UNIVERSIDADE FEDERAL DE OURO PRETO
ESCOLA DE MINAS
COLEGIADO DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO - CECAU

ANA CAROLINA CAMPOS DE ABREU LIMA

MODELAGEM E CONTROLE DE VELOCIDADE DE UM MOTOR CC POR MEIO DE CONTROLADORES PROJETADOS POR DIFERENTES MÉTODOS

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2022

ANA CAROLINA CAMPOS DE ABREU LIMA

MODELAGEM E CONTROLE DE VELOCIDADE DE UM MOTOR CC POR MEIO DE CONTROLADORES PROJETADOS POR DIFERENTES MÉTODOS

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Profa. Dra. Adrielle de Carvalho Santana

Coorientador: Prof. Dr. Ronilson Rocha

Ouro Preto
Escola de Minas – UFOP
2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

L732m Lima, Ana Carolina Campos De Abreu.
Modelagem e controle de velocidade de um motor CC por meio de controladores projetados por diferentes métodos. [manuscrito] / Ana Carolina Campos De Abreu Lima. - 2022.
91 f.: il.: color., gráf., tab.. + Equações. + Códigos de programação.

Orientadora: Profa. Dra. Adrielle de Carvalho Santana.
Coorientador: Prof. Dr. Ronilson Rocha.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Controle e Automação .

1. Controle de processo. 2. Motores elétricos de corrente contínua. 3. Laplace, Transformadas de. 4. MATLAB (Programa de computador). 5. Arduino (Controlador programável). I. Rocha, Ronilson. II. Santana, Adrielle de Carvalho. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 681.5

Bibliotecário(a) Responsável: Cristiane Maria Da Silva - CRB6-3046

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Ana Carolina Campos de Abreu Lima

Modelagem e Controle de Velocidade de um Motor CC por meio de Controladores Projetados por Diferentes Métodos

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de bacharel em Engenharia de Controle e Automação

Aprovada em 21 de Dezembro de 2022

Membros da banca

Dra. Adrielle de Carvalho Santana - Orientadora (DECAT - Universidade Federal de Ouro Preto)
Dr. Ronilson Rocha - Coorientador (DEMEC - Universidade Federal de Ouro Preto)
Dr. Bruno Nazário Coelho - Convidado (DECAT - Universidade Federal de Ouro Preto)
Dr. Agnaldo José da Rocha Reis - Convidado (DECAT - Universidade Federal de Ouro Preto)

Adrielle de Carvalho Santana, orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 22/12/2022

Documento assinado eletronicamente por **Adrielle de Carvalho Santana**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 22/12/2022, às 20:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0449275** e o código CRC **6766C7F2**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

"Afinal, se tu choras por ter perdido o sol,
as lágrimas te impedirão de ver as estrelas."

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter direcionado meu caminho ao longo deste percurso e por ter permitido com que eu chegasse até aqui.

Agradeço aos meus pais, por todo o apoio, segurança e força que me deram durante todo esse tempo, me dando ânimo para continuar seguindo em frente.

Agradeço ao meu irmão, pelos incentivos e pelas conversas descontraídas que me permitiram relaxar em alguns momentos.

Agradeço a minha orientadora, Adrielle, por todo o carinho, suporte, ajuda, incentivo e confiança depositados em mim durante esse momento, ajudando a tornar a jornada um pouco mais leve.

Agradeço aos professores, Agnaldo e Bruno, por concordarem em compartilhar dessa experiência comigo na banca de defesa.

Agradeço a todas as amizades que fiz ao longo do curso que contribuíram de alguma forma para o meu desenvolvimento e bem-estar.

Agradeço aos meus tios, primos e avós, por todo o amparo e segurança oferecidos nos momentos de maior necessidade.

“Nós somos feitos de poeira de estrelas.” (Carl Sagan)

RESUMO

O presente trabalho consiste na criação de um sistema de controle com componentes de baixo custo que, futuramente, poderá auxiliar aos discentes em aulas práticas das disciplinas relacionadas à teoria de controle. Para isso, é construído um circuito planta com um motor de corrente contínua (CC), uma ponte H e um sensor de velocidade do tipo *encoder*, assim como um circuito controlador, baseado em Arduino, para controlar o funcionamento desta força motora. Junto aos circuitos é utilizado um *software* de simulação, o MATLAB, para implementar alguns controladores do tipo Proporcional Integral Derivativo (PID) e controladores pelo método do Lugar das Raízes, de forma que se consiga visualizar e estudar a influência deles no desempenho do motor. Antes de serem testados na prática, os controladores são inicialmente simulados em *software* para o conhecimento prévio de como eles devem agir quando aplicados à planta. Com o sistema completo (planta e controlador) é possível aumentar as aplicações práticas da teoria nas disciplinas de controle, uma vez que elas apresentam elevada carga teórica e escassez de carga prática.

Palavras-chaves: Controle de processo. Transformadas de Laplace. MATLAB (Programa de computador). Motores elétricos de corrente contínua. Arduino (Controlador programável).

ABSTRACT

The present work consists in the creation of a control system with low-cost components that, in the future, will help students in practical classes of control theory disciplines. For this, a plant circuit is built with a direct current (DC) motor, a H-bridge and an encoder speed sensor, as well as an Arduino-based controller circuit to control the operation of this motor force. Together with the circuits is used a simulation software, the MATLAB, to implement some controllers of the Proportional Integral Derivative (PID) type and Root Locus, so that it is possible to visualize and study their influence in the engine performance. Before being tested in practice, the controllers are initially simulated in software for prior knowledge of how they should act when applied to the plant. With the complete system (plant and controller) it is possible to increase the practical applications of theory in control disciplines, since they present a high theoretical content and a scarcity of practical content.

Key-words: Process control. Laplace transforms. MATLAB. Direct current electric motors. Arduino.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CC	Corrente contínua
PID	Proporcional Integral Derivativo
PI	Proporcional Integral
PWM	Pulse Width Modulation
RPM	Rotações por minuto
rad/s	Radianos/segundo
K_p	Ganho proporcional
K_i	Ganho integral
K_d	Ganho derivativo
T	Tempo de amostragem
T_i	Tempo integral
T_d	Tempo derivativo
R_a	Resistência de armadura
I_a	Corrente de armadura
V_a	Tensão de armadura
K	Ganho do sistema
b	Coeficiente de atrito mecânico
J	Momento de inércia
L_a	Indutância de armadura
D	Ciclo de trabalho
T_r	Tempo de subida
T_s	Tempo de acomodação
OS	Overshoot

LISTA DE SÍMBOLOS

ζ	Coeficiente de amortecimento
ω	Velocidade
Ω	Ohms

SUMÁRIO

1 INTRODUÇÃO

1.1 Formulação do Problema

O estudante, ao ingressar na Universidade, está passando por um momento de transição na maneira como ele próprio lida com a forma de construir o conhecimento. Cada um apresentará, naturalmente, diferentes formas de aprendizado e de absorção dos novos conteúdos, devendo isso ser levado em consideração pelos professores das disciplinas.

Nos cursos de Engenharia da Universidade Federal de Ouro Preto (UFOP), a partir do quarto semestre, o estudante começa a ter um contato mais aprofundado com as disciplinas mais específicas do curso, o que, de certa forma, requer que os professores tenham a possibilidade de se adaptar à nova realidade dos alunos, uma vez que estes já não são mais aqueles ingressantes advindos do Ensino Médio. É nesse momento que o discente tende a começar a se sentir desmotivado com os estudos, tendo em vista que muitas ou todas as disciplinas cursadas até aí são teóricas e com alto nível de abstração. “Para que serve isso que estou aprendendo?” é a pergunta comumente feita pelos educandos durante esse começo.

Uma das possíveis soluções a essa situação é a substituição da pedagogia tradicional por uma abordagem mais dinâmica e ativa, que não veja o aluno apenas como um mero estudante, mas que o coloque na posição de coautor do seu próprio aprendizado. O discente deve ter a ciência de que ele é apto a possuir uma maior independência no seu processo de aquisição de conhecimento.

É nesse contexto que se fazem necessárias as novas metodologias de aprendizagem ativa que, segundo Valente (2013), devem ser utilizadas para induzir os alunos a assumirem uma postura mais ativa no âmbito educacional. Nessa situação, os próprios estudantes devem resolver problemas, desenvolver projetos e solucionar desafios por conta própria, sendo o professor apenas o orientador e motivador da ação educativa. Aqui, os discentes são os autores da sua própria aprendizagem (COLL, 2006).

O uso das metodologias supracitadas propicia o desenvolvimento do ensino prático no meio acadêmico das Universidades. De acordo com Goodwin et al. (2011), a literatura sugere que a prática é um meio fundamental para complementar a base teórica, bem como para desenvolver novas teorias, uma vez que, segundo Felder (1993), trabalhar com componentes tateáveis apoia muito mais a aprendizagem ativa e sensorial dos alunos.

No ambiente educacional das engenharias, conforme apresenta Goodwin et al. (2011), há três formas comuns de se introduzir a experiência prática aos estudantes: simulação, emulação e uso de *hardwares* de testes físicos. Em meio a essa vertente, como uma forma de contribuir com a melhoria do aprendizado nas disciplinas da área de controle, por meio do uso de metodologias

de aprendizagem ativa, apresenta-se neste trabalho um sistema de controle composto por dois circuitos eletrônicos - circuito planta e circuito controlador - de baixo custo, que possibilitarão a simulação de sistemas de controle, por meio do uso de componentes eletrônicos e *softwares*.

O circuito planta consiste basicamente em um motor de corrente contínua (CC) associado a uma lógica de acionamento baseado em ponte H e um sensor de velocidade do tipo *encoder*. Já o circuito controlador é o responsável por comandar o funcionamento do motor, sendo composto essencialmente por uma placa Arduino®, que é definida como um eletrônico acessível que oferece opções de *hardware e software open-sources* para simulações interativas (UYANIK; CATALBAS, 2018).

Os *softwares* de computação numérica e simulação como o MATLAB (The MathWorks Inc., 1994-2022a) ou o SCILAB® também foram utilizados neste projeto, para a implantação de alguns controladores do tipo PID (Proporcional Integral Derivativo), e controladores projetados pelo método do Lugar das Raízes aprendidos nas disciplinas de controle do curso.

Vale ressaltar que, além da implementação dos controladores, o sistema de controle completo (circuito controlador + circuito planta) foi inicialmente simulado em software para o conhecimento prévio da resposta que se espera obter. Posteriormente, esse sistema foi implementado nos circuitos propriamente ditos para a obtenção da resposta "real", e que foi comparada com a resposta "ideal" anteriormente obtida.

A planta deste trabalho contém um motor CC sendo controlado, mas, espera-se que futuramente, por ser um sistema modular, outros circuitos possam ser criados com o intuito de intensificarem o ensino direcionado à carga prática.

1.2 Objetivos gerais e específicos

1.2.1 Objetivo geral

O presente trabalho tem como objetivo construir e testar circuitos eletrônicos microcontrolados, visando a implementação dos diversos controladores vistos nas disciplinas da área de controle.

1.2.2 Objetivos específicos

- Montar a planta e realizar os ensaios necessários para o levantamento dos parâmetros do motor.
- Implementar uma malha de controle fazendo uso do auxílio de *softwares*, como o MATLAB® ou o SCILAB®, para simular o sistema de controle e compará-lo, posteriormente, com seu comportamento real.

- Testar o sistema com controladores vistos nas diferentes disciplinas do curso para avaliar sua utilidade e adaptabilidade em diferentes projetos.

1.3 Justificativa do trabalho

As disciplinas da área de controle têm tido uma carga horária altamente teórica, contando apenas com alguns programas de simulação, como o MATLAB® e o SCILAB®, para oferecer alguma "prática" simulada aos alunos. Dessa forma, tais disciplinas, as quais possuem um conteúdo teórico pesado, acabam sendo consideradas como complexas pelos alunos, já que apenas a teoria, sem o "mão na massa", em algum momento se torna entediante e difícil de compreender.

Essa concepção pode ser validada ao se observar os trabalhos de conclusão de curso dos estudantes de Engenharia de Controle e Automação da UFOP, que, além de serem a maioria voltados a outras áreas do curso, aparecem em uma quantidade muito menor, quando se trata da área de controle. Segundo UFOP (2022), em um estudo feito entre os anos de 2018 e agosto de 2022, esta quantidade corresponde a 43 Trabalhos de Conclusão de Curso de um total de 155 Trabalhos, o que deixa evidente a argumentação acima.

Como uma forma de melhorar o aprendizado na área, o sistema de controle proposto possibilita aos professores criarem projetos práticos e acessíveis que garantem que os variados controladores aprendidos possam ser facilmente implementados e testados em um sistema real. Sendo assim, justifica-se a sua utilidade por oferecer proposta de trabalho prático que auxiliam a diversificação das metodologias de ensino disponíveis para os professores da área em questão.

1.4 Estrutura do trabalho

O presente trabalho está disposto em 6 unidades textuais. A introdução apresenta uma breve introdução aos principais conceitos abordados neste projeto, assim como a formulação do problema a ser resolvido, os objetivos e a justificativa do trabalho. Já o capítulo 2 contém a revisão bibliográfica para conhecimento dos componentes e softwares utilizados ao longo do desenvolvimento do método, além de um estudo mais aprofundado sobre aprendizagem ativa no meio acadêmico. No capítulo 3, é exibida a metodologia utilizada no projeto, consistindo na apresentação dos materiais e dos métodos utilizados. O capítulo 4 apresenta o desenvolvimento da bancada didática, detalhando a metodologia utilizada e as etapas de execução do projeto. Os principais resultados e testes realizados estão expostos no capítulo 5. Por fim, o capítulo 6 apresenta as conclusões e as recomendações para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para se ter clareza e objetividade acerca dos assuntos que tangem este trabalho é necessário ter, primeiramente, uma fundamentação teórica e científica muito bem formulada dos procedimentos metodológicos que originam os resultados e as conclusões abaixo descritos. Tal processo aconteceu por meio da realização de diversas pesquisas e estudos relacionados a temas como aprendizagem ativa, transformada Z, componentes eletrônicos e sistemas de controle, em especial controladores PID e controladores projetados pelo método do Lugar das Raízes.

2.1 Aprendizagem Ativa

Com o desenvolvimento da educação e da globalização ao longo dos anos, novas estratégias inovadoras de aprendizagem passaram a ser adotadas com o objetivo de oferecer mais autonomia aos educandos. Além de impactar o papel do docente transformando-o, em lugar de um mero transmissor de conhecimento, em um orientador de estudos que facilita o aprendizado, as metodologias de aprendizagem ativa impactam acima de tudo o aluno, que passa a assumir o papel de protagonista do seu próprio aprendizado (SILVA, 2013).

Segundo Silva (2013), "o foco passa a ser o diálogo com os alunos, a sondagem de conhecimentos prévios e percepções sobre o tema em questão com incidência na problematização, contextualização e aplicação prática dos conhecimentos".

Com isso, diversas estratégias vem sendo utilizadas por escolas e educadores para direcionar o educando ao contexto prático, desafiando-o com problemas simulados ou, até mesmo, reais. Isso permite que o aluno "empregue os conhecimentos adquiridos de forma holística, minimizando a ocorrência de uma educação fragmentada"(FARIAS; MARTIN; CRISTO, 2015).

Diversos são os métodos ativos de aprendizagem, e, segundo Cecy, Oliveira e Costa (2013), para que sejam considerados bons métodos para aplicação prática, eles devem ser: críticos, de forma a estimular o aluno a se aprofundar no assunto e conhecer os limites das informações que chegam até ele; colaborativos, favorecendo o desenvolvimento do conhecimento em grupo; interdisciplinares, de maneira que as atividades possam ser integradas a outras disciplinas; investigativos, instigando a autonomia e a curiosidade do educando; contextualizados, para que fique claro e entendível o real objetivo da aplicação deste conhecimento na prática; construtivistas, reflexivos, humanistas, motivadores, e acima de tudo desafiadores, com o objetivo de fazer com que o aluno se sinta estimulado a buscar cada vez mais soluções para os problemas.

A ferramenta proposta neste trabalho consiste na criação de um sistema de controle de baixo-custo para auxiliar na execução de metodologias ativas na área de controle de sistemas, em especial em aulas de disciplinas que hoje são totalmente teóricas.

Alguns exemplos são o *Peer Instruction*, o Ensino Baseado em Problemas (EBP), a Problematização, a Aprendizagem Baseada em Equipes (ABE), a Aprendizagem Baseada em Projetos (ABP), a Taxonomia de Bloom, dentre outros.

2.2 Controle de Sistemas

Segundo Ogata (2010), o controle automático é essencial em qualquer campo da engenharia e da ciência, sendo desejável que a grande maioria dos profissionais esteja familiarizada com a teoria e a prática do assunto.

Controlar significa medir o valor da variável controlada do sistema e aplicar uma grandeza modificada pelo controlador, o sinal de controle ou variável manipulada, para corrigir ou limitar os desvios do valor medido, assim como estabelecer como o sistema deve funcionar, a partir de um valor desejado (*setpoint*) (OGATA, 2010).

No contexto de controle, existem diversos controladores que podem ser aplicados a sistemas com o intuito de exercer a função anteriormente descrita, devendo ser devidamente projetados antes. No projeto de um controlador, é recomendável que existam alguns parâmetros que possam influenciar o desempenho do sistema, uma vez que é por meio do acompanhamento deles que o controle será feito. Tais parâmetros devem ser levados em consideração para que um bom método de ajuste possa ser aplicado (FACCIN, 2004).

Diversos são os métodos de ajuste (ou de sintonia) existentes, que variam conforme o modelo do processo e a necessidade da aplicação do controlador, como por exemplo os métodos de *Ziegler-Nichols, Cohen-Coon, Chien-Hrones-Reswick*, Lugar das Raízes, entre outros.

2.2.1 Controlador PID

O controlador PID, acrônimo para Proporcional-Integral-Derivativo, é um eficiente e potente controlador que mistura as vantagens destes três outros reguladores para garantir sua boa usabilidade. Graças a isso, ele é certamente a estratégia de controle mais amplamente utilizada nos últimos anos, estando presente em mais de 90% das malhas de controle com realimentação (KNOSPE, 2006).

Segundo Åström e Hägglund (2001), o controlador PID pode ser aplicado a uma enorme variedade de problemas, sejam eles relacionados a controle de processos, controle de motores, controles aeronáuticos, instrumentação, automotivo, ou outros.

Os três termos que compõem o PID, de acordo com Knospe (2006), satisfazem os requisitos comuns e necessários da maioria dos problemas de controle, apesar de cada um apresentar desvantagens. O controlador Proporcional responde de forma imediata ao erro atual para diminuí-lo, porém não consegue alcançar o *setpoint* desejado sem um ganho grande, o que torna o sistema mais oscilatório e possivelmente mais instável. O controlador Integral é capaz de anular o erro ao longo do tempo, mas resulta em uma resposta lenta do sistema. Já o controlador

Derivativo consegue prever o erro futuro e atuar o corrigindo antes que ele se torne elevado, porém, ao mesmo tempo, amplifica o ruído do sensor de maior frequência.

Como demonstra Unbehauen (2009), considerando uma representação em diagrama de blocos, um sistema de controle com PID pode ser apresentado conforme a Figura 1.

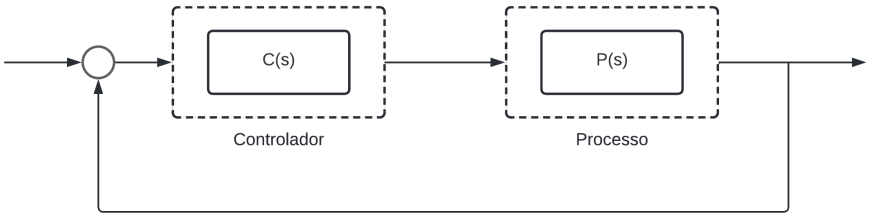


Figura 1 – Sistema de controle convencional com PID.
Fonte: (UNBEHAUEN, 2009).

O bloco de Processo consiste no objeto que será controlado pelo PID que está no bloco Controlador. Representado como o compensador C(s), a função de transferência do PID é dada por:

$$C(s) = K_p + \frac{K_i}{s} + K_d \,.\, s \tag{2.1}$$

onde os parâmetros K_p, K_i e K_d correspondem, respectivamente, ao ganho proporcional, ao ganho integral e ao ganho derivativo que vêm dos três termos, e estão relacionados a partir das equações 2.2, 2.3 e 2.4.

$$K_p = K_c \tag{2.2}$$

$$K_i = \frac{K_p}{T_i} \tag{2.3}$$

$$K_d = K_p T_d \tag{2.4}$$

Em situações reais, muitas vezes, o termo derivativo do controlador PID é substituído por um filtro na tentativa de amenizar, de forma eficiente, o ruído do sinal e evitar que ele influencie na dinâmica do regulador (SILVA; DATTA; BHATTACHARYYA, 2005).

Dessa forma, a função de transferência do PID também pode ser representada como:

$$C(s) = K_p + \frac{K_i}{s} + \frac{K_d s}{1 + T_d s} \tag{2.5}$$

onde T_d é um valor positivo pequeno que geralmente é fixado entre os valores 0,05 e 0,17 (CASTRUCCI, 1969).

2.2.2 Lugar das Raízes

Mais conhecido como Método do Lugar das Raízes, esse método, segundo Ogata (2010), possibilita que as raízes da equação característica sejam apresentadas graficamente para todos e qualquer valor de ganho K do sistema. Dessa forma, as raízes referentes a um valor específico do ganho conseguem ser estabelecidas no gráfico resultante.

Em resumo, o Lugar das Raízes mostra a mudança na resposta transitória de um sistema de malha fechada ao se variar o ganho K da malha, uma vez que a característica básica dessa resposta está profundamente relacionada à localização dos polos (raízes) dentro da malha (OGATA, 2010).

Além de ser um método bastante eficiente para a obtenção rápida de resultados próximos ao desejado, seu verdadeiro poder está na sua capacidade de resolver e fornecer soluções para sistemas de ordem superior a 2 (NISE, 2012).

De acordo com Nise (2012), além do comportamento da resposta transitória, o lugar geométrico das raízes também fornece uma representação gráfica satisfatória da estabilidade do sistema, já que podem ser visualizadas as condições que fazem com que um sistema entre em oscilação e as faixas de estabilidade e de instabilidade. Um Lugar das Raízes representado inteiramente no semiplano esquerdo do plano de Argand-Gauss implica em um sistema estável, enquanto que no semiplano direito, tem-se um sistema instável.

Considere abaixo o diagrama de blocos de um sistema de controle em malha fechada genérico:

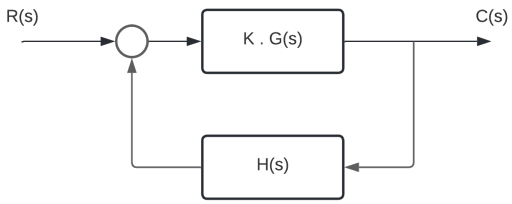


Figura 2 – Sistema de controle de malha fechada.
Fonte: (OGATA, 2010).

A função de transferência em malha fechada para este sistema é:

$$\frac{C(s)}{R(s)} = \frac{K.G(s)}{1 + K.G(s)H(s)} \tag{2.6}$$

A partir dela é possível obter a equação característica do sistema, uma vez que é necessário apenas igualar o denominador da equação a zero, ou seja:

$$1 + K.G(s)H(s) = 0 \tag{2.7}$$

A equação 2.7 pode ser dividida em outras duas equações 2.8 e 2.9 que definem a condição de módulo e de fase (ângulo) do sistema, respectivamente, sendo elas:

$$K = \frac{1}{|G(s)H(s)|} \tag{2.8}$$

$$G(s)H(s) = 180^\circ \pm 360^\circ K \; para \; K = 0, 1, 2, ... \tag{2.9}$$

Os valores de *s* que satisfazem ambas as equações são as raízes da equação característica, ou, como dito anteriormente, os polos de malha fechada desse sistema (GOMES, 2009).

O projeto de um controlador utilizando o Lugar das Raízes será melhor detalhado na seção 4.9.

2.3 Métodos de Sintonia de Controladores PID

Um método de sintonia consiste em selecionar parâmetros do controlador que influenciem a performance do sistema e utilizá-los para garantir uma determinada especificação de desempenho (FACCIN, 2004).

