Universidade Federal de São Carlos - UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Engenharia Mecânica – DEMec
Curso de Engenharia Mecânica

Trabalho de Conclusão de Curso

REDES AUTOMOTIVAS: ANÁLISE TEÓRICA E EXPERIMENTAL

Matheus Falleiros de Almeida Valladão Flores

Orientador:
Prof. Dr. Rafael Vidal Aroca

São Carlos - SP- 2017

Universidade Federal de São Carlos - UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Engenharia Mecânica – DEMec
Curso de Engenharia Mecânica

Trabalho de Conclusão de Curso

Matheus Falleiros de Almeida Valladão Flores

REDES AUTOMOTIVAS: ANÁLISE TEÓRICA E EXPERIMENTAL

Trabalho de Conclusão de Curso apresentado à Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Rafael Vidal Aroca

São Carlos - SP- 2017

Matheus Falleiros de Almeida Valladão Flores

REDES AUTOMOTIVAS: ANÁLISE TEÓRICA E EXPERIMENTAL

Trabalho de Conclusão de Curso apresentado à Universidade Federal de São Carlos, como parte dos requisitos para obtenção do título de bacharel em Engenharia Mecânica.

Trabalho aprovado. São Carlos - SP, 2017:

Prof. Dr. Rafael Vidal Aroca
Orientador

Profª. Drª. Tatiana F. P. A. Taveira Pazelli
Professor Convidado

Prof. Dr. Daniel Varela Magalhães
Professor Convidado

São Carlos - SP
2017

Para meus pais, Selma e Sylvio, pelo amor e suporte incondicional. Todas as minhas conquistas são e sempre serão para vocês. Para meu irmão Gabriel, você é o cara!

AGRADECIMENTOS

Agradeço a todos que participaram de alguma forma da minha graduação, fazendo estes anos inesquecíveis e cheios de aprendizado.

Agradeço aos professores do Departamento de Engenharia Mecânica da UFSCar pelo aprendizado, em especial ao meu orientador Rafael, sem o qual este trabalho não seria possível, pela orientação técnica, por todo o aprendizado e incentivo constante. Você inspira a curiosidade nas pessoas.

Ao Matthias Mann, por acreditar em mim e me proporcionar uma experiência única de trabalho. Danke schön.

Aos meus amigos do Polo Aquático de São Carlos, pelo companheirismo, amizade, treinos exaustivos e campeonatos. Com vocês aprendi a ter mais disciplina e trabalhar em equipe.

Aos meus companheiros de turma, em especial Marie e Rodrigo, pelo companheirismo nestes anos todos.

A todos meus amigos e técnicos que foram corajosos o suficiente e emprestaram seus carros para eu testar minhas engenhocas.

A minha namorada, por continuar sendo minha namorada após este trabalho. Você é demais.

Finalmente aos meus pais, por me ensinar desde pequeno a correr atrás dos meus objetivos e me proporcionar todas as ferramentas para alcança-los.

"We keep moving forward, opening new doors, and doing new things, because we're curious and curiosity keeps leading us down new paths"

(Walt Disney)

RESUMO

A cada ano novos itens de conforto, segurança e tecnologia são adicionais aos veículos. No Brasil, por exemplo, já é lei que qualquer veículo nacional possua freios ABS (*Antiblockier-Bremssystem*) e *Airbag*. Estes sistemas necessitam elaborados sistemas eletrônicos e conexão de diversos sensores, tipicamente em rede. Além disso, a maioria dos veículos comercializados atualmente possuem uma ou mais redes de comunicação. Em especial, o uso da rede aumenta a confiabilidade, flexibilidade dos projetos e também reduz custos e peso final do veículo com a redução de cabos. Sabendo que as redes CAN (*Controller Area Network*) automotivas podem ser acessadas via porta de diagnósticos OBD-II (*On Board Diagnostics*), permitindo acesso aos códigos e consequente análise dos mesmos, propõe-se o estudo de tecnologias de redes veiculares, através de estudo teórico, análise experimental e análise de ferramentas comerciais. Para tal, objetiva-se primeiramente a comparação entre ferramentas comerciais e posteriormente a construção de um dispositivo para análise dos dados. Este dispositivo será composto por um microcontrolador que é conectado a um controller CAN e um transciever CAN para acoplar fisicamente os sinais elétricos do controlador CAN à rede, com capacidades bidirecionais e controle de direção dos dados. Este aparelho será utilizado para colher dados de veículos, para posterior análise. Espera-se assim, desenvolver ferramentas e conhecimento para a analise e atuação em redes CAN automotivas, comparando as ferramentas comerciais disponíveis, e assim, dando mais subsídios para futuras pesquisas, ensino e desenvolvimentos na área.

Palavras-chave: CAN Automotivo. OBD. Análise de Protocolo. Experimental.

LISTA DE ILUSTRAÇÕES

Figura 1 – Sem rede Automotiva 25
Figura 2 – Com rede Automotiva 25
Figura 3 – Chicote veículo 2002 26
Figura 4 – Chicote veículo 2008 26
Figura 5 – Topologia de redes automotiva em veículos a) - Compactos; b) - Luxo . 30
Figura 6 – Topologia Bus 31
Figura 7 – Topologia Estrela 31
Figura 8 – Topologia Anel 32
Figura 9 – Topologia Mesh 32
Figura 10 – Topologia híbrida 33
Figura 11 – Método de Endereçamento: a - orientado ao assinante; b - orientado a mensagem 33
Figura 12 – Mestre Escravo 35
Figura 13 – Modelo OSI 36
Figura 14 – Interface de transmissão UART 38
Figura 15 – Largura de banda de diversas redes 40
Figura 16 – Possíveis configurações de *Gateways* em rede automotiva 41
Figura 17 – Conector OBDII 43
Figura 18 – Típico nó de uma rede CAN 45
Figura 19 – Controlador *Basic* CAN 46
Figura 20 – Controlador *Full* CAN 46
Figura 21 – Filtando a interferência na rede CAN 47
Figura 22 – Níveis de Voltagem 49
Figura 23 – Formato da mensagem CAN 50
Figura 24 – *Remote Frame* 52
Figura 25 – *Error Frames* 53
Figura 26 – Overload Frames 53
Figura 27 – Estrutura de uma Mensagem CAN 55
Figura 28 – Envio de Mensagens Segmentadas 56
Figura 29 – Arbitragem CAN 57
Figura 30 – Arquiterura de rede da BMW série 7 60
Figura 31 – Diagrama do funcionamento SF-X9 67
Figura 32 – Diagrama ELM 327 69
Figura 33 – Top-Silk ELM 327 70
Figura 34 – PCB com montagem para ELM 327 70
Figura 35 – Conector OBD - Corolla Xei MY 2014 72
Figura 36 – Resposta a solicitação do número de chassi através do comando 0902 . 73
Figura 37 – USBTIN 73

Figura 38 – Diagrama do funcionamento SF-X9 80

LISTA DE TABELAS

Tabela 1 – Trecho de mensagens capturadas na rede CAN via porta OBD de um veículo nacional "popular" 27
Tabela 2 – Classificação de redes Automotivas 39
Tabela 3 – Modos descritos na SAE J1979 42
Tabela 4 – Pinos do conector OBD-II 43
Tabela 5 – Protocolos Padrão para Diagnóstico automotivo 44
Tabela 6 – Códigos do ELM327 66
Tabela 7 – Log USBTin - Teste I 74
Tabela 8 – Log ELM327 - Teste I (ATMA Manual) 74
Tabela 9 – Headers encontrados apenas no ELM 327 no teste I 75
Tabela 10 – Log USBTin - Teste II 76
Tabela 11 – Log ELM327 - Teste II (ATMA Automático) 76
Tabela 12 – Incidência de *header* nos testes com o Corolla Xei MY 2014 77
Tabela 13 – Log da variação da posição do pedal do acelerador (0 - 100%) - Toyota Hilux SRV MY 2015 79
Tabela 14 – Códigos enviados pelo dispositivo - RPM 80
Tabela 15 – Códigos enviados pelo veículo - RPM 81
Tabela 16 – Parâmetros e cálculo do RPM 81
Tabela 17 – Códigos enviados pelo dispositivo - Líquido Refrigerante 81
Tabela 18 – Códigos enviados pelo veículo - Líquido Refrigerante 82
Tabela 19 – Códigos enviados pelo veículo - Líquido Refrigerante 82
Tabela 20 – A simple longtable example 89

LISTA DE ABREVIATURAS E SIGLAS

4WD	Four Wheel Drive
ABS	Antiblockiersystem
ACC	Adaptive Cruise Control
ACK	Acknowledgment
ASC	Automatic stability Control
ASR	Anti Slip Regulation
BMW	Bayerische Motoren Werke AG
CAN	Controller Area Network
CAN_L	CAN Low
CAN_H	CAN High
CARB	California Air Resources Board
CF	Consecutive Frame
CNC	Computer Numerical Control
CRC	Cyclic Redundancy Checksum
CSMA/CD+AMP	Sense Multiple Access with Collision Detection and Arbitration on Message Priority
CSMA/CR	Carrier Sense Multiple Access/Collision Resolution
DLC	Data length Code
DTC	Diagnostic Trouble Code
ECU	Eletronic Control Unit
EDC	Eletronic Damper Control
EOBD	European On Board Diagnostics
EoF	End of Frame
EPS	Eletronic Power Steering
ESP	Eletronic Stability Control

EPA	Environmental Protection Agency
FIFO	First in, First Out
FF	First Frame
FCF	Flow Control Frame
FTDMA	Flexible TDMA
FTT-CAN	Flexible Time-Triggered CAN
IDE	Identifier Extension Flag
IFS	Inter Frame Spaces
ISO	International Standard Organization
MOST	Media Oriented Systems Transport
MY	Model Year
LIN	Local Interconnect Network
LSP	Least Significant Bit
OBD	On Board Diagnostics
OSI	Open Systems Interconnection
PCB	printed circuit board
PCI	Protocol Control Information
RTR	Remote Transmission Request
RAM	Random Access Memory
RMA	Rate Monotonic Analysis
RxD	Receive Data
SAE	Society of Automotive Engineers
SLIO	Serial Linked Input/Output
SRR	Substitute Remote Request
SF	Single Frame
TDMA	Tme Dividion Multiple Access

TTP/C	Time-Triggered Protocol Class C
TT-CAN	Time Triggered Controller Area Network
TxD	Transmit Data
UART	Universal asynchronous receiver/transmitter
USB	Universal Serial Bus
VIN	Vehicle Identification Number

SUMÁRIO

1 INTRODUÇÃO 25
1.1 Objetivos 27
1.1.1 Objetivos Gerais 27
1.1.2 Objetivos Específicos 27
1.2 Organização do Trabalho 28

2 EMBASAMENTO TEORICO 29
2.1 Redes 29
2.2 Topologia de Rede 29
2.2.1 Barramento 29
2.2.2 Estrela 31
2.2.3 Anel 31
2.2.4 Malha 32
2.2.5 Híbrida 32
2.3 Endereçamento 32
2.3.1 Método orientado ao assinante 33
2.3.2 Método orientado à mensagem 33
2.3.3 Método orientado à transmissão 34
2.4 Método de acesso ao barramento 34
2.4.1 *Time Division Multiple Access* (TDMA) 34
2.4.2 Mestre-Escravo 34
2.4.3 Multi-Mestre 34
2.5 Modelo de referência OSI 35
2.5.1 Camada Física 36
2.5.1.1 Nível de sinal 37
2.5.1.2 *Bit Stream* 37
2.5.1.3 Interface UART 37
2.5.2 Camada de Comunicação 38
2.5.3 Camada de Aplicação 38
2.6 Mecanismos de controle 38
2.6.1 *Composability* 39
2.7 Redes Automotivas 39
2.8 Multiplas Redes e Gateways 40
2.9 *On Board Diagnostics* 41
2.9.1 SAE J1979 42
2.9.2 OBD-II 43
2.9.3 Protocolos conhecidos 43

2.10 ***Controller Area Network* (CAN)** **44**
2.10.0.1 Topologia 44
2.10.1 Camada Física da rede CAN 45
2.10.1.1 *Bit Timing* e Sincronização 45
2.10.1.2 Sistema de Transmissão de Dados 45
2.10.1.3 Controlador CAN 46
2.10.1.4 Estados lógicos do Barramento - o trabalho do *transceiver* 47
2.10.1.5 Transmissão 48
2.10.1.6 Nível de Voltagem 48
2.10.1.7 Ausência de Reflexão na terminação 48
2.10.1.8 Limites de transmissão 49
2.10.2 Camada de Comunicação 49
2.10.2.1 *Data Frame* 50
2.10.2.2 *Remote Frame* 51
2.10.2.3 *Error Frame* 51
2.10.2.4 *Overload Frame* 52
2.10.3 Normas 53
2.10.3.1 ISO 15765-2 54
2.10.3.2 Identificador 54
2.10.4 Envio de Mensagens Maiores que 8 bytes 54
2.10.5 Endereçamento 55
2.10.5.1 Arbitragem 56
2.10.6 Variações da rede CAN 57
2.10.6.1 TT-CAN 57
2.10.7 CAN FD 58

3 TRABALHOS RELACIONADOS 59

4 ANÁLISE COMPARATIVA E RESULTADOS 65
4.0.1 Metodologia 65
4.1 ELM 327 . . . . . **65**
4.1.1 Códigos do ELM 327 66
4.1.2 Testes Preliminares com o ELM 327 66
4.1.2.1 Origem e Versão 67
4.1.3 Testes no veículo com o ELM 327 68
4.1.3.1 Mudança de *Baudrate* 68
4.1.4 Comparação entre *chip* original e cópias 68
4.1.4.1 Testes com ELM 327 Original 71
4.1.5 Testes com o Docklight 71
4.2 USBTin **73**

4.2.1 Comparação entre ELM 327 e USBTin 74
4.2.2 Testes preliminares com o USBTin 76
4.2.3 Testes com Máscara no *header* 78
4.3 SF-X9 . . . . . **78**
4.3.1 Testes com o SF-X9 78
4.3.1.1 Rotação do Motor 80
4.3.1.2 Temperatura do Líquido Refrigerante 81
4.4 Comparação **82**

5 RESULTADOS **83**

6 CONCLUSÃO **85**

REFERÊNCIAS **87**

A LISTA DE COMANDOS DE DIAGNÓSTICO MODO I - SAE J1979 89

B CÓDIGO PYTHON **95**
B.1 Mudança de Baudrate para ELM327 . . . . . **95**
B.2 Injetor de dados para ELM327 . . . . . **95**
B.3 Monitorar para ELM327 . . . . . **96**

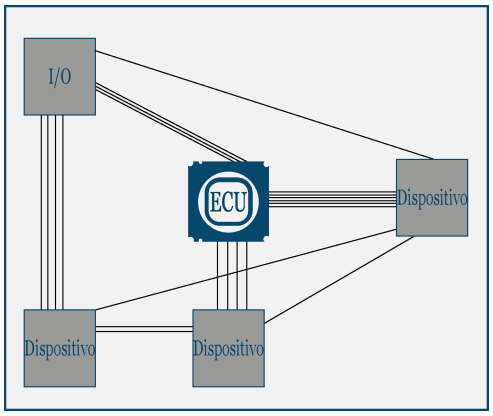
C LOG DE CÓDIGOS DA HILUX SRV MY 2015 . . . . . **97**
C.1 Trecho de Log do USBTin **97**
C.2 Trecho de Log do ELM 327 . . . . . **98**

1 . INTRODUÇÃO

O uso de eletrônica embarcada está cada vez mais presente em veículos automotores. Os custos de fabricação de veículos estão atingindo proporções até então existentes somente na indústria aeroespacial, sendo 1/3 dos custos gastos em carroceria, 1/3 no motor e propulsão e 1/3 em eletrônica embarcada (WOLF, 2007). Neste cenário criou-se a necessidade de uma rede que integrasse esta eletrônica.

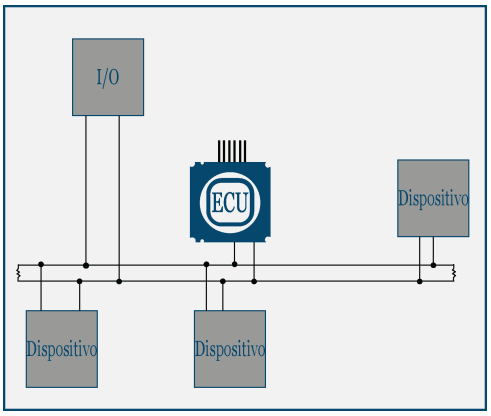
Em especial, o uso da rede aumenta a confiabilidade, flexibilidade dos projetos e também reduz custos e peso final do veículo com a redução de cabos. A Figura 1 mostra um exemplo de veículo sem rede automotiva, onde dezenas, ou até centenas de cabos devem ser conectadas a uma unidade de controle eletrônico (ECU), enquanto na nova abordagem, com rede mostrada na Figura 2, apenas um par de cabos, que estabelece a rede, interliga todos componentes do veículo, e processadores "nas pontas" da rede, fazer a interface com os dispositivos de entrada e saída.

Figura 1 – Sem rede Automotiva



Fonte: adaptado de (NATIONAL INSTRUMENTS, 2014)

Figura 2 – Com rede Automotiva



Fonte: adaptado de (NATIONAL INSTRUMENTS, 2014)

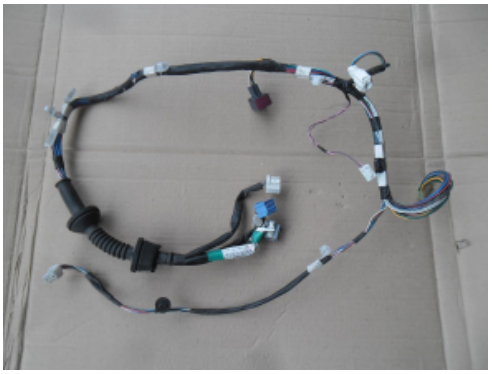
Tal fato pode ser observado com um exemplo simples ilustrado na Figuras , que mostram dois "chicotes" (conjunto de cabos) de um veículo de mesma marca e modelo para os anos 2002 e 2008, respectivamente. O chicote, basicamente, é um arranjo com vários cabos agrupados para a conexão de algum subsistema de um veículo. No exemplo das Figuras 3 e 4. , vê-se o chicote da porta esquerda do motorista de um veículo. Este chicote precisa levar cabos energia, sinal de acionamento de portas,vidros, porta dos passageiros, sensores de alarme, dentro outros. Em 2002 o chicote possuía um fio para cada sensor e atuador, enquanto em 2008, o chicote, usa cabos mais finos e em menor quantia, pois está apenas transmitindo dados digitais de comunicação em rede. Deve-se observar que este chicote passa através da carroceria para a porta, devendo suportar muitos ciclos de abertura e fechamento da porta. Sendo o chicote mais simples, com menos cabos, sua durabilidade é maior e a manutenção mais simples e barata. Com a rede CAN, ao invés

de ser necessário um fio para acionar os vidros de cada porta, um sinal digital gerado por um processador é propagado pela rede por todo carro, acionando os dispositivos devidos.

Figura 3 – Chicote veículo 2002

Fonte: Mercado Livre

Figura 4 – Chicote veículo 2008



Fonte: Mercado Livre

Em especial, destaca-se o uso da tecnologia de rede *Controller Area Network* (CAN), para conectar desde sistemas de coleta de dados de sensores e acionamento de atuadores críticos para o funcionamento de um veículo a sistemas de diagnóstico exigidos por força da lei.

Nos veículos há um conector de diagnósticos, presente em todos os veículos do mercado por força de lei, que possibilita acesso a diversos dados de sensores e operação do veículo. Em geral, esta conexão é chamada de On Board Diagnostics (OBD). Embora esta ferramenta de diagnóstico seja padronizada até certo ponto, pelo menos na camada física de rede, existem inúmeros padrões e protocolos de rede para seu uso, dentre os quais se destacam os protocolos SAE J1850-PWM, SAE J1850-VPW, ISO 9141-2, ISO 14230-4, ISO 15765-4, SAE J2411, KW1281 (SAE J2818), SAE J1939, SAE J1708 (J1587), SAE J1708 (J1922). Alguns destes protocolos são abertos e padronizados, contudo a maioria são fechados, ou especificados em normas de difícil acesso. Além disso, é comum os fabricantes implementarem mensagens e códigos especiais além da especificação determinada pelas normas. O conhecimento e uso das mensagens destes protocolos pode ser de grande valia para pesquisas acadêmicas, como por exemplo, para analisar o desempenho de motores, níveis de poluição, índices de dirigibilidade, estimar falhas e seus motivos, sistemas de telemetria, dentre outros, ou mesmo para desenvolver veículos autônomos apenas conectando um computador à porta CAN de um carro de mercado. De fato, vários modelos de carros podem ter a velocidade, frenagem e esterçamento controlados via comandos da rede CAN (VALASEK; MILLER, 2014). A Tabela 1 mostra um breve exemplo de mensagens CAN capturadas via porta OBD de um veículo durante os testes preliminares deste projeto de pesquisa.

Dessa forma, o foco deste projeto encontra-se no estudo teórico e experimental de redes automotivas e seus protocolos, para estabelecer uma base de conhecimento sobre a

Tabela 1 – Trecho de mensagens capturadas na rede CAN via porta OBD de um veículo nacional "popular"

442	40	12	00	00	02	00	00	00
630	17	80	00	00	00	00	00	00
440	42	02	00	00	00	00	00	00
758	40	03	61	E1	00	00	00	00
610	20	00	00	64	00	00	00	00
750	40	02	21	A1	00	00	00	00
758	40	03	61	A1	00	00	00	00
442	40	12	00	00	02	00	00	00
750	40	02	21	A2	00	00	00	00
758	40	04	61	A2	00	00	00	00
620	10	00	00	00	00	00	00	80

Fonte: Próprio Autor

coleta de dados disponibilizadas na rede veicular, sua interpretação e uso. Para realizar tal trabalho, é necessário o entendimento teórico de uma rede automotiva e o trabalho sistemático de análise e engenharia reversa de protocolos automotivos.

1.1 Objetivos

1.1.1 Objetivos Gerais

O objetivo principal do projeto é o estudo de tecnologias de redes veiculares, através de estudo teórico, análise experimental e engenharia reversa através de análise de dados de um de veículos. Os dados capturados serão analisados para fins de pesquisa ou aplicações de indicação de possíveis falhas, índices de dirigibilidade, economia, dentre outros parâmetros. Uma aplicação demonstrativa que poderá ser implementada, é analisar a rotação do motor através da análise e conversão dos códigos da rede. Para realizar o projeto, será necessário conectar dispositivos de diagnóstico à porta OBD II de um veículo. Assim, um objetivo do projeto é também a análise e comparação de dispositivos capazes de se conectarem a rede automotiva através da porta OBD II.

