UFRR

UNIVERSIDADE FEDERAL DE RORAIMA

PRÓ-REITORIA DE ENSINO E EXTENSÃO

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

RODRIGO DOS SANTOS TAVARES

AUTOMAIL-X: SIMULADOR DE PRÓTESE ROBÓTICA AUTÔNOMA BASEADO EM PREVISÃO DE MOVIMENTOS

Boa Vista — RR

2019

Dados Internacionais de Catalogação na publicação (CIP)
Biblioteca Central da Universidade Federal de Roraima

T231a Tavares, Rodrigo dos Santos.
Automail-X : simulador de prótese robótica autônoma baseado em previsão de movimentos / Rodrigo dos Santos Tavares. – Boa Vista, 2019.
46 f. : il.

Orientador: Prof. Dr. Herbert Oliveira Rocha.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Roraima, Curso de Ciência da Computação.

1 - Prótese. 2 - Robótica de reabilitação. 3 - Sistemas embarcados. 4 - Sensores e computação gráfica. I - Título. II - Rocha, Herbert Oliveira (orientador).

CDU - 681.5

Ficha Catalográfica elaborada pela Bibliotecária/Documentalista:
Maria de Fátima Andrade Costa - CRB-11/453-AM

RODRIGO DOS SANTOS TAVARES

AUTOMAIL-X: SIMULADOR DE PRÓTESE ROBÓTICA AUTÔNOMA BASEADO EM PREVISÃO DE MOVIMENTOS

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Roraima como requisito para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Dr. Herbert Oliveira Rocha

Boa Vista — RR

2019

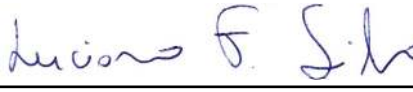
RODRIGO DOS SANTOS TAVARES

AUTOMAIL-X: SIMULADOR DE PRÓTESE ROBÓTICA AUTÔNOMA BASEADO EM PREVISÃO DE MOVIMENTOS

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Roraima como requisito para a obtenção do grau de Bacharel em Ciência da Computação. Defendido em 10 de julho de 2019 e aprovado pela seguinte banca examinadora:

Prof. Dr. Herbert Oliveira Rocha
Orientador / Curso de Ciência da Computação - UFRR

Prof. Dr. Leandro Nelinho Balico
Curso de Ciência da Computação - UFRR



Prof. Dr. Luciano Ferreira Silva
Curso de Ciência da Computação - UFRR

RESUMO

Pessoas com membros inferiores amputados geralmente buscam próteses que os ajudem a realizar as tarefas cotidianas, porém as próteses mais comuns do mercado com preços mais acessíveis são passivas e carecem de funcionalidades que podem trazer mais naturalidade aos movimentos do usuário. Este trabalho propõe um simulador de próteses para o pé humano que funciona de forma autônoma através da classificação dos movimentos das articulações, a partir dos dados coletados de acelerômetros, giroscópios e sensores flexíveis posicionados nos joelhos do usuário. Com a classificação desses dados realizada com a utilização de técnicas de aprendizado de máquina, é possível realizar ações em uma prótese virtual. Foi observado através de experimentos que o simulador é capaz de demonstrar o funcionamento de uma prótese ativa que se adapta ao uso a fim de ser utilizado no desenvolvimento de tais dispositivos.

Palavras-chave: próteses, robótica de reabilitação, sistemas embarcados, sensores e computação gráfica

ABSTRACT

People with amputated lower limbs often seek prostheses to help them perform everyday tasks, but the most common and affordable ones on the market are passive and lack functionalities to make the user's movements more natural. This study proposes a simulator of a prosthesis for the human foot that works autonomously by classifying the user's movements, based on data collected from accelerometers, gyroscopes and flexible sensors positioned on their knees. With the classification of these data performed with the use of machine learning techniques, it is possible to perform actions on a virtual prosthesis. It was observed through experiments that the simulator, in order to be used in the development of an active prosthesis that adapts itself, is able to demonstrate the functioning of such devices.

Keywords: prostheses, rehabilitation robotics, embedded systems, sensors and computer graphics

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Soluções tecnológicas são desenvolvidas cada vez mais para suprir necessidades que melhorem a qualidade de vida das pessoas. Quando nos tornamos incapazes de interagir fisicamente com o ambiente ao nosso redor, buscamos este tipo de solução. O campo da robótica de reabilitação trabalha sobre esta ideia, visando trazer conforto e restaurar a independência de pessoas com diversos tipos de limitações, incluindo aqueles com membros amputados, que precisam de próteses para voltar às atividades cotidianas (SICILIANO; KHATIB, 2008).

Segundo Siciliano e Khatib (2008), o desafio do desenvolvimento de próteses de membros humanos é manter a funcionalidade de um membro natural. Aspectos desta naturalidade incluem controle intuitivo das articulações, controle da força dos membros para situações diversas, e sentidos tátil e de movimento que existem nos membros naturais.

As próteses mais comuns para membros inferiores são otimizadas para caminhada em linha reta e, por isso, têm rigidez fixa nas articulações, o que dificulta várias ações do cotidiano que fogem desse comportamento padrão (PEW; KLUTE, 2017). Conforme a análise de Dedić e Dindo (2011), próteses comerciais ainda costumam ser passivas mesmo com o avanço tecnológico dos últimos anos, e muitas funções motoras exigem uma força maior nas articulações.

De acordo com Novak et al. (2013), vários dispositivos que visam aprimorar ou restaurar funções motoras de membros inferiores foram desenvolvidos, incluindo exoesqueletos e próteses ativas. Estes sistemas são equipados com sensores utilizados para perceber o ambiente ao redor do corpo humano. Além de sensores localizados nos dispositivos em si, existem também sistemas com sensores como acelerômetros e giroscópios (INVENSENSE, 2013) montados no corpo do usuário, com o intuito de compreender as intenções do usuário e prever seus movimentos.

Para que ações como subir escadas e rampas sejam realizadas naturalmente, é necessário que se utilize mais energia nas articulações, que precisam de mais força nessas ações do que na caminhada plana. Mesmo com o controle computadorizado de articulações que permitem velocidades diferentes de caminhada, ainda é difícil para pessoas amputadas a subida de escadas (DEDIĆ; DINDO, 2011).

Com o advento das próteses ativas, tornou-se possível definir estratégias de controle diferentes para cada tipo de ação do usuário. Estas estratégias são necessárias porque a biomecânica da perna é bem variável em relação à ação realizada, dependendo se o indivíduo está caminhando em um chão plano, ou subindo ou descendo uma rampa ou escada, por exemplo. Um campo de pesquisa atual consiste em determinar precisa e rapidamente o tipo de ação que o usuário está realizando, sem que seja necessário um dispositivo externo à prótese (STOLYAROV et al., 2017).

Ainda segundo Stolyarov et al. (2017), os métodos mais eficazes de prever estas ações de caminhada atualmente envolvem o reconhecimento de padrões de alguns sensores como unidades de medição inerciais (IMU, em inglês), e eletrodos de eletromiografia superficial (sEMG), que é um método que varia muito fora de ambientes de laboratório, pois depende de diversos fatores fisiológicos.

Dado este contexto, é pertinente o desenvolvimento de novas técnicas em robótica de reabilitação que procurem melhorar a qualidade de vida de pessoas com certas necessidades. Este trabalho visa contribuir para o desenvolvimento de próteses robóticas que auxiliem em uma interação natural com o ambiente através de um simulador aliado a um sistema computacional embarcado de baixo custo. Este sistema funcionará através de sensores de movimento posicionados nos membros inferiores do usuário, cujos dados serão processados por algoritmos de aprendizado de máquina, para que sejam previstas as ações do indivíduo e seja adaptada a prótese de acordo com a situação.

1.1 Definição do Problema

Por conta da dificuldade que pessoas amputadas têm em realizar certas atividades com as próteses mais comuns do mercado, novas formas de auxiliá-las são necessárias para promover seu bem estar. Desta forma, o problema abordado por este trabalho é representado pelo seguinte questionamento: **Como simular uma prótese para o pé humano que funcione de forma autônoma através da previsão de movimentos das articulações com o intuito de ajudar no projeto e produção de próteses robóticas de baixo custo que auxiliem na locomoção de pessoas que as necessitam?**

1.2 Objetivos

O objetivo geral deste trabalho é projetar e avaliar um sistema de simulação para próteses robóticas que funcione através da previsão de movimentos das articulações do usuário durante as suas ações executadas, utilizando sensores e atuadores disponíveis no mercado nacional, visando a produção de próteses de baixo custo.

Os objetivos específicos são os seguintes:

1. Identificar métodos para a modelagem do *software* e do *hardware*;
2. Propor um método para prever movimentos de articulações através da classificação de sinais extraídos de membros residuais;
3. Projetar e desenvolver um sistema de simulação que exiba os movimentos realizados com os sensores e a previsão de movimentos de uma prótese ativa;

4. Validar e avaliar o sistema proposto pela análise de testes práticos e simulados, de modo a examinar a sua eficácia e aplicabilidade.

1.3 Organização do Trabalho

Esta **Introdução** apresentou o contexto, a motivação o problema e os objetivos deste trabalho. Os capítulos a seguir estão organizados da seguinte forma:

O Capítulo 2, **Conceitos e Definições**, aborda os conceitos necessários para este trabalho, mais especificamente: Sistemas Embarcados, Robótica e suas Aplicações, Reconhecimento de Padrões e Modelagem e validação de sistemas.

O Capítulo 3, **Trabalhos Correlatos**, demonstra outros trabalhos relacionados, que servem como comparação ou base para este.

O Capítulo 4, **Método Proposto**, descreve as etapas para o desenvolvimento dos objetivos deste trabalho.

O Capítulo 5, **Avaliação Experimental**, relata os experimentos realizados a partir da aplicação do método proposto e analisa os resultados.

Por fim, o Capítulo 6, **Considerações Finais**, apresenta conclusões sobre este trabalho e perspectivas para o futuro.

2 CONCEITOS E DEFINIÇÕES

Este capítulo apresentará a fundamentação teórica utilizada neste trabalho, abordando os temas mais pertinentes, como Sistemas Embarcados, Robótica, Reconhecimento de padrões, e Modelagem de sistemas.

2.1 Sistemas Embarcados

Sistemas computacionais estão presentes em diversos produtos desde computadores pessoais e *laptops*, até utensílios domésticos. São conhecidos como sistemas embarcados (SE) os sistemas computacionais que fazem parte de um dispositivo eletrônico maior, não representando sua totalidade, mas oferecendo recursos computacionais específicos para o seu funcionamento (VAHID; GIVARGIS, 2002).

Também pode-se definir um SE como um sistema computacional de propósito específico, e que não assume vários papéis de acordo com a necessidade do usuário, apenas realiza tarefas predeterminadas, em contraste a um computador pessoal (HEATH, 2002). Isto é, são sistemas eletrônicos projetados para funções específicas dentro de um dispositivo maior, que pode controlar o meio físico e permite sua interação com o usuário.

Algumas características de sistemas embarcados, segundo Schlett (1998), são restrições específicas como de custo de produção ou de consumo de energia, que é o caso de dispositivos móveis, pois dependem de uma bateria para funcionar. Vahid e Givargis (2002) citam também como exemplo de sistema embarcado uma câmera fotográfica digital, pois tem um propósito específico (capturar e processar fotografias), restrições de tamanho, custo, e consumo de energia.

Uma outra restrição comum que pode ser pertinente a um SE é de tempo. Alguns sistemas precisam necessariamente realizar certas operações dentro de um tempo esperado, correndo o risco de não desempenhar seu papel caso isso falhe, sendo conhecidos como sistemas em tempo real. Por exemplo, um sistema de controle de velocidade de cruzeiro de um automóvel precisa monitorar a velocidade, aceleração e desaceleração do veículo em tempo real, e qualquer atraso é considerado uma falha do sistema (VAHID; GIVARGIS, 2002).

