Universidade de Brasília
Faculdade de Tecnologia

Estudo do controle da trajetória de uma perna de exoesqueleto

Alexandre Bernardi Peres

TRABALHO DE GRADUAÇÃO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2023

Universidade de Brasília
Faculdade de Tecnologia

Estudo do controle da trajetória de uma perna de exoesqueleto

Alexandre Bernardi Peres

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Walter de Britto Vidal Filho

Brasília
2023

Bernardi Peres, Alexandre.
B523e Estudo do controle da trajetória de uma perna de exoesqueleto / Alexandre Bernardi Peres; orientador Walter de Britto Vidal Filho. -- Brasília, 2023.
68 p.

Trabalho de Graduação (Engenharia de Controle e Automação) -- Universidade de Brasília, 2023.

1. Exoesqueleto. 2. Órtese mecânica. 3. Reabilitação. 4. Acidente Vascular Cerebral. I. de Britto Vidal Filho, Walter, orient. II. Título

Universidade de Brasília
Faculdade de Tecnologia

Estudo do controle da trajetória de uma perna de exoesqueleto

Alexandre Bernardi Peres

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 16 de Fevereiro de 2023:

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

Prof. Dr. Jones Yudi Mori Alves da Silva,
UnB/FT/ENM
Examinador interno

Prof. Dr. Guilherme Caribé de Carvalho,
UnB/FT/ENM
Examinador interno

Brasília
2023

Este trabalho é dedicado à todas as pessoas que sonham e contribuem com um futuro melhor para todos.

Agradecimentos

Agradeço primeiramente à toda a minha família e namorada que me apoiaram durante a realização do projeto. Agradeço também todo o suporte dado pela Universidade de Brasília e Faculdade de Tecnologia e ao Professor Dr. Walter de Britto pela orientação e paciência durante todo o processo.

Por fim agradeço também meus colegas do curso que como ninguém entendem os desafios vivenciados, que tornaram e continuam a tornar minha vida acadêmica uma experiência prazerosa. Em especial ao Rafael Fernandes Barbosa e Mateus Berardo de Souza Terra que concederam seu tempo e conhecimentos para me auxiliar no projeto dentro do possível.

“Like everyone, to get to where I am, I stood on the shoulders of giants. My life was built on a foundation of parents, coaches, and teachers.”

(Arnold Schwarzenegger)

Resumo

O projeto consiste em realizar um estudo sobre tecnologias de reabilitação visando a continuação de um exoesqueleto de alta qualidade e acessível à comunidade. Foram realizados estudos envolvendo sensores inerciais para a captura de dados visando a replicação do movimento ao membros inferiores comprometido, o estudo da marcha humana, o movimento de motores e reflexões a respeito da melhor maneira de simular a reabilitação de um usuário em reabilitação após um AVC.

Foi dado continuidade ao projeto e novos estudos e implementações foram realizados.

Palavras-chave: Exoesqueleto. Órtese mecânica. Reabilitação. Acidente Vascular Cerebral.

Abstract

The project consists of carrying out a study on rehabilitation technologies aimed at giving continuation to a high quality exoskeleton accessible to the community. Studies were carried out involving inertial sensors to capture data aimed at the replication of the movement of the compromised lower limb, the study of human gait, the movement of motors and reflections about the best way to simulate the rehabilitation of a user undergoing rehabilitation after a stroke.

The project was continued and new studies and implementations were carried out.

Keywords: Exoskeleton. Mechanical orthosis. Rehabilitation. Stroke.

Lista de ilustrações

Sumário

1 Introdução

Pesquisas realizadas pelo IBGE indicam que cerca de 7,8 milhões de brasileiros, ou 3,8% da população acima de 2 anos de idade sofrem com alguma deficiência nos membros inferiores, havendo assim a necessidade do surgimento de tecnologias assistivas para auxiliar na marcha e possivelmente a reabilitação de tal parcela da população não só do Brasil como do mundo (PAULINE ALMEIDA, 2021).

Diversos podem ser os motivos que levam uma pessoa a necessitar de reabilitação de seus membros inferiores, sendo as principais causas acidentes medulares e AVCs, sendo esse último responsável por mais de 100 mil mortes por ano apenas no Brasil de acordo com o Ministério da Saúde e é a segunda principal causa de mortes no mundo e a principal causa de incapacidade em adultos. O AVC ou Acidente Vascular Cerebral ocorre quando há uma alteração súbita do fluxo sanguíneo no encéfalo, depravando essa região de oxigênio e resultando em diversas complicações a depender da região e do tempo que a mesma foi comprometida. As principais consequências do AVC vistas em pacientes são citadas abaixo (MARTINS, 2020):

- Alterações na fala;
- Alterações de compreensão;
- Alterações na maneira como a pessoa pensa ou sente o mundo ao seu redor;
- Alterações e/ou perda de movimentos e sensibilidade.

Tendo em mente as alterações e perdas de movimento e sensibilidade que muitas vezes ocorrem nos membros inferiores é necessário ter uma solução acessível para a reabilitação da caminhada e explorar novas possibilidades para pessoas que podem ser reabilitadas.

Como mencionado em (AGUIAR, 2017), na primeira semana após um AVC apenas cerca de 30% das pessoas conseguem realizar a marcha de forma independente, e após seis meses esse número pode chegar à 85%, demonstrando que a capacidade de reabilitação desses pacientes não é só possível como esperada.

A reabilitação de pessoas com dificuldade motora é de grande importância em diferentes níveis. Para o paciente simboliza a chance de retornar à uma vida ativa de maneira plena, sem ter que lidar com olhares curiosos, perda de oportunidades e experiências pessoais e profissionais. Já para a comunidade pode representar o retorno de força de trabalho para diversas áreas.

Pode-se fazer um argumento para o uso de cadeira de rodas, visto que são uma opção atualmente muito mais acessível financeiramente, entretanto, o usuário deve estar preparado para ser inserido em um meio completamente desfavorável para o seu deslocamento. Cerca de 44% das pessoas com alguma deficiência já foram impossibilitadas de comparecer a uma entrevista de emprego, com 63% mencionando a má qualidade das calçadas e 36% a falta de infraestrutura e transporte público não adaptado (CARMO, 2023). Dentre os problemas de adaptação de transporte público é importante ressaltar que dos municípios que contam com ônibus intramunicipal, apenas 11,7% possuem frotas totalmente adaptadas (S. PAULO, 2018).

O presente trabalho de graduação visa dar continuação aos estudos de uma perna de exoesqueleto em desenvolvimento na Universidade de Brasília a fim de contribuir para o crescimento de uma tecnologia tão necessária.

1.1 Objetivo Geral

O projeto tem como objetivo a reprodução de movimentos realizados por uma perna saudável de um usuário em uma perna de exoesqueleto por meio da captura, armazenamento e envio de dados aos motores elétricos responsáveis pelo deslocamento dos elos, de modo que os movimentos do exoesqueleto seja o mais similar possível ao do movimento realizado pela perna saudável.

Esse exoesqueleto poderá ser utilizado futuramente para a reabilitação de pacientes acometidos por um acidente vascular cerebral (AVC), podendo acelerar o processo de reabilitação do paciente de forma mais dinâmica e eficiente.

1.2 Objetivos Expecíficos

- Estudar projetos realizados anteriormente relevantes ao projeto atual;
- Estudo das estapas da marcha humana;
- Utilizar sensores inerciais a fim de mapear a trajetória percorrida no ciclo da marcha humana;
- Utilizar os sensores de Efeito Hall presentes nos motores para se ter controle da posição, velocidade e direção dos mesmos;
- Controlar o funcionamento dos motores disponibilizados com o uso de sinal PWM em conjunto com os sensores de Efeito Hall;
- Utilizar o microcontrolador Arduino Mega 2560 para interpretar os sensores inerciais e de Efeito Hall.

- Controlar a trajetória angular percorrida pela perna de exoesqueleto.

2 Revisão Bibliográfica

Para fins de reabilitação de pacientes, é necessário estudar a categoria de exoesqueletos conhecidos como *hardsuits*, que consistem em estruturas rígidas e fixadas ao paciente como é o exemplo do *Hardiman*, que pode ser visto na figura 1, e têm como objetivo fornecer apoio extra para auxiliar na reabilitação ou mesmo compensar dificuldades motoras com o uso de atuadores.

Muitos métodos já foram desenvolvidos e aprimorados com o passar dos anos desde a década de 1960 com a criação do *Hardiman*, considerado o primeiro exoesqueleto ativo, mesmo que este não fosse para fins de reabilitação e sim militares e para movimentos repetitivos na indústria. O Hardiman foi projetado para que pudesse levantar cerca de 680kg com facilidade ao utilizar-se de atuadores hidráulicos e pneumáticos, entretanto o projeto foi considerado um fracasso e o projeto final nunca chegou a ser testado em um humano, uma vez que realizava movimentos muito bruscos e poderiam colocar o usuário em perigo.(ELECTRIC, 2016)

Figura 1 – Hardiman (ELECTRIC, 2016)

É importante fazer uma retrospectiva de tecnologias anteriormente desenvolvidas e reparar nas novidades trazidas por cada uma para assim determinar abordagens possíveis e realistas para o projeto realizado.

2.1 Vukobratovic

O Vukobratovic, desenvolvido no *Mihajlo Pupin Institute* na Sérvia e chefiado pelo professor Miomir Vukobratovic, é considerado um dos primeiros modelos de exoesqueleto para fins de reabilitação, foi desenvolvido em 1969 e possuía apenas um grau de liberdade por perna, sendo esse atuado pneumaticamente e localizado na cintura. Em 1972 foi criado um novo modelo que pode ser visto na figura 2 acrescido de atuadores elétricos nos joelhos e tornozelos de ambas as pernas, totalizando três graus de liberdade por perna.

Os atuadores utilizados no projeto não foram amplamente documentados, entretanto acredita-se que foram escolhidos motores elétricos devido à sua grande precisão na realização de movimentos.

Por fim, apesar de ter uma grande importância histórica no contexto de exoesqueletos para reabilitação, o Vukobratovic não foi amplamente aderido ao mercado devido ao seu alto custo e complexidade.

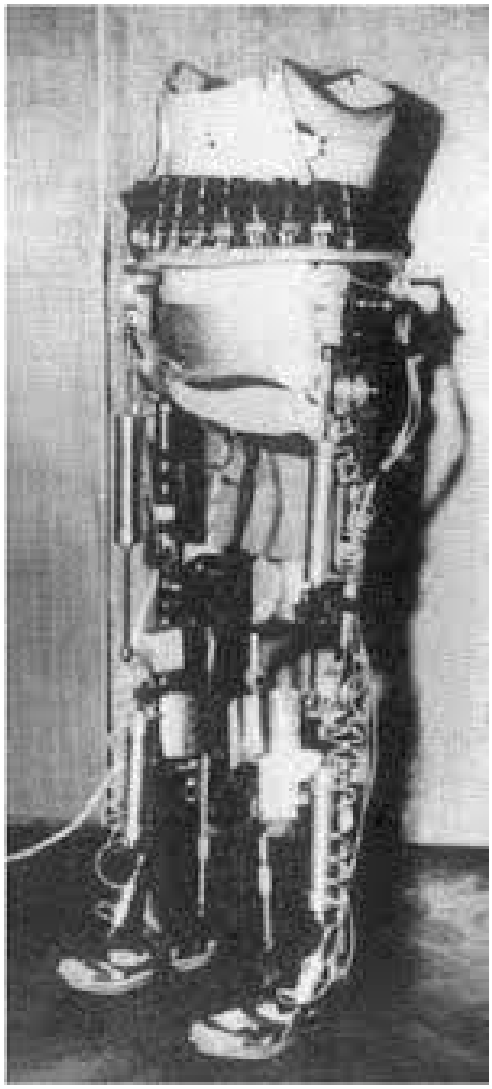

Figura 2 – Vukobratovic - 1972 (SOARES JÚNIOR, 2015)

2.2 Exoesqueleto de Wisconsin

Conhecido como exoesqueleto de Wisconsin teve seu desenvolvimento na Universidade de Wisconsin-Madison chefiado pelo professor Brooke Slavens por volta do mesmo período do modelo de Vukobratovic, entretanto foi finalizado e apresentado no ano de 1977. Foi o primeiro modelo a apresentar sete graus de liberdade, sendo três na cintura, um no joelho e três no tornozelo. O modelo utiliza motores elétricos sem escovas e sensores para gerar *feedbacks* a respeito da trajetória, e tem como grande característica a sua modularização, facilitando o deslocamento do equipamento e permitindo diferentes configurações para melhor se adequar a diferentes usuários.

O sistema de controle do exoesqueleto já era realizado através de algoritmos computacionais, que eram responsáveis por interpretar sinais provenientes dos sensores sobre a marcha realizada e equilíbrio do usuário, enviando sinais para os motores realizar os ajustes necessários para um movimento confortável ao usuário. Na figura 3 é possível ver o exoesqueleto em seus estágios iniciais e apesar da aparência o equipamento não apresenta uma grande massa.

(GRUNDMANN, 1971).

