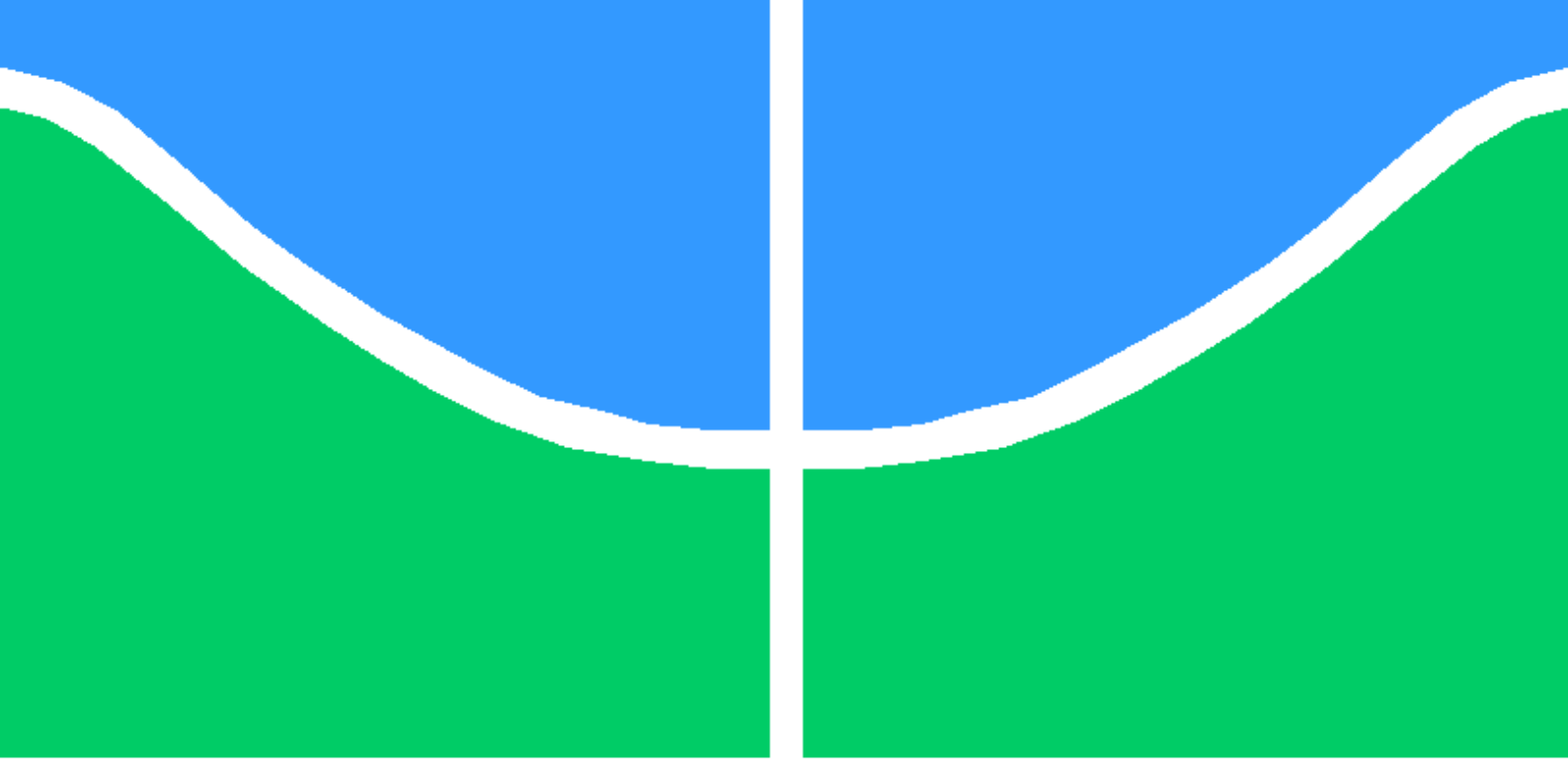


DETECÇÃO DE POSIÇÃO E QUEDAS CORPORAIS BASEADO EM K-MEANS CLUSTERING E THRESHOLD

LARINNI MALHEIROS

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DETECÇÃO DE POSIÇÃO E QUEDAS CORPORAIS BASEADO EM K-MEANS CLUSTERING E THRESHOLD

LARINNI MALHEIROS

Orientador: PROF. GEORGES DANIEL AMVAME NZE, DR., ENE/UNB

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO PPGENE.DM - 684/2017
BRASÍLIA-DF, 07 DE DEZEMBRO DE 2017.

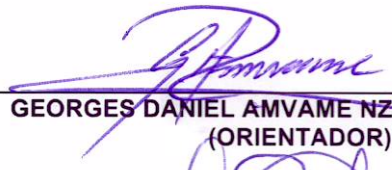
UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DETECÇÃO DE POSIÇÃO E QUEDAS CORPORAIS BASEADO EM K-MEANS CLUSTERING E THRESHOLD

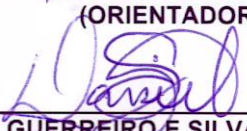
LARINNI MALHEIROS

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.

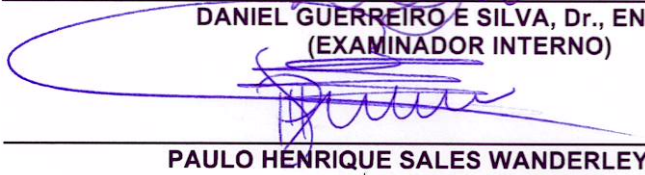
APROVADA POR:



GEORGES DANIEL AMVAME NZE, Dr., ENE/UNB
(ORIENTADOR)



DANIEL GUERREIRO E SILVA, Dr., ENE/UNB
(EXAMINADOR INTERNO)



PAULO HENRIQUE SALES WANDERLEY, Dr., IFB
(EXAMINADOR EXTERNO)

Brasília, 07 de dezembro de 2017.

FICHA CATALOGRÁFICA
LARINNI MALHEIROS
Detecção de Posição e Quedas Corporais Baseado em K-Means Clustering e Threshold
2017xv, 58p., 201x297 mm
(ENE/FT/UnB, Mestre, Engenharia Elétrica, 2017)
Dissertação de Mestrado - Universidade de Brasília
Faculdade de Tecnologia - Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

LARINNI MALHEIROS (2017) Detecção de Posição e Quedas Corporais Baseado em K-Means Clustering e Threshold. Dissertação de Mestrado em Engenharia Elétrica, Publicação 684/2017, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 58p.

CESSÃO DE DIREITOS

AUTOR: Larinni Malheiros
TÍTULO: Detecção de Posição e Quedas Corporais Baseado em K-Means Clustering e Threshold.
GRAU: Mestre ANO: 2017

É concedida à Universidade de Brasília permissão para reproduzir cópias desta dissertação de Mestrado e para emprestar ou vender tais cópias somente para propósitos acadêmicos e científicos. O autor se reserva a outros direitos de publicação e nenhuma parte desta dissertação de Mestrado pode ser reproduzida sem a autorização por escrito do autor.

_____

Larinni Malheiros
QMSW5 Lote 6 Sudoeste-Brasília.

Agradecimentos

Agradeço, primeiramente, ao meu orientador, o professor Dr. Georges Daniel Amvame Nze, que não poupou esforços e paciência para que o presente trabalho fosse realizado e por ter me ajudado a proporcionar essa realização profissional. Aos demais professores e funcionários do Departamento de Engenharia Elétrica da Universidade de Brasília.

À minha mãe, Ionara Pimentel Malheiros, pelo amor incondicional, pelo exemplo de vida, pelo carinho e, por fim, por ter me proporcionado todas as condições para a realização desta dissertação.

Ao meu pai, Juvaldo Dionisio dos Santos que sempre me apoiou emocionalmente e financeiramente durante todo o meu processo de educação, desde o maternal até esse Mestrado. Sem você, esse Mestrado não teria se realizado.

Às minhas irmãs, Tatianne Silva Santos e Thaleanne Dionisio, pelo companheirismo, pelo amor, pelo exemplo acadêmico e por todas as brincadeiras sobre a dificuldade de concluir o Mestrado. Obrigada por aliviar o nervosismo para concluir esse projeto.

Aos amigos Samara Amui, Maira Leite, Mario César, Wellington Pedro, Marcos Vinicius, que desde a graduação acompanharam o meu processo de crescimento pessoal e profissional. Com vocês, aprendi que universidade vai além do que se vê na sala de aula.

Por fim, agradeço a Deus, sem o qual nada seria possível. Larinni Malheiros

Resumo

A queda de idosos é caso de saúde pública em todo o mundo e esse assunto tem sido alvo de pesquisa e desenvolvimento tecnológico com objetivo de amenizar as consequências físicas e psicológicas para estas pessoas e seus familiares. Em 2017, 15,7% dos idosos no Brasil vivem sozinhos, de acordo com [1]. Há várias hipóteses para explicar essa tendência, entre elas, o desejo de autonomia e a dispersão e fragmentação familiar, com muitos filhos morando longe dos pais. Nesse contexto, este trabalho apresenta um dispositivo capaz de auxiliar a monitoração dos idosos em suas atividades, especialmente as domésticas.

Serão apresentados os fundamentos teóricos para o desenvolvimento do dispositivo. Os fundamentos teóricos apresentados abordam todas as fases de desenvolvimento do dispositivo, abrangendo desde a instalação da parte física até o desenvolvimento dos algoritmos utilizados para processar as informações.

Os desafios encontrados ao longo desse trabalho foram: precisão e adequação. A precisão do dispositivo é dividida em sensibilidade e especificidade. Ambas são parâmetros utilizados para determinar a acurácia do sistema. O desafio relacionado a essa atividade consistiu em avaliar se a acurácia do dispositivo é suficiente para fornecer a confiabilidade necessária para aplicações de detecção de quedas e posição corporais. Além disso, o dispositivo deve se adequar as características físicas do paciente que o utiliza, pois variáveis como altura, peso e idade influencia do resultado da predição. Será avaliado o desempenho do dispositivo utilizando vários cenários e sua aplicação no mundo real. Será apresentado o comparativo de resultados entre o dispositivo criado neste trabalho de Mestrado ao trabalho de Graduação [2].

Será apresentada uma metodologia baseada em aprendizado de máquina para realizar a predição das posições estáticas (sentado, deitado e em pé) e *threshold* para determinação de posições dinâmicas (andar e cair). Informações sobre essas posições fornecem resultados se o paciente encontra-se em queda, sendo essa uma posição que deve ser tratada imediatamente pelo cuidador. O algoritmo de aprendizado de máquinas utilizado é o *K-Means Clustering*, com o qual tem-se a posição estática que está sendo realizada pelo paciente. Uma série de condições de decisão baseadas em *thresholds* foram utilizadas para detectar posições dinâmicas como andar e cair. Para coletar as informações, será utilizado o sensor MPU6050 e para processamento e apresentação dos dados será utilizado o RaspberryPi. Os dados serão apresentados em uma aplicação Android e Web para monitoramento dos idosos através de

seus cuidadores.

Como resultado desse trabalho, observou-se que a detecção de quedas e posição corporais utilizando o aprendizado de máquinas para detecção de posições estáticas apresenta resultados confiáveis para a posição deitado e inferioridade estatística para diferenciar os movimentos como sentado e em pé. Em relação aos movimentos dinâmicos, verificou-se que é possível diferenciá-los utilizando parâmetros como regressão linear e área da integral entre o ponto de maior amplitude e o valor remanescente do vetor dos dados obtidos do sensor MPU6050.

Abstract

Fall Detection is a health issue in all over the world. This matter has been searched and developed in the technology field with the goal of decreased physical and phycological consequences to their families and themselves. There are some hypotheses to explain this trend, among them, the desire for independence and families dispersion and fragmentation, with sons and daughters living away from their parents. In this context, this work presents a device capable of auxiliary and monitors elderly in their activities, especially the domestic activities.

This work uses machine learning approach to predict static body position (standing, lying and sitting) and threshold to identify dynamic body position (walking and falling). The machine learning algorithm used in this work to detect static positions is K-Means Clustering. A series of decision conditions based on thresholds to detect dynamic movements such as walking and fall. To collect information will be used MPU6050 and to process and present the data will be used RaspberryPi.

As a result of this work, it is possible to conclude that fall and position detection using machine learning to detect static position presents reliable data to lying position and lower static data to differentiate sitting and standing positions. It is possible to differentiate dynamic movements trough linear regression and calculate the integer of the vector obtained from the MPU6050 sensor.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE TERMOS E SIGLAS

ADL *Activities of Daily Living*

BASIC *Beginner's All-Purpose Symbolic Instruction Code*

CSV *Comma-Separated Values*

CWT *Continuous Wavelet Transform*

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

DHCP *Dynamic Host Configuration Protocol*

DML *Data Manipulation Language*

DMP *Digital Motion Processor*

DWDT *Discrete Wavelet Transform*

GND *Ground*

GPIO *General Purpose Input/Output*

GPS *Global Positioning System*

HDMI *High –Definition Multimedia Interface*

HMM *Hidden Markov Models*

HTML *Hyper Text Markup Language*

I2C *Inter-Integrated Circuit*

IDE *Integrated Development Environment*

IOT *Internet Of Things*

IP *Internet Address*

KNN *K-Nearest Neighbors*

LCD *Liquid Crystal Display*

MATLAB *Matrix Laboratory*

MEMS *Microelectromechanical System*

PHP *Hypertext Preprocessor*

RAM *Random Access Memory*

RDBMS *Relational Database Management System*

RGB *Red, Green, Blue*

SCL *Serial Clock Line*

SD *Secure Digital*

SDA *Serial Data Line*

Spyder *Scientific Python Development Environment*

SVM *Support Vector Machines*

TCP *Transmission Control Protocol*

USB *Universal Serial Bus*

VCC *IC Power Supply*

WLAN *Wireless Local Area Networks*

WWAN *Wireless Wide Area Networks*

Capítulo 1

Introdução

1.1 Contextualização

À medida que as tecnologias de predição e mobilidade são desenvolvidas são realizados estudos para verificar a viabilidade de utilizá-las para detecção de queda e ADL (*Activities of Daily Living*) . Há duas diretrizes de estudo: detecção por *threshold* e, mais recentemente, está sendo utilizado o aprendizado de máquinas para realizar a predição.

A tecnologia de *threshold* é estudada a partir do valor de pico da magnitude dos vetores obtidos do acelerômetro e giroscópio [12] e o valor após o impacto para distinguir a queda dos movimentos diários [13]. Alguns movimentos diários são confundidos com quedas, tais como correr, andar e pular, então existe a necessidade de um outro parâmetro para distinguir o movimento [14]. Esse outro parâmetro pode ser, por exemplo, o tempo remanescente após a queda [15] ou o valor da integral de segunda ordem [16]. A detecção de quedas pode ser categorizada em quatro topologias: baseado em monitoração visual, baseado em sinais do ambiente, sistema baseado em contexto e sistema baseado em equipamentos vestíveis.

Os sistemas de monitoração visual consiste em utilizar câmeras e dispositivos do tipo *kinect* instaladas em posições mais altas do que os usuários, para acompanhar e caracterizar o movimento dos mesmos. Diferentes técnicas de análise de imagem têm sido propostas, tais como a modelagem utilizando características espaço-temporais, reconhecer o formato do corpo do usuário e o mesmo detectar alterações de postura, análise 3D do movimento da cabeça, entre outros. No trabalho [17] é abordada essa metodologia para detectar quedas baseado em *depth images*. O *kinect* é um dispositivo do tipo RGB-D, que permite a captura de imagens representando as cores (*Red*, *Green*, *Blue*) e a profundidade de uma cena, essa profundidade é chamada de *depth image* [18] . O trabalho apresentado em [19] utiliza um dispositivo *kinect* para capturar imagens de 20 articulações do corpo e envia essas informações para o algoritmo KNN (*K-Nearest Neighbors*) e SVM (*Kernelized Support Vector Machines*) para obter a análise dos movimentos, esses resultados são então aplicados em uma rede de IoT (*Internet of Things*) .

Os sistemas baseados em ambiente são baseados em sinais audiovisuais juntamente com outras informações capturadas pelos sensores instalados no ambiente. Nesse caso, as quedas são identificadas através da comparação entre dados de sensores que capturam a vibração do solo e dados de sensores que capturam sinais sonoros com padrões pré-definidos de valores obtidos desses sensores, correspondendo a diversas atividades como caminhar, correr, queda de pequenos objetos entre outros. No trabalho [20] é apresentado um sistema de sensores instalados no solo que coleta as informações e realiza a análise através do algoritmo SVM, o artigo [20] mostra que esses sensores fornecem resultados mais precisos do que o uso de sensores que capturam sinais sonoros.

Os sistemas baseados em contexto são baseados em sensores que realizam a coleta de imagens do ambiente [21]. Esses sistemas tem a vantagem de serem métodos não invasivos, contudo os mesmos apresentam várias desvantagens:

1. O monitoramento do paciente tem que ser realizado em uma área confinada;
2. A instalação, ajustes e manutenção dos equipamentos são de alto custo;
3. A acurácia dos sistemas de detecção de quedas é determinada por condições não controláveis, tais como, iluminação e presença de obstáculos visuais;
4. Os pacientes podem não ser favoráveis ao fato de ter uma câmera comprometendo sua privacidade.

Sistemas de detecção de quedas baseados em equipamentos vestíveis utilizam informações capturadas pelos sensores via redes WLAN (*Wireless Local Area Networks*) , WWAN (*Wireless Wide Area Networks*) e com o auxílio das informações oriundas do GPS (*Global Positioning System*) . Esses sensores frequentemente são acelerômetros e giroscópios, geralmente acoplados ao corpo. Esse sistema não restringe a mobilidade do paciente.

A mais recente abordagem é o uso da aprendizagem de máquinas para detectar os movimentos do paciente. Essa metodologia se uniu às baseadas em monitoração visual, sinais do ambiente, contexto e equipamentos vestíveis, aprimorando as descobertas de detecção baseada em *threshold*. No trabalho [22] é apresentada a utilização de Rede Neural Artificial para o reconhecimento de diferentes posturas utilizando o dispositivo *kinect* para capturar imagens baseadas em RGB (*Red*, *Green*, *Blue*) . O aprendizado de máquinas se divide em três vertentes: supervisionado, não-supervisionado e por reforço.

O aprendizado supervisionado apresenta alguns algoritmos conhecidos na academia, tais como: *K-Nearest Neighbor* (KNN), *Linear Models*, *Naive Bayes Classifiers*, *Decision Tree*, *Support Vector Machines* (SVM) e *Neural Networks* . O aprendizado não-supervisionado apresenta algoritmos baseados em *clustering*, tais como: *K-Means Clustering*, *Agglomerative*, DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [4]. O aprendizado de máquinas por reforço é entendido como o problema encontrado por um agente que

deve aprender como se comportar em um ambiente dinâmico através de interações do tipo "tentativa e erro" [23].

O serviço de detecção de quedas pode ser ampliado para o contexto de IoT. Sendo um dispositivo capaz de observar o ambiente ao seu redor, prever respostas e enviar as informações para a nuvem. Esse conceito apresentado por [24] generaliza todas as abordagens citadas acima.

Dois termos técnicos são utilizados em todos os artigos apresentados acima para definir se uma queda foi corretamente detectada. O primeiro termo é chamado de "falso positivo" e o segundo é chamado de "falso negativo". Falso positivo é um conceito que indica quando a queda foi erroneamente detectada, pois o paciente estava realizando outra atividade, como andar, correr ou levantar rapidamente de uma cadeira. Falso negativo indica quando uma queda não foi detectada. Como demonstrado acima, existe uma série de tecnologias sendo desenvolvidas para auxiliar o processo de detecção de posição e quedas. O desenvolvedor deverá verificar qual é a mais adequada para a sua aplicação. Este trabalho irá utilizar a metodologia de aprendizado não-supervisionado baseada no algoritmo *K-Means Clustering* para determinar se a posição do paciente é estática ou dinâmica, posteriormente será utilizado um conjunto de condições que utilizará *thresholds* para identificar qual posição dinâmica está sendo realizada. Esse trabalho faz complemento aos artigos acima citados, apresentando as informações em tempo real utilizando uma aplicação Android e uma página web.

1.2 Definição do Problema

O crescimento da população idosa tem motivado pesquisadores a desenvolver sistemas de cuidados à saúde para garantir a segurança das pessoas mais velhas durante suas atividade rotineiras, como as domésticas.

Quedas de idosos e suas consequências, tais como fraturas, são um grande risco, especialmente para os idosos que vivem sozinhos, necessitando assim de assistência imediata. De acordo com o trabalho [25] há uma estimativa de que 25% dos americanos, com 65 anos ou mais, cai a cada ano. A cada 11 segundos, um idoso é tratado no pronto socorro devido a uma queda. A cada 19 minutos, um idoso morre em função de uma queda. De acordo com o trabalho [25], em 2013, o custo total de lesões por queda foi de US$ 34 bilhões. Portanto, a detecção de posição dos idosos se tornou um importante objeto de pesquisa.

Sistemas de detecção automática de quedas baseados em acelerômetros e giroscópios podem reduzir o risco de complicações na saúde dos idosos através de um sistema que identifica a queda e rapidamente realiza o alerta aos cuidadores sobre o evento. O desafio é desenvolver dispositivos que podem detectar quedas com acurácia suficiente para ter confiabilidade no sistema, ao mesmo tempo em que se fornece uma estrutura não intrusiva e confortável para permitir o seu uso.

