Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Curso de Graduação em Engenharia Elétrica

JOSÉ IGOR CLEMENTINO FERREIRA MOREIRA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Campina Grande, Paraíba
Junho de 2018

JOSÉ IGOR CLEMENTINO FERREIRA MOREIRA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador:
Rafael Bezerra Correia Lima, Dr.Sc.

Campina Grande, Paraíba
Junho de 2018

JOSÉ IGOR CLEMENTINO FERREIRA MOREIRA

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em _____ / _____ / _____

_____

Professor Rafael Bezerra Correia Lima, Dr.Sc.
Universidade Federal de Campina Grande
Orientador

_____

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

AGRADECIMENTO

Agradeço aos Professores Rafael Bezerra Correia Lima, Péricles Rezende Barros e George Acioli Júnior, pela oportunidade de estagiar no LIEC e pelas orientações prestadas não só durante a realização do estágio, mas durante a maior parte do curso.

Agradeço à minha família, em especial à minha mãe Eliane Regina e minha noiva Cristina Kelly, por apoiar a realização desse sonho.

E por fim, a todos os colegas do Laboratório de Instrumentação Eletrônica e Controle, em especial à Christian Charles Dias e Simões Toledo, cuja ajuda foi de grande importância para a realização desse trabalho.

RESUMO

Este relatório apresenta as atividades realizadas pelo aluno José Igor Clementino Ferreira Moreira durante o Estágio Supervisionado no Laboratório de Instrumentação Eletrônica e Controle (LIEC), pertencente ao Departamento de Engenharia Elétrica (DEE) da Universidade Federal de Campina Grande (UFCG), sob orientação do Professor Rafael Bezerra Correia Lima e supervisão do Professor George Acioli Júnior.

Palavras-chave: Hoverboard, V-Model, Model-based Desing, Simulink, Autocode Generation, Desenvolvimento em Camadas.

LISTA DE FIGURAS

SUMÁRIO

1 Introdução

O objetivo deste relatório é apresentar as atividades desenvolvidas e os resultados alcançados pelo estagiário José Igor Clementino Ferreira Moreira, durante o estágio supervisionado realizado no Laboratório de Instrumentação Eletrônica e Controle (LIEC), situado na Universidade Federal de Campina Grande (UFCG). O período de vigência do estágio supervisionado foi de 05/04/2018 até 17/05/2018 totalizando 180h (6 créditos), sendo uma carga horária de 30 horas semanais, sob a orientação do professor Rafael Bezerra Correia Lima e supervisão do professor George Acioli Júnior.

1.1 O Plano de estágio

A elaboração do plano de estágio foi concebida de forma a possibilitar o desenvolvimento de uma camada de software intermediária para fazer a interface de comunicação entre a camada de aplicação e a camada de drivers de hardware de um hoverboard, requerendo do estagiário a aplicação dos conhecimentos adquiridos nas diversas disciplinas integralizadas durante a graduação, tendo uma maior aplicação os temas abordados nas disciplinas de Introdução à Programação, Técnicas de Programação, Eletrônica, Arquitetura de Sistemas Digitais, Controle Analógico, Controle Digital e Automação Industrial.

1.1.1 Principais atividades

1. Estudo das características de um hoverboard;
2. Estudo dos circuitos eletrônicos de controle do hoverboard utilizado no estágio.
3. Estudo do funcionamento de motores DC Brushless trifásicos.
4. Desenvolvimento do algoritmo de controle de comutação dos motores DC Brushless no Simulink®.
5. Desenvolvimento do protocolo de comunicação entre a camada de aplicação, intermediária e de drivers no Simulink®.
6. Desenvolvimento das funções para calcular posição e velocidade do hoverboard no Simulink®.
7. Explicação dos passos para integrar o código gerado pelo Simulink® às camadas de aplicação e de hardware.

1.1.2 Cronograma

Atividades	Abril				Junho			
1								
2								
3								
4								
5								
6								
7								

2 Local do Estágio

Localizado na Universidade Federal de Campina Grande (UFCG) no campus de Campina Grande, o Laboratório de Instrumentação Eletrônica e Controle (LIEC) é um laboratório pertence ao Departamento de Engenharia Elétrica (DEE). Integrado por professores doutores, alunos de pós-graduação e de graduação, esse laboratório tem como principal objetivo desenvolver atividades e projetos ligados as áreas de automação, controle e instrumentação.

Figura 1 – Fachada do LIEC.

Fonte: http://liec.ufcg.edu.br/, acessado em 11/05/2016

Com uma área de aproximadamente 600 m², o LIEC (Figura 1) conta com oito laboratórios de desenvolvimento, duas salas de apoio técnico, sala para apresentação de trabalhos, salas para pós-graduação e professores. Dentre as principais atividades desenvolvidas no laboratório, pode-se destacar as seguintes (LIEC, 2018):

- Laboratório de Aplicações Wireless - desenvolvimento de soluções baseadas em dispositivos móveis para ambientes industriais;
- Laboratório de Automação Industrial - sintonia de controladores PID industriais (mono e multivariável), automação industrial, instrumentação industrial, IHM industrial, avaliação de confiabilidade em malhas de controle;
- Laboratório de Controle e Otimização - projeto e sintonia de PID, modelagem e simulação de processos, sistemas supervisórios;
- Laboratório de Instrumentação Eletrônica - projeto e sintonia de PID;

- Laboratório de Redes Industriais - estudos de técnicas e tecnologias para a comunicação entre dispositivos industriais;
- Laboratório de RFID - desenvolvimento de aplicações baseadas em tecnologia RFID para ambientes industriais;
- Laboratório de Ultrassom - desenvolvimento de sensor de incrustação, desenvolvimento de técnicas de medição de incrustação.

No LIEC, alunos de pós-graduação e de graduação encontram um ambiente propício ao aperfeiçoamento dos conhecimentos teóricos e das habilidades práticas, por intermédio das diversas pesquisas e outras atividades realizadas no mesmo.

3 Atividades Realizadas

Conforme mencionado anteriormente, as atividades do estágio foram determinadas de forma a proporcionar ao estagiário um ambiente capaz de solidificar os conhecimentos adquiridos no decorrer da graduação, oferecendo um ambiente de trabalho, no qual, esses conhecimentos deveriam ser utilizados na elaboração de uma solução para um problema real de engenharia.

O estagiário foi encarregado de realizar o desenvolvimento de uma camada de software intermediária que proporcione a integração e comunicação da uma camada de aplicação com a camada de drivers de hardware do hoverboard.

Para o desenvolvimento da solução, pode-se relacionar as atividades realizadas em três grupos:

- Estudo do hoverboard e de seus circuitos de controle, bem como do controle de comutação do Motor DC Brushless trifásicos;
- Desenvolvimento do firmware para o sistema embarcado.
- Apresentação dos passos de integração do código gerado com a camada de aplicação e de drivers de hardware.

3.1 Hoverboard

Tecnicamente, um hoverboard é uma plataforma flutuante (que se parece com uma skate sem rodas que flutua), que pode ser usado para o transporte de pessoas. O termo hoverboard foi criado para o filme *Back to the Future II*, onde o protagonista Marty McFly viajou para o futuro e descobriu que os adolescentes estavam andando em plataformas flutuantes sem rodas, os "*hoverboards*". Apesar de um *self-balancing scooter*, o nome técnico correto para essa plataforma, não levitar como um hoverboard, as pessoas adotaram o termo "hoverboard" como uma forma coloquial para se referir a esses veículos, principalmente porque ele soa agradável.

3.1.1 Funcionamento

Um hoverboard tem alguns componentes principais que são indispensáveis para o seu correto funcionamento, entre eles: um giroscópio para determinar o ângulo de inclinação, ângulo *pitch*, motores para realizar o movimento para frente e para trás e de curvas da plataforma, microcontroladores para gerenciar toda a lógica de controle

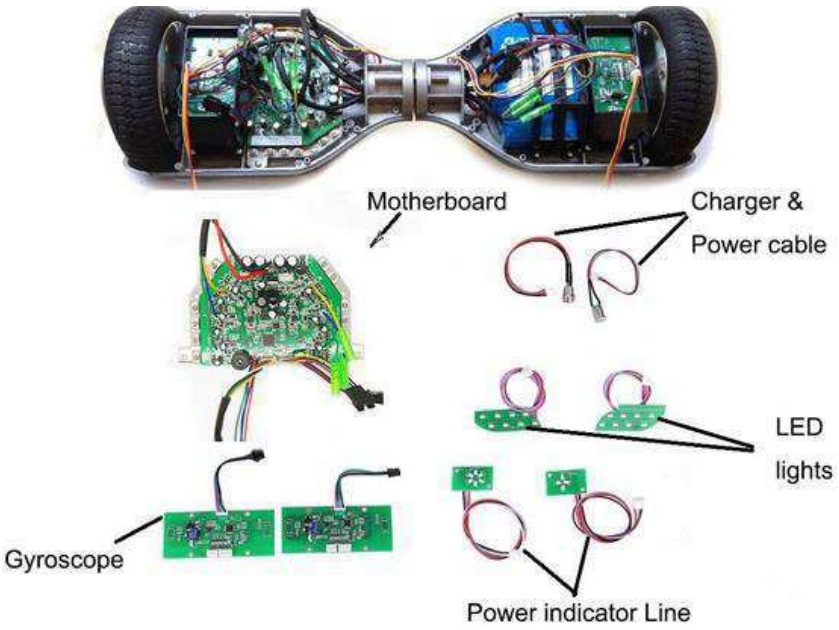
do hoverboard, controlando assim o movimento dos motores, e baterias de grande capacidade para prover a energia necessária para o funcionamento do hoverboard. Na Figura 3 temos um visão geral de um hoverboard e de seus componentes principais.

Figura 2 - a) Hoverboad do filme *Back to the Future II.* b) Self-balancing scooter, coloquialmente chamado de hoverboard.

(a) (b)

Fonte: Google Imagens .

Figura 3 - Hoverboard e partes integrantes.



Fonte: https://hoverboardrepair.com .

Como sabemos a principal função de um hoverboard é manter-se equilibrado na posição vertical. Com esse objetivo, os microcontroladores embarcados no hoverboard monitoram a direção que o piloto está se inclinando, e assim ativam os motores para realizar o movimento na direção e velocidade correta. O hoverboard pode ser caracterizado em alto nível como um sistema de controle em malhar fechada, onde a entrada do sistema é a direção e velocidade, um controlador que calcula as variáveis de controle dos atuadores, os atuadores são os motores, e feedback é o ângulo de inclinação e velocidade.

3.1.2 Legalidade de uso

Na maioria dos países não existem leis que regulamente o uso de hoverboard, sendo assim as pessoas compram e usam esses equipamentos de forma deliberada, muitas vezes de forma perigosa. Alguns países como os Estados Unidos e Inglaterra já possuem leis que limitam o uso dessas plataformas ao local privado, não sendo permitido o seu uso em vias públicas como calçadas, ruas, estradas, e estabelecimentos públicos. No que se refere ao Brasil, até o presente momento, maio de 2018, não existe leis que regulamentam o uso do hoverboard. É importante salientar o bom senso ao se usar esses equipamentos e procurar bastante informação antes de se adquirir um, de forma a garantir a qualidade, segurança de fabricaçõ e suporte técnico do hoverboard que se está comprando.

3.2 Estudo dos circuitos de controle do Hoverboard

Para esse estágio, todo o firmware da camada intermediária de software que foi desenvolvido e será apresentado posteriormente, foi criado com o intuito de ser usado no controle do hoverboard adquirido pelo LIEC.

Como o fabricante desse equipamento não fornece junto com o manual, os dados dos circuitos elétricos, foi necessário abrir o equipamento e realizar o trabalho de engenharia reversar para identificar todos os circuitos de controle, bem como seus funcionamentos. Essa atividade foi realizada pelo mestrando Christian Charles Dias. De posse dos diagramas elétricos dos circuitos de controle, foi possível pensar e

desenvolver a camada de software específica para o hoverboard comprado pelo laboratório.

A seguir será apresentado os principais circuitos que fazem parte da plataforma adquirida.

3.2.1 Microcontrolador

O microcontrolador utilizado para controle do hoverboard é o STM32F103RCT6, da STMicroelectronics. Na Figura 4, temos a representação do diagrama de blocos com os principais recursos desse microcontrolador.

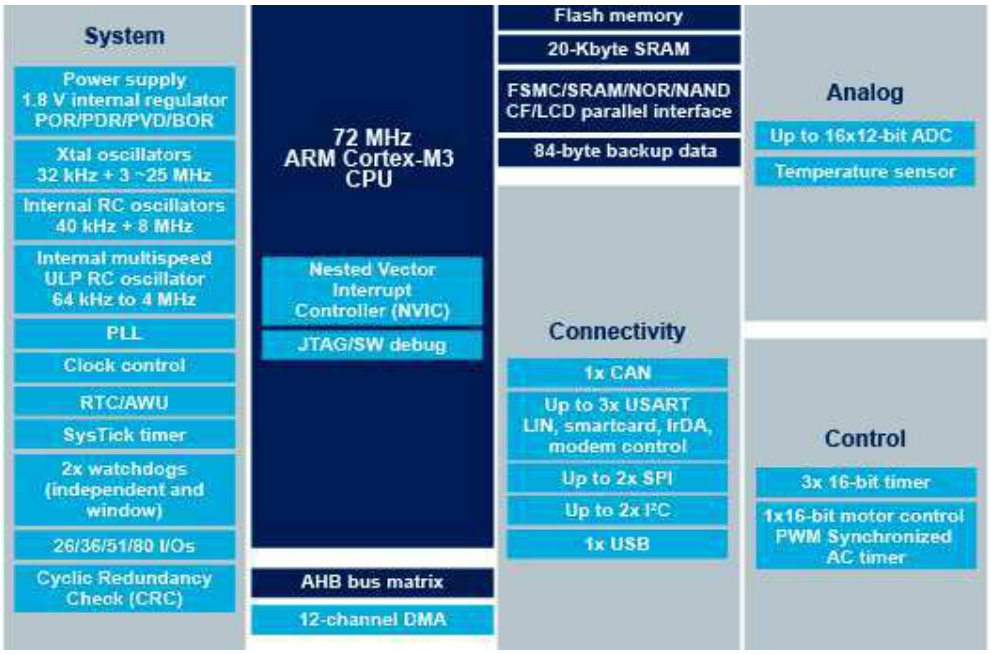
Na Figura 5, temos o esquema eletrônico desse microcontrolador, onde se pode destacar quais pinos são utilizados para o controle do hoverboard.

3.2.2 Circuito de alimentação

Para alimentar o hoverboard, um bateria de 36 volts é usada. Como diferentes partes dos circuitos funcionam com níveis de tensão de 15volts e 3.3 volts, circuitos reguladores de tensão são usados para obter-se esses níveis de tensão.

