TRABALHO DE GRADUAÇÃO

SERVOMECANISMO PARA ATUAÇÃO NAS JUNTAS DO ROBÔ EDUCACIONAL RHINO XR-4

Stephany Ribeiro Rodrigues

Brasília, julho de 2018

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

SERVOMECANISMO PARA ATUAÇÃO NAS JUNTAS DO ROBÔ EDUCACIONAL RHINO XR-4

Stephany Ribeiro Rodrigues

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Lélio Ribeiro Soares Júnior, ENE/UnB _____
Orientador

Prof. Carlos Humberto Llanos Quintero, ENM/UnB _____
Examinador interno

Prof. Thiago Felippe Kurudez Cordeiro, FGA/UnB _____
Examinador interno

Brasília, julho de 2018

FICHA CATALOGRÁFICA

STEPHANY, RIBEIRO RODRIGUES
Servomecanismo para atuação nas juntas do robô educacional Rhino XR-4,

[Distrito Federal] 2018.

vii, 49p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2018). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1. Controle
2. Identificação de sistemas
3. Servomecanismo
4. Robô manipulador

I. Mecatrônica/FT/UnB
II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

RODRIGUES, STEPHANY RIBEIRO, (2018). Servomecanismo para atuação nas juntas do robô educacional Rhino XR-4. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n°3$/2018, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 49p.

CESSÃO DE DIREITOS

AUTOR: Stephany Ribeiro Rodrigues

TÍTULO DO TRABALHO DE GRADUAÇÃO: Servomecanismo para atuação nas juntas do robô educacional Rhino XR-4.

GRAU: Engenheiro ANO: 2018

É concedida à Universidade de Brasília permissão para reproduzir cópias deste Trabalho de Graduação e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte desse Trabalho de Graduação pode ser reproduzida sem autorização por escrito do autor.

Stephany Ribeiro Rodrigues

71931-360 Brasília – DF – Brasil.

Dedicatória

Dedico este trabalho à minha família.

Stephany Ribeiro Rodrigues

Agradecimentos

Agradeço à minha família e amigos pela paciência e apoio a mim dado todos esses anos de graduação. Sem eles eu não teria chegado até aqui.

Stephany Ribeiro Rodrigues

RESUMO

O trabalho a seguir consiste no projeto de um servomecanismo que atue nos motores de juntas de um robô manipulador articulado de 5 graus de liberdade. Aqui propõe-se levantar o modelo dinâmico do motor de corrente contínua e realizar o controle de velocidade e posição usando o controle em cascata com dois controladores, usando métodos de sincronização PID encontrados na literatura e testados em simulação. Para testes no motor real, foram criadas rotinas para medida da velocidade e da posição a partir dos sinais do *encoder* presente no conjunto do motor, além de rotinas de controle feitas a partir da discretização dos controladores obtidos em tempo contínuo, e o acionamento do motor pelo controle de potência realizado por meio de sinal PWM. Os experimentos realizados em um dos motores com o controlador em cascata se mostraram satisfatórios e o motor foi capaz de seguir sem erro uma referência de posição em rampa.

Palavras Chave: Controle em cascata, Motor DC, encoder, PID, Servomecanismo

ABSTRACT

The following work consists on the design of a servomechanism that acts on the joint engines of a 5-degree-of-freedom articulated manipulator robot. The proposal is to raise the dynamic model of the DC motor and perform the speed and position control using the cascade control with two controllers, using PID synchronization methods found in the literature and tested in simulation. For tests in the real motor, routines were created to measure the velocity and position using the encoder signals present in the motor assembly, besides control routines made from the discretization of controllers obtained in continuous time, and the motor by the power control done with PWM signal. The experiments performed on one of the motors with the cascade controller were satisfactory and the motor was able to follow a ramp position reference without error.

Keywords: Cascade control, DC motor, encoder, PID, Servomechanism

SUMÁRIO

1 Introdução 1
1.1 Contextualização 1
1.2 Definição do problema 2
1.3 Objetivos do projeto 2
1.4 Apresentação do manuscrito 3

2 Fundamentos 4
2.1 Manipuladores 4
2.2 Rhino XR-4 5
2.2.1 Cinemática 7
2.3 Encoder 8
2.4 Ponte H 9
2.5 Modulação por Largura de Pulso 10
2.6 Plataforma Arduino 10
2.7 Modelo dinâmico de um motor de corrente contínua 11
2.8 Controlador 13
2.8.1 Controlador PID 13
2.8.2 Métodos de sintonização de PID 14

3 Resultados Experimentais 17
3.1 Proposta de Sistema para Controle 17
3.1.1 Motor 18
3.1.2 Excitação dos motores 21
3.1.3 Sentido de rotação do motor 22
3.1.4 Estratégia para o cálculo de velocidade 22
3.2 Estimação dos parâmetros do motor 25
3.3 Controle de velocidade 26
3.4 Controle de posição 29
3.5 Implementação no Arduino 33

4 Conclusões 35
4.1 Perspectivas Futuras 36

referências bibliográficas 37

Anexos **38**

I Cinemática do Rhino XR-4 ..... **39**
I.1 Cinemática Direta 39
I.2 Cinemática Inversa 41

II Programas utilizados **43**
II.1 Programa para controle de posição e velocidade no Arduino 43

LISTA DE FIGURAS

1.1 Rhino XR-4. 2

2.1 Robô SCARA [1]. 5
2.2 Esquema do posicionamento dos motores do Rhino XR-4 7
2.3 Cinemática Direta. 7
2.4 Cinemática Inversa. 8
2.5 Encoder incremental com sensores em quadratura 8
2.6 Flip-flop tipo D para o encoder incremental. 9
2.7 Sinais resultantes do encoder. (a) Sentido positivo. (b) Sentido negativo. 9
2.8 Ponte H 10
2.9 Modulação por largura de pulso em onda quadrada de 5V. 10
2.10 Parâmetros de resposta de sistema de primeira ordem a uma entrada degrau. 12
2.11 Sistema de controle em malha fechada. 13
2.12 Sistema de controle em cascata. 13
2.13 Curva de resposta a entrada degrau. 14

3.1 Sistema para o controle do robô. 17
3.2 Módulo escravo 18
3.3 Imagem do motor do Rhino XR-4. a - eixo de menor velocidade. b - pinos do conector do conjunto motor-*encoder*. c - conector para sensor fim de curso. d - entrada de alimentação do motor. 19
3.4 Conector do conjunto. 19
3.5 Configuração para saídas dos canais do *encoder*. 20
3.6 Módulo de ponte H L298N[2]. 21
3.7 Sentido de rotação adotado. 22
3.8 Diagrama elétrico do circuito de controle e acionamento do motor. 23
3.9 Resolução do temporizador. 23
3.10 Medição do tempo usando o *timer* 2. 24
3.11 Planta em malha aberta. 25
3.12 Gráfico de velocidade angular para entrada degrau de amplitude 100. 25
3.13 Gráfico de velocidade angular versus ciclo de trabalho. 26
3.14 Simulação da planta em malha fechada. 26
3.15 Resposta da planta para entrada degrau unitário em simulação. 27
3.16 Gráfico da resposta em simulação com controlador PI. 27

3.17 Gráfico da velocidade do motor com controlador PI para entrada degrau unitário. 28
3.18 LGR da malha interna com integrador, desconsiderando atraso. 29
3.19 LGR da malha interna com integrador, considerando atraso. 30
3.20 Diagrama de simulação em SIMULINK/MATLAB da malha de controle de posição. . 30
3.21 Gráfico da resposta do controle PID para entrada degrau unitário em simulação. 31
3.22 Gráfico da resposta do controle PID para entrada rampa unitária em simulação. 32
3.23 Resposta do motor a uma referência rampa. 32
3.24 Fluxograma que representa o algoritmo implementado no Arduino. 34

I.1 Representação dos parâmetros D-H 39
I.2 Alocação das coordenadas e parâmetros cinemáticos do Rhino XR-4[3]. 40

LISTA DE TABELAS

2.1 Envelope de trabalho baseados na três primeiros eixos ou juntas[3]. 5
2.2 Redução causada pelas engrenagens. 6
2.3 Configurações do Arduino 11
2.4 Regra de Ziegler-Nichols[4] baseado na resposta ao degrau. 15
2.5 Regra de Ziegler-Nichols baseado no período crítico P_{cr} e ganho crítico K_{cr}. 15
2.6 Regra CHR para controlador PID. 15

3.1 Descrição dos pinos do mólulo de ponte H dupla[5]. 21
3.2 Divisores de frequência de *clock* para o *timer* 2 do Arduino. 24

I.1 Parâmetros Cinemáticos - Algoritmo de Denavit-Hartenberg. 41

LISTA DE SÍMBOLOS

Símbolos Latinos

V	Tensão	[Volts]

Símbolos Gregos

ω	Velocidade angular	[rad/s]
τ	Constante de tempo	[s]
θ, α	Posição angular	[rad]
Ω	Unidade de medida de resistência	

Grupos Adimensionais

i, k	Contador
K	Ganho
c	Cosseno
s	Seno
c_{ik}	Cosseno$(\theta_i + \theta_k)$
s_{ik}	Seno$(\theta_i + \theta_k)$
q	variáveis de junta

Subscritos

ref	referência
f	final
i	inicial
cr	crítico

Sobrescritos

$\cdot$	Variação temporal
$-$	Valor médio

Siglas

PWM	Modulação por largura de pulso
DC	Corrente contínua
CI	Circuito Integrado
ZN	Método Ziegler-Nichols
CHR	Método Chien-Hrones-Reswick
PID	Controlador Proporcional-Integral-Derivativo
PI	Controlador Proporcional-Integrativo
P	Junta prismática
R	Junta de revolução

Capítulo 1

Introdução

Este capítulo propõe os objetivos do projeto realizado e também contextualiza a importância do estudo no cenário atual da tecnologia, além apresentar a organização do trabalho.

1.1 Contextualização

Atualmente a robótica está sendo utilizada amplamente na industria, na agricultura e no entretenimento, o que torna seu estudo de profunda importância. A necessidade do aumento da produtividade para atender a demanda da população mundial, torna necessário processos automatizados que excedem a capacidade humana de trabalho. Assim, os robôs vem para fazer esse trabalho, de forma mais rápida, mais precisa e intensiva.

Logo, com o aumento da robotização da produção, é preciso a formação de engenheiros capazes de projetar máquinas mais robustas e melhores, e isso só é possível com o estudo dos fundamentos mais básicos dos manipuladores.

Uma definição para robôs robôs manipuladores dada por Schilling [3], diz:

> "Um robô é um dispositivo mecânico controlado por *software* que utiliza sensores para guiar uma ou mais ferramentas por trajetórias programadas em um espaço de trabalho a fim de manipular objetos físicos."

Estes robôs podem ser fixos ou móveis, a depender do seu espaço de trabalho. Mas todos possuem uma característica básica em comum, o uso de atuadores, ou motores, que têm por função promover o movimento de suas juntas, ou do robô como um todo no caso de robôs móveis. Estes robôs são compostos por uma série de elos rígidos e juntas móveis, a serem descritos mais profundamente no capítulo 2.

Robôs manipuladores, como o alvo deste estudo, tem como função a manipulação de objetos ou ferramentas no espaço seguindo uma trajetória definida matematicamente, por meio de algoritmos computacionais. Essas trajetórias são traduzidas em movimentos angulares ou lineares das junta

no tempo de forma coordenada a fim de posicionar e orientar seu efetuador final. Robôs de solda e pintura, por exemplo, precisam que sua ferramenta sigam trajetórias específicas, que só pode ser realizado com a atuação precisa nos atuadores de junta.

Logo, o projeto de um servomecanismo capaz de fazer o controle desses atuadores é um parte importante no projeto de um robô, como proposto neste projeto.

1.2 Definição do problema

O robô educacional Rhino-XR4, figura 1.1 é um robô considerado do tipo manipulador. Possui base fixa, 5 juntas rotacionais independentes e uma garra para manipulação de objetos. A juntas são movimentadas por motores elétricos de corrente contínua com redução. Os motores do robô possuem em sua composição um *encoder* incremental no eixo de maior velocidade, que permite a realimentação de posição.

Para o correto acionamento dos motores é necessário um elemento controlador que faça com que o motor siga os valores de posição e/ou velocidade calculados na geração da trajetória desejada para a ferramenta.

Figura 1.1: Rhino XR-4.

Ao longo dos anos, alguns sistemas para o controle das juntas do Rhino foram propostos, de forma a melhorar seu controle ou sua geração de trajetória e aprimorar sua precisão no posicionamento e orientação do efetuador final no espaço.

1.3 Objetivos do projeto

O projeto tem como objetivo desenvolver o sistema para o controle de velocidade e de posição angular dos atuadores de junta. Para que isso seja possível será levantado o modelo dinâmico do motor e, usando terias de controle em malha fechada, será projetado um controlador em cascata

para que o atuador siga um sinal de referência de velocidade e/ou de posição.

Também será feito neste trabalho a criação de rotinas na plataforma Arduino para a implementação física do controle.

1.4 Apresentação do manuscrito

No capitulo 2 será apresentados os conceitos fundamentais para a realização do projeto, como o estudo de modelo dinâmico do motor, controle em malha fechada, especificações fornecidos pelo fabricante, entre outros conceitos considerados relevantes para o estudo. O capítulo 3 traz o desenvolvimento do projeto em si e como cada elemento foi pensado no sistema a ser desenvolvido e os resultados obtidos. O trabalho é fechado no capítulo 4 fazendo a análise dos resultados obtidos, além de propostas de trabalhos para dar continuação ao projeto.

Capítulo 2

Fundamentos

2.1 Manipuladores

Os robôs manipuladores, comumente chamados de braços robóticos possuem várias características que os distinguem em vários critérios como tipos de juntas, tecnologia de acionamento das juntas, espaço de trabalho, número de graus de liberdade, precisão e repetibilidade entre outras.

