André Felipe Cruvinel Portas
Daniel Xavier Juc

Interface Háptica e Ambiente Virtual para Reabilitação Motora

Orientador: Prof. Dr. Rafael Traldi Moura

São Paulo, 16 de novembro de 2017

André Felipe Cruvinel Portas
Daniel Xavier Juc

Projeto e Construção de um Dispositivo de Interface Háptica e Jogo Virtual para Reabilitação Motora

Monografia apresentada ao departamento de Engenharia Mecatrônica e Sistemas Mecânicos da Escola Politécnica da Universidade de São Paulo como projeto de conclusão de curso da graduação.

São Paulo, 16 de novembro de 2017

Catalogação-na-publicação

Portas, André Felipe Cruvinel
Interface háptica e ambiente virtual para reabilitação motora / A. F. C. Portas, D. X. Juc -- São Paulo, 2017.
142 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Reabilitação 2.Biomecânica 3.Interação Homem-Máquina 4.Jogo Sério I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t. III.Juc, Daniel Xavier

Agradecimentos

Ao nosso orientador, Prof. Dr. Rafael Traldi Moura, e co-orientador Prof. Dr. Arturo Forner Cordero, por todo o apoio, conhecimento, confiança e dedicação ao sucesso deste projeto. Ao engenheiro Rafael Souza, pelo auxílio no desenvolvimento do software do dispositivo. Aos profissionais da saúde Maria Munhoz e Thaís Terranova, pelo conhecimento dividido e acompanhamento do trabalho. Ao Instituto de Reabilitação Lucy Montoro pela visita oferecida. Aos demais integrantes do laboratório de engenharia biomecatrônica da Escola Politécnica, pela ajuda prestada ao longo do desenvolvimento do projeto. Aos familiares, amigos e todos que contribuíram de alguma maneira com esse trabalho.

Resumo

O censo do IBGE, realizado em 2010, mostrou que havia aproximadamente 13 milhões de brasileiros com alguma deficiência motora. Este tipo de deficiência demanda um intenso e longo programa de tratamento fisioterápico de reabilitação. Para aumentar a eficiência nesses tratamentos, a medicina de reabilitação tem se unido a engenharia, a fim de desenvolver soluções inteligentes. As interfaces hápticas são ferramentas que proporcionam estímulos de força e, quando aliadas a jogos em ambientes virtuais, podem se tornar uma ótima maneira de tratar doenças como o AVC e a paralisia. Atualmente estes dispositivos são caros e de difícil movimentação, limitando seu alcance aos melhores centros de reabilitação do país. Este trabalho teve como objetivo projetar e construir um dispositivo de interface háptica simples, barato e portátil, para que, quando aliado a um jogo, possa ser usado na reabilitação motora de pessoas com deficiências físicas. O dispositivo desenvolvido é capaz de detectar o movimento do braço humano e impor resistência ou assistência. Esta imposição é feita a partir de um mecanismo com dois graus de liberdade, atuado por dois motores de corrente contínua. O usuário recebe estímulo visual e sonoro através de um jogo sério (jogo com finalidade de exercício), compatível com um computador pessoal comum. Para garantir sua aplicabilidade em tratamentos, o histórico dos exercícios é armazenado num banco de dados, futuramente acessível ao fisioterapeuta responsável. Foram desenvolvidos e testados métodos de controle do motor a partir de corrente e de posição. O primeiro se mostrou adequado à proposta, sendo possível dimensionar e direcionar as forças satisfatoriamente. O modo de posição não mostrou resultados positivos, de maneira que foi descartado como solução. Foi possível desenvolver e construir um protótipo funcional leve e com baixo custo, com validação de engenharia, realizada através de cálculos e gráficos, e com validação pré-clínica, realizada por profissionais da área. Os futuros trabalhos relacionados a este projeto incluem a criação dos módulos de segurança, aumento da capacidade de carga e área útil, melhora da estética do produto e aperfeiçoamento do software desenvolvido. Esta monografia tem por objetivo expor o projeto (mecânico, elétrico, de controle e de software), a construção, os testes e as análises relacionadas ao desenvolvimento deste protótipo.

Abstract

The IBGE census, conducted in 2010, showed that there were approximately 13 million Brazilians with some kind of motor impairment. This type of disability requires a long, intense physical rehabilitation program. To increase efficiency in these treatments, is possible to join engineering and medicine in order to develop intelligent solutions. Haptic interfaces are tools capable of providing force feedback and, when coupled with games and virtual environments, can become a great way to treat diseases such as stroke and paralysis. Currently these devices are expensive and have low mobility, limiting their reach to the best rehabilitation centers in the country. This work aimed to design and build a haptic interface device that is simple, cheap and portable, so that when coupled with a game, it can be used in the motor rehabilitation of people with physical disabilities. The developed device is able to detect the movement of the human arm and impose resistance or assistance. The imposition of forces is generated by mechanism with two degrees of freedom, actuated by two DC motors. The user receives visual and sound stimulation through a serious game (game with exercise purpose) compatible with any personal computer. To ensure its applicability in treatments, the exercise records are stored in a database, in the future accessible to the responsible physiotherapist. Current and position control methods were developed and tested. The first one was adequate to the proposal, meaning that the device was able to dimension and direct the forces with reasonable precision. The position mode did not show positive results, so it was discarded as a possible solution. It was possible to develop and construct a light and low cost functional prototype, with engineering validation, performed through calculations and graphs, and with pre-clinical validation, performed by professionals of the area. Future work related to this project includes the creation of safety modules, increased load capacity and area of use, improvement of product aesthetics and improvement of software development. This monograph aims to expose the design (mechanical, electrical, control and software), construction, testing and analysis related to the development of this prototype.

Lista de Figuras

Lista de Tabelas

Sumário

I. Introdução

Com o objetivo de definir conceitos e padrões no que se refere às deficiências, a Organização Mundial da Saúde (OMS) publicou o documento de Classificação Internacional das Deficiências, Incapacidades e Desvantagens (CIDID). As incapacidades (disabilities), segundo o CIDID, são consideradas como as consequências de uma deficiência do ponto de vista funcional. Segundo a análise de Buchalla (2016), as deficiências (impairment) são definidas como anormalidades nos órgãos, sistemas e estruturas do corpo. Por último, as desvantagens (handicaps) representam a relação do indivíduo deficiente com o ambiente dificultado pelas incapacidades.

O censo do IBGE, feito em 2010, mostrou que de 191 milhões de brasileiros, aproximadamente 46 milhões apresentavam algum tipo de deficiência (visual, auditiva, motora ou mental), o que correspondia a 23,9% da população do país. Apenas no estado de São Paulo, os deficientes eram 9 milhões, dentre os 41 milhões, correspondendo a 22,6% dos paulistas. Ainda baseado no censo de 2010, observa-se que a deficiência física ou motora é a segunda mais comum no país, com 13 milhões de afetados (7,0% do total), atrás apenas da deficiência visual, que acometia 29 milhões (18,8% do total).

Segundo definição do Ministério da Educação e Cultura, as deficiências físicas são condições motoras que prejudicam pessoas, comprometendo a mobilidade e a coordenação motora geral, em consequência de lesões neurológicas, neuromusculares, ortopédicas ou más formações congênitas. Foi publicado um decreto em 2004, (número 5.296) que definiu a deficiência física como:

> A alteração completa ou parcial de um ou mais segmentos do corpo humano, acarretando o comprometimento da função física, apresentando-se sob a forma de paraplegia, paraparesia, monoplegia, monoparesia, tetraplegia, tetraparesia, triplegia, triparesia, hemiplegia, hemiparesia, ostomia, amputação ou ausência de membro, paralisia cerebral, nanismo, membros com deformidade congênita ou adquirida, exceto as deformidades estéticas e as que não produzam dificuldades para o desempenho de funções.

As deficiências físicas podem ser classificadas em três categorias, segundo sua tratabilidade: definitiva, quando a recuperação não é possível; compensável, quando o tratamento se dá por substituição (prótese); recuperável ou temporária, quando a melhora é possível por tratamento e fisioterapia. Já as causas para este problema são

classificadas em três: hereditária, quando é geneticamente transmitida; congênita, quando o problema já existe no nascimento; adquirida, quando ocorre devido a infecções, traumas, intoxicações e outros. Dois exemplos comuns de enfermidades que podem acarretar deficiências físicas são o acidente vascular cerebral ("AVC" ou derrame) e a paralisia cerebral.

De acordo com o documento "Atlas das Doenças Cardíacas e Derrame", da OMS, o AVC é causado por uma perturbação no fornecimento de sangue para o cérebro, por consequência de bloqueio ou rompimento de vasos, podendo causar danos permanentes ao encéfalo. Como é discutido por Silva (2011), as possíveis consequências do derrame são paralisias motoras, alterações sensoriais, na comunicação, cognitivas e distúrbios emocionais. Segundo ele, a paralisia é a mais comum e afeta metade do corpo, que não tem tônus suficiente para iniciar movimentos ou realizar resistência passiva. A paralisia ainda pode vir acompanhada de problemas de equilíbrio e coordenação. Foi realizada uma entrevista com a fisioterapeuta Maria Munhoz, formada pela UNIP em 2009, onde entre outros assuntos, foi abordado o tratamento de AVC. Segundo ela, o tratamento do AVC tem maiores resultados nos primeiros meses, sendo que depois de 6 meses não há ganhos relevantes.

Leite e Prado (2004) definem a paralisia cerebral ("PC") como um grupo de implicações do sistema nervoso, causada por lesões ao cérebro, nos períodos pré, peri ou pós-natal. A doença é permanente, mas não tem caráter progressivo e é caracterizada por alterações de movimento, postura, equilíbrio e coordenação, além da geração de movimentos involuntários. A doença é classificada pelo tipo e grau de disfuncionalidade. Em entrevista, a fisioterapeuta Maria Munhoz, apontou que o tratamento da paralisia não gera muita melhora, entretanto sendo essencial para o mantimento dos movimentos restantes.

Uma vez com alguma deficiência, o paciente deve passar por diversos procedimentos, tais como cirurgias e tratamentos fisioterápicos de reabilitação. Como constatado por Sari e Marcon (2008), os tratamentos de reabilitação de deficiências, como a paralisia infantil, podem durar de meses até anos. Silva (2011) atribui o tratamento de pacientes de AVC a uma equipe de profissionais de saúde formada por enfermeiros, fisioterapeutas, fonoaudiólogos, terapeutas ocupacionais, psicólogos e assistentes sociais. A grande quantidade de horas de tratamento, aliado à atitude da sociedade, podem tornar o processo desgastante e desmotivador para o paciente.

Para deficiências definitivas, não há tratamento corretivo, e o paciente carrega as disfuncionalidades para toda sua vida. Para estes casos, fica evidente a necessidade

constante do acompanhamento de cuidadores. Nestes casos, a função dos cuidadores e familiares é auxiliar na superação emocional do problema, procurando formas de proporcionar o maior nível possível de independência e conforto no seu dia-a-dia. Mesmo que não haja esperança de cura, é importante continuar os tratamentos a fim de evitar lesões e atrofiamentos, além de ajudar psicologicamente o paciente.

Nos casos de deficiências compensativas, o nível de disfuncionalidade permite a substituição dos membros prejudicados. O tratamento assistivo busca proporcionar ou ampliar habilidades motoras e funcionais, visando promover independência e inclusão ao paciente. Os exemplos mais comuns são próteses, muito utilizadas por pacientes que passaram por amputações. No campo da tecnologia, os exoesqueletos de ponta são dispositivos atuados, projetados para auxiliar o indivíduo na realização de funções, a partir da geração de forças resistivas e ativas de caráter somativo em membros com capacidade limitada. Outro exemplo recente são os membros robóticos, projetados para realizar funções apenas com a força proveniente dos atuadores.

Quando a deficiência é do tipo recuperável, ou temporária, o tratamento mais adequado é a fisioterapia. Segundo O'Sullivan, Schmitz e Fulk (2013), o processo de fisioterapia passa por 6 etapas, em ordem: exame, avaliação, diagnóstico, prognóstico, intervenção e resultados. A fase de exame compreende a aquisição de dados do paciente, através de entrevistas, testes e medições. O processo de avaliação é caracterizado pelo tratamento e análise dos dados obtidos na fase anterior, onde o fisioterapeuta identifica as dificuldades. O diagnóstico é a consequência da análise, onde o profissional organiza os resultados e definindo as síndromes e deficiências. O prognóstico é o desdobramento do diagnóstico, onde o fisioterapeuta define o nível de melhora a ser atingido e o tempo necessário para isso. Nessa fase também são definidos os procedimentos e exercícios a serem utilizados. O processo de intervenção é a interação técnica e objetiva entre o profissional e o paciente, utilizando diversos métodos e técnicas de reabilitação para produzir melhoras nas condições do paciente. Durante este processo, exames são refeitos para avaliação de melhoras e possíveis redirecionamentos no tratamento. Por fim, a fase de resultados compreende uma avaliação do desempenho e eficácia da fisioterapia, especialmente da fase de intervenção.

É possível dizer que a intervenção fisioterápica pode ser dividida em duas, sendo elas a clássica e tecnológica. A intervenção clássica é baseada em exercícios simples em métodos bem estabelecidos, podendo utilizar o auxílio objetos projetados para este propósito. Técnicas modernas e novos dispositivos eletrônicos são as características

principais da fisioterapia tecnológica. Como discutido por Dos Santos Nunes (2011), a tecnologia tem se aliado com a medicina, buscando melhorar e inovar nos diversos tratamentos de doenças e dificuldades. Nessa linha de raciocínio, a medicina de reabilitação tem se aliado a engenharia, visando encontrar métodos mais eficientes para tratamentos, a partir da criação de dispositivos inteligentes. Outra consequência da modernização é a maior precisão de movimentos e diagnósticos, além da maior segurança.

O grupo procurou e realizou entrevistas com dois profissionais da saúde que atuam em ambientes bastante distintos. A primeira foi a fisioterapeuta Maria Munhoz, que atua comercialmente e não está vinculada a nenhuma instituição de ensino ou pesquisa. Maria não conhecia nenhum aparelho do tipo interface háptica, sendo que ela trabalha principalmente com exercícios envolvendo elásticos, pesos, eletroestimulação (corrente russa) e terapia manual. Maria apontou que a maioria dos tratamentos fisioterápicos de reabilitação deveriam ser feitos para o resto da vida do paciente (para regeneração e mantimento), sendo que o ideal é a realização diária (quanto mais se exercitar, melhor). Ao ser apresentada à proposta do dispositivo, Maria afirmou que sua aplicabilidade é válida, garantindo estímulo maior que o do exercício tradicional. Também apontou que o tratamento trabalharia tanto a cognição quanto o movimento, fazendo o cérebro procurar outro caminho pra fazer a mesma atividade. Para ela, o equipamento seria útil no tratamento de doenças como AVC, paralisia, transtornos autoimunes degenerativas (esclerose), distrofia (síndrome de Duchenne) e também para o tratamento periódico de idosos (motricidade e cognição). Por fim, Maria inferiu que o preço final do produto para pacientes tem que ser bastante acessível (baseado em seu público), visto que na visão dela, os brasileiros não priorizam investimentos na qualidade de vida.

Também foi feita uma visita ao Instituto de Reabilitação Lucy Montoro (unidade Vila Mariana), guiada por Thaís Terranova, formada em terapia ocupacional pela USP e que atua na área de pesquisa. A unidade visitada atende aproximadamente 300 pacientes e oferece tratamento para pessoas que sofrem (ou sofreram) de AVC, paralisia cerebral, lesão medular e encefálica, amputação, entre outros. Os programas de reabilitação são multidisciplinares, incluindo acompanhamento médico de fisioterapeutas, psicólogos, terapeutas ocupacionais, e até fonoaudiólogos. Este instituto utiliza a robótica de forma ampla nos tratamentos fisioterápicos. Os equipamentos disponíveis lá (incluindo dispositivos de interface háptica) são utilizados tanto para o tratamento de pacientes, quanto para pesquisa e desenvolvimento na área.

I.i. Revisão Bibliográfica

Para analisar a fisioterapia tecnológica, serão utilizados três pontos de vista diferentes: eficiência/eficácia do tratamento, grau de imobilização do dispositivo e preço da aparelhagem. A necessidade de eficácia na reabilitação a partir de um novo dispositivo é óbvia, mas para que o investimento em novos projetos valha a pena, a eficiência do tratamento deve ser maior que a dos procedimentos clássicos. A mobilidade do aparelho nem sempre é necessária, mas, reconhecendo que cada paciente tem uma ergonomia e necessidades diferentes durante o tratamento, esta pode ser de grande auxílio para o fisioterapeuta. Por fim, o parâmetro presente em qualquer projeto é o custo. Neste caso em específico, o aumento do custo no tratamento deve ser compatível com o aumento na eficiência do mesmo.

Fasoli (2003) e sua equipe realizaram um experimento, compreendendo 20 pacientes que sofreram derrame. Estes foram submetidos a testes por um período de tempo antes do tratamento, que não indicaram melhoras significativas. Os pacientes foram submetidos a um tratamento para o membro superior auxiliado pela robótica, que utilizou o dispositivo MIT-MANUS (descrito posteriormente em detalhes). Os testes feitos após 6 semanas de tratamento demonstraram diminuição significante da deficiência apresentada. O texto ainda discute que o experimento serviu para embasar estudos anteriores que afirmaram que a terapia auxiliada pela robótica pode ser muito eficiente, se usada como complemento aos tratamentos clássicos. Por esse ponto de vista, para garantir maior eficiência nos processos de reabilitação, a tecnologia utiliza três recursos: feedback visual (realidade virtual), feedback tátil (interfaces hápticas) e jogos lúdicos (jogos sérios).

Como é discutida por Burdea e Coiffet (1994), a realidade virtual pode ser definida por uma representação computacional gráfica do mundo real que interage instantaneamente com o usuário. Essa interação homem-máquina pode ocorrer por meio de diversas formas, dependendo principalmente do objetivo para qual esse recurso será empregado. Contextualmente, a tecnologia de realidade virtual é aplicada na área médica já com uma grande abrangência. De acordo com Dos Santos Nunes et al (2011), esse recurso está presente em diversos segmentos como: educação, facilitando estudos da anatomia e fisiologia; treinamento, com simuladores cirúrgicos e de exames ginecológicos e de biópsias; reabilitação, com desenvolvimento de ambientes tridimensionais e interativos; ferramentas de desenvolvimento rápido. Logo, há uma demanda significativa para o desenvolvimento dessa tecnologia. No mesmo tema,

Christ e Reiner (2014) discutem a ilusão gerada pelo movimento no ambiente virtual e suas aplicações clínicas, principalmente na reabilitação motora.

A eficácia e eficiência dos tratamentos aliados à realidade virtual foram estudadas por Cameirão e sua equipe (2012). Um experimento foi realizado, utilizando três diferentes sistemas de tratamento (utilizando ambientes virtuais) em 44 pacientes: rastreamento baseado na visão, interface háptica e exoesqueleto passivo. Segundo os autores, todos os pacientes apresentaram resultados positivos após um mês. Como discutido no artigo, existem duas hipóteses para este resultado. A primeira é a de que o feedback sensorial aumenta o envolvimento e motivação do paciente e a segunda é a de que nesse sistema, diversos aspectos são trabalhados ao mesmo tempo, tais como a cognição, o feedback visual e o tátil.

No contexto da fisioterapia, o feedback tátil é introduzido pela interface háptica, ou interface tátil. Como é discutido por Da Silva (2014), as interfaces hápticas são dispositivos de reação que proporcionam estímulos a partir de geração de forças, movimentos ou texturas. Tais sistemas são responsáveis pela resposta física à interação do usuário com o ambiente virtual. No artigo de Charles, Krebs, Volpe, Lynch e Hogan (2005), o principal assunto tratado é a implementação de uma "munhequeira robótica" para fins terapêuticos de indivíduos que passaram por AVC. A munhequeira é um mecanismo sofisticado, que apresenta 3 graus de liberdade (abdução-adução, flexão-extensão, pronação-supinação) e pode ser aplicado com "jogos sérios" (definidos posteriormente). A partir de estudos controlados com pessoas que apresentavam o tipo de problema em questão, notou-se um efeito significativo com as atividades dos pacientes, mostrando resultados promissores.

De acordo com Maia (2013), jogos sérios são desafios lúdicos que possuem propósito educacional plenamente explícito e como prioridade, ao invés de proporcionar apenas entretenimento. No campo da fisioterapia, tais jogos visam preferencialmente o exercício fisioterápico, sem excluir o lado lúdico, proporcionando estímulo e feedback visual, além da motivação para o paciente realizar o exercício com empenho total. Maia (2013) ainda discute a possibilidade de o paciente utilizar os jogos em casa, com maior frequência, mas alerta para a necessidade de existir o armazenamento dos dados do exercício para posterior análise de um especialista. Moritz (2011) testou a eficácia de jogos sérios para a reabilitação em seu projeto "Neurogame Therapy", onde utilizou jogos de computador conhecidos para exercitar as disfuncionalidades de pacientes. Seu estudo, feito com pacientes que sofreram de derrame ou lesão cerebral traumática, mostrou melhoras significativas nos seguintes aspectos: alcance de movimentos,

capacidade de ativar músculos paralisados/atrofiados e habilidade de ativar músculos independentemente dos outros.

I.ii. Estado da Arte

O dispositivo de interface háptica PHANTOM foi desenvolvido pela empresa SensAble Devices Inc. em 1994. Massie e Salisburry (1994) definem o PHANTOM como sendo um aparelho que mede a posição do dedo do paciente no espaço tridimensional e exerce forças precisamente controladas e direcionadas (com três graus de liberdade), permitindo a interação de usuário com inúmeros objetos virtuais e a manipulação remota de robôs manipuladores. Apesar de antigo, o projeto é constantemente atualizado e revisto, garantindo sua compatibilidade com os aparelhos mais tecnológicos. Atualmente a 3D Systems Geomagic comercializa por todo o mundo três versões desse projeto: Geomagic Touch, Geomagic Touch X e o PHANTOM Premium (mais avançado tecnologicamente). Este último permite o feedback tátil em seis graus de liberdade e garante alta precisão, além de um maior alcance de movimento. O dispositivo tem diversas funcionalidades e é portátil, porém, no Brasil, a versão mais simples (e menos potente) é comercializada por volta de R$ 10 mil.

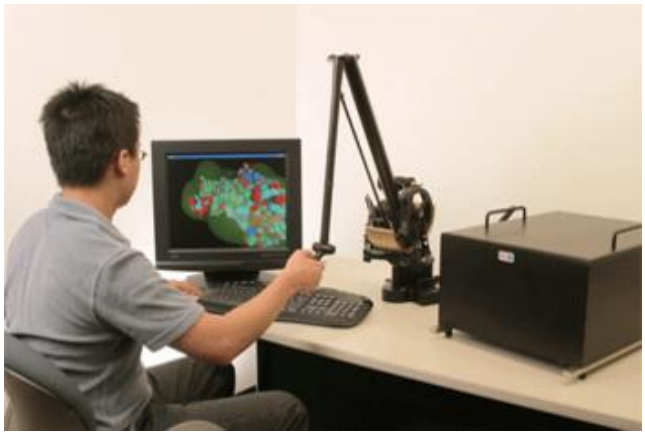


Figura 1 – Dispositivo PHANTOM Premium.
Fonte: página da Geomagic na internet([1])

A desenvolvedora de dispositivos hápticos Haption introduziu o Virtuose 6D em 2013. Com funcionalidades similares ao PHANTOM Premium, este dispositivo produz forças de até 70 N e torques de até 5 Nm, além de proporcionar um espaço de trabalho maior. A mobilidade do dispositivo é comprometida devido às suas dimensões. O dispositivo Falcon da Novit Institute foi desenvolvido em 2006 e é aprimorado desde então. Este aparelho possui apenas três graus de liberdade e proporciona forças de até

[1] Disponível em http://www.geomagic.com/en/products/phantom-premium/overview

10 N, além de um espaço de trabalho de 100mm x 100mm x 100mm. O Falcon é pequeno e portátil, sendo comercializado atualmente por cerca de USD 250,00 e pode ser facilmente integrado a jogos sérios de reabilitação, apesar desta não ser sua função primária. Campion (2005) refez o design de uma interface háptica chamada Pantograph (passando a se chamar Pantograph MK II), buscando melhorar a performance do dispositivo e disponibilizá-lo ao mercado.

Figura 2 – Dispositivos Virtuose 6D e Falcon.
Fontes: respectivamente, página da Haption(2) e da Delft Haptics(3) na internet

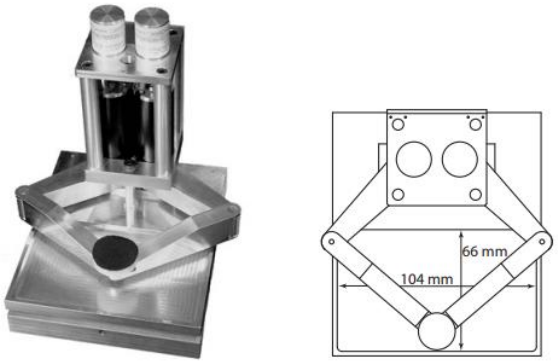


Figura 3 – Dispositivo PantographMK II.
Fonte: CAMPION, 2005, página 1

Desenvolvidos pela Interactive Motion Technologies, o InMotion ARM e o InMotion WRIST utilizam a integração entre um dispositivo háptico com 2 graus de liberdade atuados e diversos jogos sérios para auxiliar pacientes com distúrbios cerebrais a recuperar as habilidades motoras dos membros superiores. Os dispositivos são completamente embarcados (possuindo até uma tela para visualização do jogo), o que os tornam robustos, grandes e pesados, dificultando sua movimentação. Estima-se que o preço de mercado do aparelho esteja em torno de R$ 500 mil. A eficácia do sistema foi testada com mais de mil pacientes, apresentando resultados positivos. O

[2] Disponível em https://www.haption.com/site/index.php/en/products-menu-en/hardware-menu-en/virtuose-6d-menu-en
[3] Disponível em http://www.delfthapticslab.nl/device/novint-falcon/

InMotion tem a capacidade de continuamente analisar e adaptar o exercício às necessidades e habilidades paciente. Isso permite ao fisioterapeuta personalizar a terapia e maximizar os resultados. O Instituto Lucy Montoro visitado possui diversas unidades do dispositivo InMotion para reabilitação motora, utilizados principalmente para o tratamento do AVC. Segundo Thaís Terranova, os aparelhos funcionam muito bem, sendo eficazes no tratamento de reabilitação fisioterápica. Os relatórios de exercício gerados pelos dispositivos são bastante importantes, uma vez que permitem o acompanhamento do progresso pelo fisioterapeuta responsável. A pesquisadora apontou como principais dificuldades do uso deste dispositivo a movimentação do aparelho (devido a seu peso), a manutenção externa de software e hardware e a perda da suavidade do movimento em alguns exercícios. Outro ponto importante foi o relato de reclamações dos pacientes da falta de estímulo frente à monotonia dos jogos e exercícios disponibilizados para o tratamento (todos tem duas dimensões e não apresentam variações de uma sessão para outra).

Outro dispositivo háptico projetado para reabilitação motora, similar ao InMotion, é o MIT-MANUS, desenvolvido pelo laboratório de biomecânica e reabilitação do MIT a partir de 1989. Apesar de o projeto ter mais de 20 anos, ele continua sendo estudado e atualizado, sendo hoje uma das maiores referências científicas no campo da reabilitação fisioterápica assistida por robôs. Segundo Krebs (1998, 2004, 2008 e 2016), este dispositivo é planar, de cadeia fechada e possui dois graus de liberdade atuados, também usado para tratamento dos membros superiores. O mecanismo utiliza controle de impedância ao movimento para auxiliar ou resistir ao movimento do paciente (com forças de até 45 N), sem deixar de garantir total segurança. O exercício também é baseado em jogos sérios, para maximizar o feedback sensorial e a motivação do paciente. Dados do exercício, tais como posição, velocidade e forças aplicadas são armazenadas para análise posterior de especialistas. Os testes realizados por Krebs e sua equipe mostraram resultados positivos.

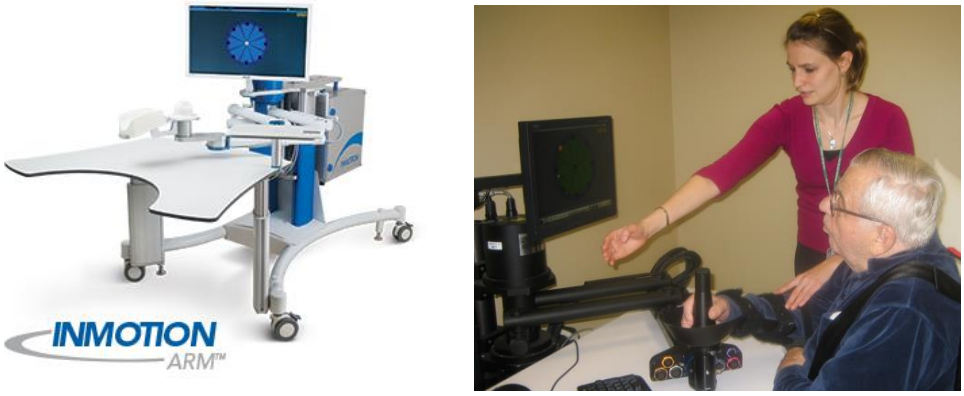


Figura 4 – Dispositivos InMotor ARM e MIT-MANUS.
Fontes: respectivamente página da Bionik(4) e MIT News(5) na internet

Um exemplo existente com os moldes deste projeto é citado por Da Silva (2014) quando explicando formas de utilização da tecnologia háptica. O dispositivo Curictus Virtual Rehabilitation System é usado por pacientes para realizar exercícios em seu ambiente familiar, havendo um monitoramento online por parte do terapeuta responsável. Ainda sobre essa tecnologia, o artigo de Chen (2011) fornece uma explicação mais aprofundada sobre o funcionamento, que envolve: computador, monitor 3D estereoscópico, dispositivo háptico PHANTOM® e um espelho de imersão, além de óculos 3D especiais. Lavatelli, Ferrise e Bordegoni (2014) concluíram o design de uma interface háptica de dois graus de liberdade para ser usada com realidade virtual para fins fisioterápicos, realizando diversas simulações envolvendo o sistema mecânico e de controle.

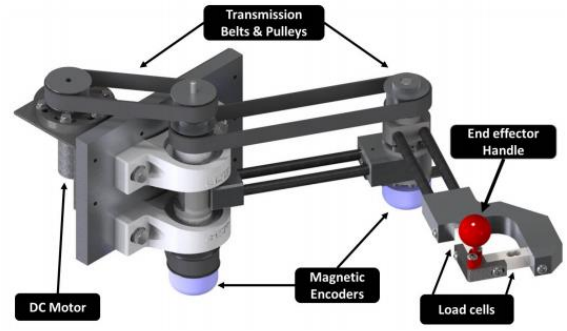


Figura 5 – Projeto do dispositivo OSHap.

Fonte: LAVATELLI, FERRISE e BORDEGONI, 2014 página 3

Apesar de todos os indícios acima mostrados, os benefícios e os problemas relacionados à reabilitação de pacientes a partir da integração de ambientes virtuais a interfaces hápticas não são totalmente conhecidos, como discutem Fluet e Deutsch

[4] Disponível em http://bionikusa.com/healthcarereform/upper-extremity-rehabilitiation/inmotion2-arm/
[5] Disponível em http://news.mit.edu/2014/mit-robot-may-accelerate-trials-for-stroke-medications-0211

(2013). Segundo eles, alguns assuntos ainda devem ser estudados, tais como: necessidade de feedback tátil; possibilidade de uso de ambiente semi-imersivo 2D; possibilidade de aliar decisões médicas ao software; níveis mínimos cognitivos e motores para que o tratamento seja adequado e qual será o custo-benefício do desenvolvimento de tal tecnologia. Além dos problemas mencionados acima, causados principalmente por ser uma tecnologia recente, pode-se discutir a imobilidade e o custo do investimento nestes aparelhos. Os melhores sistemas encontrados no mercado apresentam barreiras claras para sua implementação em massa. Os dispositivos de alta potência e área de trabalho adequada são grandes demais, o que impossibilita a movimentação do conjunto. Sistemas com estas características, não permitem que o paciente se exercite em casa, e com mais frequência, limitando o tratamento ao instituto de reabilitação motora que o possua. Outro grande problema é o preço dos aparelhos, que surge pela alta precisão e potência, implicando em consequências parecidas. Com valor de investimento alto, apenas institutos de ponta conseguem adquirir estes sistemas de tratamento, que por consequência, estão disponíveis apenas para uma parcela pequena da população.

II. Objetivo

Os sistemas hápticos aliados a jogos sérios para reabilitação motora, mencionados anteriormente, possuem diversos aspectos a serem desenvolvidos e melhorados. Desse modo, objetivo deste projeto foi criar um dispositivo de reabilitação motora para membros superiores planar de cadeia fechada e com dois graus de liberdade atuados, que primeiramente garanta eficácia no tratamento, seja seguro para o paciente, portátil, e que seja de baixo custo.

III. Análise do Problema

III.i. Descrição do Projeto

O projeto consiste na construção de um sistema de auxílio na reabilitação física de indivíduos com funções motoras comprometidas. O projeto foi dividido em três partes: desenvolvimento de um ambiente virtual, projeto/construção de uma interface háptica e criação de um sistema de informação para armazenamento e disponibilização de dados dos usuários. Enquanto o ambiente virtual se encarrega do estímulo visual e da motivação para o usuário, a interface física providencia o feedback tátil, gerando resistência mecânica ao movimento, e faz a leitura dos movimentos impostos pelo indivíduo. O projeto ainda prevê a elaboração de jogos sérios (jogos que visam exercício/treinamento). A integração dos sistemas permite a aplicação de exercícios fisioterápicos de reabilitação para pacientes, estimulando sua força de vontade e permitindo a análise do progresso do tratamento pelos envolvidos.

A interface háptica é constituída por um mecanismo de cadeia fechada de dois graus de liberdade ativos (apenas dois graus serão atuados), possibilitando seu uso em exercícios envolvendo de um a dois graus de liberdade (não há rotação e nem movimento do efetuador, onde estarão os motores). Pretendeu-se construir um dispositivo que auxilie no tratamento dos movimentos fundamentais dos braços, influenciando as articulações do cotovelo e ombro. A resistência ao movimento é criada a partir de dois motores DC (possivelmente da Maxon Motor), com controlador (possivelmente o EPOS2) que possibilita controle de corrente.

Sistemas como o mencionado já existem, mas são grandes e caros. Pretendeu-se criar, com o auxílio da tecnologia e do conhecimento obtido na graduação, um sistema mais barato e de fácil reprodução. Como consequência do projeto, espera-se (em comparação com tratamentos convencionais) facilitar e melhorar o tratamento de pacientes com dificuldades motoras.

III.ii. Requisitos de Projeto

Nesta seção serão discutidos os requisitos básicos a serem cumpridos pelo sistema ao final do projeto. Tais requisitos foram debatidos e definidos junto aos orientadores do projeto, sendo adequadamente desafiadores e estimulantes ao conhecimento.

a. Controlar a força gerada na manopla

Como mencionado anteriormente, a função da interface háptica é oferecer resistência ao movimento imposto pelo usuário, para que a experiência seja principalmente um exercício fisioterápico, e não apenas um jogo. Tal resistência provirá dos torques dos dois motores e será gerada proporcionalmente à posição atual do dispositivo, a partir de um sistema de controle adequado.

Este requisito de projeto impõe que a força na manopla deve ser dimensionada e direcionada de modo a garantir uma saída suave e segura para o usuário. Este objetivo deverá ser atingido a partir de um sistema de controle com feedback a ser projetado. Este sistema contará com motores, drivers e um controlador programável adequado.

b. Criar um sistema que permita o uso de sinais de diferentes fontes

O projeto também prevê a criação de um "jogo sério", usado para incentivar o usuário a praticar os exercícios fisioterápicos com a interface háptica. O propósito essencial do sistema proposto é implementar e testar a eficácia de ambientes virtuais e interfaces de feedback tátil no tratamento de pessoas com deficiências motoras. Ainda assim, é desejável que o sistema permita testes com outros dispositivos, tais como exoesqueletos humanos ou membros robóticos em geral.

Este requisito de projeto impõe que será utilizar um protocolo de comunicação específica entre dispositivo/computador, de modo que seja possível utilizar outros dispositivos relacionados à área de biomecatrônica para se utilizar e testar o ambiente de jogo.

c. Finalizar o projeto com custo baixo

A partir de uma pesquisa de mercado, foi encontrado um dispositivo háptico desenvolvido na FAPESP da Vivax, chamado ARM. Segundo notícia da própria instituição, o preço de venda é de aproximadamente R$ 200 mil reais. Para que o dispositivo desenvolvido neste projeto possua mais abrangência, deseja-se que seu custo teórico não ultrapasse 50 mil reais.

d. Facilitar o armazenamento e disponibilização dos dados dos usuários

Para que o tratamento de reabilitação seja eficaz, os dados provenientes do uso da interface háptica devem ser devidamente armazenados e disponibilizados para a análise dos fisioterapeutas. Deve-se criar um sistema de informação facilmente acessível e de interface com usuário tão simples quanto possível. Este requisito de

projeto impõe que o subsistema que envolve o banco de dados deve ser confiável e com fácil acesso e armazenamento para os usuários.

e. Possibilitar a eficácia no tratamento

Para que o equipamento possa ser utilizado em tratamentos de reabilitação motora, devem-se gerar forças no mesmo intervalo da capacidade do paciente. Sabe-se que as dificuldades do paciente estão associadas principalmente à elevação do braço. Nesse sentido, percebe-se que o limite de força do usuário é, então, o peso do membro superior. Para avaliar esse valor foi considerado o modelo de Zatsiorsky e Seluyanov, usado para estimar a massa de cada parte do corpo como uma porcentagem da massa total de um indivíduo. Segundo o modelo, o conjunto *upper arm* + *forearm* + *hand* (parte superior do braço, antebraço e mão) representa em torno de 4,936% da massa do corpo. Tomando como base uma pessoa cuja massa é de 80Kg, o braço teria uma massa de aproximadamente 3,95Kg. Logo, o peso limite (assumindo gravidade $10m/s^2$) que será considerado é de 40N. Este é o requisito de força para um produto final, a ser comercializado e utilizado em tratamentos fisioterápicos. Para fins de prova de conceito no primeiro protótipo do projeto, adotou-se a força de 10 N como requisito, devido à baixa capacidade de torque dos motores disponíveis. Vale ressaltar que o patamar de 10N é suficiente para fisioterapia de alguns pacientes, mas poderia limitar a continuidade e o progresso do tratamento.

IV. Projeto Básico

IV.i. Análise Mecatrônica

Para entender o que foi desenvolvido no projeto, foi elaborado o esquema que pode ser visto a seguir. Ele representa o ciclo de funcionamento do dispositivo. Buscou-se definir todos os componentes que deverão ser estudados. Explicando sequencialmente, o que ocorre é o seguinte:

1. A tela gera estímulos visuais ao usuário;
2. O usuário interage com o mecanismo, aplicando uma força F e recebendo o estímulo háptico;
3. O mecanismo e o motor interagem através da transmissão de rotação e torque;
4. Os valores do encoder e de corrente são verificados pelos drivers, que os repassam ao sistema embarcado;
5. O sistema embarcado realiza os cálculos necessários e envia a posição (x,y,z) atual para atualização no Unity. Simultaneamente são transmitidos os valores de referência (de corrente ou posição) para os drivers;
6. Após receber a posição, o Unity atualiza o "jogo", passando a informação da nova posição para a tela do jogo;
7. Ao mesmo tempo, os controladores implementados pelo driver se encarregam dos setpoints, finalizando assim o ciclo.

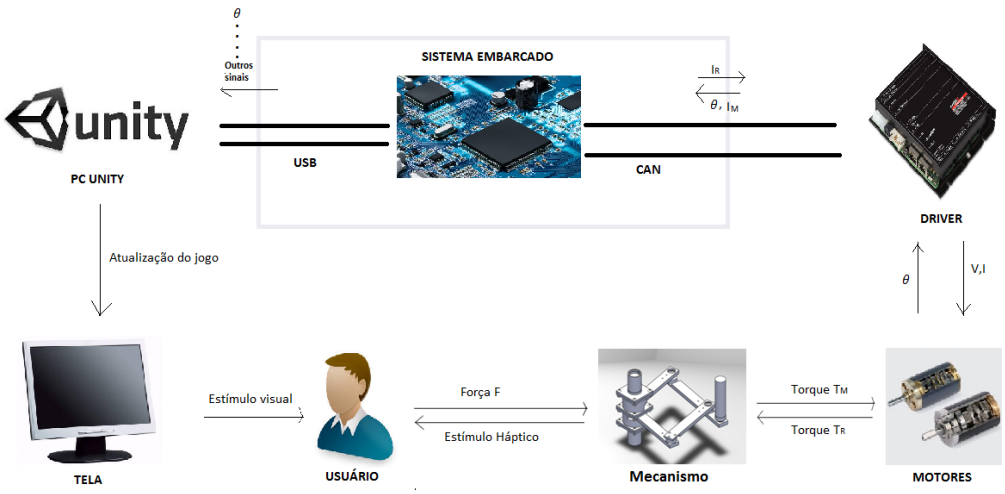


Figura 6 – Análise mecatrônica do projeto.
Fonte: própria

A seguir serão descritas as características desejadas para os componentes citados no esquema, dando ênfase ao que é relevante ao funcionamento adequado do protótipo que será construído.

IV.ii. Tela

A tela de um computador pessoal comum tem frequência de atualização de 30 a 60 Hz, e é adequada à capacidade de captação visual dos olhos humanos. Desse modo, a tela de qualquer computador é adequada ao jogo.

IV.iii. Atuadores (Motor e Redução)

As três possíveis soluções para os motores são DC brushed (DCB), DC brushless (DCBL) e AC. As seguintes afirmações são feitas com base no conteúdo do artigo sobre tipos de motores da OrientalMotor[6]. Os motores DCB dependem de um sistema mecânico para transferir corrente e apesar da boa estabilidade do torque, apresentam perdas devido ao atrito envolvido nas escovas. O atrito também é responsável pela necessidade de manutenção mais frequente nos DCB. Os motores AC utilizam um sistema eletrônico para controlar corrente, mas também apresentam perdas de energia para geração dos eletroímãs. Por último, os DCBL também utilizam sistema eletrônico para corrente, além de serem os mais eficientes em termos de energia e apresentarem boa estabilidade de torque/rotação. Sabe-se também que dada uma mesma potência, os DCBL tem as menores dimensões. Dadas as justificativas acima, para aplicação na interface háptica, foram escolhidos os motores DC brushless. Os cálculos e especificações dos motores serão mostrados nos próximos capítulos

Escolha do Motor	DC	DCBL	AC
Controle de Corrente		+	+
Eficiência		+	+
Tamanho		+	
Manutenção		+	
Estabilidade de Torque	+	+	

Tabela 1 – Atributos positivos e negativos dos tipos de motores.
Fonte: própria

[6] Disponível em http://www.orientalmotor.com/brushless-dc-motors/technology/AC-brushless-brushed-motors.html

Foram utilizados os motores de modelo 118889 da Maxon, que possuem as seguintes especificações:

- Torque nominal: 47,2 mNm;
- *Stall torque*: 355 mNm;
- Velocidade em vazio (*no load*): 11.000 rpm;
- Constante de torque: 20,5 mNm/A;
- Eficiência máxima: 80%.

