Universidade de Brasília
Faculdade de Tecnologia

Controle de caminhada de um exoesqueleto robótico

Fábio Rodrigues de Andrade Santos

PROJETO FINAL DE CURSO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2023

Universidade de Brasília
Faculdade de Tecnologia

Controle de caminhada de um exoesqueleto robótico

Fábio Rodrigues de Andrade Santos

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Walter de Britto Vidal Filho

Brasília
2023

Santos, Fábio Rodrigues de Andrade.
S237c Controle de caminhada de um exoesqueleto robótico / Fábio Rodrigues de Andrade Santos; orientador Walter de Britto Vidal Filho. -- Brasília, 2023.
76 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2023.

1. Robótica. 2. Exoesqueleto. 3. Mecatrônica. 4. Bioengenharia. I. de Britto Vidal Filho, Walter, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Controle de caminhada de um exoesqueleto robótico

Fábio Rodrigues de Andrade Santos

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 15 de Dezembro de 2023:

Prof. Dr. Walter de Britto Vidal Filho,
UnB/FT/ENM
Orientador

Carlos Humberto Llanos Quintero,
UnB/FT/ENM
Examinador interno

Guilherme Caribé de Carvalho,
UnB/FT/ENM
Examinador interno

Brasília
2023

Esse trabalho é dedicado às pessoas que se dedicam a melhorar vidas e o futuro da humanidade com projetos e estudos. Aos que trabalham para fazer a diferença, por serem inspiração e se devotarem ao próximo.

Agradecimentos

A minha gratidão se dirige à minha família, cujo apoio constante e incentivo incansável me inspiraram a buscar sempre o melhor de mim. Agradeço também aos amigos e colegas que compartilharam esta jornada comigo, enriquecendo cada passo. Meus agradecimentos se estendem ao corpo docente e a todas as pessoas que, contribuíram para moldar a minha trajetória na Universidade. Vocês foram peças fundamentais no meu percurso acadêmico e no desenvolvimento da minha jornada a me tornar minha melhor versão de mim.

”The journey of a thousand miles begins with a single step.”
(Lao Tzu)

Resumo

Este Trabalho de Graduação apresenta os estudos que serviram como base para o desenvolvimento de um projeto de controle de marcha para um exoesqueleto robótico de membros inferiores. O documento abrange a teoria subjacente a esse tipo de projeto, bem como os detalhes do desenvolvimento prático. Um código de controle de trajetória foi desenvolvido para dois motores do exoesqueleto robótico, correspondentes ao quadril e ao joelho. O equipamento utilizado inclui motores da maxon motors e o microcontrolador ESP32, a placa ESP32-DevKitC4. A validação de resultados do código foi realizada em uma bancada de testes de um único motor em ambiente controlado e simulações do movimento completo.

Palavras-chave: Robótica. Exoesqueleto. Mecatrônica. Bioengenharia.

Abstract

ABSTRACT This Undergraduate Project comprises studies utilized as a reference for the development of a gait control project for a lower limb exoskeleton. The document encompasses the theory involved in such a project, along with the practical development details. A trajectory control code was devised for two motors in a robotic exoskeleton, corresponding to the hip and knee, utilizing maxon motors equipment and the ESP32 microcontroller, the ESP32-DevKitC4 board.Code result validation was performed on a single motor test bench in a controlled environment and through simulations of the entire movement.

Keywords: Robotics. Exoskeleton. Mechatronics. Bioengineering.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas 15
AVC	Acidente Vascular Cerebral 15
AVE	Acidente Vascular Encefálico 19
BLE	*Bluetooth Low Energy* 52
DH	Denavit-Hartenberg 33
MIMO	*Multi-Input Multi-Outout* 22
UnB	Universidade de Brasília 15
USP	Universidade de São Paulo 19

Sumário

1 Introdução

1.1 Contextualização

No Brasil, uma pesquisa realizada pelo IBGE aponta que 7,8 milhões de brasileiros apresentam deficiência física nos membros inferiores, quantidade que representa 3,8% da população acima dos dois anos de idade (CNN BRASIL, 2023). A melhora da qualidade de vida das pessoas que se encontram nessa estatística é um dos focos no estudo e desenvolvimento de tecnologias baseadas em robótica assistiva. Diante desse cenário, avanços tecnológicos têm aberto novas possibilidades para superar essas limitações físicas, e um campo promissor é o desenvolvimento de exoesqueletos robóticos para membros inferiores.

O presente trabalho tem como objetivo abordar o controle de um exoesqueleto robótico para membros inferiores, um sistema tecnológico que visa auxiliar e aprimorar a capacidade locomotora de pessoas com dificuldades em locomoção. O estudo da tecnologia de exoesqueletos é uma área de crescente interesse científico e tecnológico, uma vez que oferece a possibilidade de restaurar parcialmente a mobilidade e melhorar a autonomia desses indivíduos.

A perda de movimento nas pernas pode ser ocasionada por uma variedade de fatores, tais como lesões medulares, acidentes automobilísticos, doenças neuromusculares, AVCs e outras condições que afetam o sistema nervoso ou o sistema musculoesquelético. Essas condições podem ter origem congênita ou adquirida, acarretando em deficiências permanentes ou temporárias.

Diante desse contexto, o desenvolvimento de tecnologias de suporte e reabilitação, como os exoesqueletos robóticos, tem demonstrado grande potencial para ajudar a superar as limitações impostas pela perda de movimento nos membros inferiores. Ao possibilitar a interação harmoniosa entre a tecnologia robótica e o usuário, o exoesqueleto pode restaurar a mobilidade, proporcionar maior independência nas atividades diárias e, em muitos casos, melhorar a saúde física e emocional dos pacientes.

Espera-se que este trabalho contribua para a disseminação do conhecimento sobre exoesqueletos robóticos para membros inferiores, bem como para o avanço da pesquisa e desenvolvimento de soluções tecnológicas que promovam maior inclusão e melhoria na qualidade de vida de pessoas com dificuldades de locomoção.

Por fim, destaca-se que este é um estudo inicial e que a área de exoesqueletos robóticos está em constante evolução, abrindo caminho para novas descobertas e aplicações futuras. O engajamento contínuo com pesquisas e aprimoramentos nesse campo é fundamental para tornar a tecnologia ainda mais acessível e eficiente, beneficiando um número cada vez maior

de pessoas que podem se beneficiar dessa inovadora abordagem na reabilitação e no suporte à mobilidade.

1.2 Objetivos

1.2.1 Objetivo geral

O trabalho tem como objetivo principal o desenvolvimento de um controle de trajetória para um exoesqueleto robótico de membros inferiores, com o intuito de se permitir o auxílio e autonomia para uma pessoa que possa sofrer com a falta de movimento nos membros inferiores.

1.2.2 Objetivos específicos

- Desenvolver um estudo de revisão sobre projetos que sirvam de base e motivação ao projeto proposto;
- Estudo sobre a biomecânica do movimento de caminhada humana;
- Controlar a trajetória de motores por meio do uso de sensores de Efeito Hall;
- Elaboração de um código de controle para o microcontrolador ESP32.

1.3 Estrutura do trabalho

O trabalho está dividido em 6 seções principais:

1. Introdução: Capítulo introdutório que apresenta uma ideia do tem que será discutido ao longo do documento;
2. Revisão Bibliográfica: Capítulo que contém a base dos trabalhos utilizados como referência para o projeto, como também trabalhos anteriores do mesmo projeto;
3. Revisão Teórica: Capítulo que contém a base teórica dos métodos utilizados demonstrando o conhecimento base necessário para o desenvolvimento do projeto;
4. Desenvolvimento: Capítulo que contém a elaboração do projeto, como os códigos e resultados iniciais;
5. Análises e Resultados: Capítulo que contém a observação e comentários a partir dos resultados obtidos;
6. Considerações Finais: Capítulo concluinte do projeto, contém comentários finais e sugestões de melhorias para projetos futuros.

2 Revisão Bibliográfica

Este capítulo demonstrará uma breve apresentação de projetos anteriores na UnB, projetos tomados como base ou como auxílio de etapas de desenvolvimento do projeto que será apresentado.

2.1 Trabalhos nacionais

2.1.1 Trabalhos na UnB

Este trabalho é uma continuação de trabalhos já existentes na UnB, destacam-se os projetos (CÓRDOBA, 2021), (FREIRE, 2019) e (PERES, 2023). Os dois primeiros trabalhos mostram o projeto mecânico, sendo o primeiro com a montagem do exoesqueleto, com isso se tem os detalhes do protótipo, como o modelo CAD do mesmo, suas especificações, limitações, custos, vantagens e desvantagens dos parâmetros e materiais escolhidos. Com essas informações é possível se trabalhar com maior segurança e conhecimento do equipamento utilizado, as limitações não serão discutidas nesta monografia, mas as características essenciais encontram-se na Tabela 2.1.

O último trabalho referenciado realiza um estudo sobre a trajetória de uma perna do exoesqueleto, utilizando o método de coleta de informações sobre a trajetória de uma perna saudável através de MPUs e a partir disso gera uma contra-parte no protótipo. Com essa base se tem informações relevantes sobre aspectos do próprio colete, por exemplo o método de acionamento de motores, como também se tem um ponto de partida para a aplicação em uma marcha horizontal, como as angulações do joelho e do quadril. O autor fez a sugestão de utilizar uma rede de micro-controladores composta por um Arduino UNO para cada motor e um Arduino Mega para controlar os UNOs, O MEGA realizaria o controle, enquanto os UNOs executariam e receberiam os dados dos motores, porém isso será alterado na seção 4.1.

Os trabalhos anteriores podem ser compreendidos como etapas prévias do projeto, porém outros trabalhos são fundamentais para que se possa continuar o desenvolvimento. Na UnB no *campus* do Gama foi construída uma interface para o acionamento dos motores (RODRIGUES, 2017), uma bancada de testes que garante uma maior segurança na hora de testar com o conjunto inteiro, a maior parte dos testes iniciais serão realizados nesta bancada, Figuras 2.1 e 2.2. Assim como um estudo da análise dinâmica de exoesqueleto foi realizada (ROBBI, 2018) no campus Darcy Ribeiro da UnB.

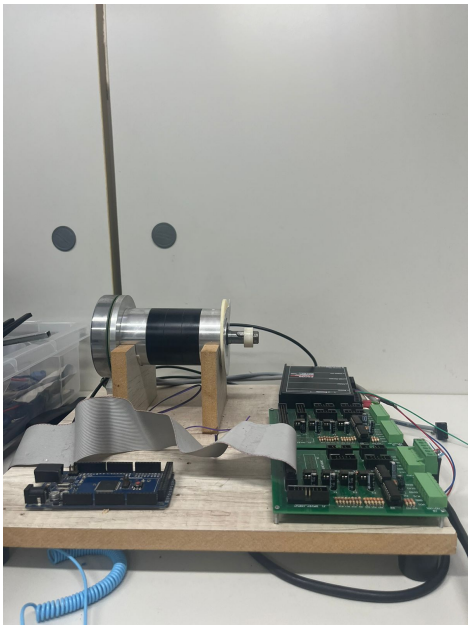


Figura 2.1 – Bancada de Testes - Visão Lateral

Fonte: Próprio Autor

Figura 2.2 – Bancada de Testes - Visão Frontal

Fonte: Próprio Autor

2.1.1.1 Projeto EMA

Na Universidade de Brasília também se encontram projetos similares que não se baseiam somente em teses de graduação, mestrado ou doutorado, o projeto EMA, Figura 2.3 é um dos exemplos que a universidade possui de trabalhos que utilizam da robótica aplicada na área da saúde. Como explicitado no site do projeto (PROJECT EMA, s.d.) "Nós desenvolvemos tecnologia para aperfeiçoar a reabilitação de pessoas com deficiências motoras. Em especial, nosso trabalho foca no uso da estimulação elétrica para pessoas com lesões e doenças que afetam o sistema nervoso.".

O projeto EMA utiliza de eletroestimulação nas pernas para que estas se mexam e possam realizar o ato de pedalar, mesmo o método de acionamento sendo de uma maneira distinta da proposta nesta monografia, ainda é um trabalho que serve de referência e estimulação. O projeto EMA teve sua primeira demonstração em 2015, quando apresentarem um triciclo adaptado com o fito de pessoas com paraplegia pedalarem com suas respectivas pernas, sendo que até os dias atuais o projeto segue na UnB sendo cada vez mais aprimorado.

Figura 2.3 – Projeto EMA

Fonte: (PROJECT EMA, s.d.)

2.1.2 Trabalhos em nível nacional

Projetos fora da UnB no contexto nacional também serão tidos como inspirações para o desenvolvimento final do projeto. Ainda em Brasília, no UniCEUB foi desenvolvido um colete exoesqueleto para membros inferiores e feito o seu controle com um controlador PID (NASCIMENTO, 2022). Adicionalmente, na Universidade Estadual de Campinas foi desenvolvido uma tese de doutorado sobre a utilização de Controle Híbrido para um exoesqueleto robótico de membro inferior com 6 graus de liberdade, um trabalho que apresenta a integração de estratégias de controle, modelagens cinemáticas e outros conteúdos os quais são imprescindíveis para o desenvolvimento do controle de caminhada robótica, tal como a geração de trajetória (CÓRDOBA, 2022).

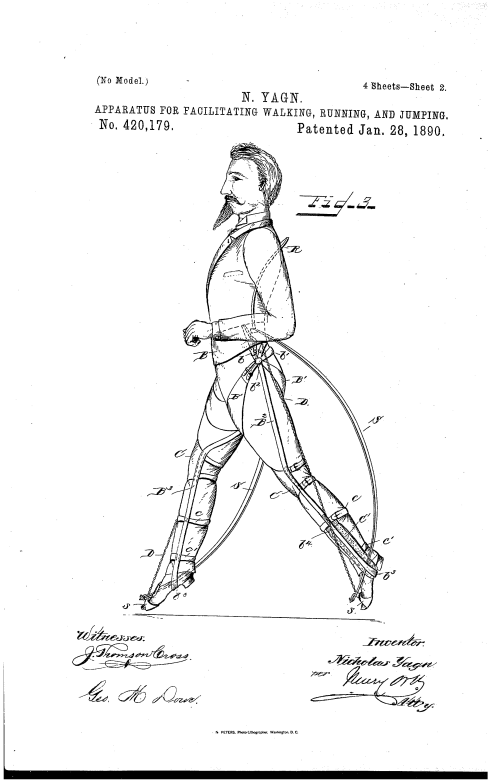
Na USP foi desenvolvido um exoesqueleto de membros inferiores para auxiliar pessoas atingidas por um AVC (JORNAL DA USP, s.d.), no vídeo disponibilizado é possível ver uma pessoa caminhando na esteira a uma velocidade padrão. O Exoesqueleto pesa 11kg e é fixado ao humano e apresenta sensores de força para a avaliação de testes, um sapato adaptado com o mesmo intuito e um conjunto de atuadores que auxiliam o movimento, ressalva-se que o paciente que utilizou do equipamento é uma pessoa saudável e não uma sobrevivente de AVC.

Um estudo sobre a eficácia da utilização deste tipo de tecnologia em pessoas que sofreram AVE (SILVA; MOURA; SOUSA, 2022) foi realizado, sendo que este concluiu que a ferramenta exoesqueleto é eficaz na reabilitação, porém suas análises não apontaram melhorias ao tratamento fisioterápico padrão, com a ressalva de que por ser uma tecnologia ainda em desenvolvimento, os estudos foram escassos para a pesquisa, como também é uma tecnologia de alto valor nos dias atuais. Com esse texto, é conclusivo que os estudos nessa área de reabilitação com robótica se mostra com um grande potencial tanto para área da saúde quanto tecnologia.

2.2 Trabalhos exteriores

Com base no artigo (DOLLAR; HERR, 2008) foi realizada uma pesquisa sobre exoesqueletos e tecnologias acerca dos exoesqueletos de membro inferiores.Esta seção tem como objetivo apreciar o estado da arte das órteses ativas, termo tipicamente usado para a descrição de aparelhos usados para o aumento da habilidade locomoção de uma pessoa que sofra de patologia nas pernas.

Desde o primeiro exoesqueleto em 1890 (NICHOLAS YAGN, s.d.) até os dias atuais, as estruturas de exoesqueleto vêm sendo aprimoradas, como mostram as figuras Figura 2.4 e Figura 2.5. Como pode ser visto, existe uma evolução de modelo, mas com características similares. Mesmo que com objetivos distintos, seja para uma maior capacidade física do usuário, como carregar mais peso de forma segura ou pular mais alto, ou seja para uma reabilitação fisioterapêutica, os exoesqueletos fazem com que a área da biomecânica do movimento humano venha sendo expandida de maneira considerável nas últimas décadas.


(No Model.) 4 Sheets—Sheet 2.

N. YAGN.

APPARATUS FOR FACILITATING WALKING, RUNNING, AND JUMPING.

No. 420,179. Patented Jan. 28, 1890.

Fig. 3.

Witnesses. Inventor.

Figura 2.4 – Primeiro Modelo de Exoesqueleto

Fonte: (NICHOLAS YAGN, s.d.)

Figura 2.5 – Exoesqueleto HAL

Fonte: (CYBERDYNE, s.d.)

Os exoesqueletos como aparato de aumento de performance de pessoas que não apresentam a dificuldade de realizar os movimentos podem ser citados pois auxiliam nos estudos, mas neste projeto o foco será nas órteses ativas. Pode-se encontrar menção sobre esse tipo de trabalho desde 1935 (GEORGE L COBB, s.d.), Figura 2.6. Foram criadas também

órteses de corpo inteiro, como o exoesqueleto produzido pela *wandercraft* Atalante, Figura 2.7, que foca em pacientes com paraplegia e órteses com ativamento modular tal como o trabalho realizado pelo MIT ((BLAYA; HERR, 2004)), Figura 2.8), com *Ankle-Foot-Orthoses*, Órtese-Calcanhar-Pé, que é consideravelmente menor que as outras apresentadas, mas não menos eficiente, vide a Figura 2.8.