A maior barreira no desenvolvimento de dispositivos de detecção de quedas é a falta de evidência na acurácia desses dispositivos, pois a efetividade é avaliada em laboratório com jovens adultos que apresentam padrões de movimentos diferentes de idosos, além dos mesmos realizarem quedas em superfícies macias como um colchão, não representando o cenário real que é, por exemplo, no chão [26].

Os desenvolvimentos recentes em mobilidade e comunicação aumentaram as estatísticas para detecção de posição e quedas. O avanço tecnológico no desenvolvimento de sensores compactos e microcontroladores com habilidade de rápido processamento, permitiram o desenvolvimento de algoritmos complexos com objetivo de resolver esse problema.

1.3 Objetivos

1.3.1 Objetivos

Este trabalho visa desenvolver um dispositivo capaz de diferenciar posições estáticas e dinâmicas utilizando o aprendizado não-supervisionado através do algoritmo *K-Means Clustering* e identificar a posição dinâmica a ser realizada utilizando uma série de decisões baseadas em *thresholds*.

1.3.2 Objetivos Específicos

Esse trabalho de mestrado é realizado com objetivo de adicionar funcionalidades e melhorar os resultados obtidos no sistema implementado em [2]. No sistema [2], é criado um dispositivo com dois componentes, um acoplado à coxa e outro no peito, utilizando *thresholds* baseados na variação da amplitude dos vetores obtidos do acelerômetro e do giroscópio. No projeto de mestrado, o primeiro objetivo é retirar o componente da coxa para fornecer maior aplicabilidade em cenário real, o segundo objetivo é utilizar aprendizado de máquinas para fornecer maior acurácia ao sistema e o terceiro objetivo é inserir calibração ao algoritmo para que o mesmo seja adequado para as diversas características físicas do corpo humano. Além disso, a dissertação irá apresentar os resultados obtidos e comparar com os desenvolvimentos recentes. Também será avaliado a sua aplicabilidade no ambiente real.

1.4 Organização do Trabalho

Esta dissertação está dividida em cinco capítulos. O presente capítulo 1, traz a contextualização, os objetivos e organização do trabalho.

O capítulo 2 introduz a fundamentação teórica, as tecnologias utilizadas para realizar esse trabalho e sua aplicabilidade no contexto atual de detecção de posição.

O capítulo 3 trata da metodologia. Será apresentado como foi desenvolvido o ambiente web, Android e embarcado.

O capítulo 4 apresenta os resultados obtidos, assim como, sua análise técnica e efetividade do dispositivo.

O capítulo 5 irá apresentar as conclusões e propostas de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Aprendizado de Máquina

Nos últimos anos, a pesquisa em aprendizado de máquina teve um crescimento notório. Aprendizado de máquina é uma área da inteligência artificial cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática. Um sistema de aprendizado é um programa de computador que toma decisões baseadas em experiências acumuladas por meio da solução bem-sucedida de problemas anteriores [27]. O aprendizado de máquina é o resultado da interseção entre estatística, inteligência artificial e ciência da computação para reconhecimento de padrões [4].

A aplicação de aprendizado de máquinas está presente no dia-a-dia através de páginas web como Facebook, Amazon e Netflix. Sua aplicabilidade varia desde a recomendação automática de filmes e música, até o reconhecimento de pessoas nas fotos [4].

Além do objetivo comercial, o aprendizado de máquina teve uma gigante influência na forma como são realizadas as pequisas atuais, como por exemplo, análise de sequências de DNA e fornecimento de tratamento personalizado para o câncer [4].

Há alguns conceitos que são comuns aos algoritmos de aprendizado de máquina, como por exemplo o *overfitting* e *underfitting*.

• *Overfitting*: se refere à situação em que o algoritmo gera um classificador que se ajusta aos dados de treinamento, mas perdeu a capacidade de generalização para dados novos. Em outras palavras, em vez de aprender, o classificador apenas memorizou os dados de treinamento [28]. Essa situação pode ocorrer quando a rede tem mais parâmetros do que necessário para a resolução do problema [27].

• *Underfitting*: se refere à situação em que o algoritmo gera um classificador que não se ajusta nem aos dados de treinamento nem aos novos [28]. Ocorre quando a rede tem menos parâmetros do que necessário para a resolução do problema [27].

A Figura 2.1 apresenta o comportamento de *overfitting* e *underfitting*. É possível verificar que a complexidade do modelo diminui o erro de predição até um ponto em que é adicionado apenas ruídos aos dados coletados. A complexidade do modelo está relacionada com a quantidade de parâmetros, ou características, dos dados a serem determinados. Ao desenvolver um sistema que utiliza aprendizado de máquina, o algoritmo deve abordar esses conceitos e encontrar o valor ótimo da aplicação para evitar *overfitting* e *underfitting*.

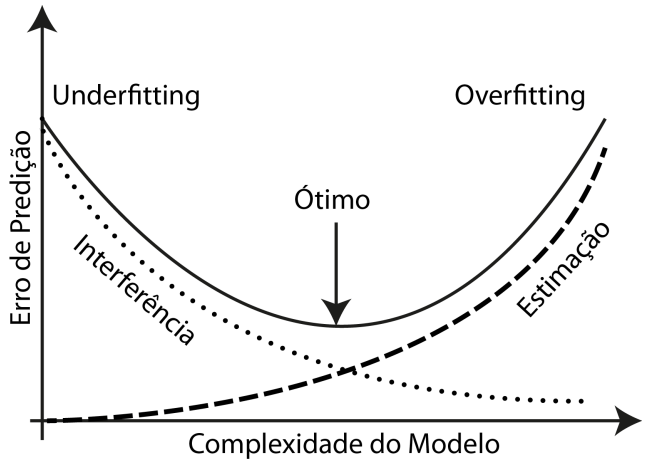


Figura 2.1: Overfitting e Underfitting. Adaptado de [3]

A fim de evitar *Overfitting* e *Underfitting* e validar a construção do algoritmo, existe uma técnica chamada Validação Cruzada, que realiza a análise de performance dos dados de treinamento. Essa análise de performance varia de acordo com o método a ser utilizado.

A Validação Cruzada é utilizada apenas em algoritmos utilizando o aprendizado supervisionado, pois esses são os únicos algoritmos capazes de avaliar os resultados obtidos com o que foi ensinado para a máquina. Sendo assim, neste projeto de mestrado não foram utilizados métodos de validação, pelo fato do mesmo utilizar apenas algoritmos baseados em aprendizado não-supervisionado [29].

2.1.1 Aprendizado Supervisionado - *Supervised Learning*

O aprendizado supervisionado compreende a abstração de um modelo de conhecimento a partir dos dados apresentados na forma de pares ordenados (entrada, saída desejada). Por entrada entende-se o conjunto de atributos ou características de entrada do algoritmo para um determinado caso. A saída desejada corresponde ao valor de uma característica-alvo que se espera que o algoritmo possa produzir sempre que receber os valores especificados em entrada [30].

Alguns algoritmos que compõe esse modelo são: K-NN, Modelos Lineares, Classificador Naive Bayes, *Support Vector Machines* e Rede Neural Artificial.

Esse trabalho de mestrado utilizou apenas o aprendizado não-supervisionado, por isso a seção referente ao mesmo será descrita com mais detalhes.

2.1.2 Apresendizado Não-Supervisionado - *Unsupervised Learning*

Caracteriza-se pela não existência de saídas desejadas para as entradas, sendo o conjunto de treinamento formado apenas por vetores de entrada [31]. Não há saída conhecida, portanto, não é possível ensinar a máquina qual resultado é esperado, mas sim, esperar que ela agrupe os dados pelas suas características.

O maior desafio do modelo não-supervisionado é avaliar se o algoritmo realizou a clusterização de dados com características similares. Modelos não-supervisionados são aplicados aos dados que não contém um rótulo, portanto, sua saída é desconhecida. Devido a este motivo é difícil dizer se o modelo realizou uma predição correta das informações. Uma aplicação comum desse modelo é ser um passo anterior ao modelo supervisionado, pois o não-supervisionado pode ser adicionado ao supervisionado e a acurácia do mesmo pode aumentar [32].

Clustering é a tarefa de particionar o conjunto de dados em grupos, chamados *clusters*. O objetivo é dividir os dados de tal forma que os pontos dentro de um mesmo *cluster* são muito similares e pontos em diferentes *clusters* são distintos [4].

2.1.2.1 *K-Means Clustering*

K-Means Clustering é uma técnica que realiza a clusterização através do método de particionamento [6]. Esse método consiste em construir várias partições dos dados e as avalia utilizando algum critério. O algoritmo *K-Means* busca encontrar o centro de regiões que representam determinados tipos de dados. O algoritmo varia entre atribuir a cada ponto o centro mais próximo, e escolher o centro do *cluster* como a média dos pontos que foram atribuídos a ele. O algoritmo termina quando as atribuições ao *cluster* não se alteram, momento em que é atingido o ponto de convergência do mesmo [4].

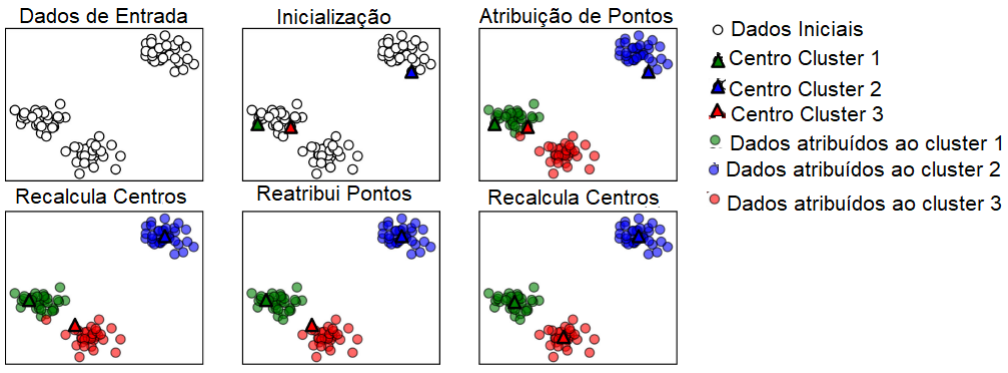


Figura 2.2: K-Means Clustering. Adaptado de [4]

A Figura 2.3 apresenta o pseudo-código do algoritmo *K-Means Clustering*. No primeiro passo, o algoritmo inicia declarando o valor "k " de centros ("*Inicialização*"), mostrado na Figura 2.2. À partir dessa etapa, o algoritmo iterativo inicia. Os pontos são atribuídos para os centros mais próximos ("*Atribuição de Pontos*"), depois o centro dos *clusters* são atualizados

```
1    Defina o número k de conjunto a ser usado
2    Designe cada ponto para um conjunto aleatoriamente
3    Repita
4          Calcule o centro de cada conjunto
5           Reorganize os elementos, designando-os para o conjunto
          cujo centro lhe for mais próximo
6    Até que nenhum elemento mude de conjunto
```

Figura 2.3: Pseudo-Código *K-Means Clustering*. Retirado de [5]

para serem a média dos pontos que foram atribuídos para ele ("*Recalcula Centros*"). O algoritmo *K-Means* termina quando a atribuição de pontos ao centro fica estável.

Para atribuir um ponto ao seu centro mais próximo, é necessária uma medida que quantifica a proximidade entre eles. A distância euclidiana é geralmente utilizada para pontos em um espaço euclidiano, que é um espaço de dimensão finita, no qual está definido um produto interno.

No passo 4 do pseudo-código, o algoritmo recalcula o centro para cada *cluster*, porque o valor do centro pode variar dependendo da medida de proximidade dos dados e do objetivo da clusterização. O objetivo da clusterização é expressa através de uma função que depende da proximidade entre os pontos ou entre os pontos e seu respectivo centro, por exemplo minimizar o quadrado da distância de cada ponto até o centro mais próximo. Uma vez especificada a medida de proximidade e uma função objetivo, o centro que deve ser escolhido é determinado matematicamente. A equação 2.2 descreve o cálculo utilizado na determinação do centro.

Considere dados cuja medida de proximidade é a distância euclidiana. A função objetivo, que mede a clusterização correspondente aos dados, utiliza a soma dos quadrados do erro residual (SSE), também chamada de dispersão. É calculada a distância euclidiana de cada ponto até o centro mais próximo, então é calculado o valor da soma dos quadrados do erro residual. A melhor escolha para o centro é aquele que fornece menor valor de dispersão, ou seja, o centro que fornece melhor representação para os dados em seu *cluster*.

Utilizando a notação da Tabela 3.2, o valor da soma dos quadrados do erro residual (SSE) é fornecido pela equação 2.1:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2 \tag{2.1}$$

Na qual *dist* é a distância euclidiana entre dois objetos em um espaço euclidiano. O valor do centro que minimiza o SSE é o valor da média entre os pontos atribuídos a esse centro. O valor do centro do *cluster* i^{th} é definido pela equação 2.2:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x \tag{2.2}$$

Os passos 4 e 5 do algoritmo são iterativos na tentativa de minimizar o valor SSE. O passo 4 forma *clusters* atribuindo pontos para o centro mais próximo a fim de minimizar a dispersão e o passo 5 recalcula os centros para minimizar a dispersão.

Quando inicializações randômicas de centros são realizadas, diferentes execuções do *K-Means* resulta em diferentes valores de SSE. Para evitar esses valores aleatórios, é fornecido um valor de *seed* à função *K-Means* do Python. Para ilustrar como é realizado o cálculo do centro do *cluster*, considere os seguintes pontos de duas dimensões: (1,1), (2,3), e (6,2). O valor do centro é calculado como $\frac{1+2+6}{3}$, $\frac{1+3+2}{3}$ = (3,2). Esse cálculo é realizado utilizando a equação 2.2.

O nome do algoritmo, *K-Means*, representa que são utilizados "k " valores declarados como centros e os valores dos mesmos são representados pela média (*means*) dos pontos do *cluster* atribuídos a eles. As equações 2.3 e 2.4 mostram que o valor do centro que minimiza o SSE de um *cluster* é determinado através da média dos pontos dos *clusters* [6]. Essas equações foram desenvolvidas baseadas na notação da Tabela 2.1.

Tabela 2.1: Notação de equações para o cálculo de centro do cluster e dispersão

Símbolo	**Descrição**
X	dado
C_i	Cluster i^th
c_i	Centro do cluster C_i
c	Centro de todos os pontos
m_i	Número de dados no cluster i^th
m	Número de dados no conjunto de dados
K	Número de clusters

$$\frac{\partial}{\partial c_k} SSE = \frac{\partial}{\partial c_k} \sum_{i=1}^{K} \sum_{x \in C_i} (c_i - x)^2 = \sum_{i=1}^{K} \sum_{x \in C_i} \frac{\partial}{\partial c_k} (c_i - x)^2 = \sum_{x \in C_k} 2 * (c_k - x_k) = 0 \quad (2.3)$$

$$\sum_{x \in C_k} 2 * (c_k - x_k) = 0 \implies m_k c_k = \sum_{x \in C_k} x_k \implies c_k = \frac{1}{m_i} \sum_{x \in C_i} x_k \quad (2.4)$$

2.1.2.2 *Agglomerative Clustering*

Agglomerative Clustering é uma técnica que realiza a clusterização através do método hierárquico [6]. O pseudo-código do algoritmo está descrito na Figura 2.4. O algoritmo inicia com a declaração de cada ponto como seu próprio *cluster*. Posteriormente ele se junta com os dois *clusters* mais similares até que alguma condição seja satisfeita. A condição mais comum é a de quantidade de *clusters* remanescentes.

A Figura 2.5 representa o algoritmo. No começo, cada ponto é seu próprio *cluster*; posteriormente, em cada passo os dois *clusters* mais próximos se unem. Nos primeiros quatro passos, dois pontos se unem para formar um *cluster*. No passo 5, um dos *clusters* de dois pontos se estende para um terceiro ponto e assim por diante. No passo 9, existem apenas três *clusters* remanescentes. Se esse for o objetivo especificado o algoritmo finaliza. *Agglomerative Clustering* não pode realizar predições para novos dados devido ao modo em que o algoritmo realiza os cálculos [4].

1. Calcule a matriz de proximidade
2 .Determine que cada ponto é um cluster
3. Repita
4. Combina os dois clusters mais próximos
5. Atualiza a matriz de proximidade
6. Até que apenas um cluster seja remanescente

Figura 2.4: Pseudo-Código *Agglomerative Clustering*. Adaptado de [6]

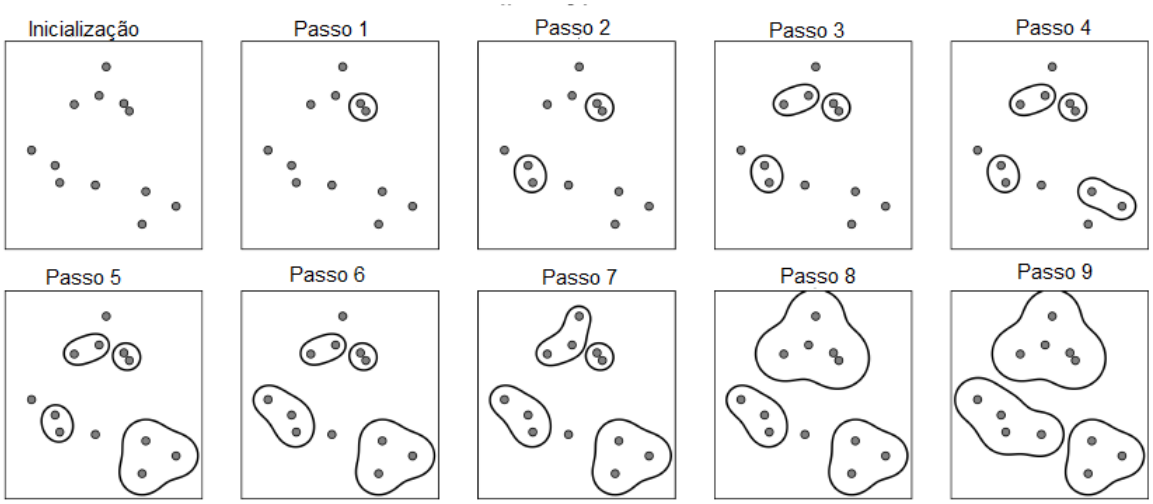


Figura 2.5: Agglomerative Clustering. Adaptado de [4]

2.1.2.3 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) é outro algoritmo de clusterização muito utilizado. Uma de suas vantagens é que ele não necessita que o usuário determine o número de *clusters*. Ele forma *clusters* de diferentes dimensões ao identificar um ponto que não é membro de nenhum *cluster*. A desvantagem é que ele é mais lento do que *Agglomerative Cluster* e *K-Means* [4].

2.1.2.4 Conclusão

Nesse trabalho, será utilizado o algoritmo *K-Means* para realizar a clusterização das posições estáticas e dinâmicas. Decidiu-se utilizar o *Unsupervised Learning* para realizar a

predição de novos dados baseados em características que foram aprendidas por dados dinamicamente coletados. Dentre essas caraterísticas, apenas o *K-Means* era adequado para essa aplicação, pois o DBSCAN e *Agglomerative Cluster* não apresentam a mesma quantidade de recursos bibliográficos como o *K-Means*, além do último ser mais simples e rápido [6]. Algumas vantagens desse algoritmo em relação aos demais algoritmos de aprendizado não-supervisionado são descritas abaixo [33]:

1. O algoritmo *K-Means clustering* é o mais simples;
2. Todos os algoritmos de aprendizado não-supervisionado apresentam desvantagens em alguns aspectos;
3. DBSCAN não é adequado para dados com grande variância de densidade;
4. O algoritmo *K-Means* é mais adequado para grandes conjunto de dados;

A saída do algoritmo de aprendizado não *K-Means* é o rótulo predito indicando em qual dos *cluster* determinados pelo algoritmo o dado coletado pertence. Esse rótulo é aplicado em uma sequência de "if-else" com *thresholds* para determinar a posição dinâmica do paciente. Esse processo foi escolhido em vez de utilizar aprendizado supervisionado porque foi de interesse desse projeto analisar os dados utilizando a clusterização e o conjunto de dados obtidos do paciente não eram suficientes para compor um conjunto de dados capaz de fornecer confiabilidade para a construção de um modelo de aprendizado supervisionado.

2.2 Android Studio

Android Studio é a IDE (*Integrated Development Environment*) oficial para desenvolvimento de aplicativos Android baseado em IntelliJ, na qual a expectativa é substituir o Eclipse. Sendo assim, toda a interface textual do software é baseado em Java. Ela fornece interface gráfica e textual para o usuário realizar o design gráfico do aplicativo. Android Studio é desenvolvido pela Google em colaboração com JetBrains [34].

O Projeto de Graduação [2] utilizou o programa "*Basic For Android*" para desenvolver o aplicativo Android, enquanto que esse trabalho de mestrado utilizou a IDE Android Studio. Android Studio foi utilizado por ser um software gratuito e possuir uma interface com o usuário mais interativa, enquanto que o "Basic For Android" é um software pago e a linguagem de programação utilizada é Basic (*Beginner's All-Purpose Symbolic Instruction Code*) , cujo uso não é comum nas aplicações Android.

