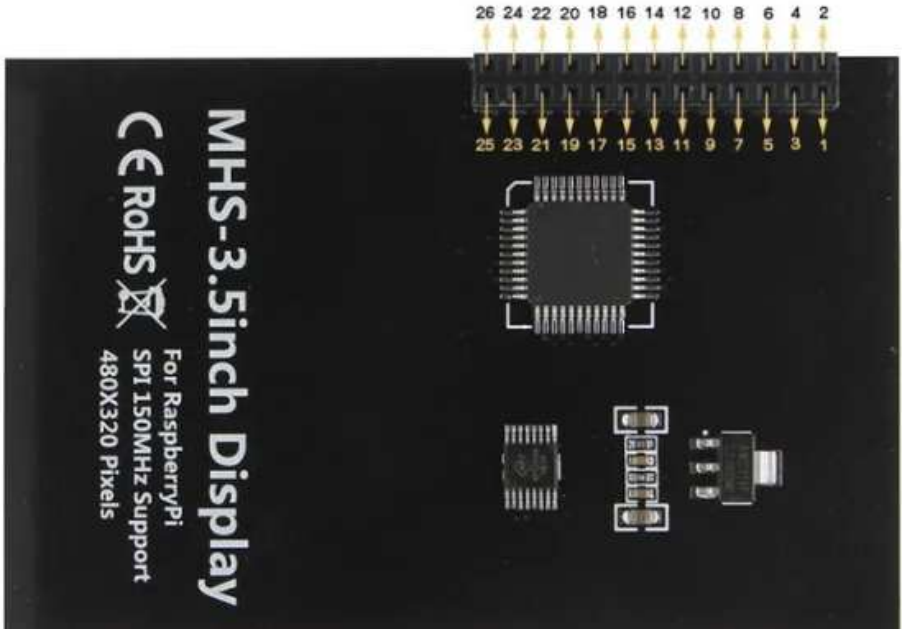
Fonte: Raspberry (2022).

Para a leitura de dados CAN, o Raspberry precisaria de dois pinos, um para receber os dados da CAN *high* e outro da CAN *low*, mas como o microcomputador não possui os sistema de Mestre-Escravo para a leitura de dados, a junção da frentes de onda *high* e *low* não é possível. Logo devido a natureza do próprio Raspberry Pi, foi utilizado o MCP2515 para a conversão de dados.

2.7 LCD

É uma tela de três polegadas e meia do tipo *touch screen* da Raspberry Pi. Ela é conectada aos GPIOs do Raspberry Pi. Na Figura 16 é mostrado o LCD assim como a pinagem. E a frente do LCD é mostrado na Figura 25.

Figura 16 - Tela LCD.



Fonte: Raspberry (2022).

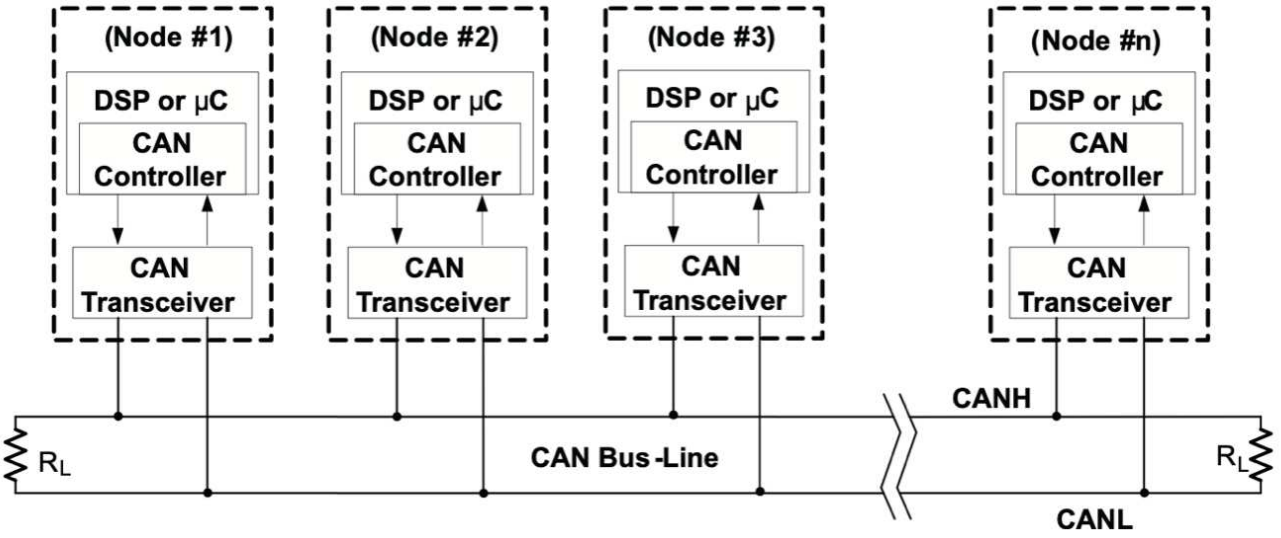
Os pinos 1 e 17 são as alimentações de 3,3 V, enquanto os pinos 2 e 4 são as alimentações de 5 V. Os pinos 6, 9, 14, 20 e 25 são os aterramentos, enquanto os pinos 3, 5, 7, 8, 10, 12, 13, 15 e 16 são pinos do tipo *Nothing Connected* (NC).

- Pinos 11, 21 e 26 são as saídas de dados do LCD;
- Pinos 18 e 19 são as entradas de dados do LCD;
- Pinos 23 e 24 são os interruptores de dados;
- Pino 22 é o *Reset*.

2.8 Circuitos Montado

Na Figura 17 tem-se a configuração esquemática das conexões existentes em uma rede CAN veicular, representado na figura como CAN *bus-line*, ou rede CAN. Alinhado com o que foi discutido no Item 2.3, é notada a associação que cada microcontrolador realiza com a rede através de suas ligações em paralelo, na figura os UCEs são representados por *Nodes*, ou seja, módulos. Cada módulo representa as unidades de controle eletrônicas veiculares.

Figura 17 - Rede CAN Veicular.



Fonte: Corrigan (2002).

Como é mostrado na Figura 17, cada módulo possui seu controlador CAN, representado por CAN *controller* e seu transceptor CAN, representado por CAN *transceiver*. Aqueles são responsáveis pela interpretação e processamento, e estes pela recepção de mensagens. Permitindo assim a transmissão e a recepção de informações enviadas na rede CAN.

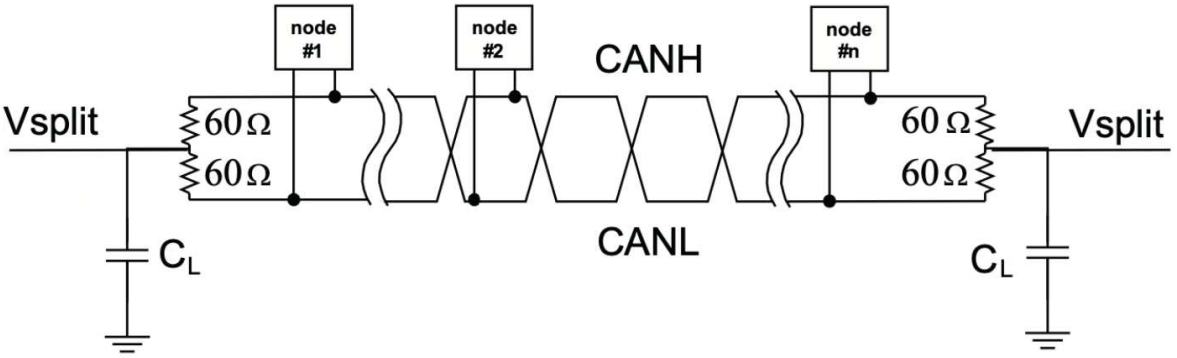
É importante notar a presença de dois resistores em cada um dos terminais da rede CAN, a presença destes resistores é para o funcionamento do protocolo CAN. A presença dos resistores, que geralmente variam de 120 Ω a 124 Ω dependendo das especificações e das capacidade de cada projeto, além de

estabilizar as variações de tensões *high* e *low* provenientes da rede CAN (CORRIGAN, 2002).

Já os capacitores funcionam como um filtro de ruídos. Já que a rede CAN é a sobreposição das ondas sequenciais para a formação das mensagens, a presença de ruídos impede o sequenciamento e alinhamento das formas de onda e por consequência, não há a formação das mensagens CAN (CORRIGAN, 2002).

É notado na Figura 18 que alguns projetos têm a presença de capacitores para se ter um filtro de ruídos.

Figura 18 - Rede CAN com Capacitores.



Fonte: Corrigan (Adaptado).

A presença de ruído não só interfere na integridade física das camadas do protocolo CAN, mas também impede a formação da própria mensagem CAN, tendo um atraso provocado nas frentes de onda de CAN *high* e *low*, atrasa e distorce as frequências de cada onda e até mesmo provoca um aumento nas amplitudes das tensões (MONROE, 2013).