Para um dos motores, foi utilizada uma caixa de redução planetária, modelo 166157 da Maxon, que possui redução de 5,75. Para o outro motor, foi utilizada uma caixa de redução planetária, modelo 166158 da Maxon, que possui redução de 14. Além disso, um encoder incremental de 2.000 *counts per turn* (antes da redução) foi usado em ambos os motores para referência de posição. As reduções têm valores diferentes devido à disponibilidade, mas ambas são suficientes para a proposta do projeto.

IV.iv. Driver

O princípio básico de uma interface háptica é o controle do feedback tátil que é providenciado ao usuário. Neste caso o feedback é de força, gerada por dois motores de corrente contínua. Por conseguinte, para adequar a impedância ao movimento, o torque dos motores deve ser controlado. A maneira mais simples de realizar tal tarefa é controlar a corrente disponível no motor, porém, os drivers de motor geralmente regulam apenas a voltagem oferecida. O driver EPOS2 foi escolhido, pois é um dos únicos no mercado que contém e possibilitam o controle de corrente.

Figura 7 – Driver EPOS2 70/10.
Fonte: Manual EPOS2 Hardware Reference da Maxon

O modelo utilizado foi o EPOS2 70/10, que pode trabalhar com motores de corrente contínua ou alternada. Este driver foi desenvolvido para ser gerenciado através da interface CAN, mas comporta controle por USB e RS232. Suas duas entradas CAN facilitam seu uso como “nós escravos” de uma rede. O driver EPOS2 se encarrega de fazer o controle local de posição, velocidade ou corrente do motor. Para tanto ele deve receber o tipo de controle e o valor de referência. Em casos no qual será aplicado uma camada superior de controle, este pode disponibilizar os valores instantâneos, tanto de posição, quanto de corrente, via protocolo de comunicação CAN. As limitações do dispositivo são a dependência de se trabalhar com motores da maxon (são de ótma qualidade, porém com preço acima da média) e necessidade de um driver para cada motor no sistema.

Algumas outras especificações importantes deste componente são:

- Voltagem de operação: 12 V;
- Máxima corrente contínua: 10 A;
- Máxima eficiência: 94%;
- Máxima rotação (DC): 100.000 rpm.

IV.v. Sistema Embarcado

O sistema embarcado opera entre os drivers e o computador com o Unity. Este é responsável pela camada de controle superior, no qual o valor de corrente/posição é ajustado em função das coordenadas do usuário. Também é responsável por transmitir os valores de corrente e posição ao computador responsável, tanto pelo jogo, quanto por salvar os dados para futura análise. Nesse contexto, serão descritas abaixo as possíveis soluções que foram identificadas, demonstrando todos os critérios seguidos para a tomada de decisão. Serão comparadas três alternativas adotadas inicialmente para o controle dos dados: o Arduino Mega, Raspberry Pi e Placa Cortex.

IV.v.a. Critérios de escolha do sistema embarcado

Os critérios de seleção do sistema embarcado considerados no projeto básico são: a capacidade de processamento; os periféricos desejados; as interfaces de comunicação; o custo e a facilidade de implementação. Seguindo essa linha, foram definidos os requisitos desejados e levantadas as alternativas viáveis.

IV.v.b. Requisitos do sistema embarcado

1. Core Mark Score. Como uma forma de avaliar comparativamente microcontroladores e microprocessadores, são desenvolvidos programas que

possibilitam testar o desempenho de diferentes plataformas (chamados de benchmark). Para o projeto, tendo em vista que se deseja selecionar a melhor plataforma sem a necessidade de adquirir todas as alternativas levantadas, procurou-se um Benchmark cujos resultados pudessem ser verificados na internet. Nesse contexto, optou-se pela Core Mark. Ela foi desenvolvida pela EEMBC[7], que é uma organização industrial fundada justamente para auxiliar designers de sistemas a selecionar processadores adequados para seu projeto;

2. Acesso CAN e USB. Para utilização dos drivers EPOS2, mencionados anteriormente, a melhor forma de comunicação com o microprocessador é a construção de uma rede CAN, que é rápida, flexível e confiável. A entrada USB é interessante para a programação da placa e para comunicação com o computador/plataforma de jogo;
3. Dificuldade de implementação. Tendo em vista que o trabalho de conclusão de curso deve ser elaborado em um período de tempo curto, buscou-se considerar em um primeiro momento que é relevante ao projeto um sistema embarcado cuja programação seja simples, para assim evitar que essa parte do projeto consuma um tempo de trabalho muito significativo;
4. Custo. Como explicado anteriormente, o custo final do projeto será relevante para a análise e, portanto, deverá ser considerado na seleção do sistema.

IV.v.c. Soluções propostas

O Arduino é uma plataforma de prototipagem eletrônica projetada com um microcontrolador de fácil manipulação. Possui interface de fácil programação voltada para um controle simplificado do hardware desejado. Pode operar com maiores tensões (5V) e consegue suportar e transmitir correntes mais elevadas (até 40mA). Do ponto de vista de processamento, é consideravelmente limitada, sendo poucos os casos com valores elevados (como o Due com 84MHz). Pode trabalhar com comunicação CAN por meio de uma das Shields existentes (placas utilizadas para adicionar novas funções ao Arduino). A variedade de modelos do Arduino é extremamente elevada. Foram separados alguns casos que seriam viáveis (Uno, Mega, Zero e Due), cujas informações podem ser vistas abaixo:

[7] Disponível em http://www.eembc.org/coremark/index.php

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	DC current I/O pins	USB	UART
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	40mA	Regular	4
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	20mA	Regular	1
Zero	ATSAMD21G18	3.3 V / 7-12 V	48 MHz	6/1	14/10	-	32	256	7mA	2 Micro	2
Due	ATSAM3X8E	3.3 V / 7-12 V	84 MHz	12/2	54/12	-	96	512	6mA or 9mA	2 Micro	4

Tabela 2 – Comparação entre tipos de Arduino
Fonte: arduino.cc/en/Products/Compare

Para efeito de comparação, será considerado o Arduino Mega 2560. O preço desse modelo é, segundo o site da plataforma dos EUA (site de referência: https://store.arduino.cc/usa/), $38,50. O CAN-BUS Shield da marca Seeed[8] tem um custo de $23,50.

O Raspberry Pi por sua vez é uma plataforma de placa única que pode ser considerada um “minicomputador”, com um microprocessador de elevadíssima capacidade de processamento (frequência de clock de 1,2GHz e 1GB de memória RAM). Assim, nesse quesito ele seria totalmente suficiente para o trabalho. Possui também periféricos bem interessantes em conjunto com um sistema operacional fácil de trabalhar (compatível com Linux). Contudo, os seus pinos GPIO não possuem proteção e tolerância elétrica como o Arduino, o que exige maiores cuidados em sua utilização. As especificações podem ser vistas abaixo. O modelo que foi analisado para o projeto foi o Raspberry Pi 3, uma das versões mais recentes da plataforma. Suas especificações são:

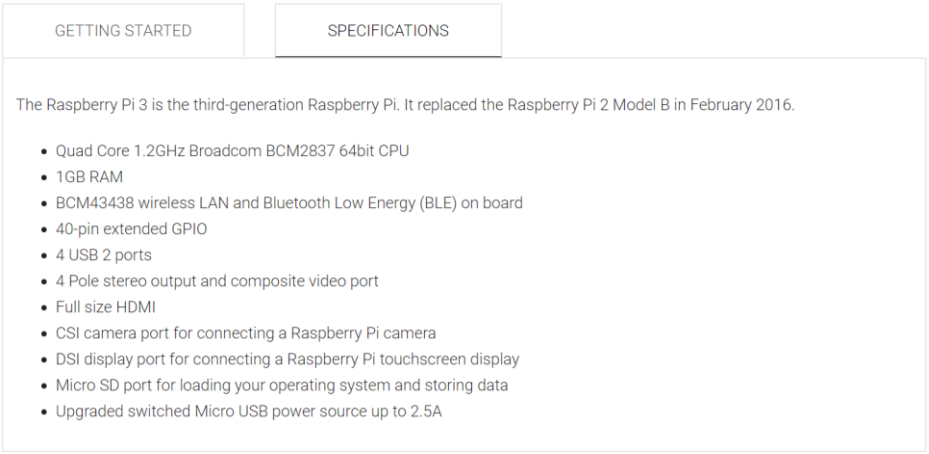


Figura 8 – Características do Raspberry Pi 3
Fonte: raspberrypi.org

O preço[9] de um Raspberry Pi 3 é de $35,00 - $39,95.

Por último, a placa Cortex possui o processador da família ARM Cortex M4. De maneira geral, ela possui todos os requisitos verificados anteriormente. Apresenta uma

[8] Disponível em: https://www.seeedstudio.com/CAN-BUS-Shield-V1.2-p-2256.html
[9] Disponível em: https://www.element14.com/community/community/raspberry-pi

variedade muito grande de kits disponíveis, sendo possível explorar uma grande diversidade de periféricos. Além disso, tem como grande vantagem a versatilidade, já que permite a utilização de inúmeros tipos de placas da mesma família sem que haja qualquer problema de compatibilidade.

Dentre as inúmeras variedades de microcontroladores disponíveis com processador da família ARM Cortex M4, foi escolhido o kit 32F429IDISCOVERY, cujo processador é o STM32F429ZI. Suas características podem ser observadas abaixo:

- ARM CORTEX-M4
- Frequência de operação = 180 MHz
- Memória FLASH = 2048 KB
- Memória RAM = 256 KB
- Portas CAN (2)
- Entrada USB
- Interface adicional = Ethernet
- Voltagem MAX 3,6V e MIN 1,8V

32F429IDISCOVERY
6 LEDs (1 USB, 2 Power, 3/4 User)
2 Botões (User e Reset)
Tela LCD 2.4"
Sensor de medição angular de 3 eixos (giroscópio)
Fornecimento de energia via USB bus ou voltagem externa
64Mbit SDRAM
USB OTG
ST-LINK/V2-1 debugger/programmer
COM port, mass storage, debug port
MCUs Embedded Software - STM32cubeF4 (RTOS)

Figura 9 – Características da Placa 32F429IDISCOVERY
Fonte: st.com

Vale destacar que existem muitos kits compostos por processadores da série STM32F4 que apresentam características semelhantes. Foi escolhida essa placa em específico porque, além de possuir especificações compatíveis com os requisitos do projeto, esta contem periféricos interessantes como uma tela de LCD onde se deseja, mais tarde, apresentar informações relativas à atividade que será realizada pelo paciente (como a força/torque que o paciente exerce no mecanismo). O preço do produto, segundo o site de vendas Digi-Key[10] é $29,90.

[10] Disponível em: https://www.digikey.com

IV.v.d. Matriz de Decisão

A matriz foi construída com base nas alternativas e nos requisitos levantados acima. O peso foi estipulado com base na necessidade do projeto.

Parâmetros	Peso	Arduino	Raspberry Pi 3	Placa Cortex
Core Mark	3	1	3	2
CAN/USB	4	3	1	2
Implementação	2	3	1	2
Custo	1	1	2	3
Total		22	17	21

Tabela 3 – Matriz de decisão dos microcontroladores.
Fonte: própria.

Na análise, obteve-se grande proximidade entre os resultados da board 32F429IDISCOVERY e do Arduino Mega**.** Logo, por uma questão de disponibilidade e facilidade de implementação, optou-se por utilizar para o protótipo do mecanismo o Arduino Mega 2560.

V. Projeto Mecânico

V.i. Soluções Propostas

Para escolha da solução mecânica, foi feito um "*brainstorm*" de mecanismos, baseado nos produtos existentes no mercado e em conhecimento prévio. Foram pré-selecionados três mecanismos com atributos e abordagens diferentes. Todas as soluções foram definidas de modo que tivessem dois graus de liberdade. As três soluções foram as seguintes:

1. Mecanismo de cadeia aberta (MCA)

Este mecanismo é similar ao utilizado no projeto OSHap, mencionado anteriormente. Por ser de cadeia aberta, ele possui maior área de trabalho, mas alcança menores velocidades e tem menor capacidade de carga. Neste caso, os dois atuadores estão fixados à base, de modo que o torque é transmitido por uma correia, a fim de que a inércia seja reduzida. Pode-se dizer que devido ao maior alcance, também estão mais sujeitos a folgas e, portanto, podem amplificar ruídos.

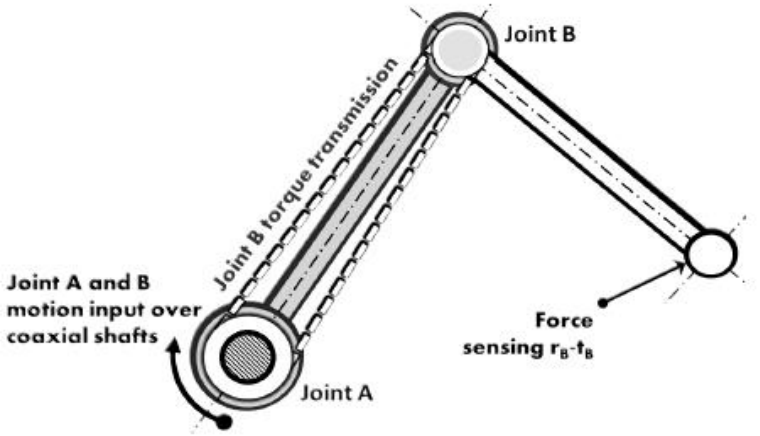


Figura 10 – Projeto do dispositivo OSHap.
Fonte: LAVATELLI, FERRISE e BORDEGONI, 2014 página 2

2. Mecanismo de cadeia fechada de 5 barras (MCF5)

Este mecanismo é similar ao utilizado no projeto Pantograph MK, mencionado anteriormente. Uma vez que os eixos dos motores não são coincidentes, este foi chamado de 5 barras, apesar de apresentar 4. Por ser de cadeia fechada, ele possui menor área de trabalho, mas alcança maiores velocidades e tem maior capacidade de carga. Neste caso, os dois atuadores estão fixados à base, de modo que os eixos dos motores são paralelos, mas não coincidentes. Esta solução implica numa grande complexidade matemática, tanto na análise cinemática, quanto na dinâmica.

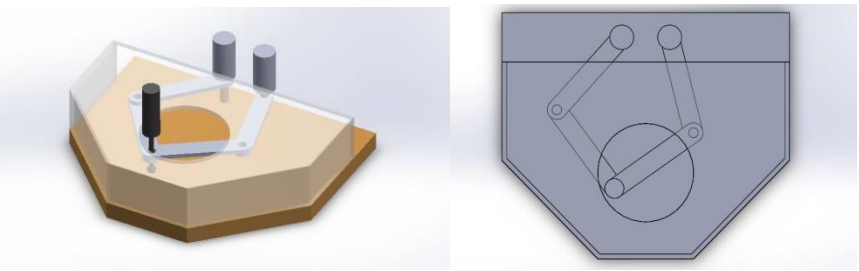


Figura 11 – Projeto preliminar e vista superior do dispositivo de 5 barras.
Fonte: própria

3. Mecanismo de cadeia fechada de 4 barras (MCF4)

Este mecanismo é similar aos utilizados nos projetos MIT-Manus e InMotion ARM, mencionados anteriormente. Por ser de cadeia fechada, ele possui menor área de trabalho, mas alcança maiores velocidades e tem maior capacidade de carga. Neste caso, os dois atuadores estão fixados à base, de modo que os eixos dos motores são coincidentes.

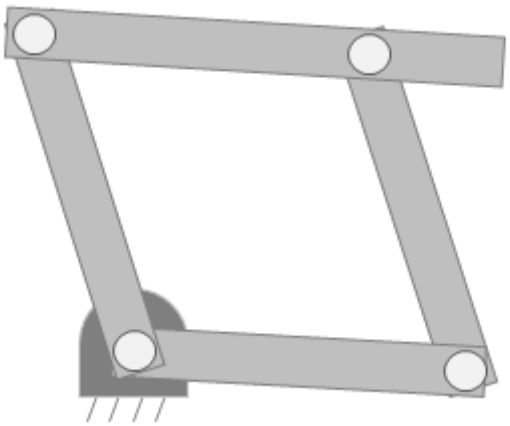


Figura 12 – Vista superior do dispositivo de 4 barras.
Fonte: própria

V.ii. Critérios e Pesos

Para seleção da solução, foram utilizados os seguintes critérios e seus respectivos pesos:

1. Área de trabalho (peso 2): este critério limita a quantidade de diferentes exercícios/jogos e a extensão dos movimentos que o mecanismo será capaz de proporcionar;
2. Inércia (peso 4): tal atributo aumenta a exigência de torque e potência dos motores, além de dificultar o controle do dispositivo;
3. Velocidade de movimento (peso 1): o mecanismo trabalha com velocidades de até 1m/s;

4. Capacidade de carga (peso 5): para realizar fisioterapia, a capacidade de gerar forças é mais importante que a velocidade. Com uma capacidade de carga maior, será possível utilizar o dispositivo em um número maior de tipos de pacientes;
5. Necessidade de processamento (peso 3): o sistema final será composto de diversos módulos (controle, interface de jogo, banco de dados e etc) ativos simultaneamente e, por isso, a economia de processamento é fundamental.

V.iii. Escolha da Solução

Após atribuir notas de 1 a 3 para cada solução, através de cada critério, foi formada a matriz de decisão abaixo:

Parâmetros	Peso	MCA	MCF5	MCF4
Área de Trabalho	2	3	1	1
Inércia	4	2	3	3
Velocidade	1	2	3	3
Capacidade de Carga	5	1	3	3
Processamento	3	3	2	3
Total		30	38	41

Tabela 4 – Matriz de decisão do mecanismo.
Fonte: própria

Como pode ser observado na matriz acima, a solução em cadeia aberta pode ser facilmente descartada devido à sua menor velocidade e capacidade de carga e também à implicação de uma maior inércia do mecanismo. As notas finais dos mecanismos em cadeia fechada foram próximas, o que demandou uma análise mais profunda das soluções. Nenhuma das duas soluções se diferencia/destaca nos 4 primeiros critérios. Pode-se dizer que a álgebra envolvida no MCF5 é consideravelmente mais complexa que a do MCF4. Isto implica que o MCF5 demanda um tempo maior de processamento quando no loop de controle e está mais sujeito a falhas de projeto e de operação. Esta justificativa foi considerada suficiente para que o mecanismo MCF4 fosse escolhido.

V.iv. Detalhamento da Solução

A solução escolhida foi baseada no dispositivo MIT-Manus, descrita previamente. O mecanismo escolhido é bidimensional de cadeia fechada, com 5 elos (incluindo a base fixa) e 5 juntas rotativas. O dispositivo foi projetado em CAD (considerando o dimensionamento descrito a seguir) e os principais desenhos de conjunto encontram-se na seção de apêndices.

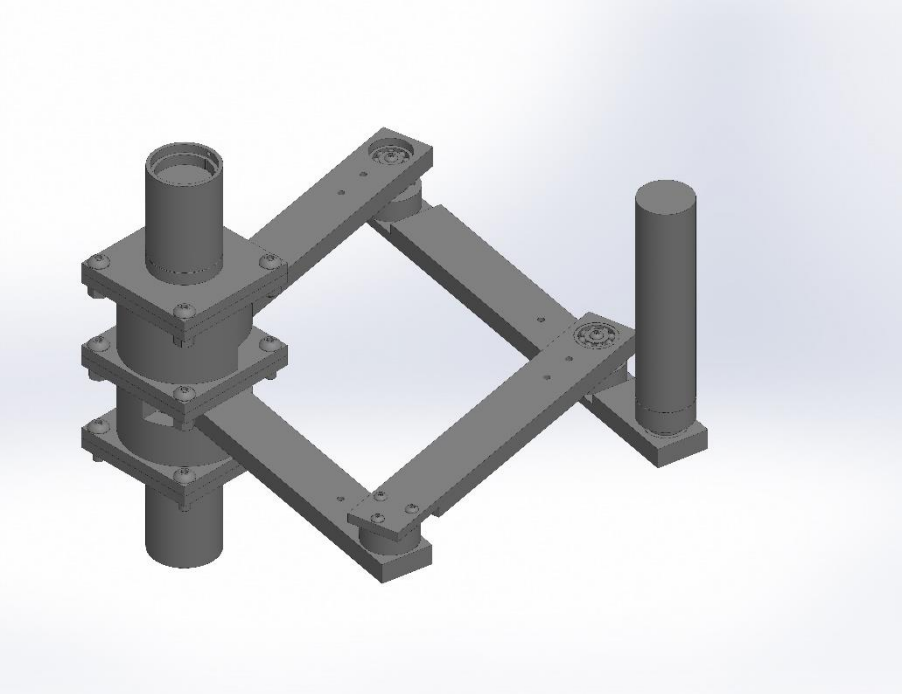


Figura 13 – Vista tridimensional do mecanismo modelado.
Fonte: própria

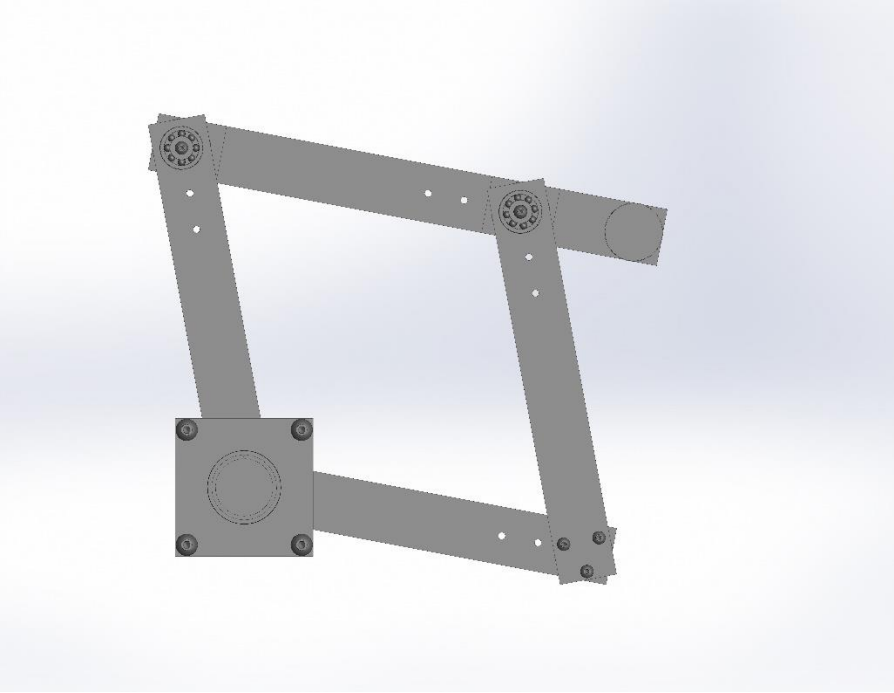


Figura 14 – Vista superior do mecanismo modelado.
Fonte: própria

Para provar que o mecanismo possui os dois graus de liberdade desejados, podemos utilizar o critério de Kutzbach-Grubler:

$$GDL = 3(B - 1) - 2n_{j1} - n_{j2} = 2$$

Onde:

- GDL: graus de liberdade
- $B = 5$: número de elos
- $n_{j1} = 5$: número de juntas com 1 GDL
- $n_{j2} = 0$: número de juntas com 2 GDL

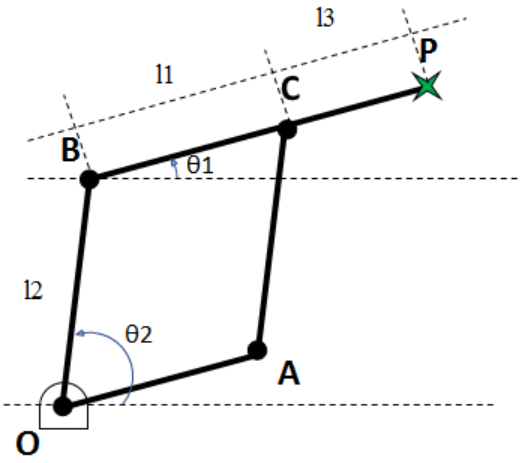


Figura 15 – Desenho do mecanismo para referência de cálculos da cinemática direta.
Fonte: própria.

Com dimensões genéricas, realizou-se a análise cinemática completa. Seguindo o modelo acima, temos para a equação de posição na análise direta:

$$(B - O) + (P - B) + (O - P) = \vec{0}$$

$$\text{Em x: } \cos(\theta_2)\,\overline{BO} + \cos(\theta_1)\,\overline{PB} = X_P$$

$$\text{Em y: } \text{sen}(\theta_2)\,\overline{BO} + \text{sen}(\theta_1)\,\overline{PB} = Y_P$$

Na forma matricial, teremos:

$$\begin{bmatrix} X_P \\ Y_P \end{bmatrix} = \begin{bmatrix} \cos(\theta_1) & \cos(\theta_2) \\ \text{sen}(\theta_1) & \text{sen}(\theta_2) \end{bmatrix} * \begin{bmatrix} \overline{PB} \\ \overline{BO} \end{bmatrix}$$

Para encontrar as equações de velocidade, basta derivar as equações de posição desenvolvidas acima, no tempo:

$$\begin{bmatrix} \dot{X}_P \\ \dot{Y}_P \end{bmatrix} = \begin{bmatrix} -\overline{PB}\,\text{sen}(\theta_1) & -\overline{BO}\text{sen}(\theta_2) \\ \overline{PB}\,\cos(\theta_1) & \overline{BO}\,\cos(\theta_2) \end{bmatrix} * \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}$$

Para encontrar as equações de aceleração, basta derivar as equações de velocidade acima, no tempo:

$$\begin{bmatrix} \dot{X}_P \\ \dot{Y}_P \end{bmatrix} = \begin{bmatrix} -\overline{PB}\,\text{sen}(\theta_1) & -\overline{BO}\text{sen}(\theta_2) \\ \overline{PB}\,\cos(\theta_1) & \overline{BO}\,\cos(\theta_2) \end{bmatrix} * \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} -\overline{PB}\cos(\theta_1) & -\overline{BO}\cos(\theta_2) \\ -\overline{PB}\text{sen}(\theta_1) & -\overline{BO}\,\text{sen}(\theta_2) \end{bmatrix} * \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix}$$

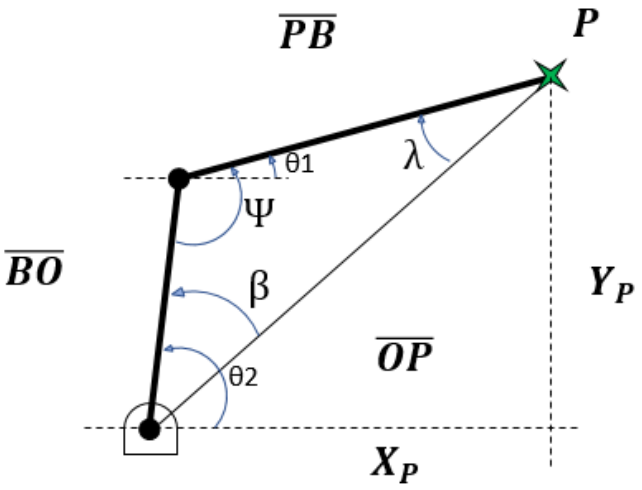


Figura 16 – Desenho do mecanismo para cálculos da cinemática inversa.
Fonte: própria.

Para a análise cinemática inversa, fixa-se um ponto (X_P, Y_P), e deseja-se saber o ângulo de entrada dos atuadores naquela posição. Para resolver este problema, utiliza-se a lei dos cossenos:

$$\varphi = \cos^{-1}\left(\frac{\overline{PB}^2 + \overline{BO}^2 - \overline{OP}^2}{2.\overline{PB}.\overline{BO}}\right)$$

$$\lambda = \cos^{-1}\left(\frac{\overline{PB}^2 + \overline{OP}^2 - \overline{BO}^2}{2.\overline{PB}.\overline{OP}}\right)$$

$$\beta = \cos^{-1}\left(\frac{\overline{OP}^2 + \overline{BO}^2 - \overline{PB}^2}{2.\overline{OP}.\overline{BO}}\right)$$

Após alguma manipulação algébrica:

$$\theta_1 = \varphi - \pi + \beta + \cos^{-1}\left(\frac{X_P}{\overline{OP}}\right)$$

$$\theta_2 = \beta + \cos^{-1}\left(\frac{X_P}{\overline{OP}}\right)$$

Com a análise cinemática completa, pode-se realizar a análise dinâmica. Visto que as velocidades impostas ao mecanismo serão baixíssimas, que o momento de inércia dos elos é relativamente pequeno (peças de alumínio com no máximo 150 mm) e que não é necessária precisão nos cálculos de forças e torques, pode-se dizer que a análise estática do mecanismo é suficiente para o escopo do projeto.

Para tanto, foi feito o diagrama de corpo livre para cada um dos 4 elos não-fixos. Isto gerou um sistema linear definido, com 12 equações e 12 incógnitas. O objetivo da análise foi encontrar a relação dos torques nos motores com as forças aplicadas na extremidade do mecanismo (manopla), eliminando assim, as forças reativas nas juntas

rotativas. Resolvendo o sistema, percebeu-se que os torques nos motores são independentes entre si e dependem apenas das forças na manopla e do ângulo de seus respectivos elos atuados. As equações dinâmicas encontradas para o mecanismo são as seguintes:

$$T_1 = sen(\theta_1)(l_1 + l_3)F_{PX} - cos(\theta_1)(l_1 + l_3)F_{PY}$$
$$T_2 = sen(\theta_2)(l_2)F_{PX} - cos(\theta_2)(l_2)F_{PY}$$

O próximo passo foi definir a área útil do dispositivo, para que fosse possível definir as dimensões dos elos dos mecanismos. Vale lembrar que neste projeto, quanto mais extensos forem os elos, maior será o momento na manopla e, por conseguinte, maior será a exigência nos motores. Por esses motivos, para o jogo, decidiu-se por um uma plataforma circular, com diâmetro de 150mm, que é suficientemente grande para realizar movimentos bem definidos, sem exigir torques demasiadamente grandes dos motores. Nota-se também que a comprimento l_3, não precisa ser tão longo quanto l_1 e l_2, pois sua dimensão apenas translada e amplifica a área útil. Reduzindo l_3, podemos diminuir os requisitos de torque dos motores. Para satisfazer a área útil desejada, decidimos trabalhar com as seguintes dimensões de elos, que a comportam com certa folga:

$$l_1 = 150mm, l_2 = 150mm \; e \; l_3 = 50mm,$$

Foi feito então um programa em Matlab® para simular a área total do sistema, onde será possível posicionar a manopla, primeiramente sem considerar as limitações físicas (ambos ângulos variando de 0º a 360º) do mecanismo. Os principais códigos desenvolvidos encontram-se na seção de apêndices. O resultado obtido é o seguinte:

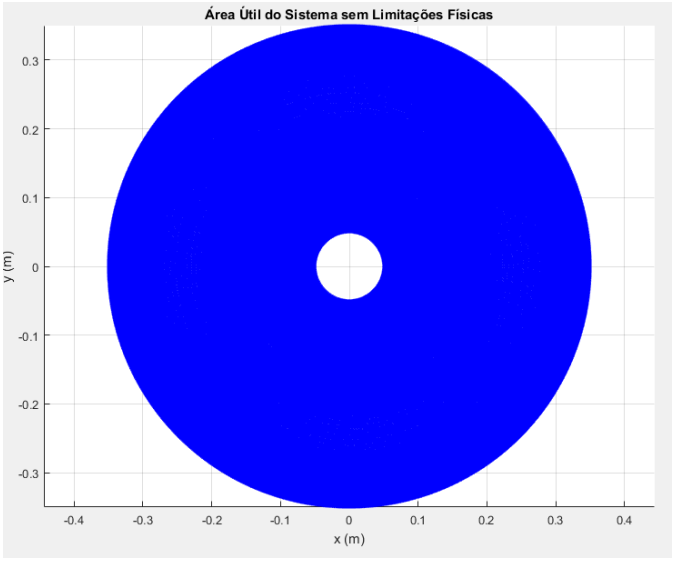


Figura 17 – Área total disponível, obtida em simulação.
Fonte: própria.

Como se pode observar, a área total é um anel com circunferência externa de 350mm e circunferência interna de 50mm, como era esperado. Entretanto, o mecanismo projetado possui limitações físicas, como será mostrado posteriormente. Estas limitações podem ser explicadas por ângulos mínimos e máximos de saída dos motores. Para este projeto, consideraremos apenas os seguintes intervalos de ângulos:

$$-40º \leq \theta_1 \leq 40º \; e \; 50º \leq \theta_2 \leq 130º$$

Definido o intervalo de trabalho, pode-se encontrar a área útil física, utilizando o mesmo programa mencionado acima. Para este intervalo o resultado é o seguinte:

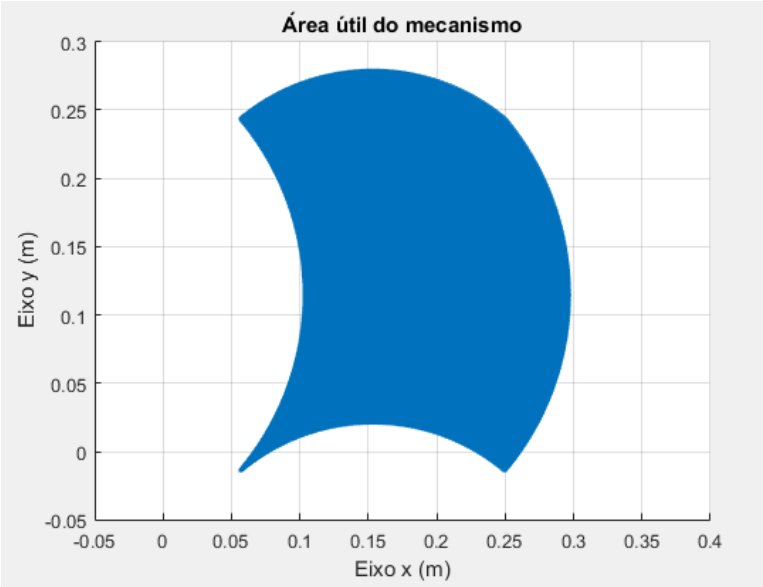


Figura 18 - Área útil disponível, obtida em simulação.
Fonte: própria.

Sabendo onde será possível posicionar a manopla de fato, é possível escolher as coordenadas da plataforma do jogo, ou seja, o espaço onde ficará contida a manopla durante o exercício. Como dito anteriormente, decidiu-se por utilizar uma circunferência com 150mm de diâmetro por causa de sua compatibilidade com o jogo que seria desenvolvido. Tal circunferência deverá estar contida na área útil mostrada acima, incluindo certa folga para que erros de fabricação/ construção não comprometam o projeto. Vale lembrar que se a área de trabalho for deslocada para a origem, ou seja, para mais próxima dos motores, o torque exigido nos motores será menor. O ponto (0,20; 0,15) como centro da circunferência de área útil, como mostra a imagem abaixo.

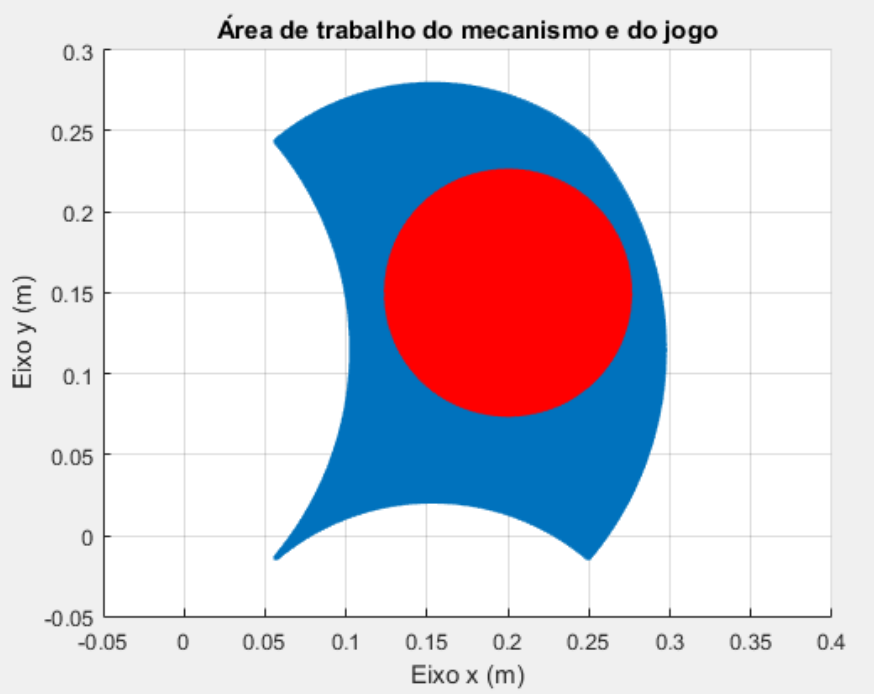


Figura 19 - Área de trabalho definida para o jogo.
Fonte: própria.

A partir da definição da área de trabalho, foi feita a simulação dinâmica do mecanismo, com o objetivo de se obter as curvas torque-rotação dos dois motores a serem usados no projeto. Para tanto, foi fixado um valor máximo para a força que o usuário aplicará na manopla e uma velocidade máxima deste mesmo ponto. Utilizamos para a simulação $F_{max} = 40\ N$ e $v_{max} = 1\ m/s$.

Foi feito um programa em Matlab para calcular os torques e velocidades angulares necessárias no motor para resistir à força e velocidade máximas, fixadas acima. O programa percorre a área de trabalho na forma de uma espiral, partindo do centro da circunferência. Cada ponto da espiral é usado para calcular o ângulo de saída dos motores (elos) a partir da cinemática inversa. Com cada par de ângulos calculados, a força e velocidade máximas são impostas com diferentes direções (ângulos). Por fim, todos os pontos são desenhados em gráficos para obter as curvas torque-rotação e potência-rotação para os motores em sua máxima exigência de torque e velocidade. Os resultados são mostrados a seguir.

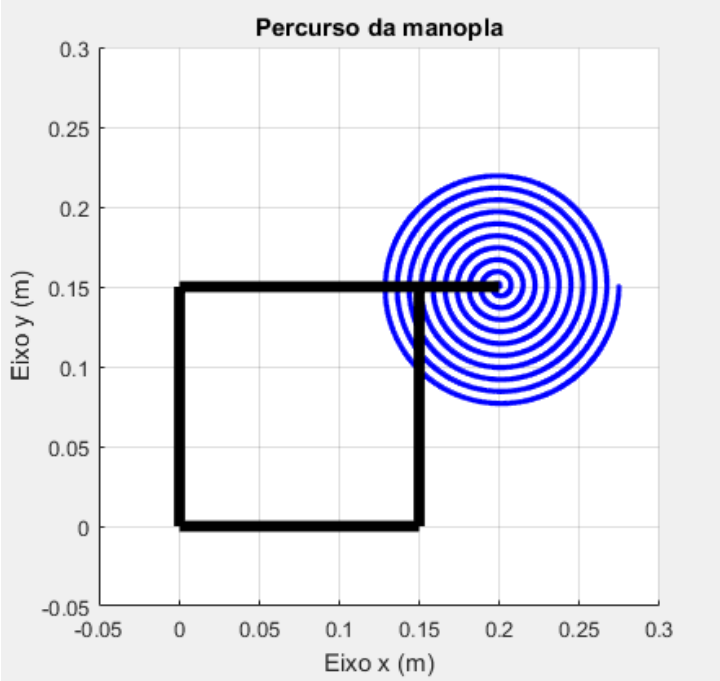


Figura 20 – Percurso realizado pela manopla na simulação.
Fonte: própria.

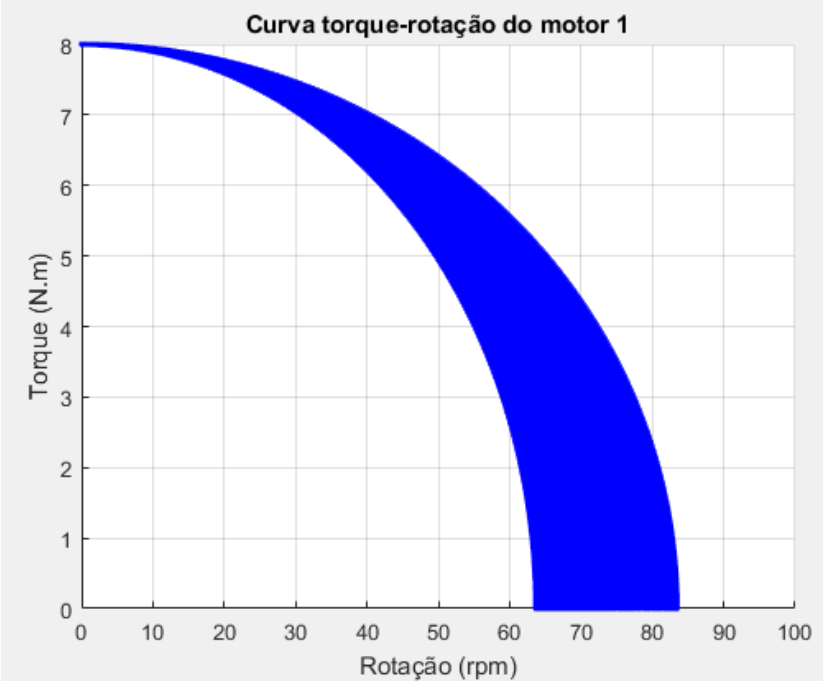


Figura 21 – Curva torque-rotação desejada para motor 1 na simulação.
Fonte: própria.

