UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ENGENHARIA ELETRÔNICA

BACHARELADO EM ENGENHARIA ELETRÔNICA

TAYSA MILLENA BANIK

PROCESSO DE DESENVOLVIMENTO BASEADO EM MODELO PARA SOFTWARE AUTOMOTIVO: MIGRAÇÃO PARA O PADRÃO AUTOSAR

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2017

TAYSA MILLENA BANIK

PROCESSO DE DESENVOLVIMENTO BASEADO EM MODELO PARA SOFTWARE AUTOMOTIVO: MIGRAÇÃO PARA O PADRÃO AUTOSAR

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel, do Departamento Acadêmico de Engenharia Eletrônica da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Max Mauro Dias Santos

PONTA GROSSA

2017

TERMO DE APROVAÇÃO

PROCESSO DE DESENVOLVIMENTO BASEADO EM MODELO PARA SOFTWARE AUTOMOTIVO: MIGRAÇÃO PARA O PADRÃO AUTOSAR

por

TAYSA MILLENA BANIK

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 09 de junho de 2017 como requisito parcial para a obtenção do título de Bacharel em Engenharia Eletrônica. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____

MAX MAURO DIAS SANTOS
Prof. Orientador

_____

FERNANDA CORREA
Membro titular

_____

JOÃO VILAN JR.
Membro titular

- A Folha de Aprovação assinada encontra-se arquivada na Secretaria Acadêmica -

Dedico este trabalho à Deus, minha força,
e à minha família, meu norte e sustento.

.

AGRADECIMENTOS

Sempre dizem que as ações valem muito mais do que as palavras. Isso é verdade. Mas registrar todo a minha gratidão à algumas pessoas que contribuíram com a minha chegada até aqui é fundamental. Se cheguei até o fim dessa caminhada é porque tive pessoas maravilhosas ao meu redor.

Agradeço a Deus por me dar a vida, a esperança e a força necessária para viver cada dia, sabendo que todas as coisas, boas ou ruins, estavam debaixo do Seu controle.

Agradeço aos meus pais, por me darem uma direção quando tudo ficou escuro e por me ensinarem a caminhar com as próprias pernas.

Agradeço ao meu amado Andre, por revisar tudo, com seu perfeccionismo e deixar tudo ainda mais perfeito. Por me dar ideias e conforto quando precisei.

Agradeço ao meu Professor Orientador Dr. Max Mauro Dias Santos por sempre trazer novas ideias e desafios. Se aprendi algo de valor é porque você me instigou para isso, contribuindo com meu crescimento.

Agradeço ao meu Co-orientador, Mestre João Henrique Zander Neme, por sempre me acompanhar em tudo. Por ser a ajuda sempre presente e amigo para todas as horas. Aprendi muito debaixo da sua orientação.

Agradeço ao meu amigo e colega de trabalho, Felipe Franco, por ser paciente e por me ensinar várias vezes as mesmas coisas. Por ser o melhor help que a MathWorks poderia oferecer.

Agradeço aos meus colegas do Grupo de Sistemas Automotivos por todas as ideias e opiniões, pelos momentos de descontração e alegria, bem como pelo enorme conhecimento compartilhado comigo. Levarei cada um de vocês dentro do coração, onde quer que for.

E por fim, para não esquecer de ninguém, agradeço a todos que de alguma forma foram importantes no decorrer do desenvolvimento desse trabalho.

RESUMO

BANIK, Taysa Millena**. Processo de Desenvolvimento Baseado em Modelo Software Automotivo: Migração para o Padrão AUTOSAR**. 2017. 68 f. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

O desenvolvimento de novas funções automotivas tem sido pautadas em versatilidade, rapidez e robustez. Com o consumidor exigindo cada vez mais conforto e inovação, os desenvolvedores precisam estar atentos a cada possibilidade de mudança, a fim de alcançar novos patamares na programação. Este trabalho visa aplicar o *Model Based Design*, desenvolvimento baseado em modelos que fornece diversas vantagens para a criação de novos sistemas. A metodologia será aplicada na construção de uma plataforma móvel, idealizada inicialmente pela Marcopolo. Um conjunto de testes de verificação será realizado a fim de aumentar a robustez do sistema e torna-lo compatível com a ISO26262, uma norma que trata da segurança funcional de software embarcado. Por fim, migrar o sistema para a arquitetura automotiva AUTOSAR com o intuito de simular seu funcionamento. Esse fluxo de trabalho vai comprovar a eficácia destes métodos no objetivo de desenvolver softwares automotivos de maneira mais eficiente e utilizando menos recursos.

Palavras-chave: Software Automotivo. Model Based Design. ISO26262. AUTOSAR.

ABSTRACT

BANIK, Taysa Millena. **Development Process in Model Based Design for Automotive Software: Migration to AUTOSAR Standard.** 2017. 68 p. Work of Conclusion Course (Graduation in Electronics Engineering) - Federal Technology University - Paraná. Ponta Grossa, 2017.

The development of new automotive functions has been based on versatility, speed and robustness. With the consumer increasingly demanding comfort and innovation, developers need to be aware of every possibility of change in order to reach new heights in programming. This work aims to apply Model Based Design, model-based development that provides several advantages for the creation of new systems. The methodology will be applied in the construction of a mobile platform, idealized initially by Marcopolo. A set of verification tests will be carried out in order to increase the robustness of the system and make it compatible with ISO26262, a standard dealing with the functional safety of embedded software. Finally, to migrate the system to the automotive architecture AUTOSAR in order to simulate its operation. This workflow will prove the effectiveness of these methods in the goal of developing automotive software more efficiently and using fewer resources.

Keywords: Automotive Software. Model Based Design. ISO26262. AUTOSAR.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ABS	Antilock Braking System
AUTOSAR	Automotive Open System Architecture
BCM	Módulo de Controle da Carroceria
BSW	Basic Software
CAN	Controller Area Network
EBCM	Módulo Eletrônico de Controle de Freios
ECM	Módulo de Controle do Motor
ECU	Electronic Control Unit
EVB	Evaluation board
HIL	Hardware-In-the-Loop
IDE	Integrated Development Environment
IEC	International Electrotechnical Commision
ISO	International Organization Standardization
MBD	Model-Based Design
MCU	Microcontrolador
MIL	Model-In-the-Loop
MSP	Mobile Seat Platform
OEM	Original Equipment Manufacturer
PCM	Módulo de Controle de Powertrain
PIL	Processor-In-the-Loop
PWM	Pulse Width Modulation
RPM	Rotations per Minute
RTE	Runtime Environment
SIL	Software-In-the-Loop
SWC	Software Component
VFB	Virtual Functional Bus

SUMÁRIO

1 INTRODUÇÃO

Não há como imaginar um mundo sem tecnologia, o dia a dia das pessoas está tão atrelado com as várias vantagens que ela traz que não há como desvencilhar-se. São desde equipamentos simples como eletrodomésticos até sistemas complexos como o sistema de controle de aeronaves.

Todo o funcionamento por trás da tecnologia é desenvolvido por codificação. São milhares de linhas de código de linguagem, C/C++ em sua grande maioria, dispostas em lógicas que realizam as mais diversas funções.

Tratando de forma mais específica o setor automotivo, é possível conferir a grande variedade de aplicações que um carro pode oferecer. Aplicações simples como o *airbag* e o freio ABS que tem como o objetivo aumentar a segurança, como também aplicações de auxílio ao motorista, como o *Intelligent Parking Assist System,* sistema inteligente de assistência de estacionamento, o *Lane Departure Warning System*, sistema de aviso e detecção de faixa, o *Braking Assistance System*, sistema de assistência na frenagem e diversos outros que necessitam de programação de alta complexidade.

Imaginar a estruturação e desenvolvimento de toda a codificação feita manualmente e por linhas de código não é uma tarefa simples. A criação de uma nova aplicação levaria anos para ser finalizada indo na contramão do grande avanço da indústria, que sempre pede por agilidade.

A crescente busca por novas aplicações, automóveis que combinem versatilidade e tecnologia com conforto e segurança gerou uma demanda de produção cada vez maior, ou seja, o *time-to-market*, tempo de desenvolvimento e lançamento de um produto, ficou bem menor do que era antes. Isso ficou claro pelo estudo de Ploss (2008) que diz que a produção de veículos aumentou 44% entre 1997 e 2008.

O *Model-Based Design*, desenvolvimento baseado em modelos é uma metodologia de desenvolvimento de software e aplicações. Seu uso por grandes desenvolvedores tem sido crescente por oferecer uma série de vantagens que a construção comum não possui, dentre eles a possiblidade de fazer testes de verificação e validação do software através de simulações. Essa abordagem reduz custos, tempo de desenvolvimento e aumenta a qualidade do produto (Dillaber, 2010).

Aliada a esse desenvolvimento, tem-se o AUTOSAR que é uma arquitetura de construção de software automotivo, padronizada e aberta, desenvolvida conjuntamente por fabricantes de automóveis, fornecedores e desenvolvedores de ferramentas para a unidade de controle eletrônica (ECU) presente em todos os sistemas automotivos.

A associação do *Model-Based Design* com o AUTOSAR significa desenvolver uma nova aplicação automotiva, software, com muito mais versatilidade, robustez e rapidez, sem se ater a detalhes do hardware que será utilizado.

Este trabalho tem como objetivo apresentar resultados utilizando a metodologia do MBD, os processos pertinentes, as ferramentas utilizadas e o padrão de arquitetura AUTOSAR, no desenvolvimento e teste de uma função de software embarcado automotivo.

1.1 TEMA

O presente trabalho aborda o projeto de desenvolvimento baseado em modelo de um sistema de controle de uma poltrona móvel de um ônibus de viagens com a migração do mesmo para arquitetura AUTOSAR.

1.2 DELIMITAÇÃO DO TEMA

No presente trabalho é aplicada a metodologia de *Model-Based Design*, para o desenvolvimento de um sistema de controle de uma poltrona móvel criada para deficientes físicos, instalada em ônibus de viagens. O modelo dessa plataforma foi realizado em parceria com a Marcopolo, fabricante de carrocerias de ônibus.

Apesar da parceria, todo o fluxo de trabalho de melhorias e verificação foi realizado pela autora, utilizando-se das etapas pertencentes ao desenvolvimento baseado em modelos, *Model-In-the-Loop* (MIL) e *Software-In-the-Loop* (SIL), e das ferramentas que o software MATLAB&Simulink® oferece, com a verificação e conformidade com a ISO 26262 – *Road Vehicles – Functional Safety*.

Após o desenvolvimento e testes do sistema, o modelo passará pela migração para a arquitetura AUTOSAR. Esta migração é possível por meio da ferramenta Embedded Coder pertencente ao conjunto MATLAB&Simulink®.

E por fim, a verificação do funcionamento do sistema da poltrona, agora em AUTOSAR, através de testes no software SystemDesk®.

1.3 PROBLEMA

Todo o funcionamento de uma aplicação veicular é controlado por um sistema feito por codificação. O método tradicional de desenvolvimento de um software, através de códigos textuais, podem levar tempo e desprendem muito trabalho. Tornando o antigo método ineficiente e inapropriado para o mercado automotivo em expansão.

O desenvolvimento baseado em modelos, a grande alternativa da programação, apesar de ser conhecida e bastante utilizada, encontra pouco espaço no ambiente acadêmico da Engenharia Eletrônica.

O padrão AUTOSAR, por mais que seja um padrão aberto e utilizado por grandes desenvolvedores, possui pouco material de referência e menos ainda aplicações explicativas, no que diz respeito a materiais brasileiros, o que torna este trabalho um possível referencial. O conhecimento adquirido pela autora através de pesquisas e criação de sistemas, bem como pelo desenvolvimento deste trabalho, fornece material de estudo para possíveis interessados

1.4 PREMISSAS

Para o desenvolvimento deste trabalho foi considerado o uso do software MATLAB&Simulink® pela disponibilidade e pelas inúmeras vantagens que ele apresenta como plataforma de testes de verificação e conformidade com diversas normas.

Durante a criação do sistema de controle não foi considerado nenhuma característica física do ônibus, dos motores de acionamento da plataforma e do possível passageiro a utilizar a poltrona. Foram considerados somente os sinais registrados na rede CAN de velocidade, freio e portas, tanto do passageiro quanto da porta principal, fornecidos pela unidade de comando eletrônico, *Electronic Control Unit* (ECU), da carroceria. Esses sinais registrados na rede CAN foram considerados como sinais booleanos unitários, a fim de diminuir as características a serem consideradas

no momento da construção do modelo. Entretanto, é possível modificar tais sinais e incluir o empacotamento e desempacotamento de *frame*, na utilização desses sinais no modelo.

1.5 OBJETIVOS GERAIS

O presente trabalho tem por objetivo geral demonstrar o controle para o sistema de poltrona móvel, mobile seat platform (MSP), utilizando as metodologias de desenvolvimento baseado em modelos (MBD) e a padronização do mesmo em AUTOSAR.

1.6 OBJETIVOS ESPECÍFICOS

Estão listados a seguir todos os objetivos específicos que se desejam alcançar no decorrer do desenvolvimento do sistema.

a) Definir como funciona o desenvolvimento baseado em modelos, MBD, as etapas de verificação e testes de conformidade e a arquitetura AUTOSAR;
b) Definir os requisitos de funcionamento do sistema de controle;
c) Modelar o sistema de controle;
d) Cumprir com as etapas de desenvolvimento baseado em modelo para o sistema automotivo da plataforma;
e) Verificar o sistema através de métodos formais – MIL, SIL e ISO26262;
f) Migrar o sistema automotivo para o padrão AUTOSAR e realizar a simulação de verificação no software SystemDesk®.

1.7 JUSTIFICATIVA

Com o mercado automotivo em constante expansão e a exigência dos usuários cada vez maior por tecnologia de ponta, versatilidade, conforto e segurança. A necessidade de desenvolver sistemas mais robustos e seguros aliado com a redução de custos e tempo de trabalho tornou-se uma premissa importante para as montadoras, fazendo com que novas metodologias sejam usadas.

O entendimento dessas metodologias, em especial, o desenvolvimento baseado em modelos e a arquitetura AUTOSAR, que estabelece um padrão para desenvolvimento de software automotivo, contribui para a difusão do conhecimento e de como aplicá-lo na engenharia.

1.8 PROCEDIMENTOS METODOLÓGICOS

O projeto em questão trata-se do desenvolvimento de um controle para uma plataforma de poltrona móvel através de métodos e ferramentas adequadas e padronizadas.

Inicialmente serão definidos os requisitos para o funcionamento do sistema de controle da plataforma, os sinais de entradas – sensores, fins de curso, sinais da CAN- as possíveis saídas, como motores, sinais para o painel do motorista, etc. A partir desses requisitos, o sistema de controle é modelado. Para isso será utilizada uma ferramenta computacional específica da MathWorks, como o MATLAB&Simulink® e o Stateflow®.

Após a construção do modelo, serão definidos sinais de entrada para o sistema. Eles serão utilizados nas etapas de verificação *Model-In-the-Loop* (MIL) e *Software-In-the-Loop* (SIL), como também na verificação do sistema quando em padrão AUTOSAR. Por meio deles, será possível observar os sinais de saída do sistema. Estes sinais não serão gerados aleatoriamente, eles serão criados a fim de comprovar que não há nenhum erro de lógica no desenvolvimento do sistema.

Logo após o teste MIL, haverá o teste de conformidade com a ISO26262. Através da ferramenta Design Verifier® pertencente ao MATLAB&Simulink®, será possível cumprir as premissas da norma para o modelo do projeto.

Com o auxílio da ferramenta Embedded Coder®, também pertencente ao MATLAB&Simulink®, migraremos o código C gerado automaticamente para o padrão AUTOSAR e suas extensões.

E por fim, com a ferramenta computacional SystemDesk®, a última verificação será feita, com as mesmas entradas geradas para a fase de MIL e SIL, agora para o sistema no padrão de interesse.

1.9 ORGANIZAÇÃO DO TRABALHO

Este trabalho está subdividido em três partes principais: o referencial teórico apresentado no segundo capítulo, toda a seção de desenvolvimento mostrada no terceiro capítulo e os resultados detalhados pelo quarto capítulo, onde é possível ver os testes e etapas de verificação em conformidade com os requisitos levantados previamente.

No Capítulo 2, intitulado por Referencial Teórico e Aplicações Iniciais, são dadas todas as informações necessárias ao desenvolvimento, tais como todo o fluxo de trabalho do Model-Based Design e as ferramentas utilizadas e o funcionamento da arquitetura AUTOSAR e a ferramenta de testes utilizada para verificar o sistema proposto.

Após todas as definições apresentadas no Capítulo 2, o Capítulo 3, intitulado como Desenvolvimento, apresenta todos os procedimentos adotados pela autora do trabalho, as etapas de verificação realizadas, MIL e SIL, e o motivo de não haver as verificações *Processor-In-the-Loop* (PIL) e *Hardware-In-the-Loop* (HIL); como também a migração para o padrão AUTOSAR.

Uma vez desenvolvido, o trabalho foi testado na plataformas Simulink e SystemDesk® para o sistema em MBD e AUTOSAR, respectivamente, e teve a geração de todas as saídas e entradas registradas a fim de comparação, todas estão apresentadas no Capítulo 3, junto com as etapas de desenvolvimento.

2 REFERENCIAL TEÓRICO E APLICAÇÕES INICIAIS

Neste capítulo, serão abordados alguns conceitos relevantes e fundamentais ao desenvolvimento da pesquisa em relação às tecnologias envolvidas, explorando metodologias, equipamentos, componentes e também softwares utilizados.

