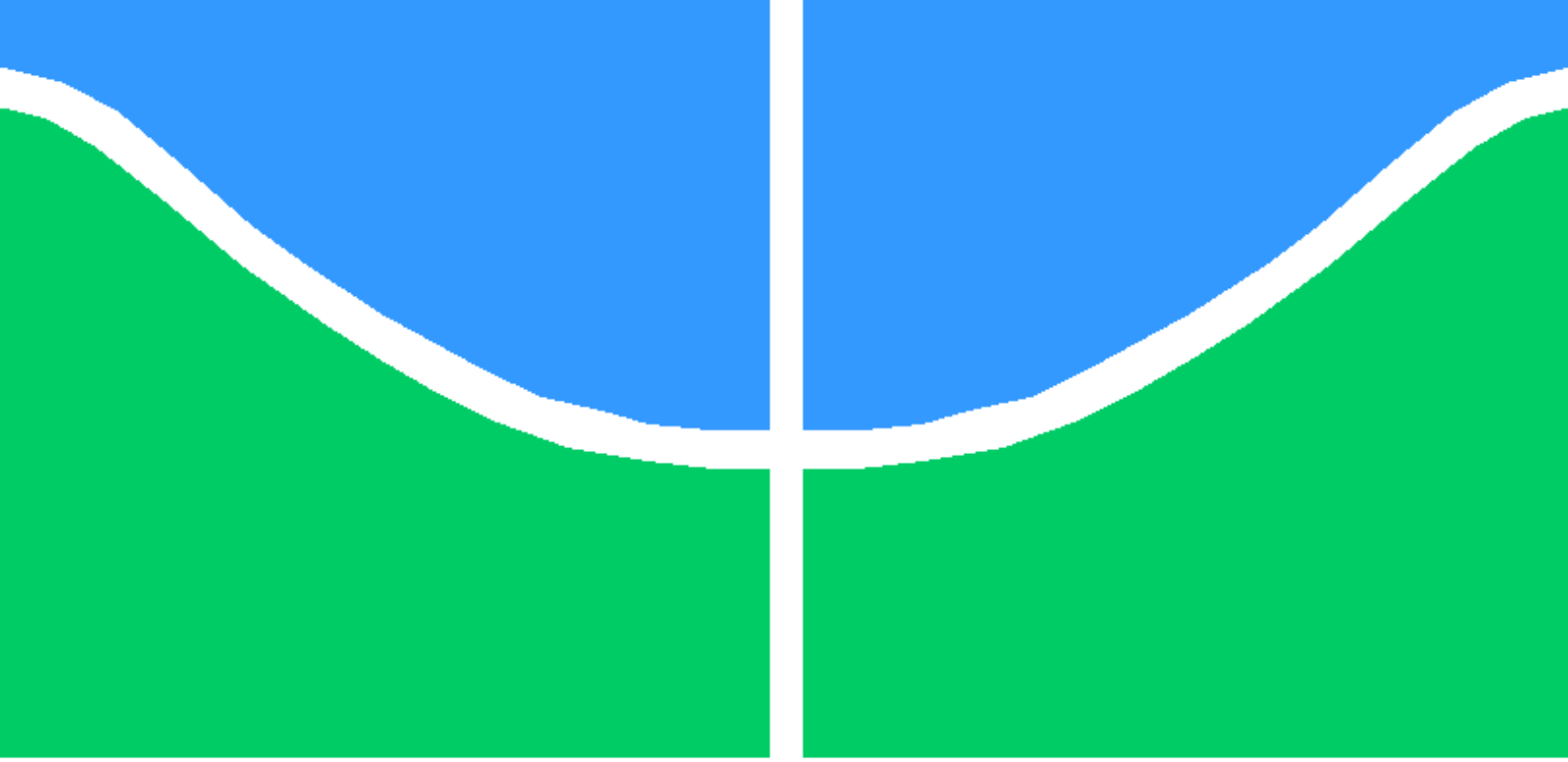


UMA ARQUITETURA DE IOT PARA DETECÇÃO DE QUEDAS DE PESSOAS EM AMBIENTES DE VIDA ASSISTIDA

FELIPE BARRETO DE OLIVEIRA

JOÃO VÍTOR MENDES LOURES

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA DE REDES DE COMUNICAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UMA ARQUITETURA DE IOT PARA DETECÇÃO DE QUEDAS DE PESSOAS EM AMBIENTES DE VIDA ASSISTIDA

FELIPE BARRETO DE OLIVEIRA

JOÃO VÍTOR MENDES LOURES

Orientador: PROF. DR. MARCELO MENEZES DE CARVALHO, ENE/UNB

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA DE REDES DE COMUNICAÇÃO

BRASÍLIA-DF, 21 DE MAIO DE 2021.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UMA ARQUITETURA DE IOT PARA DETECÇÃO DE QUEDAS DE PESSOAS EM AMBIENTES DE VIDA ASSISTIDA

FELIPE BARRETO DE OLIVEIRA

JOÃO VÍTOR MENDES LOURES

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO ACADÊMICO 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 BACHAREL EM ENGENHARIA DE REDES DE COMUNICAÇÃO.

APROVADA POR:

Prof. Dr. Marcelo Menezes de Carvalho, ENE/UnB
Orientador

Prof. Dr. João Luiz Azevedo de Carvalho, ENE/UnB
Examinador interno

Prof. Dr. Marcos Fagundes Caetano, CIC/UnB
Examinador externo

BRASÍLIA, 21 DE MAIO DE 2021.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me concedido forças para conseguir chegar até aqui. A Ele toda honra, glória e poder para todo sempre. Como está escrito "Os que confiam no Senhor serão como o monte de Sião, que não se abala, mas permanece para sempre" (Salmos 125:1), posso afirmar que essa palavra se cumpriu na minha trajetória nesses cinco anos de universidade. Agradeço também aos meus pais que sempre me apoiaram em todos os momentos, sendo estes de dificuldade ou de alegria, bem como a minha irmã que sempre me fez acreditar que isso seria possível. Devo citar aqui meus agradecimentos aos docentes dessa universidade, pois com o empenho e dedicação exercidos tornam esta uma referência nacional. Em especial agradeço ao professor Marcelo Carvalho que nos apoiou neste projeto. Por último agradeço a minha dupla, companheiro e amigo João Vitor que se fez presente em todos os desafios destes anos de caminhada juntos.

Felipe Barreto de Oliveira

Agradeço primeiramente à minha família, que fez sempre de tudo para eu poder chegar onde cheguei. Em segundo lugar, agradeço ao professor Marcelo Carvalho, que nos guiou durante todo o projeto, sempre orientando para o sucesso da dupla. Seus direcionamentos possibilitaram que o trabalho alcançasse o nível esperado. Por último, agradeço ao meu companheiro Felipe, que aceitou de prontidão a realização do trabalho em conjunto, além de sempre contribuir ativamente para o projeto.

João Vítor Mendes Loures

RESUMO

O aumento da expectativa de vida média e queda das taxas de natalidade estão provocando o envelhecimento das populações de países desenvolvidos e emergentes. A partir disso, os prognósticos apontam para a sobrecarga de cuidadores de idosos, que cada vez mais devem se preocupar com um número maior de pessoas da terceira idade. Para conseguir atender a essa demanda crescente, uma opção é utilizar tecnologias para facilitar o monitoramento. Um exemplo de uma dessas tecnologias é a Internet das Coisas (IoT, do inglês Internet of Things) que pode empregar o uso de objetos conectados à Internet para gerar um ambiente de vida assistida (AAL, do inglês Ambient-Assisted Living), onde idosos podem manter sua independência enquanto são acompanhados de forma remota. Um problema bem recorrente desta faixa etária são as quedas, que são responsáveis por cerca de 1 a cada 4 mortes de idosos nos Estados Unidos, por exemplo. Este trabalho propõe um sistema de detecção de quedas, de forma intrusiva, utilizando IoT, desde a captação dos dados à notificação aos familiares sobre a queda ocorrida. A coleta de dados é feita por um acelerômetro preso ao corpo do sujeito monitorado, que envia as informações a um *gateway* local. Este dispositivo faz a conexão com a Internet, e repassa os dados para o servidor central, que se encontra na nuvem. O terceiro componente faz o processamento dos sinais em dois passos: primeiro há a filtragem dos dados, seguida pela tomada de decisão do evento de queda ou não-queda através de limiares pré-estabelecidos. Caso o processamento indique ocorrência de queda, uma notificação é entregue via aplicativo de mensagens instantâneas. Os resultados apresentados ao longo do documento demonstram ser possível a utilização do sistema em situações cotidianas do público-alvo, além de se mostrar eficiente em relação ao tempo de resposta ao acidente.

Palavras-chave: Detecção de quedas, IoT, AAL.

ABSTRACT

The increase in average life expectancy and the fall in birth rates are causing the aging of populations in developed and emerging countries. Based on this, the prognosis points to the overload of elderly's caregivers, who are increasingly concerned with a greater number of elders. In order to meet this growing demand, one option is to use technologies to facilitate monitoring. An example of one of these technologies is the Internet of Things (IoT), which can employ the use of things connected to the Internet to generate an Ambient-Assisted Living (AAL), where the elderly can maintain their independence while being monitored remotely. A very recurring problem in this age group are falls, which are responsible for about 1 in 4 deaths of elderly people in the United States, for example. This work proposes an intrusive fall detection system, using IoT, from data collection to notification to family members about the fall that occurred. Data collection is done by an accelerometer attached to the monitored subject's body, which sends the information to a local gateway. This device makes the connection to the Internet, and passes the data to the central server, which is in the cloud. The third component does the processing of the signals in two steps: first there is the filtering of the data, followed by the decision of falling or not falling through pre-established thresholds. If processing indicates a drop, a notification via the instant messaging application is delivered. The results presented throughout the document demonstrate that it is possible to use the system in everyday situations of the target audience, in addition to being efficient in relation to the response time to the accident.

Keywords: Fall detection, IoT, AAL.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS

Acrônimos

IoT	*Internet of Things*
NCOA	*National Council on Aging*
AAL	*Ambient Assisted Living*
M2M	*Machine-to-Machine*
RFID	*Radio-Frequency Identification*
QoS	*Quality of Service*
DWT	*Discrete Wavelet Transform*
FIR	*Finite Impulse Response*
LPF	*Low-Pass Filter*
HPF	*High-Pass Filter*
HTTP	*Hypertext Transfer Protocol*
SPI	*Serial Peripheral Interface*
MEMS	*Micro Electro Mechanical Systems*
SMTP	*Simple Mail Transfer Protocol*
IMAP	*Internet Message Access Protocol*
RTOS	*Real Time Operating Systems*
RAM	*Random Access Memory*
CPU	*Central Processing Unit*
VM	*Virtual Machine*
6LoWPAN	*IPv6 over low power wireless personal area networks*
IETF	*Internet Engineering Task Force*
RPL	*Routing protocol for low-power and lossy network*
AWS	*Amazon Web Services*
SO	Sistema Operacional

Acrônimos

VPC	*Virtual Private Cloud*
RS232	*Recommended Standard 232*
Wi-Fi	*Wireless Fidelity*
GB	*Gigabyte*
EC2	*Elastic Compute Cloud*
API	*Application Programming Interface*
HTTPS	*Hypertext Transfer Protocol Secure*
XDK	*Cross Domain Development Kit*
VPS	*Virtual Private Server*
PIR	*Passive Infrared Sensor*
ECG	Eletrocardiograma
DoS	*Denial of Service*
MitM	*Man-in-the-Middle*
JSON	*JavaScript Object Notation*
NTP	*Network Time Protocol*
INFOrM	*INdoor Fall detectiOn Method*
SVM	*Support Vector Machine*
NB	*Naive Bayes*
ADL	*Activities of Daily Living*
CSV	*Comma-separated Values*
TCP	*Transmission Control Protocol*
cA	Coeficiente Aproximado
cD	Coeficiente Detalhado
CIDR	*Classless Inter-Domain Routing*
SSH	*Secure Shell*
SSD	*Solid State Drive*
ICMP	*Internet Control Message Protocol*
BLE	*Bluetooth Low Energy*
LoRaWAN	*Long Range Wide Area Network*
AMQP	*Advanced Message Queuing Protocol*
PCA	*Principal Component Analysis*

Capítulo 1

Introdução

1.1 Contextualização

Há algumas décadas, observamos uma tendência de envelhecimento da população devido ao aumento da expectativa de vida média dos habitantes, com mais notoriedade para os países de desenvolvimento econômico elevado, como Japão e Suíça. Conforme apresenta a Figura 1.1, as expectativas de vida no Japão, Suíça, Alemanha, Austrália e Brasil cresceram consistentemente no período de 2000 a 2018. Pode-se observar que os países mais desenvolvidos possuem a expectativa entre 78 e 84 anos durante todo o período analisado, indicando que a estatística está praticamente estabilizada nestes países, enquanto que no Brasil a expectativa de vida média teve um grande aumento no mesmo período, mudando de 70 a 76 em 18 anos.

Esta é uma tendência que só irá se consolidar ainda mais com o tempo. Segundo Noroozian [2], enquanto a população mundial de 1950 até 2050 crescerá por um fator de 3,7, a população de idosos, que possuem 60 anos ou mais, irá ampliar em cerca de 10 vezes neste período de 100 anos. Além disto, os mais idosos, que estão nos 80 anos ou mais, irão se multiplicar ainda mais, chegando a alcançar a incrível marca de 26 vezes a população de 1950 em apenas um século.

Em direção oposta à expectativa de vida, nota-se na Figura 1.2 a queda da taxa de fertilidade, que representa o número de filhos por mulher em todos os países citados anteriormente: Austrália, Brasil, Japão, Alemanha e Suíça. Durante o período explorado, entre 1960 e 2018, novamente o Brasil teve a maior variação de ponta-a-ponta, e seguiu de 6,0 a 1,8 filhos por mulher, enquanto a flutuação do número nos países mais avançados foi mais sutil, indicando que a baixa taxa de fecundidade já é um rumo seguido há bastante tempo por tais países.

Com o aumento evidente das populações mais idosas, pode-se perceber que o cenário atual e futuro demonstram que a população estará concentrada nas faixas etárias mais avançadas. Em contrapartida, com a queda indubitável da taxa de fertilidade por mulher, a quantidade de cuidadores de idosos irá diminuir, fazendo com que cada cuidador fique cada vez mais sobrecarregado, e

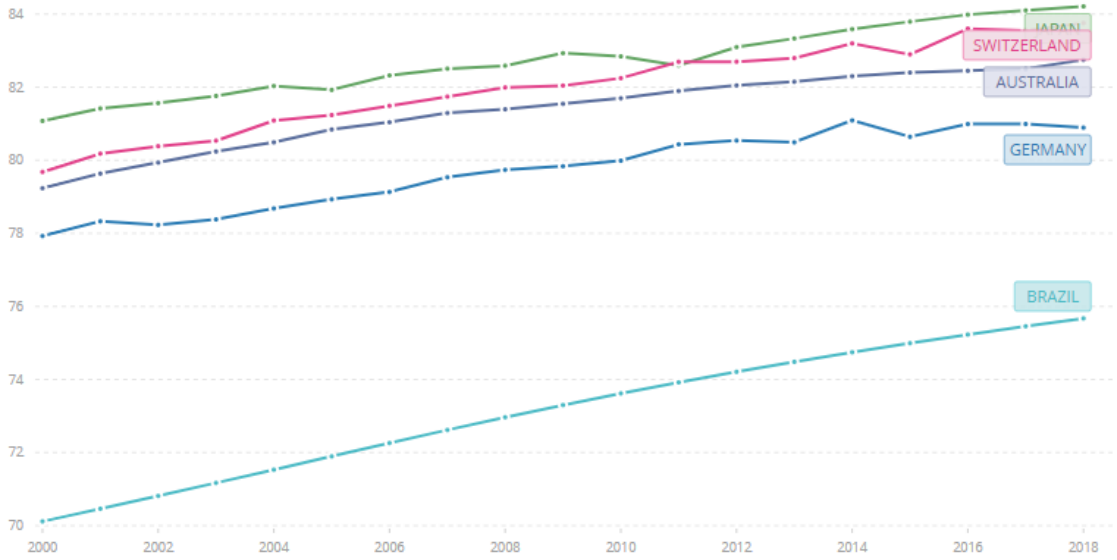


Figura 1.1: Dados sobre expectativa de vida dos anos 2000 a 2018 (The World Bank [1]). O eixo das ordenadas indica a idade correspondente ao ano, apresentado no eixo das abscissas.

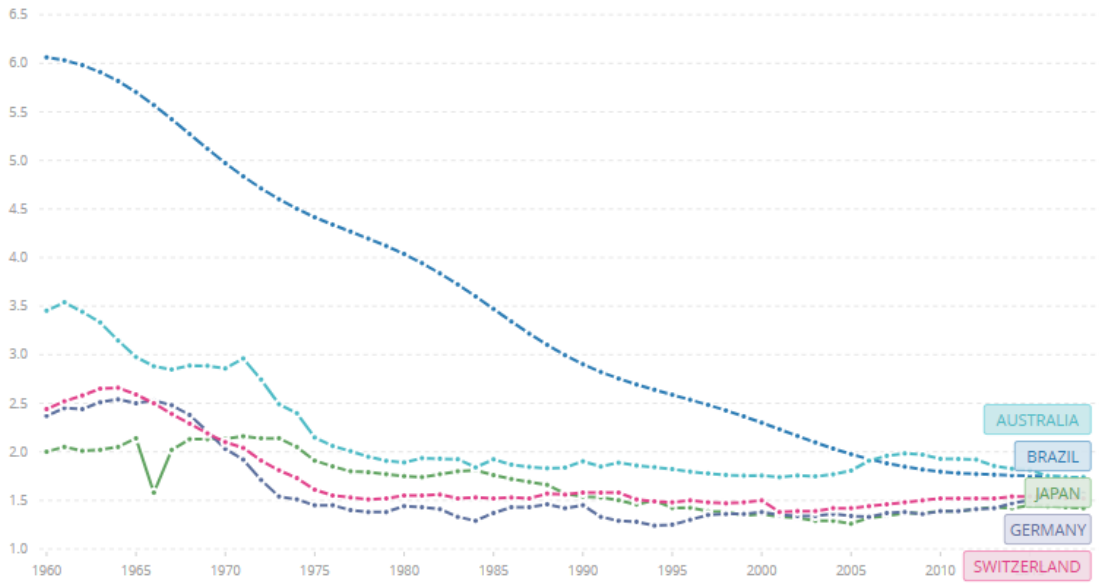


Figura 1.2: Dados sobre a taxa de fertilidade dos anos 1960 a 2018 (The World Bank [3]).O eixo das ordenadas indica a taxa de fertilidade correspondente ao ano, apresentado no eixo das abscissas.

tendo que ser responsável pelo zelo de várias pessoas em simultâneo. A cada ano, milhões de pessoas idosas (acima de 65 anos) sofrem algum tipo de queda. Conforme a organização americana NCOA (*National Council on Aging*) [4], só nos Estados Unidos, cerca de 1 a cada 4 idosos sofrem quedas por ano, além do que, as pessoas que sofreram uma queda pela primeira vez, adquirem probabilisticamente, o dobro da chance de ocorrência deste evento novamente. O monitoramento de quedas se torna complexo quando há uma quantidade alta de idosos para poucos cuidadores tomarem conta, visto que tais acidentes são recorrentes e imprevisíveis. O esforço deve ser focado em métodos e sistemas aplicados no monitoramento em massa de idosos. Como mostra a Figura 1.3, no período de 2007 a 2016 houve um acréscimo de cerca de 30% no número de mortes de pessoas idosas provocadas por quedas distintas somente nos Estados Unidos. Há também previsões demostrando que, se esse aumento for mantido, haverá cerca de 7 mortes por hora em 2030.

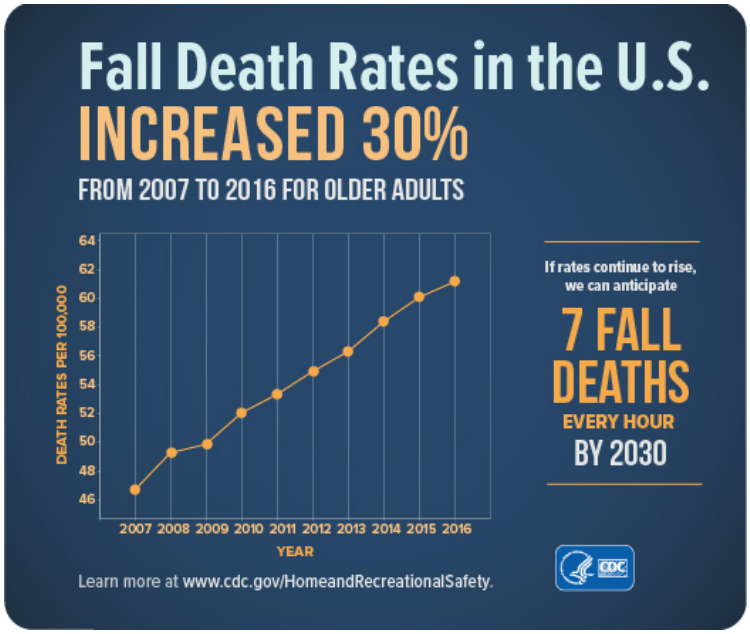


Figura 1.3: Taxas de mortes de idosos por quedas (*Center for Disease Control and Prevention* [5]).

Dada esta realidade que estamos presenciando, diversas formas de auxiliar idosos em seus afazeres têm surgido, com a utilização de tecnologias presentes. O avanço na área tecnológica possibilitou a proliferação de dispositivos com poder computacional e capacidade de coleta de dados através de sensores. Além disso, a comunicação sem fio e portabilidade facilitam o uso de tais equipamentos no dia-a-dia. Tais dispositivos podem auxiliar na coleta de uma imensa quantia de informações de diversos cenários nos quais estão inseridos. Neste intuito podemos citar o paradigma conhecido como Internet das Coisas (IoT do inglês *Internet of Things*), no qual atualmente se mostra um dos grandes assuntos explorados pelas indústrias e pesquisas, gerando vários trabalhos em diversos segmentos, desde áreas veiculares a áreas médicas.

Segundo Oracle [6], IoT pode ser definido como "fundamento que descreve a rede de objetos físicos —"coisas"— que são incorporados a sensores, *software* e outras tecnologias com o objetivo de conectar e trocar dados com outros dispositivos e sistemas pela *internet*. Esses dispositivos variam de objetos domésticos comuns a ferramentas industriais sofisticadas. Com mais de 7 bilhões de dispositivos IoT conectados atualmente, os especialistas esperam que esse número aumente para 10 bilhões até 2020 e 22 bilhões até 2025". Outros autores definem IoT como "uma rede de infraestrutura global dinâmica, na qual dispõe de capacidades de autoconfiguração, baseado em conexões físicas e virtuais de "coisas", que possuem identidades, atributos físicos e interfaces inteligentes". (Sundmaeker et al. [7])

O surgimento da IoT, com o uso da própria Internet como infraestrutura de comunicação de objetos traz diversos benefícios, possibilitando o monitoramento constante de sistemas que anteriormente necessitavam de intervenção humana. A automação do processo de acompanhamento reduz a possibilidade do erro humano, que é bem mais recorrente em monitoramentos manuais. Além disso, o projeto de redes de dispositivos IoT tem como foco a diminuição do tempo de resposta em situações de comunicação intermitente com o sistema monitorado. Por fim, esta tecnologia possibilita o uso da nuvem para processamento dos dados, na qual reduz a necessidade de adquirir equipamentos físicos sofisticados, bem como seus custos envolvidos.

Nos dias de hoje, há uma grande utilidade de IoT na área médica, a qual se pode destacar um conceito chamado Ambiente de Vida Assistida (AAL) (do inglês *Ambient-Assisted Living*) que pode ser definido como "um campo multidisciplinar emergente com o objetivo de fornecer um ecossistema de diferentes tipos de sensores, computadores, dispositivos móveis, redes sem fio e *softwares* aplicados para monitoramento de saúde pessoal e em sistemas de telessaúde". (Pitarma et al. [8]).