1.1.2 Objetivos Específicos

- Estudo e revisão bibliográfica de redes de comunicação automotivas;
- Análise de dispositivos comerciais capazes de se conectarem à rede através da porta OBDII;
- Engenharia reversa (parcial) de alguns protocolos de comunicação em redes automotivas;

- Implementação de software e procedimento para decodificação e interpretação de dados dos protocolos;
- Construção de um ou mais equipamentos de diagnóstico para porta OBDII, para captura e análise de dados na rede veicular;

1.2 Organização do Trabalho

No Capítulo 2 é realizada uma análise teórica sobre redes automotivas, evidenciando sua forma construtiva,funcionamento, protocolos, e normas. A rede CAN será então enfatizada, ressaltando-se suas particularidades e explicando a sua camada física, seu protocolo, e sua implementação. Como a empresa Robert Bosch GmbH foi a criadora da rede CAN, basearemos parte do embasamento teórico em sua literatura.

O Capítulo 3 é dedicado a uma análise de trabalhos relacionados na área, evidenciando as pesquisas mais relevantes encontradas na área.

O Capítulo 4 contém uma abordagem experimental da rede automotiva. Nele será analisado e comparado dois importantes produtos comerciais, o ELM327 e o USBTin, de conexão à rede CAN, mostrando em detalhes seu funcionamento e suas capacidades. Em seguida é mostrado a construção de um esquipamento de conexão à rede capaz de isolar a direção dos códigos. Por fim será feita uma análise prática dos códigos de uma Toyota Hilux SRV 2015, contemplando a conversão de dados *raw* da rede em valores reais de medição para a rotação do motor e para a temperatura do líquido refrigerante.

O Capitulo 5 exemplifica a importância do domínio de ferramentas que possam se conectar à rede CAN, citando possíveis aplicações comerciais e suas vantagens econômicas.

Ainda, os apêndices possuem tabelas resumidas de códigos padronizados, e códigos em Pyhton para a utilização com os hardwares comerciais.

2 . EMBASAMENTO TEORICO

2.1 Redes

Segundo (TANENBAUM; WETHERALL, 2010) redes são um sistema no qual um número grande de computadores separados, mas interconectados, servem necessidades computacionais.

No âmbito automotivo, rede é sistema no qual um grupo de elementos podem trocar informação através de um meio (BOSCH, 2014).

Os elementos podem ser interpretados como nós e são referenciados comumente como assinantes. Os meios de comunicação são dispostos normalmente como linhas e recebem o nome de barramento de uma rede (*Network Bus*). Na eletrônica automotiva, tanto sistemas complexos como o de injeção eletrônica como simples sensores podem ser assinantes em uma rede (BOSCH, 2014).

A topologia de uma rede é compreendida como a estrutura que engloba nós e conexões. Ela mostra quais nós estão interconectados e de que forma isto é feito. Cada assinante deve ter pelo menos uma conexão com outro assinante para participar da rede. Dessa forma a topologia determina algumas características gerais da rede (BOSCH, 2014). Exemplos de rede de carros compactos e de luxo estão mostrados na Figura 5.

2.2 Topologia de Rede

A palavra topologia vem do grego *topos* que significa lugar e *logos* que significa teoria ou conhecimento. Desta forma a topologia é a ciência da localização (LAWRENZ, 2013).

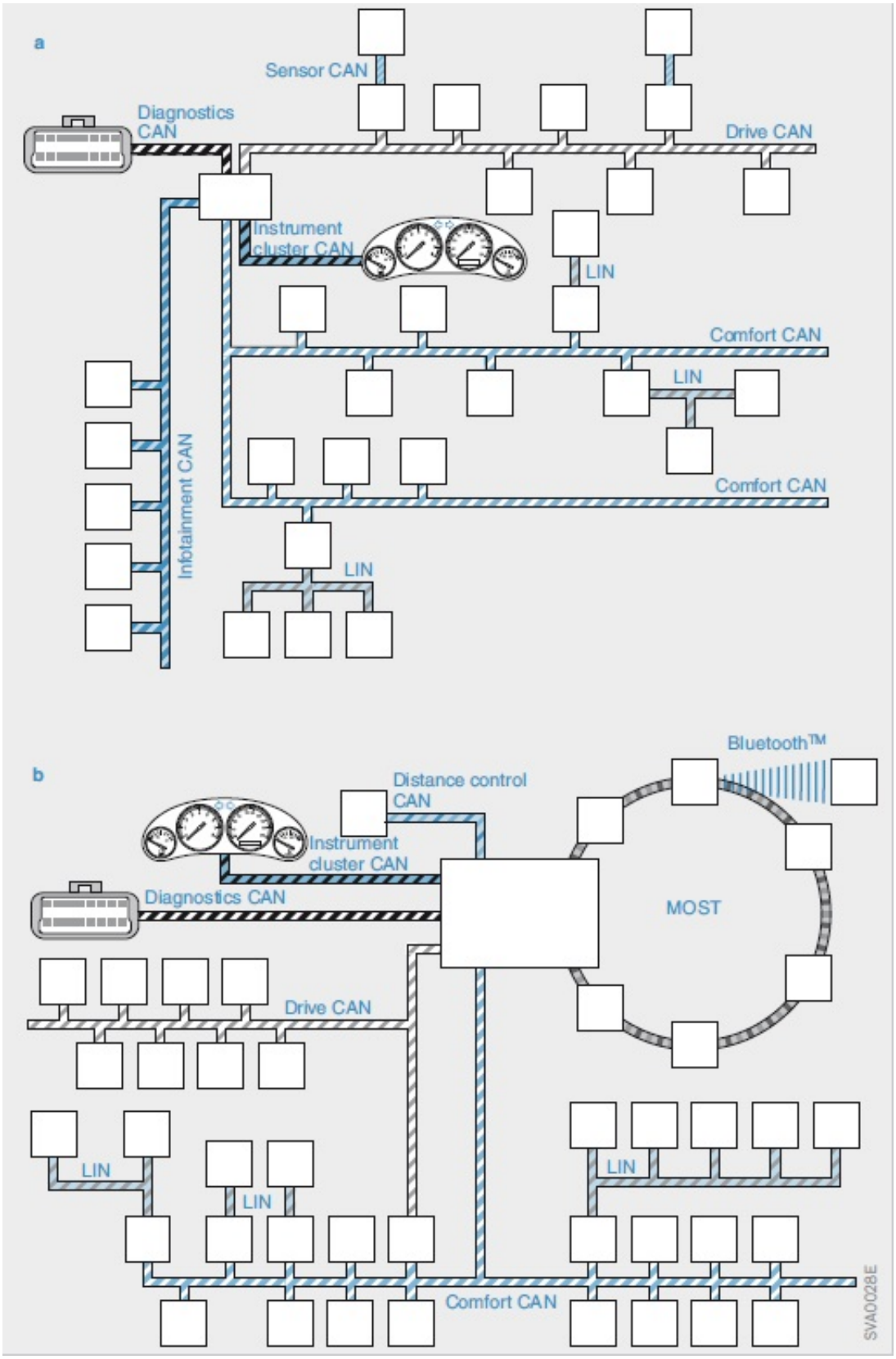
A Topologia de rede é entendida como uma estrutura consistente de nós e conexões (BOSCH, 2014), e a sua forma construtiva tem impacto em como os nós exercem influência ou reagem a influências (LAWRENZ, 2013).

Os diagramas mostram quais elementos estão interconectados em quais nós, e não mostra detalhes como as dimensões das conexões. As topologias de rede são compostas de quatro tipos basicos, ou alguma de suas combinações:Barramento, estrela, anel (*ring*), mesh e híbridos (combinações entre as anteriores) (BOSCH, 2014).

2.2.1 Barramento

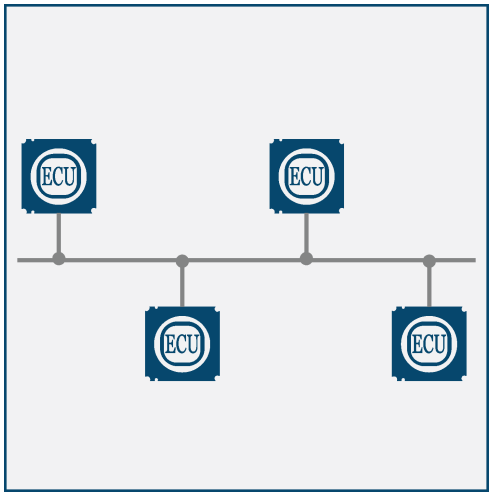
Também conhecido como linear, o elemento básico é o cabo único no qual os elementos são conectados. No caso específico da rede CAN, a rede pode se estender por 40m, com 30cm para as ramificações. (ROAD. . . , 2003).

Figura 5 – Topologia de redes automotiva em veículos a) - Compactos; b) - Luxo



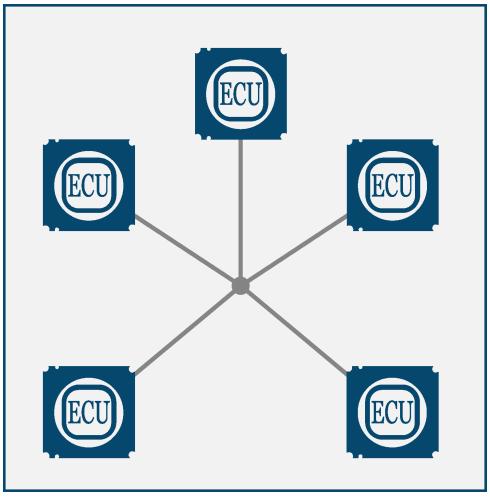
Fonte: (BOSCH, 2014)

Figura 6 – Topologia Bus



Fonte: adaptado de (BOSCH, 2014)

Figura 7 – Topologia Estrela



Fonte: adaptado de (BOSCH, 2014)

Todos os nós transmitem e recebem e, se um nó falha, o restante do barramento deixa de estar disponível (BOSCH, 2014). A Figura 6 exemplifica está construção.

2.2.2 Estrela

A topologia estrela é baseada em um *hub* repetidor ou nó central. Sua vantagem está na facilidade de se extender, e caso uma conexão falha esta não afeta as demais.

Há a distinção entre ativa e passiva, sendo que na ativa o nó central possui um computador que interpreta os sinais e os redistribui e, por consequência, sua capacidade diretamente ligada a performance do computador central. Na passiva o nó central é apenas uma ponto em curto físico. A desvantagem é que uma falha no nó central acarreta em uma falha na rede inteira.

Seu uso está em discussão para sistemas de segurança como freios, *airbag*, direção, e para aumentar a segurança pode ser construído com redundâncias no nó central (vários nós centrais físicos) (BOSCH, 2014), como a dupla estrela.

Segundo (LAWRENZ, 2013) o comprimento de cada cabo para uma rede CAN pode chegar a até 9m e a velocidade a 500 kbits/s, sendo utilizado apenas um resistor de 60 ohms no nó central para a terminação.

2.2.3 Anel

Na topologia em anel cada nó é conectado com outros dois vizinhos. Existem duas variações denominadas simples e duplo (BOSCH, 2014).

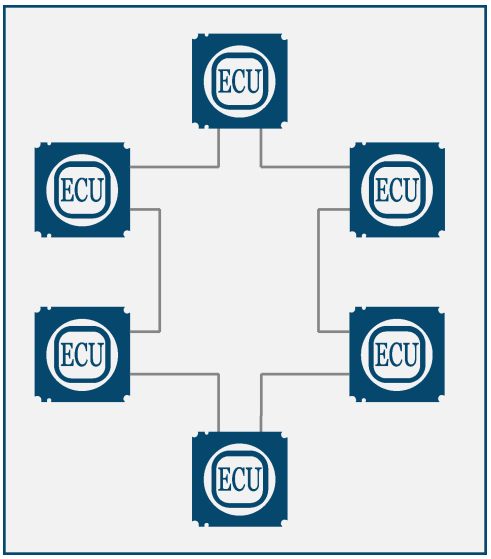
Cada nó processa e vê se é endereçado à ele, caso não seja ele passa adiante de forma unidirecional. Caso o pacote realize uma volta completa ele é descartado. Neste

caso, se uma estação (nó) falha, toda a rede falha. Na dupla, os pacotes são transferidos em duas direções. Caso uma estacao falha, a rede não falha. Mas se muitas começam a falhar, a rede deixa de funcionar.

2.2.4 Malha

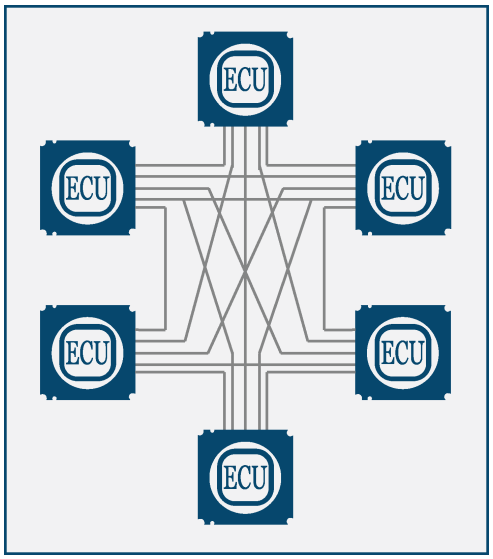
Na topologia em malha cada nó é conectado a um ou mais nós. Em uma malha completa cada nó é conectado a todos os outros nós. Esta topologia possui uma alta estabilidade, pois se um assinante falha ele pode ser redirecionado.

Figura 8 – Topologia Anel



Fonte: adaptado de (BOSCH, 2014)

Figura 9 – Topologia Mesh



Fonte: adaptado de (BOSCH, 2014)

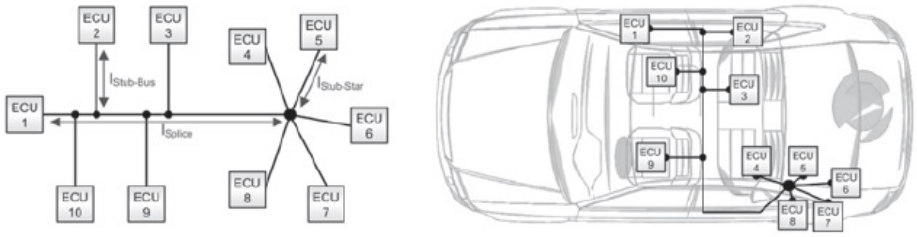
2.2.5 Híbrida

A Topologia Híbrida é a união de duas ou mais topologias, combinando as vantagens e desvantagens de cada topologia. Por exemplo em uma união entre a topologia barramento e estrela mostrada na figura 10 a velocidade máxima aumenta para 1 Mbit/s e o comprimento total de fios pode ser diminuído, porém a um custo maior (LAWRENZ, 2013).

2.3 Endereçamento

Para possibilitar a transmissão via rede as mensagens possuem informações sobre esta transferência de dados dados. Isto pode estar contido dentro da mensagem ou estar implícito usando condições ou valores pré estabelecidos, e é fundamental para que a mensagem seja entregue ao destinatário correto. Existem intrinsecamente três maneiras distintas de endereçamento chamadas de método orientado ao assinante, método orientado à mensagem e método orientado à transmissão (BOSCH, 2014).

Figura 10 – Topologia híbrida



Fonte: (LAWRENZ, 2013)

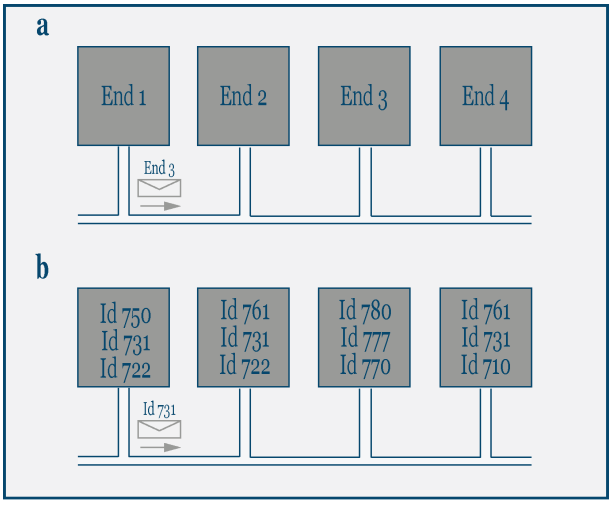
2.3.1 Método orientado ao assinante

Este método consiste em transmitir dados que contenham o endereço do destinatário. Todos os nós comparam o endereço com o seu próprio e apenas processam a mensagem caso os endereços correspondam A internet é um bom exemplo deste método de endereçamento. A figura 11 a representa este método.

2.3.2 Método orientado à mensagem

Neste método não é o destinatário que é endereçado, mas mensagem que possui um identificador pré estabelecido. Os assinantes são então programados para ler determinadas mensagens e não precisam saber nada sobre o destinatário da mensagem. Neste caso as mensagens podem ser recebidas por múltiplos destinatários (BOSCH, 2014). A rede CAN utiliza este método de endereçamento. A figura 11 b representa este método.

Figura 11 – Método de Endereçamento: a - orientado ao assinante; b - orientado a mensagem



Fonte: adaptado de (BOSCH, 2014)

2.3.3 Método orientado à transmissão

Neste caso o que define a o processamento da mensagem são características temporais. Caso uma mensagem seja enviada em uma determinada lacuna no tempo ela é processada. Este método pode ser combinado com outro quando em busca de maior segurança (BOSCH, 2014).A rede LIN (*Local Interconnect Network* e a TT-CAN (*Time Triggered CAN* utilizam este modelo.

2.4 Método de acesso ao barramento

Um nó deve ser capaz de acessar o barramento a fim de transmitir uma mensagem. Este acesso pode ser feito de forma determinística, ou seja, com características temporais definidas, ou randômica. No primeiro caso o acesso é bem definido e individual enquanto no último caso, há a possibilidade de múltiplos acessos causarem colisões (BOSCH, 2014).

2.4.1 Time Division Multiple Access (TDMA)

Neste método, o tempo de acesso à rede é pré definido como uma janela temporal para cada nó que se repete indefinidamente, o que requer sincronia entre os *clocks* internos dos mesmo, porém que possibilita acessos singulares a rede e, por extensão, evita colisões de dados.

Esse método é muito utilizado na indústria de telecomunicação e possui uma variante, a *Dynamic time division multiple access*, que pode alocar tempos diferentes para cada usuário assim como frequências de alocação de tempo (ELETRONIC DESIGN, 2013). Este método se mostra limitado em possibilidades dado que o tempo é uma grandeza também finita.

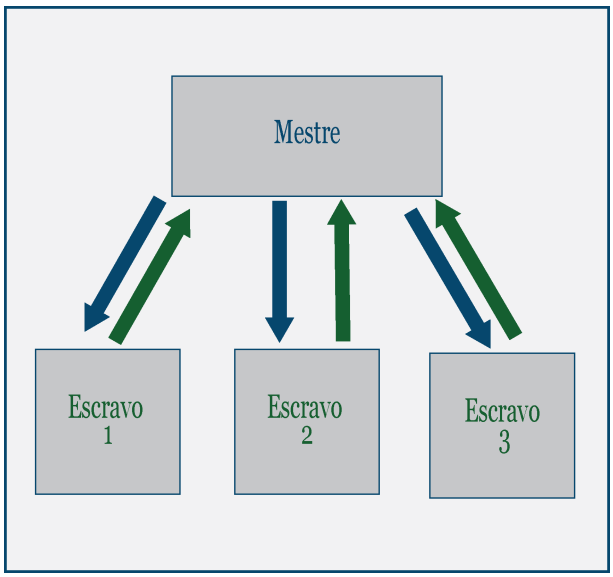
2.4.2 Mestre-Escravo

Este método acesso à rede é tal que um dos nós é denominado mestre enquanto os demais são referidos como escravos 12. O mestre determina a frequência de comunicação através de solicitações de dados dos escravos, que apenas então acessam a rede. Existem ainda alguns protocolos que permitem que os escravos comuniquem com os mestres espontaneamente para transmitir uma mensagem (BOSCH, 2014). Este nó permite uma grande flexibilidade em aplicações e é utilizado na rede LIN.

2.4.3 Multi-Mestre

Um terceiro método, onde diversos nós podem acessar a rede independentemente da interferência de um outro nó, caso a rede aparente estar livre. Cada nó age desta forma como seu próprio mestre e necessitam de recursos adicionais de detecção de colisão como priorização ou repetições de mensagens.

Figura 12 – Mestre Escravo



Fonte: adaptado de (BOSCH, 2014)

Apesar de sua maior complexidade, este método possui a vantagem que a falha de um nó específico não acarreta na falha geral da rede. (BOSCH, 2014).

A rede CAN utiliza este método de acesso à rede e seu desempenho em termo de atrasos e, por extensão, a capacidade de transmissão *real-time* é diretamente correlacionada a alocação correta no barramento. Isso acontece porque quando um nó percebe que o barramento está ocupado, a mensagem é postergada para o próximo momento em que o barramento esteja livre (LAWRENZ, 2013).

2.5 Modelo de referência OSI

O modelo ISO OSI (*Open System Interconnection*) foi desenvolvido por uma parceria entre a ISO (*International Standardization Organization*) e a então chamada CCITT (*International Telegraph and Telephone Consulttive Committee*) com início em 1977 e permanece como referência desde sua publicação em 1984 como ISO 7984.

O propósito do modelo de referência OSI é prover uma base comum para a coordenação do desenvolvimento de normas para sistemas interconectados (INFORMATION..., 1994).

Este modelo conceitual normaliza sistemas de comunicação sem levar em consideração suas especifidades e tecnologias internas e representa a base de comparação entre diferentes protocolos de comunicação.

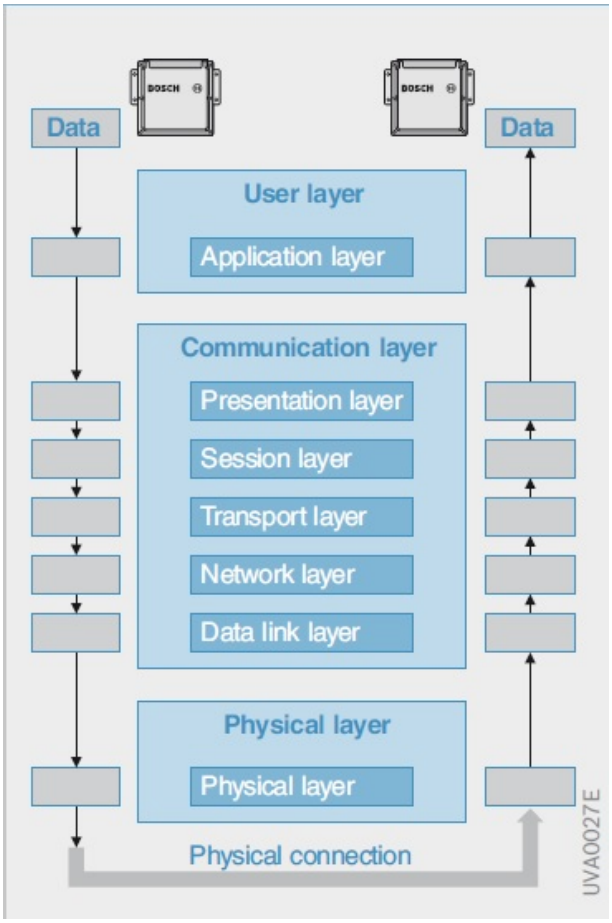
Os protocolos são usualmente definido em camadas de características e funções definidas. Ao subir as camadas, as propriedades da camada inferior são assumidas, ou

seja, camadas servem as camadas imediatamente superiores e são servidas pelas camadas imediatamente inferiores. Qualquer uma das sete camadas idealizadas na versão original são intercambiáveis desde que a interface entre as camadas permaneçam (BOSCH, 2014).