Algumas características de robôs manipuladores:

- Graus de liberdade: equivale ao número de juntas do robô manipulador. Os três eixos iniciais são responsáveis pela posição do pulso do robô no espaço e os eixos restantes são responsáveis pela orientação da ferramenta[3].
- Tipo de junta: Podem ser juntas de revolução (R) ou prismática (P) em sua maioria. A primeira produz deslocamento angular entre dois elos e a segunda deslocamento linear.
- Espaço de trabalho: é o volume total que o o efetuador final é capaz de atingir no espaço. Este é limitado pelas restrições mecânicas do manipulador e de suas articulações.
- Geometria do espaço de trabalho: baseado nas três juntas iniciais do robô temos algumas geometrias básicas que configuram a geometria do espaço de trabalho, que podem ser vistas na tabela 2.1

Os robôs podem possuir as mais variadas configuração a depender de seu uso. Ele pode ser um robô cartesiano de juntas prismáticas, que produz deslocamentos nos eixos cartesianos, como uma impressora 3D. Um robô SCARA possui a geometria de seu espaço de trabalho semelhante a de um robô cilíndrico e possui articulações paralelas, como pode ser visto na figura 2.1, muito usado em trabalhos do tipo *pick-and-place*. Um robô articulado, por sua vez, possui todas as juntas de revolução a fim de simular as articulações do braço humano, criando um espaço de trabalho mais complexo.

Exitem dois modos básicos de movimentação da ferramenta, que pode ser uma movimentação ponto a ponto, no qual a ferramenta se move de um ponto a outro no espaço porém sem uma

Tabela 2.1: Envelope de trabalho baseados na três primeiros eixos ou juntas[3].

Robô	Eixo 1	Eixo 2	Eixo 3	Número de jutas de revolução
Cartesiano	P	P	P	0
Cilíndrico	R	P	P	1
Esférico	R	R	P	2
SCARA	R	R	P	2
Articulado	R	R	R	3

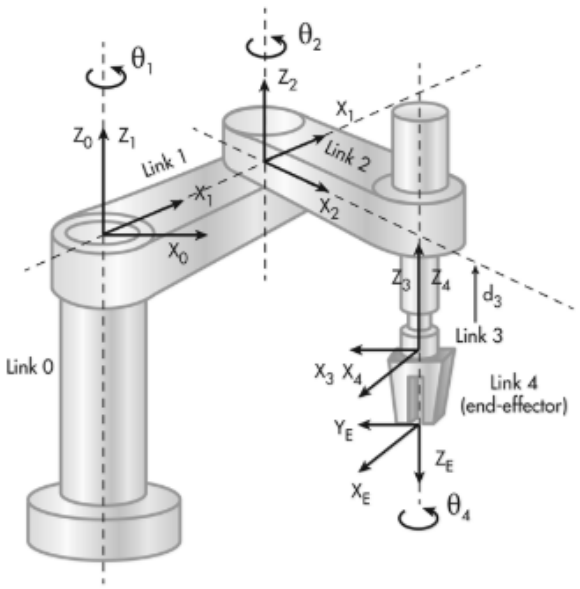

Figura 2.1: Robô SCARA [1].

trajetória definida. Este método é útil para trabalhos de carregamento ou reposicionamento de objetos em linhas de montagem, por exemplo. Outro método, utilizado para trabalhos como de pintura ou solda, exige que a trajetória bem definida no espaço tridimensional e a velocidade no trajeto também deve ser controlado.

2.2 Rhino XR-4

O Robô Rhino XR4, Figura 1.1, é um robô manipulador educacional com 5 graus de liberdade, com 5 juntas de rotação e ainda possui uma garra para manipulação de objetos. Conta com seis motores de corrente contínua com redução e acoplamentos por meio de correntes, além de *encoders* para a leitura da posição. A disposição dos motores pode ser vista na figura 2.2.

Especificações fornecidas pelo fabricante:

- Carga máxima: 2Kg. É a carga máxima que o robô é capaz de movimentar.

- Repetibilidade: 1.006mm. É a medida da habilidade do braço robótico de posicionar a ferramenta no mesmo lugar repetidas vezes. Em razão de limitações mecânicas, causada pelas correntes e atritos há um erro associado.
- Peso do robô: 7Kg.
- Envelope de trabalho. Limites de rotação das juntas:
  - Base: 360^o
  - Ombro: 250^o
  - Cotovelo: 270^o
  - Pulso: 270^o
  - Rolagem: infinita
- Motores:
  - Tensão de alimentação: até 12V DC.
  - 4 x Pittman GM9413K046-R1 (base, ombro, cotovelo e pulso) com redução de 65.5:1.
  - 2 x GM8712E465 (rolagem e garra) com redução de 96:1.
- *Encoder* incremental de 6 faixas com dois sensores em quadratura para sentido de giro do motor.
- Sensor fim de curso: Para cada uma das juntas existe um sensor fim de curso normalmente aberto que pode ser usado para calibração de posição.

Apesar do pequeno número de faixas do *encoder*, com a redução dos motores aumentamos a resolução. Originalmente temos 24 possíveis valores de contagem, porém com a redução nos motores este número passa para 1572 contagens para cada rotação do eixo de menor velocidade.

Além da redução nos motores do eixo de maior velocidade para o de menor velocidade de 1:65.5, temos que considerar também a redução causada pelas rodas dentadas e correntes que ligam a junta do robô ao motor que o aciona. A redução provocadas pelas engrenagens são dadas na tabela 2.2.

Tabela 2.2: Redução causada pelas engrenagens.

Junta	Redução
Base	0.25
Ombro	0.125
Cotovelo	0.125
Pulso	0.125
Rolagem	0.25

A excitação dos motores será feita por meio de sinal PWM - Pulse Width Modulation ou modulação por largura de pulso. O sinal é uma onda retangular que tem sua largura de pulso controlada de forma que a potência média seja a desejada para a rotação do motor.

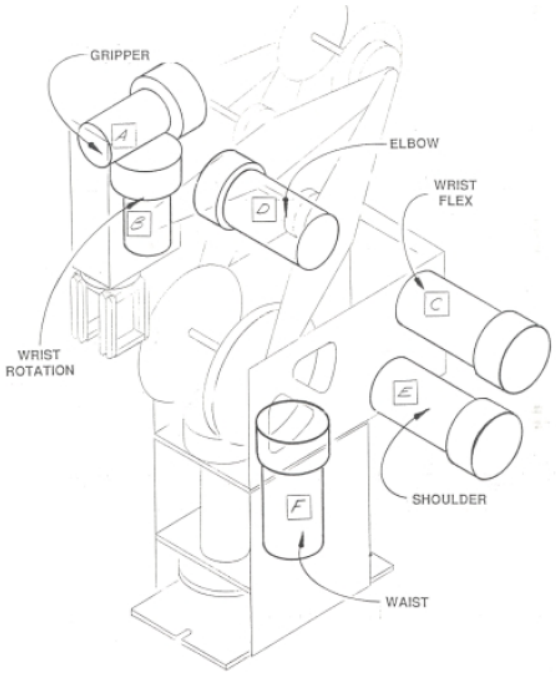

Figura 2.2: Esquema do posicionamento dos motores do Rhino XR-4

2.2.1 Cinemática

A cinemática direta consiste em, tendo conhecimento das variáveis de junta q, obter a posição p e orientação R da ferramenta ou efetuador final [3], como indicado na figura 2.3. Essas variáveis são o ângulo de rotação θ_i em caso de juntas de rotação ou revolução, ou deslocamentos d_i, no caso de juntas prismáticas.

Figura 2.3: Cinemática Direta.

Um método usado para determinar a cinemática direta é o método de Denavit-Hartenberg (D-H), que propõe a representação da posição e da orientação de uma junta em relação a junta anterior, de forma que em robôs seriais, algumas transformações simples usando matrizes homogêneas, sejam capazes de representar a posição da ferramenta em relação às coordenadas da base fixa. O método é comumente utilizado na robótica e reduz a quantidade de parâmetros necessário para a modelagem.

Cada uma das juntas é representada por uma série de transformações básicas de rotação e translação em relação ao frame anterior [6].

A cinemática inversa consiste em dadas a posição e a orientação da ferramenta, calcular as variáveis de junta correspondentes, e poder usar esses valores para a atuação nas juntas do robô.

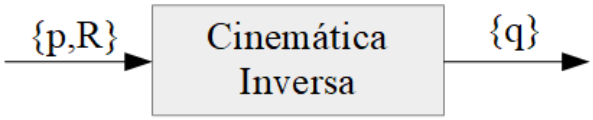

Figura 2.4: Cinemática Inversa.

O problema da cinemática inversa é mais complexo, pois não há uma fórmula que valha para todos os robôs, cada caso deve ser tratado de forma única. Outra dificuldade é achar soluções únicas, dados que o numero de equações geralmente é maior que o numero de incógnitas.

Além disso, a configuração e posição da ferramenta deve ser avaliada anteriormente para verificar sua validade. Apesar das limitações do modelo inverso, é possível usá-lo para fazer a ferramenta seguir uma trajetória definida no espaço, calculando as variáveis de junta para cada ponto da trajetória.

A cinemática direta e inversa para o robô estudado pode ser encontrado no Anexo I.

2.3 Encoder

O *encoder* é um dispositivo eletromecânico usado para determinar a posição angular ou linear de um eixo rotativo. Ele consiste em um disco ou fita com marcações e um componente emissor e receptor de luz.

As faixas tem a função de permitir ou não a passagem ou reflexão da luz emitida pelo sensor óptico. Assim, quando o receptor recebe o sinal enviado ele produz uma onda correspondente. A configuração de um *encoder* incremental pode ser vista na figura 2.5.

Figura 2.5: Encoder incremental com sensores em quadratura

Para a medida da resolução do *encoder* incremental devemos dividir o deslocamento angular de um volta completa do eixo pelo número de faixas, N, como mostrado da equação 2.1.

$$Resolução = \frac{2\pi}{N} \tag{2.1}$$

Para um eixo que gire em ambas as direções também é necessário ter um fotoemissor e receptor em quadratura, para que seja possível determinar o sentido de rotação do eixo. Dessa forma os

sinais obtidos terão uma defasagem de $+90^o$ ou -90^o.

Para determinar o sentido de rotação do motor pode ser usado um flip-flop D, que tem como função atualizar a saída Q, com o valor de entrada D a cada borda de subida do sinal de clock, figura 2.6. O primeiro canal do encoder é usado como clock e o segundo como entrada D. Dessa forma o sinal de saída será 1 quando em sentido horário e 0 quando em sentido negativo ou anti-horário, como mostrado na figura 2.7.

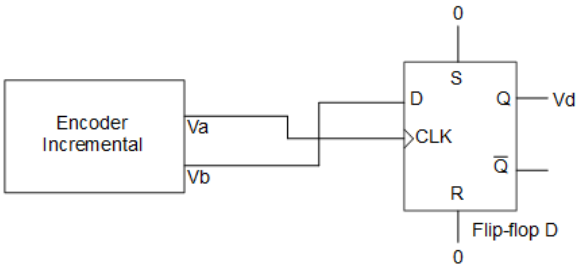

Figura 2.6: Flip-flop tipo D para o encoder incremental.

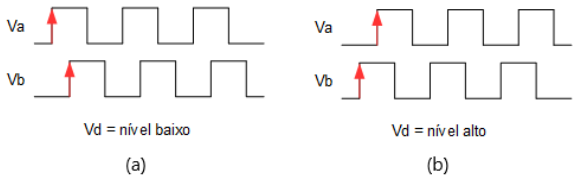

Figura 2.7: Sinais resultantes do encoder. (a) Sentido positivo. (b) Sentido negativo.

2.4 Ponte H

Para que seja possível acionar um motor em ambas as direções de rotação, é necessário uma cofiguração em ponte H, que consiste num conjunto de chaves acionadas para que a corrente flua em determinado sentido pelo motor. Essas chaves são feitas usando transistores, como pode ser visto na figura 2.8.

Essas chaves devem ser acionadas na ordem correta para indicar sentido de giro desejado no motor. O sinal IN1 devem estar em nível quando o sinal IN2 for baixo e vice-versa. Para parar o motor é necessário que os sinais IN1 e IN2 sejam zero, dessa forma não haverá passagem de corrente em nenhum sentido no motor. Os diodos presentes na ponte H tem a finalidade de dissipar a energia armazenada no campo magnético devido à armadura.

Os transistores apresentam uma queda de tensão que faz com que a tensão aplicada no motor seja menor que Vin. Para que se tenha uma tensão igual a máxima aplicada no motor é necessário uma fonte de alimentação que supra essa queda provocada pelos elementos da ponte.

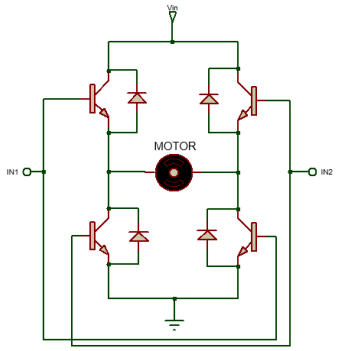

Figura 2.8: Ponte H

2.5 Modulação por Largura de Pulso

A modulação por largura de pulso, comumente chamada de PWM, sigla para *Pulse Width Modulation*, consiste na regulação da tensão por meio de uma sinal de onda cíclica. No caso de motores ela é utilizada para transferir potência com menor perda de eficiência, como no caso de um controle linear de potência. No controle linear, em que a potência no motor é controlada por meio de uma carga resistiva, há perdas de potência na forma de calor, diminuindo a eficiência no motor.

No caso do controle feito pelo chaveamento de transistores, esta perda diminui significativamente, já que não exige carga resistiva, aumentando assim a eficiência. A saída resultante da tensão é a média do sinal. No caso de uma onda quadrada, o ciclo de trabalho, ou a largura de onda, dado em porcentagem é equivalente ao valor de tensão máxima da onda, como pode ser visto na imagem 2.9.

A excitação na forma PWM só é possível de ser usada porque o motor funciona como um filtro passa-baixas, cuja frequência de corte é muito menor que a frequência do PWM.

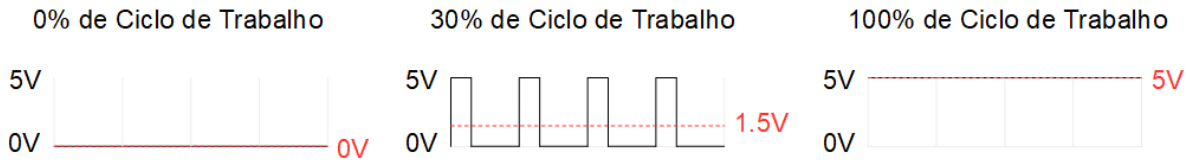

Figura 2.9: Modulação por largura de pulso em onda quadrada de 5V.

2.6 Plataforma Arduino

O Arduino é uma plataforma de código aberto programável que possui pinos de entrada e saída digitais e analógicas que permite a leitura de sensores e acionamento de atuadores.

Um dos microcontroladores da plataforma é o ATmega328 de 8bits baseado em AVR capaz de

executar instruções em um único período de *clock*. Além de contar com dois temporizadores de 8bits cada e um temporizador de 16bits com registradores de *overflow*, comparadores e divisores separados.