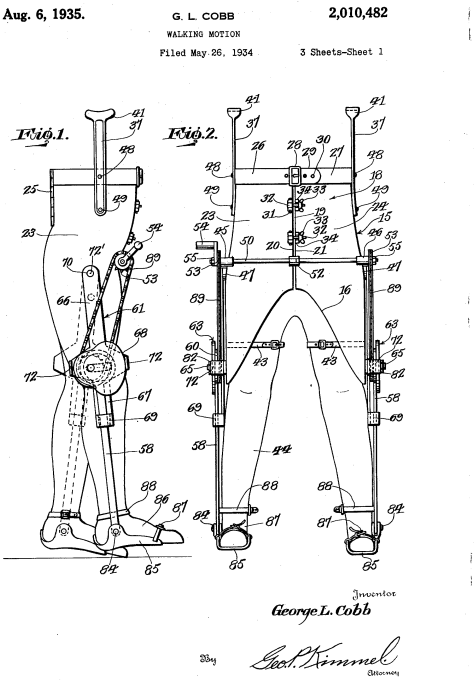


Figura 2.6 – Projeto de Órtese Ativa

Fonte: (GEORGE L COBB, s.d.)

Figura 2.7 – Exoesqueleto Atalante

Fonte: (WANDERCRAFT, s.d.)

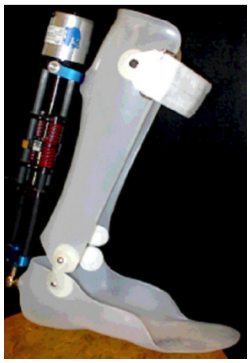


Figura 2.8 – *Ankle Foot Orthoses*

Fonte: (BLAYA; HERR, 2004)

O fato de que a limitações das pessoas que necessitariam das Órteses para auxiliar sua locomoção é característica de cada um, existem diversas limitações distintas e mesmo

as lesões similares ou ocasionadas de maneira semelhante, existem graus distintos graus para cada uma. Este aspecto faz com que o desenvolvimento de aparelhos de uso geral seja dificultado. A maioria dos exoesqueletos usados em casos clínicos se encontra em uso dentro de ambientes de testes ou controlados.

O avanço tecnológico com a finalidade de se aumentar a portabilidade destes aparatos é de alto valor para o mercado e para a sociedade, que desejam cada vez mais dispositivos práticos e acessíveis. Os avanços em motores DC, baterias, biomecânica do movimento humano e áreas relacionadas, incluindo controladores, têm impulsionado o desenvolvimento contínuo de tecnologias capazes de replicar movimentos semelhantes aos humanos. Esse progresso constante alimenta a expectativa crescente de que indivíduos com dificuldades de locomoção possam, cada vez mais, ter acesso a locais que anteriormente eram inacessíveis ou demandavam esforço e tempo consideráveis para serem alcançados.

2.2.1 Metodologias de controle

O controle de robôs bípedes é algo de uma elevada complexidade devido ao fato de ser um sistema instável, não-linear e múltiplas entradas e saídas (MIMO) (CHEN et al., 2020). Diversos métodos de controle foram testados para solucionar o desafio do caminhar humano, tais como *Zero moment point*, *Spring-loaded inverted pendulum*, *Hybrid zero dynamics*, entre outros.

O controle de caminhada avalia a angulação das juntas no plano sagital, como também o movimento de levantamento ou agachamento e a subida e descida de escada, entretanto quando se analisa movimentos no espaço 3D, como a realização de curvas, o movimento já passa a integrar mais planos, sendo a maioria dos trabalhos avaliados no plano sagital e com caminhada de sentido horizontal em um plano não-inclinado, todavia ressalva-se trabalho como (ZOU et al., 2019) que realiza o projeto em plano inclinado. O projeto efetuado neste trabalho não realiza o controle de equilíbrio do exoesqueleto, o usuário do protótipo deve realizá-lo por meio de muletas ou outro tipo de auxílio que ofereça uma maior estabilidade.

2.3 Especificações do projeto

É de suma importância conhecer os limites que foram usados no desenvolvimento do exoesqueleto, não somente em relação ao sistema de atuação, mas também as características que foram usadas para calcular os torques previstos e etc. Com o trabalho (FREIRE, 2019) foi separado um conjunto de dados iniciais sobre o exoesqueleto a ser trabalhado, Figuras 2.10 e 2.9, o motor utilizado no mesmo, Figura 2.11, e o redutor acoplado ao motor, Figura 2.12:

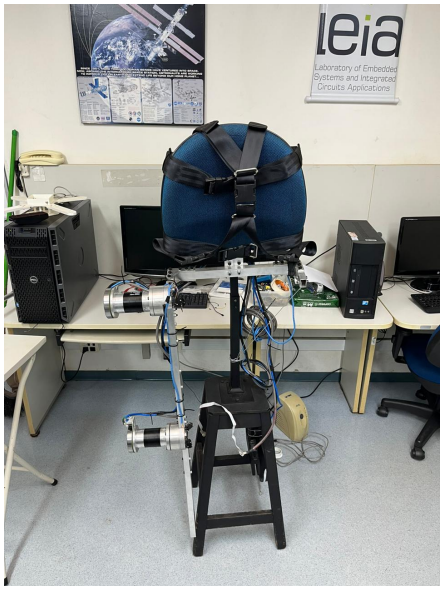


Figura 2.9 – *Exosuit* Vista Frontal

Fonte: Próprio autor

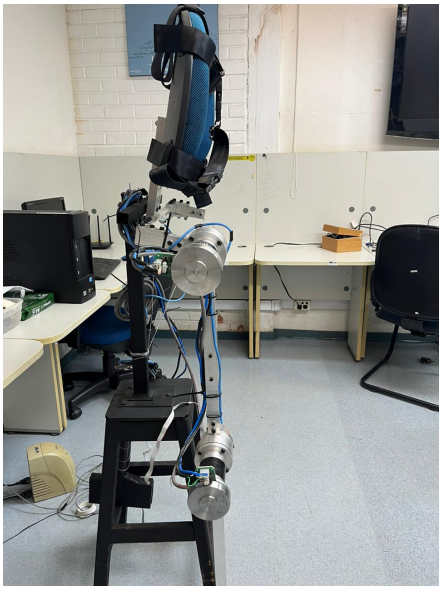


Figura 2.10 – *Exosuit* Vista Lateral

Fonte: Próprio autor

Característica	Caso Crítico
Massa do Usuário	100 kg
Altura do Usuário	1.90 m
Velocidade de Marcha	0.3 m/s
Comprimento de Passada	0.53 m
Tempo de Passada	1.76 s

Tabela 2.1 – Casos Antropométricos e de Marcha Críticos

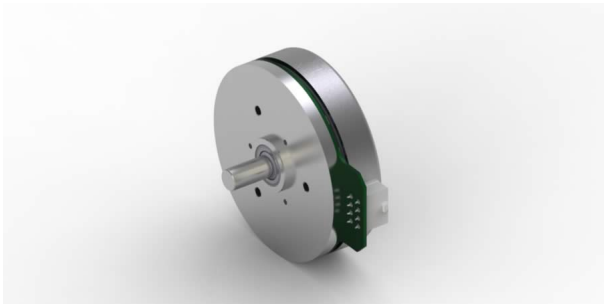


Figura 2.11 – Motor Flat EC 90 (323772)

Fonte: (MAXON MOTOR, s.d.[a])

[-1] Medidas Tiradas no centro do motor, no caso do tornozelo é a base do elo
[0] Valores negativos sinalizam uma rotação no sentido horário

Características do Motor	Dados
Tensão Nominal	24 V
Corrente Nominal	6.06 A
Corrente em Vazio	544 mA
Torque Nominal	444,4 mNm
Velocidade Nominal	2590 rpm
Velocidade Máxima	5000 rpm
Eficiência Máxima	84%
Peso	600 g

Tabela 2.2 – Dados técnicos do Motor (MAXON MOTOR, 2017)

Figura 2.12 – Redutor Planetário GP 52C (223095)

Fonte: (MAXON MOTOR, s.d.[b])

Características do Redutor	Dados
Redução	100:1[1]
Redução Absoluta	338/3
Massa de Inércia	17.3 gcm^2
Diâmetro máximo do eixo do motor	10 mm
Torque Máximo Suportado pelo eixo	50 Nm
Eficiência Máxima	70
Peso	1540 g

Tabela 2.3 – Dados técnicos do Redutor Planetário (MAXON MOTOR, 2021)

ELO	Referência	Medida
1	Quadril ao Joelho	50,0 cm
2	Joelho ao Tornozelo	45,5 cm

Tabela 2.4 – Dimensões dos Elos[2]

O controle de cada um dos motores foi sugerido a utilizaçãode Arduinos UNOs, um para cada motor, e um Arduino Mega que realizaria o controle dos outro microcontroladores,

[1] Os *Datasheet* do motor diz ser 113:1, mas experimentalmente foi verificado que a redução é de realmente 100:1

[2] Medidas Tiradas no centro do motor, no caso do tornozelo é a base do elo

ou seja, seria realizada uma rede de Arduinos para o controle geral do projeto. Esse método foi o proposto pelo trabalho predecessor a este, (PERES, 2023), porém será utilizado um ESP32 como microcontrolador para realizar o trabalho que seria feito pela rede de Arduinos. O controle de posição se dará por meio dos sensores de efeito Hall que se encontram no motor, cada sensor apresenta 12 pares de polos, então com o redutor de 100:1 acoplado, a resolução do motor é alta.

2.4 Resumo do capítulo

O estudo de Exoesqueleto de membro inferiores apresenta diversas ramificações de projetos finais, podendo ser maior capacidade humana a fins militares, como também para facilitar em trabalhos que exijam mais força física, mas também pode ser focado para reabilitação e acessibilidade. Existem diversos protótipos com diferentes resultados, vestimentas de corpo inteiro, vestimentas inferiores com suporte de lombar ou até mesmo somente nas regiões debilitadas como um tornozelo.

Mesmo com diferentes modelos e objetivos finais, o desafio de reproduzir a biomecânica dos movimentos humanos é um dos pontos chave para o avanço nesse tipo de estudo. Existem diferentes técnicas de controle que podem ser aplicadas, sendo necessário avaliar o caso em que o projeto será empregado e os requisitos e limitações dos componentes e materiais de cada projeto.

3 Revisão Teórica

3.1 Marcha humana

Traduzindo o texto de Laurie Anderson "A cada passo, você se inclina ligeiramente para frente /E então se segura para não cair /Uma e outra vez, você está caindo /E então se segurando para não cair". O movimento humano é um dos dois movimentos de marcha humana mais comuns, o outro sendo a corrida. Este movimento de caminhada é delimitado por dois eventos consecutivos de contato de pé com o chão, analisado sobre uma mesma perna. De acordo com o livo sobre a biofísica do movimento humano (ABERNETHY et al., 2013), o ciclo da marcha humana pode ser definido como o tempo levado entre o contado de um calcanhar com a superfície até o o momento que este mesmo calcanhar acerta novamente a superfície para começar um novo ciclo, uma caminhada de jovens adultos saudáveis é praticamente simétrica entre as pernas, sendo que geralmente um ciclo de marcha dividido entre duas etapas, a etapa de contato (*stance*) e a fase de balanço (*swing*), como representado na Figura 3.13.

A fase de balanço é definida no instante que os dedos de um pé perdem contato com a superfície até o momento em que o calcanhar do mesmo pé renova o contato com a superfície, essa etapa pode ser dividida em três fases menores, como mostrado na Figura 3.14. Enquanto a fase de contato pode ser dividida em fases menores, sendo sua divisão realizada em etapas de suporte duplo, que ocupam aproximadamente 20% do ciclo, assim como a de suporte único ocupam 40%. Ao todo na marcha, aproximadamente 60% do ciclo é durante a fase de contato com o chão e por 40% do ciclo de marcha seria na etapa de balanço.

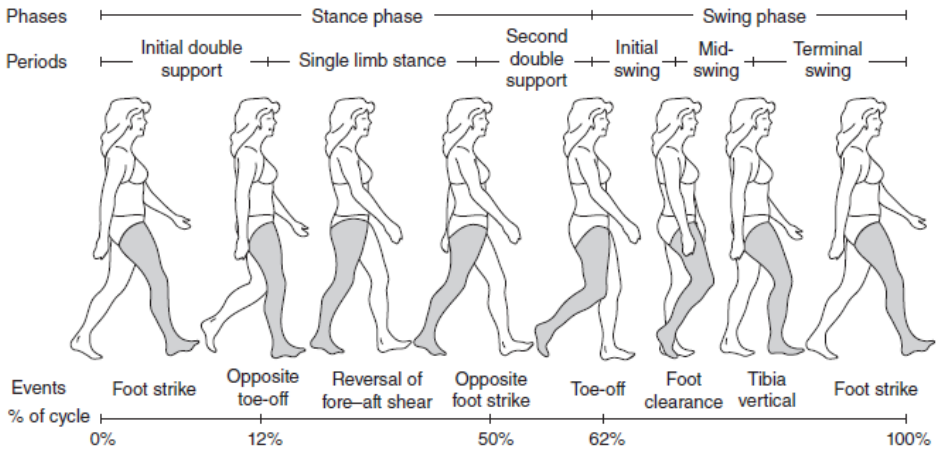


Figura 3.13 – Ciclo da Marcha Humana

Fonte: (ABERNETHY et al., 2013)

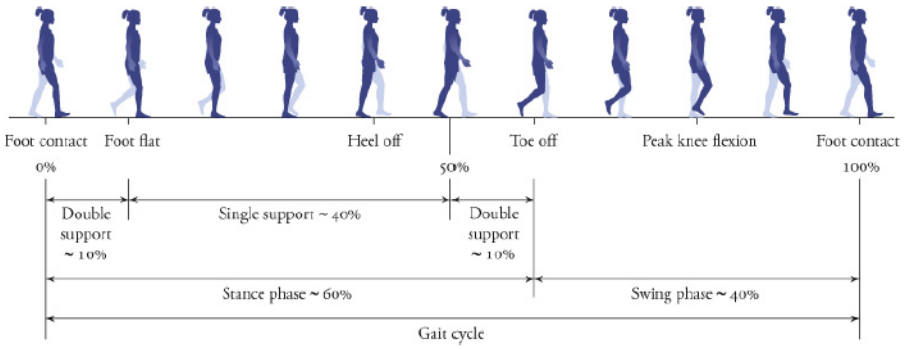


Figura 3.14 – Ciclo da Marcha Humana

Fonte: (UCHIDA; DELP, 2020)

Resumidamente, o caminhar é uma ação gerada devido a reação das forças que o pé aplica no solo. A medição das forças de reação são importantes para o estudo de como o centro de massa de uma pessoa se acelera em cada instante de tempo. Com o auxílio das figuras 3.15 e 3.16 a seguir, é possível ter uma ideia sobre como ocorre a ação e reação de forças em um ciclo de marcha da caminhada.

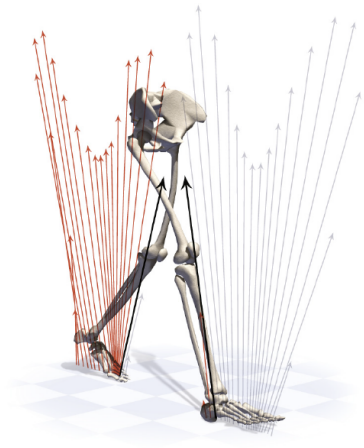


Figura 3.15 – Força Resultante das Reações ao Caminhar

Fonte: (UCHIDA; DELP, 2020)

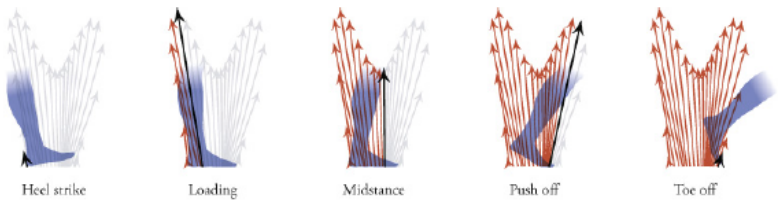


Figura 3.16 – Forças de Reação ao Caminhar

Fonte: (UCHIDA; DELP, 2020)

A velocidade de marcha de cada indivíduo varia a depender de aspectos como sua altura, peso, se a pessoa é fisicamente ativa ou não, mas adultos saudáveis tendem a caminhar em velocidades de 1.2m/s à 1.4m/s em um percurso nivelado, a uma cadência de dois passos por segundo e uma distância de 0.6m à 0.7m de passada.

O caminhar humano é um movimento de um conjunto global de juntas de corpo e músculos, mas para o entendimento total da marcha, é necessário analisar o movimento de cada junta de maneira individual. O corpo humano pode ser dividido em três planos distintos, o plano frontal, o plano transversal e o plano sagital o qual é o mais utilizado em estudos do movimento de marcha. A Figura 3.17 mostra como é realizada essa divisão. Essa análise ainda vai mais profunda pois em cada um desses planos o corpo pode realizar um conjunto de movimentos únicos e complexos, como mostrado nas Figuras 3.18 e 3.19.