O conceito de AAL aplica-se oportunamente aos ambientes e estabelecimentos em que é necessário acompanhar as atividades diárias de pessoas que não possuem condições físicas e/ou mentais, visando suas saúdes pessoais (como idosos e pessoas com necessidades especiais). Este ambiente controlado auxilia na monitoração diária da pessoa, distribuída em diversas medições, sendo a maioria adepta ao sistema IoT. Como aplicações existentes, podemos citar: o monitoramento de sono, acompanhamento cardíaco, detecção e auxílio em quedas, controle de qualidade do ar, entre outros existentes.

Os problemas de se lidar com as quedas são diversos, pois são situações em que se necessitam de uma resposta rápida, que na maioria das vezes não é fácil de se conseguir. Acrescenta-se a isto que, os cenários de uma eventual queda podem gerar situações nas quais um idoso sozinho em sua casa não terá ajuda de um responsável para lhe auxiliar. Existem dispositivos no mercado que detectam a queda de idosos, porém a maioria não possui como atividade principal a detecção de quedas utilizando o conceito de IoT. Com isso, muitas funcionalidades presentes nos equipamentos que não possuem esse recurso como elemento principal não serão utilizados. Além disso, a maioria destes dispositivos executam o processamento dos dados neles mesmos, reque-

rendo assim, mais robustez computacional nos equipamentos físicos e aumentando seu custo ao consumidor final.

Tendo em vista a necessidade de flexibilizar a criação de ambientes de vida assistida mais personalizados e permitir a sua disseminação em ambientes domésticos, propomos neste trabalho uma solução completa que começa com a coleta de dados por sensor local. Este dispositivo envia as informações de aceleração para o *gateway* local, que faz a conexão com a Internet, enviando os dados agregados de todos os sensores para o servidor central. Este componente realiza o processamento das informações em nuvem por meio da filtragem dos sinais e tomada de decisão por limiar de detecção. Por fim, há uma entrega de notificação via mensagem instantânea em caso de queda.

1.2 Objetivos

Este trabalho pretende apresentar uma solução completa para monitoramento de quedas de pessoas idosas baseada no conceito de Internet das Coisas, ou seja, é proposta e implementada a arquitetura de uma aplicação para monitoramento, tendo como base o uso de sensores, coleta de dados, transmissão via Internet para nuvem, processamento e alarme imediato ao ocorrer o evento.

1.2.1 Objetivos Específicos

- Coletar, armazenar e transportar dados de aceleração de forma contínua através de um micro-controlador.
- Coordenar requisições ao servidor central através de um *gateway* local.
- Estudar e implementar método de detecção de quedas via uso de Transformada Wavelet e decisão por limiar.
- Implementar envio de informação de retorno através de aplicativo de mensagens instantâneas.
- Realizar testes e apresentar os resultados para a validação do sistema em cenários reais.

1.3 Organização do Trabalho

A partir do próximo capítulo, o trabalho será dividido seguindo esta ordem: no Capítulo 2 apresentamos uma fundamentação teórica, onde será abordada toda a teoria julgada necessária

para o entendimento e reprodução do projeto. No Capítulo 3 são revisados trabalhos científicos que utilizam diferentes abordagens para a detecção de quedas. O objetivo da revisão é o posicionamento do trabalho frente aos projetos já publicados.

No Capítulo 4, referente à metodologia e implementação do sistema, será descrito a solução construída para a pesquisa executada, dissecando a complexidade da arquitetura distribuída e servindo como uma espécie de tutorial para uma posterior reaplicação do método científico e alcance de outros objetivos de pesquisa na área de IoT. No Capítulo 5, de análise de resultados, são apresentados dados coletados pelo sistema previamente estabelecido, e explicações demonstrando a eficácia do método aplicado para resolução do problema e conquista dos objetivos previamente estabelecidos. Por fim, no capítulo 6, temos a conclusão do projeto, onde também apresentam sugestões para trabalhos futuros na área descrita.

Capítulo 2

Fundamentação Teórica

Neste Capítulo serão apresentados os conceitos julgados necessários para entendimento total do presente trabalho abordando conteúdos de Internet das Coisas e Processamento de sinais não-estacionários.

2.1 Internet das Coisas

Para entender o conceito de Internet das Coisas, precisamos explorar primeiramente o conceito da comunicação máquina-a-máquina (M2M), (do inglês *Machine-to-Machine*). Segundo Höller et al. [9] a interação entre máquina-a-máquina é caracterizada por ser uma comunicação composta por dispositivos utilizando-se de meios cabeados ou não cabeados de transferência de dados. Com a idealização de tal sistema máquina-a-máquina, tem-se como desígnio o monitoramento remoto de eventos relacionados a algum ativo ou ainda o controle atuante de tais objetos. Exemplos de sistemas M2M podem ser encontrados principalmente em ambientes industriais, onde o controle automatizado das várias "máquinas" podem gerar ganhos de produtividade e redução de custos. Apesar da rede de dispositivos estabelecida, as aplicações M2M não tem como característica própria a intercomunicação através da Internet mundial, impossibilitando o monitoramento em qualquer lugar.

Em contraste ao conceito M2M, segundo Höller et al. [9] "IoT também se refere à conexão de tais sistemas e sensores à Internet inteira, assim como ao uso de tecnologias gerais da Internet. No longo prazo, é previsto que o ecossistema IoT se tornará equivalente à Internet que temos hoje, permitindo que as coisas e objetos do mundo real se conectem, comuniquem-se e interajam entre si do mesmo jeito que humanos fazem pela *web*". Portanto, no futuro teremos uma Internet não somente composta de interações interpessoais, como também de relações máquina-a-máquina e máquina-humano, todas gerando troca de informações e formação de conhecimento.

Com o passar dos anos, o conceito de IoT foi evoluindo e diversificando para diversas áreas do

conhecimento. Logo, a essência do que representa a "coisa" está um pouco difusa atualmente. De acordo com Sundmaeker et al. [7], coisas são participantes ativos nos processos de negócios, de informações e processos sociais, onde elas interagem entre si, trocando informações sobre o ambiente, enquanto reagem de forma autônoma aos eventos físicos e produzem influência sobre estes eventos, executando processos que desencadeiam ações e criam serviços com ou sem intervenção humana.

2.1.1 Desafios

Sendo a Internet das Coisas uma tecnologia inovadora, vários desafios deverão ser encarados para sua implementação e realização de fato. Segundo Chen et al. [10], abaixo estão os principais desafios da Internet das Coisas.

- **Arquitetura :** Com o desenvolvimento proposto pelo IoT, observamos que há uma grande abrangência de tecnologias que esse sistema possui. Esta gama de diversidade, deve-se aos sensores inteligentes que miram em formas não intrusivas para a coleta de dados. Surge assim, a utilização de redes sem fio autônomas, de modo que a transmissão de dados ocorra naturalmente. Desta forma, os sensores adquirem a possibilidade de mobilidade, requerendo arquiteturas robustas e sensíveis à latência, gerando assim, desafios antes inexistentes.

- **Complexidade :** Sistemas IoT devem ter em mente o baixo custo e a conectividade confiável, garantindo a entrega dos dados. Porém, a adição constante de dispositivos faz com que o sistema acabe tendo uma arquitetura heterogênea, para conseguir atender as demandas das "coisas". Essa heterogeneidade se dá pelo fato dos dispositivos possuírem suas particularidades próprias em questões de protocolos de comunicação, o que faz com que alguns dispositivos tenham um alcance mais limitado, enquanto outros se comuniquem apenas por protocolos legados, ou ainda tenha aqueles que não tenham segurança em mente. Esta diversidade de "coisas" traz uma complexidade a mais à configuração, que tem como finalidade a criação de uma rede que faça os dispositivos conversarem entre si.

- **Hardware :** Tecnologias como IoT levam à necessidade de dispositivos cada vez mais inteligentes, requerendo assim, alto processamento físico. Por outro lado, de modo a facilitar a usabilidade dos sensores, vemos a necessidade de ter dispositivos com sistemas de transmissão sem fio, tamanho reduzido, baixo custo e funcionalidades suficientes. Portanto, surge uma dificuldade quando verificamos o obstáculo gerado pela falta de processamento e armazenamento dos sensores, nos quais, inviabiliza qualquer forma de análise de dados no equipamento limitado.

- **Privacidade e Segurança :** Segundo Farooq et al. [11], IoT é uma grande revolução tecnológica, pois conseguiu alterar a infraestrutura atual que conhecemos da Internet, onde

os objetos presentes nessas arquiteturas são onipresentes entre si. Porém, mesmo sendo uma grande expectativa para o futuro, possui um potencial desastroso em vista da segurança. Pela sua arquitetura, sistemas IoT necessitam cobrir mais objetos e níveis de gerenciamento do que a rede tradicional. Cria-se um dilema, pois como visto, não há muita disponibilidade computacional disponível nos sensores atuais.

- **Padronização :** A eficiência do sistema IoT depende da coordenação das múltiplas partes na construção de modelos de informação e protocolos de comunicação que permitem que todos os colaboradores e entusiastas do IoT possam usar, atuar e promover novas aplicações e serviços.

- **Negócio :** O IoT não nos apresenta uma solução única para todas as categorias de indústrias. Logo, é complicado avaliar um modelo de negócio singular que viabilize a implementação de um sistema alicerçado em uma arquitetura de Internet das Coisas. Essa dificuldade na antevisão dos impactos de negócio representa uma barreira de entrada, fazendo com que muitos sistemas optem por não empregarem o uso de IoT em suas aplicações.

2.1.2 Arquitetura

Conforme Li et al. [12], um requisito crítico para implantação de uma solução baseada no paradigma de IoT surge da necessidade de as "coisas" estarem interconectadas, garantindo assim, um preenchimento da lacuna entre mundo físico e virtual. Sabendo que essas "coisas" podem se locomover fisicamente, eventualmente, surge a necessidade de definição de uma arquitetura adaptativa e dinâmica, sendo assim, descentralizada e de natureza heterogênea. A Figura 2.1 apresenta uma proposta de arquitetura de solução IoT, subdividida em 4 camadas, conforme propõe Li et al. [12].

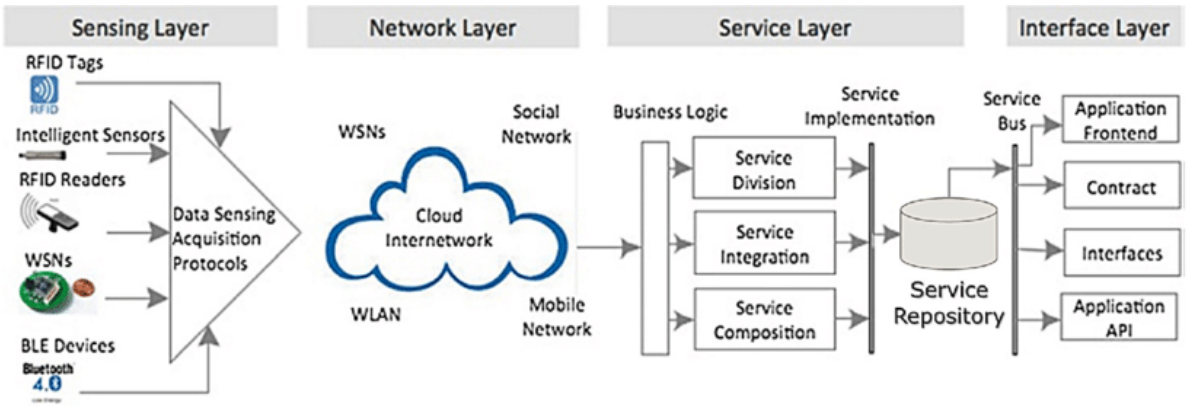


Figura 2.1: Arquitetura IoT [12].

2.1.2.1 Camada de Sensoriamento

Na camada de sensoriamento estão presentes os sensores, elementos fundamentais, responsáveis por captar informações onde estão implantados. Esta camada tem como principais funções: identificação das "coisas" e coleta de informações a partir destas. Os dispositivos presentes nessa camada variam de acordo com a aplicação. As informações que são coletadas por esses sensores podem estar relacionadas à localização, mudanças no ar, ambiente, movimento, vibração, entre outros. Exemplos de sensores encontrados nesta camada são: rótulos RFID que identificam os objetos, sensores e relógios inteligentes que coletam informações do ambiente. O sistema apresentado no Capítulo 4 possui em sua camada de sensoriamento um sensor de aceleração e o micro-controlador associado.

Na concepção da camada, vários aspectos devem ser considerados como:

- **Custos de Implementação e Manutenção:** Devido à alta quantidade de sensores, os dispositivos devem ser capazes de proporcionar que recursos requeridos sejam minimizados e, consequentemente, os custos sejam mitigados. Além disso, devido à exigência dos sensores estarem ligados a todo o tempo, há o desafio de prover energia a todo esse conjunto de dispositivos. A eficiência energética garante que os sensores trabalhem por mais tempo sem descontinuidade do serviço.

- **Escalabilidade:** Os sensores podem ser adicionados e removidos a qualquer momento, conforme demandado. Em situações de adição de novos equipamentos à camada, deve-se considerar fatores que influenciam no sistema como um todo. Por exemplo, os meios de comunicação devem atender a adição dos novos dispositivos. A utilização de protocolos como Wi-Fi 5GHz ou 5G aumentam a largura de banda e melhoram a escalabilidade do sistema. Custos de implementação e manutenção também são aspectos afetados pela escalabilidade.

- **Heterogeneidade:** As diferentes propriedades dos sensores podem tornar o sistema heterogêneo, forçando assim, os dispositivos de controle presentes no sistema a possuírem a necessidade de gerenciar toda a comunicação entre dispositivo-dispositivo e dispositivo-nuvem, requerendo a capacidade de operar em tecnologias distintas.

- **Segurança:** Dispositivos IoT, por terem foco em possuir baixo-custo, acabam deixando a parte de segurança de lado, utilizando sistemas operacionais muito antigos ou ainda protocolos de comunicação inseguros. Um ataque comum é o *Eavesdropping*, onde o atacante escuta a mensagem originada do sensor de forma não-autorizada, obtendo informações privilegiadas. Medidas como atualização frequente dos sistemas operacionais dos equipamentos, utilização de protocolos de rede seguros, e controle de acesso físico aos dispositivos ajudam a refinar a segurança do sistema.

2.1.2.2 Camada de Rede

A camada de rede tem como objetivo principal ser a parte da arquitetura capaz de interconectar a camada sensorial com a camada de serviço. Assim, ela leva os dados dos sensores aos servidores que possuem a função de tomada de decisão. O roteamento é o aspecto mais importante desta camada, visto que normalmente a solução promove o fluxo de informações por diversas redes interligadas. Estas redes podem ser de variados tamanhos como redes pessoais, locais e globais. Os dispositivos que fazem a ponte entre tais redes são os principais componentes desta camada. O sistema apresentado no Capítulo 4 possui em sua camada de rede o *gateway*.

Um dos protocolos que se destaca no paradigma do IoT é o 6LoWPAN (do inglês *IPv6 over Low Power Wireless Personal Area Networks*), sendo padronizado pelo IETF (do inglês *Internet Engineering Task Force*) tem como principal função apresentar o protocolo IP adaptativo a cada camada com baixo consumo de energia e perdas. Adjunto com o 6LoWPAN podemos destacar o RPL (do inglês *Routing Protocol for Low-Power and Lossy Network*), apresentando um protocolo com funcionalidades de roteamento sobre o 6LoWPAN, prezando o baixo consumo de energia e perdas. Outra importante função, surge do fato de ser uma camada agregadora das "coisas" presentes no sistema IoT. Com isto, torna-se possível a utilização de ferramentas para garantia de serviços (*QoS*), de modo a fornecer serviços diferenciados para usuários e/ou aplicativos.

A segurança é outro ponto de frequente responsabilidade para esta camada, pois pela sua importância, se torna um dos principais alvos do sistema. Sendo uma camada semelhante às já conhecidas em dispositivos comuns, os atacantes utilizam as mesmas estruturas de ataque, tais como: ataque de negação de serviço (DoS, do inglês *Denial of Service*) e ataque MitM (do inglês *Man-in-the-Middle*).

2.1.2.3 Camada de Serviços

Na maioria das vezes, a informação trazida da camada sensorial não está pronta para ser direcionada ao usuário final do sistema. Logo, faz-se necessário que haja uma manipulação dos dados, para que o sistema passe a informação ao usuário de forma mais clara possível. A camada de serviço tem como função principal a agregação dos dados de todos os objetos em um armazenamento local, e tratamento das informações conforme especificado pela aplicação. Exemplos de tratamentos realizados na camada de serviço são: adição de dados em um algoritmo de aprendizado de máquina, montagem de gráficos que mostram a evolução da métrica ao longo do tempo, cálculo da média móvel da medida com base nos últimos x valores, dentre outros. Esta camada interage com a camada de interface, passando a informação que será mostrada para o usuário final. Os componentes presentes nesta camada são servidores que armazenam e aplicam métodos de processamento sobre os dados recebidos. Estes dispositivos geralmente se encontram na Internet, pois desta maneira podem prover serviços a objetos geograficamente distantes. O sistema

apresentado no Capítulo 4 possui em sua camada de serviços o servidor central.

2.1.2.4 Camada de Interface

A camada de interface possibilita que o usuário final possa interagir com o sistema, recebendo as informações tratadas pela camada inferior. Ela é responsável por definir todas as aplicações que utilizam o paradigma do IoT, bem como seus relacionamentos com os dados recebidos. As aplicações podem ser diversas, desde casas inteligentes à rastreamento de animais ou ainda aplicativos de celular. Estas aplicações também podem ser usadas para verificação e gerenciamento centralizado do sistema, proporcionando uma interação mais ativa do usuário com a solução. O sistema apresentado no Capítulo 4 possui uma interface de mensagem para a notificação quando houver a detecção de quedas, sendo que esse recurso está localizado na camada apresentada.

2.2 Processamento de sinais não-estacionários

Ao longo de décadas, com a necessidade provocada por algumas aplicações, apresentou-se uma nova área de pesquisa chamada de processamento de sinais, na qual consiste na análise e/ou modificações atribuídas em sinais utilizando ferramentas já conhecidas. Estas são utilizadas para extrair ou modificar informações apresentadas por sinais complexos. No decorrer dessa seção será apresentada algumas dessas ferramentas mantendo o foco em sinais não-estacionários.

2.2.1 Transformada de Fourier e Transformada de Fourier de Tempo Curto

Segundo Barford et al. [13] olhando na perspectiva de sinais a Transformada de Fourier é mais adequada para analisar sinais estacionários, pois fornecem um espectro único para todo o sinal. Para sinais não-estacionários o maior interesse está localizado nas frequências dominantes em um determinado período finito de tempo. Assim surge a Transformada de Fourier de Tempo Curto (STFT, do inglês *Short-Time Fourier Transform*) apresentando o cálculo da Transformada de Fourier em janela subdivididas no sinal completo, logo existe uma Transformada de Fourier para cada posição da janela no sinal, onde essas transformadas produzidas com o andamento da janela ao longo do tempo constituem a STFT.

A Transforma de Fourier de Tempo Curto possui algumas limitações, na qual a principal delas surge do fato da janela ser constante em todo o sinal. Caso a janela for larga haverá uma baixa resolução temporal do sinal. Já se a mesma for estreita, isso provocará uma baixa resolução espectral do sinal, fazendo com que a análise do sinal seja prejudicada.

2.2.2 Transformada Wavelet

Wavelet é um tipo de onda característica por ser uma oscilação momentânea em um curto espaço de tempo, possuindo valores nulos antes e após a oscilação. Exemplos de Wavelets comprimidas são mostradas na Figura 2.2. Wavelets possuem duas propriedades principais: escala e localização. A escala define o quão dilatada a Wavelet é. Aumentar a escala significa alargar a Wavelet, enquanto diminuir a escala faz com que a Wavelet seja mais curta. Já a localização define em qual momento no tempo a onda está posicionada. Diminuir a localização resulta em uma Wavelet mais adiantada, enquanto aumentar a mesma atrasa a onda Wavelet.

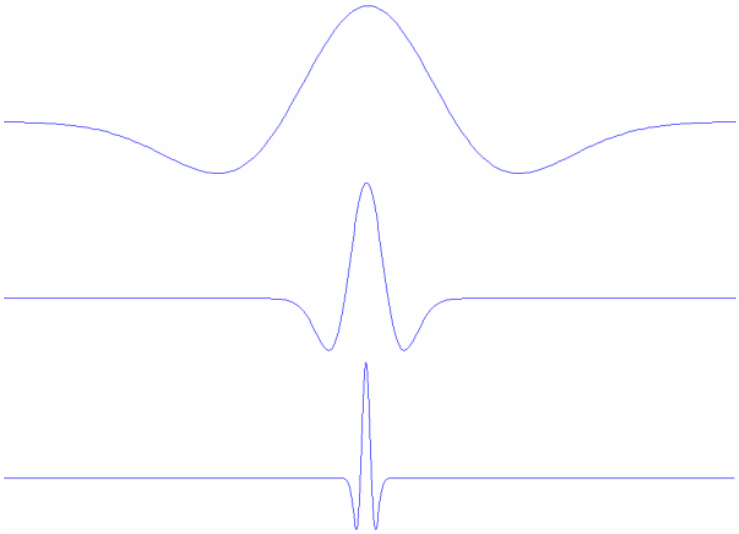


Figura 2.2: Exemplos de ondas Wavelet comprimida de modo varrer as escalas mais altas. [14]

A Transformada Wavelet tem como ideia básica extrair de um sinal o quanto de uma Wavelet está presente. Ou seja, é realizada a convolução do sinal com uma Wavelet de referência. Primeiramente, é escolhida uma Wavelet mãe, que é utilizada como base para todas as outras ondas Wavelet que serão utilizadas. A operação de convolução é feita diversas vezes, alterando os parâmetros de escala e localização da Wavelet mãe. A definição da Wavelet é dada por

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right) \tag{2.1}$$

em que a e b representam os parâmetros de escala e localização, respectivamente, a função $\psi(t)$ é chamada de Wavelet "mãe", base para as outras funções Wavelet, e t é a medida contínua do tempo. Logo, $\psi(\frac{t-b}{a})$ representa todas as Wavelets derivadas da Wavelet mãe ψ.

A Transformada Wavelet é definida por

$$WT_\psi\{x(t)\}(a,b) = \int_{\Re} x(t)\frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right)\,dt \tag{2.2}$$

em que $WT_\psi\{x(t)\}(a,b)$ é realizada sobre um sinal de entrada $x(t)$. A sua principal vantagem está no fato da operação conseguir ter janelas de análise de tamanhos diferentes. O tamanho das janelas é controlado pela escala da Wavelet, onde uma janela grande é ideal para sinais de baixa frequência, pois estes duram mais. Segundo o Princípio da Incerteza no Processamento de Sinais, quanto maior o Δt, que é o tamanho da janela, menor será o Δf, gerando um sinal mais localizado na frequência. Por outro lado, janelas pequenas são melhores usadas em sinais curtos, como Wavelets. Janelas pequenas representam um Δt pequeno e, consequentemente tornam o sinal menos preciso na frequência (Δf maior). Com a Transformada de Fourier de Tempo Curto, Δt e Δf possuem valores fixos, prejudicando a análise de um sinal de baixa frequência caso a janela seja pequena ou um sinal de alta frequência em caso de janela grande.

2.2.2.1 Transformada Wavelet Discreta

Conforme relata Tzanetakis et al. [15], a Transformada Wavelet Discreta (DWT) (do inglês *Discrete Wavelet Transform*) é um caso especial da Transformada Wavelet, provendo uma representação do sinal tanto no domínio do tempo quanto no domínio da frequência, e que podem ser calculadas de forma eficiente.

Na perspectiva da Transformada Wavelet discreta, de acordo com Edwards [16], a onda Wavelet é uma função ortogonal que pode ser aplicada a um grupo finito de dados. Na prática, a DWT é muito similar à Transformada de Fourier discreta, pois a função transformadora é ortogonal, além de que em ambas as transformadas, o sinal de entrada é mapeado em um conjunto de valores discretos no tempo. As duas transformadas são operações de convolução entre o sinal transformado e a função base, todavia, enquanto a Transformada de Fourier discreta utiliza como função base uma senoide complexa, a DWT utiliza como base uma função Wavelet, sendo transladada no tempo e com seus devidos coeficientes de compressão.

