SafeWatch: Sistema de Detecção de Quedas para Smartwatches

Por

Victor de Souza Tavares

Trabalho de Graduação

Universidade Federal da Bahia
wiki.dcc.ufba.br/DCC/

SALVADOR, Outubro/2016

Universidade Federal da Bahia
Departamento de Ciência da Computação

Victor de Souza Tavares

SafeWatch: Sistema de Detecção de Quedas para Smartwatches

Trabalho apresentado ao Departamento de Ciência da Computação da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: *Vaninha Vieira dos Santos*

SALVADOR, Outubro/2016

Dedico esta dissertação à minha família, namorada e professores que me deram todo o apoio necessário para chegar até aqui.

The most profound technologies are those that dissapear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

—MARK WEISER

Resumo

Quedas podem ter sérias consequências, a ponto de serem consideradas um grave problema de saúde pública que afeta principalmente a população idosa, onde está relacionado com a perda de confiança, autoestima, e autonomia. Este problema se mostra ainda mais relevante se consideramos o crescente número de idosos, que em busca de sua independência e autonomia decidem morar sozinhos. É crucial que o idoso tenha rápido acesso ao atendimento médico, parte importante para a sua rápida recuperação. A demora no atendimento médico está ligado ao aumento de taxas de mortalidade e gravidade em um evento de queda.

Pensando nisso, foi desenvolvido o *SafeWatch*, um sistema de detecção de quedas embarcado em relógios inteligentes (em inglês, Smartwatch). O sistema proposto irá monitorar o idoso através de sensores presentes no smartwatch, e ao detectar uma queda, além de vibrar no pulso do usuário, irá informar para uma lista de contatos de emergência do usuário a sua localização e a possibilidade do idoso estar em uma situação de perigo. Experimentos foram realizados com oito indivíduos de biótipos distintos, onde cada um deles deveria simular uma queda em sentidos distintos. Através deste experimento, foi possível detectar o grau de confiabilidade da aplicação utilizando os valores de *Sensibilidade* e *Especificidade* que atingiram 89.06% e 100% respectivamente.

Palavras-chave: Relógios Inteligentes, Computação Ubíqua, Queda, Idoso

Abstract

Falling can have serious consequences, enough to be considered a serious public health problem that affects mainly the older population, in which is related to the loss of confidence, self-esteem and autonomy.This problem is shown even more relevant if we consider the growing number of seniors, who in the search of their independence and autonomy decide to live alone. It is crucial that the elderly have quick access to medical care, a key part for quick recovery. The delay in medical care is linked with the increase of mortality rates and severity in a fall event.

Thinking about it, the *SafeWatch* was developed as a fall detection system embedded in smartwatches. The proposed system will monitor the seniors through sensors at the smartwatch, and when a fall is detected, It will vibrate on the user's wrist and report to a list of emergency contacts their location and inform the possibility of the elderly to be in a dangerous situation. Experiments were performed with eight individuals of different biotypes, in which, each one of them, simulated a fall event in different directions. According to the experiments, it was possible to evaluate the application reliability through the values of *Sensitivity* and *Specificity* that reached 89,06% and 100%, respectively.

Keywords: Smartwatch, Ubiquitous Computing, Fall, Elderly

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

PNAD	Pesquisa Nacional por Amostra de Domicílio
SDQ	Sistema de Detecção de Queda
AD	Atividades diárias
FOF	Fear of Falling
MEMS	Sensores Microeletromecânicos
SMV	Magnitude Vetorial
SMA	Soma das Acelerações
BLE	Bluetooth Low Energy
IDE	Ambiente de Desenvolvimento Integrado

Lista de Códigos Fonte

1 Introdução

First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives.

—MARK WEISER

Devido a diversos avanços tecnológicos e médicos, a população mundial vem envelhecendo de forma gradual. Projeções feitas pelas United Nations (2013), indicam que 11,57% da população mundial tem 60 anos ou mais. Este mesmo relatório aponta que em 2050 a porcentagem de idosos irá quase dobrar, correspondendo a 21,1% da população. Esta tendência não é muito diferente no Brasil, onde de acordo com as projeções do IBGE (2016), 8,17% da população irá ter 65 anos ou mais em 2016, com este número aumentando para 13,44% em 2030.

Além de ser uma parcela da população que cresce, o número de idosos que moram sozinhos também vem aumentando. De acordo com o IBGE (2012), entre 1992 e 2012, este número triplicou no Brasil, passando de 1,1 milhão para 3,7 milhões, um aumento de 215%. Na busca pela sua independência, o idoso fica vulnerável a um dos principais problemas desta faixa etária, as quedas. De acordo com um estudo da Organização Mundial de Saúde, de 28% a 35% da população maior do que 64 anos sofrem pelo menos uma queda por ano. De acordo com o Portal Brasil (2012), o SUS (Sistema Único de Saúde) registra a cada ano, um gasto de mais de R$ 51 milhões com o tratamento de fraturas decorrentes de queda.

Outra questão que torna as quedas ainda mais prejudiciais à saúde física e mental

do idoso é o tempo entre a queda e o atendimento médico. De acordo com Stephen R. Lord (2001), a demora no atendimento está fortemente ligada ao índice de mortalidade e gravidade do acidente depois de uma queda. Na ocorrência do "long-lie", ou seja, o idoso permanece mais de uma hora no chão, a chance de que o idoso faleça antes dos 6 meses do ocorrido sobe para 50% (Wild *et al.*, 1981).

Visando minimizar essas graves consequências das quedas, diversos sistemas de detecção foram desenvolvidos nos últimos anos, porém estes sistemas não fazem uso de tecnologias mais popularizadas no mercado, ou utilizam plataformas que não são vestíveis, prejudicando a mobilidade do usuário. Por exemplo, diversos sistemas utilizam o smartphone como principal plataforma na detecção de quedas. Analisando somente as questões de popularidade e hardware, o smartphone se apresenta como uma solução plausível. De acordo com Gartner (2015), foram vendidos mais de 1 bilhão de aparelhos somente em 2014. Na perspectiva de hardware, a maioria dos smartphones modernos possuem giroscópio ou acelerômetro, dois dos principais sensores utilizados no reconhecimento de atividades atualmente.

Entretanto, quando pensamos em mobilidade, o smartphone passa a ser uma solução com baixo potencial, pois para que os sistemas funcionem corretamente, os mesmos precisam estar fixos em uma posição pré-estabelecida, como o bolso ou pulso do usuário (Yi He, 2012). Isso faz com que, em situações corriqueiras, como colocar o celular na bolsa, possa gerar um número grande de falsos positivos.

Este trabalho propõe como solução a criação de um sistema de detecção de quedas através de uma solução integrada entre smartphone e o smartwatch. O smartwatch será responsável pela detecção dos eventos de queda, enquanto o smartphone será responsável pelo gerenciamento dos contatos e envio das mensagens de emergência.

O smartwatch é uma ferramenta que permite que este tipo de aplicação seja calma e invisível para o usuário, além de ter uma capacidade de processamento bastante similar aos smartphones, com uma popularidade crescente. A Samsung, umas das empresas pioneiras no mercado de smartwatches, lançou em outubro de 2015 o Samsung Gear S2. O Gear é um exemplo de como esses sistemas estão cada vez mais poderosos. Ele possui uma memória RAM de 512 MB e 4GB de armazenamento, conectividade WiFi e 4G além de diversos sensores como giroscópio e acelerômetro Samsung (2016). A popularidade desta plataforma é vista através do número de smartwatches vendidos. No ano de 2015, 30,32 milhões de aparelhos foram vendidos, e a previsão é de que, em 2016, este número suba para 50,40 milhões.

Este trabalho ter por objetivo desenvolver um sistema de detecção de quedas uti-

lizando os smartwatches já popularizados no mercado como ferramenta principal no desenvolvimento desta solução. O sistema deverá exigir o mínimo de iteração possível do usuário, sendo capaz de detectar as quedas de maneira automática. Também é desejável que o sistema utilize o mínimo de sensores possível, visando diminuir o custo computacional da solução apresentada, mantendo-se, a acurácia semelhante aos principais sistemas de detecção presentes na literatura.

Para realizar este trabalho a seguinte metodologia foi adotada: inicialmente investigamos os sistemas de detecção de quedas existentes na literatura, como foco nos algoritmos de detecção utilizados. Em seguida desenvolvemos o *SafeWatch*, utilizando um algoritmo de detecção de quedas que utiliza somente os dados do acelerômetro como entrada. Por fim, elaboramos um experimento para avaliar a solução proposta.

Participaram do nosso experimento oito pessoas, sendo três homens e cinco mulheres. Simulamos quatro tipos de queda e quatro tipos de atividades diárias. Os resultados indicaram que o sistema possui uma acurácia de 94,17%.

Os próximos capítulos estão organizados da seguinte maneira: O Capítulo 2 apresenta os conceitos teóricos usados neste trabalho referente a Sistemas de Detecção de Quedas. O Capítulo 3 se aprofunda nos sistemas de detecção de quedas que utilizam plataformas vestíveis; O Capítulo 4 apresenta o SafeWatch, o sistema de Detecção de Quedas desenvolvido através de uma solução integrada entre o smartphone e o smartwatch. O Capítulo 5 apresenta o experimento realizado, e realiza a avaliação da ferramenta. Por fim, no capítulo 6, seguem as conclusões e considerações finais.

2

Sistemas de Detecção de Queda

Um Sistema de Detecção de Queda (SDQ), pode ser descrito como um dispositivo de apoio, cujo principal objetivo é alertar o usuário em um evento de queda (Igual *et al.*, 2013). De forma geral, estes sistemas são capazes de distinguir AD de um evento de queda. Este tipo de sistema pode ser desenvolvido de diversas formas e pode estar tanto embarcado em um dispositivo vestível como um smartwatch , se basear em um sistema de monitoramento utilizando câmeras ou através de sensores posicionados ao redor de um ambiente selecionado.

Com o uso de sistemas de detecção de quedas é possível que o usuário tenha o seu medo de cair reduzido, e dependendo da solução que foi desenvolvida, possa ser socorrido de maneira muito mais rápida, caso necessário. Um indivíduo que já sofreu uma queda, pode desenvolver uma sindrome chamada Fear of Falling (FOF), que pode levar a perda da capacidade de realizar atividades rotineiras, como passear em um parque, ou assistir um filme em família (Legters, 2002).

As seções desse capítulo são organizadas da seguinte maneira: A Seção 2.1 define o conceito de quedas e expõe os diversos estados da mesma; A Seção 2.2 irá demonstrar os três diferentes tipos de sistemas de detecção quedas mais populares.