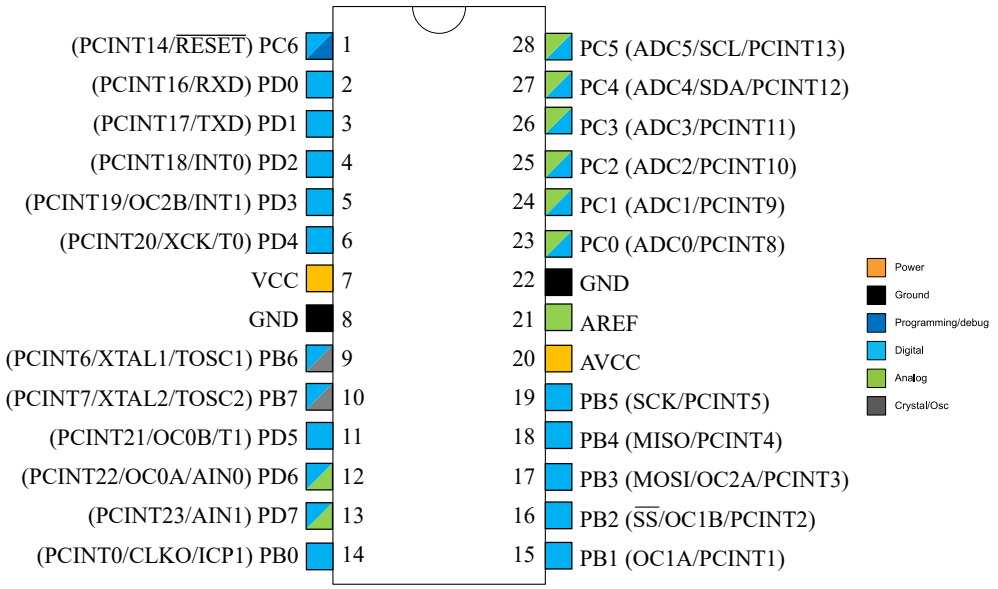
Conforme Buttazzo (2011) afirma, um sistema em tempo real não é apenas um sistema "rápido". É necessário que o tempo do sistema corresponda ao tempo *real*, ou seja, do mundo exterior. Sistemas robóticos, do ponto de vista de arquitetura, dependem de interação com ambientes dinâmicos e incertos, o que exige controle em tempo real dos sensores e atuadores, com suporte a concorrência e reagindo rapidamente a situações excepcionais (SICILIANO; KHATIB, 2008).

2.1.1 Microcontroladores *versus* Microprocessadores

Microcontroladores são processadores com vários componentes integrados, como RAM, uma memória de programa e interfaces de entrada e saída (WHITE, 2011). Esses componentes surgiram como um substituto para circuitos lógicos discretos, por serem mais facilmente programáveis e proporcionarem uma funcionalidade maior (HEATH, 2002). Além disso, segundo Marwedel (2010), os processadores em sistemas embarcados são geralmente microcontroladores, por serem simples e fáceis de usar.

O Arduino Uno Rev3, por exemplo, é uma placa de microcontrolador, equipado com o microcontrolador ATmega328/P e conta com 14 pinos de entrada/saída digitais e 6 entradas analógicas (ARDUINO, 2018). O ATmega328/P, cuja pinagem está ilustrada na Figura 1, é um microcontrolador 8-bit de baixo consumo de energia com arquitetura RISC que oferece 131 instruções. Além disso, contém uma memória *flash* programável de 32kB (ATMEL, 2016).

Figura 1 – Pinagem do microcontrolador ATmega328/P



Fonte: Atmel (2016, p.14)

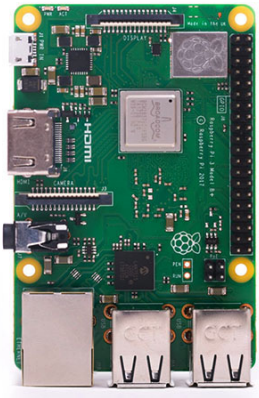
A distinção entre microprocessadores e microcontroladores não é tão trivial: Schlett (1998) afirma que, de forma simplificada, é comum diferenciá-los tendo como parâmetro seu desempenho, considerando dispositivos de 8 e 16-bit como microcontroladores. Também existem outros critérios a se considerar, como sua finalidade e suas possíveis restrições de energia, ou a necessidade de integração com vários componentes periféricos.

Microprocessadores não contam com RAM e ROM integradas e contém apenas cache. Este tipo de processador é encontrado nos computadores pessoais e costumam ter um desempenho aprimorado em relação aos microcontroladores. Por isso, alguns sistemas embarcados passaram a

usar microprocessadores também, pois alguns dispositivos como *video games* portáteis passaram a exigir maior capacidade de processamento. No entanto, estes microprocessadores embarcados ainda costumam ter restrições como de energia, custo, etc (SCHLETT, 1998).

Um computador de placa única que conta com um microprocessador é o Raspberry Pi 3 Model B+, mostrado na Figura 2, que contém um processador 64-bit Broadcom BCM2837B0, que funciona a 1.4GHz. O Raspberry Pi 3 Model B+ também tem 1GB de RAM, além de ser equipado com 4 portas USB, conectividade Bluetooth e WiFi, e oferecer 40 pinos de entrada/saída digitais (RASPBERRY PI FOUNDATION, 2018a). Pode ser utilizado como um computador pessoal, mas também para projetos eletrônicos (RASPBERRY PI FOUNDATION, 2018b).

Figura 2 – Raspberry Pi 3 Model B+



Fonte: Adaptado de Raspberry Pi Foundation (2018a)

2.1.2 IoT: Internet das Coisas

Com a constante evolução e a ubiquidade da Internet, começam a surgir objetos conectados, transformando dispositivos que já faziam parte do cotidiano em algo que possa ser autônomo e inteligente (KOPETZ, 2011). Segundo Xia et al. (2012), o termo "Internet das Coisas" (IoT) não se refere apenas à existência destes dispositivos inteligentes, mas à interconexão dos objetos do cotidiano através de sistemas embarcados, proporcionando assim ambientes em que os dispositivos se comunicam entre si e também com seres humanos de forma inteligente.

Para Gubbi et al. (2013), a visão completa da IoT se dará através de sensores e atuadores ubíquos, escondidos do usuário, atuando de forma independente. A ideia de computação ubíqua, também conhecida como pervasiva (SATYANARAYANAN et al., 2001), está bem relacionada com a IoT e se refere a um mundo repleto de sensores, atuadores e sistemas computacionais integrados nos objetos do dia-a-dia (GUBBI et al., 2013).

Um exemplo de objeto inteligente na IoT, segundo Kopetz (2011), é uma geladeira que mantém registro da validade e disponibilidade dos itens contidos, e faz pedidos automaticamente

ao supermercado mais próximo dos produtos em falta. Outro exemplo, ilustrado na Figura 3, é de um aspirador de pó automático, com sensores que detectam os obstáculos, e configurável por um aplicativo móvel (IROBOT, 2018).

Figura 3 – Aspirador de pó automático iRobot Roomba 980

Fonte: iRobot (2018)

2.2 Robótica e suas Aplicações

A robótica é uma área que busca sintetizar atividades humanas através do uso de mecanismos, sensores, atuadores e computadores. É comum que pesquisas neste campo sejam feitas por pesquisadores de outras áreas, servindo como meio para diversos fins (CRAIG, 2005).

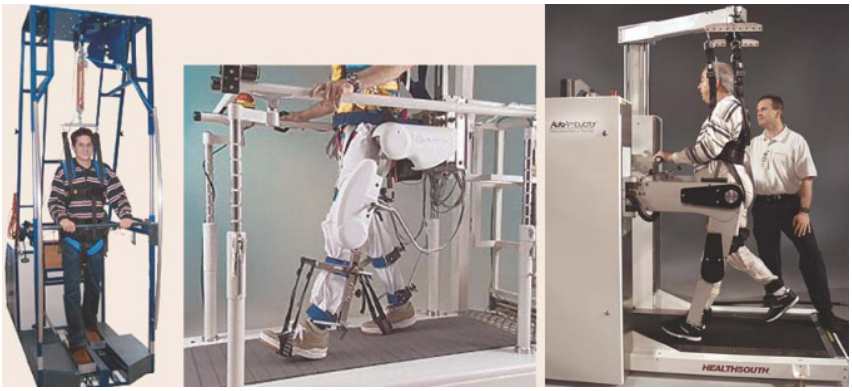
Figura 4 – Robô “Johnnie”

Fonte: Marwedel (2010, p.4)

Sistemas embarcados são tradicionalmente usados na área da robótica, relacionando-se aos aspectos mecânicos (MARWEDEL, 2010). Craig (2005) divide a robótica em quatro grandes áreas: manipulação mecânica, locomoção, visão computacional e inteligência artificial. Alguns robôs têm sido criados aos moldes de seres humanos, como o da Figura 4.

A robótica de reabilitação consiste em auxiliar pessoas com dificuldades motoras ou cognitivas através de sistemas como próteses robóticas e outros tipos de dispositivos, como os aparelhos de auxílio na restauração de caminhada de pacientes na Figura 5 (SICILIANO; KHATIB, 2008).

Figura 5 – Exemplos de robôs de auxílio na restauração de caminhada



Fonte: Siciliano e Khatib (2008, p.1232)

Neste contexto, a robótica tem sido aplicada para aprimorar a mobilidade de pessoas com membros amputados, através de próteses ativas, que podem permitir, por exemplo, uma caminhada mais natural do que uma prótese passiva comum (DEDIĆ; DINDO, 2011). Alguns projetos deste tipo, além deste trabalho, serão discutidos no Capítulo 3.

Alguns elementos cruciais da Robótica são o manipulador, além de atuadores e sensores. O manipulador refere-se à estrutura física dos robôs, como articulações. Atuadores são o que realizam os movimentos do robô; alguns tipos comuns de atuadores são: servomotores, motores de passo, atuadores pneumáticos e atuadores hidráulicos (NIKU, 2010).

Atuadores elétricos são o tipo de atuadores mais comuns na robótica, em especial servomotores, que são motores que podem ser controlados para se moverem em uma velocidade e torque desejados, e contam com um dispositivo de *feedback* que indica sua velocidade e ângulo no momento (NIKU, 2010).

Para que esses motores elétricos possam ser controlados por microprocessadores, é feita a conversão de sinal digital para analógico, e são necessárias várias saídas digitais para ter mais controle sobre a voltagem direcionada ao motor, para que se controle a velocidade, por exemplo (NIKU, 2010).

Sensores são os responsáveis por coletar informações sobre o estado interno do robô e por detectar o ambiente para que se determine como se deve interagir com ele; alguns sensores comuns são: potenciômetros, sensores de aceleração, de pressão, de proximidade, entre outros (NIKU, 2010).

A parte mecânica é um aspecto importante da Robótica que pode se referir à mecânica estrutural, relativa a peças físicas de sustentação como barras e parafusos, e a peças dinâmicas como polias, engrenagens e eixos, que permitem a movimentação (MODELIX ROBOTICS, 2010).

2.3 Reconhecimento de Padrões

O reconhecimento de padrões é um campo que busca encontrar padrões em conjuntos de dados, com finalidade em diversas áreas, e vem sendo desenvolvido ao lado de Aprendizado de Máquina (SAMUEL, 1959) no decorrer dos anos. Com isto, a intenção é usar algoritmos computacionais no intuito de descobrir automaticamente regularidades em dados, possibilitando, por exemplo, a classificação de tais dados (BISHOP, 2006).

Os dados extraídos de sensores podem ser usados para classificação a partir do reconhecimento de padrões. Esta seção abordará o aprendizado de máquina e alguns *frameworks* que podem ser usados na classificação dos movimentos a partir desses dados.

2.3.1 Aprendizado de máquina

O aprendizado de máquina é importante quando não é possível determinar todos os cenários possíveis de um sistema previamente, ou quando se deseja que este se adapte ao ambiente por conta própria. Também há o caso de ser impossível uma pessoa programar uma solução por conta própria, por exemplo: humanos conseguem reconhecer faces muito bem, mas não são capazes de desenvolver um programa que realize esta tarefa, sem usar algoritmos de aprendizagem (RUSSELL; NORVIG, 2010).

Existem três principais formas de aprendizagem que são determinadas a partir do tipo de *feedback* dado ao algoritmo. No **aprendizado não supervisionado**, o agente aprende os padrões da entrada mesmo sem ter nenhum tipo de *feedback*. Agrupamento (ou *clustering*) é o tipo de aprendizagem não supervisionada mais comum, em que se detecta grupos a partir de exemplos de entrada. O **aprendizado por reforço** consiste em aprender baseado em um *feedback* positivo ou negativo do resultado do algoritmo, indicando se algo de errado foi feito. Dessa forma, decide-se que ações pode ter ocasionado um resultado indesejado (RUSSELL; NORVIG, 2010).

No **aprendizado supervisionado**, o agente observa pares de entrada e saída e aprende uma função que mapeia os valores de entrada para os de saída, aplicando essa função em entradas futuras. Os pares de entrada e saída são chamados de conjunto de treinamento e a função a ser encontrada é chamada de hipótese. Para medir a acurácia da hipótese, é escolhido um conjunto de teste, diferente do conjunto de treinamento. Considera-se que a hipótese é uma boa generalização se for capaz de prever o valor de saída dos exemplos de teste (RUSSELL; NORVIG, 2010).

Quando os valores de saída são números, o problema é conhecido como **regressão**.

