INTERFACE DE CONTROLE PORTÁTIL PARA CICLISMO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL

GUILHERME LUSTOSA RICARTE

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA DE SISTEMA ELETRÔNICOS E DE AUTOMAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PORTABLE CONTROL INTERFACE FOR CYCLING BY FUNCTIONAL ELECTRICAL STIMULATION

INTERFACE DE CONTROLE PORTÁTIL PARA CICLISMO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL

GUILHERME LUSTOSA RICARTE

ORIENTADOR: PROF. ANTONIO PADILHA LANARI BÓ

DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA DE SISTEMA ELETRÔNICOS E
DE AUTOMAÇÃO

PUBLICAÇÃO: PPGEA.TD-762/21

BRASÍLIA/DF: JANEIRO - 2021

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

INTERFACE DE CONTROLE PORTÁTIL PARA CICLISMO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL

GUILHERME LUSTOSA RICARTE

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE ENGENHEIRO EM SISTEMAS ELETRÔNICOS E DE AUTOMAÇÃO.

APROVADA POR:

_____

Prof. Dr. Antonio Padilha Lanari Bó – ENE/Universidade de Brasília
Orientador

_____

Prof. Dr. Marcelo M. Carvalho – ENE/Universidade de Brasília
Membro Interno

_____

Dra. Ana Carolina C. de Sousa, – School of Allied Health Sciences/Griffith University
Membro Externo

BRASÍLIA, 26 DE JANEIRO DE 2021.

FICHA CATALOGRÁFICA

RICARTE, GUILHERME LUSTOSA
INTERFACE DE CONTROLE PORTÁTIL PARA CICLISMO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL
[Distrito Federal] 2021.
xiii, 79p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia de Sistema Eletrônicos e de Automação, 2021).
dissertação de mestrado – Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Estimulação Elétrica Funcional	2. Portabilidade
3. ROS	4. Lesão Medular
I. ENE/FT/UnB	II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

RICARTE, GUILHERME LUSTOSA (2021). INTERFACE DE CONTROLE PORTÁTIL PARA CICLISMO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL, dissertação de mestrado, Publicação PPGEA.TD-762/21, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 79p.

Guilherme Lustosa Ricarte

Departamento de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

DEDICATÓRIA: *Ao meu filho que me faz tentar ser melhor a cada dia e a minha mãe e pai que me deram força e base para estar aqui.*

AGRADECIMENTOS

O Projeto EMA me surpreendeu em muitos aspectos. Não apenas pela sua qualidade técnica e acadêmica, mas pelo fator humano. Todos os integrantes do grupo sempre foram muito solícitos e amigos. O professor Antonio Padilha foi muito compreensivo, amigo e mesmo com grandes mudanças na sua vida não deixou de me guiar por esse processo, que eu achei bem árduo. Sem ele me motivando, eu não teria conseguido. Tenho muita gratidão por ter escolhido e ter sido escolhido por esse orientador e ser humano admirável.

Não poderia deixar de falar do Lucas Macedo, uma amizade que foi desenvolvida nesse período e que levarei para vida. Agradeço por todas as vezes que tive dúvidas, solicitei auxilio e ele sempre estava disposto e pronto a me ajudar.

Agradeço a Laís Marques que com seu jeito descontraído, riso solto e coração enorme, sempre sabia um jeitinho de me entender e me incentivar a prosseguir. E pela sua boa vontade de ficar acordada até de madrugada para me ajudar.

Sem esse campeão Estevão Lopes, nada disso seria possível, muito obrigado por ter me ajudado com todos os testes e sempre estar disponível quando eu precisei da sua ajuda. Ele é um campeão nato e um atleta incrível.

Meu agradecimento a todos os professores da UNB, que mesmo vivendo um momento tão delicado, mantiveram sua árdua missão de buscar melhorar o mundo com a ciência.

RESUMO

Título: INTERFACE DE CONTROLE PORTÁTIL PARA CICLISMO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL
Autor: Guilherme Lustosa Ricarte
Orientador: Prof. Antonio Padilha Lanari Bó

A estimulação elétrica funcional (EEF) tem sido adotada para provocar a contração muscular e promover a execução de exercícios por indivíduos com lesão medular (LM). Vários benefícios são observados na aplicação dessa técnica, como melhora na função cardiovascular, aumento da massa muscular, redução da perda de massa óssea, melhora no metabolismo. No entanto, devido à complexidade desses sistemas pode ser bastante complicado para os terapeutas e pessoas com LM utilizarem a EEF. Outra grande limitação para esses sistemas é a falta de modularidade e flexibilidade na troca ou adição de recursos, como sensores e estimuladores. Este trabalho apresenta um sistema modular e flexível para ciclismo por estimulação elétrica funcional que possui uma interface gráfica de controle portátil. Essa solução foi embarcada em um computador de placa única de tamanho reduzido, Raspberry Pi 4 e utilizou o *Robot Operating System* (ROS) para comunicação dos dispositivos. A biblioteca roslibjs, por meio de um servidor websocket, possibilitou a comunicação da interface gráfica com o ROS. A implementação da interface gráfica utilizou as linguagens JavaScript, CSS e HTML. Os resultados mostraram que a utilização da interface gráfica não alterou de forma significativa a demanda por recursos da CPU da Raspberry Pi 4, mantendo a média de 80% da utilização dos seus recursos após iniciado o sistema. Também apresentou que o número de clientes simultâneos para o servidor websocket não é um gargalo para essa aplicação, pois o teste de carga simulando 1000 clientes não apresentou erro na transmissão de dados. Adicionalmente foi apresentada a validação preliminar do sistema por meio de um experimento piloto com um participante com LM que obteve todo o controle através de seu celular, proporcionando a portabilidade do sistema.

ABSTRACT

Title: PORTABLE CONTROL INTERFACE FOR CYCLING BY FUNCTIONAL ELECTRICAL STIMULATION
Author: Guilherme Lustosa Ricarte
Supervisor: Prof. Antonio Padilha Lanari Bó

Functional electrical stimulation (FES) has been adopted to elicit muscle contraction and promote the performance of exercises by individuals with spinal cord injury (SCI). Several benefits are observed in the application of this technique such as improvement in cardiovascular function, increase in muscular mass, reduction of bone mass loss, improvement in metabolism. However, due to the complexity of these systems it can be quite complicated for therapists and people with SCI to use FES. Another major limitation for these systems is the lack of modularity and flexibility in the exchange or addition of resources, such as sensors and stimulators. This work presents a modular and flexible system for cycling by functional electrical stimulation that has a portable control graphic interface. This solution was embedded on a small, single-board computer, Raspberry Pi 4, and used the Robot Operating System (ROS) to communicate the devices. The roslibjs library, through a websocket server, provides the communication with the graphic interface and the ROS. The implemetation of the graphical interface used JavaScript, CSS and HTML. The results showed that the use of the graphical interface did not significantly change the demand for CPU resources of the Raspberry Pi 4 maintaining an average of 80 % of the use of its resources after starting the system. It also showed that the number of simultaneous clients for the websocket server is not a bottleneck for this application since the load test simulating 1000 clients did not present an error in data transmission. Additionally, the preliminary validation of the system was presented through a pilot experiment with a participant with SCI who obtained all the control through his cell phone providing system portability.

Lista de Figuras

Lista de Códigos Fonte

Lista de Tabelas

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A medula espinhal é responsável, entre outras tarefas, por conectar os impulsos elétricos do cérebro aos músculos. Uma lesão na medula espinhal (LM) pode interromper essa comunicação e ser responsável por alterações nas funções motoras ocasionando a paralisia ou paresia dos membros. No Brasil a incidência é de 40 casos novos/ano/milhão de habitantes, ou seja, cerca de 6 a 8 mil casos novos por ano, sendo que destes 80% das vítimas são homens e 60% se encontram entre os 10 e 30 anos de idade [4].

Indivíduos com LM geralmente apresentam alteração na sensibilidade dos reflexos superficiais e profundos [5], piora do funcionamento dos sistemas cardiorrespiratório, gastrintestinal, geniturinário, alteração da sudorese, perda do controle de temperatura corpórea, atrofia muscular, fragilização óssea o que resultam em baixa independência e qualidade de vida [6].

A aplicação de técnicas de reabilitação são precedidas por uma avaliação de um fisioterapeuta que define um protocolo de tratamento a partir da necessidade do indivíduo. Alguns métodos de reabilitação utilizam impulsos elétricos aplicados aos músculos por meio de eletrodos em uma intensidade e largura de pulso suficientes para evocar a excitação muscular e auxiliam pacientes com LM no processo de readaptação e reinserção nas atividades diárias [7]. Essa técnica de contração muscular através de impulsos elétricos é conhecida como estimulação elétrica funcional (EEF) e as modalidades convencionais de exercícios utilizando a EEF estão associadas ao remo, ciclismo, caminhada, natação e outros derivados.

Além da aplicação da EEF em reabilitações, essa técnica também tem sido aplicada em competições. Em 2013, foi introduzido um campeonato chamado Cybathlon, que tem o objetivo de promover ainda mais o desenvolvimento de um sistema de assistência adequado para uso diário entre indivíduos com deficiência física. No Cybathlon as pessoas com deficiência física competem entre si nas tarefas do dia a dia, com o auxílio de dispositivos auxiliares avançados, incluindo o ciclismo assistido por EEF [8].

Vários estudos apresentam os benefícios do ciclismo assistido por EEF para a saúde de indivíduos com LM como a melhora na função cardiovascular, aumento da massa muscular, redução da perda de massa óssea, melhora no metabolismo [9, 10, 11]. No entanto, uma grande dificuldade é encontrada na utilização de sistemas de reabilitação associados a EEF. Essa dificuldade decorre da complexidade desses sistemas que torna mais difícil a interação

do profissional da saúde e do paciente no processo de reabilitação. Essa diminuição da interação pode interferir de forma negativa no desempenho dos resultados adquiridos da reabilitação [12]. Outro aspecto relacionado a essa interação é a necessidade do constante monitoramento de todo o processo, pelo profissional da saúde, a fim de manter a segurança do paciente.

As soluções apresentadas na literatura para o ciclismo assistido por EEF tem diferentes contextos de utilização. Esses contextos incluem competição, estudos clínicos, mobilidade do ciclista com deficiência motora e aplicações comerciais. Apesar de apresentarem diferentes objetivos essas soluções têm dificuldades comuns como falta de modularidade (os sensores e estimuladores não são programados como módulos), flexibilidade (não permite a adição de novos módulos sem grande esforço computacional), portabilidade (não é acessível por dispositivos móveis) e reusabilidade dos recursos gráficos (os recursos gráficos não podem ser reaproveitados em diferentes contextos de aplicação de EEF) [13, 14, 15, 16].

Essas características dificultam a adição ou troca de recursos no sistema (estimuladores, sensores) e diminuem a intercambiabilidade de soluções e as comparações de resultados entre grupos de pesquisa. Este trabalho buscou implementar uma solução utilizando o *middleware Robot Operating System* (ROS) que permite a comunicação e modularização dos equipamentos, também utilizou as linguagens JavaScript, HTML, CSS para a concepção da interface gráfica e a tecnologia *websocket* para proporcionar a portabilidade do sistema.

1.2 DEFINIÇÃO DO PROBLEMA

Considerando a complexidade dos sistemas de ciclismo assistidos por EEF, a necessidade de interação do profissional de saúde e do paciente com a reabilitação e a importância do constante monitoramento do processo. Assim como, os benefícios na troca de soluções entre grupos de pesquisa, e a capacidade de comparação de resultados. Surgem as questões centrais deste trabalho:

- Quais requisitos uma interface necessita para que o terapeuta e o paciente possam manter o monitoramento e o controle das atividades no processo de reabilitação?
- Como propiciar que essa solução seja modular, portátil e reutilizável para ciclismo por estimulação elétrica funcional?

Essas questões são abordadas no contexto do projeto Empoderando Mobilidade e Autonomia (EMA), que visa desenvolver tecnologias assistivas voltadas para pessoas com di-

ficuldade de locomoção. Utiliza-se um triciclo assistido por estimulação elétrica funcional para pesquisa e desenvolvimento de uma solução modular que implemente uma interface de controle portátil para ciclismo por estimulação elétrica funcional.

1.3 OBJETIVOS

1.3.1 Objetivo principal

Desenvolver e realizar validação preliminar de uma interface de controle portátil para ciclismo por estimulação elétrica funcional. Essa interface deve permitir a interação do usuário com o sistema em tempo de execução, sendo modular, flexível, reutilizável e portátil.

1.3.2 Objetivos secundários

- Investigar os tipos de parâmetros e variáveis que devem ser disponibilizados aos usuários e quais recursos gráficos possibilitam a aplicação do sistema em diferentes escopos do ciclismo assistido por EEF;
- Desenvolver a funcionalidade de modificação de parâmetros em tempo de execução para sistemas em ROS, aplicados ao projeto EMA;
- Implementar uma solução portátil para o ciclismo assistido por EEF que seja compatível com a maioria dos dispositivos móveis atuais;
- Desenvolver uma interface gráfica para o ciclismo assistido por EEF que possibilite a reusabilidade em diferentes aplicações de EEF;
- Implementar e documentar aspectos da solução que permitam a reutilização de códigos e a modularização, e flexibilização dos processos;
- Realizar validação preliminar do sistema em experimentos piloto com indivíduo com lesão medular.

1.4 ORGANIZAÇÃO DO MANUSCRITO

O Capítulo 2 apresenta uma revisão bibliográfica das classificações de software aplicados a dispositivos médicos, assim como, uma explanação sobre a estimulação elétrica funcional

e uma apresentação do *Robot Operating System* e do estado da arte dos sistemas de ciclismo assistidos por EEF.

O Capítulo 3 detalha os materiais e métodos utilizados neste trabalho, descrevendo as características da solução, os componentes mecânicos e eletrônicos que foram adaptados ao sistema para este fim e como foi aferido o desempenho do sistema.

O Capítulo 4 apresenta os resultados da proposta de uma interface de controle portátil para ciclismo por estimulação elétrica funcional que permite a mudança dos parâmetros em tempo de execução utilizando ROS e a discussão desses resultados.

O Capítulo 5 apresenta as conclusões do trabalho e propõe melhorias e trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Esse capítulo apresenta uma revisão bibliográfica dos seguintes temas: tipos de reabilitação e atividade física para pessoas com lesão medular, utilização da EEF como parte da reabilitação para pacientes com lesão medular, importância do software para esses sistemas e sua regulação no Brasil, Robot Operating System (ROS) e estado da arte do ciclismo assistido por EEF.

2.1 ESTIMULAÇÃO ELÉTRICA FUNCIONAL

A Estimulação Elétrica Funcional (EEF) é uma técnica que utiliza impulsos elétricos para estimular os nervos periféricos. O estímulo elétrico pode ser aplicado em qualquer lugar ao longo do comprimento do nervo, desde sua origem até seu ponto motor, onde ele se conecta ao músculo. A carga elétrica necessária para ativar uma contração muscular por estimulação direta das fibras musculares é muito grande para ser aplicada repetidamente com segurança [17]. Os nervos motores periféricos estimulados que inervam o tecido muscular proporcionam a atividade muscular mesmo que os músculos estejam fracos ou paralisados por doenças neurológicas ou lesões.

O recrutamento muscular na EEF pode ser realizado utilizando eletrodos superficiais, eletrodos percutâneos, eletrodos implantáveis. Os eletrodos de superfície são colocados sobre a pele ao longo do comprimento do nervo dos músculos a serem estimulados. Não são invasivos, são mais baratos e fáceis de aplicar, mas não conseguem isolar músculos profundos, podem irritar a pele e causar dor. Os eletrodos percutâneos são passados através da pele e colocados próximos ao ponto motor dos músculos. Eles são utilizados apenas para intervenções de curto prazo. Fornecem alta seletividade e uma resposta repetível. Os eletrodos implantáveis são implantados em procedimentos cirúrgicos e são colocados ao redor dos nervos musculares ou sobre os músculos. Eles são caracterizados por alta seletividade, embora apresentem problemas ligados ao implante cirúrgico que incluem infecção e rejeição do eletrodo. Devido ao implante de longo prazo, também ocorrem degradações eletroquímicas e falhas mecânicas.

A EEF pode ser usada para gerar uma função corporal que o paciente é incapaz de realizar. Também pode ser usada como terapia para ajudar o sistema neuromuscular a reaprender

a execução de funções prejudicadas. Os principais campos de aplicação são após o derrame e a lesão da medula espinhal. A EEF também pode ser usada para complementares outras terapias de movimento e aumentar o potencial da reabilitação.

A força do estímulo elétrico em termos de amplitude e duração do sinal determina o número de fibras nervosas ativadas e a força da contração muscular. Claramente, as fibras sensoriais dentro do nervo periférico estimulado são ativadas, assim como as fibras motoras, que podem ser dolorosas, se a sensação for preservada. Esta dor e dano ao tecido estão inversamente relacionados ao tamanho do eletrodo, por causa da alta densidade de carga e corrente.

A Figura 2.1 apresenta o nervo periférico, os eletrodos, os parâmetros da corrente e os tipos de pulsos utilizados na EEF [18]. Além da amplitude e da duração do estímulo elétrico, a frequência do pulso e a forma de onda também são importantes. A frequência de pulso precisa ser alta o suficiente para gerar uma contração do músculo liso. O estímulo em si pode ser monofásico ou bifásico, mas geralmente estímulos bifásicos balanceados são usados clinicamente, pois fornecem melhor controle sobre a força de contração muscular e são menos propensos a causar danos aos tecidos [19].

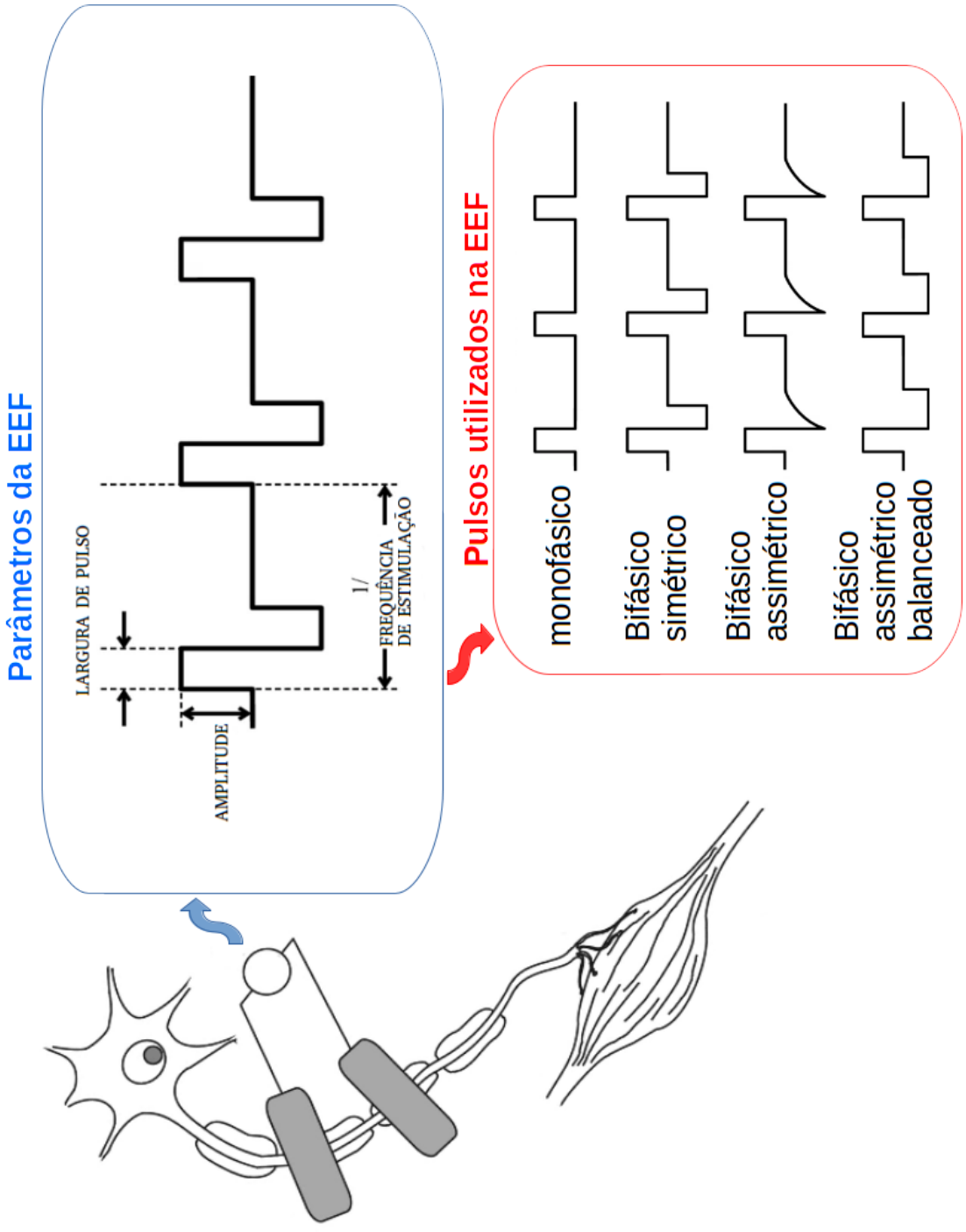


Figura 2.1: Os eletrodos, representados por retângulos cinzas com bordas chanfradas, são colocados sobre o nervo periférico conectado ao músculo. No retângulo azul com bordas chanfradas são apresentados os parâmetros do sinal de onda da EEF. No retângulo vermelho com bordas chanfradas são apresentados os tipos de sinais utilizados na aplicação da EEF.

2.2 REABILITAÇÃO E ATIVIDADES FÍSICAS PARA PESSOAS COM LESÃO MEDULAR UTILIZANDO ESTIMULAÇÃO ELÉTRICA FUNCIONAL

A escolha do tipo de reabilitação e atividade física para o paciente com lesão medular é realizada pelo fisioterapeuta. Existem diversos exercícios que podem ser assistidos por estimulação elétrica funcional para pacientes com lesão medular. A Tabela 2.1 apresenta alguns exemplos da gama de reabilitações que podem ser aplicadas à pacientes com lesão medular. Esses artigos são trabalhos publicados no período de 2011 a 2019, e exemplificam as diferentes formas de atividades físicas para pessoas com lesão medular e seus benefícios [20].

Autor (ano publicação)	Técnica de tratamento	Resultados Obtidos
ALENCAR et al (2011)	Facilitação neuromuscular proprioceptiva	Favorece a reaquisição dos padrões motores
MEDINA et al (2012)	Alongamento, fortalecimento muscular, facilitação neuromuscular proprioceptiva	Maior força de tronco, redução da hipotensão postural
SOUZA et al (2013)	Marcha com assistência robótica	Melhoria nos resultados comparados com a terapia física convencional
NAS et al (2015)	Eletroestimulação funcional (FES), treino de marcha	Evitou perda de tônus muscular, conservou a densidade óssea e melhorou o funcionamento do sistema respiratório
LEÃO et al (2017)	Videogame XBOX 360® e jogos de Kinect, na postura ortostática por 20 minutos seguido da postura sentada por 20 minutos	Melhora do equilíbrio permitindo maior segurança para realização das atividades nas posturas de sedestação e bipedestação
Bo, Antonio PL, et al (2017)	Ciclismo por EEF	Ajuda a prevenir problemas cardiovasculares, osteoporoses, obesidade e diabetes tipo II
da Fonseca, L. O., et al (2019)	Remo por EEF	Ajuda a prevenir problemas cardiovasculares, permite a ativação muscular do corpo todo

Tabela 2.1: Exemplos de estudos envolvendo técnicas de reabilitação de pacientes com lesão medular publicados no período de 2011 a 2019.

Apesar das diferentes características das técnicas de reabilitação, a maior parte dos métodos apresentados abaixo utilizam a EEF como meio de proporcionar a movimentação muscular, e dispõem de ao menos um sensor e um estimulador. Dessa forma um sistema modular

capaz de adicionar novos sensores sem grande esforço computacional, assim como uma solução flexível e portátil facilitaria o processo de controle e supervisão do sistema. Nessa seção são apresentadas algumas formas de exercício utilizando EEF.

