

UNIVERSIDADE FEDERAL DA BAHIA (UFBA)  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E COMPUTAÇÃO (DEEC)

**LABORATÓRIO INTEGRADO VI (ENGC54)**  
**PROFESSORA CRISTIANE CORREA PAIM**

**Projeto - Controle de Velocidade de Esteira**

GABRIEL CORREIA DOS SANTOS (219215605)  
MATEUS MAGALHÃES FISCINA (221118859)

AGOSTO 2024  
SALVADOR - BA

## SUMÁRIO

<b>1. INTRODUÇÃO.....</b>	<b>3</b>
<b>2. OBJETIVOS.....</b>	<b>4</b>
2.1. Objetivo Geral.....	4
2.2. Objetivos Específicos.....	4
<b>3. DESCRIÇÃO DO PROJETO.....</b>	<b>5</b>
3.1. O Arduino Mega.....	6
3.2. O Shield L293d Driver Ponte H.....	8
3.3. Sensor + Disco Encoder - Sensor de Velocidade.....	10
3.4. Motor de Corrente Contínua com Caixa de Redução.....	12
3.5. Display LCD 16x2 com I2C soldado.....	15
3.6. Outros Elementos Utilizados.....	17
3.6.1. Potenciômetro.....	17
3.6.2. Chave HH Alavanca MTS 101.....	18
3.6.3. Conector Plug P4 com Borne.....	19
<b>4. OBTENÇÃO DO MODELO TEÓRICO DO MOTOR CC.....</b>	<b>20</b>
<b>5. ENSAIO EM MALHA ABERTA, EXTRAÇÃO DO MODELO E PROJETO DO CONTROLADOR.....</b>	<b>24</b>
5.1. Aquisição dos Dados do Modelo em Malha Aberta.....	24
5.2. Obtenção da Função de Transferência (Modelo do Motor CC).....	27
5.3. Projeto do Controlador PI.....	28
5.4. Análise de Perturbações.....	31
<b>6. CONEXÕES INTERNAS DA ESTEIRA.....</b>	<b>36</b>
<b>7. IMPLEMENTAÇÃO E RESULTADOS.....</b>	<b>37</b>
7.1. Modelagem e Impressão do Projeto.....	37
7.2. Resultados Obtidos.....	41
7.3. Dificuldades e Possibilidades de Melhoria.....	42
<b>8. CONCLUSÃO.....</b>	<b>47</b>
<b>9. REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>48</b>
<b>10. ANEXO 1 - CÓDIGO DO MATLAB.....</b>	<b>50</b>
<b>11. ANEXO 2 - CÓDIGO DO ARDUINO.....</b>	<b>52</b>

## 1. INTRODUÇÃO

No contexto da automação e controle de sistemas, o desenvolvimento de protótipos que integram sensores, atuadores e algoritmos de controle tem se tornado cada vez mais relevante. Este relatório descreve o projeto e a implementação de um sistema de controle de velocidade para uma esteira, utilizando a plataforma Arduino. O objetivo principal do projeto é demonstrar a aplicação prática de técnicas de controle, especificamente o controle proporcional-integral-derivativo (PID), na regulação da velocidade de um motor de corrente contínua (CC) que movimenta a esteira.

O projeto envolve a construção de uma estrutura física impressa em 3D, que serve como base para o motor e o sistema de medição. Um encoder é utilizado para monitorar a velocidade do motor, fornecendo feedback em tempo real ao controlador PID. Este controlador ajusta continuamente o sinal de controle enviado ao motor, garantindo que a velocidade da esteira permaneça constante, apesar das variações na carga ou nas condições operacionais.

Ao integrar componentes de hardware e software, o projeto visa não apenas o controle preciso da velocidade da esteira, mas também a compreensão e a aplicação dos conceitos fundamentais de controle em sistemas dinâmicos. A realização do protótipo oferece uma oportunidade prática de explorar as capacidades do Arduino e de otimizar o desempenho de sistemas automatizados através do uso eficaz de algoritmos de controle.

## 2. OBJETIVOS

### 2.1. Objetivo Geral

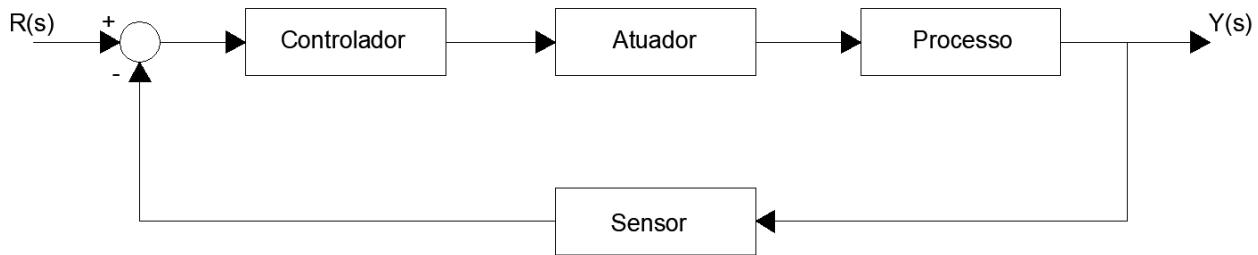
Desenvolver um projeto de controle de velocidade de uma esteira através de um controlador PID utilizando o Arduino Mega como microcontrolador.

### 2.2. Objetivos Específicos

1. Construir um sistema funcional de controle de velocidade para uma esteira utilizando um motor de corrente contínua (CC) e um Arduino;
2. Integrar um encoder para monitoramento da velocidade do motor e garantir a precisão do feedback;
3. Projetar e implementar um controlador PID para ajustar a velocidade do motor com base no feedback do encoder;
4. Modelar o sistema de controle da esteira, entendendo seu comportamento dinâmico e identificando as características principais que afetam a velocidade;
5. Realizar testes experimentais para avaliar a performance do sistema e ajustar os parâmetros do PID conforme necessário;
6. Compreender e aplicar conceitos teóricos de controle proporcional-integral-derivativo em um contexto prático;
7. Projetar e imprimir a estrutura física da esteira utilizando técnicas específicas de modelagem e impressão 3D.

### 3. DESCRIÇÃO DO PROJETO

Neste capítulo será apresentada a descrição completa de todos os elementos presentes na estrutura física e de controle do projeto. A imagem a seguir, exemplifica de forma resumida a ideia do projeto que será mais desenvolvida no capítulo sobre a Modelagem do Projeto.



*Figura 3.1 - Diagrama Simplificado de Bloco do Sistema de Controle do Projeto Proposto*

A partir da compreensão do sistema apresentado na figura 1, é possível perceber que o Controlador PID será implementado através de um Arduino Mega; o atuador será implementado por um Shield L293d Driver Ponte H acionado por PWM; o sensor será realizado através de um conjunto Encoder + Sensor Óptico de Velocidade; e o Processo nada mais é do que o Motor CC com Caixa de Redução do próprio conjunto do Arduino.

Nos próximos itens da descrição do projeto será aprofundado um pouco mais as especificações técnicas de cada um dos elementos aqui apresentados. Antes, porém, como será visualizado no desenvolvimento da impressão do projeto, para facilitar a observação dos resultados, será também implementado um LCD via protocolo I2C (o que reduz a quantidade de cabos a serem conectados no módulo LCD).

### **3.1. O Arduino Mega**

A plataforma Arduino é uma solução de código aberto, de baixo custo e fácil manipulação, com software baseado na linguagem C/C++. Originalmente planejada para auxiliar estudantes italianos de design e arte em seus projetos acadêmicos (HOCHENBAUM, 2013), a plataforma rapidamente ganhou destaque na comunidade acadêmica global.

Atualmente, o Arduino é amplamente utilizado em todo o mundo por universitários, desenvolvedores, profissionais e entusiastas, interessados em desenvolver e prototipar seus próprios projetos.

Existem diversas versões de hardware Arduino disponíveis, com variações que incluem diferentes tipos de controladores e números de portas, adaptando-se a uma ampla gama de projetos e aplicações.

Entre as versões mais comuns do Arduino, destacam-se o Arduino UNO, o Arduino MEGA e o Arduino MICRO. O Arduino UNO é a versão mais popular, ideal para iniciantes, com um microcontrolador ATmega328P, 14 pinos digitais, 6 entradas analógicas e uma interface USB que facilita a programação e a conexão com o computador. É amplamente utilizado em projetos de pequena a média escala.

O Arduino MEGA, por sua vez, é indicado para projetos mais complexos que exigem um maior número de pinos e maior capacidade de processamento. Equipado com o microcontrolador ATmega2560, o MEGA possui 54 pinos digitais, 16 entradas analógicas e uma maior quantidade de memória, tornando-o adequado para aplicações que demandam mais recursos.

O Arduino MICRO é uma versão compacta, ideal para projetos onde o espaço é limitado. Ele utiliza o microcontrolador ATmega32U4 e possui 20 pinos digitais, 12 entradas analógicas e uma interface USB integrada. Devido ao seu tamanho reduzido, o MICRO é frequentemente utilizado em projetos de wearables, dispositivos portáteis e sistemas embarcados onde a miniaturização é essencial.

Cada uma dessas versões do Arduino oferece características específicas que permitem sua adaptação a uma ampla variedade de projetos, desde protótipos simples até sistemas mais elaborados e exigentes.

No projeto a ser desenvolvido foi optado por um Arduino Mega devido a sua maior quantidade de pinos digitais e analógicos, o que permite o uso tanto do Atuador quanto do Sensor e do LCD sem grandes problemas de competição entre as portas (principalmente as portas de comunicação serial). O Shield L293d Driver Ponte H, no momento da conexão, fica exatamente em cima das portas SCL e SDA do Arduino UNO, impossibilitando a conexão de um módulo LCD via protocolo serial I2C.

A figura e a tabela a seguir apresentam os dados mais técnicos do controlador a ser utilizado neste projeto.

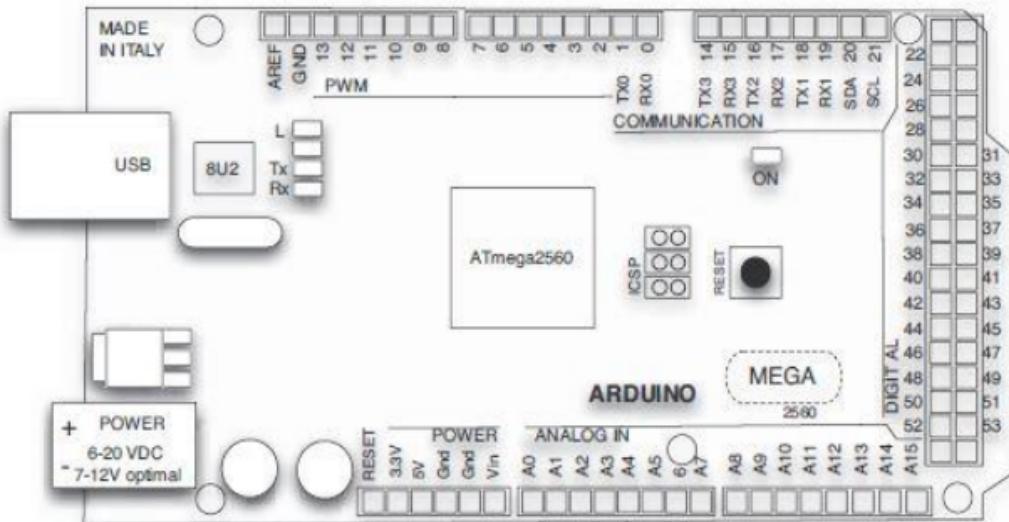


Figura 3.1.1 - Esquemático do Arduino Mega (MCROBERTS<sup>1</sup>, 2015)

<sup>1</sup> MCROBERTS, Michael. **Arduino básico**. Novatec Editora, 2011. Disponível em: [https://edisciplinas.usp.br/pluginfile.php/4287597/mod\\_resource/content/2/Ardu%C3%ADno%20B%C3%A1sico%20-%20Michael%20McRoberts.pdf](https://edisciplinas.usp.br/pluginfile.php/4287597/mod_resource/content/2/Ardu%C3%ADno%20B%C3%A1sico%20-%20Michael%20McRoberts.pdf) - Acesso em 25 de julho de 2024.

Especificações do Arduino Mega	
Microcontrolador	ATmega2560 (Arquitetura AVR 8bits)
Clock	16MHz
Memória Flash	256kB
SRAM	8kB
EEPROM	4kB
Tensão Operacional	5V
Tensão Recomendada de Entrada	7 - 12V
Tensão Máxima de Entrada	6 - 20V
Corrente Pinos E/S (I/O)	40mA
Pinos E/S (I/O) Digitais	54
Pinos Entrada Analógicas	16
Portas Seriais	4
Pinos para Interrupção Externa	6
Pinos de PWM	15

Microcontrolador	ATmega2560 (Arquitetura AVR 8bits)
Clock	16MHz
Memória Flash	256kB
SRAM	8kB
EEPROM	4kB
Tensão Operacional	5V
Tensão Recomendada de Entrada	7 - 12V
Tensão Máxima de Entrada	6 - 20V
Corrente Pinos E/S (I/O)	40mA
Pinos E/S (I/O) Digitais	54
Pinos Entrada Analógicas	16
Portas Seriais	4
Pinos para Interrupção Externa	6
Pinos de PWM	15

*Tabela 3.1.1 - Dados Técnicos do Arduino Mega (Autoria Própria)*

### 3.2. O Shield L293d Driver Ponte H

Neste trabalho, foi utilizado um circuito de ponte H para garantir o funcionamento seguro do motor de corrente contínua sem sobrecarregar o hardware do Arduino. O circuito de ponte H é essencial, pois a corrente exigida pelo motor é superior àquela que as portas de saída do Arduino podem fornecer, impossibilitando uma conexão direta entre o motor e o Arduino.