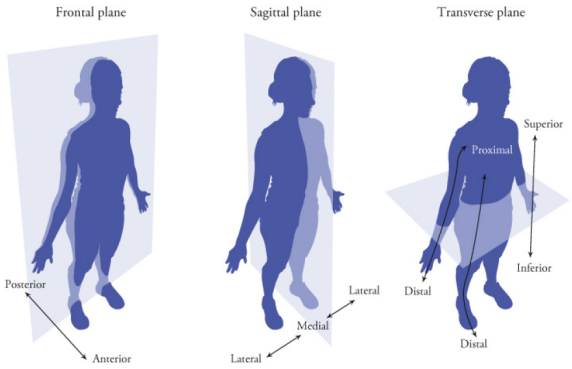


Figura 3.17 – Divisão dos planos anatômicos de um humano

Fonte: (UCHIDA; DELP, 2020)

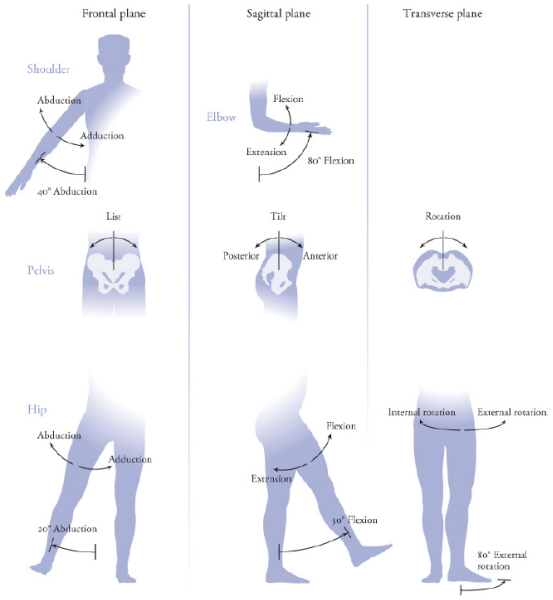


Figura 3.18 – Movimentos de diferentes partes do corpo nos 3 planos distintos

Fonte: (UCHIDA; DELP, 2020)

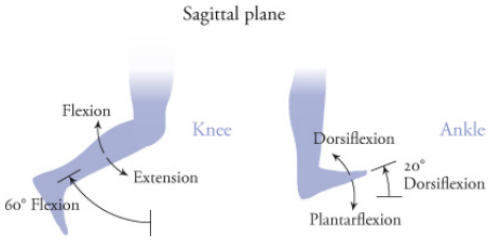


Figura 3.19 – Movimento do tornozelo no plano Sagital

Fonte: (UCHIDA; DELP, 2020)

O estudo realizado com prioridade no plano sagital deixa algumas informações de fora, para uma boa compreensão da marcha é necessário entender que existem outros movimentos que auxiliam e permitem o conjunto completo do movimento, como a ação da pélvis nos planos frontal e transversal, representado na Figura 3.20.

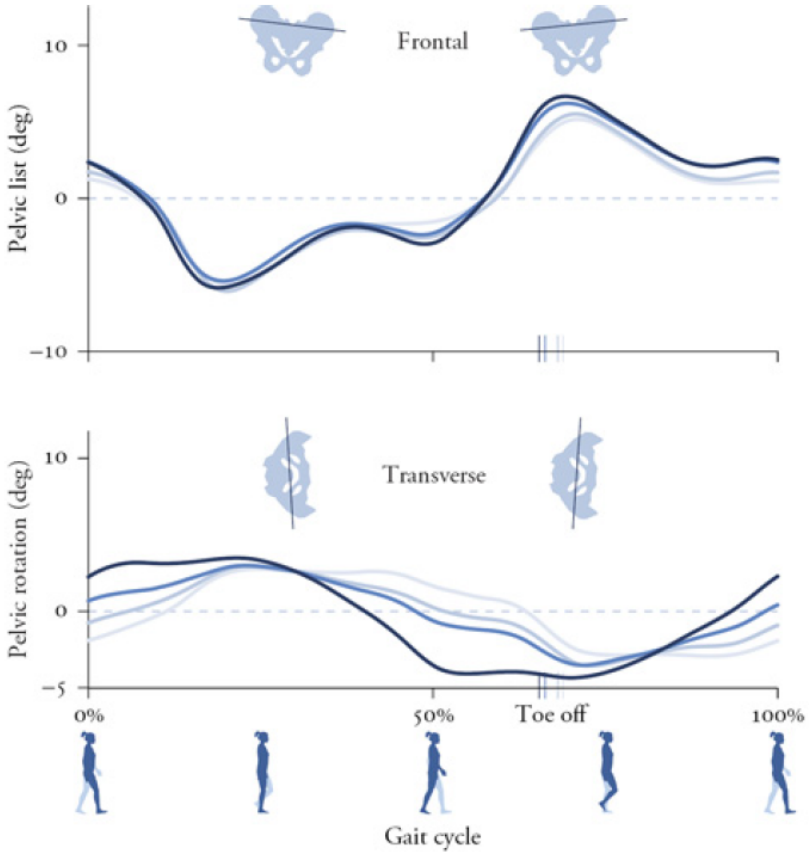


Figura 3.20 – Movimentação da Pélvis na marcha humana nos planos frontal e transversal

Fonte: (UCHIDA; DELP, 2020)

3.2 Movimentação de um robô

3.2.1 Transformações Homogêneas

A movimentação de um robô em ambiente industrial é geralmente necessário ter o posicionamento e orientação da sua ferramenta terminal, última parte do robô, a parte a qual realiza a tarefa, tal como uma garra ou uma tocha de soldagem, a depender da função que será empregada. Com isso é preciso que se realize uma conversão para plano cartesiano, pois um robô é geralmente controlado em relação as coordenadas de suas juntas, muitas vezes sendo a angulação entro os elos, pois é dependente de se a junta é do tipo rotacional ou translacional.

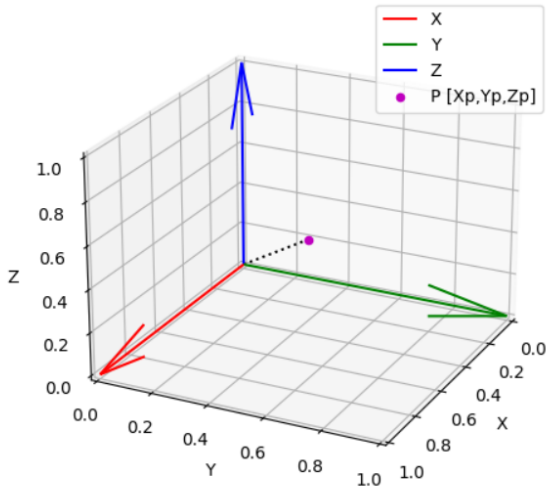


Figura 3.21 – Um ponto no Espaço Tridimensional

Fonte: Próprio Autor

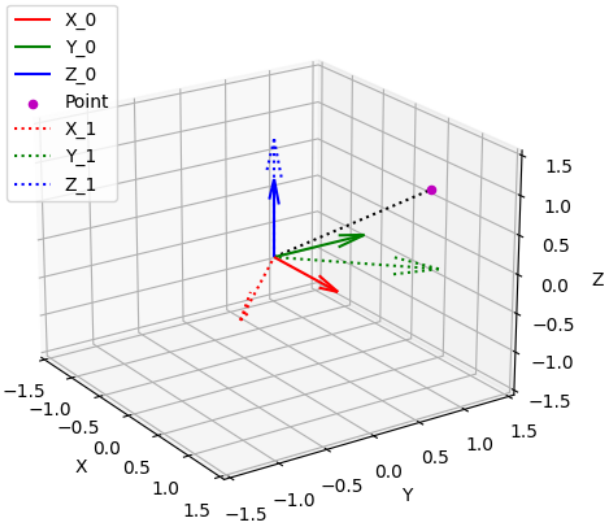


Figura 3.22 – Sistema Rotacionado no Eixo Z

Fonte: Próprio Autor

Como visto na Figura 3.21, um ponto no espaço pode ser representado por um conjunto de coordenadas tal que $p_i = [p_{xi} \;\; p_{yi} \;\; p_{zi}]^T$, sendo i um número inteiro representando o instante em que o movimento é analisado $i = 0,1,2,...,n$. Diversas vezes é preferível olhar para o ponto p de outro ângulo, isto é, usar um sistema de coordenadas diferentes para descrever o mesmo ponto, então pode ser aplicada uma matriz de Rotação R^i_{i-1}, Matriz de rotação do sistema i em relação ao sistema $i-1$. No Caso da imagem 3.22, o sistema foi rotacionado em torno do eixo Z

As Matrizes de Rotação entre dois sistemas em torno dos eixos x, y e z com uma angulação de θ são descritas por:

$$
\begin{aligned}
R_{x,\theta} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \\
R_{y,\theta} &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \\
R_{z,\theta} &= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{3.1}
$$

É necessário se atentar também que a ordem em que as matrizes são aplicadas é de extrema importância, pois o resultado final pode ser alterado, como visto com as figuras 3.23 e 3.24 a seguir:

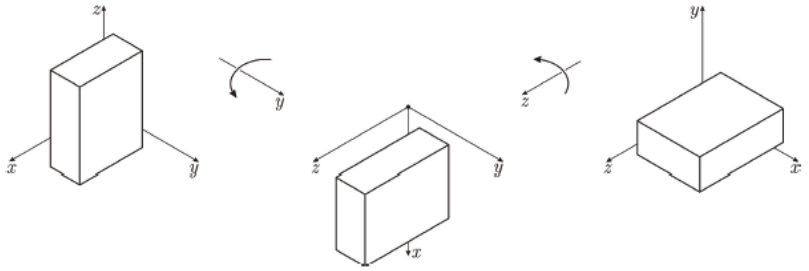


Figura 3.23 – Rotação 1

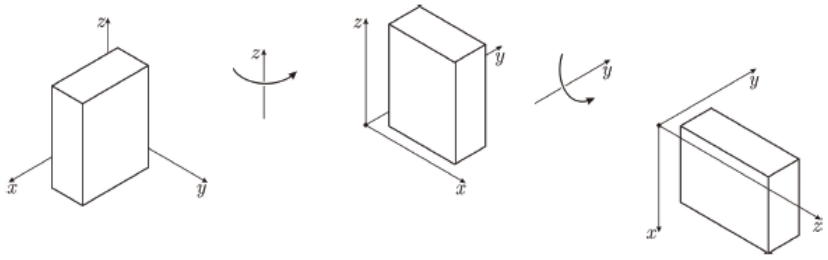


Figura 3.24 – Rotação 2

Figura 3.25 – Matrizes de Rotação aplicadas em ordens distintas

Fonte: Notas de aula da disciplina Robótica Industrial

A matriz de transformação homogênea é um conjunto da matriz de rotação com o vetor referente a translação do sistema e uma linha padrão, então a matriz de transformação homogênea é definida por:

$$\mathbf{H}_1^0 = \begin{bmatrix} \mathbf{R}_1^0 & \mathbf{d}_1^0 \\ 0 & 1 \end{bmatrix},$$
$$\mathbf{H}_1^0 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Note que r_{nm} representa o valor da matriz de rotação, então é dependente do eixo em questão, enquanto a última linha da matriz é sempre [0 0 0 1].

A matriz de transformação homogênea pode ser obtida a partir da composição de diversas outras matrizes de transformações homogêneas, supondo uma transformação de um número m ao número n, com $n > m$, pode-se definir que:

$$\mathbf{H}_m^n = \mathbf{H}_m^{m+1}\mathbf{H}_{m+1}^{m+2} \dots \mathbf{H}_{i-1}^{i} \dots \mathbf{H}_{n-2}^{n-1}\mathbf{H}_{n-1}^{n} \tag{3.3}$$

3.2.2 Cinemática de manipuladores

A questão da cinemática é descrever o movimentos de manipuladores sem a consideração de forças e torques que causariam a dinâmica, a cinemática é uma descrição geométrica. Ela pode ser dividida em duas partes, a cinemática direta e inversa, a primeira determina a posição e orientação do efetuador terminal dadas as variáveis das juntas do robô, enquanto a cinemática inversa, como o próprio nome sugere, é o oposto, com a posição e orientação do efetuador terminal, são determinadas os valores das variáveis das juntas.(MARK W. SPONG SETH HUTCHINSON, 2005)

3.2.2.1 Cinemática direta

A cinemática direta de manipuladores robóticos é realizada, de maneira geral, usando a convenção de Denavit-Hartenberg. A utilização de uma modelo padrão facilita o desenvolvimento e compreensão de projetos. Utilizando esse método é possível realizar a geração de trajetórias e de identificação de erros a partir das coordenados do efetuador terminal do robô.

Essa norma é basicamente onde uma transformação homogênea A_i é um conjunto de multiplicação de 4 transformações básicas $A_i = T_{z,d}R_{z,\theta_i}T_{x,a}R_{x,\alpha_i}$.

$$A_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & d_{x_i}\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & d_{x_i}\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_{z_i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

O método desenvolvido por DH é um algoritmo, então o processo se baseia nos seguintes passos:

1. Localize e Identifique os eixos das juntas z_0,...,z_{n-1}
2. Estabeleça a base. Coloque a origem no eixo z_0, os eixos x_0 e y_0 serão escolhidos de maneira conveniente de modo que se mantenha a regra da mão direita
    - Para i = 1, ..., n-1, repita os passos 3 a 5
3. Localize a origem O_i onde a normal de z_i e z_{i-1} intersectam z_i. Caso z_i intersecte $z_i - 1$, O_i localizará neste ponto, caso sejam paralelos, defina o_i em qualquer ponto conveniente de z_i
4. Estabeleça x_i juntamente com a normal entre z_{i-1} e z_i ao longo de o_i,ou na direção normal para o plano $z_{i-1} - z_i$, caso o plano de z_{i-1} e z_i intersectem.
5. Estabeleça y_i seguindo a regra da mão direita
6. Estabeleça a coordenada final do efetuador terminal $O_n x_n y_n z_n$. Assumindo que a n-ésima junta é de revolução, coloque $z_n = a$ juntamente da direção z_{i-1}. Estabeleça O_n ao longo de z_n, preferencialmente no centro da garra/ferramenta do manipulador. Defina $y_n = s$ na direção de fechamento da garra e defina $x_n = n$ como $s \times a$. Caso a ferramenta não seja uma simples garra, defina x_n e y_n de maneira convencional a formar a regra da mão direita
7. Crie a tabela de DH com os parâmetros $a_i, d_i, \alpha_i, \theta_i$
    - a_i = distância ao longo de x_i de O_i até a intersecção dos eixos x_i e z_{i-1}
    - d_i = distância ao longo de z_{i-1} de O_{i-1} até a intersecção dos eixos x_i e z_{i-1}. d_i é variável no caso da junta i ser prismática
    - α_i = ângulo entre z_{i-1} e z_i medido sobre x_i
    - θ_i = ângulo entre x_{i-1} e x_i medido sobre z_{i-1}. θ_i é variável no caso da junta i ser de revolução
8. Construa a Matriz de transformação homogênea A_i com os parâmetros do passo anterior, como a equação 3.4
9. Construa $T_n^0 = A_1 ... A_n$. Com isso se terá a posição e orientação da ferramenta expressa nas coordenadas da base.

3.2.2.2 Cinemática inversa

Em contrapartida com o tópico anterior, o problema agora é encontrar as variáveis de juntas em termos da posição e angulação do efetuador terminal. A cinemática inversa, em geral, é mais complexa que a direta pelo fato de não ter um algoritmo explícito como o de DH. Existem duas maneiras principais de se resolver o esse tipo de problema, o método analítico e o geométrico.

- Método Analítico

  Genericamente o problema da cinemática inversa pode ser descrito como a seguir.

  Dado uma matriz de transformação homogênea:

$$H = \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_x \\ r_{31} & r_{31} & r_{33} & d_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

  Ache a solução $T_n^0(q_1,...,q_n) = H$ onde $T_n^0(q_1,...,q_n) = A_1(q_1)...A_n(q_n)$. O resultado da equação resultará em 12 equações não lineares, mas devido algumas especificidades, tal como cada coluna da matriz de Rotação R deve ser um vetor unitário e suas colunas formam um conjunto ortonormal, colocam restrições sobre as equações. Um exemplo de uma solução que será mostrado pode ser encontrado no *Example 3.7* em (MARK W. SPONG SETH HUTCHINSON, 2005), que busca as variáveis de junta $\theta_1,\theta_2,d_3,\theta_4,\theta_5,\theta_6$ da matriz[1]:

$$H = \begin{bmatrix} 0 & 1 & 0 & -0.154 \\ 0 & 0 & 1 & 0.763 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

  Para se ter as variáveis de junta, é necessário se resolver o seguinte conjunto de equações:

$$c_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_5c_6] - s_1(s_4c_5c_6 + c_4s_6) = 0$$

$$s_1[c_2(c_4c_5c_6 - s_4s_6) - s_2s_6] + c_1(s_4c_5c_6 + c_4s_6) = 0$$

$$-s_2(c_4c_5c_6 - s_4s_6) - c_3s_5c_6 = 1$$

$$c_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] - s_1(-s_4c_5s_6 - c_4c_6) = 1$$

$$s_1[-c_2(c_4c_5s_6 + s_4c_6) + s_2s_5s_6] + c_1(s_4c_5s_6) = 0$$

[1] Será usada a ortografia do livro que ao invés de $\cos\theta$ e $\sin\theta$ se usa c e s para facilitar a leitura

$$s_2(c_4c_5s_6 + s_4c_6) + c_2s_5s_6 = 0$$

$$c_1(c_2c_4s_5 + s_2c_5) - s_1s_4s_5 = 0$$

$$s_1(c_2c_4s_5 + s_2c_5) + c_1s_4s_5 = 1$$

$$-s_2c_4s_5 + c_2c_5 = 0$$

$$c_1s_2d_3 - s_1d_2 + d_6(c_1c_2c_4s_5 + c_1c_5s_2 - s_1s_4s_5) = -0.154$$

$$s_1s_2d_3 + c_1d_2 + d_6(c_1s_4s_5 + c_2c_4s_1s_5 + c_5s_1s_2) = 0.763$$

$$c_2d_3 + d_6(c_2c_5 - c_4s_2s_5) = 0$$

Com os valores da tabela de DH que o livro oferece onde $d_2 = 0.154$ e $d_6 = 0.263$, uma solução para o conjunto de equações pode ser:

$$\theta_1 = \pi/2, \theta_2 = \pi/2, d_3 = 0.5, \theta_4 = \pi/2, \theta_5 = 0, \theta_6 = \pi/2,$$

Além da solução apresentada para cinemática inversa, também pode se resolver de uma maneira mais simplificada, para diversos casos, pois é uma solução mais direta do problema. a solução pelo método geométrico consiste em decompor o manipulador em planos e resolver problemas de trigonometria básica. A complexidade dos problemas de cinemática inversa se aumenta quanto mais parâmetros não nulos existem nos links, mas uma boa parte de manipuladores apresentam os valores de α_i e/ou d_i iguais a zero, ou valores como $\pm\pi/2$, com isso a solução geométrica é bem facilitada.

