DESENVOLVIMENTO DE UM GATEWAY BASEADO NO PROTOCOLO CAN PARA INTEGRAÇÃO E DIAGNÓSTICOS DE PERIFÉRICOS

Natanael F. G. Vitorino, Gelson José de A. Filho, Herick Yves Ribeiro, Lester de Abreu Faria e Marc Gonzalez Capdevilla[1]
Brain, Universidade Facens Sorocaba, São Paulo, Brasil
[1]Prof. Dr

RESUMO

A requisição e leitura de dados oriundos dos diversos módulos eletrônicos veiculares fazem parte de um eixo importe para o desenvolvimento de projetos embarcados. No presente artigo é descrito o desenvolvimento de um *Gateway* baseado no protocolo *CAN*, capaz de realizar a escrita, leitura e o processamento de dados provenientes da *Electronic Control Unit* (*ECU)* do radar *Off-Highway* (*OHW)*. A aplicação conta com suporte ao protocolo *UDS on CAN*, possibilitando a obtenção de diagnósticos por meio de leituras, utilizando serviços *UDS*. O Acesso ao *CAN Bus* é alcançado por meio do módulo MCP2515 que, por sua vez, realiza a leitura dos pulsos elétricos e os converte em dados numéricos. Os resultados obtidos por meio da utilização do Gateway foram validados de acordo com as padronizações, mostrando a usabilidade do *Node* desenvolvido.

PALAVRAS-CHAVE

CAN, UDS, OHW, Gateway, Node, BUS

1. INTRODUÇÃO

Cada vez mais se desenvolvem métodos, eficientes e tecnológicos, a fim de suportar o fluxo de dados provenientes de atuadores, sensores e módulos presentes em um veículo. Tais métodos tornaram-se um eixo central no desenvolvimento veicular e, com o objetivo de garantir tanto o controle quanto o bom funcionamento de todos os componentes, principalmente aqueles ligados à injeção eletrônica, foi desenvolvido um protocolo chamado *Controller Area Network (CAN*), o qual se destaca pela robustez e velocidade de transmissão de dados, possuindo uma taxa de transferência de dados de até 1 Mbps (STEVE CORRIGAN *et al*. 2002).

Ao analisar os dados técnicos do protocolo *CAN*, é notório observar a sua distinção dos demais protocolos de transmissão de dados, como o *USB* e *Ethernet*, os quais realizam o envio de grandes blocos de dados de uma só vez (Steve Corrigan *et al*. 2002). Este envio de dados é também possível pelo *CAN*, porém restrito a 8 *Bytes*, uma vez que este é o tamanho máximo de dados que uma mensagem pode possuir e ser transmitida pelo barramento. Assim, quando for necessário o envio de dados superior à quantidade máxima contida em uma única mensagem, é fundamental utilizar o protocolo de transporte chamado *UDS*.

Nesse contexto, e com base em projetos de segurança veicular baseados no radar *OHW* (*Radar Off Highway* desenvolvido pela Bosch), tornou-se imperativa a pesquisa e implementações com o protocolo *CAN*, a fim de integrar os periféricos e garantir o seu adequado funcionamento. Assim, no presente artigo, será descrita uma série de aplicações do protocolo *CAN* para o radar anteriormente mencionado, o qual possui dois canais de comunicação, nos quais os métodos de leitura e escrita são implementados. Este radar é responsável por aferir dados do ambiente, como velocidade do objeto em diferentes ângulos, *RCS* (*Radar Cross Detection*, valor numérico que refere-se ao quão detectável um objeto é), movimento e proximidade do alvo em questão. Tais dados podem ser transmitidos por meio de outros protocolos, entretanto, por conta das vantagens anteriormente mencionadas, o protocolo *CAN* foi selecionado para o estudo e desenvolvimento da aplicação deste projeto.

2. REVISÃO LITERÁRIA

As aplicações da comunicação (protocolo) *CAN* variam muito nas diversas áreas de desenvolvimento tecnológico, desde as áreas automobilísticas, nas linhas de produções industriais, até o uso militar. Desenvolvido pela Bosch, em parceria com a Intel, e introduzido no ano de 1986 durante um congresso da *SAE* (*Society of Automotive Engineers*), a comunicação *CAN* apresentou-se como uma solução ímpar para a indústria automobilística. Um dos seus principais objetivos era o de eliminar os antigos e complexos chicotes elétricos dos veículos (fios que conectam os módulos eletrônicos de controle e periféricos do veículo).