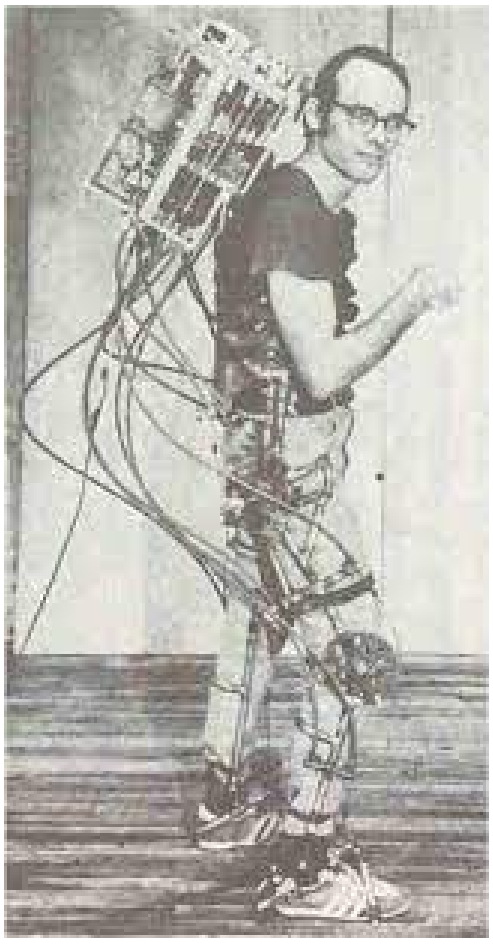

Figura 3 – Exoesqueleto de Wisconsin - (FREIRE, 2019)

2.3 MIT - *Massachusetts Institute of Technology*

O modelo de exoesqueleto do MIT teve seu desenvolvimento iniciado no começo dos anos 2000 e liderado pelo professor Hugh Herr, um grande pesquisador da área de biomecatrônica. O projeto tem como objetivo facilitar uma marcha natural à pessoas com mobilidade reduzida, necessitando assim ser leve e confortável para o uso cotidiano das pessoas. O exoesqueleto ainda segue em constante aperfeiçoamento e já passou por diversos testes clínicos e com os resultados sendo muito promissores.

O exoesqueleto utiliza motores elétricos sem escovas para movimentar as juntas do quadril e joelho do usuário além de sensores de força e posição angular para realizar ajustes em tempo real para facilitar o movimento natural da perna. Na figura 4 é possível ver a maneira como o equipamento é fixado no usuário e o posicionamento dos atuadores. (TRAFTON, 2007)

Figura 4 – Exoesqueleto do MIT - 2007 (TRAFTON, 2007)

2.4 HAL - *Hybrid Assistive Leg*

O exoesqueleto foi desenvolvido pela empresa japonesa Cyberdyne e possui modelos comerciais tanto para a reabilitação quanto para o uso pessoal. O equipamento é capaz de auxiliar usuários com a marcha e outras atividades como subir escadas.

O equipamento conta com motores elétricos e atuadores hidráulicos para a realização dos movimentos e possui como grande atrativo o uso de sensores EMG (eletromiograma), que interpretam sinais emitidos pelos músculos do usuário sendo assim capaz de assimilar o movimento desejado e ativar os atuadores para realizar o mesmo com a intensidade

necessária.

Na figura 5 é ilustrado um modelo voltado para a reabilitação dos membros inferiores, entretanto, a empresa possui também modelos mais complexos envolvendo também elos para o movimento dos membros superiores.

Figura 5 – Exoesqueleto HAL (CYBERDYNE, 2012)

2.5 Exoesqueleto do Projeto Andar de Novo

O projeto Andar de novo, chefiado por Miguel Nicolelis, é um projeto brasileiro, fruto de mais de quinze anos de pesquisa e visa a construção de exoesqueletos baseando-se no princípio interface cérebro-máquina, onde sinais emitidos pelo cérebro do usuário são captados por sensores EEG (eletroencefalografia) posicionados na cabeça do usuário e então convertidos em sinais a serem enviados aos atuadores pneumáticos na cintura e joelhos e aos motores elétricos presentes nas juntas do tornozelo, fazendo assim com que o exoesqueleto realize o movimento desejado.

A proposta foi considerada pela revista *Scientific American* como uma das dez ideias que estão além dos limites da ciência atual em uma publicação realizada no ano de 2016. O projeto foi apresentado ao mundo durante a abertura da Copa do Mundo de Futebol realizada no Brasil em 2014, onde o usuário paraplégico Juliano Pinto deu o chute inicial com o auxílio do exoesqueleto (FINEP, 2014). Na figura 6 é demonstrado o exoesqueleto utilizado na abertura do evento citado.

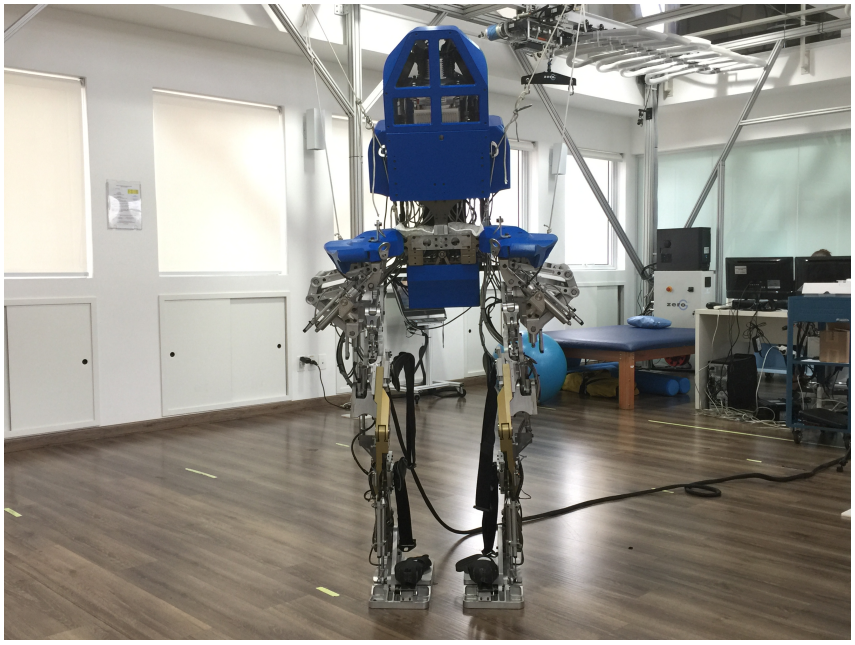

Figura 6 – Exoesqueleto do projeto Andar de Novo (FINEP, 2014)

2.6 Exoesqueleto da USP - Exoesqueleto Modular de Membros Inferiores

O projeto foi iniciado por alunos de pós-graduação da Escola de Engenharia de São Carlos em 2013 com auxílio do governo brasileiro para o desenvolvimento de uma tecnologia para auxiliar pacientes com diferentes formas de paralisia nos membros inferiores. Na figura 7 é possível ver o modelo em uma versão mais recente no ano de 2020 e o exoesqueleto continua em constante desenvolvimento.

O projeto utiliza atuadores hidráulicos nas juntas do quadril e joelho, e um atuador pneumático no tornozelo. Um grande diferencial do exoesqueleto é a sua capacidade de ser controlado a partir de um *joystick* ou equipamentos similares além de sensores para permitir correções devido à variações do equilíbrio do usuário. E assim como o exoesqueleto HAL, utiliza sensores EMG para interpretar intenções de movimento e acionar os atuadores a fim de completar o movimento apenas com o torque que o usuário não é capaz de exercer.

Figura 7 – Exoesqueleto da USP - 2020 (CHAVES, 2021)

2.7 Exoesqueleto da UnB - Projeto TAO

Um projeto de exoesqueleto está em andamento na Universidade de Brasília, no GRACO-ENM, visando o auxílio na marcha de pessoas com mobilidade afetada. Esse projeto surgiu a partir de uma iniciativa do projeto EMA, que desde 2016 trabalha em um projeto de *Transparent Active Orthosis* ou TAO. O objetivo de uma TAO é criar alternativas menos robustas de órteses de reabilitação, introduzindo opções mais diversificadas e acessíveis ao mercado. (FREIRE, 2019).

O primeiro trabalho encontrado no banco de dados da Universidade de Brasília relacionado ao exoesqueleto em questão foi o de (RODRIGUES, 2017), que desenvolveu e

implementou uma interface que possibilita o acionamento de motores e leitura de *encoders* de posição por meio de uma plataforma *System on chip*.

Já o trabalho de (BALDASSO, 2018) foi um estudo aprofundado da anatomia da marcha humana e apresentou resultados muito importantes com a utilização de sensores inerciais. Visando compreender a influência da aceleração do quadril no sistema, foram realizados dois modelos dinâmicos, onde um consideraria o quadril como um referencial inercial e o segundo leva a sua aceleração em consideração. Além disso foi realizada uma análise dos torques em cada uma das juntas dos membros inferiores, sendo esses o quadril, o joelho e o tornozelo em uma baixa velocidade, simulando uma marcha de reabilitação, sendo assim possível determinar os atuadores a serem utilizados no desenvolvimento do exo-esqueleto. Esse trabalho foi de extrema importância para o trabalho realizado em sequência por (FREIRE, 2019), onde com as informações adquiridas no trabalho de (BALDASSO, 2018) foi possível a construção do exoesqueleto em si e implementação dos atuadores e redutores, permitindo assim que um usuário obtivesse suporte na posição vertical levando em conta os critérios de rigidez estática e fadiga.

Figura 8 – Exoesqueleto da UnB (Fonte: Próprio autor)

Na figura 8 é possível notar que o exoesqueleto já se encontra com os dois motores a serem utilizados e o suporte de lombar para conforto do usuário. O exoesqueleto é suspenso e suportado por uma estrutura para que testes possam ser realizados com segurança.

3 Revisão teórica

3.1 Motores

Os motores utilizados no projeto são do modelo EC 90 (323772), que se tratam de motores de corrente contínua e sem escovas, os mesmos são acoplados a redutores planetários do modelo GP 52C (223095), ambos da Maxon Motors e podem ser vistos nas figuras 9 e 10 respectivamente.

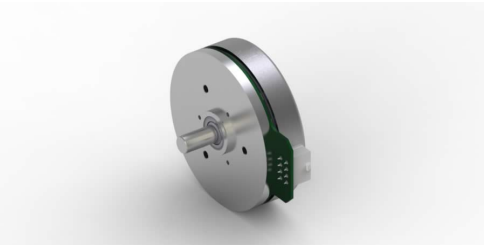

Figura 9 – Motor Flat EC 90 (323772) (MOTORS, 2017)

Figura 10 – Redutor Planetário GP 52C (223095) (MOTORS, 2021)

A tabela 1 possui informações divulgadas pelo fabricante a respeito do motor:

Tensão nominal [V]	24
Corrente [mA]	538
Velocidade nominal [rpm]	2590
Torque nominal [mN.m]	444,4
Eficiencia máxima (%)	84
Potência [W]	90
Diâmetro do motor [mm]	90

Tabela 1 – Informações a respeito do motor

A tabela 2 possui informações divulgadas pelo fabricante a respeito do redutor:

Redução	100:1
Eficiência máxima (%)	70
Peso (g)	1540
Torque máximo suportado pelo eixo (N.m)	50

Tabela 2 – Informações a respeito do redutor

3.2 Sensores

Sensores inerciais foram utilizados na perna sadia do usuário do exoesqueleto, com o propósito de obter informações em tempo real da caminhada, sendo assim possível reproduzí-los na perna afetada. É necessário que os sensores sejam capazes de medir a posição angular entre si e também manter o referencial com o solo. Dessa forma a posição da perna e sua inclinação podem ser obtidas. Sensores inerciais são ideais para esse tipo de aplicação, mais especificamente o giroscópio.

3.2.1 Giroscópio

Giroscópios são sensores capazes de medir a velocidade angular de um objeto em torno de um eixo de rotação. Por meio da velocidade angular obtida, é possível extrair a posição angular através de uma simples integração.

O problema desse processo de integração do sinal é a existência de erros, resultando em uma posição com acurácia comprometida, mas que para os fins do projeto, não são de grande significância.

3.2.2 Tecnologia MEMS (*Micro-Electro-Mechanical Systems*)

Uma vez que o giroscópio se tornou uma tecnologia indispensável em diversas áreas e produtos, como por exemplo *smartphones*, foi necessária a adaptação do mesmo para ser compatível com sistemas micro eletromecânicos.

O giroscópio representado na figura 11 funciona de maneira a explorar o efeito das forças de Coriolis, que são observadas ao expor uma massa vibrante à uma taxa de rotação (D'ANGELO, 2015).

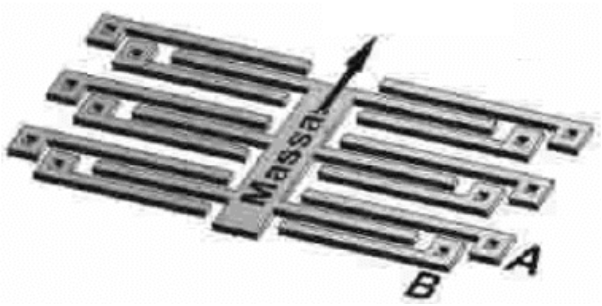

Figura 11 – Organização de um giroscópio MEMS (LOPES, 2022)

3.2.3 MPU-6050

O MPU-6050 foi o dispositivo escolhido para a aplicação devido à sua disponibilidade nos laboratórios da Universidade de Brasília, além disso é um dispositivo facilmente substituído e de baixo custo. No MPU-6050 estão presentes além do giroscópio, um acelerômetro e um magnetômetro, mas que não serão utilizados no atual projeto.

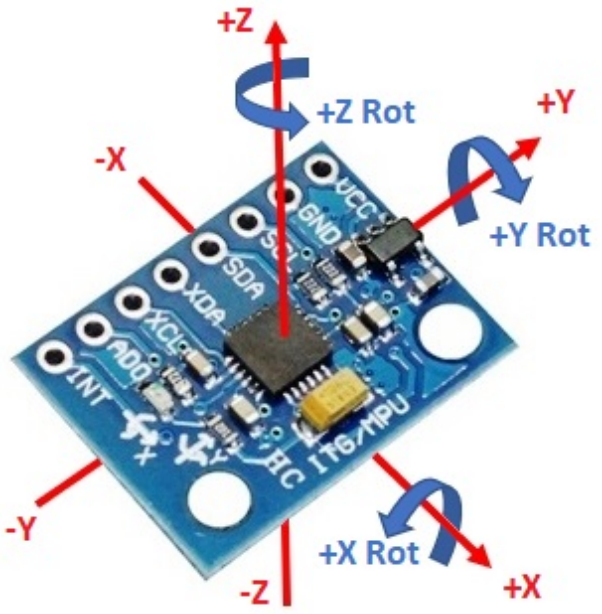

Figura 12 – MPU-6050 e seus respectivos eixos de rotação (KATHPALIA, 2022)

O MPU 6050 necessita de uma alimentação de 5V tornando-o facilmente compatível com Arduinos sem riscos de danificá-lo.

