DISSERTAÇÃO DE MESTRADO

# SISTEMA VESTÍVEL PARA A DETECÇÃO DE QUEDAS EM HARDWARE RECONFIGURÁVEL

**JAVIER ALEXIS URRESTY SANCHEZ**

**Brasília, Setembro de 2018**

**UNIVERSIDADE DE BRASÍLIA**

FACULDADE DE TECNOLOGIA

**UNIVERSIDADE DE BRASÍLIA**
**FACULDADE DE TECNOLOGIA**
**DEPARTAMENTO DE ENGENHARIA MECÂNICA**

# SISTEMA VESTÍVEL PARA A DETECÇÃO DE QUEDAS EM HARDWARE RECONFIGURÁVEL

**JAVIER ALEXIS URRESTY SANCHEZ**

**DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS**

**APROVADA POR:**

---

**Prof. Dr. Daniel Maurício Muñoz Arboleda, FGA/UnB, PPMEC/UnB**
*Orientador*

---

**Prof. Dr. Ricardo Pezzuol Jacobi, CIC/UnB**
*Membro Interno*

---

**Prof. Dr. Helon Vicente Hultmann Ayala, ENM/PUC-RJ**
*Membro Externo*

**BRASÍLIA/DF, 26 SETEMBRO DE 2018**

Urresty Sanchez, Javier Alexis
Sistema vestível para a detecção de quedas em hardware reconfigurável / JAVIER ALEXIS URRESTY SANCHEZ. –Brasil, 2018.
107 p.

Orientador: Daniel Mauricio Muñoz Arboleda

Dissertação (Mestrado) – Universidade de Brasília – UnB
Faculdade de Tecnologia – FT
Programa de Pós-Graduação em Sistemas Mecatrônicos – PPMEC, 2018.

1. FPGA. 2. Acelerômetro. 3. Sistema de detecção de quedas. I. Daniel Mauricio Muñoz Arboleda, orientador. II. Universidade de Brasília. III. Faculdade de Tecnologia.

**Dedicatória**

*Dedicado a mis amados padres Celio y Maria del Soccorro y a mi querido hermano Andres, quines siempre me han apoyado en cada paso que doy. Gracias por tanto.*

*JAVIER ALEXIS URRESTY SANCHEZ*

**Agradecimentos**

*En primer lugar, a mis padres Celio y María del Socorro y a mi hermano Andrés, que nunca me han dejado solo, siempre me han apoyado en cada paso que doy y quienes se han convirtiendo en mi más grande fuente de inspiración para continuar luchando por mis sueños; y a toda mi familia, por brindarme su apoyo y estar siempre pendiente de mí.*

*A mi orientador, el profesor Daniel Muñoz, una excelente profesional y persona. Por su paciencia, dedicación y conocimientos compartidos, muchísimas gracias ingeniero. Al profesor Carlos Llanos, por todo el apoyo y ayuda recibida, antes y durante mi estadía en Brasil.*

*A mis amigos y compañeros quienes me ayudaron y participaron en mi investigación, Ruben, Cristian, Richard, Luis Arias, Fabian Barrera, Reurison, Carlos, William, Diana, Felipe, Oscar, Fabian, Edwin, Andres Rosero, Laura, Thompson, João Carlos, Jessé, José y João Guimares, muchas gracias por su aporte y amistad.*

*JAVIER ALEXIS URRESTY SANCHEZ*

# RESUMO

Os acidentes como as quedas podem ocorrer durante as atividades diárias de uma pessoa, sendo uma das principais fontes de morbidade e mortalidade na população idosa. A detecção em tempo real das quedas e a respectiva comunicação para um centro de emergências, cuidador ou familiar da pessoa, pode permitir assistência médica rápida, aumentando assim a sensação de segurança e reduzindo algumas das consequências negativas das quedas. O presente trabalho descreve o desenvolvimento de um sistema de detecção de quedas baseado em um sensor de aceleração e um microfone colocados no pulso do usuário, arquiteturas em hardware do algoritmo de detecção de quedas implementadas em um FPGA e um dispositivo *bluetooth*, encarregado de enviar a alarme gerada pelo algoritmo para um *smartphone* com um aplicativo Android, o qual envia uma mensagem de alerta para um contato previamente estabelecido.

O algoritmo para detecção de quedas está baseado em uma rede neural artificial e em um avaliador de limiares de som. A partir da identificação de um padrão no sinal de aceleração que possuem as quedas e que está constituída por 138 amostras, em conjunto com um processo de filtragem, discretização no tempo e normalização do sinal, foi treinada a rede neural artificial no Matlab, utilizando um conjunto de movimentos realizados por 12 voluntários, composto por 170 quedas e 170 atividades da vida diária. Por outro lado, o avaliador de limiares de som, foi implementado devido a que a rede pode classificar erradamente alguns movimentos da vida diária como uma queda usando como entrada o valor da aceleração. O avaliador está encarregado de avaliar dois limiares (superior e inferior) do sinal de som e quando estes são alcançados e a rede informa da detecção de uma queda, o sistema gera um alarme.

A rede neural artificial foi mapeada em hardware em três arquiteturas diferentes: *Paralela*, *Semi-paralela* e *Serial*, fazendo uso de operadores em ponto flutuante de 27 bits. Uma comparação entre as três arquiteturas em termos de precisão nos cálculos, consumo de energia, frequência máxima de operação, quantidade de recursos empregados e tempos de execução, permitiu escolher a melhor entre elas, para ser implementada no protótipo do sistema de detecção de quedas.

A arquitetura escolhida e o avaliador de limiares foram mapeados em FPGA na placa de desenvolvimento Arty-Z7 (chip xc7z020clg400-1 da Xilinx) operando a uma frequência de 83.33 MHZ. Para a avaliação do sistema de detecção, participaram cinco voluntários, os quais fizeram 100 quedas e 225 atividades da vida diária. Os resultados mostraram que o sistema proposto obtém um 94,0% de sensibilidade, um 92.4% de especificidade e um 92,9% de precisão, os quais permitem demostrar que os algoritmos de detecção de quedas implementados no FPGA possuem uma boa capacidade de generalização, a qual possibilita a detecção de quedas independentemente das características físicas dos usuários.

# ABSTRACT

Accidents such as falls can occur during a person's daily activities, being one of the main sources of morbidity and mortality in the elderly population. Real-time detection of falls and their communication to an emergency center, caregiver or family member can enable rapid medical assistance, thus increasing the sense of safety and reducing some of the negative consequences of falls. The present work describes the development of a wearable fall detection system based on an acceleration sensor and a microphone placed on the user's wrist, hardware architectures of the fall detection algorithm implemented in an FPGA and a Bluetooth device in charge of sending the alarm generated by the algorithm to an Android-based application running on a smartphone, which sends an alert message to a previously established contact.

The fall detection algorithm is based on an artificial neural network and a sound threshold evaluator. Firstly, acceleration signals of simulated falls were acquired at 125 Hz using a set of movements performed by 12 volunteers, composed of 170 falls and 170 activities of the daily living. A pattern composed of 138 acceleration samples was identified and along with a filtering process, time discretization and signal normalization, allows an artificial neural network to be trained in Matlab. On the other hand, the experimental analysis demonstrated that the neural network may erroneously classify some movements of daily life as a fall. In order to overcome this problem, a simple sound threshold evaluator was implemented. It is in charge of evaluating two thresholds (upper and lower limits) of the sound signal and when reached and at the same time the neural network detects the fall, then the system generates an alarm.

After the design of the fall detection algorithm. The artificial neural network was mapped on hardware in three different architectures: *Parallel*, *Semi-parallel*, and *Serial* making use of 27-bit floating-point arithmetic and trigonometric operators. A comparison of the three architectures in terms of accuracy, power consumption, maximum operating frequency, number of resources employed and execution times allows the best solution to be chosen in order to be implemented in the fall detection system prototype.

The chosen architecture and the threshold evaluator were mapped on a FPGA Arty-Z7 development board (chip xc7z020clg400-1 from Xilinx), operating at a frequency of 83.33 MHz. The results showed that the proposed system obtains 94.0% sensitivity, 92.4% specificity and 92.9% accuracy. Which demonstrated the generalization capacity of the proposed fall detection algorithms and enables the fall detection regardless of the physical characteristics of the users.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E ACROGRAMAS

| | |
|---|---|
| ADC | Conversor Analógico Digital |
| ADL | Atividades da Vida Diária |
| AMBA | Barramento de Interconexão Avançada do Microcontrolador |
| AP SoC | *All-Program-System-on-Chip* |
| APU | Unidade de Processamento de Aplicações |
| ASIC | Circuito Integrado de Aplicação Específica |
| DSP | Processadores Digitais de Sinal |
| FF | *Flip-Flops* |
| FPGA | *Field Programmable Gate Arrays* |
| FSM | Máquina de Estados Finita |
| GPIO | Entradas/Saídas de Proposito Geral |
| HDL | Linguagem de Descrição de Hardware |
| Hz | Hertz |
| I2C | Inter-Integrated Circuit |
| IEEE | Institute of Electrical and Electronics Engineers |
| IMU | Unidade de Medição Inercial |
| LEIA | Laboratory of Embedded Systems and Integrated Circuits Applications |
| LMS | *Least Mean Square* |
| LUT | *Look-up Table* |
| mA | miliAmpères |
| $MA_{cc}$ | Magnitude da Aceleração |
| MATLAB | Matrix Laboratory |

| | |
|---|---|
| MHz | Megahertz |
| MLP | Perceptron Multicamadas |
| MSE | Erro Quadrático Médio |
| SCL | *Serial Clock* |
| SDA | *Serial Data* |
| SoC | System on Chip |
| SIPO | *Single-input parallel-output* |
| SPI | *Serial Peripheral Interface* |
| PCB | Placa de Circuito Impresso |
| PLD | *Programmable Logic Devices* |
| UTFPR | Universidade Tecnológica Federal de Paraná |
| VHDL | *Hardware Description Language* |

# Capítulo 1

# Introdução

Atualmente, as quedas são a segunda causa de morte por lesões acidentais ou involuntárias a nível mundial. Calcula-se que 646.000 pessoas morrem anualmente e o risco e a frequência aumentam com a idade e o nível de fragilidade do indivíduo. Por este motivo, pessoas com mais de 65 anos são os que mais sofrem quedas fatais [1]. No Brasil, segundo dados do Sistema de Informação Médica/Ministério da Saúde, entre os anos de 1979 e 1995, cerca de 54.730 pessoas morreram devido a quedas, sendo que 52% delas eram idosos [2]. Em contraste com o anterior, se tem que cerca do 30% dos idosos caem ao menos uma vez ao ano [3].

A queda é o deslocamento do corpo inadvertidamente e de repente para um plano inferior, em relação à presença de um ou vários fatores [4]. Esta pode ocorrer durante as atividades diárias de uma pessoa em qualquer idade, no entanto, as mais propensas a sofrer este tipo de acidente são os idosos, convertendo-a numa importante fonte de morbidade e mortalidade entre eles [5].

Uma queda pode estar associada a fatores de risco intrínsecos e extrínsecos. Os primeiros correspondem a alterações fisiológicas do processo de envelhecimento, medicamentos, patologias específicas como cardiovasculares, neurológicas, endócrino-metabólicas, pulmonares e distúrbios psiquiátricos. Por sua vez, os extrínsecos, correspondem a fatores ambientais, iluminação inadequada, superfícies escorregadias, tapetes soltos e ou com dobras, degraus altos ou estreitos, ausência de corrimãos em corredores e banheiros, entre outros [6], [7], [8]. O anterior pode levar a uma pessoa bater o chão ou outra superfície firme, obtendo assim ferimentos que podem ser fatais, embora a maioria deles não seja.

As principais causas de admissão hospitalar relacionadas com as quedas no Reino Unido são fratura do quadril, lesões na medula espinhal, trauma crânio-encefálico e lesões dos membros superiores [9]. Similarmente no Brasil a maior incidência são lesões de superfície, seguido pelo trauma crânio-encefálico e trauma de membros inferiores [10]. Mesmo na ausência de lesões, as quedas podem ter consequências psicológicas a longo prazo, como depressão, medo de cair e perda de confiança. Estes, por sua vez, levam a restrições nas atividades diárias e sociais, e subsequentemente, diminuirão a saúde e aumentarão o risco de futuras quedas [11], [8], [12].

Nem todas as pessoas idosas que sofrem uma queda acabam se lesionando. Mas o aumento desta população durante as últimas décadas faz com que a quantidade de lesões induzidas por quedas, esteja num crescimento rápido em muitas nações [1], [13]. É por isso que a detecção de uma queda e a diminuição

do tempo de resgate ou atendimento se tem convertido num desafio, e as pesquisas sobre sistemas para a detecção automática de quedas têm aumentado [14].

## 1.1 Definição do problema

Apesar de extensos esforços preventivos, a queda continua sendo um problema de saúde muito importante. Por este motivo, o projeto de sistemas eficientes de detecção de queda que permitam agilizar o atendimento médico têm-se tornado quase obrigatórios, dado que as quedas geram problemas físicos às pessoas, diminuem a capacidade que têm para cuidar de si mesmas e participar de atividades físicas e sociais. Isto acaba por gerar um gasto para as entidades governamentais quando a situação do paciente se torna mais crítica ao não ser atendido a tempo [15].

Na literatura, existem várias propostas e aplicações bem-sucedidas que apontam para o desenvolvimento de sistemas de monitoramento que permitem a identificação de quedas. Esses sistemas garantem a ajuda rápida e eficiente quando esse tipo de evento ocorre, minimizando as consequências fatais [16]. Alguns desses sistemas empregam dispositivos portáteis como acelerômetros, giroscópios e magnetômetros que são usados para coletar informações cinemáticas da pessoa monitorada e assim determinar se aconteceu uma queda ou não utilizando um determinado algoritmo. No entanto, verifica-se que a localização dos dispositivos na cabeça, peito, perna ou cintura, geram certo nível de desconforto, costumam afetar a portabilidade e, consequentemente, a adoção deste tipo de tecnologia vestível. Embora existam outros estudos feitos utilizando o dispositivo no pulso da pessoa, melhorando a sensação de conforto, esses sistemas não conseguem atingir uma boa precisão devido aos diversos movimentos que o braço pode realizar durante as atividades diárias [17]. Por outro lado, a utilização de mais de um dispositivo vestível para detecção de quedas, implica o emprego de um maior tempo para processamento de algoritmos complexos, sendo necessário o uso de uma maior quantidade de *hardware* e consequentemente mais espaço para instalar os mesmos e um maior custo de desenvolvimento.

Com o objetivo de realizar a detecção de uma queda, oferecendo segurança, precisão e conforto para o usuário, neste trabalho se propõe implementar um sistema que empregue um acelerômetro e microfone colocados no pulso.

## 1.2 Justificativa

Com o constante crescimento da população idosa, se faz necessário ajudá-las para que tenham uma melhor qualidade de vida e possam viver de uma maneira independente, assim como diminuir a morbidade e mortalidade causada pelas quedas.

Na atualidade existem dispositivos que cumprem de uma maneira parcial com o objetivo de detectar uma queda, mas se faz necessário que além de detectá-la e comunicá-la eficientemente, atinjam requisitos de portabilidade (tamanho, peso e consumo de energia) e conforto para o usuário.

Diversos sensores são utilizados para a detecção de quedas, sendo o mais comum e importante entre os trabalhos que visam uma implementação portável, a utilização de um acelerômetro. Este sensor fornece

valores de aceleração dos diversos movimentos realizados pelos usuários, permitindo a caracterização e classificação de cada um deles. No entanto, estes dados não são suficientes para distinguir uma queda de uma atividade da vida diária (ADL), devido a que alguns movimentos ADL possuem as mesmas caraterísticas que as quedas, gerando uma classificação errada na detecção. Para diminuir a porcentagem do erro é utilizado um sensor adicional, neste caso um microfone que captura o som produzido durante a queda, permitindo melhorar o desempenho na detecção da mesma.

Um FPGA (*Field Programmable Gate Arrays*) possui uma grande flexibilidade para desenvolver diferentes projetos com arquiteturas que precisam ser modificadas e testadas antes de se fazer uma produção em série da solução criada. Com estas caraterísticas e em conjunto com redes neurais artificiais, que têm qualidades de generalização e tolerância a falhas nos dados de entrada, e que podem ser implementadas em arquiteturas paralelas nestes dispositivos, se visa alcançar os requisitos para uma detecção de quedas precisa e eficiente em um dispositivo compacto que permita reconfigurar a estrutura da rede neural artificial quando novos treinamentos são realizados.

## 1.3 Objetivos

### 1.3.1 Objetivo geral

Desenvolver um protótipo funcional para a detecção de quedas a partir do monitoramento da aceleração e do som produzidos pelos movimentos de uma pessoa.

### 1.3.2 Objetivos específicos

Para atingir o objetivo geral no presente trabalho, foram estabelecidos os objetivos específicos a seguir:

- Construir um sistema para a aquisição de dados de aceleração e de som.
- Desenvolver um sistema de detecção de quedas baseado nos dados de aceleração e som.
- Mapear os algoritmos de detecção de quedas em hardware reconfigurável.
- Desenvolvimento de um aplicativo Android para comunicação das quedas aos organismos de socorro ou familiares do usuário e a captura dos respectivos sinais de aceleração e som.

## 1.4 Aspectos metodológicos

Para obter conhecimentos sobre as pesquisas relacionadas a este trabalho foi realizada de maneira sistemática uma revisão da literatura existente, extraindo a informação mais relevante, como ferramentas, dispositivos e técnicas empregadas. Esta é feita a partir de fontes bibliográficas de artigos científicos publicados em jornais, revistas e congressos na área de engenharia eletrônica e biomédica.

Este trabalho é uma pesquisa aplicada, pois serão gerados conhecimentos sobre a aplicação prática das redes neurais artificiais na detecção de quedas de pessoas, onde os dados têm um pré-processamento antes de serem avaliados nela. É uma pesquisa experimental quanto ao procedimento porque determina os dados de treinamento da rede neural, estabelece os critérios a ser levados em consideração para a detecção de uma queda e para a análise dos resultados obtidos.

O resultado final desta pesquisa culminou na construção de um protótipo de um sistema de coleta de dados inerciais que emprega uma rede neuronal artificial para classificar os movimentos de atividades da vida diária e de quedas. A rede neural artificial projetada foi avaliada mediante os dados capturados com o protótipo. Inicialmente, esta avaliação foi feita no Matlab, visando a minimização do erro de detecção de quedas. Foi feita uma análise dos picos de som obtidos com o microfone, permitindo estabelecer limiares para determinar se aconteceu um impacto no chão ou um grito, com o intuito de diminuir os falsos positivos da detecção. Em seguida, a rede neural artificial e o avaliador de limiares foram mapeados em um SoC (*System on Chip*) Zynq da Xilinx e em Arduino, permitindo realizar uma comparação numérica do tempo de execução entre ambas soluções embarcadas. Por último o sistema de detecção de quedas no SoC foi avaliado com a ajuda de usuários voluntários fazendo uso de um aplicativo Android para registrar a queda. Todos os resultados obtidos neste trabalho foram analisados e comparados com trabalhos correlatos desenvolvidos por outros autores.

## 1.5 Contribuições do trabalho

As contribuições relevantes do trabalho atual são:

- Um dispositivo para capturar os dados inerciais e de som dos movimentos das usuários, que pode ser utilizado em diferentes partes do corpo;
- Estudo dos movimentos e comportamento da força de gravidade presente no pulso de uma pessoa assim como a caraterização do som de uma queda;
- Desenvolvimento de uma nova estrategia de extração de características e redução de dados a serem processados por um algoritmo, a qual não esta presente na literatura;
- Desenvolvimento de um sistema embarcado de detecção de quedas considerando a magnitude da aceleração e o som produzido por uma pessoa caindo;
- Criação de um repositório de dados de quedas simuladas e movimentos da vida diária;
- Desenvolvimento de um aplicativo Android para a comunicação da queda às pessoas cadastradas pelo usuário (organismos de socorro e familiares).
- XXVI Congresso Brasileiro de Engenharia Biomédica no qual foram apresentados os resultados alcançados pelo algoritmo de classificação de quedas baseado em redes neurais artificiais perceptron multi-camadas (vide Apêndice A).
- Uma publicação que contém o desenvolvimento do sistema embarcado no SoC FPGA foi realizada e será submetida para o Journal IEEE Sensors.

## 1.6 Apresentação do documento

O conteúdo do presente trabalho encontra-se divido em 7 capítulos da seguinte maneira:

- O capítulo 2 aborda a fundamentação teórica empregada neste trabalho;
- O capítulo 3 expõe os diferentes trabalhos correlatos que implementam dispositivos vestíveis;
- O capítulo 4 descreve como foi realizada a adquisição de dados para o desenvolvimento do sistema de detecção de quedas;
- O capítulo 5 mostra o tratamento e classificação dos dados coletados segundo os padrões encontrados, a definição das entradas da rede neural, seu treinamento e a determinação dos picos de som;
- O capitulo 6 descreve a implementação em arquitetura embarcada (FPGA) da rede neural artificial e o avaliador de limiares de som para a detecção de quedas desenvolvidas no capítulo anterior;
- O capitulo 7 apresenta os resultados obtidos quanto a quantidade de recursos de *hardware* empregados, a velocidade de processamento e a precisão numérica nos cálculos da rede neuronal e a detecção de quedas;
- Finalmente as conclusões do trabalho são apresentadas no capitulo 7.

# Capítulo 2

# Fundamentação Teórica

Neste capítulo é apresentada a fundamentação teórica associada ao desenvolvimento do presente trabalho. Conceitos como redes neurais artificiais e seus benefícios, hardware reconfigurável e a métrica utiliza para a avaliação do desempenho do sistema de detecção de quedas, são abordados.

## 2.1 Rede neural artificial

Através da experiência o cérebro tem a capacidade de desenvolver suas próprias regras para fazer uma determina tarefa, onde a experiência vai sendo acumulada com o tempo e processada pelos neurônios. De forma geral uma rede neural artificial é uma máquina projetada para modelar a maneira como o cérebro realiza uma tarefa específica ou função de interesse. Seu funcionamento é baseado na interligação entre as unidades de processamento, ou neurônios, para alcançar um bom desempenho [18]. Uma rede neural artificial pode ser implementada utilizando-se componentes eletrônicos ou simulada em um computador.

As redes neurais artificiais são utilizadas em diversos campos da engenharia, entre eles se destaca o processamento de sinais, controle, análise de séries temporais e reconhecimento de padrões. Segundo Haykin [18], uma rede neural é um processador paralelamente distribuído constituído por unidades de processamento simples e interligadas, que tem a capacidade natural para armazenar conhecimento e torná-lo disponível para uso. Ela se assemelha ao cérebro em dois aspectos: (a) o conhecimento do ambiente é adquirido pela rede através de um processo de aprendizagem e; (b) as forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento [19].

### 2.1.1 Neurônio

O neurônio é a unidade fundamental de processamento de informação de uma rede neural artificial, cuja estrutura é mostrada na Figura 2.1. O neurônio é constituído por um conjunto de entradas (que representam sinapses vindo de neurônios anteriores) as quais são ponderadas por um peso e acumuladas através de um somador. A saída do somador é modificada pelo *bias* e este novo resultado é avaliado na função de ativação que restringe a amplitude do sinal de saída [18].

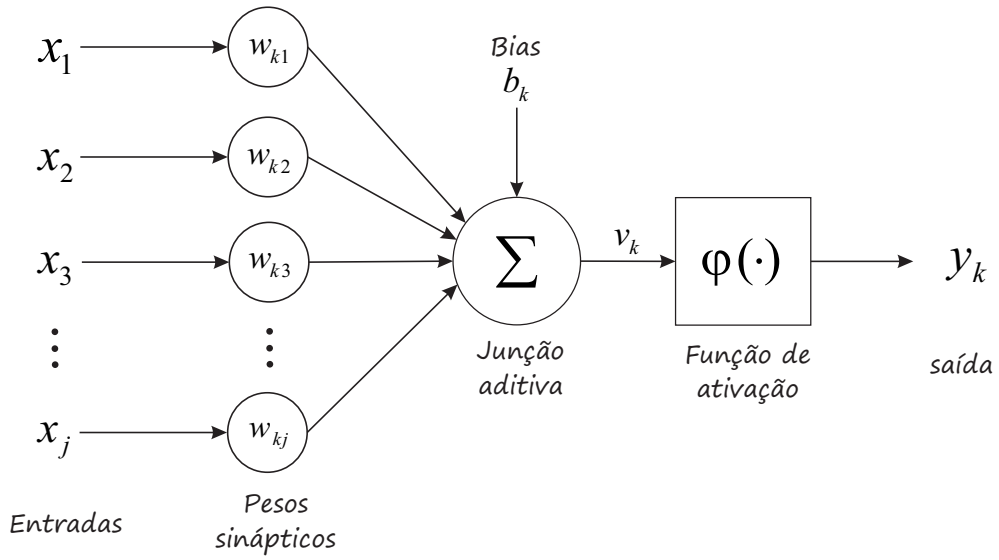



Figura 2.1: Estrutura básica de um neurônio.
Fonte: Adaptado de [19].

Em termos matemáticos um neurônio pode ser descrito da seguinte forma:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

onde $w_{k1}, w_{k2}, ..., w_{kj}$ são os pesos sinápticos do neurônio $k$, $x_1, x_2, ..., x_j$ são os sinais de entrada e $u_k$ é o resultado do somador ou combinador linear das entradas ponderadas por um peso,

$$v_k = u_k + b_k, \tag{2.2}$$

Na Equação 2.2, $u_k$ é somado com o *bias* $b_k$, que tem o efeito de aplicar um valor de *offset* dependendo se o *bias* $b_k$ é positivo ou negativo. A relação entre o campo local induzido $v_k$ do neurônio $k$ e a saída do combinador $u_k$ é modificada como pode ser visto na Figura 2.2.

Por último, na Equação 2.3 o $v_k$ é avaliado na função de ativação $\varphi(\cdot)$ tendo como resultado o sinal de saída desejada para o neurônio $y_k$.

$$y_k = \varphi(v_k) \tag{2.3}$$

### 2.1.2 Função de ativação

A função de ativação $\varphi(\cdot)$ é escolhida durante o projeto da rede neural de acordo com o tipo de saída para o neurônio [20]. Apresenta-se a seguir quatro tipos de funções de ativação mais utilizadas:

1. Função linear

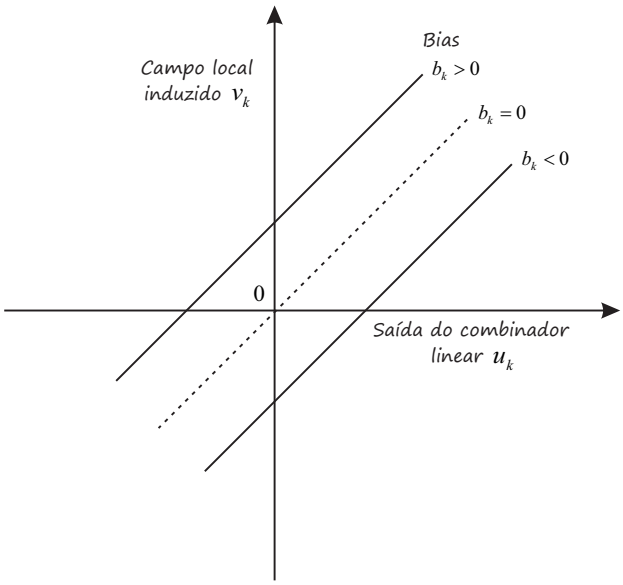



Figura 2.2: Transformação afim produzida pelo *bias*.
Fonte: Adaptado de [19].

$$\varphi(v_k) = v_k \tag{2.4}$$

2. Função degrau

$$\varphi(v_k) = \begin{cases} 1 & \varphi(v_k) > 0 \\ \\ 0 & \varphi(v_k) \leq 0 \end{cases} \tag{2.5}$$

3. Função sigmoide

$$\varphi(v_k) = \frac{1}{1 + e^{-v_k}} \tag{2.6}$$

4. Função tangente hiperbólica

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.7}$$

A representação gráfica destas funções de ativação é mostrada na Figura 2.3. Dentre as funções descritas, as mais utilizadas em tarefas de classificação são a função sigmoide e tangente hiperbólica, que com suas características de não linearidade conseguem resolver problemas complexos.

## 2.2 Arquitetura de uma rede neural artificial

A arquitetura de uma rede neural representa o padrão de conexões entre os neurônios e a propagação de dados entre eles [19]. Em geral, existem três classes de arquiteturas da rede:

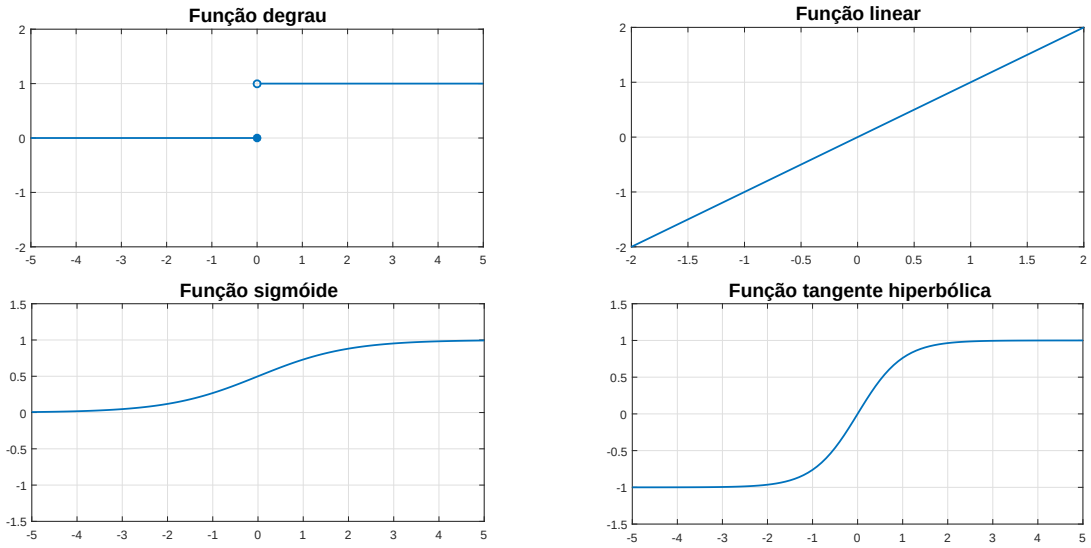



Figura 2.3: Funções de ativação.

1. ***Feedforward* de camada simples:** Também conhecida como rede com camada única, é a forma mais simples de uma rede, possuindo apenas as entradas que funcionam como um caminho que modificam os dados que entram na rede e os transfere à camada de neurônios da saída como se pode observar na Figura 2.4. Além disso, a informação só circula numa direção a partir da entrada para a saída, por isso elas são chamadas de *feedforward* [20]. São empregadas em problemas de classificação de padrões e filtragem.

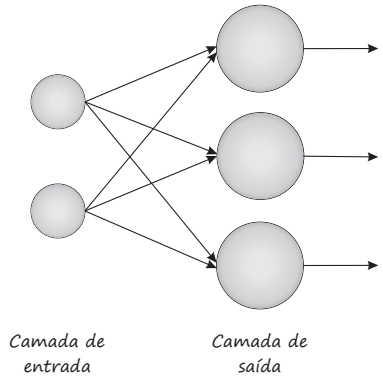



Figura 2.4: Rede neural *feedforward* de camada simples.

2. ***Feedforward* de camadas múltiplas**: Caracteriza-se pela presença de uma ou mais camadas ocultas como é mostrado na Figura 2.5. Nesta topologia os neurônios ocultos intervêm entre a entrada e a saída da rede [18]. Como no caso anterior, a informação flui em apenas uma direção, desde a entrada, passando pelas camadas ocultas e finalmente para a camada de saída, gerando assim uma conectividade total entre as camadas [20]. São empregadas em problemas de aproximação de funções, classificação de padrões, identificação de sistemas, otimização, robótica e controle de processos.

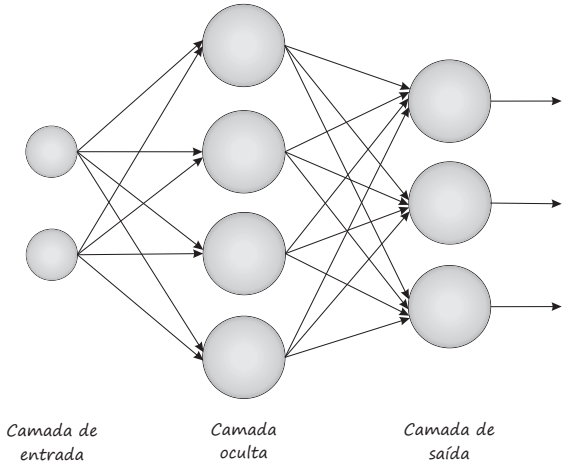



Figura 2.5: Rede neural de camadas múltiplas.

3. **Recorrente ou realimentada:** Tem o mesmo funcionamento que a feedforward, mas suas saídas são realimentadas como sinais de entrada para alguns neurônios [18]. São empregadas para o processamento de sistemas variantes no tempo, identificação de sistemas e controle de processos. Sua arquitetura é mostrada na Figura 2.6.

