UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE ENGENHARIA ELETRÔNICA

DIONNI FERNANDO RODRIGUES DE SOUZA

DESENVOLVIMENTO DE UM DISPOSITIVO PARA LEITURA DA REDE CAN VEICULAR

TRABALHO DE CONCLUSÃO DE CURSO

CAMPO MOURÃO
2019

DIONNI FERNANDO RODRIGUES DE SOUZA

DESENVOLVIMENTO DE UM DISPOSITIVO PARA LEITURA DA REDE CAN VEICULAR

Trabalho de Conclusão de Curso, apresentado à disciplina de Trabalho de Conclusão de Curso 2 – TCC2 do curso Superior de Engenharia Eletrônica do Departamento Acadêmico de Eletrônica - DAELN - da Universidade Tecnológica Federal do Paraná (UTFPR) do Campus Campo Mourão, como requisito para obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador: Prof. Dr. Márcio Rodrigues da Cunha

CAMPO MOURÃO

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Campo Mourão
Coordenação de Engenharia Eletrônica

TERMO DE APROVAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO INTITULADO

DESENVOLVIMENTO DE UM DISPOSITIVO PARA LEITURA DA REDE CAN VEICULAR

DO DISCENTE

DIONNI FERNANDO RODRIGUES DE SOUZA

Trabalho de Conclusão de Curso apresentado no dia 26 de abril de 2019 ao Curso Superior de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná, Campus Campo Mourão. O(A) discente foi arguido(a) pela Comissão Examinadora composta pelos professores abaixo assinados. Após deliberação, a comissão considerou o trabalho aprovado com alterações.

Prof. André Luiz Regis Monteiro
Avaliador(a) 1
UTFPR

Prof. Eduardo Giometti Bertogna
Avaliador(a) 2
UTFPR

Prof. Marcio Rodrigues da Cunha
Orientador(a)
UTFPR

“Mas eu não estou interessado
Em nenhuma teoria
Em nenhuma fantasia
Nem no algo mais
Longe o profeta do terror
Que a laranja mecânica anuncia
Amar e mudar as coisas
Me interessa mais.”

Belchior

AGRADECIMENTOS

Faltam palavras para agradecer a todas as pessoas que me ajudaram durante essa longa jornada, que contribuíram um pouco se quer para este momento, alguns com um simples sorriso, outros um abraço, um aperto de mão ou um simples bom dia, qualquer gesto ou atitude, que por mais simples que fosse, no final de tudo fez toda a diferença.

Agradeço aos companheiros de turma, pelos conselhos, pelas brincadeiras, pelos trabalhos em equipe, e acima de tudo pelo apoio e as conversas motivadoras, que me ajudaram muito nos momentos em que estive desanimado.

Agradeço ao Professor Doutor Márcio Cunha que aceitou me orientar nesse trabalho, que para mim foi um desafio e tanto, e nunca deixou de acreditar, agradeço pelas várias ideias postas em pauta e sempre soube das minhas potencialidades exercidas até então, não só como orientado, mas durante todo o curso.

Agradeço imensamente aos meus familiares, a minha Mãe Raquel que foi a minha inspiração, e com ela aprendi a chamar responsabilidade e não desistir durante os desafios da vida por mais difíceis que eles fossem, não me fizeram desistir, e a senhora é um exemplo de superação que levo no peito.

Em memória do meu amado Pai Paulo, a pessoa mais importante durante esse processo e a quem devo a eterna gratidão, foi a pessoa que mais me apoiou não só nesse trabalho, mas na vida, e que infelizmente não pode ver essa concretização, e a quem dedico de todo o meu coração, aonde quer que esteja Pai, saiba que esse trabalho dedico a ti, para compensar todos os momentos que estive ausente, para concretizar essa etapa da minha vida.

RESUMO

SOUZA, Dionni Fernando. **Desenvolvimento de um dispositivo para leitura da rede CAN veicular.** Trabalho de Conclusão de curso – Bacharelado em Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão 2019.

Nesse trabalho foi proposto a construção de um dispositivo que disponibiliza algumas informações de um veículo, tais como rotação por minuto, carga no motor, velocidade e temperatura, a partir de um estudo sobre o protocolo CAN. As leituras foram feitas em um veículo Polo da Wolksvagen do ano 2018, sendo utilizada uma placa de desenvolvimento Eletronic Dev 1.0 contendo um transceptor MCP 2551 conectado na OBD2 do veículo, e um microcontrolador PIC, com código desenvolvido na plataforma MPLAB, possibilitando realizar transmissão e recepção dos dados. Os dados da leitura foram convertidos e transmitidos via USB, exibidos para o usuário em um computador e gerado gráficos dos mesmos, para a correta análise dos resultados.

Palavra-chave: OBD2, Rede Can, MCP2551, Eletronic Dev 1.0.

ABSTRACT

SOUZA, Dionni Fernando. **Development of a device for reading the vehicular CAN network.** Trabalho de Conclusão de curso – Bacharelado em Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná. Campo Mourão 2019.

In this work it was proposed the construction of a device that provides some information of a vehicle, such as rotation per minute, engine load, speed and temperature, from a study on the CAN protocol. The readings were made in a Wolksvagen Polo vehicle from the year 2018, for which an Eletronic Dev 1.0 development board was used which contained an MCP 2551 transceiver connected to the OBD2 of the vehicle, and a PIC controller, developing a code on the MPLAB platform, was possible to transmit and receive the data. The read data has been converted and transmitted via USB and displayed to an user on a computer and generated graphs of the same, for the correct analysis of the results.

Keywords: OBD2, Network Can, MCP2551, Eletronic Dev 1.0

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

CAN	*Controller Area Network*
ISO	*International Organization for Standardization*
OSI	*Open System Interconnection*
SAE	*Society of Automotive Engineers*
Mbit/s	*Megabit por segundos*
CSMA/CR	*Carrier Sense Access with Colission Resolution*
CRC	*Cyclic Redundancy Check*
RTR	*Remote Transmission Request*
DLC	*Data Link Connector*
GPS	*Global Positioning System*
OBD	*On-Board Diagnostic*

SUMÁRIO

1 INTRODUÇÃO

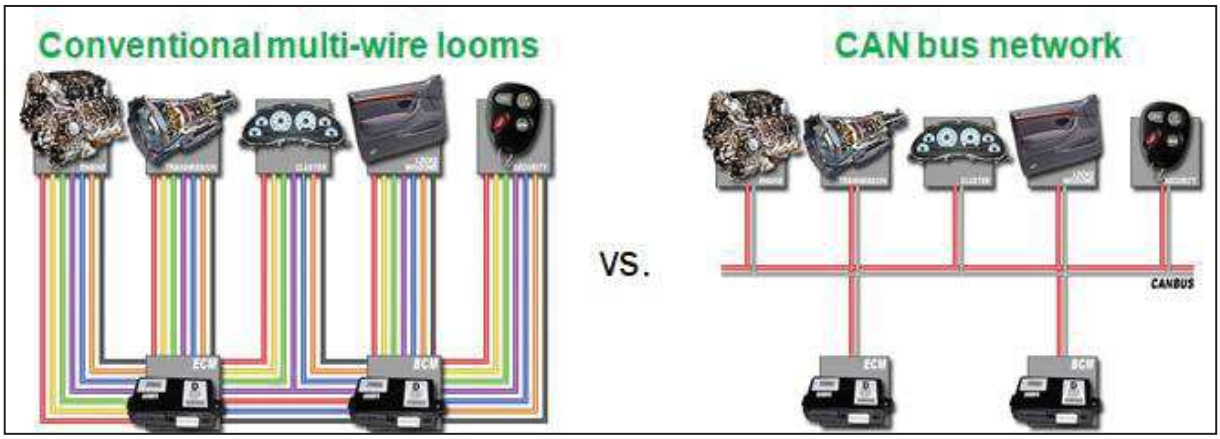
O desenvolvimento do barramento CAN (*Controller Area Network*), surgiu da necessidade de facilitar e simplificar sistemas de comunicações em veículos automotivos. Desenvolver melhorias no sistema interno de um veículo na década de 1980, era necessário que cada vez mais elementos pudessem compartilhar suas informações de forma mais rápida e de maneira mais simples (NASCIMENTO,2006).

Antes da implementação desse barramento os sistemas automotivos eram muito complexos e exigiam uma elevada taxa de cabos e conexões para fazer a comunicação dos seus módulos. Esse protocolo fez com que a quantidade de cabos e conexões, fosse minimizada, dando confiabilidade ao sistema e reduzindo seu peso, seu custo e aumentando a sua velocidade de transmissão de dados, com elevado nível de segurança (CIA,2013).

Assim, em 1983 Robert Bosch desenvolveu o protocolo CAN e seu lançamento se deu em 1986 no congresso da SAE (Sociedade Automotiva de Engenharia), que ocorreu na cidade de Detroit. Logo em seguida foi padronizado internacionalmente pela ISO (*International Organization for Standardization*) (CIA,2013).

A Figura 1 mostra um comparativo entre as redes automotivas antes e depois da inserção do protocolo CAN.

Figura 1 – Comparação da rede automotiva antes e depois do CAN.

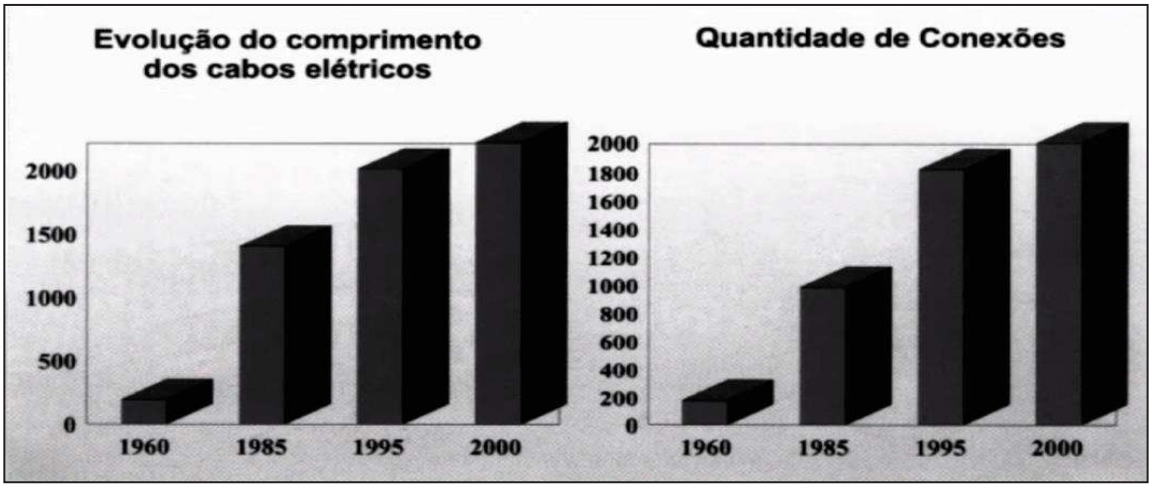


Fonte: Adaptado SCOPINO (2017).

A Figura 2 ilustra a quantidade de cabos e conexões que eram utilizados em um automóvel convencional antes da inserção desta rede, desde a década de 1960 até os anos 2000, que foi o período em que as empresas automotivas viram as

vantagens desse sistema e optaram quase unanimemente pela sua adoção, então os sistemas centralizados foram substituídos por sistemas de redes (SCOPINO,2017).

Figura 2 – Evolução do comprimento dos cabos e conexões de um automóvel.



Fonte: Adaptado SCOPINO (2017).

Com o avanço cada vez maior da eletrônica embarcada nos automóveis e com a redução dos custos de sistemas eletrônicos automotivos no fim do século 20, o número de redes CAN em uso passou de 20 milhões em 1997 para cerca de 120 milhões no ano 2000 (PETROCENTER, 2001).

Como mencionado anteriormente, esse protocolo foi padronizado e documentado pela ISO, gerando assim a norma ISO 11980, para aplicações de alta velocidade, e a norma ISO 11519, para aplicações de baixa velocidade. Esses padrões apresentam a camada física e a camada de enlace de dados do padrão OSI (*Open System Interconnection*) (SOUZA, 2008).

A utilização do protocolo CAN no Brasil aumentou significativamente nos últimos anos devido à obrigatoriedade de aumentar a segurança nos veículos nacionais. Porém, uma das preocupações relativa ao CAN está na sua manutenção, que sem mão de obra qualificada e sem equipamentos especializados, os clientes ficam cada vez mais dependentes de mecânicas despreparadas ou tem que pagar um preço exorbitante para concessionárias, que contam com equipamentos desenvolvidos pelas próprias montadoras, por isso é necessário que sejam desenvolvidas pesquisas, para que o cliente tenha mais alternativas referentes a esse serviço. (MARQUES, 2004).

Os sistemas de controle automotivo necessitam cada vez mais de precisão e pouco tempo de resposta e para possibilitar novas e complexas funções em um sistema automotivo com confiabilidade é necessário que este sistema seja eletrônico. Para garantir que este sistema possa atender todas as situações simultaneamente em um tempo de resposta mínimo, ele deve ser implementado de forma distribuída e integrada. Mas para isso é necessário que se programe controladores que se comuniquem entre si e que processem essas informações com uma alta velocidade e de maneira confiável (MARQUES, 2004).

1.1 OBJETIVOS

1.1.1 OBJETIVOS GERAIS

Desenvolver um dispositivo com interface CAN automotiva para fazer a leitura de parâmetros de um veículo, tais como rotação por minuto, carga no motor, velocidade e temperatura. Estes dados serão interpretados e disponibilizados ao usuário por meio de uma interface serial.

1.1.2 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo deste trabalho, será necessário executar as tarefas listadas a seguir.

- Estudar e compreender o protocolo CAN e suas respectivas normas;
- Fazer um estudo aprofundado sobre comunicação serial, abrangendo leitura, escrita e interpretação de dados;
- Estudar, compreender e aplicar técnicas existentes para efetuar a interpretação de dados seriais gerados por uma rede CAN de um veículo;
- Desenvolver um sistema que faça a leitura de parâmetros e exibam a mesma em uma interface serial a partir de um microcontrolador.

1.1.3 JUSTIFICATIVA

Uma vez que a tecnologia se desenvolve, ocorre um aumento no custo do processo de fabricação e manutenção no ramo automotivo, isso acaba afetando na quantidade de opções no modelos de veículos. Optou-se por esse tema, pois ele possibilita que qualquer usuário possa fazer a leitura dos parâmetros de um veículo, e fazendo uma análise dessas variáveis, a partir de um mapeamento de percurso, pode-se diagnosticar possíveis erros. Isso possibilita uma alternativa, comparando com as disponíveis no mercado, com o objetivo de alcançar pessoas que possuem veículo mais novos.

