Brazilian Journal of Instrumentation and Control
Revista Brasileira de Instrumentação e Controle

Decodificação e Análise dos dados de um Sensor Comercial de Esterçamento de Volante (SAS) com comunicação CAN

Data Decodification and Analysis from Comercial Steering Angle Sensor with CAN communication

Felipe A. Farinelli and Sergio L. Stevan Jr.

***Abstract*—This work were developed to read and decode a Steering Angle Sensor (SAS) for use information provided by it in future applications. It was made all the bibliographic survey needed by the development of the validation tool. It were shown some concepts about CAN (Controller-Area-Network) Networks, which were used in the connection between the SAS sensor and the validation tool, as well as SPI (Serial Peripheral Interface), the protocol needed to integrate a CAN device and the development board in which were implemented all logics to decode and evaluate data. Using a display, it was programmed an interface for show all the decoded data to user. With the complete development of the validation tool, sensor evaluation were made and it shown satisfactory results after the comparison between the real measurement made by a computational software and the real value.**

***Keywords* —Steering Angle Sensor, Rede CAN, Protocolo SPI.**

***Resumo*—Este trabalho teve por principal objetivo a leitura e a decodificação dos dados de um Steering Angle Sensor (SAS) para a utilização destes em aplicações futuras. Foi feita toda a fundamentação teórica necessária ao desenvolvimento da ferramenta utilizada para a validação. São demostrados conceitos base sobre Redes CAN (*Controller Área Network*), utilizada na conexão do sensor com a ferramenta de validação, bem como SPI (*Serial Peripheral Interface*), o protocolo utilizado na integração entre o dispositivo CAN utilizado e a placa de desenvolvimento na qual foi implementada toda a lógica para a decodificação e análise dos dados e, com a ajuda de uma tela, foi programada a interface de exibição dos dados ao usuário. De posse do equipamento completo para a validação, esta foi feita e apresentou resultados satisfatórios, visto que as informações obtidas do sensor foram comparados às reais e validadas através de um *software* computacional.**

***Palavras-chave*—Sensor de Ângulo de Esterçamento, Rede CAN, Protocolo SPI.**

I. INTRODUÇÃO

A evolução dos veículos levou ao crescimento do número de sensores utilizados na construção destes, tanto que atualmente, qualquer veículo tem em sua estrutura um conjunto de sensores que compõe diversos subsistemas que vão desde os controles de temperatura e ruído em cabine, quanto a sensores de pressão, velocidade, presença, Oxigênio entre outros, do sistema de tração [1]. Muitas destas grandezas podem ter grande importância à segurança dos usuários, outras podem ajudar a monitorar o desempenho, e outras ainda podem ser importantes à manutenção do veículo [2].

Neste contexto, deve ser considerado como um sensor (ou transdutor), todo elemento capaz de converter uma grandeza encontrada na natureza para outra de natureza elétrica, o que permite a medição, a análise e o controle de situações através de elementos que quase sempre são digitais, como os microcontroladores [3].

Alguns exemplos de sensoriamento veicular consistem de sistemas de segurança que utilizam sensores de esterçamento do volante (SAS), utilizados para prever comportamento do

Felipe Adalberto Farinelli, Universidade Tecnológica Federal do Paraná (UTFPR), Ponta Grossa, Paraná, Brasil, e-mail: felipe.farinelli@outlook.com); Sergio Luiz Stevan Jr. , Universidade Tecnológica Federal do Paraná (UTFPR), Ponta Grossa, Paraná, Brasil, (e-mail: sstevanjr@utfpr.edu.br).

motorista [4], através da correlação entre a velocidade do veículo e do ângulo de esterçamento do volante, e analisar este comportamento no sentido de cansaço [5], via fusão de sensores com variáveis correlacionadas, e estresse [6], através da fusão entre o ângulo do volante e sensores de batimento cardíaco.

Nota-se que existem aplicações que exigem um valor preciso vindo de sensores [4][5][6], logo um controle sobre os seus resultados de medição é importante, como no caso do SAS. Existem centrais de controle específicas para cada tipo de sensor, visto que há variados tipos destes que leem grandezas analógicas e as transformam em grandezas elétricas digitais [7] ou as traduzem para outra grandeza analógica [8].

Uma vez que há inúmeros sensores em um veículo dos mais variados tipos, então é necessário que exista uma maneira de interconectá-los em uma rede de comunicação padrão, o que torna possível a vinculação destes com todas as centrais de controle eletrônicas (ECUs) dos veículos [9].

Há vários tipos de redes de comunicação intra-veiculares, dos quais podem ser considerados como principais o LIN (*Local Interconected Network*), o K-line (Linha K) e o CAN (*Controller Area Network*), visto que os dois primeiros interconectam sensores físicos à ECUs (Unidades de Controle Eletrônicas) e o último usualmente interconecta as ECUs[10]. Uma vez que o sensor a ser avaliado neste trabalho é constituído de uma ECU com saída CAN, então este será o protocolo utilizado para a aquisição dos dados do sensor.

O protocolo de comunicação intra-veicular CAN utiliza um barramento de duas linhas com capacidade de conectar até 32 dispositivos em velocidades que variam de 5 a 1000 Kbps [10], o qual não é suportado na grande maioria dos microcontroladores de baixo custo, como no caso dos ATMEGA328p e ATMEGA2560, principais elementos das placas de prototipagem rápida Arduino UNO e MEGA [11].