- Método Geométrico

  O método geométrico é um método mais visual, a orientação da Figura 3.26 a qual será usada como base para os cálculos está com o eixo Y invertido ao visto usualmente pelo fato de que para as contas o pé do exoesqueleto é o efetuador terminal do robô. Ressalva-se que θ_1 é o ângulo o que o quadril rotacionou, no caso da imagem é uma rotação no sentido anti-horário e θ_2 é o ângulo que o joelho rotacionou no sentido horário no que está representado, assumindo que o robô partiu de uma posição paralela ao eixo Y, a perna do exoesqueleto estaria perpendicular para baixo.

  Primeiramente θ_2 será calculado, nota-se que α é o ângulo suplementar ao que será calculado, sendo que α é o ângulo que enxerga a "linha"que liga o efetuador terminal à origem e se tem como lados os elos do próprio manipulador.

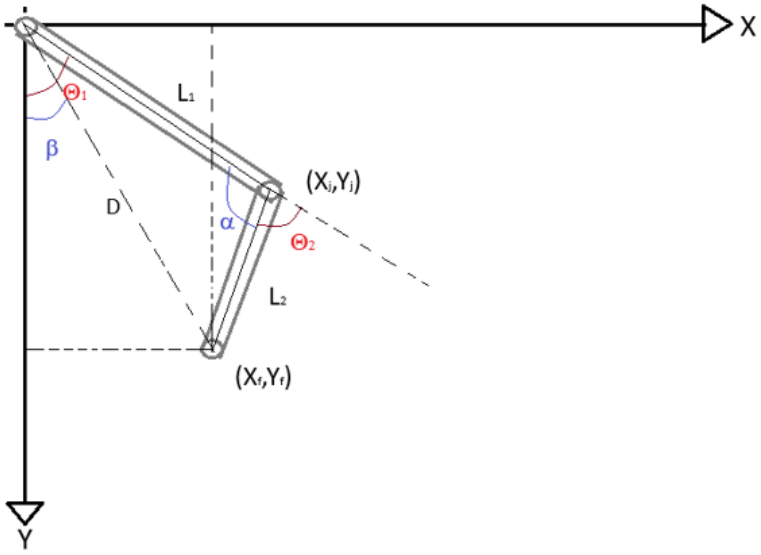


Figura 3.26 – Cinemática Inversa Método Geométrico

Fonte: Próprio Autor

$$
\begin{aligned}
D^2 &= L_1^2 + L_2^2 - 2L_1L_2\cos(\alpha) \\
\cos(\alpha) &= \frac{L_1^2 + L_2^2 - D^2}{2L_1L_2} \\
\cos(180° - \theta_2) &= \frac{L_1^2 + L_2^2 - D^2}{2L_1L_2} \\
-\cos(\theta_2) &= \frac{D^2 - L_1^2 - L_2^2}{2L_1L_2} \\
\theta_2 &= \arccos\left(\frac{X_f^2 + Y_f^2 - L_1^2 - L_2^2}{2L_1L_2}\right)
\end{aligned}
\tag{3.6}
$$

O passo em seguida é encontrar o ângulo θ_1. nota-se que β é o ângulo que enxerga a distância X que o efetuador terminal está do eixo das assíntotas e se tem como lados o eixo que conecta a origem com o efetuador terminal e o próprio eixo Y, enquanto $\theta_1 - \beta$ é o ângulo composto pelo triângulo composto pelos elos e a diagonal que os "liga"à origem. Primeiramente se calcula o ângulo β para que se possa calcular o ângulo desejado em seguida.

$$
\begin{aligned}
\tan(\beta) &= \frac{X_f}{Y_f} \\
\beta &= \arctan\left(\frac{X_f}{Y_f}\right)
\end{aligned}
\tag{3.7}
$$

$$
\begin{aligned}
L_2^2 &= L_1^2 + D^2 - 2L_1 D\cos(\theta_1 - \beta) \\
\cos(\theta_1 - \beta) &= \frac{D^2 + L_1^2 - L_2^2}{2L_1 D} \\
\theta_1 - \beta &= \arccos\left(\frac{D^2 + L_1^2 - L_2^2}{2L_1 D}\right) \\
\theta_1 &= \arccos\left(\frac{X^2 + Y^2 + L_1^2 - L_2^2}{2L_1 D\sqrt{X^2 + Y^2}}\right) + \arctan\left(\frac{X_f}{Y_f}\right)
\end{aligned}
\tag{3.8}
$$

É importante se destacar que a solução da cinemática inversa não é única, um robô pode apresentar a mesma posição e orientação de seu efetuador terminal, mas com distintas configurações, vide imagem 3.27:

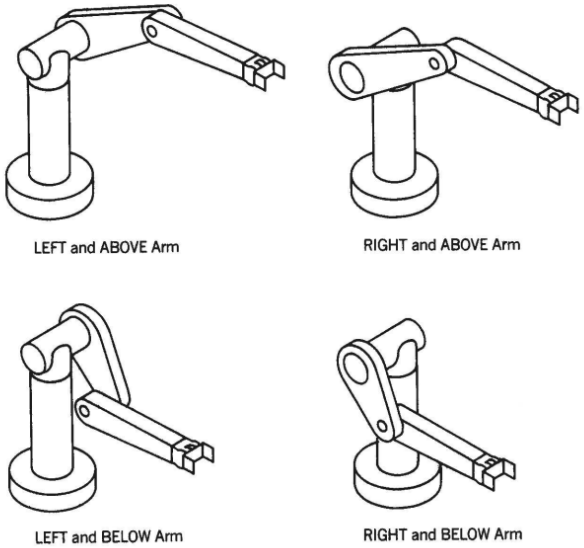


Figura 3.27 – Diferentes configurações para uma mesma posição e orientação final

Fonte: (MARK W. SPONG SETH HUTCHINSON, 2005)

Com o auxílio da *robotics toolbox* (PERER CORKE, s.d.) será mostrado um modelo simplificado e dinâmico do exoesqueleto do projeto com a Figura 3.28

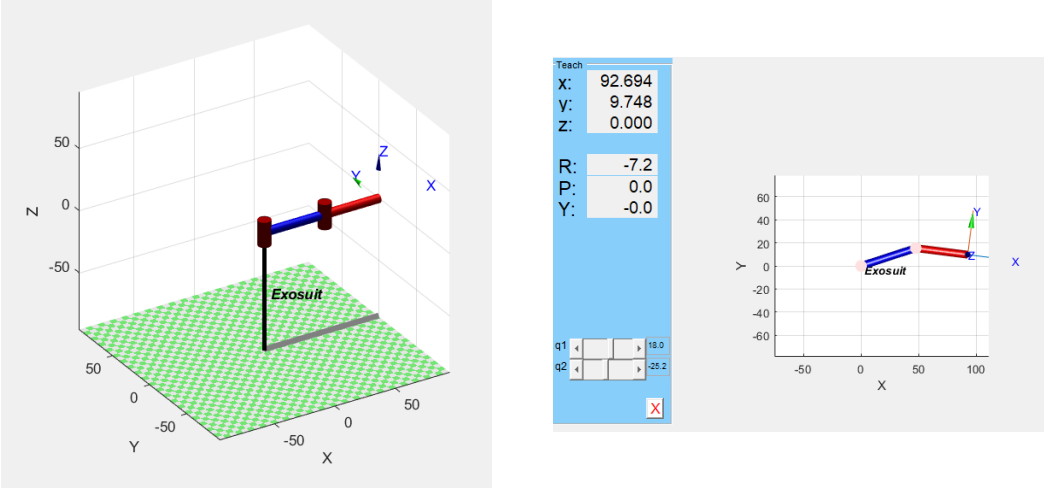


Figura 3.28 – Modelagem da exosuit no *matlab* com auxílio da *robotics toolbox*

Fonte: Próprio Autor

3.3 Controle digital - PID

Os métodos de controle digital são métodos de controle em tempo discreto, com isso, é necessário que se tenha uma base de controladores a tempo contínua, sinais e sistemas em tempo contínuo e discreto assim como uma familiaridade com diagramas de Bode. O método de projeto de controladores PID, representado na Fiugura 3.29 é uma extensão de controladores com avanço e atraso de fase, que é um método baseado na resposta em frequência do sistema. De acordo com (OGATA, 1995), esse método de projeto frequentemente utilizado devido a sua simplicidade, pois as variáveis que precisam ser atentadas são amplitude e fase da onda senoidal gerada a partir da resposta linear e invariante no tempo do sistema.

Com uma função de transferência $G(z)$ é possível se obter sua resposta em frequência realizando a substituição $z = e^{j\omega t}$. Com a resposta em frequência, é possível usar de diferentes métodos para projetar o controlador. A utilização de um diagrama de Bode apresenta vantagens, pois as assíntotas na curva de magnitude em baixas frequências é um indicativo das constantes de erros estacionários, como também as especificações da resposta transiente podem ser convertidas nas de frequência em termos de margem de fase, margem de ganho e largura de banda, sendo que essas informações podem ser obtidas de forma direta e simples do diagrama de Bode. No projeto de um compensador ou controlador digital as vezes é necessários que se atinja certos valores em termos de fase e ganho, o que é lido no Bode de maneira direta.

Como dito anteriormente, o controlador PID é um caso característico de um controlador de avanço e atraso de fase, a ação PD afeta a região de alta frequência, aumentando a fase e aumenta a estabilidade do sistema e a largura de banda do mesmo, o que causa uma resposta mais rápida. Enquanto a ação PI afeta a região de baixa frequência, aumentando o ganho e o compensador de fase nestas frequências inferiores. Um controlador por avanço e

atraso de fase tem a seguinte cara:

$$G_c(\omega) = k_c \frac{1+\tau_1\omega}{1+\beta\tau_1\omega}\frac{1+\tau_2\omega}{1+\alpha\tau_1\omega} \qquad ,\beta > 1 \quad 0 < \alpha < 1 \tag{3.9}$$

Em um controlador PID, a ação proporcional traz uma resposta mais rápida do sistema, a ação integrativa é uma maneira de se remover o erro estacionário enquanto a derivativa reduz o valor do sobressinal e antecipa a correção do valor de saída, melhorando também a velocidade de resposta do sistema (FELIPE NEVES, s.d.). A equação 3.9 é uma equação general de controladores de atraso e avanço de fase, onde o PID se encaixa, mas pode-se aprimorar a fórmula para uma generalização do PID, como será mostrado a seguir:

$$u(t) = Kp \times e(t) + Ki \times \int_{t=0}^{t} e(t)dt \; + Kd \times \frac{de(t)}{dt} \tag{3.10}$$

Com K_p sendo o coeficiente da ganho proporcional, T_i valor de tempo integral, T_d de tempo derivativo (OGATA, 2010), t o instante de tempo, $u(t)$ sinal de saída do sistema e $e(t)$ o sinal de erro na entrada do controlador. Com a fórmula definida, passa-se para o domínio da frequência com a transformada de Laplace e se obtém:

$$\frac{U(s)}{E(s)} = G(s) = K_p + \frac{K_p}{T_i s} + K_p T_d s \tag{3.11}$$

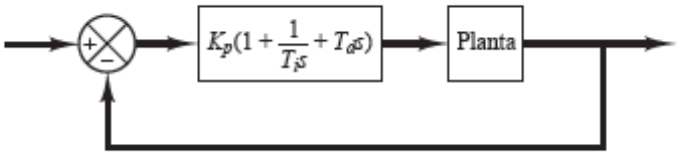


Figura 3.29 – Controle PID de uma Planta

Fonte: (OGATA, 2010)

Existem outros métodos para identificação de sistemas, como o trabalho(SÁ, 2013) que apresenta diversas maneiras de se identificar sistemas de motores de corrente contínua. Por exemplo, o método de Smith usado em sistemas de primeira ordem não se traça uma reta tangente como no método apresentado, e em outros métodos também citados nesse trabalho, mas são usados pontos da própria curva da função de transferência do motor. Os pontos na curva são em relação aos instantes os quais a saída é 28.3% do valor final e 63.2% do valor final.

Com esses valores, são calculados dois parâmetros $\tau = 1.5(t_2 - t_1$ e $\theta = t_2 - \tau$. Como mostra a Figura 3.30

Com esses valores se tem a função de transferência $G(s) = \frac{K}{\tau s+1}e^{-\theta s}$ onde K é o ganho do sistema.

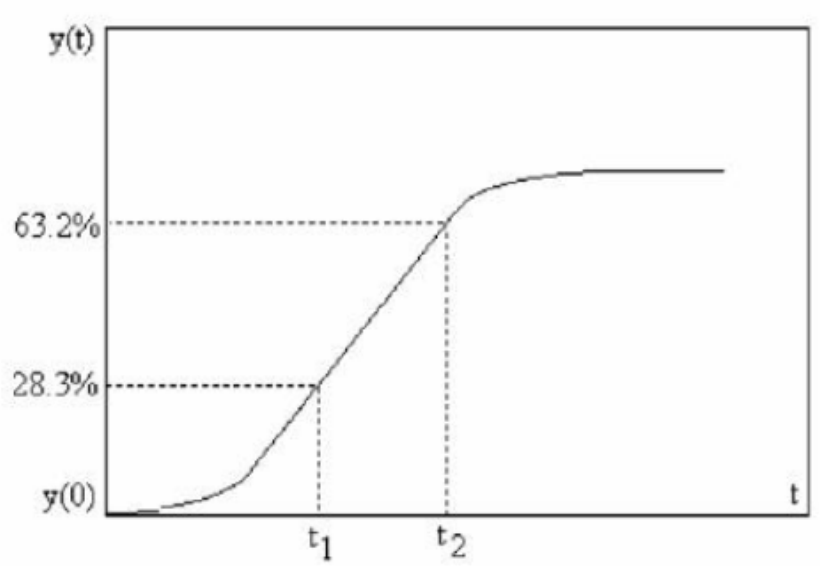


Figura 3.30 – Sintonia pelo Método de Smith

Fonte: (SÁ, 2013)

3.3.1 Sintonia Ziegler-Nichols

A sintonia de Ziegler-Nichols é um método para selecionar os parâmetros do controlador PID, pode ser dito como regras para ajustar os valores de K_p, T_i e T_d. Existem dois métodos de sintonia, sendo que será usado só um deles neste trabalho. O método consiste em analisar a resposta da planta a uma entrada degrau, caso ela não possua polos complexos conjugados dominantes ou integradores, ela terá um formato em S, como nas Fiugura 3.31 e 3.32. Traçando uma reta tangente ao ponto de inflexão pode se calcular os parâmetros de acordo com a tabela:

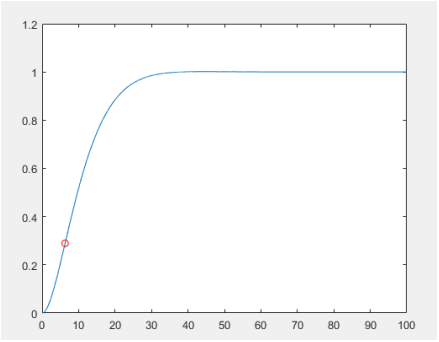


Figura 3.31 – Resposta do Motor representada no Matlab

Fonte: Próprio Autor

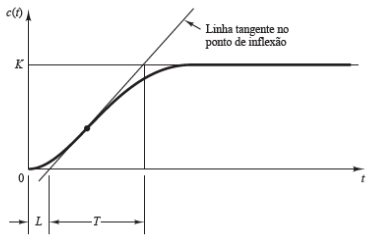


Figura 3.32 – Curva de Resposta em formato S

Fonte: (OGATA, 2010)

Figura 3.33 – Curva em S do Motor e Geral

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

Tabela 3.5 – Regra de Sintonia Ziegler-Nichols

Fonte: (OGATA, 2010)

Com a tabela 3.5 e a fórmula do controlador PID 3.11, pode se chegar no resultado $0.6T\frac{(s+\frac{1}{L})^2}{s}$ e se observa que existe um polo na origem e zero duplo em $s = -1/L$.

4 Desenvolvimento

4.1 Controlador

Inicialmente no trabalho (PERES, 2023) foi proposto a utilização de uma rede de Arduinos onde cada motor seria controlado por um Arduino Uno e estes microcontroladores seriam controlados por um Arduino Mega, devido a impossibilidade de um Arduino Uno de controlar mais de um motor, além de capacidade de processamento, este não apresenta portas suficientes para serem usadas como portas de interrupção que são ocasionadas pelos sensores de efeito hall do motor. Como uma maneira de se evitar isso e utilizar um único microcontrolador para ambos os motores, foi sugerida a utilização do ESP32.