2.2.1 Marcha assistida por estimulação elétrica funcional

A técnica da marcha assistida por EEF dispõe de bastante literatura devido ao seu tempo de aplicação. Ela também é utilizada em intervenções terapêuticas temporárias para melhorar a função voluntária. O método comumente utilizado para restauração da posição ortostática é a aplicação de estimulação elétrica no quadríceps. A restauração ou melhora da marcha envolve a estimulação de dois ou mais locais. Esses locais podem ser o quadríceps, durante a fase de apoio da marcha, e o nervo fibular, produzindo uma resposta de flexão padronizada durante a fase de balanço do membro ipsilateral [21]. Outras adaptações ortopédicas podem ser utilizada para estabilizar o tornozelo, joelhos e quadris. A aplicação dessa técnica está associada a melhora do condicionamento cardiovascular e a prevenção da atrofia muscular.

A ativação da estimulação muscular pode se realizada de forma voluntária, ou por um sistema de controle, por exemplo um sistema de detecção de intenção de movimento derivada de sensores. A qualidade da marcha é restringida pela força limitada dos músculos estimulados eletricamente e pela incapacidade de padrões de estimulação se ajustarem às mudanças das propriedades musculares (por exemplo, fadiga [22]]) ou mudanças no ambiente (por exemplo, inclinação, superfície). Um sistema de controle para caminhada por EEF deve levar em consideração as características musculares e o perfil de marcha desejado.

2.2.2 Ciclismo assistido por estimulação elétrica funcional

O ciclismo assistido por EEF utiliza um sistema de controle para a ativação coordenada dos músculos inferiores. O *software* é responsável por contrair os músculos e permitir a movimentação cíclica guiada. Geralmente, os grupos de quadríceps, isquiotibiais e glúteos são ativados em uma sequência apropriada, fora de fase bilateralmente, para manter um torque adequado nos pedais.

Os resultados da aplicação dessa técnica no contexto de reabilitação para pacientes com LM tem demonstrado melhora no metabolismo, no sistema cardíaco e na condição pulmonar [9, 11]. Também é observada a melhoria geral das condições fisiológicas e psicológicas, incluindo a possibilidade da recuperação parcial da função das pernas em indivíduos com lesão medular incompleta.

Alguns sistemas comerciais estão disponíveis como da BerkelBike (BerkelBike BV, AV's-Hertogenbosch, Holanda), Ergys e Regys (Therapeutic Alliances, Fairborn, Ohio, EUA) e Motomed (Reck, Betzenweiler, Alemanha) .

Em geral, o ciclismo associado a EEF pode ser dividido em dois tipos principais, móveis e estacionários. O tipo móvel possibilita a locomoção do usuário e tem aplicação em pesquisas, lazer e competição. O tipo estacionário é geralmente usado para treinamento de exercícios aeróbicos em indivíduos com LM para condicionamento de força muscular e melhora da função cardiopulmonar.

Vários grupos de pesquisa desenvolveram sistemas para ciclismo por EEF usando triciclos padrões ou reclinados para sujeitos com LM. Além do triciclo, conforme mostrado na Figura 2.2, o sistema é geralmente composto por um estimulador elétrico e sensores. Algumas adaptações mecânicas, também podem ser necessárias, dependendo do quadro do paciente. As adaptações comuns incluem encostos para o peito, suportes para estabilidade, alças para os pés e guidão e pedal adaptado. Apesar dessas adaptações ainda podem existir diferenças importantes no alinhamento dos membros inferiores e no desenvolvimento muscular em comparação com o outro membro, sugerindo que mais adaptações possam ser necessárias para permitir o ciclismo, dependendo do paciente [23].

A utilização dessa técnica no contexto de competição foi aplicada no Cybathlon. Em uma das suas modalidades, pessoas com deficiência física competem entre si utilizando o ciclismo por EEF. Esse evento destacou o potencial da tecnologia na participação em exercícios e treinamento físico para pessoas com deficiência e forneceu uma visão sobre os parâmetros que as equipes participantes usaram em seus esforços para maximizar a eficiência no ciclismo por EEF.

Apesar dos grandes benefícios dessa técnica para indivíduos com LM, eles podem ser mitigados dada a possibilidade de fadiga acelerada ou esgotamento físico, inerentes as características de técnicas utilizando EEF. As limitações mencionadas são associadas as características do recrutamento utilizando estimulação elétrica de superfície, que apresenta baixa seletividade [24]

2.2.3 Remo assistido por estimulação elétrica funcional

Estudos demonstraram os efeitos positivos do remo assistido por EEF na saúde cardiovascular e respiratória em indivíduos com lesão medular. Essa técnica também tem sido aplicada a indivíduos logo após a lesão na medula [25] e foi observado beneficio para a saúde óssea, embora essa fase seja esperado maior taxa de diminuição da massa óssea.

O remo por EEF é composto por um equipamento de remo ergômetro em ambiente interno e um estimulador, podendo ser adicionado sensores a esse sistema. A ativação muscular geralmente é realizada em membros inferiores por meio de estimulação elétrica funcional. Essa estimulação permite ao usuário o trabalho tanto de membros inferiores, ativados por EEF, quanto de membros superiores, recrutando grandes grupos musculares no processo.

Para a utilização dessa técnica em pacientes com lesão medular, são necessárias algumas adaptações para a execução do movimento. A estrutura se baseia em uma plataforma estática e um assento móvel, conforme mostrado na Figura 2.3, e um suporte postural. Os eletrodos são colocados de forma a recrutar o grupo muscular do quadríceps, proporcionando a extensão do joelho. Sensores podem ser adicionados para a criação de uma lógica de controle da estimulação.

2.2.4 Natação assistida por estimulação elétrica funcional

O nado assistido por EEF busca melhorar o nado de indivíduos com paralisia. O movimento da perna com o movimento voluntário do braço devem ser sincronizados para evitar um rolamento indesejado do corpo durante a braçada. Em alguns casos a estimulação da medula espinhal é usada para obter uma posição de natação reta. Duas abordagens estão sob investigação. A primeira abordagem envolve um sensor no tronco para sentir o ângulo de rotação do corpo, a fim de sincronizar a estimulação das pernas com os movimentos voluntários detectados do braço [26]. E na outra abordagem, o nadador sincroniza os movimentos do braço com a estimulação da perna por meio de um feedback eletrotátil do status da musculatura da perna [27].

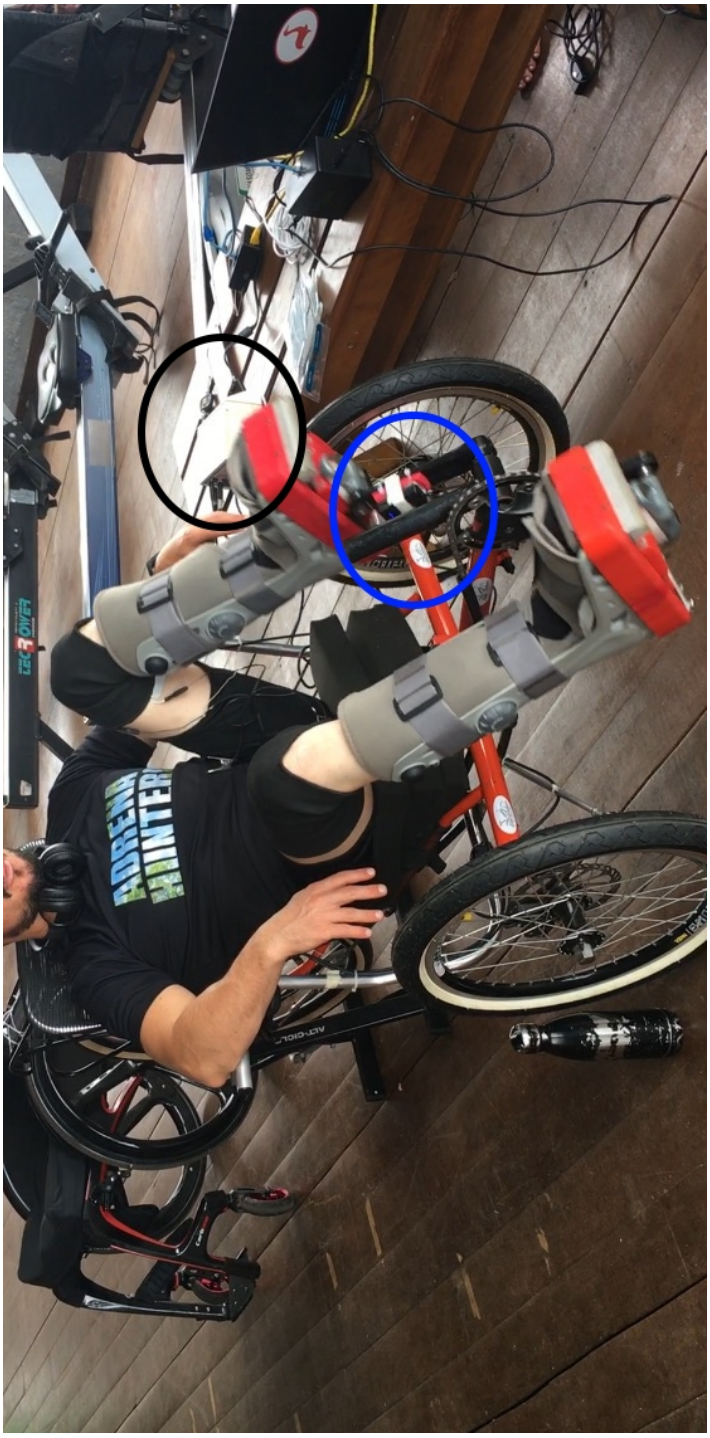


Figura 2.2: Exemplo de um triciclo utilizado para o ciclismo por EEF. Dentro do círculo preto é apresentado o estimulador elétrico. Dentro do círculo azul está apresentado o sensor de posição do pedal. Algumas das adaptações necessárias para guiar o movimento foram as botas e seu acoplamento no pedal.

Figura 2.3: Aplicação do remo assistido por EEF no Projeto EMA. Os círculos em laranja identificam sensores utilizados para o controle.

2.3 SOFTWARE E REGULAMENTAÇÃO

O início da regulamentação para produtos de interesse a saúde começou na década de 1970. Nessa década o Estado Brasileiro começou o processo de criação de ferramentas para estabelecer diretrizes para tais produtos. Esse processo se deu a partir da criação da Secretaria de Vigilância Sanitária, pela Lei nº 6.360/76, e a criação do Sistema Nacional de Metrologia, Normalização e Qualidade Industrial (Sinmetro), pela Lei nº 5.966/73. Nesse contexto, foi fomentada a política nacional de metrologia, normalização industrial e certificação de qualidade Industrial (Conmetro) e a criação do Instituto Nacional de Metrologia, Normalização e Qualidade Industrial (Inmetro).

Com a promulgação da Lei nº 9.782/99 criou-se a Agência Nacional de Vigilância Sanitária (Anvisa), que substituiu a Secretaria de Vigilância Sanitária, obtendo maior autonomia. Essa mudança possibilitou que a maior parte das normas fossem constituídas por instruções normativas (IN) e resoluções da diretoria colegiada (RDC). Atualmente, a Anvisa é responsável pela classificação e cadastro de equipamentos médicos e também disponibiliza um manual com as diretrizes para o processo de registro por meio de um manual em seu sítio [28].

Junto a Anvisa, o Sinmetro também é responsável pelo processo de certificação para garantir a conformidade dos equipamentos com as normais da Vigilância Sanitária. Nesse contexto, o Inmetro acredita os Organismos Certificadores de Produtos (OCP) e Laboratórios de Ensaio (LE). Os OCP são organismos independentes e são incumbidos de conduzir o processo de certificação. Também identificam os laboratórios para realização de ensaios e realizam auditorias no processo produtivo. Esse processo normativo visa manter a segurança e qualidade dada a sensibilidade dos produtos aplicados a saúde.

Com a emissão da nota técnica nº 04/2012/GQUIP/GGTPS/Anvisa, a Anvisa também foi responsável por direcionar as empresas que desenvolvem aplicativos para produtos de saúde. Essa nota técnica serviu como um guia para essas empresas, diferenciou o enquadramento sanitário dos softwares e apresentou os requisitos do dossiê técnico para esses softwares. Ela foi necessária devido a falta de legislação específica no Brasil [29]. O software foi enquadrado conforme o contexto de aplicação, diferenciando conforme sua aplicação na saúde. Para o seu registro seria suficiente a entrega dos arquivos de gerenciamento de risco (segundo a norma ABNT NBR ISO 14971), dos relatórios de estudos e dos testes para verificação e validação da segurança e eficácia do equipamento, realizados segundo critério definidos pelo próprio fabricante [1]. Vale ressaltar que a classificação do risco do software acompanha a classificação do hardware, e softwares embarcados são isentos de registro obrigatório apenas

do software, cujo registro deve ser incluído na solicitação de registro do equipamento médico ao qual se destina.

No Seminário de Dispositivos Médicos, promovido pela Anvisa em 2016, houve uma apresentação sobre Regulação de Software no Brasil [30]. Embora a norma IEC 62304 não tenha sido internalizada pela ABNT, ela foi apresentada como uma forma de auxiliar na comprovação de segurança e eficácia de software de dispositivos médicos. Um dos conceitos previstos pela IEC 62304 é que o ambiente de desenvolvimento do software de dispositivos médicos também realize a gestão de qualidade e gerenciamento de risco. Dessa forma, os padrões das normas ISO 13485 e ISO 14971, nortearam a gestão de qualidade e risco. A Figura 2.4 apresenta os requisitos impostos pela IEC 62304, assim como, o entendimento das diretrizes apresentadas pelas três normas para o desenvolvimento de software de sistemas mecatrônicos aplicados à saúde.

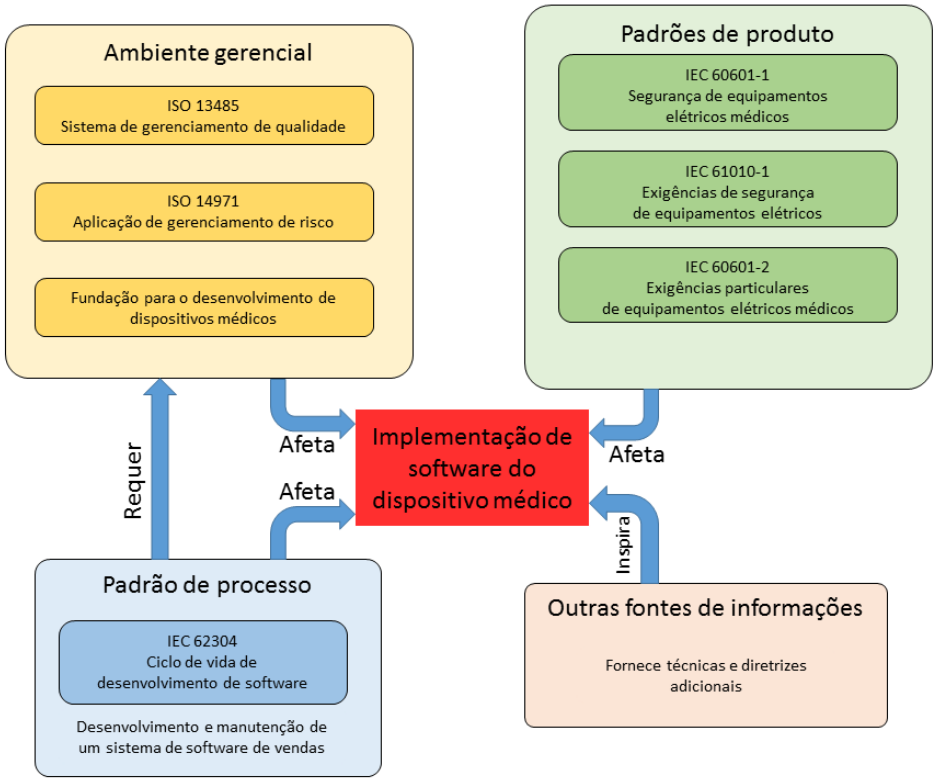


Figura 2.4: Diretrizes normativas para o desenvolvimento de software aplicado a saúde, fonte [1].

2.4 SOFTWARES

2.4.1 *Sistema Operacional*

O sistema operacional (SO) é um programa ou um conjunto de programas que gerencia os recursos do sistema definindo qual programa deve ser priorizado pelo processador ou conjunto de processadores. Ele também é responsável por gerenciar a memória, gerenciar arquivos, dispositivos de entrada e saída e dados da máquina e seus periféricos. O SO fornece uma interface entre o usuário e o sistema, a complexidade dessa interface deve ser compatível com o público pretendido.

O SO é responsável pela comunicação das pessoas que utilizam o computador como uma ferramenta dentro da sua área de atuação (usuário), dos equipamentos conectados (como exemplo a memória e outros *hardwares*), dos programas, dos utilitários e compiladores. Alguns usuários do SO podem ter diferentes acessos ao sistema. As características desses usuários são:

- Usuário sem acesso completo aos recursos do sistema, como instalação de programas ou recursos físicos;
- Operadores de computador, responsáveis pela monitoração do sistema operacional;
- Programadores de aplicação, profissionais que desenvolvem software aplicativo para um determinado tipo de máquina e determinado sistema operacional;
- Programadores de sistema;
- Responsáveis pela manutenção do sistema operacional;
- Administrador do sistema, responsável pelo controle da utilização da máquina, seus recursos e softwares, cadastramento de usuários, oferecer ou retirar direitos a determinadas operações.

Dessa forma, o sistema operacional tem as funções básicas de interpretar os comandos do usuário, controlar os periféricos (teclado, vídeo, discos, impressora, mouse, impressoras) e organizar arquivos em disco [31].

2.4.2 *Framework*

Na literatura não há uma definição única para *framework*, podendo surgir dificuldades terminológicas. Apesar disso, grande parte dos autores compreendem que um *framework* é

um conjunto de classes que incorpora um design abstrato para soluções de uma família de problemas relacionados. Em outas palavras, um *framework* é um projeto de implementação parcial abstrata para um aplicativo em um determinado domínio de problema [32].

As características básicas de um *framework* incluem a reusabilidade, extensibilidade, facilidade de utilização e documentação da solução. O *framework* contém funcionalidade abstrata e deve ser eficaz no domínio das soluções propostas [33].

2.4.3 *Middleware*

O *middleware* é um software responsável por conectar um conjunto de componentes em um ambiente distribuído para oferecer melhor funcionalidade. Esses componentes podem ser aplicativos, uma tarefa dentro de um aplicativo, uma plataforma, uma rede de comunicação, uma peça de hardware, um servidor, um cliente, um serviço, um nó, entre outros recursos.

As características do *middleware* podem ser exemplificadas pelos serviços usados para traduzir e integrar dados de diferentes fontes, funções usadas para oferecer funcionalidades operacionais comuns, ferramentas usadas para simplificar o desenvolvimento de aplicativos, modelos usados para facilitar a reutilização e integração de aplicativos (reduzindo os esforços de desenvolvimento e evitando a duplicação de serviços).

O *middleware* tem o objetivo de simplificar o desenvolvimento de aplicativos complexos, oferecendo abstrações e interfaces de alto nível para facilitar a integração, reutilização e desenvolvimento. Ele é responsável por deixar invisível para o sistema a heterogeneidade das plataformas e ocultar os detalhes de distribuição e comunicação no ambiente subjacente. Dessa forma, facilita a comunicação entre os diferentes componentes distribuídos da infraestrutura. Também visa fornecer uma arquitetura comum para adicionar novos serviços e recursos sem ter que mudar os aplicativos, oferecendo recursos de valor agregado e propriedades não funcionais como segurança, confiabilidade e qualidade. Em suma, acontece uma facilitação para manter a estabilidade e escalabilidade dos aplicativos distribuídos [34]. Abaixo são apresentados alguns *middlewares* conhecidos pela comunidade.

2.4.3.1 YARP

YARP é um conjunto de bibliotecas, protocolos e ferramentas para manter os módulos e dispositivos em um ambiente distribuído. O YARP não é um sistema operacional, ele é um middleware que não detém controle do sistema. Normalmente ele é utilizado em projetos de robótica, e busca tornar o software do robô mais estável e duradouro, sem comprometer a

flexibilidade de alteração dos sensores, atuadores, processadores e redes. Também ajuda a organizar a comunicação entre sensores, processadores e atuadores a partir de um modelo de comunicação neutro em termos de transporte, de forma que o fluxo de dados é desacoplado dos detalhes das redes e protocolos subjacentes em uso. O YARP é um software livre e aberto, junto com muitos outros benefícios, o contrato social do Software Livre pode acelerar o desenvolvimento de software para pequenas comunidades [35].

2.4.3.2 Rock

Rock é um software para o desenvolvimento de sistemas robóticos. O modelo é baseado no Orocos RTT *(Real Time Toolkit)*. Essa solução fornece todas as ferramentas necessárias para configurar e operar sistemas robóticos de alto desempenho. Ele contém uma rica coleção de *drivers* e módulos prontos para uso e pode facilitar a adição de novos componentes [36]. A estrutura foi desenvolvida para abordar especificamente os seguintes problemas:

- Relatório e o tratamento de erros;
- Escalabilidade;
- Código reutilizável.

2.4.3.3 Robot Operating System (ROS)

O Robot Operating System (ROS) é um *software* aplicado normalmente a robótica. Ele se tornou uma ferramenta amplamente utilizada para robôs, como máquinas PR2, Baxter, Fetch, Meka e UBR-1 [37]. Nesse cenário, um número crescente de estudos sobre ROS vem sendo realizado. Por exemplo o robô EL-E utilizou dessa tecnologia para controlar de forma eficiente o braço móvel [38].

Criado em 2008, o ROS é um *middleware* com uma comunidade ativa, que permite diferentes linguagens de programação, flexível e foi desenvolvido para incentivar a reutilização de código e a colaboração de seus usuários. O ROS é constituído por uma coleção de ferramentas, bibliotecas e padrões que têm como objetivo simplificar o tratamento complexo de tarefas simultâneas e funcionar em diferentes plataformas.

Esse software foi desenvolvido como uma estrutura para integrar rapidamente os recursos computacionais em ambientes operacionais distribuídos. O ROS contém ferramentas para executar uma comunicação consistente entre procedimentos como tópicos e nós. Com sua

estrutura em nós, é possível tornar o sistema modular e melhorar a atribuição dos recursos, como também obter maior repetibilidade [39].

Um nó ROS é um programa executável que contém uma parte de um código. Uma aplicação ROS pode ter mais de um nó, por exemplo, podemos ter um nó para calcular a cinemática inversa, um nó para enviar comandos aos atuadores, um nó para obter os valores do ângulo dos sensores. Todos esses nós se comunicam entre si por meio de mensagens. Existe um nó mestre em execução que terá todos os endereços dos nós em execução no sistema. É responsabilidade do nó mestre conectar os nós entre si para iniciar a comunicação [40].

Os tópicos são barramentos nomeados pelos quais os nós trocam mensagens. Este é o modo de comunicação mais comumente usado para nós em ROS. Nesse modo, um nó transmite uma mensagem com um nome de tópico. Este nó de transmissão é conhecido como nó publicador. Um nó que deseja receber a mensagem se inscreverá neste nome de tópico. Este nó é conhecido como nó assinante.

Este modo de comunicação é unidirecional. Os nós não sabem com quem estão se comunicando, ou seja, um publicador continuará publicando os dados mesmo se não houver assinantes múltiplos. É responsabilidade do nó mestre estabelecer a conexão entre o publicador e o assinante quando estiverem no mesmo tópico.

Diversas bibliotecas foram desenvolvidas pela comunidade para facilitar sua aplicação em diferentes contextos. Algumas desses bibliotecas possibilitam uma solução portátil em ROS, essas bibliotecas são a ROS Android e a roslibjs.

A biblioteca ROS Android é utilizada para criar aplicativos Android que se comuniquem com o ROS. A Figura 2.5 apresenta algumas características da biblioteca ROS Android, essas características mostram que essa biblioteca é baseada em rosjava e se comunicação com o Android Studio e seus plug-ins [2]. O código-fonte oficial para projetos Android pode ser encontrado no github. Essa biblioteca apenas tem suporte para o ROS na versão Kinetic, e dependendo do sistema operacional, não é possível a instalação dessa versão de ROS.

