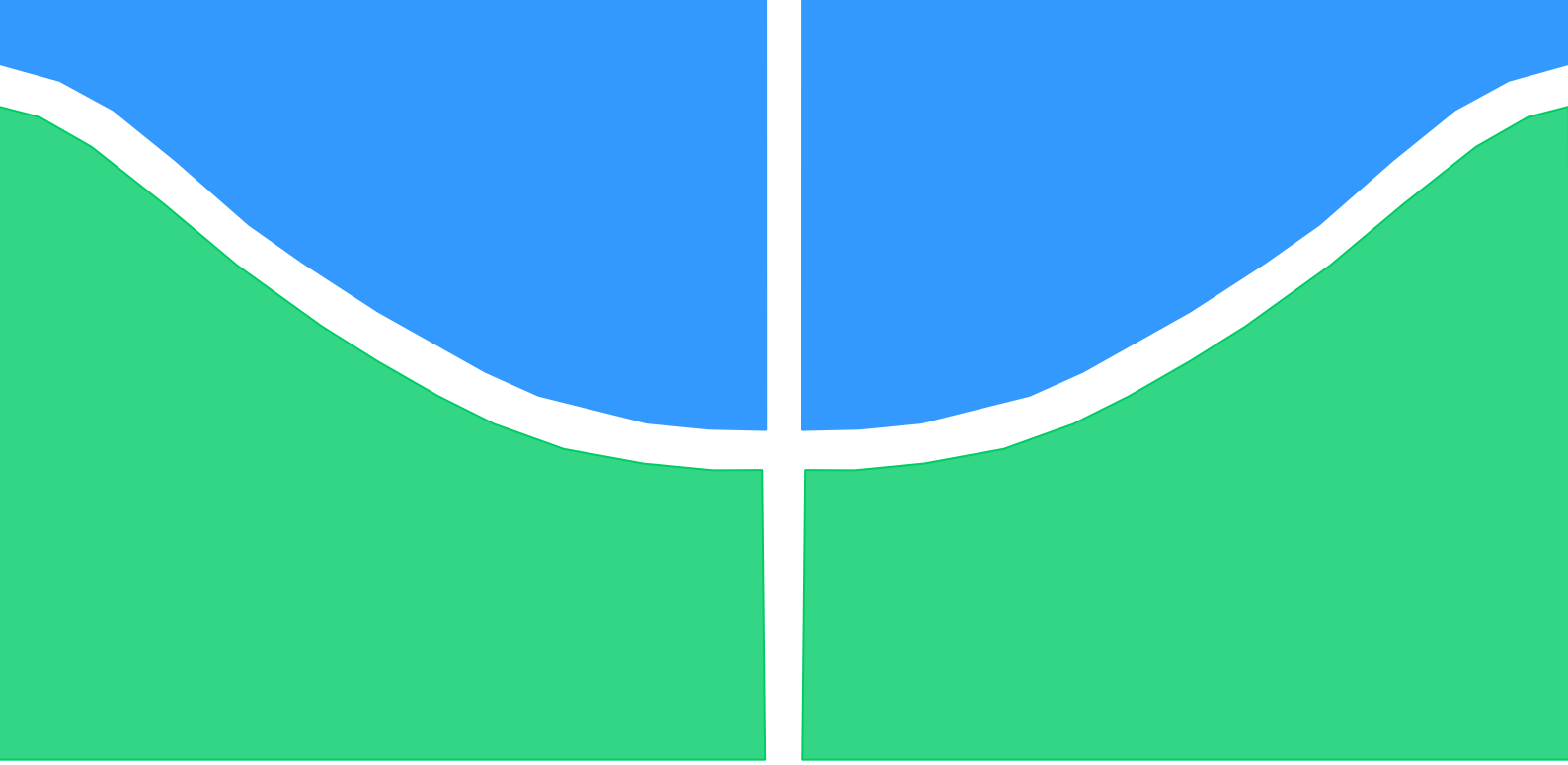


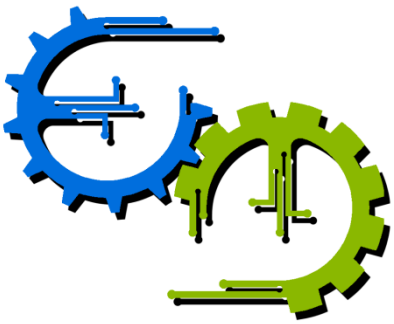
TRABALHO DE GRADUAÇÃO

ESTUDO E DESENVOLVIMENTO DE UMA INTERFACE DE COMANDO DE CADEIRA DE RODAS MOTORIZADA PARA PESSOAS TETRAPLÉGICAS

Por,

Francisco Matheus Pereira de Oliveira

Brasília, Julho de 2019



ENGENHARIA MECATRÔNICA

UNIVERSIDADE DE BRASÍLIA

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

TRABALHO DE GRADUAÇÃO

ESTUDO E DESENVOLVIMENTO DE UMA INTERFACE DE COMANDO DE CADEIRA DE RODAS MOTORIZADA PARA PESSOAS TETRAPLÉGICAS

POR,

Francisco Matheus Pereira de Oliveira

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

Banca Examinadora

Prof. Walter de Britto, UnB/ ENM (Orientador) _____

Prof. Carla Cavalcante Koike, UnB/ CIC _____

Prof. Carlos Humberto Llanos, UnB/ ENM _____

Brasília, Julho de 2019

FICHA CATALOGRÁFICA

FRANCISCO MATHEUS PEREIRA DE OLIVEIRA

Estudo e Desenvolvimento de uma interface de comando de cadeira de rodas motorizada para pessoas tetraplégicas,

[Distrito Federal] 2019.

91p., 297 mm (FT/UnB, Engenheiro, Controle e Automação. Trabalho de Graduação– Universidade de Brasília. Faculdade de Tecnologia.

1.Cadeira de Rodas 2.Tetraplegia
3.Interface de Controle 4.Baixo Custo

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

OLIVEIRA, F. M. P., (2018). Estudo e Desenvolvimento de uma interface de comando de cadeira de rodas motorizada para pessoas tetraplégicas. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 01, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 91p.

CESSÃO DE DIREITOS

AUTOR: Francisco Matheus Pereira de Oliveira

ESTUDO E DESENVOLVIMENTO DE UMA INTERFACE DE COMANDO DE CADEIRA DE RODAS MOTORIZADA PARA PESSOAS TETRAPLÉGICAS: Estudo, projeto e implementação de um sistema de comando de cadeira de rodas para pessoas com tetraplegia.

GRAU: Engenheiro ANO: 2019

Francisco Matheus Pereira de Oliveira
Sof Sul Quadra 13 Conj. B Lote 1/3 ap 104 – Guará.
71215-267 Brasília – DF – Brasil.

AGRADECIMENTOS

Primeiramente agradeço à Deus por me acompanhar e me guiar durante essa jornada na Universidade, sem o suporte dele jamais teria conseguido chegar até aqui. Agradeço ao professor e meu orientador Walter de Britto por me confiar esse projeto, acreditar no meu trabalho e me orientar. Agradeço também a minha família que me apoiou e me ajudou sempre que precisei, em especial meus pais, pois me ensinaram a ter persistência e dedicação para alcançar meus objetivos. Fica aqui também um agradecimento a todos que de alguma forma ajudaram a tornar esse trabalho possível.

Francisco Matheus Pereira de Oliveira

RESUMO

Neste trabalho foi feito um estudo acerca de soluções tecnológicas criadas com o intuito de permitir que pessoas com tetraplegia pudessem controlar uma cadeira de rodas motorizada. Dentre as soluções encontradas, foi feita uma análise dos aspectos mais desejados de cada implementação, bem como suas inconveniências, a fim de conduzir a melhor escolha de implementação a ser feita. Com isso, foi desenvolvido e testado um sistema de locomoção semi automático, que permite que ao usuário, dentro de um ambiente previamente conhecido e mapeado, selecionar um cômodo desejado em uma interface visual e ser conduzido até lá.

Palavras Chave: tetraplegia, mobilidade, cadeira de rodas, interface.

ABSTRACT

This dissertation is the result of a study of technological solutions designed to allow people with quadriplegia to control a motorized wheelchair. In order to achieve the best choice of implementation, the most desired aspects of each implementation among the studied solutions were analyzed, as well as their inconveniences. Finally, a system of semi-automatic locomotion was developed and tested, which allows the user, within a previously known and mapped environment, to select a desired room in a visual interface and be led there.

Keywords: quadriplegic, mobility, wheelchair, interface.

SUMÁRIO

REFERÊNCIA BIBLIOGRÁFICA iii
CESSÃO DE DIREITOS iii
CAPÍTULO 1 – INTRODUÇÃO 1
1.1 CONTEXTO E PROBLEMATIZAÇÃO 1
1.2 OBJETIVO GERAL 1
1.2.1 OBJETIVO SOCIAL 2
1.2.2 OBJETIVO TÉCNICO 2
1.3 A DEFICIÊNCIA 2
1.4 METODOLOGIA 3
1.4.1 ESTUDOS REALIZADOS 3
1.4.2 PLANEJAMENTO E EXECUÇÃO DO PROJETO 3
1.4.3 TESTES DE FUNCIONALIDADE 4
CAPÍTULO 2 – REVISÃO BIBLIOGRÁFICA 5
2.1 JOYSTICK 5
2.2 SOPRO E SUCÇÃO 7
2.3 MOVIMENTAÇÃO DA CABEÇA 10
2.4 MOVIMENTAÇÃO DOS OLHOS 12
2.5 MOVIMENTAÇÃO DA FACE 13
2.6 LÍNGUA 14
2.7 VOZ 15
2.8 TECNOLOGIAS ASSISTIVAS AUXILIARES 16
CAPÍTULO 3 – PROJETO 17
3.1 PREMISSAS E REQUISITOS 17
3.2 ESTRUTURA 19
3.3 POTÊNCIA 20
3.4 CONTROLE 22
3.5 INTERFACE DE COMANDO 23
3.5.1 VISÃO GERAL DO SISTEMA SEMI AUTOMÁTICO 24
3.5.2 MUDANÇAS NA ESTRUTURA 25
3.5.3 MUDANÇAS DE HARDWARE 26
3.5.4 SOFTWARE 29
CAPÍTULO 4 – IMPLEMENTAÇÃO 31
4.1 POTÊNCIA 31
4.2 INTERFACE 32
4.3 ESTRUTURA 33

4.4 HARDWARE 34
4.5 SOFTWARE 36
CAPÍTULO 5 – TESTES E RESULTADOS 38
5.1 POTÊNCIA 38
5.2 CONTROLE 39
5.3 SISTEMA SEMI AUTOMÁTICO 47
CAPÍTULO 6 – CONCLUSÕES. 50
REFERÊNCIAS BIBLIOGRÁFICAS. 52
APÊNDICE 1 55
APÊNDICE 2 59
APÊNDICE 3 79

LISTA DE FIGURAS

2.1.1 Joystick de queixo implementado em [8] 5
2.1.2 Mini joystick [10] 6
2.1.3 Joystick de pé [11] 6
2.1.4 Joystick para dedos dos pés [11] 7
2.1.5 Touchpad [12] 7
2.2.1 Sensor de pressão confeccionado em [8] 8
2.2.2 Princípio de funcionamento do sensor de pressão [8] 8
2.2.3 Transdutor de ar [13] 9
2.2.4 Painel de LEDs [13] 9
2.2.5 Indicação de função no display: seguir em frente, girar à direita, ré e girar à esquerda.[14] 9
2.2.6 Interface de controle comercializada pela empresa ASL [15] 10
2.3.1 Boné com acelerômetro fixado [16] 11
2.3.2 Interface de comando fixada no encosto de cabeça [17] 11
2.4.1 Máscara de aquisição de sinais nervosos [18] 12
2.4.1 Pupil Pro [19] 13
2.5.1 Posição dos eletrodos de controle [20] 14
2.6.1 Dispositivo de leitura, piercing magnético e interface mobile.[21] 15
2.6.2 Autor do estudo e usuário em testes do sistema.[21] 15
2.7.1 a)Tutorial da interface. b)Tela Principal. c)Gráficos de monitoramento [22] 16
2.8.1 Mouse Giroscópio [23] 16
3.2.1 Cadeira de rodas utilizada [8] 19
3.2.2 Representação dos suportes para motores 19
3.3.1 Motor MR-210 VER 240 [8] 20
3.3.2 Circuito ponte H 20
3.3.3 Transistor MOSFET IRF540N com dissipador térmico 21
3.3.4 Módulo Driver Ponte H [24] 21
3.3.5 Circuito de isolamento utilizando MOC3022 22
3.3.6 Bateria Automotiva 12 Volts 45Ah 22
3.4.1 Portas de saída Arduino Leonardo [25] 23
3.5.1 Joystick de 3 eixos 24
3.5.2 Controle por botões 24
3.5.1.1 Modos de funcionamento do sistema semi automático 25
3.5.2.1 Representação do suporte para o joystick de queixo 25

3.5.3.1 Representação da polia com imãs....26
3.5.3.2 Montagem polia-roda....27
3.5.3.3 Posicionamento do sensor Hall....27
3.5.3.4 Representação da cinemática do sistema.[adaptado de 26]....28
3.5.3.5 Cinemática de posicionamento da cadeira.[adaptado de 26]....28
3.5.3.6 Sensor de Ultrassom....29
3.5.4.1 Ambiente fictício de locomoção automática....29
3.5.4.2 Tela exibida na interface....30
4.1.1 Módulo driver ponte H implementado....31
4.1.2 Circuito de proteção com opto isoladores....32
4.2.1 Controle por botões fabricado....32
4.2.2 Cadeira com joystick e botões implementados....33
4.3.1 Cadeira com joystick de queixo....33
4.3.2 Mesa suporte para notebook instalada....34
4.4.1 Polia do motor com imãs....34
4.4.2 Sensor de efeito hall instalado....35
4.4.3 Sensores ultrassônicos instalados....35
4.5.1 Fluxograma principal do código....36
4.5.2 Fluxograma da função “vaiParaComodo”....37
4.5.3 Fluxograma da função de percorrer trajetória....37
5.1.1 Esquema de montagem dos medidores para teste....38
5.2.1 Representação do teste de aproximação....39
5.2.2 Gráfico de desempenho do teste de aproximação....40
5.2.3 Representação do teste de estacionar....41
5.2.4 Gráfico de desempenho do teste de estacionar....42
5.2.5 Representação do teste de slalom....43
5.2.6 Gráfico de desempenho do teste de slalom....44
5.2.7 Gráfico de desempenho do teste de aproximação com 3 formas de comando....45
5.2.8 Gráfico de desempenho do teste de estacionar com 3 formas de comando....46
5.2.9 Gráfico de desempenho do teste de slalom com 3 formas de comando....46
5.3.1 Sistema completo em testes com voluntária....47
5.3.2 Ambiente fictício cotado....48
5.3.3 Ambiente reproduzido para testes do sistema semi automático....48
6.1 Representação de tarjas magnéticas nos portais das portas....51

LISTA DE TABELAS

3.1.1 Comparativo entre as interfaces encontradas 18
3.3.1 Especificações do motor MR-210 VER 240 20
3.3.2 Especificações Módulo Driver Ponte H – BTS7960 [24] 21
5.1.1 Teste de acionamento dos motores 38
5.2.1 Desempenho do teste de aproximação 40
5.2.2 Desempenho do teste de estacionar 41
5.2.3 Desempenho do teste de slalom 43
5.2.4 Desempenho do teste de aproximação com 3 formas de comando 44
5.2.5 Desempenho do teste de estacionar com 3 formas de comando 45
5.2.6 Desempenho do teste de slalom com 3 formas de comando 46

LISTA DE SÍMBOLOS

Símbolos Latinos

A	Ampere	[C/s]
V	Volts	[W/A]
Hz	Hertz	[1/s]

Siglas

ABNT	Associação Brasileira de Normas Técnicas
UNB	Universidade de Brasília
OMS	Organização Mundial da Saúde
ONU	Organização das Nações Unidas
PWM	Pulse Width Modulation (Modulação de Largura de Pulso)

CAPÍTULO 1 – INTRODUÇÃO

Este primeiro capítulo tem como objetivo mostrar o panorama geral dos fatos motivadores deste trabalho, bem como uma noção do problema em si. Além disso, é mostrado a forma concepção do trabalho, tanto escrito quanto prático.

1.1 CONTEXTO E PROBLEMATIZAÇÃO

Em 2011 a OMS (Organização Mundial da Saúde) fez um levantamento de dados no mundo todo com o objetivo de estimar o número de pessoas que possuíam, ou não, algum tipo de deficiência, o resultado foi que cerca de 1 bilhão de pessoas vivem com alguma deficiência [1]. Esse número se torna preocupante quando identificamos que 80% dessas pessoas vivem em países em desenvolvimento, cujas limitações tecnológicas e financeiras inviabilizam a garantia de seus direitos a uma vida digna e 'normal', pois, segundo a ONU, possuir algum tipo de deficiência aumenta o custo de vida em cerca de um terço da renda. No Brasil, segundo o IBGE, o número de pessoas com algum tipo de deficiência chega a 45 milhões de pessoas [2].

Direito a dignidade, a tratamento médico, psicológico e funcional, bem como a desenvolver suas capacidades e habilidades ao máximo são alguns dos direitos assegurados às pessoas com deficiência e estão descritos na Declaração sobre os Direitos das Pessoas com Deficiência, proclamada pela Assembleia Geral da ONU em 9 de dezembro de 1975. Para conseguir que essas pessoas garantam seus direitos anteriormente descritos, muito tem se investido em pesquisa e desenvolvimento de tecnologias assistivas, dentro das universidades, criando soluções que deem a essas pessoas uma independência e qualidade de vida melhores e com esse mesmo objetivo esse trabalho foi feito. No Brasil, o investimento em tecnologias assistivas foi expressivo nos últimos anos, até o ano de 2015 foram feitos 3 editais que juntos somaram 65 milhões de reais, distribuídos entre diversas instituições de pesquisa e ensino superior do país [2].

1.2 OBJETIVO GERAL

O objetivo deste trabalho é compreender melhor aspectos relacionados a mobilidade de pessoas com limitações motoras de membros tanto superiores quanto inferiores, ou seja, pessoas com tetraplegia. Para isso foi feito um estudo a fim de encontrar quais soluções existem e suas principais vantagens e desvantagens e assim propor uma contribuição significativa.

1.2.1 OBJETIVO SOCIAL

Como foi dito anteriormente, a grande maioria das pessoas que possuem algum tipo de deficiência vem de países em desenvolvimento, além disso, mais de 50% delas não conseguem pagar por serviços de saúde, segundo a ONU [3]. Outro fato importante a se considerar é o de que pessoas com deficiência sofrem uma maior resistência para entrarem no mercado de trabalho, como foi investigado nos Estados Unidos em 2004, quando apenas 35% das pessoas portadoras de deficiência ativas economicamente exerciam alguma atividade, enquanto esse percentual era de 78% para pessoas sem deficiência. Um estudo feito um ano antes pela Universidade de Rutgers [4] revelou que um dos motivos para isso era que os empregadores não acreditavam que pessoas com deficiência efetivamente conseguiriam exercer suas funções de trabalho e outro motivo foi o medo dos custos necessários para as instalações especiais.

Tendo isso em vista, neste trabalho foi feito um esforço a fim de desenvolver uma solução que viabilize uma vida mais digna, de baixo custo e simples de utilizar, a fim de proporcionar melhor qualidade às pessoas que possuem tetraplegia, uma disfunção motora severa que será explicada mais adiante.

1.2.2 OBJETIVO TÉCNICO

Este trabalho busca integrar os conhecimentos adquiridos durante o curso de engenharia mecatrônica, que abrange pilares de conhecimento: Ciências da Computação, Engenharia Elétrica e Engenharia Mecânica. A base de conhecimento da computação será utilizada para interpretar a interface de comandos do usuário e gerenciar os sistemas embarcados na cadeira de rodas. Na elétrica temos o objetivo de gerar e condicionar os sinais de comando, bem como fazer o acionamento dos motores elétricos. E por fim, na mecânica temos a missão de dar suporte ao sistema da forma mais segura, eficiente e ergonômica possível.

1.3 A DEFICIÊNCIA

Tetraplegia, também conhecida como quadriplegia, é a perda dos movimentos dos braços, tronco e pernas, geralmente, provocada por lesões que atingem a medula espinhal a nível da coluna cervical, devido às situações como traumatismos em acidentes, hemorragia cerebral, sérias deformidades na coluna ou doenças neurológicas.[5]

A perda dos movimentos pode ter intensidades diferentes, que variam desde uma fraqueza até a perda total da capacidade de movimentar o membro. A depender do nível da

lesão, a capacidade respiratória também pode ser comprometida, podendo ser indicado o uso de aparelhos para auxiliar a respiração. (Adaptado de [6])

Segundo o Ministério da Saúde, a maioria dos casos de tetraplegia tem origem traumática, principalmente relacionada à acidentes automobilísticos e ferimentos por projétil de arma de fogo. As causas não traumáticas compreendem cerca de 20% dos casos e são decorrentes de diversas patologias, como tumores intra e extra medulares, fraturas patológicas (metástases vertebrais, tuberculose, osteomielite e osteoporose), estenose de canal medular, deformidades graves da coluna, hérnia discal, isquemia e autoimunes (por exemplo esclerose múltipla). [7]

1.4 METODOLOGIA

1.4.1 ESTUDOS REALIZADOS

Para este trabalho, o estudo bibliográfico iniciou-se buscando trabalhos relacionados realizados dentro da UNB utilizando a Biblioteca Digital da Produção Intelectual Discente da Universidade de Brasília (BDM) e nela foram encontrados os trabalhos [8] e [9]. Posteriormente foi feita uma pesquisa relacionando os termos “cadeira de rodas”, ‘tetraplegia” e “interface” utilizando a plataforma Google Acadêmico e Web of Science, a fim de compreender a produção de conhecimentos relacionados ao tema. Por fim, uma pesquisa foi feita a fim de encontrar soluções comerciais existentes para pessoas tetraplégicas ou com limitações severas de movimentação.

1.4.2 PLANEJAMENTO E EXECUÇÃO DO PROJETO

O planejamento se iniciou por meio de um estudo do trabalho [8], em que se teve uma implementação de duas interfaces de comando: Sopro-Sucção e Joystick de queixo. Em seguida foi feito uma ampla pesquisa em busca das implementações já realizadas com foco em mobilidade para pessoas tetraplégicas, tanto no meio acadêmico quanto no meio comercial, o que permitiu entender melhor as necessidades demandadas e quais as estratégias possíveis para atendê-las. Com isso, foram levantados os requisitos e parâmetros que o projeto deverá atender, com base nos estudos feitos.

Definidos os parâmetros e requisitos, passou-se para uma etapa de geração de ideias alternativas de interfaces possíveis, seguida pela seleção da que foi considerada melhor dentre as criadas.

Feito isso, o passo seguinte foi analisar a cadeira de rodas utilizada em [8] e identificar quais as correções necessárias para fazê-la voltar ao funcionamento, principalmente quanto ao sistema de acionamento dos motores.

Tendo em mente tudo o que deveria ser feito, tanto para a nova interface comando quanto para o acionamento dos motores, foi implementado um sistema simples de comando

para a cadeira via joystick para simplificar o processo de validação do sistema de acionamento dos motores. Com relação a interface, foi feito um estudo de viabilidade, buscando selecionar as soluções adequadas ao projeto.

