INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA

MARCIO ANTONIO AUGUSTO

SISTEMA DE AQUISIÇÃO DE DADOS E CONTROLE EM MALHA FECHADA DE FREIO A CORRENTES PARASITAS DE DINAMÔMETRO DE BANCADA

FLORIANÓPOLIS, 2019.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE ENGENHARIA ELETRÔNICA

MARCIO ANTONIO AUGUSTO

SISTEMA DE AQUISIÇÃO DE DADOS E CONTROLE EM MALHA FECHADA DE FREIO A CORRENTES PARASITAS DE DINAMÔMETRO DE BANCADA

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Engenheiro de Eletrônica

Orientador:
Flábio Alberto Bardemaker Batista,Dr. Eng.
Orientador

FLORIANÓPOLIS, 2019.

Augusto, Marcio Antonio
Sistema de aquisição de dados e controle em malha fechada de freio a correntes parasitas de dinamômetro de bancada / Marcio Antonio Augusto; Orientador:Flábio Alberto Bardemaker Batista,Dr. Eng. Orientador – Florianópolis, SC, 2019.
120 p. : il color.

Trabalho de Conclusão de Curso (Eng. Eletrônica) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.
Inclui referências.

1. Dinamômetro a correntes parasitas. 2. Teste em motores à combustão. 3. Controle digital. 4. Velocidade de rotação. 5. Torque. I. Flabio, A. B. Batista II. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. III. Sistema de aquisição de dados e controle em malha fechada de freio a correntes parasitas de dinamômetro de bancada.

SISTEMA DE AQUISIÇÃO DE DADOS E CONTROLE EM MALHA FECHADA DE FREIO A CORRENTES PARASITAS DE DINAMÔMETRO DE BANCADA

MARCIO ANTONIO AUGUSTO

Este trabalho foi julgado adequado para obtenção do Título de Engenheiro de Eletrônica e aprovado na sua forma final pela banca examinadora do Curso Superior de Engenharia Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, de de 2019.

Banca Examinadora:

_____

Flábio Alberto Bardemaker Batista,Dr. Eng.
Orientador

_____

Marcelo Vandresen, Dr. Eng. Coorientador

_____

Mauro Tavares Peraça, Dr. Eng.

_____

Júlio Feller Golin, Me. Eng.

Dedicado a todos que
de alguma forma correm
em busca de seus sonhos.

AGRADECIMENTOS

Agradeço a minha esposa Helena por todo o apoio durante meus momento de cansaço e toda a paciência durante meus momentos estresse. Durante essa longa jornada, muitos foram os momentos que a sua ajuda foi fundamental para eu me manter firme nos meus objetivos e sonhos.

Agradeço a minha mãe Alci e meu pai João que sempre me mostraram a importância de certos valores para a vida, valores esses que me serviram de base para minhas escolhas do dia-a-dia.

Agradeço ao meu filho Bernardo que me serviu de inspiração para continuar e que muitas vezes me passou a energia que eu precisei.

Agradeço a meu orientador Flabio Alberto Bardemaker Batista que aceitou entrar nesse desafio e foi como uma luz indicando o caminho a seguir nos momentos em que eu me sentia perdido.

Agradeço ao meu coorientador Marcelo Vandresen que me apresentou esse projeto desafiador e fora da minha zona de conforto, que agregou muito conhecimento à minha formação profissional.

Agradeço aos meus amigos do IFSC que durante todo o curso auxiliaram no meu processo de absorção do conhecimento adquiridos dentro e fora das aulas.

Obrigado.

“Escolha um trabalho que você ama
e você nunca terá que trabalhar
um dia sequer na vida“
Confúcio

RESUMO

Este trabalho tem como objetivo o desenvolvimento de um sistema para aquisição de dados e controle digital de um dinamômetro a correntes parasitas do laboratório do grupo de pesquisa ADEMCI, localizado no IFSC - Câmpus Florianópolis, que faz parte de uma bancada para testes em motores à combustão. Este sistema faz a aquisição dos dados de velocidade de rotação e torque do dinamômetro e implementa o controle, em malha fechada e de forma digital, visando manter constante, ou até com uma variação constante, sua velocidade de rotação. Para facilitar a análise dos testes, o sistema conta ainda com comunicação com um computador que, utilizando uma interface gráfica, permite o acompanhamento dos dados em tempo real e armazenamento dados dos testes para posterior análise. Neste trabalho apresenta-se uma revisão bibliográfica que visa rever alguns conceitos e equipamentos que fazem parte do sistema a ser controlado, é demonstrado também como foi feita a modelagem matemática do sistema e como foi projetada, e implementada, a leitura dos dados necessários. O projeto de controle desenvolvido é apresentado junto às soluções levantadas para problemas que surgiram durante seu processo de construção. O protótipo construído, tanto em *hardware* quanto em *software*, trouxe resultados próximos dos requisitos iniciais do projeto e demonstra que o método utilizado durante esta pesquisa é uma forma de solucionar o problema levantado. Por fim, são feitas algumas sugestões de melhorias para o projeto, com o intuito de refinar os resultados e a experiência de seu usuário.

Palavras-chave: Dinamômetro a correntes parasitas. Teste em motores a combustão. Controle digital. Velocidade de rotação. Torque.

ABSTRACT

This work aims to develop a system for data acquisition and digital control of a eddy current dynamometer from the laboratory of the research group ADEMCI, located at IFSC - Campus Florianópolis, which is part of a test bench which is used for testing on combustion engines. This system acquires the dynamometer's rotational speed and torque data and performs the control, in closed loop and digitally, keeping its rotational speed constant, or even a constant variation. To facilitate test analysis, the system also has communication with a computer that, using a graphical interface, allows the monitoring of data in real time and storage of test data for later analysis. This work presents a bibliographic revision in some concepts and equipments that are part of the system to be controlled, it also demonstrates how the system mathematics model was made and how the used data was read and implemented. The developed control project is presented together with the solutions raised for problems that arose during the construction process. The built prototype, in both hardware and software, presents results that are the next requirements for the project and demonstrates the method used during this research is a way to solve the problem raised. Finally, some suggestions for improvements to the project are made in order to refine the results and the user experience.

Keywords: Eddy current dynamometer. Combustion engines tests. Digital control. Rotational speed. Torque.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O Câmpus Florianópolis do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, IFSC, possui um laboratório que é utilizado para ensaios de motores utilizados em veículos automotores, motores à combustão interna. Este laboratório faz parte do grupo de pesquisa chamado ADEMCI, Avaliação do Desempenho de Motores de Combustão Interna, e pode ser utilizado para avaliar o desempenho de diferentes lubrificantes, combustíveis ou motores. Nos testes neste laboratório são levantas características de uso do motor, como eficiência, potência ou emissão de poluentes. Os ensaios são feitos acoplando o freio dinamômetro ao motor e verificando seu desempenho em certas condições de uso que são impostas pelo freio dinamômetro. Na maioria dos casos as condições de testes requerem valores constantes para a velocidade de rotação ou torque do motor, ou até mesmo uma variação constante de um dos dois parâmetros. O dinamômetro é acionado utilizando uma tensão contínua e a frenagem aplicada pelo freio é proporcional à essa tensão.

Nesta pesquisa foi desenvolvido um sistema de controle e aquisição dos dados para ensaios de potência líquida efetiva em motores à combustão interna, utilizados em veículos automotores. Esse tipo de teste tem por objetivo avaliar a potência do motor à plena carga em função da velocidade de rotação. Este sistema, objeto desta pesquisa, visa diminuir o tempo gasto nesses ensaios e permitir uma comparação mais rápida entre diferentes ensaios feitos. Para o projeto de controle deste sistema, o conjunto formado pelo motor e dinamômetro, foram utilizadas técnicas de controle clássicas que visam manter os parâmetros do motor seguindo os valores necessários para a validação dos testes.

Este trabalho está dividido em 6 capítulos. O primeiro é a introdução ao assunto desta pesquisa. No segundo são apresentados alguns conceitos teóricos que influenciam nosso sistema. No terceiro é apresentado a metodologia que foi empregada durante o projeto. No Capítulo 4 é demonstrado como foi desenvolvido o projeto, como foram implementadas as leituras dos dados, o acionamento do freio, a modelagem do sistema e o projeto e implementação do sistema de controle. No quinto capítulo são apresentados os resultados alcançados pelo sistema, objeto desta pesquisa. No capítulo 6 é feita uma análise dos custos materiais envolvidos na implementação do sistema desenvolvido. No capítulo 7 são apresentadas as considerações finais deste trabalho e sugestões para trabalhos futuros.

1.1 Justificativa

O motor à combustão interna é uma peça fundamental para nossa sociedade atual, pelo seu amplo uso, nos meios de transporte, principalmente. Esse tipo de motor vem evoluindo constantemente, resultando em motores cada vez mais eficientes, com maior potência e que poluem menos o meio ambiente. Para que essa evolução aconteça são necessários muitos testes utilizando-se curvas padrão de rotação e torque do motor. Essas curvas são valores constantes de rotação até mesmo uma variação constante e para se conseguir seguir essas curvas padrão, o motor é acoplado a um freio dinamômetro que necessita, ainda, de um sistema de controle que garanta que essas curvas padrão sejam seguidas, validando assim os testes. Um sistema de aquisição dos dados também é fundamental, tendo em vista que uma aquisição manual dos dados é muito lenta e pode inviabilizar os testes. O IFSC - Câmpus Florianópolis possui um laboratório para testes em motores à combustão interna, onde se encontra um dinamômetro a correntes parasitas, que pode realizar testes de potência, eficiência e emissão de poluentes. Este laboratório necessita de um sistema de controle e aquisição de dados para melhorar o seu funcionamento e, por esse motivo, esta pesquisa, que visa à criação de um sistema desse tipo, ganha relevância.

1.2 Definição do problema

Os testes com motores à combustão interna feitos com dinamômetros a correntes parasitas precisam ser feitos com características específicas de velocidade de rotação e torque, além de possuir um sistema de aquisição dos dados gerados. No mercado, há algumas alternativas de equipamentos que fazem o controle dessas características de teste e, ainda, fazem a aquisição dos dados necessários para posterior análise, mas essas soluções possuem preços elevados e isso dificulta, muitas vezes sua aquisição e utilização. Com base nesse contexto, esta pesquisa questiona: como desenvolver um sistema semelhante a esses com um custo mais acessível para ser utilizado no dinamômetro localizado no laboratório do grupo ADEMCI, do IFSC - Câmpus Florianópolis?

1.3 Objetivo geral

Construir um sistema completo de aquisição de dados e controle em malha fechada para o freio dinamômetro utilizado no laboratório do grupo ADEMCI, do IFSC - Câmpus Florianópolis, que será utilizado para testes de potência de motores à combustão.

1.4 Objetivos específicos

Os objetivos específicos descritos nessa seção nos levam a alcançar o objetivo geral deste trabalho:

a) desenvolver *hardware* para aquisição dos dados e acionamento do freio dinamômetro;

b) modelar matematicamente o sistema a ser controlado;

c) projetar o sistema de controle a ser utilizado;

d) criar interface gráfica para o sistema;

e) avaliar os resultados alcançados comparando-os com os obtidos em *software* matemático e os custos envolvidos no projeto.

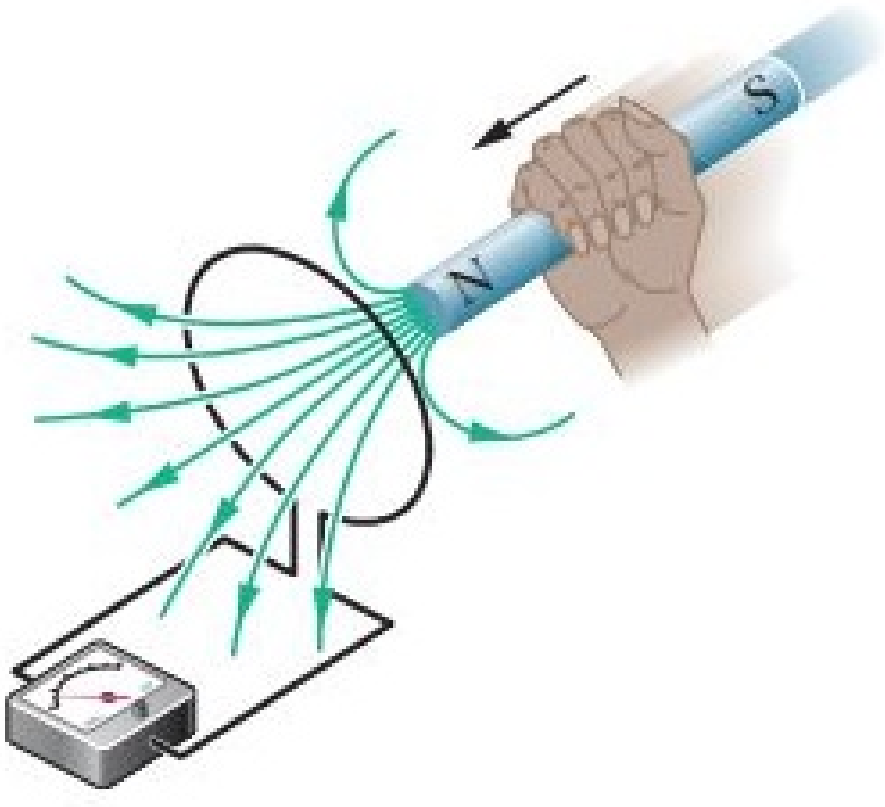
2 REVISÃO BIBLIOGRÁFICA

O objetivo deste capítulo é apresentar alguns dos conceitos teóricos que são relevantes para a esta pesquisa. Foram analisados, ainda, algumas partes que constituem a bancada de teste do laboratório do grupo ADEMCI, que serviu-nos como base para este trabalho.

2.1 Correntes Parasitas

Para entender o que são correntes parasitas, é possível imaginar a seguinte situação: aproxima-se um ímã de um circuito fechado feito de material condutor e se tem ligado a ele somente um amperímetro, como visto na Figura 1.

Figura 1 – Correntes induzidas com a variação do campo magnético



Fonte: Halliday, Resnick e Walker (2011)

Com essa aproximação aumenta-se a quantidade de linhas de campo magnético que passam através da área desse circuito, o qual também é chamado de fluxo magnético. A Equação 1 mostra a definição de fluxo magnético.

$$\Phi_B = \int \vec{B}.d\vec{A} \tag{1}$$

Onde Φ é o fluxo magnético, B representa o campo magnético e A é a área do circuito que é atravessada pelo campo.

Esse aumento do fluxo magnético criado pelo movimento relativo entre o ímã e o circuito causa uma força eletromotriz nesse circuito, que segundo(HALLIDAY; RESNICK; WALKER, 2011), pode ser descrito como na Equação 2:

$$\epsilon = -\frac{d\Phi_B}{dt} \tag{2}$$

Onde ϵ representa a força eletromotriz induzida no circuito e o sinal negativo indica que o sentido dessa força eletromotriz tem um sentido contrário à regra da mão direita; isso será muito importante para nosso sistema mais à frente.

Finalmente, essa força eletromotriz induz uma corrente elétrica no circuito, que pode ser visto nesse exemplo pela leitura do amperímetro e que pode ser calculada usando Equação 3:

$$i = \frac{\epsilon}{R} \tag{3}$$

Onde i representa a corrente no circuito e R sua resistência.

Analisando novamente as equações 1 e 2, é possível observar outra forma de alterar o valor desse fluxo magnético e induzir uma corrente no circuito. Pegando-se a Equação 1 e considerando um campo magnético constante, pode-se retirá-lo da integral, derivar dos dois lados e substituir $d\Phi_B$ na Equação 2, chegando-se assim na Equação 4.

$$\epsilon = -B\frac{dA}{dt} \tag{4}$$

Esta Equação 4 nos mostra que outra forma para criar uma força eletromotriz é alterar a área do circuito que está sob a influência deste campo magnético constante. Posteriormente substituindo o ϵ da Equação 3 pela definição da Equação 4, chega-se à Equação 5, que relaciona a corrente com a taxa de variação da área sob influência do campo magnético constante.

$$i = -\frac{B}{R}\frac{dA}{dt} \tag{5}$$

É apresentado a seguir uma característica muito importante sobre essas correntes induzidas no circuito.

Um condutor com uma corrente elétrica atravessando um campo magnético sofre efeito de uma força, conforme Equação 6 (HALLIDAY; RESNICK; WALKER, 2011).

$$F = i\vec{L} \times \vec{B} \tag{6}$$

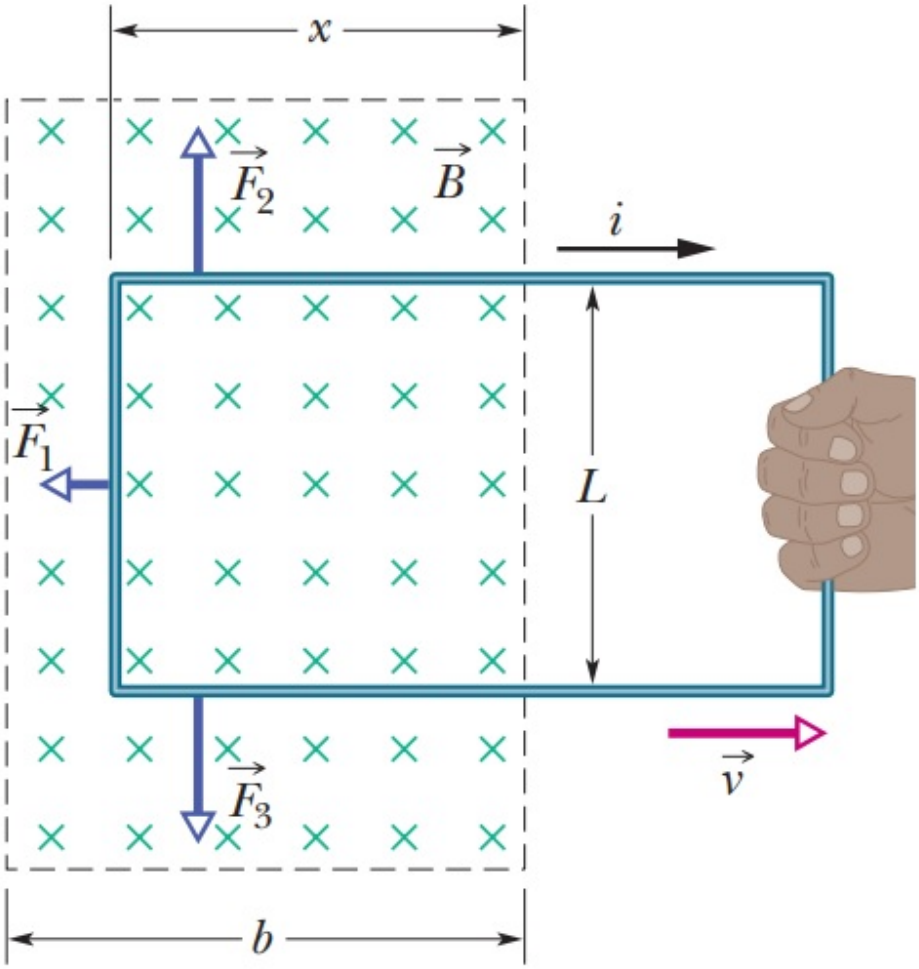
Onde F é a força que incide sobre o condutor e L é o comprimento desse condutor.

Se o campo magnético e a corrente são perpendiculares, é possível, então, utilizar a Equação 7, que trata apenas da relação do módulo da força com o módulo da corrente e do campo magnético.

$$|F| = |i|L|B| \tag{7}$$

No exemplo da Figura 2, tem-se um circuito fechado sendo retirado de uma região onde existe um campo magnético constante, essa movimentação do circuito faz diminuir a área que está sob o efeito do campo magnético e, por consequência, surge uma corrente elétrica e uma força que dependem da taxa de variação dessa área ou da velocidade do movimento do circuito.

Figura 2 – Correntes induzidas com a variação da área do circuito



Fonte: Halliday, Resnick e Walker (2011)

A Equação 8 mostra como calcular essa força que surge nesse circuito. Nela é perceptível que apenas a força que está no ramo de comprimento L é calculada, pois como a corrente nos dois ramos que tem comprimento x são de sentidos opostos, as forças geradas por esses dois ramos se cancelam.

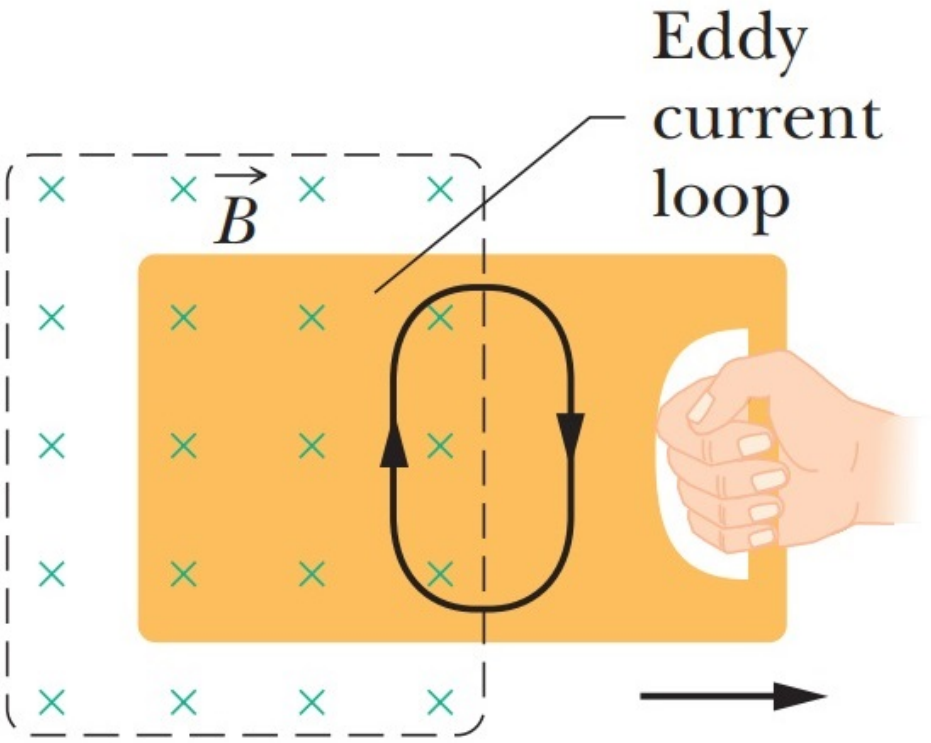
$$F = iLB = -\frac{B}{R}\frac{dA}{dt}LB = -\frac{B^2L^2}{R}\frac{dx}{dt} = -\frac{B^2L^2v}{R} \tag{8}$$

Nessa equação, v representa a velocidade em que o circuito está sendo movimentado para fora da região com o campo magnético constante. É interessante destacar que esse efeito de surgir uma força que se opõe à saída desse circuito da região do campo magnético também acontece quando o circuito está sendo empurrado para dentro dessa região, mas com sentido contrário.

Uma outra forma de apresentar o efeito descrito nesse exemplo seria tirar

uma placa sólida, feita de material condutor, de uma região onde se tem um campo magnético constante, Figura 3.

Figura 3 – Correntes parasitas

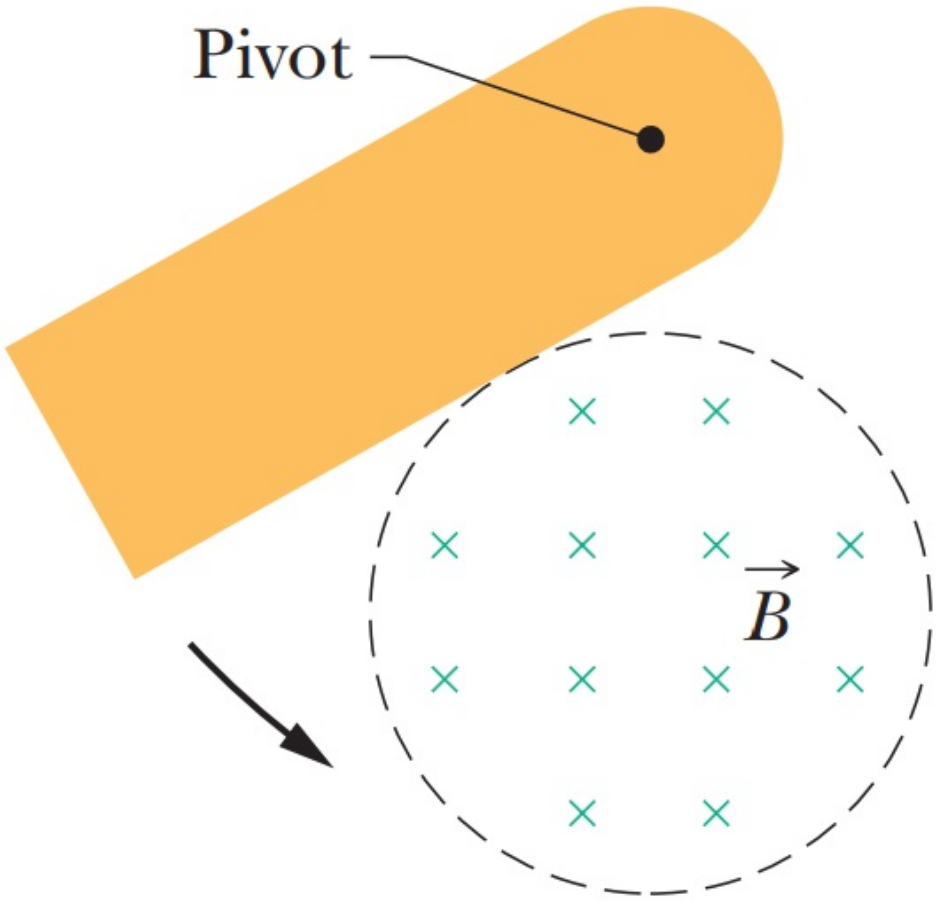


Fonte: Halliday, Resnick e Walker (2011)

Nesse caso, como no circuito fechado, haverá uma força eletromotriz induzindo correntes elétricas na placa, chamada de correntes parasitas, porém essas correntes não seguem um caminho bem definido como antes, agora é possível afirmar que "os elétrons circulam dentro da placa como se estivessem presos em um redemoinho de água", (HALLIDAY; RESNICK; WALKER, 2011). Como no caso anterior, agora também aparecerá uma força que irá se opor ao movimento de retirada da placa da região onde há o campo magnético.

Na Figura 4 é apresentado um exemplo de freio utilizando essa força gerada pelas correntes parasitas, que tem sentido contra a direção do movimento, onde a placa, presa a um pivô, é freada quando entra ou sai do campo magnético.

Figura 4 – Correntes parasitas



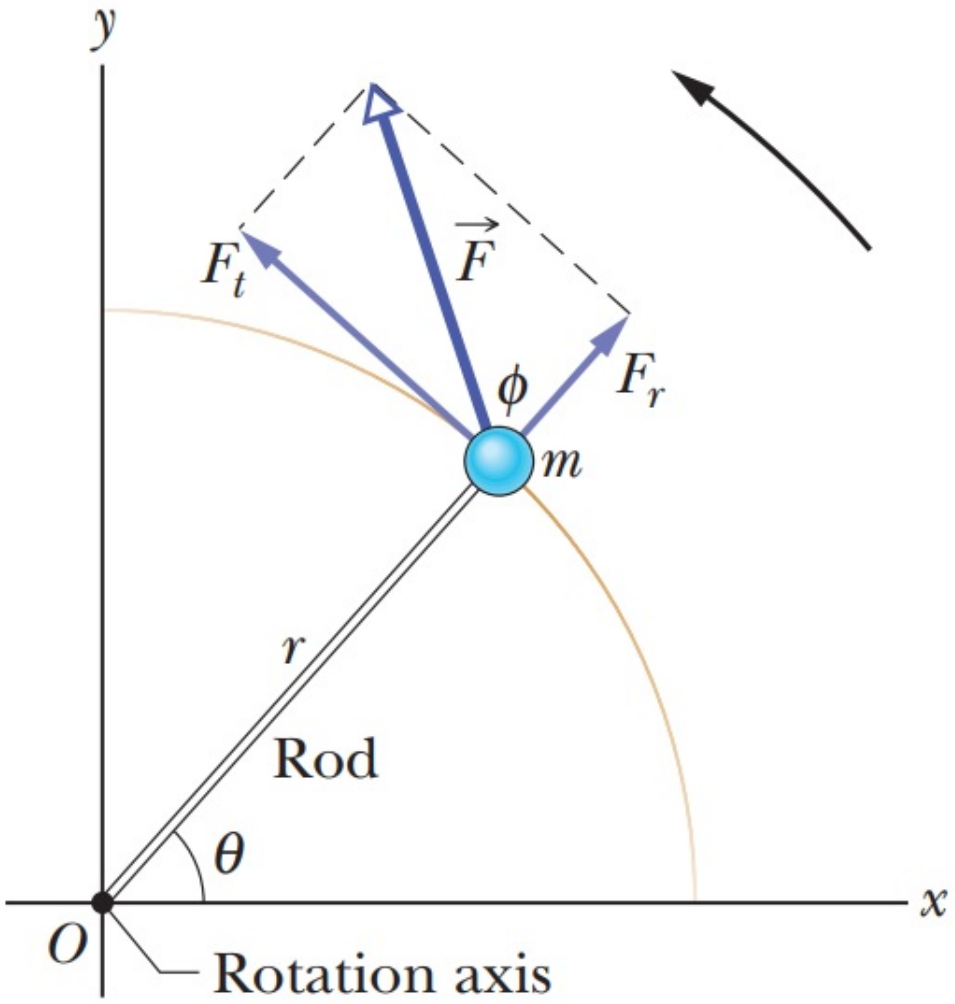
Fonte: Halliday, Resnick e Walker (2011)

A energia mecânica retirada da placa é convertida em energia térmica na mesma, já que o material, apesar de ser um condutor, possui uma resistência à passagem das correntes parasitas que surgem.

2.2 Torque

O torque é uma grandeza que está associada à capacidade de uma força girar um objeto e essa capacidade depende da força aplicada a esse objeto e também do braço de alavanca, que nada mais é que a distância entre o ponto em que é aplicada a força e o eixo de rotação (Figura 5).

Figura 5 – Torque gerado por uma força



Fonte: Halliday, Resnick e Walker (2011)

Segundo Halliday, Resnick e Walker (2011), a Equação 9 mostra a relação entre essas grandezas.

$$\tau = r(Fsin(\phi)) \tag{9}$$

Sendo τ o torque, r o comprimento do braço de alavanca, F a força aplicada e ϕ o ângulo entre a força e braço de alavanca.

Considerando que a força aplicada é perpendicular ao braço de alavanca, $\phi = 90^o$, pode-se simplificar Equação 9 à Equação 10 que relaciona apenas seus módulos.

$$\tau = rF \tag{10}$$

Para torque a unidade de medida utilizada no Sistema Internacional de Unidades é o Newton-metro, Nm, porém é comum utilizar-se para medidas de torque de motores a combustão o Quilograma-força metro, kgfm, e essa será a unidade utilizada neste trabalho. A relação entre essas duas unidades é feita utilizando a Equação 11.

$$\tau[kgfm] = g\tau[Nm] \tag{11}$$

Onde g representa a aceleração da gravidade.

Da mesma forma que a 2ª lei de Newton para movimentos lineares, Equação 12, relaciona a força, a massa (m) e a aceleração linear (a), pode-se relacionar o torque, a inércia (I) e a aceleração angular(α) utilizando a Equação 13.

$$F = ma \tag{12}$$

$$\tau = I\alpha \tag{13}$$

Nesta pesquisa é relacionado o torque à variação da velocidade de rotação, então, reescreveu-se a Equação 13 como a Equação 14, onde a aceleração angular é substituida pela derivada da velocidade de rotação em função do tempo.

$$\tau = I\frac{d\omega}{dt} \tag{14}$$

No sistema motor e dinamômetro teremos o torque resultante como a soma dos torques de ambos e podemos calcular o torque do motor utilizando o torque do dinamômetro e a variação da velocidade de rotação, Equação 15.

$$\tau_m = I\frac{d\omega}{dt} - \tau_d \tag{15}$$

Onde τ_m representa o torque do motor e τ_d o torque do dinamômetro.

É possível utilizar uma variação constante da velocidade de rotação para localizar pontos de máximo torque. Com a variação constante na velocidade o termo que depende da inércia, na Equação 15, se torna uma constante, chegando na Equação 16. Neste caso a o torque do motor será igual ao torque do dinamômetro, só que com um deslocamento feito pela constante e desta forma os pontos máximos e mínimos são iguais nos dois.

$$\tau_m = cte - \tau_d \tag{16}$$

2.3 Motor à Combustão Interna

Segundo Tillmann (2013), "Os motores de combustão interna realizam a transformação de energia térmica proveniente da combustão ou queima do combustível em energia mecânica". De uma forma bem simplificada é possível dizer que o motor a combustão interna tem como objetivo transformar a energia química armazenada no combustível em energia mecânica. Esta transformação ocorre com a queima do combustível, misturado ao ar, dentro do motor, que gera calor e causa expansão nos gases provenientes da queima, empurrando assim o pistão. Essa força que empurra o pistão é a energia mecânica que será aproveitada do motor.