3.3 Drivers

Os *drivers* a serem utilizados são os do próprio fabricante dos motores, sendo esses do modelo Escon 70/10, que serão configurados com a utilização do software recomendado pelo fabricante, o Escon Studio.

Ao configurar os *drivers* o próprio Escon Studio aponta a configuração indicada de instalação dos motores, pontos de entradas digitais e analógicas a serem utilizados no projeto como demonstrado na figura 13.

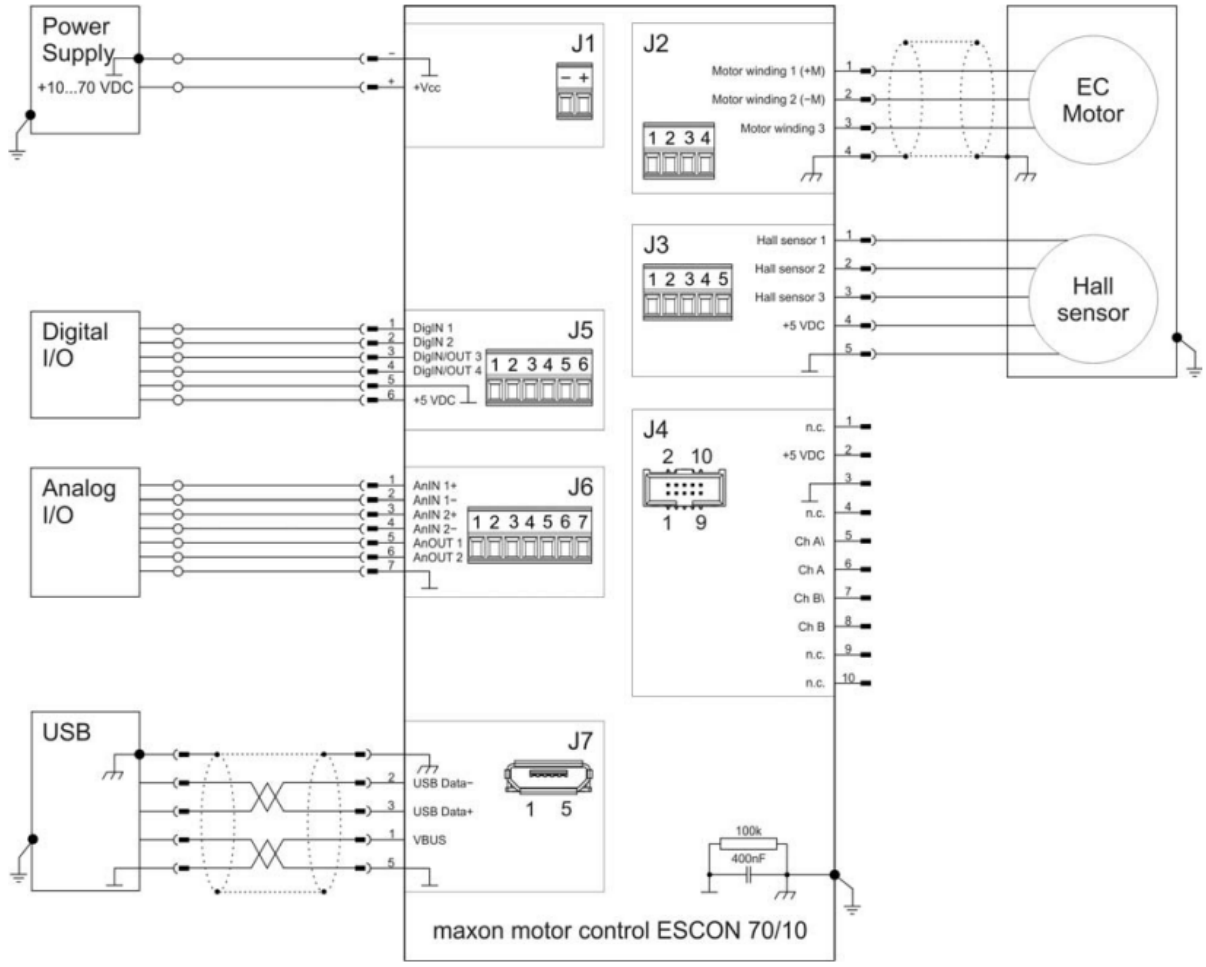

Figura 13 – Disposição do Escon 70/10 e conexão ao motor EC (MOTORS, 2017)

Os *drivers* irão receber em suas entradas digitais sinais de ligar (*enable*), o sentido de rotação e um sinal de PWM que irá realizar o controle de velocidade a partir da modulação de um sinal digital.

3.4 Arduino Mega 2560

O microcontrolador escolhido foi o Arduino Mega 2560 que além de ser uma opção acessível, atendia aos requisitos do projeto principalmente quanto à quantidade de pinos de interrupção, que são os pinos 2, 3, 18, 19, 20 e 21, que podem ser visualizados na figura 15 e foram utilizados para a conexão das MPUs e dos quatro sensores de Efeito Hall. Além disso o Arduino utilizado possui disponível uma quantidade maior de memória *flash* (256KB) quando comparado com um Arduino Uno (32KB) permitindo que códigos maiores

e mais complexos possam ser implementados. Na figura 14 é possível ver comparações mais detalhadas entre o Arduino Mega 2560, o Arduino Uno e o Arduino Micro.

	Arduino Uno	Arduino Mega 2560	Arduino Micro
Price Points	$19.99-$23.00	$36.61 - $39.00	$19.80 - $24.38
Dimension	2.7 in x 2.1 in	4 in x 2.1 in	0.7 in x 1.9 in
Processor	Atmega328P	ATmega2560	ATmega32U4
Clock Speed	16MHz	16MHz	16MHz
Flash Memory (kB)	32	256	32
EEPROM (kB)	1	4	1
SRAM (kB)	2	8	2.5
Voltage Level	5V	5V	5V
Digital I/O Pins	14	54	20
Digital I/O with PWM Pins	6	15	7
Analog Pins	6	16	12
USB Connectivity	Standard A/B USB	Standard A/B USB	Micro-USB
Shield Compatibility	Yes	Yes	No
Ethernet/Wi-FI/Bluetooth	No (a Shield/module can enable it)	No (a Shield/module can enable it)	No

Figura 14 – Comparação entre 3 modelos de Arduino (GUDINO, 2021)

Os pinos de interrupção se dão necessários pois os sensores de Efeito Hall estão sob constante atualização, e as interrupções irão fazer com que o programa, independente da posição do contador de programa, realize as operações necessárias sempre que ocorra uma variação do estado dos sensores, minimizando imprecisões a respeito da posição dos motores.

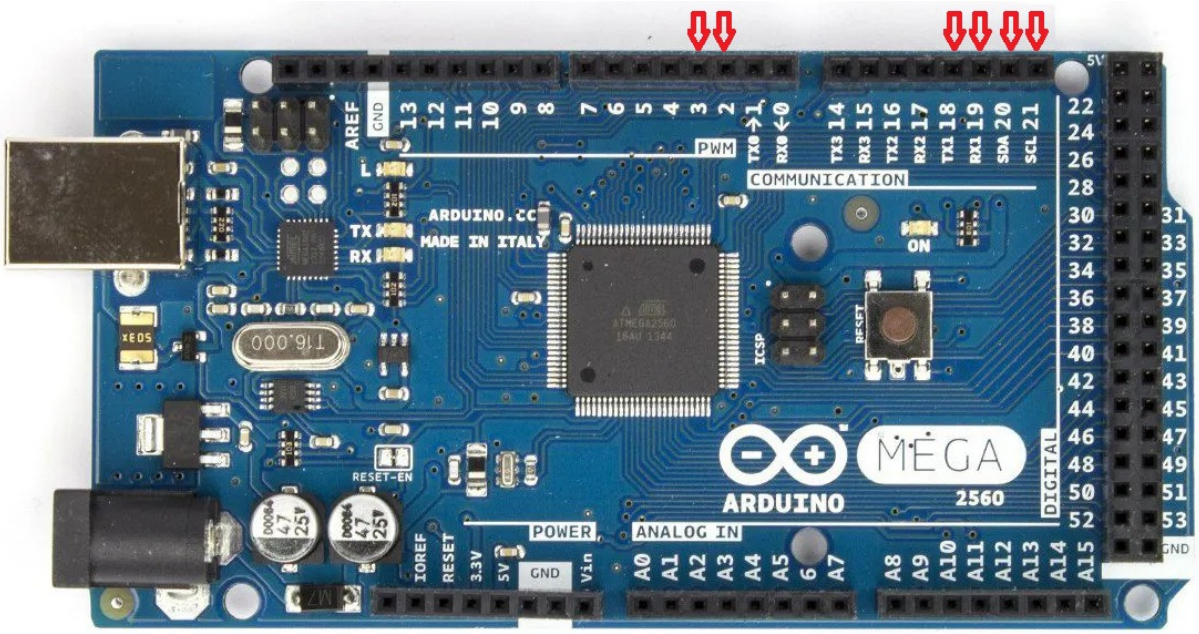

Figura 15 – Posição dos pinos de interrupção em um Arduino Mega 2560

3.5 Anatomia da caminhada

O ciclo de marcha humana pode ser separada em duas partes principais, a etapa de apoio e a de balanço. A etapa de apoio de uma perna é aquela onde o peso do indivíduo encontra-se apoiado ou parcialmente apoiado em tal perna, já a fase de balanço é quando a perna encontra-se suspensa e todo o peso do indvíduo encontra-se sobre a outra perna (MOURA, 2018).

3.5.1 Etapa de Apoio

Durante o período de apoio, a primeira ação é a de alteração do centro de massa do corpo ou também chamado de aceitação do peso, fazendo com que o peso se concentre na perna que permanecerá em contato com o solo. A segunda ação é a de apoio médio, onde a perna de apoio recebe todo o peso do corpo e a outra perna não está mais em contato com o solo. Por fim atinge-se o apoio terminal, onde a perna que estava em balanço entra em contato com o solo mas ainda o peso do corpo encontra-se principalmente na perna de apoio. O processo ocupa em média 60% do ciclo de marcha (MOURA, 2018).

3.5.2 Etapa de Balanço

Ao iniciar a etapa de balanço, a primeira ação é a de pré-balanço, onde a perna encontra-se livre para impulsionar o corpo com auxílio do tornozelo. Entra-se então na fase de balanço inicial, onde o pé perde o contato com o solo. Este é seguido pelo balanço médio, caracterizado pela passagem da perna em balanço pela perna em apoio. E finaliza-se com o balanço final, onde o calcanhar entra em contato com o solo novamente. O processo ocupa em média 40% do ciclo de marcha (MOURA, 2018).

A figura 16 ilustra as etapas de um ciclo de marcha de uma perna, além de demonstrar em média o tempo ocupado por cada uma das etapas e subetapas.

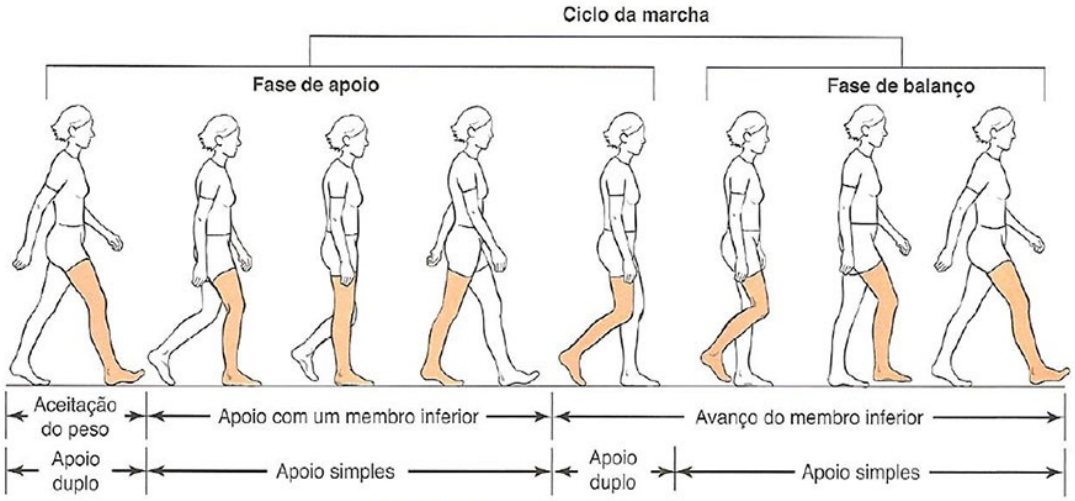

Figura 16 – Apoio e balanço de um ciclo de marcha (MOURA, 2018)

3.6 Kinovea

O movimento realizado durante um ciclo de marcha pode ser verificado com auxílio de *softwares* de vídeo. O *software* escolhido para se utilizar no projeto foi a versão 0.9.5 do Kinovea (CHARMANT, 2022), que além de ser gratuito possui ferramentas úteis para o contexto do projeto, dentre elas as principais ferramentas utilizadas foram a de rastrear um marcador que pode ser posicionado no vídeo, e a ferramenta de rastrear a variação angular entre dois pontos que podem ser selecionados.

O Kinovea possui uma boa capacidade de rastrear e desenhar a trajetória desejada, entretanto, quando utiliza-se vídeos com baixa resolução, o *software* pode afastar-se do ponto desejado, para solucionar esse problema recomenda-se utilizar pontos contrastantes no vídeo, podendo ser uma esfera de isopor em superfícies escuras ou fita isolante em superfícies claras. Mas quando isso não é possível, existe a opção de fazer ajustes manualmente, reposicionando o ponto a ser rastreado no local desejado nos quadros do vídeo onde houve deslocamento.

Para validar o *software* um teste foi realizado, onde foram posicionadas esferas de isopor em posições da perna onde era desejado rastrear a trajetória ao longo de um ciclo de marcha.