A Transformada Wavelet discreta é definida por

$$WT_\psi[j_0,k] = \sum_{n=0}^{N-1} x[n]\psi_{j_0,k}[n] \quad \forall\, k \tag{2.3}$$

em que $x[n]$ é o sinal discreto de entrada, o j_0 é o fator de dilatação da Wavelet $\psi_{j_0,k}$, k o parâmetro de translação da Wavelet, e n os valores discretos do tempo, variando de 0 a $N-1$. A Transformada Wavelet, em seu modelo discreto, pode ser interpretada como um simples procedimento de filtragem. Daubechies [17], em seu trabalho, propôs implementar algumas Wavelets utilizando filtros FIR (do inglês *Finite Impulse Response*), sendo compostos principalmente de um filtro passa baixa (LPF, do inglês *Low-Pass Filter*) e outro passa alta (HPF, do inglês *High-Pass Filter*). Assim, o sinal foi decomposto em dois tipos de componentes, sendo eles: os que representam as altas frequências do sinal (coeficientes detalhados) e os que representam as baixas

frequências (coeficientes aproximados). A Figura 2.3 ilustra essa situação.

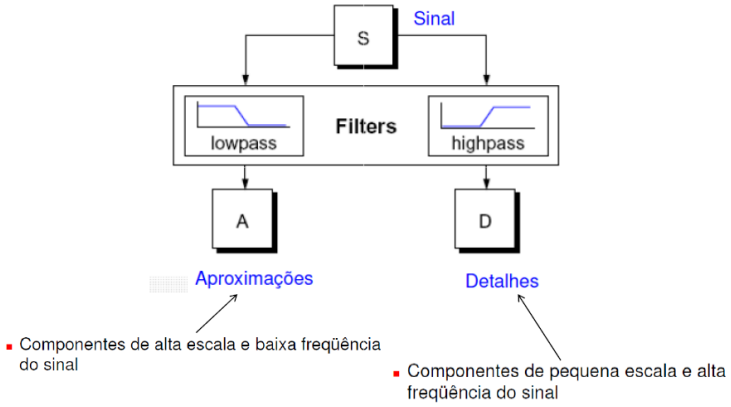


Figura 2.3: Coeficientes DWT [18].

Sendo $x[n]$ um sinal discreto, obtido por uma amostragem F_s do sinal contínuo $x(t)$, e as respostas impulsionais dos filtros HPF e LPF, em primeiro nível, representados respectivamente, por g_1 e h_1, temos que:

$$d_1[k] = \sum_n g_1[n-2k]x[n] \tag{2.4}$$

$$c_1[k] = \sum_n h_1[n-2k]x[n] \tag{2.5}$$

Em que a sequência $d_1[k]$, representa os coeficientes detalhados do sinal e $c_1[k]$, representam os coeficientes aproximados do sinal. Se executarmos outro nível de filtragem, teremos $c_1[k]$ como uma nova entrada, com uma taxa de amostragem igual a $F_s/2$, gerando assim, $d_2[k]$ e $c_2[k]$ com uma taxa de amostragem de $F_s/4$. Esta decomposição em multiníveis se apresenta como uma ferramenta poderosa para filtragem de sinais, podendo ser executada diversas vezes com o intuito de separar os elementos de alta e baixa frequência no sinal, no qual o limite é igual a $\log_2 n$. Não existe uma quantidade ideal de decomposições, dependo sempre, do sinal e do nível de filtragem requerida. No sistema apresentado no Capítulo 4 foi utilizado esta técnica, de modo a filtrar o sinal de aceleração para eliminar movimentos distintos à queda, apresentando assim, uma melhor acurácia ao sistema. A Figura 2.4 mostra um exemplo de decomposição do sinal $x[n]$ em dois níveis.

2.3 Discussão Geral

Neste capítulo foram explicados os conceitos julgados necessários para entendimento teórico do trabalho, envolvendo Internet das Coisas e Processamento de Sinais Não-Estacionários. No

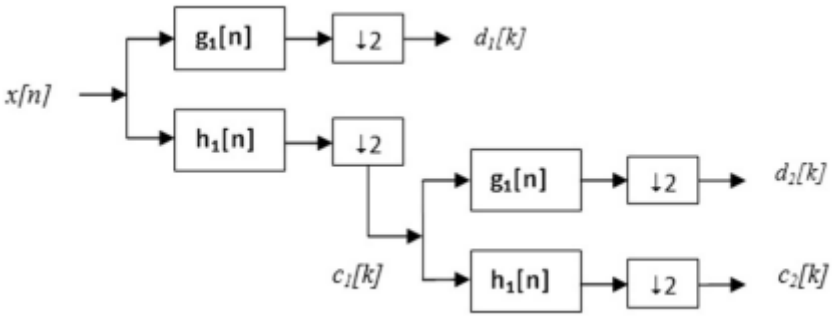


Figura 2.4: Decomposição em dois níveis [19].

Capítulo 4, onde tem-se a explicação do sistema desenvolvido para detecção de quedas de forma intrusiva, é explicado como a teoria foi aplicada no presente trabalho.

Capítulo 3

Revisão Bibliográfica

Ambientes de Vida Assistida englobam sistemas desenhados para o suporte de pessoas idosas e com necessidades especiais em suas rotinas do dia-a-dia. O objetivo principal de tais ambientes é preservar a autonomia de seus usuários em cenários residenciais. A implementação de Ambientes de Vida Assistida por meio da utilização de sensores tem sido bastante empregada em trabalhos científicos. Este capítulo expõe alguns trabalhos encontrados na área que utilizaram métodos intrusivos e não-intrusivos, com a coleta de informações de diversos tipos de sensores como acelerômetros, sensores infravermelhos, sensores de vibração, etc.

3.1 Rede de sensores baseados em acelerômetros para detecção de quedas

No trabalho de Le et al. [20] é apresentado um sistema de detecção de quedas utilizando-se de dois sensores com três eixos ortogonais de aceleração. O sistema realiza a percepção de quedas de forma intrusiva, pois os sensores ficam posicionados no tronco e na coxa da pessoa, respectivamente.

Neste sistema, a detecção é realizada através de duas etapas: A primeira etapa, ilustrada na Figura 3.1 é denominada *Acceleration Breaching and non-breaching* e envolve a análise do sinal de aceleração em uma janela de tempo chamada *Time Limit*. Caso a aceleração passe do limiar e volte a ficar dentro do limiar durante o *Time Limit*, um sinal de interrupção é gerado e a análise parte para a próxima etapa.

Na segunda etapa, temos a análise de postura do usuário. A Figura 3.2 mostra que a partir da leitura dos sensores, o sistema interpreta a posição que a pessoa está. Por exemplo, caso o acelerômetro preso a coxa tenha o componente X igual à zero (X2=0), o sistema interpreta a pessoa como levantada e não reporta queda. Caso contrário, se o eixo X do acelerômetro do tronco

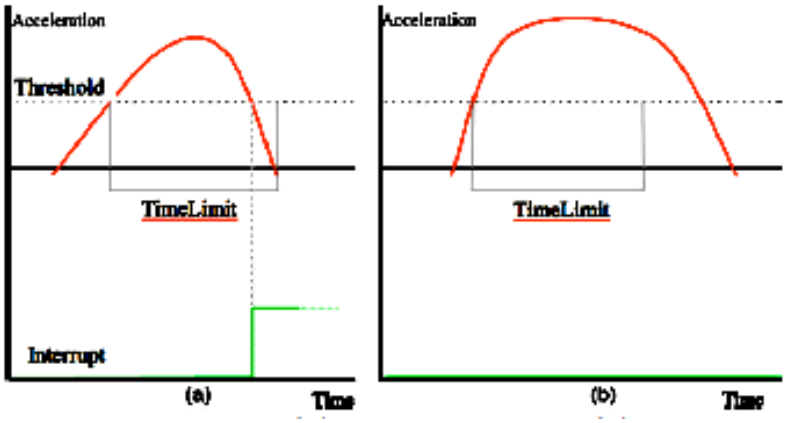


Figura 3.1: Etapa 1: a) *Acceleration Breaching* e b) *Acceleration Non-Breaching* [20].

seja igual à 1 (X1=1), é deduzido que o usuário se encontra deitado, e uma queda é detectada. Quando não, caso o eixo Z do acelerômetro da coxa seja 0 (Z2=0), o sistema entende que a pessoa está sentada. Outros cenários são compreendidos como não-queda e o sistema volta a primeira etapa.

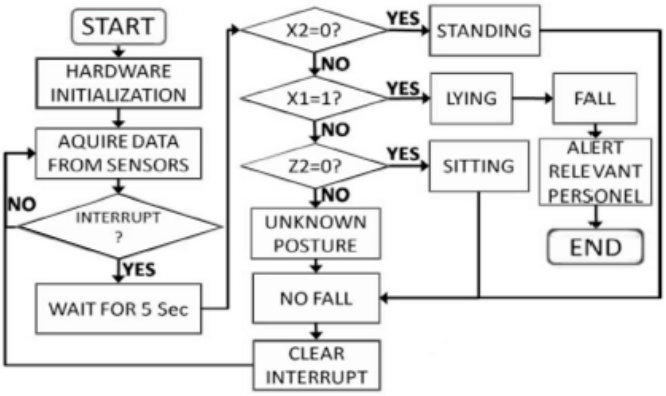


Figura 3.2: Fluxograma de detecção de quedas [20].

O processamento do sistema é realizado por um microcontrolador conectado por interface SPI (do inglês *Serial Peripheral Interface*) aos acelerômetros. Após processamento, os resultados são transmitidos para um computador por uma conexão RS232 (do inglês *Recommended Standard 232*). O computador apresenta a decisão do algoritmo de quedas por um aplicativo. Segundo Le et al. [20], o sistema proposto, com o algoritmo de detecção de quedas e registro de postura obteve sucesso em testes reais.

3.2 Detecção de quedas em casa utilizando vibrações do chão e som

Na pesquisa de Litvak et al. [21] foi descrito um estudo de caso sobre quedas domiciliares usando a vibração do chão e o som produzido pela queda. No trabalho realizado, foram utilizados um acelerômetro e um microfone como sensores. Estes estavam localizados no canto do cômodo, sendo sensíveis o suficiente para o reconhecimento dos sinais. O projeto visava o desenvolvimento de uma solução completa, consistindo assim, na utilização de algoritmos de automação para a detecção do evento esperado. A Figura 3.3 apresenta o fluxograma do sistema para a detecção de quedas.

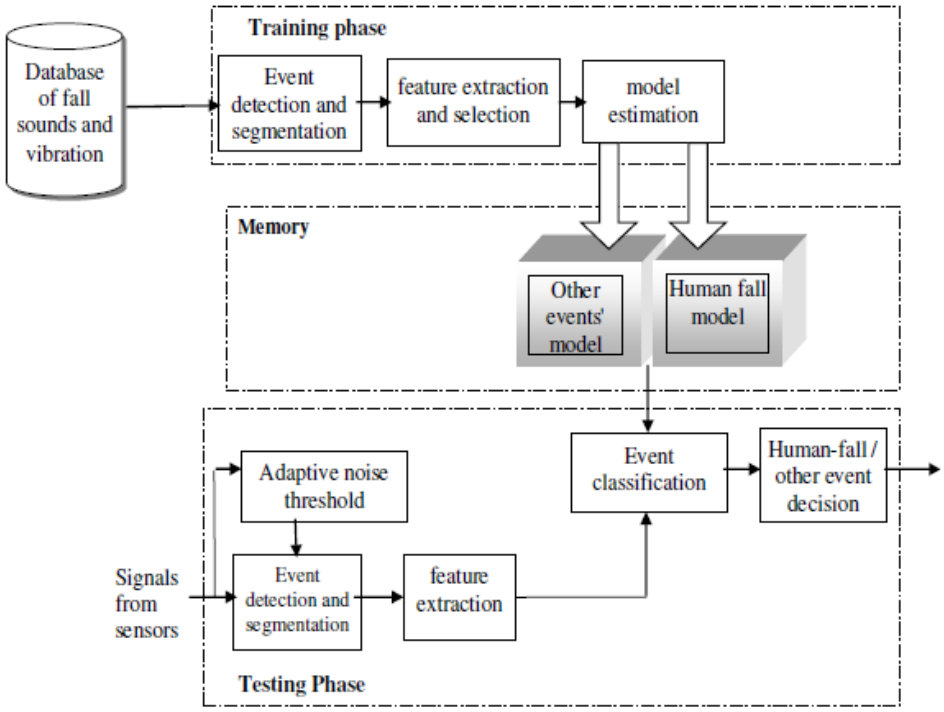


Figura 3.3: Fluxograma de detecção de quedas [21].

De modo a verificar os resultados obtidos pelo sistema criado, foi executado uma sequência de testes com um manequim especializado em quedas. A Figura 3.4 mostra os resultados obtidos com os testes. Foi alcançada uma sensibilidade de 97,5% enquanto a especificidade alcançada foi de 98,5%.

Os autores concluem que os resultados dos testes mostram que o sistema construído tem o potencial para servir como uma solução confiável na detecção de quedas. Além disso, deve-se destacar o baixo custo de implementação, e a abordagem não-intrusiva do sistema, que não necessita que o monitorado utilize algo no corpo para o funcionamento devido do projeto.

Real / Class. As	Objects ("other event")	Events close to the sensors ("other event")	"human fall"	"human fall" on a carpet
"other event"	44 (+4 undetected)	17	1	0
"human fall"	0	1	19	20

Figura 3.4: Resultados da classificação [21].

3.3 Abordagem INFOrM

O trabalho de Rodrigues [22] apresenta um sistema completo de monitoramento em Ambientes de Vida Assistida. Segundo o autor, "O INFOrM (do inglês *INdoor Fall detectiOn Method*) é um método para detecção de quedas que utiliza aprendizagem de máquina para geração de um modelo de classificação baseado em informações de alto e baixo nível, provenientes de acelerômetros e sensores infravermelhos passivos. O classificador utiliza tecnologia de fácil aceitação pelo usuário e possui fontes de informações redundantes e complementares que podem melhorar a detecção do sistema. Estas informações podem ser analisadas em conjunto ou de forma independente quando uma das informações não se encontra disponível, evitando a parada total do sistema".

O sistema apresentado utiliza a detecção de queda baseada em limiares de tempo, utilizando dados dos dois tipos de sensores. A solução recebe dados do acelerômetro (sensor presente em um *smartphone*), armazenando os tempos de entrada mínimo e máximo. Conforme observamos na Figura 3.5, se a diferença entre esses tempos for menor que 1,5 segundos, o algoritmo espera 2 segundos, de modo a identificar a postura do usuário e com isso dispara um alarme de queda, caso o mesmo esteja deitado. Se a diferença entre os tempos for maior que 1,5 segundos, o acelerômetro volta à coleta normal de dados.

Os sensores infravermelhos coletam informação de localização do usuário monitorado enquanto o acelerômetro acompanha o comportamento do indivíduo. Além disso, os dois tipos de sensores colhem dados de postura, provendo redundância. Desse jeito, têm-se informações diferentes que podem se complementar ajudando a reconhecer o estado do residente no ambiente.

Caso o *smartphone* não esteja fixo ao tronco da pessoa, apenas os sensores infravermelhos serão atuantes na detecção de quedas, informando a posição do usuário. Para que tal detecção seja realizada, é usado um limiar de tempo. Caso o idoso não seja detectado por nenhum dos sensores de luz infravermelha em um prazo de 12 segundos, o sistema considera que a pessoa se encontra em perigo. Caso contrário, se o indivíduo for encontrado em um dos cômodos, porém esteja no chão por mais de 12 segundos, também se considera que o usuário está caído.

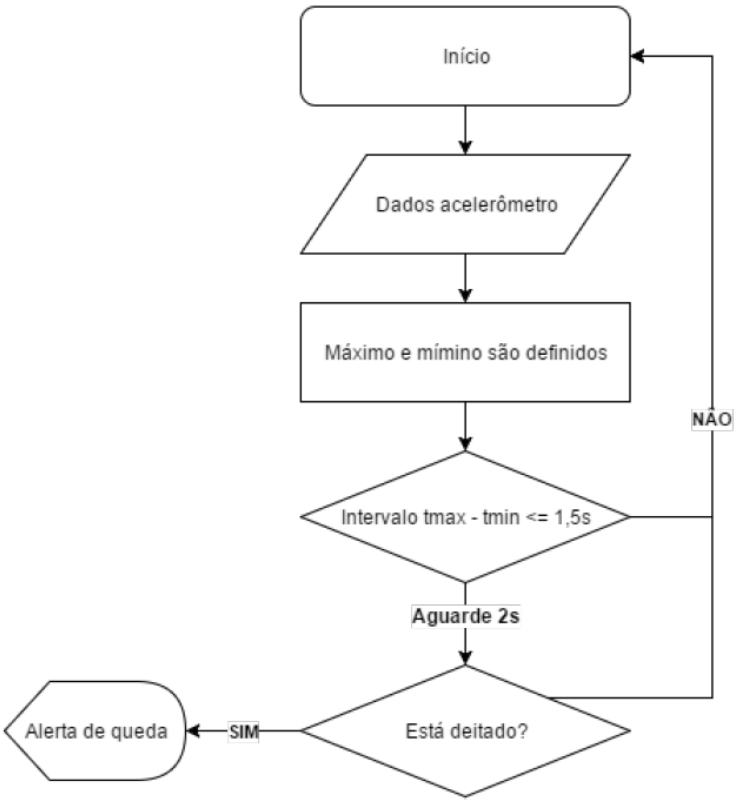


Figura 3.5: Fluxograma do algoritmo de detecção de quedas por limiares de tempo [22].

Nos resultados do trabalho, o autor valida o sistema utilizando diversos algoritmos de aprendizagem de máquina. A validação consistiu de duas formas. A primeira forma valida os resultados por validação cruzada, que é uma técnica cujo objetivo é verificar a capacidade de generalização de um modelo de *Machine Learning*. Nesta forma, as bases de teste são uma generalização do conjunto de dados de treino. Os resultados obtidos nesta fase mostraram um desempenho acima de 95% de todos os algoritmos em todas as bases dos 5 voluntários (U1, U2, U3, U4 e U5).

A segunda forma é avaliar a solução através de bases de testes independentes das bases de treinamento utilizadas, garantindo assim, uma maior consistência nos resultados, conforme é possível verificar na Figura 3.6, onde U1, U2, U3, U4 e U5 são os usuários do sistema. Observamos que o algoritmo *Bagging* foi o que saiu pior no desempenho, indiferente da base de dados utilizada, não sendo recomendado para esse sistema. Segundo Rodrigues [22], os resultados obtidos evidenciam ser possível utilizar a solução INFOrM em sistemas AAL que implementam detecção de quedas.

Bases de treino	Todos-U1	Todos-U2	Todos-U3	Todos-U4	Todos-U5
Base de teste	U1	U2	U3	U4	U5
Random Forest	91,37%	97,03%	97,41%	84,25%	96,52%
SVM	95,50%	95,32%	97,33%	86,63%	96,69%
Bagging	89,71%	92,63%	89,23%	78,90%	88,21%
Adaboost M1	92,85%	96,42%	97,38%	81,1%	91,96%

Figura 3.6: Índice de instâncias classificadas corretamente - Validação através de uma base não usada no treino [22].

3.4 Detecção de quedas utilizando acelerômetros distribuídos pelo chão

No trabalho de Werner et al. [23] foi desenvolvido um protótipo de detecção automática de quedas, que se utiliza de acelerômetros distribuídos no chão, de modo a coletar sinais de vibrações corporais que normalmente ocorrem na queda. Esta visão tenta gerar uma abordagem que minimizam limitações geradas por detecções intrusivas. O protótipo criado foi avaliado em laboratório e durante 507 dias na realidade de pessoas idosas em suas casas.

Os sensores foram distribuídos pelos cômodos, de tal forma que, cada caixa de sensores ocupe uma parede do ambiente conforme podemos observar na Figura 3.7. Cada caixa de sensores é composto por um acelerômetro, um amplificador, um filtro e um conversor A/D.

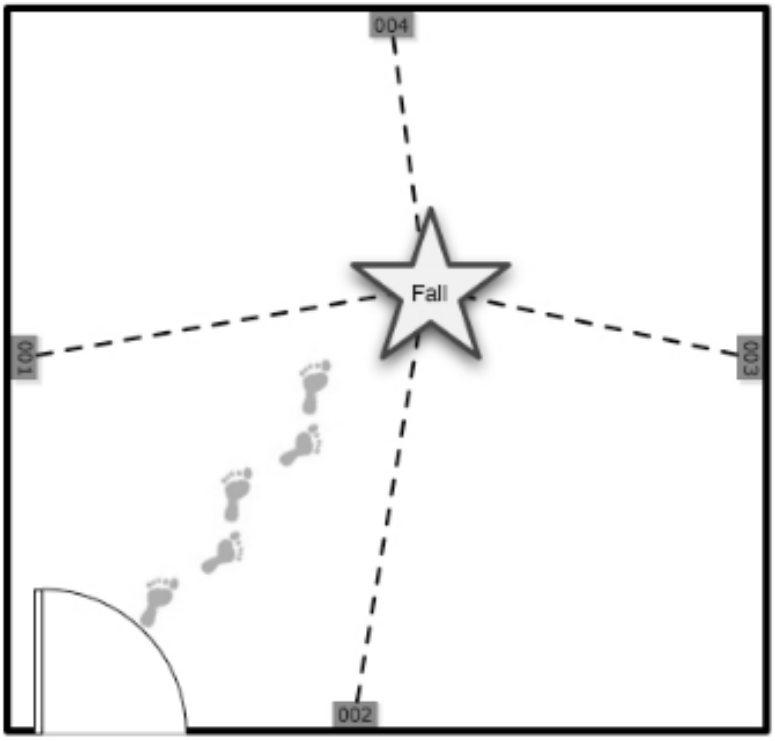


Figura 3.7: Posição das caixas de sensores em um ambiente [23].

As etapas que compõem a detecção de quedas são separadas em:

- **Coleta de dados e extração de funcionalidades**: etapa onde as vibrações são recebidas individualmente pelos sensores e há a extração de funcionalidades, tanto no domínio do tempo como no da frequência, e que serão usadas no algoritmo de detecção.
- **Fusão de dados**: consolidação das informações advindas de todos os sensores em um servidor central, que roda o algoritmo de reconhecimento de padrões baseado nas funcionalidades extraídas, e identifica potenciais eventos de queda.
- **Análise de Comportamento**: nesta etapa, o sistema interage com o monitorado de forma visual e por som. A resposta do usuário ajuda principalmente a informar o sistema se a pessoa conseguiu se levantar do tombo, diminuindo a quantidade de falsos positivos e mantendo a independência do idoso.
- **Classificação**: a última etapa classifica os eventos potenciais de quedas em não-queda ou queda baseando-se em limiares estabelecidos para cada parâmetro utilizado na detecção, realizando as ações devidas em casos de queda.

Foram realizados dois teste laboratoriais, no qual o primeiro contou com 44 quedas utilizando um boneco de teste (*dummy*) e 139 quedas de objetos aleatórios. Conforme mostra a Figura 3.8, das 44 quedas, 6 não foram identificadas, além de 11 quedas de objetos serem reconhecidas como quedas reais.

# of impacts	dummy	objects
Fall detected	38	11
No fall detected	6	128
Total	**44**	**139**

Figura 3.8: Resultados do primeiro teste laboratorial [23].

Já no segundo teste laboratorial, observou-se a interferência causada pela diferença no material do piso. Apesar do segundo teste ser feito com a ajuda de voluntários, nenhuma queda real foi realizada, o que afetou a verificação da eficácia do sistema em situações reais envolvendo quedas humanas. Para conseguir calcular a sensibilidade e especificidade do sistema, utilizando o boneco de teste, foram reproduzidas 23 quedas. Enquanto isso, 684 não-quedas foram executadas pelos voluntários, resultando assim nos resultados apresentados na Figura 3.9. Segundo estes valores, foi obtida uma sensibilidade de 87% e uma especificidade de 97,7% para os 4 pisos diferentes testados.