O ESP32 apresenta melhores capacidades técnicas neste quesito, as Figuras 4.34 e 4.35 mostram as especificações dos microcontroladores do Arduino, enquanto a Figura 4.36 apresenta as especificações dos pinos do ESP32 juntamente com algumas características físicas no canto inferior esquerdo da imagem, as quais comprovam que a placa é superior às anteriores no quesito de memória e capacidade de processamento. Além destas questões apresentadas, existem ainda mais vantagens na utilização de um único microcontrolador como o ESP32 no local de uma rede Arduinos:

- O Arduino UNO que realizaria o controle do motor apresenta somente 2 pinos de interrupção, ainda é possível se controlar o motor, mas além de um dos sensores ser inutilizado, a resolução seria reduzida em 1/3. Enquanto ESP apresenta uma grande quantidade de pinos de uso de propósito geral de *input* e *output*, GPIO, sendo que sua maioria pode ser configurada como pino de interrupção, permitindo uma quantidade maior de sensores comunicando com a placa;
- A utilização de mais de uma placa, abre mais possibilidade para erros como mal contato e aumenta a complexidade de montagem;
- Por ser uma placa menor em tamanho, facilita a integração do microcontrolador ao exoesqueleto;

A hipótese de que o ESP poderia ser utilizado para o controle de dois motores foi feita de maneira básica, inicialmente se testou a resposta do microcontrolador com um motor e em seguida foram ligados dois motores simultaneamente e avaliado se as ativações das interrupções estavam de acordo com o esperado e se o código sofria alguma alteração de comportamento. Não se observou mudança de comportamento quando se comparado ao caso de somente um motor, desse modo a placa ESP se torna uma placa promissora como

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figura 4.34 – Especificações Técnicas Arduino Uno

Fonte: (ARDUINO.CC, 2023b)

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Figura 4.35 – Especificações Técnicas Arduino Mega

Fonte: (ARDUINO.CC, 2023a)

ESP32-DevKitC

ESPRESSIF

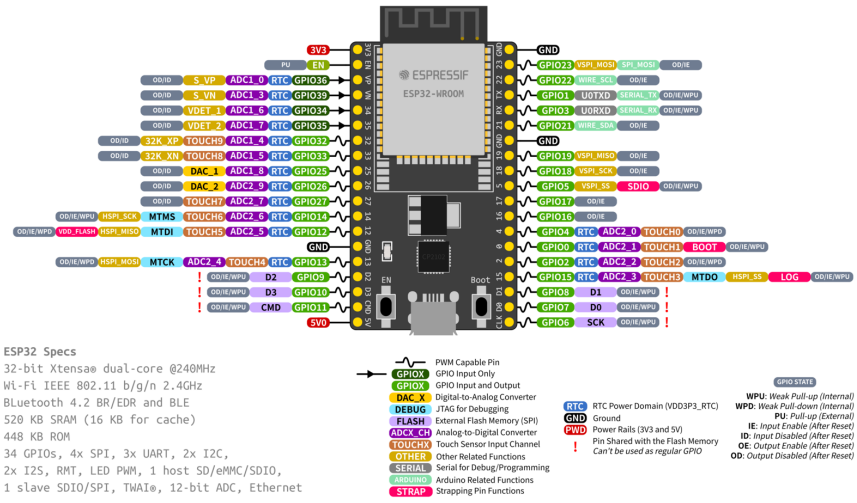


Figura 4.36 – Especificações Técnicas ESP32

Fonte: (ESPRESSIF, 2023)

unidade de processamento. O método utilizado para que se garantisse que as interrupções estavam sendo contabilizadas corretamente, foi realizado o acionamento e a finalização de movimento do motor de maneira arbitrária, o código imprimia a quantidade de ativações de cada um dos sensores em um breve intervalo de tempo dentro de um laço de repetição, então checando o deslocamento do motor e sabendo a resolução [1] do mesmo, é possível saber quantos pulsos totais deveriam ter sido dados para que aquela posição final fosse atingida.

[1] O motor contém 12 pares de polos para os sensores, então para cada um dos 3 sensores são 24 pontos de ativação, totalizando 72 pulsos totais. Juntando com o redutor 100:1, são 7.200 pulsos para uma volta completa do eixo de saída

Com essas informações obtidas, o passo seguinte foi realizar o controle de posição para o motor, os testes nesta etapa ainda serão realizados na bancada de testes, visto que essa foi projetada com esse propósito e se tem uma maior liberdade com seus movimentos visto que o conjunto motor-redutor não está acoplado a nenhuma haste metálica e não existe restrição de movimento para o motor.

4.2 Motor

4.2.1 Obtenção de dados

O desenvolvimento de um projeto de controlador se baseia em etapas, inicialmente é necessário se obter a função de transferência da planta, que no caso deste projeto é o motor. A obtenção dos dados do motor foi realizada com auxílio do *software* da ESCON Studio, que gerou uma recordação de dados de entrada e resposta do motor, com a velocidade desejada sendo a referência e a velocidade real como a saída, vide Figura 4.37. Em seguida foi calculado em *Matlab* os parâmetros da curva e estimado uma função de transferência da planta, como pode ser visto na Figura 4.38. Finalmente, o próprio *Matlab* contém bibliotecas que auxiliam com os projetos de controladores, como a função *pidtune(FT)*, a qual retorna valores dos parâmetros K_p,K_d e K_i. A função de transferência em malha fechada estimada e os valores retornados pela função *pidtune(Gmf,'PID')* podem ser vistos nas equações 4.1 e 4.2.

$$Gmf = \frac{0.03662}{s^2 + 0.32s + 0.03662} \tag{4.1}$$

$$K_p + K_i * \frac{1}{s} + K_d * s, \quad K_p = 2.03, K_i = 0.278, K_d = 3.28 \tag{4.2}$$

4.2.2 Código do controle

O código de controle foi desenvolvido usando como referência um código de controle de posição de um motor bastante similar ao encontrado no laboratório, realizando o ajuste de parâmetros necessários ((IIBAAA, 2023)). Com este código readaptado para o ESP32, usando o *framework* do ESP-IDF e com os valores do motor do projeto, se realizou o controle de posição do motor. O ESP32 recebe via *bluetooth* uma linha de texto com os valores que serão usados no caso de teste, caso se queira ajustar parâmetros do controlador também é possível via essa linha de *input*. Em seguida à desconexão do *bluetooth*, o código inicia o controle em um laço de repetição que define a velocidade de rotação do motor com o cálculo do controlador e checagem da posição por contagem de pulsos gerados pelas interrupções dos sensores de efeito Hall e quando a contagem de pulsos se iguala ao valor desejado, é desligado o movimento do motor. Onde a suavidade desse movimento é ditada pelo cálculo

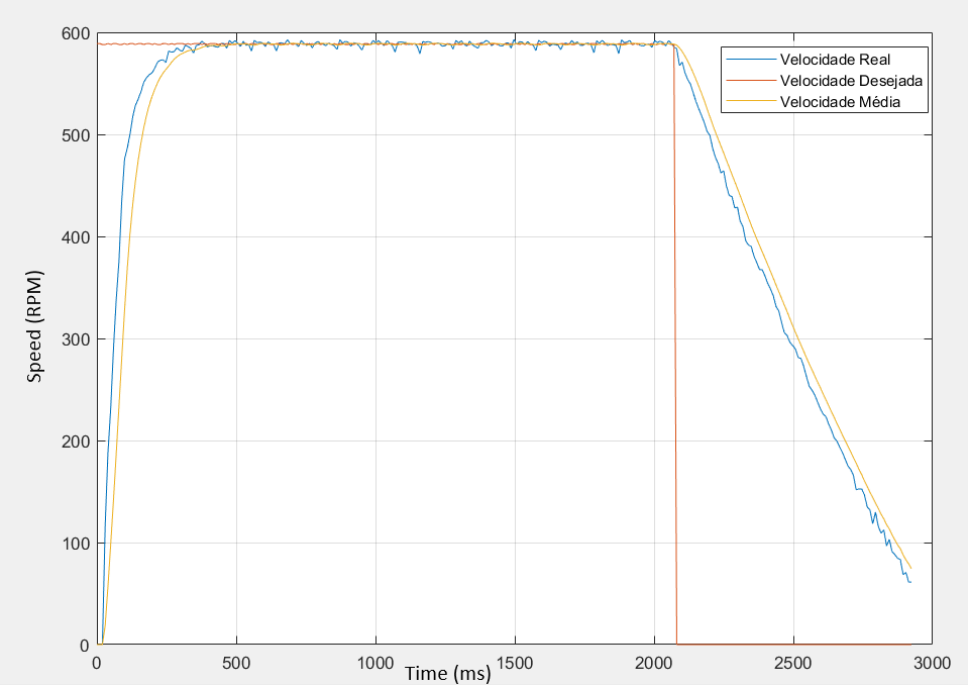


Figura 4.37 – Dados Obtidos pelo ESCON Studio

Fonte: Próprio Autor

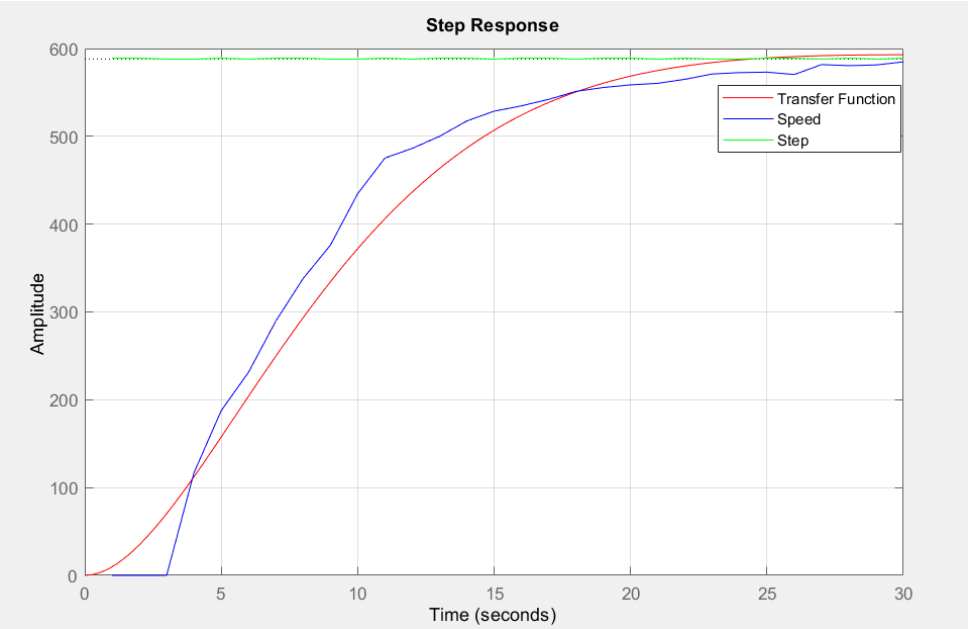


Figura 4.38 – Resposta ao Degrau da Função de Transferência e Função Real

Fonte: Próprio Autor

dos parâmetros do controlador, devido a este fato a definição destes parâmetros deve ser realizada com bastante cautela.

O código tomado como base usa uma faixa de valores que vão de -100 à 100, como pode ser identificado no fragmento de código abaixo, mas como a saída de um PWM de 8 bits vai de 0 à 255, foi realizado um remapeamento de valores para ser usado no código, o

Arduino apresenta uma função chamada de *map(from_low,from_high,to_low,to_high)* que realiza isso, mas como é uma simples linearização, no código do ESP32 foi colocado a relação matemática direta para a conversão a Figura 4.39 mostra essa relação.

O cálculo principal e a ativação do motor é realizada na seção do código a seguir, em (IIBAAA, 2023) pode ser encontrado o código em sua totalidade.

```
...
    void control()
    {
      desired_step = (int)(desired_angle * step_to_deg);

      error = desired_step - pulseCount;

      double pid_out = error * Kp+ Kd * (error-prev_error);
      prev_error = error;

      if (pid_out > 0){
        if (pid_out > 100)
          pid_out = 100;
        else if (pid_out < 10)
          pid_out = 10;
      }

      else {
        if (pid_out < -100)
          pid_out = -100;
        else if (pid_out > -10)
          pid_out = -10;
      }

      motor_start(pid_out);
    }

    void motor_start(double spd){
      if (spd != 0 && status1 == 1) {
        digitalWrite(en, HIGH);

        double out = map(spd, -100, 100, 255, 0);
        analogWrite(sped, out);
      }
      else{
        digitalWrite(en, LOW);
      }

    }
...
```

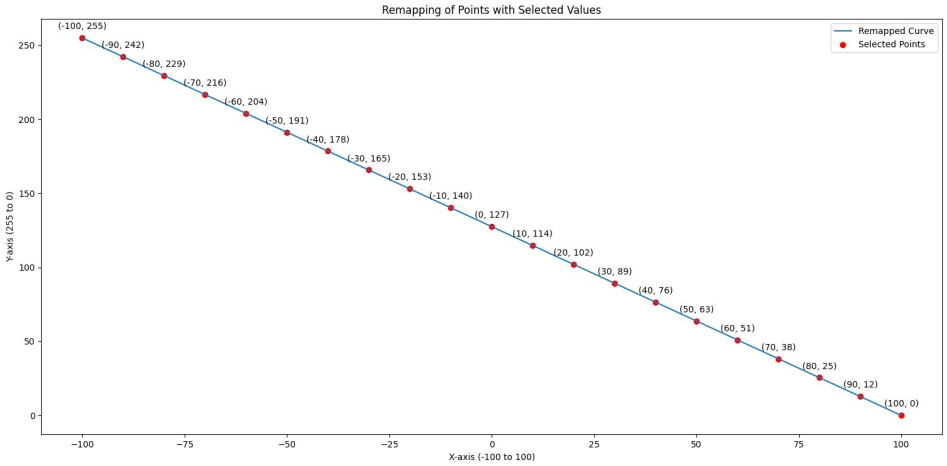


Figura 4.39 – Conversão de Valores da conta de PID para PWM

Fonte: Próprio Autor

Os valores foram estabelecidos empiricamente e os testes iniciais foram conduzidos usando as definições previamente incorporadas no código de exemplo, devido ao fato de similaridades dos equipamentos. Os valores foram sendo ajustados e testados para que apresentassem a melhor resposta, isto é, um menor erro e uma resposta mais suave do motor. Os testes foram desempenhados para diversos valores de ângulos com o intuito de se observar de maneira experimental o que as modificações de valores causariam nos resultados. A Figura 4.40 mostra que a maior variação angular realizada é de 80° no plano sagital, então essa será a faixa mantida para os testes após a definição final dos parâmetros.

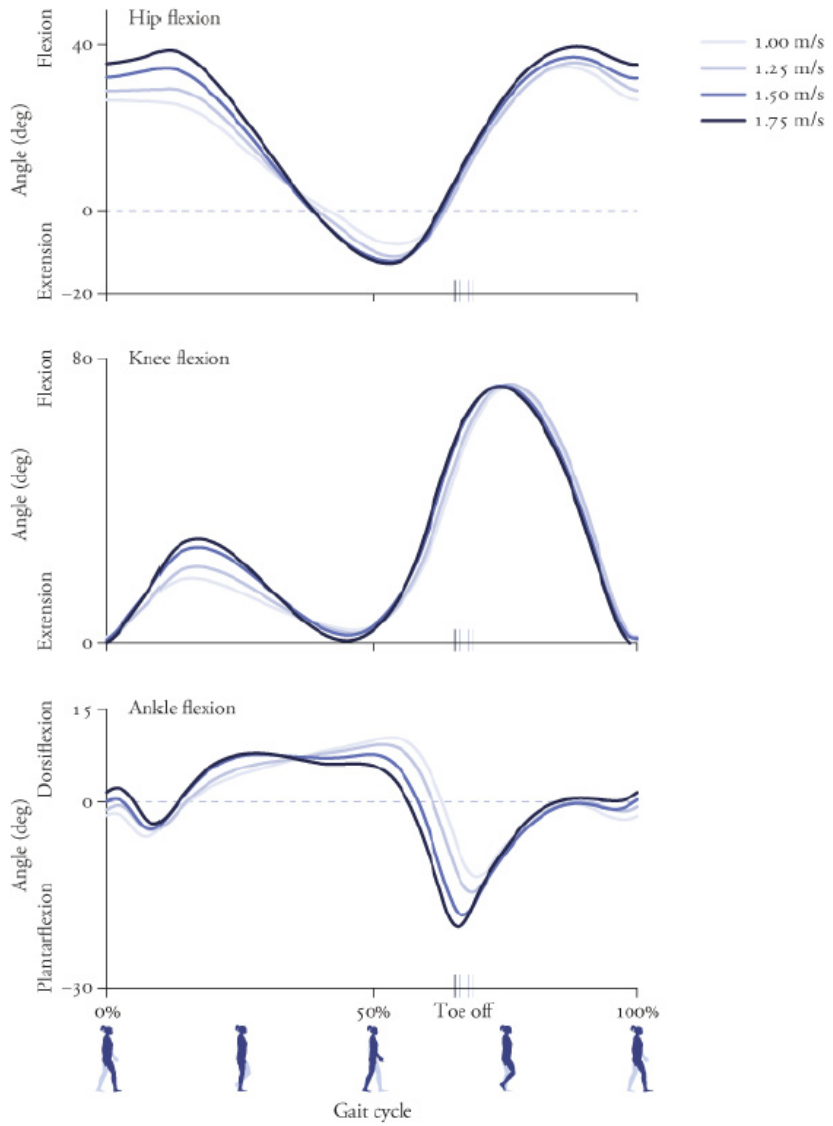


Figura 4.40 – Flexão e extensão do quadril, joelho e tornozelo ao caminhar

Fonte: (UCHIDA; DELP, 2020)

A partir dessas informações, foram realizados alguns testes para algumas combinações de valores de K_p e K_d sem que houvesse alterações significativas no código de referência com a finalidade de se obter a melhor combinação de valores para que então se pudesse realizar o ajuste de lógica e aplicabilidade para ambos os motores simultaneamente e então avançado para a etapa de controle de trajetória. A Tabela 5.6 contém os resultados finais obtidos com os valores das constantes já definidos de maneira empírica, sem ajustar o código e alterando somente os valores das constantes se constatou que, com os valores definidos, ângulos maiores apresentavam maior erro, os ângulos até 90° ficaram com o erro na faixa de 2° à 10°. Entretanto, como a variação angular dos motores não passaria de 80°, os valores foram escolhidos em $K_p = 2.03$ e $K_d = 3.28$, após ajustar o código para minimizar os erros, os resultados podem ser encontrados na Tabela 5.6 no capítulo 5.