O fato do suporte a CAN não ser nativo no Arduino, é necessária a utilização de circuitos dedicados responsáveis por a criar uma rede CAN (MCP2515 [12]) os quais se comunicam com o Arduino através de protocolos seriais padrões dos microcontroladores, tal como o SPI (*Serial Peripheral Interface*) [13].

Para tornar possível a leitura dos dados provenientes do sensor SAS decodificado neste trabalho, foi acoplado ao Arduino um microcontrolador CAN dedicado, o qual foi programado para realizar a interface entre este e o sensor.

Uma vez citadas algumas aplicações que utilizam dados provenientes do sensoriamento do volante, à rede CAN foi instalado um sensor SAS OEM (*Original Equipment Manufacturer*) totalmente codificado pela fabricante, do qual foi necessária a interpretação dos dados bem como a decodificação destes, ambos procedimentos que serão detalhados no desenvolvimento do trabalho, para uma utilização em trabalhos futuros voltados à área de controle automotivo.

A validação do sistema foi feita com base em um sistema de visão, que foi possibilitado através da utilização de um *software* voltado a análise de imagens e ângulos.

II. CAN – *CONTROLLER ÁREA NETWORK*

O protocolo CAN foi idealizado por Robert Bosch em 1986 [10] para então ser adotado mundialmente ainda na década de 80, voltado a comunicação de dispositivos veiculares e automação industrial [14].

Para que fosse estabelecido um padrão na construção de dispositivos CAN compatíveis, o protocolo foi normatizado pela *International Standardizarion Organization* (ISO) através da norma ISO 11898 em 1994 [10].

Pode-se adaptar a rede CAN ao modelo de redes OSI (*Open System Interconnection*), por se tratar de um meio de comunicação. O modelo prevê sete camadas bem definidas nomeadas por Física, Enlace de Dados, Rede, Transporte, Sessão, Apresentação e Aplicação [15], das quais são utilizadas pela CAN as apresentadas pela Fig. 1 [13].

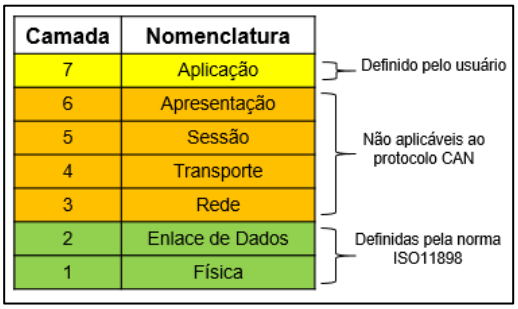

Fig. 1: O protocolo CAN conforme o modelo OSI [10].

O protocolo CAN utiliza, das sete camadas do modelo OSI, apenas a camada Física, de Enlace de Dados e de Aplicação. A última varia conforme a necessidade do usuário, porém as primeiras são devidamente normatizadas pela ISO 11898 [16].

A. *Camada Física*

A camada física do protocolo de comunicação CAN consiste de apenas dois fios, nos quais dispositivos compatíveis são conectados em topologia do tipo barramento [9].

Uma vez que circulam tensões em ambos os fios, nomeados comumente como CAH H (CAN *High*) e CAN L (CAN *Low*), é necessária a inserção de terminadores na rede para que se evitem fenômenos de perda ou distorção de informações [17]. Esses terminadores são resistores conectados em paralelo a rede, cujo valor nominal é sugerido em norma e translada de 100 a 120Ω. A Fig. 2 apresenta um exemplo teórico de barramento CAN.

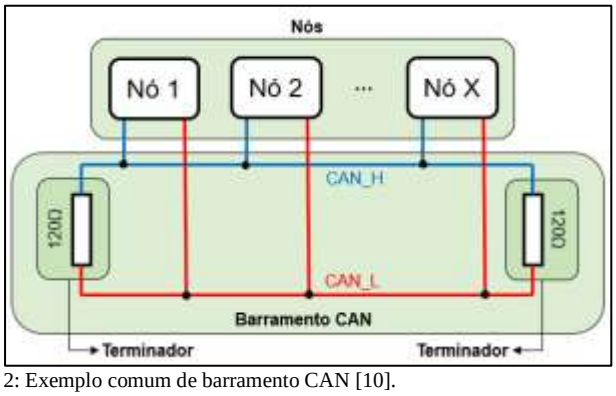

Fig. 2: Exemplo comum de barramento CAN [10].

Todos os elementos conectados à CAN, excetuando-se pelos terminadores, são chamados de nós [16]. Um nó pode ser um sensor CAN, como o SAS que será tratado posteriormente, ou outra ferramenta compatível com o protocolo, como um *scanner* de diagnóstico automotivo [18].

A tensão nominal das linhas CAN H e CAN L é de 2,5 V [15]. O que define um bit no barramento CAN é o módulo da tensão diferencial que circula nas duas linhas, conforme sugere a Fig. 3.

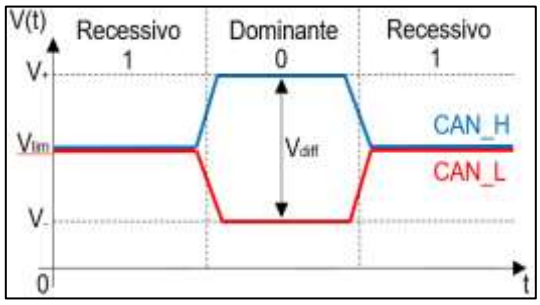

Fig. 3: Dados CAN dentro do barramento físico.