Além dos dois testes em laboratório, foram feitos testes em campo, em que o sistema foi instalado em 15 ambientes diferentes, com o objetivo de mensurar o desempenho do protótipo

Tested floors 1-4	Fall	No fall
Test positive	20	16
Test negative	3	668
Total	**23**	**684**

Figura 3.9: Resultados do segundo teste laboratorial [23].

em situações de vida real onde há a presença de muitas fontes de ruído. Porém, em um prazo de 507 dias, nenhuma queda aconteceu e por isso, a sensibilidade de quedas reais não pôde ser calculada. Logo, utilizaram-se os dados destes testes exclusivamente para o cálculo da taxa de falsos positivos. O resultado mostrou que 20 quedas foram equivocadamente detectadas, com notoriedade para os resultados envolvendo o sujeito de teste 2 em uma sala de estar, situação essa que gerou 7 quedas devido à presença frequente de uma criança brincando no recinto. No total, de 1325 eventos, 20 foram interpretados como quedas.

Após análise dos resultados dos testes de laboratório e de campo, os autores afirmam que não foi possível avaliar de forma concreta o desempenho do sistema contra quedas reais, visto que em laboratório as quedas consistiram de: bonecos de teste caindo; e voluntários, que apesar de representarem de forma honesta a queda, não podem ser consideradas realistas, pois o próprio reflexo induz a pessoa a se proteger da queda, o que segundo a literatura, é um dos principais motivos para não detecção de quedas (falsos negativos) utilizando vibrações. Em campo, ainda que em situações de vida real, nenhuma queda foi praticada, impossibilitando o cálculo da sensibilidade em tal cenário, e fazendo com que a medição do desempenho do sistema em tais situações fosse prejudicado, pois, apenas a especificidade pôde ser calculada. Ao final conclui-se que existem vantagens na forma não intrusiva de detecção comparada a forma intrusiva, mas com ela surge outras limitações, sendo a mais relevante delas a grande diversidade de pisos existentes e a grande variabilidade de vibrações geradas, tornando assim, inviável a criação de um sistema global indiferente do piso utilizado.

3.5 Detecção de quedas utilizando aplicativo próprio

O artigo de Mauldin et al. [24] apresenta o *SmartFall*, um aplicativo Android que coleta dados de um acelerômetro localizado em um relógio inteligente preso ao pulso da pessoa monitorada. No aplicativo, o processamento necessário ocorre sem a latência de comunicações de longas distâncias, o que acontece em arquiteturas onde o processamento é feito na nuvem. Os experimentos são feitos com algoritmos de aprendizado de máquina tradicionais em detecções de queda (*Support Vector Machine - SVM e Naive Bayes - NB*), e técnicas não convencionais (*Deep Learning*) para criação dos modelos de detecção de quedas utilizando três conjuntos de dados (*Smartwatch*,

Notch, *Farseeing*). O trabalho visa comparar os métodos tradicionais e não-tradicionais para detecção de quedas utilizando aprendizado de máquina. Dentre os três conjuntos de dados existentes, foram criados pelo autor os dados referentes ao conjunto *Smartwatch* e ao conjunto *Notch*, com sensores distribuídos no corpo dos voluntários.

A Figura 3.10 mostra a quantidade de quedas e atividades do dia-a-dia (ADL, do inglês *Activities of Daily Living*) presentes em cada conjunto de dados. Para avaliar os três algoritmos, foram utilizados como medidas de desempenho: sensibilidade, precisão e acurácia. Como o sistema deve fazer detecções em tempo real, não é desejado que o algoritmo erre uma queda, o que implica na necessidade de uma alta sensibilidade. Também não é pretendido que haja uma quantidade alta de falsos alarmes, o que se traduz em uma precisão alta.

	# Falls	# ADLs
Smartwatch	91	90
Notch	107	2456
Farseeing	23	27,412

Figura 3.10: Eventos capturados pelas bases de dados utilizadas [24].

Podemos observar na Figura 3.11 os resultados de cada medidor de desempenho para cada modelo apresentado. Vemos que há um excesso de falsos positivos, pois há uma diferença entre a quantidade de eventos ocorridos relacionados às atividades do dia-a-dia com os eventos de quedas amostrados. Dados estes resultados, visualizamos a grande diferença dos métodos tradicionais para os não-tradicionais, resultando em uma larga vantagem para o uso do *Deep Learning* em todos os conjuntos de dados.

		NB	SVM	Deep
Smartwatch	Precision	0.60	0.68	0.77
	Recall	0.92	0.86	1.0
	ADL Acc.	0.38	0.60	0.70
	Overall Acc.	0.65	0.73	0.85
Notch	Precision	0.58	0.53	0.79
	Recall	0.89	0.80	0.89
	ADL Acc.	0.97	0.97	0.99
	Overall Acc.	0.97	0.96	0.99
Farseeing	Precision	0.005	0.44	0.37
	Recall	0.82	0.55	1.0
	ADL Acc.	0.87	0.99	0.99
	Overall Acc.	0.87	0.99	0.99

Figura 3.11: Resultados encontrados [24].

Em resumo, pode-se observar que o modelo construído utilizando-se de *Deep Learning* obteve

resultados melhores nos parâmetros de precisão e sensibilidade. Isso é explicado pelos autores pelo fato do algoritmo conseguir aprender funcionalidades dos dados crus, diferente de NB e SVM, que utilizam funcionalidades manuais e limitadas, pois devem ser pré-estabelecidas pelo pesquisador antes do processo de treino. Além disso, a capacidade de generalização do modelo de *Deep Learning* é uma qualidade importante, pois permite que novos usuários adotem o sistema.

Apesar do resultado positivo do uso do *Deep Learning*, os experimentos mostraram que o modelo peca na classificação de atividades ADL, principalmente em atividades mais leves como sentar e caminhar. Com alguns ajustes no modelo, os autores acreditam que os resultados de classificação de atividades ADL serão melhorados, podendo chegar aos valores de precisão e sensibilidade adquiridos nos modelos NB e SVM.

3.6 Conclusões

Os trabalhos apresentados variam em diversos pontos: metodologia de testes, utilização de sensores, métodos de processamento, arquitetura, preço de implementação, etc. De acordo com Le et al. [20] há a utilização de dois sensores atrelados ao corpo, situados no tronco e na coxa da pessoa, logo tem-se o risco do idoso não lembrar de usar um ou os dois sensores, o que inviabilizaria a detecção de queda de forma correta. Além disso, o sensor utilizado possui limitação de valores de aceleração a $\pm$8G (oito vezes a aceleração da gravidade).

Outra desvantagem descrita por Le et al. [20] é a arquitetura do sistema, que faz o transporte de informação por meios cabeados para conduzir os dados processados para o computador, o que diminui a mobilidade e versatilidade da disposição dos componentes do sistema. Além disso, a necessidade da ligação direta a um computador para visualização dos dados processados dificulta a implementação do sistema, visto que nem sempre o monitoramento da pessoa poderá ser feito no local.

No trabalho de Litvak et al. [21], podemos ver algumas desvantagens tragas pelo sistema. A primeira delas surge pelo fato da utilização de um sistema não-intrusivo, gerando assim, uma dependência do piso utilizado para uma correta classificação de queda ou não-queda. Podemos frisar também a não utilização de quedas reais e as poucas amostras obtidas pelas quedas do boneco utilizado. Há também a utilização de técnicas de aprendizado de máquinas, que podem criar uma inviabilidade computacional de implementação do sistema.

Observando a tese elaborada por Rodrigues [22], vemos que a maior desvantagem se encontra pela falta de um sistema IoT completo, desde a coleta até a notificação de queda. Em seu sistema são gerados arquivos do tipo CSV com os dados coletados pelo sensor, de modo que seja ainda analisado. Essa desvantagem pode ser ocasionada pelo uso de técnicas de aprendizado de máquina não integradas a um banco de dados presente em um servidor em nuvem. Outro possível empecilho está no alto custo computacional para o uso de aprendizado de máquina para testes e

treinos, haja visto que o sistema precisaria se adequar a cada pessoa que a utilizasse.

Segundo Werner et al. [23], o protótipo de detecção de quedas utiliza métodos não-intrusivos, o que segundo a literatura, não é o método mais confiável de detecção de quedas, devido à elevada quantidade de falsos positivos causados por ruído de objetos ou outras pessoas. A dependência do ambiente onde o sistema é implementado também é um limitante. Os resultados contidos no trabalho de Werner et al. [23] apontam que há uma alta influência do tipo de piso no desempenho do algoritmo. Werner et al. ainda afirmam que se o chão do quarto não for homogêneo, a posição das caixas de sensores pode ter uma influência significativa na taxa de detecção de quedas. Há também a necessidade de aumentar a quantidade de sensores caso a área do recinto esteja acima de 25 m².

Além disso, o projeto depende de várias caixas de sensores operando ao mesmo e à todo tempo, o que dificulta a implementação e manutenção do sistema. Por fim, apesar da utilização de testes com voluntários, nenhuma queda real foi executada, o que impossibilita afirmar que o desempenho do protótipo é bom em cenários onde ocorrem quedas não simuladas.

Revisando o trabalho de Mauldin et al. [24], é identificado que a maior falha do sistema foi a dificuldade do algoritmo de *Deep Learning* conseguir identificar algumas atividades do cotidiano como sentar e caminhar. Para mais, o projeto também depende de um celular estar na proximidade do sujeito e o uso constante do relógio inteligente, o que nem sempre pode ser possível.

A Tabela 3.1 mostra um panorama sobre a comparação entre os trabalhos, sendo utilizados como parâmetros os seguintes itens: abordagem, baixo custo financeiro, uso de sistema IoT, baixa complexidade computacional (BCC) e utilização de múltiplos sensores (MS).

Tabela 3.1: Comparação de Trabalhos

Abordagem	**Tipo**	**Baixo custo**	**IoT**	**BCC**	**MS**
[20]	Intrusivo	X		X	X
[21]	Não-intrusivo	X			X
[22]	Intrusivo	X	X		X
[23]	Não-intrusivo				X
[24]	Intrusivo	X	X		

Os trabalhos apresentados aqui não são capazes de suprir todos os seguintes pontos:

- Uma solução IoT completa, desde a coleta dos dados à notificação ao usuário final.
- A não utilização de métodos e técnicas de alta complexidade computacional tais como: técnicas de aprendizado de máquina ou processamento de imagem.
- Baixo custo de implantação ou de dispositivos essenciais para o funcionamento adequado do sistema.

- Facilidade na notificação da queda, utilizando meios de comunicação solidificados no mercado, evitando assim, a necessidade de preparação do idoso e a manutenção requerida por um sistema de notificação próprio.

Esta abordagem tem o intuito de apresentar uma solução para detecção de quedas de baixo custo financeiro e computacional, bem como proporcionar uma solução IoT completa através de métodos intrusivos, de modo a obter uma confiabilidade considerável. Os detalhes desse sistema serão apresentados nos capítulos seguintes.

Capítulo 4

Metodologia e Implementação

Este capítulo apresenta a metodologia utilizada no presente trabalho, detalhando a solução de detecção de quedas e mostrando os processos do sistema e arquitetura montada.

4.1 Apresentação do Sistema Desenvolvido

O sistema desenvolvido possui a arquitetura ilustrada na Figura 4.1, sendo distribuída em diversas partes, onde cada camada possui função própria e distinta, conforme apresentado na Figura 2.1. Inicialmente, temos a coleta de dados, explanada na Seção 4.1.1 e o envio de dados na Seção 4.1.2. Posteriormente, a Seção 4.1.3 demonstra o armazenamento do sistema, seguido pela Seção 4.1.4, que apresenta os métodos de processamento dos dados coletados. Finalmente, tem-se a interação do sistema com o usuário, apresentado na Seção 4.1.5. A Figura 4.2 resume o processo todo em forma de fluxograma.

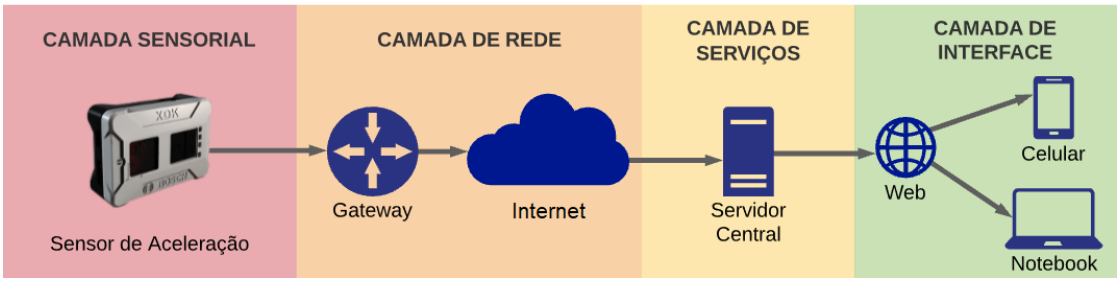


Figura 4.1: Arquitetura do sistema

4.1.1 Coleta de dados

Em consentimento com a literatura existente, vimos a grande importância dos sensores em um sistema para a identificação de quedas. Dentre os utilizados se destaca o uso de acelerômetros, no

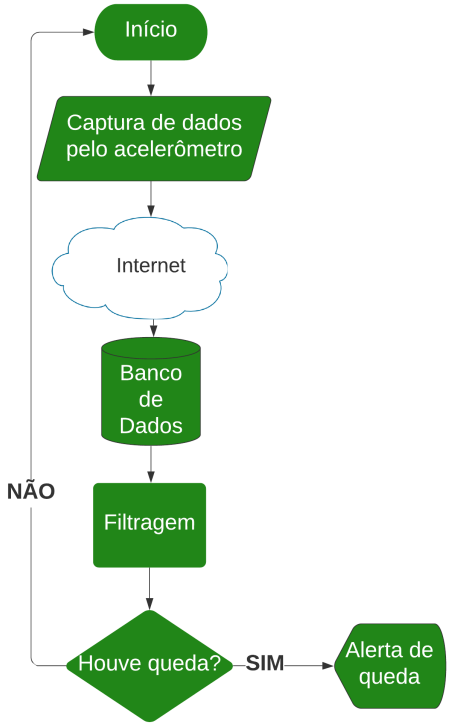


Figura 4.2: Diagrama de fluxo do sistema

qual mensura a aceleração em objetos e, quando presos ao corpo humano, são capazes de detectar movimentos.

No sistema elaborado foi utilizado o *kit* XDK da Bosch [25], que contém 8 sensores integrados, além de uma plataforma de desenvolvimento de *software* que utiliza a linguagem C, bastante usada pela liberdade que proporciona ao desenvolvedor. Os sensores presentes no *kit* XDK estão apresentados na Figura 4.3, e são eles: acelerômetro, giroscópio, magnetômetro, sensor de luz, sensores ambientais (compreende sensor de temperatura, pressão atmosférica e umidade) e o sensor acústico.

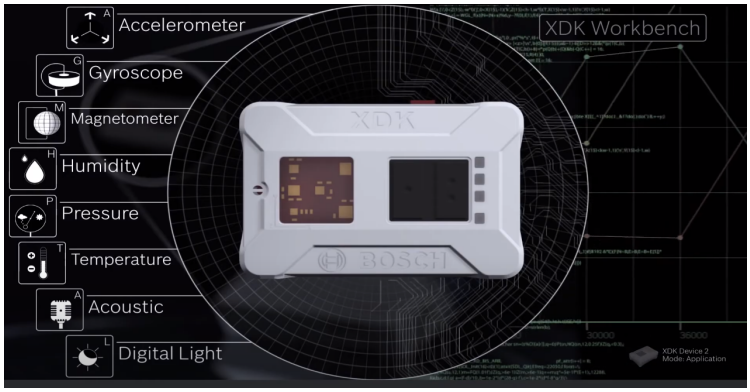


Figura 4.3: Foto da XDK e os sensores presentes [26].

Entre os sensores existentes, o escolhido para detecção de quedas foi o sensor de aceleração. Na XDK temos o BMA280 [27], um acelerômetro digital de 16 bits, definido para três eixos de direção ortogonais, com interrupções controláveis através de pinos. Sendo um acelerômetro de três eixos, o BMA280 possui a capacidade de obter o valor da aceleração, em m/s², na direção de cada eixo X, Y e Z. O acelerômetro funciona como um sistema massa e mola, medindo a deformação da mola, que consequentemente, refere-se à aceleração ocorrida. Os acelerômetros do tipo MEMS (do inglês *Micro-Electro-Mechanical Systems*), como o utilizado na XDK, utilizam uma massa de silício com 4 âncoras conectadas em suas extremidades, na qual possui uma placa ligada a dois capacitores carregados igualmente. Ao haver uma movimentação na massa, essa placa se movimenta, gerando assim, uma diferença nas capacitâncias, resultando no valor da aceleração ocorrida. Podemos ver esse funcionamento na Figura 4.4. Apesar da utilização de um *kit* específico, é possível utilizar outros dispositivos com sensores de aceleração.

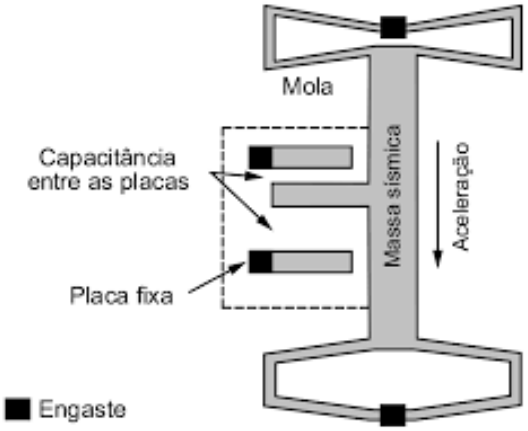


Figura 4.4: Funcionamento de acelerômetros do tipo MEMS [28].

O sistema operacional presente na XDK é o FreeRTOS, que é um sistema operacional em tempo real (do inglês RTOS - *Real-Time Operating System*). Estes sistemas são caracterizados por servir aplicações que fazem o processamento de dados a medida que eles chegam, tentando minimizar ao máximo os atrasos de sistema operacional, visto que as aplicações normalmente têm exigências de tempo de resposta bem definidas (como é o caso do projeto). De acordo com Barry [29], FreeRTOS é um *kernel* de processamento em tempo real onde aplicações podem ser construídas para garantir os requerimentos de tempo mais rígidos. O sistema operacional se destaca pela sua extensa compatibilidade com diversas arquiteturas de processadores além da priorização pela eficiência de processamento e economia de energia.

Foi decidido através de testes que a taxa de amostragem mais estável para a coleta de dados no sistema é de 20 Hz, ou seja, há 20 capturas de dados em cada eixo por segundo, resultando assim, em uma amostra a cada 50 ms em cada eixo do acelerômetro. Uma taxa maior que 20 Hz provocou disrupção do sistema dada a limitação de processamento do micro-controlador.

A captura dos dados é efetuada através de interrupções definidas por tempo, no qual a cada 50

ms o código é interrompido, chamando a função presente no Anexo A, onde é capturado o valor da aceleração nos três eixos e salvo em um vetor, sendo que na vigésima vez (quando o código entra no "*if*" da função), é iniciada uma comunicação TCP para envio dos dados via HTTP, sendo salvo também o horário através do protocolo NTP. O horário foi coletado com este protocolo de rede porque o microcontrolador não tem medição de tempo, fazendo com que o tempo correto seja coletado a partir de uma fonte externa. A informação do tempo certo é bastante importante para a aplicação pois o sistema deve informar corretamente o tempo que a queda aconteceu.

O protocolo NTP utiliza um sistema hierárquico e em camadas de troca de mensagens, como mostra a Figura 4.5, onde no ponto mais alto da hierarquia temos os servidores de referência primária, que possuem o horário mais preciso, (normalmente relógios atômicos). Logo abaixo, temos os servidores de referência secundária, que sincronizam seus relógios atuando como clientes dos servidores primários. Nas subsequentes camadas, os servidores sincronizam seus relógios com os servidores da camada acima, e servem o tempo para os dispositivos das camadas de baixo. Logo, todos os servidores atuam como servidor e cliente em simultâneo, em uma coordenação que faz com que nenhum dispositivo fique sobrecarregado no serviço de entrega de tempo.

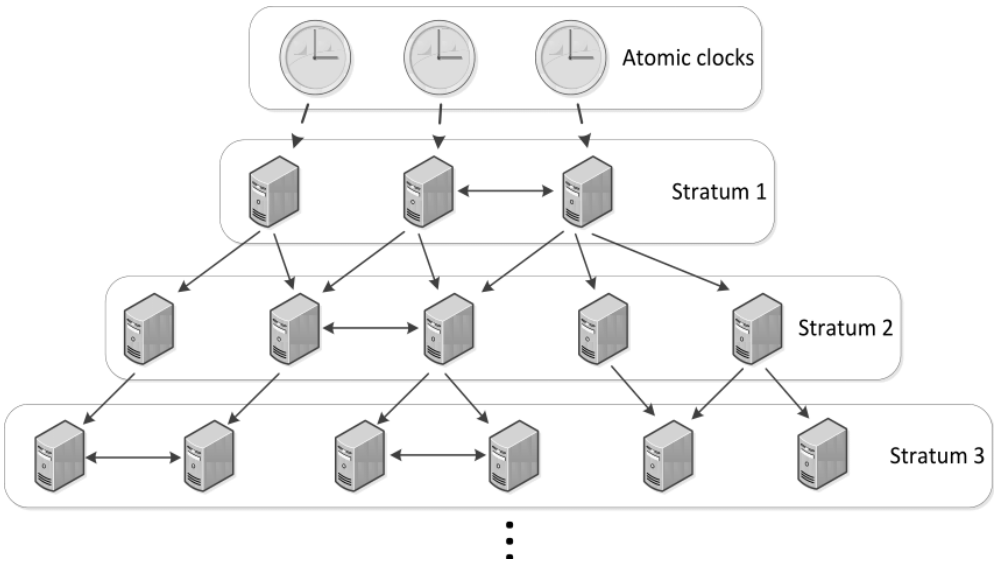


Figura 4.5: Arquitetura de servidores NTP [30].

4.1.2 Envio dos dados

O sistema de envio é subdividido em duas partes: o envio do micro-controlador XDK para o *gateway* local e o envio dos dados desse *gateway* para a nuvem, de modo a armazenar os dados e executar o processamento adequado. Todo o transporte de dados utiliza-se de tecnologia Wi-Fi.

4.1.2.1 Envio dos Dados da XDK para o Gateway

O envio de dados a partir do XDK consiste na destinação de uma mensagem em formato JSON, através do protocolo HTTP ao *gateway* local. O JSON é uma notação de objeto com intuito de ser legível por humanos e fácil para a análise e uso de dispositivos. O protocolo HTTP é definido por Fielding et al. [31], como um protocolo do tipo requisição/resposta que opera com a troca de mensagens por uma sessão. A mensagem em formato JSON consiste em três vetores referentes aos eixos do acelerômetro, pois, por uma decisão de viabilidade de processamento físico e otimização de rede, há o envio de uma mensagem do tipo JSON por segundo ao *gateway*, contendo assim, as 20 amostras captadas neste intervalo, como mostra a Figura 4.6.

Além destes dados, a mensagem possui um campo chamado *Timestamp*, que contém a data e a hora do último dado capturado naquele intervalo. A data e hora são obtidas pelo uso do protocolo NTP. No projeto, a XDK atua como cliente e sincroniza seu relógio com o *gateway*, que atua como servidor NTP. O *gateway* sincroniza seu relógio com um conjunto de servidores públicos. A arquitetura final se assemelha bastante ao exposto na Figura 4.5. Um exemplo de uma mensagem JSON enviada é mostrado na Figura 4.6.


```
{
    "xAceleration": [ 5.00, 6.2, 20.4, 34.1, 42.1, 5.00, 6.2, 20.4, 34.1, 42.1, 5.00, 6.2, 20.4, 34.1, 42.1, 5.00, 6.2, 20.4, 34.1, 42.1],
    "yAceleration": [ 4.00, 16.2, 40.4, 35.1, 46.1, 4.00, 16.2, 40.4, 35.1, 46.1, 4.00, 16.2, 40.4, 35.1, 46.1, 4.00, 16.2, 40.4, 35.1, 46.1],
    "zAceleration": [ 2.00, 26.2, 22.4, 32.1, 45.1, 2.00, 26.2, 22.4, 32.1, 45.1, 2.00, 26.2, 22.4, 32.1, 45.1, 2.00, 26.2, 22.4, 32.1, 45.1],
    "Timestamp": "14/04/2021 17:41:40"
}
```


Figura 4.6: Corpo da mensagem JSON enviada.

