UNIVERSIDADE DE CAXIAS DO SUL

WALTER SENGIK DA CRUZ

IMPLEMENTAÇÃO DO SISTEMA DE CONTROLE STEER BY WIRE ATRAVÉS DE MICROCONTROLADORES

CAXIAS DO SUL

2015

WALTER SENGIK DA CRUZ

IMPLEMENTAÇÃO DO SISTEMA DE CONTROLE STEER BY WIRE ATRAVÉS DE MICROCONTROLADORES

Trabalho de Conclusão de Curso de Engenharia de Controle e Automação, da Universidade de Caxias do Sul, apresentado como requisito final para obtenção do título de Engenheiro.

Orientador:
Prof. Dr. Fernando Augusto Bender

CAXIAS DO SUL

2015

WALTER SENGIK DA CRUZ

IMPLEMENTAÇÃO DO SISTEMA DE CONTROLE STEER BY WIRE ATRAVÉS DE MICROCONTROLADORES

Trabalho de Conclusão de Curso de Engenharia de Controle e Automação, da Universidade de Caxias do Sul, apresentado como requisito final para obtenção do título de Engenheiro.

Aprovado em 10/12/2015

Banca Examinadora

_____

Prof. Dr. Fernando Augusto Bender

Universidade de Caxias do Sul – UCS

_____

Prof. Dr. Maurício Zardo Oliveira

Universidade de Caxias do Sul – UCS

_____

Prof. Dr. Julio Cesar Ceballos Aya

Universidade de Caxias do Sul – UCS

RESUMO

Este trabalho consiste na elaboração de um algoritmo que simule o comportamento físico de um veículo com direção elétrica, o sistema *Steer by Wire*. A proposta é utilizar dois microcontroladores que se comuniquem em um barramento com o protocolo de comunicação CAN, sendo um microcontrolador programado para obter o controle PID (Proporcional Integral Derivativo) e sinal de direção e o outro programado para simular o modelo discreto dinâmico do veículo. Este trabalho é uma continuação da pesquisa denominada DRIVE, realizada em parceria com a Universidade de Caxias do Sul e a Universidade de Tecnologia da Malásia. Tal trabalho tem como principal objetivo implementar o modelo teórico proposto por Nor Shah e outros (2013) em uma rede de comunicação CAN física, somado ao controle e dinâmica do veículo, para verificar a robustez e comportamento físico deste sistema.

Palavras-chaves: direção elétrica, projeto DRIVE, microcontrolador; rede CAN.

ABSTRACT

This work is the development of an algorithm that simulates the physical behavior of a vehicle with electric steering, the system steer-by-wire. The proposal is to use two microcontrollers that communicate on a bus with the CAN communication protocol, and a programmed microcontroller for PID (Proportional Integral Derivative) control and drive signal and the other programmed to simulate the dynamic discrete model of the vehicle. This work is a continuation of research called DRIVE, held in partnership with the University of Caxias do Sul and the University of Technology Malaysia. Such work aims to implement the theoretical model proposed by Nor Shah and Others (2013) in a CAN network of physical communication, combined with the control and vehicle dynamics to verify the robustness and physical behavior of the system.

Keywords: steer by wire, project DRIVE, microcontroller, CAN network

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CAN	*Controller Area Network*
UCS	Universidade de Caxias do Sul
PID	Proporcional - Integral - Derivativo
CAN_H	*CAN High*
CAN_L	*CAN Low*
GND	*Ground (Terra)*
Kbp/s	Kilo bits por segundo
CC	Corrente contínua
SPI	*Serial Peripheral Interface*
USB	*Universal Serial Bus*

SUMÁRIO

1. INTRODUÇÃO

A área automotiva, desde seu surgimento, possui um constante crescimento tecnológico, tanto no desenvolvimento de soluções que não afetem o meio ambiente, como também soluções que podem aumentar a eficiência do motor, conforto e facilidade no uso diário do veículo automotor. Exemplos como redução de poluentes, mídias digitais embarcadas, frenagem ABS e aplicação da comunicação CAN podem ser considerados alguns dos avanços conquistados nestes últimos 40 anos do mercado automotivo.

Além dos exemplos citados anteriormente, também se destaca o sistema de direção assistida, conhecido por direção hidráulica, cujo objetivo é reduzir o esforço realizado pelo condutor no volante para realização de manobras que movimentem a as rodas dianteiras. Um sistema de conforto que praticamente entrou como série de linha nos automóveis atuais.

Com o advento destes avanços e a cada vez mais proporcionar uma melhor dirigibilidade para o condutor, surgiu a pesquisa, inicialmente desenvolvida pela Universidade de Tecnologia da Malásia, denominada DRIVE. Tal pesquisa tem por objetivo desenvolver um sistema de direção eletrônica (*Steer-by-Wire*) que substitua a direção mecânica ou direção assistida, aprimorando ainda mais a dirigibilidade do veículo e conforto do condutor.

Este trabalho é uma continuação do trabalho desenvolvido pelo Engenheiro Erich Hoffman (2014), no qual propôs a construção física de uma rede CAN para comportar o modelo apresentando no artigo de Nor Sha e Outros (2013). A proposta é utilizar o mesmo conjunto de microcontroladores do trabalho de Erich Hoffman e desenvolver uma nova rede CAN, para que possa comportar a comunicação entre dois microcontroladores que produzem a simulação da dinâmica e controle de direção do veículo. Desenvolvimentos como discretização do modelo matemático, programação na linguagem C, calibração do PID por métodos de controle e configuração física do barramento CAN se fazem necessários no desenvolvimento deste trabalho.

1.1. JUSTIFICATIVA DO TRABALHO

Com o desenvolvimento teórico proposto no artigo de Nor Sha e Outros (2013), e a proposta apresentada pelo Engenheiro Erich Hoffman (2014) em seu trabalho de graduação, se faz necessário a continuidade do projeto de pesquisa DRIVE. O projeto deve ser continuado a partir da implementação de uma nova rede CAN que faça a comunicação entre dois microcontroladores que simulam o comportamento da dinâmica do veículo, movimento da volante e aplicação do controlador PID, somada a todas as demais demandas de desenvolvimento. E assim, analisar quais são as necessidades técnicas e práticas para que o projeto possa ser continuado e aplicado em um veículo real.

1.2. OBJETIVOS

1.2.1. OBJETIVO GERAL

Desenvolver um algoritmo para ser aplicado em dois microcontroladores, em que se comunicam pelo barramento CAN e simulem a dinâmica do veículo, movimento do volante e o controlador PID.

1.2.2. OBJETIVOS ESPECÍFICOS

O presente trabalho tem por objetivos específicos os seguintes itens:

- Desenvolvimento de uma nova rede CAN baseada nas necessidades do sistema *Steer by Wire*, com a utilização dos kits *CAN BUS DEMO BOARD MCP2515DM-BM*;
- Avaliação da confiabilidade e robustez da nova rede CAN desenvolvida para comunicação entre dois microcontroladores;
- Discretização dos modelos dinâmicos do veículo para aplicação em microcontroladores.
- Entendimento do modelo de PID proposto no artigo de Nor Sha (2013) para aplicação no microcontrolador;
- Ajustes dos ganhos do controlador PID para que o modelo em malha fechada tenha a resposta de controle similar ao proposto no artigo de Nor Sha e Outros (2013).

1.3. ÁREA DE TRABALHO

O trabalho será executado no Centro de Ciências Exatas e Tecnologia, localizado no Bloco D da Universidade de Caxias do Sul, o centro é responsável por pesquisas e ensino das diversas ciências tecnológicas, entre elas automação e sistemas. A área de trabalho para este projeto é voltada a uma imersão nos princípios de programação em C e comunicação CAN. Além destas áreas de estudo, também será aprofundado o estudo nos princípios de modelagem de sistemas de controle contínuo e discreto.

1.4. LIMITES DO TRABALHO

Cabe ao autor:

- Desenvolver o algoritmo que faça a comunicação entre dois microcontroladores e simule a dinâmica do veículo e o controle PID, estipulados por Nor Sha e Outros (2013);
- Utilizar os kits eletrônicos *CAN BUS DEMO BOARD MCP2515DM-BM* para a aplicação do algoritmo desenvolvido.
- Utilizar os softwares *Canela*+, da empresa Thoreb, para a aquisição das mensagens CAN do barramento.

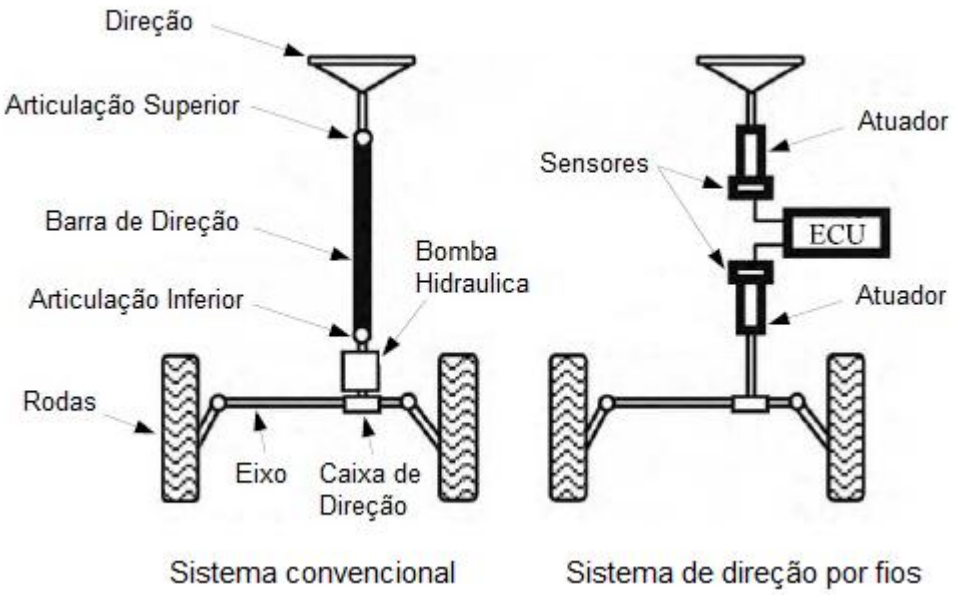
2. FUNDAMENTAÇÃO TEÓRICA

Com o intuito de fundamentar a metodologia a ser aplicada no decorrer deste projeto, será apresentado o modelo matemático dinâmico do veículo, sistemas no tempo discreto, controlador PID, conceitos de rede CAN, e funcionalidades do programa “Canela+” da empresa Thoreb.

2.1. SISTEMA STEER-BY-WIRE

Segundo Nor Shah e Outros (2013), o modelo do sistema *Steer-by-Wire* envolve três partes principais: volante no lado do condutor, caixa de direção das rodas dianteiras, e a dinâmica do veículo. A Figura 1 apresenta a proposta da estrutura do modelo para este projeto.

Figura 1 - Sistema *Steer by Wire*



Fonte: Nor Shah e Outros (2013). Modificada pelo autor (2015).

Com a abordagem na primeira estrutura do modelo *Steer by Wire*, o modelo da caixa de direção pode ser definido conforme pela equação 1. A caixa de direção consiste em um motor CC que gira um pinhão, acoplado junto à cremalheira por uma engrenagem. Com a movimentação do volante, as rodas também irão realizar o movimento em ângulo.

$$\begin{bmatrix}\dot{\theta}_r \\ \ddot{\theta}_r \\ \dot{\imath}_r\end{bmatrix} = \begin{bmatrix}0 & 1 & 0 \\ 0 & \frac{-b_r}{J_r} & \frac{\eta K_{tr}}{J_r} \\ 0 & \frac{-K_{er}}{L_r} & \frac{-R_r}{L_r}\end{bmatrix}\begin{bmatrix}\theta_r \\ \dot{\theta}_r \\ i_r\end{bmatrix} + \begin{bmatrix}0 \\ 0 \\ \frac{1}{L_r}\end{bmatrix} V_r + \begin{bmatrix}0 \\ \frac{-1}{J_r} \\ 0\end{bmatrix}\tau_a + \begin{bmatrix}0 \\ \frac{-1}{J_r} \\ 0\end{bmatrix}\tau_f \quad (1)$$

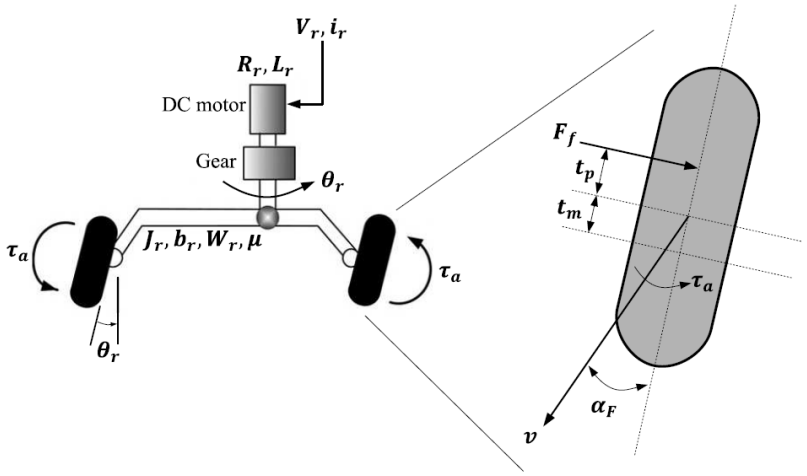
Onde θ_r é o ângulo de esterçamento dos pneus, τ_a é o torque de auto alinhamento dos pneus, τ_f é o torque de fricção, i_r é a corrente do motor CC, V_r é a tensão do motor CC, K_{tr} é a constante do torque do motor CC, K_{er} é a constante da força eletromagnética do motor CC, L_r é a indutância do motor CC, R_r é a resistência do motor CC, b_r é o coeficiente de amortecimento viscoso, J_r é o momento de inércia dos pneus e η é a taxa de esterçamento. Os torques de auto-alinhamento e de fricção são definidos pelas equações expressas abaixo:

$$\tau_a = -C_{\alpha F}\alpha_F\left(t_p + t_m\right) \quad (2)$$

$$\tau_f = g t_p \mu W_f \, sgn\left(\dot{\theta}_r\right) \quad (3)$$

Onde $C_{\alpha F}$ é o coeficiente de curvas dos pneus dianteiros, α_F é o ângulo de derrapagem do pneu dianteiro, g é a aceleração da gravidade, t_p é trilha pneumática do pneu, t_m é a trilha mecânica do pneu, W_f é o peso do pneu dianteiro, μ é o coeficiente de fricção e $sgn()$ representa a função sinal. Para melhor compreensão das equações 1, 2 e 3, a Figura 2 apresenta a dinâmica da caixa de direção, assumindo que W_f é igual a W_r.

Figura 2 - Dinâmica da Caixa de Direção



Fonte: Nor Shah e Outros (2013).

Segundo Nor Shah (2013), para a dinâmica do veículo é necessário assumir algumas premissas para linearizar seu modelo matemático. São elas:

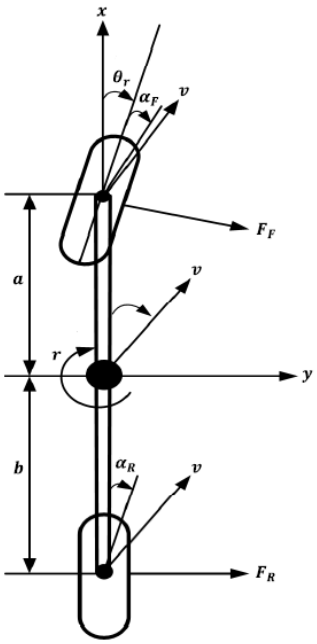
- As forças de fricção na direção x são desprezíveis quando o veículo não está freando.
- Os ângulos de esterçamento do lado esquerdo e direito são iguais.
- O veículo é simétrico.
- As forças de contato dos pneus dianteiros na direção longitudinal e lateral são idênticas para o pneu esquerdo para o pneu direito.
- As forças de contato dos pneus dianteiros no sentido longitudinal e lateral são idênticas tanto para o pneu esquerdo como para o pneu direito.

O modelo da dinâmica do veículo pode ser observado na equação 4, a qual fornece a variação do ângulo de derrapagem do pneu ($\dot{\beta}$) e a variação da taxa de guinada do veículo ($\dot{r}$). Estas duas variáveis podem ser também interpretadas na Figura 3.

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{-C_{\alpha F}-C_{\alpha R}}{mv} & -1+\frac{C_{\alpha R}b-C_{\alpha F}a}{mv^2}\\ \frac{C_{\alpha R}b-C_{\alpha F}a}{I_z} & \frac{-C_{\alpha R}a^2-C_{\alpha F}b^2}{I_z v}\end{bmatrix}\begin{bmatrix}\beta\\ r\end{bmatrix} + \begin{bmatrix}\frac{C_{\alpha F}}{mv}\\ \frac{C_{\alpha F}a}{I_z}\end{bmatrix}\theta_r \qquad (4)$$

Onde, $C_{\alpha F}$ é o coeficiente de curvas dos pneus dianteiros, $C_{\alpha R}$ é o coeficiente de curvas do pneu traseiro, a é a distância do pneu dianteiro em relação ao centro de gravidade do veículo, b é a distância do pneu traseiro em relação ao centro de gravidade do veículo, m é a massa do veículo, v é a velocidade longitudinal do veículo, I_z é momento de inércia do veículo.

