UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Mauro Rodrigo Larrat Frota e Silva

LOCALIZAÇÃO DISTRIBUÍDA EM REDES DE SENSORES SEM FIO UTILIZANDO FPGA E REDE NEURAL ARTIFICIAL

Belém
2015

Mauro Rodrigo Larrat Frota e Silva

LOCALIZAÇÃO DISTRIBUÍDA EM REDES DE SENSORES SEM FIO UTILIZANDO FPGA E REDE NEURAL ARTIFICIAL.

Dissertação de Mestrado apresentada para obtenção do grau de Mestre em Ciência da Computação.

Programa de Pós-Graduação em Ciência da Computação.

Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Área de concentração em Sistemas de Computação, linha de pesquisa em Redes de Sensores Sem Fio.

Orientador Prof. Dr. Claudomiro de Souza de Sales Júnior.

Belém

2015

Mauro Rodrigo Larrat Frota e Silva

LOCALIZAÇÃO DISTRIBUIDA EM REDES DE SENSORES SEM FIO UTILIZANDO FPGA E REDE NEURAL ARTIFICIAL.

Dissertação de Mestrado apresentada para obtenção do grau de Mestre em Ciência da Computação. Programa de Pós-Graduação em Ciência da Computação. Instituto de Ciências Exatas e Naturais. Universidade Federal do Pará.

Data de aprovação: 16/12/2015, Belém-PA.

Banca Examinadora

Prof. Dr. Claudomiro de Souza de Sales Júnior.

PPGCC – UFPA – Orientador

Prof. Dr. Aldebaro Barreto da Rocha Klautau Júnior.

PPGCC – UFPA – Membro Interno

Prof. Dr. Francisco Carlos Bentes Frey Muller.

FACOMPTEC – UFPA – Membro Externo

Belém

2015

RESUMO

Em Redes de Sensores sem Fio (RSSF), métodos de localização distribuída vem sendo desenvolvidos considerando a eficiência dos recursos de rede, diminuindo o consumo de energia através da redução da transmissão de informação por parte dos sensores onde estas restrições se aplicam. Neste trabalho, a localização distribuída é baseada em apenas uma transmissão em broadcast proveniente de um sensor alvo para quatro âncoras. A partir da interação entre dois âncoras, o método já fornece uma estimativa razoável devido as características do algoritmo que o implementa. O algoritmo também permite que o método possa fornecer uma estimativa de posição mediante a tolerância de falhas de âncoras, respeitando o mínimo de dois sensores âncoras necessários para realizar o procedimento de localização. Para o desenvolvimento deste trabalho propõe-se a implementação de Redes Neurais Artificiais em cada âncora utilizando plataforma configurável a qual opera em alta frequência. Os resultados mostram que a posição estimada do alvo melhora em função do aumento do número de âncoras e alcança rápida estabilidade de estimativa de posição sem exigir muitas âncoras no processo de localização.

PALAVRAS-CHAVE: Redes de Sensores Sem Fio, RSSI, Rede Neural Artificial, FPGA, Localização Distribuída.

ABSTRACT

In Wireless Sensor Networks (WSN), distributed localization methods have been developed considering the network resource efficiency, reducing power consumption by reducing the part for information transmitted from the targets node where these restrictions apply. In this work, the distributed location is based on only one broadcast transmission from a target node to four anchors nodes. From the interaction between two anchors, the method already provides a reasonable estimation due to the features of the algorithm that it implements. The algorithm also allows the method can provide a position estimation in presence of anchor node failure, respecting the minimum of two anchors nodes required to perform the localization procedure. To develop this work we propose the implementation of Artificial Neural Networks in each anchor node using reconfigurable platform, which operates at high frequency. The results show that the estimated target position improves due to the increased number of anchors and quickly reaches stability in estimating position without requiring many anchors in the localization process.

KEYWORDS: Wireless Sensors Network, RSSI, Artificial Neural Network, FPGA, Distributed Localization.

LISTA DE FIGURAS

LISTA DE ACRÔNIMOS

RSSF	Rede de Sensores Sem Fio.
EEPROM	Memória de Apenas Leitura Programável e Eletricamente Apagável.
UART	Transmissor/Receptor Universal Assíncrono.
DE	Detecção de Eventos.
EPE	Estimativa de Processos Espaciais.
MAC	Protocolo de Controle de Acesso ao Meio.
CSMA	Acesso Múltiplo com Sensoriamento de Portadora.
TDMA	Múltiplos Acessos por Divisão de Tempo.
RE	Retropropagação de Erros.
LSR	Função Logarítmica Sigmoide Rápida.
LoS	Linha de Visada.
MDS	Escalonamento Multidimensional.
RSSI	Indicador de Intensidade de Sinal Recebido.
ToA	Tempo de Chegada do Sinal Recebido.
AoA	Ângulo de Chegada do Sinal Recebido.
RNA	Rede Neural Artificial.
MLP	Multicamadas Perceptron.
RR	Retropropagação Resiliente.
RM	Retropropagação com Momentum.
PDS	Processador Digital de Sinais.
ASIC	Circuito Integrado de Aplicação Específica.
MLE	Estimador de máxima verossimilhança.

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Cenário da Localização Distribuída em Redes de Sensores Sem Fio

Em RSSF, conhecer o local onde ocorrem os processos monitorados pela rede é uma tarefa fundamental para diversas aplicações. Diversos sensores com posição desconhecidas (alvos) são despejados aleatoriamente em uma região de interesse e transmitem informações para outros sensores com posição conhecida (âncoras), sendo, dessa forma, possível identificar a origem das informações através de métodos que se utilizam de propriedades do sinal recebido. Métodos de localização distribuída consideram as restrições dos recursos dos sensores alvos durante o processo de estimativa de localização. O recurso energético é o principal fator que restringe o desenvolvimento dos métodos de localização e está associado principalmente a quantidade de mensagens que os sensores alvos precisam transmitir ou rotear até os sensores âncoras para poderem ser localizados (TANG, MINGJIAN, *et al.*, 2008), (MAMUN, 2012), (REZAEI, ZAHRA e MOBININEJAD, 2012). Além do recurso energético, os métodos de localização distribuída buscam considerar o menor uso de recursos adicionais possíveis em sensores alvo, evitando assim custos adicionais, principalmente pela maioria das aplicações exigirem uma grande quantidade de sensores para monitoramento. Esses recursos adicionais podem ser antenas mais sofisticadas que calculam o ângulo de chegada do sinal ou diferentes tipos de sinais transmitidos (infravermelho e acústico, por exemplo). Além disso, por uma questão de custo ou pela forma de aplicação, alguns sensores podem ser danificados durante o processo de implantação, podendo ser necessário que o método de localização consiga operar sem que as informações destes sensores sejam consideradas no processo. A principal característica contra falhas, comum entre os métodos de localização distribuída, é a capacidade de divisão do processo de localização entre vários sensores na rede. Assim, o método deve ser capaz de reconstruir uma solução a partir das informações adquiridas mesmo diante da falta de informações *a priori*. Para realizar eficientemente o processo de localização, o algoritmo que

implementa o método de localização pode se utilizar de atributos inerentes aos sensores da rede ou das propriedades do sinal transmitido.

1.2 Motivação

Esta pesquisa está sendo desenvolvida para mostrar que a utilização de FPGA aliada a rede neural artificial (RNA) pode ser considerada promissora no processo de localização distribuída em RSSF.

Um sensor inteligente pode contribuir com outros sensores na rede, distribuindo o pré-processamento, informando uma estimativa da posição de outro dispositivo. Espera-se que esta pesquisa contribua e incentive a utilização de FPGA e RNA em aplicações que envolvam localização distribuída em RSSF, mostrando que o nível de complexidade de desenvolvimento através desta abordagem é razoável em função do ganho em relação ao que se busca nos avanços em pesquisas de localização em RSSF.

A tecnologia FPGA é o que se tem de mais avançado atualmente em se tratando de hardware reconfigurável, de alto desempenho e de baixo custo, acessível para pesquisadores desenvolverem novas tecnologias antes de produzirem para o mercado (JAMEL, M., *et al.*, 2012) um produto final.

Como requisito inicial, a escolha da RNA levou em consideração a similaridade funcional com relação à modelagem do sensor, além da robustez computacional do algoritmo (JAMEL, M., *et al.*, 2012). Quanto à similaridade, foi considerada a estrutura conexionista do algoritmo e a característica de processamento paralelo. Os parâmetros que compõe a RNA podem mapear completamente as estruturas funcionais de um sensor. Pode-se modelar um sensor como apenas um neurônio ou como uma RNA com várias camadas, dependendo da necessidade da aplicação. As entradas do sensor podem ser modeladas como as entradas da RNA, as quais são processadas paralelamente pelos neurônios, como ocorre na plataforma FPGA. Além disso, fornecem uma ou mais saídas que podem ser implementadas como atuadores do dispositivo ou a interface de transmissão de informação nas aplicações.

RNAs podem ser treinadas por meio de operações matriciais que podem ser executadas na maioria das ferramentas científicas existentes, assim como linguagens de programação, ferramentas conhecidas tais quais o *Matlab* e o *RStudio*, com uma relativa facilidade. Além disso, de acordo com o teorema universal de aproximação (CSÁJI e CSANÁD, 2001), RNAs

podem aproximar qualquer função contínua com apenas uma camada escondida e muitas funções descontinuas com duas ou mais camadas escondidas. O número de neurônios nestas camadas é dependente do problema abordado, o que garante flexibilidade de implementação.

As implicações são a baixa necessidade de espaço de armazenamento (matrizes) para o algoritmo dentro do sensor e um rápido processamento. RNAs estão atualmente em foco no desenvolvimento tecnológico com novos métodos sendo recentemente desenvolvidos, como o *Deep Learning* e RNAs Pulsantes (SCHMIDHUBER, 2015), (HUANG, HUANG, *et al.*, 2015).

A escolha da plataforma FPGA levou em consideração a facilidade de aquisição do produto, o custo, o desempenho, a flexibilidade de design, a compatibilidade com o algoritmo adotado e a relevância atual da tecnologia. Das principais tecnologias disponíveis no mercado, podemos citar o PDS (*digital signal processor* - processador digital de sinais), o ASIC (*application specific integrated circuit* - circuito integrado de aplicação específica) e o FPGA (*field porgrammable gate array* - arranjo de portas programáveis em campo). O DSP possui alto poder de processamento, mas é centrado em computação sequencial, diferentemente do FPGA, que é centrado em computação paralela. Além disso, o FPGA é reconfigurável em nível tanto de software quanto de hardware. O DSP possui a configuração de hardware fixada, sendo necessária a sua substituição no caso de alteração da configuração física. A característica paralela do FPGA se adequa ao modo como o processamento ocorre dentro das RNAs, onde as unidades de processamento, os neurônios, são processadas simultaneamente em cada camada. O ASIC possui as mesmas características do FPGA com uma exceção, ele não é reprogramável em nível de hardware. Esta restrição torna o ASIC mais adequado para uso em projetos já finalizados.

Quanto ao ganho em implementar uma RNA utilizando o FPGA, podem-se destacar alguns fatores (JAMEL, M., *et al.*, 2012):

- FPGA fazem parte da família VLSI, permitindo a utilização de milhares de portas lógicas, favorecendo a aplicação de RNAs com muitas camadas.
- Podem ser reconfigurados mesmos já integrados no sistema, permitindo que a estrutura interna (assumindo o mesmo número de entradas e saídas) da RNA seja atualizada.
- Possuem um rápido tempo de desenvolvimento, incluindo design e aplicação, facilitando testes com diversos valores para os parâmetros de RNAs.

FPGAs implementam o paralelismo na execução de operações, assim como ocorrem em RNAs, resultando em maior velocidade de processamento.

1.3 Trabalhos Relacionados

Em (MOHAMMAD, PARK e KI-DOO, 2012) é desenvolvido um experimento em uma área 20x20m indoor, onde uma RNA de regressão generalizada (uma variação da rede de base radial com uma segunda camada linear) em conjunto com o algoritmo centroide ponderada são aplicados para estimar a posição de alvos através da potência do sinal recebida por quatro âncoras mais próximos do alvo em questão. 25 alvos são posicionados dentro da área espaçados por uma distância de 4m entre os alvos adjacentes, similar ao experimento desenvolvido por este trabalho. A estrutura da RNA de regressão generalizada possui uma camada escondida com quatro neurônios, assim como as quatro entradas da rede. Foram utilizados 10% da base de dados para validação após o treino com os 90% restantes. A avaliação do erro durante o treino da RNA foi calculada com base no erro quadrático médio (EQM) da distância entre a posição a priori e a posição estimada pela RNA. Os autores compararam o algoritmo do método proposto com mais dois algoritmos, a saber, perceptron multicamadas e *maximum likilihood*. Além do comparativo baseado no EQM, os autores ressaltam a vantagem estrutural da RNA auto regressiva sobre a RNA Perceptron quanto a simplicidade de decisão da quantidade de camadas escondidas e neurônios definidos na estrutura desses algoritmos. Resultados experimentais mostram que o método proposto pelos autores alcança um EQM próximo ao resultado alcançado por este trabalho, considerando o uso da centróide ponderada no auxílio do algoritmo RNA auto regressiva. Utilizando apenas a RNA auto regressiva, o método proposto pelos autores apresenta maior erro. Para os outros dois algoritmos utilizados como comparativo no referido trabalho, o perceptron multicamadas apresentou o terceiro melhor desempenho e o *maximum likelihood* foi o pior caso. Nesta dissertação, alcançamos resultados sutilmente melhores utilizando apenas a versão perceptron multicamadas.

Em (GHOLAMI, CAI e BRENNAN, 2013) é ressaltada a aplicação de RNA em localização em situações práticas onde existem intempéries tais quais ruídos, incerteza de medição e inospitalidade do ambiente mensurado. Para ambientes *outdoors*, estes problemas são causados devido à atenuação do sinal no meio de propagação, à distância ou desalinhamento dos sensores, condições desfavoráveis do ambiente (pressão, temperatura e humidade) e a multi rotas tomadas pelo sinal. Para ambientes indoors, a difração e a refração são os principais obstáculos causados

por objetos. (GHOLAMI, CAI e BRENNAN, 2013) ressalta ainda interferências em ambientes industriais, provenientes da vibração de máquinas e de interferência eletromagnética de equipamentos eletrônicos. Simulações e experimentos práticos são executados dando atenção para ambientes com obstáculos e para alvos móveis. Outro ponto importante é quanto à falta de referências que experimentam RNA em localização distribuída, onde se tem mais de uma RNA desenvolvida em sensores que desempenham a função de localização de alvos. O experimento realizado nesta referência consiste em uma simulação de dois tipos de RNAs (com uma e com duas camadas escondidas) e um algoritmo de trilateração. Tanto as duas RNAs quanto a trilateração consideram o sinal da técnica AoA como parâmetro para estimar a distância Euclidiana entre o alvo e os âncoras. Cinco sensores âncoras são utilizados no trajeto simulado pelo alvo. Foram computadas 200 amostras, igualmente espaçadas em percurso sinuoso de 200 metros, para cada um dos 75 testes realizados em um ambiente industrial simulado, que considerou sete características físicas, incluindo a inclusão de ruídos, atenuação de sinal e objetos obstrutivos. O resultado desta simulação apresenta a RNA com duas camadas escondidas como sendo a que possui melhor acurácia e precisão, seguidamente da RNA com uma camada escondida e, por fim, o algoritmo de trilateração. Para validar a simulação, um experimento prático foi realizado, consistindo em uma rota elipsoide de 60 cm por 250 cm, aproximadamente, onde um alvo desenvolve um movimento nesta rota de 6.5 cm/s^{-1} em um percurso de 519,8 cm. Os resultados deste teste prático mostraram que a RNA com duas camadas escondidas apresentou melhor acurácia e precisão sobre as duas outras opções, respectivamente, a RNA com uma camada escondida e a trilateração. A contribuição desta referência é apresentar como a RNA se comporta no processo de localização em ambiente real indoor, dados diferentes tipos de interferências que perturbam o sinal da rede, comparando dois modelos de RNA e um algoritmo comumente utilizado, a trilateração. No entanto, a utilização de dois tipos de sinais através da técnica ToA gera custo adicional de hardware e poderia ser evitado utilizando a própria intensidade da potência do sinal (RSSI).