- Fêmur proximal (Junta do quadril com o fêmur);
- Centro de massa da coxa;
- Fêmur distal (Junta do fêmur com o joelho);
- Centro de massa da panturrilha;
- Tornozelo;

- Calcanhar;
- Terceiro metatarso (Osso localizado na junção entre o pé e o terceiro dedo).

Na figura 17 estão representadas as posições onde as esferas foram posicionadas para o teste.

Figura 17 – Posição das esferas de isopor na perna

No teste, para que houvesse um maior contraste, foi utilizada uma calça preta e com um fundo preto, dessa forma o *software* teve maior facilidade de identificar as esferas de isopor brancas e rastrear sua trajetória. O resultado obtido pode ser visto na figura 18.

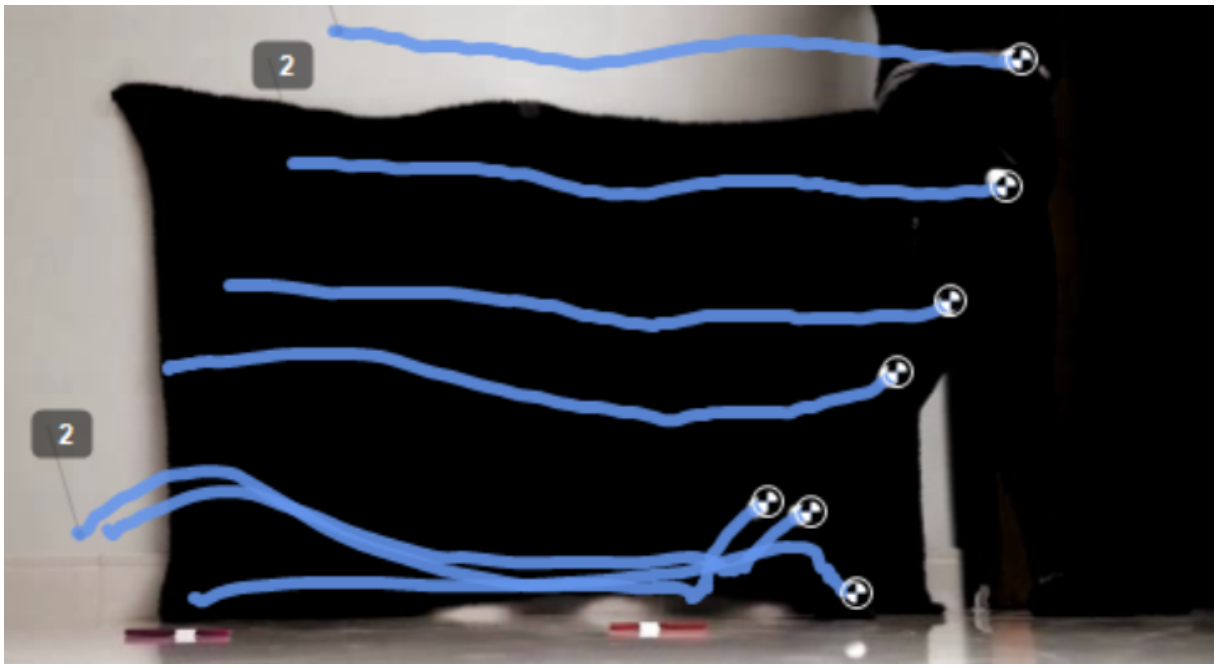

Figura 18 – Ciclo de marcha com auxílio do Kinovea (Fonte: Próprio autor)

Pode-se também verificar mais claramente a trajetória do tornozelo durante o ciclo de marcha através do uso de esteiras ergométricas.

Figura 19 – trajetória do tornozelo de um homem medindo 1,80m de altura (Fonte: Próprio autor)

Figura 20 – trajetória do tornozelo de uma mulher medindo 1,53m de altura (Fonte: Próprio autor)

Foi pedido para que os voluntários andassem a uma velocidade confortável, simulando um andar casual. O homem na figura 19 andou a uma velocidade de 5km/h, já a mulher na figura 20 andou a uma velocidade de 4km/h.

Entretanto, note que apesar da diferença de estatura entre ambos, a trajetória percorrida é extremamente similar, diferenciando-se apenas no quesito individual de como os voluntários normalmente andam. Já a diferença de velocidades na configuração da esteira ergonômica é importante para salientar que a velocidade do andar de uma pessoa é influenciado pelo comprimento da perna e da velocidade angular desenvolvida pelos elos da perna, e não necessariamente pela trajetória percorrida pela mesma. Ou seja, caso duas pessoas com comprimentos diferentes da perna desenvolvessem uma mesma velocidade angular, a pessoa com a perna mais comprida iria percorrer uma distância maior. E caso duas pessoas com o mesmo comprimento de perna desenvolvessem velocidades angulares diferentes, a pessoa com a maior velocidade também irá percorrer uma distância maior em um período de tempo.

4 Desenvolvimento inicial

4.1 Testes nos motores

Existe uma bancada, que pode ser vista na figura 21, feita por (RODRIGUES, 2017) durante seu Trabalho de Graduação e nela estão presentes uma fonte de 24V, um driver ESCON 70/10 e um motor Flat EC 90 já anexado a um redutor planetário GP 52C além de outros componentes acrescentados por outros alunos que utilizaram a bancada no passado, entretanto, os componentes a serem utilizados são mencionados previamente.

Figura 21 – Bancada de testes do motor e driver (Fonte: Próprio autor)

Na bancada foi primeiramente necessário configurar o driver através do software disponibilizado pelo fornecedor, o *Escon Studio*. A configuração foi realizada de maneira simples e majoritariamente seguindo as recomendações indicadas a cada passo, sendo feito pequenos ajustes para melhor atender as especificações do projeto.

Na configuração foram especificadas características do motor que devem se consultadas no catálogo do fabricante (MOTORS, 2017) uma vez que o *driver* é compatível com uma variedade de motores. a polaridade dos sensores de Efeito Hall, que deve ser mantida como a padrão da Maxon Motors, a resolução do *encoder*, o modo de operação, que se trata de um

loop fechado, as informações a serem recebidas pelas portas digitais, especificando como irão se comportar em caso de sinal alto ou baixo e as velocidades de giro do motor em *duty cycles* de 10% e 90% que foram definidas como 100 RPM e 900 RPM respectivamente para ambos os motores para que tenham um comportamento linear e previsível.

Foi possível realizar testes e confirmar a sua compatibilidade com o PWM do Arduino, que se mostrou suficiente para fazer o controle de velocidade do motor. O motor permite duty cycles variando entre 10% e 90% como indicado pelo fabricante, e os testes demonstraram que ao ser anexado ao redutor, velocidades muito baixas não possuem torque suficiente para movimentar o exoesqueleto, limitando assim a amplitude de velocidades que se pode obter.

Na figura 22 seguem gráficos disponibilizados pelo ESCON Studio dos testes realizados, onde em vermelho têm-se o sinal PWM enviado, e em amarelo a velocidade registrada pelo sensor de efeito hall do motor:

Figura 22 – Variação repentina de duty cycle (Fonte: Próprio autor)

Nesse primeiro experimento foi realizada a variação brusca do duty cycle do sinal de PWM, iniciando em 0%, sendo incrementado para 60%, e por fim para 90%, com um intervalo de 1 segundo entre os incrementos, e então, realizando os mesmos passos inversamente para retornar à 0%.

Com esse teste foi possível verificar o tempo de resposta do motor, ou seja, o tempo levado para que o motor atinja a nova velocidade desejada, e pôde ser observado que a velocidade, representada em amarelo, reage muito rapidamente à mudança do duty cycle.

Já na figura 23 foi realizado um teste onde o *duty-cycle* foi incrementado gradualmente entre 10% e 90%, e apesar de não acompanhar com grande precisão a curva do sinal PWM, apresentou resultados satisfatórios para a aplicação do projeto.

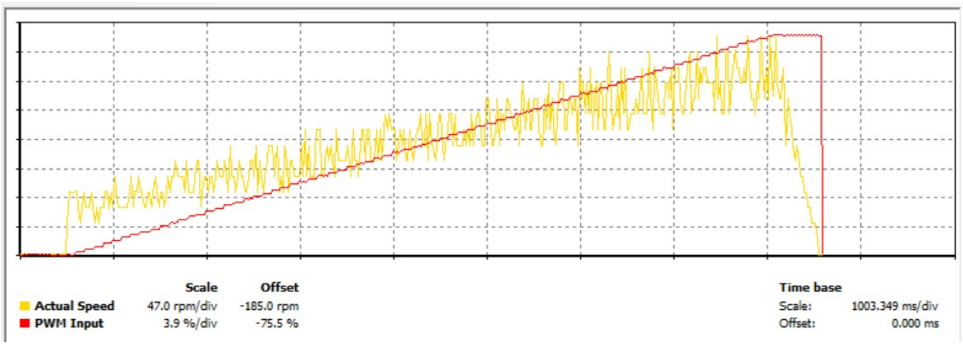

Figura 23 – Variação gradual de duty cycle (Fonte: Próprio autor)

4.2 Teste nos sensores

Os sensores testados foram duas MPU-6050, sendo testado tanto a sua função como acelerômetro quanto de giroscópio. Uma estrutura simples foi construída para verificar de maneira experimental os resultados obtidos pelos sensores:

Figura 24 – Estrutura para testar os ângulos extraídos da MPU-6050 (Fonte: Próprio autor)

Pode-se notar na figura 24 a presença de um nível, para garantir que a superfície de apoio encontra-se plana, um esquadro, para que o elo seja posicionado a um ângulo de 90°e por fim um transferidor, posicionado em posição para aferir os dados coletados da MPU.

Quando iniciado o teste, o programa primeiramente calibra a MPU de acordo com

a posição inicial, designando a essa, o ângulo de 0°, e a cada 10ms coleta um novo dado, indicando a nova posição em graus em relação à inicial.

Figura 25 – Dados indicados pela MPU na posição inicial (Fonte: Próprio autor)

Vê-se pela figura 25 que o sensor não aponta exatamente para o zero mesmo logo após a calibragem, isso se dá pela deriva inerente do *hardware*, mas essa pequena variação não apresenta grandes problemas ao projeto porque os ângulos serão tratados como inteiros, desconsiderando a parte fracionária das leituras.

Em seguida movimentou-se o equipamento para a posição que representa um ângulo de -30°como ilustrado pela figura 26, e os resultados obtidos podem ser vistos na figura 27.

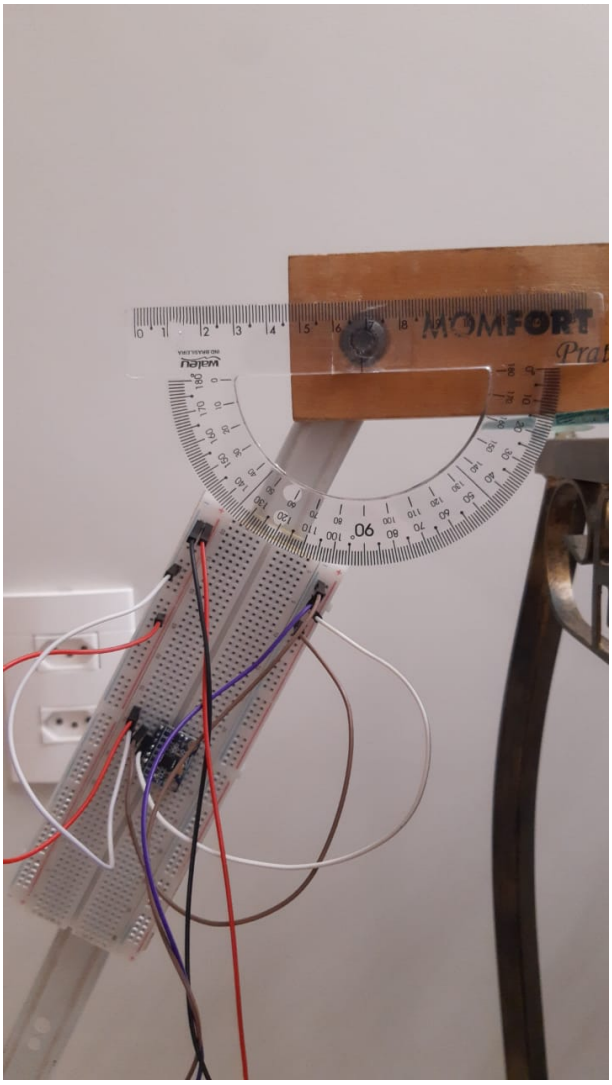

Figura 26 – posicionamento da estrutura -30°(Fonte: Próprio autor)

E ao observar os dados notou-se uma diferença que se deve tanto ao erro de *hardware* quanto o erro humano ao posicionar o elo com o transferidor utilizado.

Com os testes realizados com o motor e os sensores a serem utilizados chegou-se a conclusão que os dispositivos seriam suficientes para a aplicação desejada.

```
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
Z : -29.11
```

Figura 27 – Dados indicados pela MPU na posição -30°(Fonte: Próprio autor)

5 Integração dos componentes

5.1 Integração entre o Arduino e MPUs

O Arduino Mega 2560 se comunica com as MPUs utilizando o protocolo I2C, que permite a comunicação de um dispositivo mestre com diversos dispositivos escravos, sendo esses o Arduino e as MPUs respectivamente. A comunicação ocorre através dos pinos *Serial Data* (SDA) e *Serial Clock* (SCL), os pinos SDA e SCL do Arduino enviam aos dispositivos solicitações de comunicação e um sinal de *clock*, já os dispositivos escravos recebem o sinal de *clock* do mestre pelo pino SCL, sincronizando-os, e enviam o sinal solicitado através do pino SDA.