Figura 3 - Modelo do veículo baseado em trajeto único



Fonte: Nor Shah e Outros (2013).

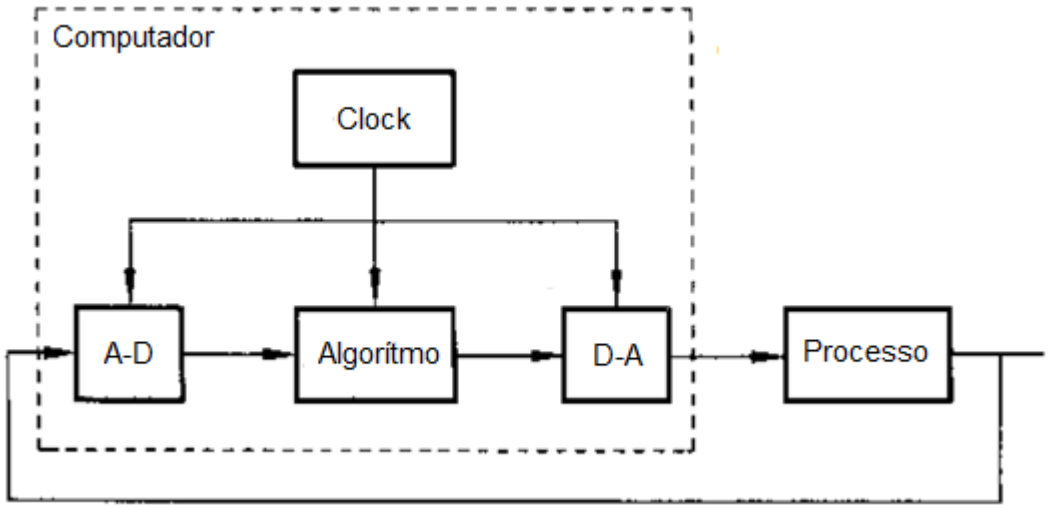
Por fim, o ângulo de derrapagem do pneu dianteiro (α_F) e o ângulo de derrapagem do pneu traseiro (α_R) podem ser obtidos pela equação 5, assumindo que estes ângulos são menores que 4°.

$$\begin{bmatrix}\alpha_F \\ \alpha_R\end{bmatrix} = \begin{bmatrix}1 & \frac{a}{v} \\ 1 & \frac{-b}{v}\end{bmatrix}\begin{bmatrix}\beta \\ r\end{bmatrix} + \begin{bmatrix}-1 \\ 0\end{bmatrix}\theta_r \tag{5}$$

2.2. SISTEMAS NO TEMPO DISCRETO

Os sistemas no tempo discreto são sistemas no qual as variáveis apenas irão mudar seus estados durante um instante específico. Este instante pode ser definido como *kh*, no qual *k* é o instante de amostragem e *h* é o período de amostragem. De acordo com K. J. Åström (1997, p.31), o período de amostragem é a propriedade fundamental de um sistema de controle discreto. Um exemplo de diagrama de um controle no tempo discreto aplicado em um processo no tempo contínuo pode ser visto na Figura 4.

Figura 4 - Diagrama esquemático de um sistema com controle discreto

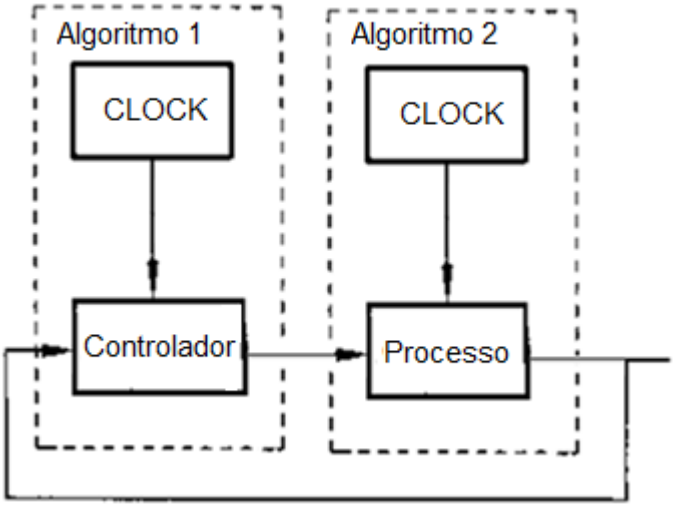


Fonte: K. J. Åström (1997, p.02). Modificada pelo autor (2015).

Na Figura 4 pode ser vista a aplicação de um conversor analógico digital (A/D) para transferir a resposta do processo (planta) para o algoritmo (controle), o período de amostragem no algoritmo definido pelo *clock* e um conversor digital analógico (D/A) para transferir a resposta do algoritmo para o processo.

No desenvolvimento deste projeto, a proposta será utilizar tanto o controlador como o modelo dinâmico do veículo no tempo discreto. Para isto, será necessário discretizar o modelo dinâmico do veículo. O diagrama da Figura 5 apresenta o esquemático da proposta deste trabalho.

Figura 5 - Diagrama esquemático de um sistema com controle e planta discreto

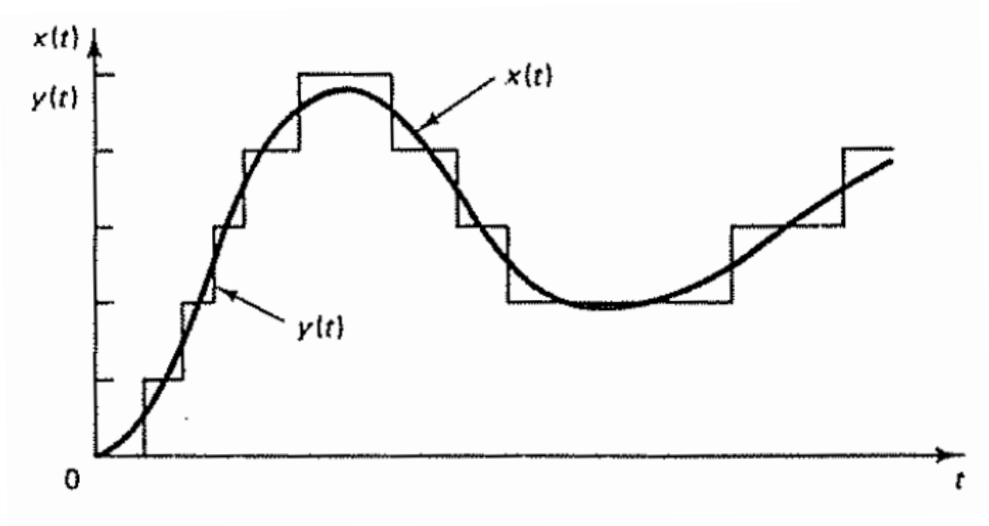


Fonte: Autor (2015).

2.2.1. AMPLITUDE DE QUANTIZAÇÃO

A amplitude de quantização é um sinal analógico representado por números finitos de um estado discreto. Esta definição pode ser compreendida na Figura 6, no qual representa um sinal analógico após a quantização, ou seja, limitação da resolução no eixo vertical (K. Ogata, 1995, p.08).

Figura 6 - Sinal analógico x(t) e saída discreta y(t)



Fonte: K. Ogata (1995, p.10). Modificada pelo autor (2015).

Para definir uma amplitude de quantização ideal para um sistema de controle discreto, é necessário que o projetista do algoritmo compreenda a relação entre o nível da quantização e o erro resultante (K. Ogata, 1995, p.09), pois o processo de quantização possui uma perda no sinal analógico. Conforme D. A. Ynoguti (2015), o importante é entender o quanto de informação pode ser descartada.

Em muitos casos, a amplitude de quantização pode ser definida simplesmente pela necessidade da resolução do sinal em determinada aplicação, como por exemplo, utilizar uma quantização de 8 bits para um sinal de 0 a 5V na indicação de um display digital. Se neste exemplo, o sinal é um indicador visual para um operador que necessita apenas verificar se este está entre 4V a 4,2V, uma resolução de 19,607mV (8bits) é suficiente para esta aplicação.

2.2.2. PERÍODO DE AMOSTRAGEM

Conforme K. J. Åström (1997, p.66), a seleção adequada do período de amostragem é essencial para um sistema de controle discreto. Aplicar um período

muito longo pode impossibilitar a reconstrução de um sinal no tempo contínuo. E caso aplicado um período muito curto, poderá gerar uma carga de processamento muito alta no microcontrolado, dificultando os cálculos internos. Devido a isto, deve ser necessário compreender corretamente as necessidades do processo de controle e limitações do *hardware*, a fim de proporcionar uma resposta satisfatória no sistema de controle.

Para um sistema de controle no tempo discreto, a quantidade de períodos de amostragem pode ser definida conforme a equação abaixo. Esta quantidade de períodos de amostragem representa o número de repetições de períodos do modelo discreto antes de chegar ao seu tempo de subida.

$$N_r = \frac{T_r}{h} \tag{6}$$

No qual, T_r é o tempo de subida do modelo dinâmico, e *h* é o período de amostragem estabelecida. Conforme define K. J. Åström (1997, p.66), o ideal é que a quantidade de períodos de amostragem deva ser entre 4 a 10. O valor definido neste trabalho será visto no Capítulo 3.2 (Discretização do modelo *Steer by Wire*).

2.2.3. DISCRETIZAÇÃO DE ESPAÇO DE ESTADOS

Espaço de estados é a representação de um modelo matemático de um sistema composto por um conjunto de varáveis de estado, de entrada e de saída (K. Ogata, 1995, p.294). Esta representação pode ser aplicada tanto no domínio de tempo contínuo como no discreto. Neste projeto, a proposta é utilizar as representações das equações apresentadas no Capítulo 2.1 (*Steer by Wire*) no domínio de tempo contínuo e transformá-las no domínio de tempo discreto com a técnica de discretização.

Para um sistema linear discreto e invariante no tempo, as equações de estados e saídas podem ser representadas conforme a equação apresentada abaixo (K. Ogata, 1995, p.295).

$$\begin{aligned} x(k+1) &= Gx(k) + Hu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned} \tag{7}$$

Onde, $x(k)$ é o vetor de estado, $y(k)$ é o vetor de saída $u(k)$ é o vetor de entrada, G é a matriz de estados, H é a matriz de entrada, C é a matriz de saída e D é a matriz de transmissão direta. Já para um sistema linear contínuo e variante no tempo, a equação de espaço de estados pode ser representada conforme a equação abaixo.

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(k) = Cx(t) + Du(t) \quad (8)$$

No qual A é a matriz de estados e B é a matriz de entrada. Para maiores informações sobre discretização de espaço de estados, consultar o livro "*Discrete-time Control Systems*" 2.ed., 1995, de K. Ogata, ou, "*Computer-Controlled Systems, Theory and Design*", 3.ed., 1997, de K. J. Åström.

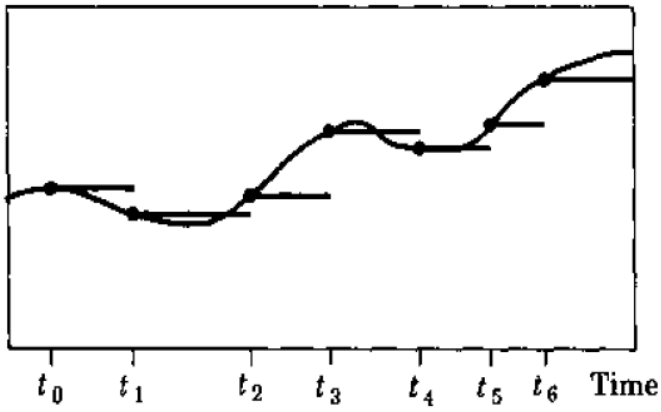
2.2.4. SEGURADOR DE ORDEM ZERO

O segurador de ordem zero, também conhecido por "ZOH", é uma técnica de discretização de modelos dinâmicos no tempo contínuo para o tempo discreto. A equação abaixo apresenta a metodologia (K. J. Åström,1997, p.247).

$$f(t) = f(t_k) \qquad t_k \leq t < t_{k+1} \quad (9)$$

Onde $f(t_k)$ é a equação no tempo discreto. Esta fórmula significa que a reconstrução do sinal para o tempo discreto é uma fração do sinal no tempo contínuo a cada repetição do período de amostragem. Este sinal fracionário será mantido como uma constante até que se complete o período de amostragem, possibilitando utilizar a próxima fração do sinal no tempo contínuo. Este embasamento pode ser melhor compreendido pela Figura 7.

Figura 7 - Reconstrução por segurador de ordem zero



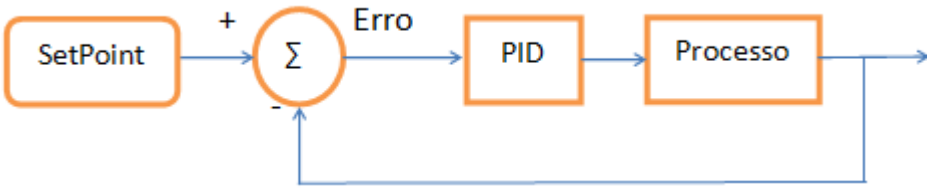
Fonte: K. J. Åström (1997, p.248).

2.3. CONTROLADOR PID

Controladores do tipo Proporcional, Integral e Derivativo, comumente denominados de PID, são controladores bastante difundidos no mercado industrial. Esta aceitação deve-se ao fato de ser um controlador facilmente implementável, com baixo custo e versatilidade. Com apenas três parâmetros, é possível condicionar o processo a uma vasta gama de comportamentos possíveis e desejados.

A aplicação do controlador PID na malha de controle de um processo pode ser representado de maneira simplificada como mostra a Figura 8.

Figura 8 - Diagrama de blocos de um sistema de controle com PID

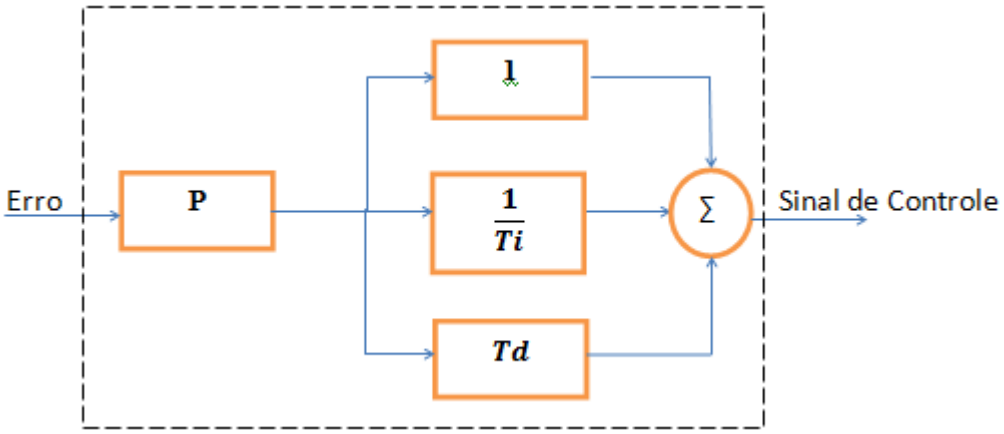


Fonte: Autor (2013).

Em geral controladores PID estão associados ao seguimento de referências, também chamadas de *SetPoint*. Exemplos de *SetPoints* são: temperatura de um forno em aquecimento, torque ou velocidade de giro de um eixo do motor, posição de um braço mecânico, etc.

No caso específico do controlador PID, seu sistema é composto por três tipos de ações de controle, conforme mostra a Figura 9.

Figura 9 - Controlador PID



Fonte: Autor (2013)

Onde, Erro é a diferença entre a referência e a realimentação do processo, P é o ganho proporcional, $\frac{1}{Ti}$ é o ganho integral, Td é o ganho derivativo e Sinal de Controle é o sinal que irá ser aplicado na planta do sistema.

Em uma forma bem simplificada, o PID é composto por três ações, conforme apresentado na Tabela 1.

Tabela 1 - Descrição das ações do PID

P	Correção proporcional ao erro	Seu valor cresce junto com o erro.
I	Correção proporcional ao erro vezes o tempo	Utilizado para corrigir o erro acumulado ao longo do tempo.
D	Correção proporcional à taxa de variação do erro	Utilizado para corrigir a rápida variação do erro.

