UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”

FACULDADE DE ENGENHARIA

CAMPUS DE ILHA SOLTEIRA

WILLIAN DE ASSIS PEDROBON FERREIRA

SISTEMA EMBARCADO EMPREGADO NO RECONHECIMENTO DE ATIVIDADES HUMANAS

Ilha Solteira

2017

WILLIAN DE ASSIS PEDROBON FERREIRA

SISTEMA EMBARCADO EMPREGADO NO RECONHECIMENTO DE ATIVIDADES HUMANAS

Dissertação apresentada à Faculdade de Engenharia – UNESP – Campus de Ilha Solteira como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica. Área de Conhecimento: Automação.

Prof. Dr. Alexandre César Rodrigues da Silva
Orientador

Ilha Solteira
2017

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

F383s

Ferreira, Willian de Assis Pedrobon.
Sistema embarcado empregado no reconhecimento de atividades humanas / Willian de Assis Pedrobon Ferreira. -- Ilha Solteira: [s.n.], 2017
81 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2017

Orientador: Alexandre César Rodrigues da Silva
Inclui bibliografia

1. Reconhecimentos de atividades humanas. 2. Redes neurais articiais. 3. Sistemas embarcados. 4. FPGA.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Ilha Solteira

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: Sistema embarcado empregado no reconhecimento de atividades humanas

AUTOR: WILLIAN DE ASSIS PEDROBON FERREIRA
ORIENTADOR: ALEXANDRE CESAR RODRIGUES DA SILVA

Aprovado como parte das exigências para obtenção do Título de Mestre em ENGENHARIA ELÉTRICA, área: AUTOMAÇÃO pela Comissão Examinadora:

Prof. Dr. ALEXANDRE CESAR RODRIGUES DA SILVA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. GUILHERME DE AZEVEDO E MELO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. MAICON APARECIDO SARTIN
Departamento de Computação / Universidade do Estado do Mato Grosso

Ilha Solteira, 24 de agosto de 2017

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me conceder saúde e por me fortificar a cada desafio enfrentado.

Aos meus pais, Emídio e Solange, por sempre me apoiarem.

Ao meu irmão, Tarcísio, companheiro incondicional.

Ao meu orientador, Prof. Dr. Alexandre César Rodrigues da Silva, por ter possibilitado o desenvolvido desse projeto sob seus ensinamentos.

À CAPES, pela concessão de uma bolsa de estudos, que me permitiu dedicar exclusivamente ao mestrado.

Aos amigos do Laboratório de Processamento de Sinais e Sistemas Digitais e a seus familiares, pela cooperação e todos os momentos de descontração.

À Melisa, pelo auxílio durante o desenvolvimento da pesquisa.

À Poliana, pelos incentivos, paciência e companhia.

A todos os integrantes do GOU Ângelus, pelos momentos de oração e partilha.

Aos amigos de república, Alexsander, Fernando, Luiz, Manoel, e Tiago, por terem me recebido gentilmente, e por toda amizade construída.

A todos direta ou indiretamente envolvidos, transmito minha imensa gratidão, e que de alguma maneira eu possa retribuir tudo o que fizeram por mim.

RESUMO

A utilização de sensores em ambientes inteligentes é fundamental para supervisionar as atividades dos seres humanos. No reconhecimento de atividades humanas, ou HAR (Human Activity Recognition), técnicas de supervisionamento são aplicadas para identificar as atividades realizadas em diversas aplicações, como no esporte e no acompanhamento de pessoas com necessidades especiais. O Sistema de Reconhecimento de Atividades Humanas (SIRAH) é empregado no reconhecimento de atividades humanas, utilizando um acelerômetro localizado na cintura da pessoa monitorada e uma Rede Neural Artificial para classificar sete atividades: em pé, deitado, sentado, caminhar, correr, sentar e levantar. Originalmente implementado no software MATLAB, realizava classificações *offline* em que os resultados não eram obtidos durante a execução das atividades. Apresenta-se, neste trabalho, o desenvolvimento de duas versões embarcadas do SIRAH, que executam o algoritmo de classificação durante a prática das atividades monitoradas. A primeira implementação foi efetuada no processador Nios II da Altera, que ofereceu a mesma exatidão do sistema *offline* com processamento limitado, pois o software consome 673 milissegundos para executar a classificação desejada. Para aprimorar o desempenho, outra versão foi implementada em FPGA utilizando a linguagem de descrição de hardware VHDL. O algoritmo de classificação opera em tempo real e é executado em apenas 236 microssegundos, garantindo total amostragem das acelerações.

Palavras-chave: Reconhecimentos de atividades humanas. Redes neurais artificiais. Sistemas embarcados. FPGA.

ABSTRACT

The use of sensors in smart environments is fundamental to monitor humans activities. In Human Activity Recognation (HAR), supervision techniques are employed to identify activities in several areas, such as in sport pratice and in people monitoring with special needs. The *Sistema de Reconhecimento de Atividades Humanas* (SIRAH) is used in human activities recognation, using an accelerometer located on the monitored person waist and an Artificial Neural Network to classify seven activities: standing, lying, seated, walking, running, sitting and standing. Originally, performed offline classifications executed in MATLAB software. In this work we present the development of two embedded SIRAH versions, which perform the classification algorithm during the monitored activities practice. The first implementation was performed on Altera's Nios II processor, that has been provided the same offline system accuracy, but with limited processing. To improve the performance, the other version was implemented in FPGA using the VHDL hardware description language, which performs real-time classifications, ensuring a lossless acceleration sampling.

Keywords: Human activity recognition. Artificial neural networks. Embedded system. FPGA.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AAL	*Ambient Assisted Living*
AC	*Alternating Current*
ADC	*Analog to Digital Converter*
ART	*Adapitive Resonance Theory*
BAN	*Body Area Network*
bps	*bits per second*
DC	*Direct Current*
FFT	*Fast Fourier Transform*
FPGA	*Field Programmable Gate Array*
FSM	*Finite State Machine*
GPS	*Global Position System*
HAR	*Human Activity Recognition*
I^2C	*Inter-Integrated Circuit*
IP	*Intellectual Property*
MEMS	*Micro-Electro-Mechanical Systems*
PCA	*Principal Component Analysis*
PDA	*Personal Digital Assistant*
RAM	*Randon Access Memory*
RFID	*Radio-Frequency Identification*
RISC	*Reduced Instruction Set Computer*
RNA	Rede Neural Artificial

ROM	*Read Only Memory*
SD	*Secure Digital*
SDRAM	*Synchronous Dynamic Random Access Memory*
SIRAH	Sistema de Reconhecimento de Atividades Humanas
SMA	*Signal Magnitude Area*
SoC	*System-on-Chip*
SPI	*Serial Peripheral Interface*
SVM	*Support Vector Machine*
VHDL	VHSIC *Hardware Description Language*
WSN	*Wireless Sensor Network*

SUMÁRIO

1 INTRODUÇÃO

No passado, a automação se restringia a processos financeiramente custosos, como aplicações industriais e militares. Contudo, a evolução tecnológica tem proporcionado a obtenção de dispositivos eletrônicos mais acessíveis, o que possibilitou a inserção de sistemas autônomos no cotidiano das pessoas. O aumento no interesse de pesquisadores e maior rentabilidade econômica também tornaram mais frequente o uso de entidades computacionais na rotina do ser humano (TOSCHI; CAMPOS; CUGNASCA, 2017).

Avanços na tecnologia da informação e em áreas como a de sensores são estudadas e desenvolvidas na Inteligência Ambiental, ou AmI (*Ambient Intelligence*), cujo objetivo é desenvolver ambientes capazes de detectar e interagir às necessidades dos usuários, de modo não intrusivo e transparente (RESENDES; CARREIRA; SANTOS, 2013).

A AmI é procedente de algumas áreas da computação, principalmente da computação ubíqua (WEISER, 1991), redes de sensores, inteligência artificial (IA) e computação consciente de contexto (SADRI, 2011). Outra área relacionada à AmI é a Internet das Coisas, ou IoF (*Internet of Things*), um paradigma que aplica conceitos de infraestruturas de redes para interconectar sensores a outros dispositivos (SAMARAH et al., 2017).

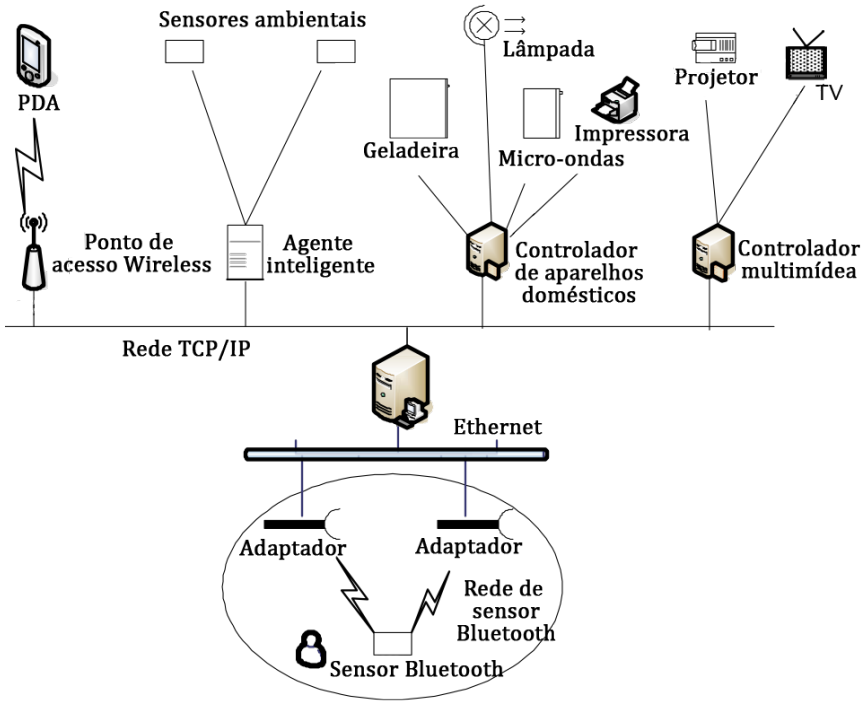
Um dos principais objetivos da AmI é oferecer funcionalidades que se adaptem às necessidades dos usuários (GARCIA-VALVERDE; SERRANO; BOTIA, 2014), que podem ser diferenciados aplicando-se metodologias que gerenciam o conhecimento extraído das informações captadas pelos sensores (HOMOLA et al., 2015). Acampora et al. (2013) destacaram algumas outras características da AmI:

- Embarcado: possui vários dispositivos interligados;
- Consciente de contexto: ações e estados são identificáveis;
- Antecipatório: é capaz de prever as necessidades de um indivíduo;
- Personalizável: o usuário pode configurar os dispositivos eletrônicos inseridos no ambiente, tornando-o mais eficaz;
- Transparente: execução discreta, não evidenciando as operações efetuadas.

As informações geradas pelas interfaces computacionais são transmitidas por redes projetadas de acordo com a aplicação. Redes de sensores sem fio (WSN, *Wireless Sensor Network*) como, por exemplo, a identificação por RFID (*Radio-Frequency Identification*), sensores infravermelhos e alguns protocolos de comunicação como *Bluetooth* (HAARTSEN, 2000) e *ZigBee* (BARONTI et al., 2007), captam e transmitem as informações presentes no ambiente. Avanços tecnológicos em microprocessadores, na nanotecnologia e a maior eficácia dos dispositivos de armazenamento têm possibilitado o gerenciamento dessas informações de forma eficiente (ACAMPORA et al., 2013; SADRI, 2011).

Exemplifica-se, na Figura 1, um ambiente equipado com equipamentos aplicados na AmI. Por meio de um PDA (*Personal Digital Assistant*), que também poderia ser um *smartphone* ou um notebook, o usuário é capaz de visualizar informações transmitidas pelas redes cabeada e *wireless*, além de acionar eletrodomésticos e aparelhos de multimídia.

Figura 1 – Exemplo de aplicação da AmI.



Fonte: Adaptado de Lu et al. (2008).

De acordo com Pauwels, Salah e Tavenard (2007), as redes de sensores presentes na AmI auxiliam na percepção e na análise de alguns aspectos, como:

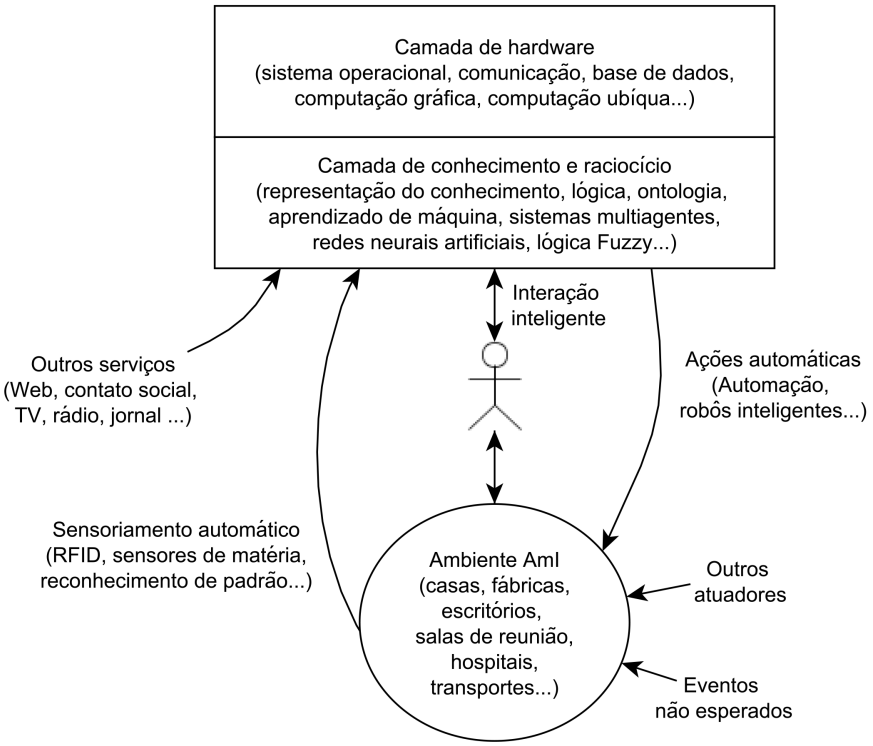
- Quem: identificação dos seres vivos inseridos no ambiente, representados por seres humanos e animais de estimação;
- Onde e quando: associação de intervalos de tempo relacionados à localização dos usuários;

- O que: reconhecimento de atividades considerando relações espaço-temporais, além de sinais linguísticos e não linguísticos;
- Por que: agregar significados e identificar padrões nas atividades desenvolvidas;
- Como: analisar os dados nas diferentes modalidades, identificando gestos, expressões e movimentos.

Kashyap et al. (2015) dividiram a AmI em três camadas principais:

- Camada de hardware: composta pelos dispositivos que captam e processam as informações analisadas. Alguns dos componentes dessa camada estão exemplificados na Figura 2, e também na Figura 1, como os adaptadores de sensor *Bluetooth*;
- Ferramentas e técnicas de IA: representam a base de conhecimento de um sistema inteligente por meio da aplicação de técnicas como RNAs (Redes Neurais Artificiais), lógica *fuzzy* e redes bayesianas. É apresentada na Figura 2 como Camada de conhecimento e raciocínio;
- Camada de rede: oferece suporte para as camadas anteriores pela interligação de todos os módulos do sistema. Devido ao rápido desenvolvimento, surgiram aplicações colaborativas por meio do uso de agentes inteligentes. Está exemplificada pela Rede TCP/IP e por tecnologias de conexão sem fio apresentadas na Figura 1.

Figura 2 – Divisão da AmI em camadas.



Fonte: Adaptado de Kashyap et al. (2015).

Os sensores utilizados na AmI são classificados em grupos, considerando-se as características e a localidade que são aplicados. Contudo, há principalmente duas categorias mais abrangentes, compreendendo os sensores ambientais e os sensores corporais.

Os sensores ambientais estão inseridos no ambiente de modo imperceptível aos usuários. Geralmente, são implementados por transdutores e por transceptores que, respectivamente, medem condições físicas do ambiente e as notificam, gerando algum sinal (ACAMPORA et al., 2013). Apresentam-se, na Tabela 1, alguns dos sensores ambientais mais comuns aplicados na AmI, incluindo o foco da medição e o formato das informações.

Tabela 1 – Exemplos de sensores ambientais utilizados na AmI.

Sensor	**Medição**	**Formato dos dados**
Sensor infravermelho passivo	Movimento	Categórico
Sensor infravermelho ativo	Movimento e identificação	Categórico
RFID	Informações de objetos	Categórico
Pressão	Pressão de superfícies	Numérico
Chaves magnéticas	Abrir e fechar de portas	Categórico
Sensor ultrassônico	Movimento	Numérico
Câmera	Atividade	Imagem
Microfone	Atividade	Som

Fonte: Adaptado de Acampora et al. (2013).

Sensores corporais são compostos por interfaces sensíveis aos sinais fisiológicos, que são de natureza analógica, gerados pelo corpo do usuário. Para possibilitar o processamento desses dados em sistemas computacionais, conversores analógico-digitais (ADC, de *Analog to Digital Converter*) são utilizados para converter os sinais analógicos em digitais (ACAMPORA et al., 2013).

Esse grupo de sensores está presente principalmente em redes de áreas corporais, ou BANs (*Body Area Networks*), cujo objetivo é monitorar parâmetros fisiológicos, aplicando-se sensores presentes nas roupas ou nos corpos das pessoas (FLOOS; AL-MOGREN, 2015). Apresenta-se, na Tabela 2, exemplos de sensores corporais, a medição que realizam e a respectiva taxa de dados gerada.

Tabela 2 – Sensores corporais presentes em BANs.

Sensor	Medição	Taxa de dados
Acelerômetro	Direção	Alta
Giroscópio	Orientação	Alta
Imagem/Vídeo	Atividade	Muito alta
Glicosímetro	Açúcar sanguíneo	Alta
Pressão sanguínea	Oscilométrico	Baixa
Sensor de gás CO2	Concentração de CO2	Muito baixa
Eletrocardiograma	Atividade cardíaca	Alta
Eletroencefalograma	Atividade cerebral	Alta
Eletromiógrafo	Atividade muscular	Muito alta
Termômetro	Temperatura sanguínea	Muito baixa

Fonte: Adaptado de Acampora et al. (2013).

Todos os sensores citados possuem diversas características específicas que devem ser previamente estudadas com o propósito de se obter o conhecimento técnico necessário para utilizá-los na AmI.

O desenvolvimento de sensores corporais, capazes de monitorar a movimentação do corpo humano durante períodos de tempo consideráveis, fez surgir várias pesquisas na área de reconhecimento de atividades humanas. Esses estudos auxiliam várias áreas, como a saúde, o monitoramento e a segurança pessoal (CHENG; LI; GUAN, 2017).

O SIRAH (Sistema de Reconhecimento de Atividades Humanas) é um sistema proposto por Durango (2017) aplicado no reconhecimento de atividades humanas. O sensor corporal utilizado é um acelerômetro, localizado na cintura do usuário, e as classificações são realizadas por uma RNA.

A metodologia desenvolvida por Durango (2017) garante até 95% de exatidão nas classificações, mas os resultados não são obtidos durante o desenvolvimento das atividades, pois os dados das acelerações são armazenados em um cartão SD (*Secure Digital*) e posteriormente o algoritmo de classificação é executado no software MATLAB.

Para aumentar a aplicabilidade do SIRAH, neste trabalho, é apresentado o desenvolvimento de duas versões embarcadas do sistema, as quais executam o algoritmo de classificação durante a prática das atividades monitoradas. A primeira implementação foi efetuada no processador Nios II da Altera, que ofereceu a mesma exatidão da versão que opera no MATLAB, mas com processamento limitado. Com a finalidade de incrementar o

desempenho, a segunda versão foi implementada em um FPGA, utilizando-se a linguagem de descrição de hardware VHDL (*VHSIC Hardware Description Language*), que realiza as classificações em tempo real.

No Capítulo 2, as técnicas aplicadas no reconhecimento de atividades humanas e as características do SIRAH são revisadas. No Capítulo 3, o desenvolvimento do SIRAH para operar no processador Nios II é descrito, além dos resultados obtidos. A descrição em hardware é detalhada no Capítulo 4 e, por fim, as conclusões são apresentadas no Capítulo 5.

1.1 REVISÃO DA LITERATURA

Atualmente a AmI está presente em diversos meios, variando desde ambientes fechados com pequenas dimensões até espaços abertos com áreas de abrangência extensas. Apresentam-se a seguir exemplos de trabalhos que aplicaram técnicas de monitoramento em ambientes inteligentes.

Al-Anbuky et al. (2015) automatizaram uma praça com a finalidade de aumentar a segurança nesse ambientes. Sensores podem ser agregados aos postes de iluminação para monitorarem os acontecimentos de modo transparente e, se alguma intervenção for necessária, atuadores como microfones são acionados. Os principais desafios destacados foram a escolha do posicionamento dos sensores e a preservação da privacidade.

Gjoreski et al. (2015) utilizaram dados de *smartphones* como áudio, tempo das ligações e dados de GPS para detectar os níveis de estresse dos estudantes de uma universidade. O trabalho avaliou técnicas de aprendizado de máquina e obteve até 60% de acerto, comprovando que efetuar a classificação do nível de estresse, apenas analisando os dados de um *smartphone*, é uma tarefa complexa.

A economia de energia é outra área em que técnicas de supervisionamento inteligentes são aplicadas. Cristani, Karafili e Tomazzoli (2015) gerenciaram o funcionamento de dispositivos eletrônicos por meio de um agendador de tarefas que obedece a ordem de prioridade predefinida pelo usuário e dos limites energéticos estabelecidos. O controle altera o estado de funcionamento dos equipamentos eletrônicos para minimizar o desperdício de energia.

Políticas para o gerenciamento de energia também são empregadas em ambientes domésticos. Mendez et al. (2015) desenvolveram um sistema que auxilia na conservação de energia em uma sala utilizando sensores e atuadores que acionam, com o auxílio da lógica

fuzzy, elementos como janelas, lâmpadas e condicionadores de ar. Além da energia elétrica, também considerou-se a energia solar com o intuito de futuramente aplicar algoritmos bio-inspirados.

O armazenamento e o consumo energético de dispositivos miniaturizados é um próprio desafio presente na aplicação de sensores corporais. Magno e Boyle (2017) revisaram as principais técnicas utilizadas no carregamento de baterias que alimentam sensores corporais, como a energia solar e a energia térmica.

Outra área em que aplica-se a AmI é no monitoramento de pessoas com necessidades especiais. Woznowski et al. (2015) desenvolveram um sistema que conecta médicos, familiares e cuidadores às pessoas que necessitam de acompanhamento. Alguns dos principais desafios existentes em AAL (*Ambient Assisted Living*), como o consumo de energia, escalabilidade e interoperabilidade foram mencionados. Os gastos financeiros foram reduzidos e os idosos obtiveram mais conforto com o monitoramento melhorado.

Tang et al. (2016) empregaram dispositivos portáteis e sensores de contexto para monitorar autistas no desenvolvimento de atividades diárias, e também no relacionamento com outras pessoas. Por meio da interação com seus mentores, os autistas foram auxiliados e obtiveram melhor desempenho nas situações avaliadas.

Shin et al. (2015) desenvolveram um andador inteligente que auxilia idosos a caminharem. Construiu-se uma estrutura com três rodas, duas acionadas por motores elétricos e outra para aumentar a estabilidade do protótipo, que é acionado por gestos reconhecidos por uma câmera de vídeo, e também e ser dirigido.

O monitoramento e identificação de atividades humanas também é contemplado com técnicas inteligentes. Ha e Choi (2016) utilizaram uma rede neural convolucional para classificar doze atividades diárias a partir dos dados gerados por acelerômetros, giroscópios e sensores de eletrocardiograma. Tran e Phan (2016) classificaram atividades humanas em um *smartphone* aplicando-se o algoritmo de aprendizado de máquina SVM (*Support Vector Machines*), que recebe dados dos sensores do próprio dispositivo.

A saúde é amplamente beneficiada aplicando-se conceitos da AmI. Chin e Tisan (2015) estimaram o nível de hidratação do corpo humano analisando líquidos que simularam a cor da urina, que foi classificada utilizando-se um microcontrolador, um sensor de cores e LEDs RGB. O sistema comprovou ser aplicável, pois obteve bons resultados na classificação proposta e preservou a privacidade do ambiente.