Além disso, conta com interfaces seriais compatíveis com redes SPI, I2C e UART, pinos de interrupção externa e interna, seis canais PWM, entre outros.

A plataforma é programada em linguagem C e possui uma infinidade de utilidades. Sua capacidade de processamento permite a utilização em sistemas para controle em tempo real e comunicação com periféricos por meio de protocolos de comunicação serial.

A plataforma permite a geração de sinais PWM para o controle de velocidades de motor. O PWM gerado pelo Arduino possui um período de 2ms ou uma frequência de 500Hz. Como o PWM é codificado em 8bits a faixa de 0% a 100% de ciclo de trabalho é mapeado nos valores de 0 a 255.

As principais características do Arduino fornecidas pelo fabricante [7] podem ser vistas na tabela 2.6.

Tabela 2.3: Configurações do Arduino

Característica	**ATmega328/P**
Frequência de Clock	16MHz
Tensão de trabalho	5V
Alimentação	7V - 12V
Pinos	28/32
Canais PWM	6
Memória Flash (bytes)	32K
SRAM (bytes)	2K
EEPROM (bytes)	1K
SPI	2
I2C	1
USART	1
Temporizador de 8bits	2
Temporizador de 16bits	1
Pinos de interrupção externa	2 e 3

Para uma resposta mais rápida do sistema também é possível a manipulação dos registradores das portas do microcontrolador, que permite leitura e escrita em vários pinos ao mesmo tempo.

2.7 Modelo dinâmico de um motor de corrente contínua

Um motor de corrente contínua quando submetido a uma entrada degrau unitário em malha aberta tem uma reposta equivalente a de um sistema de primeira ordem do tipo 0, ou seja, que não contem pólo na origem, além de um atraso, expresso na equação 2.2 [8].

$$\frac{\omega(s)}{V(s)} = \frac{K_m e^{-Ls}}{\tau_m s + 1} \tag{2.2}$$

Os motores possuem uma região inicial de zona morta, que ocorrem devido ao atrito inerente a sistemas mecânicos, que reflete no atraso inicial da resposta, dada por L em segundos. Um motor necessita de uma tensão mínima para vencer a inercia e iniciar sua rotação e depois de um tempo atinge uma velocidade angular $\omega(t)$ para a tensão aplicada $V(t)$.

Para estimar os parâmetros do modelo dinâmico de um motor, podemos aplicar uma entrada degrau de determinada amplitude e verificar sua resposta no tempo. O atraso L é dado pelo tempo, em segundos, que a resposta permanece em zero no início.

O teorema do valor final é uma ferramenta usada para obter o valor de x(t) quando t tende a infinito quando se conhece X(s). Este se dá pelo valor da resposta a entrada com determinado perfil quando s tende a zero. Para uma entrada degrau de amplitude A, na forma da equação 2.3, o valor final é dado pelo ganho da função de transferência e do ganho da entrada, como visto na equação 2.5 [8].

$$V(s) = \frac{A}{s} \tag{2.3}$$

$$\omega(s) = \frac{K_m}{(\tau_m s + 1)} \frac{A}{s} \tag{2.4}$$

$$\omega_f(\infty) = \lim_{x \to 0} s \frac{K_m}{(\tau_m s + 1)} \frac{1}{s} = K_m A \tag{2.5}$$

A constante de tempo τ_m pode ser obtida a partir do gráfico da resposta a entrada degrau, no momento em que a saída atinge 63% do valor final[8], e o ganho K_m é dado pela razão entre o valor final $\omega_f(\infty)$ e a amplitude A do degrau aplicado.

Os parâmetros do modelo são representados na figura 2.10, onde Vf é o valor final da resposta, Vref a entrada de referência e Tm a constante de tempo do sistema.

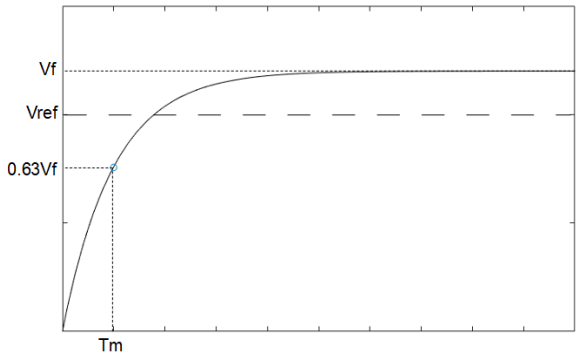

Figura 2.10: Parâmetros de resposta de sistema de primeira ordem a uma entrada degrau.

2.8 Controlador

Para sistemas que precisam de precisão na resposta, torna-se necessário o uso de um controlador em uma malha com realimentação do valor de saída, de forma que esta siga o valor desejado que é dado como referência na entrada do sistema. O controlador tem como função ajustar o valor do erro para que a planta controlada seja capaz de seguir sem erro o valor desejado.

Para um sistema de primeira ordem, ou do tipo zero, para que o erro seja nulo em regime permanente ao se aplicar uma entrada constante é necessário que o controlador tenha ação integral. Por esse motivo, um controlador na forma PI ou PID pode ser usado para a aplicação.

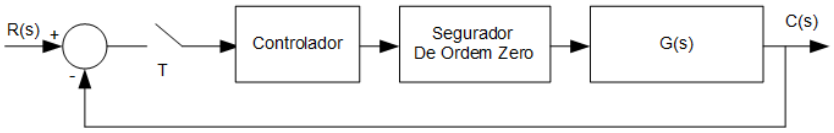

Figura 2.11: Sistema de controle em malha fechada.

Para um controle mais adequado de posição angular de um motor, pode ser usado dois controladores em cascata. Assim, um controlador é responsável pelo controle de velocidade em uma malha interna de controle, e outro controlador realiza o controle da posição em uma malha externa, gerando referências de velocidade para a malha interna. A malha em cascata pode ser vista na figura 2.12.

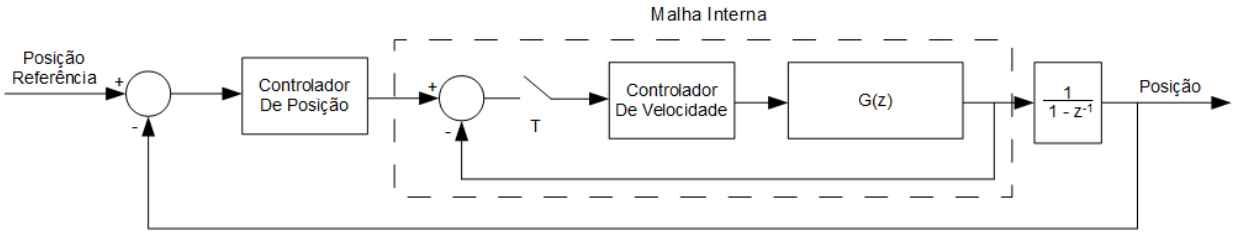

Figura 2.12: Sistema de controle em cascata.

2.8.1 Controlador PID

O controlador PID possui três termos, um proporcional, um integrativo e um derivativo como pode ser visto na equação 2.6 [4], onde K_p é a constante proporcional, T_i o tempo integral e T_d o tempo derivativo.

$$G_c(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \tag{2.6}$$

A constante proporcional, K_p tem como princípio fazer com que a saída do sistema se aproxime do valor de referência, porém, controlar uma planta apenas com um controlador proporcional pode

fazer com que haja um grande sobrepasso, e também um erro em regime permanente, no caso de uma planta do tipo zero.

Ao adicionar um termo integrativo no controle, anula-se o erro em regime permanente ao adicionar no sistema um polo na origem do plano s, porém sacrifica a estabilidade do sistema e aumenta o sobrepasso, por diminuir o amortecimento.

O termo derivativo tem o objetivo de diminuir o sobrepasso e é proporcional a taxa de variação do erro, quanto maior a variação inicial do erro na resposta transitória, mais rápida é a resposta do controlador.

2.8.2 Métodos de sintonização de PID

Existem várias formas proposta na literatura para o cálculos destes coeficiente do controlador PID. Um método muito utilizado para o controle de plantas que possuem um atraso L na resposta, como expresso na equação 2.7, é o método Ziegler-Nichols [4], que busca obter o controle com um sobrepasso máximo de 25%.

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{2.7}$$

O método consiste em estimar os coeficiente K_p, T_i e T_d a partir da resposta transitória da planta ao ser submetida a uma entrada degrau unitário em malha aberta. Na figura 2.13, podemos ver os parâmetros do transitório.

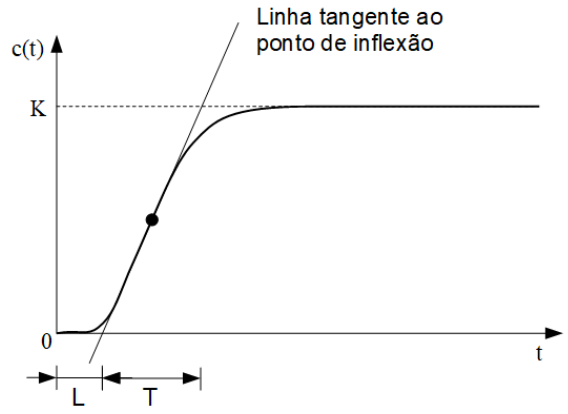

Figura 2.13: Curva de resposta a entrada degrau.

Os coeficientes do controlador podem ser calculados seguindo os valores da tabela 2.8.2.

Para plantas que possuam um integrador, o método anterior não pode ser utilizado. Para isso podemos usar um segundo método proposto por Zieglel-Nichols, que usa a resposta transitória em malha fechada. Primeiro, deve-se usar apenas o termo proporcional do controlador e variar até que a resposta final tenha uma oscilação de amplitude constante e período P_{cr} denominamos esse ganho proporcional de ganho crítico, representado por K_{cr}.

Tabela 2.4: Regra de Ziegler-Nichols[4] baseado na resposta ao degrau.

Tipo de Controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	2L	$0.5L$

Para obter o controlador a partir deve-se usar a tabela 2.8.2.

Tabela 2.5: Regra de Ziegler-Nichols baseado no período crítico P_{cr} e ganho crítico K_{cr}.

Tipo de Controlador	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Outro método proposto por Chien-Hrones-Reswick (CHR)[9], também chamado de ZN modificado, usa os mesmos parâmetros do método Ziegler-Nichols, e acrescenta outro, o parâmentro a, que é o ponto de cruzamento da reta tangente ao ponto de inclinação, vide figura 2.13, e o eixo c(t), que pode ser calculado pela equação 2.8. O método proporciona uma fórmula para o controlador para 0% e 20% de sobrepasso, que pode ser visto na tabela 2.6.

$$a = \frac{KL}{T} \tag{2.8}$$

Tabela 2.6: Regra CHR para controlador PID.

Tipo de Controlador	0% de Sobrepasso			20% de Sobrepasso		
	K_p	T_i	T_d	K_p	T_i	T_d
P	$\frac{0.3}{a}$	∞	0	$\frac{0.7}{a}$	∞	0
PI	$\frac{0.35}{a}$	$1.2T$	0	$\frac{0.6}{a}$	T	0
PID	$\frac{0.6}{a}$	T	$0.5L$	$\frac{0.95}{a}$	$1.4T$	$0.47L$

Para plantas mais complexas com pequena margem de estabilidade, também pode ser considerado a sintonização manual dos parâmetros PID [10]. Para isso, inicialmente aplicamos apenas o controle proporcional até que a resposta da planta comece a oscilar com uma amplitude contante. O valor de Kp é dado por metade deste valor obtido. Após a regulação do ganho, ajustamos o tempo de integração T_i de forma a diminuir o tempo de resposta da planta e eliminar o erro em regime permanente, caso exista. Por último, deve-se ajustar o tempo derivativo T_d para diminuir o sobrepasso até valor desejado.

O controlador PID ideal possui mais zeros que pólos, o que o torna não causal e portanto não pode ser implementado na prática. Para torná-lo causal, o termo derivativo é modificado de

forma a se tornar um filtro derivativo com uma constante de muito menor que T_d de forma a não comprometer a ação derivativa, o controlador PID causal é expresso na equação 2.9.

$$G_c(s) = K_p \left(1 + \frac{1}{T_i s} + \frac{T_d s}{\alpha T_d s + 1}\right) \tag{2.9}$$

Os controladores obtidos pelos métodos propostos podem ser usado como base para o refinamento do controlador real, empiricamente. Quanto à implantação em algoritmos computacionais, devemos usar a forma discretizada do controlador PID e obter sua equação de diferenças. Antes de tudo obtemos a função de transferência do controlador, apresentada na equação 2.10.

$$G_c(s) = K_p T_d \frac{s^2 + \frac{1}{T_d} s + \frac{1}{T_i T_d}}{s} \tag{2.10}$$

Aplicando a transformação bilinear, que consiste no método de integração trapezoidal[11], apresentada na equação 2.11, na função de transferência do controlador obtemos sua forma discreta e também a equação de recorrência que pode ser implementada em um controle digital.

$$G(z) = |G(s)|_{s=\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}} \tag{2.11}$$

Capítulo 3

Resultados Experimentais

3.1 Proposta de Sistema para Controle

Foi proposto um novo sistema de controle do robô manipulador, baseado em Arduíno na forma mestre-escravo representado na figura 3.1.

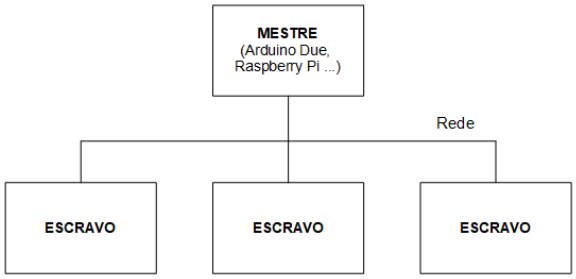

Figura 3.1: Sistema para o controle do robô.

O mestre poderá ser um Arduino Due, uma Raspberry Pi, ou qualquer outro computador ou controlador que possuam uma maior capacidade de processamento desde que utilize o protocolo adequado de comunicação com os módulos, denominados escravos. A interface para a comunicação entre mestre e escravos pode ser RS485, I2C, SPI, ou outra compatível com o sistema.

Funções do mestre:

- Geração de trajetória a ser seguida pela ferramenta.
- Cálculo da cinemática Inversa gerando os parâmetros de referência de posição angular das juntas.
- Cálculo dos reais valores de deslocamento angular dos motores considerando a redução provocadas pelas engrenagens.
- Armazenamento em memória não volátil da posição angular de cada motor a serem enviados pelos módulos escravos, e enviados a eles no inicio.