2.1 Definição de Queda

De forma geral, podemos definir uma queda como um evento súbito e involuntário, onde o indivíduo de uma posição em pé ou sentado, passa a ocupar uma posição integral ou parcialmente deitada (Igual *et al.*, 2013). Uma outra definição mais formal de queda descreveu este evento como "Vir ao chão ou algum nível mais baixo, sem a intenção, como consequência de um golpe violento, perda de consciência, ou início súbito de paralisia, como no caso de um acidente vascular cerebral ou um ataque epiléptico"(Gibson *et al.*,

1987). Quando analisamos uma queda através da aceleração do movimento, ela pode ser dividida em etapas descritas na imagem 2.1. Estas 4 etapas são as seguintes:

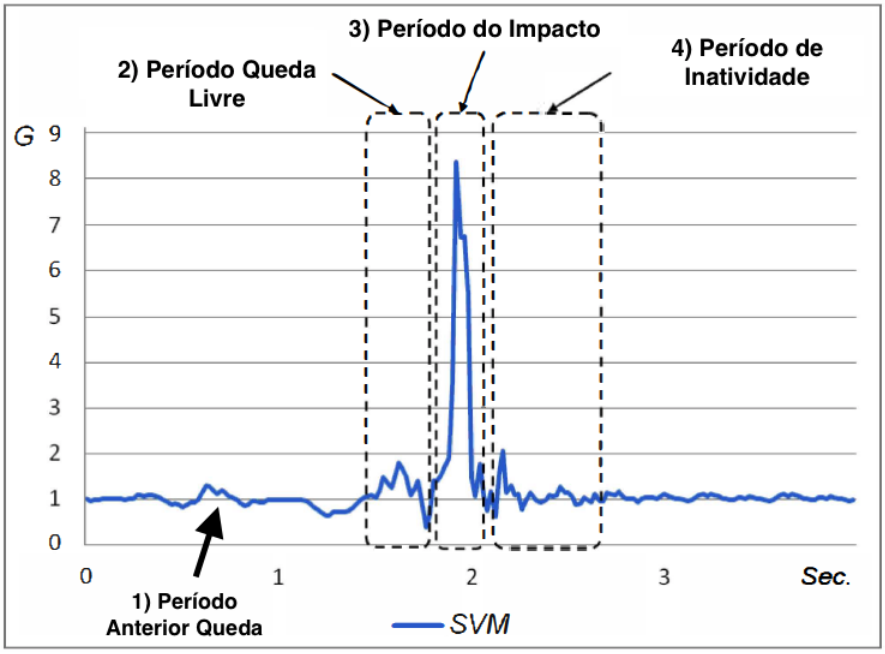

Figura 2.1 Etapas de uma queda (Hsieh *et al.*, 2014).

1. **Período Anterior à Queda**: Durante este período o indivíduo estará realizando suas atividades cotidianas, que podem levar ou não a um pico de aceleração que deve ser tratado para que se possa evitar falso-positivos.

2. **Período Queda Livre**: Durante este período o indivíduo está se descolando em direção ao chão. Nesta fase o valor de sua aceleração irá tender a 0.

3. **Período do Impacto**: Período caracterizado pelo impacto do índivíduo, este período é crítico na aplicação, pois é onde ocorre o pico de aceleração.

4. **Período de Inatividade**: Período posterior à queda, onde o usuário irá realizar o esforço para se levantar. Em quedas mais graves, onde o usuário está incapaz de se movimentar ou inconsciente, este valor se modifica de maneira muito sutil, porém de forma geral, o usuário que sofreu uma queda não se levanta imediatamente. De acordo com Mehner *et al.* (2013), o período de pós impacto e inatividade leva aproximadamente 2 segundos.

2.2 Tipos de Sistemas de Detecção de Queda

Diversos tipos de Sistemas de Detecção de Queda foram desenvolvidos nos últimos anos e estes utilizam diferentes abordagens buscando atingir o mesmo objetivo: realizar a detecção automática das quedas. De acordo com Mubashir *et al.* (2013), estes sistemas podem ser categorizados em três grupos: sistemas de detecção baseados no ambiente, sistemas de detecção baseados na visão, sistemas de detecção baseados em tecnologias vestíveis.

2.2.1 Sistemas de Detecção Baseados no Ambiente

Sistemas de detecção baseados no ambiente utilizam a fusão de dados obtidos de diversos sensores para realizar a detecção de quedas. Através desses sensores, são obtidos sinais áudio-visuais e dados vibracionais do ambiente que está sendo monitorado. Este tipo de sistema pode ser dividido em duas categorias:

- **Áudio-Visuais**:Neste tipo de sistema são analisados os sinais áudio-visuais obtidos através de câmeras e microfones espalhados no ambiente desejado. Um exemplo deste sistema foi proposto por Zhuang *et al.* (2009), ele utiliza o padrão da onda sonora capturada, com uma base de dados treinada com diferentes tipos de onda sonoras associadas com diferentes tipos de eventos. Sendo assim, é capaz de diferenciar uma queda de uma atividade diária.

- **Dados Vibracionais**: Neste tipo de sistema são analisados os dados vibracionais obtidos através de sensores de vibração espalhados no chão do ambiente desejado. Um exemplo deste sistema foi proposto por Alwan *et al.* (2006), ele utiliza o padrão da onda sonora capturada, com uma base de dados treinada com diferentes tipos de onda sonoras associadas com diferentes tipos de eventos. Sendo assim, capaz de diferenciar uma queda de uma atividade diária.

2.2.2 Sistemas de Detecção Baseados na Visão

Sistemas de detecção baseados na visão utilizam uma ou mais câmeras posicionadas ao redor do ambiente desejado para que se possa realizar a detecção de quedas. As câmeras são consideradas um meio menos intrusivo de detecção, pois, diferente dos sistemas que utilizam tecnologias vestíveis, somente o vídeo gerado pela câmera são utilizadas na detecção, sem a necessidade do usuário vestir nenhum dispositivo eletrônico. Diferentes

tipos de técnicas de processamento de vídeo e imagem são utilizadas na detecção. As principais delas são:

- **Espaço-Temporais**: Sistemas que utilizam características espaço-temporais para realizar modelagens capazes de fornecer dados cruciais na detecção de diferentes tipos de atividades. Um sistema proposto por Foroughi *et al.* (2008) realiza a extração de informações de movimento de uma sequência de vídeo. Aplicando uma técnica chamada Eigenspace sobre as informações de movimento coletadas, é possível extrair um vetor de características, que é utilizado por um algoritmo de inteligência artificial, mais especificamente um algoritmo de redes neurais, para determinar o tipo de evento ocorrido.

- **Inatividade/Mudança de Forma**: Sistemas que utilizam mudanças de forma e ausência de atividade no monitoramento em vídeo para realizar a detecção de quedas. Um exemplo deste tipo de sistema foi proposto por Rougier *et al.* (2011). Em seu artigo, são utilizadas técnicas de detecção e análise de formas para detectar a silhueta e atividade (representado por mudanças de forma) do indivíduo que está sendo monitorado. Para que se possa diferenciar quedas das atividades diárias, é utilizado o método de *Mistura de Modelos Gaussianos*, um método estatístico utilizado em visão computacional.

- **Postura**: Sistemas que identificam e localizam diversas posturas do indivíduo, calculadas através das diferentes posições corporais, utilizando uma sequência de imagens para realizar a detecção de um evento de queda. Cucchiara *et al.* (2005), desenvolveu um sistema de detecção que analisa os histogramas das imagens geradas pelas câmeras, para classificar as posturas do indivíduo monitorado e consequentemente detectar um evento de queda.

- **Análise 3D da Posição da Cabeça**: Sistemas que realizam o monitoramento da cabeça do indivíduo, e através de modelos de estado é capaz de detectar as magnitudes do movimento realizado. Um exemplo deste tipo de sistema foi proposto por Rougier (2005). Ele realiza a modelagem 3D da cabeça do indivíduo e é capaz de calcular a velocidade e a trajetória deste modelo que são posteriormente utilizadas na categorização do evento como queda.

2.2.3 Sistemas de Detecção Baseados em Tecnologias Vestíveis

Sistemas vestíveis de detecção de quedas utilizam os dados de sensores que estão acoplados sobre ou na roupa dos usuários. Este tipo de sistema apresentam claras vantagens sobre tanto os sistemas baseados no ambiente ou na visão. Ambos os tipos de sistema, exigem um custo constante de manuntenção, que dependendo de como o sistema foi desenvolvido, pode ser bastante alto, além disso a área de atuação do sistema fica limitada a uma área pré-determinada (e.g quarto, sala de estar). Utilizando tecnologias véstiveis não temos este problema, pois o usuário poderá leva o seu dispositivo para onde ele desejar.

Levando em consideração a privacidade do usuário, os sistemas vestíveis levam vantagem em relação aos baseados no ambiente e na visão. A necessidade de uma monitoração por vídeo constante necessária em alguns desses sistemas podem deixar o usuário relutante em implementar está solução.

De acordo com Igual *et al.* (2013), o uso de smartphones para realizar a detecção de quedas tem sido uma tendência devido ao seu baixo preço, altos volumes de produção e facilidade de desenvolvimento. O grande problema do uso do smartphone é justamente o seu posicionamento, para que este possa realizar a detecção de quedas, ele precisa está ligado diretamente ao corpo do usuário, se comportando como uma plataforma vestível, o que o smartphone não é em sua essência. Está incerteza dificulta bastante o uso deste tipo de dispositivo no reconhecimento de atividades.

Tentando resolver este problema, o uso de dispositivos realmente vestíveis vem sendo utilizado em sistemas de detecção de quedas, como pode ser visto nos trabalhos desenvolvidos por Hsieh *et al.* (2014) e Degen *et al.* (2003). Estes trabalhos serão descrito com mais detalhes no capítulo 3, juntamente com mais detalhes de sistemas vestíveis de detecção de quedas.

.

3

Sistemas Vestíveis

Sistemas Vestíveis podem ser definidos como dispositivos eletrônicos móveis que podem ser discretamente embutidos nos trajes do usuário, como parte da roupa ou um acessório. Diferente dos sistemas móveis convencionais, eles podem funcionar sem ou com muito pouca interferência nas atividades do usuário (Lukowicz and Kirstein, 2004).

Hoje, muitos destes dispositivos vestíveis vem com uma gama de sensores embutidos que são utilizados na detecção de quedas. O tipo de sensor mais comum utilizado em SDQ é o acelerômetro, com alguns desses sistemas também utilizando o giroscópio como um sensor auxiliar. De acordo com a revisão sistemática feito por Igual *et al.* (2013), 186 dos 197 sistemas analisados utilizam o acelerômetro como sensor principal na detecção de quedas. A utilização desses sensores em SDQ se deve muito pela popularização e o barateamento dos mesmos, além da utilização desses sensores embarcados em smartphones e smartwatches.

As seções desse capítulo são organizadas da seguinte maneira: A seção 3.1 descreve os principais tipos de sensores utilizados em SDQ; A 3.2 fala sobre o posicionamento de sensores, fazendo uma correlação entre o posiciomanento dos sensores e a accurâcia dos sistemas estudados; A seção 3.3 fala sobre os diferentes tipos de algoritmos de detecção de quedas mostrando suas vantagens e desvantagens; Por fim, a seção 3.4 irá mostrar exemplos de aplicações que realizam a detecção de quedas através de tecnologias vestíveis.

3.1 Sensores

A escolha e o bom funcionamento de sensores são uma parte essencial em sistemas de detecção de quedas. De forma geral, sensores são dispositivos que convertem fenômenos físicos em sinais elétricos. Sendo assim , eles representam a camada de comunicação

entre o mundo físico e o mundo digital.

Os dois tipos principais de sensores utilizados em sistemas de detecção de queda são o acelerômetro e o giroscópio. Eles fazem parte de um grupo chamado de Sensores Microeletromecânicos (MEMS). Estes sensores são geralmente feitos de chips de silício utilizando as mesmas técnicas usadas na confecção de chips de computadores pessoais. Para que um sensor possa ser classificado como MEMS alguma parte do seu design precisa vibrar ou se mover de alguma forma (Milette, 2012).

De forma geral, tanto o acelerômetro quanto o giroscópio utilizam três eixos para expressar seus valores. O sistema de coordenadas utilizado é relativo a cada dispositivo. Na Figura 3.1 temos um exemplo do sistema de coordenadas utilizado pelo sistema operacional Android[1]. Neste sistema o ponto de origem é o centro da tela do dispositivo, quando segurado na posição vertical (Google, 2016b).