A potência de um motor à combustão interna é a grandeza que determina a quantidade de trabalho que este motor está concedendo em determinado momento. Segundo Heywood (1988), esta potência pode ser calculada usando o torque e a velocidade de rotação do eixo. Assim, a Equação 17 tem o resultado em kW e a Equação 18 mostra o resultado em hp.

$$P = 2\pi N\tau.10^{-3} \tag{17}$$

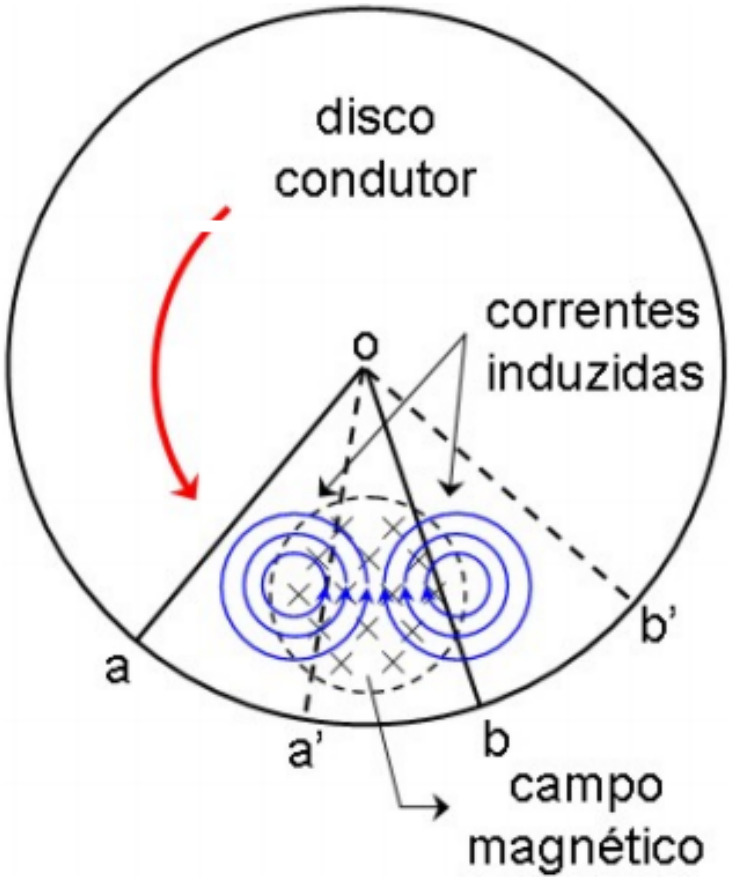
$$P = \frac{N\tau}{5252} \tag{18}$$

Onde P é a potência e N é a velocidade de rotação em RPM.

2.4 Dinamômetro

O dinamômetro é um equipamento que pode ser aplicado com o objetivo testar um motor, aplicando à ele diversas condições de uso e levantando informações, como torque, velocidade de rotação ou potência desse motor durante estes testes. Existem diversos tipos de dinamômetros, que se distinguem principalmente pelo tipo de freio que utilizam. O dinamômetro que foi utilizado nesta pesquisa foi o dinamômetro a correntes parasitas, neste tipo de dinamômetro há eletroímãs que criam regiões com campo magnético e correntes parasitas surgem em um disco de material condutor que gira, passando por essas regiões. O objetivo da criação dessas regiões é utilizar a força gerada por essas correntes parasitas para frear o disco, que está acoplado ao motor pelo eixo de rotação. A Figura 6 demonstra como são essas correntes parasitas no disco do freio.

Figura 6 – Correntes parasitas no disco



Fonte: (SOUZA, 2005)

Dentro das regiões onde há campo magnético as correntes parasitas seguem o mesmo sentido, fazendo com que a força gerada por essas correntes tenham também o mesmo sentido, contra o movimento do disco, desta forma, pode-se controlar a intensidade da força de frenagem gerada controlando a intensidade do campo magnético gerado pelos eletroímãs, o que por sua vez, é controlado pela tensão que é aplicada à eles.

3 METODOLOGIA

Para o desenvolvimento do protótipo de um sistema que possa sanar a problematização apresentada neste trabalho, foi feita inicialmente uma revisão bibliográfica sobre dinamômetros a correntes parasitas, motores a combustão interna e conceitos de física envolvendo movimentos giratórios. Após este estudo teórico, foi feita uma análise sobre o funcionamento do sistema, formado pelo conjunto motor e dinamômetro, e do atuador e sensores que seriam necessários para a solução empregada.

Foram desenvolvidos os projetos dos circuitos eletrônicos e das placas de circuito impresso destes, além de feitas simulações das respostas dos sensores e do microcontrolador. A interface, que propicia o armazenamento dos dados dos testes, possibilitou o levantamento da resposta ao degrau do sistema e fazer sua modelagem matemática.

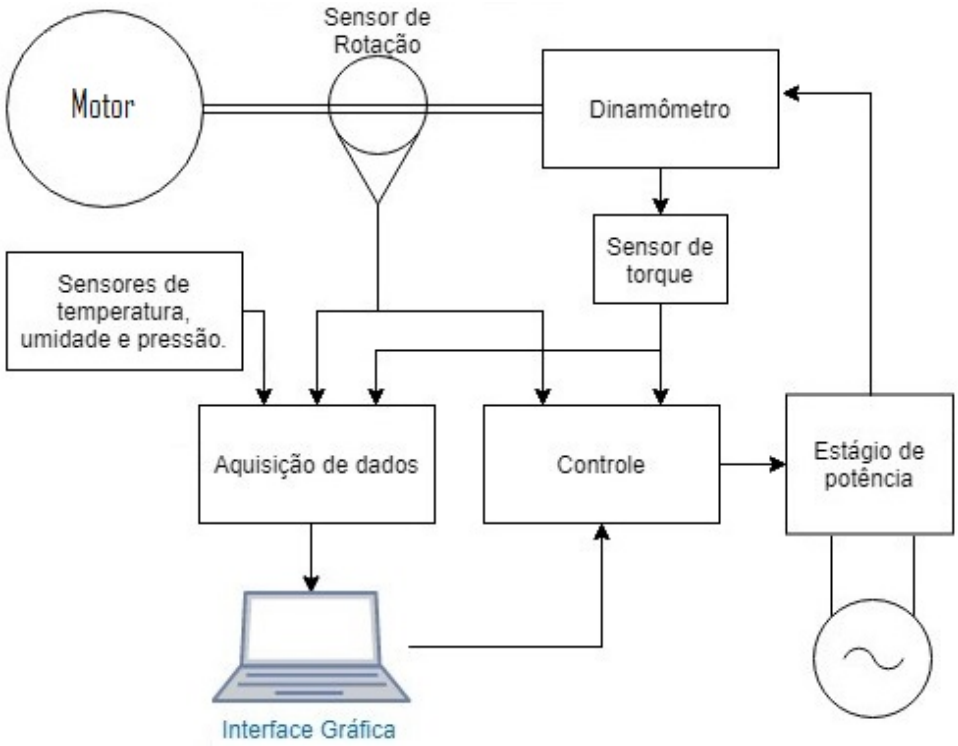
Foram feitos novos estudos sobre sistemas de controle, buscando um adequado ao trabalho e as características da resposta desejada, e após a escolha do tipo que seria empregado, iniciou-se um ciclo, de simulações e testes, a fim de refinar os resultados e se atender os requisitos especificados.

4 DESENVOLVIMENTO DO PROJETO

Este capítulo demonstra como foi organizado o sistema de controle e aquisição de dados, objeto desta pesquisa, apresenta-se como os dados necessários são medidos e como o sistema projetado atua no freio dinamômetro para promover seu controle.

A Figura 7 mostra um diagrama completo, com o dinamômetro e o motor, que é sistema a ser controlado, e o sistema de aquisição e controle que foi desenvolvido neste projeto.

Figura 7 – Diagrama completo do sistema



Fonte: Autoria própria (2019).

Este diagrama completo do sistema demonstra que o motor está acoplado ao freio dinamômetro através do eixo de rotação, sendo assim a velocidade de rotação será a mesma entre eles. O torque, que é lido no sistema, é o torque que é gerado pelo freio dinamômetro, que é diferente do torque que é gerado pelo motor. É possível calcular o torque do motor utilizando as informações de rotação e torque lidas utilizando a Equação 15. O torque apresentado nesta equação é o torque total do sistema, que em nosso caso é a soma dos torques do motor e do freio. As informações de torque do freio e velocidade de rotação é que são utilizadas pelo sistema de controle. O sistema de

aquisição de dados recebe tanto os sensores que mostram o estado atual do sistema, torque e velocidade de rotação, quanto os dados do ambiente de testes, temperatura, umidade e pressão. A interface gráfica do sistema serve para modificar os parâmetros e referências do sistema de controle e para salvar os dados e possibilitar a comparação com dados de testes anteriores. É possível observar também que o sistema de controle não atua diretamente no dinamômetro, e sim o faz utilizando um estágio de potência comercial, que tem como objetivo suprir as necessidades de tensão e corrente do freio dinamômetro.

4.1 Leitura dos Dados

Esta seção trata como foi implementada a leitura dos dados do sistema, é apresentado como os sinais foram tratados e condicionados para a adequação à leitura do microcontrolador.

4.1.1 Torque

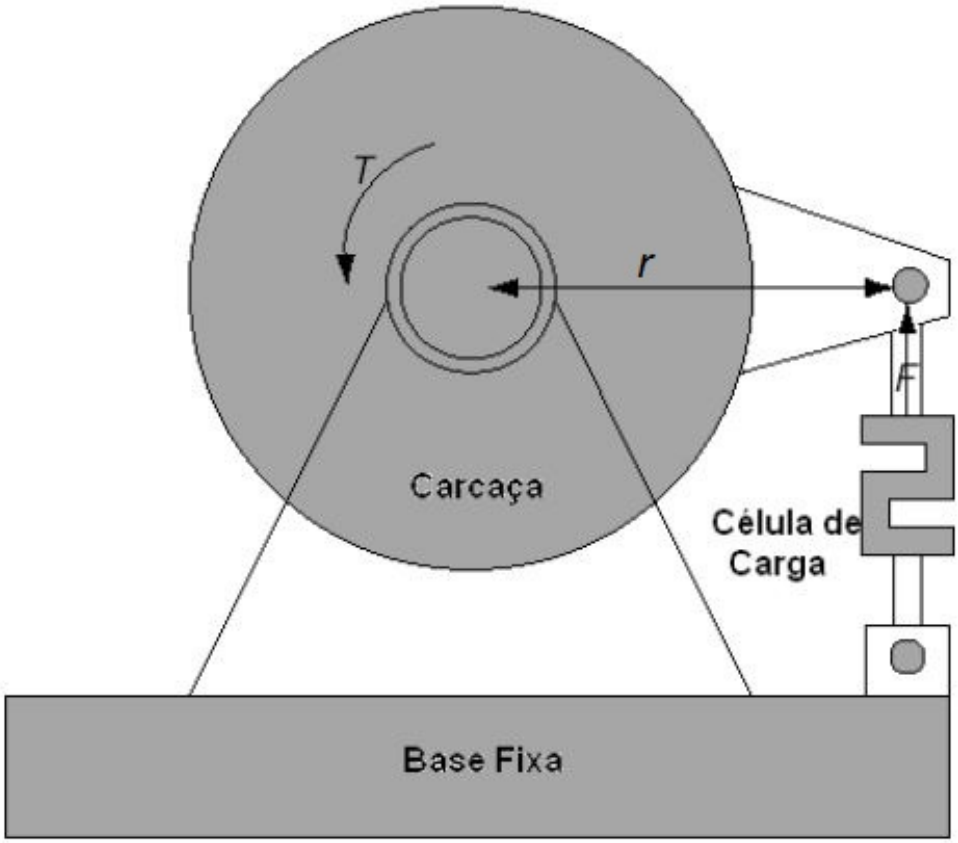
A aquisição do torque do dinamômetro foi feita através de uma célula de carga, que é um transdutor de força para um sinal elétrico. O sinal de saída da célula de carga é um sinal de tensão diferencial e que segundo TAI SENG MACHINERY (2000), tem uma resposta de acordo com a Equação 19.

$$V_s = 0,002 V_a \frac{F}{F_{max}} \tag{19}$$

Nessa equação, V_s representa o tensão diferencial de saída do sensor, V_a é a tensão em que foi alimentado o sensor, F é a força sobre o sensor e F_{max} é a força máxima que o sensor pode ler.

Esta célula de carga foi utilizada na fixação da carcaça desse dinamômetro, Figura 8.

Figura 8 – Esquema de fixação da célula de carga

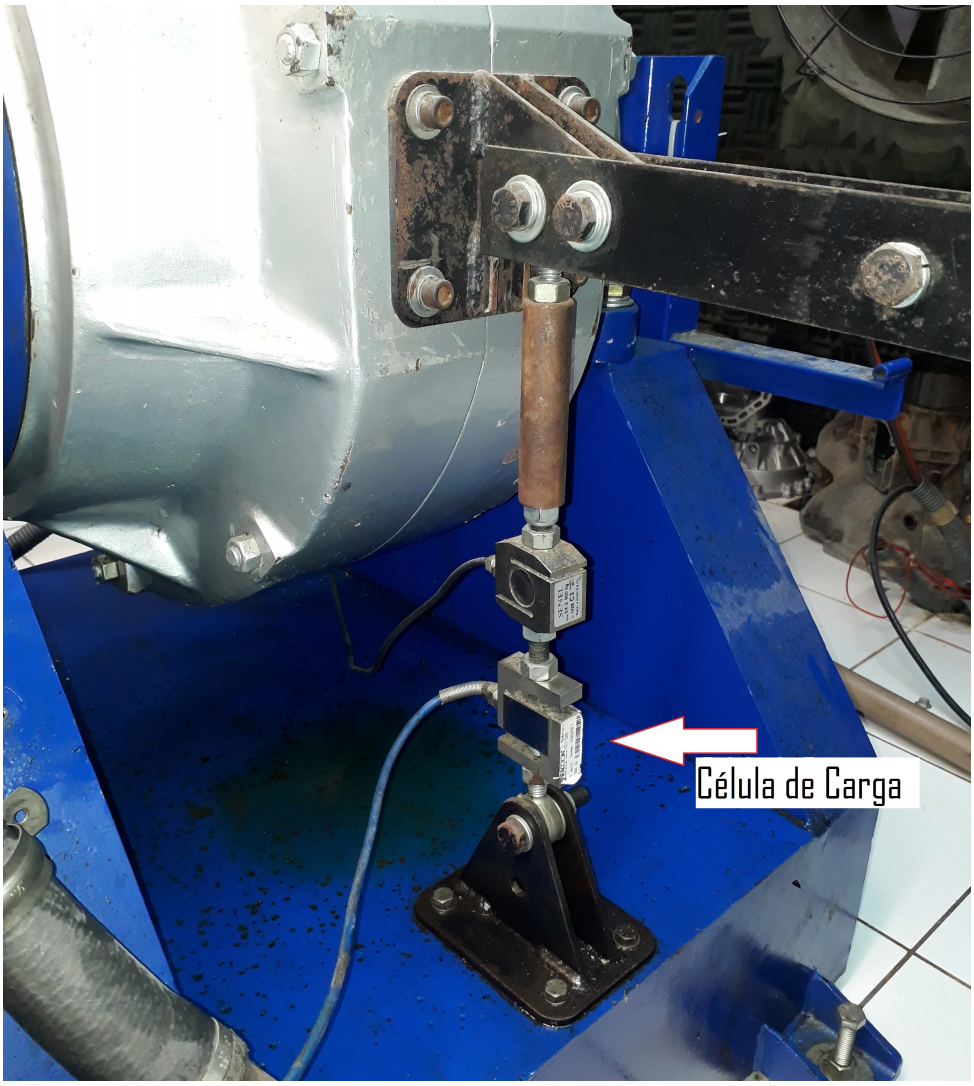


Fonte: (HAICAL, 2009)

Como a célula de carga é o único ponto de fixação da carcaça, e esta não gira, essa célula de carga sofre toda a força gerada pelo torque do freio, e como a fixação fica perpendicular ao eixo de rotação e ao braço de alavanca se tem uma relação direta entre a força lida e o torque do freio utilizando a Equação 10.

A Figura 9 mostra a célula de carga utilizada nesse projeto, já fixada à carcaça do freio.

Figura 9 – Esquema de fixação da célula de carga

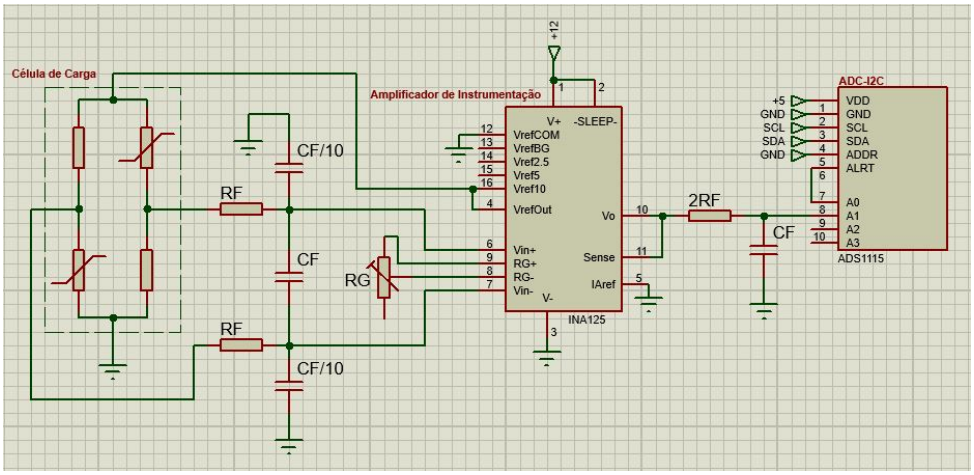


Fonte: Autoria própria (2019)

A célula de carga utilizada foi a CZL-301-500, projetada para leituras de até 500 Kg de massa apoiadas diretamente sobre ela. Utilizando a segunda lei de Newton, Equação 12, com a aceleração da gravidade, g=9,81 m/s^2, é calculado que ela pode ler uma força de até 4905 N ou 500 kgfm. Já que a célula de carga está fixada a uma distância de 30 cm do eixo de rotação, consegue fazer leituras de torque de até 1471,5 Nm ou 150 kgfm, utilizando a Equação 10.

A célula de carga é alimentada com uma tensão de 10 V. Utilizando a Equação 19, calcula-se que a tensão de resposta do sensor quando estiver com a sua máxima leitura será de 20 mV. Foi preciso, então, fazer um condicionamento para elevar a tensão deste sinal e melhorar sua leitura, Figura 10.

Figura 10 – Condicionamento do sinal da célula de carga



Fonte: Autoria própria (2019).

Nesse condicionamento, o sinal é filtrado e amplificado, para se adequar a faixa de leitura do conversor analógico digital utilizado, que é de 0 a 3,3 V. Foram implementados dois filtros RCs, que evitam que sinais de alta frequência causem leituras erradas em nosso sistema e foram feitos visando uma frequência de corte de 5 Hz, que é uma frequência que engloba o espectro de frequências do sinal a ser lido. Utilizando R_f = 220 kΩ e $2R_f$ = 470 kΩ e os capacitores C_f= 68 nF, foi alcançado frequências de corte de 5,32 Hz para o primeiro filtro e 4,98 Hz para o segundo. A amplificação do sinal foi feita utilizando um amplificador de instrumentação projetado para ser utilizado com células de carga, o INA125. Além da amplificação também fornece uma tensão de referência que é usada para regular a tensão de alimentação da célula de carga. Segundo Texas Instruments (2009), a tensão de saída desse amplificador de instrumentação tem como referência a tensão colocada no em seu pino 5 , IAREF. Foi aplicado uma tensão de 2,5 V nesse pino e com essa referência deslocada positivamente é possível fazer leituras de torque com valores negativos, teóricamente não teremos valores negativos nas leituras de torque feitas, mas leituras negativas podem ocorrer em sensores que estejam desbalanceados e assim com deslocamento em sua leitura.

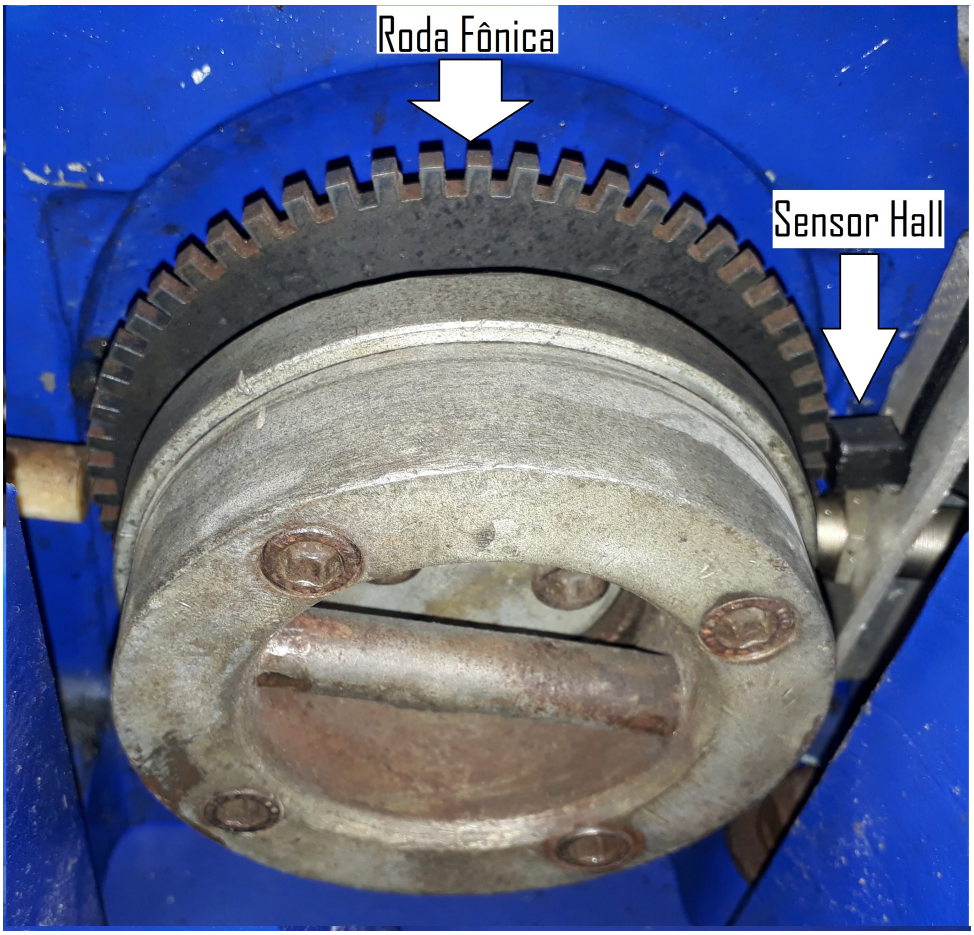
Para uma menor discretização da leitura do torque foi considerado uma leitura de torque de até 50 kgfm, ou 490,5 Nm, e com essa especificação o estágio de ganho foi ajustado manualmente para promover um melhor aproveitamento da extensão de leitura de nosso ADC. Para este ajuste manual, foi adicionado um peso de 50 kg a uma distância de 1 m do eixo de rotação, aplicando na célula de carga a mesma força que será aplicada quando o freio gerar um torque de 490,5 Nm ou 50 kgfm. O ganho, então, foi ajustado, através do resistor variável, R_g, para que a leitura feita fosse próxima ao valor limite, ou saturação.

Após o estágio de condicionamento, o sinal da célula de carga já está apropriado para a aquisição. Essa aquisição é feita por um módulo ADC que utiliza o ADS1115 para a conversão analógico digital e envia os dados lidos ao microcontrolador utilizando comunicação I2C.

4.1.2 Velocidade de Rotação

Para a leitura da velocidade de rotação foi utilizado uma roda fônica acoplada ao eixo de rotação e um sensor de efeito Hall (Figura 11).

Figura 11 – Roda fônica e sensor Hall



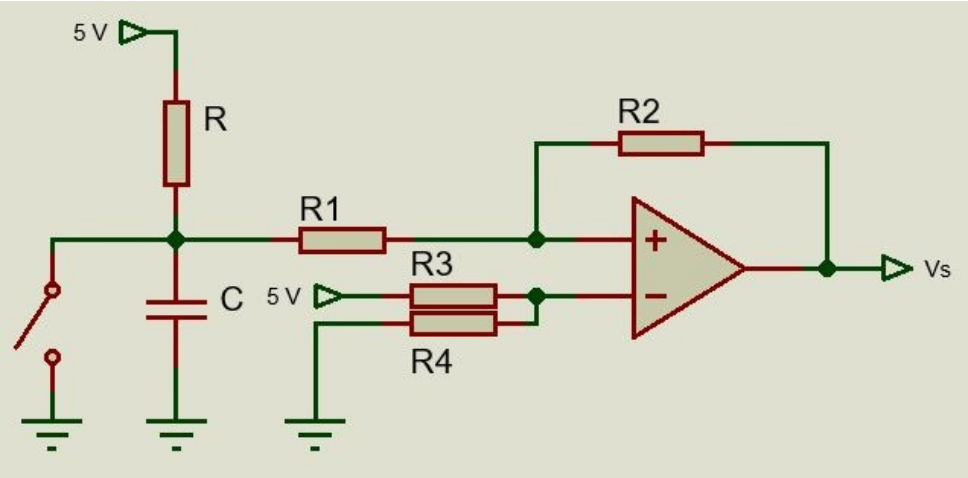
Fonte: Autoria própria (2019).

Para seu funcionamento, o sensor de efeito Hall utilizado necessita de um resistor de 10 kΩ conectado entre seu pino de saída e seu pino de alimentação e tem como sinal de saída uma série de pulsos com amplitude igual da alimentação desse sensor, onde cada pulso representa um dente da roda fônica. Utilizando o sinal de saída desse sensor ligado diretamente a um pino digital de um microcontrolador é possível detectar a passagem dos dentes, mas quando este método foi testado, foi

observado que o uso deste sinal sem um devido tratamento deixa a leitura suscetível a erros, causados por ruídos externos.

Para se evitar problemas com ruídos e promover uma correta detecção da passagem dos dentes da roda fônica, foi feito um estágio de condicionamento onde o sinal gerado pelo sensor é filtrado e comparado, com uma tensão de referência, utilizando histerese. A Figura 12 mostra como foi implementado o circuito de condicionamento de sinal.

Figura 12 – condicionamento do sinal do sensor Hall



Fonte: Autoria própria (2019).

Nesta figura o sensor de efeito Hall é representado por uma chave, que liga o sinal de entrada do condicionamento à tensão de referência. Os Resistores R_3 e R_4 servem para criar um divisor de tensão entre as tensões de alimentação desse circuito, 5 V e 0 V. Eles têm o mesmo valor, 22 kΩ, então a tensão de referência fornecida ao comparador será de 2,5 V. Os Resistores R_1 e R_2, servem para causar a histerese na resposta do condicionamento.

Para calcular as tensões necessárias no sinal de entrada para fazer o circuito comutar sua saída foi feita uma análise nas correntes elétricas, nos momentos de transição do sinal de saída. Com a Equação 20 é possível calcular o valor da corrente que passa através do resistor R_2 quando no pino de entrada do comparador tem-se 2,5 V, que é a tensão de referência, limiar das comutações da saída do comparador.

$$i = \frac{2,5 - V_o}{R_2} \tag{20}$$

O valor da corrente que entra no pino de entrada do comparador é muito pequena e pode ser desconsiderado. Assim, o valor da corrente que passa através de

R_1 será o mesmo da corrente que passa através de R_2 e com isso é possível saber sua queda de tensão, e por consequência a tensão de entrada que vai causar a comutação.

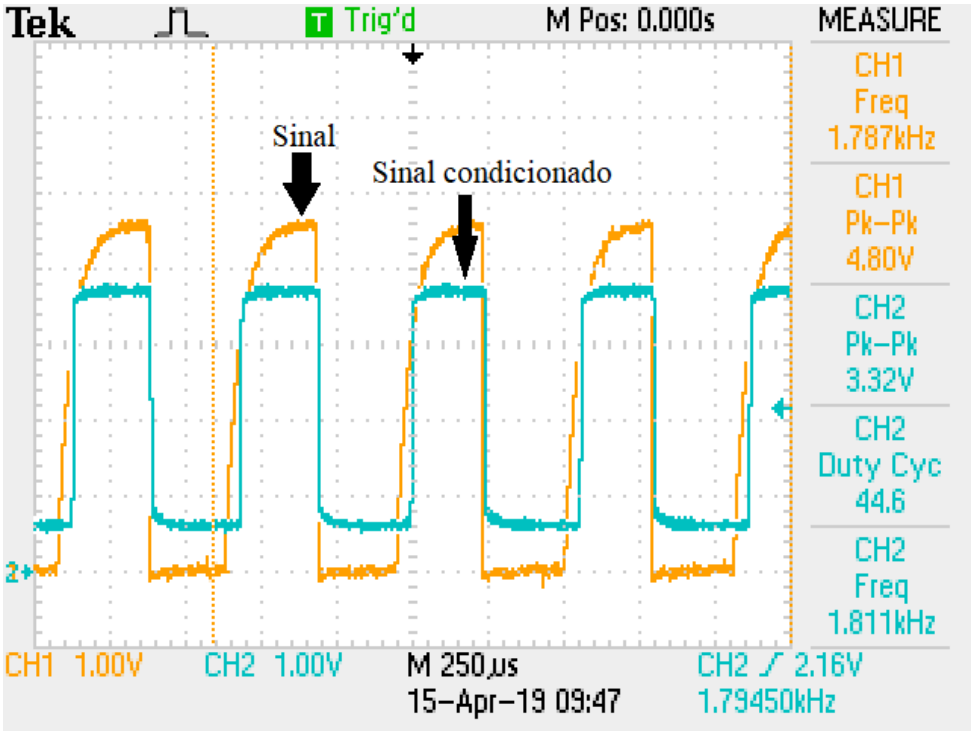
Usando os valores de saturação de V_s, 0 e 5 V, foram calculados os valores de corrente de $\pm$ 50 μA, que passando por R_1 causarão queda de tensão de $\pm$ 0,5 V. Isso quer dizer que para comutar para a saturação positiva o sinal de entrada terá que estar com 3 V, 0,5 V acima da tensão de referência e para comutar para a saturação negativa terá que estar com 2 V, 0,5 abaixo da tensão de referência. Isso nos dá uma janela de histerese de 1 V.

Na prática as tensões de saturação ficaram em 0,5 V e 4,5 V, o que levou as tensões de 2,1 V e 2,9 V nos momentos de comutação. Estes valores proporcionaram, então, uma janela de histerese de 0,8 V.

O capacitor, C, de 1 nF, foi adicionado para filtrar ruídos de alta frequência que incidam sobre o circuito. Junto com o resistor R, de 10 kΩ, ele vai limitar a velocidade da comutação quando o sensor se comporta como uma chave aberta. A frequência de corte deste filtro ficou em 15,9 kHz e foi escolhida pois não interfere na leitura do sensor Hall.

A Figura 13 mostra o sinal do sensor Hall antes e depois do condicionamento.

Figura 13 – Sinal de rotação condicionado



Fonte: Autoria própria (2019).

Após esse estágio de condicionamento, o sinal pôde ser lido por um pino

digital do microcontrolador.

Já no microcontrolador, é possível calcular a velocidade de rotação do eixo, em RPM, utilizando o número de pulsos gerados em determinado tempo de aquisição, Equação 21, ou utilizando o intervalo entre cada dente, Equação 22.

$$N = \frac{n_p}{n_d(t_a/60)} \tag{21}$$

Onde N representa a velocidade de rotação, n_p o número de pulsos,n_d o número de dentes da roda fônica e t_a o tempo de aquisição.

$$N = \frac{1}{n_d(t_e/60)} \tag{22}$$

Onde t_e representa o tempo entre um dente e outro.

A roda fônica que foi acoplada ao eixo possui 60 dentes, logo, para cada revolução do eixo obtém-se 60 pulsos gerados pelo sensor hall.

Utilizando a Equação 22 tem-se a possibilidade de uma maior taxa de aquisição, tendo em vista que é necessário a passagem de apenas dois dentes para o levantamento da rotação, porém ao utilizar este método foi percebido uma grande sensibilidade do mesmo frente a ruídos, não só eletromagnéticos como também mecânicos. O método da Equação 21 é muito menos suscetível a ruídos, porém sua discretização é grande em taxas mais altas de aquisição.

Utilizando a Equação 21 e sabendo que o número de pulsos é um número inteiro foi montada a Tabela 1 para se ter uma ideia da discretização sofrida no resultado com a roda fônica de 60 dentes.