- Rotinas de envio de parâmetros de referência por meio da rede a serem seguidos pelos módulos escravos.
- Rotinas para calibração de posição inicial do robô
- Outras rotinas a serem definidas conforme necessidade.

Cada um dos módulos escravos, figura 3.2, será composto por um Arduino com microcontrolador ATmega328P (UNO, nano ou micro) e um driver de ponte H dupla L298N a ser descrito em seção subsequente, que suporta dois motores.

Funções do escravo:

- Fazer o controle de velocidade dos motores.
- Fazer o controle de posição dos motores.
- Realizar o cálculo de velocidade e deslocamento angular.
- Receber parâmetros de velocidade ou posição do mestre pela rede.
- Enviar valores de posição angular ao mestre a cada período de tempo definido.
- Fazer o acionamento dos motores no sentido desejado por meio do driver L298N.

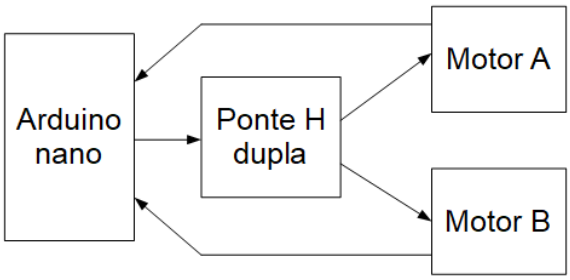

Figura 3.2: Módulo escravo

3.1.1 Motor

O motor a ser controlado é mostrado na figura 3.3. Foi decidido usar para a medida de deslocamento angular apenas as transições do canal A do *encoder* e o canal B foi usado para determinar o sentido de rotação. Com isso, a relação deslocamento angular por contagem, ou $rad/counts$, no eixo de maior velocidade é dada pela equação 3.1, onde N é o número de faixas do *encoder* incremental, resultando em uma relação de $0.5236 rad/count$ ou 30^o por transição. Considerando a redução, a relação no eixo de menor velocidade passa a ser $0.007994 rad/count$ ou 0.4580^o por transição.

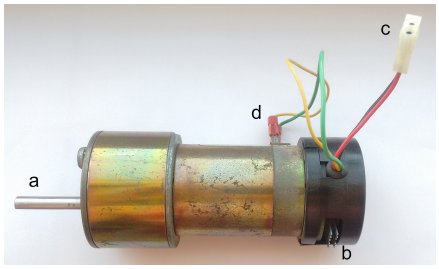

Figura 3.3: Imagem do motor do Rhino XR-4. a - eixo de menor velocidade. b - pinos do conector do conjunto motor-*encoder*. c - conector para sensor fim de curso. d - entrada de alimentação do motor.

$$rad/count = \frac{2\pi}{2N} = \frac{2\pi}{12} = 0.5236 rad/count \tag{3.1}$$

O conjunto motor-*encoder* possui em seu circuito um conector com 10 pinos para alimentação do motor e obtenção dos sinais de *encoder* e do sensor de fim de curso como mostrado na Figura 3.4.

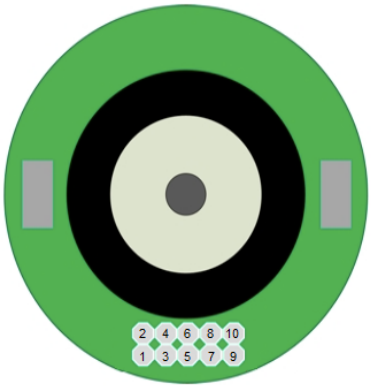

Figura 3.4: Conector do conjunto.

Identificação do conector do conjunto:

1 - Terra do circuito.

2 - Canal A do encoder.

3 - Alimentação 5 volts do circuito.

4 - Canal B do encoder.

5 - Não conectado.

6 - Canal do sensor fim de curso.

7 e 8 - Entrada positiva do motor.

9 e 10 - Entrada negativa do motor.

Os canais A e B do encoder são os sensores em quadratura, e enviam sinal alto ou baixo na

leitura do disco para um transistor na saída do circuito. Para obter uma onda quadrada da saída dos canais do encoder é necessário um resistor *pull-up*, como mostrado na figura 3.5.

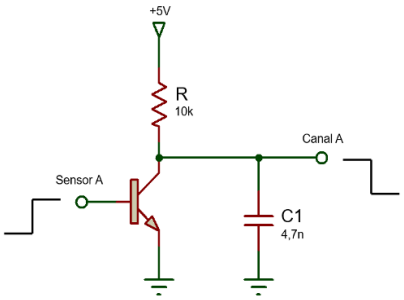

Figura 3.5: Configuração para saídas dos canais do *encoder*.

Esse transistor é o estágio de saída de um CI comparador. Quando o sinal enviado pelo *encoder* é alto ele habilita a passagem de corrente e saída do canal será 0V, quando o sinal do sensor do *encoder* é baixo o transistor não permite passagem de corrente e a saída do canal do *encoder* será 5V.

Assim como o *encoder*, a leitura do canal do sensor de fim de curso é necessário um resistor *pull-up* na saída para verificação do sinal, os resistores R_A3 e R_B3 para o sensor fim de curso para cada motor são mostrados no diagrama elétrico do servomecanismo na figura 3.8.

Para melhorar o sinal de saída do canal foi usado um filtro para evitar possíveis ruídos no sinal que pudessem causar erro nas medições. Um filtro com um tempo de resposta adequado foi calculado levando em consideração a pior situação de velocidade do motor, ou seja, quando em velocidade máxima de rotação, em que o período do sinal é de cerca de 5ms. Para isso a exigência é que o filtro siga a tensão em 10% do tempo na transição de um estado para outro. Essa transição ocorre a cada 2,5ms no pior caso. Então o para um tempo de resposta $0.25ms$ e considerando que esse tempo equivale a 5 constantes de tempo do sistema RC, representado pela equação 3.2, a contante de tempo do sistema será de 50μs, como expresso na equação 3.4.

$$G(s) = \frac{1}{RCs + 1} \tag{3.2}$$

$$\tau = RC \tag{3.3}$$

$$\tau = \frac{0.1 * 2.5ms}{5} = 5 * 10^{-5}s \tag{3.4}$$

Com o valor de R igual a $10k\Omega$ definido arbitrariamente, podemos calcular o valor do capacitor como mostrado na equação 3.5, substituindo os valores de R e τ na equação 3.3.

$$C = \frac{\tau}{R} = \frac{5 * 10^{-5}}{10 * 10^{3}} = 5nF \tag{3.5}$$

Para se ajustar ao valor de mercado de um capacitor, foi escolhido um capacitor de $4.7nF$ para ser implementado no circuito.

3.1.2 Excitação dos motores

Para a excitação dos motores nos dois sentidos de rotação, é necessário o uso de uma ponte H, como mencionado na seção 2. Para isso foi utilizado o módulo L298N que tem capacidade para o acionamento de dois motores com até dois ampéres. Na figura 3.6 podemos ver as partes do módulo, o circuito integrado L298 possui em seu circuito os transistores responsáveis pelo chaveamento na ponte H. O chaveamento é feito pelos sinais recebidos pelas entradas IN1 a IN4. As entradas ENA e ENB controlam a velocidade do motor caso sejam sinal PWM, ou apenas habilitam a alimentação com 12V DC. O chaveamento é limitado a baixas frequência por ter diodos de retificação comum de rede elétrica. A descrição dos pinos pode ser visto na tabela 3.1.

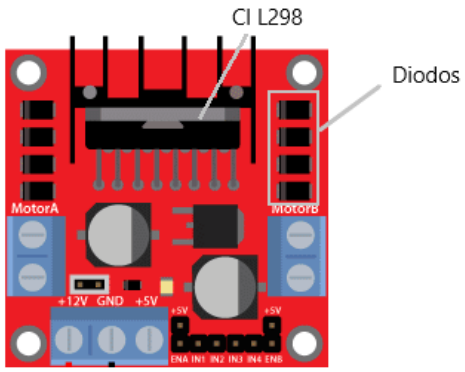

Figura 3.6: Módulo de ponte H L298N[2].

Tabela 3.1: Descrição dos pinos do mólulo de ponte H dupla[5].

Nome da porta	Tipo	Descrição
+12V	Entrada	Alimentação para motores
GND	Entrada	Referência para alimentação
+5V	Entrada	Alimentaçao para CI
IN1 e IN2	Entrada	Controle de direção do Motor A
IN3 e IN4	Entrada	Controle de direção do Motor B
Motor A	Saída	Saída para motor A
Motor B	Saída	Saída para motor B
ENA e ENB	Entrada	Controle de velocidade do motor.

3.1.3 Sentido de rotação do motor

Para determinar o sentido de giro foi usada a leitura direta dos sinais dos canais do *encoder* pelo Arduino. O canal A foi usado para gerar interrupções externas no Arduino a cada mudança de estado, para incrementar ou decrementar o contador de pulsos e gerar a posição. A cada interrupção também é lido o valor no canal B.

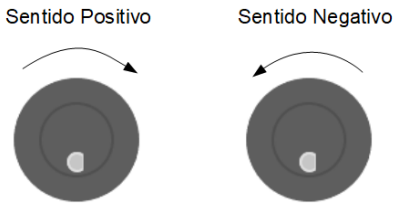

Figura 3.7: Sentido de rotação adotado.

Analisando a figura 2.5, nota-se que no sentido positivo, ao haver uma transição no estado de A o valor do sinal B é sempre diferente do novo estado de A, e quando em sentido negativo, é sempre igual. Essa lógica foi implementada para definir o sentido de giro de motor. Ou seja, sempre que há uma transição de A e o valor de B no momento é diferente, então o giro foi no sentido horário. Caso os valores sejam iguais, então a rotação foi no sentido anti-horário. Os sentidos adotados podem ser visto na figura 3.7.

3.1.4 Estratégia para o cálculo de velocidade

Para fazer o controle de velocidade e posição dos motores é necessário calcular a velocidade dos motores de forma precisa. Para isso foi necessário uma estratégia para a contagem do tempo e de transições do *encoder* no Arduino.

Para o cálculo do tempo forram utilizado os registradores do *Timer* 2 no Arduino.

O *Timer* 2 é um contador de 8 bits que muda de estado a cada mudança no *clock* do micro-controlador ATmega328. O *clock* do Arduino possui uma frequência de 16MHz, e originalmente o contador teria *overflow* a cada período de 15.9μs. Levando em consideração o fato que no pior caso o sinal do *encoder* o período é de $5ms$, e para ter um erro máximo de 1% no cálculo desse período temos que a resolução do temporizador pode ser de 50μs. Dessa forma e diminuimos o número de interrupções durante o processo de cada *overflow*. A representação do temporizador e do sinal do *encoder* por ser vista na figura 3.9.

O temporizador 2 do Arduino possui um registrador, TCCR2B, que deve ser configurado de forma a dividir a frequência máxima por determinados valores. Na tabela 3.2, vemos os valores de período, frequência, valor do registrador e tempo para o *overflow* do temporizador.

Para que o erro fique abaixo de 1% foi escolhido um divisor igual a 256. O temporizador possui ainda outros registradores. O registrador TCNT2 é o contador em si, possui 8 bit e a cada período estabelecido pelo divisor de frequência escolhido ele incrementa 1 no registrador. Um outro

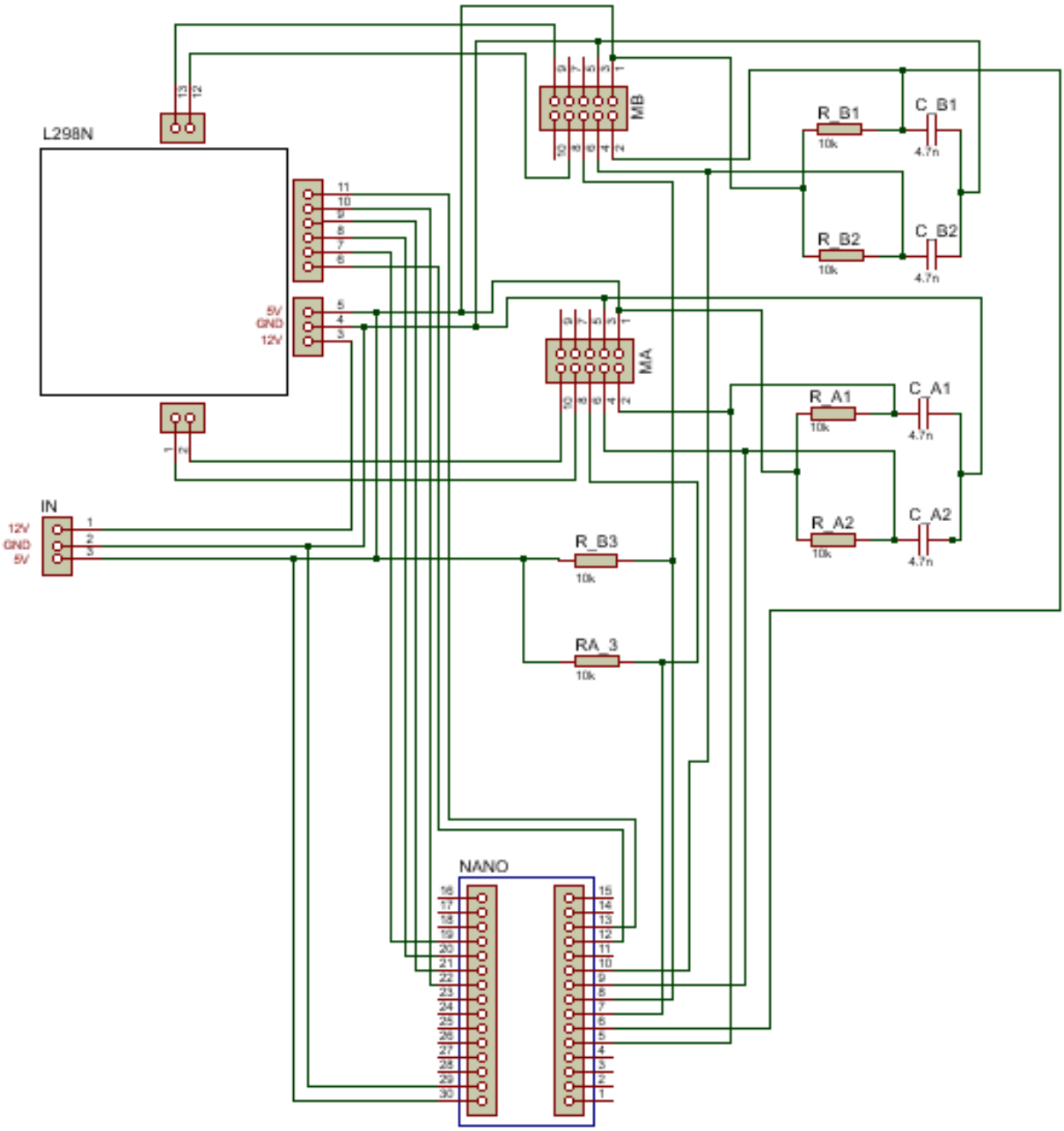

Figura 3.8: Diagrama elétrico do circuito de controle e acionamento do motor.