Fonte: Autor (2013)

2.3.1. PID DISCRETO

De acordo com o K. J. Åström (1995, p.95), as ações do controlador PID discreto podem ser representadas conforme as equações a seguir:

$$P(t_k) = K_p e(t_k) \tag{10}$$

$$I(t_{k+1}) = I(t_k) + \frac{K_p h}{T_i} e(t_k) \tag{11}$$

$$D(t_k) = \frac{T_d}{T_d + Nh} D(t_{k-1}) - \frac{K_p T_d N}{T_d + Nh} \left(y(t_k) - y(t_{k-1})\right) \tag{12}$$

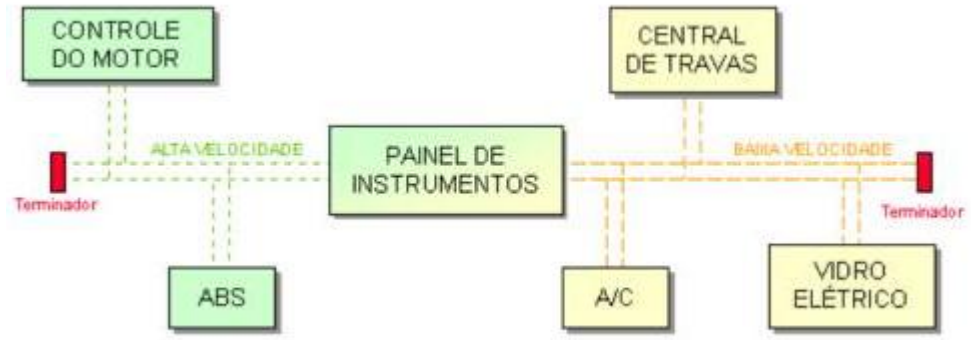
No qual, a equação 10 representa a ação proporcional e, K_p é o ganho proporcional aplicado e $e(t_k)$ é a diferença entre a referência e a realimentação do processo; a equação 11 representa a ação integral e, $I(t_k)$ é o resultado da ação atual integral atual, h é a taxa de amostragem e T_i é o ganho integral aplicado; a equação 12 representa a ação derivativa e, $D(t_{k-1})$ é o resultado da ação derivativa anterior, N é a taxa de amortecimento da ação derivativa, T_d é o ganho derivativo aplicado e $\left(y(t_k) - y(t_{k-1})\right)$ é a diferença entre a resposta atual e a resposta anterior do modelo dinâmico controlado.

As equações 10, 11 e 12 representam o mesmo PID aplicado por Nor Shah e Outros (2013).

2.4. REDE CAN

Conforme Guimarães (2002), o CAN é um protocolo de comunicação serial síncrono. Tal sincronismo é feito pela relação do início de cada mensagem enviada no barramento entre os módulos. O conceito de trabalho é definido como multimestre, onde todos os módulos que estão conectados entre si podem tornar-se mestres ou escravos, na medida em que estes necessitam. A Figura 10 ilustra um exemplo de aplicação física do barramento CAN.

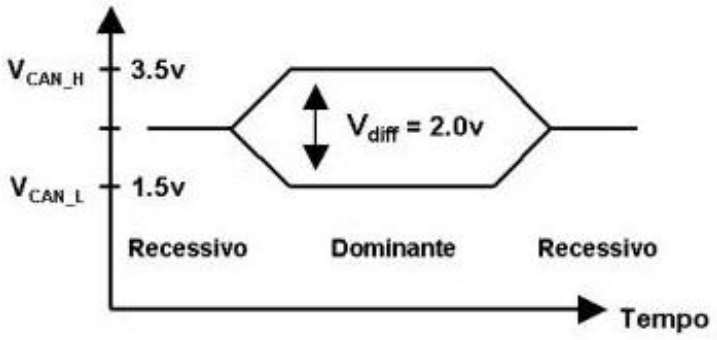
Figura 10 - Exemplo de aplicação CAN



Fonte: Guimarães (2002).

Na linha CAN, o sinal transmitido é definido pela diferença de potencial entre duas linhas (fios), o que proporciona uma melhor eficiência contra ruídos externos que não são característicos da transmissão. Estas duas linhas são definidas como CAN_H (CAN *High*) e CAN_L (CAN *Low*). O sinal “0” equivale ao sinal CAN_H, enquanto o sinal “1” corresponde ao sinal CAN_L (Universidade de Porto, 2015: p.52). A Figura 11 demonstra a diferença de potencial entre os dois sinais, formando o sinal da linha CAN.

Figura 11 - Níveis de tensão linha CAN

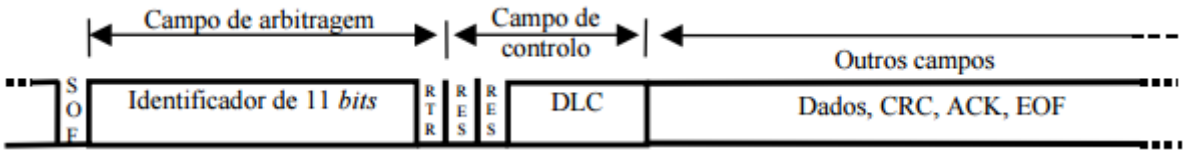


Fonte: Guimarães (2002).

Em relação à estrutura do pacote das mensagens na linha CAN, existem dois comumente aplicados, o CAN 2.0A e o CAN 2.0B. Por definição, o que diferencia estes dois formatos de mensagem é o número de identificadores no pacote. Enquanto o CAN 2.0A possui 11 bits para os identificadores, podendo definir até 2048 mensagens no barramento CAN, o CAN 2.0B possui 29 bits em seu identificador, podendo definir até 537 milhões de mensagens na rede CAN

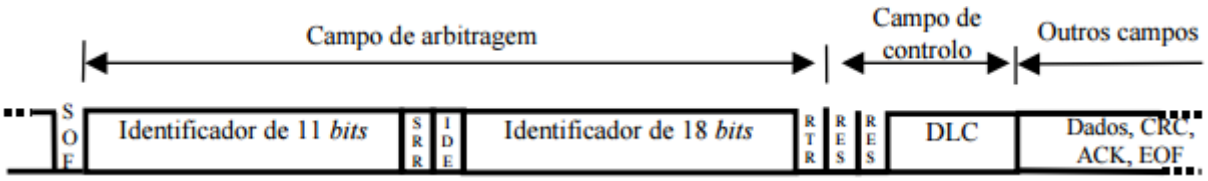
(Guimarães, 2002). As Figuras 12 e 13 apresentam o quadro de mensagens para estes dois formatos.

Figura 12 - Formato do pacote para CAN 2.0A



Fonte: Universidade de Porto (2015).

Figura 13 - Formato de pacote para CAN 2.0B



Fonte: Universidade de Porto (2015).

Outro fator importante que distingue os dois formatos de pacote é a velocidade de transmissão. Pelo formato CAN 2.0B possuir 18 *bits* a mais que o CAN 2.0A, o tempo de transmissão de todo o pacote acaba sendo maior, necessitando de um tempo de amostragem maior. Este tempo de amostragem pode ser calculado de acordo com as velocidades de comunicação aplicadas. Por recomendação (Universidade de Porto, 2015: p.49), existem 4 taxas de velocidades comumente utilizadas no mercado, são elas:

- 500 Kbit/s para distâncias de até 100 metros;
- 250 Kbit/s para distâncias de até 250 metros;
- 125 Kbit/s para distâncias de até 500 metros;
- 50 Kbit/s para distâncias até 1Km.

Para demonstrar o cálculo do tempo de amostragem para uma mensagem CAN, será assumido como exemplo uma velocidade de 250 Kbit/s (no qual será a velocidade utilizada neste trabalho) e um pacote no formato 2.0B utilizando os 8 *bytes* para dados. Conforme R. I. Davis e Outros (2007), a equação abaixo apresenta como pode ser calculado o tamanho máximo de bits em uma mensagem CAN.

$$C_m = g + 8s_m + 13 + \frac{g + 8s_m - 1}{4} \quad (13)$$

Onde g é uma constante de valor 34 para o formato CAN 2.0A (11 *bits* no identificador) ou 54 para o formato CAN 2.0B (29 *bits* no identificador) e s_m é o número de dados transmitidos (máximo 8 *bytes*).

Com este exemplo, 8 *bytes* de dados, a mensagem transmitida terá um tamanho máximo de 161 *bits* (arredondamento para um número inteiro superior). E com aplicação da divisão da quantidade de *bits* a ser transmitido pela velocidade estipulada para comunicação (250 Kbit/s), o tempo de amostragem mínimo que pode ser aplicado na mensagem será de 0,644 milissegundos.

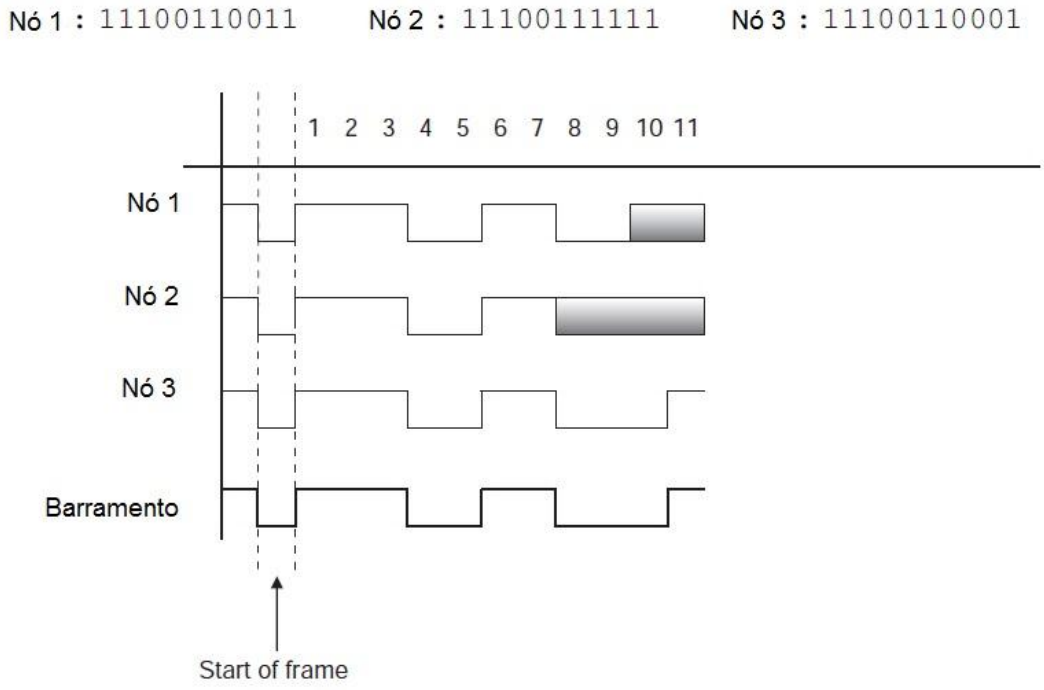
Este é um fator decisivo para não sobrecarregar o barramento de comunicação CAN e não ocasionar perdas nos pacotes enviados, conforme o Engenheiro Erich Hoffman já apresentou em seu trabalho de conclusão de curso (2014). Caso seja estipulado um período de amostragem de um milissegundo para cada mensagem, com a mesma estrutura da mensagem do exemplo acima, apenas uma mensagem poderia ser enviada, pois acima disto sobrecarregaria o barramento CAN.

2.4.1. IDENTIFICADOR

Os identificadores de uma mensagem CAN possuem duas principais funções. A primeira é estabelecer a prioridade entre cada mensagem transmitida no barramento. E a sua segunda é identificar a mensagem transmitida, para que a mesma possa ser utilizada apenas pelos módulos que necessitam da mesma (R. I. Davis e Outros, 2007).

A prioridade das mensagens é definida pelos primeiros *bits* transmitidos pelo identificador. A prioridade define qual será a mensagem que continuará sua transmissão caso haja colisão de envio entre mensagens. Esta é uma forte característica do barramento CAN, denominado como *CSMA/CD with NDA* (*Carrier Sense Multiple Acces/Collisoin Detection with Non-Destructive*). A Figura 14 ilustra a colisão entre três mensagens e qual delas é priorizada para continuar o envio.

Figura 14 - Detecção de colisão e prioridade na rede CAN



Fonte: E. Hoffman (2014).

No exemplo da Figura 14 é possível notar que a mensagem que continuará seu envio será a do “Nó 3”. No oitavo *bit* o “Nó 3” e “Nó 1” mudam seu estado para dominante, e o “Nó 2” mantém em recessivo, com isto, o “Nó 2” detecta a colisão e sua “não prioridade” e interrompe sua transmissão. Da mesma forma isto acontece no décimo *bit*, em que o “Nó 1” também interrompe sua transmissão. E com isto, o “Nó 3” continua a transmissão pela sua prioridade de envio, e os demais nós (Nó 1 e Nó 2) enviarão suas mensagens no próximo pacote de transmissão.

2.4.2. MENSAGENS

Em um barramento de comunicação CAN, o pacote de dados de uma mensagem é composto por até 8 *bytes*. Este pacote possui normalmente as informações adquiridas de um determinado sistema, tais como variáveis de controle, sensores, atuadores, sinais lógicos e entre outros. Neste trabalho, o pacote de dados será utilizado para transmitir dados de controle entre dois nós que constituem o modelo dinâmico do veículo, variação do volante e controlador PID.

O pacote de dados não necessariamente necessita utilizar os totais 8 *bytes*. Com a parametrização do DLC (*Data Lenght Code*), constituído no pacote da mensagem CAN, pode ser aplicado de 0 a 8 *bytes*. Por exemplo, com a aplicação de

um DLC com valor igual a 4, o número de *bytes* transmitidos na mensagem CAN será 4. Isto proporciona a otimização do tamanho da mensagem transmitida.

2.5. CANELA+

O *software* Canela+ é um programa desenvolvido para computador pela empresa Thoreb, com o propósito de realizar a leitura e escrita no barramento de comunicação CAN. Este *software* utiliza um *hardware*, denominado “TVI”, que tem por função aquisitar os dados do barramento CAN através das linhas CAN_H e CAN_L e enviar via comunicação USB para o computador. A Figura 15 demonstra o *hardware* TVI da Thoreb.

Figura 15 - *Hardware* de aquisição de dados

Fonte: Autor (2015).

No *software* Canela+ pode ser estipulada a velocidade de comunicação aplicada no barramento CAN, realizar leituras em tempo real das mensagens transmitidas, escrever mensagens no barramento CAN, analisar graficamente os *bytes* de mensagens, e exportar os dados de leitura em um bloco de notas. Este *software*, somado ao *hardware* de leitura, foi principalmente escolhido para aplicação neste projeto pela facilidade da exportação de dados em um arquivo em formato de texto (.txt) e pela doação que a empresa Thoreb realizou para o desenvolvimento deste trabalho.

3. DESENVOLVIMENTO

A solução proposta, com o desenvolvimento teórico e prático, é apresentada neste capítulo. Sendo este uma continuação de desenvolvimento do trabalho de conclusão de curso do Engenheiro Erich Hoffman (2014).

A primeira seção é uma análise do trabalho feito pelo Engenheiro Erich Hoffman (2014) e um *redesign* de deu desenvolvimento e resultados, a fim de aperfeiçoar o algoritmo desenvolvido para a comunicação CAN. Juntamente com a apresentação das novas metodologias utilizadas e funcionamento da rede CAN.

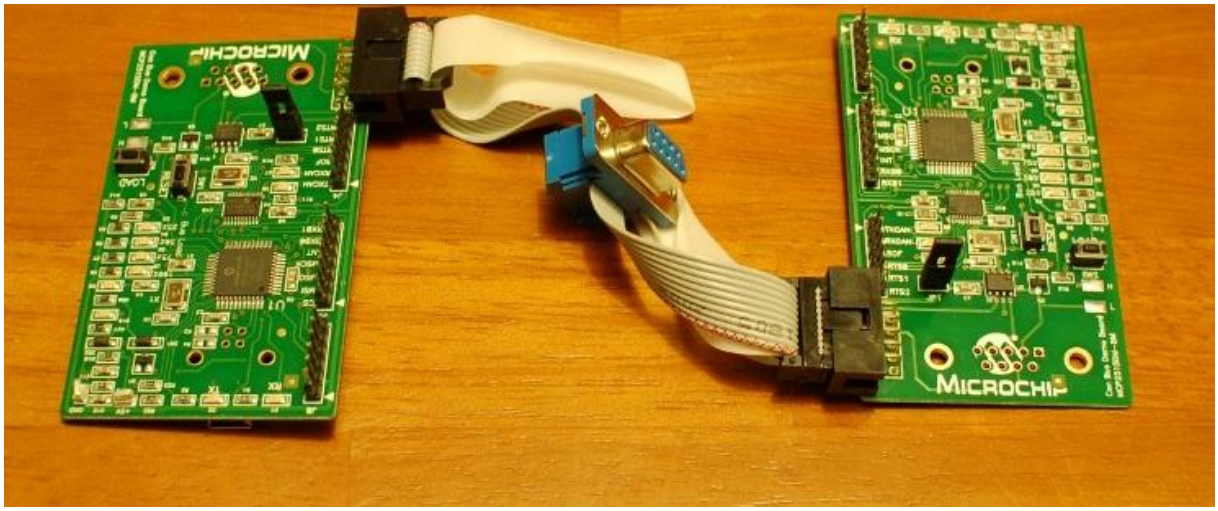
A segunda seção será destinada a discretização do modelo dinâmico do veículo proposto por Nor Shah e Outros (2013). Somada a apresentação das limitações exigidas para aplicação no microcontrolador e avaliações dos testes aplicados na rede CAN da variação do sinal de referência (variação do volante, θ_s) no modelo dinâmico do veículo.