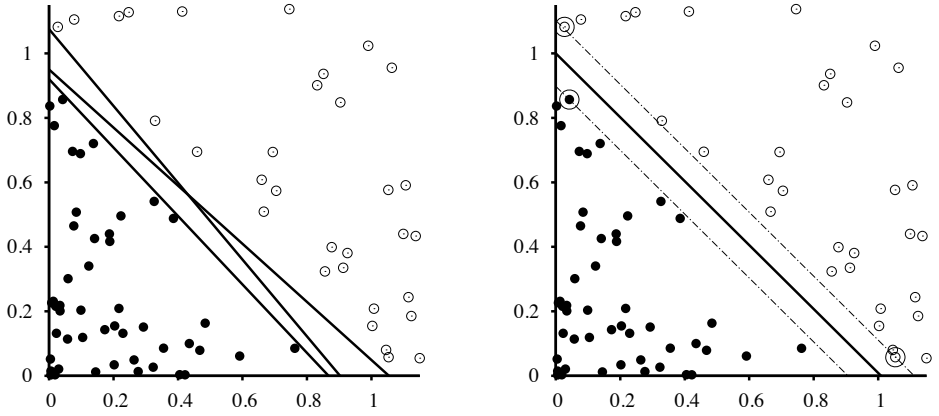
Quando a saída é um conjunto de valores finitos, o problema de aprendizagem é chamado de **classificação**, também conhecido como classificação Booleana ou binária caso hajam apenas dois valores (RUSSELL; NORVIG, 2010).

Uma das técnicas mais populares de aprendizado de máquina supervisionado é a Máquina de Vetores de Suporte (SVM, em inglês), por não exigir um conhecimento prévio do domínio em que será aplicado (RUSSELL; NORVIG, 2010). A SVM funciona ao encontrar um hiperplano que tem a maior margem possível entre as diferentes classes (HEARST et al., 1998). Como no exemplo da Figura 6, é possível ver três hiperplanos em 6a e o separador linear desejado em 6b, consistindo no hiperplano com a maior margem entre os vetores de suporte, que são os pontos circulados (RUSSELL; NORVIG, 2010). Neste exemplo, é feita a classificação entre os círculos pretos e os brancos.

Figura 6 – Classificação binária com SVM

(a) Duas classes de pontos (círculos pretos e brancos) e três candidatos a separadores lineares.

(b) O separador com maior margem (linha escura) no ponto médio da margem (entre as linhas tracejadas). Os vetores de suporte (pontos circulados) são os mais próximos ao separador.



Fonte: Adaptado de Russell e Norvig (2010, p. 745)

2.3.2 Ferramentas para aprendizado de máquina

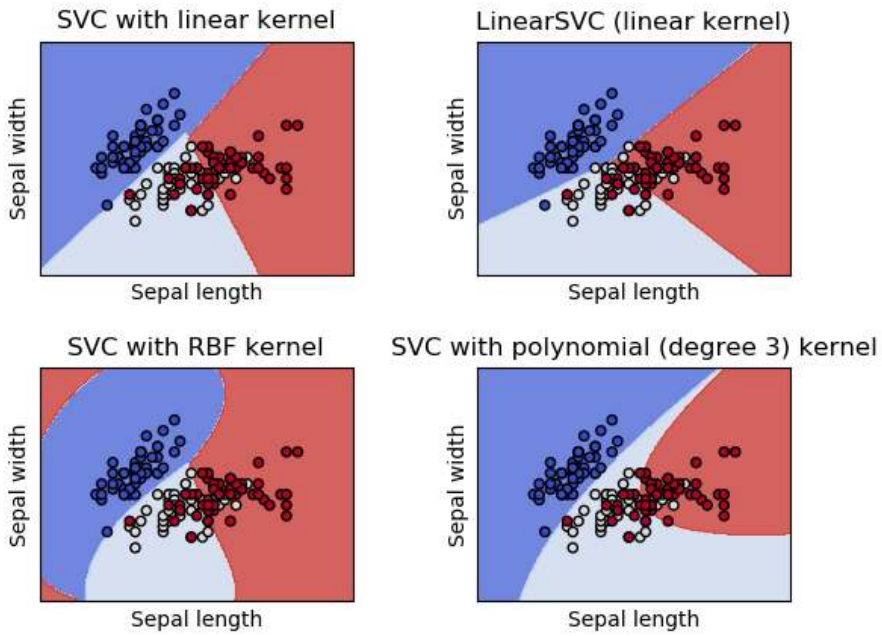
Dada a complexidade de algoritmos de aprendizado de máquina, e visando auxiliar na produção de soluções em diversas áreas, estão disponíveis diferentes ferramentas de desenvolvimento, como scikit-learn (PEDREGOSA et al., 2011) e TensorFlow (ABADI et al., 2015). Algumas destas ferramentas serão abordadas nas próximas seções.

2.3.2.1 Scikit-learn

O scikit-learn[1] é um módulo para a linguagem Python que integra uma alta gama de algoritmos de aprendizado de máquina, tanto supervisionados quanto não supervisionados, como SVM (CORTES; VAPNIK, 1995), KNN (COVER; HART, 1967), e K-Means (MACQUEEN et al., 1967), e permite a comparação fácil de diversos métodos em uma aplicação (PEDREGOSA et al., 2011).

Segundo Pedregosa et al. (2011), o scikit-learn aproveita o alto nível da linguagem Python para manter a facilidade de uso do *framework*, tornando-o utilizável por não especialistas da indústria de *software* e em campos além da ciência da computação. Além disso, Pedregosa et al. (2011) garantem maior eficiência do que outras ferramentas semelhantes em Python, pois o scikit-learn incorpora código compilado. A Figura 7 demonstra exemplos de classificação em SVM, com diferentes características, feitos com o scikit-learn.

Figura 7 – Exemplo de classificação de dados utilizando scikit-learn



Fonte: scikit-learn (2017)

Ainda usando SVM, o Programa 2.1 faz o treinamento, no qual o parâmetro `X` contém as amostras, e o parâmetro `y` contém os rótulos das classes. Já o Programa 2.2 faz a previsão: a linha `1` é o comando realizado, dado um novo valor, e a linha `2` representa o resultado, com os dados classificados (SCIKIT-LEARN, 2017).

[1] <http://scikit-learn.org/>

Programa 2.1 – Treino do classificador SVM com scikit-learn

```
1 from sklearn import svm
2 X = [[0, 0], [1, 1]]
3 y = [0, 1]
4 clf = svm.SVC()
5 clf.fit(X, y)
```

Fonte: scikit-learn (2017)

Programa 2.2 – Previsão usando classificador SVM com scikit-learn

```
1 >>> clf.predict([[2., 2.]])
2 array([1])
```

Fonte: scikit-learn (2017)

2.3.2.2 TensorFlow

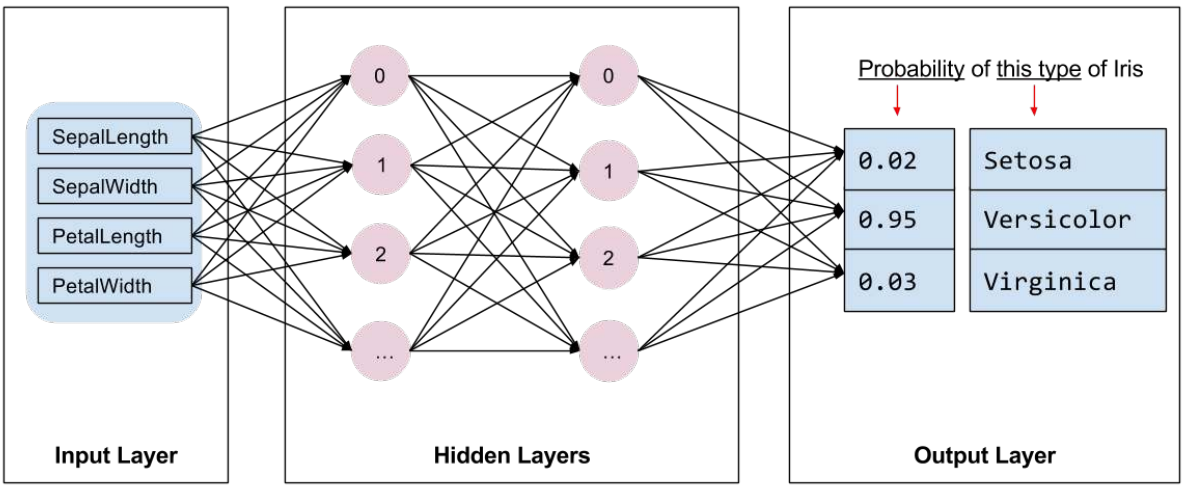
TensorFlow[2] é um sistema para algoritmos de aprendizado de máquina que funciona em diversos sistemas diferentes de forma flexível, sem exigir grandes alterações em uma variedade de sistemas heterogêneos (ABADI et al., 2016).

A ferramenta oferece classificação utilizando técnicas como Redes Neurais Convolutivas (CNN) e SoftMax, além de ser bastante popular (ERTAM; AYDIN, 2017). Empresas como Google e Mozilla utilizam o TensorFlow em projetos de Aprendizado de Máquina, como o projeto de reconhecimento de fala da Mozilla chamado *Deep Search* (TENSORFLOW, 2018b).

A Figura 8 ilustra uma Rede Neural Profunda (*Deep Neural Network*, em inglês) com duas camadas ocultas, contendo dez nós cada. Este programa utiliza dados já contidos na API para classificar flores em três espécies diferentes, baseado nos tamanhos das sépalas e pétalas delas (TENSORFLOW, 2018a).

[2] <http://tensorflow.org/>

Figura 8 – Rede Neural Profunda construída com o TensorFlow



Fonte: TensorFlow (2018a)

O Programa 2.3 mostra o código referente ao classificador representado na Figura 8, enquanto o Programa 2.4 demonstra o treino da rede neural, e o Programa 2.5 demonstra a previsão a partir do modelo treinado (TENSORFLOW, 2018a).

Programa 2.3 – Inicializando o classificador no TensorFlow

```
1 # Constrói uma DNN com 2 camadas ocultas e 10 nós em cada uma delas.
2 classifier = tf.estimator.DNNClassifier(
3   feature_columns=my_feature_columns,
4   # Duas camadas ocultas de 10 nós cada.
5   hidden_units=[10, 10],
6   # O modelo deve escolher entre 3 classes.
7   n_classes=3)
```

Fonte: Adaptado de TensorFlow (2018a)

Programa 2.4 – Treinando o classificador no TensorFlow

```
1 # Treina o modelo.
2 classifier.train(
3    input_fn=lambda:iris_data.train_input_fn(train_x, train_y, args.
        batch_size),
4    steps=args.train_steps)
```

Fonte: Adaptado de TensorFlow (2018a)

Programa 2.5 – Fazendo previsões a partir do modelo treinado no TensorFlow

```
# Gera previsões do modelo
expected = ['Setosa', 'Versicolor', 'Virginica']
predict_x = {
    'SepalLength': [5.1, 5.9, 6.9], # Comprimento da sépala
    'SepalWidth': [3.3, 3.0, 3.1], # Largura da sépala
    'PetalLength': [1.7, 4.2, 5.4], # Comprimento da pétala
    'PetalWidth': [0.5, 1.5, 2.1], # Largura da pétala
}

predictions = classifier.predict(
    input_fn=lambda:iris_data.eval_input_fn(predict_x,
                                    batch_size=args.batch_size))
```

Fonte: Adaptado de TensorFlow (2018a)

2.4 Modelagem e validação de sistemas

Sistemas embarcados são muitas vezes usados em situações críticas, em que segurança e confiabilidade são critérios muito importantes, porque podem oferecer risco à vida ou grandes prejuízos (EDWARDS et al., 1997). Buttazzo (2011) apresenta o caso do foguete Ariane 5, que em 1996 teve uma falha 37 segundos após seu lançamento e teve que ser destruído, pois desviou do curso correto. A falha foi causada por um erro de conversão de inteiros não previsto (BUTTAZZO, 2011).

O uso de modelos formais, como redes de Petri (PETERSON, 1981), para descrever o comportamento do sistema antes que este seja desenvolvido é uma forma de se aproximar da melhor confiabilidade e evitar erros, pois esses modelos podem ser validados automaticamente (EDWARDS et al., 1997).