Para que um bit seja considerado dominante (ou 0), a tensão diferencial deve variar entre 3,6 e 5 V. Se esta diferença estiver entre 0 e 2,2 V, então o bit é considerado recessivo (ou 1) [16]. O tratamento destes dados é feito na camada de Enlace de Dados, na qual cada sequência binária é tratada como parte de uma informação.

B. Camada de Enlace de Dados

A camada de enlace de dados do protocolo CAN consiste de dois tipos, o CAN 2.0A o CAN2.0B. O primeiro, utilizado por este trabalho, é o mais utilizado em veículos de pequeno porte, enquanto o segundo é o padrão em veículos pesados, vans e caminhões [10].

Uma vez que um barramento CAN é constituído de vários nós, então cada um disponibiliza informações na rede. Nota-se que em um veículo existem informações de muitos tipos, dos quais alguns são cruciais ao funcionamento do sistema e outras não, o que torna necessário a priorização alguns dados. Em apenas uma rede, podem existir até 2047 informações diferentes, conhecidas por mensagens ou *frames* [19].

A Fig. 4 apresenta um *frame* CAN do tipo 2.0A, no qual são apresentados todos os campos constituintes de uma mensagem padrão.

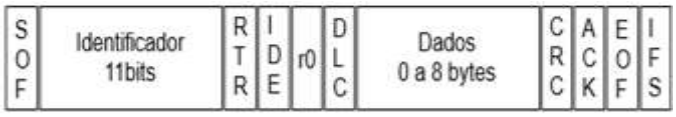

Fig. 4: Mensagem CAN do tipo 2.0A, ou mensagem CAN padrão.

A prioridade de um dado é definida no ID (Identificador) da mensagem. Uma vez que um bit dominante define a prioridade [16], então quanto menor o identificador, mais rapidamente este consegue o acesso ao barramento.

O campo do identificador é constituído de 11 bits, que variam conforme a unidade de controle eletrônica (ECU) que envia a mensagem, enquanto que o RTR (*Remote Transmission Request*) define se o *frame* é de dados ou uma simples requisição de informações [10].

É necessário identificar o tipo da mensagem, que é configurado no campo IDE (*IDentifier Extension*) como padrão (11 bits) ou 2.0B (estendido, 29 bits). Como uma mensagem de dados pode conter até 8 bytes [10], então é necessário previamente definir o comprimento desse campo, alterando os 4 bits do DLC (*Data Length Code*) entre 0000, para 0 bytes de dados e 1XXX, para 8 bytes de dados.

O protocolo CAN possui um mecanismo de detecção e correção de erros, que consiste de um CRC (*Cyclic Redundancy Check*) de 15 bits capaz de identificar até 6 bits errados em uma transmissão [16]. Se a mensagem foi recebida corretamente pelo nó de destino, então este permuta o bit ACK (*Acknowledgement*), indicando que a recepção dos dados ocorreu corretamente.

Resta ainda a definição dos campos SOF (*Start-of-Frame*) e EOF (*End-of-Frame*), que indicam o início e o fim da mensagem, respectivamente. Os campos r0 (*Reserved bit 0*) e IFS (*Inter-frame Space*) também são padrões das mensagens CAN, nos quais o primeiro é reservado a aplicações específicas e o segundo determina o espaço entre uma mensagem e outra no barramento real.

O suporte nativo a CAN ocorre em alguns microcontroladores, como nos da família PIC32 fabricados pela Microchip® [20], porém no ATMEGA 2560, microcontrolador central do Arduino MEGA®, utilizado neste trabalho, não há suporte [11].

Por outro lado, há uma interface de comunicação entre microcontroladores denominada SPI (*Serial Peripheral Interface*) que é compatível com o Arduino e permite que outras funções sejam acrescentadas à placa de desenvolvimento através de microcontroladores externos.

III. SPI – *SERIAL PERIPHERAL INTERFACE*

O SPI é um protocolo de comunicação síncrono entre microcontroladores que foi adotado devido a operar em altas frequências e mostrar-se rápido e eficiente na transferência de dados [21].

Um sistema de comunicação SPI é constituído de um dispositivo mestre que controla a operação de outros dispositivos escravos [22] através de quatro canais: MISO (*Master-in Slave-out*), MOSI (*Master-out Slave-in*), CS (*Chip Selector*) e CLK (*Clock*).

Quando o mestre manda qualquer informação a um escravo, então esta segue serialmente pelo canal MOSI, enquanto que, ao mesmo tempo, o escravo retorna informações pelo canal MISO de modo *full-duplex*, que consiste da troca de informações em dois sentidos e ao mesmo tempo através de canais diferentes [23], sincronizados por um *clock* (CLK) controlado pelo mestre.

Visto que um único mestre controla vários escravos, então é necessário definir para qual dos escravos a informação é destinada, o que é feito através do canal CS no qual é enviado um sinal em nível lógico baixo ao escravo de destino.

Há alguns modos de operação importantes compatíveis com

o SPI, dos quais há um que foi o único utilizado por possuir compatibilidade com os escravos utilizados.

A. *Modos de Operação do SPI*

Existem três modos de operação de equipamentos compatíveis com SPI: Modo 0, Modo 1, Modo 2 e Modo 3. Cada modo é definido conforme o tipo do *clock* e da borda do mesmo, onde será feita a amostragem de cada bit da transmissão [24].

São basicamente duas informações que definem o modo de operação do SPI, das quais ambas são relacionadas ao sinal de *clock* e definidas por polarização (CPOL) e fase (CPHA).