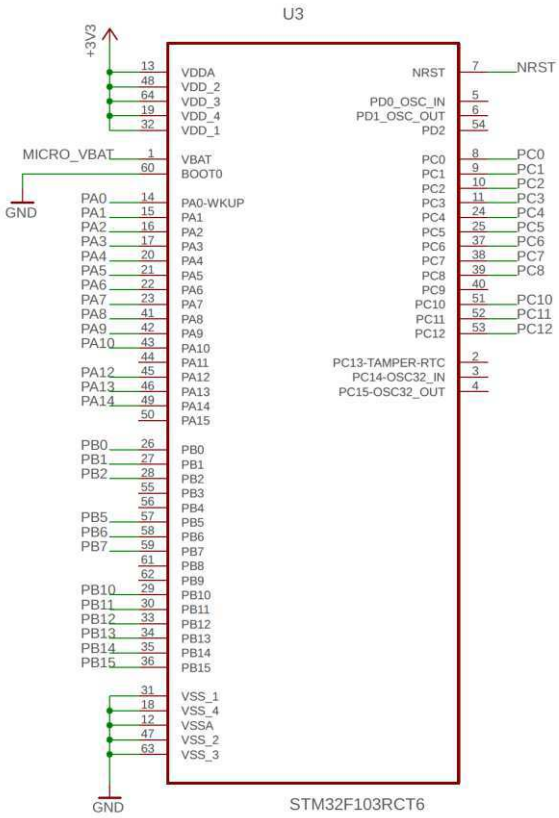
Na Figura 6, temos o esquema elétrico do circuito de alimentação, da lógica de ligar e desligar o hoverboard e da regulação de tensão para 15 volts, enquanto na Figura 7, temos o circuito de regulação de tensão para 3.3 volts.

Figura 4 - Recursos do STM32F103RCT6.



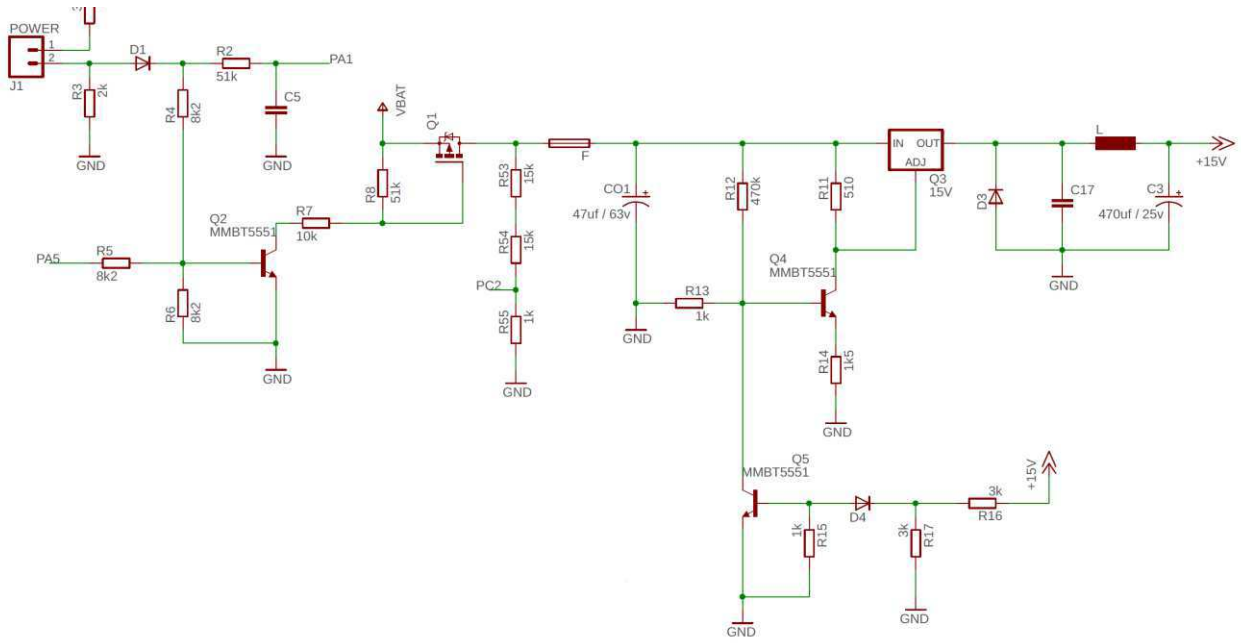
Fonte: http://www.st.com .

Figura 5 - Esquema elétrico STM32F103RTC6.



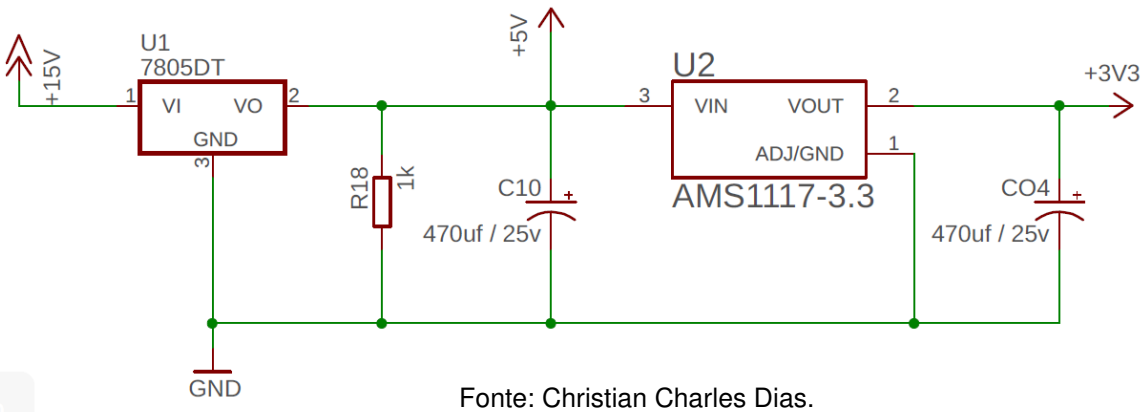
Fonte: Christian Charles Dias.

Figura 6 - Circuito de alimentação e regulação de 15 volts.



Fonte: Christian Charles Dias.

Figura 7 - Circuito de regulação de tensão para 3.3 volts.

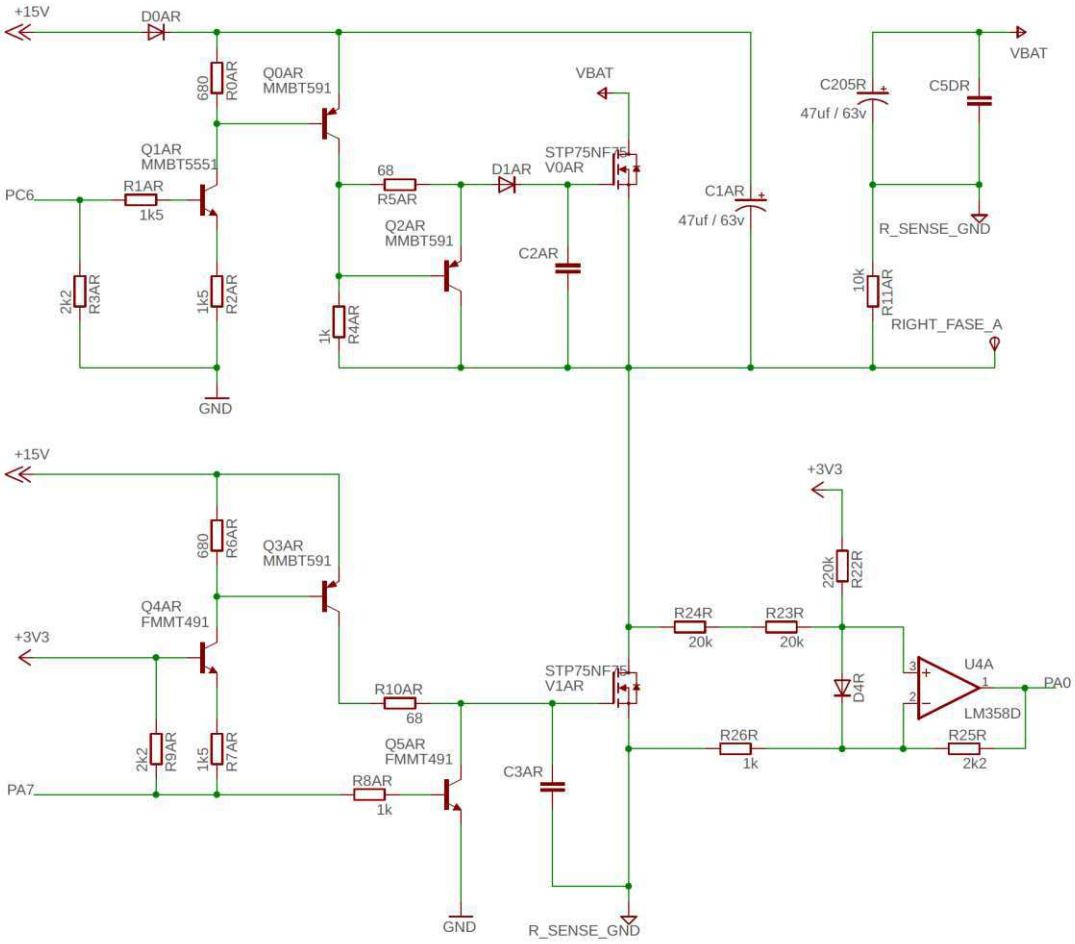


Fonte: Christian Charles Dias.

3.2.3 Circuito de comutação das fases

Como será apresentado na secção “Estudo do motor DC brushlees”, para realizar o controle da direção e velocidade de um motor DC Brushless, tipo de motor usado no hoverboard em estudo, é necessário um circuito eletrônico de comutação que permita a comutação das fases dos motores que são energizadas. Como os motores utilizados pelo hoverboard são trifásicos, o circuito de comutação de cada motor realiza o controle de três fases. Na Figura 8, temos a representação do circuito de controle de apenas uma das fases do motor direito, a fase A. Para as outras duas fases do motor esquerdo e 3 fases do motor direito, o mesmo circuito é utilizado, o que vai mudar são os pinos de controle dos transistores, que no caso da fase A do motor direito são os pinos PC6 e PA7.

Figura 8 - Circuito de controle de comutação da fase A do motor direito.

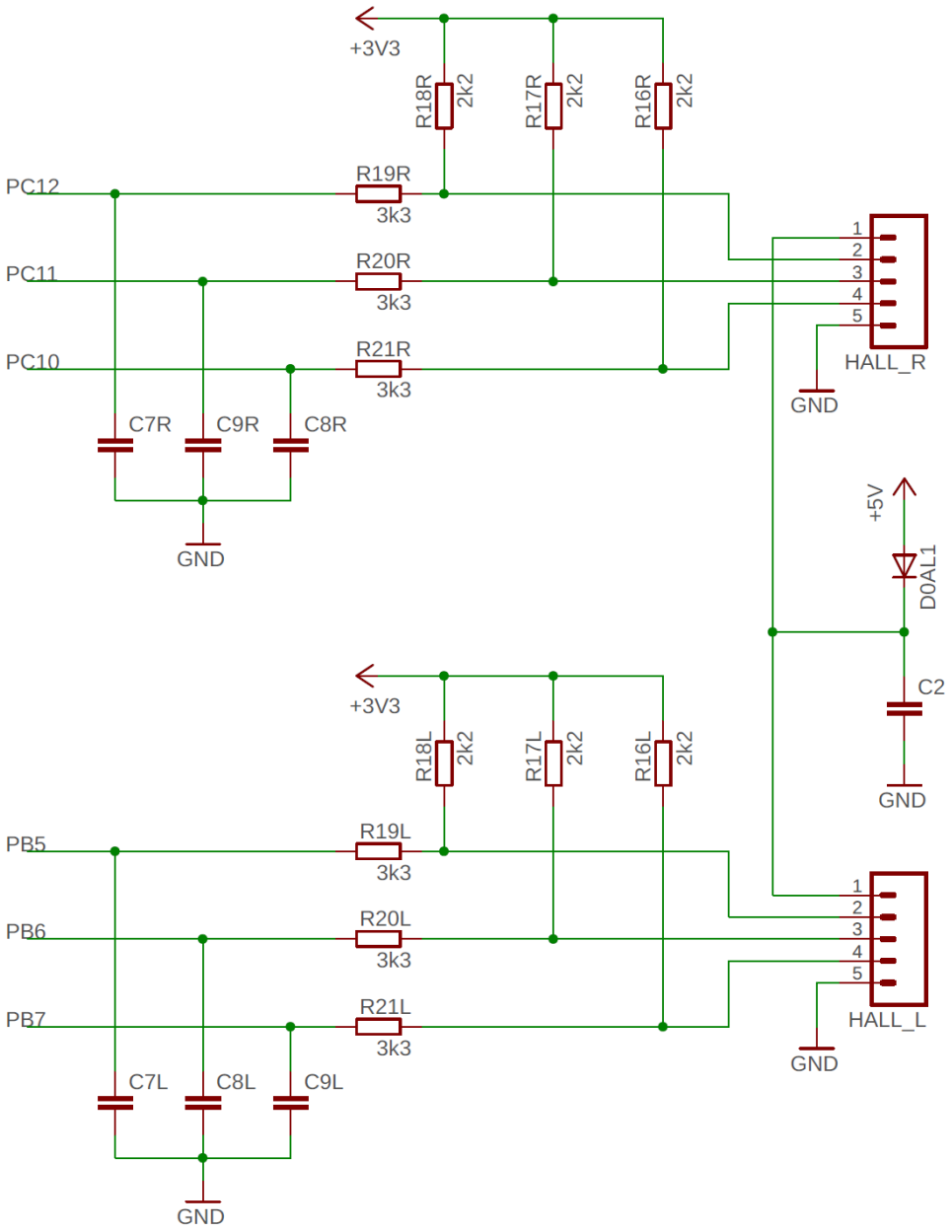


Fonte: Christian Charles Dias.

3.2.4 Circuito dos sensores Hall

Como também será visto da secção “Estudo do motor DC brushlees”, sensores Hall são utilizados para determinar a posição exata do rotor do motor, de forma que o algoritmo de controle possa determinar a sequência exata de comutação das fases para que o motor possa rotacionar continuamente tanto no sentido horário, como anti-horário. O esquema eletrônico da Figura 9, mostra como os sensores Hall do motor esquerdo (HALL_L na Figura 9) e direito (HALL_R na Figura 9) são ligados ao microcontrolador por meio dos pinos PC12, PC11, PC10, PB5, PB6 e PB7.

Figura 9 - Circuito dos sensores Hall do motor esquerdo e direito.



Fonte: Christian Charles Dias.