A roslibjs é a biblioteca de funções que permitem o estabelecimento de conexões via protocolos de Internet entre um navegador e o sistema em ROS. Essa comunicação é realizada por meio de funções em JavaScript e um servidor *websocket*. Ele é responsável por estabelecer conexões de "soquete" entre um navegador da web, um webserver e um servidor *websocket*. Ao se criar a conexão persistente entre o cliente e o servidor, ambas as partes podem começar a enviar dados a qualquer momento. Essa biblioteca utiliza um servidor de *Wwebsocket* denominado rosbridge que abstrai o core do ROS permitindo acesso as suas

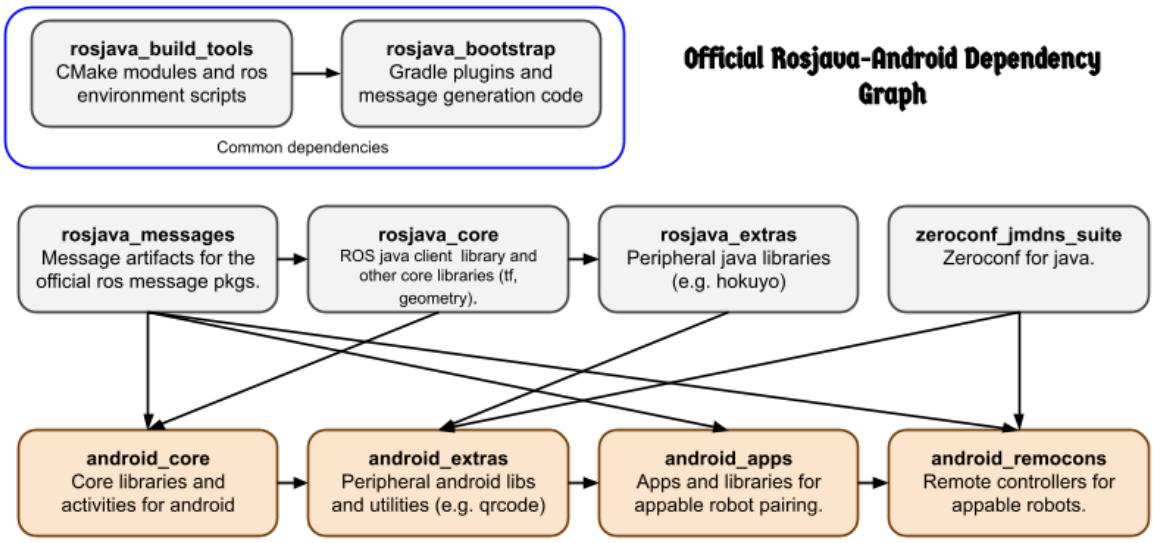


Figura 2.5: Representação gráfica das características da biblioteca ROS Android. Ela é baseada em rosjava, representado em retângulos azuis, e se comunicação com o Android Studio e seus *plug-ins*, reapresentados pelos retângulos em rosa , fonte [2].

funcionalidades. O webServer é um computador que hospeda sites. Ele é capaz de processar e entregar páginas da web aos usuários, conforme sua requisição. Essa requisição é feita usando o protocolo HTTP. Além de HTTP, um servidor da web também oferece suporte aos protocolos SMTP (Simple Mail transfer Protocol) e FTP (File Transfer Protocol) para envio de e-mail e transferência de arquivos e armazenamento.

2.4.4 WebSocket

O *websocket* define uma API que permite estabelecer conexões entre um navegador da web e um servidor. Essa conexão se inicia com o cliente enviando uma solicitação HTTP (Hypertext Transfer Protocol) para o servidor. Essa conexão *websocket* se estabelece por meio de um processo conhecido como *handshake*. Um cabeçalho é incluído no pedido do cliente que informa ao servidor que o cliente deseja estabelecer uma conexão *websocket*. Se o servidor suportar o protocolo *websocket*, ele concordará com a tentativa de conexão e comunicará isso através do cabeçalho na resposta.

Após a conclusão do *handshake*, ambos conseguem enviar dados, tanto o servidor, como o cliente. Essa conexão permite a transferência de maior quantidade de dados diminuindo a sobrecarga associada às solicitações HTTP. Os dados são transferidos como mensagens entre o cliente e o servidor. As mensagens são constituídas por *frames*, o cliente apenas será notificado sobre uma nova mensagem quando todos os *frames* forem recebidos e os

dados da mensagem original tiverem sido reconstruídos, e assim garantir que a mensagem seja adequadamente reconstruída. Cada *frame* dispõe de um cabeçalho, e são compostos por blocos apresentados abaixo [41]:

- fin (1 bit): O fim da mensagem.
- rsv1 , rsv2 , rsv3 (1 bit cada): Se um valor diferente de zero for recebido e nenhuma das extensões negociadas definirem o significado desse valor, deverá falhar a conexão.
- opcode (4 bits): Código para interpretação dos dados enviados. 0x00 : continuação do *frame*; 0x01 : *frame* de texto; 0x02 : dados binários; 0x08 : término da conexão; 0x09 : ping; 0x0a : pong.
- mask (1 bit): Indica se o conteúdo está mascarado.
- payload length (7 bits, 7+16 bits, or 7+64 bits): O tamanho da carga útil. De 0 a 125 indica o comprimento da carga útil. Se 126, os dois bytes seguintes indicam o comprimento, se 127, os próximos 8 bytes indicam o comprimento.
- masking-key (32 bits): Os quadros enviados do cliente para o servidor são mascarados por um valor de 32 bits contido no quadro.
- payload : Os "dados de carga útil" são definidos como "Dados de extensão" concatenados com "Dados do aplicativo".

Os endereços para o *websocket* apresentam um modelo de URL diferenciado, introduzido por ws://, que é descriptografado, e wss:// que requer criptografia TLS. Caso a aplicação seja disponibilizada na internet, é importante o uso da criptografia.

Além das características citadas acima, o websocket também possui ampla compatibilidade de navegadores e plataformas, demonstrando a grande portabilidade dessa tecnologia. A Figura 2.6 apresenta a compatibilidade dessa solução com a a maioria dos navegadores utilizados. Os retângulos em verde apresentam as versões dos navegadores em que essa solução é suportada, os retângulos em marrom apresentam as versões dos navegadores em que o *websocket* é parcialmente suportado, e os retângulos em vermelho são as versões em que não é suportado.

2.4.5 JavaScript

O JavaScript é uma linguagem de programação que foi projetada para ser executada no navegador do usuário. Ela foi criada em Maio de 1995, por Brendan Eich, por solicitação

Figura 2.6: Compatibilidade do *websocket* com diferentes navegadores, fonte [3].

da Netscape. Sua função inicial era manipular o básico do *front-end*. Mocha foi o primeiro nome dessa linguagem criada por Brendan.

A partir do recebimento da licença registrada pela Sun, a linguagem de programação Mocha tornou a ser oficialmente a LiveScript, com o intuito de difundir a linguagem, foi modificado, novamente, o nome da linguagem para JavaScript. A escolha do nome foi uma estratégia de marketing e trouxe dúvidas sobre as similaridades do JavaScript com o Java.

A ECMA (European Computer Manufacturers Association) é responsável pela criação de especificações relacionadas as linguagens *scripts* [42]. Em 1997, o JavaScript foi submetido as essas especificações , dessa forma, o JavaScript se tornou a implementação mais popular desse padrão. Em 1998 implementou-se o ECMAScript 2, seguido do ECMAScript 3, em 1999. Já em 2000, iniciou-se o trabalho no ECMAScript 4.

O desenvolvimento do Ajax (Asynchronous JavaScript and XML), por Jesse James Garrett, em 2005, colocou o JavaScript em foco novamente. Pois houve a criação de novas comunidades, bibliotecas e frameworks open source, por exemplo, o jQuery, e o Mootools. Em Julho de 2009 houve o desenvolvimento do ECMAScript 5, embora um marco na história do JavaScript foi o desenvolvimento do Node.js, um interpretador open source multiplataforma, criado por Ryan Dahl. Esse projeto foi apresentado em 8 de Novembro de 2009, na European JSConf.Essa criação permitiu que o JavaScript fosse executado tanto no lado do cliente quanto no servidor.

Em 2011, um gerenciador de pacotes para Node.js foi implementado estabelecendo padrões para o desenvolvimento e distribuição de bibliotecas e módulos, chamado npm. Com essa disseminação do JavaScript, é possível notar sua aplicação em diferentes contextos, tanto em aplicativos móveis, bem como HTML e CSS, Apache Cordova, PhoneGap e React Native. Outra grande aplicação para essa linguagem de programação é a utilização na Internet das Coisas (IoT) que permite a essa solução o controle também de hardware. Alguns *frameworks* auxiliam na implementação de soluções com essa aplicação, por exemplo, o Cyclon.js e Johnny-Five.

2.5 ESTADO DA ARTE DO CICLISMO ASSISTIDO POR ESTIMULAÇÃO ELÉTRICA FUNCIONAL

Como apresentado anteriormente , o ciclismo associado a EEF pode ser dividido em dois tipos principais, móveis e estacionários. As suas aplicações podem ser em pesquisas,

lazer e competição. Embora existam algumas soluções comerciais em ciclismo por EEF para pacientes com lesão medular, essas soluções não são muito difundidas na aplicação clínica por fisioterapeutas. Esses sistemas comerciais são da BerkelBike (BerkelBike BV, AV's-Hertogenbosch, Holanda), Ergys e Regys (Therapeutic Alliances, Fairborn, Ohio, EUA) e Motomed (Reck, Betzenweiler, Alemanha).

O RT300-SLSA, da empresa Restorative Therapies, é um equipamento estático. Ele possibilita o ciclismo por EEF e disponibiliza a movimentação dos braços. Esse equipamento pode ser acoplado a cadeira de rodas, sem a necessidade de transferência, ou pode ser usado a partir de uma cadeira, ou bola de equilíbrio. Por meio de um sensor no pedivela da perna as informações sobre a posição das pernas são enviadas para uma caixa eletrônica central que calcula qual músculo da perna precisa ser estimulado. A proposta dessa empresa é disponibilizar a reabilitação na casa do paciente, proporcionado conforto [43]. A BerkelBike desenvolveu um triciclo reclinado ajustável que pode funcionar através do uso de ambos os braços e pernas. A movimentação dos braços proporciona a movimentação das pernas. Também por meio de um sensor no pedivela da perna as informações sobre a posição das pernas são enviadas para uma caixa eletrônica central que calcula qual músculo da perna precisa ser estimulado. A bicicleta pode ser usada como um triciclo completo em si, ou como um acessório removível para uma cadeira de rodas

A grande maioria dos estimuladores comerciais disponíveis não proporcionam a exploração abrangente das possibilidades que essa técnica permite, pois não possibilitam a escalabilidade (adicionar outros sensores por exemplo) ou alterações no sistema de controle, como disposto em [44]. Alguns grupos de pesquisa tem desenvolvidos soluções para o ciclismo por EEF. Alguns desses sistemas buscaram utilizar o controle em malha fechada com a retroalimentação a partir dos valores dos sensores e atuadores em uma *trike*, proporcionando o uso do ciclismo por EEF móvel [45]. Outros estudos buscaram a combinação da potência do motor com a força aplicada pela perna, induzida pela estimulação, para possibilitar também o uso do ciclismo por EEF móvel por mais tempo. Essas soluções híbridas, apresentadas abaixo, possibilitam a utilização recreativa e móvel, pois pode proporcionar a movimentação mesmo após fadiga muscular.

ROUSE, C. A. et al propuseram um trabalho em que o controle da potência de saída da perna (via ajuste automático da intensidade da estimulação) é compatibilizado com a cadência do triciclo (via controle do motor elétrico) [15] e também propôs um controlador de cadência responsável por regular a cadência para a região desejada e um controlador de torque discreto, responsável por manter a potência média. Nessa solução o indivíduo é incentivado a contribuir voluntariamente e o motor apenas auxilia no rastreamento da cadência

quando necessário (por exemplo, a pessoa está fadigada ou não tem força muscular suficiente para atingir a cadência desejada) [16]. HUNT, K. J. et al propuseram um controle integrado com dois sistemas em malha fechada independente. O primeiro sistema é composto pela entrada do motor elétrico, que é ajustada automaticamente, em malha fechada, modificando seu sinal de entrada a partir do valor da cadência. O segundo sistema fornece controle da força da perna, em malha fechada. A intensidade da estimulação é modificada por meio da largura de pulso que é ajustada automaticamente para manter a potência medida próxima ao valor de referência. Esse valor é definido por meio do software de controle.

Existem algumas dificuldades para aumentar a abrangência do uso do ciclismo por EEF. Como exemplo, profissionais da saúde encontram a barreira tecnológica para a utilização desse método, isso advém das características das tecnologias aplicadas a EEF serem mais comuns na engenharia, tais como linguagens de programação, forma de apresentação dos dados coletados do ciclista e adição e remoção de recursos. Outra dificuldade encontrada é a flexibilidade dos sistemas, pois se torna mais difícil a adição de sensores e atuadores embarreirando a sua utilização em diferentes casos.

No trabalho de JASUJA, A. et al [46] um sistema utilizou dispositivos de EEF e apresentou uma estrutura de arquitetura aberta e de baixo custo sem fio. Essa arquitetura permite a programação remota da rotina de estimulação muscular pelo terapeuta, e fornece informações sobre a dinâmica do músculo exercitado por meio da eletromiografia (EMG). Essa solução apresentou uma estrutura extensível embora não seria simples para o profissional de saúde utilizar novas formas de reabilitação associada a EEF, pois para adicionar ou realocar novos recursos ao sistema é necessária uma reprogramação em MATLAB.

No trabalho de CERONE, G. L. et al [47] um novo estimulador elétrico sem fio, modular e programável foi proposto. Esse estimulador integra a possibilidade de adquirir e usar sinais biomecânicos para acionar a saída da estimulação. Nesse trabalho, embora os valores dos sensores e estimuladores serem apresentados de forma dinâmica, não é possível a modificação dos parâmetros de controle em tempo de execução. Essa característica limitaria a ação do profissional de saúde, além da especificidade dos módulos que não permitiria a interação com novos sensores de forma simples.

Geralmente os trabalhos apresentados sobre ciclismo por EEF tem o foco no sistema de controle ou na concepção eletrônica do sistema. Embora possa ser parte importante no desenvolvimento do sistema de ciclismo assistido por EEF, a literatura é escassa sobre sua interface gráfica e não a explica de forma detalhada, essa característica é visualizada no trabalho de COSTE, C. A. e WOLF, P. [48] onde são apresentadas as soluções das equipes do Cybatlhon de 2016 e não são expostas as características das interfaces que controlam o

sistema.

A partir dessa lacuna de informações, outras soluções que implementassem sistemas modulares aplicado a saúde foram pesquisadas. No trabalho de BOULMALF, M. et al. [49] é apresentado um sistema portátil para aplicação na saúde, essa solução foi implementada para o sistema Android e utiliza o *middleware* para conseguir a modularidade do por meio do sistema em nós. O trabalho de BERALDO, G. et al [50] teve o objetivo de fornecer uma descrição preliminar do *ROS-Health*, um *framework* baseado no *middleware Robot Operating System* (ROS). Essa solução busca estabelecer uma plataforma de pesquisa padronizada para facilitar a distribuição do software, a replicação de resultados experimentais e a criação de uma comunidade unificada para compartilhar e gerenciar o código. Esses trabalhos não necessariamente são aplicados a reabilitação e nem a eletroestimulação. Mas são soluções que permitiram, a partir da sua correlação com esse trabalho, demonstrar a importância da aplicação do *middleware* ROS em sistemas de saúde para permitir a modularidade e flexibilidade. O sistema também deve permitir a interação com o usuário, essa interação é facilitada pela capacidade do sistema ser portátil.

3 MATERIAIS E MÉTODOS

Este capítulo apresenta o processo de desenvolvimento da interface de controle portátil para o ciclismo por estimulação elétrica funcional (EEF). Inicialmente é explicado o contexto do ciclismo por EEF no projeto Empoderando Mobilidade e Autonomia (EMA). Também são apresentados: a estrutura das soluções propostas por esse trabalho, sua aplicação no contexto embarcado, o desenvolvimento do software que permite a utilização em dispositivos móveis e a forma de avaliação de desempenho, findando o capítulo com a descrição do estudo experimental.

3.1 CICLISMO POR EEF NO PROJETO EMA

O projeto Empoderando Mobilidade e Autonomia (EMA) visa desenvolver tecnologia assistiva para pessoas com deficiência motora [51]. Esse trabalho foi elaborado como parte do projeto EMA. O ciclismo por EEF já estava sendo aplicado a participantes com lesão medular (LM) no projeto EMA anteriormente a esse trabalho [52, 53]. Para a sua execução, alguns hardwares eram utilizados: um *tadpole trike* (modelo HP3 20 X 26 CS 24V, comercializado pela HP3 Trike), um par de botas adaptadas fixadas ao pedal, um estimulador elétrico (RehaStim Hasomed), um sensor inercial (IMU) (3-Space, Yost) e um computador portátil (executando o Ubuntu 18.04).

Em 2016, o sistema do ciclismo por EEF apresentado pelo projeto EMA na competição Cybatlhon foi desenvolvido em Python. A arquitetura desse sistema não foi concebida utilizando um middleware e esse sistema foi embarcado em um computador de placa única de tamanho reduzido (Raspberry Pi 3 B+) [45]. Após o fim do Cybathlon, o grupo começou os esforços para desenvolver um sistema que facilitasse a sua modificabilidade, reusabilidade, verificabilidade e proteção. Uma nova implementação foi proposta para o sistema, essa nova solução utilizava o *middleware Robot Operating System* (ROS) [1].

Com a implementação dessa solução em ROS, as ferramentas de apresentação dos dados da solução anterior não poderiam mais ser aplicadas e houve a necessidade de criar ferramentas para monitorar os dados dos treinos e alterar os parâmetros em tempo de execução. Para isso, foi desenvolvida, por este trabalho, uma interface utilizando o pacote *dynamic_reconfigure* e o ROS Qt. O pacote *dynamic_configure* disponibiliza um servidor

dinâmico e é uma solução eficiente para atualizar o valor de um parâmetro dinamicamente enquanto um nó está em execução. Este pacote permite atualizar parâmetros no *Parameter Server* em tempo de execução e enviar essa alteração para um determinado nó ativo que precisa do valor mais atual desse parâmetro. O *Parameter Server* (em português servidor de parâmetros) é um dicionário compartilhado que é executado internamente ao nó mestre do ROS. Ele deve ser visualizado globalmente para que as ferramentas possam inspecionar o estado de configuração do sistema e o modificar.

O ROS Qt é a implementação do Qt associado ao ROS. O Qt é uma estrutura de desenvolvimento de plataforma cruzada amplamente usada para desenvolver aplicativos que podem ser executados em várias plataformas de software e hardware com pouca ou nenhuma alteração na base de código subjacente. O ROS fornece um pacote que ajuda a construir o ambiente Qt em ROS e fornece uma WIKI para documentar a sua implementação [54].

Essa solução não havia sido embarcada anteriormente ao Cybathlon 2020, sendo executada em um computador portátil com o Ubuntu 18.04. Com os preparativos para o Cybatlhon 2020, novos objetivos para o projeto EMA foram definidos e a necessidade de embarcar o sistema ROS foi colocada como prioridade. Essa necessidade adveio da busca por mobilidade. Então, modificações no código foram implementadas por membros do projeto EMA que permitiram embarcar o sistema em ROS na Raspberry Pi 3 B+, substituindo o uso do *notebook*.

A partir do desenvolvimento da funcionalidade de modificação dos parâmetros em tempo de execução, por meio da interface em ROS Qt [55], novos esforços foram realizados para implementar uma solução de interface de controle portátil para o sistema em ROS. Essa nova interface buscava disponibilizar mais ferramentas gráficas e aumentar a acessibilidade por meio da portabilidade. Isso poderia facilitar o uso por profissionais de saúde, pacientes e também poderia ajudar no monitoramento do ciclista no contexto da competição Cybathlon. Essa melhora no monitoramento poderia proporcionar melhores desempenhos do atleta durante a competição.

3.2 ESCOLHA DA ARQUITETURA

Anterior a esse trabalho, a arquitetura foi desenvolvida utilizando a linguagem de programação Python. Essa implementação foi dedicada para um estimulador e IMU específicos. Por ser uma implementação dedicada a inserção de novos módulos demandava grande esforço computacional e dificultava a sua modularidade e reusabilidade.

Os softwares de desenvolvimento de sistemas poderiam ser utilizados como solução que disponibilizaria interfaces para o monitoramento, alteração dos parâmetros em tempo de execução e inserção de módulos, como exemplo o LabView ou Matlab/Simulink. Embora sejam soluções muito difundidas, geralmente elas são softwares proprietários. Essa característica dificultaria desenvolver ferramentas personalizadas, além do fator financeiro atrelado a tais soluções.

Então, com o intuito de implementar uma estrutura de arquitetura aberta, de baixo custo, com rápida integração de recursos computacionais em ambientes operacionais distribuídos, com capacidade de ser embarcada e que permitisse o desenvolvimento de uma interface de controle portátil, o *middlewear* ROS foi escolhido. Alguns fatores contribuíram para essa escolha, tais como:

- Integração com diferentes hardwares;
- Implementação de sistemas de controle complexos;
- Facilidade de desenvolver módulos (nós) por meio de diferentes linguagens de programação;
- Possibilidade de reutilizar códigos em diferentes aplicações, com baixo esforço computacional;
- O aspecto modular do sistema que permite a adição de novos nós, ou seja, novos equipamentos. A possibilidade desses equipamentos serem executados em sistemas distribuídos, o que facilitaria a identificação de uma possível falha.
- A ampla utilização do ROS na comunidade da robótica, muitos drivers e plugins disponibilizados para sensores e atuadores comerciais, sem necessidade de grande esforço computacional para utilizá-los.
- O grande arcabouço teórico e bibliotecas em ROS, disponibilizadas pelo projeto EMA, para aplicação no ciclismo por EEF.

Embora o ROS não possua garantia de tempo real, não tenha qualidade de serviço, não execute o gerenciamento de tempo de vida dos nós e apenas pode ser utilizado no SO Linux. O alto engajamento da comunidade é um dos pontos que diferenciam o ROS das outras soluções de *middleware*. Essa característica colaborou para a sua escolha em detrimento das outras opções.

3.3 DESENVOLVIMENTO GERAL

A solução inicialmente implementada por esse trabalho para o ciclismo por EEF [55] foi concebida utilizando o ROS sendo executado em um notebook. A interação dessa solução com o usuário era por meio de uma interface em ROS Qt (RQT). Em seguida, esse trabalho implementou uma nova solução utilizando o ROS sendo executado em um computador de placa única de tamanho reduzido (Raspberry Pi 4). A interface de interação com o usuário não utilizava mais o ROS Qt. Essa mudança ocorreu para proporcionar uma interface portátil que pudesse ser utilizada em um *tablet*, celular, computador. Essa nova interface foi programada em JavaScript e permitiu por meio de um webserver e um servidor websocket a comunicação com o sistema em ROS. A Figura 3.1 apresenta graficamente os dois sistemas, a esquerda é a implementação utilizando o ROS Qt, a direita a implementação utilizando uma interface portátil que pode se comunicar com um computador, *tablet*, celular.

A primeira solução apresentada utilizou o ROS Qt (RQT) e um notebook para executar todo o sistema. Para essa solução a camada de processo representa o sistema em ROS sendo executado em um computador. O sistema em ROS permitiria adicionar diversos nós ao sistema, tais como: o sensor inercial (IMU), o estimulador e o nó de controle. Na camada servidor, também sendo executado no computador, é apresentado o servidor dinâmico que utiliza o pacote *dynamic_reconfigure* do ROS. Esse servidor é responsável por alterar os dados do processo em tempo de execução. A camada de comunicação apresenta um plugin do ROS Qt que permite a comunicação da interface em ROS Qt (GUI) com o sistema em ROS. A camada GUI e a camada de comunicação são executadas no *notebook*. Para essa solução o acesso à interface com o usuário somente é possível por computadores que tenham instalado ROS.