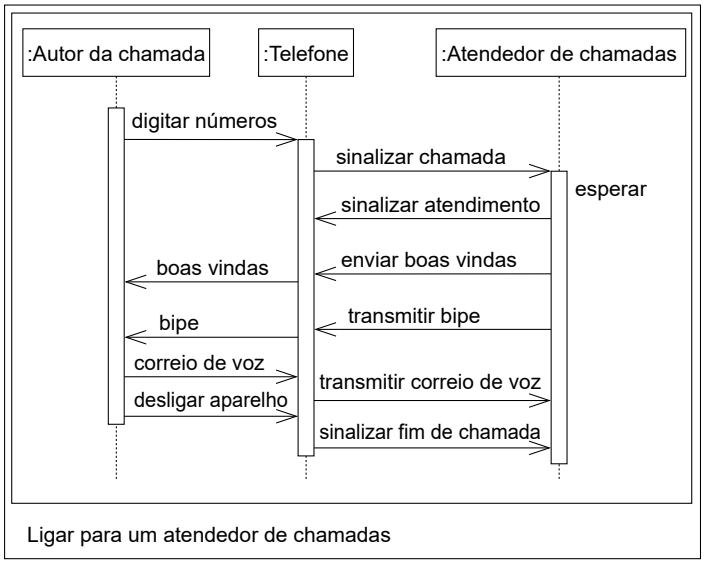
2.4.1 Diagrama de sequência

Uma forma de detalhar as especificações iniciais do *design* do sistema é através do diagrama de sequência, que é um modelo especificado pela UML. A UML (*Unified Modeling Language*) é uma padronização que visa auxiliar nas etapas iniciais de especificação de sistemas, desenvolvida por especialistas de *software* e utilizada por ferramentas comerciais (MARWEDEL, 2010).

Este diagrama indica a troca de mensagens sequencial entre os diversos componentes de um sistema. Na Figura 9, pode-se ver um exemplo de um atendedor telefônico automático. A dimensão vertical neste exemplo representa a sequência, e a horizontal representa cada componente de comunicação (MARWEDEL, 2010).

As linhas tracejadas representam as “linhas da vida”, e as mensagens são ordenadas em sequência ao longo dessas linhas. Caixas sobre as linhas da vida representam controle

Figura 9 – Exemplo de diagrama de sequência em UML



Fonte: Adaptado de Marwedel (2010, p. 37)

ativo do componente correspondente. No exemplo da Figura 9, as setas denotam mensagens assíncronas, e pode-se notar que a máquina espera o usuário atender o telefone. Caso isso não ocorra, a própria máquina atende e envia as boas vindas para o autor da chamada, que deixa uma mensagem no correio de voz (MARWEDEL, 2010).

2.4.2 Máquinas de Estado

A teoria dos autômatos é o estudo de dispositivos computacionais abstratos ("máquinas"), que servem de modelo para o projeto e construção de softwares, a partir de conceitos como autômatos finitos e gramáticas formais (HOPCROFT et al., 2001).

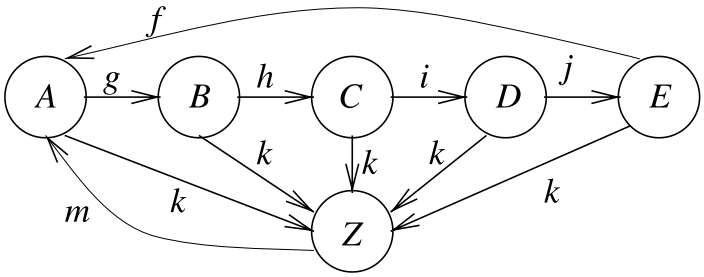
Um autômato finito, também conhecido como Máquina de Estados Finita (FSM) (WAGNER et al., 2006), se baseia em um conjunto finito de estados, entradas, saídas e transições entre estados. A descrição do comportamento baseado em estados é importante na modelagem de sistemas embarcados (MARWEDEL, 2010).

De acordo com Wagner et al. (2006), uma FSM é uma quíntupla $\{\Sigma, S, s0, \delta, F\}$, onde:

- Σ é o alfabeto de entrada (um conjunto finito não vazio de símbolos)
- S é um conjunto não vazio de estados
- $s0$ é um estado inicial, um elemento de S
- F é o conjunto de estados finais, um subconjunto de S.

Cada estado de uma FSM representa uma informação sobre o decorrer do programa, pois o estado muda de tempos em tempos durante a execução, e todos os estados representam todas as situações possíveis em que a máquina de estados pode estar. As saídas podem depender tanto do estado atual quanto das entradas da máquina (WAGNER et al., 2006).

Figura 10 – Exemplo de diagrama de estados

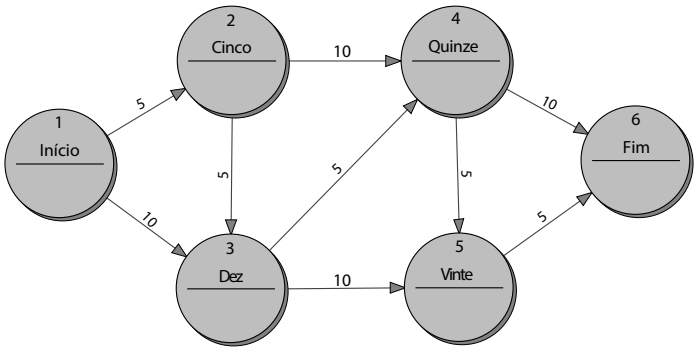


Fonte: Marwedel (2010, p. 39)

Um diagrama de estados como o da Figura 10 é uma representação clássica de uma FSM. Os círculos representam estados, as arestas são transições e o rótulo das arestas são eventos de transição, que levam a máquina a mudar de estado (MARWEDEL, 2010).

A Figura 11 representa um sistema de máquina de venda que aceita moedas de 5 e de 10 centavos, sendo 25 o valor esperado. O estado "Início" é o estado em que ainda não foi depositada uma moeda e o estado "Fim" quer dizer que a máquina recebeu 25 centavos. As transições representam as moedas depositadas na máquina. Em todos os estados qualquer moeda é aceita, exceto no estado "Vinte", em que a moeda de 10 centavos é ignorada neste exemplo por questões de simplicidade (WAGNER et al., 2006).

Figura 11 – Diagrama de máquina de estados do contador de uma máquina de vendas



Fonte: Adaptado de Wagner et al. (2006, p. 68)

3 TRABALHOS CORRELATOS

Este capítulo apresentará trabalhos existentes na literatura que compartilham de objetivos semelhantes ou são comparáveis de alguma forma e contribuem para o desenvolvimento deste trabalho. Os trabalhos descritos nas seções a seguir propõem métodos relacionados a previsão de movimentos para próteses ativas.

3.1 Translational Motion Tracking of Leg Joints for Enhanced Prediction of Walking Tasks

Neste artigo, Stolyarov et al. (2017) propuseram um método de prever tarefas de caminhada utilizando apenas uma IMU (unidade de medição inercial) em uma prótese comercial de uma articulação (tornozelo-pé). Foram classificadas cinco tarefas diferentes: caminhada em solo plano, subida de rampa, descida de rampa, subida de degrau e descida de degrau.

Usando apenas a IMU, foi possível ter uma classificação até mais precisa que próteses que usam sEMG, pois, segundo os autores Stolyarov et al. (2017), estas costumam ter ótimos resultados apenas em ambientes de laboratório, mas fora dele dependem de muitos fatores não controlados. A alta acurácia do classificador apresentado sugere que o método tenha ótimos resultados mesmo fora de laboratório.

Algumas limitações deste método são a falta de filtros para tratar problemas comuns do tipo de sensor utilizado, como *drifting*, e a baixa precisão que o algoritmo tem em superfícies escorregadias ou macias. O estudo do artigo de Stolyarov et al. (2017) proporciona ao trabalho aqui proposto a noção de que o uso de acelerômetros e giroscópios pode ser o suficiente para o objetivo proposto, principalmente quando se trata de próteses de baixo custo.

3.2 Turn Intent Detection for Control of a Lower Limb Prosthesis

No trabalho de Pew e Klute (2017) foi proposto fazer uma prótese para membros inferiores que tivesse rigidez adaptável em tempo real durante movimentos de virada, prevendo as ações de caminhada e de virada usando técnicas de aprendizagem de máquina, tais como, SVM e KNN.

A leitura dos dados foi feita a partir da simulação de uma IMU usando captura de movimentos ótica, e o controle da rigidez da articulação foi feita usando um componente

chamado VSTA (*variable stiffness torsion adapter*), que podia ser ativado com uma célula de carga comercialmente conhecida como iPecs, como pode ser visto na figura 12.

Figura 12 – Prótese incluindo o VSTA, a célula de carga iPecs e a posição da IMU simulada.

Fonte: Pew e Klute (2017)

Para a previsão de ações, foi feita uma comparação entre técnicas de classificação em aprendizagem de máquina: *Bagged Decision Tree Ensemble* (BREIMAN, 1996)(DIETTERICH, 2000), SVM (CORTES; VAPNIK, 1995) e KNN (COVER; HART, 1967). Na previsão de caminhada, o SVM teve 85% de acurácia, o KNN teve 82%, e o *Ensemble*, 97%. Na previsão de virada, o SVM teve 96%, o KNN, 93% e o *Ensemble* teve 91% de acurácia.

Embora tenha sido inferior na classificação da intenção de virada, a alta acurácia do *Bagged Decision Tree Ensemble* na detecção de caminhada compensa, o que o torna o algoritmo mais preciso dentre os testados. Também foi constatado, através da IMU simulada, que a acurácia do giroscópio (83% a 90%) foi superior à do acelerômetro (66% a 77%), o que conclui que o uso apenas do giroscópio pode ser suficiente.

A comparação dos algoritmos feita pelos autores Pew e Klute (2017) e os testes feitos com a IMU simulada podem ser úteis para este trabalho, pois valida a utilização dos sensores como forma de leitura de movimentos para previsão de intenções.

3.3 Automated detection of gait initiation and termination using wearable sensors

Novak et al. (2013) desenvolveram um método de previsão de início e término de caminhada, a partir de IMUs e uma sola sensível a pressão. O treino foi feito com pessoas com membros intactos e sem deficiências motoras, apenas para desenvolver a técnica. Foram utilizados 12 sinais das IMUs, passando por filtros de Kalman (KALMAN, 1960), e 4 sinais das duas solas. Além disso, os sinais foram classificados utilizando uma árvore de classificação,

tanto para detectar os passos individuais quanto para prever as intenções de início e término da caminhada.

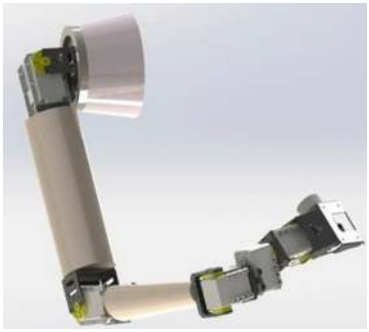
No trabalho de Novak et al. (2013) foi constatado que é possível detectar o início da caminhada usando a velocidade angular do quadril e ângulo do joelho; e o término utilizando a duração dos passos, e o padrão da pisada na sola para detectar se o passo atual será o último.

Ainda segundo Novak et al. (2013), IMUs são mais eficientes para detectar o término da caminhada, e os sensores nas solas são melhores para detectar os passos individuais. O sistema pode ser adaptado para ser usado em dispositivos como próteses ou exoesqueletos, embora precise ser reduzido para isso, devido à grande quantidade de sensores.

3.4 A Novel Design of a Full Length Prosthetic Robotic Arm for the Disabled

O trabalho de Kumar et al. (2017) propõe um protótipo de braço prostético robótico (ver Figura 13) controlado por um microcontrolador, com uso de aprendizagem de máquina. Além disso, o objetivo era fazer com que a prótese tivesse movimentos com aspecto humano e o menor torque necessário possível.

Figura 13 – Modelagem do protótipo desenvolvida no Solid Works



Fonte: Kumar et al. (2017)

O protótipo de Kumar et al. (2017) foi desenvolvido usando o *software* Solid Works[1], e a simulação foi feita com Simulink[2] e SimMechanics[3]. A classificação por aprendizado de máquina foi feita utilizando ANFIS (JANG, 1993), um tipo de rede neural artificial, a partir dos dados gerados na simulação.

Foram geradas 40 mil posições do braço, simulando um braço humano real. Estes dados foram utilizados para treinar o algoritmo ANFIS, que então foi capaz de determinar o ângulo das

[1] <https://www.solidworks.com/>
[2] <https://www.mathworks.com/products/simulink.html>
[3] <https://www.mathworks.com/products/simmechanics.html>

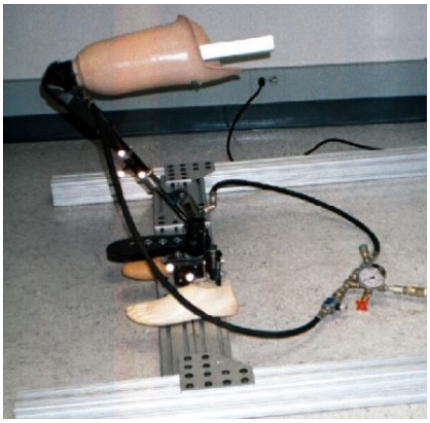
articulações. Para atuar nos movimentos do braço robótico, foram usados vários servomotores e um microcontrolador PIC16F886.