Os sistemas eletrônicos automotivos são formados por diversas unidades eletrônicas de controle, ECU, que, como seu nome já indica, controlam todo o sistema. É um dispositivo eletrônico embarcado, um computador digital, que lê sinais enviados por sensores instalados em diversas partes do veículo e em diferentes componentes do automóvel e usa essas informações para controlar diversas unidade importantes e operações automatizadas do automóvel. É importante ressaltar que o termo Unidade Eletrônica de Controle é um termo genérico, isso pelo fato de que existem diversos tipos de ECU's, como pode ser visto a seguir:

- ECM – Módulo de Controle do Motor;
- EBCM – Módulo Eletrônico de Controle de Freios;
- PCM – Módulo de Controle do Powertrain;
- BCM – Módulo de Controle da Carroceria

Esses são somente alguns exemplos de algumas ECU's. Existem muito mais, inclusive dentro de cada categoria citada. Em cada ECU são ligados diversos sensores e atuadores que realizam alguma aplicação dentro do veículo.

Uma ECU possui alguns elementos fundamentais:

- Fonte de alimentação – digital e analógica (alimentação para os sensores analógicos);
- MPU – microprocessador e memória (normalmente Flash e RAM);
- Link de comunicações – barramento;
- Entradas discretas – entradas de comutação on/off;
- Entradas de frequência – sinais de encoder;
- Entradas analógicas – sinais de feedback de sensores;
- Saídas de comutadores – saídas de comutação do tipo on/off;
- Saídas PWM – frequência e ciclo de trabalho variáveis;
- Saídas de frequência – ciclo de trabalho constante;

- Software de Controle – Baseado nas informações de entrada, um algoritmo ou estratégia de controle implementado em software determina a ação que precisa ser realizada controlando ou regulando as saídas do veículo.

A unidade eletrônica de controle é o cérebro do funcionamento de um veículo. Agora, passa-se para outro ponto importante: como essas unidades são conectadas e se comunicam umas com as outras.

2.1 O PROTOCOLO DE REDE AUTOMOTIVA CAN

Essas unidades são conectadas através de uma rede de comunicação chamada barramento CAN, *Controller Area Network,* e dentro desses barramentos, as unidades trocam informações. A rede CAN foi desenvolvida por Robert Bosch em 1996, veio para tentar solucionar o problema da grande quantidade de cabos presentes em um veículo. No início, ela era usada apenas em ônibus e caminhões, sendo posteriormente utilizada em veículos automotivos, navios, tratores e na indústria (Kvaser).

A CAN é um protocolo de comunicação serial síncrono. O sincronismo entre os módulos conectados à rede é feito em relação ao início de cada mensagem lançada ao barramento (evento que ocorre em intervalos de tempo conhecidos e regulares).

Trabalha baseado no conceito multimestre, onde todos os módulos podem se tornar mestre em determinados momento e escravo em outro, além de suas mensagens serem enviadas em regime *multicast*, caracterizado pelo envio de toda e qualquer mensagem para todos os módulos existentes na rede.

Outro ponto positivo deste protocolo é o fato de ser fundamentado no conceito CSMA/CD with NDA (Carrier Sense Multiple Access/ Collision Detection with Non-Destructive Arbitration). Isto significa que todos os módulos verificam o estado do barramento, analisando se outro módulo está ou não enviando mensagens com maior prioridade. Caso isto aconteça, o módulo cuja mensagem tiver menor prioridade cessará sua transmissão e o de maior prioridade continuará enviando sua mensagem deste ponto, sem ter que reiniciá-la.

As mensagens são compostas por uma sequência de 1 e 0, que são transmitidos por um valor de tensão específico e constante. Os bits ‘0’ são dominantes e os bits ‘1’ são recessivos.

A velocidade de transmissão dos dados é inversamente proporcional ao comprimento do barramento. Isso significa que quanto maior for o barramento, menor vai ser a velocidade de transmissão alcançada. Para um barramento de 40m, por exemplo, a velocidade é de 1Mbps.

O meio de transmissão dos dados são fios elétricos. Existem 3 modos de construir um barramento CAN, dependendo da quantidade de fios utilizados. Quando se trata de somente 1 fio (linha CAN), trata-se apenas do fio de dados, considerando a referência vindo da carcaça do equipamento. Com 2 e 4 fios o barramento fica mais confiável. As redes de 2 e 4 fios trabalham com os sinais de dados CAN_H (CAN high) e CAN_L (CAN low). No caso do barramento com 4 fios, além dos sinais de dados, tem-se também um fio VCC (alimentação) e outro GND (referência). Nos casos de 2 ou 4 fios, os cabos são trançados e não blindados, o que impedem interferências eletromagnéticas.

Existem dois tipos de mensagens que podem ser utilizadas em barramentos CAN: CAN 2.0A ou CAN 2.0B.

CAN 2.0A: mensagens com identificador de 11 bits. É possível ter até 2048 mensagens em uma rede constituída sob este formato, o que pode limitar a rede em determinadas aplicações.

CAN 2.0B: mensagens com identificador de 29 bits. É possível ter, aproximadamente, 537 milhões de mensagens em uma rede constituída sob este formato. Esse aumento de 18 bits no identificador aumentam as possibilidades de mensagens, mas também aumentam o tempo de transmissão de cada mensagem, o que pode ocasionar problemas em aplicações que trabalhem em tempo-real (problema conhecido como *overhead*).

A comunicação CAN de ônibus, como o a aplicação a ser desenvolvida neste trabalho, trabalham com o protocolo J1939, criado pela *Society of Automotive Engineers*. Ele trata de comunicação e diagnóstico em redes CAN para veículos pesados. Esse protocolo trabalha com uma mensagem de 29 bits no identificador e 8 bytes no total.

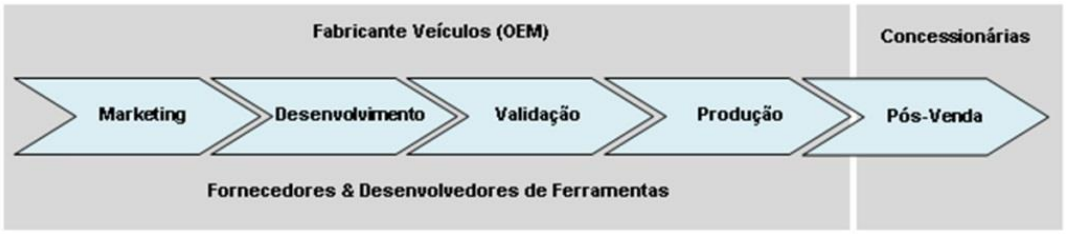
Como o objetivo desse trabalho é o desenvolvimento de um sistema baseado em modelos e a arquitetura AUTOSAR, optou-se por simplificar a comunicação entre ECU's modificando a mensagem de 8 bytes para somente 1bit de habilitação.

2.2 VISÃO GERAL DO DESENVOLVIMENTO DE SOFTWARE AUTOMOTIVO

A demanda de desenvolvimento de softwares automotivos nasceu da necessidade, a princípio, de prover mais segurança para os veículos automotivos, tanto ônibus e caminhões quanto veículos de passeio. A partir daí, novas funções foram sendo criadas para fornecer mais conforto e versatilidade.

De acordo com Neme (2017), o processo de um produto automotivo possui 5 fases: marketing, desenvolvimento, validação, produção e pós-venda. Os fabricantes de automóveis (OEM), fornecedores e desenvolvedores de ferramentas precisam estar alinhados quanto à criação e integração de todo o projeto. A figura 1 apresenta as fases de desenvolvimento de produto automotivo.

Figura 1 - Fluxo de Desenvolvimento de Produto Automotivo



Fonte: Neme, 2017.

Para que ocorra esse desenvolvimento, existem metodologias e tecnologias em todas as fases citadas acima. Aprimorar essas metodologias é prioridade para o setor automotivo. A passagem da codificação em C/C++ para o desenvolvimento baseado em modelos é uma amostra desse avanço.

As primeiras menções ao tema de desenvolvimento de software no setor automotivo foram feitas em 1990 (Tran, et al, 1990) e em 1997 foi lançada umas das mais utilizadas ferramentas no processo de criação: o Stateflow®. Por meio dele, é possível criar o controle de um sistema de forma intuitiva, sem ambiguidades, fazer a verificação do que está sendo criado e observar toda a programação de forma mais visual, através de blocos.

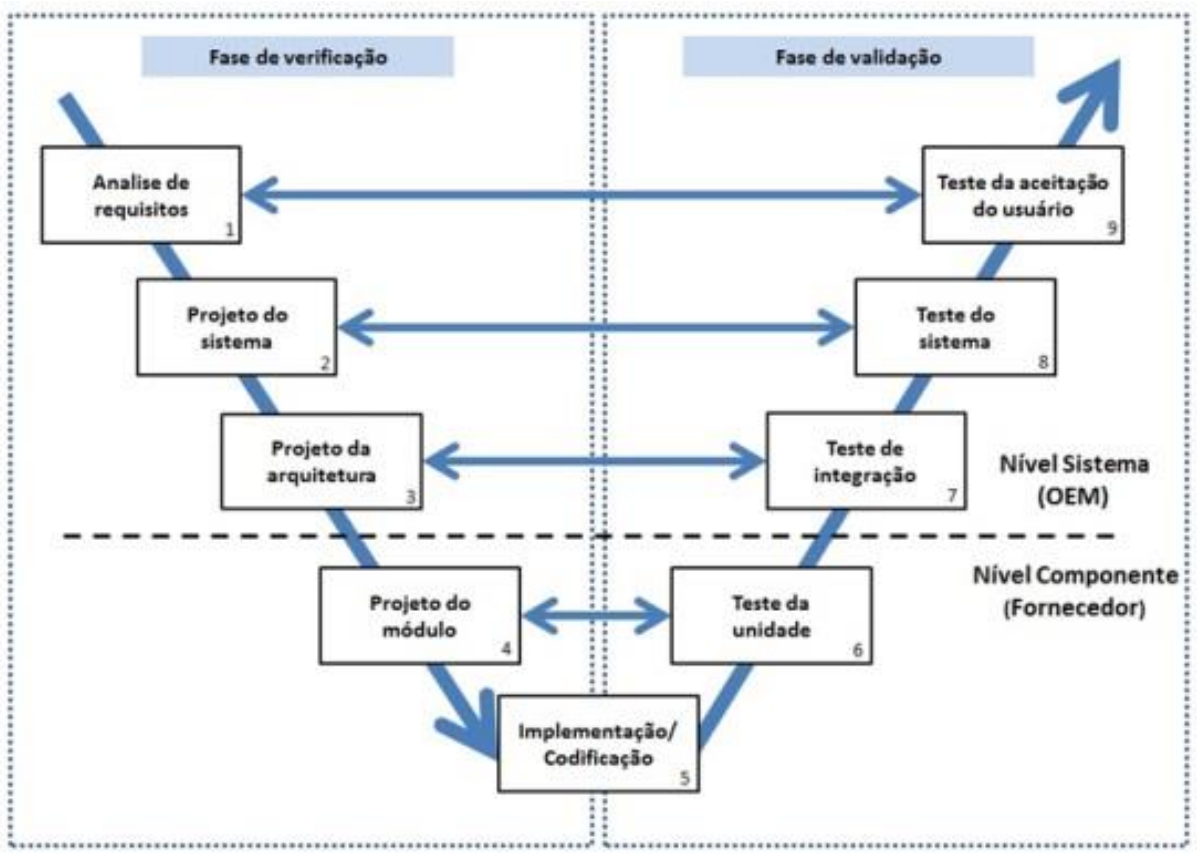
O software MATLAB&Simulink® e a ferramenta Stateflow® da MathWorks é ideal para todo o processo que o *Model-Based Design* propõe. A partir dele é possível levantar os requisitos e ligá-los a cada bloco do modelo, criar de forma intuitiva, verificar o que está sendo feito em diversos níveis, atribuindo robustez e gerar o código automático para a plataforma desejada. Esse processo será especificado no capítulo 3, onde todo o desenvolvimento do sistema será explicado.

O próximo tópico será a abordagem do *Model-Based Design*.

2.3 *MODEL-BASED DESIGN*

A base para todo o desenvolvimento baseado em modelos é o V-Model, ou ciclo V. O ciclo V já vem sendo utilizado e possui maturidade. A figura 2 mostra as atividades e fluxo desse ciclo.

Figura 2 - Diagrama do Ciclo V



Fonte: Stella (2015).

O início do processo se dá a partir do levantamento de requisitos, no lado esquerdo do ciclo, para isso, faz-se a análise minuciosa do sistema a ser criado a fim

de levantar as características principais, tipo de acionamento, as entradas e saídas do sistema. A criação do sistema em MBD vem logo após com o projeto e implementação caracterizada pela codificação. Do lado direito está toda a verificação e validação do sistema utilizados durante a criação. A verificação e validação são etapas que ocorrem durante todo o ciclo a fim de detectar erros e aumentar a qualidade do produto a ser criado.

2.3.1 ETAPAS DE VERIFICAÇÃO E VALIDAÇÃO

Dentro do ciclo, existem duas definições de testes que o entendimento se faz necessário: o conceito de verificação e validação, Ramos (2011) explana os conceitos. O conceito de verificação é assegurar consistência, completitude e corretitude do produto em cada fase e entre fases consecutivas do ciclo de vida do software, de modo que o criado indaga-se: estamos construindo o produto corretamente? Já o conceito de validação é ligado a assegurar que o produto final corresponda aos requisitos do usuário, com a indagação: estamos construindo o produto certo?

Com o conceito e processo de X-In-the-Loop, a verificação e a validação se adequam as etapas de desenvolvimento.

2.3.2 *MODEL BASED DESIGN* E *X-IN-THE-LOOP*

Como foi citado anteriormente a indústria automotiva necessita de processos de desenvolvimento que demandem menos tempo, trabalho e recursos e produza sistemas de qualidade, segurança e robustez. O desenvolvimento baseado em modelo é uma metodologia que pode suprir essa necessidade e cortar custos, como também diminuir o tempo de desenvolvimento.

Isso tudo por possuir a característica de desenvolvimento em uma única plataforma, ao permitir que seja criado tanto a planta quanto o controlador da planta utilizando a mesma ferramenta computacional. No projeto em questão, a ferramenta é o MATLAB&Simulink® da MathWorks®.

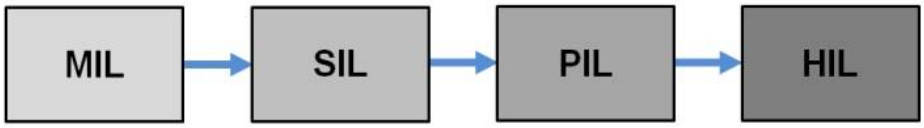
De acordo com a desenvolvedora MathWorks®, no desenvolvimento baseado em modelos, o modelo de um sistema está no foco do desenvolvimento. Desde o levantamento de requisitos, passando pela criação, implementação e por fim os testes. O MBD está transformando o modo que os engenheiros trabalham, tirando as tarefas do projeto do laboratório e do campo para uma área de trabalho.

Lennon (2007) diz ainda que o MBD simplifica o desenvolvimento de sistemas, cirando um ambiente comum para design e comunicação entre diferentes áreas de engenharia, dando margem para a interdisciplinaridade e aumentando a completitude do projeto como um todo.

Dentro do desenvolvimento baseado em modelos encontra-se a metodologia de testes baseados em simulação (Simulation Based Testing) (Iwagaya, 2013), de onde deriva o conceito de desenvolvimento X-In-the-Loop. Esse método possui técnicas de verificação em que para cada estágio têm-se definidos as etapas do projeto a serem seguidas.

Os estágios de verificação são Model-In-the-Loop (MIL), Software-In-the-Loop (SIL), Processor-In-the-Loop (PIL) e Hardware-In-the-Loop (HIL), obedecendo essa sequência, como mostra a figura 3.

Figura 3 - Sequência de Estágios para XIL



Fonte: Neme (2014).

Cada um desses estágios possui finalidades diferentes, segundo Neme (2017):

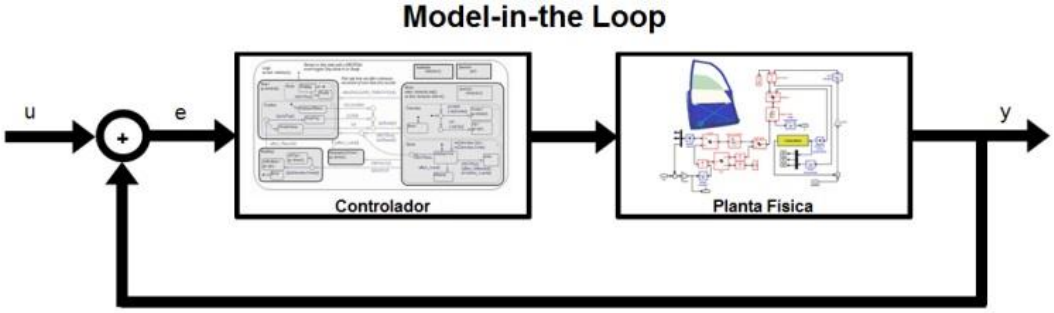
- MIL – Especificação e projeto do controlador funcional, a construção do modelo;
- SIL – Teste do gerador e da função em código C ou C++;
- PIL – Geração do código executável para testes no microcontrolador usado na ECU;
- HIL – Teste do controlador já embarcado na ECU em que já possui os componentes de entrada/saída, em que a ECU pensa já estar controlando a planta física.

2.3.2.1 *Model-In-the-Loop* (MIL)

É o primeiro estágio de simulação e serve como base para os estágios seguintes. Nesse estágio desenvolve-se a estratégia de controle e o modelo físico do sistema. Nessa fase as variáveis são criadas e utilizadas durante todo processo. Esse desenvolvimento costuma ser rápido, possibilitando mudanças e testes. No presente trabalho essa fase aconteceu no ambiente MATLAB&Simulink®.

A figura 4 mostra o diagrama de blocos do estágio MIL, onde o controlador é o sistema automotivo a ser criado, e planta física onde ele vai atuar.