2.3 Socket TCP aplicado à comunicação Raspberry/Android

É possível realizar uma comunicação entre a aplicação cliente Android sendo executada em um dispositivo móvel e uma aplicação servidor executada no RaspberryPi através de portas TCP (*Transmission Control Protocol*) predeterminadas. O diagrama da Figura 2.6 apresenta o esquemático da camada de rede e suas iterações.

O servidor fica frequentemente escutando na porta designada para conexão com o dispositivo cliente. Quando o cliente envia uma requisição ao servidor, uma conexão é estabelecida, ocorrendo a troca de informações. A porta TCP utilizada para abrir a conexão é a "8888", pois essa é a porta utilizada para aplicações em tempo-real. Por ser uma conexão TCP, a mesma é confiável. O IP utilizado pelo RaspberryPi é o "192.168.0.5" fixo, associado ao MAC do mesmo. O IP associado à aplicação cliente é o mesmo do dispositivo móvel; esse IP é obtido através do DHCP (*Dynamic Host Configuration Protocol*) . A topologia que foi utilizada para realizar a comunicação entre a aplicação servidor e a cliente é a descrita na Figura 2.6. Esse assunto será descrito com mais detalhes na seção 3.5.

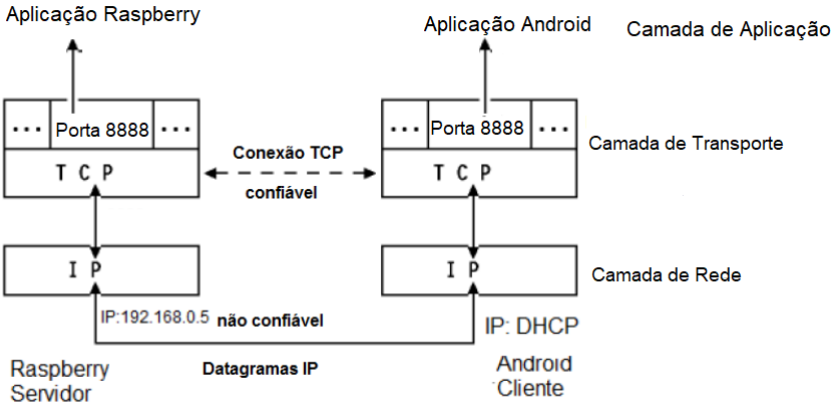


Figura 2.6: Camadas de Comunicação. Adaptado de [7]

2.4 Desenvolvimento Web

2.4.1 HTML

O HTML (*Hyper Text Markup Language*) é uma linguagem para arquivos de texto contendo pequenas etiquetas de marcação, essas são utilizadas pelo navegador Web para mostrar como a página deve ser exibida [35].

2.4.2 PHP

O PHP (*Hypertext Preprocessor*) é uma linguagem de script *open source* de uso geral, muito utilizada especialmente para o desenvolvimento de páginas dinâmicas. PHP é uma linguagem de *scripting*. O código PHP é interpretado no servidor Web quando um documento HTML no qual ele está embutido for requisitado por um navegador. O código PHP normalmente produz código HTML como saída, o que substitui o ducumento HTML [36].

2.4.3 CSS

O CSS (*Cascading Style Sheets*), que é traduzido para o português como folhas de estilo em cascata. O CSS tem por finalidade indicar como uma determinada marcação HTML deve ser exibida, como, por exemplo, fontes, cores, margens, linhas, alturas, larguras, imagens de fundo, posicionamento, entre outros [37]. O mesmo é suportado por todos os navegadores atuais. O HTML é utilizado para estruturar conteúdos, enquanto que o CSS formata os conteúdos estruturados fornecidos pelo HTML.

2.4.4 MySQL

O MySQL é o gerenciador de banco de dados relacional de código aberto mais popular do mundo e possibilita a entrega econômica de aplicativos de banco de dados confiáveis, de alto desempenho e redimensionáveis.

Banco de dados é uma aplicação separada que armazena uma coleção de dados. Cada banco de dados tem uma ou mais APIs (*Application Programming Interface*) diferente para criar, acessar, gerenciar, pesquisar e replicar os dados que o mesmo contém. Outros tipos de armazenamento de dados também podem ser utilizados, tais como arquivos no sistema de arquivos ou grandes tabelas *Hash* na memória, mas a busca e escrita de dados não seria tão rápida e fácil com esses tipos de sistemas. Então, atualmente, utiliza-se o RDBMS (*Relational Database Management Systems*) para armazenar e gerenciar grandes volumes de dados. Isso é chamado de banco de dados relacional, porque todos os dados são armazenados em diferentes tabelas, e relações são estabelecidas usando chaves primárias ou chaves estrangeiras [38].

Um RDBMS permite a implementação de um banco de dados utilizando tabelas, colunas e índices. Ele atualiza os índices automaticamente, garante a integridade referencial entre linhas de várias tabelas, interpreta um *query* SQL e combina informações originadas de várias tabelas [38].

Alguns conceitos são essenciais para entender o MySQL:

- Banco de dados: um banco de dados é uma coleção de tabelas, com dados relacionados;
- Tabela: uma tabela é uma matriz com os dados. Uma tabela em um banco de dados é

similar a uma planilha;

• Coluna: uma coluna contém os mesmos tipo de dados, como por exemplo o CEP;

• Linha: a linha (tupla, a entrada ou registro) é um grupo de dados relacionados, ou seja, compartilham alguma característica;

• Redundância: ocorre quando uma determinada informação é representada no sistema repetidas vezes de forma a proporcionar maior disponibilidade de informações;

• Chave primária: é uma restrição adicionada à coluna de uma tabela do banco de dados, de forma a identificar cada linha como sendo única. Isso garante que os valores em cada linha em determinada coluna são únicos e não se alteram. Esses valores podem ser utilizados para referenciar determinada linha;

• Chave estrangeira: uma chave estrangeira é uma referência, em uma tabela, a uma chave primária de outra tabela. Ela faz com que ocorra o relacionamento entre duas tabelas;

• Índice: um índice é um ponteiro para uma informação na tabela;

• Integridade Referencial: integridade referencial garante que um valor de chave estrangeira em uma tabela de destino sempre aponta para a chave primária de algum registro na tabela referenciada pela chave estrangeira.

MySQL é um RDBMS utilizado por muitas pequenas e grandes empresas, entre as razões estão: é liberado sob uma licença de código aberto, utiliza um formulário padrão da língua de dados do SQL, funciona em muitos sistemas operacionais e em muitas linguagens de programação, incluindo PHP, Python, Perl, C, C++, Java, etc, funciona rapidamente mesmo se apresentar um grande conjunto de dados, suporta grandes bancos de dados, até 50 milhões de linhas ou mais em uma tabela, é customizável.

O MySQL contém um objeto chamado *trigger*. Ele é associado a uma tabela e permite a realização de processamentos em consequência de uma determinada ação como, por exemplo, a inserção de um registro. *Triggers* são executados ANTES ou DEPOIS das operações de DML (*Data Manipulation Language*) . A Figura 2.7 apresenta as principais operações da linguagem MySQL.

```
1 INSERT, DELETE e UPDATE.
2 CREATE TRIGGER nome momento evento
3 ON \textit{tabela}
4 FOR EACH ROW
5 BEGIN
6     /*corpo do codigo*/
7 END
```

Figura 2.7: Linguagem DML

Nesse trabalho foi utilizado o MySQL para relacionar as bases de dados dos arquivos gerados nesse sistema. O MySQL também apresenta conexão com a linguagem de programa-

ção Python, que foi a utilizada para o desenvolvimento do protótipo. Além disso, utilizando o MySQL, facilmente é possível recuperar as informações da plataforma e apresentá-las na interface web, cujo desenvolvimento teórico é descrito nas seções subsequentes.

2.5 Desenvolvimento do Sistema Embarcado

Um sistema embarcado, também conhecido como embutido, é um sistema micro processado no qual um tipo de computador é completamente encapsulado em um dispositivo, ou sistema, controlado por ele. Ao contrário de computadores de uso geral, como os computadores pessoais, um sistema embarcado realiza um conjunto de tarefas previamente definidas, e com fins específicos [39].

2.5.1 Python

Python é uma linguagem dinâmica, interpretada, robusta, multiplataforma, multiparadigma (orientação à objetos, funcional, refletiva e imperativa) [40]. Lançada em 1991 por Guido van Rossum, é uma linguagem livre (até para projetos comerciais) e hoje pode-se programar para desktops, web e mobile [41]. São características da linguagem:

• baixo uso de caracteres especiais, o que torna a linguagem muito parecida com pseudocódigo executável;

• o uso de identação para marcar blocos;

• quase nenhum uso de palavras-chave voltadas para a compilação;

• coletor de lixo para gerenciar automaticamente o uso da memória;

A *Scikit-learn* é uma biblioteca de aprendizado de máquina de código aberto para a linguagem de programação Python. Ela inclui vários algoritmos de classificação, regressão e agrupamento incluindo SVM, *Decision Tree*, *K-means* e DBSCAN, e é projetada para interagir com as bibliotecas Python numéricas e científicas como *NumPy* e *SciPy*.

Para realizar as simulações desse projeto, foi utilizado o software Spyder (*Scientific Python Development Environment*) , que é um ambiente de desenvolvimento Python na qual o *layout* e aparição são similares ao MATLAB (*Matrix Laboratory*) [42].

2.5.2 CronJob

CronJob é uma ferramenta fornecida pelo sistema operacional Linux, que permite a realização de agendamento de tarefas programadas em dias e horários determinados pelo programador.

Para executar as tarefas, o cron utiliza uma espécie de tabela conhecida como crontab. O

arquivo crontab normalmente fica localizado no diretório "/etc" do sistema operacional, mas também pode estar em um diretório específico para cada usuário do sistema.

Para configurar tarefas cron, primeiramente deve-se abrir o crontab, utilizando o comando “crontab – e” e o configurar de acordo com a sintaxe abaixo:

[minutos] [horas] [dias do mês] [mês] [dias da semana] [usuário] [comando]

Minutos: números de 0 a 59; Horas: números de 0 a 23; Dias do mês: números de 0 a 31; Mês: números de 1 a 12; Dias da semana: números de 0 a 7; Usuário: informa o usuário que irá executar o comando (não é necessário especificá-lo caso o arquivo do próprio usuário for utilizado); Comando: a tarefa que deve ser executada

No lugar destes valores pode-se colocar “*” para especificar uma execução constante, por exemplo, se o campo “Dias do mês” conter “*”, o comando será executado todos os dias [43].

Nesse trabalho foi utilizado o Cronjob para executar a cada 0,5 segundos o algoritmo e detecção de posição escrito na linguagem Python. Esse algoritmo é responsável por atualizar as informações na interface web e no aplicativo Android.

2.6 Acelerômetro e Giroscópio (MPU6050)

O sensor InvenSense MPU6050, mostrado na Figura 2.8, integra três graus de liberdade para o giroscópio, três graus de liberdade para o acelerômetro e um DMP (*Digital Motion Processor*) no mesmo chip.

O MPU6050 contém um hardware de conversão analógico-digital de 16 bits para cada canal, logo ele coleta os canais x,y e z ao mesmo tempo. O sensor é baseado na tecnologia MEMS (*Microelectromechanical System*) e utiliza o protocolo de comunicação I2C (*Inter-Integrated Circuit*) para transmissão de dados entre o RaspberryPi utilizado nesse projeto e o sensor MPU6050. A detecção de movimentos rápidos ou lentos pode ser diferenciada com a programação de um intervalo de escala fornecido pelo equipamento. O giroscópio apresenta um intervalo de $\pm 250^o/s$, $\pm 500^o/s$, $\pm 1000^o/s$ e $\pm 2000^o/s$ e o acelerômetro apresenta um intervalo de $\pm 2g$, $\pm 4g$, $\pm 8g$ e $\pm 16g$ [44].

Figura 2.8: MPU6050. Extraído de [8]

O protocolo I2C trabalha no modelo mestre-escravo, com pelo menos um dispositivo atuando como mestre, e os demais dispositivos atuando como escravo. A função do mestre

é coordenar a comunicação entre o mestre e o escravo, ou seja, é responsável por coordenar todos os periféricos e por enviar informações a determinado escravo ou realizar consulta de informações. O escravo é responsável pela transferência de dados, podendo atuar como transmissor ou receptor dependendo da natureza do dispositivo conectado, como por exemplo teclado ou LCD (*Liquid Crystal Display*) .

SDA (*Serial Data*) é o pino que efetivamente transfere os dados para o mestre, e SCL (*Serial Clock*) serve para realizar a temporização entre os dispositivos mestre-escravo, de modo que a comunicação pela SDA possa ter confiabilidade. O envio e a recepção de dados através do MPU6050 é realizada utilizando a linha SDA, ou seja, é uma linha bi-direcional de comunicação, ora envia dados por este pino, ora recebe dados [9].

A linha SCL é responsável pelo clock do barramento e a linha SDA pela transmissão de dados, quando ambas as linhas estão em nível alto, o estado é neutro. Para iniciar a comunicação entre dispositivos conectados, a linha SDA deve ser alterada de nível alto para nível baixo, enquanto que a linha SCL permanece em nível alto. A transmissão dos dados acontece quando a linha SCL se encontra em nível baixo. Nesse momento, o estado da linha SDA pode alterar para o valor do bit que segue a sequência de transmissão. O valor dos dados da linha SDA é enviado apenas após a linha SCL é elevada para nível alto. A Figura 2.9 mostra o processo de escrita da sequência de bits "1101". A condição de interrupção de transmissão é satisfeita quando a linha SDA altera de nível baixo para nível alto enquanto a linha SCL encontra-se em nível alto.

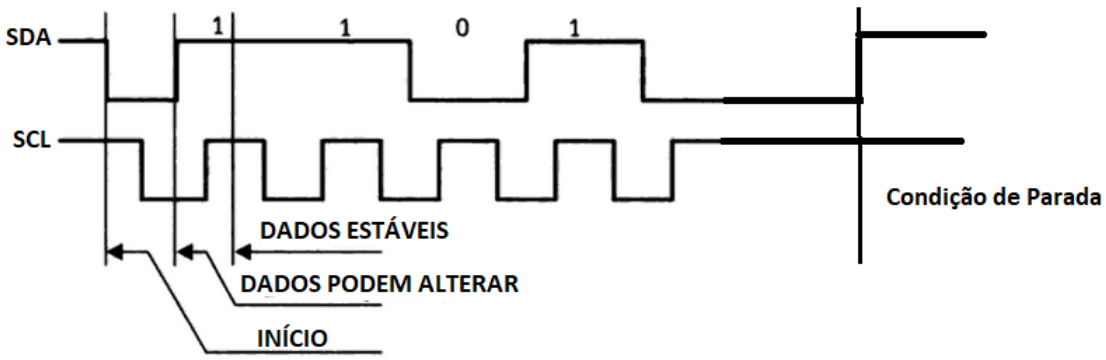


Figura 2.9: Leitura e Escrita utilizando I2C. Adaptado de [9]

Nesse trabalho, foi utilizado o MPU6050 para realizar a coleta da aceleração linear, também chamada de aceleração própria, do corpo humano e o giroscópio embutido no MPU6050 para medir a orientação do corpo humano baseado na velocidade angular. As informações dos sensores são as entradas que alimentam o funcionamento do sistema desenvolvido nesse projeto.

2.7 RaspberryPi

O RaspberryPi é um mini computador, do tamanho de um cartão de crédito, que abriga processador de dados, processador gráfico, *slots* para cartões de memória, interface USB (*Universal Serial Bus*) , HDMI (*High –Definition Multimedia Interface*) e seus respectivos controladores. Além disso, ele também contém memória RAM (*Random Access Memory*) , entrada de energia e barramentos de expansão. Ainda que pequeno, o Raspberry é um computador completo de baixo custo. O usuário pode conectá-lo a um monitor de computador ou a uma TV, juntamente com teclado e mouse e executar as atividades como navegar na Internet, escrever textos, ver vídeos, ouvir música, criar planilhas e realizar praticamente qualquer tarefa possível num computador convencional.

A sigla "Pi" é uma abreviação para Python e foi intitulada pelos desenvolvedores por eles recomendarem o Python como uma linguagem de programação para aprendizado e desenvolvimento; contudo há várias outras linguagens que também podem ser utilizadas pelo RaspberryPi. O sistema operacional, o qual deverá ser instalado em um cartão de memória SD (*Secure Digital*) visto que o mini-microcomputador não contém disco rígido próprio, é baseado no GNU/Linux. Também deve ser instalado no cartão de memória suas várias distribuições, como Raspian, Debian, Fedora Remix e Arch Linux [45].

O RaspberryPi apresenta-se em três versões e em vários modelos. Os modelos existentes são A, A+, B, B+, Compute Module,Blue Pi e Red Pi. As versões se apresentam como versão 1, versão 2 e mais recentemente, versão 3. A versão 3 surgiu em Fevereiro de 2016 e conta com um processador de 64 bits com quatro núcleos ARM Cortex A53 de 1.2GHz, dando ao dispositivo 50% mais performance e 33% mais velocidade que o RaspberryPi 2 utilizado no projeto de graduação [2] . Além disso, ele conta com Bluetooth e Wi-Fi integrado. Os modelos atuais de RaspberryPi podem ser observados na Figura 2.10.

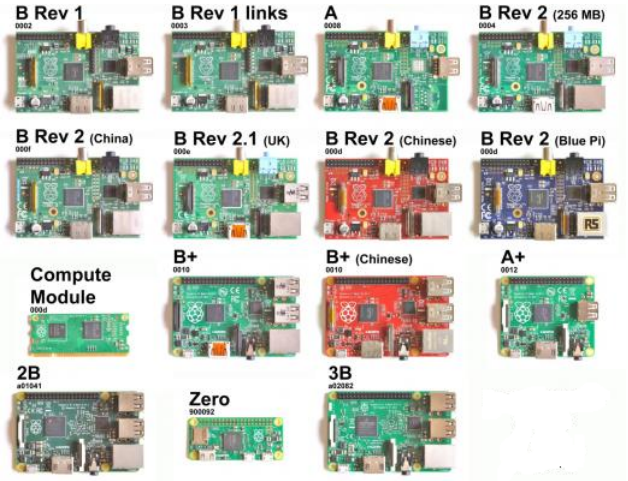


Figura 2.10: Modelos Atuais de RaspberryPi. Extraído de [10]

O RaspberryPi foi escolhido nesse trabalho, em vez de outros sistemas embarcados como o Arduino, porque ele apresenta maior processamento, mais interfaces com o usuário e vários

softwares que podem ser implementados favorecendo o desenvolvimento de projetos. Por exemplo, os softwares MySQL, Python e servidor web.

Nesse trabalho foi utilizado o modelo RaspberryPi 3B, pois o mesmo apresenta conectividade WiFi e Bluetooth, o que é fundamental para a mobilidade do protótipo desenvolvido.

2.8 Análise Matemática

Esse projeto utilizou algumas ferramentas matemáticas para determinar a posição corporal do paciente.

2.8.1 Regressão Linear

A regressão linear é uma ferramenta matemática utilizada para construir modelos que descrevem ou explicam o relacionamento que pode existir entre variáveis. O modelo mais simples é chamado de regressão linear simples, o mesmo define uma relação linear entre a variável dependente e uma variável independente. O modelo mais complexo é chamado de regressão linear múltipla, o mesmo define uma relação linear entre a variável dependente e várias variáveis independentes. Este último não será abordado nesse trabalho.

Normalmente, o objetivo ao se utilizar regressão linear é conhecer o resultado de apenas uma variável, chamada de dependente, e deseja-se saber como é a dependência da mesma com um outro conjunto de variáveis, chamado de independente. Uma vez identificada uma relação estatística entre ambas, pode-se então criar um modelo para representá-los e utilizá-los para realizar a predição [46].

A expressão para o modelo de regressão linear pode ser descrita conforme a equação 2.5:

$$y = \beta_0 + \beta_1 x + \epsilon \tag{2.5}$$

A equação 2.5 é utilizada quando deseja-se predizer o valor de y dado os valores das variáveis independentes.

• β_0 : constante de *offset* (deslocamento) da reta em relação ao eixo da variável dependente;

• x: $varivelindependente$;

• β_1 : declive ou coeficiente angular da reta;

• ϵ : erro aleatório, com média 0 e variância σ^2.

Se há n pares de dados (y_1, x_1),...,(y_n, x_n) é possível estimar os parâmetros β_0 e β_1 usando o método dos Mínimos Quadrados, tendo como objetivo minimizar a equação 2.6. Supondo

que sejam utilizados *p* pontos na regressão, temos:

$$L = \sum_{n=1}^{p} (y_i - (b_0 + b_1 x_i))^2 \tag{2.6}$$

supondo que b_0 e b_1 são estimativas amostrais de β_0 e β_1. O uso do método conduz as estimativas apresentadas nas equações 2.7 2.8 2.9 2.10 2.11 2.12:

$$b_1 = S_{xy}/S_{xx} \tag{2.7}$$

$$b_0 = \overline{Y} - b_1 \overline{X} \tag{2.8}$$

$$b_0 = \overline{Y} - b_1 \overline{X} \tag{2.9}$$

$$S_{xx} = \sum x_i^2 - (\sum x_i)^2/n \tag{2.10}$$

$$S_{yy} = \sum y_i^2 - (\sum y_i)^2/n \tag{2.11}$$

$$S_{xy} = \sum x_i y_i - (\sum x_i)(\sum y_i)/n \tag{2.12}$$

Nesse trabalho, utilizou-se a regressão linear para calcular o valor do coeficiente angular da reta formada entre os pontos de maior e menor amplitude do vetor formado pelos valores obtidos do acelerômetro e giroscópio durante o momento de coleta de dados do sensor. Isso foi utilizado como parâmetro de decisão se a posição do paciente era "andando " ou "queda ".