Segundo Faccin (2004), vários métodos de sintonia já foram propostos ao longo da história, sendo eles classificados como analíticos, empíricos ou, até mesmo, obtidos por meio de algum tipo de otimização, além de também existirem métodos baseados em processos que operam em malha aberta, em malha fechada ou ainda no domínio da frequência.

Nas sessões a seguir serão discutidos, de forma mais detalhada, alguns métodos de sintonia clássicos e mais comumente utilizados no projeto de controladores PID.

2.3.1 Método de Sintonia de Ziegler-Nichols

Um método bastante conhecido e utilizado é o de Ziegler-Nichols, que sugere regras para o ajuste dos valores de K_p, T_i (tempo integral) e T_d (tempo derivativo) que proporcionam uma operação estável do sistema. Contudo, é importante ressaltar que essas regras não fornecem os valores definitivos desses parâmetros logo na primeira tentativa, apenas estimativas de valores que proporcionam um ponto de partida na sintonia fina (OGATA, 2010).

Segundo Ogata (2010), a determinação dos parâmetros do controlador é baseada nas características da resposta transitória da planta na qual está sendo aplicado o controle.

Existem dois métodos de sintonia de Ziegler-Nichols: o Método da Curva de Reação e o Método da Sensibilidade Limite, ambos se fundamentando em determinar algumas características da dinâmica do processo (PINTO, 2014).

A principal vantagem do primeiro método se comparado ao segundo é que, uma vez determinada a curva de reação, os parâmetros podem ser ajustados de imediato sem ter a

necessidade de esperar o sistema atingir a estabilidade crítica. No entanto, a desvantagem é que em muitos casos os sistemas de controle são bem mais complexos, o que acaba requerendo a necessidade de um último ajuste no ganho antes de considerar a resposta aceitável (LOURENÇO, 1997).

2.3.1.1 1º Método: Método da Curva de Reação

Desenvolvido por J. G. Ziegler e N. B. Nichols, ambos da *Taylor Instrument Companies*, em 1942, este método se resume em aplicar uma entrada em degrau unitário à planta para obter experimentalmente, ou a partir de uma simulação dinâmica, a sua resposta em malha aberta. Se o método for aplicável à sintonia, então a curva de resposta ao degrau de entrada irá possuir o aspecto de um S. A curva com esse formato permite determinar duas constantes: o atraso, ou tempo de retardo, *L*, e a constante de tempo de resposta do sistema *T* (OGATA, 2010).

De acordo com Pinto (2014), para determinar as constantes, basta traçar um linha tangente ao ponto de inflexão da curva e determinar a interseção da linha tangente com o eixo do tempo (abscissa) e a linha, quando a saída apresenta-se constante (*K*), conforme apresenta a Figura 6. Com isso, a função de transferência da planta pode ser aproximada por um sistema de primeira ordem com atraso, como mostra a equação 2.10.

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{2.10}$$

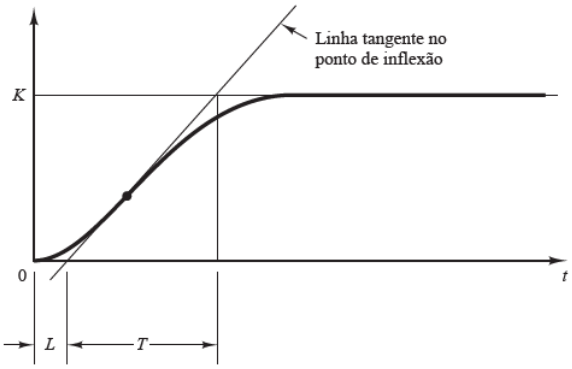


Figura 3 – Curva de resposta da planta em malha aberta.
Fonte: (OGATA, 2010).

Com os valores de *L* e *T* encontrados e dependendo do tipo de controlador a ser trabalhado, basta substituir estes valores nas fórmulas da Tabela 1 proposta por Ziegler e Nichols para escolher as melhores medidas de K_p, T_i e T_d.

Segundo Gomes (2009), é importante ressaltar que o controlador PID sintonizado pelo primeiro método resulta sempre em uma função de transferência com um polo na origem e dois zeros reais e iguais em $s = -\frac{1}{L}$, segundo a equação 2.11.

$$C(s) = 0,6T\frac{(s+\frac{1}{L})^2}{s} \tag{2.11}$$

Tabela 1 – Sintonia pelo primeiro método de Ziegler-Nichols.

Controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0,9\frac{T}{L}$	$\frac{L}{0,3}$	0
PID	$1,2\frac{T}{L}$	2L	0,5L

Fonte: (OGATA, 2010).

2.3.1.2 2º Método: Método da Sensibilidade Limite

Também desenvolvido por J. G. Ziegler e N. B. Nichols em 1942, o segundo método consiste em definir $T_i = \infty$ e $T_d = 0$ e utilizar somente a ação de controle proporcional K_p do sistema para obter, em malha fechada, o ganho crítico K_{cr} e o período P_{cr}. Se o método for aplicável à sintonia, então, ao aumentar o valor de K_p de 0 para o valor crítico K_{cr}, isto é, para o valor de K_p na qual a saída exibe uma oscilação sustentada pela primeira vez, então o K_{cr} e o P_{cr} poderão ser determinados experimentalmente, conforme pode ser visto na Figura 4 (OGATA, 2010).

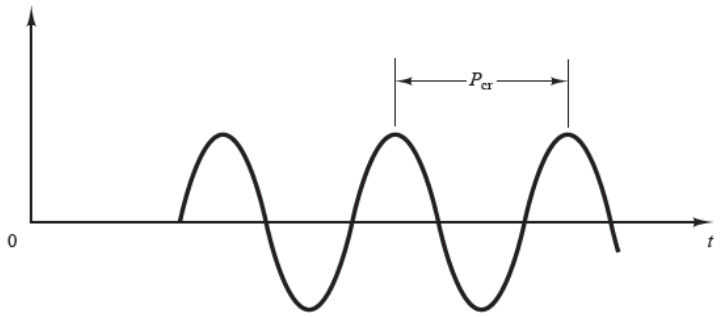


Figura 4 – Oscilação sustentada da saída em malha fechada.
Fonte: (OGATA, 2010).

Com os valores de K_{cr} e P_{cr} encontrados e dependendo do tipo de controlador a ser trabalhado, basta substituí-los nas fórmulas da Tabela 2 proposta por Ziegler e Nichols para escolher as melhores medidas de K_p, T_i e T_d.

Segundo Gomes (2009), novamente é importante ressaltar que o controlador PID sintonizado pelo segundo método resulta sempre em uma função de transferência com um polo na origem e dois zeros reais e iguais em $s = -\frac{4}{P_{cr}}$, segundo a equação 2.12.

$$C(s) = 0,075K_{cr}P_{cr}\frac{(s + \frac{4}{P_{cr}})^2}{s} \tag{2.12}$$

Tabela 2 – Sintonia pelo segundo método de Ziegler-Nichols.

Controlador	K_p	T_i	T_d
P	$0,5K_{cr}$	∞	0
PI	$0,45K_{cr}$	$\frac{1}{1,2}P_{cr}$	0
PID	$0,6K_{cr}$	$0,5P_{cr}$	$0,125P_{cr}$

Fonte: (OGATA, 2010).

2.3.2 Método de Sintonia de Cohen-Coon

Proposto por G. H. Cohen e G. A. Coon, ambos da *Taylor Instrument Companies*, em 1953, este método é uma alternativa ao primeiro método proposto por Ziegler e Nichols que também aproxima a função de transferência da planta a um sistema de primeira ordem com tempo morto (GOMES, 2009), isto é:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{2.13}$$

De acordo com Pinto (2014), uma vez que a sintonia de um controlador pode se diferenciar por causa da existência de um número infinito de modos harmônicos, isto é, de um número infinito de soluções da equação característica fundamental, foi proposto um método que obtivesse os parâmetros adequados (K, L e T), a partir do modo harmônico de menor frequência e maior amplitude.

As fórmulas de sintonia deste método são, na maioria dos casos, bastante agressivas, o que confere um bom desempenho para a rejeição a distúrbios, mas muita oscilação para mudanças no valor da variável de referência (FACCIN, 2004).

A Tabela 3 apresenta a sintonia dos tipos de controladores pelo método de Cohen-Coon.

Tabela 3 – Sintonia pelo método de Cohen-Coon.

Controlador	K_p	T_i	T_d
P	$\frac{T}{KL}(1+\frac{L}{3T})$	0	0
PI	$\frac{T}{KL}(0,9+\frac{L}{12T})$	$L(\frac{30T+3L}{9T+20L})$	0
PID	$\frac{T}{KL}(\frac{4}{3}+\frac{L}{4T})$	$L(\frac{32T+6L}{13T+8L})$	$\frac{4LT}{11T+2L}$

Fonte: (GOODWIN et al., 2001).

2.3.3 Método de Sintonia de Chien-Hrones-Reswick (CHR)

Elaborado por K. L. Chien, J. A. Hrones e J. B. Reswick no *Massachusetts Institute of Technology* em 1952, este foi o primeiro método a utilizar um modelo aproximado de primeira ordem com tempo morto como representação do comportamento de sistemas de alta ordem (FACCIN, 2004).

Assim como o método anterior, este também se trata, segundo Silva, Datta e Bhattacharyya (2005), de uma modificação ao primeiro método proposto por Ziegler e Nichols, sendo baseado em um critério de *design* de 0% de *overshoot*. Tal critério pode ser também conhecido como *resposta mais rápida sem overshoot*.

Fazendo uso do método de Ziegler-Nichols para encontrar os parâmetros L e T do sistema, basta substituí-los nas equações 2.14, 2.15 e 2.16 a seguir para obter os coeficientes do controlador e, consequentemente, projetá-lo.

$$K_p = \frac{0,6}{KT} \tag{2.14}$$

$$K_i = \frac{0,6}{KL} \tag{2.15}$$

$$K_d = \frac{0,3T}{K} \tag{2.16}$$

2.4 Introdução à Transformada Z

A Transformada Z é uma das mais importantes transformadas utilizadas na análise e no *design* de sistemas lineares, garantindo também vantagens significativas aos processos que ocorrem no domínio do tempo. Uma equação transformada é mais fácil de ser resolvida e manipulada (PHILLIPS; PARR; RISKIN, 2008).

Basicamente, segundo Junior e Cruz (2010), esse tipo de transformada converte o domínio de tempo discreto para o domínio da variável z, sendo a alternativa mais adequada quando a variável tempo não é contínua.

A partir de uma forma de onda analógica, cria-se uma sequência *g[n]* de números reais ordenados a partir de valores crescentes de *n*, que será obtida por meio da amostragem do sinal *g(t)* a intervalos regulares Δt (JUNIOR; CRUZ, 2010).

O processo de amostragem de um sinal contínuo no tempo pode ser visto na Figura 5 a seguir.

Considerando, segundo Ogata (1995), a transformada *Z* da função no tempo *g(t)*, apenas os valores amostrados são válidos, isto é, $g(0), g(T), g(2T), \ldots$, em que *T* é o período de amostragem. Dessa forma, para uma sequência de números *g(n)*, onde *n* assume apenas valores inteiros

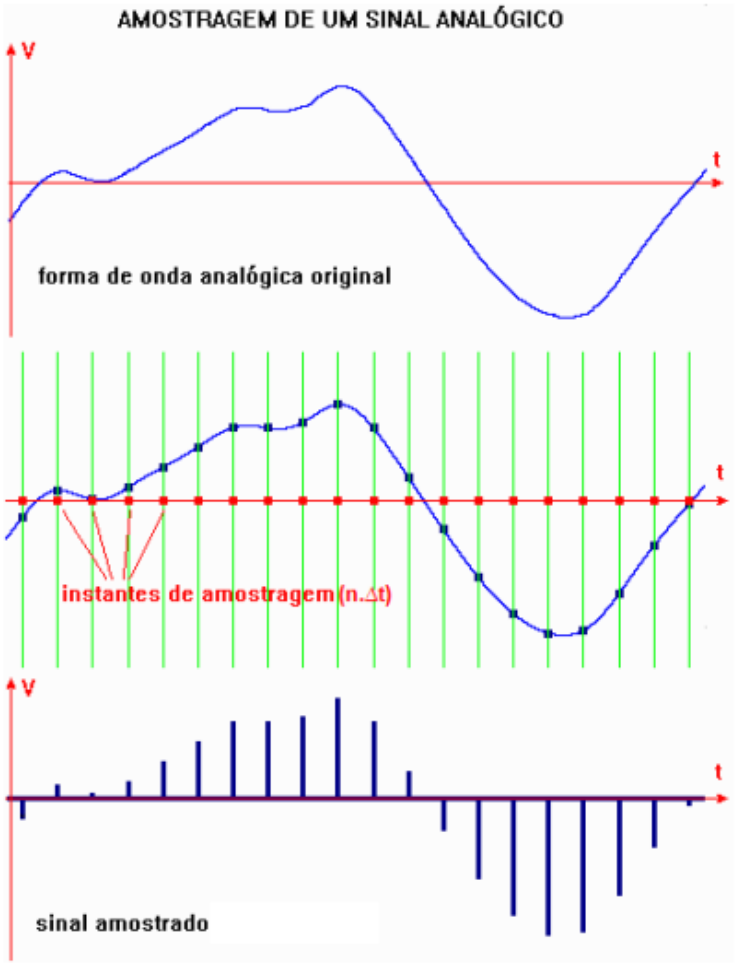


Figura 5 – Processo de amostragem de um sinal contínuo no tempo.
Fonte: (JUNIOR; CRUZ, 2010).

$(n = 0, \pm 1, \pm 2, ...)$, a transformada passa a ser definida por:

$$G(z) = \sum_{n=-\infty}^{\infty} g(n) z^{-n} \tag{2.17}$$

A localização do polo e do zero no plano-z está diretamente relacionada com a localização do polo e do zero no plano-s, permitindo com que a estabilidade do sistema em tempo discreto seja determinada por meio da localização dos polos da função de transferência (OGATA, 1995).

Quando a amostragem é incorporada ao processo, as variáveis complexas z e s estão relacionadas pela equação:

$$z = e^{Ts} \tag{2.18}$$

que indica que um polo no plano-s pode ser localizado no plano-z por meio dessa transformação.

A função de transferência do controlador PID no domínio de Z é dada por:

$$G(z) = K_p + \frac{K_i}{1 - z^{-1}} + K_d(1 - z^{-1}) \tag{2.19}$$

onde

$$K_p discreto = K_p continuo - \frac{K_i discreto}{2} = ganho\ proporcional \tag{2.20}$$

$$K_i discreto = K_i continuo\ *\ T = ganho\ integral \tag{2.21}$$

$$K_d discreto = \frac{K_d continuo}{T} = ganho\ derivativo \tag{2.22}$$

2.5 Levantamento dos parâmetros de um motor CC

Conforme detalha Rocha (2021), o método para estimação dos parâmetros de um motor CC consiste em seis passos, sendo eles:

1. Estimar a resistência de armadura R_a:
   a) Alimentar a armadura do motor com o rotor bloqueado com um sinal PWM;
   b) Medir a corrente média de armadura I_a presente no sistema e a tensão média V_a durante cerca de $3s$;
   c) Repetir os procedimentos anteriores várias vezes para diferentes posições do rotor;
   d) Obter a média da tensão V_a e a da corrente I_a encontradas anteriormente;
   e) Calcular a resistência de armadura R_a com as médias calculadas anteriormente por meio da fórmula da Lei de Ohm:

$$V = R\ .\ I \tag{2.23}$$

$$R_a = \frac{V_a media}{I_a media}\ . \tag{2.24}$$

2. Estimar a constante eletromecânica *K*:
   a) Alimentar a armadura do motor com o rotor livre (a vazio) e uma tensão V_a adequada;
   b) Medir a corrente média de armadura I_a e a velocidade do rotor ω;
   c) Repetir os procedimentos anteriores várias vezes para diferentes valores de PWM aplicados ao motor;
   d) Calcular a força contra-eletromotriz *e* para cada valor de PWM por meio da fórmula:

$$e = V_a - I_a * R_a \tag{2.25}$$

   e) Calcular a constante eletromecânica *K* para cada valor de PWM por meio da fórmula:

$$K = \frac{e}{\omega}\ . \tag{2.26}$$

3. Estimar o coeficiente de atrito mecânico *b*:

   a) Considerando que a potência elétrica no circuito seja igual a potência mecânica, calcular o coeficiente de atrito mecânico *b* para cada valor de PWM escolhido no passo 2, por meio da fórmula:

$$b = \frac{K * I_a}{\omega} \; ; \tag{2.27}$$

   b) Calcular o coeficiente de atrito mecânico *b* médio.

4. Estimar o momento de inércia *J*:

   Este parâmetro deve ser determinado por meio da análise da curva experimental de desaceleração do motor. Obtida a curva, calcular o momento de inércia *J* por meio da fórmula:

$$J = - \frac{b * (t_{37,5\%} - t_0)}{\ln(\frac{\omega_{37,5\%}}{\omega_{100\%}})} \; , \tag{2.28}$$

   em que:

   - $t_{37,5\%}$ é o tempo em que a velocidade de rotação do motor alcançou 37,5% do seu valor inicial;
   - t_0 é o tempo em que a alimentação da armadura foi retirada;
   - $\omega_{37,5\%}$ é a velocidade de rotação do motor ao alcançar 37,5% do seu valor inicial;
   - $\omega_{100\%}$ é a velocidade de rotação máxima que o motor pode atingir considerando um determinado valor de PWM.

5. Estimar a indutância de armadura L_a:

   Fazendo uso de um osciloscópio, calcular a indutância de armadura L_a por meio da fórmula:

$$L_a = - \frac{D * R_a}{f * \ln(1 - \frac{R_a * I_a pico}{V_a pico - K * \omega})} \; , \tag{2.29}$$

   em que:

   - *D* é o ciclo de trabalho;
   - R_a é a resistência da armadura encontrada anteriormente;
   - *K* é a constante eletromecânica encontrada anteriormente;
   - ω é a velocidade de rotação;
   - $I_a pico$ é a corrente de pico;
   - $V_a pico$ é a tensão de pico.

   Recomenda-se que o ciclo de trabalho *D* seja um valor pequeno, de no máximo 20%, de maneira que a força contra-eletromotriz do motor apareça na curva da tensão de armadura na forma de tensão negativa, conforme exemplo da Figura 6. No momento em que essa tensão começar a aparecer, é quando a medição do $V_a pico$ e do $I_a pico$ deve ser realizada.

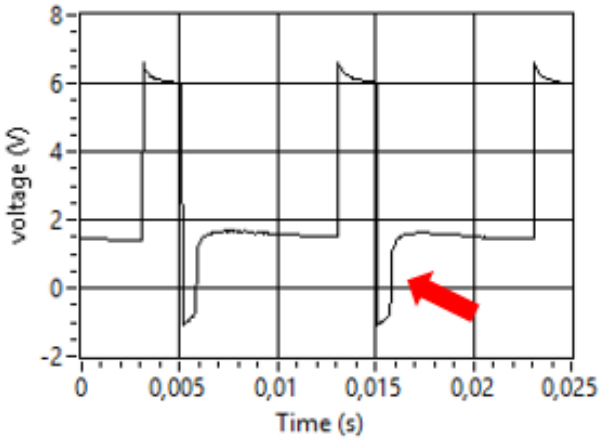


Figura 6 – Forma de onda da tensão na armadura com a força contra-eletromotriz sinalizada. Fonte: (ROCHA, 2021).

6. Encontrar a função de transferência do motor CC:

   A partir dos parâmetros do motor obtidos anteriormente, encontrar a função de transferência do motor por meio da função:

$$G(s) = \frac{\omega(s)}{V_a(s)} = \frac{K_m}{(R_a + L_a s)(Js + f) + K_m K_g}\,, \tag{2.30}$$

   sendo que $K = K_g = K_m$ e $b = f$.

2.6 Trabalhos Relacionados

Para alcançar a solução a ser desenvolvida neste trabalho, além dos outros documentos mencionados anteriormente, fez-se uso de quatro projetos base já outrora desenvolvidos, todos utilizando de microcontroladores de baixo-custo para implementar sistemas físicos e controladores com potencial para serem aplicados tanto de forma didática quanto para a solução de problemas reais. Seguem mais detalhes acerca de cada um deles.

Brito (2016), em seu projeto, faz uso de um sistema do tipo *hardware-in-the-loop* para controlar uma esfera metálica por meio de levitação magnética, utilizando do desenvolvimento de um controlador digital PID na plataforma *Simulink* do *software* Matlab e um *hardware* de baixo-custo Arduino MEGA2560.

Para a execução, foi necessária a criação de dois sistemas físicos: um contendo o eletroímã responsável por gerar o campo magnético que irá atrair a esfera, e o outro contendo um diodo emissor de infravermelho e um fototransistor para detectar a posição vertical da esfera. Entre a conexão do Arduino e do sistema de levitação, fez-se uso de um circuito de potência composto por um MOSFET IRF530 para conseguir amplificar o sinal que sai do microcontrolador e vai para o eletroímã, uma vez que a tensão e a corrente fornecidas pelo Arduino não eram suficientes para alimentar esse último componente.

O funcionamento do sistema é simples: a movimentação da esfera faz com que o fototransistor capte sua posição e envie um sinal analógico, dado em tensão, para o Arduino, que vai realizar a função de uma malha PID com o uso do Simulink. O sinal que sai do microcontrolador é um sinal PWM direcionado para o eletroímã, para que este possa controlar a posição da esfera. Em resumo, o que define a posição da esfera é a maior ou menor incidência da luz no receptor.

O módulo construído pode ser visto na Figura 7 a seguir.

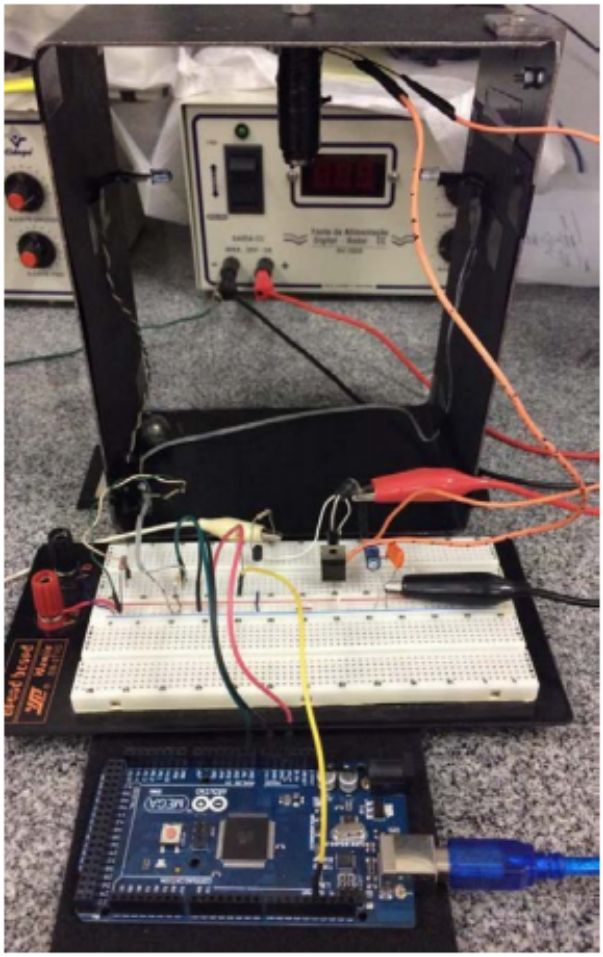


Figura 7 – Módulo de levitação magnética.
Fonte: (BRITO, 2016).

Assim como Brito (2016), Uyanik e Catalbas (2018) também criaram um módulo multiuso com componentes eletrônicos de baixo-custo que pode ser facilmente obtido em grandes quantidades.

O projeto consistiu na criação de uma planta em pequena escala que simula sistemas de controle de realimentação para um motor de corrente contínua (CC), tendo como objetivo seu uso em aulas práticas de Teoria de Controle do Departamento de Engenharia Elétrica e Eletrônica da Universidade de Bilkent, localizada em Ankara, na Turquia. Os autores fizeram uso de um *hardware* Arduino UNO, um Arduino *shield*, um motor de corrente contínua com *encoder* e o *software* Matlab/Simulink. A vantagem de se usar a interface Arduino-Simulink para o controle,

segundo Uyanik e Catalbas (2018), é poder visualizar o funcionamento do motor para cada tipo de controlador em tempo real.

