**Instituto Federal de Santa Catarina - IFSC**

**Engenharia de Controle e Automação**

**Projeto Integrador**

Mesa Birrotativa Para Solda Com Parâmetros Controlados

**Aécio Alves Da Anunciação**

**Guilherme Simioni Tauffer**

**Eduardo Cinésio Costa da Silva**

**Chapecó, 2022**

1. [**Projeto Preliminar**](#_mrbzmunbn2hz) **4**

**1.1** [Projeto elétrico](#_dck7vox8xp4u) 6

[Ajustes Necessários](#_fasyftonw2e3) 6

[Testes](#_2av7ykmmydi7) 6

1. [**Controle e Instrumentação**](#_7l80869kfe89) **7**

**2.1** [Requisitos do cliente](#_haras9ebasw7) 7

**2.2** [Detalhamento funcional](#_dom3niesihmz) 7

**2.3** [Matriz morfológica de controle e instrumentação](#_4ouledmzs15p) 8

**2.4** [Matriz de avaliação de controle instrumentação](#_30dng4ghjw8f) 9

**2.5** [Métodos práticos](#_2wcdmrk8wrs2) 10

**2.5.1** [Definição dos controladores para os eixos](#_5d7ds05bj029) 10

**2.5.2** [Hardware necessário para o sistema de controle e instrumentação](#_aui4mqy7go06) 10

1. [**Métodos e análises**](#_agstfe50ukyk) **12**

**3.1** [Controle de posição em malha aberta](#_3l1uz7hm38n9) 12

**3.1.1** [Coleta de dados](#_ukxtenoj4k7u) 12

**3.1.2** [Definição do controlador](#_f0x9utis5ui8) 12

**3.1.3** [Implementação](#_kz79dsbnfqk0) 13

**3.2** [Controle de Velocidade e posição em cascata](#_trupz5hj0efq) 13

**3.2.1** [Coleta de dados](#_bb0hn2dk36ji) 13

**3.2.2** [Definição do controlador](#_ynkxa6yaaepp) 13

**3.2.3** [Implementação](#_j544s0cb6do6) 14

**3.2.3.1** [Discretização](#_88fua2ksn4f3) 20

1. [**Referências bibliográficas**](#_we3969ps3xnq) **21**

# Introdução

Com a rápida evolução tecnológica, e com a alta competitividade, desenvolver um produto e introduzi-lo no mercado está tornando-se cada vez mais desafiador.

Não basta apenas inovar, é necessário manter a competitividade e criar um produto que se solidifique no mercado, conquistando clientes e apreço.

Segundo Dym, “Um bom projeto não acontece por acaso. Em vez disso, ele resulta do pensamento cuidadoso sobre o que os clientes e usuários querem e sobre como anunciar e alcançar os requisitos do projeto.”.

Deste modo, não basta apenas conhecer o problema, mas também é necessário conhecer as mais variadas soluções possíveis, buscando informações e integrando ao projeto profissionais de diferentes áreas, os quais devem corroborar aplicando o conhecimento para cada área envolvida.

Para que um projeto tenha sucesso é necessário um bom gerenciamento do projeto, assim, obtém-se bons resultados.

Desta forma, este trabalho contém o projeto informacional realizado com base na metodologia de projeto e produto adotado pelo Instituto Federal de Santa Catarina, de uma mesa birrotativa com parâmetros controlados, que terá como cliente final a própria instituição de ensino. O intuito é utilizar da criação da máquina desenvolvida neste projeto para aplicação de soldas de maneira controlada e automatizada.

# Projeto Preliminar

Uma mesa birrotativa de soldagem é uma máquina que facilita a realização de tarefas de solda em peças tubulares ou de geometrias mais complexas. Esta máquina foi criada com o intuito de atender a necessidade das indústrias, aumentando a eficiência de produção.

Atualmente, na indústria 4.0 houve a automatização das linhas produtivas, substituindo a mão de obra humana por robôs e braços robóticos, os quais trabalham em conjunto e de forma muito eficiente com a mesa birrotativa, rotacionado a peça a ser soldada junto ao movimento do robô. Esse tipo de tecnologia é muito implementado nas indústrias metalúrgicas, automotivas, aeroespacial e naval.

Para a realização do projeto da mesa birrotativa, primeiramente, é necessário levantar as definições do problema e as demandas do projeto, como é o que a mesa deve cumprir e suprir no final de sua construção. Tendo como pré requisitos de projeto dois graus de liberdade controlados, ou seja controle de velocidade e posição no **Eixo 1** e controle de posição no **Eixo 2**, e a integração com o robô ABB IRB140. Visando isso, o protótipo mostrado abaixo foi desenvolvido.

#### Figura 1. Representação dos eixos na mesa projetada. (Autoria própria).

Suas dimensões físicas foram definidas para proporcionar a integração com o robô, seus motores foram dimensionados para ter torque e velocidade suficiente para rotacionar tubos ou peças com até 8kg de massa.

O **Eixo 1** é onde será realizada a rotação da peça acoplada para ser soldada, neste eixo é necessário realizar o controle de velocidade e posição e não necessita de um torque tão significante como no outro eixo, dessa forma a utilização de um motor CC é a melhor alternativa, já que é possível variar sua velocidade e sentido de rotação, assim como, implementar um controle em cascata com eventuais perturbações, também é independente de ângulo de passo e pode fazer um travamento da peça em praticamente qualquer ângulo entre 0° a 360°, aumentando a precisão da máquina. Outro fator importante considerado é o custo, sendo o motor CC muito mais barato para a realização desta aplicação.

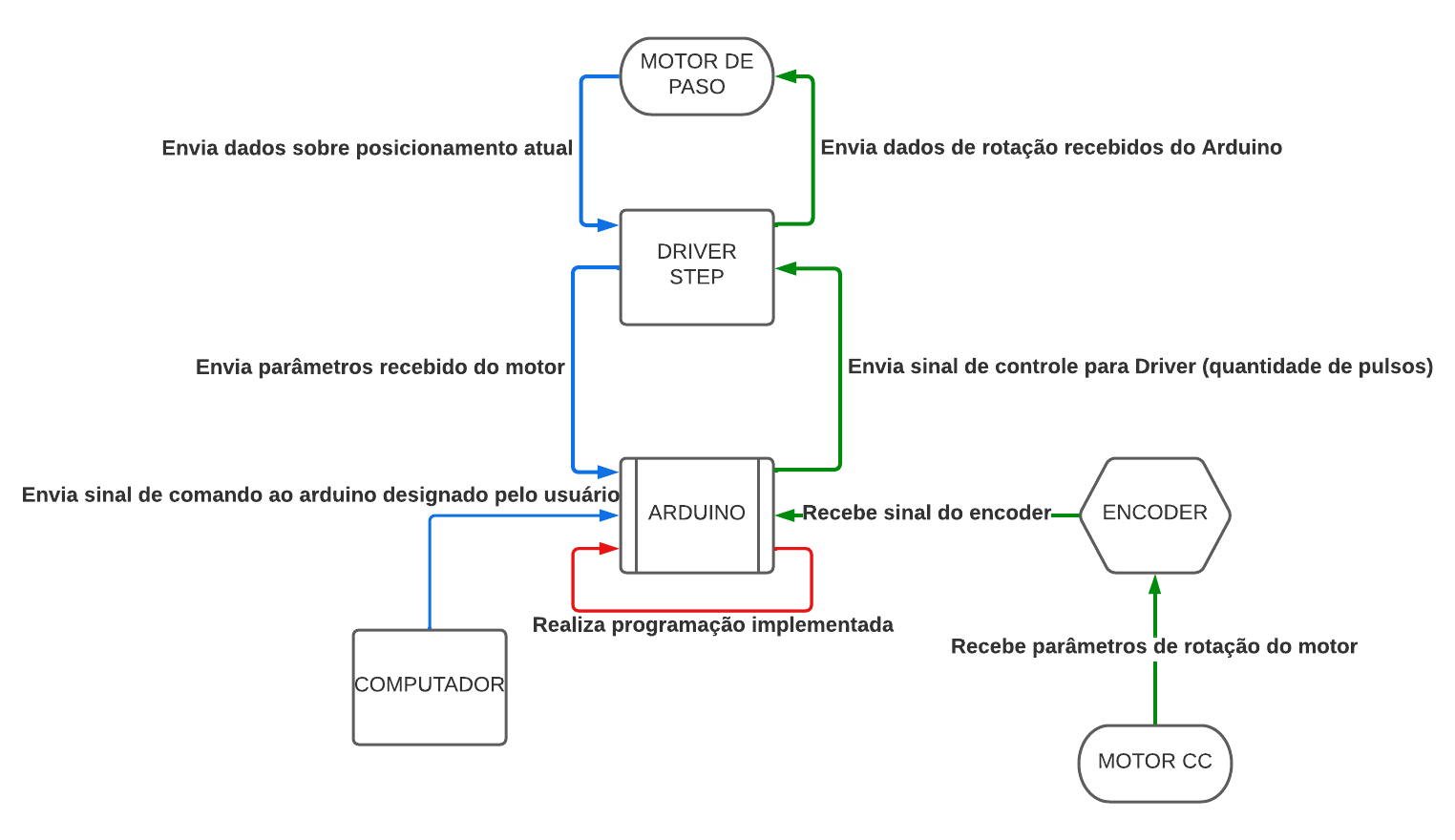
O motor escolhido para o **Eixo 1** é um motor de corrente contínua com motor-redutor, torque de 30kgf.cm e 90 RPM, com corrente máxima de 4A e tensão DC 12V, com uma potência de 10W.