4.1.2.2 Envio de dados do *gateway* para a nuvem

No sistema desenvolvido, foi utilizada uma figura central denominada de *gateway* que consiste em um coordenador de requisições para mediar o acesso à nuvem. Esta topologia visa o crescimento em escala do sistema, permitindo o acréscimo de vários sensores. Tal prerrogativa é plausível, haja visto que há a possibilidade de diversas pessoas utilizarem o sistema com sensores distintos, tornando assim, a necessidade de uma figura que coordena o sistema visando uma quantidade de requisições uniforme.

A utilização de um *gateway* local facilita a criação de um sistema onde os micro-controladores enviam requisições HTTP localmente, e o *gateway* comunica-se com a nuvem através do protocolo HTTPS, criando assim, uma conexão menos vulnerável. Além disso, os micro-controladores tornam-se menos ocupados com a transmissão de dados, diminuindo assim o atraso do sistema no geral. Para a função de *gateway* local, foi utilizada uma máquina virtual Linux, prezando pelo baixo custo de implementação.

O RabbitMQ, presente no *gateway*, foi um dos responsáveis por esta comunicação entre a XDK e a nuvem, utilizando o protocolo AMQP (do inglês *Advanced Message Queuing Protocol*).

Ele é um *software* que atua como intermediário entre dois ou mais processos que se comunicam. Nas formas tradicionais de comunicação, como ilustra a Figura 4.7, temos um diálogo síncrono, onde um processo envia uma mensagem e o outro processo fica esperando o recebimento da mensagem, para continuar suas tarefas ou continuar a comunicação enviando sua resposta no caminho contrário. O problema dessa arquitetura é a dificuldade de escalabilidade e necessidade de interrupção dos dois processos que devem estar focados na troca de informações.

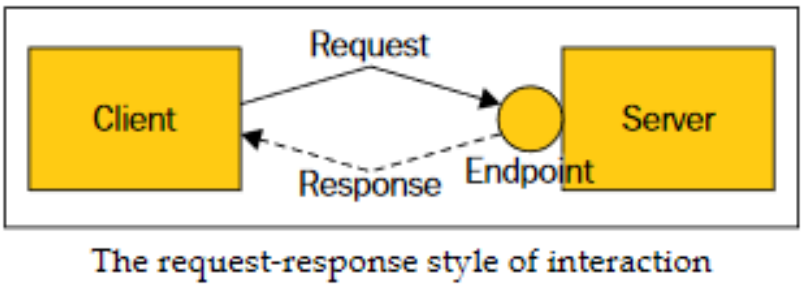


Figura 4.7: Arquitetura clássica de troca de mensagens [32].

Neste cenário, surge a ideia de uma fila de mensagens, que conforme a Amazon [33], é uma forma de comunicação assíncrona serviço-a-serviço, sendo muito utilizada em arquiteturas de microsserviços. Neste sistema, como ilustra a Figura 4.8, a interação é de via única, onde o serviço Publicador envia a mensagem para uma fila, que é mantida pelo RabbitMQ e o Consumidor recebe a mensagem por meio do intermediador. Deste modo, os dois serviços (Publicador e Consumidor) não precisam interrromper suas tarefas para dar atenção a comunicação, e o intermediário fica encarregado de garantir a entrega das mensagens.

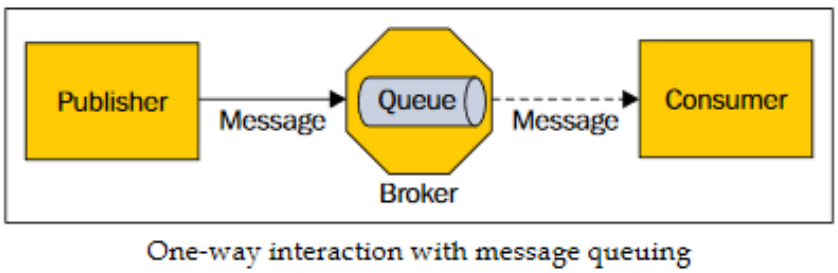


Figura 4.8: Arquitetura de via única usando *message queuing* [32].

4.1.3 Armazenamento dos dados

Após receber os dados do acelerômetro, o servidor central salva cada leitura do acelerômetro em uma tabela do banco de dados. A ferramenta utilizada foi o PostgreSQL que consiste em um sistema de gerenciamento de banco de dados de código aberto do tipo relacional. Bancos

de dados do tipo relacional armazenam os dados de forma estruturada e a disposição dos objetos é semelhante à de uma planilha ou tabela, que estabelece as "relações" (por isso o nome) entre as colunas, que representam atributos de cada objeto, e linhas da tabela, que representam cada objeto.

Na tabela relacional utilizada no projeto, cada linha da tabela representa uma coleta do acelerômetro em um instante de tempo. A tabela deve possuir colunas que identificam os atributos de cada objeto armazenado e linhas que especificam cada objeto. A Figura 4.9 mostra as colunas definidas. Na tabela utilizada, as colunas definidas foram:

- `id`: Identifica unicamente uma linha na tabela, sendo utilizado posteriormente no processamento de dados, ou seja, chave primaria da tabela relacional.
- `Timestamp`: Guarda o horário da leitura do acelerômetro em formato *Unix Epoch Time*, que representa o número de segundos entre a data e hora atual e 01/01/1970 às 00:00:00. Por exemplo: O *Unix Epoch Time* 1609459200.00 se traduz para a data 01/01/2021 à meia-noite. Devido à necessidade de coletar várias amostras por segundo, este atributo foi escolhido como *Float*, possuindo uma parte fracionária que representa os milissegundos da coleta.
- `xAceleration`: Armazena a componente do eixo X de aceleração em m/s².
- `yAceleration`: Armazena a componente do eixo Y de aceleração em m/s².
- `zAceleration`: Armazena a componente do eixo Z de aceleração em m/s².

id	timestamp	xAceleration	yAceleration	zAceleration
1	1618867657	7.887	5.1008	5.6333
2	1618867657.05	6.5099	8.4598	9.3724
3	1618867657.1	2.9809	6.5539	0.2756
4	1618867657.1499999	5.4782	4.2949	6.8167
5	1618867657.1999998	5.2509	9.3769	3.1784

Figura 4.9: Tabela no banco de dados.

Houve também a necessidade da utilização de um banco em estrutura de memória para armazenamento de uma variável necessária para o processamento de dados. Para isso, foi utilizado o Redis [34] que é um banco de dados do tipo par chave-valor, onde cada informação armazenada possui uma chave de referência única, e o seu valor. Ao efetuar qualquer operação sobre os dados, o cliente informa a chave de referência do objeto a ser manipulado.

4.1.4 Processamento de dados

O processamento de dados é composto de dois passos: filtragem dos sinais para frequências altas; e detecção de queda pelo critério de limiar. Este passo começa com a inserção dos dados advindos da etapa anterior e termina com a decisão binária referente à "queda" ou "não-queda".

4.1.4.1 Primeiro passo: filtragem dos sinais

Uma queda é interpretada pelo sensor de aceleração como uma variação de grande amplitude em um ou mais eixos em um curto espaço de tempo, conforme mostra a Figura 4.10. O tombo teve início aproximadamente no instante 16:49:37, sendo finalizado no instante 16:49:40, com o usuário permanecendo deitado após a queda. No domínio espectral, tem-se a queda localizada em uma frequência mais elevada em relação ao resto do sinal, devido ao seu comportamento de mudança repentina em um instante de tempo.

Figura 4.10: Sinais da aceleração em função do tempo para o evento "queda para trás" (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

No primeiro passo do processamento, o objetivo é que os três sinais possuam apenas o registro das quedas ocorridas para efeito de detecção. Porém, o sensor de aceleração capta todas as atividades realizadas pelo usuário, e estas ações geram um sinal ruidoso. Logo o sistema deve realizar a filtragem dos sinais para retirada do ruído.

Para a filtragem, optou-se pelo cálculo da Transformada Wavelet, que decompõe o sinal truncado até o nível 2, resultando nos coeficientes $cA2$ (coeficientes aproximados de nível 2), $cD2$ (coeficientes detalhados de nível 2) e $cD1$ (coeficientes detalhados de nível 1). Como uma queda

tem a característica de possuir frequências altas, após a obtenção dos coeficientes, $cA2$ e $cD2$ são multiplicados por zero por guardarem as componentes de baixas frequências. Logo, sobram apenas as componentes de altas frequências em $cD1$. A Transformada Wavelet Inversa é calculada utilizando os coeficientes $cA2$ e $cD2$ multiplicados por zero, e $cD1$ que tem seu valor inalterado. Desta forma, o sinal reconstruído possui apenas o registro das quedas em potencial. Como o trabalho pretende fazer uma classificação binária respondendo ao usuário se houve queda ou não, sem qualquer detalhamento da ação que a pessoa estava fazendo, não foi necessária a decomposição do sinal em mais de 2 níveis, visto que as quedas se concentram em $cD1$ e o resto é interpretado como "não-queda".

A filtragem é feita nos três sinais individualmente, cada um truncado em 80 amostras discretas, devido à Transformada Wavelet necessitar de um sinal com tamanho divisível por 4 pela decomposição ser até o segundo nível. A Figura 4.11 mostra o antes e o depois de um sinal de aceleração do Eixo Z. O pico presente nos dois sinais representa uma queda realizada. Além da queda, vários eventos foram executados, como "deitar", "sentar", "andar". Porém, após o processo de filtragem, estes eventos são minimizados.

O anexo B apresenta as funções que realizam os cálculos de Transformada Wavelet Direta e Transformada Wavelet Inversa, além da função que chama às duas anteriores.

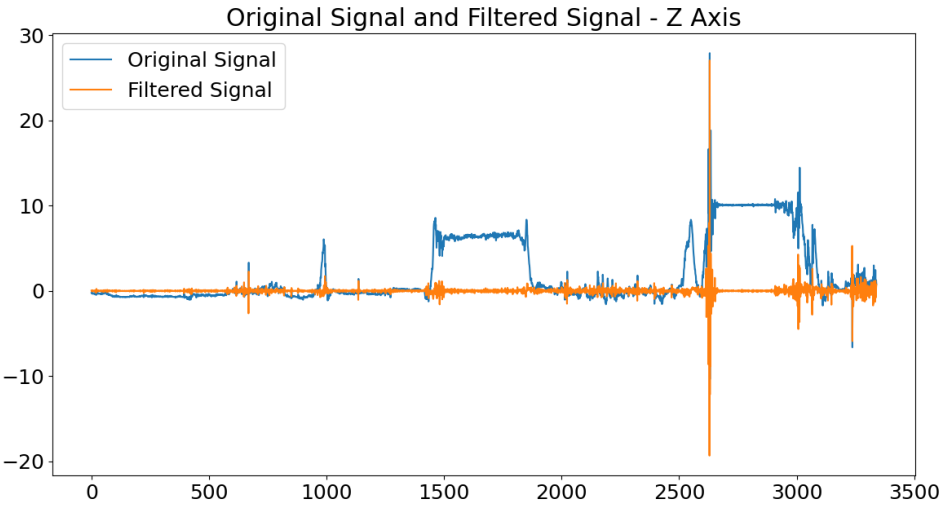


Figura 4.11: Comparação de um sinal de aceleração antes e depois do filtro. Este evento foi caracterizado pelo sujeito de teste iniciar em pé. No instante igual a 1500, no eixo das abscissas, o sujeito de teste deita, permanecendo assim, até próximo do valor 2000, quando se levanta. Já no instante próximo a 2500 ocorre uma queda, sendo que o sujeito permanece deitado até o instante 3000, onde se levanta. As outras variações presentes nos gráficos correspondem ao ruído.

4.1.4.2 Segundo passo: Decisão de queda por meio de limiar

Após os sinais serem filtrados, o sistema se depara com picos que representam potenciais momentos em que houve quedas. Para que haja uma decisão do que será interpretado como queda, é estabelecido um limiar de 15 m/s² para o eixo Z e um limiar de 4 m/s² para os eixos X e Y. O limiar para o eixo Z é maior, pois o posicionamento escolhido do micro-controlador faz com que seu eixo Z apresente acelerações maiores em situações de queda, enquanto os outros dois eixos variam menos em tais acidentes. Estes valores foram definidos através de observações e ajustes. Os limiares foram definidos antes da execução dos testes mostrados no Capítulo 5. O sistema julga como queda quando há ultrapassagem de um ou mais limiares dos três eixos.

Na Figura 4.12, são mostrados os picos detectados pela solução. O sinal analisado é o ilustrado na Figura 4.11, que representa as leituras de aceleração do eixo Z. A linha pontilhada indica que o limiar utilizado para decisão de queda é de 15 m/s², que corresponde ao valor estabelecido para os sinais do eixo Z.

No anexo B é exposta a função do programa que identifica os picos encontrados no sinal truncado. Também é passada a função que chama a detecção de quedas para os três sinais (X, Y, Z).

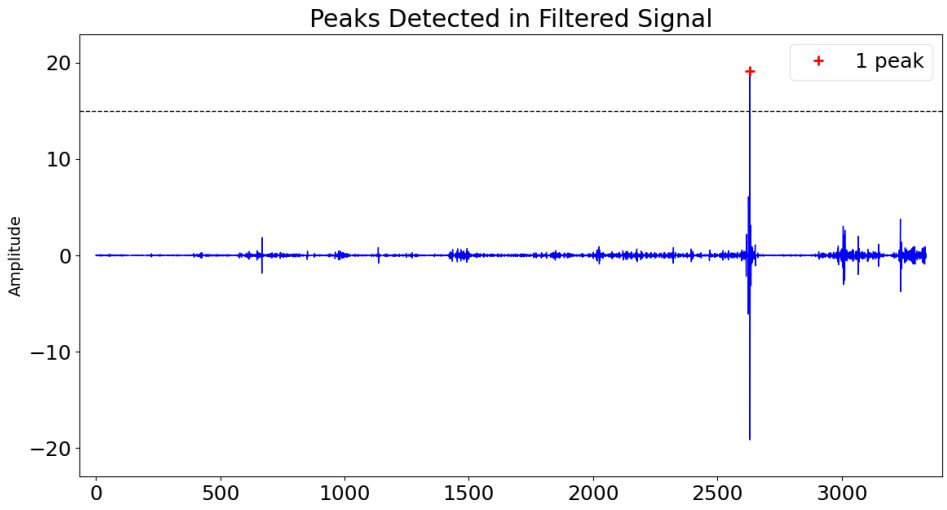


Figura 4.12: Exemplo de sinal filtrado com a presença de limiar e detecção de pico. No eixo das abscissas temos o id de cada captura no banco de dados e no eixo das ordenadas temos a aceleração em m/s².

4.1.5 Alerta de Queda

Finalmente, após a detecção de um evento de queda, o sistema precisa alertar alguém do acidente, logo é mandado o horário da queda à uma fila do RabbitMQ, que atua como intermediário entre o processo que detectou a queda e o processo que enviará a notificação da queda. Este retira o horário da queda da fila e envia uma mensagem à um canal privado do Telegram, contendo o horário exato da queda e o identificador da pessoa que sofreu o acidente. Foi utilizado o Telegram devido à facilidade de integração do *software* a aplicações existentes e ao fato da comunicação ser instantânea, diminuindo o atraso de notificação. Usuários do aplicativo que estão inscritos no canal recebem a notificação na hora. Exemplos de notificação são mostrados na Figura 4.13.

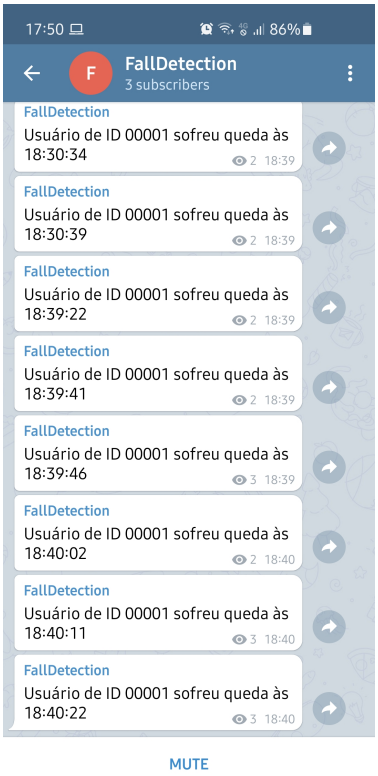


Figura 4.13: Exemplo de notificação em um canal no Telegram.

4.1.5.1 Funcionamento de um canal do Telegram

Um canal no Telegram é um tipo de conversa onde se tem comunicação de via única entre os administradores do canal e os inscritos. Nessas conversas, os administradores enviam mensagens e os inscritos podem apenas ler estas mensagens. Qualquer conta do Telegram pode se tornar administradora do canal, incluindo robôs (do inglês *bots*), que são contas do Telegram que participam de conversas de forma automatizada, enviando mensagens sem necessidade de intervenção humana.

No Telegram, um canal pode ser público ou privado. Se for público, qualquer conta pode entrar através de um *link*. Se for privado, apenas os administradores do canal podem adicionar usuários ao canal. Optou-se por criar um canal privado, pois as informações compartilhadas na conversa são sensíveis e não devem ser acessadas por qualquer usuário.

4.1.5.2 Bots do Telegram

Os *bots* do Telegram são um tipo especial de conta que servem para automatização de mensagens, normalmente usados em sistemas que necessitam enviar ou tratar mensagens enviadas sem intervenção humana. O Telegram disponibiliza uma API para criação de *bots* e interação com outros usuários através do mesmo. Com esse *bot*, é possível enviar mensagens automáticas por linhas de código em linguagem Python. Cada *bot* possui um único identificador chamado *Token*.

Para comunicar com a API do Telegram, toda requisição deve ser do tipo HTTPS e precisa ter como destino `https://api.telegram.org/bot<token>/METHOD_NAME`, onde o parâmetro "*token*" é o identificador único do *bot* e o "*METHOD_NAME*" é o nome da ação que o robô de mensagens deve realizar. Um exemplo de "*METHOD_NAME*" é *sendMessage* que envia uma mensagem e recebe como entradas obrigatórias *text*, o texto da mensagem, e *chat_id*, que identifica a conversa onde será enviada a mensagem.

Para utilizar um *bot*, é necessário criá-lo requisitando ao *Bot Father*, um robô de mensagens que tem a finalidade única de criar e gerenciar os *bots* criados através de comandos simples disponibilizados pelo Telegram. Após algumas definições do *bot* a ser feito, o *Bot Father* retorna o *Token* do *bot* criado. Com esse *token*, é possível enviar mensagens em nome do *bot*.

Após criação do *bot*, é necessário adicionar o mesmo como administrador do canal. No anexo D, é incluído o código que faz o *bot* enviar as mensagens para o canal. Primeiro, é construído um objeto Telegram do tipo "*Bot*" que recebe o *token* obtido anteriormente como parâmetro. Em seguida, na função *send*, são passados como parâmetros o *chat_id*, que corresponde ao identificador da conversa em um canal, e a mensagem que contém o horário da queda.

4.2 Detalhamento da arquitetura

A arquitetura em 4.1 representa o sistema de forma geral, porém o *gateway* local, a nuvem AWS e principalmente o servidor central possui muitas partes dentro de si, necessitando um detalhamento maior de seus elementos.

4.2.1 *Gateway* local

O sistema, como mencionado anteriormente, utiliza um *gateway* local como coordenador e concentrador de requisições. O *gateway* local implementado durante o trabalho consistiu de uma máquina virtual rodando o popular sistema operacional Ubuntu Linux. Na Figura 4.15, observamos o detalhamento do seu funcionamento na totalidade. O alicerce principal da arquitetura é o Docker. Docker é um "motor" que permite um grau de virtualização de sistema operacional por meio da criação de *containers*, que estão logicamente isolados do S.O hospedeiro na questão de utilização de recursos físicos como espaço em disco, CPU, memória. Este isolamento é similar ao obtido com a virtualização tradicional, onde se tem vários sistemas operacionais sendo executados em uma mesma máquina física através de um programa virtualizador.

O diferencial do Docker é que os *containers* são extremamente mais leves (em questão de utilização de recursos) do que os sistemas operacionais rodando em um virtualizador, pois utilizam de recursos do núcleo do sistema operacional hospedeiro, que está instalado fisicamente, enquanto as máquinas virtuais possuem seus próprios núcleos, o que aumenta o consumo de recursos físicos a cada máquina virtual sendo executada.

Segundo Merkel [35], para resumir, *containers* virtualizam no nível do sistema operacional enquanto programas de virtualização virtualizam no nível físico apenas. A Figura 4.14 mostra as arquiteturas das VMs e dos *containers* respectivamente.

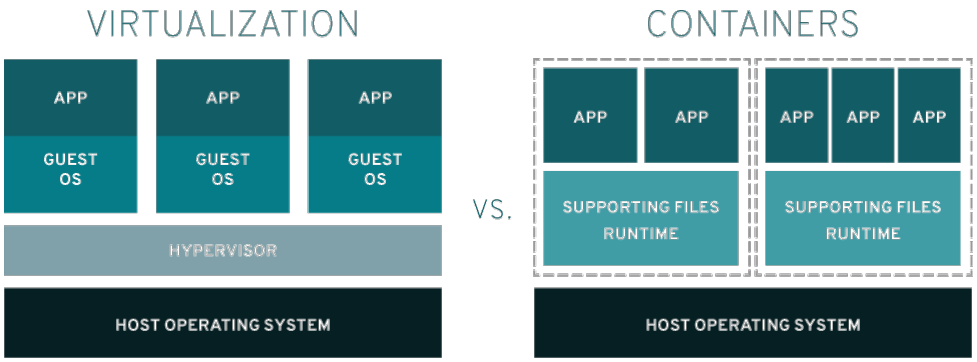


Figura 4.14: Diferença entre máquinas virtuais e containers [36].

Normalmente cada *container* executa uma função em um sistema distribuído, e o Sistema Operacional hospedeiro não precisa executar estas funções, necessitando apenas de executar o *Docker Engine*. Além disso, outra característica fundamental do Docker é sua portabilidade, pois as compatibilidades necessárias para executar a aplicação se resumem ao próprio *Docker Engine*, não sendo necessário verificar se o Sistema Operacional é compatível com Banco de Dados X ou aplicação Y, visto que estes programas serão sempre executados em *containers*.

O uso do Docker também possibilita que o servidor utilizado esteja localmente, para testes, ou na nuvem com a utilização de VPS (do inglês *Virtual Private Servers*), que são servidores

virtualizados que possuem endereçamento IP público, sendo acessíveis pela Internet.

A arquitetura do *gateway* ilustrada na Figura 4.15 pode ser dividida em dois serviços e quatro instâncias, sendo todas inseridas em *containers* com o uso do Docker. Seus serviços podem ser divididos da seguinte maneira:

1. **Envio de dados do acelerômetro**.
   (a) ***Receiver*** - Aplicação atuando como servidor HTTP recebendo as requisições advindas da XDK.
   (b) **RabbitMQ** - Funciona como um intermediador implementando uma fila de mensagens, possibilitando a XDK não esperar a resposta direta da nuvem. Foi utilizado o *software* RabbitMQ.
   (c) ***Sender*** - Aplicação em Python, devido à facilidade proporcionada pela linguagem, e responsável pelo envio dos dados para a nuvem. Os dados são retirados da fila do RabbitMQ.
2. **Sincronização de tempo**.
   (a) **NTP *server*** - Servidor NTP sincronizado com o conjunto dos servidores públicos e que provê o tempo correto à XDK.