Para simular o bloco da função de transferência no Simulink, com nome de *DC Motor Plant* na Figura 8, que foi fixado no diagrama de blocos, criaram-se dois sub-blocos: o bloco de atuação que converte a tensão de entrada gerada pelo controlador em um sinal de PWM, e o bloco de detecção que mede a velocidade do motor contando a quadratura dos pulsos do *encoder*.

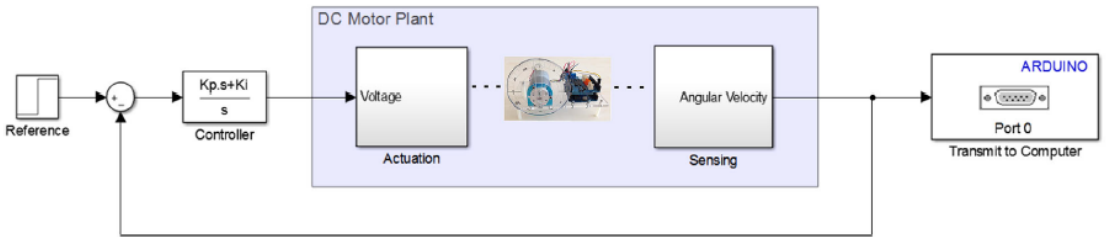


Figura 8 – Diagrama de blocos de um sistema de controle simples com realimentação.
Fonte: (UYANIK; CATALBAS, 2018).

A partir da montagem do sistema, três práticas foram realizadas: o controle da velocidade do motor por meio de um controlador PI, o ajuste dos parâmetros de controle por meio do Lugar das Raízes e a identificação do sistema no domínio da frequência.

Da mesma forma que os autores anteriores, Reguera et al. (2015) propôs um projeto para realizar melhorias em um antigo motor de corrente contínua (CC), de forma que se pudesse realizar, por meio de uma solução barata, o controle dele. Além do controle, o objetivo também foi permitir que antigos sistemas físicos pudessem ser aproveitados e utilizados juntamente com novos sistemas físicos.

O sistema físico construído, o qual foi denominado *Feedback MS-150*, é composto por sete módulos: uma fonte de energia PS150E ±15V, um motor de corrente contínua DCM150F equipado com um servo amplificador SA150D que permite mudar a direção de rotação do motor, um potenciômetro de entrada IP150H para definição do *setpoint*, um potenciômetro de saída OP150K para medir o ângulo de rotação do motor, e um tacômetro GT150X para medir a velocidade de rotação do motor. Para garantir que o *Feedback* pudesse ser usado para qualquer tipo de sinal provindo de qualquer controlador, os autores fizeram uso de um Arduino MEGA2560 e de um cartão adaptador.

A estrutura de controle inicial conecta o Arduino com o cartão adaptador, de forma que ele lê três entradas analógicas provindas dos dados do motor (ângulo de rotação, velocidade de rotação e o *setpoint*) e retorna duas saídas analógicas para o servo amplificador (se saída1 > saída2, o motor rotaciona para a direita, e se saída2 > saída1, o motor rotaciona para a esquerda). A função do cartão adaptador é adaptar as altas tensões que sai do *Feedback* para tensões suportadas pelo Arduino (entre 0-5V), além de permitir a leitura/escrita de sinais analógicos.

Para a coleta dos dados de todo o sistema, novamente realizou-se uma conexão serial

entre o *hardware* Arduino e o *software* MATLAB/Simulink para a implementação de diferentes controladores por meio de diagramas de blocos.

Finalizando o conjunto de artigos utilizados como base, Franco et al. (2021) apresentam, mais uma vez, uma alternativa de baixo-custo para a construção de módulos didáticos a serem aplicados em práticas de controle de nível. Para isso, dispõe-se de um *hardware* Arduino UNO, uma planta didática para controle de nível DCNV1 da Didaticontrol e um sistema supervisório desenvolvido inteiramente no ambiente Simulink do *software* MATLAB.

A comunicação do sistema supervisório com a planta se dá por meio da interface Simulink-Arduino, sendo o controle do circuito realizado por um controlador PID. Por meio de toda essa montagem, além de práticas de controle de sistemas, inúmeras outras práticas para o estudo de controle de processos podem ser realizadas, tais como identificação de sistemas e estudo da linearidade e estudo da técnica anti-*windup* na ação integral do PID.

3 MATERIAIS E MÉTODOS

3.1 Materiais

3.1.1 Ponte H

A ponte H é um dispositivo eletrônico que, a partir de sinais PWM alimentados em seus pinos de entrada, permite estabelecer o sentido de giro de uma carga e controlar a velocidade dela. Além dessa, a ponte H pode ter diversas outras aplicações extremamente úteis, tal como controle da corrente em cargas que puxam muita corrente, criação de um inversor de tensão, criação de um circuito elevador de tensão, entre outras.

A ideia principal do funcionamento de uma ponte H consiste no circuito da Figura 9:

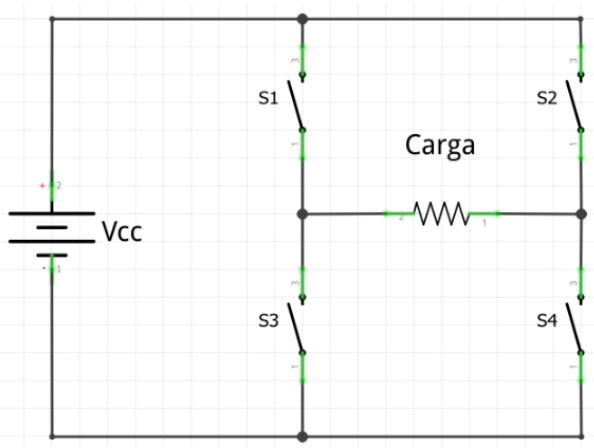


Figura 9 – Funcionamento de uma ponte H.
Fonte: (PROJETADO, 2018).

Quando acionadas as chaves S1 e S4 ao mesmo tempo, o terminal esquerdo do motor fica com uma tensão mais positiva que o direito, fazendo com que a corrente flua da esquerda para a direita da carga (no caso deste projeto, o motor). Já quando acionadas as chaves S2 e S3, o terminal direito fica com uma tensão mais positiva que o esquerdo, fazendo com que a corrente flua da direita para a esquerda da carga. Durante o funcionamento deste circuito, a carga sempre terá uma tensão constante, pois sempre um par de chave estará acionado. Se acionar S1 e S2 ou S3 e S4, haverá um curto na fonte de alimentação e nada acontecerá (PROJETADO, 2017).

Neste projeto, será utilizada a Ponte H L298N visando o controle de um motor de corrente contínua. Apesar de no atual estudo controlar apenas um motor, tal ponte H permite o controle da velocidade e do sentido de rotação de até dois motores, ao mesmo tempo.

Na pinagem (Figura 10), segundo Filipeflop (2013), a ponte H L298N apresenta dois Outputs para as saídas dos motores, sendo um dos outputs destinado à conexão do Motor A e o outro destinado à conexão do Motor B. Os pinos Ativa MA e Ativa MB são os pinos de ativação

para o controle dos motores A e B, respectivamente, através de PWM. O pino 5V serve como um regulador de tensão já integrado à placa, permitindo que outros componentes eletrônicos possam ser alimentados (através de *jumper*) com 5V quando o driver estiver operando entre 6-12V. Já os pinos 6-12V e GND são usados para a conexão da fonte de alimentação externa quando a ponte estiver sendo usada para controlar um motor que opere entre 6-12V.

Ainda, de acordo com Filipeflop (2013), os pinos categorizados como Entrada são os pinos responsáveis pela rotação do Motor A (pinos IN1 e IN2) e do Motor B (pinos IN3 e IN4). IN1 e IN3 são as entradas PWM para o controle da velocidade dos motores A e B, respectivamente, e IN2 e IN4 são as entradas de sinal digital para o controle do sentido de rotação dos motores A e B, respectivamente. Já o pino Ativa 5v serve para acionar o regulador de tensão 5v caso a alimentação da placa seja de 6-12V (VLADCONTROL, 2019).

Tais pinos podem ser vistos na Figura 10 a seguir.

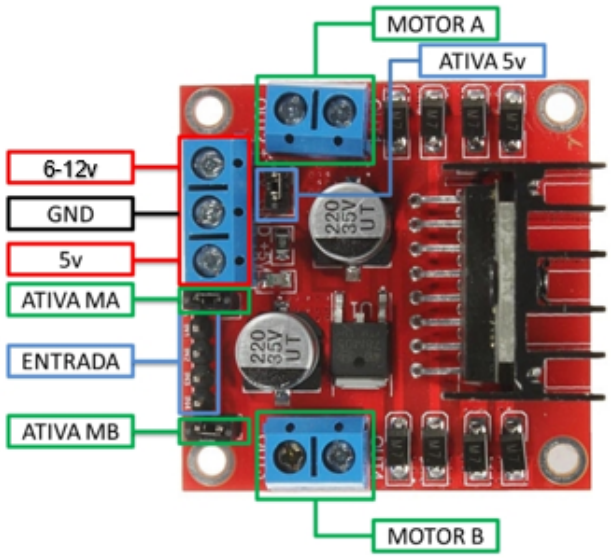


Figura 10 – Pinagem de uma ponte H L298N.
Fonte: (FILIPEFLOP, 2013).

3.1.2 Sensor de Velocidade *Encoder*

Segundo Tecnologia (2017), o *encoder* é um sensor eletromecânico capaz de transformar posição angular em sinal elétrico digital por meio do uso de um disco com marcações, um componente emissor e um receptor. Além de medir a velocidade de rotação de um motor em RPM, ele também pode ser usado para diversas aplicações, tais como contar pulsos, controlar posicionamento, quantizar distâncias, medir ângulos, entre outras.

Quanto aos componentes que podem ser utilizados como o emissor e o receptor, há o *encoder* óptico, que utiliza led como o emissor e um fotodetector como o receptor, e o *encoder* magnético, que utiliza ímãs como o primeiro e sensores de efeito hall em variadas configurações como o segundo (TECNOLOGIA, 2017).

Para este trabalho será utilizado um *encoder* óptico com um chip comparador LM393 e um optointerruptor MOCH22A com abertura para o disco *encoder* de 5mm. A tensão de operação dele varia de 3,3 a 5V e possui tanto pino para saída digital quanto para saída analógica.

De modo geral, o sensor apresenta um led indicador de tensão alimentada, um led indicador de saída digital de dados e quatro pinos no total: os dois do meio sendo para as saídas digital (D0) e analógica (A0), e os dois externos sendo um para a alimentação da tensão (V_{cc}) e o outro para o ground (GND), conforme pode ser visto na Figura 11.

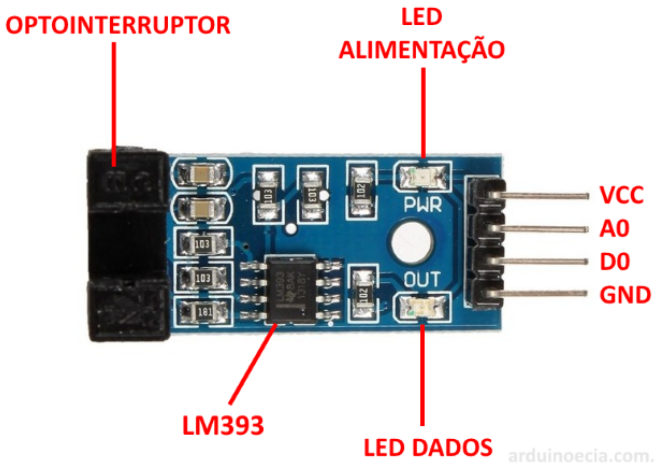


Figura 11 – Componentes de um *encoder* óptico.
Fonte: (CIA, 2016).

O optointerruptor apresenta o led infravermelho em um de seus lados e um fototransistor do outro lado, de forma que o disco fique localizado no meio das duas extremidades. O disco com duas marcações (janelas) fica conectado ao motor e bloqueia ou não o feixe de luz do led.

À medida que o disco gira, a cada pulso (um pulso corresponde à uma marcação), o fototransistor envia para a saída digital do *encoder* um sinal em forma de onda quadrada proporcional ao número de marcações do disco, de modo que, toda vez que o fotodetector não consegue detectar o feixe de luz – não está em uma marcação do disco -, a saída envia o sinal 1 (nível alto – 5V) para o microcontrolador, e, em caso contrário, a saída permanece no nível 0 (nível baixo – 0V) (CIA, 2016).

A forma como o disco funciona pode ser vista na Figura 12 a seguir, e a saída do sinal na Figura 13.

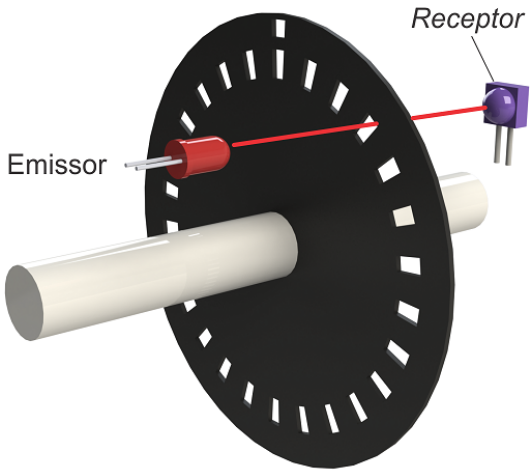


Figura 12 – Como o led e o fototransistor agem no disco do *encoder*.
Fonte: (TECNOLOGIA, 2017).

Figura 13 – Forma de onda da saída do sinal.
Fonte: (TECNOLOGIA, 2017).

Essa leitura dos pulsos precisa, posteriormente, ser tratada via programação para retornar a velocidade de rotação.

3.1.3 Resistor *SHUNT*

O resistor *SHUNT* é uma lógica eletrônica alternativa que permite obter o valor da corrente que passa por uma carga de forma bastante eficiente, principalmente se a tensão fornecida pela carga for baixa. Ele funciona utilizando o valor da queda de tensão que ocorre nele mesmo, dividido pela resistência do próprio *SHUNT* para o cálculo da corrente. Isso pode ser visto por meio do uso da fórmula da Lei de Ohm:

$$V = R * I \tag{3.1}$$

$$I = \frac{V}{R} \tag{3.2}$$

onde *I* é o valor da corrente que passa pelo motor (a que se pretende encontrar), *V* é o valor da queda de tensão no Resistor *SHUNT* e *R* o valor da resistência do Resistor.

É ainda importante lembrar que, para que o cálculo de corrente utilizando um Resistor *SHUNT* ocorra da maneira correta, o Resistor deve ser ligado em série com o motor e deve ter um valor de resistência muito baixo (menor que 1Ω e, de preferência, na casa dos 0,01Ω), de forma que a queda da tensão não seja tão alta a ponto de diminuir a potência máxima que o motor é capaz de fornecer.

Para este trabalho, foi utilizado um resistor de 1,3Ω como o Resistor *SHUNT*.

3.1.4 Arduino

O Arduino é uma plataforma acessível de *hardware* e *software open-sources* que permite a criação de diversos protótipos de projetos eletrônicos, fazendo uso da linguagem *Wiring* (baseada em C/C++) em sua programação. Ele basicamente consiste em uma placa física baseada em um microprocessador de código aberto da indústria ATMEL (TECHTUDO, 2011).

Neste trabalho, foi utilizado o Arduino UNO (Figura 14), baseado no microcontrolador ATmega328 composto por 14 pinos de entrada/saída digital (6 deles podendo ser utilizados como saídas PWM), 6 entradas analógicas, uma conexão de alimentação USB e uma de fonte externa, um cristal de quartzo de 16MHz, conectores de alimentação e um botão de *reset*. Sua tensão de operação é de 5V.

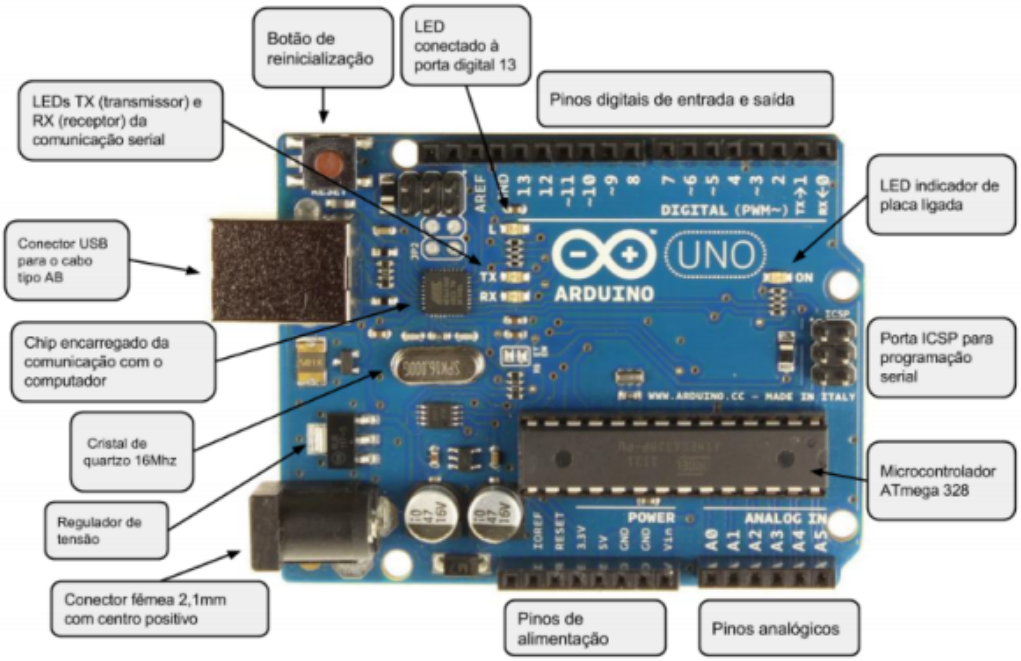


Figura 14 – Arduino UNO e seus componentes.
Fonte: (NATALMAKERS, 2018).

3.1.5 *Software*

3.1.5.1 MATLAB®

Segundo a The MathWorks Inc. (1994-2022c), fabricante do *software*, o MATLAB® é uma plataforma de computação numérica e de programação utilizada para análise de dados, desenvolvimento de algoritmos e criação de modelos. Pode ser empregado em projetos que envolvam controle de sistemas, *deep learning*, processamento de imagens e de sinais, aprendizado de máquina, robótica, comunicação sem fio, dentre outras diversas aplicações.

O *software* integra computação matemática, visualização e linguagens de programação para fornecer, ao máximo possível, um ambiente adaptável ao bom uso do usuário. Além disso,

apresenta muitas funções incorporadas e uma vasta biblioteca de ferramentas pré-construídas que podem ser utilizadas para resolver problemas específicos (ALBAGUL; KHALIFA, 2005).

3.1.5.2 Simulink®

Integrada ao MATLAB® existe a ferramenta de simulação Simulink® que, quando aplicados juntos, combinam o poder da programação textual com o da programação gráfica em um único ambiente. De acordo com The MathWorks Inc. (1994-2022b), o Simulink® é um ambiente de diagrama de blocos utilizado para projetar sistemas multidomínio, simular os programas antes de executar em *hardware*, e implementar programações sem a necessidade de escrever o código.

3.2 Métodos

Em um momento inicial do trabalho, foi necessário realizar uma revisão bibliográfica com artigos e textos acadêmicos disponíveis nas plataformas e na internet, para entender quais materiais, ferramentas e técnicas computacionais poderiam ou deveriam ser utilizados para o controle do motor de corrente contínua, assim como qual seria a melhor maneira de trabalhá-lo em um sistema com Arduino® e MATLAB/Simulink®.

Feito isso, o primeiro passo do projeto foi realizar a montagem física do circuito planta que interliga os materiais eletrônicos mencionados anteriormente. Em seguida, foi executado o levantamento dos parâmetros do motor CC para encontrar sua função de transferência e construir o projeto dos controladores.

Com o circuito planta criado e a função de transferência encontrada, o próximo passo se dividiu em duas etapas, sendo uma para cada controlador:

1ª) Encontrar os parâmetros do PID por meio da sintonia de Ziegler-Nichols pelo método da Curva de Reação, criar os diagramas de blocos no Simulink para a simulação, em *software*, do controlador, e implementá-lo no sistema físico criado.

2ª) Obter o Lugar das Raízes para o sistema e analisá-lo, de forma a se projetar o controlador, criar os diagramas de blocos no Simulink para a simulação, em *software*, do controlador obtido, e implementá-lo no sistema físico criado.

Por fim, com os resultados obtidos, foram comparadas as respostas simuladas dos dois controladores com as respostas obtidas por eles no sistema físico, de forma a analisar se o controle foi corretamente realizado em ambos os casos.

4 DESENVOLVIMENTO

Tendo sido realizado todos os estudos e pesquisas necessários para se ter a fundamentação teórica e científica bem formulada acerca dos assuntos que tangem este trabalho, neste capítulo, serão descritos com detalhes todos os procedimentos metodológicos empregados na execução e validação do projeto, assim como na coleta e obtenção dos dados.

4.1 Montagem do Sistema de Controle

Antes de se realizar qualquer medição ou algum procedimento de levantamento de dados, assim como todo projeto eletrônico, foi necessário primeiramente realizar a montagem do circuito físico que dá origem ao protótipo.

A construção do sistema (controlador + planta) foi feita com os componentes eletrônicos de baixo custo apresentados no capítulo anterior, sendo eles o motor de corrente contínua de 5V, a ponte H L298N, o dispositivo medidor de velocidade do tipo *encoder* LM393, o resistor *SHUNT* de 1,3Ω, o Arduino UNO, alguns outros componentes eletrônicos e o protoboard, todos eles conectados entre si conforme apresentado na Figura 15 a seguir.

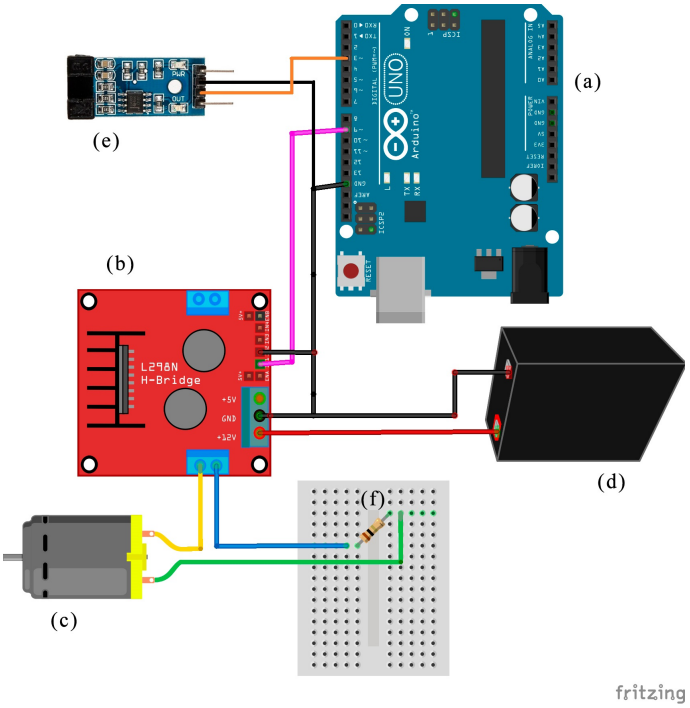


Figura 15 – Esquemático do circuito: (a) Placa Arduino UNO (b) Ponte H L298N (c) Motor CC (d) Fonte externa que alimenta a ponte H (e) Sensor de velocidade *encoder* (f) Resistor *SHUNT*.
Fonte: Acervo pessoal.

O terminal negativo do motor é conectado ao terminal negativo do *Output* da ponte H

onde a tensão é nula, enquanto que o terminal positivo é conectado em uma das extremidades do resistor *SHUNT*. Na outra extremidade do *SHUNT*, é interligado o terminal positivo do *Output* da ponte H.

A entrada *IN1* da ponte H recebe a informação em PWM do pino digital 9 do Arduino, que é quem vai fornecer o comando para o controle da velocidade do motor. Já a entrada *IN2* fica conectada diretamente ao *GND* da placa, pois, para o projeto, não se fez necessário o controle do sentido de rotação do motor. Os pinos 6-12V e *GND* da ponte H são alimentados pela fonte de alimentação externa de 6V.