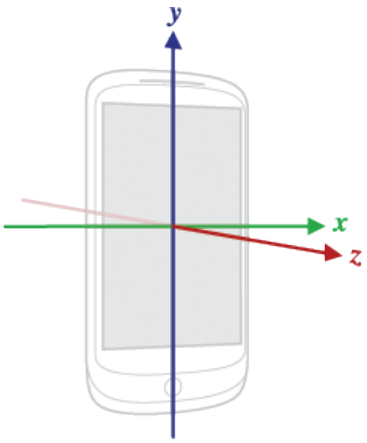

Figura 3.1 Sistema de coordenadas utilizado pelo sistema operacional Android (Google, 2016b).

3.1.1 Acelerômetro

Fisicamente, o acelerômetro é um dispositivo composto de uma pequena massa anexado a pequenas molas que são utilizadas para medir a aceleração aplicada sobre um dispositivo, incluindo a força da gravidade. A aceleração é medida analisando o quanto a massa se distancia do seu ponto de equilíbrio.

Na Figura 3.2 em A é possível ver um aparelho parado em uma mesa, sobre ele só irá agir a força de gravidade 1G de aproximadamente $9.8 m/s^2$. Em B, o aparelho foi jogado para a direita, então irá agir sobre ele, além da força da gravidade, uma aceleração no sentido para onde o aparelho se movimentou. Já em C, vemos um aparelho em queda

[1] https://www.android.com/

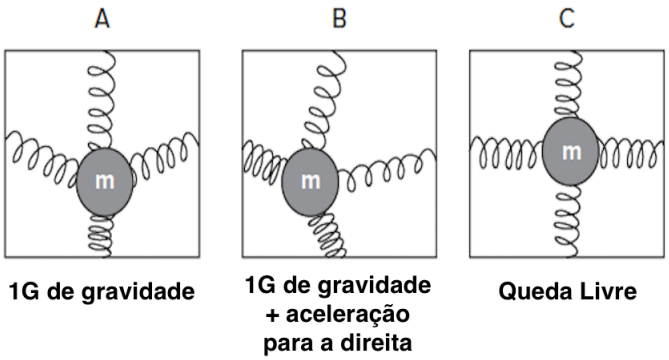

Figura 3.2 Força sendo aplicando sobre uma massa presa a molas (Milette, 2012).

livre com aceleração no sentido oposto a força da gravidade, o que faz com a massa fique localizada em seu ponto de equilibrio, e a força resultante seja de $0G$ (Milette, 2012).

3.1.2 Giroscópio

O giroscópio, similarmente aos acelerômetros, são pequenas massas em pequenas molas, só que em vez de medir a aceleração, são utilizados para medir um tipo de força chamada de Força de Coriolis. A Força de Coriolis é a tendência que um objeto livre possui de sair do curso quando visto de um ponto de referência em rotação (Milette, 2012). Por exemplo, se sentarmos em um carrossel e rolarmos uma bola pra longe, a bola irá parecer desviar em uma linha reta, como se existisse uma força agindo sobre ela. Esta força é chamada de Força de Coriolis.

Apesar de possuir uma estrutura física semelhante, o acelerómetro e o giroscópio se diferem em seu funcionamento. Em vez de esperar a força da gravidade agir sobre a massa, o giroscópio funciona vibrando esta massa sobre o eixo definido. Quando o giroscópio é rotacionado, a Força de Coriolis faz com que a massa comece a ser mover em um eixo diferente no qual ele estava vibrando anteriormente. Milette (2012).

Como a Força de Coriolis age somente quando o dispositivo está em rotação, o giroscópio só é capaz de calcular a velocidade angular, ou seja, a velocidade com que o aparelho está sendo rotacionado.

A orientação do dispositivo irá definir quando o valor da rotação será positivo ou negativo. Nos dispositivos Android, a velocidade Angular é medida em radianos por segundo (rad/s) e é positiva em rotações no sentido anti-horário (Google, 2016c).

3.2 Posicionamento de Sensores

O posicionamento dos sensores afeta diretamente a performance dos SDQ, dependendo da posição onde colocamos os sensores, o sistema pode indicar uma maior ou menor quantidade de falhas.

Não existe um consenso sobre a posição otimizada dos sensores para que se possa realizar a detecção de quedas. De acordo com Abbate *et al.* (2011), a cintura seria o local ideal para o posicionamento, já que estaria mais perto do centro de gravidade do corpo humano. Entretanto em Kangas *et al.* (2007), já foi sugerido que a cabeça seria o melhor lugar para posicionar os sensores. Outras soluções, como em Gjoreski *et al.* (2011), já propõem o uso de mais de um sensor, colocando-os em diferentes partes do corpo com o objetivo de aumentar ainda mais a precisão dos SDQ.

De acordo com Casilari (2015b), o pulso não é o local recomendado para o posicionamento de sensores em sistemas de detecção de quedas. Isso se deve a constante movimentação dos braços, que podem gerar um número grande de falso-positivos. Entretanto, alguns sistemas tem conseguido resultados satisfatórios com SDQ localizados no pulso. O sistema proposto por Hsieh *et al.* (2014), foi capaz detectar quedas em 151 das 160 quedas simuladas e obteve uma especificidade (capacidade de não reconhecer eventos de quedas, como tal) de 95%.

Além da performance do sistema, outras questões precisam ser levadas em consideração quando pensamos no posicionamento dos sensores. Uma delas é a danificação dos sensores na ocorrência de uma queda. Caso o sistema pare de funcionar, um possível alerta de emergência poderá não ser enviado e o idoso poderá estar correndo grande perigo.

Outra questão é a usabilidade, o uso de muitos sensores, apesar de poder elevar a precisão do sistema, poderá levar a um desconforto do usuário, que pode fazer até com que o mesmo desista de usá-lo.

3.3 Algoritmos de Detecção de Queda

Os algoritmos de detecção de quedas recebem como entrada os dados obtidos através dos sensores e são capazes de determinar se o que ocorreu foi um evento de queda ou somente uma AD. De acordo com Casilari (2015b), é possível separar os algoritmos de detecção em dois grandes grupos: Algoritmos de detecção através de métodos de reconhecimento de padrões e algoritmos baseados em limiares.

3.3.1 Reconhecimento de Padrões

O reconhecimento de padrões é uma área do aprendizado de máquina que foca no reconhecimento de padrões e regularidades de dados (Anzai, 2012). Diversas técnicas de aprendizado de máquina tem sido empregadas na detecção de quedas, de acordo com a revisão sistemática feita por Casilari (2015b), algoritmos como o de *Naïve Bayes*, *Redes Neurais*, e *Árvores de Decisão* tem sido utilizados.

Um exemplo de sistema que utiliza a técnica de reconhecimento de padrões foi proposto por Zhao *et al.* (2012). O seu algoritmo de detecção de quedas analisa dados do acelerômetro através de uma árvore de decisão, assim identificando um evento de queda.

Este algoritmo é composto de 2 fases, a primeira é o que chamamos em aprendizado de máquina de fase de treinamento. Será realizado a coleta de dado e a extração das características que são pertinentes, além do treinamento de um modelo de árvore de decisão. Na figura 3.3 podemos ver o modelo de árvore de decisão gerado. Este modelo de árvore de decisão é capaz de reconhecer atividades como andar, correr, estado estático ou um evento de queda através das variáveis *Std_x*, *Mean_y* e *Slope* que representam caracteristicas do sistema.

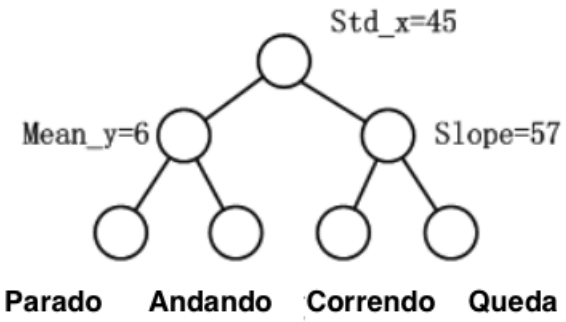

Figura 3.3 Modelo de árvore de decisão construida em Zhao *et al.* (2012).

Na segunda fase, chamada de fase de testes, o modelo de árvore de decisão é utilizado em uma aplicação no smartphone que será responsável pelo reconhecimento de atividades. Os dados dos sensores são catalogados e transformados nas características do sistema, que serviram como dados de entrada da árvore de decisão, que terá como saída o tipo de atividade que foi desempenhada.

Sistemas que utilizam reconhecimento de padrões, normalmente está suscetível a altos custos computacionais, análise massiva de dados, e acesso a grandes bancos de dados ou longos períodos de treinamentos onde o algoritmo de classificação precisa ser

parametrizado e adaptado a diferentes grupos de usuários (Casilari, 2015b). Em contraste, existem os algoritmos baseados em limiares que tendem a ser mais simples e similarmente eficientes, desde que encontremos limiares adequados.

3.3.2 Baseado em Limiares

Algoritmos baseados em limiares utilizados na detecção de quedas comparam os dados dos sensores com um ou mais valores pré-definidos, chamados de limiares. Estes valores podem ser fixos ou adaptados. Quando estes valores são adaptados, eles não mudam dinamicamente enquanto os usuários estão utilizando o sistema. Em vez disso, o usuário irá introduzir dados sobre o seu perfil fisiológico e o sistema irá informar os limiares adequados (Habib *et al.*, 2014). Um exemplo deste tipo de sistema pode ser visto em Sposaro (2009), o valor limiar mudar de acordo com os parâmetros providos pelo usuário como altura, peso e nível de atividade.

De acordo com a revisão sistemática feita por Casilari (2015b), muitos sistemas de detecção de queda utilizam o valor de Magnitude Vetorial (SMV) do vetor de aceleração como valor limiar principal em seus algoritmos de detecção. De acordo com Casilari (2015b), o valor de SMV é definido através da equação em 3.1, onde X_i, Y_i, Z_i, representam, respectivamente, os valores de aceleração dos eixos x, y, z obtidos através do acelerômetro.

$$SMV = \sqrt{X_i^2 + Y_i^2 + Zi_i^2} \tag{3.1}$$

A escolha dos limiares é um fator determinante para o sucesso deste tipo de algoritmo de detecção de quedas. A escolha dos limiares pode ser feita através de experimentos preliminares como em Zhang *et al.* (2013). Em seu trabalho, um grupo de voluntários foi escolhido para realizar diversas AD, como andar, correr, subir e descer escadas e também realizar a simulação de quedas. Através dos dados obtidos foi possível descobrir os limiares de SMV para um evento de queda.

De acordo com Cao *et al.* (2012), a performance dos algoritmos aumenta significativamente quando utilizamos valores de limiar dinâmicos. De acordo com sua pesquisa, o número de eventos de queda que não foram caracterizados como tal, caíram de 53 para 29 quando o peso, sexo idade foram levados em consideração no momento da definição dos limiares.

Outra questão importante quando utilizado este tipo de algoritmo, é o número de limiares utilizados. De acordo com Casilari (2015b), o uso de um único limiar faz com

que o algoritmo emita um número grande de alertas falsos, categorizando AD como eventos de queda, fazendo com que o uso de somente um limiar não seja adequado no desenvolvimento de SDQ.

3.4 Trabalhos Relacionados

Como vimos no decorrer desse trabalho, não existe na literatura um algoritmo ou dispositivo padrão para o desenvolvimento de SDQ. A plataforma vestível é bastante promissora por estar naturalmente aclopada a alguma parte do corpo do usuário, sendo possivel criar um algoritmo de detecção mais confiáveis, já que, como visto em Casilari (2015b), grande parte dos algoritmos de detecção de quedas tem como pré-condição para o seu bom funcionamento, que o dispositivo esteja localizado em uma posição fixa do corpo, como cintura, pulso ou cabeça. Sendo assim, falaremos sobre três sistemas de detecção de quedas que embarcaram os seus sistemas de detecção de quedas em plataformas vestíveis. Estes três sistemas foram escolhidos pela semelhança dos mesmos com a solução proposta neste trabalho.