E por fim, a terceira seção será a apresentação do modelo discreto do PID, com a aplicação dos mesmos parâmetros de controle utilizados por Nor Sha e Outros (2013) na rede CAN física, com a avaliação dos resultados. E também a nova configuração do controlador PID para obter uma resposta do modelo dinâmico do veículo com um controle mais eficaz, afim de que tenha uma resposta de controle similar com o proposto no artigo de Nor Shah e Outros (2013).

3.1. REDE CAN

Para o desenvolvimento da rede CAN no trabalho de conclusão de curso do Engenheiro Erich Hoffman (2014) foi proposta a utilização de quatro kits de comunicação CAN, a fim de criar o barramento de comunicação e verificar seu funcionamento entre os nós de comunicação. Estes kits também serão utilizados neste trabalho, visto que a proposta é a continuação do projeto e os mesmos atendem as necessidades de uso. A Figura 16 apresenta os kits de rede CAN utilizados e, para mais informações sobre os mesmos, pode ser consultado o Capítulo 3.2.1 do trabalho de conclusão de curso do Engenheiro Erich Hoffman (2014).

Figura 16 - Kit *Can Bus Demo Board* da Microchip



Fonte: E. Hoffman (2014)

As principais diferenças identificadas entre o trabalho do Engenheiro Erich e este proposto foi na configuração da comunicação CAN, algoritmo implementado para a comunicação e otimização dos kits utilizados. Abaixo será descrito estas diferenças e seus motivos para a aplicação.

3.1.1. KIT *CAN BUS DEMO BOARD*

O kit *Can Bus Demo Board MCP2515DM-BM* da Microchip possui um *software* de monitoramento do barramento CAN, denominado *Bus Monitor*. Tal *software* possui como principais funções a configuração de velocidade da rede CAN, leitura e escrita de mensagens e análise de sobrecarga e falha no barramento. Mas utilizar este *software* necessita que uma das placas eletrônicas do kit (demonstradas na Figura 16) tenha que ser utilizada para realizar a leitura do barramento CAN e transferir via USB para o *Bus Monitor*, não mais possibilitando que tal possa escrever mensagens de CAN específicas deste trabalho. Além disto, para utilizar este *software*, também é necessário utilizar como base de programação o algoritmo exemplo proposto com o kit. Este algoritmo pode ser encontrado no site da Microchip junto com as informações que descrevem o kit *Can Bus Demo Board*.

A proposta desenvolvida pelo Engenheiro Erich Hoffman (2014) foi utilizar a placa eletrônica do kit como interface entre o computador e o barramento CAN, juntamente com o *Bus Monitor*. A dificuldade encontrada neste ponto é que o *software Bus Monitor* não oferece a possibilidade de extrair os dados do barramento

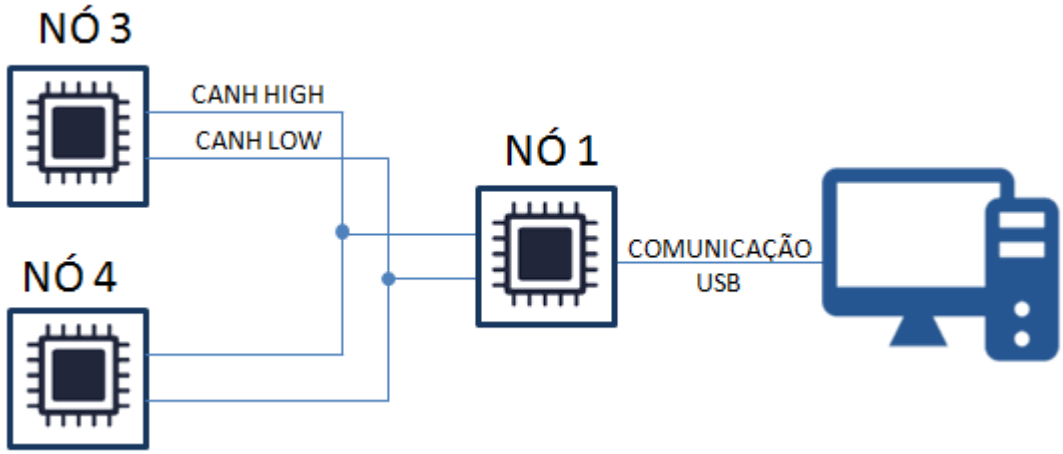
para uma análise posterior, apenas possibilita que as mensagens geradas no barramento fossem analisadas em tempo real. Isto impossibilita o diagnostico e progresso da aplicação do modelo dinâmico do veículo e o controlador PID, pois não seria possível gerar gráficos de análise. E devido a estas impossibilidades, optou-se por utilizar o *software* Canela+ para a extração das mensagens no barramento CAN, mas ainda com a utilização dos kits *Can Bus Demo Board*.

3.1.2. CANELA+ E BARRAMENTO CAN

Com a facilidade encontrada no trabalho de estágio desenvolvido do primeiro semestre de 2015 (Módulo Eletrônico de Controle Veicular para o Ônibus Hybridus Agrale) para a extração e diagnose de mensagens no barramento CAN, na utilização do *Software* Canela+ e o *hardware* TVI, optou-se por utilizar estes também neste projeto. Assim, as mensagens no barramento CAN podem ser extraídas para um documento de texto e posteriormente convertidas para aplicação em uma planilha com interpretação gráfica dos resultados.

Devida a utilização do Canela+, não se faz mais necessária a utilização do *software Bus Monitor*, e por conseguinte, o uso do algoritmo exemplo disponibilizado pela Microchip. A proposta do Engenheiro Erich Hoffman (2014), conforme demonstra a Figura 17, era utilizar 3 nós de comunicação para criar a dinâmica do veículo, mas com a nova solução, isto já não se faz necessário.

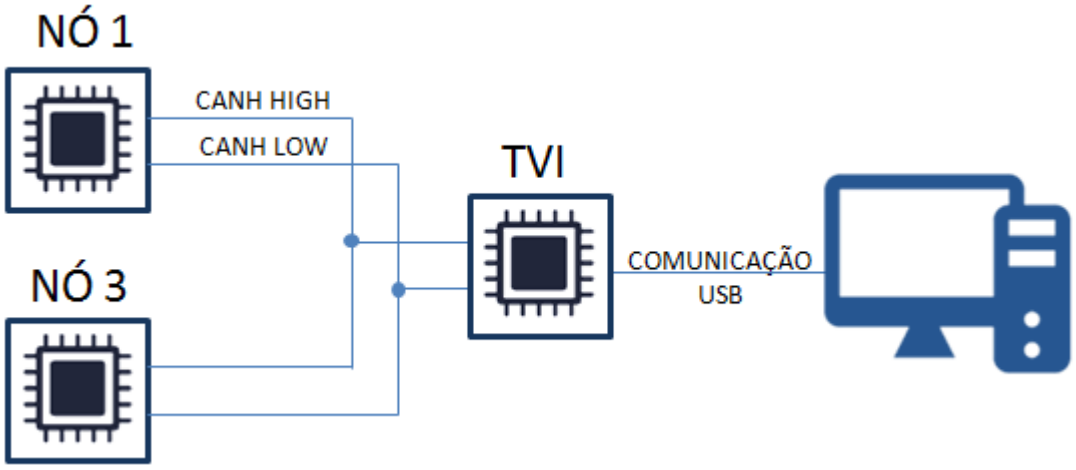
Figura 17 - Proposta barramento CAN do Engenheiro Erich Hoffman



Fonte: Autor (2015).

Na Figura 17, o Nó 1 é removido com a nova aplicação, sendo substituído pelo *hardware* TVI. O Nó 3 e 4 são destinados a dinâmica controlador do veículo. Mas para este trabalho, estes serão renomeados como Nó 1 e 3, conforme apresenta a Figura 18.

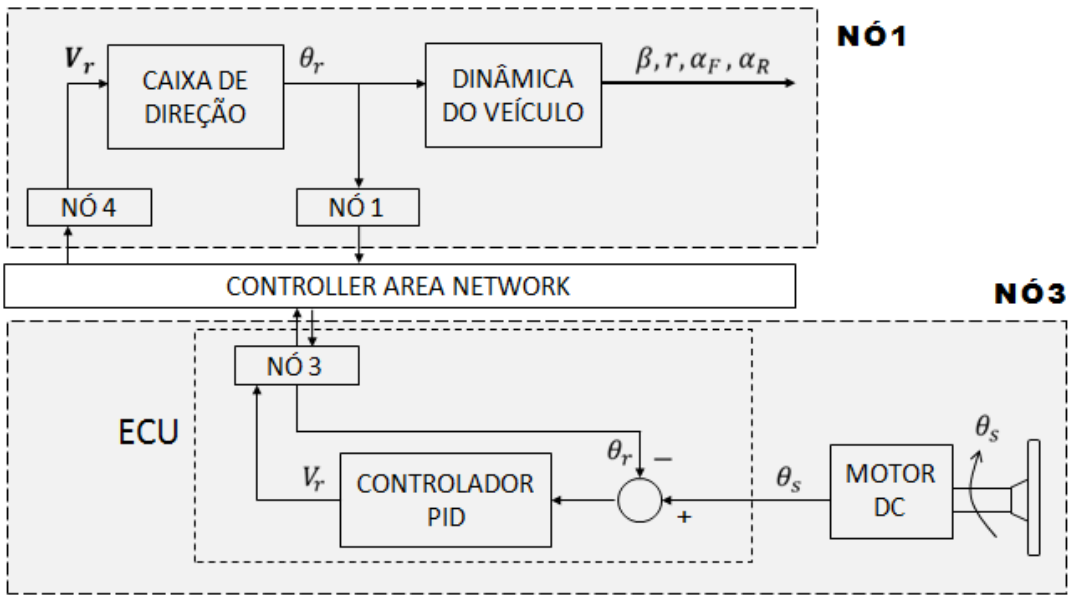
Figura 18 - Nova proposta do barramento CAN



Fonte: Autor (2015).

O objetivo de modificar a identificação dos nós foi para manter a mesma proposta apresentada no artigo de Nor Sha e Outros (2013). Esta proposta pode ser vista na Figura 19.

Figura 19 - Visão geral do sistema *Steer by Wire* no CAN



Fonte: Autor (2015).

É importante frisar sobre a Figura 19 que, não se faz necessário o Nó 4, pois o módulo (placa do kit *Can Bus Demo Board*) responsável pelo Nó 1, pode também ser responsável pela leitura que o Nó 4 realiza, não interferindo em qualquer momento nos resultados de controle e processamento.

Para facilitar a compreensão dos dois nós apresentados na Figura 18, será apresentada abaixo as mensagens CAN utilizadas e os processamentos realizados.

Tabela 2 - Responsabilidade e funções dos nós

Nó	**O que lê do barramento CAN**	**O que escreve do barramento CAN**	**O que processa no microcontrolador**
Nó 1	Mensagem que compõe o sinal da tensão do motor CC (V_r)	Mensagem que compõe o ângulo de esterçamento dos pneus (θ_r)	modelo dinâmico do veículo
Nó 2	Mensagem que compõe o ângulo de esterçamento dos pneus (θ_r)	Mensagem que compõe o sinal da tensão do motor CC (V_r), θ_s (referência) e contador	θ_s e controlador PID

Fonte: Autor (2015).

3.1.3. PARÂMETROS DO CAN

Antes de descrever o algoritmo que está implementado, é importante citar a definição de cada mensagem para a linha CAN e as configurações de velocidade e formato do pacote. Para as mensagens, serão utilizadas apenas duas no barramento, conforme apresentado na Tabela 2. Para estas mensagens, os identificadores utilizados são conforme demonstrados na definição de um trecho do código de programação abaixo (informação mais detalhada no Anexo A e B).

```
#define no1_ident 0x10  //modelo dinâmico veicular
#define no3_ident 0x20  //Referência + controlador PID
```

Conforme visto no Capítulo 2.4.1, a mensagem com o menor endereço no identificador é a que possui a maior prioridade de escrita no barramento. Por definição do artigo de Nor Sha e Outros (2013), foi estipulado que a mensagem de maior prioridade deva ser a que possui o modelo dinâmico veicular, com o endereço 0x10h. Visto que serão utilizadas apenas duas mensagens no barramento, a implementação do formato CAN 2.0A para o CAN já é suficiente.

Para o conteúdo de dados da mensagem CAN, seria necessário apenas o envio da variável θ_r e V_r para o funcionamento do trabalho proposto, mas para gerar os gráficos com os resultados finais, também será aplicada a variável de θ_s e o contador de θ_s para o Nó 3. Isto constitui duas mensagens CAN com o seguinte tamanho de dados:

- Mensagem do Nó 1: 2 *bytes* para a variável θ_r;
- Mensagem do Nó 3: 2 *bytes* para a variável V_r, 2 *bytes* para a variável θ_s e 2 *bytes* para a variável contadora.

O motivo da resolução de 2 *bytes* para cada variável será descrito no Capítulo 3.3. Por fim, o período de amostragem estipulado para as duas mensagens escritas no barramento CAN são de 8 milissegundos, conforme define Nor Shah e Outros (2013).

Com todas estas informações e a definição da velocidade de comunicação de 250 Kbit/s, pode ser feito o cálculo para verificar se o período de amostragem definido não será muito curto e terá que ser aumentada a velocidade de comunicação, pois se não, haverá sobrecarga no barramento CAN. Com a utilização da equação 13, e o resultado desta divido pela velocidade de comunicação, o período necessário para publicar o pacote de dados do Nó 1 é de 0,261 milissegundos e para o Nó 3 é de 0,461 milissegundos.

Já que está sendo publicada duas mensagens de 8 milissegundos no barramento CAN, a soma dos dois períodos calculados anteriormente pela equação 13 necessita ser menor que os 8 milissegundos. E está claramente atendendo, pois com uma análise do percentual de carga no barramento CAN, há apenas uma utilização de 9%: $((0{,}261ms + 0{,}461ms)/8ms)$. Com esta configuração, não haverá em nenhum momento perda de pacotes no barramento, pois a utilização é menos que 10% da carga máxima permitida.

3.1.4. BIBLIOTECA CAN PARA O ALGORITMO

Como já mencionado, não foi utilizado o algoritmo disponibilizado pela Microchip, e sim desenvolvido um novo com base em uma biblioteca CAN disponibilizada pela empresa *Custom Compiler Service* (CCS). Como não é objetivo deste trabalho detalhar cada linha de código do algoritmo para funcionamento da comunicação CAN, apenas será descrita as três principais funções utilizadas para a leitura e escrita de mensagens. Importante destacar que todo este trabalho será desenvolvido na linguagem de programação C. Também é importante lembrar que o responsável por manipular a leitura e escrita de mensagens no barramento CAN é o *transceiver* MCP2515, encontrado no kit *Can Bus Demo Board,* no qual o microcontrolador se comunica pelo protocolo SPI (*Serial Peripheral Interface*) com este.

Para a leitura da mensagem no barramento CAN são utilizadas duas funções. A primeira, denominada *can_kbhit()*, é responsável por verificar se há mensagem para ser lida pelo controlador. Caso sim, a mesma retorna um valor igual a “1”, e caso não há, irá retornar um valor igual a “0”. A segunda função, denominada *can_getd()*, é responsável por alocar os dados da mensagem lida em variáveis que podem ser utilizadas no código de programação. Abaixo será descrito um exemplo genérico destas duas funções em linguagem C.

```
if (can_kbhit())
{
   if(can_getd(identificador, &dados, qtd. dados, não usado))
   {
      if (identificador == constante do código)
      {
         // se identificador igual a constante estipulada,
         //irá entrar no código desta região comentada
      }
   }
}
```

Para a escrita da mensagem no barramento CAN é utilizada a função *can_putd()*, em que toda a vez que a mesma for produzida no código de programação, este irá solicitar a escrita da mensagem na linha CAN, com as informações conforme indicado no exemplo genérico a seguir:

```
can_putd(identificador, &dados, qtd. dados, prioridade, CAN2.0B, 0);
```

Mais informações sobre a leitura e escrita de mensagens, consultar os Anexo A e B.

