Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia Automotiva

MODELAGEM E ANÁLISE DE REDES AUTOMOTIVAS EM AMBIENTE VIRTUAL

Autor: Rafael Rodrigues da Silva
Orientador: Dr. Evandro Leonardo Silva Teixeira

Brasília, DF
2015

Rafael Rodrigues da Silva

MODELAGEM E ANÁLISE DE REDES AUTOMOTIVAS EM AMBIENTE VIRTUAL

Monografia submetida ao curso de graduação em Engenharia Automotiva da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Automotiva .

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dr. Evandro Leonardo Silva Teixeira

Brasília, DF

2015

Rafael Rodrigues da Silva
MODELAGEM E ANÁLISE DE REDES AUTOMOTIVAS EM AMBIENTE VIRTUAL/ Rafael Rodrigues da Silva. – Brasília, DF, 2015-
77 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Evandro Leonardo Silva Teixeira

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2015.

1. Simulação. 2. Redes Automotivas. I. Dr. Evandro Leonardo Silva Teixeira. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. MODELAGEM E ANÁLISE DE REDES AUTOMOTIVAS EM AMBIENTE VIRTUAL

CDU 02:141:005.6

Rafael Rodrigues da Silva

MODELAGEM E ANÁLISE DE REDES AUTOMOTIVAS EM AMBIENTE VIRTUAL

Monografia submetida ao curso de graduação em Engenharia Automotiva da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Automotiva .

Dr. Evandro Leonardo Silva Teixeira
Orientador

Dr. Edson Mintsu Hung
Convidado 1

Dr. André Murilo de Almeida Pinto
Convidado 2

Brasília, DF
2015

Agradecimentos

Agradeço primeiramente a Deus por me iluminar com sabedoria e perseverança em todos os momentos da minha vida.

Aos meus pais Zanir e Nephtali que são a base sólida da minha vida, agradeço por terem sempre acreditado no meu potencial e me incentivarem a ser melhor a cada dia. Agradeço também aos meus irmãos por estarem ao meu lado em todos os momentos da execução deste trabalho.

A minha amada Ana Paula pela paciência e compreensão durante a realização deste trabalho. Agradeço ainda por estar ao meu lado nesses seis maravilhosos anos.

Ao meu orientador Prof. Dr. Evandro Teixeira pelo companheirismo, paciência e principalmente por compartilhar comigo o grande conhecimento que possui, o que foi fundamental para a realização deste trabalho e para a decisão da área que pretendo seguir no campo acadêmico e profissional.

"Aprender é a única coisa
de que a mente nunca se cansa,
nunca tem medo,
e nunca se arrepende."
(Leonardo Da Vinci)

Resumo

A utilização de componentes eletroeletrônicos em veículos tem aumentado significativamente nos últimos anos. Isto se deve principalmente às vantagens que estes trazem frente aos sistemas puramente mecânicos ou hidráulicos tais como flexibilidade, redução de custos, dentre outros. Da necessidade de estabelecer a comunicação entre as unidades de controle destes sistemas de forma organizada e eficiente surgiram as redes automotivas, que consistem num barramento no qual as diversas unidades presentes no veículo podem se conectar a fim de receber e enviar mensagens que contêm informações fundamentais para realizarem suas rotinas. Neste contexto, este trabalho explora ferramentas virtuais voltadas para a modelagem e simulação em que é possível representar, de forma adequada, a rede automotiva de um veículo. E baseado-se no protocolo CAN, especificamente o padrão J1939 que é voltado para aplicações em veículos comerciais, são apresentados estudos de casos com diferentes configurações de redes automotivas e o desempenho desses cenários distintos são avaliados segundo algumas métricas apresentadas.

Palavras-chaves: Simulação de Redes Automotivas, Rede CAN, J1939.

Abstract

The use of electronic components in vehicles has increased greatly in recent years. This is mainly due to the advantages they bring forward to purely mechanical or hydraulic systems such as flexibility, cost reduction, among others. The need to establish communication between the control units of these in an organized and efficient systems emerged automotive networks, which consist of a bus in which the various units in the vehicle can connect in order to receive and send messages that contain key information for perform their routines. In this context, this paper presents the research and the use of virtual tools focused on modeling and analysis of automotive networks in a virtual environment that can be represented in an appropriate manner, the automotive vehicle network with control units and their settings . The parameters are presented which are usually taken as the key network performance indicators, and based on the CAN protocol, specifically the J1939 standard that is designed for applications in commercial vehicles, case studies are presented with different configurations of networks used in actual vehicles and the performance of these various scenarios are analyzed according to the metrics presented.

Key-words: Automotive Network Simulation, CAN Network, J1939.

Lista de ilustrações

Figura 1 – Exemplo de uma rede automotiva (DGE INC,). 18
Figura 2 – Camadas no modelo OSI. 20
Figura 3 – Camadas OSI no protocolo CAN - Adaptada de (BRUDNA, 2000). . . 22
Figura 4 – Nível de tensão utilizado para o CAN (GODOY, 2007). 22
Figura 5 – Conceito de acesso ao barramento no protocolo CAN 23
Figura 6 – Formato do frame CAN. Adaptado de (GODOY, 2007) 24
Figura 7 – Formato do frame no J1939 (FELLMETH; LöFFLER, 2008). 28
Figura 8 – PGN - Temperatura do motor. (FRITZ KüBLER GMBH, 2009). . . . 29
Figura 9 – Performance da rede. Adaptado de Lian, Moyne e Tilbury (2002). . . . 31
Figura 10 – Interface *software* CANoe. (VECTOR, 2014). 37
Figura 11 – Interface do *software* Bus Master. 38
Figura 12 – Etapas de elaboração do trabalho. 39
Figura 13 – Fluxograma do algorítimo das ECUs. 42
Figura 14 – Caminhão Scania 164/480 43
Figura 15 – Arquitetura da rede do caminhão Scania 164 45
Figura 16 – *Frames* presentes no barramento 46
Figura 17 – Variação dos valores através do *handler on key* 47
Figura 18 – Desempenho da rede do caso 1 48
Figura 19 – Caminhão R580 Fonte: Scania 50
Figura 20 – Arquitetura da rede para o R580 51
Figura 21 – Desempenho da rede do caso 2 52
Figura 22 – Volvo FH400 53
Figura 23 – Volvo FH400 54
Figura 24 – Desempenho da rede do caso 3 55
Figura 25 – Criação do *database* 64
Figura 26 – Janela de gerenciamento das mensagens 65
Figura 27 – Janela de gerenciamento das mensagens 65
Figura 28 – Tela de configuração dos sistemas simulados 66
Figura 29 – Iniciando um sistema simulado 66
Figura 30 – Inserção de novas ECUs ao sistema simulado. 67
Figura 31 – Definição do nome e endereço da ECU 67
Figura 32 – Código carregado no Bus Master 68
Figura 33 – Tela de simulação no BusMaster 69
Figura 34 – Tela de simulação no BusMaster 69

Lista de tabelas

Tabela 1 – Classificação das redes automotivas 19
Tabela 2 – Parâmetros importantes de configuração de redes CAN. (GODOY, 2007) 31
Tabela 3 – Parâmetros de desempenho de redes CAN.(GODOY, 2007) 32
Tabela 4 – Parâmetros de configuração da rede CAN para o caso 1 45
Tabela 5 – Parâmetros de configuração da rede CAN para o caso 2 50
Tabela 6 – Parâmetros de configuração da rede CAN para o caso 3 54
Tabela 7 – Execução das etapas planejadas no trabalho. 59

Lista de abreviaturas e siglas

ABS	*Anti-lock Breaking System*
ACP	*Audio Control Protocol*
BMS	*Brake Management System*
CAN	*Controller Area Network*
ECU	Unidade de Controle Eletrônico
EMS	*Engine Management System*
ESP	*Eletronic Stabilit Program*
GMS	*Gear-Box Management System*
ISO	*International Organization for Standardization*
LIN	*Local Interconnect Network*
OBD	*Onboard Diagnosic*
OME	*Original Equipament Manufaturer*
OSI	*Open Systems Interconnection*
PG	Grupo de Parâmetros
PGN	Número Grupo de Parâmetros
SAE	Sociedade dos Engenheiros Automotivos
SMS	*Suspension Management System*
SPN	*Suspect Parameter Number*
TSF	*Test feature set*

Sumário

1 INTRODUÇÃO 14
1.1 Objetivos do trabalho 15
1.2 Estrutura do texto 16

2 REVISÃO BIBLIOGRÁFICA 17
2.1 A importância das redes automotivas em veículos 17
2.2 Redes automotivas 18
2.2.1 Modelo OSI/ISO 19
2.3 Rede CAN 21
2.3.1 Camada física 22
2.3.2 Camada de enlace 23
2.3.2.1 Método de arbitragem 23
2.3.2.2 Mensagens no protocolo CAN 24
2.3.2.3 Verificação de erros 26
2.4 SAE J1939 26
2.5 Análise do desempenho de redes automotivas 30
2.5.1 Tempo de transmissão 32
2.5.2 Tempo de transmissão sob condições de erro 33
2.5.3 Taxa de utilização do barramento 34
2.6 Benefícios e desafios da simulação de redes automotivas 35
2.7 Ferramentas para a simulação de redes automotivas 36
2.7.1 CANoe 36
2.7.2 Bus Master 37

3 METODOLOGIA 39
3.1 Revisão bibliográfica 39
3.2 Definição do problema 39
3.3 Coleta de dados 40
3.4 Modelagem e Simulação 40
3.5 Validação do Modelo 42

4 ESTUDO DE CASO 43
4.1 Caso 1: Scania 164 43
4.1.1 Modelagem da rede 44
4.1.2 Simulação 46
4.1.3 Análise de resultados 47

4.2 Caso 2: Scania R580 49
4.2.1 Modelagem da rede 50
4.2.2 Análise de resultados 51
4.3 Caso 3: Volvo FH400 53
4.3.1 Modelagem da rede 54
4.3.2 Análise dos resultados 55

5 CONCLUSÃO 57

REFERÊNCIAS 60

ANEXO A – TUTORIAL BUSMASTER 63

ANEXO B – PROGRAMAÇÃO DAS ECUS DO CASO 1 70

1 Introdução

A inclusão de tecnologia embarcada nos automóveis tem crescido exponencialmente nos últimos anos. Em termos financeiros estas aplicações chegam a ser responsável por um quarto do valor total do veículos. A utilização de componentes eletroeletrônicos se tornou fundamental para o setor automotivo, pois garante grande quantidade de benefícios quando comparados a sistemas puramente mecânicos e hidráulicos (BLAKE; LEADER, 2005).

Um estudo realizado pela IBM *Institute for Business Value* defende que existem cinco fatores que motivam o desenvolvimento e a implementação de softwares e eletrônicos nos veículos (GUMBRICH; KOPPINGER, 2004)

- Competição - Empresas estão competindo cada vez mais pela preferência dos consumidores e os clientes cada vez mais críticos em relação ao produtos que desejam.
- Diferenciação do produto - Criar produtos com características únicas, que se destaquem frente aos concorrentes é um dos principais motivadores do desenvolvimento.
- Legislação - Legislações ambientais e de segurança que fomentam o desenvolvimento de novos sistemas criam uma grande demanda por sistemas embarcados.
- Expectativas dos clientes - O desejo por uma condução cada vez mais segura e de alta perfomance encoraja os clientes a buscarem produtos mais desenvolvidos tecnologicamente.
- Inovações tecnológicas - Novas tecnologias tais como sistema de navegação, monitoramento *online* surgem constantemente, o que demanda cada vez mais da aplicação de sistemas embarcados.

Este estudo mostra os principais motivos da busca por funcionalidades que impressionem cada vez mais o mercado consumidor e se diferencie frente aos concorrentes. A aplicação destas funcionalidades em veículos é possibilitada devido ao constante desenvolvimento de componentes e sistemas eletroeletrônicos.

Os sistemas eletroeletrônicos implementados nos veículos são formados basicamente por dispositivos auxiliares, sensores, atuadores e unidades de controle eletrônico (ECU). Uma ECU é frequentemente utilizada para adquirir dados de sensores e controlar diversos atuadores presentes no sistema. O número de ECUs cresce ao passo em que novas funcionalidades são implementadas nos veículos. De fato Monot et al. (2010) estima que a quantidade média de ECUs nos veículos mais que dobrou nos últimos 10 anos.

A troca de informação entre as diveras ECUs presentes no veículo se faz necessário, pois ECUs distintas podem necessitar dos mesmos dados para realizar operações diferentes. Dessa forma possibilitar a transferência de informações entre as unidades impacta diretamente na redução de sensores e na eficácia do sistema (NAVET et al., 2005).

Encontrar uma forma de organizar, diminuir custos e tornar mais eficiente a comunicação entre ECUs estimulou o desenvolvimento das redes de comunicação veiculares. Uma rede de comunicação veicular consiste em uma arquitetura de comunicação em que as todas as ECUs são conectadas ao mesmo barramento e seguindo requisitos físicos e de software são capazes de trocar informações e garantir o funcionamento dos diversos dispositivos presentes no automóvel (NAVET et al., 2005).

Para auxiliar no desenvolvimento e na implementação das redes de comunicação veicular surgiram ferramentas capazes de realizar simulações em que abordam diversos requisitos de hardware e software de um modelo real. Estas ferramentas contêm uma série de funcionalidades que permitem ao desenvolvedor analisar o comportamento da rede baseando-se em parâmetros de desempenho e desta forma possibilita realizar o correto dimensionamento da rede, além de minimizar e corrigir erros que surgiriam na implementação do sistema físico (KLüSER, 2004).

1.1 Objetivos do trabalho

Objetivo Geral:

Este trabalho tem como objetivo utilizar ferramentas de simulação para modelar e analisar redes automotivas baseadas no protocolo CAN (Controller Área Network), observando o comportamento destas redes através de métricas de desempenho. Deseja-se comparar o comportamento da rede automotiva de diferentes veículos a fim de avaliar o desempenho da rede para configurações distintas.

Objetivos Específicos:

- Estudar o desenvolvimento das redes automotivas e protocolos de comunicação;
- Explorar as métricas de desempenho que avaliam o comportamento de redes automotivas;
- Explorar o software BusMaster quanto à modelagem e análise de redes automotivas;
- Analisar o desempenho das redes automotivas segundo as métricas de desempenhos adotadas.
- Criar um modelo de rede capaz de simular o funcionamento da rede automotiva de veículo real.

1.2 Estrutura do texto

Os ítens abaixo fornecem ao leitor uma breve descrição dos capítulos que compõem este trabalho.

Capítulo 2: Neste capítulo é apresentada a revisão bibliográfica realizada para desenvolver este trabalho. Inicialmente são apresentados os conceitos de redes automotivas e de protocolos de comunicação. Por fim são apresentados tópicos referentes à modelagem e simulação de redes automotivas, métricas que avaliam o desempenho das redes de comunicação, ferramentas computacionais existentes para modelagem e simulação de redes automotivas, dentre outros.

Capítulo 3: Este capítulo descreve a metodologia definida para a realização deste trabalho, apresentando e detalhando as etapas que foram adotadas no desenvolvimento do mesmo.

Capítulo 4: Este capítulo apresenta três estudos de caso. O caso 1 e 2 tratam de redes automotivas de veículos comerciais com diferentes configurações, em que estas são modeladas e analisadas segundo parâmetros alguns parâmetros de desempenho. O caso 3 apresenta a modelagem da rede automotiva de um caminhão, em que foram obtidas informações da rede no próprio veículo e com base nestes dados realizou-se o modelo da rede. Por fim as métricas de desempenho são aplicadas para análise da rede tanto do veículo quanto do modelo.

Capítulo 5: Este capítulo apresenta considerações finais sobre o trabalho realizado e por fim propõe questões para trabalhos futuros.

2 Revisão bibliográfica

Este capítulo aborda o papel das redes automotivas em veículos, ressaltando os métodos, benefícios e desafios de sua implementação. Explora também técnicas que avaliam o desempenho de redes automotivas segundo métricas adotadas por fabricantes e pesquisadores. É apresentado também o conceito de modelagem e simulação de redes automotivas, procedimento realizado em ambientes virtuais que permitem o projeto, monitoramento, análise entre outros recursos fundamentais para avaliação das funcionalidades das redes.

2.1 A importância das redes automotivas em veículos

A tecnologia embarcada nos automóveis é uma realidade que se torna cada vez mais presente nos veículos. Segundo Leen e Heffernan (2002) os componentes eletroeletrônicos são responsáveis por cerca de 23 % do custo de fabricação em alguns modelos de luxo e no que se refere a desenvolvimento de novas ferramentas e funcionalidades a eletrônica é responsável por cerca de 80 % das criações.

A substituição de componentes mecânicos e hidráulicos por componentes eletrônicos nos automóveis tem crescido consideravelmente nos últimos anos. De fato, a indústria automotiva tem sido pressionada por novas demandas como a economia de combustível, segurança, entretenimento, desempenho entre outras necessidades. Desde o sistema de alimentação de combustível, no qual o antigo carburador foi substituído pela injeção eletrônica, até dispositivos que auxiliam o motorista na condução como ABS (*Antilock Braking System*), ESP (*Electronic Stability Program*), EPS (*Eletric Power Steering*) as inovações tecnológicas têm se tornado um dos principais focos da indústria automotiva (NAVET et al., 2005)

Um subsistema eletrônico automotivo é composto por sensores, ECUs e atuadores. Estes componentes são comumente responsáveis por detectar, processar e atuar nas mais diversas funcionalidades do veículo (NAVET et al., 2005). Por exemplo, no sistema de injeção eletrônica do tipo LE-Jetronic, a ECU responsável obtêm dados dos sensores de temperatura, medidor do fluxo de ar, sensor de posição da borboleta, rotação e carga do motor e dessa forma calcula a quantidade de combustível que precisa ser injetada no cilindro através das válvulas injetores (atuadores) (BOSCH, 2001).

O número de ECUs, unidades responsáveis pelo processamento das ações eletronicamente controladas, tem crescido ao passo que novas funcionalidades são inseridas nos veículos. Atualmente, automóveis como o volvo XC90 podem contar com mais de 40 ECUs (NAVET et al., 2005) e em alguns veículos de luxo este número pode ultrapassar as 70

unidades (ALBERT, 2004).

Com o aumento das funcionalidades do veículo surge a necessidade natural de promover a troca de informações entre ECUs. A estrutura necessária para proporcionar a comunicação, caso as ECUs sejam conectadas diretamente uma a outra (ponto-a-ponto), se torna inviável pois fios, conectores e acessórios acarretam em peso, custo e complexibilidade elevada. Diante deste cenário surgem as redes de comunicação veicular, no qual existem canais, denominados barramentos, em que diversas ECUs podem estar conectadas mutuamente para transmitir e obter informações necessárias para suas rotinas de funcionamento (NAVET et al., 2005).

Para realizar a transferência de dados através do barramento, via de regra, as redes automotivas utilizam protocolos de comunicação, que nada mais são do que padrões adotados para definir as atividades realizadas pela rede.

2.2 Redes automotivas

As redes automotivas surgiram da necessidade de promover a interconexão entre as diversas ECUs presentes nos veículos. Assim cada ECU que se encontra conectada ao barramento poderá acessar as informações adquirindo os dados necessários para executar suas tarefas (LIU; AN; YANG, n.d.).

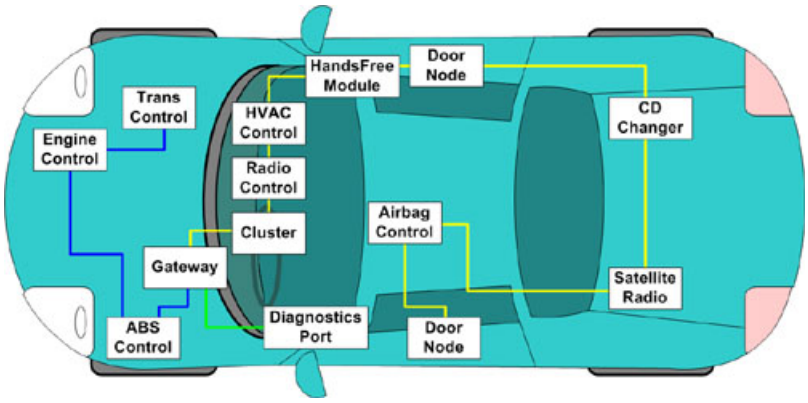


Figura 1 – Exemplo de uma rede automotiva (DGE INC,).

As redes automotivas seguem uma classificação, segundo a sociedade dos engenheiros automotivos (SAE), em que são divididas em três categorias de acordo com a taxa de transmissão e funcionalidades, como mostra a tabela 1 (LIU; AN; YANG, n.d.).

Classe A

As principais características da classe A são as baixas taxas de transmissão (até 10 kb/s). Geralmente este tipo de rede é aplicada em funções auxiliares e de conforto,

Tabela 1 – Classificação das redes automotivas

Classe	Velocidade	Aplicação
A	<10 kb/s	Áudio, ajuste dos espelhos retrovisores, travamento das portas e etc.
B	10 kb/s a 125 kb/s	Dados de sensores, informações do painel e etc.
C	acima de 125 kb/s	Dados em tempo real, dinâmica do veiculo, *brake by wire* e etc.

tais como limpadores de para-brisa automatizado, travamento das portas, abertura e fechamento das janelas, climatização do habitáculo dentre outras funções. Os principais protocolos dessa classe são ACP (*Audio Control Protocol*) , BEAN, CCD (*Chrysler Collision Detection*), LIN (*Local Interconnect Network*) entre outros sendo o LIN mais utilizado atualmente (PARET, 2007).