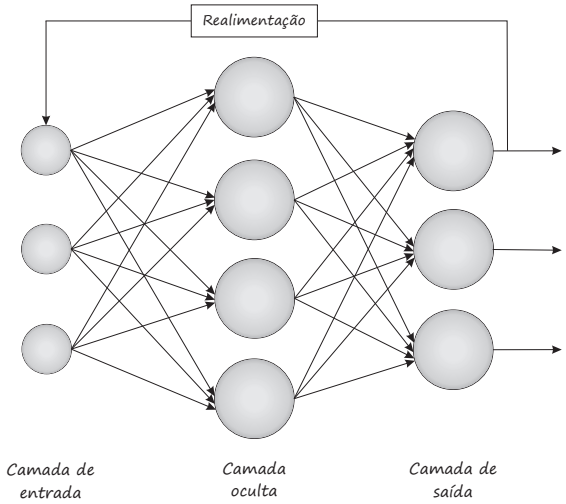



Figura 2.6: Rede neural realimentada.

## 2.3 Processo de aprendizagem

O processo de aprendizagem de uma rede neural é chamado de algoritmo de aprendizagem. A função destes algoritmos é modificar os pesos sinápticos e *bias* da rede de uma maneira ordenada, de tal forma que quando são inseridos um conjunto de entradas, serão obtidos valores desejados de saída [19].

A definição dos pesos das redes neurais pode ser feita de duas maneiras. A primeira é a definição dos pesos explicitamente, usando um conhecimento a priori. O segundo, é treinando a rede, entregando-lhes

padrões e recalculando os pesos de acordo com alguma regra de aprendizagem [18]. Partindo dos métodos anteriores, definem-se dois paradigmas de aprendizagem: o supervisionado e o não supervisionado.

### 2.3.1 Aprendizagem não supervisionada

Na aprendizagem não supervisionada, como o nome indica não há um *"professor"* para supervisionar o processo de aprendizagem. Por conseguinte, não existe um conjunto prévio de categorias nas quais os dados de entrada devem ser classificados, em vez disso, o sistema deve desenvolver suas próprias respostas aos estímulos de entrada e encontrar alguma estrutura para a organização dos dados [18].

### 2.3.2 Aprendizagem supervisionada

Na aprendizagem supervisionada ou aprendizagem com um *"professor"*, os ajustes dos parâmetros do sistema são baseados em um vetor de treinamento e um sinal de erro que é definido como a diferença entre a resposta desejada e a resposta real da rede. Os ajustes são realizados iterativamente com o objetivo de fazer com que a rede neural emule o *"professor"*, transferindo o conhecimento do ambiente para a rede neural. O processo de treinamento termina quando o sinal de erro é o menor possível. Neste ponto, a ajuda do *"professor"* não é mais necessária, e a rede neural poderá lidar com o ambiente por si mesma [19].

## 2.4 Perceptron multicamadas (MLP)

É um tipo de rede *feedforward* de camadas múltiplas, possui um conjunto de entradas que constituem a camada de entrada, uma ou mais camadas ocultas e uma camada de saída de neurônios ou nós computacionais, que processam os dados de entrada e aplicam os pesos sinápticos para gerar a saída desejada [18].

As redes perceptron multicamadas são utilizadas para resolver diversos problemas complexos, como aqueles que não são linearmente separáveis. O algoritmo de treinamento geralmente utilizado para este tipo de rede é do tipo supervisionado, chamado *back-propagation* ou retropropagação. Este algoritmo é baseado na regra de aprendizagem por correção de erro e também considerado como uma generalização do algoritmo de mínimos quadrados (LMS, *Least Mean Square*), utilizado em redes de um único neurônio linear [19].

### 2.4.1 Back-Propagation

A aprendizagem por retropropagação de erro é dívida em dois passos: um passo para frente denominado propagação e outro para trás, a retropropagação. Na propagação um vetor pertencente ao ambiente é inserido na rede, este se propaga através dela até a saída, gerando uma resposta real da rede. Ao longo desse passo, os pesos sinápticos da rede são fixos [18]. Durante a retropropagação, os pesos sinápticos da rede são ajustados de acordo com a regra de correção de erro, onde o sinal de erro é obtido da resposta real da rede e a subtração da resposta esperada. Este sinal de erro se propaga para trás, ajustando os pesos

sinápticos no intuito de que a resposta da rede fique próxima da resposta desejada. Depois de alcançar um número de iterações ou um valor de erro previamente estabelecido é possível obter os pesos sinápticos otimizados [19].

## 2.5 Propriedades das redes neurais artificiais

O poder computacional de uma rede neural está centrado na sua estrutura paralela e capacidade de aprender e generalizar. A capacidade de generalizar é obtida pela rede, quando esta pode obter respostas adequadas a vetores de entrada que não foram utilizados durante a fase de treinamento. Por conseguinte, o processamento de informação por estas duas capacidades faz com que a rede neural consiga resolver problemas complexos que são intratáveis através de métodos analíticos [19].

As redes neurais artificiais podem apresentar as seguintes propriedades e características:

1. Não linearidade. Os neurônios artificiais podem ter características lineares ou não-lineares. Uma rede neural composta por neurônios não-lineares, é também não-linear. A não-linearidade é uma propriedade muito importante, principalmente nos casos em que a aplicação envolve dados de entrada com uma natureza não-linear.

2. Mapeamento de entrada e saída. O processo de aprendizagem supervisionado neste trabalho, realiza a modificação dos pesos sinápticos da rede, baseado num conjunto de vetores de entrada e de saída, chamados amostras de treinamento. Cada amostra de entrada possui um sinal de saída desejada. Esses dados são inseridos na rede neural a fim de modificar os pesos sinápticos com o intuito de minimizar a diferença entre a resposta desejada e a resposta real da rede. A partir da realização desse processo, cria-se um mapeamento de entrada e saída da rede, que permitem obter modelos matemáticos que representam o comportamento de um sistema específico.

3. Adaptabilidade. É a capacidade da rede de adaptar os pesos sinápticos a modificações do ambiente em que a rede foi aplicada. Isso significa que uma mesma arquitetura de uma rede neural pode aprender diferentes problemas modificando seus pesos sinápticos para poder operar em um novo ambiente.

4. Tolerância a falhas. Uma rede neural é tolerante a falhas nos dados (distorções, ruído, dados incompletos) devido ao fato de que a rede não armazena o conhecimento de uma maneira localizada, mas sim de maneira distribuída por toda sua estrutura.

## 2.6 Hardware reconfigurável

Existem muitos chips que compõem alguns circuitos lógicos, chamados chips padrões, como a série 7400, que contém apenas algumas portas lógicas simples e que raramente são usados hoje em dia. Estes eram populares para a construção de circuitos lógicos até o início dos anos 80. No entanto, à medida que a tecnologia dos circuitos integrados melhorou, tornou-se ineficiente usar o espaço valioso da placa de circuito impresso (PCB, *Printed Circuit Board*) para chips com poucas funcionalidades [21]. Devido à

situação descrita, durante a década de 80, foram criados os sistemas em chip chamados circuitos integrados de aplicação específica (ASICs, *Application-Specific Integrated Circuits*). Estes eram projetados para realizar tarefas específicas, obtendo um projeto mais otimizado que geralmente leva a um melhor desempenho conseguindo incluir uma maior quantidade de circuitos lógicos num chip. O custo de produção é alto para uma quantidade pequena deste tipo de chips e sua fabricação geralmente consome um tempo considerável, além disso, a funcionalidade do chip é fixa e não pode ser alterada, convertendo-se num problema para projetos que precisam de flexibilidade. A partir disso, são criados novos chips onde o usuário pode configurar os circuitos de acordo as suas necessidades, com o fim de fazer uma tarefa específica. Estes chips são conhecidos como dispositivos lógicos programáveis (PLD, *Programmable Logic Devices*), e a maioria destes podem ser reconfiguráveis via software em repetidas ocasiões. Isto é uma vantagem para o projetista que desenvolve o protótipo de um produto, já que pode programar um PLD para cumprir uma tarefa e, mais tarde, quando o protótipo é colocado à prova, fazer correções reconfigurando-o, empregando linguagens de descrição de hardware como Verilog e VHDL. Os PLDs estão disponíveis em diferentes tamanhos. Pelos motivos apresentados anteriormente, eles podem satisfazer os requisitos de uma aplicação específica e hoje em dia estão sendo amplamente utilizados em diversos tipos de projetos [22].

### 2.6.1 FPGAs (*Field Programmable Gate Arrays*)

O arranjo de portas programáveis em campo (FPGA) é um dos tipos mais modernos de PLD [2]. O chip em silício é constituído por uma matriz de blocos lógicos configuráveis (CLBs, *Configurable Logic Blocks*) cercados por uma matriz de blocos de entradas e saídas (I/O) e interconectados entre si mediante blocos de conexão (CB, *Connection Boxes*) e blocos de chaveamento (SB, *Switch Boxes*) [23].

Os CLBs são circuitos idênticos, que fornecem a lógica básica de uma FPGA, são constituídos por *slices* interconectados localmente. Os slices estão conformados por *Look-Up Tables* (LUTs), *Flip-Flops* (FF) e multiplexadores, onde as LUTs são tabelas de verdade que representam equações booleanas básicas. As conexões entre os CLBs são feitas mediante os blocos de conexão e chaveamento, o qual permite criar operações mais complexas ao configurar estes blocos [23], [24]. A arquitetura básica de uma FPGA é apresentada na Figura 2.7.

Na Figura 2.7 foram apresentados os blocos lógicos básicos de uma FPGA, mas na atualidade muitas FPGAs contêm diferentes blocos, alguns dos quais só podem ser usados para propósitos específicos, tais como: memórias RAM, multiplicadores, somadores, blocos de processamento digital de sinal (DSP), microprocessadores, conversores analógico-digital (ADC), elementos de geração de sinais de *clock*, entre outros.

Existem também os AP SoC (*All-Program-System-on-Chip*) Zynq-7000 da Xilinx, que integra um processador *dual-core ARM Cortex A9* e uma lógica programável (FPGA). A arquitetura da Zynq APSoC apresentada na Figura 2.8, está dividida em dois subsistemas: o sistema de processamento (PS) em verde claro e a lógica programável (PL) em amarelo. O PL está constituído por DSPs, RAM, ADC, várias portas e barramentos dedicados para comunicar-se com o PS, o qual está formado por muitos componentes, incluída uma unidade de processamento de aplicações (APU) que compreende dois processadores Cortex A9, o barramento de interconexão avançada (AMBA, *Advanced Microcontroller Bus Architecture (AMBA) Interconnect*), o controlador de memória DDR3 e vários controladores periféricos com suas entradas e

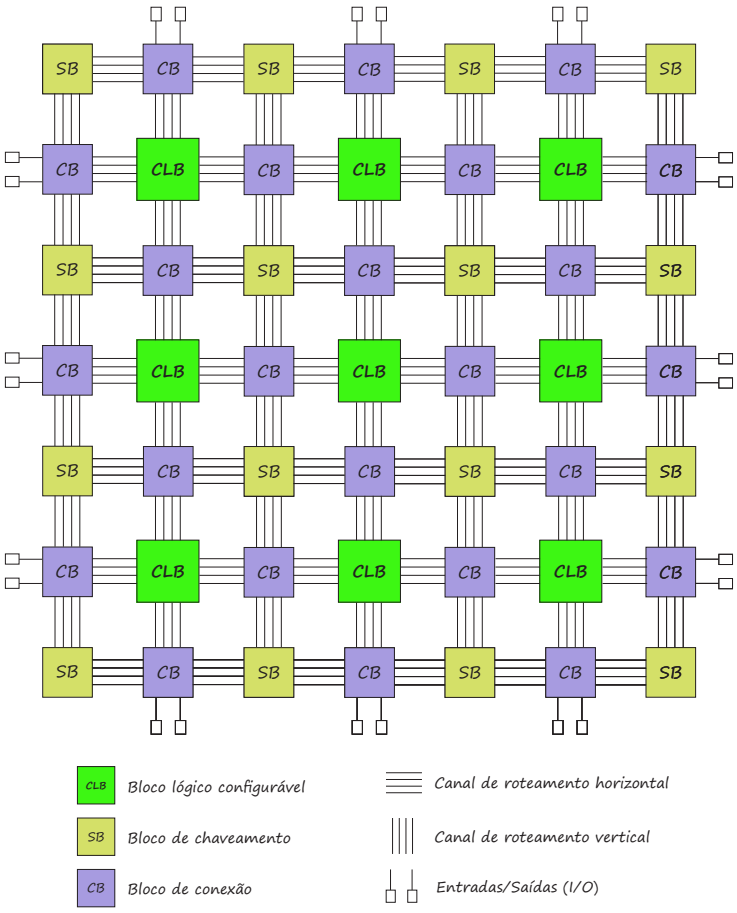



Figura 2.7: Arquitetura de uma FPGA.
Fonte: Adaptado de [25].

saídas multiplexadas a 54 pinos. Como pode ser visto na Figura 2.8, a Zynq APSoC é projetada em torno do processador, que atua como o mestre da estrutura lógica programável e de todos os outros periféricos no chip do sistema de processamento, o que torna o processo de inicialização da Zynq mais semelhante ao de um processador do que um FPGA [26].

### 2.6.2 Metrica para avaliar o desempenho do sistema

O sistema de detecção de quedas podem ser considerado um classificador binário ou binomial, o qual classifica os elementos de um conjunto de dados em dois grupos, predizendo a que grupo pertence um dado valor a partir de uma regra de classificação. Para este tipo de função, realizam-se medidas estatísticas de sensibilidade e especificidade, que permitem determinar o nível de precisão da função. Este método de avaliação é amplamente utilizado nos sistemas desenvolvidos para a detecção de quedas [27], [28], [29], [30]. Logo, com o intuito de avaliar o algoritmo de detecção de queda proposto, podem-se observar quatro situações possíveis: verdadeiro negativo (TN), movimento de não queda, classificado corretamente; falso positivo (FP), são as atividades normais que causam um alarme falso; verdadeiro positivo (TP) que são as quedas corretamente classificadas; e falso negativo (FN) que são as quedas que não foram detectadas pelo sistema. Com essas situações três índices são gerados: sensibilidade, especificidade e precisão.

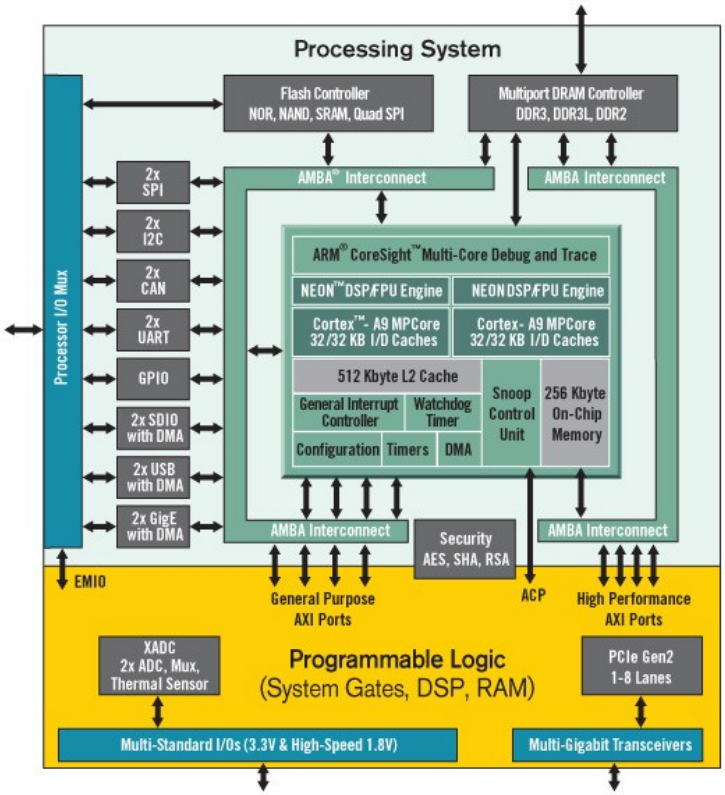



Figura 2.8: Arquitetura *Zynq APSoC*.
Fonte: Adaptado de [26].

A sensibilidade (SE) descrita pela Equação 2.8, é a capacidade de detectar quedas reais. Esta equação é definida como a relação entre o número de quedas adequadamente detectadas (verdadeiros positivos) e as quedas que realmente aconteceram (verdadeiros positivos mais falsos negativos).

$$SE = \frac{TP}{TP + FN} \times 100\% \tag{2.8}$$

A especificidade (SP) apresentada na Equação 2.9, é a capacidade de filtrar falsos alarmes e corresponde à proporção entre os movimentos sem quedas devidamente classificados (verdadeiros negativos) e o número total de movimentos que não tem uma queda (verdadeiros negativos e falsos positivos).

$$SP = \frac{TN}{TN + FP} \times 100\% \tag{2.9}$$

A precisão (AC) da Equação 2.10, é a proporção de resultados verdadeiros no conjunto de dados considerado.

$$AC = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{2.10}$$

A partir dos índices gerados pode-se observar o desempenho do sistema classificador binário que será projetado neste trabalho.

# Capítulo 3

# Trabalhos Correlatos

Os sistemas para a detecção de quedas podem ser classificados em três categorias de acordo com as tecnologias empregadas: vigilância por vídeo, identificação baseada em áudio-vibração e dispositivos de detecção vestível.

Na área de processamento de imagens, os pesquisadores conseguem detectar um evento de queda a partir dos movimentos do alvo, os quais são capturados mediante câmeras instaladas num determinado lugar. O sistema determina as características das imagens através de um algoritmo que as processa, conseguindo distinguir entre as atividades de queda e da vida diária [31], [32], [33], [33]. [34], [35]. Por outro lado, os trabalhos relacionados com áudio e vibração, empregam microfones e acelerômetros, os quais são colocados nas paredes e chão de um quarto objeto de monitoramento onde ficará o usuário. Os algoritmos desenvolvidos fazem uma análise do sinal de áudio e da vibração gerada pelo impacto do corpo no chão, obtendo assim o origem e localização. Além disso, conseguem diferenciar os sons e vibrações geradas pelas quedas humanas, das geradas por objetos caindo no chão e das atividades diárias [31], [36], [37], [38], [39], [40], [41], [42]. A última categoria faz referência aos dispositivos que podem levados por uma pessoa, chamados de dispositivos vestíveis, os quais capturam os dados dos movimentos do corpo humano, medindo a posição do corpo com respeito a um eixo de referência, a velocidade e força com que cai. Sendo estes dispositivos considerados a principal tendência de pesquisa atualmente [16], [43].

Como se pode evidenciar nas duas primeiras categorias, o usuário é obrigado a permanecer na área onde são instalados os dispositivos de monitoramento, limitando seu cuidado e mobilidade. Por este motivo, neste trabalho optou-se pelo monitoramento mediante dispositivos vestíveis que permitirão ao usuário transitar por qualquer lugar. Dentro dos dispositivos vestíveis mais utilizados pelos pesquisadores estão os acelerômetros e giroscópios, que mediante algoritmos baseados em limiares, métodos analíticos e aprendizagem de máquina conseguem detectar uma queda.

Em vários estudos, por meio de experimentos realizados com diferentes pessoas, foram obtidos diversas características das quedas e estabelecidos os limiares de tempo, da magnitude da aceleração, os ângulos da postura do corpo, a velocidade angular, entre outros [12], [31], [42], [44], [45], [17], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55] [56], [57], [58], [59], [60], [61]. Nesses trabalhos a queda é detectada quando algum dos parâmetros atinge um valor pré-definido, chamado de limiar. Entretanto, alguns dos movimentos feitos durante as atividades diárias de uma pessoa podem-se confundir com quedas, apresentando

falhas na detecção devido a que os movimentos têm caraterísticas muito parecidas às de uma queda [46]. A partir do problema que têm os algoritmos baseados em limiares ao não poder distinguir muito bem entre movimentos da vida diária e quedas [14], além de requerer um ajuste dos parâmetros de cada limiar para um determinado usuário, os métodos de aprendizagem de máquina surgem como uma nova abordagem e possível solução ao problema, aumentando a precisão da detecção. A seguir apresenta-se uma descrição dos trabalhos que utilizam dispositivos vestíveis e redes neurais artificias para a detecção de quedas usando os sensores na cintura ou peito do usuário.

Dinh and Struck [27], descrevem um sistema de detecção de quedas, onde os dados de aceleração são convertidos de coordenadas cartesianas para coordenadas esféricas. O sistema está baseado em logica Fuzzy e uma rede neural MLP com 640 entradas, duas camadas escondidas, a primeira com 30 e a segunda com 20 neurônios e uma camada de saída com um neurônio. Os resultados obtidos mostram que para a detecção de quedas de maneira confiável é suficiente utilizar um único acelerômetro e que os métodos baseados nos sistemas neuro-fuzzy são uma opção adequada para o reconhecimento de padrões de quedas. Yuwono et al. [28], propõe um algoritmo de detecção de queda, com um único acelerômetro triaxial colocado na cintura de uma pessoa e aborda a detecção de queda usando dois classificadores diferentes: uma rede perceptron multicamada e uma rede neural de base radial conseguindo um resultado promissor. Sengto et al. [62], sugere um algoritmo que consiste em dois processos, em primeiro lugar, um simples processo de limiares, que é utilizado para distinguir os movimentos lentos e as outras atividades da vida diária de uma pessoa. Posteriormente, se o valor dos sinais for maior que os limiares, uma rede neural *backpropagation*, que conta com 300 entradas na camada inicial, 90 neurônios na camada escondida e por último a camada de saída com 8 neurônios, é utilizada para classificar oito atividades diferentes de uma pessoa, incluindo a queda. Vallejo et al. [29], apresenta um sistema de detecção de quedas composto por uma rede neural MLP como um método que pretende melhorar a precisão da detecção com respeito aos métodos tradicionais baseados em limiares. A rede foi implementada num dispositivo colocado na cintura da pessoa, que possui na camada de entrada 3 neurônios, as quais recebem o cálculo da integral da aceleração de cada eixo, duas camadas escondidas com 5 neurônios cada uma e um neurônio de saída. Kerdegari et al. [63], propõe um sistema de detecção de quedas a partir dos sinais de aceleração, do cálculo da velocidade angular, velocidade linear e posição do corpo, assim como de diferentes características no domínio do tempo, como mínimo, máximo, média, intervalo, variância e desvio padrão. Tais caraterísticas compõem as entradas de uma rede neural perceptron multicamada com a seguinte estrutura: na camada de entrada tem 28 neurônios, na camada oculta 6 neurônios e, por último, na camada de saída 1 neurônio, que permitirá a classificação das atividades diárias e as quedas. Zhang et al. [64], descreve a fusão sensorial do acelerômetro, giroscópio e uma bússola magnética, mediante um filtro de Kalman estendido para determinar as atitudes dos ângulos de arfagem (*pitch*) e rolagem (*roll*) que em conjunto com a média quadrática da aceleração formam as entradas de uma rede neural MLP com 2 neurônios na camada oculta e 1 na camada de saída. Teja et al. [30], mediante a implementação de uma rede neural MLP e uma máquina de vetores de suporte (SVM, *Support Vector Machine*) com base em 6 recursos extraídos da medição da velocidade angular e a aceleração nos três eixos, desenvolveram um sistema de detecção de quedas que é colocado nas costas e na cintura de uma pessoa. Vidigal et al. [65], apresenta uma implementação e comparação de desempenho das redes neurais MLP, RBF e Kohonen em aparelhos celulares, para a detecção de quedas, utilizando os dados fornecidos pelo acelerômetro de três eixos presente em *smartphones*. Cada rede neural tem 86 entradas, 173 neurônios na camada escondida e um de saída. Nuttaitanakul et al. [66], propõe um

novo algoritmo para detectar as quedas a partir do sinal de aceleração de um sensor colocado na cintura de uma pessoa, usando a transformada Wavelet e uma rede neural perceptron multicamada, obtendo uma precisão de 85% na classificação de atividades da vida diária e 100% nos experimentos com quedas. Yodpijit et al. [67], usam um acelerômetro e um giroscópio colocados na cintura da pessoa para detectar a orientação e o movimento do corpo. Os autores, implementaram um algoritmo baseado em limiar em conjunto com uma rede neural perceptron multicamada no intuito de diminuir a quantidade de falsos positivos e melhorar a precisão do sistema de detecção de quedas.

A seguir apresenta-se uma descrição dos trabalhos que utilizam dispositivos vestíveis colocados no pulso do usuário e algoritmos baseados em análise de limiares. Zhou et al. [68], apresentam um novo dispositivo de pulso para monitoramento da frequência cardíaca e detecção de quedas com baixo consumo de energia e baixo custo. Os resultados obtidos pelos autores mostram que a queda pode ser detectada com um limiar que é significativamente diferente dos movimentos de andar e levantar-se com uma precisão de 93,75%. Hsieh et al. [69], desenvolveu um dispositivo sensor composto por um acelerômetro de três eixos e um giroscópio de três eixos. O usuário usa dois dispositivos, um em cada pulso, os quais transmitem os dados coletados para um computador através do protocolo Zigbee onde são analisados, para determinar se os movimentos excedem os limiares estabelecidos e comunicar se o evento de queda ocorreu. De acordo com os resultados experimentais, eles obtiveram uma sensibilidade de 95% e uma especificidade de 96.7%. Zhou et al. [70], apresentam um algoritmo para detectar quedas em idosos, a partir de um dispositivo usado no pulso. O algoritmo desenvolvido tem baixa complexidade de cálculo para determinar o evento de queda de acordo com o gradiente de inclinação, definindo dois limiares: um para a queda e outro para o procedimento após da queda, obtendo uma sensibilidade de 94.44% e uma especificidade de 100%.

Quadros et al. [71], avaliam o desenvolvimento de uma solução de detecção de quedas usada no pulso, que emprega um acelerômetro, um giroscópio e um magnetômetro. Estes dados são combinados e analisados mediante um conjunto de métodos baseados em limiares e aprendizado de máquina usando diferentes tipos de redes neurais artificiais para definir a melhor abordagem para detectar uma queda. O melhor resultado foi obtido utilizando os métodos de decomposição de Madgwick e aprendizagem de máquina *k-Nearest-Neighbors* (k-NN), com uma precisão de 99.0%, 100% de sensibilidade e 97.7% de especificidade.

Todos os trabalhos discutidos anteriormente são resumidos na Tabela 3.1.

A implementação de sistemas de detecção de quedas em FPGAs têm sido pouco explorados. Da revisão realizada encontrou-se que Abdelhedi et al. [49], propõe um protótipo composto por um acelerômetro tri-axial colocado na cintura do usuário, que se comunica com uma placa Zybo (SoC Zynq da xilinx) através do protocolo de I2C, a qual tem embarcado o algoritmo baseado em limiares que detecta picos de aceleração e reconhece a postura de uma pessoa após a queda, conseguindo um 96,0% de sensibilidade e um 97,5% de especificidade. Saadeh et al. [72], apresenta um sistema de detecção de quedas desenvolvido a partir de uma base de dados criada com o acelerômetro de um celular, o qual é colocado no bolso das calças dos voluntários. A base de dados contém quatro tipos diferentes de quedas e nove atividades da vida diária de 57 pessoas para um total de 500 experimentos. O sistema proposto é implementado em um FPGA e emprega um algoritmo de detecção de limiares único e especifico para cada usuário, donde o sinal obtido é divido em três janelas, cada uma com seu respetivo limiar. Este sistema conseguiu uma

sensibilidade de 98.1% e uma especificidade de 99.2% e os limiares foram determinados empregando o Matlab. Neggazi et al. [73], mediante o uso de um acelerômetro de três eixos localizado na cintura do usuário, desenvolveram um sistema basado em um FPGA. Os dados do acelerômetro são enviados mediante o *bluetooth* para o FPGA, na qual é implementado o algoritmo de detecção, que avalia constantemente a orientação e aceleração do corpo do usuário. Uma queda é detectada quando os limiares de aceleração e o angulo do corpo são atingidos em um determinado tempo. Os eventos de queda são reconhecidos com uma sensibilidade de 96.0% e uma especificidade de 98.0%.

Baseado na revisão da literatura, evidencia-se que poucos trabalhos utilizam os dispositivos vestíveis no pulso. Esta localização gera maior conforto ao usuário, mas produz alguns dados que podem ser considerados como falsos positivos. Neste sentido, verificou-se que não há estudos de um sistema de detecção de quedas, baseado em um acelerômetro e microfone colocados no pulso, uma rede neural artificial, um avaliador de limiares de som e um FPGA. Este sistemas, sera desenvolvido netes trabalho, sendo a rede neural artificial empregada para classificar os diferentes movimentos realizados por uma pessoa, e o avaliador de limiares utilizado para filtrar os falsos positivos gerados na classificação dos movimentos, o qual possibilitaria obter um bom detector. Enquanto o FPGA foi selecionado devido a que permite a aceleração de algoritmos com grande complexidade computacional e a exploração do paralelismo intrínseco que possuem as redes neurais artificiais, além de possibilitar futuras atualizações de hardware da topologia da rede e o treinamento online da mesma.

Tabela 3.1: Trabalhos correlatos que empregam dispositivos vestíveis

| Titulo do trabalho | Ano | Autor | Algoritmo | Sistema de informação | Dispositivos | Localização do dispositivo | Voluntários | Precisão | Especificidade | Sensibilidade |
|---|---|---|---|---|---|---|---|---|---|---|
| A new real-time fall detection approach using fuzzy logic and a neural network | 2009 | Dinh and Struck | Logica fuzzy e uma Rede Perceptron Multicamadas | Informação processada no computador que determina o evento de queda | Acelerômetro, Microcontrolador e Bluetooth | Cintura | 5 pessoas | - | 99,64% | 96,00% |
| Fall detection using a Gaussian distribution of clustered knowledge, augmented radial basis neural-network, and multilayer perceptron | 2011 | Yuwono et al. | Rede Neural MLP e Rede Neural ARBF | Informação processada no computador que determina o evento de queda | Acelerômetro MMA7260Q, Microcontrolador e modulo ZigBee | Cintura | 8 pessoas | - | 99,56% | 97,65% |
| Human Falling Detection Algorithm Using Backpropagation Neural Network | 2012 | Sengto et al. | Human Falling Detection Algorithm Using Backpropagation Neural Network | Informação processada no computador que determina o evento de queda | Acelerômetro MMA7631, PIC24FJ128GB110, cartao SD de 32GB | Cintura | 5 pessoas | - | 99,50% | 96,25% |
| Artificial Neural Networks as an Alternative to Traditional Fall Detection Methods | 2013 | Vallejo et al. | Rede Perceptron Multicamadas | Envia uma alarme para o computador | Acelerômetro ADXL345, um microcontrolador MCF51JM128, um módulo ZigBee | Cintura | 2 pessoas | - | 98,60% | 98,40% |
| Development of wearable human fall detection system using multilayer perceptron neural network | 2013 | Kerdegari et al. | Rede Perceptron Multicamadas | Informação processada no computador que determina o evento de queda | Acelerômetro ADXL345, microcontrolador Atmega328, cartao SD | Cintura | 50 pessoas | 91,60% | 91,07% | 92,03% |
| A Fall Detection Study Based on Neural Network Algorithm Using AHRS | 2013 | Zhang et al. | Filtro EKF e rede neural perceptron multicamada | Informação processada no computador que determina o evento de queda | Acelerômetro ADXL345, giroscópio LPY530 e LPR530, bússola magnética HMC5883, Microcontrolador STM32 | Cintura | 1 pessoa | 98,20% | - | - |
| An Efficient and Robust Fall Detection System Using Wireless Gait Analysis Sensor with Artificial Neural Network (ANN) and Support Vector Machine (SVM) Algorithms | 2014 | Teja et al. | Rede Perceptron Multicamadas e máquina de vectores de suporte | Informação enviada e processada no computador que determina o evento de queda | Acelerômetro, giroscópio, microcontrolador e transmissor RF | Cintura e costas | 2 pessoas | 98,80% | 100,00% | 94.10% |
| A low-power, wireless, wrist-worn device for long time heart rate monitoring and fall detection | 2014 | Zhou et al. | Filtro e algoritmo de limiares | Informação transmitida e processada no computador que determina o evento de queda | Acelerômetro, microcontrolador MSP430 e WiFi | Pulso | 6 pessoas | 93,75% | 95,40% | 83.33% |
| A wrist-worn fall detection system using accelerometers and gyroscopes | 2014 | Hsieh et al. | Algoritmo de limiares | Informação transmitida e processada no computador que determina o evento de queda | Acelerômetro, giroscópio e Zigbee | Pulso | 3 pessoas | - | 96,70% | 95.00% |
| Elder Falls Detection Based on Artificial Neural Networks | 2015 | Vidigal et al. | Rede neural MLP, RBF e Kohonen | Informação processada pelo celular e gera uma alarme utilizando um aplicativo Android | Celular | Braço e cintura | 6 pessoas | 96,40% | 98,30% | 85,00% |
| A Novel Algorithm for Detection Human Falling From Accelerometer Signal Using Wavelet Transform and Neural Network | 2015 | Nuttaitanakul et al. | Transformada Wavelet e uma rede neural perceptron multicamadas | Informação processada no computador que determina o evento de queda | Acelerômetro MMA7331L, Microcontrolador PIC32MX460F512L | Cintura | 5 pessoas | 85,60% | - | - |
| Inclination Gradient-based Fall Detection Algorithm For Wrist- Worn Device | 2015 | Chen et al. | Algoritmo de limiares | Informação transmitida e processada no computador que determina o evento de queda | Acelerômetro, Bluetooth e microcontrolador | Pulso | 12 pessoas | - | 100.00% | 94.44% |
| The Development of Artificial Neural Networks (ANN) for Falls Detection Nantakrit | 2017 | Yodpijit et al. | Algoritmo de limiares e uma rede MLP | Informação processada no computador que determina o evento de queda | Acelerômetro e giroscópio | Cintura | 1 pessoa | 99,23% | 99,37% | 99,00% |
| A Movement Decomposition and Machine Learning-based Fall Detection System Using Wrist Wearable Device | 2018 | Quadros et al. | Decomposição de Madgwick para determinar limiares. SVM, K-NN, LR, DT. | Informação processada no computador que determina o evento de queda | Acelerômetro, giroscópio, magnetômetro, Arduíno Uno | Pulso | 21 pessoas | 100.00% | 97.90% | 100.00% |

# Capítulo 4

# Aquisição de dados

Para o desenvolvimento do presente trabalho foi necessário coletar um conjunto de dados dos movimentos realizados pelas pessoas. Neste capitulo é apresentado o sistema de aquisição de dados, o qual é colocado no pulso e braço do voluntario. É descrito a calibração dos sensores, os parâmetros estabelecidos e as atividades realizadas para a coleta de dados. Também é exposta uma base de dados obtida por um grupo de pesquisa da Universidade Tecnológica Federal do Paraná (UTFPR), que utilizam um acelerômetro colocado no pulso da pessoa, mas com outros parâmetros de captura, a qual foi usada para fins de comparação numérica com a solução proposta neste trabalho.