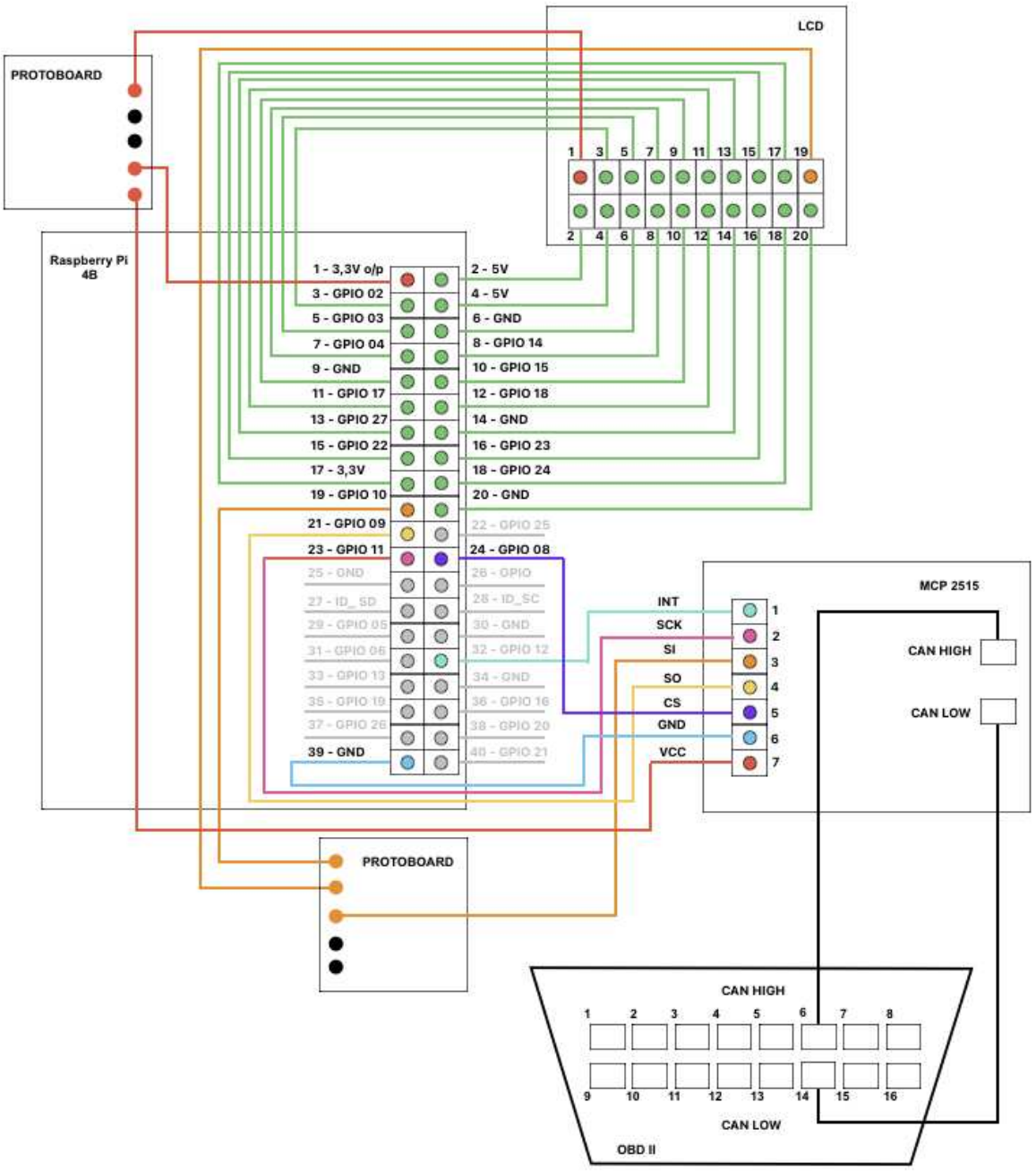
Para este projeto, foram seguidas as ligações mostradas na Figura 17 na realização da montagem do circuito de leitura e captura de dados da rede CAN utilizando o Raspberry Pi. O microcomputador tem a função do controlador CAN, responsável pelo processamento de dados de mensagens recebidas da rede CAN.

Já para o papel de transceptor CAN, foi dado ao MCP2515, responsável pela recepção de dados da rede CAN, assim transmitindo as mensagens para o Raspberry Pi. Como foi descrito no item 2.5, as mensagens CAN recebidas pelo

MCP2515 são convertidas para SPI, possibilitando o processamento de dados via Raspberry Pi.

Sendo assim, foi montado o circuito e feitas as ligações entre o Raspberry Pi, assim como o LCD e o MCP2515 conectado à rede CAN, conforme mostra a Figura 19.

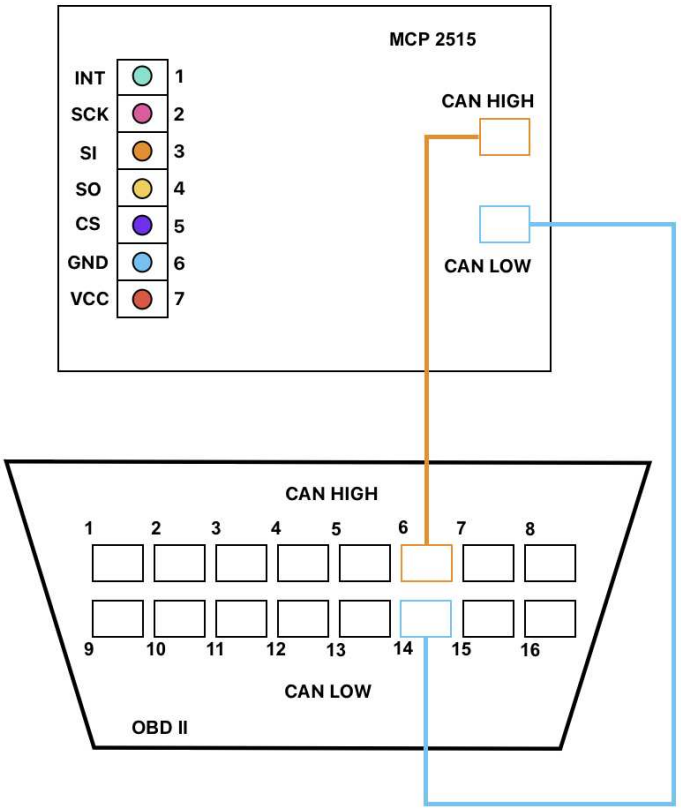
Figura 19 - Esquemático do *Datalogger* via Raspberry Pi.



Fonte: Autoria própria (2023).

É mostrado na Figura 20 as conexões presentes no transceptor CAN, função desempenhada pelo MCP2515. Ele é conectado à tomada OBD-II macho apenas com os pinos 6 e 14 do OBD, que são responsáveis pela captura de dados CAN *high* e *low* respectivamente. Nota-se que a tomada OBD-II possui um total de 16 pinos, como é mostrado na pinagem da Figura 10, mas para este projeto é apenas necessário a aquisição de dados CAN, logo não se faz o uso dos pinos restantes.

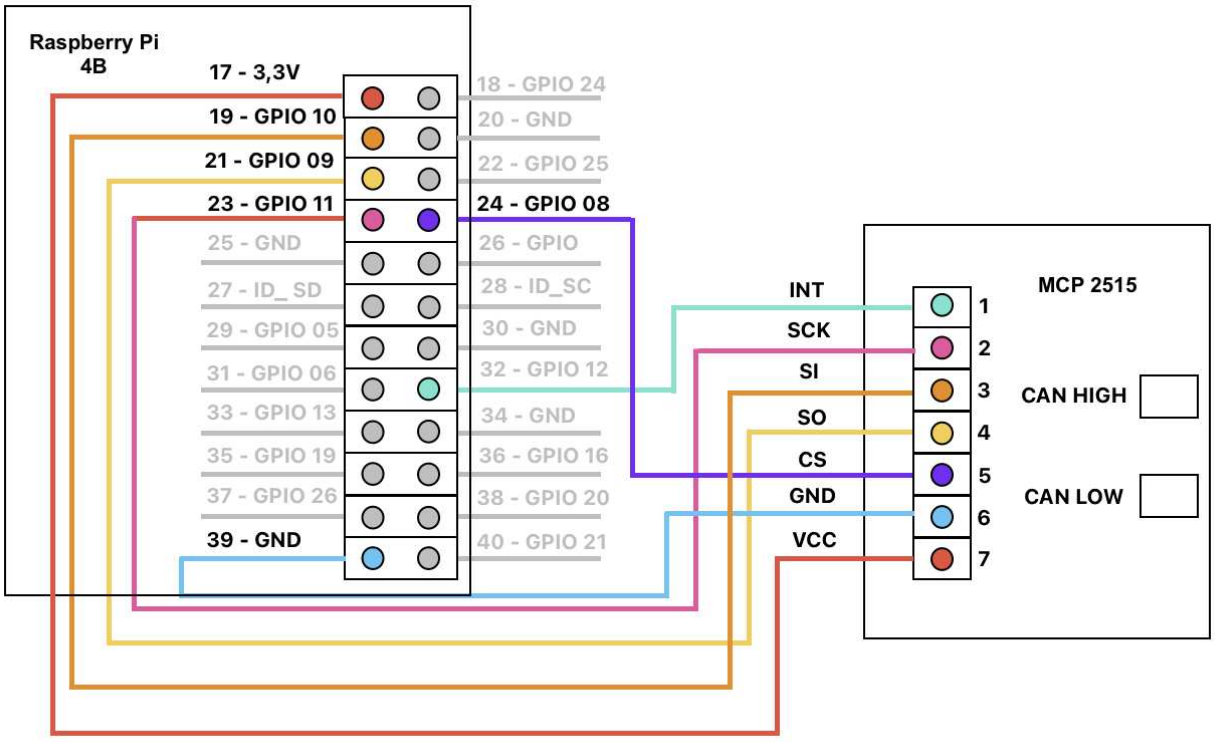
Figura 20 - Conexão entre o MCP2515 e o OBD-II.



Fonte: Autoria própria (2023).

No próprio MCP2515, tem-se a ligação de seus sete pinos de função ao Raspberry Pi, como mostrado na Figura 21. Como dito, o Raspberry Pi desempenha a função de um controlador CAN. É respeitada a pinagem das ligações do MCP2515 de acordo com o Item 2.4, na Figura 11 e a sua descrição.