3.2.5 Circuito de monitoramento de corrente

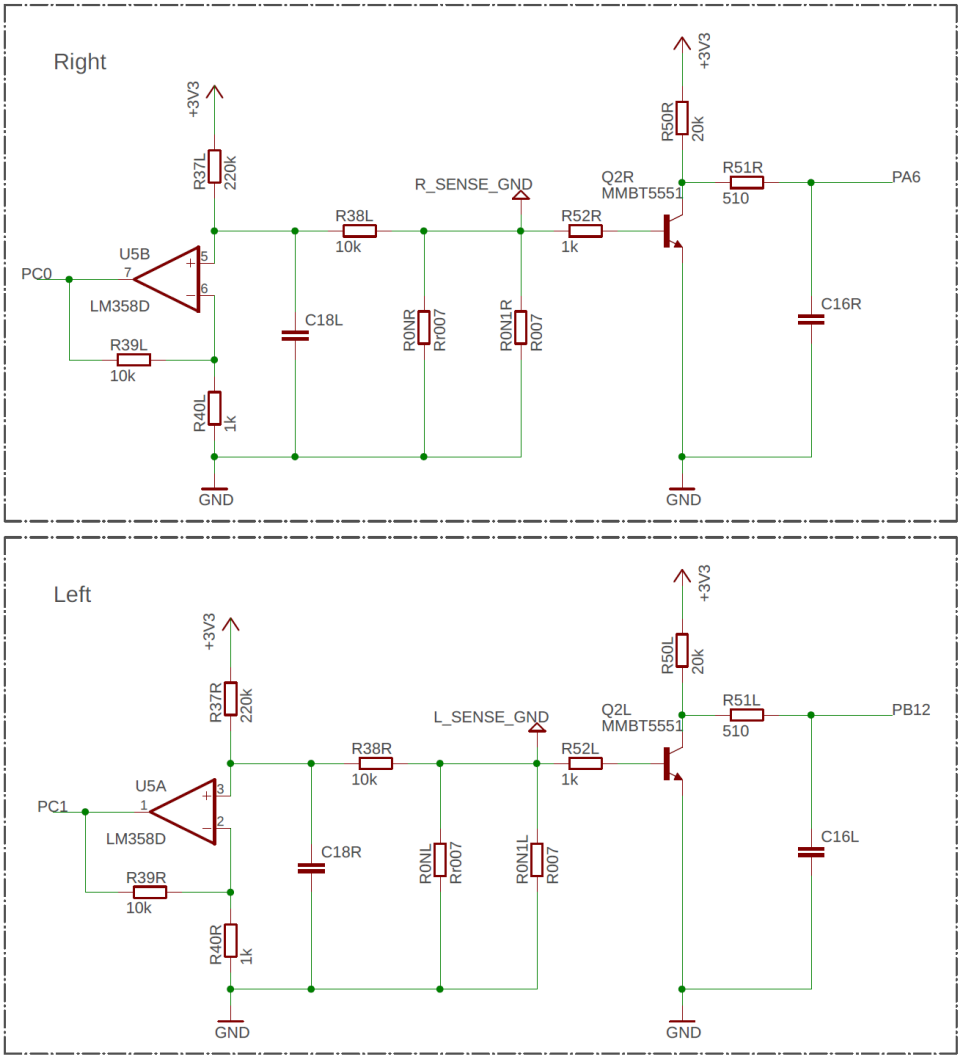
Para proteger os motores bem como outras partes da placa de controle de danificação definitiva, dois circuitos para monitorar a corrente que passa por cada um dos motores são utilizados. O algoritmo de controle pode monitorar continuamente o nível de corrente, e caso este passe de níveis aceitáveis, pode interromper a

alimentação de todo o sistema. Os esquemas elétricos desses circuitos são mostrados na Figura 10.

3.2.6 Circuitos de sinalização

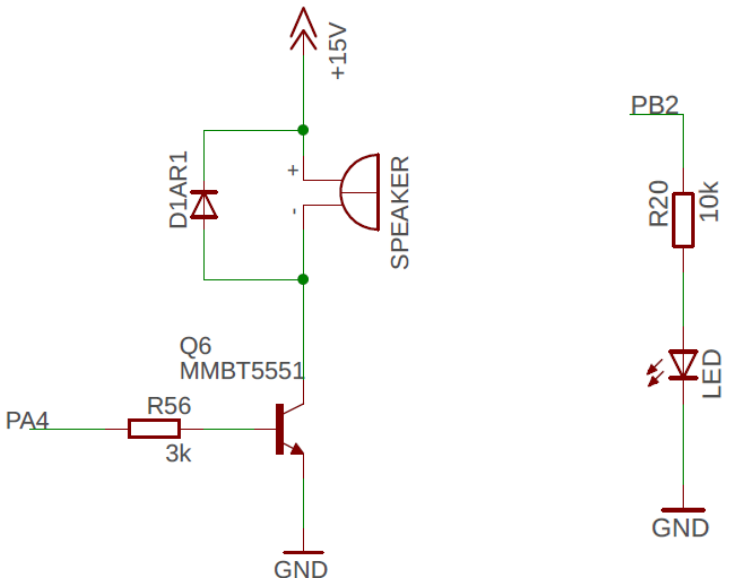
Para fazer utilização de sinais sonoros e luminosos, estão presentes um circuito de acionamento de buzzer e de um led, como pode ser visto na Figura 11.

Figura 10 - Circuitos de monitoramento de corrente.



Fonte: Christian Charles Dias.

Figura 11 - Circuito de acionamento do buzzer à esquerda e do led à direita.



Fonte: Christian Charles Dias.

3.3 Estudo do motor DC brushless

O motor DC Brushless, comumente conhecido pela sigla BLDC é amplamente usado em aplicação automotivas, aeroespaciais, equipamentos de automação industrial e instrumentação.

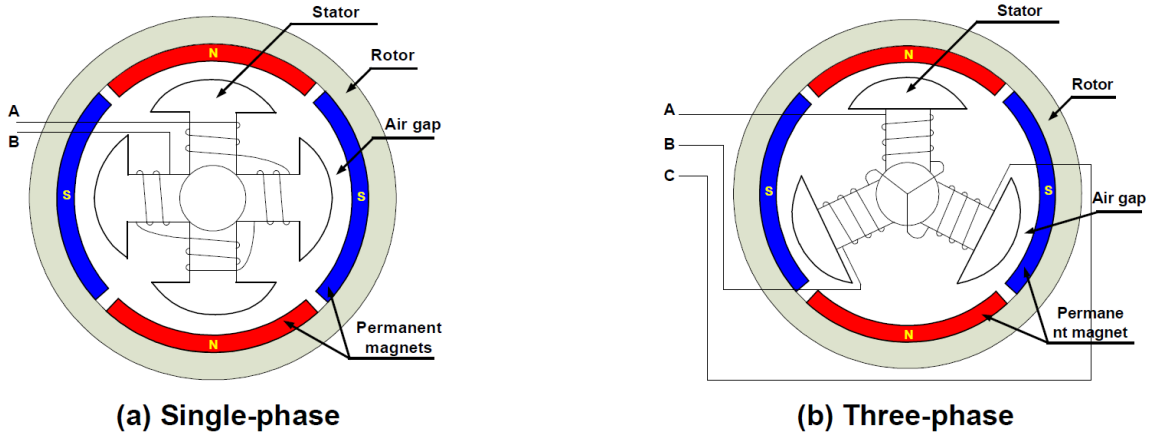
Comparado com o motor DC brushed ou um motor de indução, o motor BLDC tem muitas vantagens, dentre as quais:

- Mais eficiente;
- Baixo ruído acústico.
- Menor e mais leve.
- Melhor dinâmica de resposta.
- Melhor curva característica de velocidade *vs* torque.
- Maior gama de velocidade.
- Maior vida útil.

3.3.1 Estator

Baseado no número de enrolamentos do estator, existe três classificações para o motor BLDC: monofásico, bifásico e trifásico. Assim sendo, o motor monofásico, bifásico e trifásico, possuem respectivamente, um, dois ou três enrolamentos no seu estator. Os motores BLDC comumente usados são os monofásicos e trifásicos. Na Figura 12, temos uma representação simplificada de um BLDC monofásico e trifásico, cada um contendo um rotor com dois pares de polos magnéticos.

Figura 12 - Diagrama simplificado de BLDC monofásico e trifásico.



Fonte: Jian Zhao/Yangwei Yu, Brushless DC Motor Fundamentals, 2011

O motor monofásico possui um enrolamento no estator, enrolados em ambos os sentidos, horário e anti-horário, para produzir quatro polos magnéticos como mostrado na Figura 12 (a). Em comparação, como pode ser visto na Figura 12 (b), o motor trifásico tem três enrolamentos, onde cada fase comuta sequencialmente para fazer o rotor girar.

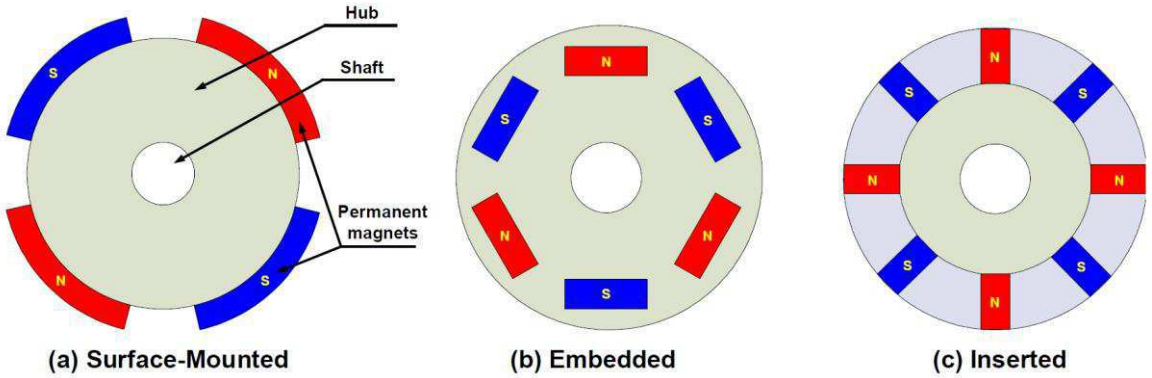
Podemos classificar o enrolamento do BLDC em dois tipo: trapezoidal e sinusoidal. Essa classificação diz respeito a forma de onda do sinal da força eletromotriz induzida, do inglês , *back electromotive force* (BFEM). A forma da BFEM depende de como as boninas do enrolamento são interconectadas e da distância de entreferro ou *air gap*. A corrente de cada fase do motor BLDC também segue uma forma de onda trapezoidal ou sinusoidal. Os motores com BFEM sinusoidal produzem um torque eletromagnético mais suave do que motores com BFEM trapezoidal, porém,

devido ao alto custo do primeiro, geralmente os motores com BFEM trapezoidal são mais utilizados na indústria.

3.3.2 Rotor

O rotor consiste de um eixo e de uma estrutura com imãs magnéticos arranjados para formar pares de polos magnéticos. Cada par de polos é constituído por um polo norte magnético e um polo sul magnético. A quantidade de polos magnéticos de um rotor varia de motor para motor a depender da especificação do projeto, sendo que motores com mais pares de polos, geralmente possuem uma curva de torque mais suave, além de permitir um melhor controle da posição do eixo do motor. Nas Figura 13 temos uma representação de três tipos diferente de arranjo do rotor de um BLCD.

Figura 13 - Diferentes tipos de configuração de rotores para motor BLDC.



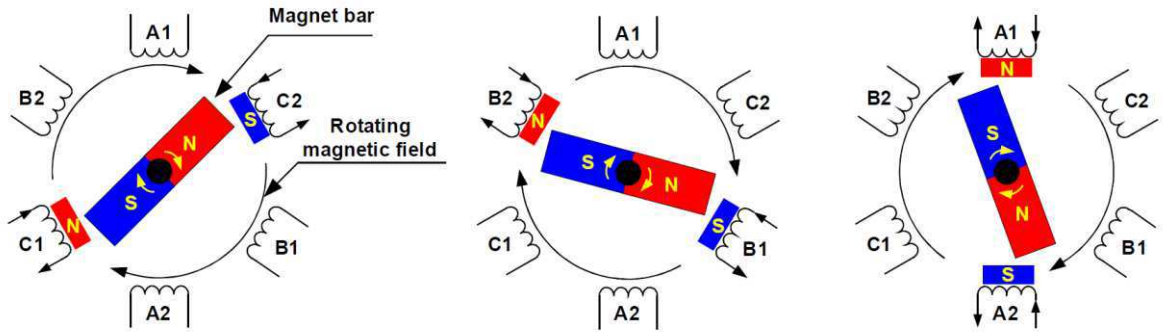
Fonte: Jian Zhao/Yangwei Yu, Brushless DC Motor Fundamentals, 2011.

3.3.3 Operação

A operação do motor BLDC é baseada na atração e repulsão entre os polos magnéticos. Usando como exemplo um motor trifásico, como mostrado na Figura 14, o processo começa quando a corrente flui em uma das três fases dos enrolamentos do estator, gerando um polo magnético que atrai o polo magnético oposto do rotor mais próximo. O rotor continuará movendo-se a medida que a corrente é desviada para o enrolamento adjacente do estator. A comutação sequencial e ininterrupta dos

enrolamentos do estator, desde que feito na ordem correta, fará com que o motor continue seu movimento de rotação. O torque produzido pelo motor depende da amplitude da corrente, bem como no número de voltas dos enrolamentos do estator, da força e tamanho dos imãs magnéticos permanentes do rotor, do entreferro entre o rotor e os enrolamentos, e do tamanho do eixo de rotação.

Figura 14 - Exemplo de rotação de um motor BLDC trifásico.



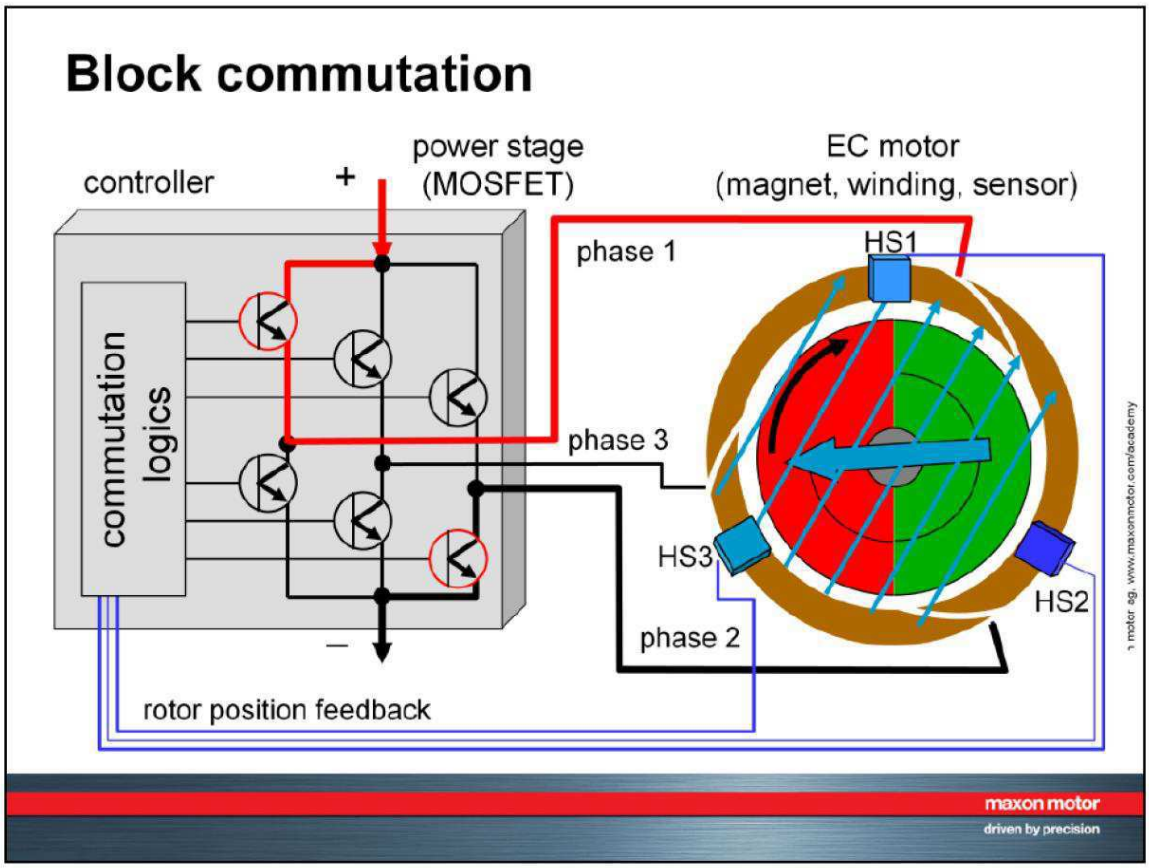
Fonte: Jian Zhao/Yangwei Yu, Brushless DC Motor Fundamentals, 2011.