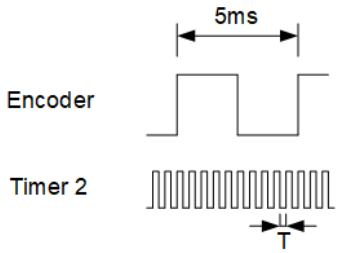

Figura 3.9: Resolução do temporizador.

Tabela 3.2: Divisores de frequência de *clock* para o *timer* 2 do Arduino.

Divisor	Valor do registrador TCCR2B (Hex)	Frequência (Hz)	Período (μs)	Tempo de estouro do contador (ms)
1	0x01	16M	0.0625	0.01594
8	0x02	2M	0.5	0.1275
32	0x03	500k	2	0.51
64	0x04	250k	4	1.02
128	0x05	125k	8	2.04
256	0x06	62.5k	16	4.08
1024	0x07	15.625k	64	16.32

registrador TIMSK2 habilita uma interrupção do sistema a cada estouro do temporizador. Essa interrupção será usada para contar o número de *overflows* ocorridos entre cada leitura do *encoder*. Assim o cálculo do tempo é feito com base na quantidade de *overflows* ocorridos desde a ultima transição e no valor atual e anterior do contador TCNT2. A figura 3.10 mostra como funciona a estratégia para o calculo do tempo e a equação 3.6 mostra o calculo do tempo.

Dessa forma o tempo pode ser calculado tanto para as transições de estado do motor A, do motor B e do período de amostragem utilizando o mesmo temporizador.

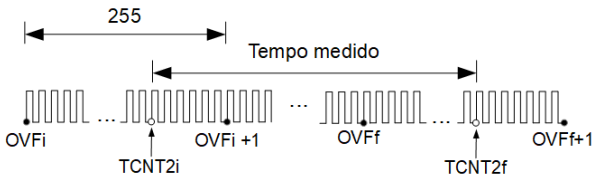

Figura 3.10: Medição do tempo usando o *timer* 2.

$$T = ((OVF_f - OVFi) * 255 + TCNT2_f - TCNT2_i) * 16\mu s \tag{3.6}$$

Foram propostos dois modos de cálculo da velocidade. O primeiro consistia em contar a quantidade de transições dentro do período de amostragem e dividir este valor pelo tempo, porém, com um período de amostragem igual a 10ms, como definido, e o pior caso sendo que o menor período do sinal seria de 5ms o método foi descartado, pois haveria um grande erro na medida da velocidade. O segundo, e também o método escolhido foi medir a velocidade a cada leitura de transição do sinal do *encoder*. E a velocidade é dada pela razão entre o deslocamento angular para cada transição do *encoder* no eixo de menor velocidade e o tempo medido entre as transições.

3.2 Estimação dos parâmetros do motor

Para modelar o motor foi enviado um sinal PWM para o motor usando o Arduino e o módulo de ponte H L298N em malha aberta, como mostrado na figura 3.11 de forma a obter um determinado valor de velocidade na saída calculado pelo algoritmo de medida de velocidade no Arduino a uma taxa de amostragem de 10ms. A taxa de amostragem foi escolhida de forma a se ter 20 amostras até que a resposta da planta em malha aberta atingisse o valor de referência, que ocorre em t igual a 200ms, aproximadamente.

Com isso foi obtido um gráfico do comportamento do motor para uma entrada degrau com amplitude 100 ou 40% de ciclo de trabalho, que pode ser visto na figura 3.12. Este ciclo de trabalho foi escolhido para a modelagem por estar na faixa de valores em que a resposta é linear.

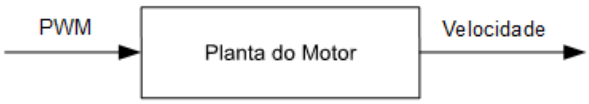

Figura 3.11: Planta em malha aberta.

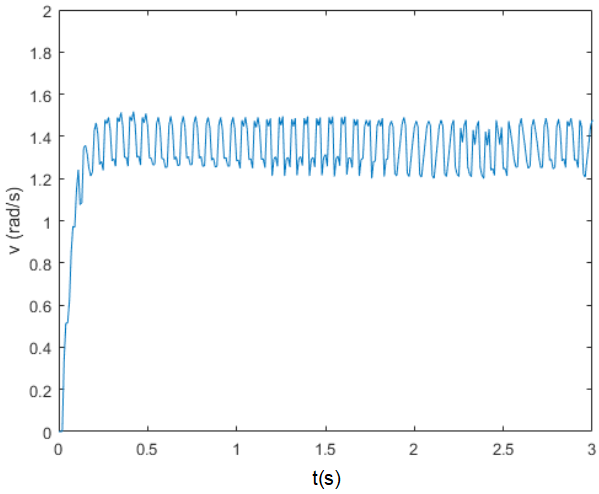

Figura 3.12: Gráfico de velocidade angular para entrada degrau de amplitude 100.

Com isso foi possível, definir por meio da análise do gráfico obtido os valores de K_m, da constante de tempo τ_m e do atraso L. O valor de K_m foi medido pela média da resposta final. O valor do atraso L foi o tempo até a resposta ser diferente de zero. E o valor da constante de tempo foi obtida verificando o tempo em que a resposta é igual a 63% do valor final médio menos o tempo de atraso inicial. Com isso obtemos o modelo dinâmico mencionado na subseção 2.7 expresso pela função de transferência 3.7.

$$G(s) = \frac{0.0138e^{-0.03}}{0.0512s + 1} \tag{3.7}$$

Além da resposta a uma referência constante, foi levantado o gráfico de velocidade versus ciclo

de trabalho em porcentagem, nos dois sentidos de giro do motor, apresentado na figura 3.13. Para isso, valores de ciclo de trabalho foram aplicados ao motor de 0% a 100% e a velocidade calculada pelo Arduino. Ao realizar o experimento foi possível saber que a zona morta fica em até 20% de ciclo de trabalho. Ou seja, valores abaixo disso não provocarão movimentação.

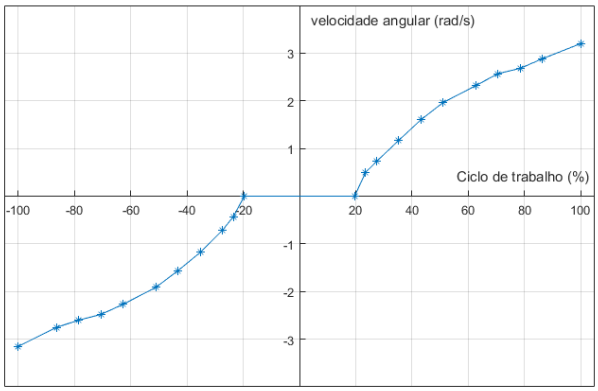

Figura 3.13: Gráfico de velocidade angular versus ciclo de trabalho.

3.3 Controle de velocidade

Para projeto do controlador de velocidade da planta, os requisitos são que resposta tenha erro nulo e não tenha sobrepasso.

Sem compensador, a resposta da planta em malha fechada, figura 3.14 , a uma entrada degrau unitário de referência possui erro considerável em regime permanente, como vemos na figura 3.15 em simulação.

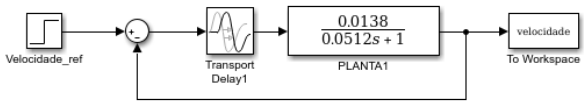

Figura 3.14: Simulação da planta em malha fechada.

Observando a função de transferência do motor verificamos que não há integrador para que a saída siga o sinal de referência sem erro em regime permanente. Por esse motivo, torna-se necessário o projeto de um compensador na forma Proporcional-Integral.

Para o projeto é desejado um controlador que produza uma resposta sem sobrepasso. Por esse motivo, foi escolhido o método de sintonização de PID CHR que foi apresentado no capítulo 2. Os parâmetros obtidos podem ser vistos nas equações 3.8, 3.9 e 3.10.

$$a = \frac{KL}{T} \tag{3.8}$$

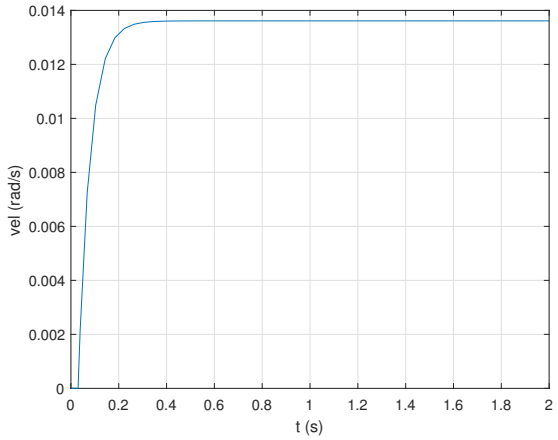

Figura 3.15: Resposta da planta para entrada degrau unitário em simulação.

$$Kp = \frac{0.35}{a} = 43.285 \tag{3.9}$$

$$Ti = 1.2T = 0.06144 \tag{3.10}$$

$$G_c(s) = 43.285\frac{(s + 16.276)}{s} \tag{3.11}$$

O controlador PI, obtido produziu uma nova resposta na planta, que agora segue sem erro o sinal de referência de 1rad/s, com um tempo de assentamento de 0.4s. O teste feito em simulação no MATLAB/SIMULINK pode ser visto na figura 3.16.

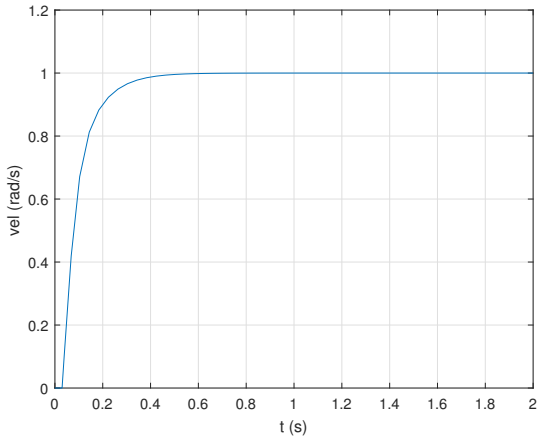

Figura 3.16: Gráfico da resposta em simulação com controlador PI.

Para implementar o algoritmo de controle no Arduíno, a função de transferência do compensador foi discretizada e depois transformada em uma equação de diferenças. Essa discretização foi realizada seguindo o método de integração trapezoidal, também chamado de bilinear ou de Tustin,

citado no capítulo 2. A função de transferência do compensador discretizada é expressa na equação 3.12.

A função de transferência relaciona a saída do controlador que chamaremos de u(z), que é a variável a ser controlada, e a entrada do controlador dada pelo erro e(z). Fazendo uma manipulação matemática temos a equação 3.13 onde z^{-1} é um atraso de 1 período de amostragem [11].

$$\frac{u(z)}{e(z)} = 46.808\frac{(1 - 0.8495z^{-1})}{(1 - z^{-1})} \tag{3.12}$$

$$u(z) - u(z)z^{-1} = 46.808\left(e(z) - 0.8495z^{-1}e(z)\right) \tag{3.13}$$

$$u[k] = 46.808(e[k] - 0.8495e[k-1]) + u[k-1] \tag{3.14}$$

A equação de diferenças 3.14 agora pode ser implementada em algoritmo para o controle do motor. Onde u[k] é o ciclo de trabalho a ser enviado para o motor, e[k] é o erro atual dado pela diferença da velocidade atual e a velocidade de referência, e[k-1] o erro na amostragem anterior, e u[k-1] o ciclo de trabalho calculado na amostragem anterior. O algoritmo completo pode ser visto no anexo de programas utilizados, que mostra o programa usado para o teste do controle de velocidade e posição no Arduino.

Na figura 3.17, vemos a saída obtida com o controle de velocidade implementado. A saída varia em torno do valor de referência dado que foi de 1rad/s. Isso ocorre devido a limitação do sinal PWM que tem uma resolução de ciclo de trabalho de 0.39%, logo valores intermediários não são suportados fazendo com que alguns valores de velocidade não sejam possíveis.

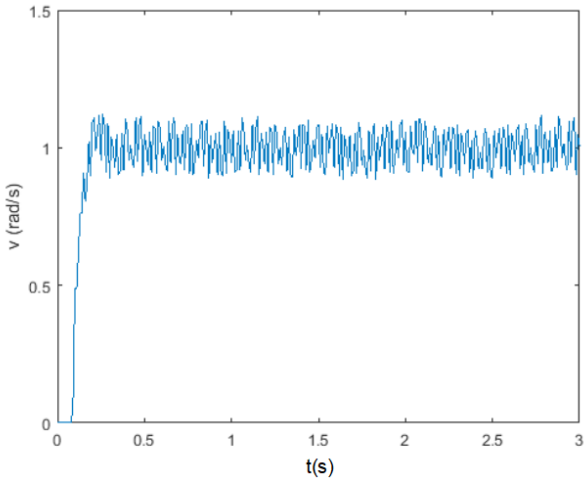

Figura 3.17: Gráfico da velocidade do motor com controlador PI para entrada degrau unitário.

3.4 Controle de posição

O controle da posição será realizado em uma malha externa como citado na subseção 2.8. O controlador PID da malha externa será responsável por gerar referências de velocidade para a malha interna. Além disso, devemos considerar o integrador responsável por transformar a velocidade em posição. Com o integrador, a planta já seria capaz de seguir uma referência de posição em degrau. Porém, para um controle mais suave de posição, queremos que a entrada de referência seja uma rampa. Assim, o controlador deve conter uma ação integral para que isso seja possível.