Assim, a alimentação da máquina rotativa de corrente contínua foi dividida em dois circuitos: um com uma fonte de alimentação externa, responsável por fornecer a

tensão adequada ao motor, e outro circuito para controlar o comportamento do motor por meio do Arduino.

Para este projeto, foi empregado o Motor Shield L293D, um módulo que integra dois circuitos L293D capazes de controlar até quatro motores de corrente contínua de forma independente. Além disso, o shield inclui um registrador de deslocamento de oito bits (74HC595), que serve como expansor de portas para o Arduino, permitindo maior flexibilidade no controle de múltiplos motores.

O Motor Shield L293D também oferece a capacidade de controlar até dois servos motores ou dois motores de passo, ampliando as possibilidades de aplicação em projetos que envolvem diferentes tipos de atuadores.

Assim como realizado para o Arduino Mega, a figura e a tabela a seguir apresentam os dados técnicos do Shield L293D Driver Ponte H utilizado no projeto:

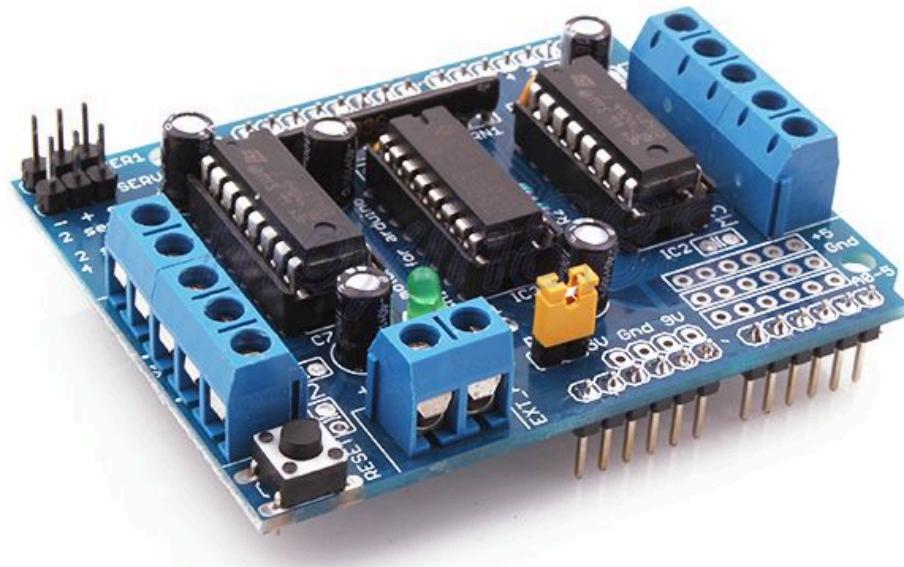


Figura 3.2.1 - Esquemático do Shield L293D Driver Ponte H (Arduino e Cia<sup>2</sup>, 2014)

---

<sup>2</sup> Arduino e Cia. **Controlando motores DC com o Arduino Motor Shield L293D.** Publicado em 3 de julho de 2014. Disponível em: <https://www.arduinoecia.com.br/arduino-motor-shield-l293d/> - Acesso em 24 de julho de 2024.

Especificações do Shield L293D Driver Ponte H	
Chip	293D
Possibilidade de Controle	4 Motores DC, 2 Motores de Passos ou 2 Servos Motores
Tensão de Operação	4 - 12V
Tensão de Saída	4,5 - 36V
Corrente de Saída (Por Canal)	600mA
Corrente de Pico (Por Canal)	1,2A (com Proteção Térmica)

Possibilidade de Controle	4 Motores DC, 2 Motores de Passos ou 2 Servos Motores
Tensão de Operação	4 - 12V
Tensão de Saída	4,5 - 36V
Corrente de Saída (Por Canal)	600mA
Corrente de Pico (Por Canal)	1,2A (com Proteção Térmica)

*Tabela 3.2.1 - Dados Técnicos do Shield L293D Driver Ponte H (Autoria Própria)*

### 3.3. Sensor + Disco Encoder - Sensor de Velocidade

O sensor de velocidade encoder é um módulo eletrônico composto por quatro pinos: entrada de 5V, terra, saída digital (D0) e saída analógica (A0). Os pinos de alimentação do sensor foram conectados à saída do servo motor da ponte H, recebendo uma tensão de 5V. O pino de saída digital (D0) foi conectado a uma das entradas digitais do Arduino, enquanto o pino de saída analógica não foi utilizado neste estudo (o código completo do projeto estará disponibilizado ao final deste relatório).

O sensor de medição de velocidade realiza a leitura das perfurações do disco encoder, interpretando a luz e sua ausência como uma sequência de sinais, alternando entre níveis lógicos alto e baixo. Esses sinais são convertidos em uma onda quadrada que representa o número de rotações por unidade de tempo. A onda gerada é então transmitida pela saída digital (D0) para o microcontrolador do Arduino.

Já o disco encoder, um dispositivo eletromecânico, é amplamente utilizado para determinar o número de rotações de um objeto e o posicionamento de equipamentos específicos. Este dispositivo é composto por um disco com perfurações que, ao ser rotacionado e submetido a um feixe de luz infravermelha constante, permite a

passagem da luz através das perfurações, enquanto as partes sólidas do disco bloqueiam a luz. Esse processo converte a posição em um sinal digital.

Neste projeto, foi utilizado um disco encoder simples, com 20 perfurações, conectado ao eixo da caixa de redução do motor CC. A seguir, é possível ver tanto as figuras do disco encoder utilizado e do sensor de velocidade encoder, assim como as informações técnicas na tabela.



Figura 3.3.1 - Disco Encoder (FERMARC<sup>3</sup>, 2024)

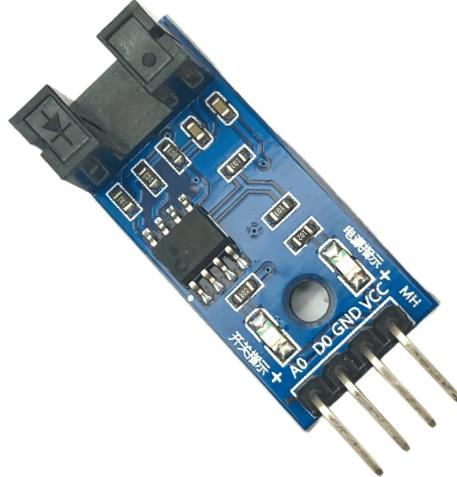


Figura 3.3.2 - Sensor de Velocidade Encoder (Casa da Robótica<sup>4</sup>, 2024)

<sup>3</sup> Loja FERMARC. **Disco Encoder para Sensor de Velocidade.** Disponível em: <https://www.fermarc.com/disco-encoder-para-sensor-de-velocidade> - Acesso em 28 de julho de 2024.

<sup>4</sup> Casa da Robótica. **Sensor de Velocidade | Módulo Encoder Acoplador óptico.** Disponível em: <https://www.casarobotica.com/sensores-e-modulos/sensores/movimento-e-proximidade/sensor-de-velocidade-modulo-encoder-acoplador-optico> - Acesso em 28 de julho de 2024.

Especificações do Sensor de Velocidade | Módulo Encoder Acoplador Óptico

Chip	LM393
Tensão de Operação	3,3 - 5V
Saída Digital	1
Saída Analógica	1
Indicador de Saída Digital	LED
Indicador de Tensão	LED
Espaço entre Leitores	5mm
Dimensões (CxLxA)	32x14x12mm

*Tabela 3.3.1 - Dados Técnicos do Sensor de Velocidade - Módulo Encoder (Autoria Própria)*

### 3.4. Motor de Corrente Contínua com Caixa de Redução

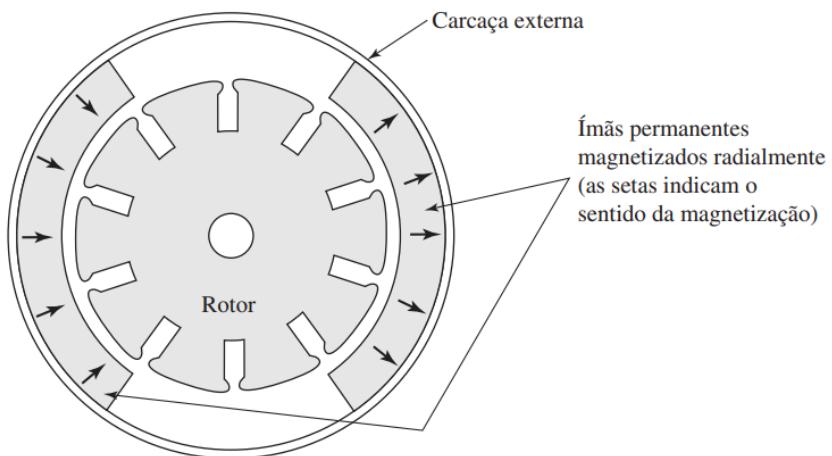
Motores elétricos são máquinas rotativas que convertem energia elétrica em energia mecânica, baseando-se no princípio da conservação de energia. Esses motores são compostos, essencialmente, por dois elementos principais: o estator, que é a parte fixa, e o rotor, que é a parte móvel.

O funcionamento das máquinas elétricas rotativas é explicado pela Lei de Lenz-Faraday. O motor elétrico opera através do fornecimento de energia elétrica ao estator, que gera um campo magnético variável no tempo, fazendo com que o estator atue como um eletroímã. Esse campo magnético induz a movimentação do rotor, que segue a variação do campo gerado.

Os motores elétricos podem ser classificados em três categorias principais: motores de corrente contínua (CC), motores de corrente alternada (CA) e motores especiais. Neste projeto, foi selecionado um motor de corrente contínua (CC), sendo esse tipo de máquina detalhado a seguir.

Máquinas rotativas de corrente contínua operam com alimentação de corrente contínua, resultando em um sistema com campo magnético estável, embora o fluxo magnético seja variável. O campo magnético gerado é perpendicular à superfície do rotor em todas as áreas diretamente sob as faces polares, e diminui rapidamente para zero além das bordas dos polos.

Dessa forma, a tensão gerada dentro da máquina é proporcional ao produto do fluxo magnético interno pela velocidade de rotação, ajustado por uma constante que reflete as características construtivas da máquina. A estrutura geral em corte da parte móvel dos motores CC pode ser melhor visualizada na figura a seguir retirada do livro *Máquinas Elétricas* de Fitzgerald e Kingsley (revisado por Stephen D. Umans).



*Figura 3.4.1 - Esquema do Núcleo de um Motor CC (FITZGERALD, KINGSLEY, UMANS<sup>5</sup>, 2014)*

Além do estator e rotor, os motores de corrente contínua incluem um comutador e escovas. Estes motores são frequentemente utilizados em sistemas de malha fechada, quando se busca controle preciso de velocidade, posição e reversão rápida e simples do sentido de rotação. Eles também são aplicados em projetos que requerem

---

<sup>5</sup> FITZGERALD, A. E.; KINGSLEY, C.; UMANS, S. D. *Máquinas Elétricas*. Revisado por Stephen D. Umans. 7. ed. Porto Alegre: AMGH, 2014.

alto torque de partida, como em veículos elétricos de golfe, que necessitam de força suficiente para superar a inércia inicial.

Neste trabalho, foi utilizado um mini motor de corrente contínua com caixa de redução, eixo duplo conforme a figura e especificações técnicas a seguir:



Figura 3.4.2 - Motor CC com Caixa de Redução (SMARTKITS<sup>6</sup>, 2024)

Perceba que, como explicado anteriormente, para esse projeto não é possível utilizar a alimentação do motor CC pela saídas de 3,3V e 5V do Arduino Mega já que, como a tabela de especificações a seguir mostra, a corrente necessária para esse motor é de no mínimo 150mA e o arduino só libera 40mA (ou 50mA se for usado o pino de 3,3V). Para contornar essa situação, o Shield L293D Driver Ponte H será alimentado diretamente por uma fonte de 5V (que alimentará também o motor CC).

---

<sup>6</sup> SMART KITS Motor DC 3-6V com Caixa de Redução e Eixo Duplo. Disponível em: <https://www.smartkits.com.br/motor-dc-3-6v-com-caixa-de-reducao-e-eixo-duplo> - Acesso em 28 de julho de 2024.

Especificações do Motor CC com Caixa de Redução	
Tensão de Operação	3 - 6V
Rotação em 3V	90 ± 10% RPM
Rotação em 6V	200 ± 10% RPM
Corrente Sem Carga em 3V	150mA
Corrente Sem Carga em 6V	200mA
Torque em 3V	0,35kgf/cm
Torque em 6V	0,80kgf/cm
Redução	1,48

Rotação em 3V	90 ± 10% RPM
Rotação em 6V	200 ± 10% RPM
Corrente Sem Carga em 3V	150mA
Corrente Sem Carga em 6V	200mA
Torque em 3V	0,35kgf/cm
Torque em 6V	0,80kgf/cm
Redução	1,48

*Tabela 3.4.1 - Dados Técnicos do Motor CC com Caixa de Redução e Eixo Duplo (Autoria Própria)*

### 3.5. Display LCD 16x2 com I2C soldado

O Display LCD 16x2 com I2C soldado é um módulo de exibição amplamente utilizado em projetos eletrônicos, especialmente em sistemas embarcados, como aqueles baseados em microcontroladores Arduino. Este módulo combina um display LCD de 16 colunas por 2 linhas, capaz de exibir até 32 caracteres simultaneamente, com um módulo I2C já soldado, o que simplifica significativamente a interface de conexão com microcontroladores e microprocessadores.