Quanto a polarização, o *clock* pode ser normalmente baixo ou alto. Isto permite que, quando não está ocorrendo transmissão, o sinal de *clock* seja de nível alto ou baixo. A fase não interfere no sinal, pois apenas controla se a amostragem da informação no barramento é na borda de subida ou de descida do *clock*.

IV. Dispositivos Utilizados

Para o desenvolvimento deste trabalho, foram utilizados quatro dispositivos importantes: um Arduino MEGA®, um módulo dedicado CAN, uma tela colorida do tipo TFT LCD (*thin-film-transistor Liquid Crystal Display*) e um sensor SAS OEM, do qual não são conhecidas todas as especificações e que foi o alvo de estudo deste trabalho.

A. *Arduino MEGA®*

O Arduino MEGA® é uma placa de desenvolvimento dotada de um microcontrolador central ATmel® ATMEGA 2560, compatível com SPI e outros protocolos de comunicação como Serial e IIC (*Inter-Integrated Circuit*) [25].

Por ser uma plataforma *open-source* (Plataforma de Código Aberto), há várias bibliotecas de programação para o Arduino disponíveis na internet que facilitam o desenvolvimento utilizando este *hardware*.

A placa é composta por 54 pinos de entrada e saída digitais, dos quais 15 são compatíveis com PWM (*Pulse Width Modulation*). Há ainda 16 conversores A/D (Analógicos para Digitais) de 10 bits cada, que possibilitam a interface entre a placa e grandezas elétricas externas.

B. *Tela TFT LCD*

Para auxílio na validação da leitura dos dados provenientes do sensor SAS, foi escolhida uma tela LCD para a exibição dos valores obtidos do sensor. O modelo escolhido foi o 3.2" TFTLCD *Shield for Mega2560*, que possui uma resolução de 480 colunas por 320 linhas e mais de 65000 cores.

C. *Módulo dedicado CAN*

O módulo CAN utilizado é composto de um *transceiver* (transceptor) CAN TJA1050 fabricado pela NXP® Semiconductors, responsável pela conversão da tensão diferencial do barramento CAN físico para uma comunicação serial [26], e de um microcontrolador MCP2515 fabricado pela Microchip®, totalmente compatível com a ISO 11898.

O microcontrolador CAN, único utilizado no desenvolvimento deste trabalho, é compatível com SPI em Modo 0 no qual o *clock* é normalmente baixo e a amostragem dos dados ocorre na borda de subida. A Fig. 5 foi obtida de um barramento SPI real utilizando uma ferramenta chamada Analisador Lógico, e representa a comunicação SPI em modo 0.

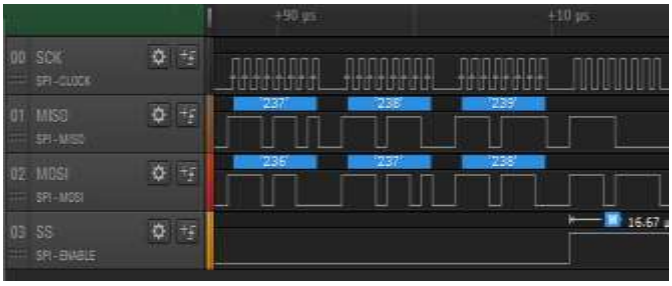

Fig. 5: Operação do SPI em Modo 0.

Nota-se que, apenas como exemplo, são enviados os bytes 236, 237 e 238 do mestre para o escravo pelo canal MOSI, enquanto que o escravo envia ao mesmo tempo para o mestre, pelo canal MISO, os bytes 237, 238 e 239. É importante avaliar que a interpretação dos dados ocorre pelos dispositivos apenas enquanto o escravo é selecionado, através do canal CS.

D. *SAS – Steering Angle Sensor*

O SAS é o sensor alvo deste trabalho, uma vez que por ser uma peça OEM, não há nenhum tipo de especificação disponível para tal.

Sabe-se que a fabricante do sensor é a Bourns [7] e que este é capaz de disponibilizar a medição do ângulo de esterçamento da barra de direção do veículo bem como a velocidade angular com a qual é realizada uma manobra no volante, além de outras informações de configuração internas.

A fabricante possibilita que sejam alterados quase todos os parâmetros do sensor, exceto os pinos de alimentação e da saída CAN (CAN H e CAN L) através de uma mensagem CAN padrão com um formato específico, que também pode ser alterado, o que permite a total codificação do sensor por parte do fabricante do veículo no qual o sensor está instalado.

Na Fig. 6 é apresentado o sensor utilizado e as únicas informações quanto ao seu modelo, às quais possibilitaram encontrar sua fabricante.

Fig. 6: Sensor SAS. A esquerda: visão superior; e à direita: visão inferior.

Uma vez citados todos os elementos necessários para a validação das informações disponibilizadas pelo sensor, encontrado facilmente para compra na internet, resta a programação dos dispositivos bem como a decodificação dos dados, apresentados nos tópicos a seguir.

V. Estrutura do Equipamento de Validação

Para que fosse possível avaliar os dados disponibilizados pelo sensor, foi necessário propor um equipamento de validação, o qual foi programado para que apresentasse o resultado da medição do sensor de uma maneira técnica e clara. Com este objetivo, foi proposta uma arquitetura semelhante à apresentada na Fig. 7.