Tabela 1 – Discretização do método de contagem de pulsos

Período de Aquisição[ms]	Discretização[RPM]
10	100
20	50
50	20
100	10
1000	1

Fonte: Autoria própria (2019).

Sabendo que o sinal gerado pelo sensor Hall é quadrado, onde o tempo em alto é igual ao tempo em baixa, é possível utilizar uma detecção de borda do sinal, detectando-se as bordas de subida e de descida. Assim a leitura nos fornece uma resolução da metade do valor que foi possível utilizando o método de contagem de

pulsos. Utilizando as mesmas frequências de amostragem apresentadas na Tabela 1 foi montada a Tabela 2, que demonstra a discretização sofrida na leitura de rotação quando é utilizado este método de contagem de bordas.

Tabela 2 – Discretização do método de contagem de bordas

Período de Aquisição[ms]	Discretização[RPM]
10	50
20	25
50	10
100	5
1000	0,5

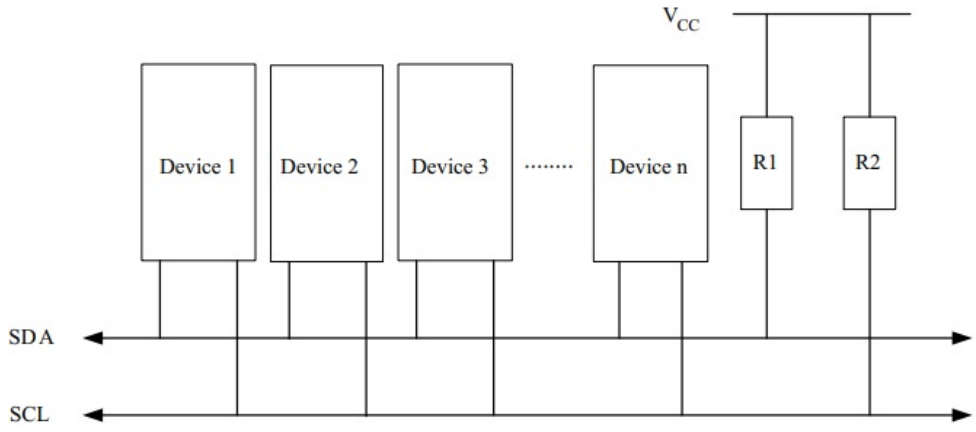
Fonte: Autoria própria (2019).

Durante esta pesquisa foram utilizados os métodos de contagem de pulsos e de contagem de bordas por serem mais imunes à ruídos, ambos com o período de amostragem de 100 ms.

4.1.3 Pressão, Temperatura e Umidade

Para a medição das condições do ambiente de testes, temperatura, pressão e umidade, foi utilizado um módulo que tem como base o sensor BMP280. Este módulo é configurado e transmite as leituras feitas utilizando a comunicação I2C. Foi necessário o desenvolvimento de uma pequena biblioteca para o microcontrolador, que conta com funções para a configuração do módulo e para a leitura das informações passadas via I2C. A conexão entre o dispositivo e o microcontrolador, Figura 14, foi feita utilizando o indicado em (ATMEGA328/P, 2016).

Figura 14 – Conexão de dispositivos I2C



Fonte: (ATMEGA328/P, 2016)

4.2 Estágio de potência

O estágio de potência utilizado em nosso sistema foi um controlador de potência da série SPC1-50-E, apresentado na Figura 15. Ele é um conversor de tensão alternada para tensão contínua e têm como objetivo fornecer a corrente e a tensão necessários ao funcionamento do freio, que não podem ser fornecidos diretamente pelo microcontrolador, já que este não tem potência suficiente para fornecer esses sinais.

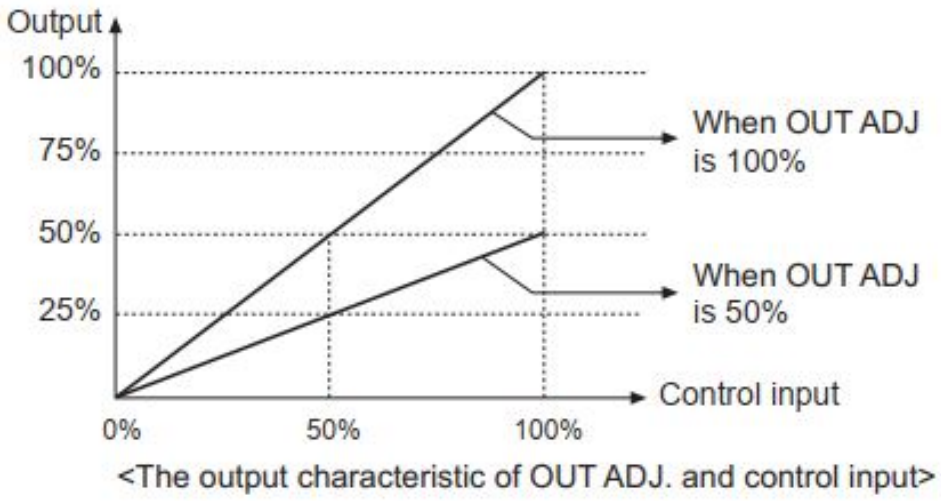
Figura 15 – Controlador de potência

Fonte: (AUTONICS, 2019)

Este módulo de potência foi utilizado no modo de saída ajustada, onde a tensão de saída desse estágio é proporcional a um sinal de tensão de acionamento. O sinal de acionamento indicado no manual fica entre 1 V e 5V, levando a tensão de saída de 0 V até um limite ajustado manualmente. A Figura 16 é um gráfico apresentado pelo manual do equipamento e que mostra como o ajuste manual influencia na tensão

de saída. A tensão máxima que esse estágio de potência pode fornecer ao freio é a tensão eficaz da rede elétrica, em nosso caso 220 V, e a corrente máxima é 50 A.

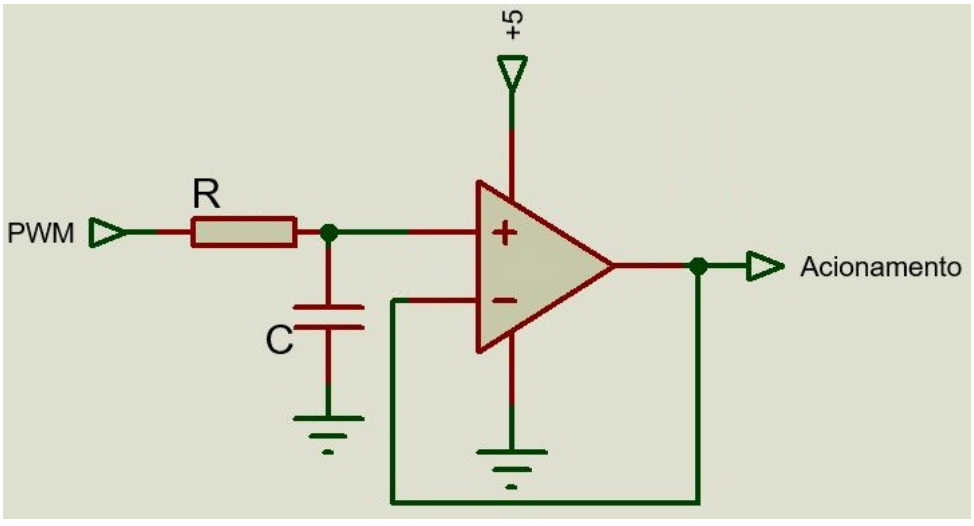
Figura 16 – Saída do controlador de potência



Fonte: (AUTONICS, 2019)

O acionamento deste estágio de potência foi feito utilizando um sinal modulado por largura de pulso, PWM, que foi filtrado com o intuito de fornecer ao estágio de potência um sinal de acionamento que seja, quase, constante e com valor da média do sinal de PWM, Figura 17.

Figura 17 – Filtro do sinal do acionamento do estágio de potência



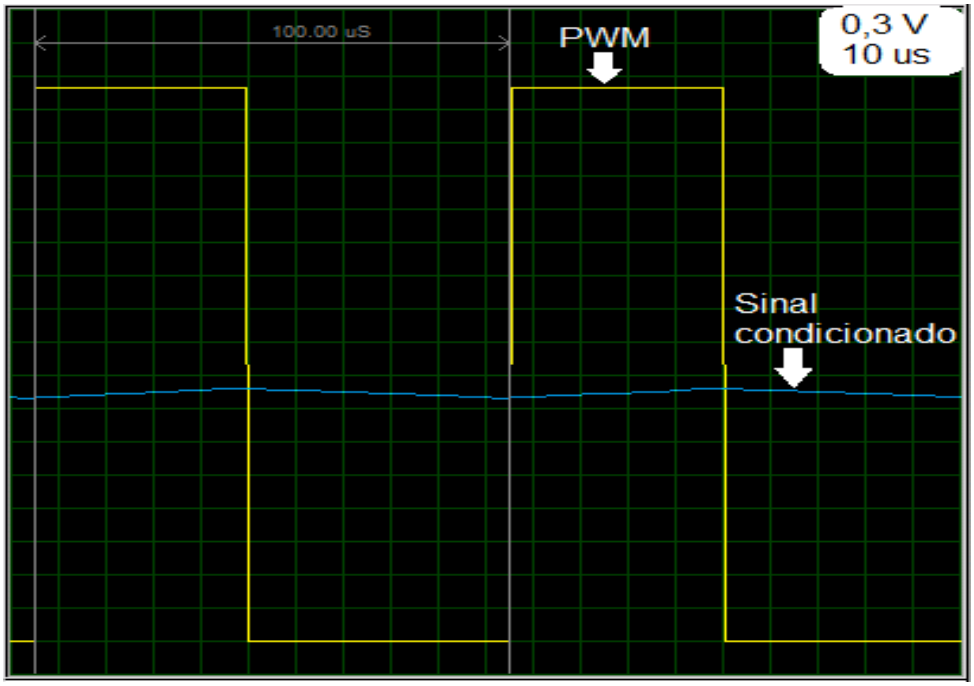
Fonte: Autoria própria (2019).

Quanto mais alta é a frequência desse sinal PWM, mais fácil será sua filtragem, se utilizando de componentes menores. A frequência deste sinal PWM foi

definida em 10 kHz. A frequência de corte do filtro ficou definido em 106,10 Hz e foi implementado utilizando R = 15 kΩ e C = 100 nF. Após o filtro foi utilizado, ainda, um amplificador operacional como um seguidor de tensão, servindo para isolar as impedâncias de saída do filtro e entrada do estágio de potência.

A Figura 18 mostra a simulação do sinal de PWM, antes e depois do condicionamento e foi constatado que o sinal após o condicionamento, pode ser usado para acionar o estágio de potência pois é praticamente constante, como proposto.

Figura 18 – Sinal do acionamento do estágio de potência



Fonte: Autoria própria (2019).

4.3 Modelagem

Esta seção trata a forma que foi utilizada para o levantamento das características do sistema, motor e freio, e o objetivo deste levantamento é obter um modelo matemático que corresponda satisfatoriamente ao sistema real e que possa ser utilizado como referência para o projeto do sistema de controle.

4.3.1 Função de transferência

A forma utilizada para análise transitória do sistema foi a resposta ao degrau. Nesta técnica é aplicado um sinal degrau na entrada do sistema a ser modelado e observada a resposta na saída do sistema, a fim de levantar algumas características da dinâmica de seu comportamento. Nesta análise foi considerado que a resposta ao

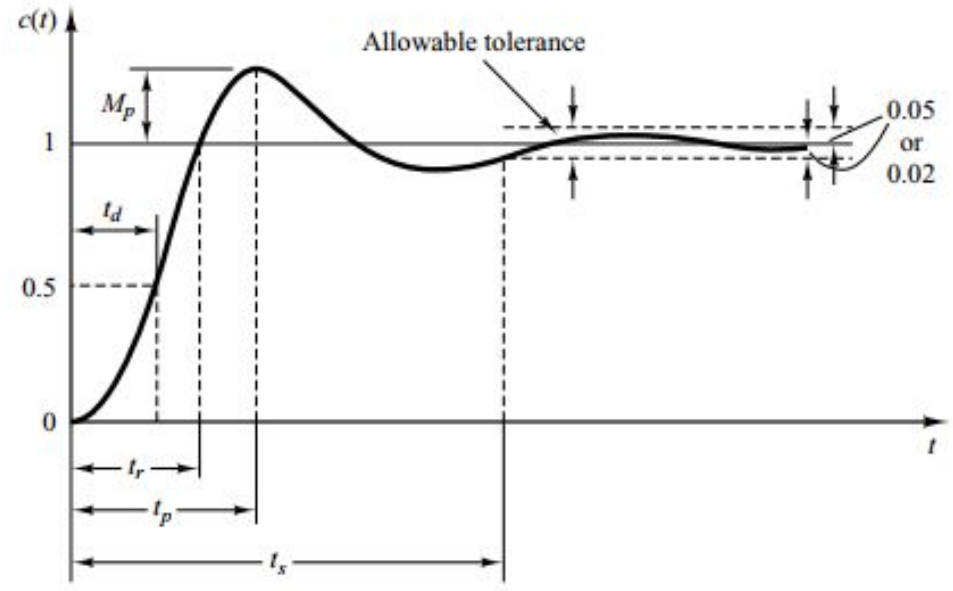
degrau do sistema em relação à velocidade de rotação pode ser modelado de forma satisfatória como um sistema de segunda ordem como apresentado em (OGATA, 2010), Equação 23.

$$G(s) = \frac{C(s)}{R(s)} = K_G \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{23}$$

Sendo ζ o fator de amortecimento do sistema ,ω_n é a frequência natural do sistema, K_G é o ganho do sistema, C(s) é o sinal de saída ,R(s) é o sinal de entrada e G(s) é a função de transferência do sistema.

A Figura 19 mostra a forma mais comum da resposta ao degrau de um sistema de segunda ordem e as características que podem ser levantas dessa resposta ao degrau.

Figura 19 – Características da resposta ao degrau



Fonte: Ogata (2010)

As características são:

a) M_p, valor de sobressinal máximo;

b) t_r, tempo de subida;

c) t_p, tempo de pico;

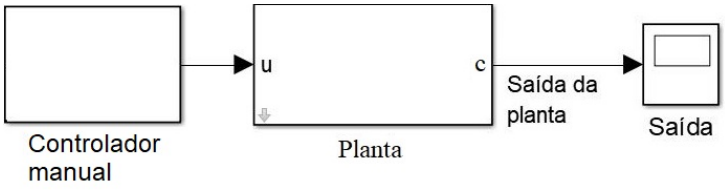
d) t_s, tempo de acomodação;

e) t_d, tempo de atraso;

Sabe-se que a dinâmica do motor a combustão não é linear, então foi definido um ponto de operação para fazer o degrau e gerar a modelagem do sistema. O ponto de operação escolhido para a modelagem foi com o motor a plena carga, pois é a plena carga que os motores à combustão tem sua potência máxima avaliada. A velocidade de rotação ficou definida em 4000 RPM, pois esta é uma velocidade de rotação média nos testes em motores a combustão, que geralmente vão de 1000 RPM a 6000 RPM. O estágio de potência foi ajustado a fornecer até 30 V em sua saída, essa tensão é suficiente para trabalhar com o motor no ponto de operação escolhido.

Para este teste de resposta ao degrau foi utilizado um controle manual do sistema, onde o operador controla a tensão que é aplicada ao freio, Figura 20.

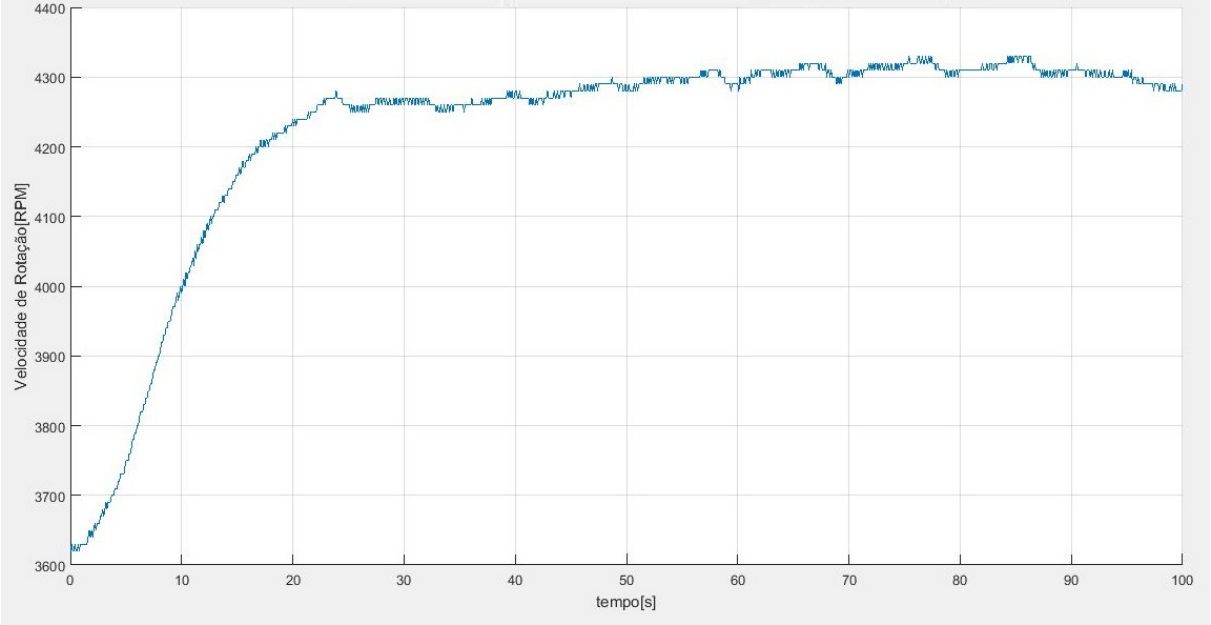
Figura 20 – Sistema de controle manual



Fonte: Autoria própria (2019).

No início do teste a saída do estágio de potência fornecia uma tensão de 18 V ao freio, mantendo a rotação em aproximadamente 3620 RPM. O valor de tensão foi reduzido em 0,67 V, ficando com 17,33 V. Este valor de degrau na tensão aplicada foi escolhido por ser suficiente para provocar uma variação na velocidade de rotação, mas mantendo-a ainda, dentro de uma faixa aceitável, próxima do ponto de operação escolhido. A velocidade de rotação se elevou até se estabilizar em torno 4300 RPM. A Figura 21 mostra a resposta da velocidade de rotação deste teste de resposta ao degrau.

Figura 21 – Resposta ao degrau da Velocidade de Rotação.



Fonte: Autoria própria (2019).

Para o cálculo do ganho do sistema, K_G, foi utilizado a Equação 24, sendo c_0 o valor inicial, c_{ss} o valor final de cada curva e V_d o valor de tensão aplicado no degrau.

$$K_G = \frac{(c_{ss} - c_0)}{V_d} \tag{24}$$

Utilizado os valores do teste e a Equação 24 foi calculado um K_G de -1020, ou seja, para um volt aplicado na entrada a rotação diminui 1020 RPM.

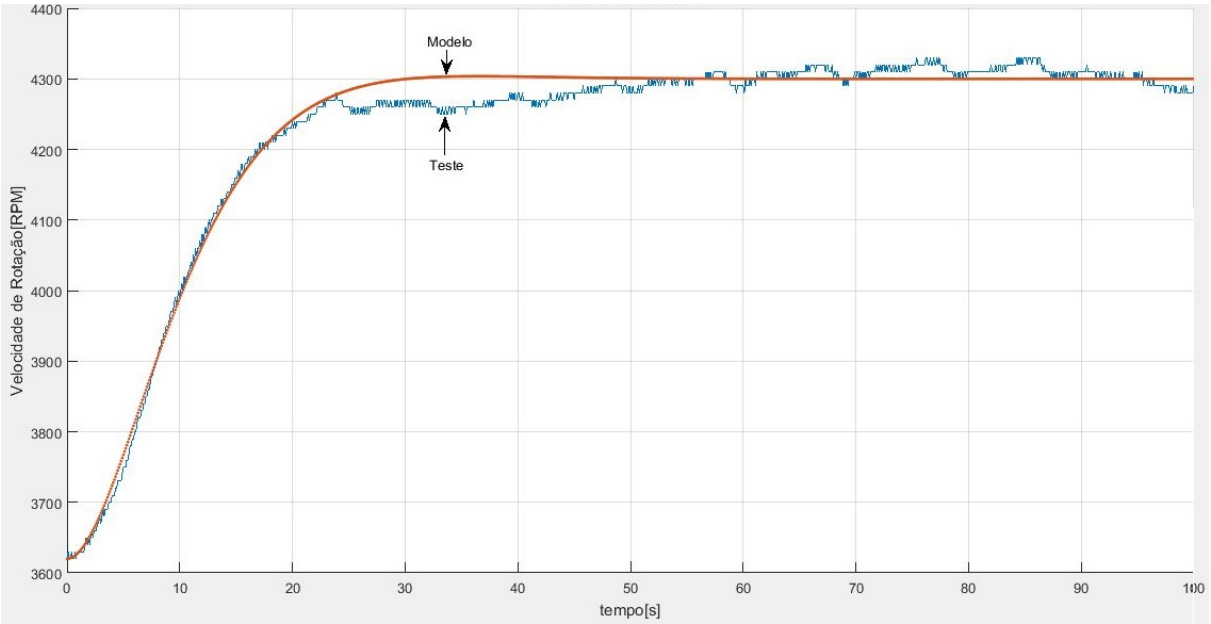
Para se determinar os valores para fator de amortecimento e frequência natural que representam satisfatoriamente o sistema, foi montada uma matriz com possíveis valores para eles e analisado a média do módulo do erro entre o modelo levantado e o resultado do teste no momento da mudança de rotação. Chegou-se a um ζ = 0,857 e um ω_n = 0,167 com um erro médio absoluto de 10,45 RPM entre os valores medidos e os valores simulados.

A função de transferência da velocidade de rotação ficou como apresentado em Equação 25.

$$G(s) = \frac{-28,45}{s^2 + 0,2862s + 0,02789} \tag{25}$$

A Figura 22 mostra o comparativo entre a simulação utilizando a modelo levantado e o teste feito na prática.

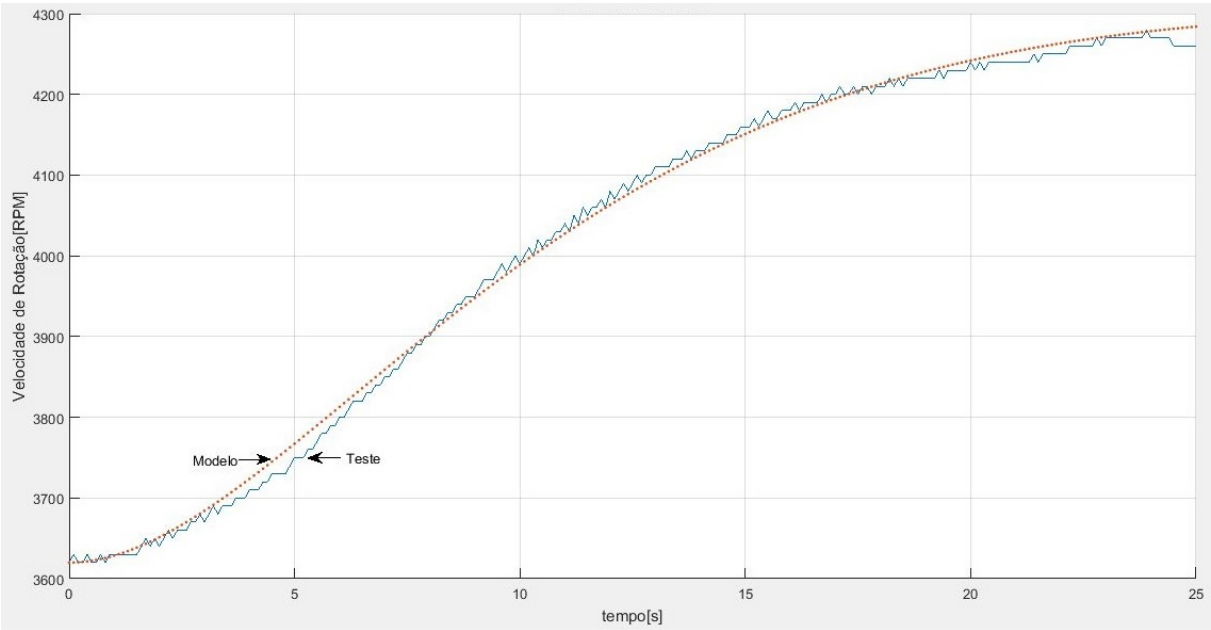
Figura 22 – Comparacao do degrau da velocidade de rotação



Fonte: Autoria própria (2019).

Aproximando a Figura 22 chegou-se à, Figura 23. Nesta figura é mais fácil observar o quanto a simulação ficou próxima ao teste feito na prática.

Figura 23 – Comparacao do degrau da velocidade de rotação



Fonte: Autoria própria (2019).

4.3.2 Função de transferência discreta

Como o estágio de controle foi implementado de forma digital, foi necessário o levantamento de um modelo discreto da planta para projetá-lo e testá-lo.

Segundo Ogata (1994), a resposta transitória de um sistema de controle digital depende do período de amostragem escolhido. Um período de amostragem muito grande causa detrimento à estabilidade do sistema, além disso, reduzir o período de amostragem permite que o valor crítico do ganho para a estabilidade seja maior. Ainda segundo Ogata (1994), em um sistema de controle digital, uma regra prática é amostrar o sinal de 8 a 10 vezes em um período da frequência amortecida do sinal de saída do sistema em malha fechada. Será abordado mais à frente a resposta desejada ao sistema em malha fechada, mas é possível adiantar que o período da frequência amortecida da resposta é ligeiramente menor que 6 segundos, com isto é necessário um período de amostragem de no máximo 0,6 segundos. Como foi visto anteriormente, um período de amostragem muito curto leva à uma discretização grande na leitura da rotação. Foi escolhido então um período de amostragem de 100ms. Este período de amostragem escolhido não prejudica a estabilidade do sistema em malha fechada e a discretização da leitura da velocidade de rotação não foi significativa para o projeto.

Para se chegar ao modelo discreto da planta, foi, então, aplicado a transformada z em sua função de transferência, utilizando o período de amostragem escolhido. Obteve-se como resultado a Equação 26.

$$G(z) = \frac{C(z)}{R(z)} = \frac{-0,1409z - 0,1395}{z^2 - 1,972z + 0,9718} \tag{26}$$

4.3.3 Equação de diferenças

Com o modelo no plano z da planta é possível montar equações de diferenças que descrevem seu comportamento. Primeiramente foram separados os termos que multiplicam a saída, C(z), e os termos que multiplicam a entrada, R(z). chegando à Equação 27.

$$C(z)(z^2 - 1,972z + 0,9718) = (-0,1409z - 0,1395)R(z) \tag{27}$$

Multiplicou-se os dois lados da equação por z^{-2} e rearranjou-se os termos para chegar à Equação 28.

$$C(z) = -0,1409R(z)z^{-1} - 0,1395R(z)z^{-2} + 1,972C(z)z^{-1} - 0,9718C(z)z^{-2} \tag{28}$$

Utilizou-se da propriedade de deslocamento no tempo da transformada Z, Equação 29, para se chegar a equação de diferença que representa o comportamento do sistema, Equação 30.

$$Z\{x[n - n_0]\} = z^{-n_0}X(z) \tag{29}$$

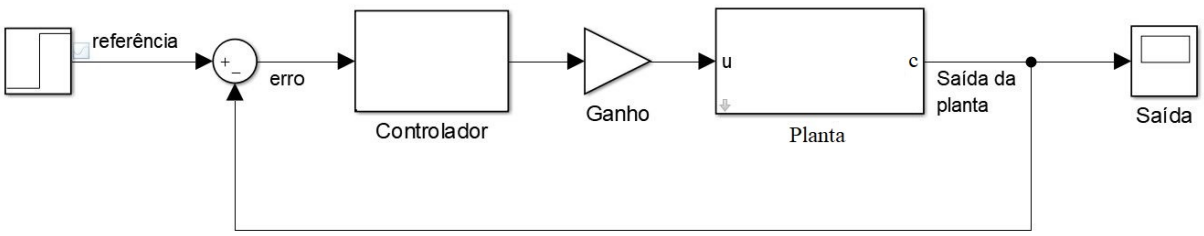
$$c(n) = -0,1409r(n-1)) - 0,1395r(n-2)) + 1,972c(n-1)) - 0,9718r(n-2)) \tag{30}$$

Foi obtido com este processo uma equação de diferenças que relaciona a rotação atual com a rotação em tempos anteriores e o sinal de entrada da planta. Esta equação de diferenças foi utilizada para modelar o comportamento da planta durante as simulações no processo de projeto do sistema de controle, feita no Matlab.

4.4 Sistema de Controle

A modelagem da planta foi feita com um degrau aplicado na entrada do estágio de potência, então, as características dinâmicas e de ganho do estágio de potência estão inclusas neste modelo matemático, porém, como foi visto anteriormente, é possível alterar o ganho do estágio de potência manualmente, modificando com isto a planta. Para se evitar a necessidade de projetar diferentes estágios de controle para diferentes ganhos ajustados no estágio de potência, foi adicionado um estágio de ganho, entre o estágio de controle e a planta, a fim de compensar o ganho do estágio de potência. O valor do ganho do estágio de potência é informado pelo usuário, utilizando a interface gráfica, assim o sistema faz a compensação do valor desse estágio de ganho, mantendo a mesma planta, independentemente do valor ajustado no estágio de potência, sob a ótica do estágio de controle e considerando que não haja saturação no estágio de potência. A Figura 24 mostra como ficou o sistema completo após a adição deste novo estágio de ganho.

Figura 24 – Sistema de controle em malha fechada



Fonte: Autoria própria (2019).

4.4.1 Requisitos

O sistema de controle utilizado neste trabalho deve seguir algumas especificações quanto à resposta do sistema. Estes requisitos do sistema de controle visam garantir a segurança e a qualidade dos testes feitos. As especificações são:

a) Estabilidade: visa garantir que nas condições de uso normais o sistema tenderá ao valor de rotação definido pelo usuário, não ocorrendo a parada do motor e nem uma disparada desenfreada do valor de sua rotação.

b) Erro nulo em regime permanente: visa garantir que os testes sejam feitos com seus valores de rotação constantes.

c) Tempo de acomodação de até 30 segundos: visa garantir que após o comando do usuário o sinal de rotação fique estável e pronto para o teste em no máximo 30 segundos.

d) Sobressinal máximo de 10%: visa garantir que, durante a transição de valores de referência, o valor de rotação não tenha uma grande diferença em comparação com o valor de referência indicado pelo usuário.

4.4.2 O Projeto

Nesta subsseção é descrito o tipo de sistema de controle utilizado, suas características e os passos percorridos para seu projeto, esses passos podem também ser conferidos no *Script* do Matlab utilizado, que está no Apêndice B.

"O controlador PID é aplicado em sistemas de controle em que é necessária a melhora tanto da resposta transitória quanto da resposta em regime estacionário."(PHILLIPS; HARBOR, 1997). Foi escolhido, então, um controlador PID para o estágio de controle. Ainda segundo Phillips e Harbor (1997), a função de transferência de um controlador PID pode ser descrita como Equação 31

$$G_c(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{31}$$

Nesta equação $G_c(s)$ representa a função de transferência do controlador, E(s) é o erro, U(s) é o sinal de controle aplicado na planta, K_p é o ganho proporcional ao erro, K_d é o ganho proporcional à derivada do erro e K_i é o ganho proporcional à integral do erro.