Figura 4 - Diagrama de Blocos Estágio Model-In-the-Loop



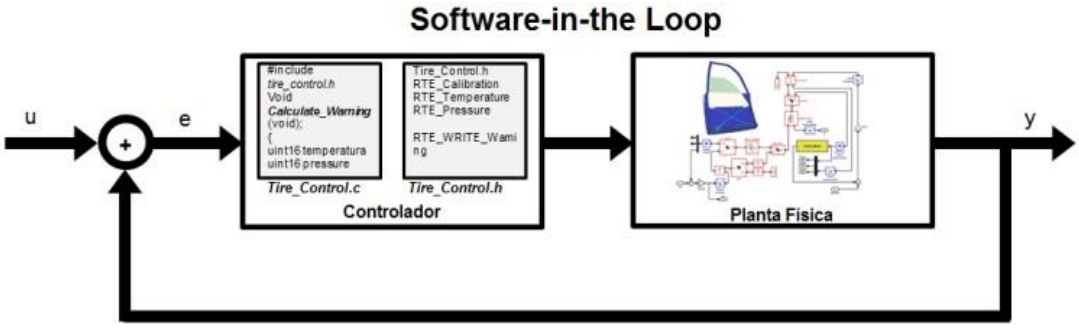
Fonte: Neme (2017).

2.3.2.2 *Software-In-the-Loop* (SIL)

Segundo estágio de simulação, no qual a simulação abrange o código C/C++ do sistema criado na fase anterior. Para obter o código C do sistema, a ferramenta computacional MATLAB&Simulink® se utiliza do Simulink Coder®, pertencente ao pacote de bibliotecas do software, para a geração automática do código. Nessa fase, poucas mudanças podem ser feitas, considerando que agora o sistema já passou por codificação. O controlador criado é mais próximo do real. É importante salientar, que os sinais de entrada utilizados na fase MIL são utilizados na fase SIL, de modo a verificar o que foi criado em blocos.

A figura 5 mostra a dinâmica de funcionamento do estágio SIL dentro da plataforma MATLAB&Simulink®. O controle é um arquivo codificado em C e a planta física continua sendo um modelo do Simulink®.

Figura 5 - Diagrama de Blocos Estágio Software-In-the-Loop



Fonte: Neme (2017).

2.3.2.3 *Processor-In-the-Loop* (PIL)

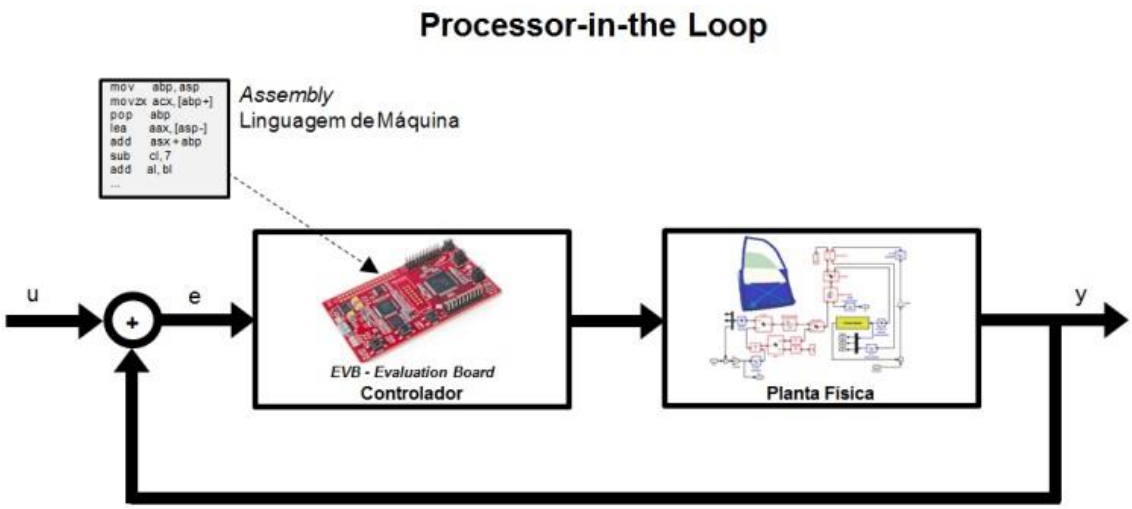
Este teste procede ao SIL, e já envolve o hardware-alvo. O modelo do sistema é traduzido em código de máquina e inserido no processador do hardware que também pode ser uma *evaluation board* (EVB), uma placa com microcontroladores para programação.

A planta é simulada em uma plataforma externa e a comunicação entre ambos não envolve periféricos, ela é feita via comunicação serial ou semelhante.

O objetivo desse estágio é verificar se o algoritmo está sendo executado da forma correta no processador. O processamento ainda não é realizado em tempo real.

A planta física do sistema ainda é um diagrama de blocos do Simulink®. A figura 6 mostra o diagrama de blocos da fase PIL.

Figura 6 - Diagrama de Blocos Estágio Processor-In-the-Loop



Fonte: Neme (2017).

2.3.2.4 *Hardware-In-the-Loop* (HIL)

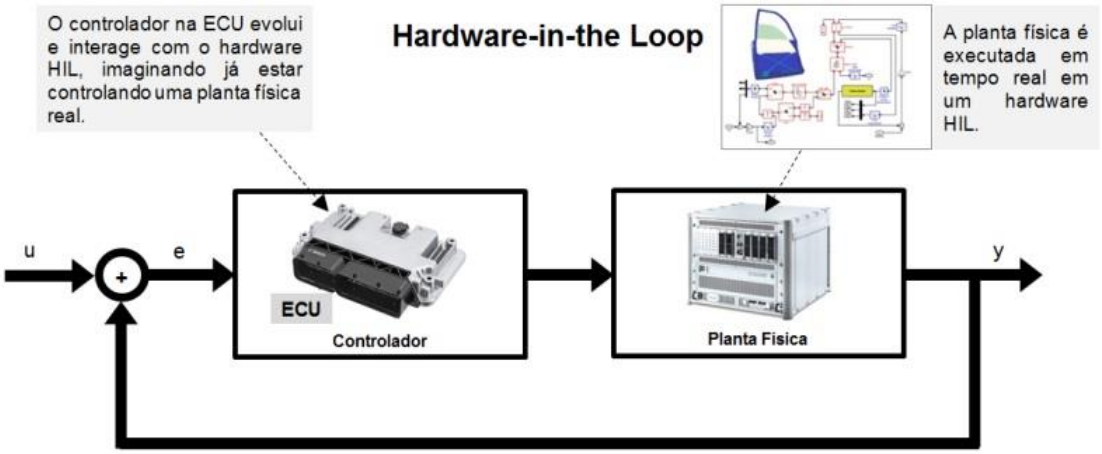
Neste estágio o sistema de controle está instalado no hardware final e pode interagir com a planta através das entradas apropriadas do controlador. No caso de um sistema automotivo o controle é embarcado em uma ECU e a planta física é executada em tempo real em um *hardware* HIL.

Isso acontece para que o sistema de controle embarcado na ECU possa funcionar num sistema o mais próximo possível do sistema real.

Considerando as fases de verificação e validação, o HIL é o estágio que gera validação para o sistema criado.

A figura 7 mostra o diagrama de blocos do estágio de *Hardware-In-the-Loop* com o controle e a planta física representado pela ECU e pela planta real/ protótipo.

Figura 7- Diagrama de Bloco Estágio Hardware-In-the-Loop



Fonte: Neme (2017).

2.3.3 TESTES DE *COMPLIANCE*

Testes de compliance não fazem parte diretamente do *Model-Based Design*, porém podem ser implementados durante o processo de desenvolvimento de um sistema. Estes testes estão relacionados com a adequação do algoritmo segundo as normas e padrões de qualidade. A ferramenta computacional MATLAB&Simulink® possui estes testes no ambiente de desenvolvimento, assim toda a adequação acontece ainda no estágio *Model-In-the-Loop*.

No presente projeto, haverá a adequação do modelo com a norma ISO 26262.

2.3.3.1 ISO 26262

A norma ISO 26262 foi criada pela *International Organization for Standardization (*ISO*)* como adaptação da norma IEC 61508, criada pela *International Electrotechnical Commision* (IEC), e que trata de segurança funcional. A ISO26262 trata de segurança funcional especificamente automotiva para sistemas embarcados.

Características de segurança funcional devem ser parte integrante de todo o desenvolvimento automotivo, desde a especificação de requisitos e funcionamento, concepção, implementação, verificação, validação e lançamento do produto.

A ISO 26262 é um padrão a partir do qual situações operacionais perigosas são avaliadas qualitativamente e as medidas de segurança são definidas para evitar ou controlar falhas de sistema e para detectar e/ou controlar falhas de hardware ou diminuir seus efeitos.

Os objetivos da ISO26262 são prover um ciclo de segurança automotiva, abrangendo todas as etapas do desenvolvimento e que suporte mudanças e adaptação nas etapas; e prever requisitos para a validação e medidas de confirmação para garantir um nível suficiente e aceitável de segurança.

A norma possui um vocabulário próprio que trata de alguns termos importantes:

- Falha: condição anormal que pode causar a falha de um elemento (hardware) ou em um item (sistema);
- Erro: discrepância entre um valor ou condição calculada, observada ou medida e o valor ou condição verdadeira, especificada ou teoricamente correta;
- Defeito: término da capacidade de um elemento executar uma função conforme necessário. A especificação do elemento define sua função requerida, logo o padrão reconhece a especificação incorreta como uma fonte potencial de falha;
- Segurança Funcional: ausência de riscos devido a perigos causados por mau funcionamento de sistemas elétricos ou eletrônicos.

2.3.3.2 ISO 26262 e Simulink®

Simulink®, como já foi citado anteriormente, é uma plataforma de desenvolvimento e simulação pertencente ao MATLAB®. Nela é possível, a partir dos requisitos de um sistema, fazer toda a modelagem em MBD.

Além da criação de um modelo, é possível gerar o código C desse modelo automaticamente. Outras ferramentas estão disponíveis dentro da plataforma Simulink®; sendo que a mais importante para a verificação de acordo com a ISO26262 é o Simulink Design Verifier®.

O *Simulink Design Verifier*® é uma ferramenta que utiliza métodos formais de verificação para encontrar erros de desenvolvimento em modelos sem a necessidade

de diversas simulações. Ele é capaz de detectar blocos que resultam em lógicas mortas, violações de requisitos, etc. A partir dessa ferramenta, é possível ter suporte ao padrão industrial através do Kit de Certificação IEC, que atende ao padrão IEC 61508 e ISO26262 que será utilizado.

Esse Kit de Certificação IEC trata-se de um instrumento de qualificação. E a partir das diretrizes que ele traz que o desenvolvedor pode garantir um modelo que siga as orientações da ISO. Isso pelo fato de que esse kit gera matrizes de teste para todas as questões pertinentes à segurança do sistema.

A verificação do Kit possui 27 pontos a serem vistos que facilitam a concepção e resolução de problemas no modelo e no código correspondente para que a aplicação desejada cumpra a norma ISO. Esses pontos podem ficar divididos, ao realizar a verificação, em três categorias:

- *Pass*: quando as condições do modelo estão de acordo;
- *Warning*: quando é necessário rever alguns detalhes;
- *Fail*: quando o modelo falha naquele ponto e não passaria na ISO, sendo necessária a correção.

A grande vantagem de possuir o teste *compliance* de ISO26262 e a possibilidade de aplicá-lo ainda no estágio de *Model-In-the-Loop*, é que o desenvolvedor não precisa conhecer as diretrizes da norma a fundo para poder adequar o sistema. O próprio MATLAB&Simulink® transforma as diretrizes em pontos de verificação e adequação; seria uma tradução em linguagem matemática das premissas da norma escrita.

2.4 CONCEITOS ESTRUTURAIS DO PADRÃO AUTOSAR

O AUTOSAR (AUTomotive Open System Architecture) é uma Arquitetura de construção de software automotivo, padronizada e aberta, desenvolvida conjuntamente por fabricantes de automóveis, fornecedores de desenvolvedores de ferramentas para a ECU (AUTOSAR,2016).

Ela é baseada no conceito de componentes de software e o principal objetivo da arquitetura é estabelecer um padrão aberto de software embarcado. Isso significa que a arquitetura vai fornecer uma infraestrutura básica para apoia o desenvolvimento

de software automotivo, interfaces de usuário e de gestão para todos os domínios de aplicações para a próxima geração de sistemas automotivos (Neme, 2017).

Os objetivos do AUTOSAR são (AUTOSAR, 2016):

- Escalabilidade para diferentes veículos e plataformas;
- Possibilidade de transferência de software;
- Integração de múltiplos fornecedores de ECU;
- Reutilização de funções;
- Capacidade de manutenção ao longo das atualizações de todo o ciclo de vida do produto.

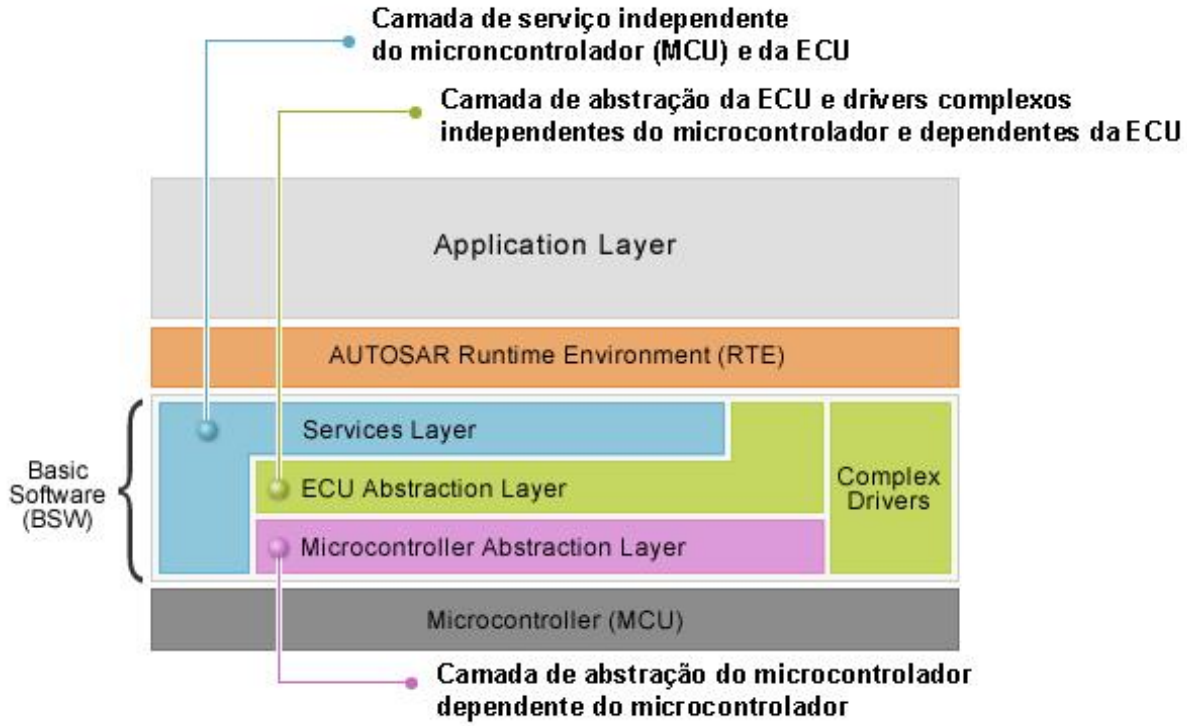
2.4.1 Arquitetura de *Software*

Como está presente nos objetivos do AUTOSAR, a reutilização de funções é possível pelo fato da arquitetura ser modular. O AUTOSAR se utiliza de uma arquitetura formada por 3 camadas principais:

- Camada de aplicação (Application Layer): nessa camada são implementadas as funções desejadas pelo desenvolvedor. Ela é composta por componentes de software de aplicativos que interagem com o RTE (ambiente de execução).
- Ambiente de Execução (Runtime Environment): realiza a comunicação entre o software básico (Basic Software) e os componentes de software e camada de aplicação.
- Camada de Software Básico (Basic Software): módulos de software padronizados sem qualquer trabalho funcional em si que oferece serviços necessários para executar a parte funcional da camada de software superior, logo ele provê abstração, separação, entre o funcionamento do software e do hardware, de forma que um software desenvolvido em AUTOSAR possa ser usado em qualquer hardware.

A figura 8 mostra a estrutura em camada do padrão AUTOSAR:

Figura 8 - Estrutura em Camadas do Padrão AUTOSAR



Fonte: RENESAS adaptado por NEME(2017).

A tabela 1 apresenta a descrição resumida com as características e finalidades de cada uma das camadas da figura 8.

Tabela 1 - Descrição das Camadas do Padrão AUTOSAR

Nome Camada		Descrição	Característica	Finalidade	
				Montagem	Interfaces
Runtime Environment (RTE)		Camada de middleware para prover serviços de comunicação aos SWC AUTOSAR e aplicações que contenham partes de sensor e atuador AUTOSAR.	Faz com que o SWC AUTOSAR seja independente do mapeamento para uma ECU específica.	Específico para a ECU e aplicação gerada para cada ECU.	A interface é completamente independente da ECU.
Basic Software (BSW)	*Services Layer*	Camada superior do *Software* básico (BSW) que fornece as seguintes funções • Sistema operacional (OS);	Fornece serviços básicos e módulos básicos para aplicações.	Específico ao MCU, parte do *hardware* ECU e aplicação.	A interface é independente do MCU e *hardware* ECU.