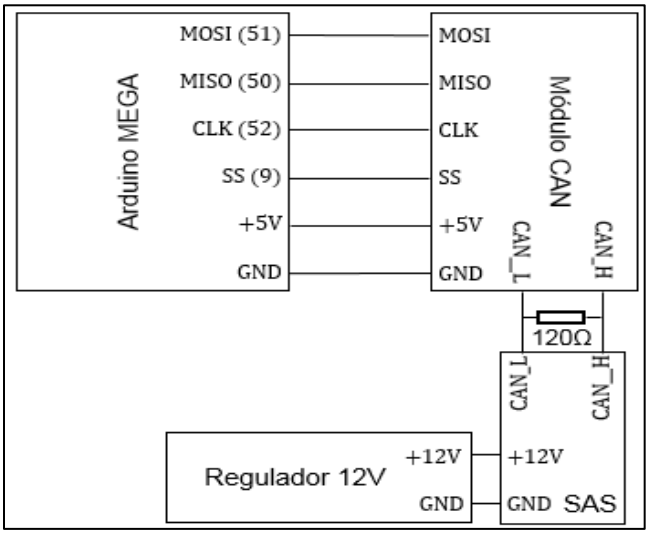

Fig. 7: Estrutura do equipamento para avaliação do sensor SAS.

Nota-se que a tela não está representada na estrutura. Isso ocorre devido a esta ter dimensões compatíveis com a o Arduino MEGA®, e encaixar-se como uma *shield* (escudo).

Já com a estrutura definida, foi possível montar o equipamento, que ficou com estrutura semelhante à apresentada na Fig. 8.

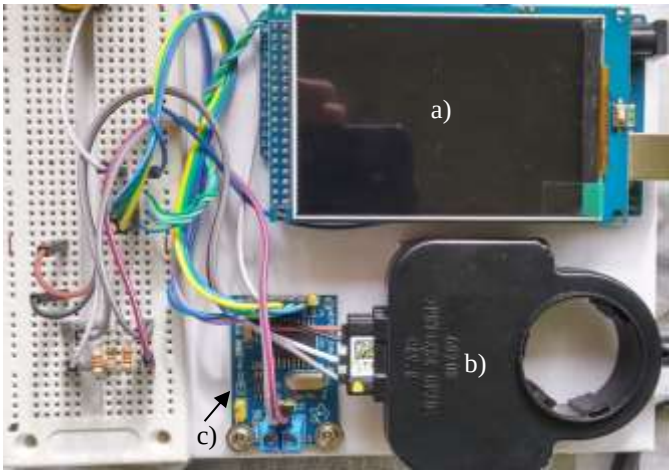

Fig. 8: Circuito completo de validação, em que: a) Arduino MEGA® com tela LCD sobreposta e encaixada; b) sensor SAS; e c) Módulo CAN.

VI. Metodologia

Para possibilitar a leitura dos dados CAN do sensor, um módulo CAN foi instalado e programado em um Arduino MEGA, o qual ficou responsável, além de gerenciar os dados CAN, tanto pela parte de decodificação dos dados quanto pela execução da interface gráfica que exibe em tempo real todos os dados obtidos do sensor.

Para realizar a validação dos dados provenientes do sensor SAS foi feita uma sequência de imagens que indicam esterçamento à esquerda e à direita, nas quais podem ser vistas tanto a tela que mostra o valor do ângulo decodificado empiricamente quanto o sensor real esterçado.

Para avaliar o ângulo real do sensor, foi utilizado o *software* Kinovea®, pois com ele é possível identificar ângulos em uma figura utilizando apenas alguns pontos de referência, que já é suficiente para validar o resultado obtido com o resultado real.

VII. Programação do Dispositivo de Validação

Com todos os elementos já conectados no sistema, deu-se início à programação do dispositivo utilizado à decodificação do sensor CAN SAS. Esta programação foi feita toda no Arduino MEGA®, o qual gerenciou todos os dispositivos conectados a ele.

O primeiro passo foi a programação da interface completa e suficiente à exibição dos dados. Foi utilizado no Arduino a biblioteca UTFT [27] que disponibiliza funções para o desenho de *pixels* e linhas, bem como funções para exibição de textos.

De posse de todos os elementos necessários à programação, esta foi feita e resultou na interface apresentada pela Fig. 9. Nota-se que é possível ver uma mensagem CAN detalhada, bem como o resultado do dado traduzido pelo sensor. Por ainda não ter ocorrido a programação da interface CAN, os dados foram simulados para tornar possível o desenvolvimento.

Fig. 9: Interface da ferramenta de validação

É possível ver várias propriedades de uma mensagem CAN tais como o Identificador (CAN ID), se a mensagem é enviada (Tx) ou recebida (Rx), o comprimento do campo de dados (DLC), o tipo (Dados ou Remota) e os dados propriamente ditos, D7 a D0, da esquerda para a direita.

O campo central da tela é dedicado a mostrar a informação mais importante lida do sensor, o ângulo de esterçamento. O tamanho dessa informação foi destacado devido a auxiliar na

validação descrita posteriormente.

Já definida a interface do equipamento com o usuário, foi programada então a interface CAN para possibilitar a leitura dos dados fornecidos pelo sensor. O primeiro passo foi a definição da velocidade do barramento que, por ser um parâmetro imutável via programação do sensor [7] tal como a alimentação e a interface CAN deste, foi seguida a especificação de 500 Kbit/s.

Para adequar o módulo CAN a trabalhar com esta velocidade, foi necessário programar o microcontrolador MCP2515 diretamente por seus registradores via SPI utilizando instruções específicas, as quais estão disponíveis na folha de dados do componente.

Uma vez programada a velocidade do barramento CAN no qual o sensor foi conectado, então é necessário habilitar o módulo CAN a receber mensagens, o que também é feito através de instruções SPI específicas. Nota-se que, por ser compatível com SPI, o microcontrolador dispensa qualquer tipo de ambiente de desenvolvimento próprio.