2.8.2 Integral Trapezoidal

A integral é o processo de medir a área sobre uma curva plotada no gráfico. A integral trapezoidal ou regra trapezoidal é uma abordagem numérica para encontrar integrais definidas. A ideia desse método é aproximar o integrando de um polinômio de n^{th} ordem [47].

$$f(x) \approx f_n(x) \tag{2.13}$$

$$f_n(x) = a_0 + a_1 x + ... + a_{n-1} x^{n-1} + a_n x^n \tag{2.14}$$

Para o intervalo $x_i \leq x \leq x_{i+1}$, a regra trapezoidal é dada por:

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{\triangle_x}{2}(f_i + f_{i+1}) \tag{2.15}$$

onde $\triangle_x = x_{i+1} - x_i$.

Nesse trabalho utilizou-se a integral Trapezoidal para calcular a área entre o ponto de maior e menor amplitude do deslocamento angular. Isso foi utilizado como mais um parâmetro de decisão do algoritmo para determinar qual tipo de movimento dinâmico é realizado.

2.8.3 Transformada *Wavelet*

Diferentemente da Transformada de Fourier, que representa o sinal como uma série de senos e cossenos e é ideal para sinais estacionários, a Transformada *Wavelet* se aplica para transformadas em janelas com tamanhos variáveis, sendo janelas grandes com informações sobre altas frequências e janelas menores com informações sobre baixas frequências. A Transformada *Wavelet* decompõe um sinal em versões escalonadas e transladadas das funções *Wavelet* [48].

É possível realizar a transformada de funções contínuas (CWT - *Continuous Wavelet Transform*) e de funções discretas (DWT - *Discrete Wavelet Transform*). Nesse trabalho a CWT foi utilizada para determinar os picos de amplitude no sinal obtido da posição do paciente. Esses picos, também chamados de *spikes*, foram necessários para identificar o impacto do movimento dinâmico.

2.9 Desenvolvimentos Recentes

Várias pesquisas recentes na área de desenvolvimento de algoritmos para detecção de quedas foram utilizadas como base para esse projeto. O artigo *Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information* [49] utiliza um sistema de detecção de quedas baseado em *thresholds*, com detecção de posição baseada no ângulo entre a coxa e o peito, e a partir disso, determina se o movimento foi intencional para inferir se a queda ocorreu. A forma mais comum de detecção de quedas na literatura utiliza acelerômetros de três eixos com algoritmos baseados em *thresholds* [50], porém esses estudos diferem-se em valores e método de processamento da informação. O estudo Lim, D., et al. [51] faz a combinação desse método com cadeia de Markov. O trabalho [51] utilizou o modelo de Markov ocultos (*Hidden Markov Models - HMM*) , onde o estado da cadeia de Markov não é diretamente observável, mas uma observação é uma função probabilística do estado da cadeia [52].

Falso positivo é um conceito recorrente nos trabalhos de detecção de quedas e indica

que ela foi erroneamente detectada, pois o paciente estava realizando outra atividade, como andar, correr ou levantar rapidamente de uma cadeira. Então, fazem-se necessários outros parâmetros para determinar se realmente ocorreram quedas.

O artigo *Increased Fall Detection Accuracy in an Accelerometer-Based Algorithm Considering Residual Movement* [15] apresenta uma técnica baseada em tempo remanescente. O autor calcula o valor resultante da aceleração nos três eixos, depois o compara com *thresholds* preestabelecidos; se os valores estiverem acima de um limiar, uma possibilidade de queda é detectada e um contador é inicializado. Esse contador mede quantas vezes os valores obtidos foram maiores do que os *thresholds* dentro da janela de tempo de seis segundos. Se este contador tiver como resultado um valor entre um e quatorze, a queda é detectada. Caso contrário, considera-se que outra atividade estava sendo realizada.

Uma abordagem recentemente publicada [53] utiliza a rede de *Petri* para detectar quedas, contudo o idoso deve constantemente carregar em seu bolso um *smartphone*, o que na maioria das vezes, é inviável. Outro artigo que utiliza essa abordagem de detecção de quedas com *smartphones* está presente em [54].

Lustrek, M., Kaluza, B. [55] algoritmo de aprendizado de máquina para avaliar o comportamento dos mesmos em uma aplicação de detecção de quedas. Foram utilizados oito algoritmos de aprendizado de máquina, entre os quais, o SVM se mostrou o classificador mais eficiente com acurácia de 97,7%.

Dinh, C., Struck, M. [56] trabalha com o conceito de "*expert knowledge*", no qual mantém-se a detecção independente das características físicas do paciente, como peso e altura. A lógica fuzzy é utilizada para aplicar esse conceito e o resultado é classificado por uma rede neural artificial. Mundher, Zaid., Zhong, J. [57] utilizam sensores *Kinect* embutidos em robôs móveis que fazem o reconhecimento de gestos e voz.

Torres, R., et al. [58] utiliza tags RFID (*Radio-Frequency Identification*) instaladas no tapete para obter informações dos pacientes e as encaminham para algoritmos de aprendizado de máquina, os quais fazem a decisão sobre a informação do paciente. Com base nos artigos mostrados acima verifica-se que há várias abordagens diferentes e todas elas tem como objetivo detectar a queda em 100% das ocorrências e não realizar falsos positivos e falsos negativos.

Este trabalho irá utilizar uma metodologia baseada em aprendizado não-supervisionado, com o algoritmo *K-Means*, e utilizará de *thresholds* para determinar a posição dinâmica do paciente. Para realizar a coleta das informações, serão utilizados acelerômetro e giroscópio que estão presentes no sensor MPU6050. Além disso, esse trabalho fará complemento aos artigos acima citados, utilizando um aplicativo Android e uma página web para fornecer uma nova abordagem sobre a metodologia de detecção de posição e queda.

Capítulo 3

Metodologia

O presente trabalho de mestrado tem como proposta desenvolver um equipamento que detecta posição e quedas corporais com precisão melhor que o trabalho desenvolvido na graduação [2]. Além disso, esse trabalho de mestrado decidiu utilizar apenas um dispositivo acoplado ao corpo do paciente, diferentemente dos trabalhos [49] e [2] que utilizaram um dispositivo acoplado ao peito e outra na coxa. Além disso, esse trabalho utilizou aprendizado de máquina para determinar se o movimento realizado pelo paciente era dinâmico ou estático. Diferentemente dos trabalhos [49] e [15] que utilizam outros parâmetros como *threshold* e tempo residual.

O dispositivo desenvolvido nesse trabalho de mestrado permite ao usuário acompanhar o estado do paciente através de um aplicativo Android e de uma plataforma web.

O dispositivo é composto por um RaspberryPi conectado a um sensor MPU6050 acoplado ao peito capaz de coletar informações nos 3 eixos (x, y e z) para acelerômetro e 3 eixos para giroscópio. O valor que é utilizado como métrica nesse trabalho é o valor da resultante obtida do acelerômetro e giroscópio. No RaspberryPi foi instalado um servidor web capaz de informar a situação real do paciente. Para o envio de informações entre o RaspberryPi e o Android foram utilizados *sockets*, conceito abordado na seção anterior, juntamente com o padrão IEEE 802.11 que já faz parte do RaspberryPi.

Conforme apresentado no capítulo de Fundamentação Teórica, para realizarmos a coleta dos dados da situação atual do paciente poderia ter sido utilizado câmeras, *kinect*, sensores de pressão, acelerômetros e giroscópios. Nesse projeto de mestrado foi decidido utilizar esses últimos porque eles são fáceis de serem implantados e oferecem boas estimativas relacionadas à aplicação de detecção de quedas. Acelerômetros são sensores que medem a aceleração própria de objetos em movimento nos três eixos coordenados (x,y e z). Essa aceleração é medida em relação a outro sistema em queda livre, de modo que esta está atrelada à força peso, sendo assim, um objeto em repouso, terá o valor de aceleração igual 1g para cima, pois em relação à um objeto em queda livre, o corpo está acelerado em 1g. Sendo assim, ele reflete a intensidade e frequência do movimento do corpo humano [59]. Giroscópio for-

nece informações sobre a velocidade angular e mede a velocidade de rotação e orientação do objeto no qual ele está anexado. Sendo assim, esse dispositivo oferece as informações necessárias para detectar se ocorreu uma orientação abrupta de movimento, como acontece na posição "queda".

Abaixo segue a definição de alguns conceitos que foram utilizados no escopo dessa dissertação:

• Movimento dinâmico: se refere ao movimento realizado pelo corpo quando o mesmo encontra-se andando ou em queda. Esse movimento é caracterizado por uma variação no valor da aceleração própria, obtida através do acelerômetro, e por uma variação no valor da velocidade angular, obtida através do giroscópio.

• Movimento estático: se refere ao movimento realizado pelo corpo humano quando o mesmo encontra-se nas posições: sentado, deitado e em pé. Esse movimento é caracterizado por um valor estático da aceleração própria, obtida através do acelerômetro, e pelo valor zero da velocidade angular, obtida através do giroscópio. A seção 3.3.3 apresenta os planos de secção que definem cada uma dessas posições estáticas.

O fluxograma da Figura 3.1 mostra as etapas realizadas para o desenvolvimento desse projeto.

Figura 3.1: Etapas para a construção do equipamento de detecção e monitoramento

• Etapa 1 - Decisão da tecnologia a ser desenvolvida: durante o período de pesquisa foi decidido qual tecnologia de aprendizado de máquina seria adequada para essa aplicação.

• Etapa 2 - Montagem física dos componentes do sistema: foi realizado a conexão física entre o RaspberryPi e o MPU6050.

• Etapa 3 - Programação Python: foi realizado o desenvolvimento do software responsável por toda a parte lógica do sistema.

• Etapa 4 - Inserção das informações no Banco de Dados: foi utilizado um diagrama de classes relacionando todas as tabelas para verificar como as informações deveriam ser inseridas e recuperadas.

• Etapa 5 - Programação Android: foi realizado o desenvolvimento do aplicativo Android

na linguagem Java utilizando a IDE Android Studio.

• Etapa 6 - Programação Web: foi realizado o desenvolvimento web responsável pela apresentação dos resultados do sistema.

• Etapa 7 - Testes: foi realizado testes do sistema obtendo o comportamento das respostas às devidas situações de teste.

3.1 Decisão da tecnologia a ser desenvolvida

Durante essa etapa foram estudadas as tecnologias atuais para aprendizado de máquina, abordados no capítulo anterior, e determinado qual delas seria a mais adequada para este trabalho. Para realizar o projeto de mestrado foi decidido utilizar o aprendizado não-supervisionado para analisar os dados. O propósito desse trabalho é avaliar a clusterização dos dados obtidos dos sensores para determinar a posição do paciente. Essa clusterização permite descobrir similaridades e diferenças entre padrões e concluir informações sobre o comportamento dos dados. O algoritmo de aprendizado não-supervisionado escolhido foi o *K-Means Clustering*. A maior vantagem desse algoritmo é a sua simplicidade de implantação e cálculo [6].

A abordagem supervisionada poderia ter sido utilizada nesse trabalho para determinar a posição do paciente. Contudo, deveriam ser utilizados algoritmos mais simples como o K- NN e Modelos Lineares, em que não há necessidade de muitos dados de treinamento como SVM e Rede Neural Artificial. Em uma aplicação de detecção de quedas, encontrar a quantidade de voluntários suficientes para alimentar a base de dados de algoritmos como SVM e Rede Neural Artificial é frequentemente inviável. Aprendizado Supervisionado requer maior atenção no desenvolvimento dos dados de treinamento, o que não foi objetivo desse trabalho. Se os dados de treinamento não forem representativos, a decisão de saída não será eficaz. Nesse projeto foi utilizado o algoritmo não-supervisionado *K-Means* para não realizar uma pré-rotulação dos dados obtidos dos sensores.

Com objetivo de desenvolver um dispositivo capaz de adaptar às condições físicas do paciente, foi adicionado uma etapa chamada calibração, na qual o paciente deve permanecer 40 segundos em cada posição estática (deitado,sendo e em pé) e na posição dinâmica (andando) enquanto o dispositivo captura em *backgroud* as informações de aceleração própria e velocidade angular referentes à determinada posição e paciente. Esse processo pode ser interpretado como um conjunto de rótulos de acordo com a posição realizada, contudo, o objetivo foi classificar os dados, em vez de fornecer pares entrada-saída. O momento da calibração do dispositivo foi útil apenas para coletar os dados dos pacientes de acordo com suas características físicas, independente se eles formavam um conjunto eficaz de treinamento.

Durante as simulações que acompanharam o desenvolvimento do dispositivo, foi identificado que a diferenciação das posições dinâmicas (andando e queda) não eram bem definidas

pelo algoritmo *K-Means*, sendo assim, foi investigado uma alternativa para determinação desses movimentos. Foi decidido utilizar uma sequência de processos de decisão baseados em *threshold* que são obtidos através da integral trapezoidal e da regressão linear. Esses dois parâmetros foram utilizados porque a curva da resultante da velocidade angular para os movimentos andando e queda são evidentemente diferentes. A curva para a posição queda tem um abrupto declínio, consequentemente apresenta um valor modular de coeficiente angular maior do que a curva do movimento andando. Quando o corpo está andando, essa curva permanece variando nos valores de amplitude, enquanto que quando o corpo está em queda, a curva tem uma variação e, após o momento de queda a curva permanece estática próxima ao valor zero, sendo assim, a área sob a curva é um bom parâmetro para diferenciar os movimentos. Por esse motivo, optou-se por utilizar esses parâmetros para diferenciar as posições dinâmicas realizadas.

3.2 Montagem física dos componentes do sistema

Nessa etapa, foi realizada a conexão física dos componentes seguindo a Figura 3.2. É necessário utilizar os pinos GPIO's (*General Purpose Input/Output*) do RaspberryPi para realizar a conexão. Segue o mapeamento, conforme a tabela 3.1:

Tabela 3.1: Conexão física entre RaspberryPi e MPU6050

Pino(RaspberryPi)	**Conexão MPU6050**
Pino 1 - 3.3V	VCC (IC Power Supply)
Pino 3 - SDA	SDA (Serial Data Line)
Pino 5 - SCL	SCL (Serial Clock Line)
Pino 6 - Ground	GND (Ground)

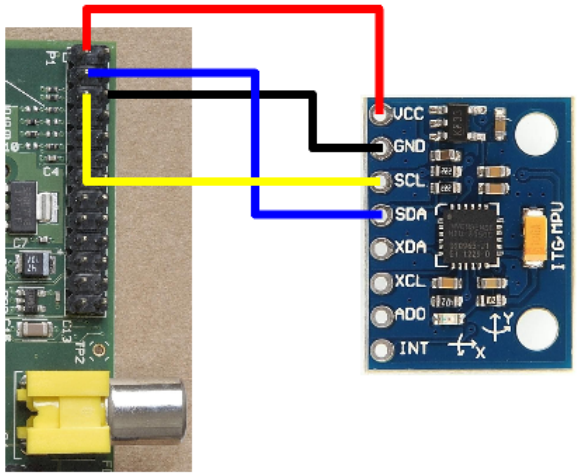


Figura 3.2: Conexões físicas

O dispositivo ficará com a disposição conforme a Figura 3.3.

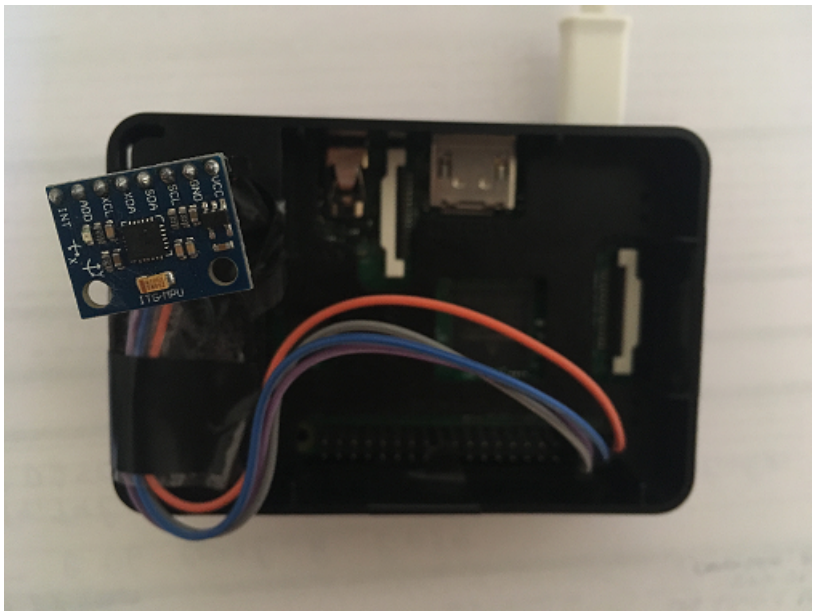


Figura 3.3: Dispositivo com conexões físicas realizadas

O sistema é composto por um único dispositivo, que deve ser acoplado ao peito do paciente, diferentemente do projeto de graduação [2]. Foi escolhido o peito em vez da coxa visto que o dispositivo teria maior usabilidade, considerando que nessa região há menos exposição a fatores externos e há menos variações em atividades físicas como correr e andar.

O dispositivo agrega todas as funções necessárias para o funcionamento do mesmo, como servidor, transmissor e processador. A Figura 3.4 apresenta como o dispositivo fica posicionado no corpo do paciente.

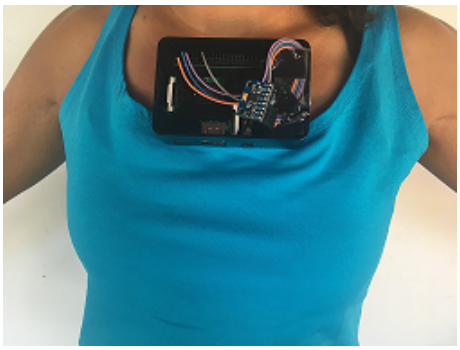


Figura 3.4: Dispositivo acoplado ao corpo

3.3 Programação Python

Foi utilizada a linguagem Python para desenvolver o software embarcado. Existe uma biblioteca Python para aprendizagem de máquina de código aberto chamada *scikit-learn*. Ela

contém algoritmos de classificação, regressão e agrupamento, entre eles o *K-Means Clustering* que foi utilizado nesse projeto. Essa biblioteca interage com outras bibliotecas numéricas como *NumPy* e *Scipy*, fornecendo todas as ferramentas necessárias para desenvolver algoritmos complexos.

Antes de iniciar o desenvolvimento no RaspberryPi foi realizado simulações do modelo no programa Spyder. Verificou-se que os resultados da simulação eram viáveis, ocasionando então o seu desenvolvimento.

Todo o dispositivo foi desenvolvido em Python. Nesse projeto foram utilizados oito *scripts*:

- Quatro para realizar a calibração do dispositivo com os dados do paciente, pois a informação obtida varia de acordo com as condições físicas do indivíduo;;
- Uma biblioteca para detectar picos no sinal obtido através da leitura dos sensores;
- Um *script* necessário para gerar o aquivo csv (*Comma-Separated Values*), com dados obtidos da calibração para ser utilizado no aprendizado não-supervisionado;
- Um *script* que realiza a predição da posição do paciente;
- Um *script* que obtém os dados dos sensores, realiza a normalização dos dados obtidos, realiza os cálculos, insere as informações no banco de dados e apresenta-os para o usuário.

O sistema pode apresentar duas abordagens que são descritas nas seções 3.3.1 e 3.3.2. A seção de resultados apresenta as razões que levaram à segunda abordagem a ser utilizada em detrimento da primeira.