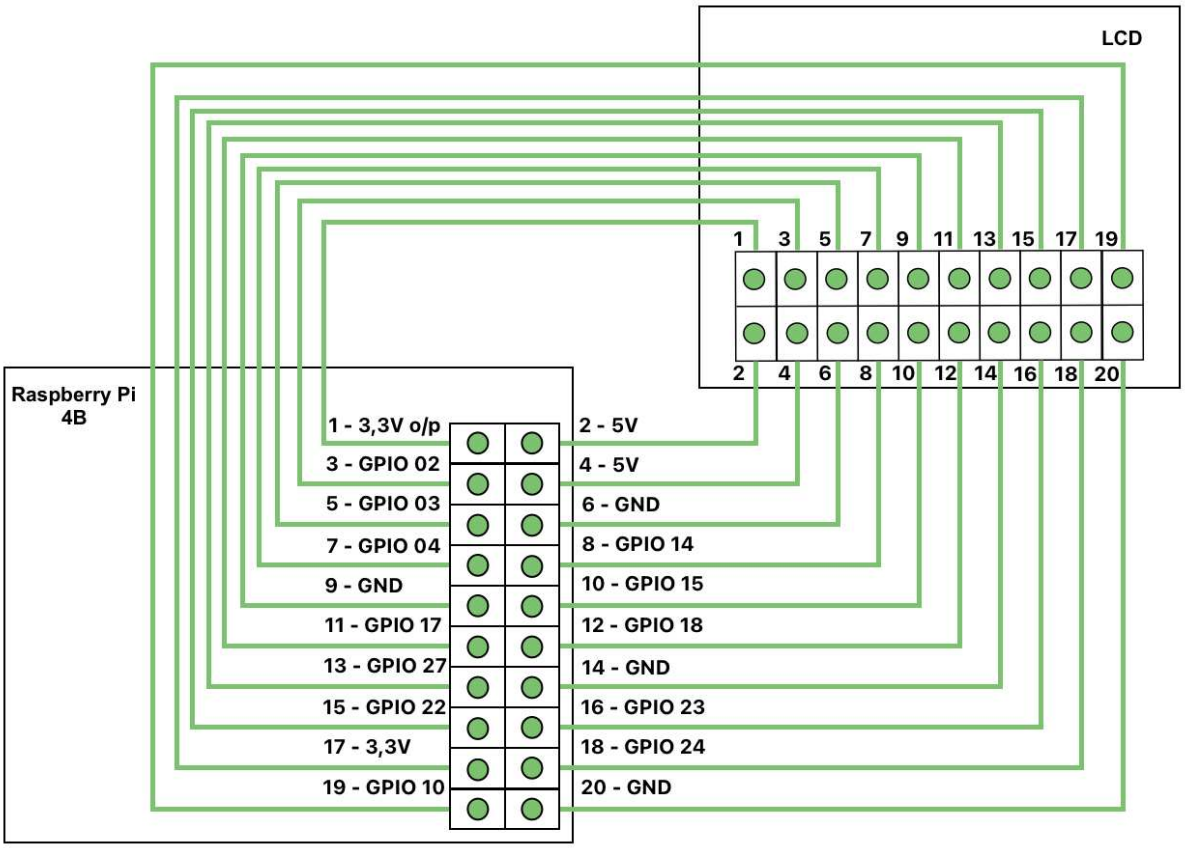
Figura 21 - Conexão entre o MCP2515 ao Raspberry Pi.



Fonte: Autoria própria (2023).

Para ter acesso aos dados obtidos da rede CAN por meio do circuito montado juntando o MCP2515 e o Raspberry Pi, este foi ligado a um LCD, representado pela Figura 22. Na qual essa conexão respeita a pinagem do LCD descrita no item 2.7, Figura 16 e a sua descrição.

Figura 22 - Conexão entre o LCD de 3,5 polegadas ao Raspberry Pi.



Fonte: Autoria própria (2023).

No Item 3.3 e seguintes é abordado o porquê das ligações da Figura 19 terem sido modificadas.

2.9 *Body Control Module* (BCM)

O módulo de controle da carroceria possui a maior quantidade de funções em um veículo, como o controle de faróis e setas (GUIMARÃES, 2007). É a UCE responsável pelas funções do habitáculo veicular, local onde é presente o motorista. A Figura 23 mostra a BCM utilizada para os testes.

Figura 23 - BCM de testes.

Fonte: Autoria própria (2023).

Foi escolhida a leitura na BCM devido a, justamente, a possibilidade de interagir com os dados de leitura. Com o acionamento e interação do freio veicular, mostrado no item 2.10, é possível visualizar a mudança nos valores hexadecimais dos dados da rede CAN, como é mostrado no Capítulo 4.

2.10 Interruptores

Os acionadores ou atuadores veiculares são dispositivos eletrônicos de interação direta entre o motorista e o automóvel (GUIMARÃES, 2007). A Figura 24 mostra o acionador do freio automotivo, dispositivo utilizado para a realização de leitura de dados.

Este acionador é responsável pelo controle eletrônico do freio veicular. Esta peça da Figura 24 é alinhada diretamente ao freio mecânico do automóvel. Enquanto o freio mecânico é responsável pela parada física do automóvel, o atuador envia dados do estado do freio mecânico para os outros dispositivos eletrônicos do veículo. Como ao acionar o freio, a luz traseira de posição é ligada.

Figura 24 - Acionador eletrônico do freio.

Fonte: Autoria própria (2023).

Diferente dos módulos eletrônicos UCEs, os atuadores não são ligados a rede CAN. Eles são ligados diretamente aos módulos eletrônicos que comandam as suas funções via protocolo LIN (ELETRONICS, 2022).

O protocolo LIN é semelhante ao CAN, porém possui o sistema de mensagens Mestre-Escravo, que diferente da rede CAN, tal que todos os UCEs mandam simultaneamente os dados para a rede, na rede LIN, o atuador só tem acesso a emissão de dados se o controlador em que ele está ligado permitir (ELETRONICS, 2022).

Neste projeto, o atuador do freio está ligado a BCM, logo ele só pode mandar a mensagem quando o módulo controlador da carroceria permitir.

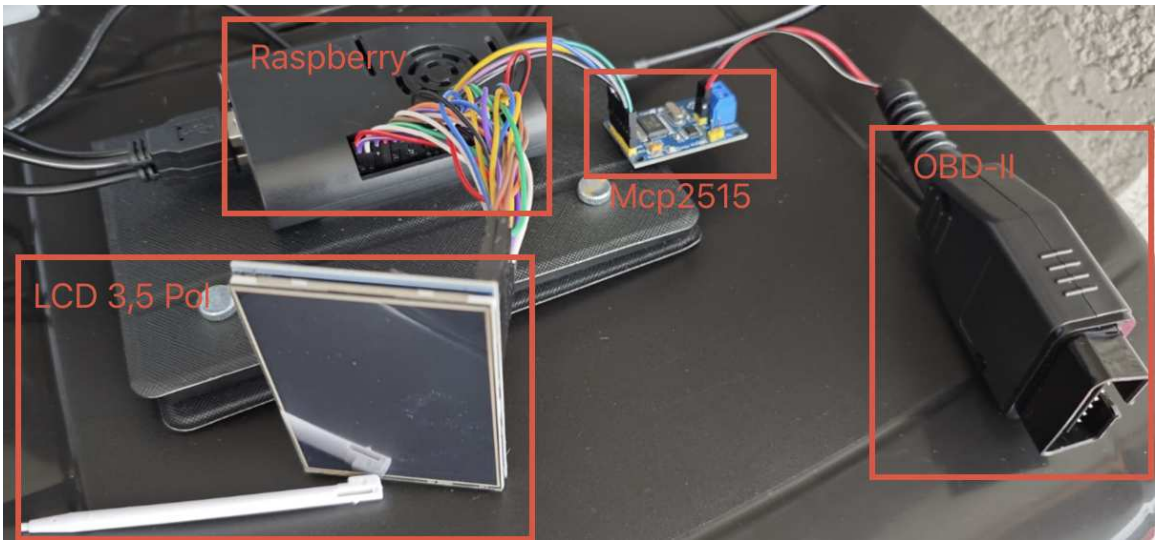
3 DESENVOLVIMENTO

Neste capítulo são mostradas as etapas do desenvolvimento do *datalogger* usando o Raspberry Pi, desde a montagem do circuito de coleta de dados da rede CAN veicular, assim como a bateria de teste realizada para a adaptação do circuito de leitura, baseado nas metodologias ágeis e de inovação apresentadas no Item 1.3.

3.1 Montagem do primeiro circuito

A partir do esquemático da Figura 19, foi montado o primeiro circuito do projeto, presentes o MCP2515, Raspberry Pi e o LCD de 3,5 polegadas, como é mostrado na Figura 25. É visto a ligação do MCP2515 e da tomada OBD-II macho, como foi apresentado na Figura 20, da Seção 2.8.

Figura 25 - Circuito *datalogger* com LCD de 3,5 polegadas.



Fonte: Autoria própria (2023).

Uma vez montado o circuito da leitura de dados da rede CAN, foram necessárias algumas configurações iniciais do Raspberry Pi para começar a programação da IHM para a obtenção de dados intersistêmicos veiculares. Logo o algoritmo da Figura 26, mostra os comandos iniciais para a atualização do sistema

Raspbian, permitindo a instalação do pacote de dados CAN, própria do sistema Debian.

Figura 26 - Configurações e instalações iniciais do Raspberry.

```
sudo apt-get udpade
sudo apt-get upgrade
sudo reboot
sudo apt-get install can-utils
```

Fonte: Autoria própria (2023).

Uma vez realizada a instalação inicial, foi necessário como passo seguinte a modificação de dados sistêmicos do Raspberry Pi para a habilitação da recepção de dados SPI, fornecidos via MCP2515. Logo, nos arquivos base do Raspberry Pi foi modificado de acordo com o algoritmo da Figura 27, na qual as quatro primeiras linhas de comando permitem a recepção e interpretação de dados SPI.