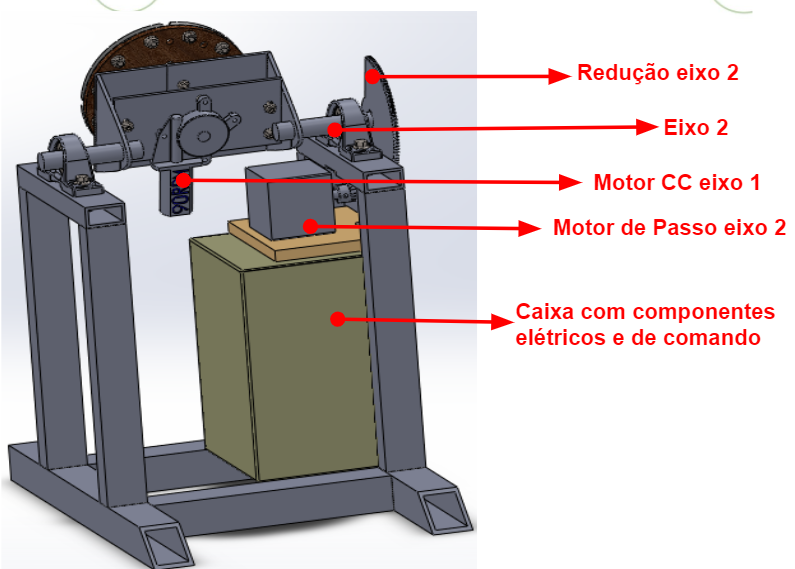
Esse motor é posicionado diretamente no eixo de rotação, sendo devidamente isolado e fixado na estrutura, paralelo a ele irá um sensor do tipo encoder incremental responsável por fazer a captação e leitura da velocidade e posição do mesmo, enviando esses valores a um Arduino, que será o controlador de toda a mesa.

Para o **Eixo 2,** foi necessário a realização de uma redução mecânica para que não seja necessário utilizar um motor com um torque muito alto, o que em termos de custo, se torna inviável. Como ainda é necessário um torque entre 30 a 40 kgf.cm o motor CC poderia ser utilizado, porém como a variável de controle mais significativa nesse eixo é o posicionamento e não mais a velocidade, um motor de passo pode ser utilizado.

O motor então escolhido foi um Motor de Passo Nema 34, com ângulo de passo de 1.8°, podendo ser melhorado através do Driver de Passo, com um torque de Holding[[1]](#footnote-0) de 52kgf.cm e corrente/fase de 5A.

Esse motor será posicionado na parte interior da máquina logo abaixo da engrenagem maior da redução mecânica. Seus parâmetros de controle são capturados diretamente pelo Driver de acionamento do motor e repassados ao Arduino.





#### Figura 2. Vista Posterior da mesa (Autoria própria).

**Ajustes Necessários**

O projeto passou por alterações de layout e correções para que tivesse êxito no atendimento dos requisitos necessários para sequência do projeto integrador, substituição das laterais estruturais para correção de problemas como empenamento, desalinhamento e erros de corte de material que ocorreram durante o processo de montagem da estrutura e dos itens do equipamento.

A solda da estrutura passou pelo processo de remoção das soldas excedentes e refeito solda da base bem como o restante das soldas estruturais refeitas e concluídas, a estrutura da base do eixo de posição passou por alterações de layout com remoção parcial do eixo 02 de movimentação e remoção da base fixa de acoplamento do motor de cc do eixo 01 a fim de contemplar o novo formato do motor e sensor que foram implementados no projeto. Foi alterado o layout do sistema de posição do eixo 02 para que a estrutura se adequar ao motor de passo como alongamento do eixo do motor e nova furação de fixação. O eixo de posição e movimentação 01 passou novamente pelo processo de usinagem para correção do excesso do diâmetro do material, o acoplamento do eixo 01 passou por ajustes para que se adequar a correia de movimentação do encoder.

O projeto demanda de alinhamento preciso dos dois eixos de movimentação e posição para que não tenha colisões estruturais bem como o funcionamento correto dos eixos a centralização foi realizada com auxílio de equipamentos de precisão para alinhamento e balanceamento estrutural para que tenha uma compensação dos erros de construção mecânica, para ajustes do nivelamento da mesa foram instalados “pés” ajustáveis que melhoram a estabilidade significativamente compensando os desníveis do local de instalação.

## Ajustes mecanicos

Neste primeiro momento não foram encontrados erros em decorrência de desalinhamento e desbalanceamento estrutural que provocam perturbações significativas ou que apresentam risco de colisão entre estrutura e base de posicionamento, foram utilizados pesos para realização do teste com carga dentro do limite estabelecido no projeto, os teste de controle do controlador em malha fechada proporcionaram resultados dentro do esperado com respostas que permitem observar a não linearidade do controlador onde é possível observar a zona morta de acionamento do motor e a zona de saturação do controlador

**Fluxograma de funcionamento elétrico**

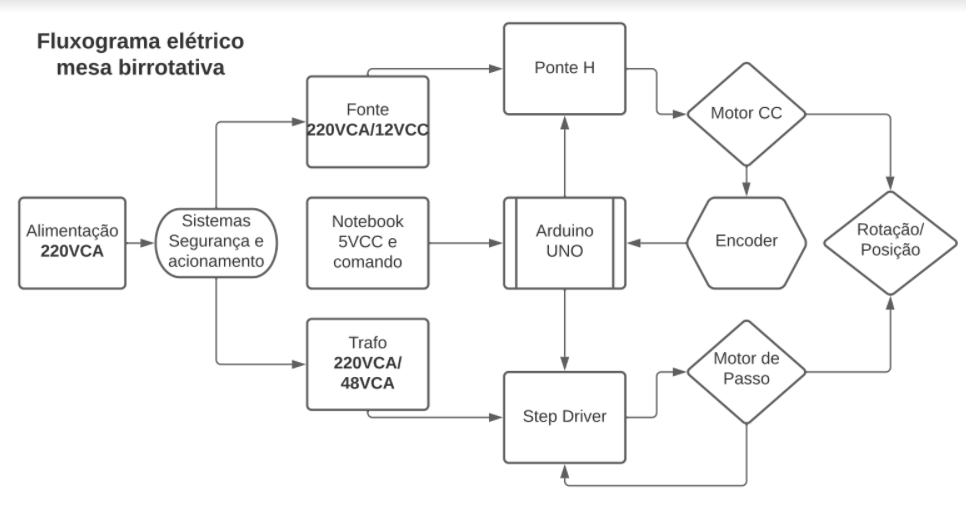
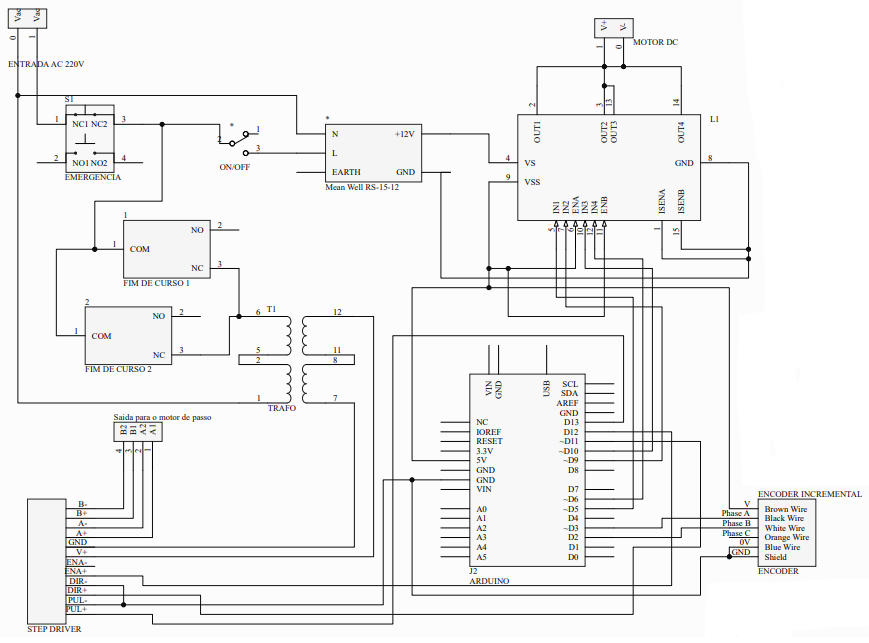