O Arduino possui seis pinos de interrupção, e dois deles são especializados para a comunicação I2C, esses pinos são o 20 e o 21, que interpretarão os sinais enviados por ambas as MPUs, diferenciando-as através do endereço atribuído a cada um. Além disso também foi utilizada a biblioteca *MPU6050 light* (FETICK, 2021), que possui funções para auxiliar a interpretação dos sinais recebidos dos sensores, como a função para adquirir a posição angular dos três eixos de rotação do sensor, e que pode ser facilmente instalada no próprio ambiente de desenvolvimento do Arduino. Na figura 28 está ilustrada a conexão realizada entre os componentes:

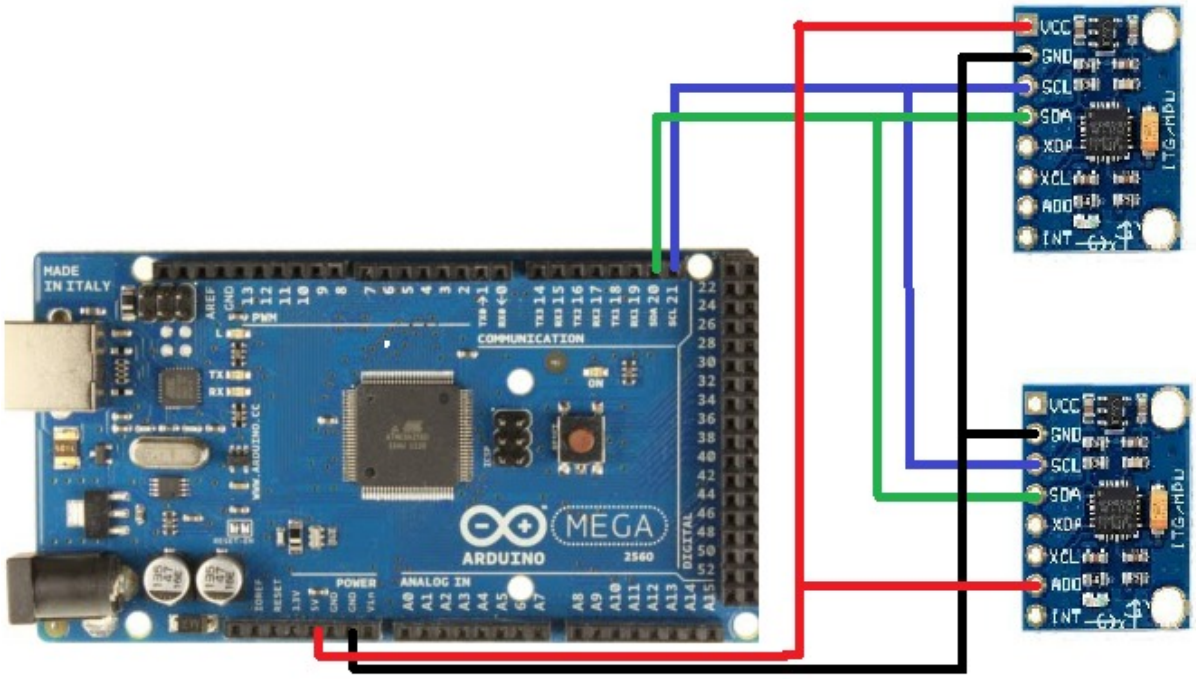

Figura 28 – Conexão entre Arduino e MPUs

Note que a segunda MPU não é conectada no VCC, e sim no pino AD0, isso estabelece

a relação mestre-escravo, possibilitando a diferenciação dos sinais recebidos pelo Arduino.

Com a conexão realizada é possível posicioná-los na coxa e na panturrilha de um usuário e verificar as variações angulares em ciclos de marcha nas figuras 29 e 30:

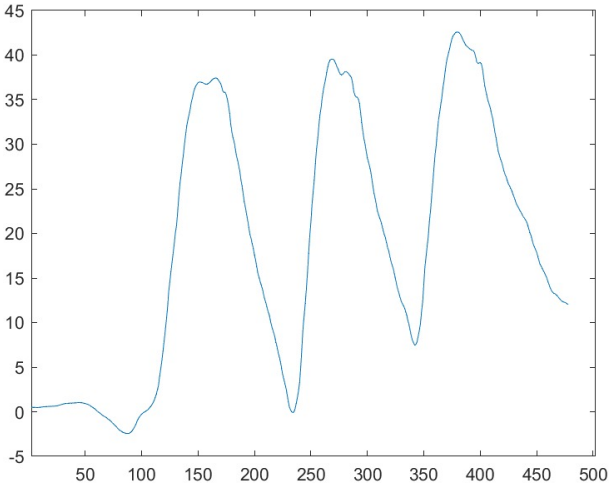

Figura 29 – Variação angular da coxa em 3 ciclos de marcha (Fonte: Próprio autor)

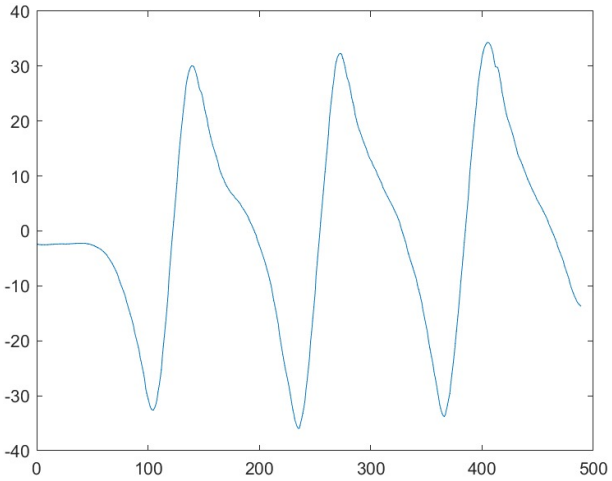

Figura 30 – Variação angular da panturrilha em 3 ciclos de marcha (Fonte: Próprio autor)

5.2 Integração entre o Arduino e Drivers

O Arduino deve enviar 3 sinais a cada um dos drivers, sendo 2 digitais e 1 sinal de PWM, que será interpretado como um sinal digital. Os dois sinais digitais compreendem aos sinais de *enable* e de direção. A maneira como ambos serão interpretados é configurável no Escon Studio, mas nesse projeto o sinal alto na entrada digital 2 do driver corresponde ao enable e habilita o motor, permitindo que se mova. Já a entrada 3 é responsável pela direção de giro, um sinal alto o fará girar no sentido horário e um sinal baixo o fará girar no sentido anti-horário. O último sinal é enviado à entrada digital 1 e receberá o sinal de PWM, recebendo um sinal entre 26 e 229, correspondendo aos *duty-cycles* de 10% e 90%

respectivamente, e esse sinal é responsável por indicar ao driver a velocidade com a qual o motor deve se mover. O sinal de PWM deve ser enviado a partir de um dos pinos do Arduino habilitados para o envio desse tipo de sinal.

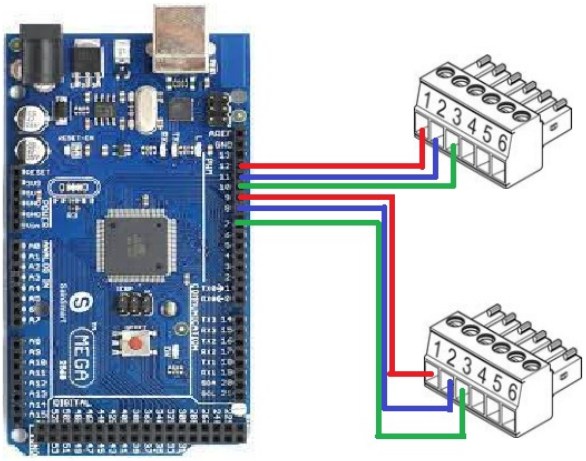

Figura 31 – Conexão entre o arduino e as entradas digitais dos drivers

5.3 Integração entre o Arduino e Sensores de Efeito Hall

Os motores utilizados são fabricados com três sensores de Efeito Hall e um *encoder*, mas por limitação da capacidade de processamento do Arduino serão utilizados apenas os sensores de Efeito Hall devido ao número reduzido de posições assumidas pelo motor. Utilizando um dos sensores de Efeito Hall é possível determinar a velocidade de giro e o número de voltas dadas pelo motor, uma vez que o número de polos no motor é conhecido, então a cada vez que o sensor detecta a passagem dos 12 polos sabe-se que foi completada uma volta, e o tempo decorrido dessa volta permite o cálculo da velocidade do mesmo.

Entretanto, é necessário saber além do número de voltas dadas, o sentido da rotação do motor para que quando o efeito da gravidade no exoesqueleto o fizer girar no sentido contrário ao desejado isso poder ser contabilizado. E para tal é necessário a leitura de um mínimo de 2 sensores de Efeito Hall em cada um dos motores. Isso é possível graças ao posicionamento dos sensores, onde, ao detectar a chegada de um polo em um dos sensores e comparando com o estado do segundo sensor, é possível determinar a direção de giro do motor.

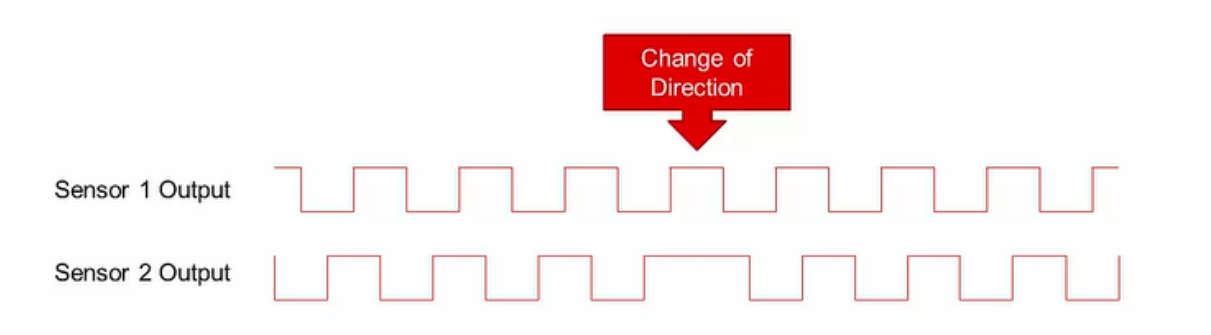

Figura 32 – Sinais enviados por sensores de Efeito Hall (HARMON, 2019)

Na figura 32 é possível notar inicialmente que quando o sensor 1 passa para o estado alto, o sensor 2 também encontra-se no mesmo estado, indicando que polo do motor em questão passou primeiramente pelo sensor 2 e depois pelo sensor 1. E quando há uma inversão de sentido, indicada pela seta em vermelho, o polo passa primeiramente pelo sensor 1 e depois pelo sensor 2, indicando que o motor gira no sentido oposto.

Sendo assim, o Arduino deverá receber em pinos digitais a saída de dois sensores de Efeito Hall por motor utilizado e utilizando a lógica descrita é feito o controle da posição dos motores.

Sabe-se que o motor possui 12 polos de acordo com as informações do fabricante, e como dois sensores estão sendo utilizados, uma volta do motor corresponde a 48 leituras de variação de estado e sabendo que o motor está acoplado a um redutor de 100:1, a relação entre o ângulo varrido pela haste do motor e o lido pelas MPUs é a que se segue:

$$Angulo_{dahaste} = \frac{48 \cdot Angulo_{MPU}}{3{,}6} \tag{5.1}$$

6 Implementação

6.1 Conceito do projeto

Para facilitar a visualização do projeto como um todo foi feito um fluxograma que resume de forma simplificada o processo computacional:

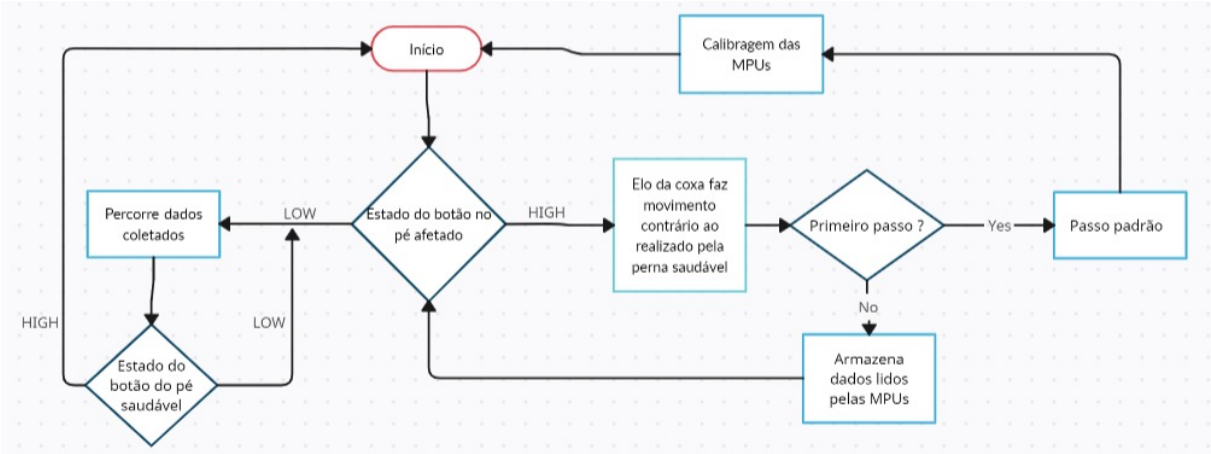

Figura 33 – Fluxograma simplificado do projeto (Fonte: Próprio autor)

E a seguir, cada uma dessas etapas serão explicadas em maior detalhe a respeito de seu funcionamento e importância.

6.2 Verificação do estado do botão no pé afetado

Ao iniciar o programa a primeira verificação é a do estado de um sensor de pressão digital, que estaria posicionado na sola do pé comprometido do usuário, mais especificamente no calcanhar, entretanto, pelo exoesqueleto ainda não possuir um suporte para os pés, será utilizado um botão na configuração de normalmente aberto para fins de simulação e será acionado manualmente. Essa etapa tem a função de indicar que a perna comprometida está sustentando o corpo do usuário, seja parado com as pernas paralelas, seja em fase de apoio da marcha. Independente da situação mencionada, o sensor indicaria estar sendo pressionado.