Figura 27 - Habilitação do canal SPI e CAN.

```
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=8000000,interrupt=12
dtoverlay=mcp2515-can1,oscillator=8000000,interrupt=12
dtoverlay=spi-overlay

dmesg | grep -i spi
dmesg | grep -i can
```

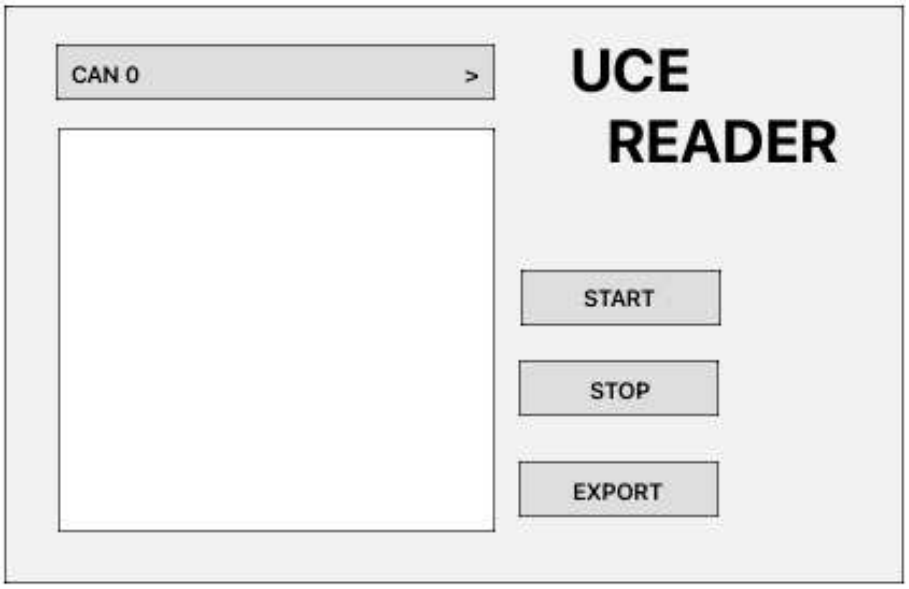
Fonte: Autoria própria (2023).

Note que na segunda e terceira linha de comando na Figura 27, os canais can0 e can1 são habilitados, justamente pelo fato de nos veículos automotivos possuírem dois canais CAN, permitindo a obtenção de dados de ambas ao mesmo tempo. Já nas últimas duas linhas de comando da figura são habilitados os GPIOs do Raspberry Pi para recepção de dados SPI e a conversão destes dados para CAN, pois como mostrado no Item 2.5, os dados binários SPI recebidos, são convertidos em hexadecimais CAN.

3.2 Primeira IHM do *Datalogger*

Com a utilização do programa Visual Studio Code, e da linguagem de programação Python, foi programada a primeira Interface de leitura de dados do *datalogger*, utilizando a biblioteca PyQt5 do Python, assim como o Qt Designer, um *framework* de gráficos e imagens. Na Figura 28 é mostrado a tela principal da interface.

Figura 28 - Interface IHM para LCD de 3,5 polegadas.



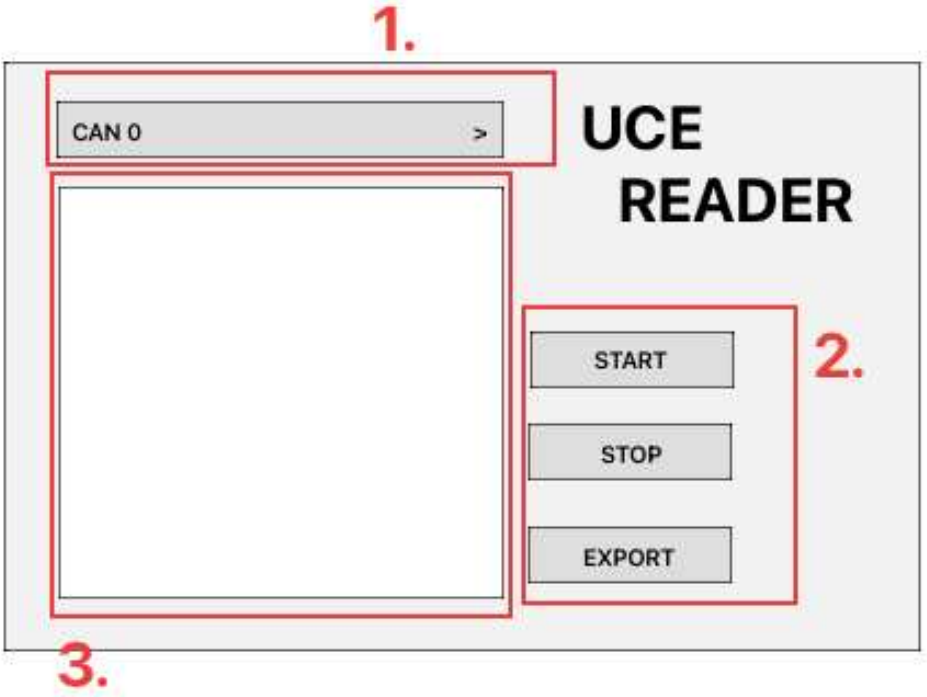
Fonte: Autoria própria (2023).

Na Figura 29 são mostradas as funcionalidades da interface programada.

- Sendo o quadro em vermelho de número 1, são mostradas a tela de função que permite a seleção de qual rede CAN a leitura de dados é realizada, uma vez que os veículos possuem até duas redes CAN;
- Já no quadro em vermelho de número dois, é mostrado os botões de funções da própria interface, sendo o botão de *Start* o responsável pela inicialização da leitura e obtenção de dados da rede CAN. O botão *Stop* é o responsável pela finalização da leitura de dados. E uma vez com os dados obtidos no quadro em vermelho de número 3, o botão *Export* é o responsável pela

exportação de dados obtidos ao um arquivo de formato de texto, salvo no cartão de memória SD do Raspberry Pi.

Figura 29 - Funções da Interface IHM para LCD de 3,5 polegadas.



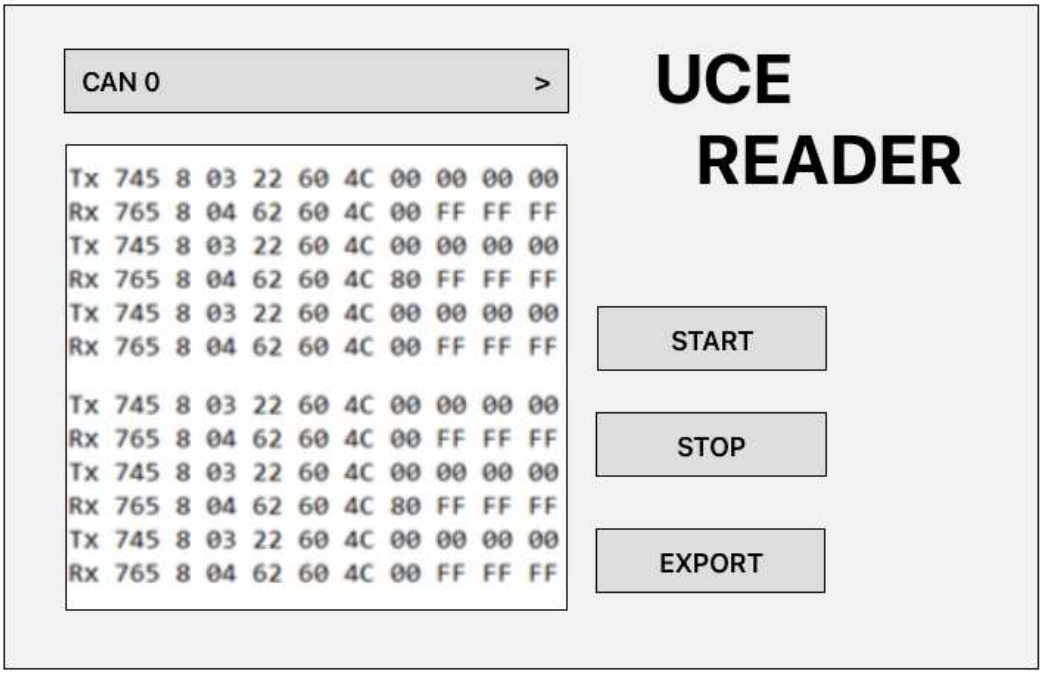
Fonte: Autoria própria (2023).

Com a funcionalidade da interface gráfica, é possível a visualização da leitura de dados da rede CAN.

3.3 Primeira leitura do *Datalogger* e as limitações

Com o circuito montado e a interface de leitura em disposição, na Figura 30 é mostrada a primeira realização da leitura de dados da rede CAN, obtidos através da plataforma de integração eletroeletrônica, apresentada na Figura 5.

Figura 30 - Primeira leitura da rede CAN do *datalogger*.



Fonte: Autoria própria (2023).

Uma vez obtidos os dados da rede CAN, é possível exportar os dados em um arquivo de formato texto para análise. Como é mostrado na Figura 31.

Figura 31 - Dados em formato texto para análise.

```
09:15:07 Tx 745 8 02 10 C0 00 00 00 00 00
09:15:07 Rx 765 8 02 50 C0 FF FF FF FF FF
09:15:16 Tx 745 8 02 10 C0 00 00 00 00 00
09:15:16 Rx 765 8 02 50 C0 FF FF FF FF FF
09:15:22 Tx 745 8 02 10 C0 00 00 00 00 00
09:15:22 Rx 765 8 02 50 C0 FF FF FF FF FF
09:16:05 Tx 745 8 02 3E 01 00 00 00 00 00
09:16:05 Rx 765 8 01 7E FF FF FF FF FF FF
09:16:26 Tx 745 8 02 3E 01 00 00 00 00 00
09:16:26 Rx 765 8 01 7E FF FF FF FF FF FF
09:16:46 Tx 745 8 02 3E 01 00 00 00 00 00
09:16:46 Rx 765 8 01 7E FF FF FF FF FF FF
```

Fonte: Autoria própria (2023).

Baseada na funcionalidade de um *datalogger* presente no mercado, foi desenvolvido o primeiro circuito deste projeto. Em sequência foi realizado o teste do protótipo, na qual foram obtidos os dados da rede CAN.

Mas durante a etapa da leitura de dados do protótipo, foi visto as limitações desta versão do projeto. Os pilotos de teste estavam tendo retrabalho após cada bateria de testes, já que os resultados são lidos somente após a finalização de cada teste, e se os resultados não atingirem o esperado, cada piloto teria de remontar o circuito de teste e seriam realizadas as remontagens e releituras até atingir o objetivo esperado.

Portanto, a possibilidade de analisarem os resultados no momento em que a leitura de dados está sendo realizada. Oferecendo uma vantagem em estudar os resultados obtidos e mudar as configurações de circuito veiculares à medida que os resultados dos testes são obtidos.