Em (SABTO e AL MUTIB, 2013), um microcontrolador é utilizado para implementar uma RNA para localizar um robô móvel. Eles consideram duas RNAs, uma para estimar cada coordenada, com apenas uma camada escondida cada uma. O ambiente interno do experimento é de cerca de $2m^2$. Eles sugerem complementar o processo de localização com os recursos adicionais, como sonar e GPS, obtendo informações de ângulo. Essa melhoria reflete na agregação de custo para projeto, além do aumento de consumo de energia. Os resultados

mostram os erros em função da localização do robô móvel, sendo o EQM igual a 0,06. Quanto a angulação, no caso de curvas, o EQM do ângulo é cerca de 0,12. Ressalta-se as dificuldades na implementação da RNA com mais de uma saída (par ordenado) e mais de uma camada oculta (mais informações armazenadas), devido às limitações de recursos computacionais da plataforma. Esta limitação fica clara na necessidade de se utilizar RNAs separadas para cada um dos parâmetros da coordenada da posição do alvo.

Uma análise similar é abordada em (BHARDWAJ, 2013), onde é apresentado uma MLP com retroalimentação, e comparado o desempenho da RNA dados dois métodos de treinamento: Levenberg-Marquardt e retropropagação resiliente (RR). A topologia da RNA foi simulada com quatro entradas, 10 ou 15 neurônios na camada escondida e duas saídas. As funções de ativação das camadas escondidas foram a log-sigmoide e a tangente sigmoide. Na camada de saída utilizou-se a função linear, devido ao contexto linear da resposta da rede. Nos resultados da simulação, a RNA treinada com o algoritmo Levenberg-Marquardt apresentou os melhores resultados com 2, 3 e 4 âncoras. A RNA com função de ativação log-sigmoide apresentou melhor acurácia sobre a função tangente sigmoide. No entanto, (BHARDWAJ, 2013) não desenvolve um experimento prático para avaliar os resultados alcançados considerando um ambiente real, sob fatores como o tempo de propagação do sinal dos sensores ou a falha de transmissão entre os mesmos.

Em (KUMAR e LEE, 2014) é realizada uma simulação experimental em uma área indoor de 5x4 m^2 onde os valores de RSSI são mensurados e passados a uma RNA para estimar a localização de alvos. Quatro sensores âncoras são utilizados nas arestas da área monitorada. Em média, 60 pontos dentro da área monitora foram utilizados para a aquisição de informação *a priori* para treinar a RNA. Estes pontos foram espaçados em uma distância de 50 cm um dos outros. O modelo de RNA utilizado no trabalho foi a multicamadas perceptron (MLP) com retro alimentação, que consiste em uma RNA com múltiplas camadas, com processamento unidirecional (esquerda para a direita) e utilizando o modelo *perceptron* como unidade de processamento. A estrutura da RNA consiste em quatro entradas (RSSI), duas camadas escondidas com doze neurônios cada uma, e duas saídas que representam as coordenadas em um plano. O algoritmo utilizado para treinamento da RNA foi o algoritmo de regulação Bayesiana, implementado na ferramenta *Matlab*. Para os doze neurônios em cada uma das duas camadas escondidas, a função de ativação utilizada foi a tangente sigmoide hiperbólica. Para os neurônios na camada de saída, utilizou-se a função de ativação linear (tecnicamente

conhecida como *pureline*, do inglês). A contribuição de (KUMAR e LEE, 2014) são os vários métodos de treinamento supervisionado empregados e comparados na referência. Os métodos de treinamento supervisionado comparados foram Levenberg-Marquardt, regularização Bayesiana (RB), retropropagação resiliente (*resilent backpropagation*, ou *rprop*, como é mais conhecido), gradiente conjugado escalado (GCE) e gradiente descendente (GD). A base de treino, ou seja, as informações *a priori* contêm 2375 amostras, das quais 60% foram usadas para treino, 20% para validação e 20% para teste. Além da base de teste, outras 105 amostras em pontos desconhecidos dentro da área foram utilizadas para um teste adicional. Nos resultados da referência, o algoritmo de regularização Bayesiana foi o que apresentou melhor desempenho. O erro quadrático médio (EQM) da distância real e da distância estimada foi utilizado como métrica de avaliação de desempenho dos algoritmos. Também considerando as implicações da falta de implementação real, esse trabalho super dimensiona a quantidade de neurônios utilizada nas camadas da RNA. Considerando que nesta dissertação alcança-se bons resultados com menos da metade de neurônios nas camadas escondidas em contraste dos 12 neurônios nas camadas escondidas utilizados em (KUMAR e LEE, 2014). .

Em (SAAD e ALHADY, 2014), uma RNA é embarcada em um microcontrolador para estimar a posição de um robô móvel. Um sinal de sensor infravermelho com alcance até 80cm é utilizado para estimar a distância entre os sensores. O objetivo é similar ao apresentado nesta dissertação: embutir a melhor RNA gerada, através do melhor método de treinamento, em uma plataforma de hardware para aproximar a posição de um alvo, dadas as leituras dos sensores como entradas. O algoritmo de treinamento LM mostrou o erro de treinamento mais baixo dentre os métodos de treinamento. A medida da área ensaiada é de 90x80cm. A RNA modelada tem uma entrada, uma camada escondida (com três neurônios) e uma saída. Os autores citam a limitação sobre o tempo de processamento e memória associada à plataforma de hardware escolhida. A RNA, como uma estimadora de posição, apresentou-se como opção atraente, mas a plataforma na qual é aplicado tem implicações significativas - quanto mais neurônios e camadas escondidas, melhor a aproximação do método mas mais recursos como memória e processamento serão necessários. Estas restrições não são relevantes na plataforma de implementação desta dissertação, pois não foram consumidas nem 10% das unidades lógicas da plataforma FPGA utilizada.

Em uma visão macro, a **Figura 1.1** a seguir apresenta o *tradeoff* entre estes diferentes trabalhos relacionados, a fim de fornecer uma verificação diversificada considerando as diversas condições de cada experimento.

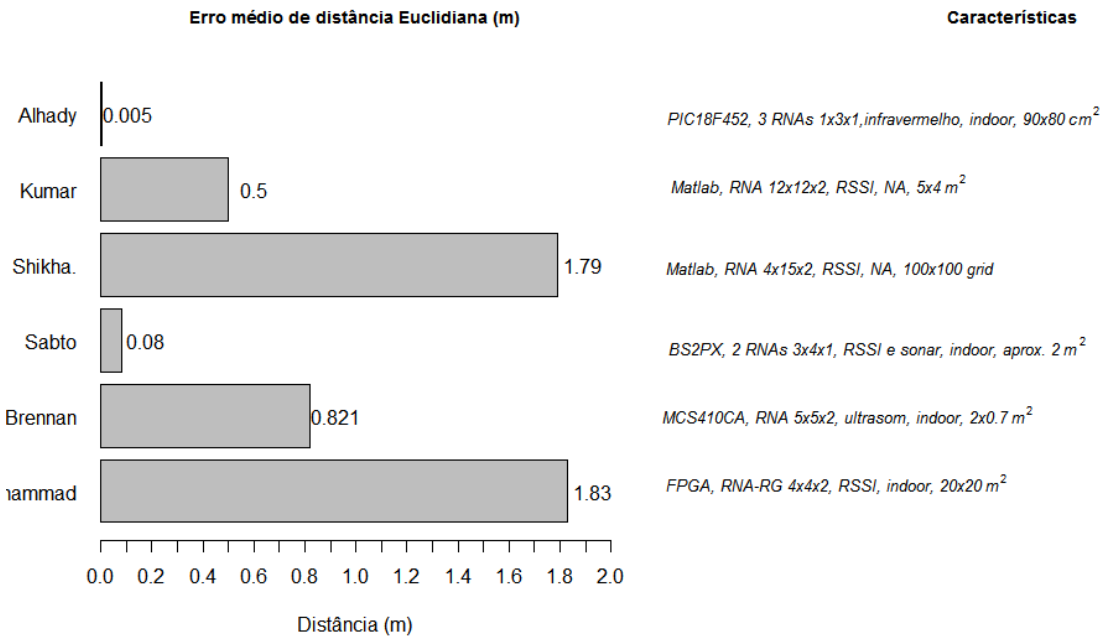

Figura 1.1: Comparação entre diferentes métodos de localização considerando vários fatores de cada experimento.

Diante dos resultados alcançados e das considerações durante o desenvolvimento prático do método, realiza-se a seguir uma discussão sobre as melhorias que foram deixadas para os trabalhos futuros, ou devido ao limite de tempo de desenvolvimento prático do método de localização ou devido as condições limitadas de recursos adicionais. Contudo, concluiu-se que o método é viável quanto a aplicação prática, oferecendo boa estimativa de localização e é comparável com os resultados alcançados em trabalhos recentes encontrados na literatura. Os resultados alcançados nesta dissertação foram publicados em (E SILVA, 2015).

1.4 Justificativa

Os trabalhos correlacionados encontrados utilizam modelos de redes neurais mais complexas ou mesmo combinadas com outros algoritmos e o ganho de desempenho não compensa a complexidade dos métodos. Estruturalmente é possível alcançar, com a mesma rede neural, uma melhor aproximação se trabalhada a informação da base de dados utilizada no treinamento da RNA. Alguns trabalhos se utilizam de outros recursos além da potência do sinal

(ToA, por exemplo) para estimar a posição. Em relação ao custo de implementação, principalmente em aplicações com redes densas e esparsas, esse custo adicional não compensa. Existem muitos trabalhos publicados até hoje que alcançam bons resultados considerando apenas a potência do sinal. Especificamente quanto a localização, alguns trabalhos se utilizam de mais de uma rede neural, uma para cada eixo da coordenada, para calcular a estimativa. Isso devido a utilização de uma plataforma de hardware baseada em micro controladores, os quais possuem baixo poder computacional. Além de aumentar o tempo de resposta, a utilização de mais de uma rede neural exige maior tempo de configuração, considerando treino e preparação da base de dados. Além disso, o erro para cada eixo da coordenada será tratado de forma independente. Alguns trabalhos buscam melhorar os resultados através do aumento desnecessário de neurônios e camadas escondidas na rede neural, contrabalanceando com o custo computacional. Muitos trabalhos encontrados não desenvolvem a implementação do método proposto, deixando a análise dos resultados com base em simulações que não consideram diversos parâmetros reais de aplicações, tais quais a falha de sensores, a atenuação do sinal e ruídos, o tempo de execução do método proposto e a complexidade de implementação considerando a plataforma de hardware. Contudo, a principal contribuição deste trabalho é a utilização de um método simples, capaz de fornecer resultados a partir de dois sensores âncoras. Isso devido ao espaço de respostas do método estar bem definido, não permitindo que o algoritmo forneça respostas (generalize) fora da região de contexto. O método é implementado por uma rede neural artificial com função de ativação Elliott, que aproxima a função log-sigmoide de forma rápida e sintetizável. Além disso, a rede neural utilizada considera o uso mínimo de neurônios e camadas escondidas em função de alcançar uma boa aproximação de resultados. Para suportar a robustez do algoritmo implementado, o método é desenvolvido em uma plataforma recente de FPGA (DE0-Nano) com alta frequência de operação, no entanto, com baixo consumo energético, baixo custo e pequena dimensão. Estas características inerentes à plataforma de desenvolvimento também contribuem para o desempenho do método de localização aqui desenvolvido. A modularidade estrutural do algoritmo, a forma de sua implementação e a capacidade de reconfiguração da plataforma permitem que o método seja facilmente modificado em caso de aperfeiçoamento.

1.5 Objetivos

O objetivo geral deste trabalho é desenvolver um método de localização distribuída para RSSF, energeticamente eficiente e tolerante a falhas e, para tal, utilizar rede neural artificial implementada em um FPGA com baixo custo que atenda às necessidades do algoritmo. Os objetivos específicos desta dissertação são apresentados nos subtópicos a seguir:

- Implementar uma aproximação para a função de ativação sigmoide.
- Avaliar o tempo de resposta do método em função da aproximação do resultado.
- Avaliar a eficiência energética do método.
- Avaliar a tolerância a falhas dos do método.

No próximo tópico a seguir é apresentada a metodologia desenvolvida neste trabalho.

1.6 Metodologia

A metodologia inicialmente consiste em coletar referencial bibliográfico (livros e periódicos) que sobre os temas em RSSF, RNA e FPGA, nos repositórios da IEEE, Sensors, Elsevier, Springer, Capes e banco de Teses e Dissertações.

A análise final é desenvolvida através da avaliação do alcance dos objetivos do trabalho em comparação com os resultados alcançados.

Nas pesquisas em periódicos recentes, foram avaliados os trabalhos que visavam à implementação de RNAs e modelos de neurônios que pudessem ser sintetizáveis em FPGAs, utilizando a linguagem VHDL que é uma das linguagens de descrição de *hardware* que são documentadas e padronizadas pelo IEEE. Alguns trabalhos em referências em repositórios relevantes não mostraram detalhes de implementação, restringindo-se a esquemáticos gráficos dos componentes do design e a mostra de resultados alcançados, o que justificou a necessidade de pesquisas em livros mais técnicos dentro dos tópicos utilizados.

Em seguida, foi definido o ambiente de desenvolvimento necessário para a realização do desenvolvimento da pesquisa. O Software Quartus II da Altera (versões 11, 12, 13 e 14), sob a licença para a Universidade Federal do Pará, é utilizado como ambiente para modelagem da RNA na plataforma reconfigurável. Este ambiente possui aplicativos próprios de análise, verificação e teste dos modelos desenvolvidos. O treinamento da RNA foi desenvolvido utilizando a linguagem R, através do ambiente RStudio. A linguagem R é bastante utilizada no meio de pesquisa na área de aprendizagem de máquina e algoritmos inteligentes, possuindo

diversos pacotes de análise, testes e implementação com algoritmos. A ferramenta RStudio é gratuita e contém muitos recursos que são suficientes para o desenvolvimento desta pesquisa no que diz respeito ao desenvolvimento do algoritmo utilizado. Quanto a escolha dos diferentes modelos de RNA, incluindo a topologia demais características, considerou-se o teorema de aproximação universal (GYBENKO, 1989), (HORNIK, 1991). Optou-se por utilizar o modelo de aproximação mais simples, a RNA multicamadas com retroalimentação, por ser considerado um aproximador universal de funções. A função de ativação foi escolhida considerando a simplicidade da primeira derivada, necessária para o treinamento da RNA no caso de implementação futura. A quantidade de neurônios utilizadas na entrada considera o número de âncoras utilizados no experimento. Na camada escondida, este número de neurônios é repetido sem perdas de desempenho, mantendo a modularidade do desenvolvimento e a reutilização do código implementado. A camada de saída da RNA considera o par ordenado que representa a estimativa das coordenadas de localização do alvo.

O método foi desenvolvido de forma modular, dividindo os principais componentes em módulos independentes e interligados na plataforma reconfigurável. Os módulos consistem em:

- Módulo de comunicação.
- Módulo Controlador.
- Módulo da RNA.

O módulo de comunicação desenvolvido foi a UART, por ser compatível com a maioria (se não todos) dos modelos de comunicação de rádios utilizados em RSSF. Contudo, mesmo operando na sua frequência padronizada mais alta, este modelo de comunicação ainda opera em uma frequência mais baixa do que o FPGA, necessitando de um modulador de frequência, o qual adequa os 1.8KHz da UART para os 50MHz do FPGA. O UART também conta com uma memória adicional para armazenar as mensagens recebias até que o módulo processado processe a mesma. Nos testes práticos, não ocorreu perda de dados por superposição de mensagens recebidas no buffer, pois a velocidade de processamento do módulo controlador implementado foi suficientemente eficiente.

O módulo controlador, além de desempenar as funções de controle de todo o sistema, executa a máquina de estados que define o método proposto.

Os testes experimentais consistem em integrar a plataforma reconfigurável com um módulo de comunicação externo, XBEE padrão IEEE 802.15.4, em uma RSSF, assim como a

plataforma FPGA DE0-Nano. Verificou-se a integridade das informações recebidas entre os sensores da rede, tempo de resposta do módulo da RNA e da máquina de estados no módulo controlador. Os testes foram realizados em um ambiente outdoor nas dependências da Universidade Federal do Pará, em uma área aberta com arborização nas extremidades.