Para o cálculo dos ganhos envolvidos no estágio de controle foi utilizado o método analítico, apresentado em Phillips e Harbor (1997). Neste método os valores de sobressinal máximo e tempo de acomodação, definidos nos requisitos do projeto, são utilizados para definir os novos valores de ζ e ω_n que a planta com o sistema de controle deve possuir. Em Ogata (2010) é demonstrada a relação do sobressinal máximo e o valor de ζ, Equação 32. ζ foi isolado nesta equação para se obter a Equação 33.

$$M_p = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} \tag{32}$$

$$\zeta = \frac{\ln{(M_p)^2}}{\sqrt{\pi^2 + \ln{(M_p)^2}}} \tag{33}$$

Para o cálculo do valor ω_n foi utilizado a Equação 34, que o relaciona com a constante de tempo, τ, e o fator de amortecimento, ζ.

$$\omega_n = \frac{1}{\tau\zeta} \tag{34}$$

Considerando-se $\zeta < 1$, um sistema sub-amortecido, os polos de um sistema de segunda ordem são:

$$s_{1,2} = -\zeta\omega_n \pm j(\omega_n\sqrt{1-\zeta^2}) \tag{35}$$

Foram definidos os ângulos β e ψ com as relações de Equação 36 e Equação 37.

$$s_1 = |s_1|e^{j\beta} \tag{36}$$

$$G(s_1) = |G(s_1)|e^{j\psi} \tag{37}$$

Para o cálculo dos ganhos do controlador, que fazem com que se tenha os polos desejados no sistema completo, Phillips e Harbor (1997) apresentam as equações, Equação 38 e Equação 39,

$$K_p = \frac{-sen(\beta + \psi)}{|G_p(s_1)H(s_1)|sen\beta} - \frac{2K_i \cos\beta}{|s_1|} \tag{38}$$

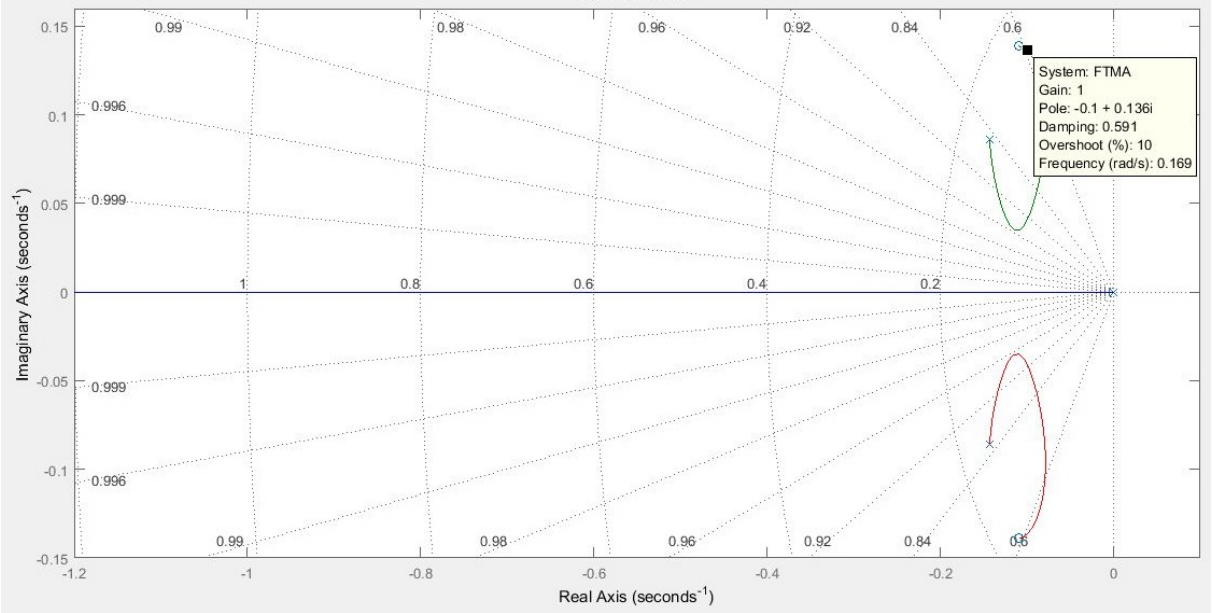
$$K_d = \frac{sen\psi}{|s_1||G_p(s_1)H(s_1)|sen\beta} + \frac{K_i}{|s_1|^2} \tag{39}$$

Como se tem 3 ganhos para determinar e somente duas equações, deve-se arbitrar um dos ganhos. Em todas as análises deste trabalho o ganho escolhido a ser arbitrado foi o K_i.

Com os requisitos do projeto chegou-se a um valor de ζ de 0,591 e ω_n 0,169, que levam o sistema a possuir os polos, $s_{1,2}$ = 0,1 $\pm$ 0,1364i.

Com este método de projeto, foi obtido como resultado um sistema que possui 3 polos, sendo o polos complexos conjugados desejados, $s_{1,2}$ e um terceiro que depende do valor arbitrado ao ganho K_i. O sistema possui, ainda, 2 zeros que também dependem do K_i arbitrado. A Figura 25 mostra o lugar das raízes do sistema projetado utilizando -0,001 como valor para o ganho K_i, é destacado que os polos desejados fazem parte desse lugar das raízes.

Figura 25 – Lugar das raízes do sistema completo



Fonte: Autoria própria (2019).

Utilizando os polos $s_{1,2}$, calculados para se obter os requisitos do projeto, foi montada a Tabela 3, que demonstra alguns exemplos de possíveis valores para o ganho K_i, os ganhos K_d e K_p calculados e os polos e zeros resultantes do sistema completo.

Tabela 3 – Possíveis sistemas de controle

K_i	K_d	K_p	Polos	Zeros
-0,0001	-4,6309.10^{-04}	-7,2446.10^{-04}	-0,1 $\pm$ 0.1364i -0.0994	-1.4114 -0.1530
-0,001	-0,032	-0,007	-0,1 $\pm$ 0,1364i -0,9941	-0,1099 $\pm$ 0,1388i
-0,01	-0,3464	-0,0699	-0,1 $\pm$ 0,1364i -9,9411	-0,1009 $\pm$ 0,1367i
-0,1	-3,4916	-0,6990	-0,1 $\pm$ 0,1364i -99,4113	-0,1001 $\pm$ 0,1365i

Fonte: Autoria própria (2019).

É possível verificar, com esta tabela, que seguindo esta metodologia de projeto sempre foi obtido como resultado um sistema que possui os polos $s_{1,2}$, que levam a resposta exatamente aos requisitos do projeto. Além do sistema possuir esse polos, ainda é preciso garantir que esses polos sejam dominantes no sistema, para que a sua resposta possa ser aproximada da resposta de um sistema de segunda ordem e se chegar ao resultado esperado. Segundo Dorf e Bishop (2013), esta dominância dos polos em relação ao terceiro polo ocorre "desde que a parte real das

raízes dominantes seja menor que um décimo da parte real da terceira raiz", ou seja, a parte real do terceiro polo que surge deve ser maior em módulo que 10 vezes a parte real dos polos desejados, $s_{1,2}$. Levando esta análise em consideração foi reavaliado a Tabela 3, e verificado se os projetos ali descritos conseguem satisfazer esta característica necessária para que a dominância dos polos ocorra. Foi verificado, então, que neste projeto, o valor de K_i deverá ser menor que -0,001. Este valor é praticamente o limiar da dominância dos polos, onde a parte real do terceiro polo existente no sistema é 9,941 vezes a parte real dos polos, $s_{1,2}$.

4.4.3 Implementação

Para a implementação do sistema de controle, foi necessário primeiro discretizar o sistema de controle com a frequência de amostragem utilizada nesta pesquisa, 10 Hz. Para isso foi aplicada a transformada Z na Equação 31, tendo como resultado a Equação 40.

$$Gc(z) = \frac{U(z)}{E(z)} = Kp + 10K_i\frac{z^{-1}}{1-z^{-1}} + \frac{Kd}{10}\frac{1-z^{-1}}{z^{-1}} \tag{40}$$

Foi definido U_p, U_d e U_i como:

$$U_p(z) = KpE(z) \tag{41}$$

$$U_d(z) = \frac{K_d}{10}\frac{1-z^{-1}}{z^{-1}}E(z) \tag{42}$$

$$U_i(z) = 10K_i\frac{z^{-1}}{1-z^{-1}}E(z) \tag{43}$$

U(z) pôde, então, ser reescrita como demonstrado na Equação 44.

$$U(z) = U_p(z) + U_d(z) + U_i(z) \tag{44}$$

Utilizou-se da propriedade de deslocamento no tempo da transformada Z, Equação 29, para se chegar a:

$$u_p(n) = K_p e(n) \tag{45}$$

$$u_d(n) = \frac{Kd}{10}(e(n) - e(n-1)) \tag{46}$$

$$u_i(n) = 10K_i e(n-1) + u_i(n-1) \tag{47}$$

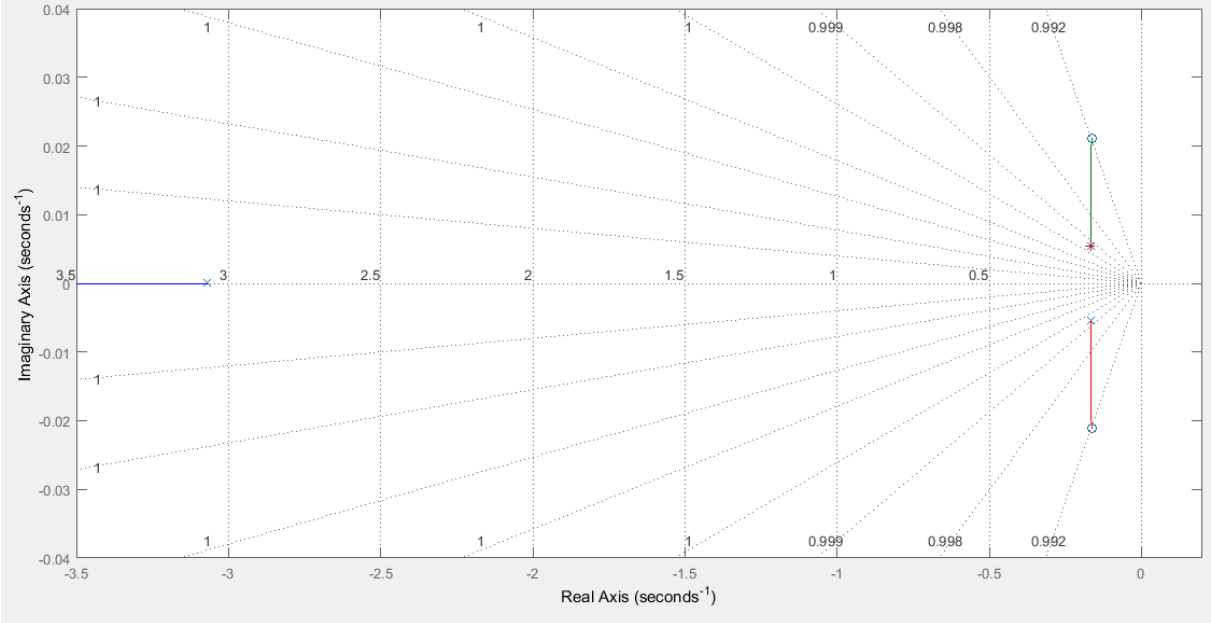
Foi obtido com este processo uma equação de diferenças que implementa o sistema de controle calculado, Equação 48.

$$u(n) = u_p(n) + u_d(n) + u_i(n) \tag{48}$$

Para um primeiro teste do projeto do sistema de controle, foi criado um projeto utilizando como requisitos um tempo de acomodação, de 5%, de 18 segundos e um sobressinal máximo nulo. Foi arbitrado um K_i de -0,003, chegando-se a um K_d de -0,1095 e um K_p -0,036.

A Figura 26 mostra os polos e zeros do sistema completo.

Figura 26 – Polos e zeros do primeiro teste de controle



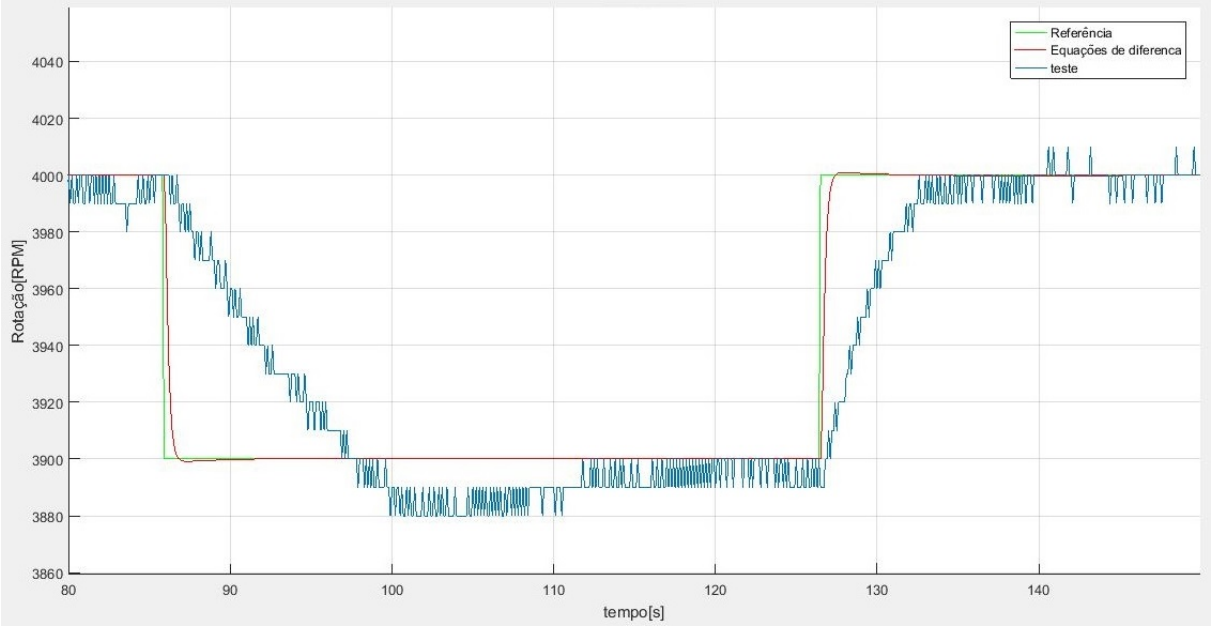
Fonte: Autoria própria (2019).

Os polos calculados que levam o sistema aos requisitos são, 0,1667 $\pm$ 0,0054i, é visto que eles aparecem polos do sistema. A parte real do terceiro polo que surge é mais de 10 vezes a parte real dos polos $s_{1,2}$, garantindo a dominância por parte dos polos complexos conjugados.

Utilizando este sistema de controle foi feito um teste com o sistema real e comparado com o resultado obtido em simulação. Neste teste o ganho do estágio de potência foi ajustado para que ele forneça ao freio no máximo 30 V. O método utilizado para a leitura de rotação foi o método de contagem de pulsos e como a frequência de amostragem foi de 10 Hz, foi obtida uma resolução de 10 RPM. Foram aplicados dois degraus à referência, um de 4000 para 3900 RPM e o outro de 3900 para 4000 RPM. Estes valores foram escolhidos por estarem próximos aos valores de rotação em que foi levantado o modelo matemático.

Foi utilizado para comparação entre os resultados, simulado e testado, o sobressinal máximo, M_p, e o tempo de atraso, t_d. A Figura 27 mostra o resultado simulado e o resultado encontrado na prática.

Figura 27 – Primeiro teste do projeto de controle

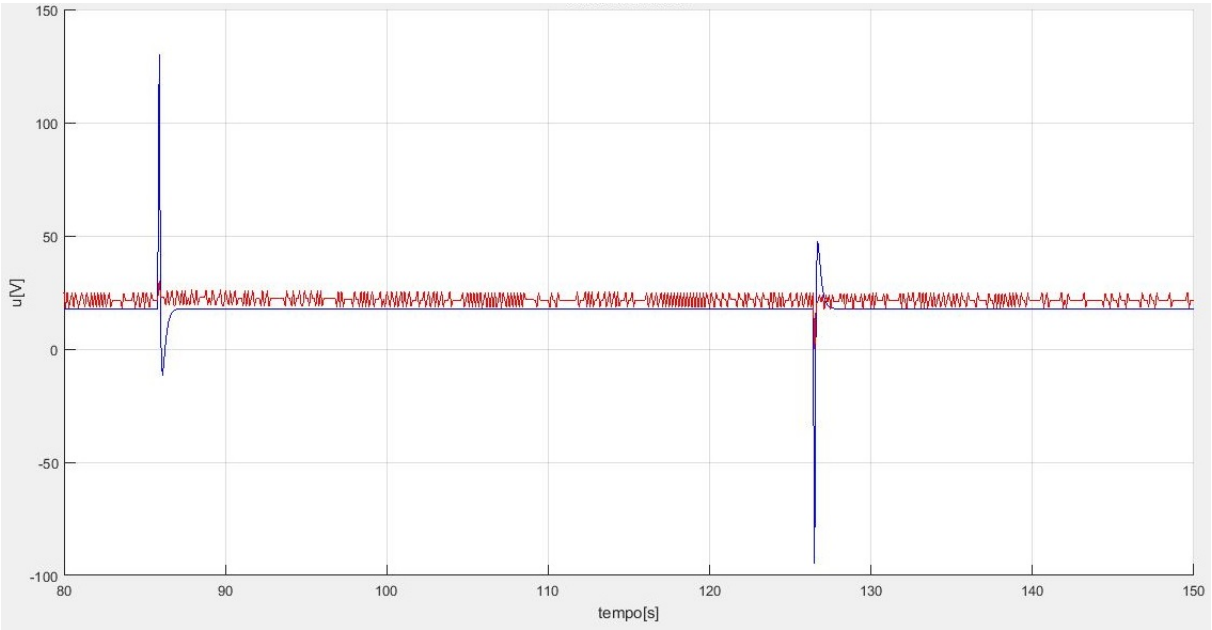


Fonte: Autoria própria (2019).

O sistema simulado teve um sobressinal máximo de 2% nos dois degraus, e um tempo de atraso de 300 ms. No teste real o sistema teve um valor de sobressinal máximo de 20% no primeiro degrau e de 10% no segundo e um tempo de atraso de 4 segundos.

Para entender melhor por que o resultado do teste foi diferente do simulado, foi analisado o sinal de controle que foi aplicado à planta, Figura 28.

Figura 28 – Sinal de controle do primeiro teste do projeto de controle



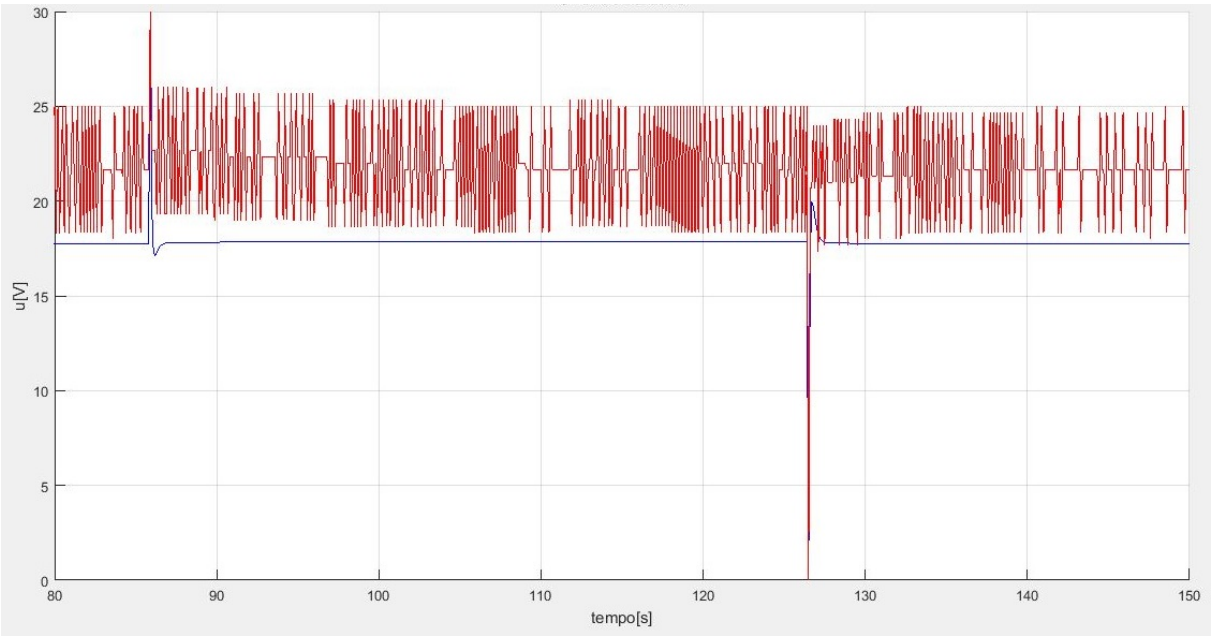
Fonte: Autoria própria (2019).

É possível verificar que o sinal de controle teve grandes picos nos momentos de mudança do valor de referência, na simulação ele chega à pouco mais de 130 V na parte positiva e a menos que -90 V na parte negativa. No teste real ele é limitado aos valores que o sistema de potência foi configurado à fornecer, 0 a 30 V.

Para avaliar se a grande diferença entre a simulação e o teste prático foi causado pela saturação do sinal de controle, foi feita uma nova simulação onde desta vez o sinal de atuação foi limitado aos valores reais, esta nova simulação foi comparada ao teste.

Na Figura 29 é apresentado o sinal de controle nesta simulação com saturação, comparando com o sinal de atuação do teste. Os picos do sinal de atuação agora são limitados entre 0 e 30 V também na simulação.

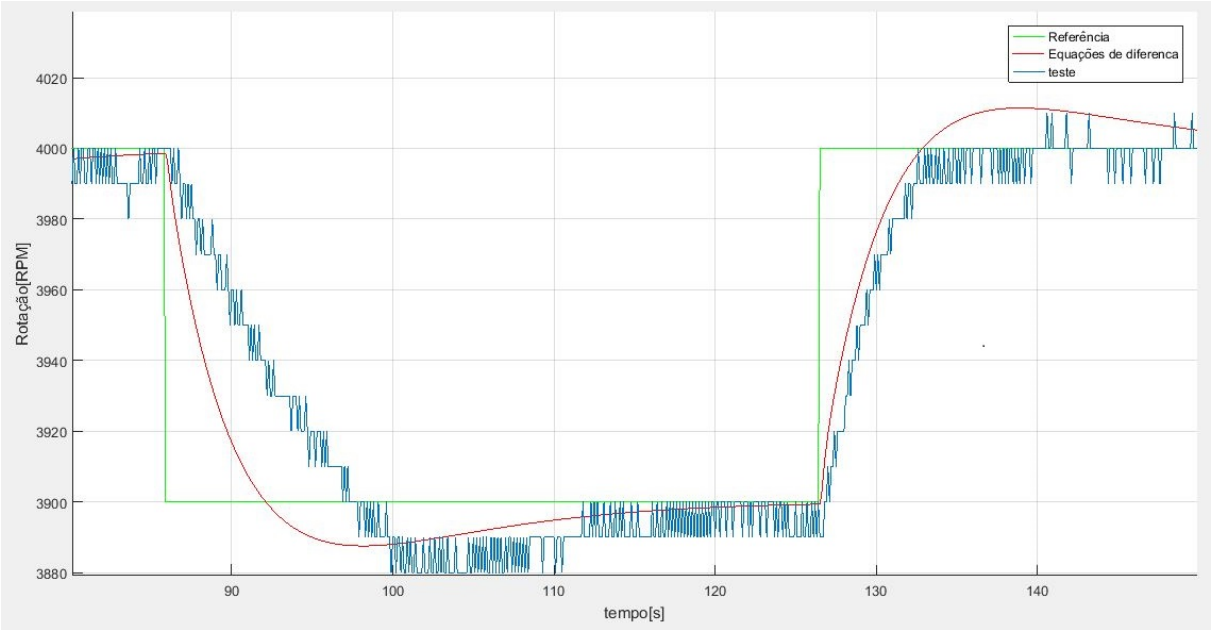
Figura 29 – Sinal de controle do primeiro teste do projeto de controle com saturação



Fonte: Autoria própria (2019).

Na Figura 30 é visto que os resultados obtidos na simulação com saturação ficaram mais próximos aos resultados obtidos no teste. Tempo de acomodação dessa simulação foi em média de 25,2 segundos e o sobressinal máximo foi em média de 12%.

Figura 30 – Primeiro teste do projeto de controle com saturação



Fonte: Autoria própria (2019).

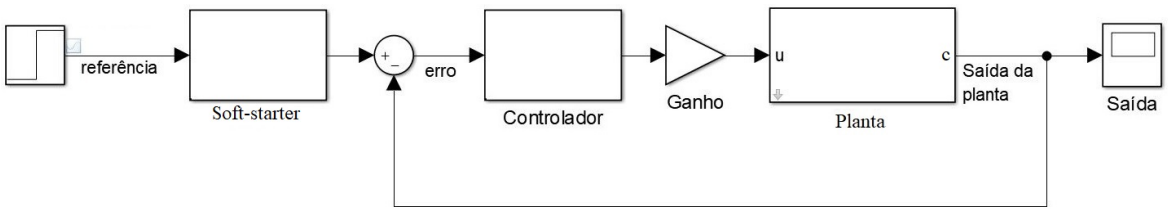
Estas comparações mostraram que a saturação, no sinal de atuação, causa

uma grande diferença nos resultados do sistema controlado, então, é necessário a busca por soluções que evitem essa saturação ou que minimizem os efeitos criados por ela.

4.5 *Soft-start*

Como o tempo de acomodação do sistema é relativamente alto, e uma variação brusca de sua velocidade de rotação causaria um estresse desnecessário às partes mecânicas, não é necessário que a resposta seja ao degrau. Tendo isto em vista, uma possível solução testada para o minimizar o problema da saturação foi adicionar um estágio no sinal de referência, antes do cálculo do erro. Esse estágio foi utilizado para limitar a variação da referência utilizada para o calculo do erro, evitando assim que grandes variações venham a gerar grandes picos ou a saturação do sinal de controle. O sistema completo ficou como apresentado na Figura 31.

Figura 31 – Sistema de controle com limitador na referência



Fonte: Autoria própria (2019).

Para o teste dessa proposta de solução, foi projetado um sistema de controle que leve a convergências rápidas da resposta da planta e que tenha um pequeno valor de sobre-sinal máximo. Normalmente esse sistema de controle leva a saturação do sinal de controle, quando é aplicado um degrau no sinal de referência, porém essa variação do valor de referência foi limitada e para se avaliar se esta técnica evita a saturação do sinal e diminui o valor de sobressinal máximo.

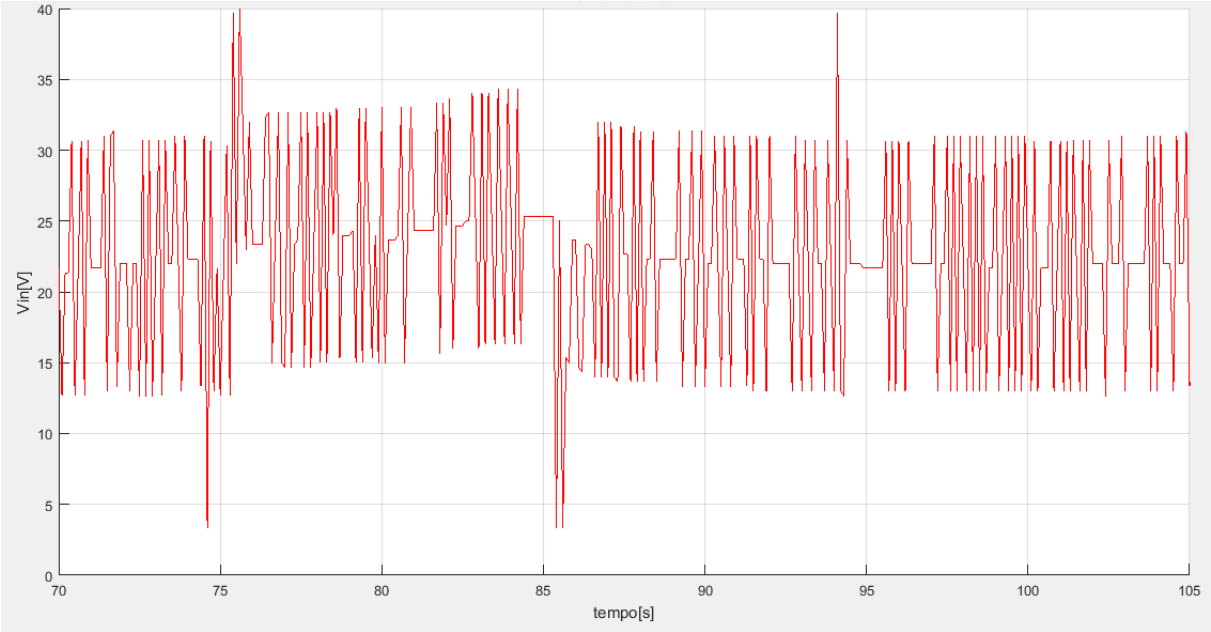
O sistema de controle projetado buscou um tempo de acomodação de 18 segundos e um sobressinal máximo de 1,5%. Os polos $s_{1,2}$ de 0,1666 $\pm$ 0.1253i que levam a esta resposta. Os ganhos ficaram, Kd = -0.1172, Kp = -0.03891 e Ki = -0.005 e a variação máxima configurada foi de 10 RPM por amostra, com a frequência de amostragem de 10 HZ, se tem uma variação máxima de 100 RPM/s. Neste teste foi ajustado o ganho do estágio de potência para que ele forneça ao freio no máximo 40 V. O teste utilizou o método de contagem de pulsos para a leitura da rotação ficando com uma resolução de 10 RPM.

A velocidade de rotação e a referência foram mantidos em 4000 RPM e foi

aplicado um degrau na referência de -1000 RPM e após a estabilização foi aplicado mais um degrau de 1000 RPM, retornando para os 4000 RPM inciais.

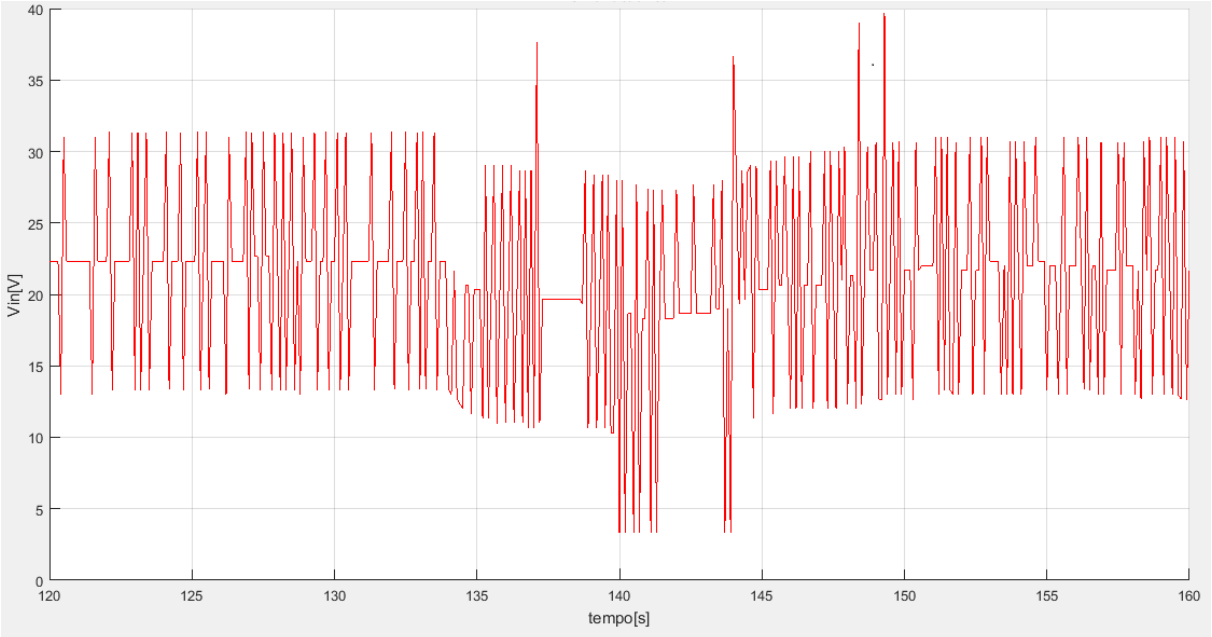
Na Figura 32 é apresentado o sinal de controle durante a resposta ao degrau negativo e na Figura 33 mostra o sinal de controle durante a resposta ao degrau positivo, é possível ver em ambos que, mesmo com a variação do sinal de referência, em nenhum momento o sinal de controle chegou aos valores de limitação, 0 a 40 V.

Figura 32 – Sinal de controle do primeiro teste da referência com limitador de variação



Fonte: Autoria própria (2019).

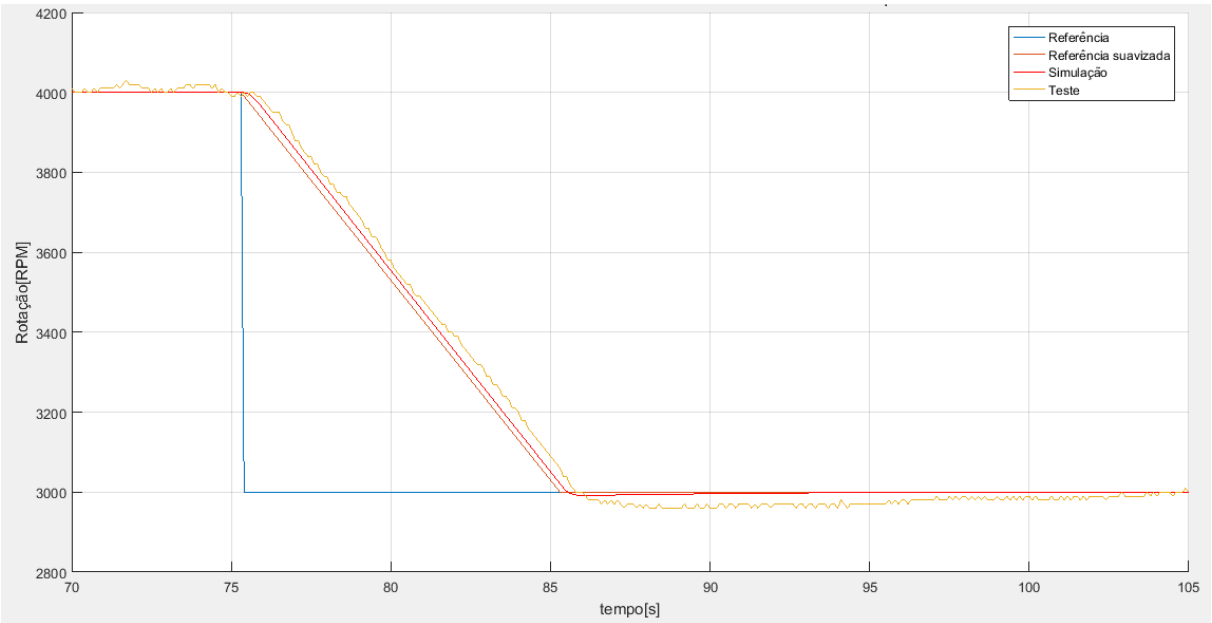
Figura 33 – Sinal de controle do primeiro teste da referência com limitador de variação



Fonte: Autoria própria (2019).