Tal que não seja necessário a análise posterior, resultando em os pilotos de teste não precisarem refazer a bancada de testes toda vez que o resultados obtidos não atingirem respostas esperadas, economizando tempo.

Logo uma nova configuração é obtida. Pois as limitações após os resultados obtidos podem ser corrigido, como:

- Limitações no tamanho da tela de LCD de 3,5 polegadas. A leitura no tamanho limitado desta tela dificulta a interpretação dos dados obtidos pelos pilotos de teste;
- Limitações na interface gráfica. A IHM projetada para a tela de 3,5 polegadas é apenas um intermédio para iniciar e parar a leitura de dados intersistemas CAN através de seus botões, vistos na Figura 29. Após a obtenção de dados, o piloto deve retirar o arquivo de texto com as informações dos testes e analisá-lo em seu computador posteriormente.

Na Figura 32 é mostrado uma nova idealização do processo, desde uma nova forma de visualização de dados e uma nova forma de interação com os dados.

Figura 32 - Etapas na resolução dos problemas presentes.

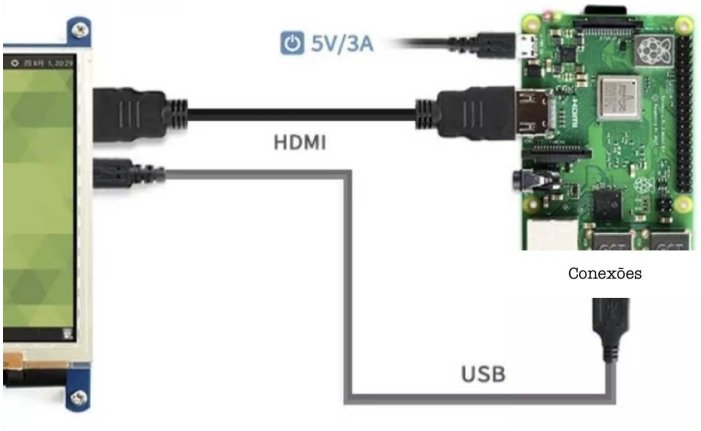
Fonte: Autoria própria (2023).

Uma vez definidas as soluções para as limitações, foi a inicializada a nova montagem do circuito.

3.4 Circuito com o LCD de 7 polegadas

Como está representado na Figura 33, a conexão entre o LCD de 7 polegadas e o Raspberry Pi, é via *High Definition Multimedia Interface* (HDMI), que é diferente da ligação entre o LCD de 3,5 polegadas da Figura 22 .

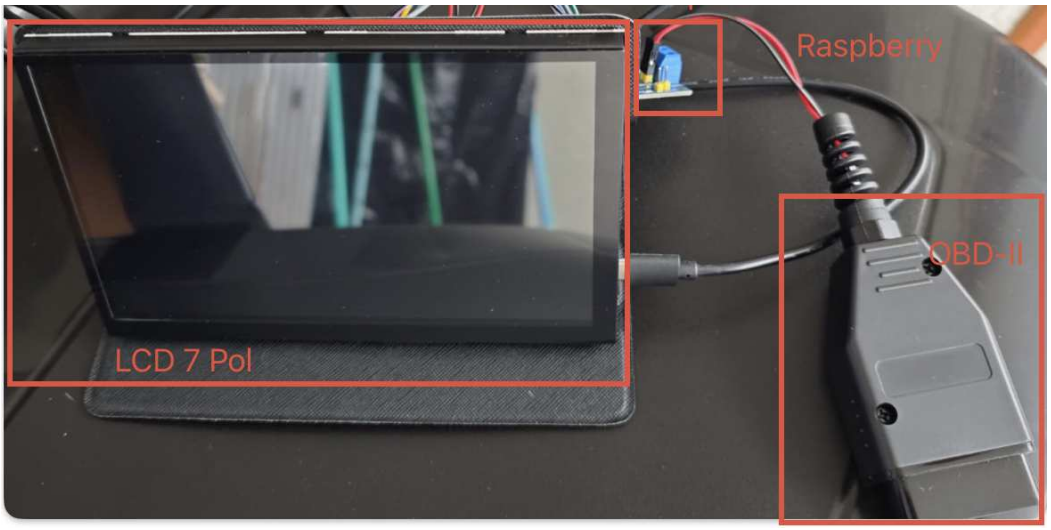
Figura 33 - Ligação entre o Raspberry Pi e o LCD 7 de polegadas.



Fonte: Raspberry (2023).

Na Figura 34 é mostrada a montagem real desta ligação. A partir desta nova configuração entre o LCD de 7 polegadas e o Raspberry Pi, os pinos 20 GPIOs antes necessários com a tela de 3,5 polegadas, mostrada da Figura 22, estão livres.

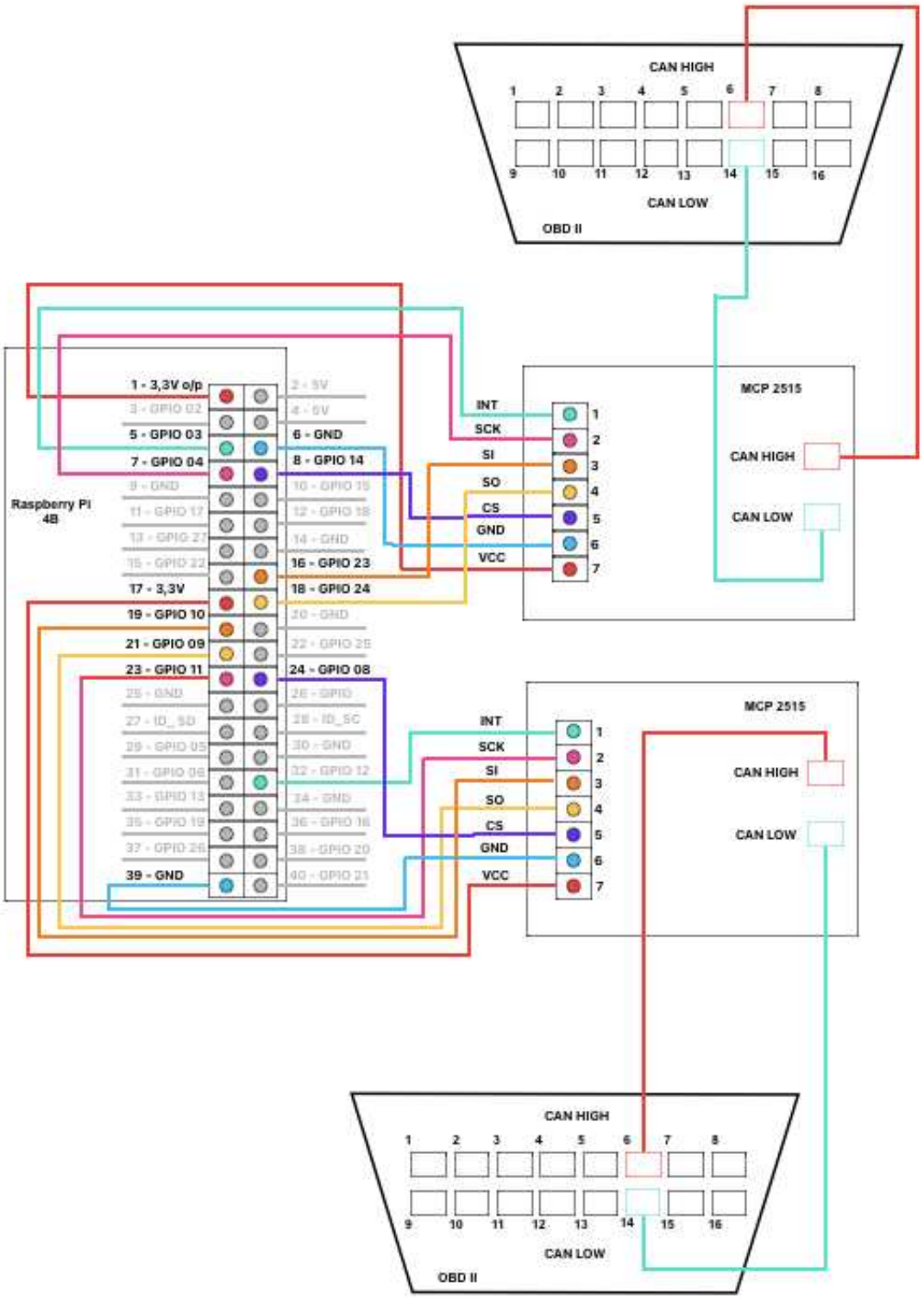
Figura 34 - Ligação física entre o Raspberry Pi e o LCD 7 polegadas.



Fonte: Autoria própria (2023).

Possibilitando assim uma nova disposição para estes pinos. Como está no esquemático da Figura 35, foi acoplado ao Raspberry Pi mais um MCP2515, tendo assim uma nova funcionalidade para a ferramenta de aquisição de dados entre sistemas veiculares.

Figura 35 - Esquemático do *datalogger* com dois MCP2515.



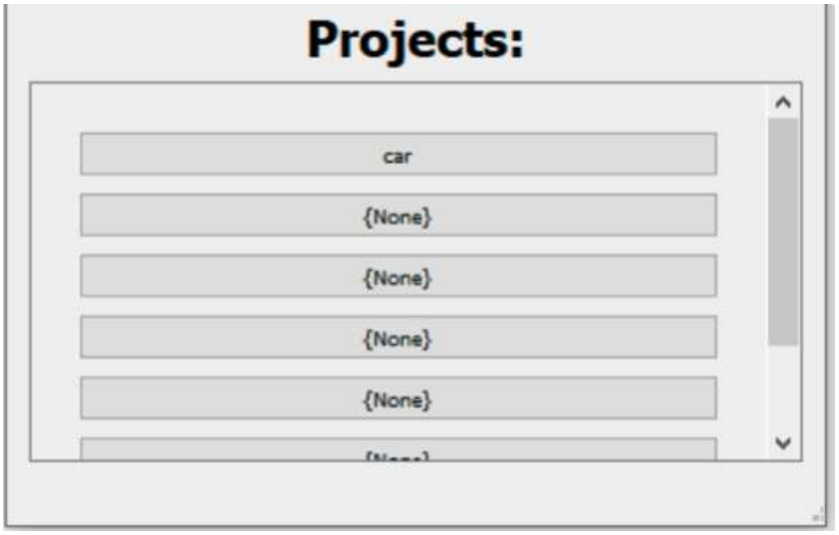
Fonte: Autoria própria (2023).