Classe B

Os protocolos classe B são geralmente utilizados para transferência de informações, instrumentação e controle de diagnósticos. Os principais protocolos dessa classe são VAN, J1850, ISO11519-2 dentre outros (LIU; AN; YANG, n.d.).

Classe C

Os protocolos da classe C operam em velocidades acima de 125 kb/s e são geralmente utilizados em controles de tempo-real, seja no gerenciamento do motor, sistemas X-*by wire* e etc. Essa classe inclui os protocolos CAN, J1939, FlexRay entre outros (BELL, 2002).

2.2.1 Modelo OSI/ISO

Para melhor compreensão das redes automotivas, é necessário entender como a comunicação é executada entre dispositivos numa rede de comunicação geral, com isso visando padronizar esta transferência de dados seja entre computadores, dispositivos móveis, ECUs ou outros dispositivos a ISO (*International Organization for Standardization*) desenvolveu o modelo de transmissão OSI (*Open Systems Interconnection*) que é uma estrutura de rede dividida em sete camadas, na qual cada uma é composta por dispositivos eletrônicos e/ou softwares que implementam determinados serviços aos próximos níveis (SOUSA, 2002). A Figura 2 ilustra as sete camadas definidas para o modelo OSI e sua descrição é baseada em Simoneau (2006).

Figura 2 – Camadas no modelo OSI.

Camada de aplicação

Essa camada é a mais próxima do usuário, é nela onde o usuário pode interagir com a rede requisitando serviços das camadas inferiores.

Camada de apresentação

A camada de apresentação é responsável por obter os dados gerados pela camada de aplicação e transformá-los numa linguagem comum a rede em que estão inseridos, ou seja, deixa as mensagens em pacotes, caracteres, formatos comuns aos dispositivos receptores.

Camada de seção

Nesta etapa é estabelecida a comunicação entre o dispositivo emissor e receptor a fim de iniciar, gerenciar e detectar erros durante a transmissão das mensagens.

Camada de transporte

A camada de transporte recebe as mensagens das camadas superiores e transmite para a camada de rede, controlando tamanho dos pacotes de dados, o fluxo de informação dentre outras variáveis.

Camada de rede

A camada de rede define o endereçamento de destino dos pacotes de dados, assim como a rota que deverão seguir para chegarem aos dispositivos de destino.

Camada de enlace

Também conhecida como camada de ligação ou *data link* é responsável pelo encapsulamento dos dados no formato do protocolo definido (*frame*) e também pela detecção de erros durante a transmissão dos bits.

Camada física

Esta camada é responsável por definir os meios físicos nos quais os *bits* serão transmitidos e também as especificações dos parâmetros da rede tais como tensão, tempo de transmissão, cabos, conectores entre outras características.

2.3 Rede CAN

O protocolo CAN (*Controller Area Network*) foi criado na década de 80 por Robert Bosh GmbH visando a comunicação entre dispositivos de controle em automóveis. Embora tenha sido inicialmente projetado no intuito de promover somente a comunicação entre ECUs, este protocolo foi posteriormente utilizado em diferentes áreas como máquinas agrícolas, robótica, satélites entre outras aplicações (SOUSA, 2002). Atualmente é considerado como sendo o protocolo mais difundido em aplicações automotivas, visto que, estima-se que aproximadamente 720 milhões de ECUs baseadas neste protocolo sejam comercializadas por ano (NEWCASTLE UNIVERSITY, 2014).

Para a rede CAN são especificadas apenas as camadas físicas e de enlace do modelo OSI/ISO, como mostra a Figura 3. A utilização das camadas superiores ficam a cargo dos desenvolvedores que as projetam segundo a necessidade de cada aplicação (BRUDNA, 2000).

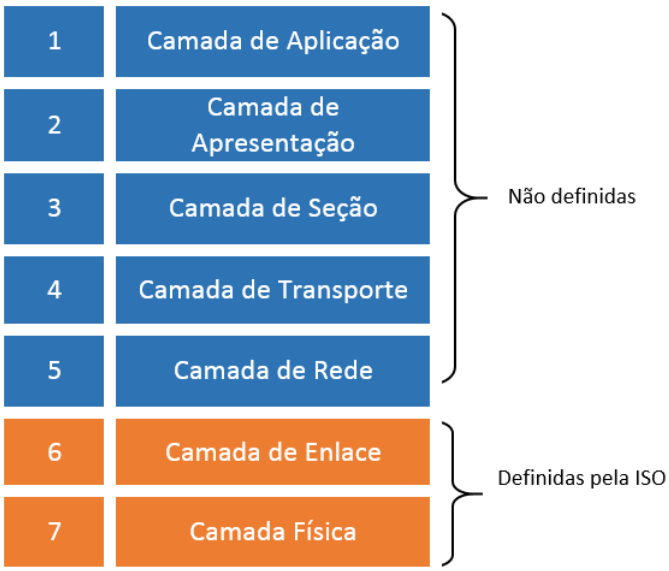


Figura 3 – Camadas OSI no protocolo CAN - Adaptada de (BRUDNA, 2000).

2.3.1 Camada física

A camada física é responsável pela transmissão de informação, na forma de *bits*, entre os dispositivos conectados a rede. Nesta camada são definidos, dentre outros parâmetros, os níveis de tensão, sincronização das ECUs, codificação e decodificação dos *bits*, cabos e conectores presentes na rede.

O barramento CAN é composto por dois condutores, geralmente do tipo cabo de par trançado que proporciona a redução de ruídos. Esses cabos são denominados CAN_H e CAN_L e podem assumir dois níveis de tensão durante a comunicação, sendo recessivo (nível lógico 1) ou dominante (nível lógico 0) como mostra a Figura 4.

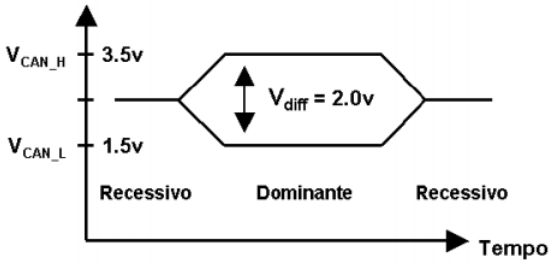


Figura 4 – Nível de tensão utilizado para o CAN (GODOY, 2007).

A dominância de *bit* é um dos conceitos mais importantes na rede CAN. A dominância de bit se assemelha a dominância genética em que a presença de um gene dominante inibe o efeito do gene recessivo (BARBOSA, 2003). O estado do barramento, recessivo ou dominante, é definido pelo diferencial de tensão entre os fios. A tensão elétrica de referên-

cia é 2.5 V que corresponde ao estado recessivo, para assumir o estado dominante o fio CAN_H tem o nível elétrico elevado para 3.5 V enquanto que o CAN_L assume o valor de 1.5V gerando uma diferença de potencial de 2V que corresponde ao estado dominante.

2.3.2 Camada de enlace

A camada de enlace é responsável por encapsular e desencapsular as mensagens, definir a arbitragem das mensagens, detectar e indicar erros nas transmissões ou pacotes de mensagens.

2.3.2.1 Método de arbitragem

No protocolo CAN o método para arbitrar a mensagem a ser transmitida no barramento, caso haja mais de uma ECU transmitindo em simultâneo, é baseado no conceito CSMA/CD (*Carrier Sense Multiple Access with Collision Detect*) em que um *bit* dominante terá prioridade frente ao recessivo em situações de colisão.

A Figura 5 ilustra este método de arbitração em que duas ECUs iniciam a transmissão emitindo um simples bit dominante SOF (*start of frame*), os *bits* enviados continuam com o mesmo valor lógico até o quarto *bit*, instante em que a ECU 1 emite um bit dominante enquanto que a ECU 2 um recessivo, desta forma a ECU 2 encerra a transmissão e a ECU 1 continua com a transmissão da mensagem. Esta verificação continuará (bit-a-bit) até haver apenas uma ECU transmitindo no barramento (PARET, 2007).

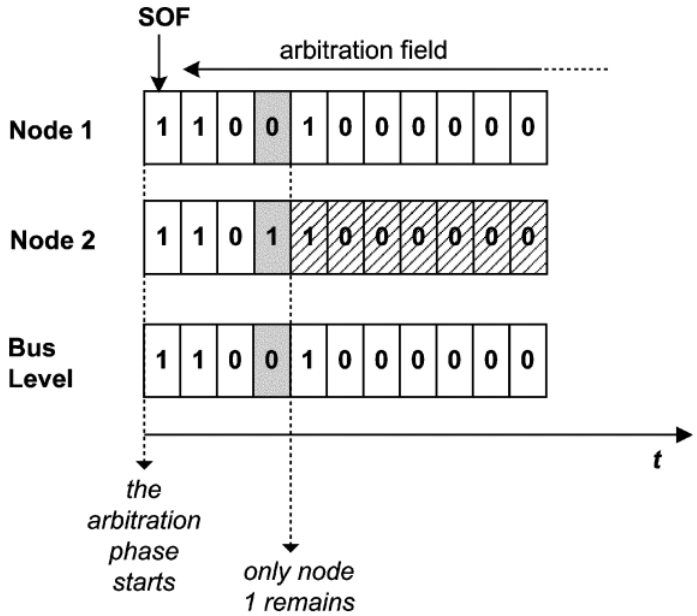


Figura 5 – Conceito de acesso ao barramento no protocolo CAN

2.3.2.2 Mensagens no protocolo CAN

As mensagens no protocolo CAN são encapsuladas na forma de *frames*. Os *frames* são compostos por uma sequência de *bits* delimitados por campos com funções específicas que juntos formam uma mensagem completa.

Os *frames* transmitidos no barramento do protocolo CAN podem ser de quatro tipos diferentes:

- *Frame* de dados - Encapsula mensagens contendo dados.
- *Frame* de requisição - Ocorre quando uma ECU requisita alguma informação proveniente de outra unidade.
- *Frame* de erro - É transmitida por uma ECU quando erros são detectados.
- *Frame* de sobrecarga - É gerado para inserir um tempo a mais entre a transmissão de dois *frames* consecutivos, a fim de evitar que alguma ECU não esteja pronta pra emitir ou receber os *frames* enviados.

O *frame* de dados no protocolo CAN é composto por sete campos principais, sendo que existem duas formas de implementação deste protocolo as quais se diferenciam apenas na quantidade de bits do campo de identificação, podendo assumir o formato padrão (CAN 2.0A) ou formato estendido (CAN 2.0B) (PARET, 2007). A Figura 6 ilustra a composição de um *frame* no formato CAN e mostra as diferenças entre o formato padrão e estendido.

	Campo de arbitragem						Campo de controle			Campo de dados	Campo CRC		Campo ACK		
	INÍCIO FRAME	IDENTIFICADOR	SRR	IDE	ID. ESTENDIDO	RTR	R1	R0	DLC	CAMPO DE DADOS	CRC		ACK		FIM DO FRAME
N. de Bits	1	11	1	1	18	1	1	1	4	64	15	1	1	1	7
		CAN 2.0A													
		IDENTIFICADOR CAN 2.0B													

Figura 6 – Formato do frame CAN. Adaptado de (GODOY, 2007)

Início do frame

Consiste em um bit dominante que tem a função de marcar o início da transmissão do *frame*. Esse *bit* é responsável pela sincronização de todos os nós da rede através do método *hard synchronization* [1]

[1] *Hard syncronization* ocorre após a mudança de estado do barramento de recessivo para dominante após um período de inatividade, este mecanismo faz com que todos as unidades da rede reiniciem o tempo de bit e passem a trabalhar sincronizadas.

Campo de Arbitragem

Para o formato 2.0A este campo é composto por 11 bits de identificação que é o cabeçalho da mensagem seguidos do bit RTR (*remote transmission request bit*) que tem a função de definir se o *frame* é do tipo dados ou requisição, de forma que assume o valor 0 (dominante) para o primeiro caso e 1 (recessivo) para o segundo.

No formato estendido 2.0B o campo contém 29 bits, sendo a primeira parte composta por 11 bits de identificação, seguidos por dois outros bits sendo estes o SRR (*substitute remote request bit* e IDE (*identifier extension bit*). Este campo ainda conta com mais 18 *bits* que complementam a identificação e por fim o *bit* RTR.

- SRR (*substitute remote request bit*) é sempre recessivo para garantir a prioridade de transmissão para os *frames* no formato padrão em detrimento do formato estendido, em casos de colisão dos dois formatos.
- IDE (*identifier extension bit*) é utilizado para determinar o formato do *frame*, adotando dominante para o formato padrão e recessivo para estendido.

Campo de Controle

É composto por 6 bits, sendo os dois primeiros indefinidos, reservado para futuras aplicações e os quatro últimos designados para informar a quantidade de *bytes* existentes no campo de dados.

Campo de Dados

Encapsula os parâmetros (como velocidade, rotação do motor e etc) a serem transmitidos. É neste campo que contém a magnitude das informações contidas nos *frames*. O comprimento deste campo varia de 0 a 8 *bytes* de acordo com o valor configurado no campo de controle.

Campo CRC (*Cyclic Redundancy Check*)

É composto por 15 bits para a verificação de redundância cíclica que é responsável pela identificação de erros na transmissão do *frame*, além de um *bit* recessivo para indicar o término do campo.

Campo ACK (*Acknowledgment Error Check*)

Consite em dois bits *ACK slot* e *ACK delimiter*: ACK slot atua no momento em que o nó receptor confirma que não houve erro na transmissão do *frame* e assim substitui o bit recessivo que se encontrava no campo ACK por um dominante. O ACK *delimiter* é um bit que deve sempre ser recessivo.

Fim do Frame

É uma sequência de 7 bits recessivos que indicam o fim do *frame*.

2.3.2.3 Verificação de erros

O protocolo CAN possui 5 métodos para verificação de erros de transmissão ou de construção da mensagem. De acordo com Lopes (2007) estas verificações fazem com que esse protocolo possua grande capacidade de se adaptar a situações adversas que podem ocorrer durante o funcionamento da rede. Os métodos utilizados são:

- **Monitoramento de *bit* -** Após enviar um *bit* dominante a ECU emissora verifica o estado do barramento. Se o *bit* lido for recessivo houve alguma falha e então a ECU emite um *frame* de erro.
- ***Bit Stuffing* -** Após a transmissão de 5 bits consecutivos idênticos o sexto *bit* é obrigatoriamente complementar aos anteriores. Em outras palavras, caso haja a necessidade de uma ECU transmitir mais de 5 bits consecutivos a unidade insere um *bit* complementar (*stuff bit*), a cada sequência de 5 *bits*, e assim cabe a ECU receptora retirar este *bit* adicionado para então e interpretar a mensagem.
- **CRC** A ECU emissora calcula um valor, baseado num polinômio, em função dos *bits* encaminhados na mensagem e esse valor é enviado junto ao *frame*. A ECU receptora faz o mesmo procedimento e se esses valores forem divergentes ocorreu algum erro durante transmissão.
- **Checagem de *frame* -** As ECUs receptoras verificam o sinal de *bits* que possuem valores fixos segundo o padrão utilizado, por exemplo o *frame* de início da mensagem que sempre possui valor dominante no protocolo CAN. Caso esses valores não estejam de acordo com a padrão é verificado um erro.
- **ACK** Após a leitura de uma mensagem sem erros as ECUs receptoras marcam o campo ACK com um *bit* dominante e enviam uma mensagem de resposta a ECU emissora. Se a ECU emissora receber uma mensagem de resposta com o campo ACK contendo um *frame* recessivo significa que a mensagem enviada inicialmente estava corrompida ou nenhuma unidade recebeu a mensagem.

2.4 SAE J1939

Como o protocolo CAN define apenas duas camadas do modelo OSI/ISO, as outras camadas ficam reservadas para aplicações específicas. Existem diversos protocolos que definem as camadas de aplicações do modelo OSI/ISO tais como CANopen, DeviceNet, SAE J1939 entre outras que são utilizadas em diversas áreas, mas utilizam o protocolo CAN como padrão para as camadas físicas e de enlace (HU et al., 2007).

SAE J1939 são práticas recomendadas para padronizar a interconexão entre ECUs em veículos comerciais, principalmente em ônibus, caminhões e equipamentos agrícolas

(JUNGER, 2010). Existem algumas extensões da J1939 para aplicações ainda mais específicas como a ISO11783 utilizadas em máquinas e equipamentos agrícolas, ISO11992 para veículos que tracionam reboque entre outras (JUNGER, 2010).

Algumas características particulares da J1939 são destacadas por (JUNGER, 2010).

- Utiliza o protocolo CAN como camada física.
- Identificador estendido (29 bits).
- Taxa de transmissão 250kbit/s.
- Comunicação *peer-to-peer* e/ou *broadcast*[2]

Existe uma série de documentos do padrão J1939 que especificam as camadas do modelo OSI/ISO estabelecendo as práticas necessárias para utilização do padrão. A lista a seguir descreve os principais documentos que incorporam o padrão J1939.

- J1939/1X – Definem características da camada física.
- J1939/21 – Camada de enlance.
- J1939/31 – Camada de rede.
- J1939/71 – Camada de aplicação.
- J1939/73 - Diagnóstico da camada de aplicação.
- J1939/81 – Gerenciamento de rede.

Mensagem no padrão J1939

Baseado na especificação CAN 2.0B, as mensagens no padrão SAE J1939 também são enviadas na forma de *frames*. O campo identificador no SAE J1939 é composto basicamente pelo número do grupo de parâmetros (PGN), endereço da ECU que realizou a transmissão e prioridade do *frame* como mostrado na Figura 7.

O padrão SAE J1939 adiciona o conceito de Grupo de Parâmetros (PGs), mecanismo no qual parâmetros distintos são reunidos em grupos e transmitidos em um mesmo *frame*. Esses grupos são identificados por números denominados *Número do Grupo de Parâmetros* (PGNs) os quais identificam unicamente cada PG. Um exemplo de PG definido pela J1939, referente a temperatura do motor, é mostrado na Figura 8 cujo PGN é

[2] *peer-to-peer* é a mensagem encaminhada para uma ECU exclusiva. *Broadcast* é uma mensagem encaminhada a um grupo de ECUs.

65,262. O campo de dados desse *frame* contém informações de diversos parâmetros, tais como temperatura do líquido do motor, temperatura do combustível entre outros.

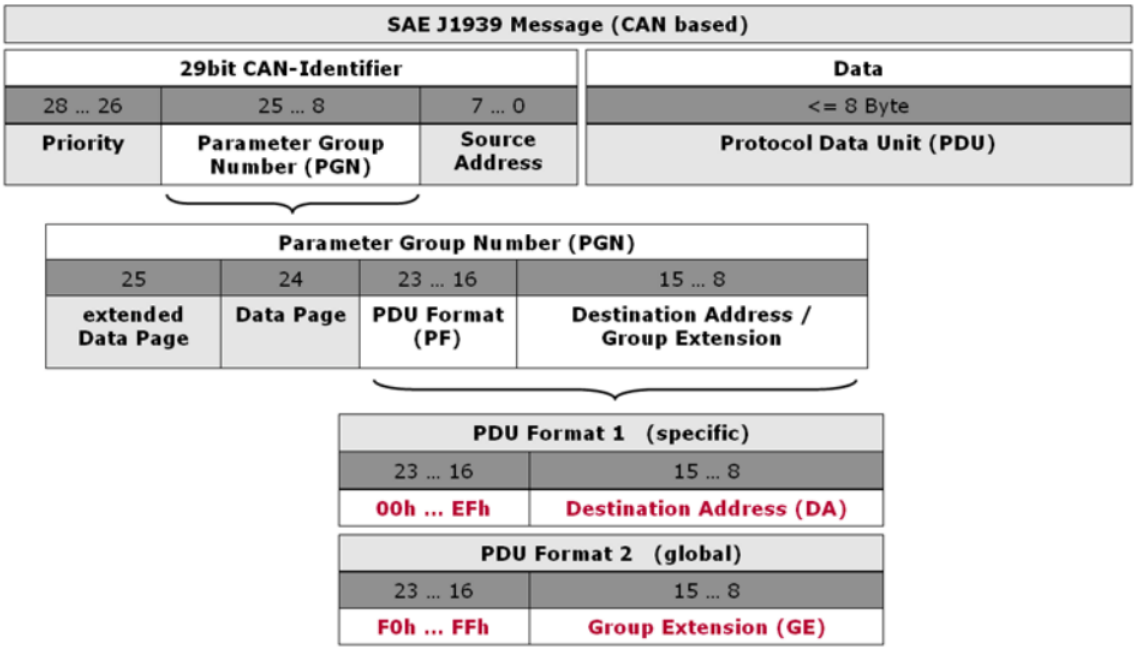


Figura 7 – Formato do frame no J1939 (FELLMETH; LöFFLER, 2008).

Os três primeiros bits são os mais significativos e têm a função de definir a prioridade do *frame* no barramento, embora como afirma (FELLMETH; LöFFLER, 2008) esses bits não substituam o mecanismo de prioridade imposto pelo protocolo CAN, eles garantem apenas que a prioridade entre os PGNs descritos pela J1939 seja atendida.

O próximo campo encapsula o PGN a ser atribuído a um determinado *frame* e tem o papel de identificar o conteúdo da mensagem no *frame* de dados, ou seja, identifica se o *frame* contém informações de velocidade, rotação do motor, temperatura do óleo ou qualquer outro parâmetro definido na SAE J1939 (ÖZCAN; GüNAY, 2012).