A nova malha a ser controlada $G_2(s)$ inclui o controlador de velocidade, $G_c, (s)$ em uma malha interna com a planta $G(s)$ e também um integrador que transforma a velocidade em posição, que pode ser visto na função de transferência de malha aberta na equação 3.15.

$$G_2(s) = \frac{G(S)G_c}{1 + G(S)G_c}\frac{1}{s} \tag{3.15}$$

O LGR desse sistema pode ser visto na figura 3.19. O LGR foi gerado a partir da ferramenta rltool() no MATLAB. Porém a ferramenta não é capaz de representar o efeito do atraso, por esse motivo ele modifica o LGR original a fim de melhor representá-lo que podemos ver na figura 3.19. Desconsiderando o atraso temos a função de transferência dada pela equação 3.16. O LGR da planta desconsiderando o atraso é mostrado na figura 3.18.

$$G_2(s) = 11.667\frac{s(s + 16.2760)(s + 19.5318)}{s^2(s + 19.5313)(s + 22.9093)(s + 8.2886)} \tag{3.16}$$

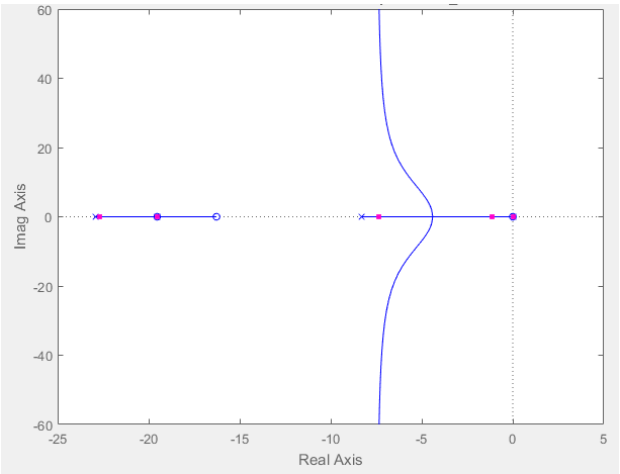

Figura 3.18: LGR da malha interna com integrador, desconsiderando atraso.

Podemos ver que a partir dele que a estabilidade do sistema ocorre para uma margem de ganho muito pequena, e se torna instável para K maior que 30, no LGR que considera o atraso e seu efeito na estabilidade do sistema.

Para esta planta mais complexa com margem de ganho pequeno para estabilidade, a sintonização do controlador PID foi realizada manualmente em simulação. Este método consistem em

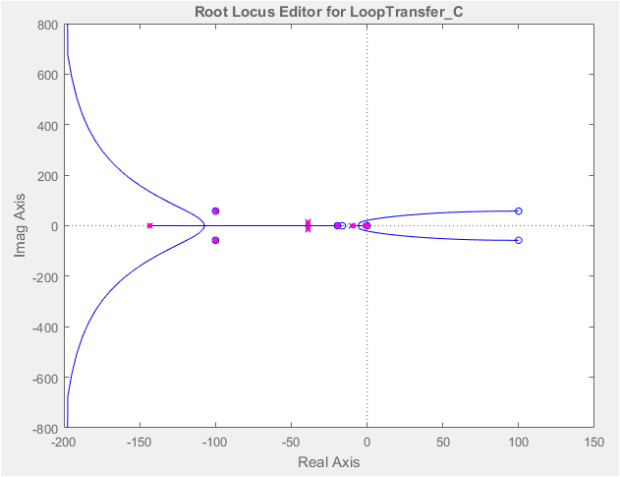

Figura 3.19: LGR da malha interna com integrador, considerando atraso.

tentativa e erro e análise da resposta da simulação até chegar aos parâmetros que resultem em uma resposta dentro das exigências de projeto, que para o caso foram de até 20% de sobrepasso e um tempo de assentamento de 2s. Este método é inspirado no segundo método de sintonização de parâmetros PID em malha fechada utilizando período e ganho crítico.

Primeiramente, foi implementado apenas uma compensador do tipo Kp, aumentando o ganho a partir de Kp igual a 1 até que a resposta seguisse o valor de referência com oscilação de amplitude constante. Este valor foi dividido por 2 para ser o ganho proporcional do compensador. O próximo a ser sintonizado foi o tempo de integração T_i, ajustado a partir de 1, até a resposta obtida seguir o valor de referência com um tempo de assentamento menor que 2s. Por último, foi feita o ajuste do tempo derivativo para diminuir o sobressinal para abaixo de 20%.

O parâmetros obtidos podem ser visto no diagrama de simulação na figura 3.20 e a função de transferência do compensador, após a soma de todos os termos, é apresentada na equação 3.17.

$$G_{pid}(s) = 0.8\frac{s^2 + 10s + 15}{s} \tag{3.17}$$

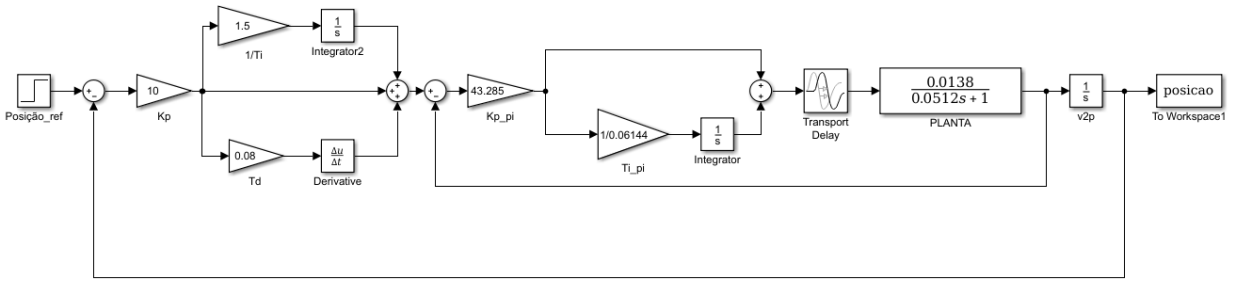

Figura 3.20: Diagrama de simulação em SIMULINK/MATLAB da malha de controle de posição.

A resposta da malha completa com os controladores em cascata a entrada degrau e rampa pode ser visto nas figuras 3.21 e 3.22, onde a curva em vermelho é a entrada de referência de posição e a curva em azul, a saída controlada. Como podemos ver, a saída segue o valor de referência sem

erro em regime permanente e com o sobrepasso abaixo do exigido para projeto. Mostrando que o método de de sintonização manual pode ser usado para o projeto de controladores PID para uma planta de maior complexidade.

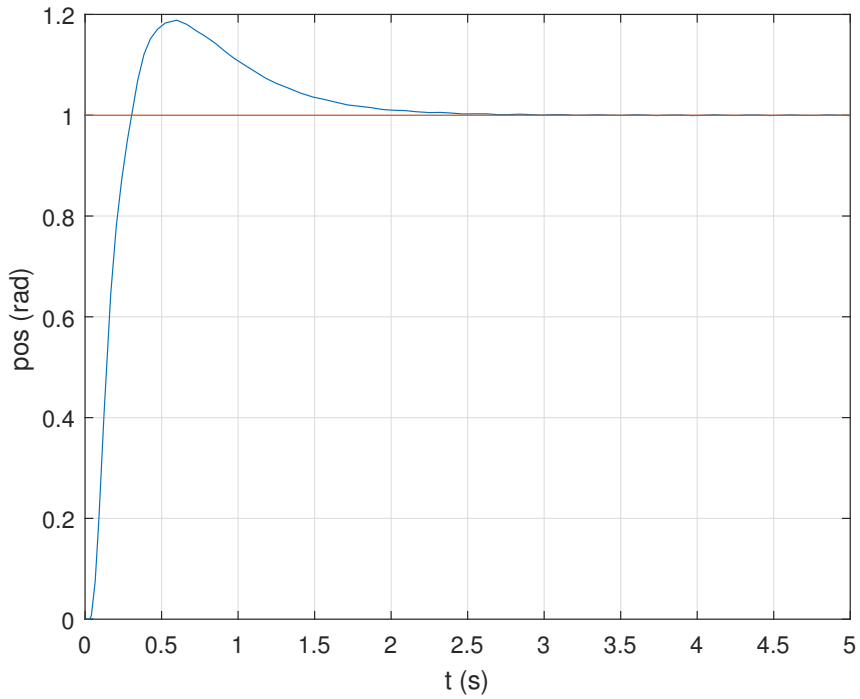

Figura 3.21: Gráfico da resposta do controle PID para entrada degrau unitário em simulação.

Para implementar no controle real do motor, transformamos a função de transferência em uma equação de recorrência a partir da função discretizada 3.18, da mesma forma feita anteriormente para o controle da velocidade. A equação de recorrência está expressa na equação 3.19.

Verificamos que a função de transferência do controlador obtido é não causal, ao possuir mais pólos que zeros, ou seja, em algum instante a saída dependerá de valores de entrada em instantes futuros, fazendo com que seja impossível de ser implementado fisicamente. Porém, ao aplicar a discretização a função se torna causal, dependendo apenas de valores em instantes anteriores.

$$G_{pid}(z) = 170.1\frac{(1 - 1.88z^{-1} + 0.8824z^{-2})}{1 - z^{-1}} \tag{3.18}$$

$$u[k] = 170.1(e[k] - 1.88e[k-1] + 0.8824e[k-2]) + u[k-1] \tag{3.19}$$

A equação de recorrência obtida foi implementada em algoritmo no programa de Arduino e a resposta obtida a entrada rampa com valor final constante é mostrada na figura 3.23

Podemos ver que a resposta do motor seguiu a referência dada em rampa como desejado. dentro dos limites exigidos de sobressinal e tempo de assentamento, chegando à posição final determinada.

Para implementar o controle de posição no Arduino foi necessário fazer com que a posição de

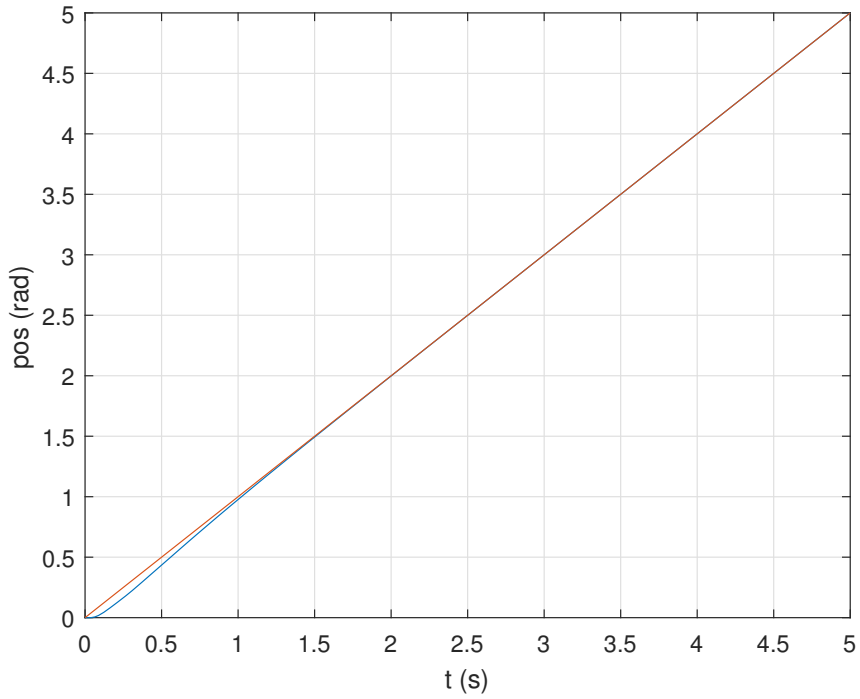

Figura 3.22: Gráfico da resposta do controle PID para entrada rampa unitária em simulação.

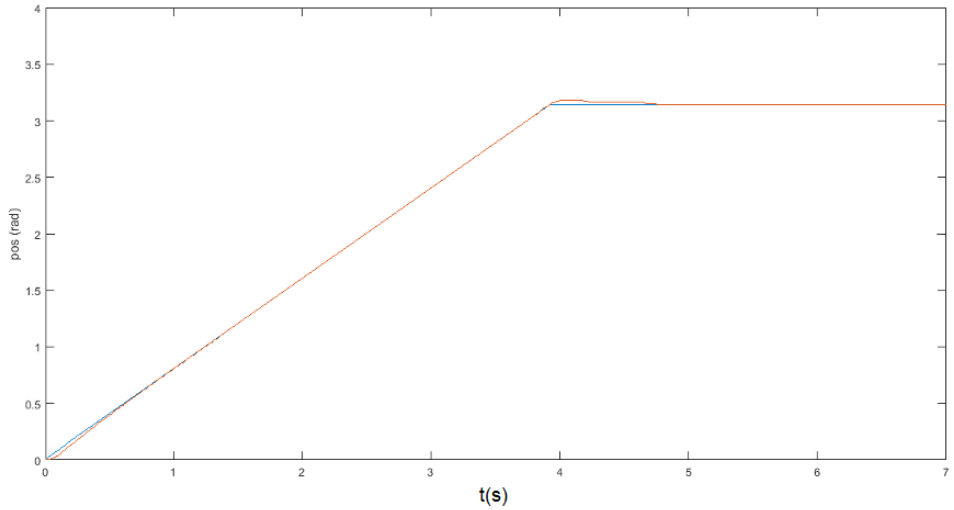

Figura 3.23: Resposta do motor a uma referência rampa.

referência sempre fosse arredondada para o múltiplo do deslocamento angular em cada transição mais próximo.

Como a posição de referência é dada em rampa, a cada *loop* de controle a nova posição de referência deve ser calculada enquanto esta não é igual a posição final desejada.O fluxograma apresentado na figura 3.24 apresenta um resumo do funcionamento do programa implementado em Arduino para o controle de um dos motores.

3.5 Implementação no Arduino

A implementação durante o projeto foi feita para um motor. No programa do módulo escravo ocorrem 3 interrupções durante a execução. Uma das interrupções é causada pelo temporizador *Timer2* para a contagem de *overflows*, usado no cálculo do tempo no programa. A outras duas interrupções são causadas pelos pinos de interrupção externa 2 e 3 no Arduino, por onde são lidas as transições do *encoder* dos motores A e B. As interrupções externas acionam uma rotina para a contagem de transições e do tempo entre as transições, para medida de velocidade e posição do motor.

Para que o controle seja feito é necessário que o *loop* inicie a cada tempo de amostragem que foi determinado em 10ms. Para calcular o *delay*, ou o tempo de espera para a próxima iteração, o tempo inicial e final é adquirido e essa diferença é subtraída do tempo de amostragem. Isso deve ser feito de modo dinâmico pois há interrupções acontecendo durante o laço de controle que podem interferir no tempo total do mesmo.