Enviar dados pelo barramento é uma tarefa complexa, mediante a qual a *International Standardization Organization (ISO*) definiu padronizações corretas e revisadas para as implementações do protocolo CAN, o qual é dividido em seis partes, descrevendo a maneira correta de se trabalhar (*ISO* 11898). A estrutura da mensagem é composta por diversos campos, onde cada um tem a sua importância para o processamento da mensagem, tanto em *hardware* quanto em *software*, uma vez que possui métodos para arbitragem e identificação de erros (Steve Corrigan *et al.* 2002).

Dentre as características que o radar possui, destacam-se as funcionalidades presentes em suas configurações de leitura e escrita, as quais são acessadas por meio do barramento *(CAN Bus)*. Estas funções são acessadas por meio do protocolo *Unified Diagnostic Services (UDS)*, o qual utiliza o protocolo *CAN* como camada de transporte, empregando os mesmos atributos de estruturação das mensagens. O principal uso do protocolo *UDS* se dá para serviços de diagnóstico *(ISO 14229)*, de diferentes tipos de módulos embarcados no veículo automotivo.

2.1 *Controlled Area Network (CAN)*

Desenvolvido pela Robert Bosch GmbH juntamente com a Intel, a comunicação *CAN* apresenta como principal objetivo a redução da quantidade de fios que conectam um módulo ao outro (Figura 1), reduzindo assim, o custo para o desenvolvimento de redes de comunicação embarcadas veiculares *(C. A. N. ESPECIFICATION)*. O protocolo *CAN* foi padronizado especificamente para comunicação veicular, proporcionando a leitura de parâmetros provenientes de módulos presentes em um veículo, possibilitando analisar, identificar diagnósticos, solucionar erros com eficiência, segurança e velocidade.

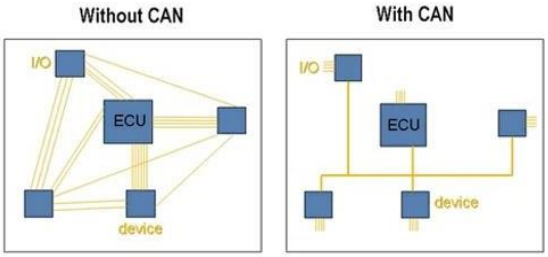

Figura 1. Redução de fios elétricos entre os módulos

Existem duas formas para estruturar as mensagens *CAN*, quais sejam: a mensagem padrão *(Standard)* e a estendida *(Extended)*. Estes dois modos possuem características que os diferem entre si, dentre as quais a principal é o tamanho do *ID* (Identificador). As mensagens *CAN* são chamadas de *Frames* (quadros) e possuem *Fields* (Campos), com responsabilidades díspares. Além de possuírem as padronizações, também possuem quatro tipos diferentes de mensagens, as quais podem ser aplicadas para propósitos diferentes.

Os tipos são:

- *Data Frame* – Realiza a transmissão de dados, sendo uma das mensagens mais comuns;
- *Remote Frame* – Faz a requisição das mensagens diretamente de um módulo específico;
- *Error Frame* – Reporta um erro ao transmissor, requisitando o envio da mensagem novamente; e
- *Overload frame* – Gera um intervalo entre o envio das mensagens.

Estes frames são estruturados de acordo com os padrões *Standard* ou *Extended* e transmitidos ao *CAN Bus* por meio de microcontroladores contidos internamente dos módulos *(ECU)*, realizando, assim, leituras e requisições de dados. A mensagem no padrão *Standard* (Figura 2), possui 11 *bits* para o campo *ID*,

proporcionando assim o envio de até 2048 mensagens com endereços diferentes. Por outro lado, a mensagem *Extended* (Figura 3), possui 29 *bits* no mesmo campo citado anteriormente, possibilitando assim o envio de até 537 milhões de mensagens com identificadores dissímeis uns dos outros. Esta é a principal característica que distingue os dois padrões de *Frames* do protocolo.