## 4.1 Sistema de aquisição de dados

O sistema de aquisição de dados foi construído considerando-se o trabalho de Nary et al. [59], que emprega um sistema compacto, portátil, vestível e pouco intrusivo colocado na cintura do usuário. A partir desse trabalho, algumas alterações foram feitas no que diz respeito a localização do sensor e o sistema de armazenamento de dados. Em particular, o sistema de aquisição de dados proposto neste trabalho utiliza um acelerômetro de 3 eixos no pulso usuário e o sistema de comunicação via *bluetooth* foi substituído por uma memória micro SD para o armazenamento dos dados, como mostrado na Figura 4.1.

O sensor utilizado é o MPU6050 do InvenSense [74], que é um sensor de detecção de movimento de seis graus de liberdade, que possui um acelerômetro de 3 eixos e um giroscópio de 3 eixos. Esse sensor de baixo custo consome $4.1mA$ de corrente e trabalha na faixa de tensão de 3.3 a 5V, o qual é conectado ao microcontrolador através de uma comunicação I2C (*Inter-Integrated Circuit*). Para coletar dados de sons foi utilizado o microfone GY-MAX4466 com ganho ajustável [75], que opera na faixa de 2.4 a 5V e consome um máximo de corrente de $60\mu A$. Este dispositivo envia um sinal de tensão dos dados capturados que é convertido em um sinal digital mediante o ADC de 10 bits do microcontrolador.

O microcontrolador utilizado para controlar o sistema de aquisição de dados é um Arduino Nano, baseado em um chip ATMega328P, que opera com um relógio de 16 MHz, uma memória flash de 32 KB e suporta tensões de 5 a 12V [76]. Na memória micro SD são armazenados os dados capturados pelo sensor de aceleração e do microfone, para depois serem analisados e classificados. A comunicação entre a memória e o microcontrolador é feita pelo barramento SPI (*Serial Peripheral Interface*).

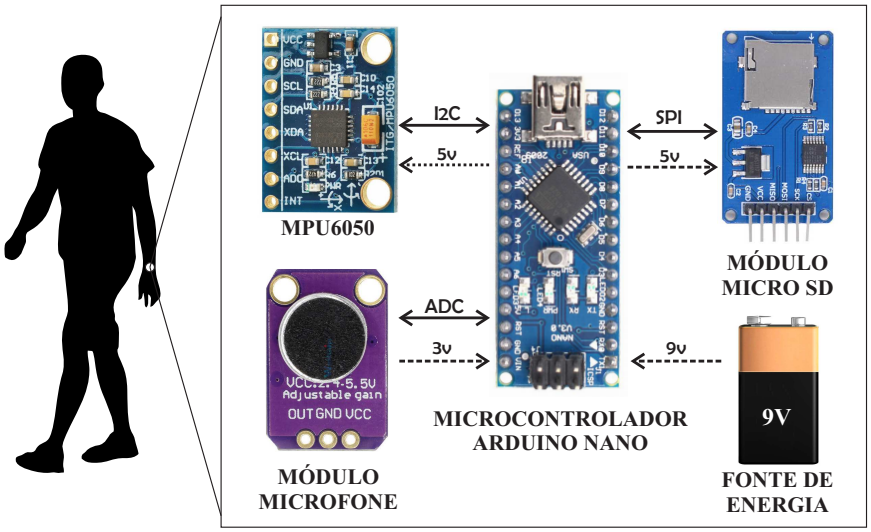



Figura 4.1: Arquitetura do sistema de aquisição de dados.

Como fonte de alimentação externa utiliza-se uma bateria de 9 volts, que está conectada ao Arduino Nano o qual possui reguladores de tensão de 3.3 e 5V, empregados para a operação do dispositivo de armazenamento e os módulos sensores (vide Figura 4.1).

## 4.2 Calibração dos sensores

A calibração dos sensores é feita com o intuito de medir os valores verdadeiros ou reais dos movimentos e sons feitos pela pessoa. Para a calibração do acelerômetro, considerou-se que o mesmo estava apoiado em uma superfície lisa e plana. Isto foi conseguido utilizando um aplicativo Android de nível de bolha, onde a inclinação pode ser observada nos eixos $x$ e $y$, conseguindo-se assim ter controle sobre a posição do sensor. Por outro lado, o microfone foi calibrado inicialmente num lugar com pouco barulho procurando que este pudera só captar os sons de um golpe e um grito.

### 4.2.1 Calibração do acelerômetro

O acelerômetro mede a aceleração presente nos eixos $x$, $y$ e $z$. É configurado com uma resolução de $\pm 2g$ e amostrado a uma frequência de 125 Hz. O sensor foi colocado junto ao microfone em um suporte que foi fabricado em uma impressora 3D, presente no laboratório LEIA-GRACO. O suporte produzido pela impressora pode ser visto na Figura 4.2. Este suporte foi construído com o objetivo de manter os dispositivos fixos no pulso da pessoa. Durante a calibração, o suporte com os sensores foi mantido sobre o celular, que por sua vez está posicionado paralelamente ao plano terrestre, como pode ser visto na Figura 4.3.

Mantendo o dispositivo estático e bem nivelado, foram coletadas 1000 amostras de cada um dos eixos. Com estes dados é obtido um valor máximo ($max_{x,y,z}$) e mínimo ($min_{x,y,z}$), para determinar o valor de deslocamento do sensor, a partir da Equação 4.1.

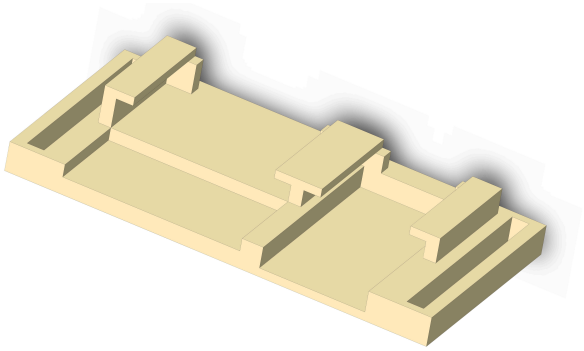


Figura 4.2: Suporte para os sensores.

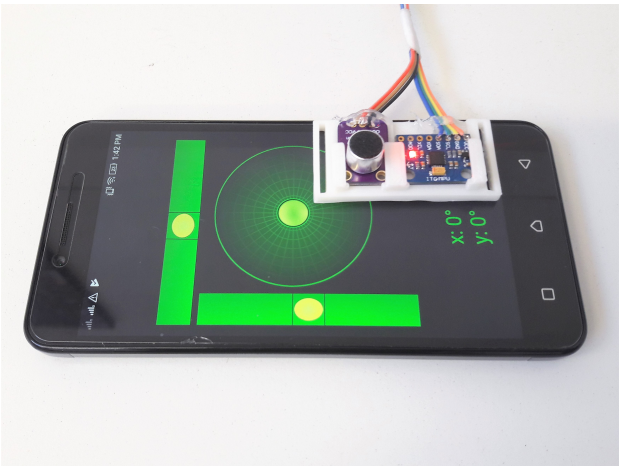


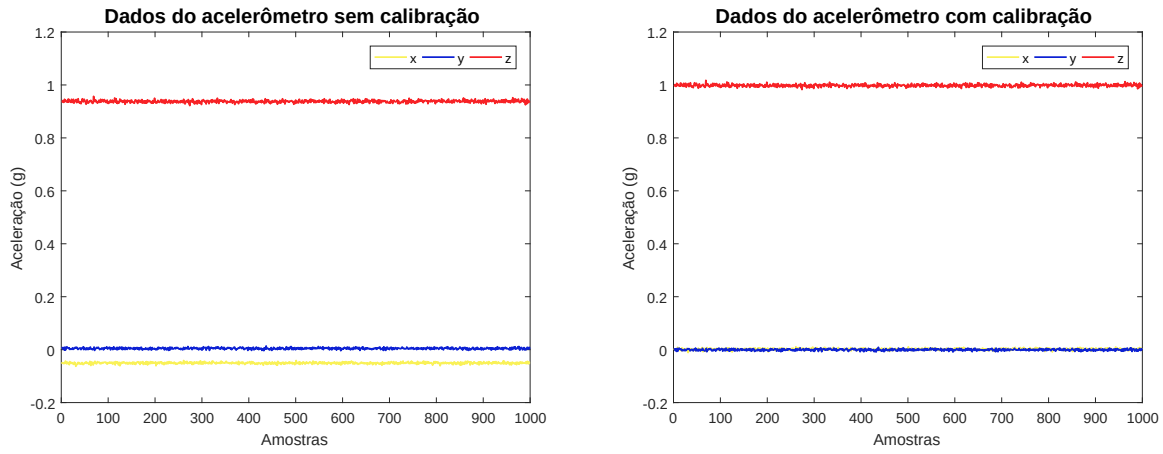

Figura 4.3: Calibração do acelerômetro.

$$offset_{x,y,z} = \frac{max_{x,y,z} - min_{x,y,z}}{2} \tag{4.1}$$

Na Figura 4.4 têm-se os resultados obtidos das medições coletadas pelo acelerômetro nos três eixos, antes e após o processo de calibração. Observa-se que os dados dos eixos $x$, $y$ e $z$ estão um pouco distantes dos valores reais, de $0g$ para os eixos $x$ e $y$ e de $1g$ para o eixo $z$, apresentados na Figura 4.4(a). A partir disto, é calculado o valor do deslocamento e aplicado às mesmas medições do acelerômetro, o resultado desta etapa é apresentado na Figura 4.4(b). Pode-se evidenciar que os valores nos três eixos, estão mais próximos aos reais.

### 4.2.2 Calibração do microfone

O microfone capta os sons presentes ao seu redor e são processados pelo Arduino Nano mediante seu conversor analógico-digital de 10bits. Visando que o microfone não capte sons a mais de um metro de distância, pois o mesmo será utilizado no pulso, ficando próximo da origem dos sons, seu ganho foi alterado através de um potenciômetro instalado no próprio microfone. Para esta calibração foi empregado um osciloscópio e um Arduino Nano ligado ao computador mostrado na Figura 4.5, os quais permitiram




(a) Acelerômetro não calibrado

(b) Acelerômetro calibrado

Figura 4.4: Dados obtidos sem e com calibração

observar as diferenças entre os sons produzidos por um grito, um golpe na mesa e outro no chão.

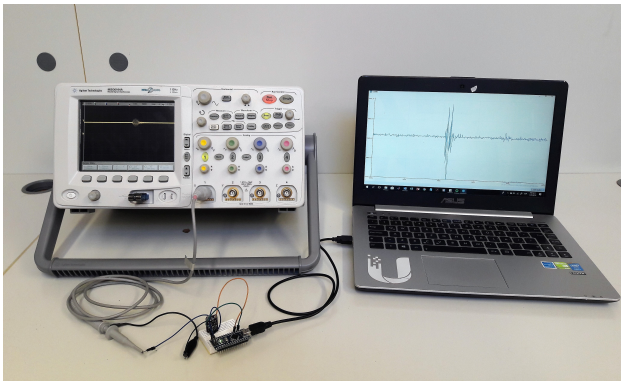


Figura 4.5: Calibração do microfone.

## 4.3 Composição da base de dados deste trabalho

Finalizada a calibração dos sensores e testes para verificação do correto funcionamento do sistema, continuou-se com a etapa experimental. Nesta etapa, o dispositivo projetado foi colocado no braço, como pode ser visto na Figura 4.6. Nesta configuração, os sensores ficam no pulso do usuário e o conjunto: Arduino Nano, cartão micro SD, bateria e o botão de início e parada de gravação dos experimentos, ficam na parte superior do braço.

Os experimentos realizados com o dispositivo foram feitos considerando-se os estudos [12], [77], que determinam que alguns movimentos da vida diária eram confundidos com quedas pelo sistema de detecção. Por este motivo, foram incluídos nos experimentos atividades da vida diária como deitar na cama, pular, correr, bater palmas, subir e descer escadas, derrubar o dispositivo no chão e quedas frontais, de costas e de lado.


Figura 4.6: Dispositivo instalado no braço do voluntário.

Através da internet, foram assistidos e analisados diferentes vídeos sobre quedas, com o objetivo de aprender como uma pessoa cai. Determinou-se que a maioria das pessoas dobram os joelhos quando caem. Considerando este fato, foi explicado para cada voluntário como deveriam realizar o movimento de queda, com o objetivo de ter dados os mais próximos da realidade. As quedas foram simuladas sobre um colchão de 20 cm de espessura, sempre tentando dobrar os joelhos para cair. Uma queda é apresentada na Figura 4.7.

Figura 4.7: Queda simulada sobre o colchão.

Neste trabalho participaram 20 voluntários, 4 mulheres e 16 homens, cujas caraterísticas físicas são apresentadas na Tabela 4.1. Na coleta de dados de treinamento dos voluntários, 2 mulheres e 10 homens fizeram as atividades descritas anteriormente, exceto as duas mulheres acima de 40 anos, que participaram apenas fazendo atividades da sua vida diária. Estes primeiros experimentos foram guardados num arquivo .txt no cartão micro SD, para depois serem analisados no MATLAB. A quantidade de experimentos feitos é apresentada na Tabela 4.2. Os voluntários restantes, participaram nas atividades de validação, cujos resultados serão apresentados no Capítulo 7.

As outras atividades, correspondem a movimentos como amarrar cordões do sapato, fazer limpeza da casa, escrever um documento com caneta, trabalhar no laptop, entre outras. Estas só fazem parte dos experimentos para testar o sistema de detecção de quedas.

Tabela 4.1: Características físicas dos voluntários

| Voluntario | Gênero | Idade | Altura | Peso |
|---|---|---|---|---|
| 1 | Mulher | 50 | 156 | 62 |
| 2 | Mulher | 84 | 154 | 58 |
| 3 | Mulher | 19 | 160 | 56 |
| 4 | Mulher | 26 | 164 | 59 |
| 5 | Homem | 27 | 167 | 61 |
| 6 | Homem | 27 | 160 | 59 |
| 7 | Homem | 27 | 176 | 84 |
| 8 | Homem | 29 | 168 | 60 |
| 9 | Homem | 30 | 168 | 68 |
| 10 | Homem | 27 | 172 | 80 |
| 11 | Homem | 30 | 176 | 74 |
| 12 | Homem | 27 | 172 | 72 |
| 13 | Homem | 26 | 168 | 59 |
| 14 | Homem | 38 | 174 | 75 |
| 15 | Homem | 22 | 180 | 76 |
| 16 | Homem | 20 | 185 | 84 |
| 17 | Homem | 23 | 175 | 69 |
| 18 | Homem | 29 | 165 | 59 |
| 19 | Homem | 34 | 177 | 76 |
| 20 | Homem | 28 | 186 | 73 |

Tabela 4.2: Quantidade de experimentos

| Atividade | Quantidade | Unidade |
|---|---|---|
| Quedas para frente | 100 | experimento |
| Quedas para trás | 50 | experimento |
| Quedas de lado | 20 | experimento |
| Caminhar | 35 | minutos |
| Correr | 35 | minutos |
| Pular | 20 | minutos |
| Descer escadas | 20 | minutos |
| Subir escadas | 20 | minutos |
| Outras atividades | 120 | minutos |

## 4.4 Base de dados do grupo de pesquisa da UTFPR

No trabalho de Quadros et al. [71], foi utilizada uma unidade de medição inercial (IMU, *Inertial Measurement Unit*), a qual é usada para detectar e medir um movimento do corpo com a combinação de dois ou mais sensores. O dispositivo utilizado para a criação da base de dados foi o GY-80, composto por um acelerômetro triaxial ADXL345, um giroscópio triaxial L3G4200D e um magnetômetro triaxial HMC5883L.

O ADXL345 de *Analog Devices*, é um acelerômetro triaxial que pode trabalhar nas faixas de $\pm$2g a $\pm$16g, de baixo consumo de potência, com um máximo de $23\mu A$ quando está realizando medições e uma taxa de amostragem de até 3200Hz [78]. O giroscópio L3G4200D da *ST Microelectronics*, tem uma resolução de 16bits que permite uma medição de qualidade, uma taxa de amostragem de 100Hz a 800Hz e uma medição da velocidade angular triaxial em três escalas: 250, 500 e 2000 graus por segundo [79]. O magnetômetro HMC5883L da *Honeywell*, é um magnetômetro triaxial capaz de atingir taxas de amostragem de até 160Hz, com resolução de 12 bits e uma faixa de medição de $\pm 8$ Gauss [80].

Para a aquisição dos dados foi utilizado um Arduino UNO, encarregado de executar o protocolo de comunicação I2C que permite obter os sinais medidos com o dispositivo IMU e posteriormente serem enviados para um computador pessoal. Os dados foram obtidos com uma taxa de amostragem de 100Hz para todos os sensores, nas faixas de 4G para o acelerômetro, 500 graus/seg para o giroscópio e 0,88Ga para o magnetômetro. O Arduino UNO e o dispositivo IMU foram montados em uma pulseira de neoprene, oferecendo uma opção confortável para utiliza-los no pulso.

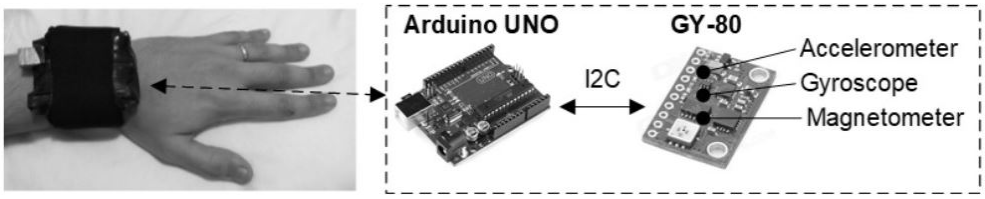



Figura 4.8: Sistema de aquisição de dados da UTFPR
Fonte: [71].

Para a aquisição dos dados utilizando voluntários, o estudo recebeu a aprovação do Comitê de Ética em Pesquisa em Seres Humanos da Universidade Tecnológica Federal de Paraná (UTFPR), sob o número de registro 2000216.0.0000.5547.

Os sinais foram adquiridos de diferentes voluntários, simulando seis atividades de queda e seis de não queda, como segue:

- Quedas: queda para a frente, queda para trás, queda lateral (para o lado com o dispositivo), queda lateral (para o lado sem o dispositivo), queda após girar a cintura no sentido horário e cair depois de girar a cintura no sentido anti-horário;
- Não quedas: andando, batendo palmas, abrindo e fechando uma porta, movendo um objeto, amarrando sapatos e sentando em uma cadeira.

Todos os movimentos de simulação de queda foram realizados em um colchão macio, evitando lesões voluntárias devido ao impacto físico. Os voluntários usaram o dispositivo no pulso não dominante.

Para as atividades propostas, vinte e dois voluntários participaram, repetindo cada atividade três vezes, totalizando 66 sinais para cada ciclo de teste, obtendo-se um total de 792 sinais. Metade deles está relacionada à simulação de queda e a outra metade a atividades da vida diária. Cada sinal de queda e não queda começa com uma posição estática (braços em repouso), seguida por alguns passos antes da simulação do evento. A média da duração dos sinais é de 9,2 segundos. As características gerais dos voluntários envolvidos são: Idade (anos): 26,09 $\pm 4,73$ anos; Altura (m): 1,68 $\pm 0,11$; Peso (kg): 67,82 $\pm 12,24$.

# Capítulo 5

# Proposta metodológica para a detecção de quedas

No presente capítulo, e realizada uma análise dos sinais obtidos com o microfone e o acelerômetro, onde inicialmente o acelerômetro fornece um padrão característico para cada movimento feito, os quais são classificados em dois grupos: queda e não queda. Os dados fornecidos pelo microfone são utilizados para reforçar a decisão tomada pela rede neural projetada. Os dados de som serão avaliados de forma independente através o uso de limiares, devido a que não possuem um padrão característico que permita sua classificação. Os limiares são estabelecidos de acordo aos picos de som presentes nas quedas.

## 5.1 Classificação dos dados

Com os experimentos realizados e com a utilização da ferramenta Matlab, foram obtidas as representações gráficas das atividades realizadas. Na Figura 5.1 a aceleração é medida em g, onde 1g é igual a $9.8m/s^2$ e o som capturado pelo microfone medido em volts. Neste gráfico são mostrados os dados de aceleração presentes nos eixos $x$, $y$ e $z$ para um movimento de queda, que apresenta diversas variações nas medições devido ao movimento realizado e à localização do sensor no pulso. Por outro lado, o sinal do microfone representa o som gerado pelo impacto do corpo sobre o colchão.

A partir do evidenciado na Figura 5.1, não foi possível determinar um padrão para cada movimento, portanto, optou-se por calcular a magnitude da aceleração $MA_{cc}$ no espaço tridimensional, empregando a Equação 5.1, que permitiu ver a influência da aceleração sobre o dispositivo colocado no pulso. O resultado deste cálculo é apresentado na Figura 5.2, onde observa-se um sinal que representa um padrão que está presente em outras quedas realizadas.

$$MA_{cc} = \sqrt{A_x^2 + A_y^2 + A_z^2} \tag{5.1}$$

Estabelecido como serão tratados os dados obtidos, continua-se pela apresentação dos gráficos dos sinais de aceleração e som na Figura 5.3 produzidos por diferentes movimentos. Esta figura mostra o com-

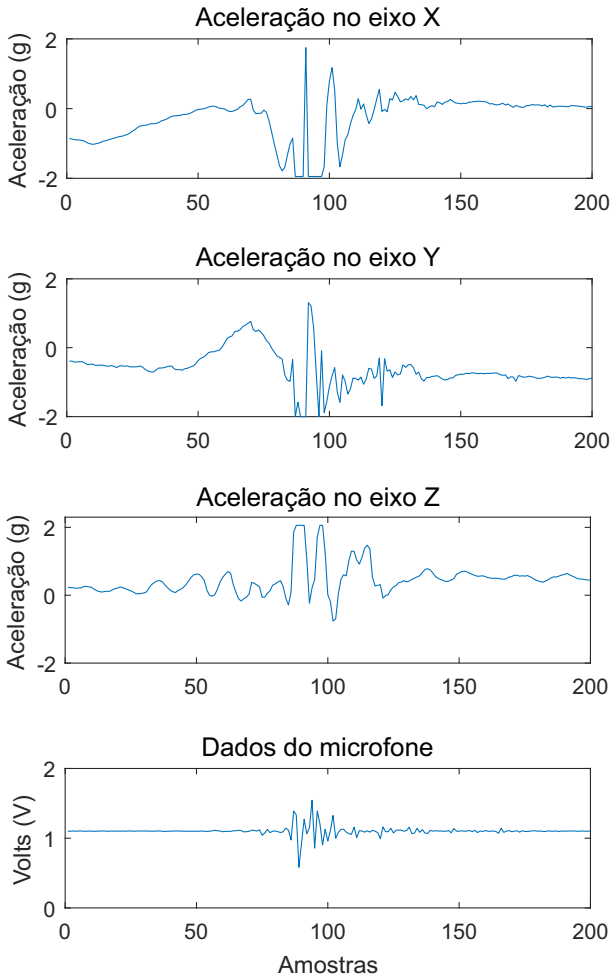



Figura 5.1: Gráfico de uma queda representada nos três eixos e o som produzido.

portamento do sinal de aceleração e som durante as atividades de caminhar, correr, pular rapidamente e bater palmas. Onde a aceleração por sua vez segue um determinado padrão para cada uma das atividades, variando apenas a amplitude em função dos movimentos. Pode-se perceber que correr e pular tem gráficos semelhantes, mas a duração dos picos na atividade de corrida é pequena apresentando uma maior amplitude, por outro lado o sinal de som apresenta mais barulho do que quando se está pulando. O experimento de bater palmas possui picos mais agudos e bem definidos na aceleração e som com tempos pouco prolongados. Quando a pessoa está caminhando a amplitude da aceleração não atinge os $2g$, tem um sinal repetitivo, prolongado no tempo e não apresenta sons com uma intensidade alta. Pelo gráfico dos sinais produzidos pelo microfone, percebe-se que ele é capaz de captar alguns sons que pertencem ao ambiente e aos movimentos realizados, não sendo possível determinar um padrão entre eles para serem classificados como no caso do sinal de aceleração.

Com alguns dos gráficos obtidos, foi determinada que uma faixa de 138 amostras pode representar um evento de queda e que ela sempre segue esse padrão independentemente das caraterísticas físicas de uma pessoa, o que pode ser visto na Figura 5.4. Onde no início, a queda apresenta uma desaceleração que vai até valores menores de 0.6g, em seguida aumentado rapidamente até valores acima de 2.8g, o que sempre

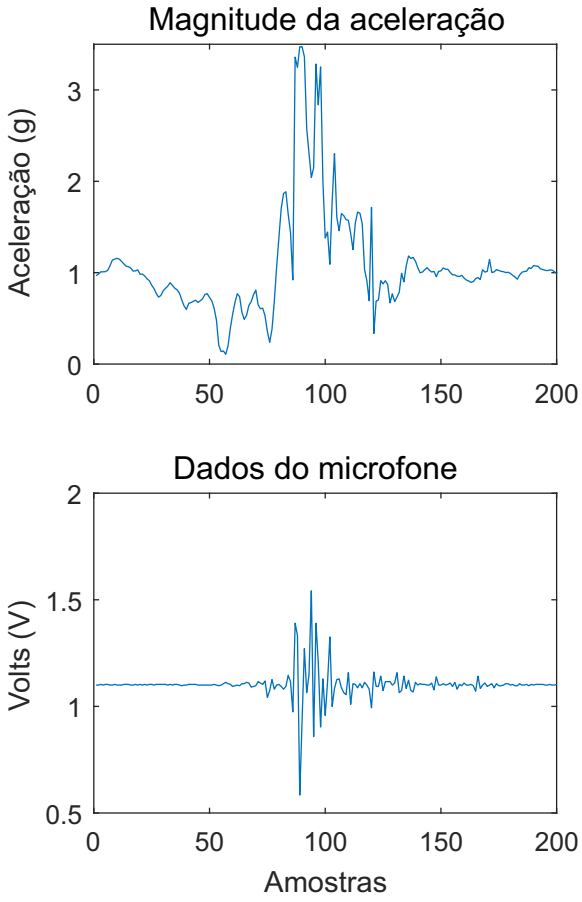


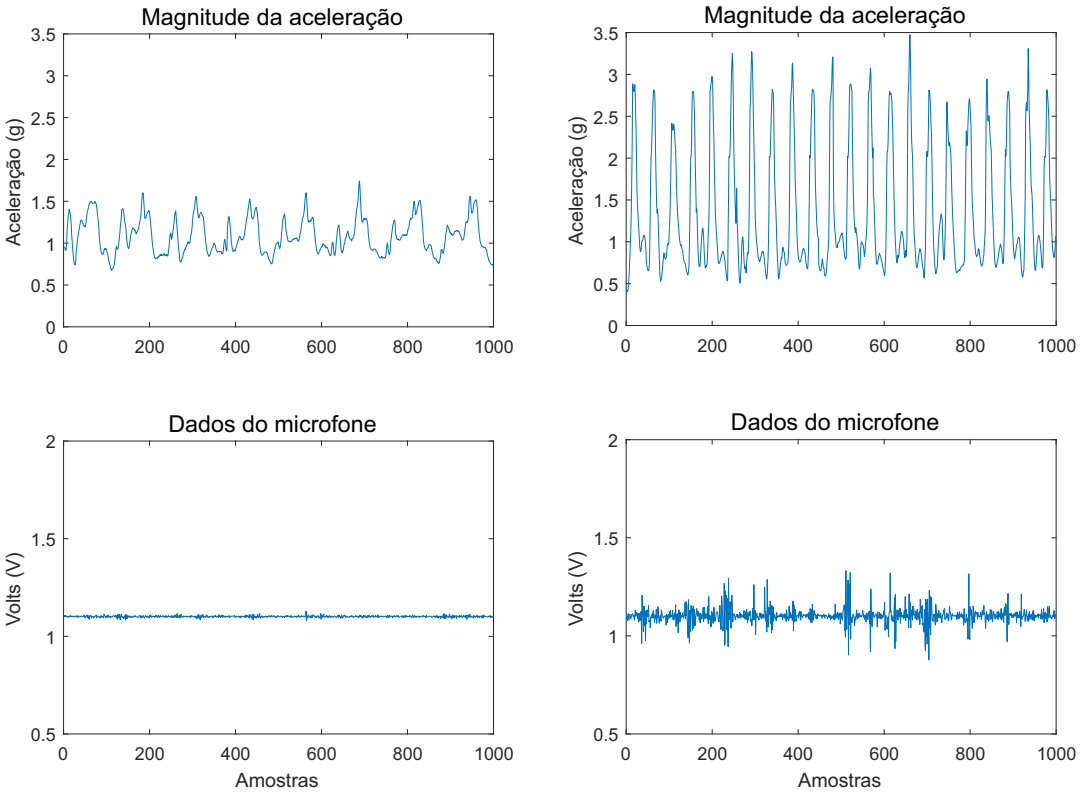

Figura 5.2: Gráfico da magnitude de uma queda e o som produzido.

acontece quando a pessoa impacta o chão. O movimento de queda termina quando a aceleração volta a estabelecer-se ao redor de 1g. Esta descrição se repete diferentes vezes nas quedas simuladas, gerando um padrão. Além disso, se determinou que uma queda tem uma duração aproximada de 1 segundo.

## 5.2 Ferramenta de extração de dados

Devido a grande quantidade de dados coletados com o Arduíno Nano, com o objetivo de treinar uma rede neural passando para ela movimentos determinados e classificados em grupos, foi desenvolvida uma ferramenta que permite visualizar de maneira mais rápida os dados capturados pelo acelerômetro e extrair as janelas de 138 amostras pertencentes às quedas e outros movimentos. Na Figura 5.5 é mostrada a ferramenta desenvolvida no Matlab para a análise dos dados.