O MCP2515 possui três *buffers* de recebimento de mensagens, dos quais dois são manipuláveis via SPI e um recebe todas as mensagens do barramento. Uma vez que o sensor SAS envia apenas uma mensagem, então não é necessário utilizar filtros de recebimento.

Quando é recebida uma mensagem, o microcontrolador habilita uma interrupção. Nesse momento, já pode ser lida via SPI a mensagem recebida, pois esta fica armazenada em registradores específicos enquanto não for desabilitada a interrupção. Ao desabilitá-la, o componente já está apto a receber outra mensagem, independente se esta vem da mesma ECU ou não.

Definida a interface com o usuário, a velocidade da rede CAN e a forma como são recebidas as mensagens, então resta agora a decodificação dos dados provenientes do sensor.

VIII. Decodificação dos Dados do Sensor

O primeiro passo para a decodificação do sensor foi a visualização dos seus dados. O microcontrolador CAN recebeu uma mensagem de dados com identificador 0x1E5 e DLC de 8 bytes, informações que já divergiram em relação às especificações originais do sensor.

Originalmente, o sensor deveria enviar uma mensagem com ID 0x280 e DLC de 7 bytes [7], o que mostra que suas configurações foram alteradas pela fabricante do veículo no qual ele é instalado.

O mesmo acontece no campo de dados da mensagem. Segundo a fabricante do sensor, os *bytes* de dados D0 e D1 são os responsáveis pela exibição do ângulo de esterçamento, porém de fato esses dados se mostraram incompatíveis, visto que D0 sempre é estático, independente do ângulo ao qual o sensor está submetido.

Após experimentação e observação prática, foi possível definir que o ângulo é mostrado pelos bytes D6 e D5, no qual D6 corresponde ao byte mais significativo do valor do ângulo e D5 ao byte menos significativo. Chegou-se a essa conclusão devido a esses bytes se alterarem enquanto era forçado um ângulo de esterçamento no sensor.

Para definir os limites mínimos e máximos, o sensor foi forçado até suas informações serem alteradas bruscamente. Segundo a folha de dados do sensor, este apresenta uma precisão de 0,1° e mede até 780° à direita ou à esquerda, porém o sensor se mostrou confiável até a angulação de 900° em ambos os sentidos.

Quando esterçado todo à direita, o sensor apresentou informações confiáveis até D6 e D5 apresentarem os valores 0xC7 e 0xC0 respectivamente, o que em decimal representa um total de 51136 amostras. Quando além desse valor, D6 passa a valer 0x38 e D5 0x3F, que em decimal significa 14399.

Ao ocorrer a alteração brusca, o sensor continuou sendo esterçado à direita até D6 e D5 serem iguais a 0. Este ponto foi definido como o ponto de origem do sensor, ou seja, como o ponto no qual o ângulo de esterçamento corresponde a 0.

O comportamento notado foi que, quando o sensor é forçado à direita, ocorre uma diminuição dos valores de D5 e D6, no qual a primeira amostra é D6 = 0xFF e D5 = 0xFF e a última é D6 = C7 e D6 = C0, o que dá um total de 14399 amostras. Após esse valor, D6 e D5 passam a valer 0x38 e 0x3F respectivamente, indicando uma alteração brusca.

Para avaliar o comportamento do esterçamento à esquerda, o sensor foi reajustado no ponto de origem. Empiricamente, foi visto que a primeira amostra para a esquerda é D6 = 0 e D5 = 0x01 e a última antes da alteração brusca é D5 = 0x38 e D6 = 0x3F. Após essa amostra, D5 e D6 passam a valer 0xC7 e 0xC0, respectivamente.

De posse dos intervalos à esquerda e à direita, é possível descobrir a precisão do sensor. Uma vez que o número de amostras a esquerda e a direita é o mesmo (14399), então há um total de 28798 amostras. Uma vez que o sensor apresentou valores confiáveis na medição de até aproximadamente 900° para a direita e para a esquerda, então pode-se afirmar que há 28798 amostras para 1800°, o que indica que cada amostra equivale a 0,0625043405792069° aproximadamente, diferente do especificado pelo fabricante, que era de 0,1° [7].

Através de experimentos empíricos, pôde-se constatar que o sensor envia além do ângulo de esterçamento, também a velocidade angular, pois há uma alta variação do byte D3 apenas quando o eixo do sensor está em movimento, conforme indicado em sua folha de dados.

A velocidade medida pelo sensor foi desconsiderada neste trabalho, visto que os dados passados pela fabricante do sensor não se mostraram mais confiáveis após a modificação realizada pela montadora do veículo (OEM) neste.

Uma vez decodificado o valor de medição do ângulo de esterçamento fornecido pelo sensor, bem como sua precisão, foi possível implementar estas informações no Arduino MEGA® e então apresentar os dados ao usuário através da interface desenvolvida anteriormente, o que possibilitou validar a decodificação destes dados.

IX. Validação e Experimentação dos Dados Obtidos do SAS

Inicialmente, o sensor foi colocado em seu ponto de origem para que fosse possível a medição de ângulos de esterçamento à direita e à esquerda à partir do mesmo ponto. Com o *software*

Kinovea®, foi inserido um pequeno zoom do campo de informações da mensagem recebida para auxiliar na visualização da imagem.

Nota-se que foi recebida uma mensagem de dados com identificador 0x1E5 e DLC de 8 bytes, como relatado no tópico anterior e pode ser visto na Fig. 10.