1.4.3 TESTES DE FUNCIONALIDADE

Os testes escolhidos têm como objetivo comprovar a funcionalidade do sistema, bem com a capacidade de repetibilidade do mesmo. Por isso, a fim de se certificar de que cada parte do sistema funcionava individualmente de acordo com o esperado, a cada nova etapa do processo de projeto concluída, o sistema era imediatamente implementado e testado. Isso fez com que o sistema recebesse uma série de melhorias no decorrer do processo.

CAPÍTULO 2 – REVISÃO BIBLIOGRÁFICA

Neste capítulo serão mostradas as soluções de mobilidade encontradas para pessoas com tetraplegia, destacando suas principais vantagens e desvantagem.

2.1 JOYSTICK

A interface mais comum para o controle de cadeiras de rodas motorizadas é o joystick, pois é simples e intuitivo. Por esses motivos o reposicionamento do Joystick até uma nova posição em que usuário consiga utilizá-lo é uma estratégia recorrente em casos mais severos de deficiência, como a tetraplegia, além de ser uma alteração relativamente barata e fácil quando comparada com as demais soluções que serão discutidas mais adiante.

Em [8], precursor deste trabalho, foi feito a implementação de um joystick de queixo (figura 2.1.1), pois tinha como objetivo a implementação de uma interface de comando pessoas com tetraplegia de baixo custo e fazer um comparativo entre a interface com joystick e a de sopro e sucção. No comparativo, o joystick se mostrou melhor em todos os testes de uso, pois é mais intuitiva para o controle direcional da cadeira, possui apenas um ponto de comando, enquanto a interface de sopro possuía dois, e o acionamento é proporcional a deflexão da haste de comando, permitindo que o usuário controle a velocidade da cadeira, o que não era possível com a interface de sopro e sucção.

Figura 2.1.1 – Joystick de queixo implementado em [8].

Outras variações de joystick podem ser encontradas no mercado e são escolhidas de acordo com o grau de limitação do usuário, como: o mini joystick (figura 2.1.2); o joystick de pé (figura 2.1.3); o joystick de dedos dos pés (figura 2.1.4) e por fim uma solução similar ao que é o Touchpad (figura 2.1.5).

O mini joystick (figura 2.1.2) é recomendado em casos em que não há tônus muscular suficiente para utilizar um joystick comum, pois sua estrutura mecânica foi projetada para minimizar a força necessária para a deflexão da haste. Além disso, o mini joystick funciona exatamente como o joystick comum, permitindo um boa e intuitiva condução da cadeira, com controle de velocidade proporcional a deflexão da haste. Essa solução é desenvolvida para indivíduos com mobilidade e forca reduzidos, ideal para ser utilizado com o dedo, o queixo ou com os lábios.

Figura 2.1.2 – Mini joystick.[10]

O joystick de pé (figura 2.1.3) foi uma solução encontrada pela empresa Smile Smart Tecnology [11] que permite que usuários que possuem o controle, mesmo que pouco, do movimento dos pés possam utilizá-lo para comandar suas cadeiras de rodas. Assim como no caso do mini joystick, as vantagens de manobrabilidade e controle de velocidade estão presentes, porém a solução contempla apenas uma pequena parcela de usuários que controlem minimamente o pé.

Figura 2.1.3 – Joystick de pé [11]

O joystick para dedos dos pés (figura 2.1.4) permite que o usuário utilize os dedos dos pés para girar uma pequena esfera e assim direcionar sua cadeira de rodas.

Novamente, é um joystick com todos as suas funcionalidades, mas também atende a poucos casos.

Figura 2.1.4 – Joystick para dedos dos pés [11]

O Touchpad (figura 2.1.5) apesar de não ser um joystick propriamente dito, seu funcionamento é muito semelhante, além disso é muito recomendado para casos em que não há tônus muscular nos membros superiores, pois para acioná-lo não é necessário força. Das interfaces com joystick, esta apresenta o menor grau de esforço, tornando-a muito confortável para o uso diário. Porém, além de necessitar do uso dos membros superiores, esta interface pode executar movimentos involuntários indesejados caso o usuário encoste acidentalmente na região de leitura, visto que pode não possuir pleno controle dos braços.

Figura 2.1.5 – Touchpad [12]

2.2 SOPRO E SUCÇÃO

Esta interface é utilizada em casos em que se tem a perda total ou quase total do movimento dos membros inferiores e superiores.

Como dito no tópico anterior, esta interface foi implementada em [8] (figura 2.2.1). Seu acionamento era feito a partir de dois tubos acoplados a sensores de pressão, cada sensor com dois tipos de comando discreto: um para sopro e outro para sucção (figura

2.2.2). Essa configuração permitiu que em um dos tubos se comandasse para frente (sopro) e para trás (sucção), enquanto em outro comandasse o giro para a esquerda (sopro) e para a direita (sucção).

Figura 2.2.1 – Sensor de pressão artesanal [8]

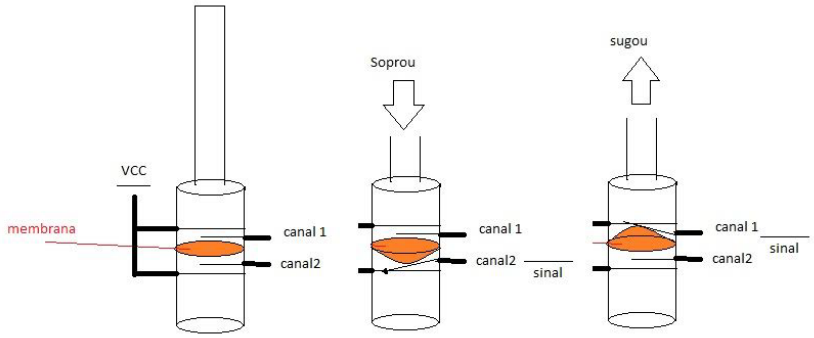


Figura 2.2.2 –Princípio de funcionamento do sensor de pressão [8]

Apesar de a implementação funcionar de forma aceitável, algumas características dificultavam seu uso, uma é a membrana utilizada, que era ligeiramente rígida para evitar movimento involuntários, mas que exigia um pouco mais de esforço do usuário para fazer o acionamento, tanto de sopro quanto sucção. Uma alternativa ao sensor de pressão utilizado sugerida no próprio trabalho é substitui-lo por um sensor de fluxo proporcional, o que também tornará um controle de velocidade proporcional à intensidade do sopro ou sucção.

Outra questão seria na implementação, que tem a manobrabilidade prejudicada por não ser possível fazer movimentos combinados de deslocamento e giro.

Além desse método, há muitos outros possíveis, como o implementado em um trabalho desenvolvido por Claudio Lima Lopes Ferreira na Universidade Estadual de Londrina [13] em 2008 (figura 2.2.3).

Nele, foi utilizado um único tubo rígido acoplado a um transdutor de ar para fazer a detecção do sopro e a determinação da direção e velocidade do movimento foi feita a partir de um display de 25 LEDs (figura 2.2.4).

Figura 2.2.3 – Transdutor de ar [13]

Figura 2.2.4 – Painel de LEDs [13]

O acionamento se dava em 4 etapas: seleção da direção (indicada de forma cíclica pelo painel de LED entre 8 diferentes possibilidades), seleção da velocidade (também mostrada pelo painel), execução do movimento selecionado e interrupção do movimento. Por mais que se tenha um controle de velocidade e 8 direções possíveis, é pouco dinâmica por necessitar de excessivos comandos para cada movimento. Além disso, caso o usuário erre a direção ou o sentido desejado, ele não tem escolha a não ser ir assim mesmo.

O autor do trabalho deixa claro que a ideia não é ser uma solução completa de mobilidade para o usuário, mas sim proporcionar uma interface viável que possibilite sua movimentação de forma amigável e confortável.

Um terceiro trabalho, também da Universidade de Londrina, propõe uma interface de controle por sopro e sucção [14]. Nele, foi projetado um LVDT (Transformador Diferencial Linearmente Variável) duplo, de forma que um ramo fosse ativado somente no sopro e o outro somente na sucção.

De maneira similar ao feito em [13], nesta implementação o usuário escolhe a direção em que vai se mover a partir das opções disponíveis mostradas em um display (figura 2.2.5), só que nesse caso com apenas 4 direções possíveis: para frente, para trás, girar à direita e girar à esquerda.

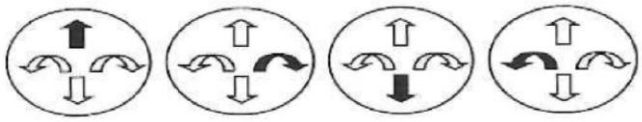


Figura 2.2.5 – Indicação de função no display: seguir em frente, girar à direita, ré e girar a esquerda. [14]

Por mais que se tenha perdido o controle de velocidade e algumas direções possíveis de comando, muito se ganhou em agilidade no processo de escolha e execução de comandos. Para controlar o sistema, o usuário pode succionar no tubo ligado ao LVDT, isso faz o comando selecionado no display se alterar para o próximo, seguindo o sentido horário. Para executar o movimento selecionado, basta soprar, o mesmo vale para interrompê-lo, basta soprar.

Diferentemente dos métodos mostrados anteriormente, o direcionamento do movimento da cadeira é feito por dois motores de passo acoplados às rodas direcionais dianteiras, que giram de acordo com a direção desejada, tornando mais preciso o direcionamento, porém mais caro e complexo.

A interface de sopro e sucção possui alguns exemplares comerciais disponíveis, como é o feito pela empresa Smile Smart Technology[11], que consiste em um simples switch comandado por sopro e sucção que pode ser incorporado a um de seus modelos comerciais. O modo de utilização desse comando é implementado de acordo com as limitações de cada usuário, uma vez que a empresa possui uma série de outros acessórios que podem ser adicionados como entradas de controle que se complementam a fim de tornar o sistema mais robusto e confortável a cada usuário.

Uma outra empresa ASL (Adaptative Switch Laboratories) possui uma solução diferente, em que também utiliza o comando de sopro e sucção para fazer a cadeira se mover para frente e para trás, respectivamente, mas o movimento para a esquerda e para a direita é dado por botões nas laterais do encosto de cabeça (figura 2.2.6). Essa solução é muito interessante, pois permite uma condução mais fluida e intuitiva ao usuário. [15]

Figura 2.2.6 – Interface de controle comercializada pela empresa ASL [15]

2.3 MOVIMENTAÇÃO DA CABEÇA

A movimentação da cabeça também pode ser utilizada comandar o movimento da cadeira de rodas, como mostra o estudo [16]. Nele foi implementado o acionamento da

cadeira de rodas a partir da inclinação da cabeça, utilizando para isso um acelerômetro no topo da cabeça do usuário (figura 2.3.1).

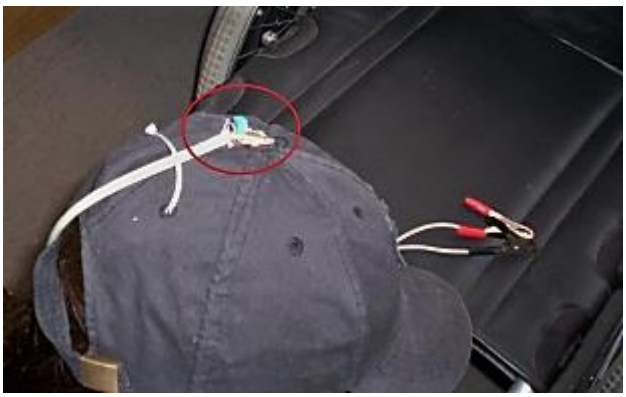


Figura 2.3.1 – Boné com acelerômetro fixado [16]

O sistema iniciava realizando uma calibração, orientada por um display, e nela eram estabelecidos os limites de movimentação a partir do qual se iniciaria a movimentação, isso fez com que o sistema fosse bem flexível às limitações de cada usuário. Para evitar que comandos involuntários ocorram, o sistema exige que durante o processo de calibragem, os comando válidos tenham uma inclinação mínima de 3,4º. Além disso, como o sistema interpreta apenas a inclinação da cabeça como comando válido, fazendo isso por meio de uma relação matemática que engloba as leituras do acelerômetro nos 3 eixos, isso reduz as chances de comandos involuntários caso o usuário gire a cabeça para olhar um objeto ao seu lado ou atrás dele, desde que não incline a cabeça.

Outra forma de utilizar a movimentação da cabeça foi implementada pela empresa Australiana Magic Mobility[17] (figura 2.3.2).

Figura 2.3.2 – Interface de comando fixada no encosto de cabeça.[17]

O sistema utiliza um joystick especial montado atrás da cabeça do usuário que converte a movimentação do encosto de cabeça em comandos de movimentação para a

cadeira. O encosto admite três tipos comandos (para esquerda, para direita e para trás) que medem a força de acionamento, de forma a fazer um controle de velocidade proporcional. Visto que apenas 3 comandos são insuficientes para as 4 direções básicas (para frente, para trás, para esquerda e para direita), a interface conta com uma chave para alternar o comando de empurrar a cabeça para trás entre andar para frente e dar ré.

2.4 MOVIMENTAÇÃO DOS OLHOS

Em casos de limitações mais severas, há a opção de utilizar a movimentação dos olhos para movimentar a cadeira de rodas, foram encontradas duas técnicas: Eletro-oculografia e Video-oculografia.

A Eletro-oculografia consiste em utilizar os sinais nervosos que movimentam os olhos como interface de comando e foi feito em [18]. Nele o equipamento funciona a partir da utilização de uma máscara (figura 2.4.1), pelo usuário, e nela estão os eletrodos para a captação dos sinais nervosos responsáveis pela movimentação dos olhos. Além disso, há um circuito de amplificação da ordem de 1 para 1 milhão, de forma que seja possível a interpretação do sinal.

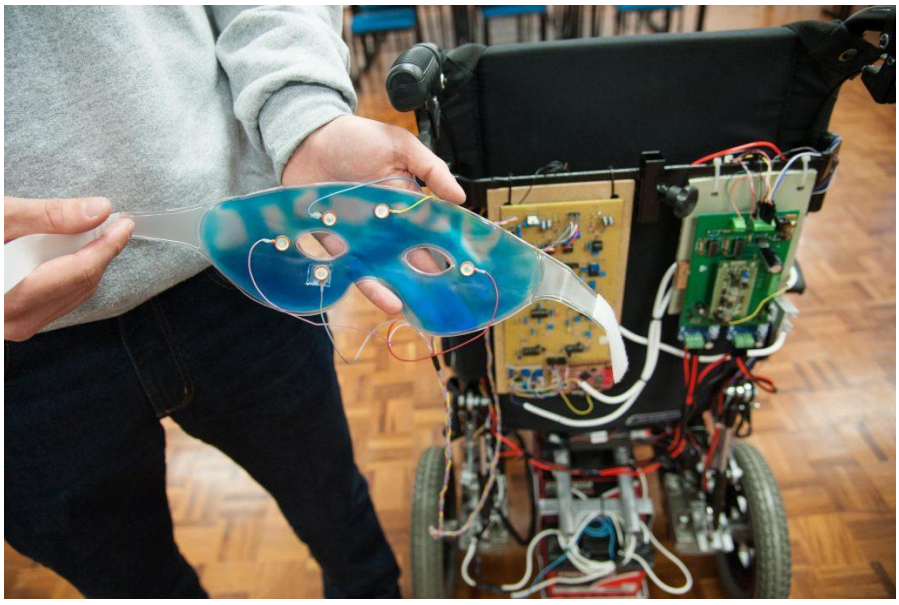


Figura 2.4.1 – Máscara de aquisição de sinais nervosos [18]

“Os comandos de movimento só são acionados na extremidade do movimento ocular. Assim, a pessoa pode olhar para pessoas e objetos normalmente, sem risco de ativar a locomoção da cadeira”, explica Maikon, desenvolvedor do projeto.

A interface é muito vantajosa para casos severos, porém ainda necessita de diversas melhorias antes de se tornar robusta para utilização segura de usuários, uma vez que seus circuitos eletrônicos são muito sensíveis a interferências eletromagnéticas.

Na Video-oculografia utiliza uma câmera para capturar a movimentação da pupila e foi implementado em [19]. Nele foi utilizado um dispositivo chamado Pupil Pro (figura 2.4.2), que possui duas câmeras acopladas a uma armação que se acomoda ao rosto do usuário, sendo uma para a leitura dos olhos e outra do ambiente, entretanto esta segunda não foi utilizada na implementação, conferindo mais agilidade ao processamento de imagens.

Figura 2.4.2 – Pupil Pro [19]

“O protocolo de navegação utiliza sequências de posicionamento da pupila (1-acima, 2- direita, 3- esquerda, 4- abaixo) e uma posição chamada de “zero”, que corresponde aos olhos fechados e determina o gerenciamento do início do movimento ou da parada por piscadas do olho. Assim que as imagens da pupila são captadas, elas são processadas em tempo real e convertidas em comandos de direção (frente, atrás, esquerda, direita), na velocidade pré-programada, pelo software desenvolvido pelo grupo. Os comandos interpretados pelo software são enviados para um microcontrolador ligado a um circuito de amplificação que aciona os motores da cadeira.” [19]

2.5 MOVIMENTAÇÃO DA FACE

De forma similar ao que foi mostrado para interfaces de comando pelo movimento dos olhos, há interfaces que utilizam os movimentos dos músculos da face como sinais de comando.

Um trabalho desenvolvido na Universidade Federal do Rio Grande do Sul [20] se dedicou a criar um método para adquirir sinais mioelétricos provenientes do movimento das pálpebras e utilizá-lo no controle de uma cadeira de rodas motorizada. O sistema é composto de um eletromiógrafo com ganho e filtragem configuráveis digitalmente através de um microcontrolador e um algoritmo de detecção de piscadas voluntárias e utilização destas para a ativação de uma cadeira de rodas. A partir de um protocolo pré-determinado, foi possível relacionar o movimento das pálpebras com comandos que são transmitidos à cadeira de rodas motorizada.

Os eletrodos de leitura foram posicionados abaixo dos olhos e o eletrodo de referência no meio da testa, como mostra a figura 2.5.1.

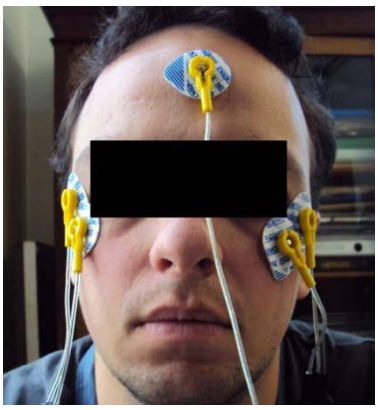


Figura 2.5.1 – Posição dos eletrodos de controle [20]

Os comandos realizados com as piscadas só são atendidos dois segundos após a finalização da primeira piscada e no mínimo 100 milissegundos após o fim da última piscada. O comando para andar para frente pode ser dado piscando uma vez os dois olhos ao mesmo tempo. Para movimentar para a esquerda, deve piscar o olho esquerdo duas vezes e uma vez o olho direito, a inversão disso resulta no movimento para a direita, ou seja, piscar duas vezes o olho direito e uma vez o esquerdo. O comando parar é dado piscando os dois olhos simultaneamente, uma ou mais vezes, durante a movimentação, isso faz com que o movimento não seja muito longo, pois logo é interrompido.

Devido ao fato de que o objetivo do trabalho foi desenvolver um sistema de aquisição de sinais mioelétricos, principalmente os do movimento da pálpebras, bem como um software para o devido tratamento e interpretação de tais sinais, não foi feita uma implementação embarcada da interface em uma cadeira de rodas de fato, o que impossibilita saber como é a experiencia do usuário com a mesma.

2.6 LÍNGUA

Outra interface desenvolvida é utilizar a língua para comandar, não somente a cadeira de rodas, mas outros dispositivos, como um smartphone (figura 2.6.1).

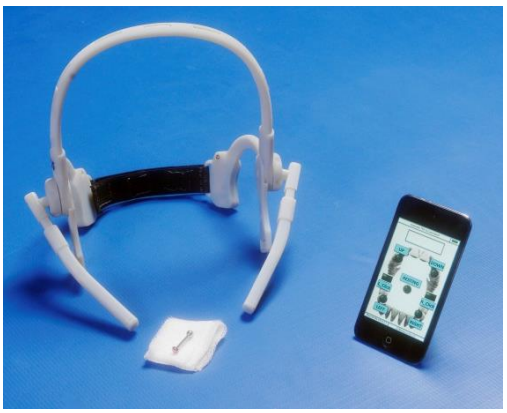


Figura 2.6.1 – Dispositivo de leitura, piercing magnético e interface mobile. [21]

Esse projeto foi desenvolvido e implementado em [21], na Georgia Institute of Tecnology, e a ideia consiste em monitorar o movimento da língua por meio de um piercing com um imã em sua extremidade, com isso sensores posicionados nas laterais do rosto conseguem determinar a posição da língua na boca (figura 2.6.2).

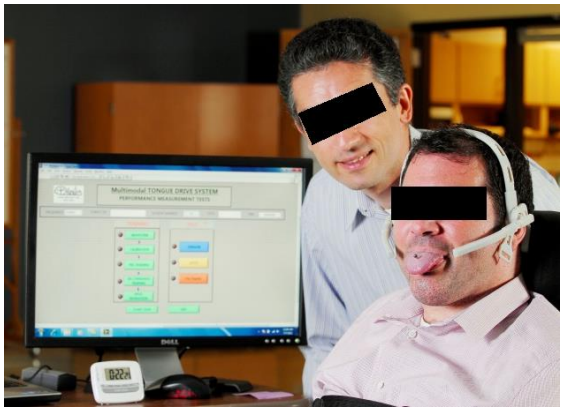


Figura 2.6.2 – Autor do estudo e usuário em testes do sistema. [21]

A interface apresenta controle discretos relacionados a cada região preestabelecida em que a língua pode estar e um empecilho um tanto obvio que é a necessidade de o usuário ter sua língua furada para a colocação do piercing.

2.7 VOZ

As soluções que utilizam comando de voz são aconselháveis em casos severos de limitação dos movimentos, porém podem ser muito mais limitados, como visto em [22]. O trabalho foi batizado de “Smart chair” e desenvolvido um protótipo em miniatura, para que fossem validadas as teorias de controle por voz e sensoriamento, de forma a tornar a cadeira o mais segura possível, com sensores de obstáculos e inclinação, evitando que o usuário solicite comandos que representem riscos.

O sistema foi desenvolvido utilizando um smartphone que recebe o comando de voz, interpreta e o envia ao controlador da cadeira, nesse caso um Arduino Mega, que verifica

todos os sensores a fim de verificar se o comando é executável sem riscos e então toma sua decisão de agir ou não.

A limitação neste caso está no conjunto de cinco comandos possíveis da cadeira se movimentar: para frente, para trás, para esquerda, para direita e parar, ou seja, não há direções intermediárias (figura 2.7.1). Além disso, o sistema não permite o controle da velocidade e é muito susceptível a interferências externas, tanto pelo usuário não ser interpretado devido a ruídos excessivos quanto por comandos involuntários devido a interpretações de comandos indevidos.