2 FUNDAMENTAÇÃO TEÓRICA

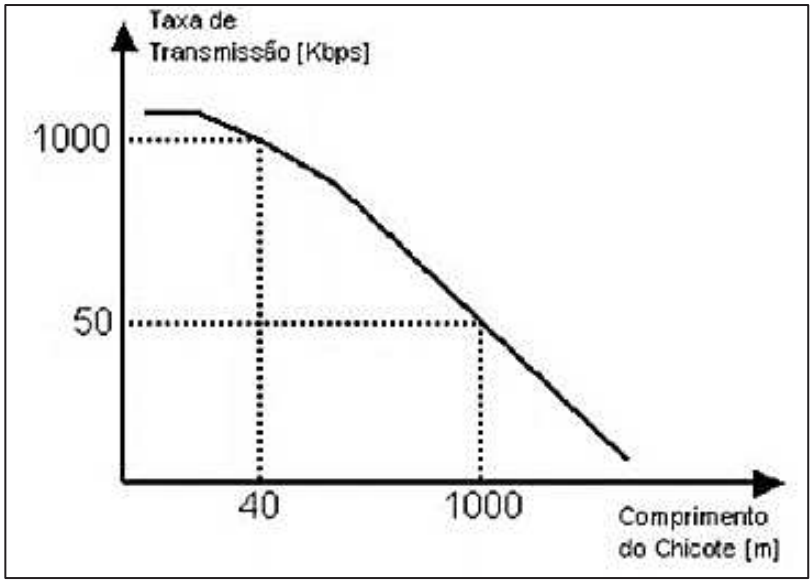
2.1 CARACTERÍTICAS GERAIS DO PROTOCOLO CAN

O CAN é um barramento serial, com comunicação em tempo real, que pode interligar vários dispositivos em apenas uma rede, sendo cada dispositivo tratado como apenas um nó (NASCIMENTO, 2006).

Esse barramento opera geralmente em 1 Mbit/s, podendo ser mais rápido ou mais lento dependendo da sua aplicação. Essa velocidade cai de acordo com o comprimento dessa rede, podendo decair para 50 kbps. Cada um desses nós pode se comunicar simultaneamente a outros nós, e todos os nós podem pedir acesso ao meio de transmissão simultaneamente, pois atua como um sistema de barramento multi-mestre (NASCIMENTO, 2006).

A Figura 3 ilustra um gráfico contendo a relação da taxa de transmissão dessa rede e o seu decaimento com o comprimento.

Figura 3 – Relação taxa de transmissão/distância.



Fonte: Adaptado de GUIMARÃES e SARAIVA (2012).

O protocolo CAN é baseado na técnica CSMA/CR (*Carrier Sense Access with Colission Resolution)* de detecção e resolução de colisões no acesso ao meio de transmissão, quer dizer que em caso de uma colisão, a mensagem de maior prioridade terá o acesso ao canal e outra deverá esperar. Outra característica é a *multicast*, que

estabelece que todos os módulos podem processar a mesma mensagem ao mesmo tempo (HUBERT, 2001).

Quando os dados dessa rede são transmitidos, não existe endereço de destino na mensagem enviada, o caracterizador único recebe o conteúdo, então cabe a cada nó da rede decidir se a mensagem é válida ou não, pois o CAN fornece alguns filtros de aceitação de *hardware* para aliviar o microcontrolador da tarefa de filtrar essas mensagens que são necessárias daquelas que não são de interesse (NASCIMENTO, 2006).

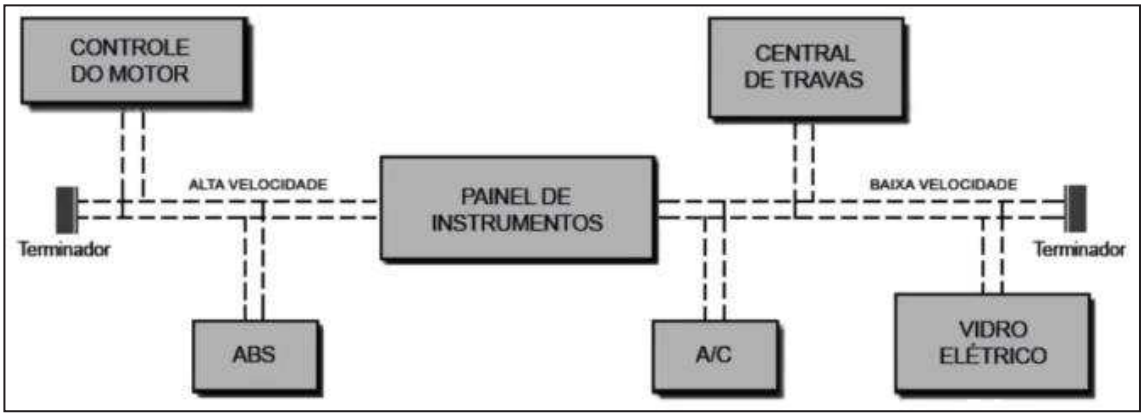
O CAN pode garantir recepção simultânea de uma mensagem por todos os nós da rede ou só de alguns, de acordo com o tipo da mensagem enviada. Esse protocolo também oferece muita flexibilidade, uma vez que com facilidade podem ser adicionados ou retirados os nós dessa rede. Outra característica é que cada módulo registra seus próprios erros e os avalia, podendo corrigir ou desligar dependendo da aplicação, tornando esse protocolo eficaz em ambiente ruidoso (NASCIMENTO, 2006).

2.2 ARQUITETURA DA REDE CAN

A rede CAN interliga quase todos os dispositivos do veículo. Geralmente esses dispositivos têm seus atributos exibidos no painel de bordo, como é o caso da temperatura, da velocidade, do nível de combustível, da bateria entre outros. Esses dispositivos são divididos por módulos que são interligados à central do veículo, que é responsável por captar, interpretar e analisar essas respectivas informações. O número de módulos pode variar de acordo com o fabricante do veículo. Quanto mais atributos o veículo tiver, maior é a necessidade de mais módulos acoplados, com um número maior de dispositivos para cada um deles (SCOPINO, 2017).

A Figura 4 ilustra um exemplo de arquitetura CAN interligada aos dispositivos de um veículo.

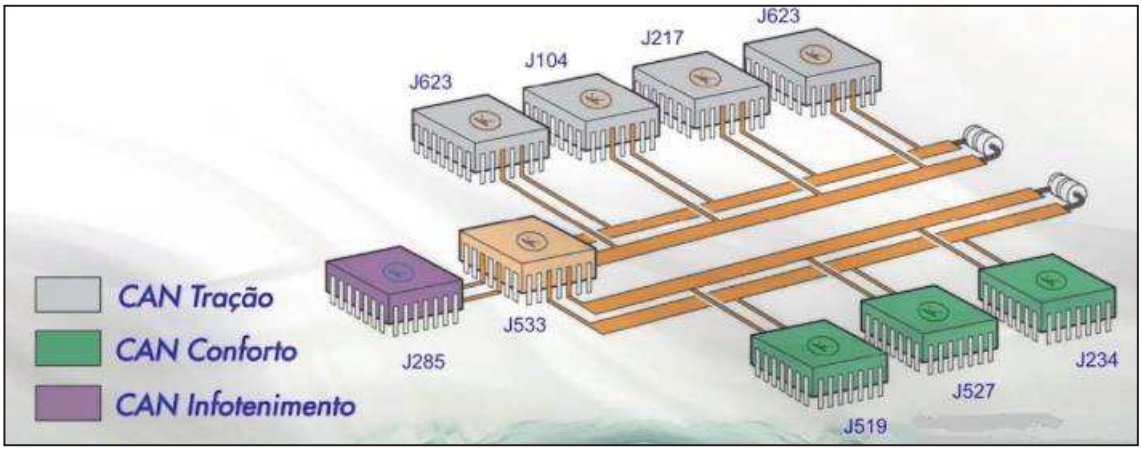
Figura 4 – Arquitetura de uma rede CAN.



Fonte: Adaptado de SCOPINO (2017)

Geralmente, os carros são constituídos por três módulos interligados a central de comando, CAN tração, CAN conforto e CAN infotenimento. Cada um desses módulos tem suas próprias características intrínsecas, como velocidade de comunicação, tempo de resposta e prioridade para o processamento de dados na central. Isso ocorre porque alguns dispositivos no veículo tem mais importância que outros. Um exemplo disso é o acionamento de um *airbag*, o qual tem uma prioridade muito maior na central do que o acionamento de trava de um vidro elétrico. Mesmo que as duas mensagens sejam enviadas no mesmo canal e ao mesmo tempo, a central sempre priorizará a mais importante para aquele determinado momento, pois nessa rede são enviadas várias mensagens por segundo, em uma velocidade extremamente alta. Na Figura 5 é mostrado um exemplo de módulos de um veículo interligados a central de processamento (SCOPINO, 2017).

Figura 5 – Módulos da rede CAN interligado a central.



Fonte: Adaptado de SCOPINO (2017).

2.2.1 CARACTERÍSTICAS DOS MÓDULOS CAN

Nesse tópico serão abordadas as características dos principais módulos da rede CAN de um veículo automotivo.

CAN tração:

- É a rede que interliga as unidades principais do veículo como motor, transmissão, ABS e outros.
- Velocidade de 500 kbit/s.
- Freio de estacionamento e paradas em aclives e declives.
- Podemos ter sub-cans interligadas, como controle de luz na curva, faróis, etc.
- Está interligada diretamente com a unidade de bordo.

A CAN tração é considerada o módulo mais importante do veículo, por gerenciar os elementos para a segurança do indivíduo, pois sua velocidade de dados e sua prioridade é bem maior comparado com os demais módulos (SCOPINO, 2017).

CAN conforto e infotenimento:

- São as redes de alarme, som, sensores de estacionamento, som, imagem, (GPS) *Global Positioning System* e outros.
- Velocidade de 100 Kbit/s.
- Em veículos sem a chave de ignição, partida via botão temos a unidade de coluna para travamento da mesma.

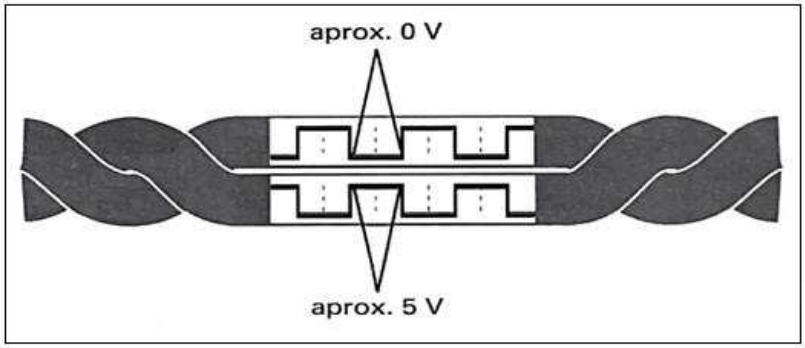
A CAN conforto e infotenimento tem as mesmas características, e em algumas montadoras elas fazem parte de um único módulo. Como pode ser observada em suas características, a velocidade é bem menor que a velocidade da CAN tração (SCOPINO, 2017).

2.2.2 COMPONENTES DO BARRAMENTO CAN E SINAL DE SAÍDA

Essa rede é composta por apenas 2 fios entrelaçados entre si, o que reduz o custo e a complexidade da implementação física. Esses fios atuam com uma tensão entre 0 a 5 volts, com um sinal espelhado entre eles, sendo um fio denominado (*high*) CAN de alta e o outro (*low*) CAN de baixa (SCOPINO, 2017).

A Figura 6 ilustra uma imagem desses 2 fios *high e low* e sua simetria.

Figura 6 – Estrutura dos fios de uma rede CAN.



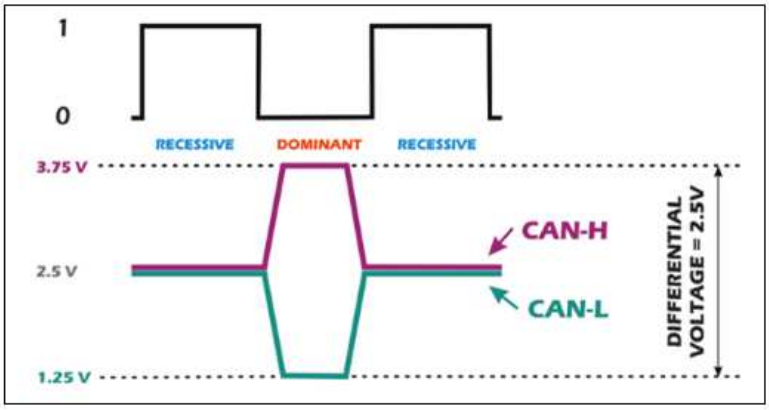
Fonte: Adaptado de SCOPINO (2017).

Observando a Figura 6, nota-se que esses dois fios são entrelaçados entre si, fazendo com que diminua as interferências eletromagnéticas geradas por todo o veículo, mas principalmente pelo motor e alternador já que esses fios estão dispersos em quase todo o espaço do veículo. Sendo assim, eles fornecem uma comunicação de protocolo segura e imune a ruído e interferências de temperatura (HUBERT,2001).

Esses dois sinais têm a mesma sequência de dados, mas suas amplitudes são opostas. Logo se um pulso na linha de CAN de alta for de 2.5 V a 3.75 V, então o pulso correspondente à linha CAN de baixa vai atuar de 2.5V a 1.25V. Enviando esses dados de formas iguais e opostas (SCOPINO, 2017).

Caso o estado da tensão diferencial for 2,5 V o estado é dominante, caso contrário o estado é recessivo. Na Figura 7 é mostrado esse funcionamento e os níveis de tensão para cada CAN.

Figura 7 – Tensão nos fios de *high* e *low* da rede CAN.

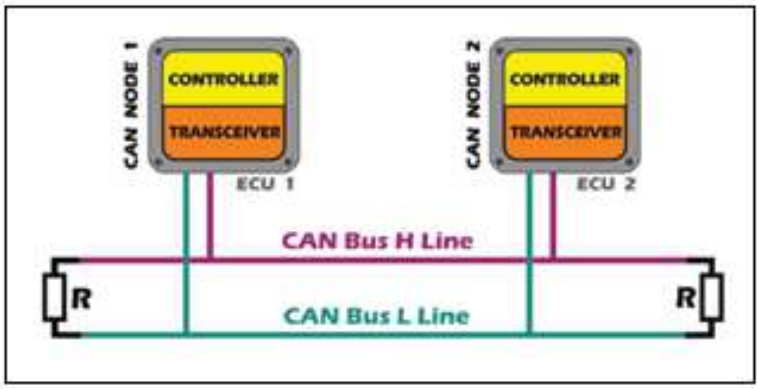


Fonte: Adaptado de MUCEVSKI (2015).

Esse par de fios entrelaçados é terminado geralmente com um resistor de 120 ohms em cada ponto da linha. Isso se deve ao fato dessa transmissão operar de forma diferencial. Para reduzir ainda mais essa sensibilidade contra a interferência eletromagnética, essas linhas podem ser envolvidas por malhas blindadas, reduzindo dessa forma a irradiação eletromagnética do próprio barramento, principalmente em elevadas taxas de transmissão (NASCIMENTO, 2006).

A Figura 8 mostra um exemplo, com dois dispositivos conectados, bem como os componentes que compõem a rede CAN, controlador CAN e o transceptor CAN, juntamente com os resistores no terminal da linha. O controlador recebe os dados de transferência da central, então ele processa esses dados e o transmite para o transceptor, além disso, o controlador pode receber os dados do transceptor processar e enviá-los para a central. O transceptor é um elemento que converte os dados que o controlador fornece em sinais elétricos e os envia pela rede, e os resistores evitam que os dados enviados sejam refletidos nas extremidades e retornando como um eco (MUCEVSKI, DATA).

Figuras 8 – Componentes de controle e transmissão da rede CAN.



Fonte: Adaptado de MUCEVSKI (2015).