3.5 SmartLeg: An intelligent active robotic prosthesis for lower-limb amputees

O trabalho de Dedić e Dindo (2011) propõe uma forma de transformar uma prótese passiva disponível comercialmente em ativa, para que seja possível a subida e descida de escadas e outros movimentos que exigem um maior esforço motor. Assim, fez-se necessário o uso de uma fonte de energia externa, que foi feita usando atuadores hidráulicos nas articulações, como visto na Figura 14.

Dedić e Dindo (2011) também discute o uso de aprendizagem de máquina para garantir conforto aos usuários, pois pode-se adaptar os atuadores para funcionarem melhor com os padrões de andadura do indivíduo, mas isso não foi implementado. Outra ideia mencionada foi o uso de sensores de proximidade e sonares para detecção de obstáculos, algo que também não foi testado, apenas sugerido para o futuro.

Figura 14 – Protótipo com dois atuadores hidráulicos: nas articulações do joelho e do tornozelo.



Fonte: Dedić e Dindo (2011)

O protótipo da Figura 14 foi desenvolvido visando baixo custo, pois utiliza uma prótese passiva disponível no mercado, apenas a equipando com atuadores. A coleta de dados realizada permitiu que se comparasse o padrão de caminhada de indivíduos saudáveis com amputados utilizando a prótese, e a prótese produzida permitiu dois graus de liberdade (DOF) com o controle feito por microcontrolador. O maior empecilho é o sistema hidráulico utilizado, que ocupa muito espaço, tornando a versão proposta atual inviável para uso diário.

4 MÉTODO PROPOSTO

Este capítulo apresentará o método proposto por este trabalho para o desenvolvimento de um simulador de prótese ativa baseada em sensores e aprendizado de máquina.

4.1 Visão geral do método

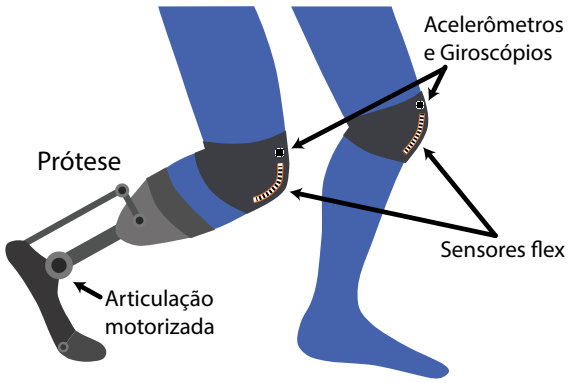
Este trabalho apresenta um simulador de prótese robótica para membros inferiores ou, mais especificamente, para as articulações do pé, visando prover suporte para a construção de modelos de próteses, aproveitando-se para simular as partes mecânicas articuladas e a inclusão de cenários de diferentes tipos de superfícies.

O simulador foi projetado para funcionar como uma prótese transtibial, ou seja, para pernas que contenham a articulação do joelho intacta, atuando sobre a rigidez de suas articulações para adaptá-la a diferentes situações. Isto é possível pois a simulação conta ainda com a utilização de um classificador de movimentos capaz de prever as ações necessárias nas partes mecânicas através da coleta de dados de sensores em um dispositivo vestível na forma de joelheira.

Este dispositivo está equipado com sensores flexíveis (SPECTRA SYMBOL, 2014) nas articulações dos joelhos além de giroscópios e acelerômetros (INVENSENSE, 2013), que são usados para capturar as ações do usuário. A Figura 15 ilustra a visão geral do sistema, incluindo o posicionamento dos sensores para o simulador juntamente com uma possível prótese implementada com atuadores reais (Figura 15a) e a simulação gerada a partir dos dados coletados dos sensores, com a movimentação do pé a partir da classificação destes dados (Figura 15b).

Figura 15 – Visão geral do sistema

(a) Proposta de implementação de prótese



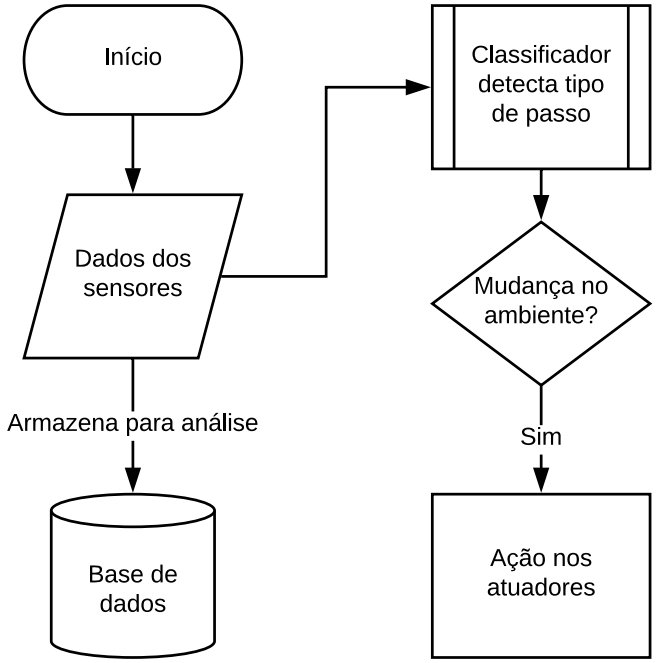
(b) Simulação virtual do protótipo

Fonte: Elaboradas pelo autor

No dispositivo vestível, os sensores são posicionados nos joelhos do usuário, e os

dados capturados são transmitidos por uma placa de processamento para o simulador em um computador. O simulador então utiliza os dados enviados para mostrar o modelo virtual, e os classifica para determinar a ação dos atuadores, conforme fluxograma ilustrado na Figura 16, em um modelo virtual da prótese em tempo real.

Figura 16 – Fluxograma da coleta de dados para o simulador



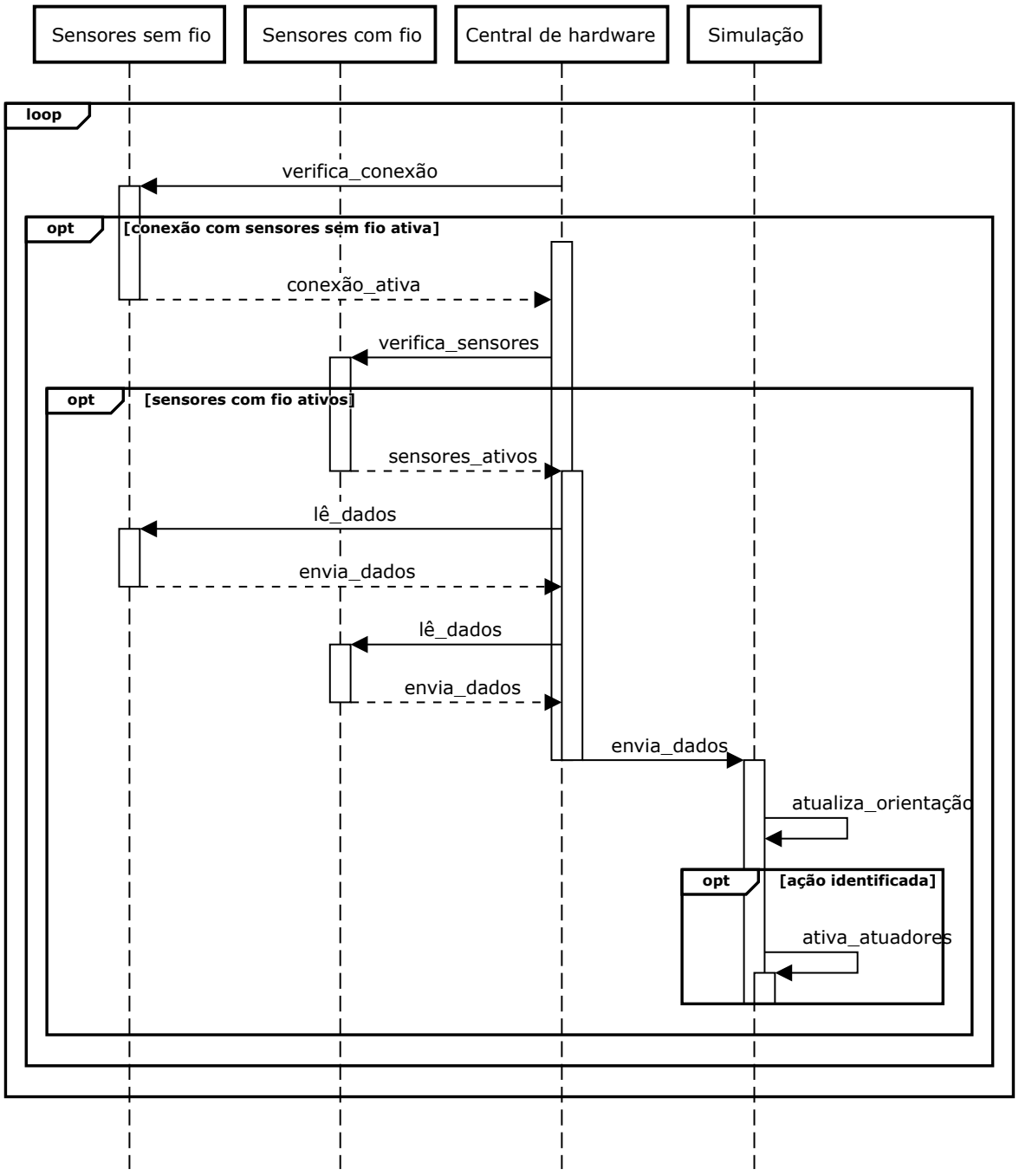
Fonte: Elaborada pelo autor

4.2 Prototipação da prótese no simulador

O diagrama de sequência da Figura 17 representa os componentes do sistema proposto por este trabalho, bem como a interação entre eles. Nela, pode-se observar a presença dos objetos caracterizados como **Sensores** com fio e sem fio e **Central de *hardware***, que compõem o dispositivo vestível, e **Simulação** que representa a visualização do modelo da perna e da prótese no computador.

A central localizada na perna da prótese recebe os dados de todos os sensores de forma constante e dados são então repassados à simulação, que os utiliza para orientar a posição da perna virtual de acordo com os dados dos giroscópios e acelerômetros e para classificar os movimentos, o que a permite que realize ações sobre as articulações da prótese virtual de acordo com o movimento detectado como, por exemplo, caso tenha sido iniciado um passo.

Figura 17 – Diagrama de sequência que descreve a interação entre os componentes do sistema



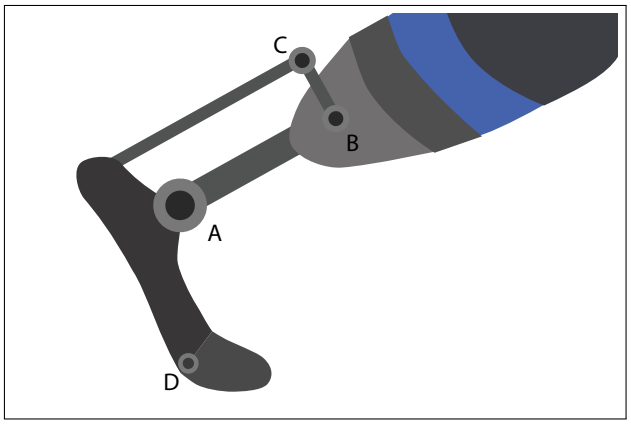
Fonte: Elaborada pelo autor

4.2.1 Arquitetura de *hardware* proposta

A prótese virtual é apresentada em um modelo virtual 3D que simula a ação dos atuadores ao prever os movimentos e conta com duas articulações ativas no pé, como visto na Figura 15b. O intuito deste modelo é proporcionar o futuro uso de uma impressora 3D para construção de uma prótese física. O uso de impressão 3D tem o intuito de manter o baixo custo do produto.

A Figura 18 mostra o funcionamento dos atuadores de um modelo proposto de prótese que se beneficiaria deste sistema, com atuadores que deverão ser acionados de acordo com a classificação dos dados dos sensores. No ponto **A**, o motor do tornozelo da prótese regula a rigidez da articulação, como forma de manter estabilidade nos passos. Os pontos **B** e **C** referem-se a motores de passo que, juntos a hastes, se prendem no calcanhar da prótese, para mover o pé e exercer força na pisada.

Figura 18 – Esquema representando o funcionamento dos atuadores



Fonte: Elaborada pelo autor

Por fim, o ponto **D** da Figura 18 é uma articulação motorizada que manipula a parte dianteira do pé, a fim de controlar a curvatura, simulando a flexibilidade de um pé humano. Alguns exemplos de configurações possíveis dos motores e articulações de acordo com diferentes cenários serão mostrados posteriormente na Figura 19. Deve-se salientar que este é um modelo de prótese proposto neste trabalho, contudo o simulador pode ser adaptado também a outros modelos.