Zhou et al. (2015) estudaram a dieta de algumas pessoas por meio da utilização

de uma superfície instrumentada com sensores de pressão, que monitoraram a atividade exercida durante as refeições. Aplicando-se processamento de imagem e técnicas de visão computacional, foi possível estimar quais alimentos foram consumidos analisando-se o peso dos pratos e dos movimentos realizados pelos talheres.

Ramos-Garcia, Tiffany e Sazonov (2016) avaliaram a possibilidade de analisar as condições de saúde baseando-se em informações temporais da respiração, que foram coletadas por sensores compostos por fitas elásticas colocadas no tórax e no abdômen dos pacientes. Os resultados mostraram que essa técnica é flexível no reconhecimento de algumas atividades diárias, e que também pode ser anexada a outros estudos.

Joaquinito e Sarmento (2016) desenvolveram um sistema embarcado para medir a temperatura corporal e a taxa de batimento cardíaco. Os sinais biológicos amostrados por sensores corporais são processados em um FPGA (*Field Programmable Gate Array*) SoC *(System-on-Chip)*, constituído por unidades lógicos e um processador ARM implementados no mesmo circuito integrado. Os resultados obtidos durante o funcionamento do sistema são transmitidos para um *smartphone* por um módulo *Bluetooth*.

A aplicação da AmI no esporte foi efetuado por Vales-Alonso et al. (2015), que desenvolveram um sistema que auxilia na análise do treinamento de jogadores profissionais de voleibol. Os batimentos cardíacos foram estudados para verificar a fadiga e o nível do esforço físico efetuado, e um acelerômetro amostrou sinais dos movimentos. Esses dados auxiliaram no trabalho desenvolvido pelos técnicos e incrementou a qualidade do treinamento.

2 RECONHECIMENTO DE ATIVIDADES HUMANAS

O reconhecimento de atividades humanas (*Human Activity Recognition*, HAR) é um campo de pesquisa recente, que visa auxiliar as pessoas em várias situações por meio do desenvolvimento e aplicação de tecnologias de monitoramento, permitindo o reconhecimento de atividades realizadas nos ambientes assistidos. A base teórica aplicada no HAR origina-se principalmente da computação ubíqua, computação consciente de contexto e áreas relacionadas à multimídia, geralmente representadas pela visão computacional (TONCHEV et al., 2015).

A análise de vídeo para caracterizar gestos é foco de várias pesquisas. Pelo rastreamento realizado por câmeras de vídeo, objetiva-se entender o comportamento humano por meio da identificação de padrões nas atividades desenvolvidas, porém essa abordagem é alvo de algumas controvérsias. Questiona-se a privacidade das pessoas inseridas nos ambientes monitorados integralmente, além do alto custo computacional existente nas técnicas de processamento de vídeo (KAKDE; GULHANE, 2015; LARA; LABRADOR, 2013).

Para contornar essas questões, a utilização de sensores corporais tornou-se mais frequente, pois são capazes de captar dados por longos períodos de tempo sem oferecer desconforto aos usuários. A obtenção de sensores cada vez menores e com baixo custo de fabricação contribuem para esse cenário, uma vez que são leves e possuem autonomia de carga relevante (LUBINA; RUDZKI, 2015).

A aplicação de sensores corporais permite avaliar sinais de movimento (aceleração e velocidade angular), localização (pressão dos pés, posição de GPS (*Global Position System*), sinais vitais (ECG), entre outros (LARA; LABRADOR, 2013; PREECE et al., 2009).

Apresenta-se, na Tabela 3, alguns exemplos de atividades similares estudadas no HAR. O reconhecimento de padrões também é empregado na medicina na realização de diagnósticos, na reabilitação de pacientes e na vigilância da saúde de idosos. A indústria também é contemplada com tecnologias para identificar características, destacando-se a

produção de jogos eletrônicos e de produtos esportivos (UDDIN; UDDINY, 2015).

Tabela 3 – Alguns grupos de atividades estudadas em HAR.

Grupo	**Atividades**
Atividades gerais	Andar, correr, deitar, sentar, permanecer parado, usar o elevador, subir e descer escadas
Transporte	Andar de ônibus, ciclismo e dirigir
Telefone	Mensagem de texto e ligação
Atividades diárias	Comer, beber, trabalhar no computador, assistir TV, ler, escovar os dentes e alongamento
Exercício	Remo, levantar pesos e fazer flexões
Militar	Rastejar e ajoelhar
Parte superior do corpo	Mastigar, falar, suspirar e mover a cabeça

Fonte: Adaptado de Lara e Labrador (2013).

O emprego de técnicas para identificar atividades geralmente se depara com alguns desafios. Benmansour, Bouchachia e Feham (2015) destacaram alguns, descritos a seguir:

- Reconhecer e distinguir tarefas que se relacionam e que ocorrem no mesmo instante e no mesmo espaço;
- Ambiguidade na interpretação das atividades em cenários similares. Por exemplo abrir a geladeira pode submeter a várias situações, como alimentação e limpeza;
- Quantidade de dados adquiridos na amostragem. Usualmente as atividades geram dados heterogêneos, ou seja, não há um padrão bem definido na quantidade de informações representadas de maneiras distintas;
- Atividades executadas por mais de uma pessoa. O contínuo monitoramento de ambientes deve ser capaz de diferenciar os momentos em que há apenas uma pessoa inserida no meio assistido dos momentos em que há várias pessoas.

Outros desafios comuns em HAR foram citados por Bulling, Blanke e Schiele (2014). A variação na execução de uma atividade pode ser alta, pois há diferenças no modo em que as pessoas desenvolvem os movimentos. Além disso, há fatores que afetam na execução de tarefas como fadiga, estresse e também o estado emocional da pessoa assistida, ou o estado do ambiente em si.

A qualidade das informações amostradas também é um fator crítico. Alguns sensores são sensíveis e sujeitos à variações paramétricas (temperatura, por exemplo), o que altera a

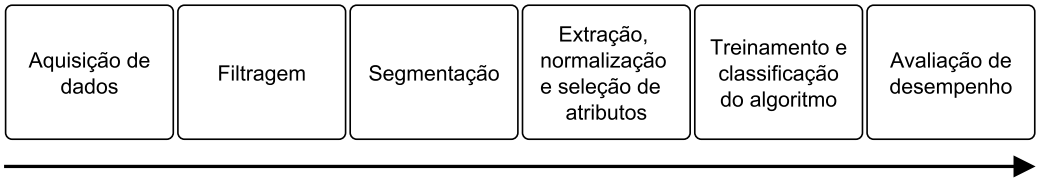
acurácia da medição. A quantidade elevada de sensores, períodos excessivos para captação de dados e grandes grupos de pessoas no mesmo ambiente também dificultam a aplicação de técnicas para o reconhecimento de ações.

Resultados eficientes na aplicação de práticas existentes em HAR estão intimamente ligados à qualidade da representação das atividades por meio dos sinais amostrados pelos sensores corporais. Geralmente, efetua-se um pré-processamento para garantir a integridade dos dados, que são filtrados e divididos em segmentos de tempos denominados janelas. Por fim, atributos de caracterização são extraídos desses sinais para serem utilizados no algoritmo de aprendizado. Na classificação, cada janela é associada a um rótulo que identifica uma atividade (LARA; LABRADOR, 2013).

2.1 ETAPAS EMPREGADAS EM HAR

Os sistemas de classificação empregados no HAR operam de duas maneiras, *online* e *offline*. Na abordagem *offline*, cujas etapas estão representadas na Figura 3, o objetivo principal é realizar o treinamento do algoritmo de classificação e, por meio da avaliação dos resultados, identificar melhorias para incrementar o desempenho do sistema.

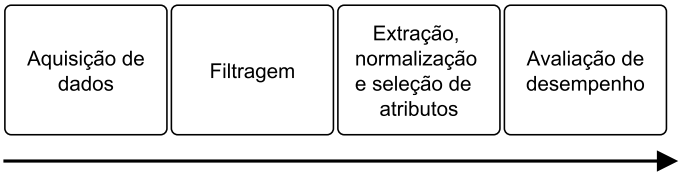
Figura 3 – Etapas da classificação *offline*.



Fonte: Adaptado de (UDDIN; UDDINY, 2015).

Nas aplicações *online*, as classificações são realizadas durante a execução das atividades. Os sistemas devem ser eficientes para garantir a obtenção de resultados em tempos reduzidos, ou até mesmo em tempo real. A abordagem *online* possui menos fases em comparação às classificações *offline*, como pode ser visualizado na Figura 4, pois não há a segmentação dos dados nem o treinamento do classificador.

Figura 4 – Fases da classificação *online*.



Fonte: Elaboração do próprio autor.

Os atributos obtidos após a filtragem do sinal amostrado de uma janela são submetidos ao algorítimo de classificação, cujos parâmetro são calculados no treinamento *offline*, gerando-se a identificação da atividade praticada.

A seguir, as fases existentes no reconhecimento de atividades humanas são apresentadas com maiores detalhes.

2.1.1 Aquisição de dados

Dentre os principais componentes utilizados no reconhecimento de atividades na AmI, os sensores são itens indispensáveis. No Capítulo 1 foram mencionadas as principais categorias de sensores que monitoram o ambiente, bem como os usuários, sendo alguns exemplificados.

Os sensores ambientais são importantes na detecção de atividades via monitoramento da interação dos usuários com o ambiente, embora sejam limitados devido à abrangência restrita, principalmente a ambientes internos. Nesse sentido, os sensores corporais são mais adequados para o reconhecimento de atividades, pois são mais flexíveis e acompanham os usuários durante a execução das atividades (WANG et al., 2016).

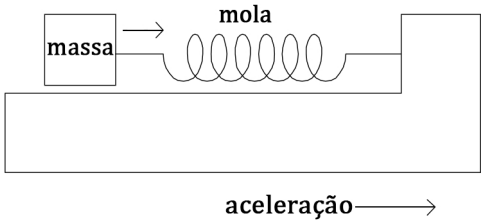
No grupo dos sensores corporais, o acelerômetro vem ganhando a preferência dos pesquisadores nas aplicações em locais fechados, como no acompanhamento clínico e também em ambientes abertos, uma vez que são vários os benefícios que oferecem (MATHIE et al., 2004), dos quais destacam-se:

- são sensíveis à frequência e à intensidade dos movimentos, ao contrário dos pedômetros que respontem apenas à inclinações e a impactos;
- mensuram a movimentação e a inclinação do corpo, tornando-se superiores em relação a outros sensores, incapazes de medir características estáticas;
- avanços na área de MEMS (*Micro-Electro-Mechanical Systems*), que desenvolve sensores e atuadores miniaturizados, permitiram o desenvolvimento de acelerômetros pequenos e de baixo custo.

Acelerômetros são dispositivos utilizados para medição da aceleração aplicada sobre um eixo de sensibilidade. Existem diferentes mecanismos que medem a mudança da aceleração, como os sensores piezoelétricos, piezosensitivos, capacitivos, entre outros. Apesar de diferentes características de fabricação, todos baseiam-se no sistema massa-mola representado na Figura 5. Esse sistema mede a aceleração por meio da variação

sofrida pela mola, que é reflexo do deslocamento do corpo ligado à extremidade móvel (MATHIE et al., 2004).

Figura 5 – Sistema massa-mola utilizado em acelerômetros.



Fonte: Adaptado de Mathie et al. (2004).

O posicionamento do acelerômetro no corpo é um fator importante e deve ser determinado de acordo com as atividades estudadas. Mathie et al. (2004) realçaram alguns locais, nos quais acelerômetros foram anexados ao corpo e as respectivas atividades mensuradas:

- coxa ou tornozelo, utilizado no estudo do movimento da perna durante uma caminhada;
- pulso para averiguar a bradicinesia, que representa a lentidão causada pelo mal de Parkinson;
- ambos os braços, para estudar a oscilação causada pelo Parkinson;
- peito, para medir a tosse;
- centro de massa do corpo, com o intuito de analisar o movimento de todo o corpo.

A utilização de acelerômetros requer pré-ajustes em configurações e também escolhas de alguns parâmetros. Ressalta-se que a largura de banda, medida em *hertz* (Hz), e a medida dos dados amostrados, expressados em g, que é o valor da aceleração da gravidade, dependem da atividade mensurada e da posição na qual o acelerômetro encontra-se anexado ao corpo (MATHIE et al., 2004). A taxa de amostragem típica utilizada em acelerômetros é de 25Hz, podendo chegar a 100Hz (BULLING; BLANKE; SCHIELE, 2014).

A evolução tecnológica empregada nos acelerômetros gerou dispositivos eficientes e possibilitou a aplicação em várias áreas. Na engenharia, por exemplo, são utilizados em sistemas de segurança para automóveis, no monitoramento de estruturas e de máquinas, na detecção de abalos sísmicos e em sistemas de navegação de aeronaves. Na medicina são empregados no monitoramento de pacientes e também na instrumentação cirúrgica (ROY; MANDAL; HANUMAIAH, 2016).

2.1.2 Filtragem

Costuma-se aprimorar a qualidade os dados brutos por meio da aplicação de filtros. Nesse processo, minimizam-se possíveis frequências indesejadas e erros aleatórios ocorridos durante a amostragem (SILVA; GALEAZZO, 2013).

Assim como em outras áreas, em HAR é comum a utilização de filtros passa alta e passa baixa. Quando os sensores utilizados são acelerômetros, no geral, pode-se afirmar:

- filtro passa baixa: utilizado para filtrar a componente AC (*Alternating Current*) do sinal original e minimiza ruídos indesejados (BAYAT; POMPLUN; TRAN, 2014);
- filtro passa alta: filtra a componente DC (*Direct Current*) do sinal original, que é altamente influenciado pela força da gravidade (SILVA; GALEAZZO, 2013).

2.1.3 Segmentação de dados

A finalidade da segmentação de dados é dividir uma atividade amostrada por um sensor em eventos menores, obtendo-se como resultado um conjunto de fragmentos consecutivos que representam o sinal original (HONG; NUGENT, 2012).

Dentre os métodos aplicados na segmentação, destacam-se a segmentação por janelas de tempo e a fundamentada em energia. Esta última se baseia na diferença de intensidade existente entre as atividades estudadas no HAR, o que reflete diretamente no nível de energia adquirido pelos sensores (BULLING; BLANKE; SCHIELE, 2014).

A segmentação por janelas percorre o sinal em séries de tempo para extrair as informações de cada segmento. Segundo Preece et al. (2009), existem três variações no conceito de janelas:

- janelas deslisantes: o sinal original é dividido em parcelas adjacentes de tamanho fixo, cuja duração costuma ser mantida entre 0,25 e 6,7 segundos nas aplicações voltadas ao reconhecimento de atividades. Não necessita de pré-processamento, o que a torna uma abordagem mais eficiente e mais utilizada;
- janelas definidas por eventos: é necessário aplicar pré-processamento no sinal original para identificar transições de eventos específicos que definem o intervalo de cada janela. Como a duração das atividades não é uniformemente distribuída no tempo, as janelas possuem tamanhos distintos;
- janelas definidas por atividades: identifica os momentos em que ocorrem mudanças

de atividades por meio da alteração na frequência do sinal. Esses eventos delimitam o intervalo de cada janela, que representa uma atividade.

2.1.4 Extração, normalização e seleção de atributos

A duração das atividades realizadas pelos humanos geralmente são mais extensas comparadas às taxas de amostragem dos sensores. Logo, a leitura momentânea realizada por um sensor não gera informações suficientes para classificar uma ação. A utilização da segmentação de dados resolve esse inconveniente, pois considera intervalos e não somente um único instante de tempo.

Contudo, outra questão que surge é como comparar duas janelas com a finalidade de classificá-las, já que dificilmente dois sinais serão exatamente iguais. Isso motiva a aplicação da extração de atributos em cada janela, cujo objetivo é obter medidas quantitativas e relevantes dos sinais para então poder compará-los (LARA; LABRADOR, 2013).

Preece et al. (2009) revisaram as principais metodologias empregadas na extração de atributos nos principais grupos citados na literatura:

- atributos heurísticos: metodologia originada e influenciada pelo estudo de como um movimento ou uma postura específica influencia nas características dos sinais amostrados por sensores corporais. Alguns exemplos são o ângulo relativo à vertical utilizado na classificação de posturas estáticas, a velocidade angular aplicada na identificação de posturas e transições entre si, a técnica de SMA (*Signal Magnitude Area*), que mede a área do gráfico da aceleração filtrada por um passa alta, acelerações pico a pico, valor médio retificado e o valor eficaz para diferenciar atividades entre estáticas ou dinâmicas;
- atributos no domínio do tempo: extraídos diretamente das janelas dos sinais amostrados, geralmente são de natureza estatística, como a média, a mediana e a variância;
- atributos no domínio da frequência: extrair atributos no domínio da frequência requer que os sinais sejam convertidos para este domínio, o que geralmente é feito por meio da aplicação da FFT (*Fast Fourier Transform*), cuja saída fornece coeficientes que caracterizam a amplitude e a distribuição de energia do sinal original. Há diferentes métodos aplicados para caracterizar a distribuição desses coeficientes no domínio da frequência, como a frequência média, a energia espectral e a entropia,

que possibilitam a classificação de atividades analisando-se os padrões nos sinais de aceleração.

Outra técnica bem difundida para se extrair características de sinais, utilizada no HAR, é a *wavelet*, cuja análise pode ser empregada no domínio do tempo e também no da frequência, sendo a versão discreta dessa transformação mais aplicada no monitoramento de atividades. A implementação decompõe o sinal original sucessivas vezes, separando-o em frequências baixas e altas nomeadas, respectivamente, como coeficientes de aproximação e de detalhe (PREECE et al., 2009).

A técnica *wavelet* separa um sinal em um determinado número de coeficientes, que agregam informações em faixas de frequências específicas. Como esses coeficientes representam o sinal original em toda extensão, possuem informações em bandas específicas de frequência e também das trocas temporais do sinal relativo às faixas de frequência (PREECE et al., 2009).

A cada decomposição do sinal original, a quantidade de amostras de tempo diminui, assim como a resolução temporal. Por outro lado, a resolução da frequência é incrementada, o que gera uma boa resposta em frequência, mesmo nos intervalos mais baixos, e também uma melhor resolução de tempo nas frequências mais altas (PREECE et al., 2009).

Uma técnica que pode ser utilizada para incrementar a qualidade e eficiência do treinamento do algoritmo de classificação, e consequentemente garantir melhores desempenhos dos classificadores, é a normalização, que transforma um grupo de informações, ajustando-as para que fiquem em um intervalo estabelecido. (DINÇ et al., 2014).

Um dos métodos de normalização mais comuns é a transformada *z-score*, também conhecida como *escore* padrão. Para cada informação original x_i, o valor *escore* padrão z_i associado é obtido por:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{1}$$

sendo μ a média e σ o desvio padrão dos atributos.

Uma das propriedades do *escore* padrão é não mudar a ordem nem a forma dos dados ajustados. Isso auxilia a visualização e a interpretação de um atributo, e também a comparação de variáveis representadas em escalas distintas (LAI; JHAN; WANG, 2010).

Quantidades elevadas de exemplos de treinamento demandam longos períodos de tempo para se extrair os atributos desejados. Com a finalidade de otimizar esse processo, é possível aplicar técnicas de seleção de atributos.

Selecionar características essenciais que representam uma atividade é um desafio, e a dimensão das informações é o ponto mais crítico, pois cresce exponencialmente à medida que o espaço dos dados se expande. Esse problema é amenizado aplicando-se taxas de amostragens menores e pelo emprego da seleção e da diminuição de atributos (TONCHEV et al., 2015).

Há chances de que alguns dos atributos selecionados na etapa de extração contenham informações irrelevantes ou repetidas, o que possivelmente afetará na qualidade do reconhecimento das atividades. A seleção de atributos auxilia os modelos de aprendizado, pois considera apenas as melhores qualidades e também diminui o custo computacional necessário (LARA; LABRADOR, 2013).

No HAR, algumas técnicas utilizadas na seleção de atributos se fundamentam principalmente em encontrar características que ofereçam menor redundância e maior relevância mutualmente entre as classes de atributos. Outra estratégia é basear-se no princípio de que os atributos pertencentes a uma classe são altamente semelhantes aos vizinhos da mesma classe, e que simultaneamente não se relacionam aos integrantes de outros grupos (LARA; LABRADOR, 2013).

Na seleção de atributos, esses são analisados separadamente, contudo há técnicas que reduzem a dimensão dos dados originais em um espaço de extensão menor. A Análise dos Componentes Principais, ou PCA (*Principal Component Analysis*) é usualmente utilizada para esse fim. Baseia-se na abordagem da correlação dos dados de entradas por meio de propriedades da ortogonalidade com o objetivo de transformar os dados originais, que geralmente são correlacionados em um conjunto menor de variáveis não correlacionadas (ANDREU; BARUAH; ANGELOV, 2011).

As novas variáveis são combinações lineares das originais, o que não reduz a quantidade de varáveis a serem mensuradas, mas sim o número de entradas para o classificador. As principais informação dos dados originais são preservadas, obtendo-se a classificação correta desejada (ANDREU; BARUAH; ANGELOV, 2011).

2.1.5 Treinamento e classificação do algoritmo

O treinamento dos algoritmos de classificação é separado em paradigmas, de acordo com a estrutura utilizada. Costuma-se dividi-los em aprendizado supervisionado e não supervisionado (ABU-MOSTAFA; MAGDON-ISMAIL; LIN, 2012).

No aprendizado supervisionado os dados utilizados são identificados com marcas, também conhecidas como rótulos, em que elementos predefinidos modelam o treinamento do sistema, relacionando cada saída com a respectiva entrada. Árvores de decisão, alguns tipos de RNAs, SVMs e estatísticas bayesianas são exemplos de classificadores que utilizam o aprendizado supervisionado (ALSHEIKH et al., 2014).

Já o aprendizado não supervisionado não agrega nenhuma identificação aos exemplos de treinamento, fazendo com que o sistema não forneça nenhum dado na saída, e o principal objetivo é classificar as amostras obedecendo algum parâmetro de semelhança. Pode-se citar o PCA como um dos principais representantes deste paradigma.

Após o treinamento, o algoritmo de classificação opera de forma autônoma. Aplicando-se técnicas de validação, é possível estimar a qualidade dos resultados obtidos por meio da análise dos resultados obtidos.

2.1.6 Avaliação de desempenho

Para avaliar o desempenho de algoritmos que empregam aprendizado de máquina, recomenda-se que o conjunto dos exemplos de treinamento seja disjunto do conjunto dos exemplos de prova, o que possibilita verificar a capacidade de predição do algoritmo quando é submetido a novos dados não utilizados no treinamento. Habitualmente duas técnicas são utilizadas nessas análises: a validação aleatória simples e a validação cruzada (LABRADOR; YEJAS, 2013; KOHAVI, 1995)

Na validação simples, o conjunto de exemplos de treinamento é dividido aleatoriamente em dois grupos mutualmente exclusivos, permanecendo uma parcela para o treinamento e a outra para os testes. Em média, costuma-se adotar a proporção de 2/3 dos elementos para o treinamento do algoritmo e 1/3 para os testes (LABRADOR; YEJAS, 2013). Por exemplo, Durango (2017) considerou 70% e 30% dos exemplos para efetuar o treinamento e os testes, respectivamente, e Basterretxea, Echanobe e Campo (2014) utilizaram 60% dos exemplos no treinamento e os 40% restantes na validação.

Já na validação cruzada, o conjunto de exemplos de treinamento D é dividido em k

conjuntos mutualmente exclusivos $D_1, D_2, ..., D_k$ de tamanhos aproximadamente iguais. São executados k treinamentos, sendo que em cada iteração um conjunto D_k é utilizado no treinamento e os demais D_{k-1} na predição. Ao final, calcula-se a média de todas as validações para determinar a exatidão geral do modelo proposto (KOHAVI, 1995).

A seguir, as principais características do SIRAH são apresentadas.

2.2 SISTEMA DE RECONHECIMENTO DE ATIVIDADES HUMANAS

Na forma de protótipo, o SIRAH é um sistema desenvolvido para ser empregado no HAR e fundamenta-se na aplicação de um acelerômetro do modelo MMA7455L, fabricado pela Freescale Semiconductor. De acordo com Semiconductor (2009), as principais características desse dispositivo são:

- 3 eixos (XYZ);
- comportamento capacitivo;
- saída digital com comunicação I^2C (*Inter-Integrated Circuit*) e SPI (*Serial Peripheral Interface*);
- acelerações configuráveis: ±2g/±4g/±8g.