1.7 Estrutura da Dissertação

Este trabalho é composto por cinco capítulos, além deste capítulo introdutório, esta dissertação está dividida em:

Capítulo 2: É o capítulo conceitual do trabalho, resume e classifica o tema pesquisado de acordo com as pesquisas desenvolvidas nesta dissertação. Devido à localização em RSSF ser uma área bastante diversificada, o capítulo começa com uma classificação entre os métodos de localização (centralizada e distribuída), apresentando algumas taxonomias adotadas em RSSF. O capítulo segue mostrando que o uso proficiente dos sensores pode afetar consideravelmente o desenvolvimento de novos métodos de localização, considerando os modos de operação dos sensores, a logística dos dados trafegado entre estes e os protocolos de acesso ao meio utilizados. Em seguida são apresentadas as técnicas fundamentais as quais se baseiam todos os métodos de localização baseados em alcance dos sensores ou em conectividade. Comenta-se sobre os dois modelos comumente encontrados nas referências e que são apresentados como soluções em localização distribuída: matrizes multidimensionais e *cluster*. Alguns trabalhos que utilizam RNA em localização também são comentados no final deste capítulo, contudo, a quantidade de modelos de algoritmos de localização existentes na literatura é muito diversificada.

Capítulo 3: Apresenta a metodologia desta dissertação. Descreve os detalhes de implementação de todo o método, incluindo os módulos de comunicação, de processamento e da RNA. Descreve a máquina de estados e fluxogramas que representam o algoritmo, e a explicação passo a passo do processo de localização implementado. Apresenta como as informações que servem de para treinamento da RNA foram adquiridas e tratadas. Define a RNA em todos os seus detalhes, incluindo a aproximação da sua função de ativação sintetizável no FPGA. Apresenta o módulo UART e sua integração no sistema através do modulador de frequências (*oversampling*) e do *buffer* de armazenamento das informações recebidas. Apresenta o formato das mensagens transmitidas na rede e como estas mensagens são tratadas dentro do módulo processador antes de serem passadas para a RNA. Além disso, o capítulo

apresenta a forma de conexão física entre os pinos do FPGA e os pinos do módulo físico de transmissão (Xbee), os quais compõem a implementação física do sistema.

Capítulo 4: Apresenta os resultados alcançados. OS resultados começam com a definição da RNA e seus parâmetros livres, gerados durante o treinamento na ferramenta RStudio. Os detalhes do método de treinamento são apresentados nesta parte. Também é apresentado o esquemático da descrição final de todo o sistema implementado em VHDL, incluindo as conexões lógicas entre todos os módulos implementados. Nos testes realizados, considera-se a falha conjunta do terceiro e quarto sensor âncora, a falha apenas do quarto âncora e por fim, nenhuma falha dos sensores âncoras. Dessa forma, três resultados diferentes são considerados nos experimentos e três erros são calculados. Além dos erros relacionados as falhas consideradas nos sensores âncoras, adicionamos o erro relativo a discretização dos valores dos parâmetros livres da rede neural, os quais são gerados devido a conversão real para a representação binária. Apresenta-se um diagrama da implementação do neurônio baseado na função aproximada para a função de ativação de forma a verificar visualmente como o processamento paralelo ocorre. A mesma visualização é apresentada para toda a RNA. Em seguida, um gráfico com os resultados dos testes que incluem as falhas dos sensores é apresentado, podendo ser visualizado como o método se comportou para alguns casos de localização. Sem perda de importância dos trabalhos relacionados, realizamos uma breve comparação entre os erros alcançados, visando mostrar que o método proposto nesta dissertação apresenta resultados melhores ou alinhados com as referências.

Capítulo 5: Apresenta as considerações finais, incluindo comentários acerca do desenvolvimento e possíveis melhorias que deixamos para desenvolver em trabalhos futuros

2 LOCALIZAÇÃO EM RSSF

Localização em RSSF é um processo fundamental para detecção da origem de eventos na região assistida pelos sensores da rede, para assistir a grupos de sensores, para decidir estratégias de roteamento eficientes ou mesmo para avaliar o comportamento do efeito físico sobre a região monitorada (PAL, 2010). A pesquisa em localização é associada a aleatoriedade associada à implantação dos sensores, característica esta, intrínseca na maioria das aplicações. Também, em algumas aplicações, os sensores não são despejados aleatoriamente mas passam a operar como alvos móveis após o início de operação, sendo necessário o seu rastreamento.

O processo de localização em RSSF é, geralmente, denominado pelo termo localização consciente ou localização conhecida (*location-aware*). A capacidade de os sensores "saberem" a sua posição na rede é a base para as principais aplicações em RSSF. Muitas propostas de métodos são apresentadas em trabalhos que envolvem localização em RSSF. Esses métodos se dividem primariamente nos que utilizam informações de distância (tempo, frequência, ângulo ou intensidade do sinal propagado) e nos que se baseiam em conectividade entre os sensores, como por exemplo, o modelo *likelihood*.

Dependendo do papel desempenhado pelos sensores, alguns nomes podem ser utilizados para referenciá-los contextualmente dentro de determinada topologia lógica de rede. Quando classificamos as RSSF pela sua topologia lógica, é necessário que os sensores da rede assumam papeis específicos dentro do contexto em que são inseridos. Em geral, os papeis que podem ser assumidos são:

- Sensor coordenador.
- Sensor roteador.
- Sensor coletor.
- Sensor *sink* (termo do inglês que significa escoar, despejar) ou *gateway*.

Os sensores também são nomeados com base no conhecimento ou não da sua posição dentro do ambiente monitorado. Se o sensor conhece a sua localização, este recebe o nome de sensor âncora (*anchor* ou *beacon node*, nas referências em inglês). Quando o sensor não "conhece" a sua posição no ambiente monitorado, este recebe o nome de sensor alvo (ou *target node*, do inglês).

Tabela 2.1

: Classificação dos sensores com base na topologia (a esquerda) e com base na localização (a direita.)

Classificação no contexto de topologia
coordenador
roteador
sink ou *gateway*

Classificação no contexto de localização
âncora ou *beacon*
alvo ou *target*

O coordenador possui diversas responsabilidades na rede, tais como a de iniciar as configurações prévias dos outros sensores, tais como endereços de rede, além de realizar demais configurações em nível de gerenciamento de rede. Um coordenador pode rotear e/ou coletar as informações (assim como os roteadores e coletores) se possuir capacidade para tal. Em RSSF que utilizam o padrão de comunicação IEEE 802.15.4, apenas um coordenador pode ser necessário para gerenciar a rede ou cada *cluster* (subdivisões contidas na topologia da rede).

O roteador, como se subentende pelo nome, é responsável pelo roteamento de informações entre os sensores na rede. Assim como o coordenador, dependendo dos recursos de fábrica, o roteador pode desempenhar a tarefa de coleta de informações do meio, além de poder realizar tarefas limitadas de gerenciamento de rede, tal como "acordar" (do inglês, *wakeup*) os sensores coletores que possam estar em algum modo de operação definido para latência. Um tipo especial de RSSF onde todos os sensores operam potencialmente como roteadores recebe a denominação de RSSF *ad hoc*. Apesar de redes *ad hoc* estarem comumente associadas à topologia lógica *mesh*, esses tipos de rede podem ser arranjados em outras topologias, dependendo da aplicação.

O coletor é responsável por realizar o sensoriamento da região dentro do seu alcance de sensitividade, pré-processar informações (não obrigatoriamente) e transmitir estas para um roteador ou coordenador geralmente até um *sink*. O coletor é de fundamental importância, pois toda a análise sobre a aplicação depende das informações coletadas por ele. Coletores em geral possuem capacidades reduzidas para atender um problema de aplicação em específico, sendo quase sempre dispostos aleatoriamente dentro de uma região monitorada. Por serem o foco das aplicações em RSSF no contexto de localização, os coletores recebem o nome de alvo. Para que as informações adquiridas pelos coletores possam fazer algum sentido durante a análise de

dados, é necessário fundamentalmente que os locais de coleta dessas informações sejam conhecidos.

O último papel que pode compor a RSSF é o s*ink* ou g*ateway*. O nome *sink* é utilizado em grande parte dos trabalhos sobre RSSF para referenciar um sensor que realiza a interface entre a RSSF e uma (ou mais de uma) estação base com maior poder de processamento e recursos onde se podem realizar as análises dos dados coletados da rede. O s*ink* é responsável por levar a uma estação base todas as informações roteadas na rede relevantes para análise do problema abordado pela RSSF.

Ainda quanto à topologia de rede, outros papéis podem ser assumidos pelos sensores, mas os apresentados anteriormente são considerados os fundamentais. Por exemplo, um sensor pode assumir o papel de *cluster head*. Sensores c*lusters heads* podem agregar as informações de um *cluster* de sensores na rede e transmitir estas a um *sink*.

Em geral, os papéis citados anteriormente para sensores são suficientes para descrever os mesmos dentro de uma topologia lógica na maioria das aplicações.

A seguir, as técnicas fundamentais as quais os métodos de localização se baseiam, considerando os sensores citados anteriormente.

2.1 Técnicas Fundamentais

Nesta sessão serão apresentadas as técnicas AoA, ToA, TDoA e RSSI, as quais são baseadas na distância, no tempo ou na angulação (técnicas baseadas em alcance – *range based*) dos sinais transmitidos pelos sensores para realizar o procedimento de localização. Em seguida, técnicas baseadas em conectividade por adjacências, que por sua vez são classificadas como r*ange-free*.

A. Angle of Arrival

A técnica AoA (CHAN e WEN, 2011) considera a direção de chegada do sinal transmitido pelos âncoras, obtidas a partir da amplitude ou da fase do sinal da antena localizada no alvo e comparada com uma orientação referencial dos âncoras. As restrições envolvendo este método são a necessidade de direcionamento da antena dos sensores e problemas de reflexão do sinal, que ocasiona na multiplicidade (*multipath*) de sinal recebido. Esta multiplicidade afeta a estimativa de localização do alvo. Para utilizar AoA, os sensores precisam adicionalmente serem equipados com uma antena inteligente (*smart antenna*) que é composta por várias minis antenas (ou *array* de antenas) que detectam o sinal transmitido com diferentes fases ou

amplitudes. De posse destas informações, AoA utiliza relações geométricas (triangulação) para estimar a posição do alvo (BOUKERCHE, 2007). A Figura **2.1** ilustra o funcionamento da técnica AoA.

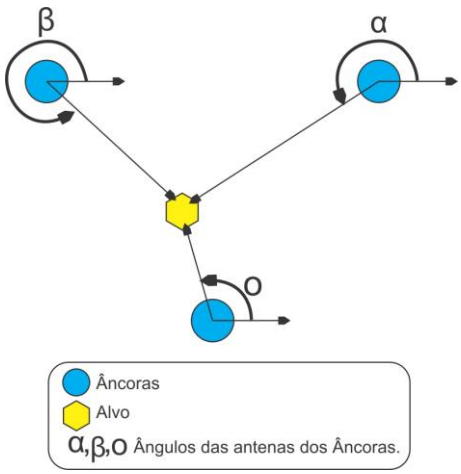

Figura 2.1: Ilustração da técnica AoA para a Localização.

Na Figura **2.1** um alvo faz a requisição de pelo menos três âncoras à angulação do seu sinal recebido por estes. Os três âncoras devolvem ao alvo a angulação do sinal, de acordo com o mecanismo de detecção empregado pela antena. Em seguida, o alvo utiliza a triangulação para estimar a sua posição relativa.

B. Time of Arrival

Esta técnica utiliza-se do tempo de propagação do sinal entre o alvo e os âncoras para, em seguida, realizar o cálculo de posição através de multilateração (HSU, 2011). A técnica ToA considera apenas uma transmissão entre o alvo e os âncoras (RAVINDRA e JAGADEESHA, 2013). Esta técnica demanda sincronia entre os sensores, ou seja, todos os sensores envolvidos no processo de localização devem conhecer o tempo exato em que o processamento do tempo de chegada do sinal será computado. A Figura **2.2** representa o modelo ilustrativo da técnica ToA, onde as circunferências simbolizam o sinal propagado a partir de cada âncora.

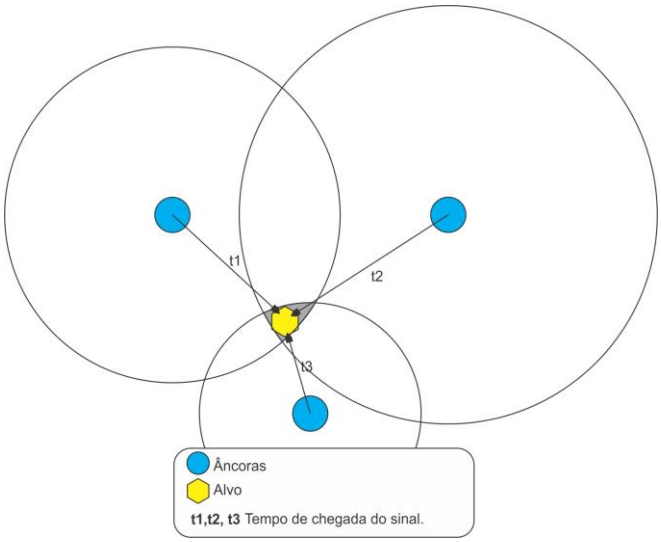

Figura 2.2:Ilustração da técnica ToA para a Localização.

A área de interseção das circunferências que representam os sinais propagados é a região em que a técnica ToA estima estar localizado o alvo. Quanto maior o número de âncoras utilizado nesta estimativa, menor será esta área de interseção e mais precisa será a resposta do algoritmo.

Uma das formas de se calcular a posição através da técnica ToA é apresentada a seguir, deduzida de (RAVINDRA e JAGADEESHA, 2013).

$$\boldsymbol{r} = \frac{[F(\mathrm{x}) + \boldsymbol{n}]}{c} \tag{1}$$

Na **Equação 1**, ***r*** é um vetor de medidas de tempo de propagação do sinal entre o alvo e os âncoras. O par ordenado x = (a, b) é a posição cartesiana do alvo a ser estimada, passada a uma função linear *F(.)*, a qual calcula a distância (Euclidiana, por exemplo) com base em parâmetros conhecidos – os pares ordenados que indicam as posições dos âncoras. O ***n*** é um vetor de ruídos de média zero. Por fim, *c* é a constante de velocidade de propagação do sinal no meio.

C. Received Signal Strength Indicator

Métodos de localização baseados em RSSI são bastante utilizados para estimar a posição dos sensores (ZHOU, ZHAO e TAN, 2013). O RSSI é um índice em uma escala genérica baseado na potência do sinal propagado do emissor até o alvo, ou vice versa, pois a potência do sinal, para a mesma mensagem transmitida, idealmente é a mesma. Alguns rádios convertem a potência do sinal recebido (RSS, geralmente em –dBm ou dBm) em um valor hexadecimal ou inteiro sem sinal, ou mesmo outra escala. O valor em dBm convertido em outra escala é

denominado RSSI. O RSS é afetado pelas condições ambientais do cenário onde é mensurado. Existem diversos modelos de propagação de sinal para o RSSI, os quais tentam descrever a relação do sinal propagado com a distância do mesmo entre os dispositivos na comunicação, considerando fatores ambientais (XU, 2010).

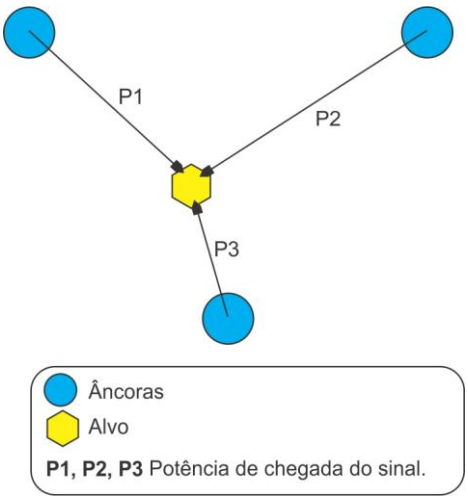

Figura 2.3:Ilustração da técnica RSSI para a Localização.

Na Figura **2.3** acima, a potência do sinal recebido $P_r(dBm)$ pode ser convertido em distância d através do modelo descrito na **Equação 2** a seguir.

$$P_r(dBm) = P_t - 10\gamma \log_{10}({}^{d}/_{d_n}) + \psi_{dBm} \quad (2)$$

onde P_t é a potência do sinal transmitido em dBm, γ é a perda que o sinal sofre no meio de propagação devido as condições ambientais. Em seguida, e d_n é a distância de referência do entre o sensor alvo para o *enésimo* sensor âncora. O parâmetro ψ_{dBm} é o efeito sombra definido como uma variável aleatória Gaussiana. O parâmetro d_n é calculado através pela **Equação 3** a seguir:

$$(x_n - x)^2 + (y_n - y)^2 = d_n \quad (3)$$

onde x_n *e* y_n são a coordenada do sensor âncora *n* para o sensor alvo localizado na coordenada (x, y).