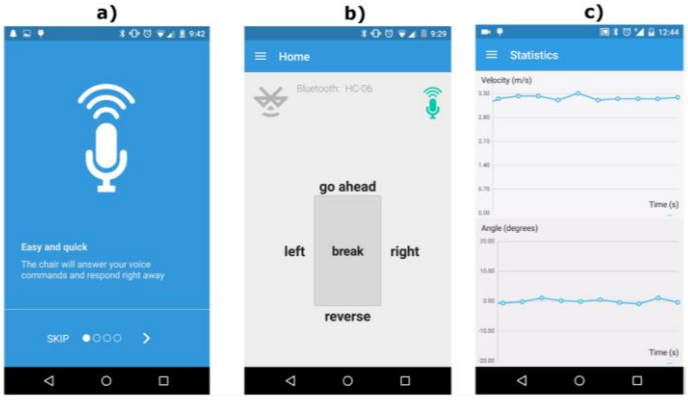


Figura 2.7.1 - a)Tutorial da interface. b)Tela Principal. c)Gráficos de monitoramento.[22]

2.8 TECNOLOGIAS ASSISTIVAS AUXILIARES

Algumas soluções utilizadas que não são propriamente um tipo de interface para a cadeira de rodas propriamente, mas viabilizam uma maior acessibilidade em alguns casos.

Um exemplo é um mouse giroscópio (figura 2.8.1) desenvolvido pela empresa Smile Smart Tecnology [23] que permite ao usuário utilizar a movimentação de qualquer parte do corpo, mesmo que de maneira limitada, para controlar um computador com uma distância de até 10 metros. O dispositivo é pequeno e pode ser fixado utilizando uma faixa que o acompanha ou por meio de um clip. Essa tecnologia permite que o usuário interaja com plataformas e interfaces mais complexas que antes necessitariam de adaptações para seu uso.

Figura 2.8.1 – Mouse Giroscópio [23].

CAPÍTULO 3 – PROJETO

Este capítulo serão mostradas todas as questões relevantes que direcionaram o projeto. Nele é feita uma análise do que foi encontrado na revisão bibliográfica, focando em nos aspectos considerados mais relevantes a fim de gerar os requisitos para a o projeto da interface a ser desenvolvido.

3.1 PREMISSAS E REQUISITOS

Este trabalho utiliza como ponto de partida outro trabalho anteriormente realizado [8], dele foi utilizado a estrutura da cadeira, juntamente com os motores, e todo o conhecimento gerado de projeto e dos testes realizados.

Com base na revisão bibliográfica feita, foi montada a tabela 3.1.1 com os aspectos considerados relevantes para classificar as soluções encontradas, em decorrência disso foram estabelecidos como requisitos a serem atendidos: baixo custo, alta manobrabilidade, pouco susceptível à interferências, fácil adaptação a maioria dos usuários e que seja de fácil implementação e adaptação à estrutura que será utilizada.

A questão de manobrabilidade foi o primeiro critério a ser analisado, pois está diretamente ligada a experiência do usuário e com isso destacam-se as interfaces usando joystick e movimentação da cabeça.

Com relação à interferência, a maioria das interfaces se saem bem, mas algumas podem ter seu funcionamento comprometido, como no caso da interface de voz, que pode não compreender o usuário em ambientes muito barulhentos. Outra que também pode não funcionar em determinadas condições é a Video-oculografia em ambientes muito escuros, pois a câmera não será capaz de captar imagens nítidas o suficiente. Por fim, a interface que mais é prejudicada por interferências é a eletromiografia da face, pois utiliza sinais de baixa amplitude e é facilmente afetado por equipamentos eletroeletrônicos que emitem ondas eletromagnéticas.

O custo da interface tem uma relação direta com os insumos utilizados e podem ser reduzidos de diferentes formas durante a implementação, entretanto métodos que necessitam de equipamentos de alta precisão, como a eletromiografia e Video-oculografia, não se enquadram dentro escopo desejado.

Com base no que foi dito nesta seção e nos dados apresentados na tabela 3.1.1, chegou-se à conclusão de que o joystick de queixo é de fato o método mais conveniente para se controlar uma cadeira de rodas motorizada para pessoas tetraplégicas, pois atende a todos os requisitos desejados. Contudo, na busca por oferecer mais que uma simples solução para um usuário, foi decidido dar ao sistema comum de acionamento por joystick de

queixo a funcionalidade automática de deslocamento dentro de um ambiente conhecido onde o usuário passe longos períodos, o que dará um maior conforto à ele. Por exemplo, durante um deslocamento automático o usuário poderá interagir e conversar com as pessoas a sua volta sem interromper seu trajeto, o que antes não era possível ser feito simultaneamente.

Tabela 3.1.1 – Comparativo entre as interfaces encontradas

Interface	Joysticks	Sopro e Sucção	Acelerômetro de cabeça	Encosto de cabeça	Movimento dos Olhos	Comando de Voz	Língua	Eletromiografia da face
Principais pontos positivos	São os mais intuitivos para o usuário para direcionar e controlar a velocidade da cadeira. Além de serem mais baratos e de simples implementação.	Boa opção para casos severos de limitação e de baixo custo.	Interface pouco invasiva, o usuário não terá qualquer equipamento em seu campo de visão, e mais confortável, isso fará com que a sua movimentação seja mais natural na experiência do usuário.	Interface de condução fluida e intuitiva. Permite várias formas de implementação diferentes: Botões, sensores de proximidade, potenciômetros lineares/joysticks e etc.	Beneficia pessoas com extrema limitação de mobilidade e, nos casos de certa mobilidade nas mãos, as deixa livre para outras atividades (como comandar um braço robótico).	Possível implementação com smartphone, o que permite comandar outras coisas além da cadeira.	Opção para casos severos de mobilidade limitada. Permite a utilização de diversas regiões musculares, bastando para isso o correto posicionamento e calibração.	Opção para casos severos de mobilidade limitada. Permite a utilização de diversas regiões musculares, bastando para isso o correto posicionamento e calibração.
Manobrabilidade	Alta	Média	Alta	Alta	Baixa	Baixa	Média	Baixa
Dificuldades do usuário	Exige mobilidade de algum membro do corpo	Obter boa dirigibilidade e fluidez	Necessita da mobilidade do pescoço	Necessita da mobilidade do pescoço	Adaptar-se a interface	Obter boa dirigibilidade e fluidez	Adaptação à interface e colocação do piercing.	Adaptação à interface
Possíveis interferências	Baixa Interferências	Baixa interferência	Movimentos involuntários	Movimentos involuntários	Movimentos involuntários, luz ambiente, contaminação das lentes etc.	Vento, barulho e interpretação de comandos dados por terceiros	Fala e agentes magnéticos externos	Espasmos e movimentos involuntários, sujeito a interferências eletromagnéticas.
Interpretação dos dados	Sinal analógico proporcional e de comando direto	Para sensores de pressão: direta (on/off); Para sensores de fluxo: Implementada em software.	Leitura de sinal analógico pelos sensores.	Similar ao joystick	Processamento de imagens	Software especializados em interpretação de comandos de voz	Circuitos de filtragem analógica	Circuitos de filtragem analógica
Controle de Velocidade	Sinal analógico proporcional e de comando direto	Discreto (para sensores de pressão) ou proporcional ao fluxo de ar (para sensores de fluxo).	Proporcional ao grau de inclinação da cabeça	Proporcional a força aplicada	Discreto e definido em software	Discreto e definido em software	Discreto e definido em software	Discreto e definido em software
Custo	baixo	baixo	alto	médio	Alto	Médio/alto	alto	alto
Dificuldade de Implementação	Ajustar mecanismo de acionamento de acordo com a habilidade do usuário.	Ajustar a sensibilidade do transdutor de ar, de forma a não sofrer interferência do vento, mas sem exigir muita força do sopro do usuário.	Fazer a correta interpretação das leituras do acelerômetro como sendo o movimento de inclinação da cabeça.	Fazer uma adaptação suficientemente robusta e confortável ao usuário, permitindo ainda que consiga movimentar a cabeça livremente quando queira.	Fazer a correta interpretação dos comandos em diversos ambientes, das mais variadas condições de iluminação.	Garantir que o usuário vai ter o controle mesmo em ambientes ruidosos.	Fazer a correta detecção da posição da língua a partir dos sensores.	Filtrar ruídos e eliminar interferências dos sinais elétricos.

3.2 ESTRUTURA

Como dito no tópico anterior, a estrutura da cadeira (figura 3.2.1) foi herdada de um trabalho anterior [8] e consiste em um modelo comercial que a princípio era de acionamento manual e foi convertida para acionamento elétrico, tocada por dois motores elétricos (figura 3.3.1).

Figura 3.2.1 – Cadeira de Rodas utilizada [8]

Para tornar a cadeira motorizada, foram confeccionados dois suportes para os motores elétricos, um de cada lado, posicionados como mostra a figura 3.2.2.

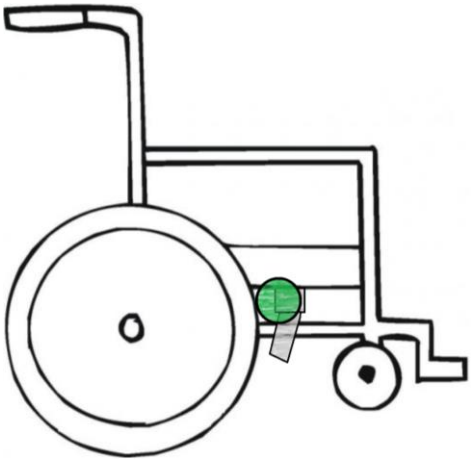


Figura 3.2.2 – Representação dos suportes para motores

O motor possui uma polia feita em alumínio acoplada em seu eixo que transmite sua rotação para as rodas da cadeira contato com os pneus. Essa implementação, fez com que houvesse uma redução de velocidade, quando comparada a velocidade do eixo do motor com a das rodas da cadeira, mas em um ganho de torque, muito importante para o sistema que funciona para grandes cargas.

3.3 POTÊNCIA

O sistema utiliza dois motores de corrente contínua da fabricante Motron e modelo MR-210 (Figura 3.3.1).

Figura 3.3.1 – Motor MR-210 VER 240 [8]

Suas especificações de funcionamento foram encontradas em uma etiqueta no próprio motor e colocadas na tabela 3.3.1.

Tabela 3.3.1 _ Especificações motor MR-210 VER 240

FABRICANTE / MODELO	MOTRON MR-210 VER 240
Tipo	Corrente Contínua
Massa	1,9 Kg
Tensão	12 V
Velocidade Angular Nominal	240 RPM
Corrente	6 A
Torque	24kgf.cm
Potência	144W

Considerando que a dinâmica desejada para a cadeira é necessário que os motores possam ser acionados em ambos os sentidos, horário e anti-horário, a topologia do circuito inicialmente escolhida foi a de ponte H (Figura 3.3.2).

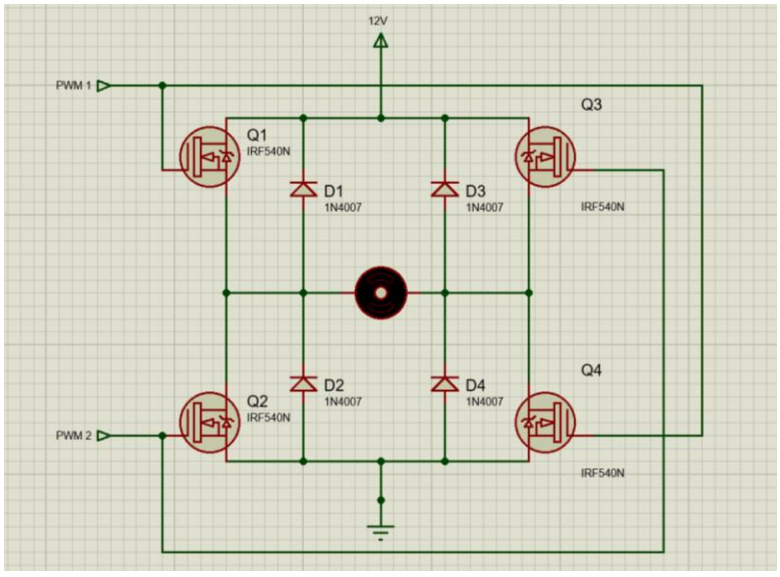


Figura 3.3.2 _ Circuito ponte H

Para montagem dessa topologia foi utilizado o transistor MOSFET IRF540N devido a suas caraterísticas de chaveamento rápido, por volta 4MHz, e grande capacidade de corrente coletor-emissor. Em cada transistor foi montado com um dissipador de alumínio, como mostra a figura 3.3.3.

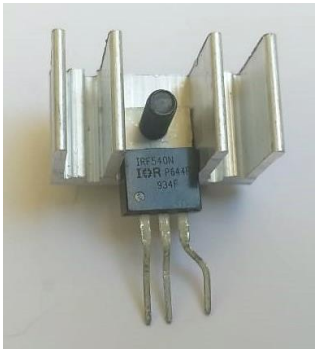


Figura 3.3.3 – Transistor MOSFET IRF540N com dissipador térmico

Devido ao comportamento do sistema observado durante os testes, viu-se a necessidade de se utilizar um módulo PWM com maior capacidade de dissipar o calor gerado pela alta corrente de acionamento dos motores, pois os dissipadores utilizados não se mostraram eficientes o bastante. Com isso foi selecionado o Módulo Driver Ponte H – BTS7960, mostrado na figura 3.3.4, que possui um dissipador consideravelmente maior ao anteriormente utilizado. As especificações do módulo estão apresentadas na tabela 3.3.2.

Figura 3.3.4 – Módulo Driver Ponte H – BTS7960 [24]

Tabela 3.3.2 _ Especificações Módulo Driver Ponte H – BTS7960

Modelo	Módulo Driver Ponte H – BTS7960
Tensão de alimentação (V)	5
Tensão de acionamento (V)	5
Tensão de saída (V)	5,5 a 27
Máxima corrente contínua (A)	43
Máxima corrente de pico (A)	60
Dimensão (mm)	50x50x48
Proteção	Térmica, sobre tensão, sub tensão e sobre corrente.

A fim de proteger o sistema houve a necessidade de fazer um isolamento do circuito de alta potência (módulo driver ponte H, motores e Bateria) da parte de baixa potência (microcontrolador), para isso foram utilizados 4 opto isoladores MOC3022, um para cada sinal de comando enviado aos módulos. O circuito mostrado na figura 3.3.5 representa o isolamento feito para um sinal PWM apenas, no caso o sinal enviado pelo controlador (PWM1_MICROCONTROLADOR) chega ao opto isolador, que por sua vez gera o sinal utilizado que comanda o módulo ponte H (PWM1_PONTE H)).

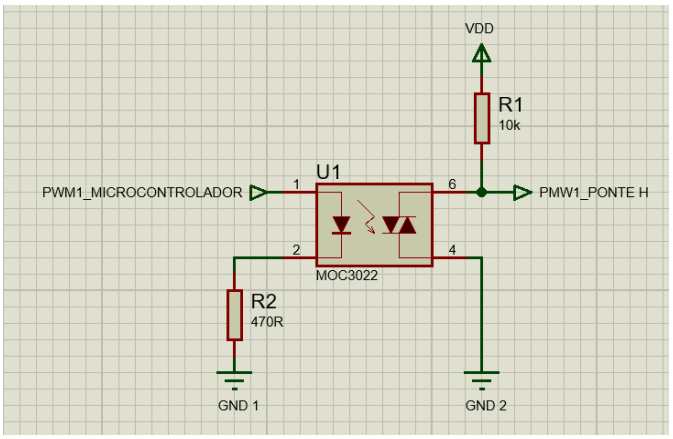


Figura 3.3.5 _ Circuito de isolamento utilizando MOC3022

Como fonte de energia para o sistema foi utilizada uma bateria de 12 Volts e capacidade de 45 Ah (Figura 3.3.6). A escolha dessa bateria foi motivada pelo fato da tensão nominal coincidir com a tensão de acionamento dos motores (12 Volts), além de possuir grande capacidade e ser de fácil obtenção no mercado.

Figura 3.3.6 – Bateria Automotiva 12 Volts 45Ah.

3.4 CONTROLE

Para fazer o controle do sistema foi escolhido a plataforma Arduino Leonardo (Figura 3.4.1), por diversos fatores, tais como facilidade e familiaridade com o ambiente de

desenvolvimento, baixo custo, grande disponibilidade no mercado, capacidade de processamento e grande quantidade de funcionalidades das portas (leitura analógica até 5V, entradas e saídas digitais, modulação PWM e etc.).

Por mais que a plataforma Arduino seja relativamente fácil de ser utilizada, ela exige que se conheça bem suas capacidades para que a aplicação funcione dentro do esperado. Por isso, foi feito um estudo acerca das funcionalidades desejadas e em quais portas estariam disponíveis. O esquema da figura 3.4.1 mostra a disposição das portas da plataforma.

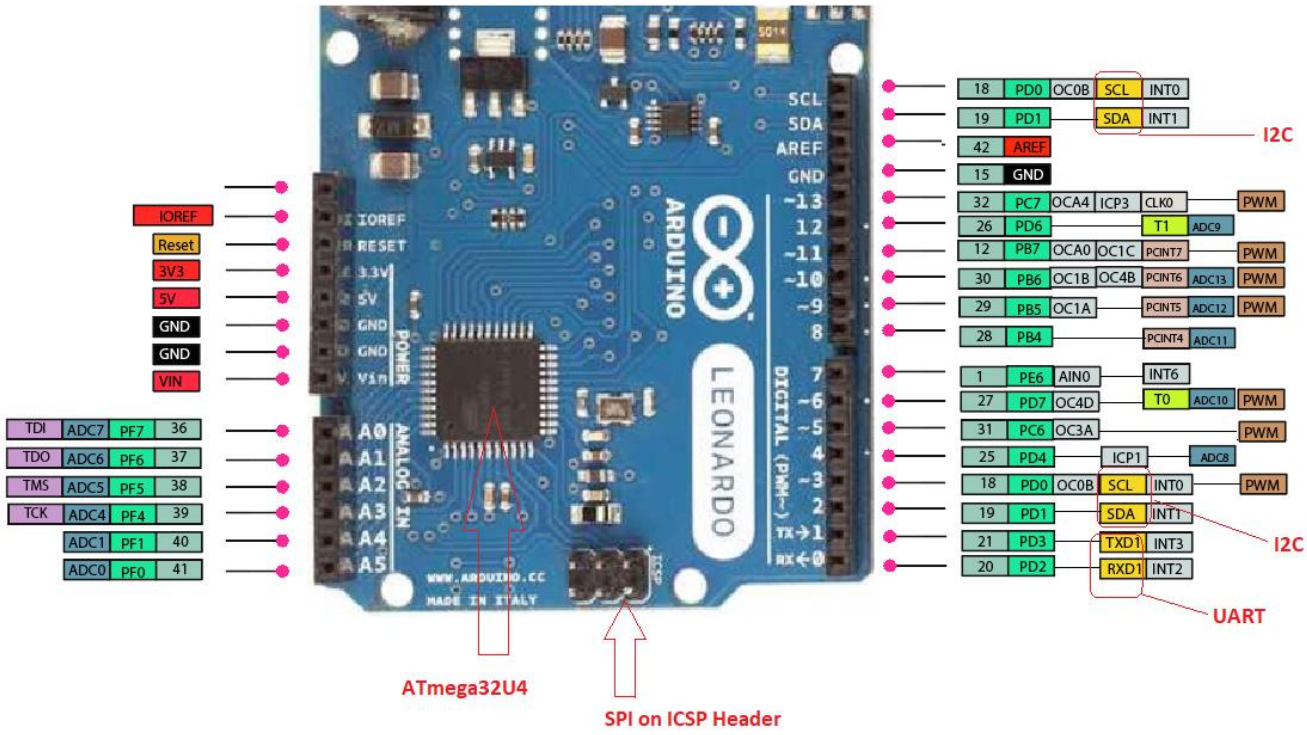


Figura 3.4.1 – Portas de entrada e saída Arduino Leonardo [25]

A partir disso, foi possível garantir que fossem utilizadas as portas corretas para a leitura do posicionamento do Joystick (A0 e A1) e também portas que emitissem sinais PWM de mesma frequência para o controle dos motores, pois o sinal de saída das portas 3 e 11 tem frequência de 980 Hertz e as demais portas tem frequência de 490 Hertz. Além disso, foram utilizadas, para a leitura dos sensores nas rodas, as portas 2 e 3 em modo de interrupção, de forma a garantir que nenhum sinal de deslocamento fosse perdido.

3.5 INTERFACE DE COMANDO

Inicialmente foi elaborada uma interface simples utilizando um joystick de 3 eixos (Figura 3.5.1) para que os primeiros testes de validação dos sistemas de potência e de controle.

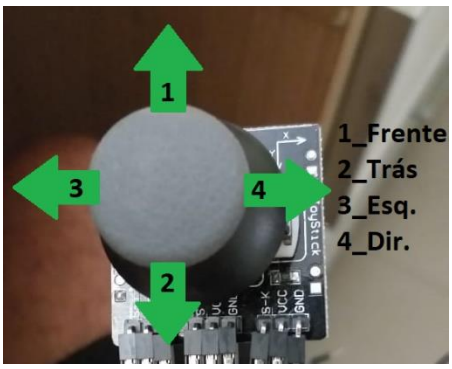


Figura 3.5.1 – Joystick de 3 eixos

A dinâmica escolhida foi simples por ser destinada apenas para teste e consiste em 5 estados: Parado, para frente, para trás, giro para a esquerda e giro para a direita, com um acionamento dos motores proporcional a deflexão do joystick.

Posteriormente foi fabricado um controle de 4 botões (figura 3.5.2) a fim de faz comparações entre os dois tipos de acionamento: proporcional (joystick) e ON/OFF (botões).

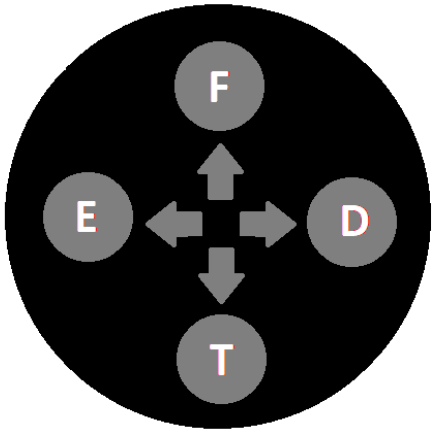


Figura 3.5.2 – Controle por botões

Para evitar trancos foi implementada uma rampa de acionamento dos motores, utilizando modulação PWM. O código de controle implementado nesta primeira etapa foi feito em linguagem C e está disponível no Apêndice 1 deste relatório.

Após a realização dos testes de estrutura, potência e controle, o sistema passou a estar apto a receber uma nova interface.

3.5.1 VISÃO GERAL DO SISTEMA SEMI AUTOMÁTICO

O sistema possui dois modos de funcionamentos: Manual e Automático (figura 3.5.1.1). No modo manual, a cadeira se movimenta conforme a movimentação do joystick posicionado frente ao queixo, com 5 comandos possíveis: parado, para frente, para trás, giro para esquerda e giro para direita. No modo automático, o usuário seleciona o cômodo

desejado mostrado na tela do computador e então o sistema calcula a trajetória a ser seguida para alcançar o destino desejado e então se desloca até lá. Para que o usuário possa visualizar e selecionar o cômodo para o qual deseja ir, foi utilizado um notebook. Além disso, o sistema não deve perder sua funcionalidade anterior, de funcionar motorizado sem a necessidade de um computador, pois esses equipamentos não possuem alta durabilidade de bateria e isso resulta em um problema para o usuário, por isso o sistema foi configurado para iniciar em modo manual, o que permite ao usuário usar a cadeira normalmente sem a função automática, senso necessário apenas fornecer uma fonte de energia como, por exemplo, uma bateria portátil.