Para realizar a comutação eletrônica na ordem correta, isto é, saber quem deverá ser a próximo enrolamento do estator que precisa ser energizado, um sensor que indique a posição atual do rotor é utilizado. Geralmente para motores BLDC trifásicos, um conjunto com três sensores Hall são utilizados. Na Figura 15 pode ser visto um esquema de controle de um motor BLDC trifásico, com seu circuito de comutação e os três sensores Hall usados para indicar a posição atual do rotor.

Os motores BLDCs usam comutadores eletrônicos para realizar a comutação de corrente, e assim rotacionar continuamente o motor. Esse comutadores são normalmente conectados em um circuito com estrutura de ponte-H quando se trata de motor BLDC monofásico e em uma estrutura semelhante a ponte-H mas com três fases, conforme pode ser visto na Figura 16, quando se trada de motor BLDC trifásico. Geralmente os circuitos comutadores são controlador por sinais PWMs (Pulse-width modulation), que converte a tensão contínua em uma tensão modulada, de forma a controlar eficientemente os picos de corrente de partida, a velocidade e torque do motor. Quando se usa sinal PWM para controle do circuito comutador é importante se ater a alguns detalhes: aumentar a frequência do sinal PWM acarreta o aumento das perdas por chaveamento do transistores, ao passo que diminuir a frequência do

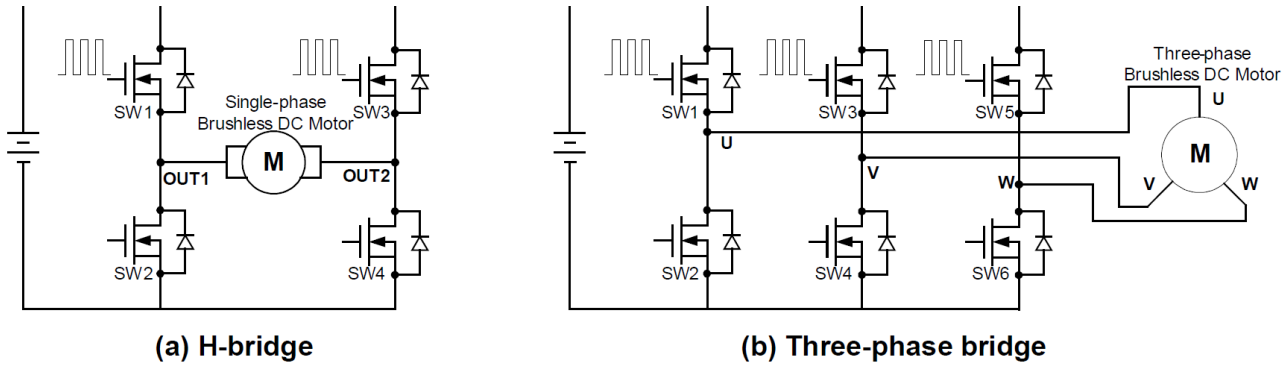
sinal PWM leva a diminuição da banda de frequência do sistema e pode aumentar o *ripple* de corrente ao ponto que pode ser destrutivo para o motor.

Figura 15 - Esquema de controle de um motor BLDC com sensores Hall.



Fonte: Xiaotian Li, Model-Based Design, Brushless DC Motor, Motion Control, 2015.

Figura 16 - Circuito de comutação para motores BLDC monofásico e trifásico.



Fonte: Jian Zhao/Yangwei Yu, Brushless DC Motor Fundamentals, 2011.

Como mencionado anteriormente, um motor BLDC trifásico requer três sensores Hall para detectar a posição do rotor. Na maioria dos casos, os sensores Hall são embarcados na estrutura do estator. Baseado na posição física do sensores Hall no estator, existem dois tipos de saída de sinais dos sensores: sinais defasados em 60° e sinais defasados de 120°. Analisando a combinação dos três sensores Hall é possível determinar a exata posição do rotor, bem como a sequência que comutação correta.

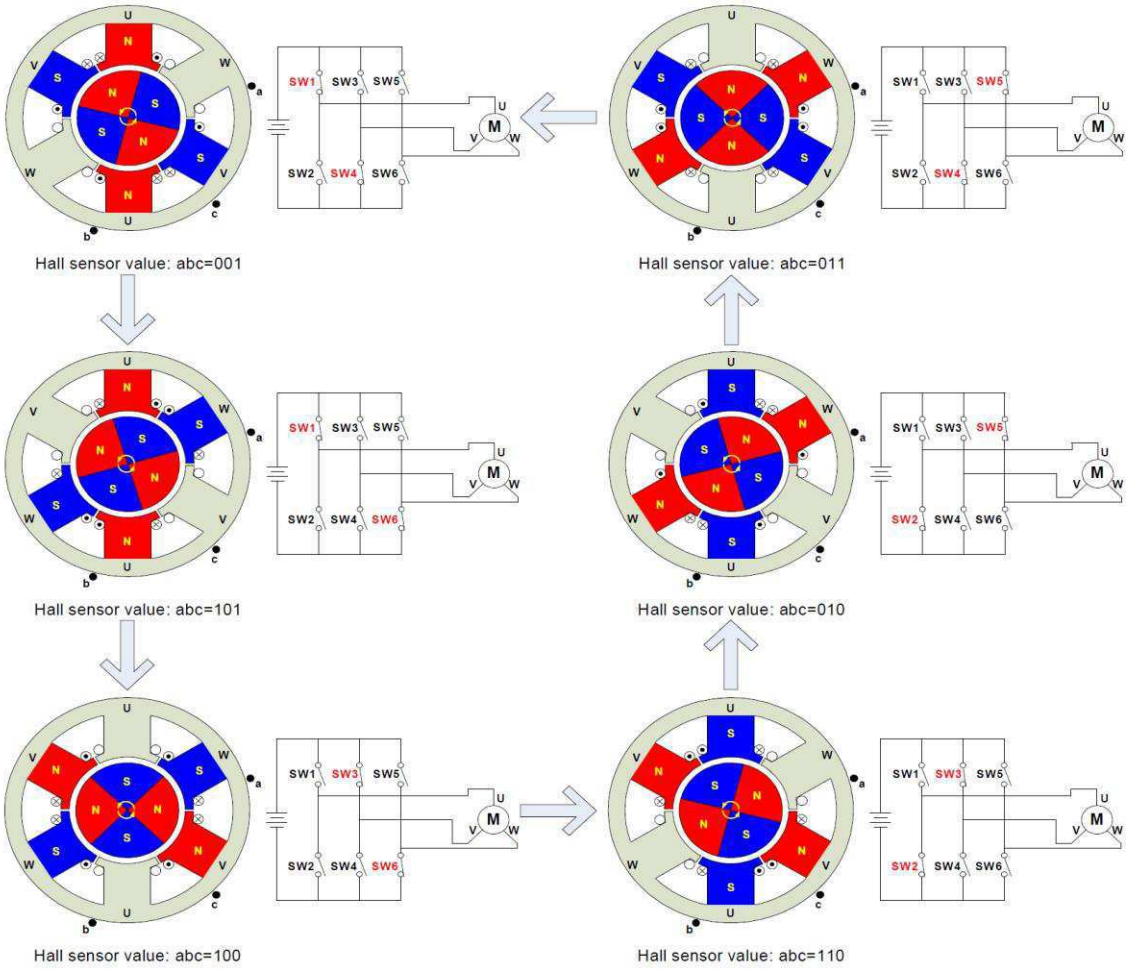
A Figura 17, mostra a sequência de comutação para um motor BLDC trifásico, rotacionando no sentido anti-horário. Os três sensores Hall, “a”, “b” e “c”, estão montados em intervalos de 120° elétricos, enquanto os três enrolamentos do estator estão montados em forma de estrela. Para cada 60° elétricos, um dos três sensores Hall muda de estado, ou passando de zero para um ou de um para zero. Para ocorrer uma rotação elétrica completa, isto é 360° elétricos, são necessários seis passos de comutação, cada passo sendo responsável por uma rotação de 60° elétricos. Em cada passo de comutação, sempre por uma das fases entrará corrente, a fase conectada ao terminal positivo, por outra sairá corrente, a fase conectada ao terminal negativo, e a outra fase ficara flutuante, isto é, nem conectada ao terminal positivo nem ao terminal negativo.

É importante destacar a diferença entre ciclo elétrico e ciclo mecânico. Um ciclo elétrico, isto é, 360° elétricos, pode não corresponder a um ciclo mecânico, 360° mecânicos. O número de ciclos elétricos para completar um ciclo mecânico ou rotação mecânica, vai depender do número de pares de polos magnéticos no rotor, sendo que o número de ciclos elétricos igual ao número de pares polos magnéticos no rotor.

Na Figura 18, temos um diagrama temporal, mostrando o como as fases dos enrolamentos, U, V e W, ficam energizadas ou flutuantes baseado na leitura dos sensores Hall, “a”, “b” e “c”. Esse é um exemplo onde os sensores Hall possuem uma defasagem de 120° no sinal gerado, um em relação ao outro, com o motor rotacionando no sentido anti-horário. Para controlar a velocidade de rotação do motor, é recomendado, como já mencionado anteriormente, usar sinais PWM com frequência bem maiores que a frequência máxima de rotação do motor, no mínimo 10 (dez) vezes maior. Como a tensão de alimentação do circuito de comutação pode ser maior que a tensão nominal do motor que se deseja controlar, outra vantagem de se usar sinal

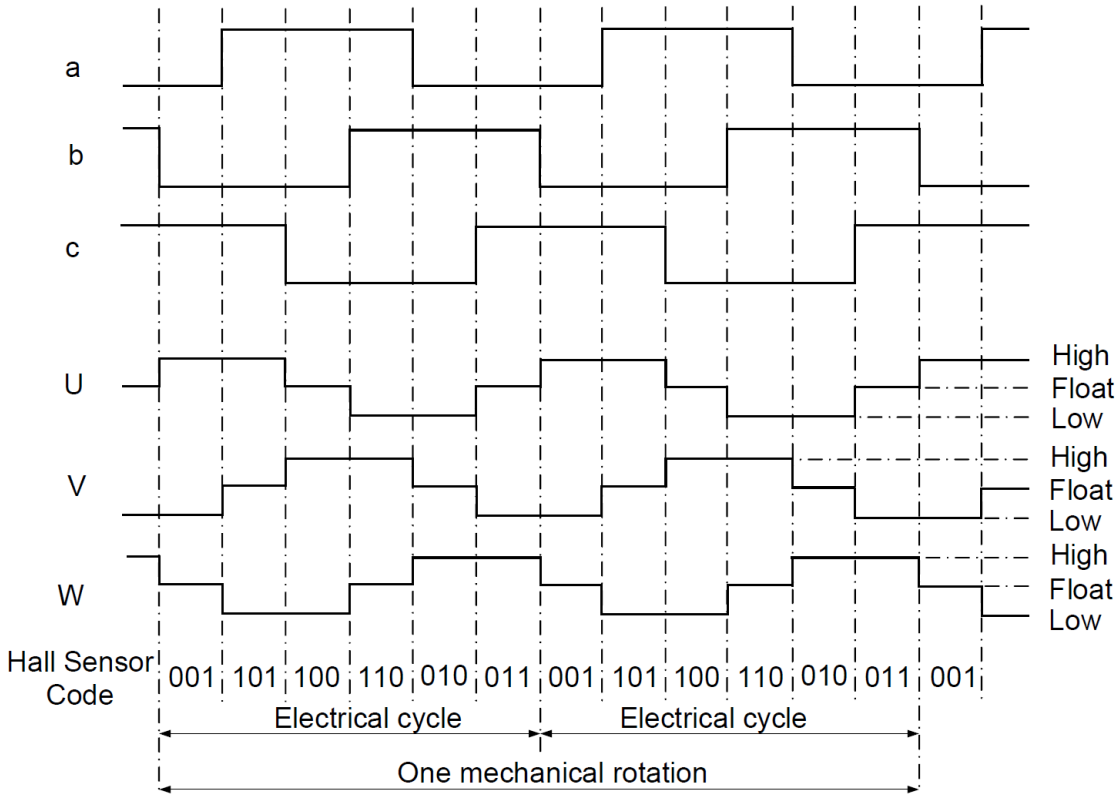
PWM é que se pode controlar o nível de tensão entregue nos terminais do motor em cada passo de comutação.

Figura 17 - Sequência de comutação para um motor BLDC trifásico. Comutação ocorre de acordo com os valores dos sensores Hall.



Fonte: Jian Zhao/Yangwei Yu, Brushless DC Motor Fundamentals, 2011.

Figura 18 - Diagrama temporal da rotação de um motor BLDC trifásico de 2 polos.



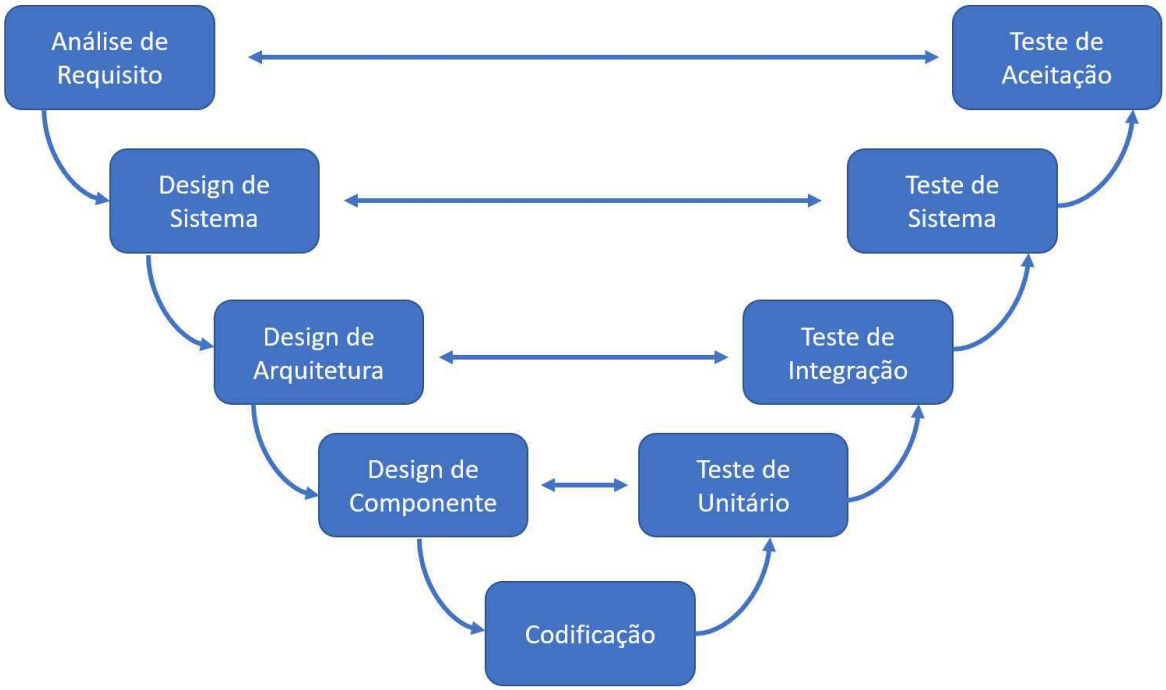
Fonte: Jian Zhao/Yangwei Yu, Brushless DC Motor Fundamentals, 2011.