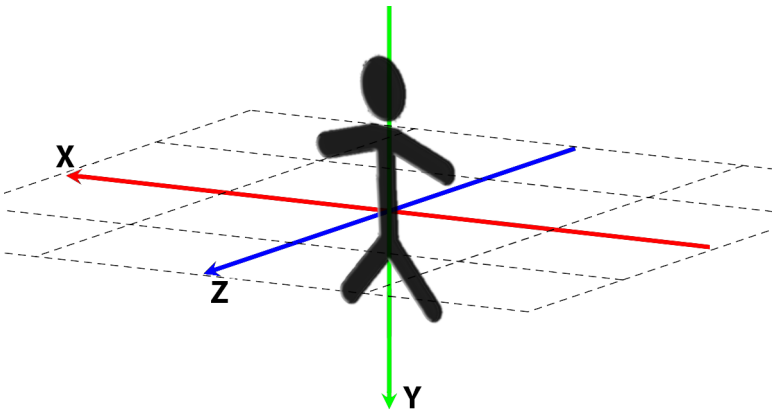
O protótipo desenvolvido é utilizado na cintura, local que oferece conforto ao usuário e garante que a amostragem dos sinais das atividades seja realizada com boa qualidade. Além do acelerômetro, os seguintes componentes completam o sistema:

- Arduino UNO, que contém um microcontrolador ATmega328P;
- leitor de cartão SD, que incrementa a memória de armazenamento;
- *ZigBee*, utilizados na transmissão de dados sem fio do protótipo para um microcomputador.

O posicionamento do acelerômetro determina a orientação de cada um dos três eixos. Representam-se, na Figura 6, as direções das acelerações do sistema SIRAH, que são decorrentes da localização e pelo aspecto estrutural do protótipo na cintura do usuário. A sensibilidade da aceleração definida foi de ±8g para garantir que todos os sinais sejam adquiridos corretamente e a taxa de amostragem utilizada foi de 40 Hz, isto é, a cada 25 milissegundos armazena-se uma amostra de cada eixo do acelerômetro no cartão SD.

No total, o sistema proposto é capaz de identificar e classificar sete atividades separadas entre dinâmicas e estáticas:

Figura 6 – Direção dos eixos do acelerômetro utilizado no SIRAH.



Fonte: Adaptado de Durango (2017).

- Atividades estáticas: em pé, deitado e sentado;
- Atividades dinâmicas: caminhar, correr, sentar e levantar.

Nos testes realizados, o mesmo indivíduo realizou todas as atividades, cada qual monitorada e amostrada durante seis minutos. As ações correr e caminhar foram executadas em uma esteira de academia.

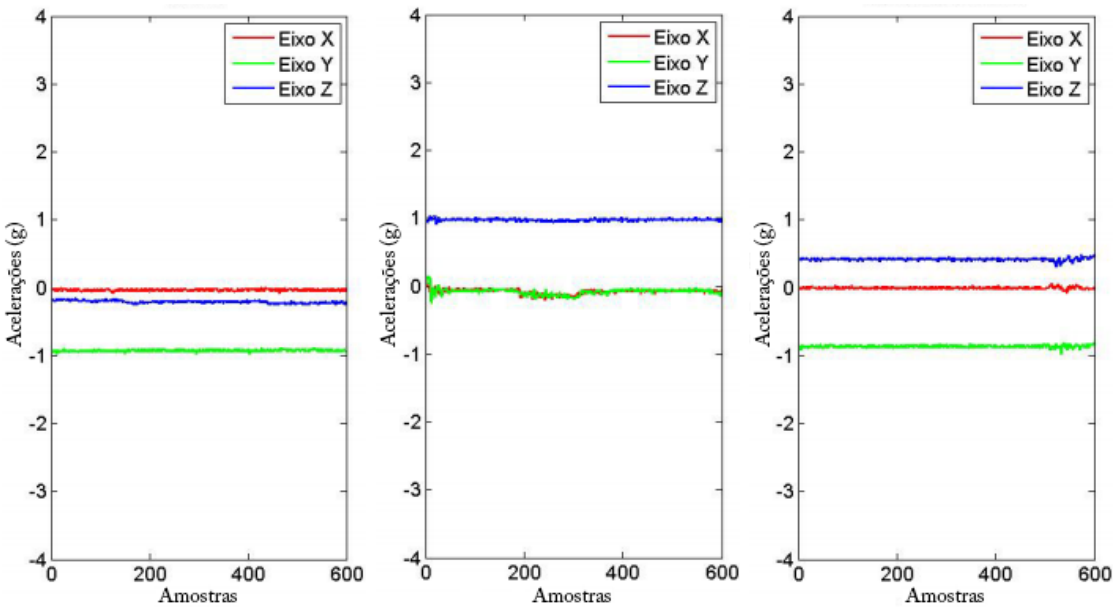
A aquisição dos dados de cada atividade é efetuada com o protótipo colocado na cintura do usuário. O microcontrolador lê os dados das acelerações via comunicação I^2C e em seguida, os armazena no cartão SD por meio da comunicação SPI.

Depois que todas as atividades são amostradas, as informações são transmitidas do cartão SD para um microcomputador. A versão do SIRAH projetado por Durango (2017), que realiza classificações *offline*, foi implementada utilizando-se o software MATLAB.

Os sinais originais das atividades em pé, deitado e sentado gerados pelo acelerômetro durante 15 segundos estão representados nas Figuras 7(a), 7(b) e 7(c), respectivamente. Quando a direção do eixo do acelerômetro está na mesma direção que a aceleração da gravidade, a medição possui sinal negativo, como pode ser visualizado no eixo Y da Figura 7(a), cujo valor é -1, que corresponde à posição em pé. Se o eixo estiver na direção oposta à aceleração da gravidade, a medição é positiva como nota-se no valor +1 do eixo Z da posição deitado. Quando o usuário está sentado, a parte frontal do acelerômetro fica com aproximadamente 45° de inclinação em relação ao solo, o que produz a aceleração aproximada de +0,4 no eixo Z presente na Figura 7(c).

Exemplos de acelerações das ações dinâmicas, como caminhar e correr são respectivamente apresentadas na Figura 8(a) e 8(b). Nota-se uma semelhança no padrão

Figura 7 – Exemplos de acelerações das atividades estáticas.

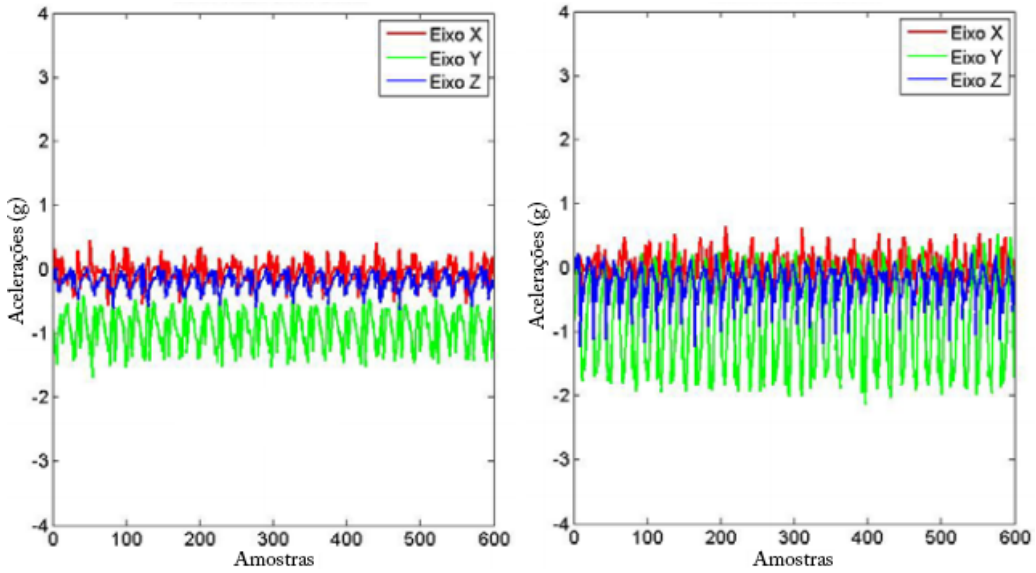


(a) Pé (b) Deitado (c) Sentado

Fonte: Adaptado de Durango (2017).

desses sinais, e a principal diferença está na amplitude, devido à maior intensidade nos movimentos realizados na atividade correr.

Figura 8 – Acelerações das atividades caminhar e correr.



(a) Caminhando (b) Correndo

Fonte: Adaptado de Durango (2017).

As atividades sentar e levantar foram coletadas continuamente e o sinal obtido está

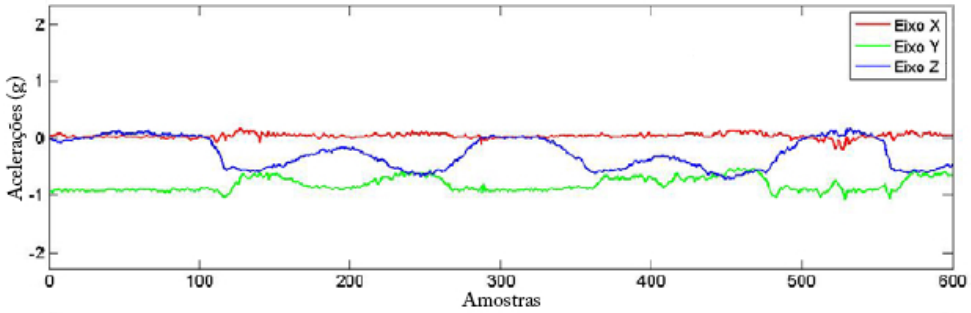
apresentado na Figura 9. Para distingui-las, desenvolveu-se um algoritmo para separar os dados de cada atividade, como segue:

1. Considerar apenas as acelerações dos eixos Y e Z, pois são as que mais sofrem variações, filtrá-las e multiplicá-las ponto a ponto para obter o sinal resultante, representado na cor preta da Figura 10. Nota-se a presença de máximos locais que correspondem às transições entre as ações sentar e levantar;

2. Identificar os pontos de máximo e dividir a distância entre dois máximos ao meio, obtendo-se o intervalo de cada atividade presentes na Figura 10;

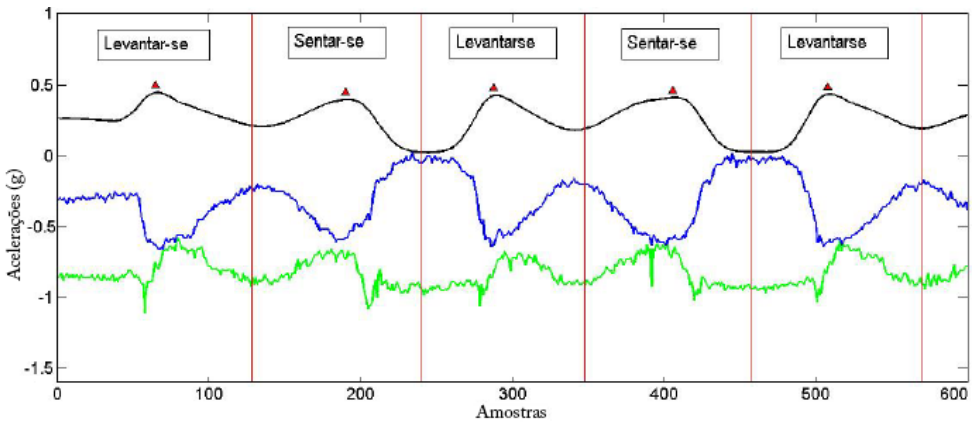
3. Para determinar a atividade que cada intervalo representa, verifica-se o valor mínimo de cada intervalo. Se estiver antes do valor máximo, a atividade é levantar, e se o mínimo estiver localizado após o máximo, a ação é sentar.

Figura 9 – Acelerações das atividades sentar e levantar captadas continuamente.



Fonte: Adaptado de Durango (2017).

Figura 10 – Multiplicação dos eixos Y e Z, identificação dos máximos locais e subdivisão da aceleração das atividades sentar e levantar.



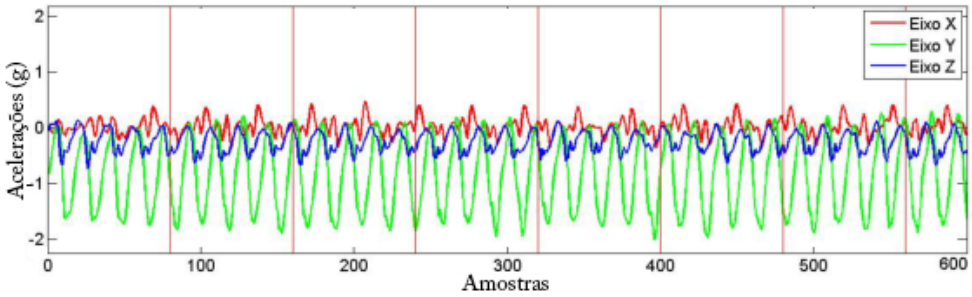
Fonte: Adaptado de Durango (2017).

Finalizada a coleta de dados, a etapa subsequente é a filtragem. No SIRAH, aplicou-se um filtro passa baixa para remover ruídos indesejados adquiridos via erros ocorridos

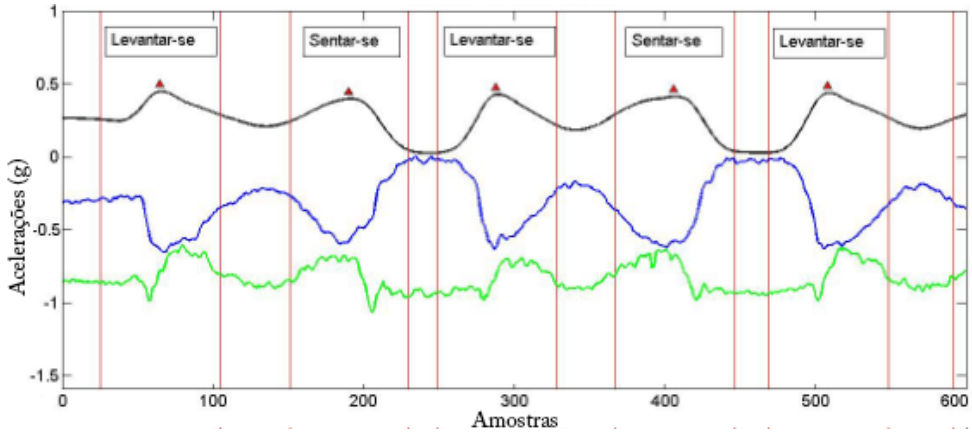
durante a amostragem. Não foi aplicado nenhum filtro passa alta, pois as baixas frequências correspondem ao elemento gravitacional nos sinais do acelerômetro, sendo uma particularidade essencial para classificar as atividades consideradas.

Na segmentação, optou-se pelo uso de janelas com tempo fixo graças à facilidade que oferecem, além de boa qualidade nos resultados. A duração das janelas foi fixada em 2 segundos e está exemplificada na Figura 11(a), que representa a atividade correr segmentada. Como o procedimento de amostragem para as atividades sentar e levantar baseou-se na identificação dos máximos locais como sendo o centro das janelas, parte dos dados são desconsiderados com o intuito de manter as janelas com 80 amostras, como pode-se visualizar na Figura 11(b).

Figura 11 – Exemplos de atividades segmentadas.



(a) Correr.



(b) Sentar e levantar.

Fonte: Adaptado de Durango (2017).

A amostragem realizada durante 6 minutos totalizou 360 segundos de medição para cada ação monitorada. Com a segmentação desses sinais em janelas com duração de 2 segundos, gerou-se 180 exemplos para cada uma das 7 atividades e no total, foram obtidos 1280 exemplos.

Na extração de atributos, de cada janela do sinal segmentado, obteve-se as características a seguir. O n que aparece em algumas equações é a quantidade de amostras de uma janela, que nesse caso é 80.

- Atributos no domínio do tempo
  - Média
    - Descrição: média aritmética dos valores das acelerações na janela. Total de atributos em uma janela: 3 (μ_x, μ_y e μ_z)
    - Equação:

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2}$$

  - Variância
    - Descrição: medida da dispersão estática, que indica o quão longe se encontram os valores obtidos dos esperados. Total de atributos em uma janela: 3 (σ_x^2, σ_y^2 e σ_z^2)
    - Equação:

$$\sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)^2 \tag{3}$$

  - Desvio padrão
    - Descrição: média da dispersão dos dados. Total de atributos em uma janela: 3 (σ_x, σ_y e σ_z)
    - Equação:

$$\sigma_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)^2} \tag{4}$$

  - Variação
    - Descrição: diferença entre os valores máximo e mínimo da janela. Total de atributos em uma janela: 3 ($var(x)$, $var(y)$ e $var(z)$)
    - Equação:

$$var(x) = max\, x_i - min\, x_i \tag{5}$$

- *Signal Magnitude Area* (SMA)
  - Descrição: medida estatística da magnitude de uma quantidade variável. Total de atributos em uma janela: 1 ($SMA(x, y, z)$)
  - Equação:

$$SMA(x, y, z) = \sum_{i=1}^{n}(|x_i| + |y_i| + |z_i|) \tag{6}$$

- Assimetria
  - Descrição: grau de afastamento de uma distribuição da unidade de assimetria. Total de atributos em uma janela: 3 ($S(x)$, $S(y)$, e $S(z)$)
  - Equação:

$$S(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2}\right)^3} \tag{7}$$

- Curtose
  - Descrição: grau de achatamento da distribuição dos dados da janela. Total de atributos em uma janela: 3 ($k(x)$, $k(y)$, e $k(z)$)
  - Equação:

$$k(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2\right)^2} \tag{8}$$

- Correlação
  - Descrição: medida da relação entre dois eixos do acelerômetro. Total de atributos em uma janela: 3 ($\rho(x, y)$, $\rho(x, z)$, e $\rho(y, z)$)
  - Equação:

$$\rho(x, y) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\left(\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\right)\left(\sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}\right)} \tag{9}$$

- Excentricidade
  - Descrição: calcula-se três autovetores da matriz de covariância. A magnitude dos autovalores indicam a variação dos dados. Total de atributos em uma janela: 3 (λ_1, λ_2, e λ_3)
  - Equação:

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix} \tag{10}$$

$$cov(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x) * (y_i - \mu_y) \tag{11}$$

    em que $*$ denota o conjugado complexo.

- Atributos no domínio da frequência
  - Energia média
    - Descrição: somatório dos coeficientes da FFT. Total de atributos em uma janela: 3 ($E(X)$, $E(Y)$, e $E(Z)$)
    - Equação:

$$E(X) = \sum_{k=1}^{m} X(k) \tag{12}$$

  - Entropia espectral
    - Descrição: mensura a irregularidade, complexidade ou a quantidade de desordem do sinal. Total de atributos em uma janela: 3 ($H(X)$, $H(Y)$, e $H(Z)$)
    - Equação:

$$H(X) = -\sum_{k=1}^{m} X(k)\, log(X(k)) \tag{13}$$

  - Magnitude máxima
    - Descrição: coeficiente com máxima magnitude. Total de atributos em uma janela: 3 ($M(X)$, $M(Y)$, e $M(Z)$)
    - Equação:

$$M(X) = max(|X(k_i)|) \tag{14}$$

Os atributos são normalizados conforme descrito na seção 2.1.4. A RNA proposta é do tipo *feedforward*, com uma camada oculta. O número de neurônios da camada de entrada representa a quantidade de atributos considerados no treinamento. A camada

oculta possui a mesma quantidade de neurônios que a camada de entrada e o número de neurônios na camada de saída foi fixado em 7, que é a quantidade de atividades classificadas.

Ressalta-se que, para cada atributo selecionado, adicionam-se três novas entradas na RNA que representam as características dos eixos X, Y e Z do atributo acrescentado. Isso só não ocorre com a SMA, pois os dados de todos os eixos do acelerômetro são equacionados em um único valor, logo apenas uma entrada é adicionada à rede.

O treinamento da RNA, realizado pelo algoritmo *backpropagation*, utiliza 70% dos exemplos totalizando 882, ou seja, 126 de cada atividade. Os outros 30%, isto é, 54 exemplos de cada atividade, sendo 378 ao todo, foram empregados na avaliação do sistema que se baseou na técnica de validação simples. A separação dos exemplos de treinamento dos exemplos de validação é feita de forma aleatória.

Três conjuntos de atributos foram verificados: 1°) média, desvio padrão, assimetria, curtose e excentricidade, que alcançaram 95,1% de precisão; 2°) todos os 34 atributos descritos anteriormente, os quais garantiram 92,8% de acerto; 3°) utilizou-se a técnica de seleção de atributos PCA, obtendo-se 95,7% de acerto considerando 22 atributos. Dessa forma, o sistema oferece boa precisão nas classificações *offline*.

3 DESENVOLVIMENTO DO SIRAH NO PROCESSADOR NIOS II

A versão do SIRAH apresentada na seção 2.2 opera de modo *offline*, ou seja, o processamento e a classificação não são imediatos, tornando necessária a transferência dos dados do cartão SD para um microcomputador, em razão do algoritmo ser executado no MATLAB.

Na abordagem *online*, os algoritmos de classificação utilizados em HAR fornecem os resultados em tempo real, ou em intervalos de tempos aceitáveis de acordo com a aplicação, o que é necessário no monitoramento remoto. Diante dessa necessidade, o presente trabalho concentrou-se em desenvolver a portabilidade do SIRAH para operar em um sistema embarcado, capaz de realizar classificações *online*.

Devido à experiências anteriores, constatou-se a compatibilidade para desenvolver o sistema proposto por Durango (2017) em um FPGA, pois há placas de desenvolvimento que, além de implementarem fisicamente circuitos modelados em linguagens de descrição de hardware, também possibilitam a implementação de processadores que executam aplicações elaboradas em linguagens de alto nível, como C e C++.

Optou-se pela codificação do SIRAH em linguagem C, já que várias plataformas suportam essa linguagem, possibilitando a comparação e a análise dos resultados gerados por diferentes dispositivos. A placa de desenvolvimento utilizada no projeto foi a DE2-115, da Altera, onde foi configurado o processador Nios II, que executa o algoritmo codificado em C.

3.1 CODIFICAÇÃO EM LINGUAGEM C

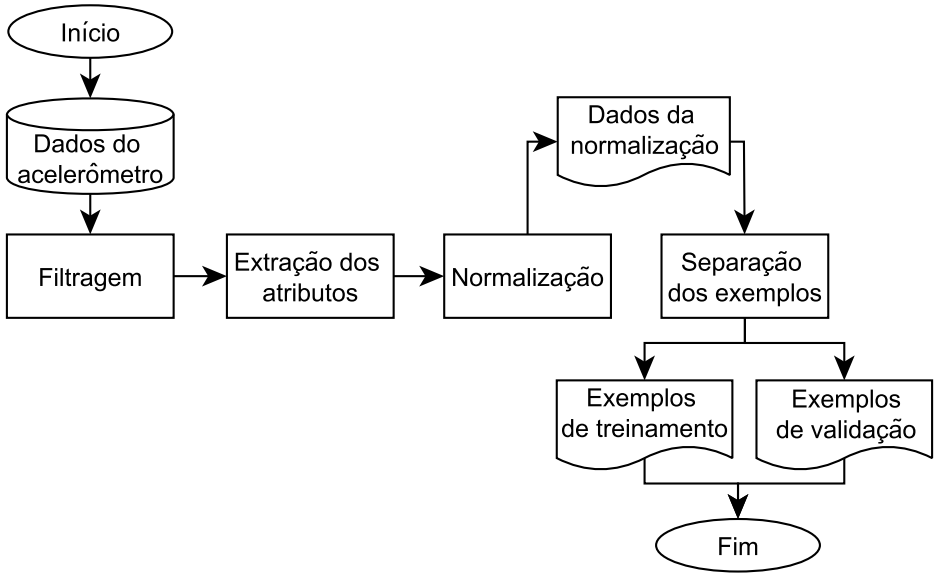
Inicialmente, o algoritmo utilizado pelo SIRAH foi implementado na linguagem do próprio MATLAB. Neste trabalho, outra versão do SIRAH foi codificada em linguagem C, seguindo as características originais com o intuito de obter a mesma exatidão da classificação *offline*. A codificação em C foi desenvolvida no compilador GCC e posteriormente utilizou-se o ambiente *Nios II Embedded Design Suite*, que configura o

software para ser executado no processador Nios II.

A programação foi realizada em duas etapas. Na primeira fase efetuou-se a implementação das funções que processam os dados gerados pelo acelerômetro, que foram validadas pela geração dos mesmos valores obtidos via MATLAB. Por fim, codificou-se a estrutura da RNA, incluindo o treinamento e a validação dos resultados.

Representa-se, no fluxograma apresentado na Figura 12, as etapas do pré-processamento, que geram os exemplos utilizados no treinamento e na validação da RNA.