Alguns dispositivos já fornecem um método documentado para calcular a potência do sinal recebido. Geralmente, o RSSI pode ser lido por estes dispositivos através de uma porta digital.

2.2 Localização centralizada e distribuída

As técnicas fundamentais são aplicadas em duas classificações dentro do contexto de localização: localização distribuída e localização centralizada. Esas classificações são consideradas juntamente com as técnicas fundamentais durante o desenvolvimento dos algoritmos de localização.

Em localização centralizada as informações necessárias para realizar o processo de localização dos alvos são todas direcionadas à um sensor central na rede ou estação base, o qual possui maior poder de processamento e suficiência energética, em comparação com os sensores e é capaz de realizar o processamento do método de localização. Na vertente centralizada, a quantidade de transmissões necessárias pode causar alto consumo energético por parte dos sensores que participam do procedimento e comprometer a rede quando ocorre a falha de um dispositivo *gateway* utilizado na rota de transmissão. Em RSSF densas com média à larga escala, as técnicas centralizadas podem comprometer o tempo de vida útil da rede devido a estas restrições, através da criação de *hot spots* (sensores com elevado tráfego de informações).

Em localização distribuída o objetivo é diminuir os contras da localização centralizada (ALHMIEDAT, SALEM e TALEB, 2013), (FAN, ZHANG e DAI, 2015), através da divisão do processamento do problema de localização entre os sensores, sejam entre *clusters* ou entre diversos âncoras e alvos na rede, evitando o volume de tráfego de informações sobre um único sensor e a centralização do método em um único ponto. Idealmente, espera-se que o método de localização distribuída possa estimar a posição do alvo mesmo na presença de falhas de outros sensores que participam do procedimento de localização, por conta do processamento estar distribuído em diferentes partes da rede.

A seguir serão brevemente apresentados os métodos de escalonamento multidimensional e de *clustering*, muito empregados em localização distribuída. Em seguida, também será comentado métodos de localização que utilizam inteligência artificial.

A. Algoritmos baseados em escalonamento multidimensional

O método de escalonamento multidimensional (e suas variantes) é comumente empregado para redes médias e densas, tanto para localização centralizada quanto para localização distribuída. O MDS (*multidimensional scaling*) baseia-se na distância Euclidiana ou conectividade (número de sensores adjacentes) entre todos os alvos para gerar um mapa da rede

ou matriz de conectividade/distância, aplicando em seguida, técnicas geométricas para estimar a posição dos sensores. Para diminuir a baixa acurácia associada ao método MDS, mais de uma matriz de localização pode ser utilizada e uma média pode ser aplicada nestas matrizes. As variações do MDS incluem métodos de relaxamento, assim como técnicas de otimização como *Levenberg-Marquardt*, algoritmo genético e *simulated anealing* que podem ser empregadas, mas que são custosas e complexas para serem implementadas de forma a serem distribuídas entre os sensores. Se o MDS considera a conectividade entre os alvos (MDS não métrico), métodos de grafos, tais como Dijkstra ou Floyd, podem ser utilizados para contabilizar as distâncias com base em sensores adjacentes. Quando o MDS é métrico, as técnicas tradicionais (RSSI, AoA e ToA) são empregadas.

Uma desvantagem do MDS é a necessidade de informações *a priori* quanto à localização de alguns alvos na rede para que se possam estimar novas posições, quando necessárias. Essa desvantagem é evidenciada em aplicações onde os sensores alvos são despejados de forma aleatória.

B. Algoritmos baseados em clusters

Nesta abordagem os alvos da rede são agrupados em clusters, onde um dos alvos assume o papel de *cluster head*. Os *cluster heads* agregam informações de localização dos alvos adjacentes e as transmitem a estação base mais próxima. Em alguns casos, o cluster head envia apenas as coordenadas calculadas para o alvo à estação base, evitando um volume excessivo de informações na rede. Apesar de alguns trabalhos considerarem esta abordagem como localização distribuída (ALHMIEDAT, SALEM e TALEB, 2013), o processamento da posição do alvo ainda ocorre em um único ponto da rede (no *cluster head*). Em versões de algoritmos que não possuem uma nova seleção de *cluster heads*, no caso de falhas do primeiro, o procedimento de localização tende a falha.

C. Algoritmos inteligentes

Localização utilizando algoritmos inteligentes é uma alternativa às técnicas geométricas (e técnicas de conectividade) supracitadas neste capítulo, utilizadas para aproximar a posição de um determinado dispositivo na rede. Um dos algoritmos inteligentes, as RNAs, também se utilizam das informações adquiridas relacionadas à distância (potência do sinal e tempo de propagação, por exemplo), à angulação e a conectividade entre os sensores da rede para aproximar a localização de um alvo. A vantagem principal de se utilizar RNA é o ganho com a

precisão e acurácia nos resultados diante da ausência de informações durante a operação do método, graças a capacidade de se criar um modelo complexo baseado na aproximação de uma função não linear. No entanto, a desvantagem pode estar associada à quantidade de informações necessárias na configuração prévia do método, para aumentar a precisão e acurácia dos resultados durante o treinamento supervisionado da rede. Este pré-processamento pode ser custoso quanto ao tempo de execução. No entanto, o avanço tecnológico em nível de software e de hardware contribui continuamente para a diminuição destas desvantagens, favorecendo a utilização de RNA sobre as técnicas geométricas originalmente empregadas. Na literatura, encontram-se diversos trabalhos que utilizam diferentes algoritmos inteligentes para realizar o processo de localização em RSSF, mas raramente usando FPGA como uma plataforma aplicada (PRASAD, 2015), (CHO, 2014). A seguir, apresenta-se o uso proficiente dos sensores e da rede aplicados nos métodos de localização.

2.3 Uso proficiente dos sensores e da rede

Para prolongar a vida útil dos sensores, em nível de energia e, consequentemente, prolongar a vida útil das aplicações, as técnicas de localização devem estar fortemente associadas ao consumo de energia de tais sensores. Quanto menor for a quantidade de transmissão de informação na rede, menor é o consumo energético dos sensores. Outros fatores também podem restringir os algoritmos de localização, tais qual o limite de alcance do sinal dos sensores LoS (*line of sight* – linha de visada) e a falta de recursos adicionais disponíveis nos sensores (mais de um tipo sinal: ultrassom, rádio, infravermelho).

Grande parte dos sensores alvos possuem bateria com carga limitada que, em geral, é muito difícil trocá-la ou recarregá-la. Quanto à troca de bateria, esta dificuldade está associada com a maioria das aplicações, onde estes sensores são depositados em áreas hostis ou inóspitas. Outro impedimento quanto à troca de bateria é a quantidade de alvos que podem ser despejados em um ambiente, principalmente em aplicações de larga escala. Quanto à recarga de bateria, que comumente é realizada por energia solar ou cinética e consequentemente necessitando de hardware adicional, pode ser custoso dependendo da quantidade de dispositivos ou mesmo da tecnologia empregada. A recarga por energia solar também exige que a aplicação ocorra obviamente em ambiente externo, restringindo o uso em ambientes internos sem acesso a esta fonte de energia. A recarga por energia cinética demanda movimento e é possivelmente

aproveitada em aplicações que envolvam rastreamento (se houver movimento) - ainda sim, uma tecnologia custosa para implantação.

Existem três abordagens principais, consideradas durante o desenvolvimento dos algoritmos de localização: modos de operação dos sensores, fluxo de dados e protocolos de acesso ao meio. Na prática, as abordagens se mesclam durante a elaboração do método de localização devido às diversas (possíveis) complexidades associadas às aplicações.

Modo de operação (do inglês, *duty-cycling*) define-se como o tempo em que um sensor está em modo de atividade durante sua operação na rede. Uma abordagem considera a redundância de funcionalidade de grupos de dispositivos na rede, os quais podem operar em alternância. Outra abordagem considera alternar os modos de operação individualmente nos sensores em intervalos predefinidos, assumindo que se conheça o tempo certo em que a aplicação exige operação.

O grupo de sensores que garantem a conectividade em toda a rede são denominados aqui como grupo de controle de topologia. O controle de topologia gerencia os outros sensores ativos quanto ao modo de operação, sendo que, nestes restantes, o *duty-cycling* opera para economia de energia na rede. Os sensores que não compõem o grupo controle de topologia serão classificados aqui como sensores de operação.

Quanto ao modo de operação dos sensores, uma classificação adequada (apresentada anteriormente) define os estados como ativo (transmissão, recepção ou processamento), inativo (*idle*) ou dormência (*sleep*). Se os sensores dependem (LAI, 2010) uns dos outros em determinada aplicação (algoritmos de localização, por exemplo), pode-se considerar que estes alternem entre estes modos com a mesma frequência. Para evitar colisão de pacotes de dados com pacotes de controle de modo de operação (problema de conectividade), adotam-se a utilização de canais diferentes para estes fins. Um trabalho mais aprofundado sobre *duty-cycling* pode ser encontrado em (LAI, 2010), (GHADIMI, 2012).

Um inconveniente de se considerar esta abordagem durante o método de localização distribuída surge da necessidade da descoberta de sensores adjacentes, seja para decisão de roteamento ou para outra funcionalidade intercooperativa. Neste aspecto, há a falta de garantia de que sensores adjacentes estejam em modo ativo ao mesmo tempo para que haja a troca de informação. Apesar de classificarmos esta abordagem como um problema topológico, claramente os modos de operação podem afetar também o desempenho da conectividade da

rede e consequentemente os métodos de localização. Os desafios neste paradigma podem ser resumidos como:

- Mudanças de modos de operação síncrona e por demanda.
- Mudanças de modos de operação síncrona e de forma escalonada.
- Mudanças de modos de operação assíncrona.

As métricas de avaliação que podem ser consideradas no desenvolvimento dos métodos de localização podem ser baseadas na latência dos sensores, capacidade de transmissão de pacotes, taxa de recebimento de pacotes, tempo de vida dos sensores e pelo consumo de energia versus operações na rede (HAN, 2013).

O termo fluxo de dados em RSSF está relacionado ao roteamento de dados entre os sensores na rede (FRERY, 2010). Os métodos de localização que consideram o fluxo de dados dão ênfase ao fluxo de dados da rede quando abordam determinado problema, diferentemente de técnicas baseadas em *duty-cycling*.

Considerando conservação de energia, métodos de localização procuram diminuir ao máximo a necessidade de transmissão de dados na aplicação (PATHAK e PRASANNA, 2010). Basicamente os métodos de localização fundamentados em fluxo de dados procuram por padrões durante o consumo de energia na rede, e aproveitam as oportunidades encontradas nestes padrões para desenvolver estratégias para economia de energia na transmissão de pacotes. Implicitamente, a coleta e agregação de uma grande quantidade de dados são necessárias para a busca desses padrões, acentuando para algumas aplicações a relevância da utilização de uma base de dados para lidar com este problema.

Existem (FRERY, 2010) métodos de localização baseados em *clustering* com fusão de informação, os quais resumem os desafios considerados com essa técnica, resumidos como:

- Como os efeitos físicos a serem coletados se apresentam na área (aplicação).
- O modo como os sensores são situados e a técnica de sensoriamento destes efeitos físicos.
- Como os sensores são agrupados logicamente para formar *clusters* ou apenas considerando a topologia de rede.
- Como os dados são arranjados em cada *cluster* ou topologia lógica, antes de serem enviados a um *sink*.

- Como a estação base agrega (base de dados) estes dados para gerar informação sobre a aplicação.

Por conseguinte, as métricas de avaliação de desempenho dos métodos de localização que utilizam esse paradigma podem ser desenvolvidas com base na Teoria de Shannon-Nyquist (NORDIO, CHIASSERINI e MUSCARIELLO, 2008), (SUNG, POOR e YU, 2009).

Protocolos MAC são métodos que definem políticas de acesso ao meio de transmissão de dados na rede. Quanto aos métodos de localização que consideram os protocolos MAC, pode-se classificar (REZAEI, ZAHRA e MOBININEJAD, 2012) (SAHOO, RATH e PUTHAL, 2012) como sendo:

- Métodos baseados em contenção (disputa) do meio de transmissão.
- Métodos baseados no escalonamento do tempo de acesso ao meio.
- Métodos híbridos contendo os dois métodos anteriores.
- Métodos definidos em mais de uma camada de rede.

Métodos de localização baseados no protocolo MAC CSMA (*carrier sense multiple access* - acesso múltiplo com sensoriamento de portadora) (SINGH e BISWAS, 2012) são à base dos protocolos baseados em concorrência do meio de transmissão, onde não existe a coordenação de acesso a este meio. Dessa forma, os sensores competem para acessar o canal de comunicação. Uma versão do protocolo CSMA com detecção de colisão (CSMA/CD) permite que os sensores que transmitem pacotes ao mesmo tempo possam retransmiti-los em um tempo aleatório. Algumas das versões de protocolos baseados em CSMA que são empregadas em métodos de localização em RSSF podem ser vistas na Tabela **2.2** a seguir.

Tabela 2.2: Protocolos que podem ser adotados nos sensores em RSSF.

Protocolos baseados em CSMA	Sensor Protocol (SMAC), Timeout Protocol (TMAC), Berkley Protocol (BMAC), Dynamic Protocol (DMAC), X Protocol (XMAC), WiseMAC Protocol, Power Control Sensor Protocol (PCSMAC), Cognitive Protocol (CMAC).
Protocolos baseados em TDMA	Dynamic Energy Efficient Protocol (DEEMAC), Slot Periodic Assignment for

	Reception Protocol (SPAR), Traffic-Adaptive Medium Access Protocol (TRAMA).
Protocolos híbridos (CSMA e TDMA)	IEEE 802.15.4, PTDMA, Energy Efficient Protocol (μ-MAC), SCPMAC, YMAC, Zebra Protocol (ZMAC), Advertisement Based Protocol (AMAC).
Protocolos de múltiplas camadas de rede	SCS Protocol.

Métodos de localização baseados em MAC TDMA (*time division multiple aAccess* - múltiplos acessos por divisão de tempo), diferentemente dos modelos baseados em CSMA, associam um intervalo de tempo específico para casa dispositivo para acessar o canal de comunicação, evitando, desta forma, a colisão de pacotes. Alguns exemplos deste protocolo podem ser vistos na Tabela **2.2**. Métodos de localização baseados em TDMA evitam que ocorra a colisão de pacotes e, portanto, a retransmissão de informação durante o processo de localização.

Alguns métodos de localização consideram protocolos híbridos, isto é, que utilizam as vantagens de ambos CSMA e TDMA. Possuem como a principal característica a de utilizar um canal com detecção de colisão para pacotes de controle de tráfego e outro canal com divisão de tempo de acesso para pacotes de dados. Atualmente, o protocolo híbrido IEEE 802.15.4 é comumente adotado em RSSF (assim como o protocolo *zigbee*) e os métodos de localização que utilizam este padrão podem abstrair os detalhes de protocolos MAC, concentrando-se no número de pacotes transmitidos e/ou quantidades de âncoras no processo de localização.

2.4 Taxionomias para Localização em RSSF

Baseado em tudo que foi visto até aqui, esta seção discuti três taxionomias para classificar os métodos de localização, duas que são retiradas da literatura e uma terceira que está mais diretamente relacionada a técnica desenvolvida nesta dissertação.

Existem muitos métodos de localização em RSSF e vários trabalhos buscam classificá-los por conta desta diversidade. Essas classificações serão apresentadas na forma de taxonomias, com o objetivo de situar o contexto do problema na área de pesquisa abordada.

Em (CHENG, 2012), os métodos de localização são classificados entre métodos que utilizam a distância, o ângulo, à frequência e o tempo de propagação dos sinais entre os sensores e métodos que utilizam a conectividade e outros padrões de tráfego de informações associados na rede, sejam eles, calculados no próprio dispositivo ou em sensores externos. A **Figura 2.4** apresenta a classificação sugerida no referido trabalho.