O display LCD 16x2 permite a exibição de caracteres alfanuméricicos e símbolos personalizados, formados em uma matriz de 5x8 pixels. Além disso, a maioria desses displays inclui retroiluminação para melhorar a visibilidade em condições de baixa luminosidade.

O protocolo de comunicação I2C (Inter-Integrated Circuit) utilizado neste módulo requer apenas dois pinos de controle: SDA (Serial Data) e SCL (Serial Clock). Isso contrasta com a conexão paralela tradicional do display LCD, que normalmente necessita de 7 a 8 pinos. O uso do módulo I2C reduz significativamente o número de

pinos necessários, simplificando a integração com o microcontrolador. Cada dispositivo I2C possui um endereço exclusivo, o que permite a comunicação direta entre o microcontrolador e o display.

A instalação do Display LCD 16x2 com I2C é bastante simples, exigindo apenas a conexão dos pinos de alimentação (VCC e GND) e dos pinos de comunicação (SDA e SCL) ao microcontrolador. Para facilitar o controle do display via I2C, é comum a utilização de bibliotecas específicas, como a "LiquidCrystal\_I2C", que agilizam o processo de programação. Este tipo de display é ideal para uma ampla gama de aplicações em projetos de eletrônica, onde a exibição de informações como medições, menus de interface do usuário, ou mensagens de status é necessária.

A combinação de sua funcionalidade versátil com a facilidade de uso proporcionada pela interface I2C torna o Display LCD 16x2 com I2C soldado uma escolha popular em projetos que exigem uma interface de usuário eficiente e de fácil implementação. A seguir, é possível ver tanto uma figura ilustrativa quanto as informações técnicas deste elemento:



*Figura 3.6.1 - Potenciômetro (ELETROGATE<sup>7</sup>, 2024)*

---

<sup>7</sup> ELETROGATE. **Display LCD 16x2 com Backlight Azul e I2C.** Disponível em: <https://www.eletrogate.com/display-lcd-16x2-i2c-backlight-azul> - Acesso em 20 de junho de 2024.

### Especificações do Display LCD com I2C Soldado

Interface	I2C
Tensão de Trabalho	4,5 - 5,5V
Corrente de Trabalho	1,0mA - 1,5mA
Tensão do LED (Backlight)	1,5 - 5,5V
Corrente do LED (Backlight)	75mA - 200mA
Número de Linhas	2
Número de Colunas	16

*Tabela 3.5.1 - Dados Técnicos do Display LCD com I2C Soldado (Autoria Própria)*

## 3.6. Outros Elementos Utilizados

Como os elementos a partir de agora citados não exigem grandes cuidados ou apresentam grandes complexidades de implementação, este sub-capítulo irá sintetizar todos os outros elementos que não foram especificados anteriormente (até por serem mais voltados a aspectos de melhoria da experiência do usuário do que aspectos de funcionalidade da esteira).

### 3.6.1. Potenciômetro

Um potenciômetro é um tipo de resistor ajustável utilizado para medir e ajustar a resistência elétrica em um circuito. É um componente eletrônico com três terminais e uma resistência interna variável que pode ser alterada manualmente ou automaticamente. Neste projeto, o valor máximo da resistência do potenciômetro é irrelevante (não importa se vai ser ou de  $1\text{k}\Omega$ , ou  $10\text{k}\Omega$  ou  $100\text{k}\Omega$ ), já que seu uso é apenas para criar uma faixa de valores para a criação do PWM que irá controlar o motor.



*Figura 3.6.1 - Potenciômetro (FARMAC<sup>8</sup>, 2024)*

### **3.6.2. Chave HH Alavanca MTS 101**

A chave HH Alavanca MTS 101 é um tipo de interruptor de alavanca projetado para aplicações de controle e comutação em sistemas eletrônicos e elétricos. Elas apresentam dois pinos sendo utilizadas para dois estados, ou ON ou OFF.



*Figura 3.6.2 - Chave HH Alavanca (EletroinfoCia<sup>9</sup>, 2024)*

---

<sup>8</sup> FERMARC. **Potenciômetro Linear 10K - L15.** Disponível em: <https://www.fermarc.com/potenciometro-linear-10k-l20> - Acesso em 20 de junho de 2024.

<sup>9</sup> EletroinfoCia. **Chave Alavanca 2 Polos On/Off.** Disponível em: <https://www.eletroinfocia.com.br/chave-alavanca-2-polos-onoff-mts-101> - Acesso em 20 de junho de 2024.

### 3.6.3. Conector Plug P4 com Borne

O conector Plug P4 é um tipo de conector utilizado principalmente para aplicações de áudio e vídeo. É conhecido por sua construção robusta e pela capacidade de fornecer conexões seguras e confiáveis.



Figura 3.6.3 - Conector Plug P4 (AutocoreRobotica<sup>10</sup>, 2024)

---

<sup>10</sup> AutocoreRobotica. **Adaptador Plug p4 Fêmea com Borne.** Disponível em: <https://www.autocorerobotica.com.br/adaptador-plug-p4-femea-com-borne> - Acesso em 20 de junho de 2024.

#### 4. OBTENÇÃO DO MODELO TEÓRICO DO MOTOR CC

Neste capítulo, será abordado um pouco sobre o processo de obtenção do modelo do motor CC do ponto de vista teórico para que seja comparado com o modelo obtido empiricamente no capítulo 5. No capítulo 2.8 do livro "Engenharia de Sistemas de Controle" do Norman S. Nise (6<sup>a</sup> edição) é apresentada a modelagem do motor de corrente contínua.

É importante lembrar que, como o projeto a ser desenvolvido é do controle da velocidade (ou melhor, a rotação por minuto) de uma esteira através de um sinal de controle PWM gerado pelo Arduino e passado pelo Shield L293d para o motor, todo o trabalho de controle será realizado através de uma entrada de tensão gerando uma rotação na saída (velocidade angular). Dessa forma, a planta  $G(s)$  do diagrama de blocos a seguir apresentará algo na unidade de medida da forma RPM/V ou RPM/PWM.

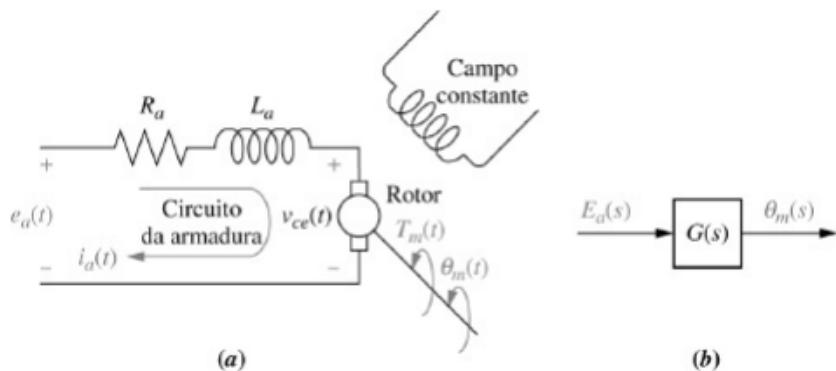


Figura 4.1 - Motor CC: a. Circuito Esquemático, b. Diagrama de Blocos (NISE<sup>11</sup>, 2013)

Da figura anterior, é possível identificar que  $V_{ce}$  é a tensão na armadura [V];  $E_a(t)$  é a força eletromotriz [V],  $R_a$  e  $L_a$  são as resistências [ $\Omega$ ] e indutâncias [H] da armadura respectivamente;  $i_a(t)$  é a corrente que circula na armadura [A];  $T_m(t)$  é o torque desenvolvido pelo motor [Nm]; e  $\theta_m(t)$  é o deslocamento angular do eixo [rad].

Tomando como ponto de partida, entretanto, a equação 7.5 do Fitzgerald, Kingsley e Umans da página 405 e a equação 2.144 da página 128 do Nise, é possível levantar duas equações básicas que descrevem o modelo da máquina CC da forma:

<sup>11</sup> NISE, Norman S. **Engenharia de sistemas de controle**. Tradução e revisão técnica Jackson Paul Matsuura. 6<sup>a</sup> edição. Rio de Janeiro: LTC, 2013

$$\begin{aligned}
 Vce(t) &= Kce \cdot \frac{d\theta m(t)}{dt} \\
 Tm(t) &= Ka \cdot \phi d \cdot ia
 \end{aligned}
 \tag{Eq. 4.1}$$

Onde  $Kce$  é a constante da força eletromotriz da máquina,  $Ka$  é a constante de enrolamento e  $\phi d$  é o fluxo do entreferro de eixo direto. O ponto interessante é que a equação do torque do motor pode ser representada de uma forma diferente, já que o produto do fluxo de eixo direto com a constante do enrolamento também é um valor constante (constante de torque do motor)  $Km$ .

Realizando a análise do circuito do modelo elétrico do motor CC pela Lei de Kirchhoff e a aplicando a lei de Newton para o balanço de torque (sabendo que  $J$  é o momento de inércia e o  $f$  coeficiente de atrito viscoso), é possível obter as equações:

$$\begin{aligned}
 Ea(t) &= Ra \cdot ia(t) + La \frac{dia(t)}{dt} + Vce(t) \\
 Tm(t) - f \frac{d\theta m(t)}{dt} &= J \frac{d^2\theta m(t)}{dt^2}
 \end{aligned}
 \tag{Eq. 4.2}$$

Aplicando a transformada de Laplace nas equações presentes 4.1 e 4.3:

$$\begin{aligned}
 Ea(s) &= Ra \cdot ia(s) + sLa \cdot ia(s) + Vce(s) \\
 Tm(s) &= s^2J\theta m(s) + sf\theta m(s) \\
 Vce(s) &= Kce \cdot s\theta(s) \\
 Tm(t) &= Km \cdot ia
 \end{aligned}
 \tag{Eq. 4.3}$$

Nessa parte, é necessário que seja aplicado um pouco de algebrismo. A ideia é escrever o deslocamento angular em função da tensão que é aplicada na máquina, dessa forma, na terceira equação de 4.3 deverá ser isolado  $\theta(s)$  e isolar  $Vce(s)$  na primeira equação, substituindo o valor obtido da tensão da armadura na equação do deslocamento angular. Dessa forma:

$$\theta(s) = \frac{V_{ce}(s)}{sK_{ce}}$$

$$V_{ce}(s) = Ea(s) - ia(s) \cdot (Ra + sLa)$$

$$\theta(s) = \frac{Ea(s) - ia(s) \cdot (Ra + sLa)}{sK_{ce}}$$

(Eq. 4.4)

Ainda nas equações de 4.3, é possível isolar a corrente da armadura na quarta equação apresentada e inserir a relação da segunda equação que descreve o torque motor.

$$ia = \frac{T_m(t)}{K_m}$$

$$ia = \frac{s^2 J \theta_m(s) + s f \theta_m(s)}{K_m}$$

(Eq. 4.5)

Com isso, é possível substituir a expressão completa da corrente da armadura na equação 4.4 do deslocamento angular.

$$\theta(s) = \frac{Ea(s) - \frac{s^2 J \theta_m(s) + s f \theta_m(s)}{K_m} \cdot (Ra + sLa)}{sK_{ce}}$$

(Eq. 4.6)

Ainda da teoria de circuitos, é sabido que a razão entre a indutância pela resistência é a constante de tempo elétrica ( $\tau_e$ ), assim como pela mecânica clássica newtoniana, a razão entre o momento de inércia e o coeficiente de atrito viscoso é a constante de tempo mecânica ( $\tau_m$ ).

$$\theta(s) = \frac{Ea(s) - \frac{s^2 J \theta_m(s) + s f \theta_m(s)}{K_m} \cdot (Ra + sLa)}{sK_{ce}} = \frac{Ea(s) \cdot Km}{s[KceKm + (Ra + sLa)(f + sJ)]}$$

$$\theta(s) = \frac{Ea(s) \cdot Km}{s[KceKm + fRa(s\tau_e + 1)(s\tau_m + 1)]}$$

(Eq. 4.7)

Na máquina CC entretanto, a indutância é tão pequena que alguns autores da literatura mais convencional chegam a aproximar o modelo de circuito apenas a um resistor de armadura, ou seja, com uma indutância nula, o que resulta também é uma constante elétrica nula. Dessa consideração, é possível obter:

$$\theta(s) = \frac{Ea(s) \cdot Km}{s[KceKm + fRa(stm+1)]} = \frac{Ea(s) \cdot Km}{s[KceKm + fRa(f+sJ)]} = Ea(s) \cdot \frac{1}{s} \left( \frac{Ea(s) \cdot Km}{KceKm + fRa(f+sJ)} \right)$$

(Eq. 4.8)

Se forem chamadas duas variáveis auxiliares para reduzir o modelo em parâmetros constantes, é então obtida uma equação próxima ao que foi introduzido neste capítulo - cujo objetivo é encontrar uma velocidade angular em função da tensão aplicada na máquina:

$$\begin{aligned}\tau &= \frac{Ra}{KceKm + fRa} \\ K &= \frac{Km}{KceKm + fRa}\end{aligned}$$

(Eq. 4.9)

Realizando a substituição das equações de 4.9 na equação 4.8:

$$\theta(s) = \frac{Ea(s)K}{s(\tau s + 1)} \Leftrightarrow \frac{\theta(s)}{Ea(s)} = \frac{K}{s(\tau s + 1)}$$

(Eq. 4.10)

Como o que é requerido não é o deslocamento angular, mas sim a velocidade angular  $W(s)$ , então a expressão da equação 4.10 tem que ser derivada. Dessa forma, obtém-se o modelo matemático do motor de corrente contínua (primeira ordem):

$$\begin{aligned}\frac{W(s)}{Ea(s)} &= s \frac{\theta(s)}{Ea(s)} = s \frac{K}{s(\tau s + 1)} \\ \frac{W(s)}{Ea(s)} &= \frac{K}{\tau s + 1}\end{aligned}$$

(Eq. 4.11)

## 5. ENSAIO EM MALHA ABERTA, EXTRAÇÃO DO MODELO E PROJETO DO CONTROLADOR

Este capítulo aborda o processo de caracterização e controle de um motor de corrente contínua (CC) utilizando uma plataforma Arduino para os ensaios e um controlador proporcional-integral (PI) para o controle do sistema. Primeiramente, realiza-se um ensaio em malha aberta para obter dados experimentais sobre o comportamento do motor, que é modelado como um sistema de primeira ordem. Com esses dados, procede-se à extração do modelo dinâmico do motor, identificando os parâmetros que descrevem sua resposta a sinais de entrada.