O modelo OSI representa sistemas de comunicação de dados em diferentes camadas funcionais denominadas *layers* como mostrado na Figura 13.Uma camada deve ser criada quando uma abstração diferente deve ser utilizada e deve performar uma função bem definida (TANENBAUM; WETHERALL, 2010).

Usualmente, em sistemas simples de comunicação, como aplicações automobilísticas, estes sistemas são divididos em três camadas: Física, Comunicação e Aplicação (BOSCH, 2014).

Figura 13 – Modelo OSI



Fonte: (BOSCH, 2014)

2.5.1 Camada Física

A camada física é responsável pelas conexões físicas entre os assinantes e as características elétricas da mesma. Nela são definidas os meios de transmissão, como fios de cobre ou ondas de rádio, incluindo características como voltagem, impedância,

frequências entre outras. A camada física determina também a topologia, o modo de transmissão, e é por fim responsável pelo recebimento e transmissão de dados *raw*, cabendo a esta camada garantir que quando se envia um bit 1 ela seja recebido como 1 e não 0 (TANENBAUM; WETHERALL, 2010).

2.5.1.1 Nível de sinal

Para se representar dados digitais utiliza-se binários 0 e 1, que devem ser reflexões de alguma forma física nos meios de transmissão. Os binarios podem ser representados por diferenças de voltagem, onde a maior voltagem é denominada *high* (estado 1) e a menor denominada *low* (estado 0). Desta forma, sequências de variações entre voltagens podem ser transformadas em sequencias de código binário em uma camada superior e posteriormente interpretadas. É importante ressaltar que um nível que sobrepõe o outro é chamado de nível dominante enquanto o nível sobreposto de recessivo (BOSCH, 2014).

2.5.1.2 Bit Stream

Informações não podem ser simplesmente transmitidas diretamente. Para serem transmitidas em uma rede CAN eles precisam ser incorporadas a um corpo de dados (*payload*) no *frame* de uma mensagem, que por sua vez contém as informações a serem transmitidas. Este *frame* precisa então ser convertido a uma sequencia de bits, denominada *Bit Stream* para fisicamente transmitir a mensagem no meio físico. Esta transmissão é então realizada como uma variação de níveis de sinal, como descrito anteriormente (BOSCH, 2014).

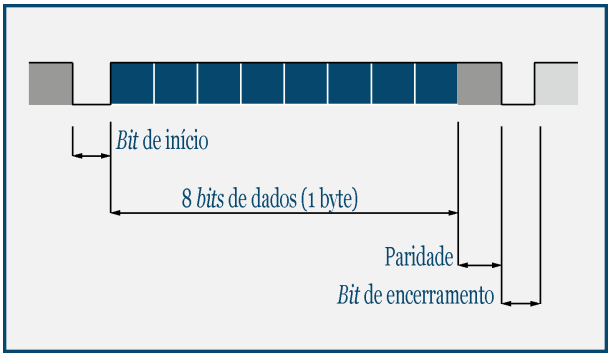
2.5.1.3 Interface UART

Microcontroladores de unidades de controle possuem uma simples interface UART (*Universal Asynchronous Reciever/Transmitter*) no chip, pela a qual eles se comunicam com computadores e outros dispositivos.

Caso o barramento esteja *idle* o nível de tensão é o nivel de operação do microcontrolador (por exemplo 5v). O bit de inicio (*Start Bit*) da comunicação é transmitido no nível dominante, 0v, e notifica o destinatário de que uma transmissão está tendo início. A duração do bit inicial determina o tempo de transmissão de todos os bits seguintes, e isto define por consequencia a taxa de transmissão de dados. Após receber o bit de inicio a transmissão de 8-bit de dados (1 byte) começa com o bit de menor significancia (*LSB).*

Os 8-bit de dados são seguidos do bit de paridade (*Parity bit*), que informa se o numero de *1* transmitidos é par ou ímpar, e possibilita testes simples de erros. A sequência é por fim completa com o bit de encerramento (*Stop Bit*) em nível dominante, para então retornar-se ao nível recessivo e possibilitar a próxima mensagem

Figura 14 – Interface de transmissão UART



Fonte: adaptado de (BOSCH, 2014)

2.5.2 Camada de Comunicação

Esta camada é responsável pela linguagem presente na troca de dados. É nela que são definidas as regras de comunicação, prepara mensagens recebidas na camada de aplicação e passa elas para a camada física, agindo como um interpretador de mensagens digitais. Nesta camada são definidos o formato do *frame* da mensagem, assim como o controle de acesso ao barramento, o endereçamento, controle de colisões e sincronia, e o cálculo do *checksum*. Esta camada representa uma simplificação para a aplicação automotiva das outras 5 camadas intermediária propostas originalmente no modelo OSI (BOSCH, 2014).

2.5.3 Camada de Aplicação

Esta camada representa a interface com o usuário ou máquinas, que recebe e processa informação e é comumente encontrada na forma de softwares. Ela contém uma variedade de protocolos necessárias para os usuários, como HTTP utilizado na web (TANENBAUM; WETHERALL, 2010).

2.6 Mecanismos de controle

O controle de envio de mensagens na rede pode ser distinto dependendo da aplicação. Para eventos singulares, como ligar o pisca alerta, as mensagens são enviadas com o acontecimento do evento. Neste caso não ha sincronia e podem haver colisões de pacotes na rede. Para evitar este tipo erro se utilizam métodos de análise do estado da rede assim como atrasos de mensagens, proporcionando reações imediatas para eventos randômicos. Erros podem ocorrer com alto uso da banda de rede, que impede mensagens de serem transmitidas, porém a rede tende a ser menos utilizada a medida que apenas mensagens úteis são enviadas. A impossibilidade de se provar que uma mensagem foi transmitida no momento correto é uma desvantagem adicional deste método (BOSCH, 2014).

Para sistemas mais complexos onde a confiabilidade é imprescindivel, como sistemas de direção elétricas, é vantajoso utilizar um método baseado em tempo, onde assinantes possuem uma determinada lacuna fixa e recorrente para enviar suas mensagens. Esta solução visa garantir o recebimento dos pacotes no tempo, diminuir a latência, e aumentar a confiabilidade do sistema, apesar da limitação do número de envios com o tempo, e da necessidade de uma grande sincronia entre os nós. Cada nó tem sua mensagem transmitida e após o ultimo concluir, o cíclo se re-inicia (BOSCH, 2014).

2.6.1 Composability

O termo *composability* se refere a capacidade de um subsistemas serem agregados a um sistema sem que haja alguma alteração nas funções deste subsistema. Desta forma, se um sistema de comunicação pode ser composto desta forma, a introdução, remoção ou alteração de uma unidade de controle podem ser feitas sem alterar a funcionalidade de outras unidades de controle, o que representa assim a unica forma de aumentar a complexidade de sistemas automotivos (BOSCH, 2014).

2.7 Redes Automotivas

A escolha de redes automotivas depende de uma série de fatores. A taxa de transferência limita o volume de dados a serem trocados na rede por unidade de tempo, enquanto a imunidade a interferência tenta garantir a consistência dos dados enviados e recebidos. Isto é feito através dos *bit* de paridade que mostra se o número de *1* recebidos é par ou ímpar, ou pelo método de *checksum*, no qual uma soma dos dados a serem enviados é feita por método pré estabelecido, enviada junto com o próprio pacote, e comparada com o valor recebido. Por fim a necessidade de capacidades de trabalho em tempo real, no qual não pode haver atrasos significativos na transmissão, assim como o número de nós a ser instalado na rede são outras características a serem definidas (BOSCH, 2014).

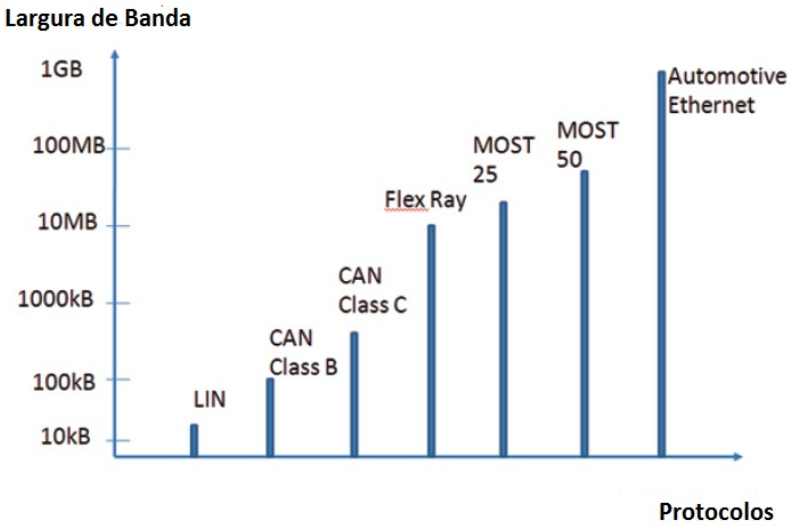
Para padronizar as possibilidades de redes automotivas, classifica-se elas quanto a sua classe, que se encontram resumidas na Tabela 2.

Tabela 2 – Classificação de redes Automotivas

Classe	Taxa de transferência	Aplicações	Representantes
Classe A	baixas (até 20kBit/s)	Atuadores e sensores	LIN
Classe B	Medias (até 125 kBit/s)	ECus de conforto	CAN baixa velocidade
Calsse C	Altas (até 1 MBit/s)	ECUs Powertrain	CAN alta velocidade
Classe C+	Ext.Altas (até 10 MBit/s)	Sistema de direção	FlexRay
Classe D	Ext.Altas (> 10 MBit/s)	Telematica e multimedia	MOST

Fonte: Próprio Autor

Figura 15 – Largura de banda de diversas redes



Fonte: Adaptado de (LAWRENZ, 2013)

2.8 Multiplas Redes e Gateways

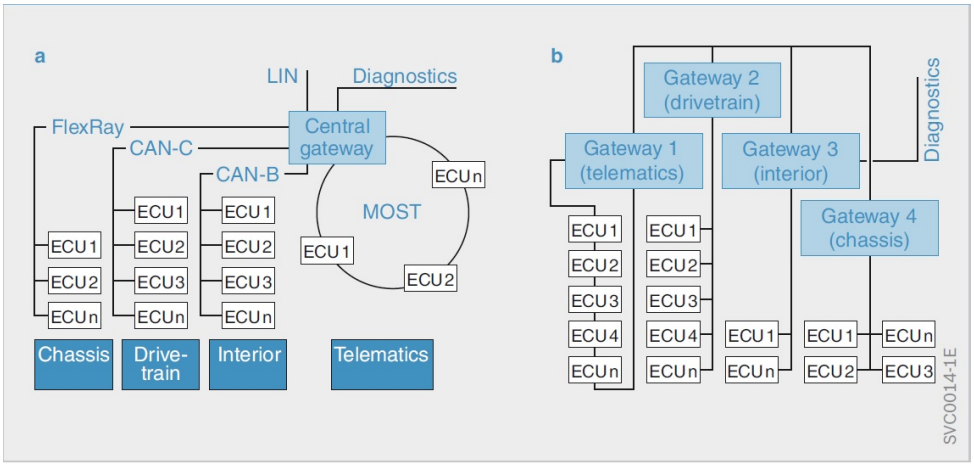
A crescente complexidade dos sistemas eletrônicos automotivos gerou o desenvolvimento de diversas soluções para redes automotivas. O controle de sistemas de multimídia podem ser controlados por redes classe B, porem a transferência de áudio e vídeo através da rede requer redes de classe D, como a *Media Oriented Systems Transport* (MOST). Redes LIN de menos de 20 KBit/s possuem baixo custo e são empregadas para conectar sensores e atuadores em redes de conforto. Redes CAN de 500 kBit/s são adequadas para o controle do *powertrain*. Desta forma a solução mais viável é o uso de múltiplas redes, que se interligam.

Porém, os diferentes tipos de rede porem possuem protocolos diferentes. Uma exemplificação disso seria se elas fossem consideradas como países distintos, cada um com sua língua própria, em uma sala de discussão. Para poder se comunicar seriam necessários tradutores que escutam em uma língua e repassam a mensagem em outra. Estes tradutores, na esfera eletrônica, são os chamados *Gateways* e podem ser centralizados (um para toda a rede) ou se encontrarem espalhados pelas redes como mostrado na Figura 16.

Os *gateways* também podem servir para tradução entre mesmos protocolos que operam com velocidades diferentes. Muitos veículos possuem múltiplas redes CAN que trabalham a taxas diferentes de transmissão de dados. Normalmente a rede que contém o motor é a rede de alta velocidade, enquanto a rede que contém as ECUs de conforto trabalha a médias ou baixas velocidades. Neste caso, pode existir um Gateway dedicado a comunicação entre as duas redes, ou uma ECU (normalmente a de conforto) que faz esta tradução.

Os *gateways* têm papel fundamental também para o diagnóstico automotivo. Ao conectar-se na porta OBD-II obtêm-se acesso à rede CAN de alta velocidade, e esta se comunica com o resto das redes. Desta forma, pode-se padronizar os dispositivos de diagnósticos, que não precisam prever todas as possibilidades de redes terminadas no conector de diagnóstico.

Figura 16 – Possíveis configurações de *Gateways* em rede automotiva



Fonte: (BOSCH, 2014)

2.9 On Board Diagnostics

O sistema OBD foi criado nas décadas de 1970 e 1980 para possibilitar o diagnóstico básico dos sistemas eletrônicos de injeção e assim ser complacente com as restrições de emissão EPA (*Environment Protection Agency*) nos EUA. Neste época surgiram as legislações de emissão de óxidos nitrosos, partículas e ruido nos EUA, Japão e Europa. Desta forma o sistema OBD foi criado para possibilitar a aquisição de dados e o controle de sistemas, alem de possibilitar o desenvolvimento de veículos complacentes com as futuras legislações.

Em 1988 a SAE recomendou a padronização dos protocolos e conectores (MCCORD, 2011), e assim o OBD-II foi introduzido na década de 1990 a fim de padronizar o sistema de diagnósticos.

Em 1996 a *California Air Resources Board* (CARB) expede a especificação do padrão OBD-II a ser implementado em todos os veículos a partir do ano-modelo (MY) 1996 a serem vendidos no mercado norte americano (CALIFORNIA AIR RESOURCES BOARD, 2016).

Em 2001 e 2003 a União europeia faz o mesmo com o seu equivalente EOBD para os veículos a gasolina e diesel respectivamente, a serem vendidos no mercado europeu.

Como consequência da padronização da produção e da exportação de tecnologia da Europa e dos EUA, é cabível extrapolar, que a grande maioria dos veículos leves do mundo produzidos após 2003 possuem o conector OBD-II. Hoje em dia o sistema de diagnóstico veicular é feito exclusivamente através do conector OBD-II, permitindo leitura e programação de todos os sistemas veiculares, de motores, a chassi, *body*, injeção eletrônica, freios, suspensão, entre outros. Para tal conecta-se aparelhos externos de diagnósticos. O conector e sua integração com as redes automotivas está exemplificado na Figura 5.

Em resumo, o sistema OBD é um conjunto de hardware e software capaze de disponibilizar informações relevantes sobre as emissões de poluentes dos veículos. Incluso neste sistema está o conector OBD-II que é a porta de acesso a este sistema. A comunicação deste sistema pode ser feita de diferentes forma, sendo uma delas a CAN.

De forma análoga, a norma ISO 11898-2 especifica a construção da rede CAN de alta velocidade, com características físicas e elétricas. A norma ISO 15765 determina o protocolo utilizado para comunicação em uma rede CAN construída conforme a norma ISO 11898. A norma SAE J1997, por sua vez, especifica os códigos de diagnósticos em uma rede automotiva, que satisfaçam os requisitos das regulações de OBD norte americanas e europeias.

2.9.1 SAE J1979

A SAE (*Society of Automotive Engineers*) publicou em 1991 uma norma que padroniza os códigos de diagnóstico básicos. Desde 1991 ela ja foi revisada 8 vezes, sendo a última revisão em 2012.

A norma divide os códigos em 10 modos, mostrados na Tabela 3. O modo 1, por exemplo,fornece dados em tempo real e possui mais de 100 códigos descritos, como para a rotação do motor, temperatura do líquido refrigerante, velocidade do veículo, entre outras. Os códigos para o módulo 1 estão contidos no Anexo A.

Tabela 3 – Modos descritos na SAE J1979

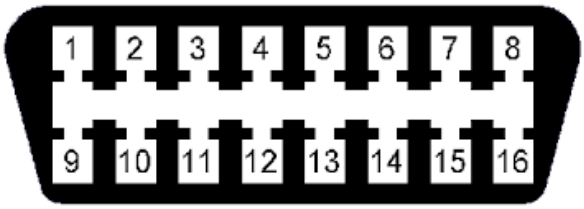
01	*Show current data*
02	*Show freeze frame data*
03	*Show stored Diagnostic Trouble Codes*
04	*Clear Diagnostic Trouble Codes and stored values*
05	*Test results, oxygen sensor monitoring (non CAN only)*
06	*Test results, other component/system monitoring*
07	*Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)*
08	*Control operation of on-board component/system*
09	*Request vehicle information*
0A	*Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)*

Fonte: (E/E. . . , 2012)

2.9.2 OBD-II

O conector OBD-II possui formato em D e conta com 16 pinos. Existem dois tipos de conectores, denominados tipo A e B. Esta diferenciação ocorre para separar sistemas que trabalhem em 12V e 24V respectivamente, e na construção física do conector, de forma a impossibilitar que um conector macho tipo A, de 12V, se conecte a um conector fêmea tipo B, de 24V. A Figura 17 mostra o formato construtivo do conector. Cada um dos 16 pinos possui função pré definida. Os pinos 4 e 5 são designados para o aterramento, enquanto o pino 16 é para a alimentação. Os demais pinos são para o uso de protocolos de rede padrões ou específicos de cada montadora. A Tabela 4 descreve a designação de cada pino do conector OBD-II

Figura 17 – Conector OBDII



Fonte: http://forums.pelicanparts.com/

Tabela 4 – Pinos do conector OBD-II

Pino	**Descrição**	**Pino**	**Descrição**
1	Opção do vendedor	9	Opção do vendedor
2	J1850 Bus +	10	j1850 BUS
3	Opção do vendedor	11	Opção do vendedor
4	Chassi Ground	12	Opção do vendedor
5	Signal Ground	13	Opção do vendedor
6	CAN (J-2234) High	14	CAN (J-2234) Low
7	ISO 9141-2 K-Line	15	ISO 9141-2 Low
8	Opção do vendedor	16	Bateria (VCC)

Fonte: adaptado de http://forums.pelicanparts.com/

2.9.3 Protocolos conhecidos

No inicio do desenvolvimento de ferramentas de diagnósticos para automóveis, cada grande montadora desenvolvia seu próprio método de comunicação com diversas variações inclusive dentro das montadoras. Com o aumento da complexidade do veículo, viu-se a necessidade de padronizar os protocolos de comunicação.

Isto ocorreu em primeira instância em esfera industrial, enquanto cada montadora trabalhava com seus próprios protocolos de comunicação. Com o tempo houve o advento

de uma padronização nacional, onde órgãos competentes nos EUA e Europa desenvolviam protocolos de referência. Destacam-se cinco padrões resumidos na Tabela 5,

Tabela 5 – Protocolos Padrão para Diagnóstico automotivo

Protocolo	Caracteristica	Especificação
SAE J1850 PWM	Pulse Width Modulation	Ford 2 fios, 32 nós até 35m @ 41,6 kbit/s
SAE J1850 VPW	Variable pulse width	Ford 1 fio, @ 10,4 kbit/s
ISO 9141-2	Serial K-Line/L-Line	Similar a RS 232 @ 10,4 kbit/s
ISO 14230 KWP2000	Keyword Protocol 2000	Fisicamente identica a ISO 9141-2
ISO 15765 CAN	CAN	Bosch @ 250/500 kbit/s

Fonte: Próprio Autor

2.10 Controller Area Network (CAN)

A rede CAN foi desenvolvida pela empresa Robert Bosch GmbH como um sistema multi-mestre de envio de mensagens em que as informações são transmitidas simultaneamente para todos os nós (CORRIGAN, 2008). Ela é utilizada extensamente em veículos automotores, em diversos domínios que diferem em seus requisitos (BOSCH, 2014) e utilizam-se variações desta rede, como redes CAN de alta velocidade (CAN C) para sistemas do *powertrain*, e de baixa velocidade (CAN B) para sistemas de conforto (BOSCH, 2014).

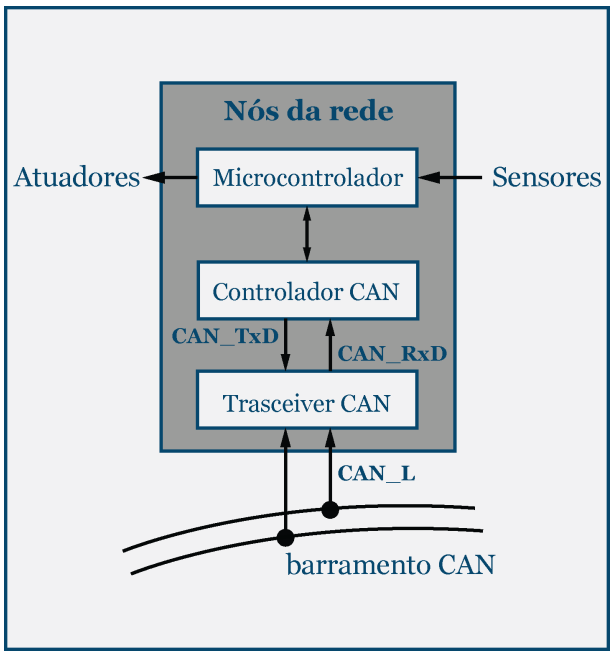
A CAN opera com velocidades que vão de 10kb/s a 1Mb/s e é normalizada como a ISO/DIS 11898 para aplicações de alta velocidade (500kbit/s) e ISO 11519-2 para aplicações de baixa velocidade (125kbit/s) (NATALE, 2008).