A segunda solução apresentada utilizou uma interface web. Para essa solução a camada de processo representa o sistema em ROS sendo executado na Raspberry. O sistema em ROS permitiria adicionar diversos nós ao sistema, tais como: o sensor inercial (IMU), o estimulador e o nó de controle. Na camada de servidor, executada também na Raspberry, temos dois servidores, o dinâmico e o servidor websocket/plugin. O servidor dinâmico, que utiliza o pacote *dynamic_reconfigure* do ROS, é responsável por alterar os dados do processo em tempo de execução. O Servidor *websocket* utiliza a biblioteca *roslibjs* do ROS que permite implementar um servidor websocket com o nome *rosbridge* e tem a natureza de plugin, pois transforma a informação HTTP em uma informação utilizável pelo sistema ROS. Na camada comunicação, que também é executada na Raspberry, foi utilizado o *webserver* nativo do Python que é responsável por endereçar as requisições da interface web para o

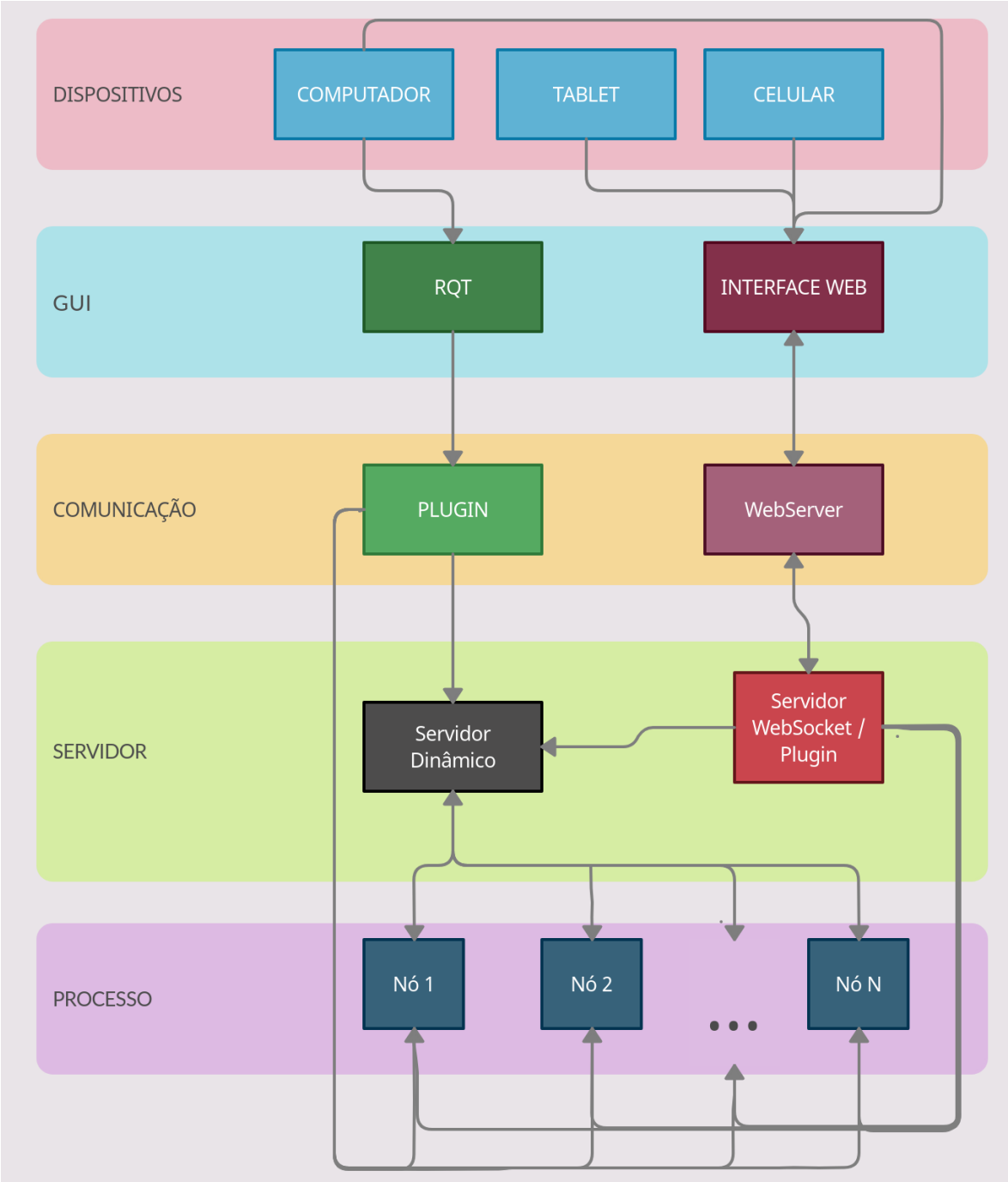


Figura 3.1: Representação gráfica de duas soluções para o ciclismo por EEF, a esquerda a implementação utilizando o ROS Qt, a direita a implementação utilizando uma interface portátil que pode se comunicar com um computador, *tablet*, celular. As setas em cinza representam a comunicação entre os módulos apresentados por retângulos.

servidor *websocket*. Na camada GUI, a comunicação com os dispositivos é realizada por meio de uma rede Wifi e permite que qualquer dispositivo que esteja na rede wifi que o

webserver possa utilizar essa interface, tornando o sistema mais acessível e portátil.

3.4 RECURSOS DAS APLICAÇÕES

Essa seção apresenta as duas soluções e suas etapas de implementação. Inicialmente são apresentadas as modificações do sistema utilizando a Robot Operating System (ROS). Em seguida, é apresentado o desenvolvimento da interface em ROS QT e o desenvolvimento da interface Web aplicado ao escopo da tecnologia embarcada.

3.4.1 Robot Operating System (ROS)

O projeto EMA já possuía experiência em estudos utilizando a estimulação elétrica funcional (EEF). Uma das vertentes de estudo do projeto EMA é o ciclismo assistido por EEF. Esse trabalho foi desenvolvido com intuito de contribuir para esse projeto. No início desse trabalho o Projeto EMA utilizava um notebook para executar o sistema em ROS, que posteriormente foi substituído por um computador de placa única de tamanho reduzido da marca Raspberry.

Antes do início desse trabalho o sistema do projeto EMA para o ciclismo por EEF já utilizava o *middleware* ROS e a interação com o usuário era realizada através de botões físicos. O sistema era controlado por um nó de controle, esse nó utilizava os dados do sensor inercial (IMU) e dois botões como variáveis de entrada para gerar o seu sinal de controle. Essa solução não possibilitava a mudança, em tempo de execução, dos parâmetros de estimulação como a largura de pulso e o intervalo angular de ativação da estimulação, apenas possibilitava o incremento ou decremento da corrente. Ela também apresentava um monitoramento deficitário dos parâmetros do ciclismo. Em conjunto com um membro do projeto foi criada a possibilidade de mudança dos valores em tempo de execução e também foi desenvolvida uma interface para o monitoramento do exercício.

Essas melhorias utilizaram o pacote *dynamic_reconfigure* que implementa um servidor dinâmico do ROS, para a mudança dos parâmetros em tempo de execução. A interface foi desenvolvida em um ambiente ROS associado ao Qt. A Figura 3.2 apresenta os nós, que permitem a modularização, identificados pelos círculos. Os tópicos são representados pelas formas retangulares e as setas para dentro e para fora representam a assinatura e a publicação em um determinado tópico. Todos esses nós se comunicam entre si por meio de mensagens. Existe um nó mestre em execução que terá todos os endereços dos nós em

execução no sistema. É responsabilidade do nó mestre conectar os nós entre si para iniciar a comunicação. Os tópicos são barramentos nomeados pelos quais os nós trocam mensagens. Assim um nó que transmite uma mensagem com um nome de tópico, nó publicador, e um nó que deseja receber a mensagem se inscreverá (assinatura) neste nome de tópico, nó assinante.

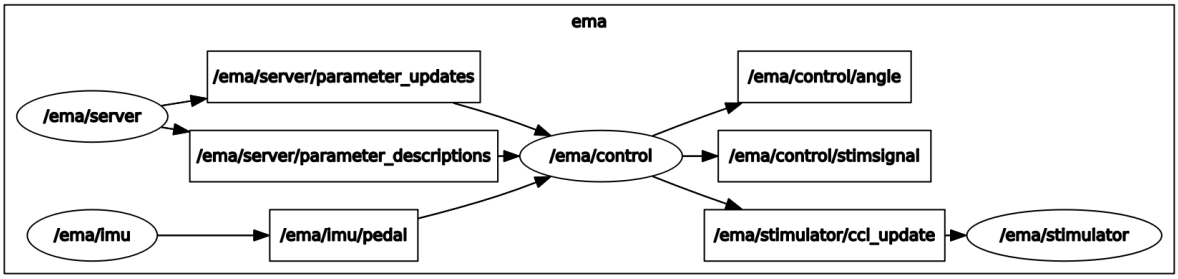


Figura 3.2: Arquitetura baseada em ROS que permite a modificação, em tempo de execução, dos parâmetros do ROS.

O */ema/server* é o servidor responsável pela configuração dinâmica, proporcionando ao sistema a capacidade de modificar os parâmetros em tempo de execução, por exemplo, o intervalo angular da estimulação. O */ema/imu* comunica-se com os sensores inerciais e obtém os dados de movimentação do pedal para publicar no tópico chamado /pedal/. O */ema/control* é o nó principal, ele controla todo o sistema. Sempre que recebe uma nova mensagem no tópico */pedal/*, ele anexa o ângulo atual a uma lista, dentro do *loop* principal e o nó de controle continua calculando quais canais devem ser ativados e desativados com base no último ângulo dessa lista. Em seguida, aumenta ou diminui progressivamente a corrente do canal, ao final do ciclo, ele publica os dados nos respectivos tópicos. O */ema/stimulator* é responsável pela atualização do estado do estimulador. O tópico *stimulator* obterá os valores atuais de acordo com o novo cálculo. No início do *loop* esse nó verifica qualquer reconfiguração de parâmetros.

Foi realizada uma investigação, baseada em entrevistas com profissionais da saúde, membros do projeto EMA e o ciclista do Cybatlhon 2016, para entender as necessidades comuns dos usuários do sistema tanto para o contexto de pesquisa, competição e reabilitação. Nessa investigação foram identificados os tipos de parâmetros e variáveis que deveriam ser disponibilizados. Apesar de serem contextos distintos esse trabalho buscou implementar as ferramentas de interação com o usuário que fossem comuns às áreas.

3.4.2 Robot Operating System (ROS) em ambiente Qt

O Robot Operating System (ROS) em ambiente Qt (ROS Qt ou RQT) e o *plugin rqt reconfigure*, foram as ferramentas utilizadas para criação da interface gráfica inicial. Então, buscando facilitar o monitoramento do sistema, utilizando como diretriz a investigação realizada previamente, foram selecionados os melhores parâmetros para serem apresentados aos usuários. Cada funcionalidade foi implementada em pequenas janelas distribuídas em uma única tela, visto na Figura 3.3.

O ROS QT permitiu implementar várias ferramentas da GUI na forma de plugins e adicionar os novos módulos como janelas encaixáveis. Essa modularidade possibilitou que a tela principal fosse desenvolvida a partir de plugins nativos, pré existentes, do próprio ROS, tais como: *robot steering, rqt plot, rqt robot monitor*. Outra demonstração de reusabilidade que o *middlewear* ROS proporcionou foi a utilização de elementos gráficos já disponíveis, por exemplo os *sliders* e o *combo box*, para a criação da interface .

Quatro funcionalidades diferentes foram implementadas para a interação com o usuário, elas são: gráfico de posição, gráfico de atuação, tabela de tópicos e alteração de parâmetros. O gráfico de posição é a janela onde o ângulo atual do pedal pode ser acompanhado durante a pedalada. No gráfico de atuação é exibida a corrente elétrica aplicada pela estimulação, em cada um dos canais. A tabela de tópicos pode ser usada para obter a comunicação entre as diferentes partes do sistema e detalhar as mensagens que estão sendo transmitidas. Já na área de alteração de parâmetros, o usuário pode modificar aspectos como a intensidade do estímulo e o intervalo angular da ativação. Embora seja possível utilizar essas funcionalidades em uma máquina diferente da que estava executando o sistema, essa solução utilizando o ROS Qt encontra um obstáculo. Esse obstáculo é caracterizado pela falta de portabilidade para diferentes dispositivos.

3.4.3 Computador de placa única de tamanho reduzido

Diversos computadores de placa única de tamanho reduzido foram pesquisados para embarcar o sistema em ROS. Inicialmente as placas com arquitetura i-386 foram cogitadas, mas devido ao alto custo e a dificuldade de aquisição, essa proposta foi inviabilizada. A Raspberry Pi 3 b+ foi o computador de placa única de tamanho reduzido que membros do projeto se empenharam para embarcar o ROS. Essa escolha foi realizada devido ao seu baixo custo, alto poder de processamento, suporte a redes sem fio, conexão Bluetooth 4.1 LE *(Low Energy)* e saída de vídeo HDMI. Essa empreitada obteve êxito e começaram os preparativos e modificações do sistema para o Cybathlon 2020. Essas modificações estavam sendo

realizadas paralelamente a este trabalho. Para possibilitar o uso de sistemas compatíveis entre as duas frentes de desenvolvimento, uma Raspberry Pi 4 com 8GB de memória RAM foi adquirida, pois possibilitaria executar a mesma solução da RaspberryPi 3 b+. O modelo foi escolhido levando em consideração sua maior capacidade computacional, comparada ao modelo anterior.

O sistema operacional (SO) utilizado foi o Ubuntu 18.04, pois era a versão mais atual do Ubuntu que seria compatível com a RaspberryPi 3 b+. Para esse sistema operacional a versão compatível do ROS é o ROS melodic. Esse sistema operacional e essa versão do ROS foram embarcados na RaspberryPi 3 b+ e na Raspberry Pi 4, pois a versão do ROS necessita ser compatível com o sistema operacional escolhido.

Após a instalação do ROS melodic na Raspberry pi 4, foi criada a estrutura do projeto utilizando os comandos em ROS e o repositório no GitHub do EMA FES Cycling [1]. Dessa maneira, foi possível obter na Raspberry pi 4 o mesmo projeto que estava sendo executado na Raspberry pi 3+, sem necessidade de alteração do código.

Embora exista a possibilidade de instalação do Ubuntu na sua versão mais atual na Raspberry Pi 4, essa escolha não possibilitaria a instalação do ROS melodic devido sua incompatibilidade com o SO. A portabilidade do ROS melodic para o ROS Noetic e do Ubuntu 18.04 para o Ubuntu 20.04 não foi realizada para manter a compatibilidade dos sistemas da Raspberry Pi 4 e da Raspberry pi 3 b+, mantendo compatíveis as duas frentes de trabalho do Projeto EMA. Dessa maneira o que foi desenvolvido nesse trabalho, também poderia ser utilizado por outras frentes de pesquisa dentro do grupo.

3.4.4 Interface portátil

Na concepção inicial, a interface de controle portátil para o ciclismo por estimulação elétrica funcional seria uma aplicação para dispositivos móveis. Dessa forma, a biblioteca ROS android seria, inicialmente, a mais indicada para esse tipo de desenvolvimento, pois ela está preparada para funcionar em conjunto com o Android Studio. Mas essa solução encontrou uma grande barreira, pois a distribuição do ROS utilizada nas Raspberry foi a Melodic e a biblioteca Android para ROS apenas tem compatibilidade com a distribuição ROS Kinetic. Dessa maneira, se tornou inviável tal implementação, pois necessitaria modificar grande parte do código já existente no sistema, para permitir a compatibilização da versão do ROS.

Como a solução ROS Android não atendia as necessidades do projeto, buscou-se outra solução para permitir a portabilidade do sistema. Essa nova solução estava pautada no de-

[1] https://github.com/lara-unb/ema_fes_cycling

senvolvimento utilizando a biblioteca roslibjs. Essa biblioteca permitiria a criação de uma interface de controle portátil para o ciclismo por estimulação elétrica programada em Javascript. A biblioteca roslibjs tem ampla aplicação, uma das suas possibilidades é a apresentação no celular de vídeos provenientes da câmera do robô, a interação tridimensional com robôs e a construção de mapas a partir de sensores do robô. Ela é desenvolvida como parte dos esforços do Robot Web Tools [56] e é composta por um servidor websocket e um plugin (rosbridge). Eles são responsáveis por receber os pacotes de dados e transformar essa informação para ser interpretada pelo ROS, o que possibilita a interação do ROS com os códigos em JavaScript.

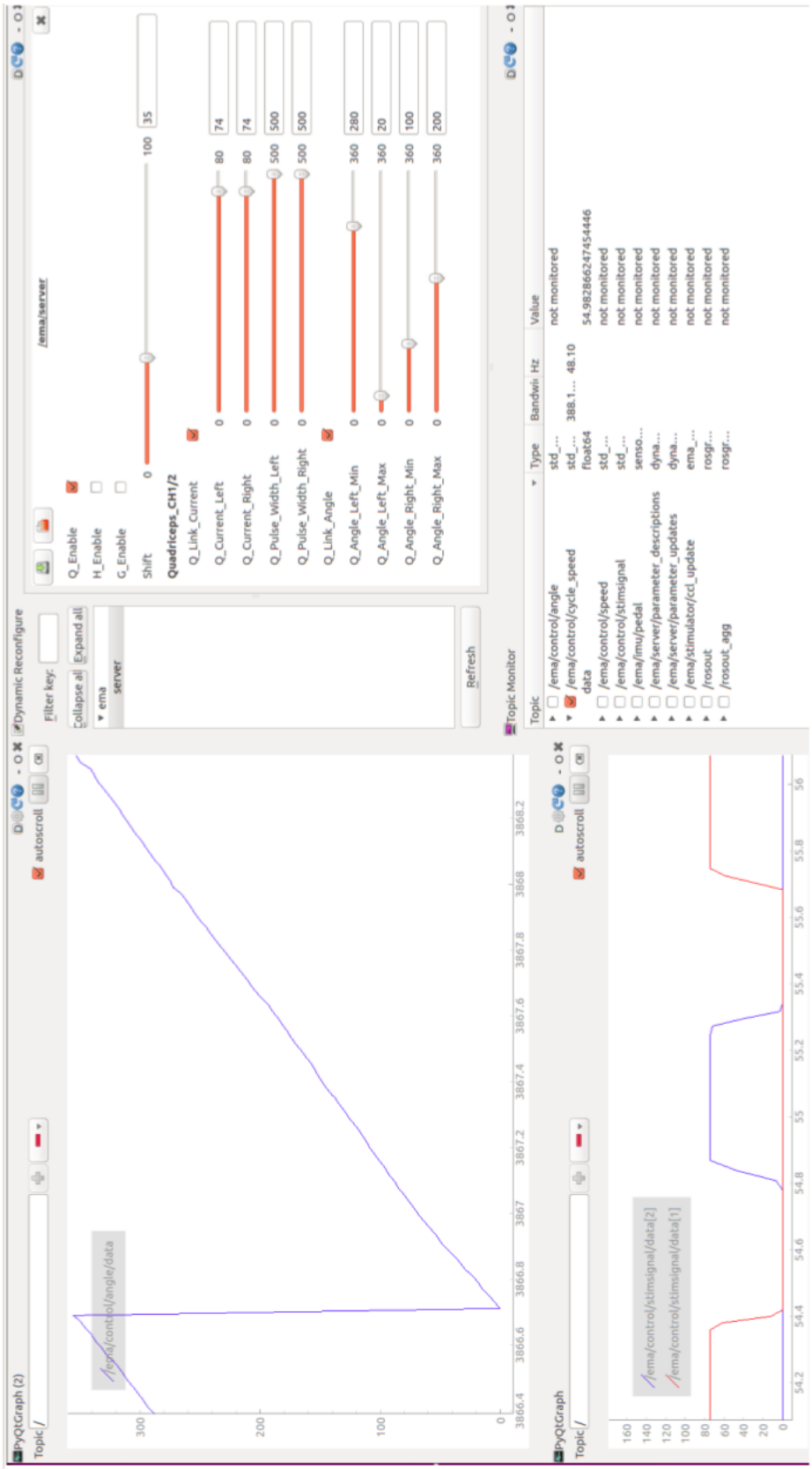


Figura 3.3: Interface gráfica desenvolvida em ROS QT.

3.5 SOLUÇÃO PARA DISPOSITIVOS MÓVEIS

Essa seção apresenta a descrição da fase de implementação da arquitetura para a interface em dispositivos móveis. Ela é dividida em ferramentas para o sistema e descrição da utilização do sistema.

3.5.1 Ferramentas para o sistema

Nessa nova etapa do trabalho voltada para o desenvolvimento da solução portátil em JavaScript surgiu a necessidade de apresentar gráficos dinâmicos (tal como o gráfico da velocidade angular do pedivela) e foram utilizados *frameworks*. O CanvasJS e o HTML 5 foram ferramentas que permitiram ao sistema web propriedades gráficas elaboradas com pouca necessidade de programação e também possibilitaram o sistema ser responsivo, ou seja, adaptável ao *display* dos dispositivos móveis. O *framework* JQuery também foi utilizado e possibilitou que requisições assíncronas AJAX fossem realizadas pelo sistema web. O *framework* em JavaScript permite a implementação de funcionalidades de forma mais simples, sem a necessidade de desenvolver desde a etapa inicial, utilizando apenas funções a partir da API. Embora essas ferramentas acelerassem o processo de desenvolvimento, também poderia diminuir a disponibilidade de recursos computacionais da Raspberry, como exemplo o poder de processamento. Testes de desempenho foram realizados para certificar que o uso dessas ferramentas não comprometeriam a solução.

A Outra prioridade para essa nova etapa foi possibilitar o uso das interfaces em ROS Qt (RQT) e da interface Web em paralelo. Dessa forma, pode-se usar a interface em ROS Qt e a interface em JavaScript. Essa característica se tornou possível pois a estrutura utilizada em nós, tópicos e servidor dinâmico do ROS permaneceu inalterada conforme apresentado no diagrama da Figura 3.1.

Para a concepção da interface utilizando JavaScript, também foi necessário o uso do HTML, página onde o JavaScript será executado. Também foi necessário o uso do CSS, responsável por modificar o estilo dos atributos dessa página, por exemplo, modificar as cores dos botões, assim como adicionar ações quando o mouse estiver por cima ou quando for clicado. Para permitir a comunicação do servidor rosbridge com a página HTML é necessário o uso de um servidor da web (webserver), foi escolhido o webserver já incluso na biblioteca Python. Um servidor da web (webserver) pode ser exposto ao público, quando conectado a internet, ou pode ser usado em uma rede interna, sem conexão externa. O uso do webserver em uma rede interna foi escolhido para minimizar os riscos de falha de segurança.

Quando o usuário solicita um site adicionando o URL ou endereço IP na barra de endereços de um navegador, um Servidor de Nomes de Domínio (DNS) converte esta URL em um Endereço IP e tal solicitação é direcionada ao servidor webserver e então o webserver apresenta o site no navegador do usuário. A biblioteca padrão do Python possui um webserver nativo capaz de ser instanciado a partir de poucas linhas de código. Embora não seja um servidor da web com todos os recursos, ele pode analisar arquivos HTML simples e servi-los com os códigos de resposta necessários. Esse webserver nativo foi escolhido para este trabalho apesar da sua pouca segurança via internet. Essa escolha foi realizada observando que inicialmente o uso da rede será apenas local, não permitindo o uso da internet e não se tornando um fator importante nessa etapa do trabalho. Além da escolha do webserver nativo do Python, também foi escolhido um ambiente virtual nativo do Python, ele foi utilizado para isolar o uso das bibliotecas do sistema operacional e do sistema do ciclismo por EEF. A Figura 3.4 apresenta os recursos que estão sendo executados na Raspberry pi 4, eles são:

- O ambiente virtual;
  - Internamente ao ambiente virtual, está sendo executado o webserver;
    - Internamente ao webserver estão os arquivos que compõe a GUI (HTML, arquivos em javascript e arquivos CSS);
- No ambiente ROS estão incluídos o SERVER (rosbridge) e o sistema de ciclismo por EEF (EMA FES Cycling).