É possível nesta disposição apresentada na Figura 35, com a presença de dois MCP2515, a aquisição de dados em dois veículos ao mesmo tempo. Permitindo uma comparação de dados em tempo real de dois possíveis testes.

3.5 Melhoria na interface de aquisição de dados do *datalogger*

Baseada nas melhorias apontadas, foi desenvolvida uma nova interface de leitura de dados. Foram utilizadas as mesmas ferramentas de programação da interface anterior. Ao executar o programa, na Figura 35 é mostrada a tela inicial.

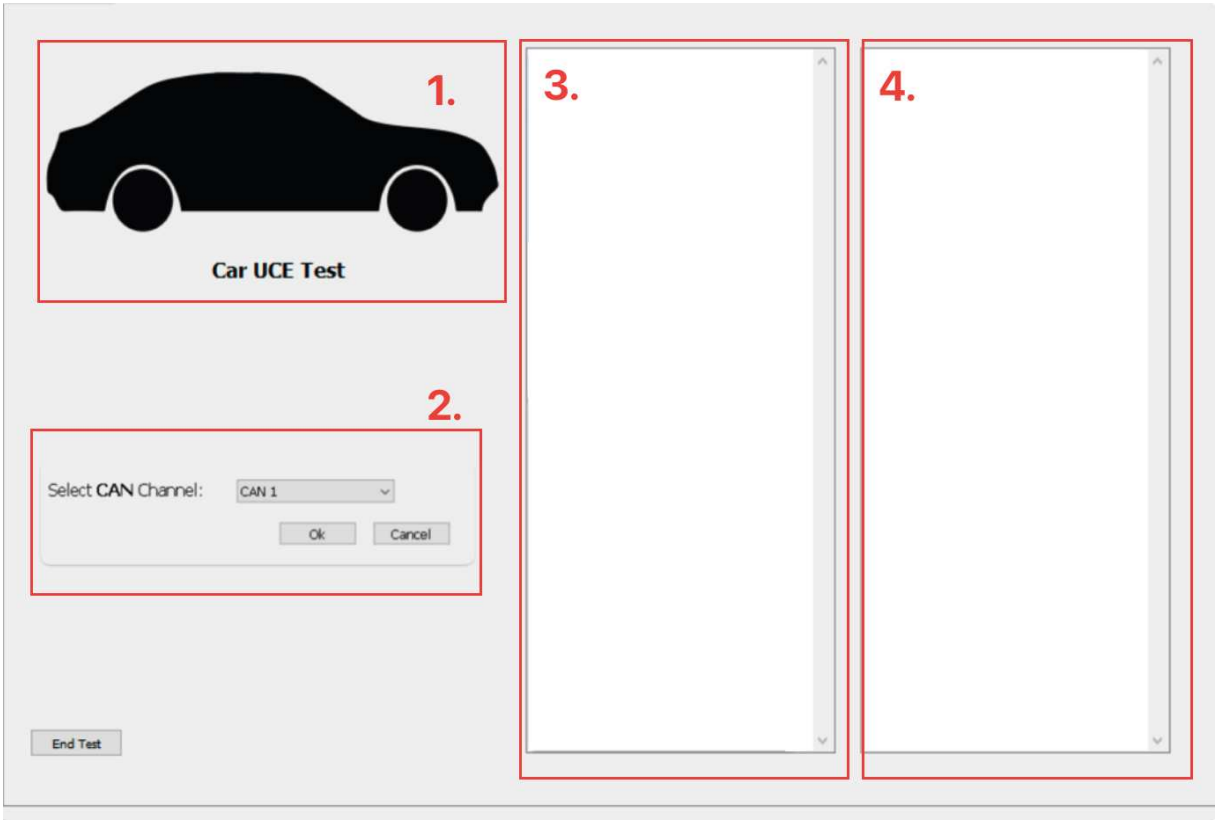
Figura 36 - Tela inicial da interface.



Fonte: Autoria própria (2023).

Nesta tela inicial é possível selecionar o projeto em que estão sendo realizados os testes de leitura, cada uma das caixas de seleção pode conter as opções de modelos de carro de acordo com o projeto. Após a tela inicial, é mostrado na Figura 37 a tela principal do programa. No quadro vermelho de número 1, a interface mostra uma imagem do modelo do carro ou projeto, selecionado na tela inicial, sendo assim um meio visual para perceber em qual projeto está.

Figura 37 - Tela principal da interface.



Fonte: Autoria própria (2023).

No quadro vermelho de número 2 da Figura 37, é possível a seleção de qual rede CAN é realizada a leitura de dados, pois como é conectado dois MCP2515, é disponível a leitura de até 4 redes CAN, já que cada veículo possui até duas redes CAN, sendo um MCP2515 por veículo.

Os quadros vermelhos de números 3 e 4 da Figura 37 é mostrado a parte da interface onde é disponível ao piloto de teste a leitura de dados em tempo real da rede CAN, como é disponibilizado dois MCP2515, a interface do quadro vermelho 3 representa a leitura da rede can0 e can1 de um automóvel, enquanto a interface do quadro vermelho 4 representa a leitura da rede can0 e can1 do outro veículo.

3.6 Leitura de dados do *datalogger* nesta nova IHM

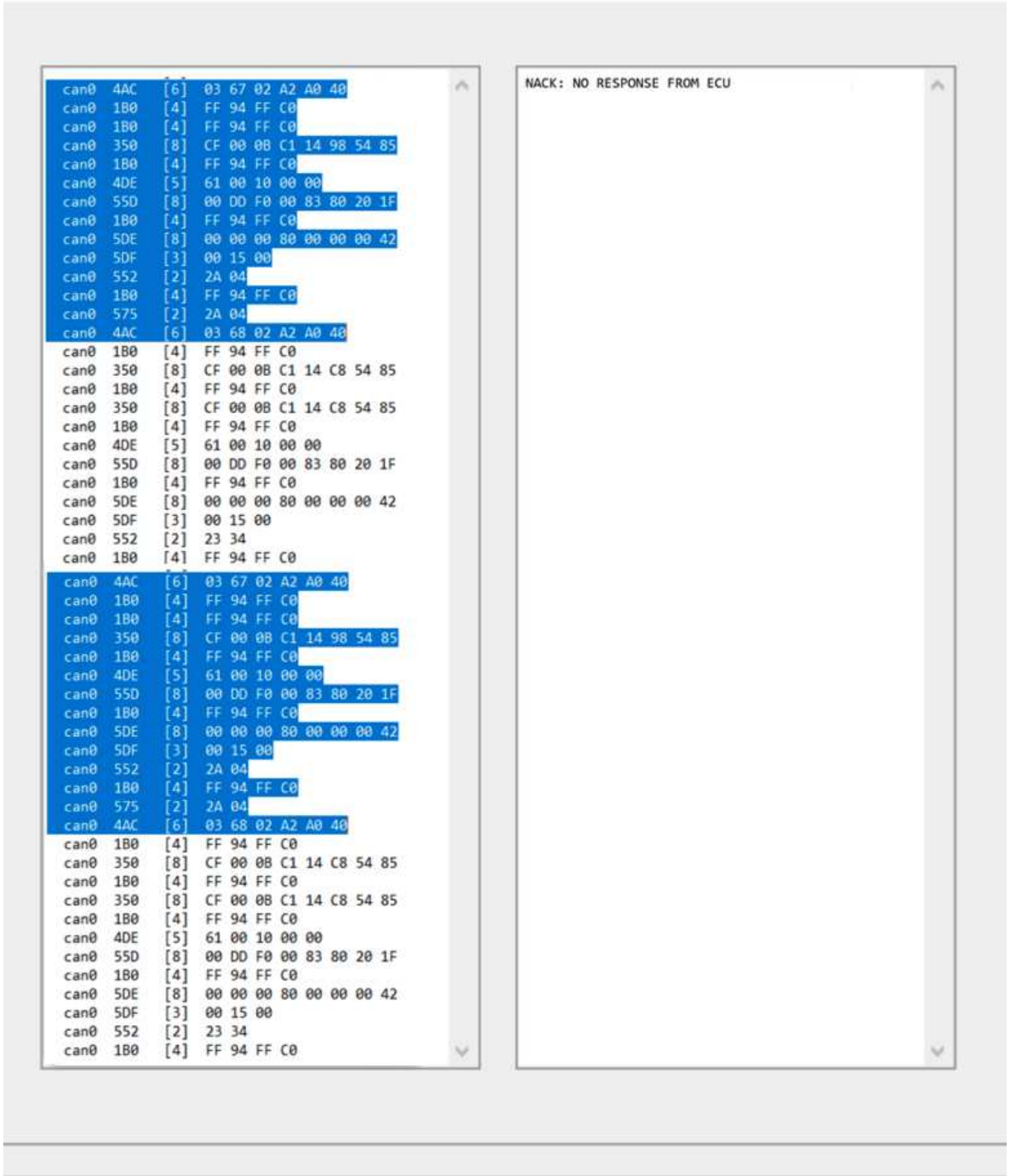
Uma vez presente a nova IHM, na Figura 38 é mostrada a realização da leitura de dados. Nesta nova bateria de testes, a leitura dos dados intersistêmicos é realizada apenas na UCE da *Body Control Module* (BCM). Esta escolha foi justificada no Item 2.9.

Figura 38 - IHM do *datalogger.*

Fonte: Autoria própria (2023).

Na Figura 39 são mostrados os resultados da leitura obtida de duas BCMs, na qual os resultados são abordados no Capítulo 4. Foi lido uma bateria de teste em duas BCMs enquanto era acionado o freio veicular. Os dados da coluna à esquerda estão propositalmente destacados em azul para a análise de resultados.