A rede CAN está se destacando principalmente devido a sua importância e versatilidade no diagnóstico automotivo. Com ela é possível conectar as ECUs diretamente à rede, e assim comunicar-se diretamente com elas dando consistência a transmissão e recebimento de mensagens (BOSCH, 2014). Ela se mostra como uma solução atrativa para controle de sistemas embarcados devido ao seu baixo custo, gerenciamento simples de protocolo, resolução determinística de disputas para detecção de erro e retransmissão (NATALE, 2008).

2.10.0.1 Topologia

A topologia mais comum encontrado na rede CAN é a barramento, que possibilita a expansão da rede e não acarreta em falha da rede se um nó falhar. A topologia estrela passiva também pode ser encontrada em alguns casos (BOSCH, 2014).

Figura 18 – Típico nó de uma rede CAN



Fonte: adaptado de (BOSCH, 2014)

2.10.1 Camada Física da rede CAN

A norma CAN não define outros aspectos relacionados a camada física do modelo ISO OSI, incluindo os tipos de cabos e conectores que podem ser utilizados na comunicação CAN, além de voltagens e correntes consideradas aceitáveis. Fica definido apenas *bit coding*, *timing* e sicronia de dados, que são relacionados na seção de *Physical Signaling* (PS) do modelo (NATALE, 2008).

2.10.1.1 Bit Timing e Sincronização

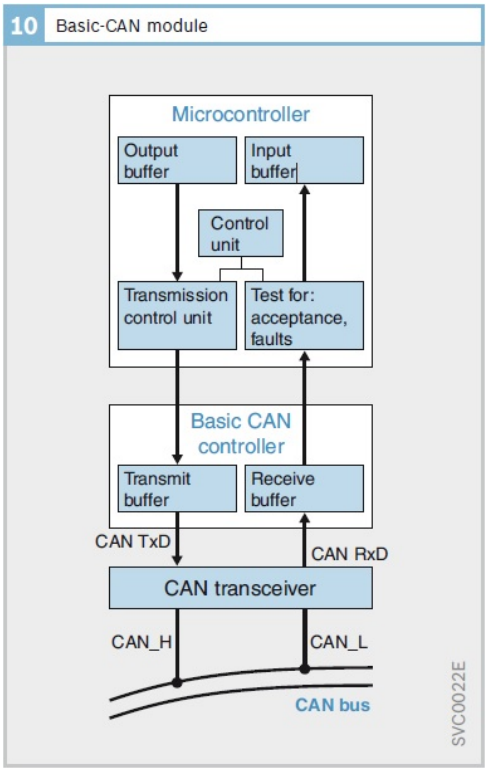
Cada nó CAN é cadenciado por um *clock*, normalmente um oscilador de quartzo (LAWRENZ, 2013).

2.10.1.2 Sistema de Transmissão de Dados

Um típico nó da rede CAN, mostrado na figura 18, consiste de um microcontrolador para processamento do software, um controlador CAN e um *transceiver* CAN.

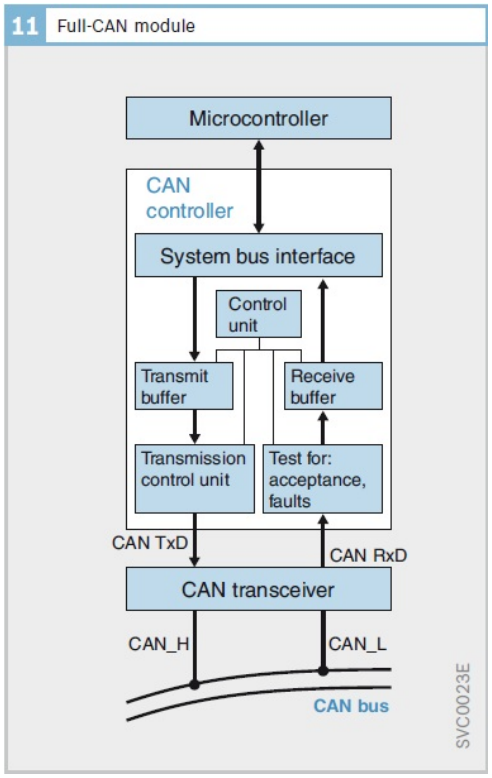
O controlador CAN é responsável pela transmissão e recebimento dos dados. Ele geralmente faz a conversão dos dados binários a serem transmitidos em *bit streams* e passa os dados para o *transceiver* pela linha TxD. Este por sua vez, amplifica os sinais, gera os níveis de voltagem adequados, e transmite em serial o *bit stream* na rede. O recebimento de dados é processado inicialmente pelo *transceiver* e enviado para o controlador CAN que traduz os dados para o microcontrolador (BOSCH, 2014).

Figura 19 – Controlador *Basic* CAN



Fonte: (BOSCH, 2014)

Figura 20 – Controlador *Full* CAN



Fonte: (BOSCH, 2014)

2.10.1.3 Controlador CAN

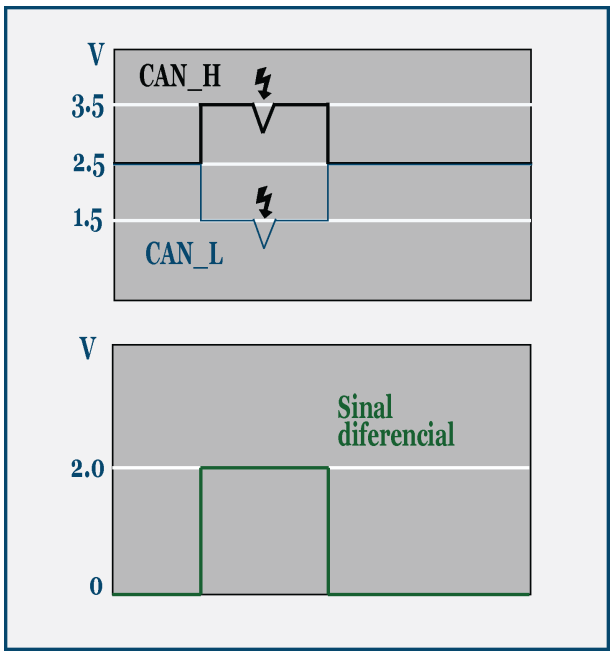
O Controlador CAN é dividido internamente em controlador de protocolo e manipulador de mensagens, sendo o primeiro definido pela especificação CAN e o segundo dependente da aplicação (NATALE, 2008).

Existem essencialmente três tipos de controladores CAN: o *Full CAN*, *Basic CAN* e e um conectado serialmente (*Serial Linked Input/Output* - SLIO) (NATALE, 2008).

No *Basic CAN*, mostrado na Figura 19, apenas a implementação básica do protocolo CAN para a geração do *bit stream* é implementado no *hardware*. Para o gerenciamento das mensagens a serem recebidas e enviadas existe um *buffer* intermediário ao qual o computador local tem acesso (BOSCH, 2014). Esse *buffer* é do tipo primeiro a entrar é o primeiro a sair (*First in, First Out (FIFO)*, que é reconfigurado a cada transmissão (LAWRENZ, 2013). Assim, o *buffer* deve ser checado regularmente e cada mensagem recebida deve ser comparada com a lista de IDs relevantes. Se esta comparação encontra mensagens relevantes, estas são transferidas para a memória RAM do controlador (LAWRENZ, 2013).

O *Full CAN* é a implementação preferencial no caso de um nó precisar gerenciar diversas mensagens a uma alta taxa e o computador local não tiver capacidade livre para tarefas de comunicação (BOSCH, 2014). Ele é caracterizado pelo fato de classificar,

Figura 21 – Filtando a interferência na rede CAN



Fonte: adaptado de (BOSCH, 2014)

baseado em filtros de aceitação, uma mensagem em um *buffer* na memória de mensagens do controlador. Isso significa que o *host* pode ler dados recebidos diretamente da RAM do controlador, e o *host* não é sobrecarregado com o processamento para filtrar as mensagens (LAWRENZ, 2013).

O *SLIO* é um tipo de controlador CAN composto de diversos pinos digitais e conversores digital-analógico e analógico-digital e necessita de software local.

2.10.1.4 Estados lógicos do Barramento - o trabalho do transceiver

O papel do *transceiver* é converter o *stream* de dados binários gerados pelo controlador em níveis de tensão nos terminais do CAN_H e CAN_L , assim como converter o estado de tensão em bits para o controlador.

O CAN, como outros sistemas utiliza dois estados para comunicação - Dominante e recessivo. O dominante é representado pelo binário 0, enquanto o recessivo pelo binário 1 (ROAD. . . , 2003).

Quando uma mensagem é recebida, o *transceiver* CAN converte o nível do sinal para os estados lógicos. Neste processo, o amplificador diferencial subtrai o nível do CAN_L do nível do CAN_H. Caso as linhas se torçam, pulsos de distúrbio possuirão o mesmo efeito em ambas as linhas, filtrando assim a interferência na linha , como mostrado na Figura 21.

Alguns *transceivers* podem também medir o nível de voltagem nas linhas separa-

damente, de forma a permitir que a operação continue em uma só linha caso uma das linhas falhe (curto circuito ou rompimento). Para isso as ECUs teriam que contar com um aterramento comum, que serveria como substituto da linha faltante BOSCH.

2.10.1.5 Transmissão

Usualmente utilizam-se dois fios trançados ou não, acoplados galvânicamente ou desacoplados (BOSCH, 2014). A linha dupla suporta transmissões simétricas e os bits são enviados em ambos os fios e representados por voltagens diferenciais. Isso reduz a sensibilidade para interferências *in-phase* porque a interferência afeta as duas linhas igualmente e pode ser filtrada (BOSCH, 2014).

A transmissão em um fio é possível, e consiste em uma medida de redução de custos. Para isso, todos os nós precisam de um aterramento comum, e assim, esse método é limitado a distâncias pequenas. Outra consequênia é que essa transmissão não pode ter as interferências filtradas e por isso utiliza um nível maior de voltagem, que por sua vez tem efeito negativo na radiação interferente. Isso resulta em uma menor taxa de transmissão, sendo este tipo possível apenas para eletrônica de conforto e conveniência (BOSCH, 2014). Desta forma, uma rede de baixa velocidade ainda se mantêm funcional caso um fio falhe.

2.10.1.6 Nível de Voltagem

O *transceiver* CAN converte os estados lógicos 0 e 1 em níveis de voltagem, que são enviados pelos fios CAN_H e CAN_L (BOSCH, 2014). Diferentes níveis de voltagem são utilizados para as redes de baixa e alta velocidades como mostrado na Figura 22 a e b, respecivamente.

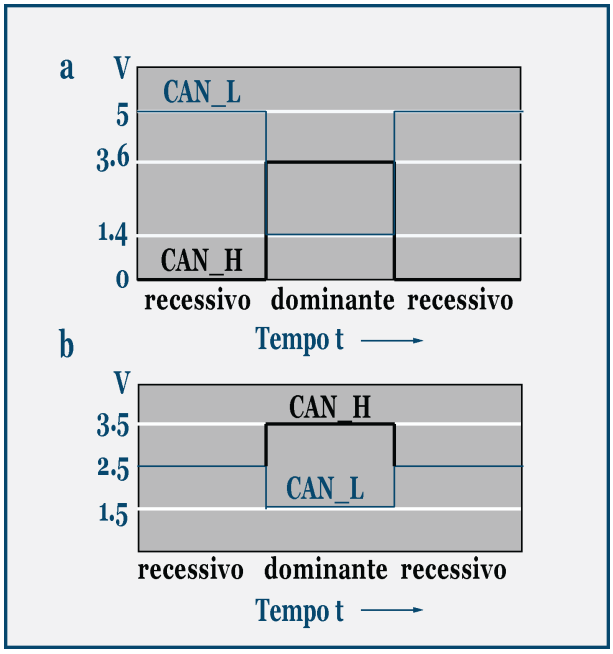
No estado recessivo o CAN C utiliza a voltagem de 2.5V em ambas a linhas. No estado dominante a voltagem de 3.5V para o CAN_H e 1.5V para o CAN_L (BOSCH, 2014). Para redes de baixa velocidade no estado recessivo, CAN_H apresenta uma voltagem de 5V enquanto CAN_L 0V. No estado dominante CAN_L e CAN_H possuem respectivamente 3.6V e 1.4V (BOSCH, 2014).

A norma ISO 11898-2 diz que a voltagem padrão varia de -2 V no CAN_L a +7 V no CAN_H (NATALE, 2008).

2.10.1.7 Ausência de Reflexão na terminação

Segundo a norma (ROAD..., 2003) deve-se utilizar um resistor de valor nominal 120 ohm, e segundo NATALE os resistores de terminação de um cabo devem corresponder a impedância do cabo. Assim, para atenuar a reflexão de sinais elétricos em terminações abertas que seriam prejudiciais a comunicação, a rede é composta de resistores de termina-

Figura 22 – Níveis de Voltagem



Fonte: adaptado de (BOSCH, 2014)

ção de 120 ohm em ambas as extremidades (LAWRENZ, 2013). Novas aplicações incluem também um capacitor de 4.7 nF ou maior que aumenta a estabilidade do nível recessivo.

2.10.1.8 Limites de transmissão

A ISO 11898 especifica uma velocidade para o comprimento de um circuito. Especifica-se 1 MBits/s para linhas de 40m e recomenda-se 500 kBits/s para até 100m (BOSCH, 2014). A norma ainda cita o *delay* normal de propagação de 5ns/m (NATALE, 2008).

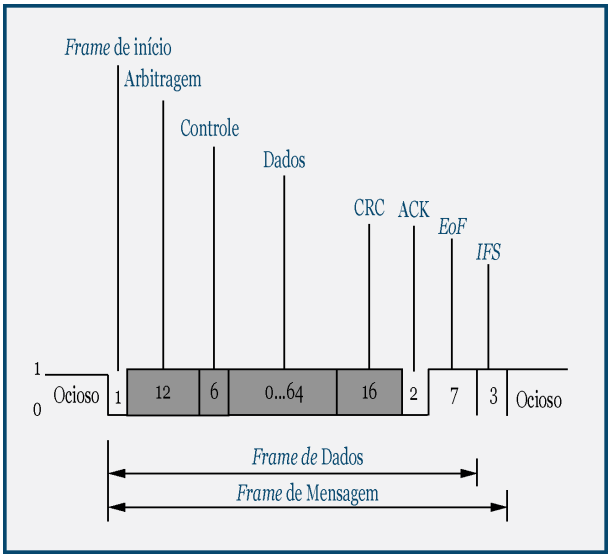
Para aplicações de alta velocidade o maior limitante para o comprimento do barramento é o *delay* de propagação do *transceiver*, de 120 ns a 250 ns, sendo o do controlador CAN de 50 ns a 62 ns e do acoplador óptico de 40 ns a 140 ns (NATALE, 2008).

É possivel conectar até 30 nós em um barramento, sem a necessidade de medidas adicionais (BOSCH, 2014).

2.10.2 Camada de Comunicação

Existem quatro formatos de *frame*: *Data frame*, com os dados a serem enviados, *Remote frame* com as requisições a serem enviadas, *Error frame* que comunicam que erros ocorreram e *Overload frame* que indica que o transmissor não está conseguindo processar mais mensagens no momento (BOSCH, 2014).

Figura 23 – Formato da mensagem CAN



Fonte: (BOSCH, 2014)

2.10.2.1 Data Frame

Eles são utilizados para transmitir informações entre a fonte e um ou mais receptores (NATALE, 2008). É o único *frame* que transporta dados (LAWRENZ, 2013), e contém os bits de início, arbitragem, controle, dados, CRC, ACK e finais, como mostrado na Figura 23.

Ele possui 1 bit dominante para o início da mensagem, 12 bits para a arbitragem (Formato normal) que contém o identificador e o bit de requerimento de transmissão remota (RTR), que especifica se a mensagem contém dados (dominante) ou não (recessivo) (NATALE, 2008). Desta forma, o *Data frame* sempre possui prioridade sobre outras mensagens de mesmo identificador (BOSCH, 2014).

Em seguida tem-se os 6 bits de controle. O primeiro é 0 (dominante) caso o identificador seja de 11-bits, indicando que o identificador terminou. O segundo bit é reservado. Por fim os quatro bits posteriores mostram o comprimento dos dados da mensagem, chamado *Data length Code* ou DLC (NATALE, 2008). O DLC permite o receptor saber se recebeu de fato todos os dados enviados (BOSCH, 2014).

Segue-se então até 8 bytes (64bits) de dados. O campo CRC (*Cyclic Redundancy Checksum)* contém 15 bits para o *checksum* dos bits anteriores e funciona como detector de erros, e não para correção de erros, e 1 bit final para o delimitador do CRC (recessivo) (NATALE, 2008). Interferências podem ser detectadas no check-sum (BOSCH, 2014).

Em seguida tem-se 2 bits de confirmação (*acknowledge*), o ACK. O transmissor envia um nível recessivo no primeiro bit, e os receptores devem sobrescrever com um nível dominante caso eles tenham encontrado uma mensagem sintaticamente correta. O

transmissor reconhece como um erro de confirmação caso a mensagem possua um nível recessivo. É importante ressaltar que a detecção de um nível dominante não significa que todos os nós receberam a mensagem, mas que pelo menos um deles reconheceu a mensagem como um *frame* CAN correto. O segundo bit do ACK é recessivo e delimita o final dele (NATALE, 2008).

Os 7 bits posteriores são o fim do *frame* (*End of Frame* ou EOF) e indica o fim dos dados. Por fim tem-se 3 bits recessivos de espaçamento chamado de *Inter Frame Spaces* ou IFS, que separa um *frame* do outro (NATALE, 2008).

No formato extendido o que muda é o campo de arbitragem que passa a ter 32 bits (ao invés de 12 bits no formato normal). Existem 29 bits de identificador consistentes de três partes. Inicia com os 11 bits mais significativos (indentificador base) seguidos de dois bits recessivos chamados *Substitute Remote Request* (SRR) e *Identifier Extension Flag* (IDE) (NATALE, 2008). É valido perceber que o bits recessivos enviados pelo SRR e IDE fazem com que o identificador de 11-bits tenha sempre prioridade sobre o de 29-bits (BOSCH, 2014).

Segue-se os 18 bits menos significativos (extensão do identificador). Os valores concatenados do identificador base e da extensão do identificador representam o endereçamento lógico e, por extensão, a prioridade da mensagem. O ultimo bit é o *Remote Transmission Request* (RTR), que distingue *frames* de dados de *frames* remotos e é seguido de 6 bits de controle que começam com dois bits reservados e quatro bits do comprimento dos dados da mensagem DLC (NATALE, 2008).

2.10.2.2 Remote Frame

Com este *frame*, estações podem solicitar dados na rede, como por exemplo o módulo de controle do limpador de vidros solicitar a grau de umidade para o sensor de chuva (BOSCH, 2014). o DLC deve ser exatamente o mesmo do *data frame* que se quer utilizar (LAWRENZ, 2013).

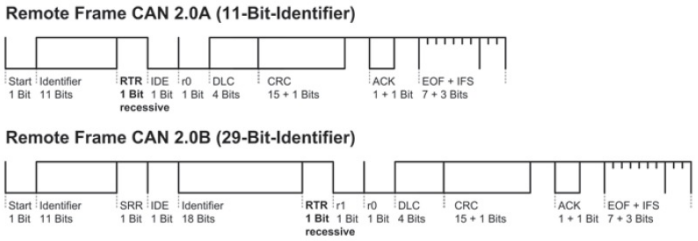
A estrutura do *remote frame* é idêntica para endereçamento normal e estendido, exceto pelo campo da arbitragem (LAWRENZ, 2013). Esta estrutura está exemplificada na Figura 24.

2.10.2.3 Error Frame

O *bit-stuffing* é utilizado para detectar erros na linha como curto-circuitos, permitindo também a sincronia dos clientes da rede (BOSCH, 2014).

A convenção adotada de *bit stuffing* estipula que cada *Data Frame* ou *Remote Frame* pode ter no máximo cinco bits de mesmo estado em sequência entre o início do

Figura 24 – *Remote Frame*



Fonte: (LAWRENZ, 2013)

frame e o final do CRC. Após cinco bits de mesmo estado o remetente envia um bit de estado oposto(BOSCH, 2014).

A rede pode sofrer interferências como, por exemplo, eletromagnéticas. Para evitar o risco de erros existem diversos mecanismos de controle de erros implementados no protocolo (BOSCH, 2014). Um nó pode detectar erros no CRC, no ACK e na estrutura do *frame*. O remetente da mensagem também checa continuamente o nível da rede para ver se ele corresponde com a mensagem sendo transmitida, comparando bit a bit (BOSCH, 2014).

Quando um nó detecta um erro, ele comunica isso através de um *error frame* (BOSCH, 2014). Para isso o chamado *Active Error Flag* sobrescreve-se o barramento com 6 bits dominantes consecutivos, que viola as regras de *bit-stuffing* e é reconhecido por todos os nós na rede (LAWRENZ, 2013).

Para evitar um distúrbio local de um nó ou grupo de nós de paralisar a rede permanentemente com um *Active Error Flag*, os nós afetados, de acordo com um algoritmo específico, gradualmente se afastam da atividade no barramento. Em um primeiro momento o nó continua podendo comunicar na rede e pode enviar os chamados *Passive Error Flags*, que consistem em enviar uma sequencia de bits recessivos (LAWRENZ, 2013). Estes processos estão descrito na Figura 25.

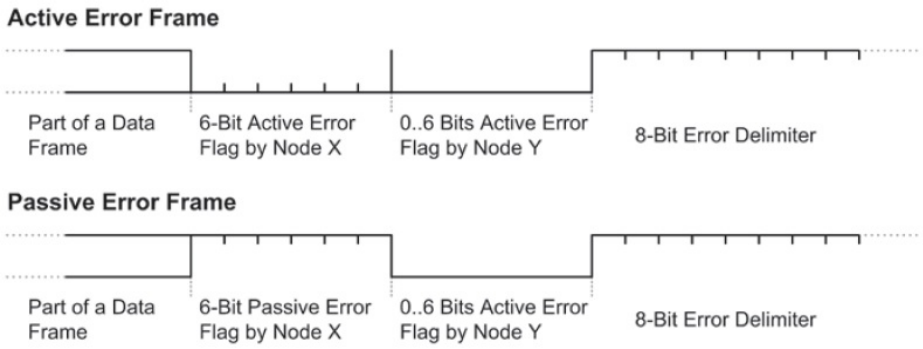
Por fim, existe o estado de *Bus Off* onde o nó é completamente desligado da rede, e só pode ser revertido com um reset em software ou hardware (LAWRENZ, 2013).

Cada CI CAN tem contadores de erros para os erros recebidos e transmitidos. Se a contagem de ambos for igual ou inferior a 127, ele se mantém em estado de erro ativo. Caso ela exceda 127 mas seja menor que 256, entra-se no estado de erro passivo. Contagens maiores do que 255 geram o desligamento do nó da rede (LAWRENZ, 2013).