6.3 Elo da coxa faz movimento contrário ao realizado pela perna saudável

Quando o sensor na sola do pé comprometido emite um sinal *HIGH*, indicando que está pressionado, o próximo passo dado pelo usuário será realizado com a sua perna saudável.

Para que o corpo do usuário se projete para frente e permitir um deslocamento é necessário um movimento contrário realizado pelo motor na região proximal do fêmur:

Figura 34 – Movimento da perna contrário ao movimento para frente (Fonte: Próprio autor)

O deslocamento angular para trás realizado da perna representado na figura 34 é muito próximo ao do ângulo percorrido da coxa movida para frente, entretanto, o deslocamento para frente é realizado mais rapidamente do que o realizado para trás:

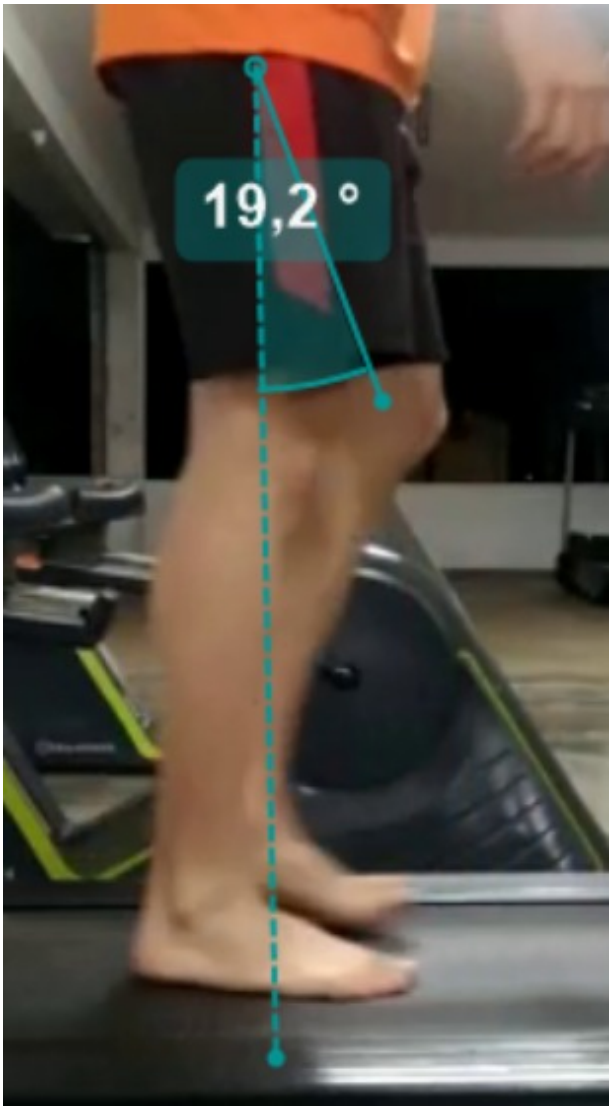

Figura 35 – Posição final da coxa ao se deslocar para frente (Fonte: Próprio autor)

Note que a coxa que se move para frente já atingiu sua posição final antes mesmo do início do movimento para trás da perna direita como pode ser visto na figura 35. Sendo o restante do movimento realizado quase exclusivamente ao redor do eixo do joelho como ilustrado na figura 36.

Figura 36 – Posição final da perna esquerda ao se deslocar para frente (Fonte: Próprio autor)

A estratégia utilizada para para simular tal fenômeno é fazer com que a perna que se desloca para trás realize o mesmo movimento angular, mas com metade da velocidade até que o sensor no calcanhar saudável identifique um contato com o solo. Caso ambas as pernas se movessem a uma mesma velocidade, o usuário não seria capaz de elevar seu pé do solo.

6.4 Identificação do primeiro passo

O primeiro passo do usuário deve ser tratado de maneira diferente, uma vez que em um cenário de reabilitação o usuário iniciaria o processo inerte e com as pernas paralelas.

O primeiro passo é realizado com a perna saudável, e uma vez que o movimento é realizado para frente, os sensores não possuem as informações de um ciclo de marcha completo, não sendo assim proveitoso reproduzi-lo. Então na situação de um primeiro passo desde a iniciação do programa, será efetuado um passo padrão e os dados lidos pelas MPUs

não serão utilizados.

Caso seja identificado que não se trata do primeiro passo do usuário, os dados lidos pelas MPUs serão armazenados.

6.5 Passo padrão

É necessário que o primeiro passo a ser dado pelo exoesqueleto seja um passo padrão, isso se deve ao fato de que o primeiro passo da perna saudável não realiza um movimento para trás, o que resultaria em dados incompletos caso os mesmos fossem coletados pelos sensores inerciais e a replicação desses movimentos resultaria em um movimento que não caracterizaria uma marcha natural.

É difícil definir qual seria o passo padrão de uma pessoa, entretanto, como pôde ser visto nas figuras 19 e 20, independente do comprimento da perna do usuário, é possível simular um passo que seja confortável para o mesmo apenas baseando-se em deslocamentos angulares e ajustando a velocidade para que seja realizado com segurança e naturalidade para o usuário.

O passo padrão é armazenado em uma matriz e pode ser ajustado caso necessário.

6.6 Armazenamento dos dados lidos pelas MPUs

O armazenamento dos dados é realizado a cada vez que se percebe uma variação angular de 5°ou mais do sensor da panturrilha.

Foi escolhido o sensor da panturrilha uma vez que é possível realizar movimentos com a coxa não detectáveis pelo sensor da panturrilha, como por exemplo uma elevação do joelho mantendo a panturrilha perpendicular ao solo. já a escolha de utilizar a variação angular ao invés de coletar os dados em intervalos de tempo se dá ao fato de que em um âmbito de reabilitação, não se consegue prever o tempo que se levará para a execução do passo, podendo ocasionar em uma quantidade de dados superior à suportada por um Arduino Mega 2560.

Os dados serão armazenados em uma matriz de tamanho definido por 150 linhas e 4 colunas um vetor de tamanho 150 para armazenar a variação no tempo em milissegundos entre cada armazenagem, onde em cada linha da matriz o primeiro e o segundo elemento deverão ser o número de detecções a serem realizadas pelos sensores de Efeito Hall para que seja alcançada a variação angular desejada pela coxa e pela panturrilha respectivamente de acordo com a equação 5.1.

Já o terceiro e quarto elemento da linha serão valores entre 26 e 229, representando a intensidade do sinal PWM médio a ser enviado para a coxa e a panturrilha respectivamente

para que seja reproduzido em um mesmo período de tempo o deslocamento angular realizado pela perna saudável.

$$RPM_{haste} = \frac{\Delta\theta \cdot 500}{\Delta t \cdot 3} \tag{6.1}$$

$$PWM = \frac{RPM \cdot 229}{9} \tag{6.2}$$

A armazenagem de dados ocorre até o momento em que o sensor no calcanhar afetado identifica um afastamento do solo, indicando o fim da etapa de balanço da perna saudável e também indicando que se pode começar o movimento da perna afetada.

6.7 Gerando a trajetória a partir dos dados coletados

Uma vez que se tem a matriz coletada durante a fase de balanço da perna saudável, lê-se linha por linha da matriz contendo dados e envia os dados aos motores, não permitindo o avanço para a próxima linha até que o angulo desejado não seja atingido tanto pela coxa quanto pela panturrilha.

7 Análise dos resultados

Os processos foram implementados com sucesso e puderam ser testados individualmente, obtendo resultados promissores para o avanço do projeto. Os dados obtidos a partir da leitura dos sensores inerciais foram armazenados com sucesso na memória do Arduino e com uma boa margem de segurança, uma vez que um ciclo de marcha, com os parâmetros escolhidos, gera em média 15 a 20 conjunto de dados, e o espaço reservado para esse armazenamento suportaria até 149 conjuntos de dados, visto que a última linha da matriz seria reservada para indicar o término da leitura dos dados.

Uma vez obtido o conjunto de dados, os mesmos podem ser enviados linha a linha para ambos os *drivers*, responsáveis por realizar o movimento dos motores. Entretanto, devido ao fato de as leituras dos sensores de Efeito Hall serem realizadas por pinos de interrupção e esse processo ser realizado em altíssimas frequências, o Arduino não era capaz de realizar o controle de ambos os motores simultaneamente, resultando em comportamentos imprevisíveis como a não parada de um dos motores uma vez que a posição desejada era atingida. Entretanto, ao não realizar movimentos com um dos motores por vez, o sistema comportava-se da maneira desejada com o motor ativo.

Uma possível solução para a limitação apontada do Arduino Mega 2560 seria conexão serial de 3 Arduinos, podendo os dois novos adicionais serem Arduinos UNO. Dessa forma será possível separar o trabalho computacional e não sobrecarregar nenhum dos microcontroladores. A sugestão do autor seria manter o Arduino Mega 2560 responsável pela leitura dos sensores inerciais MPU6050 e armazenar a matriz com as informações necessárias para o movimento dos motores, e enviar as informações necessárias para o movimento do elo da coxa para um Arduino UNO e as informações para o movimento da panturrilha para o segundo Arduino UNO. Por sua vez, os Arduinos UNO irão enviar aos seus respectivos *drivers* os sinais digitais e realizarão o monitoramento dos sensores de Efeito Hall para poder indicar para o Arduino Mega o término do movimento e receber uma nova linha com novas informações uma vez que ambos tenham terminado suas respectivas trajetórias. É necessário a utilização de dois Arduinos Uno devido ao número de pinos de interrupção, que são limitados a dois por microcontrolador, tornando-o capaz de realizar o controle de apenas um motor. A solução sugerida segue o modelo representado na figura 37.

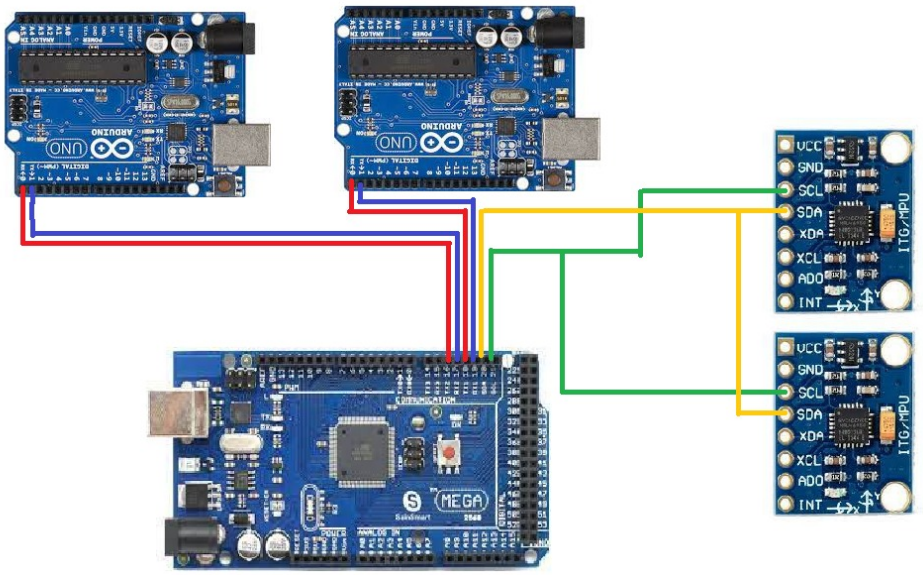

Figura 37 – Implementação sugerida (Fonte: Próprio autor)

8 Conclusão

Foram exploradas novas opções para o projeto do exoesqueleto, como a sua aplicação em pacientes acometidos por AVC, além da utilização de sensores de Efeito Hall para a realização do controle da posição dos motores ao invés dos *encoders* disponíveis, pois os sensores, apesar de obterem uma precisão menor, o movimento dos elos não apresenta valores significativos de imprecisão devido à existência do redutor com redução de 100:1. Ou seja, um erro de 7,5 graus, que seria equivalente a uma leitura a mais ou a menos do sensor de Efeito Hall, resultaria em um erro de 0,075 graus no eixo do motor, sendo assim uma discrepância não significativa.

Foram encontradas limitações no microcontrolador principalmente devido ao grande uso dos pinos de interrupção, onde por serem acionados em uma alta frequência, impedia que o contador de programa percorresse toda a extensão das instruções, fazendo com que quanto mais rápido o motor girava, e por consequência acionava os sensores, mais imprevisível era o funcionamento dos dispositivos. Uma solução para contornar a utilização dos pinos de interrupção para fazer a contagem das frações de revolução, seria a implementação de um hardware externo, um contador responsável por essa operação, onde, em uma frequência menor o arduino consultaria a posição dos motores, o que poderia ocasionar em pequenas imprecisões nos movimentos mas que seriam rapidamente corrigidas aplicando rotações no sentido contrário. Apesar das limitações encontradas no *hardware* utilizado, foi possível implementar e testar os processos individualmente, atingindo resultados satisfatórios e promissores para o avanço do projeto do exoesqueleto. Mas ainda assim ainda é necessário que novos estudos sejam realizados antes do teste do equipamento em voluntários.

9 Trabalhos futuros recomendados

9.0.1 Suporte ativo para o pé

Para trabalhos futuros relacionados ao exoesqueleto, sugere-se que seja implementado um suporte ativo para o pé do usuário, pois uma vez que se iniciam os estudos a respeito da marcha humana, chega-se rapidamente à conclusão de que os pés são de extrema importância para o movimento, e normalmente pacientes acometidos por AVC e têm sua perna afetada, também não possuem controle do pé, que durante um ciclo de marcha da estabilidade ao corpo e realiza o deslocamento do corpo para frente com auxílio do tornozelo.

9.0.2 Alteração do microcontrolador

Foram encontradas limitações no microcontrolador utilizado, e em caso de utilização de Arduinos, é necessário que seja realizada a adaptação para o funcionamento de 3 Arduinos como apontado durante a análise dos resultados 7.