Após os testes realizados com o controle de posição, o passo seguinte é o controle de trajetória. Com o auxílio da Figura 4.40 foram definidos os ângulos de flexão e extensão do quadril e os ângulos de flexão do joelho, com os valores já pré-definidos, não é necessária a realização da cinemática inversa vista em 3.2.2.2 . Pegando a relação da proporção dos *pixels* da imagem com os valores postos no gráfico, é possível definir os pontos das curvas como visto na Figura 4.41. As Figuras 4.42, 4.43 e 4.44 comprovam que os valores definidos estão satisfatórios e as formas das curvas se mantiveram similares ao representado no livro.

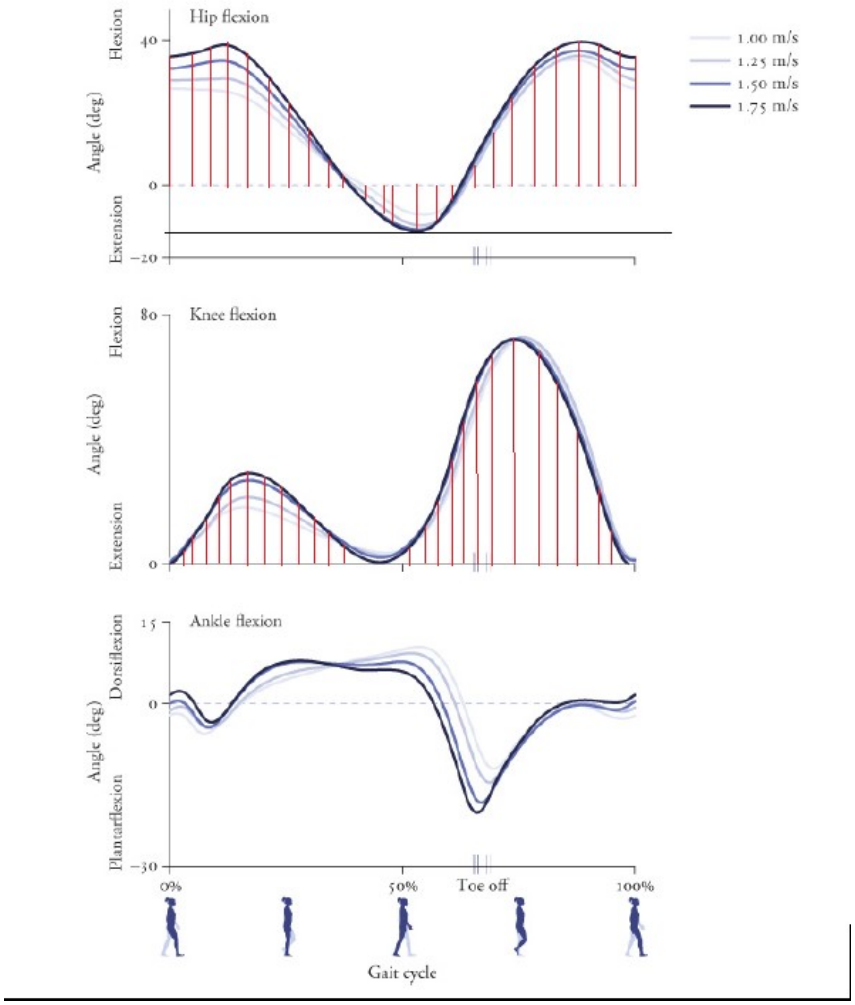


Figura 4.41 – Variação Angular com retas auxiliares

Fonte: Próprio Autor

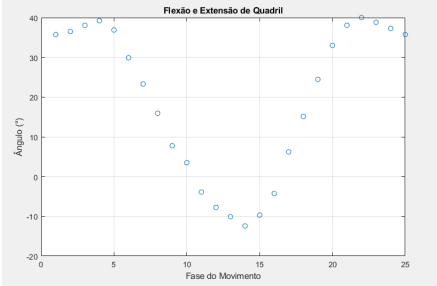


Figura 4.42 – Extensão e Flexão de Quadril ao Caminhar

Fonte: Próprio Autor

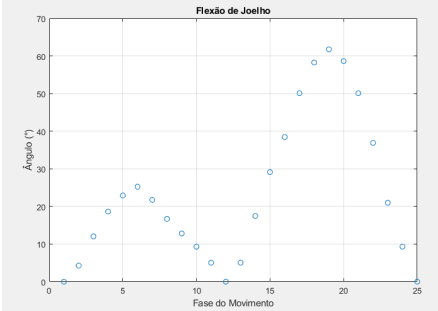


Figura 4.43 – Flexão de Joelho ao Caminhar

Fonte: Próprio Autor

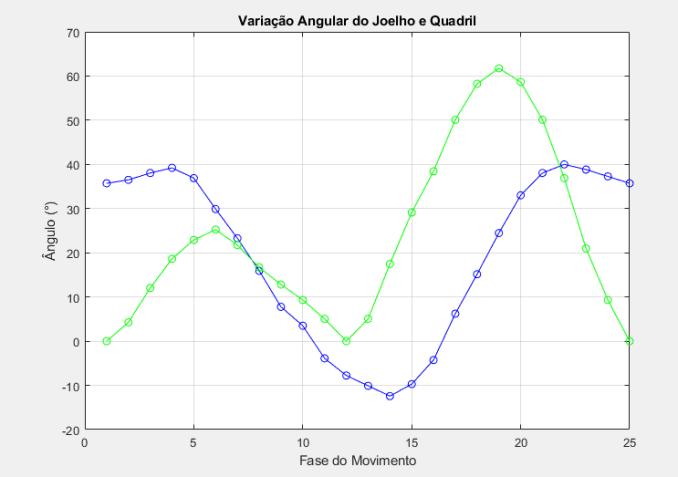


Figura 4.44 – Variação Angular de Quadril e Joelho

Fonte: Próprio Autor

Com os ângulos definidos, o código de controle finalizado e testado, basta simplesmente implementar o controle da trajetória, basicamente será realizado o controle de posição dentro de um laço de repetição para que o microcontrolador recebesse um vetor com os ângulos que deverá levar o motor, isto para cada um dos motores. O vetor de ângulos é transformado em um vetor de pulsos, visto que o motor conta a quantidade de pulsos que os sensores de efeito hall geram para definir em que posição o atuador se encontra, contudo os pulsos são definidos de maneira relativa ao pulso anterior, pois o código de controle é realizado movendo o motor na quantidade absoluta do ângulo recebido.

5 Análises e Resultados

O projeto pode ser separado em 3 etapas maiores para o desenvolvimento, etapa inicial de conhecimento de equipamentos e atualização do microcontrolador, a etapa de desenvolvimento do controle de posição a etapa final do controle de trajetória, como visto no capítulo 4. A primeira etapa já foi discutida brevemente no capítulo anterior e não representa uma fase tão substancial para o projeto a ponto de se diligenciar uma análise profunda sobre o equipamento, então o foco dos resultados deste capítulo será referente ao controle de posição e trajeto desenvolvidos.

5.1 Controle de posição

Os parâmetros foram definidos e código apurado até o ponto em que erro de posição chegasse a um valor satisfatório. De acordo com os valores da tabela 5.6 é possível notar que a precisão do controle é alta. Ressalva-se que a velocidade do motor não foi limitada, exceto pela própria definição de segurança já estabelecida no *driver*, que estabelece uma faixa de operação do motor em 10% à 90% do valor total do PWM.

Os testes de controle de posição foram realizados para diversos ângulos, o foco do controle se encontra em ângulos menores que 90° ou maiores que -90°, mas ainda sim foram avaliados ângulos quie fugissem da faixa de operação esperada pelo projeto. A Tabela 5.6 apresenta uma diferença mostrada entre o ângulo obtido e valor de erro apresentado nela, isso se deve ao fato por dois principais motivos. A avaliação do ângulo obtido foi realizada a olho nu, que não é tão preciso quanto a contagem de pulsos verificada pela placa com a movimentação do robô, a segunda razão da diferença de valores ocorre devido a imprecisão na hora de voltar o motor ao ponto de partida de ângulo 0°, que foi feita de maneira manual, somando essas imprecisões e o fato das marcações das bancadas serem distantes de 10°entre elas, gerou-se um erro na hora de avaliar onde o motor havia de fato finalizado o movimento.

A coluna de Erro da tabela é dada pelo retorno do próprio ESP32, onde após desligar a porta *enable* do motor e zerar o sinal PWM, é impresso ao usuário a diferença entre o Pulso que era desejado para chegar em tal posição e a quantidade de pulsos lidos pelo microcontrolador, convertendo os pulsos em graus para uma ideia mais precisa do erro. Tendo em vista que os erros foram menores que 1° pelo ESP32 e o máximo do erro aparente foi de 2°, é possível afirmar que o resultado esperado do controle foi obtido.

O código de controle apresenta um potencial problema quando não relacionado ao seu funcionamento e lógica, mas como pode ser visto na imagem disponibilizada pelo ESP32

[1] Valores negativos sinalizam uma rotação no sentido horário

Ângulo Desejado	Ângulo Obtido	Erro	
		Pulsos	Graus
30°	30°	9	0.45°
	31°	-2	-0.15°
-30°	-33°	18	0.9°
	-30°	9	0.45°
45°	45°	3	0.15°
	46°	4	0.2°
-45°	-45°	5	0.25°
	-46°	4	0.2°
90°	92°	11	0.55°
	91°	11	0.55°
-90°	-89°	4	0.2°
	-90°	3	0.15°
300°	300°	3	0.15°
	300°	3	0.15°
-300°	-300°	5	0.25°
	-300°	3	0.15°

Tabela 5.6 – Resultados dos Testes do Controle de Posição[1]

sobre os recursos utilizados para o que está sendo requerido pela placa, Figura 5.45, o valor da memória *SRAM* está um pouco elevada, próximo de 80% do total. Entretanto, esse fato se deve a conexão *Bluetooth* que foi feita com o próprio exemplo encontrado no GitHub do ESP-IDF para conexões BLE, pode-se comparar esses valores com o código final desenvolvido que realiza o controle de dois motores, mas mesmo assim apresenta bem menos necessidade de memória, Figura 5.46.

```
Total sizes:
Used static DRAM:   48284 bytes (  76296 remain, 38.8% used)
      .data size:   17060 bytes
      .bss  size:   31224 bytes
Used static IRAM:  102802 bytes (  28270 remain, 78.4% used)
      .text size:  101775 bytes
   .vectors size:    1027 bytes
Used Flash size :  801371 bytes
      .text     :  636499 bytes
      .rodata   :  164616 bytes
Total image size:  921233 bytes (.bin may be padded larger)
```

Figura 5.45 – Valores de memória usadas pelo ESP32 no controle de Posição

Fonte: Próprio Autor

```
Total sizes:
Used static DRAM:   15460 bytes ( 109120 remain, 12.4% used)
      .data size:   12748 bytes
      .bss  size:    2712 bytes
Used static IRAM:   50406 bytes (  80666 remain, 38.5% used)
      .text size:   49379 bytes
   .vectors size:    1027 bytes
Used Flash size :  144635 bytes
      .text     :  102971 bytes
      .rodata   :   41408 bytes
Total image size:  207789 bytes (.bin may be padded larger)
```

Figura 5.46 – Valores de memória usadas pelo ESP32 no controle do Trajeto

Fonte: Próprio Autor

5.2 Controle de trajetória

A realização do controle de trajetória do exoesqueleto é realizada com a mesma lógica da etapa anterior, controle de posição. Entretanto, os ângulos os quais serão alcançados já foram previamente definidos baseando-se na Figura 4.44 encontrada no capítulo 4. O resultado obtido pode ser visto nas Figuras 5.47,5.48,5.49 e 5.50 a seguir:

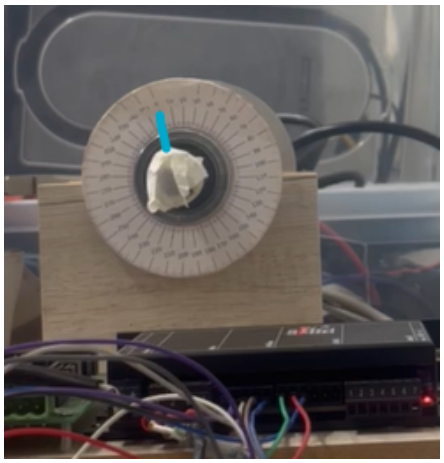


Figura 5.47 – Posição Inicial do Motor

Fonte: Próprio Autor

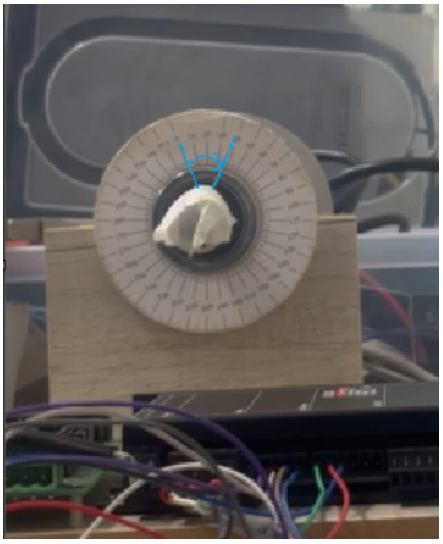


Figura 5.48 – Posição final após a "extensão"de quadril

Fonte: Próprio Autor

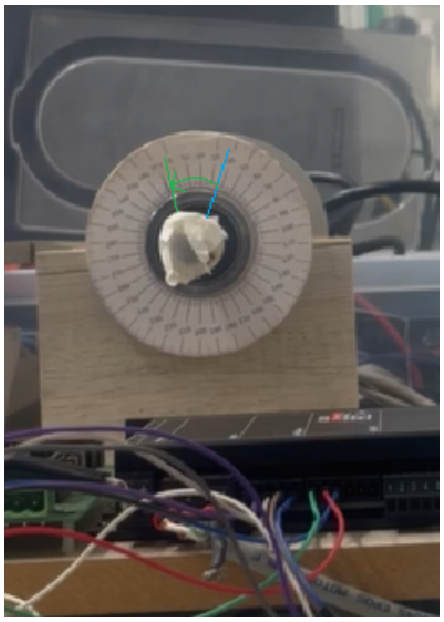


Figura 5.49 – Posição final após a "flexão"de quadril

Fonte: Próprio Autor

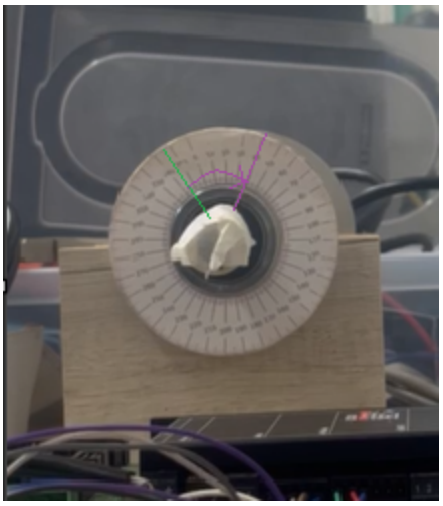


Figura 5.50 – Posição final após a última "extensão"de quadril

Fonte: Próprio Autor

O motor representante do quadril não ultrapassou os limites delimitados, como também chegou aos ângulos com a precisão demonstrada na Tabela 5.6, o que é bem satisfatório, pois o erro está próximo de zero. Também foi simulado em *matlab* com auxílio da *robotics toolbox* a trajetória que seria realizada pelo manipulador robótico, Figuras 5.51, 5.52, 5.53, 5.54 e 5.55. Com as imagens da simulação é possível verificar que o trajeto do robô está acontecendo de maneira satisfatória, os ângulos escolhidos foram os de maior valor da Figura 4.41, portanto, para uma pessoa que apresente dificuldade de movimento talvez uma leve diminuição dos ângulos seja necessária. Pela própria imagem nota-se que a diferença entre

estes valores é pequena para uma pessoa sem limitações na caminhada, então a definição desses parâmetros os quais o os motores deveriam alcançar, devem ser dados de acordo com a preferência pessoal de cada pessoa que vista o exoesqueleto.