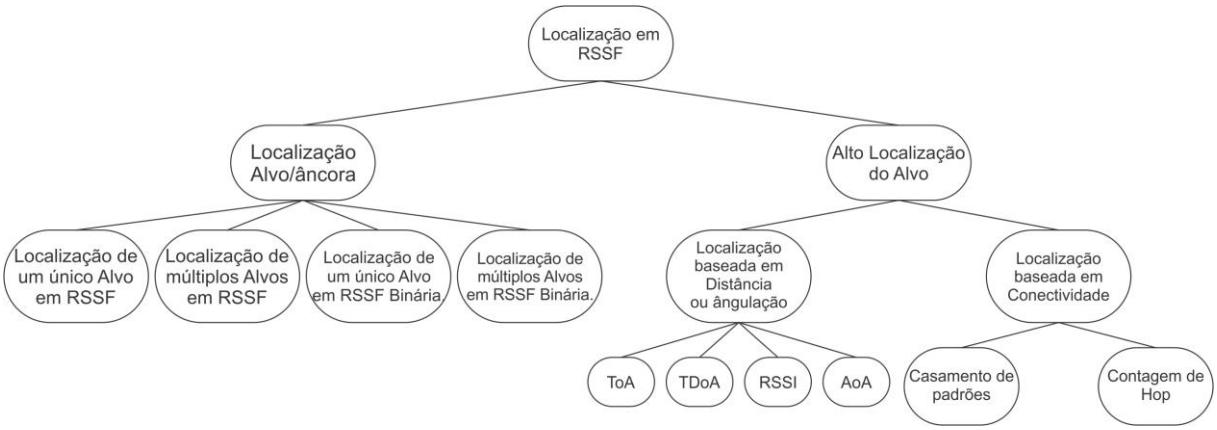

Figura 2.4: Taxonomia do processo de localização **(CHENG, 2012)**.

Na classificação da Figura **2.4**, existem alguns pontos a se ressaltar em RSSF binárias, as quais utilizam sensores âncora(s) e alvo(s) nos métodos de localização. O adjetivo "binário" está relacionado ao fato de que os sensores da rede são capazes apenas de realizar a detecção (e não detecção) de sensores adjacentes, sem a habilidade de mensurar o quão próximo (ou qual a direção) estes sensores adjacentes estão de sua localização. Em localização âncora(s) /alvo(s) para múltiplos alvos, métodos baseados em estimativas de sensores adjacentes são empregados, como por exemplo, o estimador de máxima verossimilhança (MLE - *maximum likelihood estimator*). Todos os processos de localização alvo/âncora utilizam fundamentalmente os processos apresentados nas "folhas" da sub árvore "Alto Localização do Alvo" da Figura **2.4**.

A diferença entre "Localização Alvo/Âncora" e "Auto localização do Alvo" está na participação dos âncoras no processamento de localização. A classificação apresentada na Figura **2.4** está fortemente atrelada ao modo de detecção por parte dos sensores. Essa característica está ressaltada devido ao fato de que todas as técnicas de localização existentes até o momento são baseadas em distância, angulação, tempo, frequência ou conectividade ou uma combinação destas. Outra taxonomia que classifica o processo de localização em RSSF

será ilustrada na Figura **2.5**, onde considera-se uma outra taxonomia que classifica a localização em distribuída ou centralizada.

Figura 2.5: Taxonomia que considera a localização distribuída e centralizada **(PAL, 2010)**.

Com o avanço tecnológico (VLSI), o custo computacional nos sensores passa a consumir cada vez menos energia e a velocidade de processamento tende ao aumento. Em (PAL, 2010) são citadas versões centralizadas das técnicas MDS, *simullated annealing* e baseadas em RSSI como exemplo. Quanto a localização distribuída em (PAL, 2010), as técnicas baseadas em âncoras são derivadas das técnicas baseadas em conectividade ou distância, descritas anteriormente neste capítulo. As técnicas baseadas em relações também consideram distância ou conectividade e as posições iniciais dos alvos são inferidas heuristicamente e um refinamento é realizado com base na interação entre sensores adjacentes. Quanto a técnicas baseadas em clusters, estas utilizam métodos de grafos para representar também a distância ou conectividade entre sensores. Diferentemente do que ocorre com as baseadas em âncoras ou relações, as técnicas baseadas em grafos não consideram sensores como âncoras ou alvos, mas os sensores da região de localização na rede como sendo nós de um grafo e a relação entre estes como sendo os vértices. A partir daqui, (PAL, 2010) classificam os métodos de localização através de técnicas mais sofisticadas e menos genéricas, como a abordagem híbridas, onde os autores apresentam trabalhos relacionados que utilizam combinação de diversas técnicas, tal como o MDS combinado com mapa baseado em proximidades ou com o sistema de posicionamento *ad-hoc*. Também quanto a abordagens híbridas ressaltam o uso combinado de

técnicas indutivas e dedutivas as quais respectivamente representam métodos baseados em propriedades físicas (distância, conectividade) e métodos que necessitam de um treinamento ou configuração prévia (rede neural, por exemplo). Qualquer outra técnica mais sofisticada e menos comum poderia ser classificada como uma técnica híbrida no contexto da taxonomia de (PAL, 2010). A duas classificações seguintes são relacionadas a propagação do sinal entre os sensores. Uma baseada em interferência entre frequências, que utiliza a interferometria entre dois sinais com frequências relativamente próximas que geram uma interferência que pode ser mensurada e, a partir daí, inferir a posição do alvo. A última classificação é baseada em propagação de erro: inclui técnicas baseadas nos modelos de propagação de sinal, considerando erros de propagação no meio de transmissão.

Existem diversos métodos de localização distribuída apresentados na Figura **2.5** e a classificação apresentada por (PAL, 2010) pode ser considerada bastante exaustiva e complexa. Todos os exemplos compartilham do fato de que o processamento (ou parte deste) para estimar a posição de alvos se dá em mais de um ponto na rede evitando, assim, que grande parte do volume de informação seja concentrada ou dependente de um único ponto na rede. Essa é a principal característica da localização distribuída.

Nesta dissertação, apresenta-se uma taxonomia simplificada baseada nas referências quanto a localização em RSSF.

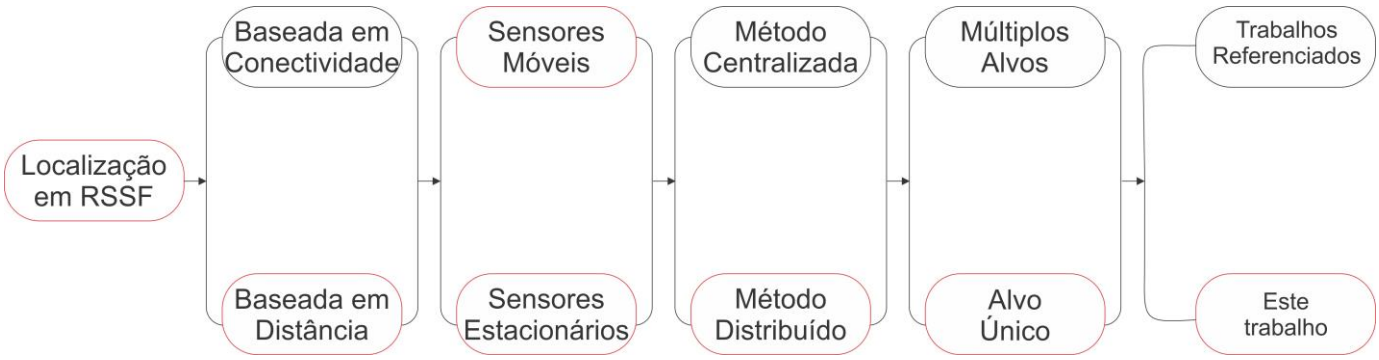

Figura 2.6: Taxonomia simplificada apresentada nesta dissertação.

Figura **2.6** pode-se classificar o trabalho desta dissertação como um processo de localização baseado em distância, considerando sensores móveis ou estacionários, aplicando um método distribuído para localizar um único sensor, onde este método implementa uma rede neural artificial.

3 Desenvolvimento do Método de Localização Distribuída

3.1 Introdução.

O método desenvolvido consiste no uso de quatro âncoras com posições absolutas conhecidas, dispostos em torno de uma área *outdoor* retangular medindo 20x22m^2, cooperando para estimar a posição relativa (à estes quatro âncoras) de um alvo posicionado aleatoriamente dentro desta área. A Figura **3.1** apresenta a área onde o experimento foi realizado.

Figura 3.1: Local de realização do método.

Na Figura **3.1**, as setas em branco apontam para a localização dos âncoras. De início, o alvo envia uma única mensagem em *broadcast*, a qual é recebida pelos quatro sensores âncoras. Os quatro sensores âncoras são identificados por um ID (identificador) numérico e compartilham em sequência adjacente, a partir do primeiro âncora, o indicador de potência RSSI do sinal *broadcast* proveniente do alvo. Em seguida, três dos âncoras estimam a localização do alvo através destes indicadores de potência, os quais são passados como parâmetros para a rede neural definida na plataforma FPGA de cada sensor âncora.

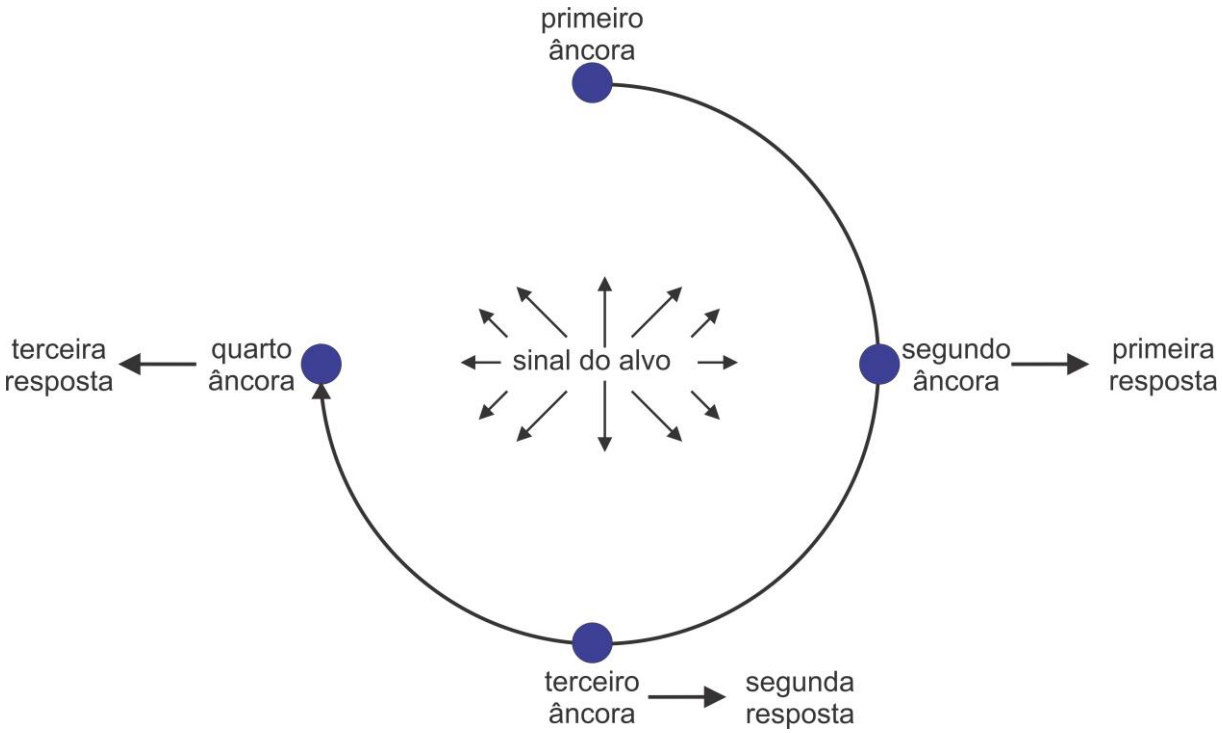

Figura 3.2: Descrição do processo de localização.

Na plataforma FPGA em cada âncora, o método de localização é definido conforme o diagrama de blocos na Figura **3.3**.

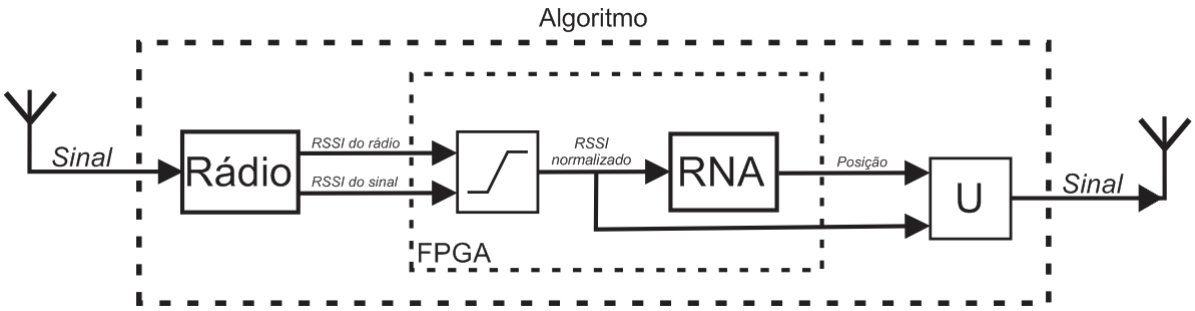

Figura 3.3: Diagrama de blocos do método executado nos sensores âncoras que estimam a posição.

A Figura **3.3** representa o método de localização implementado em cada um dos sensores âncoras que participam do processo de localização. O sinal (lado esquerdo) representa a mensagem em *broadcast* do alvo o qual chega nos âncoras. O rádio em cada dispositivo âncora toma o RSSI da potência do sinal da mensagem e o condiciona para um intervalo normalizado para a faixa de valores de potência do rádio Xbee. A partir de então, os sensores âncoras, com

exceção do primeiro âncora, aguardam as informações dos âncoras antecessores, respeitando a sequência ordenada.

O primeiro âncora não vai realizar a estimativa com a rede neural, já que possui apenas o valor para uma entrada da RNA – este vai apenas transmitir a sua informação de RSSI para o âncora em sequência. Os sensores âncoras a partir do segundo âncora já realizam a estimativa de posição do alvo passando a potência do sinal recebido dos sensores âncoras antecessores para sua respectiva RNA. Por fim, o método toma a saída da RNA, que consiste na estimativa de posição do alvo na forma de par ordenado (x, y) normalizado entre zero e um, e envia ao próximo âncora adjacente juntamente (símbolo U de União, na Figura **3**.**3**) com todos os RSSIs de posse até o momento. Observe na Figura **3**.**4** que cada entrada da RNA está associada unicamente à potência de seu respectivo sensor âncora durante o método de localização.

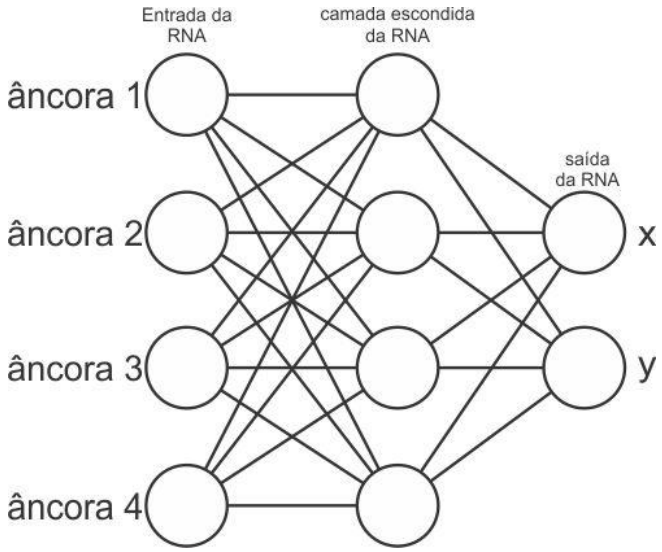

Figura 3.4: Associação entre RSSI dos sensores âncoras e as entradas da RNA.

De acordo com a definição do método descrito até aqui, observa-se que os sensores âncoras, que estimam a posição do alvo, podem receber dois tipos de mensagens – uma proveniente do alvo e outra proveniente do âncora imediatamente antecessor. Na seção a seguir, esta visão torna-se clara com a divisão do método em estados que executam sequencial e paralelamente.

3.2 Execução paralela, tolerância a falhas e tempo de execução do método.

Como pode ser visto na Figura **3**.**5**, o método proposto pode ser dividido verticalmente em cinco estados, considerando os quatro âncoras envolvidos no processo de localização.