O principal ponto do protocolo *CAN*, e o mais forte, é a forma como se aplicam as suas arbitrações, as quais viabilizam a verificação e identificação de erros nos dados recebidos ou enviados de um *Node* a outro. A sua arbitragem ocorre desde a parte eletrônica até as mais variadas lógicas binárias aplicadas ao algoritmo para a leitura e processamento das mensagens. A arbitragem é um dos pontos que influenciam o protocolo a alcançar a sua robustez.

2.1.1 *Standard Message*

Possuindo um identificador de 11 *Bits*, a *Standard Frame* possibilita o envio de no máximo 2048 mensagens com endereços distintos. Para saber a quantidade de mensagens possíveis de serem enviadas pelos padrões de mensagens, basta elevar a base binária pelo número de *bits* do campo. Este padrão pode ser utilizado em pequenos projetos nos quais possuem poucos *Nodes* conectados ao *CAN Bus*, entretanto não se descartando a sua usabilidade para grandes aplicações em combinação com o padrão *Extended*.

O *Standard Fr*ame, muitas vezes, é encontrado em barramentos onde ambos os padrões *Extended* e *Standard* são aplicados a estruturação das mensagens transmitidas pelos *Nodes* conectados ao barramento. A estrutura da *Standard Frame* é apresentada pela Figura 2. *Bits* recessivos e dominantes compõem a mensagem *Standard*, que por consequência da lógica inversa utilizada pelo protocolo, logo o *bit* recessivo significa nível lógico alto e o *bit* dominante significa nível lógico baixo.

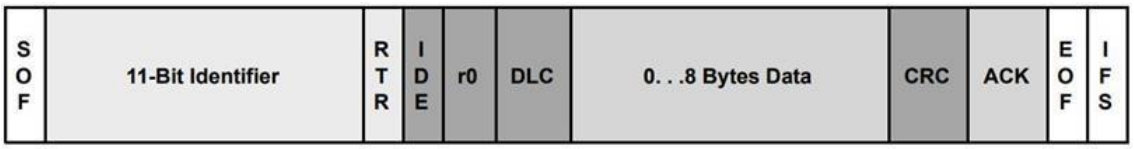

Figura 2. Estrutura do *Standard Frame*

Campos do *Standard Message*:

- *SOF (Start of Frame)* - O seu valor corresponde a um *bit* dominante que marca o início da mensagem;
- *Identifier (ID,* identificador ou endereço) – Responsável pela prioridade da mensagem;
- *RTR (Request Transmission)* - Requisição de transmissão;
- *IDE (Identifier Extension)* - *Bit* que representa a extensão do identificador, caso este *bit* seja dominante, significa que a mensagem transmitida não possui uma extensão;
- *r0* - *Bit* reservado;
- *DLC (Data Length Code)* - Responsável por armazenar o tamanho total dos dados que serão transmitidos na mensagem, possuindo 4 *bits*;
- *Data* - Dados da mensagem, possuindo 64 *bits* de armazenamento;
- *CRC (Cyclic Redundancy Check)* - Campo responsável por realizar o *Checksum* da quantidade de *bits* contidos em uma mensagem, um dos métodos de validação de erros do protocolo;
- *ACK (Acknowledgment)* - Validação da mensagem, se foi ou não transmitida corretamente;
- *EOF* (*End of Frame)* - Campo responsável por indicar o fim da mensagem; e
- *IFS (Interframe Space)* - Quantidade de tempo necessária para que o controlador mova um frame recebido corretamente para sua posição adequada em uma área de *Buffer* da mensagem.

2.1.2 *Extended Message*

Ao observar a estrutura de uma mensagem *Extended* apresentada pela Figura 3, nota-se que alguns campos foram acrescentados e o identificador recebeu 18 *bits* a mais do que o *Standard*. Assim como o padrão de estruturação citado anteriormente, o cálculo para saber a quantidade de mensagens é o mesmo, totalizando mais de 536 milhões de mensagens possíveis. Utilizado principalmente em implementações que possuem um grande número de módulos, onde é necessário uma quantidade elevada de mensagens.

Figura 3. Estrutura do *Extended Frame*.