2.3 FORMATO DAS MENSAGENS ENVIADAS NA REDE CAN

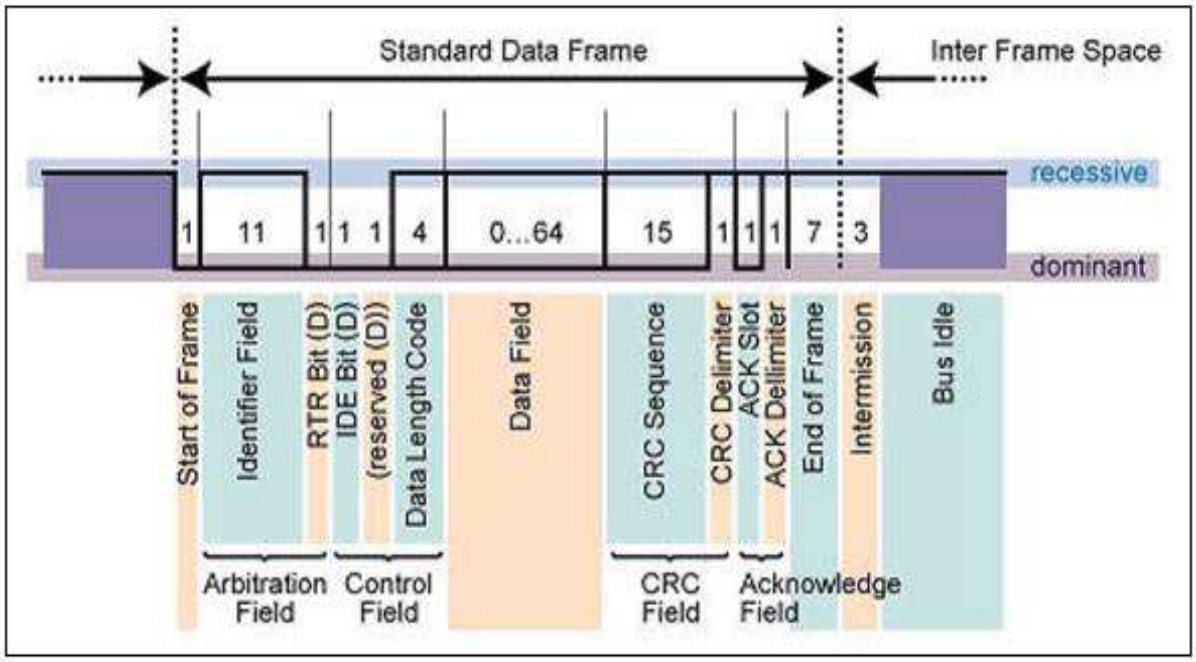
Os campos de identificação da mensagem na rede podem ter 11 bits (padrão) para versão CAN 1.0 ou 29 bits (estendido) para versão CAN 2.0, sendo que essas duas versões são totalmente compatíveis (REUSS,1993).

As mensagens enviadas pela rede CAN, são em formatos de quadro, denominados como *frame*, e nesses *frames* estão todas as informações contidas, os *frames* mais importantes são o *remote frame* e o *data frame* e os quadros de erro como *error frame* e quadro de sobrecarga *overload frame*. Durante a transmissão o quadro atual é separado do interior por espaços de interquadros denominados *interframespace*. O CAN utiliza mensagens de até 64 bits, essas mensagens não tem endereço, pois é através de seu conteúdo que a estação que recebe essa mensagem faz o reconhecimento (MARQUES, 2004).

2.3.1 QUADRO DE DADOS PADRÃO CAN 1.0

A Figura 9 mostra o Quadro de dados Padrão CAN 1.0. O padrão CAN 1.0 apresenta 5 campos (*Arbitration Field*, *Control Field*, *Data Field*, *CRC Field* e *Acknowledge Field*) delimitados por um bit de início (*Start of Frame*) e 7 bits de fim (*End of Frame*).

Figura 9 – Quadro de dados Padrão CAN 1.0.



Fonte: Adaptado de NASCIMENTO (2006).

Segue o Quadro 1, detalhando os campos da Figura 9.

Quadro 1 – Descrição do quadro padrão CAN 1.0.

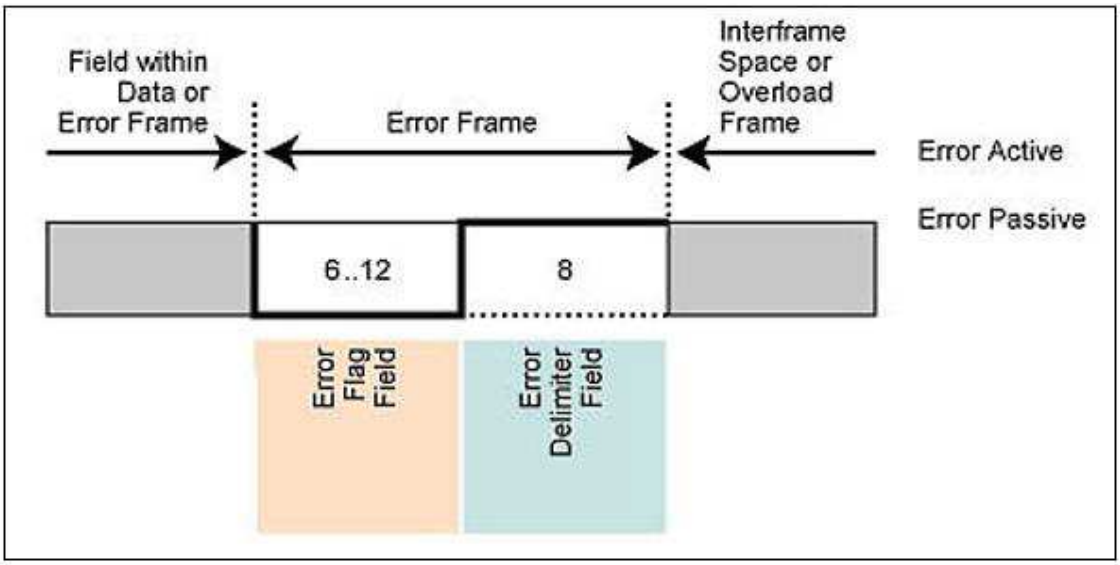
Setor	Número de bits	Descrição
Start of Frame	1	Realiza a sincronização dos nós receptores.
Identifier Field	12	Refletem o conteúdo e a prioridade da mensagem, mais o bit de transmissão remota.
Control Field	6	O primeiro bit é o identificador de extensão, o bit seguinte define se ele é dominante, e os 4 bits restantes definem o tamanho dos dados.
Data Field	1 a 64	Tamanho do conteúdo da informação.
CRC Field	15	Utilizado para apontar erros de transmissão, completada por um delimitador recessivo.
Acknowledge Field	2	Acusa o erro e reconhece a correta recepção, os 7 bits finalizam o quadro de dados.

Fonte: Autoria própria.

2.3.2 QUADRO DE ERRO

A Figura 10 mostra o quadro de erro e o posicionamento dos bits. O quadro de erro é gerado, quando qualquer nó detecta um erro. Existem erros ativos e passivos, a diferença entre ambos é que no ativo, todas as estações o reconhecem e geram um quadro de erro, o erro passivo só afeta o nó que está transmitindo de fato (BOSCH, 1991).

Figura 10 – Quadro de erro.



Fonte: Adaptado de NASCIMENTO (2006).

Segue o Quadro 2, detalhando os campos da Figura 10.

Quadro 2 – Descrição do quadro de erro.

Setor	Número de bits	Descrição
Error Flag Field	6 a 12	O erro ativo é composto por uma sequência de 6 a 12 bits recessivos, e o erro passivo é composto por 14.
Error Delimiter Field	8	Faz a classificação desse erros.

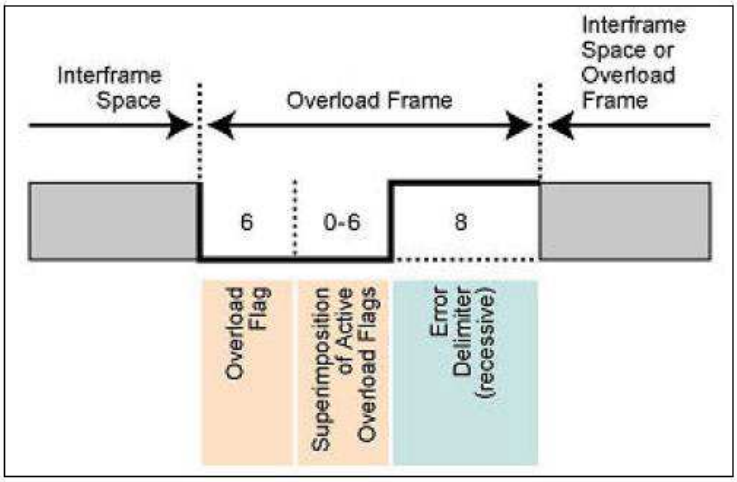
Fonte: Autoria própria.

2.3.3 QUADRO DE SOBRECARGA

A Figura 11 mostra o quadro de sobrecarga e o posicionamento dos bits. Esse quadro tem o mesmo formato que o de erro, a diferença é que esse só pode ser gerado durante espaço entre quadros. Ele é composto por 2 campos, um de *flag* de sobrecarga seguido por um delimitador (BOSCH, 1991).

Um nó pode ter no máximo 2 quadros de sobrecarga e só pode ser transmitido do primeiro bit do espaço entre quadros. Isso acontece para distinguir do quadro de erro.

Figura 11 – Quadro de sobrecarga.



Fonte: Adaptado de NASCIMENTO (2006).

Segue o Quadro 3, detalhando os campos da Figura 11.

Quadro 3 – Descrição do quadro de sobrecarga.

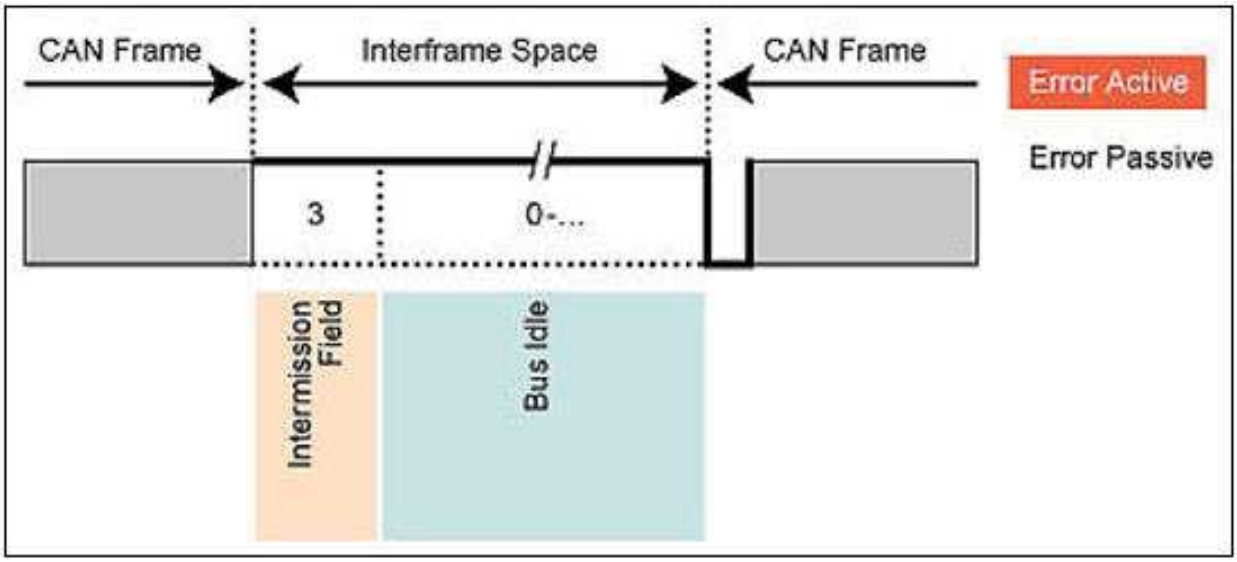
Setor	Número de bits	Descrição
Overload Flag	12	Esse flag tem 6 bits dominantes, seguidos por mais 6 de sobrecarga.
Error Delimiter	8	O delimitador consiste em oito bits recessivos.

Fonte: Autoria própria

2.3.4 ESPAÇO ENTRE QUADROS

Tem como função separar um quadro de qualquer tipo de um dado remoto, é composto por 3 bits recessivos. Esse quadro é provido para permitir que os nós tenham um tempo de processo interno antes que se inicie o próximo quadro. A Figura 12 ilustra o quadro de espaços e o posicionamento dos bits.

Figura 12 – Quadro de espaço entre quadros.



Fonte: Adaptado de NASCIMENTO (2006).

Segue o Quadro 4, detalhando os campos da Figura 12.

Quadro 4 – Descrição do espaço entre quadros.

Setor	Número de bits	Descrição
Intermission Field	3	Esse quadro é provido para permitir que os nís tenham um tempo de processo interno antes que se inicie o próximo quadro.

Fonte: Autoria própria.

2.4 PRIORIDADE DE MENSAGENS NO BARRAMENTO CAN

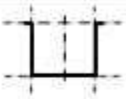
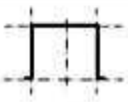
Toda informação enviada à central é definida por uma prioridade. Isso serve para verificar se ela é mais ou menos importante, e se ela deve ser processada ou não primeiro. A prioridade na rede CAN é definida pelo campo do identificador, como explicado no tópico 2.3, que além de definir o conteúdo da mensagem, também defini a sua prioridade de processamento do dado enviado para central, ou seja, ele faz a identificação e informa qual é o tipo da mensagem (temperatura do motor, velocidade e rotação) e define a sua prioridade, quanto menor for o valor do identificador maior

será a sua prioridade. Isso é importante para a atribuição do barramento, quando várias estações competem pelo mesmo endereço (HUBERT,2001).

2.5 FUNCIONAMENTO SISTÊMICO DA REDE CAN

Um parâmetro do veículo interpretado pela central e exibido no painel de bordo para o usuário, depende do nível de tensão nos fios *high* e *low* e como eles irão interagir entre si durante o envio do dado. São delimitados alguns padrões para interpretação desse dado pela central, ou seja, é possível saber o que está ocorrendo dentro do veículo só com a leitura correta dos sinais de *high* e *low*. A Figura 13 tem um exemplo genérico de recepção e interpretação de dados CAN para uma aplicação em um veículo, da informação do funcionamento do vidro elétrico e a temperatura do motor. Apenas reforçando que esse funcionamento é apenas um exemplo de aplicação, mas esse modelo pode ser usado para qualquer tipo de parâmetro a ser analisado (SCOPINO, 2017).

Figura 13 – Exemplo de funcionamento de interpretação de dados da rede CAN.

POSSÍVEL VARIÁVEL	PRIMEIRO BIT	SEGUNDO BIT	REPRESENTAÇÃO GRÁFICA	INFORMAÇÃO ESTADO DO VIDRO ELÉTRICO	INFORMAÇÃO TEMPERATURA DO LÍQUIDO REFRIGERANTE
Um	0 Volt	0 Volt		em movimento	10°C
Dois	0 Volt	5 Volt		em repouso	20°C
Três	5 Volt	0 Volt		na zona de início de parada	30°C
Quatro	5 Volt	5 Volt		detecção do bloqueio superior	40°C

Fonte: Adaptado de SCOPINO (2017).