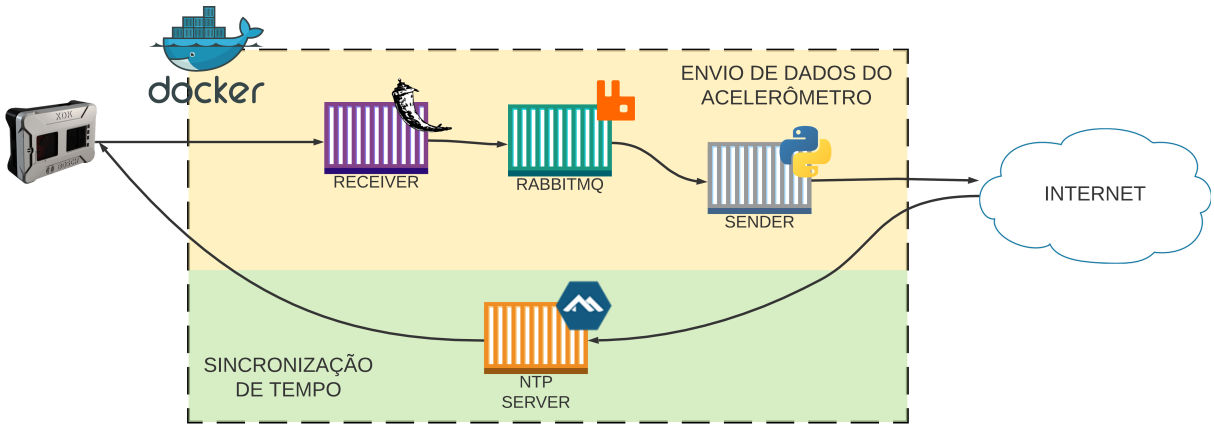


Figura 4.15: Arquitetura do *gateway* local.

4.2.2 Arquitetura na Nuvem

A computação em nuvem adquiriu este nome como uma metáfora para a Internet, que normalmente é representada em topologias e diagramas de rede como uma nuvem que representa "todas as outras coisas" que fazem a rede funcionar. Este modo de exibição também revela que o que está na nuvem é de responsabilidade administrativa de outra pessoa ou órgão.

Segundo Velte et al. [37], a computação em nuvem é uma infraestrutura que permite que o usuário acesse aplicações hospedadas em localidades diferentes da sua, normalmente em centros de processamento de dados (do inglês *datacenters*) distantes. As vantagens de se delegar a hospedagem de aplicações a terceiros englobam a redução de custos operacionais e de capital como:

- O custo de aquisição de servidores é terceirizado.
- Atualizações de *software* e sistemas operacionais são responsabilidade da empresa que hospeda a aplicação.
- Costuma-se pagar menos em computação em nuvem, devido às várias aplicações rodando simultaneamente no mesmo servidor.
- Custos físicos, como eletricidade e resfriamento de salas de servidores são incumbidos pelas empresas prestadoras de serviço de computação em nuvem.
- Usuários podem acessar a aplicação de qualquer lugar do mundo, basta apenas que estejam conectados a *Internet*.

A arquitetura montada na nuvem consiste em vários componentes conforme ilustra 4.16. O servidor central é uma instância EC2 (do inglês *Elastic Compute Cloud*, é um serviço da AWS que permite a criação de máquinas virtuais na nuvem), que funciona como um computador virtual, localizada em uma sub rede pública, o que garante que a VM tenha um IP público, e seja acessível pela Internet. Esta sub rede pública está associada a uma VPC (do inglês *Virtual Private Cloud*, é um serviço da AWS que define todas as configurações de rede) que toma o bloco CIDR 172.31.0.0/16. O *Internet Gateway* é um componente que permite a comunicação entre o VPC e a Internet.

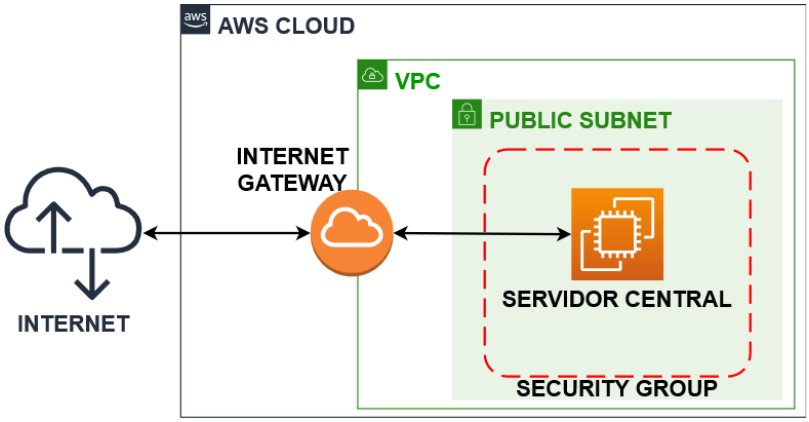
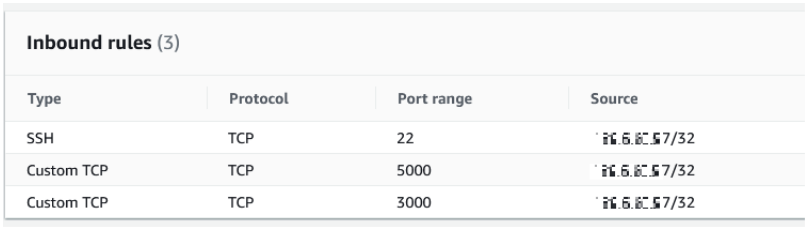


Figura 4.16: Arquitetura na Nuvem.

Temos ainda o *Security Group* que funciona como um *firewall* e libera apenas o tráfego necessário para funcionamento do sistema. As regras do *Security Group* configurado são mostradas

na Figura 4.17. Foi liberada a comunicação de entrada para o servidor (do inglês *Inbound*) nas portas TCP 22 (SSH para gerenciamento remoto do servidor), TCP 3000 (porta utiliza pela interface *web* do Grafana e TCP 5000 (porta utilizada pelo NGINX para recebimento dos dados do acelerômetro). Também foi limitado o IP de origem da comunicação para apenas um endereço utilizando a notação CIDR x.x.x.x/32. Para comunicações de saída do servidor para a Internet (do inglês *Outbound*) todo o tráfego foi liberado.



Inbound rules (3)

Type	Protocol	Port range	Source
SSH	TCP	22	[redacted]/32
Custom TCP	TCP	5000	[redacted]/32
Custom TCP	TCP	3000	[redacted]/32

Figura 4.17: Regras de comunicação *inbound* no *security group*.

4.2.3 Servidor Central

O servidor central está envolvido principalmente nas etapas de processamento de dados e alerta de quedas. Além disso, o servidor recebe as requisições diretamente do *gateway*, e faz o armazenamento das leituras periódicas do acelerômetro. Também é possível visualizar gráficos gerados pelo servidor. Na Figura 4.18, observamos o detalhamento do seu funcionamento na totalidade e as relações entre as partes. Seus serviços podem ser divididos da seguinte maneira.

1. ***Frontend***.
   (a) **NGINX** - Tem a função de receber requisições da Internet, por meio do protocolo HTTPS e repassar por HTTP o pacote para o servidor HTTP. Deste modo, este *container* funciona como *proxy* reverso. Um *proxy* reverso pode ser definido como um servidor intermediário que é acessível pela Internet, tomando o lugar do servidor da aplicação, que fica localizado na *intranet*, se comunicando apenas com o *proxy*. O *software* utilizado foi o NGINX, que é um *software* de código aberto que atua como *proxy* de protocolos da camada de aplicação como HTTP, SMTP, IMAP.
   (b) **Servidor HTTP** - Servidor HTTP que recebe as requisições *GET* e *POST* do NGINX, além de acessar e manipular o banco de dados PostgreSQL.
2. **Bancos de dados**.
   (a) **PostgreSQL** - Este *container* roda o programa PostgreSQL, que cria um banco de dados para armazenamento dos dados advindos do acelerômetro.

(b) **Redis** - O *container* é responsável por salvar a variável *lastId*, que serve como referência para o truncamento do sinal na etapa do processamento. Foi utilizado o banco de dados Redis.

3. **Processamento**.

   (a) **Wavelet-Transform** - *Container* responsável pela filtragem dos dados dos sinais advindos dos três eixos do acelerômetro, bem como a tomada de decisão sobre a ocorrência, ou não, de uma queda. (Conforme detalhado em (4.1.4))

   (b) **RabbitMQ** - Recebe o horário da queda do *container* responsável pela Transformada Wavelet e coloca em uma fila.

4. **Alertas**.

   (a) **RabbitMQ** - Tira o horário da queda da fila e envia para o *container* Telegram.

   (b) **Telegram** - Recebe o horário da queda e gera uma notificação em um canal do Telegram através de um *bot* configurado. (Conforme detalhado em (4.1.5))

5. ***Analytics***.

   (a) **Grafana** - Este módulo tem a funcionalidade de receber os dados de acelerômetro salvos no PostgreSQL e apresentar gráficos mostrando os sinais de aceleração dos três eixos em tempo real. Foi utilizado o *software* Grafana, que é uma ferramenta de código aberto para monitoramento e análise de dados em tempo real. A presença deste módulo serviu apenas para auxílio na formulação do projeto. Um exemplo de gráfico gerado na ferramenta é mostrado na Figura 4.10.

Pensando em um sistema com baixo custo computacional e de fácil adoção, o servidor central foi limitado a ter 1 GB de memória RAM, 1 processador e 8 GB de armazenamento de disco SSD. Além disso, o sistema operacional utilizado foi o Amazon Linux 2, que possui integração otimizada com o serviço AWS EC2, e faz com que os recursos de *hardware* sejam consumidos da forma eficaz, diminuindo as demandas de memória RAM e CPU.

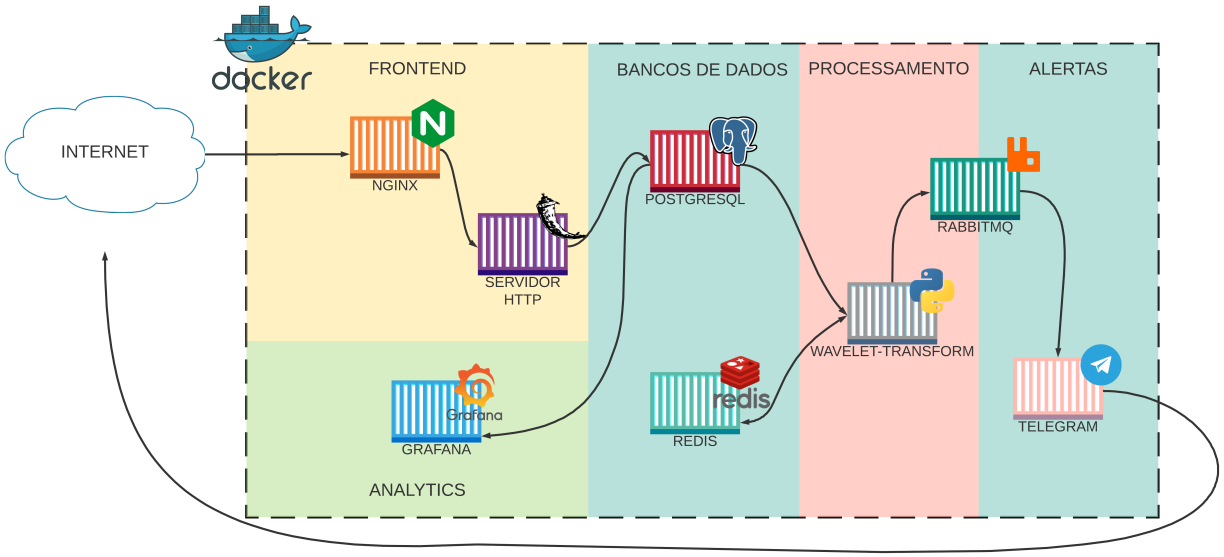


Figura 4.18: Arquitetura interna do servidor central proposto.

Capítulo 5

Experimentos e resultados

Este capítulo apresenta o processo de validação do sistema, que consistiu de experimentos realizados para construção do conjunto de dados, nos quais são apresentados e servem como entradas para fórmulas que calculam a *acurácia*, *sensibilidade* e *especificidade* da detecção de quedas, além de dados estatísticos apresentados em forma de gráficos. Após exposição dos resultados os mesmos são discutidos, bem como os valores de atrasos obtidos nos testes.

5.1 Experimentos

5.1.1 Descrição

Os testes consistiram na realização de 100 eventos de queda e não-queda, e tais ações foram executadas de maneira aleatória, para tentar reproduzir uma situação real cotidiana, onde nenhuma atividade é pré-programada. Antes de cada evento realizado, o sujeito de teste comunicava ao observador qual ação ele iria realizar e o mesmo verificava se o evento realizado resultou em detecção de queda ou não, anotando qual evento foi feito, e se o sistema detectou a queda ou não. O experimento envolveu apenas os autores do projeto e não houve uso de bonecos de testes (*Rescue Randy*). Nos testes a XDK foi fixada no tórax, pois segundo Gjoresk et al. [38], constitui-se como uma das melhores posições para a detecção desse tipo de evento no corpo humano. A Figura 5.1 apresenta a posição da XDK no corpo humano.

Apesar dos testes produzirem apenas 100 amostras, procurou-se aproximar ao máximo uma situação real, realizando todos os ensaios com sujeitos de testes humanos. Esta abordagem difere da adotada por Litvak et al. [21], que só usa bonecos de teste. Além disso, eles realizaram um total de 106 eventos, não diferenciando muito dos números obtidos no presente trabalho. Já Rodrigues [22], utilizou um conjunto de voluntários realizando algumas ações, criando assim, 100 eventos de queda e não-queda para cada usuário.

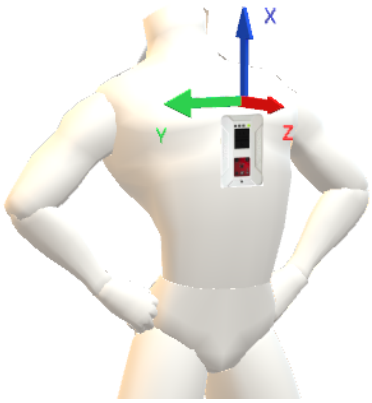


Figura 5.1: Ilustração da XDK no tórax humano, contemplando eixos X, Y e Z do acelerômetro.

5.1.1.1 Ambiente de Testes

Pensando no uso focado em situações do cotidiano, onde o usuário do sistema se encontra em casa, os testes de detecção de quedas foram todos realizados em um cenário residencial. Devido à natureza dos testes efetuados, envolvendo queda de pessoas, os ensaios foram executados em um quarto com dimensões de aproximadamente 6,5 m², piso de cerâmica e espaço livre para praticar os eventos de queda e não-queda. Realizar as provas de quedas em um único quarto permitiu que o ambiente fosse preparado com colchões para amparar os tombos do sujeito de teste, o que não seria possível em aposentos como cozinha ou banheiro, apesar de tais cômodos serem os ambientes mais comuns de quedas de idosos.

5.1.2 Metodologia

5.1.2.1 Eventos

O sistema proposto classifica a atividade de forma binária, produzindo apenas dois tipos de resultados: queda ou não-queda. Porém, durante os testes, realizamos diversos tipos de atividades, para averiguar a eficácia do sistema em diversas categorias de quedas e não-quedas. Importante destacar que após a execução de cada movimento (seja queda ou não queda), o individuo de teste aguardava cerca de 6 a 8 segundos para que o sistema reconhecesse o movimento, apresentando a decisão de queda ou não queda. O ponto inicial para cada movimento realizado era a posição em pé, na qual os valores capturados dos eixos Y e Z estavam nulos e os dados do eixo X permaneciam em 10 m/s² (Valor referente a gravidade).

As quatro categorias de quedas reproduzidas foram as seguintes:

- Queda para a frente;
- Queda para trás;

- Queda para a esquerda;
- Queda para a direita;
- Queda ao sentar em uma cadeira (em qualquer direção);

As quatro primeiras categorias de quedas foram pensadas para haver o envolvimento de todos os eixos do acelerômetro utilizado. Por exemplo, na queda para a frente, os valores coletados pelos eixos X e Z são os que mais sofrem mudanças, enquanto os dados referentes ao eixo Y praticamente se mantêm constantes. Já na queda para a direita, os valores dos eixos X e Y são os que apresentam maior alteração, enquanto os provenientes do eixo Z praticamente não se modificam. O ato da queda ao sentar em uma cadeira foi escolhido com o intuito de buscar um teste de uma queda mais "leve", pois o centro de massa da pessoa encontra-se mais próximo do chão, provocando menores oscilações dos componentes de aceleração. Vale ressaltar que esse ato é muito comum em pessoas idosas, pois apresentam problemas auditivos, gerando assim, perdas significativas no equilíbrio.

Os tipos de atividades "não-quedas" considerados durante os ensaios consistiram nos seguintes:

- Sentar (Esse ato foi caracterizado pelo indivíduo estar ao lado de uma cadeira antes de executar a ação);
- Deitar (Esse ato foi caracterizado pelo indivíduo estar perto do colchão antes de executar a ação);
- Andar (Esse ato foi caracterizado por realizar a ação de forma natural);
- Correr (Esse ato foi caracterizado por realizar a ação de andar com uma velocidade acima da normalidade);
- Agachar;
- Ajoelhar.

As três primeiras atividades representam o que consideramos as principais ações do dia-a-dia que não são quedas. A ação de deitar é a que mais se destaca devido à necessidade de movimentação completa do corpo. Em seguida, temos atividades menos comuns, porém que envolvem a movimentação em múltiplos eixos do tronco da pessoa e, consequentemente geram variação nos sinais capturados pelo acelerômetro ocasionando erros na detecção. A Tabela 5.1 mostra um resumo de todos os eventos reproduzidos durante os testes, e se tal evento é uma queda ou não-queda.

Tabela 5.1: Eventos realizados durante os testes.

Eventos	**Queda**	**Não Queda**
Sentar		X
Deitar		X
Andar		X
Correr		X
Agachar		X
Ajoelhar		X
Queda para a frente	X	
Queda para trás	X	
Queda para a esquerda	X	
Queda para a direita	X	
Queda ao sentar em uma cadeira (em qualquer direção)	X	

A Figura 5.2 apresenta o sinal de aceleração capturado em cada eixo para o evento “sentar”. O sujeito de teste esteve parado e em pé entre os instantes 16:15:55 e 16:16:20. Logo após, foi iniciado o ato de sentar. Neste momento, houve uma variação da aceleração no eixo X, um aumento e estabilização dos valores referentes ao eixo Z e, semelhantemente aos resultados no eixo X, houve uma variação somente durante o movimento nos dados do eixo Y, onde houve a transição entre os estados “em pé” e “sentado”. Durante o resto do teste, o usuário não efetuou mais ações, permanecendo sentado.

Figura 5.2: Sinais da aceleração em função do tempo para o evento “sentar” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

A Figura 5.3 ilustra a leitura do sensor em cada eixo durante um evento “deitar”. Conforme a Figura mostra, os sinais registrados partem de um ponto onde a pessoa estava em pé, onde o eixo

X apresenta valores constantes em aproximadamente 10 m/s² (aceleração da gravidade) e o eixo Z apresenta valores em 0 m/s². A ação de deitar começou às 16:08:25, onde pode-se reparar que o eixo X passou a assumir valores centrados nos 0 m/s², e o eixo Z passou a apresentar aceleração de 10 m/s². Isso indica que houve alteração na orientação do tronco do usuário, típico de uma atividade de deitar.

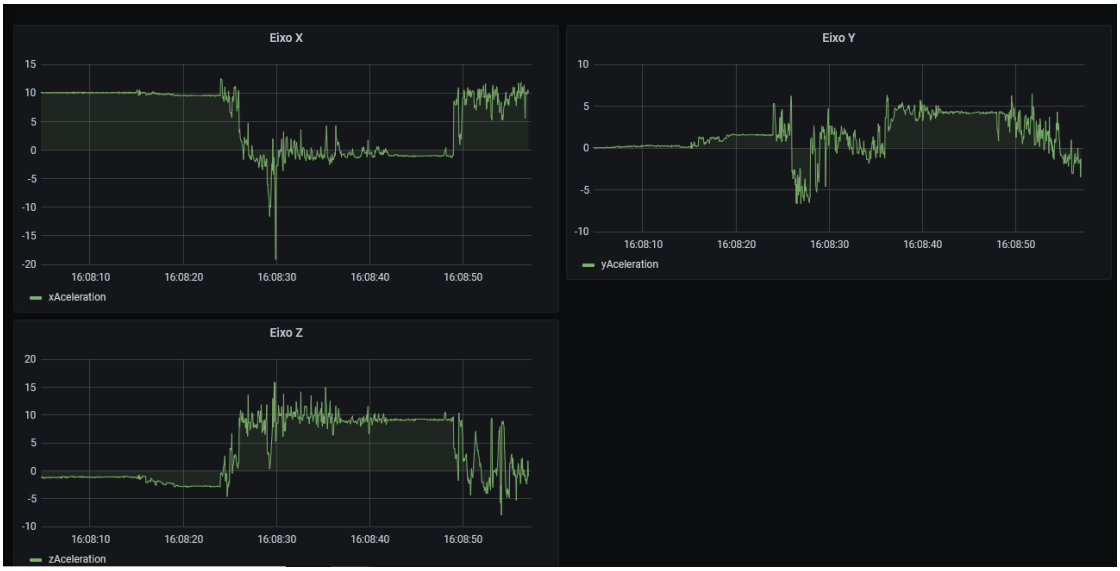


Figura 5.3: Sinais da aceleração em função do tempo para o evento “deitar” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

O terceiro evento de não-queda avaliado é o evento “andar”. Na Figura 5.4, vemos uma situação clara de tal evento. O sinal capturado começa mostrando uma pessoa parada, entre os instantes 16:21:00 e 16:21:30. Após este intervalo, o sujeito de teste começa a andar, provocando agitações dos sinais de aceleração em todos os eixos. A atividade dura cerca de 90 segundos, quando o usuário para o movimento, e fica parado durante o resto do tempo.

A Figura 5.5 apresenta os sinais captados para o evento “correr” nos três eixos do acelerômetro, no qual exibem os sinais coletados pelos sensores durante a realização do evento “correr”. Inicialmente, podemos verificar que os sinais obtidos são bastante similares aos da Figura 5.4, devido às oscilações de todos os eixos. Entretanto, diferente do evento “andar”, as variações dos sinais são maiores, com a aceleração chegando a picos e vales de 30 m/s² e 0 m/s², respectivamente, no eixo X, enquanto o mesmo eixo do evento “andar” apresenta picos e vales de 15 m/s² e 6 m/s², nesta ordem.