Campos do *Extended Frame*:

- *SRR (Substitute Remote Request*) – *Bit* cuja função é a solicitação remota substitutiva, substituindo o *Bit RTR (Remote Transmission Request),* no local da *Standard Message* como um espaço reservado no formato estendido. Indica se o *Frame* é uma requisição de transmissão remota ou não;
- *IDE (Identifier Extention)* - *Bit* recessivo no identificador estendido que indica a existência de mais*Bits* de identificação (*ID Extended*); e
- *r1 (Reserved Bit)* - *Bit* reservado adcional.

2.1.3 Arbitragem

Com a responsabilidade de detectar e relatar erros nos pacotes transmitidos no *CAN Bus*, a arbitragem é uma característica de grande importância para o bom funcionamento do protocolo. A lógica utilizada pelo protocolo é inversa, ou seja, quando o nível lógico é alto, em contexto eletrônico, significa que é recessivo e se o nível lógico for baixo, significa que é dominante. Para melhor entendimento, a Figura 4 apresenta duas mensagens no barramento, nas quais ambas disputam à arbitragem, pelo endereço, e quanto menor for o endereço, maior a sua prioridade.

Na Figura 4 existem dois *Nodes*, os quais estão conectados ao *CAN Bus*: o *Node* B e o *Node* C. Ambos estão enviando mensagens ao barramento, porém a arbitragem é aplicada a cada uma das mensagens, principalmente em seus endereços, ocorrendo em baixo nível, ou seja, *bit* a *bit*. A imagem contém três linhas, duas para os *Nodes* e uma para o *CAN Bus. A*o observar a mensagem proveniente do *Node* C, nota-se que ela possui mais *bits* dominantes (neste caso, 0 ou nível lógico baixo) e se mantém no barramento. Em suma, a mensagem que tiver o menor *ID* obterá acesso prioritário ao *CAN Bus*, bem como também será a primeira a ser recebida pelo módulo receptor, uma vez que o endereço da mensagem está ligado diretamente à arbitragem de prioridade.

O protocolo possui detecção de colisão não-destrutiva, ou seja, se duas mensagens são enviadas ao mesmo *Node* no mesmo instante, elas serão obrigatoriamente arbitradas e a mensagem que tiver o menor *ID* será a prioritária. A outra mensagem não é perdida, continuando presente no *CAN Bus* até ser requisitada pelo módulo. A inversão dos valores no barramento se dá por conta da configuração eletrônica, da forma com que o barramento é construído. Na Figura 5 observa-se o funcionamento e configuração do *CAN Bus*.

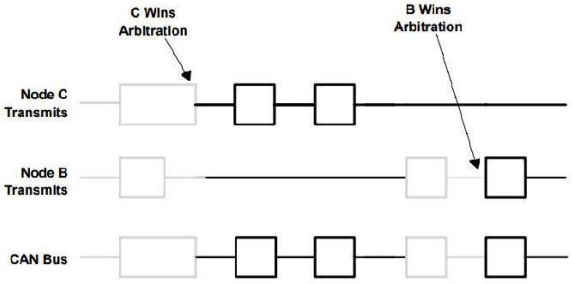

Figura 4. Funcionamento da Arbitragem

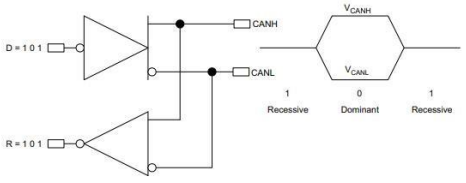

Figura 5. Bus Reverse Logic

2.1.4 *CAN Bus*

Constitui-se em um barramento no qual as mensagens transitarão de um módulo ao outro, normalmente utilizando dois fios, porém também existindo aplicações com apenas um fio. Em suas extremidades, encontra-se resistores de 120 Ohms resposáveis por assegurar o bom funcionamento do barramento, garantindo a reflexão dos pulsos elétricos corretamente pelos fios. A configuração do *CAN Bus* é apresentada pela Figura 6, a qual demonsta um dos pontos para a diminuição de chicotes elétricos veiculares entre os módulos, sensores e demais atuadores. Além da redução de chicotes, o par diferencial do barramento conforme observado na

Figura 7, é responsável por reduzir os ruídos na comunicação, deixando assim, o protocolo robusto e resistente. Nela é possível notar a modulação das ondas de pulsos elétricos e suas características, na qual é apresentada a forma como os fios do barramento se comportam e o espelhamento característico do protocolo.