4 Desenvolvimento do firmware

Nesse estágio, adotou-se duas metodologias para o desenvolvimento da camada intermediária do firmware que irá ser embarcado no hoverboard. Essas duas metodologias são o V-Model e o Model-Based Design que são complementares, e foram adotadas por se encaixarem perfeitamente ao tipo de projeto proposto.

Nas sessões seguintes será apresentado um pouco de cada metodologia e como elas foram utilizadas no desenvolvimento do projeto do estágio.

Figura 19 - V-Model



Fonte: Próprio autor.

4.1 V-Model

O V-Model é uma metodologia de desenvolvimento de software onde a execução do processo acorre de forma sequencial. Essa metodologia é também conhecida como modelo de Verificação e Validação.

Para cada fase de desenvolvimento, lado esquerdo do V, temos uma fase de teste associada, lado direito do V. Isso significa que para cada fase do ciclo de

desenvolvimento, existe uma fase de teste diretamente associada. No V-Model todas as etapas são bem definidas e só se pode passar para etapa seguinte uma vez que a etapa anterior foi concluída.

Na Figura 19 - V-ModelFigura 19 temos uma representação do ciclo completo do V-Model e suas etapas. O lado esquerdo é caracterizado pelas etapas de verificação e o lado direito pelas etapas de validação. A etapa de implementação, ou codificação, faz a junção das etapas de verificação com as etapas de validação.

4.1.1 Análise de Requisitos

Essa é a primeira etapa do ciclo de desenvolvimento, onde os requisitos do produto são entendidos do ponto de vista do cliente. Essa etapa envolve uma comunicação detalhada com o cliente para entender suas expectativas e os requisitos exatos. A atividade de levantamento de requisitos é muito importante e deve ser bem gerenciada pois, na maioria das vezes o cliente não sabe o que realmente quer. O teste de aceitação, do inglês, *acceptance test design*, é desenvolvido nessa parte baseado nas requisitos coletados. Esse teste de aceitação é um acordo feito com o cliente, caso o teste de aceitação seja validado com sucesso, o projeto deve ser aceito pelo cliente.

4.1.2 Design de Sistema

Depois de ter os requisitos do produto claros e detalhados, é hora de projetar o sistema completo. O design do sistema terá a compreensão e detalhamento da configuração completa de hardware e comunicação do produto em desenvolvimento. O plano de teste do sistema é desenvolvido com base no design do sistema. Fazer isso em um estágio anterior deixa mais tempo para a execução real do teste mais tarde.

4.1.3 Design da arquitetura

Especificações da arquitetura do sistema são entendidas e projetadas nesta fase. Geralmente, mais de uma abordagem técnica é proposta e, com base na viabilidade técnica e financeira, a decisão final é tomada. O design do sistema é dividido em módulos, ocupando diferentes funcionalidades. Isso também é conhecido como *High Level Design* (HLD), em português, design de alto nível.

A transferência de dados e a comunicação entre os módulos internos e com o mundo externo (outros sistemas) é claramente entendida e definida neste estágio.

Com essas informações, os testes de integração podem ser projetados e documentados durante esse estágio.

4.1.4 Design de módulo

Nessa fase, o design interno detalhado de todos os módulos do sistema é especificado, conhecido como design de baixo nível. É importante que o design seja compatível com os outros módulos na arquitetura do sistema e nos outros sistemas externos. Os testes de unidade são uma parte essencial de qualquer processo de desenvolvimento e ajudam a eliminar as falhas e erros críticos em um estágio muito inicial. Estes testes unitários podem ser projetados neste estágio com base nos designs internos de cada módulos.

4.1.5 Implementação ou codificação

A codificação real dos módulos do sistema projetados na fase de design é retomada na fase de codificação. A melhor linguagem de programação adequada é decidida com base nos requisitos de arquitetura e sistema.

A codificação é realizada com base nas diretrizes e padrões de codificação. O código passa por várias revisões de código e é otimizado para melhor desempenho antes que a compilação final seja colocada em algum repositório.

4.1.6 Teste unitário

Testes unitários são projetados na fase de design de módulo e são executados no código durante essa fase de validação. O teste unitário é o teste em nível de código e ajuda a eliminar erros em um estágio inicial, embora todos os defeitos não possam ser descobertos por testes de unidade.

4.1.7 Teste de integração

O teste de integração está associado à fase de design da arquitetura. Testes de integração são realizados para testar a coexistência e comunicação dos módulos internos dentro do sistema.

4.1.8 Teste de sistema

O teste do sistema está diretamente associado à fase de design do sistema. Testes de sistema verificam toda a funcionalidade do sistema e a comunicação do sistema em desenvolvimento com sistemas externos. A maioria dos problemas de compatibilidade de software e hardware pode ser descoberta durante a execução do teste do sistema.

4.1.9 Teste de aceitação

O teste de aceitação está associado à fase de análise de requisitos de negócios e envolve o teste do produto no ambiente do usuário. Os testes de aceitação revelam os problemas de compatibilidade com os outros sistemas disponíveis no ambiente do usuário. Ele também descobre os problemas não funcionais, como defeitos de carga e desempenho no ambiente real do usuário.

4.1.10 Quando usar o V-Model

O V-Model é um modelo onde as etapas de projeto são bem definidas e sequenciais. Os requisitos precisam ser muito claros antes do início do projeto, porque geralmente é caro voltar e fazer alterações. Este modelo é utilizado no campo do desenvolvimento médico, automotivo e aeroespacial, pois são domínios estritamente disciplinados.

Os pontos abaixo mostram cenários que são bem cotados para utilização do V-Model.

- Os requisitos são bem definidos, claramente documentados e corrigidos.
- A definição do produto é estável.
- A tecnologia não é dinâmica e é bem compreendida pela equipe do projeto.
- Não há requisitos ambíguos ou indefinidos.
- O projeto é curto.

4.1.11 Prós e contras do V-Model

A vantagem do método V-Model é ser muito fácil de entender e aplicar. A simplicidade deste modelo também facilita o gerenciamento. A desvantagem é que o modelo não é flexível para mudanças e, caso haja uma mudança de requisito, o que é muito comum no mundo dinâmico de hoje, torna-se muito caro fazer a alteração.

As vantagens do V-Model são as seguintes:

- Este é um modelo altamente disciplinado e as fases são completadas uma de cada vez.
- Funciona bem para projetos menores, onde os requisitos são bem compreendidos.
- Simples e fácil de entender e usar.

- Fácil de gerenciar devido à rigidez do modelo. Cada fase tem resultados específicos e um processo de revisão.

As desvantagens do V-Model são as seguintes:

- Alto risco e incerteza.
- Não é um bom modelo para projetos complexos.
- Não se adequa bem para projetos longos e em andamento.
- Não é adequado para os projetos em que os requisitos apresentam um risco de moderado a alto de alteração.
- Quando um o projeto está no estágio de teste, é difícil voltar e alterar uma funcionalidade.
- Nenhum parte do software é produzido até a fase de codificação.

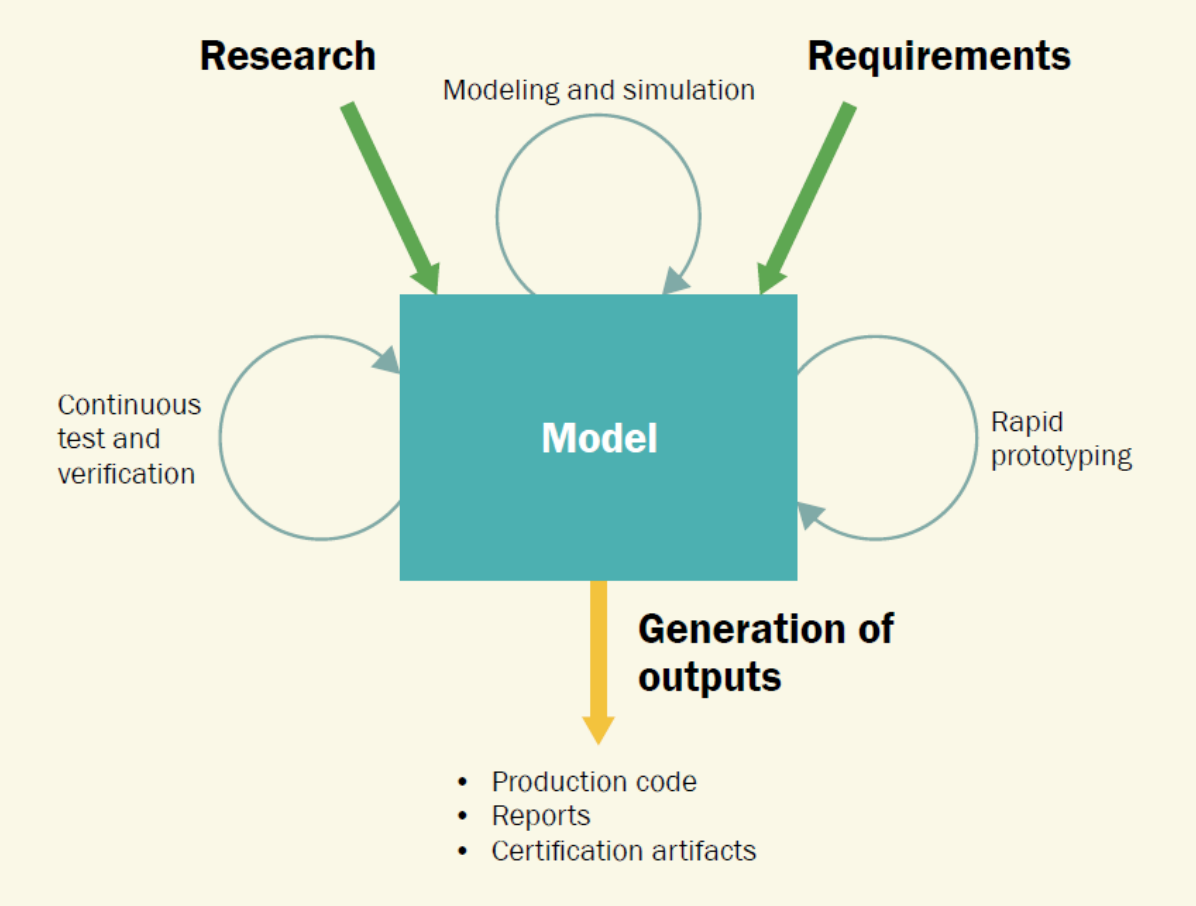
4.2 Model-Based Design (MBD)

O Model-Based Design (MBD) é uma abordagem centrada no modelo para o desenvolvimento de sistemas de controle, processamento de sinais, comunicações e outros sistemas dinâmicos. Em vez de depender de protótipos físicos e especificações textuais, o MBD usa modelos ao longo desenvolvimento. O modelo inclui todos os componentes relevantes para o comportamento do sistema, algoritmos, lógica de controle, componentes físicos e propriedade intelectual (IP). Uma vez que o modelo é desenvolvido (elaborado), ele se torna a origem de muitas saídas, incluindo relatórios, código C e código HDL. O MBD permite o design no nível do sistema e no nível do componente, simulação, geração automática de código, teste contínuo e verificação.

O MBD pode suportar praticamente qualquer tipo de organização e foi implementado com sucesso em muitos fluxos de desenvolvimentos diferentes. Como você implementa depende do tamanho, estrutura e cultura da sua organização, bem como os sistemas em desenvolvimento e as demandas de seu mercado alvo. Você pode optar por adotar Model-Based Design em toda a empresa, transformando todo o seu desenvolvimento processo. Alternativamente, você pode aplicá-lo seletivamente

para abordar um desafio específico, como um gargalo no fluxo de trabalho, uma mudança súbita requisitos de design ou maior complexidade do sistema.

Figura 20- Model-Based Design workflow.



Fonte: MathWorks.

4.3 Implementação do V-Model e MBD no projeto

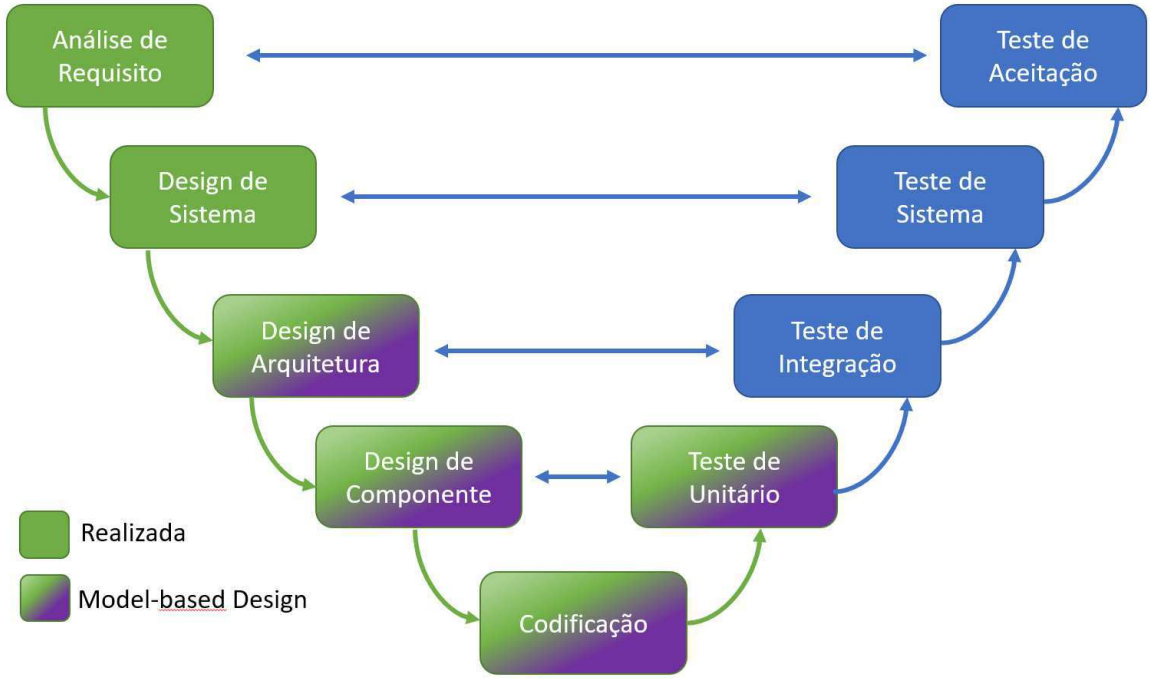
Como mencionado anteriormente, as atividades realizadas nesse projeto de estágio foram desenvolvidas utilizando-se a metodologia V-Model e Model-Based Design. Nas secções seguintes iremos percorrer as etapas do V-Model de forma a destacar o que foi realizado em cada etapa.

O projeto teve um escopo definido de tal forma que as etapas de análise de requisitos, design de sistema, design da arquitetura, design de componente, implementação ou codificação e teste de componente foram cobertas, ficando as

etapas de teste de integração, teste de sistemas e teste de aceitação para trabalhos futuros.

Na Figura 21 temos em destaques as etapas do V-Model que foram cobertas nesse estágio, bem como em quais dessas etapas foi utilizado o Model-Based Design.

Figura 21 - V-Model. A etapas nas cores verdes ou verdes e roxas, foram implementadas durante o estágio.



MathWorks

Fonte: Próprio autor.

4.3.1 Análise de requisitos

Como sabemos, em todo projeto a primeira etapa que deve ser realizada é o levantamento e análise dos requisitos coletados com o cliente. Nesse projeto o cliente é o próprio LIEC, local onde o estágio foi realizado, representado pela pessoa do professor Rafael Bezerra Correia Lima. Todos os requisitos foram recolhidos e analisados juntamente com o professor Rafael.