Figura N. Fluxograma elétrico. Autoria Própria.

## 2.1 Projeto elétrico

Para o projeto elétrico foi utilizado o software (CircuitMaker) de esquemático de ligações elétricas.



#### Figura 3. Esquemático do circuito elétrico. (Autoria própria).

**Tabela de tags da parte de controle e instrumentação**

| INPUT DESCRIÇÃO | OUTPUT DESCRIÇÃO |
| --- | --- |
| **D2** Microcontrolador | **Fase B** encoder |
| **D3** Microcontrolador | **Fase A** encoder |
| **D5** Microcontrolador | **IN1** ponte H |
| **D6** Microcontrolador | **IN4** ponte H |
| **D9** Microcontrolador | **IN2** ponte H |
| **D10** Microcontrolador | **IN3** ponte H |
| **D11** Microcontrolador | **DIR+** Step Driver |
| **D12** Microcontrolador | **ENA+** Step Driver |
| **D13** Microcontrolador | **PUL+** Step Driver |
| **GND** Microcontrolador | **GND** Encoder  **PUL- / DIR- / ENA-** StepDriver  **GND** ponte H |
| **5VCC** Microcontrolador | **Vss / ENA / ENB** ponte H  **V** encoder |

**Tabela de tags acionamento dos motores**

| INPUT DESCRIÇÃO | OUTPUT DESCRIÇÃO |
| --- | --- |
| **ISENA** Microcontrolador | **GND** microcontrolador |
| **ISENB** Microcontrolador | **GND** microcontrolador |
| **V+** motor CC | **OUT2 / OUT3** ponte H |
| **V-** motor CC | **OUT1 / OUT4** ponte H |
| **12Vcc** Fonte | **Vs** ponte H |
| **D10** Microcontrolador | **IN3** ponte H |
| **B-** Step Driver | **B2** Step Driver |
| **B+** Step Driver | **B1** Step Driver |
| **A-** Step Driver | **A2** Step Driver |
| **A+** Step Driver | **A1** Step Driver |
| **V+** Step Driver | **48VCA** Saída Tensão trafo |
| **GND** Step Driver | **GND** trafo |

**Tabela de tags restantes**

| INPUT DESCRIÇÃO | OUTPUT DESCRIÇÃO |
| --- | --- |
| **220VCA** | **NC1** emergência  **L** trafo  **ON** **/ OFF**  **COM** sensor fim de curso |
| **ON** **/ OFF** | **L** fonte 12VCC |
| **NEUTRO** | **NEUTRO** fonte 12VCC  **N** trafo |

# Controle e Instrumentação

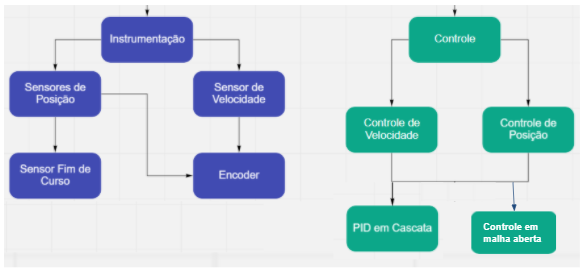
Os seguintes itens correspondem à segunda etapa da realização do projeto, equivalente à disciplina de “Projeto Integrador 2”. Nessa etapa serão idealizados e implementados conceitos de Controle e Instrumentação.

## Requisitos do cliente

Visando atender a aplicação dos conhecimentos adquiridos a respeito de Controle e Instrumentação foi proposto o atendimento a três requisitos do cliente nessa etapa, que são o controle de velocidade, controle de posição ambos no eixo 1, controle de posição em malha aberta do eixo 2.

## Detalhamento funcional

As funções parciais e elementares do projeto, na área de controle e instrumentação estão demonstradas conforme o diagrama abaixo.



#### Figura 4. Funções de controle e instrumentação. (Autoria própria).

## Matriz morfológica de controle e instrumentação

A matriz morfológica traz as possíveis soluções para realizar as funções propostas no detalhamento funcional. O objetivo da matriz é mostrar para cada uma das funções elementares descritas, as possíveis soluções disponíveis dentro do contexto do projeto.

Dentre as possibilidades apresentadas em cada um dos tópicos, os destacados em verde foram utilizados.

| **Sensores de velocidade e posição** | Encoder incremental | Encoder Absoluto | Resolver |
| --- | --- | --- | --- |
| **Sensores de posição limite - Fim de curso** | Eletromecânico | Eletromagnético | Indutivo |
| **Acionamento Motor CC** | L298N | TIP 120 | L293 |
| **Acionamento motor de passo** | Stepper-Driver AC |  |  |
| **Controle dos motores** | Arduino UNO | Arduino Mega |  |

#### Tabela 1 - Componentes de controle e instrumentação.(Autoria própria).

## Matriz de avaliação de controle instrumentação

Para auxiliar na tomada de decisão a matriz de avaliação consiste em manipular os dados disponíveis a partir da matriz morfológica, cada tabela corresponde aos itens elencados anteriormente na tabela 1.

Selecionados três itens de avaliação para cada matriz é possível distribuir pesos de importância (0 a 10) e em seguida, realizar a avaliação para cada item incluindo notas para os mesmos. Com a tabela e valores montados pode-se calcular a média de cada item e por fim escolher o mais adequado.

| **Sensores de velocidade e posição** | **Peso** | **Encoder incremental** | **Encoder absoluto** | **Resolver** |
| --- | --- | --- | --- | --- |
| Custo | **10** | 9 | 3 | 1 |
| Resolução | **10** | 9 | 7 | 10 |
| Precisão | **9** | 7 | 10 | 7 |
| **Total** |  | **8.1** | **6.3** | **5.7** |

| **Sensores de posição limite - Fim de curso** | **Peso** | **Mecânico** | **Magnético** | **Indutivo** |
| --- | --- | --- | --- | --- |
| Custo | **10** | 9 | 10 | 6 |
| Sensibilidade | **8** | 6 | 6 | 8 |
| Praticidade | **9** | 10 | 8 | 5 |
| **Total** |  | **7.6** | **7.3** | **5.6** |

| **Acionamento Motor CC** | **Peso** | **L298N** | **TIP120** | **L293** |
| --- | --- | --- | --- | --- |
| Custo | **10** | 10 | 10 | 5 |
| Corrente de pico | **10** | 9 | 0 | 10 |
| Tensão de Operação | **9** | 9 | 10 | 10 |
| **Total** |  | **9** | **6.3** | **8** |

| **Acionamento do motor de passo** | **Peso** | **Driver-stepper AC** |  |  |
| --- | --- | --- | --- | --- |
| Custo | **10** | 10 |  |  |
| Tensão de operação | **7** | 7 |  |  |
| Confiabilidade | **9** | 10 |  |  |
| **Total** |  | **8** |  |  |

| **Controle dos motores** | **Peso** | **Arduino UNO** | **Arduino MEGA** |  |
| --- | --- | --- | --- | --- |
| Custo | **10** | 10 | 6 |  |
| Tensão de operação | **7** | 7 | 7 |  |
| Confiabilidade | **9** | 10 | 10 |  |
| **Total** |  | **8** | **6.6** |  |

#### Tabela 2 - Matriz avaliação de controle e instrumentação. (Autoria própria).

## Métodos práticos

### Definição dos controladores para os eixos

Devido a necessidade de controlar a velocidade no eixo 1, e posição em ambos os eixos, e necessitar de um motor de passo no eixo 2, foi optado por um controlador em malha aberta no eixo 2, e um controlador em cascata de velocidade e posição no eixo 1.