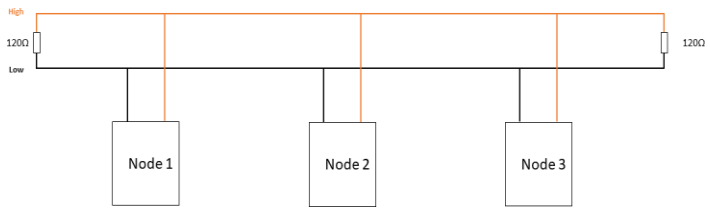

Figura 6. Barramento *CAN*

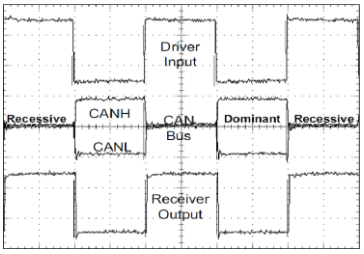

Figura 7. Mensagem *CAN* no *CAN Bus*

2.1.5 Verificação de Erros

Eficiente, preciso e robusto, o *CAN Bus* possui cinco métodos de verificação: três no nível da mensagem e dois no nível dos *bits*. Os tipos de verificação estão listados e separados em dois tópicos, a nível da mensagem e a nível dos *bits*.

Nível da mensagem:

- *CRC*, *ACK* e checagem dos *frames* (*SOF*, *EOF*, *ACK delimiter* e *CRC delimiter*, se for encontrado um *bit* dominante em campos onde sempre devem conter *bits* recessivos, um erro é gerado. Da mesma forma ocorre com os campos onde são esperados *bits* dominantes).

Nível dos *bits*:

- Cada *bit* transmitido é monitorado pelo transmissor, se o *bit* escrito no barramento for diferente do bit enviado pelo transmissor, é gerado um erro. O último é o método de detecção com a regra *Bit Stuffing*, no qual após cinco *bits* de mesmo nível lógico, se o próximo *bit* não for um complementar, é gerado um erro.

2.1.6 Características de um Módulo *CAN*

Um módulo *CAN* é caracterizado como um *Node* conectado ao *CAN Bus*, responsável por requisitar e transmitir mensagens a outros módulos conectados ao barramento. A comunicação entre os *Nodes* é estruturada com três componentes: microcontrolador, responsável pelo processamento da mensagem; *CAN Transceiver*, responsável por disponibilizar a mensagem ao *CAN Bus*; e o *CAN Controller*, responsável por realizar a interface para a aplicação dos filtros do protocolo, disponibilização das mensagens e proporcionar a manipulação delas. Estes componentes podem ser observados na Figura 8, a qual apresenta a estrutura de um módulo *CAN* e a forma como é conectado ao *CAN Bus*.

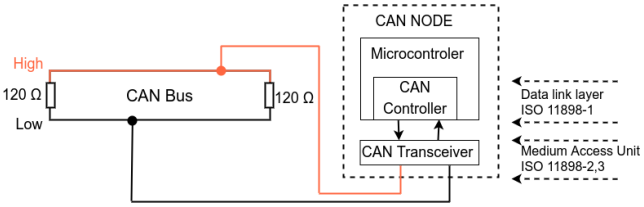

Figura 8. Estrutura de um módulo *CAN*

2.2 *UDS on CAN*

Protocolo padronizado para acessar serviços de escrita e leitura de uma determinada *ECU* ou algum módulo de controle, sendo sua padronização especificada pela ISO 14229-1. Utilizando o protocolo *CAN* como camada de transporte, o protocolo *UDS* possibilita o diagnóstico de diversos serviços e a estruturação da sua mensagem é distinta do *CAN*, embora seja aplicado ao mesmo. A estrutura da mensagem utilizando o protocolo *UDS* é apresenta pela Figura 9, na qual podem-se identificar os campos. Este padrão é aplicado ao *CAN*, mantendo

em comum o seu *ID* e o campo de dados *(Data)*, sendo nestes campos onde o *UDS* é aplicado. Em seu *ID*, deve-se definir o endereço de requisição da *ECU*, enquanto no campo de dados, de 8 *Bytes*, é o local no qual o *Protocol Control Info*, *Service Identifier* e *Sub Function*, *Request Data Parameters* e *Padding* são aplicados.