3.4.1 SPEEDY - Detector de Quedas em um Relógio de Pulso

SPEEDY foi a primeiro protótipo de um relógio detector de quedas construído em um smartwatch (Degen *et al.*, 2003). Em seu trabalho, ele utilizou dois sensores que são capazes de medir a aceleração através de 3 eixos *x, y, z*. Na Figura 3.4 é possível ver o protótipo do sistema desenvolvido e os 3 eixos utilizados para calcular o valor da aceleração.

O algoritmo de detecção de quedas do Speedy utiliza um algoritmo baseado em limiares com 3 valores distintos: o valor de SMV calculado através da formula que pode ser vista em 3.1, dois valores de velocidade distintos chamados de v_1 e v_2. A velocidade v_1 é o valor aproximado da velocidade vertical (de queda), e o valor da velocidade v_2 representa a velocidade do dispositivo Speedy.

No primeiro passo do algoritmo, um alto valor de velocidade precisa ser identificado indicando uma possível queda. Depois disso, nos próximos 3 segundos um impacto precisa ser detectado, representado por alto valores de aceleração. Depois disso, o usuário é observado por mais 60 segundos, se durante este tempo, pelo menos 40 segundos forem marcados por inatividade um alerta sonoro é emitido.

O Speedy foi avaliado através de quedas simuladas por 3 indivíduos em um colchão. Cada indivíduo simulou quedas em 3 posições diferentes: Frente, lado e costas. Foram

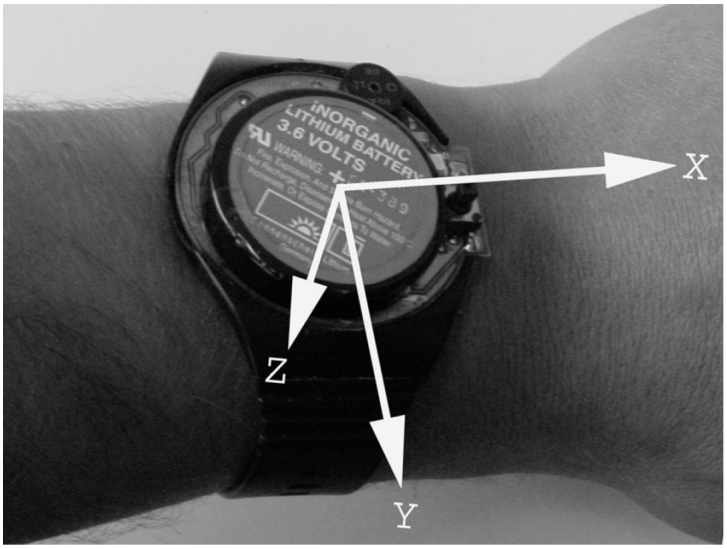

Figura 3.4 SPEEDY e seus eixos (Degen *et al.*, 2003).

realizadas um total de 45 quedas, onde 65% delas foram corretamente marcadas como um evento de queda.

O Speedy foi desenvolvido utilizando um dispositivo próprio, fazendo com que os usuários precisem adquirir um dispositivo vestível que irá realizar exclusivamente a detecção de quedas fazendo com que a sua adesão seja mais complexa do que em um sistema que já utilize smartwatches já consolidados no mercado.

3.4.2 F2D - Sistema de Detecção de Quedas

F2D é uma applicação Android embarcada em um smartwatch *AW-420.RX* da Simvalley Mobile[2] (Kostopoulos *et al.*, 2015). O algoritmo implementado no F2D tem como entrada os dados do acelerômetro, levando em consideração os movimentos realizados depois de um evento de queda e a localização do usuário.

Para que se possa detectar as quedas, o F2D utiliza um algoritmo baseado em limiares, onde os limiares foram definidos utilizando um banco de dados com mais de 150 eventos simulados de queda. O algoritmo de detecção de quedas presente no F2D é composto de quatro etapas (Kostopoulos *et al.*, 2015):

1. **Padrão de Queda**: Para que um evento possa ser identificado como uma possível queda, o valor da aceleração precisa ultrapassar um limite que varia de $10m/s^2$ a $18m/s^2$ que representa o impacto da queda, e depois de um intervalo de tempo,

[2]http://www.simvalley-mobile.de/

precisa ultrapassar um limiar de $2m/s^2$ a $7m/s^2$, caracterizado como o movimento residual da queda. Tanto os valores exatos dos limiares, quanto o intervalo de tempo entre a análise dos mesmos variam de acordo com o perfil do usuário.

2. **Módulo de Decisão**: Toda vez que ambas as condições da etapa anterior são satisfeitas é acrescido 1 em um contador. São estabelecidos dois valores X e Y. O valor de X representa o limiar do contador e Y representa o seu limite. Sendo assim, o valor do contador precisa ficar entre X e Y, ou seja, $X \leq Contador < Y$. Caso este valor seja maior que Y, outra atividade estava sendo desempenhada, como por exemplo correr, e caso este valor do contador seja menor que X, onde $X = 1$, então um movimento brusco do braço aconteceu, mas que não caracteriza uma queda.

3. **Ação posterior à Queda**: Logo depois que um evento é caracterizado como queda, o F2D é capaz de identificar se o usuário conseguiu se recuperar, e voltou a exercer suas atividades normais, caso isto ocorra, o sistema não irá emitir um alerta para o seu cuidador, caso contrário, um alerta é emitido

4. **Ação baseada na Localização**: O F2D também se baseia na localização do usuário. Caso este esteja na rua, e todas as etapas anteriores se concretizaram, um alarme é enviado para o cuidador. Caso o usuário esteja em casa, o sistema faz uso da tecnologia *iBeacon* para categorizar certos locais como seguros ou potencialmente perigosos. O iBeacon utiliza o sistema de detecção de proximidade do Bluetooth Low Energy (BLE) para enviar um identificador único para aplicações ou sistemas operacionais compatíveis que estejam ao alcance do mesmo (Kostopoulos *et al.*, 2015). Caso o local esteja marcado como potencialmente perigoso, uma mensagem é enviada ao cuidador, caso contrário, o usuário terá a oportunidade de cancelar o envio, em um possível evento de queda.

O *F2D*, além de realizar a detecção de quedas, utiliza dados do contexto para definir se o evento ocorrido foi potencialmente perigoso ou não, o que o diferencia da maioria das soluções presentes na literatura. Entretanto, para que ele possa mapear essas áreas é necessário a montagem de toda uma infraestrutura com a utilização de iBeacons, o que torna a sua implementação e adesão mais complexa.

3.4.3 Sistema de Detecção de Quedas de Pulso

O sistema proposto por Hsieh *et al.* (2014) utiliza dois dispositivos vestíveis acoplados no pulso do usuário como pode ser visto na Figura 3.5.

Figura 3.5 Dispositivos vestíveis circulados. (Hsieh *et al.*, 2014).

Cada um dos dispositivos está equipado com um módulo Zigbee[3] responsável pela transmissão dos dados e um acelerômetro e giroscópio de 3 eixos. A frequência tanto do acelerômetro quanto do giroscópio foram configuradas para $50Hz$, ou seja, os dados são coletados a cada *20 ms*.

O algoritmo proposto utiliza ambos os dados do acelerômetro e do giroscópio para realizar a detecção de quedas. Os dados do giróscopio funcionam como um filtro inicial, desconsiderando a maioria das atividades cotidianas, enquanto os dados do acelerômetro são responsáveis por realizar o julgamento final. O algoritmo utilizado, assim como os demais, é baseado em limiares onde os limiares foram definidos atravês de um treinamento inicial.

O algoritmo proposto é capaz de diferenciar AD como bater palmas e deitar-se, de um evento de queda em 95% dos casos. A principal desvantagem do sistema proposto é a necessidade de se utilizar 2 dispositivos, podendo torna-se desconfortável para o usuário.

O sistema de detecção de pulso, assim como o *Speedy*, utiliza de um dispositivo próprio para realizar a detecção, o que faz com sua adesão seja mais complexa. Além disso, para realizar a detecção o sistema necessita que o usuário utilize 2 relógios, o que pode ser incômodo para o usuário final. Por fim, ele necessita dos dados do giroscópio além do acelerômetro, o que pode exigir um maior custo computacional e consequentemente um maior consumo de bateria.

[3] http://www.zigbee.org/

4

SafeWatch: Sistema de Detecção de Quedas

Neste trabalho, apresentamos o SafeWatch, uma solução integrada entre smartphone e smartwatch, onde quedas são detectadas de maneira automatizada, e se necessário, os contatos de emergência do idoso são informados de sua localização para que possa prestar socorro de forma mais rápida possível.

As seções desse capítulo são organizadas da seguinte maneira: A Seção 4.1 mostra a arquitetura que foi definida e utilizada pela ferramenta construída; A Seção 4.2 mostra as ferramentas que foram utilizadas para auxiliar à construção do SafeWatch; A Seção 4.3 descreve detalhes da implementação do SafeWatch; Por fim, a seção 4.4 ilustra a execução da ferramenta.

4.1 Arquitetura

De acordo com Garlan (1993), a arquitetura de um software define o sistema em termos de componentes e as interações existentes entre esses componentes. Em outras palavras, a arquitetura de software tem o objetivo de mostrar uma visão completa do sistema.

O SafeWatch foi desenvolvido para funcionar como um aplicativo Android Wear[1] para smartwatches que funciona em conjunto com o smartphone Android do usuário, através de uma aplicação homônima, que está sincronizado com o mesmo. A ferramenta foi desenvolvida com base em uma arquitetura pré-definida e possui os seus módulos desacoplados para facilitar futuras mudanças ou melhorias. A ferramenta está dividida em 5 módulos como ilustra a Figura 4.1.

[1] https://www.android.com/wear/

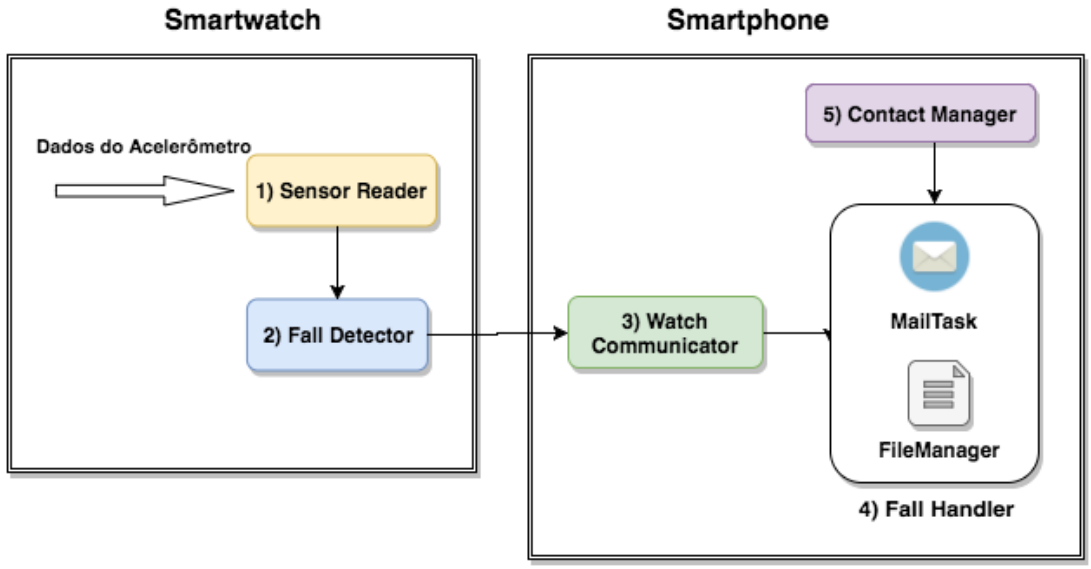

Figura 4.1 Arquitetura do SafeWatch. Figura Elaborada pelo autor (2016).