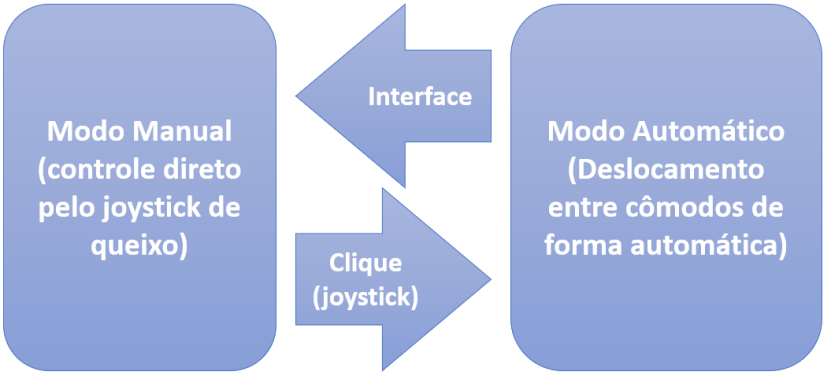


Figura 3.5.1.1 – Modos de funcionamento do sistema semi automático.

3.5.2 MUDANÇAS NA ESTRUTURA

Na estrutura original da cadeira, seria necessário elaborar dois suportes para o novo sistema, um para o joystick ser posicionado frente ao queixo e outro para o computador que exibe a interface visual. O primeiro suporte deve ser construído com tubos de metal de ¾ de polegada, mesmo diâmetro do que foi utilizado nos apoios de braços da cadeira, onde deseja-se fixá-lo. Além disso, esse suporte deve ser móvel de forma não atrapalhe o embarque e desembarque do usuário, como mostra figura 3.5.2.1.

Figura 3.5.2.1 – Representação do suporte para o joystick de queixo

O segundo suporte não foi necessário ser fabricado, pois a prefeitura da universidade disponibilizou uma mesa auxiliar que se adequou perfeitamente para o projeto e a estrutura.

3.5.3 MUDANÇAS DE HARDWARE

Para que o usuário realizasse o evento de clique foi utilizado o terceiro eixo disponível no joystick, exigindo apenas a adição de um fio para a leitura do joystick até o controlador.

A fim de medir o deslocamento da cadeira, dois encoders, um de cada lado, são necessários. Partindo da premissa de que não haja desacoplamento do motor e nem escorregamento entre a polia de alumínio acoplada ao eixo do motor e o pneu da cadeira, foram fixados dez imãs de neodímio na parte interna da polia, conforme mostrado na figura 3.5.3.1, e para detectá-los foi utilizado um sensor de efeito hall. A quantidade de imãs foi estabelecida buscando maximizar o número pulsos por volta para uma boa resolução, sem que houvesse a sobreposição dos campos magnéticos.

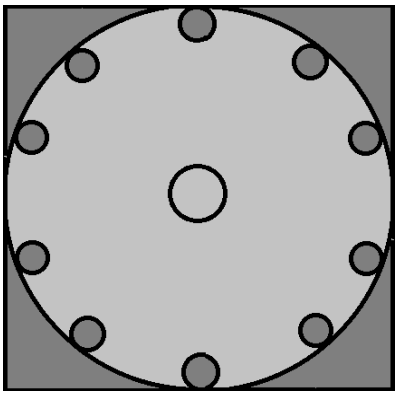


Figura 3.5.3.1 _ Representação da polia com imãs

Sabendo que a polia de alumínio tem um diâmetro de 10 cm (figura 3.5.3.2), é possível obter a resolução de deslocamento dividindo o perímetro da polia por 10, número de pulsos por volta, e com isso obter que cada pulso detectado no encoder equivale a 7,85 cm de deslocamento da roda em que a polia está acoplada. Com isso, também é possível obter angular com relação a roda maior, dividindo seu perímetro pela resolução de deslocamento obtida, resultando em 6º de rotação da roda a cada pulso.

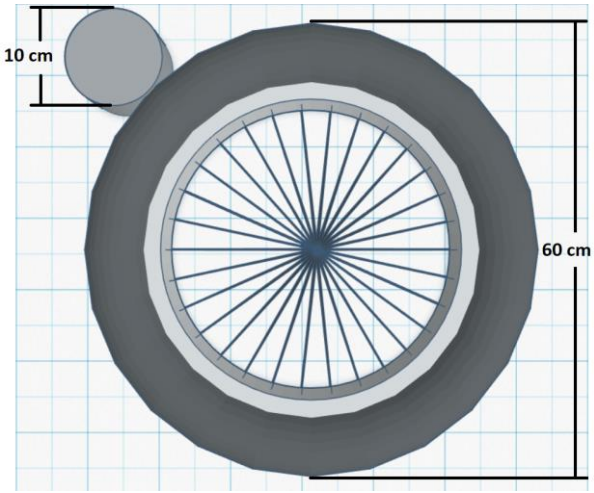


Figura 3.5.3.2 – Montagem polia-roda.

O sensor de efeito Hall deve ser fixado na estrutura em que o motor é parafusado conforme mostrado na figura 3.5.3.3.

Figura 3.5.3.3 _ Posicionamento do sensor Hall

Com a informação de deslocamento de cada roda é possível calcular a posição e direção da cadeira. A modelagem da cadeira é mostrada na figura 3.5.3.4, sendo X_I e Y_I os eixos das coordenadas inerciais relativas ao ambiente mapeado, X_R e Y_R os eixos de coordenadas da cadeira, P o ponto central do eixo que liga o centro das duas rodas tracionada e Θ_R a rotação do sistema de coordenadas da cadeira com relação ao sistema inercial [26].

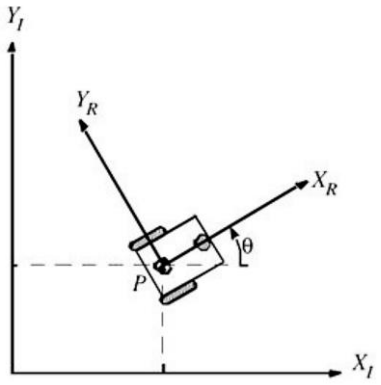


Figura 3.5.3.4 – Representação da cinemática do sistema. (Adaptado de [26])

Para obter a cinemática de posicionamento da cadeira, representado pelo ponto P central, é preciso modelar a contribuição do movimento das rodas em X, Y e rotação (figura 3.5.3.5). O movimento em Y será nulo, uma vez que é perpendicular a direção de rolagem das rodas e os efeitos de escorregamentos estão sendo desconsiderados.

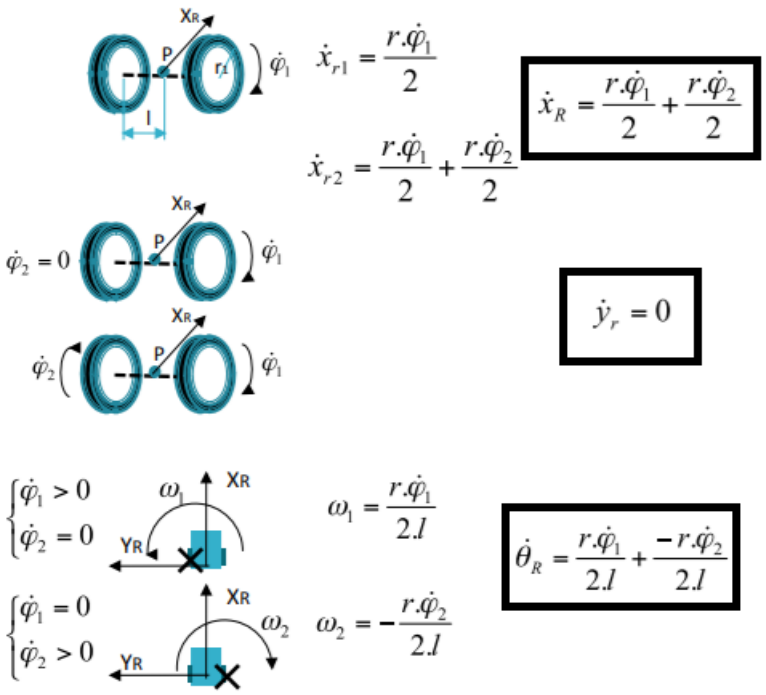


Figura 3.5.3.5 – Cinemática de posicionamento da cadeira. (adaptado de [26])

Sendo *r* o raio da roda da cadeira, *l* metade da distância entre as rodas, φ_1 e φ_2 os deslocamentos angulares das rodas direita e esquerda, respectivamente.

Para evitar colisões durante a movimentação automática, foram instalados dois sensores ultrassom modelo HC – SR04 (figura 3.5.3.6) para monitorar a distância dos objetos a frente da cadeira.

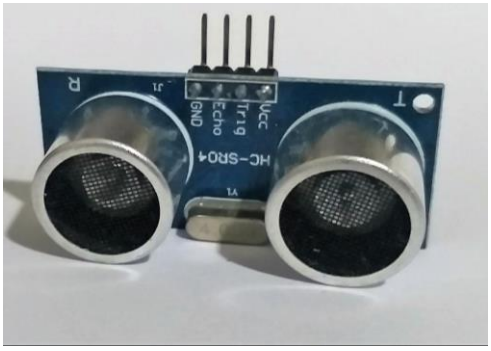


Figura 3.5.3.6 _ Sensor de Ultrassom

3.5.4 SOFTWARE

Para a interface visual implementada no computador para exibir o ambiente mapeado foi utilizado o Processing [27], um software voltado a prototipagem e visualização de dados. Seu ambiente de desenvolvimento inclui um editor de texto, um compilador e uma janela de exibição.

O primeiro passo antes de iniciar a criação da interface propriamente foi necessário elaborar um ambiente fictício em que a locomoção automática aconteceria, senso assim foi utilizada uma plataforma online chamada Visual Paradigm [28], que possui ferramentas voltadas a elaboração de plantas baixa, para a criação do ambiente idealizado, o qual se assemelha a um pequeno apartamento e é mostrado na figura 3.5.4.1. Os pontos marcados como C1, C2, S1, S2, S3, B1, Q1 e Q2 foram estabelecidos a fim de serem utilizados para a criação da trajetória de locomoção automática entre os cômodos.

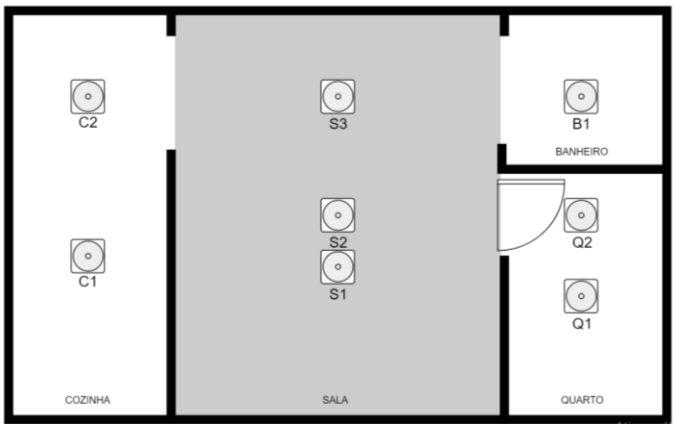


Figura 3.5.4.1 _ Ambiente fictício de locomoção automática

Além do ambiente, o usuário terá disponível na tela dois botões, um chamado “PARAR”, que interrompe qualquer movimentação que esteja ocorrendo, e outro chamado “MODO MANUAL”, que dá ao usuário o controle da movimentação da cadeira diretamente pelo joystick, sendo dispostos como mostra a figura 3.5.4.2.

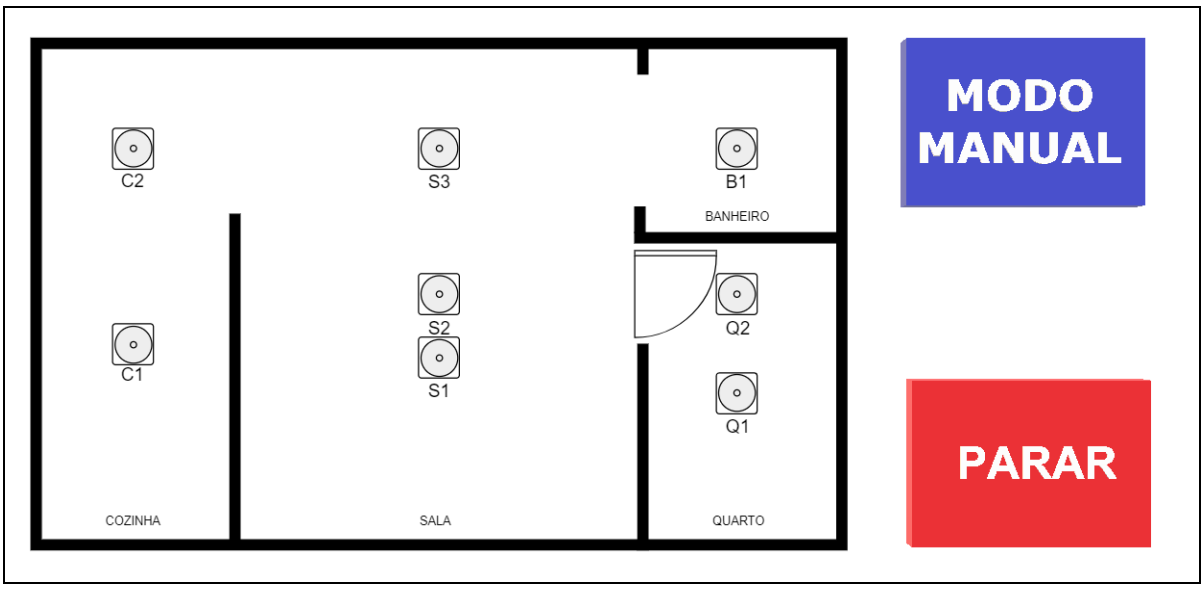


Figura 3.5.4.2 _ Tela exibida na interface

A interface visual descrita acima envia 6 possíveis comandos para o microcontrolador, 4 deles são referentes aos cômodos de destino, um de parada e um de ativação do modo manual de controle. A comunicação entre a interface e o microcontrolador é feita por meio do barramento serial estabelecido por meio de um cabo de dados USB conectado ao notebook.

CAPÍTULO 4 – IMPLEMENTAÇÃO

Após a fase de projeto, a implementação nos permite validar tudo o que foi proposto anteriormente e é nisto que está pautado este capítulo.

4.1 POTÊNCIA

O primeiro circuito a ser implementado foi o circuito de ponte H mostrado na figura 3.3.2, de forma a verificar se seu funcionamento correspondia ao esperado e se a integridade dos motores foi mantida após muito tempo parados.

Posteriormente foi adquirido um módulo Driver Ponte H, por ser mais robusto para alta potência e possuir diversos mecanismos de proteção, como mostrado na tabela 3.3.2, evitando problemas futuros ao sistema em casos de mau funcionamento. Com isso, os dois módulos de ponte H (figura 4.1.1), um para cada motor, foram implementados.

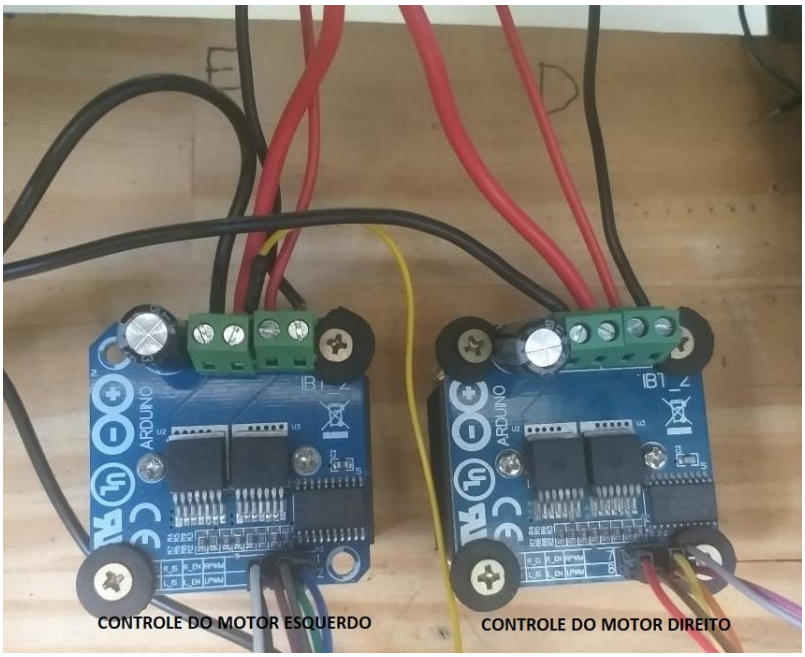


Figura 4.1.1 _ Módulo driver ponte H implementado

Como explicado na seção 3.3, o circuito de isolamento mostrado na figura 3.3.5 foi montado a fim de proteger a parte de baixa potência do sistema. A figura 4.1.2 mostra o circuito montado.

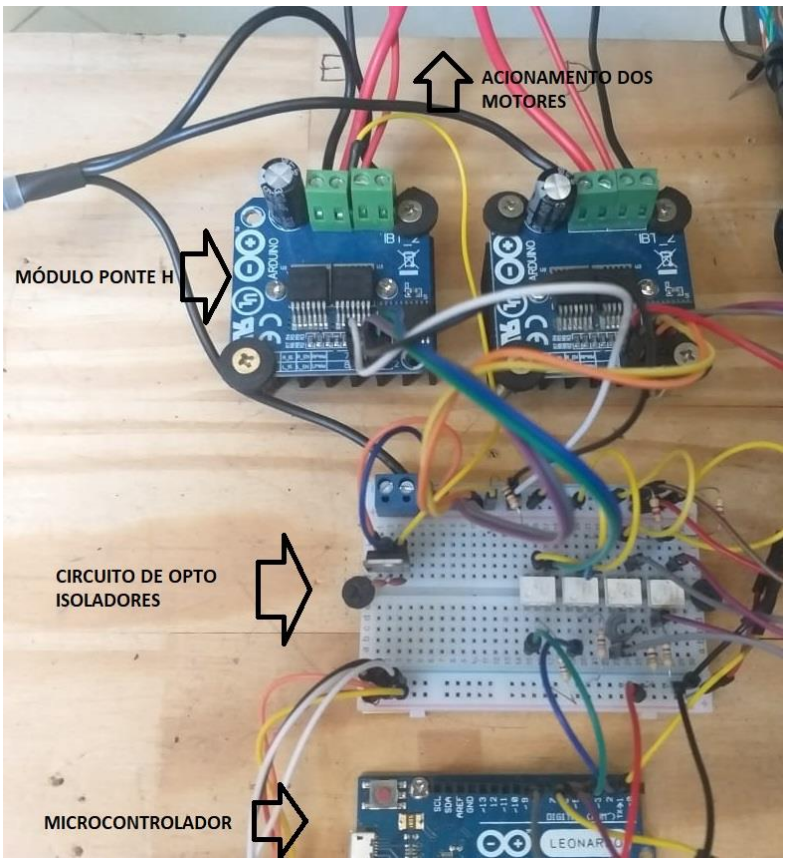


Figura 4.1.2 _ Circuito de proteção com opto isoladores

4.2 INTERFACE

A primeira forma de comando implementada foi o joystick, sendo fixado ao apoio de braço, local onde os modelos mais comuns de cadeira de rodas motorizada ficam. Em seguida foi fabricado um controle de 4 botões, utilizando uma tampa de tinta spray e botões com mola (figura 4.2.1), e fixado no outro apoio de braço de forma que fosse possível fazer testes com ambos os comandos sem a necessidade de qualquer mudança na estrutura (figura 4.2.2).

Figura 4.2.1 _ Controle por botões fabricado

Figura 4.2.2 _ Cadeira com joystick e botões implementados

4.3 ESTRUTURA

A partir do que foi especificado, o suporte para o joystick foi fabricado a partir de canos de ¾ de polegada, com uma articulação que permite que o usuário embarque e desembarque facilmente da cadeira (figura 4.3.1).

Figura 4.3.1 _ Cadeira com joystick de queixo

A mesa que foi utilizada para suportar o notebook foi facilmente adaptada à cadeira, pois o furo parafuso que fixa o apoio de braço serviu perfeitamente para posicionar a mesa, bastando apenas um parafuso longo para substituir o que fixava o apoio de braço (Figura 4.3.2).

Figura 4.3.2 _ Mesa suporte para notebook instalada.

4.4 HARDWARE

Os imãs do encoder foram posicionados na polia de alumínio de forma equidistante utilizando fita dupla face 3M, o resultado é mostrado na figura 4.4.1.

Figura 4.4.1 _ Polia do motor com imãs

O sensor de efeito HALL foi fixado de acordo com o estabelecido na figura 3.5.3.2. Durante o processo de montagem, percebeu-se a necessidade de aproximar o sensor da polia com imãs, por isso foi fabricado um pequeno suporte. O sensor instalado é mostrado na figura 4.4.2.

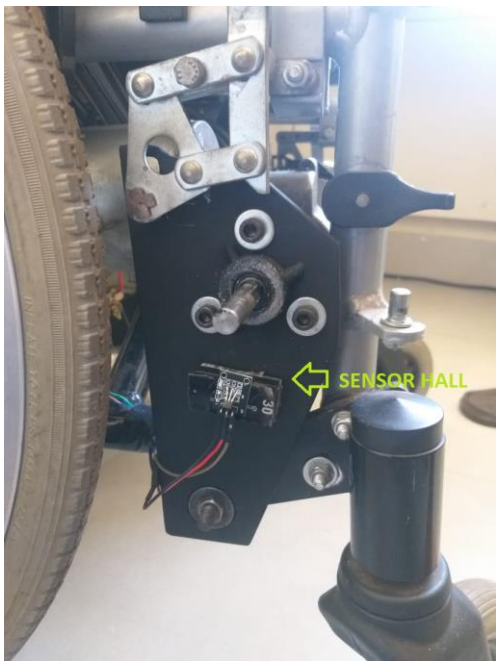


Figura 4.4.2 – Sensor de efeito hall instalado.

Na frente da cadeira foram instalados dois sensores ultrassônicos (Figura 4.4.3)

Figura 4.4.3 _ Sensores ultrassônicos instalados

4.5 SOFTWARE

Foram implementados dois códigos: um para o microcontrolador e outro para a interface visual.

O código da interface visual (apêndice 3) é executado no notebook e sua funcionalidade é exibir a imagem 3.5.4.2, contendo o ambiente mapeado e dois botões, e enviar, via barramento serial-USB, o comando selecionado na interface.

O código do microcontrolador está disponível no apêndice 2 e segue o fluxograma mostrado na figura 4.5.1.