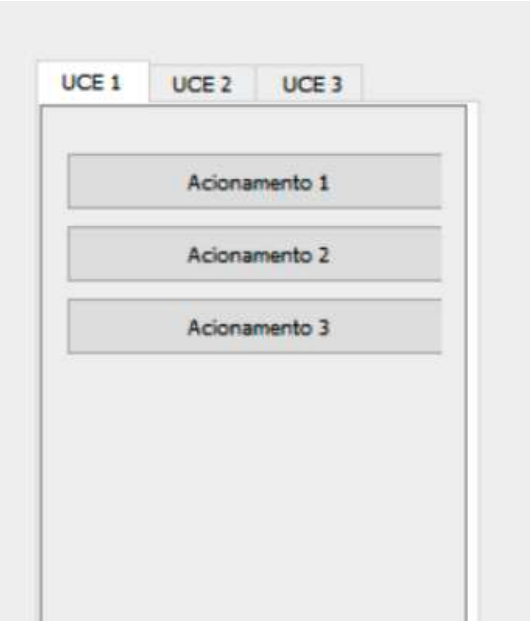
Figura 39 - Tela de leitura de dados.



Fonte: Autoria própria (2023).

Como é mostrado na Figura 40, é notado uma nova funcionalidade da interface, com o intuito de facilitar a visualização de dados aos pilotos de testes, é possível a identificação de quais módulos, ou UCEs, estão sendo realizados as leituras, assim como a identificação de quais acionadores estão presentes.

Figura 40 - Tela de identificação dos módulos e acionadores.



Fonte: Autoria própria (2023).

Com a implementação da nova interface, os pilotos de testes podem realizar a leitura e interpretação dos dados obtidos. Sem a necessidade de transpor os dados para máquinas de trabalhos diferentes para realizar esta leitura.

A interface gráfica permite ao piloto um discernimento de qual projeto está sendo realizado a leitura de dados sistêmicos com o intermédio da tela inicial, apresentado na Figura 36, assim como no quadro vermelho 1, da Figura 37.

É possível o discernimento de quais módulos e acionadores estão presentes no momento da leitura de dados.

Com a presença de dois MCP2515, é possível a leitura em paralelo de dois veículos de teste ao mesmo tempo, comparado em tempo real, dados para um possível estudo.

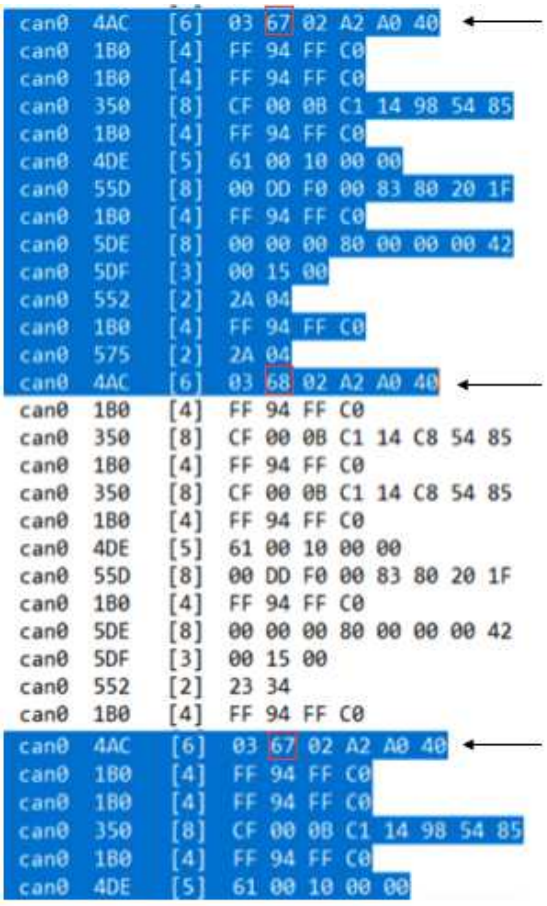
4 RESULTADOS

Neste capítulo é apresentada a análise de resultados obtidos no Capítulo 3. Assim como é apresentada a função do teste aplicado da BCM e do acionador.

4.1 Resultados obtidos nos testes

No Item 3.6, Figura 39, são mostrados os resultados da leitura de dados entre as BCMs e os atuadores de freio. Os resultados são replicados na Figura 41 e 42, sendo eles respectivamente, os dados entre o primeiro MCP2515 com uma BCM e atuador funcional, seguida do segundo MCP2515 ligado a uma segunda BCM não funcional, em que o módulo apresentava falhas em suas componentes internas.

Figura 41 - Resultados da BCM e atuador funcional.



Fonte: Autoria própria (2023).

Figura 42 - Resultados da BCM não funcional.

Fonte: Autoria própria (2023).

Na figura 42 apresenta a mensagem de sem respostas da UCE, pois ela possui falhas internas. É mostrada na Figura 41, indicada por flechas, as mensagens do módulo da BCM e destacadas em vermelho, os bits da mensagem a serem analisados. Neste teste as mensagens analisadas são:

can0	4AC	[6]	03	67	02	A2	A0	40
can0	4AC	[6]	03	68	02	A2	A0	40
can0	4AC	[6]	03	67	02	A2	A0	40

O primeiro dado “can0” é em função da programação IHM deste projeto, e indica em qual rede CAN a mensagem está sendo lida. Já a segunda informação é o identificador da mensagem, ou seja, o ID do microcontrolador e neste caso a BCM possui o identificador “4AC”. A informação “[6]” representa em qual posição o UCE BCM está localizado sendo uma informação do próprio módulo.

A partir do bit “03” até o “40”, é a informação transmitida do BCM até a rede CAN. Como é mostrado na Figura 41, nota-se que o único bit mudado em toda a leitura de dados é o de “67” para o “68” do calculador da BCM, isso é devido ao acionamento do atuador do freio da Figura 24.

Ao separar o hexadecimal 67 em binário, tem-se:

6 - 0 1 1 0 | 7 - 0 1 1 1

E ao separar o hexadecimal 68 em binário, tem-se:

6 - 0 1 1 0 | 8 - 1 0 0 0

Logo, o binário 0111, que em hexadecimal é 7 representa o freio não acionado, já ao pressionar o freio do automóvel, tem-se a mudança do bit 7 para 8, que em binário é 1000. Nota-se que após o estado do freio ser pressionado, ele volta ao estado não pressionado, pois os bits 68 mudam de estado para 67. A razão da rápida mudança de estado é explicada no Item 4.2, que explica a natureza deste teste entre a BCM e o atuador.

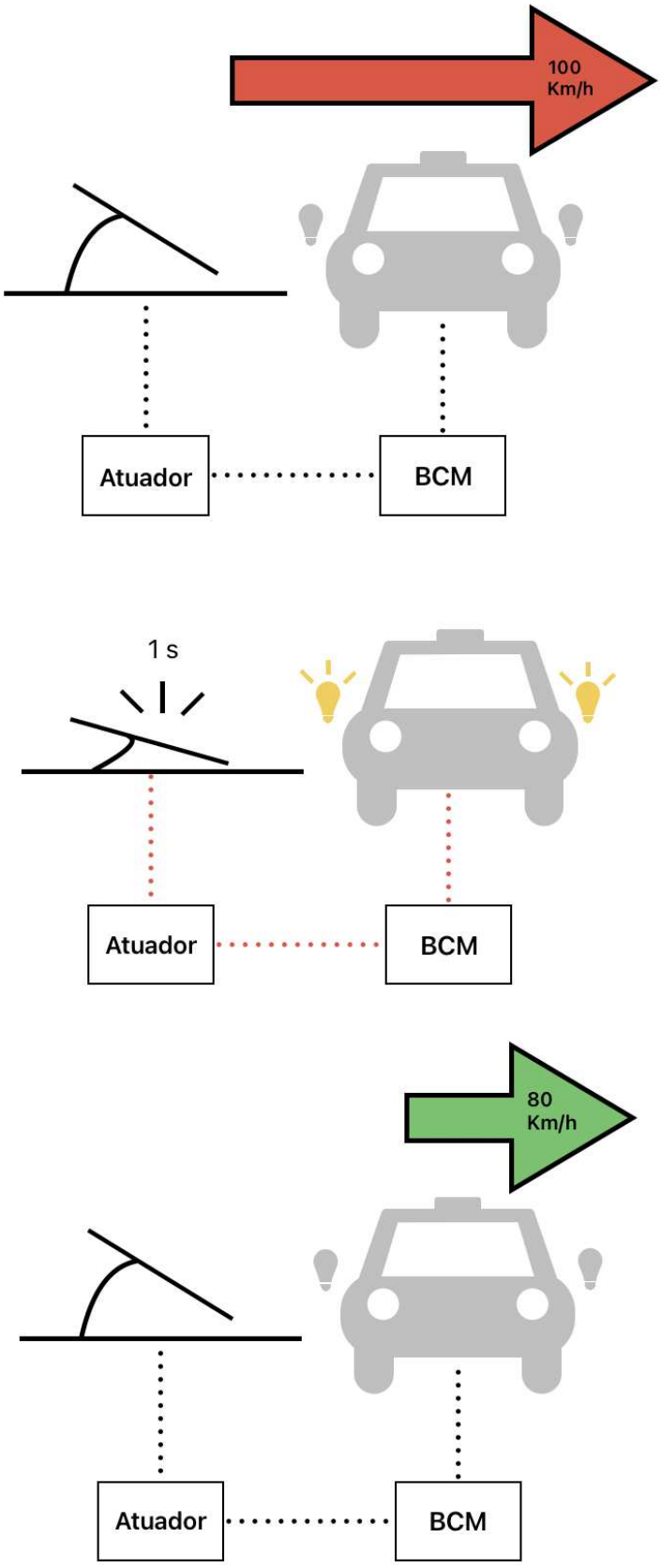
Na Figura 41 tem-se a mensagem traduzida para a resposta da UCE, indicando falha na leitura de dados da rede CAN. Este resultado era esperado, pois a segunda BCM para a realização deste teste estava conectada a um chicote veicular que apresentava defeitos em suas conexões, logo a mensagem de conformidade.