Fig. 10: Cabeçalho da Mensagem recebida

Validada a mensagem recebida, é possível então analisar os dados. No mesmo *software*, foi definido o ponto de origem, onde D6 e D5 equivalem a zero. Foi inserido um elemento para a visualização de ângulo, posicionado no centro do sensor e a resposta obtida é apresentada pela Fig. 11.

Fig. 11: Sensor posicionado no ângulo de origem.

É possível perceber que, assim como o ângulo, tanto D5 quanto D6 mostrados pelo quadro superior são iguais a 0, bem como o ângulo demarcado pelo *software*.

Uma vez calculada a precisão do sensor, foi possível validar esta também, movendo o sensor até a primeira amostra, o que torna D6 = 0 e D5 = 0x01. Como o equipamento apenas mostra três casas decimais, então a precisão foi arredondada corretamente a 0.063° na primeira amostra, como pode ser visto na Fig. 12.

Fig. 12: Avaliação da precisão do sensor.

Já avaliadas tanto a precisão do sensor quanto seu ponto de origem, é possível realizar medições. Foram estudados quatro ângulos, 90° à esquerda e 270° à direita, bem como o esterçamento máximo em ambos as direções.

Para 90° à esquerda, D6 e D5 resultaram em 0x05 e 0xA0, que, em unidades decimais, significa um total de 1440 unidades. Esse valor multiplicado pela precisão resultou no ângulo proposto, como pode ser visto na Fig. 13.

Fig. 13: Medição de um ângulo de 90°.

Quanto ao ângulo de 270° à direita, o sensor foi deslocado até chegar à esse ponto. Como a origem foi definida no ponto 0, D6 e D5 apresentaram um valor de 0xEF e 0x20

respectivamente, o que em decimal significa 61227.

Como foi invertida o sentido de deslocamento do sensor e o valor final é de 16 bits, então o número máximo de amostras é 65535. Se subtraído o valor obtido do número máximo de amostras, então obtêm-se 4308 que, multiplicado pela precisão resulta em 270°, o que valida a medição como mostrado na Fig. 14.

Fig. 14: Validação do ângulo de 270° à direita.

Resta apenas a validação dos ângulos máximos, tanto à direita quanto à esquerda. Dessa forma, o sensor foi novamente posicionado em seus extremos, nos quais foi possível validar os valores máximos à direita e à esquerda dos bytes D6 e D5 citados anteriormente e mostrados nas Figuras Fig. 15 e Fig. 16.

Fig. 15: Esterçamento máximo à direita

Fig. 16: Esterçamento máximo à esquerda.

Uma vez validados os principais ângulos medidos pelo sensor, então pode-se dizer que o tanto o sensor quanto a ferramenta de medição são confiáveis, o que leva a conclusão de que pode-se medir um ângulo de esterçamento real utilizando a ferramenta desenvolvida nesse trabalho.

X. CONCLUSÃO

Este trabalho demonstrou de forma abrangente variadas informações quanto aos protocolos CAN e SPI, utilizados no desenvolvimento da ferramenta de validação voltada a análise dos dados disponibilizados pelo sensor SAS, a qual possuía por principal objetivo exibir ao usuário, de maneira simplificada, o valor de medição enviado pelo sensor.

Para a exibição de dados ao usuário, foi desenvolvida uma interface gráfica em uma tela LCD que apresenta tanto o ângulo de esterçamento medido, já decodificado, quanto todas as outras informações codificadas que o sensor envia em formato de mensagem CAN.

Uma vez programada a interface, foi possível desenvolver a comunicação entre o módulo CAN e o sensor, que se mostrou eficiente e suficiente na validação dos resultados obtidos da medição utilizando o SAS.

No início do trabalho foram citadas algumas aplicações nas quais são utilizados os dados de esterçamento no volante. Nota-se que, como os dados apresentados nesse trabalho foram validados, então este sensor pode ser usado sem nenhum impedimento nas aplicações citadas.

Foram utilizadas poucas informações provenientes da folha de dados do sensor SAS, visto que muitas de suas especificações foram alteradas pela montadora que o utiliza em seus automóveis. Pode-se considerar que a única informação de fato verídica obtida do sensor foram os dados que ele fornece e como são dispostos os pinos da alimentação e do barramento CAN.

Foi proposta a decodificação do sensor para a utilização em

outros trabalhos. Dessa forma, pode-se concluir que este objetivo foi alcançado visto que a principal informação fornecida pelo sensor foi de fato lida e decodificada e, portanto, torna possível a utilização dos dados fornecidos em trabalhos nos quais estes são necessários.

REFERÊNCIAS

[1] POSADA, Francisco. On-board diagnostics for heavy-duty vehicles: Considerations for Mexico. International Council on Clean Transportation, 2014.

[2] SUWATTHIKUL, Jittiwut; MCMURRAN, Ross; JONES, R. Peter. Automotive network diagnostic systems. In: 2006 International Symposium on Industrial Embedded Systems. IEEE, 2006. p. 1-4.

[3] FLEMING, William J. Overview of automotive sensors. **IEEE sensors journal**, v. 1, n. 4, p. 296-308, 2001.

[4] KIM, Jinwoo; KIM, Kitae; YOON, Daesub. Fusion of driver-information based driver status recognition for co-pilot system. **2016 Ieee Intelligent Vehicles Symposium (iv),** [s.l.], p.1398-1403, jun. 2016. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/ivs.2016.7535573.