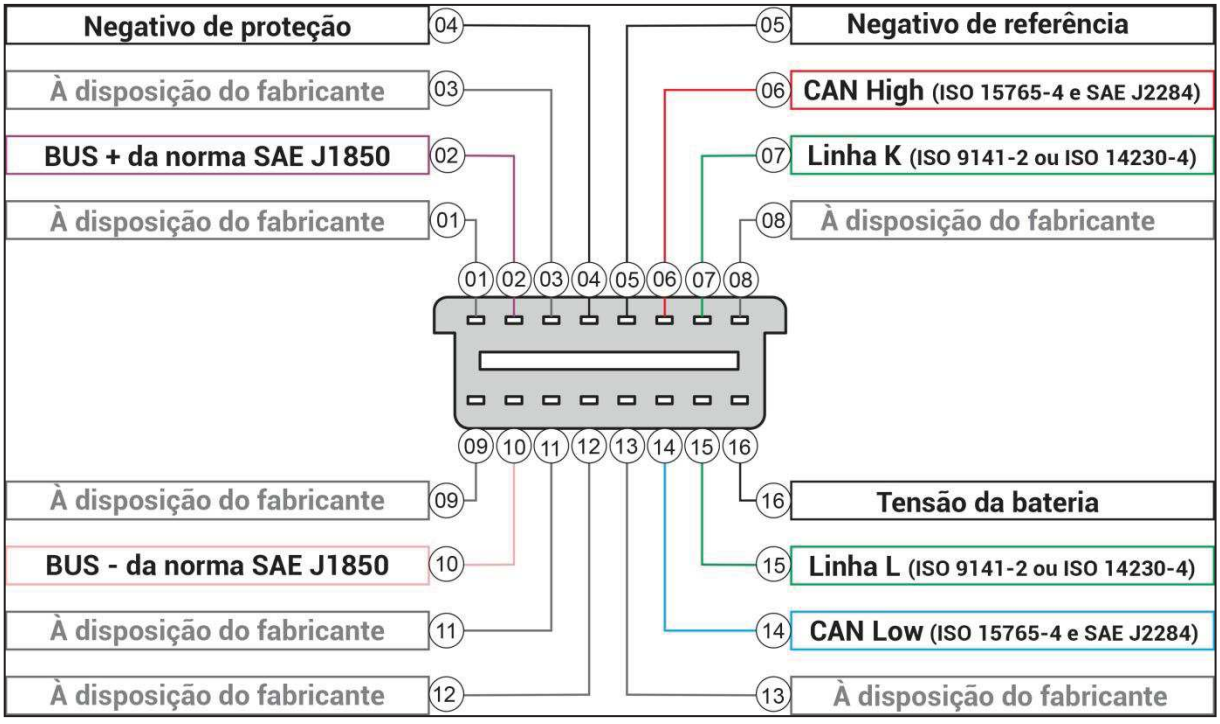
Observando atentamente a Figura 13, é possível perceber nesse exemplo, quatro possibilidades de parâmetros e informações que podem ser transmitidas para a central, apenas com 2 sinais de *high* e *low*.

2.6 ENTRADA OBD 2

A Figura 14 mostra uma entra OBD 2 (*On-Board Diagnostic*) que é um conector padrão, composto por uma entrada serial de 16 pinos, utilizado em quase todos os automóveis fabricados depois do ano 1996. Esse conector está interligado à rede CAN do veículo, e essa ligação fornece uma conexão padronizada, para fazer o diagnóstico da rede.

A função de alguns pinos deve seguir os padrões exigidos pela SAE, outros ficam a critério do próprio fabricante. Os veículos atuais tem muitos módulos com muitas informações importantes que não são disponibilizadas pelo fabricante. Pode-se ter acesso a essas informações, fazendo uma leitura correta dos pinos desse conector, pois a maioria das informações do funcionamento do veículo estão contidas nele (SINDIREPA,2017).

Figura 14 – Pinos da entrada OBD 2.



Fonte: Adaptado de CICLO ENGENHARIA EM INFORMAÇÃO AUTOMOTIVA (2016).

2.6.1 NORMAS DE TRANSMISSÃO DE DADOS DO CONECTOR

Como mostrado na Figura 14, existem cinco tipos de protocolos OBD (On-Board Diagnostic) em uso: J1850 PWM, J1850 VPW, ISO 9141-2, ISO 14230 KWP 2000 e ISO 15765 CAN. Cada um desses protocolos difere eletricamente e pelo seu formato de comunicação. Abaixo está a descrição dos protocolos mais utilizados:

Norma ISO 15765-4:

- Utiliza o pino 6 para CAN-alto e o pino 14 para CAN-baixo, e o pino 4 e 5 para o terra.
- A sua velocidade de transmissão e recepção varia entre 250 Kbit por segundos a 500 Kbit/s.

Norma ISO 9141-2:

- Comunicação serial assíncrona a 10,4 Kbauds.
- Pode ter até 12 bytes de mensagem, excluindo delimitadores de quadro *Bit Timing*.
- Utiliza o pino 7 para a comunicação bidirecional, o pino 4 e 5 para o terra, e opcionalmente pode utilizar o pino 15 unidirecional para ativar a ECU (Unidade eletrônica de controle).
- Tem sinalização UART 8 bits de dados, sem paridade com 1 *stop*.

Norma J1850 PMW:

- Utiliza modulação por largura de pulso a 41,6 kbps, diferencial de 12 fios.
- Pode ter até 12 bytes de mensagem, excluindo delimitadores de quadro *Bit Timing*.
- Utiliza o pino 2 para sinal CAN alto e o pino 10 para baixo, o pino 4 e 5 para o terra.
- E também utiliza um bit de sincronismo.

3 METODOLOGIA

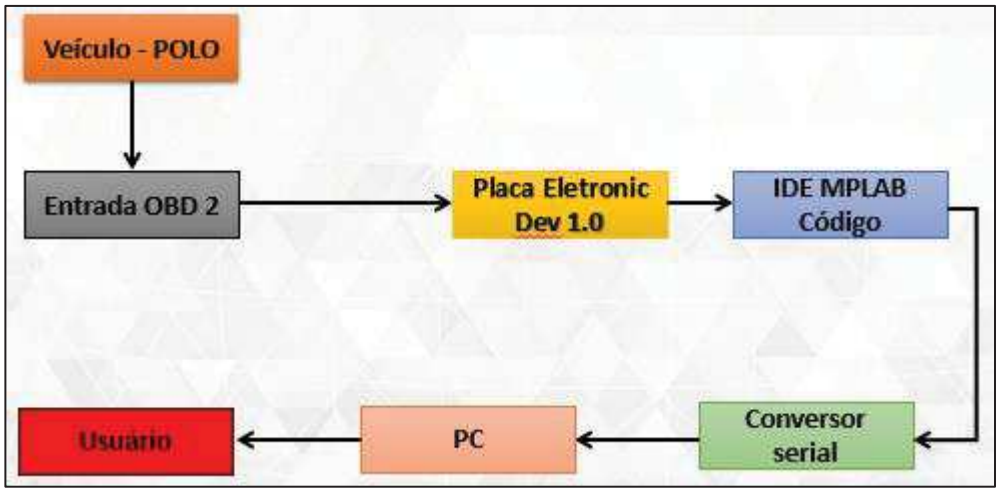
Para a execução deste trabalho foram utilizados os elementos listados abaixo:

- Veículo Polo Volkswagem do ano 2018;
- Placa Eletronic Dev 1.0;
- IDE MPLAB;
- Conversor UC-1000;
- Placa CAN-Bus Shield Arduino 1.2.

3.1 DIAGRAMA DE BLOCOS DO SISTEMA

Uma demonstração básica do que esquema realizado, é representado na Figura 15.

Figura 15 – Esquema básico do sistema.



Fonte: Autoria própria.

O diagrama básico mostra que a partir da entrada OBD2 do veículo, ligada na placa de desenvolvimento, os dados do veículo são interpretados a partir de um código e exibidos ao usuário pelo computador através de um conversor UC-1000.

3.2 VEÍCULO

Para fazer a leitura e a interpretação dos dados a partir da entrada OBD2 foi utilizado um veículo Polo fabricado no ano de 2018 pela Volkswagen. Nesse veículo em específico a entrada OBD2 está localizado no lado do motorista no compartimento de passageiros perto do console central. Nas imagens abaixo é possível observar o veículo utilizado, juntamente com sua entrada OBD2. Os detalhes e especificações desse veículo serão explicados nos próximos tópicos.

Figura 16 – Veículo Polo.

Fonte: Autoria própria.

Figura 17 – Entrada OBD 2 do Polo.

Fonte: Autoria própria.

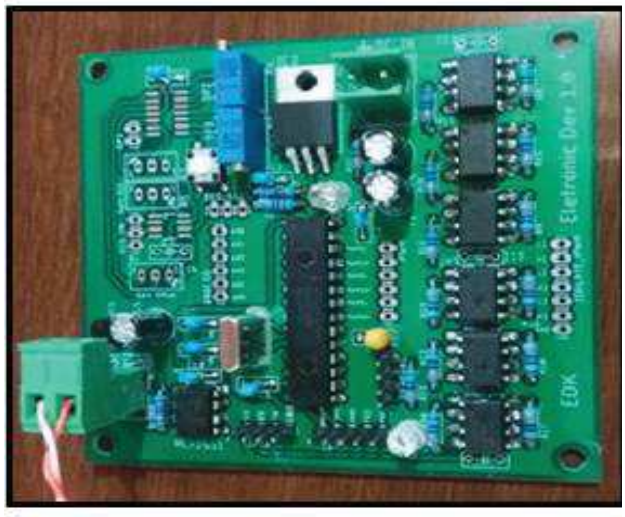
3.3 PLACA ELETRONIC DEV 1.0

Foi a placa de desenvolvimento escolhida para o desenvolvimento deste trabalho, pois a mesma atende os requisitos para a elaboração de aplicações com o protocolo CAN, de transmissão leitura e escrita, também levando em consideração aplicações de interpretação do protocolo, com um volume de informações não elevado. Essa placa tem um microcontrolador dsPIC33EP64MC502 e o transceptor 2551, que serão detalhados nos próximos tópicos. Logo abaixo pode se observar algumas características dessa placa voltadas para aplicações com o protocolo CAN que é o foco desse trabalho:

- Implementa o CAN V2.0B com velocidade de transmissão de até 1 Mbit/s.
- Trabalha com o padrão OBD1 (11 bits) e OBD2 estendido (29 bits) e quadros remotos.
- Indicadores de LED (Diodo Emissor de Luz) para sinalização de gravação, escrita e ligado e desligado.
- Opera com transmissão de dados via saída serial e UART (Universal Asynchounous Receiver/Transmiter).

Essa são apenas algumas possíveis aplicações dessa placa de desenvolvimento. A Figura 18 mostra a placa Eletronic Dev 1.0.

Figura 18 – Placa Eletronic Dev 1.0.



Fonte: Autoria própria.

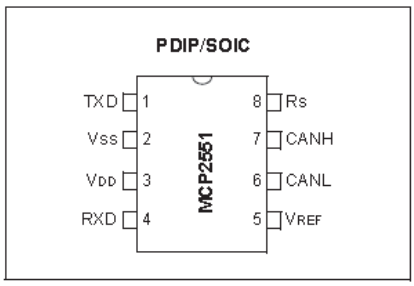
3.3.1 TRANSCEPTOR MCP 2551

Esse transceptor suporta uma velocidade de operação de 1 Mbit/s, ele é adequado para sistemas de 12V e 24V, tem proteção contra danos devido às condições de curto-circuito, e proteção de desligamento térmico automático, ele pode controlar até 112 nós na rede, e tem alta imunidade de alto ruído.

O MCP2551 pode trabalhar em uma alta velocidade, ele serve como interface entre um controlador do protocolo CAN e o barramento físico, esse dispositivo permite que o controlador transmita, receba e controle os dados do protocolo CAN.

Cada nó em um sistema CAN deve ter um dispositivo para converter os sinais digitais gerados por um controlador para os sinais apropriados para transmissão sobre o barramento. Na Figura 19 é mostrado os pinos do MCP 2551.

Figura 19 – Pinagem do MCP 2551.



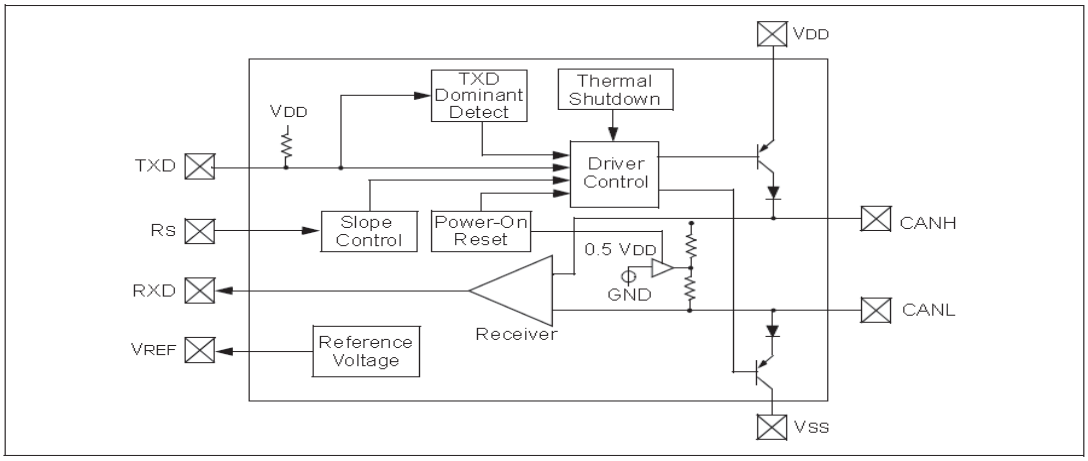
Fonte: MICROCHIP (2010).

Seguem abaixo a listagem dos pinos desse circuito integrado, referente a Figura 19.

1. **(TXCAN)** – Pino transmissor para o barramento CAN.
2. **(Vss)** – Pino do terra
3. **(VDD)** – Pino de alimentação
4. **(RXS)** – Pino receptor para o barramento CAN
5. **(Rs)** – Pino de buffer de transmissão.
6. **(CANH)** – Pino CAN alta
7. **(CANL)** – Pino CAN baixa
8. **(VREF)** – Pino de referência

Para compreender melhor o funcionamento do dispositivo, na Figura 20 é mostrado o diagrama de bloco do MCP 2551.

Figura 20 – Diagrama de blocos do MCP 2551.



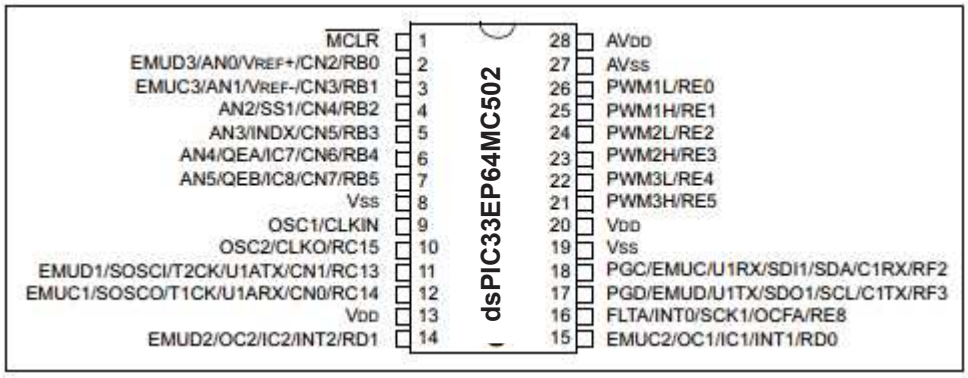
Fonte: MICROCHIP (2010).

Detalhados todos os pinos do transceptor, é importante qualificar os pinos TXD e RXD, correspondentes aos pinos 1 e 4, que são os responsáveis por receber e fornecer o sinal, o qual será feito a interpretação das mensagens CAN pelo microcontrolador.