O *encoder*, apesar de não estar fielmente representado na Figura 15, fica localizado bem próximo do motor, de forma que possa ler as rotações do seu eixo por meio do disco de rotação utilizado. Por testes, o disco escolhido contém dois furos, que foi uma quantidade boa para medir assertivamente e com segurança a velocidade de rotação do motor, levando em consideração que não se deseja saber o sentido de rotação neste trabalho.

Os pinos *GDN* e *VCC* (alimentação) do *encoder* foram conectados, respectivamente, aos pinos *GDN* e 5V do Arduino para a alimentação do dispositivo. O pino digital D0 do *encoder* vai enviar o sinal de rotação para o pino digital 3 do Arduino ao mesmo tempo que o pino analógico A0 envia o sinal para o pino 2. No sistema a conexão do pino analógico A0 do *encoder* no Arduino é realizado apenas por questões triviais, isto é, não é necessária, pois não tem utilidade nesse projeto.

A leitura da tensão no terminal positivo da ponte H é coletada pela entrada analógica A5 do Arduino, enquanto que a corrente entre o motor e o *SHUNT* é lida pela entrada A2 e a tensão no terminal negativo pela entrada A0.

Para cada um dos sinais acima coletados, foi feito um filtro passa-baixo com um capacitor de 10μF e um resistor de 10kΩ para atenuar valores muito altos e tornar os sinais lidos menos variantes e mais estáveis, o que acabou sendo necessário durante a execução do projeto.

Além dos componentes físicos citados foi utilizado também o software MATLAB® e seu complemento simulacional Simulink para o projeto dos controladores, tanto em *software* quanto em *hardware*, e obtenção das respostas simuladas e reais do sistema para cada regulador.

4.2 Levantamento dos parâmetros do motor CC

Cada passo realizado na estimação dos parâmetros do motor foi executado nos componentes eletrônicos dos circuitos físicos construídos, utilizando-se do Arduino para a criação da rotina e a leitura dos dados pelo *Monitor serial*.

Para o primeiro passo, foi programada uma rotina simples de leitura das portas analógicas, uma vez que, nesse primeiro momento, era necessário apenas ler as tensões V_a e as correntes I_a presentes no circuito para diversas repetições do experimento. Considerando aleatoriamente um

sinal PWM de 180 (no Arduino a variação é de 0 a 255) e o bloqueio do rotor de forma manual feito rapidamente durante cada medição, foram realizadas seis repetições sob essas condições utilizando o *script* de código apresentado no Apêndice A. Os valores para cada iteração podem ser vistos na Tabela 4 a seguir.

Tabela 4 – Medições de tensão e corrente para PWM de 180 e rotor bloqueado.

Leitura/Repetição	1	2	3	4	5	6
Va (V)	2,33	2,38	2,43	2,37	2,39	2,33
Ia (A)	0,26	0,24	0,20	0,26	0,23	0,28

Calculando a média das tensões V_a e das correntes I_a, obteve-se:

$$V_amedia = 2,37V \text{ e } I_amedia = 0,24A$$

Portanto, a resistência de armadura R_a estimada é:

$$R_a = 9,67\Omega$$

É importante ressaltar que a leitura nas portas analógicas é inicialmente feita em *bits* (1023), o que torna necessário converter os valores para tensão (5V).

Para o segundo passo foi programada outra rotina de leitura de portas, porém agora do *encoder*, de forma a se conseguir obter também, além de V_a e I_a, a velocidade de rotação do motor em RPM. Considerando o rotor livre (a vazio), foram novamente realizadas seis repetições, no entanto, agora variando o sinal PMW em cada uma, conforme *script* apresentado no Apêndice B. Os resultados para cada iteração estão apresentados na Tabela 5 a seguir.

Tabela 5 – Medições de tensão, corrente e velocidade do rotor para PWM variável e rotor livre.

Leitura/Repetição	1	2	3	4	5	6
PWM	110	130	160	180	210	230
Va (V)	2,64	3,00	3,46	2,37	2,39	2,33
Ia (A)	0,02	0,02	0,04	0,26	0,23	0,28
ω (rad/s)	376,99	439,82	502,65	565,49	628,32	691,15

Calculando a força contra-eletromotriz *e* para cada repetição, obteve-se:

Tabela 6 – Força contra-eletromotriz *e* para PWM variável e rotor livre.

Leitura/Repetição	1	2	3	4	5	6
PWM	110	130	160	180	210	230
e (V)	2,4466	2,8066	3,0732	3,5099	4,0866	4,1899

Calculando a constante eletromecânica *K* para cada PWM, adquiriu-se:

Tabela 7 – Constante eletromecânica *K* para PWM variável e rotor livre.

Leitura/Repetição	1	2	3	4	5	6
PWM	110	130	160	180	210	230
K ($x10^{-3}$)	6,49	6,38	6,11	6,21	6,50	6,06

Dessa forma, realizando a média de todos os valores de *K*, a constante eletromecânica *K* estimada é:

$$K_{medio} = 6,29x10^{-3} = 0,00629 = 0,01$$

Vale salientar que a leitura de velocidade do rotor foi realizada através de RPM, tendo que ser convertida para *rad/s* para que pudesse ser utilizada nas fórmulas.

Em seguida, fez-se uso dos resultados encontrados nos passos anteriores para estimar o coeficiente de atrito mecânico *b* para cada valor definido para o PWM no passo 2, podendo os números serem vistos na Tabela 8 a seguir.

Tabela 8 – Coeficiente de atrito mecânico *b* para PWM variável e rotor livre.

Leitura/Repetição	1	2	3	4	5	6
PWM	110	130	160	180	210	230
b ($x10^{-6}$)	0,344	0,290	0,486	0,329	0,207	0,263

Assim, por meio da média de todos os valores de *b*, o coeficiente de atrito mecânico *b* estimado é:

$$b_{medio} = 0,3198x10^{-6}$$

O próximo passo consistiu em estimar o momento de inércia *J* do motor por meio de sua curva de desaceleração, que pode ser obtida acelerando a máquina até que a velocidade de rotação se estabilize e, logo em seguida, desligando a alimentação da armadura. Para isso, fez-se uso do Simulink.

O PWM escolhido para essa medição foi 51, de forma que o $\omega_{100\%}$ fosse 188,5 *rad/s*. Desse jeito, a curva obtida é apresentada na Figura 16, a seguir.

Considerando que a alimentação da armadura foi retirada no instante $t_0 = 0,6s$ e que a velocidade de rotação alcançou 37,5% ($\omega_{37,5\%} = 127\ rad/s$) de seu valor inicial em $t_{37,5\%} = 0,72s$, o momento de inércia *J* calculado é:

$$J = \frac{-(0,3198x10^{-6}) * (0,72-0,6)}{ln(\frac{127}{188,5})}$$

$$J = 97,18x10^{-9}\ Kg.m^2$$

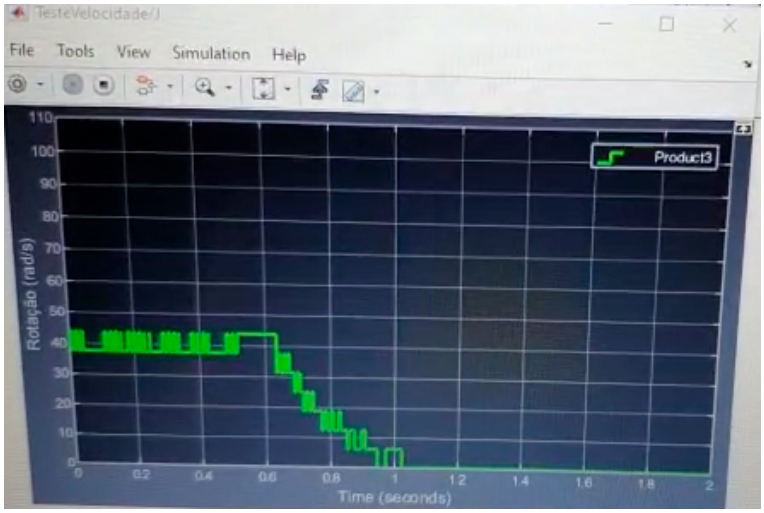


Figura 16 – Curva de desaceleração experimental do motor CC.
Fonte: Acervo pessoal.

Por fim, no último passo, fez-se necessário a utilização de um osciloscópio juntamente com uma ponteira de prova, conectados ao circuito físico criado, para obter os parâmetros da equação 2.30 citada na seção 2.5. Estes equipamentos se encontram no Laboratório de Eletrônica Analógica e Digital da Escola de Minas da UFOP, onde tais testes foram realizados.

Utilizando a ponteira, duas conexões diferentes precisaram ser realizadas: uma sobre o motor para obter o valor da tensão na armadura (V_apico), e a outra sobre o resistor *SHUNT* para obter o valor da corrente de armadura (I_apico). Antes da realização das medições foi necessário configurar o acoplamento do osciloscópio para DC, o fator da ponta de prova para 1x e o canal para a configuração de não invertido.

Segundo Rocha (2021), o ciclo de trabalho *D* tem relação linear com o range de 0 - 255 do PWM no Arduino. Dessa forma, como se desejava trabalhar com o *D* de 20%, bastou calcular 20% de 255 e usar o valor no PWM do Arduino para executar ambas as medições. Esse valor corresponde a 51 de PWM que, quando aplicado no *script* do Apêndice A, resulta em uma velocidade de 1800 RPM e, consequentemente, uma velocidade de rotação de 188,5 *rad/s*.

Na medição da tensão, a ponta da ponteira foi conectada no terminal positivo do motor e a garra "jacaré" no terminal negativo, sendo a escala vertical do osciloscópio diminuída para 2 *V/div*. A forma de onda obtida pode ser vista na Figura 17 a seguir.

Pode-se observar que a tensão negativa decorrente da força contra-eletromotriz está presente na curva, conforme deveria.

Centralizando o cursor do canal e pressionando o botão *Measure* do osciloscópio, obteve-se o valor de pico da tensão V_a, que foi:

$$V_apico = 6,08\ V$$

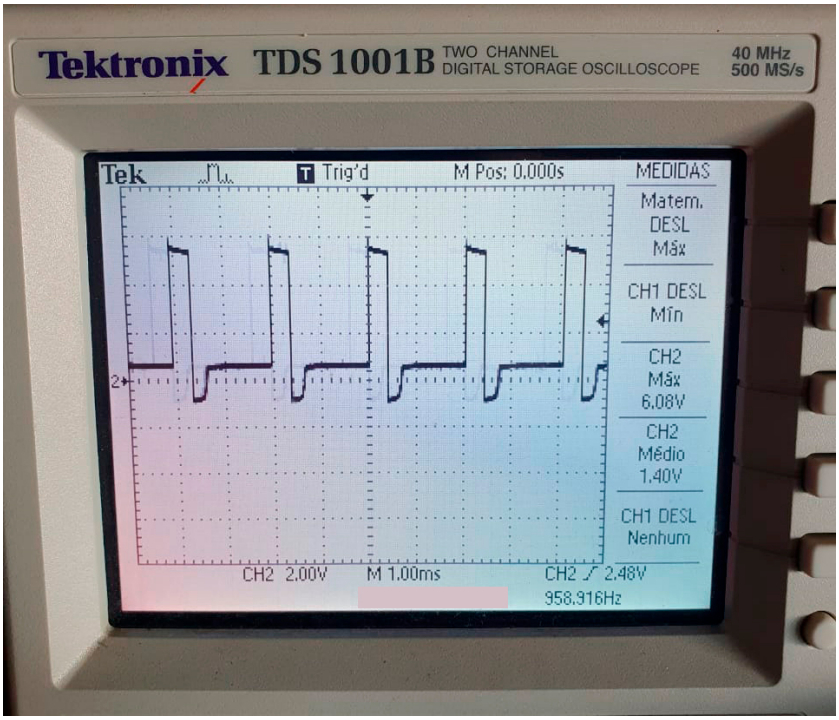


Figura 17 – Forma de onda obtida na medição da tensão.
Fonte: Acervo pessoal.

Já na medição da corrente, a ponta da ponteira foi conectada na perna do resistor que recebe sinal diretamente do motor, e a garra "jacaré" na outra perna (a que transmite o sinal para a ponte H). A forma de onda alcançada pode ser vista na Figura 18 a seguir.

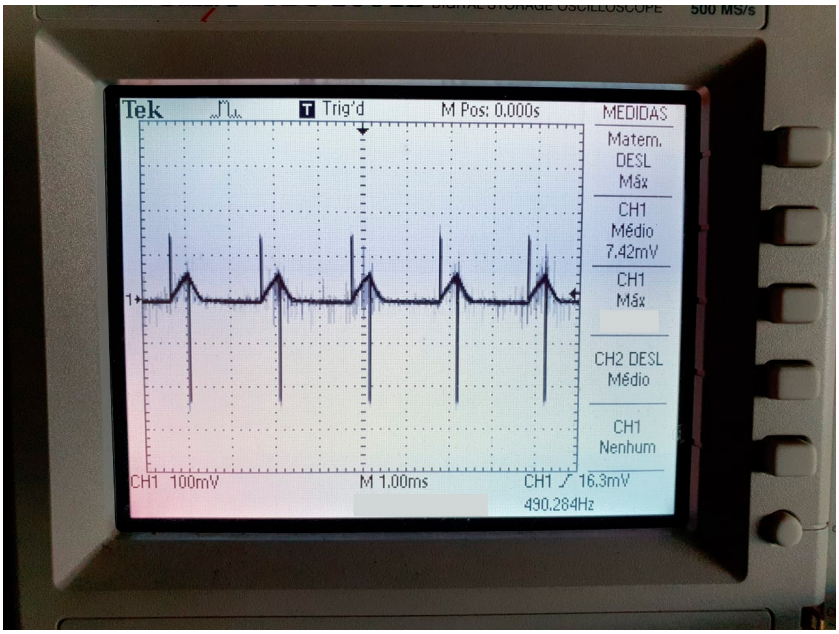


Figura 18 – Forma de onda obtida na medição da corrente.
Fonte: Acervo pessoal.

Dando um *zoom* em uma das ondas da curva, conforme pode ser observado na Figura 19, a tensão de pico obtida foi de 60 *mV*.

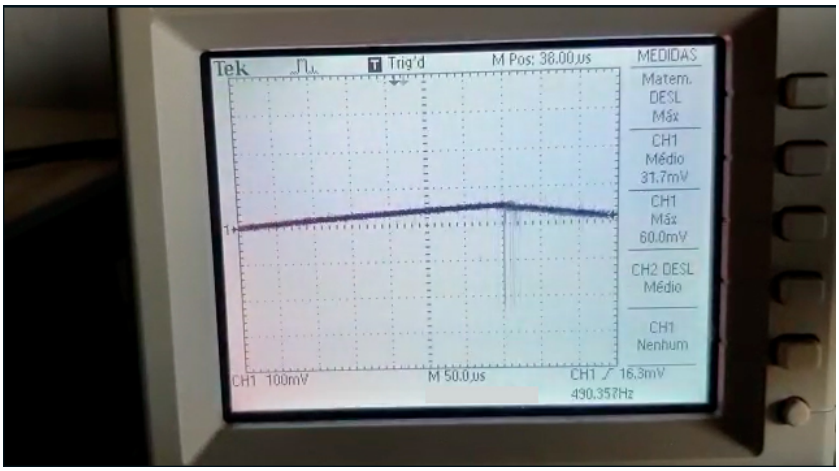


Figura 19 – Forma de onda, com *zoom*, obtida na medição da corrente.
Fonte: Acervo pessoal.

Como se desejava obter a corrente, bastou dividir esse valor pelo valor da resistência do resistor *SHUNT* (1,3Ω), obtendo:

$$I_apico = 0,0461\ A$$

Dessa forma, o cálculo do valor da indutância de armadura L_a resulta em:

$$L_a = \frac{-\ 0{,}2 * 9{,}67}{490 * ln(1 - \frac{9{,}67\ *\ 0{,}0461}{6{,}08\ -\ (6{,}29x10^{-3})\ *\ 188{,}5})}$$

$$L_a = 46,54mH$$

Finalmente, a partir de todos esses parâmetros obtidos nos passos anteriores, a função de transferência do motor utilizado é:

$$\frac{\omega(s)}{V_a(s)} = \frac{6,29x10^{-3}}{(9,67\ +\ 46,54x10^{-3}s)\ (97,18x10^{-9}s\ +\ 0,3198x10^{-6})\ +\ 6,29x10^{-3}\ .\ 6,29x10^{-3}} \tag{4.1}$$

$$\frac{\omega(s)}{V_a(s)} = \frac{6,29x10^{-3}}{4,52x10^{-9}s^2\ +\ 9,55x10^{-7}s\ +\ 4,27x10^{-5}} \tag{4.2}$$

4.3 Configuração do Simulink para trabalhar com o Arduino

Antes que o Simulink pudesse ser utilizado para a obtenção da curva e dos dados foi preciso instalar no MATLAB os pacotes de suporte de *hardware*: o *MATLAB Support Package for Arduino Hardware, o Legacy MATLAB and Simulink Support for Arduino e o Simulink Support Package for Arduino Hardware*, que possibilitam a conexão e o envio e recebimento de sinais entre o *software* e o Arduino.

4.4 Encontrando os parâmetros do PID por meio do 1º Método de Sintonia de Ziegler-Nichols

Com a função de transferência do motor de corrente contínua encontrada anteriormente, o próximo passo foi simular toda a montagem através de diagramas de blocos do Simulink para um sistema de malha aberta. Uma forma prática de realizar a montagem que atenda essa configuração é apresentada por Freitas e Jesus (2012), em seu texto, e pode ser visualizada na Figura 20 a seguir.

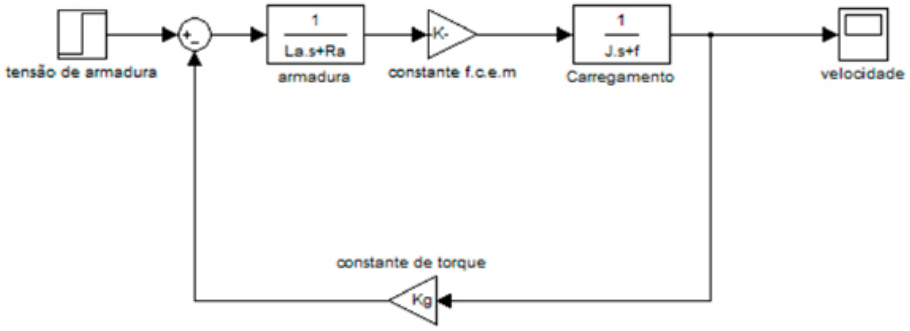


Figura 20 – Diagrama de blocos para o sistema em malha aberta.
Fonte: (FREITAS; JESUS, 2012).

Dessa forma, fazendo uso da equação 4.1, o circuito em malha aberta simulado para o motor CC utilizado neste trabalho é ilustrado na Figura 21.

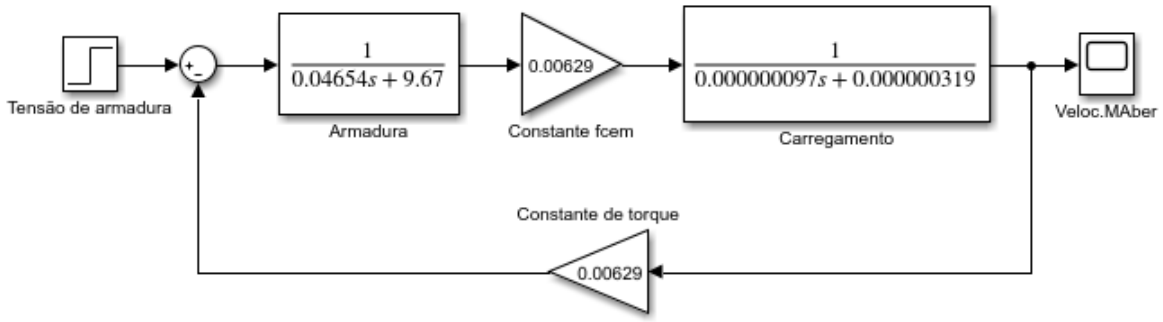


Figura 21 – Diagrama de blocos para o sistema em malha aberta do motor CC utilizado neste trabalho.
Fonte: Acervo pessoal.

Alimentando o sistema com um degrau unitário de valor inicial 0 e valor final 100, e seguindo os conceitos da Figura 4, através da reta tangente traçada sobre a curva de resposta ao degrau, obteve-se o *tempo de retardo* $L = 2,683\ ms$ e a *constante de tempo* $T = 30,317\ ms$, conforme pode ser visto nas Figuras 22 e 23 a seguir.

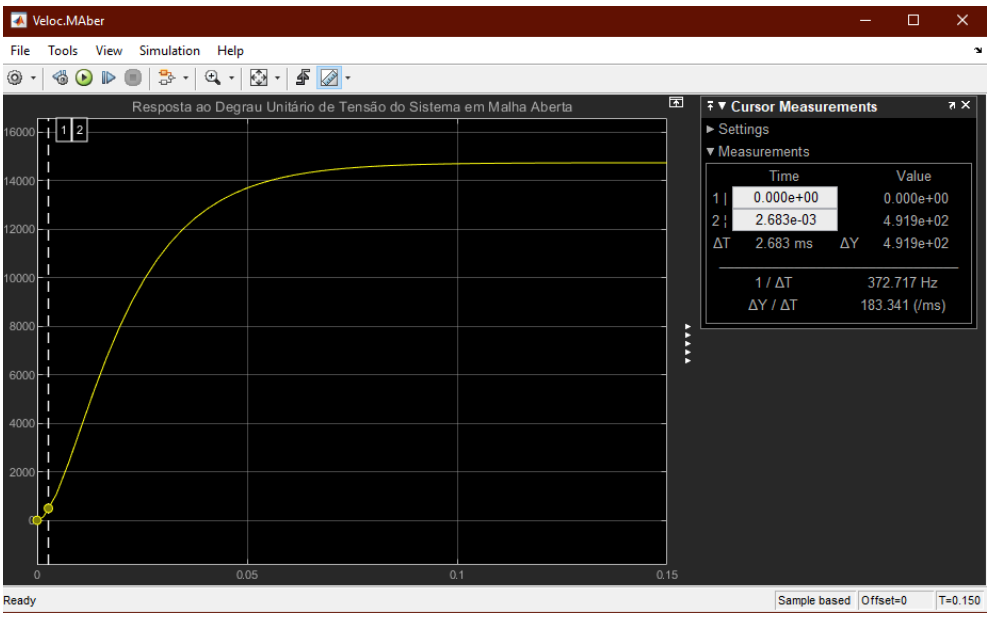


Figura 22 – Tempo de retardo *L* da Curva S do sistema em malha aberta.
Fonte: Acervo pessoal.

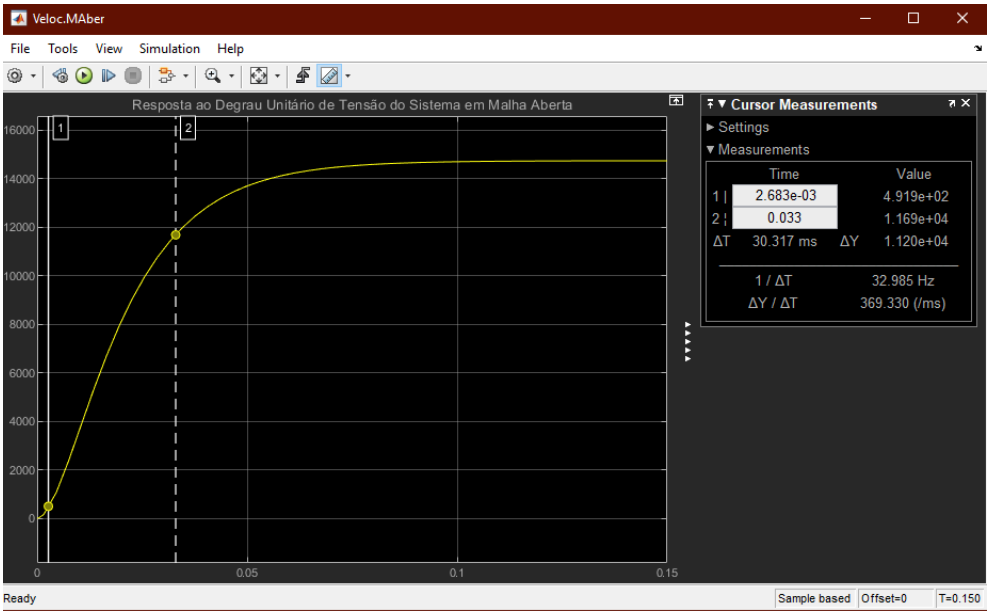


Figura 23 – Constante de tempo *T* da Curva S do sistema em malha aberta.
Fonte: Acervo pessoal.