Com base no modelo obtido, desenvolve-se e projeta-se um controlador PI com o objetivo de controlar o sistema de forma eficaz, assegurando estabilidade e uma resposta adequada às variações de carga e comandos. Este capítulo detalha cada uma dessas etapas.

### 5.1. Aquisição dos Dados do Modelo em Malha Aberta

Um sinal PWM com amplitude de 5 volts foi aplicado aos terminais do motor. A velocidade desenvolvida pelo motor foi medida com um encoder externo, utilizando um intervalo de amostragem de 0,1 segundos, período que garantiu a integridade dos dados sem degradar significativamente o sinal de saída. Esses dados foram utilizados para gerar um gráfico de tempo versus velocidade, fundamental para a extração dos parâmetros do modelo dinâmico do motor.

O gráfico resultante permite uma análise detalhada da resposta do motor ao sinal PWM, possibilitando a identificação das características necessárias para o desenvolvimento do modelo e subsequente controle do sistema.

A tabela a seguir apresenta os resultados obtidos. Observa-se que os valores de velocidade medidos são superiores aos especificados na ficha técnica do motor. Esse desvio pode ser atribuído a imprecisões na medição pelo sensor de velocidade ou à baixa resolução do disco encoder, que possui apenas 20 pulsos por revolução (como visto nos capítulos anteriores da ficha técnica).

No entanto, esse erro de medição não teve impacto significativo no desenvolvimento do compensador, uma vez que, apesar das discrepâncias nos valores, os dados ainda representam adequadamente o comportamento do sistema.

Resposta da Velocidade (RPM) no Tempo (s)	
Tempo [s]	Velocidade [RPM]
0	0
0,1	330
0,2	480
0,3	540
0,4	550
0,5	570
0,6	540
0,7	460
0,8	450
0,9	450
1,0	480
1,1	480
1,2	510
1,3	515
1,4	580
1,5	600
1,6	450
1,7	600
1,8	570
1,9	540

Tempo [s]	Velocidade [RPM]
0	0
0,1	330
0,2	480
0,3	540
0,4	550
0,5	570
0,6	540
0,7	460
0,8	450
0,9	450
1,0	480
1,1	480
1,2	510
1,3	515
1,4	580
1,5	600
1,6	450
1,7	600
1,8	570
1,9	540

2,0	480
2,1	480
2,2	510
2,3	470
2,4	450
2,5	450
2,6	600
2,7	510
2,8	530
2,9	540
3,0	570
3,1	540
3,2	450
3,3	460
3,4	465
3,5	540
3,6	510
3,7	570
3,8	540
3,9	530
4,0	510
4,1	600
4,2	510
4,3	540
4,4	535

4,5	510
4,6	600
4,7	570
4,8	540
4,9	540
5,0	480

---

Tabela 5.1.1 - Dados da Velocidade pelo Tempo (Autoria Própria)

## 5.2. Obtenção da Função de Transferência (Modelo do Motor CC)

Com base no modelo apresentado no capítulo 4, é possível observar que os principais parâmetros a serem coletados são a constante de tempo  $\tau$  e o ganho do sistema. Ao contrário do que foi discutido em sala sobre a extração de modelos, não foi obtido um valor para o atraso. Isso se deve à alta velocidade do sistema, tornando a determinação desse atraso desnecessária. Em um intervalo de 100 ms, a velocidade aumentou de 0 a 300 rpm, indicando que, mesmo que o atraso fosse modelado, ele seria bastante reduzido e teria uma influência mínima nas respostas fornecidas pelo modelo.

Para determinar o valor de K, é suficiente calcular a razão entre o valor de regime permanente e o valor do degrau aplicado (neste caso, um PWM de 255 de largura). No entanto, devido ao ruído dos dados, foi necessário aplicar um tratamento para obter o valor de regime. Esse tratamento consistiu na aplicação de uma média em um intervalo de tempo entre 0,3 a 5 segundos, descartando-se a parte transitória. O valor encontrado para o regime foi de 520,3 RPM, conforme ilustrado na figura a seguir.

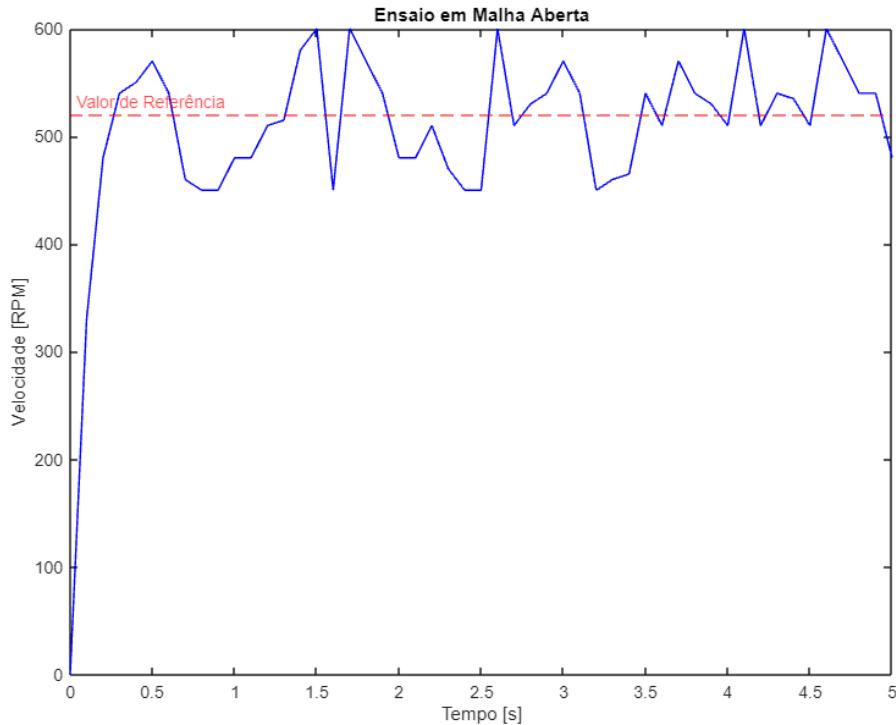


Figura 5.2.1 - Ensaio em Malha Aberta do Sistema Físico Real (Autoria Própria)

Com isso K será a razão entre a velocidade em regime permanente (RPM) e o valor do sinal de controle em PWM aplicado ao sistema, o que resulta na razão 520,3/255 que é igual a 2,04. Com o valor de regime é possível achar qual o valor da velocidade para 63% do valor final e interpolar os dados de forma a achar a constante do tempo. O valor de 63% é 327,8 RPM e interpolando teremos  $\tau$  como 0,0985. Dessa maneira o modelo que representa a esteira é:

$$G(s) = \frac{K}{\tau s + 1}$$

$$G(s) = \frac{2,04}{0,0985s + 1}$$

(Eq. 5.2.1)

### 5.3. Projeto do Controlador PI

Foi escolhido um projeto por emulação devido à concepção do controlador em tempo discreto, considerando a futura implementação. Embora essa implementação

em tempo discreto não tenha se concretizado, a abordagem ainda se mostrou válida. Além disso, um controlador PI foi considerado uma boa opção para o sistema, já que o sistema é de primeira ordem sem zeros, permitindo a eliminação do erro e o uso de uma estrutura de controle relativamente simples.

Para converter o sistema do tempo contínuo para o domínio discreto, utilizou-se a função `c2d` do MATLAB, conforme mostrado no código anexo. O tempo de amostragem adotado foi equivalente à constante de tempo do sistema, resultando na seguinte equação de malha aberta para o sistema no domínio discreto:

$$G(z) = \frac{1,29}{z-0,3679}$$

(Eq. 5.3.1)

O gráfico a seguir mostra a resposta de ambos os sistemas obtidos (em tempo contínuo e discreto) a um degrau unitário:

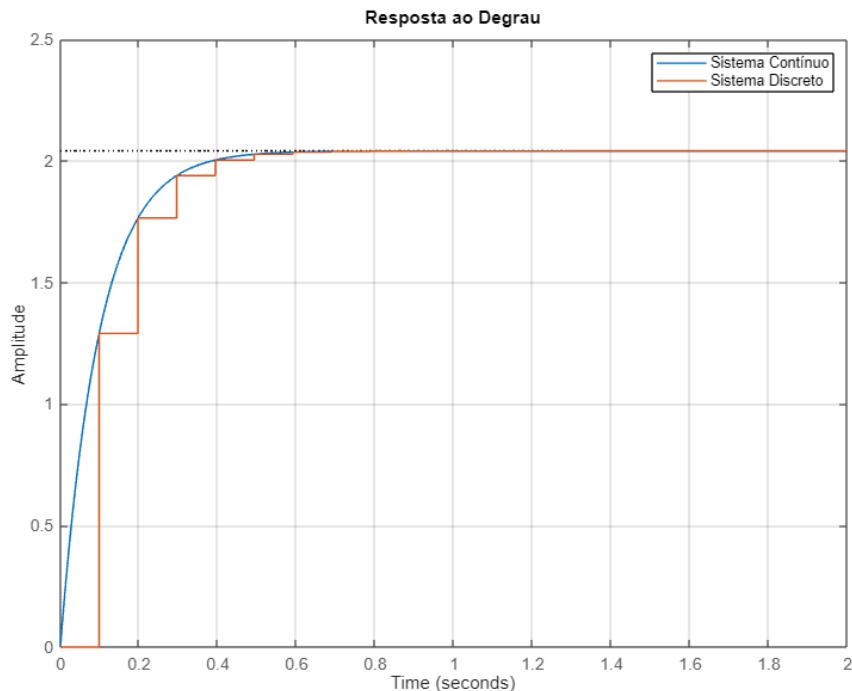
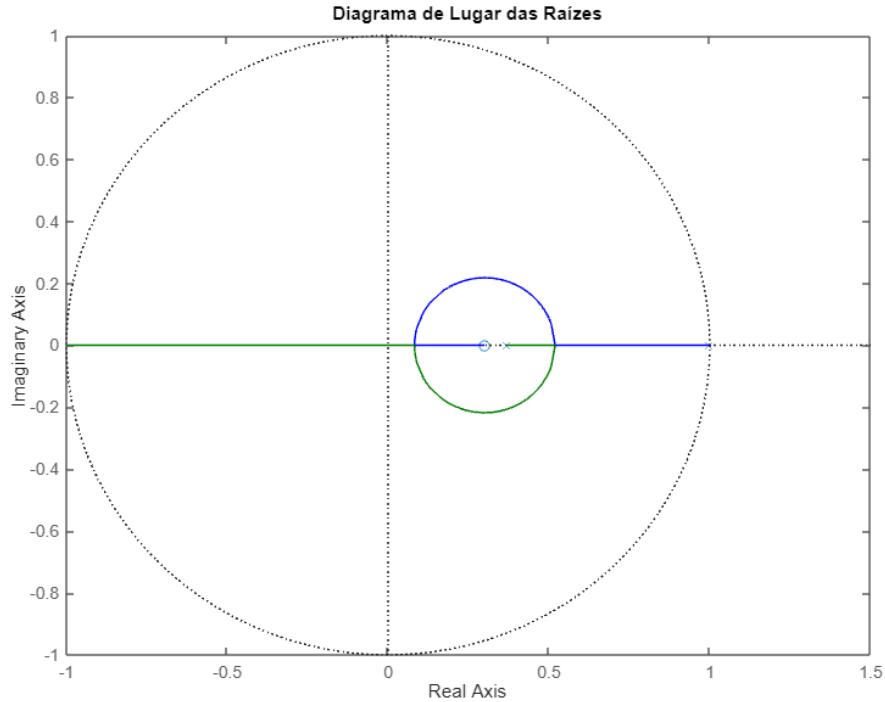


Figura 5.3.1 - Resposta ao Degrau de  $G(s)$  e  $G(z)$  (Autoria Própria)

Com a função de transferência, é possível utilizar o Lugar das Raízes para projetar o controlador PI. Dado que o sistema possui um pólo em 0,3679, o zero foi fixado em 0,3 com a intenção de obter um Lugar das Raízes com o formato desejado.