A Figura 7 também mostra que o campo PGN é subdividido em *PDU Format* (8 bits), *PDU Specific* (8 bits), *Data Page* (1 bit) e *Extended Data Pagee* (1 bit), sendo o *Extended Data Page* reservado para aplicações futuras e o *Data Page* expande o número de PGs que podem ser representados pelo identificador. Os campos PDU *Format* e *Specific* possuem dois cenários como define (JUNGER, 2010).

- Se o valor decimal do PDU *format* for igual ou superior a 240 a mensagem é do tipo *broadcast* e o campo PDU *specific* se refere a extensão de um grupo que deve receber a mensagem.
- Se o valor decimal do PDU *format* for inferior a 239 a mensagem é do tipo *peer-to-peer* e o PDU *specific* contém o endereço da ECU que deverá receber a mensagem.

O último campo *Source Adress* tem a função de identificar a ECU que realizou a transmissão do frame.

SAE J1939/71

O SAE J1939/71 é o documento que especifica a camada de aplicação (modelo OSI/ISO) do SAE J1939. Este documento contém informações essenciais para se realizar o desenvolvimento da rede de comunicação como: PGNs, taxas de transmissão dentre outros parâmetros (FRITZ KüBLER GMBH, 2009). A Figura 8 mostra um exemplo da especificação de um PGN de acordo com o SAE J1939/71.

Nome: Temperatura do motor
Taxa de transmissão: 1s
Comprimento campo de dados: 8 bytes
Formato PDU: 254
PDU específico: 238
Prioridade padrão: 6
Número PG: 65,262(FEEE16)
Descrição do campo de dados:

(Byte nr.)		**SPN**
1	Temperatura do líquido de arrefecimento do motor	110
2	Temperatura do combustível	174
3,4	Temperatura do óleo do motor	175
5,6	Temperatura óleo do turbo	176
7	Temperatura *Intercooler* do motor	52
8	Termostato do *Intercooler* do motor aberto	1134

SPN 174 Temperatura do Combustível
Comprimento do parâmetro: 1 Byte
Resolução: 1 ºC / *bit*; 40 ºC de *offset*
Intervalo de dados: -40 a 210 ºC
Tipo: Medido

Figura 8 – PGN - Temperatura do motor. (FRITZ KüBLER GMBH, 2009).

O número deste PG é $FEEE_{16}$, que identifica o *frame* responsável por armazenar informações relativas a temperatura do motor, em que são especificadas: a taxa de transmissão de 1s, Formato do PDU de valor 254 caracterizando uma mensagen do tipo *broadcast*, o PDU específico (238) que define o grupo de ECUs que receberá a mensagem, a prioridade do *frame* frente a outros PGs e por fim são definidos os 8 *bytes* do campo de dados, mostrando os *bytes* específicos para cada parâmetro que compõe o *frame*.

Cada um dos parâmetros agrupados no *frame* é identificado por um SPN (*Suspect Parameter Number*), que é um conjunto de especificações do parâmetro definido no documento SAE J1939/71. Por exemplo, o parâmetro temperatura do combustível possui SPN de número 174. O documento SAE J1939/71 especifica que este SPN ocupa 1 dos 8 *bytes* do campo de dados do *frame*, especifica também a resolução do SPN em que neste caso para cada variação de 1 bit a temperatura se altera em 1 grau *Celsius*, para este caso existe o *offset* de forma que para cada temperatura calculada deve haver a subtração em 40º para assim corresponder a temperatura real do combustível. Por fim são definidos o intervalo de dados do SPN, em que neste caso pode variar de -40ºC a 210ºC.

2.5 Análise do desempenho de redes automotivas

Entender o comportamento de um sistema é de fundamental importância para conhecer a capacidade do mesmo. Testar a capacidade de um sistema já implementado demanda na grande maioria dos casos de tempo e recurso. Desta forma, analisar o comportamento do sistema através da utilização de modelos é uma alternativa que tem se mostrado eficaz uma vez que possui a capacidade de representar com alto grau de confiabilidade o sistema real, quando desenvolvido de forma satisfatória (GODOY, 2007).

Para avaliação do desempenho, por exemplo do controlador de um sistema de controle, devem-se definir quais parâmetros são confiáveis para representar o comportamento do modelo. Outra vertente importante na avaliação é definir o método no qual serão realizadas estas análises, tais como métodos analíticos, simulações dentre outros (GODOY, 2007).

Utilizar a simulação como método de análise tem se mostrado uma alternativa eficaz devido principalmente ao fato de permitir testar o sistema para as mais variadas condições de funcionamento. Com aumento da utilização do protocolo CAN e variedade de aplicações em que tem sido utilizado, aplicar análises de desempenho têm se tornado cada vez mais importante na avaliação destes sistemas (LOPES, 2007).

De acordo com Godoy (2007) nos protocolo CAN os parâmetros a serem analisados se dividem em dois grupos:

Parâmetros de configuração: - Dados de configuração (de entrada) do protocolo necessário para operação da rede.

Parâmetros de Desempenho: - Referem-se aos dados relacionados à operação da rede e o comportamento do sistema. Dados de saída.

A tabela 2 mostra o conjunto de parâmetros de configurações da rede CAN que Godoy (2007) adota como importantes para serem considerados na análise de desempenho:

Tabela 2 – Parâmetros importantes de configuração de redes CAN. (GODOY, 2007)

Nome	Descrição
Formato do *frame*	Formato padrão (CAN2.0A) ou Formato estendido (CAN2.0B)
Número de mensagens	Número total de mensagens de dados que trafegam pela rede
Tipo de mensagem	Periódica ou aperiódica
Taxa de transmissão	Valor da taxa de transmissão de dados pelo barramento
Período ou tempo de amostragem	Intervalo de tempo entre transmissões consecutivas de uma mensagem
Esquema de prioridade	Prioridade de acesso das mensagens que trafegam pelo barramento
Tamanho do campo de dados das mensagens	Quantidade de bits de dados (campo de dados) de uma dada mensagem

De acordo com Lian, Moyne e Tilbury (2002), um dos parâmetro de configuração que possui maior impacto no desempenho da rede é o período ou tempo de amostragem da mensagem. A Figura 9 mostra a influência do tempo de transmissão das mensagens para o desempenho da rede.

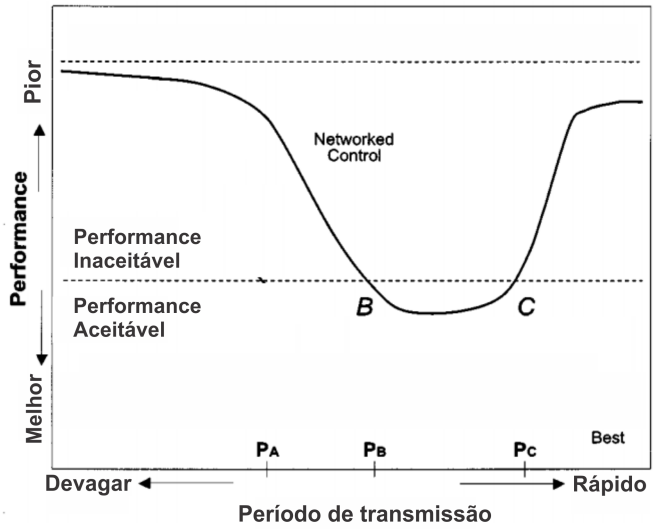


Figura 9 – Performance da rede. Adaptado de Lian, Moyne e Tilbury (2002).

Com base no gráfico é possível perceber que em períodos mais lentos (da origem ao ponto B) a rede pode apresentar menos problemas em seu funcionamento, todavia esta configuração faz com que a rede possua altos níveis de ociosidade, uma vez que a capacidade total da rede não é explorada.

Do ponto C em diante, apesar da diminuição na ociosidade, a rede volta a apresentar desempenho desfavorável devido a grande quantidade de *frames* sendo transmitidos

simultaneamente, ocasionando mais choques de mensagem no acesso ao barramento e consequentemente atrasos no tempo de comunicação.

Há diversos parâmetros que são adotados como medidores de desempenho das redes de comunicação, todavia definir aqueles que devem ser adotados requer uma ponderação por parte dos projetistas a fim de definirem quais são os mais representativos.

A tabela 3 mostra os parâmetros que segundo Godoy (2007) são comumente adotados em análises de desempenho de redes CAN.

Tabela 3 – Parâmetros de desempenho de redes CAN.(GODOY, 2007)

Nome	**Descrição**
Tempo de transmissão	Tempo entre o início da transmissão de uma mensagem e o recebimento da mesma
Tempo de resposta total	Tempo total de leitura ou transmissão de mensagens de todo os dispositivos da rede
Taxa de utilização	Porcentagem utilizada pela comunicação de dados em relação à capacidade total de uma rede
Número de mensagens enviadas e recebidas	Relação entre o número de mensagens enviadas e recebidas pelos nodos da rede
Cumprimento de *deadline*	Verifica-se o cumprimento do requisito tempo de cada mensagem
Número de mensagens no *buffer*	Monitoramento do número de mensagens nos *buffers* de recepção e transmissão de cada ECU da rede.

Portanto definir quais os parâmetros são mais importantes para medir o desempenho de determinada rede de comunicação é papel do desenvolvedor, que pode ponderar os parâmetros segundo os requisitos do projeto.

2.5.1 Tempo de transmissão

As equações que definem o tempo de transmissão de uma mensagem no pior caso são definidas por Tindell, Burns e Wellings (1995) como:

$$R_m = J_m + w_m + C_m \tag{1}$$

O termo J_m é o *jitter* da mensagem, o qual corresponde ao atraso na entrega de pacotes de mensagens no pior caso, este valor é determinado empiricamente e geralmente utiliza-se o valor de 0,1 milissegundo. O termo w_m representa o tempo de espera da mensagem na fila de transmissão no pior caso, ou seja, o maior tempo entre o instante em

que é inserida na fila de prioridades até o início da transmissão. O termo C_m representa o maior tempo para transmissão de uma mensagem (m) sobre o barramento físico.

De acordo com os trabalhos de Tindell e Burns (1994) o pacote de mensagens no protocolo CAN possui 47 bits de *overhead* e *stuff* com largura de 5 bits. Resultando para o termo C_m a seguinte equação:

$$C_m = \left(\left[\frac{34 + 8s_m}{5} \right] + 47 + 8s_m \right) \tau_{bit} \tag{2}$$

O termo s_m corresponde ao comprimento da mensagem m em bytes. O termo τ_{bit} se refere ao tempo necessário para transmitir um bit sob o barramento. Por exemplo, para taxas de transmissão de 1 Mbit/s o bit percorre o barramento em 1 milissegundo.

O tempo de espera de uma mensagem m na fila de prioridades é dado por:

$$w_m = B_m + \sum_{\forall j \epsilon hp(m)} \left[\frac{w_m + J_j + \tau_{bit}}{T_j} \right] C_j \tag{3}$$

O termo *hp(m)* é o grupo de mensagens com prioridades superior a mensagem m. Tj se refere ao período da mensagem m. Jj é o jitter de uma mensagem. Bm é o tempo de bloqueio no pior caso da mensagem m é dado por:

$$B_m = \underset{\forall k \epsilon lp(m)}{max} (C_k) \tag{4}$$

O termo *lp(m)* se refere ao conjunto de mensagens com prioridade inferior a mensagem m. Os valores de C_k e C_j podem ser obtidos utilizando a equação 2.

Na equação 3 o termo w_m aparece em ambos os lados da equação o que requer a utilização de uma relação de recorrência conforme apresentado na equação 5.

$$w_m^{n+1} = B_m + \sum_{\forall j \epsilon hp(m)} \left[\frac{w_m^n + J_j + \tau_{bit}}{T_j} \right] C_j \tag{5}$$

2.5.2 Tempo de transmissão sob condições de erro

Durante a o funcionamento da rede CAN podem surgir erros de diversas naturezas que comprometam a transmissão das mensagem. Para aplicações automotivas, uma fonte que pode causar erros de transmissão no barramento CAN é a interferência eletromagnética, devido principalmente a presença de aparelho como celulares e rádios no interior do veículo, além de grande quantidade de dispositivos elétricos utilizados nas mais diversas

aplicações (PUNNEKKAT; HANSSON; NORSTROM, 2000). O tempo de transmissão sob condições de erro é definido em Tindell e Burns (1994) como:

$$R_m = J_m + w_m + C_m \tag{6}$$

Os termos J_m e C_m são os mesmos utilizados para a Equação 1, porém o termo w_m aplicado sob condições de erro pode ser calculado através da seguinte equação:

$$w_m^{n+1} = B_m + \sum_{\forall j \epsilon hp(m)} \left\lceil \frac{w_m^n + J_j + \tau_{bit}}{T_j} \right\rceil .C_j + E_m(w_m + C_m) \tag{7}$$

O termo E_m representa o tempo máximo necessário para detecção e recuperação do erro em um intervalo de tempo t e é dado por:

$$E_m(t) = \left(n_{error} + \left\lceil \frac{t}{T_{error}} \right\rceil - 1 \right) (31_{bit} + \max_{\forall k \epsilon hp(m) \cup m} (C_k)) \tag{8}$$

n_{error} é o número de erros em sequencia que podem ocorrer em um intervalo arbitrário.

T_{error} período de ocorrência do erro.

Em cada erro, o *overhead* de recuperação de erros pode ser aumentado em 31 *bits* seguidos pela retransmissão da mensagem. Apenas mensagens de prioridade maior que a mensagem m podem ser retransmitidas e atrasar a mensagem m. A maior destas mensagens é dada pela equação 9

$$\max_{\forall k \epsilon hp(m) \cup m} (C_k) \tag{9}$$

2.5.3 Taxa de utilização do barramento

A taxa de utilização do barramento é outro parâmetro importante na análise de redes automotivas e mede o carregamento relativo à situação onde o barramento está totalmente carregado. Por exemplo um barramento com 25% de carregamento significa que durante 25% do tempo o barramento é utilizado para a transmissão de mensagens, enquanto que para 75% do tempo o barramento está ocioso (HOFSTEE; GOENSE, 1999).

A taxa de utilização é definida em Godoy (2007) como sendo:

$$U = \sum_{Ti}^{N} \frac{Ci}{Ti} \times 100 \tag{10}$$

Em que:

Ci = Tempo de tranmissão da mensagem i dado pela Equação 2;

Ti = Período de amostragem da mensagem i (parâmetro da rede);

N = Número total de mensagens na rede.

2.6 Benefícios e desafios da simulação de redes automotivas

A simulação de redes automotivas traz uma série de benefícios que impactam diretamente na redução de tempo e custos tanto na fase de desenvolvimento quanto na avaliação de sistemas de comunicação veicular. Emaus (2008) e Klüser (2004) listam os principais benefícios da simulação de redes automotivas:

- Permite simular um sistema de redes automotivas com múltiplas ECUs. Além da capacidade de integrar ECUs que trabalham com diferentes protocolos tais como CAN, LIN dentre outros.
- Avaliação dos canais de comunicação verificando a integridade dos sinais durante a transferência de informações entre as unidades.
- Análise dos parâmetros de desempenho da rede.
- Permite avaliar o carregamento no barramento.
- Adquire-se conhecimento das situações críticas de carregamento em que os dispositivos físicos estarão submetidos e a partir daí permite projetá-los de forma a suprir as condições de operação solicitadas.
- Os desenvolvedores também podem se beneficiar e otimizar o projeto de novas aplicações através da reutilização de mecanismos abordados em simulações anteriores.
- Possibilita a integração total ou parcial entre nós reais e virtuais.
- As ECUs fabricadas pelos fornecedores e adquiridas pelos OEM(*Original Equipament Manufaturer*) podem apresentar erros no momento da implementação, devido principalmente ao não cumprimento dos requisitos de projeto, utilizar as ferramentas de simulação é uma forma de contornar este problema, pois os requisitos são detalhados de forma clara e o erros potenciais são detectados durante a simulação.

Desafios na simulação de redes automotivas também são citados por Emaus (2008), Klüser e Fetzër et al (2004).

- Necessidade de *hardwares* específicos e seus respectivos *drivers* para integração de ECUs reais à simulação.

- Dificuldade na simulação de funções que envolvem controle como ABS, ESP, controle de tração e etc.

2.7 Ferramentas para a simulação de redes automotivas

Existem diversos *softwares* disponíveis para análise e simulação de redes automotivas. Estas ferramentas possibilitam ao usuário reproduzir de maneira adequada o comportamento da rede de um veículo real, abordando todos as mensagens e sinais que são transmitidas pelas diversas ECUs. Além de controlar o disparo dos *frames* de forma a representar o funcionamento do veículo em determinadas situações como acelerar, freiar, acionar os dispositivos de iluminação e etc. Estes *softwares* permitem também uma análise detalhada do comportamento da rede quanto ao carregamento do barramento, quantidade de mensagens transmitidas, cumprimento do *deadline* entre outras métricas de desempenho.

2.7.1 CANoe

O CANoe® é uma ferramenta proprietária criada pela companhia alemã Vector® e tem por objetivo o desenvolvimento, teste e análise de redes automotivas e ECUs individuais através da simulação. Este *software* permite desenvolver um sistema composto apenas por ECUs virtuais quanto a integração de nós reais com virtuais(ZHOU; LI; HOU, 2008). Além de simular vários nós, o CANoe também possibilita simular nós com diferentes protocolos, por exemplo, para um veículo que possua o protocolo CAN no gerenciamento do motor, LIN para dispositivos de entretenimento e GPS e FlexRay no controle da suspensão pode ser modelado com todas suas funcionalidades no CANoe (VECTOR, 2009). No primeiro momento do desenvolvimento a ferramenta é utilizada para simular o modelo de funcionamento das ECUs, em seguida esses modelos são analisados e integrados a outras unidades a fim de constituir uma rede. Essa metodologia possibilita principalmente a detecção e correção de erros de forma rápida (VECTOR, 2014).

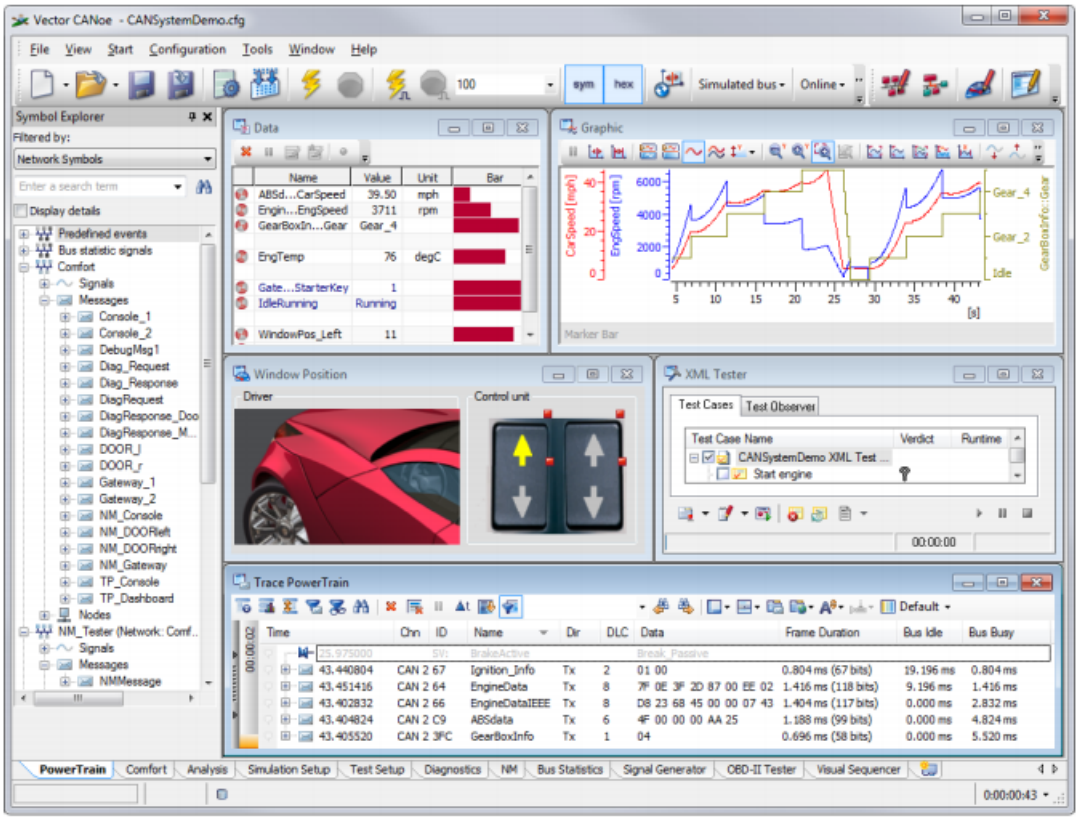


Figura 10 – Interface *software* CANoe. (VECTOR, 2014).

2.7.2 Bus Master

O BusMaster® é uma ferramenta desenvolvida em conjunto por ETAS, *Robert Bosch Engineering* and *Business Solutions* (RBEI) que permite o projeto, monitoramento, análise e simulação de redes automotivas. É um software de código aberto, livre para contribuições de pesquisadores, estudantes, indústrias entre outros que tenham interesse em adicionar ou gerenciar funcionalidades a esta aplicação que é desenvolvida em uma arquitetura modular[3]. A Figura 11 mostra a interface principal do Bus Master.

[3]Divisão do software em partes com funções bem definidas e o mais independente possível do restante do algoritmo, facilitando a interpretação, modificação, inclusão de novas funcionalidades entre outros benefícios.