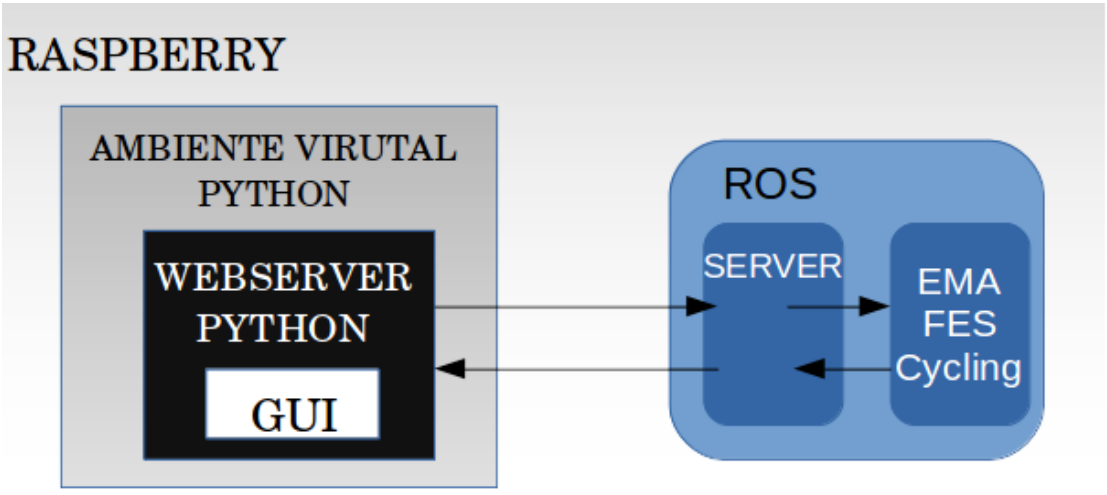


Figura 3.4: Escopos das ferramentas e a forma de comunicação entre elas.

3.5.2 Funções do Sistema

A execução do sistema é composta por algum passos a serem realizados no terminal antes do incio do ambiente em ROS. Para ser utilizado pelo usuário final, esses passos irão ser transformados em serviços do próprio sistema operacional. Primeiro é inciado o ambiente virtual nativo do Python, virtualenv. Depois do ambiente virtual iniciado, acessamos a pasta do arquivo em HTML da página principal, nomeada index e iniciamos o webserver, conforme apresentado no Código 3.1. Por fim o ambiente ROS é iniciado por meio do comando roscore.

Código 3.1: Código utilizado no shell do Ubuntu 18.04 da Raspberry pi 4, responsável por iniciar o ambiente virtual e o webserver.

```
1 $ source venv/bin/activate
2 (venv) cd webpages
3 ~/webpages$ python -m http.server
```

Nessa segunda etapa, em que as ferramentas de pré-configuração já foram iniciadas, apresentaremos a parte exclusiva do ROS. O sistema do ciclismo assistido por EEF (EMA FES Cycling) é iniciado em uma janela do shell. Em outra janela do shell iniciamos o servidor/plugin rosbridge, da biblioteca roslibjs a partir dos códigos apresentados na Figura 3.5. Na representação gráfica da Figura 3.4, o rosbridge está representado pelo SERVER. Após seu início, ele aguarda uma solicitação do webserver para converter essa solicitação em uma informação que possa ser interpretada pelo ROS e interagir modificando valores no EMA FES Cycling ou requisitando dados.

A Figura 3.6 apresenta as funções do sistema web de forma resumida. Após a conexão da página web por meio do webserver com o rosbridge, a página é atualizada com os valores das variáveis do servidor dinâmico do ROS. AS por meio da função parametros() e também as funções DrawCircle() e GraphVel() apresentam os gráficos de monitoramento a cada 100 ms. Os botões da interface web podem incrementar ou decrementar os valores das variáveis no servidor dinâmico ROS por meio das funções somaupdateitem() e menosupdateitem(), respectivamente. A função alterbool() é responsável por alterar os valores booleanos.

3.5.2.1 Conexão da interface web com o websocket

Antes de se iniciar as requisições para o servidor rosbridge, como é uma ferramente com tecnologia websocket, é necessário que ocorra o handshake entre a página/interface, que pode estar em uma rede Wifi ou internamente na Raspberry, e o server. Então, na interface gráfica do sistema de reabilitação, foi adicionado um botão que permite essa conexão a partir

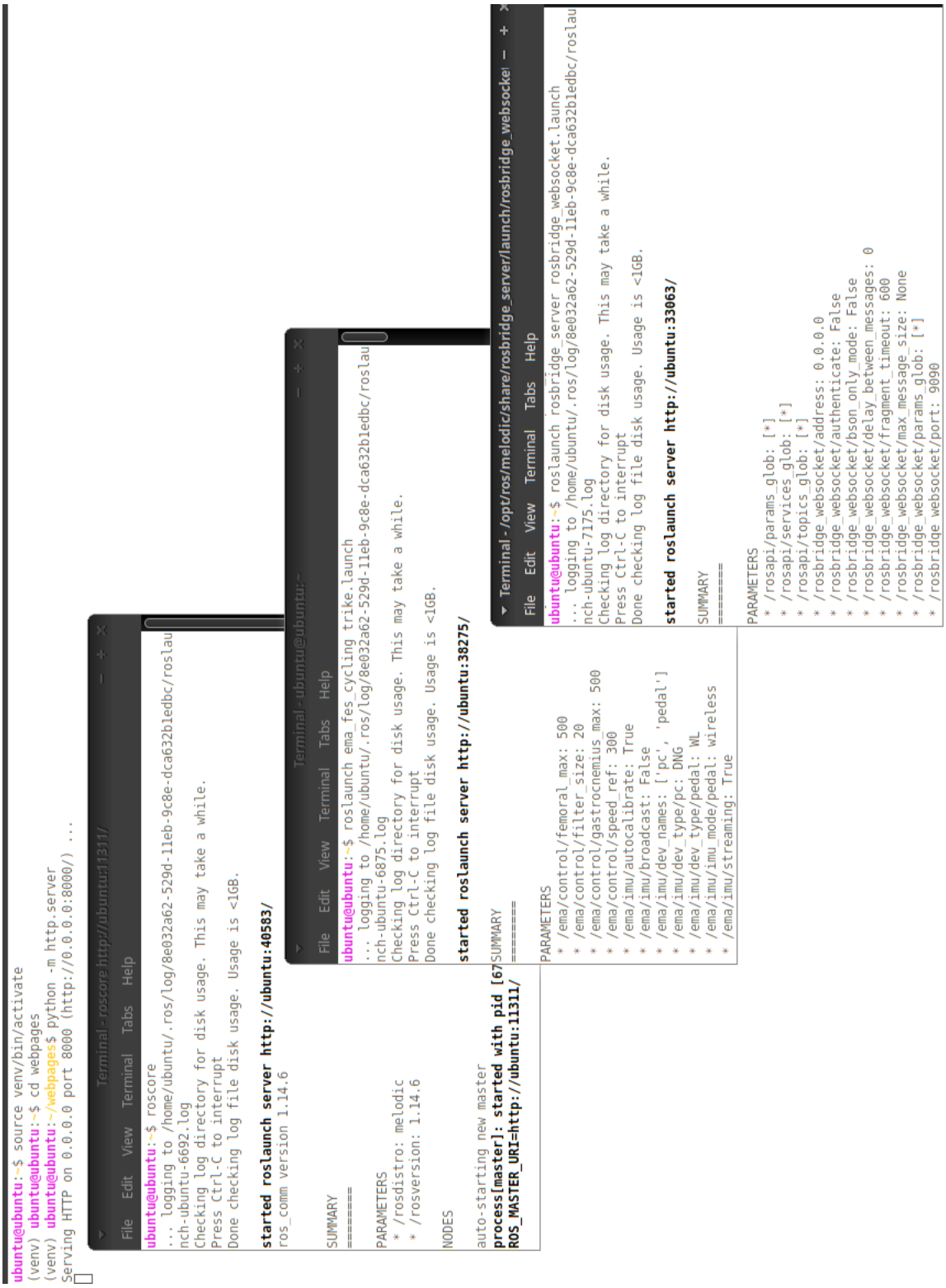


Figura 3.5: Iniciação do ROS, webserver, ambiente virtual e o servidor websocket a partir do shell.

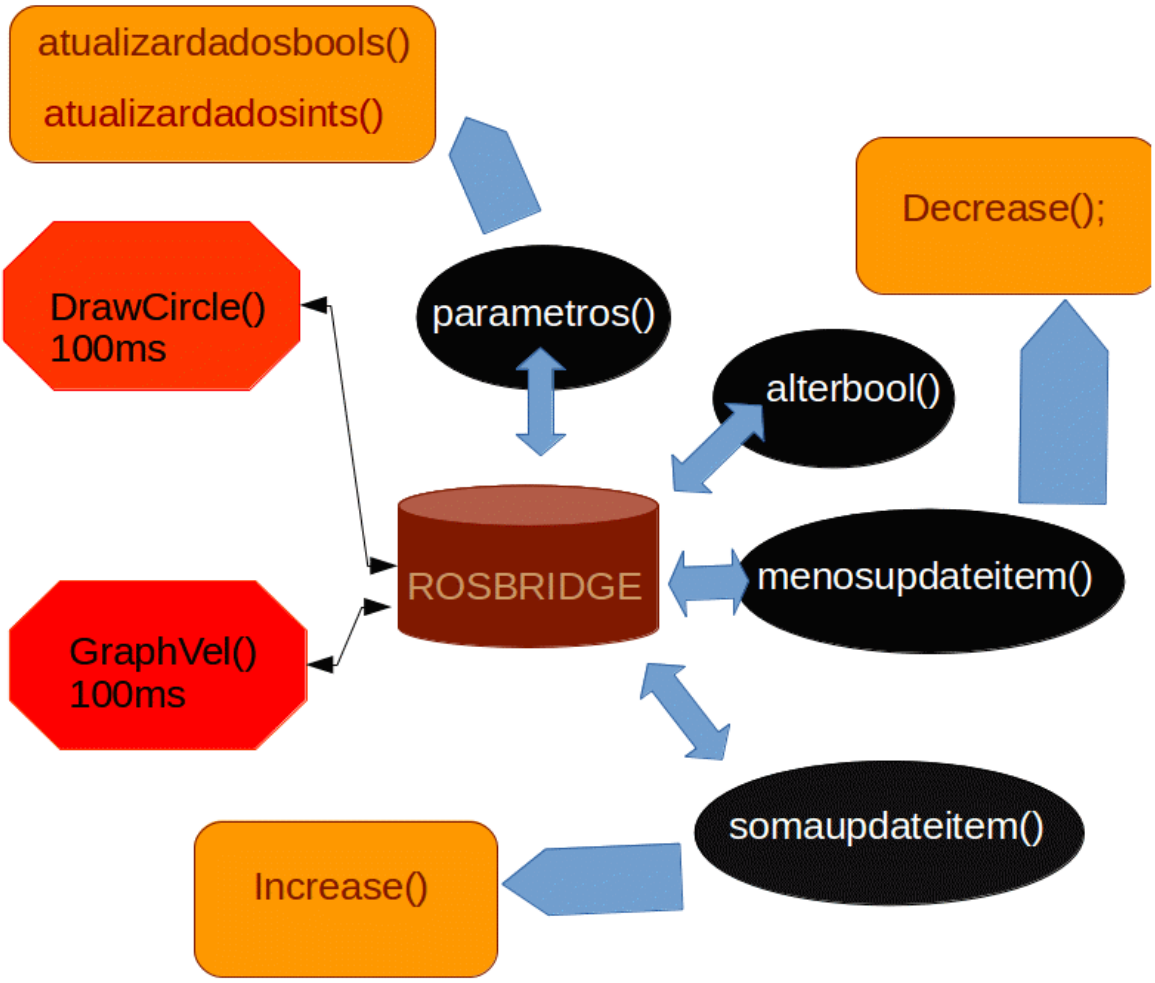


Figura 3.6: Funções de atualizar os valores da interface web e modificar os valores no servidor dinâmico ROS.

do IP da Raspberry. O Código 3.2 em javascript é responsável por tal funcionalidade. Também foi adicionado a interface um campo que apresenta o registro do status da conexão, para seu monitoramento.

Após ocorrer a conexão, por meio da porta 9090, os métodos graphVel e o DrawCircle são chamados. Esses métodos são responsáveis por iniciar os dois gráficos de monitoramento. A Figura 3.7 apresenta esses dois gráficos. O primeiro gráfico representa a velocidade angular do pedivela em rad por segundo e o segundo gráfico apresenta a posição angular do pedivela em tempo de execução, a partir dos dados da IMU. Como a tecnologia utilizada para gerar esses gráficos é o javascript, eles são executados no navegador do dispositivo, esse dispositivo pode acessar o sistema através de uma rede Wifi ou esses gráficos podem ser visualizados em um navegador do SO da Raspberry.

Código 3.2: Código utilizado para gerar o handshake entre o servidor websocket rosbridge e a interface gráfica em JavaScript.

```
connect: function() {
        this.loading = true
        ros = new ROSLIB.Ros({
             url: this.ws_address
        })
        ros.on('connection', () => {
             this.logs.unshift(
             (new Date()).toTimeString()+'-Connected!')
             this.connected = true
             chart.options.title.text ="Velocidade Angular";
             setInterval(graphVel,100)
             setInterval(drawCircle,100)
             parametros();
       })
```

3.5.2.2 Apresentação dos gráficos de monitoramento

Para realizar o monitoramento foi necessário definir uma taxa de atualização dos valores. Como o sistema de estimulação e o sistema de controle utilizam uma taxa de atualização de 50 Hz, uma taxa de amostragem de 10 Hz não sobrecarregaria o sistema, e apresentaria os dados de forma satisfatória e segura.

No Código 3.3 é apresentado o código responsável pela criação e atualização do gráfico de velocidade angular. O tópico /ema/control/speed utiliza uma mensagem do tipo Float e foi escolhido para a extração dos dados da velocidade angular em rad/s. Após a definição das características do layout do gráfico e após a assinatura no tópico, os dados de tempo e de velocidade recebidos por meio do tópico são adicionados ao gráfico. Esse processo de aquisição de dados é repetido de forma recursiva a cada 100 ms.

No Código 3.4 é apresentado o código responsável pela criação e atualização do gráfico de monitoramento da posição do pedivela a partir dos dados do sensor inercial (IMU). O tópico /ema/control/angle foi escolhido para a extração dos dados do ângulo da IMU. Após se definir as características do layout do gráfico e realizar a assinatura no tópico, os dados são adquiridos e o gráfico é desenhado na interface do usuário, utilizando o HTML 5, de forma recursiva a cada 100 ms.

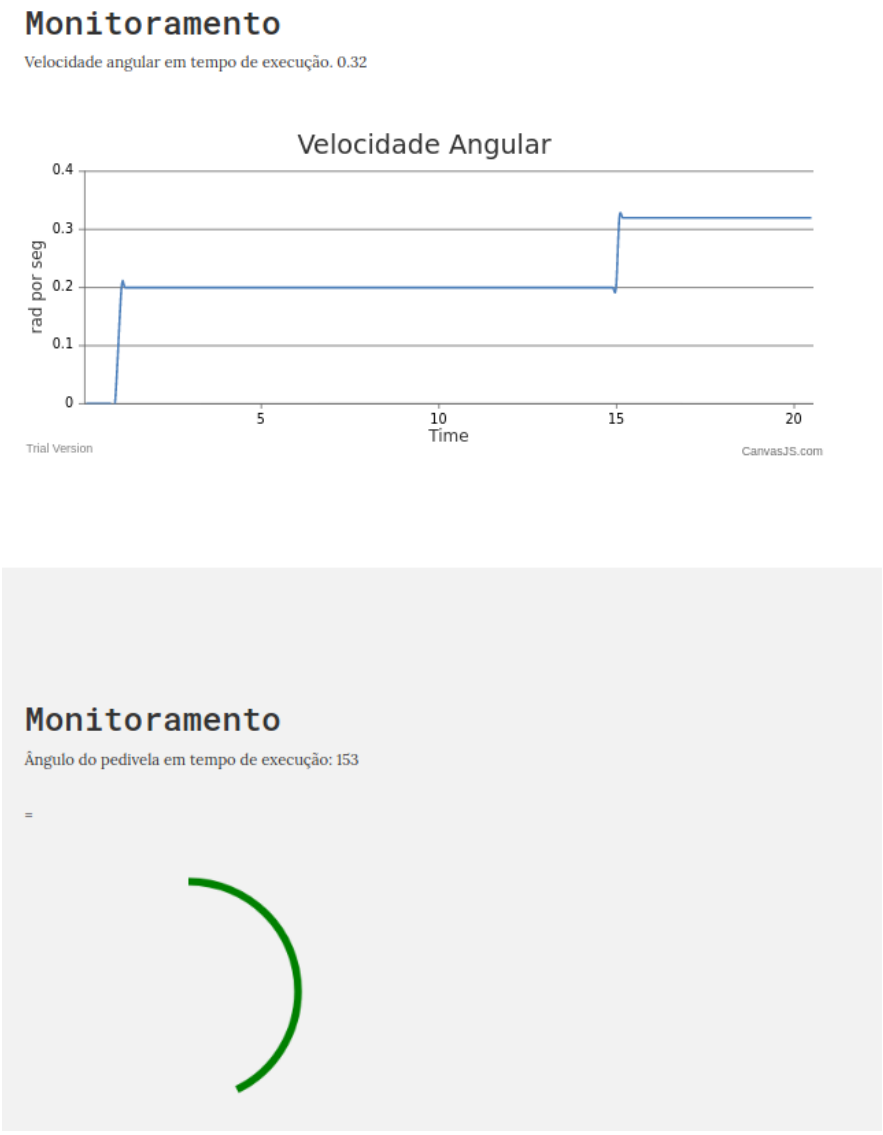


Figura 3.7: Apresentação do Layout dos gráficos de monitoramento.

3.5.2.3 Alteração dos valores no sistema ROS

A mudança dos parâmetros em tempo de execução é permitida utilizando o servidor dinâmico de parâmetros, dessa forma, pode-se utilizar a interface web e o ROS Qt em paralelo, pois ambas as soluções utilizam o mesmo servidor dinâmico.

Por meio da interface gráfica é possível habilitar ou desabilitar um canal do estimulador, ativar ou desativar o sistema de controle em malha fechada, selecionar o ângulo mínimo e máximo de estimulação do canal, a largura de pulso da estimulação e o valor da corrente de estimulação, assim como definir se as correntes dos canais adjacentes permaneceram interligadas.

Código 3.3: Código que permite a criação e atualização do gráfico de velocidade angular.

```
function graphVel(){
            var listener = new ROSLIB.Topic({
                ros : ros,
                 name : '/ema/control/speed',
                  messageType : 'std_msgs/Float64'
                  });

            listener.subscribe(function(message) {
            document.getElementById("Display-vel")
            .innerText = 2*Math.PI*(message.data/360);
            auxfreq=auxfreq+freq;
            dps.push({
                x:(auxfreq/1000),
                y:2*Math.PI*(message.data/360)
                });
            chart.render();
            listener.unsubscribe();
            });
            };
```

Para a modificação dos parâmetros no servidor dinâmico do ROS, por meio da interface em javaScript (interface web), foi utilizado o código apresentado no Código 3.9. Após a conexão com o rosbridge é feita a requisição para modificação de parâmetros utilizando o nome do parâmetro, que está definido no ROS, e seu novo valor. Ao chamar o serviço de modificação de parâmetros é iniciado um callback, onde são apresentados os novos valores no console do navegador. Ressalta-se que após a conexão ser realizada, os valores das variáveis armazenadas no servidor dinâmico são apresentados em tela.

Para a modificação dessas variáveis é utilizado um slider, um botão para aumentar o valor, e outro para diminuir. Esse layout permite o controle do valor a ser inserido no servidor dinâmico. Essa forma modular foi utilizada para quase todas as variáveis, conforme apresentado na Figura 3.8. Essa Figura é a imagem da tela de um sistema android acessando a interface.

3.5.3 Ferramenta Ambiente Virtual

Essa ferramenta foi utilizada por permitir separar o projeto, suas dependências e bibliotecas propiciando que o sistema no ambiente em ROS não interfira no sistema operacional da Raspberry. Dessa forma, esse ambiente é isolado e permite o uso de diferentes bibliotecas em projetos distintos, executados simultaneamente. A ferramenta de ambiente virtual cria uma pasta dentro do projeto, chamada de venv podendo ser renomeada. Essa pasta é consti-

Código 3.4: Código que permite a criação e atualização do gráfico de posição do pedivela.

```
    chart = new CanvasJS.Chart("chartContainer", {
        title : { text : "Dynamic Data" },
        axisX: { title: "Time"    },
        axisY: {title:"rad por seg"},
        data : [{type : "spline",dataPoints : dps}]
    });
    function drawCircle() {
        var listener = new ROSLIB.Topic({
            ros : ros, name : '/ema/control/angle',
            messageType : 'std_msgs/Float64'});
        listener.subscribe(function(message) {
             ang= message.data;
             var canvas =
              document.getElementById('myCanvas');
             var context = canvas.getContext('2d');
              var centerX = canvas.width / 2;
              var centerY = canvas.height / 2;
              var radius = canvas.width/3;
             context.clearRect(
             0, 0, canvas.width, canvas.height);
             document.getElementById("Display-angle")
             .innerText = ang;
             ang=2*Math.PI*(ang/360);
             if(ang < (2*Math.PI)){
                context.beginPath();
                context.arc(
                centerX, centerY, radius,
                -(0.5)*Math.PI,
                (-(0.496)*Math.PI)+ang,
                false);
                  context.lineWidth = 7;
                  context.strokeStyle = 'green';
                context.stroke();}
              if(ang  >= (2*Math.PI)){
                  context.beginPath();
                  context.arc(
                      centerX, centerY, radius,
                      -(0.5)*Math.PI,-(0.5)* Math.PI,
                      false);
                  context.lineWidth = 7;
                  context.strokeStyle = 'green';
                  context.stroke();};
            listener.unsubscribe();  });};
```

Código 3.5: Código para modificações dos parâmetros do servidor dinâmico do ROS.

```
function somaupdateitem(label,parametro,channel,valor){
    valor++;
    var dynaRecClient = new ROSLIB.Service({
    ros : ros,
    name : '/ema/server/set_parameters',
    serviceType : 'dynamic_reconfigure/Reconfigure'
    });
    var request = new ROSLIB.ServiceRequest({
    config: {bools: [{}],ints: [{
    name: channel,value: valor}],
    strs: [],doubles: [],groups: []}
        });
    dynaRecClient.callService(request, function(result) {
    parametro.value=valor;
    label.innerHTML=valor;
        });
    }

    function menosupdateitem(
    label,parametro,channel,valor)
    {
    valor--;
    var dynaRecClient = new ROSLIB.Service({
    ros : ros,
    name : '/ema/server/set_parameters',
    serviceType:'dynamic_reconfigure/Reconfigure'
    });
    var request = new ROSLIB.ServiceRequest({
    config: {
    bools: [{}],
    ints: [{
        name: channel,
        value: valor
          }],
    strs: [],
    doubles: [],
    groups: []}
    });
    dynaRecClient.callService(request, function(result) {
    parametro.value=valor;
    label.innerHTML=valor;
    });
}
```

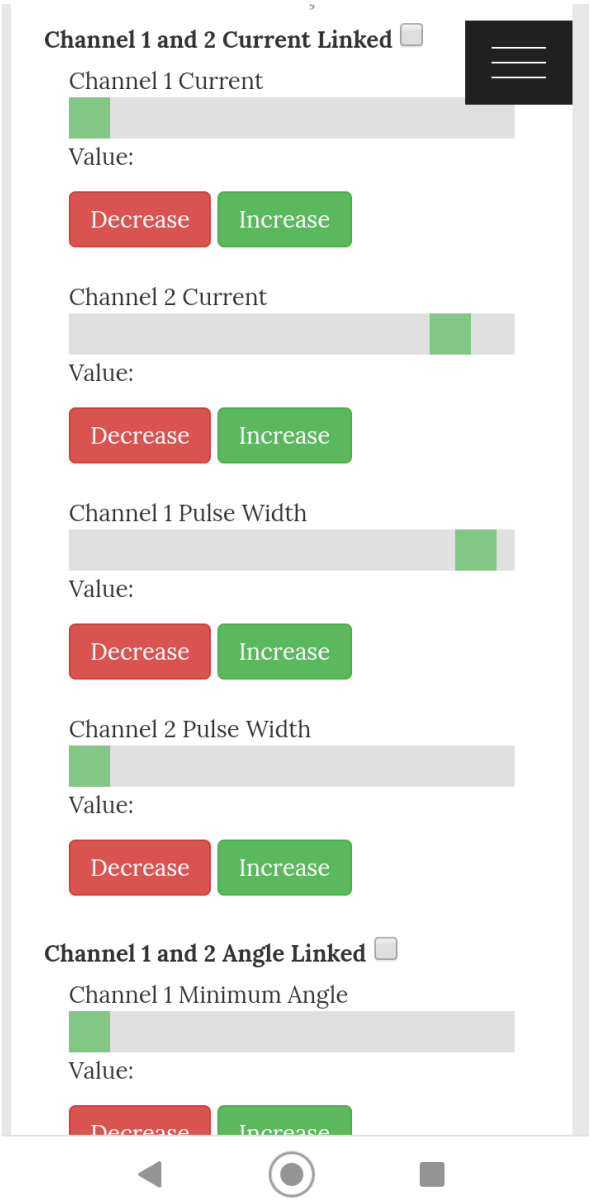


Figura 3.8: Apresentação da função do sistema de modificar os parâmetros por meio da interface em javascript sendo utilizada em um sistema android.

tuída de executáveis do Python e pip. A partir o momento da ativação do ambiente virtual, todas os pacotes instalados ficam internos a esse ambiente, sendo específico dele.