2.10.2.4 Overload Frame

Caso um nó não consiga processar outro *frame* no momento, ele envia um *overload frame* criando um *delay* entre o *data frame*s ou *remote frames*.

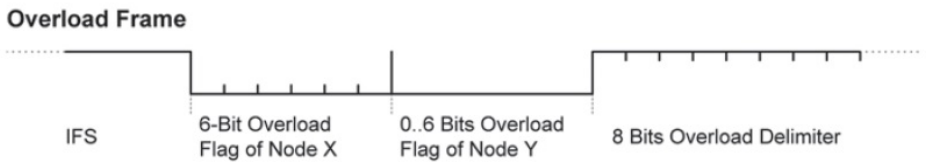
Figura 25 – *Error Frames*



Fonte: (LAWRENZ, 2013)

A estrutura do *Overload Frame* é exatamente a mesma do *Error Frame*, com a única diferença que ele não sobrescreve dados, sendo enviado apenas no IFS (LAWRENZ, 2013). O *Overload Frame* é enviado como um ou dois *bits* dominantes no IFS, e não tem efeito na contagem de erros (LAWRENZ, 2013). A estrutura está exemplificada na Figura 26.

Figura 26 – Overload Frames



Fonte: (LAWRENZ, 2013)

2.10.3 Normas

A *ISO 11898-2 (Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling*) especifica as características de *setup* de um intercâmbio de informações digitais através dos módulos implementando a camada *data link* CAN (ISO, 2015). A *ISO 11898-2 Road vehicles – Controller area network (CAN) – Part 2: High-speed medium access unit* especifica a unidade de acesso ao meio a alta velocidade que compõe a camada física da rede CAN (ROAD. . . , 2003).

A ISO 15765 é a norma internacional para diagnóstico via CAN (BOSCH, 2014) e foi escolhida para um estudo mais aprofundado devido a forte tendência mundial em sua utilização, proveniente de sua maior capacidade e flexibilidade. O protocolo é mandatório para todos os veículos produzidos e vendidos nos EUA desde 2008 devida a regulação federal 40 C.F.R. § 86.1806-05 (CODE. . . , 2013). Quando mapeado os serviços especificados na

ISO 15765, sobre o modelo ISO OSI, dividiu-os em três camadas: a camada 7 chamada de serviços de diagnóstico unificados, contemplado na ISO 15765-3; a camada 3 chamada de serviços da camada de rede, contempladas na camada ISO 15765-2; e as camadas 1 e 2 juntas chamadas de serviços CAN, e contempladas na ISO 11898 (ROAD..., 2004).

2.10.3.1 ISO 15765-2

A ISO 15765-2 foi criada para definir requisitos comuns para sistemas de diagnóstico automotivo implementados em uma rede CAN como especificado na norma ISO 11898 (ROAD..., 2004).

O protocolo CAN não requer um controlador central. Qualquer nó pode tentar enviar mensagens a qualquer momento, e o sucesso da transmissão depende apenas se o barramento está livre e se a fase de arbitragem foi feita com sucesso (BOSCH, 2014). Este comportamento é característico de um método de acesso multimestre múltiplo acesso com controle de colisão e arbitragem na prioridade de mensagens (do inglês *Carrier Sense Multiple Access with Collision Detection and Arbitration on Message Priority* ou CSMA/CD+AMP) (LAWRENZ, 2013).

2.10.3.2 Identificador

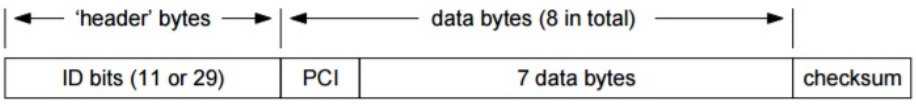
A diferença mais importante entre o CAN 2.0 A e o CAN 2.0 B está no tamanho do identificador, de 11 bit e 29 bit respectivamente, sendo que os dois são compatíveis entre si e podem ser utilizados na mesma rede. CAN 2.0 A sempre tem prioridade sobre CAN 2.0 B (BOSCH, 2014). Existem identificadores de 11 bits e 29 bits, chamados de formato *standard* e formato estendido. O formato *standard* pode distinguir 2048 identificadores, enquanto o formato estendido permite mais de 536 milhões de identificadores únicos.

2.10.4 Envio de Mensagens Maiores que 8 bytes

O protocolo permite a transmissão de mensagens que excedam o máximo de 8 bytes do CAN através do uso de mensagens consecutivas. Mensagens com até 7 bytes (6 para endereçamento extendido) são enviadas como *Single Frame* (SF) enquanto mensagens maiores são enviadas como *First Frame* (FF), *Consecutive Frames* (CF) e *Flow Control Frame* (FCF) (ROAD..., 2004). A estrutura da mensagem *Single Frame* segue o formato ilustrado na Figura 27.

Os bytes do *header* representam o endereçamento. O PCI (*protocol control information*) é composto de um, dois ou três bytes. O campo inicial mostra o tipo de *frame*, e implicitamente descreve o comprimento do PCI, para distinguir se a mensagem é *Single Frame* ou *Consecutive Frame*.

Figura 27 – Estrutura de uma Mensagem CAN



Fonte: (ELM327, 2016)

Mensagens *Single Frame* são enviadas com o byte inicial (PCI) contendo o dominante (0) que especifica sua condição de *Single Frame*. Mensagens maiores que 7 bytes são segmentada em múltiplos *frames*. O primeiro segmento é o *First Frame*, com o PCI de dois bytes onde os primeiros 4 bit representam o tipo e os 12 consecutivos, o comprimento da mensagem (ROAD. . . , 2004).

O receptor confirma o recebimento com um *Flow Control Frame*, que contem PCI de três bytes especificando o intervalo de recebimento de cada pacote e quantos *Consecutive Frames* podem ser enviados antes de esperar a próxima mensagem de *Flow Control*. O resto da mensagem é transmitida utilizando *Consecutive Frames* como mostrado na Figura 28, onde cada um possui um byte PCI indicando que é *Consecutive Frame*, seguido de uma sequencia de números de 4-bit, que indica qual a ordem da mensagem. Por padrão *First Frame* ocupa a posição numero 0 enquanto as *Consecutive Frames* seguintes serão incrementadas de 1, a cada mensagem, zerando novamente uma vez que chega a 15 (1, 2, 3...15, 0, 1) (ROAD. . . , 2004). Na teoria (FF) permite o envio de 4095 bytes em uma mensagem segmentada de comprimento 12-bit, porém na pratica limitações de hardware diminuem este número.

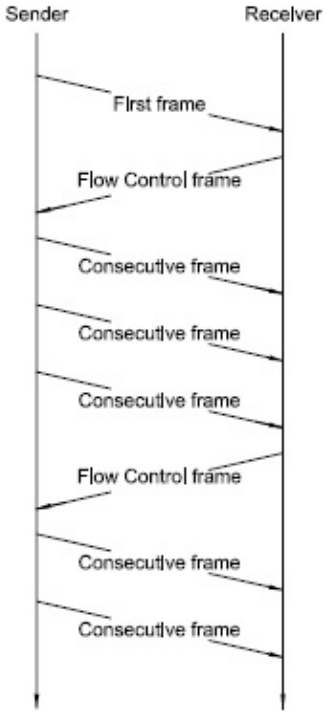
2.10.5 Endereçamento

Diferente de outras redes, a CAN não endereça os nós individuais mas sim as mensagens, utilizando o método orientado a mensagem. Cada mensagem possui um identificador ou marca única. Esta classifica o conteúdo da mensagem (BOSCH, 2014).

Assim, um nó pode realizar um *broadcast* uma mensagem para todas as outras estações, que lêem apenas as mensagens que estão em sua lista de aceitação. Cada indivíduo na rede decide por si só se tem interesse na mensagem ou não, o que permite a total independência entre a operação deles (BOSCH, 2014).

Sendo assim, um novo nó pode ser adicionado a qualquer momento sem a necessidade de alteração da rede já existente, e a falha de um nó não acarreta, neste aspecto, uma falha na rede.

Figura 28 – Envio de Mensagens Segmentadas



Fonte: (ELM327, 2016)

2.10.5.1 Arbitragem

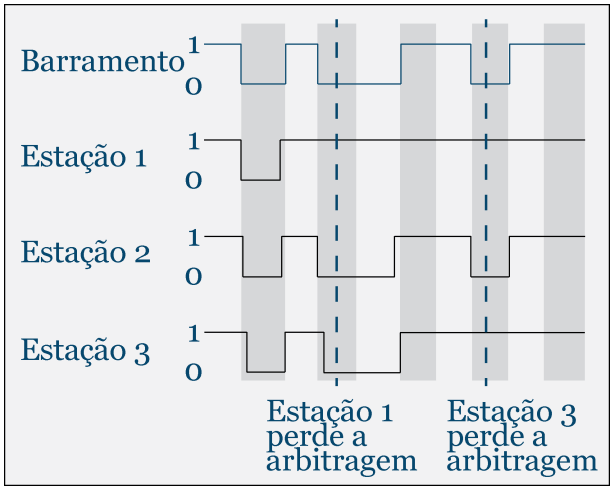
Toda mensagem CAN é iniciada por um bit dominante 0 (*start-of-frame-bit*) seguido do pelo identificador. O restante da mensagem é enviado bit a bit de forma serial. Ao longo do *frame* cada nó irá, através de seu *transceiver*, ler o estado lógico do barramento e comparar com o seu naquele instante (LAWRENZ, 2013), caracterizando isso como uma detecção de colisão. Se o estado presente na rede for diferente do enviado, ele encerrará sua transmissão e reiniciará o envio da mensagem na próxima vez que o barramento estiver livre (*idle*).

Para isso o protocolo ISO CAN possui arbitragem (*wired-and arbitration*) que da preferência da transmissão para mensagem para o nó com o ID de valor binário mais baixo, permitindo que um bit dominante enviado por ele se sobreponha a um recessivo. Para que isso seja possível, não é permitido que dois nós transmitam mensagens de identificador igual (BOSCH, 2014).

Arbitragem se utiliza assim do operador lógico AND, que resulta em um valor dominante 0, no caso pelo menos um nó ter enviado um valor 0.

No exemplo da imagem 29 as três estações transmitem simultaneamente o bit dominante zero para iniciarem suas transmissões. Em seguida elas iniciam a transmissão de seus identificadores. O segundo bit enviado pela estação 1 é o recessivo 1 porém

Figura 29 – Arbitragem CAN



Fonte: (BOSCH, 2014)

o encontrado no barramento naquele momento é o dominante 0, e neste momento ela encerra sua transmissão e se torna receptora das mensagens. As estações 2 e 3 continuam transmitindo simultaneamente os mesmos bits até que a estação 3 transmite a o bit recessivo 1 enquanto a rede apresenta um estado dominante 0. Neste momento a estação 3 também se torna receptora, e posterga o envio de sua mensagem para a próxima vez que o barramento estiver livre.

O tempo de espera de mensagens de prioridade máximo a uma taxa de 500 kbits/s é de 260 um para CAN 2.0 A e de 300 um para CAN 2.0 B (BOSCH, 2014), porem quanto maior o uso da banda da rede, mais incerto é o tempo de recepção de mensagens com prioridade baixa.

2.10.6 Variações da rede CAN

2.10.6.1 TT-CAN

O protocolo de comunicação CAN implementa um acesso ao barramento puramente baseado na prioridade, que por si só não garante nenhum tempo específico de latência. Sistemas de segurança veiculares demandam um solução que garanta o determinismo, e por isso criou-se a ISO 11898-4 que especifica a implementação da *Time-Triggered CAN* (LAWRENZ, 2013).

O princípio básico desta abordagem consiste no uso de um relógio sincronizado em todos os nós participantes da comunicação, seja através de um relógio global ou pela transmissão do relógio de cada um na rede. Existem janelas cíclicas no tempo onde um nó deve transmitir,ou seja, *slots* de tempo onde qualquer nó pode transmitir, seguindo a lógica da arbitragem, e ainda *slots* de tempo livres para futuras expansões (LAWRENZ, 2013).

2.10.7 CAN FD

CAN com taxa de dados flexível é um novo protocolo que combina os recursos principais da CAN com taxas de transferências mais altas, que pode chegar a 2.5 MBit/s. A migração para este tipo de sistema é simples pois o software e o hardware da camada física podem permanecer idênticos (LAWRENZ, 2013).

O CAN FD aumenta a velocidade ao permitir sequencias maiores de dados a serem transmitidas aumentando a relação entre o *header* e o *payload*, e diminuindo o *bit time* (LAWRENZ, 2013).

A ISO 15765-2 passou a especificar em 2016 a implementação de CAN FD (ROAD..., 2016).

3 . TRABALHOS RELACIONADOS

O Trabalho de KESKIN (2009) apresenta uma pesquisa bibliográfica em redes de comunicação intra-veiculares. O relatório faz uma revisão de trabalhos até 2009 e compara protocolos e tendências futuras.

É enfatizado, ao longo do texto, que os veículos possuem uma eletrônica embarcada crescente para fazer frente as demandas de performance, custo, *time to market* e confiabilidade.

A rede é dividade em cinco domínios: o do *powertrain*, *chassi*, *body*, *telematics/wireless* e segurança passiva (*passive safety*). Segundo o autor, o sistema do *powertrain* possui grande complexidade computacional, e necessidades de trocas de dados *real-time* com tempos de resposta muito baixos. Por este motivo a rede que possibilita comunicação do *powertrain* possuí uma demanda para um grande banda de rede.

O chassi possui funções que também demanda funções de comunicação em tempo real, dinâmica de direção e assistência com sistemas como o ABS, ESP (*Eletronic Stability Control*), ASC (*Automatic Stability Control*), ACC (*Adaptive Cruise Control*), ASR (*Anti Slip Regulation*), EPS (*Eletronic Power Steering*), 4WD (*Four Wheel Drive*, EDC (*Eletronic Damper Control*) e suspensões ativas. Neste sistema se incluem os sistemas eletrônicos do futuro que substituem os antigos mecânicos eliminando elementos mecânicos, como direções elétrica que eliminam a coluna da direção, chamados *x-by-wire* (no caso da direção de *steer-by-wire*).

O domínio do *Body* é segundo KESKIN (2009) , o que possui maior número de ECUs. Ele não é crítico para a segurança do veículo, sendo representado pelos sistemas de vidros, portas, limpa vidros, luzes, painel entre outros. Ele não precisa de uma alta banda, e por isso é constituído de redes de baixo custo.

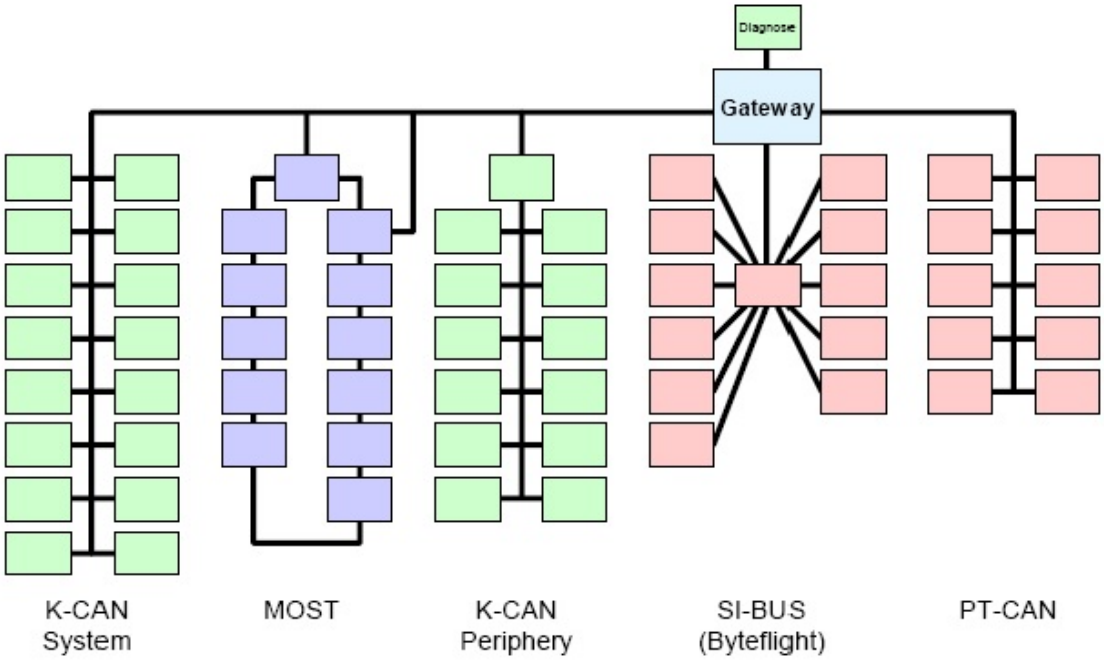
O domínio do *telematics* consiste de multimedia e *infotainment* e sistemas *wireless*. Devido a alta conectividade deste domínio é comum que quantidades significativas de dados sejam trocadas na rede e com o exterior do veículo, exigindo uma banda maior.

O domínio de segurança passiva contem os *airbags*, pré-tensionadores de cintos de segurança e sensores de batida e capotamento. Como estes são essenciais para a segurança, eles precisam ser extremamente confiáveis.

A arquitetura das redes também é exemplificada pelo autor. A Figura 30 mostra a arquitetura da rede de uma BMW série 7. Percebe-se que há um *gateway* logo após o conector de diagnóstico, que liga diretamente a três redes, e indiretamente a mais duas através de ECUs que funcionam como *gateways* auxiliares.

O autor, em seguida, discute os paradigmas das classificações de comunicações intra veiculares, como *time-triggered* e *event-triggered*.

Figura 30 – Arquiterura de rede da BMW série 7



Fonte: (KESKIN, 2009)

É discutido que muitas tarefas dependem de valores de sensores e da comunicação destes na rede, e garantir que as mensagens seja entregues a tempo é o desafio. Na comunicação *event-triggered*, a transmissão de mensagens é baseada em eventos significativos e mensagens asíncronas são transmitidas assim que possível. O autor ressalta que a maioria dos protocolos que funcionam desta maneira são baseados no método CSMA/CR (Carrier Sense Multiple Access/Collision Resolution), e isso torna elas mais fácil de ser estendida e mais flexível em termos de tempo de resposta e uso de banda. A rede CAN é um exemplo que utiliza esse método.

Na *time-triggered*, a comunicação é definida no tempo e baseada no método TDMA (*time division multiple access*). Esse modelo determinístico é mais previsível e erros podem ser facilmente encontrados (falta de uma mensagem), porém faz com que o sistema seja mais difícil de ser modificado, sendo necessário reorganizar todo o sistema. Outra desvantagem é a necessidade de alta sincronia temporal entre os indivíduos da rede. Esse tipo de abordagem é proposta para sistemas *x-by-wire* que precisam ser altamente confiáveis

Em resumo, o método baseado no tempo possui as vantagens de impactar menos no tempo de recebimento de mensagens em caso de uma mudança na rede, e de não precisar de arbitragem e controle de colisão, sendo muito mais robusta e confiável. Porém, a desvantagem de se ter que refazer todo o agendamento ao se introduzir novas mensagens

ou ECUs, mostra uma baixa flexibilidade e por isso seu uso não é tão comum.

É citado também o grande uso de redes mistas. Redes TT-CAN (*Time-triggered CAN* e byteflight são exemplos que usam as qualidades de ambos os métodos para se ter uma rede mais confiável e mesmo assim com certa flexibilidade para comportar futuras mudanças.

KESKIN (2009) conclui que o desempenho destes métodos depende do uso da banda e que para baixos e médios usos de banda, o método *event-triggered* funciona muito bem.

O autor compara os diferentes protocolos de comunicação e seus usos, e cita que o modelo híbrido FTT-CAN (*Flexible Time-Triggered CAN* utiliza um nó mestre como o da rede LIN para o agendamento de mensagens *time-triggered* enquanto reserva largura de banda para mensagens *event-triggered*, mostrando o uso de mecanismos FTDMA (*Flexible TDMA*) e CSMA/CR juntos.

Em seguida, o autor resume os métodos utilizados pelas diferentes redes como LIN e CAN, já discutidas, assim como formato de mensagens. Ele cita o uso de redes LIN como baixo custo para o domínio *Body* e o fato da rede can ser o protocolo *de-facto* mais utilizado para transmissão de dados intraveiculares. Ele descreve o protocolo CAN mostrando o formato das mensagens CAN, arbitragem e a performance da rede baseado no método de agendamento de mensagens utilizado. Por fim ele pauta que a rede CAN possui grande flexibilidade e confiabilidade, mas possui também desvantagens como o potencial atraso de mensagens devido a repetição de mensagens enviadas com erro, e o potencial de um nó de alta prioridade ficar enviando mensagens sem parar na rede, dominando a rede.

O autor então discute a *Byteflight*, e seu uso em sistema de seguranças com um protocolo FTDMA, para então abordar os protocolos TTP/C, TTCAN, *FlexRay* e MOST.

Por fim é discutida brevemente o conceito de criação de mais uma camada entre a camada física e a camada de aplicação, chamada *Middle Layer*, que seria uma espécie de sistema operacional para ECUs com diversas vantagens como a diminuição do tempo de desenvolvimento, a melhoria da qualidade de comunicação, e padronização dos serviços de comunicação independente do protocolo e localização dos nós.

Os autores WOLF; WEIMERSKIRCH; PAAR (2004) discutem duas arquiteturas de rede existentes nos veículos e aspectos técnicos delas, expondo elas em seguida. Eles indicam possíveis ataques e ataques factíveis para cada uma das arquiteturas. Por fim eles mostram formas elementares de aumentar a segurança dos veículos.

Os diferentes tipos de redes são separados em quatro grupos e ataques a representantes de cada um dos grupos é discutido, assim como as potenciais consequências destes. A rede LIN por exemplo pode ser acessada o espelho retrovisor de carros de luxo, e um ataque pode potencialmente desligar a rede toda com um simples *frame sleep*. A rede

CAN pode ser inutilizada ao se enviar repetidamente mensagens de alta prioridade, ou qualquer ECU pode ser individualmente desabilitada ao se enviar mensagens *error flag*.

É enfatizado que as redes se comunicam livremente entre si, através de *gateways*, e desta forma a exposição de uma rede, expõe todas.

Como soluções, se propõe o uso de autenticações para todos os indivíduos da rede, mensagens criptografadas e *firewal*l. Ressalta-se que isso seria oneroso para as montadoras, pois a implementação seria complexa, enquanto o valor agregado não seria atrativo para os consumidores. Conclui-se que esforços devem ser feitos para garantir a segurança na comunicação automotiva.