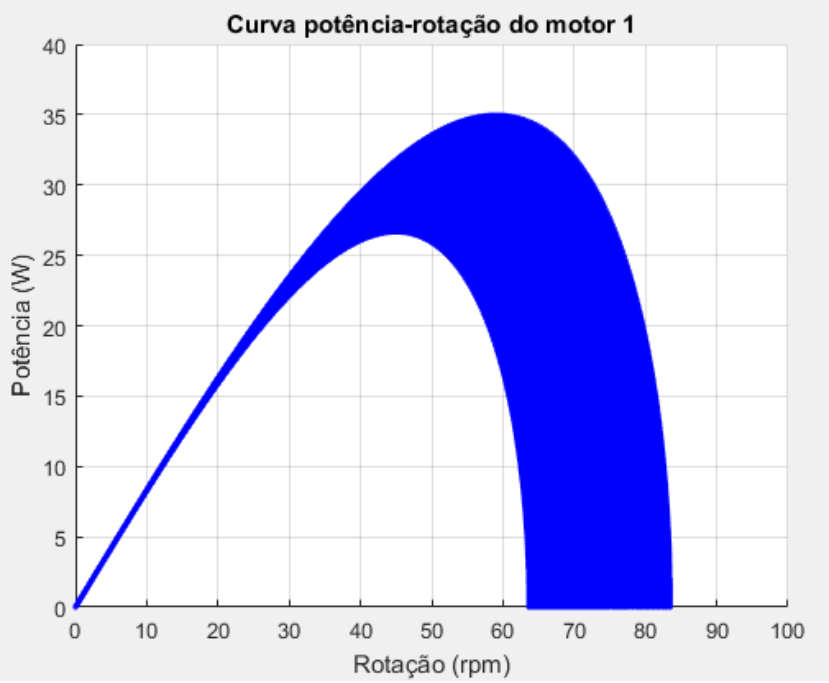


Figura 22 - Curva potência-rotação desejada para o motor 1 na simulação.
Fonte: própria.

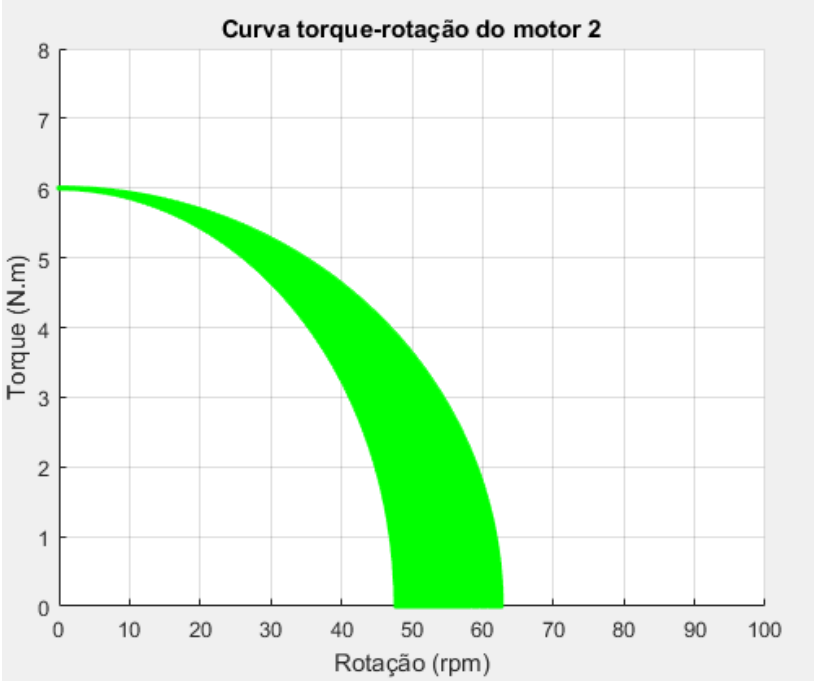


Figura 23 - Curva torque-rotação desejada para o motor 2 na simulação.
Fonte: própria.

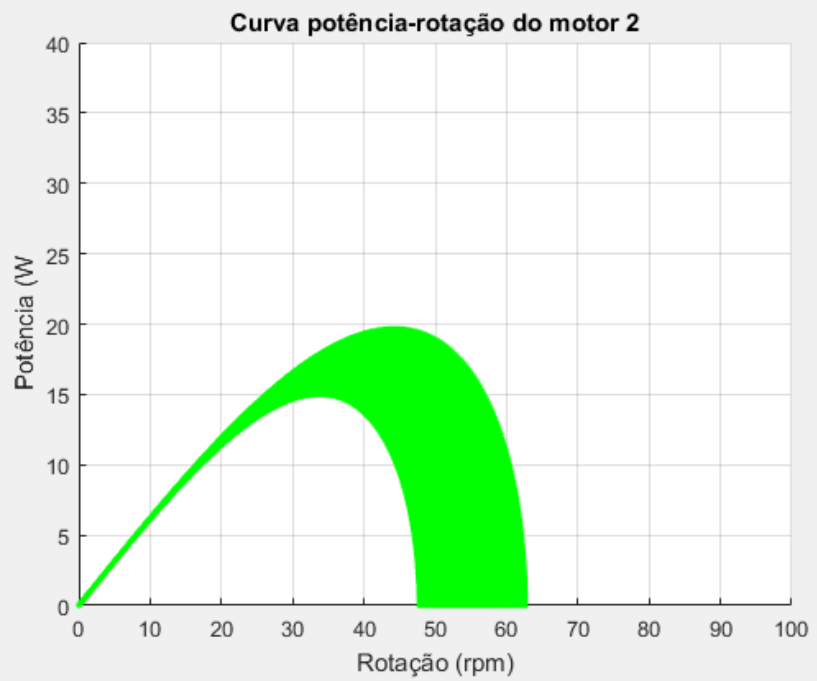


Figura 24 – Curva potência-rotação desejada para o motor 2 na simulação.
Fonte: própria.

Foi decidido que serão utilizados motores Maxon, os únicos compatíveis com o driver EPOS2, que por sua vez foi o único driver com controle de corrente (e portanto torque) encontrado. O site da Maxon possui um programa de seleção de sistemas de atuação, onde são escolhidos os parâmetros desejados quanto a carga, fonte, redução e tipo de motores. De acordo com o projeto básico, a busca foi filtrada por: motor DC brushless; encoder embutido; redução acoplada. Por fim, foram utilizados os seguintes parâmetros de busca, compatíveis com os cálculos realizados e demonstrados acima:

$$Torque: 8\ N.m \text{ e } Rotação: 100\ rpm$$

$$Potência: 35\ W\ (24V)$$

Os motores ideais para o projeto do dispositivo seriam os seguintes:

Products		Technical data ø [mm]	Length [mm]	Load [%]	Current [A]	Price	
▷	Motor RE 50 GB Gearhead GP 52 C, 53:1 Encoder HEDS, 500 cpt, 3°K	52	186.6	91	4.81	CHF 960	Details
▷	Motor RE 50 GB Gearhead GP 52 C, 21:1 Encoder HEDS, 500 cpt, 3°K	52	173.1	96	4.39	CHF 923.8	Details
▷	Motor EC 90 flat Gearhead GP 52 C, 19:1 Encoder MILE, 800 cpt, LD, 2°K	90	96.4	81	4.42	CHF 720.9	Details

Figura 25 – Motores ideais para o projeto e seu preço.
Fonte: loja virtual da Maxon(11).

[11] Disponível em https://www.maxonmotor.com/maxon/view/product/118889

Estes motores já possuem módulo de redução mecânica e encoder acoplados, mas têm preço em torno de CHF 900 (aproximadamente R$ 3 mil em 15/11/2017), antes dos impostos. Além do alto preço, os componentes demorariam alguns meses para entrega. Por se tratar de um protótipo, o grupo aceitou trabalhar com os motores e reduções disponíveis no laboratório, incorrendo em menores torques/forças máximas exercidas pelo dispositivo. Para os motores usados, segundo os datasheets, os torques máximos serão:

	Redução	Máx. Torque Contínuo	Stall Torque
Motor + redução 1	5,75	0,66 Nm	2,04 Nm
Motor + redução 2	14,00	0,27 Nm	4,97 Nm

Tabela 5 – Especificações de torque dos atuadores utilizados.
Fonte: própria

V.v. Fabricação e Montagem

Foram fabricadas 18 peças, sendo que 10 peças diferentes foram projetadas. Uma vez que as forças no mecanismo serão relativamente baixas (cerca de no máximo 15 N), foi possível fabricar todas as partes a partir de alumínio, assim possibilitando diminuir a massa do protótipo, a inércia do mecanismo e o custo das peças. A fabricação das peças foi feita apenas com métodos tradicionais de usinagem (torneamento, fresagem, corte de serra, guilhotina e outros), visando diminuição no custo de produção. Para fixação das peças, foram utilizados parafusos de aço Allen M3 e M5 de cabeça abaulada.

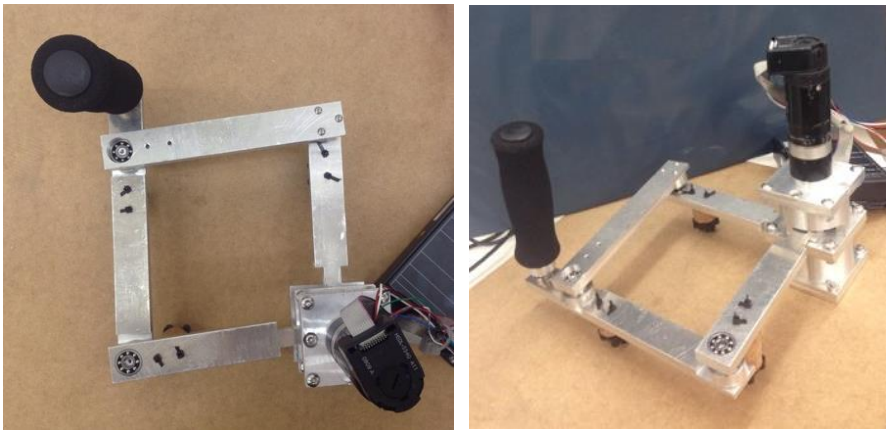


Figura 26 (a) e (b) – Mecanismo fabricado e montado, em vista superior e tridimensional.
Fonte: própria

As juntas rotativas foram implementadas a partir de rolamentos autocompensadores (2x NSK 2203 e 4x NSK 126), de modo a diminuir folgas e travamentos que podem prejudicar os exercícios fisioterápicos. Este tipo de rolamento é similar a uma junta esférica, introduzindo dois graus de liberdade adicionais, possibilitando a realização de movimentos com o pulso e garantindo movimentos mais suaves. A fixação dos rolamentos foi feita exclusivamente a partir de ajuste mecânico de precisão com interferência, do tipo K7 (ajuste aderente duro) para o sistema eixo-base (parte exterior dos rolamentos) e k6 para o sistema furo-base (parte interior dos rolamentos). Este tipo de ajuste é utilizado para peças de acoplamento fixo, sem desmontagem frequente. O desacoplamento é feito a partir de golpes de martelo.

Figura 27 (a) e (b) – Rolamento autocompensador (a) e esfera deslizante (b).
Fontes: respectivamente lojas virtuais da KQYK(12) e Robocore(13)

Para evitar que a gravidade influencie no movimento (principalmente porque todos os rolamentos são autocompensadores), foram utilizadas três esferas deslizantes para apoio. As esferas foram escolhidas devido à necessidade de redução do atrito gerado nos apoios. O apoio é fundamental para que o usuário não faça movimentos significativos na direção Z, uma vez que o dispositivo só identifica movimentos em nas direções X e Y.

Visando proporcionar conforto no exercício, foi utilizada uma manopla de espuma no mecanismo e uma de apoio. Por se tratar de um protótipo, o dispositivo foi montado numa base de madeira mdf, sem grandes preocupações com estética.

Para testar o aumento da portabilidade do dispositivo, o equipamento completo (incluindo fonte de energia, placas e outros) foi pesado numa balança comum, chegando a uma massa de aproximadamente 8kg.

[12] Disponível em http://pt.kqyqbearings.com/self-aligning-bearing/41611484.html
[13] Disponível em https://www.robocore.net/loja/produtos/esfera-deslizante-robocore.html

VI. Projeto de Controle

O protótipo possibilita o uso de dois tipos diferentes de controle: o de corrente e por consequência de torque gerado nos motores (também chamado de controle de impedância) e o de posição da manopla. O tipo de controle poderá ser escolhido para que o dispositivo se adeque ao exercício para o qual ele será utilizado. O dispositivo de interface háptica desenvolvido neste projeto poderá ser utilizado no futuro para que a eficiência e eficácia de ambos os modos sejam estudados.

Além da opção de modo de controle, foram criadas duas classes de exercício:

- Iniciante, quando os motores são usados para ajudar o usuário a se manter no caminho correto (recomendado para iniciantes e pacientes com deficiências graves);
- Avançado, quando os motores são usados para atrapalhar o usuário e tirá-lo do caminho correto (recomendado para usuários experientes ou pacientes com nível baixo de deficiência);

Todas as classes de exercício podem ser aplicadas ao paciente com intensidades diferentes, bastando adicionar um fator multiplicativo à corrente a ser disponibilizada aos motores. Caso a deficiência seja grande, a intensidade pode ser maior no exercício Iniciante e menor no Avançado. Para deficiências pequenas, vale o contrário. O propósito da variedade de exercícios é flexibilizar o dispositivo, de modo que ele possa ser utilizado em diversos tipos de tratamento.

VI.i. Modelo da Planta

O modelo do sistema da planta para o controle de posição para um motor está representado abaixo, nas cores azul e vermelha. Primeiramente, o comando de posição é interpretado e o driver gera um trajeto de pontos. O trajeto contém os setpoints de posição (ângulo) no motor. O controlador de posição é implementado a partir de um PID, que utiliza a referência de posição e o valor atual para gerar uma referência de corrente. Esta referência, juntamente com o valor de corrente atual disponibilizada, é usada pelo controlador de corrente para selecionar dinamicamente o duty cycle no amplificador de potência. Este controlador, por sua vez, é implementado a partir de um PI. Por fim, o estágio de amplificação gera um sinal PWM (com duty cycle definido pelo controlador) para acionar o motor DC.

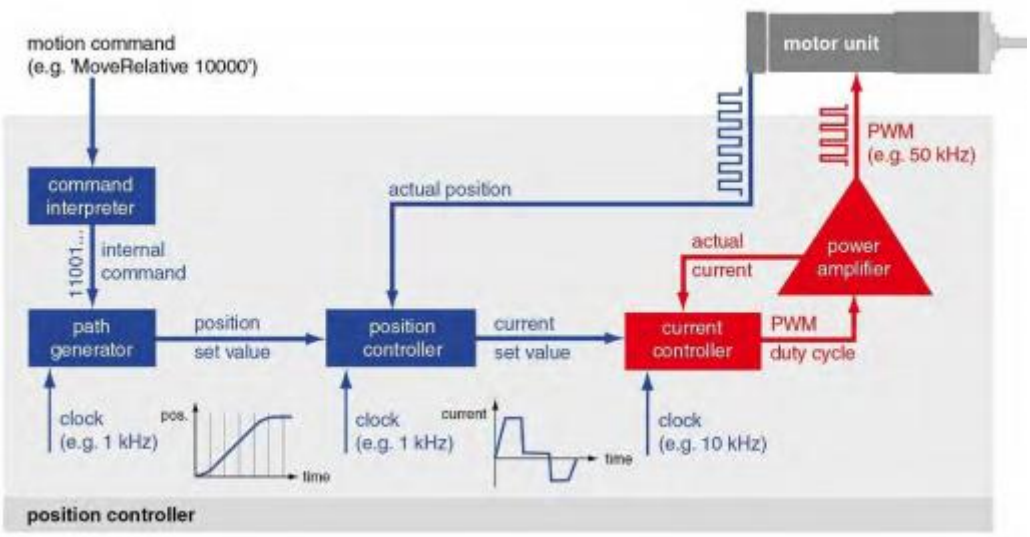


Figura 28 – Modelo do sistema de controle implementado pelos driver EPOS2 da Maxon.
Fontes: KAFADER, Urs. Motion Control for Newbies. Maxon Academy, 2007. Página 39.

O modelo do sistema da planta para o controle de corrente utiliza a mesma lógica descrita acima, porém a referência de corrente é imposta pelo código do microprocessador. Na representação acima, o modelo de controle de corrente corresponde apenas aos blocos vermelhos.

O driver de motor EPOS2 70/10 possui um aplicativo de configuração e testes chamado "EPOS Studio", desenvolvido para ajustar, estudar e debugar o componente eletrônico em diversas aplicações. Os controladores PI (corrente) e PID (posição) são implementados através de software e seus ganhos podem ser definidos pelo usuário. O aplicativo mencionado disponibiliza uma ferramenta chamada de "Regulation Tuning", que realiza testes no sistema motor/redução a ser usado, a fim de ajustar os ganhos dos controladores. A figura seguinte mostra os resultados deste processo, mostrando o input e o output dos sistemas de controle de corrente e de posição. No gráfico de corrente, o intervalo de escala de tempo é de 1 ms, enquanto que no de posição, o intervalo é de 10 ms. Como se pode observar, os tempos de assentamento de ambos os sistemas de controle são muito pequenos, quando comparados ao período de atualização do sistema, que é de $1/60Hz = 16{,}67\ ms$. Através dos gráficos, também se pode inferir que não há overshoot significativo nem erro estático relevante.

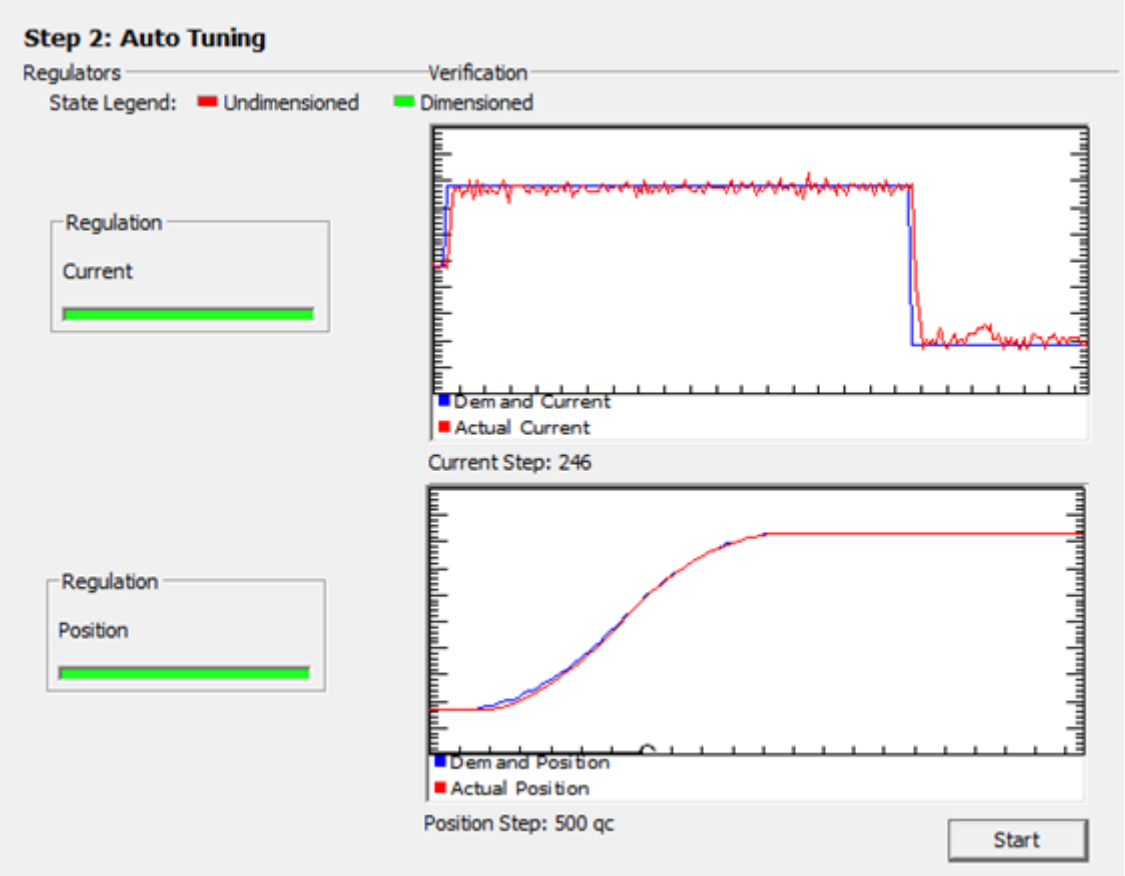


Figura 29 – Tela da ferramenta de auto-tuning.
Fonte: Software de posicionamento da Maxon

Para o sistema motor 1/redução planetária 1, a ferramenta calculou os melhores ganhos como:

- Corrente: $PI(s) = 434 + \frac{108}{s}$
- Posição: $PID(s) = 817 + \frac{6201}{s} + 574s$

Para o sistema motor 2/ redução planetária 2, a ferramenta calculou os melhores ganhos como:

- Corrente: $PI(s) = 454 + \frac{127}{s}$
- Posição: $PID(s) = 288 + \frac{1276}{s} + 336s$

VI.ii. Controle de Torque e Corrente (Impedância)

O modo de impedância gera uma corrente/torque proporcional ao desvio entre a posição imposta pelo usuário e o trajeto ideal, utilizando o conceito de campo de forças. Para este modo, consideramos que a relação entre torque e corrente é constante, definida pela constante de torque K_t do motor/redução.

Foi realizado um experimento para obter a curva torque versus corrente dos motores. Foram usados pesos predefinidos, gerando um torque conhecido, que deveria

ser igualado pelo motor, através da imposição de correntes. Foram registrados os valores mínimos (acima do qual o motor igualava o peso e, portanto o torque) e o máximo (onde o motor superava o torque) de corrente, dentre os quais os pesos ficavam parados. Essa diferença surge principalmente devido ao atrito e ao "*backlash" (folga)*. Foi considerada, então, a regressão linear da reta média entre a máxima e a mínima.

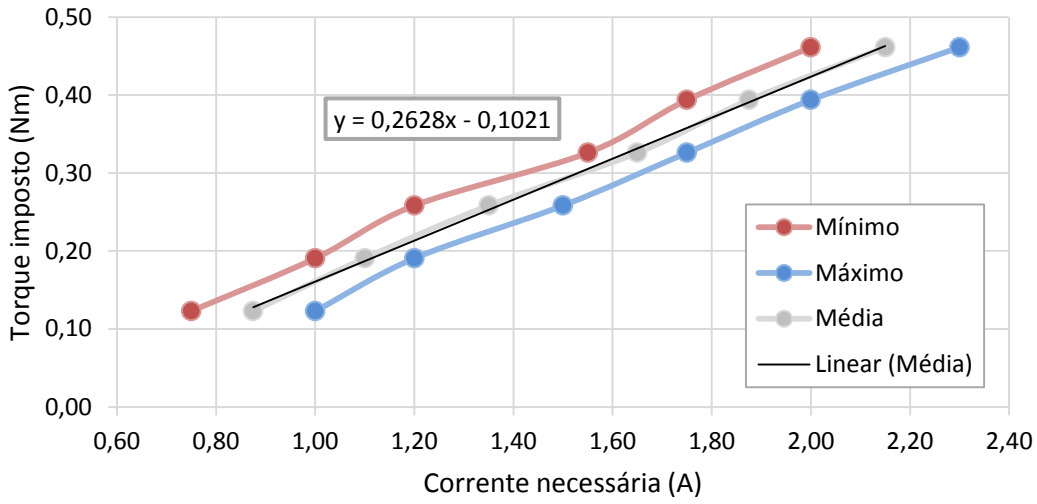


Figura 30 – Curvas Torque (Nm) versus Corrente (A) encontrada para o motor 1 (acoplado à redução 1, de 5,75:1).
Fonte: própria

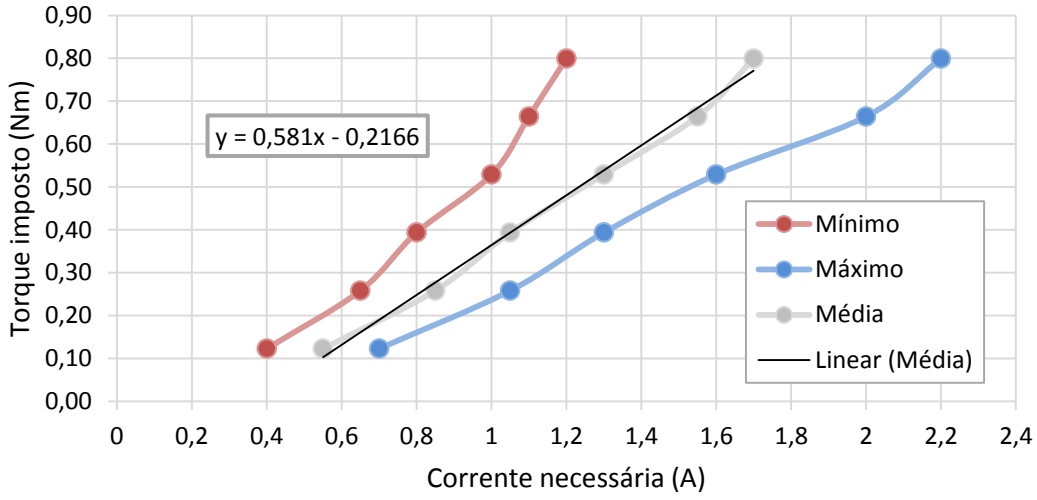


Figura 31 – Curvas Torque (Nm) versus Corrente (A) encontrada para o motor 2 (acoplado à redução 2, de 14:1).
Fonte: própria

Como pode ser observado nos gráficos:

$$T_{m1}\ [Nm] = 0{,}2628\ I\ [A] - 0{,}1021$$

$$T_{m1}\ [Nm] = 0{,}5810\ I\ [A] - 0{,}2166$$

Para implementação deste modo de controle, o módulo da força desejada na manopla $|F|$ simula a força resultante num corpo que se movimenta sobre dois objetos distintos. As imagens a seguir exemplificam os dois cenários. Como demonstrado nas imagens, o módulo da força a ser imposta equivale ao desvio Δd da atual posição do usuário, divido pelo raio da seção transversal do objeto, multiplicado por uma força máxima, que por sua vez, simula o peso do objeto.

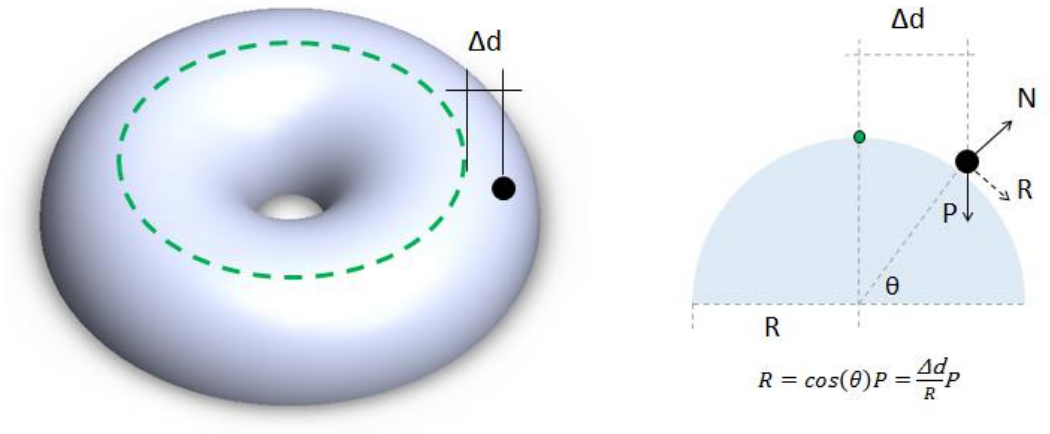


Figura 32 – Representação do cálculo da força no modo de controle de corrente “Avançado”.
Fonte: própria.

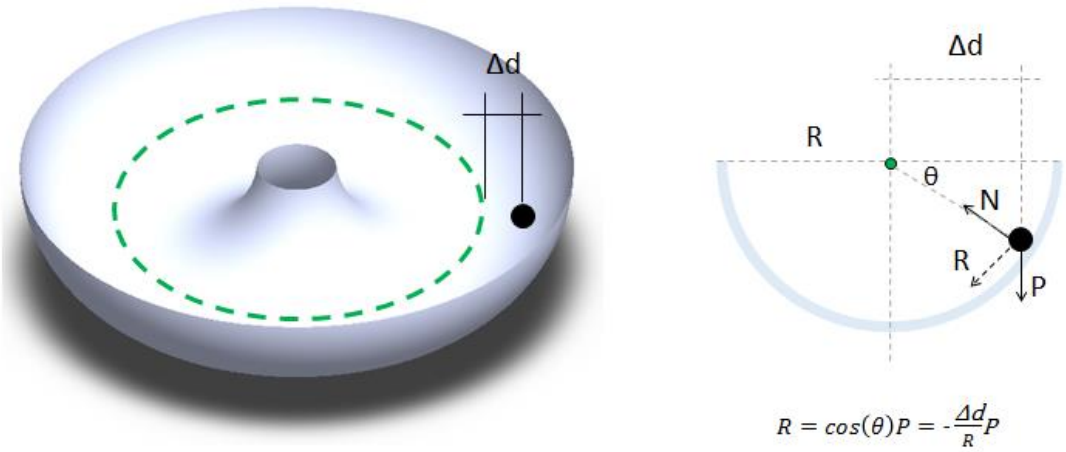


Figura 33 – Representação do cálculo da força no modo de controle de corrente “Iniciante”.
Fonte: própria.

A direção da força é representada pela reta que une a posição atual do usuário e o centro da plataforma. O sentido da força, por sua vez, depende da classe de exercício. No modo Iniciante (exemplificado na figura b), a força tenta levar o usuário até o trajeto ideal, sendo que nessa posição, a força é nula. No modo Avançado (exemplificado na

figura a), a força tenta levar o usuário para longe do raio médio, sendo que a força é maior conforme o usuário se distancia deste raio.

Para que a força desejada seja transferida à manopla, deve se calcular os torques necessários e posteriormente as correntes necessárias. O cálculo das correntes desejadas é feito a partir dos resultados das análises cinemáticas e dinâmicas mostradas anteriormente.

$$|F| = |R| = \pm\frac{\Delta d}{R} * P$$

$$\alpha = \text{atan}(\frac{y}{x})$$

$$T_{m1} = sen(\theta_1)\ (l_1 + l_3)\ |F|\cos(\alpha) - cos(\theta_1)(l_1 + l_3)\ |F|\ \text{sen}(\alpha)$$

$$T_{m2} = sen(\theta_2)\ (l_2)\ |F|\ \cos(\alpha) - cos(\theta_2)\ (l_2)\ |F|\text{sen}(\alpha)$$

Por fim, do experimento realizado com os pesos:

$$I_1 = 3{,}8002\ T_{m1} + 0{,}3902$$

$$I_2 = 1{,}7101\ T_{m2} + 0{,}3779$$

VI.iii. Controle de Posição

No modo de posição, o sistema lê as coordenadas atuais do usuário e procura impor uma posição diferente que por sua vez, depende da classe de exercício escolhida. Vale ressaltar que o torque (portanto a força) não é controlado, mas apenas calculado (estimado) nesse modo de controle, sendo que a intensidade do exercício pode ser ajustada pelas distâncias que o sistema tenta impor ao usuário.

O dispositivo sempre tentará fazer com que a manopla fique num raio ideal, definido pelo tipo de jogo (geometria da plataforma). Conforme o usuário se afasta deste raio, as forças aumentam na direção radial. Serão testados dois controladores para este exercício: um com PI e um proporcional.

No modo Iniciante, o dispositivo tenta levar o usuário até o trajeto ideal, onde a ação é nula. No modo Avançado, a força tenta levar o usuário para longe do trajeto ideal, sendo que nessa posição, a força é máxima.

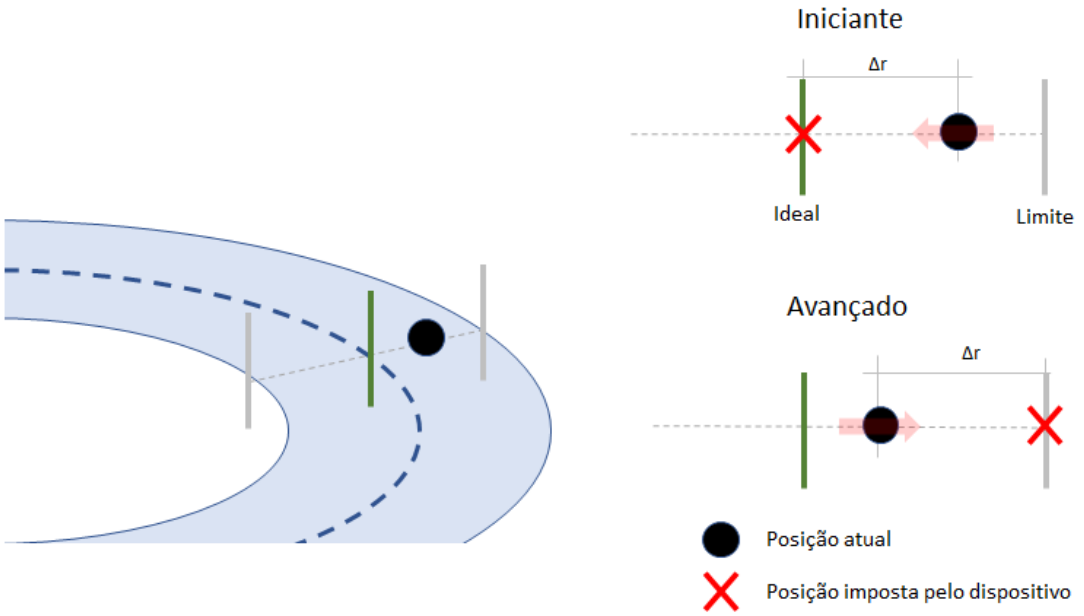


Figura 34 – Representação do cálculo da força no modo de controle de posição.
Fonte: própria.

Para calcular a posição a ser imposta, são utilizados os resultados da análise cinemática mostradas anteriormente. Além dessa análise, é necessário fazer a tradução dos ângulos desejados para a posição desejada do encoder. Para que isso possa ser feito, deve-se conhecer a posição inicial do encoder incremental, que por sua vez é encontrada pelo processo de Homing.

VI.iv. Método de Homing

O modelo de encoder usado nos motores é do tipo incremental, o que gera a necessidade de definir a origem do sistema toda vez que ele é inicializado. Para tanto, foi elaborado um processo que utiliza o sistema de leitura de corrente e o controle de posição dos drivers para encontrar os limites físicos do sistema. Cada elo atuado do sistema é acionado, incrementando lentamente o setpoint de posição, fazendo a leitura da corrente demandada a cada iteração. Quando o elo chega ao seu limite físico e a próxima posição é imposta, o driver tenta aumentar a corrente no motor para atingir a posição-alvo. Desse modo, quando um patamar de corrente é detectado, pode-se inferir que esta posição é um limite físico. O limite de corrente escolhido, também chamado de *threshold*, foi definido experimentalmente, com o valor de 3.000 mA. O mesmo processo é repetido para os elos na direção contrária de movimento. Um exemplo do processo, num experimento realizado no projeto está representado nos gráficos abaixo. Cada pico de corrente corresponde a um setpoint de posição. Os valores negativos representam correntes e movimentos no sentido anti-horário.

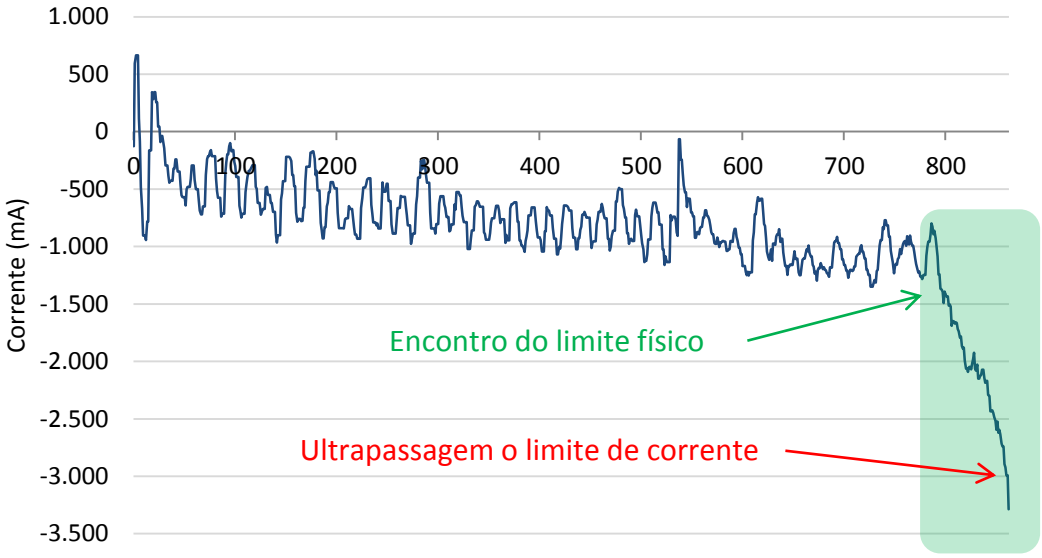


Figura 35 – Corrente lida por um dos motores no método de Homing, para o sentido anti-horário.
Fonte: própria

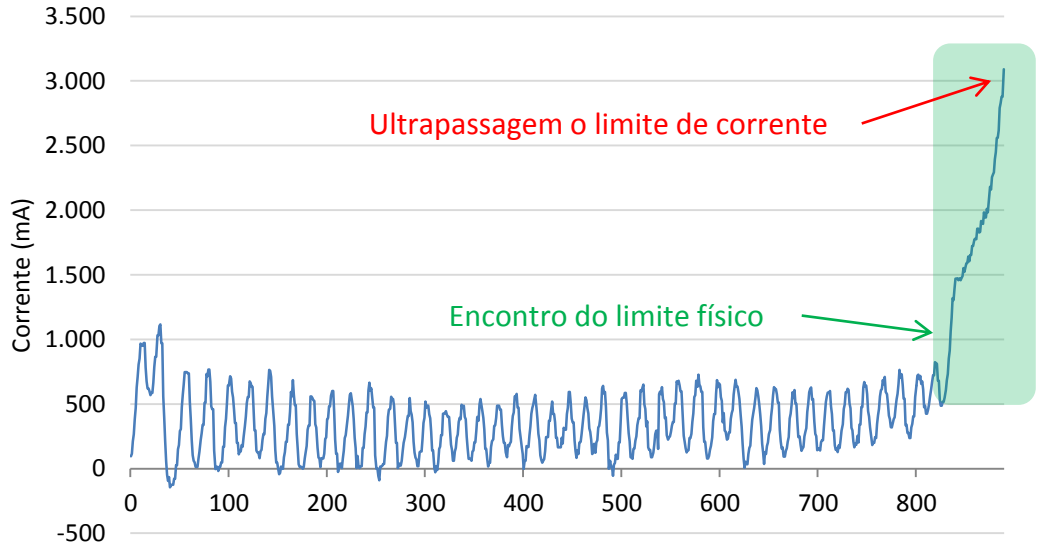


Figura 36 – Corrente lida por um dos motores no método de Homing, para o sentido horário.
Fonte: própria

VII. Projeto de Software para o Jogo

VII.i. Soluções Propostas

O projeto de software leva em conta a construção do sistema que irá envolver o ambiente virtual a ser utilizado. Para escolher os recursos mais adequados aos requisitos dessa etapa, era necessário considerar um ambiente que:

a) Interagisse com o usuário;
b) Apresentasse um menu de acesso;
c) Pudesse ser adaptável a qualquer tipo de mecanismo de interação;

Com isso, foi preciso estudar o tipo de ferramenta que permitiria atingir pelo menos esses itens citados.

O processo de seleção da plataforma de desenvolvimento foi realizado usando como referência, além dos itens citados acima, o estado da arte e os professores da universidade que já possuíam algum conhecimento associado a esse tipo de projeto. Dentro do que foi pesquisado, foram verificadas ferramentas com focos diferenciados, como o VESUP no artigo de Wauke (2004) mencionado previamente e que tem como finalidade ambientes para indivíduos com fobias. Contudo, nada que pudesse ser relevante à seleção da plataforma do trabalho foi encontrado, lembrando que o foco é a criação de um jogo sério para reabilitação física.

Através de conversas com professores experientes no assunto, concluiu-se que seria possível tomar como base a plataforma Unity, que é utilizada para criação de jogos e é de uso gratuito. Esta é uma solução interessante, pois cumpre todos os requisitos que foram citados nessa etapa e na seção de análise do problema.

VII.ii. Escolha da Solução

Com base no que foi dito acima, optou-se então por adotar como ferramenta de desenvolvimento o Unity em conjunto com o visual Studio 2017, o que permite a confecção de toda a programação na linguagem *C#*. Como descrito no projeto básico, a comunicação e integração será feita com auxílio do sistema embarcado adquirido.

VII.iii. Diagramas

Com a definição dos requisitos do projeto de software, foi necessário agora decidir como começar o seu desenvolvimento. É nesse contexto que se iniciou a elaboração dos diagramas UML (Unified Modelling Language). Não são recursos obrigatórios para o projeto, mas de uma maneira geral eles auxiliam muito na visualização de como funcionará o sistema e em como realizar sua estruturação.

No trabalho, foram utilizados como referência 3 tipos de diagramas: diagrama de casos de uso, diagrama de atividade e diagrama de classe. Os primeiros são diagramas comportamentais e o último é estrutural. Os primeiros foram construídos e finalizados antes do término do software, o diagrama de classes, por outro lado, será apresentado com o objetivo de facilitar a compreensão do código final, que se encontra na seção de apêndices.

VII.III.a. Casos de Uso

Para o sistema desenvolvido, considerou-se primeiramente a forma como o ambiente deveria ser manipulado. Assim, tomou-se como base quem seriam os usuários. Portanto, tem-se como atores: o paciente, que acessa o "jogo" criado; o terapeuta, responsável por auxiliar o paciente e também acompanhar suas atividades e resultados; o gerenciador, que será o administrador do ambiente virtual. O diagrama abaixo mostra, resumidamente, as atividades que cada um deles é capaz de realizar a partir das funcionalidades do sistema.