A criação de ambientes virtuais é feita executando o comando venv: python3 -m venv /path/to/new/virtual/environment. A execução desse comando cria o diretório de destino e coloca um arquivo pyvenv.cfg nele com uma chave home apontando para a instalação do Python a partir da qual o comando foi executado, também, é criado um subdiretório bin que contém uma cópia ou link para o Python, [57].

3.6 ESTUDO EXPERIMENTAL

Esse trabalho disponibilizou duas interfaces para o sistema de ciclismo por EEF. A primeira interface em ROS Qt e a segunda interface um sistema web. A primeira interface foi utilizada pelo projeto EMA pelo período de um ano, seguindo o protocolo descrito abaixo. A segunda interface, inicialmente, foi separada em módulos e foram isolados em blocos (IMU, Estimulador, Interface) para garantir que cada parte do sistema estava funcionando corretamente.

Após a comprovação do funcionamento correto dos blocos, o sistema foi validado de forma completa. Então, foi realizado um experimento como prova de conceito em um paciente com lesão medular. A validação seguiu o protocolo descrito abaixo e ocorreu uma vez. A solução do sistema web foi utilizada sem prévia descrição para o usuário (tanto pelo paciente quanto pelo aplicador da técnica de reabilitação).

3.6.1 Sujeitos

Esse trabalho tem a aprovação pelo comité de ética (CAAE 50337215.1.0000.0030, número de aprovação 1.413.934, 18 de fevereiro de 2016) e o consentimento do voluntário. O voluntário é um atleta ativo, competindo em vela e remo, de sexo masculino, 41 anos, 69 kg, com lesão medular de nível T9 há 7 anos. Antes do experimento, descrito aqui, o indivíduo já estava acostumado a treinamentos utilizando o ciclismo por EEF. O voluntário passou por 6 meses de monitoramento, incluindo testes preliminares, pré-treinamento e fases de treinamento, onde se acostumou às terapias de estimulação elétrica.

3.6.2 Protocolo

A periodicidade dos treinamentos da interface em ROS Qt era semanal. Para a interface web, foi realizado um experimento piloto com o paciente com lesão medular. Em ambas as soluções os exercícios duravam 40 minutos e nesses treinos foi testado o sistema modificando os valores de forma dinâmica.

Inicialmente era realizado um aquecimento com duração de quinze minutos, ainda com movimentos guiados pela pessoa responsável pelo exercício. Após o aquecimento iniciava o incremento da corrente em pequenos valores até ser possível a movimentação do voluntário sem a necessidade de auxílio manual. Mesmo após adquirida a estabilidade na movimentação é necessária a supervisão atenciosa do exercício, pois paradas abruptas podem acontecer

devido a fadiga muscular.

Para diminuir o número de paradas abruptas, o sistema pode ser colocado em malha fechada através da interface gráfica. O controle é realizado baseado na velocidade. Um valor limite mínimo de velocidade é definido e a medida que ocorre um decréscimo na velocidade até atingir esse valor o sistema incrementa a corrente com valores pré definidos, isso leva a um aumento de fibras musculares selecionadas e propicia a manutenção da movimentação. A intensidade e duração de cada sessão de treinamento foi definida de acordo com o programa de treinamento do participante.

3.7 AVALIAÇÃO DE DESEMPENHO

Três métricas para avaliar o desempenho de aplicações web são utilizadas geralmente: O tempo gasto para o primeiro aparecimento de conteúdo; O tempo que leva para o conteúdo de uma página ser preenchido visivelmente; Tempo até a página ficar interativa. Apesar dessas métricas trazerem informações importantes do sistema, foi observado que elas não seriam parâmetros interessantes para análise de desempenho da posposta de interface web desse trabalho. Pois a aplicação iria ser executada em uma rede local, sem acesso à internet. Para a análise de desempenho desse sistema houve uma segregação de contextos da aplicação. Essa metodologia foi escolhida para possibilitar análise por partes. Essas partes foram: comunicação por meio da rede wifi, recursos computacionais da Raspberry e a capacidade do servidor websocket. Isso permitiu uma apreciação mais criteriosa das possibilidades que poderiam causar depreciação de desempenho.

Inicialmente foi utilizada a ferramenta Google Chrome Audits. Essa ferramenta é disponibilizada pelo navegador Google Chrome como ferramentas para desenvolvedores [58] e busca identificar em um sistema web fatores que possam ser melhorados. Essa ferramenta fez uma análise macro da solução web implementada e demonstrou ser uma boa ferramenta para adequação do sistema as boas práticas de programação, mas não foi possível extrair informações específicas sobre os fatores influenciadores de desempenho dessa interface. Essa falta de aplicabilidade é devido a essa interface manter a comunicação via rede Wifi com um servidor local websocket.

A posteriori, foram utilizadas outras funções dessas ferramentas para desenvolvedor do navegador Google Chrome, a função de desempenho e a função rede. A função de desempenho fornece dados para análise dos processos que estão sendo executados pelo sistema web. A função rede fornece os dados de comunicação do sistema web pela rede, com esses dados

é possível identificar o que afetaria a comunicação e o sistema de forma mais acentuada. As ferramentas de desenvolvimento web do Mozilla Firefox [59] também foram utilizadas como método de comparar as informações das ferramentas do Google Chrome para realizar a análise dos dados. As funções de análise do sistema web utilizadas do navegador Mozilla Firefox foram as mesmas do Google Chrome, rede e desempenho.

Além da comunicação e dos processos do sistema web, o servidor websocket executado na Raspberry também foi analisado. Essa análise foi realizada por meio de um teste de carga. A ferramenta utilizada foi a Apache Jmeter [60]. Os testes simularam uma alta taxa de dados e de tráfego e registram como o sistema responde nestas condições. Por fim, foi realizada a avaliação de desempenho dos recursos da Raspberry (uso do processamento, uso da memoria, uso do potencial de comunicação) por meio do Gnome System Monitor. Os dados de esforço computacional eram observados e analisados a partir de requisições da interface web. A Figura 3.9 é a representação gráfica em forma de tabela das ferramentas citadas acima e suas aplicações.

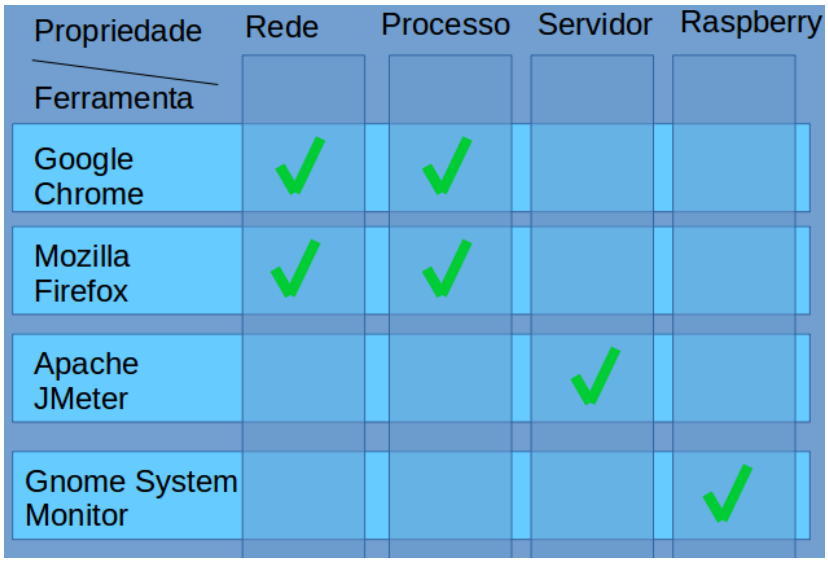


Figura 3.9: As ferramentas utilizadas para avaliar o desempenho do sistema estão dispostas na primeira coluna. As propriedades estão apresentadas de forma resumida na primeira linha. As marcações em verde representam quais propriedades essas ferramentas estão aferindo.

4 RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados os resultados do sistema. Os resultados de desempenho do sistema para a comunicação e demanda de processos foram encontrados por meio das ferramentas dos navegadores web Google Chrome e Mozilla Firefox. Também foi utilizada a ferramenta Apache Jmeter para análise do servidor websocket e o Gnome System Monitor para a verificação de utilização dos recursos computacionais da Raspberry. Na segunda parte desde capítulo foram apresentados os resultados da utilização do sistema em um prova de conceito. Como etapa inicial da validação do sistema, os resultados foram adquiridos a partir dos sistemas isolados. Após os testes dos módulos, o resultado do sistema completo foi adquirido. O teste para o paciente com lesão medular buscou identificar se todas as funcionalidades da solução foram executadas corretamente e os fatores que poderiam influenciar na sua utilização. Por fim, são apresentadas as discussões da solução desenvolvida. Todo o projeto está disponibilizado em um repositório no GitHub [1]. Nesse repositório estão disponibilizados os códigos da interface na pasta webpages e nas demais pastas os códigos para os nós em ROS.

4.1 VALIDAÇÃO DO SISTEMA

Nessa seção são apresentados os resultados da utilização experimental do sistema web. São mostrados os testes de validação do sistema por módulos e os resultados da utilização desse sistema completo.

4.1.1 Testes dos módulos do sistema

O sistema web que se comunica com os dispositivos utilizando ROS foi analisado inicialmente por módulos. Durante a utilização do sistema, os blocos da IMU e estimulador foram inspecionados e foi verificado se os resultados foram obtidos conforme o esperado.

[1] https://github.com/glricarte/EMA_FES_GUI.git

4.1.1.1 Módulo IMU

A figura 4.1 apresenta a primeira etapa dessa inspeção. Os testes iniciais foram realizados com o bloco da IMU. Os dados do sensor foram apresentados na interface em JavaScript por de uma representação gráfica dinâmica em forma de círculo como mostra na figura 4.1 e através console do navegador. Os valores apresentados foram comparados entre o console e o tópico que publica os ângulos da IMU no ROS. Essa comparação foi realizada mantendo a IMU em determinadas posições e averiguando se os valores seriam iguais. Os resultados do bloco da IMU para a aquisição dos ângulos foi satisfatório..

A velocidade angular do pedivela era extraída das informações da IMU. Para a aquisição desses valores foi necessário reprogramar as configurações iniciais da IMU, pois a versão anterior do sistema não permitia a aquisição desses dados. Essa reprogramação foi realizada a no arquivo YAML e no nó da IMU. Após essas modificações, os resultados apresentados no console do navegador eram iguais aos apresentados no tópico da IMU responsável por publicar o valor da velocidade angular.

4.1.1.2 Módulo estimulador

Os testes para o módulo do estimulador foram realizados alterando o valor da corrente de estimulação por meio da interface web e verificando se a atualização dos valores ocorria tanto na interface quanto no servidor dinâmico do ROS. Todas as mudanças realizadas foram visualizadas no servidor dinâmico.

4.1.2 Resultados do sistema completo

Essa etapa de aquisição de resultados buscou identificar se o sistema estava operando conforme o esperado, permitindo a alteração dos parâmetros em tempo de execução por meio de uma interface web portátil. Foi avaliado, também, se o sistema estava seguro para a próxima etapa de experimento com o participante com lesão medular. Essa análise foi realizada verificando se as mudanças nos parâmetros não apresentavam atraso que pudessem comprometer a utilização do sistema. Os resultados obtidos nessa fase não apresentaram comportamentos anômalos e o sistema funcionou sem apresentar atrasos que pudessem interferir com a segurança do sistema.

A Figura 4.2 apresenta a configuração para a utilização do sistema de ciclismo por EEF com o sistema web. Ela é composta por dois canais do estimulador, o sensor inercial anexado ao pedivela, o celular que mantêm a comunicação com a Raspberry via Wifi e um monitor

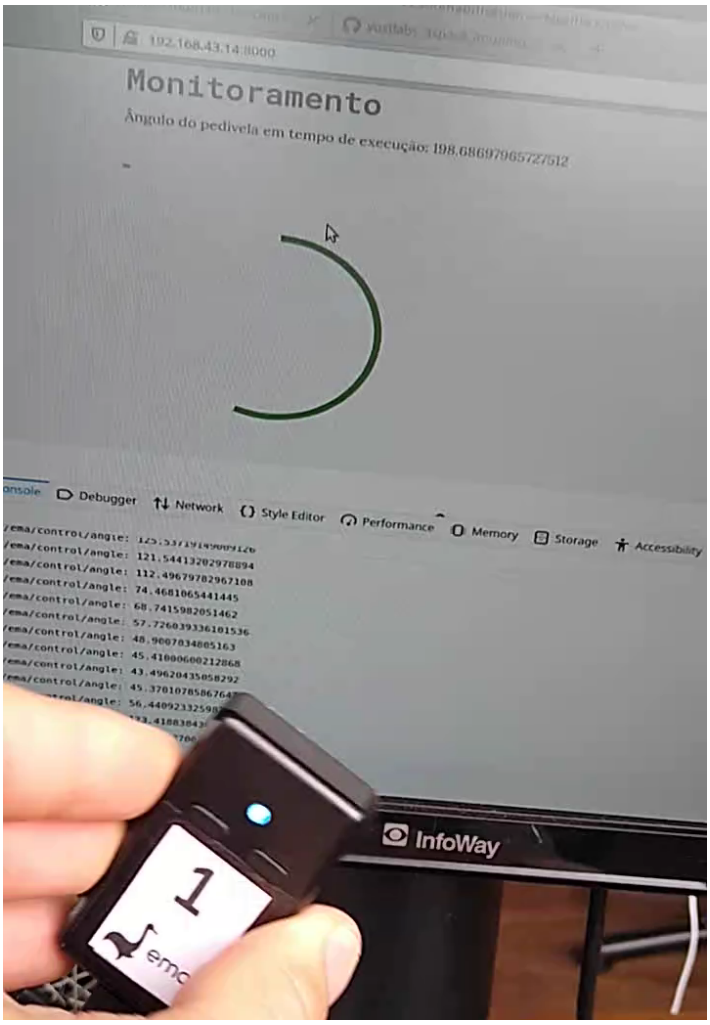


Figura 4.1: Teste realizado com os sistemas isolados. Na imagem é apresentado a IMU e seus respectivos valores no console do navegador e sua posição graficamente representada.

conectado a Raspberry.

Inicialmente foi realizada a conexão utilizando o IP da Raspberry, essa conexão é o handshake do webserver com o websocket. A Figura 4.3 apresenta a tela desse processo. A conexão ocorreu sempre que foi solicitada e não ocorram falhas no processo.

Após a conexão, na seção parâmetros da página do navegador, o participante realizada a etapa do PRESET. A Figura 4.4 apresenta a tela para a etapa do PRESET. Nessa etapa os canais de estimulação podem ser ativos ou desativados, o controle em malha fechado pode ser selecionado e seus parâmetros definidos. Na tela apresentada os valores foram atualizados a partir do servidor dinâmico do ROS, essa atualização aconteceu após a conexão realizada na página da Figura 4.3 .

A etapa de PRESET pode ser realizada a qualquer momento durante o uso do sistema

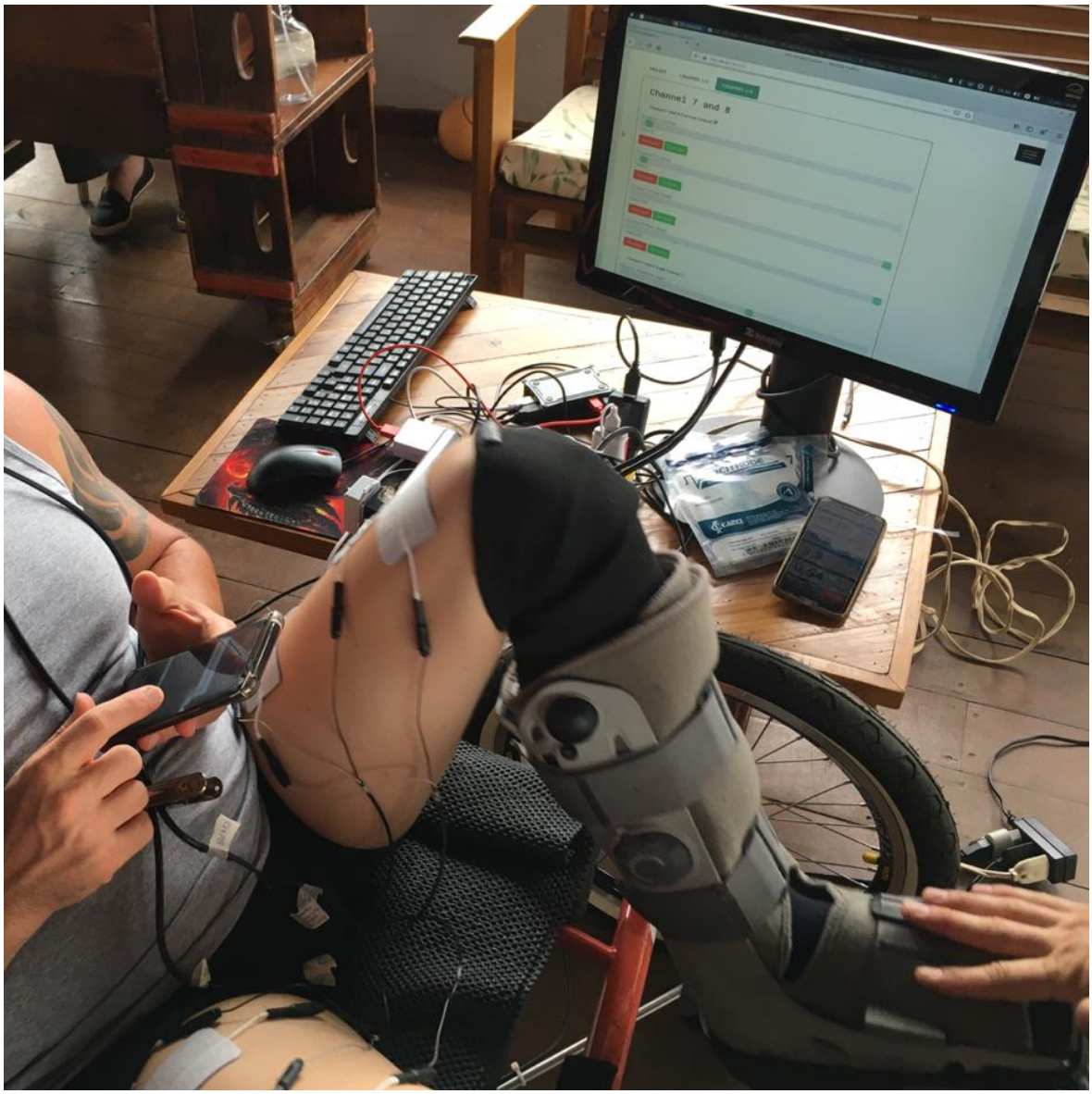


Figura 4.2: São apresentados dois canais do estimulador, o sensor inercial anexado ao pedivela, o celular do paciente e um monitor conectado a Raspberry.

e foi utilizada, por exemplo, para desativar o canal do estimulador no fim do experimento. A Figura 4.5 apresenta a janela de edição dos parâmetros de estimulação relacionados aos canais 1 e 2. A mudança entre as telas é realizada por abas. As abas se tornam visíveis a partir da seleção dos canais no PRESET. As modificações dos parâmetros para os demais canais seguem a mesma estrutura que apresentado na Figura 4.5.

Figura 4.3: Janela que realiza a conexão com o websocket, nessa janela é apresentado o botão para conectar ou desconectar e o status da conexão com o servidor.

Figura 4.4: Janela do PRESET, ativa ou desativa os canais de estimulação, habilita ou desabilita o controle em malha fechada para a compensação da fadiga.

Figura 4.5: Nessa janela são apresentados os botões para o incremento ou decremento dos valores dos parâmetros. A modificação pelos sliders foi desabilitada por segurança.

4.2 RESULTADOS DE DESEMPENHO

Os resultados apresentados nessa seção foram utilizados para identificar os fatores que influenciavam o desempenho do sistema.

4.2.1 Tempo de execução dos processos

Por meio da ferramenta para desenvolvedores do navegador Mozilla Firefox a demanda de tempo dos processos do sistema web no navegador foi aferida. A Figura 4.6 apresenta em ordem decrescente de demanda de tempo os processos do sistema web. Utilizando tal ferramenta foi observado que os processos de apresentação dos gráficos e sua atualização era o que demandava mais tempo do sistema. A atualização dos dois gráficos era vinculada a comunicação do webserver com o servidor websocket.

4.2.2 Tempo demandado na comunicação

A comunicação do webserver e do websocket era realizada por meio de uma rede Wifi. Para analisar essa comunicação foram utilizadas as ferramentas para desenvolvedores do navegador Google Chrome e Mozilla Firefox na função rede. A figura 4.7 (a) apresenta a demanda de tempo de comunicação utilizando as ferramentas para desenvolvedores do Google Chrome. A figura 4.7 (b) apresenta a demanda de tempo de comunicação utilizando as ferramentas para desenvolvedores do Mozilla Firefox. Em ambas ferramentas é possível notar que os gráficos de monitoramento continuam sendo os elementos que demandam maior consumo de tempo do sistema. Com a análise desses resultados foi possível notar que a maior parte desse consumo era ocasionada por esperas na comunicação.

4.2.3 Capacidade de comunicação do servidor websocket

O teste de carga foi realizado para verificar se o volume de transações, acessos simultâneos ou usuários que o servidor suporta é satisfatório. Esse teste foi motivado para dar continuidade na análise de desempenho do sistema e identificar o motivo que o processo de apresentação e atualização dos gráficos de monitoramento da interface web portátil demanda mais tempo de comunicação com o servidor websocket que os outros processos do sistema. A ferramenta Apache Jmeter, que simula usuários se conectando ao servidor através de requisições HTTP, foi utilizada para investigar a capacidade do servidor websocket rosbridge. A figura 4.7 mostra o resultado do teste de carga do servidor utilizando 1000 requisições.

Esse número foi escolhido pois é um numero de clientes muito maior que no pior dos casos de utilização do sistema e demonstra que a capacidade do servidor atende a demanda da solução. O objetivo do teste não foi exaurir o servidor websocket, mas observar que o servidor rosbridge satisfaz as necessidades do sistema.