DAVIS et al. (2007) realizam uma análise de um método de análise de agendamento CAN de 1994 amplamente utilizado para o desenvolvimento destas redes na indústria e citado mais de 200 vezes. É mostrado que o método é falho e uma abordagem aprimorada é apresentada, assim como uma nova política de atribuição. É discutido, por fim, o possível impacto em sistemas CAN comerciais desenvolvidos com a metodologia falha, e são feitas recomendações para a revisão das ferramentas de análise de agendamento CAN.

PRETSCHNER et al. (2007) ilustra os desafios de desenvolvimento de *software* para automóveis. Segundo os autores o software começou a chegar nos veículos em 1976 e em 2010 era previsto que os carros de luxos contivessem um gigabyte de software embarcado, e que ele pode ajudar a diferenciar os carros. O valor de mercado da eletrônica automotiva era projetado para 316 bilhões em 2015. Inicia-se a obra descrevendo o *mix* de características que definem os softwares automotivos e suas consequências. O trabalho é desenvolvido explorando metodologias de desenvolvimento de software para veiculos, ressaltando problemas e desafios que se relacionam com evolução e integração.

JUNIOR (2012) faz um estudo dos protocolos de comunicação utilizados em arquiteturas eletroeletrônicas automotivas separando as redes em classes A, B, C, D, Entretenimento e *wireless* citando as principais características de cada rede. O autor conclui citando as possibilidades futuras paras as redes de comunicação automotivas, apontando para redes de maiores taxas de transmissão para atender veículos mais conectados.

KRAUS et al. (2016) analisa a substituição da rede CAN por uma rede óptica que permite maiores bandas de rede, segurança, e resistência à interferência eletromagnética. Segundo os autores, os veículos do futuro precisarão se conectar com o ambiente, como prédios, placas, semáforos e outros carros e para isso a rede CAN é insuficiente. Os autores mostram os motivos por os quais eles julgam a rede como falha, e as vantagens em redução de peso e economia de espaço com o uso de uma rede óptica de um fio.

MITTAL; SMART (2015) fazem uma análise de optimização, predição de saúde, e diagnóstico *offline* da rede CAN. RMA (*Rate Monotonic Analysis*) é utilizado na análise e se mostra uma ferramenta muito útil no entendimento de como o sistema vai se comportar

na vida real. Resultados mostram que este simples procedimento encontrou mensagens desconhecidas de log, sobras de mensagens de test, mensagens proprietarias de outros softwares, entre outros.

ZHOU; LUO (2016) desenvolve um protocolo baseade em *CANopen* sem a característica Mestre-escravo. Uma simulação utilizando o software *CANoe* verifica o protocolo.

4 . ANÁLISE COMPARATIVA E RESULTADOS

Buscou-se um método de acessar a rede automotiva e colher dados em tempo real. Primeiramente procurou-se, no mercado, dispositivos capazes de ler a rede CAN e comunicar estes dados com um computador. Analisou-se os dispositivos em conjunto com diferentes *softwares* e suas capacidades, e comparou-se seus desempenhos. Realizou-se testes de interação com o veículo, isolando códigos da posição do pedal do acelerador e realizando a solicitação do número do chassi do veículo.

Em seguida, construiu-se um dispositivo pela a união de dois dispositivos comerciais, a fim de poder isolar a origem e o destino de mensagens de origem externa ao veículo que fossem enviadas na rede.

Utilizou-se o equipamento para isolar os códigos de rotação do motor e de temperatura do líquido refrigerante. Por fim analisou-se estes códigos e traduziu-os para valores físicos em rpm e °C.

4.0.1 Metodologia

O desenvolvimento do projeto seguirá uma metodologia que segue o paradigma de engenharia de software denominado prototipagem, baseado na produção de protótipos que evoluem rumo ao produto final (paradigma evolutivo) (Pressman, 2015). Tanto o hardware quanto o software serão desenvolvidos em conjunto, permitindo constantes refinamentos do protótipo. Um aspecto importante desta abordagem é que já no inicio do projeto um protótipo funcional é construído, porém com funções limitadas. O próprio protótipo ajuda a entender desafios, problemas e planejar os próximos passos.

4.1 ELM 327

Em uma pesquisa inicial encontrou-se a popular ferramenta ELM 327. Trata-se de um *microchip* criado e fabricado pela canadense *ELM Eletronics* que, segundo a própria empresa, é desenvolvido para atuar como uma ponte entre o OBD e a interface serial RS232. Ele pode interpretar nove protocolos OBD e é altamente customizável para a fabricação de produtos relacionados a diagnósticos automotivos. Desta forma, para se ter acesso a rede através da porta OBD procurou-se primeiramente soluções prontas. Algumas peças comumente encontradas no mercado e denominadas ELM 327 com variações *Bluetooth*, WIFI e USB foram adquiridas e estudadas. As peças foram adquiridas tanto em mercados nacionais através de portais de compras como mercado livre, assim como nos estados unidos e no mercado europeu.

Inicialmente utilizou-se o modelo USB pois considerou-se que esta interface seria mais bem caracterizada de forma a mitigar fontes de erros, além da alimentação do

dispositivo ter fonte na conexão USB e não na porta OBD, o que permitiria testes sem a necessidade do veículo.

4.1.1 Códigos do ELM 327

A comunicação com o ELM 327, ocorre através dos comandos AT. Os códigos AT do ELM seguem alguns padrões e estão documentados em seu *datasheet*. Normalmente são composto por letras que representam suas funções e ocasionalmente os numerais 0 e 1 representando os estados desligado e ligado, respectivamente. Alguns comandos podem incluir um HEX variável como parâmetro configurável. A Tabela 6 ressalta alguns dos códigos e suas funções.

Tabela 6 – Códigos do ELM327

Comando	Descrição	
ATZ	Reiniciar o ELM 327	Reinicia o ELM 327, pisca os led, e imprime a versão no terminal
ATL0/ATL1	Linefeed off/on	Envia um linefeed após o recebimento de um carriage return, o que na prática se traduz a receber códigos subsequentes em linhas subsequentes.
ATH0/ATH1	Header off/on	Mostra os bytes adicionais do header caso ligado, o que acarreta na resposta do veiculo ser formatada com os headers.
ATS0/ATS1	Printing of Spaces off/on	Controla a adição ou não de espaços na resposta da ECU, o que pode facilitar na leitura dos bytes.
ATAL	Allow Long Messages	Permite o recebimento de mensagens com mais de sete bytes, o que vem restringido como padrão
ATDP	Display Protocol	Imprimi no terminal o protocolo que está sendo utilizado.
ATSP0	Set Protocol Auto	Definir o protocolo a ser utilizado, sendo 0 definido por automático, onde todos os protocolos são testados até receber uma resposta.
ATMA	Monitor All	Imprimir no terminal todos os códigos que estão chegando na rede.

Fonte: Próprio Autor

4.1.2 Testes Preliminares com o ELM 327

Devido a ausência de diagramas elétricos de veículos e, por consequência, o conhecimento da arquitetura da rede veicular procedeu-se com testes de tentativa e erro a fim de determinar a estrutura de rede implementada.

Utilizado em conjunto com o software PuTTy para Windows, com suas configurações de *setup* sendo adquiridas no *datasheet* do ELM 327, impôs-se o *Baudrate* padrão de 38400 para uma conexão serial. Ao acessar Meu Computador>Propriedades do sistema> Gerenciador de Dispositivos, encontrou-se na sessão portas o nome da porta designada (COM 09) para o ELM 327. Com estas configurações conseguiu-se iniciar uma seção no terminal e utilizar os comandos pré definidos do ELM 327.

Destacam-se alguns comandos, como o ATZ que reseta o ELM e imprimi no terminal a versão do software.

4.1.2.1 Origem e Versão

Percebeu-se logo no primeiro teste, ainda sem a conexão com o veiculo (apenas ELM 327 conectado ao computador), a versão v1.5, e após rápida pesquisa no fabricante e no mercado, concluiu-se que, apesar do nome veiculado, tratava-se de uma cópia não legitima do chip da empresa canadense. De fato, todos os produtos adquiridos até o final dos testes mostraram a mesma qualidade.

Ao desmontar alguns componentes percebeu-se que as cópias não legitimas possuíam um microcontrolador PIC que emulava o funcionamento do chip original. Encontrou-se também bibliotecas com as funcionalidades do chip para sua eventual reprodução. A Figura 31 mostra um ELM 327 genérico desmontado.

Figura 31 – Diagrama do funcionamento SF-X9

Fonte: www.fadvidor.net

4.1.3 Testes no veículo com o ELM 327

Em um primeiro teste, configurou-se a aplicação com a sequência de comando: ATH1, ATS1, ATL1, ATAL, e utilizou-se o código ATMA. Imediatamente recebeu-se cerca de 40 códigos e uma mensagem de "Buffer Full". Percebeu-se que a quantidade de bytes da rede de alguma forma excedia a capacidade de processamento. Percebeu-se também que o numero de códigos recebidos até a falha variava e não seguia nenhum padrão perceptível.

4.1.3.1 Mudança de Baudrate

Pesquisou-se no *datasheet* a possibilidade de expandir o *buffer*, sem nenhum sucesso. Encontrou-se a possibilidade de alterar o *baudrate* para uma taxa de 115200 bit/s, e optou-se por tentar explorar esta modificação a fim de contornar a restrição, tendo em vista que uma taxa quase três vezes maior de processamento de dados poderia resultar em menos dados alocados no *buffer* e, por extensão, um recebimento contínuo de dados. Para tal utilizou-se o comando ATBRD XX, onde o *baudrate* em kbps seria 4000/XX e, no caso de 115.200 kbps, aproximadamente 28. Percebeu-se porem que, devido ao tempo de resposta esperado pelo programa para a confirmação da mudança ser entre 75 msec e 1,27 s, esta mudança seria apenas possível através de software. Elaborou-se então uma rotina em Python que realizava a mudança, mostrada no Anexo B.

Apesar de o número de bytes recebidos ser consideravelmente maior com a mudança do Baudrate, a mensagem de *Buffer Full* continuava sendo recebida. Aprendeu-se posteriormente a mudar o *baudrate* através de parâmetros programáveis do ELM 327 usando o comando programável PP 0C.

4.1.4 Comparação entre chip original e cópias

Neste ponto havia a incerteza da origem do erro. Julgava-se possível que este fosse advento da versão não oficial do *chip*, ou mesmo da fabricação sem a utilização do chip genuíno.

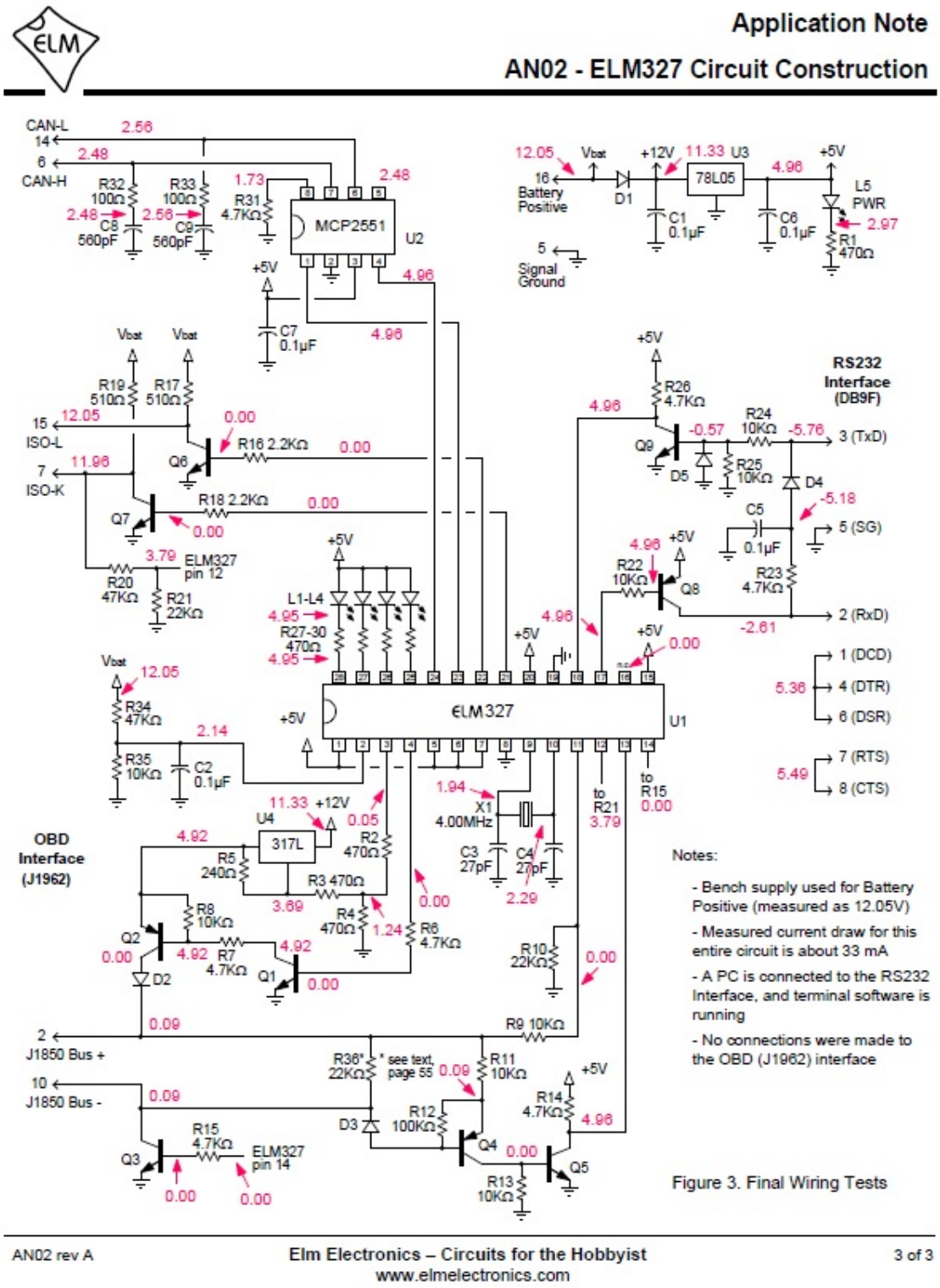
Encomendou-se portanto o chip genuíno e tratou-se de fazer o circuito PCB para a instalação do mesmo.

Desta forma foi contruido um circuito com o ELM327 em proto-board segundo o esquema oficial disponível no site da ELM Eletronics e mostrado na Figura 32.

É valido notar que o esquema possui mais opções de diagnóstico, além das opções baseadas em CAN.

Resolveu-se criar uma placa de circuito impresso para ter-se uma solução mais robusta, e mitigar os erros provindos de conexões removíveis. Utilizou-se o modelo disponível no site da ELM Eletronics, o que pareceu ser uma solução satisfatória, apesar

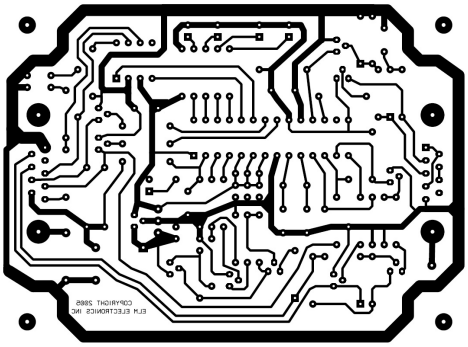
Figura 32 – Diagrama ELM 327



Fonte: ELM Eletronics

do tamanho do circuito e a pequena espessura de suas vias. A Figura 33 mostra o *top silk* da placa.

Figura 33 – Top-Silk ELM 327



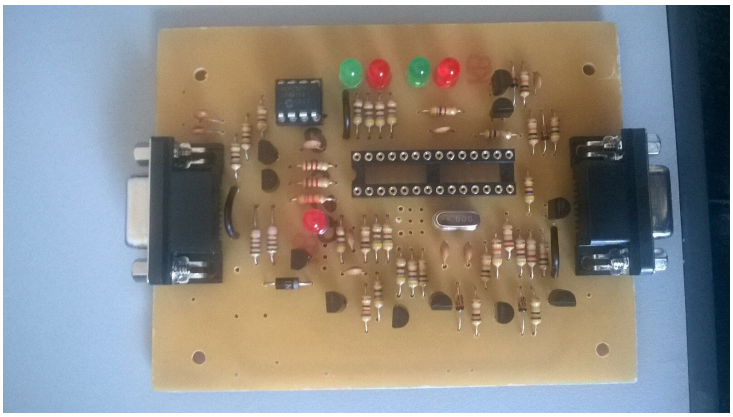
Fonte: ELM Eletronics

Buscou-se um orçamento em fábrica de PCBs recebendo um orçamento de aproximadamente U$ 700,00 para fabricação e montagem de duas peças, sem contar o custo dos componentes.

Sem verba suficiente para tal, utilizou-se impressora laser e prensa hidráulica para passar o circuito à placa e escolheu-se corroer a placa manualmente. Após perceber-se a baixa qualidade do resultado, além da dificuldade de realizar todos os furos da peça, decidiu-se por realizar uma segunda tentativa com o uso de uma CNC.

Com uma boa qualidade de placa, soldou-se os componentes da placa, sendo o resultado mostrado na Figura 34.

Figura 34 – PCB com montagem para ELM 327



Fonte: Próprio Autor

4.1.4.1 Testes com ELM 327 Original

Testou-se por fim a placa, e conseguiu-se utilizá-la no carro. Percebeu-se, porém o mesmo erro de *buffer full*, exatamente como nos modelos não autênticos. Desta forma resolveu-se retornar aos modelos comprados, por serem mais estáveis, e procurar uma solução em software.

4.1.5 Testes com o Docklight

Nesta etapa o software Docklight, um terminal serial, foi utilizado, pois este permite a programação de envios automáticos. Progrou-se o envio de um comando ATMA a cada vez que fosse recebida a mensagem "*Buffer Full*", de forma a tentar contornar o problema. Seria uma forma de adquirir dados do veículo, mesmo que perdendo alguns bits.

De fato, o modelo funcionou e pode-se fazer um log de alguns minutos de dados da rede. Com a solução pronta, começou-se a investigação em um automóvel.

Os testes iniciais foram feitos com uma Toyota Corolla XEI, MY 2014. O veiculo era equipado com travas e vidros elétricos, faróis de milhas, acionamento elétrico dos vidros e portas (inclusive porta malas), freios ABS (*Antiblockiersystem*), transmissão automática, central multimídia com controle no volante, piloto automático, travamento automático das portas a partir de certa velocidade, *airbag*, além das funções elétricas básicas como pisca alerta, setas, luzes internas e faróis.

Avaliou-se o conector OBD-II, presente atrás de uma portinhola à esquerda e abaixo do painel do motorista. Ele possuía, como esperado, os pinos 4, 5, e 16 ativos, além dos pinos 6 e 14 correspondentes à rede CAN, como pode ser observado na Figura 35. Percebeu-se também os pinos 9, 12 e 13 que não fazem parte do padrão OBD-II, e são portanto, exclusivos da montadora.

Com o ELM 327 aliado ao software DockLight, realizou-se as mesmas programações que anteriormente e registrou-se um log de códigos com o carro com o motor desligado e a chave na posição IGN. Percebeu-se que após o desligamento da chave, a rede permanecia ativa por cerca de 1 minuto e então desligava.

Primeiramente, copiou-se o *log* de dados recebidos e utilizou-se o *software* Excel para agrupar os dados. Resolveu-se separar os dados de *header* para saber quantas ECUs estavam comunicando entre si, e talvez obter uma estimativa do número de ECUs trabalhando na rede. Encontrou-se aproximadamente 20 *headers* diferentes.

A ideia presente era comparar um *log* no estado normal de atividade (sem nenhuma ação externa), com um *log* com ações externas. Diversos códigos se repetiriam, porém os que não se repetissem eram candidatos a representantes das ações externas. Para a primeira ação externa, julgou-se interessante o travamento e destravamento das portas.

Figura 35 – Conector OBD - Corolla Xei MY 2014

Fonte: Próprio Autor

Após alguns registros, comparou-se os dados e percebeu-se o destaque de *headers* começados em 6. Em consulta ao *datasheet* do ELM 327 reconheceu-se a possibilidade de uso do comando ATCRA XXX, que filtra a recepção apenas para os códigos com o *header* XXX.

Dado o pequeno numero de *headers*, resolveu-se aplicar a técnica a todos os headers encontrados.

Como resultado, encontrou-se o *header* 621 que variava de acordo com o estado de trancamento das portas do veículo. Repetiu-se o mesmo tipo de testes para ações como acionamento das setas, pressionar o acelerador, acionamento do pisca alerta, entre outros. Percebeu-se então que o *header* 2C1 representava o acelerador. Os valores dos códigos variavam conforme a pressão no pedal, até um valor limite.

Explorou-se um código padrão para confirmar que o envio estava funcionando. enviou-se o código 0902 que solicita VIN (*Vehicle Identification Number*, mais conhecido como o número do chassi do veículo).

A resposta tem três partes, devidamente numerada como 0, 1 e 2, e mostradas na Figura 36. Os primeiros dois bytes (49 02) mostram que se trata de um resposta ao comando 09 02. O terceiro byte (01) mostra a quantidade de dados que serão entregues (o veículo só pode ter um VIN). Em seguida tem-se 17 bytes que correspondem a representação do número VIN do veículo em Hexadecimal. Convertemos os valores da Figura 36 encontramos precisamente o VIN 9BRBD48EXE2613168.

Figura 36 – Resposta a solicitação do número de chassi através do comando 0902

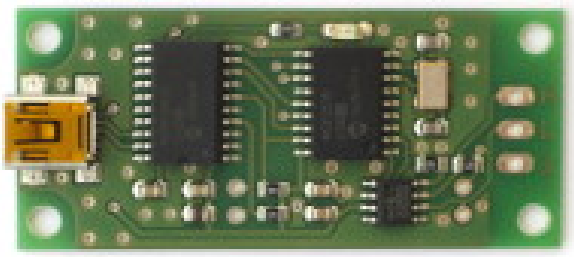
```
0: 49 02 01 39 42 52    <CR>
1: 42 44 34 38 45 58 45 <CR>
2: 32 36 31 33 31 36 38 <CR>
```

Fonte: Próprio Autor

4.2 USBTin

Devidos às limitações do *buffer* do ELM327, buscou-se uma solução mais eficiente para conectar-se ao veículo. Comprou-se então o dispositivo USBTIN do desenvolvedor alemão Thomas Fischl. O dispositivo é composto de um microcontrolador PIC 18f14k50, um controlador CAN MCP2515 e um *transceiver* CAN MCP 2551. Há ainda a possibilidade de colocar um resistor de terminação na placa. A Figura 37 mostra o produto comprado.

Figura 37 – USBTIN



Fonte: Fischl.de

A placa se conecta ao computador via USB e para utilizá-la é necessário instalar primeiro a biblioteca para sistema operadiconal Linux e configurar a velocidade da rede a qual irá se conectar o dispositivo. O dispositivo é reconhecido pelo sistema Operacional como uma porta serial virtual, sendo a primeira denominada ACM0, e existem oito opções para a configuração de velocidades, que se iniciam em 10,4 kbit/s e vão até 1 mbit/s.