Entretanto podem existir opções melhores disponíveis nos laboratórios da UnB, como placas *Field Programmable Gate Array* (FPGA) utilizada por (RODRIGUES, 2017) e a placa *Pynk*, que apresentam capacidade de ler os *encoders* dos motores e controlar ambos de forma simultânea.

Referências

AGUIAR, H. V. S. S. de. **Análise cinemática da coordenação entre membros superiores e inferiores durante a marcha de indivíduos acometidos por acidente vascular cerebral**. 2017. F. 12. Monografia (Trabalho de conclusão de curso de Fisioterapia) – FCE, Universidade de Brasília, Brasília. Citado na p. 13.

BALDASSO, D. **Análise dinâmica de um exoesqueleto de membros inferiores utilizado no contexto de reabilitação de indivíduos com lesão medular**. 2018. Monografia (Trabalho de conclusão de curso de Engenharia Mecânica) – FT, Universidade de Brasília, Brasília. Citado na p. 23.

CARMO, J. **Obstáculos da inclusão: PcD no mercado de trabalho**. 2023. Disponível em: <https://www.catho.com.br/carreira-sucesso/colunistas/noticias/obstaculos-da-inclusao-pcd-no-mercado-de-trabalho/>. Acesso em: 10 fev. 2023. Citado na p. 14.

CHARMANT, J. **Kinovea**. 2022. Disponível em: <https://www.kinovea.org/download.html>. Acesso em: 23 fev. 2023. Citado na p. 31.

CHAVES, L. R. **Robôs que ajudam a andar**. 2021. Disponível em: <https://revistapesquisa.fapesp.br/robos-que-ajudam-a-andar/>. Acesso em: 10 fev. 2023. Citado na p. 22.

CYBERDYNE. **Exoesqueleto para reabilitação da marcha HAL**. 2012. Disponível em: <https://www.medicalexpo.com/pt/prod/cyberdyne/product-92403-645387.html>. Acesso em: 10 fev. 2023. Citado na p. 20.

D'ANGELO, T. **Sensores Inerciais E Derivação De Atitude**. 2015. Disponível em: <http://www2.decom.ufop.br/imobilis/sensores-inerciais-e-derivacao-de-atitude-parte-1/>. Acesso em: 23 fev. 2023. Citado na p. 26.

ELECTRIC, G. **Do You Even Lift, Bro? Hardiman Was GE's Muscular Take On The Human-Machine Interface**. 2016. Disponível em: <https://www.ge.com/news/reports/do-you-even-lift-bro-hardiman-and-the-human-machine-interface>. Acesso em: 10 fev. 2023. Citado na p. 16.

FETICK, R. J. **MPU6050**$_{l}$*ightlibrarydocumentation*. 2021. Disponível em: <https://github.com/rfetick/MPU6050_light/blob/master/documentation_MPU6050_light.pdf>. Acesso em: 23 fev. 2023. Citado na p. 39.

FINEP. **Projeto Andar de Novo**. 2014. Disponível em: <http://www.finep.gov.br/a-finep-externo/aqui-tem-finep/projeto-andar-de-novo>. Acesso em: 10 fev. 2023. Citado nas pp. 20, 21.

FREIRE, J. P. C. D. **Projeto mecânico de um exoesqueleto com atuação no quadril**. 2019. Monografia (Trabalho de Conclusão de Curso de Engenharia Mecânica) – FT, Universidade de Brasília, Brasília. Citado nas pp. 18, 22, 23.

GRUNDMANN, J. G. **A COMPUTER CONTROLLED MULTI-TASK POWERED EXOSKELETON FOR PARAPLEGIC PATIENTS**. 1971. Disponível em: <https://cybernetiszoo.com/bionics/1971-computer-controlled-multi-task-powered-exoskeleton-paraplegic-patients-jack-george-grundmann-american/>. Acesso em: 10 fev. 2023. Citado na p. 18.

GUDINO, M. **Arduino Uno vs. Mega vs. Micro**. 2021. Disponível em: <https://www.arrow.com/en/research-and-events/articles/arduino-uno-vs-mega-vs-micro>. Acesso em: 23 fev. 2023. Citado na p. 29.

HARMON, D. **Using Hall-effect position sensors for rotary encoding**. 2019. Disponível em: <https://training.ti.com/ti-precision-labs-magnetic-sensors-using-hall-effect-position-sensors-rotary-encoding-applications>. Acesso em: 10 fev. 2023. Citado na p. 42.

KATHPALIA, N. **3 Axis Gyro Accelerometer Artificial Intelligence based Enhancement of GPS Accuracy**. 2022. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2665917422002525>. Acesso em: 10 fev. 2023. Citado na p. 27.

LOPES, C. R. H. **Dispositivos Inerciais: Projeto e desenvolvimento aplicado à solução de problemas de engenharia**. 2022. Disponível em: <https://www.nucleodoconhecimento.com.br/engenharia-mecanica/projeto-e-desenvolvimento>. Acesso em: 23 fev. 2023. Citado na p. 27.

MARTINS, D. S. **O QUE É O AVC? ACIDENTE VASCULAR CEREBRAL**. 2020. Disponível em: <https://www.acaoavc.org.br/pacientes-e-familiares/o-avc/o-que-e-o-avc/o-que-e-o-avc-acidente-vascular-cerebral>. Acesso em: 13 out. 2022. Citado na p. 13.

MOTORS, M. EC 90 flat. **May 2017 edition**, p. 1, 2017. Citado nas pp. 25, 28, 34.

MOTORS, M. Planetary Gearhead GP 52 C. **March 2021 edition**, p. 1, 2021. Citado na p. 25.

MOURA, J. **Cinesiologia e Biomecânica do ciclo da marcha**. 2018. Disponível em: <https://www.treinoemfoco.com.br/qualificando-seu-treino/cinesiologia-biomecanica-da-corrida/>. Acesso em: 10 fev. 2023. Citado nas pp. 30, 31.

PAULINE ALMEIDA, L. J. e. **Brasil tem mais de 17 milhões de pessoas com deficiência, segundo IBGE**. 2021. Disponível em: <https://www.cnnbrasil.com.br/nacional/brasil-tem-mais-de-17-milhoes-de-pessoas-com-deficiencia-segundo-ibge/>. Acesso em: 13 out. 2022. Citado na p. 13.

RODRIGUES, A. P. C. **Desenvolvimento de uma interface para acionamento de atuadores e leitura de encoders para um exoesqueleto de membro inferior com a plataforma SoC-FPGA Zybo**. 2017. Monografia (Trabalho de Conclusão de Curso de Engenharia Eletrônica) – FGA, Universidade de Brasília, Brasília. Citado nas pp. 22, 34, 52.

S. PAULO, F. de. **Frota adaptada a deficientes não existe em 88% das cidades, diz IBGE**. 2018. Disponível em: <https://www.mobilize.org.br/noticias/11105/frota-adaptada-a-deficientes-nao-e-cumprida-em-88-das-cidades-diz-ibge.html>. Acesso em: 10 fev. 2023. Citado na p. 14.

SOARES JÚNIOR, G. D. L. **Desenvolvimento de um exoesqueleto para movimentação/reabilitação de paraplégicos**. 2015. Monografia (Dissertação de mestrado em Engenharia Mecânica) – UFU, Universidade Federal de Uberlândia, Uberlândia. Citado na p. 17.

TRAFTON, A. **21st-century pack mule: MIT's 'exoskeleton' lightens the load**. 2007. Disponível em: <https://news.mit.edu/2007/21st-century-pack-mule-mits-exoskeleton-lightens-load>. Acesso em: 10 fev. 2023. Citado na p. 19.

ANEXO A – Códigos

A.1 Realiza o movimento dos motores conforme o movimento das MPUs

```
#include "Wire.h"
#include <MPU6050_light.h>

MPU6050 mpu(Wire);
MPU6050 mpu2(Wire);
int iterador=0;

const byte MPU_PANTURRILHA_ADDR = 0x69;
const byte MPU_COXA_ADDR = 0x68;

const int COXA_ROTATE_SENSE_ADDRESS = 7;
const int COXA_ROTATE_MOTOR_ADDRESS = 8;
const int COXA_ROTATE_PWM_ADDRESS = 9;

const int PANT_ROTATE_SENSE_ADDRESS = 10;
const int PANT_ROTATE_MOTOR_ADDRESS = 11;
const int PANT_ROTATE_PWM_ADDRESS = 12;


int z=0, z2=0;
int aCOXAState;
volatile bool aCOXALastState;
volatile bool bCOXALastState;

int aPANTState;
volatile bool aPANTLastState;
volatile bool bPANTLastState;


unsigned long timer = 0;
unsigned long timerPWM = 0;

int FracaoDeVoltaCOXA;
int FracaoDeVoltaPANT;
int Input=0;
int Input2=0;

//CONTAGEM DE VOLTAS DA COXA
```

```
39 void aCOXAChanged(){
     aCOXALastState = !aCOXALastState;
41   if (bCOXALastState != aCOXALastState)
       FracaoDeVoltaCOXA++;
43   else
       FracaoDeVoltaCOXA --;
45 }

47 void bCOXAChanged(){
     bCOXALastState = !bCOXALastState;
49   if(bCOXALastState == aCOXALastState)
       FracaoDeVoltaCOXA++;
51   else
       FracaoDeVoltaCOXA--;
53 }

55 //CONTAGEM DE VOLTAS DA PANTURRILHA
   void aPANTChanged(){
57   aPANTLastState = !aPANTLastState;
     if (bPANTLastState != aPANTLastState)
59     FracaoDeVoltaPANT++;
     else
61     FracaoDeVoltaPANT --;
   }
63
   void bPANTChanged(){
65   bPANTLastState = !bPANTLastState;
     if(bPANTLastState == aPANTLastState)
67     FracaoDeVoltaPANT++;
     else
69     FracaoDeVoltaPANT--;
   }
71
   //INICIALIZA  O E CALIBRAGEM DAS MPUs
73 void initializeMPU(MPU6050* mpuToConfigure, String name){
     byte status = mpuToConfigure->begin();
75   Serial.print(name);
     Serial.print(F(" status: "));
77   Serial.println(status);
   //  while (status != 0) { } // stop everything if could not connect to
      MPU6050
79   Serial.println(F("Calculating offsets, do not move MPU6050"));
     delay(100);
81   mpuToConfigure->calcOffsets(); // gyro and accelero
     Serial.println("Done!\n");
83 }
```

```
85 void setup() {
   Serial.begin(115200);
87 //DECLARA O DOS PINOS DE INTERRUP O
   attachInterrupt(digitalPinToInterrupt(2), aCOXAChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 2)
89 attachInterrupt(digitalPinToInterrupt(3), bCOXAChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 3)
   attachInterrupt(digitalPinToInterrupt(18), aPANTChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 18)
91 attachInterrupt(digitalPinToInterrupt(19), bPANTChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 19)
   FracaoDeVoltaCOXA = 0;
93 FracaoDeVoltaPANT = 0;
   Wire.begin();
95
   mpu.setAddress(MPU_PANTURRILHA_ADDR);
97 mpu2.setAddress(MPU_COXA_ADDR);
   initializeMPU(&mpu, "Panturrilha");
99 initializeMPU(&mpu2, "Coxa");

101 pinMode (2,INPUT);
   pinMode (3,INPUT);
103 pinMode (18,INPUT);
   pinMode (19,INPUT);
105 aCOXALastState = digitalRead(2);
   bCOXALastState = digitalRead(3);
107 aPANTLastState = digitalRead(18);
   bPANTLastState = digitalRead(19);
109
   analogWrite(COXA_ROTATE_PWM_ADDRESS, 100);
111 analogWrite(PANT_ROTATE_PWM_ADDRESS, 70);
}
113
void setRotationClockwise(int rotationSenseAddress){
115 digitalWrite(rotationSenseAddress, LOW); // 7 pra coxa
}
117
void setRotationCounterclockwise(int rotationSenseAddress){
119 digitalWrite(rotationSenseAddress, HIGH); // 7 pra coxa
}
121
void rotateMotor(int rotateMotorAddress){
123 digitalWrite(rotateMotorAddress, HIGH);
}
125
void stopMotor(int rotateMotorAddress){
127 digitalWrite(rotateMotorAddress, LOW);
```

```
}

void loop() {
  //VERIFICA ALTERA ES DO ESTADO DOS MOTORES (GIROU ?)
  aCOXAState = digitalRead(2); // Reads the "current" state of the outputA DA COXA
  aPANTState = digitalRead(18); // Reads the "current" state of the outputA DA PANTURRILHA
  if (aCOXAState != aCOXALastState){
    if (digitalRead(3) != aCOXAState)
      FracaoDeVoltaCOXA++;
    else
      FracaoDeVoltaCOXA --;
   }
  aCOXALastState = aCOXAState;

  if (aPANTState != aPANTLastState){
    if (digitalRead(19) != aPANTState)
      FracaoDeVoltaPANT++;
    else
      FracaoDeVoltaPANT --;
   }
  aPANTLastState = aPANTState;

  mpu.update();
  mpu2.update();

  //CAPTURA VALORES DAS MPUS A CADA 100ms
  if ((millis() - timer) > 100) {
     z = mpu2.getAngleZ();
     z2 = mpu.getAngleZ();
     timer = millis();
     Input = ((48*z)/3.6);
     Input2 = ((48*z2)/3.6);
     //Input2 = ((48*(z2-z))/3.6); //Para funcionar com os dois sensores na mesma perna
  }

  //MOVE OS MOTORES
  int diff = abs(Input-FracaoDeVoltaCOXA);


  if(diff<5) {
    stopMotor(COXA_ROTATE_MOTOR_ADDRESS);
  } else if (Input<FracaoDeVoltaCOXA) {

    rotateMotor(COXA_ROTATE_MOTOR_ADDRESS);
```

```
    setRotationCounterclockwise(COXA_ROTATE_SENSE_ADDRESS);
  } else if(Input>FracaoDeVoltaCOXA) {

    rotateMotor(COXA_ROTATE_MOTOR_ADDRESS);
    setRotationClockwise(COXA_ROTATE_SENSE_ADDRESS);
  }

  int diffPANT = abs(Input2-FracaoDeVoltaPANT);
  if(diffPANT<5) {
    stopMotor(PANT_ROTATE_MOTOR_ADDRESS);
  } else if (Input2<FracaoDeVoltaPANT) {
    rotateMotor(PANT_ROTATE_MOTOR_ADDRESS);

    setRotationCounterclockwise(PANT_ROTATE_SENSE_ADDRESS);
  } else if(Input2>FracaoDeVoltaPANT) {
    rotateMotor(PANT_ROTATE_MOTOR_ADDRESS);
    setRotationClockwise(PANT_ROTATE_SENSE_ADDRESS);
  }

}
```