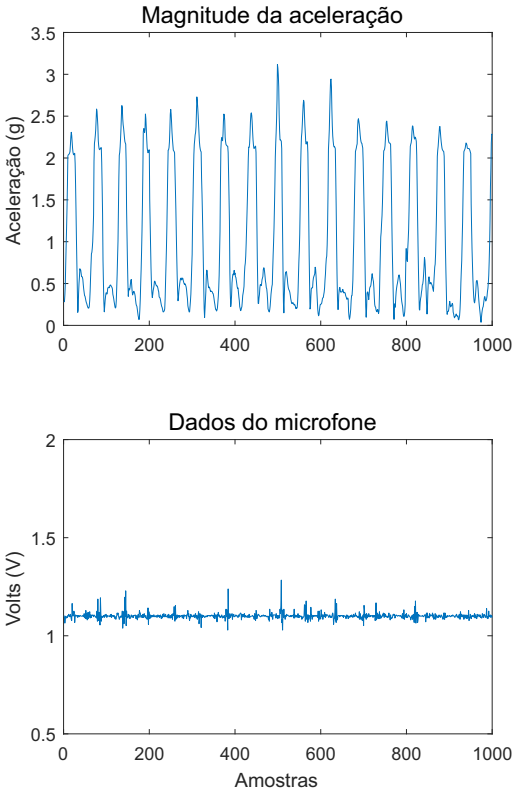
Para extrair uma queda mediante a ferramenta da Figura 5.5, os dados dos arquivos .txt são importados mediante o botão *Arquivo*, estes são mostrados no primeiro quadro *magnitude da aceleração*, onde se pode evidenciar todo os dados capturados antes e após a queda. Para extrair o movimento de queda, se estabelece o ponto inicial, neste caso 530, o qual é inserido no espaço chamado *mínimo*. A faixa que se deseja extrair é de 138 amostras, mas este valor pode ser mudado no espaço *A x faixa* se necessário. Para salvar o novo



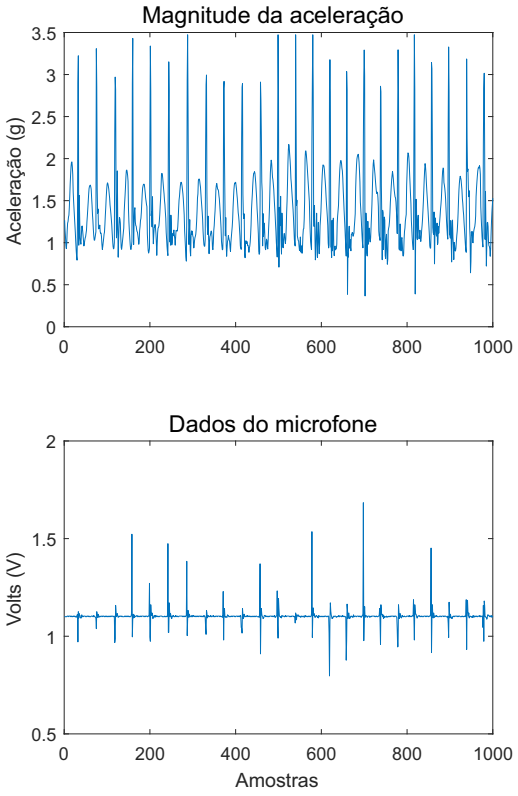

(a) Caminhando




(b) Correndo




(c) Pulando rapidamente




(d) Batendo palmas

Figura 5.3: Dados obtidos para os movimentos (a) Caminhar, (b) Correr, (c) Pular Rapidamente e (d) Bater Palmas

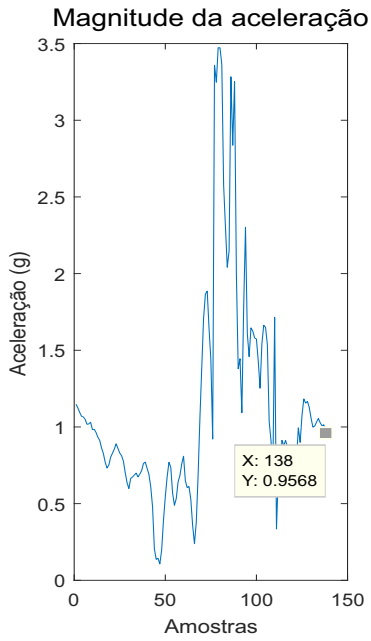



Figura 5.4: Padrão do sinal de aceleração que segue uma queda.

gráfico, deve-se colocar o nome do arquivo no espaço *Nome do arquivo novo*, após clicar na opção gráfico, que gerara a figura no quadro *Janela*, depois de verificar o gráfico, finalmente, este é salvado num arquivo *.mat* clicando a opção *Salvar*.

## 5.3 Rede Neural

Neste trabalho é utilizada uma rede neural tipo perceptron multicamada, treinada mediante um algoritmo do tipo supervisionado chamado *BackPropagation*. Inicialmente foi proposta uma rede com 138 entradas, onde cada entrada corresponde a uma amostra do sinal padrão estabelecido anteriormente, o qual representa uma queda. Contudo, para esta quantidade de entradas, as operações a serem feitas para obter uma resposta na rede é grande, pois dependem da quantidade de neurônios empregados. Por conseguinte, foi proposto um pré-processamento dos dados da janela obtida, com o intuito de diminuir as entradas e obter uma rede neural pequena e eficiente, mediante a filtragem do sinal, discretização e posterior normalização.

### 5.3.1 Filtragem e discretização

Para a filtragem do sinal é utilizado um filtro de média, representado pela Equação 5.2. Este filtro permite a análise de um conjunto $n$ de dados da variável $x$, que chegam em um fluxo ordenado, onde serão somados e divididos por $n$.

$$\bar{x}_n = \frac{x_1 + x_2 + ... + x_n}{n} \tag{5.2}$$

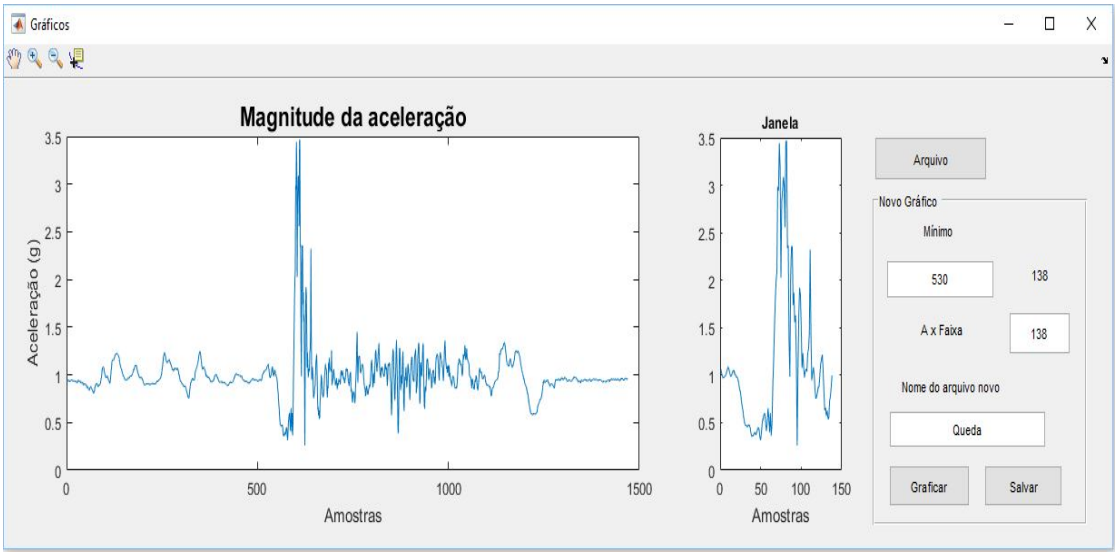



Figura 5.5: Ferramenta para extrair movimentos.

Para o processamento dos dados obtidos com o acelerômetro, na janela de 138 amostras foi aplicado o filtro $138/n$ vezes, e os valores obtidos para cada aplicação são mantidos durante $n$ amostras para representar o movimento feito. Por exemplo, aplicando o filtro com $n = 16$ a uma janela que representa uma queda, deve-se aplicar o valor inteiro do resultado, que é $8.625$. Do lado esquerdo da Figura 5.6 são mostrados os oito valores de maneira constante, durante 16 amostras, o que permite evidenciar uma nova representação da queda ou movimento, com um comportamento que acompanha o formato do sinal original.

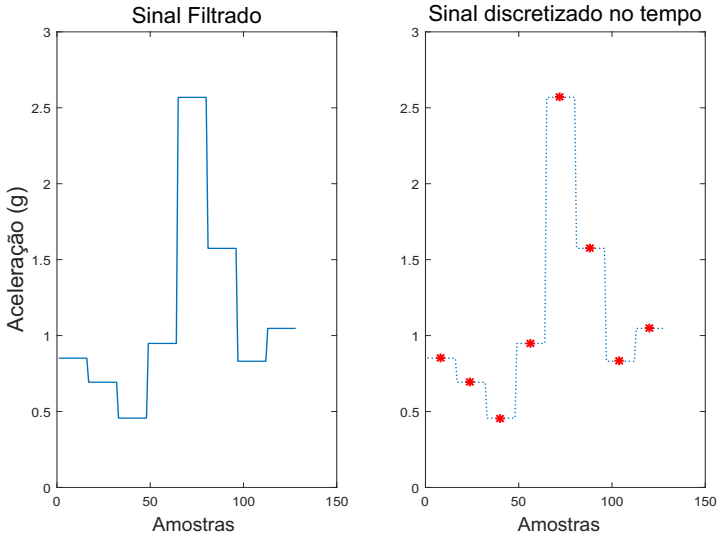



Figura 5.6: Sinal filtrado e discretizado.

A partir do sinal filtrado da queda, o mesmo é discretizado, pegando um dado por cada nível estabelecido, como se pode evidenciar à direita da Figura 5.6. Com esta discretização, se consegue uma diminuição na quantidade de entradas, neste caso de 138 entradas para 8, as quais serão as novas entradas da rede neural artificial.

### 5.3.2 Normalização dos dados

Antes do treinamento da rede neural artificial, os dados devem passar por uma transformação de escala, chamada de normalização, onde são utilizadas funções que ajudam a distribuir uniformemente os valores de entrada em uma faixa uniforme. Comumente, é realizado um redimensionamento dos dados originais que se encontram em uma faixa, para uma nova faixa de $[0, 1]$ ou $[-1, 1]$ [81]. Devido ao comportamento dos dados de entrada e aos limites da função de ativação tangente hiperbólica usada neste trabalho, a faixa empregada para a normalização dos dados da rede neural é de $[-1, 1]$, permitindo que todas as entradas da rede recebam igual atenção no processo de treinamento [82]. Se a normalização não for feita, os dados de entrada terão um efeito adicional no neurônio, levando a decisões incorretas. Para a normalização dos dados é utilizada a Equação 5.3.

$$y = \frac{(y_{max} - y_{min}) * (x - x_{min})}{x_{max} - x_{min}} + y_{min} \tag{5.3}$$

Onde $x$ é a entrada a normalizar, $x_{max}$ e $x_{min}$ o valor máximo e mínimo respectivamente da entrada, $y$ é o dado normalizado e os valores $y_{max}$ e $y_{min}$ são os máximos e mínimos do mesmo.

Neste trabalhou foi escolhida a função de ativação tangente hiperbólica, levando em consideração as comparações realizadas com outras funções de ativação apresentadas no trabalho de Kerdegari, H. et al [63]. Nesse trabalho, os autores empregam uma rede neural artificial perceptron multicamada para detectar quedas, utilizando um acelerômetro e giroscópio colocado na cintura, além de apresentar uma análise de desempenho das redes.

### 5.3.3 Treinamento da rede neural artificial

Antes do treinamento da rede, a Equação 5.1 correspondente ao cálculo da magnitude do sinal de aceleração foi modificada, isto com o fim de diminuir a quantidade de operações que devem ser mapeadas em hardware e o tempo de processamento gerado pela raiz quadrada, que demora mais que a multiplicação e soma dos valores de cada eixo. Portanto, é empregada a magnitude elevada ao quadrado, descrita pela Equação 5.4.

$$MA_{cc}^2 = A_x^2 + A_y^2 + A_z^2 \tag{5.4}$$

Depois de definido o pré-processamento dos dados, o treinamento da rede neural artificial é feito utilizando a ferramenta *nntraintool* do Matlab mostrada na Figura 5.7. Nesta etapa emprega-se uma rede neural perceptron multicamada que tem uma grande capacidade de aprender e reconhecer padrões, generalizando o seu comportamento ante a presença de novos dados de entrada, além de possuir uma paralelização intrínseca na sua topologia.

A arquitetura da rede neural artificial é estabelecida considerando-se que a quantidade de entradas e neurônios escondidos mudam dependendo das configurações do filtro e discretização utilizada. Assim, a topologia da rede foi definida em função dos resultados de treinamento de cada configuração. A saída da rede neural é constituída por apenas um neurônio dado que o presente estudo está centrado na determinação

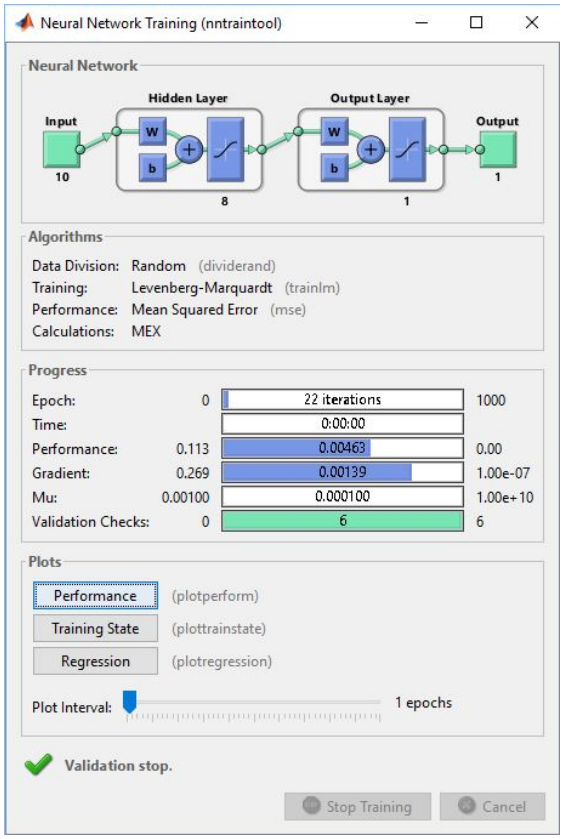



Figura 5.7: Ferramenta para o treinamento da rede neural.

da ocorrência de um evento de queda (os dados são classificação como queda ou não queda).

Para o treinamento da rede foi utilizado o algoritmo *Levenberg-Marquardt Backpropagation* caracterizado por ser um algoritmo eficiente e rápido para o treinamento de redes neurais de tamanho moderado. Para este algoritmo o valor mínimo do gradiente escolhido é 1e-7. Este valor foi escolhido com o objetivo de que o treinamento pare quando o erro da validação, calculado mediante o erro quadrático médio (MSE, ou *mean squared error*), não melhore durante 6 iterações consecutivas. Adicionalmente, é estabelecido um máximo de 1000 iterações (*Epoch*) para o treinamento caso não sejam atingidos os dois parâmetros anteriores.

Após de definir os parâmetros para o treinamento, a base de dados constituída por 340 experimentos (170 quedas e 170 não quedas de 9 homens e 3 mulheres) cada um com uma janela de 138 amostras, é dividida em três grupos: treinamento (70%), validação do treinamento (15%) e teste do treinamento (15%). Os subgrupos um e dois são utilizados para que a rede neural artificial aprenda os padrões que foram determinados e o terceiro, para testar se a rede neural consegue generalizar com dados que não pertencem aos dois primeiros grupos.

Ao finalizar o treinamento, a melhor rede obtida conseguiu um desempenho menor ao 80% no teste, classificando erroneamente os movimentos. Por conseguinte, decidiu-se utilizar esta rede para avaliar os experimentos de não queda, que no total estão constituídos por 1875 amostras cada um, extraindo novas janelas puntuais para os dados que foram classificados erradamente (não quedas detectadas como quedas,

ou também conhecidos como falsos positivos). Com as novas janelas foi retreinada a rede, mas não se conseguiu um bom resultado. Assim, voltou-se a extrair novas janelas dos movimentos de não queda e esse processo foi repetido até obter um desempenho aceitável, procurando que a rede não ficasse sobre-treinada, ou seja, que só identificasse os dados de treinamento e não consiga generalizar com o grupo de teste. Ao final deste processo, foram obtidas 1740 novas janelas pertencentes aos 170 experimentos de não quedas, que somadas com as 170 janelas ja existentes, dão como resultado 1910 dados de não queda para o treinamento da rede. Logo, o trabalho de aprendizagem concentrou-se em corrigir classificações de movimentos da vida diária que a rede está detectando como queda.

Para definir a quantidade de neurônios na camada escondida, foram feitas diferentes redes neurais MLP, variando o $n$ do filtro entre 32, 23, 16 e 13 amostras, obtendo assim 4, 5, 8, e 10 entradas para a rede, constituídas por 2, 4, 6, 8, 15 e 20 neurônios na camada escondida. A Tabela 5.1 mostra a porcentagem de acertos entre o resultado desejado e a saída obtida pela rede usando os dados de teste.

Tabela 5.1: Porcentagem de acertos usando dados de teste após o treinamento.

| **Entradas** | **Neurônios** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **2** | **4** | **6** | **8** | **10** | **15** | **20** |
| **4** | 96,18 | 96,91 | 95,52 | 96,61 | 96,72 | 97,21 | 90,38 |
| **6** | 98,03 | 97,19 | 98,25 | 98,72 | 98,39 | 97,77 | 98,69 |
| **8** | 97,19 | 97,94 | 93,59 | 99,31 | 98,96 | 98,74 | 98,73 |
| **10** | 96,31 | 97,51 | 98,99 | 98,37 | 99,04 | 98,70 | 98,26 |

De acordo com os resultados da Tabela 5.1, optou-se por uma topologia com 8 entradas e 8 neurônios na camada escondia, a qual apresenta os melhores resultados na classificação de padrões, com um 99,31%.

## 5.4 Avaliação do desempenho da rede neural artificial

A rede neural artificial selecionada na etapa anterior (8 neurônios de entrada, 8 neurônios na camada oculta e 1 neurônio de saída) foi implementada no Matlab e seu desempenho foi avaliado em função da sensibilidade, especificidade e precisão. Para isto, três homens diferentes aos que participaram da etapa de treinamento, fizeram 78 quedas simuladas e 52 atividades da vida diária em intervalos de 2 minutos. Adicionalmente, das pessoas maiores de 40 anos foram tomadas 70 atividades da vida diária. De acordo com as equações estabelecidas na Subseção 2.6.2, os resultados experimentais da detecção de quedas foram calculados e mostrados na Tabela 5.2, a qual esta baseada em uma matriz de confusão, que permite a visualização do desempenho do sistema e o erro na classificação das classes. Sendo o valor predito obtido a partir de um valor real inserido no sistema, as quedas consideradas como valores positivos e as ADL como valores negativos.

Os resultados obtidos, mostram que a rede projetada para a detecção de quedas, tem uma sensibilidade de 93,59%, uma especificidade de 95.08% e uma precisão de 94.50%, evidenciando que o pré-processamento dos dados e posterior avaliação da rede neural tem um bom desempenho.

Tabela 5.2: Resultados da avaliação da rede implementada no Matlab.

| Experimentos | | 200 | **Valor Predito** | |
|---|---|---|---|---|
| | | | Positivos | Negativos |
| **Valor Verdadeiro** | Positivos | 78 | TP = 73 | FN = 5 |
| | Negativos | 122 | FP = 6 | TN = 116 |
| Sensibilidade = 93,59 | | | Especificidade = 95,08 | Precisão = 94,50 |

## 5.5 Avaliador de limiares de som

Com o fim de incrementar a especificidade do sistema e, como consequência, a precisão do mesmo, os sinais de som gerados durante uma queda, serão tratados como um verificador do evento. Como mencionado no início deste capítulo, não foi possível identificar um padrão do sinal de som que permita classificar os movimentos de queda e não queda. Portanto, foi escolhido realizar uma análise dos picos do sinal de som, chamado de *avaliador de limiares*, de forma que possa ser usado para confirmar a queda detectada pela rede neural artificial e assim diminuir os falsos positivos detectados.

No algoritmo 1 é apresentado o pseudocódigo que deve seguir o avaliador de limiares para seu funcionamento. Antes de iniciar o módulo avaliador, na linha 1 é inicializado o contador em zero. A avaliação é realizada quando o sinal $Start$ é igual a um e em seguida é avaliado se o contador está em zero. Se esta condição é cumprida, o sinal do microfone é analisado mediante duas condições: verificando se está por baixo do limiar inferior $Li$ ou por cima do limiar superior $Ls$. Quando o sinal atinge uma de estas condições, é informado na saída $La$ que o som pode pertencer a uma queda, e o contador é incrementado em um durante a avaliação de 75 novas amostras. O sinal na saída $La$ permanece em 1 durante estas 75 amostras, devido a que o som do impacto é registrado na metade do padrão do movimento de queda (vide Figura 5.2). O contador do avaliador é reiniciado quando chega a 75, a saída muda seu valor para 0 e o avaliador repete o processo do algoritmo para novas amostras.

O mapeamento em *hardware* do sistema de detecção de quedas conformado pela rede e o avaliador de limiares, será apresentado no seguinte capítulo.

**Algoritmo 1:** Avaliador de limiares

**Entrada:** Som capturado, $S$. Realizar avaliação $Start$, Limiar superior, $Ls$. Limiar inferior, $Li$.

**Saída:** Limiares atingidos, $La$

```
1  contador = 0
2  i = 0
3  início
4      repita
5          se Start = 1 então
6              se contador == 0 então
7                  se S < Li ou S > Lu então
8                      La = 1
9                      contador++
10                 fim
11                 senão
12                     La = 0
13                 fim
14             fim
15             senão
16                 se contador == 75 então
17                     La = 0
18                     contador = 0
19                 fim
20                 senão
21                     contador++
22                 fim
23             fim
24         fim
25     até i = 1;
26 fim
```

# Capítulo 6

# Implementação do sistema de detecção de quedas no FPGA

Este capítulo descreve a arquitetura de *hardware* do sistema de detecção de quedas, projetada a partir da metodologia apresentada no Capítulo 5, a qual será implementada em um FPGA, com o fim de avaliar o desempenho desta em tempo real. Para a implementação dos circuitos, foram usadas as bibliotecas aritméticas de ponto flutuante personalizadas de 27 bits, baseadas no padrão IEEE-754 (IEEE Standards Board, 1985), desenvolvidos e avaliados pelo Muñoz et al. [83], [84]. Estas bibliotecas foram usadas para obter uma melhor precisão nos cálculos realizados pelo detector de quedas.

## 6.1 Arquitetura geral do detector de quedas

O módulo do algoritmo de detecção de quedas mostrado na Figura 6.1, possui quatro entradas e duas saídas. A porta *Entrada* recebe os dados do acelerômetro para serem avaliados pelo algoritmo e o resultado obtido é apresentado na porta *Saída* e o sinal *Ready* informa quando a avaliação está pronta.

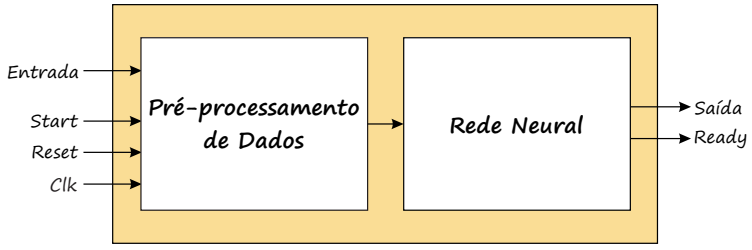



Figura 6.1: Componentes do módulo detector de quedas.

Este módulo está constituído por dois blocos: o *Pré-processamento de Dados* e a *Rede Neural*. O primeiro está encarregado de calcular a magnitude, filtrar, discretizar em tempo e normalizar o sinal de aceleração. O segundo está conformado pelos neurônios e recebe os dados pré-processados e avalia se existe ou não uma queda no sinal. O algoritmo foi projetado em três tipos de arquiteturas: *paralela*, *semi-paralela* e *serial* com o fim de avaliar seu desempenho, consumo de recursos e precisão. Cada uma destas

serão apresentadas nas seguintes seções. Todas as arquiteturas fazem uso de máquinas de estados finitos e são sincronizadas através dos sinais de *start* e *ready*.

## 6.2 Implementação do modulo pré-processamento de dados

O pré-processamento dos dados coletados pelo acelerômetro está representado pelos blocos mostrados na Figura 6.2, que estão encarregados de obter um novo sinal dos movimentos realizados pelo usuário, diminuindo a quantidade de amostras coletadas, para logo serem avaliadas na rede neural artificial.

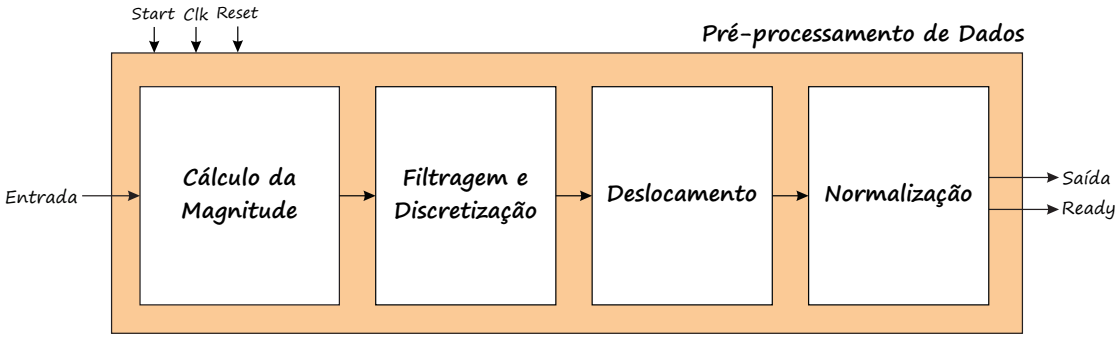



Figura 6.2: Blocos do *Pré-processamento de dados*.

O primeiro processo é o cálculo da magnitude produzida pelos valores da aceleração presentes nos eixos $x$, $y$ e $z$. Isto é feito no bloco nomeado *Cálculo da Magnitude*. Em seguida, o resultado obtido é filtrado mediante um filtro de média móvel e discretizado no tempo (vide bloco de *Filtragem e Discretização*). No bloco de *Deslocamento*, os valores discretizados são armazenados e deslocados cada vez que chega um novo dado discretizado e por último estas saídas são normalizadas no bloco *Normalização*. A continuação é apresentado detalhadamente cada bloco que faz parte do pré-processamento de dados.

### 6.2.1 Bloco cálculo da magnitude

A partir da Equação 5.4, são projetadas as operações que deve ter o modulo que calcula a magnitude do sinal, neste caso multiplicações e somas. O modulo foi desenvolvido em uma *arquitetura paralela* e uma *arquitetura serial*.

Na Figura 6.3 é apresentada a *arquitetura paralela* para o cálculo da magnitude. Este bloco está composto por três multiplicadores encarregados de multiplicar cada componente da aceleração por si mesma para obter o quadrado do seu valor ($A_x \times A_x = A_x^2$), estas multiplicações são feitas em paralelo e ao terminar, seus resultados são somados mediante dos somadores. A saída obtida (*Saida_mag*) é enviada ao modulo *Filtragem e Discretização*.

A *arquitetura serial* do cálculo da magnitude é apresentada na Figura 6.4 e esta constituída por um multiplicador, um acumulador e um contador. Este bloco possui só uma entrada que recebe um a um os valores de aceleração capturados pelo acelerômetro. Cada valor ao ser inserido no bloco é multiplicado por si mesmo quando o sinal de *Start* seja igual a 1. O produto é somado com o valor que tem o acumulador,

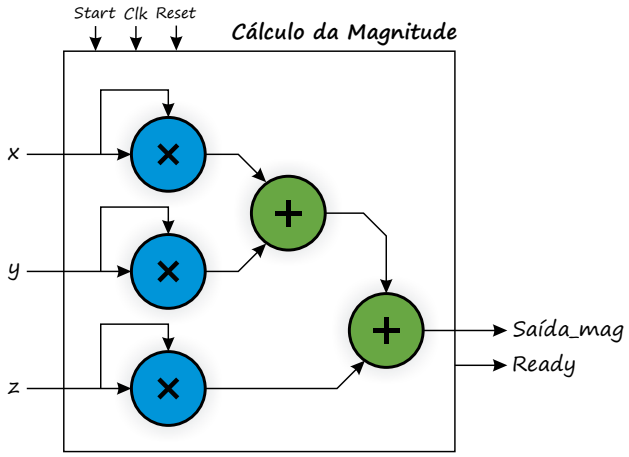



Figura 6.3: Bloco *Cálculo da magnitude* com *arquitetura paralela*.

que no início é zero. O resultado é registrado para ser utilizado na próxima operação. Este processo de multiplicação, soma e acumulação é repetido três vezes para obter a magnitude. O contador indica quando o resultado da saída (*Saida_mag*) está pronto, o qual é enviado ao seguinte bloco e o valor do acumulador muda novamente para zero por meio do sinal *rst_add*, o qual permitirá um novo cálculo da magnitude.

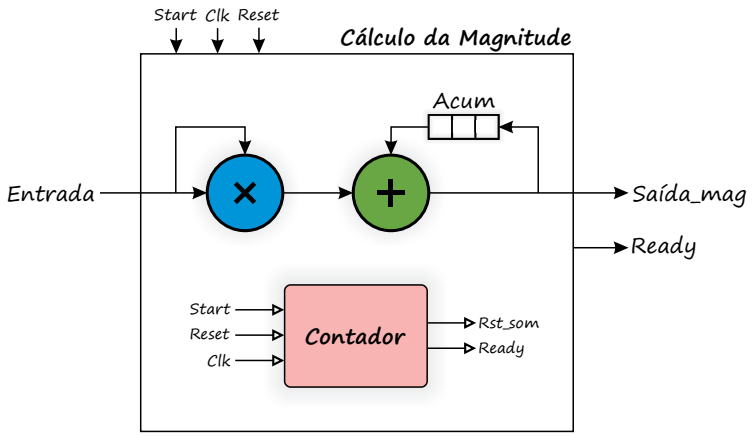



Figura 6.4: Bloco *Cálculo da magnitude* com *arquitetura serial*.

### 6.2.2 Bloco de filtragem e discretização

Após do cálculo da magnitude, o sinal é filtrado e discretizado mediante o bloco da Figura 6.5 nomeado *Filtragem e Discretização*. A arquitetura do bloco foi projetada a partir da Equação 5.2 e esta constituído por um acumulador, um multiplicador e um contador. Cada valor que chega ao bloco pela porta *Entrada_disc* é somado com o valor presente no acumulador, que no início é zero. O resultado da soma é registrado no acumulador para a próxima operação do somador. A quantidade de somas a realizar é controlada pelo contador e são estabelecidas por $n$, que é o comprimento ou número de dados da janela a ser filtrada. Após as $n$ somas, o resultado é inserido no multiplicador para ser multiplicado pela constante $1/n$ e o resultado do produto é apresentado na *Saída_disc* e o acumulador muda seu valor para zero, o qual

permite um novo filtrado e discretizado de $n$ novos valores. Este bloco foi projetado de maneira serial, devido a que o número de dados da entrada dependem do comprimento $n$, os quais não chegariam ao bloco no mesmo instante de tempo para ser processadas de maneira paralela, dado que o cálculo da magnitude da aceleração é realizado cada vez que é coletada uma nova amostra.

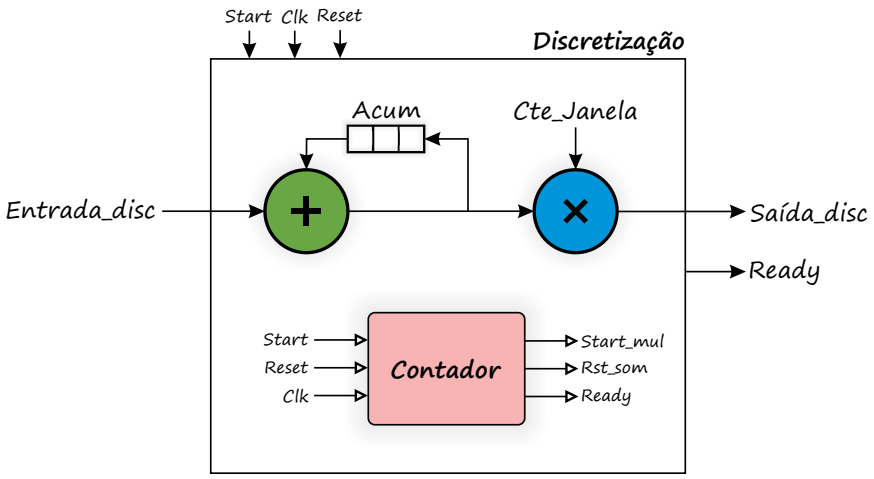



Figura 6.5: Bloco *Discretização*. O bloco *contador* realiza o janelamento do sinal em conjuntos de $n$ amostras.

### 6.2.3 Bloco de deslocamento

Um padrão de queda ou não-queda se obtém a partir do deslocamento das amostras atuais (filtradas e discretizadas), isto se faz repetidamente no bloco *Deslocador* da Figura 6.6 o qual é um registrador do tipo SIPO (*Single-input parallel-output*), onde as amostras filtradas e discretizadas são inseridas pela entrada *Ent_des*. Assim cada valor atual é deslocado e um novo valor é armazenado com o qual se obtém um sinal de oito dados que pode representar um padrão de queda. Por último, a saída *Ready* informa ao seguinte bloco cada vez que um deslocamento é realizado.