Houve a necessidade de soldar-se três fios à placa: o GND de aterramento e os dois da rede CAN (CANH e CANL). Conectou-se os fios diretamente ao conector OBD, sendo o GND conectado ao pino 5, o CAN_H ao pino 6 e o CAN_L ao pino 14.

Após configurar e conectar a placa ao veículo, utilizou-se o comando *candump* para receber um log de todos os dados que estavam trafegando na rede.

Inicialmente encontrou-se uma quantidade grande de códigos sendo publicados a uma alta frequência na rede, e não se encontrou erros de *buffer* mesmo em tempos prolongados de monitoramento. Uma breve comparação visual, sem nenhum embasamento científico, levava a crer que a quantidade de códigos recebidos era significativamente maior

que a encontrada no log do ELM 327.

4.2.1 Comparação entre ELM 327 e USBTin

Em um segundo teste conectou-se o ELM327 e o USBTIN em paralelo para fazer um *log* simultâneo dos dados recebidos pelos dois dispositivos.. Percebeu-se que o ELM327 mostrava menos dados do que o USBCAN, como mostrado no *log* do Anexo C.

Tabela 7 – Log USBTin - Teste I

Header	**Incidência**
020	908
022	871
023	402
025	737
223	455
224	437
420	11
423	10
2C1	334
2C4	445
2D0	327
2D2	314
3D0	40
4C1	12
4C2	11
4C3	11
4C6	10
4C7	12
Total Geral	**5347**

Fonte: Próprio Autor

Tabela 8 – Log ELM327 - Teste I (ATMA Manual)

Header	**Incidência**
00	1
020	1
022	2
023	7
025	16
223	4
224	8
2C1	28
2C4	26
2D0	26
2D2	5
2E3	6
3	7
3C1	17
3C4	60
3D0	15
3D2	16
3E3	140
3E4	24
404	25
4C1	26
4C3	27
4C7	28
6A3	29
7C4	160
7D2	35
7E2	36
7E3	114
7E4	40
7E5	83
BUF	43
Total Geral	1055

Fonte: Próprio Autor

Estimou-se que devido as diferentes interações do IC, o ELM327 ignorava diversos

dados da rede, enquanto o USBTin mostrava os dados puros. Não se encontro uma solução para este problema com o ELM327.

Testou-se ambos os equipamentos em paralelo fazendo o mesmo log, a fim de comparar a quantidade dos pacotes recebidos. Realizou-se este teste de duas formas: primeiramente apertando repetidamente a tecla ENTER a fim de repetir o comando ATMA para o ELM 327 toda que se recebia a mensagem "*Buffer Full*", enquanto o USBTin realizava um log com o comando *candump*.

Para o segundo código programou-se o programa *Docklight* para enviar o código ATMA automaticamente a cada vez que se recebia a mensagem *Buffer Full*, enquanto o USBTin novamente realizava um log com o comando *candump*. É valido notar que nas análises, com acionamento manual e automático do ELM 327, não foram realizadas em períodos de tempo equivalentes. Os *headers* e sua incidência para o teste I (acionamento manual do ATMA) estão mostrados nas Tabelas 7 e 8

Ao comparar-se os dados do primeiro log, realizado de forma manual com o ELM327, percebeu-se que o ELM captou um total de 1055 mensagens enquanto o USBTin, 5347. Assim, o ELM 327 obteve apenas 19,73% dos dados da rede. Percebe-se que o ELM 327 emitiu a mensagem de *Buffer Full* 43 vezes, ou seja, uma vez a cada 24,53 mensagens processada pelo ELM 327, ou uma vez a cada 124 mensagens obtidas pelo USTin.

Percebe-se também que o ELM recebeu mensagens com únicas, mostrado em destaque na Tabela 9. Como estes códigos não foram registrdos pelo USBTin, entendeu-se que eles são provindos de algum erro, ou publicação do ELM 327.

Tabela 9 – Headers encontrados apenas no ELM 327 no teste I

Header	**Incidência**
2E3	6
3	7
3C1	17
3C4	60
3D2	16
3E3	140
3E4	24
404	25
6A3	29
7C4	160
7D2	35
7E2	36
7E3	114
7E4	40
7E5	83

Fonte: Próprio Autor

O teste de forma automática resultou em uma maior captura de pacotes por parte do ELM 327. Em um log onde o USTin retornou 2601 códigos, o ELM 327 retornou 977 linhas de códigos, sendo apenas 703 representavam códigos da rede (o restante sendo composto de dados de log como *timestamps* e erros de *buffer full*. Assim, percebe-se que o ELm 327 recebeu apenas 27% dos códigos da rede.Os dados estão mostrados nas Tabelas 10 e 11.

Neste log encontrou-se 45 mensagens de *buffer full*, ou seja, uma a cada 15,62 códigos recebidos pelo ELM 327. Não encontrou-se headers singulares no novo log.

Tabela 10 – Log USBTin - Teste II

Header	**Incidência**
020	442
022	407
023	191
025	371
223	222
224	217
2C1	163
2C4	221
2D0	156
2D2	152
3D0	21
420	4
423	6
4C1	5
4C2	5
4C3	6
4C6	6
4C7	6
(vazio)	
Total Geral	**2601**

Fonte: Próprio Autor

Tabela 11 – Log ELM327 - Teste II
(ATMA Automático)

Header	**Incidência**
020	107
022	95
023	91
025	102
223	57
224	49
2C1	45
2C4	60
2D0	45
2D2	34
3D0	4
420	1
423	2
4C1	3
4C2	2
4C3	3
4C6	1
4C7	2
BUF	45
(vazio)	
Total Geral	**748**

Fonte: Próprio Autor

Devido a maior facilidade de utilização, a maior consistência dos dados, e ao maior número de códigos registrados, concluiu-se que o USBTin era uma ferramenta mais completa e escolheu-se ela para continuar as pesquisas.

4.2.2 Testes preliminares com o USBTin

Resolveu-se analisar os códigos em Excel, tentando contabilizar quantos códigos diferentes eram exibidos. Percebeu-se que os códigos variavam tanto em valores como em quantidades de bytes.

Para o Toyota Corolla 2014 criou-se um log durante seu estado normal de funcionamento denominado aqui de *baseline*.

Encontrou-se 75 códigos diferentes em um total de 9905 códigos. Contabilizou-se, em seguida, 38 headers diferentes encontrados no mesmo *baseline*, a variação de códigos com o mesmo endereçamento. Os headers incidiram segundo mostrado na Tabela 12.

Pode-se perceber que a incidência de headers não foi homogênea, o que era esperado. Como foi citado, é normal que sistemas mais importantes se comuniquem a uma frequência maior do que sistemas secundários.

Tabela 12 – Incidência de *header* nos testes com o Corolla Xei MY 2014

Header	**Incidência**	**Header**	**Incidência**
0B0	201	0B2	169
0B4	5	223	34
224	70	260	105
262	104	2C1	64
2C4	85	2D0	65
380	2	38A	2
394	6	399	2
3A0	21	3A1	2
3B0	2	3B1	2
3B3	4	3B4	2
3B7	7	440	11
442	10	4C1	3
4C3	2	4C8	2
4DC	2	4DD	2
610	4	611	2
617	2	620	7
621	2	622	2
624	2	630	2
638	2	640	2

Fonte: Próprio Autor

Procedeu-se a tentar isolar algum código do veículo de forma eficaz e criou-se uma rotina em Excel que comparava os códigos de um log específico, com ações adicionais, com os códigos do log do carro em estado *idle*, chamados aqui de *baseline*.

Esperava-se que esta rotina isolaria os códigos que variavam em relação ao *baseline*, destacando os códigos responsáveis pelas mudanças. Desta forma, qualquer ação externa que fosse publicada na rede poderia ser isolada.

Rapidamente entendeu-se que esta abordagem seria muito trabalhosa pois o log produzia um número muito grande de códigos, de forma a exceder a capacidade computacional disponível, deixando o trabalho ineficiente.

Além disso a rede CAN não permite isolar o remetente da mensagem, de forma que mesmo que uma mensagem fosse devidamente isolada, não seria possível ter certeza de quem a enviou.

4.2.3 Testes com Máscara no header

Testou-se inicialmente pressionar o pedal do acelerador de uma Toyota Hilux SRV MY 2015 e obter os valores do mesmo. Após a análise, percebeu-se o header 2C1 novamente e toda a sua variação de códigos entre o estado de percurso nulo e o estado de percurso máximo do pedal.

De fato, ao utilizar-se uma máscara para filtrar apenas os dados deste header, percebeu-se claramente, em tempo real, a variação destes dados.

Percebeu-se ainda que os primeiros 6 bytes permanecem constantes, enquanto os últimos dois bytes variam. Quando não há aceleração nenhuma os dois bytes finais são 00 2B, enquanto na posição máxima percebe-se os valores C8 EB. Parte do log está mostrado na Tabela 13.

4.3 SF-X9

Percebeu-se que para comparar os códigos utilizados pelas montadoras e analisar sua padronização seria mais simples isolar os códigos enviados por um dispositivo externo e sua consequente resposta.

Criou-se então um dispositivo composto de dois USBTin e uma interface USB que faria um log independente de cada lado, já que na rede não é possivel saber a origem de um pacote, e repassaria os dados para o outro lado, sem a duplicação dos dados. Este equipamentopermitiria mostrar a direção dos códigos enviados, e facilitar assim o entendimento da comunicação CAN.

Este dispositivo denominado aqui de SF-X9 pode ser conectado entre um nó e a rede, e isolar exatamente o que o nó (carro ou dispositivo externo) envia, sem interferir no envio das mensagens, como mostrado na Figura 38.

Percebeu-se também, no mercado, diversos dispositivos que interagem com veículos através da porta OBD, sendo considerados um nó na rede, e exibem diversas informações como a rotação, temperatura do óleo, velocidade do veículo, entre outros. Um deles é o aplicativo iDiag para IOS que utiliza o conector de mesmo nome como interface.

4.3.1 Testes com o SF-X9

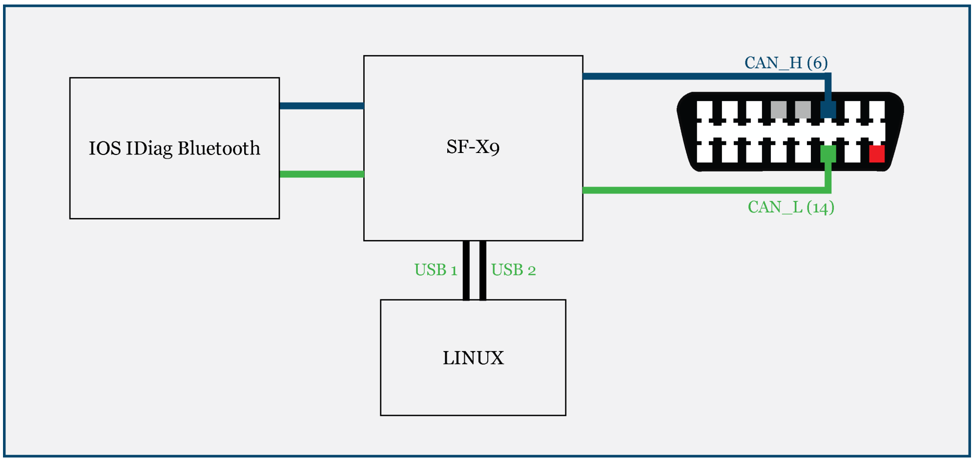
Testou-se o dispositivo em ponte entre o aplicativo e o veículo e solicitou-se a exibição independente da rotação do motor e da temperatura do líquido refrigerante.

Tabela 13 – Log da variação da posição do pedal do acelerador (0 - 100%) - Toyota Hilux SRV MY 2015

Dispositivo	**Header**	**PCI**	**Data Bytes**
slcan0	2C1	[8]	08 FC 60 FC 60 A0 00 2B
slcan0	2C1	[8]	08 FC 60 FC 60 A0 00 2B
slcan0	2C1	[8]	08 FC 60 FC 60 A0 00 2B
slcan0	2C1	[8]	08 FC 60 FC 60 A0 00 2B
slcan0	2C1	[8]	00 FC 60 FC 60 A0 01 24
slcan0	2C1	[8]	00 FC 60 FC 60 A0 13 36
slcan0	2C1	[8]	00 FC 60 FC 60 A0 22 45
slcan0	2C1	[8]	00 FC 60 FC 60 A0 2E 51
slcan0	2C1	[8]	00 FC 60 FC 60 A0 37 5A
slcan0	2C1	[8]	00 FC 60 FC 60 A0 3E 61
slcan0	2C1	[8]	00 FC 60 FC 60 A0 43 66
slcan0	2C1	[8]	00 FC 60 FC 60 A0 47 6A
slcan0	2C1	[8]	00 FC 60 FC 60 A0 4A 6D
slcan0	2C1	[8]	00 FC 60 FC 60 A0 4D 70
slcan0	2C1	[8]	00 FC 60 FC 60 A0 50 73
slcan0	2C1	[8]	00 FC 60 FC 60 A0 55 78
slcan0	2C1	[8]	00 FC 60 FC 60 A0 57 7A
slcan0	2C1	[8]	00 FC 60 FC 60 A0 5B 7E
slcan0	2C1	[8]	00 FC 60 FC 60 A0 60 83
slcan0	2C1	[8]	00 FC 60 FC 60 A0 65 88
slcan0	2C1	[8]	00 FC 60 FC 60 A0 6A 8D
slcan0	2C1	[8]	00 FC 60 FC 60 A0 6F 92
slcan0	2C1	[8]	00 FC 60 FC 60 A0 74 97
slcan0	2C1	[8]	00 FC 60 FC 60 A0 79 9C
slcan0	2C1	[8]	00 FC 60 FC 60 A0 7E A1
slcan0	2C1	[8]	00 FC 60 FC 60 A0 83 A6
slcan0	2C1	[8]	00 FC 60 FC 60 A0 88 AB
slcan0	2C1	[8]	00 FC 60 FC 60 A0 8D B0
slcan0	2C1	[8]	00 FC 60 FC 60 A0 91 B4
slcan0	2C1	[8]	00 FC 60 FC 60 A0 96 B9
slcan0	2C1	[8]	00 FC 60 FC 60 A0 9B BE
slcan0	2C1	[8]	00 FC 60 FC 60 A0 A0 C3
slcan0	2C1	[8]	00 FC 60 FC 60 A0 A2 C5
slcan0	2C1	[8]	00 FC 60 FC 60 A0 A9 CC
slcan0	2C1	[8]	00 FC 60 FC 60 A0 B1 D4
slcan0	2C1	[8]	00 FC 60 FC 60 A0 B6 D9
slcan0	2C1	[8]	00 FC 60 FC 60 A0 BA DD
slcan0	2C1	[8]	00 FC 60 FC 60 A0 BD E0
slcan0	2C1	[8]	00 FC 60 FC 60 A0 BE E1
slcan0	2C1	[8]	00 FC 60 FC 60 A0 BF E2
slcan0	2C1	[8]	00 FC 60 FC 60 A0 C0 E3
slcan0	2C1	[8]	00 FC 60 FC 60 A0 C1 E4
slcan0	2C1	[8]	00 FC 60 FC 60 A0 C1 E4
slcan0	2C1	[8]	00 FC 60 FC 60 A0 C1 E4

Fonte: Próprio Autor

Figura 38 – Diagrama do funcionamento SF-X9



Fonte: Próprio Autor

Criou-se simultaneamente os logs do lado do veículo e do dispositivo.

4.3.1.1 Rotação do Motor

No lado do dispositivo notou-se, como esperado, apenas a presença de dados enviados pelo dispositivo. O dispositivo utiliza o *header* padrão 7DF. Os códigos estão evidenciados na Tabela 14.

Tabela 14 – Códigos enviados pelo dispositivo - RPM

Dispositivo	Header	PCI	Data Bytes
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00
slcan1	7DF	[8]	02 01 0C 00 00 00 00 00

Fonte: Próprio Autor

Do lado do veículo percebeu-se a resposta do veículo, com header igual ao header 7E8. A Tabela 14 mostra o código associado a rotação do veículo observada nos ponteiros.

Percebe-se que o quarto e quinto byte são uma representação da rotação em hexadecimais. Se chamarmos o quarto byte de A e o quinto byte de B, podemos calcular a rotação real através da Fórmula 4.1, mostrada na norma SAE 1979.

Tabela 15 – Códigos enviados pelo veículo - RPM

Dispositivo	**Header**	**PCI**	**Data Bytes**	**Rotação visualizada [RPM]**
slcan0	7E8	[8]	04 41 0C 00 00 00 00 00	0
slcan0	7E8	[8]	04 41 0C 0D 83 00 00 00	850
slcan0	7E8	[8]	04 41 0C 17 89 00 00 00	1500
slcan0	7E8	[8]	04 41 0C 20 9A 00 00 00	2000

$$rpm = \frac{((A * 256) + B)}{4} \tag{4.1}$$

Os valores calculados através da Fórmula 4.1, e mostrados na Tabela 16 comprovam a aproximação.

Tabela 16 – Parâmetros e cálculo do RPM

A [Hex]	**B [Hex]**	**A [Dec]**	**B [Dec]**	**Rotação calculada [RPM]**
0D	83	13	131	864.75
17	89	23	137	1506
20	9A	32	154	2086.5

4.3.1.2 Temperatura do Líquido Refrigerante

Do lado do veículo percebeu-se a resposta do veículo, com header igual ao header 7E8. A Tabela 18 mostra o código associado a temperatura do líquido refrigerante do veículo observada nos ponteiros.

Percebe-se que o quarto byte é uma representação da temperatura em hexadecimais. Segundo a norma SAE 1979, a temperatura pode ser calculada segundo a Fórmula 4.2.

Tabela 17 – Códigos enviados pelo dispositivo - Líquido Refrigerante

Dispositivo	**Header**	**PCI**	**Data Bytes**
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00
slcan1	7DF	[8]	02 01 05 00 00 00 00 00

Fonte: Próprio Autor

Tabela 18 – Códigos enviados pelo veículo - Líquido Refrigerante

Dispositivo	**Header**	**PCI**	**Data Bytes**	**Temperatura no Display [°C]**
slcan0	7E8	[8]	04 41 05 56 00 00 00 00	46

$$rpm = C - 40 \tag{4.2}$$

onde C é o valor do quarto byte em decimal.

Calculou-se o valor através da Fórmula 4.2, que, como pode ser observado na Tabela 19, comprova a aproximação.

Tabela 19 – Códigos enviados pelo veículo - Líquido Refrigerante

A [Hex]	**A [Dec]**	**Temperatura Calculada [°C]**
56	86	46

4.4 Comparação

Comparou-se o código com o caro pelo descrito pela norma SAE 1997. Confirmou-se os parâmetros

Percebe-se que existem códigos padrões para todas as montadoras, evidenciados aqui na exibição da rotação do motor. Qualquer dispositivo que enviar o código 7DF 02 01 0C na rede fará com que a ECU responsável responda a rotação do motor em forma de mensagem CAN.

Pode-se perceber que, devido a padronização, qualquer veículo que contenha rede CAN pode ser solicitado desta forma e que muitos outros parâmetros do motor podem também ser analisados desta forma. Eles incluem a temperatura do óleo, a velocidade, quantidade de combustível no tanque, entre muitos outros mostrados em detalhe no Anexo A.

5 . RESULTADOS

A criação da rede CAN há quase 30 anos resultou na expansão das possibilidades para industria automotiva. Com a diminuição de fios e da integração dos diversos sistemas em uma rede robusta e de implementação padronizada, pode-se criar sistemas cada vez mais integrados que, por sua vez, possibilitaram tecnologias como o trancamento automático da porta após velocidade determinada. Com isso a rede CAN possibilitou o aumento da segurança e da versatilidade dos veículos.

A análise teórica de redes automotivas mostra que as tecnologias são robustas, com características que evitam a interferência eletromagnética, e são criadas para isolar a falha de dispositivos na rede aumentando a confiabilidade do sistema como um todo. A padronização dos meios físicos e dos protocolos de comunicação, por sua vez, contribuíram para a evolução dos dispositivos e as ferramentas disponíveis no mercado, integrando as diversas montadoras mundiais.

A análise das ferramentas disponíveis no mercado, suas performances e interações com o veículo, mostrou que o é possível colher dados de veículos em tempo real. Devido a padronização de códigos na norma SAE J1979, grandezas como o de velocidade, rotação do motor, temperatura do líquido refrigerante se mostram possíveis de serem feitas. Diversos outros dados apresentados no Anexo I, como tempo em que o motor está ligado e quantidade de combustível no tanque aumentam ainda mais as possibilidades de medição feitas na rede CAN.

A importância disto se da ao passo que grandezas que deveriam ser medidas por observação, ou por sensores adicionais instalados em veículos, podem ser medidas de forma digital e, por extensão, transmitidas de forma automática.

A padronização, a possibilidade de leitura de sensores já existentes e a transmissão de dados, aliada a dispositivos robustos com custos relativamente pequenos, permite criar soluções comerciais eficientes de telemetria que podem ser implementados em qualquer veículo moderno.

Neste sentido frotas de ônibus podem ser monitoradas quanto a forma de condução do motorista, recebendo em tempo real dados que podem impactar diretamente no custos de operação e na qualidade do serviço.

Como exemplo, a variação na rotação do motor pode ser diretamente relacionada ao consumo de combustível. A mesma, aliada a velocidade do veículo, pode apontar para frenagens e acelerações bruscas que resultam em uma experiência ruim para o usuário. A medição do nível de combustível do tanque pode servir como uma medida de segurança contra furtos de combustíveis. A medição do sensor de oxigênio ou de qualquer outro parâmetro do motor pode auxiliar no agendamento de manutenções e evitar a quebra de veículos durante o uso, impactando também diretamente no custo de operação e na

experiência dos usuários.

Estas vantagens estão também disponíveis para motoristas de veículos que querem ter mais informações sobre seus veículos em tempo real. Dispositivos comerciais com interfaces *bluetooth*, possibilitam a integração destes dados com *smartphones*, e facilitam o acesso a estas ferramentas.

Assim, as redes padronizadas e de fácil acesso, permitem maior controle sobre os veículos, e criam assim, novas possibilidades para veículos cada vez mais conectados.

6 . CONCLUSÃO

O trabalho de conclusão de curso analisou os diversos aspectos das redes veiculares. Abordou-se configurações físicas das redes e suas características construtivas. Comparou-se as capacidades e o uso dos diferentes tipos de redes. Focou-se finalmente na área de diagnóstico automotivo e aprofundou-se no estudo de redes CAN, seus protocolos de comunicação e suas variações.

Na parte prática iniciou-se o estudo com a ferramenta mais disponível e acessível no mercado para conexão com a rede CAN. Analisando a diferença entre a versão original e a versão chinesa não se percebeu diferenças significativas. Explorou-se uma ferramenta mais sofisticada, o USBTin e, utilizando-a em paralelo com o ELM 327, mostrou-se analiticamente as limitações do ELM 327.