Figura 12 – Fluxograma do pré-processamento.



Fonte: Elaboração do próprio autor.

Os dados originais do acelerômetro foram organizados em 7 arquivos de textos separados, um para cada atividade monitorada, que são carregados na memória do programa. Na filtragem, aplica-se um filtro de média móvel calculado por:

$$y[i] = \frac{1}{M} \sum_{j=-(M-1)/2}^{(M-1)/2} x[i+j], \quad 1 < i < n \tag{15}$$

sendo M o número de coeficientes para calcular a média, $y[\ ]$ e $x[\ ]$ são respectivamente os sinais de saída e de entrada e n é a quantidade de amostras pertencentes a $x[\ ]$. No SIRAH, utilizou-se um filtro com 3 coeficientes, e como o intervalo do cálculo das médias é simétrico, o filtro é aplicado da segunda amostra até a penúltima. Os filtros de média móvel são de fácil compreensão e, quando aplicados, eficientemente reduzem a taxa de ruídos aleatórios ocorridos durante a amostragem (SMITH, 1999).

Os atributos utilizados nessa codificação foram os mesmos do primeiro teste feito por

Durango (2017): média, desvio padrão, assimetria, curtose e excentricidade. Para obter cálculos mais eficientes, implementou-se uma função que extrai todos os atributos dos sinais filtrados simultaneamente. Como complemento, foram codificadas cinco funções separadas, uma para calcular cada atributo, o que pode facilitar possíveis mudanças, como a escolha de outras combinações de atributos.

A normalização dos atributos não é aplicada em todos os valores simultaneamente, mas sim em grupos menores, constituídos por um único atributo de todas as atividades de um eixo do acelerômetro. Por exemplo, normaliza-se o atributo média somente do eixo x das 7 atividades, que corresponde aos valores da terceira coluna da Tabela 4.

No intervalo normalizado, calcula-se a média e o desvio padrão, que são adotados na transformada *escore* padrão. Todos esses valores são armazenados em um arquivo de texto, pois são utilizados na classificação *online*.

Tabela 4 – Representação parcial dos atributos normalizados.

Atividade	**Num. do exemplo**	**Média**			**Desvio Padrão**			**...**	**Excentricidade**		
		x	**y**	**z**	**x**	**y**	**z**	**...**	**x**	**y**	**z**
Em pé	1	-0,22	-0,62	-0,33	-0,73	-0,72	-1,08	...	-0,61	-0,68	-0,56
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
	180	-1,86	-0,58	-0,40	-0,19	-0,71	-1,01	...	-0,60	-0,67	-0,55
Deitado	181	-0,97	2,32	2,02	-0,27	-0,46	-0,98	...	-0,59	-0,64	-0,53
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
	360	-0,94	2,50	2,01	-0,78	-0,71	-0,52	...	-0,61	-0,68	-0,54
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
Levantar	1080	-0,73	-0,33	-0,60	-0,22	-0,23	1,16	...	-0,49	-0,08	-0,22
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	⋮	⋮
	1280	2,32	-0,09	-0,87	0,04	-0,34	0,32	...	-0,51	-0,50	-0,39

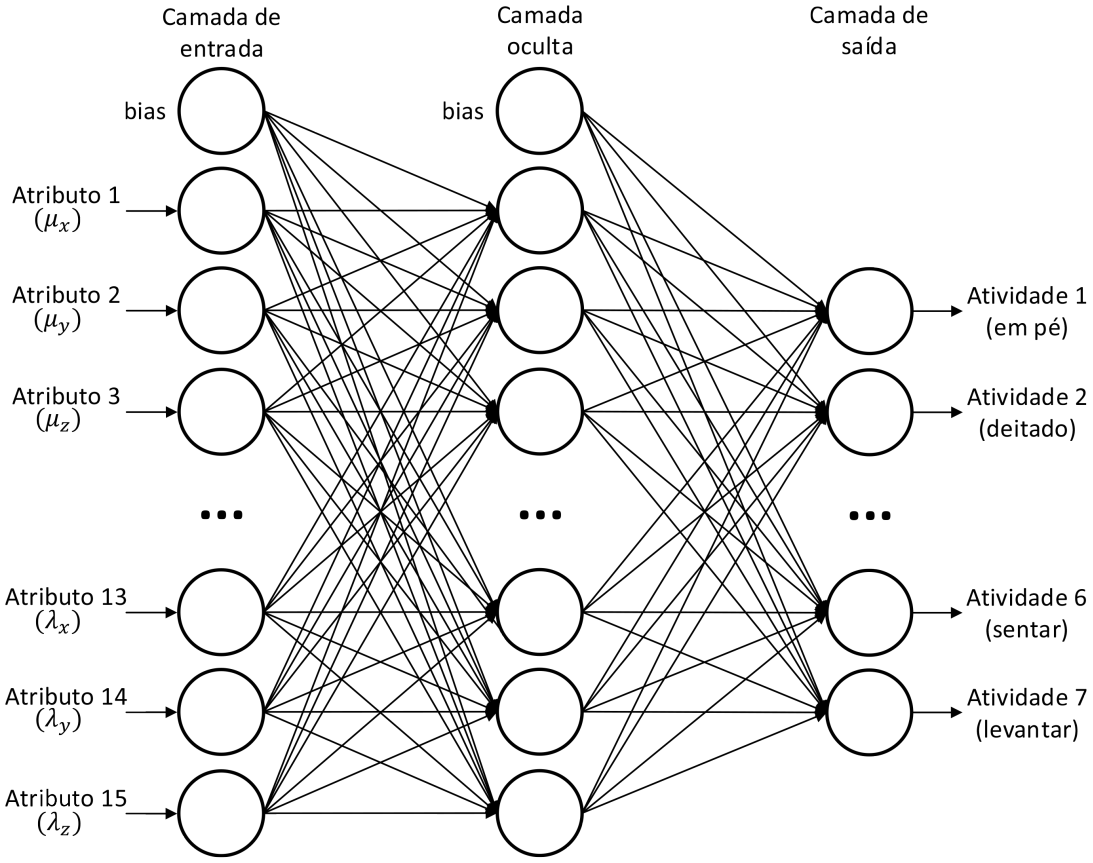
Fonte: Elaborado pelo próprio autor.

A Tabela 4 é um esboço para facilitar a visualização dos dados obtidos após a normalização dos atributos. Cada linha corresponde a um dos 1280 exemplos que, individualmente, são formados pelos atributos dispostos nas colunas.

Na última etapa do pré-processamento, os exemplos são separados gerando-se uma ordem aleatória entre 1 e 180, sendo os primeiros 126 (70% dos 180 exemplos de cada atividade) alocados para o treinamento e os 54 restantes para a validação dos resultados.

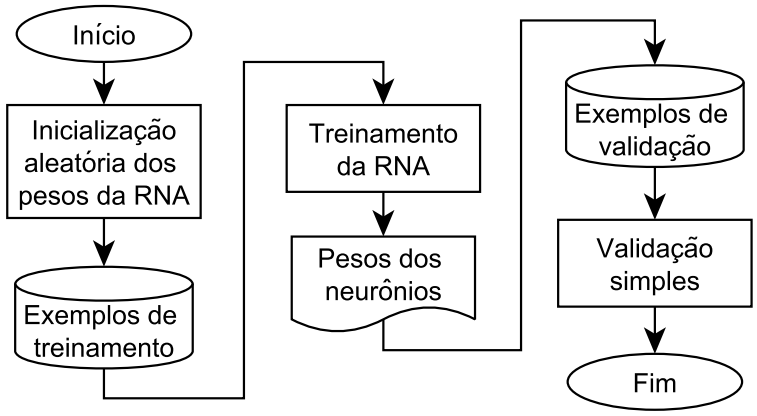
A arquitetura da RNA, apresentada na Figura 13, é a mesma que foi implementada no MATLAB. Nessa configuração, a RNA contém 15 neurônios na camada de entrada, um para cada atributo considerado. Representa-se, na Figura 14, as etapas presentes no processamento da RNA e na validação do sistema.

Figura 13 – Arquitetura da RNA implementada.



Fonte: Elaboração do próprio autor.

Figura 14 – Fluxograma da RNA e da avaliação do sistema.



Fonte: Elaboração do próprio autor.

A inicialização dos pesos dos neurônios da rede é feita gerando-se números aleatórios

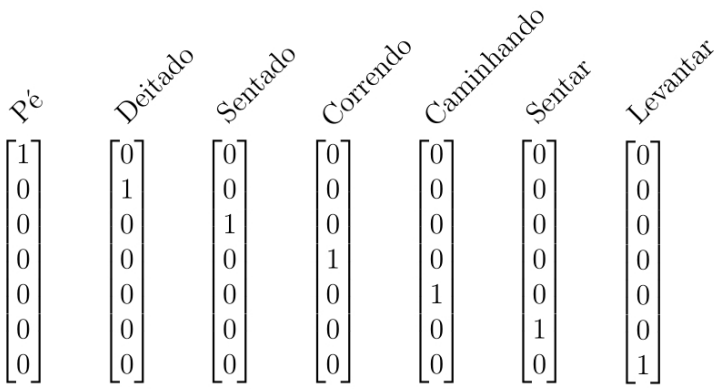
entre 0 e 1. Os exemplos de treinamento que foram gravados em disco são lidos e utilizados no treinamento da RNA.

Durante o treinamento, grande parte dos cálculos realizados no MATLAB são vetorizados, isto é, as funções recebem matrizes e vetores como parâmetros e automaticamente efetuam as operações. Na versão em linguagem C, essas funções foram implementadas considerando-se todas as dimensões das estruturas que armazenam os dados existentes. No final do treinamento, os pesos atualizados são gravados em um arquivo de texto, pois são utilizados na classificação *online*.

Para verificar o desempenho da rede, os exemplos de validação são carregados e realiza-se a validação simples. Neste processo, a saída da RNA calculada é comparada com a saída desejada, isto é, as respostas obtidas na classificação são comparadas aos rótulos reais de cada exemplo, e ao final é informada a porcentagem de acerto obtido.

Os padrões de saída da RNA, isto é, os valores das saídas desejadas, estão representados na Figura 15, consistido em vetores de 7 índices preenchidos com o dígito "0", exceto na posição que identifica a atividade em questão que contém o valor "1". O resultado de uma classificação é obtido comparando-se os valores dos neurônios da camada de saída da RNA, sendo o número da atividade identificada o índice neurônio que possui a maior saída calculada.

Figura 15 – Padrões de saída da RNA proposta.



Fonte: Elaboração do próprio autor.

3.2 CONFIGURAÇÃO DO PROCESSADOR NIOS II

A configuração do processador Nios II foi efetuada com base em um manual escrito por Dócusse (2014) que exemplificou o desenvolvimento de um sistema, descrito em Verilog, para executar aplicações em C++ em uma placa DE2-115 da Altera. No presente trabalho,

utilizou-se a linguagem de descrição de hardware VHDL para interligar o componente do processador Nios II com as demais interfaces existentes.

O Nios II é um *core* de propriedade intelectual, ou *Intellectual Property* (IP) *core*, da Altera, que implementa um processador configurável baseado na arquitetura RISC (*Reduced Instruction Set Computer*), nas células lógicas de FPGAs da Altera que suportam essa tecnologia (CORPORATION, 2016a). Esses blocos de IP são implementações em hardware, projetados e otimizados para um produto específico, como controladores de memória e até mesmo processadores, como o Nios II (CORPORATION, 2017).

A configuração do hardware é feita no SOPC *Builder*. Além do processador, estão disponíveis periféricos de propriedade intelectual da Altera que desempenham funções específicas, sendo anexadas ao projeto de acordo com a necessidade. A codificação do software é efetuada no *Nios II Embedded Design Suite*, que automaticamente gera a estrutura interpretada pelo hardware configurado.

Nesse trabalho, utilizou-se a versão 12.0 *Web Edition* do software Quartus II, que pode ser obtido gratuitamente no *site* da Altera, assim como o SOPC *Builder* e o *Nios II Embedded Design Suite*.

Representam-se na Figura 16 os componentes do sistema embarcado projetado no SOPC *Builder*. O *cpu_0* é o processador Nios II/f, versão que oferece maior taxa de processamento. O componente *jtag_uart_0* implementa uma comunicação serial entre um microcomputador e a placa DE2-115, possibilitando, por exemplo, a gravação do sistema descrito no FPGA e também efetuar depurações do código.

Figura 16 – Processador e periféricos utilizados no projeto.

Name	Description
⊟ **cpu_0**	Nios II Processor
instruction_master	Avalon Memory Mapped Master
data_master	Avalon Memory Mapped Master
jtag_debug_module	Avalon Memory Mapped Slave
⊟ **jtag_uart_0**	JTAG UART
avalon_jtag_slave	Avalon Memory Mapped Slave

Name	Description
⊟ **sdram_0**	SDRAM Controller
s1	Avalon Memory Mapped Slave
⊟ **uart_0**	UART (RS-232 Serial Port)
s1	Avalon Memory Mapped Slave
⊟ **performance_counter_0**	Performance Counter Unit
control_slave	Avalon Memory Mapped Slave

Fonte: Elaboração do próprio autor.

O *sdram_0* é o controlador de memória principal, que utiliza a memória SDRAM (*Synchronous Dynamic Random Access Memory*) da placa de desenvolvimento como memória principal para o Nios II. A comunicação serial RS-232 é feita por meio do componente *uart_0* e utilizou-se o *performance_counter_0* para verificar a quantidade de ciclos de *clock* consumidos na classificação de uma janela. Nos experimentos que utilizaram o cartão SD para transferir os dados do acelerômetro para o FPGA, o

componente *uart_0* foi substituído pela interface de comunicação com o cartão SD.

Com os componentes adicionados, fez-se a ligação do sistema com as interfaces de entrada e saída da placa de desenvolvimento, como o sinal de *clock* que é fornecido por um cristal externo ao Cyclone® IV EP4CE115F29C8, modelo do FPGA presente na placa DE2-115. Na compilação do projeto no Quartus II, geram-se informações sobre a quantidade de recursos utilizados na descrição. Lista-se, na Tabela 11, o consumo dos componentes utilizados no desenvolvimento do hardware listado na Figura 16.

Tabela 5 – Porcentagem de uso dos componentes do FPGA consumidos pelo processador Nios II.

Componente	**Total**	**Utilizado**	**Porcentagem**
Elementos lógicos	114480	4660	4,07%
Funções combinacionais	114480	4206	3,67%
Registradores lógicos dedicados	114480	2737	2,39%
Pinos de entrada e saída	529	78	14,74%
Bits de memória	3981312	65664	1,645%

Fonte: Elaborado pelo próprio autor.

Devido ao baixo consumo de recursos físicos, é possível expandir o sistema adicionando-se mais periféricos ao processador e também incluir entidades modelas em linguagem de descrição de hardware, que para o presente projeto é o VHDL.

Na compilação, o Quartus II gera os arquivos que são gravados na placa utilizando-se um cabo USB ligado ao microcomputador. Na gravação padrão, o programa é carregado em uma memória volátil, isto, é se a alimentação for interrompida, a configuração será perdida. Há a possibilidade de realizar a gravação em uma memória estática que mantém a descrição armazenada mesmo após o sistema ser desligado.

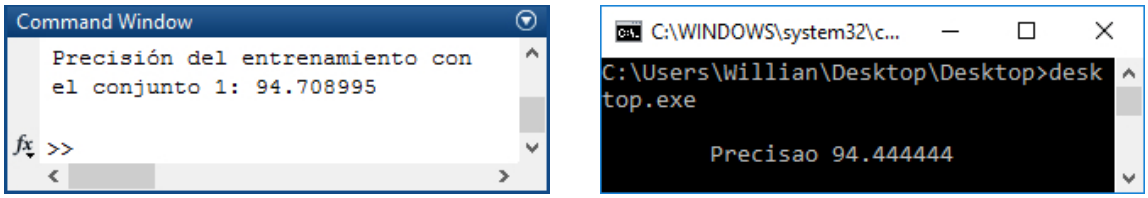
3.3 AVALIAÇÃO DOS SOFTWARES DESENVOLVIDOS

Para validar o desempenho da RNA, a codificação em C e o algoritmo do MATLAB foram executados com os mesmos parâmetros. No processador Nios II verificou-se o desempenho do treinamento da RNA e da classificação *online*. Nesta seção, apresentam-se os resultados gerados pelos softwares implementados.

3.3.1 Classificação *offline* do algoritmo em linguagem C

Visualiza-se nas Figuras 17(a) e 17(b) a exatidão de 94,71% e 94,44%, obtidas respectivamente pela classificação *offline* executada do MATLAB e da versão em linguagem C compilada no GCC. A mesma metodologia utilizada na avaliação do desempenho da versão implementada no MATLAB foi mantida na codificaçãop em C. Alguns atributos extraídos na codificação em C apresentaram pequenas diferenças em relação aos atributos calculados pelo MATLAB, o que causou a variação pouco significativa nos resultados.

Figura 17 – Comparação da exatidão das diferentes versões *offline* do SIRAH.



(a) execução no MATLAB

(b) codificação em C executada em um microcomputador

Fonte: Elaboração do próprio autor.

Essa avaliação envolveu todas as etapas da classificação *offline* efetuada pelo SIRAH, desde a filtragem dos dados originais do acelerômetro até o treinamento da RNA e a classificação das atividades. Validada a codificação em C, avaliou-se o comportamento do processador Nios II.

3.3.2 Treinamento da RNA no processador Nios II

A quantidade de dados utilizados no treinamento da RNA é relativamente alta, o que torna custosa a transferência das informações ao sistema configurado no FPGA. Para resolver essa questão, as comunicação disponíveis entre os dispositivos existentes foram estudadas.

Inicialmente, focou-se na interface de cartão SD existente na placa DE2-115, que transfere sequencialmente pacotes com tamanho de 1 byte a cada leitura. Testes realizados com arquivos de texto carregados com poucos bytes foram bem sucedidos. Contudo, a partir de um limite não foi mais possível transferir integralmente os dados para o software. Desse modo, para transferir todos os dados seria necessário uma grande quantidade de arquivos de texto, tornando essa alternativa inviável.

O segundo método estudado foi a transferência serial, especificamente utilizando o protocolo RS-232. A interface de comunicação UART da Avalon fornece a configuração necessária para se implementar uma comunicação serial entre o sistema embarcado e um dispositivo externo. A comunicação foi configurada entre o FPGA e um microcomputador utilizando o MATLAB, com um *baud rate* de 9600 bps (*bits per second*), sem paridade e com um bit de parada.

Para transferir as amostras de uma das 7 atividades do MATLAB para o FPGA levou em torno de 15 minutos, logo iria demorar por volta de 105 minutos para se transferir todos os dados necessários. Embora seja possível incrementar a velocidade da transferência escolhendo-se maiores taxas de *baud rate*, o tempo elevado consumido tornou essa possibilidade pouco atrativa.

A opção que ofereceu melhores condições foi a de incluir os dados do acelerômetro ao código-fonte. O sistema comportou a quantidade significativa de dados na memória principal, que possui 128 MB de capacidade.

Analisando-se a etapa do pré-processamento, constatou-se que o processador Nios II obteve os mesmos resultados que a versão executada no microcomputador, mas mostrou-se pouco otimizado em relação ao tempo. Para extrair os atributos de 180 exemplos, os quais representam apenas uma das 7 atividades estudadas, foram necessários 15 minutos. Para aprimorar o tempo de execução, diminuiu-se a quantidade de exemplos utilizados no treinamento, sendo necessário avaliar a eficiência da RNA treinada com uma quantidade de exemplos reduzida.

Em um microcomputador, a codificação em C foi executada por 2000 vezes para cada quantidade de exemplos analisada, aproximadamente na proporção de 70% para o treinamento e 30% para a validação. A exatidão das classificações decrementou à medida que menos exemplos de treinamento foram utilizados, como pode ser visualizado na Tabela 6. Com 180 exemplos de cada atividade (126 para o treinamento e 54 para a validação), a exatidão obtida foi de 94,23%, diminuindo para 92,78% empregando-se 22 modelos de cada atividade.

Tabela 6 – Avaliação do treinamento da RNA diminuindo-se a quantidade de exemplos.

Número de exemplos de treinamento	Número de exemplos de validação	Média da exatidão das classificações
126	54	94,23%
63	27	93,09%
32	13	93,05%
15	7	92,78%

Fonte: Elaborado pelo próprio autor.

Esses dados comprovam que a RNA não necessita ser submetida ao processo de aprendizado com quantidades excessivas de exemplos de treinamento, pois o desempenho da classificação permaneceu elevado. Essa é uma situação interessante se o treinamento for realizado em dispositivos com capacidade de processamento limitado.

Utilizando-se 30 exemplos de cada atividade, o treinamento da RNA no Nios II obteve a exatidão de 92,81% e consumiu aproximadamente 40 minutos. Apesar da implementação do processador ser feita em linguagem de descrição de hardware, a arquitetura que executa o software em C não foi gerada para aplicações específicas, tornando operações como a leitura e escrita na memória RAM e cálculos com ponto flutuante pouco otimizadas. Para obter um tempo de operação viável, manteve-se o treinamento da RNA no microcomputador e no sistema embarcado foi implementado apenas o algoritmo de classificação.

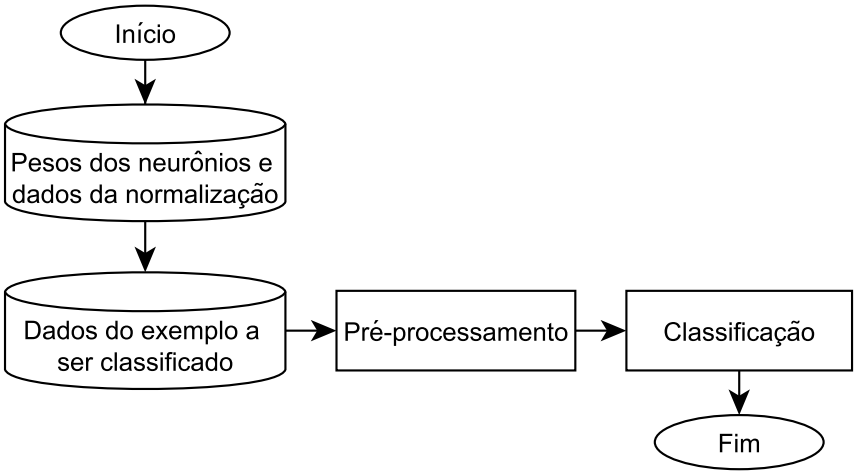
3.3.3 Classificação de uma janela

Os passos realizados para gerar a classificação de uma janela no Nios II estão reapresentados na Figura 18. As médias e os desvios padrões gerados na normalização dos atributos e os pesos dos neurônios obtidos no treinamento da rede foram incorporados ao código-fonte que executa a classificação. Nessa versão, as amostras do acelerômetro que representam a atividade classificada também foram incluídas ao código-fonte do algoritmo.

Após a leitura dos dados, realiza-se o pré-processamento nos dados brutos: filtragem, extração de atributos e normalização, que utilizaram os coeficientes calculados durante o aprendizado da RNA efetuado em um microcomputador. O treinamento efetuado para gerar os dados utilizados nesta avaliação utilizou 126 exemplos de cada atividade.

Os atributos calculados são processados pela RNA que classifica a janela e informa a

Figura 18 – Etapas para se classificar uma janela.



Fonte: Elaboração do próprio autor.

resposta. Realizaram-se 2000 iterações do algoritmo apresentado na Figura 18 no Nios II, que obteve 1900 acertos, totalizando 95% de exatidão.

Utilizando o *Performance Counter Core*, o tempo necessário para efetuar a classificação de uma janela foi medido. Em média, o algoritmo consumiu 673,57 milissegundos para ser finalizado.

Mesmo a resposta sendo obtida em um tempo aceitável, por se tratar de um sistema que executa instruções sequencialmente, a amostragem das acelerações não é contínua, pois é interrompida durante o processamento da classificação. Essa situação pode afetar a qualidade dos resultados, já que várias informações das atividades executadas são perdidas.