### 6.2.4 Bloco de normalização

A normalização de cada um dos dados da janela composta por oito amostras é realizada mediante a Equação 5.3, a qual foi reduzida matematicamente, substituindo os valores de saída $y_{min} = -1$ e $y_{max} = 1$, que foram determinados na fase de projeto da rede. O processo de redução é como segue:

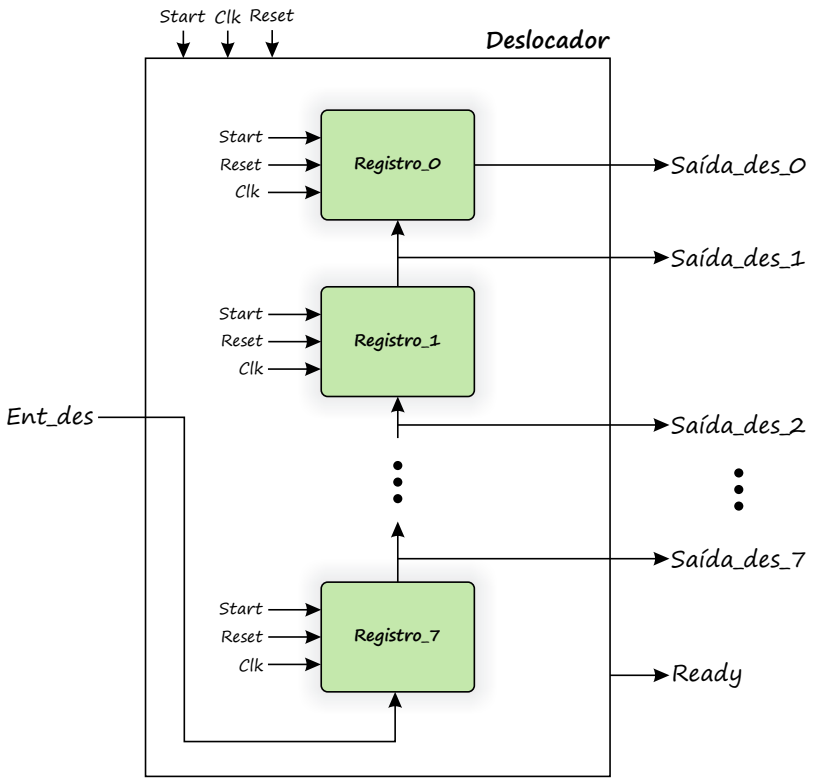



Figura 6.6: Bloco *Deslocador*.

$$y = \frac{2(x - x_{min})}{x_{max} - x_{min}} - 1$$

$$= \frac{2(x - x_{min}) - (x_{max} - x_{min})}{x_{max} - x_{min}}$$

$$= \frac{2x - 2x_{min} - x_{max} + x_{min}}{x_{max} - x_{min}}$$

$$= x\frac{2}{x_{max} - x_{min}} - \frac{x_{max} + x_{min}}{x_{max} - x_{min}}$$

A partir da redução é obtida a Equação 6.1. Onde o valor normalizado $y$ é conseguido mediante a multiplicação do valor a normalizar $x$ pela constante $a$ e a subtração da constante $b$.

$$y = xa - b, \tag{6.1}$$

onde as constantes $a$ e $b$ estão constituídas pelos valores fixos $x_{max}$ e $x_{min}$ de cada entrada da rede neural artificial. Estes valores também são determinados na etapa de treinamento.

$$a = \frac{2}{x_{max} - x_{min}} \qquad b = \frac{x_{max} + x_{min}}{x_{max} - x_{min}}$$

A Figura 6.7 apresenta a *Unidade de normalização* que esta composta por dois operadores, uma multiplicação e uma substração. O funcionamento é simples, cada vez que chega um dado, este é multiplicado pela *Cte_a* e depois é subtraída a *Cte_b*, obtendo assim o valor normalizado em *Saída_norm*.

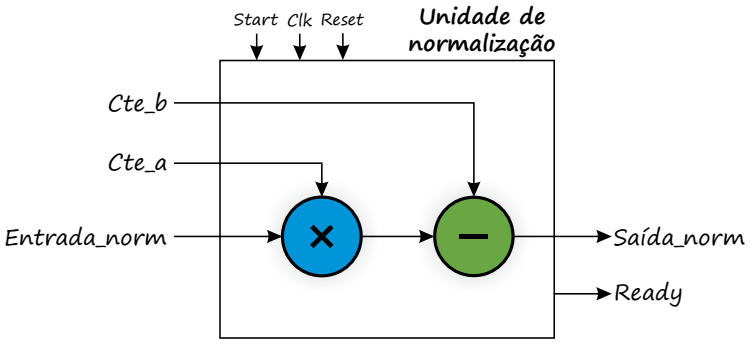



Figura 6.7: Bloco *Unidade de normalização*.

Com a *Unidade de normalização*, é construída o bloco normalizador da Figura 6.8. Este possui uma *arquitetura paralela* composta por oito unidades de normalização e oito valores diferentes das constantes $a$ e $b$. A finalização do cálculo da normalização das entradas, é informado mediante o sinal *Ready* e os valores normalizados são enviados ao bloco da rede neural artificial para sua respectiva avaliação.

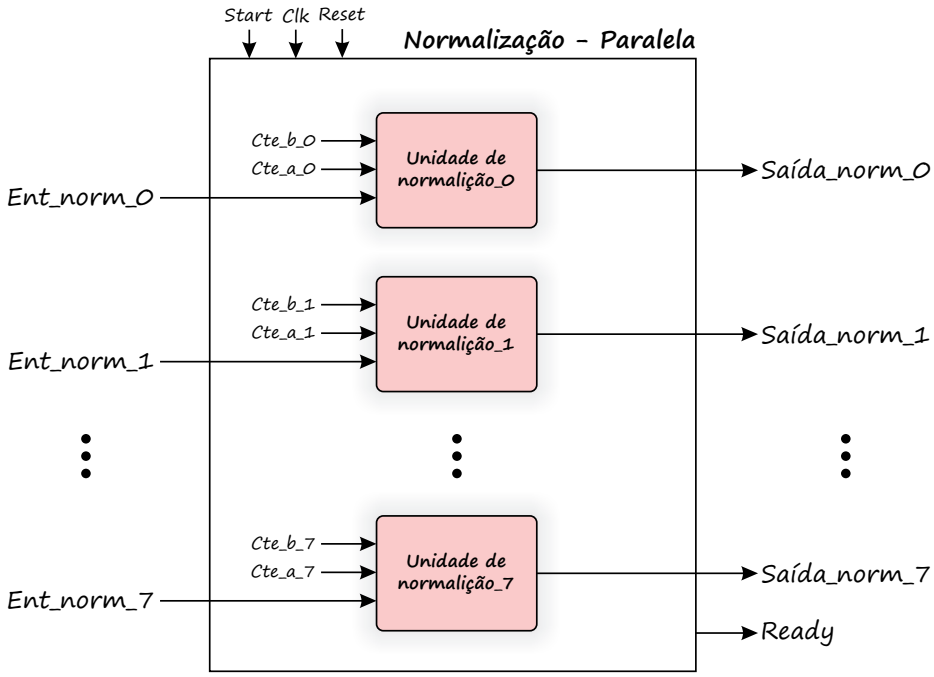



Figura 6.8: Bloco de *Normalização* com *arquitetura paralela*.

Uma *arquitetura serial* do bloco normalizador é mostrada na Figura 6.9, a qual está constituída por três multiplexadores de oito entradas, um multiplicador, uma subtração e uma FSM. Neste bloco as oito constantes $a$ e $b$ são inseridas nas entradas dos multiplexadores 1 e 3 respetivamente e os valores a normalizar no multiplexador 2.

O processo de multiplexação das entradas e constantes é controlado por uma FSM de dez estados, mostrada na Figura 6.10. Quando chegam os valores a normalizar, a FSM é ativada e no primeiro estado começa a normalização pela primeira entrada do bloco, deixando o sinal *Sel* dos multiplexadores em zero. Ao terminar o cálculo da substração, o resultado é enviado ao deslocador que armazena e desloca a posição de cada valor normalizado. No segundo estado, o $Sel$ muda para 1, se faz a normalização da segunda entrada, armazenagem e desloca novamente o resultado. O procedimento é repetido até processar os oito dados de entrada e ao finalizar no estado nove, a FSM informa mediante o sinal *Ready* que os cálculos terminaram. Na Tabela 6.1 são apresentadas as condições e sinais de controle presentes em cada estado da FSM.

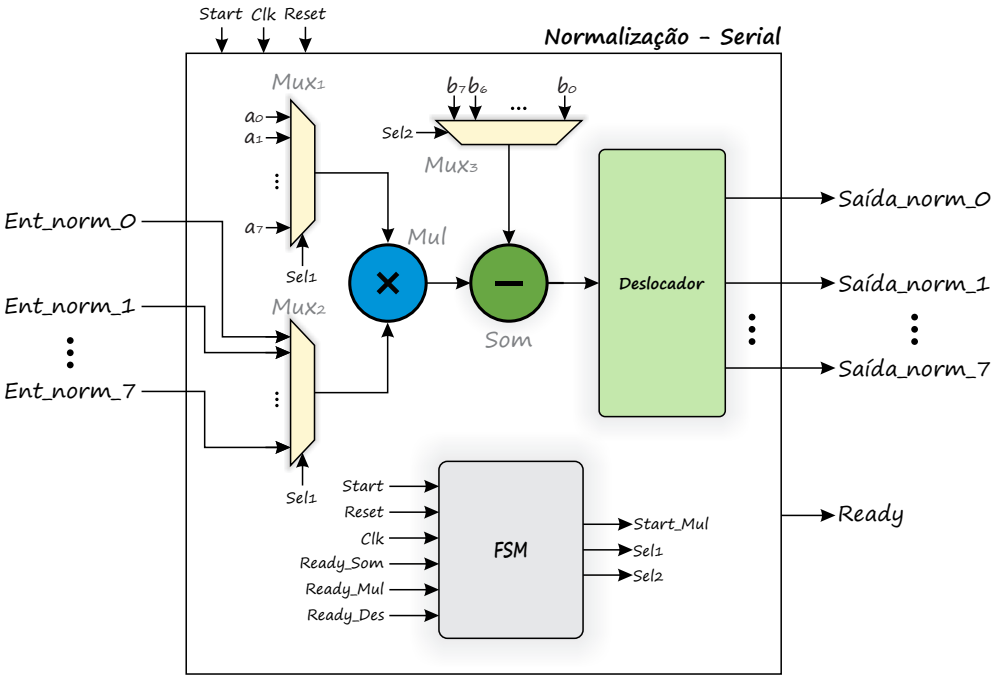



Figura 6.9: Bloco de *Normalização* com *arquitetura serial*.

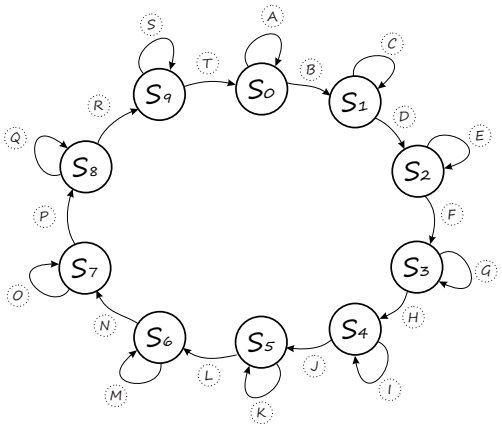



Figura 6.10: FSM do bloco de *Normalização* com *arquitetura serial*.

Tabela 6.1: Condições para a transições de estados do módulo *Normalização* com *arquitetura serial*.

| Abreviação | Desde | Até | Condição | Saída |
|---|---|---|---|---|
| **A** | S0 | S0 | start = ’0’ | sel1 = "000"<br>sel2 = "000"<br>ready = ’0’ |
| **B** | S0 | S1 | start = ’1’ | start_mul = ’1’ |
| **C** | S1 | S1 | ready_som = ’0’ | start_mul = ’0’ |
| **D** | S1 | S2 | ready_mul = ’1’ | sel1 = "001"<br>start_mul = ’1’ |

| | | | | |
|---|---|---|---|---|
| **E** | S2 | S2 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "001" |
| **F** | S2 | S3 | ready_mul = ’1’ | sel1 = "010"<br>start_mul = ’1’ |
| **G** | S3 | S3 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "010" |
| **H** | S3 | S4 | ready_mul = ’1’ | sel1 = "011"<br>start_mul = ’1’ |
| **I** | S4 | S4 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "011" |
| **J** | S4 | S5 | ready_mul = ’1’ | sel1 = "100"<br>start_mul = ’1’ |
| **K** | S5 | S5 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "100" |
| **L** | S5 | S6 | ready_mul = ’1’ | sel1 = "101"<br>start_mul = ’1’ |
| **M** | S6 | S6 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "101" |
| **N** | S6 | S7 | ready_mul = ’1’ | sel1 = "110"<br>start_mul = ’1’ |
| **O** | S7 | S7 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "110" |
| **P** | S7 | S8 | ready_mul = ’1’ | sel1 = "111"<br>start_mul = ’1’ |
| **Q** | S8 | S8 | (a) ready_som = ’0’ ou ready_mul = ’0’<br>(b) ready_som = ’1’ | (a) start_mul = ’0’<br>(b) sel2 = "111" |
| **R** | S8 | S9 | ready_mul = ’1’ | start_mul = ’0’ |
| **S** | S9 | S9 | ready_des = ’0’ | ready = ’0’ |
| **T** | S9 | S0 | ready_des = ’1’ | ready = ’1’ |

## 6.3 Implementação do módulo da rede neural artificial

A arquitetura geral da rede neural artificial projetada para o reconhecimento de padrões de queda, esta constituída pelas unidades fundamentais de processamento (neurônios). Um neurônio está composto pelo somatório das multiplicações dos pesos pelas entradas, representada mediante a Equação 2.1 e pela função de ativação da Equação 2.7. Estas duas equações se encontram mapeadas em hardware nos blocos *Combinação linear* e *Tangente Hiperbólica* respetivamente, os quais são mostrados na Figura 6.11

Este módulo possui uma entrada para o *bias*, oito entradas para os pesos ($w_0 : w_7$) e oito entradas ($Ent_0 : Ent_7$) para os valores a serem avaliados no neurônio. As operações são iniciadas mediante o sinal

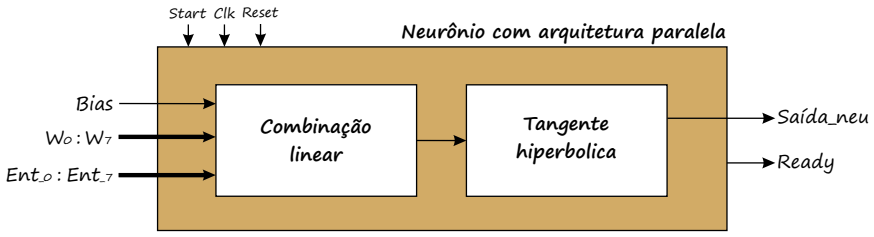



Figura 6.11: Arquitetura geral do módulo que representa um neurônio.

de *Start* e o resultado é avaliado no bloco *Tangente hiperbólica*. Quando se tem terminado a avaliação, o resultado desta é apresentado na *Saída_neu* e informa-se disto através do sinal *Ready*.

### 6.3.1 Bloco de combinação linear

A rede neural projetada neste trabalho está composta por oito neurônios na camada escondida e um na camada de saída. As nove unidades de processamento têm oito entradas e um peso por cada uma delas e uma entrada para o *bias*. Estes dados são processados de maneira paralela na Figura 6.12, que possui oito multiplicadores e somadores.

Para cada neurônio, os pesos $w\_0, w\_1, ..., w\_7$ são estabelecidos no treinamento da rede ao igual que o *bias*. Primeiro são realizadas de maneira paralela todas as multiplicações, os resultados destas são somados e o resultado das somas, é por último, somado com o *bias*. O resultado final das operações é apresentado na porta *Saída_ope*.

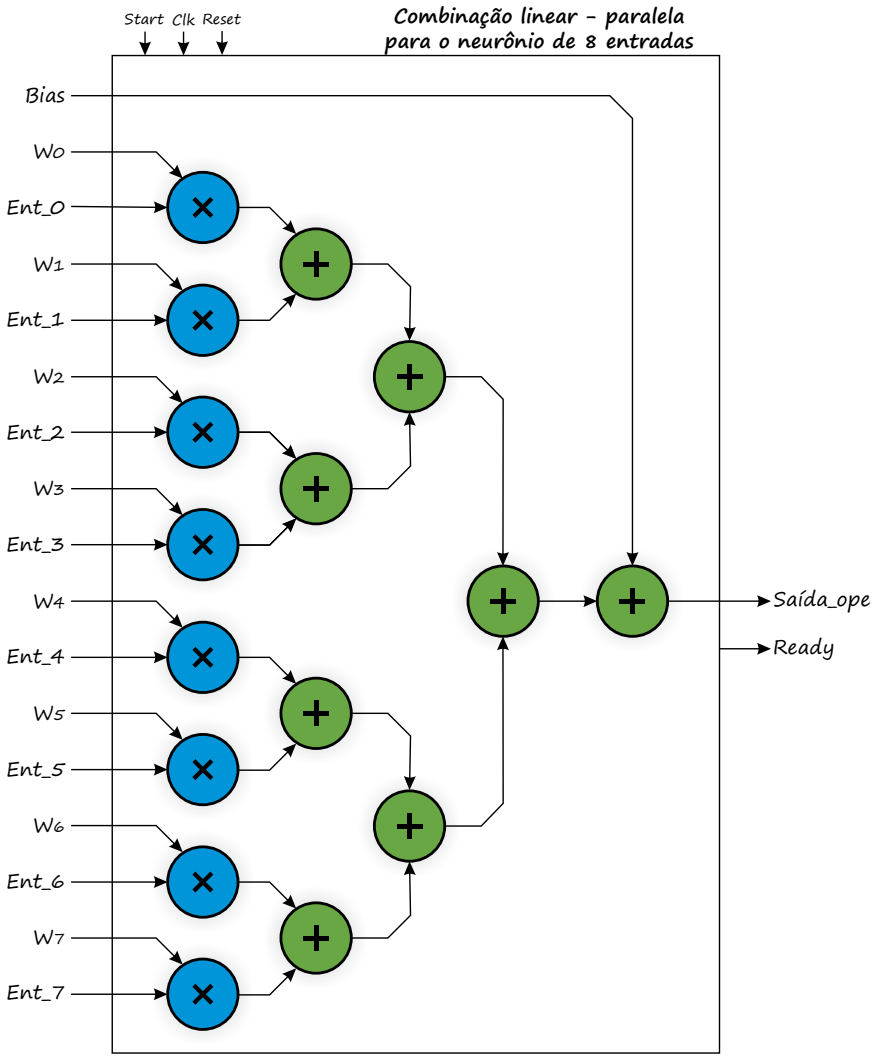



Figura 6.12: Bloco *Combinação linear* do neurônio - *arquitetura paralela*.

Uma *arquitetura semi-paralela* do bloco *Combinação linear* é mostrada na Figura 6.13, a qual foi projetada implementando quatro multiplexadores com quatro entradas, um multiplexador com três entradas, dois multiplicadores, um acumulador e uma FSM de dez estados encarregada de controlar o chaveamento dos multiplexadores e execução das operações. O início das operações se dá quando chegam os valores de $Ent\_$ e o sinal *Start* é inserido. Para o primeiro estado da FSM, o *bias* é somado com o valor presente no acumulador, que no início é zero, e o resultado da soma é armazenado no acumulador para as próximas operações. Os multiplexadores ligados aos multiplicadores enviam o dado presente no primeiro canal de cada um e estes são multiplicados. O $Mux_5$ faz o chaveamento para o segundo canal, que permite que o resultado do multiplicador $Mul_1$ seja somado com o valor presente no acumulador, e o resultado salvado no mesmo. No segundo estado o $Mux_5$ faz o chaveamento para o terceiro canal, com o qual o valor do multiplicador $Mul_2$ é somado com o valor do acumulador, e o resultado guardado no mesmo acumulador. O processo de chaveamento e execução das operações é realizado até operar as sete entradas restantes com os seus respetivos pesos. Ao finalizar os cálculos, a FSM restabelece o valor do acumulador para zero e informa através do sinal *Ready* para o seguente bloco, que o resultado está pronto. A Tabela 6.2 apresenta

as condições para a transição entre estados e os sinais de controle gerados em cada um deles.

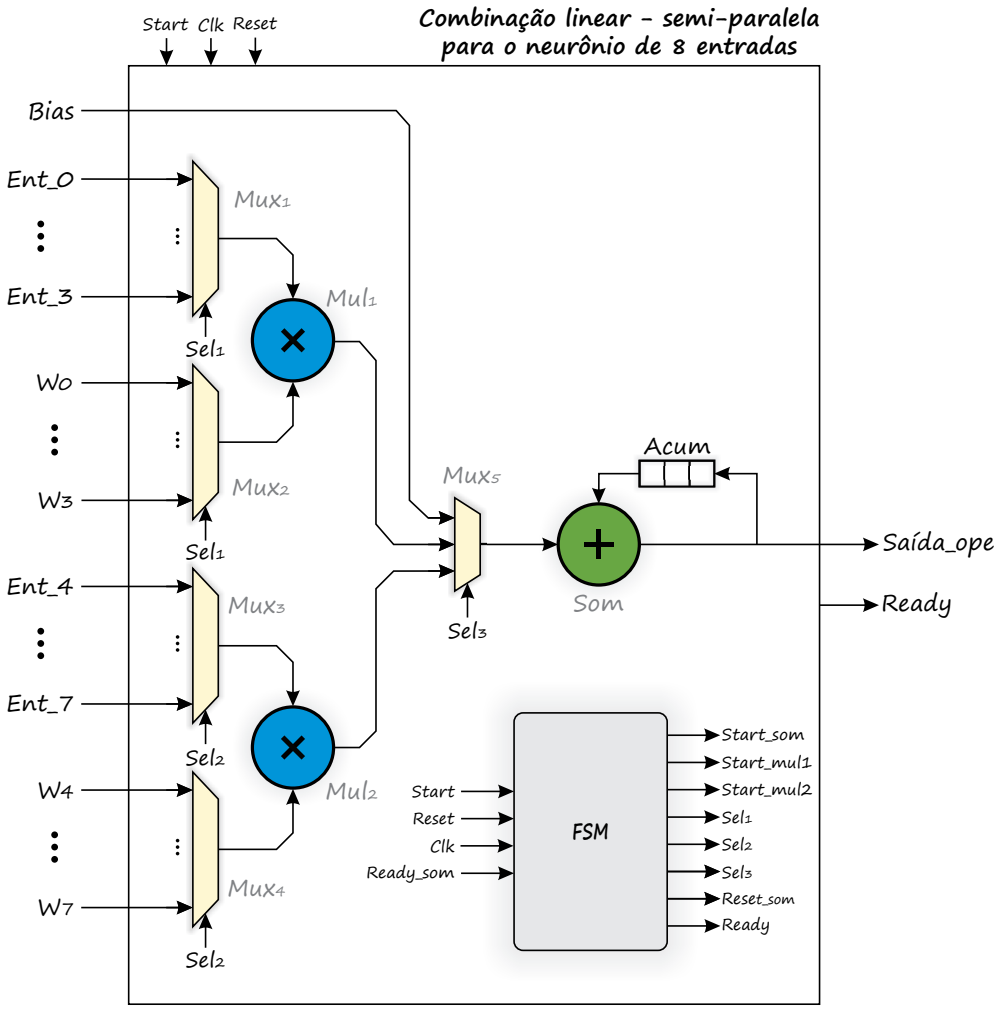



Figura 6.13: Bloco *Combinação linear* do neurônio - *arquitetura semi-paralela*.

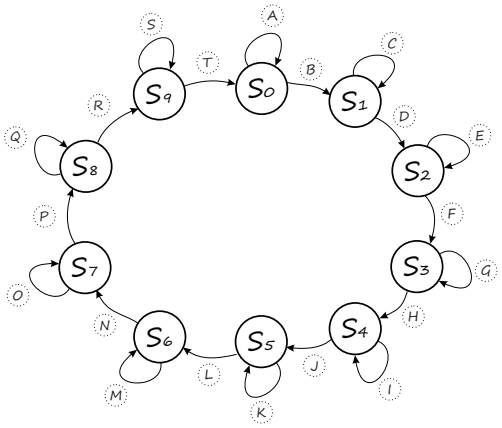



Figura 6.14: FSM do bloco de *Combinação linear* com *arquitetura semi-paralela*.

Tabela 6.2: Condições para as transições de estados do módulo *Combinação linear* com *arquitetura semi-paralela*.

| Abreviação | Desde | Até | Condição | Saída |
|---|---|---|---|---|
| **A** | S0 | S0 | start = ’0’ | sel1 = "00"<br>sel2 = "00"<br>sel3 = "00"<br>reset_som = ’0’<br>ready = ’0’ |
| **B** | S0 | S1 | start = ’1’ | start_mul1 = ’1’<br>start_mul2 = ’1’<br>start_som = ’1’ |
| **C** | S1 | S1 | ready_som = ’0’ | start_mul1 = ’0’<br>start_mul2 = ’0’<br>start_som = ’0’ |
| **D** | S1 | S2 | ready_som = ’1’ | sel1 = "01"<br>sel2 = "01"<br>sel3 = "01"<br>start_mul1 = ’1’<br>start_som = ’1’ |
| **E** | S2 | S2 | ready_som = ’0’ | start_mul1 = ’0’<br>start_som = ’0’ |
| **F** | S2 | S3 | ready_som = ’1’ | sel3 = "10"<br>start_mul2 = ’1’<br>start_som = ’1’ |
| **G** | S3 | S3 | ready_som = ’0’ | start_mul2 = ’0’<br>start_som = ’0’ |
| **H** | S3 | S4 | ready_som = ’1’ | sel1 = "10"<br>sel3 = "01"<br>start_mul1 = ’1’<br>start_som = ’1’ |
| **I** | S4 | S4 | ready_som = ’0’ | start_mul1 = ’0’<br>start_som = ’0’ |
| **J** | S4 | S5 | ready_som = ’1’ | sel2 = "10"<br>sel3 = "10"<br>start_mul2 = ’1’<br>start_som = ’1’ |
| **K** | S5 | S5 | ready_som = ’0’ | start_mul2 = ’0’<br>start_som = ’0’ |

| | | | | |
|---|---|---|---|---|
| **L** | S5 | S6 | ready_som = ’1’ | sel1 = "11"<br>sel3 = "01"<br>start_mul1 = ’1’<br>start_som = ’1’ |
| **M** | S6 | S6 | ready_som = ’0’ | start_mul1 = ’0’<br>start_som = ’0’ |
| **N** | S6 | S7 | ready_som = ’1’ | sel2 = "11"<br>sel3 = "10"<br>start_mul2 = ’1’<br>start_som = ’1’ |
| **O** | S7 | S7 | ready_som = ’0’ | start_mul2 = ’0’<br>start_som = ’0’ |
| **P** | S7 | S8 | ready_som = ’1’ | sel3 = "11"<br>start_som = ’1’ |
| **Q** | S8 | S8 | ready_som = ’0’ | start_som = ’0’ |
| **R** | S8 | S9 | ready_som = ’1’ | sel3 = "10"<br>start_som = ’1’ |
| **S** | S9 | S9 | ready_som = ’0’ | start_som = ’0’ |
| **T** | S9 | S0 | ready_som = ’1’ | ready = ’1’<br>reset_som = ’1’ |

Na Figura 6.15 é detalhada uma *arquitetura serial* composta por três multiplexadores, um acumulador, um multiplicador e uma FSM (vide Figura 6.16). Mediante o sinal do *Start* se dá início às operações de um grupo de oito entradas, o acumulador é inicializado sempre em zero e o controle dos multiplexadores e a execução das operações é realizado por uma FSM de dez estados. No primeiro estado, é realizada a multiplicação da entrada um por seu respetivo peso e o produto é somado com o valor presente no acumulador. O resultado da soma é guardado no acumulador para a seguente operação. A partir do segundo estado, são operadas as sete entradas restantes, fazendo o chaveamento respetivo dos multiplexadores $Mux_1$ e $Mux_2$ até o estado sete. No estado oito é realizado o chaveamento do $Mux_3$ para o segundo canal e é realizada a soma do valor do acumulador com o *bias*. No último estado o resultado da soma é inserido na saída do bloco $Saida\_ope$, a FSM informa da finalização das operações e o acumulador é restabelecido para zero. Os sinais de controle de cada estado da FSM são mostradas na Tabela 6.3.

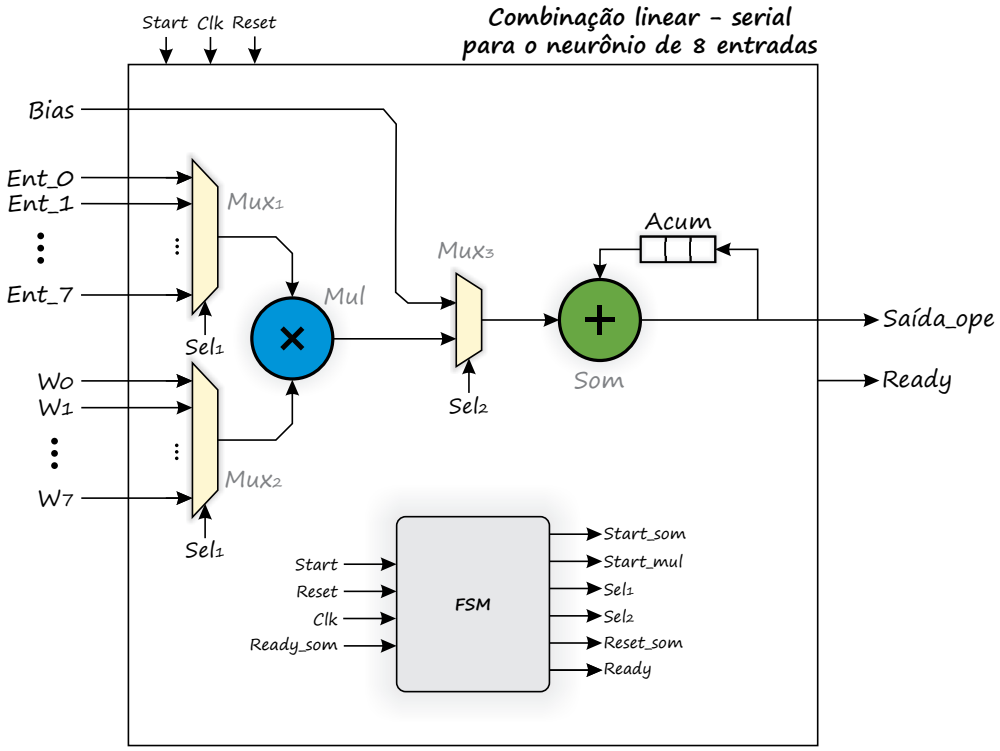



Figura 6.15: Bloco *Combinação linear* do neurônio - *arquitetura serial*.

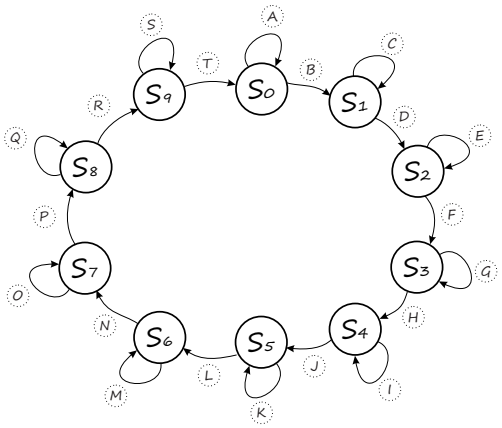



Figura 6.16: FSM do bloco de *Combinação linear* com *arquitetura serial*.

Tabela 6.3: Condições para a transições de estados do módulo *Combinação linear* com *arquitetura serial*.

| **Abreviação** | **Desde** | **Até** | **Condição** | **Saída** |
|---|---|---|---|---|
| **A** | S0 | S0 | start = ’0’ | sel1 = "000"<br>sel2 = ’0’<br>reset_som = ’0’<br>ready = ’0’ |
| **B** | S0 | S1 | start = ’1’ | start_mul = ’1’ |
| **C** | S1 | S1 | ready_som = ’0’ | start_mul = ’0’ |

| | | | | |
|---|---|---|---|---|
| **D** | S1 | S2 | ready_som = ’1’ | sel1 = "001"<br>start_mul = ’1’ |
| **E** | S2 | S2 | ready_som = ’0’ | start_mul = ’0’ |
| **F** | S2 | S3 | ready_som = ’1’ | sel1 = "010"<br>start_mul = ’1’ |
| **G** | S3 | S3 | ready_som = ’0’ | start_mul = ’0’ |
| **H** | S3 | S4 | ready_som = ’1’ | sel1 = "011"<br>start_mul = ’1’ |
| **I** | S4 | S4 | ready_som = ’0’ | start_mul = ’0’ |
| **J** | S4 | S5 | ready_som = ’1’ | sel1 = "100"<br>start_mul = ’1’ |
| **K** | S5 | S5 | ready_som = ’0’ | start_mul = ’0’ |
| **L** | S5 | S6 | ready_som = ’1’ | sel1 = "101"<br>start_mul = ’1’ |
| **M** | S6 | S6 | ready_som = ’0’ | start_mul = ’0’ |
| **N** | S6 | S7 | ready_som = ’1’ | sel1 = "110"<br>start_mul = ’1’ |
| **O** | S7 | S7 | ready_som = ’0’ | start_mul = ’0’ |
| **P** | S7 | S8 | ready_som = ’1’ | sel1 = "111"<br>start_mul = ’1’ |
| **Q** | S8 | S8 | ready_som = ’0’ | start_mul = ’0’ |
| **R** | S8 | S9 | ready_som = ’1’ | sel2 = ’1’<br>start_mul = ’1’ |
| **S** | S9 | S9 | ready_som = ’0’ | start_mul = ’0’ |
| **T** | S9 | S0 | ready_som = ’1’ | ready = ’1’<br>reset_som = ’0’ |

### 6.3.2 Bloco tangente hiperbólica

Depois das operações o valor obtido é avaliado na função tangente hiperbólica representada pela Equação 6.2, sendo está um equivalente da função apresentada na Subseção 2.1.2

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{6.2}$$

O bloco da função de ativação do neurônio é mostrado na Figura 6.17, criado a partir de uma multiplicação, uma função exponencial, uma divisão, um multiplexador e um único modulo para as operações soma e subtração. Além disso, uma FSM encarregada dos cálculos e chaveamento, cuja estrutura se encontra na Figura 6.18 e na Tabela 6.4 descrevem-se os sinais de controle gerados em cada estado.