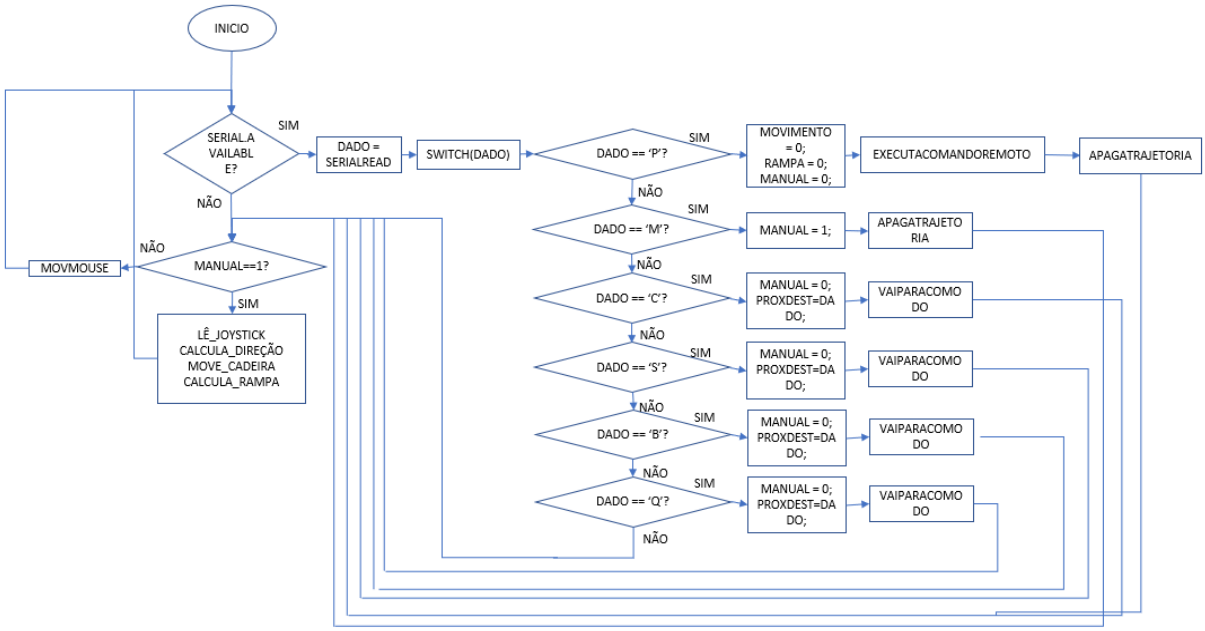


Figura 4.5.1 _ Fluxograma principal do código.

O funcionamento do sistema consiste em dois modos de funcionamento principais: manual e automático.

No modo manual, há cinco possíveis movimentações (para frente, para trás, giro para esquerda, giro para direita e parado) definidas pela posição do joystick e executadas de forma proporcional, ou seja, a velocidade dos motores é proporcional ao grau de deflexão do joystick. Neste modo de condução, o terceiro eixo do joystick, quando acionado, troca o modo de condução atual para o modo automático. Além disso, durante toda a movimentação, os encoders acoplados as rodas registram o deslocamento realizado pelas rodas e atualizam a posição atual da cadeira no ambiente.

No modo automático o sistema permite que o joystick o controle do mouse do computador, possibilitando que o usuário selecione o destino desejado na interface, isso só é possível porque o microcontrolador utilizado, Arduino Leonardo, é baseado no processador ATmega32u4 que trata da comunicação USB diretamente, sem a necessidade de um processador secundário para isso, como é o caso do Arduino UNO. Para fazer a seleção com o mouse foi utilizado o terceiro eixo do joystick. Quando o usuário seleciona um destino na interface, é enviado um dado via barramento serial sinalizando o destino

desejado, em seguida o microcontrolador verifica que há um dado disponível e guarda essa informação para interpretá-la.

Se o dado recebido seja o comado “PARAR”, o sistema interrompe a movimentação atual e apaga a trajetória armazenada. Caso seja o comando “Modo Manual”, o sistema muda o modo de condução para manual. Caso o dado seja um cômodo de destino desejado, o sistema salva esta informação e chama a função “vaiParaComodo”, representada no fluxograma da figura 4.5.2, que basicamente apaga a trajetória anterior, caso exista, verifica o cômodo atual em que se encontra, cria a trajetória, que basicamente é uma lista de pontos ordenados que devem ser alcançados para se alcançar o destino desejado, e então percorre a trajetória criada. Para descobrir o cômodo atual basicamente é utilizada a posição atual comparada as limitações dos cômodos, que foram previamente estabelecidos. São ao todo 12 possibilidades de trajetória dentro do ambiente fictício elaborado, tendo o cômodo atual e o destino desejado, é então possível estabelecer a trajetória a ser percorrida.

Figura 4.5.2 – Fluxograma da função “vaiParaComodo”.

O próximo passo é percorrer a trajetória, para isso cada trecho é calculado e executado separadamente da seguinte forma: a partir da posição atual e do próximo ponto da trajetória é calculada a direção e distância entre eles, então é executado um primeiro movimento de giro até que a direção da cadeira atinja a calculada, com a cadeira direcionada corretamente ela avança até alcançar o ponto alvo e então esse processo se repete para todos os pontos da trajetória (figura 4.5.3). Durante esse processo, o barramento serial é constantemente verificado a fim de possibilitar a parada caso seja solicitada via interface. A cadeira deve começar em um ponto e direção determinados, que servem como o referencial inicial.

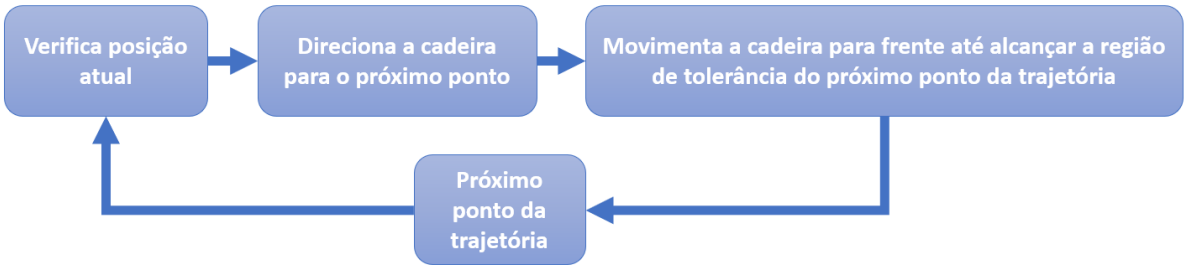


Figura 4.5.3 – Fluxograma da função de percorrer trajetória.

CAPÍTULO 5 – TESTES E RESULTADOS

5.1 POTÊNCIA

O primeiro teste realizado foi o de acionamento do motor sem carga com o circuito de ponte H da figura 3.3.2. Além de verificar o qualitativamente se o circuito gerava a rotação em ambos os sentidos possíveis, o teste também consistiu em monitorar a tensão e a corrente fornecidas ao motor durante a partida e em regime constante.

Para realização do teste foram utilizados dois multímetros comuns, um posicionado em paralelo com a bateria funcionando como voltímetro para monitorar a tensão da bateria e outro posicionado em série com polo positivo da bateria, a fim de medir a corrente utilizada pelo motor. O esquema abaixo (figura 5.1.1) ilustra como foi feita a montagem dos medidores.

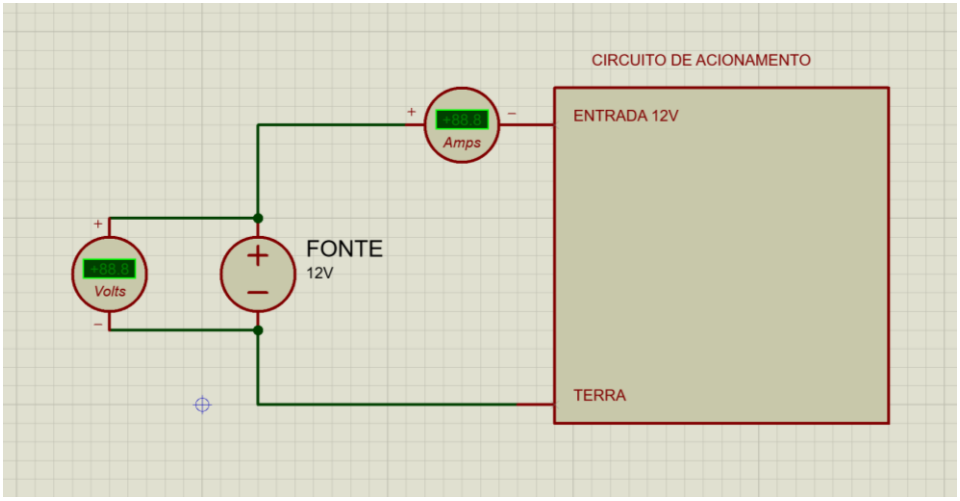


Figura 5.1.1 – Esquema de montagem dos medidores para teste

Por se tratar de medidores simples, não foi possível registrar com certeza o valor máximo de corrente durante a partida do motor, o que seria possível caso possuísse uma função chamada “*max*” que registra o maior valor lido desde que foi ativada, entretando foi registrado o maior valor visto. As leituras foram feitas durante 1 minuto para cada motor e estão representadas na tabela 5.1.1.

Tabela 5.1.1 – Teste de acionamento dos motores

Motor	Direito	Esquerdo
Máxima corrente de partida (A)	9,21	9,09
Corrente em regime permanente (A)	3.02	2,98
Tensão em vazio (V)	12,70	12,63
Menor tensão de partida (V)	12,58	12,52
Tensão em regime permanente (V)	12,61	12,54

Como registrado na tabela, há uma pequena diferença de consumo de energia entre os dois motores, porém isso não é um problema de fato, mas sim uma consequência do gasto energético feito pelo motor direito testado primeiro à bateria utilizada.

5.2 CONTROLE

As implementações de controle foram testadas alternadamente, ou seja, cada repetição do teste foi feita primeiro com o joystick e logo em seguida com o controle de botões, sempre com o mesmo usuário. O código mostrado no apêndice 1 foi implementado de forma a permitir que tanto os comandos feitos pelo joystick quanto pelos botões estivessem disponíveis simultaneamente para facilitar os testes de desempenho. Posteriormente à instalação do suporte de queixo, os mesmos testes foram feitos a fim de avaliar a diferença de esforço necessária para realizar a mesma tarefa, só que para o caso de uma pessoa tetraplégica.

Como formar de comparação, foram elaborados 3 testes: Teste de aproximação de objeto, teste de estacionar e teste de Slalom.

Como métrica de desempenho para os testes, foi cronometrado o tempo necessário para realizar o percurso, pois este está diretamente ligado a quantidade de acionamentos necessários para realizar tal manobra, ou seja, a manobrabilidade do método.

Teste de aproximação de objeto: Este primeiro teste consiste em posicionar a cadeira de rodas a uma distância 3 metros de um objeto e se aproximar dele o máximo possível (figura 5.2.1), até que se esteja dentro de uma região de tolerância estabelecida em 5 mm.

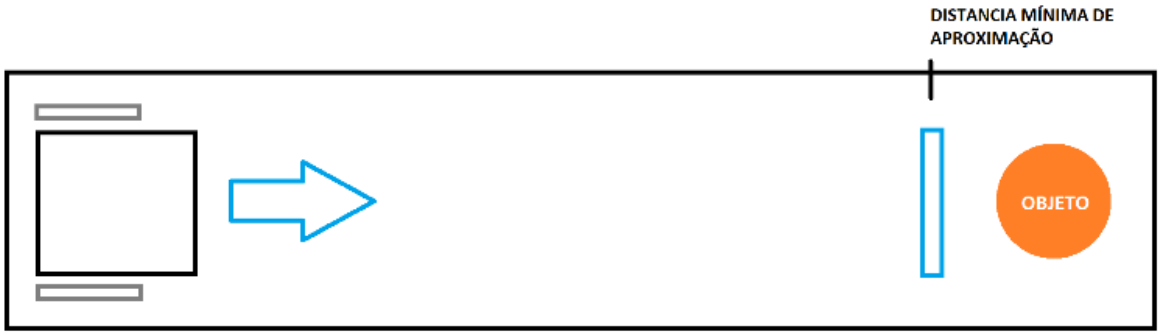


Figura 5.2.1 _ Representação do teste de aproximação

O teste foi feito com o auxílio de uma trena para que fosse possível medir distância até o objeto e verificar quando a tolerância fosse atendida. A tabela 5.2.1 mostra as medições de tempo para os métodos de acionamento proporcional (Joystick) e ON/OFF com rampa (Botão).

Tabela 5.2.1_Desempenho do teste de aproximação.

Nº Teste	Tempo (segundos)	
	Joystick	Botão
1	10,44	12,28
2	9,04	7,82
3	7,78	8,27
4	6,02	6,78
5	5,51	6,37
6	6,16	5,00
7	4,00	4,50
8	4,58	4,00
9	5,17	4,81
10	4,92	5,05
Média	6,36	6,49

Tendo a tabela 5.2.1 como base, foi feito o gráfico de desempenho no teste de aproximação (figura 5.2.2).

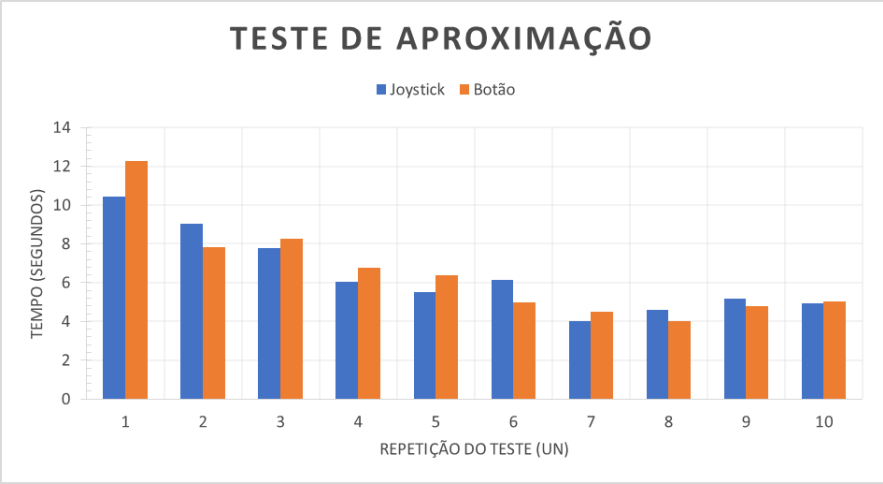


Figura 5.2.2_Gráfico de desempenho do teste de aproximação

Deste teste foi possível observar que a performance o método ON/OFF com rampa apresenta na média um desempenho ligeiramente pior devido ao seu período de aceleração (rampa), mas isso não é algo muito significativo para o usuário visto que elimina o desconforto dos trancos no acionamento dos motores. Apesar de, na média, existir essa pequena diferença de desempenho, o gráfico (figura 5.2.2) mostra que os métodos para esse teste se equivalem em desempenho, pois o melhor desempenho alterna entre os dois métodos no decorrer das repetições. É possível notar uma curva de aprendizado característica por parte do usuário, o que justifica a escolha por fazer os testes alternando entre o joystick e os botões, caso contrário o aprendizado do usuário otimizaria os testes feitos por último.

Teste de estacionar: Este teste parte dos mesmos princípios do teste de aproximação acrescido de uma manobra de 90º durante o percurso (figura 5.2.3). A situação simulada se assemelha a manobra feita por cadeirantes nos principais meios de transporte coletivo, como ônibus e metrô, a fim de se posicionar nos locais apropriados à cadeira de rodas.

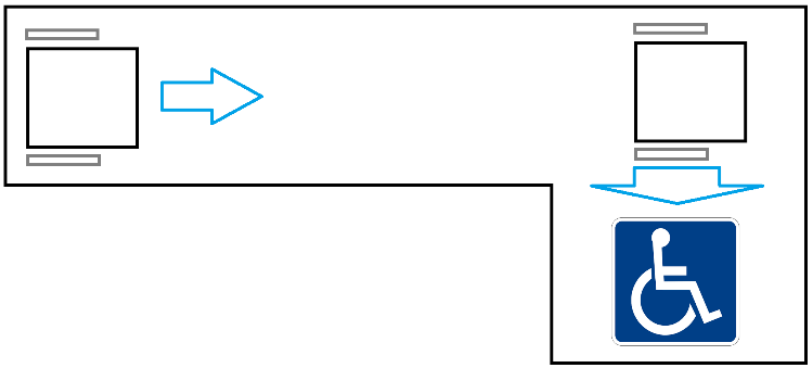


Figura 5.2.3 _ Representação do teste de estacionar

A tabela 5.2.2 contém as medições de tempo feitas durante o teste.

Tabela 5.2.2 _ Desempenho do teste de estacionar.

Nª do teste	Tempo (segundos)	
	Joystick	Botão
1	9,55	8,95
2	8,36	9,26
3	6,55	8,19
4	8,00	10,60
5	6,85	6,01
6	7,84	8,63
7	7,08	8,37
8	8,22	7,94
9	6,87	7,29
10	7,16	9,04
Média	7,81	8,43

A partir da tabela 5.2.2 foi feito o gráfico mostrado na figura 5.2.4.

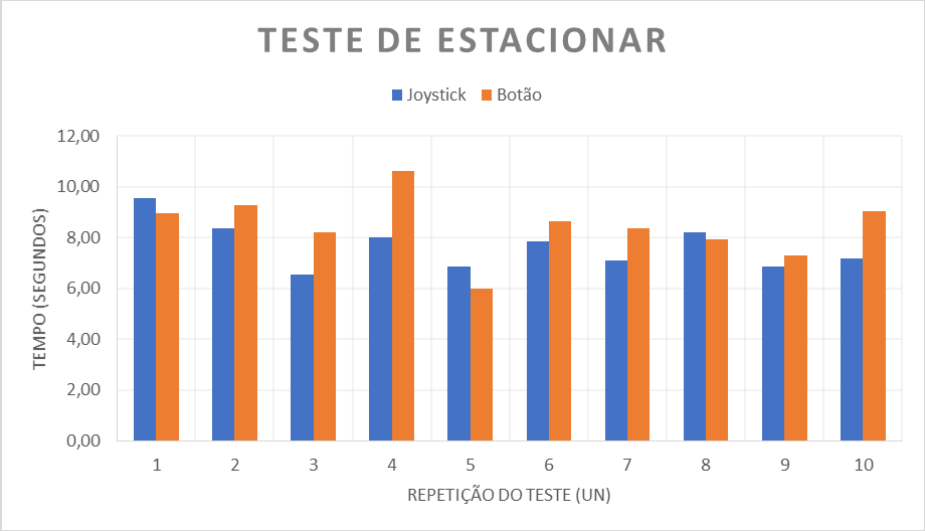


Figura 5.2.4 _ Gráfico de desempenho do teste de estacionar

Neste teste o desempenho entre joystick e controle por botões se assemelha, porém todas as mudanças de movimento do controle por botões são retardadas pela rampa de aceleração, o que nos casos em que foram necessárias várias manobras gerou uma considerável diferença, como, por exemplo, a quarta repetição. Uma vez que o teste de aproximação já havia sido feito, a curva de aprendizagem neste caso já não se mostra tão evidente quanto no teste anterior, além disso, a posição em que se escolhe realizar o giro é determinante para entrar mais facilmente na vaga, pois influencia na quantidade de correções de movimento necessárias, o que gera uma variação inconstante entre as repetições.

Teste de Slalom: Este teste foi escolhido buscando evidenciar a característica de manobrabilidade de cada modo de acionamento, visto que para realizar o percurso é necessário muitas manobras. Diferentemente dos testes anteriores, não foi utilizada uma trena, apenas demarcado no chão a posição de início e fim do percurso, bem como a área limite de manobra. A figura 5.2.5 mostra uma representação do teste realizado.

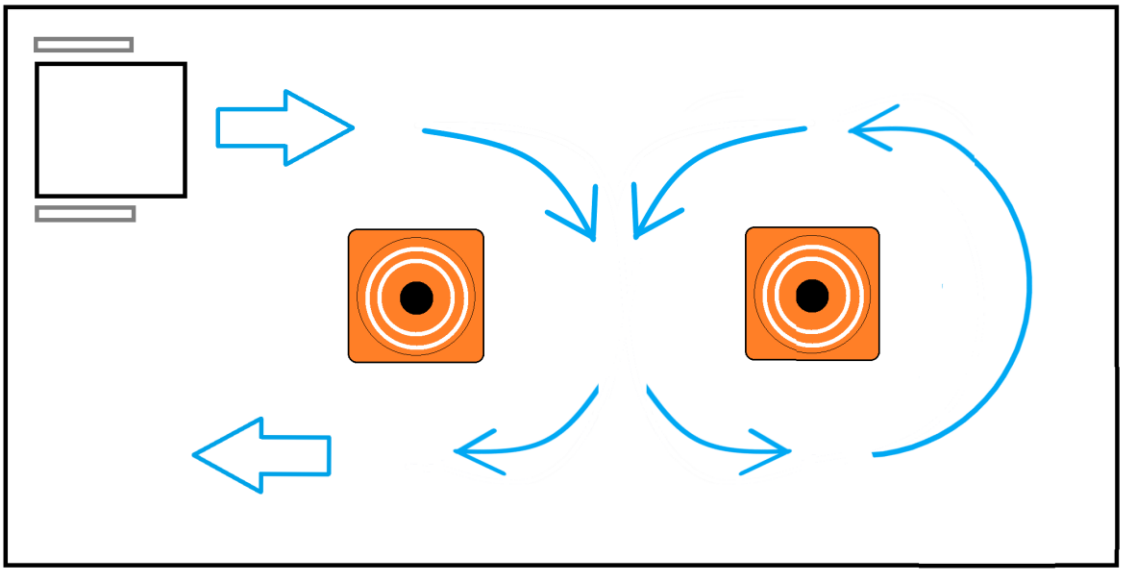


Figura 5.2.5 _ Representação do teste de Slalom

As medições de tempo realizadas no teste estão registradas na tabela 5.2.3.

Tabela 5.2.3 _ Desempenho do teste de slalom.

Nª do teste	Tempo (segundos)	
	Joystick	Botão
1	33,55	33,70
2	28,72	35,24
3	30,67	34,90
4	35,16	36,52
5	31,60	33,85
6	26,91	32,66
7	28,30	35,49
8	27,42	31,42
9	32,83	32,20
10	29,05	33,02
Média	30,42	33,90

A partir dos dados registrados na tabela 5.2.3, foi feito o gráfico mostrado na figura 5.2.6.

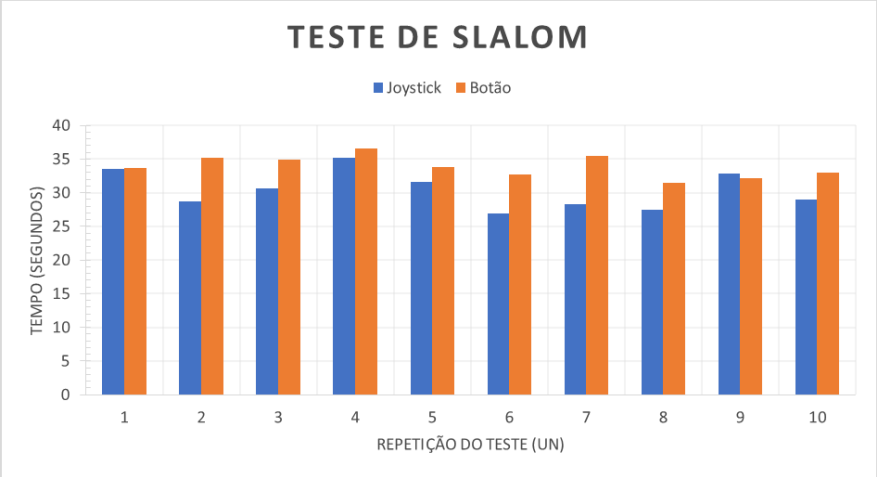


Figura 5.2.6 _ Gráfico de desempenho do teste de Slalom

Neste teste o desempenho do joystick se sobressai em relação ao controle por botões, o motivo para isso é o mesmo comentado no teste anterior: a rampa de aceleração, esse mecanismo utilizado para suavizar o acionamento gera um atraso pequeno a cada manobra quando comparamos com o joystick, porém como o percurso exige muitas manobras para completa-lo, esse atraso de acumula e resulta em uma diferença de 3,48 segundos na média para esse teste.