3.2. DISCRETIZAÇÃO DO MODELO STEER BY WIRE

No Capítulo 2.1 foram apresentadas as equações que compõe o modelo dinâmico do veículo (*Steer by Wire*). Porém, antes de utilizar estas equações para a discretização, é necessário combiná-las em uma equação única de espaço de estados, conforme apresentado nas equações abaixo:

$$\dot{x} = A.x + B.u \tag{14.1}$$

$$x = \left[\beta\ , r\ \ , \theta_r\ , \dot{\theta}_r\ , i_r\ \right]^T \tag{14.2}$$

$$u = \left[V_r\ , \tau_f\right]^T \tag{14.3}$$

Onde, manipulando as equações 1, 2 e 4, os valores de A e B são os respectivos abaixo:

$$A = \begin{bmatrix} \frac{-C_{\alpha F} - C_{\alpha R}}{mv} & -1 + \frac{C_{\alpha R}b - C_{\alpha F}a}{mv^2} & \frac{C_{\alpha F}}{mv} & 0 & 0 \\ \frac{C_{\alpha R}b - C_{\alpha F}a}{I_z} & \frac{-C_{\alpha R}a^2 - C_{\alpha F}b^2}{I_z v} & \frac{C_{\alpha F}a}{I_z} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{-C_{\alpha F}(t_p + t_m)}{J_r} & \frac{-C_{\alpha F}(t_p + t_m)\frac{a}{v}}{J_r} & \frac{-C_{\alpha F}(t_p + t_m)}{J_r} & \frac{-b_r}{J_r} & \frac{\eta K_{tr}}{J_r} \\ 0 & 0 & 0 & \frac{-K_{er}}{L_r} & \frac{-R_r}{L_r} \end{bmatrix} \tag{14.4}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & \frac{-1}{J_r} \\ \frac{1}{L_r} & 0 \end{bmatrix} \tag{14.5}$$

Os valores para os parâmetros dos estados acima são dados por Nor Shah e outros (2013), informados na tabela abaixo:

Tabela 3 - Parâmetros do sistema *Steer by Wire*

$C_{\alpha F} = 2300\ \mathrm{N/rad}$	$b_r = 70\ N \times m \times s/rad$
$C_{\alpha R} = 4600\ \mathrm{N/rad}$	$W_f = 150\ Kg$
$m = 1961\ \mathrm{Kg}$	$\eta = 150$
$v = 5\ m/s$	$\mu = 0{,}192\ N \times m$
$a = 1{,}05\ m$	$g = 9{,}8\ m/s^2$
$b = 1{,}71\ m$	$K_{er} = 0{,}573\ V/rad/s$
$I_z = 3136\ Kg \times m^2$	$K_{tr} = 0{,}573\ V/rad/s$
$t_p = 0{,}0381\ m$	$Rr = 5.68\ \Omega$
$t_m = 0{,}04572\ m$	$Lr = 0.0203\ H$
$J_r = 3{,}5\ N \times m \times s^2/rad$	$V_r = -24\ a\ 24\ V$

Fonte: Nor Shah e outros (2013).

Com a substituição dos parâmetros da Tabela 3 nas equações 14.4 e 14.5, tem-se se:

$$A = \begin{bmatrix} -0{,}7037 & -0{,}8888 & 0{,}2346 & 0 & 0 \\ 1{,}7382 & -0{,}7524 & 0{,}7701 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0{,}550817 & 11{,}5672 & -55{,}0817 & -20 & 24{,}5571 \\ 0 & 0 & 0 & -0{,}1637 & -279{,}8030 \end{bmatrix} \quad (15.1)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -0{,}2857 \\ 49{,}2611 & 0 \end{bmatrix} \quad (15.2)$$

$$C_d = [0 \quad 0 \quad 1 \quad 0 \quad 0] \quad (15.3)$$

$$D_d = 0 \in \mathbb{R}^{5X2} \quad (15.4)$$

Antes de desenvolver o sinal de referência dinâmico (simulação da variação do volante) e aplicar no modelo dinâmico representado nas equações acima, é necessário a utilização da equação 3, que define τ_f. E com os parâmetros descritos

na Tabela 3 já é possível calcular a mesma, conforme apresentado nas equações abaixo.

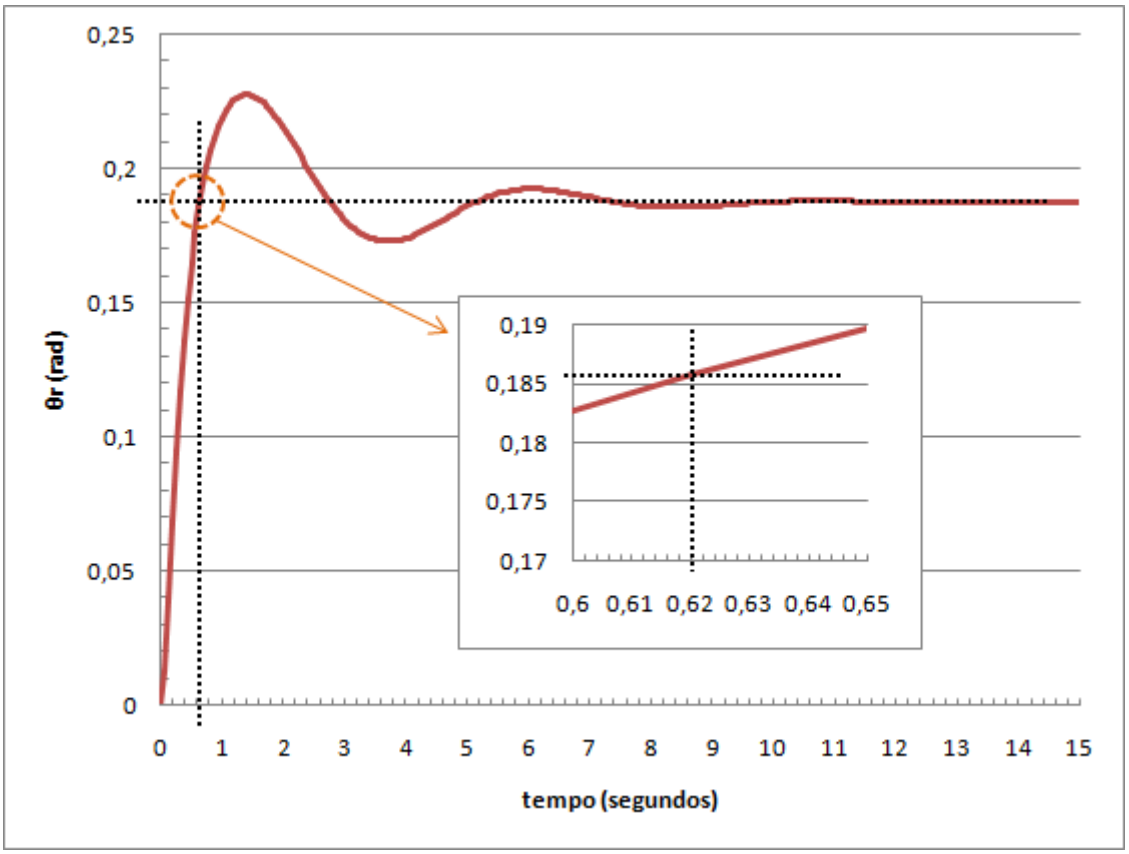
$$\tau_f = 9{,}8 \times\ 0{,}0381 \times 0{,}192 \times 150 sgn\left(\dot{\theta}_r\right) \tag{16.1}$$

$$\tau_f = 10{,}753344 \times sgn\left(\dot{\theta}_r\right) \tag{16.2}$$

Com as equações definidas, a resposta ao salto do modelo dinâmico pode ser realizada, e assim, verificar qual a tempo de subida do modelo contínuo para definir o período de amostragem para o modelo discreto.

Com as realizações de testes e avaliação da resposta ao salto para θ_r, identificou-se que o tempo de subida para o modelo dinâmico, possui um valor aproximado de 0,62 segundos. Isto pode ser verificado pelo gráfico na Figura 20, no qual foi aplicado um valor de 0,24V para V_r, equivalente a 10% da máxima tensão possível, e considerado o distúrbio $sgn\left(\dot{\theta}_r\right)$ igual a 0, assumindo que todos os estados no instante de tempo 0 são iguais a 0.

Figura 20 - Resposta ao salto de θ_r



Fonte: Autor (2015).

Com a utilização da equação 6, é possível definir o período de amostragem em 0,062 segundos, com um período de amostragem igual a 10, conforme define K. J. Åström (1997: p.66). Importante reforçar que o valor de 8 milissegundos que Nor Shah e Outros (2013) define é para o tempo de amostragem da mensagem CAN, e não para o tempo de amostragem do modelo dinâmico do veículo. Por fim, com o auxílio do Matlab, já é possível converter as equações contínuas em equações discretas, com a aplicação da técnica de ZOH apresentada no Capítulo 2.2.4, utilizando o seguinte comando:

```
>> c2d(ss(A,B,C,D),0.062,'zoh')
```

Equações discretas:

$$A = \begin{bmatrix} 0{,}9548 & -0{,}0526 & 0{,}0126 & 0{,}0003 & 0 \\ 0{,}1042 & 0{,}9518 & 0{,}0461 & 0{,}0010 & 0 \\ 0{,}0711 & 0{,}0134 & 0{,}9289 & 0{,}0343 & 0{,}0030 \\ 1{,}8630 & 0{,}3237 & -1{,}8630 & 0{,}2421 & 0{,}0235 \\ -0{,}0011 & -0{,}0002 & 0{,}0011 & -0{,}0002 & 0 \end{bmatrix} \quad (17.1)$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0{,}0051 & -0{,}0004 \\ 0{,}1442 & -0{,}0098 \\ 0{,}1760 & 0 \end{bmatrix} \quad (17.2)$$

$$C_d = [0 \quad 0 \quad 1 \quad 0 \quad 0] \quad (17.3)$$

$$D_d = 0 \in \mathbb{R}^{5X2} \quad (17.4)$$

Com as equações discretas do espaço de estados, as mesmas podem ser expandidas para serem aplicadas no microcontrolador, com base na teoria fundamentada da equação 7.

$$\beta[k+1] = 0{,}9548\beta[k] - 0{,}0526r[k] + 0{,}0126\theta_r[k] + 0{,}0003\dot{\theta}_r \quad (18.1)$$

$$r[k+1] = 0{,}1042\beta[k] + 0{,}9518r[k] + 0{,}0461\theta_r[k] + 0{,}0010\dot{\theta}_r \quad (18.2)$$

$$\theta_r[k+1] = 0{,}0711\beta[k] + 0{,}0134r[k] + 0{,}9289\theta_r[k] + 0{,}0343\dot{\theta}_r + 0{,}0030i_r + 0{,}0051V_r - 0{,}0043sgn(\dot{\theta}_r) \quad (18.3)$$

$$\dot{\theta}_r[k+1] = 1{,}8630\beta[k] + 0{,}3237r[k] - 1{,}8630\theta_r[k] + 0{,}2421\dot{\theta}_r + 0{,}0235i_r + 0{,}1442V_r - 0{,}1054sgn(\dot{\theta}_r) \quad (18.4)$$

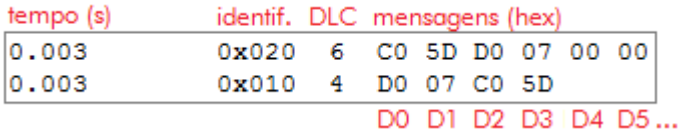
$$i_r[k+1] = -0{,}0011\beta[k] - 0{,}0002r[k] + 0{,}0011\theta_r[k] - 0{,}0002\dot{\theta}_r + 0{,}1760V_r \quad (18.5)$$

3.3. DEFINIÇÃO DA RESOLUÇÃO DOS SINAIS

Um ponto importante a ressaltar é a definição da resolução dos sinais aplicados neste trabalho, tanto para o modelo discreto quanto para o PID. Como já descrito, as três principais variáveis que são aplicadas na linha CAN são θ_r, V_r e θ_s. A linha CAN, por padrão, não tem possibilidade de aplicar sinais negativos ou

fracionários, apenas números inteiros, pois suas mensagens são definidas em hexadecimal. A Figura 21 ilustra uma captura de tela do *software* Canela+, no qual, *D0*, *D1*, *D2*, *D3*, *D4* até *D8*, representam os 8 *bytes* de dados da mensagem CAN.

Figura 21 - Formato de mensagens lidas do Canela+



Fonte: Autor (2015).

Como não é possível utilizar mensagens com valores negativos ou fracionários, foi-se adotado um novo *range* (escala e variação) para θ_r, V_r e θ_s. Estes podem ser vistos na Tabela 4.

Tabela 4 - *Range* para variáveis publicadas no barramento CAN

Variável	*Range conforme Nor Shah e Outros (2013)*	Novo *range* para aplicação do barramento CAN	Resolução do sinal
θ_r	Variação de -2 a 2 rad sem *offset*	Variação de -2000 a 2000 com *offset* de 2000	2/2000: 0,001 rad
V_r	Variação de -24 a 24 V sem *offset*	Variação de -24000 a 24000 com *offset* de 24000	24/24000: 0,001 V
θ_s	Variação de -2 a 2 rad sem *offset*	Variação de -2000 a 2000 com *offset* de 2000	2/2000: 0,001 rad

Fonte: Autor (2015).

Visto que, os valores das variáveis escritas no barramento CAN possuem valores de 0 a 4000 e de 0 a 48000, foi necessária a aplicação de 2 *bytes* para cada variável, constituindo uma variável de configuração "*long"* (16 *bits*). E a proposta do *offset* é para compensar os valores negativos representados na resposta de controle e do sistema dinâmico. Estes depois são subtraídos nos cálculos do PID e modelo dinâmico do veículo.

Também houve a multiplicação por 10000 das variáveis que constituem o modelo dinâmico (equações 18.1 a 18.5), para que quando forem aplicadas no microcontrolador, não haja a necessidade de trabalhar com números fracionários nos cálculos internos, evitando a utilização de variáveis de configuração "*float*"

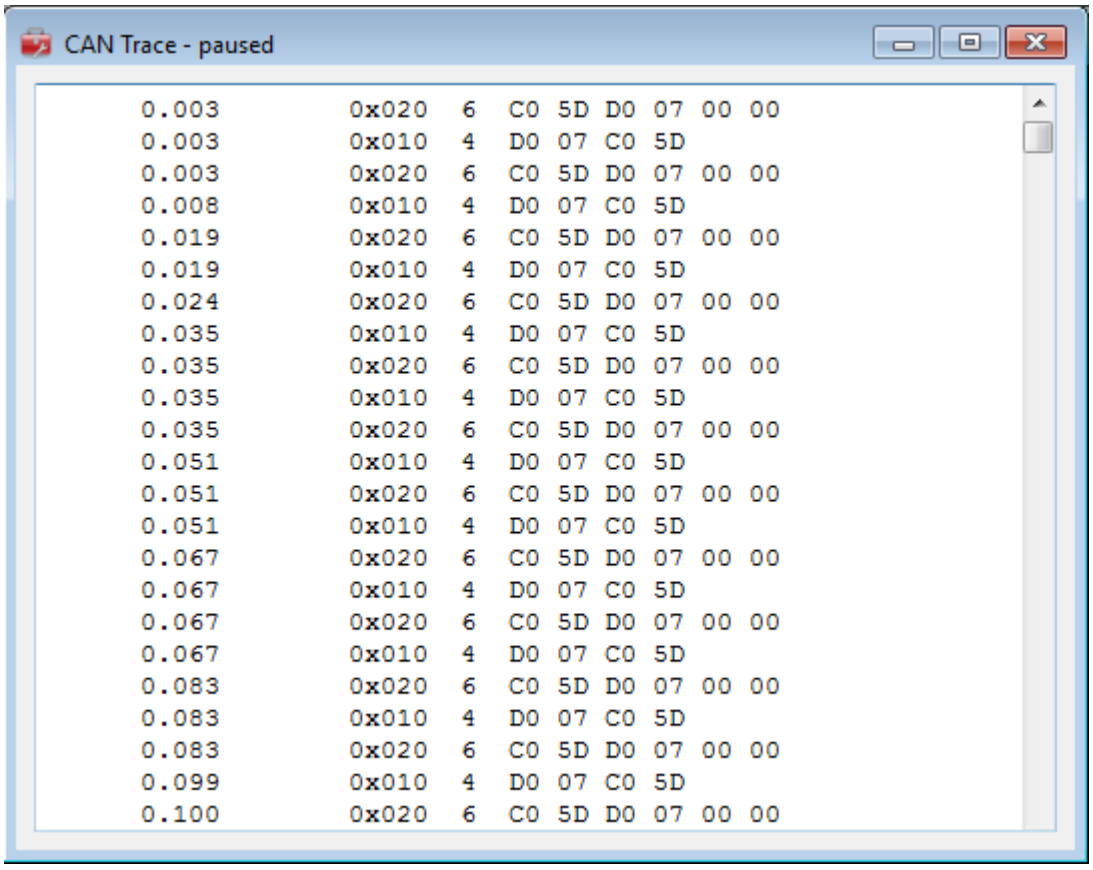
(float32), o que proporciona sobra de memória e processamento. Isto fixa o modelo dinâmico em uma resolução de cinco casas decimais, com a utilização de variáveis de configuração “*long long*” (int32).

Para maior conhecimento das manipulações matemáticas utilizadas nos algoritmos deste trabalho, consultar o Anexo A e B.

3.4. AQUISIÇÃO DE DADOS DO BARRAMENTO CAN

Como apresentado no Capítulo 2.5, um dos motivos do uso do *software* Canela+ é pela sua facilidade de aquisição de mensagens do barramento CAN. A Figura 22 ilustra como estas mensagens são publicadas na interface visual do Canela+.