Para um movimento mais suave dos motores e evitar sobrepassos muito grandes, foi utilizada referência de posição em rampa, com uma inclinação fixa, dada por uma velocidade dentre dos limites do motor. Para essa inclinação em rampa, o código calcula a diferença entre a posição inicial atual e a posição final desejada e incrementa a cada *loop* até chegar no valor final de posição, que deve ser seguido pelas equações de recorrência para o controle de posição.

O controle é feito inicialmente pela posição e esta gera o valor de referência de velocidade para o controle de velocidade. O algoritmo de controle de velocidade gera os valores de ciclo de trabalho compatíveis com os valores do Arduino e faz o acionamento dos motores. Estes valores são testados primeiramente com os limites possíveis. Valores acima de 255 são remapeados em 255.

O funcionamento básico do programa pode ser visto no fluxograma apresentado na figura 3.24, e o código se encontra no Anexo II.

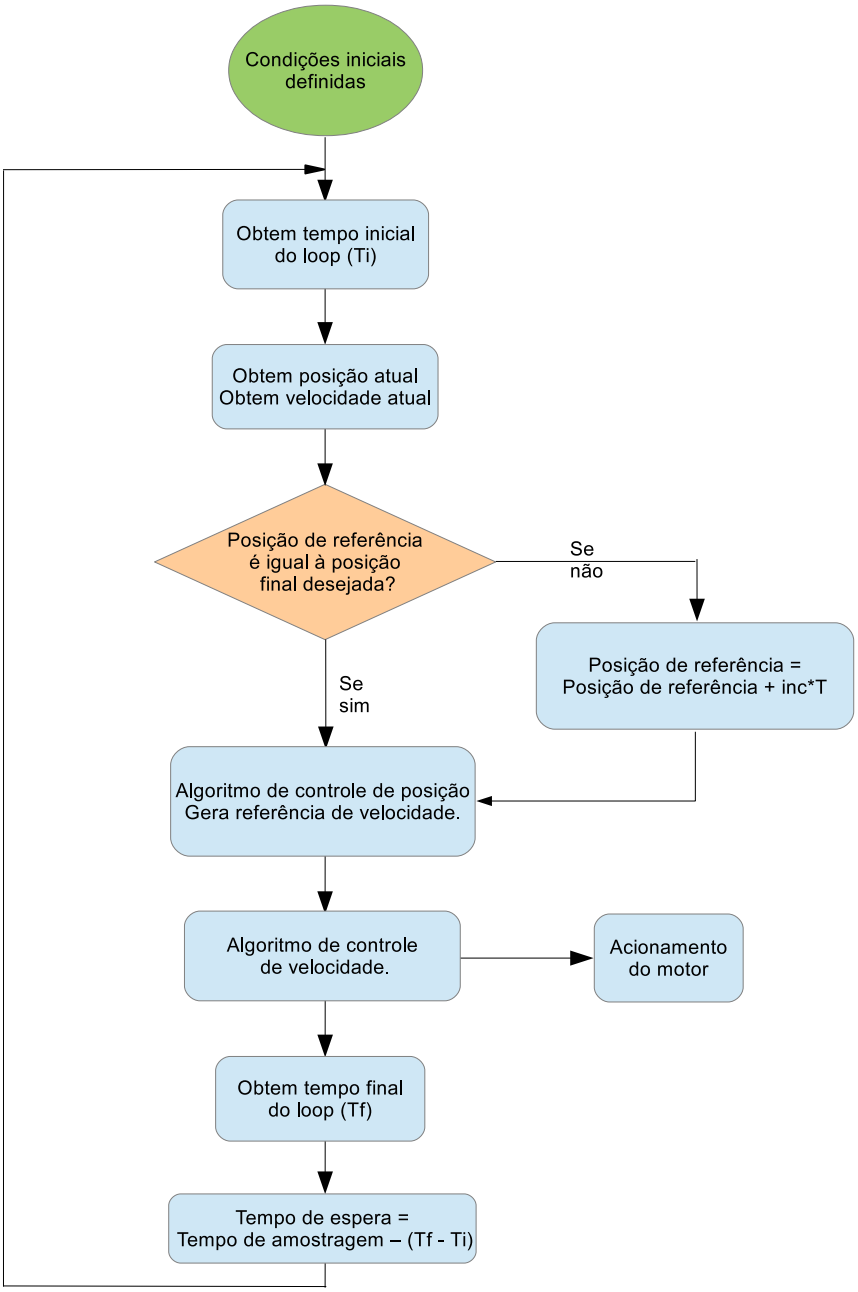

Figura 3.24: Fluxograma que representa o algoritmo implementado no Arduino.

Capítulo 4

Conclusões

Como proposto inicialmente, o objetivo deste projeto era fazer o controle de velocidade e posição dos motores que atuam nas juntas do Rhino XR-4 e propor o sistema para o controle de todas as juntas.

O controle do motor usando o controle em cascata com dois controladores, sendo o externo de controle de posição e o interno para controle de velocidade, se mostrou satisfatório. O motor foi capaz de seguir os valores de referência de posição em degrau e rampa durante os experimento sem erro em regime permanente e com o tempo de reposta dentro do desejado.

Verificou-se que o controle de velocidade aplicado no motor obteve uma resposta que variou em torno do valor de referência, seguindo o valor de referência com média sem erro em regime permanente. A oscilação da resposta pode ter ocorrido devido as limitações no controle da potência do motor que foi feita por meio de um CI e enviado um sinal com ciclo de trabalho de forma quantizada.

Além disso a baixa resolução do *encoder* diminui a precisão medida da posição e consequentemente de velocidade. Com um numero maior de faixas no codificador óptico seria possível implementar o primeiro método de calculo de velocidade, proposto no capitulo 3, e obter uma velocidade angular mais precisa.

Outro possível motivo para a presença de erro pode ter ocorrido no próprio cálculo de velocidade. O cálculo de velocidade foi feito com o intuito de se calcular a velocidade instantânea a partir do tempo decorrido entre uma interrupção e a próxima. Caso haja a perda de uma contagem do temporizador 2, que pode ocorrer durante as instruções, há um erro no cálculo do tempo e isso irá refletir na medida de velocidade.

As interrupções que ocorrem a cada estouro do temporizador também podem interferir no funcionamento do sistema, já que ele leva a uma rotina para incremento do contador de *overflows* como mencionado na subseção 3.1.4.

4.1 Perspectivas Futuras

Para dar continuidade ao projeto ainda há muito a ser feito. Em relação à parte física dos módulos escravos, é necessário obter mais dois Arduínos e dois módulos de ponte H. Em relação ao software é necessário replicar o algoritmo de controle de posição e velocidade para o segundo motor.

Ainda para o módulo escravo, é necessário a criação de rotinas para a comunicação serial com o mestre para que possa receber os valores de referência de posição ou velocidade, o tipo de controle que deve ser implementado, a depender da referência, e também rotinas para enviar para o mestre os valores atuais da posição da posição, para que possa ser armazenado em uma memória não volátil.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] IMAGEM robô SCARA. `https://www.machinedesign.com/robotics/what-s-difference-between-industrial-robots`. [Online; acesso em 1-Junho-2018].

[2] FONTE L298N modificado. `http://engineerts.weebly.com/uploads/8/3/3/5/83350222/untitled.png?667`. [Online; acesso em 1-Junho-2018].

[3] SCHILLING, R. J. *Fundamentals of robotics: analysis and control*. [S.l.]: Prentice Hall New Jersey, 1990.

[4] OGATA, K. *Modern control engineering*. [S.l.]: Prentice hall, 2002.

[5] TRONXILABS. *Tutorial - L298N Dual Motor Controller Module 2A and Arduino*. `https://tronixlabs.com.au/news/tutorial-l298n-dual-motor-controller-module-2a-and-arduino/`. [Online; acesso em 1-Junho-2018].

[6] SPONG, M. W. et al. *Robot modeling and control*. [S.l.]: Wiley New York, 2006.

[7] ATMEL. *8bit AVR Microcontroller ATmega328/328P Datasheet*. [S.l.], 2012.

[8] NISE, N. S. *Control Systems Engineering*. 6. ed. [S.l.]: Wiley New York, 2010.

[9] DAFUL, A. G. Comparative study of pid tuning methods for processes with large & small delay times. In: IEEE. *Advances in Science and Engineering Technology International Conferences (ASET), 2018*. [S.l.], 2018. p. 1–7.

[10] OMEGA ENGINEERING, INC. Tuning a pid controller. 2016.

[11] OGATA, K. *Discrete-time control systems*. [S.l.]: Prentice Hall Englewood Cliffs, NJ, 1995.

ANEXOS

I. CINEMÁTICA DO RHINO XR-4

I.1 Cinemática Direta

Na figura I.1 vemos a representação dos parâmentros DH, onde θ_i é a rotação sofrida pelo frame anterior em torno de seu eixo z, α_i a rotação sofrida em torno do eixo x, d_i a translação do frame ao longo do eixo z, e a_i ao longo do eixo x.

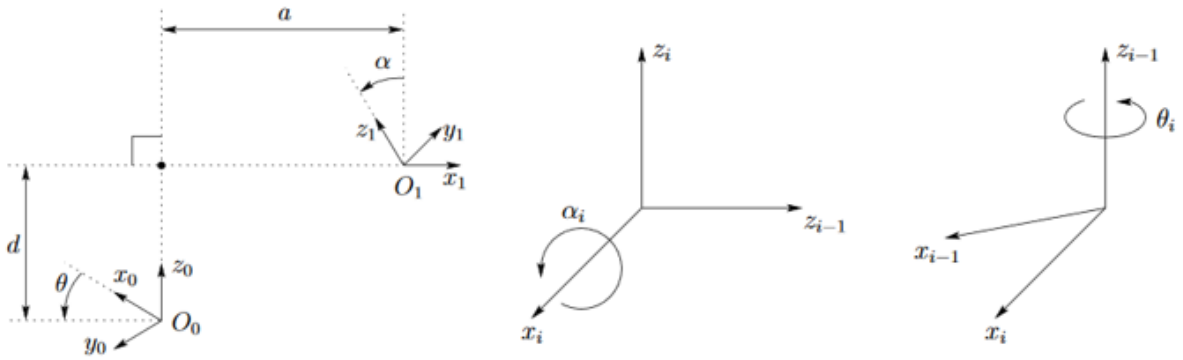

Figura I.1: Representação dos parâmetros D-H

Para a definição dos parâmetros a,d,α e θ de forma padronizada, deve-se seguir o algoritmo abaixo [3]:

1. Definir o número das juntas de 1 a n, começando pela base e terminando com a ferramenta.
2. Determinar um frame L_0 de coordenadas ortonormal à base do robô, fazendo com que o eixo z^0 fique alinhado ao eixo da junta 1. Definir k = 1.
3. Alinhar z^k com o eixo da junta k+1.
4. Localizar a origem de L_k na interseção dos eixos z^k e $z^{(k-1)}$. Se eles não se cruzarem, usar a interseção de z^k e um eixo normal entre z^k e $z^{(k-1)}$.
5. Selecionar x^k de forma que seja normal aos eixos z^k e $z^{(k-1)}$. Se z^k e $z^{(k-1)}$ forem paralelos, apontar x^k para longe de $z^{(k-1)}$.
6. Selecionar y^k para completar o frame de coordenadas ortonormais.
7. Fazer k = k + 1. Se k < n, vá para o passo 3; caso contrário, continue.
8. Definir a origem de L_n na ponta da ferramenta. Alinhe o eixo z_n com o vetor de aproximação, y_n com o vetor de deslizamento e o eixo x_n com o vetor normal a ferramenta. Faça k = 1.

9. Localize o ponto b^k na interseção dos eixos x^k e $z^{(k-1)}$. Se não se interceptarem usar a interseção de x^k com a normal comum entre x^k e $z^{(k-1)}$.

10. Calcular o ângulo θ_k como o ângulo de rotação de $x^{(k-1)}$ a x^k em torno de $z^{(k-1)}$.

11. Calcular d_k como a distância da origem do frame $L_{(k-1)}$ ao ponto b_k medido ao longo do eixo $z_{(k-1)}$.

12. Calcular a_k como a distância da origem do frame $L_{(k-1)}$ ao ponto b_k medido ao longo do eixo x_k.

13. Calcular α_k como o ângulo de rotação de z^k a $z^{(k-1)}$ em torno do eixo x_k.

14. Defina k = k+1. Se $k \leq n$, vá para o passo 9; caso contrário, pare.

Seguindo o algoritmo, temos a configuração de juntas para o Rhino XR4 mostrada na figura I.2.

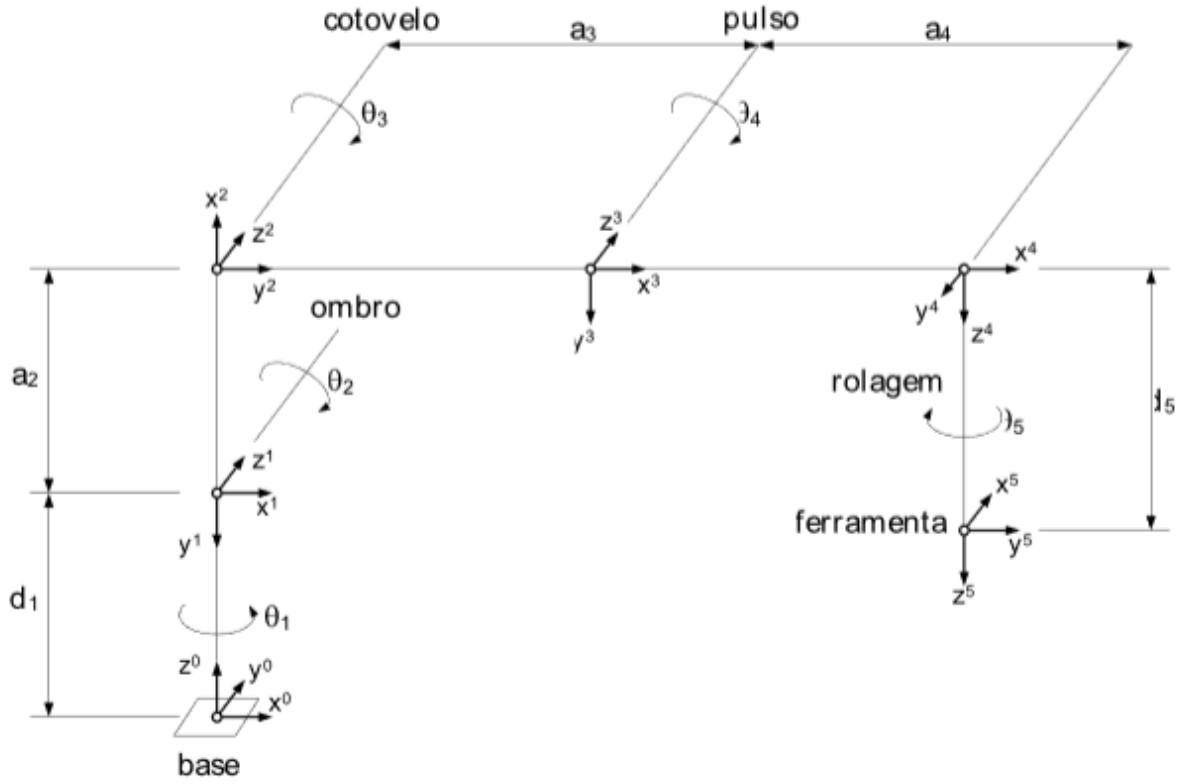

Figura I.2: Alocação das coordenadas e parâmetros cinemáticos do Rhino XR-4[3].

$$q = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 \end{bmatrix} \tag{I.1}$$

Dadas as especificações do robô, temos nas equações I.2 e I.3 os valores dados em milímetros.

$$d = \begin{bmatrix} 260.0 & 0.0 & 0.0 & 0.0 & 171.1 \end{bmatrix} \tag{I.2}$$

Tabela I.1: Parâmetros Cinemáticos - Algoritmo de Denavit-Hartenberg.