4.3 Previsão de movimentos no simulador

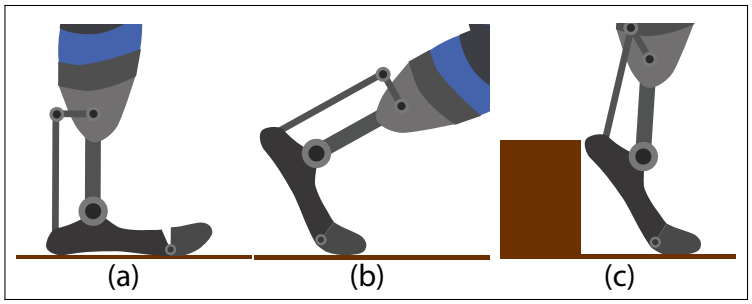
Os dados coletados são analisados por um algoritmo de aprendizado de máquina para que se classifique e se preveja as ações do usuário. Essas ações são classificadas de acordo com o passo a ser dado, podendo ser caminhada plana, subida ou descida de degrau. Enquanto o usuário realiza um passo, o sistema deverá identificar em que ambiente este passo será dado. Isto poderá ser feito a partir do passo anterior, e a partir do movimento atual extraído do dispositivo vestível posicionado no membro residual que se está analisando.

Considerando que haverá um conjunto predeterminado de cenários possíveis — caminhada plana, subida e descida de escada — a técnica de aprendizado de máquina supervisionado para classificação utilizada no simulador será escolhida e apresentada conforme analisado em testes feitos e apresentados na seção 5.2. Quando o tipo de passo a ser realizado é identificado

pelo sistema, acionam-se os atuadores, que terão sua ação determinada de acordo com o cenário identificado. Na Figura 19 pode-se observar a forma em que a prótese deve se posicionar de acordo com cada tipo de ação. As ações diferem para o início e o fim de cada passo em diferentes cenários.

As três ações representadas na Figura 19 referem-se a (a) pisada plana, (b) fim de um passo em piso plano, e (c) descida de escada. No primeiro caso, os motores só mantêm estabilidade para garantir equilíbrio; no segundo caso, é necessária maleabilidade na dianteira do pé, para que seja possível impulsionar o passo do usuário. No terceiro caso, antes que o usuário desça o próximo degrau, a prótese deve inclinar-se e contrair a dianteira do pé para evitar tropeços e auxiliar na descida.

Figura 19 – Algumas posições da prótese de acordo com cenários distintos



Fonte: Elaborada pelo autor

4.4 Diagnóstico do uso da prótese e recomendações de melhorias no caminhar

Os dados dos sensores também poderão ser armazenados para geração de estatísticas que podem ser usadas para diagnóstico relacionado à saúde da caminhada do futuro usuário da prótese (SABATINI et al., 2005). A partir das informações geradas, poderão ser feitas recomendações de melhorias ao indivíduo, via consulta a um especialista, para que se melhore a postura, a caminhada, ou que se utilize algum tipo de equipamento adicional.

Como trabalho futuro, pretende-se adicionar ao simulador um analisador de eficácia da prótese em relação aos sensores utilizados na joelheira para simulação da prótese, visto que tais sensores poderão ser utilizados na prototipação de uma prótese física. Para que este diagnóstico seja possível, será analisada a hipótese do uso de sensores adicionais aos definidos pelo protótipo atual, caso estes não sejam suficientes para gerar os dados necessários de análise de saúde da caminhada.

5 AVALIAÇÃO EXPERIMENTAL

Este capítulo aborda o planejamento, execução e análise do método proposto por este trabalho, a fim de verificar a viabilidade do projeto, bem como sua eficiência. O sistema computacional desenvolvido denomina-se Automail-X e tem seu código-fonte disponível em <https://github.com/rodrigost23/automailx>.

5.1 Planejamento dos experimentos

O intuito desta avaliação experimental é verificar a capacidade do sistema Automail-X de classificar as ações do usuário a partir da leitura de sensores, e exibir um modelo virtual animado que demonstra os dados lidos e a detecção dos movimentos realizados. Neste sentido, foram definidas as seguintes questões de pesquisa:

QP1: O sistema Automail-X é capaz de replicar os movimentos realizados pelo usuário em um ambiente virtual?

QP2: A previsão de movimentos do sistema Automail-X é precisa o suficiente para garantir a sua confiabilidade?

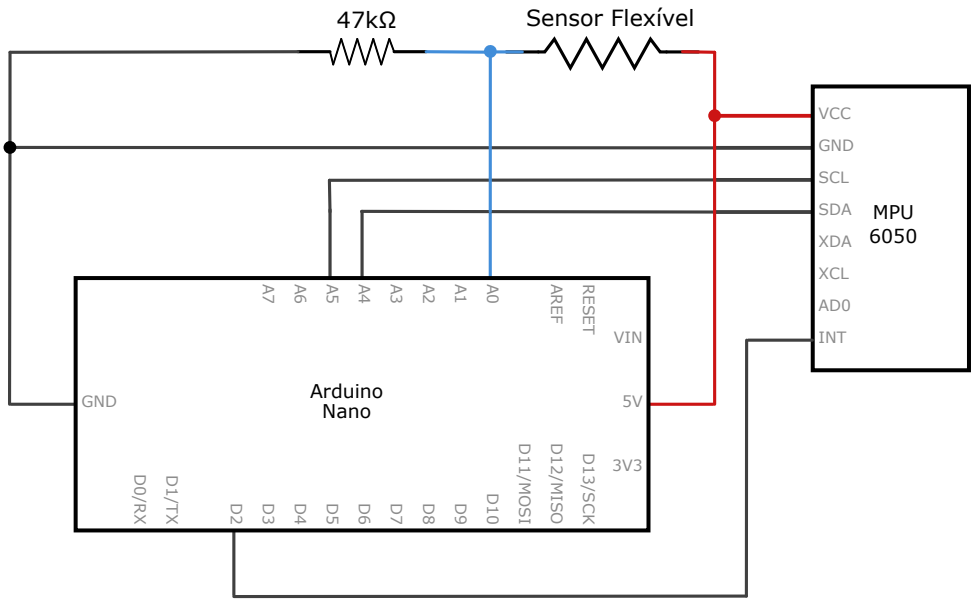
QP3: O sistema Automail-X pode prever as ações da prótese virtual em tempo real a partir dos dados dos sensores?

Visando responder essas questões, foi definido um ambiente que consiste em um sistema de captura de movimentos acoplado ao *software* simulador sendo executado em um computador. O protótipo para captura de dados consiste em um módulo GY-521 com um sensor MPU-6050 (INVENSENSE, 2013), que contém um acelerômetro e um giroscópio, e um sensor flexível SparkFun de $2{,}2$ polegadas (SPECTRA SYMBOL, 2014), conectados em um Arduino Nano 3.0 conforme o esquema da Figura 20.

Todos esses equipamentos foram fixados em uma joelheira de material flexível não rígido, como visto na Figura 21, com o MPU-6050 (a) posicionado acima do joelho, e o sensor flexível (b) posicionado na parte de trás. Este último sensor teve de ser posicionado desta forma devido ao seu comprimento de apenas $55{,}88\,\mathrm{mm}$, já que posicionamento na parte frontal do joelho, como planejado na seção 4.1, impediria que a resistência do sensor variasse o suficiente.

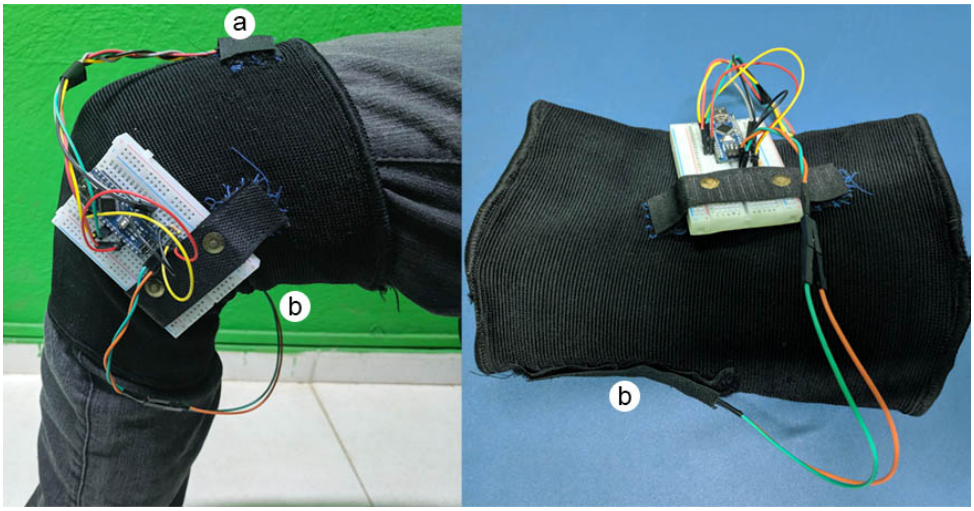
Para que fossem feitas a leitura e a classificação dos dados capturados, o Arduino foi conectado via cabo USB em um computador que executava o simulador. Os dados do giroscópio, acelerômetro e resistência do sensor flexível, respectivamente, são enviados continuamente via comunicação serial pelo *software* carregado no Arduino.

Figura 20 – Esquema das conexões dos sensores ao Arduino



Fonte: Elaborada pelo autor

Figura 21 – Joelheira com o Arduino Nano, o MPU-6050 (a) e o sensor flexível (b) fixados



Fonte: Elaborada pelo autor

O sistema de simulação é executado em um computador e foi desenvolvido na linguagem Python 3.7, para facilitar o uso da ferramenta Scikit-learn (PEDREGOSA et al., 2011). O programa do simulador é composto por diversos componentes que se integram para realizar diferentes atividades: leitura dos dados, através da biblioteca *pyserial*[1]; gravação dos dados em arquivos; classificação dos dados, utilizando Scikit-learn; e a exibição 3D com o uso das

[1] <https://github.com/pyserial/pyserial>

bibliotecas *PyOpenGL*[2] e *pygame*[3].

5.2 Execução dos experimentos e análise dos resultados

Com o protótipo confeccionado e os *softwares* desenvolvidos, deu-se início aos experimentos, visando responder às questões de pesquisa apresentadas na seção 5.1 para analisar a eficácia do sistema Automail-X em diferentes cenários de uso.

5.2.1 Transmissão de dados e ambiente virtual

O primeiro experimento realizado, visando responder à primeira questão de pesquisa (QP1), foi focado na transmissão de dados dos sensores para o simulador através da comunicação serial enquanto o ambiente virtual replicava os movimentos realizados pelos sensores.

Os dados de orientação, dentre os que são enviados pelo *software* carregado no Arduino, são recebidos pelo simulador e utilizados como ângulos de rotação do modelo da perna virtual, para replicar a orientação do membro do usuário que está utilizando o protótipo. Durante os experimentos, observou-se que foi necessário fazer uma calibragem do MPU-6050 a partir de sua posição vertical, pois os ângulos de rotação precisam ser relativos a uma posição inicial para que a orientação seja precisa.

Feito isso, o ambiente virtual foi capaz de replicar a posição da perna do usuário, com apenas uma ressalva: a falta de um magnetômetro no MPU-6050 utilizado impede que ele realize rotações em torno do eixo perpendicular ao solo, fazendo com que a simulação não reconhecesse bem movimentos de virada. Na Figura 22 é possível ver os valores dos sensores na parte superior, enquanto o modelo virtual os representa graficamente.

5.2.2 Classificação de movimentos

Para tornar possível a previsão dos movimentos e para responder à segunda questão de pesquisa (QP2), foi decidido que seriam armazenados os valores do acelerômetro e do sensor flexível, ignorando os dados de orientação do giroscópio, que ainda são utilizados para orientar o modelo virtual. Diversos conjuntos de dados foram gravados com diferentes tipos de ações, como caminhada, subida e descida de degrau.

Todos os dados foram capturados a partir de um cabo USB de $1\,\mathrm{m}$ conectado a um *notebook* (com um processador Intel i7-7500U, que conta com *clock* de até $3{,}5\,\mathrm{GHz}$, e $8\,\mathrm{GB}$ de RAM) no sistema operacional Manjaro Linux[4] 18.0.4, estabelecendo comunicação serial com o programa.

[2] <http://pyopengl.sourceforge.net/>
[3] <https://www.pygame.org/>
[4] <http://manjaro.org/>

Figura 22 – Posicionamento da perna virtual conforme dados dos sensores

Fonte: Elaborada pelo autor

Os indivíduos selecionados para a coleta dos dados tinham os membros intactos e vestiram a joelheira com os equipamentos na perna direita e realizavam as ações necessárias para cada cenário (apresentados nas seções a seguir) do experimento enquanto as transições entre os movimentos eram gravadas em um arquivo.