Com isso, substituindo o valor de L e de T no tipo de Controlador PID na Tabela 1, adquiriu-se os parâmetros K_p, T_i e T_d do sistema:

$$K_p = 13,5596,\ T_i = 5,3660\ ms \text{ e } T_d = 1,3415\ ms$$

Utilizando-se desses parâmetros para calcular os ganhos K_i e K_d, têm-se:

$$K_p = 13,5596,\ K_i = 2526,9474 \text{ e } K_d = 0,0182$$

Ao substituir os valores dos ganhos acima calculados no bloco do PID do circuito no Simulink, entretanto, a curva de resposta do sistema não apresentou um bom resultado para a planta, o que já era esperado, uma vez que os valores obtidos para os ganhos proporcional (K_p) e integrativo (K_i) eram muito altos.

Dessa forma, optou-se por fazer uso de um *app* de sintonia para controladores PID oferecido pela própria The MathWorks Inc. (1994-2022a) dentro do MATLAB, o *Pid Tuner*. Essa aplicação fornece um método de ajuste de PID aplicável a blocos do controlador no Simulink, de forma que os ganhos sejam automaticamente calculados para modelos de planta lineares, sempre buscando obter um equilíbrio entre melhor performance e maior robustez (THE MATHWORKS INC., 2009).

Com isso, os valores dos ganhos sugeridos em tempo contínuo pelo *app* foram:

$$K_p = 0,0137,\ K_i = 0,9209 \text{ e } K_d = 0,00004318$$

4.5 Calculando o Período de Amostragem (*T*) do sistema

Segundo Ogata (1995), o período de amostragem *T* de um sistema pode ser calculado observando a curva de resposta ao degrau executada por ele em malha aberta, bastando encontrar apenas o tempo de subida (T_r) na curva, que, por teoria, vai de 10% a 90% do seu valor final, e dividir por qualquer valor maior que 8. Como o valor final é de 147 *rad/s*, conforme a Figura 24 a seguir, e utilizando a função pronta *step()* e o valor para divisão de 10, então:

$$90\% \text{ de } 147 \ = \ 0,0438s$$

$$10\% \text{ de } 147 \ = \ 0,0056s$$

$$T = 0,0038s \tag{4.3}$$

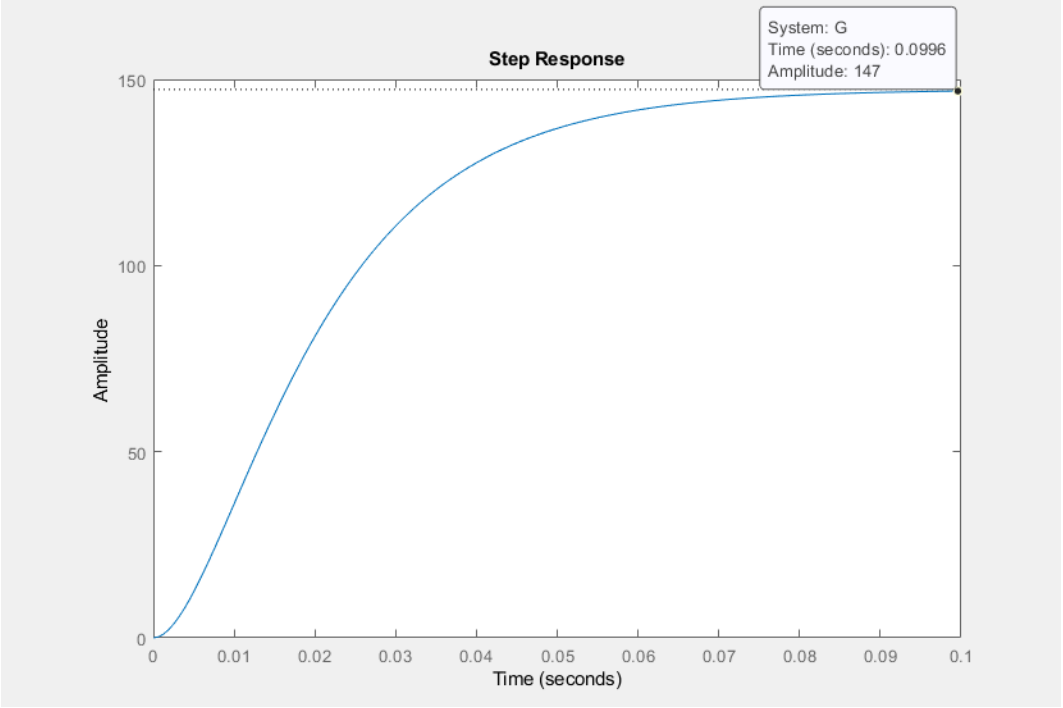


Figura 24 – Resposta ao degrau do sistema em malha aberta.
Fonte: Acervo pessoal.

4.6 Simulando o controle do sistema ideal pelo controlador PID

Para a simulação do PID através do diagrama de blocos no Simulink foi necessário fazer uso do controlador em tempo discreto, tendo que K_p, K_i e K_d serem adaptados conforme as equações 2.20, 2.21 e 2.22. Utilizando das equações de conversão, obteve-se:

$$K_p discreto = K_p continuo - (\frac{K_i discreto}{2}) = 0,011950$$

$$K_i discreto = K_i continuo * T_{amostragem} = 0,003499$$

$$K_d discreto = \frac{K_d continuo}{T_{amostragem}} = 0,011363$$

Por fim, com os ganhos do PID em mãos, implementou-se, no circuito anteriormente montado na Figura 21, o bloco do controlador em tempo discreto para que o processo de simulação do controle do motor através da compensação PID fosse realizada. A montagem da simulação pode ser vista na Figura 25 a seguir.

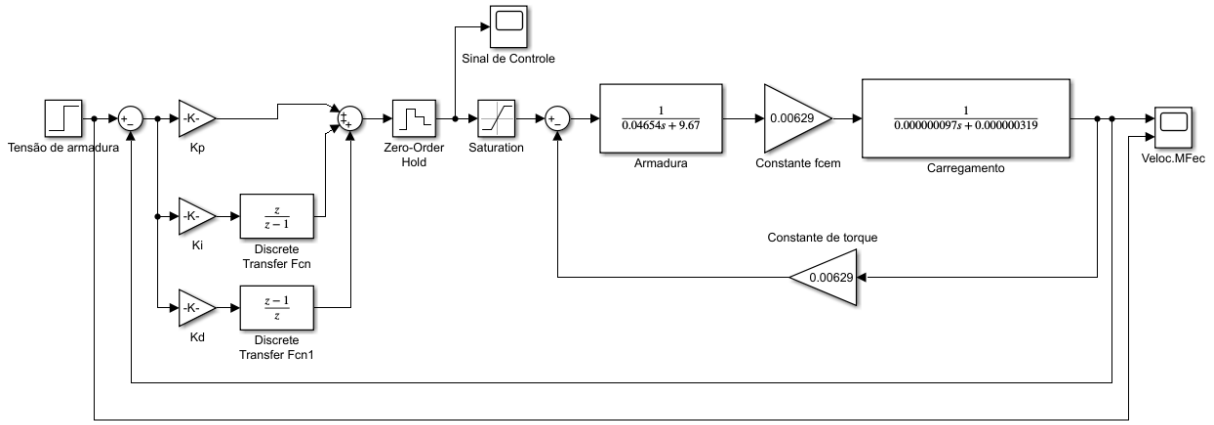


Figura 25 – Diagrama de blocos para o sistema ideal em malha fechada compensado com o controlador PID.

Fonte: Acervo pessoal.

4.7 Calculando o Coeficiente de Amortecimento (ζ) e a Frequência Natural (ω_n) do sistema

Todo sistema de 2ª ordem que se preze apresenta seus parâmetros em função de duas variáveis de amortecimento, o coeficiente de amortecimento (ζ) e a frequência natural (ω_n), calculados através das fórmulas:

$$\zeta = \frac{-\ln(OS)}{\sqrt{\pi^2+(\ln OS)^2}}$$

$$T_s = \frac{4}{\zeta\omega_n}$$

Uma vez que o *app PID Tuner* utilizado para a simulação do controle PID já fornece, além dos ganhos K_p, K_i e K_d, os parâmetros de tempo de acomodação (T_s) e *overshoot* (*OS*) do motor CC utilizado neste trabalho, serão utilizados estes valores nos cálculos do ζ e do ω_n, assim como para o projeto de todos os outros controladores, de forma a se padronizar os parâmetros.

Desse jeito, como já se tem os valores de $T_s = 0,0535$ e $OS = 6,95\% = 0,0695$, os valores de ζ e ω_n são:

$$\zeta = 0,6471 \tag{4.4}$$

$$\omega_n = 115,5406\ rad/s \tag{4.5}$$

4.8 Calculando os Polos desejados e de referência

Tendo os valores de $T_s = 0,0535$ e $OS = 6,95\% = 0,0695$ já definidos, é possível calcular os polos específicos e desejados do sistema que irão atender a esses parâmetros de projeto. Dessa forma:

$$|polo| = \exp^{-\zeta\omega_n T} = \exp^{-0,6471*115,5406*0,0038} = 0,7527 \text{ e}$$

$$angulo_{polo} = \omega_n T\sqrt{1-\zeta} = 115,5406 * 0,0038\sqrt{1-0,6471} = 0,2608\, rad$$

Os polos foram calculados no plano polar, então, torna-se necessário convertê-los para o plano cartesiano através da função pronta *pol2cart* do MATLAB, obtendo:

$$polos = 0,7272 \pm 0,1941j \tag{4.6}$$

4.9 Projetando o controle do sistema pelo método do Lugar das Raízes

A realização da simulação pelo Lugar das Raízes se deu, primeiramente, através de linhas de código no MATLAB e, em seguida, através de diagramas de bloco no Simulink, uma vez que diversos cálculos e plotagens de gráficos precisaram ser feitos para o projeto do avanço e do atraso de fase.

Da teoria do Lugar das Raízes, sabe-se que a característica básica transitória de um sistema em malha fechada, o seu comportamento, depende essencialmente da localização de seus polos em uma malha desse tipo. Diante disso, com a função de transferência encontrada anteriormente para o motor, plotou-se o Lugar das Raízes do sistema em malha fechada, e contínuo, utilizando o comando *rlocus*, obtendo o gráfico da Figura 26 a seguir:

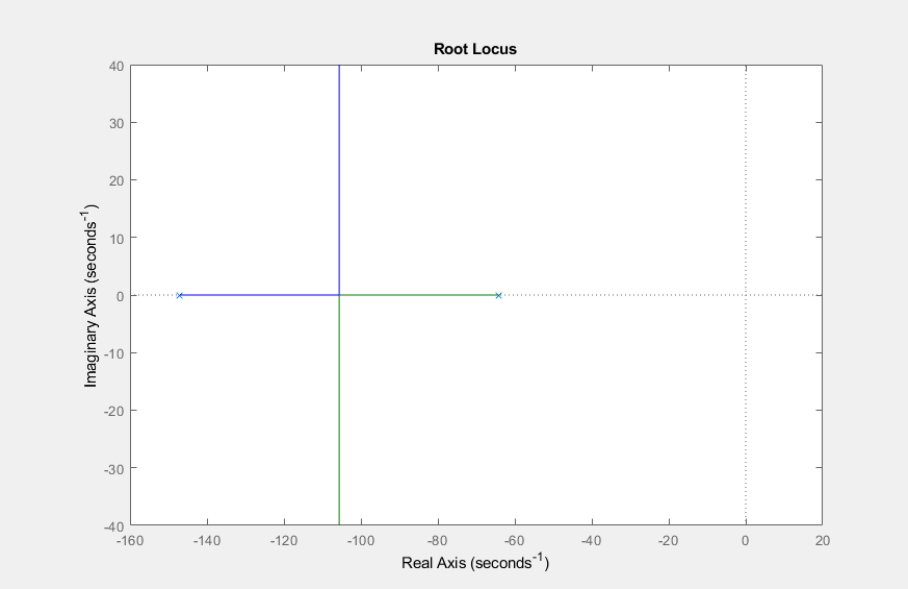


Figura 26 – Lugar das Raízes do sistema.
Fonte: Acervo pessoal.

Conforme pode ser visto na figura acima, como o denominador da função de transferência é de grau dois, então o sistema possui dois polos reais iguais e múltiplos que apresentam ramos simétricos com relação ao eixo real e que se dirigem para zeros situados no infinito. Dessa forma, o sistema contabiliza duas assíntotas que se cruzam em um ponto de abscissa -106 e ordenada nula, chamado centro das assíntotas. Os valores dos polos para *K = 0* são:

$$p_1 = -64,3 \text{ e } p_2 = -147$$

Uma vez que os polos apresentam essas características e o coeficiente de amortecimento é $\zeta = 1$, o sistema é estável e criticamente amortecido para o tempo contínuo, já que os polos estarão sempre representados no semiplano esquerdo do plano de Argand-Gauss.

No entanto, para a realização da simulação por meio do Simulink e posterior implementação no Arduino, foi necessário discretizar a função de transferência do motor CC, considerando o período de amostragem calculado em 4.3, obtendo, dessa forma:

$$G_Z = \frac{7,738z + 5,922}{z^2 - 1,355z + 0,448} \tag{4.7}$$

Plotando agora o Lugar das Raízes do sistema discretizado, tem-se a representação ilustrada na Figura 27 a seguir.

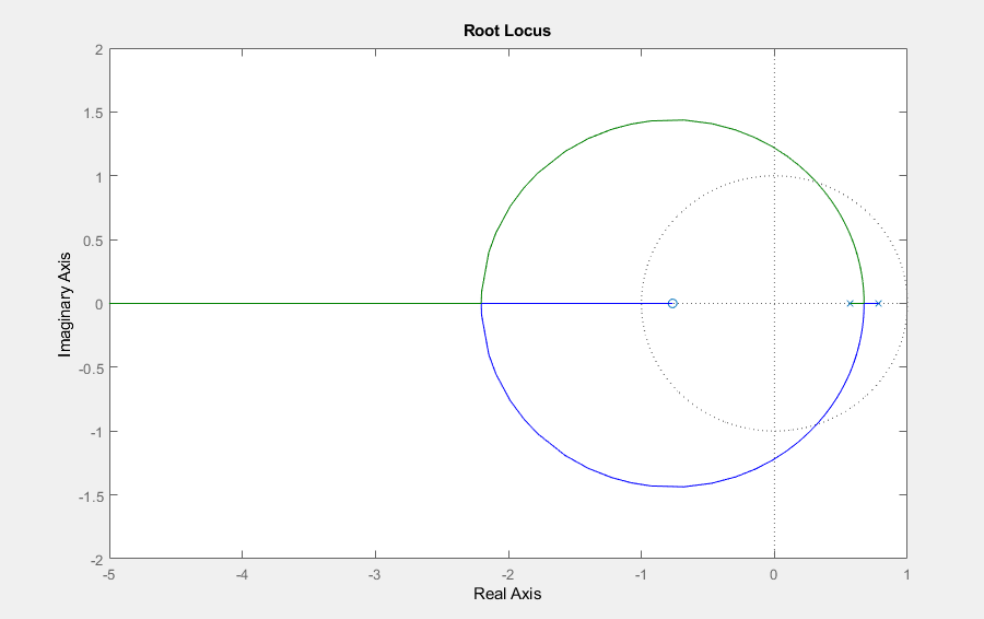


Figura 27 – Lugar das Raízes da função de transferência discretizada.
Fonte: Acervo pessoal.

Com isso vê-se que os valores dos polos e do zero são:

$$polo_1 = 0,783\ (o\ da\ direita), polo_2 = 0,572\ (o\ da\ esquerda) \text{ e } zero = -0,765$$

Observando a Figura 27 é possível perceber que o sistema em tempo discreto é estável, uma vez que seus dois polos se encontram dentro do círculo unitário, porém o Lugar das Raízes não passa pelos polos calculados na equação 4.6, o que torna necessária uma correção. Sendo assim, foi colocado o zero do controlador sobre um dos polos do sistema, pois isso garante que o Lugar das Raízes seja "empurrado"para a esquerda (avanço de fase). O polo escolhido para ser sobreposto foi o da direita, o $polo_1$, por ser o mais lento (mais próximo do círculo unitário).

Para o cálculo do novo polo do compensador, foi desenvolvido o círculo trigonométrico apresentado na Figura 28 a seguir, em que, para achar o valor de *x*, fez-se necessário calcular os valores de θ e de ϕ (ângulos relacionados ao zero e ao $polo_2$), e o valor de β.

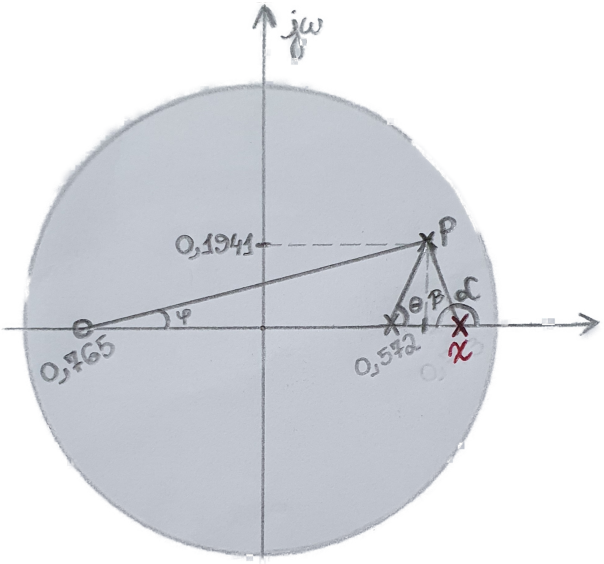


Figura 28 – Círculo trigonométrico para o cálculo do novo polo do compensador.
Fonte: Acervo pessoal.

Considerando um dos polos da equação 4.6 como o polo dominante:

$$\theta = \arctan(\frac{0{,}1941}{0{,}7272 - 0{,}572}) = 51,35°$$

$$\phi = \arctan(\frac{0{,}1941}{0{,}7272 + 0{,}765}) = 7,41°$$

Da relação de que $\sum zeros - \sum polos = -180°$:

$$\phi - \theta - \alpha = 7,41° - 51,35° - \alpha = -180°$$

$$-\alpha = -136,06°$$

$$\alpha = 136,06°$$

Fazendo o complementar de α:

$$180° - \alpha = 180° - 136,06° = 43,94° = \beta$$

Assim:

$$\tan\beta = \frac{0{,}1941}{x - 0{,}7272}$$

$$0,9637 = \frac{0,1941}{x - 0,7272}$$

$$0,9637x - 0,7008 = 0,1941$$

$$x = 0,929$$

Então, o valor do novo polo é de $polo_n = 0,929$.

Dessa forma, o compensador por avanço de fase para correção do Lugar das Raízes é dado por:

$$C = \frac{z - zero}{z - polo_1} * K$$

$$C = \frac{z - 0,783}{z - 0,929} * K \tag{4.8}$$

Implementando o compensador ao sistema e plotando o Lugar das Raízes com a presença dele, obteve-se o *plot* da Figura 29 a seguir.

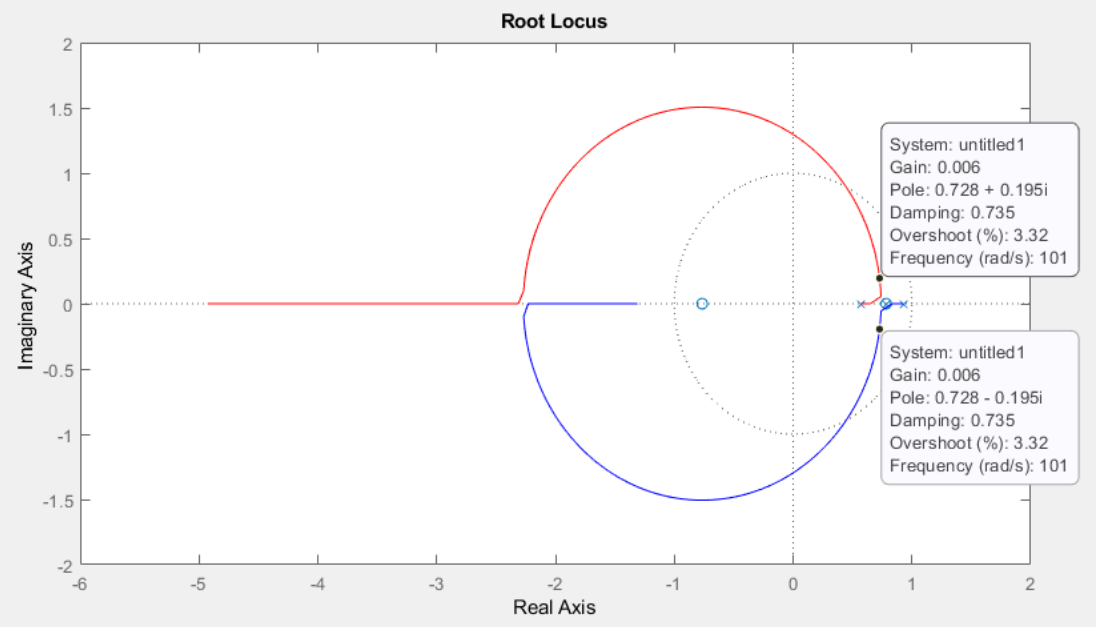


Figura 29 – Lugar das Raízes da função de transferência discretizada e com compensador por avanço de fase.

Fonte: Acervo pessoal.

Considerando uma aproximação dos polos complexos calculados em 4.6, no gráfico da Figura 29, tem-se que o ganho $K = 0,006$. Substituindo *K*, em 4.8, a função de transferência discreta do compensador por avanço de fase é:

$$C = \frac{z - 0,783}{z - 0,929} * 0,006$$

$$C = \frac{0,006z\ -\ 0,004698}{z\ -\ 0,929} \tag{4.9}$$

Ao aplicar apenas o avanço de fase no sistema, foi observado que a resposta teve uma boa dinâmica, ocorrendo mais rapidamente e tendo um *overshoot* reduzido. Contudo, a estabilização não estava ocorrendo no *setpoint* desejado (100 *rad/s*), o que fez ser necessário corrigir o sistema mais uma vez, só que agora com um compensador por atraso de fase.

Fazendo uso da função de transferência discretizada do motor CC (equação 4.7), calculou-se o valor do numerador e do denominador da equação, substituindo *z = 1*, aplicando, assim, o Teorema do Valor Final. Dessa forma, os valores obtidos estão apresentados a seguir.

$$valor_{num} = 13,6600$$

$$valor_{den} = 0,0930$$

Considerando um dos polos dominantes escolhido em 4.6, no caso o $polo = 0,7272 + 0,1941j$ e substituindo-o no lugar de z na função de transferência discretizada (equação 4.7), obtém-se:

$$num_p = 11,5491 + 1,5019i$$

$$den_p = -0,0462 + 0,0193i$$

$$valor = \frac{num_p}{den_p} = -2,0127x10^2 - 1,1653x10^2i$$

Assim, utilizando a condição de módulo na *valor*:

$$K_p = 0,0043$$

Calculado K_p a partir do *script* do Apêndice C, o próximo passo foi encontrar um valor de *K* qualquer que satisfazia a equação do erro em regime permanente para uma entrada do tipo degrau. Considerando o erro tolerado como sendo 0,01:

$$e(\infty) = \frac{A}{1 + K_{qualquer}}$$

Se $K_{qualquer} = \lim_{z->1} G_c(z)G(z)$, então:

$$G_c(z) = 0,6740$$

Dessa forma, o compensador por atraso de fase para correção do Lugar das Raízes é dado por:

$$G_c(z) = K_p * \frac{z - \beta 1}{z - \beta 2}$$

$$G_c(z) = \frac{z \ - \ 0,843}{z \ - \ 0,999} \tag{4.10}$$

O valor de β_2 foi escolhido aleatoriamente para ser um valor o mais próximo possível de 1, mas sem ser o 1.