*Figura 5.3.2 - Root Locus de  $G(z)$  (Autoria Própria)*

É importante salientar que a opção foi por não cancelar pólos e zeros, devido às possíveis imprecisões na modelagem. O ganho do controlador PI foi definido em 0,25, pouco antes do ponto de ramificação mostrado na figura. Considerando que o sistema controla a velocidade de um motor, evitar o sobressinal é fundamental, pois isso resultaria em uma mudança no sentido da aceleração da esteira, o que não é desejável. Com isso, o controlador PI ficou configurado da seguinte forma:

$$C(z) = \frac{0,25(z-0,3)}{(z-1)}$$

(Eq. 5.3.2)

Neste estágio do projeto, foi identificada uma biblioteca no Arduino que permitiu a implementação direta do controlador em tempo contínuo, apesar de o controlador já ter sido desenvolvido em tempo discreto. Para utilizar essa biblioteca, foi aplicada a aproximação de Tustin, empregando o mesmo tempo de amostragem anterior, a fim de retornar ao tempo contínuo.

$$C(S) = \frac{(0,1625S+1,776)}{S}$$

(Eq. 5.3.3)

Como a forma que o programa implementa o PI é da forma  $Kp + Ki/s$ , então o valor de  $Kp = 0,1625$  e o de  $Ki = 1,776$  - valores implementados (sem a necessidade de um derivativo).

É importante lembrar que a visualização do funcionamento do modelo poderá ser realizada através dos vídeos disponibilizados nas Referências Bibliográficas deste relatório, como a mudança de referência, por exemplo.

#### 5.4. Análise de Perturbações

Foi montado o diagrama do sistema no simulink para essas simulações tendo por base o modelo do sistema e do controlador. Optou-se por simular a perturbação por um degrau, pois o sistema real sofre trepidações constantemente devido ao funcionamento do próprio motor. A figura abaixo mostra como ficou no simulink:

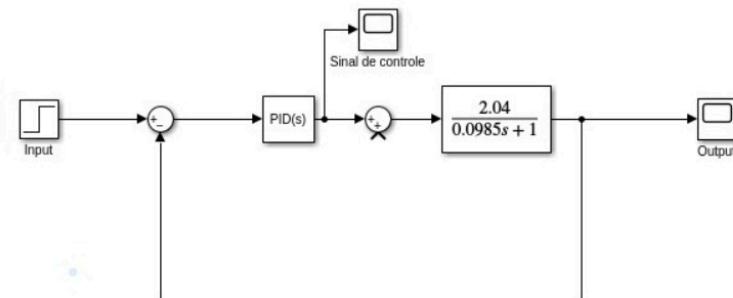


Figura 5.4.1 - Planta do Sistema de Controle sem Perturbação (Autoria Própria)

No primeiro momento se simulou a referência de velocidade de 150 RPM. O sinal de controle e saída para o sistema com essa referência foi como seguem:

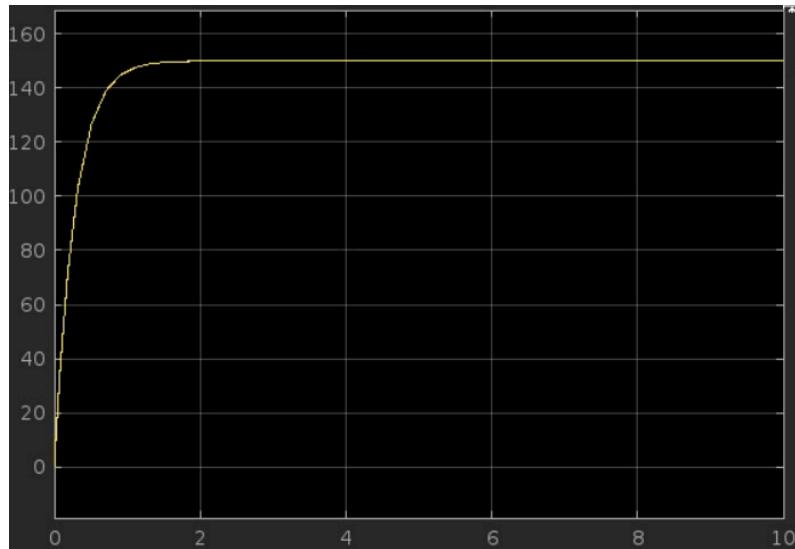


Figura 5.4.2 - Saída do Sistema Sem Perturbação a uma Entrada Degrau 150RPM (Autoria Própria)

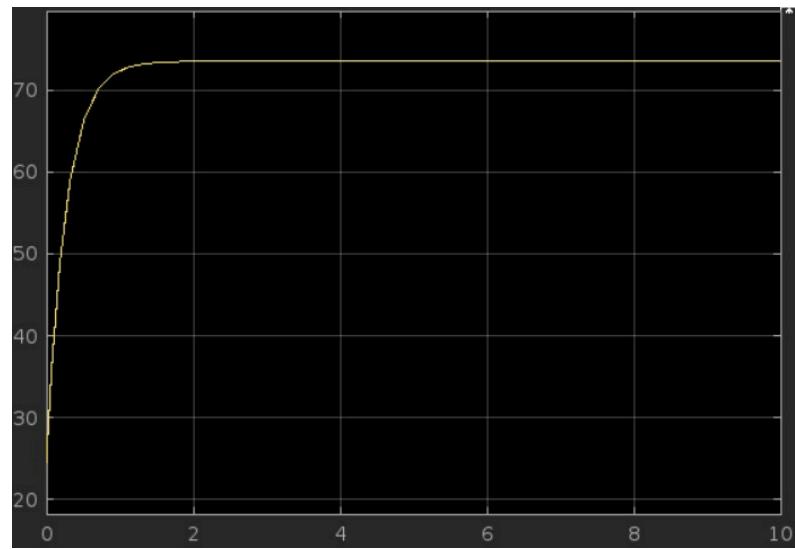


Figura 5.4.3 - Sinal de Controle do Sistema Sem Perturbação a uma Entrada Degrau 150RPM (Autoria Própria)

Em seguida, se aplicou uma perturbação de 10 % do valor de referência 5 segundos após o começo do sistema. Esse tempo foi escolhido para que se pudesse ver com clareza os seus efeitos. O sinal de controle e sinal de saída estão presentes abaixo. Faz-se notável o fato do sistema ser capaz de rejeitar esse tipo de perturbação.

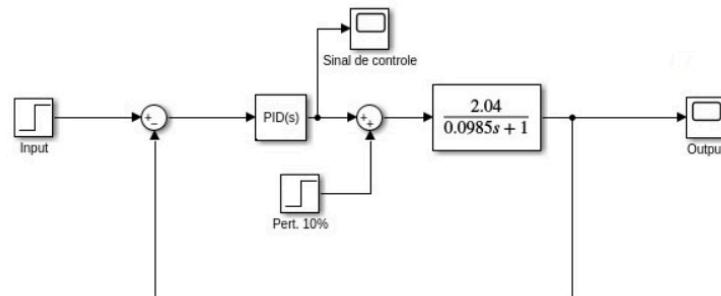


Figura 5.4.4 - Planta do Sistema com Perturbação de 10% (Autoria Própria)

Ademais, mudou-se a referência para um velocidade de referência de 200 RPM. A perturbação foi modelada da mesma forma, inserida com a mesma magnitude de 10% do valor de referência e no mesmo intervalo de 5 segundos após o início do sistema . As imagens abaixo mostram a saída e o sinal de controle para essa configuração. Mais uma vez o sistema foi capaz de rejeitar a perturbação.

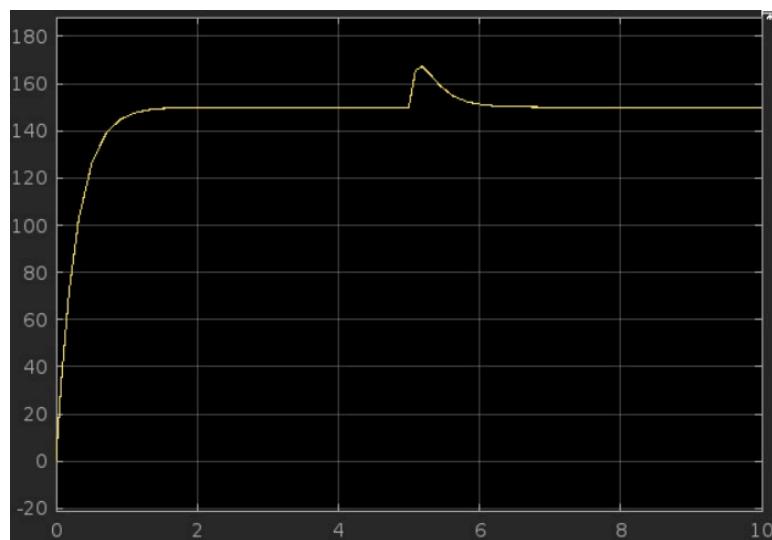


Figura 5.4.5 - Saída do Sistema Perturbação de 10% a uma Entrada Degrau 150RPM (Autoria Própria)

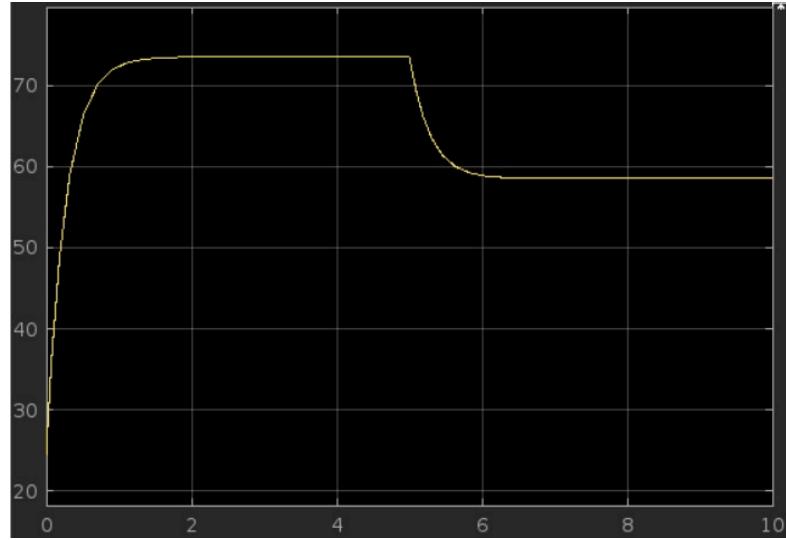


Figura 5.4.6 - Sinal de Controle do Sistema com Perturbação 10% a uma Entrada Degrau 150RPM  
(Autoria Própria)

É possível também realizar o mesmo procedimento para uma nova entrada de referência, como 200RPM por exemplo.

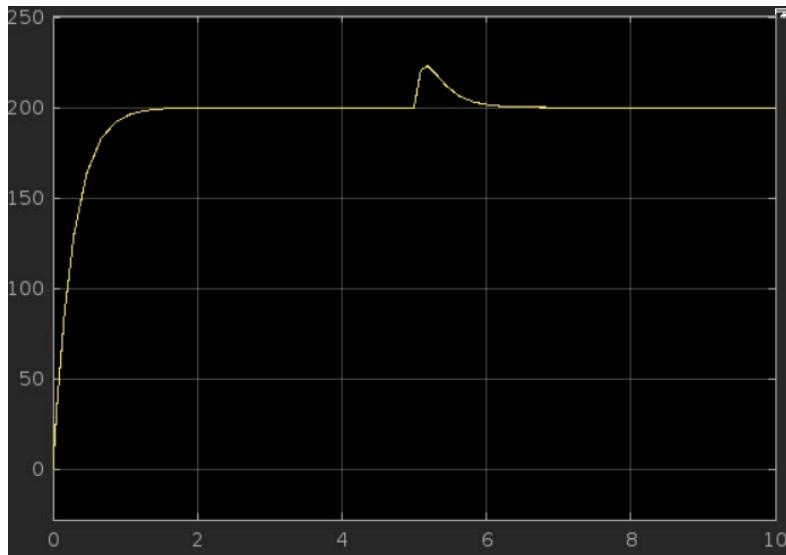


Figura 5.4.7 - Saída do Sistema Perturbação de 10% a uma Entrada Degrau 200RPM (Autoria Própria)

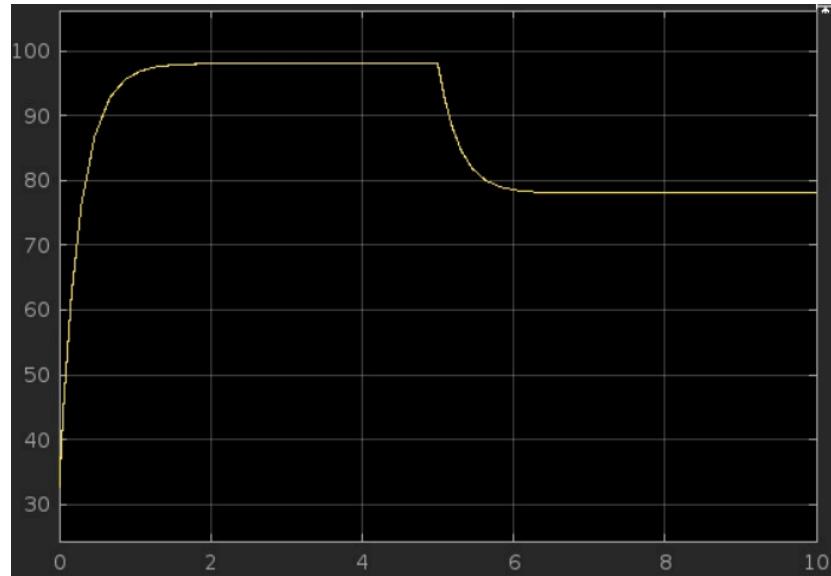


Figura 5.4.8 - Sinal de Controle do Sistema com Perturbação 10% a uma Entrada Degrau 200RPM  
(Autoria Própria)

Ou seja, com o controle PI dimensionado nos itens anteriores, o sistema apresenta um comportamento dinâmico coerente com o que seria esperado.