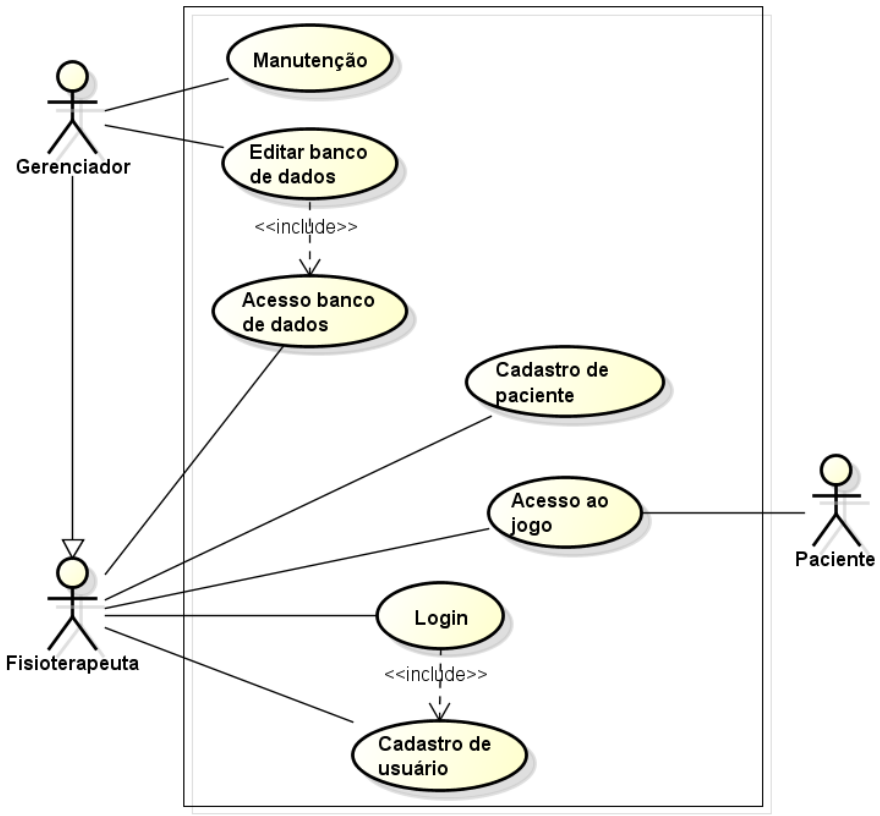


Figura 37 – Diagrama de Casos de Uso.
Fonte: própria.

Nota-se que os casos, em sua maioria, envolvem o terapeuta e o gerenciador, lembrando que o paciente possui limitação física de movimentação dos membros superiores (braços), restringindo suas ações às interações com o jogo.

VII.III.b. Diagrama de atividade

O diagrama de atividade é a forma utilizada para descrever os procedimentos gerais que ocorrerão quando os usuários interagirem com o software. É uma espécie de fluxograma que ajuda a visualizar como será o funcionamento do programa final. A seguir serão apresentados os diagramas considerados pertinentes representar. Eles estão associados aos casos de uso que levantados anteriormente.

a) Cadastro de usuário

Aqui são descritos os procedimentos de cadastramento de usuário. Isso ocorre na página inicial do login. O usuário indicará que deseja fazer um cadastro. Fornecerá as informações necessárias (nome e e-mail) e, se o sistema não verificar nenhuma inconsistência (como a já existência de um usuário com essas informações), um e-mail de confirmação será enviado ao indivíduo seguindo os dados fornecidos. Vale destacar que apenas os terapeutas e o gerenciador criam um login. Os pacientes têm acesso ao jogo através do login do terapeuta responsável.

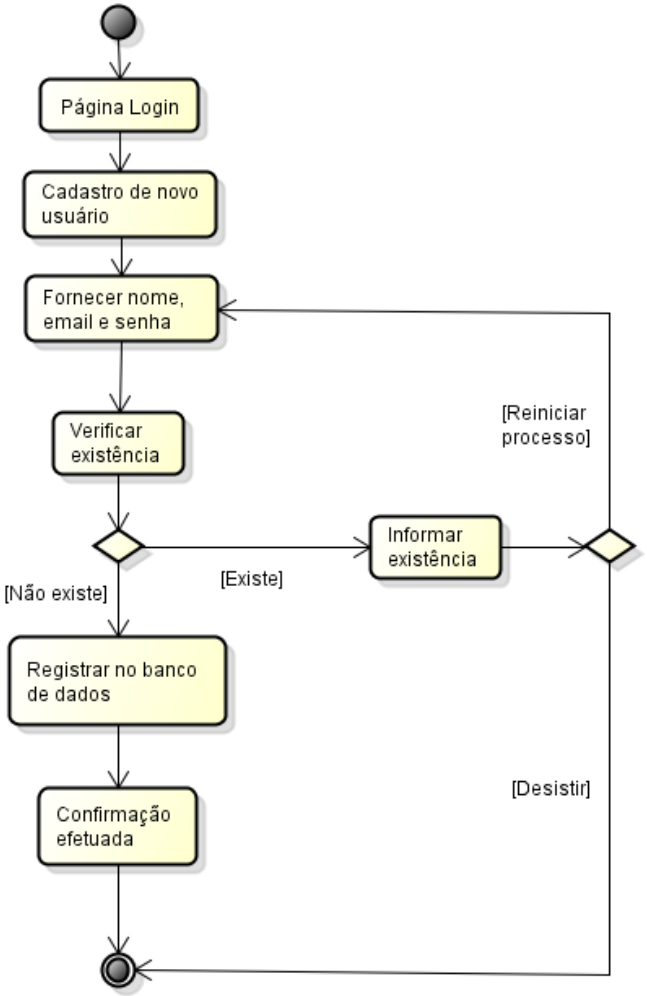


Figura 38 – Diagrama de Atividade do cadastro de usuário.
Fonte: própria.

b) Realiza login

O login é uma etapa simples. O usuário fornece suas informações na página de entrada. Se não existirem as informações correspondentes no banco de dados, é exibida uma mensagem de erro e o sistema permanece na tela inicial. Caso as informações sejam encontradas, o indivíduo terá acesso ao sistema principal.

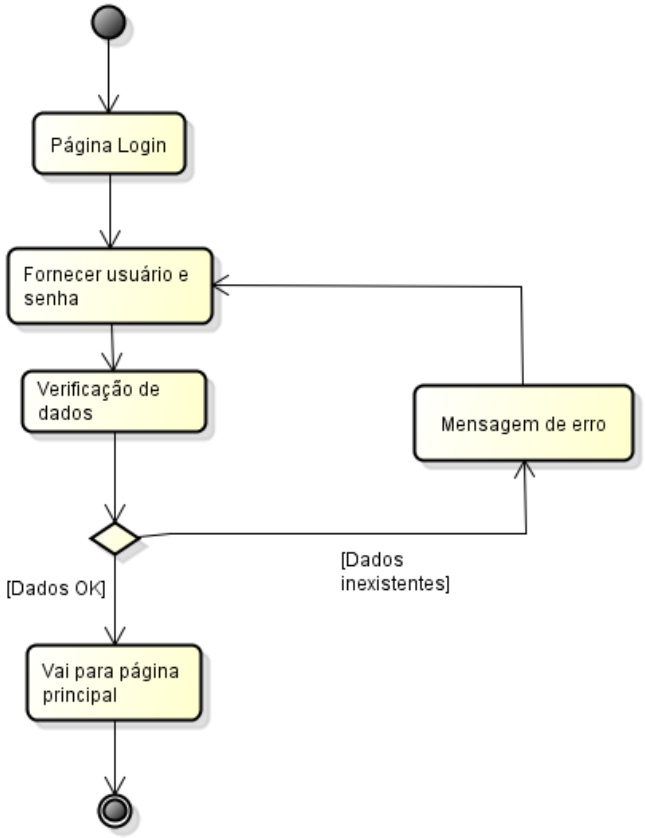


Figura 39 – Diagrama de Atividade da realização do login.
Fonte: própria.

c) Cadastro de pacientes

O cadastro de pacientes ocorre após o acesso ao sistema principal. Sendo realizado o login, o usuário vai para o menu principal. Lá, ele seleciona a opção de cadastrar um novo paciente. Nisso, deverão ser inseridas as informações relevantes sobre o paciente. Caso haja algum problema (como a verificação de que já existem esses dados), é disparada uma mensagem de erro. Se tudo estiver correto, o cadastro é realizado com sucesso.

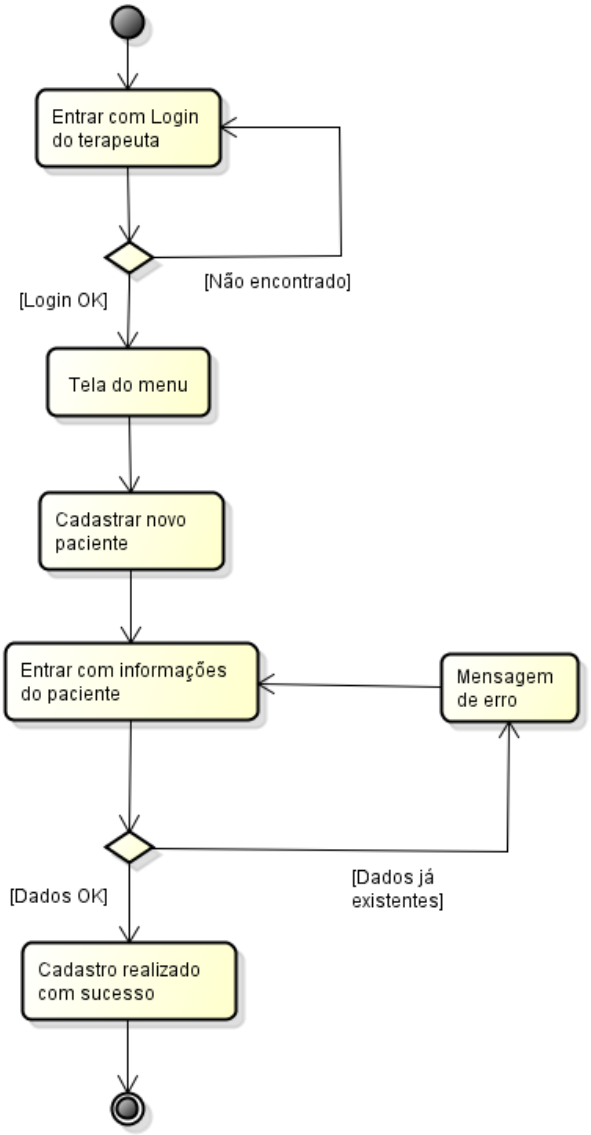


Figura 40 – Diagrama de Atividade do cadastro de pacientes.
Fonte: própria.

d) Acesso ao jogo

O acesso à atividade que o paciente irá executar começa com o terapeuta. Ele realiza o login e vai para o menu principal. Nesse momento, ele deverá inserir os dados do paciente que fará a seção de terapia. Se os dados não forem encontrados no banco de dados, aparece uma mensagem de erro. Se tudo estiver de acordo, realiza-se a configuração da atividade que em seguida será iniciada.

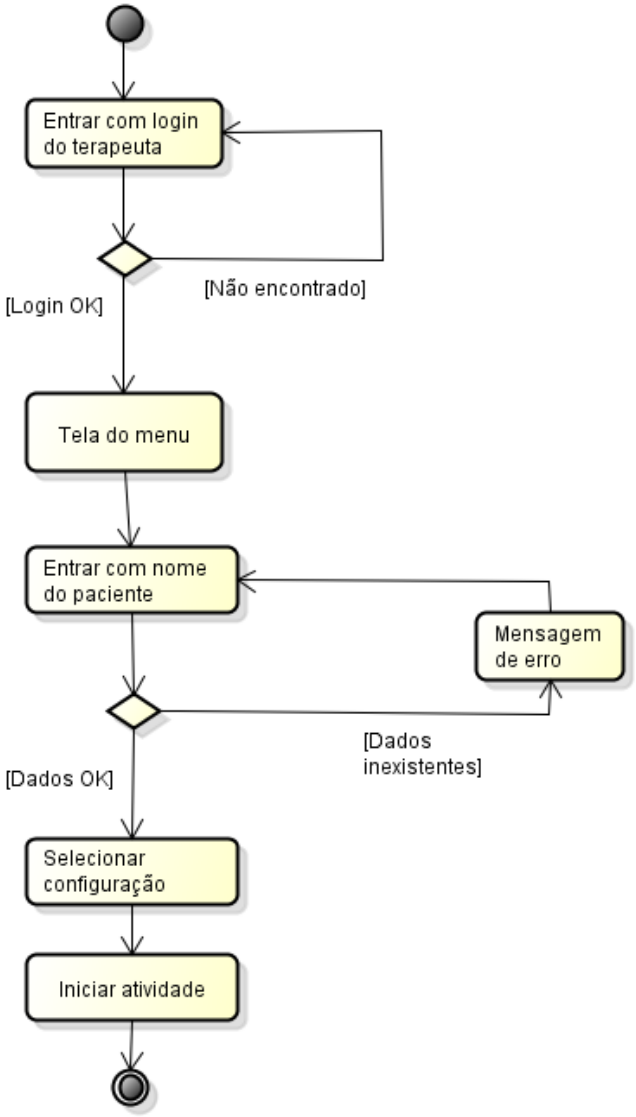


Figura 41 – Diagrama de Atividade de acesso ao jogo.
Fonte: própria.

VII.III.c. Diagrama de Classe

Para desenvolver o diagrama de classes do projeto é importante entender primeiramente as funcionalidades do Unity. Assim, será explicanda a estrutura do software e qual a sua lógica para, então, ser possível compreender o diagrama estrutural.

a) Estrutura funcional do Unity

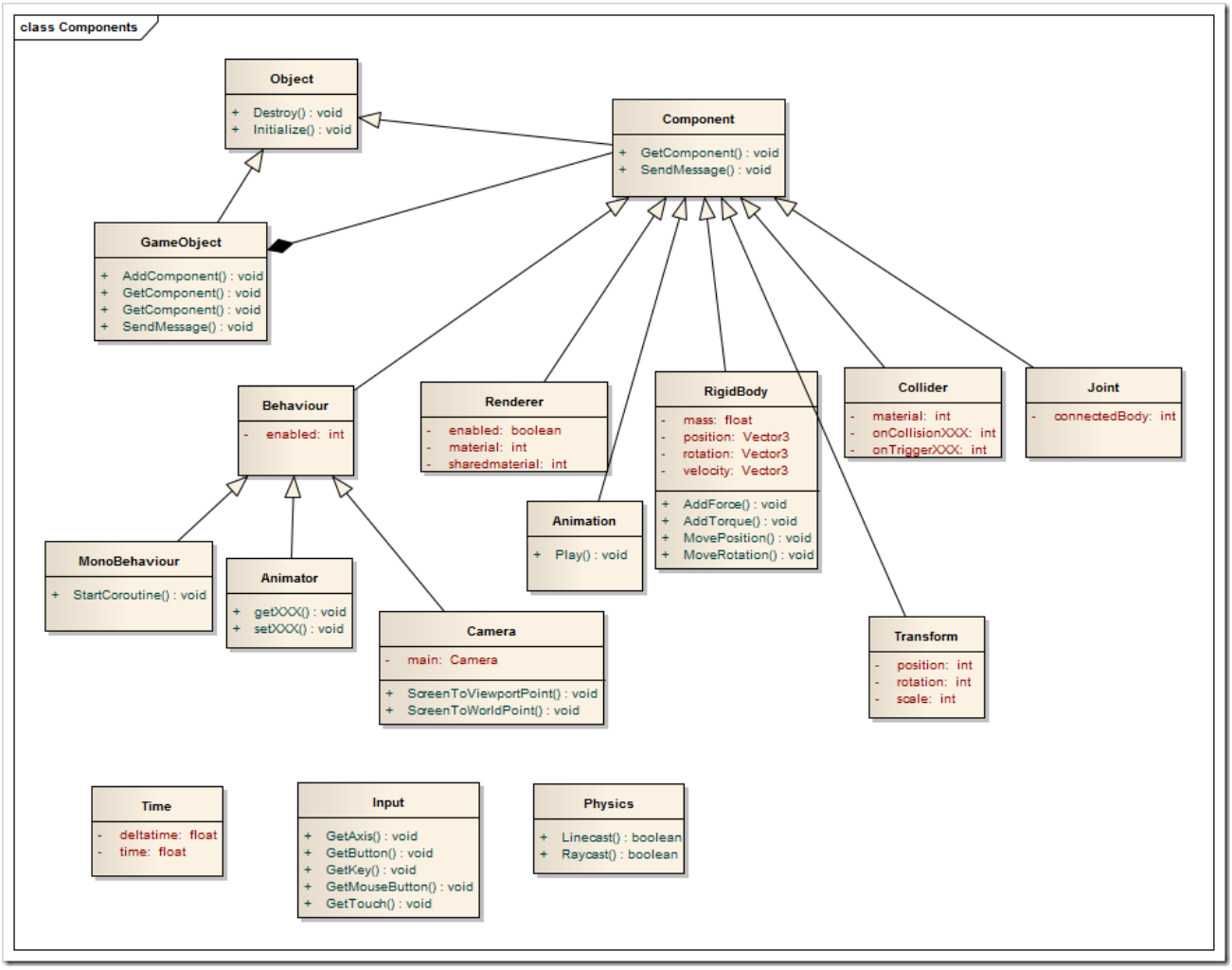


Figura 42 – Diagrama de Classe do Unity.
Fonte: programering.com.

Por ser uma plataforma de desenvolvimento de jogos eletrônicos, o Unity já possui diversas ferramentas prontas para facilitar o trabalho de criação dos usuários. Essas ferramentas são chamadas de componentes. Elas têm diversas finalidades, como definir a geometria dos objetos, impor conceitos físicos e etc. Acima está um diagrama bem simplificado das classes existentes, mas que consegue dar uma noção das relações compartilhadas entre elas.

Para o ambiente virtual projetado, os principais componentes utilizados são descritos abaixo. Eles foram essenciais para a elaboração da mecânica do jogo.

- Vector3: Esse componente serve como manipulador dos ponteiros e vetores 3D;
- Rigidbody: É uma classe que, quando atribuída a um objeto, lhe torna sujeito às características físicas (como sofrer ação da gravidade e poder colidir com outros objetos);
- Collider: É a classe responsável por tratar de colisões;

- Mesh: Classe que serve para tratar das superfícies 3D utilizadas no jogo (chamadas de meshes);
- GameObject: Classe base para todas as entidades criadas no jogo (jogador, câmera, obstáculo, superfície, etc);
- AudioSource: Representação de áudio no jogo;
- Time: Classe para trabalhar com o tempo;
- Transform: Classe existente em qualquer objeto criado, servindo para posicioná-lo, rotacioná-lo e mexer em sua escala (dimensão).
- Text: Classe que serve para representar textos na interface com usuário;
- GUIText: Classe de base gráfica e que também serve como representação de textos na interface.
- SceneManager: Serve para administrar as scenes (ambientes do Unity) usadas no jogo.
- IEnumerator: essa é uma ferramenta um pouco complexa de entender, mas que resumidamente serve para "forçar" rotinas dentro do loop de atualização natural da plataforma, sendo usada como exemplo em situações onde se deseja um delay entre atividades.

Vale lembrar que esses são apenas alguns dos componentes usados no programa. A quantidade de ferramentas existentes no Unity é consideravelmente mais vasta. Houve a necessidade de citá-los porque foram os mais relevantes parar entender o código.

b) Diagrama de classes do jogo principal

No Unity os programas/scripts existem como método de caracterização dos objetos. São gerados no formato de classes, de modo que a classe no código tem o mesmo nome que o script. Como consequência disso, não há uma rotina principal envolvendo os programas escritos (como um *"main"*), de modo que o jogo é composto por um conjunto de classes que descrevem de maneira segmentada o comportamento (*"behavior"*) dos elementos presentes.

O jogo principal é simples, onde o usuário controla uma nave que deverá coletar estrelas ao realizar uma trajetória circular na plataforma construída. Assim, foi desenvolvido para essa atividade um total de 5 scripts principais:

- Rotation: Controla a rotação e movimentação das estrelas que serão coletadas;

- GameController: É utilizada para controle principal da evolução do jogo (indicando quando há game over, quando fazer o restart e etc);
- PlayerController: Script com as rotinas que envolvem o personagem controlado pelo usuário;
- OnOff: Código utilizado para enviar o sinal de “start” e “stop” do jogo para o arduino;
- Serial: Script composto pelas rotinas necessárias para comunicação serial com o sistema embarcado do projeto (Arduino Mega). Este script está disponível de maneira pública na página UnitySerialPort (do usuário prossel) no GitHub.

Vale ressaltar que ainda existem mais scripts, gerados para a descrição dos menus do jogo. Nesta seção, o foco está no jogo principal, sendo que os menus serão melhor explicados ao abordar o banco de dados.

Apesar da grande quantidade de classes, o relacionamento entre elas pode ser destacado apenas entre GameController e PlayerController. Portanto, tendo em vista que a quantidade de funções e variáveis presentes nas classes é muito extensa, segue abaixo um diagrama bem resumido que mostra a ligação entre as duas classes.

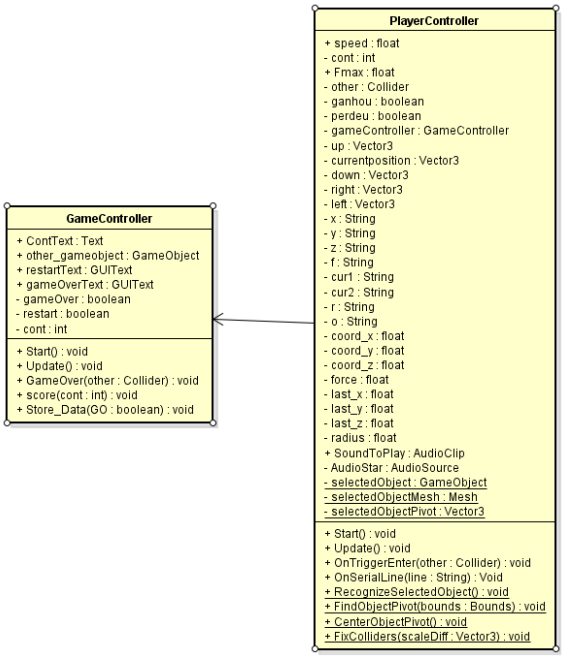


Figura 43 – Diagrama de Classe (GameController e PlayerController).
Fonte: programering.com.

VIII. Projeto Software para o Banco de Dados

Tomando como referência os diagramas de atividade apresentados na seção anterior, será descrito agora o processo de elaboração da base de dados que será aplicada em no projeto. Como foi possível observar previamente, será necessário guardar informações em três momentos:

- No login do usuário;
- No cadastro de pacientes;
- Nas tarefas executadas pelos pacientes.

Nesse contexto, considera-se a construção de cinco tabelas de armazenamento. A seguir, serão detalhados os procedimentos realizados, demonstrando as ferramentas utilizadas bem como os diagramas de entidade-relação.

VIII.i. Diagrama Entidade-Relação

Tendo em vista as informações que devem ser armazenadas, as tabelas consideradas para construção apresentavam as seguintes características:

1. Login: guardar as informações dos fisioterapeutas que realizarão o login no sistema;
2. Pacientes: armazenar os dados cadastrais do paciente que realizará atividade no sistema;
3. Atendimento: realiza a conexão entre os fisioterapeutas e os pacientes na base de dados;
4. Exercício: guarda as informações das atividades realizadas pelos pacientes;
5. Realização: faz o link da base de Pacientes com a base de Exercício.

A tabela Atende faz o relacionamento entre a identificação do fisioterapeuta com a do paciente na proporção 1:n, ou seja, um fisioterapeuta pode estar associado a mais de um paciente. Analogamente, a tabela Realização também faz uma relação 1:n entre os pacientes e os exercícios realizados (respectivamente). Assim, o diagrama de entidade relação (ER) para o sistema de armazenamento de dados foi desenvolvido e pode ser visualizado abaixo.

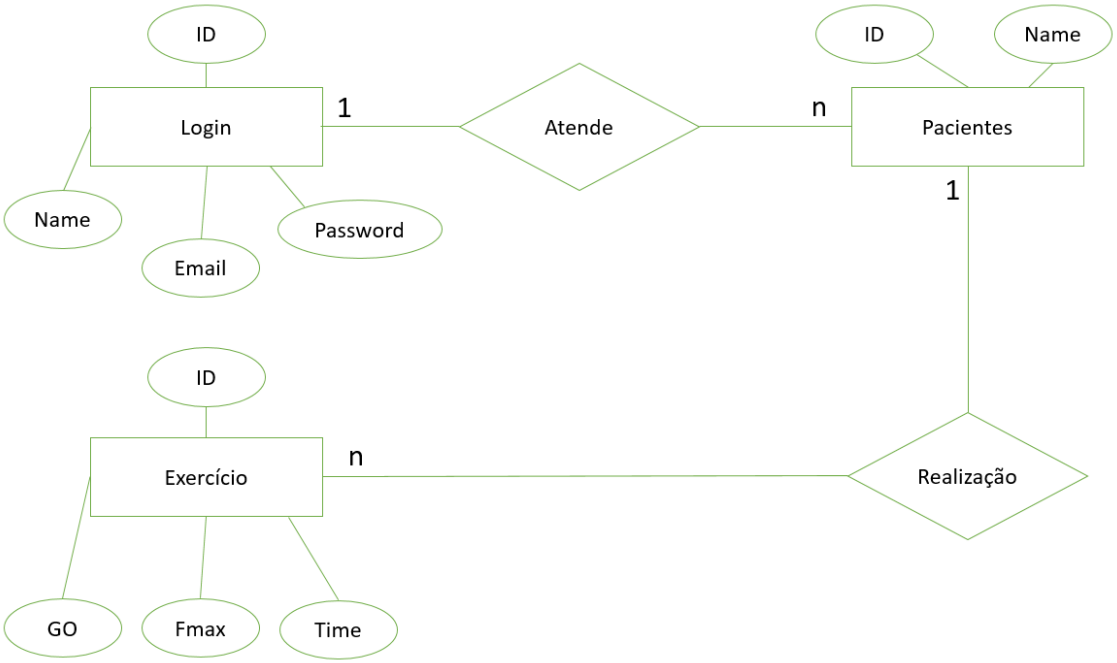


Figura 44 – Diagrama ER do banco de dados.
Fonte: própria.

Com relação aos atributos escolhidos para as tabelas, será explicado agora quais suas características e o porquê de sua existência no projeto. Primeiramente, para o Login, tem-se:

- Name (*string*): É o nome do usuário (fisioterapeuta) armazenado no login;
- Password (*string*): É a senha criada pelo usuário;
- Email (*string*): É o email fornecido pelo usuário no momento de cadastro;
- ID (*integer*): é o número de identificação de cada usuário (gerado automaticamente).

Para a tabela de Pacientes, tem-se:

- Name (*string*): É o nome do paciente cadastrado pelo fisioterapeuta;
- ID (*integer*): É o número de identificação de cada paciente (gerado automaticamente).

Para a tabela Exercícios, temos:

- ID (*integer*): É o número de identificação de cada exercício (gerado automaticamente);
- GO (*boolean*): Representa Game Over e guarda se o paciente "venceu" (completou) a atividade (True) ou não (False);
- Fmax (*integer*): É a força máxima que o mecanismo aplica (feedback) no usuário;
- Time (*long*): É o tempo de execução do exercício pelo usuário.

VIII.ii. Ferramentas de desenvolvimento

Para a construção do banco de dados, foi utilizada como base a plataforma XAMPP. Elaborada com o intuito de facilitar o desenvolvimento da estrutura de dados de websites, essa é uma plataforma livre que permite a criação de servidores web locais para testes. Por conta disso, disponibiliza em seu download aplicação para o servidor HTTP Apache, suporte para base de dados MariaDB (uma ramificação do MySQL) e script na linguagem PHP.

No contexto do projeto, seria necessário implementar as 5 tabelas mencionadas anteriormente e realizar sua comunicação com o Unity. Assim, recorreu-se inicialmente a ferramenta phpMyAdmin obtida com o XAMPP. Sua utilidade se resume à administração das bases do projeto.

Para o processo de interação, dada a necessidade de inserir e consultar informações nas tabelas, foi utilizada a linguagem PHP. Por meio desta, geraram-se códigos simples que possibilitam a comunicação através do caminho do arquivo que contém as rotinas de conexão (exemplo localhost/TCC_DB/Search_login.php).

VIII.iii. Layout do login

Dentro do que foi especificado, a página de Login desenvolvida para o protótipo foi criada de modo a conter duas janelas separadas: uma para o acesso do usuário e outra para o cadastro. A primeira foi composta por entradas de nome do usuário e senha. A outra possui entradas para nome, senha, confirmação de senha e email. As etapas de ambos os processos seguiram estritamente o que foi descrito nos diagramas de atividade mostrados anteriormente. Para sua dinâmica de funcionamento, foram criados 2 scripts: “Login”, que trata da parte de acesso do menu; “Registration”, que trata das etapas de cadastramento. Abaixo, é possível observar como ficou o design do login.

Figura 45 – Layout do login desenvolvido para o game.
Fonte: própria.

VIII.iv. Layout do menu

O menu principal do protótipo foi criado com base em duas etapas diferentes. Ambas seguiram um mesmo padrão de layout. Do lado esquerdo tem-se uma janela com botões indicando cada uma das opções. Dependendo do que for selecionado, a janela do lado direito apresenta o que pode ser feito para a opção clicada.

Partindo do princípio de que cada fisioterapeuta será responsável por trabalhar com mais de um paciente, o controle das atividades realizadas por paciente é feito nessa primeira tela. Assim, clicando no botão "Pacientes" a primeira opção disponibilizada foi cadastro ou entrada com nome do paciente. Além dessa opção, criou-se também o botão "Demonstração", onde testa-se o jogo, e o botão "Logout" para deixar o menu e retornar à página de Login. O script que define o funcionamento dessa etapa é o "Main_Menu".

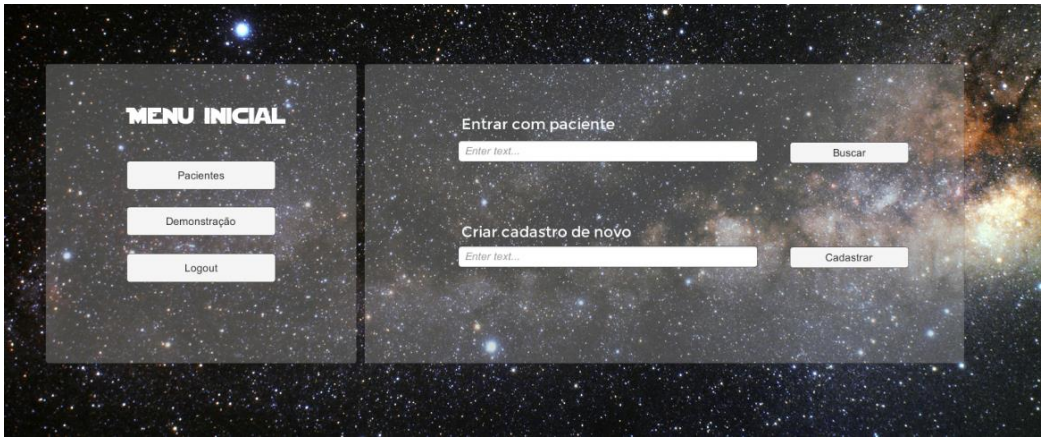


Figura 46 – Layout do login desenvolvido para o game.
Fonte: própria.

IX. Comunicação e Integração dos Subsistemas

Este capítulo visa explicar e justificar a escolha dos protocolos de comunicação usados no sistema. Também será mostrada a forma como todos os componentes se conectam e conversam para possibilitar a operação do sistema. Nesta interface háptica, existem 4 dispositivos eletrônicos (também chamados de nós), que utilizam dois protocolos de comunicação diferentes: serial (USB) e CAN. Os nós são representados por dois drivers de motor EPOS2, uma placa microprocessadora (Arduino Mega, ou "Placa") e um notebook (o qual será usado para rodar o jogo). Os drivers se comunicam com a Placa a partir de uma rede CAN, enquanto que o notebook conversa com a mesma Placa a partir de uma conexão serial (cabo USB).

IX.i. Conexão USB

A conexão USB será utilizada para a comunicação entre a Placa e o computador, onde estará sendo rodado o jogo. As informações transmitidas por USB serão a posição x e y da manopla no plano e a estimativa do módulo da força sendo aplicada pelos motores na extremidade do mecanismo. As informações transmitidas do computador para a Placa serão os comandos de início e parada do exercício, além das configurações escolhidas pelo usuário. Todos os cálculos e conversões serão realizados na Placa, enquanto que todos os aspectos relacionados ao jogo (pontuação, usuário, tempo de exercício e armazenamento de dados) serão processados no computador.

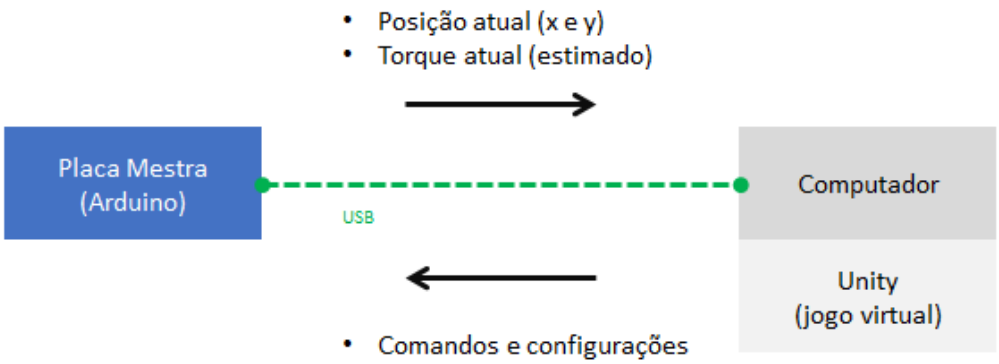


Figura 47 – Esquema de conexão USB.
Fonte: própria.

O protocolo de comunicação serial utiliza *"data frames"*, formados por um bit de início, os bits de dados (de 5 a 9 bits) e mais um bit para indicar o fim da mensagem. Desse modo, geralmente só é enviado um byte de dado a cada ciclo de transmissão. O sistema utiliza um canal de comunicação serial assíncrona, com "*baud rate*" de

115.200 bps para o recebimento e transmissão de cada parte. Foi escolhida uma alta frequência de transmissão, pois serão transmitidas mensagens relativamente longas, e uma vez que os comandos de impressão/leitura nas portas seriais são lentos, o processamento do programa poderia ser prejudicado.

A informação enviada pela Placa será enviada periodicamente durante o andamento do jogo, para atualização dos valores utilizados por ele. Uma vez que os monitores de computadores pessoais geralmente têm uma frequência de atualização de 60 Hz, e dado que esta frequência é suficiente para o olho humano perceber os movimentos virtuais como contínuos, a atualização dos valores de posição e força no escopo do jogo podem ser atualizados em frequências da mesma magnitude. Para otimizar o programa na Placa, foi definido que as mensagens contendo os valores físicos do mecanismo seriam transmitidos com frequência de 60 Hz (a cada 16,67 ms). Tal mensagem é transferida seguindo um padrão predefinido, no formato de *"string"*, para que apenas uma impressão seja feita na porta serial. Tal padrão é necessário para que o código por trás do jogo possa ler e entender as informações. O formato escolhido utiliza o carácter ';' (ponto e vírgula) para separação das informações. Por exemplo, caso a manopla esteja na posição $(x, y) = (10, 15)$ e a força estimada seja de 19 N, a mensagem enviada na atualização será a seguinte:

$$(valor\ posição\ x); (valor\ posição\ y); (módulo\ da\ força) \rightarrow 10; 15; 19$$

Já a informação enviada pelo computador para a placa será realizada apenas duas vezes durante um ciclo de exercício. As primeiras informações são as de configuração do jogo, seguidas do comando de início do jogo, representado pelo carácter 's' de *"start"*. Quando a placa detecta e lê tal informação na porta serial, é iniciado o ciclo de controle dos motores. Ao mesmo tempo, é iniciado o jogo no computador. A última informação é a de fim de jogo (imposta pelo usuário ou pela vitória/derrota no jogo), e é representada pelo carácter 'f' de *"finish"*. Quando a placa detecta e lê tal informação na porta serial, os motores são desligados e a placa volta para o estado de *"idle"*. Outros comandos são utilizados para escolha do tipo de controle, intensidade do exercício e tipo de jogo. Foi observado que a informação enviada do Unity para o Arduino nem sempre é recebida e interpretada, sendo que a solução foi enviar cada comando três vezes consecutivas a cada clique do usuário.

IX.ii. Rede CAN

De acordo com a National Instruments, em seu "*white paper"* "Controller Area Network (CAN) Overview", o protocolo de comunicação CAN foi desenvolvido pela

Bosch em 1985, para resolver o problema de conexões eletrônicas em automóveis. Com o aumento de dispositivos eletrônicos nos carros, se tornou necessário o uso de um *"bus"* que unisse todos os componentes, reduzindo custo, complexidade e peso dos cabos. Os principais benefícios e motivos para escolha desta interface para controle dos drivers são:

- Facilidade de comunicação com múltiplos componentes (no caso do dispositivo são 2 encoders, 2 leitores de corrente, 2 drivers e outros), usando apenas um cabo/canal;
- Velocidade de transmissão, com priorização de mensagens;
- Boa detecção e tratamento de erros
- Flexibilização da rede, possibilitando adição/subtração de nós sem impacto significativo na complexidade do código e do hardware.

As seguintes explicações são baseadas nos manuais de comunicação do tipo CAN do driver EPOS2. A estrutura de rede CAN sempre possui um mestre (gerenciador de rede) e seus escravos, conectados por um único barramento. Neste caso, o mestre será o Arduino Mega e os escravos serão os drivers EPOS2. Como pode ser visto nas imagens abaixo, a interface CAN é dividida em três principais níveis: físico, de "*datalink"* e de aplicação.

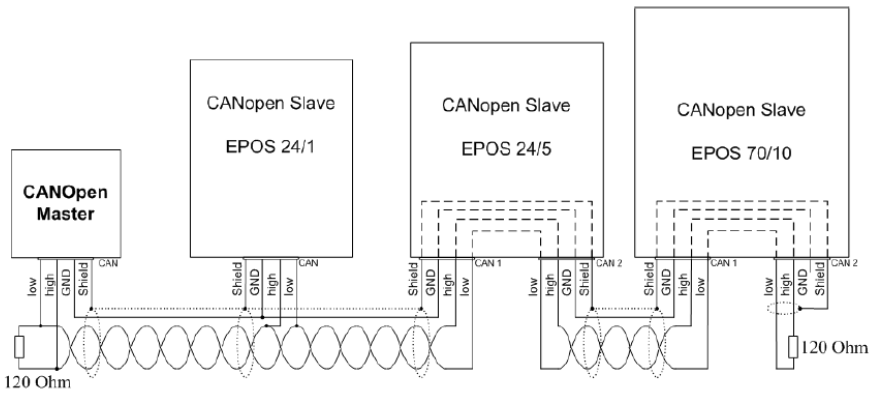


Figura 48 – Esquema de conexão de uma rede CAN genérica com drivers EPOS.
Fonte: EPOS Communication Guide.

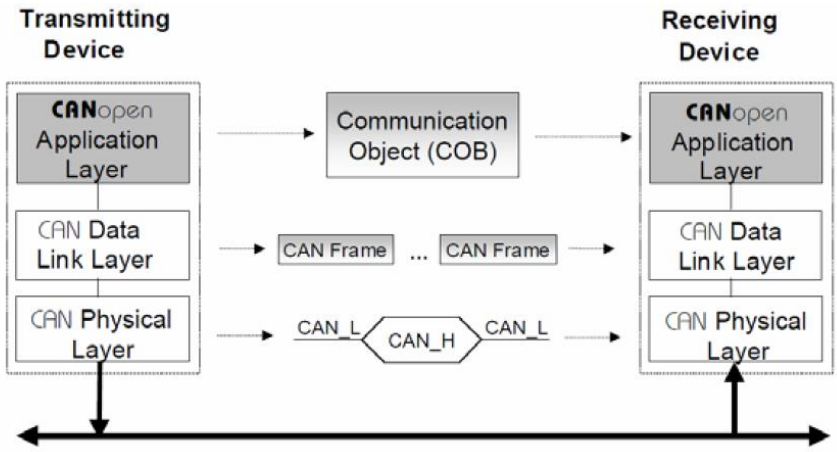


Figura 49 – Representação dos níveis de comunicação de uma rede CAN.
Fonte: EPOS Communication Guide.

No nível físico, a interface é caracterizada por um barramento de transmissão que contém duas linhas: CAN high e CAN low. Cada nó da rede CAN precisa conter um "*transceiver"* e um controlador compatíveis com a interface, para processamento dos sinais. Os drivers EPOS2 utilizados contém estes componentes internamente. Já para a placa microprocessadora, foi utilizado um "*shield"* de comunicação CAN para Arduino, chamado de módulo CAN MCP2515.

Acima do nível físico está o nível de data link, o qual é responsável pela transmissão/recebimento de informações dentro da rede, através de "*dataframes*" ou "*packets*". Tais pacotes contém as seguintes informações:

- Identificador de início e fim;
- Campo de arbitragem, que indica a prioridade da mensagem;
- Campo de controle, que indica o tamanho da mensagem;
- Campo de dados, onde fica a informação a ser transmitida;
- O Cyclic Redundancy Check (CRC), usado para detecção de erros;
- O Acknowledgment (ACK), usado para reconhecimento de recepção.

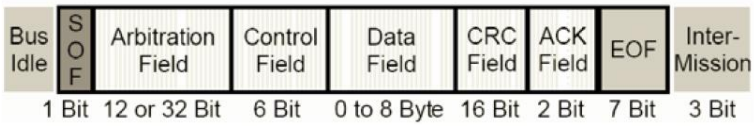


Figura 50 – Representação de uma mensagem no protocolo CANopen.
Fonte: EPOS Communication Guide.

Para o último nível, no escopo da aplicação, foram criados diversos protocolos para implementação da interface CAN. Neste projeto foi utilizado o protocolo CANopen. O principal aspecto desta camada é o dicionário de objetos, que é

predefinido pelo protocolo utilizado. Nele estão padronizadas todas as mensagens, conversas e dados a serem acessados entre os nós de uma rede CAN.

Na rede CAN, para efetuar as leituras de encoders e de corrente, foi implementada a comunicação do tipo *Process Data Objects* (PDO), que tem boa velocidade e flexibilidade. Neste protocolo, os nós da rede são programados para enviar/receber informações a partir de requerimento, acontecimento de eventos ou transmissão síncrona. Neste projeto foi utilizado o PDO do tipo "*Event Triggered*", onde o driver EPOS envia uma mensagem à placa mestre quando uma mudança é detectada (por exemplo, mudança na corrente ou posição).

As principais informações enviadas pela placa aos drivers são os comandos de referência de corrente e de posição, mas também são necessários envios de comandos de configuração e também de início/parada. No caminho inverso, os drivers enviam para a placa valores de corrente e posição, lidos nos motores.