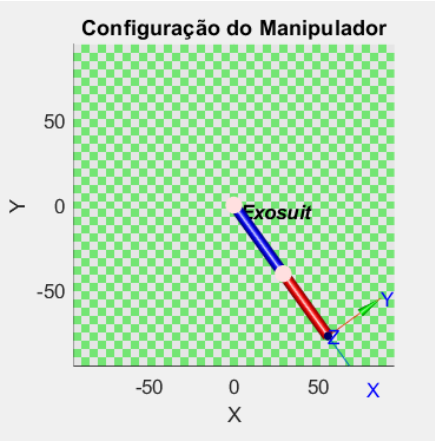


Figura 5.51 – Posição Inicial da Caminhada

Fonte: Próprio Autor

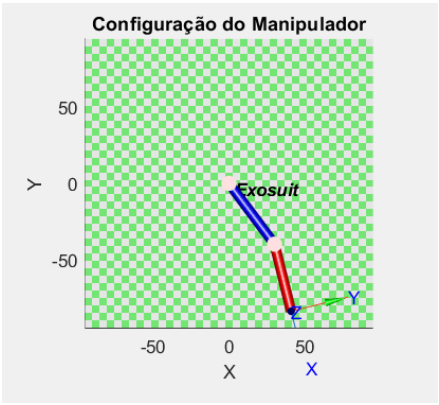


Figura 5.52 – Início do Movimento

Fonte: Próprio Autor

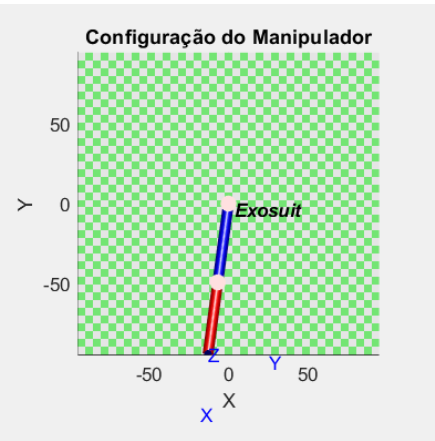


Figura 5.53 – Ponto de contato do Pé com o Solo

Fonte: Próprio Autor

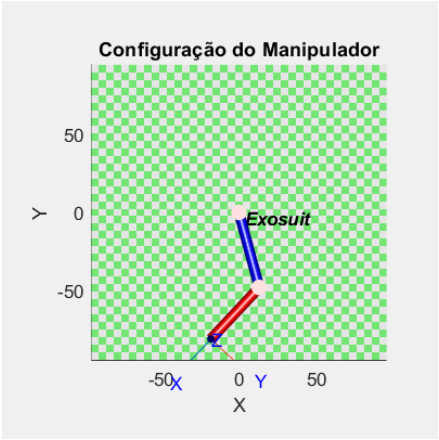


Figura 5.54 – Posição intermediária da Caminhada, realizando a volta à configuração inicial

Fonte: Próprio Autor

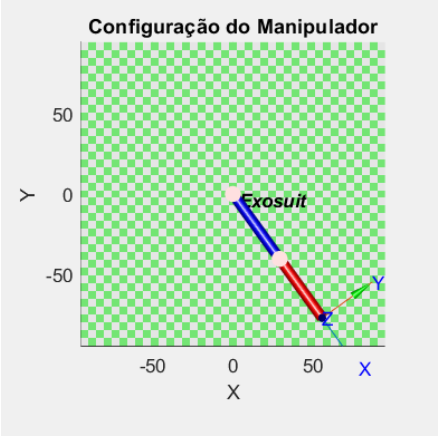


Figura 5.55 – Posição final da Caminhada e inicial de um novo ciclo

Fonte: Próprio Autor

6 Considerações Finais

6.1 Conclusão

A caminhada humana é um movimento complexo e envolve diversas articulações, estruturas corporais, musculatura e coordenação de uma pessoa, esse processo natural para uma grande parte da humanidade pode ser afetado de inúmeras maneiras, AVCs, acidentes automobilísticos, amputações, como também podem ser doenças herdadas desde o nascimento. O desenvolvimento de tecnologias que auxiliam essas pessoas para uma vida com menos limitações e se possível, sem limitações, são estudadas diariamente pela sociedade. Esse projeto apresentou uma dessas tecnologias.

Para efetuar um estudo de uma caminhada é necessário realizar mais que uma simples análise no plano sagital, como dito no capítulo 3. No entanto, o primeiro passo com o controle dos motores dentro dos limites e funcionamento do equipamento foi dado. O código desenvolvido permite fácil alteração de parâmetros e é de simples aplicabilidade, replicabilidade e expansibilidade para o momento que for montado o lado esquerdo do exoesqueleto. A adição de mais motores ao código torna-se uma tarefa sem complicações, bastando somente copiar o código já criado, ajustando nome das variáveis e definir os ângulos, mantendo os valores inalterados, mas a ordem é distinta. Para a aplicação final também é necessário realizar testes para conhecimento de se a placa é capaz de suportar 4 motores simultaneamente, pois haveria grande acréscimo na quantidade de interrupções geradas pelos 12 sensores de efeito Hall.

A simulação e a bancada de testes não levam em conta a carga carregada pelo motor, então o momento de inércia gerado não foi levado em consideração, visto que a análise era somente sobre a angulação e trajetória da perna. A bancada de testes também não contém nada acoplado ao motor, o que pode gerar diferenças na resposta dos motores do exoesqueleto quando comparadas as respostas à bancada. Os testes de movimento iniciais realizados no protótipo mostraram que o manipulador consegue se movimentar sem vibração e de maneira satisfatória com angulações maiores que as que seriam utilizadas na caminhada, sugerindo que mesmo sem análise definitiva do efeito do momento de inércia, o motor realizaria o movimento desejado. Entretanto quando se for realizar testes com pessoas vestindo o exoesqueleto, o peso e resistência ao movimento seria aumentado de maneira significativa, podendo gerar resultados que não foram previstos neste trabalho.

O capítulo 5 evidenciou que os motores seguem os ângulos definidos de forma satisfatória e contínua, portanto, o controle de trajetória seguindo como base uma caminhada de uma pessoa saudável foi realizado de forma aceitável. O conjunto motor-redutor permite

uma alta resolução, são 7.200 pulsos gerados pelos 3 sensores de efeito hall em conjunto, 2.400 para cada, considerando uma volta completa, o que proporciona uma boa precisão no controle do motor e esta foi alcançada no trabalho. O projeto teve uma melhora no que se diz respeito sobre aspectos computacionais com o microcontrolador. Porém, em relação ao microcontrolador, para realizar o controle de duas pernas, não há garantia que o ESP32 suporte o número de interrupções totais geradas por 4 motores simultaneamente sem que teste experimentalmente essa situação, pois a depender da velocidade de rotação do motor, as interrupções geradas pelos sensores podem vir a impedir que o código seja percorrido por completo, mas para o controle de dois motores não foi necessário limitar o sinal PWM gerado além dos que já pré-definidos no *driver*, 90% do valor total.

6.2 Projetos futuros

6.2.1 Ampliação para segunda perna

O projeto foi desenvolvido quando o exoesqueleto havia somente a perna direita e esta não tinha o apoio para o pé. Portanto a realização dos testes para averiguar se o microcontrolador escolhido teria capacidade de processamento para o acréscimo de dois motores não foram realizados. A ampliação do código para a segunda perna seria realizada com a mesma lógica do código presente neste trabalho. bastando então a definição de variáveis e dos ângulos da perna oposta de maneira que os movimentos fossem complementares entre as duas pernas.

6.2.2 Realização de testes

A próxima fase do projeto engloba a realização de testes, inicialmente com pessoas saudáveis e, posteriormente, com aqueles que apresentam algum grau de dificuldade com o movimento. Serão necessários estudos de caso específicos para a definição de novos parâmetros, controle de velocidade, entre outros. Os testes podem abranger diversas situações, como planos inclinados e superfícies com certas irregularidades o que seria de grande valor para um desenvolvimento abrangente do projeto.

Referências

ABERNETHY, B.; KIPPERS, V.; HANRAHAN, S.; PANDY, M.; MCMANUS, A.; MACKINNON, L. **Biophysical foundations of human movement**. 3. ed.: Human Kinetics, 2013. ISBN 978-1-4504-3165-1. Citado na p. 26.

ARDUINO.CC. **Arduino Mega 2560 R3**. Disponível em: <https://store.arduino.cc/products/arduino-mega-2560-rev3>. Acesso em: 1 dez. 2023. Citado na p. 44.

ARDUINO.CC. **Arduino UNO R3**. Disponível em: <https://store.arduino.cc/products/arduino-uno-rev3>. Acesso em: 1 dez. 2023. Citado na p. 44.

BLAYA, J.; HERR, H. Adaptive control of a variable-impedance ankle-foot orthosis to assist drop-foot gait. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, v. 12, n. 1, p. 24–31, 2004. DOI: 10.1109/TNSRE.2003.823266. Citado na p. 21.

CHEN, G.; HOU, B.; GUO, S.; WANG, J. A Guide for Human Walking Model and Control — Insights from Mechanical Property Analysis of Human Walking. In: 2020 Chinese Control And Decision Conference (CCDC). 2020. P. 4785–4790. DOI: 10.1109/CCDC49329.2020.9164160. Citado na p. 22.

CNN BRASIL. **Brasil tem mais de 17 milhões de pessoas com deficiência, segundo IBGE**. Disponível em: <https://www.cnnbrasil.com.br/nacional/brasil-tem-mais-de-17-milhoes-de-pessoas-com-deficiencia-segundo-ibge/#:~:text=A%5C%20pesquisa%5C%20detalha%5C%20que%5C%207,pessoas%5C%20t%C3%AAm%5C%20nos%5C%20membros%5C%20superiores>. Acesso em: 12 mai. 2023. Citado na p. 15.

CÓRDOBA, L. M. I. **Estratégia de Controle Híbrido Bioinspirado Para um Exoesqueleto Robótico de Membro Inferior**. 2022. Tese de Doutorado apresentada à Faculdade de Engenharia mecânica da Universidade Estadual de Campinas. Disponível em: https://hdl.handle.net/20.500.12733/5278. Citado na p. 19.

CÓRDOBA, L. M. I. **MECHANICAL DESIGN OF A LOWER LIMB EXOSKELETON FOR REHABILITATION OF PARAPLEGIC PATIENTS**. 2021. Tese de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade de Brasília. Disponível em: https://repositorio.unb.br/handle/10482/43325. Citado na p. 17.

CYBERDYNE. **Figura exoesqueleto HAL**. Disponível em: https://www.cyberdyne.jp/english/products/LowerLimb_medical.html – acesso em 11/05/2023. Citado na p. 20.

DOLLAR, A. M.; HERR, H. Lower Extremity Exoskeletons and Active Orthoses: Challenges and State-of-the-Art. **IEEE Transactions on Robotics**, v. 24, n. 1, p. 144–158, 2008. DOI: 10.1109/TRO.2008.915453. Citado na p. 20.

ESPRESSIF. **ESP32-WROOM-32 PINOUT**. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/_images/esp32-devkitC-v4-pinout.png>. Acesso em: 1 dez. 2023. Citado na p. 44.

FELIPE NEVES. **Controlador PID Digital**. Disponível em: https://embarcados.com.br/controlador-pid-digital-parte-1/ – acesso em 31/05/2023. Citado na p. 40.

FREIRE, J. P. C. D. **Projeto Mecânico de um Exoesqueleto com atuação no Quadril**. 2019. Monografia do curso de Engenharia Mecânica na UnB. Disponível em: https://bdm.unb.br/handle/10483/26065. Citado nas pp. 17, 22.

GEORGE L COBB. **Walking motion**. Citado nas pp. 20, 21.

IIBAAA. **maxon-motor-arduino-control**. Disponível em: <https://github.com/iibaaa/maxon-motor-arduino-control/blob/main/arduino_code.ino>. Acesso em: 18 mai. 2023. Citado nas pp. 45, 47.

JORNAL DA USP. **USP cria exoesqueleto robótico para reabilitar pessoas que sofreram AVC**. Disponível em: https://jornal.usp.br/ciencias/usp-cria-exoesqueleto-robotico-para-reabilitar-pessoas-que-sofreram-avc/ – acesso em 29/05/2023. Citado na p. 19.

MARK W. SPONG SETH HUTCHINSON, M. V. Robot Modeling and Control. In: JOHN WILEY & SONS, INC, 2005. cap. 3, p. 65–100. Citado nas pp. 33, 35, 38.

MAXON MOTOR. **EC 90 flat 90 mm, brushless, 90 Watt**. 2017. Datahseet Motor Maxon Motor. Disponível em: https://www.maxongroup.com/medias/sys_master/root/8825435389982/17-EN-271.pdf – acesso 04/05/2023. Citado na p. 24.

MAXON MOTOR. Motor EC 90 flat. Figura do Motor. Disponível em: https://www.maxongroup.com/maxon/view/product/323772 – acesso em 04/05/2023. Citado na p. 23.

MAXON MOTOR. Planetary Garhead GP 52 C. Figura do Redutor. Disponível em: https://www.maxongroup.com/maxon/view/product/gear/planetary/gp52/223080 – acesso em 04/05/2023. Citado na p. 24.

MAXON MOTOR. **Planetary Gearhead GP 52 C 52 mm, 4.0–30.0 Nm**. 2021. Datahseet Redutor Maxon Motor. Disponível em: https://www.maxongroup.com/medias/sys_master/root/8882781421598/EN-21-410-411.pdf – acesso em 04/05/2023. Citado na p. 24.

NASCIMENTO, M. S. Desenvolvimento e Controle Digital de um Exoesqueleto de baixo custo para membros inferiores. Monografia do curso de Engenharia Elétrica no CEUB, 2022. Citado na p. 19.

NICHOLAS YAGN. **Apparatus for facilitating walking**. Disponível em: https://patents.google.com/patent/US420179A/en – acesso em 11/05/2023. Citado na p. 20.

OGATA, K. **Discrete-Time Control Systems**. 2. ed.: Pearson-Prentice-Hall, 1995. ISBN 0-13-328642-8. Citado na p. 39.

OGATA, K. Engenharia de Controle Moderno. In: 5. ed.: Pearson-Prentice-Hall, 2010. cap. 8, p. 522–525. ISBN 978-85-4301-375-6. Citado nas pp. 40–42.

PERER CORKE. Robotics Toolbox. Disponível em: https://petercorke.com/toolboxes/robotics-toolbox/ – acesso em 04/05/2023. Citado na p. 38.

PERES, A. B. Estudo do controle da trajetória de uma perna de exoesqueleto. Monografia do curso de Engenharia de Controle e Automação na UnB, 2023. Citado nas pp. 17, 25, 43.

PROJECT EMA. **Project EMA**. Disponível em: http://www.ene.unb.br/antonio/ema/pt/index.html – acesso em 29/05/2023. Citado nas pp. 18, 19.

ROBBI, D. B. **ANÁLISE DINÂMICA DE UM EXOESQUELETO DE MEMBROS INFERIORES UTILIZADO NO CONTEXTO DE REABILITAÇÃO DE INDIVÍDUOS COM LESÃO MEDULAR**. 2018. Monografia do curso de Engenharia Mecânica na UnB. Disponível em: https://bdm.unb.br/handle/10483/25198. Citado na p. 17.

RODRIGUES, A. P. C. **Desenvolvimento de uma interface para acionamento de atuadores e leitura de encodes para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo**. 2017. Monografia do curso de Engenharia Eletrônica na UnB. Disponível em: https://bdm.unb.br/handle/10483/20237. Citado na p. 17.

SÁ, L. L. de. **IDENTIFICAÇÃO E PROJETO DE CONTROLADORES APLICADOS EM UM MOTOR CC**. 2013. Monografia do curso de Engenharia Eletrônica na UnB. Disponível em: https://bdm.unb.br/handle/10483/6963. Citado nas pp. 40, 41.

SILVA, I. da; MOURA, C.; SOUSA, I. M. de. A EFICÁCIA DO USO DE EXOESQUELETO ROBÓTICO NA REABILITAÇÃO DA MARCHA PÓS ACIDENTE VASCULAR ENCEFÁLICO. In: v. 2. DOI: 10.52600/2763-583x.bjcr.2022.2.suppl.3.1073-1077. Citado na p. 19.

UCHIDA, T. K.; DELP, S. L. **Biomechanics of Movement: The Science of Sports, Robotics, and Rehabilitation**. The MIT Press, 2020. ISBN 9780262044202. Citado nas pp. 27–30, 49.

WANDERCRAFT. **Atalante**. Disponível em: https://en.wandercraft.eu – acesso em 11/05/2023. Citado na p. 21.

ZOU, C.; HUANG, R.; CHENG, H.; CHEN, Q.; QIU, J. Adaptive Gait Planning for Walking Assistance Lower Limb Exoskeletons in Slope Scenarios. In: 2019 International Conference on Robotics and Automation (ICRA). 2019. P. 5083–5089. DOI: 10.1109/ICRA.2019.8793863. Citado na p. 22.

Apêndices

Apêndice A – Códigos de programação

A.1 Arquivo de Organização (Controle.h)

Código A.1 – Código em C

```c
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <math.h>
#include "rom/gpio.h"
#include "driver/gpio.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/ledc.h"
#include "esp_log.h"
#include "sdkconfig.h"
#include "driver/uart.h"
#include "esp_timer.h"
#include "freertos/event_groups.h"
#include "esp_system.h"
#include "nvs_flash.h"
#include "string.h"
#include "esp_attr.h"


#ifndef FUNCTIONS_H
#define FUNCTIONS_H


//MACROS E VARIAVEIS
#define on 1
#define off 0

//Direction
#define CW 1
#define CCW 0

//Digital Signals
#define Direction 32
#define Enable 33
#define Motor 25


//Digital Signals2
```

```
#define Direction2 26
#define Enable2 27
#define Motor2 14


//Hall Sensors
#define HallSensorU 34
#define HallSensorV 39
#define HallSensorW 36

//Hall Sensors2
#define HallSensorX 23
#define HallSensorY 22
#define HallSensorZ 21

//Botao de comeco
#define Botao 35

extern volatile int hallSensorU_Count;
extern volatile int hallSensorV_Count;
extern volatile int hallSensorW_Count;

extern volatile bool hallSensorU_triggered;
extern volatile bool hallSensorV_triggered;
extern volatile bool hallSensorW_triggered;


extern volatile int hallSensorX_Count;
extern volatile int hallSensorY_Count;
extern volatile int hallSensorZ_Count;

extern volatile bool hallSensorX_triggered;
extern volatile bool hallSensorY_triggered;
extern volatile bool hallSensorZ_triggered;



extern bool HSU_Val;
extern bool HSV_Val;
extern bool HSW_Val;

extern int direct_m1;
extern int pulseCount_m1;

extern bool HSX_Val;
extern bool HSY_Val;
extern bool HSZ_Val;

extern int direct_m2;
extern int pulseCount_m2;


```

```
92 //Funcoes
93 void hallU_ISR();
94 void hallV_ISR();
95 void hallW_ISR();
96
97 void hallX_ISR();
98 void hallY_ISR();
99 void hallZ_ISR();
100
101 void init_gpio();
102 void init_ledc_channel();
103
104 void init_controle();
105 void control_loop();
106 void control();
107 void motor_start(double spd1, double spd2);
108 void inverseKinematics(double x, double y, double *theta1, double
      *theta2);
109 void read_next_value();
110
111 #endif
```

A.2 Arquivo de Configurações (Configuracao.c)