## 6. CONEXÕES INTERNAS DA ESTEIRA

Este capítulo tem como propósito apenas fazer um levantamento geral das conexões realizadas no projeto através dos jumpers e conexões diretas.

- **Motor CC:** ambos os terminais do motor estarão conectados às saídas +Vcc e GND da lateral do Shield L293D Driver Ponte H;
- **Sensor de Velocidade - Módulo Encoder:** os pinos de +Vcc e GND do Sensor Encoder serão conectados aos pinos 5V e GND do Arduino Mega. Já o pino D0 do sensor Encoder será conectado ao pino digital 18 do Arduino Mega. O pino A0 (análogo) do sensor Encoder ficará sem conexão;
- **Shield L293D Driver Ponte H:** o Shield será encaixado de forma direta em cima do Arduino Mega conforme o encaixe equivalente do Arduino Uno. Não haverá jumpers conectando o Shield ao Arduino. Na parte de alimentação do Shield, o pino de +Vcc será conectado a um dos pinos do Switch ON-OFF enquanto que o pino GND será conectado ao pino de negativo (-) da entrada P2 da esteira;
- **Potenciômetro:** o pino central do Potenciômetro será conectado ao pino analógico A10 do Arduino Mega, tendo os pinos de suas extremidades conectados aos pinos de 5V e GND;
- **Chave (Switch) ON-OFF:** um dos pinos será conectado à alimentação principal do Shield L293D Driver Ponte H enquanto o outro pino será conectado ao pino positivo (+) na entrada P2 da Esteira;
- **Chave (Switch) Inversora de Sentido:** um dos pinos será conectado ao pino de 5V enquanto que o outro será conectado ao pino digital 30 do Arduino Mega. Por apresentar apenas dois pinos, nesse switch terá que ser utilizado um resistor de Pull-Down para garantir que haja GND no pino 30 quanto o switch estiver sem acionamento;
- **Display LCD via I2C:** por apresentar uma conexão via protocolo I2C, a partir do LCD sairão 4 cabos para conectar nos pinos 5V, GND, SCL e SDA do Arduino Mega (conforme indicações escritas nos hardwares).

## 7. IMPLEMENTAÇÃO E RESULTADOS

Como todo o processo de desenvolvimento do projeto foi descrito nos capítulos anteriores, partindo tanto do levantamento técnico dos materiais até a modelagem e projeto do sistema com suas respectivas conexões, é possível então analisar os resultados finais obtidos.

### 7.1. Modelagem e Impressão do Projeto

A parte física de estrutura do projeto foi toda desenvolvida através de modelagem 3D na plataforma *ON Shape* com o auxílio do aluno Ernesto Fernandes<sup>12</sup> (que atualmente trabalha nessa área).

A ideia então é criar um projeto que deixe ainda mais visual a interação do usuário com a esteira, já que o objetivo do projeto não é somente ser funcional, mas ser útil como um modelo didático de estudo para outras disciplinas e outros alunos (como também ficar na exposição da Onda Elétrica).

Do ponto de vista técnico, foi criado o projeto modelado da forma:

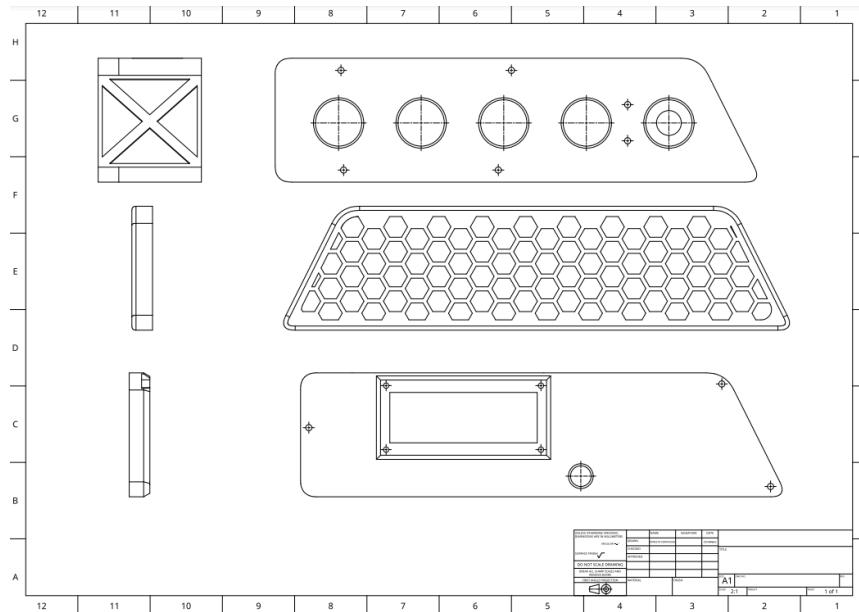
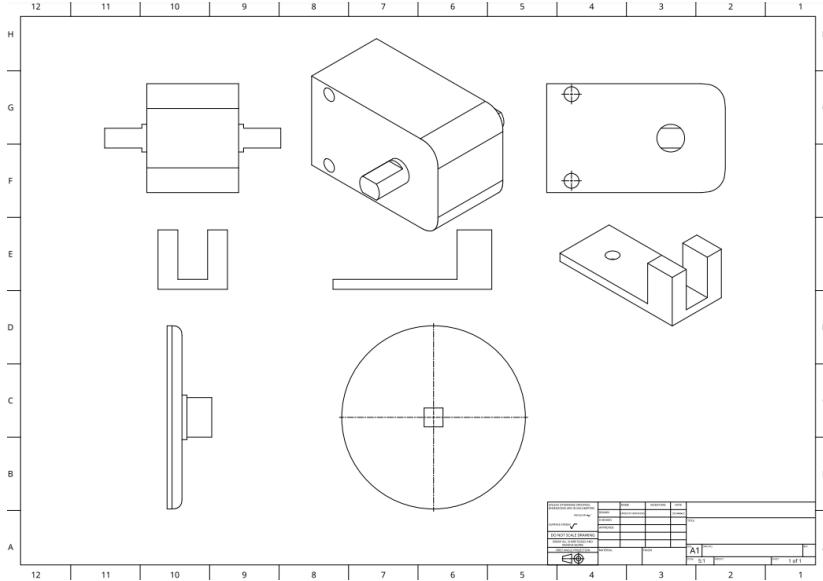


Figura 7.1.1 - Planta da Estrutura Física da Impressora (Autoria Própria)

<sup>12</sup> Ernesto Fernandes Viana é aluno do curso de Engenharia Elétrica da Escola Politécnica da Universidade Federal da Bahia (UFBA), realizando o oitavo período do curso.



*Figura 7.1.2 - Planta do Motor e Encoder para Impressão (Autoria Própria)*

Um ponto que pode gerar dúvida é quanto ao motivo de não haverem cotas nessa planta, mas isso é respondido com o fato de que o software *ON Shape* faz a modelagem diretamente na tela em 3D, como ocorre no *DIALUX* (modelagem de projetos Luminotécnicos) ou no *SOLIDWORKS* (modelagem de projetos principalmente mecânicos).

Na figura a seguir é possível entender internamente melhor como ocorrem as conexões. Em laranja, na parte da direita é onde serão fixados o Arduino Mega e o Shield L293d Driver Ponte H, tendo duas saídas para cabos de dados e de potência para alimentar o Microcontrolador.

Já no lado da esquerda, é possível ver em cinza o Sensor Encoder posicionado na frente da fixação do Motor CC. Ao fundo, em cinza também, são fixadas as estruturas para os mancais de rolamento (de skate) que serão usados para fazer a parte móvel da esteira.

Na região superior é possível também ver os furos onde serão alocados o potenciômetro e os dois switchs (um para controle de inversão de velocidade e o outro para ligar e desligar a esteira).

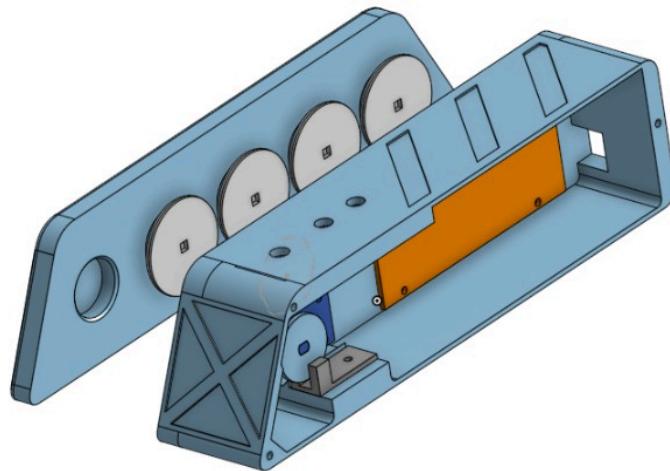


Figura 7.1.3 - Modelagem 3D da Esteira - Vista Frontal (Autoria Própria)

Na próxima figura também é possível de observar uma vista posterior da esteira, já com todo o acabamento concluído e posicionamento dos cilindros de rolamento que irão ser circundados pela esteira (que utilizará material EVA, um polímero emborrachado, flexível, com características adesivas e componentes que são à prova d'água).

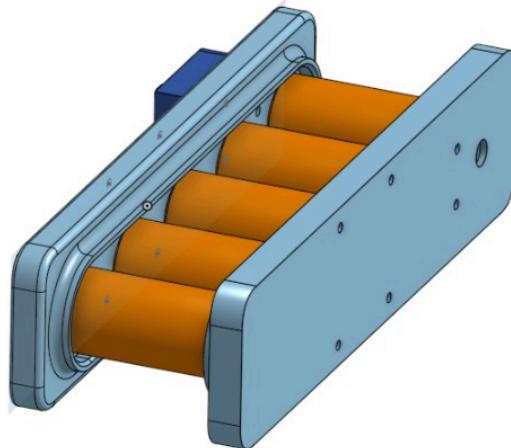
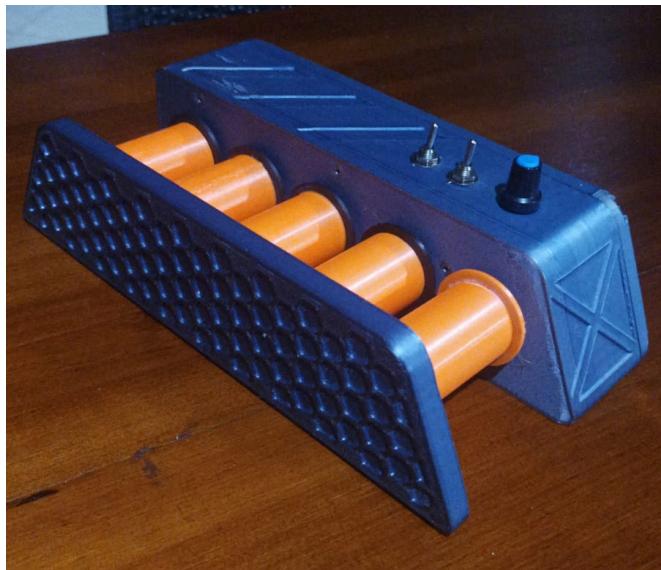


Figura 7.1.4 - Modelagem 3D da Esteira - Vista Posterior (Autoria Própria)

Com a modelagem finalizada, é possível então realizar a impressão da esteira em PLA (Poliácido Láctico) que é um polímero termoplástico biodegradável altamente usado pela indústria de impressão de objetos em 3D. Eles são produzidos a partir de recursos renováveis biológicos.

É importante lembrar que, para o projeto em questão, foi ao máximo tentado utilizar materiais de fácil acesso e que fossem não poluentes, reciclando muita coisa que poderia ser de fácil acesso para qualquer pessoa (salvo a disponibilidade de uma impressora 3D que, hoje, pode ser acessada na UFBA tanto no Laboratório de Ciência e Tecnologia no Campus Ondina quanto no PET Elétrica<sup>13</sup>).

A impressão e montagem final do projeto pode ser visualizada a seguir:



*Figura 7.1.5 - Projeto final da Esteira Impresso (Autoria Própria)*

---

<sup>13</sup> PET Elétrica é um Programa de Educação Tutorial da Engenharia Elétrica (ligado ao Ministério da Educação - MEC) que desenvolve experimentos e estudos, na área da Engenharia Elétrica, a fim de produzir novos conhecimentos.



Figura 7.1.6 - Projeto final da Esteira Completa (Autoria Própria)

A ideia então é criar um projeto que deixe ainda mais visual a interação do usuário com a esteira, já que o objetivo do projeto não é somente ser funcional, mas ser útil como um modelo didático de estudo para outras disciplinas e outros alunos (como também ficar na exposição do Onda Elétrica).

## 7.2. Resultados Obtidos

Como o funcionamento da esteira só faz real sentido quando visualizado seu funcionamento, o presente relatório optou por deixar os primeiros links das referências bibliográficas deste relatório somente para visualização do funcionamento da esteira (tanto na etapa parcial do projeto quanto na etapa final do projeto).

Em suma, o projeto apresentou resultados satisfatórios quando ajustados os valores de  $K_i$  e  $K_p$  conforme capítulo 5 deste relatório. Entretanto, por causa de trepidações causadas pelo método de montagem, a realimentação com o valor medido no encoder acaba demorando um pouco para ajustar o valor da rotação da esteira ao setpoint colocado no potenciômetro - o que pode acabar levando cerca de 5 segundos para a esteira chegar ao resultado esperado.