		▪ Comunicação e gerenciamento da rede veicular; ▪ Serviços de memória (gerenciamento NVRAM); ▪ Serviços de diagnóstico (incluindo o protocolo de comunicação UDS e memória de erros - DTC); ▪ Gerenciamento do estado ECU.			
	ECU Abstraction Layer	Interface com MCAL (incluindo os drivers de dispositivos externos); pode fornecer o seguinte: ▪ Acesso a periféricos e dispositivos, independentemente de estarem dentro ou não do MCU; ▪ Interface de aplicação (API) com o MCU (pinos da porta, tipo de interface).	Faz com que a camada de *Software* superior seja independente do layout do *hardware* ECU.	A montagem é independente do MCU e dependente do *hardware* ECU.	A interface é independente do MCU e hardware ECU.
	Complex Drivers	Camada de *Software* usada para funções complexas que não são encontradas em outras camadas. Esta camada acessa diretamente o MCU. Ex.: Controle de injeção, controle de valores elétricos, detecção da posição, etc.	Cumpre as funções especiais e requisitos de tempo necessários para operação de sensores e atuadores complexos.	Altamente dependente do MCU, ECU e aplicação.	A interface é padronizada e montada de acordo com AUTOSAR (Interface AUTOSAR).
	Microcontroller Abstraction Layer (MCAL)	Módulo de *Software* que acessa diretamente o MCU, módulos periféricos e dispositivos externos que são mapeados pela memória.	Faz com que a camada de *Software* superior seja independente do MCU.	Dependente do MCU.	Interfaces independentes do MCU padronizado.

Fonte: Neme (2017).

2.4.2 Projeto em AUTOSAR

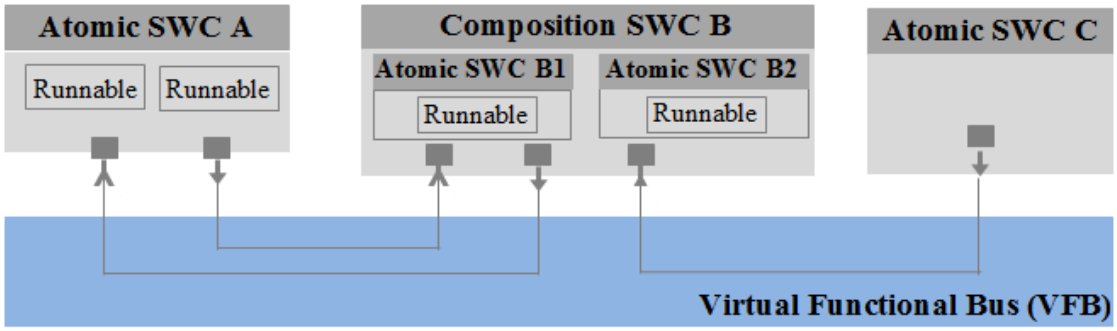
A criação de um sistema baseado na arquitetura AUTOSAR é uma abordagem que abrange algumas etapas, elas estão citadas a seguir:

- Definição da Funcionalidade: através do componente de software (SWC). O componente de software é o elemento chave do projeto e será executado na ECU.
- Definição de modo de Comunicação: a comunicação é chamada de barramento funcional virtual (VFB). Ele tem a função de permitir a implementação independente dos SWC's com a camada de hardware abaixo.

Componentes de software possuem entidades executáveis chamadas *runnables* quem definem o comportamento do software. Um SWC pode possui um ou mais *runnables* dependendo da aplicação. Nesse nível do projeto, o objetivo é desenvolver a função que será encapsulada no SWC sem a preocupação com as características do hardware (ECU) que será utilizado.

A figura 9 mostra um exemplo de componentes de software comunicando-se através do barramento funcional.

Figura 9 - Componentes de Software e Comunicação



Fonte: Neme (2017).

Para se comunicar com outros SWC's, cada componente de software possui portas de comunicação, que são tipificadas de acordo com a interface, por exemplo, padrão emissor/receptor (sender/receiver) para troca de dados.

No final do processo, a fim de integrar a ECU, o ambiente de execução e os módulos do software básico são configurados para trabalhar com as definições do SWC criado.

2.4.3 MATLAB&Simulink® e AUTOSAR

O padrão AUTOSAR atrelado com o desenvolvimento baseado em modelos é uma ferramenta de grande utilidade para a indústria automotiva. Isso pelo fato de que ambos, AUTOSAR e MBD, seguirem o ciclo V de criação e possuírem a característica modular.

Dentro da ferramenta de MBD, MATLAB&Simulink® é possível criar um componente de software completo, com todas as características desejadas pelo desenvolvedor. E através da ferramenta Embedded Coder®, pertencente ao Simulink®, exportar todas as características desse SWC's através das extensões arxml (descrição XML AUTOSAR).

Essas extensões possuem todas as características referente ao sistema divididas em 5 arquivos: *behavior, component, datatype, implementation e interface*; além das extensões, a ferramenta gera automaticamente o código C do sistema.

Além disso, dentro do Simulink, existe a configuração das interfaces de portas de entradas e saídas do sistema, como também dos *dataelements*, que são os valores, os sinais que passarão pelas portas.

3 DESENVOLVIMENTO

Neste capítulo é apresentado todo o desenvolvimento do trabalho, desde as definições estabelecidas pela autora quanto ao funcionamento do sistema até a implementação de todos os conceitos citados no capítulo anterior, necessários para atingir o objetivo proposto pelo trabalho.

O primeiro passo para que o desenvolvimento pudesse ser iniciado foi a definição de todos os requisitos do sistema da plataforma elevatória. Para que tudo seja bem compreendido, o próximo tópico abordará o sistema.

3.1 MOBILE SEAT PLATFORM (MSP)

A Mobile Seat Platform (MSP), ou plataforma de poltrona móvel é um sistema criado pela Marcopolo, uma das maiores fabricantes de carrocerias de ônibus do mundo. A MSP é um projeto de automação veicular de auxílio às pessoas com deficiência física, que possuem dificuldades e constrangimento ao embarcar em veículos de transporte coletivo como o ônibus de viagens. A função básica do dispositivo é possibilitar a movimentação do assento do ônibus através de uma plataforma motorizada, como pode ser vista na figura 10 abaixo.

Figura 10 - Sistema da Plataforma de Poltrona Móvel

Fonte: Nunes (2016).

A plataforma desce até o solo para embarque e desembarque do passageiro. O projeto evita que o passageiro precise ser transportado pelos funcionários das companhias de ônibus, como ocorre atualmente.

O controlador principal dessa plataforma foi desenvolvido primeiramente por Lauro Nunes, colaborador da Marcopolo (Nunes, 2016).

3.1.1 REQUISITOS DO SISTEMA MSP

O objetivo desse sistema é desenvolver o controlador da plataforma, considerando diversas entradas que serão citadas a seguir. A planta física, que corresponde aos motores, o sistema físico e mecânico da plataforma, o ônibus e sua movimentação não foram considerados nem modelados.

Para o funcionamento do sistema, serão analisados sinais da ECU da carroceria, que no modelo, será chamada de ECU *Body.* Nesse caso, os requisitos podem ser divididos em duas ECU's: a ECU Body e a ECU MSP. Os requisitos estão listados abaixo, respeitando essa divisão (Nunes, 2016).

- ECU MSP: a operação da plataforma é determinada pelos sinais recebidos da ECU Body como também de sensores controlados pela ECU MSP;
    - Chave fim-de-curso 1 da plataforma: ativado em nível zero, sempre que a plataforma estiver a bordo do ônibus;
    - Chave fim-de-curso 2 da plataforma: ativado em nível zero, sempre que a plataforma estiver no solo;
    - Sensor de alavanca móvel: o sistema possui uma alavanca móvel que aciona o sistema em caso de falha elétrica. Este sensor é ativado em nível zero e detecta se a alavanca móvel foi usada ou não;
    - Sensor de proteção dos pés: ativado em nível zero, detecta se o passageiro está com os pés na posição correta;
    - Sensor de poltrona: este sensor é ativado em nível zero e detecta se a poltrona está na posição correta ou não;
    - Sensor de corrente: cada vez que há sobrecorrente, o sensor envia um sinal para o motor;
    - Controle: botões de pressão ativos na plataforma.

  Enquanto a plataforma estiver em operação, a ECU enviará um sinal ativando uma sirene.

- ECU Body (BCM): os sinais controlados pela BCM que influenciam a operação da plataforma são:
    - Sensor da porta 1: instalado em uma das portas da plataforma, detecta se está aberta ou fechada;
    - Sensor de porta 2: operação similar ao sensor da porta 1, instalado na porta 2;
    - Sinal de freio: este sinal está disponível na unidade elétrica, impõe a condição que deve ser ativado para habilitar a plataforma;
    - Sinal de velocidade: também está disponível na unidade elétrica, impõe que a velocidade do veículo esteja abaixo de 5km/h para ativar o sistema;
    - Sinal de travamento do chassi: envia informações para a unidade elétrica e bloqueia o movimento do veículo;
    - Sinal de elevação da RPM: envia informações para a unidade elétrica, eleva a RPM do motor para carregar as baterias.

A partir dos requisitos é possível dar início à construção do modelo e por conseguinte o teste de verificação Model-In-the-Loop e Compliance ISO26262.

3.2 DESENVOLVIMENTO EM MBD

Uma vez que os requisitos do sistema foram levantados, e já se sabe como o modelo precisa operar, passa-se para a fase de construção do controle em *Model-Based Design* no ambiente *Simulink*®.

Como são duas ECU's, BCM e MSP, a construção do modelo ficará separada em duas, para obter-se uma descrição completa do que foi realizado. A criação dos sinais de teste, que serão utilizados durante toda a verificação, também ficará separada em duas de acordo com as ECU's de destino.

3.2.1 *Body Control Module* (BCM)

A ECU da carroceria, BCM, é responsável por monitorar e controlar diversos sistemas em um veículo. Num veículo de passeio por exemplo, ela controla o sistema de vidro elétrico, espelhos, ar condicionado, central de alarme. Em um veículo de grande porte como o ônibus, os sistema controlados aumentam tanto de quantidade quanto de complexidade.

Por isso, a BCM desenvolvida no projeto ficou simplificada e condicionada para controlar somente os sinais de portas, velocidade e freios.

Com o auxílio da ferramenta *Stateflow*®, foi criada uma tabela verdade que controla os sinais de entrada. Considerando que existem 5 sinais de entrada, tem-se 32 possibilidades, a tabela de ações é que vai determinar o que acontece com cada possibilidade. A figura 11 mostra a tabela verdade construída com as 5 entradas: speed (velocidade), door1 e door2 (portas da plataforma), frontdoor (porta principal) e stationary brake (freio estacionário) (Nunes, 2016).

Figura 11 - Tabela Verdade Controle BCM

Condition Table | Action Table

	Description	Condition	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30	D31	D32
1	Speed	x==0	T	F	T	T	T	T	F	T	T	T	F	F	F	T	T	T	T	T	F	F	F	F	F	F	F	T	T	F	F	T	F
2	door1	y~=0	T	T	F	T	T	T	F	F	T	T	T	T	T	F	F	T	T	F	F	F	T	T	F	F	F	T	F	F	T	F	F	F
3	door2	z~=0	T	T	T	F	T	T	T	F	F	T	F	T	T	T	T	F	F	T	F	T	F	T	F	F	T	F	F	F	F	T	F	F
4	Front door	w~=0	T	T	T	T	F	T	T	T	F	F	T	F	T	F	T	T	F	F	T	F	F	F	F	T	F	F	F	T	T	T	F	F
5	Stationary	k==0	T	T	T	T	T	F	T	T	T	F	T	T	F	T	F	F	F	F	T	T	T	F	T	F	F	F	F	F	F	F	T	F
		Actions: Specify a row from the Action Table	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Fonte: autora.

A figura 12 mostra a tabela de ações que é especificação das saídas de acordo com as entradas. Note a primeira ação, que habilita a plataforma enviando um sinal p = 1. As outras ações somente emitem avisos no painel de controle do ônibus.

Figura 12 - Tabela de Ações BCM

Condition Table | Action Table

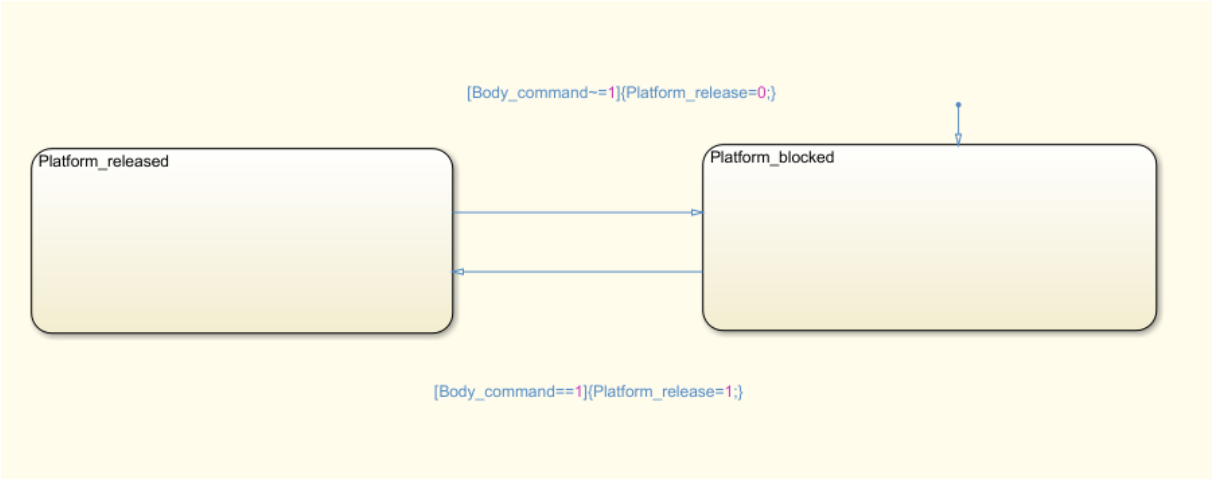
#	Description	Action
1	DPM is free to move	p=1
2	Vehicle on move	p=10
3	Door1 closed	p=11
4	Door2 closed	p=12
5	Front door closed	p=13
6	Stationary brake off	p=14
7	Vehicle on move	p=10
8	Doors closed	p=15
9	Front door closed	p=13
10	Stationary brake off	p=14
11	Vehicle on move	p=10
12	Vehicle on move	p=10
13	Vehicle on move	p=10
14	Front door closed	p=13
15	Stationary brake off	p=14
16	Stationary brake off	p=14
17	Stationary brake off	p=14
18	Stationary brake off	p=14
19	Vehicle on move	p=10
20	Vehicle on move	p=10

Fonte: autora.

Feito o controle principal dos sinais de entrada, foi necessário estabelecer que o sinal p = 1, da ação da tabela, habilitaria a plataforma de fato. Para isso, foi criado uma lógica simples em chart do Stateflow®, com dois estados: um que dependendo do sinal, habilita, senão, existe o outro que desabilita.

A figura 13 mostra a lógica criada, platform_blocked é o estado em que não há habilitação e platform_release é o estado que habilita o funcionamento da plataforma, enviando um sinal, platform_release = 1.

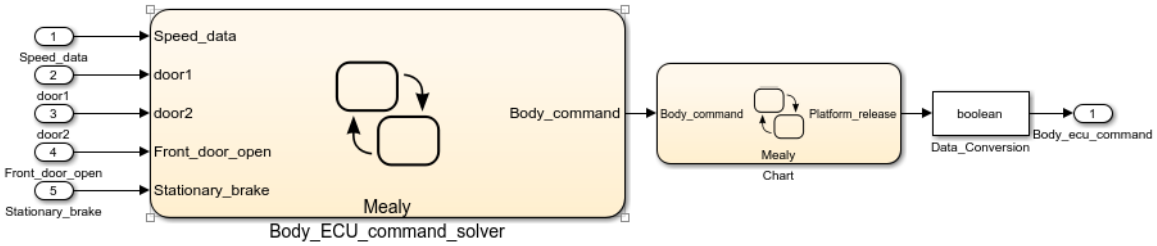
Figura 13 - Lógica de Habilitação



Fonte: autora.

A visão geral do conteúdo do controle, com os charts criados, as entradas e a saída *Body_ecu_command* pode ser vista na figura 14.

Figura 14 - Controle da BCM



Fonte: autora.

3.2.2 Mobile Seat Platform ECU

A ECU da plataforma (MSP ECU) é responsável por controlar os seguintes sinais:

- Body_ecu_command: que vem da BCM;
- Level_sensor: sensor de nível da plataforma;
- Feet_protection: sensor de posicionamento dos pés;
- Seat_reclined: sensor que verifica se a poltrona está reclinada;
- Current_failure_status: verificação da corrente do motor;
- Voltage_failure_status: verificação da tensão do motor;

- Platform_onboard: sensor que verifica se a plataforma está dentro do ônibus;
- Platform_overground: sensor que verifica se a plataforma está no nível do chão;
- Button_up: botão de acionamento para subir a plataforma;
- Button_down: botão de acionamento para descer a plataforma.

3.2.2.1 Controle de Verificação de Corrente

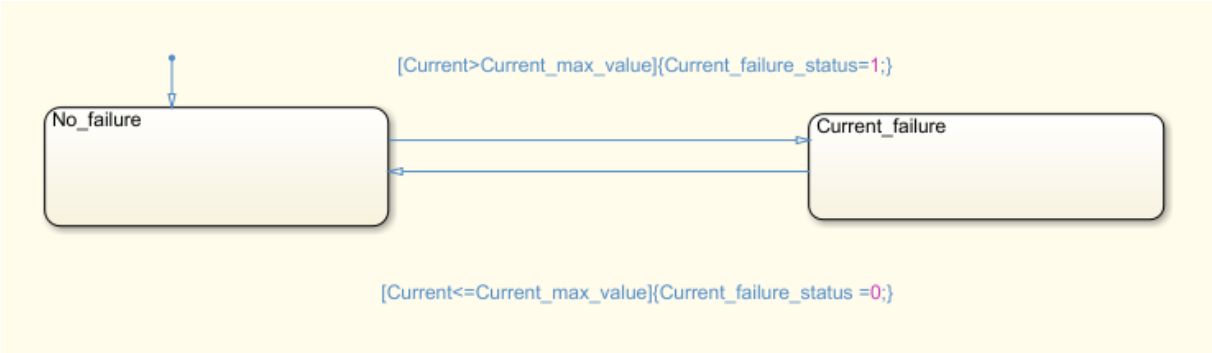
A corrente possui um valor máximo ao qual o sinal não pode ultrapassar. No caso do projeto, a corrente não pode ultrapassar 100A (valor fictício a fim de realizar o controle), senão o sistema desliga.