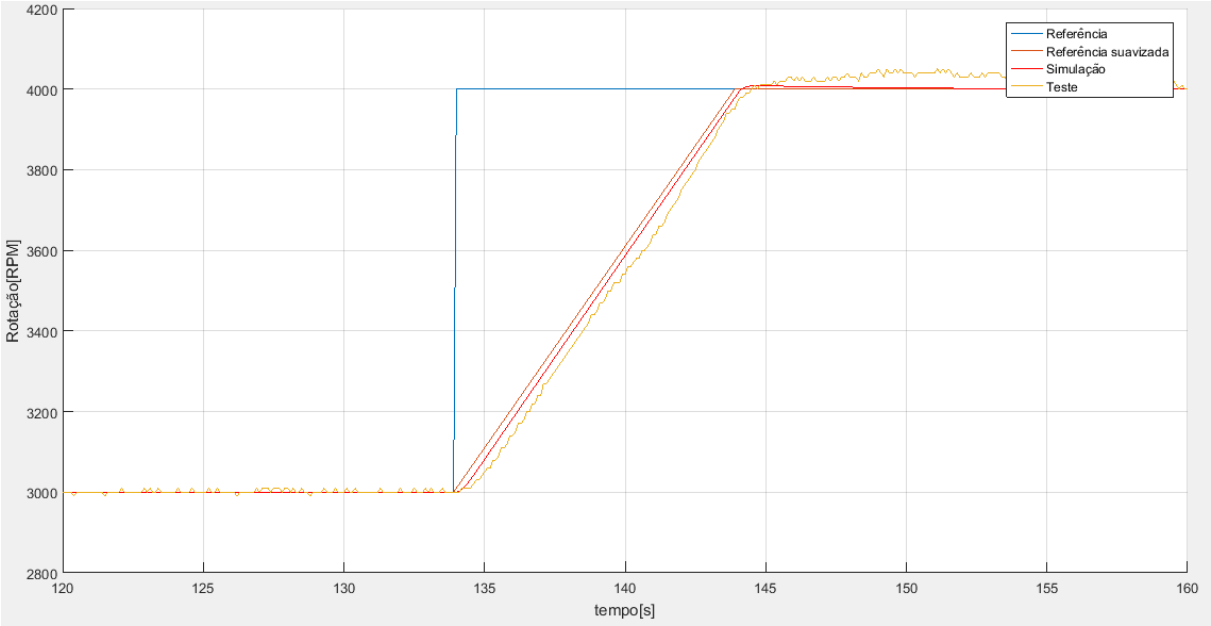
A Figura 34 mostra a velocidade de rotação durante a resposta ao primeiro degrau aplicado e na Figura 35 é possível ver a velocidade de rotação durante o segundo. Em ambos a velocidade de rotação variou de forma quase que constante durante as mudanças de referência, tendo um atraso em relação a referência, em torno de 700 ms. Os valores de sobressinal máximo ficaram de até 5%, ou 50 RPM.

Figura 34 – Primeiro teste da referência com limitador de variação



Fonte: Autoria própria (2019).

Figura 35 – Primeiro teste da referência com limitador de variação



Fonte: Autoria própria (2019).

Embora em simulação o tempo de acomodação para grandes degraus seja sempre menor do que o tempo que a referência leva para chegar ao valor solicitado, Equação 49, foi observado nos teste que há oscilações do sinal antes da estabilização do sinal de velocidade de rotação, o que fez com que em alguns casos este tempo de acomodação fosse maior que o simulado.

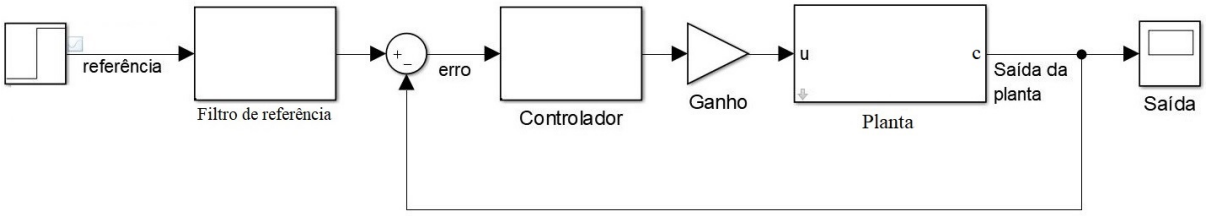
$$t_s < \frac{\Delta L}{\Delta R} f_s \tag{49}$$

Onde ΔL representa a máxima variação permitida entre as medidas, f_s a frequência de amostragem e ΔR é a amplitude do degrau da referência.

4.6 Filtro de Referência

Foi visto anteriormente que esse projeto de controle PID nos leva a um sistema em malha fechada que possui os polos desejados ao sistema, e além de um terceiro polo e mais dois zeros. A influência do terceiro polo foi contornada garantindo a dominância dos polos desejados, mas os zeros que surgem também alteram a resposta do sistema e para compensar este fato foi adicionado ao sistema um filtro no sinal de referência (Figura 36).

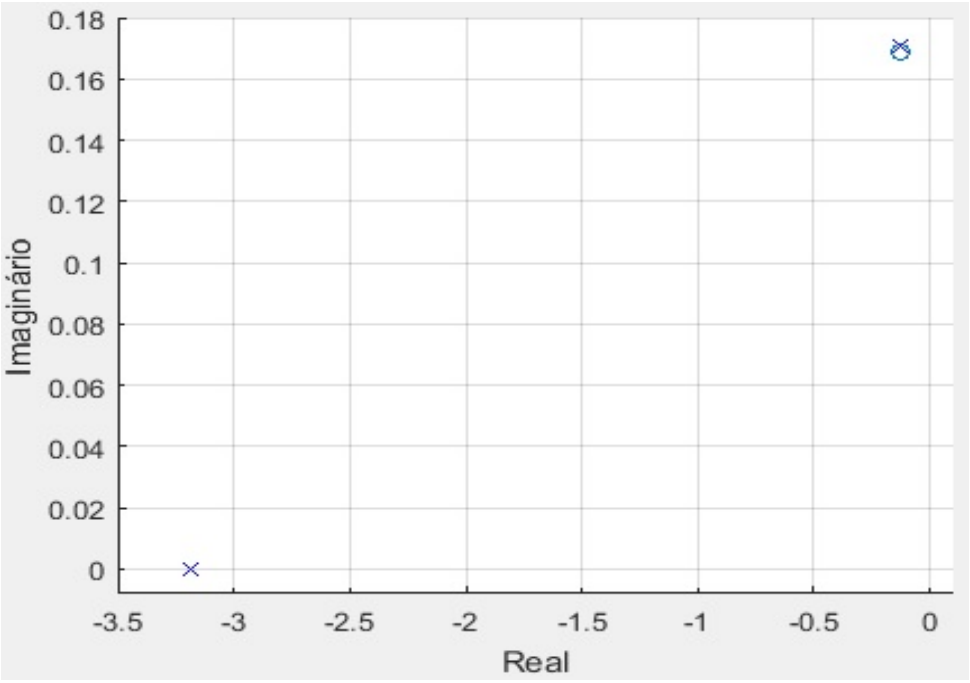
Figura 36 – Sistema de controle com filtro de referência



Fonte: Autoria própria (2019).

Segundo Ogata (2010), o filtro de referência pode ser utilizado para modificar os zeros do sistema em malha fechada. Estes zeros podem tanto ser cancelados como substituidos, e têm como ideia básica evitar sinais de controle elevados e como ele está fora da malha fechada, não altera a estabilidade dessa parte do sistema. Neste trabalho os zeros do sistema em malha fechada foram cancelados, e para isto o filtro adicionado necessita ter dois polos exatamente nos pontos onde se tem os zeros do sistema em malha fechada. Com a adição deste filtro e o cancelamento dos zeros mudou-se a dinâmica da resposta do sistema, se aproximando da resposta de um sistema de segunda ordem com os polos dominantes. A Figura 37 demonstra os polos e zeros de um sistema completo, sem o filtro de referência, que foi projetado para se ter um tempo de acomodação de 24 segundos, um sobressinal máximo de 10% e utilizando -0,005 para valor do ganho K_i, chegando-se a um K_d de -0,1106 e um K_p -0,0285.

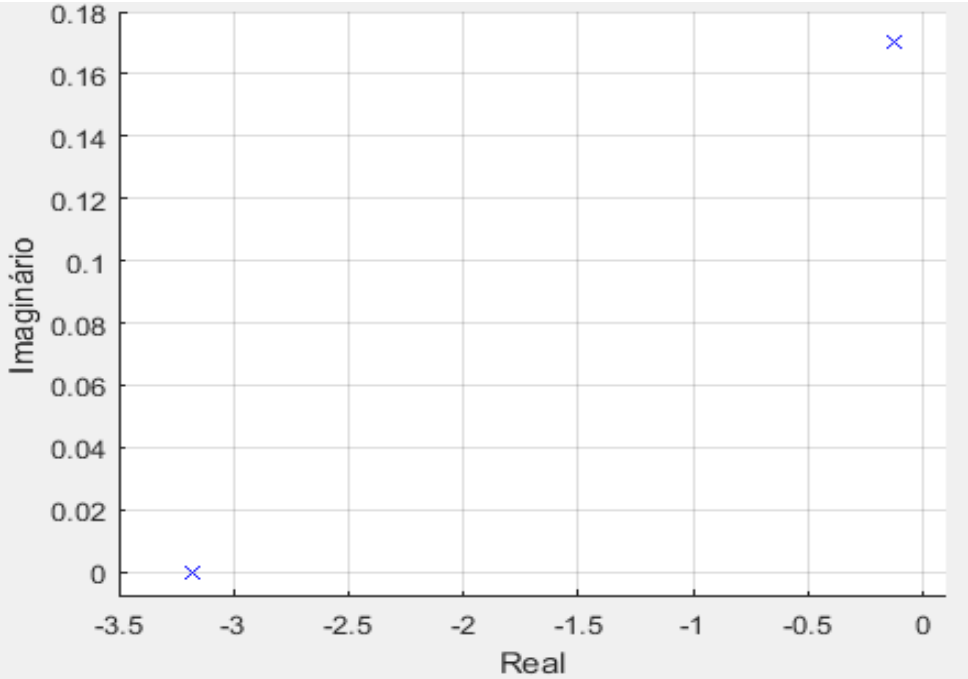
Figura 37 – Polos e zeros do sistema sem o filtro de referência



Fonte: Autoria própria (2019).

É possível observar que os zeros que surgiram decorrentes do sistema de controle ficaram muito próximos aos polos que nos levam a resposta desejada. A Figura 38 demonstra o lugar das raízes com o sistema utilizando o filtro de referência.

Figura 38 – Polos e zeros do sistema com filtro de referência



Fonte: Autoria própria (2019).

Como visto anteriormente, a adição dos novos polos, do filtro de referência, anulou o efeito causado pelos zeros do sistema em malha fechada e isto fez com que mudasse a dinâmica da resposta.

Trabalhando a Equação 31 e reescrevendo-a como na Equação 50 é mais fácil visualizar os zeros do sistema de controle, que têm o mesmo valor dos zeros do sistema em malha fechada com sistema de controle.

$$Gc(s) = \frac{K_d s^2 + Kps + K_i}{s} \tag{50}$$

O filtro de referência, então, pode ser definido como na Equação 51. Foi adicionado em seu numerador o ganho K_i, isto garante que o ganho estático desse estágio seja unitário.

$$G_f(s) = \frac{K_i}{K_d s^2 + Kps + K_i} \tag{51}$$

O polos desse filtro ficam posicionados no mesmo ponto dos zeros do sistema de controle, Equação 52.

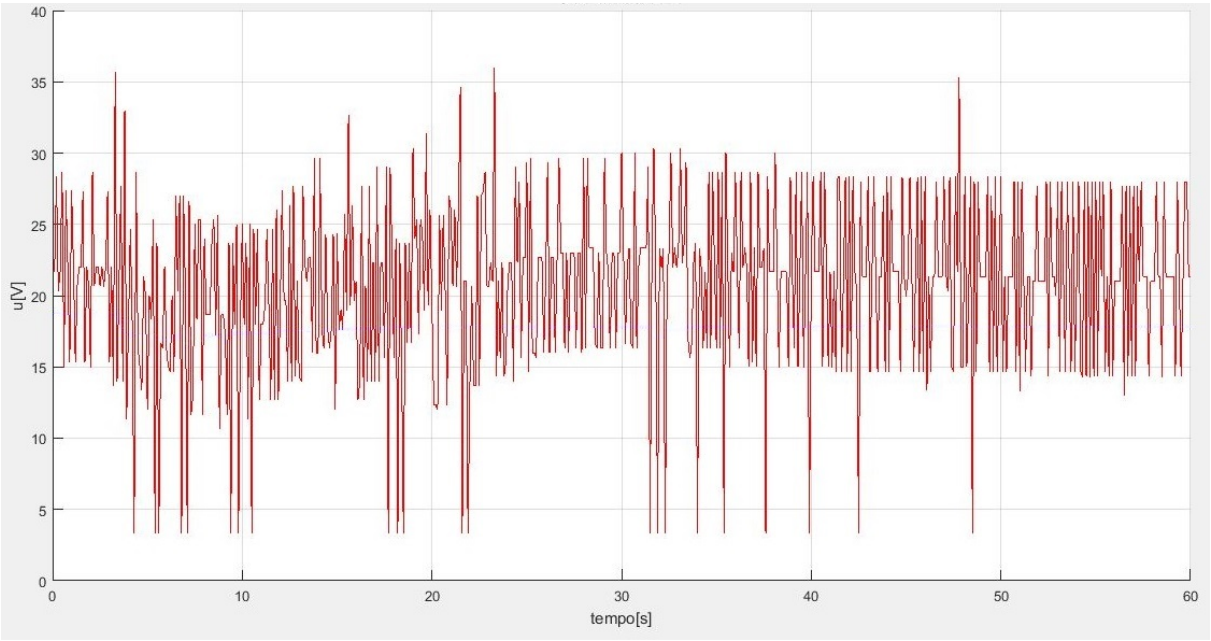
$$P_f = -\frac{Kp}{2K_d} \pm \frac{\sqrt{K_p^2 - 4K_dK_i}}{2K_d} \tag{52}$$

Este sistema projetado foi testado utilizando o filtro de referência, seu filtro discretizado ficou como apresentado na Equação 53,onde r representa o valor da referência antes do filtro, e r_f representa o sinal de referência após o filtro.

$$r_f(n) = 2,24.10^{-4}r(n-1) + 2,22.10^{-4}r(n-2) - 0,97r_f(n-1) + 1,97r_f(n-2) \tag{53}$$

Com esta configuração é esperado que, com o efeito dos zeros sendo cancelados, se tenha um sinal de controle sem saturação e um sinal de rotação próximo dos valores definidos no projeto. Neste teste, o estágio de potência foi ajustado para fornecer até 40 V e foi utilizado o método de contagem de bordas para a leitura da velocidade de rotação, nos dando uma resolução de 5 RPM. A Figura 39 mostra que não houve saturação no sinal de controle, durante o teste.

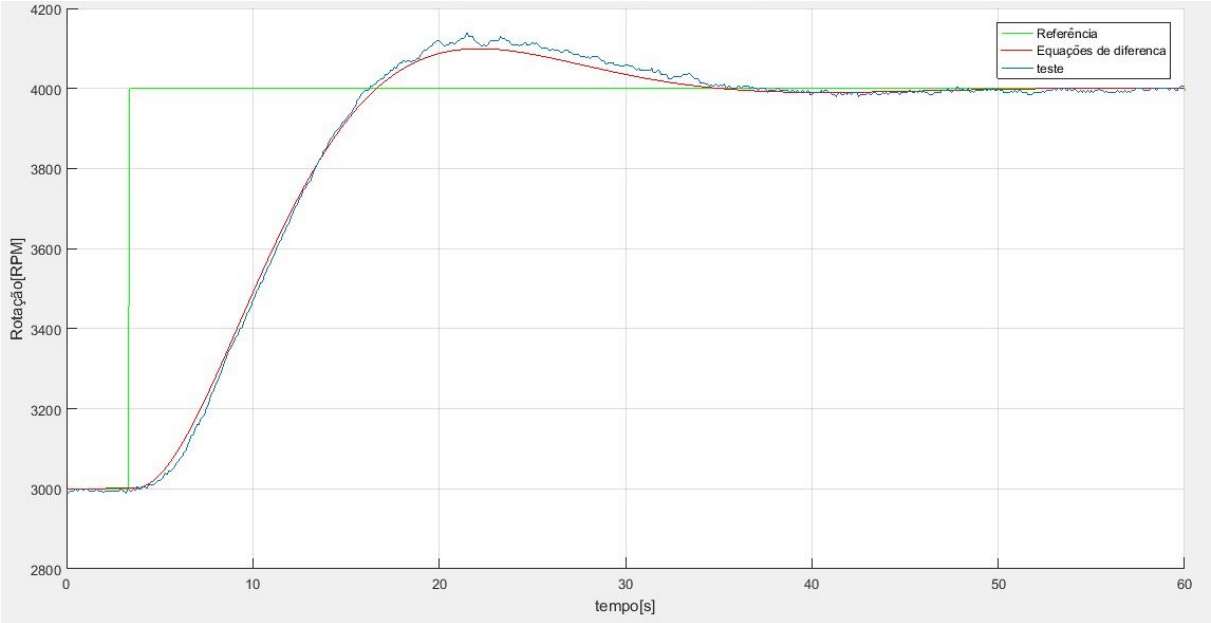
Figura 39 – Sinal de controle do primeiro teste com o filtro de referência



Fonte: Autoria própria (2019).

A Figura 40, mostra o sinal de rotação durante o teste e o compara com a simulação.

Figura 40 – Primeiro teste com o filtro de referência



Fonte: Autoria própria (2019).

No inicio do teste o sinal de referência se encontrava em 3000 RPM e o sinal de rotação se encontrava estabilizado também neste valor. Aos 3,4 segundos foi aplicado um degrau de 1000 RPM no sinal de referência. O sinal de rotação demorou 27,1 para se estabilizar e teve um pico de 4140 RPM, que nos dá um sobressinal máximo de 14%.

Em simulação o sistema demorou 25,2 segundos para se estabilizar, em 5%, e o pico de rotação foi de 4099 RPM, levando a um valor sobressinal máximo de 9,9%.

4.7 Microcontrolador

"Microcontroladores são circuitos integrados que possuem em seu interior todos os componentes necessário ao seu funcionamento dependendo unicamente da fonte de alimentação externa", (KERSCHBAUMER, 2018). Com esses componentes como conversores A/D, memória e módulos de comunicação, juntos no mesmo encapsulamento dos microprocessadores, os microcontroladores facilitam o desenvolvimento de dispositivos eletrônicos digitais e atualmente são amplamente utilizados já fazendo parte de praticamente todos os dispositivos eletrônicos digitais.

O microcontrolador utilizado nesse projeto foi o ATmega328p, segundo seu *datasheet*, ATmega328/P (2016), ele é um microcontrolador de 8 bits da família AVR, da fabricante Atmel, que pode trabalhar com tensão de alimentação variando entre 1,8V e 5,5V, possui três *timers*, seis canais de ADC de 10 *bits*, pode ser utilizado com *clock*

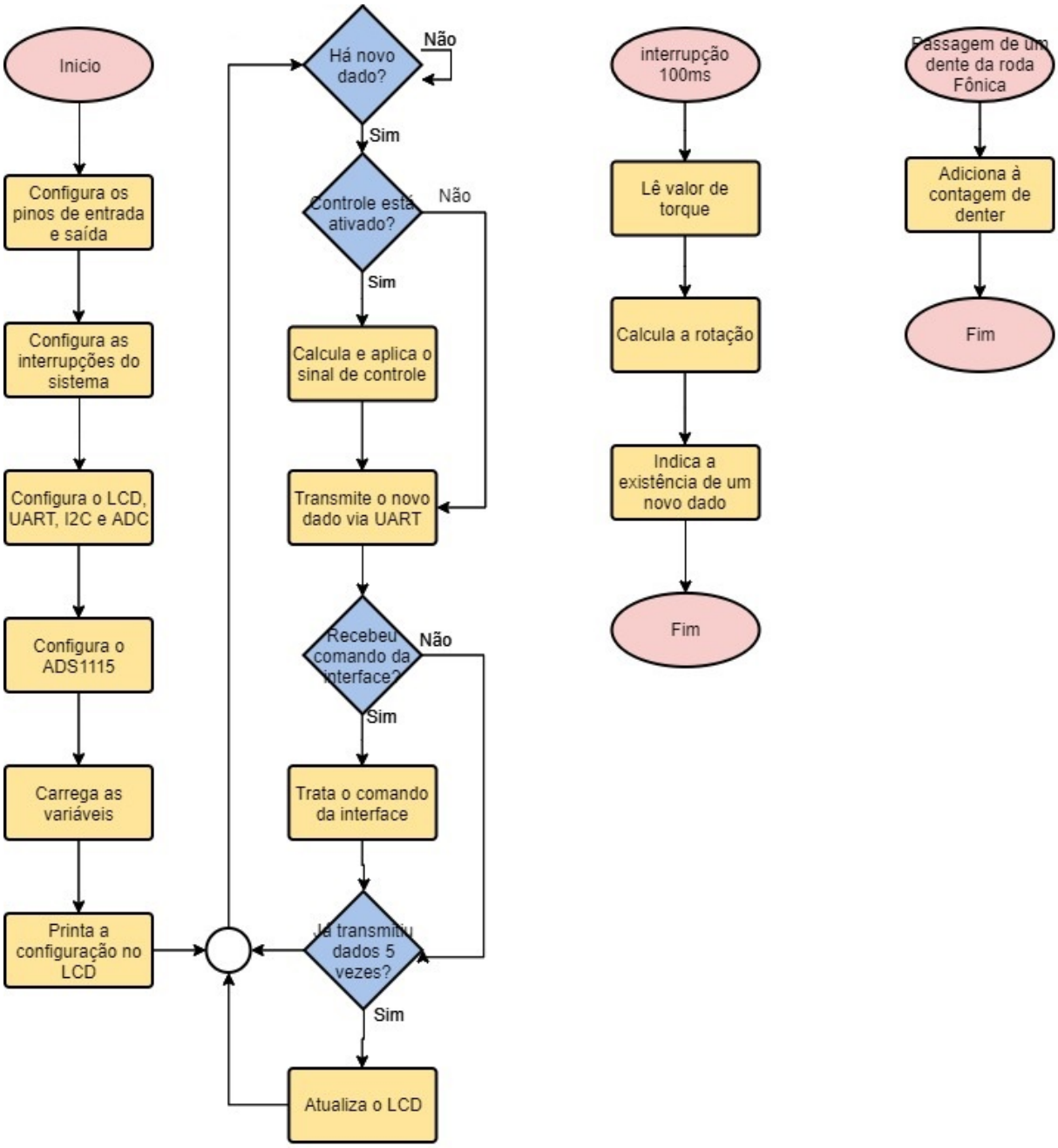
gerado por cristal externo com frequência de até 20 MHz à 5V de alimentação, possui comunicação USART programável além de módulo de comunicação serial a dois fios, que é compatível com o protocolo de comunicação I2C. A compatibilidade com esses protocolos de comunicação foi o principal motivo para a escolha deste microcontrolador, além de ele suprir as necessidades de velocidade de processamento do projeto.

4.7.1 Programação

A programação do microcontrolador foi feita em linguagem C , utilizando a IDE Atmel Studio 7.0. O código completo pode ser encontrado no Apêndice A, mas são tratados nesta subseção alguns destaques de sua implementação.

A Figura 41 mostra um fluxograma simplificado do funcionamento do código implementado no microcontrolador.

Figura 41 – Fluxograma da programacao do microcontrolador



Fonte: Autoria própria (2019).

Quando o microcontrolador é inicializado, a primeira tarefa que ele faz é configurar seus pinos de entradas e saídas, configura, após esta primeira etapa, as interrupções do sistema e os meios de comunicações, com o usuário e com a interface gráfica. Com os meios de comunicação configurados, é possível configurar o módulo ADS1115, lembrando que ele utiliza a comunicação I2C. Com as configurações todas prontas, já é possível carregar os valores iniciais às variáveis, demonstrar no LCD a configuração atual e, finalmente ,entrar no laço principal do programa.

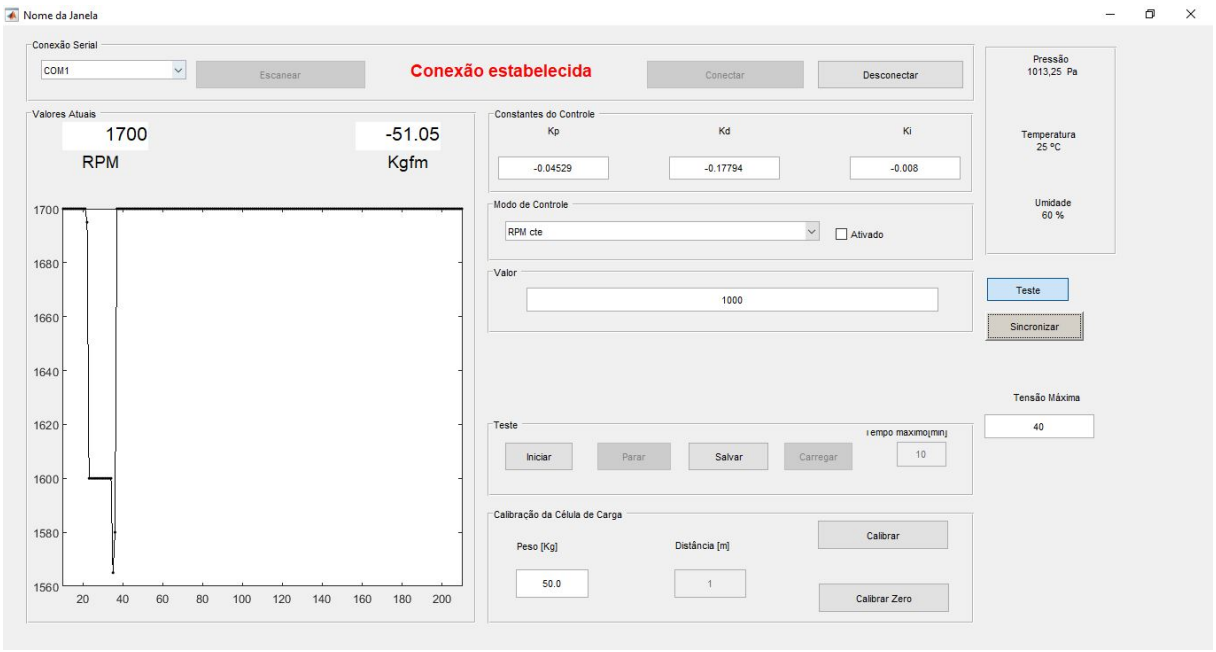
Neste laço principal, a primeira tarefa feita é verificar se há um dado novo a ser tratado. Em caso afirmativo, é verificado também se o sistema de controle está ativo. Com o sistema de controle ativo, é necessário calcular o sinal de controle e aplica-lo ao sistema, em caso contrario essa etapa não é executada. O dado que está disponível agora é transmitido para a interface gráfica através da comunicação UART. Neste ponto o dado novo do sistema já foi tratado, o microcontrolador verifica, então, se o sistema recebeu algum comando da interface gráfica, se recebeu o comando é tratado. Quando o sistema já fez este processo de transmissão dos dados à interface 5 vezes, o LCD é atualizado com os dados atuais do modo de controle, velocidade de rotação e torque. Como os dados são coletados a cada 100 ms, que garantem os 10 Hz da frequência de amostragem, o LCD é atualizado a cada 500 ms, ou duas vezes a cada segundo.

Outras etapas importantes do sistema são executados fora do laço principal, essas etapas são acionadas por interrupções do sistema. Uma delas ocorre toda vez que o sistema detecta a passagem de um dente da roda fônica, ela é responsável por adicionar essa passagem a contagem do número de dentes. A outra etapa acionada por interrupção ocorre na frequência de amostragem, 10 Hz. Um contador interno do microcontrolador que é responsável por gerar esta interrupção do sistema no tempo correto. É nesta etapa que ocorre a leitura do valor do torque, o calculo do valor da velocidade de rotação e a indicação de que há um novo dado a ser tratado.

4.8 Interface gráfica

A interface gráfica, Figura 42, se comunica com o protótipo através de uma comunicação serial, é executada no ambiente do software MATLAB e foi desenvolvida utilizando a ferramenta Matlab GUI, que é uma ferramenta para desenvolvimento de interfaces gráficas que permitem interação com o usuário. Esta ferramenta implementa a interface utilizando orientação a objetos, então, cada item é executado de forma isolada, simulando uma execução paralela. Na Apêndice D mostra as funções que são executadas durante o uso da interface.

Figura 42 – Interface gráfica



Fonte: Autoria própria (2019).

A execução desta interface é feita utilizando o próprio Matlab e permite ao usuário executar diversas funções durante os testes, é possível destacar aqui algumas das principais como: Ligar e desligar o sistema de controle, alterar os valores dos parâmetros do sistema de controle, alterar o valor da referência, calibrar a célula de carga responsável para a leitura do torque, além de, claro, acompanhar os valores de rotação e torque durante o teste. O teste feito utilizando esta interface gera um arquivo que armazena as medidas feitas durante o teste, são armazenados os valores de rotação, torque, sinal aplicado no freio e o sinal de referência. Não foi implementado nenhum tipo de limitação para os valores dos parâmetros ou da referência, então o usuário precisa tomar cuidado com esses valores para evitar a instabilidade no sistema.

5 RESULTADOS

Este capítulo mostra o protótipo que foi desenvolvido durante esta pesquisa, os resultados obtidos e demonstra também alguns testes feitos com ele.

O esquemático completo do sistema está em Apêndice C, e para a simplificação do desenvolvimento do protótipo, ele foi dividido em 3 placas diferentes:

1 Placa de interface: Essa placa concentra toda a parte de interface com o usuário, como botões, display e comunicação serial que é usada para a comunicação com a interface gráfica.

2 Placa de condicionamento de sinais: Essa placa é responsável pela aquisição e adequação dos sinais provenientes dos sensores. Fica responsável também pela entrada de tensão de alimentação, que será feita com uma fonte de 12 Volts, comercial.

3 Placa de processamento dos sinais: Essa placa que faz todo o gerenciamento das informações vindas da interface com o usuário e da placa de condicionamento, nela que se encontra o microcontrolador que faz todos os cálculos necessários ao sistema de controle.

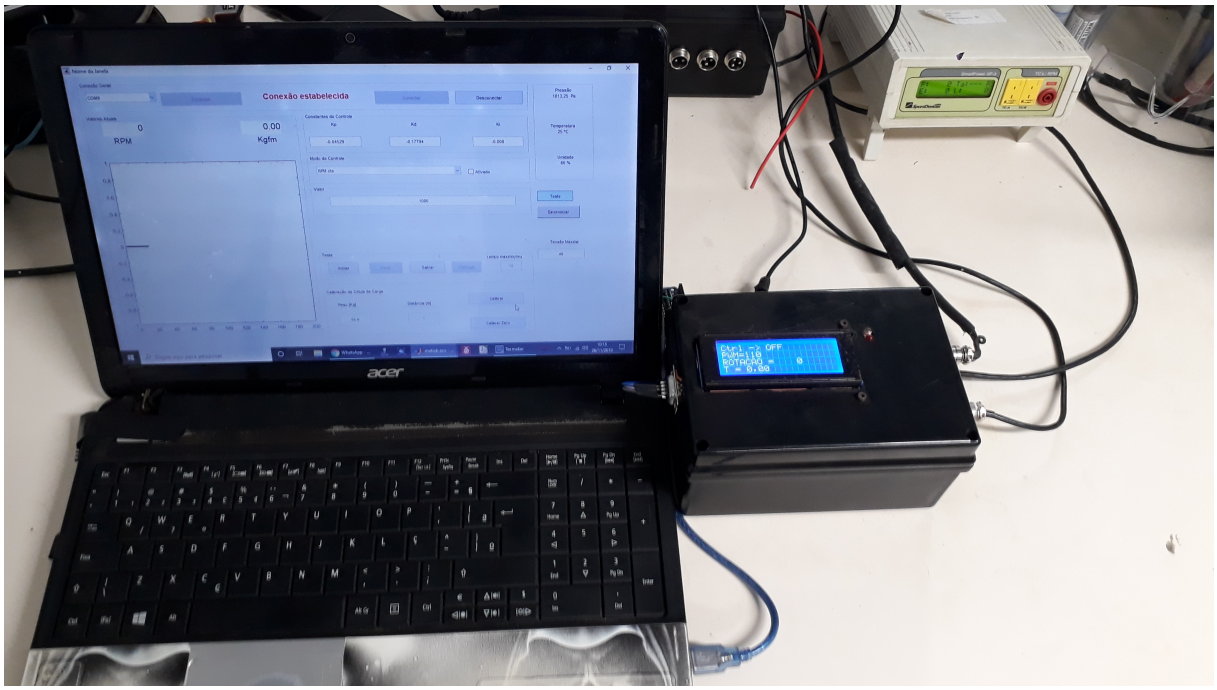
A Figura 43 mostra as placas desenvolvidas e como estas placas ficam posicionadas dentro do protótipo. A comunicação entre estas placas foi feita através de cabos *flat*, isso propiciou uma flexibilidade para a fixação das placas e também para permitir a abertura do protótipo.