3.3.1 Primeira Abordagem do Algoritmo de Decisão: Utilizando *threshold* a partir do valor da integral trapezoidal

Nessa abordagem, para determinar se o movimento é estático ou dinâmico, utilizou-se o valor da integral trapezoidal como *thresold*. O *flowchart* é apresentado na Figura 3.5

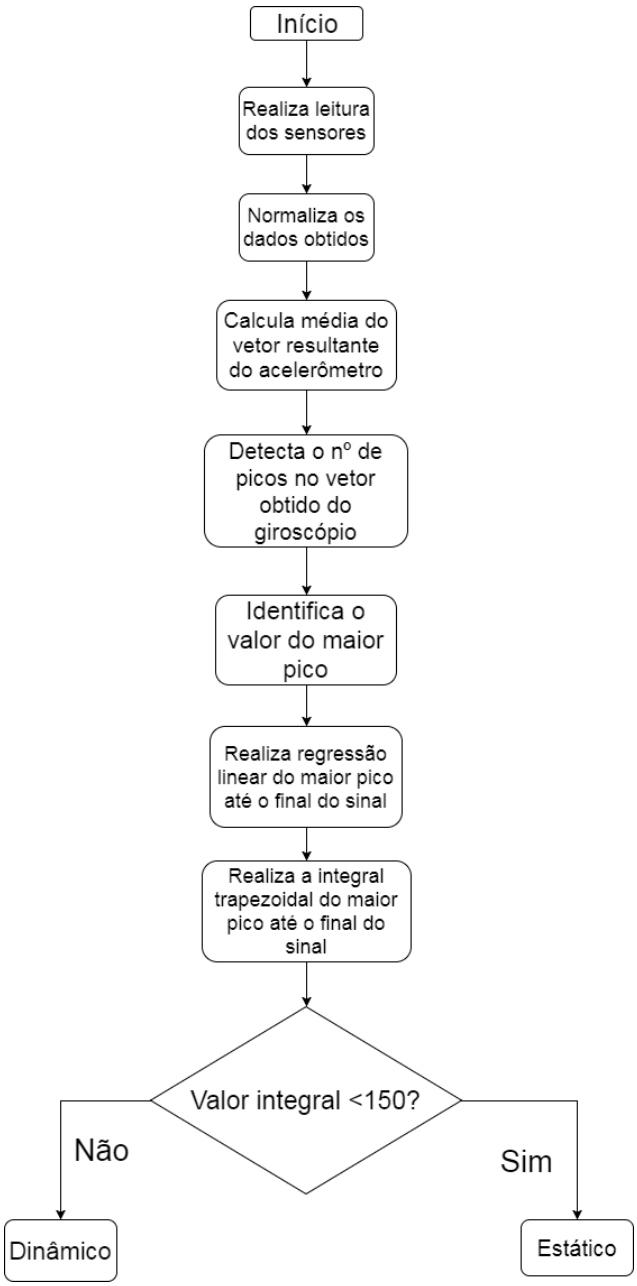


Figura 3.5: Flowchart verifica tipo de movimento - Primeira Abordagem

Primeiramente realiza-se a leitura dos sensores, normaliza os valores recebidos pelo *off-set* apresentado pelos sensores e depois calcula-se a média do vetor resultante do acelerômetro. Todas essas etapas estão detalhadas na seção 3.3.3.

A partir da próxima tarefa, o algoritmo se diferencia da segunda abordagem. Detecta-se o número de picos do vetor resultante do giroscópio, identifica-se o valor de maior pico, realiza-se a regressão linear no intervalo entre o maior e menor amplitude do vetor obtido do giroscópio, realiza-se a integral trapezoidal nesse intervalo e finalmente, verifica-se se o valor da integral é menor do que 150, a escolha desse valor será discutido no Capítulo 4. Caso positivo, o movimento realizado é estático e o algoritmo da 3.6 é iniciado, caso contrário, o algoritmo da Figura 3.13 é realizado caracterizando o movimento dinâmico.

Dado que o movimento é estático, é iniciado uma outra parte do *script*. Realiza-se a

predição da posição chamando a função *K-Means* do *scikit-learn*. Os parâmetros fornecidos para a mesma são: número de *clusters* (nesse caso 4 *clusters* pois são 4 possíveis leituras: dinâmico, deitado, sentado e em pé), número de *seeds* da função *random-state* (que impedem a variação das *labels* a cada nova chamada da função *K-Means*), e o vetor utilizado para ensinar a máquina. Esse vetor foi coletado no momento de calibração do programa, como discutido na seção 3.3.3. A função então fica conforme descrito na equação 3.1:

$$kmeans = KMeans(n - clusters = 4, random - state = 3425).fit(mat) \tag{3.1}$$

Nesse momento, é necessário chamar o objeto predição, passando como parâmetro o vetor resultante do acelerômetro.

$$Label = kmeans.predict(a_{magnitude}) \tag{3.2}$$

Em seguida, verifica-se o valor do *label*, e, se for igual a "0", o paciente está sentado; se o valor for igual a "1", o paciente encontra-se deitado; caso o valor seja igual a "2", o paciente encontra-se em pé; finalmente, caso o valor seja igual a "3", trata-se de uma posição dinâmica, significando que o paciente está andando ou caindo.

A seção Resultados especifica com mais detalhes o estudo realizado previamente à execução desse algoritmo.

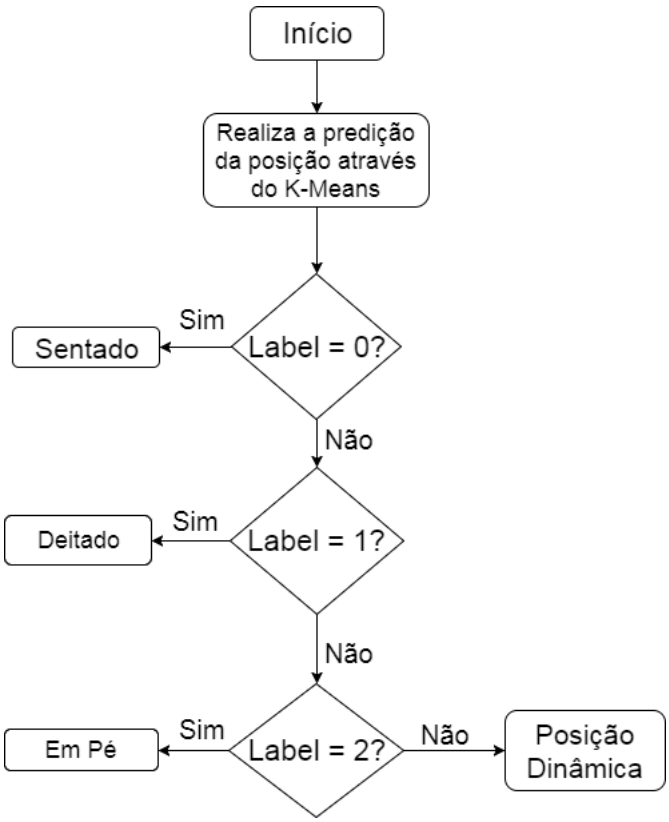


Figura 3.6: Movimento Estático

3.3.2 Segunda Abordagem do Algoritmo de Decisão: Utilizando *K-Means* para determinar se o movimento é dinâmico ou estático

O algoritmo inicia realizando a leitura dos sensores, normalizando os dados, calculando a média do vetor resultante do acelerômetro, da mesma forma que foi realizado na primeira abordagem. A partir da próxima atividade, o algoritmo se diferencia da primeira abordagem. Em vez de detectar o número de picos no vetor obtido pelo giroscópio, é realizada a predição no *K-Means*, passando como parâmetro da função a média do vetor resultante.

O retorno dessa função indica qual posição foi predita. Se o *label* for igual a "3", o corpo está em movimento dinâmico e é iniciado a contagem de picos no vetor obtido do giroscópio. Realiza-se o cálculo da regressão linear e da integral trapezoidal no intervalo de maior e menor amplitude do vetor obtido do giroscópio. Em seguida, é realizado o algoritmo de movimento dinâmico apresentado na Figura 3.13. Caso contrário, identifica-se qual é o *label* de saída e define-se qual movimento estático está sendo realizado.

A Figura 3.7 apresenta esse processo.

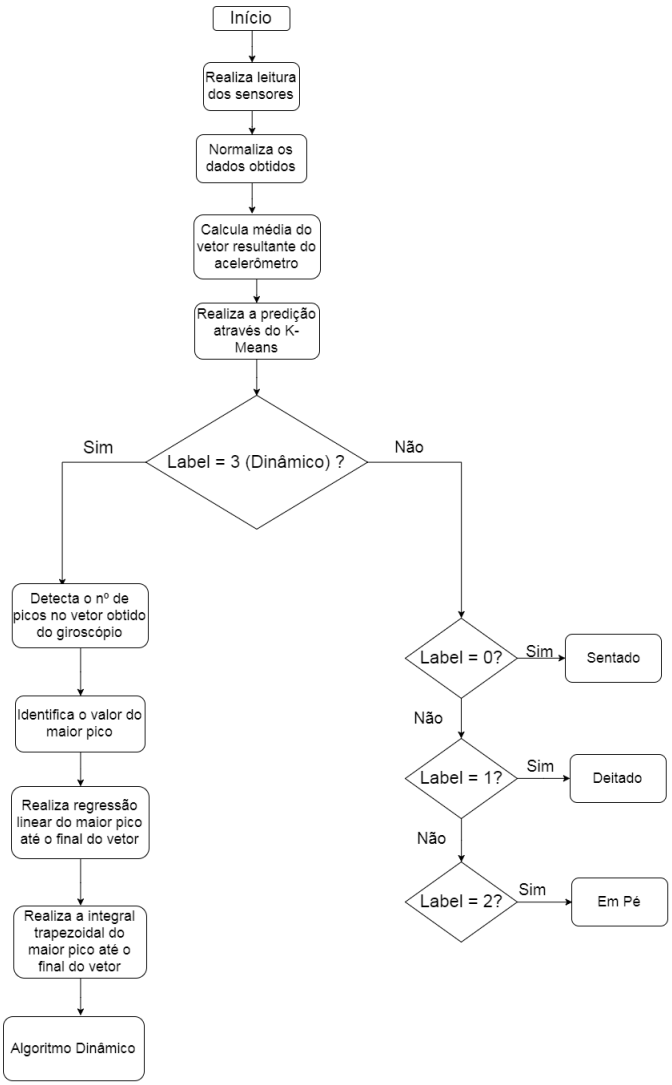


Figura 3.7: Flowchart verifica tipo de movimento - Segunda Abordagem

3.3.3 Atividades relativas as duas abordagens

A primeira atividade para utilizar o programa consiste em calibrar o equipamento para que ele realize as predições baseadas nas características físicas do paciente. Isso fornece adaptabilidade ao sistema. O paciente deverá sincronizar o aplicativo Android com o algoritmo em Python, para isso o paciente deverá ativar a função calibração do aplicativo Android e permanecer em cada uma das posições estáticas e na posição dinâmica "andando" durante 40 segundos enquanto o programa realiza a leitura das posições e armazena as informações em um arquivo no formato csv que será posteriormente utilizado para ensinar a máquina como deverá ser a predição. A Figura 3.8 apresenta o *flowchart* que descreve esse processo.

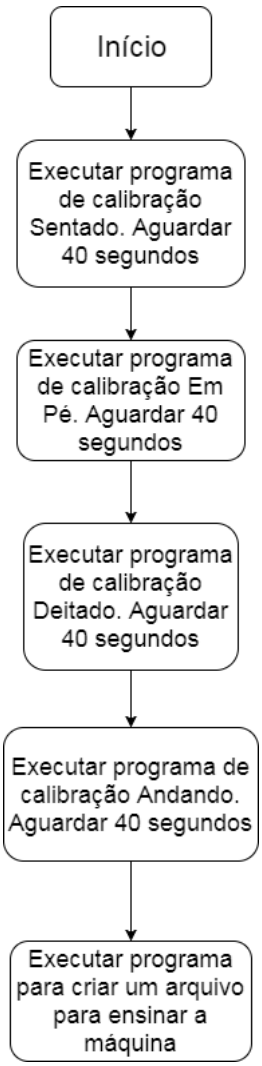


Figura 3.8: Calibração do Sistema

Assim como o Arduino, o RaspberryPi apresenta uma biblioteca de leitura I2C chamada *smbus*. Os dados utilizam uma porta de 9600 *bauds* e são amostrados a frequência de 2 Hz, o que é equivalente a uma janela de análise de 0,5 segundos. A leitura dos sensores é realizada utilizando as linhas SCL e SDA do RaspberryPi.

A calibração dos sensores foi realizada no projeto de graduação [2]. A configuração

padrão do MPU6050 apresenta uma sensibilidade de $\pm$2g para o acelerômetro e $\pm$250 °/s para o giroscópio, que é o intervalo de maior sensibilidade, e este foi o valor utilizado nesse projeto. Para obter o valor correspondente de força gravitacional aplicada ao corpo, devemos utilizar um fator de sensibilidade, neste caso o valor adimensional de 16384. Logo, todos os valores obtidos pelo acelerômetro foram divididos por 16384 para chegar ao valor correspondente à força gravitacional, (como exemplo, tendo o sensor na posição horizontal e sem movimento, o valor obtido deverá ser aproximadamente "0" para o eixo x, "0" para o eixo y e "1"para o eixo z).

O *script* de calibração utilizado em [2] inicia assumindo valores iniciais de *offset* para os eixos e depois realiza um conjunto de médias de tal forma que o resultado final esperado para o sensor em posição horizontal sem movimento seja “0 0 16384 0 0 0”. Os valores de *offset* obtidos para os sensores são mostrados na tabela 3.2.

	Peito
x accelerometer(g):	-4415
y accelerometer(g):	1795
z accelerometer(g):	1405
x gyroscope(°/s):	308
y gyroscope(°/s):	1
z gyroscope(°/s):	53

Tabela 3.2: Valores de *Offset*

Os valores recebidos pela porta serial devem ser divididos pelo valor adimensional 16384, para os dados provenientes do acelerômetro e pelo valor adimensional 131, para os dados provenientes do giroscópio. Isso é necessário para normalizar os valores obtidos do sensor MPU6050, chamados de valores "crus" [60]. Depois é calculado o *signal vector magnitude* desses vetores através das fórmulas abaixo:

$$a_{magnitude} = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{3.3}$$

$$g_{magnitude} = \sqrt{g_x^2 + g_y^2 + g_z^2} \tag{3.4}$$

Os valores $a_x, a_y, a_z, g_x, g_y, g_z$ equivalem aos graus de liberdade do giroscópio e acelerômetro. Os valores $a_{resultante}$ e $g_{resultante}$ equivalem à média dos valores obtidos pelo acelerômetro e giroscópio nos três eixos coordenados. Esses valores são calculados para todas as posições estáticas (sentado, deita e em pé) e para a posição dinâmica andando.

A partir do vetor resultante da média dos valores obtidos pelo giroscópio é realizado o cálculo do número de picos nesse vetor, também conhecidos por *spikes*. Utilizou-se as bibliotecas *find peaks cwt* e *detect peaks* do Scipy para realizar esse cálculo. A Figura 3.9 apresenta como o programa retorna o valor calculado por essas funções.

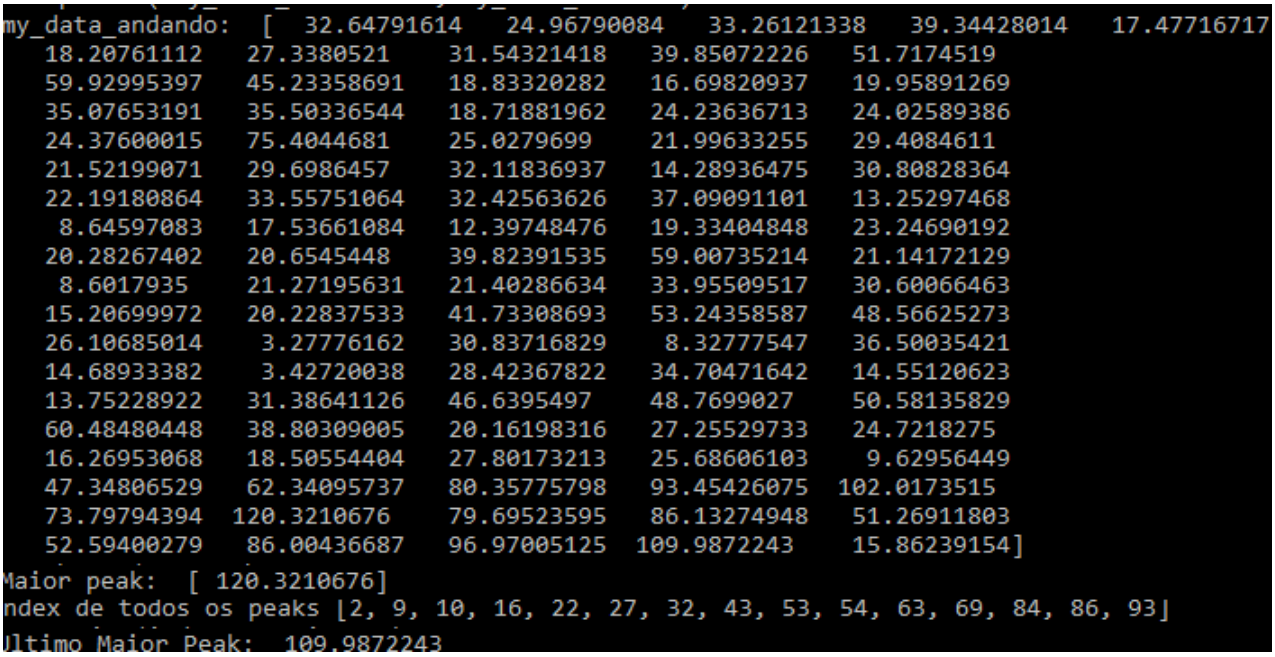


Figura 3.9: Picos do sinal obtido pelo acelerômetro

Em seguida, é realizada a regressão linear no intervalo referente ao *index* de maior amplitude até o *index* referente ao de menor amplitude no vetor obtido do giroscópio. A regressão linear fornece os valores do coeficiente angular e o deslocamento linear das curvas que representam os movimentos dinâmicos. Esse cálculo foi realizado utilizando os fundamentos teóricos de regressão linear abordados na seção 2.8.1.

Na primeira abordagem, realizou-se o cálculo da integral trapezoidal para todas as posições do paciente a fim de decidir se o movimento realizado é estático ou dinâmico. Na segunda abordagem, o cálculo da integral trapezoidal foi realizado apenas no caso em que o movimento tinha sido determinado pelo algoritmo como dinâmico, com o objetivo de verificar se o movimento dinâmico realizado era "cair" ou "andar".

Se for constatado que o corpo está estático, deve-se determinar em qual posição ele se encontra. As informações dos planos de secção do corpo humano são utilizadas pelo *K-Means* para clusterizar as diferentes posições estáticas. Para fazer essa inferência o *K-Means* utiliza os valores obtidos do acelerômetro para decidir qual posição estática está sendo realizada. A posição do corpo humano se diferencia através de seus planos de secção. Como pode ser visto na Figura 2.8, o sensor MPU6050 indica a orientação dos eixos coordenados. Essa orientação é utilizada pelo acelerômetro e giroscópio, por essa razão, esse dispositivo foi construído para utilizar essa orientação padrão e aplicá-la aos planos de secção do corpo humano. Os planos de secção podem ser vistos nas Figuras 3.10, 3.11 e 3.12.

Figura 3.10: Planos que atravessam o corpo quando está em pé

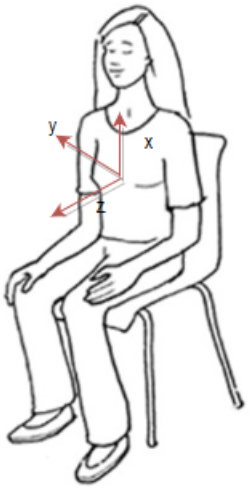


Figura 3.11: Planos que atravessam o corpo quando está sentado

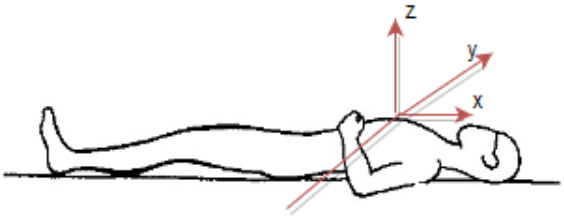


Figura 3.12: Planos que atravessam o corpo quando está deitado

A Figura 3.13 apresenta o *flowchart* do processo de determinação do movimento dinâmico. Após o algoritmo *K-Means* indicar que o movimento realizado é dinâmico, é utilizado uma sequência de "if-else" baseada em *threshold* para determinar qual tipo de movimento dinâmico está sendo realizado.

Verifica-se o valor da integral trapezoidal e do coeficiente angular obtido da regressão linear calculada no vetor obtido do giroscópio no intervalo entre o valor de maior e menor amplitude. Em seguida é verificado se o valor obtido da integral é maior que 600 e se

o coeficiente angular é maior do que 0,5. Caso a resposta seja afirmativa, o movimento realizado é "queda", caso alguma das condições anteriores não sejam satisfeitas o movimento é "andando".