4.2.4 Capacidade dos recursos computacionais da Raspberry

Para averiguar se o uso de *frameworks* afetou a capacidade da Raspberry, foi utilizada a ferramenta Gnome System Monitor. Essa ferramenta permite analisar o processamento, a memoria e a capacidade de comunicação da Raspberry. A Figura 4.9 apresenta a demanda dos recursos computacionais da Raspberry pi 4 durante a utilização do sistema web portátil para ciclismo por EEF. Os testes foram realizadas modificando os parâmetros e executando o sistema por meio da interface em JavaScript. Não foi observado uma mudança na demanda de recursos computacionais da Raspberry, após o início do sistema para ciclismo por EEF portátil, mesmo modificando as variáveis no servidor dinâmico ou apresentando os dados de monitoração por meio dos gráficos.

4.2.5 Discussão dos resultados de desempenho

As requisições da interface portátil do ciclismo por EEF precisam ser traduzidas pelo servidor websocket para o sistema ROS. Essa comunicação apenas é estabelecida quando a requisição anterior tenha acabado o seu ciclo de comunicação (requisição e resposta). Os gráficos de velocidade e de posicionamento do pedivela são atualizados a cada 100ms, essa atualização gera uma necessidade de resposta do servidor websocket e pode gerar um impossibilidade de comunicação do servidor naquele intervalo. Essa requisição que aguardará a disponibilidade do servidor pode ser um o clique do botão para aumentar a corrente, ou diminuir a corrente, ou alterar os ângulos de acionamento fazendo com que essa nova interação do usuário fique armazenada no *buffer* do navegador utilizado.

A demanda do processo de apresentação e atualização dos gráficos de monitoramento no desempenho do sistema web apresentado na Figura 4.6 e a demanda da comunicação desse processo apresentada na Figura 4.7 são explicadas pelo fato do JavaScript apenas executar um trecho de código por vez, ou seja, operar com thread único. Outro fator associado a essas demandas é a comunicação com o servidor websocket, essa comunicação pode influenciar e bloquear novas requisições assíncronas desse sistema.

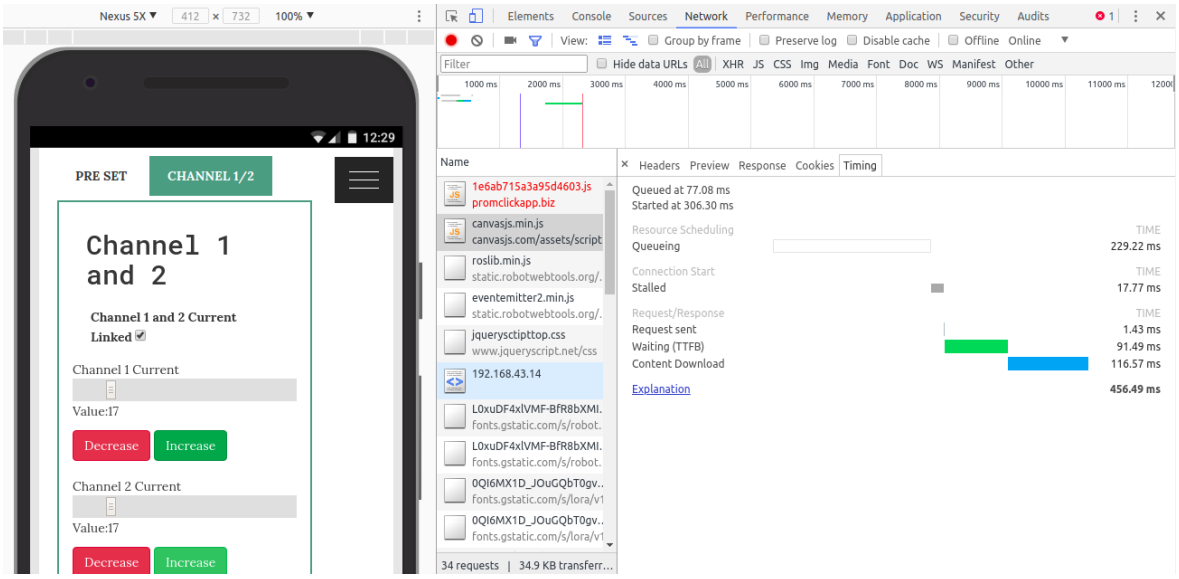
Quando um botão é pressionado (evento assíncrono), a informação é armazenada no

buffer para ser executado posteriormente, mas se o buffer for sobrecarregado o evento assíncrono irá ser perdido. Essa possibilidade de falha de comunicação pode ser diminuída pelo navegador web. O navegador web permite o armazenamento das requisições a partir de um buffer, mas uma vez que forem realizadas várias requisições sem a resposta adequada, esse buffer substitui as requisições antigas pelas novas e começa a ocorrer perda de informações. A atualização do gráfico de velocidade angular e atualização da posição do pedivela faz com que todo o sistema necessita aguardar a resposta do servidor rosbridge a essas requisições de atualização.

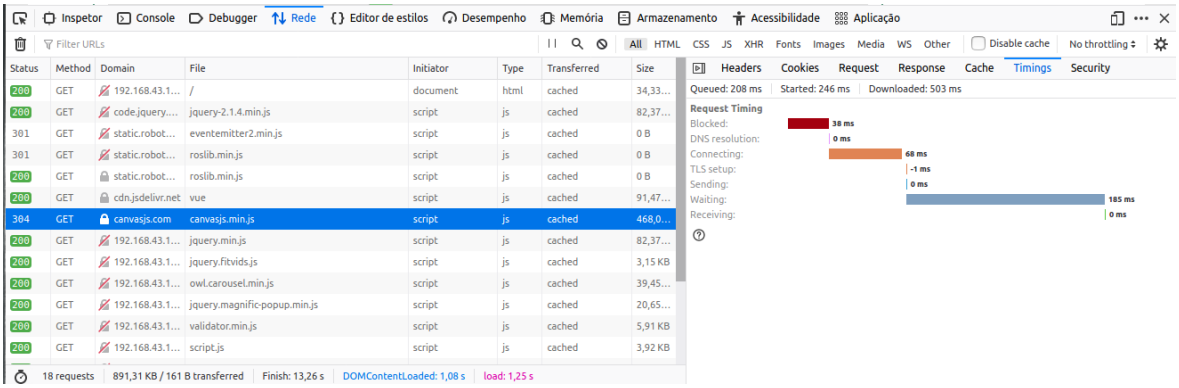
Durante o uso do sistema, não foram caracterizadas falhas que pudessem comprometer a sua utilização, mas foi observado uma lentidão quando foi necessário modificar o ângulo de ativação da estimulação pelo botão, devido o excesso de requisições em um curto intervalo de tempo, mas sem afetar a segurança do participante.

Por meio desses resultados foi possível observar que os recursos utilizados para a concepção dessa solução foram suficientes para possibilitar uma solução modular, portátil e funcional para o ciclismo por EEF, pois a demora apresentada pelo sistema não afetou a sua utilização. A reusabilidade dos códigos e a modularidade do sistema é uma característica do *middleware* ROS, essa solução manteve essa característica fazendo com que outras modalidades de exercícios utilizando EEF pudessem utilizar recursos dessa solução sem grande esforço computacional. A utilização de *frameworks* não demonstrou impactar na demanda de recursos computacionais da Raspberry, então outros *frameworks* poderiam ser utilizadas para criar melhores interfaces e melhorar a comunicação com o servidor websocket.

Figura 4.6: A figura apresenta relatórios de desempenho do sistema utilizando a ferramenta de desenvolvimento web do navegador Mozilla Firefox. Esses processos são apresentados em ordem decrescente de demanda de tempo do navegador. O primeiro processo apresentado na lista é o processo responsável por criar e atualizar os gráficos da velocidade angular e da posição do pedivela do ciclista.



(a) Relatório de demanda da rede utilizando as ferramentas do desenvolvedor do Google Chrome



(b) Relatório de demanda da rede utilizando as ferramentas de desenvolvimento do Mozilla Firefox

Figura 4.7: Apresentação da demanda de tempo na comunicação do processo de apresentação dos gráficos de monitoramento com o servidor websocket.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/...	Sent ... ↑	Avg. Byt...
HTTP Request	1000	6086	599	16279	3525.37	0.00%	59.2/sec	53.61	10.28	928.0
TOTAL	1000	6086	599	16279	3525.37	0.00%	59.2/sec	53.61	10.28	928.0

Figura 4.8: Teste do servidor websocket rosbridge utilizando a ferramenta Apache Jmeter. Esse teste de carga foi realizado utilizando 1000 requisições.

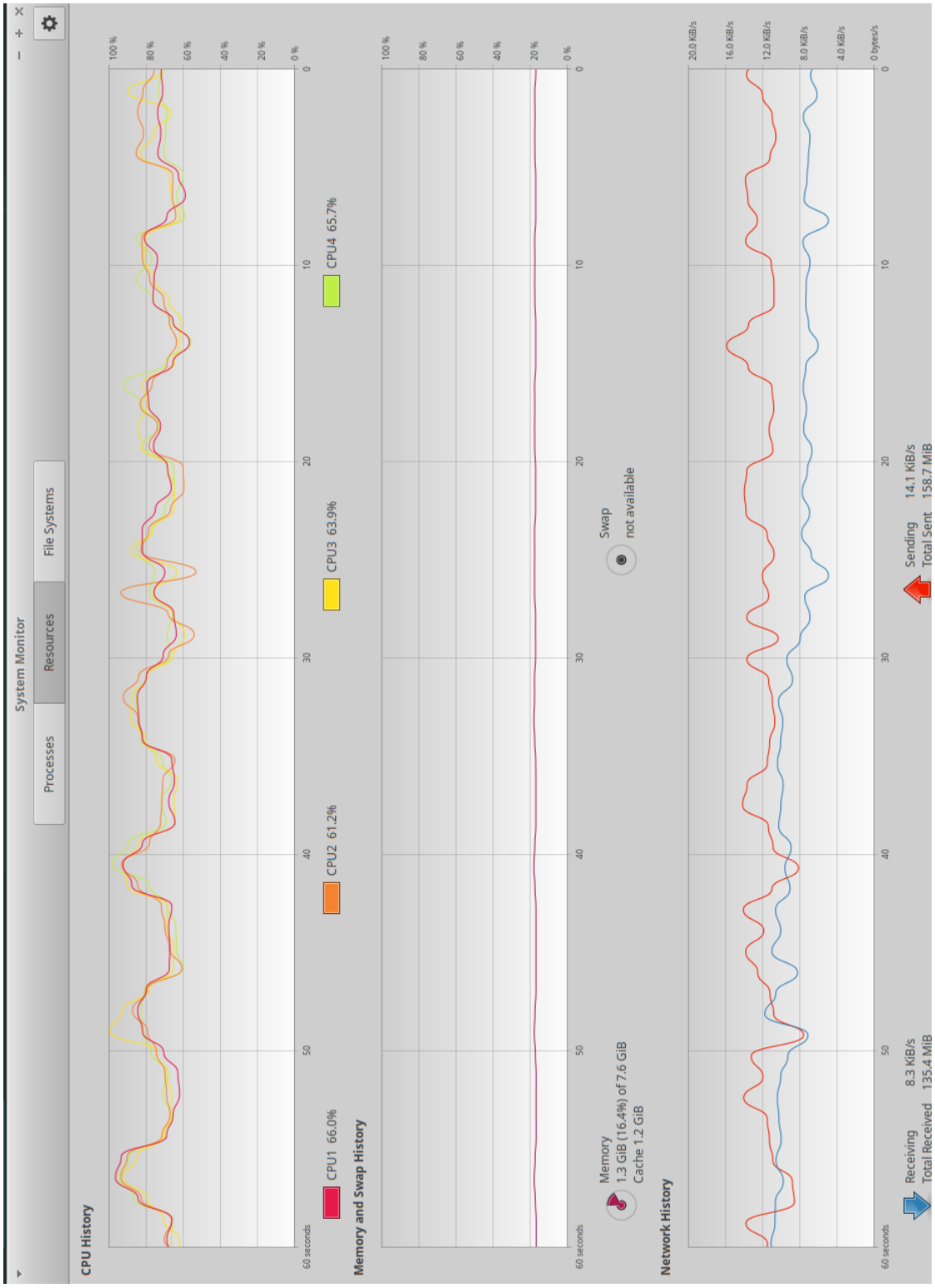


Figura 4.9: Demanda de recursos computacionais da Raspberry durante o uso do sistema utilizando a ferramenta Gnome System Monitor.

4.3 EXPERIMENTO COM PARTICIPANTE COM LESÃO MEDULAR

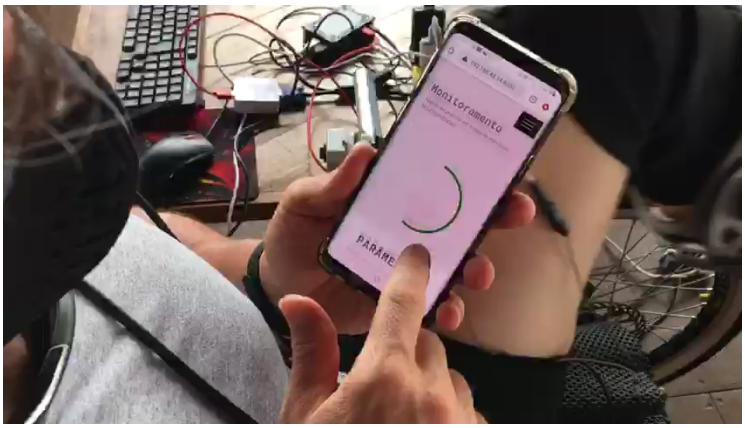
Após a junção dos blocos e seus resultados avaliados como satisfatórios, o participante com lesão medular utilizou o sistema como prova de conceito. Essa abordagem foi escolhida para identificar se o sistema atenderia a proposta de ser portátil e capaz de ser utilizado por diferentes usuários.

Os resultados do experimento com o participante com lesão medular foram obtidos seguindo o protocolo do estudo experimental. Começando pelo aquecimento até a movimentação sem o auxilio manual. Inicialmente os ângulos foram definidos pelo aplicador do testes através do sistema web. Após sua definição, o sinal da corrente foi aumentado por meio do sistema web e a alteração dos valores foi visualizada no servidor dinâmico do ROS. Isso permitiu identificar se as alterações realizadas na interface web modificavam o servidor dinâmico do ROS conforme esperado.

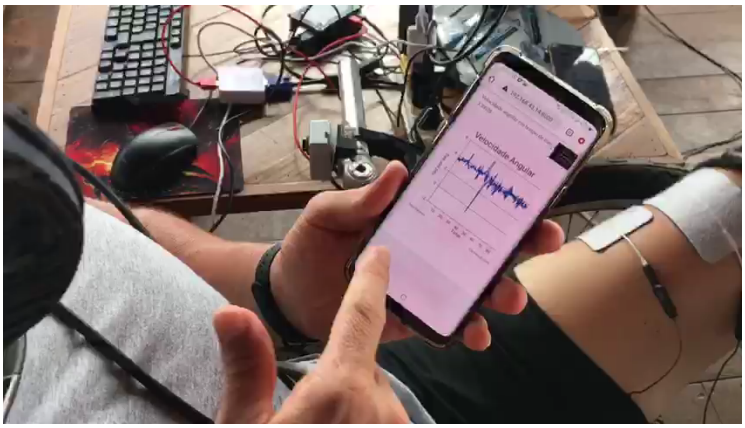
Então, o participante começou a participar de forma ativa do experimento. A Figura 4.10 mostra o participante utilizando o seu aparelho celular para controlar o sistema de ciclismo por EEF. Na Figura 4.10(a) o participante está acompanhando o posicionamento do pedivela durante a execução do movimento. Na figura 4.10(b) o participante está acompanhando a evolução da velocidade angular da pedalada. Na Figura 4.10(c) o participante está modificando os parâmetros de estimulação, adicionando corrente aos canais 7 e 8 por meio do botão da interface web.

O monitoramento da velocidade e da posição do pedivela foi realizado por meio de gráficos da interface web. A Figura 4.11 apresenta as telas de monitoramento a partir do navegador. Na Figura 4.11 (a) é apresentado o gráfico de velocidade angular com dados referentes ao experimento com duração de aproximadamente 6 minutos. Na 4.11 (b) é apresentada a continuação desse gráfico até o fim do experimento. Na 4.11 (b) o fim dos testes é representado quando o valor da velocidade angular decai para zero. Durante os testes, toda a alteração de corrente estava sendo realizada pelo participante que utilizava o seu celular.

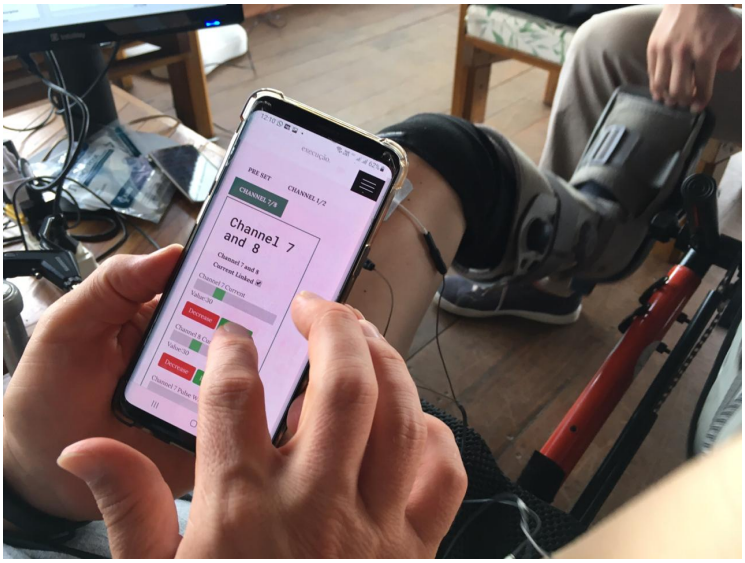
As modificações foram realizadas pela interface web durante a execução do experimento. Além do monitoramento pela interface web, a equipe também monitorou a interface em ROS Qt. A interface em ROS Qt foi amplamente utilizada nos experimentos do projeto EMA. Com a comparação dos dados entre as duas interfaces seria possível identificar qualquer anomalia e observar se os dados alterados na interface web em Javascript surtiriam efeito no servidor dinâmico e nos tópicos.



(a) Participante acompanhando o posicionamento do pedivela por meio do seu celular.

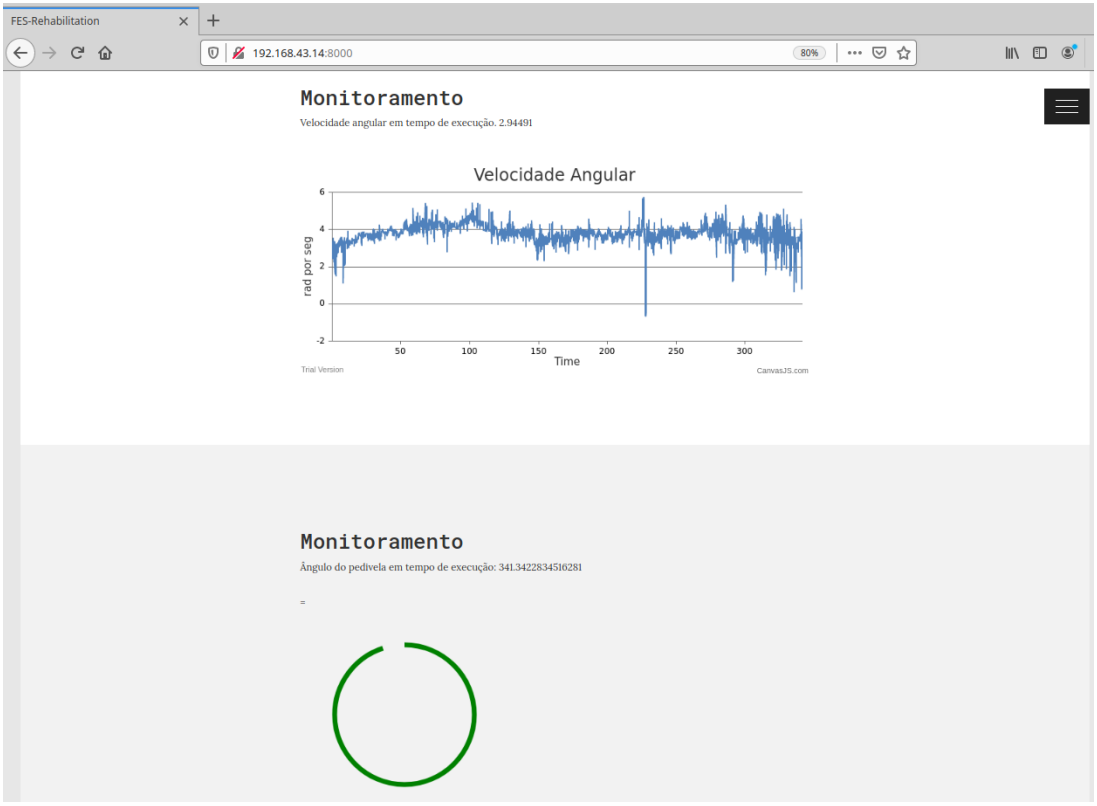


(b) Participante acompanhando pelo seu celular o gráfico atualizado dinamicamente da velocidade angular

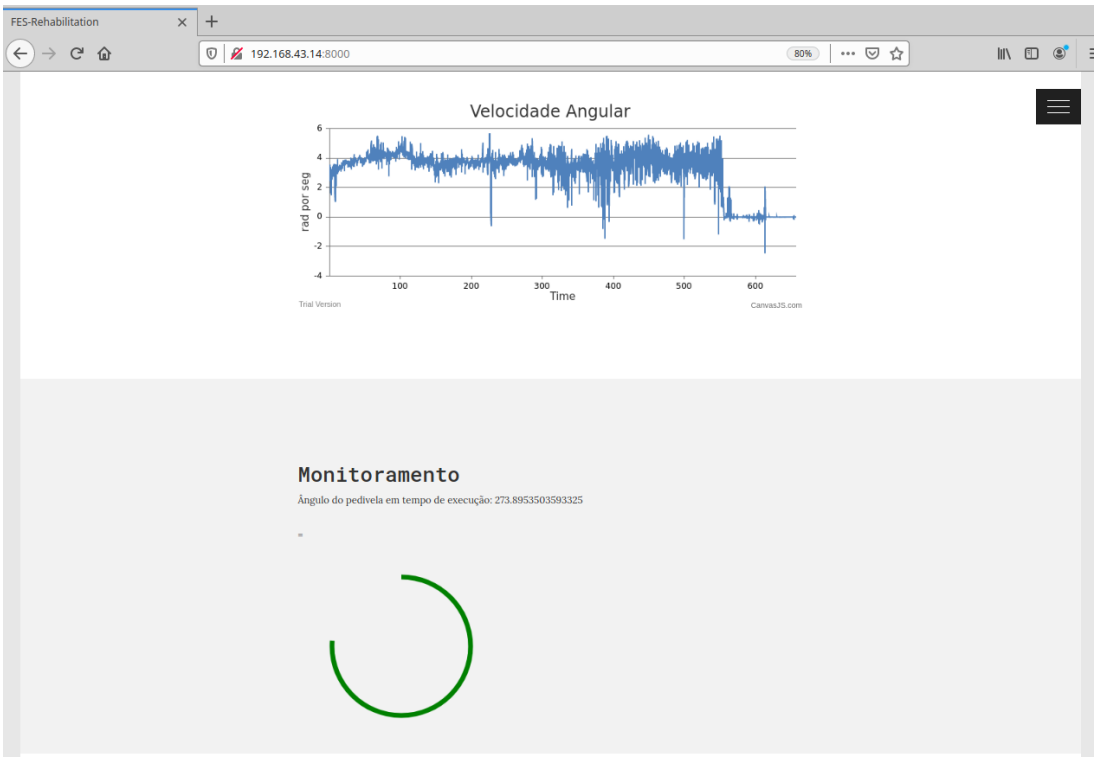


(c) Participante alterando os valores dos parâmetros da estimulação em tempo de execução da reabilitação

Figura 4.10: Participante utilizando o seu aparelho celular para controlar o sistema de ciclismo por EEF.



(a) Gráfico de velocidade angular com dados referentes ao teste com duração de aproximadamente 6 minutos.



(b) Momento que os testes se findam e a velocidade angular decai para zero

Figura 4.11: Telas de monitoramento da velocidade angular e da posição do pedivela.