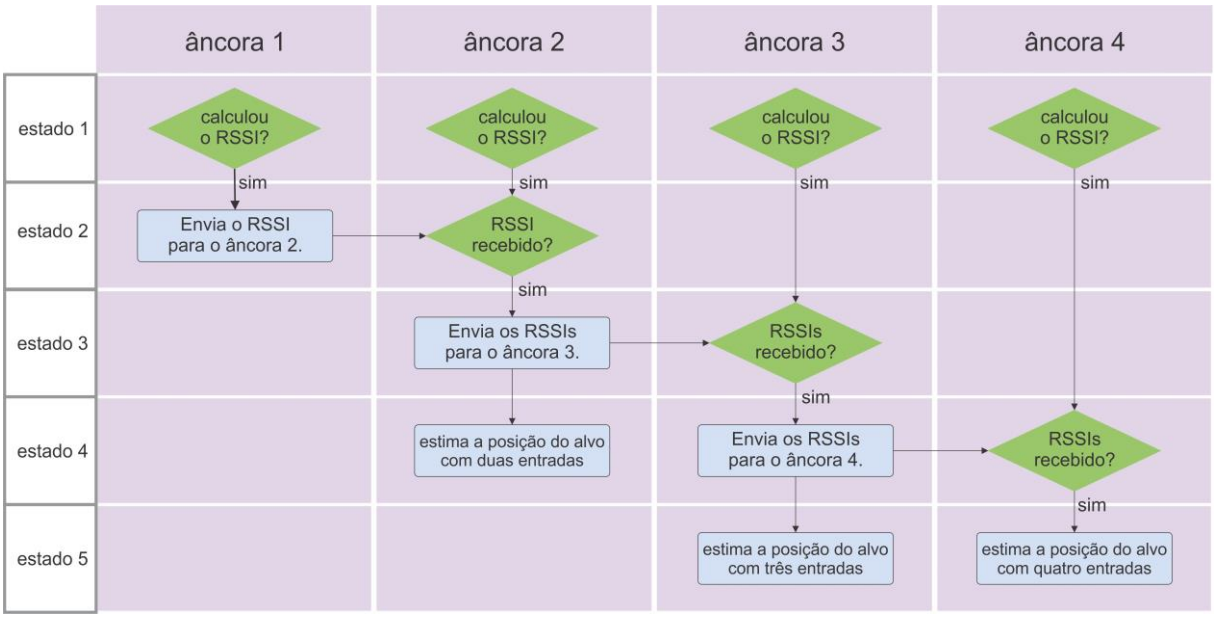

Figura 3.5: Fluxo paralelo de processos que representam o método proposto apresentado na seção anterior.

Na Figura **3.5**, o processamento de localização no mesmo estado é, idealmente, executado em paralelo considerando os quatro âncoras, após o recebimento da mensagem *broadcast* do alvo. Note-se que o âncora dois oferece resposta mais rapidamente mas com menor acurácia. Os âncoras três e quatro oferecem respostas em maior tempo, mas de forma mais eficaz, em função da quantidade de RSSI utilizadas na estimativa.

Quanto a tolerância a falhas, se o quarto âncora falhar, o método pode oferecer a posição estimada através do segundo e terceiro âncora. Se o terceiro âncora falhar, o método ainda pode oferecer alguma resposta razoável através do segundo âncora. Ressalta-se que o significado de falha consiste no não recebimento da mensagem do âncora anterior pelo âncora atual, num determinado período configurado previamente. Em redes *mesh* é necessária a existência de pelo menos um dispositivo coordenador para iniciar as configurações da rede. Com base neste requisito, este trabalho assume que o primeiro âncora é o coordenador da rede. Logo, se o coordenador da rede falhar, este deve ser substituído, ou uma nova seleção de âncoras deve ser aplicada (ZEN, 2015).

Contudo, nesta dissertação não considera-se falhas no primeiro âncora, deixando esta restrição para trabalhos futuros. Se o segundo âncora falhar, analogamente o terceiro e o quarto âncora deveriam estimar a posição, assim como o primeiro e o segundo âncora o fazem. A

estimativa de posição envolvendo apenas o terceiro e quarto âncoras também é deixada para trabalhos futuros. A justificativa para deixar estas duas implementações para trabalhos futuros é, para o primeiro caso, a grande quantidade de trabalhos existentes sobre métodos de seleção de novos âncoras coordenadores na rede. Para o segundo caso, se deu ao fato de que a constatação da capacidade de realizar uma aproximação razoável com apenas dois sensores âncoras pode ser verificada apenas durante a análise final dos resultados.

A taxa de erro de distância considerando falhas no terceiro e quarto sensores âncoras será apresentada na seção de resultados.

Com relação ao tempo de execução do método, o tempo de espera o qual um âncora se submete para receber do âncora adjacente a potência do sinal é um parâmetro configurado diretamente na implementação. A tolerância de espera do tempo configurado considerou a distância de propagação do sinal no meio (ar) entre os âncoras, e desconsiderou o tempo de processamento do método devido a este último ser muito menor do que o primeiro, como será apresentado na seção de resultados alcançados. Este tempo de execução será melhor comentado em uma seção a seguir, a qual trata da máquina de estados que vai implementar o controlador apresentado na Figura **3.3**, considerando o tempo de execução do algoritmo.

A seguir será descrito como foram adquiridas as informações *a priori* para a realização do método de estimativa a partir da RNA e como essas informações foram adequadas para lidar com as falhas descritas anteriormente nos sensores âncoras.

3.3 Definição da Base de Dados utilizada no Experimento.

A RNA foi treinada considerando uma base de dados gerada a partir da coleta de informações *a priori* provenientes da área de experimentação, como descrito como na **Figura 3.6**.

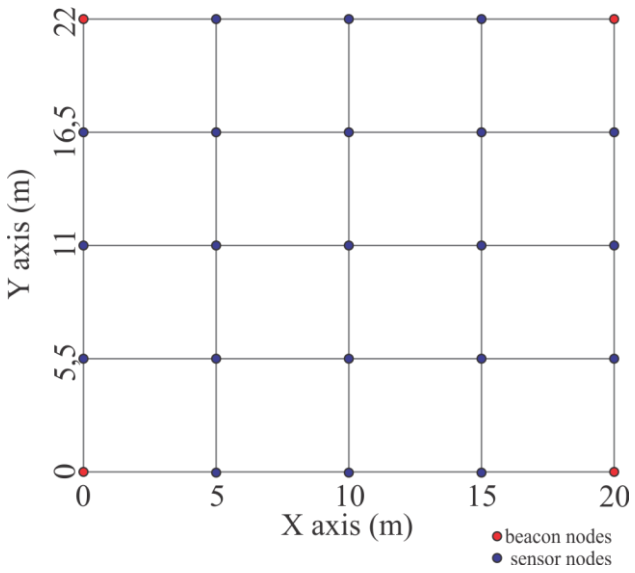

Figura 3.6: Esquema da área do experimento outdoor. Os pontos em vermelho são os âncoras. Os pontos em azul são os locais onde a posição do alvo foi coletada.

Neste experimento sensores âncoras foram inseridos (pontos em vermelho) nas extremidades da área experimentada. Os âncoras recebiam de um sensor alvo, disposto em cada posição (pontos em azul), a potência do sinal (dBm) de uma mensagem em *broadcast*. Desta mensagem, cada âncora tomou a potência do sinal convertido na forma de um indicador (RSSI).

A relação entre a normalização para a faixa de valores de 26 até 88, utilizada como entrada da RNA, está relacionada proporcionalmente com a faixa de valores do RSSI do rádio XBEE PRO utilizado no experimento – a saber, de -26dBm até -88dBm (ou de 0x1A até 0x58, em hexadecimal). Os valores adquiridos pelo rádio dos âncoras são coletados na forma de dBm e posteriormente normalizados para o intervalo definido anteriormente, associando-se a posição do alvo em cada local definido na Figura **3.6**.

Com os quatro indicadores de potência e a posição conhecida do alvo em cada ponto, gerou-se uma base de dados em texto simples de 21000 amostras (linhas). Este valor de amostras foi definido no experimento para coletar mil amostras em um minuto em cada ponto de coleta em azul descritos na Figura **3.6**. Estas informações foram arranjadas na forma da *tupla* (colunas) descritas como a seguir.

$$a_1, b_1, c_1, d_1, x_1, y_1$$
$$\vdots$$
$$a_n, b_n, c_n, d_n, x_n, y_n$$

Na listagem acima, a_n, b_n, c_n e d_n representam o indicador de potência recebida pelo respectivo sensor âncora (a, b, c ou d) considerando o ponto *n* de localização do alvo (**Figura 3.6**).

Um pós-processamento nesses dados coletados permite que a RNA entenda quando ocorrerá falhas no terceiro e quarto sensor âncora. Esta adequação se dará através da duplicação dessas 21 mil amostras adquiridas inicialmente, considerando o valor zero para os valores duplicados para o terceiro e quarto sensores âncoras – dando a entender para a RNA que estes sensores não cooperaram durante o processo na estimativa de localização do alvo. Na prática, durante o procedimento de treino da RNA, a função de aproximação resultante gerada pela mesma vai se adequar a estas entradas consideradas com o valor zero para poder oferecer uma resposta aproximada. Por exemplo, considerando uma instância para a tupla na listagem a seguir:

$$26, 30, 45, 50, 0.5, 0.5$$

onde $a_n = 26, b_n = 30, c_n = 45, d_n = 50, x_n = 0.5$ e $y_{n=} = 0.5$, serão inseridas na base de dados de informação *a priori* as duas linhas seguintes:

$$26, 30, 45, 0, 0.5, 0.5$$

para induzir a RNA a entender o erro no quarto sensor âncora e,

$$26, 30, 0, 0, 0.5, 0.5$$

para induzir a RNA a entender o a falta de informação de RSSI relacionada ao terceiro e ao quarto âncora.

A seguir será apresentada a definição e descrição do método de localização, considerando os seus componentes na plataforma de implementação FPGA.

3.4 Descrição e Definição do Método na Plataforma FPGA.

Para que o método de localização deste trabalho possa ser desenvolvido, toda a sua lógica deve ser sintetizável, ou seja, deve ser convertida em uma lógica combinatória ou sequencial de registradores capazes de serem implementadas em uma plataforma reconfigurável de hardware, no caso, o FPGA, utilizando uma linguagem de descrição de hardware, como por exemplo, o VHDL.

Os componentes do método de localização foram desenvolvidos separadamente, de forma modular, e depois integrados para compor o sistema completo, de acordo com a Figura **3.7**.

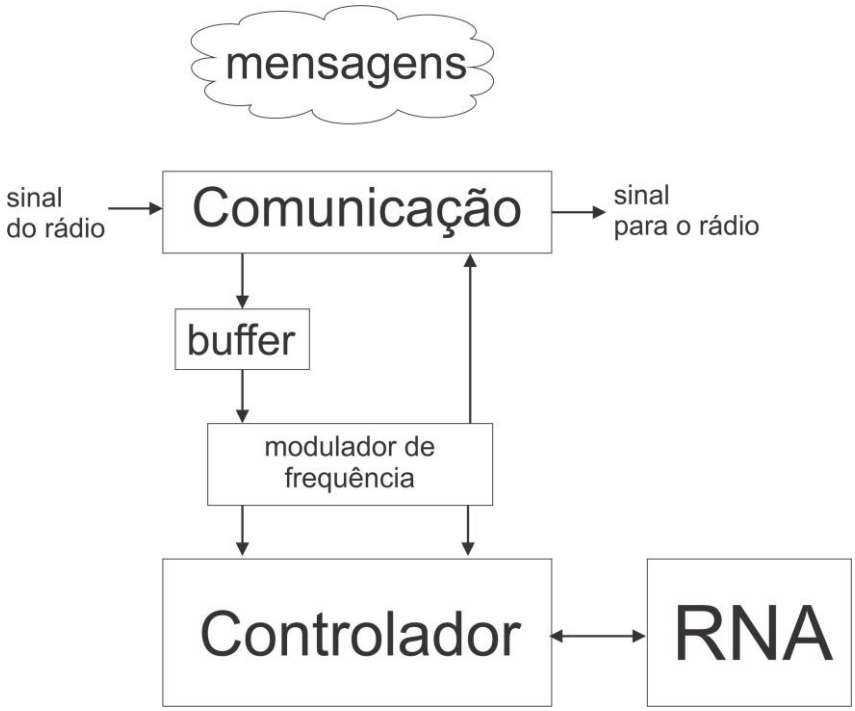

Figura 3.7: Diagrama de blocos da divisão modular do método de localização implementado nos sensores âncoras.

Na Figura **3.7**, os sinais do módulo de comunicação são conectados lógica e fisicamente com o emissor e o receptor do rádio Xbee utilizado nos experimentos. O modulador de frequência e o *buffer* são componentes auxiliares desenvolvidos para oferecer suporte de memória ao módulo comunicador e de integração de frequência entre este módulo e o módulo controlador no sistema, respectivamente. O formato da mensagem define a forma das informações que são transmitidas entre os sensores. A seguir apresenta-se um resumo dos módulos desenvolvidos e, posteriormente, explica-se com mais detalhes cada um desses módulos nas subseções seguintes. Em resumo, tem-se:

- O formato das mensagens trocadas entre os sensores.
- O módulo de comunicação serial assíncrona (UART).
  - Entrada serial.
  - Saída serial.
- O *buffer* para armazenar os frames recebidos pelo módulo de comunicação.

- O modulador de frequência (50MHz do FPGA para 1.8KHz da UART-115200kbps) para integrar a frequência de operação do controlador com o padrão de frequência do UART implementado no módulo de comunicação.
- O módulo da Rede Neural Artificial.
    - Neurônio com quatro entradas e uma saída, utilizando a função Elliot.
    - Camada genérica contendo quatro neurônios.
    - Camada de saída contendo dois neurônios.
    - A RNA composta pelas camadas anteriores.
- E o módulo do controlador, definido como uma máquina de estados finitos.

O detalhamento apresentado a seguir consiste na descrição dos módulos diretamente na plataforma FPGA, utilizando o software Quartus II, com a linguagem de descrição VHDL.

A. Troca de mensagens entre os sensores

A primeira mensagem recebida pelos âncoras, proveniente do alvo, pode ser de qualquer tipo a partir deste alvo. O rádio utilizado para comunicação fornece a potência do sinal recebido da última mensagem recebida pelo sensor âncora em questão, a contar do primeiro byte recebido, independente do conteúdo da informação da mensagem.

As mensagens trocadas entre os âncoras, diferentemente da primeira mensagem em *broadcast* recebida por todos estes a partir do alvo, possui informações relevantes ao processo de localização. Esta mensagem é constituída de um identificador de âncora emissor, seguida da potência do sinal recebido por este âncora e seus antecessores. A mensagem trocada entre os âncoras são similares, diferindo apenas na quantidade de informação de potência.

<table>
<tr><th>Identificador do âncora emissor.</th><th>Potência do âncora 1.</th><th>Potência do âncora 2.</th><th>Potência do âncora 3.</th></tr>
<tr><td colspan="2">Forma a mensagem do âncora 1 para o 2</td><td></td><td rowspan="2"></td></tr>
<tr><td colspan="3">Forma a mensagem do âncora 2 para o 3</td></tr>
<tr><td colspan="4">Forma a mensagem do âncora 3 para o 4.</td></tr>
</table>

Figura 3.8: Mensagem trocada entre os âncoras. O identificador é um caractere ASCII único para cada âncora.

A mensagem na Figura **3.8** possui no mínimo 3 bytes e no máximo 7 bytes. A potência transmitida é descrita como dois caracteres ASCII, representando dois valores hexadecimais. O identificador dos sensores âncoras é representado por um caractere ASCII que não representa valor hexadecimal, ou seja, símbolos tais quais o "(", ")" e o "{", não conflitando com o valor de potência do RSSI, o qual vai assumir valores ASCII entre "0" e "F".

Em sua máquina de estados, o controlador simplesmente aguarda o identificador da mensagem e lê, em seguida, a potência representada pelos dois bytes, respectivamente, para cada âncora. O controlador converte os dois bytes ASCII para dois valores hexadecimais, os quais compõe um mesmo byte.

A conversão entre ASCII e hexadecimal é necessária, pois a comunicação serial da UART utiliza o padrão ASCII para a comunicação e a RNA utiliza um valor inteiro como entrada, o qual é facilmente convertido a partir do valor hexadecimal.