Ao fim dessa etapa os seguintes requisitos foram levantados:

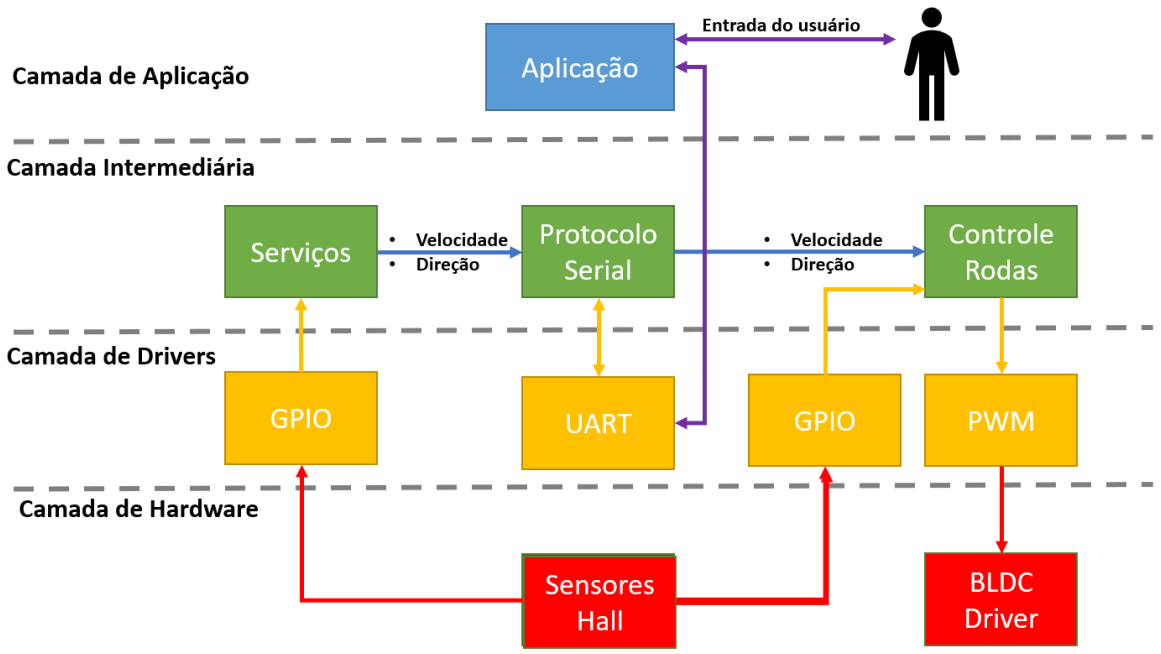
- Desenvolvimento de uma camada intermediária de firmware para o hoverboard que faça a interface de comunicação entre uma camada de aplicação e a camada de drivers de hardware.

- A camada intermediária deve conter a lógica de controle para ambas as rodas do hoverboard.
- A camada intermediária deve prover para a camada de aplicação a velocidade e direção de ambas as rodas quando requisitado.
- A camada de aplicação deve ter acesso para modificar a velocidade e direção de ambas as rodas individualmente.
- Deve existir um protocolo de comunicação serial para todas a mensagens trocadas entre a camada de aplicação e a camada intermediária.

4.3.2 Design de sistema

De posse dos requisitos levantados na etapa de análise de requisitos, foi possível elaborar um design para todo o sistema. Na Figura 22 temos uma representação em diagrama blocos de todo o sistema.

Figura 22 - Arquitetura do sistema.



Fonte: Próprio autor.

Na Figura 22 podemos observar as três camadas de software, e uma camada de hardware.

Na camada de aplicação as aplicações de alto níveis serão desenvolvidas, por exemplo, processamento de imagem, controle de trajetória, comunicação com servidor web. A ideia é que a camada de aplicação seja implementada em uma outra plataforma (Arduino, Rapsberry PI, ...) e que se comunique com a camada intermediária através de UART, utilizando um protocolo que será apresentado mais à frente.

A camada intermediária como já mencionado, faz a conexão da camada de aplicação com a camada de drivers. Nela são implementados o controle de ambas as rodas do hoverboard, os serviços que fornecem a direção e velocidade de cada roda e a lógica de codificação e decodificação do protocolo utilizado para se comunicar com a camada de aplicação.

Na camada de drivers é realizada as configurações dos periféricos do microcontrolador STM32F103RCT6.

A camada de hardware foi usada para representar os componentes físicos, sensores e esquemas elétricos que compõe o sistema. É importante observa que na Figura 22 não foi colocado todos os componentes de hardware do sistema, mas somente os que são representativos para camada intermediária.

4.3.3 Design de arquitetura

Uma vez que o design do sistema foi realizado e aprovado na etapa anterior, mais detalhes do sistema são abordados na etapa de design de arquitetura. Como o objetivo desse estágio é o desenvolvimento da camada intermediária, aqui mostraremos como foi feito design de cada subsistema dessa camada. Veja Figura 23.

O subsistema Serviços tem a tarefa de determinar a velocidade e direção de cada roda e enviar esses dados para o subsistema responsável pelo protocolo de comunicação, o Protocolo Serial.

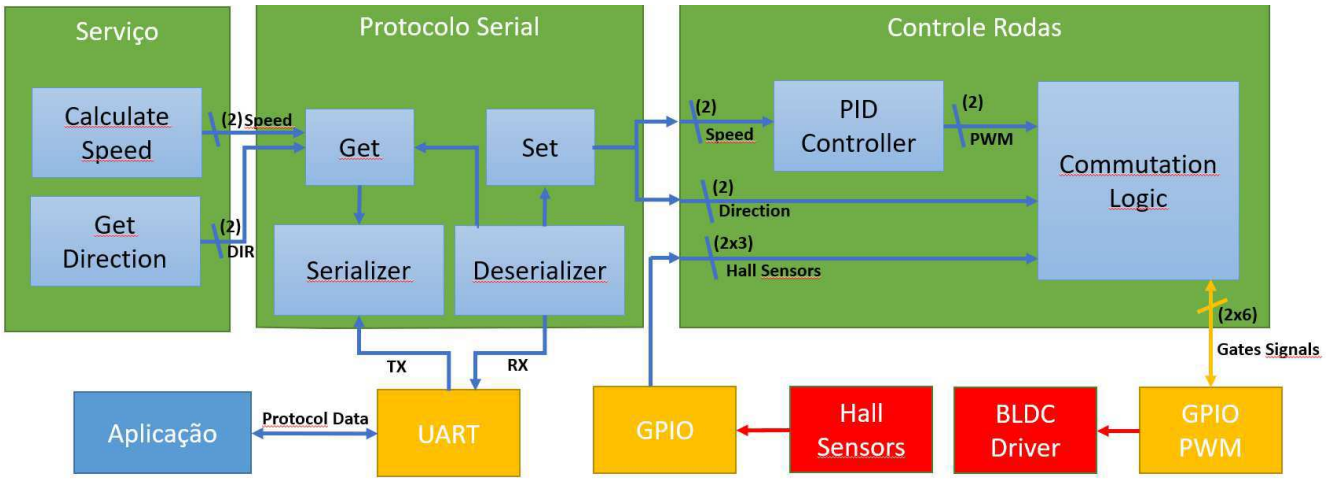
O subsistema Protocolo Serial recebe requisições da camada de aplicação, trata esses dados e faz a comunicação própria com os subsistemas Serviços e Controle de Rodas.

O subsistema Controle de Rodas foi dividido em dois blocos principais, o *PID Controler* que tem a funcionalidade de controlar a velocidade de cada roda e o *Commutation Locig* que possui toda a lógica de comutação dos drivers do BLDC e controle de direção das rodas.

4.3.4 Design de componente

Nessa fase, o design interno detalhado de todos os módulos do sistema é especificado, conhecido como design de baixo nível. A título de exemplificar como o V-Model é usado para o desenvolvimento do projeto, nessa e próximas etapas do V-Model abordadas nesse estágio, utilizaremos o bloco *Commutation Logic* do subsistema Controle de Rodas como exemplo.

Figura 23 - Camada Intermediária.



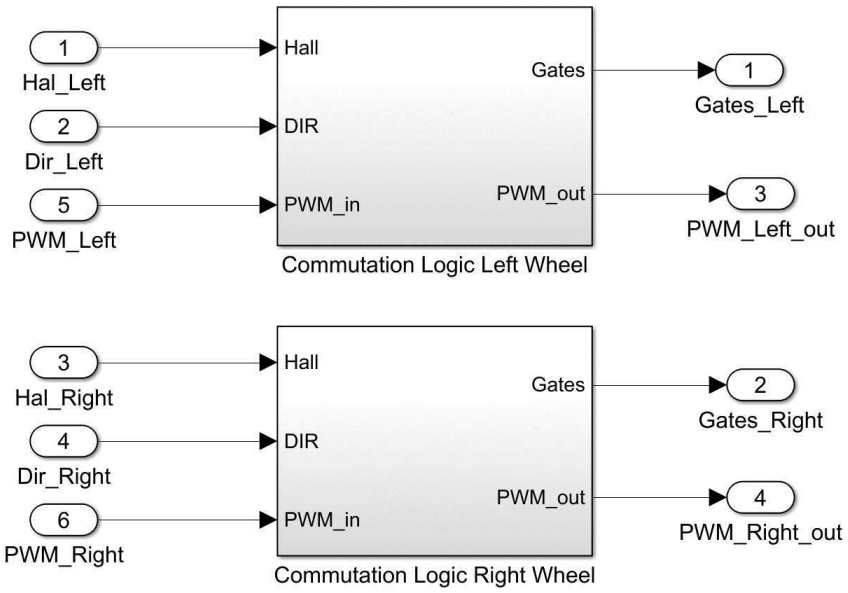
Fonte: Próprio autor.

O bloco *Comuntation Logic* tem a funcionalidade de controlar a lógica de comutação e direção das duas rodas do hoverboard. Esse bloco recebe como entradas dois valores de PWM provindos do bloco PID Controller. Esses sinais PWM são determinados de acordo com os ganhos do controlador, recebe também dois sinais indicando a direção de cada roda e os estados dos sensores Hall de cada motor.

Como pode ser visto na Figura 24, o subsistema *Commutation Logic* foi dividido em outros dois subsistemas, *Commutation Logic Left Wheel* e *Commutatino Logic Right Wheel*, responsáveis por realizar a comutação e controlar a direção das rodas esquerda e direita respectivamente. Essa divisão foi feita com a finalidade de modularizar o desenvolvimento do software. A lógica desses dois subsistemas são iguais, já que se trata do controle de comutação e direção das rodas, logo a partir desse ponto iremos detalhar apenas o subsistema *Commutation Logic Left Wheel*.

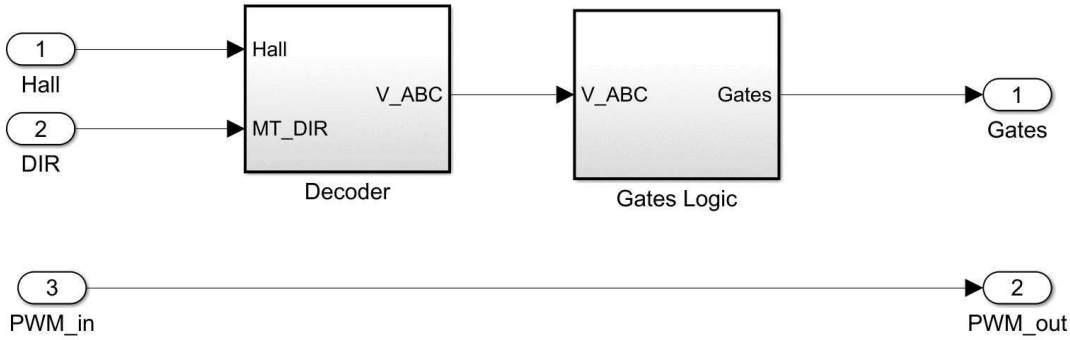
O bloco *Commutation Logic Left Wheel*, ver Figura 25, para manter a modularidade do software, foi dividido em outros dois subsistemas, o *Decoder* e o *Gates Logic*. A entrada do bloco *Decoder* recebe os valores dos estados dos sensores Hall associados ao motor/roda e da direção que se deseja seguir. O sinal de saída do bloco *Decoder* entra no bloco *Gates Logic* ,que define quais transistores do driver que controla o motor BLDC devem ser ativados ou desativados. Nenhuma lógica é feita com o sinal PWM, o valor que entra no subsistema *Commutation Logic Left Wheel* é o mesmo valor que sai do subsistema, o motivo do sinal PWM ser colocado aqui, é que se pode ter a necessidade de aplicar alguma lógica ao mesmo no futuro.

Figura 24 - *Commutation Logic*.



Fonte: Próprio autor.

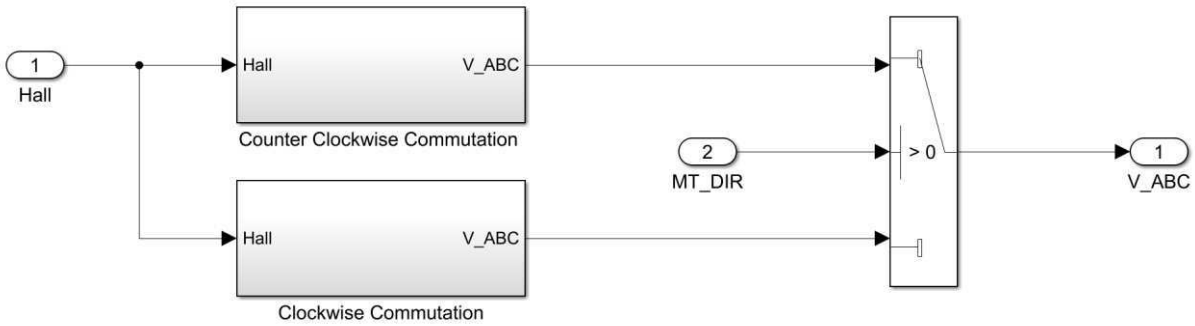
Figura 25 - Commutation Logic Left Wheel



Fonte: Próprio autor.

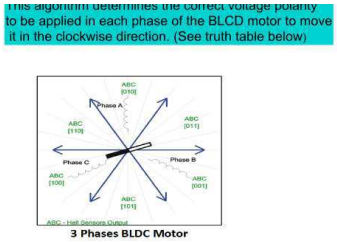
Dentro do subsistema *Decoder* temos outros dois subsistemas, o *Counter Clockwise Commutation* e o *Clockwise Commutation*, ver Figura 26. Esses dois subsistemas têm a funcionalidade de determinar qual a polaridade correta que deve ser aplicada em cada uma das fases do motor BLDC. O *Counter Clockwise Commutation* determina os passos de comutação do sentido anti-horário, seguindo a tabela da Figura 27. O *Clockwise Commutation* determina os passos de comutação do sentido horário, seguindo a tabela da Figura 28. De acordo com as tabelas, podemos observar que para cada conjunto de valores do sensores Hall, existe um conjunto de valores de saída para cada uma das fases do motor, VA, VB, e VC. Os valores que VA, VB e VC pode assumir são 1(um), para indicar que a fase deve ter polaridade positiva, 0(zero), para indicar que a fase deve ficar em estado flutuante ou aberta e -1(menos um), para indicar que fase deve ter polaridade negativa.