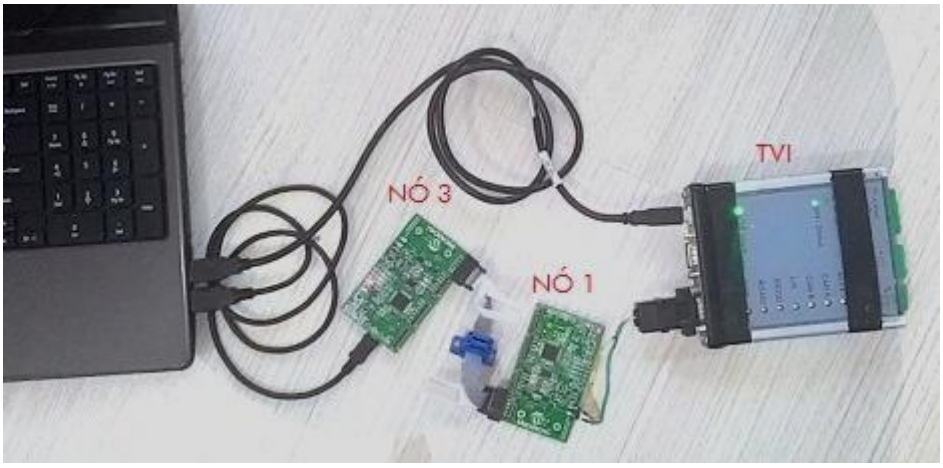
Figura 22 - Aquisição de mensagens CAN



Fonte: Autor (2015).

A Figura 23 apresenta os dois kits *Can Bus Demo Board* se comunicando pelo barramento CAN, e o *hardware* TVI realizando a aquisição de mensagens e enviando para o *software* Canela+.

Figura 23 - Estrutura física da comunicação CAN

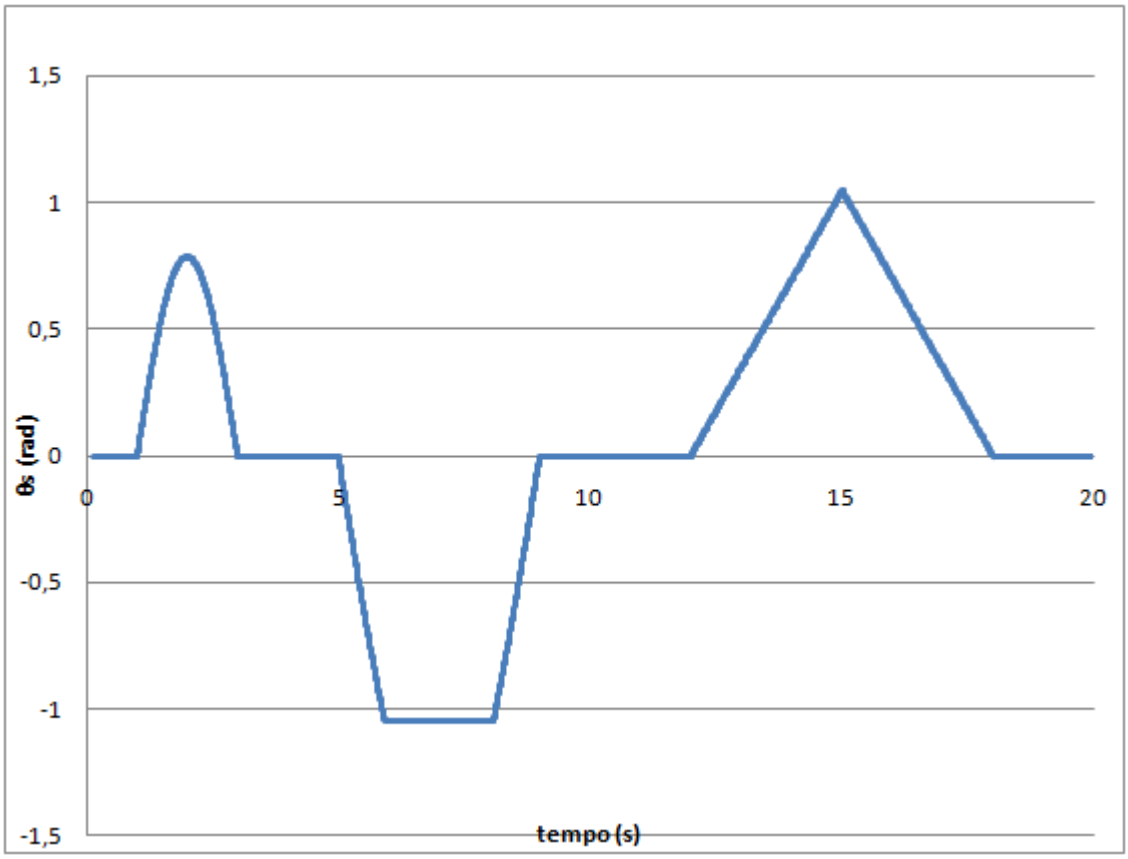


Fonte: Autor (2015).

Após aquisição das mensagens e publicação na interface, é possível exportar estas para um documento de texto. As informações neste documento são: tempo em segundos, identificador, DLC e dados. Tais informações são manipuladas para serem importadas para o *software* Excel (Microsoft Office), e formar um gráfico que representa a resposta do controlador, referência e modelo dinâmico do veículo no domínio tempo.

A Figura 24 apresenta a curva publicada da referência dinâmica (θ_s) no domínio tempo no barramento CAN, extraída para o Excel e criação do gráfico. O algoritmo que representa a resposta desta curva foi desenvolvido com as premissas estipuladas por Nor Shah e Outros (2013).

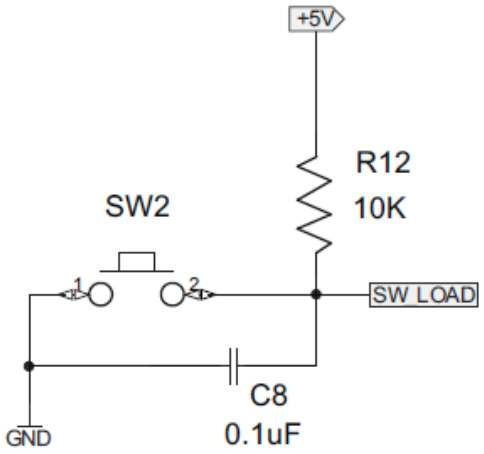
Figura 24 - Resposta do sinal de referência θ_s



Fonte: Autor (2015).

Em que θ_s apenas inicia sua variação pelo algoritmo após o acionamento de um botão ligado à entrada RA4 (Pino 21) do microcontrolador do Nó 3. Este circuito de acionamento pode ser visto na Figura 25, com a indicação do sinal como "*sw_load*".

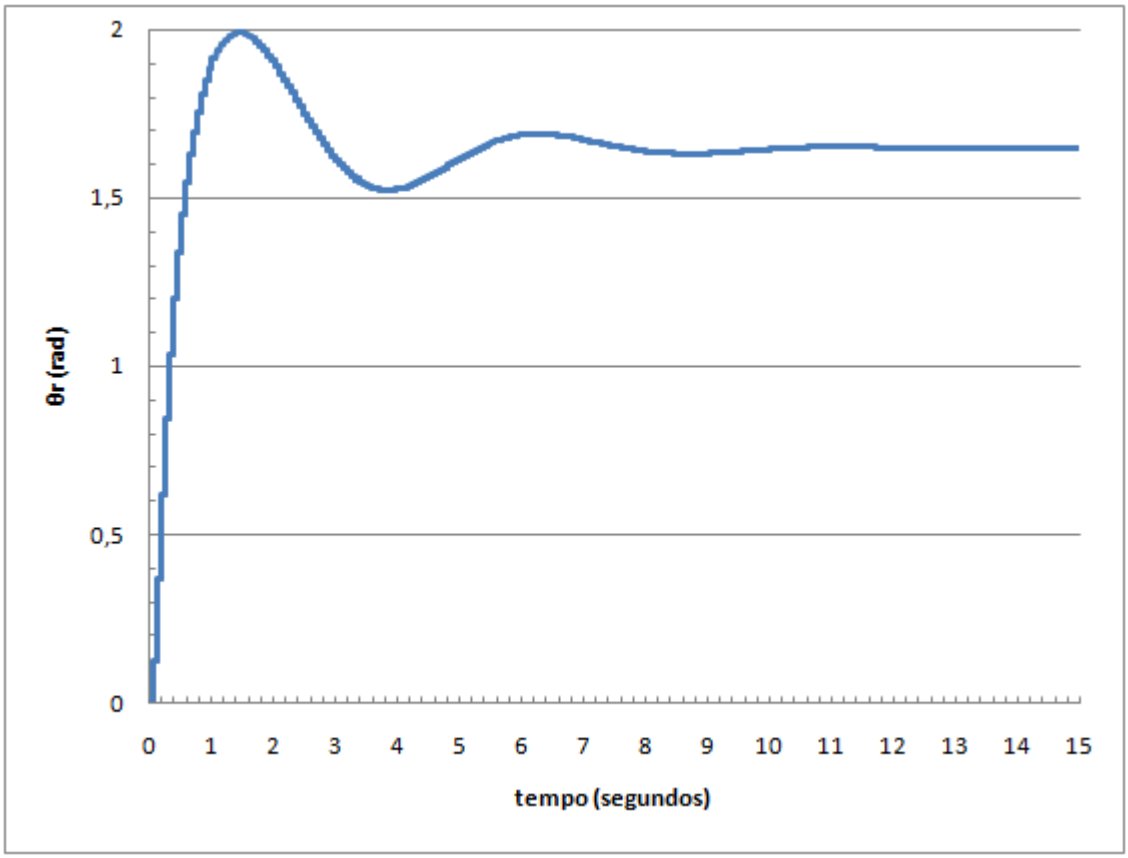
Figura 25 - Resposta do sinal de referência θ_s



Fonte: Microchip (2015).

Por fim, é possível avaliar pela Figura 26 a resposta ao salto do modelo dinâmico do veículo com uma aplicação de 24V em V_r. No qual demonstra a efetividade da publicação da mensagem do Nó 3 que contém V_r, a leitura desta mensagem pelo Nó 1 e a publicação de θ_r.

Figura 26 - Resposta ao salto de θ_r



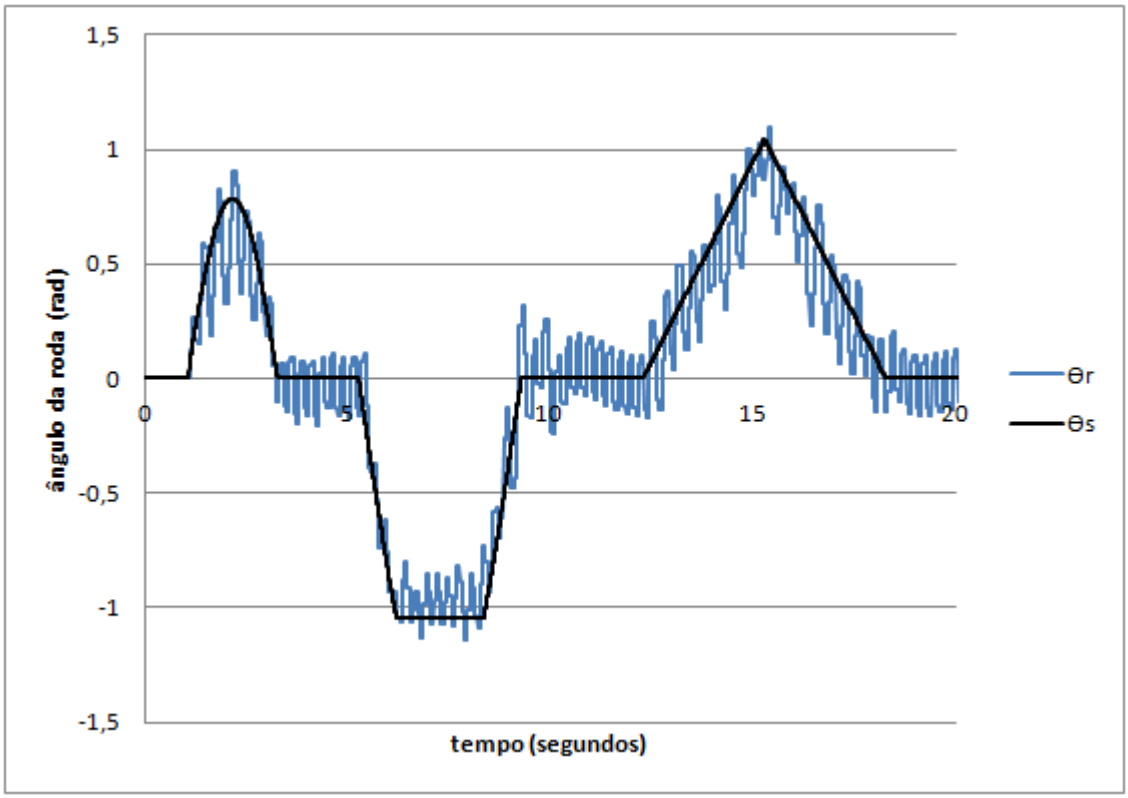
Fonte: Autor (2015).

3.5. CONTROLE

A proposta deste capítulo é apresentar as respostas do modelo dinâmico do veículo com o controlador PID. Inicialmente com a aplicação dos parâmetros assumidos por Nor Shah e Outros (2013), e posteriormente seguindo para uma nova calibração do PID.

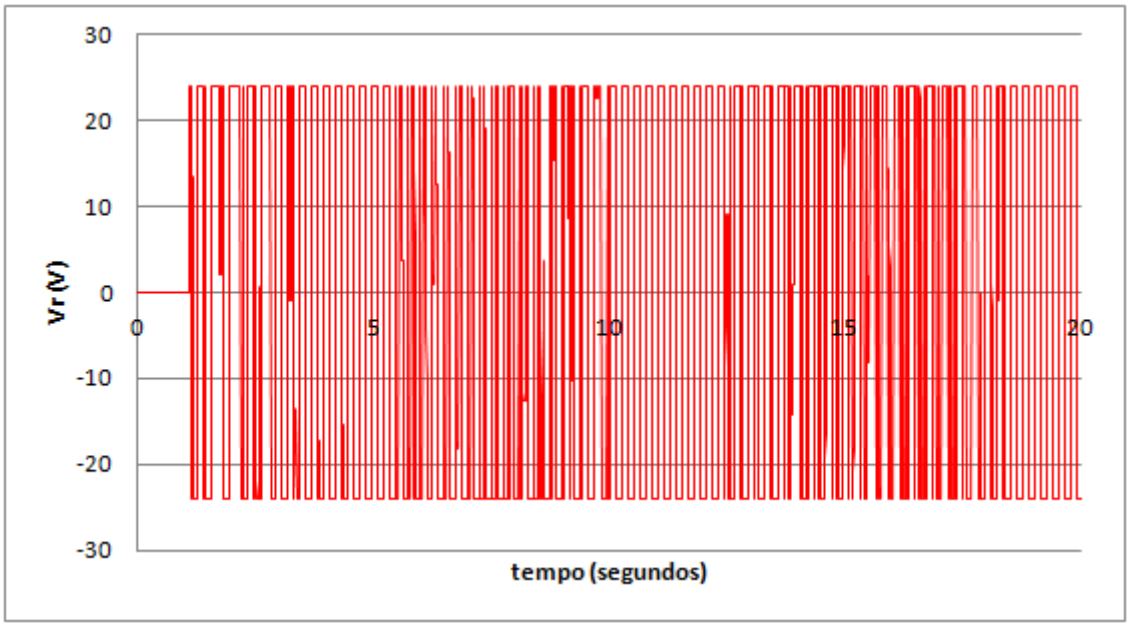
Nor Shah e Outros (2013) indica que os parâmetros ideias para o controlador PID com o modelo dinâmico veículo no tempo contínuo são $K_p = 906$, $T_d = 0{,}02$ e $N = 100$, sem uso da ação integral. Com a aplicação destes parâmetros, a resposta do sistema pode ser vista nas Figuras 27 e 28.

Figura 27 - Resposta de controle com parâmetros de Nor Shah e Outros (2013)



Fonte: Autor (2015).

Figura 28 - Resposta de V_r com parâmetros de Nor Sha e Outros (2013)



Fonte: Autor (2015).