Na Figura **3.9** é apresentado um fluxograma que demonstra a conversão do valor ASCII, recebido do rádio para o FPGA, para um valor hexadecimal e, posteriormente, para um valor inteiro sem sinal, o qual é utilizado como entrada para a RNA. Nesta **Figura 3.9**, palavras em negrito (***byte*** e ***RSSI_int***) são interpretadas como registradores que armazenam um *byte*. O registrador ***nibble*** armazena quatro *bits*. ***RSSI_hex*** representa um registrador de dois bytes indexados no colchete pelo índice inteiro ***endereço***. A função ***converter*** representa uma função interna da biblioteca VHDL que realiza conversão entre bases numéricas. Este procedimento ocorre exatamente entre a aquisição de informação da potência do rádio Xbee e a passagem dessa potência como entrada para a RNA.

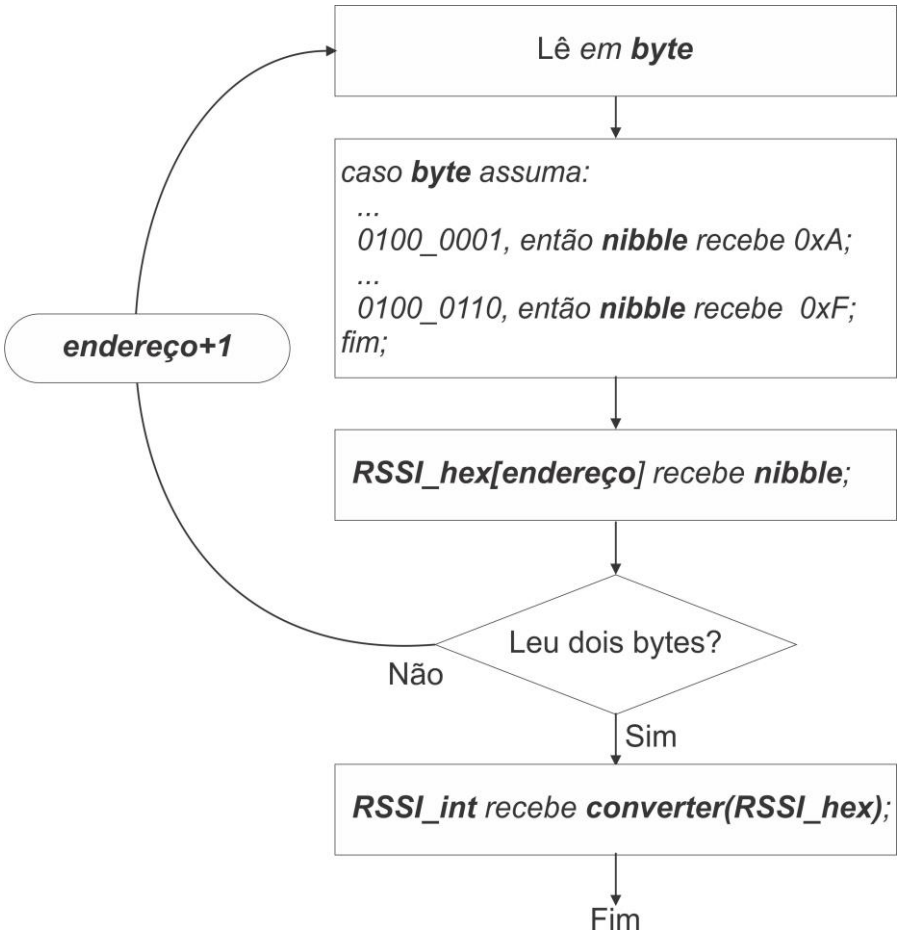

Figura 3.9: Fluxograma de execução da conversão da informação de RSSI do rádio (em ASCII) para um inteiro sem sinal utilizado na RNA.

Assim que o controlador solicita o RSSI da última mensagem recebida pelo rádio Xbee, o rádio envia para a sua porta de transmissão UART dois caracteres ASCII, que representam a potência do sinal e que são lidos pelo FPGA. Para cada um dos 16 bytes possíveis em ASCII que representam os números hexadecimais armazenados no registrador ***byte***, são associados aos seus respectivos valores em hexadecimal de quatro *bits* armazenados no registrador ***nibble***, que por sua vez armazenará individualmente cada metade do registrador ***RSSI_hex*** considerando a sequência do laço de repetição. O processo de leitura e conversão do registrador ***byte*** se repete mais uma vez e, após o registrador ***RSSI_hex*** receber um byte completo, este é convertido para um inteiro sem sinal e armazenado no registrador ***RSSI_int***. O valor resultante em ***RSSI_int*** é então normalizado para a faixa de valores entre 0 e 1. Dessa forma a RNA pode trabalhar com valores padrões para qualquer modelo de rádio, apenas realizando a reescala dos valores brutos de potência para os valores entre 0 e 1. Esta normalização também contribui para que a RNA

possua valores baixos em seus parâmetros livres, durante a execução do treinamento com algoritmo retropropagação resiliente, refletindo na reduzida quantidade de bits para a representação binária desses valores e de seus produtos durante o processo executado nas funções de ativação Elliott, como será visto na próxima seção.

B. Módulo da Rede Neural Artificial descrita no FPGA.

Após a execução da coleta de dados para fornecer informação *a priori* para a RNA, esta foi modelada seguindo as seguintes associações:

- O indicador de potência dos quatro âncoras equivalem às quatro entradas da RNA.
- A posição do alvo equivale as duas saídas da RNA, considerando um plano Cartesiano.

Dadas estas características, optou-se por utilizar o algoritmo RNA com retroalimentação (*feedforward*) com múltiplas camadas, com treinamento retropropagação resiliente (*resilient backpropagation*), para implementar paralelamente 04 (quatro) RNAs, uma em cada âncoras.

Como função de ativação, optou-se pelo uso da função de ativação logística sigmoide, pelo fato de a primeira derivada desta função ser facilmente sintetizável e necessária em uma possível futura implementação para treinamento em tempo real da RNA implementada.

Apesar disso, funções como a sigmoide não possuem representação sintetizável por possuírem elementos exponenciais, devido a inexistência de um algoritmo que execute em tempo polinomial para encontrar uma solução que represente tal função em lógica combinatória e/ou sequencial (CHU, 2006).

Existem diversas aproximações para a função de ativação sigmoide. Nesta dissertação, considerou-se a simplicidade de implementação e o não uso de memória adicional e a acurácia na aproximação. Antes de apresentar a aproximação escolhida para o desenvolvimento deste trabalho, serão apresentadas as principais aproximações encontradas durante a pesquisa na literatura.

Uma forma de aproximação parcialmente não linearizada da log-sigmoide é encontrada em (XIE, 2012), onde a função log-sigmoide é aproximada por duas funções – uma linear e uma quadrática – e por 25 faixas de valores (piece-wise) armazenados em memória. Para a parte linear (no meio da log-sigmoide), a função foi aproximada por uma equação linear. Para a primeira parte curva, no início da função, uma função do segundo grau é utilizada para

aproximar o resultado. Para a parte curva mais acima da função log sigmoide, uma LUT (l*ookup table*) com 25 valores é utilizada armazenada em uma memória de acesso randômico. Considerando RNAs, os erros em (XIE, 2012) são razoavelmente pequenos mas ainda sim afetam consideravelmente a resposta do algoritmo – por isso consideramos que a complexidade de implementação dessa referência, apesar de apresentar erros próximos aos alcançados nesta dissertação, não se justifica para este trabalho. Em (MANISH PANICKER, 2012) uma versão linearizada da aproximação de log-sigmoide baseada em LUT é apresentada. Apesar de ser computacionalmente mais viável (considerando implementação), os erros de aproximação são maiores e o consumo de memória e maior do que as apresentadas em (XIE, 2012).

Uma outra alternativa que também utiliza LUTs é apresentada em (I. DEL CAMPO, 2010), onde séries de Taylor são utilizadas para representar as regiões da log-sigmoide as quais não são consideradas áreas de saturação (regiões das extremidades da log-sigmoide, as quais tendem para zero ou para um). Os resultados em mostram que quanto maior a região de Taylor utilizada para aproximar a função menor é o erro. Contudo, devido à complexidade, o algoritmo para a série de Taylor foi implementado como um sistema embarcado na forma de um processador digital de sinal, dentro do FPGA. Também foram utilizados dois módulos de memórias de leitura e pelo menos um multiplicador no circuito implementado.

A modelagem da RNA considerou a sua posterior implementação em hardware. Neste caso, a unidade fundamental do estimador, o neurônio, foi escolhida como sendo uma aproximação para a função de ativação logarítmica sigmoide. Além das aproximações que foram consideradas anteriormente, optou-se pela escolha da função Elliott como representante da função de ativação sigmoide. A função Elliott é sintetizável em hardware, ou seja, pode ser representada através de lógica combinatória e/ou sequencial pelos algoritmos de síntese existentes, os quais executam em tempo polinomial. A **Figura 3.10** apresenta uma comparação visual entre a função log sigmoide e a sua aproximação - a função Elliott.

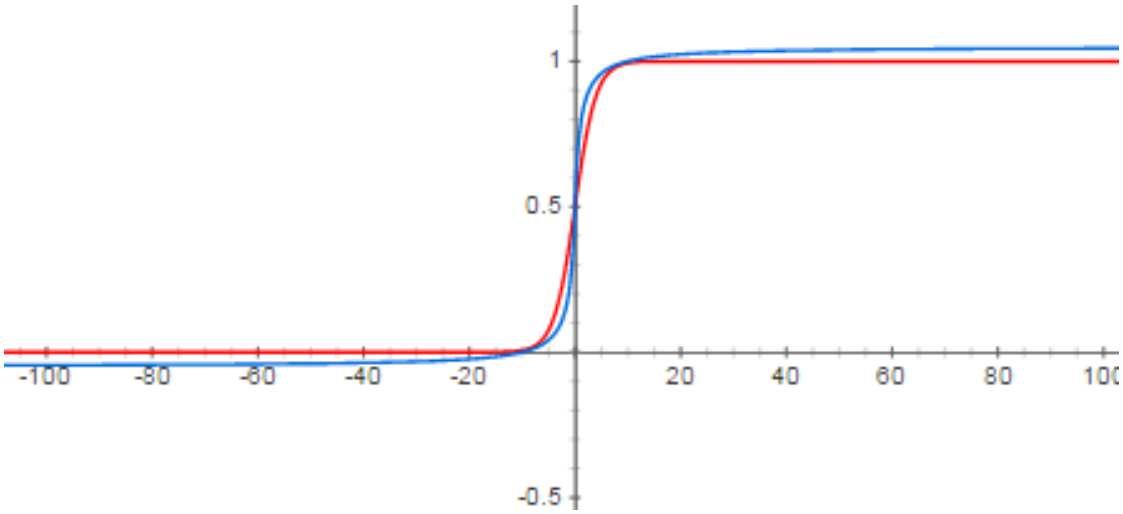

Figura 3.10: Comparação visual entre as funções Log Sigmoide (em azul) e Elliott (em vermelho).

A equação que define a função Elliott é apresentada na **Equação 4**.

$$f(x) = \left\{ \left[{0.55x}/{(1 + abs(x))} \right] + 0.5 \right\} - T \tag{4}$$

onde *abs*(.) equivale ao valor absoluto. A constante *T* representa o valor de *threshold* associado ao neurônio.

Não foram encontrados na literatura trabalhos que utilizem esta função Elliott implementada em FPGA, até o momento da escrita deste tópico, no entanto, esta função é bastante conhecida em nível de programação em software como a função *fast* sigmoide. Os dois trabalhos, (BEIU, 1994) e (SIBI, JONES e SIDDARTH, 2013) foram relevantes para a implementação da função de aproximação Elliott para a função log-sigmoide, utilizado nesta dissertação.

A seguir, a definição do algoritmo na plataforma FPGA, incluindo a RNA, o módulo de comunicação e o controle definido como uma máquina de estados finitos.

C. Módulo de Comunicação descrito no FPGA.

O primeiro módulo a ser desenvolvido foi o de comunicação serial assíncrona. Essa escolha (ordem de implementação) se deu devido à necessidade de se verificar se os dados internos do algoritmo estariam apresentando os resultados esperados. Este módulo possui duas partes, uma de transmissão e uma de recepção de dados. O controle desta comunicação é realizado pelo desenvolvedor. A escolha deste método serial e assíncrono (*universal asynchronous receiver/transmitter* (UART) – receptor/transmissor universal assíncrono) de comunicação foi

relevada pelo fato de os módulos Wi-Fi XBEE PRO, assim como a maioria dos módulos de comunicação disponíveis no mercado, utilizarem este método de comunicação. A UART foi desenvolvida para operar na maior frequência padronizada no IEEE, 115200bps. Contudo, para auxiliar a recepção de dados da UART para o controlador, um buffer foi desenvolvido como memória intermediária entre esses dois módulos. Este buffer possui 256 bytes de memória sendo o seu tamanho superdimensionado em comparação ao tamanho das mensagens transmitidas entre os sensores durante o processo experimental de localização, com o intuito de demonstrar que o método de localização permite a transmissão de mais informações no caso de aplicações práticas. Além do buffer, foi desenvolvido outro componente que exerce a função de integrar, em nível de frequência, o UART e o controlador. Todos os detalhes sobre a UART, o buffer e o modulador de frequência podem ser verificados nos **Apêndices 6A, 6B** e **6C**. A seguir, é apresentado a descrição do controlador que define a máquina de estados finitos de Moore, e é responsável por gerenciar toda a execução do método de localização implementado nos sensores âncoras.

D. Módulo do Controlador descrito no FPGA.

O controlador é representado por uma máquina de estados finitos de Moore (MEF) descrita em hardware e implementada em VHDL através da estrutura *case* da linguagem. A MEF possui 39 estados no total, apresentados no **Apêndice 6D**. Como cada estado precisa executar em um *clock* do FPGA, o que equivale a 20 nano segundos por *clock*, a plataforma garante que todos os estados sejam executados em 7,8 × 10-7 segundos. O de desenvolvimento das instruções executadas em cada estado considerou o limite de 20 nano segundos por estado, garantindo que erros lógicos não aconteçam por conta disso. Contudo, um estado na forma de *delay* de um segundo é executado por conta das limitações do módulo de rádio XBEE PRO, que precisa aguardar este tempo para responder solicitações internas quando em modo de comando. As solicitações internas provenientes do controlador para o rádio são os comandos “+++” e, posteriormente após 1 segundo, o comando “ATDB”, os quais realizam a solicitação da potência da última mensagem recebida pelo rádio.

O controlador é modelado através da lógica de estados baseada em registradores, onde em cada ciclo de clock, os registradores assumem um valor atual com base no seu valor anterior mais as operações lógicas sobre esses valores anteriores. Essas operações lógicas que operam sobre esses registradores utilizam as portas de entradas do controlador e/ou outros registradores.

Por fim, os registradores atualizam a saída do controlador e/ou outros registradores internos. A Figura **3.11** apresenta um diagrama de funcionamento da lógica implementada no controlador.

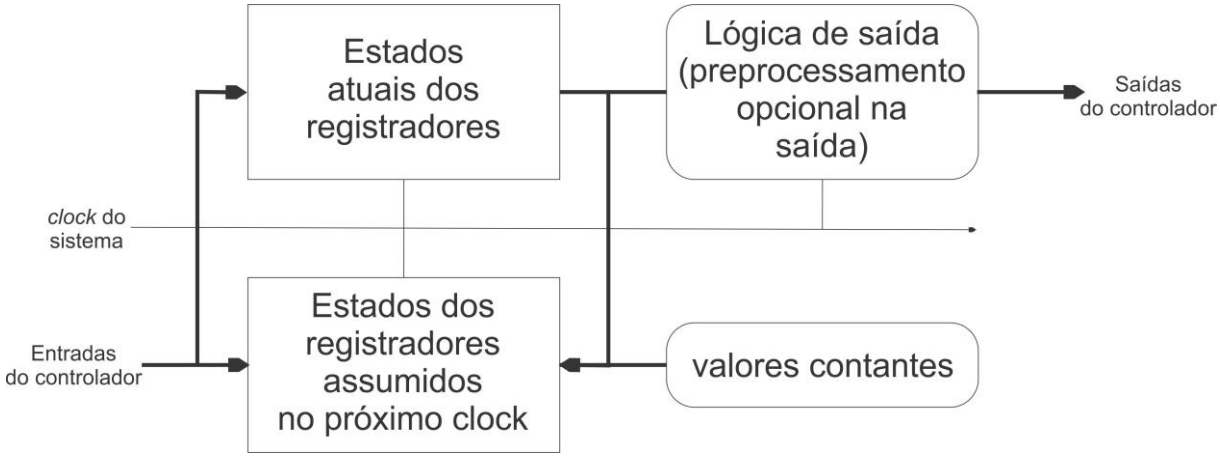

Figura 3.11: Lógica de registradores que implementa o controlador.