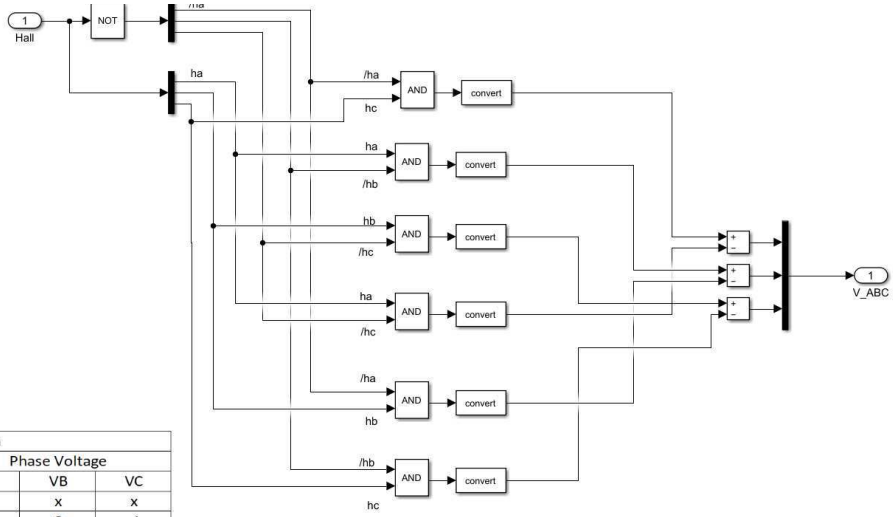
Figura 26- Subsistema *Decoder*.



Fonte: Próprio autor.

Figura 28 - Subsistema *Counter Clockwise Commutation.*

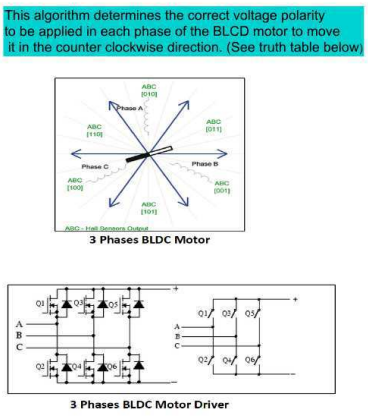


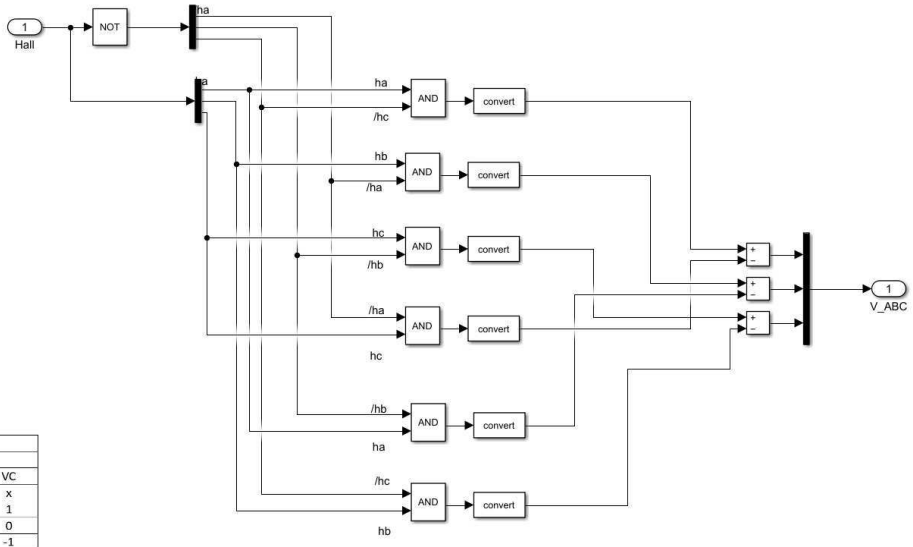


BLDC 3 Phases Clockwise Commutation							
Step	Hall Sensors				Phase Voltage		
	HA	HB	HC	Binary	VA	VB	VC
x	0	0	0	0	x	x	x
1	0	0	1	1	1	0	-1
2	1	0	1	5	0	1	-1
3	1	0	0	4	-1	1	0
4	1	1	0	6	-1	0	1
5	0	1	0	2	0	-1	1
6	0	1	1	3	1	-1	0
x	1	1	1	7	x	x	x

Fonte: Próprio autor.

Figura 27 - Subsistem *Clockwise Commutation*.





BLDC 3 Phases Counter Clockwise Commutation							
Step	Hall Sensors				Phase Voltage		
	HA	HB	HC	Binary	VA	VB	VC
x	0	0	0	0	x	x	x
1	0	0	1	1	-1	0	1
2	0	1	1	3	-1	1	0
3	0	1	0	2	0	1	-1
4	1	1	0	6	1	0	-1
5	1	0	0	4	1	-1	0
6	1	0	1	5	0	-1	1
x	1	1	1	7	x	x	x

Fonte: Próprio autor.

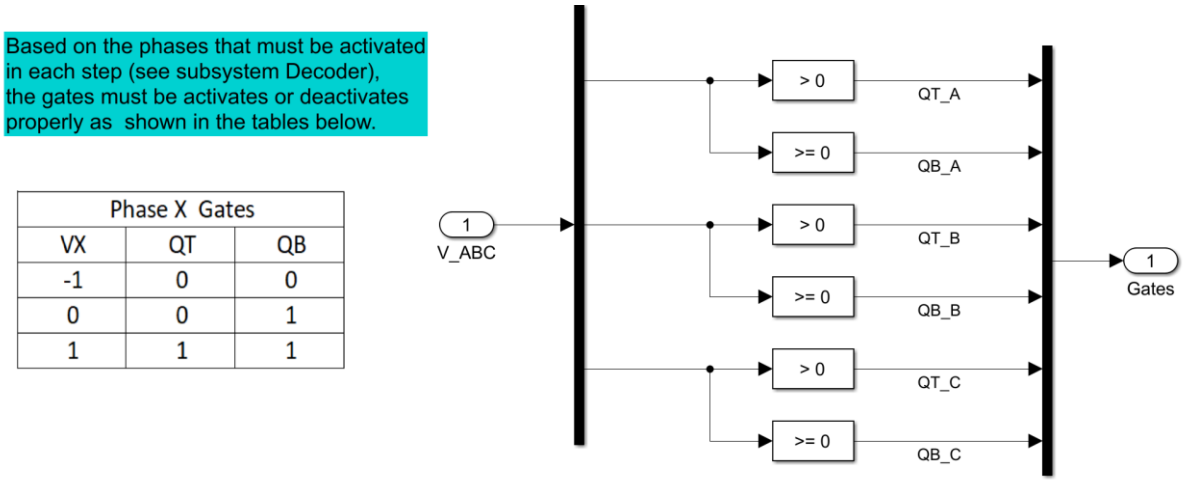
Ainda dentro do subsistema *Decoder* temos o bloco que determina qual deve ser a sequência de comutação que será usada baseado no valor da entrada MT_DIR. Caso MT_DIR tenha valor 0 (zero) o motor gira no sentido horário, caso MT_DIR tenha valor 1 (um) o motor gira no sentido anti-horário.

O subsistema *Gates Logic* tem a funcionalidade de determinar quais transistores de circuito de comutação de cada fase, devem ser acionados ou não para que as fases sejam polarizadas da forma correta. Na Figura 29 temos a lógica implementada para esse subsistema no Simulink.

4.3.5 Teste de componente

Uma vez que foi realizada a lógica de cada componente na etapa anterior, antes da geração do código, foi realizado os testes de componentes. A etapa de teste de componente, seguindo o V-Model, deveria ser realizada após a etapa de codificação, mas como estamos utilizando Mode-based Design, podemos adiantar a etapa de testes de componentes, por meio de simulação, evitando assim gerar código com erros.

Figura 29 - Subsistema *Gates Logic*.

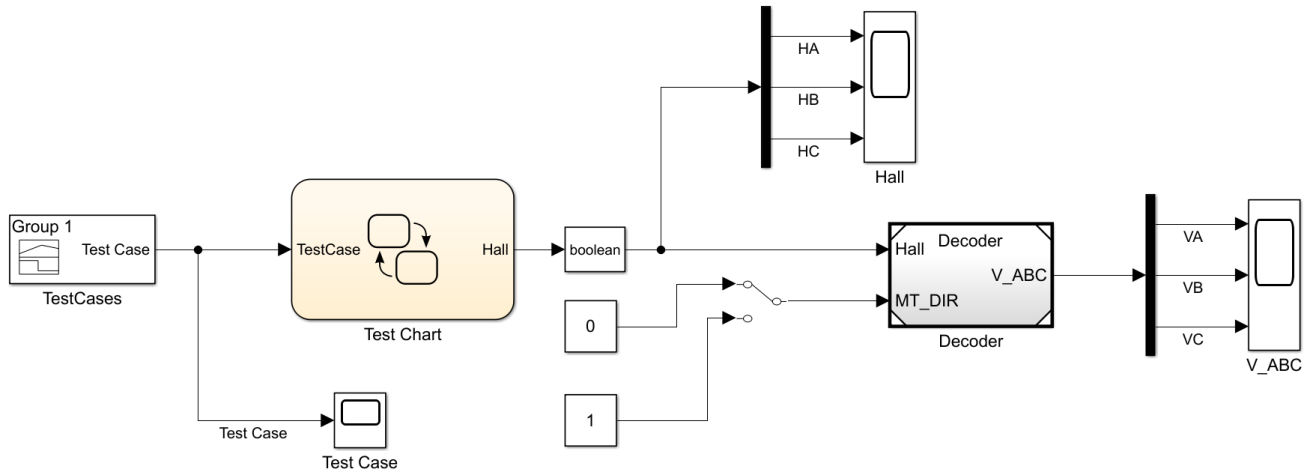


Fonte: Próprio autor.

Na Figura 30 temos a estrutura que foi montada no Simulink para validar a lógica implementada no subsistema *Decoder*. A lógica testada foi a das tabelas das

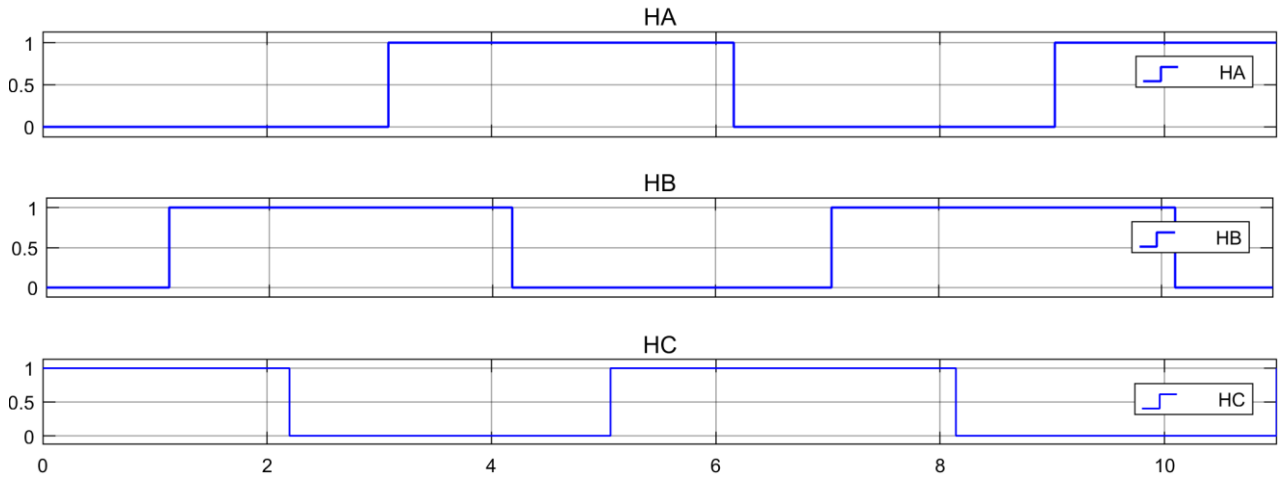
figuras Figura 27 e Figura 28. Na Figura 32 temos os sinais dos sensores Hall usados no teste e na Figura 31 a saída do subsistema *Decoder* para os testes realizados.

Figura 30 - Sistema montado para realizar teste de componente do subsistema *Decoder*.



Fonte: Próprio autor.

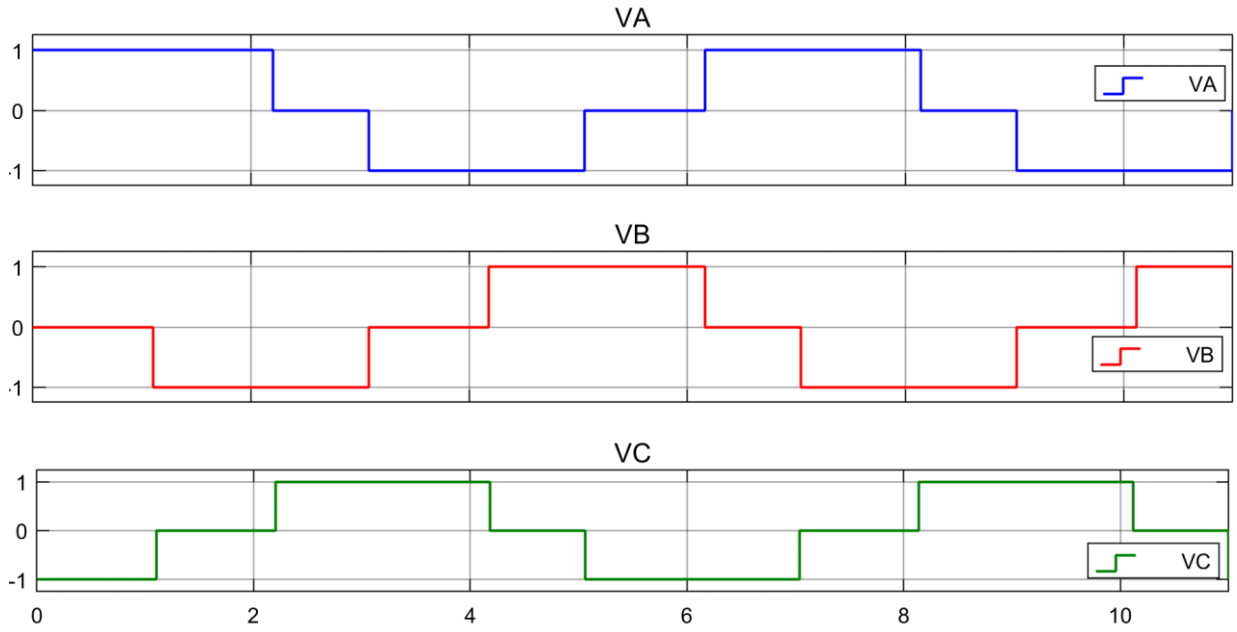
Figura 31 - Sinais dos sensores Hall utilizados no teste do subsistema *Decoder*.



Fonte: Próprio autor.

4.3.6 Implementação ou codificação

Figura 32 - Sinais de saída do subsistema *Decoder*.