Analisou-se então os códigos de um veículo, conseguindo extrair os valores do acelerador através de análise e comparação de logs utilizando software.

Construiu-se então uma ferramenta a ser instalada entre um nó da rede e capaz de mostrar as direções dos códigos de forma a isolar os códigos.

Testou-se a ferramenta com um aplicativo de diagnóstico, que utiliza o sistema operacional IOS, capaz de exibir os códigos de rotação do motor e conseguiu-se isolar tanto os códigos enviados pela ferramenta externa como as respostas enviadas pelo veículo. Explicou-se na prática a utilização da ferramenta e a conversão dos códigos. Por fim repetiu-se os testes para a temperatura do líquido refrigerante.

Por fim, analisou-se as aplicações comerciais, citando possíveis aplicações práticas da tecnologia e suas vantagens refletidas de forma comercial como na experiência de usuários de veículos modernos.

REFERÊNCIAS

BOSCH, R. **Bosch Automotive Electrics and Automotive Electronics**. Wiesbaden: Springer Fachmedien 3, 2014.

CALIFORNIA AIR RESOURCES BOARD. **Frequently Asked Questions**. 2016. Disponível em: <https://www.arb.ca.gov/msprog/obdprog/obdfaq.htm>. Acesso em: 18 de jan. 2017.

CODE of Federal Regulations. 2013. Disponível em: <http://www.ecfr.gov/cgi-bin/ECFR?page=browse>. Acesso em: 18 de jan. 2017.

CORRIGAN, S. **Introduction to the Controller Area Network (CAN)**. [S.l.], 2008. Acesso em Janeiro/2016. Disponível em: <http://www.ti.com/lit/an/sloa101a/sloa101a.pdf>.

DAVIS, R. I. et al. Controller area network (can) schedulability analysis: Refuted, revisited and revised. **Real-Time Systems**, v. 35, n. 3, p. 239–272, 2007. ISSN 1573-1383. Disponível em: <http://dx.doi.org/10.1007/s11241-007-9012-7>.

E/E Diagnostic Test Modes. Society of Automobile Engineers, 2012. v. 2012.

ELETRONIC DESIGN. **Fundamentals of Communications Access Technologies: FDMA, TDMA, CDMA, OFDMA, AND SDMA**. 2013. Disponível em: <http://electronicdesign.com/communications/fundamentals-communications-access-technologies-fdma-tdma-cdma-ofdma-and-sdma#\T1\textquotedblrightTDMA\T1\textquotedblright>. Acesso em: 18 de jan. 2017.

ELM327. **OBD to RS323 Interpreter**. 2016. Disponível em: <https://www.elmelectronics.com/wp-content/uploads/2016/07/ELM327DS.pdf>. Acesso em: 18 de jan. 2017.

INFORMATION technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. Geneva, CH, 1994. v. 1994.

ISO. **SO 11898-1:2015**. 2015. Disponível em: <http://www.iso.org/iso/catalogue_detail.htm?csnumber=63648>. Acesso em: 18 de dez. 2016.

JUNIOR, H. T. **Estudo dos protocolos de comunicação das arquiteturas eletroeletrônicas automotivas, com foco nas suas características e respectivas aplicações, visando o direcionamento para o uso adequado e customizado em cada categoria de veículo**. Dissertação (Mestrado) — Instituto Maua de Tecnologia, São Caetano do Sul - Brazil, 2012.

KESKIN, U. **In-vehicle communication networks : a literature survey**. Den Dolech 2, 5600 AZ Eindhoven, The Netherlands, 2009.

KRAUS, D. et al. Replacement of the controller area network (can) protocol for future automotive bus system solutions by substitution via optical networks. In: **2016 18th International Conference on Transparent Optical Networks (ICTON)**. [S.l.: s.n.], 2016. p. 1–8.

LAWRENZ, W. **Bosch Automotive Electrics and Automotive Electronics**. Wolfenbüttel: Springer, 2013.

MCCORD, K. **Automotive Diagnostic Systems**. 39966Grand Avenue, North Branch, MN: CarTecj Inc., 2011.

MITTAL, M.; SMART, D. Can network analysis #x2014; optimization, health prediction and diagnosis. In: **2015 IEEE International Transportation Electrification Conference (ITEC)**. [S.l.: s.n.], 2015. p. 1–4.

NATALE, M. D. **Understanding and using the Controller Area Network**. [S.l.], 2008.

NATIONAL INSTRUMENTS. **Controller Area Network (CAN) Overview**. 2014. Disponível em: <http://www.ni.com/white-%paper/2732/en/>. Acesso em: 18 de jan. 2017.

PRETSCHNER, A. et al. Software engineering for automotive systems: A roadmap. In: **2007 Future of Software Engineering**. Washington, DC, USA: IEEE Computer Society, 2007. (FOSE '07), p. 55–71. ISBN 0-7695-2829-5. Disponível em: <http://dx.doi.org/10.1109/FOSE.2007.22>.

ROAD Vehicles – Controller area network (CAN) - Part 2: High-Speed medium access unit. Geneva, CH, 2003. v. 2003.

ROAD vehicles — Diagnostics on Controller Area Networks (CAN) - Part 2: Network layer services. Geneva, CH, 2004. v. 2004.

ROAD vehicles — Diagnostics on Controller Area Networks (CAN) - Part 2: Network layer services. Geneva, CH, 2016. v. 2016.

TANENBAUM, A. S.; WETHERALL, D. J. **Computer Networks**. Wiesebaden: Prentice Hall, 2010.

VALASEK, C.; MILLER, C. **Adventures in Automotive Networks and Control Units**. Seattle, USA, 2014.

WOLF, M.; WEIMERSKIRCH, A.; PAAR, C. Security in automotive bus systems. In: **IN: PROCEEDINGS OF THE WORKSHOP ON EMBEDDED SECURITY IN CARS (ESCAR)'04**. [S.l.: s.n.], 2004.

ZHOU, C.; LUO, F. Design of the redundant can network based on canopen. In: **2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)**. [S.l.: s.n.], 2016. p. 782–786.

A . LISTA DE COMANDOS DE DIAGNÓSTICO MODO I - SAE J1979

Tabela 20 – A simple longtable example

PID	Bytes	Descrição	Min	Máx	Unidade
00	4	*PIDs supported [01 - 20]*			
01	4	*Monitor status since DTCs cleared. (Includes malfunction indicator lamp (MIL) status and number of DTCs.)*			
02	2	*Freeze DTC*			
03	2	*Fuel system status*			
04	1	*Calculated engine load*	0	100	%
05	1	*Engine coolant temperature*	-40	215	°C
06	1	*Short term fuel trim—Bank 1*	-100 (Reduce Fuel: Too Rich)	99.2 (Add Fuel: Too Lean)	%
07	1	*Long term fuel trim—Bank 1*			
08	1	*Short term fuel trim—Bank 2*			
09	1	*Long term fuel trim—Bank 2*			
0A	1	*Fuel pressure (gauge pressure)*	0	765	kPa
0B	1	*Intake manifold absolute pressure*	0	255	kPa
0C	2	*Engine RPM*	0	16,383.75	rpm
0D	1	*Vehicle speed*	0	255	km/h
0E	1	*Timing advance*	-64	63.5	° before TDC
0F	1	*Intake air temperature*	-40	215	°C
10	2	*MAF air flow rate*	0	655.35	grams/sec
11	1	*Throttle position*	0	100	%
12	1	*Commanded secondary air status*			
13	1	*Oxygen sensors present (in 2 banks)*			
14	2	*Oxygen Sensor 1A: VoltageB: Short term fuel trim*	0-100	1.27599.2	volts %
15	2	*Oxygen Sensor 2A: VoltageB: Short term fuel trim*			
16	2	*Oxygen Sensor 3A: VoltageB: Short term fuel trim*			

Continua na próxima página

Tabela 20 – *Continuação da página anterior*

PID	Bytes	Descrição	Min	Máx	Unidade
17	2	*Oxygen Sensor 4A: VoltageB: Short term fuel trim*			
18	2	*Oxygen Sensor 5A: VoltageB: Short term fuel trim*			
19	2	*Oxygen Sensor 6A: VoltageB: Short term fuel trim*			
1A	2	*Oxygen Sensor 7A: VoltageB: Short term fuel trim*			
1B	2	*Oxygen Sensor 8A: VoltageB: Short term fuel trim*			
1C	1	*OBD standards this vehicle conforms to*			
1D	1	*Oxygen sensors present (in 4 banks)*			
1E	1	*Auxiliary input status*			
1F	2	*Run time since engine start*	0	65,535	seconds
20	4	*PIDs supported [21 - 40]*			
21	2	*Distance traveled with malfunction indicator lamp (MIL) on*	0	65,535	km
22	2	*Fuel Rail Pressure (relative to manifold vacuum)*	0	5177.265	kPa
23	2	*Fuel Rail Gauge Pressure (diesel, or gasoline direct injection)*	0	655,350	kPa
24	4	*Oxygen Sensor 1AB: Fuel–Air Equivalence RatioCD: Voltage*	00	<2<8	ratioV
25	4	*Oxygen Sensor 2AB: Fuel–Air Equivalence RatioCD: Voltage*			
26	4	*Oxygen Sensor 3AB: Fuel–Air Equivalence RatioCD: Voltage*			
27	4	*Oxygen Sensor 4AB: Fuel–Air Equivalence RatioCD: Voltage*			
28	4	*Oxygen Sensor 5AB: Fuel–Air Equivalence RatioCD: Voltage*			
29	4	*Oxygen Sensor 6AB: Fuel–Air Equivalence RatioCD: Voltage*			
2A	4	*Oxygen Sensor 7AB: Fuel–Air Equivalence RatioCD: Voltage*			

Continua na próxima página

Tabela 20 – *Continuação da página anterior*

PID	Bytes	Descrição	Min	Máx	Unidade
2B	4	*Oxygen Sensor 8AB: Fuel–Air Equivalence RatioCD: Voltage*			
2C	1	*Commanded EGR*	0	100	%
2D	1	*EGR Error*	-100	99.2	%
2E	1	*Commanded evaporative purge*	0	100	%
2F	1	*Fuel Tank Level Input*	0	100	%
30	1	*Warm-ups since codes cleared*	0	255	count
31	2	*Distance traveled since codes cleared*	0	65,535	km
32	2	*Evap. System Vapor Pressure*	-8,192	8191.75	Pa
33	1	*Absolute Barometric Pressure*	0	255	kPa
34	4	*Oxygen Sensor 1AB: Fuel–Air Equivalence RatioCD: Current*	0-128	<2<128	ratiomA
35	4	*Oxygen Sensor 2AB: Fuel–Air Equivalence RatioCD: Current*			
36	4	*Oxygen Sensor 3AB: Fuel–Air Equivalence RatioCD: Current*			
37	4	*Oxygen Sensor 4AB: Fuel–Air Equivalence RatioCD: Current*			
38	4	*Oxygen Sensor 5AB: Fuel–Air Equivalence RatioCD: Current*			
39	4	*Oxygen Sensor 6AB: Fuel–Air Equivalence RatioCD: Current*			
3A	4	*Oxygen Sensor 7AB: Fuel–Air Equivalence RatioCD: Current*			
3B	4	*Oxygen Sensor 8AB: Fuel–Air Equivalence RatioCD: Current*			
3C	2	*Catalyst Temperature: Bank 1, Sensor 1*	-40	6,513.5	°C
3D	2	*Catalyst Temperature: Bank 2, Sensor 1*			
3E	2	*Catalyst Temperature: Bank 1, Sensor 2*			
3F	2	*Catalyst Temperature: Bank 2, Sensor 2*			
40	4	*PIDs supported [41 - 60]*			
41	4	*Monitor status this drive cycle*			
42	2	*Control module voltage*	0	65.535	V
43	2	*Absolute load value*	0	25,700	%
44	2	*Fuel–Air commanded equivalence ratio*	0	<2	ratio

Continua na próxima página

Tabela 20 – *Continuação da página anterior*

PID	Bytes	Descrição	Min	Máx	Unidade
45	1	*Relative throttle position*	0	100	%
46	1	*Ambient air temperature*	-40	215	°C
47	1	*Absolute throttle position B*	0	100	%
48	1	*Absolute throttle position C*			
49	1	*Accelerator pedal position D*			
4A	1	*Accelerator pedal position E*			
4B	1	*Accelerator pedal position F*			
4C	1	*Commanded throttle actuator*			
4D	2	*Time run with MIL on*	0	65,535	minutes
4E	2	*Time since trouble codes cleared*			
4F	4	*Maximum value for Fuel–Air equivalence ratio, oxygen sensor voltage, oxygen sensor current, and intake manifold absolute pressure*	0, 0, 0, 0	255, 255, 255, 2550	ratio, V, mA, kPa
50	4	*Maximum value for air flow rate from mass air flow sensor*	0	2550	g/s
51	1	*Fuel Type*			
52	1	*Ethanol fuel %*	0	100	%
53	2	*Absolute Evap system Vapor Pressure*	0	327.675	kPa
54	2	*Evap system vapor pressure*	-32,767	32,768	Pa
55	2	*Short term secondary oxygen sensor trim, A: bank 1, B: bank 3*	-100	99.2	%
56	2	*Long term secondary oxygen sensor trim, A: bank 1, B: bank 3*			
57	2	*Short term secondary oxygen sensor trim, A: bank 2, B: bank 4*			
58	2	*Long term secondary oxygen sensor trim, A: bank 2, B: bank 4*			
59	2	*Fuel rail absolute pressure*	0	655,350	kPa
5A	1	*Relative accelerator pedal position*	0	100	%
5B	1	*Hybrid battery pack remaining life*	0	100	%
5C	1	*Engine oil temperature*	-40	210	°C
5D	2	*Fuel injection timing*	-210.00	301.992	°
5E	2	*Engine fuel rate*	0	3276.75	L/h

Continua na próxima página

Tabela 20 – *Continuação da página anterior*

PID	Bytes	Descrição	Min	Máx	Unidade
5F	1	*Emission requirements to which vehicle is designed*			
60	4	*PIDs supported [61 - 80]*			
61	1	*Driver's demand engine - percent torque*	-125	125	%
62	1	*Actual engine - percent torque*	-125	125	%
63	2	*Engine reference torque*	0	65,535	Nm
64	5	*Engine percent torque data*	-125	125	%
65	2	*Auxiliary input / output supported*			
66	5	*Mass air flow sensor*			
67	3	*Engine coolant temperature*			
68	7	*Intake air temperature sensor*			
69	7	*Commanded EGR and EGR Error*			
6A	5	*Commanded Diesel intake air flow control and relative intake air flow position*			
6B	5	*Exhaust gas recirculation temperature*			
6C	5	*Commanded throttle actuator control and relative throttle position*			
6D	6	*Fuel pressure control system*			
6E	5	*Injection pressure control system*			
6F	3	*Turbocharger compressor inlet pressure*			
70	9	*Boost pressure control*			
71	5	*Variable Geometry turbo (VGT) control*			
72	5	*Wastegate control*			
73	5	*Exhaust pressure*			
74	5	*Turbocharger RPM*			
75	7	*Turbocharger temperature*			
76	7	*Turbocharger temperature*			
77	5	*Charge air cooler temperature (CACT)*			
78	9	*Exhaust Gas temperature (EGT) Bank 1*			
79	9	*Exhaust Gas temperature (EGT) Bank 2*			
7A	7	*Diesel particulate filter (DPF)*			
7B	7	*Diesel particulate filter (DPF)*			

Continua na próxima página

Tabela 20 – *Continuação da página anterior*

PID	Bytes	Descrição	Min	Máx	Unidade
7C	9	*Diesel Particulate filter (DPF) temperature*			
7D	1	*NOx NTE control area status*			
7E	1	*PM NTE control area status*			
7F	13	*Engine run time*			
80	4	*PIDs supported [81 - A0]*			
81	21	*Engine run time for Auxiliary Emissions Control Device(AECD)*			
82	21	*Engine run time for Auxiliary Emissions Control Device(AECD)*			
83	5	*NOx sensor*			
84		*Manifold surface temperature*			
85		*NOx reagent system*			
86		*Particulate matter (PM) sensor*			
87		*Intake manifold absolute pressure*			
A0	4	*PIDs supported [A1 - C0]*			
C0	4	*PIDs supported [C1 - E0]*			
C3	?	*?*	?	?	?
C4	?	*?*	?	?	?

Fonte: resumido da norma SAE J1979 e do Wikipédia no tópico OBD PIDs

B . CÓDIGO PYTHON

B.1 Mudança de Baudrate para ELM327

```
#Mudanca de Baudrate
import serial
import time

print "Abre serial"
ser = serial.Serial('/dev/ttyUSB0', 38400)

print "Envia ATBRD 23"
ser.write("atbrd 23\r\n")
print ser.readline()

ser.setBaudrate(115200)
print ser.readline()
ser.write("\r\n")
print ser.readline()
```

B.2 Injetor de dados para ELM327

```
#Injetor de dados
import serial
import time
#ser = serial.Serial('COM10', 38400)

entrada = open('obd-input.txt', 'r')

for line in entrada:
        print "Linha original:"
        print line
        header = line.split(" ", 1)[0]
        #ser.write("ATSH")
        #ser.write(header)
        print "ATSH " + header
        data = line[4:28]
        print "Enviado para o OBD"
        print data
        #ser.write(data)
        print "--------------------"
```

```
        #print ser.read()
        time.sleep(0.5)
```

B.3 Monitorar para ELM327

```
#Monitorar

import serial
import time

print "Abre serial"
ser = serial.Serial('/dev/ttyUSB0', 38400)

ser.write("atma\r\n")

while 1:
        a=ser.readline()
        print a
        if a.startswith('BUFF') :
                print 'recebi buffer full'
                ser.write("atma\r\n")
        if a.startswith('CAN') :
                ser.write("atl1\r\n")
                time.sleep(0.1)
                ser.write("atma\r\n")
```

C . LOG DE CÓDIGOS DA HILUX SRV MY 2015

C.1 Trecho de Log do USBTin

```
1486139318.323035) slcan0 223#002000000000004D
(1486139318.323054) slcan0 2C4#040E0021008030B1
(1486139318.328074) slcan0 020#000007
(1486139318.328090) slcan0 224#0000000000000000
(1486139318.328094) slcan0 022#020002010000002F
(1486139318.328098) slcan0 023#01FB020600002E
(1486139318.332190) slcan0 423#00
(1486139318.336317) slcan0 2C1#0801F101F1A00057
(1486139318.336329) slcan0 2D0#080C080050000046
(1486139318.346523) slcan0 020#000007
(1486139318.346540) slcan0 025#0FF70FFE787878A8
(1486139318.346545) slcan0 022#020002010000002F
(1486139318.346548) slcan0 2D2#00
(1486139318.346552) slcan0 223#002000000000004D
(1486139318.346555) slcan0 2C4#040E0021008030B1
(1486139318.354319) slcan0 020#000007
(1486139318.354330) slcan0 224#0000000000000000
(1486139318.354335) slcan0 025#0FF70FFE828282C6
(1486139318.354339) slcan0 022#020002010000002F
(1486139318.354342) slcan0 3D0#00
(1486139318.357016) slcan0 4C3#0300030200000000
(1486139318.366824) slcan0 020#000007
(1486139318.366843) slcan0 025#0FF70FFE787878A8
(1486139318.366848) slcan0 022#020002010000002F
(1486139318.366853) slcan0 023#01FB020600002E
(1486139318.366859) slcan0 2C1#0801D001D2A00017
(1486139318.366864) slcan0 2D0#080B080050000045
(1486139318.370069) slcan0 223#002000000000004D
(1486139318.370081) slcan0 2C4#040D0021008030B0
(1486139318.376319) slcan0 2D2#00
(1486139318.376329) slcan0 020#000007
(1486139318.376333) slcan0 224#0000000000000000
(1486139318.376336) slcan0 022#020002010000002F
(1486139318.376339) slcan0 023#01FE0205000030
(1486139318.388305) slcan0 020#000007
(1486139318.388318) slcan0 025#0FF70FFE787878A8
(1486139318.388322) slcan0 022#020002010000002F
```

```
(1486139318.393674) slcan0 223#002000000000004D
(1486139318.393693) slcan0 2C4#040D0021008030B0
(1486139318.400925) slcan0 2C1#0801EC01ECA0004D
(1486139318.400941) slcan0 020#000007
(1486139318.400946) slcan0 224#0000000000000000
(1486139318.400949) slcan0 2D0#0809080050000043
(1486139318.400953) slcan0 022#020002010000002F
(1486139318.400956) slcan0 023#01FB020600002E
(1486139318.405449) slcan0 2D2#00
(1486139318.412910) slcan0 020#000007
(1486139318.412921) slcan0 025#0FF70FFE787878A8
(1486139318.412925) slcan0 022#020002010000002F
(1486139318.412928) slcan0 023#01FD0206000030
```

C.2 Trecho de Log do ELM 327

```
223 00 20 00 00 00 00 00 4D <CR><LF>
2C4 04 0E 00 21 00 <CR><LF>
020 00 00 07 <CR><LF>
224 00 00 00 00 00 00 00 00 <CR><LF>
025 0F F7 0F FE 82 82 82 C6 <DATA ERROR<CR><LF>
022 02 00 02 <CR><LF>
023 01 FD <CR><LF>
3D0 00 <CR><LF>
4C3 03 00 03 02 <CR><LF>
020 00 00 07 <CR><LF>
025 0F F7 0F FE 78 78 78 A8 <DATA ERROR<CR><LF>
022 02 00 02 <CR><LF>
023 01 FB <CR><LF>
2C1 08 01 D0 01 D2 A0 00 17 <DATA ERROR<CR><LF>
2D0 08 0B 08 00 50 00 00 45 <DAT<CR><LF>
BUFFER FULL<CR><LF>
<CR><LF>
>
03/02/2017 14:27:24.20 [TX] - atma<CR><LF>

03/02/2017 14:27:24.21 [RX] - atma<CR><LF>
020 00 00 07 <CR><LF>
022 02 00 02 <CR><LF>
025 0F F7 0F FE 78 78 78 A8 <DATA ERROR<CR><LF>
023 01 FD <CR><LF>
```

```
2C4 04 09 00 21 00 <CR><LF>
223 00 20 00 00 00 00 00 4D <CR><LF>
2C1 08 01 E0 01 E0 A0 00 35 <DATA ERROR<CR><LF>
2D0 08 04 08 00 50 00 00 3E <DATA ERROR<CR><LF>
020 00 00 07 <CR><LF>
224 00 00 00 00 00 00 00 00 <CR><LF>
022 02 00 02 <CR><LF>
025 0F F7 0F FE 78 78 78 A8 <DATA<CR><LF>
BUFFER FULL<CR><LF>
<CR><LF>
>
```