Por isso, para essa verificação foram criadas duas variáveis:

- Current: variável de entrada do sistema, valor real;
- Current_max_value: variável de comparação, fixada no valor máximo.

O controle foi feito por meio de um chart do Stateflow®, com lógica de máquina de estados. A figura 15 mostra o controle desenvolvido. Quando a lógica é satisfeita (sem falha), o sinal de saída (Current_failure_status) é 0.

Figura 15 - Controle de Verificação de Corrente



Fonte: autora.

3.2.2.2 Controle de Verificação de Tensão

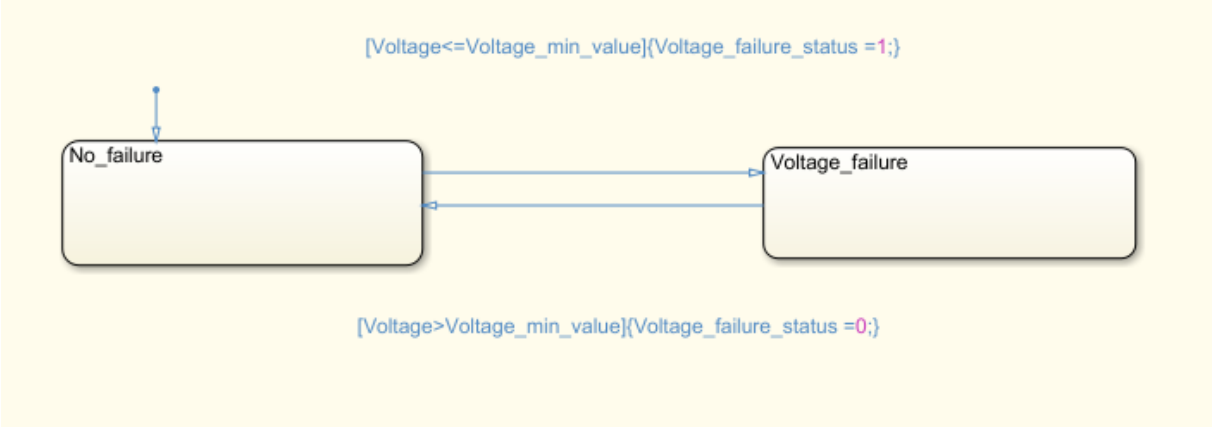
O teste de verificação de tensão possui a característica inversa da verificação de corrente. Aqui a tensão precisa ter um valor mínimo de 200V (valor fictício) para

poder funcionar. Foram criadas duas variáveis em chart do Stateflow®, que controlam a verificação.

- Voltage: variável de entrada do sistema, valor real;
- Voltage_min_value: variável de comparação, fixada no valor mínimo.

O controle foi feito por meio de um chart do Stateflow®, com lógica de máquina de estados. A figura 16 mostra o controle desenvolvido. Quando a lógica é satisfeita (sem falha), o sinal de saída (Voltage_failure_status) é 0.

Figura 16 - Controle de Verificação de Tensão



Fonte: autora.

3.2.2.3 Controle de Habilitação

O controle de Habilitação é um pouco mais complexo que os dois anteriores, por se tratar de uma grande quantidade de sinais, que serão controlados por uma tabela verdade, criada em Stateflow®. Os sinais controlados na habilitação serão: Body_ecu_command, Level_sensor, Seat_reclined, Feet_protection, Current_failure_status e Voltage_failure_status. Serão 6 sinais distribuídos em 64 possibilidades. A tabela de ações podem ser resumida em 6 ações, sendo a mais importante a habilitação para movimento e as outras desabilitação e avisos ao painel.

A figura 17 mostra um fragmento da tabela verdade implementada para esse controle.

Figura 17 - Tabela Verdade do Controlador

Condition Table

	Description	Condition	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19	D20	D21	D22	D23	D24	D25	D26	D27	D28	D29	D30
1	Body ECU command	a == 1	T	F	T	F	F	F	F	F	T	F	F	F	F	T	T	T	T	F	F	F	F	F	F	T	F	F	F	T	T	T
2	Level sensor input	b == 0	T	F	F	T	F	F	F	F	T	T	F	F	F	F	F	F	F	T	T	T	F	F	F	T	T	F	F	F	F	F
3	Feet protection sensor	c == 0	T	F	F	F	T	F	F	F	F	T	T	F	F	F	F	F	T	F	F	F	T	T	F	T	T	T	F	F	F	T
4	Seat reclined sensor	d == 0	T	F	F	F	F	T	F	F	F	F	T	T	F	F	F	T	F	T	F	F	F	F	T	F	T	T	T	F	T	T
5	Current failure	e == 0	T	F	F	F	F	F	T	F	F	F	F	T	T	F	T	F	F	F	T	F	T	F	F	F	F	T	T	T	T	F
6	Voltage failure	f == 0	T	F	F	F	F	F	F	T	F	F	F	F	T	T	F	F	F	F	F	T	F	T	T	F	F	F	T	T	F	F
		Actions: Specify a row from the	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30

Fonte: autora.

A tabela de ações correspondente a tabela verdade da figura 17, pode ser vista na figura 18. Dando ênfase para a saída Condition_code (g), g = 1, que habilita o processo.

Figura 18 - Tabela de Ações

Condition Table | Action Table

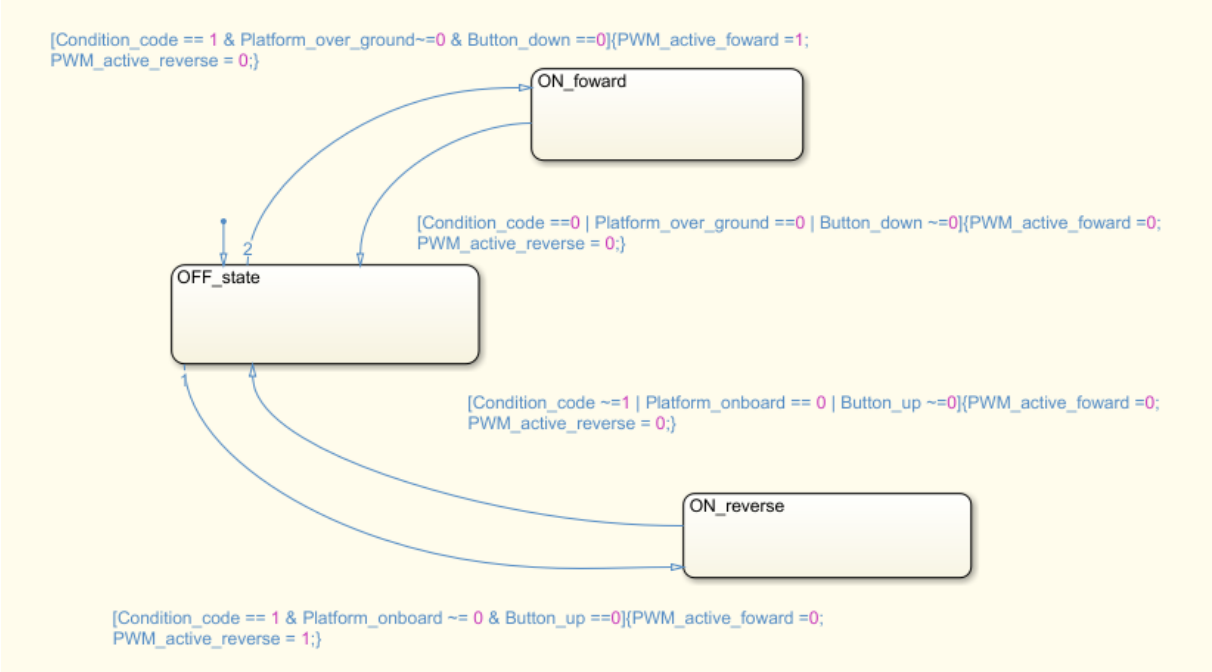
#	Description	Action
1	All signal conditions are met	g = 1
2	Body ECU blocking command	g = 10
3	Lever pulled	g = 11
4	Body ECU blocking command	g = 10
5	Body ECU blocking command	g = 10
6	Body ECU blocking command	g = 10
7	Body ECU blocking command	g = 10
8	Body ECU blocking command	g = 10
9	Feet protection out of position	g = 12
10	Body ECU blocking command	g = 10
11	Body ECU blocking command	g = 10
12	Body ECU blocking command	g = 10
13	Body ECU blocking command	g = 10
14	Lever pulled	g = 11
15	Lever pulled	g = 11
16	Lever pulled	g = 11
17	Lever pulled	g = 11
18	Body ECU blocking command	g = 10
19	Body ECU blocking command	g = 10
20	Body ECU blocking command	g = 10

Fonte: autora.

3.2.2.4Controle de Sentido

O controle de sentido da plataforma é que estabelece se a plataforma vai subir ou descer. Os sinais que serão controlados são Condition_code, platform_onboard, platform_over_ground, button_down, button_up. Através de uma lógica em chart foi possível criar 3 estados: on_foward, on_reverse e off_state, que correspondem aos estados descer, subir e desligado. A figura 19 mostra o controle implementado.

Figura 19 - Controle de Sentido



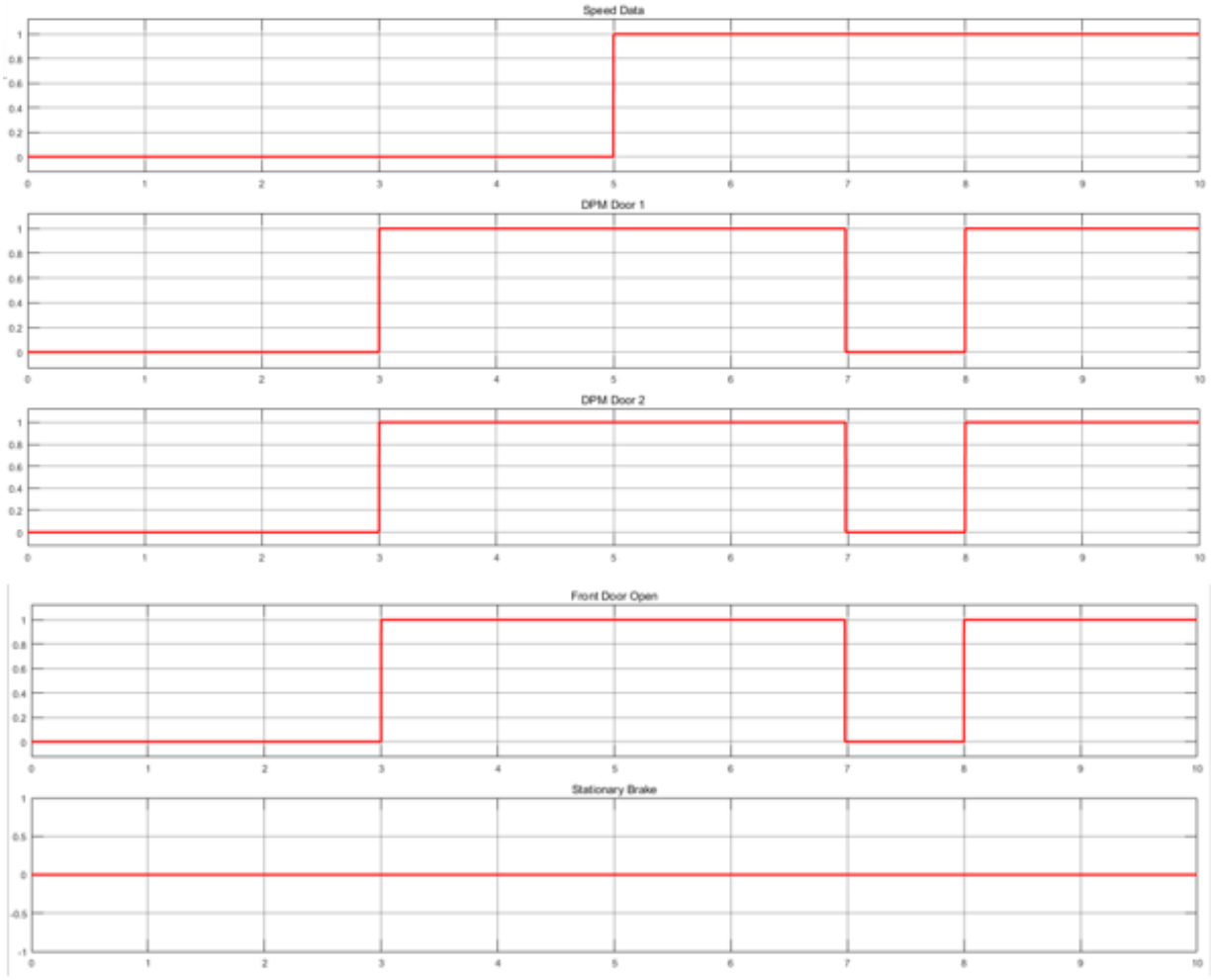
Fonte: autora.

As saídas desse controle são PWM_active_foward e PWM_active_reverse que são o acionamento dos motores.

3.2.3 Sinais de Entrada BCM

Após a construção do modelo, é preciso criar vetores de teste, que são os sinais de entrada, a fim de verificar o funcionamento do sistema criado. As entradas para a BCM são todas booleanas (1 ou 0). Com o auxílio de um bloco Signal Builder foi possível construir 5 blocos de sinais com tempo de execução de 10 segundos. A figura 20 mostra o bloco de sinais construído.

Figura 20 - Bloco de Sinais BCM

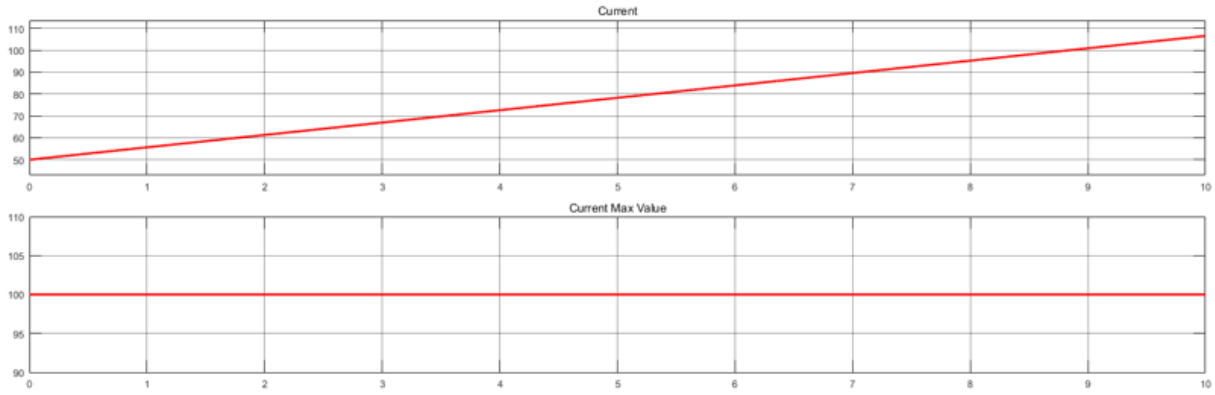


Fonte: autora.

3.2.4 Sinais de Entrada MSP ECU

De igual modo, as entradas para a MSP ECU foram criadas, com o auxílio do signal builder. A figura 21 mostra as entradas para a verificação de corrente.

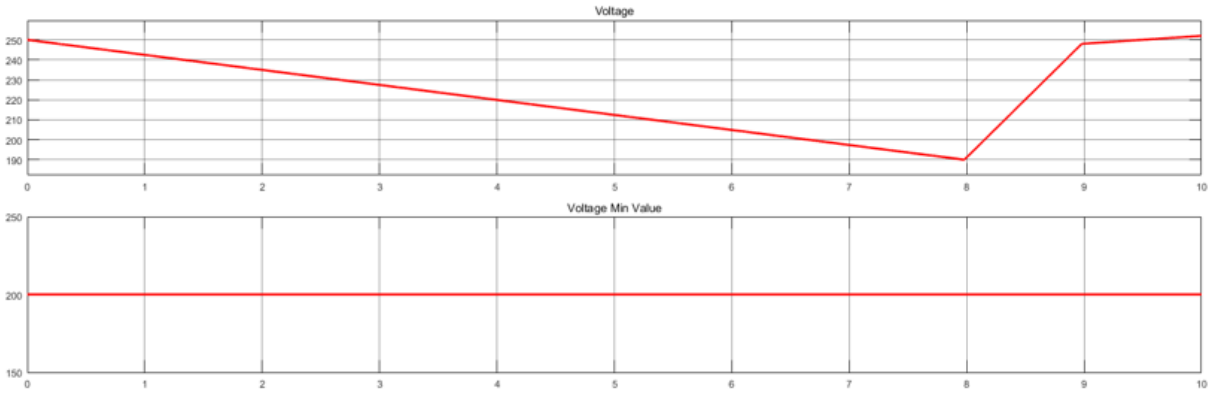
Figura 21 - Entradas para Verificação de Corrente



Fonte: autora.

A figura 22 mostra as entradas para verificação de tensão.

Figura 22 - Entradas para Verificação de Tensão



Fonte: autora.