Código A.2 – Código em C

```
1  #include "Controle.h"
2
3  volatile int hallSensorU_Count = 0;
4  volatile int hallSensorV_Count = 0;
5  volatile int hallSensorW_Count = 0;
6
7  volatile bool hallSensorU_triggered = false;
8  volatile bool hallSensorV_triggered = false;
9  volatile bool hallSensorW_triggered = false;
10
11
12 volatile int hallSensorX_Count = 0;
13 volatile int hallSensorY_Count = 0;
14 volatile int hallSensorZ_Count = 0;
15
16 volatile bool hallSensorX_triggered = false;
17 volatile bool hallSensorY_triggered = false;
18 volatile bool hallSensorZ_triggered = false;
19
20
21 bool HSU_Val = false;
22 bool HSV_Val = false;
23 bool HSW_Val = false;
24
```

```
bool HSX_Val = false;
bool HSY_Val = false;
bool HSZ_Val = false;


void IRAM_ATTR hallU_ISR(){
    hallSensorU_Count++;
    HSU_Val = gpio_get_level(HallSensorU);
    HSW_Val = gpio_get_level(HallSensorW);
    direct_m1 = (HSU_Val == HSW_Val) ? 1 : -1;
    pulseCount_m1+=1;
}
void IRAM_ATTR hallV_ISR(){
    hallSensorV_Count++;
    HSV_Val = gpio_get_level(HallSensorV);
    HSU_Val = gpio_get_level(HallSensorU);
    direct_m1 = (HSV_Val == HSU_Val) ? 1 : -1;
    pulseCount_m1+=1;
}
void IRAM_ATTR hallW_ISR(){
    hallSensorW_Count++;
    HSW_Val = gpio_get_level(HallSensorW);
    HSV_Val = gpio_get_level(HallSensorV);
    direct_m1 = (HSW_Val == HSV_Val) ? 1 : -1;
    pulseCount_m1+=1;
}


void IRAM_ATTR hallX_ISR(){
    hallSensorX_Count++;
    HSX_Val = gpio_get_level(HallSensorX);
    HSZ_Val = gpio_get_level(HallSensorZ);
    direct_m2 = (HSX_Val == HSZ_Val) ? 1 : -1;
    pulseCount_m2+=1;
}
void IRAM_ATTR hallY_ISR(){
    hallSensorY_Count++;
    HSY_Val = gpio_get_level(HallSensorY);
    HSX_Val = gpio_get_level(HallSensorX);
    direct_m2 = (HSY_Val == HSX_Val) ? 1 : -1;
    pulseCount_m2+=1;
}
void IRAM_ATTR hallZ_ISR(){
    hallSensorZ_Count++;
    HSZ_Val = gpio_get_level(HallSensorZ);
    HSY_Val = gpio_get_level(HallSensorY);
    direct_m2 = (HSZ_Val == HSY_Val) ? 1 : -1;
    pulseCount_m2+=1;
}


//PWM CONFIG
```

```
77  void init_ledc_channel() {
78      ledc_channel_config_t ledc_channel_motor1 = {
79          .speed_mode = LEDC_HIGH_SPEED_MODE,
80          .channel = LEDC_CHANNEL_0,
81          .timer_sel = LEDC_TIMER_0,
82          .intr_type = LEDC_INTR_DISABLE,
83          .gpio_num = Motor,
84          .duty = 0,
85          .hpoint = 0
86      };
87
88      ledc_channel_config_t ledc_channel_motor2 = {
89          .speed_mode = LEDC_HIGH_SPEED_MODE,
90          .channel = LEDC_CHANNEL_1,
91          .timer_sel = LEDC_TIMER_0,
92          .intr_type = LEDC_INTR_DISABLE,
93          .gpio_num = Motor2,
94          .duty = 0,
95          .hpoint = 0
96      };
97
98      ledc_timer_config_t ledc_timer = {
99          .speed_mode = LEDC_HIGH_SPEED_MODE,
100         .duty_resolution = LEDC_TIMER_8_BIT,
101         .timer_num = LEDC_TIMER_0,
102         .freq_hz = 5000,
103         .clk_cfg = LEDC_AUTO_CLK
104     };
105     ledc_channel_config(&ledc_channel_motor1);
106     ledc_channel_config(&ledc_channel_motor2);
107     ledc_timer_config(&ledc_timer);
108     ESP_ERROR_CHECK(ledc_timer_config(&ledc_timer));
109     ESP_ERROR_CHECK(ledc_channel_config(&ledc_channel_motor1));
110     ESP_ERROR_CHECK(ledc_channel_config(&ledc_channel_motor2));
111 }
112
113
114 //Ports Setup
115 void init_gpio(){
116     ESP_ERROR_CHECK(gpio_install_isr_service(0));
117     gpio_pad_select_gpio(Enable);
118     gpio_set_direction(Enable,GPIO_MODE_OUTPUT);
119     gpio_pad_select_gpio(Direction);
120     gpio_set_direction(Direction,GPIO_MODE_INPUT_OUTPUT);
121
122     gpio_pad_select_gpio(Enable2);
123     gpio_set_direction(Enable2,GPIO_MODE_OUTPUT);
124     gpio_pad_select_gpio(Direction2);
125     gpio_set_direction(Direction2,GPIO_MODE_INPUT_OUTPUT);
126
127     // Configure Hall sensor pins as inputs
128     gpio_pad_select_gpio(HallSensorU);
```

```
129     gpio_set_direction(HallSensorU, GPIO_MODE_INPUT);
130     gpio_pad_select_gpio(HallSensorV);
131     gpio_set_direction(HallSensorV, GPIO_MODE_INPUT);
132     gpio_pad_select_gpio(HallSensorW);
133     gpio_set_direction(HallSensorW, GPIO_MODE_INPUT);
134 
135     gpio_pad_select_gpio(HallSensorX);
136     gpio_set_direction(HallSensorX, GPIO_MODE_INPUT);
137     gpio_pad_select_gpio(HallSensorY);
138     gpio_set_direction(HallSensorY, GPIO_MODE_INPUT);
139     gpio_pad_select_gpio(HallSensorZ);
140     gpio_set_direction(HallSensorZ, GPIO_MODE_INPUT);
141 
142 
143     gpio_set_intr_type(HallSensorU, GPIO_INTR_ANYEDGE);
144     gpio_isr_handler_add(HallSensorU, hallU_ISR, NULL);
145     gpio_set_intr_type(HallSensorV, GPIO_INTR_ANYEDGE);
146     gpio_isr_handler_add(HallSensorV, hallV_ISR, NULL);
147     gpio_set_intr_type(HallSensorW, GPIO_INTR_ANYEDGE);
148     gpio_isr_handler_add(HallSensorW, hallW_ISR, NULL);
149 
150     gpio_set_intr_type(HallSensorX, GPIO_INTR_ANYEDGE);
151     gpio_isr_handler_add(HallSensorX, hallX_ISR, NULL);
152     gpio_set_intr_type(HallSensorY, GPIO_INTR_ANYEDGE);
153     gpio_isr_handler_add(HallSensorY, hallY_ISR, NULL);
154     gpio_set_intr_type(HallSensorZ, GPIO_INTR_ANYEDGE);
155     gpio_isr_handler_add(HallSensorZ, hallZ_ISR, NULL);
156 
157     //Config BOTAO
158     gpio_config_t io_conf = {
159         .pin_bit_mask = (1ULL << Botao),
160         .mode = GPIO_MODE_INPUT,
161         .intr_type = GPIO_INTR_ANYEDGE,
162         .pull_up_en = GPIO_PULLUP_ENABLE,
163         .pull_down_en = GPIO_PULLDOWN_DISABLE,
164     };
165     gpio_config(&io_conf);
166 
167 }
```

A.3 Arquivo principal de Controle (`Controle.c`)

Código A.3 – Código em C

```
1  #include "Controle.h"
2 
3  //Variaveis do Controle.c
4  const int motor_steps = 72; //24 para cada sensor -> 3*24 = 72
5  const int reduction = 100;
6 
```

```
7  //Decidido empiracamente
8  double Kp = 2.03;
9  double Kd = 3.28;
10
11 double step_to_deg = (motor_steps*reduction)/360;
12
13 //Motor1
14 int desired_step_m1 = 0.0;
15 int direct_m1 = 1;
16 int pulseCount_m1 = 0;
17 double desired_angle_m1 = 0.0;
18 double angle_m1 = 0.0;
19 double error_m1 = 0.0;
20 double prev_error_m1 = 0.0;
21 bool finalizou_controle_m1 = 0;
22
23
24
25 //Motor2
26 int desired_step_m2 = 0.0;
27 double desired_angle_m2 = 0.0;
28 double angle_m2 = 0.0;
29 double error_m2 = 0.0;
30 double prev_error_m2 = 0.0;
31 int direct_m2 = 1;
32 int pulseCount_m2 = 0;
33 bool finalizou_controle_m2 = 0;
34
35 double Quadril[] = {35.76,39.22,-12.43,40.0,39.22};
36 double Joelho [] = {0.0,24.75,0.0,75.58,0.0};
37
38 double Quadril_Pulsos[5];
39 double Joelho_Pulsos[5];
40
41 double Quadril_Pulsos_Movimento[5];
42 double Joelho_Pulsos_Movimento[5];
43
44
45
46 int Contador_Trajeto = 0;
47
48 //Funcoes
49 void control();
50 void control_loop();
51 void motor_start(double spd1, double spd2);
52 void Pulsos_Generator();
53
54
55
56 void app_main() {
57     init_ledc_channel();
58     init_gpio();
```

```
59      Pulsos_Generator();
60      int contador = 0;
61
62
63      while (true) {
64          //Botão usado para iniciar e finalizar o código
65          if (gpio_get_level(Botao) == 1) {
66              contador++;
67          }
68          if(contador==1){
69              vTaskDelay(500/portTICK_PERIOD_MS);
70          }else if(contador==2){
71              while(Contador_Trajeto<5){
72                  init_controle();
73                  //vTaskDelay(2500/portTICK_PERIOD_MS);
74                  Contador_Trajeto++;
75              }
76          }
77          if(contador == 3){
78              break;
79          }
80      }
81
82
83  }
84
85
86
87
88
89  void init_controle(){
90      //Motor1
91      hallSensorU_Count = 0;
92      hallSensorV_Count = 0;
93      hallSensorW_Count = 0;
94      finalizou_controle_m1 = 0;
95      pulseCount_m1=0;
96      angle_m1 = pulseCount_m1/step_to_deg;
97      desired_angle_m1 = 0.0;
98      angle_m1 = 0.0;
99      desired_step_m1 = 0.0;
100     error_m1 = 0;
101
102
103     //Motor2
104     hallSensorX_Count = 0;
105     hallSensorY_Count = 0;
106     hallSensorZ_Count = 0;
107     finalizou_controle_m2 = 0;
108     angle_m2 = pulseCount_m2/step_to_deg;
109     pulseCount_m2=0;
110     desired_angle_m2 = 0.0;
```

```
111     angle_m2 = 0.0;
112     desired_step_m2 = 0.0;
113     error_m2 = 0;
114 
115 
116     desired_angle_m1 = Quadril_Pulsos_Movimento[Contador_Trajeto];
117 
118     int aux = gpio_get_level(Direction);
119 
120     if(desired_angle_m1<0){
121         desired_angle_m1*=-1;
122         Quadril_Pulsos_Movimento[Contador_Trajeto]*=-1;
123         gpio_set_level(Direction,CW);
124     }else{
125         gpio_set_level(Direction,CCW);
126     }
127 
128     desired_angle_m2 = Joelho_Pulsos_Movimento[Contador_Trajeto];
129     if(desired_angle_m2<0){
130         desired_angle_m2*=-1;
131         gpio_set_level(Direction2,CW);
132     }else{
133         gpio_set_level(Direction2,CCW);
134     }
135 
136 
137     vTaskDelay(50/portTICK_PERIOD_MS);
138     control_loop();
139 }
140 
141 void control_loop(){
142     while(!finalizou_controle_m1 && !finalizou_controle_m2){
143         control();
144         vTaskDelay(100/portTICK_PERIOD_MS);
145     }
146     finalizou_controle_m1 = 0;
147     pulseCount_m1=0;
148     desired_angle_m1 = 0.0;
149     angle_m1 = 0.0;
150     desired_step_m1 = 0.0;
151 
152 
153     finalizou_controle_m2 = 0;
154     pulseCount_m2=0;
155     desired_angle_m2 = 0.0;
156     angle_m2 = 0.0;
157     desired_step_m2 = 0.0;
158 
159 }
160 
161 void control(){
162 
```

```
163     int desired_step_m1 =
           Quadril_Pulsos_Movimento[Contador_Trajeto];
164     int desired_step_m2 =
           Joelho_Pulsos_Movimento[Contador_Trajeto];
165
166     error_m1 = desired_step_m1 - pulseCount_m1;
167     double pid_out_m1 = error_m1 * Kp + Kd *
           (error_m1-prev_error_m1);
168     if(prev_error_m1==error_m1 || error_m1<12){
169         pid_out_m1 = 0;
170     }
171     prev_error_m1 = error_m1;
172     if(pid_out_m1>0){
173         if(pid_out_m1>100){
174             pid_out_m1 = 100;
175         }else if(pid_out_m1 < 10){
176             pid_out_m1 = 10;
177         }
178     }else{
179         if(pid_out_m1<-100){
180             pid_out_m1 = -100;
181         }else if(pid_out_m1 > -10 && pid_out_m1<0){
182             pid_out_m1 = -10;
183         }
184     }
185
186
187
188     error_m2 = desired_step_m2 - pulseCount_m2;
189     double pid_out_m2 = error_m2 * Kp + Kd *
           (error_m2-prev_error_m2);
190     if(prev_error_m2==error_m2 || error_m2<12){
191         pid_out_m2 = 0;
192     }
193     prev_error_m2 = error_m2;
194     if(pid_out_m2>0){
195         if(pid_out_m2>100){
196             pid_out_m2 = 100;
197         }else if(pid_out_m2 < 10){
198             pid_out_m2 = 10;
199         }
200     }else{
201         if(pid_out_m2<-100){
202             pid_out_m2 = -100;
203         }else if(pid_out_m2 > -10 && pid_out_m2<0){
204             pid_out_m2 = -10;
205         }
206     }
207
208     motor_start(pid_out_m1,pid_out_m2);
209 }
210
```

```
211 void motor_start(double spd1, double spd2){
212     if(spd1!=0){
213         gpio_set_level(Enable,on);
214         double out_m1 = 255*(spd1+100)/200;
215         if(out_m1==0){
216             if(error_m1>12){
217                 out_m1=25;
218             }
219         }else if(out_m1>230){
220             out_m1 = 230;
221         }
222         if(error_m1<100){
223             out_m1 = 25;
224         }
225         if(error_m1<12){
226             out_m1=0;
227         }
228         ledc_set_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_0,
               out_m1);
229         ledc_update_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_0);
230     }else{
231         gpio_set_level(Enable,off);
232         finalizou_controle_m1 = 1;
233     }
234 
235 
236     if(spd2!=0){
237         gpio_set_level(Enable2,on);
238         double out_m2 = 255*(spd2+100)/200;
239         if(out_m2==0){
240             if(error_m2>10){
241                 out_m2=25;
242             }
243         }else if(out_m2>230){
244             out_m2 = 230;
245         }
246         if(error_m2<100){
247             out_m2=25;
248         }
249         if(error_m2<10){
250             out_m2=0;
251         }
252         ledc_set_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_1,
               out_m2);
253         ledc_update_duty(LEDC_HIGH_SPEED_MODE, LEDC_CHANNEL_1);
254     }else{
255         gpio_set_level(Enable2,off);
256         finalizou_controle_m2 = 1;
257 
258     }
259 }
260 
```

```
261
262 void Pulsos_Generator(){
263     for(int i=0;i<5;i++){
264         Quadril_Pulsos[i] = Quadril[i]*step_to_deg;
265         Joelho_Pulsos[i] = Joelho[i]*step_to_deg;
266     }
267     for(int i=0;i<5;i++){
268         if(i==0){
269             Quadril_Pulsos_Movimento[i] = Quadril_Pulsos[i];
270             Joelho_Pulsos_Movimento[i] =
                    Joelho_Pulsos_Movimento[i];
271         }else{
272             Quadril_Pulsos_Movimento[i] = Quadril_Pulsos[i] -
                    Quadril_Pulsos[i-1];
273             Joelho_Pulsos_Movimento[i] = Joelho_Pulsos[i] -
                    Joelho_Pulsos[i-1];
274         }
275     }
276
277 }
```

A.4 Modelagem e Simulação do Exoesqueleto

Código A.4 – Código em liguagem de Matlab

```
1  robot = SerialLink([0 0 50 0; 0 0 45.5 0], 'name', 'Exosuit');
2  q = [0, 0]; % Posição inicial vertical para baixo
3  figure;
4  ax = axes;
5
6
7  quadril = ([35.76,36.50,38.06,39.22,36.89,29.90,23.30,
8  15.92,7.77,3.50,-3.88,-7.77,-10.10,-12.43,-9.71,-4.27,
9  6.21,15.15,24.47,33.01,38.06,40.0,38.84,37.28,35.73]);
10
11 joelho = ([0,4.27,12.04,18.64,22.91,25.24,21.75,16.70,
12 12.81,9.32,5.05,0.0,5.05,17.48,29.13,38.45,50.10,58.25,
13 61.75,58.64,50.10,36.89,20.97,9.32,0.0]);
14
15 %quadril = [35.76,39.22,-12.43,40.0,39.22];
16 %joelho = [0.0,24.75,0.0,75.58,0.0];
17 for i = 1:length(quadril)
18     quadril(i) = 270-quadril(i);
19 end
20
21 for i = 1:length(quadril)
22     quadril(i) = deg2rad(quadril(i));
23 end
24 for i = 1:length(joelho)
25     joelho(i) = deg2rad(-joelho(i));
```

```
26 end
27
28 for i = 1:length(quadril)
29     T = robot.fkine([quadril(i), joelho(i)]);
30     robot.plot([quadril(i), joelho(i)], 'delay', 0.1);
31     title('Configuração do Manipulador');
32     pause(0.1)
33     %disp(T)
34 end
```