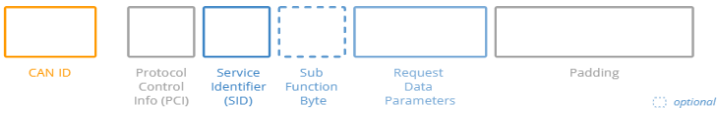

Figura 9. Mensagem de Requisição padrão *UDS*

Os campos de uma mensagem no padrão UDS on CAN são:

- *CAN ID* – Endereço de requisição do módulo de controle;
- *Protocol Control Info (PCI)* - Quantidade de *Bytes* utilizados pelos campos de dados que virão em seguida;
- *Service Identifier (SID)* – Endereço do serviço a ser requisitado ao módulo de controle;
- *Sub Function Byte* – Serviço opcional, mas por vezes, este campo é um complemento a *SID*;
- *Request Data Parameters* – Parâmetros de requisição de dados, é onde os dados serão enviados pelo usuário ou dados complementares provenientes da *ECU* serão informados, notificando como os dados virão em seguida; e
- *Padding* – *Bytes* restantes que compõem os dados enviados ou requisitados.

3. METODOLOGIA

Com a necessidade de realizar a leitura e a escrita de dados aplicando os conceitos dos protocolos *CAN* e *UDS*, desenvolveu-se um *Gateway* capaz de receber, enviar mensagens e processá-las de acordo com as suas padronizações. Este *Gateway* utiliza a linguagem de programação C para realizar o processamento dos dados contidos no barramento e requisições diretamente ao radar. O radar *OHW* retorna suas aferições por meio de seus canais utilizando o protocolo *CAN* padrão, porém, para solicitar mensagens objetivas de diagnósticos, utiliza-se o *UDS on CAN*.

Ao desenvolver a aplicação, lógicas para manipulações binárias foram implementadas, uma vez que as mensagens são recebidas em dados hexadecimais e somente os campos de *ID*, *DLC* e *Data* são os que podem ser manipulados de forma digital, ou seja, por meio da programação. Para isso, optou-se pela utilização da plataforma *Arduino*, a qual proporciona aplicações *CAN* de forma simples e intuitiva. O *Arduino Uno* não possui suporte ao protocolo *CAN*, porém existem formas de converter os dados provenientes do barramento. Com esse objetivo, obtou-se pela utilização do módulo MCP2515, o qual realiza a conversão do protocolo *CAN* para o protocolo *SPI (Serial Peripheral Interface*), possibilitando assim, a implementação do protocolo *CAN* com o microcontrolador, uma vez que o mesmo tem suporte ao *SPI*.

Aprofundando-se no desenvolvimento do *Node*, o *CAN Shield* MCP2515 é estruturado com um *CAN Transceiver* e um *CAN Controller*, sendo o próprio MCP2515 como *Controller* e o TJA1050 como *Transceiver*. Contando com a capacidade de transmissão de até 1 *Mbps*, o *Controller* é capaz de realizar o envio, recebimento e ainda aplicar os filtros do protocolo para as mensagens (Microchip *et al.* 2019), proporcionando, assim, todo o suporte para trabalhar com o protocolo *CAN* V2.0B ou *CAN* V2.0A *(Standard Frame)*. O módulo TJA1050, desenvolvido pela Philips, possibilita a integração do *CAN* com o barramento físico, isso de forma eletrônica diretamente com o *CAN Bus* (Philips Semiconductors *et al.* 2003).

3.1 Desenvolvimento do *Gateway*

Fundamentado nestes conceitos, implementou-se o *CAN Gateway*, contando com o *Shield* MCP2515, um módulo *Step-Down* LM2596, um conector DB9 e uma chave para ligar ou desligar o *gateway* com segurança. As respectivas funcionalidades de cada componente são descridas abaixo:

- MCP2515 - realiza a conexão do núcleo de processamento com o *CAN Bus*, possibilitando oacesso as mensagens;
- LM2596 – responsável por regular a tensão proveniente de uma fonte externa para a tensão

correta de acordo com a necessidade do projeto (*Texas Instruments et al*);

- Conector DB9 – conector padrão para módulos *CAN* como o *PCAN Router* da *Peak System*; e
- Chave Kcd4 – para ligar e desligar o *Node*.