Incluindo o compensador por atraso de fase ao diagrama de blocos do Lugar das Raízes juntamente com o compensador por avanço de fase, obteve-se a seguinte montagem:

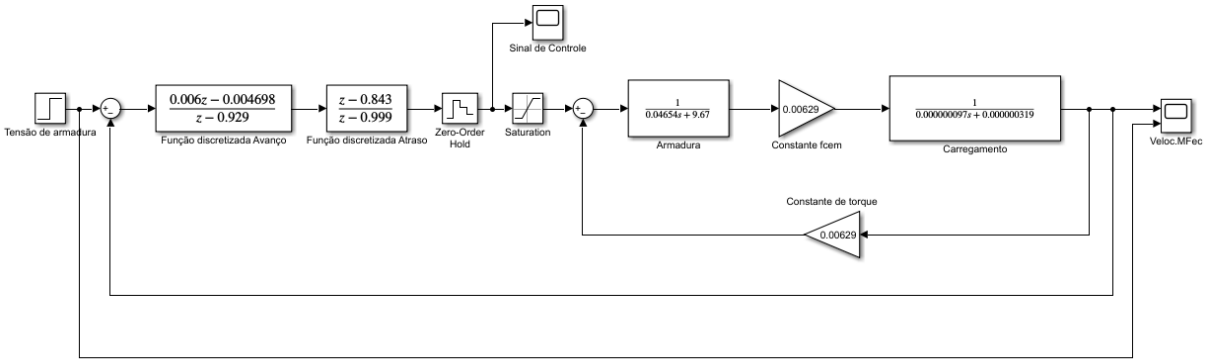


Figura 30 – Diagrama de blocos para o sistema ideal em malha fechada compensado por Avanço e Atraso de Fase.

Fonte: Acervo pessoal.

4.10 Simulando o controle do sistema físico real

Para a execução "real" dos dois controles realizados neste trabalho, fez-se uso da IDE do Arduino como substituto do Simulink, visto que a lógica de recebimento e envio dos dados por essa plataforma acontecia em tempo real e com menos *delay*. Desse modo, a lógica de conversão da entrada PWM do *encoder* para *rad/s*, a implementação do controlador e a lógica do traçado visual da curva de resposta foram todas executadas por meio de linhas de comando na IDE.

Além disso, para todos os controladores projetados foi necessário implementar no código a lógica de um filtro de média móvel que faz a média de 25 leituras de velocidade, já em *rad/s*, e a usa como saída no lugar do valor instantâneo do *encoder*. Dessa forma, os ruídos que estavam afetando a dinâmica de saída do sistema são filtrados e não ocasionam mais tanta oscilação durante o processo. A única desvantagem desse uso é obter a curva com os valores filtrados com um pouco de atraso se comparada com a original, atraso esse que vai variar conforme o número desejado de leituras de velocidade.

4.10.1 Simulando o controle do sistema real pelo controlador PID

A lógica da programação para o cálculo da velocidade em *rad/s* é a mesma utilizada no código anteriormente criado do Apêndice B, que faz uso de uma interrupção e das fórmulas 4.11

e 4.12 para a mensuração.

$$RPM \ = \ 10 \ * \ pulsos \ * \ (60 \ / \ quantidade - de - furos - do - disco) \tag{4.11}$$

$$rad/s \ = \ RPM \ * \ 0,10472 \tag{4.12}$$

Sabendo-se que a função de transferência de um controlador PID em tempo contínuo é dada pela equação 2.1, segundo PLC (2022), basta adicionar um *Zero-Order Hold* à entrada da planta para tornar o sinal de controle contínuo novamente. Dessa forma, como uma adaptação da equação 2.5, obteve-se a seguinte equação 4.13:

$$\frac{U(z)}{E(z)} \ = \ K_p + K_i T \frac{1}{z \ - \ 1} \ + \ \frac{K_d}{T} \frac{z \ - \ 1}{z} \tag{4.13}$$

Levando em consideração que os ganhos K_p, K_d e K_i são os ganhos do sistema em tempo contínuo, foi possível obter a seguinte equação de diferenças (4.14) para o PID discreto:

$$u(n) \ = \ u(n-1) \ + \ (K_p \ + \ \frac{K_d}{T}) \, e(n) \ + \ (-K_p \ + \ K_i T \ - \ 2\frac{K_d}{T}) \, e(n-1) \ + \ \frac{K_d}{T} \, e(n-2) \tag{4.14}$$

onde ***e(n)*** corresponde a: $erro = setpoint - velocidade$.

A implementação em linhas de código da equação 4.14, juntamente com o cálculo da velocidade e a execução do filtro de média móvel, está apresentada no *script* do Apêndice D.

4.10.2 Simulando o controle do sistema real por Lugar das Raízes

Ainda fazendo uso das equações 4.11 e 4.12 para o cálculo da velocidade em *rad/s* e da lógica para a execução do filtro de média móvel, também foi necessário transformar a função de transferência discreta de controle em uma equação de diferenças que fosse possível de ser implementada no Arduino, conforme pode ser visto no *script* do Apêndice E.

A função de controle do Lugar das Raízes é dada pelo avanço e pelo atraso de fase simultaneamente, o que fez ser necessário unir as funções de ambos em uma única função de transferência, resultando na equação 4.15 a seguir.

$$\frac{U(z)}{E(z)} \ = \ \frac{0,006z^2 \ - \ 0,009756z \ + \ 0,003960}{z^2 \ - \ 1,928z + 0,928071} \tag{4.15}$$

Multiplicando cruzado e, posteriormente, multiplicando ambos os lados por z^{-2} para escrever a função em termo de atrasos, obteve-se:

$$U(z) \ - \ 1,928z^{-1}U(z) \ + \ 0,9281z^{-2}U(z) \ = \ 0,006E(z) \ - \ 0,0098z^{-1}E(z) \ + \ 0,0039z^{-2}E(z) \tag{4.16}$$

Agora com tudo em termo de atrasos, foi possível realizar a transformada *Z* inversa e voltar a função para o domínio do tempo discreto, conforme descrito na equação 4.17.

$$u(n) = 0,006e(n) - 0,0098e(n-1) + 0,0039e(n-2) + 1,928u(n-1) - 0,9281u(n-2) \tag{4.17}$$

Obtida essa equação, bastou implementá-la no código como a função responsável pelo controle do motor em que *u(n)* é o sinal de controle.

5 RESULTADOS

No presente capítulo, são apresentados os resultados provenientes da implementação dos controladores, tanto considerando-se os testes simulados através do Simulink quanto considerando-se a resposta real com uso direto do motor. Além disso, as especificações e o desempenho de cada controle também são avaliados e discutidos para validação de cada procedimento.

5.1 Projeto de controle utilizando PID

Conforme mencionado na seção 4.4, fez-se uso do *app Pid Tuner* do MATLAB para obter os ganhos K_p, K_i e K_d ideais do sistema formulado, assim como seus melhores parâmetros de performance e robustez.

Os parâmetros fornecidos pelo *app* estão apresentados na Figura 31 a seguir, sendo o $Rise\,time = tempo\,de\,subida = 0,016\,segundos$, o $Settling\,time = tempo\,de\,acomodacao = 0,0535\,segundos$ e o $Overshoot = 6,95\%$. Os ganhos do controlador também estão sendo mostrados na figura.

Controller Parameters

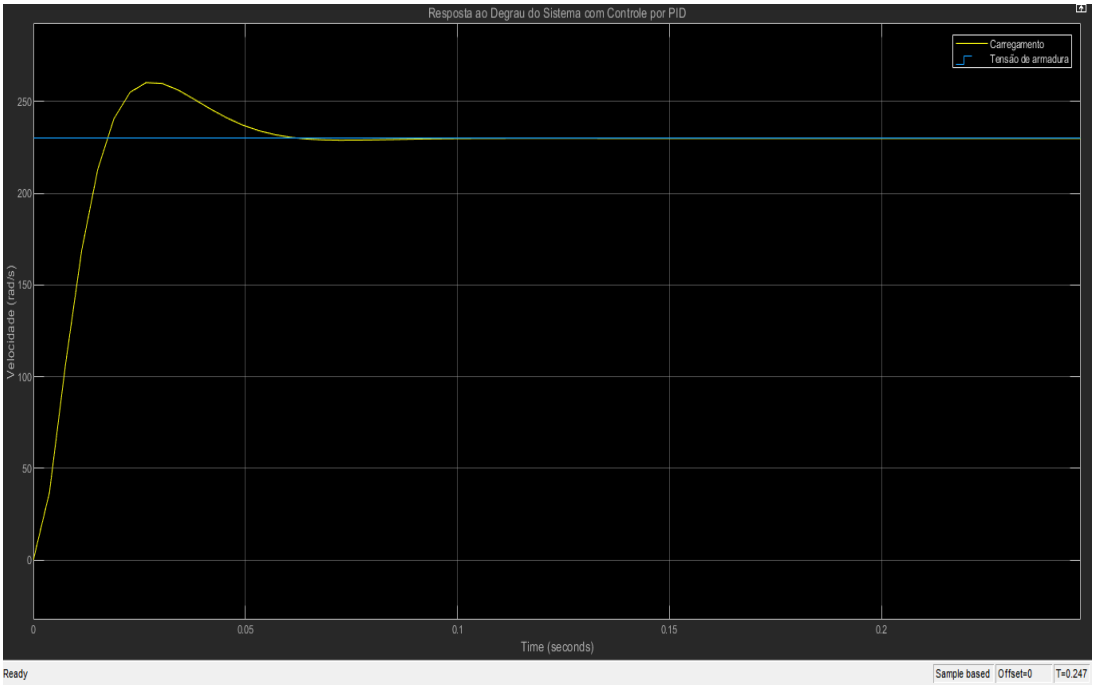
	Tuned	Block
P	0.013709	0.013709
I	0.9209	0.9209
D	4.3182e-05	4.3182e-05
N	11107.9871	11107.9871

Performance and Robustness

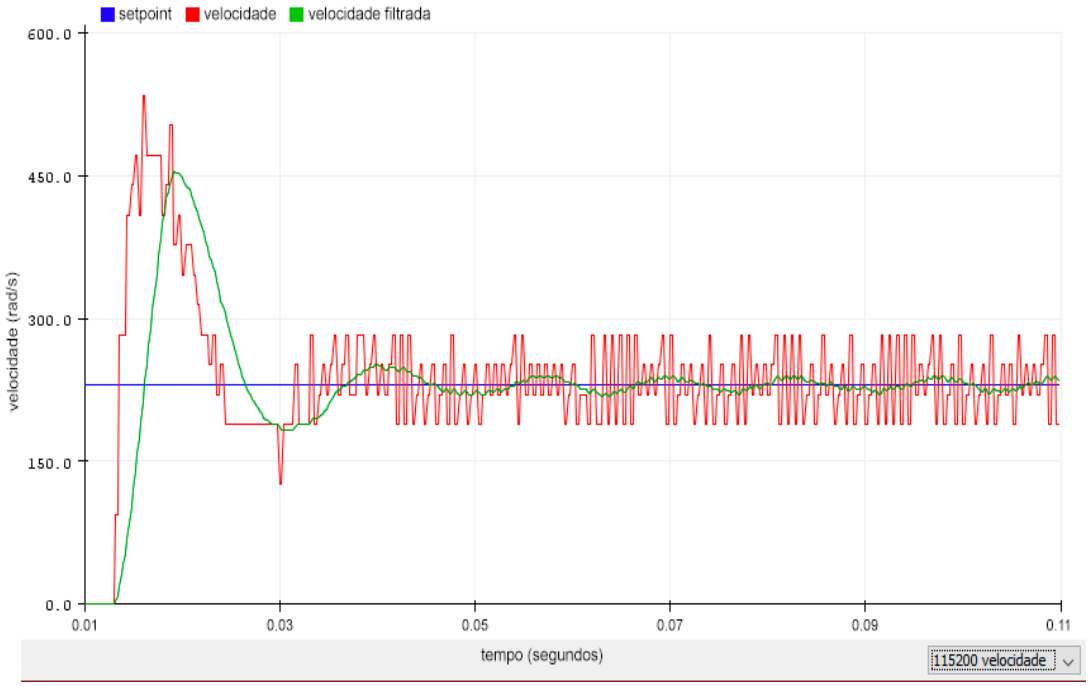
	Tuned	Block
Rise time	0.016 seconds	0.016 seconds
Settling time	0.0535 seconds	0.0535 seconds
Overshoot	6.95 %	6.95 %
Peak	1.07	1.07
Gain margin	Inf dB @ Inf rad/s	Inf dB @ Inf rad/s
Phase margin	69 deg @ 97.2 rad/s	69 deg @ 97.2 rad/s
Closed-loop stability	Stable	Stable

Figura 31 – Parâmetros de performance do PID.
Fonte: Acervo pessoal.

Fazendo uso dos dados fornecidos na figura anterior, foi possível obter a resposta ao degrau simulada para o sistema com o uso do PID (Figura 25), assim como a resposta do sistema físico (*script* de código do Apêndice D), podendo ambos os resultados serem visualizados na Figura 32 a seguir.



(a) Resposta ideal.



(b) Resposta do sistema físico.

Figura 32 – Resposta em malha fechada do motor compensado com o controlador PID e *setpoint* de 230 *rad/s*.

Fonte: Acervo pessoal.

O *setpoint* inicialmente considerado foi de 230 *rad/s*, porém, como será apresentado

posteriormente, a metodologia aplicada e o controlador projetado foram eficientes para outros *setpoints* também.

A partir da resposta ideal obtida na Figura 32(a), é possível observar que os parâmetros de performance do sistema se comportam de maneira satisfatória, uma vez que a curva começa com oscilação no regime transitório, mas rapidamente se estabiliza em torno do *setpoint* no regime permanente, e seus valores, apesar do *overshoot* ter sido um pouco maior, são mantidos dentro dos valores projetados na Figura 31. O *overshoot* máximo é de $13,13\%$, o tempo de acomodação considerando 2% de erro (234,6 rad/s) é de aproximadamente $0,053s$ e o tempo de subida é de $0,016s$.

Já na resposta do sistema físico obtida na Figura 32(b) é visível que o sistema se comporta de maneira semelhante à resposta ideal, se estabilizando por completo no regime permanente com o passar do tempo, porém com um *undershoot* após o *overshoot* inicial.

O traçado do gráfico utilizando o *Plotter serial* do Arduino acaba apresentando muitas limitações se comparada à plotagem por meio do *Scope* do Simulink, sendo uma delas a forma como o eixo X é apresentado. Ele também representa o tempo, porém em termos do que o fabricante chama de pontos. O eixo tem 500 pontos, sendo que o tempo entre cada ponto é o tempo entre duas chamadas da função *Serial.println()* consecutivas, isto é, geralmente é igual ao tempo de duração da função *loop()*. Sendo assim, foi necessário realizar uma conversão de valores do eixo X (de pontos para tempo (s)) para que os eixos da curva do sistema físico ficassem iguais aos eixos da curva da resposta ideal.

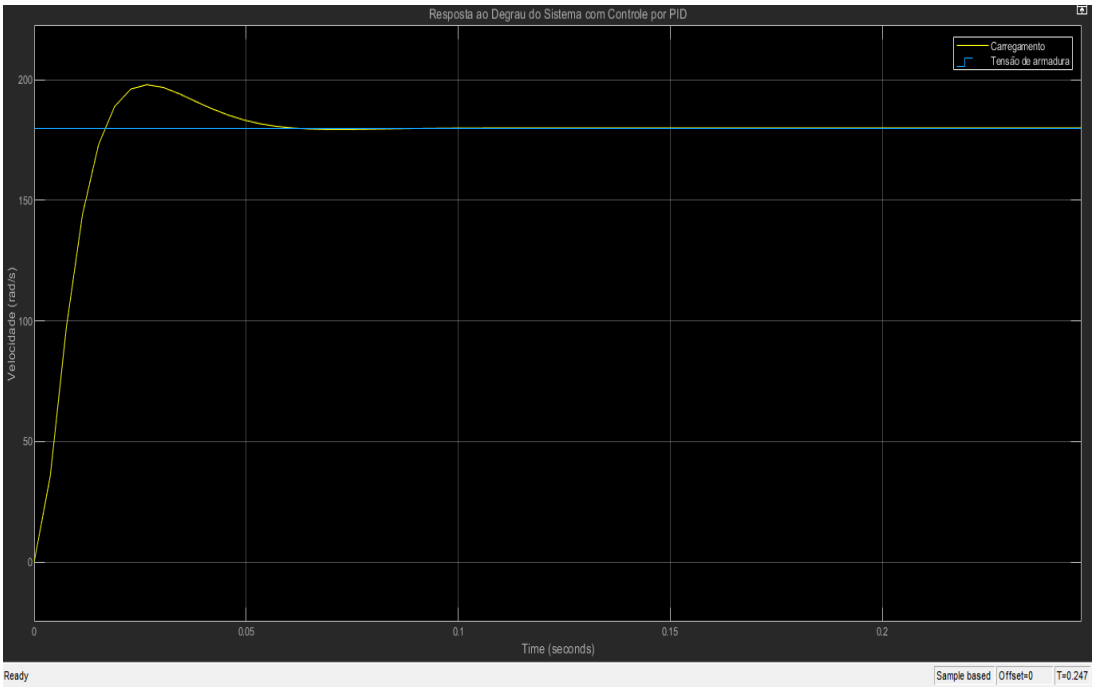
Dessa forma, conforme pode ser visto na Figura 32(b), o tempo de acomodação da curva do sistema físico considerando 2% de erro (225,4 rad/s) é de aproximadamente $0,023s$, o tempo de subida é de $0,002s$ e o *overshoot* máximo é de $95,65\%$, cerca de aproximadamente o dobro do tamanho da curva obtida até o tempo de subida.

É importante ressaltar que o motor apresenta uma velocidade de partida alta para todas as curvas de resposta do sistema físico, possivelmente porque o motor se encontra parado antes das execuções, ao invés de já apresentar algum movimento.

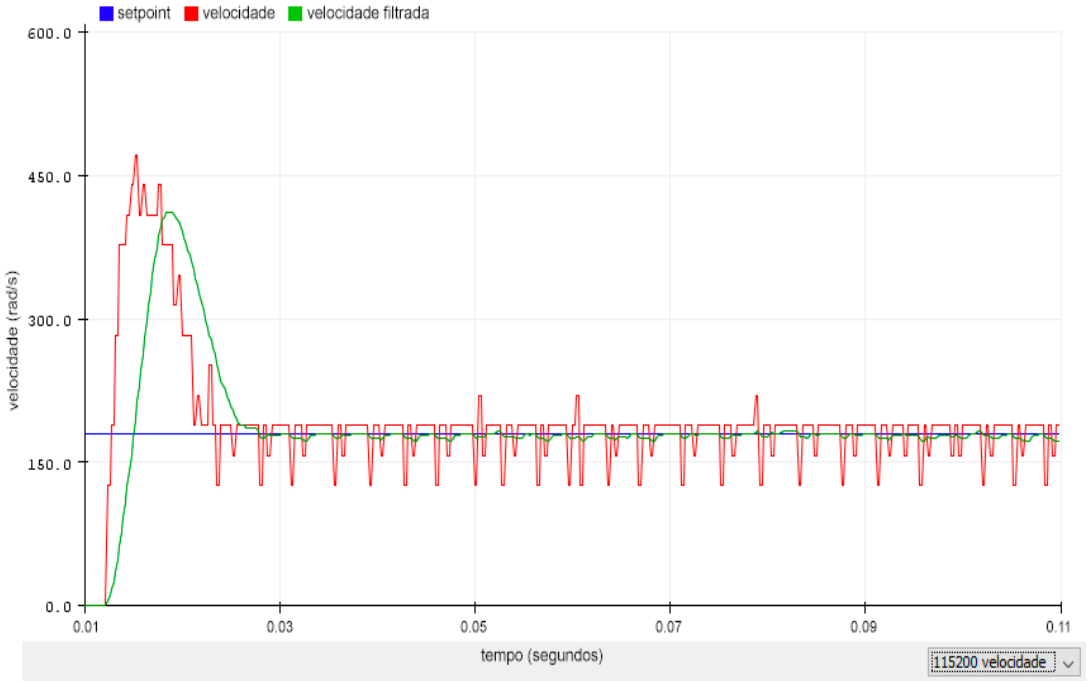
Apesar de existir uma variação do sinal do sistema físico em torno do *setpoint*, a resposta tende a se acomodar em seu valor ao longo do tempo, tornando o controle do motor através do controlador PID satisfatório. Além disso, os tempos de subida e acomodação do sistema físico são menores que seus valores ideais e são mantidos dentro dos parâmetros projetados na Figura 31.

Era de se esperar que os resultados obtidos em ambos os casos, ideal e real, apresentassem algumas diferenças no desempenho, uma vez que há diversas variáveis que afetam o sistema físico, mas não o simulado, como por exemplo, ruídos provocados durante a leitura do *encoder*, *delays* ao longo do circuito na comunicação entre os dispositivos, simplificações nas modelagens e cálculos dos parâmetros, especificações internas dos periféricos, entre muitas outras.

Para se ter certeza de que o controle estava acontecendo e que o controlador estava funcionando corretamente, outros valores de *setpoints* foram testados para ambas as situações. A Figura 33 a seguir apresenta as curvas de resposta para o valor de 180 *rad/s*.



(a) Resposta ideal.



(b) Resposta do sistema físico.

Figura 33 – Resposta em malha fechada do motor compensado com o controlador PID e *setpoint* de 180 *rad/s*.

Fonte: Acervo pessoal.

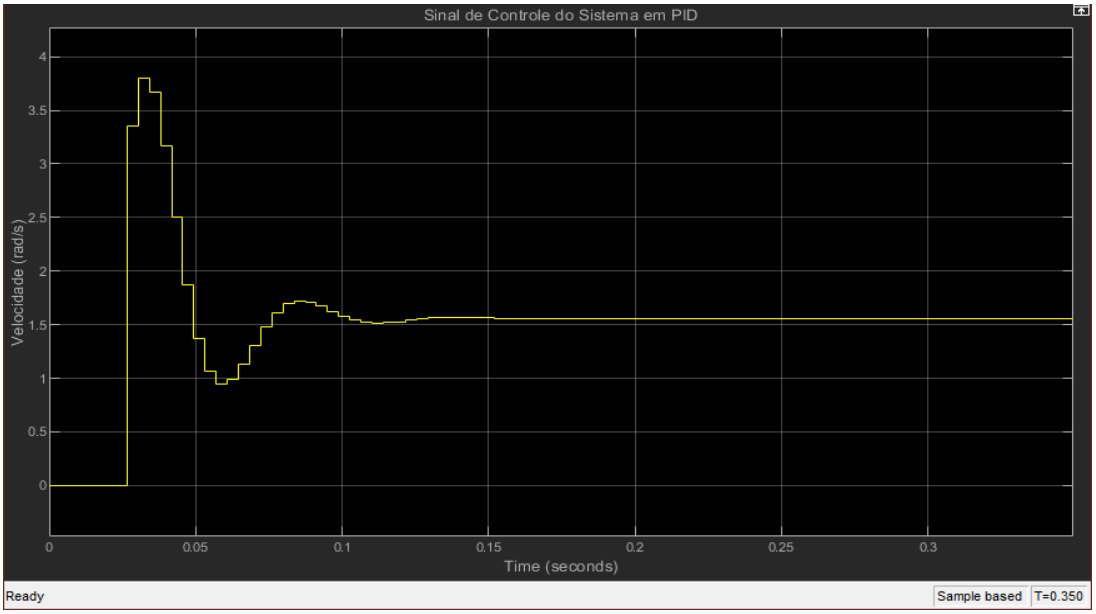
Conforme pode ser visto na Figura 33, para a resposta ideal, o *overshoot* é de $9,89\%$, o tempo de acomodação considerando 2% de erro (183,6 rad/s) é de aproximadamente $0,049s$ e o tempo de subida de $0,016s$. Já para a resposta do sistema físico, o *overshoot* é de $122,22\%$, o tempo de subida de $0,004s$ e o tempo de acomodação considerando 2% de erro (183,6 rad/s) de aproximadamente $0,011s$. Sendo assim, percebe-se que, para ambos os casos, os parâmetros de projeto do sistema continuam sendo mantidos dentro dos valores projetados na Figura 31 e continuam sendo satisfatórios, mesmo que o *overshoot* em ambos continue sendo maior.

Comparando-se a Figura 33 com a 32, observa-se que o sistema se comporta de maneira pior para um *setpoint* maior, apresentando um *overshoot* seguido de um *undershoot* que se estende por mais tempo e logo se estabiliza com a presença de ruídos, enquanto que, para o *setpoint* menor, a variação em torno do referencial é baixa e não há presença de *undershoot*. Além de os parâmetros de desempenho serem melhores para o *setpoint* menor.