3.3.2 PIC33EP63MC502

Esse PIC, que nesse trabalho denominaremos como controlador, já que o mesmo que tem o papel de configurar todas as informações, implementa o CAN V2.0 com uma velocidade de transmissão de até 1 Mbit/s. O comprimento dos campo de dados é de até 8 bytes, incluindo dados padrão, estendidos e quadros remotos. Na Figura 21 é possível observar os pinos do dsPIC33EP64MC502.

Figura 21 – Pinagem do dsPIC33EP64MC502.

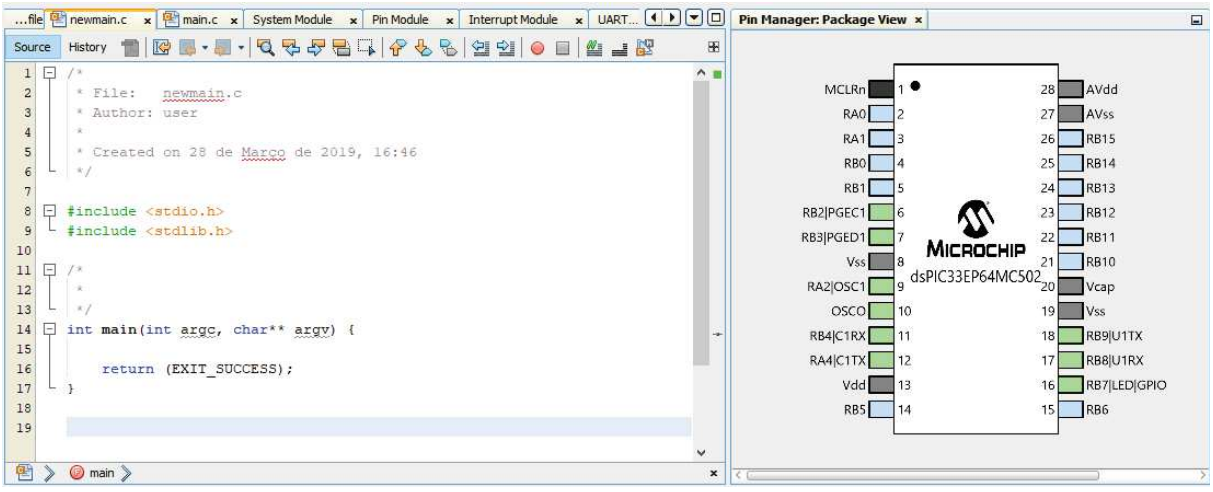


Fonte: Adaptado de MICROCHIP (2005).

3.4 MPLAB

O MPLAB é um *software* editor para projetos de programação, Essa plataforma é mais eficiente e usual para aplicações com microcontroladores PIC. Neste trabalho ela foi utilizada, pela sua simplicidade de aplicação, e também porque ela é fornecida pela *Microchip Technology* gratuitamente. Na Figura 22 é possível observar essa IDE.

Figura 22 – IDE MPLAB.



Fonte: Autoria própria.

Essa IDE integra vários moldes de trabalho de simulação e gravação, para vários modelos de microcontroladores. O código implementado nesse trabalho foi feito em C, pois é a linguagem de programação mais usual, e que mais se encontra bibliografias a respeito do assunto.

3.5 CONVERSOR UC-1000

Esse conversor foi necessário para exibição dos valores da leitura dos parâmetros do veículo para o usuário através da plataforma serial disponibilizada no próprio MPLAB, na Figura 23 é possível observar o conversor.

Figura 23 – Conversor UC-1000.

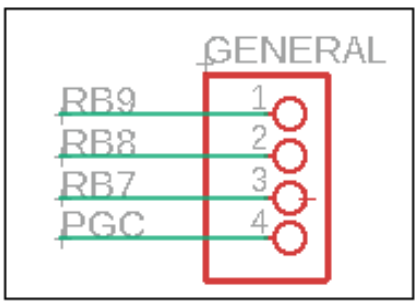
Fonte: Adaptado de INHAOS.

Os detalhes de funcionamento, estes serão explicados no decorrer do trabalho, pois esse dispositivo exerce uma função indispensável no sistema, uma vez que somente com ele permite interpretar a interface com o usuário.

3.5.1 TESTE DO UC-1000 NO SISTEMA

O conversor é ligado a um bloco específico com os pinos 11 e 12, RX e TX, onde é conectado o conversor para a exibição dos valores para o usuário na IDE, o bloco que intermedia o conversor com o computador está na Figura 24.

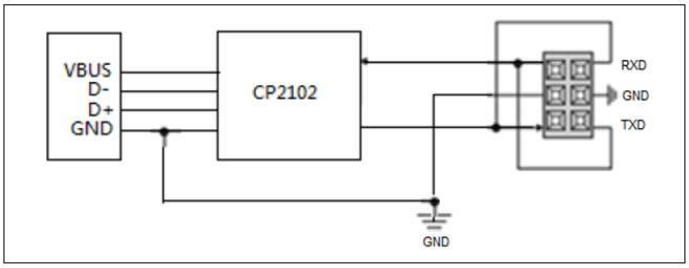
Figura 24 – Bloco conversor.



Fonte: Adaptado de Knoner (2018).

Os dados de transmissão e recepção do controlador através do barramento, são transmitidos pelo protocolo UART, e para serem exibidos na IDE para usuários, devem ser convertidos para USB, o que justifica a escolha desse conversor. Na Figura 25 é possível observar o esquema elétrico do Conversor UC-1000.

Figura 25 – Esquema Conversor UC-1000.



Fonte: Adaptado de INHAOS.

Para utilizar esse dispositivo no ambiente MPLAB foi necessário a instalação de outro *plugin Simple Serial Port Terminal*, para fazer a comunicação. Na Figura 26 pode-se observar o teste que foi feito para dar validação a esse terminal para exibição desses valores.

Figura 26 – Teste de interface para o usuário

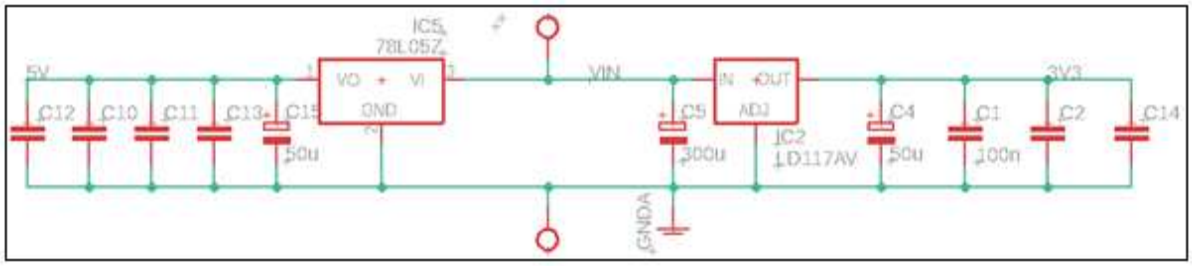
Fonte: Autoria própria

Na Figura 26 é possível observar que a comunicação do conversor entre a placa e a IDE através do plug-in foi satisfatória, pois ela exibe os valores requeridos na tela para o usuário. É importante lembrar que a velocidade de comunicação desse conversor é a mesma do barramento CAN, ou seja 500 kbit/s.

3.6 CONCEITOS DO HARDWARE

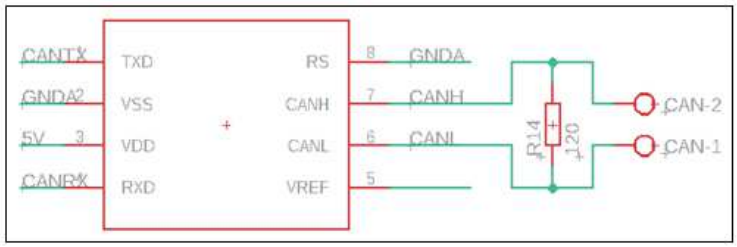
Para a compreensão do funcionamento do dispositivo e todas as suas interações nesse sistema, é indispensável compreender as ligações do transceptor em relação ao microcontrolador e sua alimentação, para assim desenvolver o código que fará a interpretação dos dados. As Figuras 27, 28 e 29, mostram como estão dispostas todas as ligações internas dessa placa em relação a esses componentes em específico, que estão divididos em 3 imagens, alimentação, transceptor e controlador, que podem ser observadas logo abaixo.

Figura 27 – Alimentação Eletronic Dev 1.0.



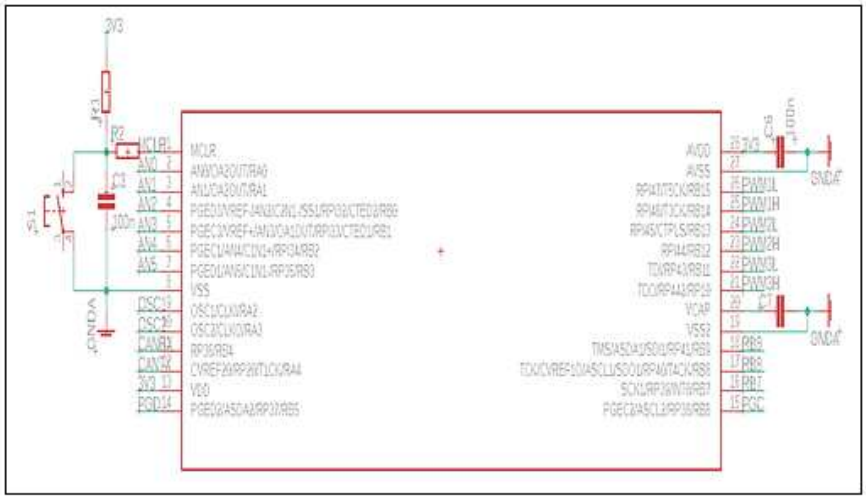
Fonte: Adaptado de Knoner (2018).

Figura 28 – Transceptor da placa Eletronic Dev 1.0.



Fonte: Adaptado de Knoner (2018).

Figura 29 – Controlador da placa Eletronic Dev 1.0.



Fonte: Adaptado de Knoner (2018).

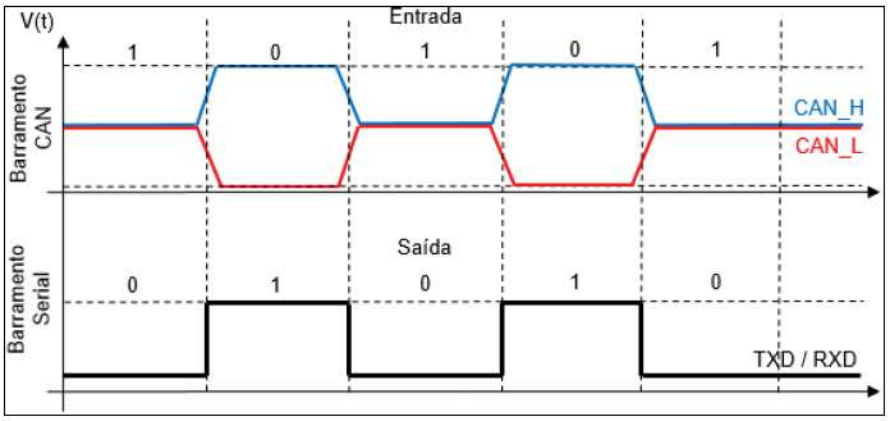
Analisando a Figura 27, é possível observar o bloco de alimentação dessa placa de desenvolvimento, o mesmo recebe uma tensão externa do pino 16 da OBD do próprio veículo e o terra do 4. Essa tensão é fornecida direto da bateria de 14 V, então quando essa tensão chega ao borne da placa, é direcionada para um regulador de tensão que rebaixa essa tensão e a direciona para o controlador e o transceptor dessa placa. Recapitulando a Figura 14 é possível fazer essa análise de maneira simples.

Analisando a Figura 28 é possível compreender como o transceptor dessa placa está disposto. No pino 7 e 6 vão ser conectados diretamente com os pinos 6 e 14 da OBD2 do veículo respectivamente, para assim receberem os sinais do veículo durante o processo de leitura, isso ocorre porque o veículo em questão utiliza da norma ISO 15765-4, para leitura de dados. Ao receber o sinal do veículo, ele funciona apenas como *hardware*, comparando as tensões de chegada.

É importante entender que o sinal de CAN H e CAN L, tem a mesma amplitude, por serem simétricos e enviado aos pares, só que com valores opostos. Se esses dois sinais que passam pelo comparador tiverem uma amplitude mínima de 2,5 V esse sinal no barramento é dominante.

A cada bit recessivo decodificado do transceptor, é enviado por um bit 1 via serial pelo pino TXD/RXD, e a cada bit dominante, é enviado um bit 0 como mostrado na Figura 30, isso faz com que as mensagens sejam transferidas bit a bit do transceptor para o controlador. Na Figura 30 é possível compreender o funcionamento desse dispositivo em relação aos sinais de entrada:

Figura 30 – Diagrama de entrada e saída do transceptor MCP 2551.



Fonte: Autoria própria.

Por último, analisando a Figura 30, é possível observar como o controlador está ligado à placa. Os pinos do transceptor TX e RX, então ligados ao bloco da serial, que direcionam para os pinos 11 e 12 do controlador, eles passam pelo bloco serial que faz essa intermediação. Quando o sinal do barramento é dominante o pino 11 RXD do controlador, recebe um sinal recessivo, e quando o sinal no barramento é recessivo, o sinal nesse pino é dominante. O mesmo comportamento ocorre no pino 12 TXD do controlador. A função desse transceptor é a recepção do CAN H e CAN L do veículo fazendo interface do barramento como o controlador, funcionando apenas como *hardware*, comparando as tensões de chegada.

Esse sinais de tensão que são gerados a partir do transceptor são transferidos para o pinos TX e RX do controlador, ele recebe esses dados, trata o fluxo de informação transmitida e os identifica e interpreta.

3.7 VELOCIDADE DO SISTEMA

Foi necessário compreender a velocidade no qual a ECU do veículo executa suas informações, para assim ajustar a velocidade do barramento através do controlador.

Uma vez que o mesmo fabricante de veículos pode utilizar protocolos diferentes para um mesmo modelo. Pode-se utilizar o site klavkarr, que é voltado para aplicações com o ELM327, e tem nele contido especificações do protocolo CAN para mais de 10000 veículos. Pesquisando o modelo de carro utilizado nesse trabalho, Polo 5 2018, foi possível obter dados importantes, como a versão e a velocidade da comunicação

do mesmo, na Figura 31 tem as especificações do veículo escolhido disponibilizados pelo site em questão.

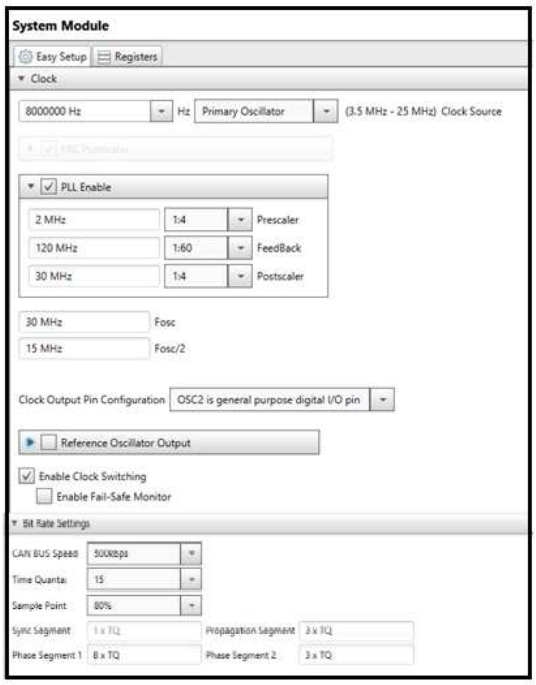
Figura 31 – Especificações do protocolo do veículo.