4.4 DISCUSSÃO DAS SOLUÇÕES

Um dos conceitos previstos pela IEC 62304 é que o ambiente de desenvolvimento do software de dispositivos médicos também realize a gestão de qualidade e gerenciamento de risco. Dessa forma, os padrões das normas ISO 13485 e ISO 14971, nortearam a gestão de qualidade e risco. A importância de se observar essas normais e suas diretrizes para a manutenção da segurança do sistema, esta embasado na sua sensibilidade. Em sistemas aplicados a saúde que utilizam a EEF, por exemplo, a estimulação pode ser um fator de risco dada a corrente elétrica que é aplicada no paciente. Então, períodos de estimulação muito longos, valores de corrente elevados, grandes variações na intensidade, ou uma estimulação inadequada de grupos musculares podem gerar lesões no paciente. Outros fatores podem colocar a saúde do paciente em risco, como a falha no recebimento de dados dos sensores, má interpretação dos dados experimentais, corrupção dos dados, e instabilidade do sistema são exemplos da problemática em se desenvolver e disponibilizar uma interface de controle ao usuário que possibilite controle em tempo de execução da energia transferida ao paciente. Dentro do escopo deste trabalho, a classificação do risco do software acompanha a classificação do hardware, sendo isento de registro obrigatório do software, como é o caso do software embarcado, cujo registro deve ser incluído na solicitação de registro do equipamento médico ao qual se destina.

Apesar dos esforços em desenvolver uma interface em ROS Qt, foi observado por diferentes usuários, que as informações apresentadas na GUI não disponibilizavam recursos gráficos suficientes para uma boa interpretação do ciclismo assistido por EEF para um profissional da saúde ou um ciclista com lesão medular sem conhecimento técnico. A solução desenvolvida em Javascript (interface Web) permitiu a disponibilidade de mais recursos gráficos e conforme relato dos usuários, a interface tornou-se mais interpretativa para usuários com diferentes níveis de conhecimento técnico. Embora, sejam necessários mais profissionais da saúde e ciclistas com LM utilizando o sistema web para identificar e quantizar a usabilidade desse sistema.

A interface em JavaScript é apresentada como uma página web que tem os seus três módulos acessíveis através do rolamento da janela. Esses três módulos são: conexão, monitoramento e parâmetros. Foi sugerido pelos usuários desse sistema que esses módulos fossem transformados em dois. O Primeiro módulo seria o de parâmetros e o segundo módulo o de monitoramento. Os dois módulos estariam em páginas distintas e o módulo de conexão não se tornaria mais necessário. A conexão com a Raspberry seria realizada automaticamente por meio de um IP fixo. Essa conexão se efetuaria ao se iniciar o sistema.

No módulo de parâmetros, foi identificado durante o uso do sistema, que a escolha das mesmas cores dos botões para parâmetros diferentes e a falta de divisão entre eles, podem levar o usuário a alterar um parâmetro indesejado, por confusão visual. Esse erro pode ocasionar um mau desenvolvimento da reabilitação, ou até causar lesões, caso a alteração do parâmetro seja a faixa angular de ativação do canal. Outra dificuldade encontrada foi na utilização do sistema web por diversos dispositivos simultaneamente, uma vez que a atualização dos dados do servidor dinâmico do ROS apenas ocorre no momento da conexão, a alteração realizada por um dispositivo não é simultaneamente observada pelo outro dispositivo, sendo necessário reconectar para visualizar a modificação.

No módulo de monitoramento, os dados da IMU foram apresentados sem tratamento da informação. Dessa forma é possível notar grandes variações nos gráficos da Figura 4.11. Essas variações são causadas por pequenos travamentos na movimentação do pedivela e não refletem a dinâmica da execução. Seria interessante o tratamento dos dados antes da apresentação, pois diminuiriam essas variações pontuais e tornaria o gráfico mais didático. Também foi sugestionado pelos usuários que a velocidade angular fosse apresentada em graus por segundo, em vez de rad por segundo, pois tornaria a compreensão mais intuitiva.

Outro entrave desse módulo é a falta de armazenamento dos dados dos gráficos. Essa característica é observada quando a página é recarregada e todos os valores anteriores são apagados, iniciando um novo gráfico com os novos valores obtidos. Para transpor essa barreira, seria necessário pesquisar outros *frameworks* que permitissem gravar e exportar os dados do gráfico com a variável de tempo associada ao dado, assim seria possível manter o histórico do treino.

Utilizando o seu celular os participantes controlaram todo o sistema, aumentando sua interação com o exercício. Porém, conforme alegado pelos participantes, a interface atual não atenderia as necessidades na utilização do sistema em ambiente externo de forma móvel. Essa incapacidade estaria relacionada a ocupação das mãos guiando a Trike e a impossibilidade do manuseio do celular para a alteração dos parâmetros. Ressalta-se que para os participantes a parte de monitoramento poderia ser utilizado como display e o celular poderia ser fixado em um suporte no guidão.

Para manter a segurança da aplicação foi desabilitada a possibilidade de modificar os valores por meio dos *sliders*, essa decisão de fato trouxe maior segurança no quesito aumento ou diminuição da corrente do estimulador, porém, trouxe morosidade na alteração inicial dos ângulos de ativação.

Durante os testes, a funcionalidade de modificação dos parâmetros em tempo de execução

para o sistema em ROS, ROS Qt e interface em JavaScript obteve êxito. Embora, tenha sido observado na interface em JavaScript pequenas latências do sistema. Essas latências estavam associadas a comunicação do webserver com o servidor websocket e eram mais comuns quando solicitações de alterações de parâmetros eram realizadas de forma repetitiva e em um curto intervalo. Embora tenha sido observado tal fato, ele não interferiu na execução do experimento para o participante com LM.

Este trabalho implementou a comunicação do ROS com uma interface portátil para um sistema de ciclismo por EEF. Não foi observado problemas de incompatibilização da aplicação Web para os dispositivos utilizados: notebook utilizando Ubuntu 18.04, celular utilizando Android e celular utilizando IOS. Esse sistema foi desenvolvido de maneira responsiva, adaptável ao tamanho da tela do dispositivo. Todas as funcionalidades do sistema em ROS Qt (RQT) foram portadas para o sistema web.

O desenvolvimento da interface em ROS Qt demanda grande esforço computacional para a criação de novas ferramentas de apresentação de dados e criação de gráficos. Essa característica associada a necessidade do ROS estar instalado no dispositivo para a sua utilização não propiciou a acessibilidade desejada para o sistema. Utilizando a tecnologia JavaScript e suas ferramentas gráficas não foi identificada a falta de acessibilidade característica da solução em ROS Qt. Com o potencial gráfico do JavaScript, HTML e CSS seria possível a implementação de novas interfaces sem grande esforço computacional. Isso possibilitaria o uso das funcionalidades do sistema desenvolvido por este trabalho em diferentes modalidades de exercícios assistidos por EEF.

No trabalho de CERONE, G. L. et al [47] um novo estimulador elétrico sem fio, modular e programável foi proposto. Esse sistema apresentou módulos específicos que não permitiria a interação com novos sensores de forma simples. Isso inibiria a intercambialidade de códigos entre grupos de pesquisa. As soluções desenvolvidas por este trabalho também proporcionou a possibilidade de intercambialidade de códigos entre grupos de pesquisa. Para isso, seria necessário que as outras equipes de pesquisa estivessem desenvolvendo um sistema de exercícios assistidos por EEF em ROS utilizando a biblioteca roslibjs ou ROS Qt. A troca e utilização dos códigos seria possível com pouca demanda computacional e poderia ser utilizada em diferentes contextos de aplicação da EEF.

Os testes demonstraram que a proposta deste trabalho utilizando o JavaScript e ROS QT possibilitou o monitoramento e controle do sistema de ciclismo utilizando EEF. A interface em JavaScript possibilitou maior flexibilidade e portabilidade ao sistema. Apesar de não ter sido implementado em diferentes contextos, apenas no ciclismo, foi observado que a adição de novos nós seria possível com baixo esforço computacional e possibilitaria a utilização de

outras configurações de exercícios utilizando EEF.

5 CONCLUSÃO E TRABALHOS FUTUROS

Esse capítulo apresenta a conclusão do trabalho com suas considerações finais. Em seguida são apresentadas algumas propostas de implementação e melhoria.

5.1 CONSIDERAÇÕES FINAIS

A literatura tem apresentado que o uso da estimulação elétrica funcional no processo de reabilitação tem sido aplicado a diversas patologias e traumas. Essa aplicação tem benefícios a saúde dos pacientes [61]. Esse trabalho apresentou que no contexto de reabilitação utilizando EEF, para pacientes acometidos com acidente vascular cerebral (AVC), 52.6% dos fisioterapeutas gostariam de aumentar a utilização dessa técnica de reabilitação. Esse estudo enfatizou que a utilização da EEF pelos fisioterapeutas ainda tem sido de baixa adesão. A complexidade técnica e falta de acessibilidade por profissionais profissionais da saúde colaboram para tal contexto. Outras aplicações da EEF tem sido em lazer, estudos clínicos e a partir do Cybathlon em competição.

No intuito de aumentar a acessibilidade dessa técnica para fisioterapeutas e obter melhores resultados no Cybathlon e no monitoramento dos experimentos utilizando EEF, este trabalho abordou aspectos de desenvolvimento de uma interface de controle portátil para ciclismo por estimulação elétrica funcional. Essa interface deve permitir a interação do usuário com o sistema em tempo de execução, sendo modular, flexível, reutilizável e portátil. Inicialmente foi realizada revisão bibliográfico da EEF para compreender o estado da arte do ciclismo assistido por EEF. Nessa etapa foram avaliados os sistemas comerciais e a literatura de sistemas aplicados a reabilitação utilizando EEF e suas interfaces.

Observou-se que embora seja parte importante no desenvolvimento do sistema, a literatura é escassa sobre a usabilidade da interface por fisioterapeutas na aplicação clínica. Nos estudos clínicos, geralmente, o objeto alvo não é o software, sendo pouco descrito. Dessa forma, a principal contribuição deste trabalho, foi a implementação de uma interface portátil para monitorar e controlar o ciclismo assistido por EEF. Essa nova solução manteve as características das soluções anteriores do projeto EMA para ciclismo por EEF de qualidade, modificabilidade, reusabilidade, verificabilidade e proteção.

Este estudo proporcionou ao usuário maior independência e interatividade com o sistema

por EEF. Essa independência e interatividade está associada a concepção de uma interface portátil com vários recursos gráficos. A utilização do ROS associado ao JavaScript permitiu a reusabilidade, modularidade e flexibilidade ao código possibilitando o desenvolvimento de novas interfaces e utilização de novos dispositivos sem grande esforço computacional.

5.2 TRABALHOS FUTUROS

Pela escolha das linguagens JavaScript, HTML e CSS é possível desenvolver diversos recursos gráficos, com baixa demanda computacional, comparado ao ROS Qt. Então são propostos como trabalhos futuros habilitar a modificação dos ângulos de ativação por meio dos sliders. Adicionar diferentes cores, conforme as especificidades da propriedade. Como exemplo, os atributos que aumentam a corrente do canal 1 devem apresentar cores diferentes dos atributos que modificam o canal 2. Criar botões que adicionem ou diminuam, na mesma função, a corrente de todos os canais habilitados. Desenhar, junto a posição angular, os valores dos ângulos mínimos e máximos para a ativação do canal.

A linguagem JavaScript também permite o uso de diferentes frameworks. Dessa forma, é proposto utilizar o Node.js [62] para auxiliar na conexão websocket, diminuindo o tempo de espera na comunicação. Essa solução utilizaria um buffer e um controle de comunicações entre o sistema web o Node.js e o rosbridge. Isso possibilitaria aumentar a velocidade de atualização dos dados dos gráficos sem prejuízo do desempenho do sistema. Essa atualização permitiria que os dados provenientes do servidor dinâmico do ROS fossem requisitados com a mesma taxa de execução que o nó de controle do ROS e permitiria o uso simultâneo de todos os dispositivos conectados.

O acesso ao sistema via internet pode ser implementado, ressaltando a necessidade de assegurar a segurança dos dados por se tratar de um sistema sensível e aplicado a saúde. Isso permitiria o monitoramento do sistema por um fisioterapeuta remotamente na utilização do ciclismo móvel por EEF no contexto de lazer.

Esse trabalho foi desenvolvido utilizando o ROS na versão 1. Durante o período de desenvolvimento desse trabalho o ROS foi lançado na versão 2. Uma nova implementação utilizando o ROS 2 poderia ser desenvolvida. Essa nova versão em ROS 2 possibilitaria o uso de bibliotecas utilizadas em ROS 1 e permitiria maior compatibilidade de sistemas operacionais tais como Ubuntu Xenial, OS X El Capitan, Windows 10. O ROS 2 também busca garantir que as consultas sejam de forma eficiente em tempo de execução e que o ciclo de vida de um nó possa ser gerenciado.

1 BRINDEIRO, G. A. Software embarcado de controle para triciclo assistido por estimulação elétrica. 2017.

2 ROS ANDROID. <http://mirror-ap.wiki.ros.org/rosjava(2f)Official(20)Dependency(20)Graph.html>. Acessado em 09/03/2020.

3 CANIUSE. <https://caniuse.com/websockets>. Acessado em 01/01/2021.

4 BRASIL. *Diretrizes de Atenção à Pessoa com Lesão. 1nd ed.* Brasília, DF, 2013.

5 MORENO-FERGUSSON, M. E.; REY, M. C. d. P. A. Cuerpo y corporalidad en la paraplejia: significado de los cambios. *Avances en Enfermería*, v. 30, n. 1, p. 82–94, 2012.

6 GONZÁLEZ, J.; PONS, J. L.; RAYA, R. *Emerging Therapies in Neurorehabilitation II*. [S.l.]: Springer International Publishing, 2016.

7 BABAMOHAMADI, H.; NEGARANDEH, R.; DEHGHAN-NAYERI, N. Coping strategies used by people with spinal cord injury: a qualitative study. *Spinal cord*, Nature Publishing Group, v. 49, n. 7, p. 832–837, 2011.

8 RIENER, R. The cybathlon promotes the development of assistive technology for people with physical disabilities. *Journal of neuroengineering and rehabilitation*, Springer, v. 13, n. 1, p. 1–4, 2016.

9 DAVIS, G. M.; HAMZAID, N. A.; FORNUSEK, C. Cardiorespiratory, metabolic, and biomechanical responses during functional electrical stimulation leg exercise: health and fitness benefits. *Artificial organs*, Wiley Online Library, v. 32, n. 8, p. 625–629, 2008.

10 DELEY, G.; DENUZILLER, J.; BABAULT, N. Functional electrical stimulation: cardiorespiratory adaptations and applications for training in paraplegia. *Sports Medicine*, Springer, v. 45, n. 1, p. 71–82, 2015.

11 MAZZOLENI, S. et al. Fes-cycling training in spinal cord injured patients. In: IEEE. *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. [S.l.], 2013. p. 5339–5341.

12 HARVEY, L. A. Physiotherapy rehabilitation for people with spinal cord injuries. *Journal of physiotherapy*, Elsevier, v. 62, n. 1, p. 4–11, 2016.

13 HUNT, K. J. et al. Control strategies for integration of electric motor assist and functional electrical stimulation in paraplegic cycling: utility for exercise testing and mobile cycling. *IEEE Transactions on Neural systems and rehabilitation engineering*, IEEE, v. 12, n. 1, p. 89–101, 2004.

14 WOODS, B. et al. Mechanomyography based closed-loop functional electrical stimulation cycling system. In: IEEE. *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*. [S.l.], 2018. p. 179–184.

15 ROUSE, C. A. et al. FES and motor assisted cycling to track power and cadence to desired voluntary bounds. *IFAC-PapersOnLine*, Elsevier, v. 51, n. 34, p. 34–39, 2019.

16 ROUSE, C. A. et al. Cadence tracking for switched fes cycling combined with voluntary pedaling and motor resistance. In: IEEE. *2018 Annual American Control Conference (ACC)*. [S.l.], 2018. p. 4558–4563.

17 RAGNARSSON, K. Functional electrical stimulation after spinal cord injury: current use, therapeutic effects and future directions. *Spinal cord*, Nature Publishing Group, v. 46, n. 4, p. 255–274, 2008.

18 MARQUEZ-CHIN, C.; POPOVIC, M. R. Functional electrical stimulation therapy for restoration of motor function after spinal cord injury and stroke: a review. *BioMedical Engineering OnLine*, Springer, v. 19, p. 1–25, 2020.

19 PECKHAM, H.; GORMAN, P. Functional electrical stimulation in the 21st century. *Topics in Spinal Cord Injury Rehabilitation*, Thomas Land Publishers Inc., v. 10, n. 2, p. 126–150, 2004.

20 MARI, K. L. da S. et al. Técnicas fisioterapêuticas utilizadas na reabilitação de pacientes com lesão medular: Estudo de revisão. *CONNECTION LINE-REVISTA ELETRÔNICA DO UNIVAG*, n. 20, 2019.

21 BAJD, T. et al. Restoration of walking in patients with incomplete spinal cord injuries by use of surface electrical stimulation—preliminary results. *Prosthetics and orthotics international*, SAGE Publications Sage UK: London, England, v. 9, n. 2, p. 109–111, 1985.

22 CARROLL, S. G. et al. Tetanic responses of electrically stimulated paralyzed muscle at varying interpulse intervals. *IEEE transactions on biomedical engineering*, IEEE, v. 36, n. 7, p. 644–653, 1989.

23 ARMSTRONG, E. L. et al. Effects of a training programme of functional electrical stimulation (fes) powered cycling, recreational cycling and goal-directed exercise training on children with cerebral palsy: a randomised controlled trial protocol. *BMJ open*, British Medical Journal Publishing Group, v. 9, n. 6, p. e024881, 2019.

24 GFÖHLER, M. Technical rebuilding of movement function using functional electrical stimulation. In: *Biomimetics–Materials, Structures and Processes*. [S.l.]: Springer, 2011. p. 219–247.

25 LAMBACH, R. L. et al. Bone changes in the lower limbs from participation in an FES rowing exercise program implemented within two years after traumatic spinal cord injury. *The Journal of Spinal Cord Medicine*, Taylor & Francis, v. 43, n. 3, p. 306–314, 2020.

26 WIESENER, C. et al. An inertial sensor-based trigger algorithm for functional electrical stimulation-assisted swimming in paraplegics. *IFAC-PapersOnLine*, Elsevier, v. 51, n. 34, p. 278–283, 2019.

27 WIESENER, C.; NIEDEGGEN, A.; SCHAUER, T. Electrotactile feedback for FES-assisted swimming. In: SPRINGER. *International Conference on NeuroRehabilitation*. [S.l.], 2018. p. 922–925.

28 BRASIL. *Manual para regularização de equipamentos médicos na Anvisa*. [S.l.]: ANVISA Brasília, 2010.

29 Gerência-Geral de Tecnologia de Produtos para a Saúde (GGTPS) and Gerência de Tecnologia em Equipamentos (GQUIP), "Guia orientativo às empresas do setor de produtos para saúde para o software para a saúde," Agência Nacional de Vigilância Sanitária (Anvisa), Nota Técnica nº 04/2012, Mar. 2012.

30 T. da Silva Bonfim, "Regulação de Software no Brasil," Slides apresentados no I Seminário ANVISA de Dispositivos Médicos, Oct. 2016.

31 SISTEMA Operacional. <http://www.inf.ufsc.br/~j.barreto/cca/sisop/sisoperac.html>. Acessado em 15/07/2020.

32 BOSCH, J. *Object-oriented frameworks: Problems & experiences*. 1997.

33 FRAMEWORK. <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acessado em 18/11/2020.

34 AL-JAROODI, J.; MOHAMED, N. Middleware is still everywhere!!! CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE, 2012.

35 YARP. <https://www.yarp.it/git-master/what_is_yarp.html>. Acessado em 09/03/2020.

36 ROCK. <https://www.rock-robotics.org/documentation/about/index.html>. Acessado em 09/03/2020.

37 FENG, H.-M. et al. Humanoid robot field-programmable gate array hardware and robot-operating-system-based software machine co-design. *Sensors and Materials*, v. 31, n. 6, p. 1893–1904, 2019.

38 COUSINS, S. Ros on the pr2 [ros topics]. *IEEE Robotics & Automation Magazine*, IEEE, v. 17, n. 3, p. 23–25, 2010.

39 ESTEFO, P. et al. The robot operating system: Package reuse and community dynamics. *Journal of Systems and Software*, Elsevier, v. 151, p. 226–242, 2019.

40 ROS basic. <https://www.rosroboticslearning.com/basics-of-ros>. Acessado em 09/03/2020.

41 THE rfc6455. <https://tools.ietf.org/html/rfc6455#page-27>. Acessado em 01/01/2021.

42 ECMA. <https://www.ecma-international.org/memento/history.htm>. Acessado em 01/01/2021.

43 RESTORATIVE Terapies. <https://restorative-therapies.com/>, NOTE = "acessado em 24/11/2020",.

44 GUIRAUD, D. et al. Wireless electrical stimulators and sensors network for closed loop control in rehabilitation. *Frontiers in Neuroscience*, Frontiers, v. 14, p. 117, 2020.

45 BO, A. P. et al. Cycling with spinal cord injury: A novel system for cycling using electrical stimulation for individuals with paraplegia, and preparation for cybathlon 2016. *IEEE Robotics & Automation Magazine*, IEEE, v. 24, n. 4, p. 58–65, 2017.

46 JASUJA, A. et al. Development of an extensible, wireless framework for personalized muscle rehabilitation. In: IEEE. *2019 IEEE Healthcare Innovations and Point of Care Technologies,(HI-POCT)*. [S.l.], 2019. p. 17–20.

47 CERONE, G. L. et al. Design of a wireless, modular and programmable neuromuscular electrical stimulator. In: IEEE. *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. [S.l.], 2019. p. 3815–3818.

48 COSTE, C. A.; WOLF, P. FES-cycling at cybathlon 2016: Overview on teams and results. *Artificial organs*, v. 42, n. 3, p. 336–341, 2018.

49 BOULMALF, M. et al. A lightweight middleware for an e-health wsn based system using android technology. In: IEEE. *2012 International Conference on Multimedia Computing and Systems*. [S.l.], 2012. p. 551–556.

50 BERALDO, G. et al. Ros-health: An open-source framework for neurorobotics. In: IEEE. *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. [S.l.], 2018. p. 174–179.

51 BO, A. P. et al. Experimental results and design considerations for fes-assisted transfer for people with spinal cord injury. In: SPRINGER. *International Conference on NeuroRehabilitation*. [S.l.], 2018. p. 939–943.

52 FONSECA, L. et al. Cadence tracking and disturbance rejection in FES cycling for paraplegic subjects: a case study. *Eur J Transl Myol*, 2016.

53 BÓ, A. P. L. et al. Experiments on lower limb FES control for cycling. *SBAI 2015*, 2015.

54 ROS Qt. <http://wiki.ros.org/qt_create/Tutorials/Qt%20App%20Templates>. Acessado em 02/11/2018.

55 RICARTE, G.; PINHEIRO, L. de M.; BÓ, A. P. L. Intuitive and modular software architecture for functional electrical stimulation rehabilitation. In: IEEE. *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. [S.l.], 2020. p. 1–6.

56 ROSLIBJS. <http://robotwebtools.org/>. Acessado em 02/01/2021.

57 VENV. <https://docs.python.org/pt-br/3/library/venv.html>. Acessado em 15/07/2020.

58 FERRAMENTA para desenvolvedores Google Chrome. <https://support.google.com/campaignmanager/answer/2828688?hl=pt-BR>. Acessado em 02/01/2021.

59 FERRAMENTA para desenvolvedores Mozilla Firefox. <https://developer.mozilla.org/pt-BR/docs/Tools/Web_Console/Opening_the_Web_Console>. Acessado em 02/01/2021.

60 APACHE JMeter. <https://jmeter.apache.org/>. Acessado em 20/10/2020.

61 ISLAM, A. et al. Effect of functional electrical stimulation on gait restoration after stroke-a review. *International Journal of Neurologic Physical Therapy*, Science Publishing Group, v. 5, n. 2, p. 63, 2019.

62 ROSNODEJS. <http://wiki.ros.org/rosnodejs>. Acessado em 0/10/2020.