### Hardware necessário para o sistema de controle e instrumentação

Por meio das conclusões obtidas a partir da matriz de avaliação em associação com a matriz morfológica foi possível determinar o Hardware que seria utilizado no sistema de controle e instrumentação, sendo este descrito logo abaixo:

* 1 Microcontrolador Arduíno Uno
* 1 Driver-stepper AC
* 1 Ponte H L298N
* 1 Encoder Incremental (E6B2-CWZ6C)

Na sequência, estes componentes serão apresentados.

* **Microcontrolador Arduino Uno**

O microcontrolador é utilizado para as variáveis controladas do sistema, combinado com o sistema de controle elaborado por meio de análises realizadas pelo grupo é possível alcançar os objetivos de controle desejados, sendo eles velocidade e posicionamento.

Os dados retirados do datasheet do Arduino são:

* Memória: AVR CPU até 16 MHz; 32KB Flash 2KB SRAM; 1KB EEPROM
* Periféricos: 2x 8-bit Timer/Counter com um período dedicado do registrador e canais comparadores; 1x 16-bit Timer/Counter com um período dedicado de registrador, input de captura e canal comparador; 1x USART com baud rate gerador fraccionado; 1x controlador serial periférico de interface (SPI); 1x Dual mode controlador I2C; 1x Comparador analógico com uma input de referência escalada; Watchdog Timer com um on-chip oscillator separado; Seis canais PWM Interrupt e wake-up em mudança de pino.
* Alimentação: 2.7- 5.5 volts.
* Os seguintes pinos estarão em uso no projeto: GND; 5V; D11; D10; D9; D6; D5; D4; D3; D2. Na página 6, no projeto elétrico é possível verificar as conexões.
* Uma placa de circuito foi elaborada de forma manual para melhor conexão, evitando mal contato nos pinos.
* **Ponte H L298N**

A utilização da Ponte H faz-se necessária para poder inverter o sentido de rotação do motor CC

Os dados da Ponte H retirados do datasheet da mesma são:

* Tensão de alimentação operacional até 46 V; Corrente total DC até 4 A; Baixa tensão de saturação; Proteção de sobretemperatura; Tensão de entrada Lógica de 0 até 1.5 V com imunidade a altos ruídos.
* **Encoder incremental E6B2-CWZ6C**

Para que seja possível fazer o controle da planta é necessário alguns sensores, sendo o principal para o projeto o encoder, o qual fará a leitura de velocidade e posição do sistema de rotação.

* Os dados do encoder encontram-se abaixo e foram retirados do datasheet oficial do sensor:

Tensão de alimentação 5 VDC a 24 VDC +15%, ripple: 5% máx; Corrente de consumo 80 mA máx; Resolução (pulsos/rotação) 10, 20, 30, 40, 50, 60, 100, 200, 300, 360, 400, 500, 600, 720, 800, 1,000, 1,024, 1,200, 1,500, 1,800, 2,000; Fases de saída A, B e Z; Configuração de saída coletor aberto NPN; Resposta máxima de frequência 100 kHz; Tempos de subida e descida da saída 1 µs máx; Velocidade máxima permitida 6000 r/min; Temperatura de operação de -10 a 70°C; Peso aproximado 100 g; Zona morta (momento de inércia) 1×10−6 kg·m2 max.; 3 × 10−7 kg·m2 max. at 600 P/R max.

* **Driver-stepper AC**

O Drive do motor de passo servirá para acionamento, proteção e controle do motor de passo implementado no projeto. O modelo utilizado em questão possui diversas correntes e Pulsos/revolução, sendo selecionados via switch de seleção, de acordo com a necessidade

* Tensão de alimentação 18VAC a 80 VAC; Corrente de referência 2A, 2.57A, 3.14A, 3.71A, 4.28A, 4.86A, 5.43A, 6A; Corrente de pico 2.4A, 3.08A, 3.77A, 4.45A, 5.14A, 5.83A, 6.52A, 7.20A; Pulsos/revolução 400, 800, 1600, 3200, 6400, 12800, 25600, 51200, 1000, 2000, 4000, 5000, 10000, 20000, 40000.

# Métodos e análises

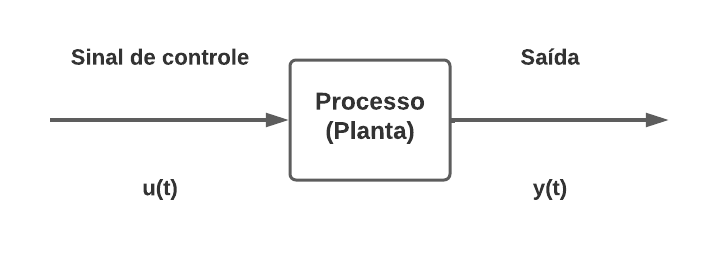
## Controle de posição em malha aberta

### Coleta de dados

Devido ao controle de posição do eixo 2 ser de malha aberta, não há coleta de dados em tempo real por sensores, então será analisado a partir da entrada de pulsos uma relação entre a movimentação do eixo e o número de pulsos por segundo enviados para a lógica do driver-stepper.

### Definição do controlador

Como um controle de malha aberta consiste em aplicar um sinal de controle, esperando-se que ao final de um tempo a variável controlada atinja um determinado valor. Nesse sistema de controle não serão utilizadas informações obtidas a partir da coleta contínua de dados. Mais especificamente, o sinal de controle não é calculado a partir de uma medição do sinal de saída, conforme demonstrado na figura abaixo :



**Figura 5. Diagrama de controle em malha aberta**

### Implementação

Posteriormente a análise do sistema em malha aberta, uma relação entre a quantidade de passos, é dimensionada, para isso alguns parâmetros são considerados, como a relação de 10:1 entre as engrenagens a equação do controlador será implementada na plataforma do Arduino por meio de código conforme será descrito no Apêndice A.

## Controle de Velocidade e posição em cascata

### Coleta de dados

Serão realizadas coletas de dados para identificação da função de transferência do sistema variando o fornecimento de potência via alteração da razão cíclica do PWM. Após serem definidas as faixas de coletas, os dados que forem coletados serão analisados via software MATLAB, utilizando de uma toolbox chamada *ident*, onde a será possível gerar uma função de transferência G(s), a qual possuirá uma taxa de aproximação, denominada Best Fits, que determina a taxa de aproximação da função obtida com os dados coletados.

A partir da função de transferência G(s) será possível analisar a resposta a um degrau unitário da planta em malha aberta e também observar um tempo de assentamento (Ts), utilizando então essas informações será definido o controlador, conforme a se explicará brevemente a seguir.

### Definição do controlador

Após a análise dos dados, será utilizado um sistema de controle em cascata, porque as variáveis de controle são diretamente correlacionadas, de modo que considerando uma função de posição como **θn** é possível derivá-la para obter uma função de velocidade **θ’n**, e da mesma forma é possível integrar **θ’n**, obtendo **θn**, por causa dessa relação é possível utilizar o método de cascata, para simplificar o controle do sistema.