Partindo para a interface para tetraplégicos, o joystick de queixo foi montado e repetido os mesmos três testes anteriores. Para cada um foi adicionado à tabela, previamente feita, uma coluna extra para registrar as novas medições de tempo, com isso o gráfico de desempenho feito também os dados registrados para as três formas de comando.

Teste de aproximação de objeto:

Tabela 5.2.4_Desempenho do teste de Aproximação com 3 formas de comando.

Nº Teste	Tempo (segundos)		
	Joystick	Botão	Joystick de queixo
1	10,44	12,28	17,75
2	9,04	7,82	18,77
3	7,78	8,27	15,94
4	6,02	6,78	10,40
5	5,51	6,37	21,86
6	6,16	5,00	14,68
7	4,00	4,50	18,93
8	4,58	4,00	17,22
9	5,17	4,81	16,74
10	4,92	5,05	18,20
Média	6,36	6,49	17,05

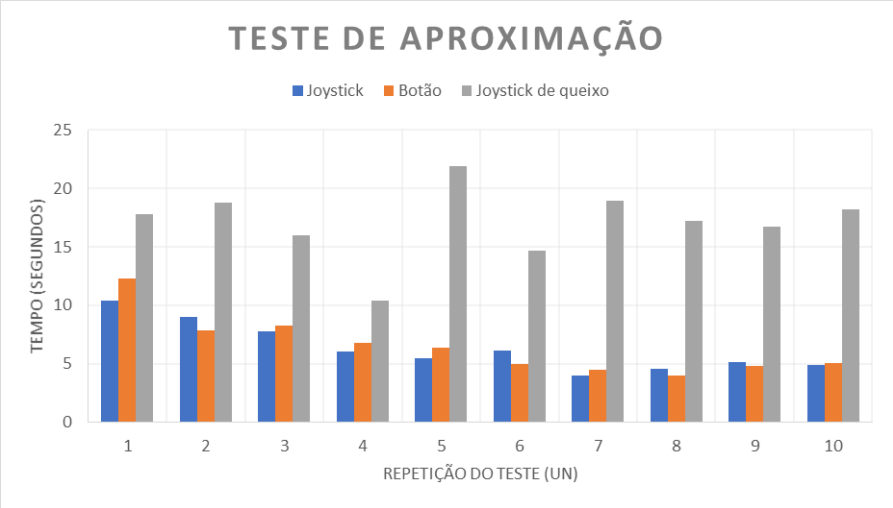


Figura 5.2.7_Gráfico de desempenho do teste de aproximação com 3 formas de comando

Neste teste com o joystick de queixo, foi possível sentir inicialmente certa dificuldade em controlar a cadeira com o queixo, uma vez que o usuário não estava habituado a esta forma de controle, isso resultou em uma grande variação de desempenho entre as repetições do teste.

Teste de estacionar:

Tabela 5.2.5 _ Desempenho do teste de estacionar com 3 formas de comando.

Nª do teste	Tempo (segundos)		
	Joystick	Botão	Joystick de queixo
1	9,55	8,95	17,54
2	8,36	9,26	18,30
3	6,55	8,19	16,75
4	8,00	10,60	18,33
5	6,85	6,01	17,94
6	7,84	8,63	19,31
7	7,08	8,37	17,78
8	8,22	7,94	16,34
9	6,87	7,29	16,96
10	7,16	9,04	17,01
Média	7,81	8,43	17,63

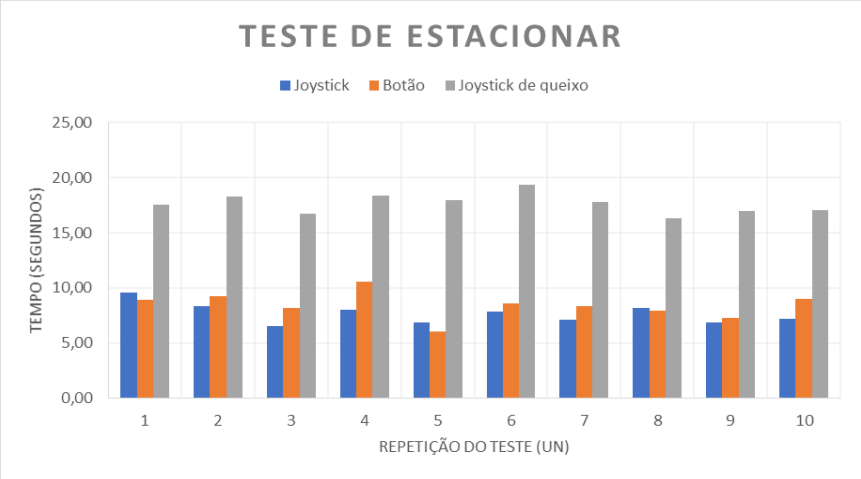


Figura 5.2.8 _ Gráfico de desempenho do teste de estacionar com 3 formas de comando.

Neste segundo teste com o joystick de queixo, o usuário mostrou um melhor domínio sobre a forma de comando, o que resultou em um desempenho mais constante entre as repetições.

Teste de Slalom:

Tabela 5.2.6 _ Desempenho do teste de slalom com 3 formas de comando.

Nª do teste	Tempo (segundos)		
	Joystick	Botão	Joystick de queixo
1	33,55	33,70	44,51
2	28,72	35,24	43,86
3	30,67	34,90	47,12
4	35,16	36,52	44,63
5	31,60	33,85	42,98
6	26,91	32,66	43,38
7	28,30	35,49	52,14
8	27,42	31,42	41,25
9	32,83	32,20	42,06
10	29,05	33,02	40,83
Média	30,42	33,90	44,28

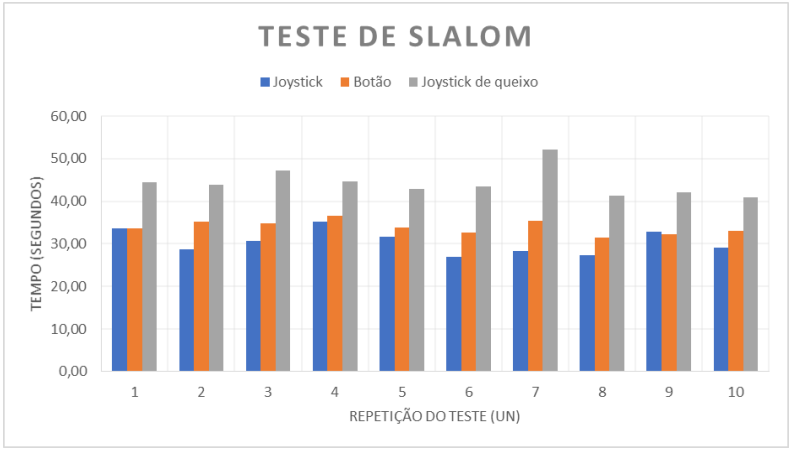


Figura 5.2.9 _ Gráfico de desempenho do teste de slalom com 3 formas de comando.

Durante os testes com o joystick de queixo, foi possível notar um certo desconforto durante o início de cada acionamento pois em acelerações rápidas, a princípio permitidas para o joystick, a inércia da cabeça gerava comandos indesejados e trancos, com isso optou-se por implementar uma rampa de aceleração para o acionamento com o joystick também, reduzindo esses problemas. Além disso, durante as paradas, foi possível sentir um certo incomodo também devido a essa inercia da cabeça e tronco, o que não era muito evidente nos testes anteriores pois o usuário fazia o controle com as mãos com os braços apoiados na cadeira, devido a isso optou-se por reduzir a potência cedida aos motores, diminuindo a velocidade e que a cadeira se movimenta.

5.3 SISTEMA SEMI AUTOMÁTICO

Os testes do sistema semi automático, foram realizados não somente com o usuário padrão dos demais testes, mas também com o auxílio de uma voluntária (figura 5.3.1), a fim de coletar suas impressões sobre a experiência de uso do sistema.

Figura 5.3.1 – Sistema completo em testes com voluntária

Foi reproduzido o espaço fictício mostrado na figura 5.3.2, marcando com fita no piso as paredes que limitam os cômodos, bem como os pontos de trajetória (figura 5.3.3).

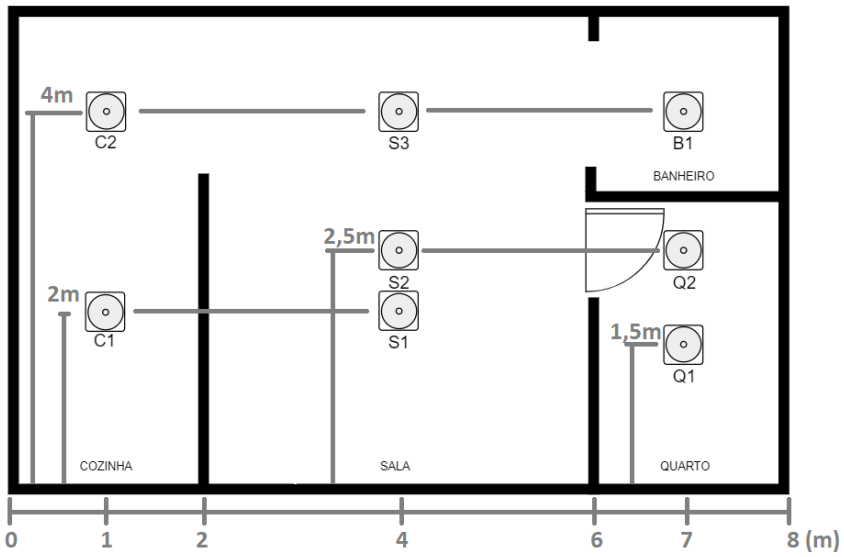


Figura 5.3.2 – Ambiente fictício cotado

Figura 5.3.3 – Ambiente reproduzido para testes do sistema semi automático.

Uma vez que durante os testes dos motores individualmente eles se comportaram de forma muito próxima, foi considerado que os movimentos das rodas ocorreria de forma simétrica desde que a potência enviada aos motores fosse a mesma, por isso, inicialmente, foi utilizado apenas o encoder de uma roda.

O primeiro teste foi realizado com a cadeira elevada, de forma a validar o código implementado, verificando se a movimentação das rodas estava de acordo com a trajetória solicitada. Após pequenos ajustes, o sistema mostrou ser confiável para os testes no ambiente demarcado.

A cadeira foi posicionada no ponto de partida conforme o estabelecido em código, então o usuário se deslocou entre os cómodos em modo manual. Em determinado momento foi solicitado que a cadeira se deslocasse para a outro cômodo, então notou-se que as rodas estavam girando com velocidades diferentes, principalmente para mudar de direção. Nos testes relatados na seção 5.2 esse comportamento não foi notado, uma vez que não

havia a necessidade de ser exatamente simétrico o deslocamento das rodas, pois o usuário controlando o acionamento tem a percepção de quando o movimento deve ser interrompido. Já no caso do sistema projetado, esse comportamento gera um erro de posicionamento, por isso, foi instalado o segundo encoder a fim de monitorar o deslocamento de ambas as rodas, de forma a realizar o correto rastreamento da posição da cadeira no ambiente e possibilizar uma correção desta diferença deslocamento entre as rodas.

A forma escolhida para atuar sobre a o deslocamento desigual das rodas foi o de reduzir a potência cedida à roda que se deslocar mais até que essa diferença desapareça. Essa estratégia resultou em resultados excelentes para deslocamento em linha reta. Para verificar a acurácia da medição, uma linha de 5 metros, demarcada no chão com fita, foi percorrida de frente e depois para trás, no primeiro caso a medição calculada pelo sistema foi precisa e a cadeira praticamente não desviou da linha marcada durante o trajeto. Quando o percurso foi feito andando para trás, a cadeira se desviou cerca de 60 centímetros da linha marcada ao final dos 5 metros, embora seja um desvio grande, foi considerado que essa manobra de percorrer grandes distancias dessa forma dificilmente será feita pelo usuário, uma vez que não terá visão do caminho que estará percorrendo. Para o giro da cadeira foi possível notar que o sistema de correção age bem para grandes amplitudes, pois sua baixa resolução, cerca de 6º, gera poucas leituras em pequenas amplitudes de giro e isso faz com que a correção não atue por tempo suficiente.

Outro comportamento indesejado que ocorreu foi a interferência do apoio de pés na movimentação das rodas dianteiras, o que ocorria em movimentos para trás e de giro. Devido às novas condições de potência reduzida, esse comportamento fazia com que a velocidade do movimento diminuísse e, em alguns casos, interrompesse o movimento. Contudo, os ajustes de altura existentes no apoio não foram suficientes para resolver o problema, por isso optou-se por removê-los para os testes do sistema.

Tendo em mente as considerações feitas acima, novamente foi realizado o teste no ambiente demarcado. Desta vez, partindo do ponto inicial, o sistema conseguiu de deslocar na trajetória solicitada até que o ponto central do cômodo desejado. Considerando que a resolução angular proporcionada pelos encoders, foi estabelecido um critério para considerar um ponto como alcançado uma área de tolerância de 20 cm ao seu redor.

CAPÍTULO 6 – CONCLUSÕES

Inicialmente, a busca por soluções já desenvolvidas para o problema de mobilidade de tetraplégicos foi a principal forma de aprendizagem acerca das dificuldades enfrentadas por essas pessoas. Para a implementação do sistema foi necessário desenvolver o sistema de acionamento dos motores, primeiro foi projetado e implementado um circuito ponte H com transistores MOSFET, posteriormente o sistema foi substituído por um módulo mais robusto, a fim de tornar o sistema mais confiável e seguro. A fim de adotar a melhor forma de acionamento, foram implementados o controle proporcional com o joystick e o controle por botões com rampa e em seguida realizados testes para comparar ambos os métodos.

A partir dos indicativos encontrados nos testes foi possível perceber que o joystick proporcional permitiu um melhor desempenho na condução da cadeira. Então foi fabricado um suporte para a adaptação do joystick para o controle com o queixo, em seguida foram repetidos os testes de percurso.

Para reduzir o desconforto que foi observado durante estes últimos testes, percebeu-se a necessidade de implementar também uma rampa de aceleração para o acionamento com o joystick e por reduzir a potência cedida aos motores, diminuindo a velocidade que a cadeira se movimenta.

Em suma, as etapas de testes com a cadeira revelaram uma série de inconveniências que não eram evidentes até que fossem de fato vivenciadas. Situações cotidianas e simples para a maioria das pessoas, como se deslocar dentro da própria casa, podem ser uma tarefa árdua para indivíduos com tetraplegia. Com isso, foi possível projetar uma solução que agrega conforto à essas pessoas, fazendo de forma automática os trajetos mais frequentes de um usuário.

Para o rastreio do posicionamento da cadeira, foi feito o cálculo da cinemática de movimentação, bem como a implementação de encoders nas rodas tracionadas a partir de sensores de efeito Hall.

Foi estabelecido um ambiente fictício, a partir do qual foi criada uma interface de visualização para o usuário. Com o mapeamento do ambiente em mãos, foi possível estabelecer uma série de pontos chaves, os quais foram utilizados para estabelecer as trajetórias de deslocamento possíveis entre os cômodos. Em seguida, foi implementada uma estratégia locomoção automática de forma a seguir a trajetória escolhida. Para testes e validação do sistema, o ambiente foi reproduzido por marcações do chão.

Como se trata de um primeiro protótipo do sistema de locomoção automático, algumas melhorias são necessárias para tornar o seu funcionamento mais confiável e robusto. O primeiro item a ser melhorado é o joystick, embora o modelo seja de baixo custo,

possui baixa amplitude de leitura e isso prejudica o controle proporcional da potência dos motores, que é a principal vantagem deste com relação ao controle por botões.

O sistema de odometria está sujeito a escorregamentos que geram erros de posicionamento no decorrer da utilização do sistema. Tendo isso em mente, faz-se necessário a implementação de mecanismos que forneçam um posicionamento absoluto da cadeira no ambiente como forma a eliminar esses erros. Uma solução possível seria a instalação de tarjas magnéticas nos portais entre os cômodos e sensores de efeito hall nas laterais da cadeira, fazendo com que o erro de posição seja zerado ao passar com a cadeira pelo portal (figura 6.1).

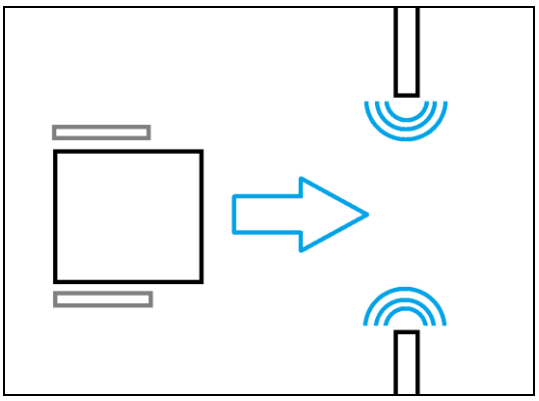


Figura 6.1 – Representação de tarjas magnéticas nos portais das portas

Mesmo que a utilização do notebook embarcado funcione bem, o uso do sistema fica limitado à duração da bateria dele, por isso seria de grande valia utilizar dispositivos mais portáteis para a implementação da interface visual, como tablet ou Raspberry pi com um display, o que dará uma maior autonomia de funcionamento.

Embora este trabalho tenha se comprometido a atender um ambiente específico criado para validar o sistema, é necessário que se crie ferramentas que permitam que o usuário forneça o ambiente de locomoção automática desejado com facilidade.

O estudo realizado permitiu avaliar as soluções implementadas explicitando suas principais características, de modo a fornecer fundamentação para a escolha da solução do joystick de queixo como interface de comando com a implementação de um sistema semi automático.

REFERÊNCIAS BIBLIOGRÁFICAS.

[1] Organização da Nações Unidas no Brasil. Disponível em https://nacoesunidas.org/acao/pessoas-com-deficiencia > Acesso em: 20 de Agosto de 2018.

[2] Governo do Brasil. Disponível em http://www.brasil.gov.br/noticias/educacao-e-ciencia/2015/09/brasil-financia-tecnologia-para-pessoas-com-deficiência > Acesso em: 20 de Agosto de 2018.

[3] The invisibility of Disability, United Nations. Disponível em http://www.un.org/disabilities/documents/sdgs/infographic_statistics_2016.pdf > Acesso em: 20 de Agosto de 2018.

[4] K.A. Dixon with Doug Kruse, Ph.D. and Carl E. Van Horn, Ph.D. Americans' Attitudes About Work, Employers and Government Work Trends. John J. Heldrich Center for Workforce Development Rutgers, The State University of New Jersey, 2003. Disponível em: <https://smlr.rutgers.edu/sites/default/files/documents/Heldrich%20disability%20survey%20report.pdf >. Acesso em 24/09/18.

[5] Diretrizes de atenção às pessoas com lesão medular, Ministério da Saúde. Disponível em http://bvsms.saude.gov.br/bvs/publicacoes/diretrizes_atencao_pessoa_lesao_medular.pdf > Acesso em: 21 de Agosto de 2018.

[6] O que é tetraplegia e como identificar, Drª Elaine Aires – Clinico Geral. Disponível em https://www.tuasaude.com/o-que-e-tetraplegia/ > Acesso em: 21 de Agosto de 2018.

[7] Tetraplegia, Ana Lucia Santana. Disponível em http://bvsms.saude.gov.br/bvs/publicacoes/diretrizes_atencao_pessoa_lesao_medular.pdf > Acesso em: 21 de Agosto de 2018.

[8] PINTO, F. F., (2016). Interfaces de controle de cadeira de rodas para pessoas com tetraplegia. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 29/2016, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 63p.

[9] IVO, Regina Marcela. Sistema de controle de cadeira de rodas motorizada para usuários portadores de tetraplegia. 2017. 90 f., il. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrônica)—Universidade de Brasília, Brasília, 2017.

[10] Mini joystick, Permobil. Disponível em https://permobilus.com/product/mini-joystick/#product-mini-joystick > Acesso em: 25 de Maio de 2019.

[11] Smile Smart Technology, Wheelchair Foot Control. Dispinível em https://www.smilesmart-tech.com/assistive-technology-products/wheelchair-controls/wheelchair-foo-control/ > Acesso em: 2 de outubro de 2018.

[12] Sean Loughran, Alternative ways to control a power wheelchair. Disponível em https://www.atandme.com/?p=950 > Acesso em: 25 de Outubro de 2018.

[13] Ferreira, Claudio Lima Lopes, (2008). INTERFACE DE SOPRO E SUCÇÃO PARA CONTROLE DE CADEIRA DE RODAS, Trabalho de Graduação, Universidade Estadual de Londrina.

[14] FONSECA SOBRINHO, A.S; FELIZARDO, K. R.; SILVA, M. A. da; OLIVEIRA, H. P.; LONE, L. P.; GERMANOVIX, W.; GAINO, R. Cadeira de rodas controlada por sopros e sucções. Semina: Ci. Exatas/ Tecnol. Londrina, v. 21, n. 4, p. 3-7, dez. 2000.

[15] ASL, Sip and Puff Head Array. Disponível em http://www.asl-inc.com/products/product_detail.php?prod=111> Acesso em: 20 de outubro de 2018.

[16] Fusco, D.A (2010). Acionamento de uma cadeira de rodas através de um acelerômetro bi-axial como inclinômetro, UFRS. Disponível em https://www.lume.ufrgs.br/bitstream/handle/10183/33065/000788108.pdf?sequence=1 > Acesso em: 13 de Setembro de 2018.

[17] Magic Mobility, Wheelchair Control Systems. Disponível em https://www.magicmobility.com.au/wheelchairs/wheelchair-options/wheelchair-controls/ > Acesso em: 13 de Outubro de 2018.

[18] “Acadêmico desenvolve circuito para condução de cadeira de rodas por movimentos oculares”, Universidade de Caxias do Sul. Disponível em https://www.ucs.br/site/noticias/academico-do-carvi-desenvolve-circuito-para-conducao-de-cadeira-de-rodas-por-movimentos-oculares/ > Acesso em: 2 de setembro de 2018.

[19] DEPANNEMAECKER, Damien T; AGRA, Alexandre L.; SALLES, Bruno A.; FRANCO, Mateus S.; JOUEN, François, AMORIM; Henrique A., FABER, Jean. Sistema de navegação para cadeirantes através do rastreamento do movimento ocular (eye-tracking). In: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA, 2016, Foz do Iguaçu. Anais: CBEB. p. 1478-1481. Disponível em: <http://cbeb.org.br/pt/anais-e-certificados >. Acesso em 13 de setembro de 2018.

[20] Albrecht, Bruno Landau (2010). Controle de uma cadeira de rodas motorizada através de eletromiografia em uma plataforma embarcada, Trabalho de conclusão de curso UFRS. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/27977/000767670.pdf?sequence=1> Acesso em : 25 de agosto de 2018.