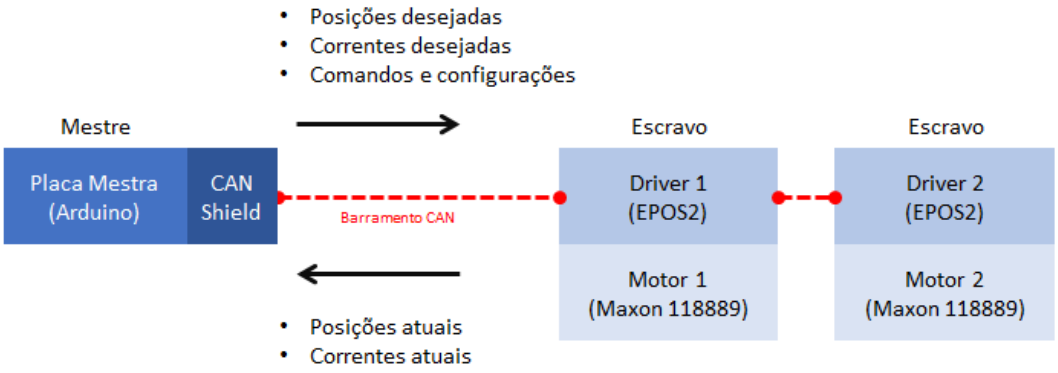


Figura 51 – Esquema da rede CAN do dispositivo.
Fonte: própria.

IX.iii. Banco de Dados e Unity

No Unity, foi necessário desenvolver dois ambientes diferentes: um para o Login e outro para o Menu. Para esses ambientes, foram criados scripts específicos para cada situação de interação usuário - unity - base de dados. O acesso à base foi realizado por meio das classes *WWWForm* e *WWW*. Ambas trabalham em conjunto, de maneira que a primeira possibilita a transmissão dos dados obtidos do usuário para o código PHP e a segunda é a que "guarda" o caminho do arquivo para a interação.

X. Metodologia Experimental e Testes

Uma vez que a fase de desenvolvimento do dispositivo foi finalizada, foi iniciado o processo de testes e análise. Este processo foi dividido em quatro diferentes partes.

X.i. Análise Mecatrônica

Nesta parte do processo, foram analisados aspectos de controle, cinemática, dinâmica, elétrica, buscando validar os requisitos de força e controle, além de analisar a capacidade de proporcionar segurança ao usuário. Foram feitos alguns testes com dois tipos de controle diferentes: de corrente e de posição.

Para o controle de corrente, primeiramente foram feitos testes padrões de controle, com entrada degrau, buscando identificar os parâmetros principais do sistema de atuação controlado.

Em seguida, com o equipamento completo, foram feitos dois testes, já com o jogo em funcionamento, seguindo o padrão projetado para um exercício no tratamento de reabilitação, com um usuário navegando pelo jogo. O primeiro foi com forças baixas (até 6N) e o segundo com forças altas (até 12N). Todos os dados lidos e calculados pela Placa foram impressos na porta serial, para posterior análise. O controlador PI utilizado aqui foi aquele obtido pelo software de *"autotuning"*.

Com o controlador de posição regulado, foi feito um teste de leitura dos encoders e validação da análise cinemática direta calculada. Para tanto, o dispositivo foi utilizado sem acionamento, de modo que o usuário o movimentava livremente. No lugar da manopla, foi utilizada uma caneta para comparar o caminho real da extremidade do mecanismo (desenhado numa folha de papel) e as coordenadas lidas pelo microprocessador. Em seguida, foi testada a análise cinemática inversa, fazendo com que o dispositivo desenhasse um círculo e uma espiral na folha de papel. Para estes testes primários, foi utilizado o controlador PID obtido pelo software de *"autotuning"*.

Por fim, com o equipamento completo, foram feitos mais dois testes, já com o jogo em funcionamento, seguindo o padrão projetado para um exercício no tratamento de reabilitação, com um usuário navegando pelo jogo. O primeiro foi com forças baixas (utilizando apenas um controlador proporcional) e o segundo com forças altas (utilizando um controlador PI). Todos os dados lidos e calculados pela Placa foram impressos na porta serial, para posterior análise.

X.ii. Análise Técnica

Na análise técnica, o equipamento em completa funcionalidade foi demonstrado e testado por uma profissional da saúde com experiência em robótica aplicada a tratamentos de reabilitação motora. Com estes testes, buscou-se primordialmente validar a aplicabilidade do equipamento desenvolvido no âmbito da fisioterapia. Também foi possível conseguir sugestões de melhora e posterior desenvolvimento do projeto, principalmente no que diz respeito aos limites de força, aspectos de segurança e de análise dos exercícios.

O equipamento também foi testado por alguns usuários, que não possuíam deficiências físicas, a fim de validar aspectos como a capacidade motivacional deste tratamento fisioterápico alternativo, a sensibilidade do controle de força e a dificuldade envolvida no jogo.

X.iii. Análise do Software

Na análise de software foi avaliado o funcionamento do jogo. Estudou-se o desempenho do software quando em atividade, verificando se ocorriam muitas interrupções ou travamentos e falhas na mecânica do jogo. Além disso, o protótipo foi apresentado para profissionais da área com o intuito de entender se o que foi elaborado atendia de fato às necessidades existentes em atividades terapêuticas.

X.iv. Análise de Viabilidade Econômica

Na análise de aplicabilidade, foi estimado o custo do dispositivo desenvolvido e sugerido um preço para o mercado. Para tanto, todos os custos incorridos pelos autores foram registrados. Os custos subjetivos foram estimados e incorporados nos custos finais. A partir do preço de mercado e custos incorridos, foi feita uma projeção do resultado obtido a partir das vendas de 1, 20 ou 50 unidades. Esta análise de resultado parte das vendas brutas, subtraindo custos e despesas até chegar ao lucro líquido.

XI. Resultados

Para análise dos resultados foi adotado um padrão nos gráficos para facilitar a leitura: linhas e pontos em verde estão relacionados à posição, em azul à corrente e em vermelho à força.

XI.i. Controle de Corrente

Os gráficos para análise do sistema de controle foram obtidos com a ajuda do software para o driver EPOS mencionado anteriormente. Foi imposta uma entrada degrau de 2 A para ambos os motores e a resposta foi adquirida com intervalo de 200 microssegundos. Dos gráficos obtidos, pode-se extrair que os tempos de assentamento de 95% para os motores são $t_{s1} = 1{,}00ms$ e $t_{s2} = 0{,}75ms$, o que é satisfatório, uma vez que o loop de controle acontece com intervalo de aproximadamente 15ms. Também é possível observar que a respota é estável e que não existe overshoot.

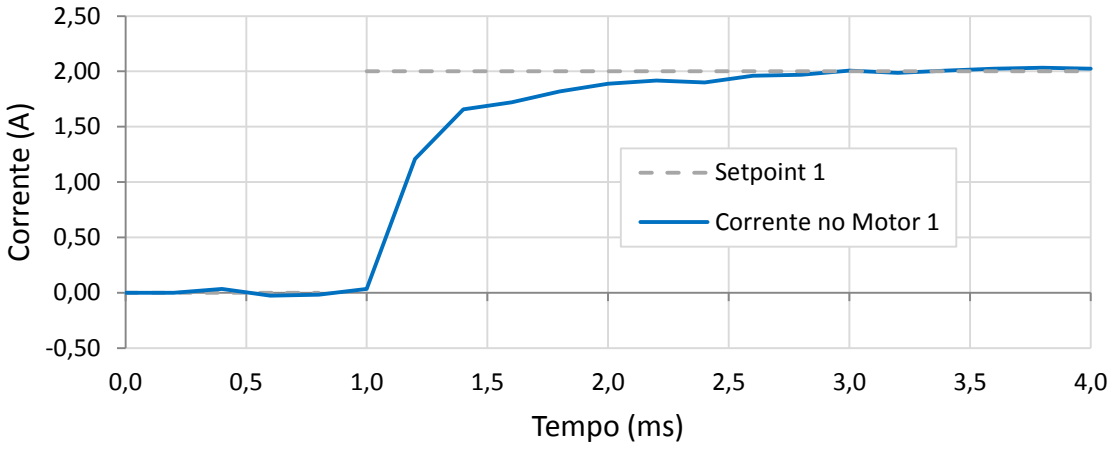


Figura 52 – Gráfico do teste de controle de corrente para o motor 1.
Fonte: Própria

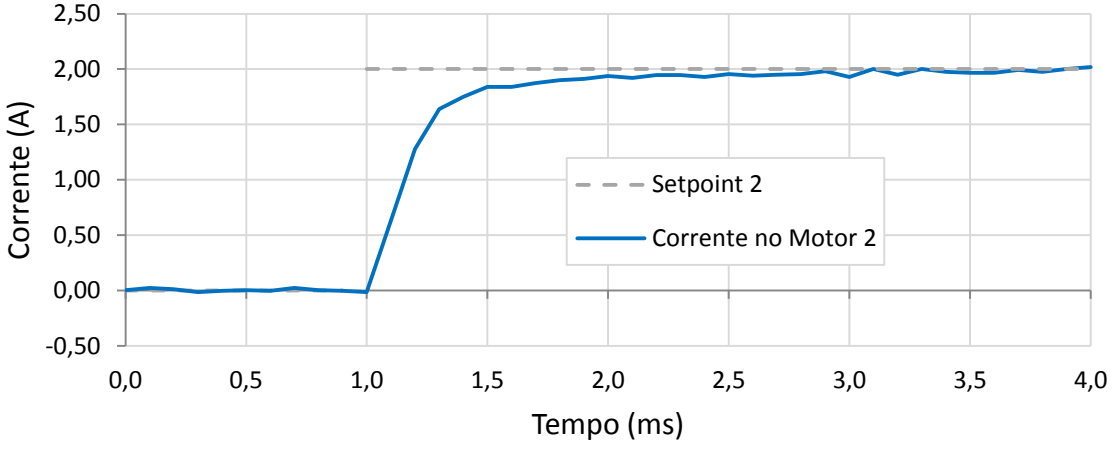


Figura 53 - Gráfico do teste de controle de corrente para o motor 2.
Fonte: Própria

O gráfico abaixo mostra um trecho da resposta dos motores no regime estático. A corrente estável oscila nos motores com amplitude de aproximadamente 4% do nível de corrente. Isto não é prejudicial ao projeto, já que uma variação de 4% na força não é facilmente identificada pelo usuário.

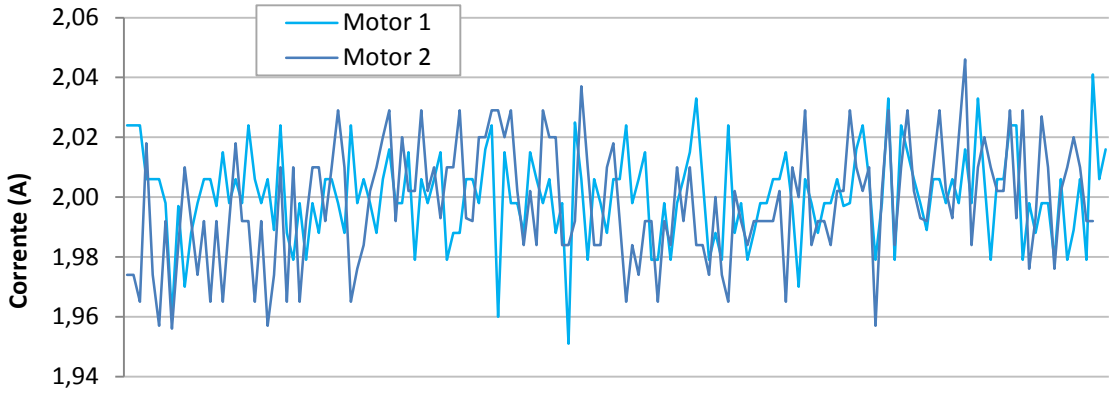


Figura 54 – Gráfico de controle de corrente em regime estático.
Fonte: Própria

Os seguintes gráficos correspondem ao teste realizado com controle de corrente e força máxima de 12 N. O gráfico abaixo mostra o caminho percorrido pelo usuário durante o experimento (linha verde). Como é possível observar, uma grande parte da área de trabalho foi percorrida, o que garante uma boa amostragem para a análise. Pode-se inferir também que o mecanismo criou uma grande resistência perto dos limites, visto que o usuário superou os limites apenas uma vez.

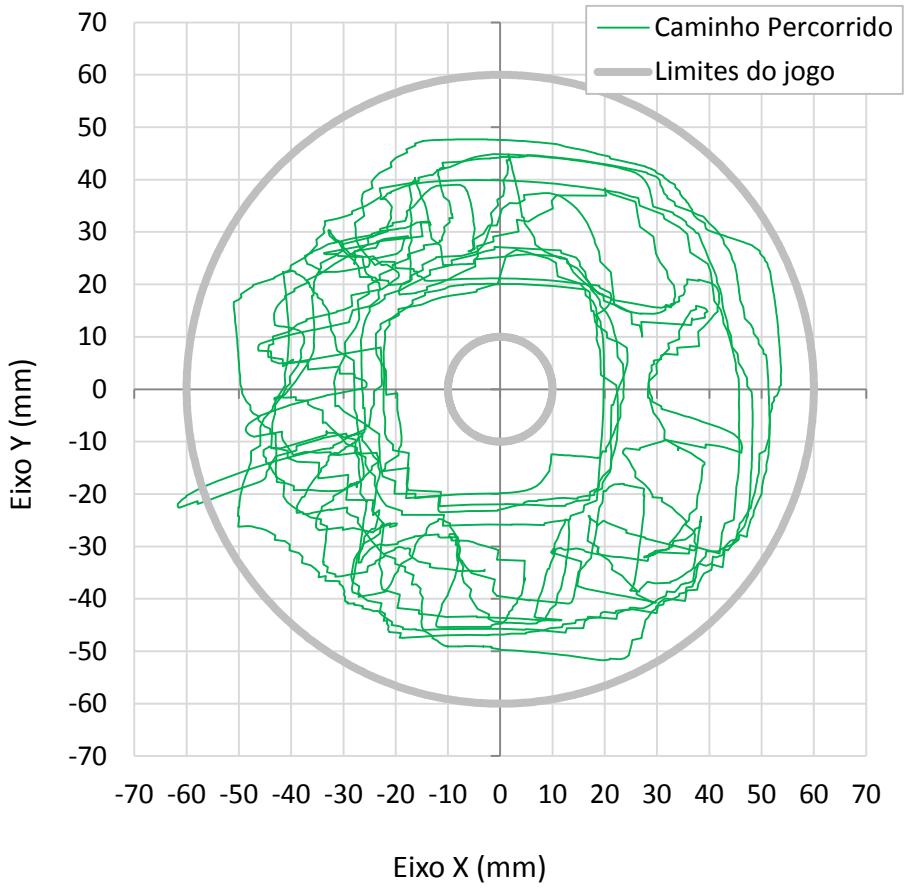


Figura 55 - Gráfico do percurso realizado durante o exercício analisado.
Fonte: Própria

Os dois próximos gráficos mostram o comportamento da corrente nos motores em um trecho do experimento. A referência exigida pela Placa está representado pela linha pontilhada cinza, enquanto que a azul é a corrente fornecida e medida pelos drivers. Neste teste específico, foi implementada uma "saturação" de corrente por código, criando um limite de 3 A para a corrente, a fim de testar a segurança do dispositivo neste tipo de controle. No gráfico do motor 1 é possível observar que em alguns momentos, o setpoint foi maior que 4 A, mas a corrente proporcionada foi de no máximo 3 A e sem grandes picos, o que mostra que é possível garantir a limitação dos torques e forças e, portanto, a segurança do usuário através do código. O segundo motor, por ter uma redução maior, tem um nível de exigência de corrente menor e não chegou ao limite de corrente.

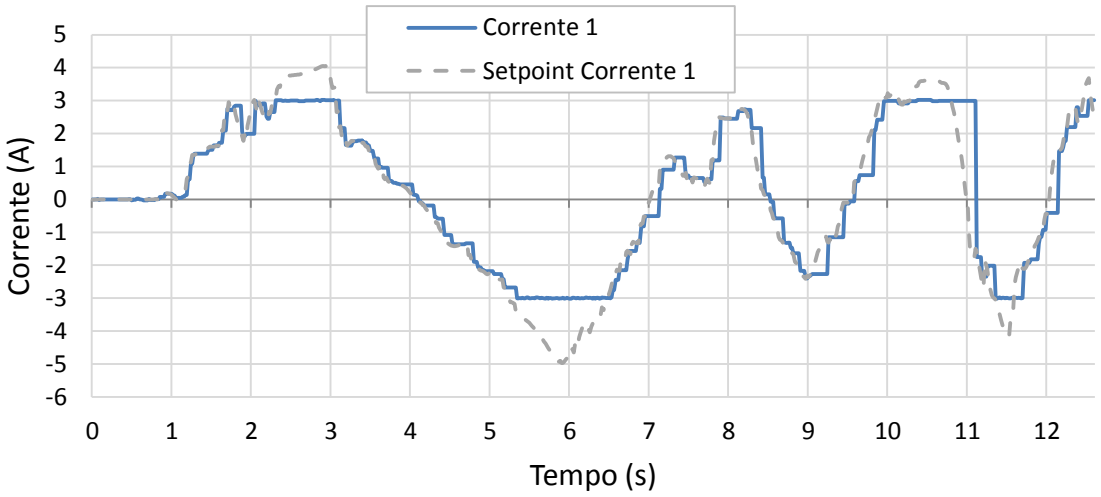


Figura 56 - Gráfico de controle de corrente durante trecho do exercício (motor 1).
Fonte: Própria

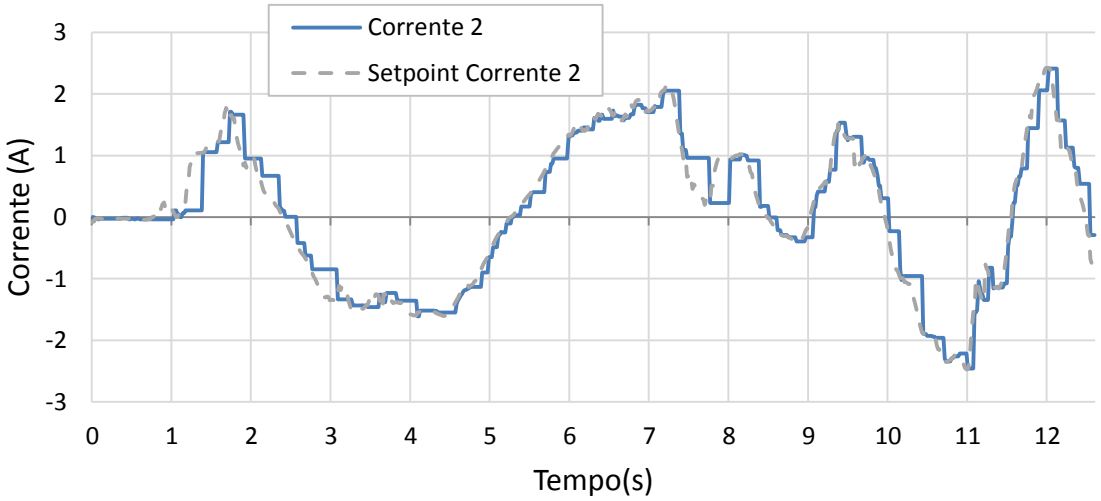


Figura 57 - Gráfico de controle de corrente durante trecho do exercício (motor 2).
Fonte: Própria

Outro aspecto interessante a ser analisado nos gráficos é a defasagem entre a corrente e a referência. Em alguns trechos, a corrente parece ter um grande atraso, sendo que a corrente permanece perfeitamente no mesmo nível (não há ruído). Isto indica que este deve ser um problema de atraso na leitura de corrente e não de atraso no controle. Numa rede CAN, apenas um barramento é compartilhado por todos os nós (tanto para recepção quanto para transmissão). Como descrito anteriormente, o método PDO é orientado a eventos e pode ficar preso no pipeline de processamento via CAN por um curto período de tempo, causando a defasagem.

No gráfico abaixo é possível analisar a referência do módulo da força na manopla e a força efetiva para o mesmo trecho do exercício mostrado nos gráficos anteriores. Vale ressaltar que a força efetiva foi calculada com base na leitura de

corrente, nos experimentos de torque versus corrente descritos anteriormente e na análise dinâmica feita.

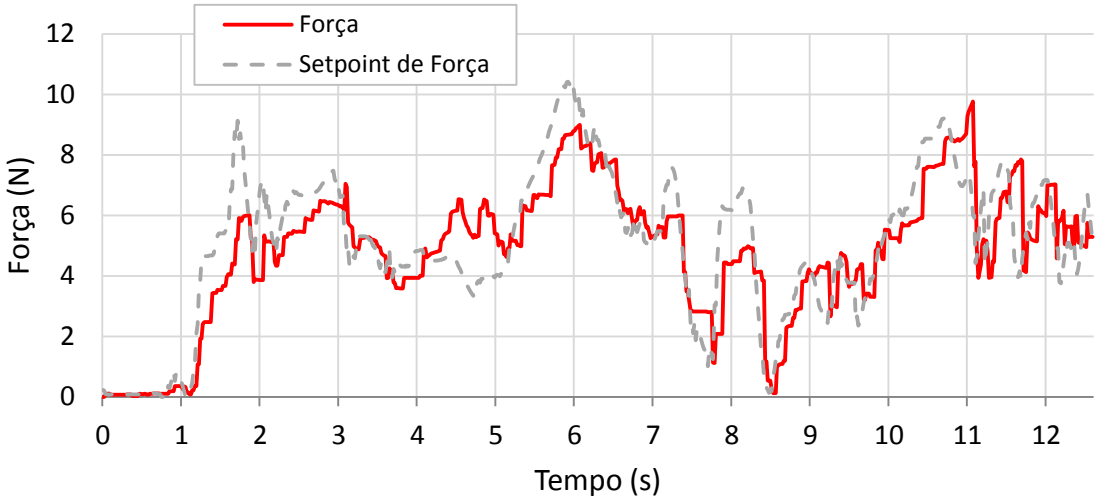


Figura 58 - Gráfico de controle de força durante trecho do exercício.
Fonte: Própria

Pode-se observar que neste trecho a força seguiu a referência satisfatoriamente, sem grande defasagem. A força atingiu o patamar de 10 N, compatível com o máximo de 12 N escolhido para o teste. O setpoint de força poderia chegar a 12 N, caso o usuário chegasse ao limite da plataforma (local onde a força é máxima). Em alguns trechos do gráfico, a força exercida não atinge a referência, provavelmente devido À limitação de corrente imposta.

Para o entendimento dos próximos gráficos, vale lembrar a proposta da dinâmica do jogo/exercício. Existe uma plataforma circular cujos limites são o raio interno (10 mm) e o externo (60 mm), sendo que o raio médio (35 mm) é o "ideal". A proposta é aliar as forças da manopla á geometria da plataforma, de modo que seja criado um campo gravitacional: a força na manopla é proporcional à diferença entre o raio "ideal" e o atual. Desse modo, quando o usuário está no raio "ideal" o módulo da força é mínimo e nos limites ele é máximo. Todos os raios mencionados são medidos em relação ao centro da plataforma.

O gráfico abaixo tenta demonstrar esta relação entre o raio atual e o módulo da força para o mesmo trecho dos outros gráficos. A linha verde mostra o módulo da diferença entre o raio atual e o médio. Como esperado, a força segue de forma bastante próxima o nível da diferença. Por exemplo, próximo aos 8,5 segundos, o raio médio é atingido (diferença é zero) e a força é então zerada. Aos 3, 6 e 11 segundos, a distância do raio "ideal" é máxima e a força tem seu pico. Entre os 2s e 3s, a força não

acompanha a tendência da diferença de raios por causa da "saturação" imposta para a corrente.

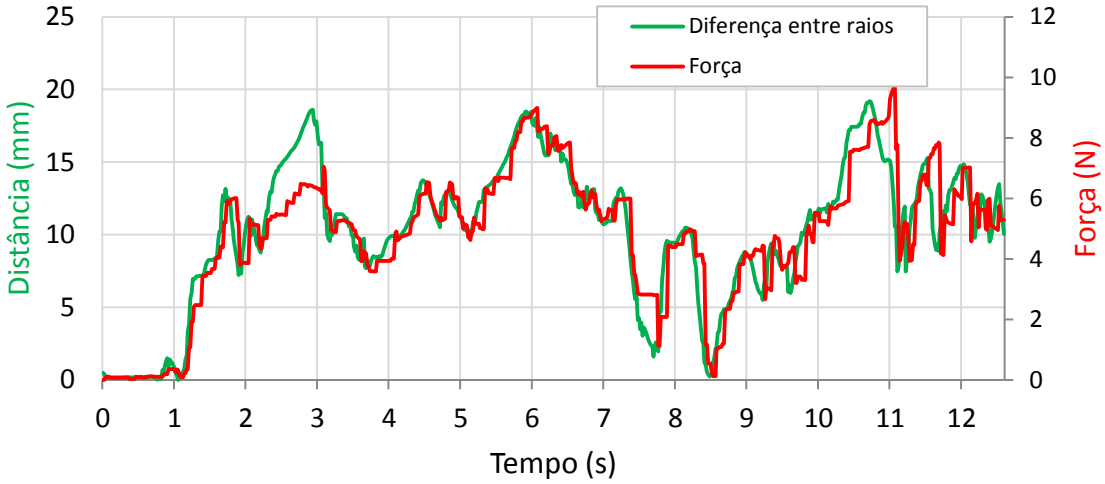


Figura 59 - Gráfico de relação entre posição e força.
Fonte: Própria

Outra boa forma de analisar o controle de força relacionado ao raio encontra-se no gráfico abaixo, que contém toda a amostragem do exercício. O eixo x representa o raio referente ao centro da plataforma e o eixo y o módulo da força. Cada ponto vermelho representa uma leitura de corrente/posição e cada ponto cinza, o setpoint correspondente. As setas mostram o sentido das forças. Como pode ser observado, a força seguiu razoavelmente bem o exigido, com exceção de alguns pontos próximos ao raio médio (35 mm), provavelmente por causa de pequenas defasagens. Isto não representa um problema, pois no gráfico de força versus tempo, ficou claro que o sistema consegue zerar a força nos raios médios com pouco atraso. Também é possível verificar que apesar de ocorrer alguns setpoints acima de 10 N (e fora da plataforma do jogo), o dispositivo não correspondeu, provavelmente devido à limitação de corrente.

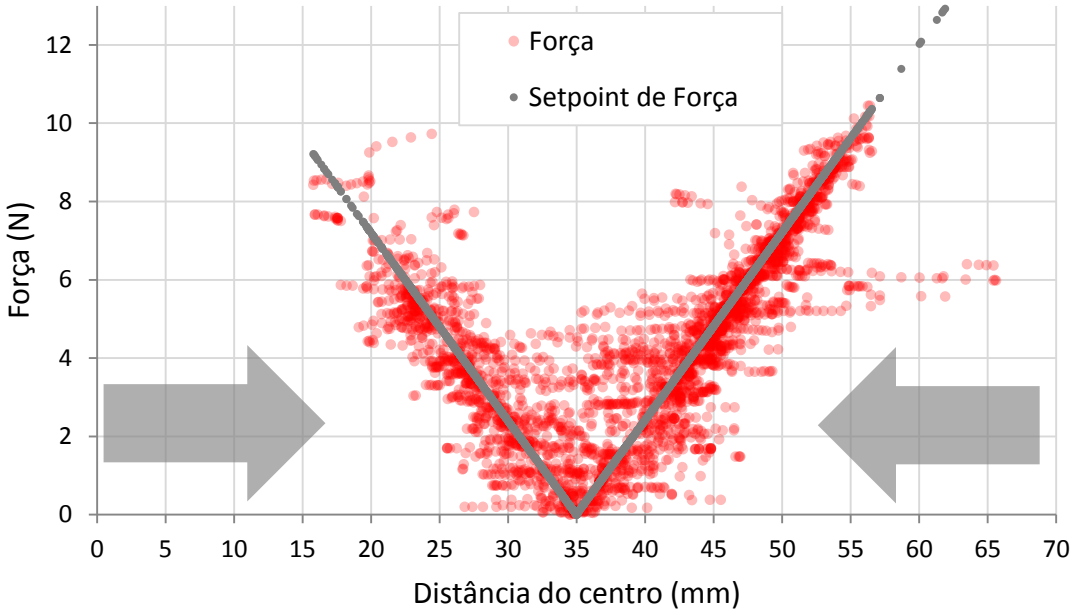


Figura 60 - Gráfico de relação entre raio e força (exercício de 12N).
Fonte: Própria

A seguir está o representado o mesmo gráfico para o exercício com forças menores (de até 6N). Novamente, o protótipo se comportou como o esperado. Nesta configuração ocorreu um desvio menor na região do raio médio

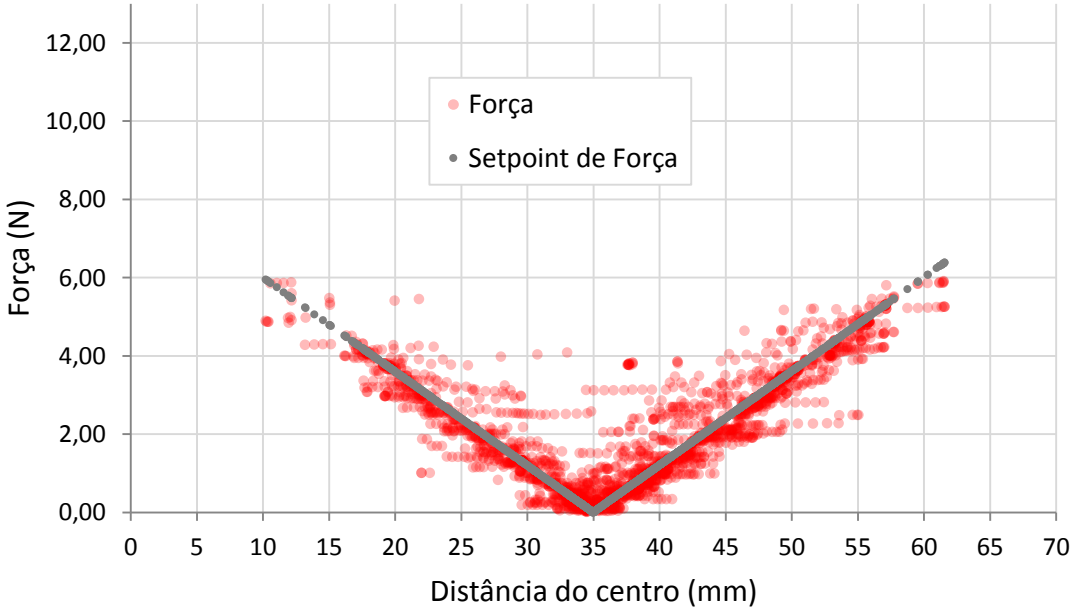


Figura 61 - Gráfico de relação entre raio e força (exercício de 6N).
Fonte: Própria

Para mostrar o funcionamento do "campo gravitacional" explicado anteriormente, os dados de posição e força do experimento com forças mais altas foram convertidos num gráfico tridimensional de superfície. Os eixos x e y representam as

coordenadas do usuário, enquanto que o z representa o módulo da força imposta na manopla. No primeiro gráfico está representada a força ideal, exigida pela placa neste experimento. Quando o raio é próximo de 35 mm, a força exigida é baixa ou nula, e quando a diferença para este raio médio aumenta, a força exigida também aumenta.

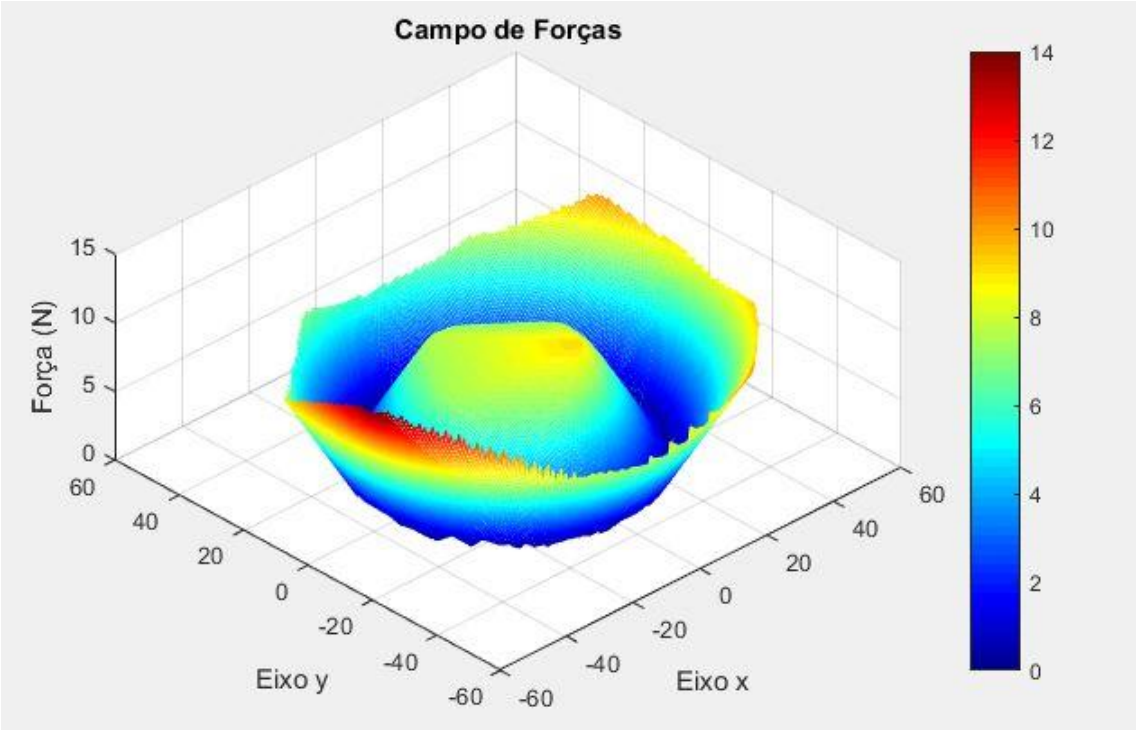


Figura 62 - Gráfico 3D do setpoint de força por posição (x,y).
Fonte: Própria

No próximo gráfico está representada a força que efetivamente foi transmitida à manopla. Pode-se observar que a geometria das superfícies são bastante similares. Assim como visto anteriormente nesta análise, existe algum desvio perto do raio médio. Um fato que chama atenção neste gráfico é o de que a força ideal não é atingida satisfatoriamente nas extremidades onde x é mais negativo. Não foi encontrado nenhum motivo para este comportamento.

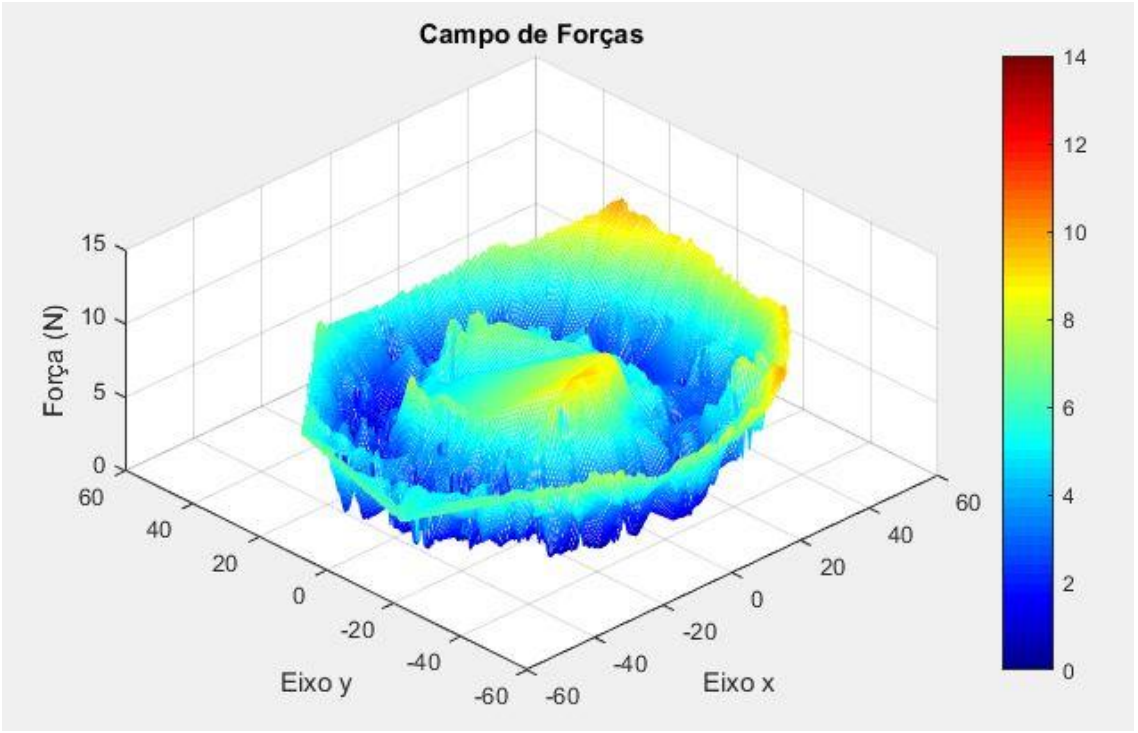


Figura 63 - Gráfico 3D da força lida por posição (x,y).
Fonte: Própria

Os mesmos gráficos podem ser analisados através da vista superior, com a ajuda do mapeamento de cor.

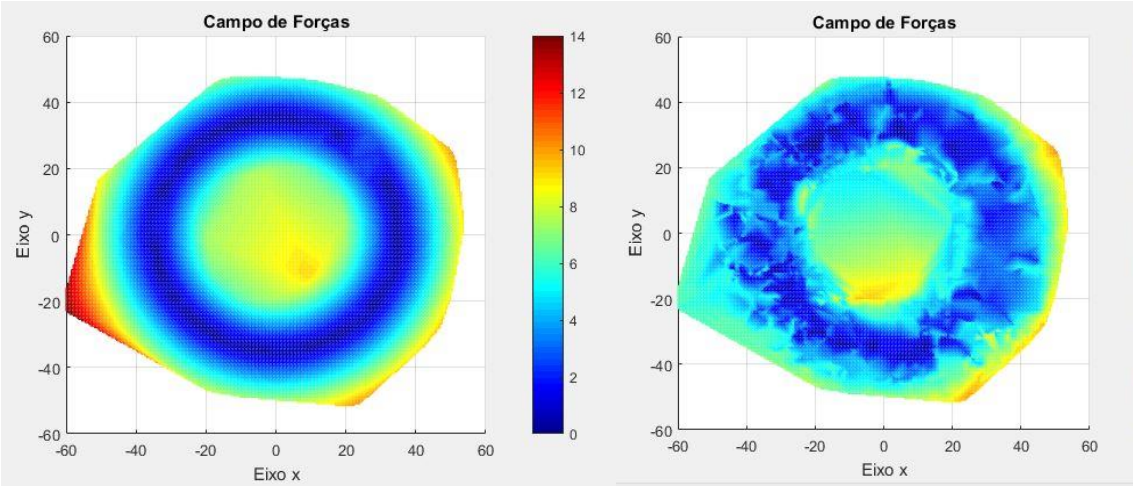


Figura 64 – Vistas superiores dos gráficos 3D.
Fonte: Própria

A capacidade de dimensionamento de força foi mostrada no gráfico acima. Para mostrar a capacidade de direcionamento das forças, o melhor gráfico é o de flechas. Cada flecha no gráfico representa um ponto da amostra. A flecha começa nas coordenadas do usuário naquela amostra. A direção e o sentido da flecha representam a direção e o sentido da força. Por fim, o comprimento da flecha representa a dimensão relativa da força. Primeiramente, no gráfico abaixo está o campo de forças desejado,

calculado pela placa. Como previsto, as forças são maiores conforme o raio do usuário se distancia do médio (35mm). A direção das setas mostra claramente a tentativa do dispositivo de levar a manopla até o raio médio, simulando a gravidade da plataforma do jogo. Neste gráfico, fica claro que a angulação das forças é perfeita.

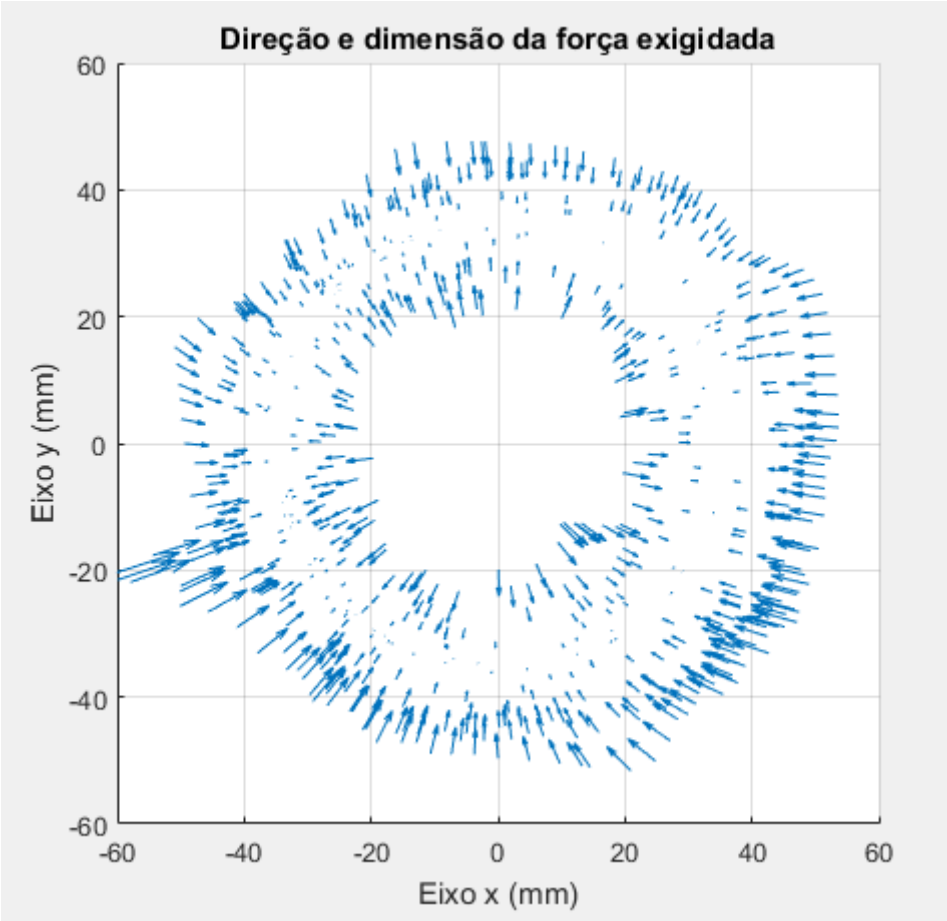


Figura 65– Gráfico de direcionamento de forças ideal
Fonte: Própria

O gráfico abaixo mostra o direcionamento real de forças exercido pelo mecanismo. É possível identificar novamente que as forças são menores do lado esquerdo da plataforma. Também é facilmente identificável que as forças aumentam com o distanciamento do raio médio. O direcionamento das forças é bastante claro e preciso nos maiores e menores raios, sendo um pouco impreciso nos raios médios. Esse fato não representa um problema, uma vez que a força nessa área é bastante baixa.