Após a consolidação do conjunto de dados para cada etapa dos experimentos, foi realizada uma comparação, através de validação cruzada (SCIKIT-LEARN, 2019b), entre a acurácia dos diversos algoritmos de classificação disponíveis na ferramenta scikit-learn: *Logistic Regression* – LR (SCIKIT-LEARN, 2019i), *Linear Discriminant Analysis* – LDA (SCIKIT-LEARN, 2019h), *K-Nearest Neighbors* – KNN (SCIKIT-LEARN, 2019g), *CART* (SCIKIT-LEARN, 2019c), *Gaussian Naive Bayes* – NB (SCIKIT-LEARN, 2019e), *Support Vector Machine* – SVM (SCIKIT-LEARN, 2019k), *AdaBoost Classifier* – ADB (SCIKIT-LEARN, 2019a), *Random Forest Classifier* – RFC (SCIKIT-LEARN, 2019j), *Extra-Trees Classifier* – ETC (SCIKIT-LEARN, 2019d) e *Gradient Boosting Classifier* – GBC (SCIKIT-LEARN, 2019f).

5.2.2.1 Cenário: caminhada em linha reta

O primeiro conjunto de dados gerado foi a partir de dois indivíduos que apenas caminhavam em linha reta sobre uma superfície plana. Cada um dos passos de cada perna deveria ser classificado de forma independente. Ou seja, além do estado de repouso (estado 0), foram armazenados o ponto em que a perna esquerda estava para frente (estado 1), e o ponto em que a perna direita estava para frente (estado 2), ilustrados pela Figura 23. Ao todo, foram coletadas

140 amostras.

Figura 23 – Demonstração das três ações capturadas para a classificação da caminhada

Fonte: Elaborada pelo autor

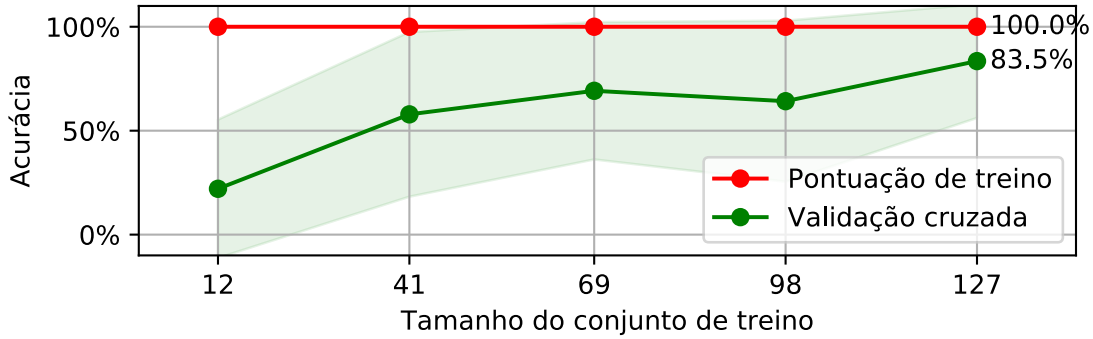
A partir dos dados coletados dos usuários, verificou-se que *Random Forest* (RFC) seria o algoritmo mais apropriado, com 82,81% de acurácia em relação aos outros algoritmos (conforme a Tabela 1), e com acurácia de 83,5% para os dados combinados de todos os usuários, como pode ser visto no gráfico da Figura 24.

Tabela 1 – Comparação dos classificadores com o conjunto de dados combinados para o cenário de caminhada

Algoritmo	LR	LDA	KNN	CART	NB
Acurácia	50,86%	69,19%	62,30%	75,76%	69,19%
Algoritmo	SVM	ADB	RFC	ETC	GBC
Acurácia	12,05%	48,48%	82,81%	81,33%	78,48%

Fonte: Elaborada pelo autor

Figura 24 – Gráfico de acurácia do classificador *Random Forest* para os dados combinados no cenário de caminhada



Fonte: Elaborada pelo autor

Ao serem analisados os conjuntos de dados de cada indivíduo, a Análise de Discriminantes Lineares (LDA) (SCIKIT-LEARN, 2019h) obteve maior acurácia, com aproximadamente 96,9%, como demonstram a Tabela 2 e a Figura 25. Devido ao pequeno número de amostras, não foi possível gerar um conjunto de dados genérico que funcionasse para todos os indivíduos a partir da coleta realizada. Portanto, uma possível solução seria que cada usuário teria que fazer a própria calibragem do dispositivo para prever os próprios movimentos.

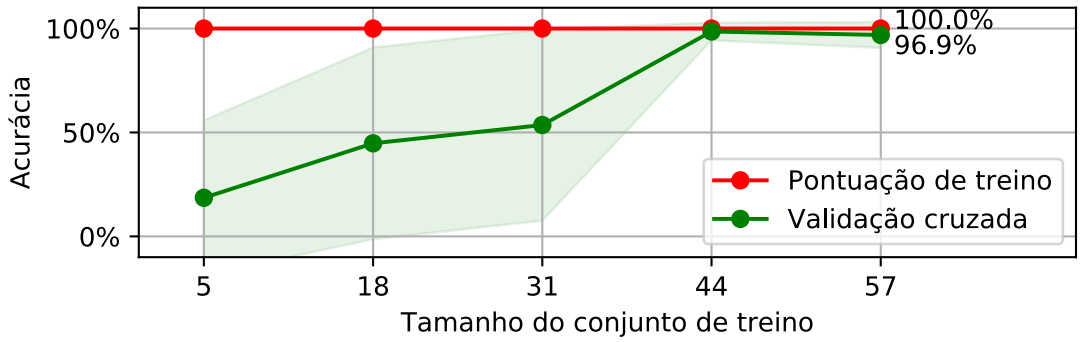
Tabela 2 – Comparação dos classificadores com o conjunto de dados individual para o cenário de caminhada

Algoritmo	LR	LDA	KNN	CART	NB
Acurácia	95,71%	96,9%	89,76%	92,62%	95,71%
Algoritmo	SVM	ADB	RFC	ETC	GBC
Acurácia	5,00%	94,29%	94,29%	94,29%	91,43%

Fonte: Elaborada pelo autor

Além disso, ao longo dos experimentos realizados, notou-se que o sensor flexível ficou cada vez menos preciso, o que tornou a visualização 3D um pouco menos realista, pois o ruído nos dados do sensor tornou-se maior, mas não impediu que os dados fossem classificados e as ações previstas. O uso de um sensor maior, que permitisse seu uso na parte frontal do joelho, pode fazer com que ele se desgaste menos.

Figura 25 – Gráfico de acurácia da LDA para um conjunto de dados individual no cenário de caminhada



Fonte: Elaborada pelo autor

5.2.2.2 Cenário: Subida e descida de escada

Após os experimentos com dados de caminhada em linha reta, foram iniciados os experimentos para classificação de subida e descida de degrau. Neste cenário, o usuário deveria subir um degrau com a perna direita (equipada com a joelheira), e descer o degrau com a perna esquerda. Estas ações foram identificadas como 3 e 4, respectivamente, e estão ilustradas na Figura 26.

Figura 26 – Ambiente utilizado para coleta de amostras de subida e descida de escada

Fonte: Elaborada pelo autor

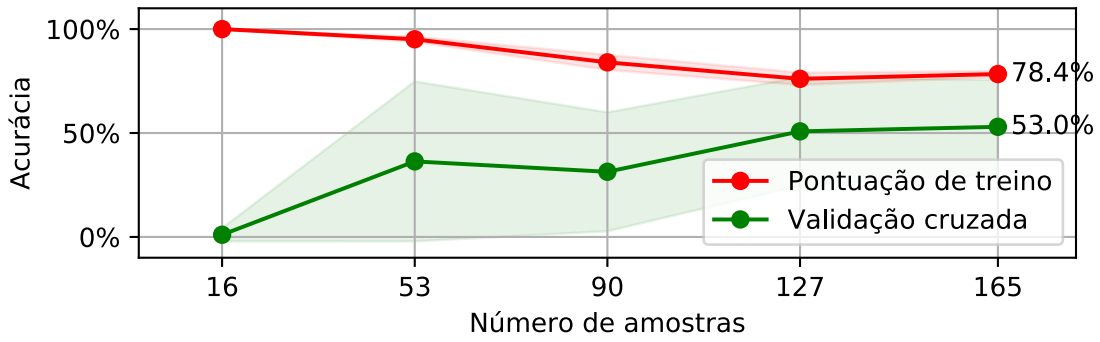
Ao todo, foram coletadas 76 amostras para este cenário. Ao unir ao conjunto de dados de caminhada plana do mesmo usuário, totalizaram-se 184 amostras. A comparação dos algoritmos vista na Tabela 3 resultou com o melhor desempenho do classificador *Gaussian Naive Bayes* (NB), com 52,43%. A acurácia deste classificador para os conjuntos de dados combinados foi de apenas 53%, como mostra a Figura 27.

Tabela 3 – Comparação dos classificadores com o conjunto de dados individual para todos os cenários combinados

Algoritmo	LR	LDA	KNN	CART	NB
Acurácia	49,44%	39,82%	36,20%	45,03%	52,43%
Algoritmo	SVM	ADB	RFC	ETC	GBC
Acurácia	4,91%	22,02%	46,2%	50,09%	45,23%

Fonte: Elaborada pelo autor

Figura 27 – Gráfico de acurácia do classificador *Gaussian Naive Bayes* com o conjunto de dados individual para todos os cenários



Fonte: Elaborada pelo autor

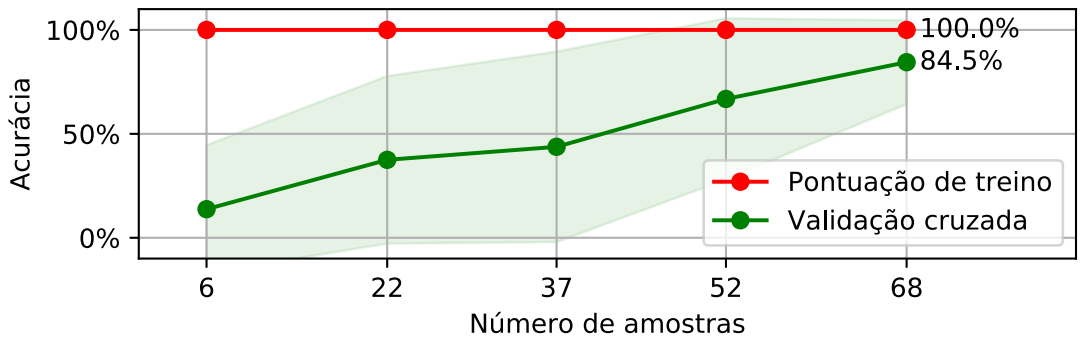
Em seguida, portanto, foram realizados testes com apenas os dados capturados para este cenário, removendo da classificação os dados gerados no cenário de caminhada. Neste caso, o algoritmo com melhor desempenho, conforme a Tabela 4, foi o *Extra-Trees* (ETC), com 81,96% de acurácia. Já na validação deste algoritmo isoladamente, a acurácia foi de 84,5%, como visto no gráfico da Figura 28.