Make (1)	Model	Engine	AM	Fuel	P (2)	Protocol	Mode 1	Mode 2	Mode 5 (3)	Mode 6	Mode 7	Mode 9
Volkswagen (x2)	Polo 5	1.6 TDi	2010	Diesel	75	CAN 11bit 500kb	983BA013 B019A001 CCD20000	583B8003 20182001 4CD00000	No	80000001 00008001 00000001 00000001 08000001 00004000	Yes	54600000
Volkswagen (x6)	Polo	1.6 TDi	2010 2011 2015	Diesel	90	CAN 11bit 500kb	983BA013 B019A001 CCD20000	583B8003 20182001 4CD00000	No	80000001 00008001 00000001 00000001 08000001 00004000	Yes	54600000
Volkswagen (x2)	Polo 5	1.2 TSI	2013 2018	Gas	90	CAN 11bit 500kb	BE3EB813 A007A011 FED08400	7E3E8003 20062001 7ED08400	No	C0000001 80000809 C0000001 00000001 00000001 78000000	Yes	55400000
Volkswagen	Polo 5	1.2 TSI	2011	Gas	105	CAN 11bit 500kb	BE3EA813 A005B011 FED00400	7E3E8003 20042001 7ED00400	No	C0000001 80000009 C0000001 00000001 00000001 78000000	Yes	54400000

Fonte: Autoria própria.

Uma vez que se sabe a velocidade de transmissão do veículo 500 kbit/s, para o recolhimento de dados, utilizou-se um *plugin* denominado MCC voltado para aplicações de rede CAN e executado pelo MPLAB, para facilitar as configurações de transmissão de dados pelo controlador. Na Figura 32 é possível observar as configurações.

Figura 32 – *Plugin* MCC.



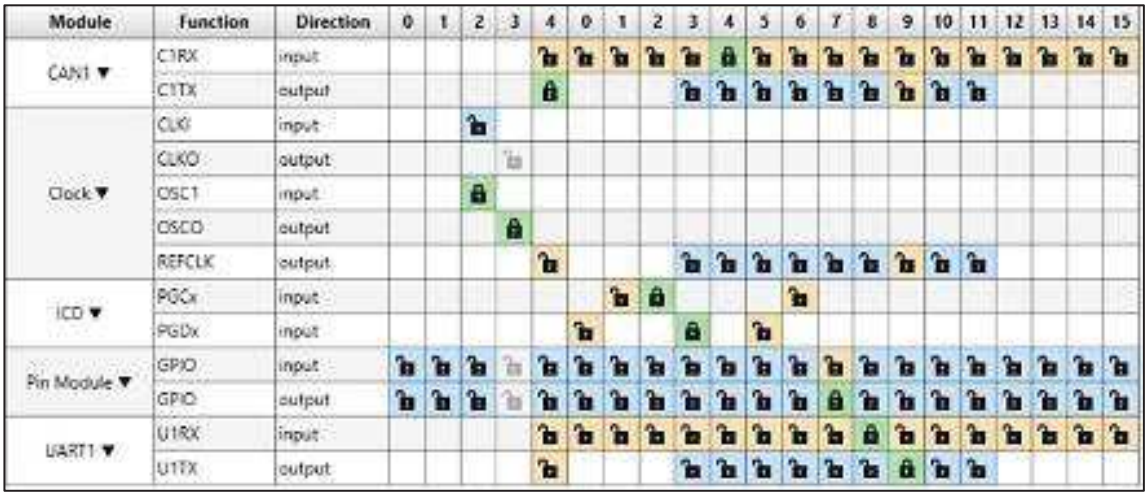
Fonte: Autoria própria

Utilizando essas configurações previstas em tabela, a partir de um cristal oscilador da placa de desenvolvimento ligado ao controlador de 8 MHz, foi possível então efetuar uma velocidade de transmissão do barramento idêntica com a do veículo de 500 kbit/s.

3.8 CONFIGURAÇÃO DO CONTROLADOR

Através do *plugin* MCC é possível configurar os pinos do controlador do sistema que ficaram como na Figura 33.

Figura 33 – Configuração do controlador através do MCC.



Fonte: Autoria própria.

No módulo CAN1 são configurados os pinos de transmissão e recepção do barramento que desempenham o papel de enviar o pacote de dados para a ECU, e receber o mesmo de resposta, com os parâmetros alterados. A mensagem é enviada e recebida através de um *buffer*.

3.9 SERVIÇO DE DIAGNÓSTICO

A norma SAE J1979, engloba vários modos de operação para leitura e escrita nas unidades centrais de controle do veículo, o Quadro 5 é mostrado os serviços que podem ser utilizados nessa norma, através da ferramenta.

Quadro 5 – Serviços disponibilizados pela norma SAE J1979.

Serviço	Descrição
$01	Faz a requisição do valor atual de dados da ECU de *Powertrain*.
$02	Faz a requisição do valor congelado pela ECU de *Powertrain*
$03	Faz a requisição de DTC's
$04	Reinicia as informações de falhas presentes na rede
$05	Faz a requisição dos valores de monitoramento do sensor de oxigênio
$06	Faz a requisição de monitoramento de bordo para testes
$07	Faz a requisição de de DTC's ocorridos durante a viagem atual ou passada
$08	Faz a requisição de controle de um sistema de bordo, teste ou componente
$09	Faz a requisição de informações do veículo
$0A	Faz a requisição de DTC's permanentes depois de uma limpeza geral

Fonte: Adaptado de SAE J1979.

Existem várias opções de modos de leitura de dados, e para cada tipo de requisição existe um serviço diferente, os fabricantes de veículos não são obrigados a suportar todos os serviços. Nesse trabalho foi utilizado o serviço 01, pois se mostrou o mais eficiente para obtenção dos dados disponibilizados na OBD2, já que ela trabalha com os dados atuais de leitura na ECU.

A norma SAE J1979 prevê para este modo, requisições de PID's (Parameter Identifiquer), para disponibilização dos dados para este serviço. Da mesma maneira a interpretação da resposta também sugerida pela norma e pode variar de acordo com o dado solicitado, como é mostrado na Quadro 2.

Quadro 6 – PIDs padrões na norma SAE J1979.

PID	Descrição	Unidade	Interpretação
$04	Carga do motor calculada	%	$\frac{100}{255} * A$
$05	Temperatura do Líquido de Arrefecimento	ºC	$A - 40$
$0C	Rotação do Motor	RPM	$\frac{256 * A + B}{4}$
$0D	Velocidade do Veículo	Km/h	A
$1F	Tempo desde o acionamento do motor	s	$256 * A + B$
$2F	Nível de Combustível	%	$\frac{100}{255} * A$
$42	Tensão da Bateria	V	$\frac{(256 * A + B)}{1000}$
$46	Temperatura Ambiente	ºC	$A - 40$

Fonte: Adaptado de SAE J1979.

Quando é feita a requisição de um PID utilizando o serviço 1, a ECU do veículo retorna uma mensagem CAN de dados com DLC de 8 bytes nomeados de

A a E, onde cada byte carrega toda a informação requisitada ou apenas uma parte dela. É importante, antes de fazer qualquer tipo de requisição, descobrir qual informação deseja solicitar, e se o PID em questão é compatível com o veículo.

3.9.1 REQUISIÇÃO DE UM PID ATRAVÉS DE UM SERVIÇO

Antes de começar a ler os dados da OBD2, é necessário entender os conceitos da estrutura de mensagem que devem seguir um padrão bem específico para que a ECU os reconheça. O valor de 7DF no identificador para CAN 1.0, é uma mensagem de broadcast para ECU, no qual ela entende que o usuário quer fazer o acesso dos dados.

A requisição de um PID através de um serviço deve ser feita por meio de uma mensagem de dados CAN com DLC de 8 *bytes* (ISO 15765) estruturada conforme sugere a Figura 34. É importante entender que todo o campo de dados da mensagem deve ser preenchido, mesmo que com zeros, para que se respeite a condição do DLC.

Figura 34 – Estrutura da mensagem de requisição de Diagnóstico.

Identificador	DLC	Dados							
		-	-	-	A	B	C	D	E
$7DF	8	Numero de Bytes de dados	Serviço	PID	$00	$00	$00	$00	$00

Fonte: Adaptado de ISO 15765.

O campo “Número de Bytes de Dados” deve ser preenchido conforme a quantidade de bytes que procedem na requisição. Sendo assim, para uma mensagem de requisição de um PID qualquer utilizando o serviço $01, este campo deve ser preenchido com 2, o que indica que a ECU deverá ler os próximos dois *bytes,* que indicam o serviço e o PID requisitado, e descartar o resto.

3.9.2 RESPOSTA A UMA REQUISIÇÃO

Uma vez enviada a requisição ao barramento, é necessário aguardar a resposta antes de enviar uma nova. Dessa forma, a ECU retornará uma mensagem de dados também com um DLC de 8 bytes porém de identificador diferente, que no mesmo caso da ECU do *powertrain* conforme a norma ISO 15765, deve ser

$7E8, estruturada de maneira semelhante à requisição, conforme mostra a Figura 35.

Figura 35 – Estrutura da mensagem de resposta de Diagnóstico.

Identificador	DLC	Dados							
		-	-	-	A	B	C	D	E
$7E8	8	Numero de Bytes de dados	Serviço + $40	PID	?	?	?	?	?

Fonte: Adaptado de ISO 15765.

Nesse caso, o veículo retorna a resposta ao PID solicitado através do barramento e preenche automaticamente todos os campos. Um detalhe importante é que o serviço solicitado retorna acrescido de 40 unidades hexadecimais e, como dependendo do parâmetro podem ser enviados mais ou menos dados, o número de bytes de dados deve ser diferente do enviado, pois além do serviço e do PID, a resposta vem seguida pelos bytes de A a E.

Nesse trabalho foi possível fazer a leitura de 4 variáveis dentro da OBD2, que serão detalhadas junto como cálculo para obtenção do valor.

3.9.3 VELOCIDADE

A velocidade do veículo é obtido através do PID $0D junto ao serviço $01. Essa requisição retorna um byte que dispensa codificação, e retorna a variável contida dentro do byte A, logo pode variar de 0 a 255 Km/h.

3.9.4 TEMPERATURA DE ARREFECIMENTO

A temperatura do líquido de arrefecimento é obtida através do PID $05 junto ao serviço $01. Mas há uma fórmula na norma SAE J1979 para fazer essa decodificação que é calculado através da equação (A-40).

3.9.5 ROTAÇÃO DO MOTOR

A rotação do motor pode ser obtido através do PID $0C com o serviço $01. Essa requisição retorna dois bytes referentes ao valor da variável, que pode ser

decodificada através da equação ((A.256+B)/4), sendo A o byte mais significativo e B o byte menos significado.

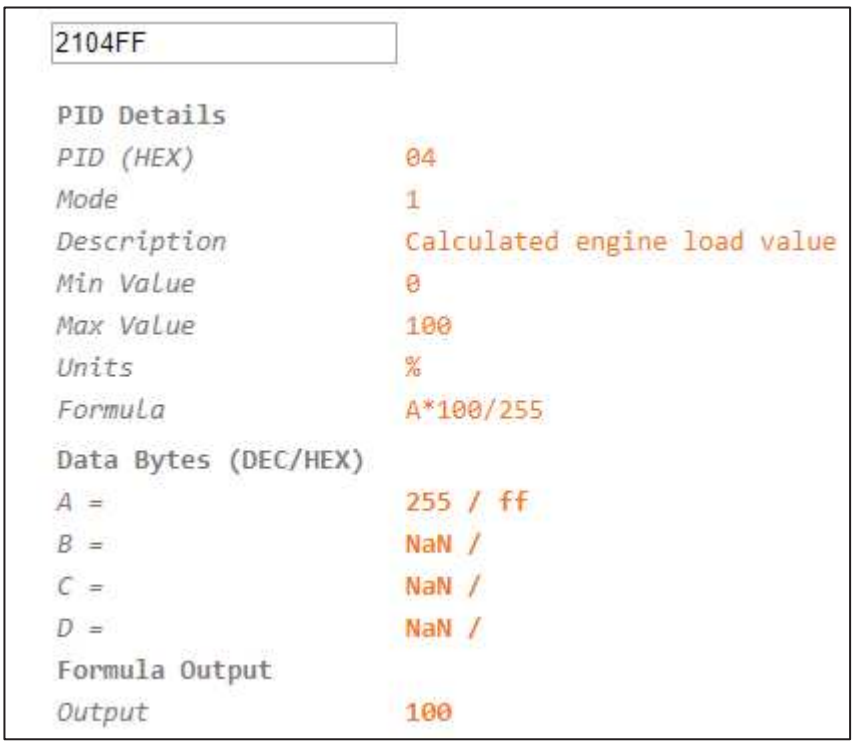
3.9.6 CARGA DO MOTOR CALCULADA

Essa requisição pode ser feita através do PID $04 junto ao serviço $01, e retorna o valor percentual calculado da carga do motor do veículo. Essa informação pode ser decodificada através da fórmula (100/255).A.

3.10 SIMULADOR DE DADOS DE RESPOSTA DA ECU

Para compreender ainda mais as funções que seriam implementadas nos pacotes e enviadas à ECU, todas foram testadas em um simulador online CSS Eletronics que fazia o cálculo da variável com os parâmetros de retorno, esses valores são genéricos, como pode-se observar na Figura 36.

Figura 36 – Simulando valores de resposta da requisição.



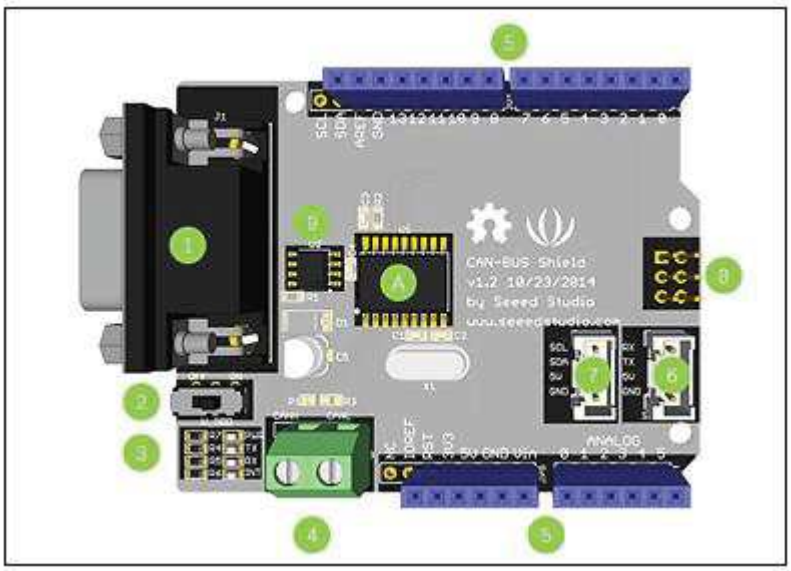
Fonte: Autoria própria.

Observando os valores genéricos para análise, a variável A de carga do motor depende apenas de uma variável de resposta. Já que o valor de carga varia de 0 a 100%, se for colocado o maior valor em hexadecimal de dois bytes para a variável de retorno no cálculo FF, quer dizer que a carga do valor está em seu valor máximo de 100% convertido em decimal, como pode ser observado na Figura 37.