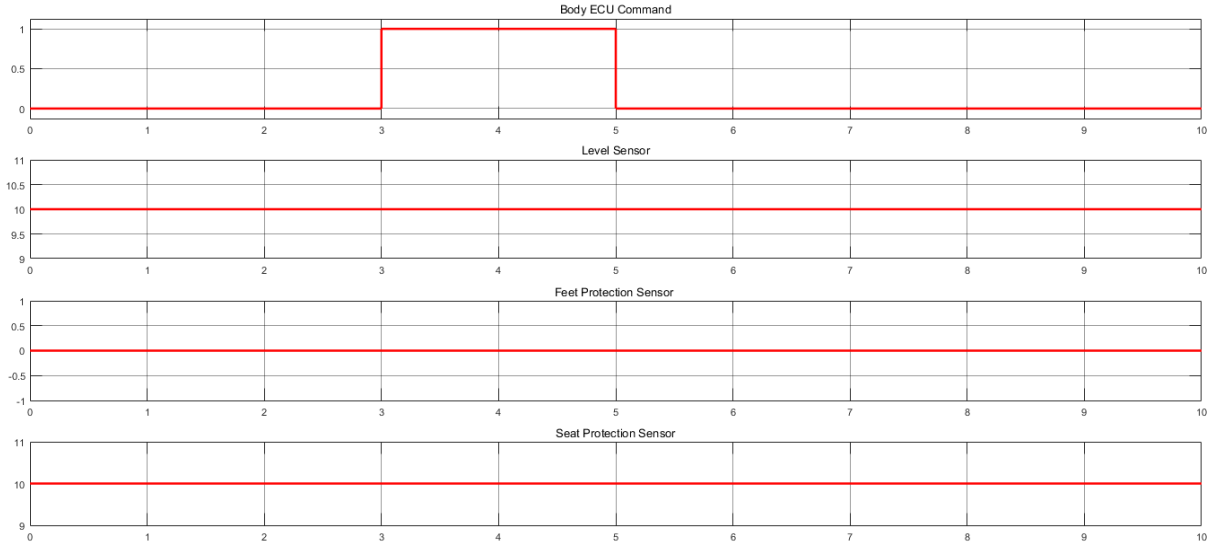
A figura 23 mostra as entradas para controle de habilitação da plataforma, que são resultado da verificação de corrente e tensão.

Figura 23 - Corrente e Tensão

Fonte: autora.

A figura 24 mostra as entradas para controle de habilitação da plataforma, vindas dos sensores.

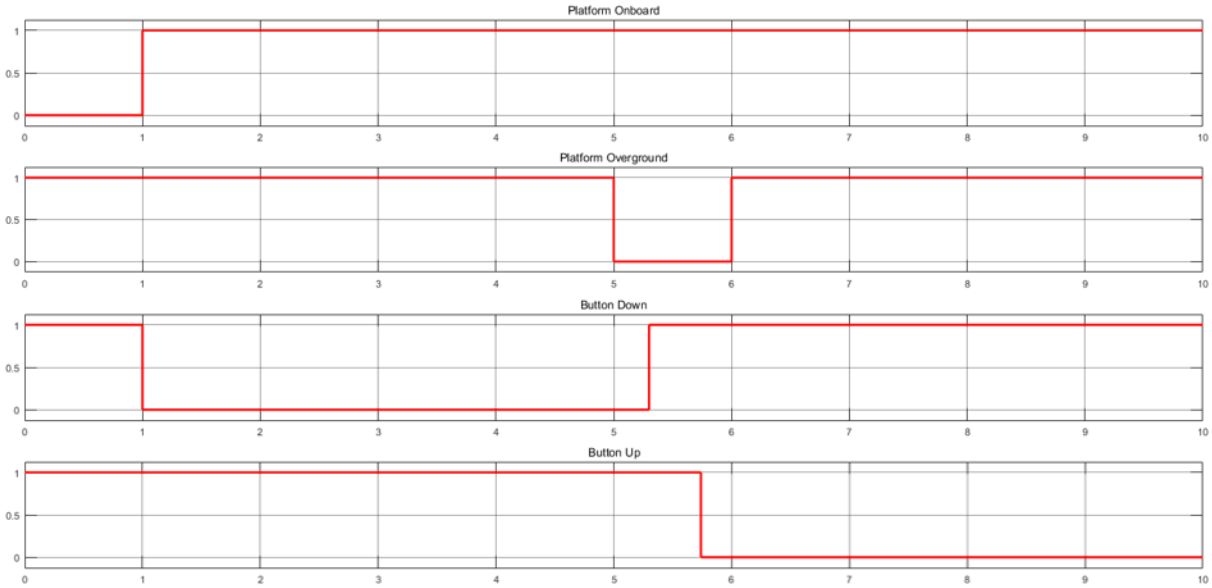
Figura 24 - Entrada dos Sensores



Fonte: autora.

E por fim, os sinais gerados para simular os botões de acionamento e os sensores de presença da plataforma. Os sinais podem ser vistos na figura 25.

Figura 25 - Entradas Sentido de Rotação



Fonte: autora.

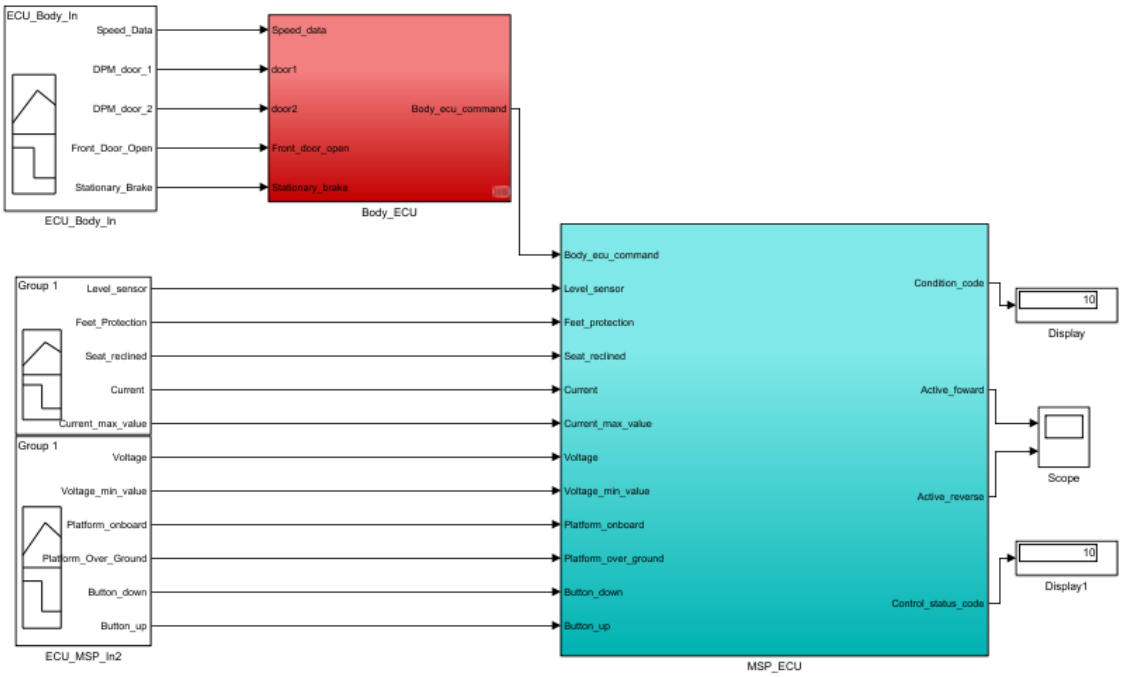
3.3 TESTES DE VERIFICAÇÃO E COMPLIANCE

Por questões de organização, foi de extrema importância inicialmente mostrar o desenvolvimento/criação dos modelos e sinais, para a partir de agora, demostrar todos os testes realizados no processo.

É importante deixar claro, que das etapas de verificação optou-se por desenvolver apenas os estágios MIL e SIL, por abrangerem toda a construção e codificação em C, a fim de comparar com o que será desenvolvido em AUTOSAR, que também gerará um código em C.

O sistema completo, com as duas ECU's pode ser visto na figura 26; onde em vermelho está todo o controle da BCM em um subsistema e em azul o controle da MSP ECU em outro subsistema. Do lado esquerdo da imagem, podem ser vistos os geradores de sinais com as entradas citadas nos dois tópicos anteriores.

Figura 26 - Sistema Completo



Fonte: autora.

3.3.1 Model-In-the-Loop

Após a completa construção do sistema, o próximo passo são as verificações. Dando início, com o primeiro estágio está a verificação MIL. Nessa fase, existe uma comparação entre entradas e saídas do sistema. Conferindo se naquele momento, com aquela determinada entrada, o sistema deveria ter respondido dessa ou daquela forma.

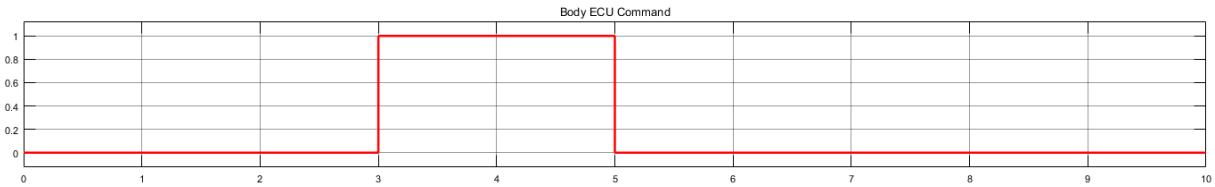
Para que essa comparação fosse possível, com a ajuda de scopes, que são com visualizadores de sinal, do Simulink, foram tiradas todas as imagens das saídas.

Os sinais considerados para essa comparação foram as saídas Body_ecu_command (para verificar a construção de BCM), active_foward e active_reverse (para verificar a construção de MSP ECU).

A figura 27 mostra a saída Body_ecu_command. A verificação pode ser feita, fazendo uma comparação entre a figura 27 e a figura 20 com os pontos a seguir:

- Dos instantes 0 a 3s: Body_ECU_Command = 0, porque as portas estão fechadas, apesar da velocidade do ônibus ser menor que 5km/h, sinal ‘0’, e o freio estacionário estar acionado;
- Dos instantes 3 a 5s: Body_ECU_Command = 1, porque as portas estão abertas, velocidade do ônibus é menor que 5km/h e o freio estacionário está acionado;
- Dos instantes 5 a 10s: Body_ECU_Command = 0, porque apesar das portas estarem abertas e o freio estacionário estar acionado, o ônibus está em movimento, sinal ‘1’.

Figura 27 - Verificação BCM MIL



Fonte: autora.

A próxima verificação será do sistema da MSP ECU. As saídas active_foward e active_reverse estão presentes na figura 28. A verificação pode ser feita com os seguintes pontos:

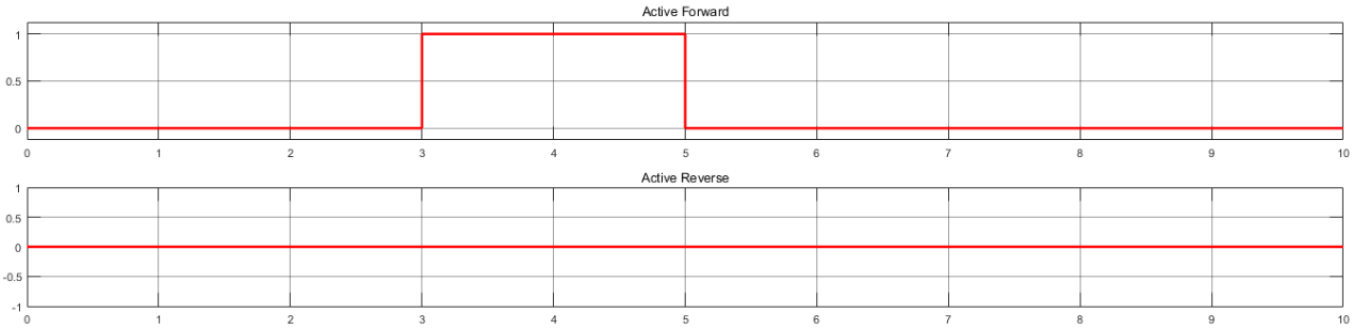
- Instantes 0 a 3s:

- Active_foward e Active_reverse não realizam movimento porque não houve habilitação do funcionamento da plataforma através da variável Body_ECU_Command.

- Instantes 3 a 5s:
  - Active_foward realiza movimento porque houve habilitação do funcionamento por Condition_Code, o sensor Platform_over_ground não está acionado (plataforma não está no chão) e o botão Button_down foi pressionado.
  - Active_reverse não realiza movimento porque apesar de haver habilitação da plataforma e o sensor Platform_onboard estar acionado (plataforma está no chão), o botão Button_up não foi pressionado.
- Instantes 5 a 10s:
  - Active_foward e Active_reverse não realizam movimento porque não houve habilitação do funcionamento da plataforma através da variável Body_ECU_Command.

A habilitação do funcionamento pela ECU da carroceria tem prioridade em relação às outras entradas, de modo que se ela não estiver habilitada não movimentação da plataforma.

Figura 28 - Verificação MSP ECU MIL



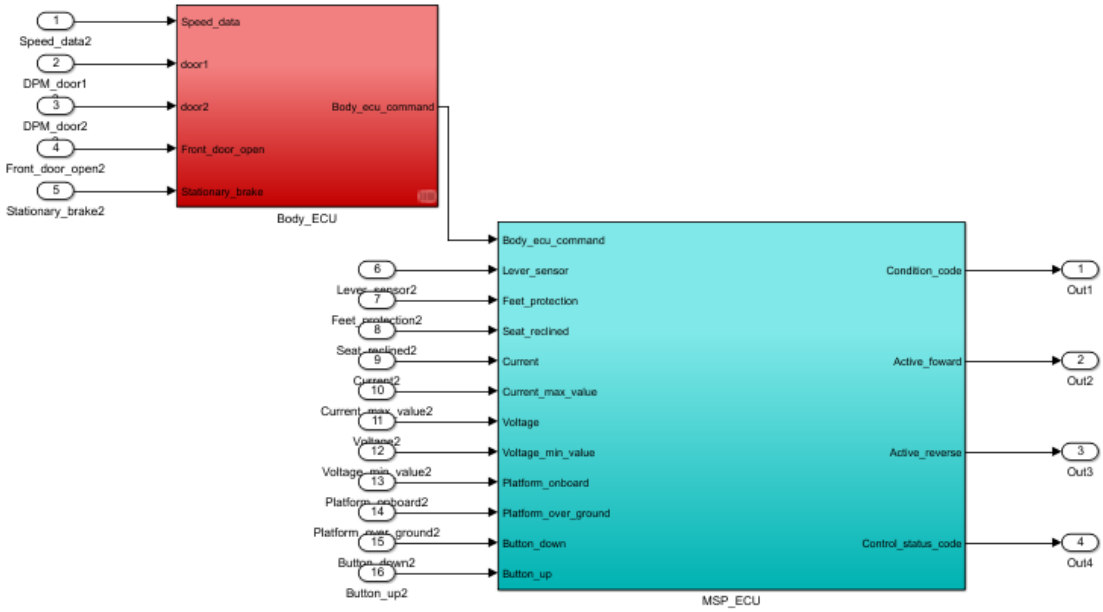
Fonte: autora.

A comparação entre os sinais evidencia o bom funcionamento e o cumprimento com os requisitos iniciais do projeto.

3.3.2 Teste de Compliance ISO26262

Para que o teste de compliance possa ser realizado, é necessário retirar todas as entradas do sistema e deixá-lo com portas de entrada e saída genéricas. A figura 29 mostra o sistema já pronto para passar pela verificação da ISO.

Figura 29 - Sistema Simplificado

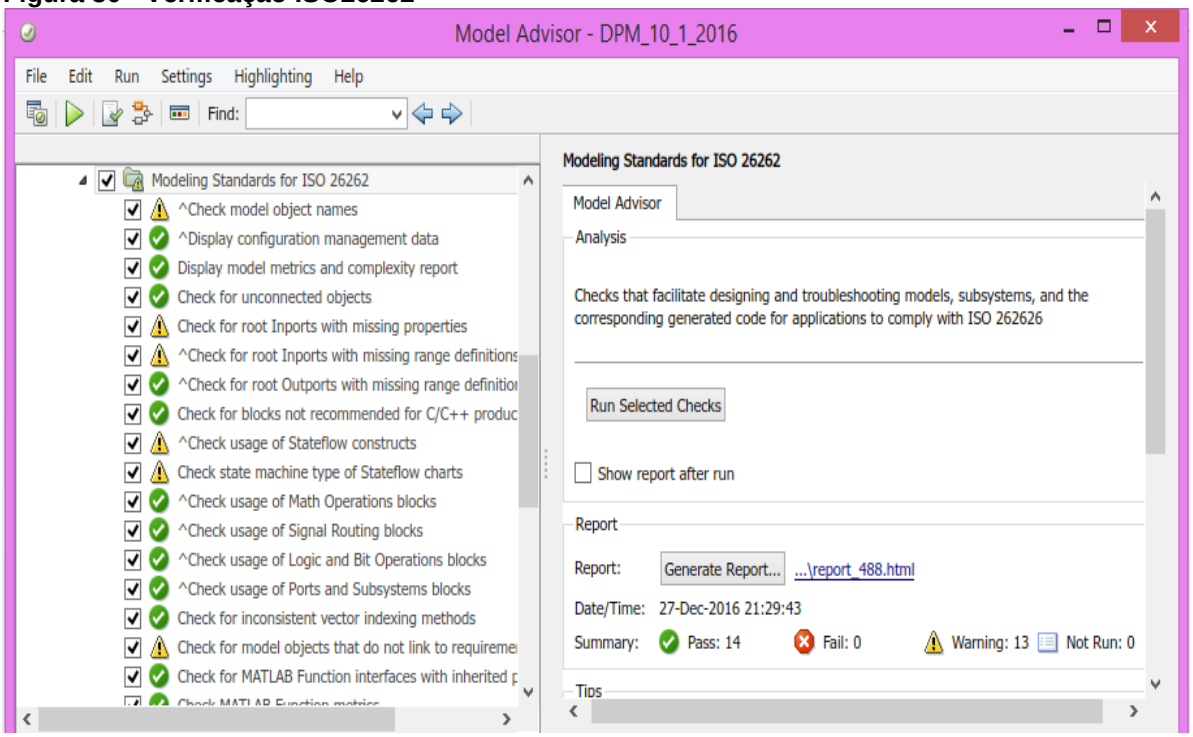


Fonte: autora.