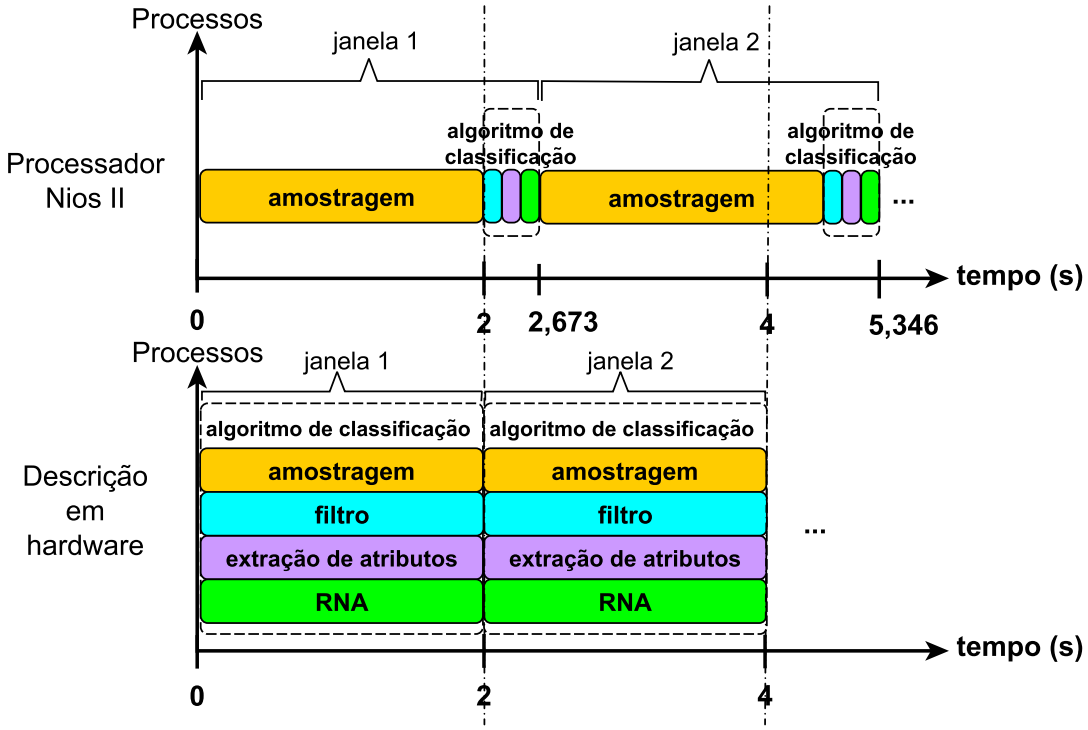
Para obter um sistema que opere em tempo real, o algoritmo de classificação de uma janela foi implementado fisicamente nas células lógicas do FPGA, utilizando-se a linguagem de descrição de hardware VHDL. Apresenta-se, no próximo capítulo, o desenvolvimento e as avaliações do sistema implementado em hardware.

4 IMPLEMENTAÇÃO DO SIRAH EM HARDWARE

A classificação *online* no processador Nios II foi validada, porém o processamento limitado estimulou o desenvolvimento do sistema em hardware.

A implementação do sistema em software efetuada no processador Nios II utilizou apenas processamento sequencial, característica comum das linguagens de programação de alto nível. Por outro lado, na descrição em VHDL apresentada nesse capítulo, implementou-se entidades que operam simultaneamente, ou seja, há processamento paralelo. Apresenta-se, na Figura 19, um diagrama para facilitar a compreensão da diferença de desempenho do processador Nios II e da descrição em hardware.

Figura 19 – Comparação das classificações executadas no processador Nios II e na descrição em hardware.



Fonte: Elaboração do próprio autor.

Com o processamento sequencial, o processador Nios II não amostra continuamente

as informações do acelerômetro, que são perdidas durante a execução do algoritmo de classificação. A rotulação da primeira janela é obtida em 2,673 segundos, sendo a soma dos 2 segundos gastos na amostragem das acelerações e dos 673 milissegundos consumidos pelo algoritmo de classificação. A cada janela processada, o atraso da classificação é acumulado, o que não ocorre na descrição em hardware, pois o algoritmo de classificação ocorre paralelamente desde o início da amostragem, garantindo resultados em tempo real.

4.1 DEFINIÇÃO DE PARÂMETROS

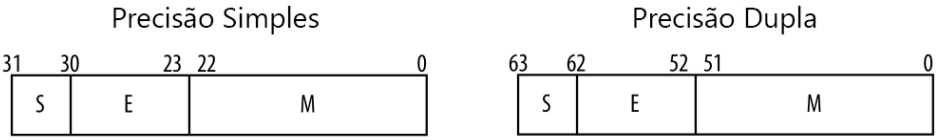
Durante o projeto do hardware, a codificação em C foi analisada com o propósito de determinar os parâmetros necessários para que a descrição implementada em VHDL tenha a mesma funcionalidade do algoritmo em C, como os cálculos de ponto flutuante, que representam quase todas as operações da classificação *online*.

Estudando-se a literatura constatou-se que a Altera disponibiliza *cores* de IP que executam operações com ponto flutuante, que estão listados em Corporation (2016b). Para utilizá-los, basta criar o componente no Quartus II com as configurações desejadas, adicionar a descrição gerada ao projeto e instanciar a entidade conforme a necessidade.

A codificação de ponto flutuante adotada nesses *cores* é o padrão IEEE-754 (IEEE, 2008). É necessário determinar a precisão utilizada na representação dos pontos flutuantes, que pode ser precisão simples, precisão dupla ou precisão simples estendida. Todas utilizam padrões binários com diferentes quantidades e configurações de bits.

Representa-se, na Figura 20, o formato da precisão simples e dupla, em que o bit mais significativo S contém o valor do sinal, E são os bits do expoente, e M são os bits da mantissa. Diferentes quantidades de bits são reservados para o expoente e para a mantissa em cada precisão, sendo que na precisão simples estendida essas parcelas não possuem tamanhos fixos, pois podem ser definidas pelo usuário. Porém, devem ser definidos respeitando os limites impostos (CORPORATION, 2016b).

Figura 20 – Precisões de ponto flutuante padrão IEEE-754.



Fonte: Adaptado de (CORPORATION, 2016b).

Nas avaliações realizadas para se determinar qual é a precisão adequada na implementação do SIRAH em hardware, também foram analisados dois grupos de

atributos aplicados no treinamento e na classificação da RNA. Além dos cinco atributos utilizados na implementação do processador Nios II, isto é, média, desvio padrão, assimetria, curtose e excentricidade, avaliou-se também uma nova configuração de características, em que a excentricidade foi desconsiderada, mantendo-se a média, o desvio padrão, a assimetria e a curtose.

Considerando a dimensão dos dados existentes no SIRAH, é necessário determinar as raízes de um polinômio de terceiro grau e resolver sistemas lineares de ordem 3 para se calcular a excentricidade. A Altera não disponibiliza gratuitamente os *cores* que efetuam essas operações. Dessa forma, verificou-se a exatidão da RNA sem o atributo excentricidade.

Para verificar a exatidão da RNA com as combinações de parâmetros considerados, ou seja, precisão de ponto flutuante simples e dupla, com e sem excentricidade, os experimentos foram executados na codificação em C em um microcomputador. Apresentam-se, na Tabela 7, os resultados das avaliações, que representam a média de 1000 execuções para cada combinação.

Tabela 7 – Comparação da exatidão da RNA com diferentes parâmetros.

	Precisão simples	**Precisão Dupla**
Com excentricidade	94,31%	94,32%
Sem excentricidade	94,22%	94,21%

Fonte: Elaborado pelo próprio autor.

Utilizando-se precisão dupla e mantendo-se a excentricidade, que é a configuração implementada no processador Nios II, a exatidão foi de 94,32%. No outro extremo, com precisão simples e sem a excentricidade, obteve-se 94,22% de acerto, evidenciando que esses últimos parâmetros causaram uma pequena diferença em comparação à configuração original proposta por Durango (2017).

Na codificação em linguagem C com precisão simples utilizou-se variáveis *float*, e com precisão dupla, *double*, sendo esses tipos de dados representados, respectivamente, no padrão IEEE-754 de precisão simples e dupla (MATTER, 2001). Portanto, foi possível estimar como a escolha da precisão dos pontos flutuantes utilizadas no VHDL influenciaria na exatidão das classificações efetuadas em hardware.

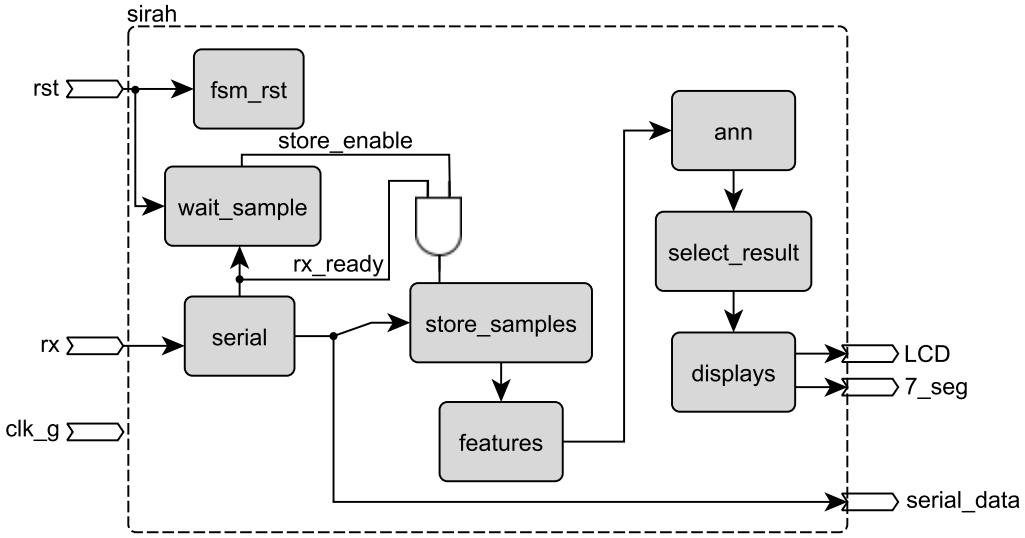
Dessa forma, definiu-se a precisão de ponto flutuante simples e o não uso do atributo excentricidade na descrição em hardware. Esses atributos não geraram diferenças significativas na exatidão das classificações, e também tornam a implementação em

hardware mais otimizada, pois a precisão simples consome metade do espaço físico utilizado pela precisão dupla e gasta menos ciclos de *clock* durante a execução de operações matemáticas.

4.2 DESENVOLVIMENTO E SIMULAÇÕES

Na descrição em hardware, o algoritmo aplicado na classificação *online* foi fragmentado em subsistemas, gerando entidades que comunicam entre si. Representa-se, na Figura 21, a entidade *sirah*, que é o topo da hierarquia definida, ou seja, contém todos os subsistemas criados.

Figura 21 – Diagrama de blocos da entidade *sirah*.



Fonte: Elaboração do próprio autor.

As interfaces de entrada e saída do sistema são:

- Entradas:
  - *clk_g*: *clock* gerado pelo cristal de 50 Mhz, que sincroniza o funcionamento de todas as entidades;
  - *rst*: *reset* acionado por uma chave H da placa de desenvolvimento. Quando está em nível lógico baixo, inibe o *clk_g* das entidades *wait_sample* e *fsm_rst*, fazendo com que o sistema permaneça em modo de espera;
  - *rx*: recebimento das acelerações transmitidas serialmente por um microcontrolador Atmega 328P, que gerencia a comunicação I^2C com o acelerômetro.
- Saídas:

- *LCD*: interfaceamento com o display de LCD, utilizado para imprimir o nome da atividade classificada;
- *7_seg*: impressão do número da atividade classificada, referente ao número do neurônio da camada de saída da RNA com maior valor. Adotou-se o dígito "0" para a saída do primeiro neurônio, que representa a atividade em pé, o dígito "1" para a segundo neurônio, e assim suscetivamente, até o dígito "6" associado ao sétimo neurônio, que simboliza a atividade levantar;
- *serial_data*: valor do byte recebido na entrada *rx*, mostrado nos LEDs verdes da placa DE2-115.

Enquanto o *rst* permanecer em nível lógico alto, o algoritmo é executado continuamente, sendo necessário, ao final de cada classificação, resetar algumas máquinas de estados finitos, ou FSMs (*Finite State Machine*), que devem estar preparadas para a próxima classificação.

O *reset* de sincronização é gerado pela entidade *fsm_rst*, cuja saída não foi representada na Figura 21. O sinal *clk_g*, que se conecta à todas entidades, também não foi apresentado, pois optou-se em destacar o fluxo do processamento de dados, que inicia com o recebimento das acelerações na entrada *rx* e termina com a apresesentação da classificação nos displays. Os únicos sinais de controle mantidos foram as conexões com a porta *and*, que coordena a sincronização das acelerações recebidas na comunicação serial.

A cada byte recebido, a entidade *serial* gera a *flag rx_ready*, informando que há uma amostra de aceleração disponível para leitura. Após a sincronização das leituras realizada pela entidade *wait_sample*, o sinal *store_enable*, que antes estava em nível lógico baixo, assume nível lógico alto, habilitando o armazenamento das acelerações no FPGA. Antes de serem armazenadas, as acelerações são filtradas aplicando-se o filtro de média móvel, apresentado na seção 3.1.

Com a amostragem de uma janela, os atributos são extraídos na entidade *features*. No módulo *ann*, a classificação é realizada aplicando os pesos obtidos no treinamento *offline*, e, finalmente, em *select_result* o número da classificação é obtido, que é o índice do neurônio da camada de saída com maior valor. A entidade *display* gerencia a comunicação com o display de LCD.

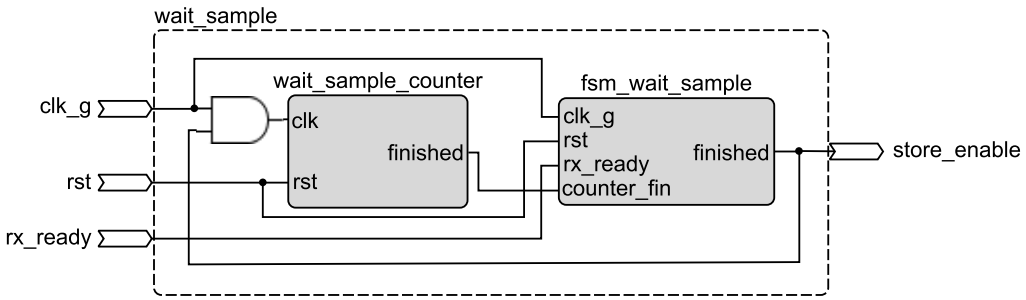
Nas próximas seções, os principais subsistemas da entidade *sirah* serão detalhados.

4.2.1 Entidade *wait_sample*

A cada amostragem, uma aceleração por eixo é transmitida do acelerômetro para o microcontrolador Atmega 328P, que envia os três valores sequencialmente ao FPGA a uma velocidade de 9600 bps pela comunicação serial. Devido à taxa de amostragem de 40 Hz definida no algoritmo original do SIRAH, esse processo se repete em períodos de 25 milissegundos. Cada aceleração possui o tamanho de 1 byte, que é convertida em um conversor ADC presente na placa do acelerômetro.

É necessário sincronizar a comunicação entre o FPGA e o microcontrolador, pois o primeiro valor recebido deve ser a aceleração do eixo X, garantido a integridade dos dados de uma janela. A sincronização é efetuada na entidade *wait_sample*, representada na Figura 22.

Figura 22 – Diagrama de blocos da entidade *wait_sample*.



Fonte: Elaboração do próprio autor.

A sincronização é realizada verificando-se o tempo entre o recebimento de duas acelerações. Se o intervalo for de aproximadamente 1,04 milissegundos, que é o tempo necessário para se transmitir um byte via comunicação serial com *baud rate* de 9600 bps, as acelerações pertencem à mesma amostragem. Por outro lado, se o intervalo for de 25 milissegundos, as duas acelerações pertencem à amostragens distintas.

No início da operação do sistema, já no primeiro byte recebido, o sinal *rx_ready* dispara um contador implementado na entidade *wait_sample_counter*, programado para elevar a saída *finished* de nível lógico baixo para nível logico alto após 4 milissegundos do seu acionamento. A contagem de tempo desse contador deve ser maior que 1,04 milissegundos e menor que 20 milissegundos, aproximadamente, possibilitando identificar se duas acelerações recebidos em sequência pertencem à amostragens distintas.

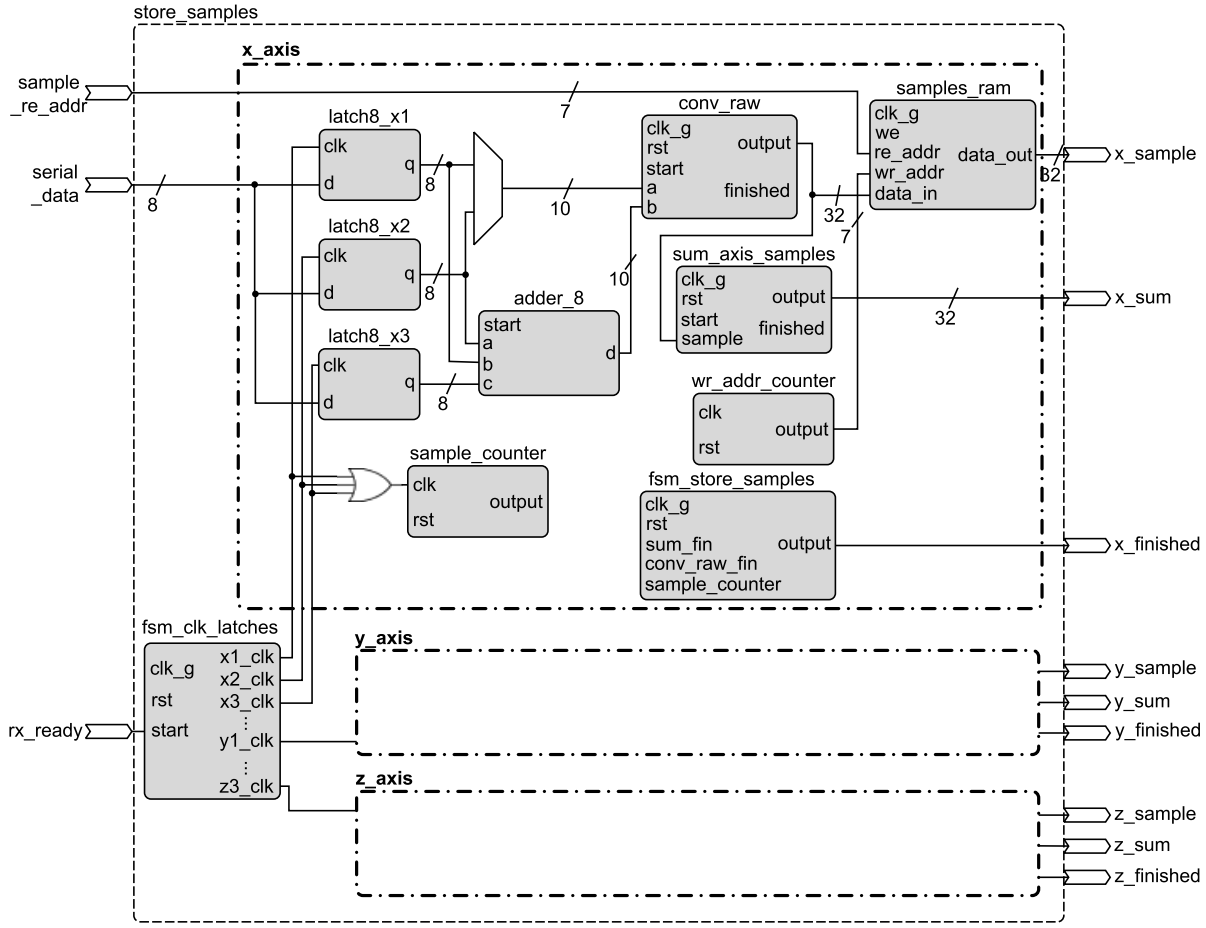
Quando ocorre a sincronização, a entidade *fsm_wait_sample* eleva o sinal *store_enable* para nível lógico alto, permitindo o armazenamento das próximas acelerações recebidas na comunicação serial.

4.2.2 Entidade *store_samples*

Essa entidade gerencia o recebimento e o armazenamento das acelerações de uma janela. Antes de serem gravadas em uma memória RAM (*Randon Acess Memory*), os valores recebidos em formato decimal são filtrados e convertidos para o padrão IEEE-754 utilizando-se o IP fornecido pela Altera especificamente para essa operação.

Representa-se, na Figura 23, os módulos pertencentes à entidade *store_samples*. Utilizou-se processamento paralelo nessa etapa, instanciando-se as entidades do subsistema *x_axis* também para os módulos *y_axis* e *z_axis*, que respectivamente filtram e armazenam as acelerações dos eixos X, Y e Z. A cada aceleração recebida, a *fsm_clk_latches* sincroniza o armazenamento das acelerações, em formato decimal, nos *latches* de entrada de cada eixo.

Figura 23 – Diagrama de blocos da entidade *store_samples*.

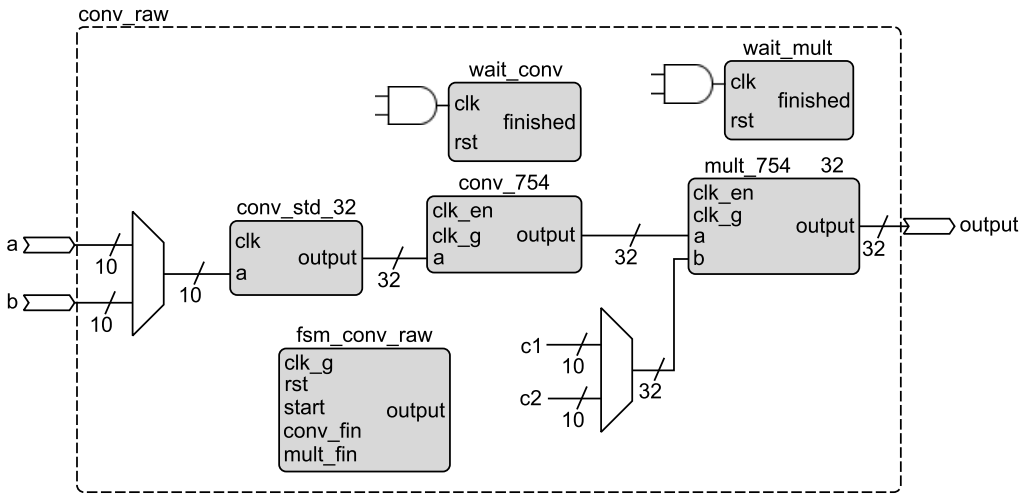


Fonte: Elaboração do próprio autor.

O funcionamento da entidade *x_axis* é detalhado a seguir. O mesmo processo ocorre internamente nas entidades *y_axis* e *z_axis*.

Armazenam-se, nas entidades *latch8_x1*, *latch8_x2* e *latch8_x1*, 3 acelerações do eixo X amostradas sequencialmente, que são utilizadas no cálculo da média aplicada na filtragem. A soma desses valores é calculado por *adder_8*, cujo resultado é convertido para o padrão IEEE-754 no módulo *conv_raw*, que está detalhado na Figura 24.

Figura 24 – Diagrama de blocos da entidade *conv_raw*.



Fonte: Elaboração do próprio autor.

A entrada *a* é destinada à primeira e à última aceleração de uma janela, pois não são alteradas na filtragem. A entrada *b* é o valor da soma calculada por *adder_8*. A entidade *conv_std_32* transforma as entradas *a* e *b* em valores inteiros de 32 bits, que são convertidos para o padrão IEEE-754 com precisão de ponto flutuante simples na entidade *conv_754*.

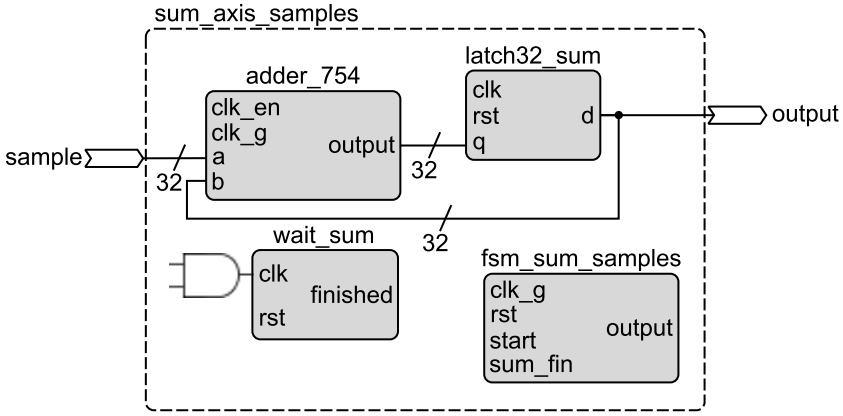
Os *cores* da Altera que realizam as operações com valores no padrão IEEE-754 consomem uma quantidade específica de ciclos de *clock* até gerarem os resultados desejados. Dessa forma, o algoritmo deve aguardar até que o processamento dessas operações sejam finalizados para prosseguir. Por exemplo, o *core* que realiza a conversão de um valor inteiro para o padrão IEEE-754, e vice-versa, consome 6 ciclos de *clock*, e a espera desse tempo é controlada na entidade *wait_conv*. Após os ciclos necessários, a *fsm_conv_raw* recebe um sinal indicando a conclusão da conversão.