Tabela 4 – Comparação dos classificadores com o conjunto de dados individual para o cenário de subida e descida de degrau

Algoritmo	LR	LDA	KNN	CART	NB
Acurácia	63,93%	71,96%	67,68%	80,54%	70,71%
Algoritmo	SVM	ADB	RFC	ETC	GBC
Acurácia	26,79%	30,54%	80,36%	81,96%	67,50%

Fonte: Elaborada pelo autor

Figura 28 – Gráfico de acurácia do classificador *Extra-Trees* com o conjunto de dados individual para o cenário de subida e descida de degrau



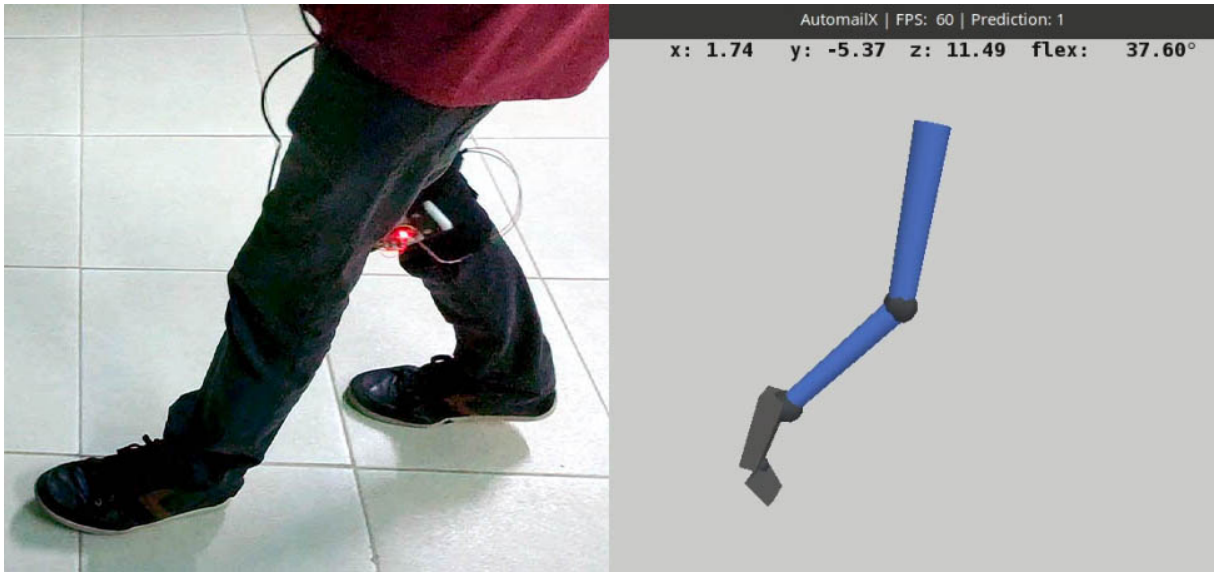
Fonte: Elaborada pelo autor

Em vista destes resultados, constata-se que quanto maior o número de classes, maior é o número de amostras necessárias para o classificador e no caso de ter a disponibilidade de um número maior de amostras, o sistema deve conseguir funcionar de forma mais precisa com diversos tipos de cenários.

5.2.3 Simulação da prótese em tempo real

Por fim, com o intuito de responder à terceira questão de pesquisa (QP3), a animação da prótese virtual foi avaliada em relação à confiabilidade da ação de seus atuadores. Utilizando apenas o conjunto de dados correspondente ao usuário que estava vestindo o dispositivo, observou-se a ativação da prótese, conforme o usuário caminhava em linha reta.

Figura 29 – Simulação dos atuadores da prótese a partir da classificação dos dados dos sensores



Fonte: Elaborada pelo autor

A simulação animada em 3D mostrou-se capaz de exibir o movimento realizado pelos dois sensores em tempo real, além da ação realizada na prótese simulada. A Figura 29 mostra a ação correspondente a um passo com a perna oposta à perna com os sensores, ativando os atuadores da prótese ao detectar a classe 1.

Neste experimento de caminhada, de 8 passos realizados com a perna direita, apenas 2 não foram reconhecidos corretamente, enquanto todos os 8 passos realizados com a perna esquerda (Figura 29) e o estado de repouso foram classificados corretamente. Para aprimorar a precisão desta classificação, o uso de um conjunto de dados com mais amostras deverá ser suficiente para que as ações com as duas pernas sejam classificadas corretamente.

6 CONSIDERAÇÕES FINAIS

O presente trabalho abordou o desenvolvimento de um simulador de prótese robótica para o pé humano baseado em reconhecimento de movimentos através do uso de sensores em um sistema embarcado e utilização de técnicas de aprendizado de máquina, com o intuito de facilitar o desenvolvimento de próteses ativas. Desta maneira, foi criado um protótipo a fim de realizar experimentos e avaliar a eficácia do método proposto.

A partir dos experimentos realizados, constatou-se que o método proposto apresenta resultados significativos no que se propõe: prever ações do usuário com o uso de sensores, e exibir uma simulação contando com uma prótese virtual cujos atuadores são acionados em tempo real de acordo com a classificação dos movimentos. Dentro dos cenários avaliados, foi necessária a captura dos dados individuais dos usuários para garantir que o sistema fosse capaz de identificar cada ação de seus respectivos usuários.

Como propostas de trabalhos futuros, propõe-se a coleta de um conjunto de dados maior para se propiciar um modelo de previsão de movimentos genérico, para mais usuários e análise mais profunda da eficiência dos sensores adotados. Adicionalmente, deverão ser investigadas formas de analisar a saúde do usuário, por exemplo, em terrenos e superfícies variadas, de acordo com diretrizes ortopédicas, para que sejam usadas em um relatório gerado a partir dos dados dos sensores.

REFERÊNCIAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Disponível em: <https://www.tensorflow.org/>.

ABADI, M. et al. TensorFlow: A System for Large-Scale Machine Learning. In: **OSDI**. [S.l.: s.n.], 2016. v. 16, p. 265–283.

ARDUINO. **Arduino Uno Rev3**. 2018. Disponível em: <https://store.arduino.cc/arduino-uno-rev3>. Acesso em: 3 jul. 2018.

ATMEL. **ATmega328/P**: Datasheet complete. Microchip Technology, 2016. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf>. Acesso em: 3 jul. 2018.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York: Springer Science & Business Media, 2006. (Information Science and Statistics). ISBN 978-0387-31073-2.

BREIMAN, L. Bagging predictors. **Machine learning**, Springer, v. 24, n. 2, p. 123–140, 1996.

BUTTAZZO, G. C. **Hard real-time computing systems: predictable scheduling algorithms and applications**. 3. ed. [S.l.]: Springer Science & Business Media, 2011. v. 24.

CORTES, C.; VAPNIK, V. Support vector machine. **Machine learning**, v. 20, n. 3, p. 273–297, 1995.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, IEEE, v. 13, n. 1, p. 21–27, 1967.

CRAIG, J. J. **Introduction to Robotics: Mechanics and Control**. 3. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005. v. 3. ISBN 0-13-123629-6.

DEDIĆ, R.; DINDO, H. SmartLeg: An intelligent active robotic prosthesis for lower-limb amputees. **2011 23rd International Symposium on Information, Communication and Automation Technologies, ICAT 2011**, 2011.

DIETTERICH, T. G. Ensemble methods in machine learning. In: SPRINGER. **International workshop on multiple classifier systems**. [S.l.], 2000. p. 1–15.

EDWARDS, S. et al. Design of embedded systems: Formal models, validation, and synthesis. **Proceedings of the IEEE**, IEEE, v. 85, n. 3, p. 366–390, 1997.

ERTAM, F.; AYDIN, G. Data classification with deep learning using Tensorflow. In: **2017 International Conference on Computer Science and Engineering (UBMK)**. [S.l.: s.n.], 2017. p. 755–758.

GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. **Future generation computer systems**, Elsevier, v. 29, n. 7, p. 1645–1660, 2013.

HEARST, M. A. et al. Support vector machines. **IEEE Intelligent Systems and their applications**, IEEE, v. 13, n. 4, p. 18–28, 1998.

HEATH, S. **Embedded Systems Design**. [S.l.]: Elsevier Science, 2002. ISBN 9780080477565.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introduction to automata theory, languages, and computation**. 2. ed. [S.l.]: Addison-Wesley, 2001.

INVENSENSE. **MPU-6000 and MPU-6050 Product Specification**. TDK, 2013. Disponível em: <https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>. Acesso em: 3 jun. 2018.

IROBOT. **Roomba 980**: Robô Aspirador de Pó Inteligente Bivolt. 2018. Disponível em: <https://www.irobotloja.com.br/roomba-980---robo-aspirador-de-po-inteligente-bivolt-irobot-r980400/p>. Acesso em: 3 jul. 2018.

JANG, J.-S. Anfis: adaptive-network-based fuzzy inference system. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 23, n. 3, p. 665–685, 1993.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of basic Engineering**, American Society of Mechanical Engineers, v. 82, n. 1, p. 35–45, 1960.

KOPETZ, H. Internet of Things. In: **Real-Time Systems: Design Principles for Distributed Embedded Applications**. Boston, MA: Springer US, 2011. p. 307–323. ISBN 978-1-4419-8237-7.

KUMAR, V. S. et al. A Novel Design of a Full Length Prosthetic Robotic Arm for the Disabled. In: KIM, J.-H. et al. (Ed.). **Robot Intelligence Technology and Applications 4**. Cham: Springer International Publishing, 2017. p. 271–287. ISBN 978-3-319-31293-4. Disponível em: <https://link.springer.com/chapter/10.1007%2F978-3-319-31293-4_22>.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [S.l.: s.n.], 1967. v. 1, n. 14, p. 281–297.

MARWEDEL, P. **Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems**. 2. ed. [S.l.]: Springer, 2010. 412 p. ISBN 9789400702578.

MODELIX ROBOTICS. **Introdução à Robótica**. 2010. Disponível em: <http://www.leomar.com.br/brinquedos/images/stories/manuais/laboratorio/guia%20de%20robotica.pdf>. Acesso em: 3 jul. 2018.

NIKU, S. **Introduction to Robotics**. [S.l.]: John Wiley & Sons, 2010. ISBN 9780470604465.

NOVAK, D. et al. Automated detection of gait initiation and termination using wearable sensors. **Medical Engineering and Physics**, v. 35, n. 12, p. 1713–1720, 2013. ISSN 13504533.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PETERSON, J. L. **Petri Net Theory and The Modeling of Systems**. Englewood Cliffs, NJ.: Prentice-Hall, 1981.

PEW, C.; KLUTE, G. Turn Intent Detection for Control of a Lower Limb Prosthesis. **IEEE Transactions on Biomedical Engineering**, 2017. ISSN 15582531.

RASPBERRY PI FOUNDATION. **Raspberry Pi 3 Model B+**. 2018. Disponível em: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>. Acesso em: 3 jul. 2018.

RASPBERRY PI FOUNDATION. **Raspberry Pi FAQs**. 2018. Disponível em: <https://www.raspberrypi.org/help/faqs/>. Acesso em: 3 jul. 2018.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. Upper Saddle River, NJ: Prentice Hall, 2010. (Prentice Hall Series in Artificial Intelligence). ISBN 9780136042594.

SABATINI, A. M. et al. Assessment of walking features from foot inertial sensing. **IEEE Trans. Biomed. Engineering**, v. 52, p. 486–494, 2005.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of research and development**, IBM, v. 3, n. 3, p. 210–229, 1959.

SATYANARAYANAN, M. et al. Pervasive computing: Vision and challenges. **IEEE Personal communications**, v. 8, n. 4, p. 10–17, 2001.

SCHLETT, M. Trends in embedded-microprocessor design. **Computer**, v. 31, n. 8, p. 44–49, 1998. ISSN 0018-9162.

SCIKIT-LEARN. **scikit-learn documentation**: Support vector machines. 2017. Disponível em: <http://scikit-learn.org/stable/modules/svm.html#classification>. Acesso em: 1 jul. 2018.

SCIKIT-LEARN. **Ada Boost Classifier**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Cross-validation: evaluating estimator performance**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/cross_validation.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Decision Tree Classifier**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Extra Trees Classifier**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **GaussianNB**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Gradient Boosting Classifier**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **KNeighbors Classifier**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Linear Discriminant Analysis**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Logistic Regression**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **Random Forest Classifier**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>. Acesso em: 30 jun. 2019.

SCIKIT-LEARN. **SVC**. scikit-learn, 2019. Disponível em: <https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>. Acesso em: 30 jun. 2019.

SICILIANO, B.; KHATIB, O. **Springer Handbook of Robotics**. [S.l.]: Springer International Publishing, 2008. (Springer Handbooks). ISBN 9783319325521.

SPECTRA SYMBOL. **Flex Sensor Data Sheet 2014**. SparkFun, 2014. Disponível em: <https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEX\%20SENSOR\%20DATA\%20SHEET\%202014.pdf>. Acesso em: 3 jul. 2018.

STOLYAROV, R. M.; BURNETT, G.; HERR, H. Translational Motion Tracking of Leg Joints for Enhanced Prediction of Walking Tasks. **IEEE Transactions on Biomedical Engineering**, v. 9294, n. c, p. 1–7, 2017. ISSN 15582531.

TENSORFLOW. **Premade Estimators**. 2018. Disponível em: <https://www.tensorflow.org/get_started/premade_estimators>. Acesso em: 3 jul. 2018.

TENSORFLOW. **TensorFlow In Use**. 2018. Disponível em: <https://www.tensorflow.org/about/uses>. Acesso em: 3 jul. 2018.

VAHID, F.; GIVARGIS, T. **Embedded System Design: A Unified Hardware/Software Introduction.** New York: Wiley, 2002. 1–324 p. ISBN 978-0-471-45303-1.

WAGNER, F. et al. **Modeling Software with Finite State Machines: Practical Approach**. [S.l.]: Auerbach Publications, 2006.

WHITE, E. **Making Embedded Systems: Design Patterns for Great Software**. [S.l.]: O'Reilly Media, 2011. ISBN 1449320589.

XIA, F. et al. Internet of things. **International Journal of Communication Systems**, v. 25, n. 9, p. 1101–1102, 2012. ISSN 10745351.