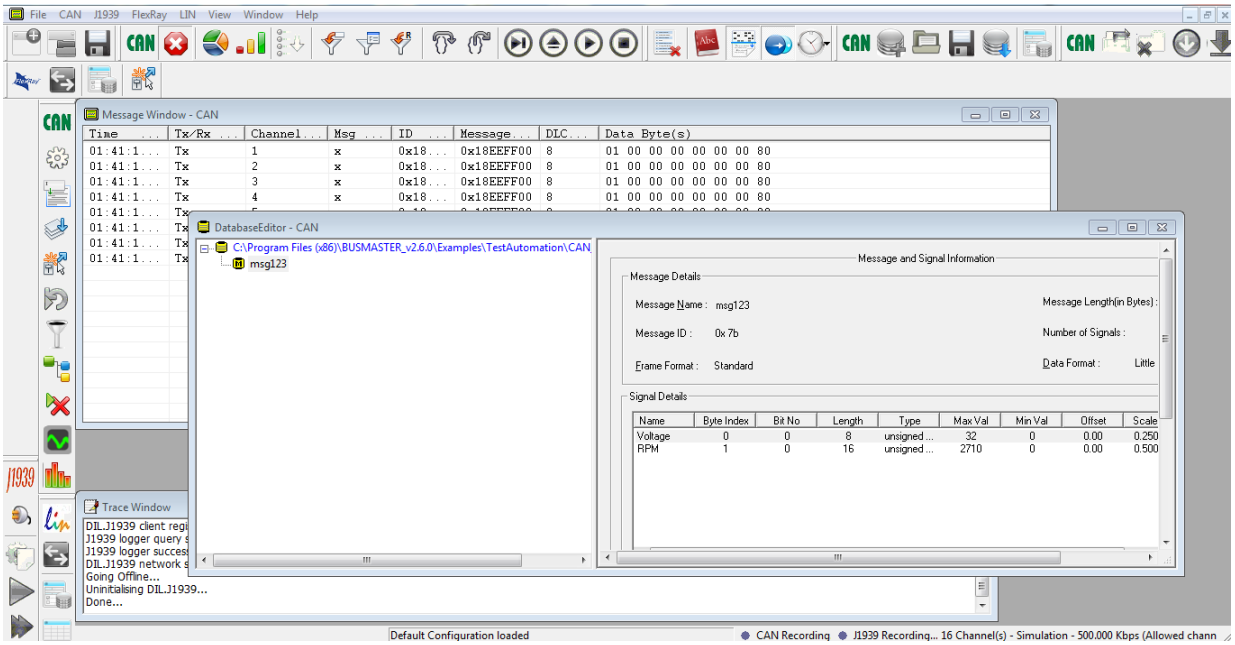


Figura 11 – Interface do *software* Bus Master.

As principais funcionalidades do *software* BusMaster são:

- Integração de *hardwares* conectados ao computador para a simulação e monitoramento da rede.
- Compatível com diversos tipos de protocolo: CAN 2.0A e 2.0B, FlexRay, Lin e etc.
- As mensagens podem ser exibidas em formato decimal facilitando a visualização da informação para o usuário ou em formato hexadecimal.
- Monitoramento de erros ocorridos ao longo da transmissão.
- Mecanismos para filtro de mensagens, nos quais podem ser definidos os *frames* específicos que serão destacados e exibidos para o usuário.
- O filtro de mensagens, que define quais *frames* devem ser exibidos, podem ser feitos tanto a nível de software quanto a nível de hardware, caso este esteja presente na simulação.
- As mensagens podem ser destacadas e analisadas individualmente utilizando o *time stamp*[4].
- Possui editores que permitem definir cores específicas para *frames*, de acordo com os IDs (campo de identificação do *frame*).
- Atende parcialmente as métricas de desempenho.

[4]Mecanismo que define o tempo (baseado num cronômetro gerado pelo próprio software) em que determinado *frame* foi transmitido

3 Metodologia

Este capítulo mostra a metodologia para a realização deste trabalho. De forma que são explicitadas as 5 grandes etapas, bem como o detalhamento de cada etapa.

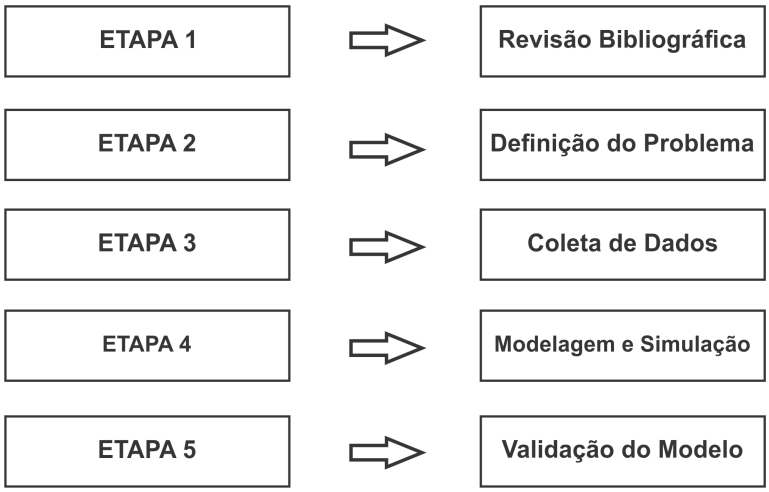


Figura 12 – Etapas de elaboração do trabalho.

3.1 Revisão bibliográfica

Nesta etapa foram revisadas as bibliografias pertinentes ao desenvolvimento deste trabalho. Abordou-se primeiramente estudos sobre o avanço da tecnologia nos automóveis, o que gerou uma demanda por dispositivos controladores denominados ECUS. Com o aumento da quantidade de ECUs e a necessidade da troca de informações entre essas unidades surgiram as redes automotivas.

Explorou-se os principais protocolos de comunicação utilizados em aplicações automotivas, observando suas características e aplicabilidades. Explorou-se o protocolo CAN e suas principais características de modo que este foi o protocolo utilizado nos estudos de caso deste trabalho. Em seguida foram explorados dois *softwares* de simulação de redes automotivas sendo eles *CANoe* e o *BusMaster*, ressaltando as características, benefícios e desafios da utilização destas ferramentas no desenvolvimento de uma rede veicular. Abordou-se também as métricas de desempenho que avaliam o comportamento da rede segundo alguns parâmetros de funcionamento.

3.2 Definição do problema

Esta etapa consistiu basicamente da definição do problema a ser explorado, ou seja, foram definidas as redes automotivas que foram utilizadas como estudo de caso. De forma

que definiu-se as redes automotivas de três caminhões com diferentes configurações de rede. O primeiro e o segundo estudo de caso foram realizados com as redes automotivas de dois caminhões Scania, as quais apresentam algumas ECUs em comum. Todavia o caminhão utilizado no caso 2 conta com uma rede automotiva mais desenvolvida em termos de quantidade de ECUs e de mensagens quando comparada a rede caso 1. Para o caso 3 modelou-se a rede baseando-se na rede automotiva de um caminhão real. O *software* utilizado para modelagem e análise das redes automotivas foi o *BusMaster*, apesar de ser uma ferramenta ainda em desenvolvimento, o principal motivo de sua utilização foi o fato de ser uma ferramenta gratuita e de código aberto, além de apresentar diversos recursos de modelagem e análise de redes que foram fundamentais para a realização dos estudos de caso. Nesta etapa definiu-se também quais métricas de desempenho foram adotadas para análise do comportamento das redes modeladas.

3.3 Coleta de dados

Esta etapa consistiu em obter informações relativas às redes automotivas exploradas nos estudos de caso. O fabricante dos caminhões Scania disponibiliza informações relativas as redes automotivas de seus veículos, de modo que foi possível modelar a rede observando quais *frames* são emitidos pelas ECUs presentes nos caminhões, portanto as redes dos caminhões para os casos 1 e 2 foram modeladas a partir das informações do fabricante. No caso 3 a rede automotiva foi modelada com base nas informações do próprio veículo de modo que através de equipamentos de aquisição de dados obteve-se o registro dos *frames* que transitaram no barramento deste veículo e de posse dessas informações modelou-se uma rede automotiva correspondente.

3.4 Modelagem e Simulação

Esta etapa consistiu na modelagem, simulação e análise das redes automotivas no *software* BusMaster. Optou-se por este *software* devido principalmente ao fato de ser uma ferramenta gratuita, de código aberto e capaz de atender os principais requisitos de análise de desempenho. Modelar uma rede automotiva no software BusMaster é um processo que envolve prioritariamente a criação do *database* que se refere à definição das mensagens e sinais que podem ser transmitidos no barramento e a elaboração das rotinas de funcionamento das ECUs de forma a definir as unidades de origem, destino, tipo de mensagem e os tempos de transmissão. O Anexo A apresenta um explicativo dos passos necessários para modelagem de uma rede automotiva no *software* BusMaster.

Para a modelagem da rede automotiva no BusMaster é necessário desenvolver a programação pertinente a cada ECU, ou seja, uma rotina que estabelece o funcionamento das unidades durante a simulação. No software BusMaster a transmissão dos *frames* se

dá através dos *handlers*, ou seja, manuseadores que permitem ao usuário definir o método de disparo destas mensagens. Existem cinco tipo de *handlers* disponíveis:

- *Message Handlers*
- *Timer Handlers*
- *Key Handlers*
- *Error Handlers*
- *DLL Handlers*

Esses *handlers* após serem carregados na ECUs serão executados, respectivamente, da seguinte forma:

- Baseado na recepção de uma mensagem, ou seja, a ECU a qual está configurada com esta função emite uma mensagem após receber uma mensagem proveniente de outra unidade.
- Os *frames* são enviados em intervalos periódicos.
- Os *frames* são transmitidos através de comandos no teclado.
- Os *frames* são transmitidos após a detecção de erros.
- Responsável por carregar o código como um DLL (*Dynamic link library*).

Desta forma cada ECU é carregada com uma programação específica, em que são definidos os *frames* e os respectivos *handlers* os quais estas unidades podem transmitir. A Figura 13 mostra um fluxograma da programação presente em cada ECU da rede.

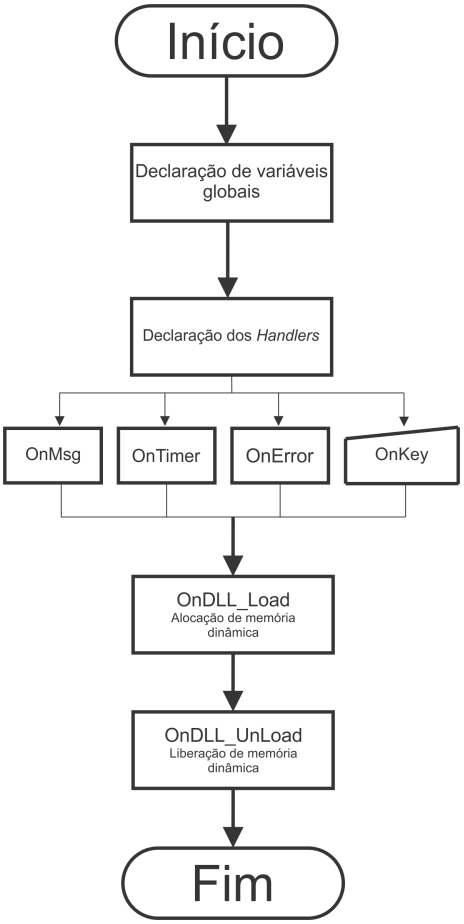


Figura 13 – Fluxograma do algorítimo das ECUs.

A programação completa de todas as ECUs simuladas no primeiro estudo de caso (Scania 164) encontra-se no Anexo B.

No que tange às métricas de desempenho o software BusMaster apresenta suporte para a análise dos parâmetros de carregamento do barramento e quantidade de mensagens enviadas, todavia a métrica que avalia o tempo de transmissão das mensagens ainda é inexistente neste *software*. Dessa forma para análise do tempo de resposta das mensagens utilizou-se a ferramenta desenvolvida em Godoy (2007), a qual calcula os tempos de transmissão das mensagens a partir das formulações pertinentes.

3.5 Validação do Modelo

Nesta etapa não foi utilizada nenhuma técnica específica de validação, todavia buscou-se assegurar que a rede modelada comporte-se o mais próximo possível da rede real. O processo de validação resultou do refino sucessivo do modelo no intuito de melhorá-lo.

4 Estudo de caso

Neste capítulo são apresentados os estudos de casos desenvolvidos para a modelagem e simulação da rede automotiva. Para demonstrar a potencialidade desta técnica, três diferentes casos foram explorados. O primeiro caso trata da modelagem e da simulação da rede de um caminhão Scania 164 que apresenta uma rede menos sofisticada em relação aos demais casos. O segundo caso trata da modelagem e da simulação da rede de veículos Scania R580 que é um caminhão moderno e apresenta uma rede mais desenvolvida se comparada ao primeiro caso. Já no terceiro estudo de caso é apresentada a modelagem e a análise da uma rede automotiva que foi desenvolvida baseando-se na rede de um veículo Volvo FH400.

4.1 Caso 1: Scania 164

A primeira rede a ser analisada será de um veículo Scania 164 (Figura 14). O caminhão 164 da Scania foi lançado em 2001 e ficou conhecido como "rei da estrada"por ser o modelo mais potente do mercado nacional na época.

Figura 14 – Caminhão Scania 164/480

Este caminhão possui motor V8 de 480 cavalos com torque de 2300 N.m. controlado eletronicamente e conta com uma série de dispositivos que não eram tão utilizados no ano de sua fabricação, tais como: freios ABS, controle de tração, freio auxiliar *retarder* dentre outros.

Para o controle destes sistemas eletroeletrônicos o veículo dispõem de algumas ECUs, sendo as princiais: EMS (*Eletronic Management System*), SMS (*Suspension Management System*) e BMS (*Brake Management System*).

A ECU responsável pelo sistema de gerenciamento do motor (EMS) é presente em grande parte dos veículos pesados da Scania que possuem motores controlados eletronicamente. Esta unidade é responsável principalmente por fazer o gerenciamento eletrônico de motores de combustão interna, obtendo dados de sensores e operando através dos atuadores na busca de fornecer a mistura ar/combustível o mais próximo possível do balanço estequiométrico (PUJATTI et al., 2012).

A ECU de gerenciamento da suspensão (SMS) trabalha em conjunto com a suspensão a ar e se faz presente com a implementação do sistema denominado ELC *eletronic level control* que é responsável por levantar ou abaixar a traseira do caminhão a fim de compensar a inserção ou retirada de carga.

A unidade de gerenciamento dos freios, *Brake Management Systems*, é responsável pelo ABS - *Anti-lock Braking Systems* e EBS - *Electronic Braking System*. O sistema ABS tem a função de evitar que uma ou mais rodas travem durante a frenagem, garantindo uma desaceleração mais eficiente. Já o sistema EBS proporciona menor tempo de resposta dos freios entre a solicitação do condutor e o acionamento dos atuadores presentes nas rodas quando comparado a sistemas puramente hidráulicos. Outro benefício do EBS é a distribuição de pressão de frenagem específica para cada roda, baseada na carga suportada por esta no momento da frenagem.

4.1.1 Modelagem da rede

Baseado em Actia (2013) que apresenta os mecanismos eletroeletrônicos presentes em diversos veículos de grande porte, é possível inferir quais ECUs compõem a rede automotiva do Scania 164, visto que, estes mecanismos são controlados por ECUs específicas. Uma vez que definiu-se quais unidades fazem parte da rede automotiva do caminhão adotou-se, para a modelagem da rede, todos os *frames* possíveis de serem emitidos por essas unidades.

Para este caso os parâmetros de configuração que definem as características de entrada da rede são mostrados na Tabela 4. Foi adotado o formato estendido para os *frames*, as mensagens enviadas são do tipo periódicas e os tempos de transmissão para cada *frame* são os definidos com base no documento J1939-71. O método de arbitragem para garantir o acesso das mensagens ao barramento é o CSMA-CD e a taxa de transmissão adotada foi de 250 Kbps, configurações estas geralmente utilizadas em aplicações do protocolo CAN.

Tabela 4 – Parâmetros de configuração da rede CAN para o caso 1

Nome	Descrição
Formato do quadro de mensagem	Formato estendido (CAN2.0B)
Número de mensagens	18 Mensagens
Tipo de mensagem	Periódica (tempo de transmissão determinado)
Velocidade de transmissão	250 Kbps
Período ou tempo de amostragem	Definido pela norma J1939
Esquema de prioridade	CSMA-CD
Tamanho do campo de dados das mensagens	64 bits

Baseado nos mecanismos eletroeletrônicos presentes no veículo foram definidas quatro ECUs para a rede automotiva deste veículo conforme mostra a Figura 15. A programação dessas ECUs foi desenvolvida na ferramenta do BusMaster denominada *Node Simulation* a qual é responsável por criar as funções que controlam a rotina de funcionamento das ECUs. O Anexo A apresenta a programação da ECU COO modelada para o caso 1.

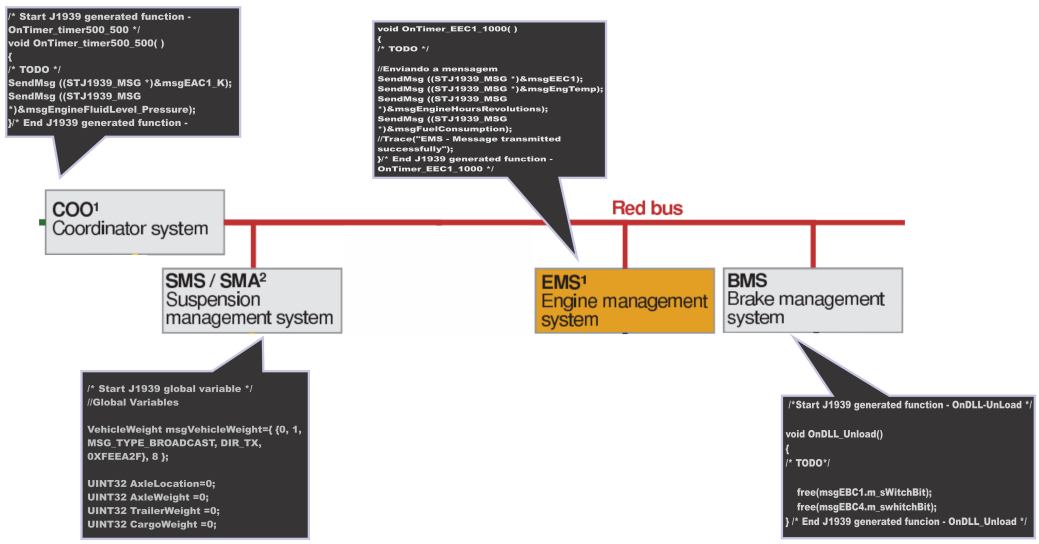


Figura 15 – Arquitetura da rede do caminhão Scania 164
Adaptado de Bartish (2011)

Os *handlers*, funções responsáveis por definir o modo com que os *frames* são enviados ao barramento, utilizados na simulação foram:

- *on timer* - Baseado no documento J1939/71 foram definidos os tempos de transmissão para cada *frame* utilizando *handlers* do tipo *on timer* em que os *frames* periódicos são disparados em um intervalo de tempo definido para cada mensagem.
- *on key* - Para representar uma solicitação externa foram adotadas as teclas "g"e

"h"responsáveis respectivamente por incrementar e decrementar a magnitude de alguns parâmetros tais como a ângulo da válvula borboleta da rotação do motor.

- on DLL - Este *handler* foi utilizado a fim de iniciar e finalizar as DLLs (*Dynamic Link Library*) que são os arquivos possuidores de recursos e códigos da ECU. Dessa forma ao executar os *handelrs on DLL* todos os *handlers* e funcionalidades das ECUs estão disponíveis para a simulação.

4.1.2 Simulação

Foi adotado tempo total de 1 minuto para a simulação. A figura 16 mostra o quadro de simulação do *software* BusMaster, em que podem ser observadas informações relativas aos campos dos *frames* bem como os tempos de transmissão das mensagens.

Destacando o *frame* de PGN FEF2 denominado *FuelEconomy* que apresenta sinais referentes ao consumo de combustível no veículo, é possível através da figura obter as informações sobre a mensagens e os sinais presentes no campo de dados. A mensagem é do tipo de dados, é uma mensagem enviada pela ECU de gerenciamento do motor (EMS), informação esta mostrada no campo de endereço (*source*). A norma define para esta mensagem o período de 100 milissegundos. O campo de dados deste *frame* possui comprimento de 8 *bytes*.

O documento J1939-71 define para este *frame* que os *bytes* 3 e 4 do campo de dados se referem ao consumo de combustível instantâneo do veículo, desta forma para o instante mostrado na Figura 16 este sinal possui valor de 3132_{16} valor que corresponde a 12529 na escala decimal. A norma estabelece a resolução de $1/512\frac{bit}{km/l}$ para este sinal, desta maneira o veículo apresenta consumo instantâneo de 24 km/l.