O acelerômetro fornece o valor inteiro 64 para a aceleração da gravidade com intensidade de 1g. No algoritmo original do SIRAH, os dados amostrados são normalizados, ou seja, são divididos por 64, gerando valores unitários quando um eixo do acelerômetro está em repouso e no sentido da gravidade. No hardware, essa normalização é efetuada após a conversão das acelerações filtradas para o padrão IEEE-754 na entidade *mult_754*.

Com a finalidade de se otimizar o desenvolvimento da descrição em hardware, sempre que possível, substituiu-se as divisões por multiplicações pelo inverso do divisor, pois a divisão consome mais ciclos de *clock* do que a multiplicação. A constante *c1* é utilizada apenas para normalizar a primeira e a última amostra das acelerações de uma janela, pois não são filtradas. Já a constante *c2*, além de normalizar o resultado das somas das 3 últimas amostragens, também aplica o filtro nesse intervalo considerado, gerado uma aceleração normalizada e filtrada que é armazenada na sequência.

As acelerações normalizadas e filtradas são armazenadas na entidade *samples_ram*, que é a descrição de uma memória RAM com 80 posições de 32 bits cada. Na entidade *sum_axis_samples*, representada na Figura 25, calcula-se o somatório das acelerações normalizadas, que é utilizada no cálculo dos atributos. O resultado das somas calculadas pelo *adder_754* é acumulado no *latch32_sum*.

Figura 25 – Diagrama de blocos da entidade *sum_axis_samples*.



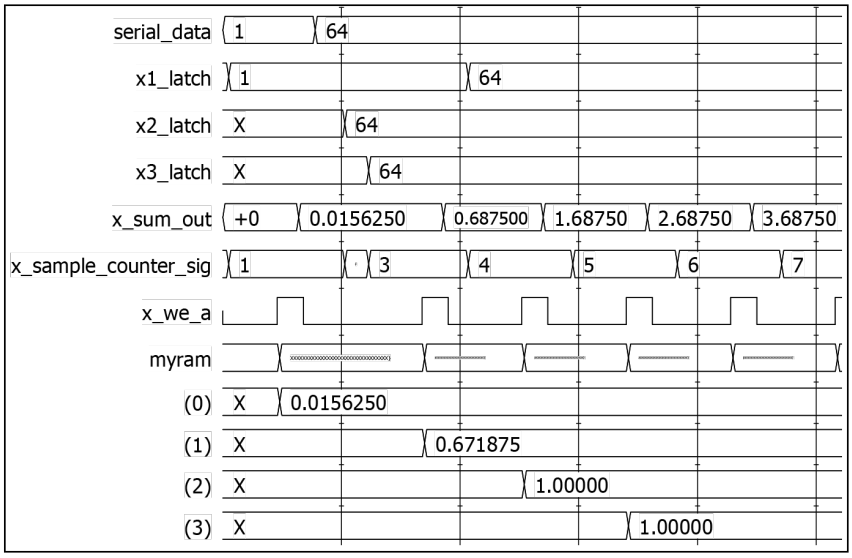
Fonte: Elaboração do próprio autor.

A simulação do funcionamento da entidade *store_samples*, efetuada no software ModelSim, está parcialmente apresentada na Figura 26. O primeiro dado recebido na entrada *serial_data* (valor "1") não é filtrado, mas apenas normalizado, isto é, multiplicado pelo inverso de 64, gerando o resultado "0,015625". Esse valor é gravado na posição 0 de *myram*, que é a instanciação da entidade *samples_ram* para armazenar as acelerações do eixo X. O sinal *x_sum_out* é a saída de *sum_axis_samples*, que no início possuía o valor nulo, acumulando o resultado da normalização da primeira amostra.

A partir da segunda amostra recebida em *serial_data*, manteve-se o valor "64" para facilitar a visualização dos resultados. Quando a segunda amostra é recebida, seu valor é apenas armazenado em *x2_latch*, pois será utilizado na aplicação do filtro. Na chegada da terceira aceleração, calcula-se a média das 3 últimas acelerações gravadas em *x1_latch*, *x2_latch* e *x3_latch*, que é normalizada e armazenada na posição 1 de *myram*. A partir

da posição 2 de *myram* todos os valores são “1”, pois a entrada de dados permaneceu constante, gerando o valor unitário após o cálculo da média e da normalização a cada 3 acelerações. Os dados armazenado em *myram* são somados e acumulados em *x_sum_out*, obtendo-se o somatório das acelerações do eixo X da janela considerada.

Figura 26 – Simulação da entidade *store_samples*.



Fonte: Elaboração do próprio autor.

4.2.3 Entidade *features*

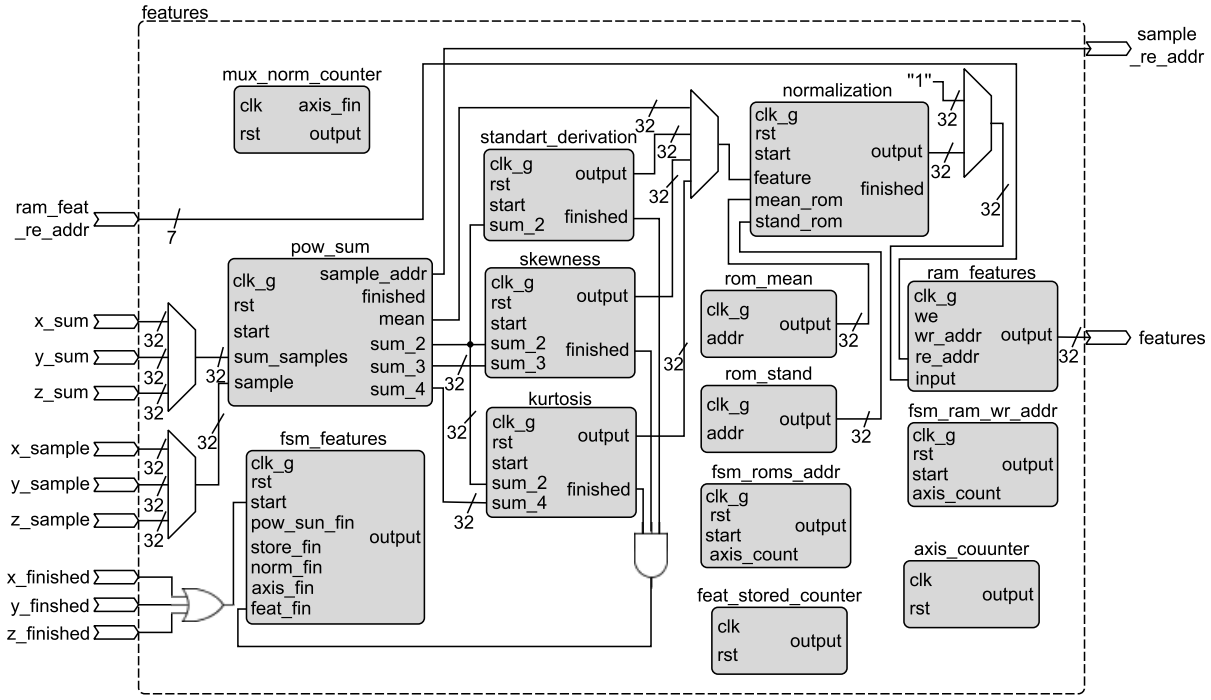
Os sinais *x_finished*, *y_finished*, e *z_finished*, que respectivamente indicam a finalização da amostragem das acelerações dos eixos X, Y e Z, controlam o início do cálculo dos atributos, que são processados na entidade *features*, representada na Figura 27.

Os atributos de cada eixo são calculados separadamente pelo mesmo hardware. Dessa forma, para gerar os atributos do eixo X, a entidade *features* aguarda a transição de borda de subida do sinal *x_finished*, e utiliza as acelerações filtradas do eixo X e o somatório dessas acelerações, cujos valores estão armazenados, respectivamente, nos sinais *x_sample* e *x_sum*. As etapas descritas a seguir são referentes somente ao processamento das informações do eixo X. O mesmo algoritmo é repetido mais duas vezes, considerando-se as devidas entradas no cálculo dos atributos dos eixos Y e Z.

O atributo média é obtido multiplicando-se o somatório armazenado em *x_sample* pelo inverso de 80, que é quantidade de acelerações de um eixo de uma janela. Essa operação é realizada no módulo *pow_sum*.

Para facilitar o entendimento da metodologia utilizada no cálculo dos demais atributos, as equações do desvio padrão, da assimetria e da curtose são novamente

Figura 27 – Diagrama de blocos da entidade *features*.



Fonte: Elaboração do próprio autor.

apresentadas na Tabela 8, sendo n o número de acelerações de um eixo da janela.

Tabela 8 – Equações dos atributos desvio padrão, assimetria e curtose.

Atributo	**Equação**
Desvio Padrão	$\sigma_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)^2}$
Assimetria	$S(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2}\right)^3}$
Curtose	$k(x) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)^2\right)^2}$

Fonte: Elaborado pelo próprio autor.

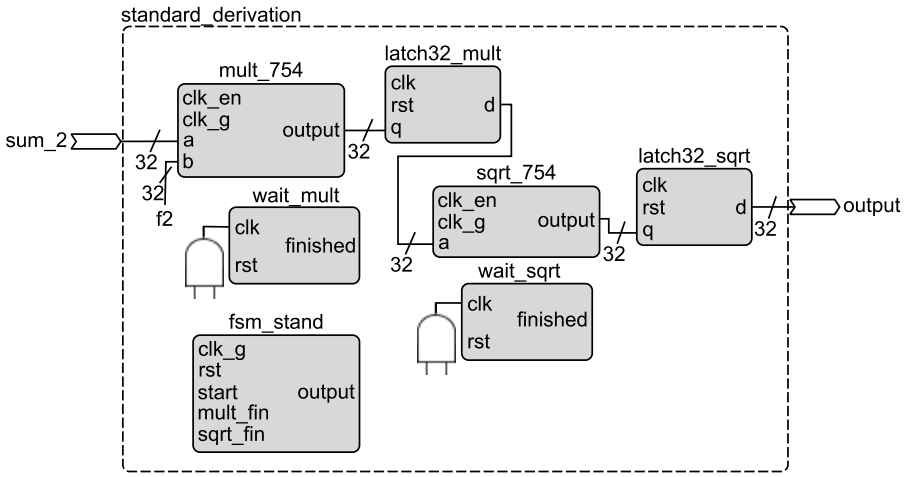
Nota-se que as equações aplicam diferentes potenciações do somatório da diferença de cada amostra com a média dessas acelerações calculada anteriormente. Com o objetivo de

tornar o cálculo dessas potenciações mais eficiente, as amostras da memória RAM definida no módulo *store_samples* são acessadas uma única vez pela entidade *pow_sum*, que calcula o somatório e as potenciações necessárias.

O valor do somatórios da diferença de cada amostra com a média é elevado ao quadrado, ao cubo e à quarta, e são respectivamente armazenados em *sum_2*, *sum_3* e *sum_4*. Esses sinais vão para as entidades que calculam, em processos paralelos, o desvio padrão (*standard_derivation*), a assimetria (*skewness*) e a curtose (*kurtosis*), de acordo com as equações apresentadas na Tabela 8.

A entidade *standard_derivation* está detalhada na Figura 28. A constante *f2* armazena o valor $1/(n-1)$, que é multiplicado com o sinal *sum_2* no módulo *mult_754*. Por fim, a entidade *sqrt_754*, que é um IP da Altere, extrai a raiz quadrada do resultado dessa multiplicação, obtendo-se o atributo desvio padrão que é armazenado no *latch32_sqrt*.

Figura 28 – Diagrama de blocos da entidade *standard_derivation*.

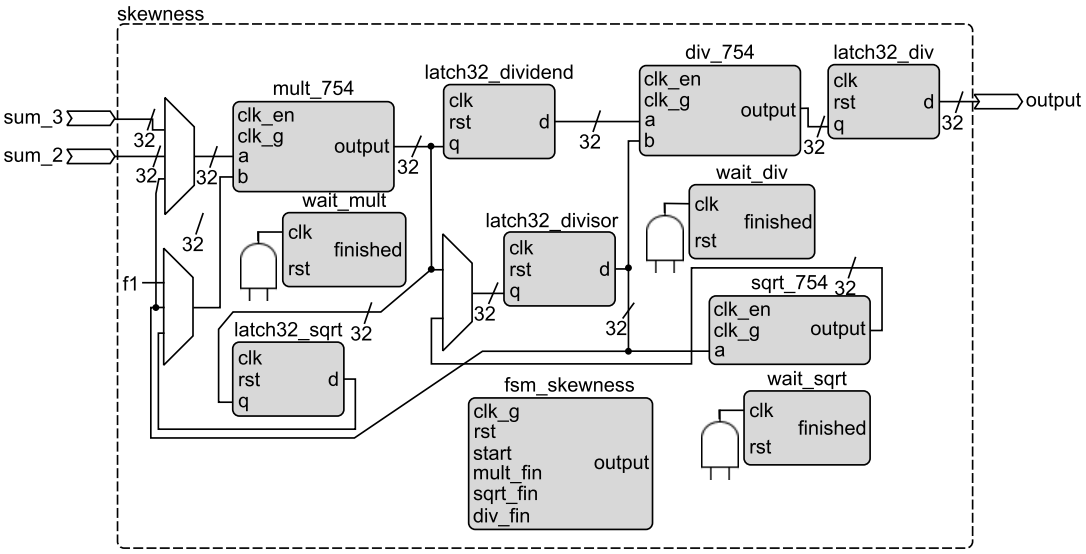


Fonte: Elaboração do próprio autor.

As entidades *skewness* e *kurtosis* estão respectivamente apresentadas nas Figuras 29 e 30, que foram implementadas seguindo os mesmos princípios, devido à similaridade no cálculo desses atributos.

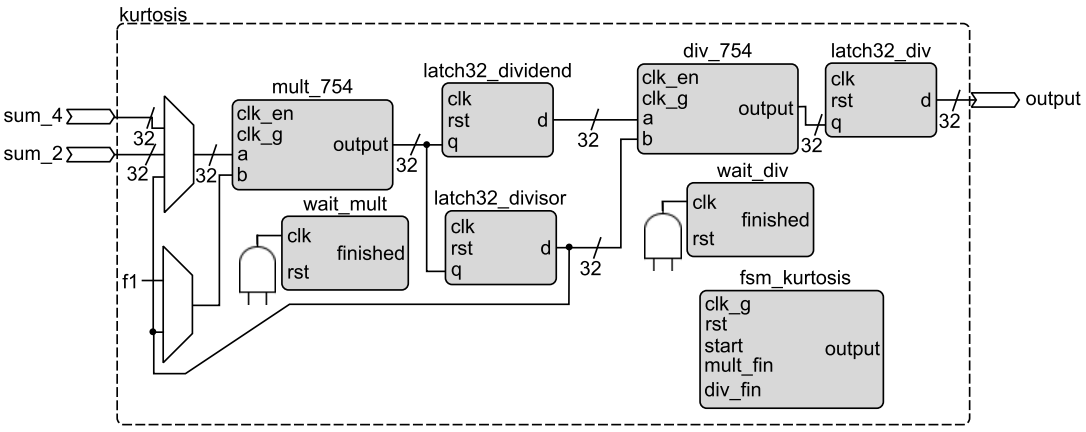
Nota-se que ambas equações são compreendidas por uma divisão e por multiplicações no numerador e no denominador. Em cada entidade o circuito gerado foi otimizado instanciando-se apenas um bloco de multiplicação, cujas entradas são selecionadas por multiplexadores. As multiplicações são efetuadas em sequência, e os resultados do numerador e do denominador são armazenados, respectivamente, em *latch32_dividend* e *latch32_divisor*. Na entidade *skewness*, antes de realizar a divisão, extrai-se a raiz quadrada do divisor, e na *kurtosis*, o módulo *div_754* calcula a divisão do *latch32_dividend* e *latch32_divisor* diretamente.

Figura 29 – Diagrama de blocos da entidade *skewness*.



Fonte: Elaboração do próprio autor.

Figura 30 – Diagrama de blocos da entidade *kurtosis*.



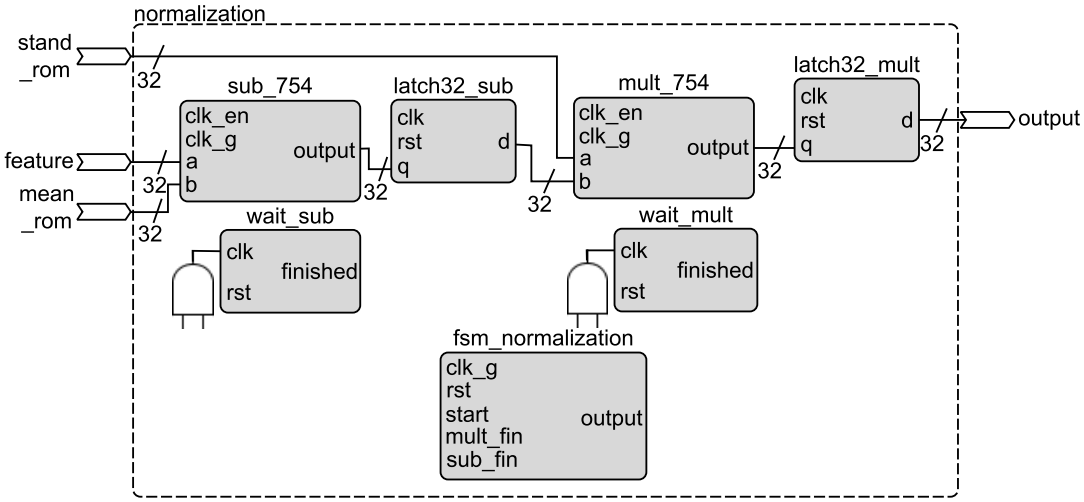
Fonte: Elaboração do próprio autor.

A normalização é aplicada somente após o término do cálculo dos 4 atributos. Apresenta-se, na Figura 31, os detalhes do módulo *normalization*, que normaliza os atributos como mencionado na seção 2.1.4.

As constantes da normalização obtidas no treinamento *offline* executado na codificação em C foram convertidas para o padrão IEEE-754 e armazenadas em duas memórias ROM (*Read Only Memory*), a *rom_mean* e *rom_stand*, que podem ser visualizadas na Figura 27, e respectivamente armazenam as constantes das médias e dos desvios padrão obtidas no treinamento da RNA. Os atributos normalizados são armazenados na *ram_features*, que é o vetor de entrada da RNA.

Nas simulações da entidade *features*, constatou-se que a descrição em hardware obteve resultados com exatidão similar a da codificação em C, como pode ser visualizado nas

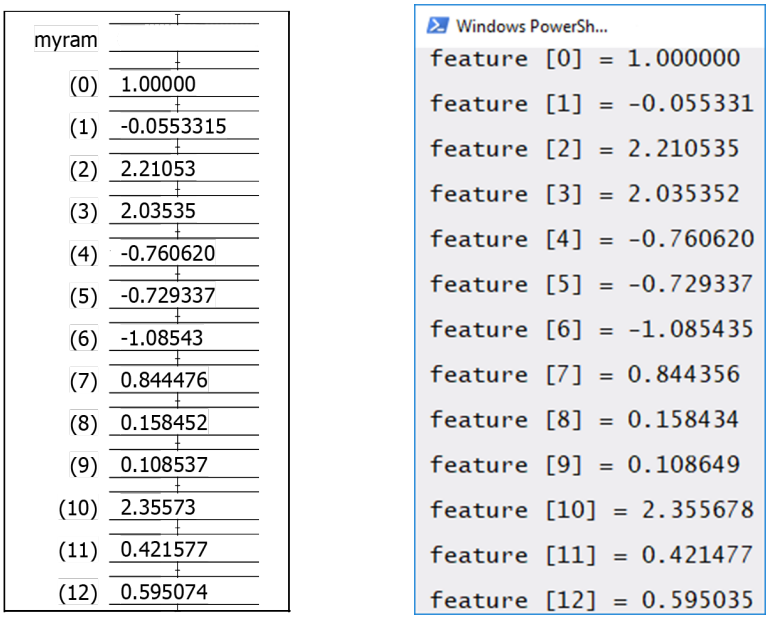
Figura 31 – Diagrama de blocos da entidade *normalization*.



Fonte: Elaboração do próprio autor.

Figuras 32(a) e 32(b). O componente *myram* é a memória RAM que contém os resultados dos atributos calculados em hardware. Na simulação no ModelSim, utilizou-se os mesmos dados que o algoritmo executado na codificação em C.

Figura 32 – Atributos gerados por diferentes codificações.



(a) simulação no ModelSim (b) codificação em C

Fonte: Elaboração do próprio autor.

Contudo, quando o sistema descrito em hardware foi gravado no FPGA, obteve-se resultados diferentes dos gerados na simulação, que estão apresentados na coluna "Execução 1" da Tabela 9. Nota-se que as médias e os desvios padrão obtidos nessa execução em hardware ficaram próximos dos valores esperados, mas a assimetria e

a curtose tiveram valores com diferenças maiores, diferindo em até 66% dos valores esperados.

Tabela 9 – Atributos calculados em diferentes configurações de hardware.

Atributo	**Execução 1**	**Execução 2**
Bias	1,0	1,0
Média (X)	-0,055331547	-0,055331547
Média (Y)	2,2102902	2,2102902
Média (Z)	2,0353525	2,0353525
Desvio padrão (X)	-0,7506848	-0,7605592
Desvio padrão (Y)	-0,72967035	-0,72939795
Desvio padrão (Z)	-1,0872083	-1,0854348
Assimetria (X)	2,012483	0,8444197
Assimetria (Y)	1,0375152	0,15843877
Assimetria(Z)	0,2560741	0,108620204
Curtose (X)	3,5622458	2,3556867
Curtose (Y)	0,34879953	0,42144766
Curtose (Z)	0,6822332	0,5950288

Fonte: Elaborado pelo próprio autor.

Analisando os resultados, notou-se que as maiores diferenças estão nos atributos que utilizam o somatório da diferença entre as acelerações e a média das acelerações de um eixo elevada ao cubo e à quarta. Ou seja, o erro inesperado ocorre nas operações de ponto flutuante com números pequenos, aproximadamente a partir da ordem de 10^{-7}.