1. **Sensor Reader**: Responsável pela configuração e gerenciamento dos sensores, mais especificamente do sensor utilizado, o acelerômetro.

2. **Fall Detector**: Recebe informações provindas do *Sensor Reader* para através do algoritmo de detecção de quedas categorizar um determinado evento como queda ou não.

3. **Watch Communicator**: Realiza a comunicação entre o smartwatch e o smartphone do usuário. Irá receber dados do smartwatch que são tratados pela módulo chamado de *Fall Handler*.

4. **Fall Handler**: Será responsável pelas ações do smartphone após um evento de queda, como o envio de emails e o gerenciamento dos dados do acelerômetro recebidos do smartwatch.

5. **Contact Manager**: Será responsável pelo gerenciamento dos contatos de emergência do usuário. Ações como visualização, adição e remoção de contatos estão encapsuladas neste módulo.

Os módulos *Sensor Reader* e *Fall Detector* estão presentes na aplicação embarcada no smartwatch, os demais módulos estão presentes na aplicação para smartphones.

4.2 Ferramentas Utilizadas

Durante o desenvolvimento do *SafeWatch* foram utilizadas diversas ferramentas que serviram para dar suporte a sua implementação e execução. Tanto o aplicativo para smartphones, quanto o aplicativo embarcado no smartwatch foram desenvolvidos utilizando o Android Studio[2]. O Android Studio é a Ambiente de Desenvolvimento Integrado (IDE) oficial do Google no desenvolvimento de aplicações móveis ou vestíveis.

Nas classes do projeto relacionadas às telas do aplicativo foi utilizado o ButterKnife[3]. O Butterknife é responsável por fazer a ligação entre os arquivos responsáveis pela criação das telas, e as classes que utilizam os componentes visuais destas telas.

Para realizar os cálculos do desvio padrão foi utilizada a biblioteca do Apache chamada Commons-Math[4]. Esta biblioteca contém um conjunto de funções matemáticas e de estatística não presentes na biblioteca padrão do JAVA.

Para que possamos enviar os dados do acelerômetro do smartwatch para o smartphone, eles precisam estar codificados em algum padrão, o padrão escolhido foi o JSON. O JSON é um formato de dados utilizado para comunicação entre dispositivos (JSON, 2016). Para que possamos fazer a codificação e decodificação dos dados do acelerômetro foi utilizada a biblioteca chamada Gson[5].

Por fim, para o envio de emails para os contatos de emergência foi utilizada a biblioteca padrão criada pela Oracle[6] chamada de JavaMail[7].

4.3 Implementação

A implementação do SafeWatch foi dividida em várias partes, onde cada uma delas é representada por um módulo independente dos demais. A linguagem de programação utilizada foi Java[8], linguagem padrão no desenvolvimento de aplicações Android. Nas seções a seguir são detalhados detalhes da implementação e funcionamento de cada módulo.

[2]https://developer.android.com/studio/
[3]texthttps://github.com/JakeWharton/butterknife
[4]http://commons.apache.org/proper/commons-math/
[5]http://www.json.org/
[6]http://www.oracle.com/
[7]http://www.oracle.com/technetwork/java/javamail/index.html
[8]http://www.oracle.com/technetwork/java/index.html

4.3.1 Sensor Reader

O módulo *Sensor Reader* é responsável pela configuração e gerenciamento do acelerômetro. Aqui, o acelerômetro é configurado para atualizar seus dados a uma frequência de 50 Hz, ou seja, a cada 20 ms. Os dados do acelerômetro são coletados a todo momento, mesmo quando a aplicação não está em primeiro plano.

Este módulo também será responsável por armazenar os dados do acelerômetro nos últimos 0.4 segundos para posterior uso no algoritmo de detecção, caso necessário. Tanto a escolha da frequência de 50 Hz quanto o tempo de 0.4 segundos para o armazenamento de dados do acelerômetro serão explicados com mais detalhes na Seção 4.3.2.

4.3.2 Fall Detector

O módulo *Fall Detector* encapsula o algoritmo de detecção de quedas baseado em limiares utilizado pelo SafeWatch. Este é o módulo mais complexo da aplicação, pois nele se encontra a lógica responsável por decidir, através dos dados obtidos do acelerômetro, se um evento de queda ocorreu ou não. O algoritmo proposto é uma adaptação do algoritmo desenvolvido por Hsieh *et al.* (2014). O algoritmo proposto neste trabalho se diferencia do algoritmo proposto em Hsieh *et al.* (2014) pela não utilização do giroscópio como sensor auxiliar, acredita-se que sem o uso do giroscópio é possível obter-se resultados satisfatórios como visto na Seção 3.4.2.

Um grande desafio quando utilizamos um algoritmo baseado em limiares é a definição dos valores dos limiares. Caso este valor seja muito alto, o sistema irá deixar escapar alguns eventos de queda, mas não irá categorizar uma SDQ como uma queda. Do outro lado, se este valor for muito baixo, o sistema irá detectar todos os eventos de queda, mas algumas SDQ pode ser categorizadas como eventos de queda de maneira equivocada. De acordo com o treinamento inicial realizado por Hsieh *et al.* (2014), o valor de SMV, representado pela fórmula 3.1, será maior que $6G$, onde $G \approx 9.8m/s^2$, no momento do impacto em um evento de queda.

Também foi identificado por Hsieh *et al.* (2014), que caso o valor de aceleração atinja o valor de $6G$, o valor do desvio padrão ficava com valores em torno de $1.07G$ em movimentos regulares do braço realizados 0.4 segundos antes ou depois deste pico de aceleração.Entretanto, em eventos de queda, este valor estava mais próximo de $1.69G$.

Por fim, o período de inatividade posterior a uma queda foi analisado. De acordo com Hsieh *et al.* (2014), o valor da Soma das Acelerações (SMA), expresso através da Equação 4.1, tem uma relação diretamente proporcional com o nível de movimentação de um

corpo. Foi identificado que em eventos de queda, o indíviduo tende a ficar parado por pelo menos 2 segundos, com valores de SMA inferiores a $200G$. É importante ressaltar que este valor de $200G$ é encontrado quando a frequência do acelerômetro é de 50 Hz.Caso contrário, o número de amostras coletados será diferente, afetando diretamente no valor de SMA.

$$SMA = \sum_{i=1}^{N} (|\, X_i \,| + |\, Y_i \,| + |\, Zi_i \,|) \tag{4.1}$$

Nesta equação X_i, Y_i, Z_i, são os valores da aceleração no tempo i e N é o número de amostras desejadas. Levando como base o treinamento inicial descrito acima foi possível desenvolver o algoritmo descrito na imagem 4.2.

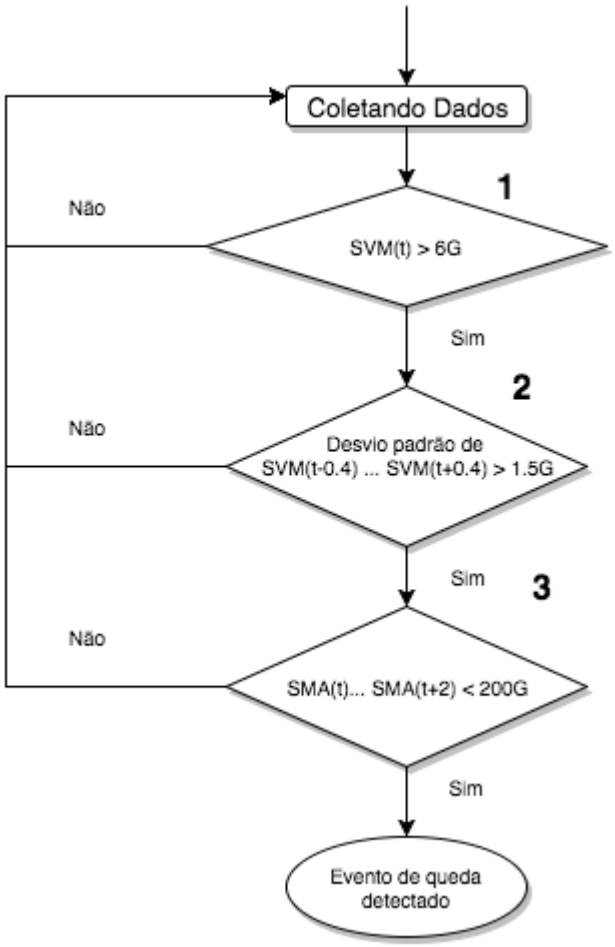

Figura 4.2 Fluxograma do algoritmo proposto. Figura Elaborada pelo autor (2016).

1. Os valores do acelerômetro são monitorados, caso SMV seja maior do que $6G$, os demais valores de SMV são monitorados por mais 0.4 segundos. O maior dos

valores observado neste tempo é marcado como pico de aceleração e o algoritmo prossegue para o passo 2.

2. O desvio padrão de *SMV* é calculado, nos 0.4 segundos anteriores e posteriores a detecção do maior valor de SMV. Caso este valor não seja menor do que $1.5G$, o algoritmo irá para o passo 3.

3. O valor de *SMA* é calculado, e caso este valor seja menor do que $200G$ finalmente confirmamos que um evento de queda ocorreu.

Caso um evento de queda seja detectado, o relógio irá vibrar por 15 segundos, na espera de um feedback do usuário informando se ele está bem ou não. O tempo de 15 segundos foi escolhido, pois o usuário terá tempo de dar um feedback para o sistema caso ele não esteja em uma situação de perigo. Da mesma forma, o tempo de 15 segundos não irá atrasar o envio da mensagem de emergência quando o usuário realmente precisar.