Figura 43 – Protótipo aberto

Fonte: Autoria própria (2019).

A Figura 44, demonstra como fica a bancada com o uso do protótipo em conjunto com a interface gráfica.

Figura 44 – Conjunto interface e protótipo

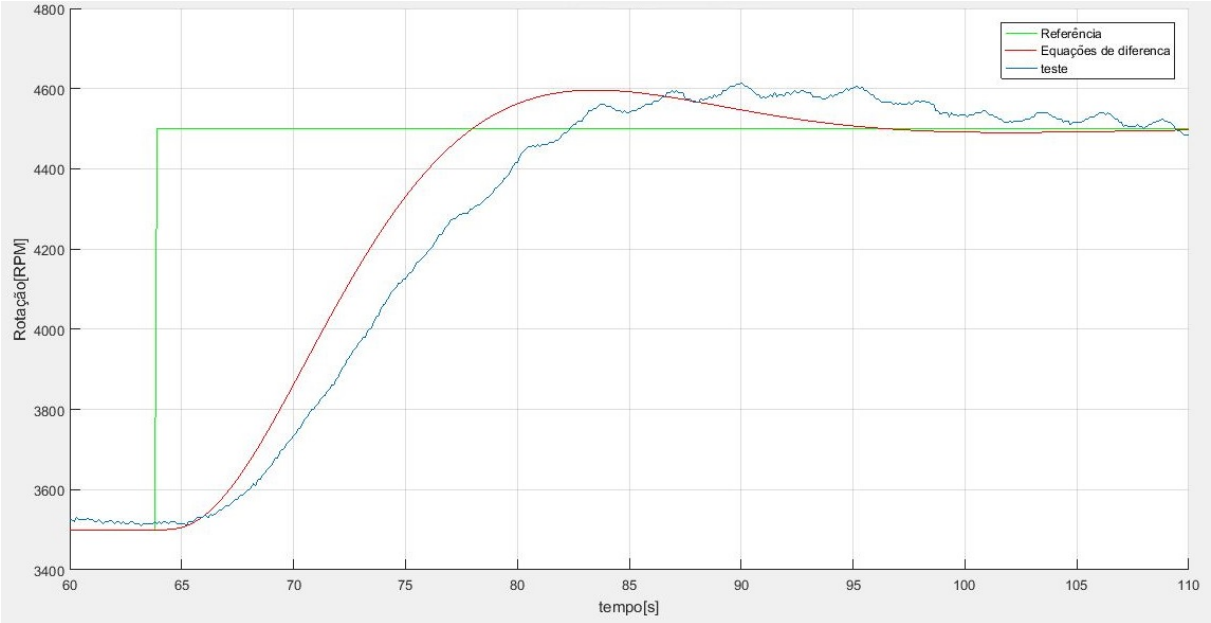


Fonte: Autoria própria (2019).

Com um projeto de controle, utilizando o filtro de referência, feito a alcançar um tempo de acomodação de 24 segundos e um sobressinal máximo de 10% chegou-se a um sistema que os polos dominantes são, 0.1250 $\pm$ 0.1705i. A parte real do terceiro polo deve ser igual ou maior a 10 vezes à parte real dos polos dominantes e isto deve garantir que este terceiro polo não influencie de forma significativa na resposta do sistema. Foram desenvolvidos 3 projetos, onde foram modificados apenas o terceiro polo do sistema, isso foi feito alterando o valor de K_i e calculando os novos valores de K_p e K_d. No primeiro projeto o terceiro polo foi posicionado a levar com que sua parte real tenha aproximadamente 10 vezes a parte real dos polos dominantes, o segundo foi projetado a posicionar o terceiro polo em 15 vezes e terceiro projeto foi feito para levar a 20 vezes. Em todos estes 3 testes feitos, o estágio de potência foi ajustado para fornecer até 40 V, e foi utilizado o método de contagem de bordas para a leitura da velocidade de rotação. Em todos os testes a velocidade de rotação e a referência se encontravam em 3500 RPM no inicio, e foi aplicado um degrau de 1000 RPM, levando a referência à 4500 RPM.

A Figura 45 mostra a resposta do sistema utilizando o primeiro projeto, com o terceiro polo em 10 vezes a parte real dos polos dominantes. Neste teste foi alcançado um sobressinal máximo de 11,5% e um tempo de acomodação de 34.6 segundos.

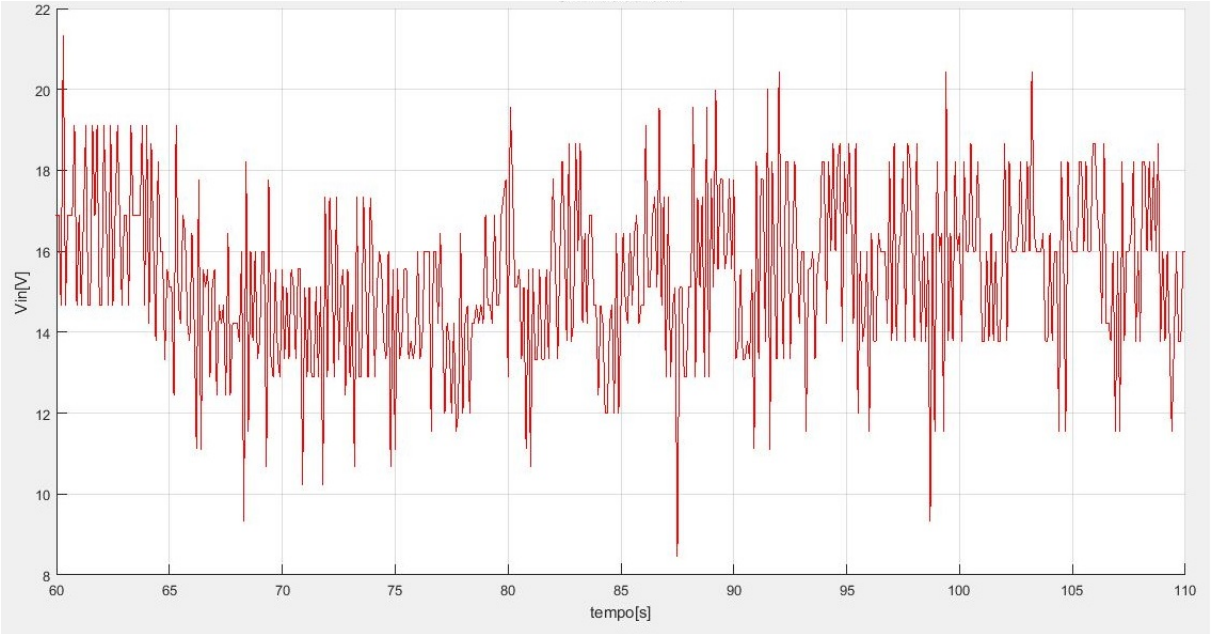
Figura 45 – teste com o terceiro polo 10 vezes a parte real dos polos dominantes



Fonte: Autoria própria (2019).

A Figura 46 mostra o sinal de controle deste teste do primeiro projeto, este sinal teve como valor máximo 25,33 V e como valor mínimo 8,44 V.

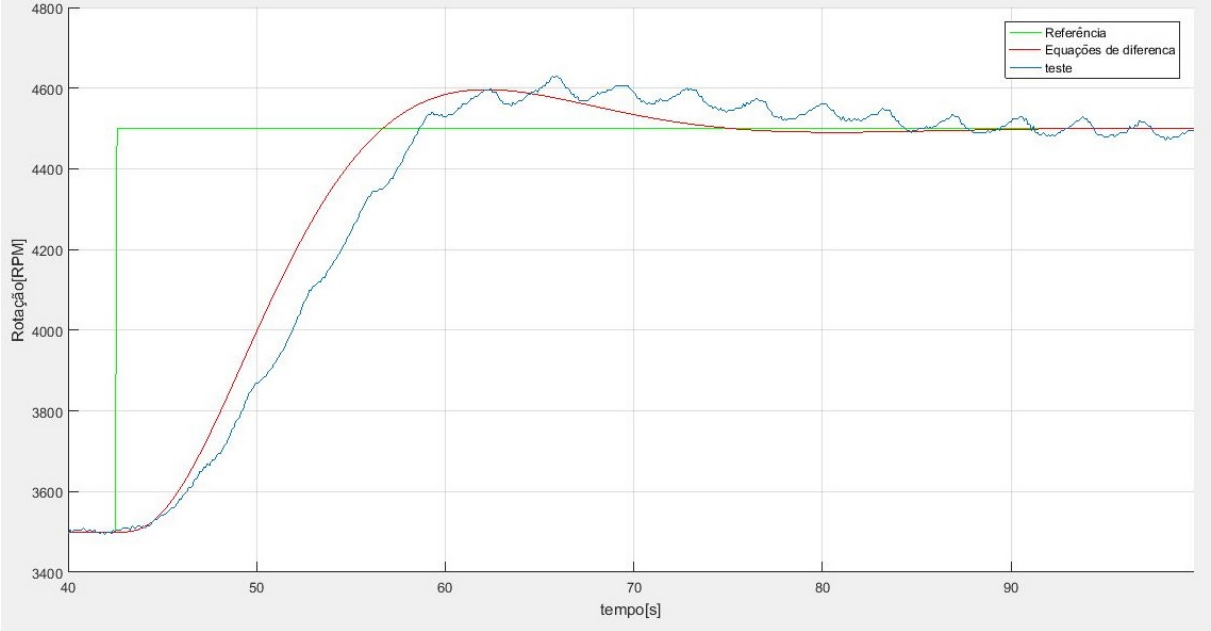
Figura 46 – Sinal de controle do teste com o terceiro polo 10 vezes a parte real dos polos dominantes



Fonte: Autoria própria (2019).

O resultado do teste do segundo projeto está na Figura 47, neste teste foi obtido um sobressinal máximo de 13% e um tempo de acomodação de 37,7 segundos.

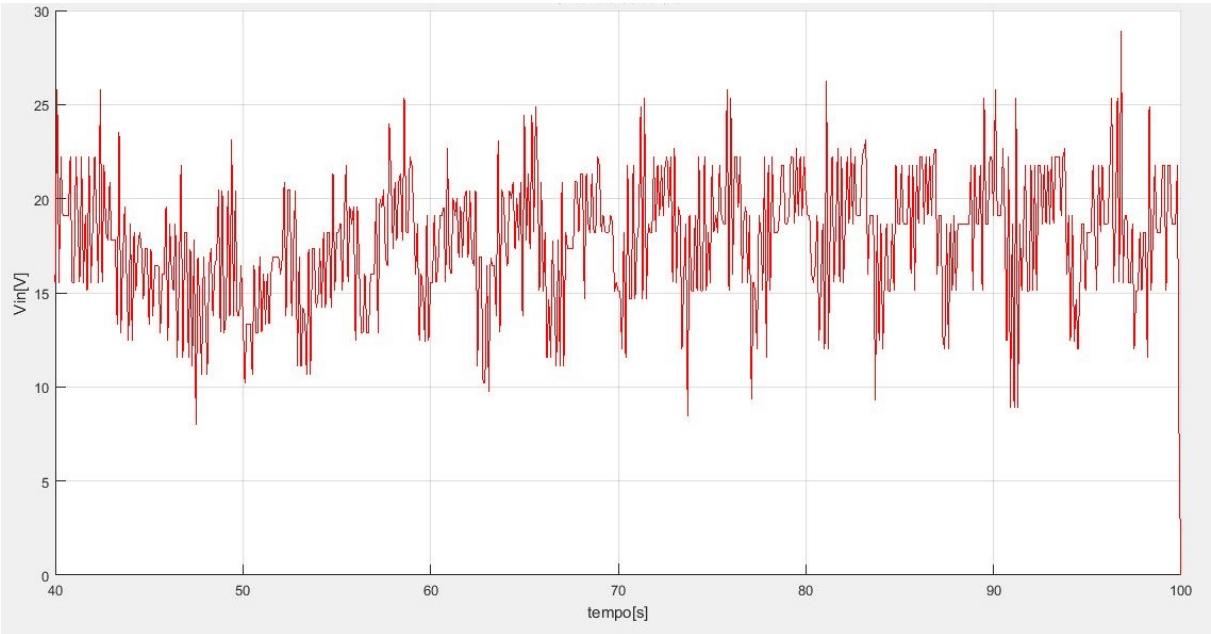
Figura 47 – teste com o terceiro polo 15 vezes a parte real dos polos dominantes



Fonte: Autoria própria (2019).

O sinal de controle do teste deste segundo projeto está em Figura 48, seus picos ficaram em 28,8 V e 8 V.

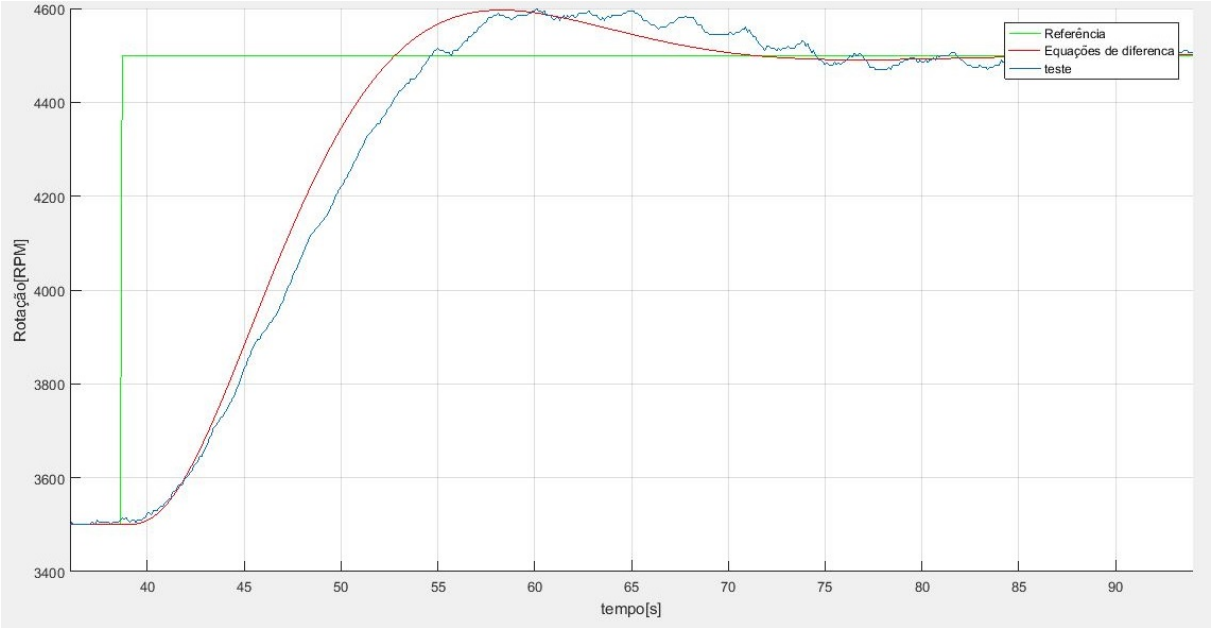
Figura 48 – sinal controle do teste com o terceiro polo 15 vezes a parte real dos polos dominantes



Fonte: Autoria própria (2019).

Já no terceiro teste foi alcançado um sobressinal máximo de 10% e um tempo de acomodação de 32,3 segundos.

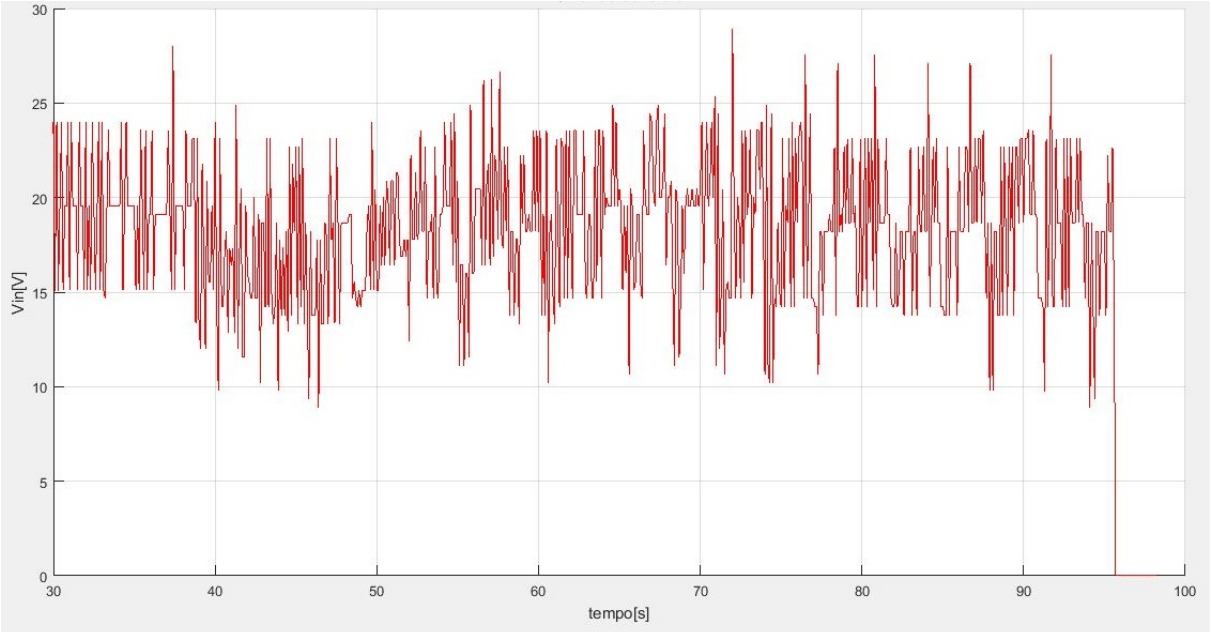
Figura 49 – teste com o terceiro polo 20 vezes a parte real dos polos dominantes



Fonte: Autoria própria (2019).

O sinal de controle do terceiro projeto ficou entre 28.89 e 8,89 V, Figura 50.

Figura 50 – sinal controle do teste com o terceiro polo 20 vezes a parte real dos polos dominantes



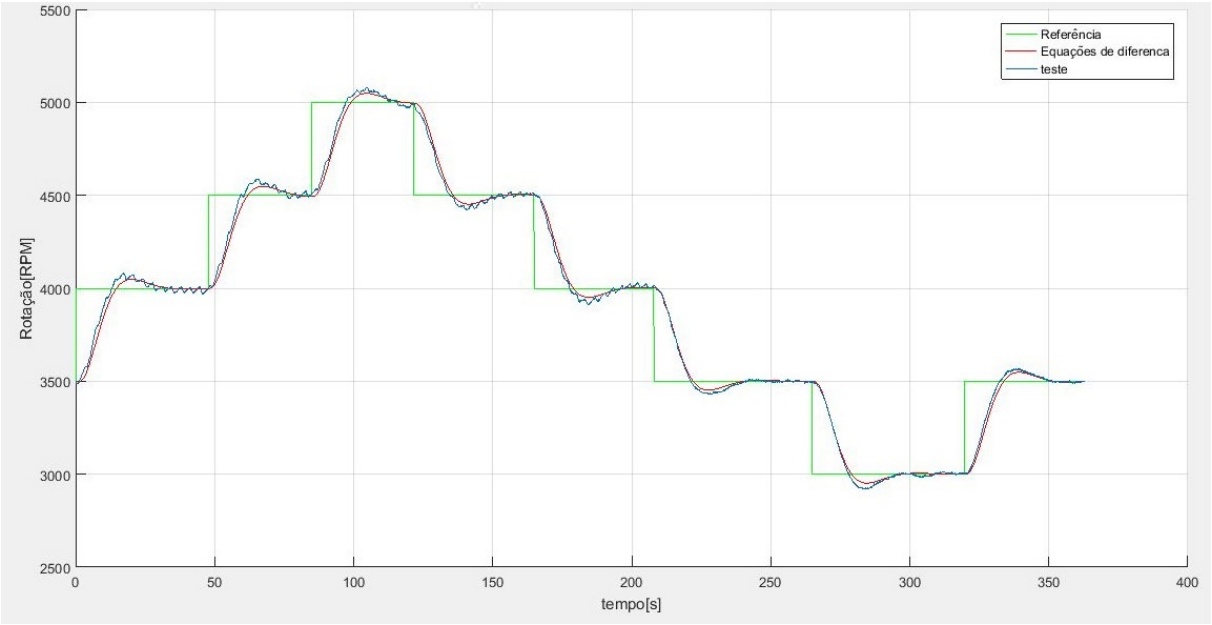
Fonte: Autoria própria (2019).

Com estes testes com diferentes valores para o terceiro polo, foi possível ver que embora a diferença entre as respostas seja pequena,a resposta do sistema

fica um pouco mais próxima do valor teórico quando se utiliza um valor maior para o terceiro polo. É possível ver, também, que o sinal de controle é mais oscilatório com os valores maiores para o terceiro polo, embora nestes testes não tenha acontecido, fica evidente que um valor exagerado ao terceiro polo deve levar o sistema à saturação, podendo assim prejudicar a resposta do sistema.

Para observar como este último sistema de controle projetado se comporta em outros pontos de operação, foi feito o teste demonstrado na Figura 51, neste teste foram mantidos todas as configurações do teste anterior, foram aplicados diversos degraus de mesmo módulo, 500 RPM, mas em diferentes velocidades de rotação.

Figura 51 – Teste com degraus



Fonte: Autoria própria (2019).

A Tabela 4 mostra os resultados deste teste, mostrando o tempo de acomodação e o valor do sobressinal máximo em cada degrau aplicado. É visto que os tempos de acomodação ficaram maiores que o projetado, ficando em média 27,19 segundos e o os valores de sobressinal máximo também ficaram maiores, ficando em média em 15,63%.

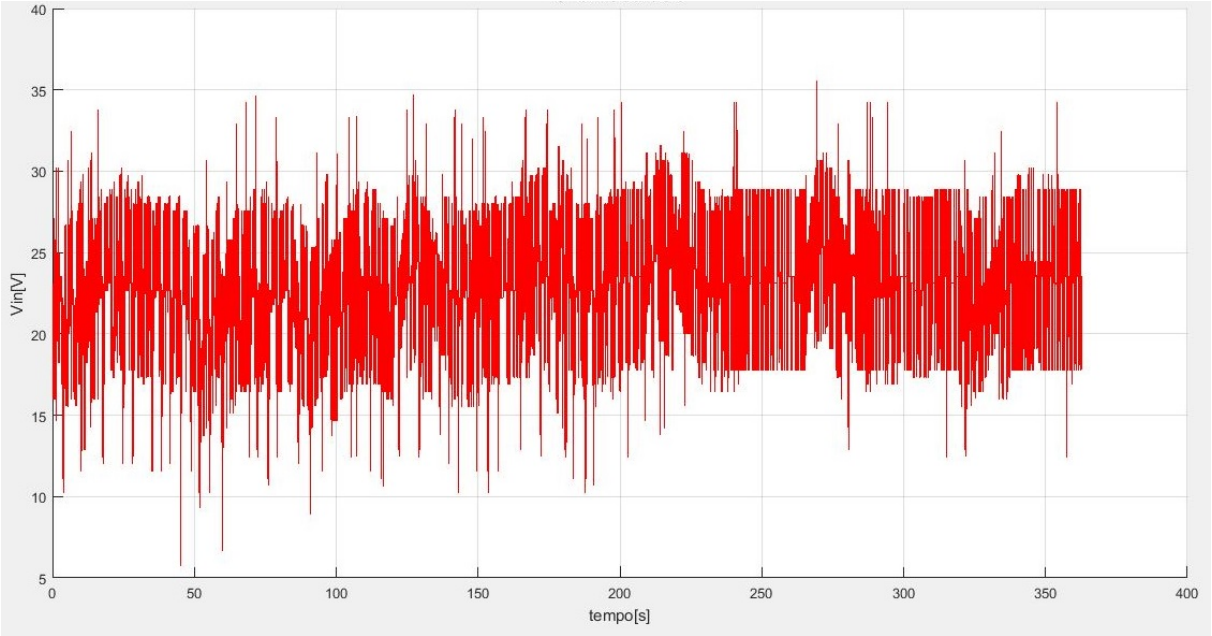
Tabela 4 – Tabela com os resultados do teste com diferentes degraus

Rotação inicial	Rotação final	M_p	t_s
3500	4000	16	27,9
4000	4500	18	27,7
4500	5000	16	26,9
5000	4500	14	26,8
4500	4000	18	27,2
4000	3500	14	27,9
3500	3000	16	26,7
3000	3500	13	26,4

Fonte: Autoria própria (2019).

A Figura 52, mostra o sinal de controle do teste feito com estes degraus em diferentes pontos de operação, não houve saturação neste sinal em nenhum momento durante o teste.

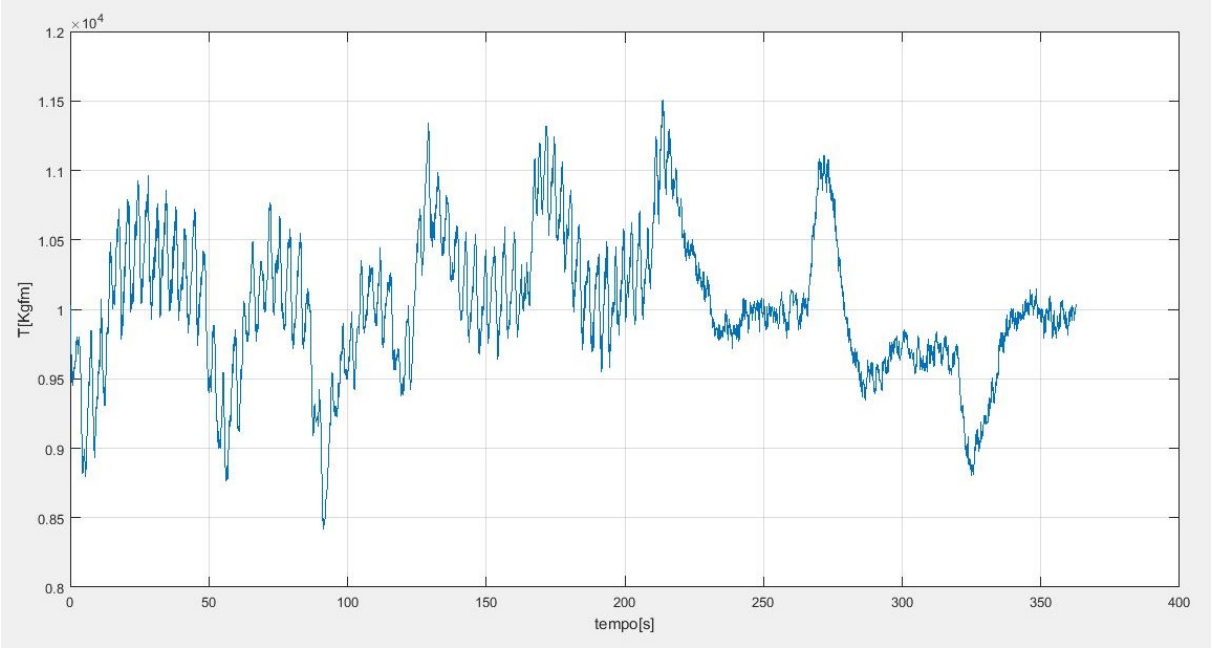
Figura 52 – Sinal de controle teste com degraus



Fonte: Autoria própria (2019).

A Figura 53 mostra o valor de torque lido no freio durante esse teste apresentado.

Figura 53 – Torque do teste com degraus

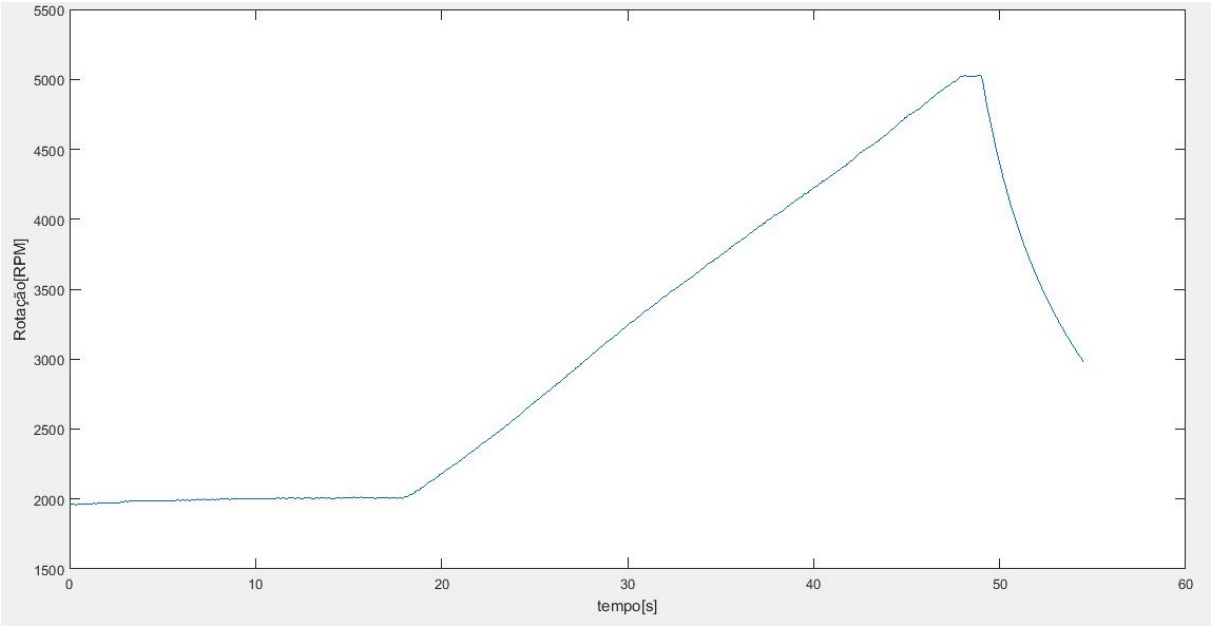


Fonte: Autoria própria (2019).

Foi feito um teste com o intuito de levantar o ponto de máximo torque do motor. Neste teste utilizou-se o *softstart* para causar uma variação constante no valor da velocidade de rotação, iniciando-se em 2000 e indo até 5000 RPM.

Na Figura 54, é apresentado a velocidade de rotação do teste feito, onde em um trecho foi aplicado uma variação constante nesta velocidade.

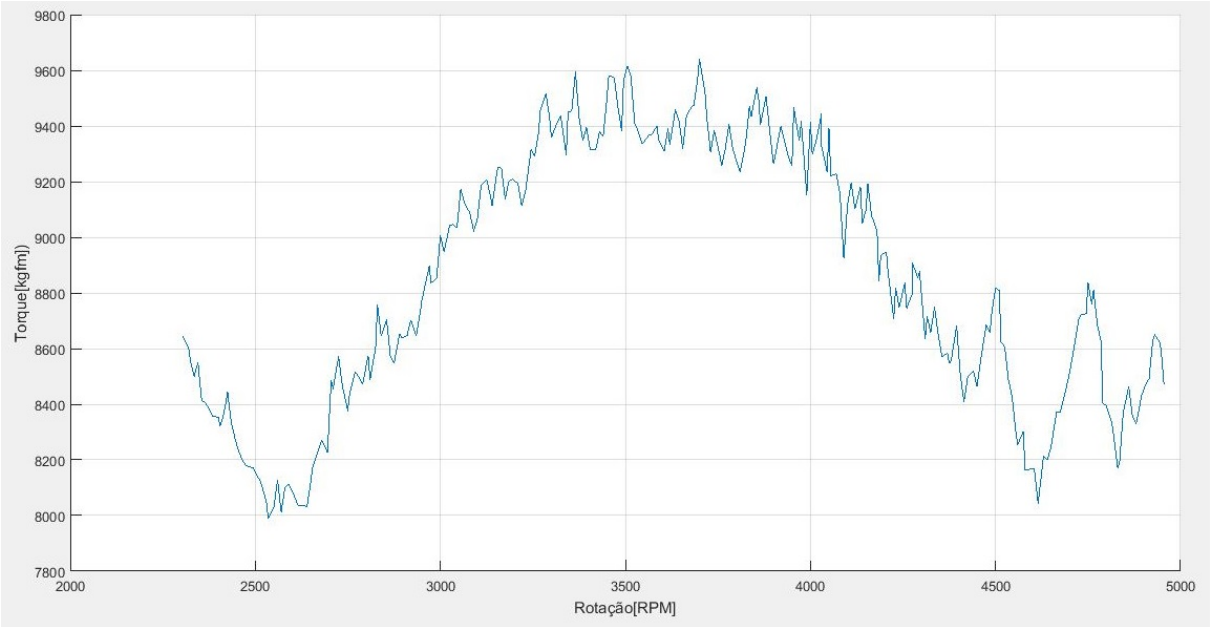
Figura 54 – Velocidade do teste com variação constante



Fonte: Autoria própria (2019).