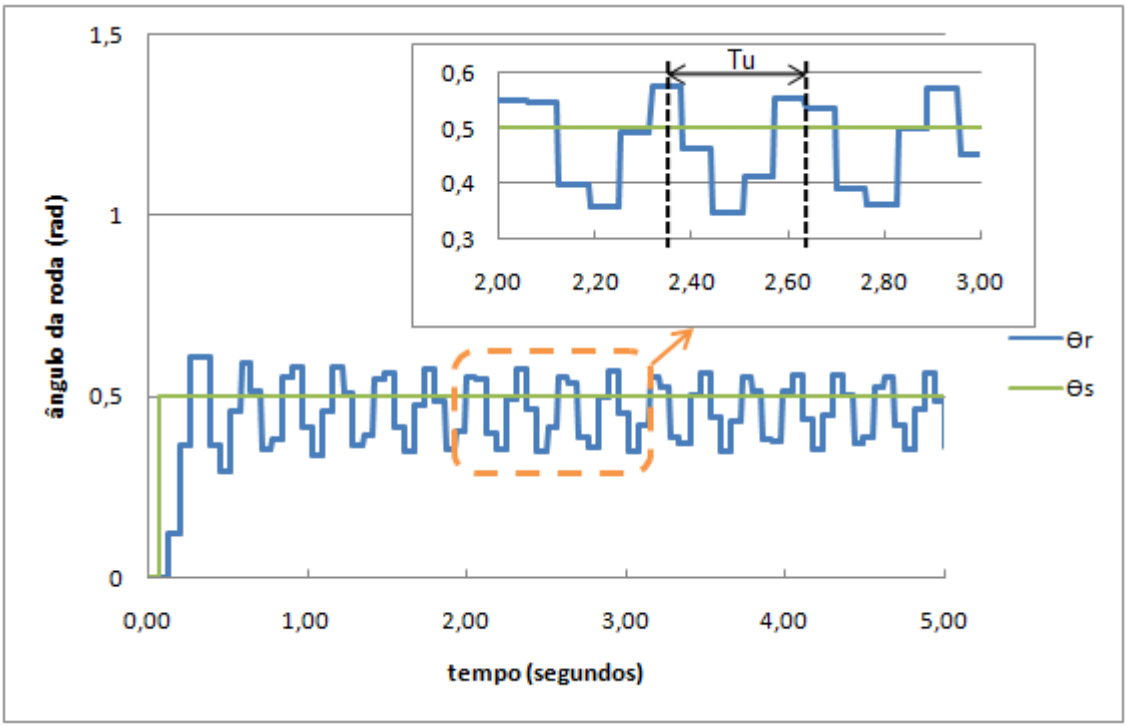
Nestes gráficos está claro que os valores assumidos para o PID no modelo contínuo não satisfazem na aplicação física do barramento, havendo a necessidade de uma nova parametrização dos ganhos do PID. A seguir será proposta uma nova calibração com base no segundo método de Ziegler-Nichols (ARAÚJO, 2007, p.42).

3.5.1. RESPOSTA EM FREQUÊNCIA DE ZIEGLER-NICHOLS

O método utilizado para a parametrização do PID será o resposta em frequência de Ziegler-Nichols, no qual de acordo com K. J. Åström (1995, p.136), é aplicado um valor no ganho proporcional (chamado de K_u), que torna o sistema marginalmente instável. Para isto, são assumidos ganhos iguais à zero para a ação integral e derivativa, ou seja, $T_i = \infty$ e $T_d = 0$. Isto torna a saída do modelo dinâmico com oscilações constantes.

A metodologia assume que o ganho proporcional deva ser aumentado até que a saída do sistema mantenha as oscilações. Visando isto, o ganho K_u encontrado para manter estas oscilações em um tempo considerável foi 170. A Figura 29 apresenta o comportamento oscilatório do sistema para este valor, assumindo um sinal de referência igual a 0,5 rad.

Figura 29 - θ_r marginalmente instável



Fonte: Autor (2015).

No gráfico da Figura 29 é possível verificar a indicação da constante T_u, definida como o período de oscilações do sistema. Pela análise gráfica, T_u possui um valor aproximado de 0,30. Esta constante é utilizada para determinar, junto com o K_u, os parâmetros do controlador PID, conforme indicado na Tabela 5.

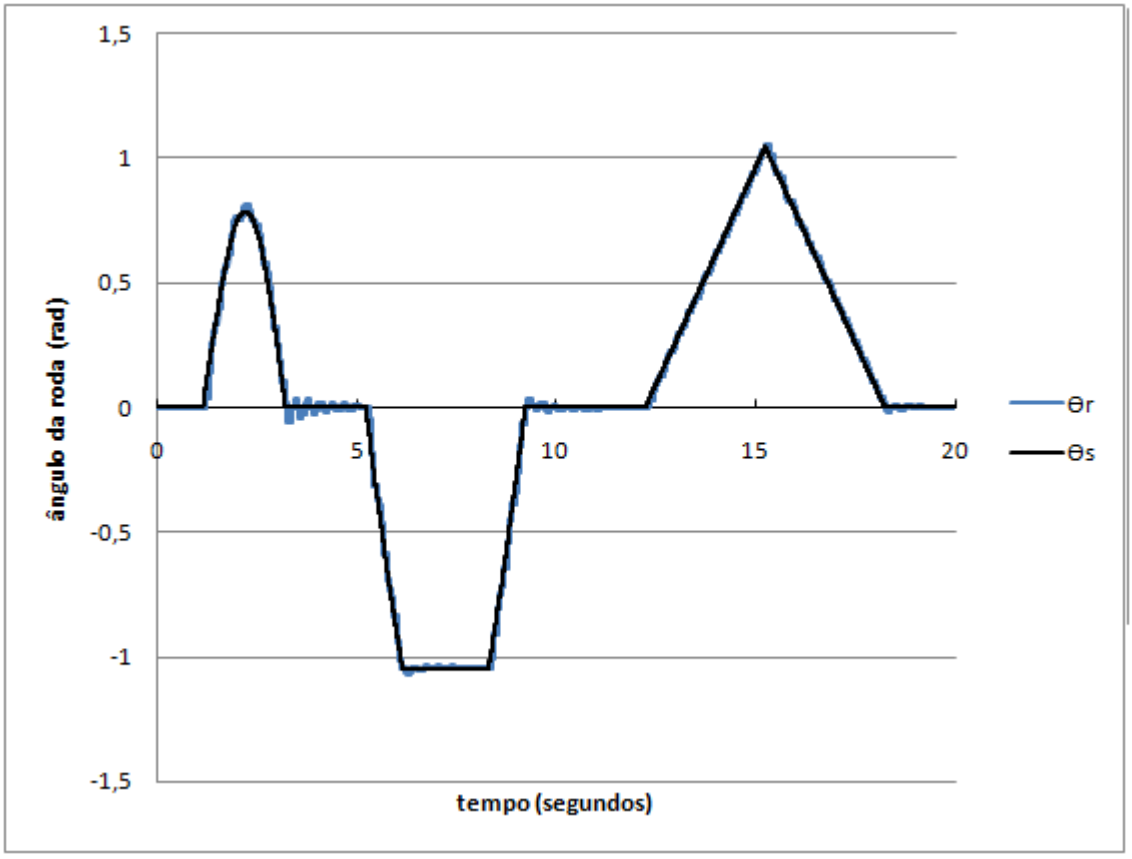
Tabela 5 - Definição dos parâmetros PID por Ziegler-Nichols

Controlador	K_p	T_i	T_d
PID	$0{,}6K_u$	$0{,}5T_u$	$0{,}125T_u$

Fonte: K. J. Åström (1995, p.136). Modificada pelo autor (2015).

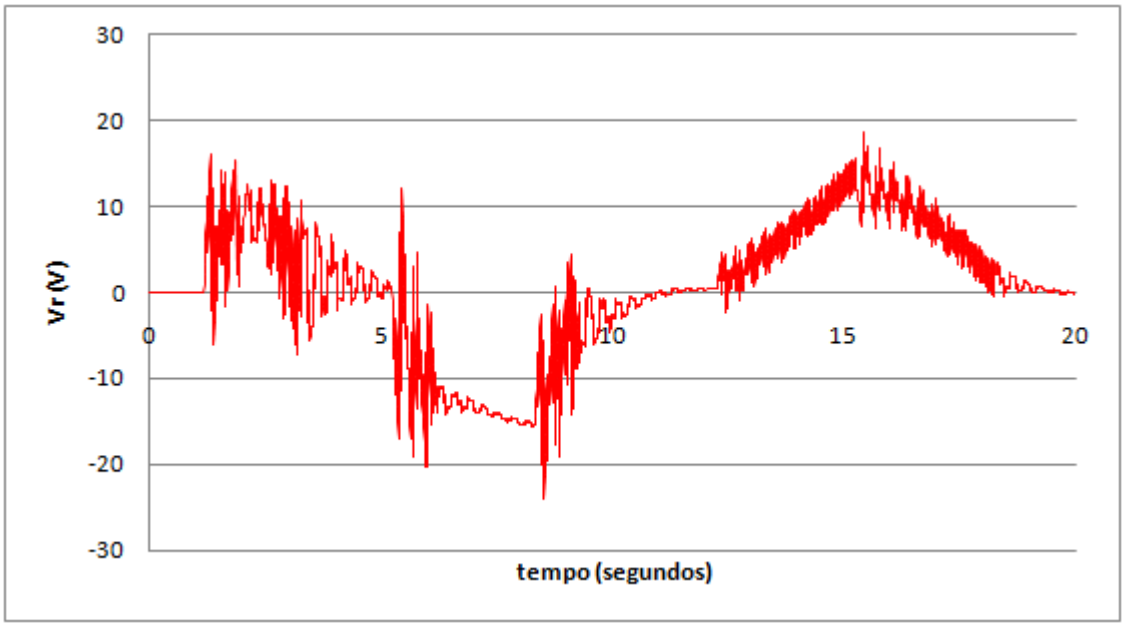
Assumindo o uso da Tabela 5, têm-se os seguintes valores para os parâmetros PID: $K_p = 102$ $T_i = 0{,}15$, $T_d = 0{,}0375$ e $N = 1$. Isto proporciona uma ação de controle mais eficaz, se comparada com o desempenho apresentado na Figura 27. As Figuras 30 e 31 apresentam a resposta do sistema com esta nova parametrização do controlador PID.

Figura 30 - Resposta de controle com parâmetros novos do PID



Fonte: Autor (2015).

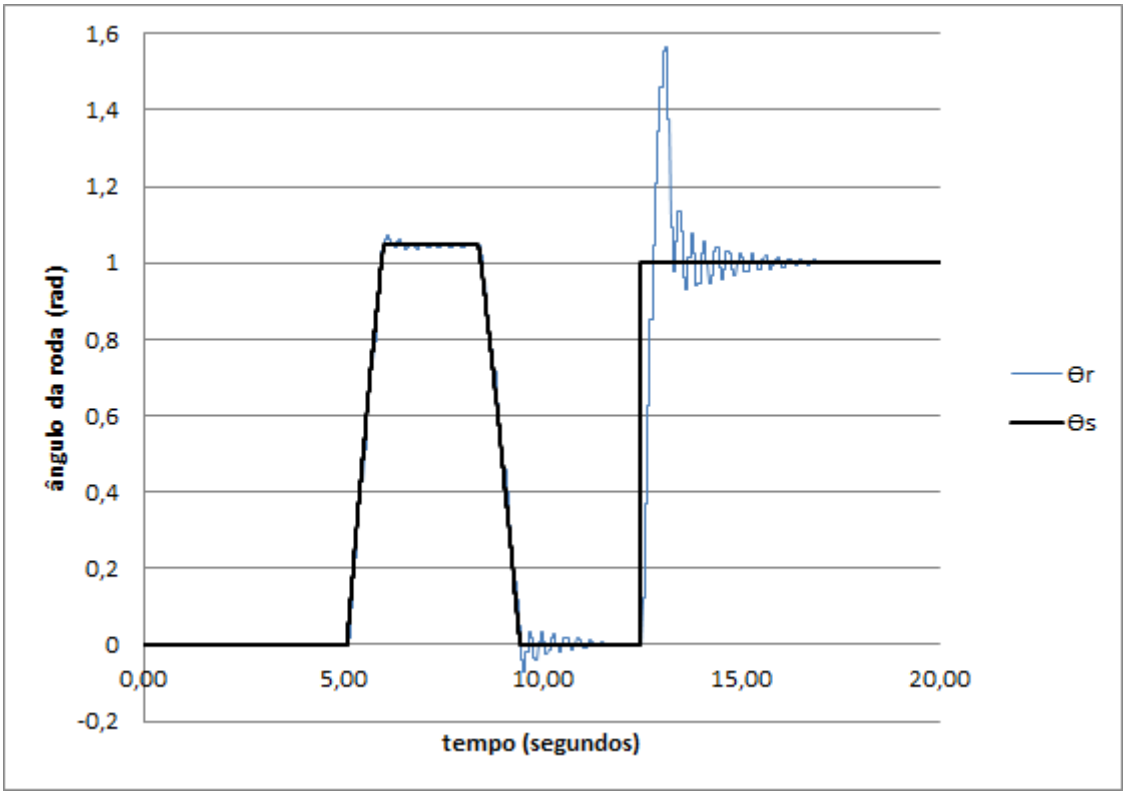
Figura 31 - Resposta de V_r com parâmetros novos do PID



Fonte: Autor (2015).

Esta nova parametrização dos valores do PID também podem ser analisada pela resposta em rampa e resposta ao salto apresentada na Figura 32. Esta figura apresenta que o PID ainda pode ser ajustado manualmente para obter uma melhor resposta ao salto. Este ajuste será deixado para trabalhos futuros, pois o objetivo principal deste trabalho foi alcançado, a aplicação física do barramento de comunicação CAN e comunicação entre o sistema de controle e modelo dinâmico.

Figura 32 – Nova resposta de controle com parâmetros novos do PID



Fonte: Autor (2015).

4. TRABALHOS FUTUROS

Visto que o desenvolvimento de um sistema de comunicação CAN físico que simule a dinâmica do veículo através de microcontroladores foi efetuado, atendendo os requisitos do trabalho proposto, o próximo passo a ser realizado é a construção física do modelo dinâmico veicular. Isto se define em desenvolver um protótipo de um sistema *Steer by Wire* em um veículo físico, onde haja um condutor para comandar o veículo em situações reais. Isto envolve alguns trabalhos futuros, que podem ser definidos como:

- Construção física do sistema *Steer by Wire* em um veículo real;
- Análise do novo modelo da dinâmica do veículo para uma aplicação física;
- Construção de um novo barramento físico de comunicação CAN
- Utilização do algoritmo do sistema de comunicação CAN e controlador PID estipulados neste trabalho;
- Recalibração dos parâmetros PID com base no novo modelo da dinâmica do veículo.
- Avaliações e testes para ajustes do controlador PID no novo sistema *Steer by Wire* com o veículo físico.

5. CONCLUSÃO

A proposta deste trabalho foi o desenvolvimento de um sistema físico de comunicação CAN que simule a dinâmica do veículo, variação do volante (referência, θ_r) e controlador PID através de microcontroladores, com base no modelo *Steer by Wire*.

A aplicação da rede CAN demonstrou-se funcional e robusta. Seu funcionamento com os kits, que representam os módulos do sistema *Steer by Wire*, atendeu as exigências descritas. Com resultados teóricos e práticos que apresentam a não necessidade de utilizar uma rede CAN com velocidade de 500 Kbp/s, podendo aplicar uma velocidade de 250 Kbp/s e ocupar apenas 9% de carga do barramento CAN.

Uma análise do modelo proposto por Nor Shah e Outros (2013) possibilitou a discretização do modelo dinâmico do veículo e sua aplicação no barramento CAN. Isto foi realizado através dos microcontroladores que processaram o PID discreto, a variação do volante e a resposta do modelo dinâmico discreto do veículo. Através da elaboração de gráficos, com parâmetros do PID estipulados por Nor Shah e Outros (2013), viu-se que a resposta de controle não é tão eficaz em um sistema de aplicação real, necessitando de uma nova calibração para o PID.

Com a utilização da metodologia de Ziegler-Nichols, com o objetivo de melhorar o desempenho do controle PID para o modelo discreto do veículo, foi realizada uma nova calibração dos ganhos proporcional, integral e derivativo. Isto possibilitou uma melhor resposta de controle, que poderá ser utilizada como referência para os próximos passos a serem realizados na pesquisa DRIVE.

Por fim, este foi um projeto desafiador, que aprofundou o conhecimento em diversas áreas da engenharia, almejando desde o início o cumprimento dos objetivos traçados.

REFERÊNCIA BIBLIOGRÁFICA

HOFFMAN, Erich; **Implementação de uma Rede CAN visando a simulação de um Sistema de Direção Eletrônica em Veículo Automotor (Steer-By-Wire System)**. Universidade de Caxias do Sul, Centro de Ciências Exatas e Tecnologia, 2014.

NOR SHAH, Mohd Badril; HUSAIN, Abdul Rashid; DAHALAN, Amira Sarayati Ahmad. ***An Analysis of CAN-based Steer-by-Wire System Performance in Vehicle***. IEEE *International Conference on Control System, Computing and Engineering*, Penang, Malásia, 2013.

ÅSTRÖM, Karl Johan; WITTENMARK, Björn. ***Computer- Controlled Systems, Theory and Design***. 3.ed. Universidade Tsinghua, Pequim, China. 1997. 557 p.

YNOGUTI, Carlos Alberto. **Processamento Digital de Sinais, Conversão A/D e D/A**. Instituto Nacional de Telecomunicações. Disponível em: <www.inatel.br/docentes/ynoguti/component/docman/.../16-digitalizacao> Acesso em 16 de Setembro de 2015.

OGATA, Katsuhiko. ***Discrete-Time Control Systems***. 2.ed. Universidade de Minnesota, Minneapolis, Estados Unidos. 1995. 744 p.

ÅSTRÖM, Karl Johan; HÄGGLUND, Tore. ***PID Controllers: Theory, Design, and Tuning***. 2.ed. *Instrument Society of America*, Estados Unidos. 1995. 343 p.