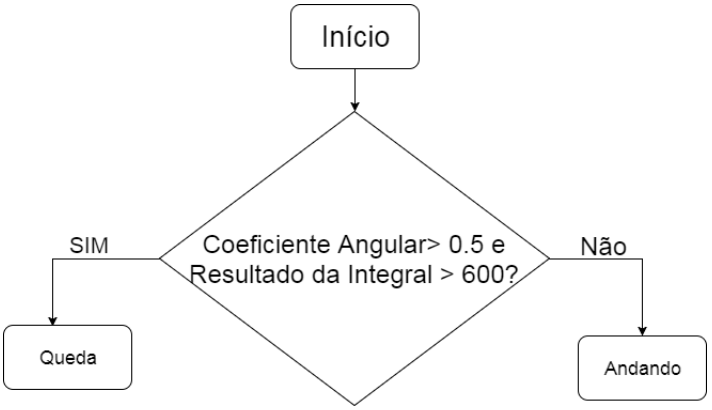


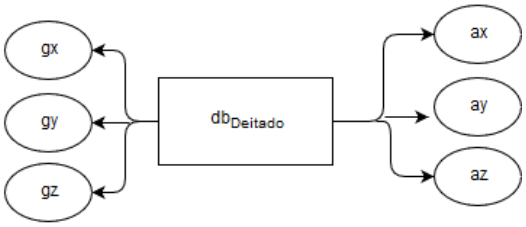
Figura 3.13: Movimento Dinâmico

3.4 Inserção das informações no Banco de Dados

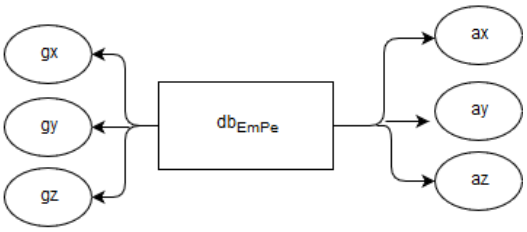
Nesse projeto de mestrado foi utilizado a linguagem de banco de dados MySQL e o servidor de hospedagem local Apache do RaspberryPi. Foram criados quatro banco de dados: $db_{Sentado}$, $db_{Deitado}$, db_{EmPe}, $db_{Andando}$ e $db_{Cotidiano}$. Os bancos $db_{Sentado}$, $db_{Deitado}$, db_{EmPe} e $db_{Andando}$ contém os dados coletados no momento da calibração do dispositivo através do aplicativo Android. Esses dados são coletados durante 40 segundos em cada posição, totalizando 40 entradas, mais detalhes na seção 4.3. O banco de dados $db_{Cotidiano}$ contém 6 tabelas: $posicao_{Andando}$, $posicao_{Sentado}$, $posicao_{Deitado}$, $posicao_{EmPe}$, $posicao_{Caiu}$ e $tabela_{posicao}$. Todas as tabelas exceto a a tabela $tabela_{posicao}$ são necessárias para incrementar o contador para contabilizar a quantidade de eventos que ocorreram em determinada posição. A $tabela_{posicao}$ é utilizada para armazenar a posição atual. Essa é a tabela utilizada pela página web e aplicativo Android.

Os atributos utilizados pelos bancos $db_{Sentado}$, $db_{Deitado}$, db_{EmPe}, $db_{Andando}$ e $db_{Cotidiano}$ são apresentados na Figura 3.14.

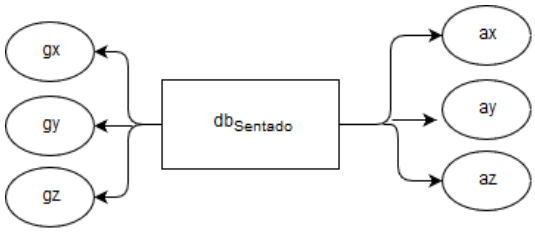
O diagrama de classe da Figura 3.15 foi realizado utilizando a aplicação web *Drawio* e mostra como ocorrem os relacionamentos entre as classes [61].



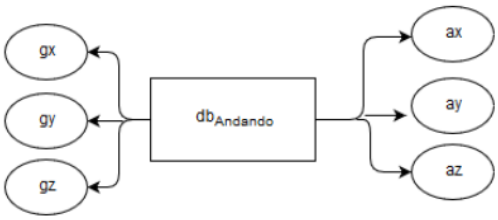
(a): Banco de Dados da posição deitado



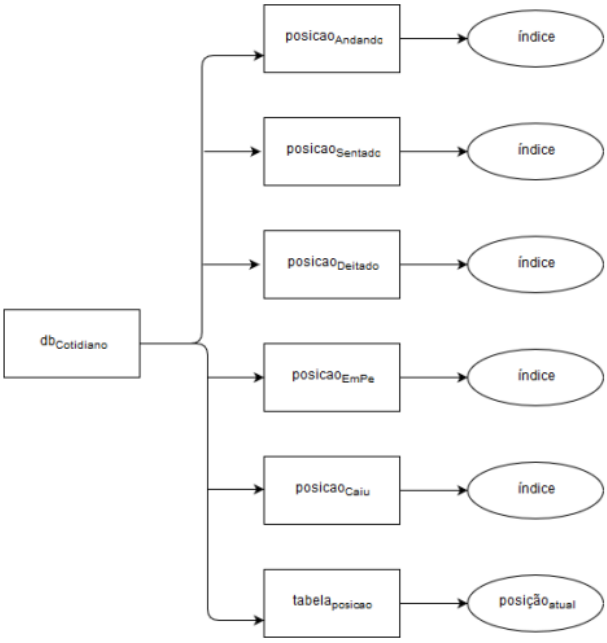
(b): Banco de Dados da posição em pé



(c): Banco de Dados da posição sentado



(d): Banco de Dados da posição andando



(e): Banco de Dados de todas as posições

Figura 3.14: Banco de Dados e as correspondentes tabelas

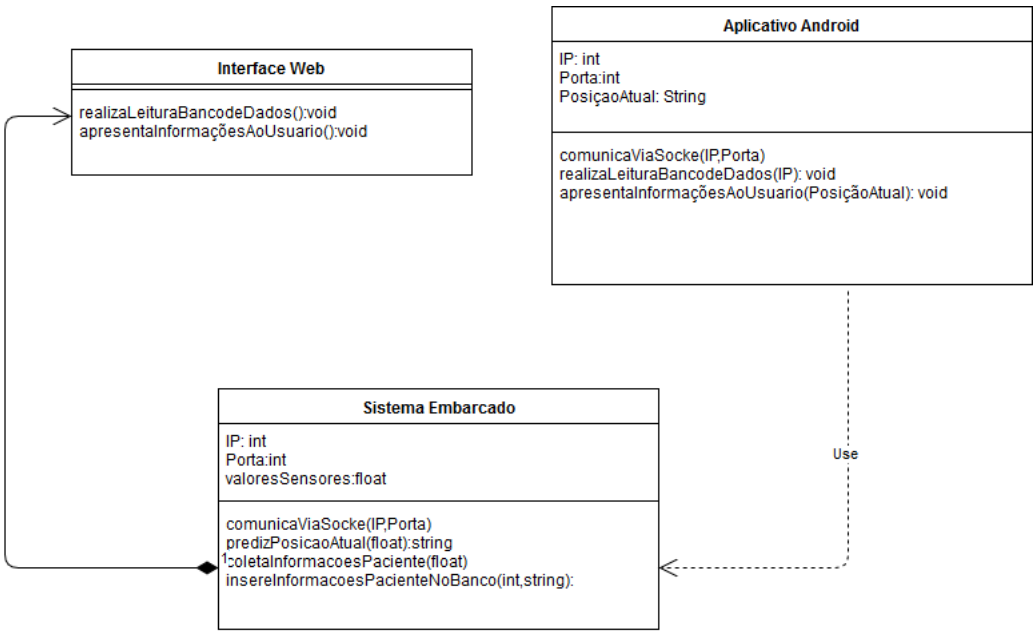


Figura 3.15: Diagrama de Classe

Existem três classes nesse sistema: interface web, aplicativo Android e sistema embarcado. A interface web está agregada ao sistema embarcado, pois a mesma foi configurada utilizando um servidor local no RaspberryPi. Ela é responsável por realizar a leitura da posição no banco de dados e apresentar essa informação ao usuário. O dispositivo é dependente do sistema embarcado, que atua como servidor, para que esse realize uma comunicação via *socket* entre o aplicativo Android e o RaspberryPi para calibrar o algoritmo de acordo com as características físicas do paciente. O sistema embarcado também realiza o cálculo da posição atual do paciente, coleta as informações do paciente através do sensor MPU6050 conectado ao mesmo e insere as informações sofre a posição atual do paciente no banco de dados local. Por essas razões, o sistema embarcado é considerado a classe principal.

3.5 Programação Android

Foi utilizada a IDE Android Studio para realizar o desenvolvimento do aplicativo Android. Essa IDE é baseada na linguagem Java. Para realizar a conexão entre o aplicativo Android e o RapberryPi foi configurado um endereço IP fixo no RaspberryPi e uma porta TCP fixa. Isso foi necessário para estabelecer a comunicação via *socket* entre a aplicação cliente e servidor. A Figura 3.16 apresenta essa arquitetura.

A seção 4.3 irá apresentar o endereço IP e número de porta utilizada.

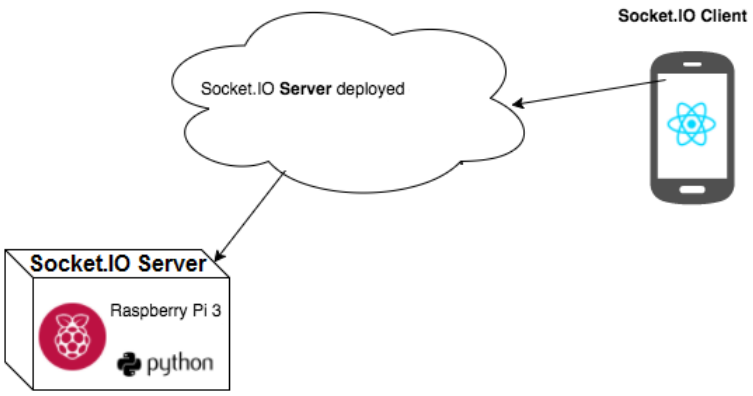


Figura 3.16: Conexão via Socket. Adaptado de [11]

O fluxograma do aplicativo Android é apresentado na Figura 3.17. O algoritmo inicia criando dois fragmentos, que são correspondentes as telas *"Home"* e *"Calibrate"*. Se o usuário clicar em *calibrate*, o mesmo deverá selecionar sequencialmente as posições indicadas e permanecer nelas por 40 segundos enquanto o sistema coleta as informações e armazena no banco de dados.

Após o sistema ser calibrado, as informações sobre a posição atual do paciente são apresentadas na tela "*Home*".

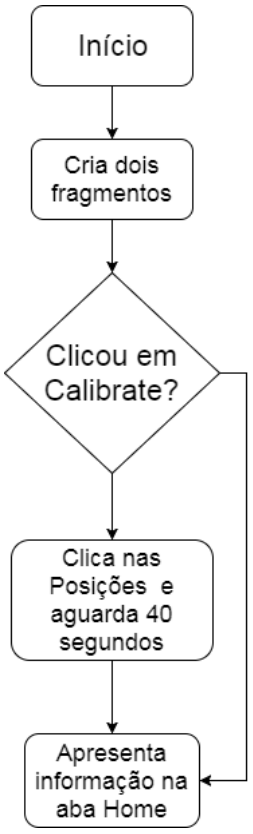


Figura 3.17: Fluxograma do Aplicativo Android

3.6 Programação Web

Foram utilizadas as linguagens de programação HTML, CSS e MySQL para desenvolver a interface web. A interface web oferece informações sobre a posição atual do paciente em tempo real e uma página inicial introdutória sobre a vida cotidiana de idosos. A interface web apresenta duas páginas de apresentação ao usuário. A primeira página mostra situações cotidianas de idosos, com objetivo comercial, a segunda página apresenta as informações em tempo real sobre o paciente.

3.7 Testes

Durante essa etapa foi verificado o comportamento do dispositivo em todas as possíveis posições. Os testes foram realizados em ambiente real, sendo as quedas realizadas no chão sem o apoio de amenizadores de impactos, mostrando de forma confiável a resposta do sistema.

Os testes foram realizados em uma pessoa do sexo feminino, 26 anos, 1,63cm de altura e 55 kg. Os testem foram realizados em uma repetição de 10 vezes em cada posição. Os resultados são apresentados no capítulo 4.

Capítulo 4

Resultados

4.1 Diagrama de Caso de Uso

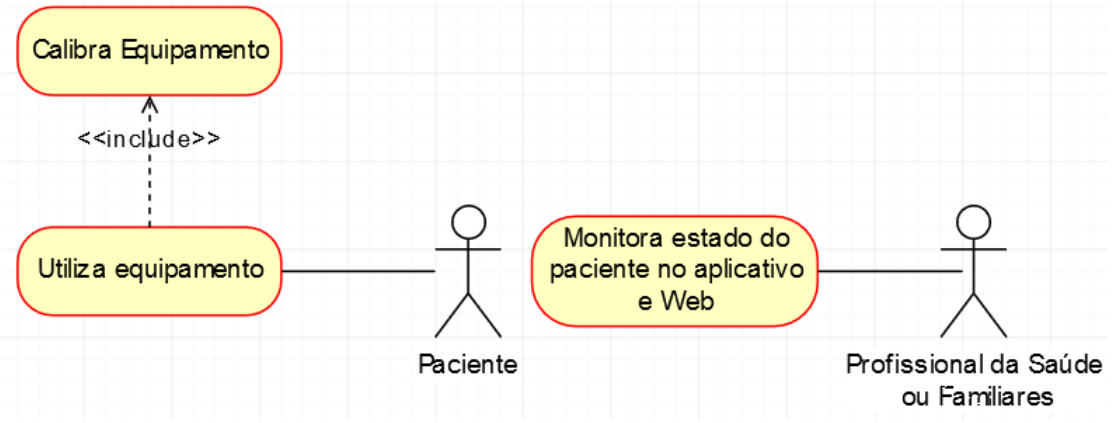


Figura 4.1: Diagrama de Caso de Uso

A Figura 4.1 apresenta o diagrama de caso de uso desenvolvido utilizando a aplicação web *drawio*. Esse diagrama descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário. Os atores desse sistema são: paciente e o profissional da área da saúde ou familiares. O paciente deverá acoplar o dispositivo no peito, posteriormente deverá calibrá-lo utilizando o aplicativo Android. O paciente não poderá inserir nenhuma informação ao sistema. O equipamento automaticamente irá inserir as informações à medida que as leituras dos sensores forem realizadas. O profissional da área de saúde ou familiares estarão frequentemente monitorando a situação do paciente na página web ou no aplicativo Android. Os mesmos não poderão inserir nenhuma informação no sistema, apesar de conseguir acessar e visualizar a informação do seu paciente.

4.2 Gráficos utilizados como embasamento para construção do algoritmo

Os dados utilizados para simulação do algoritmo a ser desenvolvido nesse projeto de mestrado foram os mesmos utilizados no trabalho de graduação [2]. Esses dados foram obtidos utilizando o monitor serial do software Coolterm e depois foi realizada a respectiva plotagem no software Spyder (*Scientific Python Development Envinonment*), que é um ambiente para edição, testes e *debugging* de programas Python [42].

Quedas geralmente são simuladas utilizando um colchão [62][63], nesse projeto, as quedas foram realizadas no chão para mostrar a eficácia do dispositivo para ambientes reais. O indivíduo que realizou as quedas é do sexo feminino, 26 anos, 1,63 metros de altura e 55 quilos, esses dados são importantes porque a velocidade angular e a aceleração própria do indivíduo variam com o peso, idade e altura. Os testes foram realizados em cada posição com 10 repetições cada.

No projeto de graduação [2], foi realizado uma pesquisa do comportamento dos movimentos estáticos em relação aos dinâmicos ao longo do tempo. A Figura 4.2 apresenta os resultados obtidos nessa pesquisa. É possível observar que a variação $\triangle$a da aceleração quando o corpo está em movimento é notoriamente maior do que quando o mesmo encontra-se estático. Por essa razão, no projeto de graduação [2] foi utilizado *thresholds* baseados no valor da aceleração para diferenciar os movimentos.

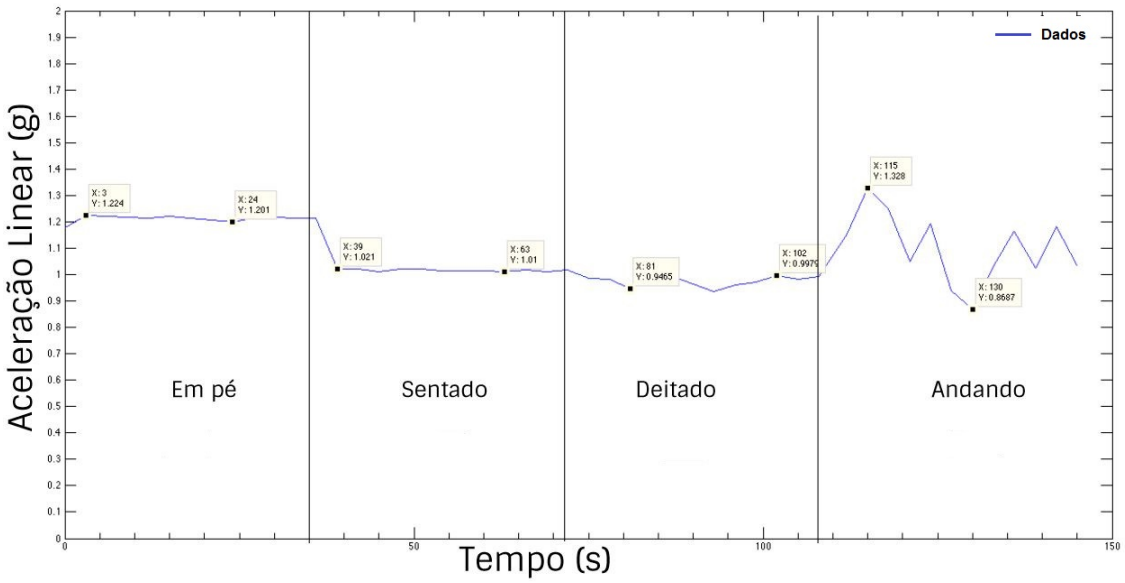


Figura 4.2: Gráfico dos movimentos estáticos X dinâmicos - Comportamentos dos vetores aceleração ao longo do tempo

Conforme mencionado no Capítulo 3, o presente projeto de mestrado poderia utilizar duas abordagens: baseado no *threshold* do valor resultante da integral trapezoidal ou utilizar o aprendizado de máquina para determinar se a posição do paciente é estática ou dinâmica. A

Figura 4.3 apresenta os resultados utilizados na primeira abordagem. É possível verificar que o valor da integral trapezoidal para movimentos dinâmicos é muito maior do que para movimentos estáticos, se tornando então, um bom parâmetro para ser utilizado na diferenciação de movimentos estáticos e dinâmicos.

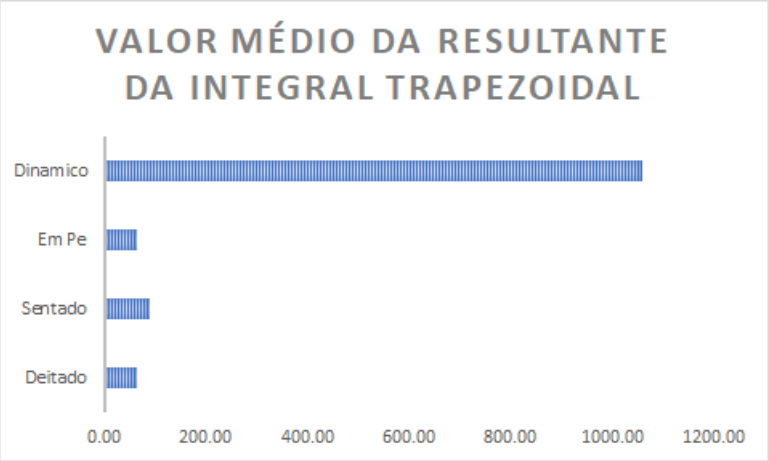


Figura 4.3: Comparativo de valor da Integral Trapezoidal

A Figura 4.4 apresenta os resultados utilizados na segunda abordagem, na qual foi utilizado o valor retornado pela função *K-Means*. É possível observar que o algoritmo divide os dados em *clusters* de acordo com o tipo de movimento predito. Cada *cluster* apresenta seu *centroid* e os valores pertencentes à sua região.

Os dados são divididos em cinco *clusters* : "sentado", "deitado", "em pé", "andando" e "caiu". É possível observar que a predição quando o corpo está "sentado" ou "em pé" são muito semelhantes e podem causar erros de predição. Esse é o caso em que o algoritmo não diferencia com acurácia essas posições, gerando os índices apresentados na Figura 4.19. Isso ocorre porque no dispositivo, o sensor MPU6050 tem o eixo coordenado "x" na mesma orientação para ambas as posições, conforme as Figuras 3.10 e 3.11. A diferença de quando se está "sentado" ou "em pé" é apenas a altura em relação solo. O presente trabalho de mestrado não conseguiu verificar como distinguir esses movimentos.