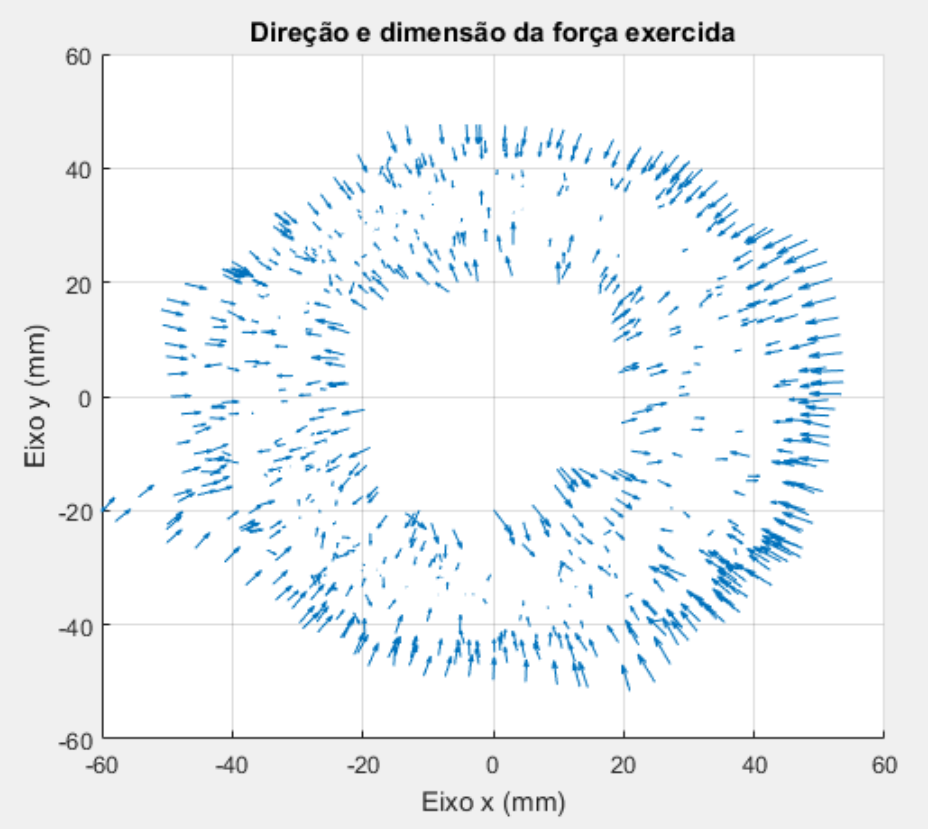


Figura 66 – Gráfico de direcionamento de forças real
Fonte: Própria

De modo geral, o controle de corrente para implementação do sistema háptico se mostrou bastante satisfatório frente à proposta do projeto. Foi possível dimensionar e direcionar a força na manopla com acurácia suficiente para o propósito do dispositivo. A partir de alguns testes com este tipo de controle, o campo gravitacional vislumbrado foi facilmente identificado, e aliado ao estímulo visual do jogo, se tornou um exercício bastante interessante. Os erros e delays mencionados nesta seção não foram identificados pelos usuários que testaram o equipamento. A limitação da força que foi obtida para fins de segurança também foi aceitável, ainda que sejam necessárias outras medidas de segurança.

XI.ii. Controle de Posição

Os resultados do teste de análise cinemática direta e de leitura de encoders encontram-se abaixo. Como pode-se observar, a leitura de posição funciona com acurácia e precisão suficientes.

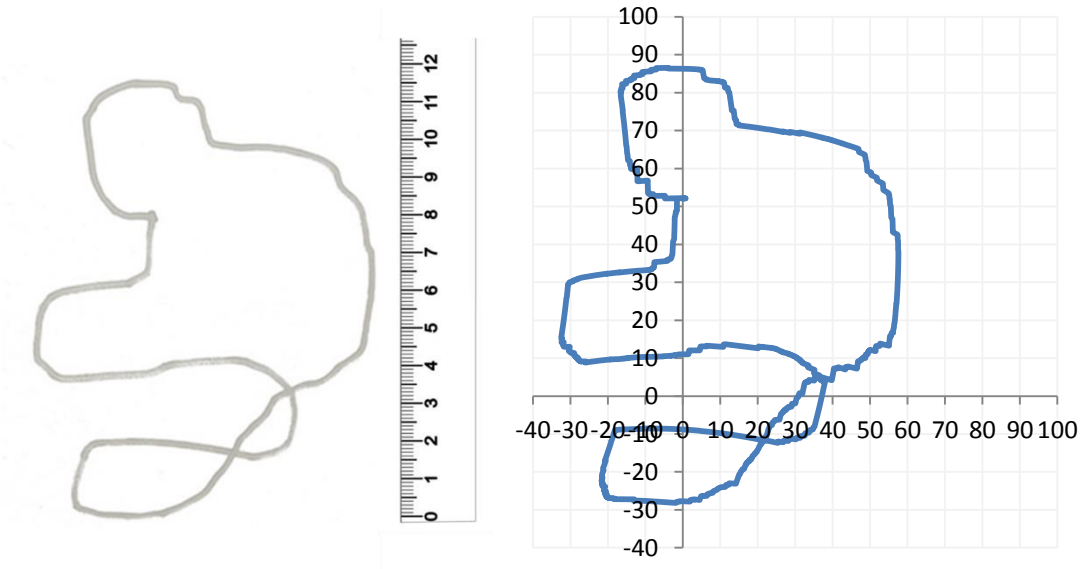


Figura 67 – Comparação entre desenho e leitura de posição.
Fonte: Própria

Em seguida foi realizado o teste em que o equipamento foi programado para desenhar um círculo, a fim de se validar os cálculos da análise cinemática inversa. Foram desenhadas três voltas completas com o dispositivo. Do gráfico abaixo, pode-se inferir que o controle de posição com o controlador do tipo PID funciona muito bem, visto que a leitura dos encoders segue os setpoints dados de maneira próxima. Em contrapartida, apesar de respeitar o mesmo trajeto nas três voltas, o círculo desenhado no papel não é geometricamente perfeito. Em outras palavras, apesar de ser preciso, o sistema não é acurado. Este fato leva a crer que os erros mais relevantes não são provenientes de folgas no mecanismo, mas sim de erros de fabricação.

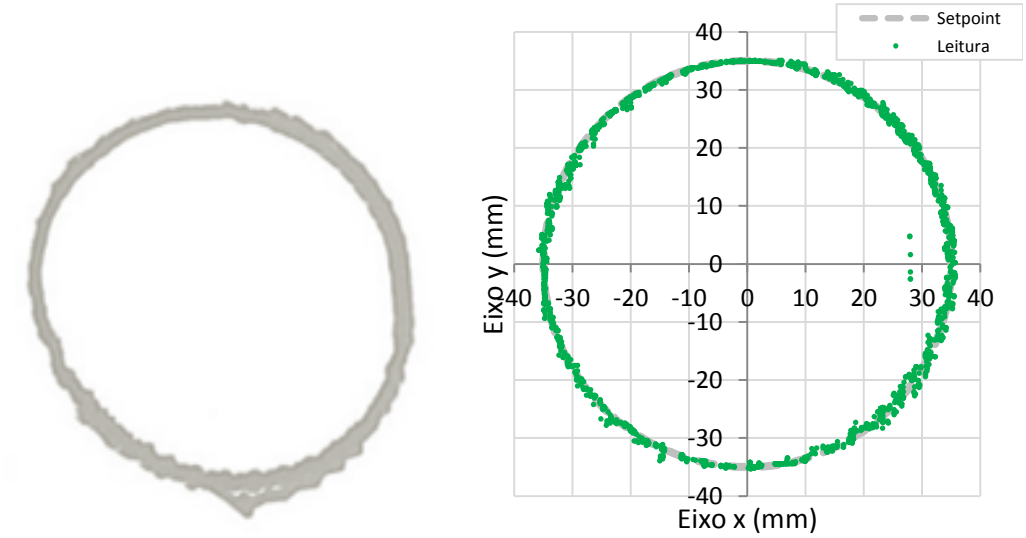


Figura 68 – Comparação entre círculo desenhado, setpoints e leituras.
Fonte: Própria

Para averiguar o impacto dos erros no projeto, foi feito um teste similar com uma forma de espiral (diferenças de raio de aproximadamente 6 mm), que é geometricamente mais complexa. Apesar do aparecimento dos mesmos indicativos de erros geométricos, percebeu-se que o sistema de posicionamento é suficiente para este projeto, uma vez que para o dispositivo, só é necessário direcionar e dimensionar a força na manopla a partir de referências de posição, e não efetivamente posicionar o sistema. Em outras palavras, a acurácia no posicionamento é menos importante do que a precisão.

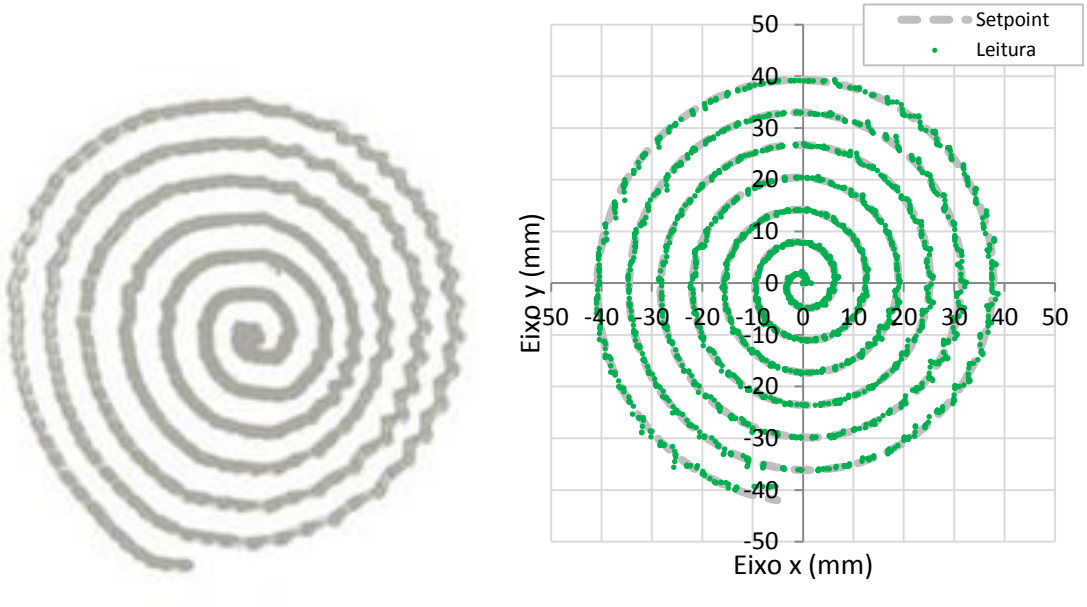


Figura 69 – Comparação entre espiral desenhado, setpoints e leituras.
Fonte: Própria

Para as próximas análises, vale lembrar a lógica de controle para este exercício. O setpoint de posição sempre será um ponto que está no raio médio e está no mesmo ângulo que o usuário (referente à origem da plataforma). Quanto mais longe do setpoint, maior será o esforço dos motores para chegar àquela posição.

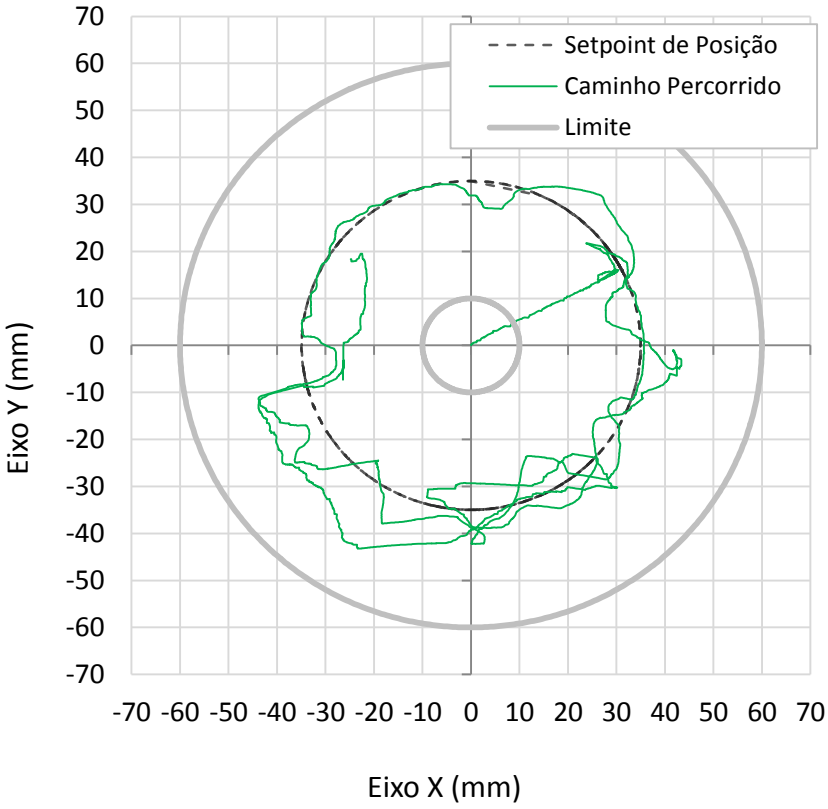


Figura 70 – Percurso percorrido durante o exercicio.
Fonte: Própria

Os próximos gráficos mostram a capacidade do dispositivo de seguir a referência de posição durante o jogo. Com o controle proporcional, o dispositivo é capaz de seguir o setpoint proximamente, apesar de apresentar pontos de *"overshoot"* relevante e defasagens maiores do que o sistema de controle de corrente. Também se pode observar pelo gráfico que os movimentos neste teste foram muito mais lentos do que no outro.

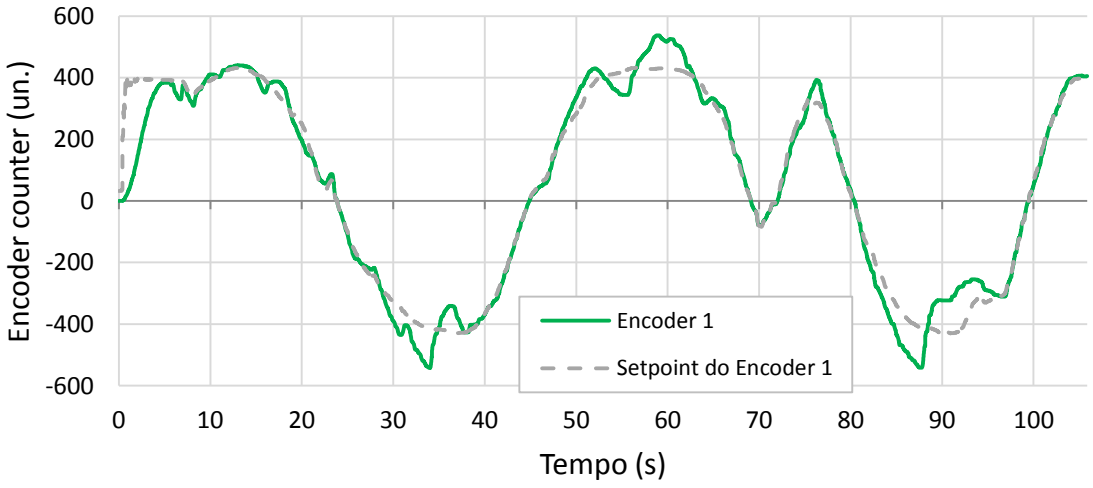


Figura 71 – Gráfico de controle de posição durante o exercício (motor 1).
Fonte: Própria

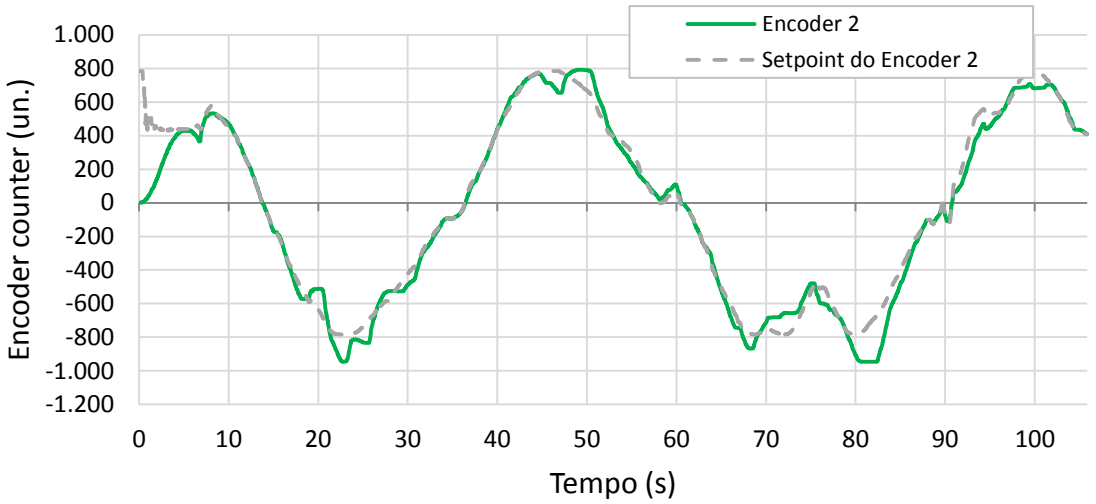


Figura 72 – Gráfico de controle de posição durante o exercício (motor 1).
Fonte: Própria

O gráfico a seguir mostra a relação entre a força estimada na manopla e a diferença entre o raio atual e o raio médio. Como era de se esperar, devido ao controle proporcional, a força aumenta conforme a posição do usuário se distancia do setpoint. Quando comparado ao mesmo gráfico na seção de controle de corrente, percebe-se que no de posição a força não é zerada no raio "ideal", o que mostra a resistência ao movimento em outras direções. O nível da força nesta situação chama a atenção, sendo que é necessário pelo menos de 4 a 6N para movimentar a manopla, mesmo que o usuário esteja no raio médio.

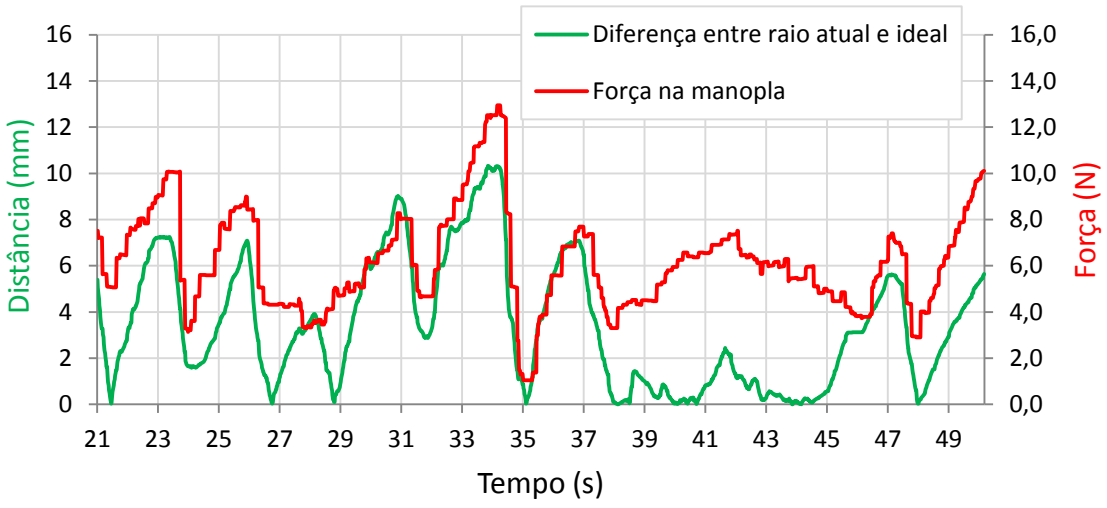


Figura 73 – Gráfico de relação entre posição e força.
Fonte: Própria

O gráfico abaixo também pode ser comparado ao do controle de corrente. Novamente, é possível identificar o formato de "V" neste experimento, apesar de que o ângulo interno é bem menor desta vez. Também há forças mais altas na região do raio

médio, chegando até o nível de 10N. Tais resultados levam a crer que o ganho proporcional do controlador foi mais alto do que o necessário.

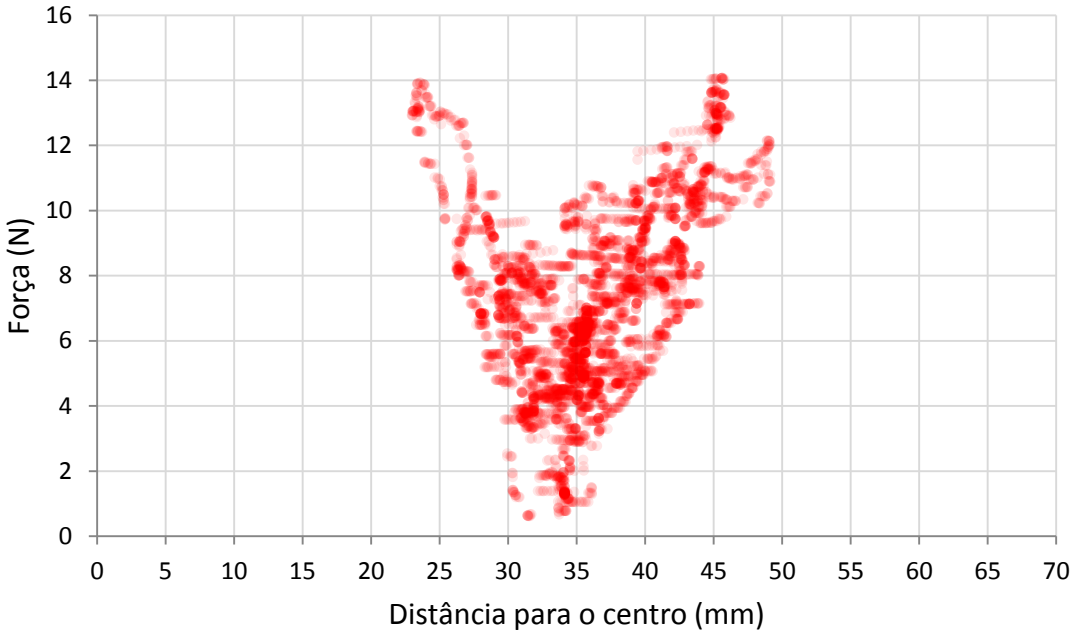


Figura 74 - Gráfico de relação entre raio e força (controlador proporcional).
Fonte: Própria

O gráfico abaixo é do mesmo tipo, mas foi obtido com o controlador de posição do tipo PI. Como pode ser observado, as maiores forças ocorreram perto do raio médio e o formato de "V" não foi identificado no gráfico. Dessa forma, O controlador PI não atendeu às expectativas do projeto e foi descartado como possível solução.

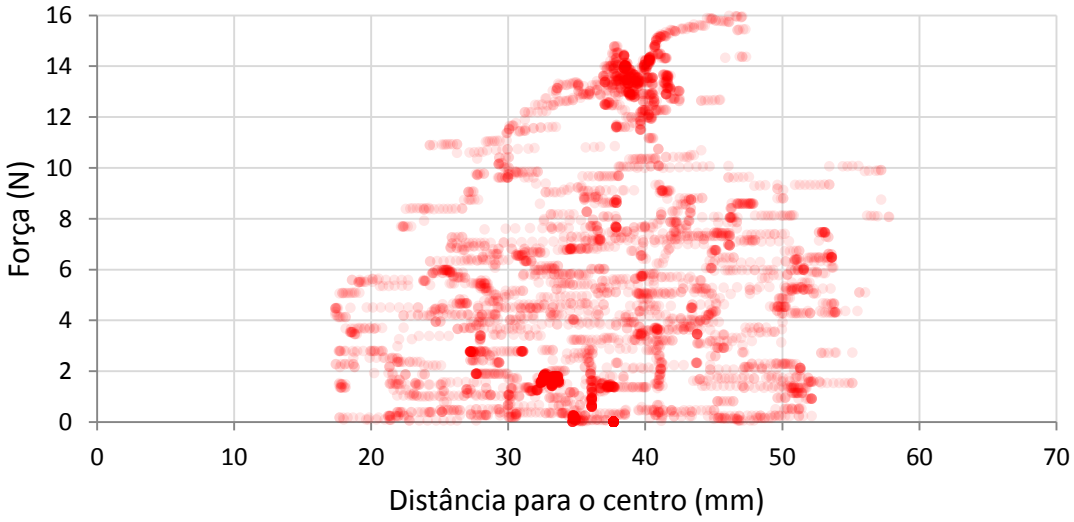


Figura 75 - Gráfico de relação entre raio e força (controlador PI).
Fonte: Própria

Voltando ao teste do controlador proporcional, o mesmo gráfico tridimensional foi gerado para o controle de posição. Entretanto, a amostragem não foi suficiente para

produzir um bom resultado, de modo que apenas pode se identificar o aumento das forças no limite externo da plataforma.

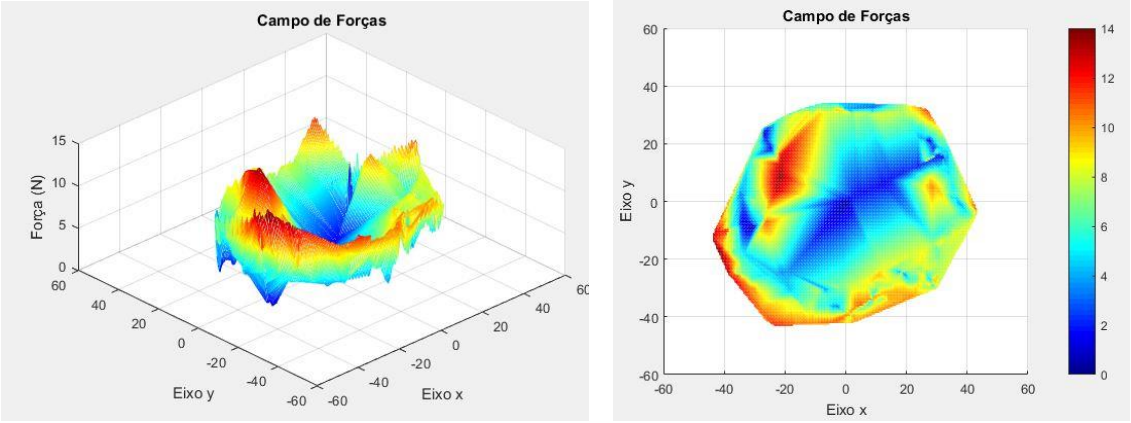


Figura 76 - Gráficos 3D e vista superior da força lida por posição (x,y).
Fonte: Própria

O gráfico abaixo mostra o direcionamento das forças durante o teste com controlador proporcional. Pode-se identificar que as forças direcionam o usuário até o raio idal (médio), mas nem sempre estas forças tem direção puramente radial, como desejado. Também não é fácil observar o dimensionamento das forças, principalmente porque não foi possível alcançar raios mais distantes do médio nos exercícios de controle de posição.

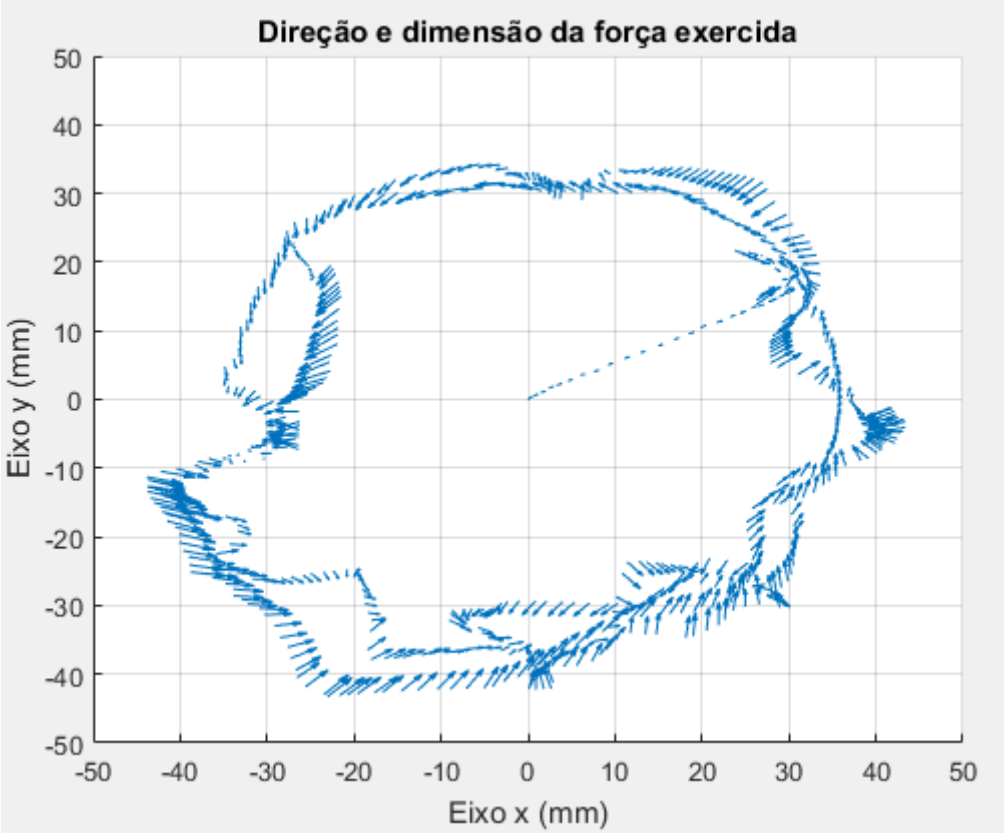


Figura 77 - Gráfico de direcionamento de forças real.
Fonte: Própria

De modo geral, o controle de posição para implementação do sistema háptico não se mostrou satisfatório frente à proposta do projeto. Foi possível dimensionar a força na manopla com certa dificuldade, de modo que surgiram forças em direções não desejadas. Não foi possível dimensionar as forças pelo código, sendo necessário ajustar os ganhos proporcionais dos controladores nos drivers para cada intensidade de exercício. O campo gravitacional vislumbrado não foi facilmente identificado, principalmente devido à rigidez do movimento. Este modo de controle seria adequado apenas para pacientes com deficiências leves, visto que a exigência de força e energia é muito maior. Outro ponto negativo é o de que devido às maiores forças (e portanto maior nível de corrente) no motor, existe maior dissipação de energia e por consequência, os motores se aqueciam mais rapidamente, limitando o exercício a alguns minutos.

XI.iii. Análise Técnica

O dispositivo em funcionamento foi demonstrado à pesquisadora Thaís Terranova, durante a visita ao instituto Lucy Montoro. Segundo ela, a resistência/assistência ao movimento foi identificada durante o exercício, validando o dimensionamento e direcionamento das forças. Para ela, o processo de desenvolvimento do dispositivo está no caminho correto, sendo necessários alguns ajustes. Segundo a pesquisadora, os maiores pontos positivos são o aumento da portabilidade (permitindo o tratamento fora do instituto), o maior estímulo e motivação proporcionados pelo jogo e a capacidade de variar e dosar a força durante as sessões de tratamento. Thaís apontou a necessidade de 2 botões de emergência no dispositivo (um para o paciente e um para o responsável) e de relatórios de acompanhamento objetivos, principalmente com dados de força e alcance dos pacientes. Como críticas, a pesquisadora apontou a existência de pequenos trancos durante os movimentos. Também foi apontada a necessidade de aumentar a capacidade de geração de força do dispositivo, a fim de permitir a ampliação da aplicabilidade do equipamento desenvolvido.

XI.iv. Análise do Software

A seguir, serão apresentados os resultados obtidos e que ainda não foram discutidos previamente. Além disso, serão mencionados os problemas encontrados e como foram analisados e tratados.

Ambos os níveis (Iniciante e Avançado) foram desenvolvidos com a mesma lógica, tendo como diferença apenas as superfícies descritas previamente. Abaixo seguem duas imagens que mostram os cenários finais.

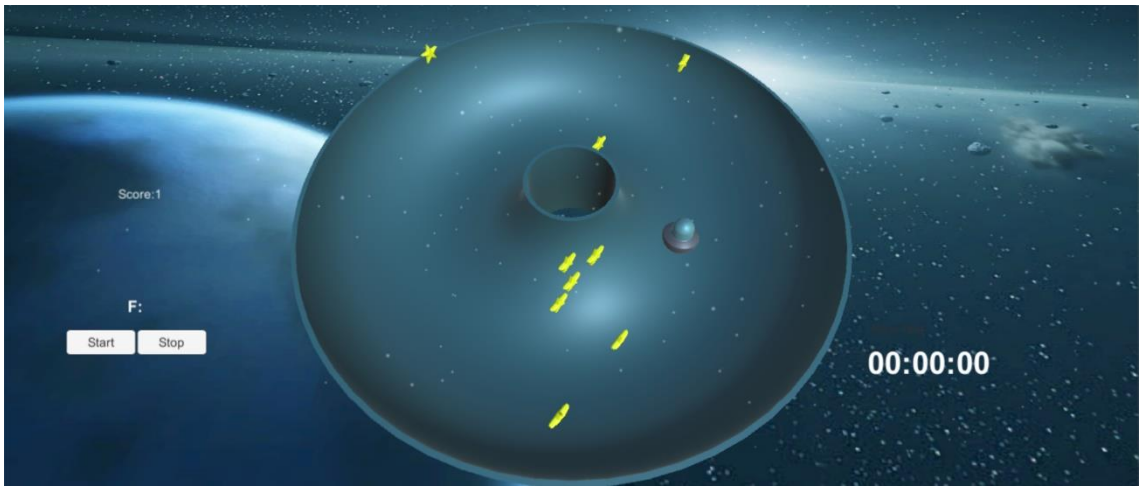


Figura 78 - Cenário final para nível Iniciante.
Fonte: própria

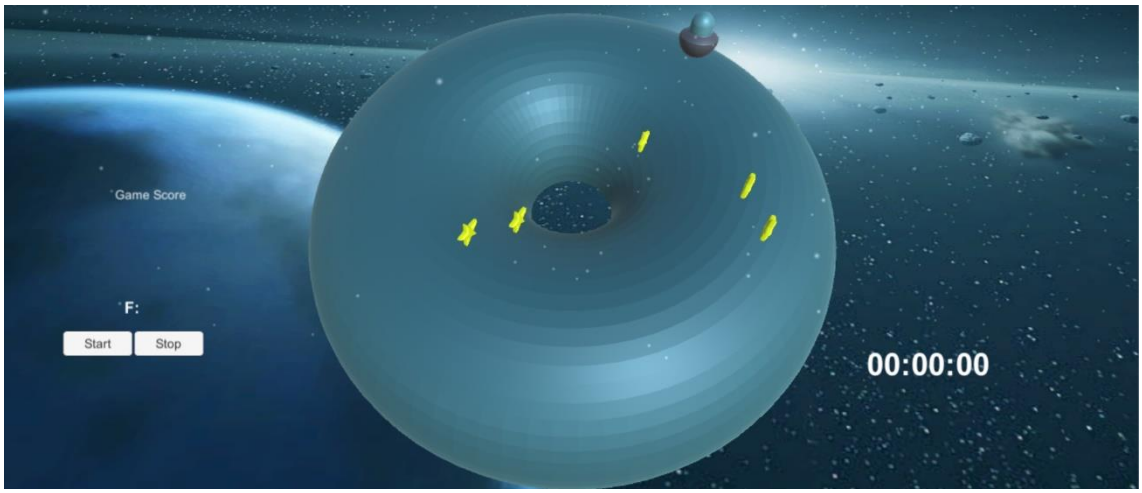


Figura 79 - Cenário final para nível Avançado.
Fonte: própria

Notemos que as estrelas estão em posições aleatórias na superfície. Isso foi feito para efeito de teste apenas. A ideia é distribuir as estrelas coletáveis em posições específicas para estimular o usuário a seguir um trajeto ideal de exercício.

Para os cenários finais, um dos problemas encontrados se trata do contato entre objetos. Em alguns casos, a nave e as estrelas acabam atravessando as superfícies de contato. Geralmente são situações discretas e não perceptíveis, mas em algumas exceções é possível visualizar os corpos muito atravessados, o que pode ser verificado na imagem abaixo. Isso é consequência do formato dos objetos que, por terem sido elaborados no Solidworks, não possuem região de colisão bem definida no Unity. Assim, as únicas soluções encontradas foram redefinir da melhor forma possível essa região e ajustar o posicionamento dos objetos (no caso mostrado abaixo a estrela está em uma posição em que seria impossível de ser coletada, o que então é irrelevante para o jogo).

Figura 80 - Estrela atravessando a superfície.
Fonte: própria

Outro problema encontrado se deu nos momentos em que eram enviadas informações do Unity para o Arduino, pois nem sempre a Placa recebia os dados que eram mandados. Esse fenômeno foi corrigido enviando 3 vezes a informação com um delay intercalando-as.

Fora o que foi citado anteriormente, não houve nenhum outro tipo de problema de funcionamento. Mesmo assim, com intuito de analisar a execução do software por completo, recorreu-se a ferramenta do Unity que é conhecida por "Profiler". Essa permite avaliar diversas características do programa, como o uso da CPU e o consumo de memória. Com isso foi possível entender os pontos com maior tempo gasto e os componentes que dependem de mais memória. A imagem a seguir mostra o que foi obtido.

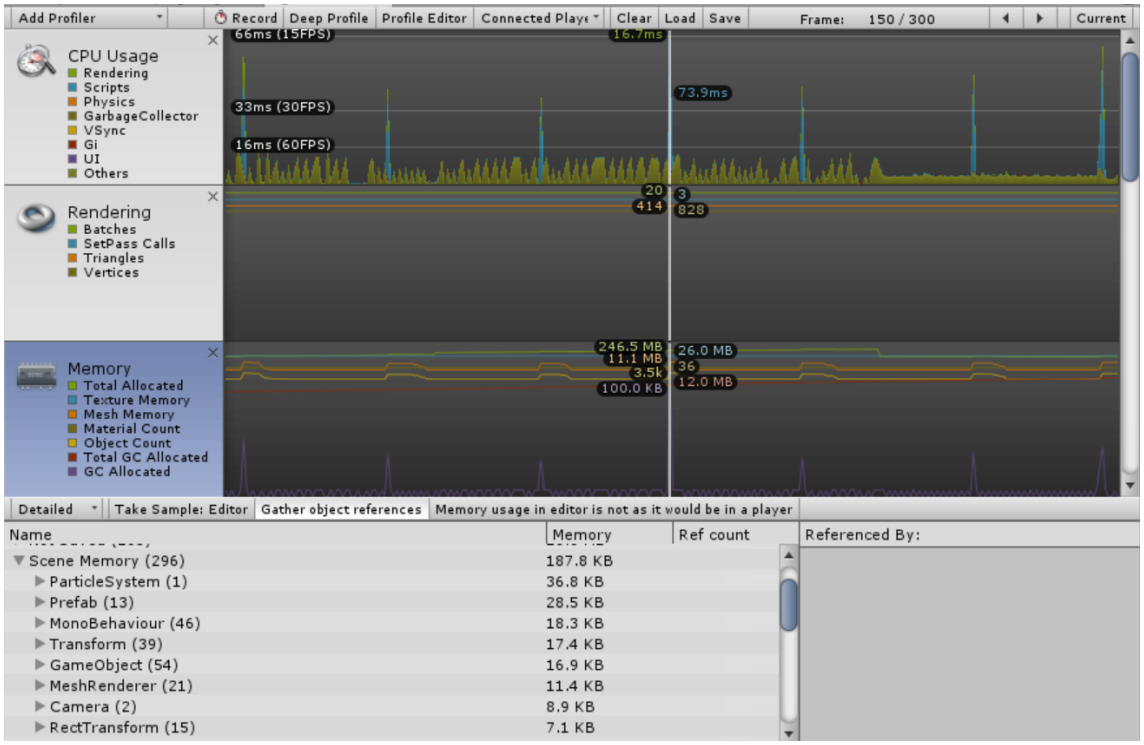


Figura 81 - Janela "Profiler" com informações do jogo executado.
Fonte: própria

Tendo sido finalizada cada etapa do software (login, menu e jogo), juntou-se o conjunto de cenários e criou-se o arquivo executável para testar o programa completo. De um modo geral, não foram encontrados demais problemas que atrapalhassem o desempenho do jogo. Como comentado na análise técnica, o software elaborado se mostrou conceitualmente adequado, necessitando apenas de aperfeiçoamentos futuros para que possa se tornar um produto.

XI.v. Análise de Viabilidade Econômica

Para esta análise, foram registrados os custos incorridos pelos integrantes e estimados outros custos relevantes para o projeto. O objetivo primário foi sugerir um preço de mercado para o produto proposto. Este produto tem como base o protótipo desenvolvido, porém alguns componentes foram substituídos por outros, considerados como ideais pelos integrantes. Também foi feita uma análise de rentabilidade a partir do preço sugerido, a fim de projetar a lucratividade de uma eventual empreitada. Esta análise considera um "reprojeto" total do protótipo, a fim de adequá-lo ao seu mercado.

Os custos do dispositivo podem ser divididos entre variáveis (que são proporcionais ao número de equipamentos produzidos) e fixos (que são diluídos pelo número de produtos). Para os custos fixos, foi estimada a produção de 20 interfaces hápticas.

A conta de matéria-prima inclui todo material utilizado para construir o protótipo, tais como MDF, alumínio, parafusos, rolamentos e outras peças. Grande parte desse valor foi obtido a partir de registros durante o projeto. A conta de usinagem estima a remuneração de um técnico, considerando um tempo de produção de uma semana para as peças (40h) e uma remuneração de 20 R$/hora. Os drivers EPOS2 utilizados foram precificados segundo a loja online da Maxon (traduzindo francos suíços para a moeda brasileira na data de 31/10/2017). Os motores listados no custo correspondem aos ideais, calculados no capítulo de Projeto Mecânico, encontrados na mesma loja online (também traduzindo as moedas, na mesma data). Por se tratarem de importações, os drivers e motores estão atrelados a um custo de tributação de importação. Com base em pesquisas, o valor de taxa de importação de 2 drivers e 2 motores foi estimado em 60% do preço dos produtos. Grande parte da conta de frete também surge dessa importação. Finalizando o escopo dos custos variáveis, o preço da placa microprocessadora foi obtido com base de valor de mercado. Vale ressaltar que por decisão dos autores, o computador não seria incluso no produto, permitindo o comprador utilizar qualquer computador pessoal.