A.2 Realiza o movimento dos motores conforme sequência de dados armazenados em uma matriz

```
#include "Wire.h"
#include <MPU6050_light.h>

//#define outputA 2
//#define outputB 3

MPU6050 mpu(Wire);
MPU6050 mpu2(Wire);
int i = 0, j = 0;
int matrizPWM[150][4];
unsigned long delta_T[150];
int passoPadrao [150][4] = {                    //Para testar trajetoria
    da coxa
                           {-174, 0, 40, 0 },
                           {174, 0, 50, 0 },
                           {174, 0, 70, 0 },
                           {93, 0, 90, 0 },
                           {67, 0, 100, 0 },
                           {93, 0, 100, 0 },
```

```
19                              {-414, 0, 50, 0 },
                                {'s', 0, 0, 0 }
21 };
   //int passoPadrao [150][4] = {                    //Para testar trajetoria
       da panturrilha
23 //                              {0, -186, 0, 40 },
   //                              {0, -93, 0, 35 },
25 //                              {0, -106, 0, 50 },
   //                              {0, -120, 0, 50 },
27 //                              {0, -106, 0, 50 },
   //                              {0, -106, 0, 50 },
29 //                              {0, 106, 0, 50 },
   //                              {0, 133, 0, 55 },
31 //                              {0, 146, 0, 55 },
   //                              {0, 133, 0, 30 },
33 //                              {0, 120, 0, 40 },
   //                              {0, 80, 0, 40 },
35 //                              {'s', 0, 0, 0 }
   //
37 //};

39 const byte MPU_PANTURRILHA_ADDR = 0x69;
   const byte MPU_COXA_ADDR = 0x68;
41
   const int botao = 44;
43 const int botaoPeAfetado = 45;

45 const int COXA_ROTATE_SENSE_ADDRESS = 7;
   const int COXA_ROTATE_MOTOR_ADDRESS = 8;
47 const int COXA_ROTATE_PWM_ADDRESS = 9;

49 const int PANT_ROTATE_SENSE_ADDRESS = 10;
   const int PANT_ROTATE_MOTOR_ADDRESS = 11;
51 const int PANT_ROTATE_PWM_ADDRESS = 12;


53
   int z = 0, z2 = 0, z2anterior = 0, zAnterior = 0;
55 volatile bool aCOXALastState;
   volatile bool bCOXALastState;
57
   volatile bool aPANTLastState;
59 volatile bool bPANTLastState;

61 unsigned long timer = 0;
   unsigned long timerPWM = 0;
63
   int FracaoDeVoltaCOXA=0;
```

```
65 int FracaoDeVoltaPANT=0;
   int Input = 0;
67 int Input2 = 0;

69 void processMatrix(int matriz[150][4]){
     int j=0;
71 //  Serial.print("Processa matriz");
     while(matriz[j][0] != 's'){
73     processLine(matriz[j]);

75     j++;
     }
77 }

79 // 0 - z coxa
   // 1 - z pant
81 // 2 - pwm coxa
   // 3 - pwm panturrilha
83 void processLine(int matriz[4]) {
   //  Serial.print("LINHAAA");
85   int zCoxaTarget = matriz[0];
     Serial.println(zCoxaTarget);
87   int zPantTarget = matriz[1];
     int pwmCoxa = matriz[2];
89   int pwmPant = matriz[3];
     setCoxaSpeed(pwmCoxa);
91   setPantSpeed(pwmPant);
     bool positionCoxaDone = true; //Para testar trajet ria de 1 por vez,
      torne o outro TRUE
93   bool positionPantDone = false;
     int oldCoxaFracaoDeVolta = FracaoDeVoltaCOXA;
95   int oldPantFracaoDeVolta = FracaoDeVoltaPANT;
     while(!(positionCoxaDone && positionPantDone)){
97     int fracoesCoxa = (FracaoDeVoltaCOXA-oldCoxaFracaoDeVolta);
       int fracoesPant = (FracaoDeVoltaPANT-oldPantFracaoDeVolta);
99     //positionCoxaDone = (abs(fracoesCoxa-zCoxaTarget) < 2); //Para testar
      trajet ria de 1 por vez, comente o outro
       positionPantDone = (abs(fracoesPant-zPantTarget) < 2);
101 //    Serial.print("Coxa: ");
   //    Serial.print(fracoesCoxa);
103 //    Serial.print("/");
   //    Serial.println(zCoxaTarget);
105


107    if(!positionCoxaDone){
         if(zCoxaTarget > 0){
109        setRotationClockwise(COXA_ROTATE_SENSE_ADDRESS);
```

```
        rotateMotor(COXA_ROTATE_MOTOR_ADDRESS);
111     Serial.print("Gira coxa horario\n");
      } else {
113     setRotationCounterclockwise(COXA_ROTATE_SENSE_ADDRESS);
        rotateMotor(COXA_ROTATE_MOTOR_ADDRESS);
115     Serial.print("Gira coxa anti\n");
      }
117   }
    else {
119 //    Serial.print("PAROU !");
      stopMotor(COXA_ROTATE_MOTOR_ADDRESS);
121   }
    if(!positionPantDone){
123   if(zPantTarget > 0){
        setRotationClockwise(PANT_ROTATE_SENSE_ADDRESS);
125     rotateMotor(PANT_ROTATE_MOTOR_ADDRESS);
      } else {
127     setRotationCounterclockwise(PANT_ROTATE_SENSE_ADDRESS);
        rotateMotor(PANT_ROTATE_MOTOR_ADDRESS);
129   }
    } else {
131   stopMotor(PANT_ROTATE_MOTOR_ADDRESS);
    }
133 }
}
135

void setCoxaSpeed(int PWM){
137 analogWrite(COXA_ROTATE_PWM_ADDRESS,PWM);
}
139

void setPantSpeed(int PWM){
141 analogWrite(PANT_ROTATE_PWM_ADDRESS,PWM);
}
143

int Calc_PWM_COXA(unsigned long timerPWM, int Zcoxa, int Zcoxaanterior) {
145 int delta_COXA;
  delta_COXA = abs(abs(Zcoxaanterior) - abs(Zcoxa));
147 if (delta_COXA < 3)
    return 0;
149 double RPM_COXA = (delta_COXA * 500) / (timerPWM * 3);
  if (RPM_COXA <= 2)
151   RPM_COXA = 2;
  else if (RPM_COXA >= 18)
153   RPM_COXA = 18;
  //   Serial.println(RPM_COXA);
155 int PWM_Set = map(RPM_COXA, 2, 18, 26, 229);
  return (PWM_Set);
```

```
157 }
    int Calc_PWM_PANT(unsigned long timerPWM, int Zpant, int Zpantanterior) {
159   int delta_PANT;
      delta_PANT = abs(abs(Zpantanterior) - abs(Zpant));
161   //Serial.println(delta_PANT);
      double RPM_PANT;
163   RPM_PANT = (delta_PANT * 500) / (timerPWM * 3);
      if (RPM_PANT <= 1)
165     RPM_PANT = 1;
      else if (RPM_PANT >= 9)
167     RPM_PANT = 9;
      int PWM_Set = map(RPM_PANT, 1, 9, 26, 229);
169   return (PWM_Set);
    }
171
    void aCOXAChanged() {
173   aCOXALastState = !aCOXALastState;
      if (bCOXALastState != aCOXALastState)
175     FracaoDeVoltaCOXA++;
      else
177     FracaoDeVoltaCOXA --;
    }
179
    //CONTROLE DA COXA
181 void bCOXAChanged() {
      bCOXALastState = !bCOXALastState;
183   if (bCOXALastState == aCOXALastState)
        FracaoDeVoltaCOXA++;
185   else
        FracaoDeVoltaCOXA--;
187 }

189 //CONTROLE DA PANTURRILHA
    void aPANTChanged() {
191   aPANTLastState = !aPANTLastState;
      if (bPANTLastState != aPANTLastState)
193     FracaoDeVoltaPANT++;
      else
195     FracaoDeVoltaPANT --;
    }
197
    void bPANTChanged() {
199   bPANTLastState = !bPANTLastState;
      if (bPANTLastState == aPANTLastState)
201     FracaoDeVoltaPANT++;
      else
203     FracaoDeVoltaPANT--;
```

```
}

void initializeMPU(MPU6050* mpuToConfigure, String name) {
  byte status = mpuToConfigure->begin();
  Serial.print(name);
  Serial.print(F(" status: "));
  Serial.println(status);
 // while (status != 0) { } // stop everything if could not connect to
    MPU6050
  Serial.println(F("Calculating offsets, do not move MPU6050"));
  delay(100);
  mpuToConfigure->calcOffsets(); // gyro and accelero
  Serial.println("Done!\n");
}

void zerarPosicaoMPU(){
  mpu.update();
  mpu2.update();

  z = mpu2.getAngleZ();
  z2 = mpu.getAngleZ();
  zAnterior = z;
  z2anterior = z2;
}

void setup() {

  Serial.begin(115200);
//  attachInterrupt(digitalPinToInterrupt(2), aCOXAChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 2) //Para testar um
    por vez, comente os 2 interrupts do outro
//  attachInterrupt(digitalPinToInterrupt(3), bCOXAChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 3)
  attachInterrupt(digitalPinToInterrupt(18), aPANTChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 18)
  attachInterrupt(digitalPinToInterrupt(19), bPANTChanged, CHANGE);//
    Initialize the intterrupt pin (Arduino digital pin 19)
  FracaoDeVoltaCOXA = 0;
  FracaoDeVoltaPANT = 0;
  Wire.begin();

  mpu.setAddress(MPU_PANTURRILHA_ADDR);
  mpu2.setAddress(MPU_COXA_ADDR);
  initializeMPU(&mpu, "Panturrilha");
  initializeMPU(&mpu2, "Coxa");

  pinMode (botao, INPUT);
```

```
245   pinMode (2, INPUT);
      pinMode (3, INPUT);
247   pinMode (18, INPUT);
      pinMode (19, INPUT);
249   aCOXALastState = digitalRead(2);
      bCOXALastState = digitalRead(3);
251   aPANTLastState = digitalRead(18);
      bPANTLastState = digitalRead(19);
253
//    analogWrite(COXA_ROTATE_PWM_ADDRESS, 50);
255 //  analogWrite(PANT_ROTATE_PWM_ADDRESS, 50);
      timer = millis();
257   zerarPosicaoMPU();
}
259
void setRotationClockwise(int rotationSenseAddress) {
261   digitalWrite(rotationSenseAddress, LOW); // 7 pra coxa
}
263
void setRotationCounterclockwise(int rotationSenseAddress) {
265   digitalWrite(rotationSenseAddress, HIGH); // 7 pra coxa
}
267
void rotateMotor(int rotateMotorAddress) {
269   digitalWrite(rotateMotorAddress, HIGH);
}
271
void stopMotor(int rotateMotorAddress) {
273   digitalWrite(rotateMotorAddress, LOW);
}
275
void recordStep(){
277   mpu.update();
      mpu2.update();
279
      z = mpu2.getAngleZ();
281   z2 = mpu.getAngleZ();

283   if (abs(abs(z2) - abs(z2anterior)) >= 5) {
        timerPWM = millis() - timer;
285     timer = millis();
        delta_T[j] = timerPWM;
287     z2=z2-z;
        if (abs(abs(z) - abs(zAnterior)) < 3) {
289       z = zAnterior;
        }
291     matrizPWM[j][i] = map((z-zAnterior), -360, 360, -4800, 4800);
```

```
    matrizPWM[j + 1][i] = 's';
293     matrizPWM[j][i + 1] = map((z2-z2anterior),-360,360,-4800,4800);
    if (j != 0) {
295       matrizPWM[j][i + 2] = Calc_PWM_COXA(delta_T[j], z, matrizPWM[j - 1][i
    ]);
        Serial.print("Coxa: ");
297       Serial.println(matrizPWM[j][i + 2]);
        matrizPWM[j][i + 3] = Calc_PWM_PANT(delta_T[j], z2, matrizPWM[j - 1][i
    + 1]);
299       Serial.print("Panturrilha: ");
        Serial.println(matrizPWM[j][i + 3]);
301       Serial.print("Botao: ");
        Serial.println(digitalRead(botao));
303     }
    else {
305       delta_T[j] = 0;
        matrizPWM[j][i + 2] = 0;
307       matrizPWM[j][i + 3] = 0;
    }
309     zAnterior = z;
    z2anterior = z2;
311     j++;
  }
313 }

315 bool healthyFootIsOnGround(){
  return digitalRead(botao)==HIGH ;
317 }

319 bool shouldStartWalking(){
  return true;
321 }

323 bool primeiroPasso = true;

325 void loop() {
  if(primeiroPasso){
327 //    Serial.print("Primeiro passo");
//    while(healthyFootIsOnGround())
329 //      delay(10);
//    while(!healthyFootIsOnGround())
331 //      delay(10);
//    delay(100);
333     processMatrix(passoPadrao);
    primeiroPasso = false;
335     zerarPosicaoMPU();
  }
```

```
337 //   while(healthyFootIsOnGround())
    //     delay(10);
339 //   j=0;
    //   while(!healthyFootIsOnGround()){
341 //     recordStep();
    //   }
343 //   delay(100);
    //   processMatrix(matrizPWM);
345 }
```