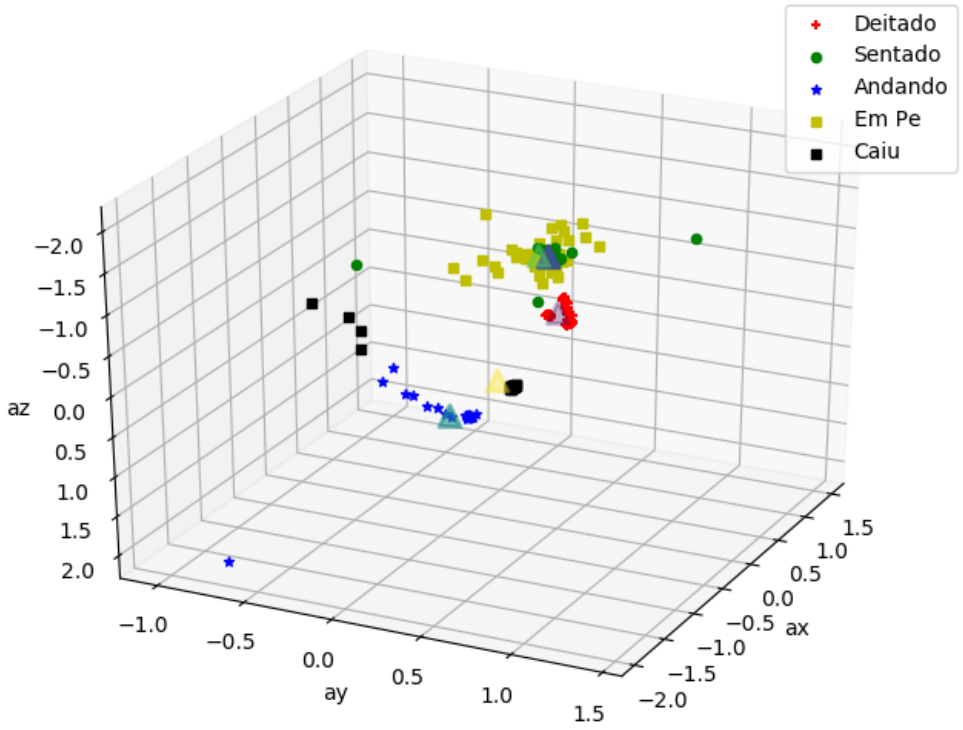


Figura 4.4: Gráfico de todos movimentos

Após determinar que o movimento realizado pelo paciente é dinâmico, a próxima tarefa é diferenciar os movimentos "queda" e "andando". A Figura 4.6 apresenta o comportamento do vetor obtido do giroscópio quando o paciente está nas posições "andando" e em "queda". Quando o corpo está "andando" o valor da área sob a curva no intervalo de maior e menor amplitude do vetor da velocidade angular é menor do que a área sob a curva do vetor da velocidade angular obtida quando o corpo está em "queda". O ponto de menor amplitude é representado pelo último ponto do vetor. Pode-se concluir então que para movimentos em "queda", há um brusco declínio nesse intervalo. Podemos mensurar esse valor através da regressão linear, obtendo o coeficiente angular da reta. Além disso, a integral corresponde à área sob a curva para o movimento "queda" é maior do que para quando o corpo está andando, como mostra a Figura 4.5.

```
area_andando_trapz = 464.341032545
area_caiu_trapz = 1097.48006882
```

Figura 4.5: Valor da intregral trapezoidal para movimentos dinâmicos

Após os testes, verificou-se que o valor ideal de *thresolds* para a área sob a curva e do coeficiente angular da reta para classificar o movimento como "queda" é: coeficiente angular superior a 0,5 e valor da integral trapezoidal maior do que 600.

O trabalho de graduação [2] utilizou *tresholds* baseados na variação média da aceleração própria e da velocidade angular. O presente trabalho de mestrado apresenta outro ponto de vista, que é a regressão linear e a área da integral calculada no intervalo de maior e menor amplitude do vetor obtido do giroscópio.

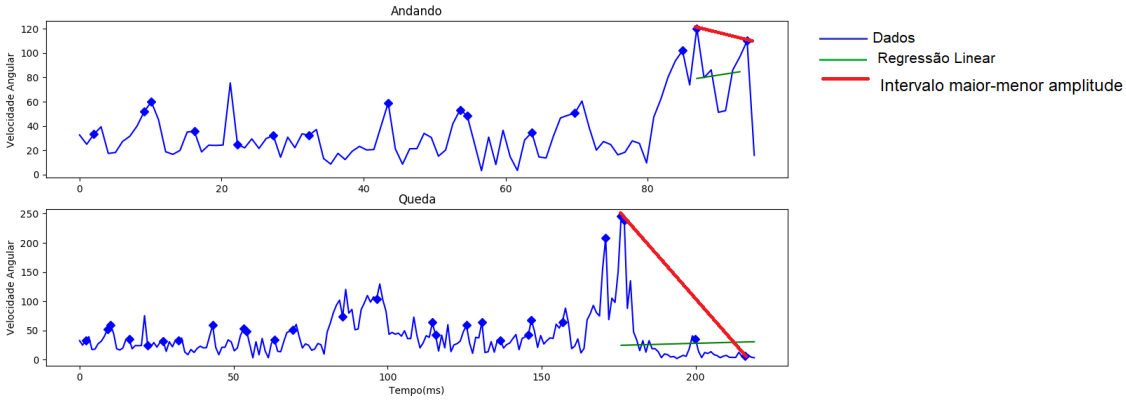


Figura 4.6: Gráfico de movimentos dinâmicos

O *K-Means* não forneceu uma predição confiável dos dados referentes aos movimentos dinâmicos. A Figura 4.7 apresenta a simulação dos dados coletados com o giroscópio quando o corpo encontra-se nas posições "andando" e "queda". Há uma pequena quantidade de dados na posição "caiu", representado pela cor verde, se comparado com a posição "andando", representado pela cor vermelho. Isso aconteceu porque o algoritmo *K-Means* clusterizou os dados de menor amplitude que faziam parte do movimento "queda", como o movimento "andando". Os pontos em verde representam o movimento quando o corpo está exatamente em "queda", com altos valores de amplitude. O esperado é que o algoritmo realizasse a clustarização dos movimentos "andando" e "queda" independentemente do valor da amplitude.

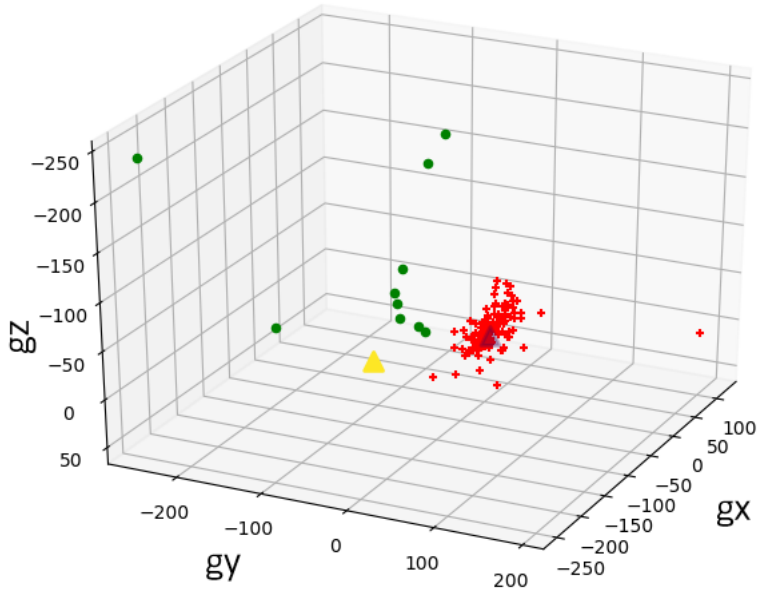


Figura 4.7: K-Means para Movimentos Dinâmicos

Foi decido utilizar nesse projeto a segunda abordagem, pois a mesma é uma evolução do método utilizado pelo trabalho anterior [2], realizando a diferenciação de movimentos estáticos e dinâmicos através de aprendizado de máquina utilizando o algoritmo *K-Means*.

Para determinar movimentos dinâmicos foram utilizados outros parâmetros de decisão do que os utilizados na graduação.

4.3 Aplicativo Android

Por se tratar de uma aplicação direcionada para a saúde o ícone do aplicativo pode ser observado na Figura 4.8. O aplicativo tem o nome fictício de "*My Position*".

Figura 4.8: Ícone do aplicativo

Ao iniciar o aplicativo, é apresentado um menu ao usuário conforme a Figura 4.9. O usuário deverá escolher quais das opções o mesmo deseja ser redirecionado. Primeiramente ele deverá calibrar o dispositivo, como apresenta a Figura 4.10.

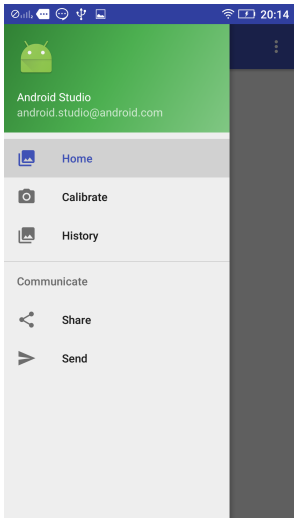


Figura 4.9: Menu do Aplicativo

O paciente deverá executar cada uma das posições apresentadas na Figura 4.10 e permanecer nas mesmas durante 40 segundos. Em *background*, são executados os quatro primeiros algoritmos do RaspberryPi apresentados na seção 3.3.3. O sensor MPU6050 coleta os valores referentes à aceleração própria e velocidade angular de acordo com as características físicas do paciente e as armazena no banco de dados. As informações entre o aplicativo Android e o RaspberryPi são trocadas através de *socket*. Para utilizar o *socket*, foi necessário atribuir o endereço IP (*Internet Address*) fixo "192.168.0.5" ao RaspberryPi. Isso foi necessário para que o servidor configurado no RaspberryPi fique escutando uma conexão cliente nesse IP, na porta fixa "8888".

A Figura 4.11 apresenta o aplicativo processando enquanto o paciente encontra-se na posição de calibração.

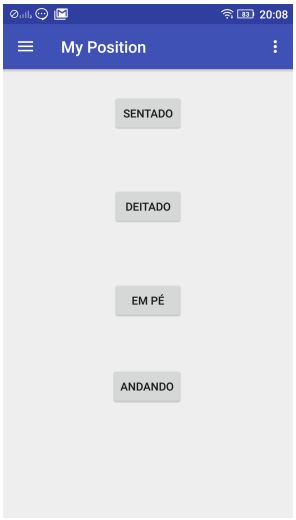


Figura 4.10: Tela para calibrar

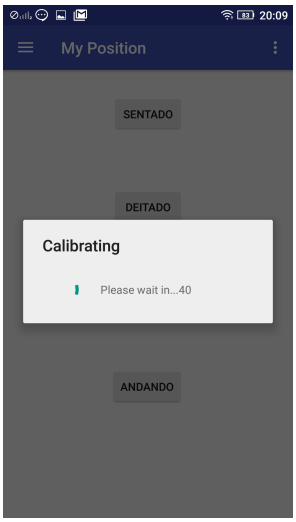


Figura 4.11: Troca de informações entre RaspberryPi e Aplicativo Andoird

O resultado da predição da posição do paciente é coletado do banco de dados e é apresentado ao usuário através do aplicativo Android. A Figura 4.12 mostra a resposta do aplicativo em todas as possíveis posições.

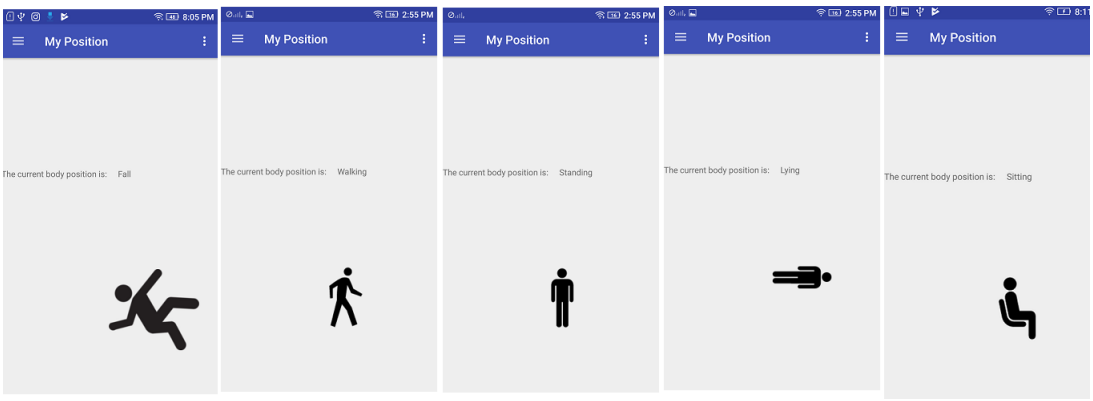


Figura 4.12: Apresentação de dados no aplicativo Android

Nesse aplicativo Android não é possível o paciente e o profissional da saúde realizarem algum tipo de alteração ou inserção de dados, pois todas as informações disponibilizadas são automaticamente atualizadas pelo sistema. A aplicação tem um tamanho total de 712KB.

4.4 Aplicação Web

Para atualizar a aplicação web e o aplicativo Android foi necessário enviar as informações preditas pelo algoritmo em Python para a base de dados. A cada nova requisição do usuário utilizando o aplicativo Android ou a página web, deverá ser realizada uma coleta da última posição do usuário cadastrada no banco de dados, para isso é necessário realizar tarefas agendadas através do *CronJob* que executa essa ação a cada 0,5 segundos, ou seja, esse é o tempo para que a informação seja atualizada nas interfaces.

A interface web foi desenvolvida através do servidor local Apache no RaspberryPi. A página inicial apresenta uma introdução do sistema, simulando um ambiente comercial, como pode ser verificado na Figura 4.13. As Figuras 4.14, 4.15, 4.16, 4.17 e 4.18, apresentam os dados da atual situação do paciente na página web.

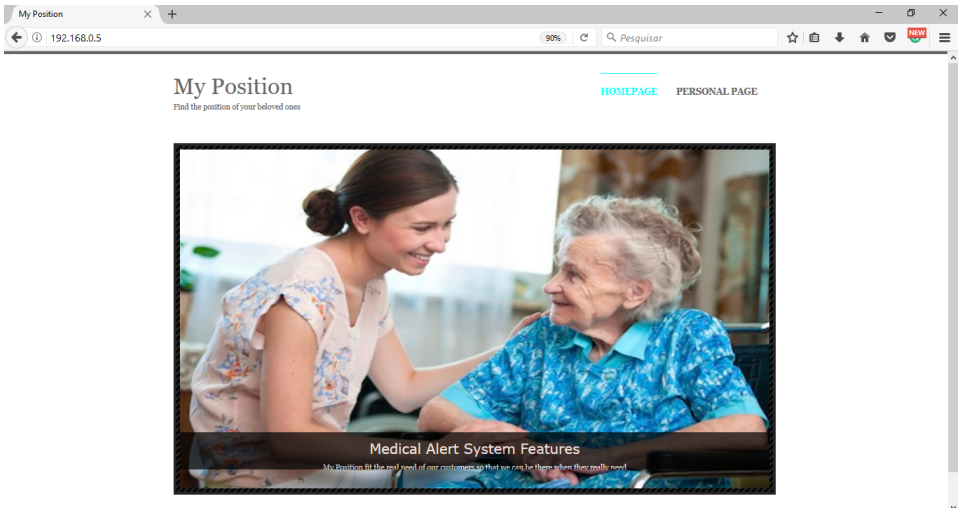


Figura 4.13: Página Web - Apresentação

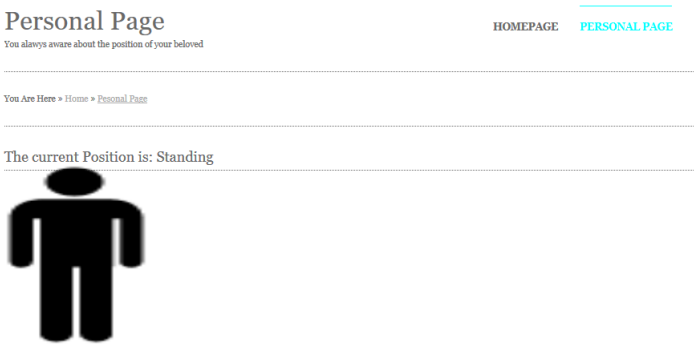


Figura 4.14: Página Web - Em Pé

Figura 4.15: Página Web - Deitado

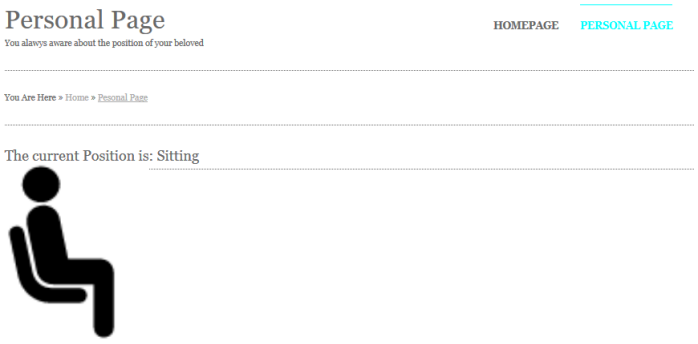


Figura 4.16: Página Web - Sentado

Figura 4.17: Página Web - Andando

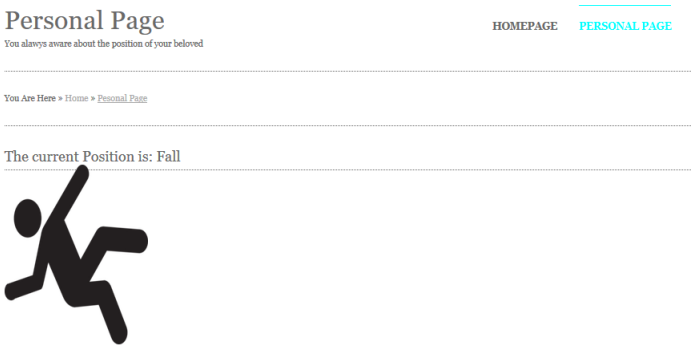


Figura 4.18: Página Web - Caindo

4.5 Validação de Resultados

Depois dos testes serem realizados, foi analisado a utilização do sistema para fins comerciais e sua aplicabilidade.

Constatou-se que não é possível utilizar o dispositivo para fins comerciais, pois a acurácia de um sistema composto por apenas um sensor é baixa para o grau de confiabilidade necessária para esse tipo de aplicação.

O trabalho de mestrado não diferenciou em 100% as posições estáticas "Sentado" e "Em Pé", devido ao fato de ambas apresentarem o eixo coordenado "x" do sensor MPU6050 na mesma orientação, ou seja, mesma aceleração própria, conforme as Figuras 3.10 e 3.11, diferenciando-se apenas a elevação em relação ao solo. O índice de acerto da posição "Sentado" foi de 60%, enquanto que na posição "Em Pé" o acerto foi de 40% como mostra a Figura 4.19. A posição estática "Deitado" foi corretamente identificada em todos os testes, pois essa posição tem a orientação do eixo coordenado "x" diferente das posições anteriores, conforme a Figura 3.12.

No projeto de graduação [2], foram utilizados dois equipamentos, um acoplado ao peito e outro na coxa. Esse projeto determinou corretamente todas as posições estáticas como mostra a Figura 4.19. Isso ocorreu porque as orientações do eixo coordenado "x" dos equipamentos acoplados no peito e na coxa se diferenciam em todas as posições, conforme a Figura 4.20 .

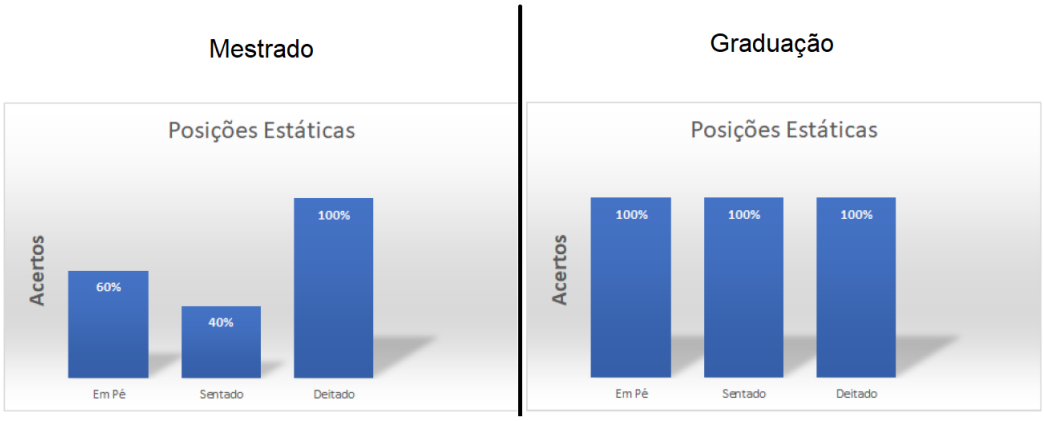


Figura 4.19: Acurácia do Sistema Estático

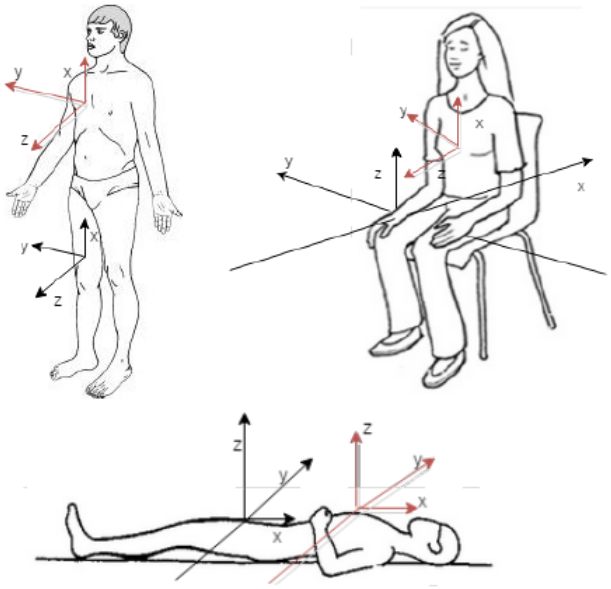


Figura 4.20: Planos de secção do corpo humano - Projeto de Graduação

A análise do comportamento dos sistemas desenvolvidos na graduação e no mestrado em relação aos movimentos dinâmicos seguem a metodologia apresentada na Figura 3.13. Os resultados obtidos para as posições dinâmicas em ambos os sistemas são equivalentes, como pode-se observar na Figura 4.21. O motivo para isso é o fato de apesar do projeto de mestrado utilizar *K-Means* para determinar se o movimento realizado é estático ou dinâmico, a decisão de qual movimento dinâmico está sendo realizado é baseada na mesma premissa utilizada na graduação, *threshold*. O projeto de mestrado utilizou parâmetros diferentes do trabalho de graduação com objetivo de aumentar a especificidade e sensibilidade do sistema, contudo o mesmo não foi efetivo porque os parâmetros continuam sendo ajustados manualmente, portanto são sujeitos a erros. Os parâmetros utilizados no projeto de mestrado são: regressão linear e integral trapezoidal. No projeto de graduação foi utilizado o valor médio dos vetores obtidos do acelerômetro e giroscópio.