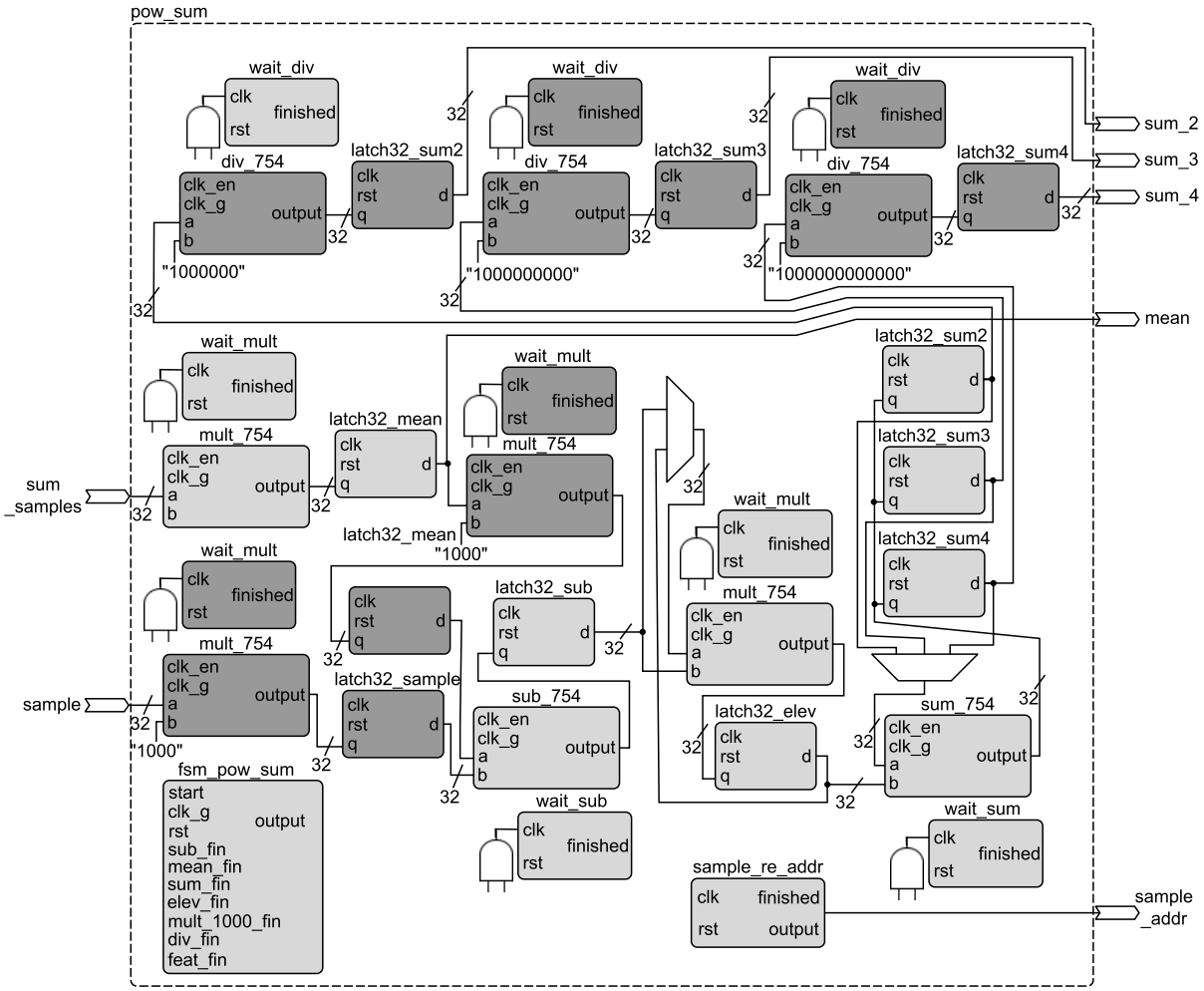
Para solucionar os erros imprevistos, na entidade *pow_sum* o valor da média e o resultado da diferença das acelerações com a média foram multiplicados por 1000. Essa multiplicação é aplicada com o intuito de deixar esses números maiores, minimizando o erro calculado durante os somatórios. Ao final dos cálculos, os valores dos somatórios das potências são divididos por potências de base 10, corretamente dimensionadas para cada caso, convertendo os resultados para a mesma grandeza da primeira implementação.

Na Figura 33, representa-se a entidade *pow_sum*. Os módulos em tom de cinza mais claro é a implementação que gerou os resultados errados. A solução foi adicionar os blocos em tom de cinza mais escuro, que realizam as multiplicações e divisões por números em base de 10 descritas anteriormente.

Após a adição desses componentes, o sistema foi novamente gravado no FPGA, que

gerou os atributos apresentados na coluna “Execução 2” da Tabela 9, garantido resultados corretos na saída da entidade *features*.

Figura 33 – Diagrama de blocos da entidade *pow_sum*.



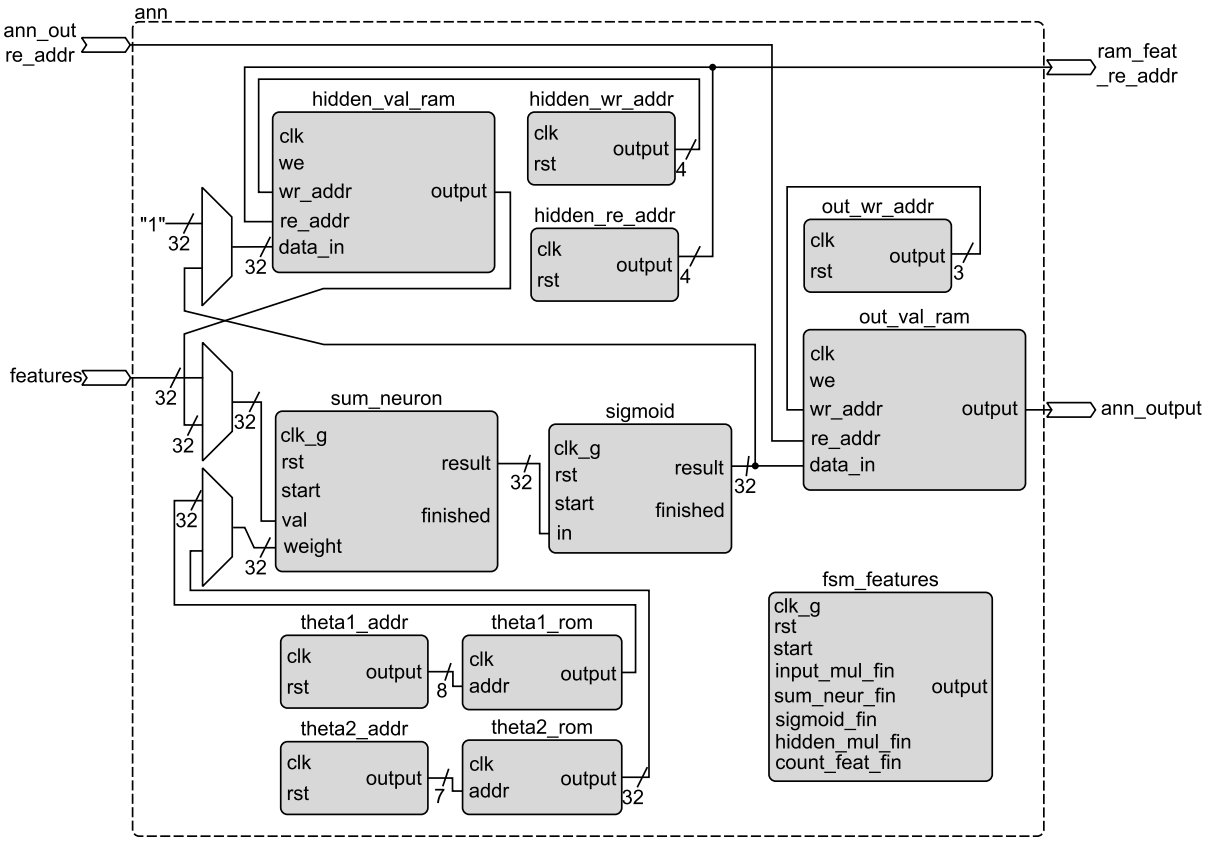
Fonte: Elaboração do próprio autor.

4.2.4 Entidade *ann*

A entidade *ann* é a implementação da RNA, e está representada na Figura 34. A entrada *features* é a saída da memória RAM que armazena os atributos calculados em *features*, que são os valores fornecidos à camada de entrada da RNA. Os pesos da camada de entrada e da camada oculta, obtidos no treinamento *offline* pela codificação em C, estão respectivamente armazenados nas memórias ROM *theta1_rom* e *theta2_rom*.

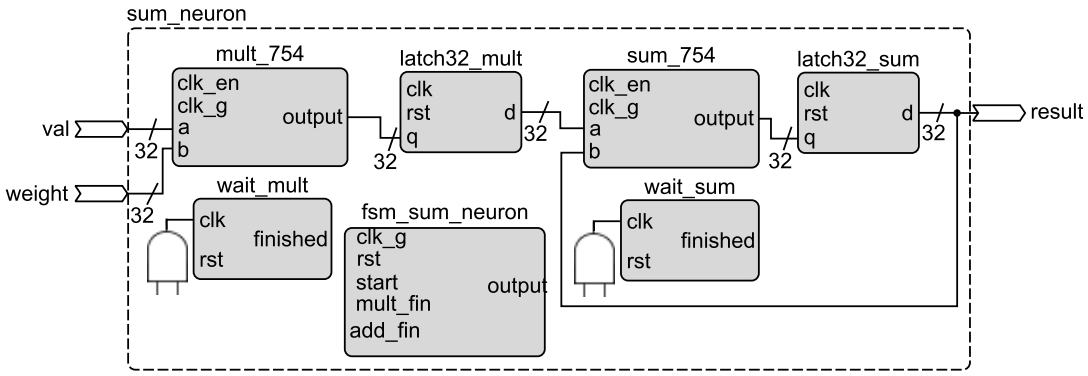
Para realizar o processamento dos dados, definiu-se apenas um neurônio, que é o módulo *sum_neuron* representado na Figura 35. Primeiramente, essa entidade calcula as combinações lineares entre os valores da entrada com os pesos de cada neurônio da camada de entrada.

Figura 34 – Diagrama de blocos da entidade *ann*.



Fonte: Elaboração do próprio autor.

Figura 35 – Diagrama de blocos da entidade *sum_neuron*.

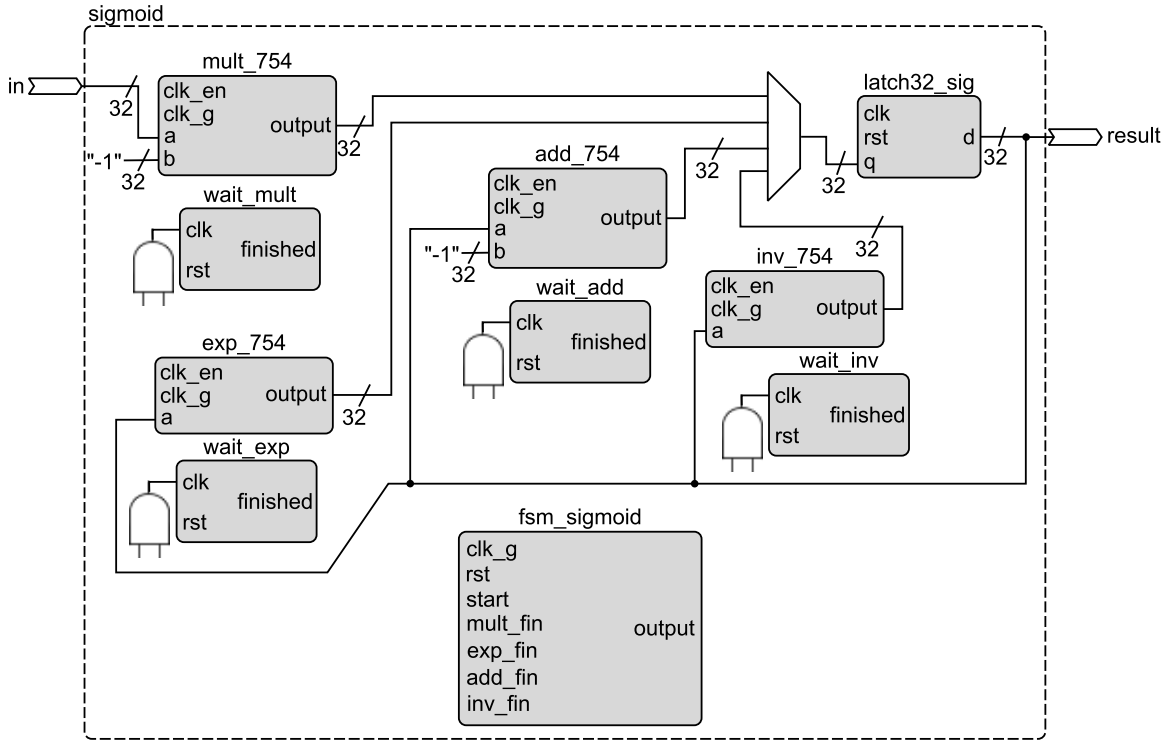


Fonte: Elaboração do próprio autor.

Os resultados das combinações lineares passam pela função de ativação sigmoide implementada no módulo *sigmoid*, representada na Figura 36, e são armazenados na memória RAM *hidden_val_ram*, que contém os valores de saída da camada oculta.

Na etapa subsequente são efetuadas as combinações lineares entre os valores de saída da camada oculta e os pesos dos neurônios dessa camada, e calculada a sigmoide de cada uma dessas combinações lineares. O mesmo hardware efetua essas operações, ou

Figura 36 – Diagrama de blocos da entidade *sigmoid*.



Fonte: Elaboração do próprio autor.

seja, as entidades *sum_neuron* e *sigmoid*. Contudo, os valores da camada de saída são armazenados na memória RAM *out_val_ram*.

Para validar os cálculos realizados pela RNA, comparou-se os resultados obtidos na codificação em C com os gerados em hardware, ambos apresentados na Tabela 10. Nessa avaliação, a entidade *ann* foi adicionada ao sistema que gerou os atributos apresentados na seção anterior, e o software em C é o mesmo que gerou os atributos apresentados na Figura 32(b).

Tabela 10 – Comparação dos valores da saída da RNA calculados em software e hardware.

Índice do neurônio de saída da RNA	**Saída gerada pela codificação em C**	**Saída gerada pelo hardware**
0	9,121523 x 10^{-5}	9,120778 x 10^{-5}
1	0,9459634	0,9459023
2	8,148517 x 10^{-2}	8,14852 x 10^{-2}
3	1,011978 x 10^{-5}	1,0118834 x 10^{-5}
4	4,308245 x 10^{-6}	4,3082487 x 10^{-6}
5	1,690275 x 10^{-11}	1,6902745 x 10^{-11}
6	1,114940 x 10^{-8}	1,1166731 x 10^{-8}

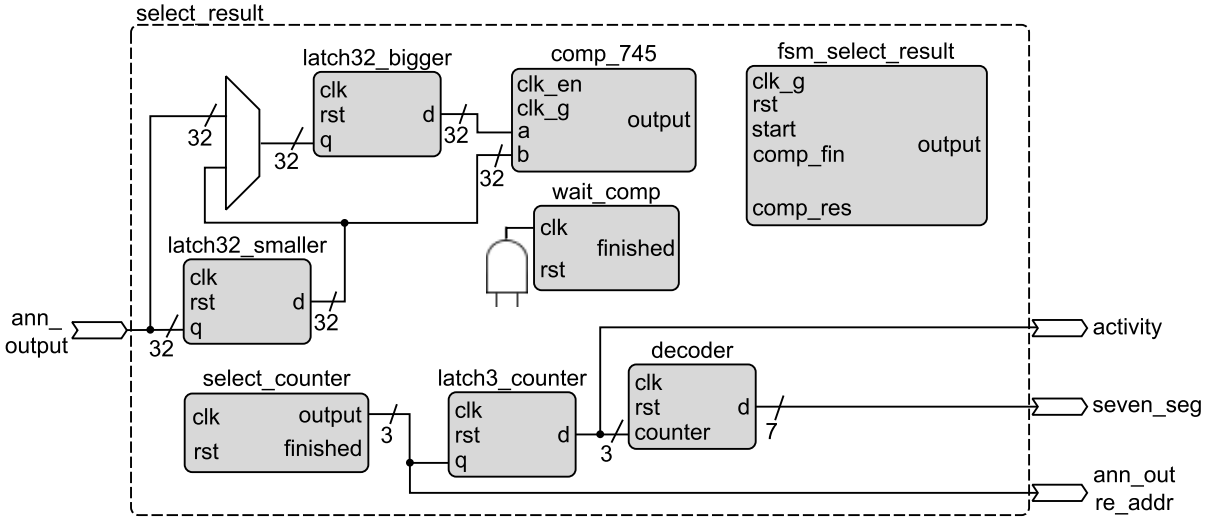
Fonte: Elaborado pelo próprio autor.

O resultado gerado pelo hardware teve exatidão próxima do resultado calculado em software, comprovando a integridade dos cálculos das combinações lineares e da aplicação da função de ativação sigmoide implementados em VHDL.

4.2.5 Entidade *select_result*

A última etapa da classificação é verificar a saída da RNA para determinar qual é a atividade realizada. Essa operação é efetuada pela entidade *select_result* que está representada na Figura 37.

Figura 37 – Diagrama de blocos da entidade *select_result*.



Fonte: Elaboração do próprio autor.

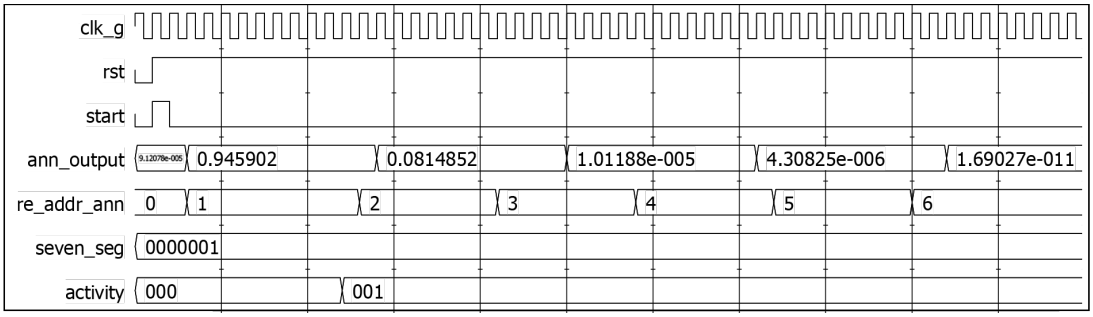
Pela entrada *ann_output*, o módulo *select_result* recebe sequencialmente os valores dos 7 neurônios da camada de saída da RNA. A cada leitura, os números são comparados, e o índice do neurônio com maior saída é armazenado no *latch3_counter*, que está ligado à saída *activity*.

No final das comparações, obtém-se o rótulo da classificação realizada pela RNA, que é armazenado em *latch3_counter*. A entidade *decoder* transforma o número para o padrão de escrita do display de 7 segmentos, onde é mostrado o número da atividade classificada. O sinal de saída *activity* vai para o módulo controlador do display de LCD, que imprime o nome da atividade classificada.

Apresenta-se, na Figura 38, a simulação da entidade *select_result* com os valores gerados pelo hardware mostrados na Tabela 10. O sinal *activity* é iniciado com o valor "000" em representação binária, que é o índice do primeiro neurônio. Quando o sinal *ann_output* recebe o valor do segundo neurônio, ou seja, "0,945902", que é a maior saída

da RNA, o sinal *activity* é atualizado com o valor "001". Nesse caso, a atividade classificada é deitado, e o número de identificação mostrado no display de 7 segmentos é 1.

Figura 38 – Simulação da entidade *select_result*.

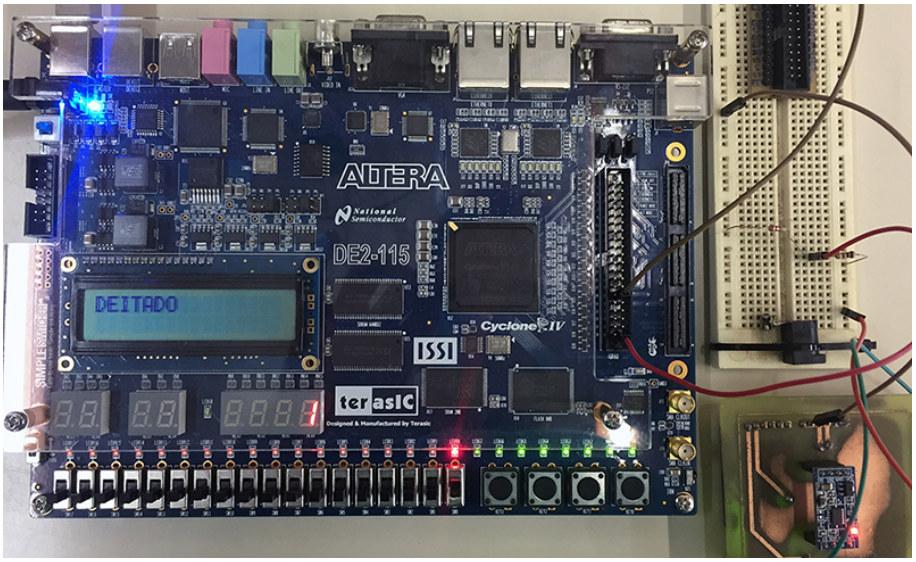


Fonte: Elaboração do próprio autor.

4.3 AVALIAÇÃO DO SISTEMA NO FPGA

Com a finalização da implementação em VHDL, o sistema foi configurado no FPGA, como pode ser visualizado em operação na Figura 39. Nesse exemplo de classificação, a posição do acelerômetro posicionado sobre a bancada reproduz a postura da atividade deitado. No display de LCD é apresentado o nome da atividade, e o display de 7 segmentos mostra o número da atividade classificada. O diagrama de blocos das entidades implementadas no VHDL está apresentado no Apêndice A.

Figura 39 – Exemplo de funcionamento da implementação em VHDL.

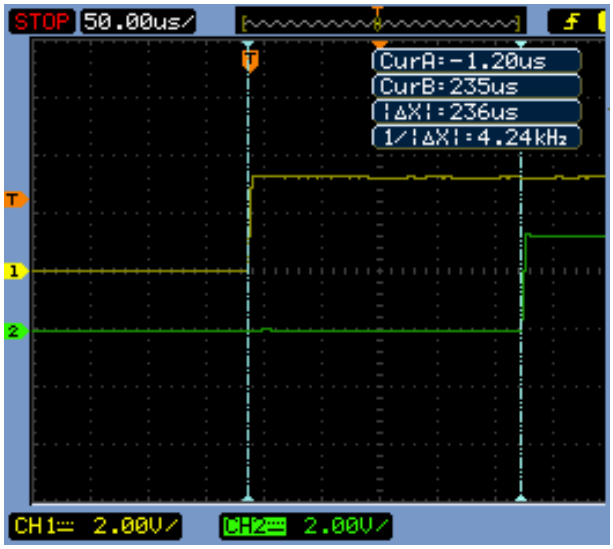


Fonte: Elaboração do próprio autor.

Apresenta-se, na Figura 40, o tempo necessário para o hardware executar uma parte do algoritmo de classificação. A transição da borda de subida do sinal no canal 1 do

osciloscópio (representado na cor amarela) indica o recebimento da última amostra da janela, e a transição de borda de subida do sinal do canal 2 (cor verde) representa o término da operação da entidade que seleciona a maior saída da RNA, ou seja, o momento que obteve-se a classificação da atividade realizada.

Figura 40 – Tempo de resposta da classificação realizada em hardware.



Fonte: Elaboração do próprio autor.

No intervalo de 236 microssegundos, as seguintes operações são executadas: filtragem da última amostra do eixo Z; extração e normalização dos atributos do eixo Z; processamento do algoritmo da RNA e a seleção do resultado da classificação.

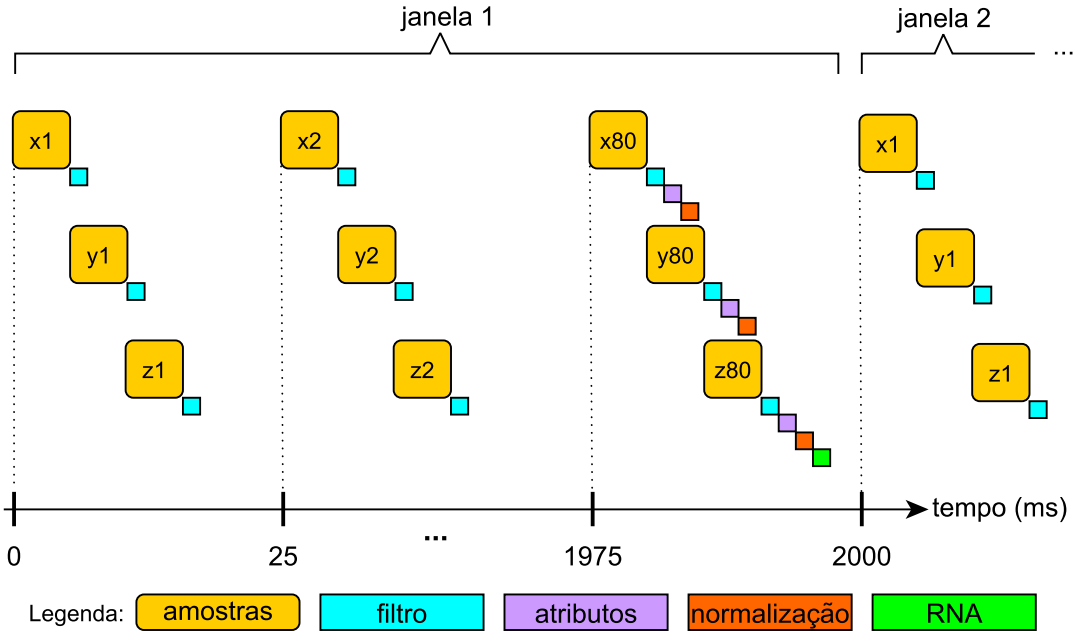
Dessa forma, a descrição em hardware opera em tempo real, pois gera o resultado da classificação da janela atual antes do recebimento da primeira amostra da próxima janela. Essa característica garante que nenhuma aceleração seja perdida.