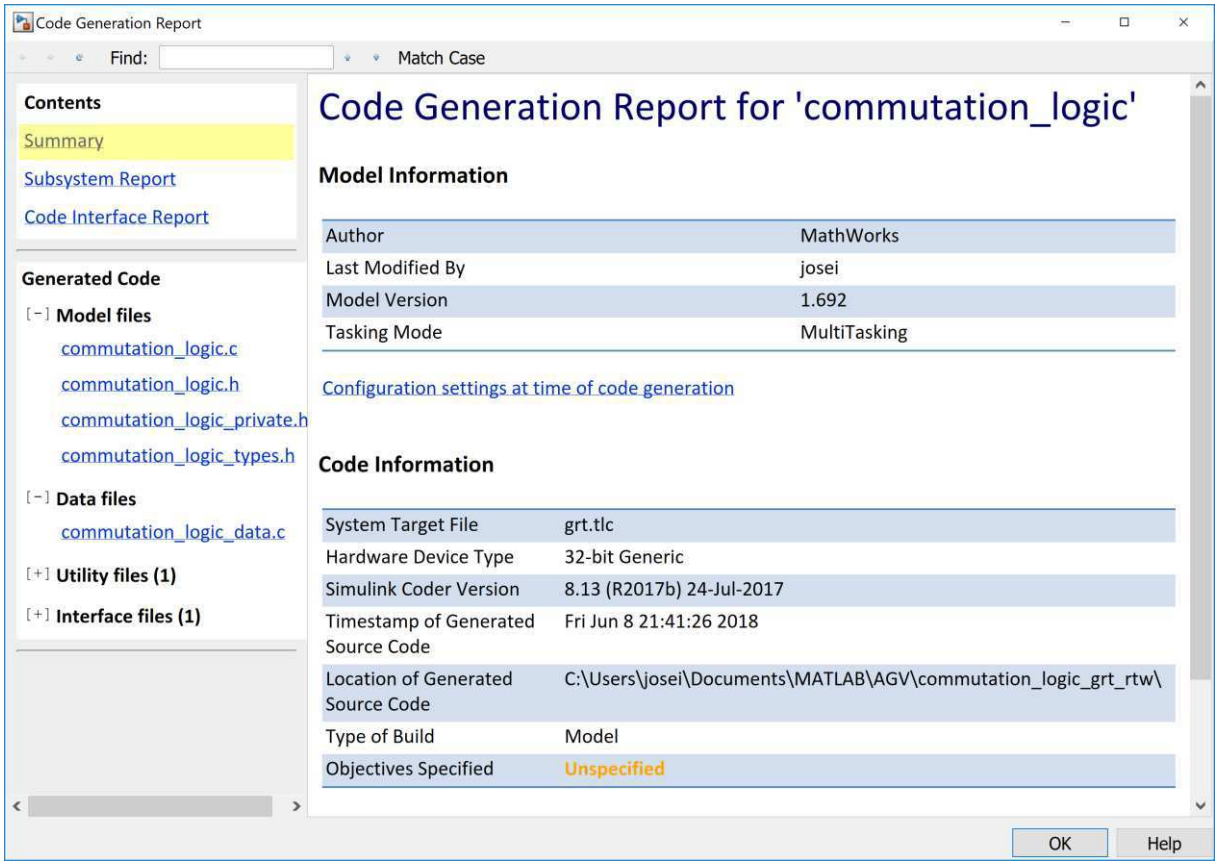


Fonte: Próprio autor.

Para realizar a codificação da lógica desenvolvida, foi utilizado a ferramenta de geração automática de código do Simulink. Essa ferramenta permite além da geração automática de código e geração de relatórios e documentação de como integrar o código gerado automaticamente com códigos de terceiros ou código legado.

Na Figura 33, temos o relatório criado pelo Simulink após a geração automática do código, bem como parte do código gerado para os subsistemas *Commutation Logic Right Wheel* e *Commutation Logic Left Wheel*. A lista de todo o código gerado bem como os relatórios de todos os subsistemas será entregue em um arquivo zipado.

Figura 33 - Relatório gerado automaticamente pelo Simulink.



Fonte: Próprio autor.

4.3.7 Protocolo de comunicação

Para gerenciar a comunicação entre a camada intermediária e a camada de aplicação, um protocolo simples de troca de mensagens foi desenvolvido.

Para realizar o acesso aos dados, foram criadas duas funções básicas:

- Função *GET*: usada quando a camada de aplicação requisita a leitura da velocidade e ou direção das rodas.
- Função *SET*: usada quando a camada de aplicação deseja modificar o valor da velocidade e ou direção das rodas.

As mensagens que são trocadas entre a camada de aplicação e intermediária possuem o formato de uma estrutura JSON modificada. Os seguintes valores podem ser tomados pelos campos da estrutura:

- FCode, representa o tipo de função.

    - FCode = 0, função GET
    - FCode = 1, função SET
- WCode, representa a roda que a função definida de acordo com o valor de FCode vai atuar.
    - WCode = 0, roda esquerda.
    - WCode = 1, roda direita.
    - WCode = 2, rodas direita e esquerda.
- ACode, representa os atributos da roda que está sendo requisitado.
    - ACode = 0, velocidade da roda.
    - ACode = 1 , direção da roda.
    - ACode = 2, velocidade e direção da roda.
- SCode, representa a velocidade da roda em cm/s.
- DCode, representa o sentido de rotação da roda.
    - DCode = 0 , sentido horário.
    - DCode = 1, sentido anti-horário.

4.3.8 Exemplos de utilização do protocolo

- Exemplo 1: Requisitar a velocidade da roda direita.

Função GET	Resposta

{
F: 0,
W: 1,
A: 0
}
	{
	W:{
	I: 1 //Roda direita
	S: 10 //Velocidade da roda direita
	}
	}

- Exemplo 2: Requisitar a direção da roda esquerda

Função GET	Resposta

{
F: 0,
W: 0,
A: 1
}
	{
	W:{
	I: 0 //Roda esquerda
	D: 0 //Direção da roda esquerda
	}
	}

- Exemplo 2: Requisitar a velocidade e a direção das rodas direita e esquerda.

Função GET	Resposta

{
F: 0,
W: 2,
A: 2
}
	{
	W:{
	I: 0 , //Roda esquerda
	S:10, //Velocidade da roda esquerda
	D: 0 //Direção da roda esquerda
	},
	W:{
	I: 1, //Roda direita
	S: 12, //Velocidade da roda direita
	D: 1 //Direção da roda direta
	}
	}

A função SET deve ser usada quando desejamos mudar a velocidade ou direção das rodas.

- Exemplo 3: Mudar velocidade da roda esquerda.

Função SET	Resposta

{
F: 1, // Função SET
W: 0, // Roda Esquerda
S: 0020 //Velocidade
D: 1 //Direção de rotação
}
	Sem Resposta

- Exemplo 2: Mudar velocidade e direção da roda direita.

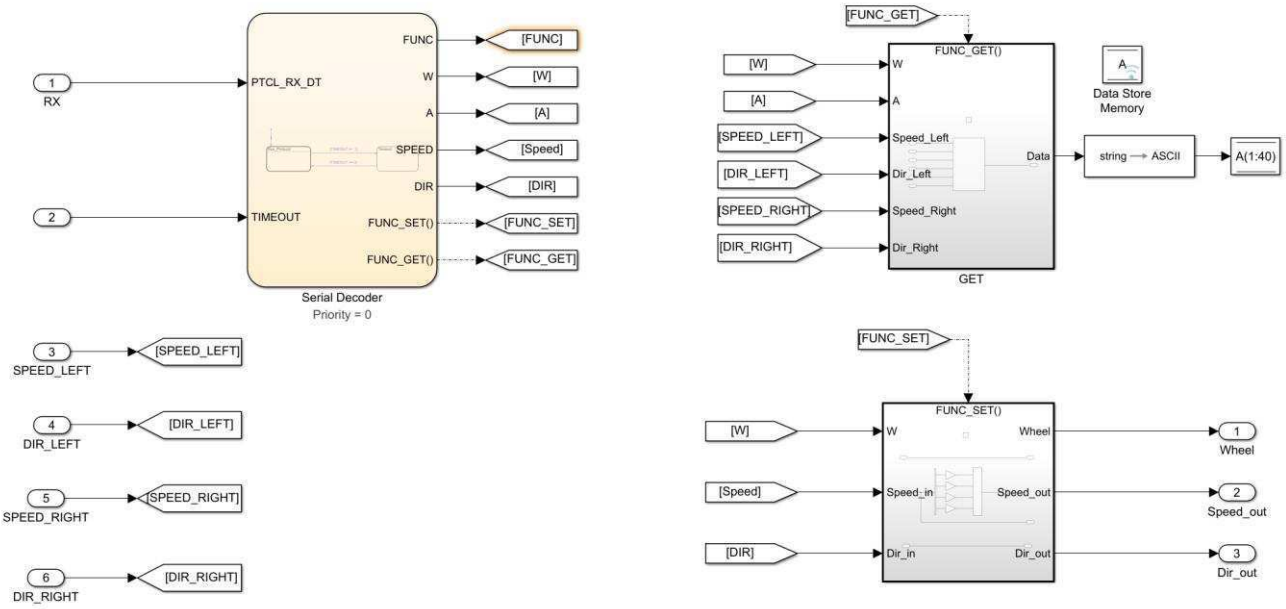
Função SET	Resposta

{
F: 1, // Função SET
W: 1, // Roda Direita
S: 0020 //Velocidade
D: 1 //Direção de rotação
}
	Sem Resposta

Caso deseje-se mudar a direção e ou sentido de ambas as rodas, deve-se enviar dois comando SET separadamente, um para cada roda.

Na Figura 34, temos a ilustração da implementação do protocolo de comunicação no Simulink.

Figura 34- Protocolo Serial implementado no Simulink para geração automática de código.

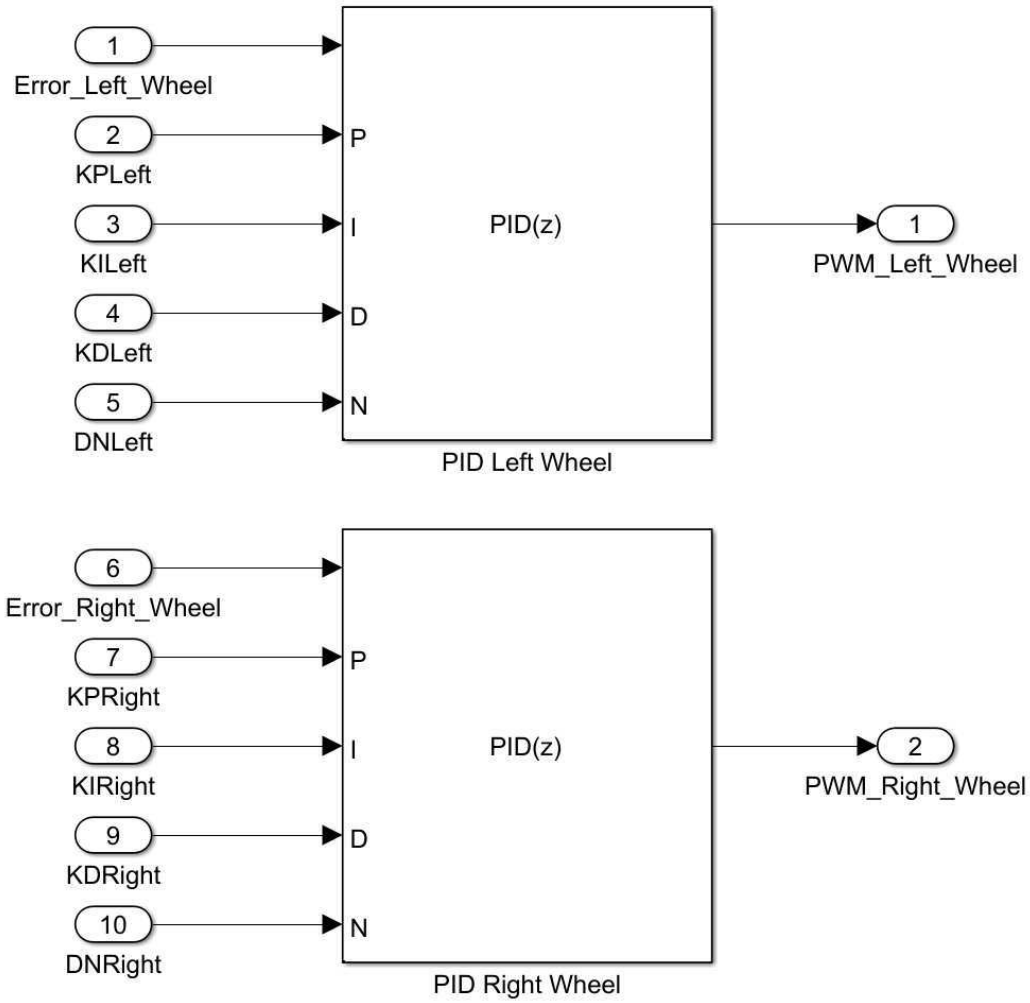


Fonte: Próprio autor.

4.4 Controlador PID

Como pode ser visto na Figura 23, dentro do subsistema Controle Rodas, temos um outro subsistema chamado *PID Control*. Nesse subsistema temos dois controladores PID, para realizar o controle da velocidade das duas rodas do hoverboard. Na Figura 35, temos a implementação desses dois controladores realizada no Simulink. É importante destacar que a sintonia desses controladores não está no escopo desse estágio, porém uma vez que a sintonia seja feita, as ganhos de cada controlador pode ser carregados no controlador por meio dos parâmetros KPLeft, KILeft, KDLeft e DNLeft para roda esquerda e KPRight, KIRight, KDRight e DNRight para roda direita.

Figura 35 - Implementação dos controladores PID para controle de velocidade de ambas as rodas do hoverboard.



Fonte: Próprio autor.

5 Conclusão

No decorrer desse estágio, onde foi desenvolvido uma camada intermediária de software para comunicação entre uma camada de aplicação e uma camada de drivers do hoverboard, foi possível associar o conhecimento adquirido em diversas disciplina acadêmicas e extra acadêmico.

Foi possível colocar em prática muito do que foi aprendido durante os últimos anos no curso de Engenharia Elétrica, além de poder explorar outros ferramentais como o consagrado V-Model e metodologias emergentes no âmbito de desenvolvimento de projeto, como é o caso do Model-Based Design.

É certo dizer que as atividades desenvolvidas nesse estágio foi de suma importância para formação profissional do estagiário, uma vez que apresentou um conjunto de desafios, permitindo assim a aplicação dos mais diversos conhecimentos adquiridos.

Outra importante contribuição desse trabalho é que ele pode servir de base para outros colegas usarem as metodologias aplicadas durante o desenvolvimento da solução proposta nesse estágio em seus futuros projetos.

6 Bibliografia

Aarenstrup, R., 2015. *Managing Model-Based Design.* s.l.:The MathWorks, Inc..

Editor, T., 2018. *Tutorials Point.* [Online] Available at: https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm [Acesso em 23 Maio 2018].

LIEC, 2018. *Laboratório de Instrumentação Eletrônica e Controle.* [Online] Available at: http://liec.ufcg.edu.br/ [Acesso em 30 maio 2018].

Li, X., 2015. *MODEL-BASED DESIGN OF BRUSHLESS DC MOTOR CONTROL AND MOTION CONTROL MODELLING FOR ROBOCUP SSL ROBOTS,* Vaasa, Finland: VAASAN AMMATTIKORKEAKOULU UNIVERSITY OF APPLIED SCIENCES.

Mubeen, M., July,2008. Brushless DC Motor Primer. *Motion Tech Trends.*

Zhao, J. & Yangwei Yu, 2011. Brushless DC Motor Fundamentals.

APÊNDICE