Da mesma forma como ocorreu no projeto de graduação, há um maior número de acertos

para a posição "andando" do que para a posição "queda". A razão para isso, é que para o sistema identificar o movimento como "queda" duas condições baseadas em *threshold* devem ser satisfeitas. A primeira condição é baseada no valor do coeficiente angular obtido da regressão linear e a segunda condição é o valor obtido da integral trapezoidal.

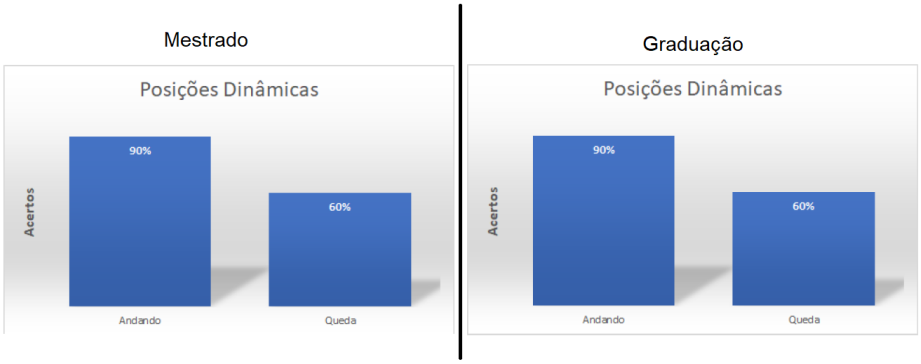


Figura 4.21: Acurácia do Sistema Dinâmico

Em relação à análise de erro, um dos erros apresentados pelo dispositivo é quando ocorrem quedas e o sistema afirma que o corpo está andando, nesse caso o sistema apresenta falsos negativos. O outro tipo de erro é quando o corpo está andando e ele afirma que ocorreu queda, nesse caso o sistema apresenta falsos positivos.

Os valores obtidos pelo giroscópio para movimentos dinâmicos, como andar e quedas, são aleatórios, ou seja, eles variam de acordo com o momento da coleta dos dados, como mostra a Figura 4.6. Esse projeto trabalha com padrões que são determinados para cada tipo de movimento, se algo ocorrer fora do padrão, a resposta do sistema poderá ser diferente do esperado. Isso faz com que o algoritmo apresente falhas. Falsos positivos são fenômenos conhecidos por algoritmos que detectam queda, pois ocorre um dilema, se eleva-se o *threshold*, mesmo em casas decimais, irá aumentar o número de falsos alarmes, se diminui-se o *threshold*, quedas podem não ser detectadas, então encontrar os valores que irão fornecer exatidão em 100% dos casos ainda não é possível, pois há várias atividades realizadas por uma pessoa que assemelham a uma queda, por exemplo, levantar rápido de uma cadeira ou andar em maior velocidade.

Dois conceitos são muito importantes no momento de se determinar a validade do sistema: sensibilidade e especificidade. No contexto de detecção de quedas, acurácia é medida através da sensibilidade e especificidade. Ambas são calculadas pelas razões descritas nas equações 4.1 e 4.2 :

$$Sensibilidade = \frac{Quedascorretamentedetectadas}{N^{o}totaldeQuedas} \tag{4.1}$$

$$Especificidade = \frac{ADLcorretamentedetectadas}{N^{o}totaldeADL} \tag{4.2}$$

No contexto desse trabalho, a sensibilidade obtida foi de 0,60 e a especificidade foi de 0,73. No trabalho de graduação, a sensibilidade foi de 0,60 e especificidade foi de 0,98. Nesse caso, no projeto de graduação, teve uma melhor especificidade, mostrando que teve uma melhor performance na detecção de ADL *(Activities of daily living*), enquanto que a sensibilidade continuou a mesma, ou seja, a mesma acurácia na detecção de queda. Esses números estão de acordo com os trabalhos relatados por [64]. Movimentos denominados ADL são identificados por aqueles que são realizados diariamente pelo corpo humano. No contexto desse trabalho, esses movimentos são: sentar, deitar, ficar em pé e andar.

Capítulo 5

Conclusão e Trabalhos Futuros

O tema desenvolvido neste trabalho de Mestrado, teve como objetivo principal construir um dispositivo capaz de detectar quedas e posicionamento corporal com capacidade de oferecer mobilidade ao usuário, além de apresentar as informações em tempo real via interface web e aplicativo Android. Podendo assim, contribuir para melhorar a qualidade de vida de idosos, pessoas que necessitam de monitoramento e de seus familiares.

No projeto de graduação [2] foi desenvolvido um sistema de detecção de quedas e posição corporais baseado em *threshold*, contudo a acurácia do mesmo não foi efetiva, logo, o primeiro objetivo do presente trabalho de mestrado é melhorar a sensibilidade e especificidade do sistema desenvolvido na graduação [2]. O segundo objetivo desse trabalho de mestrado é utilizar apenas um equipamento acoplado ao corpo do paciente para lhe fornecer maior mobilidade e usabilidade, em vez de utilizar dois dispositivos acoplados ao corpo como foi realizado na graduação. Além disso, o sistema desenvolvido na graduação não adequava o algoritmo para trabalhar com diferentes características físicas dos pacientes, o terceiro objetivo desse trabalho de mestrado é introduzir o conceito de calibração do algoritmo de tal forma a criar processos baseados nas características físicas do paciente.

Foi desenvolvido um aplicativo Android e uma interface web para monitorar as posições do paciente em tempo real. O projeto contemplou a construção do hardware e do software do equipamento. O dispositivo de monitoramento é composto por um sensor MPU6050 anexado ao RaspberryPi.

O algoritmo utiliza aprendizado de máquina para diferenciar posições estáticas e dinâmicas. Caso seja identificado uma posição dinâmica é realizada uma sequência de processos de decisões baseados em *threshold* para determinar se a posição executada é "andando" ou "queda". Os dados são calculados e armazenados localmente no RaspberryPi. Os *thresholds* foram baseados na regressão linear e na integral trapezoidal correspondente à área sob a curva no intervalo entre o valor de maior e menor amplitude do vetor obtido do giroscópio.

O equipamento apresentou uma precisão de 60% na detecção de posição estática, sendo que, destes foram obtidos 100% de acertos na detecção da posição "deitado", 60% da posição

"em pé" e 40% da posição "sentado". Isso ocorreu porque os valores da aceleração própria no eixo coordenado "x" para as duas últimas posições são semelhantes, diferenciando apenas a altura em relação ao solo. Por essa razão, *K-Means* realiza um falso positivo de que o paciente está em "em pé" mas na verdade, ele está "sentado".

Para posições dinâmicas, o sistema detectou corretamente que o paciente estava andando em 90% dos casos e detectou corretamente a posição queda em 60% dos casos. O sistema não conseguiu distinguir todos os casos dinâmicos porque o algoritmo utiliza *thresholds*, que são determinados manualmente, favorecendo o erro. Assim como no caso do trabalho de graduação [2] a probabilidade do algoritmo identificar o movimento como queda é baixa, pois o mesmo tem que satisfazer duas condições de *threshold* baseadas no valor da regressão linear e da integral trapezoidal. A escolha o algoritmo *K-Means* não melhorou a acurácia do trabalho de graduação [2], pois o mesmo não foi capaz de diferenciar as posições estáticas "sentado" e "em pé" devido ao fato de ambas apresentarem o mesmo plano de secção, além disso, ele não foi capaz de diferenciar as posições dinâmicas do paciente. Esse algoritmo foi escolhido com objetivo de verificar a clusterização dos dados obtidos e avaliar o comportamento do dispositivo quando a resposta é fornecida através de um algoritmo de aprendizado não supervisionado, ou seja, sem depender de um conjunto de treinamento baseado em rotulação.

Em geral, o sistema apresentou uma evolução em relação ao projeto de graduação por utilizar apenas um dispositivo acoplado ao corpo do paciente, melhorando a mobilidade e usabilidade. Além disso, também foi uma melhora em relação ao projeto de graduação a inserção do aprendizado de máquina para diferenciar posições dinâmicas e estáticas e realizar a calibração do algoritmo para o mesmo se adequar as diferentes características físicas do paciente. Contudo, os resultados obtidos não foram tão satisfatórios quanto o projeto de graduação para aferir posições estáticas, concluindo então que um sistema com maior quantidade de variáveis, como sensores na coxa e no peito, oferecem mais acurácia ao sistema mesmo utilizando aprendizado de máquina para determinar a posição do paciente. Além disso, sistemas de detecção de posição baseados em *threshold* não oferecem a acurácia necessária aplicações de detecção de quedas.

Como propostas para trabalhos futuros sugere-se realizar a etapa de calibração com o diferencial de aplicar os dados coletados no algoritmo supervisionado (*K-Nearest Neighbor* K-NN) ou Modelos Lineares, pois ambos oferecem uma abordagem de pré-rotulação com menos complexidade do que os algoritmos de Rede Neural Artificial e SVM. Os novos dados a serem coletados pelo acelerômetro e giroscópio seriam aplicados em um processo de decisão baseado no valor de entrada-saída. Além disso, é necessário acoplar dois sensores localizados em diferentes partes do corpo humano para realizar uma inferência com confiabilidade do movimento. A sugestão é colocar um sensor no peito e outro na coxa. Essas são as duas propostas de trabalhos futuros que possivelmente irão trazer maior acurácia ao sistema de detecção de posição e quedas corporais.

Referências Bibliográficas

[1] SCHWARTZ, R. *Número de pessoas que moram sozinhas quase dobrou nos últimos 10 anos*. Disponível em: <https://www.gazetaonline.com.br/noticias/cidades/2017/04/numero-de-pessoas-que-moram-sozinhas-quase-dobrou-nos-ultimos-10-anos-1014042907.html>.

[2] MALHEIROS, L. *Sistema de Detecção de Quedas e de Posicionamento Corporal com Monitoramento de Batimentos Cardíacos*. Dissertação (Trabalho de Graduação) — Universidade de Brasilia, Faculdade de Tecnologia, 2015.

[3] DIETERLE, F. *Multianalyte Quantifications by Means of Integration of Artificial Neural Networks, Genetic Algorithms and Chemometrics for Time-Resolved Analytical Data*. Tese (Doutorado) — University Tübingen, 2003.

[4] GUIDO, A. C. . S. *Introduction to Machine Learning with Python*. [S.l.]: O'Reilly Media, 2016. ISBN 9781491917213.

[5] LINDEN, R. *Algoritmos Genéticos*. [S.l.]: BRASPORT, 2008. 150–154 p. ISBN 9788574523736.

[6] TAN, M. S. P.-N.; KUMAR, V. *Introduction to Data Mining*. [S.l.]: PEARSON, 2014.

[7] CORPORATION, I. T. S. O. I. *TCP/IP Tutorial and Technical Overview*. 1995. Disponível em: <http://www.danzig.jct.ac.il/tcp-ip-lab/ibm-tutorial/3376c212.html>.

[8] VARANIS, e. a. Instrumentation for mechanical vibrations analysis in the time domain and frequency domain using the Arduino platform. *Revista Brasileira de Ensino de Física*, v. 38, n. 1, p. 1301–1–1301–10.

[9] SUSNEA, I.; MITESCU, M. *Microcontrollers in Practice*. [S.l.]: Springer, 2005. ISBN 14370387.

[10] ALEX. *Raspberry Pi Zero – Updated Pi Family Photo*. Disponível em: <http://raspi.tv/2015/11>.

[11] STEPHANIE. *Raspberry Pi Project Build 1 - Controlling the Hardware Remotely With Socket.IO*. Disponível em: <http://newcodegirl.blogspot.com.br/2016/11/raspberry-pi-project-build-1.html>.

[12] SAFIER, F. *Pré-Cálculo*. [S.l.]: Bookman, 2011.

[13] SHI, T. e. a. Fall detection algorithm based on triaxial accelerometer and magnetometer. In: *Conference: E-Health and Bioengineering Conference (EHB)*. [S.l.: s.n.].

[14] MALHEIROS L.; NZE, G. C. L. Fall detection system and Body positioning with Heart Rate Monitoring. *IEEE Latin America Transactions*, v. 15, n. 6, p. 1021 – 1026.

[15] KOSTOPOULO, P. e. a. Increased Fall Detection Accuracy in an Accelerometer-Based Algorithm Considering Residual Movement. *International Conference on Pattern Recognition Applications and Methods*, 2015.

[16] SEIFERT, K.; CAMACHO, O. Implementing Positioning Algorithms Using Accelerometers. *Sensors*, 2007.

[17] GASPARRINI, S. e. a. A Depth-Based Fall Detection System Using a Kinect® Sensor. *Sensors*, 2014.

[18] GUIMARAES, R. M. *Desenvolvimento de um protótipo de software de reconhecimento facial de tempo real para registro eletrônico de ponto em ambientes indoor com utilização do dispositivo kinect*. Dissertação (Mestrado) — Faculdade de Ciências Empresariais, Belo Horizonte, 2015.

[19] SHRIVASTAVA, R.; PANDEY, M. Human Activity Recognition by Analysis of Skeleton Joint Position in Internet of Things (IOT) Environment. *Indian Journal of Science Tecnology*, 2017.

[20] PRINCIPI, E. e. a. Acoustic cues from the floor: A new approach for fall classification. *Expert Systems with Applications: An International Journal*, v. 60, p. 51–61.

[21] AYYOOB.MP. Tools Pros and Cons of Clustering Algorithms using Weka Tools. *International Journal of Computer Applications*, p. 4–15, 2015.

[22] ADHIKARI K.; BOUCHACHIA, H. N.-C. H. Activity recognition for indoor fall detection using convolutional neural network. *IEEE IAPR International Conference on Machine Vision Applications*, 2017.

[23] MAXWELL. *Aprendizado por reforço*. [S.l.]: Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, 2017.

[24] HSU C.; WANG, M. S.-H. FallCare+: An IoT surveillance system for fall detection. *International Conference on Applied System Innovation (ICASI),*, p. 921–922, 2017.

[25] SENA, M. *Impacto Financeiro e de Qualidade de Vida Devido a Quedas de Idosos*. Disponível em: <www.seniorconcierge.com.br>.

[26] J, B.; AMIN, M. Fall Detection Using Deep Learning in Range-Doppler Radars. *IEEE Transactions on Aerospace and Electronic Systems* , PP, n. 99, p. 1–1.

[27] REZENDE, S. O. *Sistemas inteligentes: fundamentos e aplicações*. [S.l.]: Manole, 2005. ISBN 8520416837.

[28] FREITAS, A. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. [S.l.]: Springer, 1998. ISBN 3540433317.

[29] BEN-HUR ASA, A. E.; GUYON, I. A Stability Based Method for Discovering Structure in Clustered Data. *Pacific symposium on biocomputing* , 2002.

[30] GOLDSCHMIDT, E. P. R. *Data Mining*. [S.l.]: Campus, 2015. ISBN 9788535278224.

[31] REZENDE, S. O. *Sistemas Inteligentes, Fundamentos e Aplicações*. [S.l.]: Manole, 2005. 150-151 p. (International series of monographs on physics). ISBN 8520416837.

[32] KAREM, F.; DHIBI, M.; MARTIN, A. Combination of supervised and unsupervised classification using the theory of belief functions. *Belief Functions: Theory and Applications*, v. 164, p. 85–92, 2012.

[33] IGUAL, E. e. a. Challenges, issues and trends in fall detection systems. *BioMedical Engineering OnLine*, p. 14–15, 2013.

[34] HOHENSEE, B. *Getting Started with Android Studio*. [S.l.: s.n.], 2013. ISBN 3540433317.

[35] DAPONT, P. *Curso de HTML com PHP*. Disponível em: <http://www.inf.ufrgs.br/ fdmschmidt/coisas/inf/apostilaHTML.pdf>.

[36] SEBESTA, R. W. *Conceitos de Linguagens de Programação*. [S.l.]: ATMED, 2010. 122–131 p. ISBN 9780136073475.

[37] BONATTI, D. *Desenvolvimento de Sites Dinâmicos com Dreamweaver CC*. [S.l.]: BRASPORT, 2013. ISBN 9788574526133.

[38] DUBOIS, P. *MySQL*. [S.l.: s.n.], 2013. ISBN 0321833872.

[39] JOBSTRAIBIZER, F. *Desvendando as redes sem fio*. [S.l.: s.n.], 2010. ISBN 9788578731011.

[40] LAGE, B. *Aprendendo a programar em Python - Introdução*. Disponível em: <https://www.devmedia.com.br/aprendendo-a-programar-em-python-introducao/17093>.

[41] BASSI, S. *Python for Bioinformatics*. [S.l.: s.n.], 2009. ISBN 1584889292.

[42] SPYDER - Documentation. Disponível em: <https://pythonhosted.org/spyder/>.

[43] SMITH, R. W. *Linux+ Study Guide*. [S.l.: s.n.], 2007. ISBN 1119021219.

[44] JOSEPH, L. *Learning Robotics Using Python*. [S.l.]: Packt, 2015. 152–165 p. ISBN 9781783287536.

[45] SCOTTS, J. *Raspberry Pi :Raspberry Pi Guide On Python Projects Programming In Easy Steps*. [S.l.]: Speedy Publishing, 2013. ISBN 9781628847437.

[46] SEBER, G. A. F.; LEE, A. J. *Linear Regression Analysis*. [S.l.]: Wiler, 2003. ISBN 0471415405.

[47] MOIN, P. *Engineering Numerical Analysis*. [S.l.]: Cambridge, 2010. 30–34 p. ISBN 9780521884327.

[48] VIEIRA, M.; OLIVEIRA, H. *Transformada Wavelet e suas Aplicações no Processamento de Imagens*. Departamento de Engenharia Elétrica - Universidade de São Paulo.

[49] LI, e. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. *IEEE Latin America Transactions*, p. 3–7.

[50] DELAHOZ, Y.; LABRADOR, M. Survey on Fall Detection and Fall Prevention Using Wearable and External Sensors. *Sensors*, p. 5–20, 2014.

[51] LIM, e. a. Fall-Detection Algorithm Using 3-Axis Acceleration: Combination with Simple Threshold and Hidden Markov Model. *Journal of Applied Mathematics*, v. 2014, n. 896030, 2014.

[52] MAXWELL. *Modelos de Markov Ocultos*. [S.l.]: Pontifícia Universidade Católica do Rio de Janeiro - PUC-Rio, 2017.

[53] SHEN, V.; LAI, H.; LAI, A. Application of High-Level Fuzzy Petri Nets to fall detection system using smartphone. *Machine Learning and Cybernetics*, v. 38, p. 1–3.

[54] SHEN, V.; LAI, H.; LAI, A. The implementation of a smartphone-based fall detection system using a high-level fuzzy Petri net. *Applied Soft Computing*, v. 26, p. 390–400.

[55] LUšTREK, M.; KALUžA, B. Fall Detection and Activity Recognition with Machine Learning. *International Journal of Computing and Informatics*, v. 33, p. 205–212.

[56] DINH, C.; STRUCK, M. A new real-time fall detection approach using fuzzy logic and a neural network. *Wearable Micro and Nano Technologies for Personalized Health (pHealth) International Workshop*, 2009.

[57] MUNDHER, Z.; ZHONG, J. A Real Time Fall Detection System in Elderly Care Using Mobile Robot and Kinect Sensor. *International Journal of Materials, Mechanics and Manufacturing*, v. 2, p. 133–137.

[58] TORRES, R. e. What if Your Floor Could Tell Someone You Fell? A Device Free Fall Detection Method. *Artificial Intelligence in Medicine*, p. 86–95.

[59] YANG, C.; HSU, Y. A Review of Accelerometry-Based Wearable Motion Detectors for Physical Activity Monitoring. *Sensors*, p. 7772–7788.

[60] INVENSENSE. *MPU6050 Product Specification Revision 3.4*. Disponível em: <https://store.invensense.com/datasheets/invensense/MPU-6050$_{D}ataSheet_{V}3$

[61] FOWLER, M. *UML Essencial: Um breve guia para a linguagem padrão de modelagem de objetos*. [S.l.]: Bookman, 2005. 52–60 p. ISBN 0321193687.

[62] KANGAS, E. e. a. Comparison of low-complexity fall detection algorithms for body attached accelerometers. *Gait Posture*, v. 28, p. 285–291, 2008.

[63] ROUGIER, E. e. a. Fall Detection from Human Shape and Motion History Using Video Surveillance. *AINAW '07 Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, v. 2, p. 875–880, 2007.

[64] HU, X.; QU, X. Pre-impact fall detection. *BioMedical Engineering OnLine*, 2016.