Representa-se, na Figura 41, o comportamento da classificação realizada em hardware. Quando uma amostra da aceleração é recebida, realiza-se a filtragem antes da leitura da próxima. Após a filtragem da última amostra de cada eixo, os atributos desse eixo são extraídos e normalizados. Finalmente, aplica-se o algoritmo da RNA que gera a classificação da janela atual, antes do recebimento da primeira amostra da próxima janela.

Apresenta-se, na Tabela 11, os recursos do FPGA consumidos na descrição em hardware. Os componentes mais limitados foram os multiplicadores embarcados, que não foram o suficiente para descrever processos paralelos na RNA. Dessa forma, o sistema foi implementado com apenas um neurônio, obtendo-se processamento sequencial na entidade da RNA.

Verificou-se a possibilidade de instanciar 12 conjuntos das entidades *sum_neuron* e

Figura 41 – Representação da execução no hardware em tempo real.



Fonte: Elaboração do próprio autor.

sigmoid, um para cada neurônio da camada de entrada, com a finalidade de realizar as combinações lineares paralelamente. Porém esse projeto não foi compilado, pois os multiplicadores embarcados foram totalmente consumidos.

Tabela 11 – Hardware consumido na descrição em VHDL.

Componente	Total	Utilizado	Porcentagem
Elementos lógicos	114480	24671	21,55%
Funções combinacionais	114480	21853	19,09%
Registradores lógicos dedicados	114480	14870	12,99%
Pinos de entrada e saída	529	33	6,24%
Bits de memória	3981312	20949	0,53%
Multiplicadores embarcados	532	133	25%

Fonte: Elaborado pelo próprio autor.

Contudo, a implementação de apenas um neurônio ofereceu o desempenho desejado e garantiu a operação do sistema em tempo real.

A implementação em VHDL descrita nesse capítulo gerou uma versão embarcada mais eficiente comparada ao sistema configurado no processador Nios II. Com o desenvolvimento de um hardware específico, obteve-se a otimização necessária para processar as informações em tempo real, possibilitando a aplicação do SIRAH em situações que exigem respostas instantâneas.

5 CONCLUSÕES

O SIRAH é um sistema aplicado em HAR, que utiliza os dados de um acelerômetro localizado na cintura do usuário e uma RNA para classificar sete atividades atividades: em pé, deitado, sentado, andar, caminhar, sentar e levantar. Na primeira versão, o algoritmo do SIRAH foi implementado no MATLAB e realiza classificações *offline*.

Para realizar classificações *online*, isto é, durante a execução das atividades, foram desenvolvidas duas versões embarcadas do SIRAH, uma baseada em software, codificada em linguagem C que é executada no processador Nios II da Altera, e a outra implementada em hardware, descrita em VHDL e gravada em um FPGA.

Verificou-se que o treinamento da RNA no processador Nios II não é eficiente pelo alto consumo de tempo. A melhor alternativa é manter o treinamento no modo *offline*, podendo ser efetuado no MATLAB ou com a codificação em C executada em um microcomputador, pois oferecem a mesma exatidão. Para classificar uma janela, o processador Nios II consome 673 milissegundos, que é um intervalo de tempo viável. Nessa implementação, a exatidão das classificações é a mesma obtida no software MATLAB, que é de 95%.

Com o processamento paralelo desenvolvido utilizando-se a linguagem de descrição de hardware VHDL, implementada em FPGA, obteve-se um sistema que realiza a classificação de uma janela em 236 microssegundos, ou seja, em tempo real. Desse modo, o objetivo do trabalho foi alcançado com sucesso, pois obteve-se classificações *online* com processamento eficiente e com a mesma exatidão das classificações *offline*.

Como proposta de trabalhos futuros, outros tipos de atividades podem ser classificadas. Uma alternativa de acesso à essas informações são bancos de dados disponibilizados gratuitamente na internet, que são utilizados em outras pesquisas da área. Outra possibilidade é a implementação de diferentes arquiteturas de RNA e outras combinações de atributos, que também podem ser descritas em hardware. Por exemplo, cita-se as redes da família ART (*Adapitive Resonance Theory*).

REFERÊNCIAS

ABU-MOSTAFA, Y. S.; MAGDON-ISMAIL, M.; LIN, H. **Learning from data**. Pesadena: AMLBook, 2012. 215 p.

ACAMPORA, G.; COOK, D.; RASHIDI, P.; VASILAKOS, A. A survey on ambient intelligence in healthcare. **Proceedings of the IEEE**, New York, v. 101, n. 12, p. 2470-2494, 2013.

AL-ANBUKY, A.; RUDOLPH, S.; HAEHNER, J.; TOMFORDE, S. Public space ambient intelligence systems: benefits, approaches and challenges. In: INTERNATIONAL CONFERENCE ON ARCHITECTURE OF COMPUTING SYSTEMS - ARCS, 28., 2015, Porto. **Anais...** Porto: Springer, 2015. p. 1–6.

ALSHEIKH, M. A.; LIN, S.; NIYATO, D.; TAN, H. P. Machine learning in wireless sensor networks: algorithms, strategies, and applications. **IEEE Communications Surveys Tutorials**, New York, v. 16, n. 4, p. 1996-2018, 2014.

ANDREU, J.; BARUAH, R. D.; ANGELOV, P. Real time recognition of human activities from wearable sensors by evolving classifiers. In: INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS - FUZZ, 20., 2011, Taipei. **Anais...** Taipei: IEEE, 2011. p. 2786-2793.

BARONTI, P.; PILLAI, P.; CHOOK, V. W.; CHESSA, S.; GOTTA, A.; HU, Y. F. Wireless sensor networks: a survey on the state of the art and the 802.15.4 and ZigBee standards. **Computer Communications**, Amsterdam, v. 30, n. 7, p. 1655-1695, 2007.

BASTERRETXEA, K.; ECHANOBE, J.; CAMPO, I. D. A wearable human activity recognition system on a chip. In: CONFERENCE ON DESIGN AND ARCHITECTURES FOR SIGNAL AND IMAGE PROCESSING - DASIP, 8., 2014, Madrid. **Anais...** Madrid: IEEE, 2014. p. 1-8.

BAYAT, A.; POMPLUN, M.; TRAN, D. A. A study on human activity recognition using accelerometer data from smartphones. **Procedia Computer Science**, Niagara Falls, v. 34, n. 1, p. 450-457, 2014.

BENMANSOUR, A.; BOUCHACHIA, A.; FEHAM, M. Human activity recognition in pervasive single resident smart homes: state of art. In: INTERNATIONAL SYMPOSIUM ON PROGRAMMING AND SYSTEMS - ISPS, 12., 2015, Algiers. **Anais...** Algiers: IEEE, 2015. p. 1-9.

BULLING, A.; BLANKE, U.; SCHIELE, B. A tutorial on human activity recognition using body-worn inertial sensors. **ACM Computing Surveys**, New York, v. 46, n. 3, p. 33:1–33:33, 2014.

CHENG, L.; LI, Y.; GUAN, Y. Human activity recognition based on compressed sensing. In: ANNUAL COMPUTING AND COMMUNICATION WORKSHOP AND CONFERENCE - CCWC, 7., 2017, Las Vegas. **Anais...** Las Vegas: IEEE, 2017. p. 1-7.

CHIN, J.; TISAN, A. Ubiquitous approach to body hydration testing. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ENVIRONMENTS - IE, 11., 2015, Prague. **Anais...** Prague: IEEE, 2015. p. 144-147.

CORPORATION, A. **Nios II classic processor reference guide**. San Diego: Altera Corporation, 2016a. 268 p.

CORPORATION, A. **Floating-point IP cores user guide**. San Diego: Altera Corporation, 2016b. 164 p.

CORPORATION, I. **Introduction to Intel® FPGA IP cores**. [S.l.]: Intel Corporation, 2017. 39 p.

CRISTANI, M.; KARAFILI, E.; TOMAZZOLI, C. Improving energy saving techniques by ambient intelligence scheduling. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS - AINA, 29., 2015, Guwangiu. **Anais...** Guwangiu: IEEE, 2015. p. 324-331.

DINÇ, I.; SIGDEL, M.; DINÇ, S.; SIGDEL, M. S.; PUSEY, M. L.; AYGUN, R. S. Evaluation of normalization and pca on the performance of classifiers for protein crystallization images. In: IEEE SOUTHEASTCON, 28., 2014, Lexington. **Anais...** Lexington: IEEE, 2014. p. 1-6.

DURANGO, M. DE J. B. **SIRAH: sistema de reconhecimento de atividades humanas e avaliações de equilíbrio**. 2017. 120 f. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia, Universidade Estadual Paulista - UNESP, Ilha Solteira, 2017.

DÓCUSSE, T. A. **Sistema embarcado para suporte ao diagnóstico de microcalcificações em mamografias digitais**. 2014. 138 f. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia, Universidade Estadual Paulista - UNESP, Ilha Solteira, 2014.

FLOOS, A.; AL-MOGREN, A. Smart bodynet for hypercube body sensor network. In: NATIONAL SYMPOSIUM ON INFORMATION TECHNOLOGY: TOWARDS NEW SMART WORLD - NSITNSW, 5., 2015, Riyadh. **Anais...** Riyadh: IEEE, 2015. p. 1-6.

GARCIA-VALVERDE, T.; SERRANO, E.; BOTIA, J. A. Combining the real world with simulations for a robust testing of ambient intelligence services. **Artificial Intelligence Review**, Norwell, v. 42, n. 4, p. 723–746, 2014.

GJORESKI, M.; GJORESKI, H.; LUTREK, M.; GAMS, M. Automatic detection of perceived stress in campus students using smartphones. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ENVIRONMENTS - IE, 11., 2015, Prague. **Anais...** Prague: IEEE, 2015. p. 132-135.

HA, S.; CHOI, S. Convolutional neural networks for human activity recognition using multiple accelerometer and gyroscope sensors. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS - IJCNN, 4., 2016, Vancouver. **Anais...** Vancouver: IEEE, 2016. p. 381-388.

HAARTSEN, J. C. The Bluetooth radio system. **IEEE Personal Communications**, Piscataway, v. 7, n. 1, p. 28-36, 2000.

HOMOLA, M.; PATKOS, T.; FLOURIS, G.; SEFRÁNEK, J.; SIMKO, A.; FRTÚS, J.; ZOGRAFISTOU, D.; BALÁZ, M. Resolving conflicts in knowledge for ambient intelligence. **The Knowledge Engineering Review**, London, v. 30, n. 5, p. 455–513, 2015.

HONG, X. Segmenting sensor data for activity monitoring in smart environments. **Personal and Ubiquitous Computing**, Berlin, v. 17, n. 3, p. 545–559, 2012.

IEEE, S. IEEE standard for floating-point arithmetic. **IEEE Std 754-2008**, v. 1, n. 1, p. 1-70, 2008.

JOAQUINITO, R.; SARMENTO, H. A wireless biosignal measurement system using a SoC FPGA and Bluetooth low energy. In: INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS - ICCE, 6., 2016, Berlin. **Anais...** Berlin: IEEE, 2016. p. 36-40.

KAKDE, A.; GULHANE, V. Real time composite user activity modelling using hybrid approach for recognition. In: INTERNATIONAL CONFERENCE ON ELECTRICAL, COMPUTER AND COMMUNICATION TECHNOLOGIES - ICECCT, 1., 2015, Coimbatore. **Anais...** Coimbatore: IEEE, 2015. p. 1-6.

KASHYAP, H.; SINGH, V.; CHAUHAN, V.; SIDDHI, P. A methodology to overcome challenges and risks associated with ambient intelligent systems. In: INTERNATIONAL CONFERENCE ON ADVANCES IN COMPUTER ENGINEERING AND APPLICATIONS - ICACEA, 7., 2015, Ghaziabad. **Anais...** Ghaziabad: IEEE, 2015. p. 245-248.

KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: PROCEEDINGS OF THE 14TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE - VOLUME 2, 14., 1995, Montreal. **Anais...** Montreal: Morgan Kaufmann Publishers, 1995. p. 1137–1143.

LABRADOR, M. A.; YEJAS, O. D. L. **Human activity recognition: using wearable sensors and smartphones**. Boca Raton: CRC Press, 2013. 207 p.

LAI, C. C.; JHAN, C. F.; WANG, W. S. Digital image watermarking using dct and z-score transform. In: 2010 INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS, 9., 2010, Qingdao. **Anais...** Qingdao: IEEE, 2010. p. 2933-2937.

LARA, O. D.; LABRADOR, M. A. A survey on human activity recognition using wearable sensors. **IEEE Communications Surveys Tutorials**, New York, v. 15, n. 3, p. 1192-1209, 2013.

LU, L. X.; HUANG, Z.; HOU, Y.; LI, J.; WANG, S. A safe and fast connecting strategy of the Bluetooth identification sensors in ambient intelligence environment. In: INTERNATIONAL SYMPOSIUMS ON INFORMATION PROCESSING - ISIP, 1., 2008, St. Louis. **Anais...** St. Louis: IEEE, 2008. p. 538-542.

LUBINA, P.; RUDZKI, M. Artificial neural networks in accelerometer-based human activity recognition. In: ITERNATIONAL CONFERENCE MIXED DESIGN OF INTEGRATED CIRCUITS SYSTEMS - MIXDES, 22., 2015, Torun. **Anais...** Torun: IEEE, 2015. p. 63-68.

MAGNO, M.; BOYLE, D. Wearable energy harvesting: from body to battery. In: INTERNATIONAL CONFERENCE ON DESIGN TECHNOLOGY OF INTEGRATED SYSTEMS IN NANOSCALE ERA - DTIS, 12., 2017, Palma de Mallorca. **Anais...** Palma de Mallorca: IEEE, 2017. p. 1-6.

MATHIE, M. J.; COSTER, A. C.; LOVELL, N. H.; CELLER, B. G. Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement. **Physiological measurement**, Philadelphia, v. 25, n. 2, p. R1, 2004.

MATTER, B. **Numerical computing with IEEE floating point arithmetic**. Philadelphia: Society for Industrial and Applied Mathematics, 2001. 121 p.

MENDEZ, G.; CASILLAS, M.; BALTAZAR, R.; LINO, C.; MANCILLA, L.; LOPEZ, S. Intelligent management system for the conservation of energy. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ENVIRONMENTS - IE, 11., 2015, Prague. **Anais...** Prague: IEEE, 2015. p. 120-123.

PAUWELS, E.; SALAH, A. A.; TAVENARD, R. Sensor networks for ambient intelligence. In: WORKSHOP ON MULTIMEDIA SIGNAL PROCESSING - MMSP, 9., 2007, Chania Crete. **Anais...** Chania Crete: IEEE, 2007. p. 13-16.

PREECE, S. J.; GOULERMAS, J. Y.; KENNEY, L. P. J.; HOWARD, D.; MEIJER, K.; CROMPTON, R. Activity identification using body-mounted sensors - a review of classification techniques. **Physiological Measurement**, Philadelphia, v. 30, n. 4, p. R1, 2009.

RAMOS-GARCIA, R. I.; TIFFANY, S.; SAZONOV, E. Using respiratory signals for the recognition of human activities. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY - EMBC, 38., 2016, Orlando. **Anais...** Orlando: IEEE, 2016. p. 173-176.

RESENDES, S.; CARREIRA, P.; SANTOS, A. C. Conflict detection and resolution in home and building automation systems: a literature review. **Journal of Ambient Intelligence and Humanized Computing**, New York, v. 5, n. 5, p. 699–715, 2013.

ROY, S.; MANDAL, S.; HANUMAIAH, N. MEMS accelerometer: from engineering to medicine. **IEEE Potentials**, New York, v. 35, n. 2, p. 30-33, 2016.

SADRI, F. Ambient intelligence: a survey. **ACM Computing Surveys - CSUR**, New York, v. 43, n. 4, p. 36, 2011.

SAMARAH, S.; ZAMIL, M. G. A.; ALEROUD, A. F.; RAWASHDEH, M.; ALHAMID, M. F.; ALAMRI, A. An efficient activity recognition framework: toward privacy-sensitive health data sensing. **IEEE Access**, Riyadh, v. 5, n. 5, p. 3848-3859, 2017.

SEMICONDUCTOR, F. **±2g/±4g±8g three axis low-g digital output accelerometer**. Austin: Freescale Semiconductor, 2009. Disponível em: <http://www.nxp.com/assets/documents/data/en/data-sheets/MMA7455L.pdf>. Acesso em: 22 dez. 2016.

SHIN, J.; ITTEN, D.; RUSAKOV, A.; MEYER, B. Smartwalker: towards an intelligent robotic walker for the elderly. In: INTERNATIONAL CONFERENCE ON

INTELLIGENT ENVIRONMENTS - IE, 11., 2015, Prague. **Anais...** Prague: IEEE, 2015. p. 9-16.

SILVA, F. G. D.; GALEAZZO, E. Accelerometer based intelligent system for human movement recognition. In: INTERNATIONAL WORKSHOP ON ADVANCES IN SENSORS AND INTERFACES - IWASI, 5., 2013, Bari. **Anais...** Bari: IEEE, 2013. p. 20-24.

SMITH, S. W. **The scientist and engineer's guide to digital signal processing**. San Diego: California Technical Publishing, 1999. 640 p.

TANG, Z.; GUO, J.; MIAO, S.; ACHARYA, S.; FENG, J. H. Ambient intelligence based context-aware assistive system to improve independence for people with autism spectrum disorder. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES - HICSS, 49., 2016, Koloa. **Anais...** Koloa: IEEE, 2016. p. 3339-3348.

TONCHEV, K.; SOKOLOV, S.; VELCHEV, Y.; BALABANOV, G.; POULKOV, V. Recognition of human daily activities. In: INTERNATIONAL CONFERENCE ON COMMUNICATION WORKSHOP - ICCW, 25., 2015, London. **Anais...** London: IEEE, 2015. p. 290-293.

TOSCHI, G. M.; CAMPOS, L. B.; CUGNASCA, C. E. Home automation networks: a survey. **Computer Standards & Interfaces**, Amsterdam, v. 50, n. 1, p. 42 - 54, 2017.

TRAN, D. N.; PHAN, D. D. Human activities recognition in android smartphone using support vector machine. In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS, MODELLING AND SIMULATION - ISMS, 7., 2016, Bangkok. **Anais...** Bangkok: IEEE, 2016. p. 64-68.

UDDIN, M. T.; UDDINY, M. A. Human activity recognition from wearable sensors using extremely randomized trees. In: INTERNATIONAL CONFERENCE ON ELECTRICAL ENGINEERING AND INFORMATION COMMUNICATION TECHNOLOGY - ICEEICT, 2., 2015, Dhaka. **Anais...** Dhaka: IEEE, 2015. p. 1-6.

VALES-ALONSO, J.; CHAVES-DIEGUEZ, D.; LÓPEZ-MATENCIO, P.; ALCARAZ, J. J.; PARRADO-GARCÍA, F. J.; GONZÁLEZ-CASTAÑO, F. J. Saeta: a smart coaching assistant for professional volleyball training.. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, New York, v. 45, n. 8, p. 1138-1150, 2015.

WANG, A.; CHEN, G.; YANG, J.; ZHAO, S.; CHANG, C. Y. A comparative study on human activity recognition using inertial sensors in a smartphone. **IEEE Sensors Journal**, New York, v. PP, n. 99, p. 1-1, 2016.

WEISER, M. The computer for the 21st century. **Scientific american**, v. 265, n. 3, p. 94–104, 1991.

WOZNOWSKI, P.; FAFOUTIS, X.; SONG, T.; HANNUNA, S.; CAMPLANI, M.; TAO, L.; PAIEMENT, A.; MELLIOS, E.; HAGHIGHI, M.; ZHU, N.; HILTON, G.; DAMEN, D.; BURGHARDT, T.; MIRMEHDI, M.; PIECHOCKI, R.; KALESHI, D.; CRADDOCK, I. A multi-modal sensor infrasructure for healthcare in a residential environment. In: INTERNATIONAL CONFERENCE ON COMMUNICATION WORKSHOP - ICCW, 25., 2015, London. **Anais...** London: IEEE, 2015. p. 271-277.

ZHOU, B.; CHENG, J.; LUKOWICZ, P.; REISS, A.; AMFT, O. Monitoring dietary behavior with a smart dining tray. **Pervasive Computing, IEEE**, Los Alamitos, v. 14, n. 4, p. 46-56, 2015.

APÊNDICE A – PRINCIPAIS ENTIDADES DO SISTEMA IMPLEMENTADO EM HARDWARE

- *sirah*
  - *wait_sample*
  - *store_samples*
    - *conv_raw*
    - *sum_samples*
  - *features*
    - *pow_sum*
    - *standard_derivation*
    - *skewness*
    - *kurtosis*
  - *ann*
    - *sum_neuron*
    - *sigmoid*
  - *select_result*
  - *display*

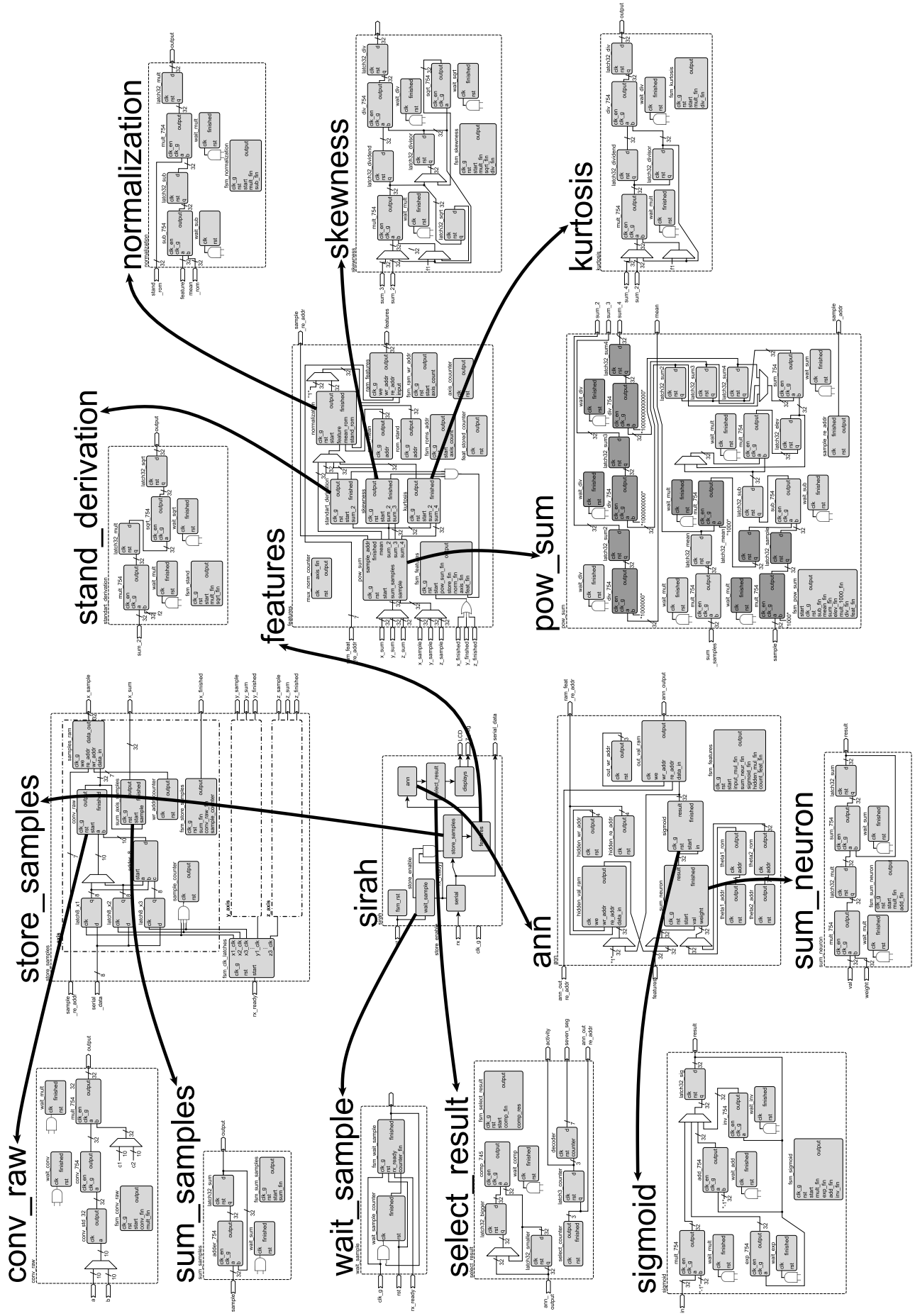


Figura 42 – Esquemático simplificado do sistema implementado em hardware.