Time	Chan...	CAN ID	PGN	PGN Name	Type...	Src	De...	Prio...	T...	D...	Data Byte(s)
00:10:02:5273	1	0x1CFEAC02	0x00FEAC	EBC4	DATA	02	AC	007	Rx	8	00 00 00 00 00 00 00 00
00:10:02:4843	1	0xFFA02F	0x00FFA0	TransmissionPropriet...	DATA	2F	A0	000	Rx	8	00 00 00 73 74 72 69 6E
00:10:02:4844	1	0xF00304	0x00F003	EEC2	DATA	04	03	000	Rx	8	4A 00 8A 0F 38 E2 8A 0F
00:10:02:4662	1	0x18FEF204	0x00FEF2	FuelEconomy	DATA	04	F2	006	Rx	8	60 00 32 31 00 00 00 00
00:10:02:4814	1	0x18F00102	0x00F001	EBC1	DATA	02	01	006	Rx	8	00 00 30 00 38 00 00 00
00:10:02:4815	1	0xFEEA05	0x00FEEA	VehicleWeight	DATA	05	EA	000	Rx	8	00 00 00 0F 00 00 00 00
00:10:02:4813	1	0xFFB02F	0x00FFB0	CooGenInfo	DATA	2F	B0	000	Rx	8	20 00 63 61 70 61 2D 75
00:10:02:4663	1	0xFEF104	0x00FEF1	CC_VehicleSpeed	DATA	04	F1	000	Rx	8	50 00 00 21 30 00 00 00
00:10:02:4665	1	0xFF8104	0x00FF81	DLN2_Proprietary	DATA	04	81	000	Rx	8	50 00 00 30 30 30 30 00
00:10:02:2663	1	0xF0062F	0x00F006	EAC1_K	DATA	2F	06	000	Rx	8	00 00 40 70 74 72 69 6E
00:10:02:2664	1	0xFEEF2F	0x00FEEF	EngineFluidLevel_Pre...	DATA	2F	EF	000	Rx	8	00 00 00 00 00 00 00 00
00:10:02:2668	1	0xFEF52F	0x00FEF5	AmbientConditions	DATA	2F	F5	000	Rx	8	17 00 00 57 4E 4F 52 4D
00:10:02:2669	1	0xFEFC2F	0x00FEFC	DashDisplay	DATA	2F	FC	000	Rx	8	00 00 00 00 00 00 30 00
00:10:02:2673	1	0xFFAF2F	0x00FFAF	CooGenInfo2	DATA	2F	AF	000	Rx	8	11 00 72 69 6E 67 2D 6C
00:10:02:2667	1	0xFEEE04	0x00FEEE	EngTemp	DATA	04	EE	000	Rx	8	00 00 00 00 00 00 00 00
00:10:02:2670	1	0xFEE504	0x00FEE5	EngineHoursRevolutions	DATA	04	E5	000	Rx	8	00 00 00 00 30 00 30 00
00:10:02:2671	1	0xFEE904	0x00FEE9	FuelConsumption	DATA	04	E9	000	Rx	8	76 00 30 30 00 00 00 00
00:10:01:2634	1	0xF00404	0x00F004	EEC1	DATA	04	04	000	Rx	8	01 00 8A 0F 00 00 8A 02

Figura 16 – *Frames* presentes no barramento

A utilização dos *handlers on key* é de grande importância para a simulação da rede automotiva. Isto por que através destes *handlers* é possível emular o comportamento do veículo referente a variação de um parâmetro físico, por exemplo, o sinal advindo do sensor

de rotação. A Figura 17 mostra a variação dos valores de aceleração e pedal do acelerador, sinais que podem ser alterados através da utilização das teclas G e H no teclado.

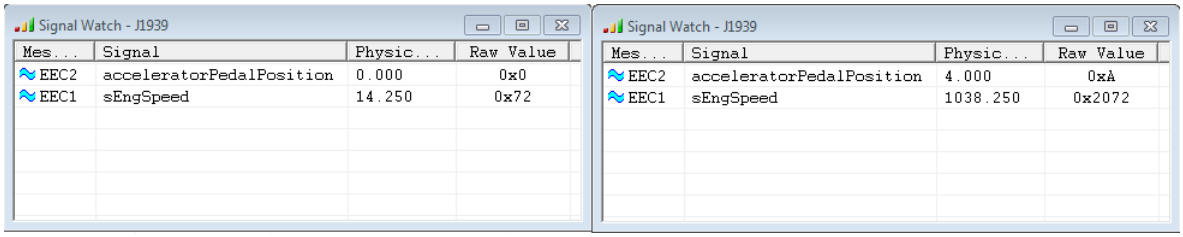
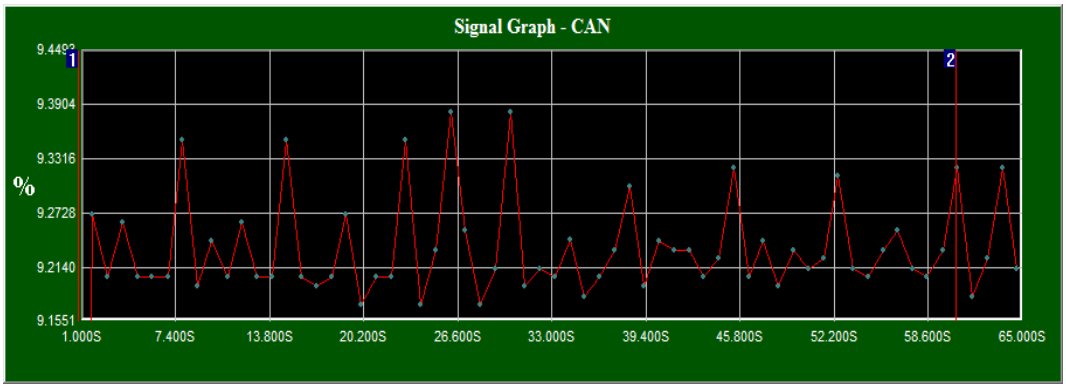


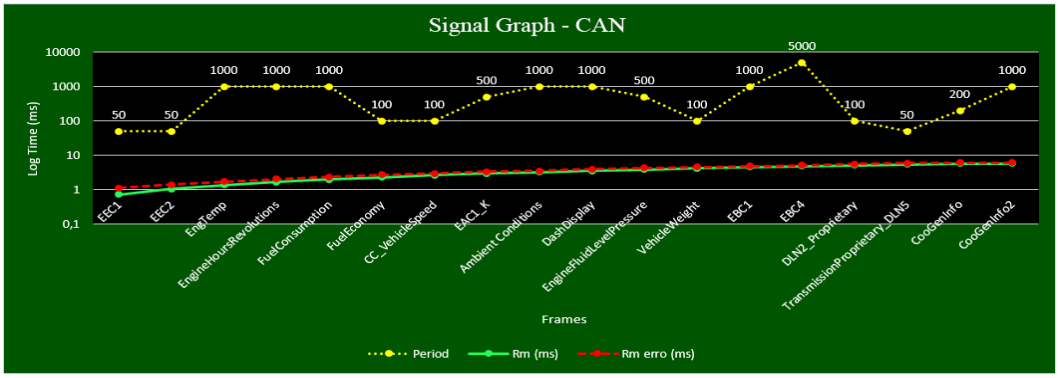
Figura 17 – Variação dos valores através do *handler on key*

4.1.3 Análise de resultados

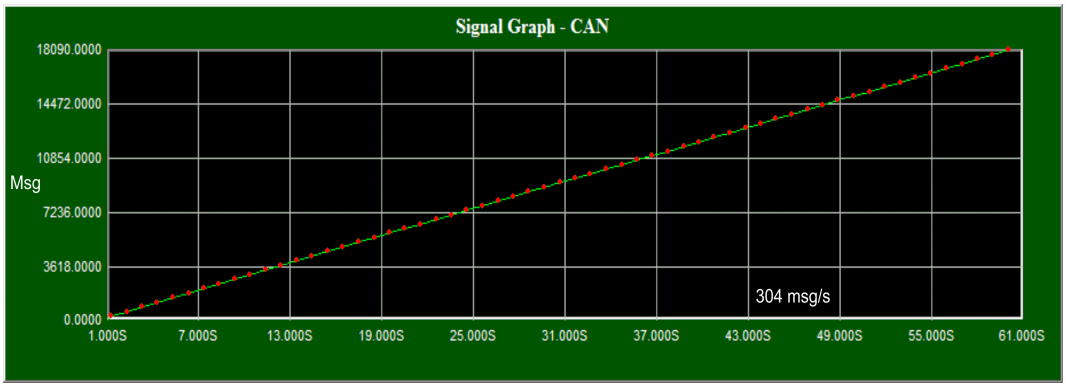
Esta simulação foi realizada para análise de três parâmetros importantes da rede automotiva que são a taxa de utilização do barramento, tempo de transmissão da mensagem e por fim a quantidade total de mensagens enviadas ao barramento.



(a) Taxa de utilização do barramento



(b) Tempo de transmissão das mensagens



(c) Quantidade de mensagens enviadas

Figura 18 – Desempenho da rede do caso 1

A Figura 18a mostra a taxa de utilização do barramento, parâmetro definido na Equação 9, que mostra o quão disponível o barramento se encontra durante o funcionamento da rede.

Pode se observar que a taxa de ocupação do barramento se manteve baixa ao longo da simulação, levando em consideração o total de 100% da rede, tendo como mínimo e máximo 9.17% e 9.38% respectivamente. Esses valores mostram que durante aproximadamente 9.22% do tempo o barramento é utilizado efetivamente para transmissão de mensagens ficando ocioso por mais de 90% do tempo (HOFSTEE; GOENSE, 1999).

Observa-se então que o barramento apresenta grande disponibilidade de banda ou capacidade disponível na rede. De fato, isto possibilita, por exemplo, inserção de novas ECUs e aumento do tráfego de dados em caso de implementação ou modificação de funcionalidades ao caminhão.

A figura 18b mostra um gráfico com o tempo de transmissão de cada mensagem no barramento em condições normais e sob condições de erro, curvas verde e vermelha respectivamente, sendo que para condições de erros são levadas em consideração o tempo de transmissão e retransmissão da mensagem. É mostrado também o período definido pela norma J1939 em que as mensagens devem estar disponíveis no barramento. Por exemplo, o *frame* EEC1 tem período definido de 50 ms e nesse caso apresenta tempo de transmissão de 0.74 ms e 1.12 ms para condições normais e sob condições de erros respectivamente. Isto mostra que esta mensagem atende o requisito temporal, pois o tempo de transmissão é menor que o período determinado, caso esse tempo fosse superior a mensagem não estaria disponível no barramento no momento adequado.

Dessa forma, percebe-se que os tempos de transmissão mantiveram-se abaixo dos períodos estabelecidos para os *frames* tanto em condições normais quanto em condições de erro em todas as mensagem. De fato, isto permite concluir, que a rede modelada apresenta um tempo de resposta satisfatório para cumprir o requisito temporal de cada mensagem.

A Figura 18c mostra um gráfico com a evolução do número de mensagens enviadas ao barramento durante o tempo de execução da simulação, a taxa de mensagens enviadas é de aproximadamente 304 mensagens/seg.

4.2 Caso 2: Scania R580

No caso 2 utilizou-se para simulação a rede automotiva do caminhão Scania R580 (Figura 19). Este veículo possui diversos adicionais de entretenimento, conforto e segurança. Além de contar com a ECU de gerenciamento da transmissão (GSM) no barramento principal, inexistente no caso anterior.

Figura 19 – Caminhão R580 Fonte: Scania

A unidade GMS *gearbox management system* é responsável pelo controle da transmissão denominada*Opticruse*, este sistema possui uma caixa de transmissão mecânica, todavia a troca de marcha é feita por atuadores elétricos ou eletro-hidráulicos controlados pela GMS. Além de controlar a transmissão esta ECU opera também no sistema de freio *Retarder* que tem a função de auxiliar os freios principais em situações de declive. (BARTISH, 2011).

4.2.1 Modelagem da rede

A Tabela 5 mostra os parâmetros de configuração para a rede modelada no caso 2, divergindo da primeira simulação apenas na quantidade de mensagens.

Tabela 5 – Parâmetros de configuração da rede CAN para o caso 2

Nome	**Descrição**
Formato do quadro de mensagem	Formato estendido (CAN2.0B)
Número de mensagens	34 Mensagens
Tipo de mensagem	Periódica (tempo de transmissão determinado)
Velocidade de transmissão	250 Kbps
Período ou tempo de amostragem	Definido pela norma J1939
Esquema de prioridade	CSMA-CD
Tamanho do campo de dados das mensagens	64 bits

Com a inclusão de novas funcionalidades, o caminhão adotado para o caso 2 apresenta uma rede automotiva mais desenvolvida, em termos de ECUs e quantidade de mensagens, quando comparada ao caso 1. Dessa forma a topologia da rede é mostrada na Figura 20.

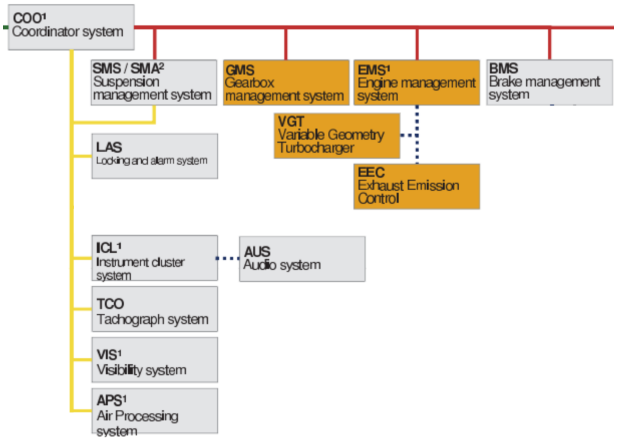
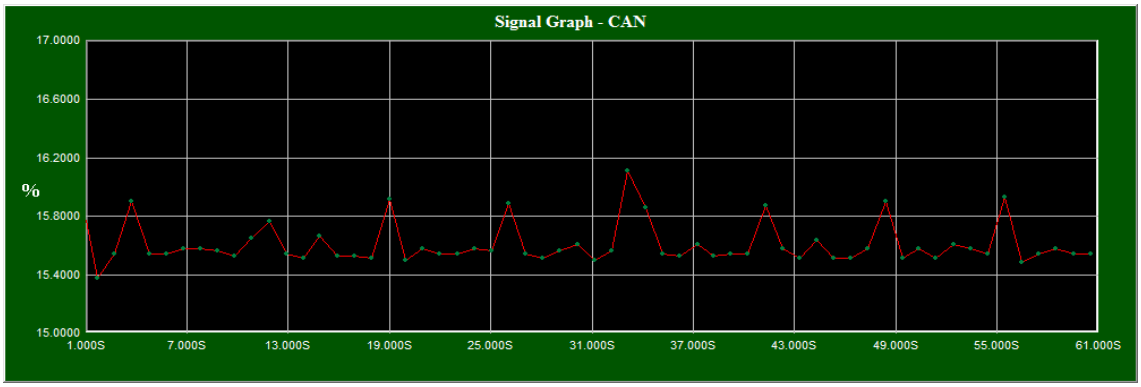


Figura 20 – Arquitetura da rede para o R580

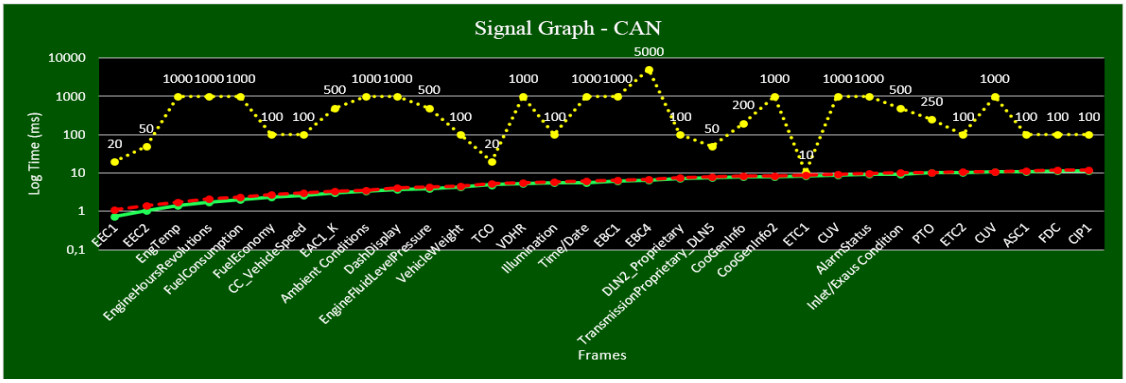
Esta rede é composta por dez ECUs as quais são dividas em dois barramentos CAN definidos como vermelho e amarelo e classificados segundo o grau de importância para veículo, de modo que o barramento vermelho conta com ECUs que impactam diretamente no funcionamento e segurança do veículo, enquanto que as unidades presentes no barramento amarelo possuem funções não tão críticas para estas finalidades. As unidades que se encontram no mesmo barramento podem se conectar diretamente umas às outras, enquanto que a troca de informações entre unidades de barramentos diferentes é realizada através da COO - *Coordinator System* (BARTISH, 2011).

4.2.2 Análise de resultados

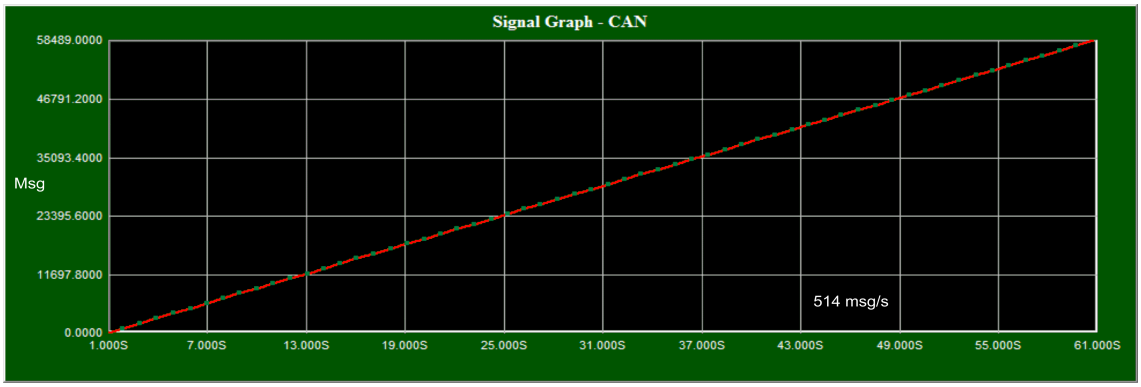
Para o caso 2 adotou-se tempo de simulação de um minuto nos Gráficos 21a e 21c de forma que os valores de carregamento e quantidade de mensagens foram tomados a cada um segundo. Analisou-se três importantes métricas de desempenho que são a taxa de utilização do barramento, tempo de transmissão das mensagens e a quantidade total de mensagens enviadas ao barramento.



(a) Taxa de utilização do barramento



(b) Tempo de transmissão das mensagens



(c) Quantidade de mensagens enviadas

Figura 21 – Desempenho da rede do caso 2

Para este caso nota-se o aumento da taxa de utilização do barramento em relação à simulação anterior, em que o carregamento médio é cerca de 14.5% como mostra a Figura 21a. Isto se dá devido ao fato da rede possuir seis ECUs a mais conectadas ao barramento e consequentemente maior número de *frames* quando comparado ao caso 1. Ainda que possua quantidade de mensagens superior ao caso 1, a rede do caso 2 também apresenta carregamento relativamente baixo o que demonstra capacidade de expansão da rede caso seja necessária a inserção de novas ECUs e/ou mensagens, visto que, o carregamento da rede ainda está distante da saturação (100% de utilização).

Como pode ser observado no Gráfico 21b grande parte das mensagens cumpriram o

requisito temporal de transmissão, ou seja, os *frames* estarão disponíveis para as ECUs no período estipulado tanto para condições normais quanto para condições de erro. Todavia pode-se perceber que as mensagens TCO e ETC1, que contêm informações do tacógrafo e da transmissão eletrônica respectivamente, apresentam período próximo ao tempo de transmissão, de forma que essas mensagens podem não ser entregues no tempo adequado o que poderia comprometer o funcionamento de alguns subsistemas. Dessa maneira algumas alternativas que podem ser levadas em consideração para garantir a entrega da mensagem dentro do tempo determinado seriam aumentar aumentar o período das mensagens ou elevar a prioridade das mensagens frente à outros *frames*.

O Gráfico 21c mostra q quantidade de mensagens enviadas ao barramento, a rede apresenta uma média 514 Msg/s sendo a unidade GMS responsável pela maior quantidade de mensagens enviadas por segundo, isto se dá devido a presença do *frame* ETC1 referente ao controle eletrônico da transmissão que possui um período de 10 ms sendo a menor taxa dentre todas as mensagens.

4.3 Caso 3: Volvo FH400

O caso 3 apresenta o modelo da rede do veículo Volvo FH 400 (Figura 22) . Este veículo conta com um motor 6 cilindros de 12.8 litros controlado eletronicamente que fornece uma potência de 400 cavalos e 2000 Nm de torque.

Figura 22 – Volvo FH400

Este veículo conta com uma série de sistemas e dispositivos eletroeletrônicos para as mais diversas funcionalidades tais como: ABS, EBS, sistema de gerenciamento do motor, controle eletrônico da suspensão, computador de bordo, dentre outros. Dependendo da versão podem haver outras funcionalidades que acabam por impactar diretamente na rede automotiva do veículo.

4.3.1 Modelagem da rede

Para este caso a modelagem da rede foi realizada com base nas informações obtidas de um veículo real. Para tanto, foram realizados ensaios utilizando um sistema de aquisição de dados conectado ao *software* BusMaster. Após o ensaio um arquivo Log[1] contendo as mensagens que transitaram no barramento durante o teste foi obtido. Observa-se que também durante o ensaio o veículo não foi submetido a condições de dirigibilidade extremas, dessa forma não houve situações atípicas no momento da coleta, apenas uma condução padrão.