3.11 TESTE DE COMUNICAÇÃO COM A PLACA CAN-BUS SHIELD 1.2V

Como já explicado, para acessar a informação da ECU do veículo é necessário uma solicitação por meio de um envio de mensagem com um pacote de informações. Para testar o envio desses pacotes, foi enviado um pacote de dados pelo pinos CAN H e CAL L da placa Eletronic Dev para os pinos CAN H e CAN L da placa CAN BUS SHIELD 1.2, que também é uma placa de desenvolvimento voltada para aplicações de rede CAN. É possível observar na Figura 37 dessa placa e seus respectivos componentes com sua devida numeração.

Figura 37 – Placa CAN-BUS Shield 1.2.



Fonte: Adaptado de SEEEK (2015).

1. **Conector DB9** - para conectar a interface OBD 2 através de um cabo DB9/OBD.
2. **V_OBD** - Obtém energia da interface OBD 2 (do DB9)
3. **LEDs Indicadores**:
   - **PWR** : Ligado
   - **TX** : Pisca quando os dados estão enviando

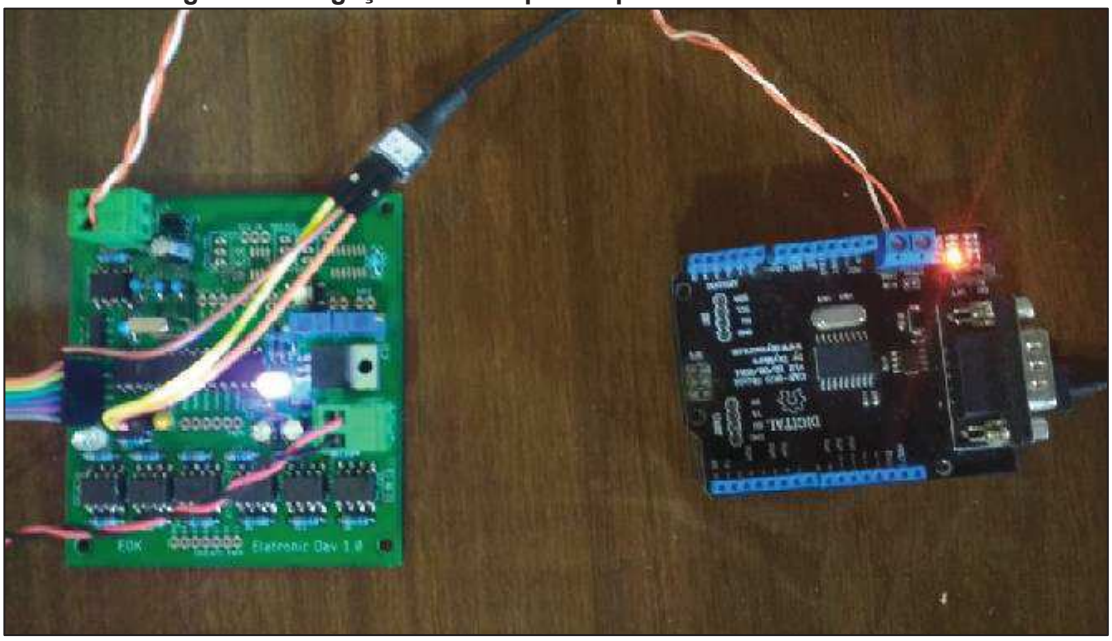
- **RX** : Pisca quando houver dados sendo recebidos
- **INT** : Interrupção de dados

4. **Terminal** – CAN H e CAN L
5. **Conectores do Shield Arduino** - MCP2551, um transceptor CAN de alta velocidade
6. **Conectores do Shield Arduino** - MCP2515, controlador CAN autónomo com interface SPI

Esta placa desenvolvida pela Arduino é voltada para aplicações com o protocolo CAN, de leitura e escrita. Ela é composta por um controlador MCP2515 que utiliza o protocolo SPI e um transceptor MCP2551 para fornecer os níveis de tensões ao longo do barramento.

A ligação entre as placas é feita de maneira simples, como pode ser observado na Figura 38.

Figura 38 – Ligação entre as placas para teste de transmissão.



Fonte: Autoria própria.

O pacote de informações enviadas para o CAN-Bus Shield, é idêntico ao que deve ser enviado a ECU do veículo e deve ser sincronizada com a mesma velocidade do sistema. Como a placa não entende que é uma aplicação especifica da norma 15765-4 ela não retorna uma resposta, mas recebe os dados transmitidos.

Esse pacote enviado a partir da placa Eletronic Dev, foi lido pela placa CAN-Bus Shield a partir de uma função *read*, disponibilizada na biblioteca “Canbus.c” e os

valores foram exibidas na serial da IDE do Arduino, como podem ser observados nas Figuras 39 e 40.

Figura 39 – Transmissão do pacote de mensagem.

```
ECAN1_ReceiveEnable();
UART1_Initialize();

MENSAGEM.frame.id = 0x000007E8;
MENSAGEM.frame.idType = CAN1_FRAME_STD;
MENSAGEM.frame.msgtype = CAN1_MSG_DATA;
MENSAGEM.frame.dlc = 0x09;
MENSAGEM.frame.data0 = 0x08;
MENSAGEM.frame.data1 = 0x01;
MENSAGEM.frame.data2 = 0x0c;
MENSAGEM.frame.data3 = 0x01;
MENSAGEM.frame.data4 = 0x01;
MENSAGEM.frame.data5 = 0x01;
MENSAGEM.frame.data6 = 0x00;
MENSAGEM.frame.data7 = 0x00;

TESTE = ECAN1_transmit(ECAN1_PRIORITY_HIGH,&MENSAGEM);

LED_SetLow();
```

Fonte: Autoria própria.

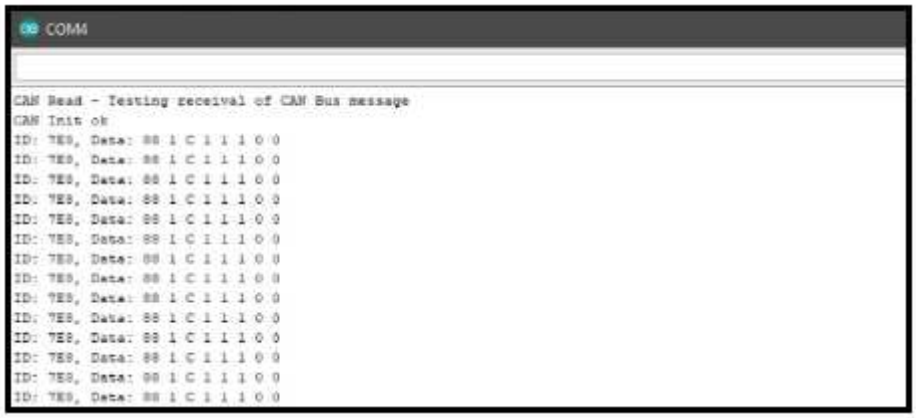
Figura 40 – Recepção do pacote de mensagem.

```
void loop(){

  tCAN message;
if (mcp2515_check_message())
  {
    if (mcp2515_get_message(&message))
  {
        |

              Serial.print("ID: ");
              Serial.print(message.id,HEX);
              Serial.print(", ");
              Serial.print("Data: ");
              Serial.print(message.header.length,DEC);
              for(int i=0;i<message.header.length;i++)
               {
                 Serial.print(message.data[i],HEX);
                 Serial.print(" ");
               }
              Serial.println("");
            //}
          }}

}
```

Fonte: Autoria própria.

O pacote de informações enviadas para a CAN-BUS, foi recebido com êxito, e exibido na serial da IDE do Arduino, como pode se observar na Figura 41.

Figura 41 – Resultado da recepção.



Fonte: Autoria própria.

Observa-se que na Figura 41 foi obtido êxito nessa aplicação, mas nesse caso foi enviado apenas uma mensagem, enquanto que no caso da ECU do veículo deverá ser enviada 4 mensagens pois são 4 parâmetros.

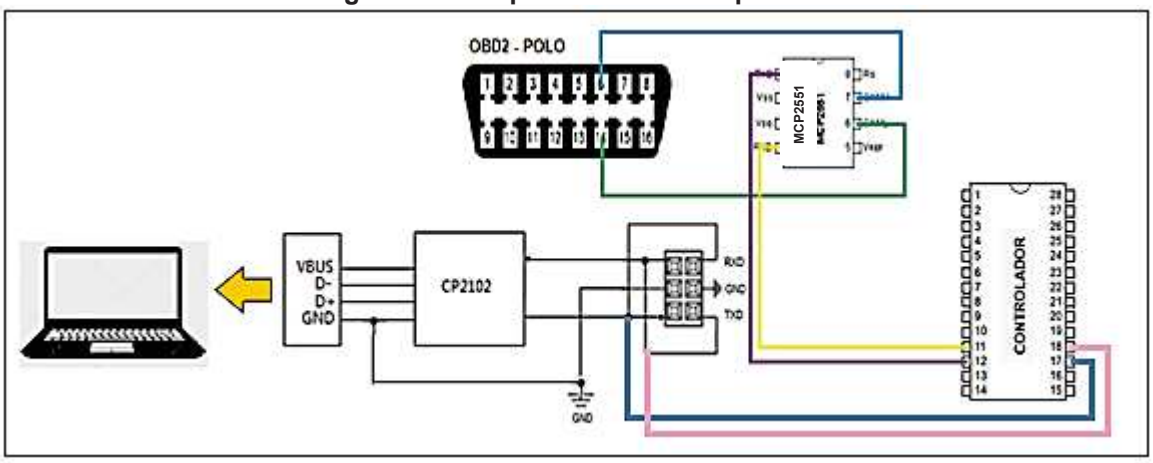
A ECU deverá retornar 4 respostas diferentes, uma vez que se sabe através do PID qual informação foi transmitida e através do registrador, é possível tratar os dados de retorno e exibir os mesmos para o usuário.

4 RESULTADOS

4.1 ESQUEMA FINAL DO DISPOSITIVO

A Figura 42 apresenta os dispostos e os componentes do projeto.

Figura 42 – Esquema final do dispositivo.



Fonte: Autoria própria.

A alimentação do sistema é derivada do pino 16 da OBD do veículo, já que esse pino corresponde a tensão da bateria de aproximadamente 14 V, e o terra da placa é derivado do pino 4 ou 5. O dispositivo UC-1000 é conectado a um computador, pois os valores interpretados são mostrados ao usuário na plataforma MPLAB.

4.2 MAPEAMENTO PARA COLETA DE DADOS DO VEÍCULO

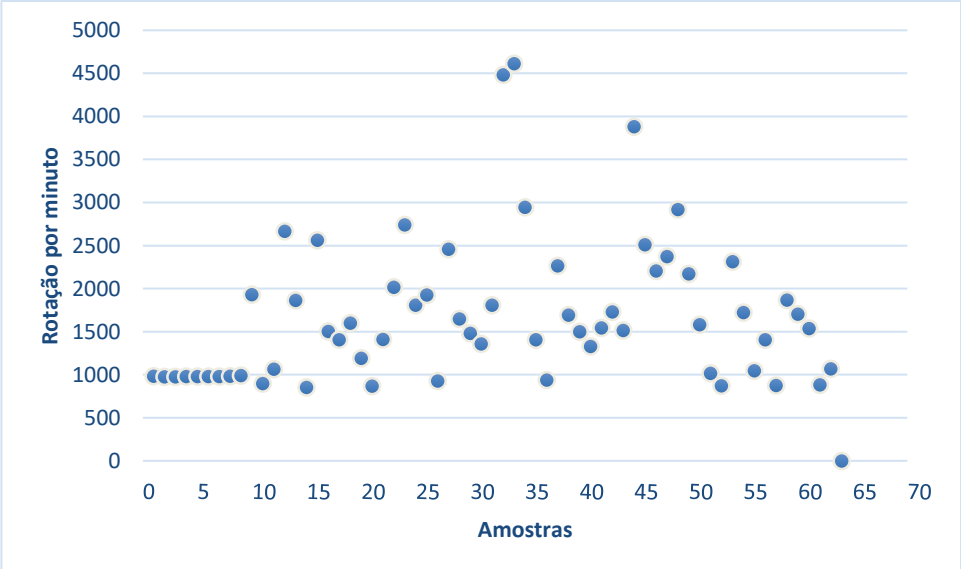
No teste foi realizado o mapeamento de um trajeto de aproximadamente 2 km/h, nesse trajeto foram feitas 4 leituras de variável da ECU no veículo, o trajeto pode ser visto na Figura 43.

Figura 43 – Trajeto mapeado.

Fonte: Autoria própria.

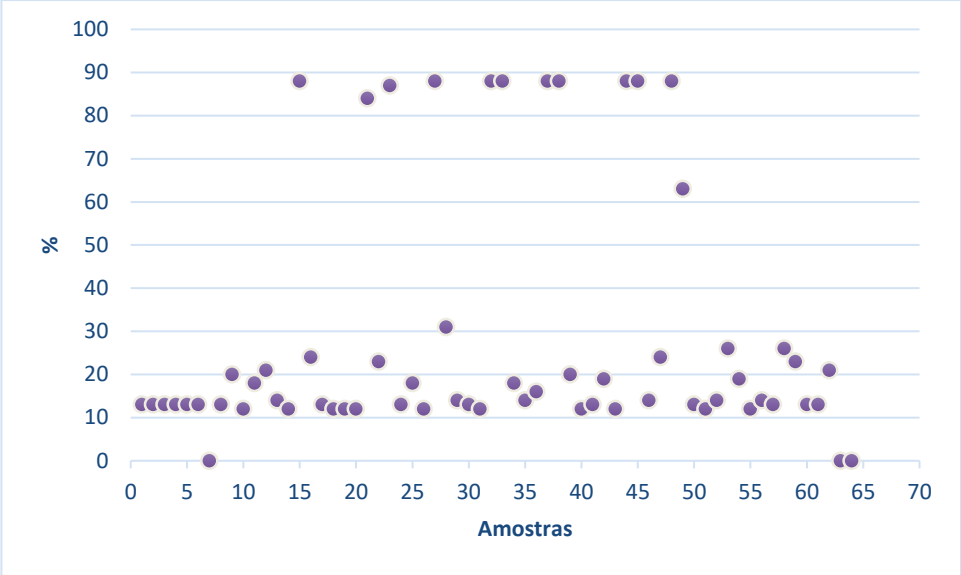
No percurso da figura 43, foi possível fazer a leitura de 64 valores para cada parâmetro. Os valores foram salvos em um documento de *world*, e as Figuras 44, 45, 46 e 47 mostram os gráficos que foram gerados para cada um deles, em função do número das amostras.

Figura 44 – Gráfico da variável RPM.