Com o auxílio da ferramenta Design Verifier®, através do Model Advisor, aba de ferramentas que contém o Kit de Certificação IEC, e escolhendo a opção “Modeling Standards for ISO26262” encontram-se os 27 pontos de verificação da norma.

Ao submeter o modelo à essa verificação encontram-se 14 pontos corretos, classificados com o termo ‘pass’, 13 pontos que necessitam de atenção com o termo ‘warning’, e nenhum erro ou ‘fail’. Como pode ser visto na figura 30. As bolinhas verdes significam ‘pass’ e os triângulos amarelos ‘warning’.

Figura 30 - Verificação ISO26262



Fonte: autora.

Como a intenção é tornar o sistema mais aceitável pela norma, dessa forma, a tarefa agora é corrigir cada um dos pontos que indicam *warning*. Ao olhar a descrição do ponto, ficam indicadas as ações que devem ser tomadas a fim de obter o 'pass'.

Alguns dos pontos corrigidos foram:

- As especificações de portas de entrada que não devem ser genéricas. A indicação é que todos os valores de mínimo, máximo, tipo de variável, dimensionamento da porta sejam especificados. Isso evita que o programa aceite outro tipo de variável com intervalos de valores de entrada diferentes.
- Configurações de detecção de ultrapassagem de valores. Se houver algum valor atípico ao sistema, este tratará como erro, não somente aviso.
- Classificações das máquinas de estados (chart) presentes no modelo. Nenhuma delas deve ficar como clássica, padrão Simulink®, elas devem estar definidas entre Mealy e Moore. No caso do modelo do sistema do projeto, todas estão na classificação Mealy.
- Todos os blocos do modelo devem estar ligados a itens na lista de requisitos. Para isso um documento de texto é criado e cada bloco é ligado ao seu requisito. Criando assim uma possível rastreabilidade entre modelo e requisitos, inclusive no código.

Com o suporte da ferramenta, foi possível adequar o modelo e já deixa-lo pronto para a próxima fase de verificação.

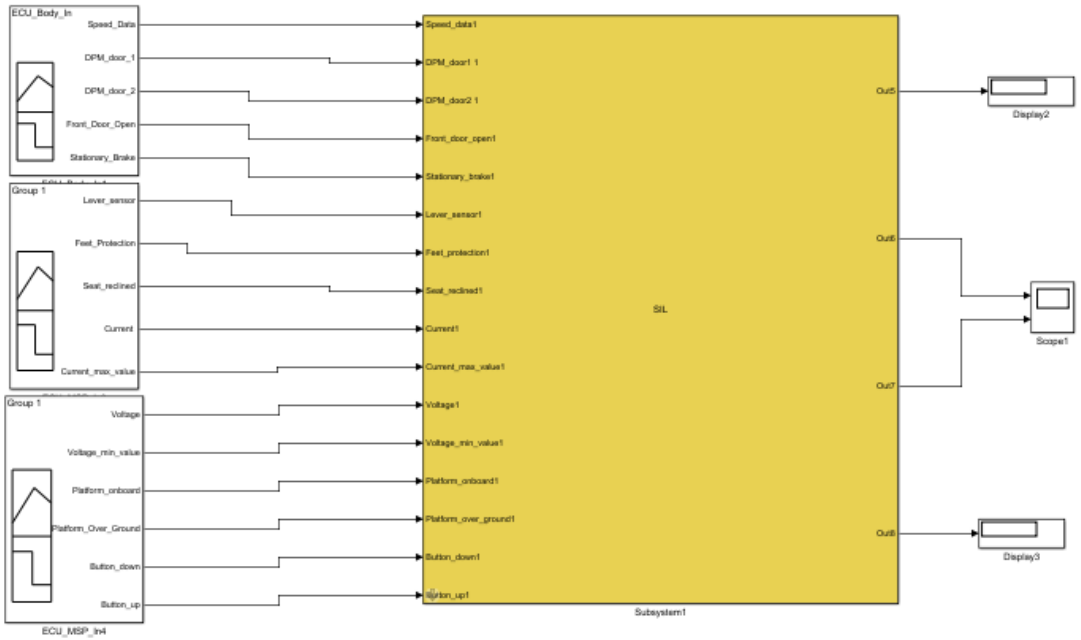
3.3.3 Software-In-the-Loop

Após a adequação do modelo segundo a norma ISO26262, o próximo estágio é fazer a verificação Software-In-the-Loop. Com isso testaremos o modelo num nível mais real do sistema.

Parte-se do modelo da maneira como ele está na figura 29. Com alguns comandos pede-se a ferramenta que gere o código para SIL. Um novo modelo vai abrir com um único bloco. Esse bloco é o onde está o código C do sistema. Não é possível abri-lo. Todo o sistema criado estará inserido ali. O único procedimento que cabe ao desenvolvedor é colocar os blocos de entradas para verificar a robustez e o comportamento do sistema.

A figura 31 mostra o conjunto do bloco de SIL, em amarelo, e os blocos de entrada e saída.

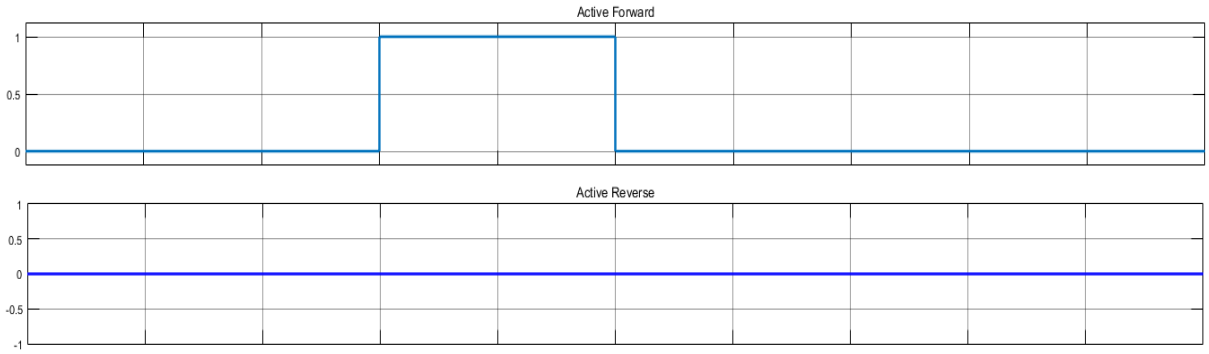
Figura 31 - Verificação SIL



Fonte: autora.

Considerando que as entradas são as mesmas do processo MIL, as únicas saídas que poderão ser visualizadas e comparadas são: active_foward e active_reverse. A figura 32 mostra as saídas desejadas.

Figura 32 - Verificação Final SIL



Fonte: autora.

Como pode ser observado nas figuras 28 e 32, a verificação é satisfatória, por possuírem os mesmos valores para as saídas e cumprirem com os requisitos do sistema.

3.4 PROCESSO DE MIGRAÇÃO PARA ARQUITETURA AUTOSAR

Em um fluxo de trabalho AUTOSAR, uma variedade de ferramentas oferecidas por diferentes fabricantes podem ser encontrados para executar uma parte especifica do processo. Estas ferramentas podem ser divididas em 3 classes:

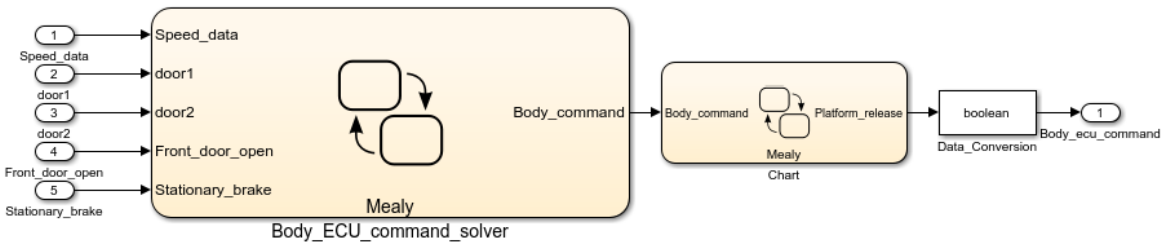
- C IDEs: correspondem ao desenvolvimento de componentes de software. No caso do projeto em questão é a ferramenta Simulink®. Cada SWC é composto por um arquivo de descrição XML AUTOSAR (ARXML) e seu respectivo arquivo .c. Isto corresponde à camada superior no nível AUTOSAR.
- Ferramentas de design de nível de sistema: responsável pelo desenvolvimento dos sistemas e subsistemas. Algumas ações podem ser executadas como ligações SWC e mapeamento ECU, topologia hardware, gestão de comunicação de barramento, mapeamento de elementos de dados, etc. Nesse trabalho, foi utilizado o software SystemDesk® da dSpace®.

- Ferramentas de configuração de software básico: estas ferramentas são responsáveis por configurar cada ECU individualmente e gerar seus arquivos hex que serão gravados nos micro controladores da ECU. Nenhuma ferramenta foi utilizada com essa função.

Para o desenvolvimento desse trabalho, foi utilizado o Simulink e o Embedded Coder da MathWorks para construir o componente de software como descrito nos tópicos anteriores. Com o auxílio do software SystemDesk (dSPACE) para construir a RTE e uma parte da camada de software básico, necessária para a virtualização da ECU e com a ferramenta VEOS (dSPACE) a simulação da ECU virtual.

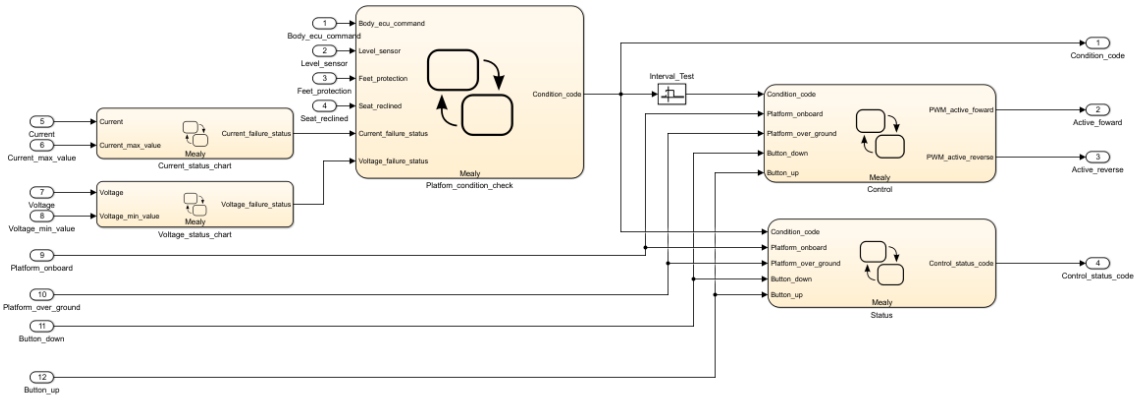
Antes de iniciar o processo de configuração e migração para o padrão AUTOSAR, houve a divisão de ECU's. Cada ECU ficou em um modelo separado do outro (dois arquivos .slx). Isso para que ao migrar para AUTOSAR, existam dois componentes de software (SWC). Por isso tem-se na figura 33, o modelo da BCM e na figura 34, o modelo da ECU MSP.

Figura 33 - Modelo BCM



Fonte: autora.

Figura 34 - Modelo ECU MSP



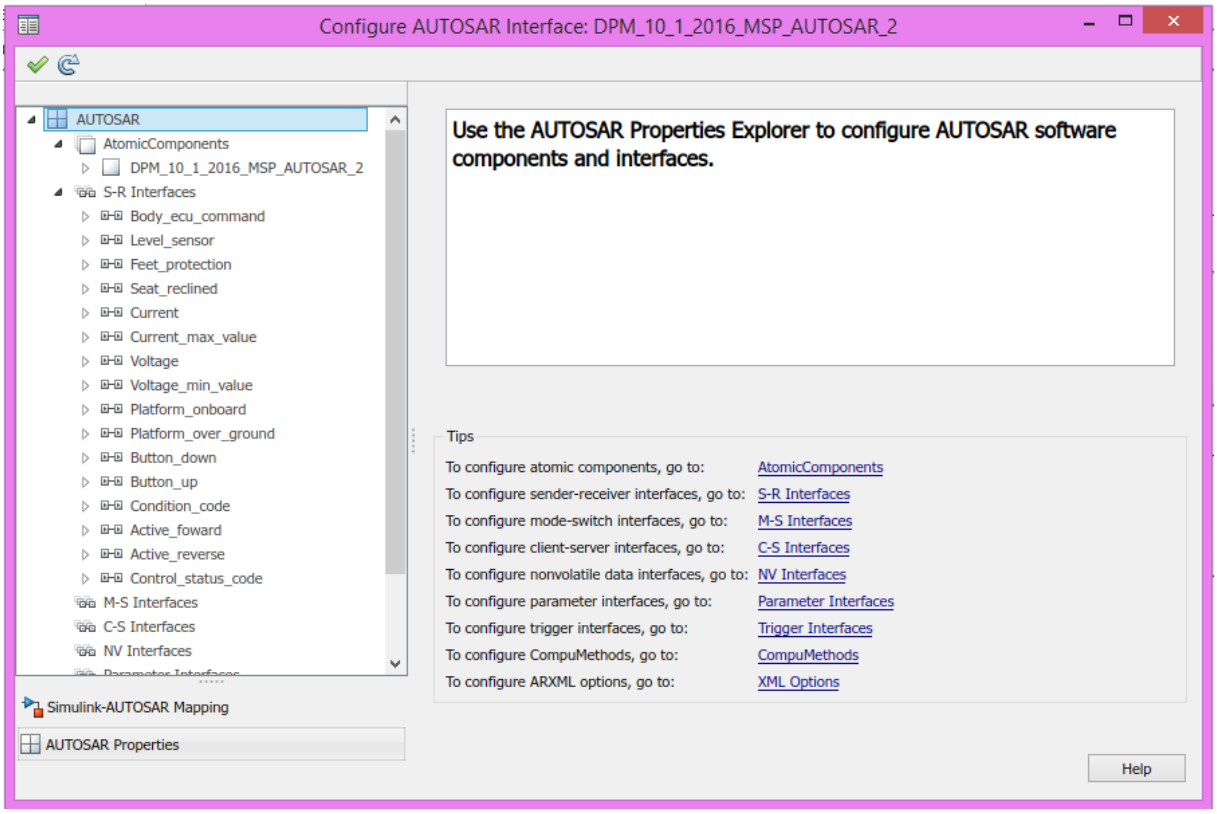
Fonte: autora.

Após a divisão, continuou-se a trabalhar com os modelos separados. O Simulink é quem faz a definição dos parâmetros dos componentes usando um compilador interno chamado Embedded Coder®. Cada componente do modelo é relacionado com as especificações de variáveis, tipos de dados, e funcionalidades.

Cada modelo será definido como um componente de software, com a definição das portas do SWC com a respectiva interface de comunicação. Essa interface carrega informações como elementos de dados, escalas, tipos de dados e variáveis.

A configuração de todos os valores ainda no ambiente Simulink® é extremamente importante a fim de evitar erros e perda de tempo durante a criação do RTE. A figura 35 mostra o painel de configurações do modelo da ECU MSP.

Figura 35 - Janela de Configuração AUTOSAR



Fonte: autora.

Após a configuração, validação e geração de código automático pelo Embedded Coder de todos os arquivos. O próximo passo é a exportação destes arquivos para o SystemDesk®, que fará a conexão entre os SWC.

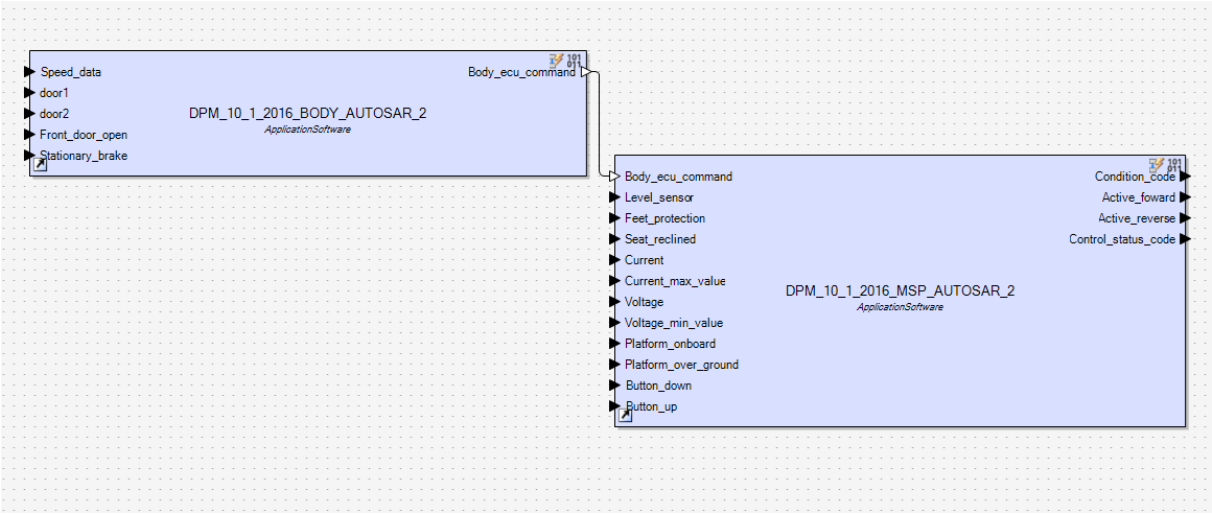
3.5 CRIAÇÃO DE SINAIS NO SYSTEMDESK

É importante deixar claro que a comunicação entre os dois SWC's criados, apesar de serem denominados duas ECU's, serão implementadas com comunicação Intra-ECU. Isso significa que todos os SWC que precisam se comunicar uns com os outros estão disponíveis sobre o mesma ECU, conectados via RTE e software básico. Esse tipo de comunicação foi implementado por questões de disponibilidade de licença e ferramentas de software.