Graças a esse método que é amplamente utilizado no controle de braços robóticos, haverá no sistema um controlador para velocidade, que ficará numa malha secundária, seguido por um integrador, tornando o controlador de velocidade dependente do controlador de posição, conforme diagrama a seguir:

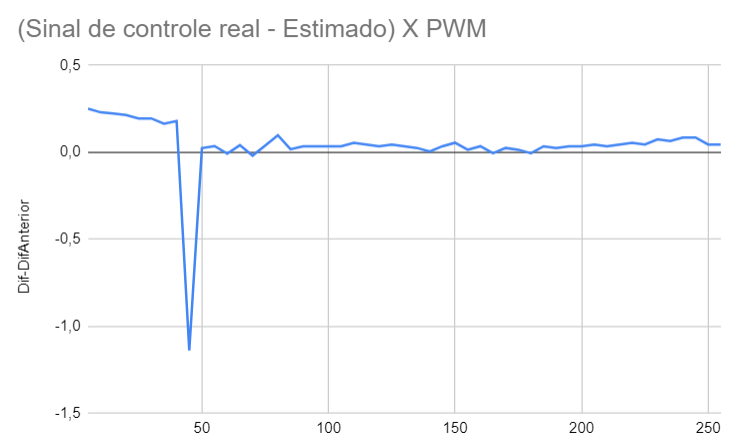
#### Figura 6. Diagrama de controle em cascata de um braço robótico.

### Implementação

Após a análise do sistema, a equação do controlador em cascata será implementada na plataforma do Arduino por meio de código.

Com o posicionamento do motor e do encoder incremental utilizado no projeto, e a partir de um código desenvolvido foi possível coletar dados, com variáveis de controle definidas pelo grupo. Estes dados foram analisados e tratados, dos mesmos foi possível retirar dois gráficos informativos sobre o comportamento do motor CC de acordo com o valor do PWM escolhido.

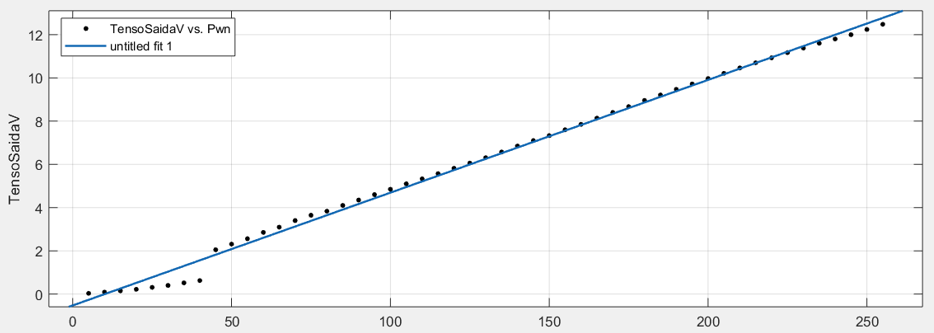
Utilizando uma tensão de entrada constante de 14,5 V e selecionando valores do PWM a partir de 5 até 255, incrementado de 5 em 5 é possível medir a tensão de saída real para o motor e comparando com uma tensão estimada calculada (figura 6) a partir de



**Figura 6. Gráfico (Tensão real - tensão estimada) x PWM (Autoria Própria)**

É possível perceber uma zona morta no motor, onde ele não é ativado, pois a tensão de saída do mesmo não é suficiente para acioná-lo. Essa zona morta está nos valores de PWM de 5 até 40, fazendo com que o motor comece a rotacionar somente com o valor de PWM=45.

No gráfico abaixo (tensão de saída x PWM) é possível observar essa zona morta e o instante em que o mesmo começa seu movimento (Tensão saída= 2V) de forma constante de acordo com valores de PWM.



**Figura 7. Gráfico Tensão saída x PWM (Autoria Própria)**

Considerando os dados anteriores é possível afirmar que há uma perda de tensão na ponte H, que pode ser representada a partir da equação , obtida pela ferramenta *Curve Fitting* do software Matlab.

Tendo isso em consideração na seguinte coleta de dados foi utilizado um filtro projetado para amenizar esses efeitos a fim de gerar uma planta mais fiel para o modelo.

Na seguinte coleta de dados, a fim de obter uma planta que atuasse na região desejada de operação e desconsiderasse a região de zona morta do motor e a região de não linearidade que é até 30 RPM, os dados foram coletados em um intervalo de 102 a 255 PWM, aplicando diversos sinais do tipo degrau de diferentes amplitudes, coletando um bom número de amostras por degrau e utilizando um período de amostragem de 0,02 segundos foram coletados 215 dados, conforme a tabela exemplo abaixo:

| u | y |
| --- | --- |
| 204.00 | 14.75 |
| 204.00 | 43.75 |
| 204.00 | 58.25 |
| 204.00 | 65.00 |
| 204.00 | 67.50 |
| 204.00 | 69.25 |
| 204.00 | 70.00 |
| 204.00 | 70.50 |
| 204.00 | 70.75 |

#### Tabela 4. Demonstração dos dados coletados.(Autoria própria)

Onde u é a entrada do degrau em PWM, e y é a saída do encoder já em RPM, então os dados foram tratados no Matlab utilizando o *framework ident* para gerar uma função de transferência para o modelo. Conforme abaixo:

##### 

##### Equação (1)

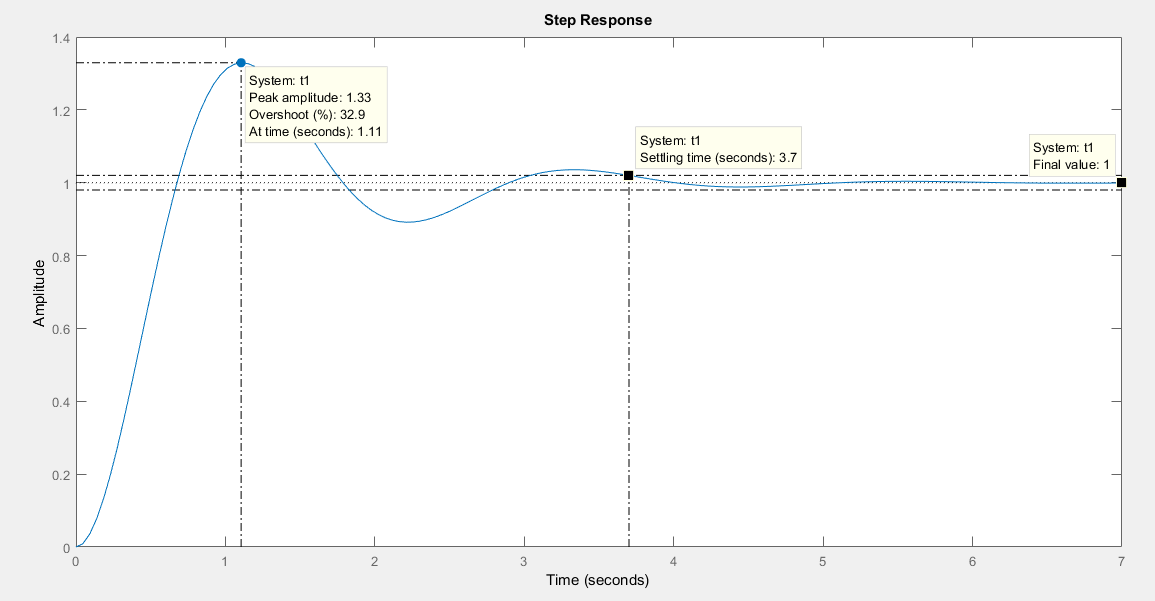
Agora com a planta do sistema em mãos é possível analisar o comportamento do sistema. Segundo, Nise(verificar ano e referência) é possível separar em três tipos as respostas de um sistema de segunda ordem com uma entrada ao degrau e as respectivas influências de seus pólos em suas respostas que são:

O sistema subamortecido, sobreamortecido e criticamente amortecido.