Na Figura 5.6 temos a ilustração do evento de “agachamento”. A ação acontece durante o intervalo entre 16:33:32 e 16:33:40, onde houve a transição entre a pessoa em pé e a pessoa agachada, em cócoras. Nota-se que, durante a ação, os três eixos tiveram oscilações nítidas e, depois da atividade, os dados dos eixos X e Y voltaram aos seus valores naturais, semelhantes ao início dos respectivos gráficos. Os dados do eixo Z ficaram constantes em aproximadamente -2

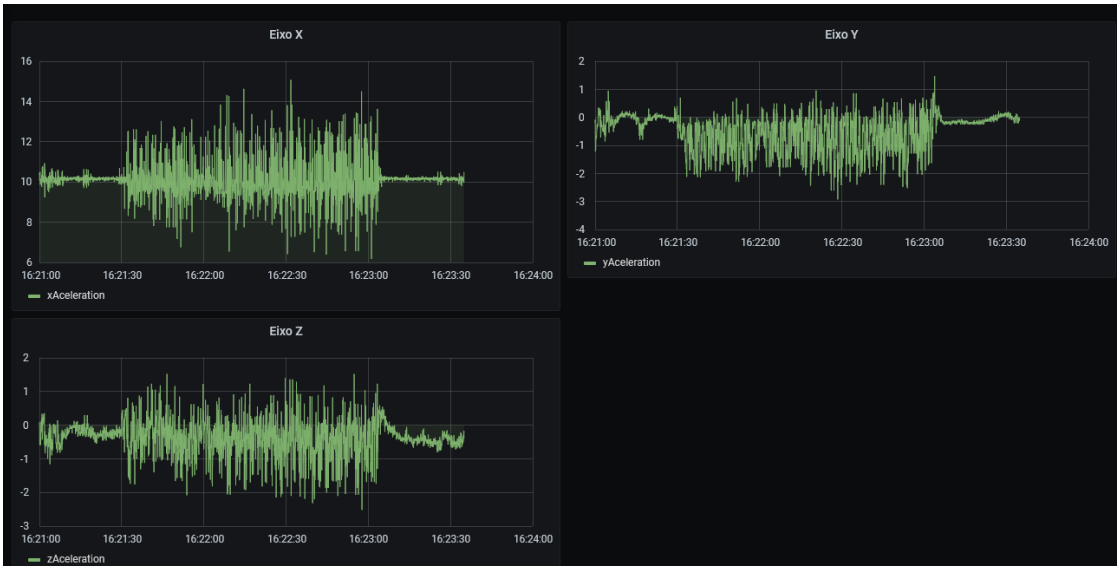


Figura 5.4: Sinais da aceleração em função do tempo para o evento “andar” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

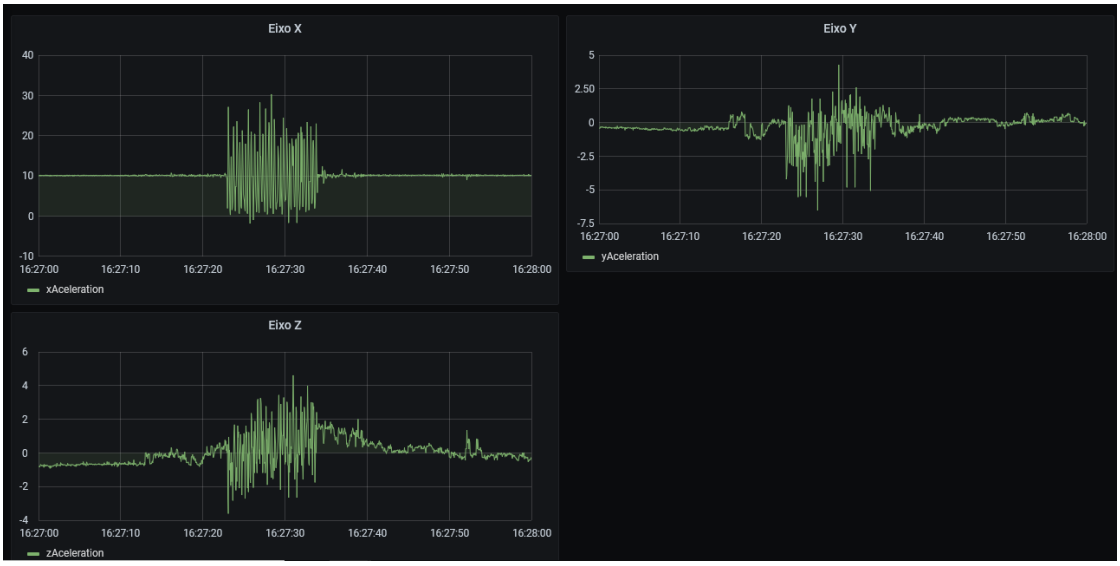


Figura 5.5: Sinais da aceleração em função do tempo para o evento “correr” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

m/s², pois o sujeito de teste finalizou o movimento com o tronco levemente inclinado para a frente. Considerando a orientação dos eixos conforme a Figura 5.1 e sabendo a posição final do indivíduo na queda, concluímos que o eixo Z aponta para o chão, explicando a aceleração negativa.

Figura 5.6: Sinais da aceleração em função do tempo para o evento “agachar” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

Os gráficos da Figura 5.7 mostram os sinais de aceleração lidos pelo sensor durante um evento “ajoelhar”. A ação começou no instante 16:36:50, onde pode-se ver a presença de tremulações nos três eixos até o momento onde a pessoa ajoelha-se, que nos gráficos corresponde ao instante 16:37:00. Durante o intervalo onde se encontra ajoelhada, os valores do eixo Z se mantém em -2 m/s², indicando uma leve inclinação do tronco, normal durante tal tipo de atividade. Depois de 25 segundos ajoelhada, a pessoa se levanta, gerando novas oscilações. No final, a pessoa se encontra em pé, com os dados dos eixos X, Y e Z apresentando valores constantes de aceleração em cerca de 10 m/s², 0,8 m/s² e -0,2 m/s², respectivamente.

A Figura 5.8 mostra os valores de aceleração referentes ao evento “queda para trás” em cada eixo. A queda ocorreu aproximadamente no instante 16:49:37, sendo que o sujeito de teste permaneceu deitado após a queda. Nota-se que os valores dos três eixos são afetados na forma de variações repentinas, gerando picos. Os sinais das quedas (independente do seu tipo) são bem semelhantes, sendo constituídos por atingir altos níveis de aceleração em um curto espaço de tempo. Estas descontinuidades podem aparecer em um ou mais gráficos dos componentes de aceleração.

5.1.2.2 Medidas de Desempenho

Como o sistema produz apenas dois tipos de respostas: “queda” ou “não-queda”, pode-se dizer que é empregada a classificação binária dos dados. Há muitas formas de se medir o desempenho

Figura 5.7: Sinais da aceleração em função do tempo para o evento “ajoelhar” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

Figura 5.8: Sinais da aceleração em função do tempo para o evento “queda para trás” (em m/s²). Nos eixos das abscissas temos o horário da coleta dos dados e nos eixos das ordenadas temos a aceleração em m/s².

de um classificador binário, e a escolha de tais métricas normalmente depende da aplicação. A *acurácia*, *sensibilidade* e *especificidade* foram as medidas escolhidas para avaliar o sistema. De modo a definir as medidas de desempenho mencionadas, necessitamos classificar os eventos de detecção de queda conforme identificados por nosso sistema. No universo das amostras coletadas, temos os dados subdivididos em:

- Verdadeiros positivos (VP): Eventos de quedas que foram identificados corretamente pela solução;
- Verdadeiros negativos (VN): Eventos de não-quedas que foram identificados corretamente pela solução;
- Falsos positivos (FP): Eventos de não-quedas que foram incorretamente identificados como quedas pela solução;
- Falsos negativos (FN): Eventos de quedas incorretamente identificados como não-quedas pela solução;

Para que o sistema seja confiável, é necessário minimizar a quantidade de falsos positivos e falsos negativos. A Eq. (5.1) apresenta a definição da medida de desempenho "*acurácia*", a qual mede o quão corretamente a classificação binária consegue discriminar os dados que possuem a condição testada (queda), dos que não possuem a condição (não-queda), isto é, a *acurácia* mede a *proporção de predições corretas (verdadeiros positivos e verdadeiros negativos) dentre todos os dados coletados*.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

A *sensibilidade* mede a proporção de eventos positivos conforme apresentado na Eq. (5.2), sendo interpretada como a proporção de quedas que foram detectadas corretamente.

$$\textit{Sensibilidade} = \frac{VP}{VP + FN} \tag{5.2}$$

Já a *especificidade* representa o complemento da *sensibilidade*, pois a medição tem como resultado a proporção de negativos corretamente previstos, conforme a Eq. (5.3). Interpreta-se a *especificidade* do sistema como a taxa de eventos que não possuem condição de queda e que são corretamente identificados como não-queda.

$$\textit{Especificidade} = \frac{VN}{VN + FP} \tag{5.3}$$

5.2 Resultados

5.2.1 Apresentação e Discussão dos Resultados

Nesta seção apresentamos os resultados obtidos através dos ensaios reproduzidos. Foram reproduzidos um total de 24 eventos de "queda" e 76 eventos de "não-queda". A Figura 5.9 apresenta um gráfico de setores que mostra que 72 ações foram classificadas como não sendo quedas, enquanto 28 foram detectadas como quedas.

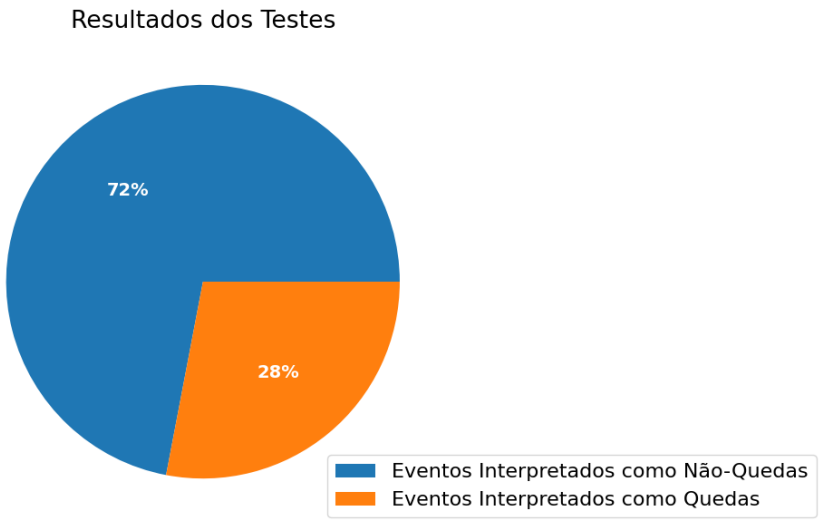


Figura 5.9: Resultados dos testes.

Com os resultados dos testes coletados, foi elaborada uma matriz de erro, mostrada na Tabela 5.2. A Figura 5.10 mostra a proporção de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos. Segundo o gráfico, a maior parte dos eventos foram verdadeiros negativos, obtendo muito mais proporção que os verdadeiros positivos. Isso se dá pelo fato da quantidade de eventos "não-queda" executados ser bem maior do que a quantidade de eventos "queda". Além disso, como explicado anteriormente, o ideal é que o sistema possua uma baixa proporção de falsos negativos, visto que tal grupo representa as quedas reais que o sistema não conseguiu identificar corretamente. O gráfico mostra que os falsos negativos corresponderam a 6% dos resultados totais. Também percebe-se a baixa taxa de falsos positivos, que não ultrapassou dos 10%. A partir dos valores apresentados na matriz de erro na Tabela 5.2, as medidas de desempenho foram calculadas, produzindo os valores apurados conforme mostra a Tabela 5.3.

Analisando os resultados da Tabela 5.3, observa-se que a solução teve *acurácia* de 84%, acertando 84 vezes e errando 16 vezes. Dentre os erros, observamos que o mais crítico deles é a ocorrência de falsos negativos, pois é a situação de ocorrer a queda e não detectá-la. Na solução,

Tabela 5.2: Matriz de Erro dos resultados dos testes

Valor Real \ Valor Classificado	**Queda**	**Não-Queda**
Queda	18	6
Não-Queda	10	66

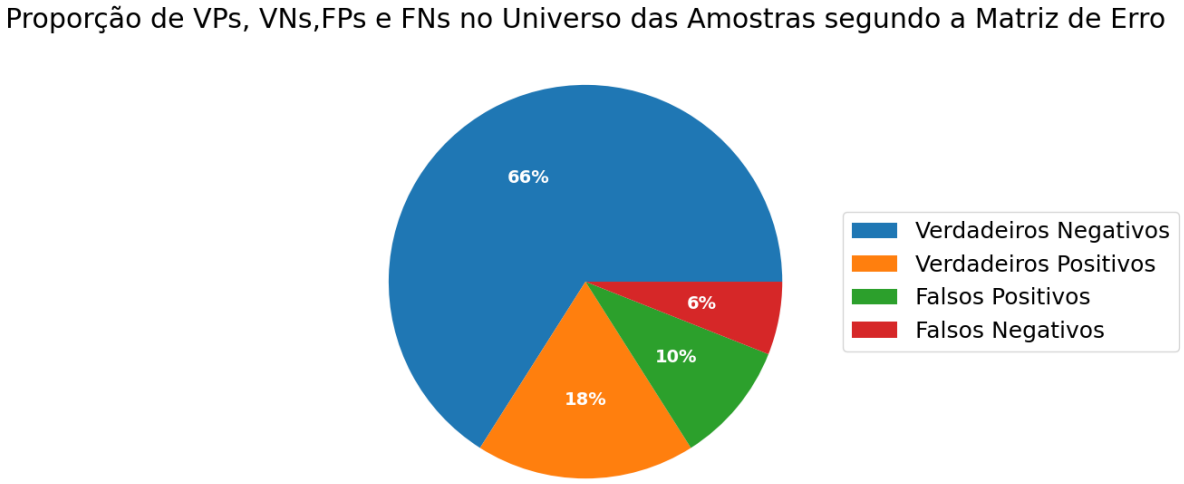


Figura 5.10: Proporção de VPs,VNs,FPs e FNs no Universo das Amostras segundo a Matriz de Erro.

Tabela 5.3: Medidas de desempenho calculadas

Medida de desempenho	**Valor Obtido**	**Valor de referência segundo a literatura**
Acurácia	84,00%	Mínimo de 90%
Sensibilidade	75,00%	Mínimo de 80,74%
Especificidade	86,84%	Mínimo de 98%

este caso específico foi o que ocorreu menos vezes comparado aos outros, conforme a Figura 5.10.

Dentre às três medidas de desempenho, a que atingiu menor resultado foi a *sensibilidade*, apresentando valores abaixo dos 80%. Pode-se explicar tal número pela quantidade de verdadeiros positivos não ser tão alta, devido à dificuldade de se testar as quedas em grande quantidade. Com uma quantidade maior de amostras seria evidenciado uma melhora no desempenho, apresentando assim, um teste mais sólido, mas pelo fato de envolver quedas de pessoas reais, não foi possível esse aumento desejado. Segundo a Eq. (5.2), quanto menor a quantidade de verdadeiros positivos, menor a *sensibilidade*. Já a *especificidade* apresentou-se elevada, devido à grande quantidade de verdadeiros negativos, que são os que se destacam na Figura 5.10 além de ser uma característica

desses tipos de sistema. A baixa presença de falsos positivos também influencia no alto valor atingido neste medidor. Comparando as métricas obtidas com a literatura, temos que os valores ficaram próximos, com exceção da *especificidade*, como mostra a Tabela 5.3. Os medidores não atingiram os valores adequados à literatura devido à quantidade limitada de amostras, na qual se apresentou insuficiente perante o estudo. Vale a pena ressaltar que as amostras foram limitadas pelo motivo das quedas serem reais, sendo executadas pelos autores do projeto.

A Figura 5.11 apresenta todos os tipos de quedas realizados durante os ensaios, e mostra o número de quedas detectadas (verdadeiros positivos) e não detectadas (falsos negativos) de cada classe de queda. A partir desses dados, pode-se inferir que as "quedas para trás" e "quedas ao sentar em uma cadeira" são as que mais confundem o sistema, devido à grande porcentagem de quedas interpretadas como não-quedas.

Nas "quedas para trás", a porcentagem de quedas mal-classificadas foi de 44,44% (4 das 9 quedas), muito provavelmente pelo fato da pessoa não se jogar completamente no momento do tombo, produzindo uma caída desacelerada, que se assemelham ao evento "deitar".

Nas quedas ao sentar em uma cadeira, 33,33% (1 de 3) foram detectadas de forma errônea. Pode-se argumentar que tal resultado ocorreu porque se trata de quedas mais leves, devido ao centro de massa do indivíduo estar mais próximo do chão. Quedas mais leves nem sempre atingem o limiar de detecção, logo a detecção de tais acidentes não é assegurada. Uma possível solução para detectar quedas ao sentar em uma cadeira de forma mais eficiente seria diminuir os limiares, porém tal prática pode introduzir novos falsos positivos, algo indesejado para o sistema. Estas quedas também são mais complicadas de serem realizadas de forma segura e autêntica. Por instinto, para executar a queda de maneira segura, o sujeito de teste desacelerava ou amparava a queda com as mãos, fazendo com que o teste perdesse um pouco de sua autenticidade, pois a aceleração é reduzida fazendo com que o limiar de detecção não fosse ultrapassado em alguns casos. Por outro lado, temos que as quedas para frente, para a esquerda e para a direita foram as que o sistema mais conseguiu compreender corretamente, com "Frente" e"Direita" atingindo 100% de acerto, e "Esquerda" alcançando 83,33%. Uma explicação das taxas de acerto obtidas para estes tipos de eventos está no fato das quedas acontecerem quando o indivíduo estava em pé (diferente das quedas ao sentar em uma cadeira), situação em que seu centro de massa está mais distante do chão. Quanto mais distante do chão está o centro de massa da pessoa, mais aceleração é captada pelo sensor durante a queda, o que garante que tais quedas quase sempre ultrapassem pelo menos um dos limiares pré-estabelecidos.

Como ilustra a Figura 5.12, o evento que mais introduziu falsos positivos foi o evento "correr" com 6 quedas detectadas incorretamente. Isso pode ser explicado pelo fato do evento se parecer muito com uma queda, pois envolve a vibração de todos os eixos do sensor do acelerômetro e gera a movimentação ligeira do tronco, causando picos nas leituras do sensor e consequente superação do limiar de detecção. Como a solução visa um ambiente AAL, com foco em idosos, este tipo de limitação não influenciará de maneira relevante a detecção de quedas, visto que idosos

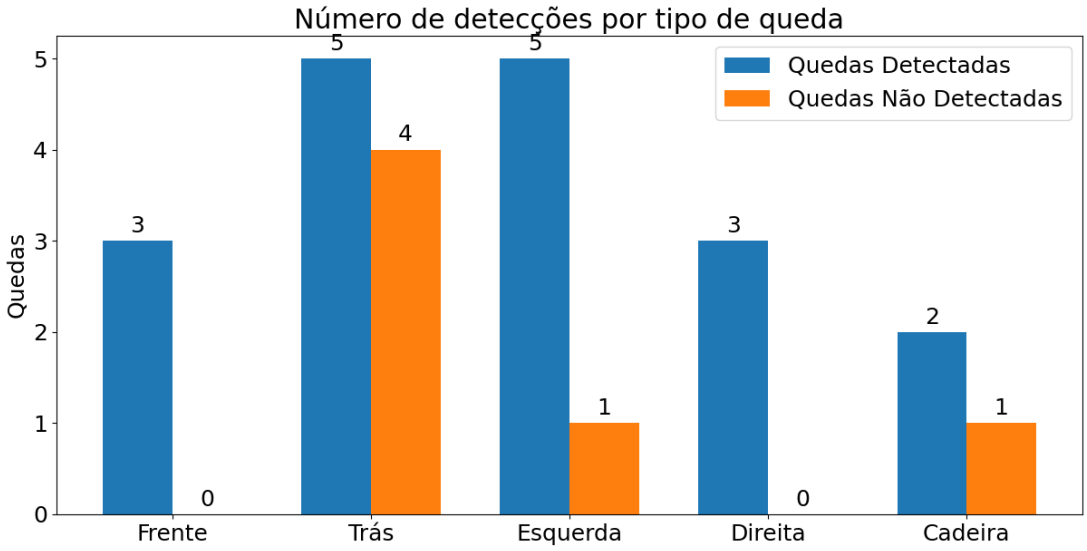


Figura 5.11: Número de Detecções por tipo de queda.

dificilmente correm dentro de seus domicílios.

A Figura 5.12 faz uma análise de cada tipo de atividade "não-queda", mostrando a quantidade de cada atividade que foi identificado como queda (falso positivo) em azul e a quantidade que foi identificado como "não-queda" (verdadeiro negativo) em laranja. Nos testes, foram realizados: 22 eventos "sentar", 14 eventos "deitar", 21 eventos "andar", 10 eventos "correr", 4 eventos "agachar" e 5 eventos "ajoelhar". Analisando os três principais eventos: "sentar", "deitar" e "andar", pode-se dizer que o desempenho do sistema é satisfatório, pois a quantidade de falsos positivos durante a realização de tais eventos é quase desprezível. Os eventos de "sentar" e "deitar" apresentaram um erro cada (em laranja), entretanto a ação de "andar" resultou em duas falhas, totalizando 4 erros do sistema em 57 amostras dos respectivos eventos (22 eventos de sentar, 14 eventos de deitar e 21 eventos de andar). Partindo destes dados, temos que a solução tem taxas de acerto de: 95,45% para os eventos "sentar"; 92,85% para os eventos "deitar"; 90,47% para os eventos "andar". Tomando como referência a Eq. 5.3, a quantidade baixa de falsos positivos nestes três eventos, além da grande quantidade de verdadeiros negativos fez com que a *especificidade* do sistema seja bastante alta, chegando ao valor de 86,84%. A *especificidade* do sistema foi o índice de desempenho que alcançou o maior valor conforme apresentado na Tabela 5.3.

Além da quantidade baixa de erros, os falsos positivos podem ter sido gerados pela velocidade do movimento. Por exemplo, analisando as Figuras 5.4 e 5.5 é possível observar as semelhanças nas variações da aceleração entre os eventos de "andar" e "correr", diferenciando-se apenas pela amplitude maior na ação de "correr", ou seja, se o sujeito de teste andar de forma mais rápida que o comum, os sinais gerados pelo acelerômetro se aproximam dos gráficos típicos de um evento "correr", reduzindo a taxa de acerto na ação realizada.

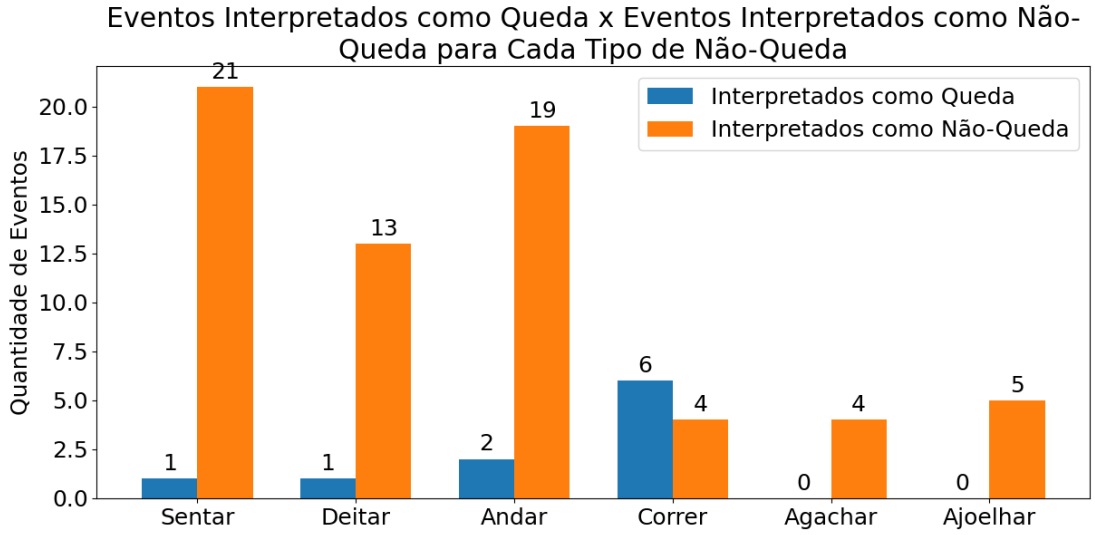


Figura 5.12: Eventos Interpretados como Queda x Eventos Interpretados como Não-Queda para Cada Tipo de Não-Queda.

Por fim, temos os eventos “agachar” e “ajoelhar”. Pela menor recorrência de tais atividades no dia-a-dia, foram realizadas apenas 4 e 5 repetições de cada evento, respectivamente. Porém, dentre as 9 iterações, nenhuma ocasionou falso positivo. Pode-se explicar tal resultado pelo fato dos dois eventos serem mais lentos e cadenciados, o que provoca um baixa amplitude nos componentes de aceleração do sensor, que acabam ficando abaixo do limiar de detecção.

5.2.2 Apresentação dos Atrasos Obtidos

Além dos resultados acima, foi considerado o tempo de resposta do sistema, que deve compreender aos eventos de queda de forma rápida. Como o sistema envolve diversas etapas, foram calculados três atrasos típicos da literatura: atraso de envio dos dados para o servidor central (atraso 1), atraso de processamento do servidor em nuvem (atraso 2), e atraso de notificação (atraso 3). O primeiro atraso é dependente do estado da conexão com a Internet e da distância do servidor, logo pode produzir valores mais variados e menos constantes. Já o segundo atraso obedece ao tempo de processamento das informações coletadas, e não sofre grandes oscilações, como 5 segundos, por exemplo. Por último, há o atraso de notificação, que também é sujeito da condição da conexão com a Internet.