Especificações da Esteira com Controle de Velocidade	
Tensão de Operação	6 - 9V
Corrente de Operação	300mA
Tamanho (CxLxA)	22x13x6cm
Peso	0,65kg
Material da Estrutura	PLA
Material da Esteira	EVA
Controlador	Arduino Mega
Atuador	Shield L293d Driver Ponte H
Sensor	Sensor Encoder LM393
Motor	Motor CC com Caixa de Redução
Visor	Display LCD 16x2 com I2C Soldado
Rolamento	Rolamentos Abec de Skate
PWM Aplicado	0 - 255
Rotação Verdadeira em 255	200RPM
Rotação Medida em 255	600RPM
Tempo para Regime Permanente	5s
Possibilidade de Inversão de Sentido	Sim

*Tabela 7.2.1 - Dados Técnicos da Esteira com Controle de Velocidade (Autoria Própria)*

### 7.3. Dificuldades e Possibilidades de Melhoria

Neste último item técnico sobre a concepção, modelagem, projeto e implementação do projeto do Controle de Velocidade de uma Esteira utilizando um Controlador PI (já que o Kd é nulo), o presente relatório irá debruçar-se sobre explorar as maiores dificuldades que foram encontradas durante o desenvolvimento do projeto.

De início, pela análise técnica dos materiais utilizados, é possível perceber uma incompatibilidade entre a Fonte de 5V que foi utilizada, o Shield L293d e o Motor CC, uma vez que, por medições realizadas durante o funcionamento, percebeu-se que o Driver Ponte H apresenta uma queda de tensão em torno de 1,6V, restando apenas 3V para o Motor CC, o que não permite o motor funcionar em sua máxima potência.

Esse comportamento pode ser melhor analisado nas imagens de medição realizadas apresentadas a seguir:

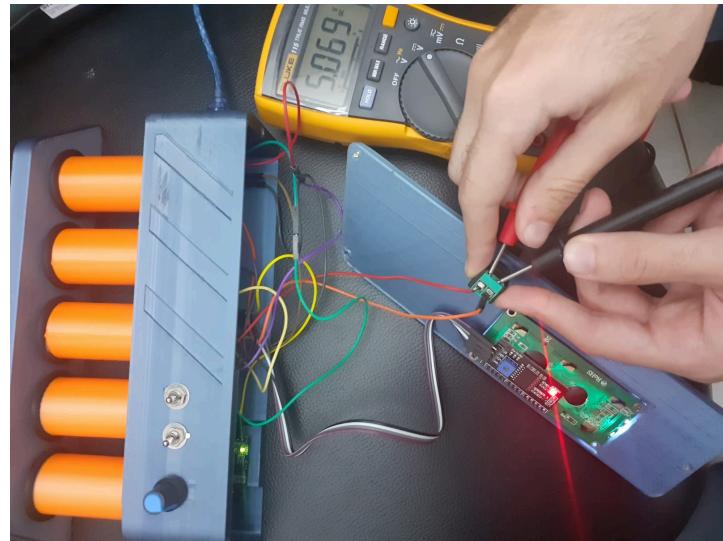


Figura 7.3.1 - Medição da Tensão de Entrada da Esteira para Fonte de 5V (Autoria Própria)

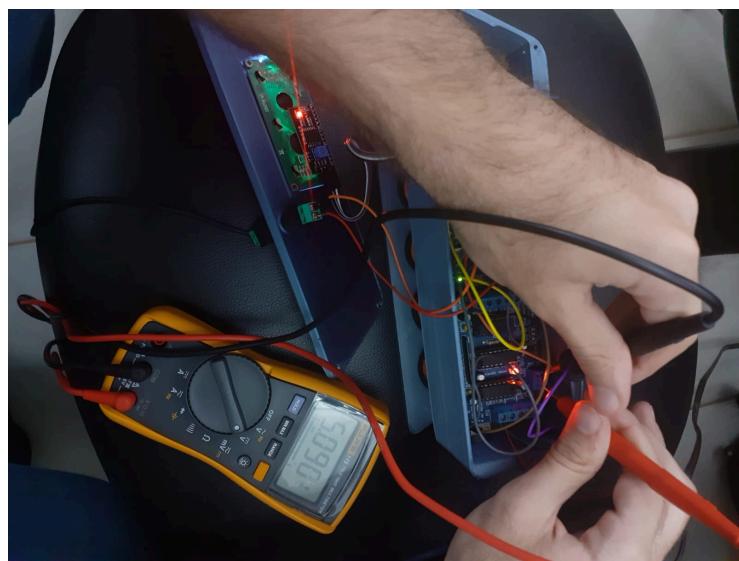


Figura 7.3.2 - Medição da Tensão de Entrada do Shield para Fonte de 5V (Autoria Própria)



Figura 7.3.3 - Medição da Tensão na ponte do Shield para Fonte de 5V (Autoria Própria)

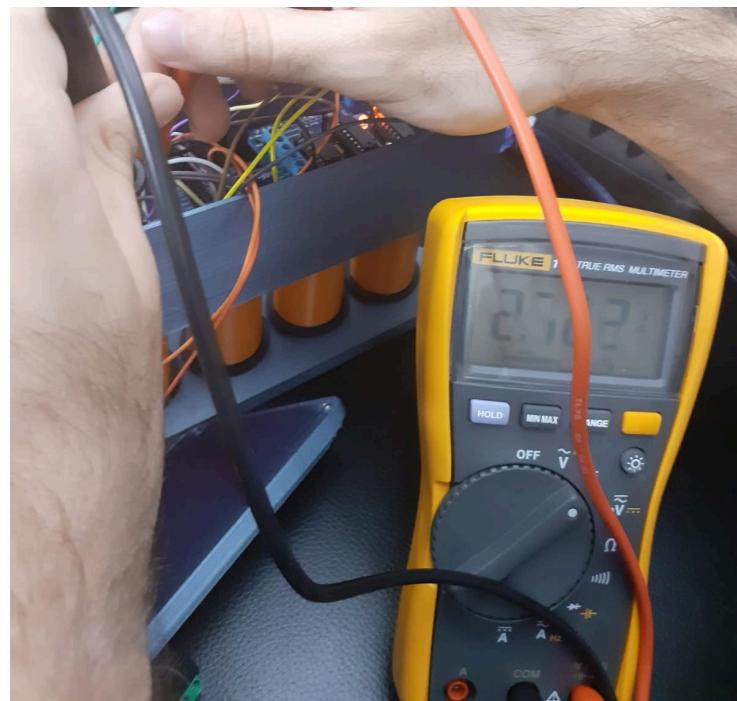


Figura 7.3.4 - Medição da Tensão nos terminais do Motor para Fonte de 5V (Autoria Própria)

Um segundo problema muito complicado, inclusive, de ser contornado é quanto a quantidade de cabos que estão presentes no interior da esteira. O ideal seria procurar uma forma de soldar os cabos diretamente a parte metálica dos pinos das placas utilizadas sem o uso de jumpers conectores. A figura a seguir apresenta essa dificuldade que, inclusive, pode atrapalhar o aperto dos parafusos utilizados no projeto.

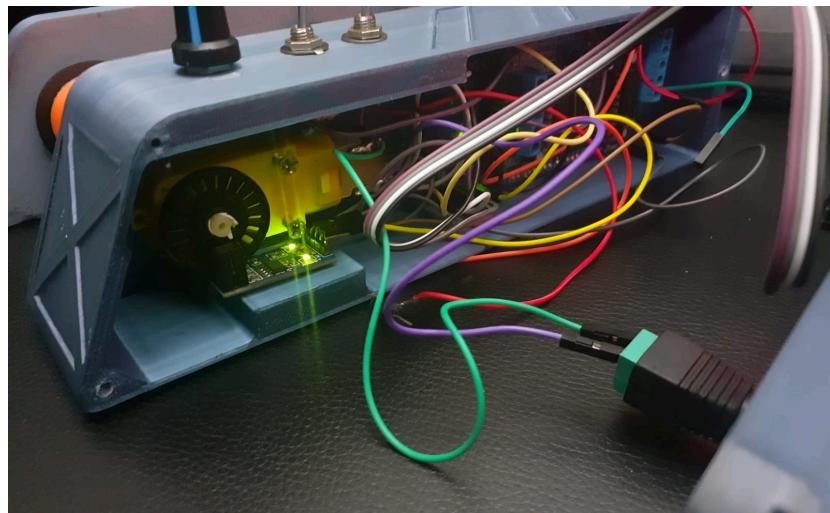


Figura 7.3.5 - Demonstração do acúmulo de jumpers no interior da Esteira (Autoria Própria)

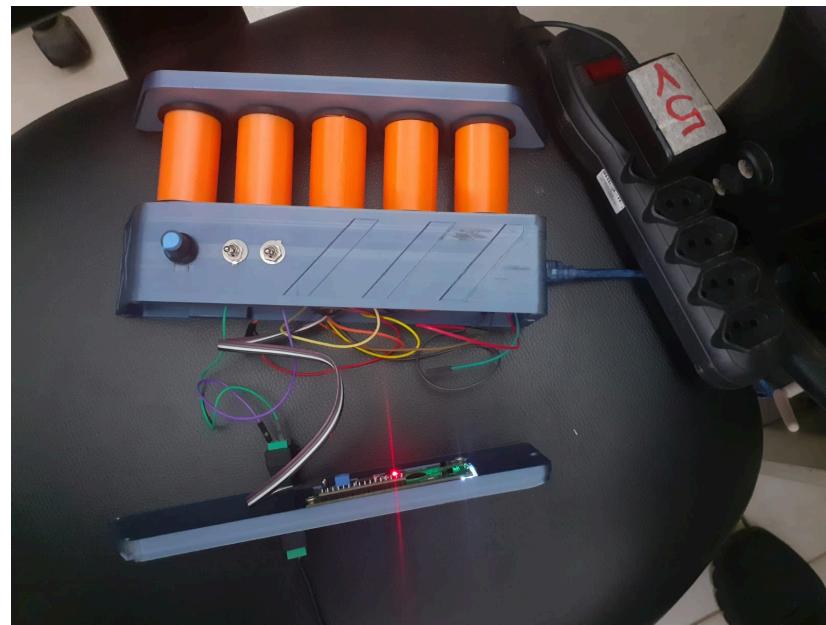


Figura 7.3.6 - Vista superior da esteira aberta (Autoria Própria)

Um problema muito complicado também percebido foi no início do desenvolvimento do projeto sob a forma da impressão da peça girante da esteira que está acoplada coaxialmente ao motor. Por causa do formato de impressão (ou pela precisão da impressora 3D utilizada), quando o motor partia com o máximo de controle PWM possível, o torque era tão alto que rompia a peça de conexão com o cilindro.

O problema foi contornado modificando a impressora que realizou a impressão, já que existem modelos que são muito rápidos, mas apresentam baixa precisão; e modelos que são muito precisos, mas apresentam baixa velocidade de impressão. É uma relação de compromisso que só pode ser decidida quando se analisa qual a prioridade que o projeto precisa ter.

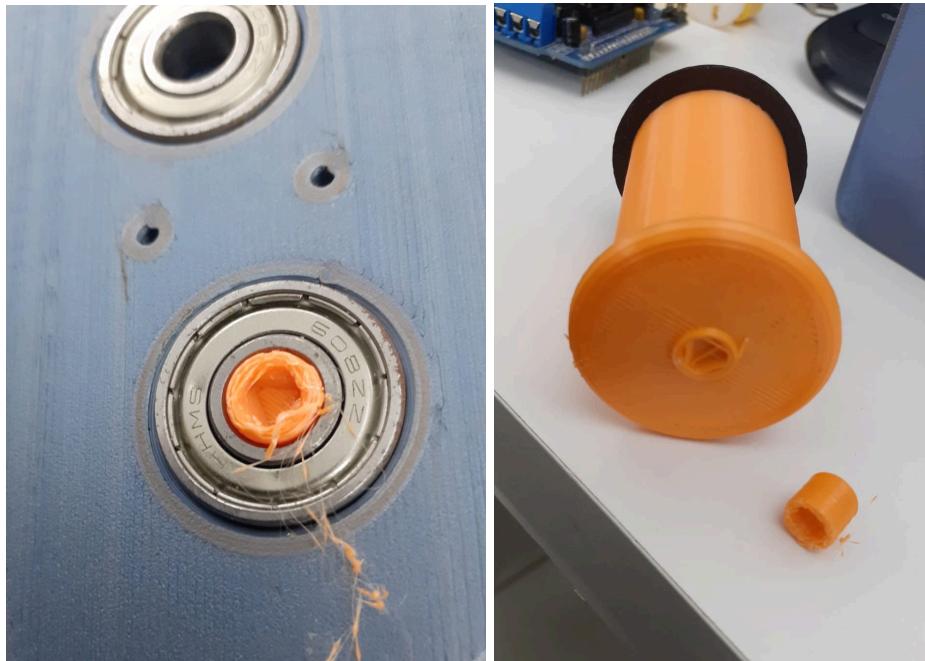


Figura 7.3.7 - Peça Rompida do Cilindro de Rotação (Autoria Própria)

## 8. CONCLUSÃO

A conclusão deste relatório reforça a importância da integração entre hardware e software no desenvolvimento de sistemas de controle em ambientes automatizados. Ao implementar um sistema de controle de velocidade para uma esteira utilizando o controlador PID em uma plataforma Arduino, foi possível demonstrar na prática a aplicação de conceitos teóricos em um cenário real. O projeto não apenas validou a eficácia do PID na regulação da velocidade de um motor de corrente contínua, mas também destacou a versatilidade e a acessibilidade do Arduino Mega como microcontrolador para projetos de controle.