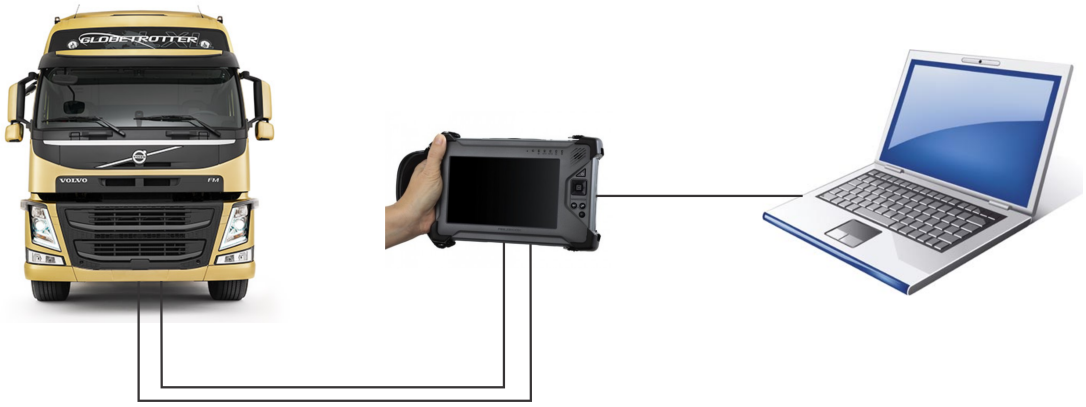


Figura 23 – Volvo FH400

Através do *software* BusMaster é possível fazer a leitura deste arquivo de forma ordenada e emular o comportamento da rede. Utilizando a opção de tempo denominada *relative time* [2] é possível determinar o período de transmissão de cada mensagem. Sendo assim o modelo foi construído com as mensagens presentes no Log do veículo e o tempo de transmissão foi definido de acordo com o *relative time* mostrado no *software*. A Tabela 6 ilustra os parâmetros de entradas adotados para o estudo de caso 3.

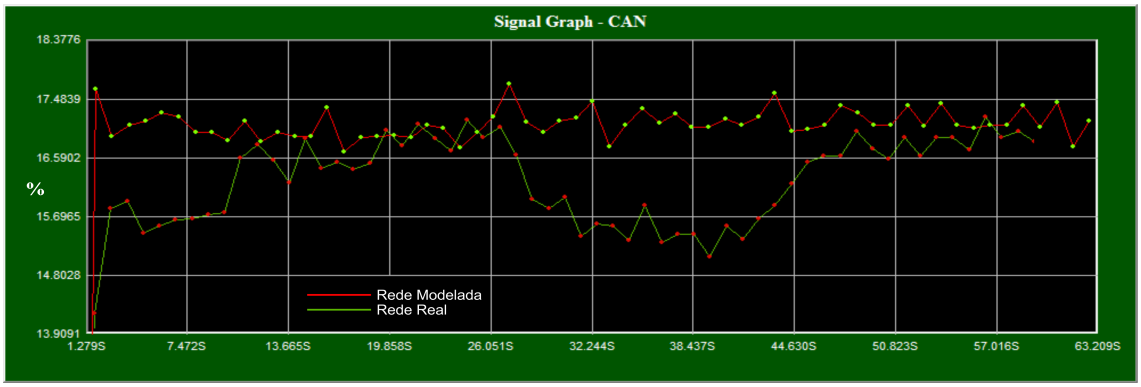
Tabela 6 – Parâmetros de configuração da rede CAN para o caso 3

Nome	**Descrição**
Formato do quadro de mensagem	Formato estendido (CAN2.0B)
Número de mensagens	28 Mensagens
Tipo de mensagem	Periódica (tempo de transmissão determinado)
Velocidade de transmissão	250 Kbps
Período ou tempo de amostragem	Definido pela norma J1939
Esquema de prioridade	CSMA-CD
Tamanho do campo de dados das mensagens	64 bits

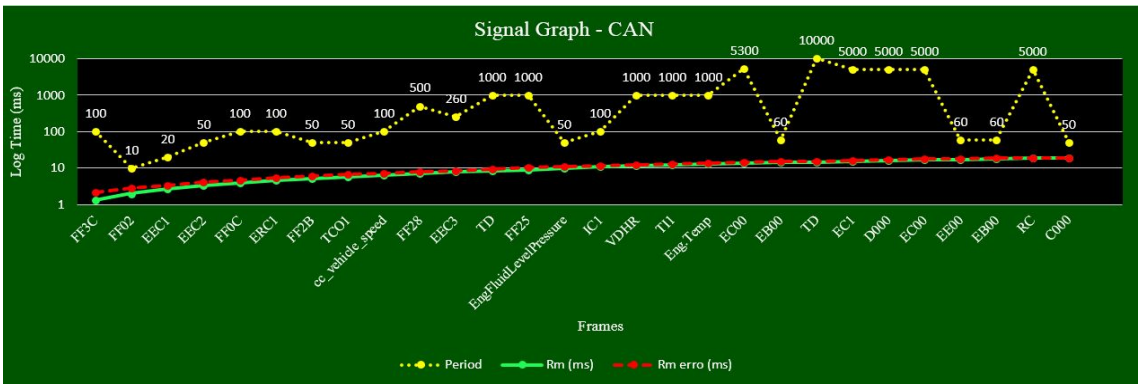
[1] Registro dos *frames* que transitaram no barramento
[2] Tempo relativo entre a transmissão de duas mensagens com o mesmo identificador

4.3.2 Análise dos resultados

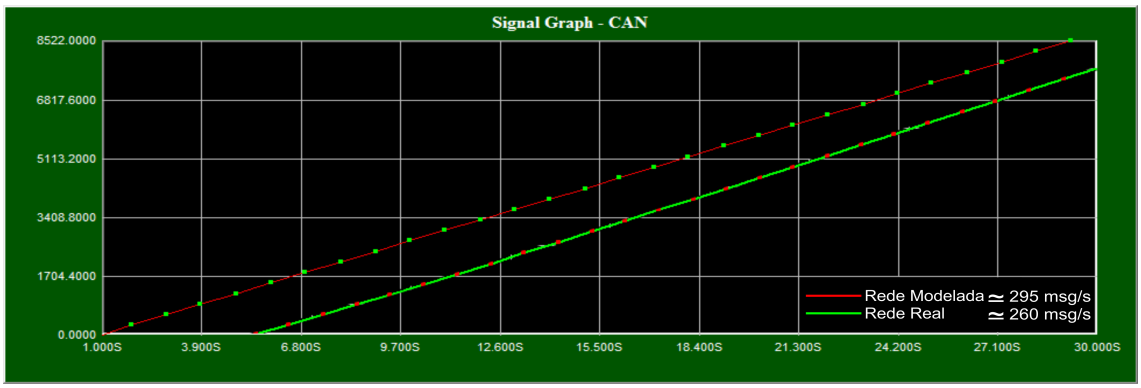
Três parâmetros de desempenho foram considerados para análise do desempenho das duas redes; a primeira referente ao *Log* do veículo real e a segunda referente à rede modelada no BusMaster. Os gráficos a seguir comparam os dois casos, através dos parâmetros de desempenho, a fim de verificar se a rede modelada representa de forma satisfatória a rede do veículo real.



(a) Taxa de utilização do barramento



(b) Tempo de transmissão das mensagens



(c) Quantidade de mensagens enviadas

Figura 24 – Desempenho da rede do caso 3

A Figura 24a mostra a taxa de utilização do barramento para dois casos: o primeiro representado pela curva em vermelho se refere à taxa de utilização da rede modelada e o

segundo representado pela curva em verde refere-se à taxa de utilização da rede do veículo real. Para as duas curvas é possível observar que os carregamentos se mantêm baixo ao longo de toda a simulação, indicando que as redes não estão sobrecarregadas e podem ser expandidas, se necessário, com a inserção de novas ECUs ou mensagens. O carregamento da rede modelada oscila entre 16% e 17% enquanto que a rede do veículo real apresenta um carregamento menor variando de 15 % a 16.3 % aproximadamente.

Percebe-se no gráfico mostrado na Figura 24a que apesar da rede modelada conter os mesmos *frames* da rede do veículo real, há diferenças entre as curvas, o que pode ser explicado pelo fato das mensagens da rede modelada não possuírem exatamente os mesmos períodos de transmissão das mensagens definidas na rede do veículo real. Diferenças estas que podem ser reduzidas com maior aproximação dos tempos de transmissão das mensagens do modelo com as mensagens da rede real.

Em relação ao tempo de transmissão das mensagens no barramento a Figura 24b mostra que todos os *frames* da rede cumpriram o requisito temporal, de forma que tanto para situações de transmissão normal quanto para situações de erro os tempos de transmissão estão abaixo do período definido.

A Figura 24c apresenta a quantidade de mensagens enviadas para os dois casos de forma que a curva em vermelho representa a modelagem realizada no BusMaster e a curva n representa o comportamento da rede do veículo real. A diferença entre as curvas pode ser explicada pelo mesmo fato citado na análise da taxa de utilização do barramento. E uma melhor aproximação seria também a busca por definir os períodos de transmissão cada vez mais próximo aos períodos das mensagens no veículo real.

5 Conclusão

Considerando o objetivo principal deste trabalho:

"Utilizar a modelagem e simulação para avaliar o desempenho de redes automotivas em ambiente virtual"

A utilização desta técnica mostrou-se como sendo eficaz para o desenvolvimento e melhoria da arquitetura eletroeletrônica automotiva, uma vez que auxilia na redução de custos e no tempo de desenvolvimento dedicado. Com modelagem e simulação é possível identificar erros e falhas em potenciais na rede projetada, promover mais facilmente a integração de ECUs reais e, sobretudo promover a análise de desempenho para entender o comportamento da rede.

O protocolo CAN tem se mostrado uma alternativa eficaz quando adotado como padrão de comunicação entre dispositivos de controle. Esta eficácia tem se comprovado com aumento significativo de sua implementação ao longo dos anos, sendo considerado o protocolo mais utilizado em aplicações automotivas. Verificou-se neste trabalho que, com o aumento do número de dispositivos e componentes eletroeletrônicos, definir os parâmetros de configuração que resultam em um bom desempenho da rede é um dos desafios dos projetistas.

Ademais, a hipótese de que a simulação de redes automotivas é fundamental para avaliar o comportamento de redes automotivas quando estas são expandidas através de ECUs e/ou mensagens foi demostrada, como nos estudos de caso 1 e 2 em que são abordadas duas redes automotivas do mesmo fabricante, porém com configurações diferentes. Os efeitos das diferentes configurações da rede, prioritariamente no número de mensagens enviadas, puderam ser percebidos através das análises de desempenho, de modo que o carregamento do barramento sofreu variação em média de 9% para 15% entre o primeiro e o segundo caso respectivamente.

A métrica de desempenho que avalia o tempo de transmissão das mensagens no barramento também mostrou-se fundamental para avaliação da rede, pois através dela pode-se perceber quais mensagens da rede possuem tempo de transmissão maior que seu período, fato que ocorreu no estudo de caso 2. E através desta métrica é possível destacar também as mensagens que possuem tempo de transmissão baixo a ponto de elevar o carregamento do barramento o que pode vir a comprometer o funcionamento do dede.

Além de reproduzir o comportamento real da rede, outra característica importante da modelagem simulação de redes automotivas é a capacidade representar um modelo

virtual baseado na rede de um veículo real. No terceiro estudo de caso modelou-se uma rede automotiva baseada na rede um veículo real, observou-se que o comportamento da rede modelada, quanto ao carregamento do barramento e taxa de transmissão das mensagens, se aproximou do comportamento da rede do veículo real. De posse desse modelo é possível simular a rede para diversas configurações seja através da inserção de ECUs, quantidade de mensagens, alteração das taxas de transmissão dentre outras possibilidades com uma redução significativa de custos e tempo aos projetistas. Além de prever comportamentos do sistema, antecipando acontecimentos que não seriam apropriados de ocorrerem após a implementação da rede.

Desta forma a principal contribuição dada com este trabalho foi elucidar a técnica de modelagem e simulação como um método eficiente no desenvolvimento e melhoria de redes automotivas, devido a sua capacidade de representar a topologia e funcionamento destes sistemas. Outra importante contribuição foi avaliar as métricas de desempenho que permitem avaliar o comportamento da rede e predizer erros potenciais que acarretariam em maiores despesas aos desenvolvedores. As limitações do estudo encontram-se principalmente na dificuldade de modelar com exatidão as redes dos veículos, visto que, para o caso 1 e 2 as ECUs existente nas redes foram determinadas baseando-se nos sistemas eletroeletrônicos presentes nos caminhões e desta forma foram utilizadas todas as mensagens possíveis de serem enviadas pelas ECUs que controlam esses sistemas, o que pode não corresponder a arquitetura real da rede dos veículos descritos. Para o caso 3 a inexatidão nas definições dos períodos impediu que o modelo se aproximasse ainda mais da rede real, quanto as métricas de desempenho.

Desta maneira, estudos que podem ser considerados para trabalhos futuros seria a obtenção de informações mais precisas quanto as redes dos veículos de forma a obter um modelo ainda mais fidedigno ao modelo real e então aplicar alguma técnica de validação, para validar o modelo desenvolvido. Outra contribuição a ser levada em consideração é a inserção de novas métricas de desempenho no software BusMaster tais como tempo de espera das mensagens no *buffer* e tempo de resposta da mensagem. O BusMaster carece ainda de materiais que proporcionem melhorias e disseminação do programa, visto que, por ser um *software* recente e de plataforma aberta esta ferramenta ainda carece de tutoriais, manuais, exemplos e aprimoramento.

Diante disso a Tabela 7 mostra as etapas que foram planejadas para serem executadas neste trabalho elucidando quais foram efetivamente realizadas.

Tabela 7 – Execução das etapas planejadas no trabalho.

Etapas Planejadas	Tarefas Executadas
Revisão Bibliográfica	Executada
Definição do Problema	Executada
Coleta de Dados	Executada
Modelagem e Simulação	Executada
Validação do Modelo	Executada Parcialmente

Referências

ALBERT, A. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded World*, v. 2004, p. 235–252, 2004. Citado na página 18.

BARBOSA, L. R. G. Rede can. *Escola de Engenharia da UFMG. Universidade Federal de Minas Gerais, Belo Horizonte*, 2003. Citado na página 22.

BARTISH, M. *Analysis and System Test of Powertrain Embedded Control Systems in Heavy Vehicles during Start-Up and Shutdown*. Dissertação (Mestrado) — Royal Institute of Technology, 2011. Citado 3 vezes nas páginas 45, 50 e 51.

BELL, J. Network protocols used in the automotive industry. *The University of Wales, Aberystwyth*, p. 07–24, 2002. Citado na página 19.

BLAKE, D.; LEADER, A. Embedded systems and vehicle innovation. *Automotive Engineering International*, SAE INTERNATIONAL, v. 113, n. 10, p. 42, 2005. Citado na página 14.

BOSCH. *Linha de Injeção e Ignição Eletrônica*. [S.l.], 2001. Citado na página 17.

BRUDNA, C. *Desenvolvimento de sistemas de automação industrial baseados em objetos distribuídos e no barramento CAN*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 09 2000. Citado 3 vezes nas páginas 9, 21 e 22.

DGE INC. Disponível em: <http://www.dgeinc.com/network_validation.htm>. Citado 2 vezes nas páginas 9 e 18.

FELLMETH, P.; LöFFLER, T. Networking heavy-duty vehicles based on sae j1939. *Vector*, p. 12, 09 2008. Citado 2 vezes nas páginas 9 e 28.

FRITZ KüBLER GMBH. *Absolute Singleturn Enconder with SAE J19139*. 78054 VS - Schwenningen / Germany, 2009. Citado 2 vezes nas páginas 9 e 29.

GODOY, E. Desenvolvimento de uma ferramenta de análise de desempenho de redes can para aplicações em sistemas agrícolas. *Desenvolvimento de uma Ferramenta de Análise de Desempenho de Redes CAN para Aplicações em Sistemas Agrícolas*, 2007. Citado 9 vezes nas páginas 9, 10, 22, 24, 30, 31, 32, 34 e 42.

GUMBRICH, S.; KOPPINGER, P. Embedded systems – here, there and everywhere. 2004. Citado na página 14.

HOFSTEE, J.; GOENSE, D. Simulation of a controller area network-based tractor—implement data bus according to iso 11783. *Journal of agricultural engineering research*, Elsevier, v. 73, n. 4, p. 383–394, 1999. Citado 2 vezes nas páginas 34 e 48.

HU, J. et al. Design and application of sae j1939 communication database in city-bus information integrated control system development. In: IEEE. *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*. [S.l.], 2007. p. 3429–3434. Citado na página 26.

JUNGER, M. Introduction to j1939. *Vector*, 2010. Citado 2 vezes nas páginas 27 e 28.

KLüSER, J. Reasons for system simulation with canoe j1939. *Vector*, 06 2004. Citado na página 15.

LEEN, G.; HEFFERNAN, D. Expanding automotive electronic systems. *Computer*, v. 35, n. 1, p. 88–93, Jan 2002. ISSN 0018-9162. Citado na página 17.

LIAN, F.-L.; MOYNE, W.; TILBURY, D. Network design consideration for distributed control systems. *Control Systems Technology, IEEE Transactions on*, IEEE, v. 10, n. 2, p. 297–307, 2002. Citado 2 vezes nas páginas 9 e 31.

LIU, H.; AN, J.-P.; YANG, J. Vehicle network communication protocol - a case of study. n.d. Citado 2 vezes nas páginas 18 e 19.

LOPES, W. C. *Análise de desempenho do protocolo CAN para aplicação na área agrícola utilizando redes de Petri coloridas*. Dissertação (Mestrado) — Universidade de São Paulo, 2007. Citado 2 vezes nas páginas 26 e 30.

MONOT, A. et al. Multicore scheduling in automotive ecus. In: *Embedded Real Time Software and Systems-ERTSS 2010*. [S.l.: s.n.], 2010. Citado na página 14.

NAVET, N. et al. Trends in automotive communication systems. *Proceedings of the IEEE*, v. 93, n. 6, p. 1204–1223, June 2005. ISSN 0018-9219. Citado 3 vezes nas páginas 15, 17 e 18.

NEWCASTLE UNIVERSITY. *Development of an open network communication protocol standard*. 2014. Disponível em: <www.impact.ref.ac.uk/CaseStudies/CaseStudy.aspx?Id=21783>. Citado na página 21.

PARET, D. *Multiplexed Networks for Embedded Systems: CAN, LIN, Flexray, Safe-by-Wire....* Chichester, UK: John Wiley & Sons, Ltd, 2007. Citado 3 vezes nas páginas 19, 23 e 24.

PUJATTI, F. J. et al. Design and tests of electronic management system for small motorcycle spark ignition engines. 2012. Citado na página 44.

PUNNEKKAT, S.; HANSSON, H.; NORSTROM, C. Response time analysis under errors for can. In: IEEE. *Real-Time Technology and Applications Symposium, 2000. RTAS 2000. Proceedings. Sixth IEEE*. [S.l.], 2000. p. 258–265. Citado na página 34.

SIMONEAU, P. The osi model: understanding the seven layers of computer networks. *Global Knowledge*, 2006. Citado na página 19.

SOUSA, R. V. de. *CAN (Controller Area Network): uma abordagem para automação e controle na área agrícola*. Dissertação (Mestrado) — Universidade de São Paulo, 2002. Citado 2 vezes nas páginas 19 e 21.

TINDELL, K.; BURNS, A. *Guaranteed Message Latencies For Distributed Safety-Critical Hard Real-Time Control Networks*. 1994. Citado 2 vezes nas páginas 33 e 34.

TINDELL, K.; BURNS, A.; WELLINGS, A. Calculating controller area network (can) message response times. *Control Engineering Practice*, v. 3, p. 1163–1169, 1995. Citado na página 32.

VECTOR. *CANoe Test Feature Set Tutorial*. [S.l.], 2009. Citado na página 36.

VECTOR. *Product Information CAnoe*. [S.l.], 2014. Citado 3 vezes nas páginas 9, 36 e 37.

ÖZCAN, M.; GüNAY, H. Control of diesel engines mounted on vehicles in mobile cranes via can bus. *Turkish Journal of Electrical Engineering & Computer Sciences*, 2012. Citado na página 28.

ZHOU, F.; LI, S.; HOU, X. Development method of simulation and test system for vehicle body can bus based on canoe. In: *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*. [S.l.: s.n.], 2008. p. 7515–7519. Citado na página 36.

ANEXO A – Tutorial BusMaster

O primeiro passo na modelagem de uma rede automotiva no *software* BusMaster é o desenvolvimento do *database*, que consiste na configuração dos *frames* que serão utilizados na rede, definindo assim as mensagens, configurações e seus respectivos sinais. Neste explicativo é tomando como exemplo o *frame* **EEC1** o qual, baseado no documento J1939-71, é responsável pelo controle eletrônico do motor. Na tabela 8 são explicitados todos os sinais que pertencentes ao *frame* **EEC1**.

Tabela 8 - Sinais do *frame* **EEC1**.