GUIMARÃES, Alexandre de Almeida; SARAIVA, Antônio Mauro. **O Protocolo CAN: Entendendo e Implementando uma Rede de Comunicação Serial de Dados baseada no Barramento "*Controller Area Network*"**. SAE 2002. Disponível em: <http://www.alexag.com.br/Artigos/SAE2002.pdf> Acesso em 05 de Setembro de 2015.

UNIVERSIDADE DE PORTO. **Protocolo de Comunicações CAN**. Disponível em <https://web.fe.up.pt/~ee99058/projecto/pdf/Can.pdf> Acesso em 06 de Setembro de 2015.

DAVIS, Robert I.; BURNS, Alan; BRIL, Reinder J.; LUKKIEN, Johan J. ***Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised***. *Springer Science + Business Media*, LLC 2007. 30 de Janeiro de 2007.

CRUZ, Walter Sengik. **Módulo Eletrônico de Controle Veicular para o ônibus Hybridus Agrale**. Universidade de Caxias do Sul, Centro de Ciências Exatas e Tecnologia, 2015.

MICROCHIP. **MCP2515 CAN Bus Monitor Demo Board User's Guide**. 2008 - 2014 Microchip Technology, USA, Inc 2005. Disponível em: <http://www.microchip.com/Developmenttools/ProductDetails.aspx?PartNO=MCP2515DM-BM> Acesso em 15 de Agosto de 2015.

ANEXO A – ALGORITMO PRINCIPAL 1 (MODELO DINÂMICO VEICULAR – NÓ 1)

```
//=====================================================//
//======= ESTE PROGRAMA TRABALHA COM O NÓ 1 ===========//
//==== o nó 1 faz a leitura do nó 3 e trabalha ========//
//== o nó 1 tem o modelo da dinâmica do veículo =======//
//=====================================================//
//AUTOR: Walter Sengik da Cruz (waltersengikcruz@gmail.com)

#include "18F4550_SbW.h"            //configurações do PIC
#include <math.h>                   //inclusão da função seno sin()
#fuses HS,NOPROTECT,NOWDT,NOLVP     //fuses
#use delay(crystal=20MHz)           //clock

//definindo taxa de transmissão (250kbits)
#define CAN_125 FALSE
#define CAN_250 TRUE
#define CAN_500 FALSE
#include "mcp2515_SbW.c"            //biblio mcp2515

//definicao dos registradores do nó
#define no1_ident 0x10      //modelo dinâmico veicular
#define no3_ident 0x20      //controlador PID

//constantes para a transformação LONG
#define zero_angulo 2000  //centraliza em zero com um range de 0 a 2000
#define zero_motor 24000  //centraliza em zero com um range de 0 a 48000

//============ variáveis da interrupção do TIMER2
int cont_msg = 0;  //variável contadora da interrupção da mensagem escrita
int cont_mod = 0;  //variável contadora da interrupção do modelo dinâmico

//============ interrupção do TIMER2
#int_timer2
void TIMER2_isr(void)
{
   cont_msg++; //conta a cada 200us
   cont_mod++; //conta a cada 200us
}


//###################### FUNCAO PRINCIPAL ####################//
void main()
{
   //variaveis de leitura dos nós
   struct rx_stat rxstat;     //retorno do buffer de leitura
   int in_data[8];            //pacote de dados de entrada
   int32 rx_id = 0;           //variavel para leitura do buffer
   unsigned int8 rx_len;      //retorna tamanho dos dados do buffer
   int i;                     //variavel contadora para limpar os buffers
   signed long long Vr = 0;   //variavel de controle do motor das rodas

   //variaveis do modelo matematico
   long out_data[3];        //pacote de dados de saída
   signed long long B[2];   //ângulo de derrapagem da carroceria do veículo
   signed long long r[2];   //taxa de guinada
   signed long long Or[2];  //ângulo de esterçamento dos pneus
   signed long long DOr[2]; //derivada ângulo de esterçamento dos pneus
   signed long long ir[2];  //corrente do motor CC
```

```
signed long long sgn;    //torque de fricção

//funções de abertura / inicialização
setup_adc_ports(NO_ANALOGS|VSS_VDD);
setup_adc(ADC_OFF);
setup_psp(PSP_DISABLED);
setup_spi(SPI_SS_DISABLED);
setup_wdt(WDT_OFF);
setup_timer_2(T2_DIV_BY_4,124,10); //interrupção a cada 1ms
setup_comparator(NC_NC_NC_NC);
setup_vref(FALSE);
enable_interrupts(INT_TIMER2);
enable_interrupts(GLOBAL);

//ciclo para limpar os buffers de dados de entrada
for (i=0;i<8;i++)
   in_data[i]=0;

//zerando variáveis
B[0]=0; B[1]=0; r[0]=0; r[1]=0; Or[0]=0; Or[1]=0;
DOr[0]=0; DOr[1]=0; ir[0]=0; ir[1]=0;

can_init();   //inicia o CAN


while(TRUE)   //lembrando que ms trabalha a cada 200us
{
  //============= LEITURA DO NÓ 1 =================//
  if (can_kbhit()) //can_kbhit: deixa ler se tem dados no buffer
  {
      if(can_getd(rx_id, &in_data[0], rx_len, rxstat))
      {
          if (rx_id == no3_ident) //se identificador é igual ao NÓ 1
          {
              //faz o cálculo para conversão de HEX para DEC
              Vr = in_data[1];//alocado para converter para LONG LONG
              Vr = Vr*256 + in_data[0] - zero_motor;
          }
      }
  }

  //====== CONSTRUÇÃO MODELO MATEMÁTICO =======//
  //=== CICLO DE INTERRUPÇÃO PARA MODELO DINÂMICO >> a cada 62ms
  if (cont_mod >= 62)
  {
    cont_mod = 0; // zera o contador da interrupção

    //lógica para realizar o sgn('Or) do modelo
    if(DOr[0] > 0)
        sgn = 1;
    else if (DOr[0] < 0)
        sgn = -1;
    else
        sgn = 0;

    //construção do modelo matemático para v = 5m/s
    B[1]= 9548*B[0] -526*r[0] +126*Or[0] +3*DOr[0];
    r[1]= 1042*B[0] +9518*r[0] +461*Or[0] +10*DOr[0];
    Or[1]= 711*B[0] +134*r[0] +9289*Or[0] +343*DOr[0] +30*ir[0] +51*Vr
           ...-43*sgn;
     DOr[1] = 18630*B[0] +3237*r[0] -18630*Or[0] +2421*DOr[0]
```

```
                ...+235*ir[0] +1442*Vr -1054*sgn;
        ir[1] = -11*B[0] -2*r[0] +11*Or[0] -2*DOr[0] +1760*Vr;

        //atualizando as variáveis do modelo
        B[0] = B[1]/10000;
        r[0] = r[1]/10000;
        Or[0] = Or[1]/10000;
        DOr[0] = DOr[1]/10000;
        ir[0] = ir[1]/10000;

         //=== colocando no buffer de saída as variáveis desejadas

         //escrita do ângulo de esterçamento dos pneus + offset
         out_data[0] = Or[0] + zero_angulo;
      }

      //======== CICLO DE INTERRUPÇÃO PARA MSG NO CAN >> a cada 8ms
      if (cont_msg >= 8 && can_tbe())
      {
        cont_msg = 0; // zera o contador da interrupção

        //============ ESCRITA DO NÓ 3 ===============//
        can_putd(no1_ident, &out_data, 2, 0, 0, 0);
      }

    }//=============== FINAL DO WHILE(TRUE) =================//
}//============== FINAL VOID MAIN() =================//
```

ANEXO B – ALGORITMO PRINCIPAL 2 (CONTROLADOR PID E REFERÊNCIA – NÓ 3)

```
//=====================================================//
//======= ESTE PROGRAMA TRABALHA COM O NÓ 3 ===========//
//==== o nó 3 faz a leitura do nó 1 e trabalha ========//
//==== o nó 3 tem a lógica do PID, resposta ao salto ==//
//=====================================================//
//AUTOR: Walter Sengik da Cruz (waltersengikcruz@gmail.com)

#include "18F4550_SbW.h"            //configurações do PIC
#include <math.h>                   //inclusão da função seno sin()
#fuses HS,NOPROTECT,NOWDT,NOLVP     //fuses
#use delay(crystal=20MHz)           //clock

//definindo taxa de transmissão (250kbits)
#define CAN_125 FALSE
#define CAN_250 TRUE
#define CAN_500 FALSE
#include "mcp2515_SbW.c"    //biblio mcp2515

//definicao dos registradores do nó
#define no1_ident 0x10      //modelo dinâmico veicular
#define no3_ident 0x20      //controlador PID

//definição de uso do botão
#define BOTAO input(PIN_A4)

//constantes
#define zero_angulo 2000    //centraliza em zero com um range de 0 a 2000
#define multip_angulo 1000  //para variar entre -2° a 2°
#define zero_motor 24000    //centraliza em zero com um range de 0 a 48000
#define pi_w 3.141592       //pi

//definição das constantes de ganho do PID
#define Kp 102      //ganho do Proporcional
#define Ti 0.15     //ganho do integrador (inversamente proporcional)
#define Td 0.0375   //ganho do derivativo
#define N 1         //suavização do derivativo
#define h 0.004     //taxa de amostragem do PID (4ms)

//============ variáveis da interrupção do TIMER2
int cont_msg = 0; // contadora da interrupção da mensagem escrita
int cont_PID = 0; // contadora da interrupção do PID e resposta ao salto
int cont_SP = 0;  // contadora da referência

//============ interrupção do TIMER2
#int_timer2
void TIMER2_isr(void)
{
   cont_msg++; //conta a cada 200us
   cont_PID++; //conta a cada 200us
   cont_SP++;
}

//###################### FUNCAO PRINCIPAL ####################//
void main()
{
```

```
//variaveis de leitura dos nós
struct rx_stat rxstat;  //retorno do buffer de leitura
int in_data[8];         //pacote de dados de entrada
int32 rx_id = 0;        //variável para leitura do buffer
unsigned int8 rx_len;   //retorna tamanho dos dados do buffer
int i;                  //variável contadora para limpar os buffers
signed long Or[2] = {0,0}; //variável do angulo das rodas

//variaveis resposta ao salto e escrita no nó
signed long Os = 0;        //resposta ao salto em float para o PID
long contador_RS = 0;      //contador da resposta ao salto
long out_data[3];          //pacote de dados de saída
int botao_desl = 0;        //verificar se o botão esta desligado
signed long temp_var = 0;  //variável temporária para cálculos
signed long long Vr = 0;   //variável de controle do motor das rodas

//variaveis do controlador PID
signed long long erro = 0;   //erro (Os - Or)
signed long long dif = 0;    //dif (Or(k+1) - Or(k))
signed long long aD = 0;     //primeira parte constante diferencial
signed long long bD = 0;     //segunda parte constante diferencial
signed long long aI = 0;     //parte constante integral

signed long long Pk = 0;     //variável do ganho proporcional
signed long long Dk = 0;     //variável do ganho derivativo
signed long long Ik = 0;     //variável do ganho integral


//funções de abertura / inicialização
setup_adc_ports(NO_ANALOGS|VSS_VDD);
setup_adc(ADC_OFF);
setup_psp(PSP_DISABLED);
setup_spi(SPI_SS_DISABLED);
setup_wdt(WDT_OFF);
setup_timer_2(T2_DIV_BY_4,124,10); //interrupção a cada 1ms
setup_comparator(NC_NC_NC_NC);
setup_vref(FALSE);
enable_interrupts(INT_TIMER2);
enable_interrupts(GLOBAL);

//ciclo para limpar os buffers de dados de entrada
for (i=0;i<8;i++)
   in_data[i]=0;

// limpa os buffer de dados de saída
out_data[0] = 0;
out_data[1] = 1;

can_init();   //inicia o CAN

//calculo das constantes do PID
aI = (Kp*h*10)/Ti;           // ganho integral
//multiplicado por 10 pela resolução... se não dá menor que 1

aD = (Td*1000)/(N*h+Td);    // ganho derivativo parte A
//multiplicado por 1000 pela resolução... se não dá menor que 1

bD = (N*Kp*Td)/(N*h+Td);  // ganho derivativo parte B

while(TRUE)     //lembrando que ms trabalha a cada 200us
{
```

```
//============= LEITURA DO NÓ 1 =================//
if (can_kbhit()) //can_kbhit: deixa ler se tem dados no buffer
{
    if(can_getd(rx_id, &in_data[0], rx_len, rxstat))
    {
        if (rx_id == no1_ident)    //se identificador é igual ao NÓ 1
        {
            Or[1] =  in_data[1];
            Or[1] =  Or[1]*256 + in_data[0] - zero_angulo;
            //Or - offset
        }
    }
}

//======= CONSTRUÇÃO LÓGICA CONTROLADOR PID + REFERÊNCIA ====//
//============================================================//

//======== CICLO DE INTERRUPÇÃO DO PID >> a cada 4 x 1ms = 4ms
if (cont_PID >= 4)
{
  cont_PID = 0;      // zera o contador da interrupção

  //====================== PID ==================//
    erro = (Os - Or[1]);
    //colocado o erro para converter para LONG LONG as variáveis
    dif = Or[1] - Or[0];
    //colocado a difer para converter para LONG LONG as variáveis

    // ação proporcional
    Pk = Kp*erro;

    //acao integral (/10 devido a resolucao)
    Ik = Ik + aI*erro/10;

    //acao derivativa (/1000 devido a resolucao)
    Dk = aD*Dk/1000 - bD*dif;

   //equações não utilizadas para não ocupar muito processamento:
      //Ik = Ik + (Kp*h*erro)/Ti;
      //Dk = (Td*Dk)/(N*h+Td) - (N*Kp*Td*dif)/(N*h+Td);

    Vr = Pk + Ik + Dk; // soma das ações para controle do motor
    if(Vr > +24000) Vr = +24000; // se maior que 24V, satura em 24V
    if(Vr < -24000) Vr = -24000; // se menor que -24V, satura em -24V

    Or[0] = Or[1];
}

// OBS: o botao tem o acionamento NF... a lógica é ao contrário
if(BOTAO == 1 && botao_desl == 0)
    botao_desl = 1; //indica que o botão estava desligado

// se foi apertado o botao e antes estava desapertado
if(BOTAO == 0 && botao_desl == 1)
    botao_desl = 2; //indica que o botão foi apertado

//======== CICLO DA CURVA DO REFERÊNCIA >> a cada 4 x 1ms = 4ms
if (cont_SP >= 4 && botao_desl == 2) //se fechou 4ms e botão apertado
{
  cont_SP = 0;      // zera o contador da interrupção
```

```
        //==== resposta do volante conforme o artigo do Rashid
        if(contador_RS>=0 && contador_RS<=250) //=== t de 0 a 1s
            Os = 0; // recebe zero

        if(contador_RS>250 && contador_RS<750) //=== t de 1 a 3s
            Os = multip_angulo*((pi_w/4)*sin(-pi_w*((float)contador_RS/250)
                  .../2-pi_w/2)); //funcao senoidal positiva

        if(contador_RS>=750 && contador_RS<=1250) //=== t de 3 a 5s
            Os = 0; // recebe zero

        if(contador_RS>1250 && contador_RS<2250) //=== t de 5 a 9s
        {
            Os = multip_angulo*(pi_w/2)*(sin(pi_w*0.25*
                   ...((float)contador_RS/250)-pi_w/4)); //função senoidal

            temp_var = multip_angulo*(-pi_w/3);

            if(Os < temp_var) // se menor que -pi/3, satura
                Os = temp_var;
        }

        if(contador_RS>=2250 && contador_RS<=3000) //=== t de 9 a 12s
            Os = 0; // recebe zero

        if(contador_RS>3000 && contador_RS<=3750) //=== t de 12 a 15s
            Os = multip_angulo*((pi_w/9)*((float)contador_RS/250)
                   ...-4*pi_w/3); //função linear positiva

        if(contador_RS>3750 && contador_RS<4500) //=== t de 15 a 18s
            Os = multip_angulo*((-pi_w/9)*((float)contador_RS/250)+2*pi_w);
            //função linear negativa

        if(contador_RS>=4500 && contador_RS<=5000) //=== t de 18 a 20s
            Os = 0; // recebe zero


        if(contador_RS > 5000) //====== faz o envio ateh 20 segundos
        {
            botao_desl = 0;
            contador_RS = 0;
        }

        contador_RS++; //contador da logica resposta ao salto
      }

      //======== CICLO DE INTERRUPÇÃO PARA MSG NO CAN >> a cada 8ms
      if (cont_msg >= 8 && can_tbe
      {
        cont_msg = 0; // zera o contador da interrupção

        out_data[0] = Vr + zero_motor; //escrita da tensão de saída do motor
        out_data[1] = Os + zero_angulo; //escrita da resposta ao salto
        out_data[2] = contador_RS;  //sinal do botão

        //============ ESCRITA DO NÓ 3 ===============//
        can_putd(no3_ident, &out_data, 6, 0, 0, 0);
       }
    }//=============== FINAL DO WHILE(TRUE) =================//
}//============== FINAL VOID MAIN() =================//
```