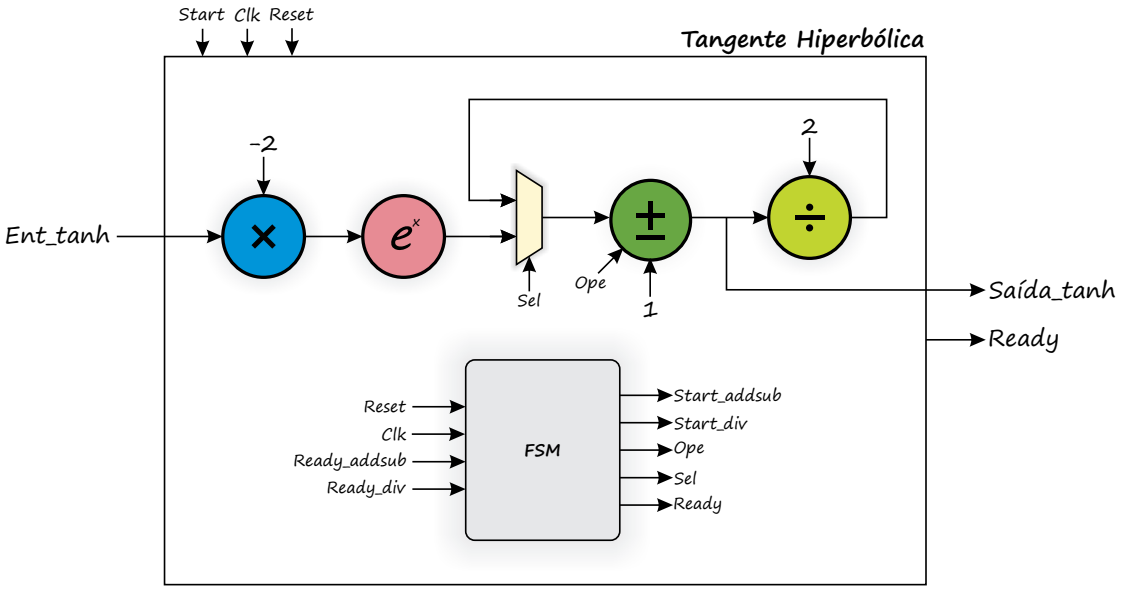



Figura 6.17: Bloco *Tangente hiperbólica*.

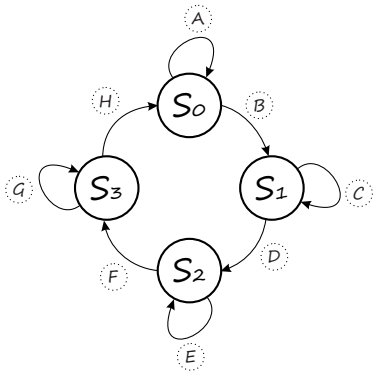



Figura 6.18: FSM do bloco *Tangente hiperbólica*.

A avaliação do valor de entrada *Ent_tanh*, inicia quando com o sinal de *Start*. Primeiro é realizada a multiplicação pela constante $-2$, o produto é avaliado no modulo do exponencial (baseado no algoritmo CORDIC [83]) e o sinal *Ready_exp* é o encarregado de iniciar a FSM para as seguintes operações. O resultado do expoente é somado (*ope*='0') com a constante 1, que é iniciada pelo sinal *Start_addsub*. Logo, a constante 2 é dividida pelo resultado da soma e por último é realizada subtração (*ope*='1') de 1 ao resultado da divisão, com o que se obtém a saída da função no sinal *Saída_tanh*, que é confirmada por um nível alto do sinal de *Ready*.

Tabela 6.4: Condições para as transições de estados do módulo *Tangente hiperbólica*.

| **Abreviação** | **Desde** | **Até** | **Condição** | **Saída** |
|---|---|---|---|---|
| **A** | S0 | S0 | ready_exp = '0' | sel = '0'<br>ope = '0'<br>ready = '0' |

| | | | | |
|---|---|---|---|---|
| **B** | S0 | S1 | ready_exp = '1' | start_som = '1' |
| **C** | S1 | S1 | ready_som = '0' | start_som = '0' |
| **D** | S1 | S2 | ready_som = '1' | start_div = '1' |
| **E** | S2 | S2 | ready_div = '0' | start_div = '0' |
| **F** | S2 | S3 | ready_div = '1' | sel = '1'<br>ope = '1'<br>start_som = '1' |
| **G** | S3 | S3 | ready_som = '0' | start_som = '0' |
| **H** | S3 | S0 | ready_som = '1' | ready = '1' |

### 6.3.3 Rede neural

Para a construção da rede neural artificial, são interligados nove neurônios. Oito pertencentes à camada oculta e um à camada de saída. Na Figura 6.19, é mostrada a arquitetura que representa a implementação realizada da rede neural artificial, para um processamento de dados paralelo e paralelo-serial. Na primeira, os neurônios ($Neu\_x$) estão constituídos pela *arquitetura paralela* do bloco de *combinação linear* da Figura 6.12 e a segunda pela *arquitetura semi-paralela* do bloco de *combinação linear* da Figura 6.13. Os dois tipos de neurônios utilizam o bloco *Tangente hiperbólica*. Os valores dos pesos e *bias* para cada neurônio, são estabelecidos como constantes em cada um destes. Para o funcionamento da rede, os oito dados a serem avaliados, provenientes do bloco de pré-processamento de dados são inseridos nas entradas $Ent\__0 : Ent\__7$ e o início da execução dos cálculos é dado pelo sinal *Start*. O resultado é apresentado na saída *Saída_red* e a finalização da avaliação é informada pela saída *Ready*.

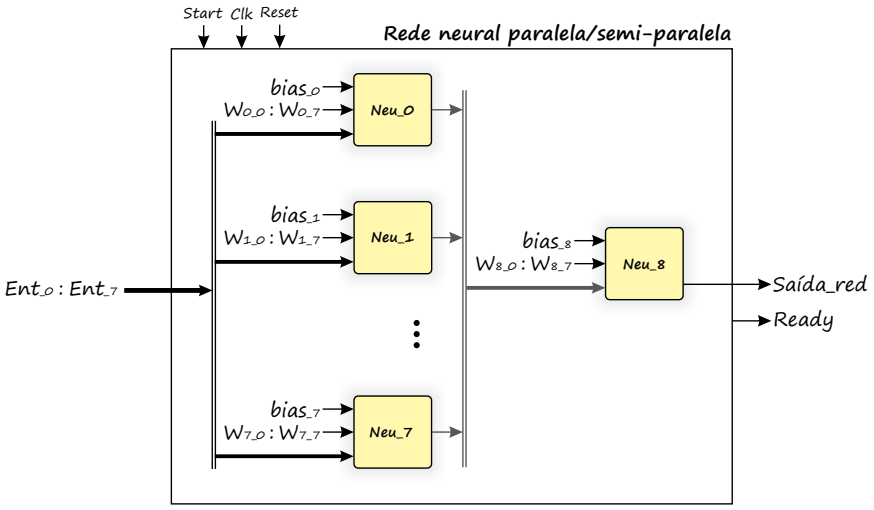



Figura 6.19: Bloco rede neural paralela/semi-paralela.

Una rede neural artificial com um processamento de dados de maneira serial é apresentada na Figura 6.20. Esta arquitetura está constituída pela *arquitetura serial* do bloco de *Combinação linear* da Figura 6.15, un bloco *Tangente hiperbólica*, um multiplicador, um acumulador, doce multiplexadores e uma FSM para seu controle. Nesta arquitetura, o bloco de *Combinação linear* é aproveitado para o respec-

tivo cálculo dos neurônios da camada oculta. Nas entradas deste bloco estão ligados nove multiplexadores de oito canais, um para os $bias_i$ de cada neurônio (onde $i \in [0, 8]$) e oito para os pesos de cada neurônio e cada entrada $w_{i\_j}$ (onde $j \in [0, 7]$). Os dados a serem avaliados pela rede, são inseridos nas entradas $Ent\_0 : Ent\_7$. O início da avaliação é dado pelo sinal *Start*.

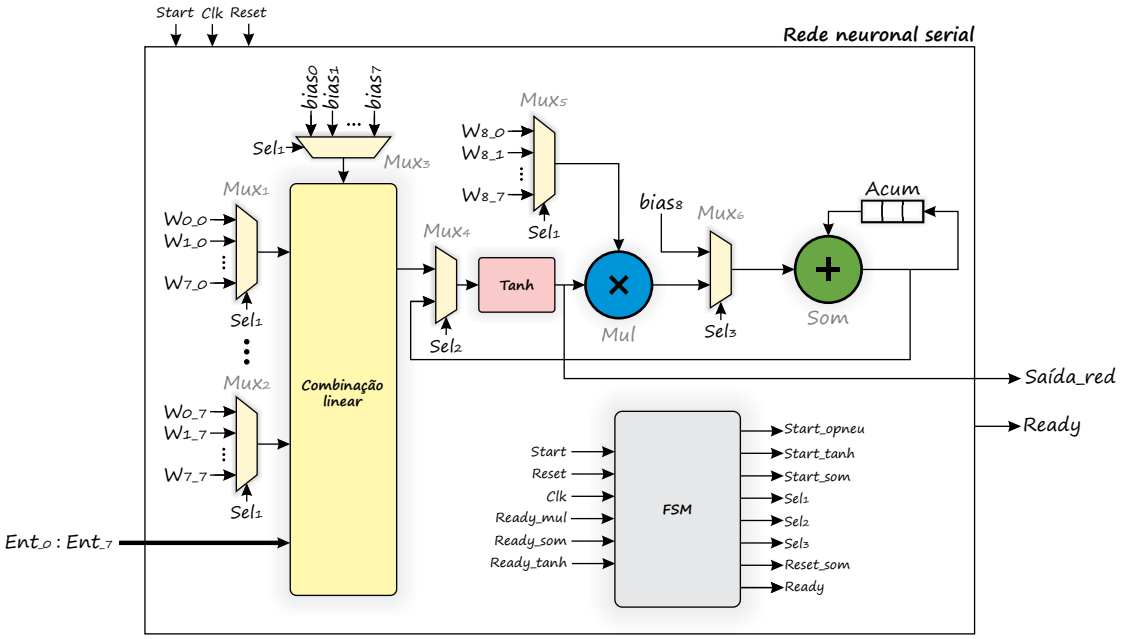



Figura 6.20: Bloco *rede neural* com *arquitetura serial*.

A FSM mostrada na Figura 6.21, está baseada em onze estados. No primeiro estado, o $bias_8$ do neurônio de saída é somado com o valor inicial do acumulador, que inicialmente é zero, e o resultado fica guardado no acumulador. É iniciada a execução da combinação linear entre as entradas, pesos e *bias* do primeiro neurônio oculto. O resultado é avaliado no bloco $Tanh$ e o produto da avaliação é multiplicado pelo primeiro peso $w_{8\_0}$ do neurônio da camada de saída. A saída do multiplicador é multiplexada para depois ser somada com o valor do acumulador. O resultado da soma é armazenado no registrador do acumulador. No segundo estado, o *bias* e os pesos ligados ao bloco de *combinação linear*, são multiplexados para o segundo valor, que correspondem aos dados do segundo neurônio da camada oculta. O bloco de *combinação linear* realiza sua execução com os dados de entrada, o resultado é avaliado no bloco $Tanh$, a avaliação é multiplicada pelo segundo valor dos pesos do neurônio de saída, o resultado da multiplicação é somado com o valor presente no acumulador e a soma é novamente acumulada. Para os estados dos neurônios três a oito, se repete o processo do estado dois para completar as operações correspondentes aos oito neurônios da camada escondida. No estado nove, a entrada da $Tanh$ é multiplexada, e nela é inserida o último valor armazenado no acumulador. Este é avaliado e o resultado é enviado à saída $Saida\_red$. No estado dez é restabelecido a zero o valor do acumulador e é enviado o sinal $Ready$ informando que os cálculos terminaram. Na Tabela 6.5 são descritas as diferentes sinais da FSM.

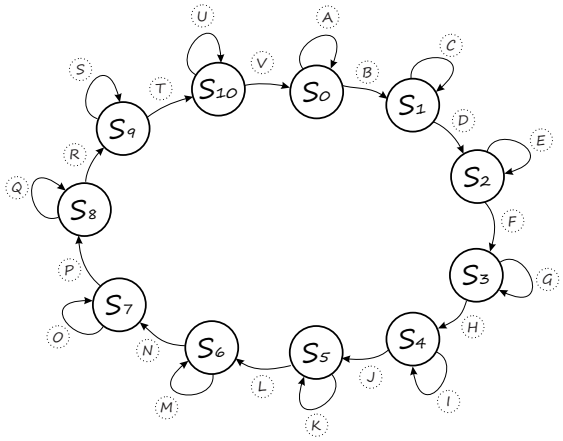



Figura 6.21: FSM da rede neural artificial com *arquitetura serial*.

Tabela 6.5: Condições para a transições de estados do módulo *rede neural* com *arquitetura serial*.

| Abreviação | Desde | Até | Condição | Saída |
|---|---|---|---|---|
| **A** | S0 | S0 | start = ’0’ | sel1 = "000"<br>sel2 = ’0’<br>sel3 = ’0’<br>reset_som = ’0’<br>ready = ’0’ |
| **B** | S0 | S1 | start = ’1’ | start_som = ’1’<br>start_ope = ’1’ |
| **C** | S1 | S1 | (a) ready_mul = ’0’ ou ready_tanh = ’0’<br>(b) ready_tanh = ’1’ | (a) start_som = ’0’ e start_ope = ’0’<br>(b) sel3 = ’1’ |
| **D** | S1 | S2 | ready_mul = ’1’ | sel1 = "001"<br>start_ope = ’1’ |
| **E** | S2 | S2 | ready_mul = ’0’ | start_ope = ’0’ |
| **F** | S2 | S3 | ready_mul = ’1’ | sel1 = "010"<br>start_ope = ’1’ |
| **G** | S3 | S3 | ready_mul = ’0’ | start_ope = ’0’ |
| **H** | S3 | S4 | ready_mul = ’1’ | sel1 = "011"<br>start_ope = ’1’ |
| **I** | S4 | S4 | ready_mul = ’0’ | start_ope = ’0’ |
| **J** | S4 | S5 | ready_mul = ’1’ | sel1 = "100"<br>start_ope = ’1’ |
| **K** | S5 | S5 | ready_mul = ’0’ | start_ope = ’0’ |
| **L** | S5 | S6 | ready_mul = ’1’ | sel1 = "101"<br>start_ope = ’1’ |
| **M** | S6 | S6 | ready_mul = ’0’ | start_ope = ’0’ |

| | | | | |
|---|---|---|---|---|
| **N** | S6 | S7 | ready_mul = ’1’ | sel1 = "110"<br>start_ope = ’1’ |
| **O** | S7 | S7 | ready_mul = ’0’ | start_ope = ’0’ |
| **P** | S7 | S8 | ready_mul = ’1’ | sel1 = "111"<br>start_ope = ’1’ |
| **Q** | S8 | S8 | ready_mul = ’0’ | start_ope = ’0’ |
| **R** | S8 | S9 | ready_mul = ’1’ | sel2 = ’1’ |
| **S** | S9 | S9 | ready_som = ’0’ | start_tanh = ’0’ |
| **T** | S9 | S10 | ready_som = ’1’ | start_tanh = ’1’ |
| **U** | S10 | S10 | ready_som = ’0’ | start_tanh = ’0’ |
| **V** | S10 | S0 | ready_tanh = ’1’ | reset_som = ’1’<br>ready = ’1’ |

Após de projetar cada módulo do detector de quedas, a *arquitetura paralela* de este módulo é construída empregando um bloco de pré-processamento de dados, constituído por um bloco de cálculo da magnitude em paralelo, um bloco de discretização e filtragem, um bloco de normalização de maneira paralela e uma rede neural com o processamento de dados em paralelo. Para arquitetura *semi-paralela* é utilizado um bloco para o cálculo da magnitude em paralelo, um bloco de discretização e filtragem, um bloco de normalização de maneira paralela e uma rede neural artificial que tem uma distribuição de processos de maneira paralela e serial. Por último, para a *arquitetura serial* é empregado um módulo para o cálculo da magnitude, um bloco de discretização e filtragem, um normalizador e uma rede neural artificial, todas elas com um processamento de dados de maneira serial.

## 6.4 Arquitetura do avaliador de limiares

A partir do algoritmo 1 é projetada a arquitetura do avaliador de limiares mostrado na Figura 6.22, a qual está baseada em uma serie de condicionais. Para o seu funcionamento, inicialmente o contador do avaliador está em zero. Quando chega uma amostra de som junto com o sinal $Start$ é avaliado o valor presente no contador. Ao cumprir a condição de estar em zero, é avaliado se a amostra de som ultrapassa os limiares $Ls$ e $Li$ e, se algum destes é alcançado, o contador é incrementado e a porta *Saída_aval* informa que se detectou um nível de intensidade de som que corresponde a uma possível queda. Por outro lado, ao não serem alcançados os limiares, o módulo continua com a avaliação de uma nova amostra. Caso o contador seja maior a zero, a cada nova amostra se incrementa o valor do mesmo. Quando o valor do contador alcança o valor de 75 (75 amostras avaliadas) o seu valor é restabelecido e a porta *Saída_aval* muda seu valor para o zero.

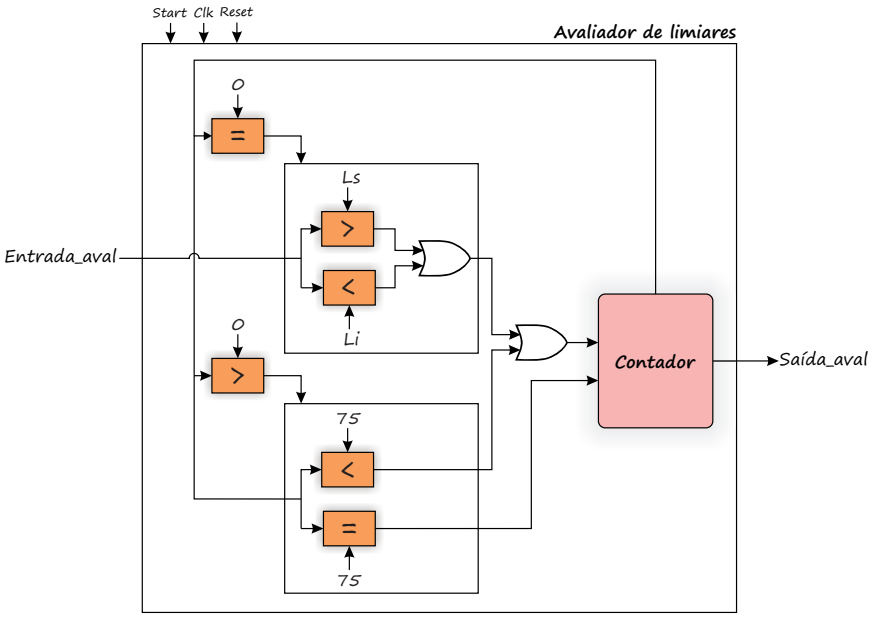



Figura 6.22: Circuito avaliador de limiares de som. Durante uma queda o som produzido quando o usuário impacta uma superfície rígida é detectado aproximadamente na metade do padrão estabelecido para uma queda. Isto corresponde, aproximadamente, a 75 amostras de som.

## 6.5 Arquitetura geral do sistema de detecção de quedas

Arquitetura geral do sistema de detecção de quedas é mostrado na Figura 6.23 e esta constituído pelos seguintes módulos: (a) processador *ARM Cortex A9*, (b) Acelerômetro, (c) *Bluetooth*, (d) Microfone e (e) o módulo de detecção de quedas.

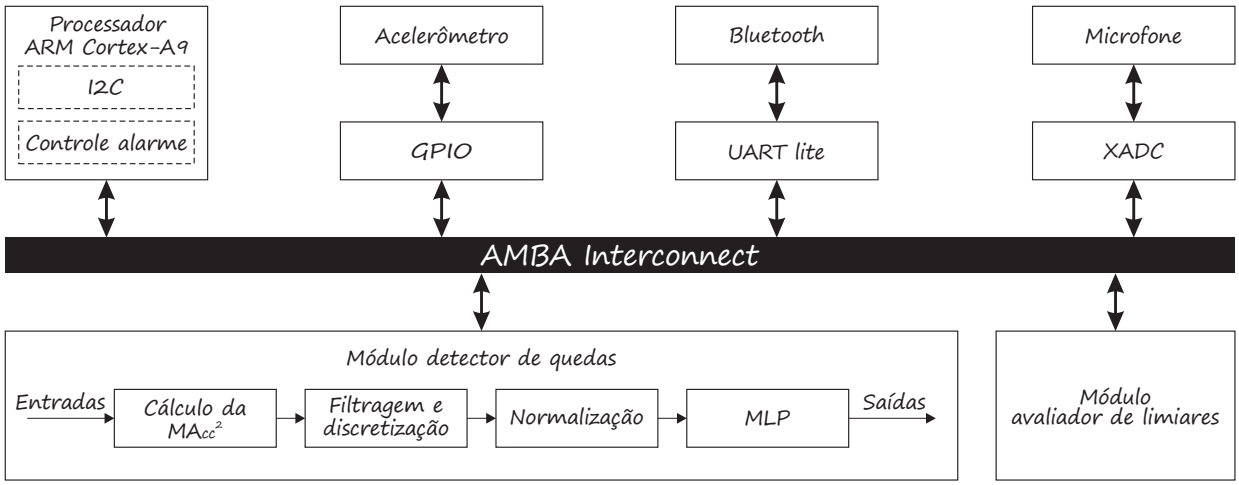



Figura 6.23: Arquitetura geral do sistema de detecção de quedas. O bloco *Deslocador* apresentado na Subseção 6.2.3 foi agregado ao bloco *Filtragem e Discretização*.

No processador *ARM Cortex A9* é implementado em linguagem C o protocolo de comunicação I2C e o controle do alarme. Mediante o protocolo são obtidos os dados capturados pelo acelerômetro que está conectado às entradas/saídas de propósito geral (GPIO, *General Purpose Input/Output*). Este, por sua

vez, está ligado ao AMBA *Interconnects*, que permite a comunicação entre o processador e os diversos periféricos.

Os dados de aceleração dos eixos $x$, $y$ e $z$, são enviados ao módulo detector de quedas. Neste, primeiro é calculado a ${MA_{cc}}^2$, o resultado é filtrado e discretizado no tempo, para, após da normalização, ser avaliado na rede neuronal artificial. O microfone envia um sinal de tensão para o FPGA o qual é interpretado pelo conversor analógico-digital on-chip (XADC) de 12 bits e os valores obtidos são avaliados no módulo *avaliador de limiares*. É importante ressaltar que neste trabalho o módulo *avaliador de limiares* usa o sinal de som para confirmar se aconteceu uma queda após a mesma ser detectada pela rede neural artificial a partir dos dados de aceleração. Isto foi feito no intuito de minimizar o número de falsos positivos.

Dessa maneira, quando um padrão de queda é reconhecido pela rede (saída: 1 queda, -1 não queda) e os limiares de som são alcançados, o controle do alarme envia uma mensagem de alerta para um celular, empregando o módulo *bluetooth* que está ligado ao bloco UARTLite da Xilinx, configurado com uma velocidade de transmissão de 115200bps. No celular foi implementado um aplicativo Android, encarregado de comunicar mediante uma mensagem de texto para o cuidador, organismo de socorro ou familiar do usuário, que uma queda aconteceu. Alem disso, permite guardar 200 dados de cada sensor, coletados dos movimentos de queda, assim como dos falsos positivos que eventualmente tenham sido gerados. Um movimento é considerado um falso positivo se a rede neural artificial detecta uma queda, mas a mesma não é confirmada pelo módulo *avaliador de limiares* do som. Os dados guardados servirão num futuro para retreinar a rede com dados reais de quedas e falsos positivos.

# Capítulo 7

# Resultados

Neste capítulo são apresentadas diversas comparações entre as arquiteturas projetadas com o objetivo de escolher a mais adequada para ser implementa no FPGA. Adicionalmente, mostram-se os resultados da implementação do sistema de detecção de quedas com e sem microfone e sua respectiva avaliação feita por cinco usuários.

## 7.1 Caracterização das arquiteturas embarcadas no FPGA

Com o objetivo de verificar a precisão dos cálculos e a funcionalidade adequada dos blocos projetados para as diferentes arquiteturas (*Paralela*, *Semi-paralela*, *Serial*), foi realizada uma comparação numérica via simulação, entre os resultados obtidos (com uma representação de dados em ponto flutuante de 27 bits) e os resultados em Matlab (do tipo *double* 64 bits). Na Tabela 7.1 é apresentado o erro MSE entre essas duas abordagens (hardware e software), obtido a partir da avaliação de 1000 dados de entrada para cada bloco.

Tabela 7.1: Erro MSE dos módulos implementados usando o Matlab como modelo de referência. $NA$ indica que o bloco não existe para a respectiva arquitetura.

| Módulo | Arquitetura | | |
|---|---|---|---|
| | *Paralela* | *Semi-Paralela* | *Serial* |
| Cálculo magnitude | 5,2897478E-11 | $NA$ | 5,2897478E-11 |
| Filtragem e discretização | $NA$ | $NA$ | 1,1929375E-10 |
| Normalizador | 2,4626240E-12 | $NA$ | 2,4626240E-12 |
| Tangente hiperbólica | $NA$ | $NA$ | 8,6651749E-03 |
| Rede Neural | 1,3185561E-10 | 1,178465E-10 | 1,0497153E-10 |
| Detector de quedas (Pré-processamento de dados e Rede neural artificial) | 5,0241969E-11 | 1,988893E-10 | 4,7710056E-11 |

Observa-se que o maior valor de MSE para o bloco geral de detecção de quedas é da ordem de $10^{-10}$, com o qual, pode-se concluir que as arquiteturas têm uma alta precisão nos cálculos para determinar os diferentes padrões de movimentos, utilizando uma representação de ponto flutuante de 27 bits.

Os circuitos do sistema foram sintetizados na ferramenta de desenvolvimento *Vivado 2016.4*. A utilização de recursos no SoC Zynq Z7-20 (chip xc7z020clg400-1) por parte de cada bloco que constitui o detector de quedas das arquiteturas *Paralela*, *Semi-paralela* e *Serial*, é mostrado na Tabela 7.2, Tabela 7.3 e Tabela 7.4, respetivamente e na Tabela 7.5 o consumo do bloco *Avaliador de limiares*. Os recursos utilizados são apresentados em termos de LUTs, FFs e DSPs. Além disso, é mostrado a porcentagem dos recursos utilizados com respeito ao total de recursos do FPGA. De acordo com as tabelas apresentadas, o avaliador de limiares de som é quem menos componentes de *hardware* utiliza com um 0.18% de LUTs e um 0.20% de FFs e das arquiteturas propostas para o *Detector de quedas*, a *arquitetura serial* apresenta um consumo de 7,8% de LUT, 1.44% de FF e 3.18% de DSPs.

Tabela 7.2: Consumo de recursos do *Detector de quedas* com *arquitetura paralela*

| Módulo | **LUT** (53200) | | **Flip-Flops** (106400) | | **DSP** (220) | |
|---|---|---|---|---|---|---|
| **Top module detector de quedas (Arq. Paralela)** | 45131 | 84,83% | 12140 | 11,41% | 110 | 50,00% |
| **Pré-processamento de dados (Arq. Paralela)** | 4773 | 8,97% | 1097 | 1,03% | 11 | 5,00% |
| Cálculo da magnitude (Arq. Parallela) | 646 | 1,21% | 181 | 0,17% | 3 | 1,36% |
| Filtragem e discretização | 542 | 1,02% | 292 | 0,27% | 0 | 0,00% |
| Normalizador (Arq. Paralela) | 3066 | 5,76% | 624 | 0,59% | 8 | 3,64% |
| **Rede neural artificial (Arq. Paralela)** | 40362 | 75,87% | 11043 | 10,38% | 99 | 45,00% |
| Combinação linear (Arq. Paralela) | 2611 | 4,91% | 636 | 0,60% | 8 | 3,64% |
| Tangente hiperbólica | 1840 | 3,46% | 590 | 0,55% | 3 | 1,36% |

Tabela 7.3: Consumo de recursos do *Detector de quedas* com *arquitetura semi-paralela*

| Módulo | **LUT** (53200) | | **Flip-Flops** (106400) | | **DSP** (220) | |
|---|---|---|---|---|---|---|
| **Top module detector de quedas (Arq. Semi-paralela)** | 23670 | 44,49% | 7449 | 7,00% | 49 | 22,27% |
| **Pré-processamento de dados (Arq. Semi-paralela)** | 2399 | 4,51% | 780 | 0,73% | 4 | 1,82% |
| Cálculo da magnitude (Arq. Paralela) | 738 | 1,39% | 181 | 0,17% | 3 | 1,36% |
| Filtragem e discretização | 287 | 0,54% | 292 | 0,27% | 0 | 0,00% |
| Normalizador (Arq. Serial) | 1208 | 2,27% | 307 | 0,29% | 1 | 0,45% |
| **Rede neural artificial (Arq. Semi-paralela)** | 21352 | 40,14% | 6669 | 6,27% | 45 | 20,45% |
| Combinação linear (Arq. Semi-paralela) | 564 | 1,06% | 150 | 0,14% | 2 | 0,91% |
| Tangente hiperbólica | 1808 | 3,40% | 590 | 0,55% | 3 | 1,36% |

O mapeamento do circuito realizado no FPGA pelo *Vivado 2016.4* é mostrado na Figura 7.1. Nesta, pode-se apreciar a área ocupada por cada um dos módulos das três arquiteturas do sistema de detecção de quedas e o avaliador de limiares. Como esperado, a *arquitetura serial* ocupa menos área no FPGA.

Tabela 7.4: Consumo de recursos do *Detector de quedas* com *arquitetura serial*

| Módulo | **LUT** (53200) | | **Flip-Flops** (106400) | | **DSP** (220) | |
|---|---|---|---|---|---|---|
| **Top module detector de quedas (Arq. Serial)** | 3875 | 7,28% | 1535 | 1,44% | 7 | 3,18% |
| **Pré-processamento de dados (Arq. Semi-paralela)** | 1133 | 2,13% | 706 | 0,66% | 2 | 0,91% |
| Cálculo da magnitude (Arq. Serial) | 426 | 0,80% | 107 | 0,10% | 1 | 0,45% |
| Filtragem e discretização | 304 | 0,57% | 292 | 0,27% | 0 | 0,00% |
| Normalizador (Arq. Serial) | 405 | 0,76% | 307 | 0,29% | 1 | 0,45% |
| **Rede neural artificial (Arq. Serial)** | 2729 | 5,13% | 829 | 0,78% | 5 | 2,27% |
| Combinação linear (Arq. Serial) | 448 | 0,84% | 118 | 0,11% | 1 | 0,45% |
| Tangente hiperbólica | 1816 | 3,41% | 591 | 0,56% | 3 | 1,36% |

Tabela 7.5: Consumo de recursos do *Avaliador de limiares*

| Módulo | **LUT** (53200) | | **Flip-Flops** (106400) | | **DSP** (220) | |
|---|---|---|---|---|---|---|
| **Avaliador de limiares de som** | 96 | 0,18% | 211 | 0,20% | 0 | 0,00% |

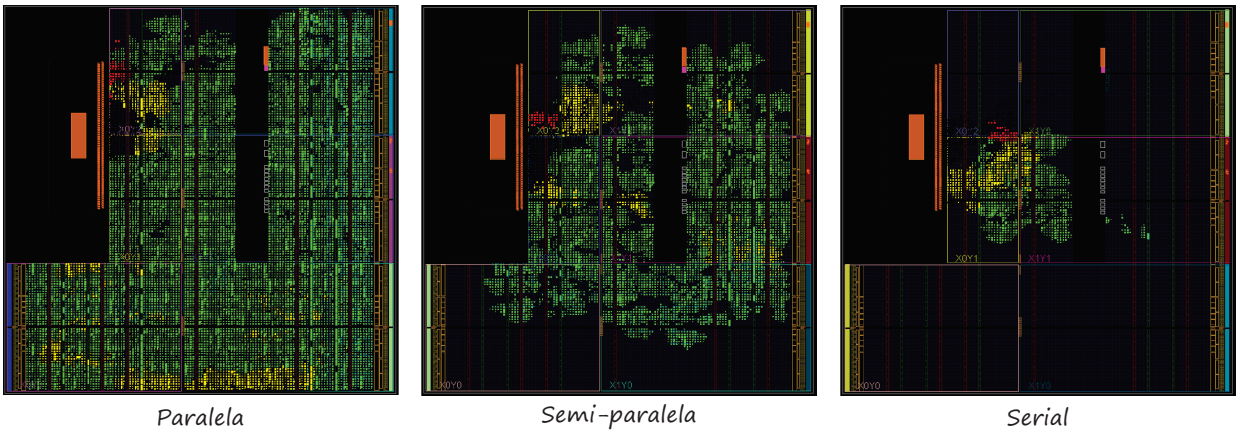



Figura 7.1: Mapeamento dos módulos que compõe o sistema de detecção de quedas. Da esquerda para a direita, *Paralela*, *Semi-paralela*, *Serial*. Em amarelo observa-se a ocupação do bloco de *pré-processamento*, em verde a ocupação da rede neural artificial, e em vermelho a ocupação do avaliador de limiares.