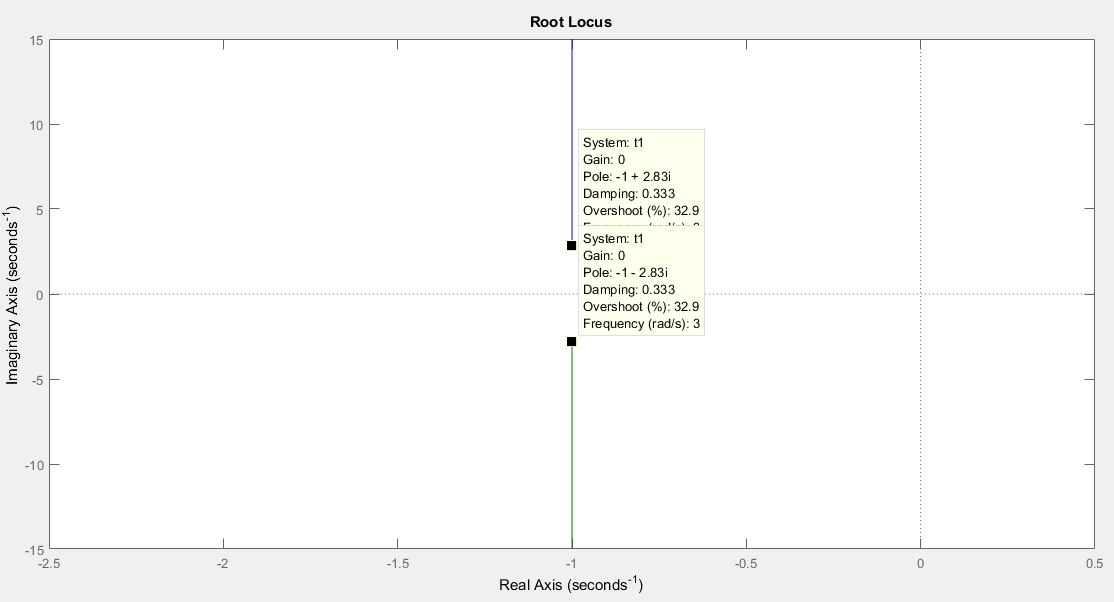
* O sistema subamortecido possui dois pólos (raízes da equação característica) complexos e conjugados;
* O sistema sobreamortecido possui dois pólos reais e distintos;
* O sistema criticamente amortecido possui dois pólos reais iguais;

Analisando graficamente obtemos:

* Sistema subamortecido:



#### Figura x. Resposta ao degrau de um sistema subamortecido. Fonte(Autoria própria)



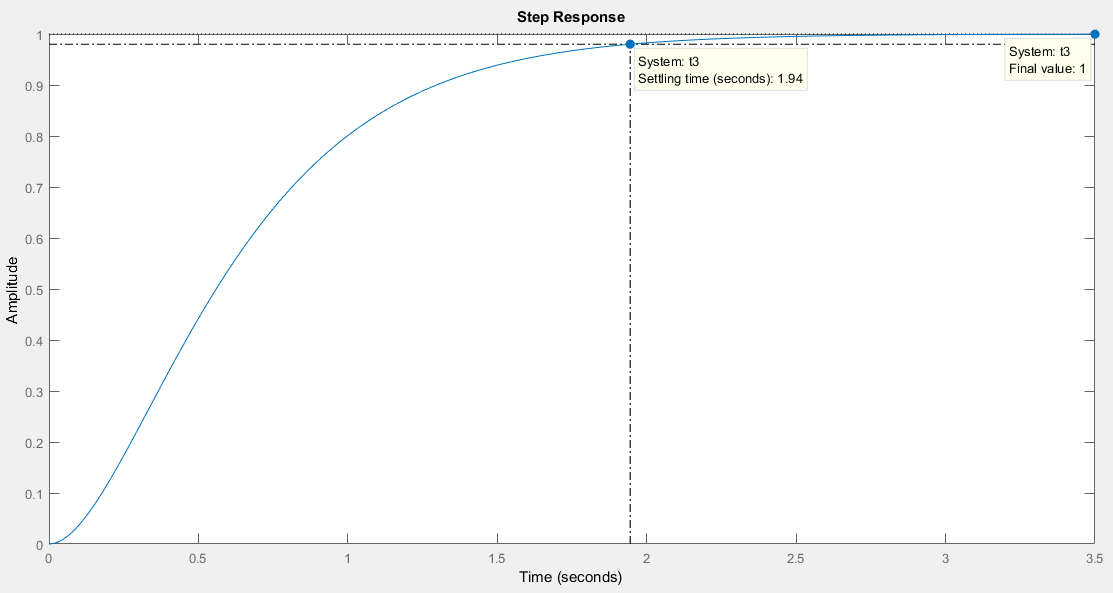
#### Figura x. Lugar das raízes de um sistema subamortecido. Fonte(Autoria própria)

* Sistema sobreamortecido:

#### **Figura x. Resposta ao degrau de um sistema s**obre**amortecido. Fonte(Autoria própria)**

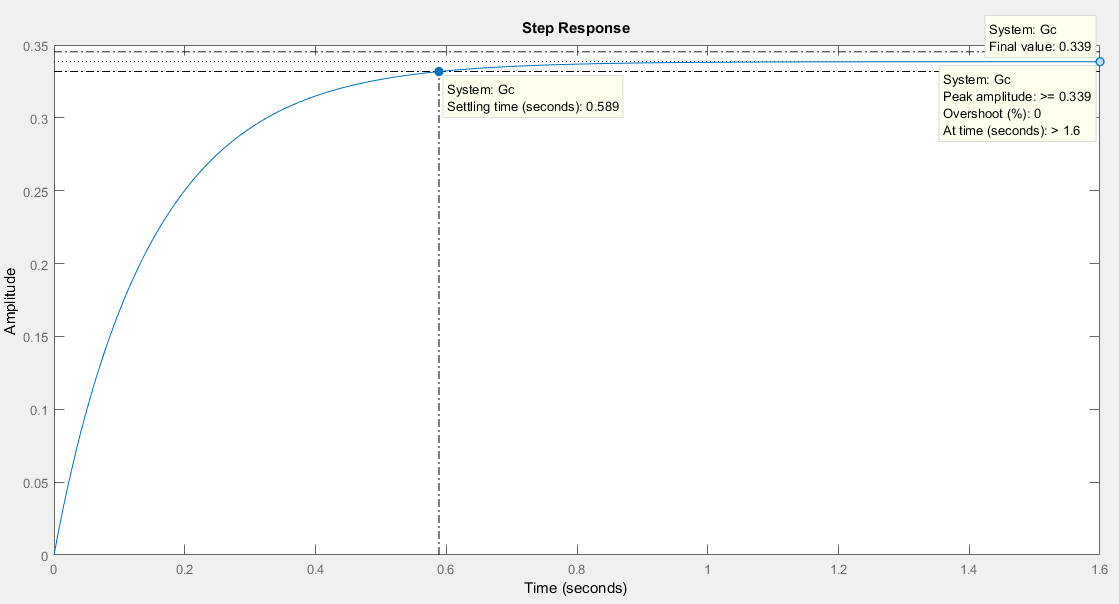
#### **Figura x. Lugar das raízes de um sistema s**obrea**mortecido. Fonte(Autoria própria)**

* Sistema criticamente amortecido:



#### Figura x. Resposta ao degrau de um sistema criticamente amortecido. Fonte(Autoria própria)

#### **Figura x. Lugar das raízes de um sistema** criticamente **amortecido. Fonte(Autoria própria)**

Tendo em vista esses conceitos de comportamento do sistema então a planta é analisada a fim de definir o comportamento da mesma, a seguir é possível ver a resposta ao degrau e o lugar das raízes da planta.

#### Figura x. Resposta ao degrau da planta do sistema.

#### **Figura x.** Lugar das raízes **da planta do sistema.**

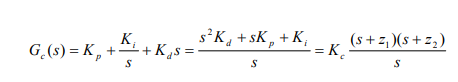
Como é possível ver nas figuras acima a planta dois pólos reais e distintos, -6,61 e -34,1 e um zero em -31,2, logo o sistema é sobreamortecido.