[21] Ghovanloo, Maysam (2014). Tongue-Driven Wheelchair Out-Maneuvers the Competition, Georgia Institute of Tecnology. Dispinível em: https://www.nibib.nih.gov/news-events/newsroom/tongue-driven-wheelchair-out-maneuvers-competition > Acesso em: 13 de setembro de 2018.

[22] Aline K. Borges , Jociane F. L. Buriola , Juliano Eloi, Renan F. Teles, (2015). SmartChair: Cadeira de rodas controlada por voz, Relatório técnico, UTFPR. Disponível em http://paginapessoal.utfpr.edu.br/gustavobborba/if66j-s71-projetos/files/IF66J-15a_RT_SmartChair.pdf > Acesso em: 15 de setembro de 2018.

[23] Smile Smart Technology, Quha Zono Gyroscopic Mouse. Disponível em https://www.smilesmart-tech.com/assistive-technology-products/wheelchair-controls/quha-zono-gyroscopic-mouse/ > Acesso em: 2 de outubro de 2018.

[24] Módulo Driver Ponte H – BTS7960, Baú da Eletrônica (fornecedor). Disponível em http://www.baudaeletronica.com.br/driver-bts7960.html > Acesso em 20 de Janeiro de 2019.

[25] Portas de entrada e saída Arduino Leonardo, The eng projects. Disponível em https://www.theengineeringprojects.com/2018/10/introduction-to-arduino-leonardo.html > Acesso em 20 de fevereiro de 2019.

[26] Prof. Dr. Alexandre da Silva Simões, UNESP. Robótica Móvel, Cinemática dos robôs móveis. Disponível em http://www.gasi.sorocaba.unesp.br/assimoes/lectures/rm/Aula12%20-%20Cinematica%20dos%20robos%20moveis%20-%20cinematica.pdf > Acesso em 20 de maio de 2019.

[27] Processing. Disponível em https://processing.org/reference/environment/ > Acesso em 04 de Maio de 2019.

[28] Visual Paradigm. Disponível em https://online.visual-paradigm.com/pt/features/floor-plan-designer/ > Acesso em: 02 de Maio de 2019.

APÊNDICE 1

Código implementado com controle por joystick e botões simultaneamente.

```
int eixoX = 0;                //Leitura eixo X do Joystick
int eixoY = 0;                //Leitura eixo Y do Joystick
int pinoX = 0;                //Pino de Leitura eixo X do Joystick
int pinoY = 1;                //Pino de Leitura eixo Y do Joystick
int motorE_PWMEF = 3;         //Pino do transistor do motor 1
int motorE_PWMET = 5;         //Pino do transistor do motor 1
int motorD_PWMDF = 6;         //Pino do transistor do motor 1
int motorD_PWMDT = 9;         //Pino do transistor do motor 2
int botao_F = 10;             //Pino do botão F
int botao_T = 11;             //Pino do botão T
int botao_E = 12;             //Pino do botão E
int botao_D = 13;             //Pino do botão D
int PWM1 = 0;                 //Valor da modulação do motor 1 (0 - 255)
int PWM2 = 0;                 //Valor da modulação do motor 2 (0 - 255)
int F = 0;                    //Valor correspondente a deflexão do joystick para frente
int T = 0;                    //Valor correspondente a deflexão do joystick para trás
int E = 0;                    //Valor correspondente a deflexão do joystick para esquerda
int D = 0;                    //Valor correspondente a deflexão do joystick para direita
int ativa_motor1 = 0;         //Status do motor 1
int ativa_motor2 = 0;         //Status do motor 2
int i;                        //Variável contadora genérica
int offsetPWM = 105;
int maxPWM = 150;
int rampa = 0;

void setup() {
  pinMode(motorE_PWMEF, OUTPUT);
  pinMode(motorE_PWMET, OUTPUT);
  pinMode(motorD_PWMDF, OUTPUT);
  pinMode(motorD_PWMDT, OUTPUT);
  pinMode(botao_F, INPUT);
  pinMode(botao_T, INPUT);
  pinMode(botao_E, INPUT);
  pinMode(botao_D, INPUT);
  Serial.begin(9600);
}

void loop() {                 //ROTINA PRINCIPAL
```

```
eixoX = analogRead(pinoX);      //Leitura do eixo X do joystick
eixoY = analogRead(pinoY);      //Leitura do eixo Y do joystick
Serial.print(eixoX);
Serial.print(" - ");
Serial.print(eixoY);

//Cálculo do posicionamrnto do joystick
if(eixoY>550){ //Cálculando F
  F = abs(eixoY-512)/2;
  if(F>255){
    F = 255;
  }
}
else{
  F = 0;
}
if(eixoY<470){ //Cálculando T
  T = abs(eixoY-512)/2;
  if(T > 255){
    T = 255;
  }
}
else{
  T = 0;
}
if(eixoX>550){ //Cálculando E
  E = abs(eixoX-512)/2;
  if(E>255){
    E = 255;
  }
}
else{
  E = 0;
}
if(eixoX<470){ //Cálculando D
  D = abs(eixoX-512)/2;
  if(D>255){
    D = 255;
  }
}
else{
  D = 0;
}
//VERIFICANDO QUAL ESTADO DE MOVIMENTO DESEJADO
```

```
        /*     PARADO     */  if(eixoY >= 470 && eixoY <= 550 && eixoX >= 470 && eixoX <= 550 &&
!(digitalRead(botao_F)) && !(digitalRead(botao_T)) && !(digitalRead(botao_E)) && !(digitalRead(botao_D))){
                            analogWrite(motorE_PWMEF, (0));
                            analogWrite(motorE_PWMET, (0));
                            analogWrite(motorD_PWMDF, (0));
                            analogWrite(motorD_PWMDT, (0));
                           }
        /*     FRENTE     */  if(eixoY > 550 && eixoX > 470 && eixoX < 550 || (digitalRead(botao_F)) &&
!(digitalRead(botao_T)) && !(digitalRead(botao_E)) && !(digitalRead(botao_D))){
                            if(F != 0 && F < rampa){
                              analogWrite(motorE_PWMEF, (F));
                              analogWrite(motorE_PWMET, (0));
                              analogWrite(motorD_PWMDF, (F));
                              analogWrite(motorD_PWMDT, (0));
                            }
                            else{
                              analogWrite(motorE_PWMEF, (rampa));
                              analogWrite(motorE_PWMET, (0));
                              analogWrite(motorD_PWMDF, (rampa));
                              analogWrite(motorD_PWMDT, (0));
                            }

                           }
        /*     TRÁS      */  if(eixoY < 470 && eixoX > 470 && eixoX < 550 || !(digitalRead(botao_F)) &&
(digitalRead(botao_T)) && !(digitalRead(botao_E)) && !(digitalRead(botao_D))){
                            if(T != 0 && T < rampa) {
                              analogWrite(motorE_PWMEF, (0));
                              analogWrite(motorE_PWMET, (T));
                              analogWrite(motorD_PWMDF, (0));
                              analogWrite(motorD_PWMDT, (T));
                            }
                            else{
                              analogWrite(motorE_PWMEF, (0));
                              analogWrite(motorE_PWMET, (rampa));
                              analogWrite(motorD_PWMDF, (0));
                              analogWrite(motorD_PWMDT, (rampa));
                            }
                           }
        /* GIRO ESQUERDA  */  if(eixoX > 550 && eixoY > 470 && eixoY < 550 || !(digitalRead(botao_F)) &&
!(digitalRead(botao_T)) && (digitalRead(botao_E)) && !(digitalRead(botao_D))){
                            if(E != 0 && E < rampa){
                              analogWrite(motorE_PWMEF, (0));
                              analogWrite(motorE_PWMET, (E));
                              analogWrite(motorD_PWMDF, (E));
                              analogWrite(motorD_PWMDT, (0));
                            }
```

```
                    else{
                      analogWrite(motorE_PWMEF, (0));
                      analogWrite(motorE_PWMET, (rampa));
                      analogWrite(motorD_PWMDF, (rampa));
                      analogWrite(motorD_PWMDT, (0));
                    }
                  }
          /* GIRO DIREITA   */  if(eixoX < 470 && eixoY > 470 && eixoY < 550 || !(digitalRead(botao_F)) &&
!(digitalRead(botao_T)) && !(digitalRead(botao_E)) && (digitalRead(botao_D))){
                    if(D != 0 && D < rampa){
                      analogWrite(motorE_PWMEF, (D));
                      analogWrite(motorE_PWMET, (0));
                      analogWrite(motorD_PWMDF, (0));
                      analogWrite(motorD_PWMDT, (D));
                    }
                    else{
                      analogWrite(motorE_PWMEF, (rampa));
                      analogWrite(motorE_PWMET, (0));
                      analogWrite(motorD_PWMDF, (0));
                      analogWrite(motorD_PWMDT, (rampa));
                    }
                  }
                  //calcula rampa
                  if(!(digitalRead(botao_F)) && !(digitalRead(botao_T)) && !(digitalRead(botao_E)) &&
!(digitalRead(botao_D)) && !F && !T && !E && !D){
                    rampa = 0;
                  }
                  else{
                    rampa += 5;
                    if(rampa > 255){
                      rampa = 255;
                    }
                  }
                  delay(20);    //Delay em ms entre os ciclos e incremento da rampa de acionamento
                  Serial.print(" _ ");
                  Serial.print(F);
                  Serial.print(" - ");
                  Serial.print(T);
                  Serial.print(" _ ");
                  Serial.print(E);
                  Serial.print(" - ");
                  Serial.print(D);
                  Serial.print(" - ");
                  Serial.println(rampa);
                }
```

APÊNDICE 2.