Após a implementação de cada circuito e a partir de uma análise de *timing* é obtida a frequência máxima de operação de cada bloco. A Tabela 7.6 apresenta os resultados para cada circuito, onde o *Avaliador de limiares*, alcança a maior frequência com 196 MHz.

Com a implementação do sistema de detecção de quedas, foi feita uma estimativa do consumo de energia de cada uma das arquiteturas projetadas. A Figura 7.2 apresenta o consumo em termos de potência estática (consumida enquanto não há atividade de circuito) e potência dinâmica (consumida pela atividade de comutação lógica). Evidencia-se que o menor consumo estático e dinâmico é realizado pela arquitetura serial com $153mW$ e $1305mW$, respetivamente. Além, disso o consumo gerado pelo processador *ARM Cortex A9* é o maior com $1256mW$, sendo constante nas três arquiteturas.

Tabela 7.6: Frequência máxima dos circuitos implementados

| Módulo | Frequência Máxima (MHz) |
|---|---|
| Sistema de deteccao de quedas (Arq. Paralela) | 99,94 |
| Sistema de deteccao de quedas (Arq. Semi-Paralela) | 92,85 |
| Sistema de deteccao de quedas (Arq. Serial) | 85,83 |
| Avaliador de limiares de som | 196.0 |

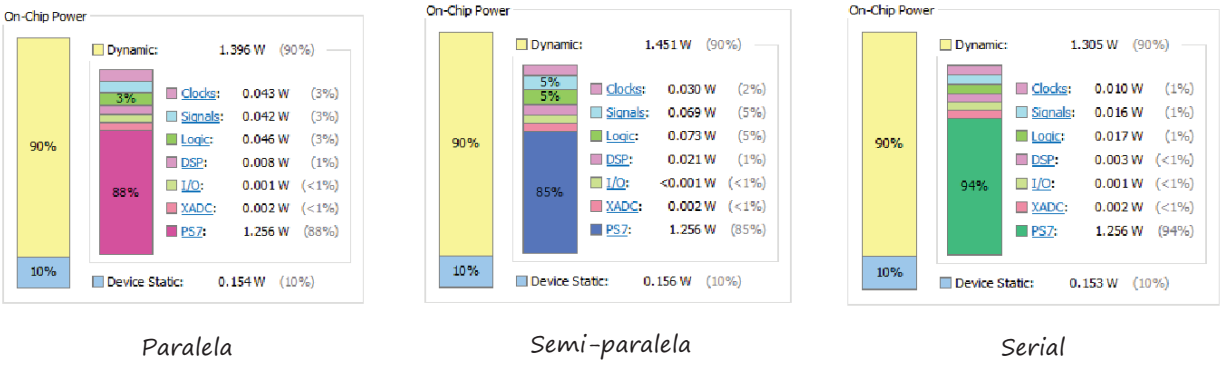



Figura 7.2: Consumo de potência das arquiteturas projetadas

Uma comparação entre o tempo de execução do sistema de detecção de quedas entre as diversas implementações de hardware e software é apresentada na Tabela 7.7. Inicialmente as três arquiteturas de hardware desenvolvidas foram configuradas para operar a uma frequência de 83.33 MHz. Os resultados obtidos mostraram que as arquiteturas propostas apresentam melhores resultados de desempenho em termos de tempo de processamento se comparado com o Arduino Nano operando a 16MHz e o processador *ARM Cortex A9* operando a 650MHz. As arquiteturas *paralela*, *semi-paralela* e *serial* são 2128, 1652, e 341, respectivamente, mais rápidas que o Arduino Nano e 136, 106, 22, respectivamente, mais rápidas que o processador *ARM Cortex A9*. Por outro lado, se tem que a arquitetura do avaliador de limiares é 983 vezes mais rápida que o Arduino Nano e 8 vezes mais rápida que o processador *ARM Cortex A9*.

Tabela 7.7: Tempos de processamento entre a solução em FPGA, Arduino Nano e *ARM Cortex A9*. *NA* indica que não foi realizada a implementação.

| Módulo | Dispositivo | | | | | |
|---|---|---|---|---|---|---|
| | Arduino | | ARM Cortex A9 | | FPGA | |
| | t [us] | Cic/Rel | t [us] | Cic/Rel | t [us] | Cic/Rel |
| Sistema de detecção de quedas (Arq. Paralela) | NA | NA | NA | NA | 2,50 | 208 |
| Sistema de detecção de quedas (Arq. Semi-Paralela) | NA | NA | NA | NA | 3,22 | 268 |
| Sistema de detecção de quedas (Arq. Serial) | 5320 | 85120 | 340 | 221000 | 15,6 | 1299 |
| Avaliador de limiares de som | 11,8 | 19 | 0,10 | 65 | 0,012 | 1 |

A partir dos resultados obtidos ao implementar as diferentes arquiteturas, decidiu-se utilizar a arquitetura serial para a avaliação do sistema de detecção de quedas, ja que ela é quem ocupa menos espaço no FPGA e, como consequência, consume menor energia, além de ter uma boa precisão nos cálculos e um tempo de processamento inferior à taxa de amostragem do sistema de aquisição.

Devido ao tamanho do kit de desenvolvimento onde encontra-se o chip FPGA, uma caixa de proteção foi fabricada em uma impressora 3D, na qual também foi colocado o dispositivo *bluetooth* e uma bateria de 9 volts. Para fixar o sensor de aceleração e o microfone ao pulso do voluntário foi utilizado o mesmo suporte apresentado no Capítulo 4. O protótipo do sistema de detecção de quedas é apresentado na Figura 7.3, que em conjunto com um celular, serão utilizados por cinco voluntários para avaliar seu desempenho, e os resultados serão apresentados na seguinte seção.

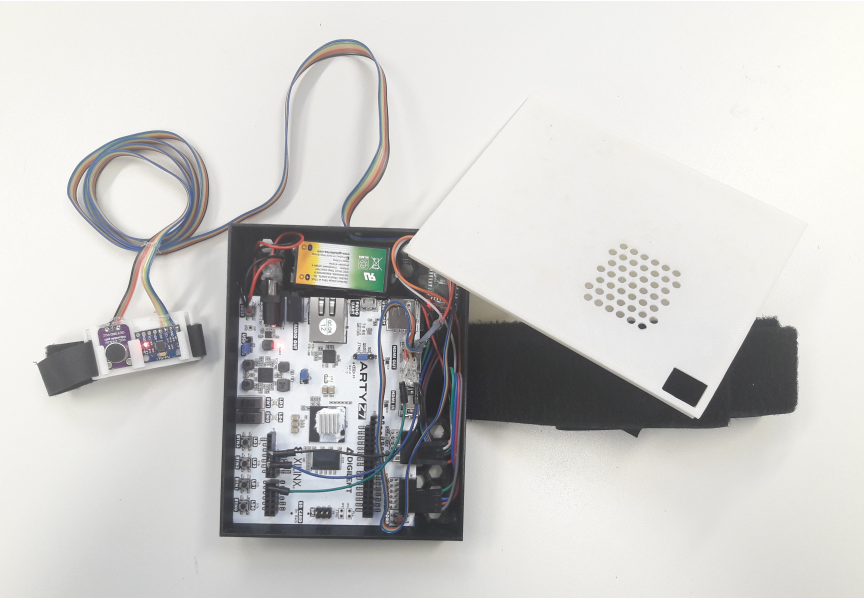


Figura 7.3: Protótipo do sistema de detecção de quedas.

Com o protótipo do sistema de detecção de quedas funcionando, foi realizada a medição do consumo de corrente de cada componente mediante um multímetro. As medições mostraram que o microfone consume $280\mu A$, o acelerômetro $6.08mA$, o Bluetooth $4.1mA$, o kit de desenvolvimento $154.5mA$ e o sistema completo $164.96mA$. Tendo em conta o resultado do consumo de potência das arquiteturas projetadas, mostrado na Figura 7.2, se estima que os circuitos implementados têm um consumo estático de $15,45mA$ e dinâmico de $8.3mA$ sem a intervenção do processador, o qual consume $130.7mA$. Assim, o sistema de detecção de quedas sem a utilização do processador teria um consumo aproximado de $34,26mA$ ($0.280mA + 6.08mA + 4.1mA + 15,45mA + 8.3mA$), que alimentado com uma bateria comercial para *smartwatch* de $380mAh$, poderia conseguir uma autonomia de até aproximadamente 11 horas.

## 7.2 Avaliação do sistema de detecção de quedas

Para a avaliação do sistema participaram quatro homens e uma mulher do grupo de voluntários (vide Capítulo 4). Cada um realizou 20 quedas, e repetiram 5 vezes os seguintes movimentos ADL: (a) bater palmas 5 vezes; (b) trotar 5 segundos; (c) pular rapidamente 5 vezes; sentar e levantar de uma cadeira; (d) subir e descer escadas; e (e) caminhar continuadamente em intervalos de 15 segundos. O protótipo foi configurado da seguinte maneira. Quando a rede neural artificial detecta uma possível queda, uma

mensagem é apresentada na tela do telefone celular. Adicionalmente, quando módulo avaliador de limiares confirma a queda detectada pela rede neural artificial, um alarme visual e táctil é gerado no celular além de realizar a mensagem de texto ao cuidador ou familiar além de armazenar as últimas 200 amostras de cada sensor. Este procedimento foi realizado no intuito de avaliar o sistema com e sem a intervenção do sinal do microfone. Os resultados para cada caso são apresentados na Tabela 7.8 e Tabela 7.9.

Tabela 7.8: Resultados da avaliação da rede implementada no Matlab.

| Experimentos | | 325 | **Valor Predito** | |
|---|---|---|---|---|
| | | | Positivos | Negativos |
| **Valor Verdadeiro** | Positivos | 100 | TP = 94 | FN = 6 |
| | Negativos | 225 | FP = 26 | TN = 199 |
| Sensibilidade = 94,0 | | | Especificidade = 88,4 | Precisão = 90,15 |

Tabela 7.9: Resultados da avaliação da rede implementada no Matlab.

| Experimentos | | 325 | **Valor Predito** | |
|---|---|---|---|---|
| | | | Positivos | Negativos |
| **Valor Verdadeiro** | Positivos | 100 | TP = 94 | FN = 6 |
| | Negativos | 225 | FP = 17 | TN = 208 |
| Sensibilidade = 94,0 | | | Especificidade = 92,4 | Precisão = 92,9 |

De acordo com os resultados apresentados, o sistema com e sem microfone consegue uma boa sensibilidade (94.0%), ou seja, ambos sistemas detectaram 94 de 100 quedas realizadas. Entretanto, a especificidade, ou capacidade de filtrar falses alarmes, é baixa sem a utilização do microfone (88.4%), mas a especificidade aumenta aproximadamente em um 4%, chegando a alcançar o 92.4%, quando se utiliza o microfone. Dos experimentos realizados se evidenciou que atividades como pular rapidamente, bater as palmas e trotar são as que geram mais falsas alarmes.

Uma comparação entre o resultado deste trabalho e os resultados obtidos em trabalhos correlatos que utilizam os dispositivos no pulso, é apresentada na Tabela 7.10. Pode-se observar que o resultado obtido neste trabalho se encontra dentro da faixa de sensibilidade e especificidade dos outros estudos, indicando que o sistema de detecção proposto consegue classificar adequadamente as quedas. Os trabalhos de Zhou et al. [68], Hsieh et al. [69] e Chen et al. [70] estão baseados em algoritmos de avaliação de limiares estabelecidos para um grupo determinado de pessoas, o que limita a detecção de quedas exclusivamente para esses usuários. Quadros et al. [71] emprega três sensores (acelerômetro, giroscópio e magnetômetro), extrai uma série de características dos sinais tais como aceleração angular, aceleração vertical, orientação, velocidade vertical, deslocamento vertical, entre outros, para serem avaliados por algoritmos de limiares e uma rede neural artificial, obtendo assim uma precisão de 99%. No entanto, esse resultado foi obtido com o grupo de dados de teste da rede, os quais pertencem às mesmas pessoas que constituem o grupo de treinamento e, portanto, não é possível evidenciar a generalização que deveria alcançar o algoritmo ao ser avaliado com pessoas diferentes às do grupo de treinamento. Se usado os dados de teste do processo de treinamento, o presente trabalho alcança uma precisão de 98.81%, uma sensibilidade de 98.24% e uma especificidade de 99.95%. Por fim, não foi realizada a comparação com os trabalhos que utilizam um FPGA para a detecção de quedas, devido a que essas abordagens usaram os sensores na cintura do usuário.

Tabela 7.10: Comparação com trabalhos correlatos usando os sensores no pulso do usuário.

| Autor | Precisão | Sensibilidade | Especificidade |
|---|---|---|---|
| Zhou et al. 2014 [68] | 93.75% | 83.33% | 95.40% |
| Hsieh et al. 2014 [69] | | 95.0% | 96.7% |
| Chen et al. 2015 [70] | | 94.44% | 100% |
| Quadros et al. 2018 [71] | 99.0% | 97.9% | 100% |
| **Este trabalho** | 92.9% | 94.0% | 92.4% |

## 7.3 Avaliação da rede neural artificial projetada com a base de dados da UTFPR

Uma avaliação adicional foi realizada usando a base de dados do grupo de pesquisa da UTFPR descrita na Seção 4.4. Neste caso os algoritmos de pré-processamento e a rede neural artificial projetada na Seção 5.3 e implementados no Matlab foram usadas para efeitos de avaliar o desempenho do sistema usando os dados coletados por Quadros et al. [71]. Desta forma, os 792 experimentos (396 ADL e 396 quedas) foram pré-processados com $n = 16$ para a filtragem e posterior discretização. Adicionalmente, sua magnitude foi limitada a $\pm 2g$. Os resultados apresentados na Tabela 7.11 mostram que a solução proposta consegue uma especificidade de 92.42%, mas uma baixa sensibilidade (81.57%) e, portanto, uma baixa precisão de 86.99%. Cabe ressaltar, que a base de dados da UTFPR foi realizada com uma frequência de amostragem de 100Hz diferente à utilizada neste trabalho (125 Hz), alterando a quantidade de dados que pode ter uma janela que contém o padrão de queda e, desta maneira, afetando o sistema de detecção baseado na rede neural artificial.

Tabela 7.11: Resultados da avaliação da rede implementada no Matlab.

| Experimentos | | 792 | **Valor Predito** | |
|---|---|---|---|---|
| | | | Positivos | Negativos |
| **Valor Verdadeiro** | Positivos | 396 | TP = 323 | FN = 73 |
| | Negativos | 396 | FP = 30 | TN = 366 |
| Sensibilidade = 94,0 | | | Especificidade = 92,4 | Precisão = 92,9 |

# Capítulo 8

# Conclusões e trabalhos futuros

Neste capítulo são apresentadas as conclusões e sugestões para trabalhos futuros que possam complementar a solução proposta para a detecção de quedas.

## 8.1 Conclusões do trabalho

Neste trabalhou foi apresentado o desenvolvimento de um algoritmo e sistema vestível para detecção de quedas constituído por um módulo de aquisição de sinais de sensores de aceleração e som colocados no pulso do usuário, algoritmos de detecção embarcados em um sistema em chip (SoC) baseado em processador ARM e um FPGA Artix7 (chip xc7z020clg400-1), assim como em um módulo de comunicação *bluetooth* e um aplicativo Android. No entendimento do autor deste trabalho, atualmente esta é a primeira implementação em FPGA de um sistema de detecção de quedas baseado na utilização de dos sensores colocados no pulso do usuário e uma rede neural artificial MLP.

Conseguiu-se identificar um padrão existente no sinal de aceleração nas quedas realizadas pelos voluntários, independentemente da constituição física das pessoas. Onde no início do movimento, a queda apresenta uma desaceleração que vai até valores menores de 0.6g e quando a pessoa impacta uma superfície, a aceleração aumenta rapidamente até valores acima de 2.8g. O padrão da queda termina quando a aceleração volta a estabelecer-se ao redor de 1g.

Um pré-processamento da magnitude do sinal de aceleração, fazendo uma filtragem e discretização no tempo, permitiu a diminuição da quantidade de dados capturados que representam o padrão de uma queda ou atividades da vida diária, diminuindo assim o número de entradas de uma rede neural artificial tipo perceptron multicamadas encarregada de classificar os movimentos de quedas e não quedas, permitindo simplificar a sua topologia e minimizar a sua complexidade computacional.

O projeto e implementação de três arquiteturas de *hardware* (*Paralela*, *Semi-paralela*, *Serial*) as quais se compararam entre si em termos de consumo de recursos *hardware*, consumo de energia, precisão nos cálculos e tempo de execução, permitiram escolher a mais adequada para o desenvolvimento de um protótipo inicial. Os resultados mostraram que a *arquitetura serial* apresenta baixo consumo de energia e recursos hardware, uma boa precisão nos cálculos e um tempo de processamento pequeno, que se encontra

dentro da faixa de amostragem dos sensores, dessa maneira esta arquitetura é a mais adequada para um sistema portável.

O treinamento da rede neural artificial para a detecção de quedas, concentrou-se nos movimentos pertencentes às atividades da vida diária, devido à existência de mais movimentos deste tipo. O resultado da implementação em *hardware*, mostrou que a rede conseguiu uma alta sensibilidade de 94%, mas com uma especificidade relativamente baixa de 88.4% devido à presença de falsos positivos detectados quando realizadas atividades da vida diária tais como bater palmas, pular rapidamente e trotar.

Uma análise inicial dos sinais obtidos pelo microfone permitiu evidenciar que não existia um único padrão durante as quedas que pudesse ser processado por uma rede neural artificial. Portanto, uma avaliação de limiares para os dados capturados pelo microfone foi implementada em *hardware*, o que permitiu, em conjunto com os dados de aceleração avaliados pela rede neural artificial, incrementar a especificidade do sistema de detecção de quedas de 88.4% para um 92.4% e incrementar a precisão do sistema de 90.15% para 92.9%.

A implementação do algoritmo proposto para a detecção de quedas também foi realizada em um Arduino Nano e no processador *ARM Cortex A9* embarcado no SoC, conseguindo demonstrar que a implementação dos algoritmos de pré-processamento e detecção de quedas usando redes neurais foi possível de implementar em um microcontrolador de baixo custo Arduino Nano, porém, devido à baixa frequência de processamento e tamanho da memória de programa não consegue implementar o sistema geral, especificamente os módulos de armazenamento dos sinais e de comunicação. Adicionalmente, o algoritmo proposto para a detecção de quedas também foi eficientemente implementado no processador embarcado *ARM Cortex A9*, porém devido à alta frequência de oscilação do processador (650 MHz), o seu consumo energético é superior à implementação em hardware no FPGA, limitando assim o seu uso em dispositivos portáveis.

Determinou-se que a detecção de quedas deve se centrar no aumento da sensibilidade, visando que todas as quedas sejam detectadas, mas isto implicaria um possível aumento nos falsos alarmes, o qual pode ser mitigado com a utilização de sensores adicionais que confirmem a queda ou mediante a intervenção do usuário.

## 8.2 Sugestões para trabalhos futuros

A seguir são apresentadas algumas sugestões para trabalhos futuros a serem desenvolvidos para melhorar a solução proposta para a detecção de quedas:

a) Estratégias para aquisição de dados de quedas reais e treinamento online.

- A aquisição de dados de quedas não simuladas deve ser realizada em trabalhos futuros. Entretanto, uma etapa necessária é a redução do tamanho do dispositivo de aquisição de dados, de forma que possa ser facilmente usado e distribuído entre pessoas que sofrem constantemente quedas, o que permitiria treinar uma rede neural artificial ou criar um novo algoritmo para a detecção de quedas.

- A exploração de algoritmos de treinamento *online* implementados em conjunto com o algoritmo de detecção e quedas, poderiam permitir a obtenção de uma rede neural artificial que se adapte automaticamente ao usuário que esteja usando o dispositivo.
- Uma base de dados na *nuvem*, integrada com o protótipo de detecção é um passo fundamental para coletar a informação de quedas reais, permitindo o treinamento em nuvem da rede neural artificial e o compartilhamento dos novos pesos de conexão da rede entre os usuários.

b) Estratégias para aumento da autonomia do sistema.

- A miniaturização do sistema geral é um aspecto importante para a fabricação de uma nova versão do protótipo do sistema de detecção de quedas. Nesse sentido, é importante destacar que o consumo de recursos da arquitetura serial dos algoritmos de pré-processamento e da rede neural usam apenas 7% de LUTs, 2% FFs e 4% de DSPs, evidenciando que a escolha de um SoC menor poderia reduzir consideravelmente o consumo energético. Uma possível solução seria a adoção do kit de desenvolvimento *Minized* o qual é baseado em um SoC Z7007S e inclui o acelerômetro, microfone e módulo *bluetooth* 4.1. Adicionalmente, as soluções fornecidas pelas placas trenz (*https://shop.trenz-electronic.de/de/*) favorecem a miniaturização dos protótipos almejados neste trabalho.

c) Sugestões adicionais.

- Uma análise espectral dos sinais capturados pelo microfone permitiria estabelecer limiares em frequência para distinguir entre os diferentes sons gerados por uma queda e os gerados por atividades da vida diária existentes.
- Para fornecer uma melhor portabilidade do protótipo proposto, se sugere a implementação de outros módulos de comunicação como o GSM, o qual permitiria a transmissão de um alarme sem a intervenção de um celular.
- Uma análise adicional do sinal de aceleração permitiria também a detecção de padrões anormais no caminhar do usuário, os quais podem ser causados por alguma doença.

# REFERÊNCIAS BIBLIOGRÁFICAS


[1] World Health Organization. (2018) Caidas. Disponível em http://www.who.int/mediacentre/factsheets/fs344/es/. Acesso em: 25 maio 2018.

[2] S. C. Coelho Fabrício, R. A. Partezani Rodrigues, and M. Lobo da Costa, "Causas e conseqüências de quedas de idosos atendidos em hospital público," *Revista de Saúde Publica*, vol. 38, no. 1, pp. 93–99, 2004.

[3] L. S. Vieira, A. P. Gomes, I. O. Bierhals, S. Farías-Antúnez, C. G. Ribeiro, V. I. Miranda, B. H. Lutz, T. G. Barbosa-Silva, N. P. Lima, A. D. Bertoldi *et al.*, "Quedas em idosos no sul do brasil: prevalência e determinantes," *Rev. Saúde Pública*, vol. 52, 2018.

[4] A. P. Ribeiro, E. R. d. Souza, S. Atie, A. C. d. Souza, and A. O. Schilithz, "A influência das quedas na qualidade de vida de idosos," *Ciência & Saúde Coletiva*, vol. 13, pp. 1265–1273, 2008.

[5] K. Onaga Jahana *et al.*, "Quedas em idosos: principais causas e conseqüências," *Saúde coletiva*, vol. 4, no. 17, 2007.

[6] H. G. Espínola, *Caidas en el adulto mayor*, Escuela de Medicina, Pontificia Universidad Católica de Chile Std.

[7] S. R. M. Pereira, S. Buksman, M. Perracini, L. Py, K. M. L. Barreto, V. M. M. Leite *et al.*, "Quedas em idosos," *Sociedade Brasileira de Geriatria e Gerontologia*, pp. 1–8, 2001.

[8] S. Buksman, A. Vilela, S. Pereira, V. Lino, and V. Santos, "Quedas em idosos: Prevenção," *Brasil: Sociedade Brasileira de Geriatria e Gerontologia*, 2008.

[9] World Health Organization. Ageing and Life Course Unit, "Who global report on falls prevention in older age," 2007.

[10] E. M. De Carvalho, T. C. d. O. Delani, and A. A. Ferreira, "Atenção à saúde do idoso no brasil relacionada ao trauma," *Revista Uningá review*, vol. 20, no. 3, 2018.

[11] V. S. Stel, J. H. Smit, S. M. Pluijm, and P. Lips, "Consequences of falling in older men and women and risk factors for health service use and functional decline," *Age and ageing*, vol. 33, no. 1, pp. 58–65, 2004.

[12] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, P. Corsini, and A. Vecchio, "A smartphone-based fall detection system," *Pervasive and Mobile Computing*, vol. 8, no. 6, pp. 883–899, dec 2012.

[13] P. Kannus, M. Palvanen, S. Niemi, and J. Parkkari, "Alarming rise in the number and incidence of fall-induced cervical spine injuries among older adults," *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences*, vol. 62, no. 2, pp. 180–183, 2007.

[14] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, "Evaluation of accelerometer-based fall detection algorithms on real-world falls," *PloS one*, vol. 7, no. 5, p. e37062, 2012.

[15] T. M. Gill, M. M. Desai, E. A. Gahbauer, T. R. Holford, and C. S. Williams, "Restricted activity among community-living older persons: incidence, precipitants, and health care utilization," *Annals of Internal Medicine*, vol. 135, no. 5, pp. 313–321, 2001.

[16] M. Kangas, "Development of accelerometry-based fall detection," University of Oulu, Tech. Rep., 2011.

[17] M. Kangas, A. Konttila, P. Lindgren, I. Winblad, and T. Jämsä, "Comparison of low-complexity fall detection algorithms for body attached accelerometers," *Gait & posture*, vol. 28, no. 2, pp. 285–291, 2008.

[18] S. Haykin, *Redes neurais: princípios e prática*. Bookman, São Paulo, Brasil, 2007.

[19] ——, *Neural networks and learning machines*. Pearson, NY, USA, 2009, vol. 3.

[20] E. Caicedo and J. Lopez, *Redes neuronales artificiales*. Programa editorial universidad del Valle, 2010, vol. 1.

[21] S. D. Brown and Z. G. Vranesic, *Fundamentos de lógica digital con diseño VHDL*. McGraw-Hill, D.F., Mexico, 2007.

[22] N. Einspruch and J. L. Hilbert, *Application specific integrated circuit (ASIC) technology*. Academic Press, West Virginia University, Morgantown, WV 26506, USA, 2012.

[23] R. E. Haskell and D. M. Hanna, *Introduction to Digital Design: Using Digilent FPGA Boards, Block Diagram VHDL Examples*. LBE Books, Michigan, USA, 2009.

[24] V. A. Pedroni, *Circuit design with VHDL*. MIT press, London, England, 2004.

[25] U. Farooq, Z. Marrakchi, and H. Mehrez, "Fpga architectures: An overview," in *Tree-based Heterogeneous FPGA Architectures*. Springer, 2012, pp. 7–48.

[26] Xilinx. Arty z7 reference manual. Disponível em https://reference.digilentinc.com/reference/programmable-logic/arty-z7/reference-manual. Acesso em: 24 jul. 2018.

[27] C. Dinh and M. Struck, "A new real-time fall detection approach using fuzzy logic and a neural network," *International Workshop on Wearable Micro and Nano Technologies for Personalized Health, Oslo, Norway*, pp. 57–60, 2009.

[28] M. Yuwono, S. W. Su, and B. Moulton, "Fall detection using a Gaussian distribution of clustered knowledge, augmented radial basis neural-network, and multilayer perceptron," in *6th International Conference on Broadband Communications and Biomedical Applications, Melbourne, VIC, Australia*, 2011, pp. 145–150.

[29] M. Vallejo, C. V. Isaza, and J. D. Lopez, "Artificial Neural Networks as an alternative to traditional fall detection methods." in *IEEE International Conference in Medicine and Biology Society, Osaka, Japan*, 2013, pp. 1648–51.

[30] B. T. Nukala, N. Shibuya, A. Rodriguez, J. Tsay, J. Lopez, T. Nguyen, S. Zupancic, and D. Y.-C. Lie, "An Efficient and Robust Fall Detection System Using Wireless Gait Analysis Sensor with Artificial Neural Network (ANN) and Support Vector Machine (SVM) Algorithms," *Open Journal of Applied Biosensor*, pp. 29–39, 2014.

[31] C. N. Doukas and I. Maglogiannis, "Emergency fall incidents detection in assisted living environments utilizing motion, sound, and visual perceptual components," *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 2, pp. 277–289, 2011.

[32] K. Sehairi, F. Chouireb, and J. Meunier, "Elderly fall detection system based on multiple shape features and motion analysis," in *Intelligent Systems and Computer Vision (ISCV), Fez, Morocco*. IEEE, 2018, pp. 1–8.

[33] A. Shojaei-Hashemi, P. Nasiopoulos, J. J. Little, and M. T. Pourazad, "Video-based human fall detection in smart homes using deep learning," in *Circuits and Systems (ISCAS), Florence, Italy*. IEEE, 2018, pp. 1–5.

[34] S. Kim, M. Ko, K. Lee, M. Kim, and K. Kim, "3d fall detection for single camera surveillance systems on the street," in *Sensors Applications Symposium (SAS), Seoul, South Korea*. IEEE, 2018, pp. 1–6.

[35] X. Kong, L. Meng, and H. Tomiyama, "Fall detection for elderly persons using a depth camera," in *Advanced Mechatronic Systems (ICAMechS), Xiamen, China*. IEEE, 2017, pp. 269–273.

[36] M. Popescu, Y. Li, M. Skubic, and M. Rantz, "An acoustic fall detector system that uses sound height information to reduce the false alarm rate," in *Engineering in Medicine and Biology Society (EMBS), Vancouver, BC, Canada*. IEEE, 2008, pp. 4628–4631.

[37] Y. Li, Z. Zeng, M. Popescu, and K. Ho, "Acoustic fall detection using a circular microphone array," in *Engineering in Medicine and Biology Society (EMBC), Buenos Aires, Argentina*. IEEE, 2010, pp. 2242–2245.

[38] D. Litvak, I. Gannot, and Y. Zigel, "Detection of falls at home using floor vibrations and sound," in *Electrical and Electronics Engineers in Israel, Eilat, Israel*. IEEE, 2008, pp. 514–518.

[39] X. Zhuang, J. Huang, G. Potamianos, and M. Hasegawa-Johnson, "Acoustic fall detection using gaussian mixture models and gmm supervectors," in *Acoustics, Speech and Signal Processing (ICASSP), Taipei, Taiwan*. IEEE, 2009, pp. 69–72.

[40] Y. Li, K. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 5, pp. 1291–1301, 2012.

[41] M. S. Khan, M. Yu, P. Feng, L. Wang, and J. Chambers, "An unsupervised acoustic fall detection system using source separation for sound interference suppression," *Journal Signal processing*, vol. 110, pp. 199–210, 2015.

[42] M. Cheffena, "Fall detection using smartphone audio features," *IEEE journal of biomedical and health informatics*, vol. 20, no. 4, pp. 1073–1080, 2016.

[43] W. Ye and B. Xiang-yu, "Research of fall detection and alarm applications for the elderly," in *Mechatronic Sciences, Electric Engineering and Computer (MEC), Shengyang, China*. IEEE, 2013, pp. 615–619.

[44] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy, "Wearable sensors for reliable fall detection," in *Engineering in Medicine and Biology Society (EMBS), Shanghai, China*. IEEE, 2006, pp. 3551–3554.

[45] A. Bourke, J. Obrien, and G. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," *Gait & posture, Elsevier*, vol. 26, no. 2, pp. 194–199, 2007.

[46] T. Zhang, J. Wang, P. Liu, and J. Hou, "Fall detection by embedding an accelerometer in cellphone and using kfd algorithm," *International Journal of Computer Science and Network Security*, vol. 6, no. 10, pp. 277–284, 2006.

[47] A. K. Bourke and G. M. Lyons, "A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor," *Medical engineering and physics*, vol. 30, no. 1, pp. 84–90, 2008.

[48] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *Wearable and Implantable Body Sensor Networks, Berkeley, CA, USA*. IEEE, 2009, pp. 138–143.

[49] S. Abdelhedi, M. Baklouti, R. Bourguiba, and J. Mouine, "Design and implementation of a fall detection system on a zynq board," in *Computer Systems and Applications (AICCSA), Agadir, Morocco*. IEEE, 2016, pp. 1–7.

[50] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan, "Mobile phone-based pervasive fall detection," *Personal and ubiquitous computing*, vol. 14, no. 7, pp. 633–643, 2010.

[51] P. Mostarac, R. Malarić, M. Jurčević, H. Hegeduš, A. Lay-Ekuakille, and P. Vergallo, "System for monitoring and fall detection of patients using mobile 3-axis accelerometers sensors," in *Medical Measurements and Applications Proceedings (MeMeA), Bari, Italy*. IEEE, 2011, pp. 456–459.

[52] J. Jacob, T. Nguyen, D. Y. Lie, S. Zupancic, J. Bishara, A. Dentino, and R. E. Banister, "A fall detection study on the sensors placement location and a rule-based multi-thresholds algorithm using both accelerometer and gyroscopes," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference, Taipei, Taiwan*. IEEE, 2011, pp. 666–671.

[53] Y. Cao, Y. Yang, and W. Liu, "E-falld: A fall detection system using android-based smartphone," in *Fuzzy systems and knowledge discovery (FSKD), 2012 9th international conference, Sichuan, China*. IEEE, 2012, pp. 1509–1513.

[54] W. Wibisono, D. N. Arifin, B. A. Pratomo, T. Ahmad, and R. M. Ijtihadie, "Falls detection and notification system using tri-axial accelerometer and gyroscope sensors of a smartphone," in *Technologies and Applications of Artificial Intelligence (TAAI), Taipei, Taiwan*. IEEE, 2013, pp. 382–385.