O código fonte em 4.1 , representa o algoritmo utilizado para realizar a detecção de quedas implementado utilizado a linguagem de programação *JAVA*.

```
package com.victortavares.safewatch.Models;

import android.content.Context;
import android.os.Handler;
import android.util.Log;

import org.apache.commons.math3.stat.descriptive.
    DescriptiveStatistics;

import java.util.ArrayList;
import java.util.List;

/**
* Created by Victor Tavares on 7/14/16.
*/
public class FallDetector implements SensorCommunicator {

private AccelerometerReader accelerometerReader;
private static final String TAG = "FALL_DETECTOR";
private Boolean detectingFall = false;
```

```
//gravity Constants
private static float gravity = 9.81f;
private static float gravity6G = gravity *6f;
private static float gravity1AndHalfG = gravity * 1.5f;
private static float gravity200G = gravity *200f;


private static FallDetectorCommunicator fallObserver;


public FallDetector(FallDetectorCommunicator fallObserver)
    {
accelerometerReader = new AccelerometerReader(this);
this.fallObserver = fallObserver;
}

public void startMonitoring(Context  c) {
accelerometerReader.startMonitoring(c);
}


public void accelerometerAboveLimit(SensorBase
   accelerometerMeasurement) {

if (detectingFall) return;

Log.i(TAG, "Possible␣Fall.␣accelerometerData:␣" +
   accelerometerMeasurement.getSpeed()/gravity + "␣at␣" +
   accelerometerMeasurement.getDate() + "␣on␣" +
   accelerometerMeasurement.getTimestamp());
analyseData(accelerometerMeasurement.getTimestamp());
}
```

```
private void analyseData(final long timeOfOcurrence) {
detectingFall = true;
accelerometerReader.setTriggered(true);


Handler handler = new Handler();
handler.postDelayed(new Runnable() {

@Override
public void run() {
SensorBase maxSpeedGatheringAfter4Seconds = new SensorBase
   (0f,0f,0f, (long) 0);
DescriptiveStatistics statsCalculation = new
   DescriptiveStatistics();
Float SMAAfter2Seconds = 0f;


//checking 4 seconds after the accelerometer indicator
for (SensorBase currentSensor : accelerometerReader.
   getlTrigerredAccelerometerData()) {
float timeBetweenFallAndDetection = (currentSensor.
   getTimestamp() - timeOfOcurrence);
//Log.i(TAG, "Collecting max speed:" + currentSensor.
   getSpeed() + " at " + currentSensor.getDate() + " on "
   + currentSensor.getTimestamp() );
//getting the max value of occurrence 4 seconds after the
   fall
if (timeBetweenFallAndDetection <= 400 && currentSensor.
   getSpeed() > maxSpeedGatheringAfter4Seconds.getSpeed())
    {
maxSpeedGatheringAfter4Seconds = currentSensor;
}
}
```

```
//Log.i(TAG, "----------------------------\n");

//getting 4 seconds before and after the max value of
    speed.
List<SensorBase> combinedValues = accelerometerReader.
    getlDefaultAccelerometerData();
combinedValues.addAll(accelerometerReader.
    getlTrigerredAccelerometerData());
//Log.i(TAG, "value of pick:" +
    maxSpeedGatheringAfter4Seconds.getSpeed()/gravity + "
    at " + maxSpeedGatheringAfter4Seconds.getDate() + " on
    " + maxSpeedGatheringAfter4Seconds.getTimestamp());
for (SensorBase currentSensor : combinedValues) {

float timeBetweenPickAndCurrentSensor = currentSensor.
    getTimestamp() - maxSpeedGatheringAfter4Seconds.
    getTimestamp();

if (timeBetweenPickAndCurrentSensor >= -400 &&
    timeBetweenPickAndCurrentSensor <= 400) {
//Log.i(TAG, "Adding value to stats calculation:" +
    currentSensor.getSpeed() + " at " + currentSensor.
    getDate() + " on " + currentSensor.getTimestamp() );
statsCalculation.addValue(currentSensor.getSpeed());
}
}


Log.i(TAG, "----------------------------\n");

//getting 2 seconds after the max value of speed, ignoring
    the max value
for (SensorBase currentSensor : accelerometerReader.
    getlTrigerredAccelerometerData()) {
float timeBetweenPickAndCurrentSensor = currentSensor.
```

```
    getTimestamp() - maxSpeedGatheringAfter4Seconds.
    getTimestamp();

if (timeBetweenPickAndCurrentSensor <= 2000 &&
    timeBetweenPickAndCurrentSensor > 0) {
//Log.i(TAG, "Adding value to SMA calculation:" +
    currentSensor.getSpeed() + " at " + currentSensor.
    getDate() + " on " + currentSensor.getTimestamp() );
SMAAfter2Seconds += currentSensor.getSmaSum();
}
}



//Log.i(TAG, "------------------------------\n");

//Detection algorithm
//checking 0.4 seconds after the gathering is bigger than
    6G
if (maxSpeedGatheringAfter4Seconds.getSpeed() > gravity6G)
     {
Log.i(TAG, "Fall␣detected␣on␣maxValue:" + (
    maxSpeedGatheringAfter4Seconds.getSpeed()/gravity) + "G
    ");
} else {
Log.i(TAG, "Not␣Fall(maxValue4Seconds):" + (
    maxSpeedGatheringAfter4Seconds.getSpeed()/gravity) + "G
    ");
finishDataAnalysis();
return;
}



//standard deviation 0.4 seconds before and after the
    maxSpeedGatheringAfter4Seconds is higher than 1.5G
Double standardDeviation = statsCalculation.
```

```
    getStandardDeviation();
if (standardDeviation > gravity1AndHalfG) {
Log.i(TAG, "Fall␣detected␣on␣standard␣deviation:" + (
    standardDeviation/gravity) + "G");
} else {
Log.i(TAG, "Not␣Fall␣(standardDeviation):␣" + (
    standardDeviation/gravity) + "G");
finishDataAnalysis();
return;
}

//SMA value above 2 seconds is below 200G
if (SMAAfter2Seconds < gravity200G) {
Log.i(TAG, "Fall␣detected␣on␣SMA:" + (SMAAfter2Seconds/
    gravity) + "G");
} else {
Log.i(TAG, "Not␣Fall␣detected␣on␣SMA:" + (SMAAfter2Seconds
    /gravity) + "G");
finishDataAnalysis();
return;

}

fallObserver.fallDetected(combinedValues);
finishDataAnalysis();
}



}, 2400);

}

private void finishDataAnalysis() {
accelerometerReader.setTriggered(false);
detectingFall = false;
```

```
}

}
```

Código Fonte 4.1 Algoritmo de Detecção de Quedas em Java

4.3.3 Watch Communicator

O módulo *Watch Communicator* é responsável pela comunicação entre o smartwatch e o smartphone do usuário. Fisicamente, esta comunicação é realizada via bluetooth, já a nível de software, está comunicação é realizada através do que chamamos na arquitetura Android de *Services*.

No Android, um *Service* é um componente da aplicação capaz de realizar operações de longa duração em segundo plano Google (2016a). No SafeWatch, os *Services* recebem os dados do acelerômetro referentes ao evento de queda, ou seja, todos os registros do acelerômetro 0.4 segundos antes do pico de aceleração, até 2 segundos depois deste valor. Além disso, também é enviado, uma variável booleana indicando se deve-se ou não enviar um e-mail para a lista de contatos de emergência do usuário. Os emails de emergência são enviados para a lista de contato do usuário 15 segundos após um evento de queda, ou antes disso, caso o usuário confirme que precisa de ajuda.

Para que os contatos da lista de emergência não sejam incomodados desnecessariamente na ocorrência de falsas detecções de quedas, o usuário poderá cancelar o envio dos emails de emergência, dentro de 15 segundos após um evento de queda, caso ele informe que está bem.

4.3.4 Fall Handler

Este módulo é responsável pelo envio de e-mails e a manipulação de arquivos com os dados de uma queda. Para que se possa realizar o envio de e-mails, foi criado um email padrão do SafeWatch. O envio de e-mail é feito de forma assíncrona, sem bloquear a interação do usuário com a aplicação.

Os dados referentes a um evento de queda são salvos na raiz do sistema de arquivos do smartphone android na pasta */SafeWatch/smartwatch*. O arquivo é nomeado com o padrão *experimentData_timeStamp*, onde t*timeStamp* representa o momento do salvamento do arquivo, em milisegundos. O arquivo está salvo no formato *CSV*, com os valores de aceleração nos eixos x,y,z, o valor de SMV correspondente ao registro e o tempo em

milisegundos em que ele ocorreu. Apesar destes valores não serem úteis para o usuário final, eles poderão servir para uma possível replica dos experimentos realizados.

4.3.5 Contact Manager

Neste módulo, estão encapsuladas as ações de adição, remoção e listagem dos contatos de emergência do usuário. Além do e-mail, nas informações do contato também constará o nome completo do mesmo.

4.4 Telas e Funcionamento

O sistema foi desenvolvido de forma que o usuário necessite interagir o mínimo possível com os aplicativos tanto no smartphone quanto no smartwatch.

De forma geral, a aplicação smartwatch irá monitorar as atividades do usuário através do acelerômetro e no momento em que uma queda for detectada emitirá um alerta vibratório juntamente com um sinal para o smartphone. No smartphone está presente uma aplicação de gerenciamento geral do sistema, nesta aplicação o usuário será capaz de adicionar, visualizar e remover os contatos de emergência que seriam notificados no momento de uma queda.

Inicialmente, o usuário deve realizar o cadastro dos contatos de emergência, a tela de cadastro pode ser vista na figura 4.4. O usuário necessita informar o nome completo e email do contato desejado. Todos os contatos de emergência do usuário podem ser visualizados em forma de lista como pode ser visto na Figura 4.5.

Depois de adicionar os contatos de emergência, o usuário não necessita realizar mais nenhum tipo de cadastro ou configuração no sistema. Na ocorrência de uma queda o sistema irá se comportar como pode ser visto na Figura 4.3.

Enquanto um evento de queda não é detectado, o smartwatch irá realizar o monitoramento constante dos dados do acelerômetro, durante esta fase, o sistema irá mostrar uma mensagem, indicando que está realizando o monitoramento, como pode ser visto na figura 4.7.

Na Figura 4.6, é possível ver o momento que um evento de queda é detectado. Neste momento, o smartwatch irá emitir um sinal de vibração, além de uma mensagem perguntando se está tudo bem com o usuário. Esta mensagem ficará visível por 15 segundos, caso o usuário não cancele o envio, uma mensagem é enviada para o smartphone informando que uma queda ocorreu.

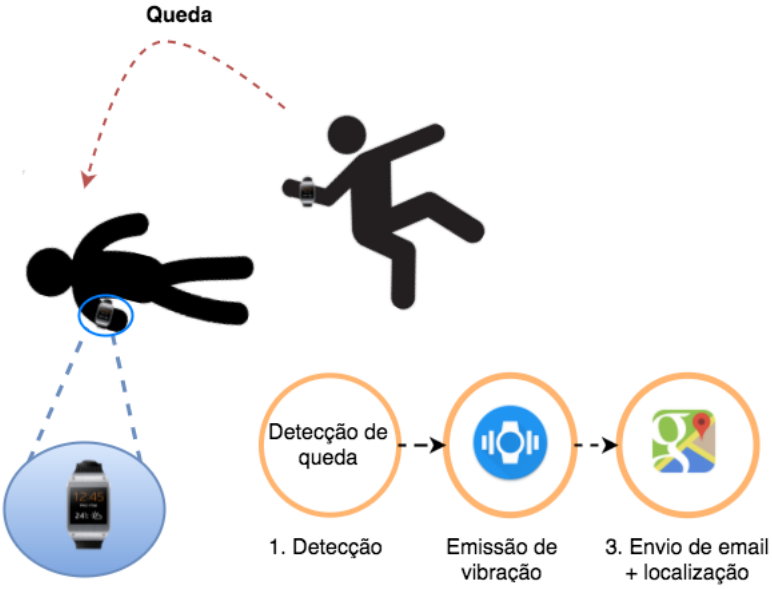

Figura 4.3 Aplicação no evento de queda. Figura Elaborada pelo autor (2016).

O smartphone irá enviar um email para o usuário. O modelo do email pode ser visto na figura 4.8. Além de uma mensagem informando que o usuário pode está em uma situação de perigo, um link do *Google Maps*[9] é anexado com a última localização do usuário obtida pelo sistema.