Na Figura 55 tem-se o resultado torque pela rotação, deste teste.

Figura 55 – Torque do teste com variação constante da velocidade de rotação



Fonte: Autoria própria (2019).

É possível verificar que a velocidade de rotação que proporcionou o maior valor de torque foi em 3700 RPM, com 9,64 kgfm, porém em 3505 RPM obteve-se um valor próximo, 9,62 kgfm. Como a diferença entre as medidas é pequena não possível afirmar que o ponto de máximo torque do motor seja em 3700. Seriam necessários alguns testes deste tipo ou até mesmo um teste com a rotação constante, para ser possível esta afirmação.

6 CUSTOS

Neste capítulo é feita uma estimativa do custo material utilizado para a construção do protótipo utilizado nesta pesquisa. O custo levantado, demontrado na Tabela 5, é aproximado e se limita ao custo de componentes eletrônicos, encapsulamento e sensores do sistema, não levando em consideração o custo dos *softwares* e equipamentos utilizados durante esta pesquisa e nem da mão de obra empregada para sua elaboração.

Tabela 5 – Tabela de custos

Item	Quantidade	Preço unitário[R$]
módulo ADS1115	1	39,90
módulo BMP280	1	32,00
Caixa Patola pb-170/3	1	36,40
display LCD 20x4	1	42,90
Célula de carga	1	373,90
Roda fônica	1	11,48
Sensor Hall	1	217,39
SPC1-50-E	1	398,48
ATmega328p	1	14,90
7805	1	2,40
cabo usb/ttl	1	17,75
led	2	0,20
ina125	1	27,85
conector 20 pinos	2	0,59
conector 10 pinos	2	0,42
trimpot	3	1,31
conector jack	1	0,90
lm358	1	0,89
bc547	2	0,14
capacitor	16	0,07
resistor	20	0,06
diodo	2	0,08
cristal 16 MHz	1	0,64
Placa dupla face 20X20	1	14,40
Fonte 12 V	1	5,69

Fonte: Autoria própria (2019).

O custo material empregado na construção do protótipo desta pesquisa foi de R$ 1247,08. Este custo pode variar, dependendo do fornecedor utilizado ou a quantidade de equipamentos a serem confeccionados, e serve apenas como estimativa para novos projetos do tipo.

7 CONSIDERAÇÕES FINAIS

O sistema desenvolvido durante este trabalho mostrou-se adequado para se alcançar o objetivo geral. Foi possível, com este sistema, controlar, de forma adequada, a velocidade de rotação do motor, possibilitando, assim, efetuar testes de potência no mesmo.

A interface gráfica facilitou muito o acompanhamento e controle do sistema durante os testes, porém havia a necessidade de o dispositivo ter o software MATLAB, pois ele era executado em seu ambiente, e essa necessidade limitou os dispositivos disponíveis à sua utilização.

A modelagem do sistema, feita utilizando a resposta ao degrau, mostrou-se válida e serviu de base para o projeto de um sistema de controle adequado; porém, por se tratar de um sistema não linear houve pequenas diferenças nos resultados obtidos em diferentes velocidades de rotação.

No projeto do sistema de controle, o erro nulo ao regime permanente e a estabilidade já são contemplados, tendo em vista que é acrescentado um integrador ao sistema e que todos os polos tem sua parte real negativa. Os resultados obtidos em simulação e na prática validam e vão ao encontro do que é apresentado na teoria, mas na prática surgiram pequenas diferenças na resposta, dependendo do ponto de operação.

Para se chegar aos requisitos de tempo de acomodação e sobressinal máximo foi necessário a utilização do filtro de referência, que evita a saturação do sinal de controle. Para testes onde se deseja uma variação constante da velocidade de rotação, utilizou-se também o *softstart*.

Como no sistema de controle, com o filtro de referência, testado o tempo de acomodação era de 24 segundos, e o requisito requer um tempo de acomodação de 30 segundos, mesmo com o erro entre o valor calculado e o teste foi possível atender a este requisito de controle. Já no requisito de sobressinal máximo, o projeto de controle visava ao valor exato e, por este motivo, não conseguiu atendê-lo.

Para garantir que as especificações de projeto fossem alcançadas, independente do ponto de operação, seria necessário um projeto de sistema de controle mais restritivo, ou com uma margem de erro aos requisitos, tempo de acomodação e sobressinal máximo. Este projeto de controle foi feito e simulado, mas não foi testado, com o sistema real, até a entrega deste trabalho.

O projeto iniciado nesta pesquisa abre um leque de possibilidades para

trabalhos futuros, em que este sistema pode ser melhorado, tanto na parte técnica quanto em funcionalidades ao usuário. Sugere-se, a incorporação da fonte de alimentação e do estágio de potência ao protótipo, com isso o sistema ficaria mais portátil e com menos fios, o que deixaria o ambiente de testes mais agradável. Indica-se, também, a mudança do sistema de comunicação entre o protótipo e a interface gráfica para um sistema de comunicação sem fio, além de se evitar a necessidade do protótipo se encontrar próximo ao computador que está executando a interface gráfica, possibilitaria, ainda, a criação de interfaces gráficas em outros tipos de dispositivos, como *smartphones* ou *tablets*. A melhoria no sistema de controle é uma outra possibilidade que pode ser explorada em trabalhos futuros; pode-se implementar outros tipos de sistemas de controles encontrados na literatura, como sistemas de controles baseadas no espaço de estados, ou até mesmo um sistema de controle adaptativo, tendo em vista que um equipamento deste tipo normalmente é utilizado para se testar diferentes motores à combustão, o que leva o sistema a ter diferentes respostas e necessitando, com isto, de diferentes sistemas de controle. Como o sistema projetado trabalha com sensores localizados em pontos diferentes do ambiente de testes e o usuário normalmente se encontra afastado do conjunto motor/dinamômetro, por questões de segurança, é interessante também para trabalhos futuros um sistema que trabalhe com módulos, e que estes módulos se comuniquem utilizando comunicação sem fio. O projeto, nestes moldes, proporcionaria uma melhora na leitura dos sensores, onde os sinais lidos não sofreriam tanto com a influência de ruídos e diminuiria a quantidade de fios que atravessam o ambiente de testes. Poderiam, ainda, ser acrescentados outros parâmetros na aquisição de dados, como a emissão de poluentes e quantidade de combustível disponível ou consumido.

REFERÊNCIAS

ATMEGA328/P: Datasheet complete. 2016. ATMEGA328/P DATASHEET COMPLETE. Disponível em: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. Acesso em: 03/06/2019.

AUTONICS. *SPC1 Series*: Single-phase, power controller. [S.l.], 2019.

DORF, R. C.; BISHOP, R. H. *Sistemas de controle modernos*. [S.l.: s.n.], 2013.

HAICAL, R. da C. *Desenvolvimento de um sistema de controle de dinamômetro para testes de motores de combustão interna*. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, Porto Alegre, 2009.

HALLIDAY, D.; RESNICK, R.; WALKER, J. *Fundamentals of Physics Extended*. [S.l.: s.n.], 2011.

HEYWOOD, J. B. *Internal Combustion Engine Fundamentals*. [S.l.]: R. R. Donnelley Sons Company, 1988.

KERSCHBAUMER, R. *Microcontroladores*. 2018. Disponível em: <http://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/wp-content/uploads/sites/43/2018/02/Apostila-Microcontroladores.pdf>.

OGATA, K. *Discrete-Time Control Systems*. 2. ed. [S.l.]: Prentice Hall, 1994.

OGATA, K. *Modern Control Engineering*. 5. ed. [S.l.]: •, 2010.

PHILLIPS, C. L.; HARBOR, R. D. *Sistemas de Controle e Realimentação*. [S.l.: s.n.], 1997.

SOUZA, R. D. de. *BALANÇA CURIE E CORRENTES DE FOUCAULT*. 2005.

TAI SENG MACHINERY. *CZL-301*: Load cell. [S.l.], 2000.

TEXAS INSTRUMENTS. *INA125*: Instrumentation amplifier with precision voltage reference. [S.l.], 2009. Disponível em: <http://www.ti.com/lit/ds/symlink/ina125.pdf>.

TILLMANN, C. A. da C. *Motores de combustão interna e seus sistemas*. 2013.

Apêndices

APÊNDICE A – CÓDIGO FONTE DO MICROCONTROLADOR

```
/*
 * main.c
 *
 * Created: 25/03/2019 11:29:35
 * Author : Marcio Antonio Augusto
 */

#include "main.h"

inline void check_receiver(float *pf_Kp, float *pf_Kd, float *pf_Ki,
   uint16 *ui16_rot_ref, float *f_ui16_rot_ref_filtrada,uint8 *
   pui8_manual_value, float *pf_u3, uint8 *pui8_modo, uint8 *pui8_Vmax,
   float *pf_ang_torque){
  int8 i8_usart_msg[50], palavra[50];
  uint8 seta = 0;
  if(!usartIsReceiverBufferEmpty()){
    while(!usartIsReceiverBufferEmpty()){
      palavra[seta] = usartGetDataFromReceiverBuffer();
      if(palavra[seta] == '$'){//inicio da palavra
        seta = 1;
        palavra[0] = '$';
      }
      else{
        seta ++;
        if(palavra[seta-1] == ';'){//caracter de final de msg - final da
    palavra
          palavra[seta] = '\0';//proximo caracter recebe final de
   palavra
          seta = 0;
          if(palavra[0] == '$'){//inicio esta correto, pronto para
   avaliar a msg
            if(!strncmp((const char*)palavra,"$MODE=",6)){
              unsigned int AA = 0;
              sscanf((const char*)palavra,"$MODE=%u;",&AA);
              *pui8_modo = (uint8) AA;
              if(*pui8_modo == MODO_DRPM_CTE || *pui8_modo ==
   MODO_RPM_CTE){

                *pf_u3 = 0;//zera o valor acumulado no integrador

                ui16_rot_ref[1] = ui16_rot;
                ui16_rot_ref[0] = ui16_rot_ref[1];
```

```
              //inicia a referencia filtrada no ponto atual
              f_ui16_rot_ref_filtrada[2] = (float)ui16_rot;
              f_ui16_rot_ref_filtrada[1] = f_ui16_rot_ref_filtrada[2];
              f_ui16_rot_ref_filtrada[0] = f_ui16_rot_ref_filtrada[2];
            }
            sprintf(i8_usart_msg,"$MODE = %1d;\n",(unsigned int)*
pui8_modo);
            usart_send_string(i8_usart_msg);
          }
          else{
            if(!strncmp((const char*)palavra,"$CNT=",5)){
              usart_send_string("control ON/OFF command identyfied");
            }
            else{
              if(!strncmp((const char*)palavra,"$Vmax=",6)){
                sscanf((const char*)palavra,"$Vmax=%u;",(unsigned int
*)pui8_Vmax);
                sprintf(i8_usart_msg,"\n$Vmax = %1d;\n",(unsigned int)
*pui8_Vmax);
                usart_send_string(i8_usart_msg);
              }
              else{
                if(!strncmp((const char*)palavra,"$Ki=",4)){
                  *pf_Ki = strtof((const char *)&palavra[4] , NULL);
                  char tst[50];
                  if(*pf_Ki >= 0){
                    ftoa(*pf_Ki, tst, 4);
                    sprintf(i8_usart_msg,"\n$Ki = %s;\n",tst);
                  }
                  else {
                    ftoa(-*pf_Ki, tst, 4);
                    sprintf(i8_usart_msg,"\n$Ki = -%s;\n",tst);
                  }
                  usart_send_string(i8_usart_msg);
                }
                else{
                  if(!strncmp((const char*)palavra,"$Kp=",4)){
                    *pf_Kp = strtof((const char *)&palavra[4] , NULL);
                    char tst[50];
                    if(*pf_Kp >= 0){
                      ftoa(*pf_Kp,  tst, 4);
                      sprintf(i8_usart_msg,"\n$Kp = %s;\n",tst);
                    }
                    else {
                      ftoa(-*pf_Kp,  tst, 4);
                      sprintf(i8_usart_msg,"\n$Kp = -%s;\n",tst);
                    }
```

```
                    usart_send_string(i8_usart_msg);
                  }
                  else{
                    if(!strncmp((const char*)palavra,"$Kd=",4)){
                      *pf_Kd = strtof((const char *)&palavra[4] , NULL
);
                      char tst[50];
                      if(*pf_Kd >= 0){
                        ftoa(*pf_Kd,  tst, 4);
                        sprintf(i8_usart_msg,"\n$Kd = %s;\n",tst);
                      }
                      else {
                        ftoa(-*pf_Kd,  tst, 4);
                        sprintf(i8_usart_msg,"\n$Kd = -%s;\n",tst);
                      }
                      usart_send_string(i8_usart_msg);
                    }
                    else{
                      if(!strncmp((const char*)palavra,"$Rot=",5)){
                        //usart_send_string("Rot command identyfied");
                        sscanf((const char*)palavra,"$Rot=%i;",&
ui16_rot_ref[2]);
                        sprintf(i8_usart_msg,"\n$Rot = %d;\n",
ui16_rot_ref[2]);
                        usart_send_string(i8_usart_msg);
                      }
                      else{
                        if(!strncmp((const char*)palavra,"$Tor=",5)){
                          usart_send_string("Tor command identyfied");
                          usart_send_string("\nFuncao nao implementada
");
                        }
                        else{
                          if(!strncmp((const char*)palavra,"$Man=",5))
{
                            sscanf((const char*)palavra,"$Man=%i;",(
int*)pui8_manual_value);
                            sprintf(i8_usart_msg,"\n$Man = %3d;\n",(
unsigned int)*pui8_manual_value);
                            usart_send_string(i8_usart_msg);
                          }
                          else{
                            if(!strncmp((const char*)palavra,"
$Sincronismo;",12)){
                              sprintf(i8_usart_msg,"\n$MODE = %d;\n",*
pui8_modo);
                              usart_send_string(i8_usart_msg);
```

```
                                sprintf(i8_usart_msg,"\n$Rot = %d;\n",
ui16_rot_ref[2]);
                                usart_send_string(i8_usart_msg);
                                char tst[50];
                                if(*pf_Kd >= 0){
                                  ftoa(*pf_Kd,  tst, 4);
                                  sprintf(i8_usart_msg,"\n$Kd = %s;\n",
tst);
                                }
                                else {
                                  ftoa(-*pf_Kd,  tst, 4);
                                  sprintf(i8_usart_msg,"\n$Kd = -%s;\n",
tst);
                                }
                                usart_send_string(i8_usart_msg);

                                if(*pf_Kp >= 0){
                                  ftoa(*pf_Kp,  tst, 4);
                                  sprintf(i8_usart_msg,"\n$Kp = %s;\n",
tst);
                                }
                                else {
                                  ftoa(-*pf_Kp,  tst, 4);
                                  sprintf(i8_usart_msg,"\n$Kp = -%s;\n",
tst);
                                }
                                usart_send_string(i8_usart_msg);

                                if(*pf_Ki >= 0){
                                  ftoa(*pf_Ki, tst, 4);
                                  sprintf(i8_usart_msg,"\n$Ki = %s;\n",
tst);
                                }
                                else {
                                  ftoa(-*pf_Ki,  tst, 4);
                                  sprintf(i8_usart_msg,"\n$Ki = -%s;\n",
tst);
                                }
                                usart_send_string(i8_usart_msg);

                                sprintf(i8_usart_msg,"\n$Vmax = %d;\n",*
pui8_Vmax);
                                usart_send_string(i8_usart_msg);

                              }
                              else
                              {
```

```
                            if(!strncmp((const char*)palavra,"$Zero"
,5)){
                              char i;
                              uint32 ui32_temp = ui16_torque;
                              for(i = 0;i < 1;i++){
                                ads1115StartSingleConversion();
                                ads1115WaitUntilConversionFinish();
                                ads1115GetResult((uint16*)&
ui16_torque);
                                ui32_temp += ui16_torque;
                              }
                              ui16_torque_0 = ui32_temp/(i+1);
                              *pf_ang_torque = 50./(ui16_torque_50-
ui16_torque_0);
                              sprintf(i8_usart_msg,"$disp('Zero
calibrado - %d');",ui16_torque_0);
                              usart_send_string(i8_usart_msg);
                            }
                            else
                              if(!strncmp((const char*)palavra,"
$Calibrar=",10)){
                                float f_temp = strtof((const char *)
&palavra[10] , NULL);
                                char i;

                                uint32 ui32_temp = ui16_torque;
                                for(i = 0;i < 1;i++){
                                  ads1115StartSingleConversion();
                                  ads1115WaitUntilConversionFinish()
;
                                  ads1115GetResult((uint16*)&
ui16_torque);
                                  ui32_temp += ui16_torque;
                                }
                                ui16_torque_50 = ui32_temp/(i+1);

                              char tst[50];
                              if(f_temp >= 0){
                                ftoa(f_temp,  tst, 4);
                                sprintf(i8_usart_msg,"$disp('
Tcalibracao = %s Kg');\n",tst);
                                usart_send_string(i8_usart_msg);
                              }

                              *pf_ang_torque = f_temp/(
ui16_torque_50-ui16_torque_0);
                              sprintf(i8_usart_msg,"$disp('Calibrado
```

```
    - %d');",ui16_torque_50);
                                    usart_send_string(i8_usart_msg);
                                  }
                                }
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
              else{
                usart_send_string("Nao avaliado corretamente -> );");
                usart_send_string((char*)palavra);
                usart_send_string("\n");
              }
            }
          }
        }
      }
}

int main(void){

  // led_error configuration
  setBit(LED_ERROR_DDR,LED_ERROR_BIT);

  setBit(DDRD,PD3);//seta pino OC2B como saida//pino usado pelo PWM
  setBit(PORTD,PD3);

  //buttons configuration
  setBit(BOTAO_CIMA_PORT,BOTAO_CIMA_BIT);
  setBit(BOTAO_BAIXO_PORT,BOTAO_BAIXO_BIT);
  setBit(BOTAO_SELECT_PORT,BOTAO_SELECT_BIT);
  setBit(BOTAO_VOLTAR_PORT,BOTAO_VOLTAR_BIT);
  setBit(SW_MANUAL_PORT,SW_MANUAL_BIT);

  // LCD configuration
  lcdSetDataPort(&display, &DISPLAY_DATA_DDR, &DISPLAY_DATA_PORT,
   DISPLAY_DATA_FIRST);
  lcdSetControlPort(&display, &DISPLAY_CTRL_DDR, &DISPLAY_CTRL_PORT,
   DISPLAY_CTRL_E, DISPLAY_CTRL_RS);
  lcdInit(&display, LCD_20X4, LCD_FONT_5X8);
  lcdStdio(&display);
```

```
//UART configuration
usartConfig(USART_MODE_ASYNCHRONOUS, USART_BAUD_57600,
 USART_DATA_BITS_8, USART_PARITY_NONE, USART_STOP_BIT_SINGLE);
usartEnableReceiver();
usartEnableTransmitter();
usartClearReceptionBuffer();
usartActivateReceptionCompleteInterrupt();
usart_send_string("Hello UART!\n");

//int0 configuration, usado para detectar a rotacao do motor
int0Config(PORT_INPUT_PULL_UP,SENSE_ANY_EDGE);//entrada do sinal da
 roda fonica
int0ClearInterruptRequest();
int0ActivateInterrupt();

//timer2 configuration, usado para gerar o PWM
timer2SetCompareAValue(PWM_COMPARE_A_VALUE);//define a frequencia do
 PWM
timer2SetCompareBValue(110);//define o DC do PWM para 0%
timer2OutputConfig(TIMER_PORT_NO_CHANGE, TIMER_PORT_SET_ON_COMPARE);
timer2Config(TIMER_A_MODE_FAST_PWM_OCRA, PWM_PRESCALER);//PWM, clear
 on MAX

setBit(DDRD,PD3);//seta pino OC2B como saida//pino usado pelo PWM


    //timer0 configuration//(compAValue = F_CPU*20ms/(divisor*
 prascaler))-1 <= (2^8-1)// 100 ms -> 10 Hz // usado para gerar as
 interrupcoes de controle
    timer0Config(TIMER_A_MODE_CTC,TIMER_A_PRESCALER_256);
    //timer0ActivateCompareAInterrupt();
    timer0SetCompareAValue(249);

#endif

sei();

//i2c configuration
twiMasterInit(400000);

//ADS1115 configuration
uint16 ui16_ADS1115_config = 0;

ui16_ADS1115_config = ADS1115_DISABLE_COMPARATOR|ADS1115_CONFIG_DR_860
 |ADS_1115_SINGLE_SHOT_MODE|ADS1115_GAIN_2_3|ADS1115_CONFIG_INA125_GND
 ;
```

```
}
ads1115Config(&ui16_ADS1115_config);

uint8 ui8_send_cont = 1;//usado para contar o numero de vezes que eh
 enviado dados pela usart, para marcar tempo

uint16 ui16_rot_ref[]  = {1000,1000,1000};//usado para receber o valor
  de referencia para a rotacao

float f_rot_ref_filtrada[] = {1000,1000,1000};//usado para receber o
 valor de referencia filtrada para a rotacao

float f_torque_ref[]   = {10,10,10};//usado para receber o valor de
 referencia para o torque

float f_torque_ref_filtrada[] = {10,10,10};//usado para receber o
 valor de referencia filtrada para o torque

const int16 ui16_fs = FS;//frequencia de amostragem

uint8 ui8_manual_value = 0;//armazena o valor do pwm manual

float f_erro_ant = 0;//usado para armazenar o valor de erro anterior

float f_torque = 0;//usado para receber o valor do torque

float f_ang_torque = 50./(ui16_torque_50-ui16_torque_0);//angulo para
 calculo do torque

float f_u3 = 0;//valor armazenado pelo integrador

float f_Kd = -0.1779352951, f_Ki = -0.0080000000, f_Kp =
 -0.0452896212;

enum modes_t ui8_modo = MODO_OFF;// modo de controle

uint8 ui8_Vmax = 40;

printa_config();
send_config();

timer0ActivateCompareAInterrupt();//inicia as interrupcoes de
 referencia para o controle

for(;;){
  if((ui8_new_data_flag != 0)){
```

```
    ui8_new_data_flag--;
    ads1115WaitUntilConversionFinish();
    ads1115GetResult((uint16*)&ui16_torque);
    f_torque = f_ang_torque*(int16)(ui16_torque-ui16_torque_0);

    if(ui16_rot >= 1000 || ui16_rot == 0){
      switch(ui8_modo){
        case MODO_OFF:
          seta_tensao(0,ui8_Vmax);
          break;
        case MODO_DRPM_CTE:
          aplica_filtro_de_referencia(ui16_rot_ref,f_rot_ref_filtrada,
D_MAX);
          seta_tensao(calcula_controle(f_Kp, f_Kd, f_Ki, (float)
ui16_rot, f_rot_ref_filtrada[2], &f_u3, &f_erro_ant,ui16_fs),ui8_Vmax
);
          break;
        case MODO_RPM_CTE:
          aplica_filtro_de_referencia(ui16_rot_ref,f_rot_ref_filtrada
,0);
          seta_tensao(calcula_controle(f_Kp, f_Kd, f_Ki, (float)
ui16_rot, f_rot_ref_filtrada[2], &f_u3, &f_erro_ant,ui16_fs),ui8_Vmax
);

          break;
        case MODO_DTORQUE_CTE:
          aplica_filtro_de_referencia(ui16_rot_ref,f_rot_ref_filtrada
,1);
          seta_tensao(calcula_controle(f_Kp, f_Kd, f_Ki, f_torque,
f_torque_ref_filtrada[2], &f_u3, &f_erro_ant,ui16_fs),ui8_Vmax);
          break;
        case MODO_TORQUE_CTE:
          aplica_filtro_de_referencia(ui16_rot_ref,f_rot_ref_filtrada
,0);
          seta_tensao(calcula_controle(f_Kp, f_Kd, f_Ki, f_torque,
f_torque_ref_filtrada[2], &f_u3, &f_erro_ant,ui16_fs),ui8_Vmax);
          break;
        case MODO_MANUAL:
          atualiza_PWM(ui8_manual_value);
          break;
      }
    }
    else
      seta_tensao(0,ui8_Vmax);

    send_state(ui16_rot_ref[2],f_torque);
    check_receiver(&f_Kp, &f_Kd, &f_Ki, ui16_rot_ref,
```

```
    f_rot_ref_filtrada, &ui8_manual_value, &f_u3, &ui8_modo, &ui8_Vmax, &
    f_ang_torque);

       if(--ui8_send_cont==0){//ocorre a cada 500 ms
         ui8_send_cont=FS/2;
         atualiza_LCD(ui8_modo,ui16_rot, (uint16)f_rot_ref_filtrada[1],
    f_torque, f_torque_ref[1]);
       }
     }
   }
}

//interrupcao que ocorre a cada 4 ms e serve de referencia para o
    sistema de controle
ISR(TIMER0_COMPA_vect){
  if(--ui8_divider_t0==0){//ocorre na frequencia FS
    ui8_divider_t0 = INICIAL_VALUE_DIVIDER_T0;//usado para dividir e
    garantir que ocorra na frequencia FS
    setBit(LED_ERROR_PORT,LED_ERROR_BIT);
    if(!twiMasterIsBusy()){
      ads1115StartSingleConversion();
    }
    else{
      error(TWI_BUSY_ERROR_CODE);
    }
    ui16_rot = ui16_cont;
    ui16_cont = 0;//recomeca a contagem

    ui8_new_data_flag++;
  }
}

//Interrupcao que ocorre na passagem de um dente
ISR(INT0_vect){
  ui16_cont += FS/2;//recebe metade do valor da frequencia de amostragem
}

//funcao utilizada para enviar uma string pela usart
void usart_send_string(char* msg){
  uint8 size = strlen(msg);
  uint8 i;
  for(i=0;i<size;i++)
    usartTransmit(msg[i]);
}

//funcao que trava o sistema e informa o erro ocorrido
void error(int id_erro){
```

```
  cli();//desabilita todas as interrupcoes
  while(id_erro == TWI_BUSY_ERROR_CODE){
    setBit(PORTD,PD3);//pino OC2B em alto, PWM -> 0%
    lcdClearScreen(&display);
    printf("ERROR\nTWI_BUSY...");
    usart_send_string("ERROR\nTWI_BUSY...");
    _delay_ms(1000);
  }
  while(id_erro == EMERGENCY_ERROR_CODE){
    setBit(PORTD,PD3);//pino OC2B em alto, PWM -> 0%
    lcdClearScreen(&display);
    printf("ERROR\nTWI_BUSY...");
    usart_send_string("ERROR\nTWI_BUSY...");
    _delay_ms(1000);
  }

  while(1);//trava o sistema
}

//funcao que atualiza o LCD
inline void atualiza_LCD(enum modes_t ui8_modo,uint16 ui16_rot, uint16
   ui16_rot_ref,float f_torque, float f_torque_ref){
  int16 i16_ipart = (int16)(f_torque);//parte inteira
  int16 i16_fpart = (int16)(100*(f_torque - i16_ipart));
  if(i16_fpart < 0)
    i16_fpart = -i16_fpart;

  lcdClearScreen(&display);

  printf("Ctrl -> ");

  switch(ui8_modo){
    case MODO_OFF:
      printf("OFF\n");
      break;
    case MODO_DRPM_CTE:
      printf("D_RPM cte\n");
      break;
    case MODO_RPM_CTE:
      printf("RPM cte\n");
      break;
    case MODO_DTORQUE_CTE:
      printf("D_TORQUE cte\n");
      break;
    case MODO_TORQUE_CTE:
      printf("TORQUE cte\n");
      break;
```

```
    case MODO_MANUAL:
      printf("Manual\n");
      break;
  }

  printf("PWM=%3d\n",timer2GetCompareBValue());

  printf("ROTACAO = %4d",ui16_rot);
  if((ui8_modo == MODO_RPM_CTE) | (ui8_modo == MODO_DRPM_CTE))
    printf("->%4d",ui16_rot_ref);

  printf("\nT = %i,%02i",i16_ipart,i16_fpart);

  if(ui8_modo == MODO_TORQUE_CTE){
    i16_ipart = (int16)(f_torque_ref);//parte inteira
    i16_fpart = (int16)(100*(f_torque_ref - i16_ipart));
    if(i16_fpart < 0)
      i16_fpart = -i16_fpart;
    printf("-> %i,%02i",i16_ipart,i16_fpart);

  }
}

//funcao que atualiza o PWM
inline void atualiza_PWM(uint8 ui8_valor){
  timer2SetCompareBValue(ui8_valor);
  clrBit(LED_ERROR_PORT,LED_ERROR_BIT);
}

//funcao que calcula a rotacao
inline void calcula_rot(){
  #if(METODO_CALCULO_ROTACAO == 1)//metodo utilizando tempo entre os
   dentes
    #if(N_AMOSTRAS_ROTACAO == 16)//calcula com uma media de 16 valores
      ui16_rot = (F_CPU/((ui16_cont[0]) + (ui16_cont[1]) + (ui16_cont
   [2]) + (ui16_cont[3]) + (ui16_cont[4])+ (ui16_cont[5]) + (ui16_cont
   [6]) + (ui16_cont[7])));//calcula a rotacao do motor, com a media do
   tempo entre os ultimos dentes
      ui16_rot = (ui16_rot>>1) + ((F_CPU/((ui16_cont[0]) + (ui16_cont
   [1]) + (ui16_cont[2]) + (ui16_cont[3]) + (ui16_cont[4])+ (ui16_cont
   [5]) + (ui16_cont[6]) + (ui16_cont[7])))>>1);
    #else
      #if(N_AMOSTRAS_ROTACAO == 8)//calcula com uma media de 8 valores
        ui16_rot = (F_CPU/((ui16_cont[0]) + (ui16_cont[1]) + (ui16_cont
   [2]) + (ui16_cont[3]) + (ui16_cont[4])+ (ui16_cont[5]) + (ui16_cont
   [6]) + (ui16_cont[7])));//calcula a rotacao do motor, com a media do
   tempo entre os ultimos dentes
```

```
      #else
        #if(N_AMOSTRAS_ROTACAO == 2)//calcula com uma media de 2 valores
          ui16_rot = 2*( F_CPU/((ui16_cont[0]) + (ui16_cont[1])));
        #else
          ui16_rot = (F_CPU/(ui16_cont));
        #endif
      #endif
    #endif
  #else
    //metodo utilizando a contagem do numero de pulsos
    ui16_rot = ui16_cont;
    ui16_cont = 0;//recomeca a contagem
  #endif
}

void printa_config(){
  printf("fs = %u\n",FS);

  #if METODO_ADS == 0
    printf("INads = INA\n");
  #else
    printf("INads = CELULA\n");
  #endif

  printf("MET_CALC_ROT = %u\n",METODO_CALCULO_ROTACAO);

  switch(PWM_PRESCALER){
    case TIMER_A_PRESCALER_OFF:
      printf("fPWM = %lu",F_CPU/(PWM_COMPARE_A_VALUE));
      break;
    case TIMER_A_PRESCALER_8:
      printf("fPWM = %lu",F_CPU/(PWM_COMPARE_A_VALUE*8));
      break;
    case TIMER_A_PRESCALER_64:
      printf("fPWM = %lu",F_CPU/(((uint16)PWM_COMPARE_A_VALUE)*64));
      break;
    case TIMER_A_PRESCALER_256:
      printf("fPWM = %lu",F_CPU/(((uint16)PWM_COMPARE_A_VALUE)*256));
      break;
    case TIMER_A_PRESCALER_1024:
      printf("fPWM = %lu",F_CPU/(((uint32)PWM_COMPARE_A_VALUE)*1024));
      break;
    default:
      printf("fPWM = ???");
      break;
  }
  _delay_ms(500);
```

```
}

void send_config(){
  int8 i8_usart_msg[50];;

  sprintf(i8_usart_msg,"fs = %u\n",FS);
  usart_send_string(i8_usart_msg);

  if (METODO_ADS == 0)
    sprintf(i8_usart_msg,"INads = INA\n");
  else
    sprintf(i8_usart_msg,"INads = CELULA\n");

  usart_send_string(i8_usart_msg);

  sprintf(i8_usart_msg,"MET_CALC_ROT = %u\n",METODO_CALCULO_ROTACAO);
  usart_send_string(i8_usart_msg);

  switch(PWM_PRESCALER){
    case TIMER_A_PRESCALER_OFF:
    sprintf(i8_usart_msg,"fPWM = %lu",F_CPU/(PWM_COMPARE_A_VALUE));
    break;
    case TIMER_A_PRESCALER_8:
    sprintf(i8_usart_msg,"fPWM = %lu",F_CPU/(PWM_COMPARE_A_VALUE*8));
    break;
    case TIMER_A_PRESCALER_64:
    sprintf(i8_usart_msg,"fPWM = %lu",F_CPU/(((uint16)
   PWM_COMPARE_A_VALUE)*64));
    break;
    case TIMER_A_PRESCALER_256:
    sprintf(i8_usart_msg,"fPWM = %lu",F_CPU/(((uint16)
   PWM_COMPARE_A_VALUE)*256));
    break;
    case TIMER_A_PRESCALER_1024:
    sprintf(i8_usart_msg,"fPWM = %lu",F_CPU/(((uint32)
   PWM_COMPARE_A_VALUE)*1024));
    break;
    default:
    sprintf(i8_usart_msg,"fPWM = ???");
    break;
  }
  usart_send_string(i8_usart_msg);

}

inline void send_state(uint16 ui16_rot_ref, float f_torque){
  int8 i8_usart_msg[50];
```

```
  sprintf(i8_usart_msg,"$dado = [%u,%li,%u,%u];\n",ui16_rot,(int32)(
   f_torque*1000),ui16_rot_ref,timer2GetCompareBValue());
  usart_send_string(i8_usart_msg);
}

//define a interrupcao de recebimento da usart
USART_RECEIVER_BUFFER_FUNCTION_HANDLER

//funcao que aplica o filtro de referencia e o softstart
inline void aplica_filtro_de_referencia(uint16* pui16_rot_ref, float *
   pf_rot_ref_filtrada, uint8 ui8_Dmax){
  float f_numerador_filtro[] = {0,0.000222897281685,0.000221014137942};
   //numerador do filtro de referencia
  float f_denominador_filtro[] =
   {1,-1.974424421951404,0.974868333371031};//denominador do filtro de
   referencia

  //aplica o softstart
  if(ui8_Dmax != 0){
    //variacao da referencia
    int16 i16_rot_ref_dif = (int16)pui16_rot_ref[1] - (int16)
   pui16_rot_ref[0];

    //softstart
    if(i16_rot_ref_dif > ui8_Dmax)
    pui16_rot_ref[1]  = pui16_rot_ref[0] + ui8_Dmax;
    else{
      if(i16_rot_ref_dif < -ui8_Dmax)
      pui16_rot_ref[1]  = pui16_rot_ref[0] - ui8_Dmax;
    }
  }

  //filtro de referencia
  pf_rot_ref_filtrada[2] = (((f_numerador_filtro[1] * (float)(
   pui16_rot_ref[1])+ f_numerador_filtro[2] * (float)(pui16_rot_ref[0]))
   po9wse3
  -(f_denominador_filtro[1] * pf_rot_ref_filtrada[1] +
   f_denominador_filtro[2] * pf_rot_ref_filtrada[0])));

  pf_rot_ref_filtrada[0] = pf_rot_ref_filtrada[1];
  pf_rot_ref_filtrada[1] = pf_rot_ref_filtrada[2];

  pui16_rot_ref[0] = pui16_rot_ref[1];
  pui16_rot_ref[1] = pui16_rot_ref[2];
}

//funcao que aplica uma tensao na saida do estagio de potencia
```

```
inline void seta_tensao(float f_Vout, uint8 ui8_Vmax){
  /*
  pwm 0 a 110 ?
  */
  if(f_Vout < ui8_Vmax && f_Vout > 0){
    uint16 value = (90*(ui8_Vmax-f_Vout))/ui8_Vmax;
    atualiza_PWM(value);
  }
  else{
    if(f_Vout >= ui8_Vmax){
      atualiza_PWM(0);
    }
    else{
      atualiza_PWM(110);
    }
  }
}

inline float calcula_controle(float f_Kp, float f_Kd, float f_Ki, float
   f_val, float f_val_ref ,float *pf_u3, float *f_erro_ant,uint8 fs){
  float f_u;//saida
  float f_u1;//saida do ganho proporcional ao erro
  float f_u2;//saida do ganho proporcional a derivada do erro
  float f_erro;//erro entre o valor atual e o valor de referencia

  //calcula o erro
  f_erro = f_val_ref - f_val;

  //calcula saida referente ao ganho proporcional ao erro
  f_u1 = f_Kp*f_erro;

  //calcula saida referente ao ganho proporcional a derivada do erro
  f_u2 = (f_Kd*fs)*(f_erro-*f_erro_ant);

  //calcula saida referente ao ganho proporcional a integral do erro
  *pf_u3 = (f_Ki*f_erro)/fs + *pf_u3;

  //calcula saida completa
  f_u = f_u1 + f_u2 + *pf_u3;

  //guarda o erro
  *f_erro_ant = f_erro;

  //retorna valor a ser aplicado
  return f_u;
}
```