Byte	Bit	Length	Explanation	State	Resolution	Limits	Note
1	1	4	**Engine and retarder torque mode**				A
			Low idle governor	0000			
			Accelerator pedal	0001			
			Cruise control	0010			
			PTO governor	0011			
			Road speed governing	0100			
			ASR control	0101			
			Transmission control	0110			
			Not used	0111			
			Torque limiting	1000			
			High speed governor	1001			
			Not used	1010			
			Not used	1011			
			Not defined	1100			
			Not used	1101			
			Other	1110			
			Not available	1111			
	5	4	**Actual engine - percent torque high resolution**		0.125	0 to 0.875%	B
			Not available	1000			
			Not available	1001			
			Not available	1010			
			Not available	1011			
			Not available	1100			
			Not available	1101			
			Not available	1110			
			Not available	1111			
2	1	8	**Drivers demand engine - percent torque**		1%	-125% to +125%	
			Error indicator	FEh			
			Not available	FFh			
3	1	8	**Actual engine - percent torque**		1%	-125% to +125%	
			Error indicator	FEh			

Byte	Bit	Length	Explanation	State	Resolution	Limits	Note
			Not available	FFh			
4	1	16	**Engine speed**		0.125 rpm	0 to 8 031.875 rpm	
			Error indicator	FExxh			
			Not available	FFxxh			
6	1	8	**Source address of controlling device for engine control**		1	0 to 250	
			Error	FE			
			Take no action	FF			
7	1	4	**Engine starter mode**		1	0 to 15	
			Start not rqed	0000			
			Starter active gear not engaged	0001			
			Starter active gear engaged	0010			
			Strt fnshd strtr nt actv aftr hvng bn a	0011			
			Strtr inhbtd d to eng already running	0100			
			Strtr inhbtd d to eng nt ready for start	0101			
			Strtr inhbtd d to drv in enggd othr trns	0110			
			Strtr inhbtd d to active immobilizer	0111			
			Strtr inhbtd due to starter overtemp	1000			
			1011 reserved	1001			
			Starter inhibited reason unknown	1100			
			Error	1110			
			Not available	1111			
	5..8		Not defined				
8	1	8	**Engine demand - percent torque**		1	-125 to 125%	
			Error	FE			
			Not available	FF			

Através da tabela 8 observa-se que este *frame* contém 8 sinais diferentes, ou seja, 8 tipos de informações podem ser transmitidas nesta mensagem.

Para o desenvolvimento do *database* o usuário seleciona o protocolo em que se deseja trabalhar e inicia um novo *database*. Como mostra a Figura 25.

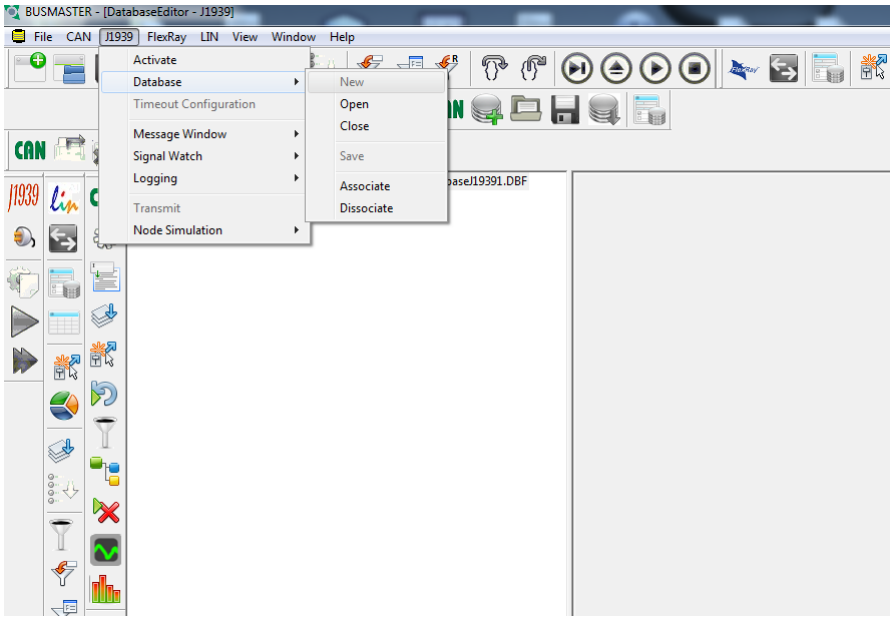


Figura 25 – Criação do *database*

A próxima etapa é a inserção de novos *frames* e a definição dos sinais que os compõem. A Figura 26 mostra a tela de configuração das mensagens, em que no quadro a esquerda são mostrados os *frames* já incluídos no *database* enquanto que no quadro a direita são mostradas as configurações e o sinais pertencentes à estes *frames*.

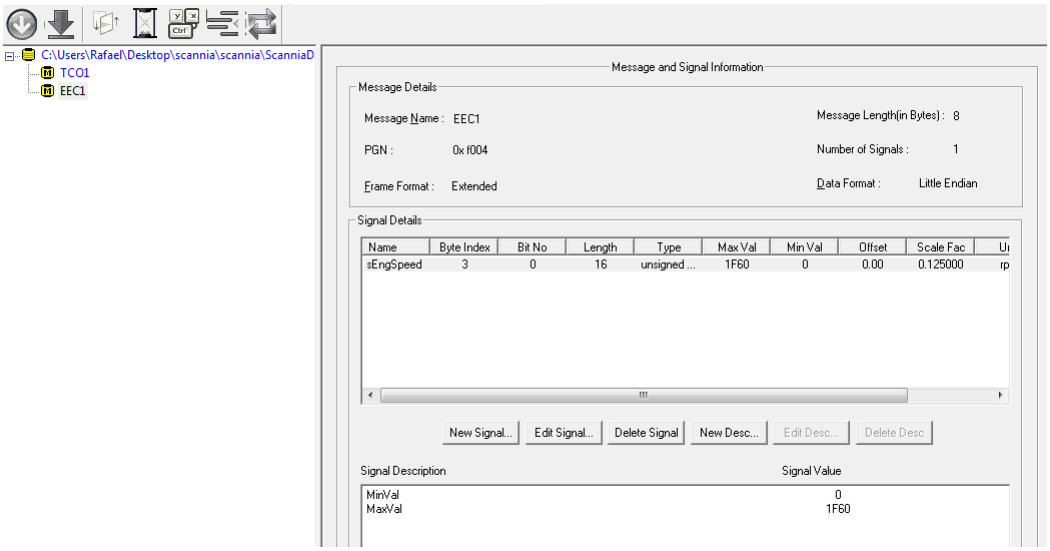


Figura 26 – Janela de gerenciamento das mensagens

Para inserção de novos sinais é necessário o conhecimento de alguns parâmetros de configuração, tais como: o tipo de sinal (booleano, inteiro, unsigned int), o *byte* e o respectivo *bit* em que este sinal se encontra dentro do campo de dados, o comprimento do sinal em *bits*, o fator de escala para conversão em unidades reais, os limites máximos e mínimos e por fim a unidade de medida do parâmetro em questão. A Figura 27 mostra a janela de configuração para definição de um novo sinal ao *frame*.

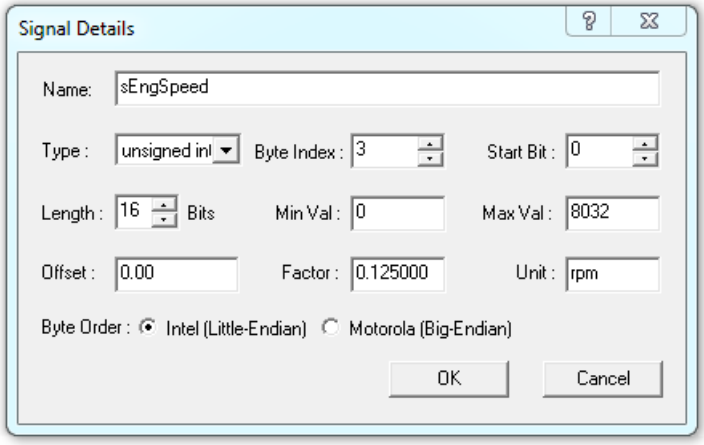


Figura 27 – Janela de gerenciamento das mensagens

Neste exemplo, foi inserido ao *frame* **EEC1** o sinal referente a velocidade do motor. Em é definido como um sinal do tipo *unsigned int*, ou seja, um valor inteiro positivo. Este

sinal ocupa o terceiro e quarto *byte* do campo de dados e possui cumprimento de 16 *bits*. Este sinal pode variar de 0 a 8032 rpm e apresenta uma resolução de 0.125 rpm/bit, isto é, o valor que o terceiro e quarto *byte* apresentam em escala decimal ao ser multiplicado por esta resolução indica a rotação real do motor do veículo.

Sistemas simulados

Os sistemas simulados possuem a função de definir as ECUs que farão parte da rede automotiva a ser modelada e também atribuir os algorítmos à estas unidades.

A Figura 28 mostra a tela principal de configuração dos sistemas simulados, em que no quadro a esquerda são definidas as ECUs que fazem parte da rede, enquanto que no quadro a direita são mostradas os parâmetros de cada ECU, explicitando os *handlers* (manipuladores) presentes na rotina de funcionamento das unidades.

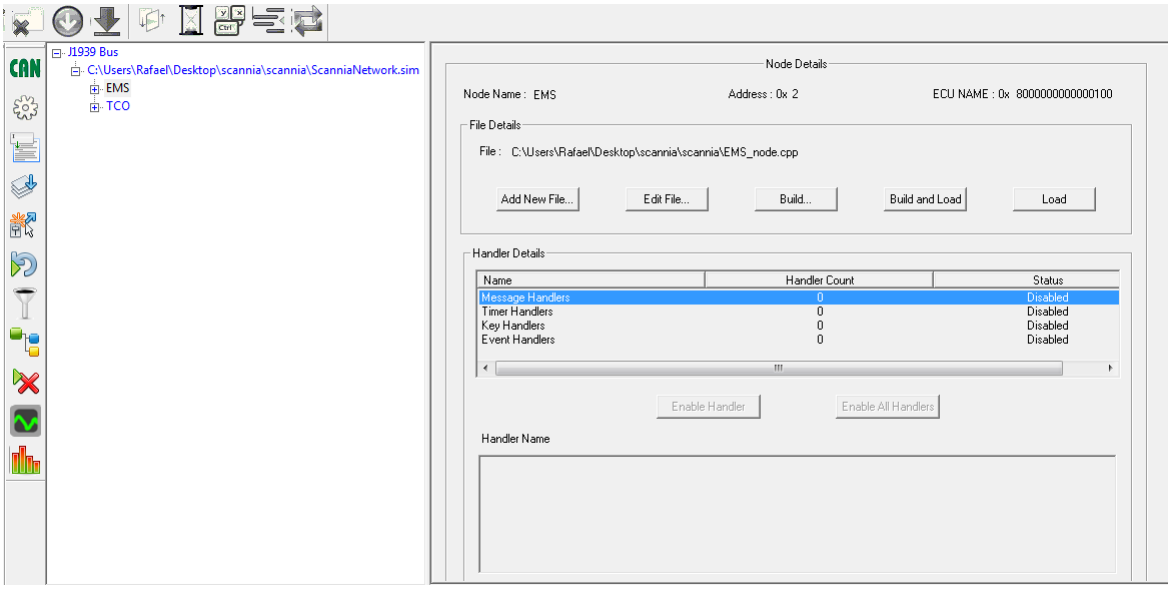


Figura 28 – Tela de configuração dos sistemas simulados

Para iniciar um novo sistema simulado o usuário deve clicar com o botão direito sobre a raiz no quadro a esquerda, como mostra a Figura 29.

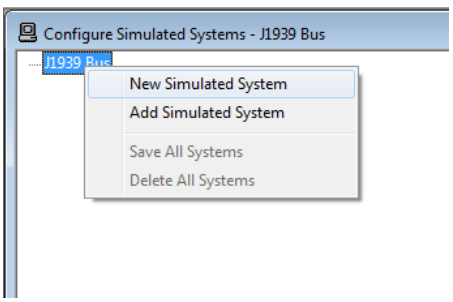


Figura 29 – Iniciando um sistema simulado

A próxima etapa é a inserção das ECUs no sistema simulado criado. Para isto o usuário clica com o botão direito sobre o sistema simulado e pressiona o botão **Add Node** conforme a Figura 30.

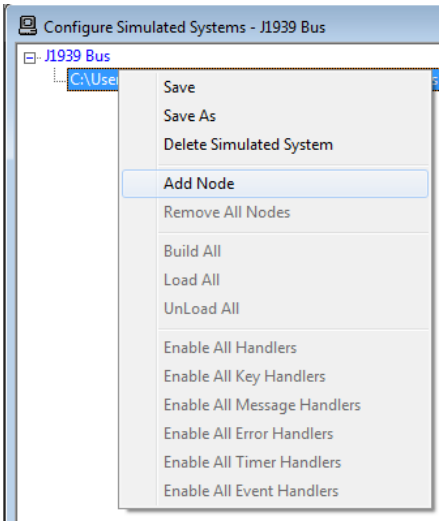


Figura 30 – Inserção de novas ECUs ao sistema simulado.

Surgirá então uma tela (Figura 31) em que será definido o nome e o endereço da ECU.

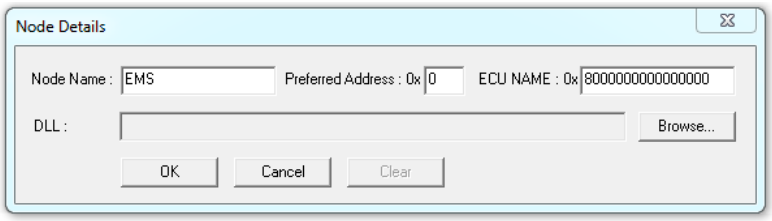


Figura 31 – Definição do nome e endereço da ECU

Uma vez que a ECU foi criada e o algorítimo da unidade desenvolvido é necessário que este seja adicionado ao sistema através do botão **Add New File** e em seguida convertido e carregado como uma DLL (Dynamic Link Library) para o BusMaster através do botão **Build and Load** como mostra a Figura 32.

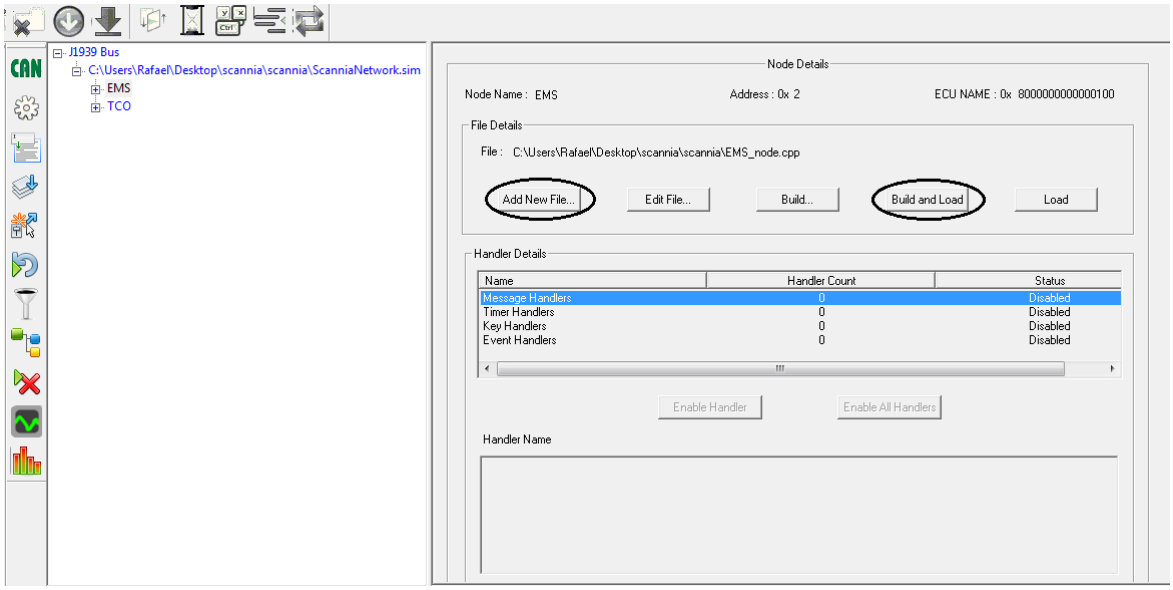


Figura 32 – Código carregado no Bus Master

Assim que a DLL for carregada todos os *handlers* estarão disponíveis para serem simulados, sendo que estes podem ser ativados e desativados total ou parcialmente através dos botões **Enable Handler** e **Disable Handler** respectivamente.

Simulação e estatísticas da rede

Para dar início a simulação o usuário deve clicar no botão verde no canto superior esquerdo da tela como mostra a Figura 33 , nesta etapa é mostrado a tela da simulação contendo todos os *frames* utilizados na programação, juntamente com diversas informações das mensagens tais como ID, endereço de origem e destino, prioridade e magnitude dos sinais no campo de dados.

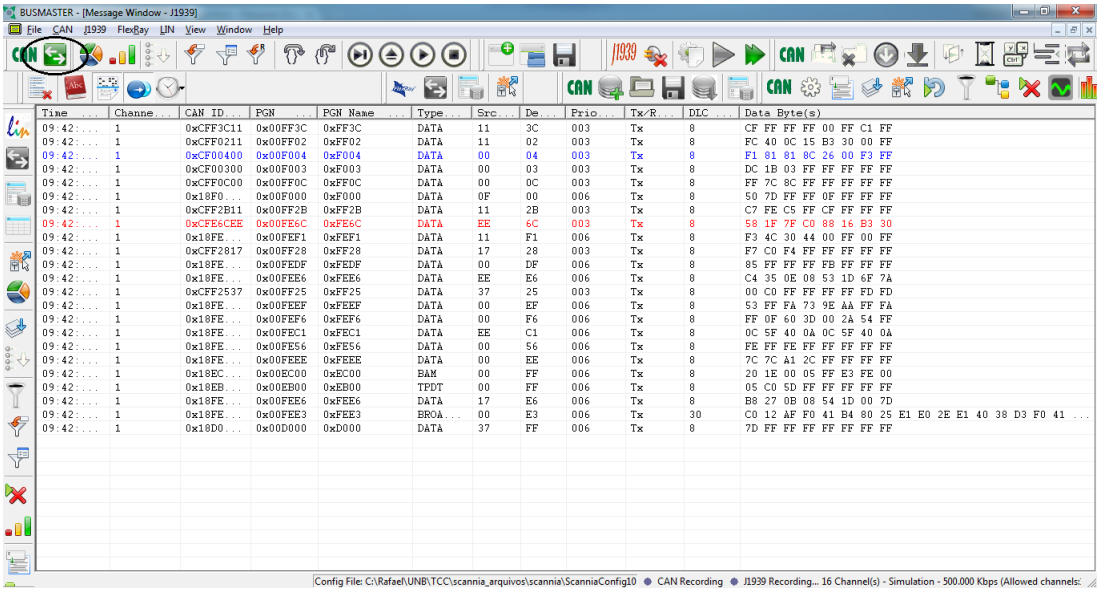


Figura 33 – Tela de simulação no BusMaster

O **software** apresenta o recurso para visualização das estatísticas da rede, informações estas como carregamento do barramento, número de mensagens enviadas e recebidas, quantidade de erros ocorridos dentre outras informações relevantes para análise do desempenho da rede. Para acessar estas informações o usuário clica no botão do "gráfico de pizza"localizado na parte superior da tela, como mostra Figura 34.