[55] L. Tong, Q. Song, Y. Ge, and M. Liu, "Hmm-based human fall detection and prediction method using tri-axial accelerometer," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1849–1856, 2013.

[56] P. Salgado and P. Afonso, "Fall body detection algorithm based on tri-accelerometer sensors," in *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium, Budapest, Hungary*. IEEE, 2013, pp. 355–358.

[57] L. Ren, W. Shi, Z. Yu, and J. Cao, "Alarm: A novel fall detection algorithm based on personalized threshold," in *E-health Networking, Application & Services (HealthCom), 2015 17th International Conference, Boston, MA, USA*. IEEE, 2015, pp. 410–415.

[58] T. Shi, X. Sun, Z. Xia, L. Chen, and J. Liu, "Fall detection algorithm based on triaxial accelerometer and magnetometer." *Engineering Letters*, vol. 24, no. 2, 2016.

[59] M. I. Nari, S. S. Suprapto, I. H. Kusumah, and W. Adiprawita, "A simple design of wearable device for fall detection with accelerometer and gyroscope," in *Electronics and Smart Devices (ISESD), International Symposium, Bandung, Indonesia*. IEEE, 2016, pp. 88–91.

[60] J. Santiago, E. Cotto, L. G. Jaimes, and I. Vergara-Laurens, "Fall detection system for the elderly," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual, Las Vegas, NV, USA*. IEEE, 2017, pp. 1–4.

[61] C. Doukas and I. Maglogiannis, "Advanced patient or elder fall detection based on movement and sound data," in *Pervasive Computing Technologies for Healthcare, Tampere, Finland*. IEEE, 2008, pp. 103–107.

[62] A. Sengto and T. Leauhatong, "Human falling detection algorithm using back propagation neural network," in *Proceedings of Biomedical Engineering International Conference (BMEiCON), Ubon Ratchathani, Thailand*, 2012, pp. 1–5.

[63] H. Kerdegari, K. Samsudin, A. Rahman Ramli, and S. Mokaram, "Development Of Wearable Human Fall Detection System Using Multilayer Perceptron Neural Network," *International Journal of Computational Intelligence Systems*, pp. 127–136, feb 2013.

[64] Z. Qingbin, T. Guohui, D. Nana, and Z. Yanru, "A fall detection study based on neural network algorithm using AHRS," in *Information and Automation (ICIA), 2013 IEEE International Conference, Yinchuan, China*, 2013, pp. 773–779.

[65] M. Vidigal, M. Lima, and A. d. A. Neto, "Elder Falls Detection Based on Artificial Neural Networks," in *2015 Mexican International Conference on Artificial Intelligence (MICAI), Cuernavaca, Mexico*, 2015, pp. 226–230.

[66] N. Nuttaitanakul and T. Leauhatong, "A novel algorithm for detection human falling from accelerometer signal using wavelet transform and neural network," in *Information Technology and Electrical Engineering (ICITEE), 2015 7th International Conference, Chiang Mai, Thailand*, 2015, pp. 215–220.

[67] N. Yodpijit, T. Sittiwanchai, and M. Jongprasithporn, "The development of Artificial Neural Networks (ANN) for falls detection," in *2017 3rd International Conference on Control, Automation and Robotics, ICCAR 2017, Nagoya, Japan*, 2017, pp. 547–550.

[68] C.-C. Zhou, C.-L. Tu, Y. Gao, F.-X. Wang, H.-W. Gong, P. Lian, C. He, and X.-S. Ye, "A low-power, wireless, wrist-worn device for long time heart rate monitoring and fall detection," in *Orange Technologies (ICOT), Xian, China*. IEEE, 2014, pp. 33–36.

[69] S.-L. Hsieh, C.-C. Chen, S.-H. Wu, and T.-W. Yue, "A wrist-worn fall detection system using accelerometers and gyroscopes," in *Networking, Sensing and Control (ICNSC), Miami, FL, USA*. IEEE, 2014, pp. 518–523.

[70] S. Zhou, J. Chen, X. Wang, L. Zhou, B. Zhen, and J. Cui, "Inclination gradient-based fall detection algorithm for wrist-worn device," in *Consumer Electronics-Taiwan (ICCE-TW), Taipei, Taiwan*. IEEE, 2015, pp. 148–149.

[71] T. de Quadros, A. E. Lazzaretti, and F. K. Schneider, "A movement decomposition and machine learning-based fall detection system using wrist wearable device," *IEEE Sensors Journal*, 2018.

[72] W. Saadeh, M. A. B. Altaf, and M. S. B. Altaf, "A high accuracy and low latency patient-specific wearable fall detection system," in *Biomedical & Health Informatics (BHI), 2017 IEEE EMBS International Conference, Orlando, FL, USA*. IEEE, 2017, pp. 441–444.

[73] M. Neggazi, A. Amira, and L. Hamami, "A wireless reconfigurable system for falls detection," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference, Montreal, QC, Canada*. IEEE, 2012, pp. 77–82.

[74] InvenSense Inc. Mpu-6000 and mpu-6050 product specification revision 3.4. Disponível em https://www.invensense.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf. Acesso em: 25 maio 2018.

[75] Maxim Integrated Products. (2018) Low-cost, micropower, sc70/sot23-8, microphone preamplifiers with complete shutdown. Disponível em https://cdn-shop.adafruit.com/datasheets/MAX4465-MAX4469.pdf. Acesso em: 25 maio 2018.

[76] Arduino. (2013) Arduino nano. Disponível em https://store.arduino.cc/usa/arduino-nano. Acesso em: 25 maio 2018.

[77] A. T. Özdemir, "An analysis on sensor locations of the human body for wearable fall detection devices: Principles and practice," *Sensors, Basel, Switzerland*, vol. 16, no. 8, p. 1161, 2016.

[78] Analog Devices Inc. (2015) Adxl345 datasheet. Disponível em http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf. Acesso em: 24 jul. 2018.

[79] ST Microelectronics. (2011) L3g4200d datasheet. Disponível em https://www.elecrow.com/download/L3G4200_AN3393.pdf. Acesso em: 24 jul. 2018.

[80] Honeywell. (2013) Hmc5883l datasheet. Disponível em http://www.interaxllc.com/GettingStarted/hmc5883.pdf. Acesso em: 24 jul. 2018.

[81] K. L. Priddy and P. E. Keller, *Artificial neural networks: an introduction*. SPIE press, Bellingham, Washington, USA, 2005, vol. 68.

[82] R. F. López and J. M. F. Fernández, *Las redes neuronales artificiales*. Netbiblo, La Coruña, España, 2008.

[83] D. M. Muñoz, D. F. Sanchez, C. H. Llanos, and M. Ayala-Rincón, "Fpga based floating-point library for cordic algorithms," in *Programmable Logic Conference (SPL), 2010 VI Southern, Ipojuca, Brazil*. IEEE, 2010, pp. 55–60.

[84] ——, "Tradeoff of fpga design of a floating-point library for arithmetic operators," *Journal of Integrated Circuits and Systems*, vol. 5, no. 1, pp. 42–52, 2010.

# APÊNDICES

# APÊNDICE A.

## Fall Detection using Accelerometer on the User's Wrist and Artificial Neural Networks

Javier Alexis Urresty Sanchez[1] and Daniel M. Muñoz[1,2]

[1] Department of Mechanical Engineering, University of Brasilia, Brasilia, DF, Brazil
[2] Electronics Engineering Program, University of Brasilia, Gama, DF, Brazil,
[1]javierur10@gmail.com

**Abstract.** Falls can occur during daily activities, being one of the main sources of morbidity and mortality in the elderly population. Real-time detection of falls and their communication to an emergency center can enables rapid medical care, increasing the sense of security for the elderly and reducing some of the negative consequences of falls. This work presents an artificial neural network based algorithm for fall detection using a three axis accelerometer wrist wearable device. Samples were filtered and discretized in time domain allowing a multilayer perceptron neural network to classify the events. The training process was performed using 792 signals acquired by 22 volunteers. Validation results point-out that the proposed solution obtains a sensitivity of 98.10%, a specificity of 98.10% and a accuracy of 98.10%, concluding that the neural network generalization capacity allows the fall detection without the necessity of complex feature extraction algorithms.

**Keywords:** Fall detection, Artificial neural network, Embedded systems.

### 1 Introduction

According to the World Health Organization, 646.000 people worldwide die every year due to falls. Falls are caused by a lack of balance or inability to regain balance, causing the person to hit the ground or other firm surface. Falls injuries can be fatal, being the second cause of death from accidental or unintentional injuries worldwide. The risk and frequency of falls increases with age and fragility, being the population over 65 years the one which most suffers from morbidity and mortality due to falls [22].

The main hospital admissions related to falls are surface injuries, hip fracture, spinal cord injuries, cranioencephalic trauma and upper and lower limb injuries [21], [4]. Even in the absence of injury, falls can produce long-term psychological consequences, such as depression and loss of confidence, that can lead to restrictions in daily and social activities, and subsequently, will decrease health and increase the risk of future falls [18], [1].

The increase in the elderly population during the last decades is reflected in the rapid increment in the number of injuries caused by falls, becoming a

relevant health problem [22], [10]. In this context, real-time fall detection and rapid communication to an emergency center is a technological challenge, and developments on automatic fall detection systems has been increased in the last 20 years [2].

In the scientific literature there are several proposals for developing monitoring systems which allow the identification of different falls events. Some of these systems also ensure efficient communication to medical care, minimizing the fatal consequences. Most of the current proposals make use of portable sensors such as accelerometers, gyroscopes and magnetometers, enabling data acquisition of the monitored person's inertial information [9]. In addition, most of the previous works place the inertial systems in the waist or chest. However, this fact creates discomfort for users, making it difficult the acceptance of this technology. In this context, a fall detection system that facilitates usability must be studied from the technological feasibility perspective.

The main contribution of this work is the development of a fall detection system based on the acceleration measurements using a wrist wearable device. A processing algorithm based on an artificial multilayer perceptron (MLP) neural network was developed to perform the classification of movements, including the detection of falls. Numerical comparisons were done to select the best topology of the neural network and validation results point-out that daily living activities can be distinguished from falls with a sensitivity of 98.10%, specificity of 98.10% and accuracy of 98.10%. Experimental results demonstrate that it is possible to achieve high accuracy fall detection using only one sensor (accelerometer), directly processing the signals using a MLP neural network, avoiding complex pre-processing algorithms for features extraction.

The remainder of this paper is organized as follows: Section 2 presents the theoretical basis for understanding the proposed solution. Section 3 shows the related works. Section 4 describes the system architecture for data acquisition. Section 5 presents the proposed artificial neural network for fall detection and, before concluding, Section 6 presents several validation results.

## 2 Theoretical Background

### 2.1 Artificial neural networks

Artificial neural networks are used in a variety of engineering fields, such as signal processing, control systems, time series analysis and pattern recognition. According to Haykin, a neural network is a parallel distributed processor composed of simple and interconnected processing units and has the natural ability to store knowledge and make it available for use. It resembles the brain in two main aspects: (a) the knowledge from the environment is acquired by the network through a learning process and; (b) the connecting forces between neurons, known as synaptic weights, are used to store knowledge [7].

### 2.2 Neuron

The neuron is the fundamental processing unit of neural networks. The *perceptron* is an artificial neuron which consists of a set of inputs (representing synapses from previous neurons) which are weighted and then accumulated. The output of neuron is modified by the bias which has the effect of applying an offset, depending on whether the bias polarization is positive or negative. The new result is called *induced local field*, which is applied to an activation function that restricts the amplitude of the output signal [6]. In mathematical terms, a perceptron neuron can be described with the following equations

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{1}$$

$$v_k = u_k + b_k \tag{2}$$

$$y_k = \varphi(v_k) \tag{3}$$

where in Equation 1, $w_{k1}, w_{k2}, ..., w_{km}$ are the synaptic weights of neuron $k$. $x_1, x_2, ..., x_i$ are the input signals and $u_k$ is the result of the linear combination of the weighted inputs. In Equation 2 the bias $b$ is added to $u_k$ thus obtaining the induced local field $v_k$. In Equation 3, $v_k$ is evaluated in the activation function $\varphi$ resulting in the output signal of the neuron $y_k$.

The activation function is chosen during the design phase according to the type of output [3]. In this work an hyperbolic tangent function was used (see Equation 4), limiting the output values of the neuron and consequently the output of the neural network between -1 to 1 [7].

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4}$$

### 2.3 Multilayer Perceptron Neural Network

This network, also called multilayer feedforward network, have a set of source nodes that constitute the input layer, one or more hidden layers, and an output layer [6]. Multilayer perceptron networks are used to solve many complex problems, such as those that are not linearly separable [7]. Supervised training algorithms are generally used for this type of network, for instance, the backpropagation algorithm which is based on the error correction learning rule, also considered as a generalization of the least mean squares (LMS) algorithm used in networks of a single linear neuron [7].

The Fig. 1 shows a multilayer perceptron neural network, with $m$ neurons in the input layer, a hidden layer with $n$ neurons and three neurons in the output layer.

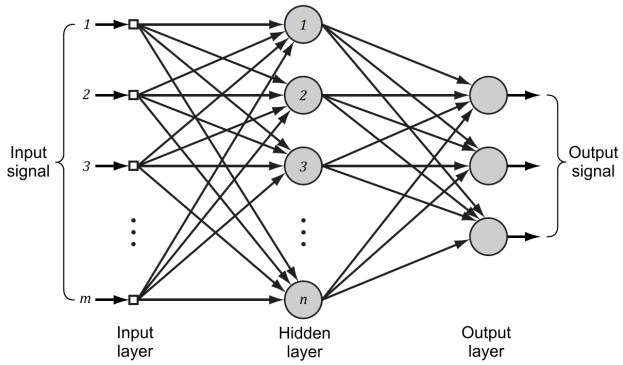



**Fig. 1.** Structure of a multilayer perceptron neural network [7].

### 2.4 Backpropagation algorithm

The learning process by backpropagation of error is divide in two steps: one step forward called propagation and another backwards, the backpropagation. During propagation, a vector belonging to the environment is inserted into the network, it propagates through the network producing an output response. Throughout this step, the network's synaptic weights are fixed [6]. During backpropagation, the synaptic weights of the network are adjusted according to the error correction rule, where the error signal is obtained from the subtraction of the actual network response and the expected response. This error signal propagates backwards, adjusting the synaptic weights so that the network response is close to the desired response. After reaching a number of iterations or a previously established error value it is possible to obtain optimized synaptic weights [7].

## 3 Related Works

Fall detection systems employing wearable devices are most preferred because users can be monitored in indoor and outdoor environments [9]. In general, most of the developed systems make use of triaxial accelerometers and peak detection methods for data processing. This section presents an analysis of the previous works that make use artificial neural networks for fall detection. [24] proposes a fall detection algorithm, with a single triaxial accelerometer placed on the user's waist and using two different classifiers: A multilayer perceptron network and an increased radial base neural network (RBF) achieving a promising result. [20] presents an implementation and performance comparison of MLP, RBF and Kohonen neural networks for the detection of falls, using the data provided by the triaxial accelerometer present in smartphones. [17] suggests an algorithm with two processes: firstly, a simple threshold analysis, which is used to distinguish slow movements and other activities from a person's daily life. Secondly, if the

value of the signals are greater than the thresholds, a back-propagation neural is used to classify eight different activities, including falls. [19] presents a fall detection system composed of an MLP neural network, which aims to improve detection accuracy with respect to traditional threshold-based methods. The network was implemented in a device placed on the user's waist and used the integral of the acceleration of each axis.

[11] proposes a fall detection system using acceleration, angular velocity, speed and position measurements as well as different time domain characteristics, such as minimum, maximum, mean, interval, variance and standard deviation. These characteristics constitutes the inputs of a multi-layer perceptron neural network which allow the classification of daily activities and falls. [23] uses an accelerometer and a gyroscope placed on the user's waist to detect the orientation and the movement of the body. They implemented a threshold-based algorithm in conjunction with a multilayer perceptron neural network, in order to decrease the amount of false positives and improve the accuracy of the fall detection system. [15] proposes a movement decomposition feature extraction and machine learning method for a fall detection system using a wrist-wearable device with accelerometer, gyroscope and magnetometer.

[5] describes a fall detection system where the acceleration data is transformed from Cartesian coordinates to spherical coordinates. The system is based on Fuzzy logic and a MLP neural network. The achieved results show that it is single accelerometer measurements and the methods based on the neuro-fuzzy systems are feasible solutions in comparison with standard pattern recognition methods. [14] proposes a sensor fusion between accelerometer, gyroscope and magnetometer, by means of an extended Kalman filter in order to determine the attitudes of the pitch and roll angles that, together with the mean square of the acceleration, conform the entries of an MLP neural network. [12] implemented an MLP neural network and a support vector machine (SVM) based on 6 features extracted from angular velocity measurements and accelerations on the three axes, achieving a fall detection system that is placed on the back and waist of the user. [13] proposes a Wavelet transform combined with a MLP neural network algorithm to detect falls from the acceleration signals of a sensor placed on the user's waist.

The proposal presented in this work makes use of a triaxial accelerometer placed in the user's wrist, improving usability and the sensation of comfort; however, representing a challenge to treat movements that generate false positives. In addition, low computational cost preprocessing methods and MLP neural networks were used allowing for filtering such movements.

## 4 Data Acquisition System

### 4.1 Data Acquisition Architecture

The data acquisition system was developed using a microcontroller, a three axis accelerometer and a data storage device. All the subsystems are powered up by a 9 volts battery as can be seen in Fig. 2.

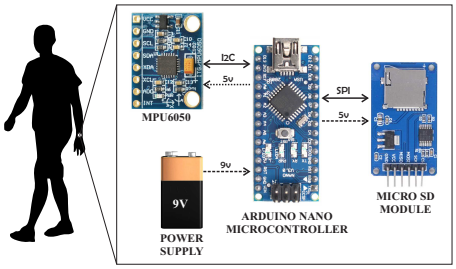



**Fig. 2.** General system architecture.

A six-axis movement detection sensor (MPU650 from InvenSense) was used. It is a low cost and low power consumption device which includes a 3-axis accelerometer and a 3-axis gyroscope and operates at 3.3 or 5 volts. An *Inter-Integrated Circuit* (I2C) communication was used to connect the sensor to a microcontroller. It is important to highlight that in the proposed system only the acceleration is used for fall detection.

An Arduino Nano microcontroller was used to control the data acquisition system. It is based on an ATMega328P processor, which uses a 32 KB flash memory and operates at 16 MHz clock frequency. The data captured by the accelerometer sensor are stored in a micro SD memory and then analyzed and classified by the MLP neural network. A *Serial Peripheral Interface* (SPI) communication was used to send the data from the microcontroller to the micro SD memory.

### 4.2 Data Acquisition Procedure

In this work we have used the data set reported in [15]. This data base is composed of acceleration, gyroscope and magnetometer signals acquired at 100 Hz by twenty-two volunteers which performed for six falls and six non-falls activities. A total of 792 signals are available including forward fall, backward fall, sideways fall and fall after rotating the waist. The non-falls includes movements clapping hands, walking, opening and closing doors, moving an object, tying shoes and sitting on a chair. The volunteers have in average, an age of 26 years, height of 1.68 m and weight of 67.82 Kg. It is important to highlight that the data acquisition system used in [15] is similar to the one proposed in this work. A first analysis of the experimental data determined that, taking into account the data acquisition rate, all the falls can be represented using 138 samples. Therefore, we adapted the data set to the maximum range value supported by our data acquisition architecture and manually extracted windows with 138 samples for each movement signal.

All the falls movements were simulated using a proper soft mattress. If compared with real falls, this controlled environment minimize the intensity of the

signal and generate oscillating movements after striking the mattress. However, the shape of the signal into the window of 138 samples is well preserved.

In order to analyze the action of the gravity on the user's wrist, the magnitude of the acceleration vector was computed. Fig. 3 depicts the magnitude of the acceleration for walking, clapping and falling movements, respectively. In the latter a deceleration from 1 to 0.5 g followed by a rapid acceleration from 0.5 to 3.5g can be observed. This pattern was observed in almost all the 396 fall samples. Vallejo et al. [19] point-outs that the threshold-based analysis algorithms present several faults when the accelerometer is placed in the wrist, producing constant changes in the acceleration signal. However, our experimental observations allow us to verify that all the falls have repetitive patterns that can be interpreted and classified by ANNs.

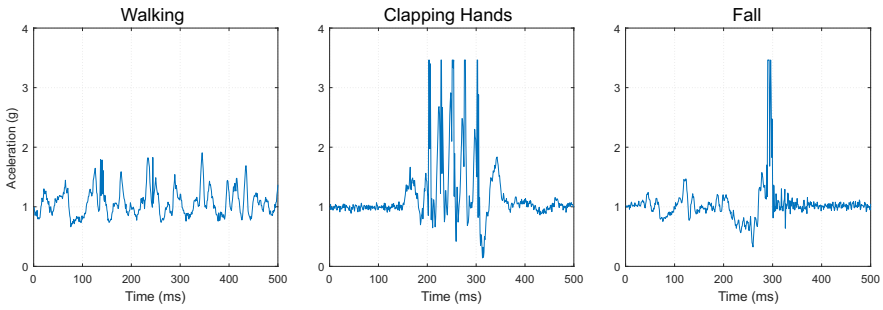



**Fig. 3.** Magnitude of acceleration for walking, clapping hands and falling movements.

## 5 Fall Detection Using MLP Neural Networks

The data set was divided in training and validation data. The training data are composed of 288 falls and 288 samples with daily living movements. The validation data set is composed of 108 falls and 108 daily living activities collected from users that do not belong to the training and testing data set.

In order to minimize the number of inputs, a mean filter was applied to the signals using windows composed of 32, 23, 16 and 13 samples, obtaining signals with 4, 6, 8 and 10 levels, respectively. Afterwards, the signal was discretized in time using one sample per level, minimizing the number of inputs to 4, 6, 8 and 10 inputs. Fig. 4 represents the above mentioned process for a 16 samples window.

The neural network process was performed in the *NNtool* from Matlab using the *Levenberg-Marquardt* backpropagation algorithm. Different MLP neural networks with 4, 6, 8 and 10 inputs, each with 2, 4, 6, 8, 15 and 20 neurons in the

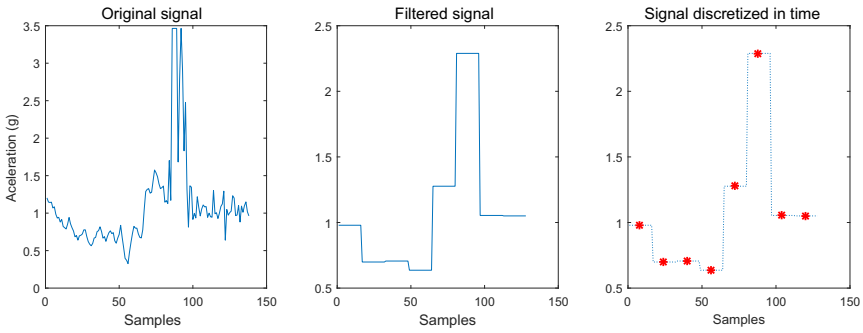



**Fig. 4.** Data preprocessing in the case of 16 samples window.

hidden layer, were trained and tested using the method known as k-fold cross-validation. This method divides the training data set into $k$ subsets of size $n/k$, where $n = 4165$. The network is then trained and validated k-fold, always leaving a subset for validation and k-1 for training [8]. In this work the most common validation with $k = 10$ was implemented, which allowed numerical comparisons to determine the best network topology [16]. Table 1 shows the performance results for each topology using the 416 samples of the testing data set.

**Table 1.** Percentage of hits for different network topologies.

| **Inputs** | **Neurons** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **2** | **4** | **6** | **8** | **10** | **15** | **20** |
| **4** | 97.25 | 97.89 | 98.38 | 98.42 | 98.49 | 98.49 | 98.52 |
| **6** | 98.40 | 96.45 | 98.52 | 98.08 | 98.54 | 98.11 | 98.15 |
| **8** | 98.26 | 98.27 | 97.93 | 98.55 | 98.41 | 98.55 | 98.04 |
| **10** | 97.26 | 98.52 | 97.92 | 98.24 | 98.33 | 98.35 | 98.57 |

According to the performance results, the best classification was obtained by the topology composed of 10 inputs, 20 neurons in the hidden layer and one output neuron. However, the topology composed of 8 inputs, 8 neurons in the hidden layer and one output neuron achieved the second best performance and has a lower computational complexity. Therefore, this topology was selected to validate the neural network, as presented in next section.

## 6 Results

To evaluate the proposed fall detection algorithm we used the method reported in [5], [24], [19] [12]. Four possible situations can be observed: true negative (TN), movement without a fall classified correctly; false positive (FP), movement of daily living detected as a fall; true positive (TP), fall movement correctly classified; and false negative (FN), non detected falls. Three commonly performance metrics are estimated using these situations, namely sensitivity, specificity, accuracy.

Sensitivity (SE) is the ability to detect real falls and is defined as the ratio between the number of properly detected falls (true positives) and the falls that actually occurred (true positives and false negatives), as shown in Equation 5.

$$SE = \frac{TP}{TP + FN} \times 100\% \tag{5}$$

Specificity (SP), see Equation 6, it is the ability to filter false alarms and corresponds to the ratio between the movement without a fall classified correctly (true negatives) and the total number of movements without a fall (true negatives and false positives).

$$SP = \frac{TN}{TN + FP} \times 100\% \tag{6}$$

Accuracy (AC) is the ratio of true results in the data set, as presented in Equation 7.

$$AC = \frac{TP + TN}{TP + FP + FN + TN} \times 100\% \tag{7}$$

In the data set, 108 falls and 108 non-falls were used to validate the network. The performance metrics of the fall detection system were calculated and shown in Table 2.

**Table 2.** Results of the network validation.

| | | | | | |
|---|---|---|---|---|---|
| Positives | 108 | TP | 106 | FN | 2 |
| Negatives | 108 | FP | 2 | TN | 106 |
| Samples | 216 | | | | |
| Sensitivity: | 98.10% | Specificity: | 98.10% | Accuracy: | 98.10% |

One can conclude that, based on the validation process using 216 samples, falls can be distinguished from normal activities with a sensitivity of 98.10%, specificity of 98.10%, and accuracy of 98.10%.

Table 3 shows a comparison with several related work in which the acceleration sensor was placed on the user's waist and only [15] uses the sensor on the wrist. It can be observed that the results are within the range of previous

studies, indicating that the proposed algorithm is feasible to classify falls with the sensor placed on the user's wrist.

**Table 3.** Numerical comparisons with related works.

| Autor | Accuracy | Sensitivity | Specificity |
|---|---|---|---|
| Dinh and Struck, 2009 [5] | - | 99,64% | 96,00% |
| Yuwono et al., 2011 [24] | - | 99,56% | 97,65% |
| Sengto and Leauhatong, 2012 [17] | - | 99,50% | 96,25% |
| Vallejo et al., 2013 [19] | - | 98,60% | 98,40% |
| Kerdegari et al., 2013 [11] | 91,60% | 91,07% | 92,03% |
| Qingbin et al., 2013 [14] | 98,20% | - | - |
| Nukala et al., 2014 [12] | 98,80% | 100,00% | 94,10% |
| Vidigal et al., 2015 [20] | 96,40% | 98,30% | 85,00% |
| Nuttaitanakul and Leauhatong, 2015 [13] | 85,60% | - | - |
| Yodpijit et al., 2017 [23] | 99,23% | 99,37% | 99,00% |
| Quadros et al., 2018 [15] | 99,00% | 100,00% | 97,90% |
| **This work** | **98,10%** | **98,10%** | **98,10%** |

## 7 Conclusions

This work proposed a fall detection algorithm, based on the data acquisition of an acceleration sensor placed on the user's wrist, the time discretization of the signals and a multilayer perceptron neural network.

Different neural network topologies were tested and numerical comparisons demonstrates that the best MLP network topology is composed of 8 inputs, 8 neurons in the hidden layer and 1 neuron in the output layer, achieving a classification performance of 100% using the testing data set. During validation phase 108 falls and 108 daily living activities were collected from new users and results demonstrates the feasibility of the proposed MLP network to distinguish daily living movements from falls, achieving a sensitivity of 98.10%, specificity of 98.10% and accuracy of 98.10%. It is important to point-out that the proposed solution allows the classification of fall patterns regardless of the person's physical structure.

The achieved results are similar to those reported in related works in which one or more sensors were placed on the chest and/or waist. Therefore, one can conclude that the proposed fall detection system using the sensor on the user's wrist improves the acceptability and usability of the solution. For future works, we intend to use sensor fusion techniques in order to combine information from embedded heart beat and microphone sensors.

## References


1. Abbate, S., Avvenuti, M., Bonatesta, F., Cola, G., Corsini, P., and Vecchio, A. A smartphone-based fall detection system. *Pervasive and Mobile Computing* (2012), 883–899.
2. Bagalá, F., Becker, C., Cappello, A., Chiari, L., Aminian, K., Hausdorff, J. M., Zijlstra, W., and Klenk, J. Evaluation of accelerometer-based fall detection algorithms on real-world falls. *PloS one* (2012).
3. Caicedo, E., and Lopez, J. *Redes neuronales artificiales*, vol. 1. Programa editorial universidad del Valle, 2010.
4. De Carvalho, E. M., Delani, T. C. D. O., and Ferreira, A. A. Atenção à saúde no idoso no Brasil relacionada ao trauma. *Revista Uningá Review* (2018).
5. Dinh, C., and Struck, M. A new real-time fall detection approach using fuzzy logic and a neural network. *International Workshop on Wearable Micro and Nano Technologies for Personalized Health* (2009), 57–60.
6. Haykin, S. *Redes neurais: princípios e prática*. Bookman, 2007.
7. Haykin, S. *Neural networks and learning machines*, vol. 3. Pearson, NY, USA, 2009.
8. Joanneum, F. Cross-validation explained. *Institute for Genomics and Bioinformatics, Graz University of Technology* (2005).
9. Kangas, M. Development of accelerometry-based fall detection. Tech. rep., University of Oulu, 2011.
10. Kannus, P., Palvanen, M., Niemi, S., and Parkkari, J. Alarming rise in the number and incidence of fall-induced cervical spine injuries among older adults. *The Journals of Gerontology Series A: Biological Sciences and Medical Sciences* (2007), 180–183.
11. Kerdegari, H., Samsudin, K., Rahman Ramli, A., and Mokaram, S. Development Of Wearable Human Fall Detection System Using Multilayer Perceptron Neural Network. *International Journal of Computational Intelligence Systems* (feb 2013), 127–136.
12. Nukala, B. T., Shibuya, N., Rodriguez, A., Tsay, J., Lopez, J., Nguyen, T., Zupancic, S., and Lie, D. Y.-C. An Efficient and Robust Fall Detection System Using Wireless Gait Analysis Sensor with Artificial Neural Network (ANN) and Support Vector Machine (SVM) Algorithms. *Open Journal of Applied Biosensor* (2014), 29–39.
13. Nuttaitanakul, N., and Leauhatong, T. A novel algorithm for detection human falling from accelerometer signal using wavelet transform and neural network. In *Information Technology and Electrical Engineering (ICITEE), 2015 7th International Conference on* (2015), IEEE, pp. 215–220.
14. Qingbin, Z., Guohui, T., Nana, D., and Yanru, Z. A fall detection study based on neural network algorithm using AHRS. pp. 773–779.
15. Quadros, T., Lazzaretti, A., and Schneider, F. A movement decomposition and machine learning-based fall detection system using wrist wearable device. *IEEE Sensors 18* (2018), 5082 – 5089.
16. Refaeilzadeh, P., Tang, L., and Liu, H. Cross-validation. *Encyclopedia of database systems* (2016), 1–7.
17. Sengto, A., and Leauhatong, T. Human falling detection algorithm using back propagation neural network. In *Proceedings of Biomedical Engineering International Conference (BMEiCON)* (2012), pp. 1–5.

18. Stel, V. S., Smit, J. H., Pluijm, S. M., and Lips, P. Consequences of falling in older men and women and risk factors for health service use and functional decline. *Age and ageing* (2004), 58–65.
19. Vallejo, M., Isaza, C. V., and Lopez, J. D. Artificial Neural Networks as an alternative to traditional fall detection methods. In *IEEE International Conference in Medicine and Biology Society* (2013), pp. 1648–51.
20. Vidigal, M., Lima, M., and Neto, A. d. A. Elder Falls Detection Based on Artificial Neural Networks. *2015 Mexican International Conference on Artificial Intelligence (MICAI)* (2015), 226–230.
21. World Health, O. *W.H.O. global report on falls prevention in older age*. World Health Organization, 2017.
22. World Health, O. *Caidas*. World Health Organization, Augusto 2017.
23. Yodpijit, N., Sittiwanchai, T., and Jongprasithporn, M. The development of Artificial Neural Networks (ANN) for falls detection. *2017 3rd International Conference on Control, Automation and Robotics, ICCAR 2017* (2017), 547–550.
24. Yuwono, M., Su, S. W., and Moulton, B. Fall detection using a Gaussian distribution of clustered knowledge, augmented radial basis neural-network, and multilayer perceptron. In *6th International Conference on Broadband Communications and Biomedical Applications* (2011), pp. 145–150.