Junta	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	$-\pi/2$
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0
4	θ_4	0	a_4	α_i
5	θ_5	d_5	0	0

$$a = \begin{bmatrix} 0.0 & 228.6 & 228.6 & 9.5 & 0.0 \end{bmatrix} \tag{I.3}$$

Os valores de θ_i, chamados de variáveis de junta, que são os ângulos que os motores devem rotacionar para atingir uma determinada posição e orientação desejada para a ferramenta que podem ser representadas na forma da equação I.1. Para obter a matriz de transformação de cada uma das juntas k em relação ao frame de coordenadas da base é necessário multiplicar a matriz de transformação de cada um dos links em relação ao anterior, até a base, expresso na equação I.4.

$$T_0^k = T_0^1 * T_1^2 ... T_{k-2}^{k-1} * T_{k-1}^k \tag{I.4}$$

$$T_{i-1}^i = Rot_{(z,\theta_i)} Trans_{(z,d_i)} Trans_{(x,a_i)} Rot_{(x,\alpha_i)} = \begin{bmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_1} & -s_{\alpha_1} & 0 \\ 0 & s_{\alpha_1} & c_{\alpha_1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{I.5}$$

Para o caso do Rhino XR4, temos 5 juntas. Para cada junta, seguindo o algoritmo D-H e as equações I.4 e I.5, temos a matriz homogênea de transformação da ferramenta em relação à base expressa na equação I.6, onde c_{234} equivale a $\cos(\theta_2 + \theta_3 + \theta_2)$.

$$T_0^5 = \begin{bmatrix} c_1 c_{234} c_5 + s_1 s_5 & -c_1 c_{234} s_5 + s_1 c_5 & -c_1 s_{234} & c_1 (a_2 c_2 + a_2 c_{23} + a_4 c_{234} - d_5 s_{234}) \\ s_1 c_{234} c_5 - c_1 s_5 & -s_1 c_{234} s_5 - c_1 c_5 & -s_1 s_{234} & s_1 (a_2 c_2 + a_2 c_{23} + a_4 c_{234} - d_5 s_{234}) \\ -s_{234} c_5 & s_{234} s_5 & -c_{234} & d_1 - a_2 s_2 - a_3 s_{23} - a_4 s_{234} - d_5 c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{I.6}$$

I.2 Cinemática Inversa

Vamos considerar a posição e a orientação da ferramenta como sendo o vetor w:

$$W = \begin{bmatrix} W^1 \\ W^2 \end{bmatrix} \tag{I.7}$$

Onde W^1 representa a posição e W^2 a orientação ou atitude da ferramenta. Igualando às equações obtidas do modelo cinemático direto, temos.

$$W(q) = \begin{bmatrix} c_1\left(a_2c_2 + a_2c_{23} + a_4c_{234} - d_5s_{234}\right) \\ s_1\left(a_2c_2 + a_2c_{23} + a_4c_{234} - d_5s_{234}\right) \\ d_1 - a_2c_2 - a_2c_{23} - a_4c_{234} - d_5s_{234} \\ -exp(q_5/\pi)c_1s_{234} \\ -exp(q_5/\pi)s_1s_{234} \\ -exp(q_5/\pi)c_{234} \end{bmatrix} \tag{I.8}$$

Fazendo os cálculos necessários, obtemos a solução completa dada pelas equações I.9 a I.17 [3].

$$q_1 = atan2(W_2, W_1) \tag{I.9}$$

$$q_{234} = atan2(-(c_1W_4 + s_1W_5), -W_6) \tag{I.10}$$

$$b_1 = c_1W_1 + s_1W_2 - a_4c_{234} + d_5s_{234} \tag{I.11}$$

$$b_2 = d_1 + a_4s_{234} - d_5c_{234} - W_3 \tag{I.12}$$

$$||b||^2 = b_1^2 + b_2^2 \tag{I.13}$$

$$q_3 = \pm arccos\frac{||b||^2 - a_2^2 - a_3^2}{2a_2a_3} \tag{I.14}$$

$$q_2 = atan2\left[(a_2 + a_3c_3)b_2 - a_3s_3b_1, (a_2 + a_3c_3)b_1 + a_3s_3b_2\right] \tag{I.15}$$

$$q_4 = q_{234} - q_2 - q3 \tag{I.16}$$

$$q_5 = \pi ln(W_4^2 + W_5^2 + W_6^2)^{\frac{1}{2}} \tag{I.17}$$

Tendo as variáveis de junta calculadas pela cinemática inversa, é possível usa-las como referência para o controle dos atuadores.

II. PROGRAMAS UTILIZADOS

II.1 Programa para controle de posição e velocidade no Arduino

```
/*
Programa elaborado para controle de velocidade e posicao
de um motor de corrente continua. O programa gera uma referencia
rampa de posicao com inclinacao definida pela variavel 'inc'.
Velocidade medida em rad/seg.
posicao medida em radianos.
*/

#define pos 1
#define neg 0
#define off 2

// PINAGEM ARDUINO:********
// os pinos 2 e 3 sao usados para interrupcao.

#define chA_MOTORA 2
#define chA_MOTORB 3

#define fimA_PIN 4
#define fimB_PIN 5
#define chB_MOTORA 6
#define chB_MOTORB 7

//L298n:
#define IN1 A0
#define IN2 A1
#define IN3 A2
#define IN4 A3
#define ENA 9
#define ENB 10

//*************************
// VARIAVEIS TIMER 2

unsigned long counterT2 = 0;      // OVERFLOW DA VARIAVEL EM 2^31-1*16
float T2res = 16;                 //cada pulso do timer equivale a 16us
```

```
//*************************
// VARIAVEIS GLOBAIS
float changeRes = 0.00799;        // 2*pi/786
float risingRes = 0.01599;        // 2*pi/393
float Tloop = 10000;              // periodo de amostragem desejado em us.

//*************************
// VARIAVEIS DENTRO DO LOOP

unsigned long timeL,timer;      // para calculo de tempo dentro do loop.
unsigned long timeLaux;
unsigned long OVFTL;            // conta overflow atual
unsigned long OVFTLaux;         // conta overflow anterior
unsigned long TL;               // guarda periodo atual do LOOP
float delayL = 0;               // guarda periodo atual do LOOP

//CONTROLADOR VELOCIDADE:
float erro_velAaux = 0;
float erro_velA = 0;
int PWM_Aaux = 0;
int PWM_A = 0;
float ref_velA = 0;             // limites: 0.43rad/s a 3.2rad/s

//CONTROLADOR POSICAO:
float erro_posAaux2 = 0;
float erro_posAaux = 0;
float erro_posA = 0;

float ref_velAaux2 = 0;
float ref_velAaux = 0;

float ref_posA;                 // referencia de posicao.
float final_posA;
float final_posAaj = 0;
float ref_posAaux;
float inc;
float posA;

//*************************
// MOTOR A
```

```
unsigned long timeA;            // para calculo de tempo e frequencia.
unsigned long timeAaux;
unsigned long OVFTA;            // conta overflow atual
unsigned long OVFTAaux;         // conta overflow anterior
unsigned long Ta;               // guarda periodo atual do motor A
unsigned long Ta_aux;

long counterA = 0;
long counterAaux = 0;
float velA = 0;
int chA = 0;
int chB = 0 ;
int chAaux = 0;
int flagA = 1;                  //indica direcao de rotacao;

//INICIO DO PROGRAMA

void countA();

void L298N_MOTORA(int dir, int pwm);

//TIMER2 PARA TER TEMPO MAIS PRECISO
ISR(TIMER2_OVF_vect)
{
   counterT2++;   // conta quantos overflows desde o inicio do programa.
}

void setup() {
  Serial.begin(115200);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(13, OUTPUT);

  attachInterrupt(0, countA, CHANGE);

  //INICIA O LOOP APENAS QUANDO ENVIA 'A' PELA SERIAL. teste de conexao.
  digitalWrite(13,HIGH);
  Serial.println('a');
  char a = 'b';
  while(a!='a'){
```

```
    a = Serial.read();
  }
  digitalWrite(13,LOW);          // teste de conexao visual

  TCCR2A = 0x00;                 // Timer operando em modo normal
  TCCR2B = 0x06;                 // Prescaler 1:256 - resolucao de 16us
  TIMSK2 = 0x01;                 // Habilita interrupcao do Timer2

  // inicializacao de variaveis:
  chAaux = digitalRead(chA_MOTORA);
  Ta = 10000000000;
  velA = 0;
  posA = 0;

  erro_posAaux2 = 0;
  erro_posAaux = 0;
  ref_velAaux = 0;
  ref_posA = 0;                  //referencia inicial
  ref_posAaux = 0;
  final_posA = 3.1416;           //posicao final desejada
  inc = 1.5;                       //inclinacao da rampa fixa

  ref_velA = 0;
  final_posA = 3.1415;
  final_posA = round(final_posA/changeRes)*changeRes;

  counterT2 = 0;                 // zera contador de overflows
  TCNT2 = 0x00;                  // zera contador
  timeAaux = TCNT2;
  OVFTAaux = counterT2;
}

void loop() {
  OVFTLaux = counterT2;
  timeLaux = TCNT2;

  ////////////////////////////////////////
  //calculo da velocidade atual:
  if(flagA == 1) velA = changeRes/(Ta*0.000001);
  else velA = (0.0-changeRes)/(Ta*0.000001);
  //Serial.println(velA,4);
```

```
posA = counterA*changeRes;      // verifica posicao angular atual

//geracao de referencia degrau.
if(posA != final_posA){
   if(ref_posA != final_posA){
      if(final_posA >= posA ){
        ref_posA = ref_posAaux + inc*T;
        ref_posA = round(ref_posA/changeRes)*changeRes;
      }
      else{
        ref_posA = ref_posAaux - inc*T;
        ref_posA = round(ref_posA/changeRes)*changeRes;
      }
   }
}else{
  ref_posAaux = posA;
  ref_posA = posA;
}

ref_posAaux = ref_posA;
Serial.println(ref_posA,4);
Serial.println(posA,4);
//Serial.println(counterA);

//CONTROLE PID DE POSICAO MOTOR A:
erro_posA = ref_posA - posA;
ref_velA = 170.1*(erro_posA - 1.88*erro_posAaux + 0.8824*erro_posAaux2)
+ ref_velAaux;
erro_posAaux2 = erro_posAaux;
erro_posAaux = erro_posA;
ref_velAaux = ref_velA;

controle_vel();

OVFTL = counterT2;
timeL = TCNT2;

// periodo instantaneo em microssegundos:
TL = ((OVFTL-OVFTLaux)*255+timeL-timeLaux)*T2res;
// calcula delay para que tempo de amostragem seja 10ms
delayL = Tloop - TL;
delayMicroseconds(delayL);
```

```
  //Serial.println(TL+delayL,5);
}

//FUNCAO PARA CONTAGEM DE PULSOS NO MOTOR A
void countA(){
  OVFTA = counterT2;
  timeA = TCNT2;
  noInterrupts();
  // periodo instantaneo us:
  Ta = ((OVFTA-OVFTAaux)*255+timeA-timeAaux)*T2res;
  interrupts();
  timeAaux = timeA;
  OVFTAaux = OVFTA;

  chB = digitalRead(chB_MOTORA);
  chA = digitalRead(chA_MOTORA);

  if(chB != chAaux){
    counterA--;
    flagA = 0;
  }
  else{
    counterA++;
    flagA = 1;
  }

  chAaux = chA;          // guarda ultimo estado do canal A do encoder.
}

//Acionamento do motor:
void L298N_MOTORA(int dir, int pwm){
    if (dir == 0){
        //digitalWrite(IN1,LOW);
        //digitalWrite(IN2,HIGH);
        PORTC &= ~(1<<PORTC0);
        PORTC |= (1<<PORTC1);
    }
    else if(dir == 1){
        //digitalWrite(IN1,HIGH);
        //digitalWrite(IN2,LOW);
        PORTC |= (1<<PORTC0);
        PORTC &= ~(1<<PORTC1);
```

```
    }
    else{          // usado para erro ou desligar o motor
       //digitalWrite(IN1,LOW);
       //digitalWrite(IN2,LOW);
       PORTC &= ~(1<<PORTC0);
       PORTC &= ~(1<<PORTC1);
    }
    analogWrite(ENA,pwm);
}

void controle_vel(){
  //CONTROLE DE VELOCIDADE MOTOR A:
  erro_velA = ref_velA - velA;
  PWM_A = round(46.808*(erro_velA - 0.8495*erro_velAaux) + PWM_Aaux);
  erro_velAaux = erro_velA;
  PWM_Aaux = PWM_A;

  //ENVIO DO PWM PARA O MOTOR A
  if(ref_velA >= 0){
        if(PWM_A > 255){
              L298N_MOTORA(pos,255);
        }
        else if(PWM_A > 0){
             L298N_MOTORA(pos,PWM_A);
          }
        else{
            L298N_MOTORA(off,0);
        }
  }
  else{
        if(PWM_A < -255){
              L298N_MOTORA(neg,255);
        }
        else if(PWM_A < 0){
             L298N_MOTORA(neg,abs(PWM_A));
          }
        else{
            L298N_MOTORA(off,0);
        }
  }
}
```