4.2 Teste de validação das luzes de alerta

O teste com os resultados anteriores foi para validar na bancada da PIE a situação presente da Figura 43. A bancada está simulando um automóvel viajando a uma velocidade de 100 km/h, nesta velocidade o pedal do freio foi acionado em um breve instante de 1 segundo, tal que o veículo desacelera para 80 km/h.

Neste intervalo de tempo em que o carro estava desacelerando, as luzes traseiras de alerta do veículo devem piscar. Esta situação simula um caso real, tal que um automóvel viajando a alta velocidade, sofrendo uma freada repentina, tem seus pisca de alerta traseiros acionados, possibilitando um controle visual para o motorista que está dirigindo atrás do veículo simulado. Isso permite ao veículo traseiro ter alguma resposta mais intuitiva na estrada.

Figura 43 - Teste do sinal de alerta traseiro.



Fonte: Autoria própria (2023).

Alinhado o teste da Figura 43 com os resultados apresentados da Figura 39, a mensagem do pedal do freio que foi acionado e desacionado em um curto intervalo de tempo, teve o resultado esperado. Apresentados no Item 4.1 é visto a mudança dos bits 67 para 68 e voltando para 67 na mesma proporção de tempo mostrado na Figura 43.

Porém, durante os testes, os piscas de alerta traseiros não acionaram, ou não tiveram um contato visual de ser acionado. Mas com os dados obtidos, pode ser concluído que:

1. Não apresentou falhas como o apresentado na Figura 42, logo não tem uma falha física durante o teste, eliminado a possibilidade de falhas no chicote veicular, uma vez que os faroís estarem acesos;
2. Como teve respostas da rede CAN, tal como, resposta de comportamento esperado para este teste, apresentado na Item 4.1, a falha não poderia ser na BCM, nem no atuador, pois foi obtida a resposta esperada de ambas.

Logo, o modo de falha foi encontrado no próprio pisca alerta traseiro. Não houve um defeito nas luzes, mas a lógica programada nesses faróis traseiros estavam em uma determinada frequência e como o período da ação do freio durante o teste foi muito curto, a frequência ainda não tinha sido ajustada para piscar neste período desejado.

Com o ajuste na frequência de acordo com o novo período de oscilação dos piscas de alerta traseiro, ao refazer o teste apresentado na Figura 43, realmente se obteve êxito na visualização das luzes de alerta do veículo teste, assim como se obteve êxito ao validar os resultados obtidos da rede CAN.

Para o teste de confiabilidade do circuito, foram realizadas sequência de 10 leituras com as seguintes condições:

- 10 leituras com um módulo conectado;
- 10 leituras com três módulos conectados;
- 10 leituras com cinco módulos conectados;
- 10 leituras com sete módulos conectado;

Foi escolhido o máximo de sete módulos, pois é a quantidade de módulos que um carro popular de 2024 normalmente possui. A tabela da Figura 44 mostra os resultados.

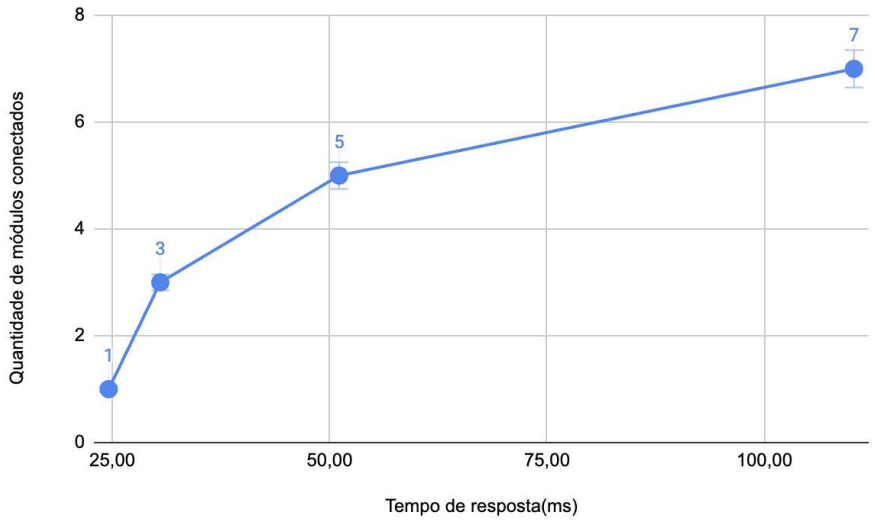
Figura 44 - Tabela do tempo de resposta pela quantidade de módulo.

Tempo de resposta(ms)	Quantidade de módulos conectados	Tempo de resposta(ms)	Quantidade de módulos conectados	Tempo de resposta(ms)	Quantidade de módulos conectados	Tempo de resposta(ms)	Quantidade de módulos conectados
24,01	1	30,03	3	50,54	5	110,01	7
25,45		31,25		51,89		111,01	
24,02		30,05		50,57		110,01	
24		30,03		50,54		110,02	
25,43		31,29		51,93		111,01	
25,46		31,24		51,91		109,99	
24,01		30,02		50,54		110,01	
25,44		31,25		51,89		111,01	
24		30,01		50,55		110,02	
24,02		30,04		50,54		110,01	
Média(ms)	24,58	**Média(ms)**	30,52	**Média(ms)**	51,09	**Média(ms)**	110,31

Fonte: Autoria própria (2024).

O gráfico da Figura 45 mostra esse aumento no tempo de resposta da mensagem com o aumento da quantidade de módulos conectados.

Figura 45 - Gráfico do tempo de resposta versus a quantidade de módulo.



Fonte: Autoria própria (2024).

Logo, apesar do tempo de resposta aumentar com a presença de mais módulos conectados à rede CAN, as mensagens recebidas são as mesmas obtidas no teste de validação.

5 CONCLUSÃO

Neste trabalho foi atingido o objetivo em desenvolver uma ferramenta de leitura de dados da rede CAN utilizando um Raspberry Pi, a metodologia da inovação foi utilizada para a idealização até a prototipagem. Foi utilizado o Raspberry Pi como o processador, ligado ao LCD para a leitura de dados obtidos, juntamente da IHM programada para a interpretação dos dados obtidos.

O método ágil foi utilizado para melhorar a primeira versão do *datalogger*, na qual foi adotada os *feedbacks* dos pilotos de testes, utilizando um LCD de 7 polegadas ao invés de 3,5, permitindo durante os teste a leitura de dados em tempo real, sem a necessidade de transposição do local de trabalho.

Após a validação da rotina de teste final, é concluída a última etapa do projeto visando a confiabilidade do sistema. A ferramenta *datalogger* via Raspberry Pi desenvolvida permite a leitura de dados da rede CAN veicular, obtendo dados das unidades de controle eletrônica, assim como os atuadores neles conectados, em tempo real. Permite a obtenção de dados intersistêmicos de 4 redes CAN distintas, ou seja, duas leituras veiculares em paralelo.

A interface desenvolvida permite a distinção dos dados dos diferentes projetos de teste, assim como a distinção dos dados por rede CAN, por microcontrolador e por atuador, permitindo uma distinção dos resultados obtidos.

Essa ferramenta ainda possui espaços para ser melhorado. Em trabalhos futuros pode ser desenvolvida uma adaptação da alimentação do Raspberry Pi para ser alimentado diretamente pela tomada OBD-II, ao invés de uma tomada. Permitindo a expansão dos testes para veículos em movimento, não limitando as bancadas de teste.

Por fim, essa ferramenta *datalogger* Raspberry Pi permite um baixo custo para leitura de dados de rede CAN, servindo como mais uma base para validação de teste intersistêmicas.

REFERÊNCIAS

ADJEI, D.; RWAKATIWANA, P. ***Application of traditional and agile project management in consulting firms: a case study of PricewaterhouseCoopers***. Umea: Umea School of Business, 2009. 113 p.

CAMPOS, G. L.; SOUZA, A. G. Rede CAN veicular: levantamento bibliográfico e apresentação de conceitos iniciais. **ForScience:** revista científica do IFMG, Formiga, v. 5, n. 1, jan./jun. 2017. Disponível em: http://www.forscience.ifmg.edu.br/forscience/index.php/forscience/article/view/234/142. Acesso em: 07 maio 2022

CAPELLI, A. **Eletroeletrônica automotiva**: injeção eletrônica, arquitetura do motor e sistemas embarcados. São Paulo: Saraiva, 2010.

CORRIGAN, S. Introduction to the Controller Area Network (CAN). **Application report**, Dallas. 2002. Disponível em: https://www.ti.com/lit/an/sloa101b/sloa101b.pdf. Acesso em: 05 jun. 2022.

COUTINHO, H. **Da estratégia ágil aos resultados**. São Paulo: Saraiva, 2019.

DHAKER, P. ***Introduction to SPI Interface***, Analog Devices, 2023. Disponível em: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html. Acesso em: 05 jun. 2023.

GUIMARÃES, A. A. **Eletrônica embarcada automotiva**. São Paulo: Saraiva, 2007.

ECKERMANN, Erick. **World History of the Automobile**. 1° ed. Pennsylvania, Warrendale: SAE International, 2001.

ELETRONICS, Css. **CAN BUS the ultimate guide**. 1° ed. Dinamarca: CSS ELETRONICS, 2022.

IDEALI, W. **Conectividade em automação e IoT**: protocolos I2C, SPI, USB, TCP-IP entre outros. Funcionalidade e interligação para automação e ToT. Rio de Janeiro: Alta Books, 2021.

MICROCHIP. **MCP2515**, Chandler: Microship, 2018. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf. Acesso em: 16 mai. 2022.

MONROE, S. Basics of debugging the controller area network (CAN) physical layer. **Analog Applications Journal**, Dallas. 2013. Disponível em:https://www.ti.com/lit/an/slyt529/slyt529.pdf?ts=1699869267047&ref_url=https%253A%252F%252Fwww.google.com%252F. Acesso em: 05 jun. 2023.

RASPBERRY, PI. **Raspberry pi model 4b**. Disponível em: https://www.raspberrypi.com/documentation//?page_id=8. Acesso em: 10 de set. 2022.