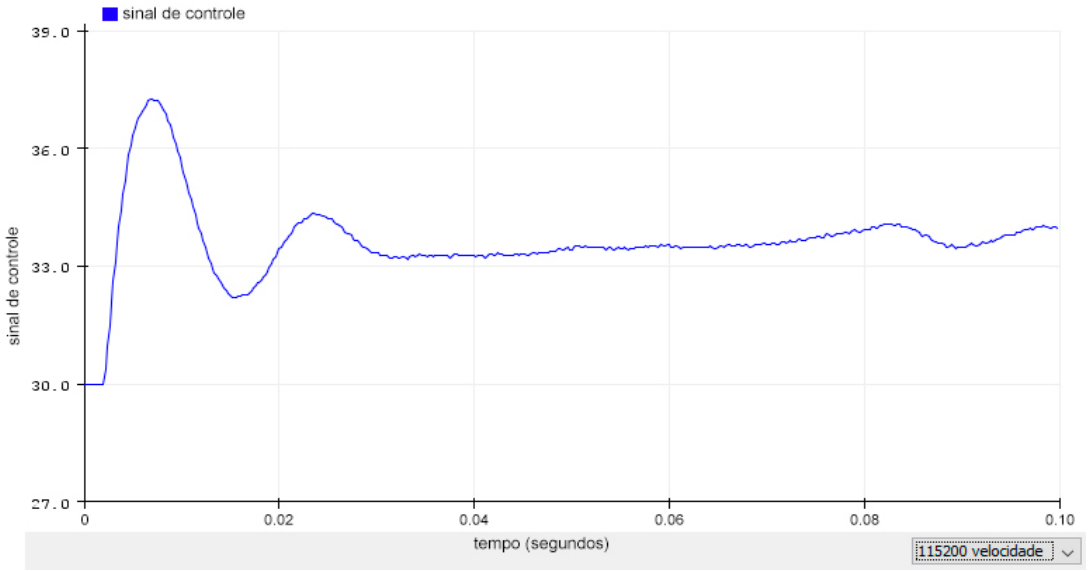
Para ambas as respostas, ideal e real, os valores dos tempos de subida e de acomodação e do *overshoot* se alteraram minimamente entre os casos, de forma que puderam ser considerados praticamente iguais para os diferentes *setpoints*, o que já era de se esperar, uma vez que os ganhos do controlador são os mesmos e a função de transferência do motor também.

Dessa forma, confirmou-se que o controlador PID está funcionando conforme deveria e que o projeto de controle através dessa sintonia foi corretamente realizado e é convincente, apesar das diferenças.

Analisando também o sinal de controle do sistema na Figura 34, coletado na saída do controlador antes de entrar na malha fechada do motor, percebe-se que, tanto para o ambiente ideal quanto para o ambiente real, o sinal apresenta a mesma forma de onda e o mesmo comportamento, reafirmando mais uma vez a validade do controlador.



(a) Resposta ideal.



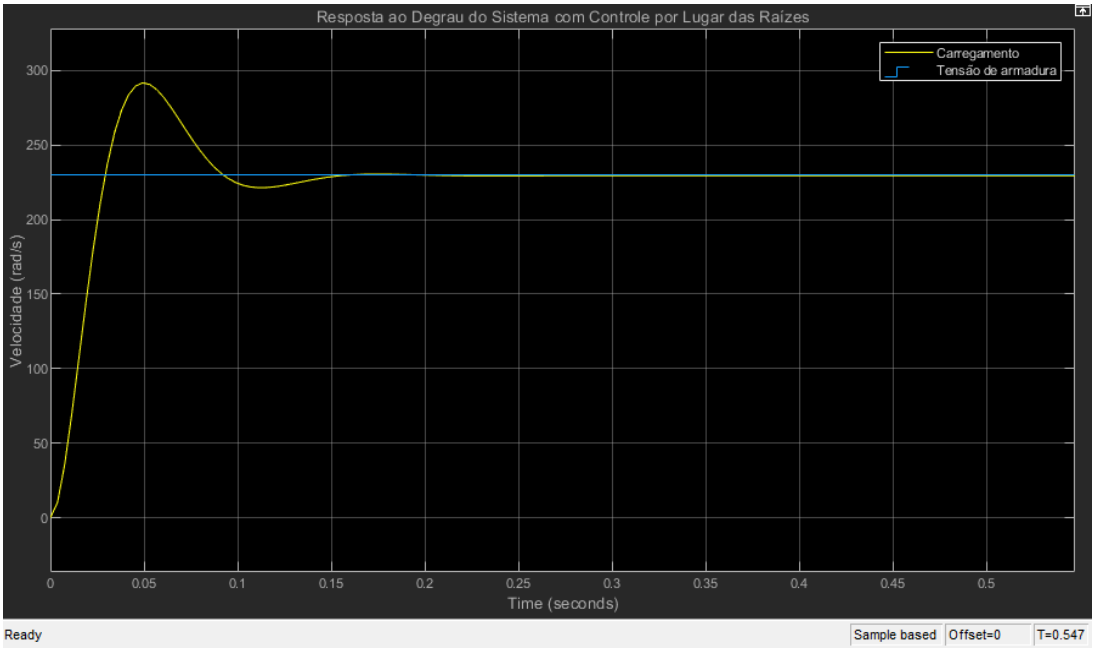
(b) Resposta do sistema físico.

Figura 34 – Sinal de controle do sistema com uso do PID.
Fonte: Acervo pessoal.

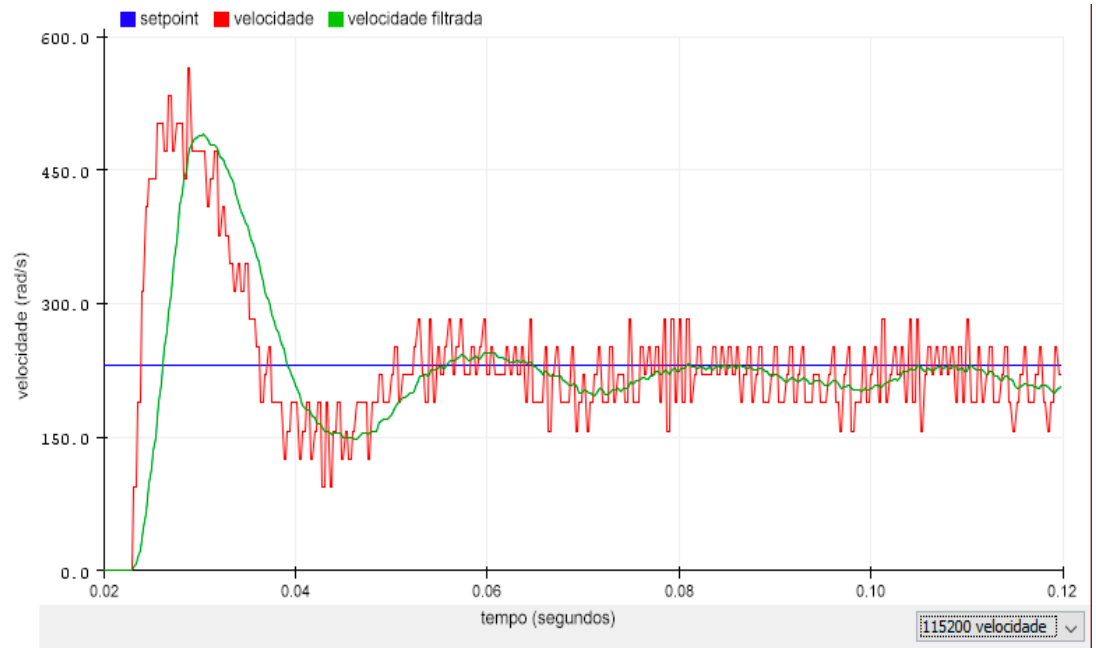
5.2 Projeto de controle utilizando Lugar das Raízes

Utilizando dos valores calculados e dos diagramas construídos nas seções 4.9 e 4.10.2, obteve-se a resposta ao degrau simulada e a resposta do sistema físico (*script* de código do Apêndice E) com o uso do Avanço e do Atraso de Fases nos conceitos de Lugar das Raízes, podendo tais respostas serem visualizadas na Figura 35 a seguir.

Assim como na análise do PID, o *setpoint* inicialmente considerado foi de 230 *rad/s*, porém, posteriormente, será mostrado que a metodologia aplicada e o controlador projetado foram eficientes para outros *setpoints* também.



(a) Resposta ideal.



(b) Resposta do sistema físico.

Figura 35 – Resposta em malha fechada do motor compensado com Avanço e Atraso de Fase e *setpoint* de 230 *rad/s*.

Fonte: Acervo pessoal.

A partir da resposta ideal obtida na Figura 35(a), observa-se que os parâmetros de performance do sistema se comportam de maneira satisfatória, uma vez que a curva começa com oscilação no regime transitório, mas rapidamente se estabiliza em torno do *setpoint* no regime permanente. O *overshoot* máximo é de $26,78\%$, o tempo de acomodação considerando 2% de erro (225,4 rad/s) é de aproximadamente $0,135s$ e o tempo de subida é de $0,029s$.

Quando comparada com a curva de resposta ideal do PID (Figura 32a), vê-se que os

parâmetros de projeto para este controlador, utilizando o mesmo valor de *setpoint*, são maiores, o que provavelmente é resultado da inserção do atraso de fase, que tornou o regime transitório mais instável.

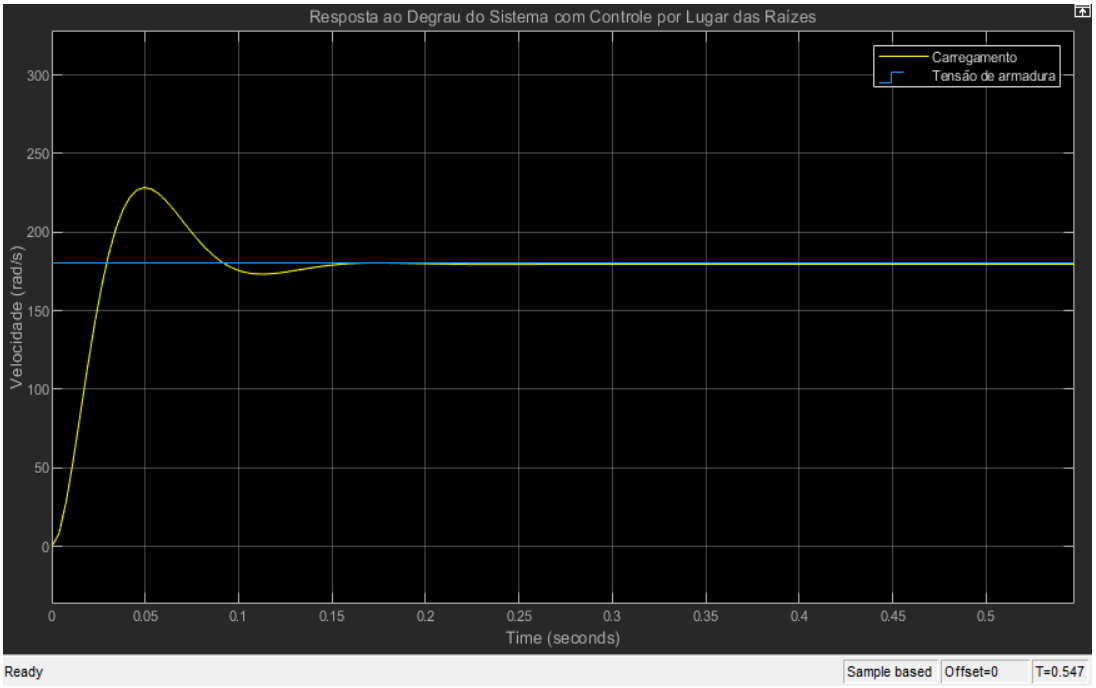
Já na resposta do sistema físico obtida na Figura 35(b), entende-se que o sistema se comporta de maneira semelhante à resposta ideal se observado de uma forma macro, mesmo apesar de algumas pequenas diferenças. Ainda que as variações de valores do eixo de velocidade sejam visíveis, o sistema apresenta um *overshoot* inicial seguido de um *undershoot* antes de se estabilizar por completo no regime permanente.

Fazendo uso da definição apresentada anteriormente do eixo X do *Plotter Serial* do Arduino, tem-se que o tempo de acomodação da curva do sistema físico considerando 2% de erro (225,4 rad/s) é de aproximadamente $0,037s$, o tempo de subida é de $0,002s$ e o *overshoot* máximo é de $113,04\%$, cerca de aproximadamente o dobro do tamanho da curva obtida até o tempo de subida.

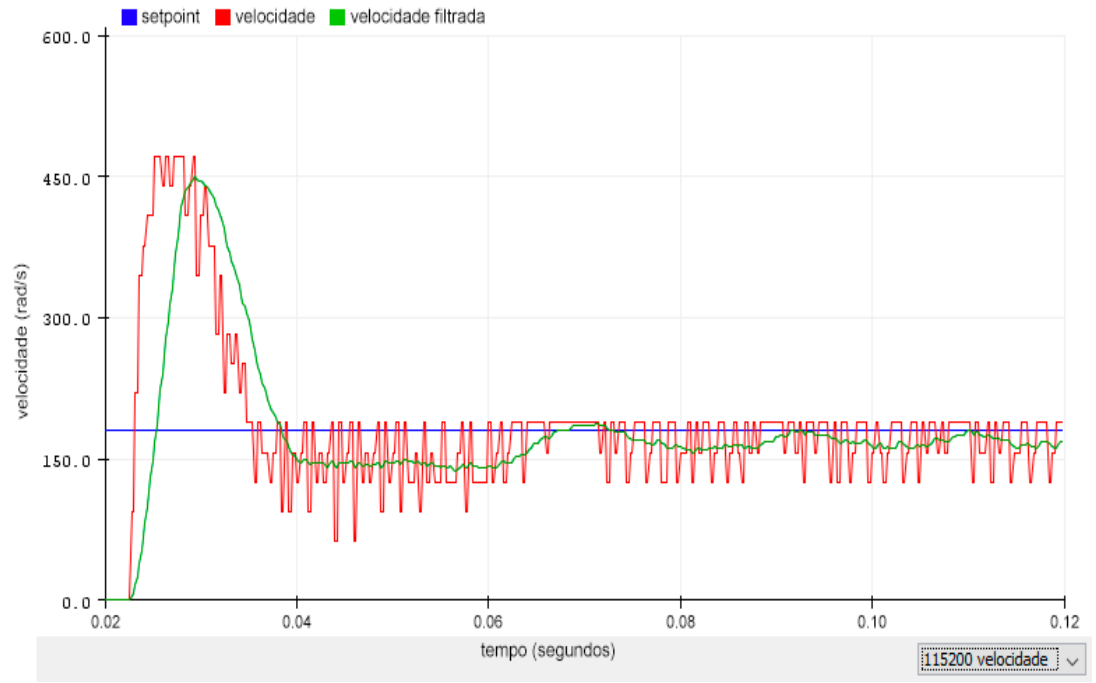
O mesmo problema de variação do sinal do sistema físico em torno do *setpoint* acontece aqui, porém mesmo o sinal apresentando uma constante oscilação, a resposta tende a se acomodar em torno do seu valor ao longo do tempo, tornando o controle do motor através deste tipo de controle satisfatório. Além disso, os tempos de subida e de acomodação do sistema físico são menores que seus valores ideais.

As mesmas justificativas e observações levantadas na seção anterior são válidas para este controle no que diz respeito às diferenças de desempenho entre os casos ideal e real. Diversas variáveis afetam o sistema real, mas não o simulado, e devem ser levadas em consideração na resposta.

Mais uma vez, assim como para a avaliação do PID, outros valores de *setpoints* foram testados para ambas as situações real e ideal do sistema para confirmar de que o controle estava acontecendo de forma devida e que o controlador estava funcionando corretamente. A Figura 36 a seguir apresenta as curvas de resposta para o valor de 180 *rad/s*.



(a) Resposta ideal.



(b) Resposta do sistema físico.

Figura 36 – Resposta em malha fechada do motor compensado com Avanço e Atraso de Fase e *setpoint* de 180 *rad/s*.
Fonte: Acervo pessoal.

Conforme pode ser visto na Figura 36, para a resposta ideal, o *overshoot* é de $26,83\%$, o tempo de acomodação considerando 2% de erro (183,6 rad/s) é de aproximadamente $0,134s$ e o tempo de subida de $0,029s$. Já para a resposta do sistema físico, o *overshoot* é de aproximadamente $147,22\%$, o tempo de subida de $0,002$ e o tempo de acomodação considerando 2% de erro (176,4 rad/s) é de aproximadamente $0,053$. Como era de se esperar, para outros valores de

setpoint, os parâmetros de desempenho do sistema continuam sendo maiores que os projetados na Figura 31.

Comparando a Figura 36 com a 35, observa-se que o sistema físico se comporta de maneira pior para um *setpoint* menor, apresentando um *overshoot* seguido de um *undershoot* que se estende por mais tempo e logo se estabilizando um pouco abaixo do referencial da entrada. Apesar de todos os cálculos terem sidos cuidadosamente realizados e as análises também, o sistema físico insistiu em se acomodar antes de chegar ao valor do *setpoint* para valores mais baixos do referencial, problema provavelmente proveniente dos arredondamentos feitos ao longo dos cálculos e resultados.

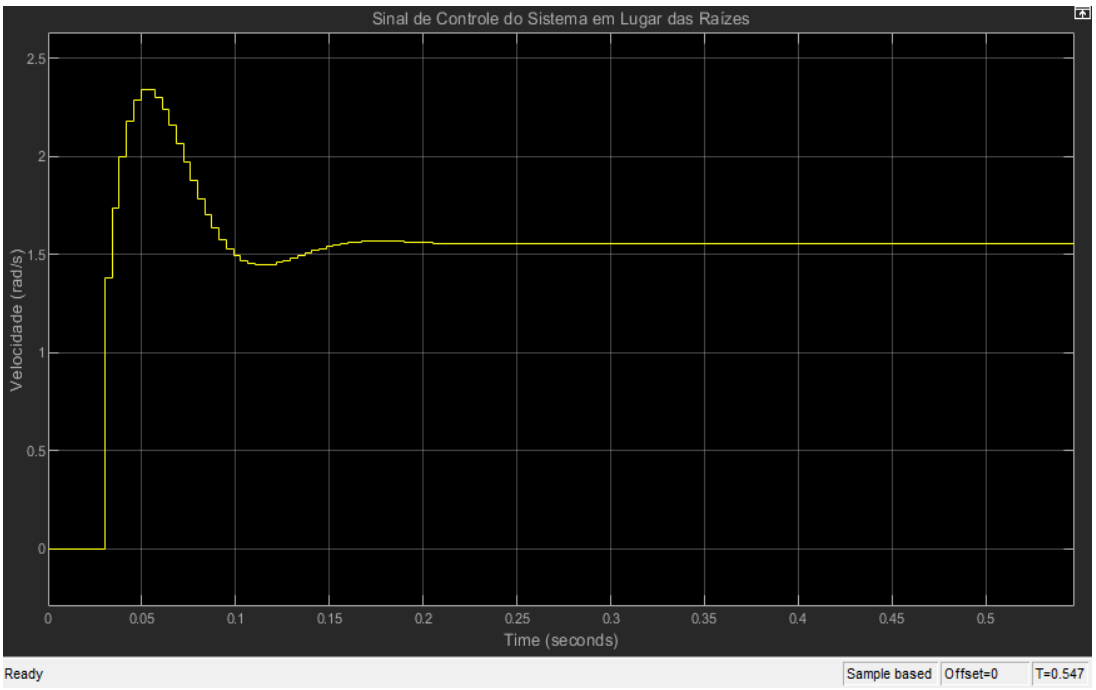
Apesar das diferenças para o valor de *setpoint* mais baixo, para ambas as respostas, ideal e real, os valores dos tempos de subida e de acomodação e do *overshoot* não se alteraram muito para os diferentes *setpoints*, o que já era de se esperar uma vez que os ganhos do controlador são os mesmos e a função de transferência do motor também.

Ainda que a resposta do sistema esteja se estabilizando antes de chegar ao *setpoint*, pode-se afirmar que o controlador por Avanço e Atraso de Fase pelo método do Lugar das Raízes está funcionando conforme deveria, uma vez que o sinal se estabiliza em torno de um valor, mesmo assim. Obviamente, algumas melhorias são necessárias, porém o projeto de controle através dessa sintonia foi corretamente realizado e é convincente.

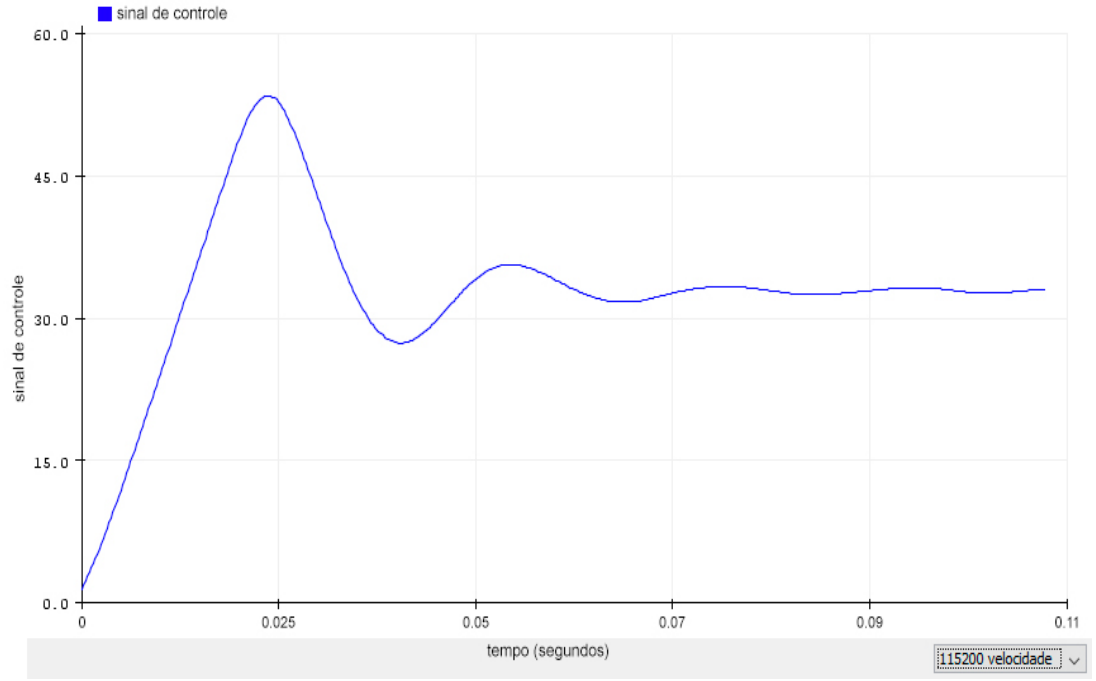
Analisando também o sinal de controle do sistema na Figura 37, percebe-se que, tanto para o ambiente ideal quanto para o ambiente real, o sinal apresenta forma de onda e comportamento parecidos, reafirmando novamente a validade do controlador.

É importante ressaltar ainda, que, para todas as respostas dos sistemas físicos, o motor demorou alguns segundos para a sua inicialização, o que ocasionou as curvas se iniciarem depois de $0s$.

Todos os códigos utilizados neste projeto e apresentados nos Apêndices estão disponíveis na plataforma GitHub no seguinte diretório: https://github.com/Pumpkin4/DC-motor-Control.git.



(a) Resposta ideal.



(b) Resposta do sistema físico.

Figura 37 – Sinal de controle do sistema com uso do Lugar das Raízes.
Fonte: Acervo pessoal.

6 CONCLUSÃO

Conforme era de se esperar, devido ao valor considerado do coeficiente de amortecimento ($0 < \zeta < 1$) para todos os controladores, todos os controles realizados provaram que o sistema é um sistema subamortecido, isto é, que apresenta uma oscilação senoidal no regime transitório e um decaimento exponencial ao longo do tempo.

Em todas as execuções do sistema físico de ambos os controladores, foi observável que o sinal de resposta apresenta dispersão em torno do valor original, porém que essa variação não afeta o desempenho da montagem, uma vez que a curva sempre tende a seguir o *setpoint*. Obviamente que alguns destes resultados, principalmente os do Lugar das Raízes com valores de referências mais baixos, precisam ser melhorados de alguma forma viável e otimizadora, para tentar diminuir a oscilação dos sinais de saída dos circuitos e a acomodação, no mais em torno possível do referencial.