Então foram iniciados os projetos dos controladores em cascata levando isso em consideração. Conforme previamente abordado para o controle de velocidade o sistema será controlado por um controle PID em cascata, onde foi definido o modelo do PID como:

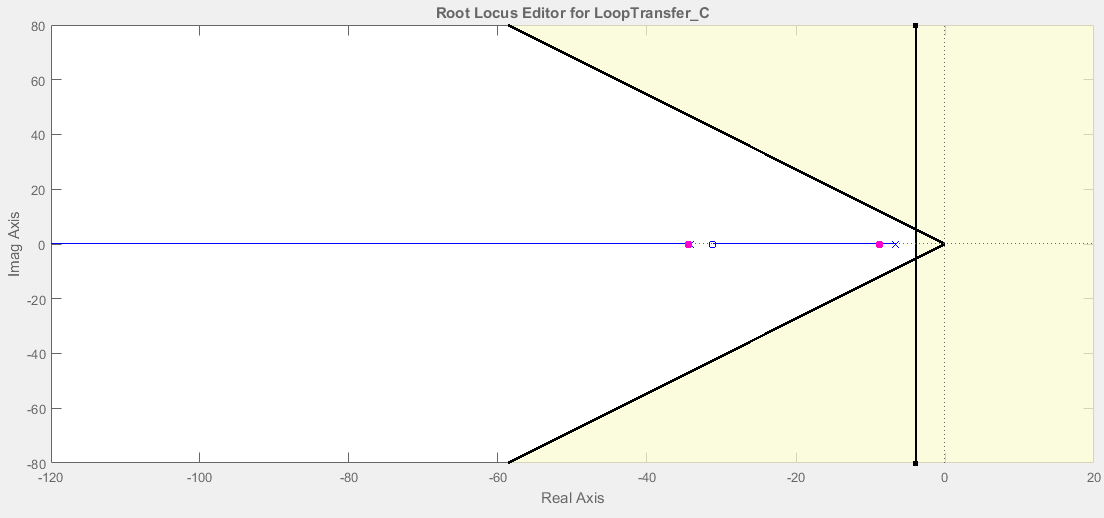
Tendo em vista que os requisitos para controle do sistema eram de:

* Erro em Regime permanente aproximadamente 0;
* Overshoot menor que 10%;
* Tempo de assentamento aproximadamente 1 segundo;

Após a definição dos requisitos de controle, é dado início ao projeto do controle PID de velocidade, utilizando a equação anterior de PID que pode ser reescrita da seguinte forma:



A partir da equação acima é mais claro que para o projeto do controlador será necessário alocar um polo em zero e dois zeros (), para isso então foi utilizado a ferramenta *rltool* do Matlab, onde os requisitos foram adicionados na opção de design da ferramenta a fim de gerar um guia para o projeto.



#### Figura x. Planta do sistema e os designs no rltool.

Para definir os dois zeros do controlador foi utilizado

Tendo esses requisitos em mente foi calculado o controlador PID de velocidade, com os seguintes parâmetros.

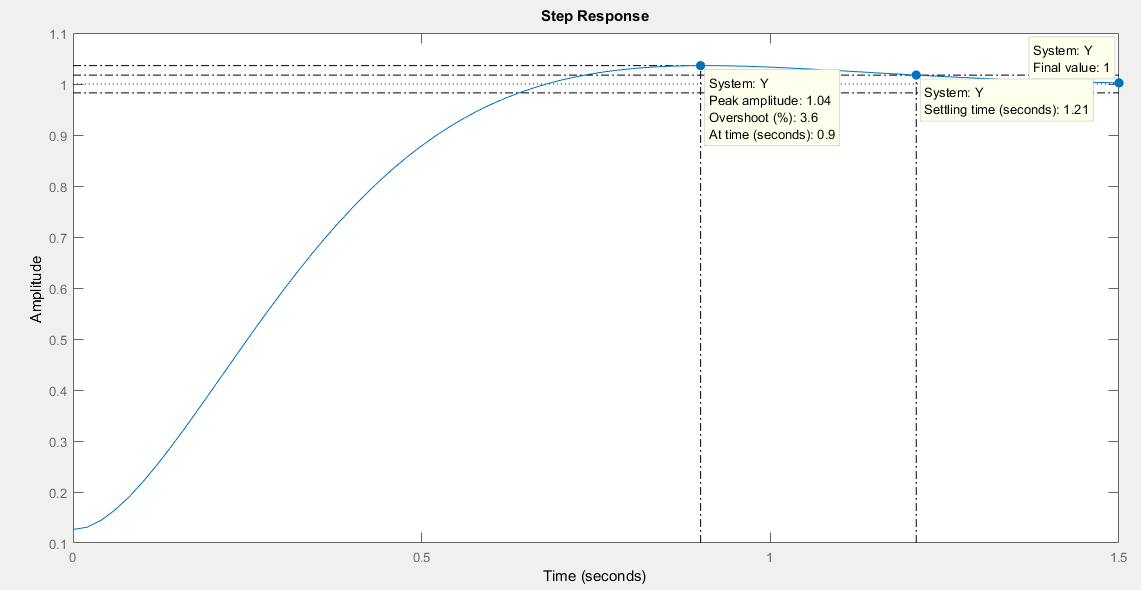
**Kp** = 0.53282;

**Ki** = 11.9894;

**Kd** = 0.059198;

Gerando assim o seguinte controlador:

##### Equação(2)

A resposta do sistema com o controlador a um degrau pode ser vista na figura(10):

#### Figura 10. Resposta ao degrau do sistema controlado.

Onde é possível ver que todos requisitos foram atendidos, assegurando assim um bom controle de velocidade para o sistema.

Como é necessário além do controlador de velocidade outro para posicionamento é utilizado o método em cascata, onde os controladores são colocados em sequência atuando em conjunto no sistema.

Para o controle de posição o projeto inicial previa outro PID, porém conforme o projeto avançou foi visto que os termos Ki e Kd tendiam a zero em todas propostas, devido a robustez do controlador de velocidade, então o controlador acabou resumindo-se a um Controlador P, onde é calculado apenas o Kp, a partir de agora chamado de Kp2.

Como o modelo da planta está em RPM, é necessário uma conversão para o modelo de posição visto que a integração de RPM, não gera um constante que o sensor possa fazer a leitura, devido a leitura em quadratura do encoder e a relação de redução entre o eixo do motor CC e o encoder, cada volta do motor corresponde a 12000 passos do encoder.

Considerando uma velocidade de 40 RPM como a velocidade máxima para uma solda consistente, logo o controlador de posição não pode mandar um sinal maior que 40 para o controlador de velocidade, então a partir disso é possível fazer dizer que para um sinal de referência de 1 volta, o controlador de posição deve receber um sinal de 12000, enquanto o controlador de posição deve receber um sinal de 40.

Tendo esses parâmetros em mente, foi calculado um Kp2 utilizando uma regra de três, resultando em aproximadamente 0,03333 da seguinte forma:

Kp2 está para 40, como 12000 está para 1.

Levando todas informações anteriores em consideração foi necessário uma atualização do diagrama de blocos, conforme na figura a seguir:

#### Figura 11. Diagrama de blocos da malha de controle atualizado.

### Discretização

Para a discretização do controlador de velocidade foi adotado uma estratégia de discretização onde é considerado a equação do PID já no domínio do tempo, e também a discretização é feita termo a termo, a partir dos métodos integradores digitais e derivadores digitais são obtidas as equações a diferenças.