[5] DAZA, I. G. et al. Drowsiness monitoring based on driver and driving data fusion. **2011 14th International Ieee Conference On Intelligent Transportation Systems (itsc),** [s.l.], p.1199-1204, out. 2011. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/itsc.2011.6082907.

[6] RAMESH, Maneesha V.; NAIR, Aswathy K; KUNNATHU, Abishek Thekkeyil. Real-Time Automated Multiplexed Sensor System for Driver Drowsiness Detection. **2011 7th International Conference On Wireless Communications, Networking And Mobile Computing,** [s.l.], p.1-4, set. 2011. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/wicom.2011.6040613.

[7] BOURNS (Org.). Non-Contacting Steering Angle Sensor Type 6002. 2012. Disponível em: < http://www.bourns.com/docs/automotive/datasheets/non-contact-steering-angle-sensor-type-6002.pdf?sfvrsn=16>. Acesso em: 26 nov. 2016.

[8] KISTLER (Org.). Kistler MSW Sensors: Universal Measurement Steering Wheels. 2013. Disponível em: < http://www.techtarget.com.br/portal/sites/default/files/CMSWB_003-026e.pdf>. Acesso em: 14 dez. 2016.

[9] LEEN, Gabriel; HEFFERNAN, Donal; DUNNE, Alan. Digital networks in the automotive vehicle. Computing and Control Engineering Journal, v. 10, n. 6, p. 257-66, 1999.

[10] LUGLI, Alexandre Barantella; SANTOS, Max Mauro Dias. **Sistemas FIELDBUS para automação industrial:** DeviceNet, CANopen, SDS e Ethernet. Tatuapé: Erica, 2009.

[11] ARDUINO. **Compare Board Specs.** 2016. Disponível em: <https://www.arduino.cc/en/Products/Compare>. Acesso em: 05 nov. 2016.

[12] MICROCHIP TECHNOLOGIES (Org.). **MCP2515:** Stand-Alone CAN Controller With SPI Interface. 2005. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/21801d.pdf>. Acesso em: 27 set. 2016.

[13] CYPRESS PERFORM (Org.). **Serial Peripheral Interface (SPI) Slave.** Document Number: 001-65237, 34p**.** 2010. Disponível em: <http://www.cypress.com/file/132126/download>. Acesso em: 27 set. 2016.

[14] FARSI, M.; RATCLIFF, K.; BARBOSA, Manuel. An Overview of Controller Area Network. **Computing & Control Engineering Journal.** Stevenage, p. 113-120. jun. 1999.

[15] TANENBAUM, Andrew S.. **Redes de Computadores.** 4. ed. Amsterdam: Campus, 2010. 632 p.

[16] INTERNATIONAL STANDARDIZATION ORGANIZATION. **ISO 11898**: Road vehicles — Controller area network (CAN). Geneva: Iso, 2003

[17] RICHARDS, Pat (Arizona). Microchip. **A CAN Physical Layer Discussion.** Chandler: Microchip, 2002. 11 p. Disponível em: <http://ww1.microchip.com/downloads/en/AppNotes/00228a.pdf>. Acesso em: 02 set. 2016.

[18] JOHANSON, Mathias; KARLSSON, Lennart; RISCH, Tore. Relaying Controller Area Network Frames over Wireless Internetworks for Automotive Testing Applications. **2009 Fourth International Conference On Systems And Networks Communications,** [s.l.], p.1-5, set. 2009. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/icsnc.2009.68.

[19] CORRIGAN, Steve. **Introduction to the Controller Area Network.** Dallas: Texas Instruments, 2008. 15 p. Disponível em: <http://www.ti.com/lit/an/sloa101a/sloa101a.pdf>. Acesso em: 07 set. 2016.

[20] MICROCHIP TECHNOLOGIES (Org.). **Section 34:** Controller Area Network. 20011. Disponível em: < http://ww1.microchip.com/downloads/en/DeviceDoc/61154C.pdf>. Acesso em: 26 nov. 2016.

[21] Anand N, G. Joseph, S. S. Oommen and R. Dhanabal. **Design and implementation of a high speed Serial Peripheral Interface**. *Advances in Electrical Engineering (ICAEE), 2014 International Conference on*, Vellore, 2014, pp. 1-3.

[22] MARINKOVIC, Stevan J. et al. Nano-Power Wireless Wake-Up Receiver With Serial Peripheral Interface. **Ieee J. Select. Areas Commun.,** [s.l.], v. 29, n. 8, p.1641-1647, set. 2011. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/jsac.2011.110913.

[23] FOROUZAN, Behrouz A.. **Comunicação de Dados e Redes de Computadores.** 3. ed. Porto Alegre: Bookman, 2006.

[24] CYPRESS PERFORM (Org.). **Serial Peripheral Interface (SPI) Slave.** 2010. Disponível em: <http://www.cypress.com /file/132126/download>. Acesso em: 27 set. 2016.

[25] ATMEL CORPORATION (Estados Unidos). **ATMEGA2560:** Datasheet Complete. San Jose: Atmel, 2016. 444 p. Disponível em: <http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf>. Acesso em: 05 nov. 2016.

[26] NXP PHILIPS SEMICONDUCTORS (Org.). **TJA1050:** High Speed CAN Transceiver. 2003. Disponível em: <http://www.nxp.com/documents/data_sheet/TJA1050.pdf>. Acesso em: 29 set. 2016.

[27] KARLSEN, Henning. **Library:** UTFT. 2016. Disponível em: <http://www.rinkydinkelectronics.com/resource/UTFT/UTFT.pdf>. Acesso em: 28 nov. 2016.