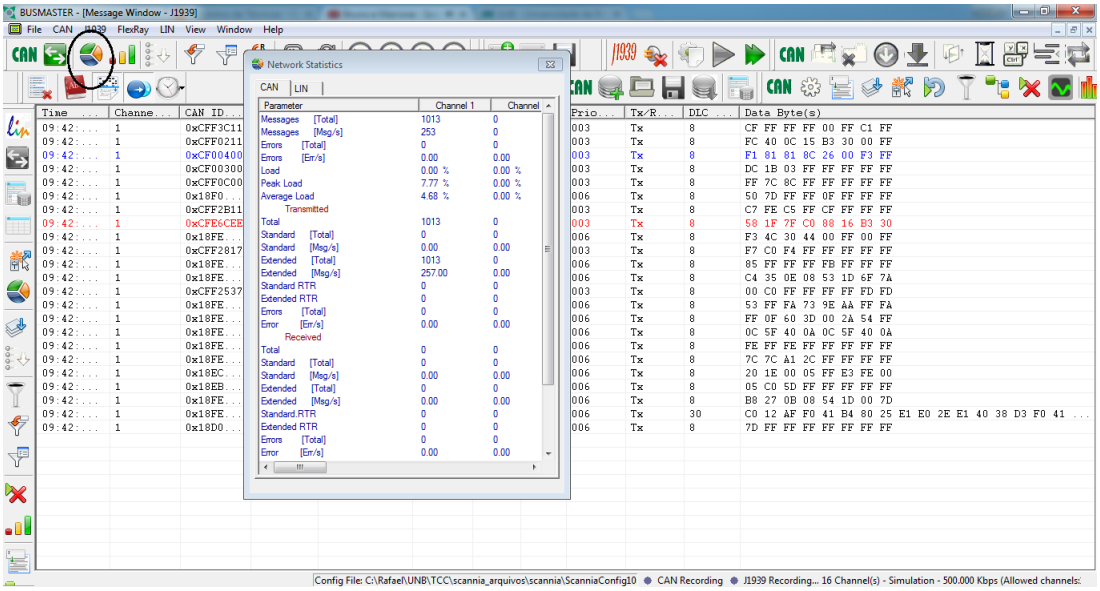


Figura 34 – Tela de simulação no BusMaster

ANEXO B – Programação das ECUs do caso 1

```
#ALGORÍTIMO ECU - COO
/* Start J1939 include header */
#include <Windows.h>
#include <J1939Includes.h>
#include "database_Unions.h"
/* End J1939 include header */

//Global Variables
    EngineFluidLevel_Pressure msgEngineFluidLevel_Pressure=
    { {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xFEEF27}, 8 };
        UINT32 EngineOilLevel=0;
        UINT32 EngineOilPressure =0;
        UINT32 FuelDeliveryPressure =0;
        UINT32 ExtCrankBlowPressure =0;
        UINT32 CrankcasePressure =0;
        UINT32 CoolTemp =0;
        UINT32 CoolantLevel=0;

/* Start J1939 Function Prototype  */
GCC_EXTERN void GCC_EXPORT OnTimer_timer500_500( );
GCC_EXTERN void GCC_EXPORT OnDLL_Load();
GCC_EXTERN void GCC_EXPORT OnDLL_Unload();
/* End J1939 Function Prototype  */

/* Start J1939 generated function - OnTimer_timer500_500 */
void OnTimer_timer500_500( )
{
SendMsg ((STJ1939_MSG *)&msgEngineFluidLevel_Pressure);
}/* End J1939 generated function - OnTimer_timer500_500 */

/* Start J1939 generated function - OnDLL_Load */
void OnDLL_Load()
{
/* TODO */
Trace("COO - Basic node configuration");

//Initialising frame data
    msgEngineFluidLevel_Pressure.m_sWhichBit=
        (EngineFluidLevel_Pressure_*)malloc(sizeof(EngineFluidLevel_Pressure_));
//Updating data
msgEngineFluidLevel_Pressure.m_sWhichBit-> mEn  gineOilLevel = EngineOilLevel;
msgEngineFluidLevel_Pressure.m_sWhichBit-> mEngineOilPressure = EngineOilPressure;
msgEngineFluidLevel_Pressure.m_sWhichBit-> mFuelDeliveryPressure = FuelDeliveryPressure;
msgEngineFluidLevel_Pressure.m_sWhichBit-> mExtCrankBlowPressure = ExtCrankBlowPressure;
msgEngineFluidLevel_Pressure.m_sWhichBit-> mCrankcasePressure = CrankcasePressure;
msgEngineFluidLevel_Pressure.m_sWhichBit-> mCoolTemp= CoolTemp;
msgEngineFluidLevel_Pressure.m_sWhichBit-> mCoolantLevel = CoolantLevel;

}/* End J1939 generated function - OnDLL_Load */

/* Start J1939 generated function - OnDLL_Unload */
void OnDLL_Unload()
{
free(msgEngineFluidLevel_Pressure.m_sWhichBit);
}/* End J1939 generated function - OnDLL_Unload */

```

```
#ALGORÍTIMO ECU - EMS
/* Start J1939 include header */
#include <Windows.h>
#include <J1939Includes.h>
#include "ScanniaDatabase_Unions.h"
/* End J1939 include header */
/* Start J1939 global variable */
//Global Variables
EEC1 msgEEC1 ={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xF00400}, 8 };
UINT32 rpmVal=0;
UINT32 EngineRetarderTorqueMode=1;
UINT32 DriversDemandEngine=0;
UINT32 EngineStarterMode=2;
EEC2 msgEEC2 ={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xF00300}, 8 };
UINT32 aceleratorPosition=0;
UINT32 rSpeedLimit=0;
FuelEconomy msgFuelEconomy ={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0x18FEF200}, 8 };
UINT32 average=0;
UINT32 tPosition=0;
EngTemp msgEngTemp ={ {18, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xFEEE00}, 8 };
UINT32 engCoolTemp=0;
UINT32 fuelTemp=0;
UINT32 engOilTemp=0;
UINT32 turboOilTemp=0;
UINT32 engInterTemp=0;
EngineHoursRevolutions msgEngineHoursRevolutions ={ {18, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xFEE500}, 8 };
UINT32 totalEngineHours=0;
FuelConsumption msgFuelConsumption ={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xFEE900}, 8 };
UINT32 totalFuelUsed=0;
CC_VehicleSpeed msgCC_VehicleSpeed ={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0xFEF100}, 8 };
UINT32 VehicleSpeed=0;
UINT32 BrakeSwitch=0;
UINT32 TwoSpeedAxleSwitch=0;
UINT32 ParkingBrakeSwitch=0;
UINT32 ClutchSwitch=0;
DLN2_Proprietary msgDLN2_Proprietary={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0XFF8100}, 8 };
UINT32 LowEngineOilPressure=0;
UINT32 HighEngineCoolTemp=0;
UINT32 Charge61=0;
UINT32 EngineControlMode=0;

/* End J1939 global variable */


/* Start J1939 Function Prototype  */
GCC_EXTERN void GCC_EXPORT OnTimer_EEC1_1000( );
GCC_EXTERN void GCC_EXPORT OnTimer_EEC2_50( );
GCC_EXTERN void GCC_EXPORT OnDLL_Load();
GCC_EXTERN void GCC_EXPORT OnKey_g(unsigned char KeyValue);
GCC_EXTERN void GCC_EXPORT OnDLL_Unload();
GCC_EXTERN void GCC_EXPORT OnKey_h(unsigned char KeyValue);
GCC_EXTERN void GCC_EXPORT OnTimer_time100_100( );
GCC_EXTERN void GCC_EXPORT OnKey_a(unsigned char KeyValue);
GCC_EXTERN void GCC_EXPORT OnKey_d(unsigned char KeyValue);
/* End J1939 Function Prototype  */

/* Start J1939 generated function - OnTimer_EEC1_1000 */
void OnTimer_EEC1_1000( )
{
/* TODO */

//Enviando a mensagem
SendMsg ((STJ1939_MSG *)&msgEEC1);
SendMsg ((STJ1939_MSG *)&msgEngTemp);
SendMsg ((STJ1939_MSG *)&msgEngineHoursRevolutions);
SendMsg ((STJ1939_MSG *)&msgFuelConsumption);
//Trace("EMS - Message transmitted successfully");
}/* End J1939 generated function - OnTimer_EEC1_1000 */
/* Start J1939 generated function - OnTimer_EEC2_50 */
void OnTimer_EEC2_50( )
{
```

```
 72: /* TODO */
 73:
 74: //Enviando a mensagem
 75: SendMsg ((STJ1939_MSG *)&msgEEC2);
 76: //Trace("EMS - Message transmitted successfully");
 77: }/* End J1939 generated function - OnTimer_EEC2_50 */
 78: /* Start J1939 generated function - OnKey_h */
 79: void OnKey_h(unsigned char KeyValue)
 80: {
 81: /* TODO */
 82: Trace("EMS - Pressed H key");
 83: rpmVal=rpmVal-100;
 84: aceleratorPosition=aceleratorPosition-10;
 85:
 86: //Updating data
 87: msgEEC1.m_sWhichBit->sEngSpeed=rpmVal;
 88: msgEEC2.m_sWhichBit->acceleratorPedalPosition=aceleratorPosition;
 89:
 90: }/* End J1939 generated function - OnKey_h */
 91: /* Start J1939 generated function - OnTimer_time100_100 */
 92: void OnTimer_time100_100( )
 93: {
 94: /* TODO */
 95:
 96: //Enviando a mensagem
 97: SendMsg ((STJ1939_MSG *)&msgFuelEconomy);
 98: SendMsg ((STJ1939_MSG *)&msgCC_VehicleSpeed);
 99: SendMsg ((STJ1939_MSG *)&msgDLN2_Proprietary);
100:
101: //Trace("EMS - Message transmitted successfully");
102: }/* End J1939 generated function - OnTimer_time100_100 */
103: /* Start J1939 generated function - OnKey_a */
104: void OnKey_a(unsigned char KeyValue)
105: {
106: /* TODO */
107: Trace("EMS - Pressed A key");
108:
109: if(rSpeedLimit==0){
110: rSpeedLimit==1;
111: msgEEC2.m_sWhichBit->mRoadSpeedLimit=rSpeedLimit;
112: }
113: }/* End J1939 generated function - OnKey_a */
114: /* Start J1939 generated function - OnKey_d */
115: void OnKey_d(unsigned char KeyValue)
116: {
117: /* TODO */
118: Trace("EMS - Pressed D key");
119:
120: if(rSpeedLimit!=0){
121: rSpeedLimit==0;
122: msgEEC2.m_sWhichBit->mRoadSpeedLimit=rSpeedLimit;
123: }
124: }/* End J1939 generated function - OnKey_d */
125: /* Start J1939 generated function - OnDLL_Load */
126: void OnDLL_Load()
127: {
128: /* TODO */
129: Trace("EMS - Basic node configuration");
130:
131: //Initialising frame data
132: msgEEC1.m_sWhichBit = (EEC1_*)malloc(sizeof(EEC1_));
133: msgEEC2.m_sWhichBit = (EEC2_*)malloc(sizeof(EEC2_));
134: msgFuelEconomy.m_sWhichBit = (FuelEconomy_*)malloc(sizeof(FuelEconomy_));
135: msgEngTemp.m_sWhichBit = (EngTemp_*)malloc(sizeof(EngTemp_));
136: msgEngineHoursRevolutions.m_sWhichBit = (EngineHoursRevolutions_*)malloc(sizeof(EngineHoursRevolutions_));
137: msgFuelConsumption.m_sWhichBit = (FuelConsumption_*)malloc(sizeof(FuelConsumption_));
138: msgCC_VehicleSpeed.m_sWhichBit = (CC_VehicleSpeed_*)malloc(sizeof(CC_VehicleSpeed_));
139: msgDLN2_Proprietary.m_sWhichBit = (DLN2_Proprietary_*)malloc(sizeof(DLN2_Proprietary_));
140:
141: //Updating data
142: msgEEC1.m_sWhichBit->sEngSpeed=rpmVal;
```

```
msgEEC1.m_sWhichBit->mEngineRetarderTorqueMode=EngineRetarderTorqueMode;
msgEEC1.m_sWhichBit->mDriversDemandEngine=DriversDemandEngine;
msgEEC1.m_sWhichBit->mEngineStarterMode=EngineStarterMode;
msgEEC2.m_sWhichBit->acceleratorPedalPosition=aceleratorPosition;
msgEEC2.m_sWhichBit->mRoadSpeedLimit=rSpeedLimit;
msgFuelEconomy.m_sWhichBit->averageFuelEconomy=average;
msgFuelEconomy.m_sWhichBit->throttlePosition=tPosition;
msgEngTemp.m_sWhichBit->mEngCoolTemp=engCoolTemp;
msgEngTemp.m_sWhichBit->mFuelTemp=fuelTemp;
msgEngTemp.m_sWhichBit->mEngOilTemp=engOilTemp;
msgEngTemp.m_sWhichBit->mTurboOilTemp=turboOilTemp;
msgEngTemp.m_sWhichBit->mEngInterTemp=engInterTemp;
msgEngineHoursRevolutions.m_sWhichBit->mtotalEngineHours=totalEngineHours;
msgFuelConsumption.m_sWhichBit->mtotalFuelUsed=totalFuelUsed;
msgCC_VehicleSpeed.m_sWhichBit->mVehicleSpeed=VehicleSpeed;
msgCC_VehicleSpeed.m_sWhichBit->mBrakeSwitch=BrakeSwitch;
msgCC_VehicleSpeed.m_sWhichBit->mTwoSpeedAxleSwitch=TwoSpeedAxleSwitch;
msgCC_VehicleSpeed.m_sWhichBit->mParkingBrakeSwitch=ParkingBrakeSwitch;
msgCC_VehicleSpeed.m_sWhichBit->mClutchSwitch=ClutchSwitch;
msgDLN2_Proprietary.m_sWhichBit-> mLowEngineOilPressure= LowEngineOilPressure;
msgDLN2_Proprietary.m_sWhichBit-> mHighEngineCoolTemp= HighEngineCoolTemp;
msgDLN2_Proprietary.m_sWhichBit-> mCharge61= Charge61;
msgDLN2_Proprietary.m_sWhichBit-> mEngineControlMode= EngineControlMode;


if(EnableKeyHandlers(TRUE))
{
    Trace("EMS - Key handler enabled successfully");
}
if(EnableKeyHandlers(TRUE))
{
    Trace("EMS - Key handler enabled successfully");
}

if(EnableAllHandlers(TRUE))
{
    Trace("EMS - All handler enabled successfully");
}
}/* End J1939 generated function - OnDLL_Load */
/* Start J1939 generated function - OnKey_g */
void OnKey_g(unsigned char KeyValue)
{
/* TODO */
Trace("EMS - Pressed G key");
rpmVal=rpmVal+100;
aceleratorPosition=aceleratorPosition+10;

if(aceleratorPosition>8031){

aceleratorPosition=254;
}


//Updating data
msgEEC1.m_sWhichBit->sEngSpeed=(rpmVal/0.125);
msgEEC2.m_sWhichBit->acceleratorPedalPosition=aceleratorPosition;
}/* End J1939 generated function - OnKey_g */
/* Start J1939 generated function - OnDLL_Unload */
void OnDLL_Unload()
{
/* TODO */
free(msgEEC1.m_sWhichBit);
free(msgEEC2.m_sWhichBit);
free(msgFuelEconomy.m_sWhichBit);
free(msgEngTemp.m_sWhichBit);
free(msgEngineHoursRevolutions.m_sWhichBit);
free(msgFuelConsumption.m_sWhichBit);
free(msgCC_VehicleSpeed.m_sWhichBit);
free(msgDLN2_Proprietary.m_sWhichBit);
}/* End J1939 generated function - OnDLL_Unload */

```

```
#ALGORÍTIMO ECU - BMS
/* Start J1939 include header */
#include <Windows.h>
#include <J1939Includes.h>
#include "ScanniaDatabase_Unions.h"
/* End J1939 include header */
/* Start J1939 global variable */
//Global Variables
EBC1 msgEBC1 ={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0x18F0010B}, 8 };
UINT32 AbsActive =0;
UINT32 BrakePedalPosition =0;
UINT32 EbsBrakeSwitch =0;
UINT32 AbsOffroadSwitch =0;
UINT32 AsrEngineControlActive =0;
UINT32 AsrBrakeControlActive =0;
UINT32 TractionControlOverrideSwitch=0;
UINT32 AcceleratorInterlockSwitch=0;
UINT32 EngineDerateSwitch=0;
UINT32 AuxEngShutdownSwitch=0;
UINT32 RemoteAcceEnableSwitch=0;
UINT32 EngineRetarderSelection=0;
UINT32 AbsFullyOperational=0;
UINT32 AbsEbsAmberWarningState =0;
UINT32 AbsRedWarningState =0;
UINT32 AtcAsrLampState =0;
UINT32 SourceBrakeControl =0;
UINT32 HaltBrakeSwitch=0;
UINT32 TrailerAbsStatus=0;
UINT32 Tractor=0;
EBC4 msgEBC4={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0x1CFEAC0B}, 8 };
UINT32 FrontAxleLeft=0;
UINT32 FrontAxleRight=0;
UINT32 RearAxle1Left=0;
UINT32 RearAxle1Right =0;
UINT32 RearAxle2Left =0;
UINT32 RearAxle2Right =0;
UINT32 RearAxle3Left =0;
UINT32 RearAxle3Right =0;
/* End J1939 global variable */

/* Start J1939 Function Prototype  */
GCC_EXTERN void GCC_EXPORT OnTimer_EBC1_100( );
GCC_EXTERN void GCC_EXPORT OnDLL_Load();
GCC_EXTERN void GCC_EXPORT OnDLL_Unload();
GCC_EXTERN void GCC_EXPORT OnTimer_time5_5( );
/* End J1939 Function Prototype  */

/* Start J1939 generated function - OnTimer_BMS_1000 */
void OnTimer_EBC1_100( )
{
/* TODO */

//Sending message
SendMsg ((STJ1939_MSG *)&msgEBC1);
//Trace("BMS - Message transmitted successfully");
}/* End J1939 generated function - OnTimer_BMS_1000 */
/* Start J1939 generated function - OnTimer_time5_5 */
void OnTimer_time5_5( )
{
/* TODO */
SendMsg ((STJ1939_MSG *)&msgEBC4);
}/* End J1939 generated function - OnTimer_time5_5 */
/* Start J1939 generated function - OnDLL_Load */
void OnDLL_Load()
{
/* TODO */
Trace("BMS - Basic node configuration");

//Initialising frame data
msgEBC1.m_sWhichBit = (EBC1_*)malloc(sizeof(EBC1_));
msgEBC4.m_sWhichBit = (EBC4_*)malloc(sizeof(EBC4_));
```

```
 72:
 73:
 74: //Updating data
 75:
 76: msgEBC1.m_sWhichBit-> mAbsActive = AbsActive;
 77: msgEBC1.m_sWhichBit-> mBrakePedalPosition = BrakePedalPosition;
 78: msgEBC1.m_sWhichBit-> mEbsBrakeSwitch = EbsBrakeSwitch;
 79: msgEBC1.m_sWhichBit-> mAbsOffroadSwitch = AbsOffroadSwitch;
 80: msgEBC1.m_sWhichBit-> mAsrEngineControlActive = AsrEngineControlActive;
 81: msgEBC1.m_sWhichBit-> mAsrBrakeControlActive = AsrBrakeControlActive;
 82: msgEBC1.m_sWhichBit-> mTractionControlOverrideSwitch= TractionControlOverrideSwitch;
 83: msgEBC1.m_sWhichBit-> mAcceleratorInterlockSwitch= AcceleratorInterlockSwitch;
 84: msgEBC1.m_sWhichBit-> mEngineDerateSwitch= EngineDerateSwitch;
 85: msgEBC1.m_sWhichBit-> mAuxEngShutdownSwitch= AuxEngShutdownSwitch;
 86: msgEBC1.m_sWhichBit-> mRemoteAcceEnableSwitch= RemoteAcceEnableSwitch;
 87: msgEBC1.m_sWhichBit-> mEngineRetarderSelection= EngineRetarderSelection;
 88: msgEBC1.m_sWhichBit-> mAbsFullyOperational= AbsFullyOperational;
 89: msgEBC1.m_sWhichBit-> mAbsEbsAmberWarningState= AbsEbsAmberWarningState;
 90: msgEBC1.m_sWhichBit-> mAbsRedWarningState = AbsRedWarningState;
 91: msgEBC1.m_sWhichBit-> mAtcAsrLampState = AtcAsrLampState;
 92: msgEBC1.m_sWhichBit-> mSourceBrakeControl = SourceBrakeControl;
 93: msgEBC1.m_sWhichBit-> mHaltBrakeSwitch= HaltBrakeSwitch;
 94: msgEBC1.m_sWhichBit-> mTrailerAbsStatus= TrailerAbsStatus;
 95: msgEBC4.m_sWhichBit-> mFrontAxleLeft= FrontAxleLeft;
 96: msgEBC4.m_sWhichBit-> mFrontAxleRight = FrontAxleRight;
 97: msgEBC4.m_sWhichBit-> mRearAxle1Left= RearAxle1Left;
 98: msgEBC4.m_sWhichBit-> mRearAxle1Right = RearAxle1Right;
 99: msgEBC4.m_sWhichBit-> mRearAxle2Left = RearAxle2Left;
100: msgEBC4.m_sWhichBit-> mRearAxle2Right = RearAxle2Right;
101: msgEBC4.m_sWhichBit-> mRearAxle3Left = RearAxle3Left;
102: msgEBC4.m_sWhichBit-> mRearAxle3Right = RearAxle3Right;
103:
104: }/* End J1939 generated function - OnDLL_Load */
105: /* Start J1939 generated function - OnDLL_Unload */
106: void OnDLL_Unload()
107: {
108: /* TODO */
109: free(msgEBC1.m_sWhichBit);
110: free(msgEBC4.m_sWhichBit);
111: }/* End J1939 generated function - OnDLL_Unload */
112:
113:
```

```
#ALGORÍTIMO ECU - SMS
/* Start J1939 include header */
#include <Windows.h>
#include <J1939Includes.h>

#include "ScanniaDatabase_Unions.h"
/* End J1939 include header */
/* Start J1939 global variable */
//Global Variables

VehicleWeight msgVehicleWeight={ {0, 1, MSG_TYPE_BROADCAST, DIR_TX, 0XFEEA2F}, 8 };
UINT32 AxleLocation=0;
UINT32 AxleWeight =0;
UINT32 TrailerWeight =0;
UINT32 CargoWeight =0;

/* End J1939 global variable */


/* Start J1939 Function Prototype  */
GCC_EXTERN void GCC_EXPORT OnDLL_Load();
GCC_EXTERN void GCC_EXPORT OnDLL_Unload();
GCC_EXTERN void GCC_EXPORT OnTimer_time100_100( );
/* End J1939 Function Prototype  */

/* Start J1939 generated function - OnTimer_time100_100 */
void OnTimer_time100_100( )
{
/* TODO */

//Enviando a mensagem
SendMsg ((STJ1939_MSG *)&msgVehicleWeight);

//Trace("SMS - Message transmitted successfully");
}/* End J1939 generated function - OnTimer_time100_100 */

/* Start J1939 generated function - OnDLL_Load */
void OnDLL_Load()
{
/* TODO */
msgVehicleWeight.m_sWhichBit = (VehicleWeight_*)malloc(sizeof(VehicleWeight_));
//updtating Data
msgVehicleWeight.m_sWhichBit-> mAxleLocation = AxleLocation;
msgVehicleWeight.m_sWhichBit-> mAxleWeight = AxleWeight;
msgVehicleWeight.m_sWhichBit-> mTrailerWeight = TrailerWeight;
msgVehicleWeight.m_sWhichBit-> mCargoWeight = CargoWeight;
}/* End J1939 generated function - OnDLL_Load */
/* Start J1939 generated function - OnDLL_Unload */
void OnDLL_Unload()
{
/* TODO */

free(msgVehicleWeight.m_sWhichBit);
}/* End J1939 generated function - OnDLL_Unload */

```