Os custos fixos incluem os projetistas mecatrônico e de software, além das ferramentas adquiridas e do aluguel de um espaço para desenvolvimento. Para o engenheiro, foi considerado um salário mensal de R$ 20 mil e uma carga horária total de 200h para este projeto. Para o designer de jogos, um salário de R$ 15 mil e uma carga horária total de 100h no projeto. Para ferramentas, foi disponibilizada uma verba de R$ 3 mil. O aluguel foi calculado com base em um espaço de 1.000 m², com aluguel de mercado de R$ 20/m². Vale ressaltar que os custos fixos foram diluídos na produção estimada de 20 unidades.

Tabela de Custos	1 Und (R$)	%
Matéria-Prima	647	3%
Usinagem	800	4%
Drivers EPOS2	3.439	17%
Motores, Reduções e Encoders Maxon	4.118	20%
Impostos de Importação	5.062	25%
Fretes	500	2%
Arduino Mega	70	0%
Subtotal Variáveis	14.637	72%
Engenheiro (200h)	1.163	6%
Designer de Software (100h)	434	2%
Ferramentas	150	1%
Aluguel de Galpão	4.000	20%
Subtotal Fixos	5.747	28%
Total	20.383	100%

Tabela 6 – Estimativa de custos do protótipo.
Fonte: Prórpia

Com base no custo de aproximadamente R$ 20 mil por unidade e no valor de aquisição dos dispositivos similares no mercado, foi sugerido um preço de R$ 30 mil, o que corresponde a uma margem de venda de 50%. A tabela abaixo mostra a projeção da rentabilidade para três cenários diferentes: produção de 1, 20 e 50 dispositivos. A receita bruta foi estimada com base na venda de 100% dos produtos fabricados. A métrica de custos de mercadoria foi igual à descrita acima. As despesas gerais são custos fixos não relacionados à produção e as com vendas estão relacionadas à comissão e marketing. O último gasto é o de imposto de renda sobre o lucro, estimado em 25% do lucro antes do IR. Caso sejam vendidos 50 produtos, a margem de lucro líquido seria 28%, principalmente por causa da diluição de custos e despesas. Dada esta análise,

apenas a proposta de fabricação destes dispositivos em larga escala é economicamente viável, sendo o maior desafio encontrar uma demanda dos dispositivos num período de forte recessão no país. Outro fator que vai contra a expectativa é o baixo investimento em qualidade de vida e saúde, apontado pela fisioterapeuta Maria.

Análise de Rentabilidade	1 Und (R$)		20 Und (R$)		50 Und (R$)	
Receita Bruta	30.000	111%	600.000	111%	1.500.000	111%
(-) Impostos sobre venda	-3.000	-11%	-60.000	-11%	-150.000	-11%
Receita Líquida	27.000	100%	540.000	100%	1.350.000	100%
(-) Custo da Mercadoria Vendida	-20.383	-75%	-347.665	-64%	-786.764	-58%
Lucro Bruto	6.617	25%	192.335	36%	563.236	42%
(-) Despesas Gerais	-5.000	-19%	-5.000	-1%	-5.000	0%
(-) Despesas com Vendas	-1.000	-4%	-20.000	-4%	-50.000	-4%
Lucro antes do IR	617	2%	167.335	31%	508.236	38%
(-) Imposto de Renda	-154	-1%	-41.834	-8%	-127.059	-9%
Lucro Líquido	463	2%	125.502	23%	381.177	28%

Tabela 7 – Projeção de resultado
Fonte: Própria

XII. Conclusões e Desenvolvimentos Futuros

As principais motivações no desenvolvimento de dispositivos como o mostrado neste trabalho são aumentar a eficiência na reabilitação motora, a partir do conceito de neuroreabilitação e do desenvolvimento da cognição, além de garantir maior motivação emocional. Com base neste projeto, foi observado que a partir de investimentos e incentivos na área de tecnologia, é viável desenvolver um dispositivo háptico para reabilitação que seja leve, barato e eficaz no tratamento fisioterápico. O protótipo desenvolvido poderá ser utilizado como bancada para novos testes, impulsionando o surgimento de novos mecanismos nesse segmento.

Na visão do grupo, um dos maiores aspectos positivos do protótipo foi o desenvolvimento mecânico, desde o projeto até a fabricação e montagem. A escolha dos ajustes forçados, dos rolamentos autocompensadores e do posicionamento de parafusos resultou num mecanismo bastante robusto e com baixíssima folga. O pouco atrito que pode ser identificado aparece principalmente das caixas de redução planetárias e do encaixe delas nos motores. Alguns problemas surgiram pelo desalinhamento de furos e dificuldade, mas foram rapidamente contornados.

Na implementação do software desenvolvido, foi mostrado que o controle de corrente é suficiente para dimensionar e direcionar as forças no mecanismo fabricado. O campo de forças proposto foi identificado pelos usuários que testaram o dispositivo, validando a interação do mecanismo com a lógica do jogo. Em contrapartida, o controle de posição não se mostrou satisfatório e foi descartado como solução para o produto final. Com relação ao jogo, os estímulos visual e sonoro garantiram uma experiência bastante interessante para os usuários. Na visão dos integrantes, os aspectos lúdicos e o desafio de vencer o jogo aumentariam significativamente a motivação de um paciente para fazer exercícios de fisioterapia.

É importante ressaltar que existem alguns aspectos a serem revisados para dar continuidade ao desenvolvimento de um produto. O primeiro deles está relacionado aos motores, que na visão dos integrantes não geraram força compatível com outros dispositivos no mercado. Assim sendo, o ideal seria substituir os atuadores utilizados pelos motores definidos através da análise dinâmica do mecanismo, mostrada na seção de projeto. Esta substituição garantiria maior capacidade de geração de força na manopla e um aquecimento mais lento dos motores. Apesar deste fato, foi mostrado que a partir das técnicas de controle utilizadas com o driver é possível gerar feedback tátil

que segue especificações predefinidas, e que a força gerada até este momento seria suficiente para o tratamento de pacientes com maiores graus de deficiência.

Outro ponto a ser observado diz respeito ao dimensionamento e à estrutura do mecanismo. Na montagem do protótipo observou-se que houve redução da área de trabalho prevista para o mecanismo por conta da não consideração dos parafusos e porcas que estariam presentes na estrutura montada. Isso não prejudicou a análise de eficiência do projeto, mas reduziu a área de operação em que os usuários poderiam movimentar a manopla. Seria interessante então que os elos tivessem maior comprimento, a fim de ser criada uma área de trabalho maior. Vale ressaltar que o aumento da área requer aumento na capacidade dos motores, sendo esta a maior limitação na área usada neste projeto.

A segurança é um dos principais aspectos dentro do universo de tratamentos e saúde. Toda nova tecnologia deve garantir total segurança e conforto para que ela seja aceita, tanto pelos órgãos reguladores, quanto pelos usuários finais. No projeto, todo o módulo de segurança é implementado por software, limitando a corrente nos motores e, portanto, o torque. É de extrema importância que no produto final também haja um botão de emergência e algum sistema físico de amortecimento.

Com relação ao projeto de software, foram separados alguns dos principais fatores que poderiam ser otimizados para um produto final. Na visão dos autores, para o jogo seria interessante a implementação de uma maior quantidade de níveis de atividade, de modo a explorar diferentes tipos de exercícios para diferentes tipos de limitações de movimentação, possibilitando a evolução gradual no âmbito da reabilitação. Para a base de dados, vale lembrar que foi utilizada a plataforma Xampp para criar uma base de acesso local. Assim, seria necessário desenvolver uma arquitetura de armazenamento mais sofisticada para compor o produto final, além de uma interface voltada para os fisioterapeutas, para análise do histórico de exercícios.

A proposta de desenvolvimento de uma interface háptica funcional aliada a um jogo virtual foi cumprida dentro do cronograma previsto. Foi possível criar um dispositivo de baixo custo e com peso reduzido, o que facilitaria o acesso da população e instituições ao produto final. Apesar de muitas dificuldades ao longo do projeto e de pontos a serem melhorados no futuro, para o grupo, o resultado final do trabalho foi bastante satisfatório.

XIII. Bibliografia

BUCHALLA, Cassia Maria. A classificação internacional de funcionalidade, incapacidade e saúde. Acta Fisiátrica, v. 10, n. 1, p. 29-31, 2016.

BURDEA, G., COIFFET, P. "Virtual Reality Technology", John Wiley & Sons, p.2. 1994.

BRASIL. Decreto Nº 5.296, de 02 de dezembro de 2004.

CAMEIRÃO, M.S., i BADIA, S.B., DUARTE, E., FRISOLI, A. and VERSCHURE, P.F., 2012. The combined impact of virtual reality neurorehabilitation and its interfaces on upper extremity functional recovery in patients with chronic stroke. Stroke, 43(10), pp.2720-2728.

CAMPION, Gianni. "The pantograph MK-II: a haptic instrument." The Synthesis of Three Dimensional Haptic Textures: Geometry, Control, and Psychophysics. Springer London, 2005. 45-58.

CENSO DEMOGRÁFICO 2010. Características gerais da população, religião e pessoas com deficiência. Rio de Janeiro: IBGE, 2012. Acesso em maio de 2017.

CHARLES, S.K, KREBS, H.I, VOLPE, B.T, LYNCH, D, HOGAN, N. "Wrist Rehabilitation Following Stroke: Initial Clinical Results" International Conference on Rehabilitation Robotics, 2005

CHEN, D. Touch Me, Heal Me: Haptic Solutions for Rehabilitation. <http://www.medicaldesignbriefs.com/component/content/article/mdb/features/8938>. 2011

CHRIST, O. and REINER, M. Perspectives and possible applications of the rubber hand and virtual hand illusion in non-invasive rehabilitation: Technological improvements and their consequences. Neuroscience & Biobehavioral Reviews, 44, pp.33-44. 2014.

COSTA, R. M. E. M. da. Ambientes Virtuais na Reabilitação Cognitiva de Pacientes Neurológicos e Psiquiátricos. 2000.

DA SILVA, Jorge Miguel Ferreira. Estudo e desenvolvimento de um dispositivo háptico de dois graus de liberdade baseada em atuadores de corrente contínua. 2014.

DOS SANTOS NUNES, Fátima de Lourdes et al. Realidade Virtual para saúde no Brasil: conceitos, desafios e oportunidades. Rev. Bras. Eng. Biom, v. 27, n. 4, p. 243-258, 2011.

DRUMMOND, R. et al. A Estimulação cognitiva de Pessoas com Transtorno Autista através de Ambientes Virtuais. 2002.

FASOLI, Susan E. et al. Effects of robotic therapy on motor impairment and recovery in chronic stroke. Archives of physical medicine and rehabilitation, v. 84, n. 4, p. 477-482, 2003.

FLUET, G.G. and DEUTSCH, J.E. Virtual reality for sensorimotor rehabilitation post-stroke: the promise and current state of the field. Current physical medicine and rehabilitation reports, 1(1), pp.9-20. 2013

FISCHER, Franz. "O efeito da intervenção com realidade virtual em indivíduos com dificuldades de coordenação motora.": 50-f. 2013.

KAFADER, Urs. Motion Control for Newbies. Maxon Academy, 2007. 132 páginas.

KREBS, H. Igo et al. Robot-aided neurorehabilitation. IEEE transactions on rehabilitation engineering, v. 6, n. 1, p. 75-87, 1998.

KREBS, Hermano I. et al. Rehabilitation robotics: pilot trial of a spatial extension for MIT-Manus. Journal of NeuroEngineering and Rehabilitation, v. 1, n. 1, p. 1, 2004.

KREBS, Hermano I. Effects of robot-assisted therapy on upper limb recovery after stroke: a systematic review. Neurorehabilitation and neural repair, v. 22, n. 2, p. 111-121, 2008.

KREBS, Hermano Igo, Dylan Edwards, and Neville Hogan. "Forging Mens et Manus: The MIT Experience in Upper Extremity Robotic Therapy." Neurorehabilitation Technology. Springer International Publishing, 2016. 333-350.

LAVATELLI, Alberto, Francesco FERRISE, and Monica BORDEGONI. "Design of an open-source low cost 2DOF haptic device." Mechatronic and Embedded Systems and Applications (MESA), 2014 IEEE/ASME 10th International Conference on. IEEE, 2014.

LEITE, J. M. R. S.; PRADO, GF do. Paralisia cerebral: aspectos fisioterapêuticos e clínicos. Revista Neurociências, v. 12, n. 1, p. 41-45, 2004.

MAIA, Daniela C. et al. Projetando serious games para tratamento do controle de tronco em pacientes com avc. In: XIII Workshop de Informática Médica (WIN'13), CSBC. 2013.

MASSIE, Thomas H. et al. The phantom haptic interface: A device for probing virtual objects. In: Proceedings of the ASME winter annual meeting, symposium on haptic interfaces for virtual environment and teleoperator systems. 1994. p. 295-300.

MAXON MOTORS. Application Note "CANopen Basic Information". EPOS Positioning Controllers. Edição de maio de 2008. 15 páginas.

MAXON MOTORS. Application Note "Device Programming". EPOS Positioning Controllers. Edição de abril de 2009. 15 páginas.

MAXON MOTORS. Command Library. EPOS Positioning Controllers. Edição de maio de 2017. ID do documento: rel7138. 178 páginas.

MAXON MOTORS. Communication Guide. EPOS2 Positioning Controllers. Edição de abril de 2013. ID do documento: rel4054. 54 páginas.

MONTEIRO JUNIOR, R. S. et al. Efeito da Reabilitação Virtual em diferentes tipos de tratamento. 2011.

MORITZ, Chet et al. 'Neurogame Therapy'for Improvement of Movement Coordination after Brain Injury: Developing a Wireless Biosignal Game Therapy System. In: Global Humanitarian Technology Conference (GHTC), 2011 IEEE. IEEE, 2011. p. 72-77.

ORGANIZAÇÃO MUNDIAL DA SAÚDE, International Classification of
Impairments, Disabilities, and Handicaps. Geneva, 1980.
and Handicaps

O'SULLIVAN, Susan B.; SCHMITZ, Thomas J.; FULK, George. Physical rehabilitation. FA Davis, 2013, p.2-6

RODRIGUES, M.A.F. et al. LERDORT: Um Serious Game para Correção de Desvios Posturais e Fixação de Sequências de Alongamento. 2012.

SARI, Franciele Leiliane; MARCON, Sonia Silva. Participação da família no trabalho fisioterapêutico em crianças com paralisia cerebral. Journal of Human Growth and Development, v. 18, n. 3, p. 229-239, 2008.

SILVA, Emanuel de Jesus Alves da. Reabilitação após o AVC. 2011.

WAUKE, Ana Paula T.; COSTA, Rosa Maria EM; CARVALHO, Luis Alfredo V. de. VESUP: O uso de Ambientes Virtuais no tratamento de Fobias Urbanas. In: IX Congresso Brasileiro de Informática em Saúde, Ribeirao Preto, SP, Brasil. 2004.

XIV. Apêndices

XIV.I. Apêndice A – Código de Matlab

a. Código para encontrar a área útil

```
n = 1000;

PB = 0.20;
BO = 0.15;

hor = ones(1,n);
ver = ones(n,1);

th1 = linspace(-40*pi/180,40*pi/180,n);
th2 = linspace(50*pi/180,130*pi/180,n);

TH1 = ver*th1;
TH2 = th2'*hor;

x = cos(TH1)*PB+cos(TH2)*BO;
y = sin(TH1)*PB+sin(TH2)*BO;

scatter(x(:),y(:),'.')
axis equal
axis([-0.05 0.4 -0.05 0.3])
title("Área útil do mecanismo")
xlabel("Eixo x (m)")
ylabel("Eixo y (m)")
grid

hold on

rai = linspace(0,0.075,n*1000);
ang = linspace(0,2*pi*1000,n*1000);

x = cos(ang).*rai+0.20;
y = sin(ang).*rai+0.15;

scatter(x(:),y(:),'.','r')
grid
title("Área de trabalho do mecanismo e do jogo")
xlabel("Eixo x (m)")
ylabel("Eixo y (m)")
```

b. Código para encontrar as curvas torque-rotação e potência-rotação

```
n = 10000;
Fmax = 40;
vmax = 1;

rai = linspace(0,0.075,n);
ang = linspace(0,2*pi*10,n);

x = cos(ang).*rai+0.20;
y = sin(ang).*rai+0.15;
```

```
PB = 0.20;
BO = 0.15;
OP = (x.^2+y.^2).^0.5;

phi = acos((PB^2+BO^2-OP.^2)/(2*PB*BO));
bet = acos((-PB^2+BO^2+OP.^2)./(2*OP*BO));

th1 = phi-pi+bet+acos(x./OP);
th2 = bet+acos(x./OP);

ang2 = linspace(0,2*pi,n/10);

Fx = cos(ang2)*Fmax;
Fy = sin(ang2)*Fmax;
vx = cos(ang2)*vmax;
vy = sin(ang2)*vmax;

T1 = (sin(th1)*(PB))'*Fx - (cos(th1)*(PB))'*Fy;
T2 = (sin(th2)*(BO))'*Fx - (cos(th2)*(BO))'*Fy;

w1 = (tan(th1)./(BO*(cos(th2).*tan(th1)-sin(th2))))'*vy +
(1./(BO*(cos(th2).*tan(th1)-sin(th2))))'*vx;
w2 = (tan(th2)./(PB*(cos(th1).*tan(th2)-sin(th1))))'*vy +
(1./(PB*(cos(th1).*tan(th2)-sin(th1))))'*vx;

figure(1)
scatter(abs(w1(:)*30/pi),abs(T1(:)),'.','b')
grid
axis([0 100 0 8])
title("Curva torque-rotação do motor 1")
xlabel("Rotação (rpm)")
ylabel("Torque (N.m)")

figure(2)
scatter(abs(w2(:)*30/pi),abs(T2(:)),'.','g')
grid
axis([0 100 0 8])
title("Curva torque-rotação do motor 2")
xlabel("Rotação (rpm)")
ylabel("Torque (N.m)")

figure(3)
scatter(abs(w1(:)*30/pi),abs(w1(:).*T1(:)),'.','b')
grid
axis([0 100 0 40])
title("Curva potência-rotação do motor 1")
xlabel("Rotação (rpm)")
ylabel("Potência (W)")

figure(4)
scatter(abs(w2(:)*30/pi),abs(w2(:).*T2(:)),'.','g')
grid
axis([0 100 0 40])
title("Curva potência-rotação do motor 2")
xlabel("Rotação (rpm)")
ylabel("Potência (W")
```

XIV.II. Apêndice B – Código do Arduino

```
1  #include <SPI.h>
```

```
 2
 3  //************************************************************************
 4  //------- Laboratorio de Biomecatronica @ Universidade de Sao Paulo --------
 5  //
 6  // Version: 1.0
 7  // Date: 10.08.2017
 8  //
 9  //************************************************************************
10
11  #include "SPI.h"
12  #include "math.h"
13  #include "mcp_can.h"
14  #include "mcp_can_dfs.h"
15
16  #define Startup               1
17  #define PDOConfiguration       2
18  #define Operational           3
19  #define OperationalControl 4
20  #define GoHome 8
21
22  #define PositionControl 0
23  #define CurrentControl 1
24  #define NoControl 2
25
26  #define MaxCurrent 3000
27  #define CurrentThreshold 3000
28
29  byte State = Startup;
30  const int SPI_CS_PIN = 9;
31  MCP_CAN CAN(SPI_CS_PIN);
32
33  // Sensor variables
34  uint32_t encoder_data = 0;
35  uint16_t current_data = 0;
36  uint32_t actualposition_data = 0;
37  bool sync_flag = 0;
38  bool start = false;
39
40  int controlmode = NoControl;
41
42  // ***************************
43  // EPOS2 CANOpen Communication
44  // ***************************
45
46  // Statemachine
47  unsigned char set_preoperational[2] = {0x80, 0};
48  unsigned char set_operational[2] = {0x1, 0};
49  unsigned char enable_epos[8] = {0x2B, 0x40, 0x60, 0, 0x0F, 0, 0, 0};
50  unsigned char disable_epos[8] = {0x2B, 0x40, 0x60, 0, 0x06, 0, 0, 0};
51  unsigned char current_mode[8] = {0x22, 0x60, 0x60, 0, 0xFD, 0, 0, 0};
52  unsigned char position_mode[8] = {0x22, 0x60, 0x60, 0, 0xFF, 0, 0, 0};
53
54  // PDO Configuration
55  unsigned char pdo_sync[1] = {0x00};
56
57  unsigned char pdo_actual_position_1_1[8] = {0x22, 0x02, 0x18, 0x01, 0x81, 0x03, 0,
0};
58  unsigned char pdo_actual_position_1_2[8] = {0x22, 0x02, 0x18, 0x02, 0x01, 0, 0, 0};
59  unsigned char pdo_actual_position_2_1[8] = {0x22, 0x02, 0x18, 0x01, 0x82, 0x03, 0,
0};
60  unsigned char pdo_actual_position_2_2[8] = {0x22, 0x02, 0x18, 0x02, 0x01, 0, 0, 0};
61
62  unsigned char pdo_actual_current_1[8] = {0x22, 0x00, 0x1A, 0x00, 0x00, 0, 0, 0};
63  unsigned char pdo_actual_current_2[8] = {0x22, 0x00, 0x1A, 0x01, 0x10, 0, 0x78,
0x60};
64  unsigned char pdo_actual_current_3[8] = {0x22, 0x00, 0x1A, 0x02, 0x10, 0, 0x27,
0x20};
65  unsigned char pdo_actual_current_4[8] = {0x22, 0x00, 0x18, 0x02, 0x01, 0, 0, 0};
66  unsigned char pdo_actual_current_5[8] = {0x22, 0x00, 0x1A, 0x00, 0x01, 0, 0, 0};
67
68  // *****************
69  // Object Writing
70  // *****************
71
72  unsigned char set_max_following_error[8] = {0x22, 0x65, 0x60, 0, 0x08, 0x07, 0, 0};
//1800
73  unsigned char set_max_acceleration[8] = {0x22, 0xC5, 0x60, 0, 0x05, 0, 0, 0}; //5
74  unsigned char set_max_profile_velocity[8] = {0x22, 0x7F, 0x60, 0, 0x64, 0, 0, 0};
//100
```

```
 75  unsigned char set_min_position_limit[8] = {0x22, 0x7D, 0x60, 0x01, 0xF8, 0xF8,
0xFF, 0xFF}; // -1800qc = 0xFFFFF8F8
 76  unsigned char set_max_position_limit[8] = {0x22, 0x7D, 0x60, 0x02, 0x08, 0x07, 0,
0}; // 1800qc = 0x00000708
 77
 78  unsigned char set_PID1_P[8] = {0x22, 0xFB, 0x60, 0x01, 0x40, 0x06, 0, 0};
 79  unsigned char set_PID1_I[8] = {0x22, 0xFB, 0x60, 0x02, 0x18, 0x00, 0, 0};
 80  unsigned char set_PID1_D[8] = {0x22, 0xFB, 0x60, 0x03, 0, 0, 0, 0};
 81  unsigned char set_PID2_P[8] = {0x22, 0xFB, 0x60, 0x01, 0x58, 0x02, 0, 0};
 82  unsigned char set_PID2_I[8] = {0x22, 0xFB, 0x60, 0x02, 0x08, 0, 0, 0};
 83  unsigned char set_PID2_D[8] = {0x22, 0xFB, 0x60, 0x03, 0, 0, 0, 0};
 84
 85  unsigned char setpoint_position[8] = {0x22, 0x62, 0x20, 0x00, 0, 0, 0, 0};
 86  unsigned char setpoint_current[8] = {0x22, 0x30, 0x20, 0x00, 0, 0, 0, 0};
 87
 88  // *****************
 89  // Object Reading
 90  // *****************
 91
 92  unsigned char get_actual_position[8] = {0x40, 0x64, 0x60, 0, 0, 0, 0, 0};
 93  unsigned char get_actual_velocity[8] = {0x40, 0x6C, 0x60, 0, 0, 0, 0, 0};
 94  unsigned char get_actual_current[8] = {0x40, 0x78, 0x60, 0, 0, 0, 0, 0};
 95  unsigned char get_demand_current[8] = {0x40, 0x31, 0x20, 0, 0, 0, 0, 0};
 96
 97  // ******************
 98  // System Parameters
 99  // ******************
100
101  double x = 0;
102  double y = 0;
103  double z = 0;
104  double r = 0;
105  double o = 0;
106  double OP = 0;
107  double dif = 0; // R-r
108  double intensidade = 1; // Ganho da corrente
109  double Fmax = 10; // forï¿½a mï¿½xima
110  double dr = 0;
111  double F = 0;
112  double fx = 0;
113  double fy = 0;
114  double sentido = 3.1415;
115
116  double ang1 = 0;
117  double ang2 = 0;
118  double tor1 = 0;
119  double tor2 = 0;
120  int cur1 = 0;
121  int cur2 = 0;
122  long enc1 = 0;
123  long enc2 = 0;
124
125  double angaux = 0;
126
127  double ang1_demand = 0;
128  double ang2_demand = 0;
129  double x_demand = 0;
130  double y_demand = 0;
131  double cur1_demand = 0;
132  double cur2_demand = 0;
133  double tor1_demand = 0;
134  double tor2_demand = 0;
135  double enc1_demand = 0;
136  double enc2_demand = 0;
137  double F_demand = 0;
138  double r_demand = 0;
139
140  long offset1 = 0;
141  long offset2 = 0;
142  long limsup;
143  long liminf;
144  uint32_t auxHome;
145  bool Home;
146
147  String msg_Unity;
148  char comando_unity;
149
150  //********************
151  // EPOS COMMUNICATION
```

```
152 //********************
153
154 void doStartup(void)
155 {
156     // Configuration EPOS node 1..................
157     CAN.sendMsgBuf(0x601, 0, 8, current_mode);
158     delay(10);
159     CAN.sendMsgBuf(0x601, 0, 8, disable_epos);
160     delay(10);
161     CAN.sendMsgBuf(0x601, 0, 8, enable_epos);
162     delay(10);
163     CAN.sendMsgBuf(0x601, 0, 8, set_max_following_error);
164     delay(10);
165     CAN.sendMsgBuf(0x601, 0, 8, set_max_acceleration);
166     delay(10);
167     CAN.sendMsgBuf(0x601, 0, 8, set_max_profile_velocity);
168     delay(10);
169     CAN.sendMsgBuf(0x601, 0, 8, set_min_position_limit);
170     delay(10);
171     CAN.sendMsgBuf(0x601, 0, 8, set_max_position_limit);
172     delay(10);
173
174     // Configuration EPOS node 2..................
175     CAN.sendMsgBuf(0x602, 0, 8, current_mode);
176     delay(10);
177     CAN.sendMsgBuf(0x602, 0, 8, disable_epos);
178     delay(10);
179     CAN.sendMsgBuf(0x602, 0, 8, enable_epos);
180     delay(10);
181     CAN.sendMsgBuf(0x602, 0, 8, set_max_following_error);
182     delay(10);
183     CAN.sendMsgBuf(0x602, 0, 8, set_max_acceleration);
184     delay(10);
185     CAN.sendMsgBuf(0x602, 0, 8, set_max_profile_velocity);
186     delay(10);
187     CAN.sendMsgBuf(0x602, 0, 8, set_min_position_limit);
188     delay(10);
189     CAN.sendMsgBuf(0x602, 0, 8, set_max_position_limit);
190     delay(10);
191
192     State = PDOConfiguration;
193 }
194
195 void PDOConfig(void)
196 {
197     delay(10);
198     // PDO Configuration EPOS node 1..................
199     CAN.sendMsgBuf(0x00, 0, 2, set_preoperational);
200     delay(10);
201     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_position_1_1);
202     delay(10);
203     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_position_1_2);
204     delay(10);
205     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_current_1);
206     delay(10);
207     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_current_2);
208     delay(10);
209     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_current_3);
210     delay(10);
211     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_current_4);
212     delay(10);
213     CAN.sendMsgBuf(0x601, 0, 8, pdo_actual_current_5);
214     delay(10);
215
216     // PDO Configuration EPOS node 2..................
217     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_position_2_1);
218     delay(10);
219     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_position_2_2);
220     delay(10);
221     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_current_1);
222     delay(10);
223     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_current_2);
224     delay(10);
225     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_current_3);
226     delay(10);
227     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_current_4);
228     delay(10);
229     CAN.sendMsgBuf(0x602, 0, 8, pdo_actual_current_5);
230     delay(10);
```

```
231      CAN.sendMsgBuf(0x00, 0, 2, set_operational);
232      delay(100);
233
234      Serial.println("TPDO Configured!");
235
236      delay(10);
237      State = Operational;
238      Serial.println("State: Waiting for command");
239  }
240
241
242  void setpositioncontrol()
243  {
244      CAN.sendMsgBuf(0x601, 0, 8, position_mode);
245      CAN.sendMsgBuf(0x602, 0, 8, position_mode);
246  }
247
248  void setcurrentcontrol()
249  {
250      CAN.sendMsgBuf(0x601, 0, 8, current_mode);
251      CAN.sendMsgBuf(0x602, 0, 8, current_mode);
252  }
253
254  void positionSetpoint1(int32_t angle)
255  {
256      setpoint_position[4] = angle & 0xFF;
257      setpoint_position[5] = (angle >> 8) & 0xFF;
258      setpoint_position[6] = (angle >> 16) & 0xFF;
259      setpoint_position[7] = (angle >> 24) & 0xFF;
260
261      CAN.sendMsgBuf(0x601, 0, 8, setpoint_position);
262  }
263
264  void positionSetpoint2(int32_t angle)
265  {
266      setpoint_position[4] = angle & 0xFF;
267      setpoint_position[5] = (angle >> 8) & 0xFF;
268      setpoint_position[6] = (angle >> 16) & 0xFF;
269      setpoint_position[7] = (angle >> 24) & 0xFF;
270
271      CAN.sendMsgBuf(0x602, 0, 8, setpoint_position);
272  }
273
274  void currentSetpoint1(int16_t current)
275  {
276      if(current > MaxCurrent)
277          current = MaxCurrent;
278      setpoint_current[4] = current & 0xFF;
279      setpoint_current[5] = (current >> 8) & 0xFF;
280
281      CAN.sendMsgBuf(0x601, 0, 8, setpoint_current);
282  }
283
284  void currentSetpoint2(int16_t current)
285  {
286      if(current > MaxCurrent)
287          current = MaxCurrent;
288      setpoint_current[4] = current & 0xFF;
289      setpoint_current[5] = (current >> 8) & 0xFF;
290
291      CAN.sendMsgBuf(0x602, 0, 8, setpoint_current);
292  }
293
294  void sync(void)
295  {
296      CAN.sendMsgBuf(0x80, 0, 1, pdo_sync);
297  }
298
299  float DataControl()
300  {
301      unsigned char len = 0;
302      unsigned char buf[8];
303
304      if(CAN_MSGAVAIL == CAN.checkReceive())
305      {
306          CAN.readMsgBuf(&len, buf);
307          unsigned int canId = CAN.getCanId();
308
309          switch (canId)
```

```
310         {
311         case 0x181:
312             current_data = buf[1];
313             current_data = (current_data <<= 8) | buf[0];
314             cur1 = current_data;
315             break;
316 
317         case 0x381:
318             actualposition_data = buf[5];
319             actualposition_data = (actualposition_data <<= 8) | buf[4];
320             actualposition_data = (actualposition_data <<= 8) | buf[3];
321             actualposition_data = (actualposition_data <<= 8) | buf[2];
322             enc1 = actualposition_data;
323             break;
324 
325         case 0x182:
326             current_data = buf[1];
327             current_data = (current_data <<= 8) | buf[0];
328             cur2 = current_data;
329             break;
330 
331         case 0x382:
332             actualposition_data = buf[5];
333             actualposition_data = (actualposition_data <<= 8) | buf[4];
334             actualposition_data = (actualposition_data <<= 8) | buf[3];
335             actualposition_data = (actualposition_data <<= 8) | buf[2];
336             enc2 = actualposition_data;
337             break;
338         }
339     }
340 }
341 
342 void doHoming()
343 {
344             auxHome = -1;
345         Home = false;
346 
347         // ******** Homing 1 inï¿½cio ***********
348         while (Home == false)
349         {
350             positionSetpoint1((uint32_t) auxHome);
351             DataControl();
352             DataControl();
353             DataControl();
354 
355             Serial.println(enc1);
356 
357             if(cur1 >= CurrentThreshold || cur1 < -CurrentThreshold)
358             {
359                 Home = true;
360                 Serial.println("first Home 1");
361             }
362             if(enc1 <= auxHome+5 || enc1 >= auxHome-5)
363             {
364                 auxHome += -5;
365             }
366 
367             delay(10);
368             if (sync_flag)
369             {
370                 sync_flag=0;
371                 sync();
372             }
373         }
374         liminf = enc1;
375 
376         Home = false;
377         auxHome = auxHome + 100;
378         positionSetpoint1((uint32_t) auxHome);
379 
380         delay(1000);
381         DataControl();
382         DataControl();
383 
384         while (Home == false)
385         {
386             positionSetpoint1((uint32_t) auxHome);
387             DataControl();
388             DataControl();
```

```
389             DataControl();
390 
391             Serial.println(enc1);
392 
393             if((cur1 > CurrentThreshold || cur1 < -CurrentThreshold) && enc1 >
liminf + 1000)
394             {
395                 Home = true;
396                 Serial.println("second Home 1");
397             }
398             if(enc1 <= auxHome+5 || enc1 >= auxHome-5)
399             {
400                 auxHome += 5;
401             }
402             delay(10);
403 
404             if (sync_flag)
405             {
406                 sync_flag=0;
407                 sync();
408             }
409         }
410         limsup = enc1;
411 
412         delay(100);
413         offset1 = (liminf+limsup)/2;
414         Serial.println("parametros");
415         Serial.println(liminf);
416         Serial.println(limsup);
417         Serial.print((uint32_t) offset1);
418 
419         setcurrentcontrol();
420 
421         while ((enc1>offset1 +20)||(enc1<offset1 -20)){
422                currentSetpoint1((uint16_t) -(enc1-offset1));
423         }
424         setpositioncontrol();
425 
426         delay(100);
427 
428         // ******** Homing 1 fim ***********
429 
430         // ******** Homing 2 inï¿½cio ***********
431 
432         auxHome = -1;
433         Home = false;
434 
435         while (Home == false)
436         {
437             positionSetpoint2((uint32_t) auxHome);
438             DataControl();
439             DataControl();
440             DataControl();
441             Serial.println(cur2);
442             if(cur2 >= CurrentThreshold || cur2 < -CurrentThreshold)
443             {
444                 Home = true;
445                 Serial.println("first Home 2");
446             }
447             if(enc2 <= auxHome+5 || enc2 >= auxHome-5)
448             {
449                 auxHome += -5;
450             }
451 
452             delay(10);
453             if (sync_flag)
454             {
455                 sync_flag=0;
456                 sync();
457             }
458         }
459         liminf = enc2;
460 
461         Home = false;
462         auxHome = auxHome + 100;
463         positionSetpoint2((int32_t) auxHome);
464 
465         delay(1000);
466         DataControl();
```

```
467           DataControl();
468
469           while (Home == false)
470           {
471               positionSetpoint2((int32_t) auxHome);
472               DataControl();
473               DataControl();
474               DataControl();
475
476               Serial.println(cur2);
477
478               if((cur2 > CurrentThreshold || cur2 < -CurrentThreshold) && enc2 >
liminf + 1000)
479               {
480                   Home = true;
481                   Serial.println("second Home 2");
482               }
483               if(enc2 <= auxHome+5 || enc2 >= auxHome-5)
484               {
485                   auxHome += 5;
486               }
487               delay(10);
488
489               if (sync_flag)
490               {
491                   sync_flag=0;
492                   sync();
493               }
494           }
495           limsup = enc2;
496
497           delay(100);
498           offset2 = (liminf+limsup)/2;
499           Serial.print(offset2);
500
501           setcurrentcontrol();
502
503           while ((enc2>offset2 +20)||(enc2<offset2 -20)){
504                  currentSetpoint2((int16_t) -(enc2-offset2));
505           }
506
507           if (offset2 < 500){
508           positionSetpoint2((int32_t) offset2);
509           }// ******** Homing 2 fim ***********
510           delay(100);
511
512           if(controlmode == CurrentControl)
513               setcurrentcontrol();
514
515           if(controlmode == PositionControl)
516               setpositioncontrol();
517
518           delay(100);
519           State = OperationalControl;
520           Serial.println("Program operational");
521   }
522
523   //***********************
524   // INTERRUPTION
525   //***********************
526   ISR(TIMER1_COMPA_vect)
527   {
528       sync_flag = 1;
529   }
530
531   //****************
532   // ARDUINO
533   //****************
534   //----------------------------------------------------------------------
535   void setupIntHap()
536   {
537       Serial.begin(115200);
538
539       while (CAN_OK != CAN.begin(CAN_1000KBPS))              // init can bus :
baudrate = 500k
540       {
541           Serial.println("CAN BUS Shield init fail");
542           Serial.println("Init CAN BUS Shield again");
543           delay(100);
```

```
544      }
545
546      Serial.println("CAN BUS Shield init ok!");
547
548  // TIMER SETUP- the timer interrupt allows precise timed measurements of the reed
switch
549  //for mor info about configuration of arduino timers see
http://arduino.cc/playground/Code/Timer1
550
551      cli();//stop interrupts
552
553  //set timer1 interrupt at 1kHz
554      TCCR1A = 0;// set entire TCCR1A register to 0
555      TCCR1B = 0;// same for TCCR1B\
556      TCNT1  = 0;//initialize counter value to 0
557  // set timer count for 1khz increments
558
559  // OCR1A 1999 to 1kHZ, 19999 to 0.1kHz
560      OCR1A = 19999;// = (16*10^6) / (1000*8) - 1
561  //had to use 16 bit timer1 for this bc 1999>255, but could switch to timers 0 or 2
with larger prescaler
562  // turn on CTC mode
563      TCCR1B |= (1 << WGM12);
564  // Set CS11 bit for 8 prescaler
565      TCCR1B |= (1 << CS11);
566  // enable timer compare interrupt
567      TIMSK1 |= (1 << OCIE1A);
568
569      sei();//allow interrupts
570  //END TIMER SETUP
571
572      Serial.println("Interrupt init ok!");
573  }
574
575  void setPID()
576  {
577              CAN.sendMsgBuf(0x601,     0, 8, set_PID1_P);
578                  delay(10);
579                  CAN.sendMsgBuf(0x601, 0, 8, set_PID1_I);
580                  delay(10);
581                  CAN.sendMsgBuf(0x601, 0, 8, set_PID1_D);
582                  delay(10);
583                  CAN.sendMsgBuf(0x602, 0, 8, set_PID2_P);
584                  delay(10);
585                  CAN.sendMsgBuf(0x602, 0, 8, set_PID2_I);
586                  delay(10);
587                  CAN.sendMsgBuf(0x602, 0, 8, set_PID2_D);
588                  delay(10);
589  }
590
591  //-------------------------------------------------------------------
592  void loopIntHap()
593  {
594
595      switch (State)
596      {
597      case Startup:
598          doStartup();
599          break;
600
601      case PDOConfiguration:
602          PDOConfig();
603          break;
604
605      case Operational:
606
607          if(Serial.available() > 0)
608              comando_unity = Serial.read();
609              if(comando_unity == 's')
610                  if(controlmode!=3)
611                      start = true;
612              if(comando_unity == 'c'){
613                  controlmode = CurrentControl;
614                  setcurrentcontrol();
615                  Serial.println("Current control activated");
616                  comando_unity = '0';
617              }
618              if(comando_unity == 'p'){
619                  controlmode = PositionControl;
```

```
620                 setpositioncontrol();
621                 Serial.println("Position control activated");
622                 setPID();
623                 comando_unity = '0';
624             }
625             if(comando_unity == 'n'){
626                 controlmode = NoControl;
627                 setcurrentcontrol();
628                 Serial.println("No control activated");
629                 comando_unity = '0';
630             }
631             if(comando_unity == 'a'){
632                 Fmax = 6;
633                 set_PID1_I[4] = 0;
634                 set_PID1_I[5] = 0;
635                 set_PID2_I[4] = 0;
636                 set_PID2_I[5] = 0;
637                 setPID();
638                 Serial.println("Low forces");
639                 comando_unity = '0';
640             }
641             if(comando_unity == 'b'){
642                 Fmax = 12;
643                 setPID();
644                 Serial.println("High forces");
645                 comando_unity = '0';
646             }
647             if(comando_unity == 'r'){
648                 sentido = 3.1416;
649                 Serial.println("Easy Level");
650                 comando_unity = '0';
651             }
652             if(comando_unity == 'h'){
653                 sentido = 0;
654                 Serial.println("Hard Level");
655                 comando_unity = '0';
656             }
657         //se o start for apertado
658         if(start == true)
659         {
660             comando_unity = '0';
661             State = OperationalControl;
662             delay(10);
663         }
664         break;
665 
666     case GoHome:
667         doHoming();
668         break;
669 
670     case OperationalControl:
671 
672             DataControl();
673             DataControl();
674             DataControl();
675             DataControl();
676 
677             if (sync_flag)
678             {
679                 sync_flag=0;
680                 sync();
681             }
682 
683             DataControl();
684             DataControl();
685             DataControl();
686             DataControl();
687 
688             if (sync_flag)
689             {
690                 sync_flag=0;
691                 sync();
692             }
693 
694 
695 
696         ang1 = (enc2 - offset2) * 0.0002244; // 2pi/28000
697         ang2 = 1.5708 + (-enc1 - offset1) * 0.0005464; // 2pi/11500
698 
```

```
699
700           x = cos(ang1) * 200 + cos(ang2) * 150 - 200;
701           y = sin(ang1) * 200 + sin(ang2) * 150 - 150;
702
703           o = atan2(y,x);
704           r = sqrt(x*x+y*y);
705
706           dif = (35 - r); //raio ideal
707
708           if(abs(dif) < 25)
709               z = 30-sqrt(25*25-dif*dif);
710           else
711               z = 0;
712
713           tor1 =  (0.0005810 * (float) cur2);// - cur2/abs(cur2)*0.1021;
714           tor2 = -(0.0002628 * (float) cur1);// + cur1/abs(cur1)*0.2166;
715
716           fx = +(0.33*cos(ang1)*tor2-0.25*cos(ang2)*tor1)/(sin(ang2-ang1)*0.05);
717           fy = +(0.33*sin(ang1)*tor2-0.25*sin(ang2)*tor1)/(sin(ang2-ang1)*0.05);
718           F = sqrt(fx*fx+fy*fy);
719
720           Serial.print(";");
721           Serial.print(x,2);
722           Serial.print(";");
723           Serial.print(y,2);
724           Serial.print(";");
725           Serial.print(z+10,2);
726           Serial.print(";");
727           Serial.print(F,2);
728           Serial.print(";");
729           Serial.print(r,2);
730           Serial.print(";");
731           Serial.print(o,2);
732           Serial.print(";");
733           Serial.print((int) cur1);
734           Serial.print(";");
735           Serial.print((int) cur2);
736           Serial.println(";");
737
738           if(controlmode == CurrentControl)
739           {
740               if (abs(dif) < 0.1 || abs(dif) > 30)
741                   F_demand = 0;
742                else
743                   F_demand = (dif / 25) * Fmax;
744
745                   tor1_demand = sin(ang1)*(0.20)* F_demand * cos(o+sentido) -
cos(ang1)*(0.20)*F_demand * sin(o+sentido); //sin(ang1)*(l1+l3)* F_demand * cos(o) -
cos(ang1)*(l1+l3)*F_demand * sin(o);
746                   tor2_demand = sin(ang2)*(0.15) * F_demand * cos(o+sentido) -
cos(ang2)*(0.15)*F_demand * sin(o+sentido); //sin(ang2)*(l2) * F_demand * cos(o) -
cos(ang2)*(l2)*F_demand * sin(o);
747
748                   cur1_demand = 3800.2 * tor2_demand;
749                   cur2_demand = 1710.0 * tor1_demand;
750
751                   //comandar correntes
752                   currentSetpoint1((int16_t)-cur1_demand);
753                   delay(4);
754                   currentSetpoint2((int16_t)cur2_demand);
755                   delay(4);
756           }
757
758           if(controlmode == PositionControl)
759           {
760               if(abs(dif) < 2){
761                   r_demand = 35;
762               }
763               else{
764                   r_demand = 35;
765               }
766
767               x_demand = (r_demand) * cos(o)+200;
768               y_demand = (r_demand) * sin(o)+150;
769               OP = sqrt((x_demand)*(x_demand)+(y_demand)*(y_demand))/1000;
770
771               ang2_demand = acos((OP*OP - 0.0175) / (0.3 * OP)) +
acos(x_demand/OP/1000);
772               ang1_demand = acos((0.0625 - OP*OP) / 0.06) - 3.1416 + ang2_demand ;
```

```
773
774             enc1_demand = -ang2_demand * 1830 + 2875 - offset1;
775             enc2_demand = ang1_demand * 4456 + offset2;
776
777             positionSetpoint1((int32_t) enc1_demand);
778             delay(1);
779             positionSetpoint2((int32_t) enc2_demand);
780             delay(1);
781         }
782
783         break;
784
785     case SetPreOperational:
786         CAN.sendMsgBuf(0x00, 0, 2, set_preoperational);
787         delay(10);
788         break;
789
790     case SetOperational:
791         CAN.sendMsgBuf(0x00, 0, 2, set_operational);
792         delay(10);
793         break;
794
795     }
796
797     if(Serial.available() > 0)
798         if(Serial.read() == 'f'){
799             State = Operational;
800             start = false;
801         }
802
803     if (sync_flag)
804     {
805         sync_flag=0;
806         sync();
807     }
808  }
809
810 /*********************************************************************************
811    END FILE
812*********************************************************************************/
```