Código implementado para o sistema com funcionalidade automática.

```
#include "Mouse.h"
#include "math.h"
//PORTAS UTILIZADAS DO MICROCONTROLADOR
int pinoX = 0;              //Pino de leitura analogica eixo X do Joystick
int pinoY = 1;              //Pino de leitura analogica eixo Y do Joystick
int pinoZ = 10;             //Pino de leitura analogica eixo Z do Joystick
int encoderLE = 2;          //Pino de leitura do encoder acoplado a roda esquerda
int encoderLD = 3;          //Pino de leitura do encoder acoplado a roda direita
int motorE_PWMEF = 12;      //Pino do transistor do motor 1
int motorE_PWMET = 5;       //Pino do transistor do motor 1
int motorD_PWMDF = 6;       //Pino do transistor do motor 2
int motorD_PWMDT = 9;       //Pino do transistor do motor 2
//VARIÁVEIS
int eixoX = 0;              //Leitura eixo X do Joystick
int eixoY = 0;              //Leitura eixo Y do Joystick
int F = 0;                  //Valor correspondente a deflexão do joystick para frente
int T = 0;                  //Valor correspondente a deflexão do joystick para trás
int E = 0;                  //Valor correspondente a deflexão do joystick para esquerda
int D = 0;                  //Valor correspondente a deflexão do joystick para direita
int i;                      //Variável contadora genérica
int movimento = 0;          //Flag para indicar movimento atual da cadeira : 0_parado, 1_frente, 2_frente&esq, 3_frente&dir, 4_tras, 5_tras&esq, 6_tras&dir, 7_esq e 8_dir
int rampa = 0;              //Valor da rampa de acionamento dos motores
char dado = 'P';            //Variável que contem o ultimo dado recebido via interface serial
char comodoAtual;           //Armazena o comodo em que a cadeira se encontra durante a locomoção autonoma.
char proxDest;              //Contém o comodo alvo da locomoção autonoma
float trajetoria[6][2];     //Matriz de pontos a serem alcançados durante a trajetória
int proxPonto = 0;          //Indice do proximo ponto da trajetoria a ser atingido
float Px = 4, Py = 1;       //Armazena a posição atualizada da cadeira
double teta = PI/2;         //Armazena a direção da cadeira em radianos
double tetaAlvo = 0;        //Direção desejada durante o movimento em radianos
int correcaoPWMLD = 0;      //Valor PWM que servirá para corrigir a trajetoria alterando a velocidade da roda direita
int correcaoPWMLE = 0;      //Valor PWM que servirá para corrigir a trajetoria alterando a velocidade da roda esquerda
int pulsosE = 0;            //Conta a quantidade de pulsos lidos pelo encoder do lado esquerdo
int pulsosD = 0;            //Conta a quantidade de pulsos lidos pelo encoder do lado direito
int raioRoda = 30;          //Raio da roda da cadeira em centimetros
int larguraCadeira = 60;    //Distancia entre o centro do pneu duas rodas
```

```
//CONSTANTES
int zonaMortaJoystick = 100;     //Amplitude de leitura imprecisa do joystick
int maxPWM = 130;                //Valor máximo valor de potencia permitido aos motores
double resolucaoAngulo=(PI/30); //Resolução em radianos proporcionada pelo encoder a cada pulso
float resolucaoDist = 0.0325;     //Resolução em centímetros proporcionada pelo encoder a cada pulso
float toleranciaDist = 0.3;     //Tolerancia de aproximação dos pontos da trajetoria
//FLAGS
int manual = 1;                  //Flag para sinalizar o modo de funcionamento atual do sistema (autonomo ou manual)
int movimentoConcluido = 1;         //Flag para sinalizar se o ponto desejado no modo autonomo foi atingido
int modoGiro = 0;               //Flag para sinalizar quando a cadeira estiver rodando ou não
int giroPositivo = 0;                //Flag para sinalizar se o giro que está sendo realizado incrementa o valor do teta atual
int movimentoPositivo = 0 ;     //Flag para sinalizar se o movimento é para frente
//
void setup() {
  pinMode(motorE_PWMEF, OUTPUT);
  pinMode(motorE_PWMET, OUTPUT);
  pinMode(motorD_PWMDF, OUTPUT);
  pinMode(motorD_PWMDT, OUTPUT);
  pinMode(encoderLE, INPUT);
  pinMode(encoderLD, INPUT);
  pinMode(pinoZ, INPUT);
  attachInterrupt(digitalPinToInterrupt(encoderLE), verificaPosicao, RISING);
  attachInterrupt(digitalPinToInterrupt(encoderLD), verificaPosicaoD, RISING);
  Serial.begin(9600);
  Mouse.begin();
}

void loop() {                   //ROTINA PRINCIPAL
  if(Serial.available()){
    dado = Serial.read(); //Lê da porta serial/ comunicação com a interface
    switch (dado){
      case 'P': //PARADO
        Serial.println("P - R");
        movimento = 0;
        rampa = 0;
        manual = 0;
        executaComandoRemoto();
        apagaTrajetoria();
        break;
      case 'M': //ACIONAMENTO MANUAL
        Serial.println("A - R");
```

```
      manual = 1;
      apagaTrajetoria();
      break;
    case 'C': //PARA COZINHA
      Serial.println("C - R");
      manual = 0;
      proxDest = dado;
      vaiParaComodo();
      break;
    case 'S': //PARA SALA
      Serial.println("S - R");
      manual = 0;
      proxDest = dado;
      vaiParaComodo();
      break;
    case 'B': //PARA PANHEIRO
      Serial.println("B - R");
      manual = 0;
      proxDest = dado;
      vaiParaComodo();
      break;
    case 'Q': //PARA QUARTO
      Serial.println("Q - R");
      manual = 0;
      proxDest = dado;
      vaiParaComodo();
      break;
  }
}

if(manual == 1){
    if(digitalRead(pinoZ)== LOW){
      manual = 0;
      Serial.println("Clicou");
      movimento = 0;
      executaComandoRemoto();
      delay(300);
    }
    else{
      leJoystick();
      calculaDirecao();
      movCadeira();
      calcRampa();
    }
```

```
  }
  else{
    moveMouse();
  }
}


//////////////_____________________________________FUNÇÕES_____________________________________///////
//////////////////////////////

void leJoystick(void){
  eixoX = analogRead(pinoX);      //Leitura do eixo X do joystick
  eixoY = analogRead(pinoY);      //Leitura do eixo Y do joystick
  return;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void apagaTrajetoria(void){
  for(int j=0; j<=5; j++){
    trajetoria[j][0] = -1;
    trajetoria[j][1] = -1;
  }
  return;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
void verificaPosicaoD(void){
  pulsosD++;
  //Sistema de correção
  if(pulsosE > 100 && pulsosD > 100){//Evita estourar a variável contadora de pulsos
    pulsosE = pulsosE - 50;
    pulsosD = pulsosD - 50;
  }
  if((pulsosE-pulsosD)<0){ //Roda direita ta mais rapida
    correcaoPWMLD += (3*(pulsosD-pulsosE));
  }
  else{
    if(correcaoPWMLD>0){
      correcaoPWMLD  = 0;//-= correcaoPWMLD/2;
    }
    else{
      correcaoPWMLD = 0;
    }
  }
  if((maxPWM-correcaoPWMLD)<=0){
    correcaoPWMLD = maxPWM-5;
  }
```

```
if((pulsosE-pulsosD)>0){
  correcaoPWMLE += (3*(pulsosE-pulsosD));
}
else{
  if(correcaoPWMLE>0){
    correcaoPWMLE = 0;//-= correcaoPWMLE/2;
  }
  else{
    correcaoPWMLE = 0;
  }
}
if((maxPWM-correcaoPWMLE)<=0){
  correcaoPWMLE = maxPWM-5;
}
//Atualizar direção
if(modoGiro==1){//Atualiza ângulo
  if(giroPositivo == 1){//O angulo é incrementado quando o giro é para a esquerda
    teta += (raioRoda*resolucaoAngulo)/larguraCadeira;
    Px += (resolucaoDist/2)*cos(teta);
    Py += (resolucaoDist/2)*sin(teta);
  }
  else{//Quando o giro é para a
    teta -= (raioRoda*resolucaoAngulo)/larguraCadeira;
    Px -= (resolucaoDist/2)*cos(teta);
    Py -= (resolucaoDist/2)*sin(teta);
  }
  if(teta > (2*PI)){
    teta -= (2*PI);
  }
  if(teta < 0){
    teta += (2*PI);
  }
  if(manual == 0){//Verifica se a movimentação é autonoma
    //Verificar se a direção desejada foi alcançada
    if(teta >= (tetaAlvo-(resolucaoAngulo/2)) && teta <= (tetaAlvo+(resolucaoAngulo/2))){
      movimentoConcluido = 1;
    }
  }
}
//Atualiza posição
else{
  if(movimentoPositivo == 1){//Verifica se esta andando para frente
    Px += (resolucaoDist/2)*cos(teta);
    Py += (resolucaoDist/2)*sin(teta);
```

```
      teta += (raioRoda*resolucaoAngulo)/larguraCadeira;
     }
     else{
      Px -= (resolucaoDist/2)*cos(teta);
      Py -= (resolucaoDist/2)*sin(teta);
      teta -= (raioRoda*resolucaoAngulo)/larguraCadeira;
     }
     if(Px >= (trajetoria[proxPonto][0]-toleranciaDist) && Px <= (trajetoria[proxPonto][0]+toleranciaDist)){//Verifica se a posição em X atual está dentro do esperado
      if(Py >= (trajetoria[proxPonto][1]-toleranciaDist) && Py <= (trajetoria[proxPonto][1]+toleranciaDist)){//Verifica se a posição em X atual está dentro do esperado
       movimentoConcluido = 1;
      }
     }
    }
    return;
   }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   void verificaPosicao(){
    pulsosE++;
    if(pulsosE > 100 && pulsosD > 100){//Evita estourar a variável contadora de pulsos
     pulsosE = pulsosE - 50;
     pulsosD = pulsosD - 50;
    }
    if((pulsosE-pulsosD)>0){
     correcaoPWMLE += (3*(pulsosE-pulsosD));
    }
    else{
     if(correcaoPWMLE>0){
      correcaoPWMLE = 0;//-= correcaoPWMLE/2;
     }
     else{
      correcaoPWMLE = 0;
     }
    }
    if((maxPWM-correcaoPWMLE)<=0){
     correcaoPWMLE = maxPWM-5;
    }
    if((pulsosE-pulsosD)<0){
     correcaoPWMLD += (3*(pulsosD-pulsosE));
    }
    else{
     if(correcaoPWMLD>0){
      correcaoPWMLD = 0;//-= correcaoPWMLD/2;
     }
```

```
      else{
        correcaoPWMLD = 0;
      }
    }
    if((maxPWM-correcaoPWMLD)<=0){
      correcaoPWMLD = maxPWM-5;
    }
    if(modoGiro==1){//Atualiza ângulo
      if(giroPositivo == 1){//O angulo é incrementado quando o giro é para a esquerda
        teta += (raioRoda*resolucaoAngulo)/larguraCadeira;
        Px -= (resolucaoDist/2)*cos(teta);
        Py -= (resolucaoDist/2)*sin(teta);
      }
      else{//Quando o giro é para a direita
        teta -= (raioRoda*resolucaoAngulo)/larguraCadeira;
        Px += (resolucaoDist/2)*cos(teta);
        Py += (resolucaoDist/2)*sin(teta);
      }
      if(teta > (2*PI)){
        teta -= (2*PI);
      }
      if(teta < 0){
        teta += (2*PI);
      }
      if(manual == 0){//Verifica se a movimentação é autonoma
        //Verificar se a direção desejada foi alcançada
        if(teta >= (tetaAlvo-(resolucaoAngulo/2)) && teta <= (tetaAlvo+(resolucaoAngulo/2))){
          movimentoConcluido = 1;
        }
      }
    }
    else{//Atualiza posição
      if(movimentoPositivo == 1){//Verifica se esta andando para frente
        Px += (resolucaoDist/2)*cos(teta);
        Py += (resolucaoDist/2)*sin(teta);
        teta -= (raioRoda*resolucaoAngulo)/larguraCadeira;
      }
      else{
        Px -= (resolucaoDist/2)*cos(teta);
        Py -= (resolucaoDist/2)*sin(teta);
        teta += (raioRoda*resolucaoAngulo)/larguraCadeira;
      }
      if(Px >= (trajetoria[proxPonto][0]-toleranciaDist) && Px <= (trajetoria[proxPonto][0]+toleranciaDist)){//Verifica se a posição em X atual está dentro do esperado
```

```
                    if(Py >= (trajetoria[proxPonto][1]-toleranciaDist) && Py <= (trajetoria[proxPonto][1]+toleranciaDist)){//Verifica se a posição em X atual está dentro do esperado
                      movimentoConcluido = 1;
                    }
                   }
                  }
                  //Caso de colisão ou sair da casa, para o movimento e usuário assume o controle novamente
                  if(Px<0 || Px>=8 || Py<0 || Py>=5){
                   //movimento = 0;
                   //movimentoConcluido = 1;
                   //executaComandoRemoto();
                   //apagaTrajetoria();
                   //manual = 1;
                   Serial.println("Fora da área permitida!");
                  }
                  //Serial.print(" Px = ");
                  //Serial.print(Px);
                  //Serial.print("_ Py = ");
                  //Serial.print(Py);
                  //Serial.print(" _ teta =");
                  //Serial.println((teta*360)/(2*PI));
                  return;
                 }
        ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                 void vaiParaComodo(void){
                  apagaTrajetoria();
                  Serial.println("Apagou a tragetória");
                  verificaComodoAtual();
                  Serial.print("Comodo atual: ");
                  Serial.println(comodoAtual);
                  criaTrajetoria();
                  Serial.println("Criou trajetoria");
                  percorreTrajetoria();
                  Serial.println("Chegou ao destino desejado");
                  return;
                 }
        ////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                 void verificaComodoAtual(void){
                  //verifica posição em X
                  if(Px<=2){
                   comodoAtual = 'C';
                  }
                  if(Px>2 && Px <6){
                   comodoAtual = 'S';
```

```
                    }
                    if(Px>=6){
                      if(Py>3){
                        comodoAtual = 'B';
                      }
                      if(Py<=3){
                        comodoAtual = 'Q';
                      }
                    }
                    //Coloca o ponto central do comodo na primeira posição da lista
                    switch(comodoAtual){
                      case 'C':
                        trajetoria[0][0] = 1; //Px de C1
                        trajetoria[0][1] = 2; //Py de C1
                        break;
                      case 'S':
                        trajetoria[0][0] = 4; //Px de S1
                        trajetoria[0][1] = 2; //Py de S1
                        break;
                      case 'Q':
                        trajetoria[0][0] = 7; //Px de Q1
                        trajetoria[0][1] = 1.5; //Py de Q1
                        break;
                      case 'B':
                        trajetoria[0][0] = 7; //Px de B1
                        trajetoria[0][1] = 4; //Py de B1
                        break;
                    }
                    return;
                  }
  ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
                  void criaTrajetoria(void){

                    //Verifica se o comodo desejado é o comodo atual, caso seja termina de preenche a lista de pontos com "null" e sai da função
                    if(comodoAtual == proxDest){
                      return;
                    }
                    //Verifica o comodo atual
                    switch(comodoAtual){
                      case 'C':
                        //Toda trajetoria iniciada na cozinha passa por C2, logo esse ponto já será incluido na trajetoria
                        trajetoria[1][0] = 1;   //Px de C2
                        trajetoria[1][1] = 4;   //Py de C2
```

```
  switch(proxDest){
    case 'S':
      trajetoria[2][0] = 4;   //Px de S3
      trajetoria[2][1] = 4;   //Py de S3
      trajetoria[3][0] = 4;   //Px de S1
      trajetoria[3][1] = 2;   //Py de S1
      break;
    case 'Q':
      trajetoria[2][0] = 4;   //Px de S3
      trajetoria[2][1] = 4;   //Py de S3
      trajetoria[3][0] = 4;   //Px de S2
      trajetoria[3][1] = 2.5; //Py de S2
      trajetoria[4][0] = 7;   //Px de Q2
      trajetoria[4][1] = 2.5; //Py de Q2
      trajetoria[5][0] = 7;   //Px de Q1
      trajetoria[5][1] = 1.5; //Py de Q1
      break;
    case 'B':
      trajetoria[2][0] = 7;   //Px de B1
      trajetoria[2][1] = 4;   //Py de B1
      break;
  }
  break;
case 'S':
  switch(proxDest){
    case 'C':
      trajetoria[1][0] = 4;   //Px de S3
      trajetoria[1][1] = 4;   //Py de S3
      trajetoria[2][0] = 1;   //Px de C2
      trajetoria[2][1] = 4;   //Py de C2
      trajetoria[3][0] = 1;   //Px de C1
      trajetoria[3][1] = 2;   //Py de C1
      break;
    case 'Q':
      trajetoria[1][0] = 4;   //Px de S2
      trajetoria[1][1] = 2.5; //Py de S2
      trajetoria[2][0] = 7;   //Px de Q2
      trajetoria[2][1] = 2.5; //Py de Q2
      trajetoria[3][0] = 7;   //Px de Q1
      trajetoria[3][1] = 1.5; //Py de Q1
      break;
    case 'B':
      trajetoria[1][0] = 4;   //Px de S3
      trajetoria[1][1] = 4;   //Py de S3
```

```
            trajetoria[2][0] = 7;   //Px de B1
            trajetoria[2][1] = 4;   //Py de B1
            break;
          }
          break;
        case 'Q':
          //Toda trajetoria iniciada na cozinha passa por C2, logo esse ponto já será incluido na
trajetoria
          trajetoria[1][0] = 7;   //Px de Q2
          trajetoria[1][1] = 2.5; //Py de Q2
          switch(proxDest){
            case 'S':
              trajetoria[2][0] = 4;   //Px de S2
              trajetoria[2][1] = 2.5; //Py de S2
              trajetoria[3][0] = 4;   //Px de S1
              trajetoria[3][1] = 2;   //Py de S1
              break;
            case 'B':
              trajetoria[2][0] = 4;   //Px de S2
              trajetoria[2][1] = 2.5; //Py de S2
              trajetoria[3][0] = 4;   //Px de S3
              trajetoria[3][1] = 4;   //Py de S3
              trajetoria[4][0] = 7;   //Px de B1
              trajetoria[4][1] = 4;   //Py de B1
              break;
            case 'C':
              trajetoria[2][0] = 4;   //Px de S2
              trajetoria[2][1] = 2.5; //Py de S2
              trajetoria[3][0] = 4;   //Px de S3
              trajetoria[3][1] = 4;   //Py de S3
              trajetoria[4][0] = 1;   //Px de C2
              trajetoria[4][1] = 4;   //Py de C2
              trajetoria[5][0] = 1;   //Px de C1
              trajetoria[5][1] = 2;   //Py de C1
              break;
          }
          break;
        case 'B':
          switch(proxDest){
            case 'S':
              trajetoria[1][0] = 4;   //Px de S3
              trajetoria[1][1] = 4;   //Py de S3
              trajetoria[2][0] = 4;   //Px de S1
              trajetoria[2][1] = 2;   //Py de S1
              break;
```

```
            case 'Q':
              trajetoria[1][0] = 4;   //Px de S3
              trajetoria[1][1] = 4;   //Py de S3
              trajetoria[2][0] = 4;   //Px de S2
              trajetoria[2][1] = 2.5; //Py de S2
              trajetoria[3][0] = 7;   //Px de Q2
              trajetoria[3][1] = 2.5; //Py de Q2
              trajetoria[4][0] = 7;   //Px de Q1
              trajetoria[4][1] = 1.5; //Py de Q1
              break;
            case 'C':
              trajetoria[1][0] = 1;   //Px de C2
              trajetoria[1][1] = 4;   //Py de C2
              trajetoria[2][0] = 1;   //Px de C1
              trajetoria[2][1] = 2;   //Py de C1
              break;
          }
          break;
        }
        return;
      }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
      void percorreTrajetoria(void){
        float dX, dY, dist;
        double dTeta, tetaPonto;
        proxPonto = 0;
        while(proxPonto >=0  && proxPonto <=5){//Percorre a lista de pontos da trajetória
          if(trajetoria[proxPonto][0] >= 0 && trajetoria[proxPonto][1] >= 0){ //Verifica se o proximo
ponto é valido
            Serial.print("Proximo ponto x: ");
            Serial.print(trajetoria[proxPonto][0]);
            Serial.print(" y: ");
            Serial.println(trajetoria[proxPonto][1]);
            //Apaga o valor das variáveis auxiliares para o cálculo do proximo movimento
            dX = trajetoria[proxPonto][0]-Px;//Calcula a distância em X do ponto a ser alcançado e a
posição atual
            dY = trajetoria[proxPonto][1]-Py;//Calcula a distância em Y do ponto a ser alcançado e a
posição atual
            tetaAlvo = atan2(dY,dX);//Rotação necessária para alcançar o proxPonto da trajetória
            //if(trajetoria[proxPonto][0]<Px){
            //  tetaAlvo = (PI-tetaAlvo);
            //}
            if(tetaAlvo<0){
              tetaAlvo+= (2*PI);
            }
            dist = sqrt(pow(dX,2)+pow(dY,2));
```

```
                    dTeta = tetaAlvo - teta;
                    if(dTeta< -PI){
                      dTeta+=2*PI;
                    }
                    if(dTeta>PI){
                      dTeta -= 2*PI;
                    }
                    movimentoConcluido = 0;
                    //Gira para a direção do proximo ponto a ser alcançado
                    Serial.print("GIRA ");
                    Serial.print((dTeta*360)/(2*PI));
                    Serial.print(" _ Teta Alvo ");
                    Serial.println((tetaAlvo*360)/(2*PI));
                    while(movimentoConcluido == 0){
                      moveMouse();
                      //Caso chegue um comando de parada, desejasse que a movimentação seja interrompida
                      //Serial.println(movimento);
                      if(Serial.available()){
                        dado = Serial.read();
                        if(dado == 'P'){
                          movimento = 0;
                          executaComandoRemoto();
                          apagaTrajetoria();
                          return;
                        }
                      }
                      if(dTeta>=0){
                        movimento = 7;//Gira para a esquerda
                      }
                      if(dTeta<0){
                        movimento = 8;
                      }
                      if(teta >= (tetaAlvo-(resolucaoAngulo/2)) && teta <= (tetaAlvo+(resolucaoAngulo/2))){//Verifica se a direção atual já é a desejada
                        movimentoConcluido = 1;
                      }else{
                        executaComandoRemoto();
                      }

                    }
                    //Serial.println("Girou");
                    movimento = 0;//Programa o comando de parada do movimento da cadeira
                    executaComandoRemoto(); //Para a cadeira
                    movimento = 1; //Programa o movimento de andar para frente
```

```
                movimentoConcluido = 0;
                Serial.println("Desloca para o ponto");
                while(movimentoConcluido == 0){
                  moveMouse();
                  //Caso chegue um comando de parada, desejasse que a movimentação seja interrompida
                  if(Serial.available()){
                    dado = Serial.read();
                    if(dado == 'P'){
                      movimento = 0;
                      executaComandoRemoto();
                      apagaTrajetoria();
                      return;
                    }
                  }
                  //Verifica se a posição atual já é a desejada
                  if(Px >= (trajetoria[proxPonto][0]-toleranciaDist) && Px <= (trajetoria[proxPonto][0]+toleranciaDist && Py >= (trajetoria[proxPonto][1]-toleranciaDist) && Py <= (trajetoria[proxPonto][1]+toleranciaDist))){//Verifica se a posição em X atual está dentro do esperado
                    movimentoConcluido = 1;
                    Serial.println("Já to aqui.");
                  }
                  else{
                    executaComandoRemoto();
                    //Serial.println("Andando...");
                  }
                 }
                }
                proxPonto++; //Passa para a proxima posição da lista
               }
               //Para a cadeira
               movimento = 0;
               executaComandoRemoto();
               return;
              }
        ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
              void moveMouse(void){
               leJoystick();
               if (eixoY > (500+zonaMortaJoystick)) {
                 Mouse.move(0, -2);
                 //Serial.println("F");
               }
               if (eixoY < (500-zonaMortaJoystick)) {
                 Mouse.move(0, 2);
                 //Serial.println("T");
               }
```

```
  if (eixoX > (500+zonaMortaJoystick)) {
    Mouse.move(-2, 0);
    //Serial.println("E");
  }
  if (eixoX < (500-zonaMortaJoystick)) {
    Mouse.move(2, 0);
    //Serial.println("D");
  }
  if (digitalRead(pinoZ) == LOW) {
    Mouse.click(MOUSE_LEFT);
    Serial.println("Clicou");
    delay(300);
  }
  delay(10);
  return;
 }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
 void executaComandoRemoto(void){
  mostraStatusSistema();
  calcRampa();
  //Serial.print(rampa);
  switch (movimento){
     case 0: //PARADO
       correcaoPWMLD = 0;
       correcaoPWMLE = 0;
       pulsosE = 0;
       pulsosD = 0;
       rampa = 0;
       modoGiro = 0;
       giroPositivo = 0;
       movimentoPositivo = 0 ;
       //Serial.println("P - R");
       analogWrite(motorE_PWMEF, (0));
       analogWrite(motorE_PWMET, (0));
       analogWrite(motorD_PWMDF, (0));
       analogWrite(motorD_PWMDT, (0));
       break;
     case 1: //PARA FRENTE
       modoGiro = 0;
       giroPositivo = 0;
       movimentoPositivo = 1 ;
       //Serial.println("F - R");
       analogWrite(motorE_PWMEF, (rampa-correcaoPWMLE));
       analogWrite(motorE_PWMET, (0));
```

```
                analogWrite(motorD_PWMDF, (rampa-correcaoPWMLD));
                analogWrite(motorD_PWMDT, (0));
                break;
              case 4: //PARA TRÁS
                modoGiro = 0;
                giroPositivo = 0;
                movimentoPositivo = 0 ;
                //Serial.println("T - R");
                analogWrite(motorE_PWMEF, (0));
                analogWrite(motorE_PWMET, (rampa-correcaoPWMLE));
                analogWrite(motorD_PWMDF, (0));
                analogWrite(motorD_PWMDT, (rampa-correcaoPWMLD));
                break;
              case 7: //GIRO PARA ESQUERDA
                modoGiro = 1;
                giroPositivo = 1;
                movimentoPositivo = 0 ;
                //Serial.println("E - R");
                analogWrite(motorE_PWMEF, (0));
                analogWrite(motorE_PWMET, (rampa-correcaoPWMLE));
                analogWrite(motorD_PWMDF, (rampa-correcaoPWMLD));
                analogWrite(motorD_PWMDT, (0));
                break;
              case 8: //GIRO PARA DIREITA
                modoGiro = 1;
                giroPositivo = 0;
                movimentoPositivo = 0 ;
                //Serial.println("D - R");
                analogWrite(motorE_PWMEF, (rampa-correcaoPWMLE));
                analogWrite(motorE_PWMET, (0));
                analogWrite(motorD_PWMDF, (0));
                analogWrite(motorD_PWMDT, (rampa-correcaoPWMLD));
                break;
            }
            return;
          }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
          void calcRampa(void){
            if(F = 0 && T == 0 && E==0 && D==0 && manual==1){
              rampa = 0;
            }
            else{
              rampa += 1;
              if(rampa > maxPWM){
```

```
      rampa = maxPWM;
    }
   }
   delay(8);
   return;
  }
 ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
  void calculaDirecao(void){          // CALCULO DA AMPLITUDE DO POSICIONAMENTO DO JOYSTICK_
   if(eixoY>(500+zonaMortaJoystick)){ //Cálculando F
    F = abs(eixoY-512)/2;
    if(F>255){
     F = 255;
    }
   }
   else{
    F = 0;
   }
   if(eixoY<(500-zonaMortaJoystick)){ //Cálculando T
    T = abs(eixoY-512)/2;
    if(T > 255){
     T = 255;
    }
   }
   else{
    T = 0;
   }
   if(eixoX>(500+zonaMortaJoystick)){ //Cálculando E
    E = abs(eixoX-512)/2;
    if(E>255){
     E = 255;
    }
   }
   else{
    E = 0;
   }
   if(eixoX<(500-zonaMortaJoystick)){ //Cálculando D
    D = abs(eixoX-512)/2;
    if(D>255){
     D = 255;
    }
   }
   else{
    D = 0;
   }
```

```
                return;
              }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
              void movCadeira(void){          //MOVIMENTAÇÃO DA CADEIRA

    /*  PARADO  */  if(eixoY >= (500-zonaMortaJoystick) && eixoY <= (500+zonaMortaJoystick) && eixoX >= (500-zonaMortaJoystick) && eixoX <= (500+zonaMortaJoystick)){
                pulsosE = 0;
                pulsosD = 0;
                correcaoPWMLD = 0;
                correcaoPWMLE = 0;
                pulsosE = 0;
                pulsosD = 0;
                modoGiro = 0;
                giroPositivo = 0;
                movimentoPositivo = 0 ;
                movimento = 0;
                rampa = 0;
                //Serial.print(rampa);
                //Serial.println("P - M");
                analogWrite(motorE_PWMEF, (0));
                analogWrite(motorE_PWMET, (0));
                analogWrite(motorD_PWMDF, (0));
                analogWrite(motorD_PWMDT, (0));
              }
    /*  FRENTE  */  if(eixoY > (500+zonaMortaJoystick) && eixoX > (500-zonaMortaJoystick) && eixoX < (500+zonaMortaJoystick) && F != 0 && T == 0 && E == 0 && D == 0){
                modoGiro = 0;
                giroPositivo = 0;
                movimentoPositivo = 1 ;
                //Serial.print(F);
                //Serial.println("F - M");
                movimento = 1;

                if(F != 0 && F < rampa){
                  analogWrite(motorE_PWMEF, (F-correcaoPWMLE));
                  analogWrite(motorE_PWMET, (0));
                  analogWrite(motorD_PWMDF, (F-correcaoPWMLD));
                  analogWrite(motorD_PWMDT, (0));
                }
                else{
                  analogWrite(motorE_PWMEF, (rampa-correcaoPWMLE));
                  analogWrite(motorE_PWMET, (0));
                  analogWrite(motorD_PWMDF, (rampa-correcaoPWMLD));
                  analogWrite(motorD_PWMDT, (0));
```

```
                    }
                  }
        /*      TRÁS      */  if(eixoY < (500-zonaMortaJoystick) && eixoX > (500-zonaMortaJoystick) && eixoX < (500+zonaMortaJoystick) && F == 0 && T > 0 && E == 0 && D == 0){
                    modoGiro = 0;
                    giroPositivo = 0;
                    movimentoPositivo = 0;
                    //Serial.print(T);
                    //Serial.println("T - M");
                    movimento = 4;

                    if(T != 0 && T < rampa) {
                      analogWrite(motorE_PWMEF, (0));
                      analogWrite(motorE_PWMET, (T-correcaoPWMLE));
                      analogWrite(motorD_PWMDF, (0));
                      analogWrite(motorD_PWMDT, (T-correcaoPWMLD));
                    }
                    else{
                      analogWrite(motorE_PWMEF, (0));
                      analogWrite(motorE_PWMET, (rampa-correcaoPWMLE));
                      analogWrite(motorD_PWMDF, (0));
                      analogWrite(motorD_PWMDT, (rampa-correcaoPWMLD));
                    }
                  }
        /* GIRO ESQUERDA  */  if(eixoX > (500+zonaMortaJoystick) && eixoY > (500-zonaMortaJoystick) && eixoY < (500+zonaMortaJoystick) && F == 0 && T == 0 && E > 0 && D == 0){
                    modoGiro = 1;
                    giroPositivo = 1;
                    movimentoPositivo = 0 ;
                    //Serial.print(E);
                    //Serial.println("E - M");
                    movimento = 7;
                    if(E != 0 && E < rampa){
                      analogWrite(motorE_PWMEF, (0));
                      analogWrite(motorE_PWMET, (E-correcaoPWMLE));
                      analogWrite(motorD_PWMDF, (E-correcaoPWMLD));
                      analogWrite(motorD_PWMDT, (0));
                    }
                    else{
                      analogWrite(motorE_PWMEF, (0));
                      analogWrite(motorE_PWMET, (rampa-correcaoPWMLE));
                      analogWrite(motorD_PWMDF, (rampa-correcaoPWMLD));
                      analogWrite(motorD_PWMDT, (0));
                    }
                  }
```

```
        /* GIRO DIREITA   */  if(eixoX < (500-zonaMortaJoystick) && eixoY > (500-zonaMortaJoystick) && eixoY
< (500+zonaMortaJoystick) && F == 0 && T == 0 && E == 0 && D > 0 ){
                        modoGiro = 1;
                        giroPositivo = 0;
                        movimentoPositivo = 0 ;
                        //Serial.print(D);
                        //Serial.println("D - M");
                        movimento = 8;
                        if(D != 0 && D < rampa){
                          if(correcaoPWMLE == maxPWM){
                            analogWrite(motorE_PWMEF, 0);
                          }
                          else{
                            analogWrite(motorE_PWMEF, (D-correcaoPWMLE));
                          }
                          analogWrite(motorE_PWMET, (0));
                          analogWrite(motorD_PWMDF, (0));
                          if(correcaoPWMLD == maxPWM){
                            analogWrite(motorD_PWMDT, 0);
                          }
                          else{
                            analogWrite(motorD_PWMDT, (D-correcaoPWMLD));
                          }
                        }
                        else{
                          if(correcaoPWMLE == maxPWM){
                            analogWrite(motorE_PWMEF, 0);
                          }
                          else{
                            analogWrite(motorE_PWMEF, (rampa-correcaoPWMLE));
                          }
                          analogWrite(motorE_PWMET, (0));
                          analogWrite(motorD_PWMDF, (0));
                          if(correcaoPWMLD == maxPWM){
                             analogWrite(motorD_PWMDT, 0);
                          }
                          else{
                            analogWrite(motorD_PWMDT, (rampa-correcaoPWMLD));
                          }
                        }
                      }
                      return;
        }
```

APÊNDICE 3.

Código para a interface visual implementado no Processing.

```
/**
 * Envia_comando_interace.
 *
 * Lê o comando selecionado na tela e envia para o arduino
 */
import processing.serial.*; //Importa a biblioteca para abrir uma comunicação Serial
Serial myPort; //Instância a biblioteca para a comuniação Serial

PImage bg;
int y;

void setup() {
  String portName = Serial.list()[0]; //Lista as portas COM (Serial) encontradas
                                      //Pega a primeira porta (Posição 0 em "Serial.list()[0]" e
                                      //armazena na variável portName
                                      //Se você tiver mais de uma porta COM, identifique qual a do Garagino
  myPort = new Serial(this, portName, 9600); //Abre uma comunicação Serial com baud rate de 9600
  size(1346, 642);
  bg = loadImage("ambienteTeste.png");
}

void draw() {
  //IMAGEM DE FUNDO
  background(bg); //Exime uma imagem de fundo
  stroke(0, 0, 0);//cor da linha ou borda desenhada
  //MOSTRA POSIÇÃO DO MOUSE E DIREÇÃO
  fill(255, 255, 255);
  rect(-width/2, height/2-200, 250, 230);//Fundo de imagem para o mouse
  textSize(32); //Define o tamanho da letra
  fill(204, 102, 0); //Define a cor Laranja (R, G, B)
  text("MouseX = " + (mouseX), width/2+280, height/2); //Escreve MouseX + o valor da posição do mouse na tela
  text("MouseY = " + (mouseY), width/2+280, height/2-30); //Escreve MouseY + o valor da posição do mouse na tela
  fill(255, 255, 255, 255); //Define a cor Branca (R, G, B)
}




void mouseClicked(){
```

```
  //VERIFICA SE UMA DIREÇÃO FOI SELECIONADA
  if(mouseX >= 950 && mouseX<=1200 && mouseY>=400 && mouseY<=575){ //Botão de parada
    textSize(32);
    fill(0,0,0,255);
    text("Parado", width/2+280, height/2-60);
    myPort.write('P');
  }
  else if(mouseX >= 950 && mouseX<=1200 && mouseY>=45 && mouseY<=220){ //Modo Manual
    textSize(32);
    fill(0,0,0,255);
    text("Manual", width/2+280, height/2-60);
    myPort.write('M');
  }
  else if(mouseX >= 50 && mouseX<=250 && mouseY>=45 && mouseY<=570){ //Cozinha
    textSize(32);
    fill(0,0,0,255);
    text("Cozinha", width/2+280, height/2-60);
    myPort.write('C');
  }
  else if(mouseX >= 251 && mouseX<=680 && mouseY>=45 && mouseY<=570){ //Sala
    textSize(32);
    fill(0,0,0,255);
    text("Sala", width/2+280, height/2-60);
    myPort.write('S');
  }
  else if(mouseX >= 681 && mouseX<=880 && mouseY>=45 && mouseY<=255){ //Banheiro
    textSize(32);
    fill(0,0,0,255);
    text("Banheiro", width/2+280, height/2-60);
    myPort.write('B');
  }
  else if(mouseX >= 681 && mouseX<=880 && mouseY>=256 && mouseY<=570){ //Quarto
    textSize(32);
    fill(0,0,0,255);
    text("Quarto", width/2+280, height/2-60);
    myPort.write('Q');
  }
  //delay(100);
}
```