Para a discretização do controlador foi utilizado o método backward de Euler. Desta forma, a função do controlador discreta C(z), pode ser obtida a partir de C(s), através da seguinte equação:

##### 

##### Equação (3)

Com isso a função transferência do controlador, descrita na equação (2) é substituindo a equação (3) gerando o seguinte resultado:

##### 

##### Equação(4)

**Resultados práticos**

**Conclusao**

# Referências bibliográficas

Robótica ABB, Acesso em 18 de out. 2021. Disponível em: <<https://new.abb.com/products/robotics/pt>>;

[2]Denise Ying , ... Hongliang Ren , em [Robótica Flexível em Medicina](https://www.sciencedirect.com/book/9780128175958/flexible-robotics-in-medicine) , 2020, disponível em: < <https://www.sciencedirect.com/topics/engineering/holding-torque>>, Acesso em: 05 de mar. 2022

Livro: soldagem fundamentos e tecnologia ,3 edição, editora UFMG de 2009, AUTORES: Paulo Villani Marques, Paulo José Modenesi, Alexandre Queiroz Bracarense;

Acesso em 22 de out. 2021 Disponível em: <<https://www.filipeflop.com/categoria/motores/>>;

Schneider Eletric, Acesso em 22 de ago 2021. Disponível em: <<https://www.regulacni-pohony.cz/soubor/manual-bmh-2016-03-en-pdf>>;

De Motor, Acesso em 22 de out. 2021. Disponível em: <<https://pt.demotor.net/motores-electricos/motores-de-corrente-alternada>>;

CNC, Acesso em 23 de out. 2021. Disponível em: <<https://www.amazon.com/Router-Machine-Rotational-2-Phase-Stepper/dp/B07G94C6DX>>;

Eixo Rotativo, Acesso em 23 de out. 2021. Disponível em: <<https://pt.aliexpress.com/item/32881756224.html>>;

Tubos de aço, Acesso em 30 de out. 2021. Disponível em: <<https://acotubo.com.br/wp-content/uploads/2016/08/ACO_005_Catalogos_Acotubo2016_OnLine_02_TubosAco.pdf>>

Datasheet Arduino UNO, Acesso em 01 de fev. 2022. Disponível em: <<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>>

Siciliano et al, Robotics – Modelling planning and control, 2009

# Apêndice A - Códigos Arduino

## Bibliotecas utilizadas

No projeto foram utilizadas duas bibliotecas a fim de otimizar a performance do Arduino, Encoder.h e PID\_v1.h, ambas disponíveis na interface de programação disponibilizada pelo fabricante.

## Por que utilizar uma biblioteca?

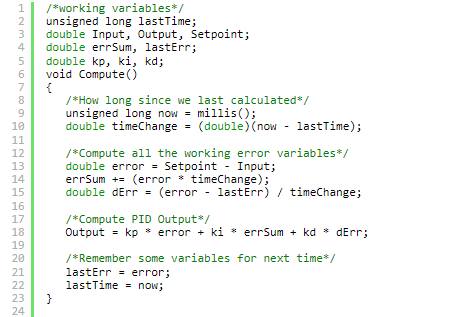
Uma biblioteca se assemelha a uma função criada pelo usuário, a diferença é que é que há mais agilidade, pois pode-se usar programas distintos; É possível escrever códigos menores e mais organizados. Assim, se reduz a chance de erros e retrabalhos; Ela facilita a atualização de programas diversos. Como previamente dito foram utilizadas as bibliotecas Encoder.h e PID\_v1.h, descritas a seguir:

## Encoder.h

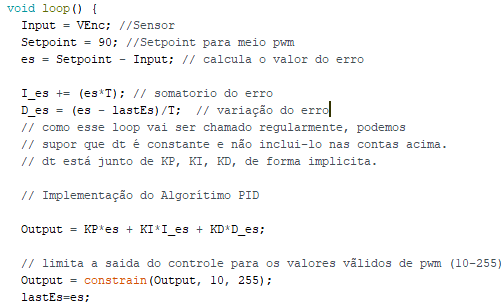
Essa biblioteca é a responsável pela leitura dos pulsos do encoder, torna-se necessário o uso da mesma devido a necessidade de performance na leitura, já que uma leitura otimizada requer uma pesquisa muito maior.

## PID\_v1.h

Essa biblioteca é responsável pelo cálculo do PID, utilização dessa justifica-se por dois motivos, reduzir o tamanho do código principal, e otimização do código, porque a biblioteca trabalha com um tempo de amostragem real e um alocamento de memória otimizado, gerando um tempo de resposta menor até a ação do controlador. Um comparativo entre os dois códigos é demonstrado a seguir:



#### Figura A1. Parte do código implementado na biblioteca.



#### Figura A2. Parte do Código implementado pelo Grupo.

Como pode ser observado pelas imagens anteriores, a parte principal de ambos códigos é praticamente idêntica, ocorrendo apenas uma diferença que onde, foi utilizado um tempo de amostragem T, como um valor fixo na figura A2, o mesmo é um valor dinâmico na biblioteca, onde o mesmo acaba gerando mais precisão na hora do controlador atuar sobre o sistema.

## Código Utilizado Completo

#include <PID\_v1.h>

#include <Encoder.h>

//declaracao dos pinos utilizados para controlar o motor

const int PINO\_IN1 = 5;

const int PINO\_IN4 = 6;

const int PINO\_IN2 = 9;

const int PINO\_IN3 = 10;

//Variaveis do encoder

Encoder Enc(2, 3);

float EncAtual = 0.0;

float EncAnterior = 0.0;

double T = 0.02;//Período de amostragem (T = )

unsigned long lastSend = 0;

//Define variaveis pro pid de velocidade

double Setpoint, vSetpoint, Input, Output;

//Variaveis para o controle de posicao

double pSetpoint, ErroP, Cp;

double P0 = 0.0; //Posicao inicial

float DEnc = 0.0; //Diferenca entre leituras

float VEnc = 0.0; // Velocidade em Rpm

//Parametros dos controladores

double Kp = 0.53282;

double Ki = 11.9894;

double Kd = 0.059198;

Kp2= 0.003183098862

PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

//Timer 1 interrupcao

ISR(TIMER1\_COMPA\_vect) {

EncAtual = (Enc.read()); //recebe o valor da interrupção

DEnc = EncAtual - EncAnterior;

EncAnterior = EncAtual;

VEnc = (60 \* DEnc / (12000 \* T));

}

//Configuracao timer1

void Tempo\_Config\_1(void)

{

TCCR1B = 0;

TCCR1A = 0;

TIMSK1 = 0;

// Timer de 16bits, conta até 65535

TCCR1B |= (1 << WGM12) | (1 << CS12); //Timer em CTC e prescaler de 256

//TCCR1B |= (1 << WGM12) | (1 << CS10)| (1 << CS12); //Timer em CTC e prescaler de 1024

TIMSK1 = (1 << OCIE1A); //Interrupção no COMPA

//Com prescaler de 8, no modo CTC f = fcpu/(Prescaler\*(1+OCR1A))

//f = 16Hz

//T = 0.02s

OCR1A = ((16000000 / 256) \* T) - 1;

}

void setup() {

Tempo\_Config\_1(); //Seta o tempo de interrupção

Serial.begin(115200);

//configuracao dos pinos como saida

pinMode(PINO\_IN1, OUTPUT);

pinMode(PINO\_IN2, OUTPUT);

pinMode(PINO\_IN3, OUTPUT);

pinMode(PINO\_IN4, OUTPUT);

//inicia o codigo com os motores parados

digitalWrite(PINO\_IN1, LOW);

digitalWrite(PINO\_IN2, LOW);

digitalWrite(PINO\_IN3, LOW);

digitalWrite(PINO\_IN4, LOW);

//Inicia led de controle

pinMode(LED\_BUILTIN, OUTPUT);

P0 = Enc.read();

Input = 0;

Setpoint = 0;

myPID.SetMode(AUTOMATIC);

}

void loop() {

pSetpoint = 12000; //Setpoint para 1 volta

vSetpoint = 40;

ErroP = (pSetpoint + P0) - (EncAtual);

Input = EncAtual \* Kp2;

Cp = min(((P0 + pSetpoint) \* Kp2), vSetpoint);

Setpoint = Cp;

myPID.Compute();

//Define sentido

if (pSetpoint > 0) {

analogWrite(PINO\_IN1, Output);

analogWrite(PINO\_IN4, Output);

} else {

Cp = min(((P0 - pSetpoint) \* Kp2), vSetpoint);

Setpoint = Cp;

myPID.Compute();

analogWrite(PINO\_IN2, Output);

analogWrite(PINO\_IN3, Output);

}

if (millis() - lastSend > 80) {

lastSend = millis();

Serial.print(Setpoint);

Serial.print(" ");

Serial.print(ErroP);

Serial.print(" ");

Serial.print(Input);

//Serial.print(" ");

// Serial.print(Output);

Serial.println(" ");

}

}

1. Uma propriedade específica do [motor de passo](https://www.sciencedirect.com/topics/engineering/stepper-motor) é o torque de retenção ou holding torque . Isso significa que uma corrente contínua é extraída da fonte de alimentação para garantir que o eixo do motor de passo seja mantido estacionário e não gire quando submetido a forças externas. [2] [↑](#footnote-ref-0)