O circuito do *Gateway* é apresentado na Figura 10(a), contendo todos os componentes descritos anteriormente. Após a validação do circuito por meio de testes, os quais se resumem em enviar, receber mensagens e processá-las, a manufatura do invólucro foi realizada. Este invólucro foi projetado por meio de *software CAD* e impresso em filamento *ABS* por uma impressora 3D, com o objetivo principal de proteger os componentes e garantir a durabilidade do *Gateway*. O *Gateway* em fase de montagem também pode ser observado na Figura 10(b).

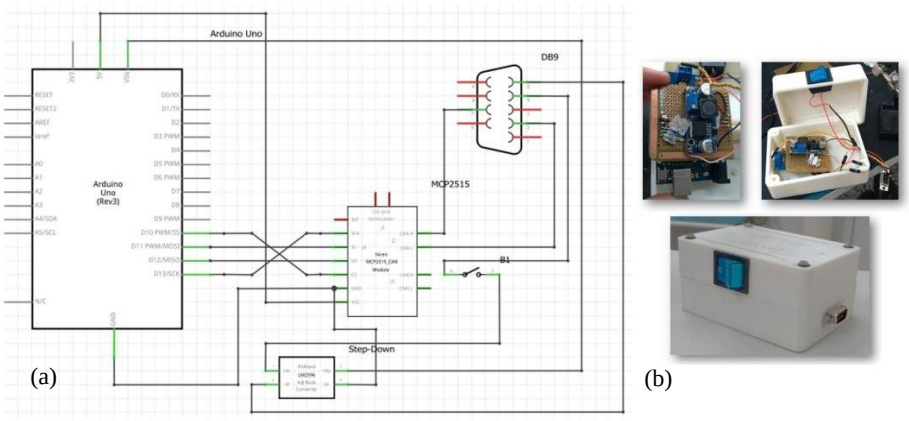

Figura 10. *Gateway* desenvolvido, a parte (a) é o círcuito eletrônico do módulo e a parte (b) o módulo montado

4. RESULTADOS E DISCUSSÕES

Após a montagem do *Gateway*, validações foram realizadas ao processar as mensagens no barramento. O radar *OHW* e o *Gateway* foram conectados ao *CAN Bus*, após o que ele foi capaz de realizar requisições ao radar e processar suas mensagens de forma sincronizada, possibilitando assim a implementação de algoritmos como o de segurança que funciona diretamente pelo uso do *protocolo UDS* (Ryo Kurachi *et al.* 2019). Dessa forma, a escrita e leitura de dados provenientes do radar *OHW* foram realizadas com sucesso, mostrando a versatilidade e o funcionamento distinto do protocolo *CAN*. Na figura 11, é demonstrada a requisição de leitura do serviço 0xFD28 à *ECU* do radar, observando assim, o comportamento esperado e documentado pelas padronizações do protocolo *(ISO 14229; Road vehicles – Unified Diagnostic Services (UDS)).*

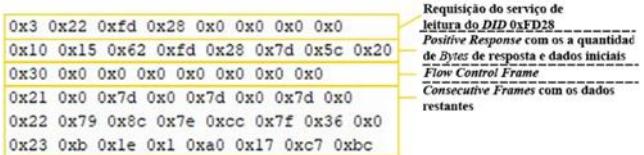

Figura 11. Requisição e Leitura de mensagens *CAN Multiframe*

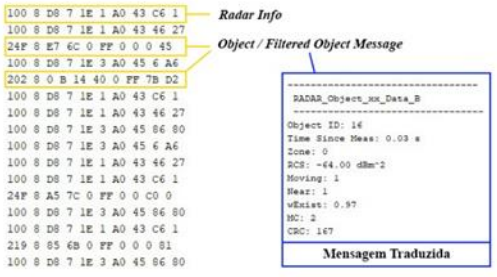

Figura 12. Leitura e Tradução de mensagens automáticas da *ECU*

Por padrão, o radar utilizado para os experimentos provia quatro diferentes mensagens diretamente ao *CAN Bus*. Na Figura 12 podem-se observar três delas, as quais foram processadas corretamente de acordo com as suas padronizações originárias do fabricante. Assim como as mensagens presentes no barramento, a Figura 12 apresenta uma das mensagens de *Object / Filtered Object* (mensagem com dados de objetos detectados) traduzida e de fácil interpretação de seus resultados obtidos pelo *Gateway*. Porém o ponto principal do

desenvolvimento foi a implementação do algoritmo de segurança, o qual viabilizou a escrita de dados na *ECU* do radar. Na Figura 13 observa-se o comportamento do envio das mensagens por meio do algoritmo, bem como a confirmação de que o radar aceitou a *Key* por meio da resposta 0x67 0x64.