A construção da estrutura física com impressão 3D, a escolha criteriosa dos componentes, como o Shield L293D e o encoder, e a implementação cuidadosa do algoritmo de controle, resultaram em um sistema robusto e eficiente. Os testes experimentais realizados comprovaram a capacidade do controlador em manter a velocidade da esteira estável, mesmo diante de variações de carga, confirmando a adequação do modelo teórico à prática.

Este trabalho contribuiu para uma compreensão mais aprofundada dos desafios e soluções na implementação de sistemas de controle, evidenciando a importância do ajuste fino dos parâmetros do PID e da escolha dos componentes adequados para garantir a performance desejada. Além disso, o uso de técnicas como a modelagem teórica do motor CC permitiu uma análise mais detalhada do comportamento do sistema, essencial para a otimização do controle.

Por fim, o projeto demonstrou que, com o uso de ferramentas e tecnologias acessíveis, é possível desenvolver sistemas de controle eficientes e aplicáveis a uma vasta gama de situações, promovendo a aprendizagem e a inovação no campo da automação.

## 9. REFERÊNCIAS BIBLIOGRÁFICAS

**Correia, G. Fiscina, M.** Vídeo Demonstrativo do Experimento Parcial realizado.  
<https://drive.google.com/file/d/15fsb7qfMwpVVwrnJq4V1zZ82ygcJWvta/view?usp=sharing>.

**Correia, G. Fiscina, M.** Vídeo Demonstrativo do Experimento Final realizado.  
[https://drive.google.com/file/d/1CE-3VeYUvCCT9EBNY9pARu3fcmC31Are/view?usp=drive\\_link](https://drive.google.com/file/d/1CE-3VeYUvCCT9EBNY9pARu3fcmC31Are/view?usp=drive_link).

**COELHO.** Antonio A. R., **ALMEIDA.** Otacílio M., **SANTOS.** José E. S., **SUMAR,** Rodrigo R. **Laboratório de Simulação no Ensino de Sinais e Sistemas Lineares.** Universidade Federal de Santa Catarina (UFSC). Departamento de Automação e Sistemas. 2001. Florianópolis - SC.

**FITZGERALD, A. E.; KINGSLEY, C.; UMANS, S. D.** *Máquinas Elétricas*. Revisado por Stephen D. Umans. 7. ed. Porto Alegre: AMGH, 2014.

**VARRIALE.** Eduardo da Silva. **Automação de uma Esteira Ergométrica para Aplicação em Realidade Virtual.** Universidade Federal do Rio Grande do Sul (UFRGS). Trabalho de Conclusão de Curso em Engenharia de Controle e Automação. Dezembro de 2017 - Porto Alegre.

Canal **FunBots**. Controle de Motor com Encoder | Controle PID e Sensor Infravermelho Disponível no link: <https://www.youtube.com/watch?v=tr83BTwlujY> - acesso em 01 de abril de 2024.

Canal **Irfan's Idiotic Ideas**. How To Make A Conveyor Belt System At Home || Conveyor Belt Model || Homemade Conveyor Belt. Disponível no link: <https://www.youtube.com/watch?v=8Vnoso8rhaw> - acesso em 03 de abril de 2024.

Canal **Avila's Technology**. Cómo Hacer Una Banda Transportadora Con Sensores Infrarrojos y Un Limit Switch | Avila's Technology. Disponível no link: <https://www.youtube.com/watch?v=eww3crwOPv8> - acesso em 03 de abril de 2024.

Canal **WR. Kits**. MEDINDO RPM DE MOTOR DC COM ARDUINO. Disponível no link: <https://www.youtube.com/watch?v=JKosbahBcMQ> - acesso em 03 de abril de 2024.

**MCROBERTS**, Michael. **Arduino básico**. Novatec Editora, 2011.

**NISE**, Norman S. **Engenharia de sistemas de controle**. Tradução e revisão técnica Jackson Paul Matsuura. 6<sup>a</sup> edição. Rio de Janeiro: LTC, 2013

## 10. ANEXO 1 - CÓDIGO DO MATLAB

```
% Universidade Federal da Bahia
% Aluno: Gabriel Correia e Mateus Fiscina
% ENGC54 - Laboratório Integrado VI
% AT4 - Projeto Controle de Esteira
% Configurações Iniciais

clc
clear()
close('all')

% Carregar os dados
tabela = readtable('dados.tsv', 'FileType', 'text', 'Delimiter', '\t');
tempo = tabela.Var2;
velocidade = tabela.Var3;

% Valor RPM aplicado
valor_rpm_aplicado = 255;

% Calcular a média da parte ruidosa para obter o valor de regime permanente
velocidade_regime = mean(velocidade(3:51));

% Criar o gráfico
figure(1);
plot(tempo, velocidade, '-b', 'DisplayName', 'Velocidade'); % Gráfico de Tempo vs Velocidade
xlabel('Tempo [s]');
ylabel('Velocidade [RPM]');
title('Ensaio em Malha Aberta');

% Adicionar linha horizontal para o valor de referência
yline(velocidade_regime, 'r--', 'Valor de Referência',
'LabelHorizontalAlignment', 'left');

% Calcular K e a constante de tempo
K = velocidade_regime / valor_rpm_aplicado;
velocidade_constantedetempo = 0.63 * velocidade_regime;

% Interpolação para achar a constante de tempo
Tau = 0.1 + (velocidade_constantedetempo - 330) * (0.2 - 0.1) / (480 - 330); %
Tau agora é o Ts

% Criar a função de transferência e a resposta ao degrau em tempo contínuo
figure(2);
malha_continuo = tf(K, [Tau 1]); % Nota: K e Ts são usados diretamente na função de transferência continua
step(malha_continuo);
```

```
hold on;

% Converter o sistema para tempo discreto com Ts como período de amostragem
malha_discreto = c2d(malha_continuo, Tau);
% Plotar a resposta ao degrau do sistema discreto
step(malha_discreto);
legend('Sistema Contínuo', 'Sistema Discreto');
title('Resposta ao Degrau');
grid on;
% Definir o controlador discreto com o mesmo período de amostragem
% Note: Ajuste os coeficientes conforme necessário
controlador = tf([1 -0.3], [1 -1], Tau);
% Criar o sistema em malha fechada para análise do lugar das raízes
sistema_controlado = malha_discreto * controlador;
% Gerar o diagrama de lugar das raízes
figure(3);
rlocus(sistema_controlado);
title('Diagrama de Lugar das Raízes');
%Definindo como ganho K perto do ponto de ramificação
ganho_discreto=0.25;
PI_discreto=controlador*ganho_discreto;
%Jogando de volta para o tempo contínuo
PI_continuo=d2c(PI_discreto,'tustin');
%Dessa forma teremos com base no PI_continuo
Kp=0.1625;
Ki=1.776;
```

## 11. ANEXO 2 - CÓDIGO DO ARDUINO

```

/*
Universidade Federal da Bahia (UFBA)
Departamento de Engenharia Elétrica e da Computação (DEEC) – Escola Politécnica
Alunos: Gabriel Correia e Mateus Fiscina
Disciplina: Laboratório Integrado VI
Professora: Cristiane Paim

MODELO DE CONTROLE DE VELOCIDADE DE ESTEIRA UTILIZANDO PID
*/

// ----- DEFINIÇÃO E CONFIGURAÇÕES INICIAIS -----
// Importação das Bibliotecas do Código (importante baixar antes de executar)
#include <AFMotor.h> // Biblioteca para controlar motores DC usando o Adafruit
Motor Shield
#include <PID_v1.h> // Biblioteca para controle PID
#include <Wire.h> // Biblioteca para comunicação I2C
#include <LiquidCrystal_I2C.h> // Biblioteca para controlar LCD I2C

// Configuração do Motor
AF_DCMotor motor(1); // Cria um objeto motor na porta M1 do Adafruit Motor
Shield

// Constantes do Controle PID
#define MIN_PWM 0 // Valor mínimo de PWM
#define MAX_PWM 255 // Valor máximo de PWM
#define KP 0.1625 // Constante proporcional do PID
#define KI 1.776 // Constante integral do PID
#define KD 0 // Constante derivativa do PID

// Variáveis do Sensor Infravermelho e PID
double rpm; // Armazena o valor atual de RPM do motor
volatile byte pulsos; // Contador de pulsos do sensor infravermelho
unsigned long timeold; // Marca o tempo da última atualização de RPM
int pinoSensor = 18; // Pino do Arduino ligado ao pino D0 do sensor
unsigned int pulsosDisco = 20; // Número de pulsos por rotação completa do
disco encoder
double velocidade = 0; // Armazena o valor de saída do PID (PWM)
double velocidadeSetpoint = 100; // Velocidade desejada (setpoint) em RPM

// Constante para o pino do potenciômetro
const int potPin = A10; // Pino analógico onde o potenciômetro está conectado
const int switchPin = 30; // Pino digital onde o switch está conectado

// Substitua 0x27 pelo endereço do seu LCD encontrado pelo scanner I2C
LiquidCrystal_I2C lcd(0x27, 16, 2); // Inicializa o display LCD I2C com
endereço 0x27, 16 colunas e 2 linhas

// Cria PID para controle

```

```

PID motorPID(&rpm, &velocidade, &velocidadeSetpoint, KP, KI, KD, DIRECT);

// ----- PROGRAMA A SER RODADO -----
// Função executada a cada interrupção
void contador() {
    pulsos++; // Incrementa o contador de pulsos
}

// SETUP
void setup() {
    // Inicia Serial
    Serial.begin(9600); // Inicia a comunicação serial com velocidade de 9600 bps

    // Inicia o LCD I2C
    lcd.init(); // Inicializa o display LCD
    // lcd.begin(16, 2);
    lcd.backlight(); // Liga a luz de fundo do LCD

    // Configura Interrupção
    pinMode(pinoSensor, INPUT); // Define o pino do sensor como entrada
    pinMode(switchPin, INPUT); // Configura o pino do switch como entrada com
    pull-up interno

    attachInterrupt(digitalPinToInterrupt(pinoSensor), contador, FALLING); //
    Configura interrupção para contar pulsos em borda de descida
    pulsos = 0; // Inicializa o contador de pulsos
    rpm = 0; // Inicializa o valor de RPM
    timeold = 0; // Inicializa o tempo antigo

    // Configura controle PID
    motorPID.SetOutputLimits(MIN_PWM, MAX_PWM); // Define os limites de saída do
    PID
    motorPID.SetMode(AUTOMATIC); // Define o modo do PID para automático
}

// LOOP
void loop() {
    // Lê o valor do potenciômetro e ajusta o setpoint de velocidade
    int potValue = analogRead(potPin); // Lê o valor do potenciômetro
    velocidadeSetpoint = map(potValue, 0, 1023, 0, 500); // Mapeia o valor do
    potenciômetro para a faixa desejada de RPM

    // Lê o estado do switch
    bool switchState = digitalRead(switchPin); // Lê o estado do switch (HIGH se
    pressionado, LOW se não pressionado)

    // Calcula RPM a cada 1 Segundo
}

```

```

if (millis() - timeold >= 1000) { // Se um segundo se passou desde a última
medição
    detachInterrupt(digitalPinToInterrupt(pinoSensor)); // Desabilita a
interrupção durante o cálculo para evitar inconsistências
    rpm = (60 * 1000 / pulsosDisco) / (millis() - timeold) * pulsos; // Calcula
RPM baseado nos pulsos contados
    timeold = millis(); // Atualiza o tempo antigo para o tempo atual
    pulsos = 0; // Reseta o contador de pulsos

    // Exibe TODOS os valores no serial monitor
    Serial.print("Setpoint: ");
    Serial.print(velocidadeSetpoint); // Exibe o setpoint de velocidade atual
    Serial.print(" ");
    Serial.print("Vel - saída PID: ");
    Serial.print(velocidade, 2); // Exibe a velocidade calculada pelo PID com
duas casas decimais
    Serial.print(" ");
    Serial.print("RPM: ");
    Serial.println(rpm, 0); // Exibe o valor de RPM calculado
    Serial.print("Erro: ");
    Serial.println(velocidadeSetpoint - rpm); // Exibe o valor do erro do
SetPoint em relação ao RPM medido pelo sensor

    // Habilita novamente a interrupção
    attachInterrupt(digitalPinToInterrupt(pinoSensor), contador, FALLING);
}

// Calcula o PWM do motor conforme Controle PID
motorPID.Compute(); // Calcula o valor de saída do PID baseado no RPM atual e
no setpoint

// Lê o estado do switch
//int switchState = digitalRead(switchPin);

// Ajusta PWM no motor com base no estado do switch
if (switchState == LOW) {
    motor.run(FORWARD); // Se o switch estiver pressionado, roda o motor para
frente

} else {
    motor.run(BACKWARD); // Caso contrário, roda o motor para trás
}

motor.setSpeed(velocidade); // Define o PWM do motor com o valor calculado
pelo PID

```

```
// Apresenta os parâmetros no visor do LCD I2C
lcd.clear(); // Limpa o display LCD
lcd.setCursor(0, 0); // Move o cursor para a primeira linha
lcd.print("Setpoint: "); // Exibe "Setpoint:" no LCD
lcd.print(velocidadeSetpoint); // Exibe o setpoint de velocidade atual
lcd.setCursor(0, 1); // Move o cursor para a segunda linha
lcd.print("Snsr: "); // Exibe "Sensor:" no LCD
lcd.print(rpm); // Exibe o valor de RPM calculado
lcd.print(" RPM"); // Exibe " RPM" após o valor de RPM
//delay(100); // Aguarda 100 milissegundos antes de repetir o loop
}
```