Dentro do objetivo proposto, conclui-se que o sistema de controle de baixo custo desenvolvido se mostrou ser bem-sucedido e eficiente, já que, ao se variar o valor de referência (*setpoint*), o sistema sempre se estabiliza em torno do referencial ao longo do regime permanente, o que caracteriza o conceito de controle de sistemas. Dessa forma, tal sistema já consegue ser replicado em vários outros sistemas físicos para que cada grupo de alunos de uma sala consiga utilizar, pelo menos, uma montagem durante as aulas práticas de controle do curso.

Conforme avanço nas etapas do trabalho, concluiu-se que não é recomendado utilizar o Arduino Nano para a realização de um projeto desse tipo, uma vez que ele não oferece o suporte necessário para a conexão entre *hardware* e MATLAB.

Como sugestões para trabalhos futuros, têm-se a tentativa de montagem do sistema físico de controle utilizando um Arduino Mega em lugar de um Uno para construir os códigos apresentados nos Apêndices D e E no formato de diagrama de blocos dentro do Simulink, de forma que se consiga padronizar as montagens e características para ambos os resultados (ideal e real). Como o Mega apresenta dois pinos TX e RX em cada um, a comunicação serial entre Simulink e Arduino via blocos prontos do *software* consegue ser realizada, já que um par de pinos fica para o envio e o outro para o recebimento dos dados.

Também é válido sugerir, utilizando-se o Arduino Mega, testar dar a partida no motor, nos testes no sistema físico, após ele já estar funcionando por alguns segundos, de forma a tentar reduzir o *overshoot* obtido nos gráficos dos testes reais.

Uma outra sugestão seria tentar utilizar um *encoder* de maior resolução para a leitura de RPM do motor, visando obter leituras mais estáveis. Para além do *encoder*, é necessária uma análise mais criteriosa dos ruídos encontrados nos ensaios dos sistemas físicos.

Ainda, é válida a aplicação de outras técnicas de controle na montagem, para que seja atendida, ao máximo possível, a necessidade prática de outras disciplinas de teoria de controle do curso. Além disso, pode ser uma boa ideia fazer a soldagem dos componentes eletrônicos físicos em uma placa de fenolite perfurada, no intuito de manter a montagem esteticamente melhor e mais estável.

REFERÊNCIAS

ALBAGUL, A.; KHALIFA, O. O. Matlab and simulink in mechatronics education. 2005. Citeseer, 2005. Citado na página 40.

ÅSTRÖM, K. J.; HÄGGLUND, T. The future of pid control. *Control engineering practice*, 2001. Elsevier, v. 9, n. 11, p. 1163–1175, 2001. Citado na página 20.

BRITO, A. B. B. Integração entre o matlab/simulink e a plataforma arduino para desenvolvimento por linguagem gráfica e hardware-in-the-loop (hil) de algoritmos de controle para levitação magnética de uma esfera. 2016. Universidade Estadual Paulista (UNESP), 2016. Citado 2 vezes nas páginas 31 e 32.

CASTRUCCI, P. *Controle automático: teoria e projeto*. [S.l.: s.n.], 1969. Citado na página 21.

CECY, C.; OLIVEIRA, G. A. d.; COSTA, E. M. d. M. B. Metodologias ativas: aplicações e vivências em educação farmacêutica. *Brasília: Abenfarbio*, 2013. 2013. Citado na página 19.

CIA, A. e. *Como medir a rotação de um motor com o sensor de velocidade LM393*. 2016. Acesso em: 4 set. 2022. Disponível em: <https://www.arduinoecia.com.br/sensor-de-velocidade-lm393-arduino/>. Citado na página 37.

COLL, C. *Psicologia e currículo: uma aproximação psicopedagógica à elaboração do currículo escolar*. In: . 5th. ed. São Paulo: Ática, 2006. Citado na página 16.

FACCIN, F. *Abordagem inovadora no projeto de controladores PID*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2004. Citado 4 vezes nas páginas 20, 23, 26 e 27.

FARIAS, P. A. M. d.; MARTIN, A. L. d. A. R.; CRISTO, C. S. Aprendizagem ativa na educação em saúde: percurso histórico e aplicações. *Revista Brasileira de Educação Médica*, 2015. SciELO Brasil, v. 39, p. 143–150, 2015. Citado na página 19.

FELDER, R. M. Reaching the second tier: Learning and teaching styles in college science education. *J. College Science Teaching*, 1993. v. 23, p. 286–290, 1993. Citado na página 16.

FILIPEFLOP. *Motor DC com Driver Ponte H L298N*. 2013. Acesso em: 27 ago. 2022. Disponível em: <https://www.filipeflop.com/blog/motor-dc-arduino-ponte-h-l298n/>. Citado 2 vezes nas páginas 35 e 36.

FRANCO, I. C. et al. Desenvolvimento de um sistema supervisório utilizando matlab/simulink® para uma planta didática de controle de nível via arduíno. *The Journal of Engineering and Exact Sciences*, 2021. v. 7, n. 3, p. 12676–01, 2021. Citado na página 34.

FREITAS, A. d. O.; JESUS, T. R. d. Kit didático controle motor cc usando técnicas de controle dinâmico. *Instituto Federal de Educação, Ciência e Tecnologia de Goiás*, 2012. 2012. Citado na página 48.

GOMES, S. C. P. *Lugar geométrico das raízes incremental e sua aplicação na sintonia de controladores PID*. Dissertação (Mestrado) — Centro Universitário do Instituto Mauá de Tecnologia, 2009. Citado 4 vezes nas páginas 23, 24, 25 e 26.

GOODWIN, G. C. et al. *Control system design*. [S.l.]: Prentice Hall Upper Saddle River, 2001. Citado na página 26.

GOODWIN, G. C. et al. Emulation-based virtual laboratories: A low-cost alternative to physical experiments in control engineering education. *IEEE Transactions on Education*, 2011. v. 54, p. 48–55, 2011. Citado na página 16.

JUNIOR, W. A. A.; CRUZ, R. R. W. Introdução à transformada z. *Universidade Federal do Paraná, Paraná, Relatório*, 2010. 2010. Citado 2 vezes nas páginas 27 e 28.

KNOSPE, C. Pid control. *IEEE Control Systems Magazine*, 2006. IEEE, v. 26, n. 1, p. 30–31, 2006. Citado na página 20.

LOURENÇO, J. Sintonia de controladores pid. *Escola Superior de Tecnologia*, 1997. 1997. Citado na página 24.

NATALMAKERS. *Dispositivo: conhecendo as partes do Arduino UNO*. 2018. Acesso em: 31 ago. 2022. Disponível em: <https://http://www.natalmakers.com/ dispositivo-conhecendo-as-partes-do-arduino-uno/>. Citado na página 39.

NISE, N. Engenharia de sistemas de controle. *LTC Editora*, 2012. 2012. Citado na página 22.

OGATA, K. *Discrete-time control systems*. [S.l.]: Prentice-Hall, Inc., 1995. Citado 3 vezes nas páginas 27, 28 e 50.

OGATA, K. *Engenharia de controle moderno*. 5th. ed. São Paulo: Pearson Prentice Hall, 2010. Citado 6 vezes nas páginas 20, 22, 23, 24, 25 e 26.

PHILLIPS, C. L.; PARR, J. M.; RISKIN, E. A. *Signals, Systems, and Transforms*. 4th. ed. Nova Jersey: Prentice Hall, 2008. Citado na página 27.

PINTO, J. E. M. G. *Aplicação prática do método de sintonia de controladores PID utilizando o método do relé com histerese*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2014. Citado 3 vezes nas páginas 23, 24 e 26.

PLC, J. C. *Control PID para Motor DC con encoder - Arduino*. 2022. Disponível em: <https://youtu.be/bl2k6eXDAGM>. Acesso em: 24 nov. 2022. Citado na página 59.

PROJETADO, M. *Inversor de tensão - conversor CC para CA*. 2017. Acesso em: 27 ago. 2022. Disponível em: <https://mundoprojetado.com.br/inversor-de-tensao-conversor-cc-para-ca/>. Citado na página 35.

PROJETADO, M. *Ponte H - O que é e como funciona*. 2018. Acesso em: 27 ago. 2022. Disponível em: <https://mundoprojetado.com.br/ponte-h-o-que-e-e-como-funciona/>. Citado na página 35.

REGUERA, P. et al. A low-cost open source hardware in control education. case study: Arduino-feedback ms-150. *IFAC-PapersOnLine*, 2015. Elsevier, v. 48, n. 29, p. 117–122, 2015. Citado na página 33.

ROCHA, R. *Estimação dos parâmetros de um motor CC*. 2021. Material de aula da disciplina de Acionamentos Elétricos do curso de engenharia de controle e automação da UFOP. Citado 3 vezes nas páginas 29, 31 e 45.

SILVA, G. J.; DATTA, A.; BHATTACHARYYA, S. P. *PID controllers for time-delay systems*. [S.l.]: Springer, 2005. Citado 2 vezes nas páginas 21 e 27.

SILVA, S. Aprendizagem ativa. *Revista Ensino. Editora Segmento. Edição*, 2013. v. 257, 2013. Citado na página 19.

TECHTUDO. *Controle luzes, motores e muito mais com o Arduino*. 2011. Acesso em: 4 set. 2022. Disponível em: <https://www.techtudo.com.br/tudo-sobre/arduino.html>. Citado na página 39.

TECNOLOGIA, H. *O que é Encoder? Para que serve? Como escolher? Como interfacear?*. 2017. Acesso em: 4 set. 2022. Disponível em: <https://www.hitecnologia.com.br/blog/o-que-\'e-encoder-para-que-serve-como-escolher-como-interfacear/?rdst_srcid=96>. Citado 2 vezes nas páginas 36 e 38.

The MathWorks Inc. 1994–2022. Acesso em: 2022-08-02. Disponível em: <https://www-.mathworks.com/>. Citado 2 vezes nas páginas 17 e 50.

The MathWorks Inc. Design. Simulate. Deploy. 1994–2022. Acesso em: 2022-09-02. Disponível em: <https://www.mathworks.com/products/simulink.html?s_tid=hp_ff_p_simulink>. Citado na página 40.

The MathWorks Inc. Math. Graphics. Programming. 1994–2022. Acesso em: 2022-09-02. Disponível em: <https://www.mathworks.com/products/matlab.html?s_tid=hp_ff_p_matlab>. Citado na página 39.

THE MATHWORKS INC. *PID Controller Tuning in Simulink*. 2009. Acesso em: 2022-10-19. Disponível em: <https://www.mathworks.com/help/slcontrol/gs/automated-tuning-of-simulink-pid-controller-block.html>. Citado na página 50.

UFOP. *Biblioteca Digital de Trabalho de Conslusão de Curso*. 2022. Acesso em: 2022-08-10. Disponível em: <https://www.monografias.ufop.br/handle/35400000/73>. Citado na página 18.

UNBEHAUEN, H. *Control systems, robotics and automation*. [S.l.]: Eolss Publishers Company Limited Oxford, 2009. Citado na página 21.

UYANIK, I.; CATALBAS, B. A low-cost feedback control systems laboratory setup via arduino–simulink interface. *Computer Applications in Engineering Education*, 2018. Wiley Online Library, v. 26, n. 3, p. 718–726, 2018. Citado 3 vezes nas páginas 17, 32 e 33.

VALENTE, J. A. Aprendizagem ativa no ensino superior: a proposta da sala de aula invertida. *Notícias, Brusque*, 2013. 2013. Citado na página 16.

VLADCONTROL. *Ponte H L298N: Módulo de controle de motores DC*. 2019. Acesso em: 27 ago. 2022. Disponível em: <http://www.vladcontrol.com.br/arduino-basico/ponte-h-l298n/>. Citado na página 36.

APÊNDICE A – SCRIPT COMENTADO: LEVANTAMENTO DOS PARÂMETROS DO MOTOR CC - PARTE 1

```
//Definindo os pinos Arduino que sao ligados a entrada da Ponte H
int vel = 9;
int pwm = 0;

//Definindo variaveis globais
float val0 = 0.0;
float val2 = 0.0;
float val5 = 0.0;
float corr = 0.0;
float tensao = 0.0;

//Definindo o baudrate
void setup()
{
 Serial.begin(9600);
}

//Criando a logica para controlar a velocidade de rotacao do motor
void loop()
{
 if (pwm == 0){ //seta pwm so uma vez no loop
  pwm = 180; //variar esse valor conforme a velocidade desejada
  analogWrite(vel, pwm);
 }

//Convertendo a leitura em PWM para tensao em volts
val0 = (analogRead(A0)*5.0)/(1023.0);
val2 = (analogRead(A2)*5.0)/(1023.0);
val5 = (analogRead(A5)*5.0)/(1023.0);

//Printando na serial os valores de tensao em cada porta
Serial.print("\nA0:");
Serial.println(val0,2);

Serial.print("A2:");
Serial.println(val2,2);
```

```
Serial.print("A5:");
Serial.println(val5,2);

//Calculando a corrente e a tensao
corr = (val2 - val0)/1.3; //o resistor SHUNT e de 1,3 Ohms
tensao = val5 - val2;

//Printando na serial os valores de tensao e corrente desejados
Serial.print("\nCorrente:");
Serial.println(corr,2);

Serial.print("Tensao:");
Serial.println(tensao,2);

delay(2000);
}
```

APÊNDICE B – SCRIPT COMENTADO: LEVANTAMENTO DOS PARÂMETROS DO MOTOR CC - PARTE 2

```
//Definindo os pinos Arduino que sao ligados ao Encoder
#define   pinoD0   3
#define   analog   2

//Definindo as variaveis globais
int           rpm;
volatile byte pulsos;
unsigned long timeold;

//Alterando o numero abaixo de acordo com a quantidade de furos do
   disco encoder
unsigned int pulsos_por_volta = 2;

//Definindo os pinos Arduino que sao ligados a entrada da Ponte H
int vel = 9;
int pwm = 0;

//Interrupcao
void contador()
{
  //Incrementa o contador
  pulsos++;
}

//Definindo as configuracoes iniciais
void setup()
{
  Serial.begin(115200);

  pinMode(pinoD0, INPUT);
  pinMode(analog,  INPUT);

  //Aciona o contador a cada pulso
  attachInterrupt(digitalPinToInterrupt(pinoD0), contador, FALLING);
  pulsos = 0;
  rpm = 0;
  timeold = 0;
```

```
  //Criando a logica para controlar a velocidade de rotacao do motor
  analogWrite(vel, 110);
}

//Definindo o loop infinito
void loop()
{
  //Atualiza o contador a cada segundo
  if (millis() - timeold >= 1000)
  {
    //Desabilita a interrupcao durante o calculo
    detachInterrupt(digitalPinToInterrupt(pinoD0));

    //Calculo de RPM
    rpm = ((60 * 1000 / pulsos_por_volta ) / (millis() - timeold)) *
       pulsos;
    timeold = millis();
    pulsos = 0;

    //Mostra o valor de RPM no serial monitor
    Serial.print("RPM = ");
    Serial.println(rpm, DEC);

    //Habilita a interrupcao
    attachInterrupt(digitalPinToInterrupt(pinoD0),contador, FALLING);
  }
}
```

APÊNDICE C – SCRIPT COMENTADO: PLOTAGEM DO LUGAR DAS RAÍZES

```
clc
clear all;
close all;

//Passando os parametros da funcao de transferencia do motor:
K = 0.00629; %ganho
num = K;
den = [0.00000000452 0.000000955 0.0000427];

//Gerando a funcao de transferencia do motor:
Glr = tf(num,den);

//Gerando o Lugar das Raizes do sistema em tempo continuo:
rlocus(Glr);

//Discretizando a funcao de transferencia:
Gz = c2d(Glr,0.0038);

//Gerando o Lugar das Raizes do sistema em tempo discreto:
rlocus(Gz);

//Avanco de Fase:
//Incluindo ao sistema o compensador por avanco de fase calculado:
C = tf([1 -0.783],[1 -0.929],0.0038);

//Plotando o Lugar das Raizes com a presenca do compensador:
rlocus(C*Gz,0:0.001:1);

//Obtendo a funcao de transferencia do avanco de fase:
C = C*0.006;

//Plotando a resposta ao degrau unitario do novo sistema:
step(feedback(C*Gz,1));

//Atraso de Fase:
//Fazendo uso da funcao de transferencia discretizada e calculando o
   valor do numerador e do denominador:
```

```
valornum = (7.738 * 1) + 5.922;
valorden = (1^2) - (1.355 * 1) + 0.448;
resp = valornum/valorden;

//Substituindo um dos polos dominantes na funcao de transferencia
   discretizada:
polo = 0.7272 + 0.1941j;
nump = (7.738 * polo) + 5.922;
denp = polo^2 - (1.355 * polo) + 0.448;
valor = nump/denp;

//Utilizando a condicao de modulo:
modulo = abs(valor);
Kp = 1/modulo;
```

APÊNDICE D – SCRIPT COMENTADO: IMPLEMENTAÇÃO DO CONTROLE REAL DO PID

```
//Definindo os pinos Arduino que sao ligados ao Encoder:
#define   pinoD0   3
#define   analog   2

//Numero de pontos da media movel:
#define   n        25

//Definindo variaveis globais:
volatile byte pulsos;
unsigned long antigoMillis = 0;
long          intervalo = 100;
int           saidaPWM = 9;
float         setpoint;
float         rpm;
float         rad;

//Declarando variaveis para o filtro de media movel:
long moving_average(); //Funcao para filtro de media movel
int  filtrado; //Recebe o valor original filtrado
int  numbers[n]; //Vetor com os valores para media movel

//Alterando o numero abaixo de acordo com a quantidade de furos do
   disco encoder:
unsigned int pulsos_por_volta = 2;

//Definindo as variaveis gerais do controlador:
float cv;
float cv1;
float erro;
float erro1;
float erro2;

//Definindo os ganhos do controlador:
float Kp = 0.0137;
float Ki = 0.9209;
float Kd = 0.00004318;
float T = 0.0038; //Periodo de amostragem
```

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);

  pinMode(pinoD0, INPUT);
  pinMode(analog,  INPUT);
  pinMode(saidaPWM, OUTPUT);

  //Interrupcao 1 - pino digital 3
  //Aciona o contador a cada pulso
  attachInterrupt(1, contador, FALLING);
}

void loop() {
  unsigned long atualMillis = millis();

  if ((atualMillis - antigoMillis) >= intervalo)
  {
    //Calculo da velocidade em RPM:
    antigoMillis = atualMillis;
    rpm = 10 * pulsos * (60.0 / pulsos_por_volta);
    pulsos = 0;

    //Calculo da velocidade em rad/s:
    rad = rpm * 0.10472;
  }

//Inicio da logica do Controlador:

  //Declarando o setpoint:
  setpoint = 230;

  //Chamando a funcao de media movel criada mais abaixo:
  filtrado = moving_average();

  //Declarando o erro:
  erro = setpoint - filtrado;

  //Executando a equacao de diferencas de um PID discreto:
  cv = cv1 + (Kp + (Kd/T))*erro + (-Kp + (Ki * T) - 2*(Kd/T))*erro1 +
      (Kd/T)*erro2;
```

```
  cv1 = cv;
  erro2 = erro1;
  erro1 = erro;

  //Saturando a saida do PID:
  if (cv > 230.00) {
    cv = 230.0;
  }
  if (cv < 30.0) {
    cv = 30.0;
  }

  //Realimentando a saida do sistema no pino vinculado a ponte H:
  analogWrite(saidaPWM,cv);

  //Plotando o grafico:
  Serial.print(setpoint); //Curva do setpoint
  Serial.print("\t");
  Serial.print(rad); //Curva da velocidade em rad/s
  Serial.print("\t");
  Serial.println(filtrado); //Curva da velocidade com o filtro de
     media movel

  delay(50);
}

//Implementando a interrupcao:
void contador()
{
  //Incrementa contador
  pulsos++;
}

//Implementando a Funcao de Media Movel:
long moving_average()
{
  //Deslocando os elementos do vetor de media movel:
  for (int i = n - 1; i > 0; i--)
  {
    //Posicao inicial do vetor recebe a leitura original:
    numbers[i] = numbers[i - 1];
  }
```

```
  //Armazenando a velocidade em rad/s dentro do vetor criado:
  numbers[0] = rad;

  //Somando os pontos da media movel atraves de um acumulador:
  long acc = 0;
  for (int i = 0; i < n; i++)
  {
    //Fazendo a somatoria do numero de pontos:
    acc += numbers[i];
  }

  //Retornando a media movel:
  return acc/n;
}
```

APÊNDICE E – SCRIPT COMENTADO: IMPLEMENTAÇÃO DO CONTROLE REAL DO LUGAR DAS RAÍZES

```
//Definindo os pinos Arduino que sao ligados ao Encoder:
#define   pinoD0   3
#define   analog   2

//Numero de pontos da media movel:
#define   n        25

//Definindo variaveis globais:
volatile byte pulsos;
unsigned long antigoMillis = 0;
long          intervalo = 100;
int           saidaPWM = 9;
float         setpoint;
float         rpm;
float         rad;

//Declarando variaveis para o filtro de media movel:
long moving_average(); //Funcao para filtro de media movel
int  filtrado; //Recebe o valor original filtrado
int  numbers[n]; //Vetor com os valores para media movel

//Alterando o numero abaixo de acordo com a quantidade de furos do
   disco encoder:
unsigned int pulsos_por_volta = 2;

//Definindo as variaveis gerais do controlador:
float u = 0;
float u1 = 0;
float u2 = 0;
float erro = 0;
float erro1 = 0;
float erro2 = 0;

float T = 0.0038; //Periodo de amostragem

void setup() {
  // put your setup code here, to run once:
```

```
  Serial.begin(115200);

  pinMode(pinoD0, INPUT);
  pinMode(analog,  INPUT);
  pinMode(saidaPWM, OUTPUT);

  //Interrupcao 1 - pino digital 3
  //Aciona o contador a cada pulso
  attachInterrupt(1, contador, FALLING);
}

void loop() {
  unsigned long atualMillis = millis();

  if ((atualMillis - antigoMillis) >= intervalo)
  {
    //Calculo da velocidade em RPM:
    antigoMillis = atualMillis;
    rpm = 10 * pulsos * (60.0 / pulsos_por_volta);
    pulsos = 0;

    //Calculo da velocidade em rad/s:
    rad = rpm * 0.10472;
  }

//Inicio da logica do Controlador:

  //Declarando o setpoint:
  setpoint = 230;

  //Chamando a funcao de media movel criada mais abaixo:
  filtrado = moving_average();

  //Declarando o erro:
  erro = setpoint - filtrado;

  //Executando a equacao do Lugar das Raizes no dominio do tempo
     discreto:
  u = (0.006*erro) - (0.009756*erro1) + (0.003960*erro2) + (1.928*u1)
      - (0.928071*u2);
  u2 = u1;
  u1 = u;
```

```
  erro2 = erro1;
  erro1 = erro;

  //Saturando a saida do Lugar das Raizes:
  if (u > 230.00) {
    u = 230.0;
  }
  if (u < 30.0) {
    u = 30.0;
  }

  //Realimentando a saida do sistema no pino vinculado a ponte H:
  analogWrite(saidaPWM,u);

  //Plotando o grafico:
  Serial.print(setpoint); //Curva do setpoint
  Serial.print("\t");
  Serial.print(rad); //Curva da velocidade em rad/s
  Serial.print("\t");
  Serial.println(filtrado); //Curva da velocidade com o filtro de
     media movel

  delay(50);
}

//Implementando a interrupcao:
void contador()
{
  //Incrementa contador
  pulsos++;
}

//Implementando a Funcao de Media Movel:
long moving_average()
{
  //Deslocando os elementos do vetor de media movel:
  for (int i = n - 1; i > 0; i--)
  {
    //Posicao inicial do vetor recebe a leitura original:
    numbers[i] = numbers[i - 1];
  }
```

```
  //Armazenando a velocidade em rad/s dentro do vetor criado:
  numbers[0] = rad;

  //Somando os pontos da media movel atraves de um acumulador:
  long acc = 0;
  for (int i = 0; i < n; i++)
  {
    //Fazendo a somatoria do numero de pontos:
    acc += numbers[i];
  }

  //Retornando a media movel:
  return acc/n;
}
```