Contudo, é essencial a sincronização dos processos realizados pelos métodos no núcleo de processamento, uma vez que qualquer dessincronia ao enviar as mensagens com o acesso de segurança pode causar erros. Para o auxílio a identificação de erros, o protocolo *UDS* retornará o tipo do serviço que não foi realizado e o motivo pelo qual ocorreu a falha.

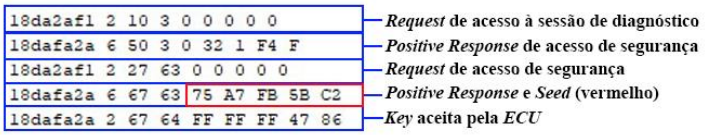

Figura 13. Execução do Algoritmo de Segurança

5. CONCLUSÃO

Este trabalho apresentou um método alternativo para a captura de dados provenientes do *CAN Bus*, utilizando um microcontrolador de baixo custo e, demonstrando assim, uma alternativa viável a projetos com orçamentos baixos. A implementação aborda a aplicação de dois protocolos, o *CAN (ISO 11898-1:2015*), o qual conta com uma gama de verificações, deixando assim, resistente a ruídos, e o protocolo *UDS on CAN (ISO 14229).*

Ao analisar os resultados obtidos por meio de testes contidos neste artigo, o *Gateway* demonstrou-se usual, viabilizando diversos pontos centrais para a realização de configurações do radar *OWH*, pois comunica-se apenas por meio dos protocolos anteriormente citados. Os principais objetivos para o seu desenvolvimento eram obter dados, escrever e realizar diagnósticos do sensor e, como pôde ser observado neste artigo, todos estes objetivos foram alcançados.

Por fim, a utilização do *Gateway* possibilita total acesso às informações contidas no *CAN Bus* e na *ECU* do radar, mostrando, assim, a sua aplicabilidade. Como trata-se de um microcontrolador programável, é possível modificar o programa interno, adaptando-o para qualquer outra aplicação onde os protocolos abordados são utilizados para a comunicação, demonstrando assim, a sua flexibilidade.

Conclui-se que a aplicação é funcional, capaz de substituir outros módulos que realizam a mesma função e flexível a outras aplicações, não somente com o radar *OHW*, mas sim com qualquer módulo que faz o uso da comunicação *CAN*. Comprova, ainda, a importância da utilização dos protocolos *CAN* e *UDS on CAN*, pois sensores e *ECUs* automotivas utilizam estes protocolos e, em sua maioria, é o unico meio pelo qual se alcança a comunicação com os módulos. Contudo, o *Gateway* apresenta potencial para aprimoramentos, inserções de novas funcionalidades para aumentar a flexibilidade já existente do projeto e futuros estudos para atualização de *software*.

REFERÊNCIAS

Corrigan, S. (2002). Application Report Introduction to the Controller Area Network (CAN). [online] Available at: https://www.rpi.edu/dept/ecse/mps/sloa101.pdf.

SPECIFICATION, C. A. N. Bosch. Robert Bosch GmbH, Postfach, v. 50, 1991.

Kurachi, Ryo, et al. "Evaluation of Security Access Service in Automotive Diagnostic Communication." 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring). IEEE, 2019.

ISO - International Organization for Standardization. ISO 11898-1:2015; Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signaling. Geneva: ISO, 2015

ISO - International Organization for Standardization. ISO 14229; Road vehicles – Unified Diagnostic Services (UDS). Geneva: ISO, 2013

Instruments, Texas. "LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator." LM2596D datasheet, Nov (1999).

Microchip, Arizona. “Stand-Alone CAN Controller with SPI Interface” MCP2515 datasheet, Jan (2019)

Semiconductors, Philips. "TJA1050 High speed CAN transceiver." California USA (2003).