Na Figura 5.13 são mostrados os três atrasos do sistema, além dos componentes que influenciam em cada atraso. O atraso 1 é contado a partir do dispositivo XDK, que envia seus dados para o *gateway*. Este, no que lhe concerne, envia os dados para o servidor central pela Internet. O atraso 1 é calculado até o momento em que os dados chegam ao servidor central. Em seguida, o

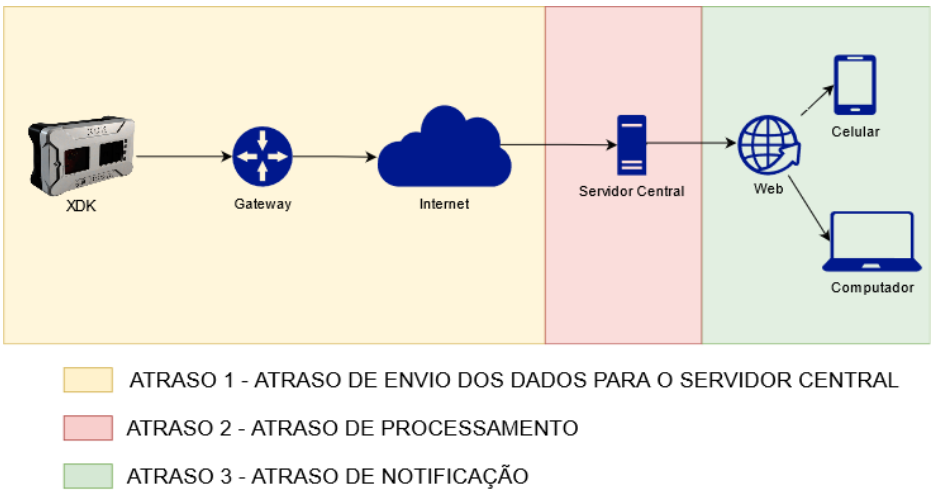


Figura 5.13: Atrasos do sistema.

atraso 2 começa a ser calculado, e é avaliado até o ponto em que os dados são processados. Já o terceiro atraso corresponde ao tempo que leva para que a decisão do algoritmo seja notificada.

Foram realizados testes para a descoberta dos valores típicos de cada atraso supracitado. Para esta execução foi utilizada um servidor central em São Paulo, no qual apresentava um atraso de cerca de 45 ms, que foi calculado utilizando o protocolo ICMP (do inglês *Internet Control Message Protocol*). Durante estes testes, foram obtidos os valores típicos em um intervalo de 0-1 segundos para o atraso 1, 2-4 segundos para o atraso 2, e 0-1 segundos para o atraso 3. No final, obteve-se um atraso total variando em aproximadamente 2-6 segundos.

Capítulo 6

Conclusão

Este capítulo refere-se à conclusão do trabalho, sendo este subdivido em alguns subtópicos relacionados. São eles: argumentações gerais (Seção 6.1); as limitações encontradas em todo o projeto (Seção 6.2); finalizando com algumas propostas para trabalhos futuros na área (Seção 6.3).

6.1 Argumentações Gerais

Neste trabalho, foi implementado um sistema para ambientes de vida assistida, consistindo-se de um projeto com o intuito de detectar quedas utilizando a arquitetura e comunicação proporcionada pelo paradigma da Internet das Coisas. Este sistema é pensado para os idosos, pois este grupo possui um alto índice de morte e traumas relacionados a quedas. O sistema foi pensado para ser uma solução completa, desde a coleta de dados através de sensores até a identificação e notificação da queda.

O sistema visou quatro objetivos gerais, os quais são:

- Possibilitar uma rápida reação familiar e hospitalar, através de uma detecção eficiente e notificação veloz.
- Proporcionar um gerenciamento remoto de familiares sobre possíveis quedas do usuário.
- Obter um sistema de fácil implementação e baixo custo.
- Apresentar um projeto com o mínimo custo computacional.

O sistema começa com um micro-controlador da empresa Bosch, chamado XDK, composto por diversos sensores. Este dispositivo é posicionado no tórax, de modo que o sensor de acelerômetro, presente no micro-controlador, capte a aceleração contida nos movimentos feitos pelo ser

humano. Estes movimentos são captados a cada 50 ms, formando assim um vetor de 20 amostras por segundo, no qual é enviado via requisição HTTP ao *gateway* local. O *gateway* envia as requisições ao servidor na nuvem desenvolvido, no qual armazena os dados coletados em um banco de dados, e depois filtra estes sinais utilizando a Transformada Wavelet Discreta, de modo a separar os componentes de altas frequências à procura de picos repentinos, que poderiam ser quedas. Após a filtragem, há a tomada de decisão, que identifica quedas através de limiares alcançados pela aceleração. Por fim, é acionado um robô no Telegram, que notifica os responsáveis sobre a queda, informando o horário do acontecimento.

Para avaliar o desempenho da técnica de detecção de quedas proposta, foi desenvolvido um experimento contendo 100 amostras aleatórias de movimentos referentes a "quedas" e "não-quedas". Estas ações denominadas de "não-quedas" foram realizadas pensando nas atividades do dia-a-dia: andar, correr, sentar, deitar, agachar e ajoelhar. Com relação às quedas foram consideradas as seguintes possibilidades: quedas para frente, para a direita, para a esquerda, para trás e quedas de uma cadeira.

Conforme os resultados apresentados no Capítulo 5, mesmo com poucas amostras, o sistema obteve resultados de acurácia, sensibilidade e especificidade próximos da literatura, sendo eficiente em quedas para frente, direita e esquerda. Já nas ações consideradas "não-quedas" o sistema obteve resultados honestos, gerando assim, poucos alarmes falsos em movimentos cotidianos como sentar, andar e deitar.

6.2 Limitações

Devido à abordagem intrusiva, o sistema possui limitações características. A principal é que o monitoramento não poderá ser feito a todo momento, como, por exemplo, quando o idoso estiver tomando banho, o que é ainda mais agravado pela alta ocorrência de acidentes em banheiros. Além disso, a pessoa pode esquecer de usar o equipamento, o que incapacitaria o sistema de monitorar a situação posicional da pessoa. Há ainda aqueles que se recusam a utilizar o sensor por motivos de desconforto estético visto que o equipamento não é discreto o suficiente, o que é mais uma barreira à adoção plena da solução.

Pensando a longo prazo, o servidor em nuvem será uma limitação pela falta de espaço no armazenamento da versão gratuita. As técnicas utilizadas no sistema também introduziram limitações. O uso de um microcontrolador sem fio gera a necessidade de utilização de uma bateria interna, na qual o usuário deve sempre recarregar a mesma. Outro ponto importante é a definição dos limiares de decisão de quedas por métodos empíricos, impossibilitando que tais limiares sejam os mais eficazes, sendo assim recomendado um estudo para encontrar os valores ideais para o sistema. Existe também, a utilização de um sensor apenas, gerando assim, uma dependência de uma única fonte de dados, o que pode ocasionar erros e falhas.

Com relação à arquitetura, a utilização do protocolo Wi-Fi restringe a mobilidade do monitorado, que deve sempre estar no alcance de um ponto de acesso associado a uma rede sem fio com Internet. Além disso, este protocolo não é pensado para o baixo consumo energético. Pensando nisso, o projeto poderia ter utilizado protocolos com foco nesse aspecto como os protocolos BLE (do inglês *Bluetooth Low Energy*) e LoRaWAN (do inglês *Long Range Wide Area Network*).

Examinando os experimentos realizados, tem-se como limitação a coleta de dados feita com simulação de quedas de voluntários que não tem o perfil de pessoas idosas, o que impossibilita assegurar a eficácia do sistema em cenários reais de quedas de pessoas mais velhas. Além disso, os ensaios de teste envolveram apenas um voluntário, devido às dificuldades do atual momento global, o que não é o ideal. Por fim, as quedas da cadeira não foram bem representadas pelo sujeito de teste, pois o cenário não permitia que a queda fosse feita de maneira segura e autêntica.

6.3 Trabalhos Futuros

A solução construída se mostrou eficaz na detecção de quedas, porém ainda há pontos em que trabalhos futuros podem focar para melhorar o desempenho e funcionalidade do sistema. Como possibilidades de melhorias, temos:

- A utilização de múltiplos sensores, de modo que haja uma monitoração constante em todo momento. Desta maneira, não haveria a total dependência em um único sensor, gerando assim uma maior acurácia, e consequentemente, uma maior confiabilidade. Os sensores mais recomendados seriam os dispositivos PIR (*Passive infrared sensor*), pois são eficientes em cômodos que não aceitam a utilização de sensores intrusivos como banheiros. Outros sensores que podem ser utilizados para agregar à detecção são os dispositivos de detecção acústica e sensores Doppler, nos quais reforçariam o método de decisão utilizado.
- Geração de gráficos e relatórios sob demanda que analisam quedas passadas e ilustram os horários do dia em que o usuário mais sofreu acidentes.
- Estudo apropriado para a definição de limiares ideais.
- Utilização de técnicas como análise de componentes principais (do inglês *Principal Component Analysis*) para melhores resultados na utilização da Transformada Wavelet.
- Detecção da pós-queda pode ser um caminho interessante, pois com isto seria possível saber a gravidade da queda, proporcionando assim, notificações variadas por grau de periculosidade.
- Integração com hospitais e sistemas de atendimento rápido, de modo a proporcionar uma maior agilidade no atendimento médico.

- Utilização de protocolos de baixo consumo energético, como BLE e LoRaWAN.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BANK, T. W. *Life expectancy at birth, total (years) - ADAPTADO*. Disponível em: <https://data.worldbank.org/indicator/SP.DYN.LE00.IN>. Acesso em: 06/03/2021.

[2] NOROOZIAN, M. The elderly population in iran: An ever growing concern in the health system. *Iranian journal of psychiatry and behavioral sciences*, v. 6, p. 1–6, 10 2012.

[3] BANK, T. W. *Fertility rate, total (births per woman) - ADAPTADO*. Disponível em: <https://data.worldbank.org/indicator/SP.DYN.TFRT.IN>. Acesso em: 06/03/2021.

[4] NCOA. *Falls Prevention Facts*. Disponível em: <https://www.ncoa.org/news/resources-for-reporters/get-the-facts/falls-prevention-facts/>. Acesso em: 13/11/2020.

[5] CONTROL, C. for D.; PREVENTION. *Important Facts about Falls*. Disponível em: <https://www.cdc.gov/homeandrecreationalsafety/falls/adultfalls.html>. Acesso em: 06/03/2021.

[6] ORACLE. *O que é Internet of Things*. Disponível em: <https://www.oracle.com/br/internet-of-things/what-is-iot>. Acesso em: 13/11/2020.

[7] SUNDMAEKER, H. et al. *Vision and Challenges for Realising the Internet of Things*. [S.l.]: European Union, 2010. 12 p. ISBN 978-92-79-15088-3.

[8] MARQUES, G.; PITARMA, R. An indoor monitoring system for ambient assisted living based on internet of things architecture. *International Journal of Environmental Research and Public Health*, v. 13, n. 11, 2016. ISSN 1660-4601. Disponível em: <https://www.mdpi.com/1660-4601/13/11/1152>.

[9] HÖLLER, J. et al. *From Machine-to-Machine to the Internet of Things*. [S.l.]: Elsevier Ltd., 2014. 10-13 p. ISBN 978-0-12-407684-6.

[10] CHEN, S. et al. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things Journal*, v. 1, n. 4, p. 349–359, 2014.

[11] FAROOQ, M. et al. A critical analysis on the security concerns of internet of things (iot). *International Journal of Computer Applications*, v. 111, p. 1–6, 02 2015.

[12] LI, S.; XU, L. D.; ZHAO, S. The internet of things: A survey. *Information Systems Frontiers*, Kluwer Academic Publishers, USA, v. 17, n. 2, p. 243–259, abr. 2015. ISSN 1387-3326. Disponível em: <https://doi.org/10.1007/s10796-014-9492-7>.

[13] BARFORD, L.; FAZZIO, S.; SMITH, D. An introduction to wavelets. *Instruments and Photonics Laboratory - HEWLETT PACKARD (HP)*, 1992.

[14] CARVALHO, J. L. A. de. *Ferramenta para Análise Tempo-Freqüencial da Variabilidade da Freqüencia Cardiaca*. 99 p. Dissertação (Mestrado em Engenharia Elétrica) — Universidade de Brasília, Brasília, 2003.

[15] TZANETAKIS, G.; ESSL, G.; COOK, P. Audio analysis using the discrete wavelet transform. In: . [S.l.: s.n.], 2001. p. 318–323.

[16] EDWARDS, T. *Discrete Wavelet Transforms: Theory and Implementation*. 1991.

[17] DAUBECHIES, I. *Ten Lectures on Wavelets*. USA: Society for Industrial and Applied Mathematics, 1992. 56-106 p. ISBN 0898712742.

[18] BARBON, S. *Transformada Discreta Wavelet DWT - Departamento de computação - Universidade Estadual de Londrina*. 2013. Disponível em: <http://www.barbon.com.br/wp-content/uploads/2013/02/PDS_Aula4.pdf>. Acesso em: 22/03/2021.

[19] Vieira Filho, J.; BAPTISTA, F. G.; INMAN, D. J. Time-domain analysis of piezoelectric impedance-based structural health monitoring using multilevel wavelet decomposition. *Mechanical Systems and Signal Processing*, v. 25, n. 5, p. 1550–1558, 2011. ISSN 0888-3270. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888327010004279>.

[20] LE, T. M.; PAN, R. Accelerometer-based sensor network for fall detection. In: *2009 IEEE Biomedical Circuits and Systems Conference*. [S.l.: s.n.], 2009. p. 265–268.

[21] LITVAK, D.; GANNOT, I.; ZIGEL, Y. Detection of falls at home using floor vibrations and sound. In: *2008 IEEE 25th Convention of Electrical and Electronics Engineers in Israel*. [S.l.: s.n.], 2008. p. 514–518.

[22] RODRIGUES, C. de A. P. *INFOrM - Uma abordagem para detecção de quedas baseada em sensores de movimento infravermelhos e acelerômetros*. 83 p. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Ceará, Fortaleza, 2016.

[23] WERNER, F. et al. Fall detection with distributed floor-mounted accelerometers: An overview of the development and evaluation of a fall detection system within the project ehome. In: *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*. [S.l.: s.n.], 2011. p. 354–361.

[24] MAULDIN, T. et al. Smartfall: A smartwatch-based fall detection system using deep learning. *Sensors*, v. 18, p. 3363, 10 2018.

[25] BOSCH. *XDK - Overview - Bosch Developer Portal*. Disponível em: <https://developer.bosch.com/web/xdk>. Acesso em: 10/05/2021.

[26] BOSCH. *Bosch XDK – Solução em Conectividade*. Disponível em: <https://www.youtube.com/watch?v=TJG3MqbLJk4ab_channel=BoschBrasil>. Acesso em: 27/03/2021.

[27] BOSCH. *Datasheet BMA280 Digital, triaxial acceleration sensor*. 1.8. ed. Kusterdingen, 2014. 120 p. Disponível em: <https://www.mouser.com/datasheet/2/783/BST-BMA280-DS000-11_published-786496.pdf>. Acesso em: 20/03/2021.

[28] BRUXEL, Y. *Sistema para Análise de Impacto na Marcha Humana*. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/61788/000825591.pdf?sequ>. Acesso em: 24/05/2021.

[29] BARRY, R. *Mastering the FreeRTOS™ Real Time Kernel*. 2 p. Disponível em: <https://freertos.org/fr-content-src/uploads/2018/07/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf>. Acesso em: 23/03/2021.

[30] WEOCK. *What is NTP/SNTP Protocols, How NTP works*. Disponível em: <https://iot.samteck.net/enablers/networking/what-is-ntp-sntp-protocols/>. Acesso em: 27/03/2021.

[31] FIELDING, R.; RESCHKE, J. *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Disponível em: <https://tools.ietf.org/html/rfc7230>. Acesso em: 27/03/2021.

[32] DOSSOT, D. *RabbitMQ Essentials*. [S.l.]: Packt Publishing, 2014. 7-10 p. ISBN 978-1-78398-320-9.

[33] AWS. *Message Queues*. Disponível em: <https://aws.amazon.com/message-queue/>. Acesso em: 22/03/2021.

[34] LABS, R. *Redis*. Disponível em: <https://redis.io/>. Acesso em: 22/05/2021.

[35] MERKEL, D. Docker: Lightweight linux containers for consistent development and deployment. *Linux J.*, Belltown Media, Houston, TX, v. 2014, n. 239, mar. 2014. ISSN 1075-3583.

[36] REDHAT. *Containers vs VMs*. Disponível em: <redhat.com/en/topics/containers/containers-vs-vms>. Acesso em: 26/03/2021.

[37] VELTE, A. T.; VELTE, T. J.; ELSENPETER, R. *Cloud Computing, A Practical Approach*. 1st. ed. [S.l.]: McGraw-Hill Education, 2009. 3-4 p. ISBN 978-0071626941.

[38] GJORESKI, H.; LUSTREK, M.; GAMS, M. Accelerometer placement for posture recognition and fall detection. In: *2011 Seventh International Conference on Intelligent Environments*. [S.l.: s.n.], 2011. p. 47–54.

ANEXOS

ANEXO A

Função principal - XDK coleta e envio dos dados

Listagem A.1: XDK coleta e envio dos dados

```
1   static void AppControllerFire(void* pvParameters)
2   {
3       [...]
4
5     if((i%frequency) == (frequency-1))
6     {
7       valorX = ((long int) sensorValue.Accel.X * 0.00980665);
8       valorY = ((long int) sensorValue.Accel.Y * 0.00980665);
9       valorZ = ((long int) sensorValue.Accel.Z * 0.00980665);
10
11      AcelX[i]= valorX;
12      AcelY[i]= valorY;
13      AcelZ[i]= valorZ;
14
15
16        struct tm tm =
17                {
18                        .tm_sec = 0, /* Seconds: 0-59 */
19                        .tm_min = 0, /* Minutes: 0-59 */
20                        .tm_hour = 0, /* Hours since midnight: 0-23 */
21                        .tm_mday = 0, /* Day of the month: 1-31 */
22                        .tm_mon = 0, /* Month of the year *since* January :
    0-11 */
23                        .tm_year = 0, /* Years since 1900 */
24                        .tm_wday = 0, /* not required */
25                        .tm_yday = 0, /* not required */
26                        .tm_isdst = 0, /* not required */
27                };
```

```
28
29          do
30              {
31                  retcode = SNTP_GetTimeFromServer(&sntpTimeStampFromServer,
    APP_RESPONSE_FROM_SNTP_SERVER_TIMEOUT);
32                  if ((RETCODE_OK != retcode) || (0UL ==
    sntpTimeStampFromServer))
33                  {
34                      printf("AppControllerFireTask : SNTP server time was
    not synchronized. Retrying...\r\n");
35                  }
36              } while (0UL == sntpTimeStampFromServer);
37
38              if (RETCODE_OK == retcode)
39              {
40                  retcode = TimeStamp_SecsToTm(sntpTimeStampFromServer, &tm);
41              }
42
43            [...]
44
45            HttpClient_initRequest(&destAddr, port, &msg_prt);
46            HttpMsg_setReqMethod(msg_prt, Http_Method_Post);
47            HttpMsg_setReqUrl(msg_prt,"/sensors");
48            HttpMsg_setHost(msg_prt,IP);
49
50            [...]
51       }
52
53
54       else {
55
56         valorX = ((long int) sensorValue.Accel.X * 0.00980665);
57         valorY = ((long int) sensorValue.Accel.Y * 0.00980665);
58         valorZ = ((long int) sensorValue.Accel.Z * 0.00980665);
59
60         AcelX[i]= valorX;
61         AcelY[i]= valorY;
62         AcelZ[i]= valorZ;
63
64
65           i += 1;
66       }
67     }
68
69
```

ANEXO B

Funções de cálculo de Transformada Wavelet no processamento de dados

Listagem B.1: Funções de cálculo de Transformada Wavelet no processamento de dados

```
    [...]

    import pywt

    [...]


def multi_level_wavelet_transform(signal,level,wavelet='haar'): # Multi-Level
    Wavelet Transform Function
    coeffs=pywt.wavedec(signal,wavelet,level=level)
    return coeffs

def multi_level_inverse_wavelet_transform(cA2,cD2,cD1,wavelet='haar'): #
   Multi-Level Inverse Wavelet Transform Function
    reconstructedSignal=pywt.waverec([cA2,cD2,cD1],wavelet)
    return reconstructedSignal

 [...]

def multi_level_analysis(axisArrayDict): # Function that filters all three
   signals (X,Y,Z), calls detect function, and gets fall's timestamps

    indexes = []

    for axis,axisArray in axisArrayDict.items():
        cA2,cD2,cD1 = multi_level_wavelet_transform(axisArray,level=2)  #
   Wavelet Transform each signal in 2 levels
        cA2 = cA2*0               # Filter cA2 out
```

```
25          cD2 = cD2*0                 # Filter cD2 out
26          reconstructed = multi_level_inverse_wavelet_transform(cA2,cD2,cD1)  #
     Reconstruct signal with inverse wavelet transform
27          if axis == "Eixo Z":    # Z-Axis has different threshold
28
29              indexes = [*indexes,*detect_fall_event(reconstructed,threshold
    =15).tolist()]
30          else:
31
32              indexes = [*indexes,*detect_fall_event(reconstructed,threshold=4)
    .tolist()]
33
34      if indexes:     # If detect function returns fall events, get these
    events's timestamps
35          get_timestamps(remove_redundant_indexes(indexes))
36
```

ANEXO C

Funções de decisão de quedas no processamento de dados

Listagem C.1: Funções de decisão de quedas no processamento de dados

```
1
2  [...]
3
4  from detecta import detect_peaks
5
6  [...]
7
8  def detect_fall_event(reconstructedSignal,threshold): # Detect Falls
     Function
9
10     indexes = detect_peaks(reconstructedSignal, show=False, threshold=
     threshold, mpd=20, mph=threshold, kpsh = True , title=False, edge='both')
11     #print("Detected fall events at the following moments: ",indexes)
12     return indexes
13
14 [...]
15
16 def multi_level_analysis(axisArrayDict): # Function that filters all three
     signals (X,Y,Z), calls detect function, and gets fall's timestamps
17
18     indexes = []
19
20     for axis,axisArray in axisArrayDict.items():
21         cA2,cD2,cD1 = multi_level_wavelet_transform(axisArray,level=2)  #
     Wavelet Transform each signal in 2 levels
22         cA2 = cA2*0               # Filter cA2 out
23         cD2 = cD2*0               # Filter cD2 out
24         reconstructed = multi_level_inverse_wavelet_transform(cA2,cD2,cD1)  #
```

```
    Reconstruct signal with inverse wavelet transform
25          if axis == "Eixo Z":    # Z-Axis has different threshold
26
27              indexes = [*indexes,*detect_fall_event(reconstructed,threshold
    =15).tolist()]
28          else:
29
30              indexes = [*indexes,*detect_fall_event(reconstructed,threshold=4)
    .tolist()]
31
32      if indexes:     # If detect function returns fall events, get these
    events's timestamps
33          get_timestamps(remove_redundant_indexes(indexes))
34
```

ANEXO D

Funções de envio de notificações para o Telegram

Listagem D.1: Funções de envio de notificações para o Telegram

```
 [...]

from telegram import Bot, Message
bot = Bot('INSIRA_TOKEN_AQUI')
channel_id='INSIRA_ID_DO_CANAL_AQUI'

[...]

def main():

[...]

   def callback(ch, method, properties, body):

        fallTime = body.decode()
        message = bot.sendMessage(channel_id,"Usuario de ID 00001 sofreu
    queda as {}".format(fallTime))

[...]

```