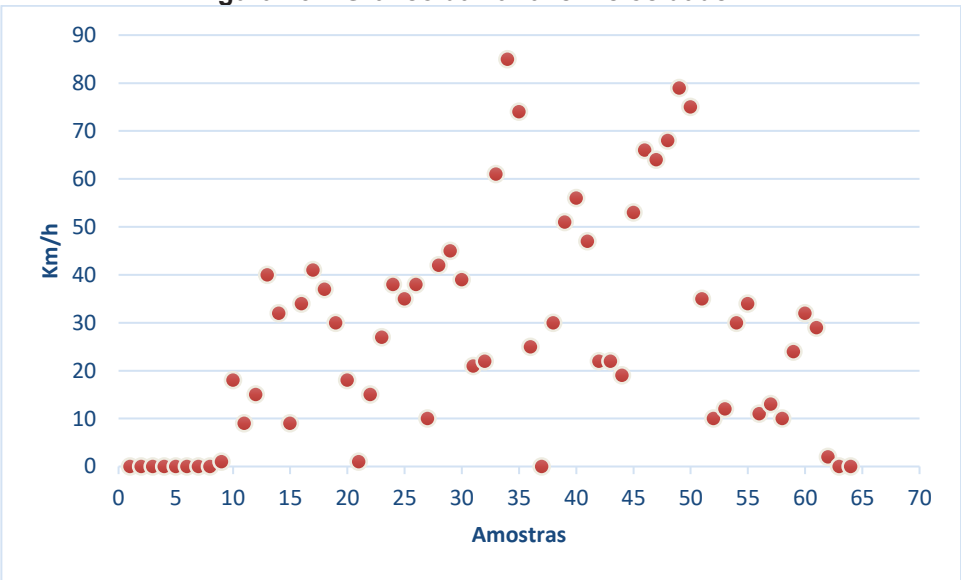
Fonte: Autoria própria.

Figura 45 – Gráfico da variável Acelerador.



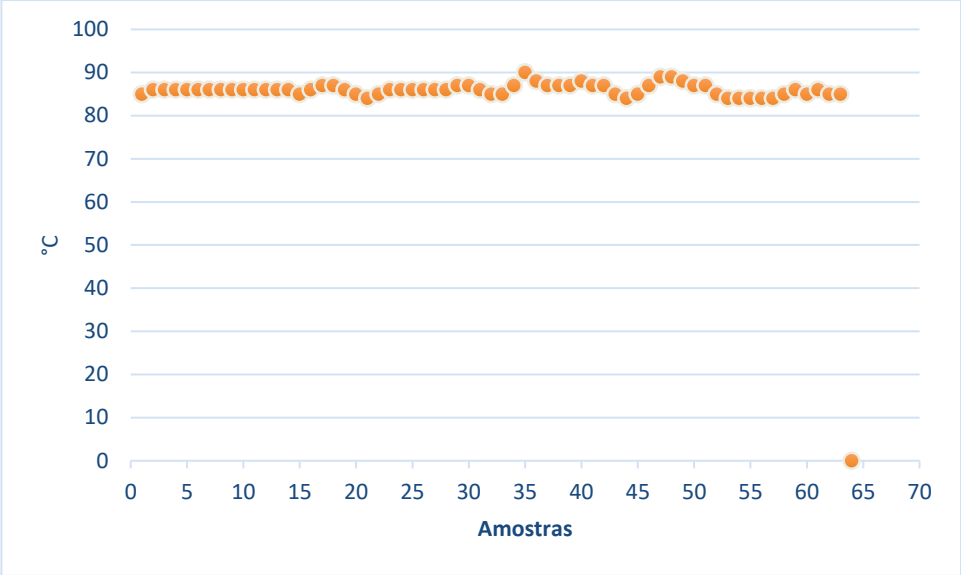
Fonte: Autoria própria.

Figura 46 – Gráfico da variável Velocidade.



Fonte: Autoria própria.

Figura 47 – Gráfico da variável Temperatura.

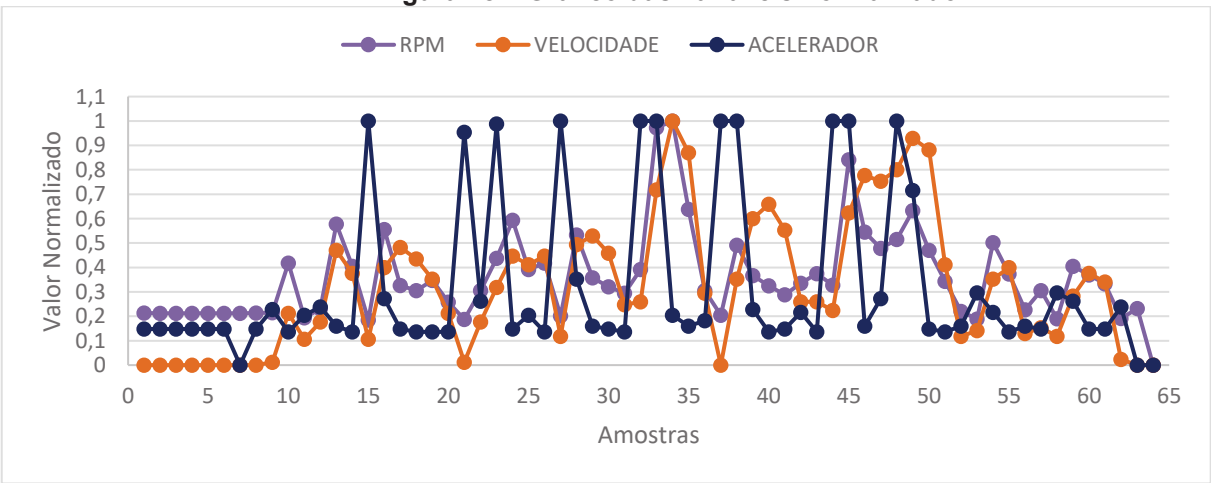


Fonte: Autoria própria.

Os dados foram lidos com êxito, e cada leitura foi realizado em um tempo de 600 ms, esse valor foi escolhido para que os gráficos pudessem ser visto de maneira nítida em função das amostras, esse valor de leitura pode ser diminuído para até 100 ms, para uma maior precisão de leitura.

Se observar os gráficos atentamente é possível perceber que os parâmetros de RPM e aceleração são proporcionais, mas dependem do volume da marcha engatada, a variável velocidade depende dessas duas, mas o carro precisa estar em movimento. Confeccionando um gráfico normalizado dessas 3 variáveis, como na Figura abaixo, é possível vem como essas variáveis estão diretamente ligadas.

Figura 48 – Gráfico das variáveis normalizado.



Fonte: Autoria própria.

O parâmetro temperatura mostra a temperatura do líquido de arrefecimento do motor que varia pouco, pois o trajeto é relativamente curto. Confeccionando um gráfico normalizado. A ideia inicial era ler 6 parâmetros do veículo, infelizmente os parâmetros de tensão da bateria e a taxa de fluxo de ar, não foram possíveis de serem realizadas suas respectivas leituras, pois nesse veículo, no qual foram realizados os testes, os PIDs não seguiam as normas padrão, e os valores do PIDs para o ideal funcionamento, não são disponibilizados pelo fabricante.

Para assegurar o funcionamento correto desses parâmetros, durante esse teste todos os valores foram comparados com os disponíveis do painel de bordo do veículo, seguindo o mesmo comportamento. Foi constatado que esse dispositivo funcionou de maneira eficiente para leitura de dados.

5 CONCLUSÃO

Os objetivos de trabalho foram alcançados com êxito, as leituras de parâmetros dos veículo permitem fazer uma análise minuciosa sobre o seu funcionando, proporcionando que o usuário a faça de maneira simples.

Nesse trabalho foram lidos informações simples, que seguem valores padrões dentro da norma específica, mas com alterações simples nos PIDs é possível ler outras informações, inclusive informações mais especificas, que não mostradas no painel de controle.

Em relação ao mapeamento do circuito, caso fosse necessário fazer uma análise mais robusta em relação ao veículo e o trajeto percorrido, seria necessário, um módulo GPS para fazer a análise, pois abrangeriam mais variáveis, para aumentar a precisão, como tempo, latitude e longitude.

Algumas informações de leitura de dados, ainda são muito restritas, pois não são disponibilizadas pelo fabricante, tendo que fazer um estudo mais aprofundado dependendo o tipo de aplicação a ser realizada.

Esse estudo foi muito importante para compreensão de uma área que vem crescendo muito nos últimos anos, não só na área automotiva, mas na área de monitoramento de plantio, com a manipulação de múltiplos sensores com apenas uma rede.

REFERÊNCIAS

BOSCH, R. **CAN Specification**, Setembro 1991.

CICLO ENGENHARIA EM INFORMAÇÃO AUTOMOTIVA. **Dica Técnica Fiat - DCT B1009 e luz do Airbag acesa**, 15 out. 2016. Disponivel em: <http://blog.ciclo.eng.br/author/blogciclo/page/3/>. Acesso em: 15 set. 2017.

GUIMARÃES, A. A.; SARAIVA, A. M. **O Protocolo CAN: Entendendo e Implementando uma Rede de Comunicação Serial de Dados baseado no Barramento "Controller Area Network"**, 2002.

HUBERT, M. K. **O protocolo CN como solução para aplicações distríbuidas, baseados em objetis entre PCs e computador**, Janeiro 2001. 12.

MARQUES, M. C. **Can Automotivo Sistemas e monitoramento**, Abril 2004. Itajubá.

MUCEVSKI, K. **Automotive CAN Bus System**, 8 dezembro 2015. Disponivel em: <https://www.linkedin.com/pulse/automotive-can-bus-system-explained-kiril-mucevski/>. Acesso em: 2 nov. 2017.

NASCIMENTO, L. M. **Protocolo de comunicação CAN e suas aplicações na indústria automobilísticas**, Junho 2006.

NUNES, T. F. **Telemetria de um veículo Baja SAE através de rede CAN**, 16 Junho 2016.

REUSS, H. C. **Extended Frame Format - A New Option of the CAN protocol**, 3 Abril 1993.

SCOPINO, P. L. **Oficina do Saber MTE-THOMSON**, 14 de Julho 2017. Disponivel em: <http://cursosonline.mte-thomson.com.br/curso/curso-rede-can/>. Acesso em: 22 ago 2017.

SINDIREPA-MT. **Diagnóstico de falhas no sistema de comunicação entre módulos através do conector DLC**, 29 Agosto 2017. Disponivel em: <http://www.sindicatodaindustria.com.br/noticias/2017/08/72,115830/diagnostico-de-falhas-no-sistema-de-comunicacao-entre-modulos-atraves-do-conector-dlc.html>. Acesso em: 2 em nov. 2017.

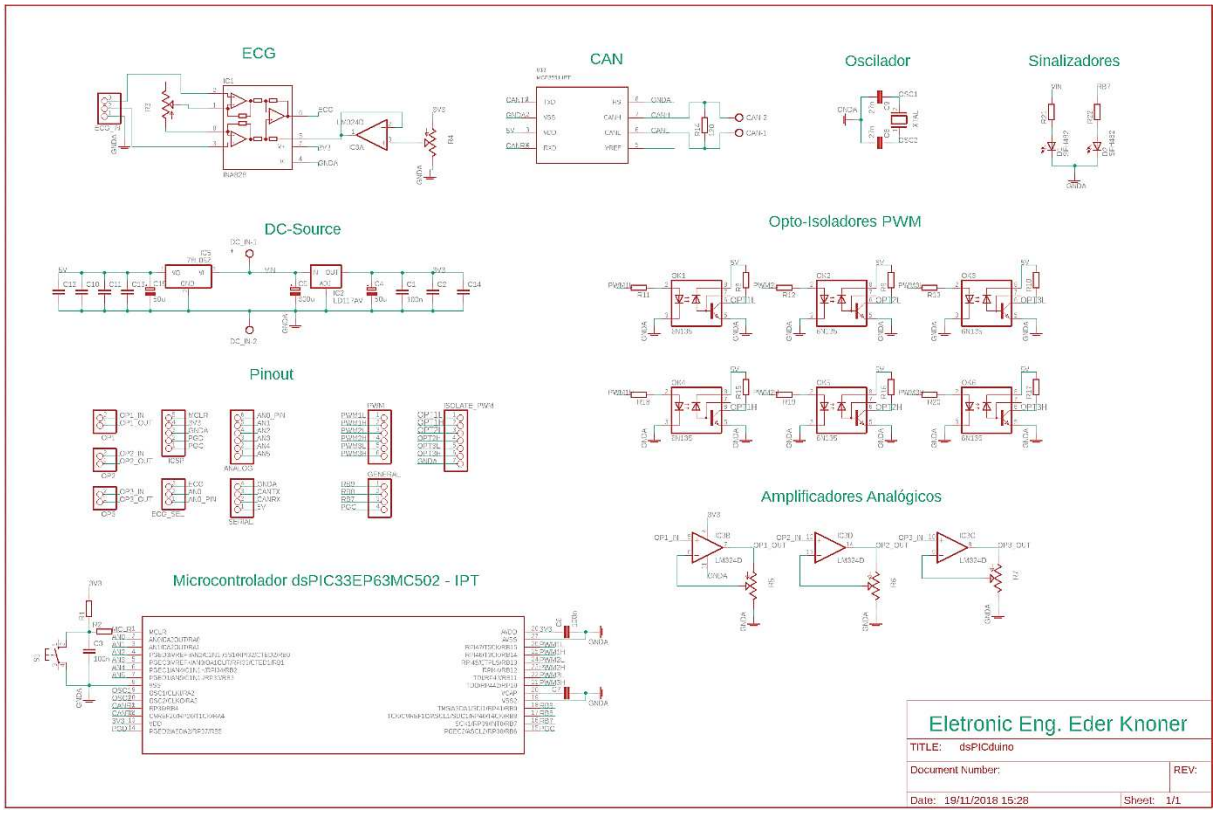
PETROCENTER. **Equipamentos estáticos e dinâmicos**, Setembto 2012. Disponivel em: <https://lcsimei.files.wordpress.com/2012/08/equipamentos-estc3a1ticos-e-dinc3a2micos_simei3.pdf>. Acesso em: 20 set. 2017.

CIA. **History of Can technology**, 2013 Disponível em: <https://www.can-cia.org/can-knowledge/can/can-history/>. Acesso em: 4 set. 2017.
SOUZA, D. T. **Sistemas Automotivos Embarcados**, Novembro 2008.

MICROCHIP. (2010). MCP2551.**CAN Tranceiver**. Atlanta, EUA.

ANEXOS

ANEXO A – ESQUEMÁTICO DA PLACA ELETRONIC DEV 1.0



ANEXO B – SIMULAÇÃO DE DADOS DA RESPOSTA DA ECU

210d65		210580	
PID Details		**PID Details**	
PID (HEX)	0D	*PID (HEX)*	05
Mode	1	*Mode*	1
Description	Vehicle speed	*Description*	Engine coolant temperature
Min Value	0	*Min Value*	-40
Max Value	255	*Max Value*	215
Units	km/h	*Units*	C
Formula	A	*Formula*	A - 40
Data Bytes (DEC/HEX)		**Data Bytes (DEC/HEX)**	
A =	101 / 65	*A =*	128 / 80
B =	NaN /	*B =*	NaN /
C =	NaN /	*C =*	NaN /
D =	NaN /	*D =*	NaN /
Formula Output		**Formula Output**	
Output	101	*Output*	88

210c3030		2104FF	
PID Details		**PID Details**	
PID (HEX)	0C	*PID (HEX)*	04
Mode	1	*Mode*	1
Description	Engine RPM	*Description*	Calculated engine load value
Min Value	0	*Min Value*	0
Max Value	16,385.75	*Max Value*	100
Units	RPM	*Units*	%
Formula	((A*256)+B)/4	*Formula*	A*100/255
Data Bytes (DEC/HEX)		**Data Bytes (DEC/HEX)**	
A =	48 / 30	*A =*	255 / ff
B =	48 / 30	*B =*	NaN /
C =	NaN /	*C =*	NaN /
D =	NaN /	*D =*	NaN /
Formula Output		**Formula Output**	
Output	3084	*Output*	100

ANEXO C – ESQUEMÁTICO DA PLACA CAN-BUS SHIELD 1.2