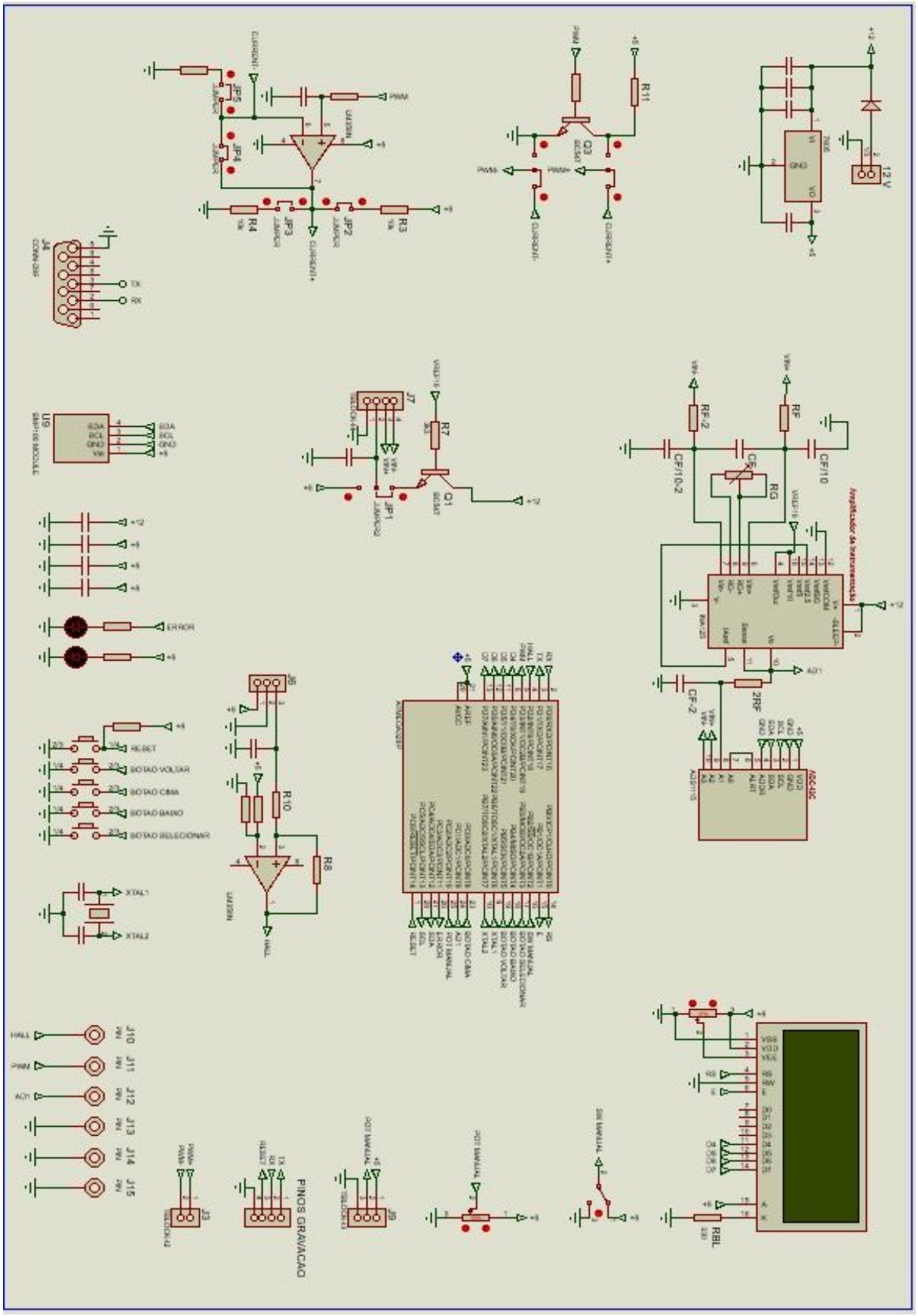
APÊNDICE B – *SCRIPT* DO MATLAB PARA O PROJETO DE CONTROLE

```
clear all;
close all;
clc;
disp('PROJETO DE CONTROLE DA ROTACAO')

%% define zeta, wn e K do sistema a ser controlado
zeta = 0.857;
wn = 0.167;
K = -1020;

%% define as especificacoes da resposta controlada
tau_requerido = 8;
Mp_requerido = 0.10;

%% define Ki do PID
Ki_escolhido = -.002;

%% define a funcao de transferencia da rotacao
GpH = K*tf(wn^2,[1 (2*zeta*wn) wn^2])

%% define zeta e wn, requeridos ao sistema controlado
zeta_requerido = 1/sqrt(((pi^2)+((log(Mp_requerido)^2)))/...
     ((log(Mp_requerido)^2)));

wn_requerido = 1./(tau_requerido*zeta_requerido);

%% calcula o polo que gera as especificacoes da resposta, s1
real_s1 = -1./(tau_requerido);
imag_s1 = (wn_requerido*sqrt(1-zeta_requerido^2));

s1 = real_s1 + j*imag_s1;
fprintf('s1 = %.4f + %.4fi\n',real(s1),imag(s1));

%% calcula o angulo de s1, em relacao a origem
beta = angle(s1);

%% extrai o numerador e o denominador da funcao de transferencia da
   planta
%e cria uma planta simbolica
[Num_GpH,Den_GpH] = tfdata(GpH,'v');
syms s
GpH_syms = poly2sym(Num_GpH,s)/poly2sym(Den_GpH,s);
```

```
%% calcula o modulo e angulo da planta no polo desejado
s = s1;
mod_GpH_s1 = abs(eval(GpH_syms));
phi = angle(eval(GpH_syms));

%% calcula os valores de Kp e Kd, utilizando as equacoes do livro
Kp = (-sin(beta + phi))./(mod_GpH_s1.*sin(beta))...
     - (2*Ki_escolhido.*cos(beta))./(abs(s1))

Kd = sin(phi)./(abs(s1).*mod_GpH_s1.*sin(beta)) ...
      + Ki_escolhido./(abs(s1)).^2

Ki = Ki_escolhido

%% Define o sistema de controle
Gc = tf([Kd Kp Ki],[1 0]);

%% define a funcao de transferencia de malha fechada do sistema
   controlado,
%% FTMF
FTMF = minreal(feedback(Gc*GpH,1));

%% mostra os polos e zeros do sistema controlado
disp('SISTEMA CONTROLADO');
[z p k] = zpkdata(FTMF,'v')

%% define o filtro de referencia, sem zeros e com os polos em cima dos
%% zeros do sistema controlado em malha fechada
filtro = zpk([],z,1);
[a b c] = tfdata(filtro,'v');
filtro = filtro * b(3);

%% plota o lugar das raizes do sistema controlado e compara com o polo
%% desejado
rlocus(filtro*FTMF);
title('Lugar das raizes filtroXFTMF');

grid on;hold on;
plot(real_s1,imag_s1,'r+');
hold off;

%% Simulacoes e comparacoes
%%
figure();
step(filtro*FTMF)
```

```
%% Transformada z
%% discretiza a funcao de transferencia e o filtro de referencia
Ts = 1/10;
filtro_z = c2d(filtro,Ts)

%% Equacoes de diferencas
[numerador_filtro denominador_filtro] = tfdata(filtro_z,'v');
```

APÊNDICE C – ESQUEMÁTICO COMPLETO DO SISTEMA



APÊNDICE D – CÓDIGO DA INTERFACE

```
function varargout = UI(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @UI_OpeningFcn, ...
    'gui_OutputFcn',  @UI_OutputFcn, ...
    'gui_LayoutFcn',  [] , ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function UI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of Matlab
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to UI (see VARARGIN)

% Choose default command line output for UI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% This sets up the initial plot - only do when we are invisible
% so window can get raised using UI.
if  strcmp(get(hObject,'Visible'),'off')
    plot(rand(5));
end
global testing_flag;
testing_flag = 0;

contents = get(handles.popupmenuSerialCom,'String');
s1 = serial(contents{get(handles.popupmenuSerialCom,'Value')});
disp(s1)
set(s1, 'baudrate', 57600);
```

```
set(s1, 'parity', 'N');
set(s1, 'stopbits', 1);
setGlobalSerial(s1);

function varargout = UI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes on selection change in popupmenuModoDeOperacao.
function popupmenuModoDeOperacao_Callback(hObject, eventdata, handles)
contents = get(hObject,'Value');
if contents == 5%MANUAL
    set(handles.uipanelCteValue,'Visible','Off');
    set(handles.uipanelManual,'Visible','On');
    set(handles.sliderManual,'Value',0);
    set(handles.text11,'String','0');
else
    set(handles.uipanelCteValue,'Visible','On');
    set(handles.uipanelManual,'Visible','Off');
end

function popupmenuModoDeOperacao_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editPeso_Callback(hObject, eventdata, handles)
disp('editPeso_Callback')

function editPeso_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editDist_Callback(hObject, eventdata, handles)
disp('editDist_Callback')

function editDist_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushButtonCalibrar_Callback(hObject, eventdata, handles)
disp('pushButtonCalibrar_Callback')
fprintf(getGlobalSerial,'$Calibrar=%s;$;',get(handles.editPeso,'String')
```

```
    );
% printfGlobalSerial('$Calibrar;$;');

function pushbuttonZero_Callback(hObject, eventdata, handles)
disp('pushbuttonZero_Callback')
printfGlobalSerial('$Zero;$;');

function sliderManual_Callback(hObject, eventdata, handles)
set(hObject,'Value',round(get(hObject,'Value')))
if(globalSerialIsConnected)
    global testing_flag;
    disp(testing_flag);
    fprintf(getGlobalSerial,'$Man=%d;$;',get(hObject,'Value'));
    set(handles.text11,'String',num2str(get(hObject,'Value')));
else
    set(hObject,'Value',0);
    set(handles.text11,'String','0');
end

function sliderManual_CreateFcn(hObject, eventdata, handles)
if isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function text11_CreateFcn(hObject, eventdata, handles)

function textRotacao_CreateFcn(hObject, eventdata, handles)

function pushbuttonTesteIniciar_Callback(hObject, eventdata, handles)
disp('iniciar teste')
global testing_flag;
if globalSerialIsConnected
    cla();
    global h
    h = animatedline('Marker','.');
    axes(handles.axes1);
    testing_flag = 1;
    set(hObject,'Enable','off');
    set(handles.pushbuttonTesteParar,'Enable','on');
    setGlobal_n(1);

    monitora_serial(handles,0);

    set(hObject,'Enable','on');
    set(handles.pushbuttonTesteParar,'Enable','off');
```

```
    global dado_vect
    if(getGlobal_n > 1)%recebeu os dados na forma de vetor
        rot_vect = dado_vect(1,1:getGlobal_n-1);
        T_vect = dado_vect(2,1:getGlobal_n-1);
        entrada = dado_vect(4,1:getGlobal_n-1);

        tempo_vect = (0:1:getGlobal_n-2)/10;

        figure();hold on
        plot(tempo_vect,rot_vect);
        plot(tempo_vect,entrada);
        figure();
        plot(tempo_vect,T_vect,'r');

        button = questdlg('Deseja salvar os dados do teste?', ...
            'USB dialog','Yes','No','Yes');

        switch button
            case 'Yes',
                salva_teste();
            case 'No',
                ;
        end
    end
else
    testing_flag = 0;
    set(hObject,'Enable','on');
    set(handles.pushbuttonTesteParar,'Enable','off');
end

function pushbuttonTesteParar_Callback(hObject, eventdata, handles)
global testing_flag;
testing_flag = 0;
disp('Parar');

function pushbuttonTesteSalvar_Callback(hObject, eventdata, handles)
salva_teste
message(sprintf('Dados do teste salvos em \"%s\"\n',file_name));
disp('Salvar')

function pushbuttonTesteCarregar_Callback(hObject, eventdata, handles)
disp('Carregar nao implementado')

function editCteValue_Callback(hObject, eventdata, handles)
if(globalSerialIsConnected)

    fprintf(getGlobalSerial,'$Rot=%s;$;',get(hObject,'String'));
```

```
    set(hObject,'String',NaN);
else
    set(hObject,'String',NaN);
end

function editCteValue_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function checkboxAtivado_Callback(hObject, eventdata, handles)
if (globalSerialIsConnected())
    if(get(hObject,'Value')==1)
        fprintf(getGlobalSerial,'$MODE=%d;$;',get(handles.
   popupmenuModoDeOperacao,'Value'));
        set(handles.editKi,'Enable','off')
        set(handles.editKd,'Enable','off')
        set(handles.editKp,'Enable','off')
        set(handles.editVmax,'Enable','off')
        set(handles.popupmenuModoDeOperacao,'Enable','off')
    else
        fprintf(getGlobalSerial,'$MODE=0;$;');
        set(handles.editKi,'Enable','on')
        set(handles.editKd,'Enable','on')
        set(handles.editKp,'Enable','on')
        set(handles.editVmax,'Enable','on')
        set(handles.popupmenuModoDeOperacao,'Enable','on')
    end
else
    set(hObject,'Value',0);
end

function pushbuttonConectar_Callback(hObject, eventdata, handles)
disp('Botao Conectar');

contents = get(handles.popupmenuSerialCom,'String')
s1 = serial(contents{get(handles.popupmenuSerialCom,'Value')});%variavel
    local%variavel local
set(s1, 'baudrate', 57600); % Change baudrate
set(s1, 'parity', 'N');    % Changes parity checking (config becomes 5-E
   -1),
set(s1, 'stopbits', 1);    % Changes stop bits (config becomes 5-E-2),
   possible
setGlobalSerial(s1);

while 1
```

```
    try
        disp('Conectando...');
        fopenGlobalSerial();
        break; %sai do while da conexao
    catch
        button = questdlg('porta serial nao detectada, tentar novamente?
   ', ...
            'USB dialog','Yes','No','No');
        switch button
            case 'Yes',
                disp('Tentando Conectar novamente.');
            case 'No',
                disp('Conexao nao estabelecida.');
                break;
        end
    end
end
if(globalSerialIsConnected)
    set(handles.textConexaoEstabelecida,'Visible','on');
    set(hObject,'Enable','off');
    set(handles.pushbuttonDesconectar,'Enable','on');
    disp('Conectado.');
    pause(1);
    flushinputGlobalSerial();
    pause(1);
    flushinputGlobalSerial();
    disp('Sincronizando...');
    printfGlobalSerial('$Sincronismo;$;');

    monitora_serial(handles,1);
end

function editKp_Callback(hObject, eventdata, handles)
if(globalSerialIsConnected)
    fprintf(getGlobalSerial,'$Kp=%s;$;',get(hObject,'String'));
    set(hObject,'String',NaN);
else
    set(hObject,'String',NaN);
end

function editKp_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editKd_Callback(hObject, eventdata, handles)
```

```
if(globalSerialIsConnected)
    fprintf(getGlobalSerial,'$Kd=%s;$;',get(hObject,'String'));
    set(hObject,'String',NaN);
else
    set(hObject,'String',NaN);
end

function editKd_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function editKi_Callback(hObject, eventdata, handles)
if(globalSerialIsConnected)
    fprintf(getGlobalSerial,'$Ki=%s;$;',get(hObject,'String'));
    set(hObject,'String',NaN);
else
    set(hObject,'String',NaN);
end

function editKi_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function checkboxAtivado_ButtonDownFcn(hObject, eventdata, handles)

function popupmenuSerialCom_Callback(hObject, eventdata, handles)

function popupmenuSerialCom_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function pushbuttonEscanear_Callback(hObject, eventdata, handles)

function setGlobal_n(value)
global n
n = value;

function r = getGlobal_n
global n
r = n;
```

```
function r = incrementGlobal_n
global n
n = n+1;
r = n;

function setGlobalSerial(s1)
global s_global
s_global = s1;
disp(s1)

function printfGlobalSerial(msg)
if(globalSerialIsConnected)
    fprintf(getGlobalSerial,msg);
end

function s1 = getGlobalSerial
global s_global
s1 = s_global;

function fopenGlobalSerial
fopen(getGlobalSerial);

function fcloseGlobalSerial
fclose(getGlobalSerial);

function flushinputGlobalSerial
flushinput(getGlobalSerial);

function r = fscanfGlobalSerial
s = getGlobalSerial;
if(s.BytesAvailable > 0)
    r = fscanf(s);
else
    r = '';
end

function r = globalSerialIsConnected
s = getGlobalSerial;
if strcmp(s.status, 'open')
    r = 1;
else
    r = 0;
end

function figure1_CloseRequestFcn(hObject, eventdata, handles)
global testing_flag
testing_flag = 0;
```

```
fcloseGlobalSerial
pause(1)
delete(hObject);

function editTempoMaximo_Callback(hObject, eventdata, handles)

function editTempoMaximo_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function salva_teste
file_name = datestr(now);
file_name(end-2)='_';%%colocando '_' no lugar de ':', que nao e
    suportado para nome
file_name(end-5)='_';
file_name = [file_name '.xlsx'];
global dado_vect

if(getGlobal_n > 1)
    tempo_vect = (0:1:getGlobal_n-2)/10;
    disp(size(tempo_vect'))
    disp(size(dado_vect(:,1:getGlobal_n-1)'))
    xlswrite(file_name,[[getGlobal_n,5,0,0,0];[tempo_vect' dado_vect
    (:,1:getGlobal_n-1)']]);
    fprintf('Dados do teste salvos em \"%s\"\n',file_name);
    message(sprintf('Dados do teste salvos em \"%s\"\n',file_name));
else
    disp('Nao ha dados a serem salvos.');
end

function monitora_serial(handles,testing)
disp('iniciando monitoramento serial')
disp(getGlobalSerial)
if(globalSerialIsConnected)
    flushinputGlobalSerial();
    dado=[NaN,NaN,NaN,NaN];
    Ki = NaN;
    Kd = NaN;
    Kp = NaN;
    Rot = NaN;
    MODE = NaN;
    Man = NaN;
    Vmax = NaN;
    txt = '';
```

```
    fs = 100;
    t_max = str2double(get(handles.editTempoMaximo,'String'));
    n_max = 60*fs*t_max;

    global h
    global testing_flag;
    global dado_vect;
    dado_vect = zeros(4,n_max);%para t_max

    while (globalSerialIsConnected &&(testing || testing_flag))
        data = fscanfGlobalSerial();
        if(~isempty(data))
            txt = [txt char(data)];
            start_bit = strfind (txt, '$');
            end_bit = strfind(txt, ';');
            i = length(start_bit);
            while i > 0
                if(~isempty(end_bit))
                    if(end_bit(1) > start_bit(1))
                        STR = (txt(start_bit(1)+1:end_bit(1)));
                        try
                            eval(STR);
                        catch
                            disp('ERROR in evaluation string');
                            disp(STR);
                        end
                        if(~isnan(dado(1)))
%                           disp(STR);
                            set(handles.textRotacao,'String',num2str(
   dado(1)));
                            set(handles.textTorque,'String', num2str(
   dado(2)/1000,'%.2f'));

                            if(testing_flag == 1)
                                dado_vect(:,getGlobal_n) = dado;
                                if(getGlobal_n > n_max)
                                    testing_flag = 0;
                                end
                                if(mod(getGlobal_n,1)==0)%para nao
   mostrar todos os pontos...
                                    addpoints(h,getGlobal_n,dado(1));
                                    if(getGlobal_n > 200)
                                        xlim([(getGlobal_n-200)
   getGlobal_n])
                                    else
                                        xlim([0 200])
                                    end
```

```
                                drawnow limitrate
                            end
                            disp(getGlobal_n);
                            incrementGlobal_n;
                        end

                        dado=[NaN,NaN,NaN,NaN];
                    end
                    if(~isnan(Vmax))
                        set(handles.editVmax,'String',num2str(Vmax))
;
                        Vmax = NaN;
                        disp(STR);
                    end
                    if(~isnan(Ki))
                        set(handles.editKi,'String',num2str(Ki));
                        Ki = NaN;
                        disp(STR);
                    end
                    if(~isnan(Kd))
                        set(handles.editKd,'String',num2str(Kd));
                        Kd = NaN;
                        disp(STR);
                    end
                    if(~isnan(Kp))
                        set(handles.editKp,'String',num2str(Kp));
                        Kp = NaN;
                        disp(STR);
                    end
                    if(~isnan(Rot))
                        set(handles.editCteValue,'String',num2str(
Rot));
                        Rot = NaN;
                        disp(STR);
                    end
                    if(~isnan(MODE))
                        if (MODE == 0)
                            set(handles.checkboxAtivado,'Value',0);
                            set(handles.popupmenuModoDeOperacao,'
Enable','on');
                            set(handles.editKi,'Enable','on')
                            set(handles.editKd,'Enable','on')
                            set(handles.editKp,'Enable','on')
                            set(handles.editVmax,'Enable','on')
                            set(handles.popupmenuModoDeOperacao,'
Enable','on')
                        else
```

```
                                set(handles.checkboxAtivado,'Value',1);
                                set(handles.popupmenuModoDeOperacao,'
    Value',MODE)
                                set(handles.popupmenuModoDeOperacao,'
    Enable','off');
                                set(handles.editKi,'Enable','off')
                                set(handles.editKd,'Enable','off')
                                set(handles.editKp,'Enable','off')
                                set(handles.editVmax,'Enable','off')
                                set(handles.popupmenuModoDeOperacao,'
    Enable','off')
                            end
                            MODE = NaN;
                            disp(STR);
                        end
                    end
                    txt = txt(end_bit(1)+1:end);
                else
                    break;
                end
                start_bit = strfind (txt, '$');
                end_bit = strfind(txt, ';');
                i = length(start_bit);
            end
        end
        pause(0.01);
    end
end

function pushbuttonDesconectar_Callback(hObject, eventdata, handles)
fcloseGlobalSerial
set(handles.textConexaoEstabelecida,'Visible','off');
set(hObject,'Enable','off');
set(handles.pushbuttonConectar,'Enable','on');
set(handles.editKi,'String','NaN');
set(handles.editKd,'String','NaN');
set(handles.editKp,'String','NaN');
set(handles.editVmax,'String','NaN');
disp('Desconectado.')

function pushButtonSincronizar_Callback(hObject, eventdata, handles)
if(globalSerialIsConnected)
    printfGlobalSerial('$Sincronismo;$;');
    disp('Sincronizando...');
end

function pushButtonSincronizar_ButtonDownFcn(hObject, eventdata, handles
```

```
    )

function pushButtonSincronizar_KeyPressFcn(hObject, eventdata, handles)

function pushbuttonTeste1_Callback(hObject, eventdata, handles)
if(strcmp(get(hObject,'String'),'Teste'))
    disp(get(hObject,'String'));
    set(hObject,'String','Teste2')
    while(1)
        pause(1)
    end
else
    disp(get(hObject,'String'));
    set(hObject,'String','Teste')
end

function editVmax_Callback(hObject, eventdata, handles)
if(globalSerialIsConnected)
    fprintf(getGlobalSerial,'$Vmax=%s;$;',get(hObject,'String'));
end
set(hObject,'String',NaN);

function editVmax_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of Matlab
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of
    slider


% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of Matlab
% handles    empty - handles not created until after all CreateFcns
    called
```

```
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

APÊNDICE E – BIBLIOTECA DO BMP280

```
//adress
#define BMP180_ADRESS     0x77

//registers
#define BMP180_TEMPERATURE_REQ  0x2E
#define BMP180_PRESSURE_REQ   0x34
#define BMP180_CTRL_MEAS_REG  0xF4
#define BMP180_RESULT_REG   0xF6

void bmp180GetParameters(uint16 *parameters);
void bmp180RequestTemp(void);
void bmp180RequestPress(void);
void bmp180GetResult(uint16 *ui16_res);
uint32 bmp180CalculeTrueTemp(uint16 *parameters,uint16 UT);

void bmp180RequestTemp(){
  uint8 ui8_twi_msg[2];
  ui8_twi_msg[0] =BMP180_CTRL_MEAS_REG;
  ui8_twi_msg[1] =BMP180_TEMPERATURE_REQ;
  twiMasterSendData(BMP180_ADRESS, TWI_MASTER_WRITE,(uint8*)&ui8_twi_msg
    , 2);
}

void bmp180RequestPress(){
  uint8 ui8_twi_msg[2];
  ui8_twi_msg[0] =BMP180_CTRL_MEAS_REG;
  ui8_twi_msg[1] =BMP180_PRESSURE_REQ;
  twiMasterSendData(BMP180_ADRESS, TWI_MASTER_WRITE,(uint8*)&ui8_twi_msg
    , 2);
}

void bmp180Config(uint8 *ui8_new_config){
  uint8 ui8_twi_msg[2];
  ui8_twi_msg[0] = BMP180_CTRL_MEAS_REG;
  ui8_twi_msg[1] = ui8_new_config[0];
  twiMasterSendData(BMP180_ADRESS, TWI_MASTER_WRITE,ui8_twi_msg, 2);
}

void bmp180GetParameters(uint16 parameters[11]){
  uint8 adress,i,value;
  uint8 *temp;
  for (i=0,adress = 0xAA;adress<=0xBF;adress=adress+2,i++){
    twiMasterSendData(BMP180_ADRESS, TWI_MASTER_WRITE,&adress, 1);
```

```
    twiMasterSendData(BMP180_ADRESS,TWI_MASTER_READ,(uint8*)(parameters+
   i),2);
    temp = (uint8*)(parameters+i);//recebe
    value = temp[0];
    temp[0]=temp[1];
    temp[1] = value;
  }
}

void bmp180GetResult(uint16 *ui16_res){
  uint8 *ui8_res = (uint8*)ui16_res;

  uint8 ui8_twi_msg = BMP180_RESULT_REG;
  twiMasterSendData(BMP180_ADRESS, TWI_MASTER_WRITE,&ui8_twi_msg, 1);//
   aponta para o registrador a ser lido

  twiMasterSendData(BMP180_ADRESS,TWI_MASTER_READ,ui8_res,2);//pega os
   dois bytes do resultado da conversao, estao LSB e MSB trocados
  ui8_twi_msg = ui8_res[0];//pega a parte de baixo
  ui8_res[0] = ui8_res[1];
  ui8_res[1] = ui8_twi_msg;
}


void bmp180CalculeTrueTempAndPress(uint16 *parameters,uint16 UT,uint16
   UP,uint32 *T,uint32 *P){
  uint16 AC4,AC5,AC6;
  int16 AC1,AC2,AC3,/*MB,*/MC,MD,B1,B2;
  int32 X1,X2,X3,B3,B5,B6,i32_UT,p;
  uint32 B4,B7;
  i32_UT = UT;
  AC1 = parameters[0];
  AC2 = parameters[1];
  AC3 = parameters[2];
  AC4 = parameters[3];
  AC5 = parameters[4];
  AC6 = parameters[5];
  B1 = parameters[6];
  B2 = parameters[7];
  //MB = parameters[8];
  MC = parameters[9];
  MD = parameters[10];

  X1=(i32_UT - AC6)*AC5/(32768);
  X2 = (int32)MC*2048/(X1+MD);
  B5 = X1+X2;
```

```
  *T = (B5 + 8)/(16);

  B6=B5-4000;
  X1=(B2*(B6*B6/4096))/2048;
  X2=AC2*B6/2048;
  X3=X1+X2;
  B3=(((AC1*4+X3))+2)/4;
  X1 = AC3*B6/8192;
  X2=(B1*(B6*B6/4096))/65536;
  X3=((X1+X2)+2)/4;
  B4=AC4*(uint32)(X3+32768)/32768;
  B7=((uint32)UP-B3)*(50000);
  if(B7 < 0x80000000){
    p = (B7*2)/B4;
  }
  else {
    p=(B7/B4)*2;
  }
  X1=(p/256)*(p/256);
  X1=(X1*3038/65536);
  X2=(-7357*p)/65536;
  p = p+(X1+X2+3791)/16;
  *P = p;
}
```