[9]https://www.google.com/maps

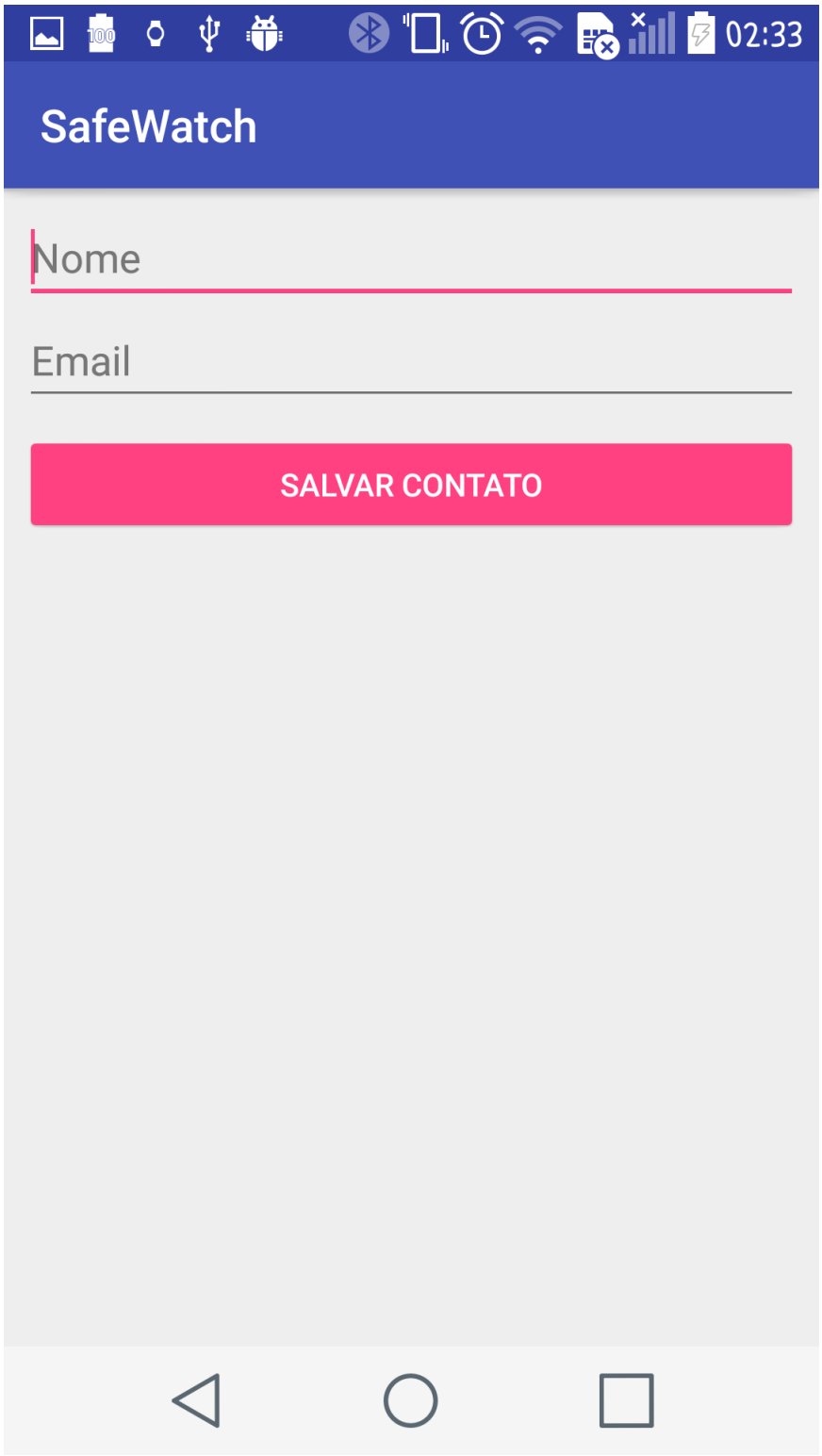

Figura 4.4 Tela de adição de um contato de emergência. Figura Elaborada pelo autor (2016).

Figura 4.5 Tela de visualização dos contatos de emergência. Figura Elaborada pelo autor (2016).

Figura 4.6 Tela apresentada quando uma queda é detectada. Figura Elaborada pelo autor (2016).

Figura 4.7 Tela de monitoramento presente no smartwatch. Figura Elaborada pelo autor (2016).

Figura 4.8 Modelo de email enviado no evento de queda. Figura Elaborada pelo autor (2016).

5

Estudo Experimental

Neste capítulo será apresentado o processo de avaliação utilizado para verificar a precisão do sistema de detecção de quedas proposto. Espera-se que o SafeWatch apresente uma precisão similar aos demais sistemas de detecção de quedas presentes na literatura.

As seções desse capítulo são organizadas da seguinte maneira: A Seção 5.1 apresenta os detalhes da metodologia utilizada para avaliar o SafeWatch; A Seção 5.2 mostra as métricas utilizadas na avaliação; A Seção 5.3 apresenta os resultados obtidos no experimento realizado e faz uma comparação com os resultados obtidos em outros trabalhos.

5.1 Metodologia

Para avaliarmos o desempenho do SafeWatch foi realizado uma série de experimentos. O algoritmo de detecção de quedas proposto foi avaliado através de um conjunto de quedas simuladas e também um conjunto de atividades diárias realizadas pelos participantes do experimento. O grupo de voluntários possui um perfil bem diversificado, sendo composto de 3 homens e 5 mulheres como pode ser visto na tabela 5.1.

O smartwatch escolhido para realizar o experimento foi um *Moto 360* da 1º geração com as seguintes características (Motorola, 2016):

1. Sistema Operacional: Android Wear 2.0.
2. CPU: Qualcomm Snapdragon 400, 1.2GHz.
3. Memória RAM: 512 MB.
4. Capacidade de Armazenamento: 4GB.

Tabela 5.1 Participantes do experimento

Individuo	Idade	Sexo	Peso	Altura
Individuo 1	28	Masculino	82kg	1.80m
Individuo 2	20	Feminino	63kg	1.63m
Individuo 3	25	Feminino	62kg	1.59m
Individuo 4	29	Masculino	130kg	1.65m
Individuo 5	27	Feminino	57kg	1.69m
Individuo 6	14	Feminino	45kg	1.62m
Individuo 7	20	Masculino	88kg	1.80m
Individuo 8	30	Feminino	62kg	1.55m

Já o smartphone escolhido foi um *LG G2* com as seguintes características (Arena, 2016):

1. Sistema Operacional: Android Lollipop 5.0.2.
2. CPU: Quad-core 2.26 GHz Krait 400.
3. Memória RAM: 2 GB.
4. Capacidade de Armazenamento: 16GB.

Para avaliar o algoritmo de detecção de quedas implementado, foi realizado o seguinte experimento, composto de três etapas:

- **Preparação**: Nesta etapa é solicitado que o usuário coloque o smartwatch em seu pulso e ajuste a pulseira do relógio de uma maneira que o smartwatch permaneça firme, mas confortável. Depois disso, é informado que o usuário deverá simular duas quedas em cada um dos sentidos escolhidos: Costas, Frontal, Lado Direito, Lado Esquerdo. A ordem das quedas é decidida pelo usuário, o único requisito é que ele realize todas as oito quedas.
- **Realização das Quedas**: O usuário irá se posicionar de pé, na frente de um colchão coberto de almofadas como pode ser visto na Figura 5.1. A partir desta posição, ele irá realizar as oitos quedas, duas de cada tipo, como descrito na etapa anterior.
- **Realização de Atividades Diárias**: Nesta etapa solicitamos que o usuário realize 4 atividades do seu cotidiano. Elas são: sentar em uma cadeira, levantar de uma cadeira, deitar no colchão e levantar do colchão. Estas atividades são realizadas afim de verificar se o sistema proposto é capaz de distinguir atividades diárias de um evento de queda.

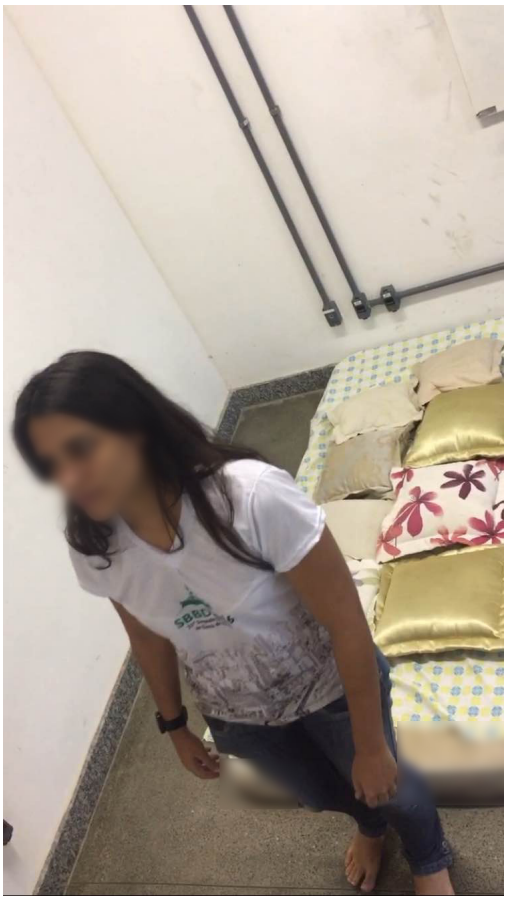

Figura 5.1 Usuário em preparação para uma queda de costas. Figura Elaborada pelo autor (2016).

5.2 Métricas de Avaliação

Para que possamos analisar a performance do sistema de detecção de quedas foram utilizadas três métricas, a *Sensibilidade*, *Especificidade* e *Acurácia*. De acordo com Casilari (2015a), os valores de *Sensibilidade* e *Especificidade* são duas métricas bastante utilizadas na literatura para a análise de performance em sistemas de detecção de quedas. Elas representam, respectivamente, a proporção de eventos de queda e AD que foram classificadas corretamente como tal. Já a *Acurácia* é uma combinação da *Sensibilidade* e da *Especificidade* e nos dá uma ideia geral da performance do sistema.

A *Sensibilidade* é expressa pela fórmula 5.1. As variáveis *TP* e *FN* são, respectivamente, acrónimos para True Positive (Verdadeiro Positivo em inglês) e False Negative (Falso Negativo em inglês). A variável TP representa o número de eventos de queda corretamente classificadas, enquanto FN representa as quedas que não foram detectadas pelo sistema.

$$Sensibilidade = \frac{TP}{TP+FN} \tag{5.1}$$

Já a *Especificidade* é expressa pela fórmula 5.2. As variáveis *TN* e *FP* são, respectivamente acrónimos para True Negative (Verdadeiro Negativo em inglês) e False Positive (Falso Positivo em inglês). A variável TN representa o número de atividades diárias corretamente classificadas como tal, enquanto FP representa as atividades diárias que foram classificadas como queda.

$$Especificidade = \frac{TN}{FP+TN} \tag{5.2}$$

Para calcularmos a acurácia geral do sistema, devemos utilizar a fórmula 5.3.

$$Acuracia = \frac{TP+TN}{TP+FP+TN+FN} \tag{5.3}$$

5.3 Resultados

Na tabela 5.2 podemos ver os resultados dos experimentos de queda para cada um dos indivíduos. Como descrito na Seção 5.1 cada individuo realizou duas quedas em quatro sentidos diferente. Cada elemento da tabela representa o número de quedas que foram identificadas com sucesso pelo sistema em cada uma das direções.

Tabela 5.2 Resultados do experimentos de Queda.

Individuo	**Frente**	**Costas**	**Direita**	**Esquerda**
Individuo 1	2	1	2	2
Individuo 2	1	2	2	2
Individuo 3	2	2	1	2
Individuo 4	2	2	2	2
Individuo 5	2	1	2	1
Individuo 6	2	2	2	1
Individuo 7	2	2	2	2
Individuo 8	2	1	2	2

Como podemos ver na tabela 5.2, o número de verdadeiros positivos é de 57, enquanto o número de falsos negativos é de somente 7, o que nos dá uma *especificidade* de $89,06\%$. Foi possível observar que, o limiar inicial de $6G$ no algoritmo de detecção de quedas não foi alcançado em todos os eventos erroneamente não categorizados como queda.