XIV.III. Apêndice C – Código do Unity

GameController:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class GameController : MonoBehaviour {

    // INICIALIZAÇÃO DAS VARIÁVEIS USADAS //

    // Variáveis de interface
    public GUIText restartText;
    public GUIText gameOverText;
    public GUIText timerText;
    public GUIText youwinText;
    public Text fText;
    public Text scoretext;
    public Text game_start;

    // Variáveis booleanas de controle
    private bool gameOver;
    private bool restart;

    // Contador do placar
```

```
    private int cont;

    // Variáveis de aúdio
    public AudioClip SoundGameOver;
    public AudioClip SoundYouWin;
    private bool AudioOverPlayed = false;
    private bool AudioWinPlayed = false;
    AudioSource AudioOver;
    AudioSource AudioWin;

    // Informações que serão inseridas na base de dados
    private bool GO;
    private string store_time;
    private string Fmax;

    // Variáveis de tempo
    private float t0;
    private float t;

    // Inicialização
    void Start()
    {
        cont = 0;
        restartText.text = "";
        gameOverText.text = "";
        youwinText.text = "";
        gameOver = false;
        restart = false;
        GO = false;

        AudioOver = GetComponent<AudioSource>();
        AudioWin = GetComponent<AudioSource>();
    }

    // Update chamado uma vez por frame
    void Update()
    {

        //Realiza o restart do jogo
        if (restart)
        {
            if (Input.GetKeyDown(KeyCode.Space))
            {
                SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
            }
        }

        //Cuida da atualização do tempo
        if (game_start.text == "OK")
        {

            if (restart == true)
            {
                timerText.text = store_time;
            }
            else
            {
                t = Time.timeSinceLevelLoad - t0;
                if (((t * 100) % 100) < 99)
                {
                    timerText.text = ((int)((t - t % 60f) / 60f)).ToString("00")
+ ":" + ((int)t % 60f).ToString("00") + ":" + ((t * 100) % 100).ToString("00");
                }
            }
```

```
    }
    else
        t0 = Time.timeSinceLevelLoad;

}


//Rotina para o fim de jogo
public void GameOver(Collider other)
{
    if (restart == false)
    {
        gameOverText.text = "Não foi dessa vez...";
        if (!AudioOverPlayed)
        {
            AudioOver.PlayOneShot(SoundGameOver, 1);
            AudioOverPlayed = true;
        }

        gameOver = true;
        restartText.text = "Aperte espaço para jogar de novo";
        other.attachedRigidbody.useGravity = true;
        restart = true;

        GO = false;
        Store_Data(GO);
    }
}

//Rotina para a pontuação
public void score(int cont)
{
    scoretext.text = "Score:"+cont.ToString();
    if (cont >= 10)
    {
        if (!AudioWinPlayed)
        {
            AudioWin.PlayOneShot(SoundYouWin, 1);
            AudioWinPlayed = true;
        }
        youwinText.text = "Parabéns, você conseguiu!";
        restartText.text = "Aperte a tecla espaço para recomeçar";
        restart = true;

        GO = true;
        Store_Data(GO);
    }
}

public void Store_Data(bool GO)
{
    store_time = timerText.text;
    Fmax = fText.text;
    Send_Data(GO);

}

public void Send_Data(bool GO)
{
    WWWForm form = new WWWForm();
    form.AddField("time_post", store_time);
    form.AddField("fmax_post", Fmax);
    if (GO)
        form.AddField("gameover_post", "True");
```

```
        else
            form.AddField("gameover_post", "False");

        WWW url_path = new WWW("//localhost/TCC_DB/Insert_Data.php", form);
        StartCoroutine(WaitForRequest(url_path));
    }

    IEnumerator WaitForRequest(WWW www)
    {
        yield return www;

        // Veirificação de erro
        if (www.error == null)
        {
            Debug.Log("WWW Ok!: " + www.text);
        }
        else
        {
            Debug.Log("WWW Error: " + www.error);
        }
    }

}
```

PlayerController:

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class PlayerController : MonoBehaviour {

    // INICIALIZAÇÃO DAS VARIÁVEIS USADAS //

    public float speed; //Velocidade do player
    private int cont; //Conta as etrelas coletadas
    public float Fmax; //Guarda a força máxima
    private Collider other; //Componente collider do player

    //Variáveis de controle do jogo
    private bool ganhou;
    private bool perdeu;
    private GameController gameController;

    //Variáveis para rotação do player
    private Vector3 up = Vector3.zero;
    private Vector3 currentposition = Vector3.zero;
    private Vector3 down = new Vector3 (0,180,0);
    private Vector3 right = new Vector3(0, 90, 0);
    private Vector3 left = new Vector3(0, 270, 0);

    //Variáveis de deslocamento
    public Text xText;
    public Text yText;
    public Text zText;
    public Text fText;
    public Text cur1Text;
    public Text cur2Text;
    public Text rText;
    public Text oText;
```

```
    private int i = 0;
    private string x;
    private string y;
    private string z;
    private string f;
    private string cur1;
    private string cur2;
    private string r;
    private string o;

    private float coord_x;
    private float coord_y;
    private float coord_z;
    private float force;

    private float last_x;
    private float last_y;
    private float last_z;
    private float radius;

    //Variáveis de audio
    public AudioClip SoundToPlay;
    AudioSource AudioStar;

    //Variáveis de centralização do pivot
    static GameObject selectedObject;
    static Mesh selectedObjectMesh;
    static Vector3 selectedObjectPivot;

    // FUNÇÕES BÁSICAS DE CONTROLE DO JOGO //

    // Inicialização
    void Start () {

        QualitySettings.vSyncCount = 0;

        selectedObject = GameObject.FindWithTag("Player");
        RecognizeSelectedObject();
        CenterObjectPivot();

        other = GetComponent<Collider>();

        cont = 0;
        Fmax = 0;

        GameObject gameControllerObject =
GameObject.FindWithTag("GameController");
        if (gameControllerObject != null)
        {
            gameController = gameControllerObject.GetComponent<GameController>();
        }
        if (gameController == null)
        {
            Debug.Log("Cannot find 'GameController' script");
        }

        gameController.score(cont);

        last_x = transform.position.x;
        last_y = transform.position.z;
        last_z = transform.position.z;

        AudioStar = GetComponent<AudioSource>();
```

```
        ganhou = false;
        perdeu = false;
    }

    // Update chamado uma vez por frame
    void Update () {

        var Moving = true;

        if (Input.GetKey(KeyCode.UpArrow)) currentposition = up;
        else if (Input.GetKey(KeyCode.DownArrow)) currentposition = down;
        else if (Input.GetKey(KeyCode.RightArrow)) currentposition = right;
        else if (Input.GetKey(KeyCode.LeftArrow)) currentposition = left;
        else Moving = false;

        transform.localEulerAngles = currentposition;

        if (Moving) transform.Translate(Vector3.forward * speed *
Time.deltaTime);

        last_x = transform.position.x;
        last_z = transform.position.z;
        radius = Mathf.Sqrt(last_x * last_x + last_z * last_z);

        if (radius > 60 || radius < 10)
        {
            gameController.GameOver(other);
            perdeu = true;
            cont = -1000;
        }

        if (ganhou == true)
            transform.position = new Vector3(transform.position.x*(float) 0.99,
transform.position.y+(float) 0.09, transform.position.z -(float)0.05);

    }

    // RECEBE E ATUALIZA A POSIÇÃO DO PLAYER VIA ARDUINO //

    void OnSerialLine(string line)
    {
        Debug.Log (line);
        i = 0;
        if (line[i] == ';')
        {
            i = i + 1;
            while (line[i] != ';')
            {
                x += line[i];
                i += 1;
            }
            i += 1;
            while (line[i] != ';')
            {
                y += line[i];
                i += 1;
            }
            i += 1;
                    while (line[i] != ';')
            {
                z += line[i];
                i += 1;
            }
            i += 1;
```

```
        while (line[i] != ';')
                {
                        f += line[i];
                        i += 1;
                }
        i += 1;
        while (line[i] != ';')
        {
            r += line[i];
            i += 1;
        }
        i += 1;
        while (line[i] != ';')
        {
            o += line[i];
            i += 1;
        }
        i += 1;
        while (line[i] != ';')
        {
            cur1 += line[i];
            i += 1;
        }
        i += 1;
        while (i <= line.Length - 1)
        {
            cur2 += line[i];
            i += 1;
        }

        coord_x = float.Parse(x);
        coord_y = float.Parse(y);
        coord_z = float.Parse(z);
                force = float.Parse(f);

        if (Fmax < force) Fmax = force;

        if(ganhou == false && perdeu == false)
            transform.position = new Vector3(coord_x, coord_z, coord_y);

        xText.text = "X: " + x.ToString();
        yText.text = "Y: " + transform.position.y.ToString();
        zText.text = "Z: " + y.ToString();
        fText.text = "F: " + f.ToString();
        cur1Text.text = "cur1: " + cur1 + "mA";
        cur2Text.text = "cur2: " + cur2 + "mA";
        rText.text = "r: " + r;
        oText.text = "o: " + o + "º";

        x = string.Empty;
        y = string.Empty;
        z = string.Empty;
        f = string.Empty;
        r = string.Empty;
        o = string.Empty;
        cur1 = string.Empty;
        cur2 = string.Empty;

    }
}

// TRECHO QUE TRATA DO CONTATO DO PLAYER COM AS ESTRELAS //

void OnTriggerEnter(Collider other)
```

```
    {
        if (other.gameObject.CompareTag("Pick Up"))
        {
            other.gameObject.SetActive(false);
            cont = cont + 1;
            if (cont >= 10)
                ganhou = true;
            gameController.score(cont);
            AudioStar.PlayOneShot(SoundToPlay, 1);
        }

    }

    // TRECHO DE MODIFICAÇÃO DO CENTRO DO PLAYER //

    static void RecognizeSelectedObject()
    {
        selectedObjectMesh = null;

        Transform recognizedTransform = selectedObject.GetComponent<Transform>();
        if (recognizedTransform)
        {
            selectedObject = recognizedTransform.gameObject;
            if (selectedObject)
            {

                MeshFilter selectedObjectMeshFilter =
selectedObject.GetComponent<MeshFilter>();
                if (selectedObjectMeshFilter)
                {
                    selectedObjectMesh = selectedObjectMeshFilter.sharedMesh;
                    if (selectedObjectMesh)
                    {
                        selectedObjectPivot =
FindObjectPivot(selectedObjectMesh.bounds);
                    }
                }

            }
        }
    }

    static public Vector3 FindObjectPivot(Bounds bounds)
    {
        Vector3 offset = -1 * bounds.center;
        Vector3 extent = new Vector3(offset.x / bounds.extents.x, offset.y /
bounds.extents.y, offset.z / bounds.extents.z);
        return Vector3.Scale(bounds.extents, extent);
    }

    static void CenterObjectPivot()
    {
        // Move object position by taking localScale into account
        selectedObject.transform.position -= Vector3.Scale(selectedObjectPivot,
selectedObject.transform.localScale);

        // Iterate over all vertices and move them in the opposite direction of
the object position movement
        Vector3[] verts = selectedObjectMesh.vertices;
        for (int i = 0; i < verts.Length; i++)
        {
            verts[i] += selectedObjectPivot;
        }
```

```
        selectedObjectMesh.vertices = verts; //Assign the vertex array back to
the mesh
        selectedObjectMesh.RecalculateBounds(); //Recalculate bounds of the mesh,
for the renderer's sake

        FixColliders(selectedObjectPivot);
    }

    static void FixColliders(Vector3 scaleDiff)
    {
        Collider selectedObjectCollider =
selectedObject.GetComponent<Collider>();

        if (selectedObjectCollider)
        {
            if (selectedObjectCollider is BoxCollider)
            {
                ((BoxCollider)selectedObjectCollider).center += scaleDiff;
            }
            else if (selectedObjectCollider is CapsuleCollider)
            {
                ((CapsuleCollider)selectedObjectCollider).center += scaleDiff;
            }
            else if (selectedObjectCollider is SphereCollider)
            {
                ((SphereCollider)selectedObjectCollider).center += scaleDiff;
            }
            // missing calculation to compensate for MeshCollider
        }

        selectedObjectPivot = Vector3.zero;
    }

}
```

Rotation:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class Rotation : MonoBehaviour {

    // INICIALIZAÇÃO DAS VARIÁVEIS USADAS //

    //Variáveis de centralização do pivot
    static GameObject selectedObject;
    static Mesh selectedObjectMesh;
    static Vector3 selectedObjectPivot;

    //Variáveis para posicionar aleatoriamente as estrelas
    private float x;
    private float y;
    private float z;
    private float r;
    private float dif;
    private float ang;

    //Variável de identificação da cena
    private int scene_id;

    // FUNÇÕES BÁSICAS DE CONTROLE DO JOGO //
```

```
    // Inicialização
    void Start()
    {
        scene_id = SceneManager.GetActiveScene().buildIndex;

        selectedObject = GameObject.FindWithTag("Pick Up");
        RecognizeSelectedObject();
        CenterObjectPivot();

        r = Random.value * 45 + 15;
        ang = Random.value * 618 / 100;
        x = Mathf.Cos(ang) * r;
        z = Mathf.Sin(ang) * r;
        dif = (35 - r); //raio ideal

        if (scene_id == 1)
            y = 37 - Mathf.Sqrt(25 * 25 - dif * dif);
        else if (scene_id == 2)
            y = 12 + Mathf.Sqrt(25 * 25 - dif * dif);

        transform.position = new Vector3(x, y, z); //coord_z
    }

    // Update chamado uma vez por frame
    void Update()
    {
        transform.Rotate(new Vector3(0, 30, 0) * Time.deltaTime);
    }

    // TRECHO DE MODIFICAÇÃO DO CENTRO DAS ESTRELAS //

    static void RecognizeSelectedObject()
    {
        selectedObjectMesh = null;

        Transform recognizedTransform = selectedObject.GetComponent<Transform>();
        if (recognizedTransform)
        {
            selectedObject = recognizedTransform.gameObject;
            if (selectedObject)
            {

                MeshFilter selectedObjectMeshFilter =
selectedObject.GetComponent<MeshFilter>();
                if (selectedObjectMeshFilter)
                {
                    selectedObjectMesh = selectedObjectMeshFilter.sharedMesh;
                    if (selectedObjectMesh)
                    {
                        selectedObjectPivot =
FindObjectPivot(selectedObjectMesh.bounds);
                    }
                }

            }
        }
    }

    static public Vector3 FindObjectPivot(Bounds bounds)
    {
        Vector3 offset = -1 * bounds.center;
        Vector3 extent = new Vector3(offset.x / bounds.extents.x, offset.y /
bounds.extents.y, offset.z / bounds.extents.z);
```

```
        return Vector3.Scale(bounds.extents, extent);
    }

    static void CenterObjectPivot()
    {
        // Move object position by taking localScale into account
        selectedObject.transform.position -= Vector3.Scale(selectedObjectPivot,
selectedObject.transform.localScale);

        // Iterate over all vertices and move them in the opposite direction of
the object position movement
        Vector3[] verts = selectedObjectMesh.vertices;
        for (int i = 0; i < verts.Length; i++)
        {
            verts[i] += selectedObjectPivot;
        }
        selectedObjectMesh.vertices = verts; //Assign the vertex array back to
the mesh
        selectedObjectMesh.RecalculateBounds(); //Recalculate bounds of the mesh,
for the renderer's sake

        FixColliders(selectedObjectPivot);
    }

    static void FixColliders(Vector3 scaleDiff)
    {
        Collider selectedObjectCollider =
selectedObject.GetComponent<Collider>();

        if (selectedObjectCollider)
        {
            if (selectedObjectCollider is BoxCollider)
            {
                ((BoxCollider)selectedObjectCollider).center += scaleDiff;
            }
            else if (selectedObjectCollider is CapsuleCollider)
            {
                ((CapsuleCollider)selectedObjectCollider).center += scaleDiff;
            }
            else if (selectedObjectCollider is SphereCollider)
            {
                ((SphereCollider)selectedObjectCollider).center += scaleDiff;
            }
            // missing calculation to compensate for MeshCollider
        }

        selectedObjectPivot = Vector3.zero;
    }

}
```

OnOff:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class OnOff : MonoBehaviour {

    public Text game_start;

    // Envia sinal de start para o arduino
```

```
    public void TaskOnClick () {
        Serial.WriteLn("s");
        StartCoroutine(Delay());
        Serial.WriteLn("s");
        StartCoroutine(Delay());
        Serial.WriteLn("s");
        game_start.text = "ON";
    }

    // Envia sinal de stop/finish para o arduino
    public void TaskOffClick()
    {
        Serial.WriteLn("f");
        StartCoroutine(Delay());
        Serial.WriteLn("f");
        StartCoroutine(Delay());
        Serial.WriteLn("f");
        game_start.text = "OFF";
    }

    //Atraso entre envio de informação
    IEnumerator Delay()
    {
        yield return new WaitForSeconds((float)(0.5));
    }

}
```

PlaySound:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlaySound : MonoBehaviour {

    // Variáveis de aúdio
    public AudioClip SoundToPlay;
    public float volume;
    AudioSource audio;
    public bool alreadyplayed = false;

       // Inicialização
       void Start () {
        audio = GetComponent<AudioSource>();
       }

    // Update chamado uma vez por frame
    void Update () {
        if (!alreadyplayed)
        {
            audio.PlayOneShot(SoundToPlay, volume);
            alreadyplayed = true;
        }

       }
}
```

Login:

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Text.RegularExpressions;
using UnityEngine.SceneManagement;

public class Login : MonoBehaviour {

    // Variáveis de acesso ao Login
    public GameObject username;
    public GameObject password;
    private string Username;
    private string Password;

    // Inicialização
    void Start () {

       }

    // Update chamado uma vez por frame
    void Update () {

        if(Input.GetKeyDown(KeyCode.Tab))
        {
            if (username.GetComponent<InputField>().isFocused)
                password.GetComponent<InputField>().Select();
        }

        Username = username.GetComponent<InputField>().text;
        Password = password.GetComponent<InputField>().text;

    }

    // ROTINAS DE CONEXÃO COM O BANCO DE DADOS //

    public void Search_User_Login()
    {
        WWWForm form = new WWWForm();
        form.AddField("username_post", Username);
        WWW url_search = new WWW("//localhost/TCC_DB/Search_login.php",form);
        StartCoroutine(WaitForRequest(url_search));
    }

    IEnumerator WaitForRequest(WWW www)
    {
        yield return www;

        // Veirificação de erro
        if (www.error == null)
        {
            string found = www.text;

            if (found == "")
            {
                Debug.LogWarning("Usuário não encontrado");
            }
            else
            {
                if (found != Password)
                {
                    Debug.LogWarning("Senha incorreta");
                }
                else
                {
```

```
                    print("Usuário logado!");
                    SceneManager.LoadScene(1);
                }
            }
        }
        else
        {
            Debug.Log("WWW Error: " + www.error);
        }
    }
}
```

Registration:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Text.RegularExpressions;

public class Registration : MonoBehaviour {

    // Variáveis usadas para registro do Login
    public GameObject username;
    public GameObject password;
    public GameObject email;
    public GameObject confpassword;
    private string Username;
    private string Password;
    private string Email;
    private string ConfPassword;

    // Inicialização
    void Start () {

       }

       // Update chamado uma vez por frame
       void Update () {

        if (Input.GetKeyDown(KeyCode.Tab))
        {
            if (username.GetComponent<InputField>().isFocused)
                email.GetComponent<InputField>().Select();

            if (email.GetComponent<InputField>().isFocused)
                password.GetComponent<InputField>().Select();

            if (password.GetComponent<InputField>().isFocused)
                confpassword.GetComponent<InputField>().Select();
        }

        Username = username.GetComponent<InputField>().text;
        Password = password.GetComponent<InputField>().text;
        Email = email.GetComponent<InputField>().text;
        ConfPassword = confpassword.GetComponent<InputField>().text;

    }

    // Envia informação para base de dados
    public void Send_Info()
    {
        WWWForm form = new WWWForm();
```

```
        form.AddField("username_post", Username);
        form.AddField("password_post", Password);
        form.AddField("email_post", Email);

        WWW url_path = new WWW("//localhost/TCC_DB/Login.php", form);
        StartCoroutine(WaitForRequest(url_path));
    }

    // Função auxiliadora de envio
    IEnumerator WaitForRequest(WWW www)
    {
        yield return www;

         // Veirificação de erro
        if (www.error == null)
        {
            Debug.Log("WWW Ok!: " + www.text);
        }
        else
        {
            Debug.Log("WWW Error: " + www.error);
        }
    }

    //Verifica informação de cadastro
    public void Verification()
    {
        bool UN = true;
        bool PW = false;
        bool EM = true;

        if (Username != "" && Password != "" && Email != "" && ConfPassword !=
"")
        {

            if (ConfPassword == Password)
            {
                PW = true;
            }
            else
            {
                print("A senha está incorreta");
                email.GetComponent<InputField>().text = "";
                confpassword.GetComponent<InputField>().text = "";
            }

            if (UN == true && PW == true && EM == true)
            {
                Send_Info();
                print("Cadastro confirmado");
                username.GetComponent<InputField>().text = "";
                password.GetComponent<InputField>().text = "";
                email.GetComponent<InputField>().text = "";
                confpassword.GetComponent<InputField>().text = "";
            }
        }
        else
        {
            Debug.LogWarning("As informações não estão completas");
        }
    }
}
```

Main_Menu:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Main_Menu : MonoBehaviour {

    // INICIALIZAÇÃO DAS VARIÁVEIS USADAS //

    // Variáveis de interação
    public GameObject try_out;
    public GameObject patients;
    public GameObject logout;
    public Animator animator = null;

    //Variáveis de conexão com a base de dados
    public GameObject pacient_search;
    public GameObject pacient_insert;
    private string pacient_s;
    private string pacient_i;
    private bool exist;

    // ROTINAS DE INTERAÇÃO DO USUÁRIO //

    public void TryOut()
    {
        try_out.gameObject.SetActive(true);
        patients.gameObject.SetActive(false);
        logout.gameObject.SetActive(false);
    }

    public void Patients()
    {
        patients.gameObject.SetActive(true);
        try_out.gameObject.SetActive(false);
        logout.gameObject.SetActive(false);
    }

    public void LogOut()
    {
        logout.gameObject.SetActive(true);
        patients.gameObject.SetActive(false);
        try_out.gameObject.SetActive(false);
    }

    public void Go_Game()
    {
        animator.SetBool("Go", true);
    }

    public void Exit()
    {
        SceneManager.LoadScene(0);
    }

    public void Return()
    {
        Patients();
    }

    // ROTINAS DE CONEXÃO COM O BANCO DE DADOS //
```

```
    public void Search_pacient()
    {
        pacient_s = pacient_search.GetComponent<InputField>().text;
        WWWForm form = new WWWForm();
        form.AddField("username_post", pacient_s);
        WWW url_search = new WWW("//localhost/TCC_DB/Search_pacient.php", form);
        StartCoroutine(Funcao_Busca(url_search));
    }

    public void Insert_pacient()
    {
        exist = false;
        pacient_i = pacient_insert.GetComponent<InputField>().text;
        WWWForm form = new WWWForm();
        form.AddField("username_post", pacient_i);
        WWW url_aux = new WWW("//localhost/TCC_DB/Search_pacient.php", form);
        StartCoroutine(Aux_Criar(url_aux,form));
    }

    IEnumerator WaitForRequest1(WWW www)
    {
        yield return www;

        // Veirificação de erro
        if (www.error == null)
        {
            Debug.Log("WWW Ok!: " + www.text);
        }
        else
        {
            Debug.Log("WWW Error: " + www.error);
        }
    }

    IEnumerator Aux_Criar(WWW www, WWWForm form)
    {
        yield return www;

        // Veirificação de erro
        if (www.error == null)
        {
            string found = www.text;


            if (found == pacient_i)
            {
                Debug.LogWarning("Usuário já existe!");
                exist = true;
            }
            else
            {
                WWW url_path = new WWW("//localhost/TCC_DB/Create_pacient.php",
form);
                StartCoroutine(WaitForRequest1(url_path));
            }
        }
        else
        {
            Debug.Log("WWW Error: " + www.error);
        }
    }

    IEnumerator Funcao_Busca(WWW www)
    {
```

```
        yield return www;

        // Veirificação de erro
        if (www.error == null)
        {
            string found = www.text;

            if (found != pacient_s)
            {
                Debug.LogWarning("Usuário não encontrado");
            }
            else
            {
                Debug.Log("Usuário encontrado!");
                animator.SetBool("Go", true);
            }
        }
        else
        {
            Debug.Log("WWW Error: " + www.error);
        }
    }
}
```

Game_Menu:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class Game_Menu : MonoBehaviour {

    // INICIALIZAÇÃO DAS VARIÁVEIS USADAS //

    // Variáveis de interação
    public GameObject control;
    public GameObject level;
    public GameObject intensity;
    public GameObject resume;
    public Animator animator = null;

    // Variáveis de envio para arduino
    private string control_aux;
    private string level_aux;
    private string intensity_aux;

    // Variáveis de interação
    public Text text_control;
    public Text text_level;
    public Text text_intensity;

    // Guarda o level da scene desejada
    private int scene_level = 0;

    // ROTINAS DE INTERAÇÃO DO USUÁRIO //

    public void Control()
    {
        control.gameObject.SetActive(true);
        level.gameObject.SetActive(false);
        intensity.gameObject.SetActive(false);
```

```
    resume.gameObject.SetActive(false);
}

public void Level()
{
    control.gameObject.SetActive(false);
    level.gameObject.SetActive(true);
    intensity.gameObject.SetActive(false);
    resume.gameObject.SetActive(false);
}

public void Intensity()
{
    control.gameObject.SetActive(false);
    level.gameObject.SetActive(false);
    intensity.gameObject.SetActive(true);
    resume.gameObject.SetActive(false);
}

public void Resume()
{
    control.gameObject.SetActive(false);
    level.gameObject.SetActive(false);
    intensity.gameObject.SetActive(false);
    resume.gameObject.SetActive(true);
}

public void Control_current()
{
    control_aux = "c";
    Send_Parameters(control_aux);
    text_control.text = "Controle de corrente";
}

public void Control_position()
{
    control_aux = "p";
    Send_Parameters(control_aux);
    text_control.text = "Controle de posição";
}

public void Level_regular()
{
    level_aux = "r";
    Send_Parameters(level_aux);
    text_level.text = "Nível de atividade regular";
    scene_level = 1;
}

public void Level_hard()
{
    level_aux = "h";
    Send_Parameters(level_aux);
    text_level.text = "Nível de atividade difícil";
    scene_level = 2;
}

public void Intensity_strong()
{
    intensity_aux = "b";
    Send_Parameters(intensity_aux);
    text_intensity.text = "Intensidade de feedback forte";
}
```

```
    public void Intensity_weak()
    {
        intensity_aux = "a";
        Send_Parameters(intensity_aux);
        text_intensity.text = "Intensidade de feedback fraca";
    }

    public void Start_game()
    {

        if (scene_level == 0)
            Debug.LogError("Escolha um tipo de exercício!");
        else
            SceneManager.LoadScene(scene_level);

    }

    // ROTINAS DE COMUNICAÇÃO COM O ARDUINO //

    public void Send_Parameters(string send)
    {
        Serial.WriteLn(send);
        StartCoroutine(Delay());
        Serial.WriteLn(send);
        StartCoroutine(Delay());
        Serial.WriteLn(send);
    }

    IEnumerator Delay()
    {
        yield return new WaitForSeconds((float)(0.5));
    }


    void OnSerialLine(string line)
    {
        Debug.Log(line);
    }

    public void Back_Menu()
    {
        animator.SetBool("Go", false);
    }
}
```

XIV.IV. Apêndice D – Código do banco de dados (PHP)

Login.php (insere/cria usuário no login):

```
<?php

    //Connection variables
    $servername = "localhost";
    $server_username = "root";
    $server_password = "";
    $database = "TCC_DB";

    //Unity variables
    $unity_username = $_POST["username_post"];
    $unity_password = $_POST["password_post"];
```

```
	$unity_email = $_POST["email_post"];

	//Database Conection
	$connect = new mysqli($servername, $server_username, $server_password, $database);
	if(!$connect)
	{
		die("conection failed". mysqli_connect_error());
	}

	$insert = "INSERT INTO login (Name, Password, Email)
			VALUES ('".$unity_username."','".$unity_password."','".$unity_email."')";
	$send = mysqli_query($connect , $insert);

	if(!send) echo ("Error");
	else echo ("OK");

?>
```

Search_login.php (procura usuário no banco de dados):

```
<?php

	//Connection variables
	$servername = "localhost";
	$server_username = "root";
	$server_password = "";
	$database = "TCC_DB";

	//Unity variables
	$unity_username = $_POST["username_post"];

	//Database Conection
	$connect = new mysqli($servername, $server_username, $server_password, $database);
	if(!$connect)
	{
		die("conection failed". mysqli_connect_error());
	}

	$select = "SELECT Password FROM login
			WHERE Name='".$unity_username."'";
	$send = mysqli_query($connect , $select);

	$result = mysqli_fetch_assoc($send);

	echo $result['Password'];

?>
```

Create_pacient.php (insere paciente na base de dados):

```
<?php
```

```
	//Connection variables
	$servername = "localhost";
	$server_username = "root";
	$server_password = "";
	$database = "TCC_DB";

	//Unity variables
	$unity_username = $_POST["username_post"];

	//Database Conection
	$connect = new mysqli($servername, $server_username, $server_password, $database);
	if(!$connect)
	{
		die("conection failed". mysqli_connect_error());
	}

	$insert = "INSERT INTO pacient (Name)
			VALUES ('".$unity_username."')";
		$send = mysqli_query($connect , $insert);
		echo ("OK")
?>
```

Search_pacient.php (busca paciente no banco de dados):

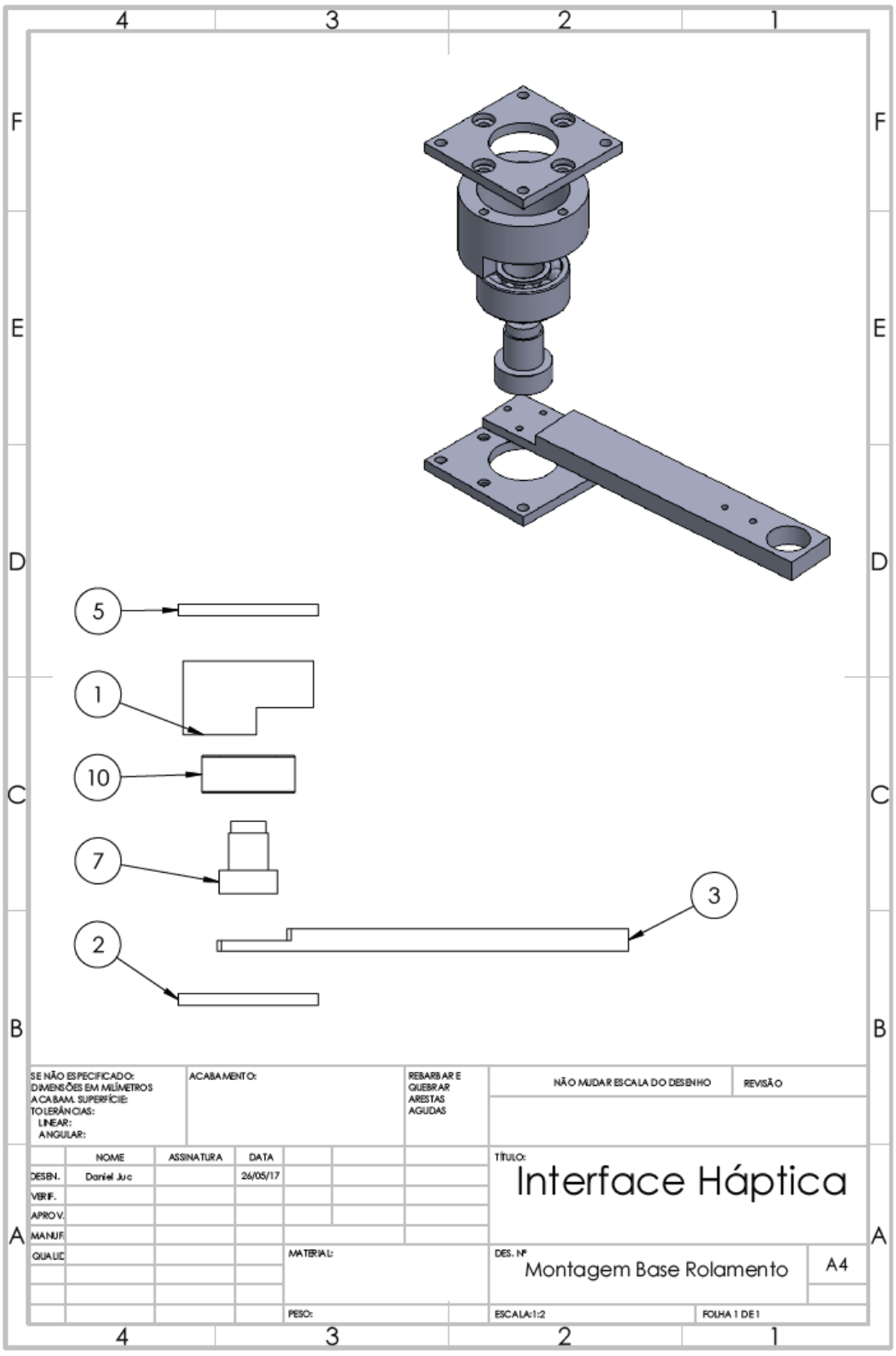
```
<?php

	//Connection variables
	$servername = "localhost";
	$server_username = "root";
	$server_password = "";
	$database = "TCC_DB";

	//Unity variables
	$unity_username = $_POST["username_post"];

	//Database Conection
	$connect = new mysqli($servername, $server_username, $server_password, $database);
	if(!$connect)
	{
		die("conection failed". mysqli_connect_error());
	}

	$select = "SELECT Name FROM pacient
				WHERE Name='".$unity_username."'";
	$send = mysqli_query($connect , $select);

	$result = mysqli_fetch_assoc($send);

	echo $result['Name'];
```

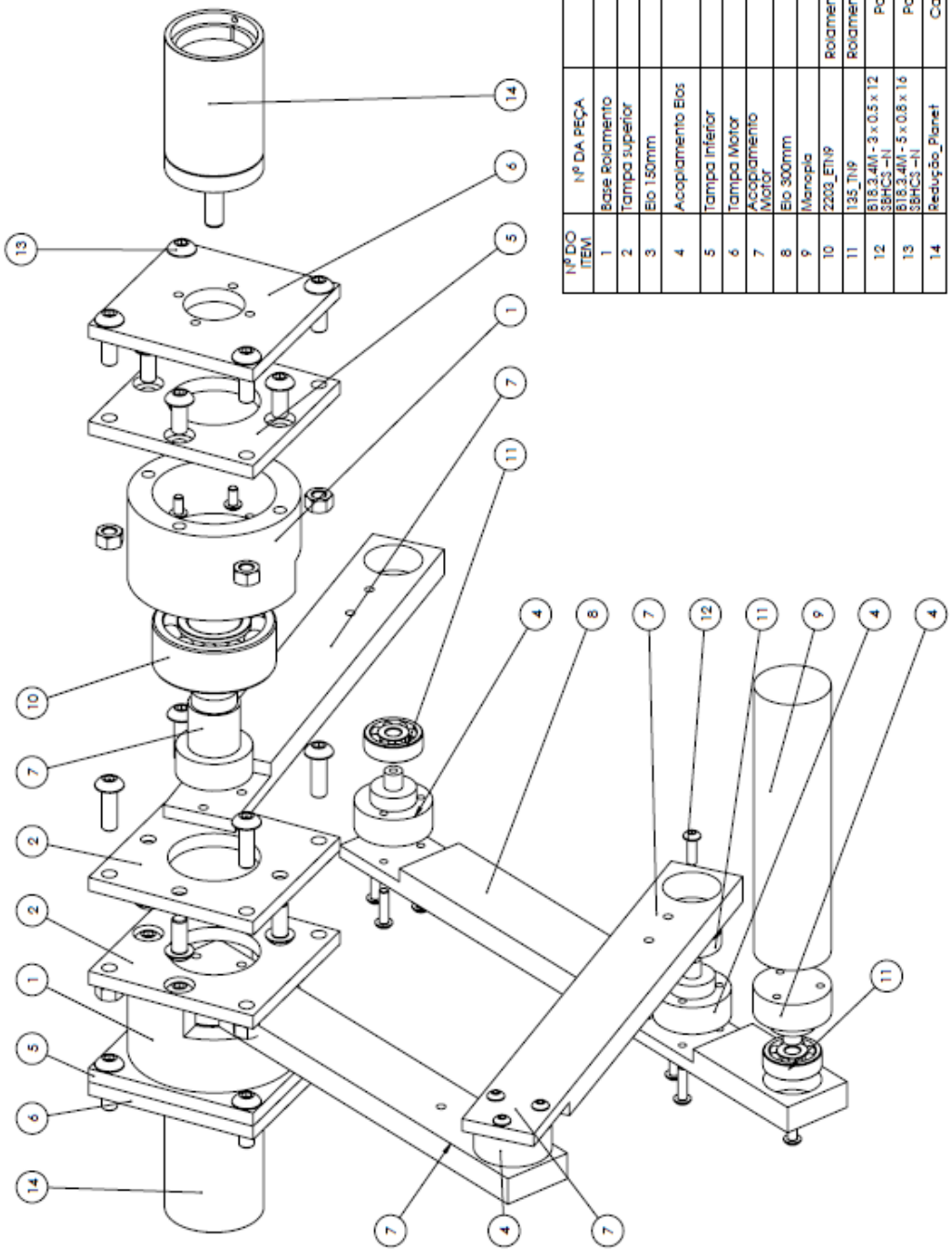
```
?>
```

Insert_Data.php (insere informações do exercício no banco de dados):

```
<?php

	//Connection variables
	$servername = "localhost";
	$server_username = "root";
	$server_password = "";
	$database = "TCC_DB";

	//Unity variables
	$time_post = $_POST["time_post"];
	$fmax_post = $_POST["fmax_post"];
	$gameover_post = $_POST["gameover_post"];

	//Database Conection
	$connect = new mysqli($servername, $server_username, $server_password, $database);
	if(!$connect)
	{
		die("conection failed". mysqli_connect_error());
	}

	$insert = "INSERT INTO exercicio (Time, Fmax, GO)
				VALUES ('".$time_post."','".$fmax_post."','".$gameover_post."')";
	$send = mysqli_query($connect , $insert);

	if(!send) echo ("Error");
	else echo ("OK");

?>
```

XIV.V. Apêndice E – Desenhos de Conjuntos





Nº DO ITEM	Nº DA PEÇA	DESCRIÇÃO	QTD.
1	Base Rolamento		2
2	Tampa superior		2
3	Elo 150mm		3
4	Acoplamento Elos		4
5	Tampa Inferior		2
6	Tampa Motor		2
7	Acoplamento Motor		2
8	Elo 300mm		1
9	Manopla		1
10	2203_ETN9	Rolamento Autocompensador	2
11	135_TN9	Rolamento Autocompensador	4
12	B18.3.4M - 3 x 0.5 x 12 SBHCS –N	Parafuso Allen M3	27
13	B18.3.4M - 5 x 0.8 x 16 SBHCS –N	Parafuso Allen M5	33
14	Redução_Planet	Caixa de Redução	2

Interface Háptica

vfMontagem 01

A2

Se não indicado, Alumínio