O SystemDesk é a ferramenta que cria a estrutura do sistema, de modo que haja o funcionamento correto da ECU. Nessa etapa do projeto, as fases serão: arquitetura de software, topologia de hardware, comunicação pelo barramento e

simulação. Um projeto foi criado na ferramenta SystemDesk. A figura 36 mostra os SWC's importados e relacionados.

Figura 36 - Sistema em AUTOSAR



Fonte: autora.

A configuração da topologia de hardware é a próxima a ser criada. Essa topologia trata-se de uma configuração de ECU que precisa ser criada; como a comunicação é intra-ECU, os SWC's serão colocados na mesma ECU.

A fim de obter uma simulação virtual de ECU, é preciso que uma pequena configuração da camada de software básico seja feita, para que haja o gerenciamento da estrutura AUTOSAR. O SystemDesk oferece uma pilha de software básico, que já é suficiente para a construção da ECU virtual.

A próxima etapa é a simulação do sistema com a ECU e o barramento de comunicação. Para isso, é preciso criar as variáveis desejadas para estímulo e medição.

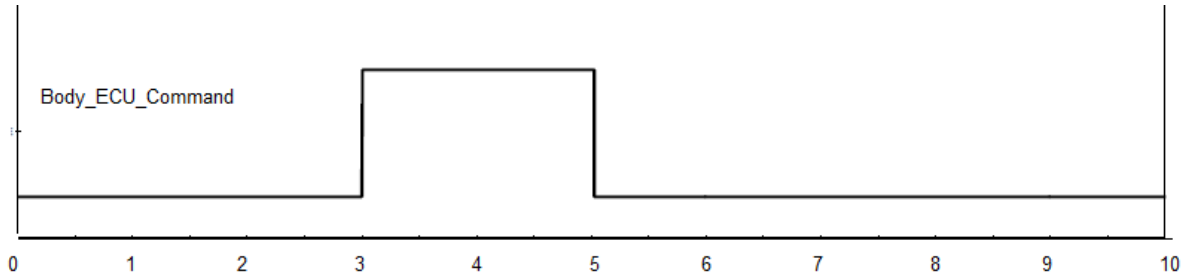
As variáveis de estímulos serão as entradas do sistema. Elas terão o mesmo formato dos sinais criados e que podem ser vistos nas figuras 20 a 25. As variáveis de medição serão as que possuem a opção de medir um elemento de dados de uma porta do componente.

A configuração da simulação é importante nessa fase, porque gerencia toda o ambiente de teste. Os parâmetros de stop time, break time foram definidos.

Depois da criação das variáveis é possível executar a simulação de um V-ECU (ECU virtual). A partir daí, as medições são feitas e são comparadas com as entradas e requisitos do projeto.

O comportamento do sistema pode ser verificado através dos gráficos. A figura 37 mostra a variável Body_ECU_Command, saída do bloco BCM e entrada para o MSP.

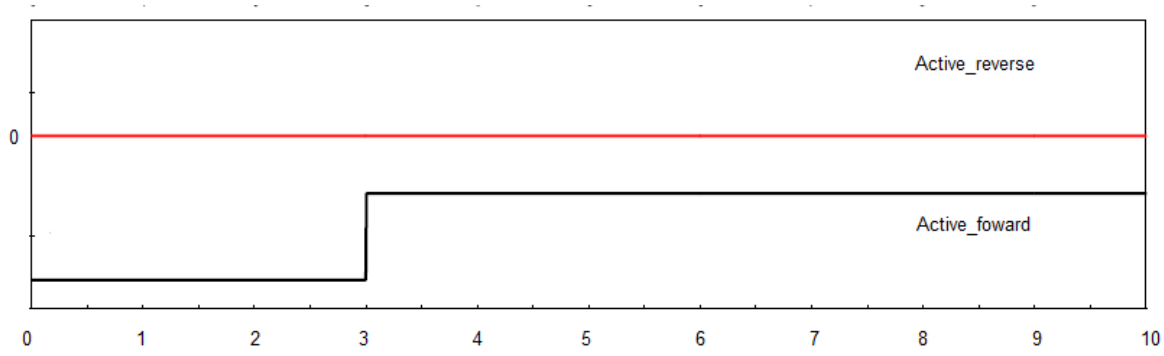
Figura 37 - Saída AUTOSAR BCM



Fonte: autora.

E por fim, a figura 38 mostra o comportamento das saídas active_foward e active_reverse como resposta ao funcionamento do sistema em arquitetura em AUTOSAR.

Figura 38 - Saída AUTOSAR MSP



Fonte: autora.

As saídas verificadas nas figuras 37 e 38 evidenciam o funcionamento correto do sistema, considerando as entradas geradas (figuras 20 a 25) e os requisitos do sistema.

Nesse caso, o sistema foi verificado encerrando o processo de desenvolvimento proposto pela autora.

4 CONCLUSÃO

O desenvolvimento de software automotivo se baseia na adoção de métodos, processos, ferramentas e padronizações que assegurem ao sistema qualidade, segurança, satisfação às regulamentações e conforto a todos os envolvidos no processo; sem contar a redução de custos e de tempo despendido.

Com o aumento da complexidade dos sistemas, realizar a programação manual e tradicional fica comprometida em relação a toda a demanda citada acima. Foi necessário encontrar meios alternativos, e com essas metodologias alternativas, vem a necessidade de compreendê-las academicamente, na Engenharia Eletrônica.

O início deste trabalho consistiu de um levantamento bibliográfico que se mostrou importante para consolidar de como a metodologia do Model-Based Design e a arquitetura AUTOSAR funcionam e de como poderiam trabalhar em conjunto a fim de se obter um software mais robusto, seguro e que ofereça o máximo de inovação.

O Model-Based Design, desenvolvimento baseado em modelos mostrou-se uma excelente ferramenta que minimiza gastos e maximiza a produção de uma nova função. As diversas verificações e por fim, no término do processo, a validação faz com que a probabilidade de um *recall*, quando a fabricante precisa recolher o produto por falhas técnicas, seja muito menor.

A utilização do MBD com a ferramenta MATLAB&Simulink®, além de permitir um desenvolvimento mais intuitivo, permite que o usuário adeque sua criação à diversas normas industriais, como no caso da norma ISO26262, que trata da segurança funcional em sistemas eletroeletrônicos embarcados em veículos automotivos.

Por fim, a arquitetura AUTOSAR, que promete uma escalabilidade de hardware muito maior, o que significa que o desenvolvedor pode criar uma nova função sem necessariamente saber que tipo de hardware será utilizado. Aumentando as possibilidades da indústria automotiva com suas vantagens. Unir o desenvolvimento baseado em modelos com a arquitetura AUTOSAR é expandir ainda mais essas vantagens.

O sistema de poltrona móvel, tem a proposta de possibilitar uma melhor movimentação e embarque de pessoas deficientes em ônibus de viagens. Desenvolver esse sistema sob a MBD e AUTOSAR significa que em breve essa nova função estará disponível no mercado, facilitando a vida de diversas pessoas.

Neste trabalho observou-se que, utilizando os conceitos do MBD e AUTOSAR, foi possível verificar o funcionamento do sistema, durante todo o processo, através de testes. Estes testes possibilitaram que os erros encontrados pudessem ser rapidamente corrigidos e melhorias de performance fossem incluídas, ainda na primeira fase.

É importante deixar claro que modificações poderiam ser feitas na estruturação e nos testes aplicados.

A comunicação entre as ECU's poderia ser via rede CAN, não somente bits únicos como foi o caso da implementação a fim de tornar mais real a funcionalidade do sistema.

Outra modificação que poderia ser feita, seria a realização da verificação Processor-In-the-Loop, embarcando o sistema em um processador (FRDM-KL25Z, processador da Freescale) e criando as entradas por meio de outro hardware de aquisição de dados (NI-DAQ, da National Instruments), a fim de verificar o comportamento do sistema em um controlador físico.

Academicamente, esse trabalho fica disponível para que outro possa, senão implementar as mudanças citadas, aperfeiçoar a sua maneira o desenvolvimento até aqui gerado, dando continuidade à geração de conhecimento.

REFERÊNCIAS

ANFAVEA. **Anuário da Indústria Automobilística Brasileira**. São Paulo, 2015.

AUTOSAR GbR. **AUTOSAR Basics**. Disponível em: <http://www.autosar.org/index.php?p=1&up=1&uup=0>. Acesso em: 10 out. 2016.

AUTOSAR GbR. **AUTOSAR Technical Overview**. Disponível em: <http://www.autosar.org/index.php?p=1&up=2&uup=0>. Acesso em: 10 out. 2016.

AUTOSAR GbR. **AUTOSAR - The Worldwide Automotive Standard for E/E Systems**. Disponível em: < https://www.autosar.org/fileadmin/files/presentations/AUTOSAR_Brochure_EN.pdf>. Acesso em 10 out. 2016.

BARBIERI, G.; et al. **A Model-Based Design Methodology for the Development of Mechatronics Systems.** Elsevier, v.24, p.833-843, 2014. Disponível em: < http://dx.doi.org/10.1016/j.mechatronics.2013.12.004>.

BROY, M. **Challenges in Automotive Software Engineering**. ICSE'06 – Proceedings of the 28th International Conference on Software Engineering, New York, p.33-42, 2006.

CHARETTE, R. N. This Car Runs on Code. **IEEE Spectrum**, New York, fev. 2009.

CTE Embedded Systems. **Embedded Systems Guide**. Disponível em:< http://www.embedded-systems-portal.com/CTB/AUTOSAR_ECU_Abstraction_Layer,10022.html>. Acesso em: 10 out. 2016.

DILLABER, E.; KENDRICK, L.; JIN, W.; REDDY, V. **Pragmatic Strategies for Adopting Model-Based Design for Embedded Applications**. Disponível em: https://www.mathworks.com/tagteam/63207_Strategies%20for%20Adopting%20MBD%20for%20Embedded%20Apps.pdf

dSPACE. **MicroAutoBox Overview of Board Revisions Release 2014B** Disponível em: <https://www.dspace.com/files/pdf1/MicroAutoBoxOverviewofBoardRevisions_Release2014B.pdf>. Acesso em: 10 out. 2016.

dSPACE. **System Desk Tutorial.** Disponível em: http://www.dspace.com/ww/en/pub/home/products/sw/system_architecture_software/systemdesk.cfm>. Acesso em: 10 out. 2016.

Electromagnetic Compatibility for Electric Vehicles. **CAN bus: Controller Area Network.** 8 set 2015. Disponível em: <http://www.flexautomotive.net/EMCFLEXBLOG/post/2015/09/08/can-bus-for-controller-area-network>. Acesso em: 10 out. 2016.

EHSANI, M.; GAO, Y.; EMADI, A. **Modem Electric, Hybrid Electric, and Fuel Cell Vehicles:** Fundamentals, Theory and Design.2.ed. Boca Raton: FL, CRC Press, 2009.

HAMMARSTRÖM, J.; NILSSON, J. **A Comparision of Three Code Generators for Models Created in Simulink.** 2006. Dissertação (Mestrado) – Chalmers University of Technology, Department of Computer Science and Engineering. Göteborg, 2006.

HEBIG, R. **Methodology and Templates in AUTOSAR.** Technical Report – Hasso-Plattner – Institut für Softwaresystemtechnik, 2009.

HELLINGRATH, B. **Future Automotive Industry Structure 2015.** Mercer Management Consulting und Fraunhofer Gesellschaft, Stuttgart. 2004.

HEINECKE, H.; et al. **AUTOSAR:** current results and preparations for exploitation. In: 7th EUROFORUM Conference Software in the Vehicle.2006, Stuttgart.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO26262-1:2011.** Disponível em: https://www.iso.org/standard/43464.html. Acesso em: 02 de fev. de 2017.

IWAGAYA, T.; YAMAGUCHI, T.; **Speed Improvements for XIL Simulation Based on Symbolic-Algebric Method.** SICE Annual Conference 2013, p.1338-1342, Set 14-17, 2013, Nagoya University, Nagoya, Japan.

KELEMENOVÁ, T.; KELEMEN, M.; MIKOVÁ, L.; PRADA, E.; LIPTÁK, T.; MENDA, F.; MAXIM, V. **Model Based Design and HIL Simulations.** American Journal of Mechanical Engineering, v1, 2013, p.276-281.

KVASER. **Introduction:** The CAN Bus. Disponível em: https://www.kvaser.com/can-protocol-tutorial/. Acesso em: 02 de maio de 2017.

LEE, O. **Part 5: AUTOSAR CAN.** 2014. Disponível em: <http://www.autoelectronics.co.kr/article/articleView.asp?idx=1315>. Acesso em: 04 out. 2016.

LENNON, T. **Model-Based Design for Mechatronics Systems**. 2007. Disponível em: <http://machinedesign.com/archive/model-based-design-mechatronics-systems>. Acesso em: 23 maio 2017.

LEPPLA, G. **Mapping Requirements to AUTOSAR Software Components.** 2008. Dissertação (Mestrado) – Waterford Institute of Technology, Ireland, 2008.

LUO, F.; HUANG, Z. **Embedded C Code Generation and Embedded Target Development Based on RTW-EC.** In: 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT 2010). 07, 2010, Chengdu, China. p. 532-536.

NEME, J. H. Z. **Aplicação do Método de Desenvolvimento Baseado em Modelos para Função de Software Automotivo:** Sistema de Iluminação Externa. Trabalho de Conclusão de Curso. Engenharia Eletrônica –EE. Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2014.

NEME, J. H. Z. **Fluxo de Trabalho para o Desenvolvimento de Funções Embarcadas Automotivas:** Padrão AUTOSAR. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

NISHIKAWA, K.; KAJIO, K.; LANGE, K.; SCHARNHORST, T.; KUNKEL, B. **Achievements and Exploitation of the AUTOSAR Development Partnership.** CTEA, 2006.

NUNES, L. R.; NEME, J. H. Z.; SANTOS, M.M.D.; FRANCO, F.R.; BANIK, T. M.; **Design of a Embedded Software Controller for a Mobile Seat Platform for Commercial Vehicle.** In:12th IEEE/IAS International Conference on Industry

Applications – INDUSCON 2016, 2016, Curitiba.1,v.1.p17. Disponível em: http://www.induscon2016.com.br.

OICA. **Production Statistics 2015.** Disponível em: <http://www.oica.net/category/production-statistics/>. Acesso em: 10 out. 2016.

PLOSS, R.; MUELLER, A.; LETEINTURIER, P. **Solving Automotive Challenges with Electronics.** In: 2008 International Symposium On Technology, Systems and Applications (VLSI-TSA). **Anais of Institute of Electrical and Electronics (IEEE)**, p.327-345, 2008. Disponível em: http://dx.doi.org/10.1109/vtsa.2008.4530772.

PRETSCHNER, A; BROY, M; KRUGER, I. H.; STAUNER, T. **Engineering Automotive Software.** Proceedings of the IEEE, New York, v.5, p.356-373, fev. 2007.

RAMACHANDRAN, M.; CARVALHO, R. A. **Handbook of Research on Software Engineering and Productivity Technologies:** Implications of Globalization. 2009. p. 556.

RAMOS, R. A.; **Introdução a Verificação, Validação e Teste de Software**. Disponível em: http://www.labes.icmc.usp.br. Acesso: 20 de maio de 2017.

RENESAS. **AUTOSAR Layered Architecture.** Disponível em: <https://www.renesas.com/en-us/solutions/automotive/technology/autosar/autosar-layerdarch.html>. Acesso em: 10 out. 2016.

SANTOS, M. M. D. **Redes de Comunicação Automotiva:** Características, Tecnologias e Aplicações. 1. ed. São Paulo, Editora Érica, 2010.

SANTOS, M. M. D.; FRANCO F.; NEME. J. H. Z.; STEVAN, S. L.; LUGLI, A. B.; TORRES, W.; FILHO, J. F. J.; **Rapid Control Prototyping for Automotive Software at Power Windows Systems**. International Journal of Innovative Computing, Information & Control, v.11, p.1341-1356, 2015. Disponível em: http://www.ijicic.org/ijicic-14-09020.pdf.

SCHEID, O. **AUTOSAR Compendium:** Part 1 – Application and RTE. 1 ed. Bruchsal: Createspace Independent Publishing Platform, 2015.

SCHIRRMEISTER, F. **Automotive System & Software Development Challenges – Part 1.** Cadence Design Systems. 05 nov. 2013. Disponível em :< http://www.edn.com/design/systems-design/4423874/1/Automotive-System---Software-Development-Challenges---Part-1>. Acesso em: 10 out. 2016.

SCHNEIDER, A.; SOUZA, F. **Sistemas Embarcados:** Hardware e Firmware na Prática. 2. ed. São Paulo, Editora Érica, 2014.

SHATAT, Tariq S.; ABDULLAH, Bassem A.; SALEM, A. **System C based Simulation of AUTOSAR software components.** In: 10th International Conference On Computer Engineering & Systems (ICCES).2015, Cairo. Anais Institute of Electrical and Electronics Engineers (IEEE), 2015. p. 1-10. Disponível em: http://dx.doi.org/10.1109/icces.2015.7393028.

STELLA, G. N. D. **Aplicando a Metodologia de Desenvolvimento Baseado em Modelos para Funções de Software Automotivo.** 2015.123f. Dissertação (Mestrado) – Programa de Pós-Graduação em Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2015.

TRAN, S; CULLYER, J; HINES, E; MARKS, K. **On the Development of a High Quality Software Design Methodology for Automotive Applications.** IEE Colloquium on Safety Critical Software in Vehicle and Traffic Control, 1990.

VINCENTELLI, A. S. **New Vistas on Automotive Embedded Systems.** Chess Review. Alexandria, VA, 2006.

WARSCHOFSKY, R. **AUTOSAR Software Architecture.** Technical Report – Hasso-Plattner - Institut für Softwaresystemtechnik, 2009.