Já na tabela 5.3, podemos ver os resultados do experimentos de AD para cada um dos indivíduos. Cada um deles realizou quatro tipos de AD, como descrito na Seção 5.1.

Cada elemento da tabela representa o número de AD que não foram identificadas pelo sistema como um evento de queda.

Tabela 5.3 Resultados do experimentos de AD.

Individuo	Levantar(Cadeira)	Sentar(Cadeira)	Levantar(Cama)	Deitar(Cama)
Individuo 1	2	2	2	2
Individuo 2	2	2	2	2
Individuo 3	2	2	2	2
Individuo 4	2	2	2	2
Individuo 5	2	2	2	2
Individuo 6	2	2	2	2
Individuo 7	2	2	2	2
Individuo 8	2	2	2	2

Neste experimento, nenhum dos 64 eventos AD foi identificado como um evento de queda pelo sistema, levando a uma *especificidade* de 100%. De forma geral o sistema possui uma *acurácia* de $94,17\%$.

O algoritmo proposto neste trabalho apresentou resultados satisfatórios identificando um evento de queda em quase 90% dos casos, e não apresentando nenhum falso positivo nas AD testadas. Em comparação com o Speedy, sistema desenvolvido por Degen *et al.* (2003), o nosso sistema apresentou uma sensibilidade 24.06% maior, ou seja, o SafeWatch apresentou uma especificidade de $89,06\%$ contra 65% do Speedy em experimentos bastantes similares.

Entretanto em comparação com Hsieh *et al.* (2014), este trabalho apresentou uma *sensibilidade* 5.94% menor e uma *especificidade* $3,3\%$ maior. Os experimentos realizados por Hsieh *et al.* (2014) foram em uma superfície acolchoada mais fina que um colchão convencional, o que leva a uma aceleração maior no impacto, diminuindo o número de falsos negativos no experimento realizado. Em relação ao tipos de quedas realizados, tanto este, quanto o trabalho proposto por Hsieh *et al.* (2014) apresentaram quatro tipos de quedas: frontais, laterais (Direito e Esquerdo), Costas. Já em relação as AD, ele apresentou um número maior de atividades, incluindo andar e correr.

6

Conclusão

A proposta deste trabalho foi criar um sistema de detecção de quedas que exige-se o mínimo de iteração possível do usuário, consumindo o mínimo de recursos possíveis com uma precisão similar aos demais SDQ existentes através de uma plataforma vestível que esteja se popularizando no mercado.

As maiores dificuldades encontradas foram no desenvolvimento do algoritmo de detecção de quedas. Com a proposta inicial de somente o acelerômetro, diferente de demais sistemas que também utilizam o giroscópio a acurácia do sistema poderia ser afetada, caso algum dos limiares não se adaptassem a essa nova proposta. Outra dificuldade foi entender as características de uma queda a partir de sua aceleração. Alguns conceitos físicos não são tão triviais, o que levou a muita pesquisa.

Por fim, foi desenvolvido o SafeWatch, um sistema de detecção embarcado em um smartwatch que utiliza o smartphone como uma plataforma auxiliar. A interface é simples, e permite que o usuário interaja com o sistema de maneira fácil e somente quando necessário. O SafeWatch foi analisado em questão de performance e apresentou resultados satisfatórios com uma acurácia similar ou melhor que outros SDQ existentes.

Como trabalhos futuros, existem as seguintes possibilidades:

1. Buscar meios de otimizar o consumo de bateria do smartwatch mesmo com o constante monitoramento dos dados dos sensores.

2. Integração com outros tipos de sensores, como o sensor de batimento cardíaco, afim de aumentar ainda mais a precisão do sistema.

3. Reconhecimento de outros tipos de atividades além da queda (e.g detector de possível AVC), visando expandir o sistema de um simples sistema de detector de quedas, para um sistema completo de monitoramento.

4. Realizar o teste da aplicação com um número maior de AD, principalmente aquelas que exigem uma movimentação maior do braço do individuo, como andar ou correr.

Referências Bibliográficas

Abbate, S., Avvenuti, M., Cola, G., Corsini, P., Light, J., and Vecchio, A. (2011). Recognition of false alarms in fall detection systems. In *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 23–28. IEEE.

Alwan, M., Rajendran, P. J., Kell, S., Mack, D., Dalal, S., Wolfe, M., and Felder, R. (2006). A smart and passive floor-vibration based fall detector for elderly. In *2006 2nd International Conference on Information & Communication Technologies*, volume 1, pages 1003–1007. IEEE.

Anzai, Y. (2012). *Pattern Recognition & Machine Learning*. Elsevier.

Arena, G. (2016). Lg g2. http://www.gsmarena.com/lg_g2-5543.php. Accessado em 16 de Outubro de 2016.

Cao, Y., Yang, Y., and Liu, W. (2012). E-falld: A fall detection system using android-based smartphone. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 1509–1513. IEEE.

Casilari, Eduardo e Oviedo-Jiménez, M. A. (2015a). Automatic fall detection system based on the combined use of a smartphone and a smartwatch. *PLoS one*, **10**(11), e0140929.

Casilari, Eduardo e Luque, R. e. M. M.-J. (2015b). Analysis of android device-based solutions for fall detection. *Sensors*, **15**(8), 17827–17894.

Cucchiara, R., Grana, C., Prati, A., and Vezzani, R. (2005). Probabilistic posture classification for human-behavior analysis. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, **35**(1), 42–54.

Degen, T., Jaeckel, H., and Rufer, Michael e Wyss, S. (2003). Speedy: A fall detector in a wrist watch. In *ISWC*, pages 184–189.

Foroughi, H., Naseri, A., Saberi, A., and Yazdi, H. S. (2008). An eigenspace-based approach for human fall detection using integrated time motion image and neural network. In *2008 9th International Conference on Signal Processing*, pages 1499–1503. IEEE.

Garlan, David e Shaw, M. (1993). An introduction to software architecture. *Advances in software engineering and knowledge engineering*, **1**(3.4).

Gartner (2015). Gartner says smartphone sales surpassed one billion units in 2014. http://www.gartner.com/newsroom/id/2996817/. Accessado em 22 de Abril de 2016.

Gibson, M. J., Andres, R. O., Isaacs, B., Radebaugh, T., and Wormpetersen, J. (1987). The prevention of falls in later life-a report of the kellogg-international-work-group on the prevention of falls by the elderly. *Danish Medical Bulletin*, **34**, 1–24.

Gjoreski, H., Lustrek, M., and Gams, M. (2011). Accelerometer placement for posture recognition and fall detection. In *Intelligent environments (IE), 2011 7th international conference on*, pages 47–54. IEEE.

Google (2016a). Android services. https://developer.android.com/guide/components/services.html. Accessado em 24 de Setembro de 2016.

Google (2016b). Sensor coordinate system. https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-intro. Accessado em 10 de Setembro de 2016.

Google (2016c). Using the gyroscope. http://developer.android.com/guide/topics/sensors/sensors_motion.html#sensors-motion-gyro. Accessado em 08 de Maio de 2016.

Habib, M. A., Mohktar, M. S., Kamaruzzaman, S. B., Lim, K. S., Pin, T. M., and Ibrahim, F. (2014). Smartphone-based solutions for fall detection and prevention: challenges and open issues. *Sensors*, **14**(4), 7181–7208.

Hsieh, S.-L., Chen, C.-C., Wu, S.-H., and Yue, T.-W. (2014). A wrist-worn fall detection system using accelerometers and gyroscopes. In *Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on*, pages 518–523. IEEE.

IBGE (2012). Número de idosos que moram sozinhos triplica em 20 anos. http://www.ibge.gov.br/home/estatistica/populacao/trabalhoerendimento/pnad2012/default_sintese.shtm. Accessado em 21 de Abril de 2016.

IBGE (2016). Projeção da população do brasil e das unidades da federação. http://www.ibge.gov.br/apps/populacao/projecao/. Accessado em 21 de Abril de 2016.

Igual, R., Medrano, C., and Plaza, I. (2013). Challenges, issues and trends in fall detection systems. *Biomedical engineering online*, **12**(1), 1.

JSON (2016). Json. http://www.json.org/. Accessado em 12 de Outubro de 2016.

Kangas, M., Konttila, A., Winblad, I., and Jamsa, T. (2007). Determination of simple thresholds for accelerometry-based parameters for fall detection. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1367–1370. IEEE.

Kostopoulos, P., Nunes, T., Salvi, K., Deriaz, M., and Torrent, J. (2015). F2d: A fall detection system tested with real data from daily life of elderly people. In *2015 17th International Conference on E-health Networking, Application & Services (HealthCom)*, pages 397–403. IEEE.

Legters, K. (2002). Fear of falling. *Physical therapy*, **82**(3), 264–272.

Lukowicz, P. and Kirstein, Tnde Troster, G. (2004). Wearable systems for health care applications. *Methods of Information in Medicine-Methodik der Information in der Medizin*, **43**(3), 232–238.

Mehner, S., Klauck, R., and Koenig, H. (2013). Location-independent fall detection with smartphone. In *Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments*, page 11. ACM.

Milette, Greg e Stroud, A. (2012). *Professional Android sensor programming*. John Wiley & Sons.

Motorola (2016). Moto 360. https://www.motorola.com/us/products/moto-360. Accessado em 16 de Outubro de 2016.

Mubashir, M., Shao, L., and Seed, L. (2013). A survey on fall detection: Principles and approaches. *Neurocomputing*, **100**, 144–152.

Portal Brasil (2012). Quedas. http://www.brasil.gov.br/saude/2012/04/quedas/. Accessado em 21 de Abril de 2016.

Rougier, C., Meunier, J., St-Arnaud, A., and Rousseau, J. (2011). Robust video surveillance for fall detection based on human shape deformation. *IEEE Transactions on Circuits and Systems for Video Technology*, **21**(5), 611–622.

Rougier, Caroline e Meunier, J. (2005). Demo: Fall detection using 3d head trajectory extracted from a single camera video sequence. *Journal of Telemedicine and Telecare*, **11**(4), 37–42.

Samsung (2016). Gear s2. http://www.samsung.com/global/galaxy/gear-s2/#!/spec/. Accessado em 22 de Abril de 2016.

Sposaro, Frank, T. G. (2009). ifall: an android application for fall monitoring and response. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6119–6122. IEEE.

Stephen R. Lord, C. S. e. H. B. M. (2001). *Falls in older people: risk factors and strategies for prevention*. Cambridge University Press, Cambridge.

United Nations (2013). World population ageing 2013. Report, The Department of Economic and Social Affairs of the United Nations.

Wild, D., Nayak, U., and Isaacs, B. (1981). How dangerous are falls in old people at home? *Br Med J (Clin Res Ed)*, **282**(6260), 266–268.

Yi He, Ye Li, S.-D. B. (2012). Fall detection by built-in tri-accelerometer of smartphone. *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*, **25**, 184–187.

Zhang, Q., Ren, L., and Shi, W. (2013). Honey: A multimodality fall detection and telecare system. *Telemedicine and e-Health*, **19**(5), 415–429.

Zhao, Z., Chen, Y., Wang, S., and Chen, Z. (2012). Fallalarm: Smart phone based fall detecting and positioning system. *Procedia Computer Science*, **10**, 617–624.

Zhuang, X., Huang, J., Potamianos, G., and Hasegawa-Johnson, M. (2009). Acoustic fall detection using gaussian mixture models and gmm supervectors. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 69–72. IEEE.