Na **Figura 3.11**, os valores constantes são, por exemplo, parâmetros constantes da função de ativação Elliott e o ID que identifica o sensor âncora. Quanto a lógica de saída, a qual é um pré-processamento opcional antes de o controlador enviar seus dados - estão relacionadas com a formatação dos dados de saída nas portas, adequando os valores numéricos dos resultados da RNA para sequência de *bits* genéricos, garantindo que estes *bits* sejam passados sem formatação e reinterpretados na outra ponta.

Em seguida, será mostrado como foi realizada a ligação física entre o FPGA e o rádio Xbee, considerando a utilização dos pinos de uso geral do FPGA e pinos de alimentação, assim como os botões que foram configurados como *reset* do sistema de localização.

4 Resultados

Nesta seção são apresentados os resultados alcançados durante o desenvolvimento, implementação e simulação da RNA e do método de localização como um todo. Os resultados são focados no erro de aproximação da distância Euclidiana real e estimada e no tempo de execução e resposta do método de localização. Algumas outras considerações relevantes encontradas durante a verificação dos resultados, porém não esperadas no início do projeto, também são discutidas, sendo consideradas melhorias a serem implementadas em trabalhos futuros.

A seguir, apresenta-se o desenvolvimento prático da RNA, desde a geração da sua topologia até os seus parâmetros livres que são adequados posteriormente à plataforma FPGA.

4.1 Desenvolvimento da RNA

Para gerar os parâmetros livres da RNA e posteriormente inseri-los na plataforma FPGA, utilizou-se a ferramenta RStudio, através da linguagem R e seu pacote Neuralnet, cujo resultado alcançou os parâmetros livres (pesos da RNA) apresentados na **Figura 4.1**. Como descrito na seção 3.3, sobre a informação *a priori*, ou seja, a base de dados desenvolvida para treino e teste da RNA, cerca de 1000 amostras foram coletadas em cada ponto *n* do alvo totalizando as 21000 amostras. A temperatura ambiente média foi de 26° Celsius e a umidade relativa do ar sendo igual a 80g/m^3.

Para testes, 25% das 21000 amostras da base de dados do experimento foram utilizadas na RNA implementada. A topologia da RNA considerou os resultados alcançados em trabalhos desenvolvidos pelo grupo (MACHADO, LARRAT e MONTEIRO, 2013) e (E SILVA, MACHADO e MONTEIRO, 2013), estendendo a RNA para mais uma camada escondida para aumentar a capacidade de aproximação do algoritmo. A ferramenta RStudio utilizada para gerar o modelo permitiu a modelagem de uma única RNA com duas saídas, diferente do alcançado nos trabalhos correlatos em (SAAD e ALHADY, 2014) e (SABTO e AL MUTIB, 2013), o que permitiu alcançar a estimativa das duas variáveis da coordenada estimada do sensor alvo ao mesmo tempo. O treinamento da RNA decorreu em 5 horas utilizando um computador com processador *i7* geração 3 e 8GB RAM. Os parâmetros de treinamento da RNA são descritos na Tabela **4.1**.

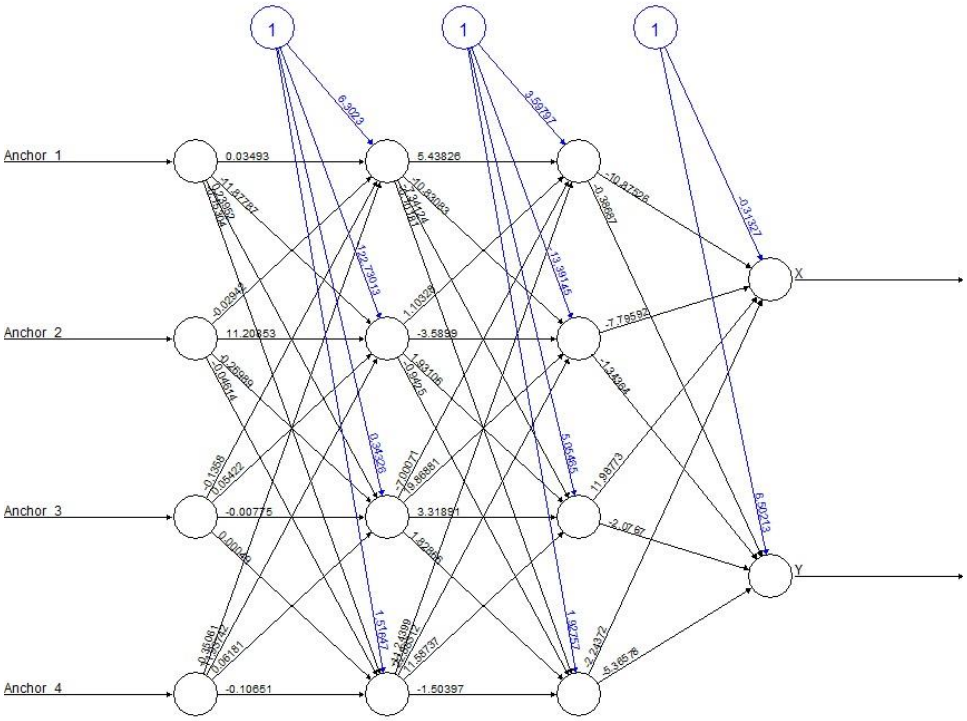

Figura 4.1: Modelo da RNA gerada através da ferramenta RStudio.

Tabela 4.1: parâmetros para o treinamento da RNA.

Parâmetro	*Valor*
Algoritmo	Rprop-.
Função de ativação	Log Sigmoid
Taxa de Aprendizado da RNA	0.001
Erro de parada considerado no treinamento	0.14

Na Tabela **4.1**, Rprop- ou retropropagação resiliente é o método de treinamento escolhido para a RNA, o qual possui uma atualização dos parâmetros livres de forma mais rápida do que a sua versão mais comum, o algoritmo retropropagação, observando a mudança do sinal do parâmetro livre antes de realizar a multiplicação deste por um fator de adaptação (RIEDMILLER e BRAUN, 1992). A função de ativação logística sigmoide foi escolhida conforme a escolha da aproximação do neurônio da RNA, através da função de ativação Elliott, utilizada na implementação, com discutido no **Capítulo 3.4**, **seção B**. A taxa de aprendizado escolhida é uma *rule of thumb* utilizada no treinamento de RNA, e é comumente empregada

com os algoritmos baseados em retropropagação de erros, podendo variar entre 0.001 e 1. O valor da taxa de aprendizagem de 0.001 foi considerado pelo fato de que o algoritmo retropropagação resiliente pode executar mais rapidamente que sua versão original - retropropagação. Quanto ao erro de parada de treinamento, ainda relacionado a Tabela **4.1**, durante várias execuções de treinamentos realizados previamente para se alcançar melhores resultados, observou-se empiricamente que o a média do menor erro alcançado próximo do final do treino da RNA era de 1.5. Portanto, consideramos a razoável taxa de erro médio como sendo 1.4.

Continuando para a etapa de descrição da RNA no FPGA, a Figura **4.2** apresenta o diagrama que representa a descrição em VHDL da função de ativação Elliott, a qual vai compor as camadas da RNA. Esse diagrama é representado internamente por operações de multiplicação, divisão, adição, subtração e um cálculo de valor absoluto.

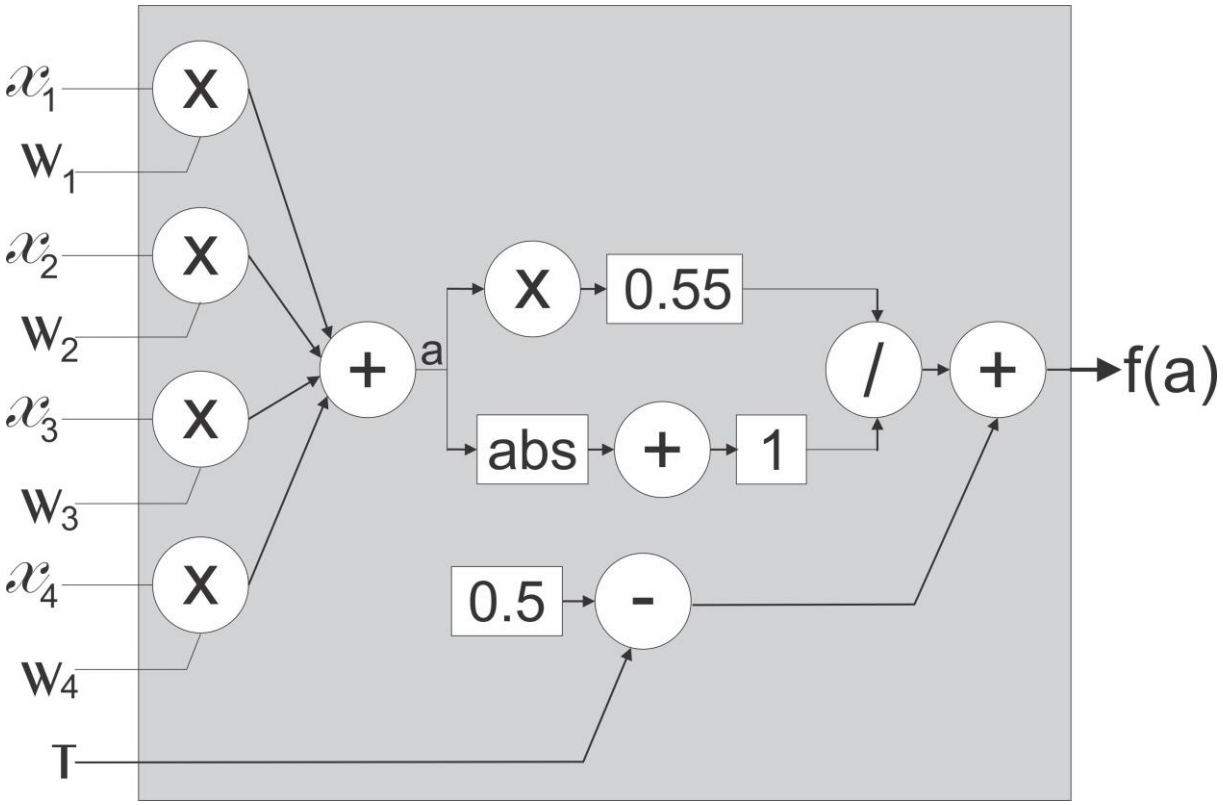

Figura 4.2: Diagrama do modelo da função de ativação Elliott.

Na Figura **4.2**, x_n (n = 1, ..., 4) representam as *n* entradas do neurônio, assim como , w_n referem-se aos parâmetros livres associados em cada sinapse. O produto entre as entradas e os parâmetros livres são executados paralelamente. Em seguida, um somatório destes produtos é realizado, resultando em *a*. O produto de *a* com a constante 0,55 e a adição do valor absoluto

de *a* com a unidade (ABS(a)+1) ocorrem em paralelo. A resultante das operações sobre *a* são divididas um pela outra respectivamente e, por fim, o resultado é adicionado ao valor da subtração do *threshold T* com a constante 0,5. Tanto 0,55 quanto 0,5 são parâmetros que ajustam a função Elliott para o intervalo de resposta entre [0; 1]. Observe que esta operação de subtração ocorre em paralelo com as primeiras operações de produtos entre as entradas e os parâmetros livres do neurônio.

Após as etapas anteriores de definição da topologia e parâmetros livres da RNA e descrição da função de ativação Elliott, as camadas serão formadas pelo agrupamento destes neurônios, na forma de componentes, que é um recurso da linguagem de descrição de hardware utilizada no FPGA. O mesmo ocorre para a RNA completa, que será formada por componentes de camadas. O diagrama apresentado na Figura **4.4** representa as entradas da RNA, as duas camadas escondidas e a camada de saída, com suas funções de ativação Elliott definidas dentro de cada camada. Observa-se que a camada de saída possui apenas duas funções de ativação e que os parâmetros livres são implementados a parte das conexões entre as funções de ativação, mas ainda dentro do componente de implementação da RNA.

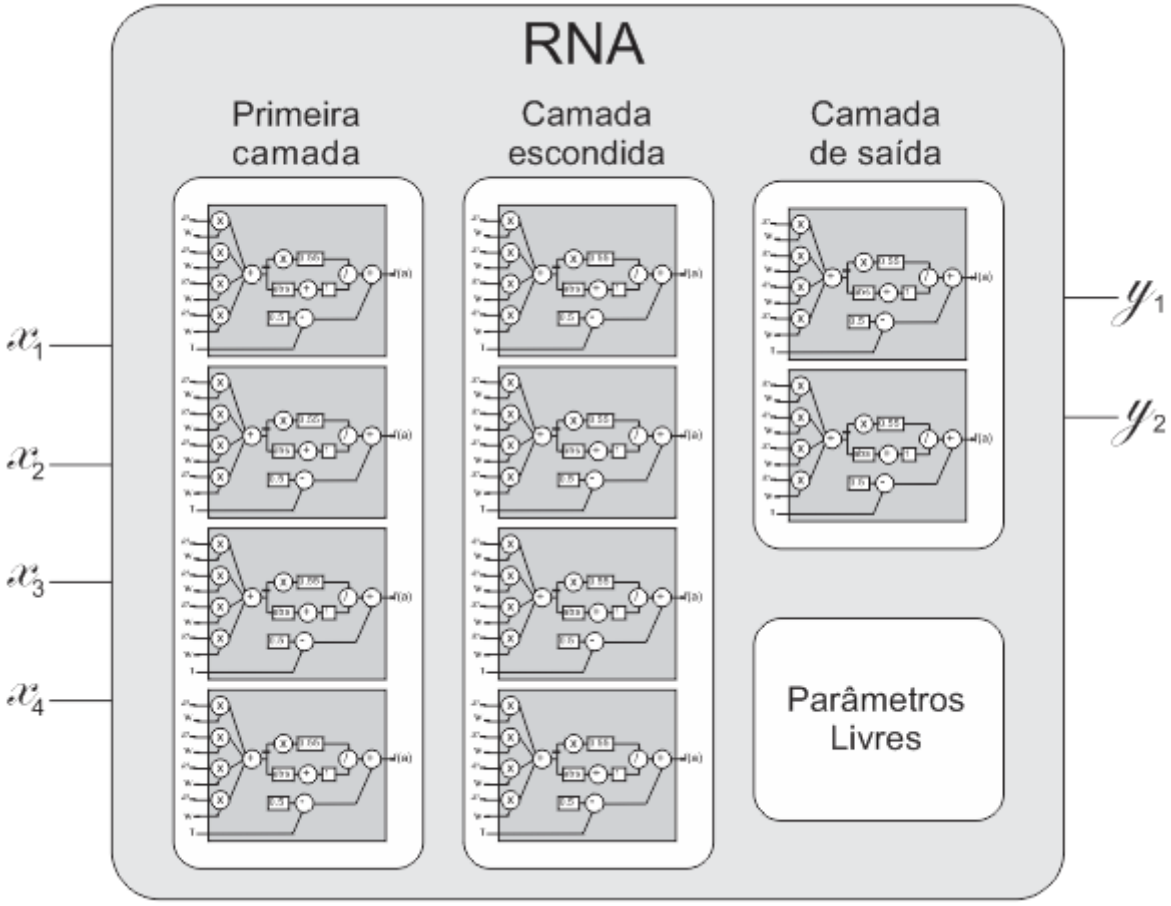

Figura 4.3: Representação em diagrama de componentes para a RNA.

Na seção seguinte será apresentada os resultados da interconexão da RNA, juntamente com os outros módulos do método de localização e a relação das entradas e saídas compartilhadas entre os módulos.

4.2 Interconexão entre os módulos do método de localização

A descrição de hardware apresentada na Figura **4.4** constitui todo o sistema implementado no FPGA DE0-nano. RST é o botão de reset geral do sistema. EN é o botão que executa passo-a-passo os estados do controlador durante a depuração do sistema. Os oito LEDs representam o estado corrente durante a depuração.

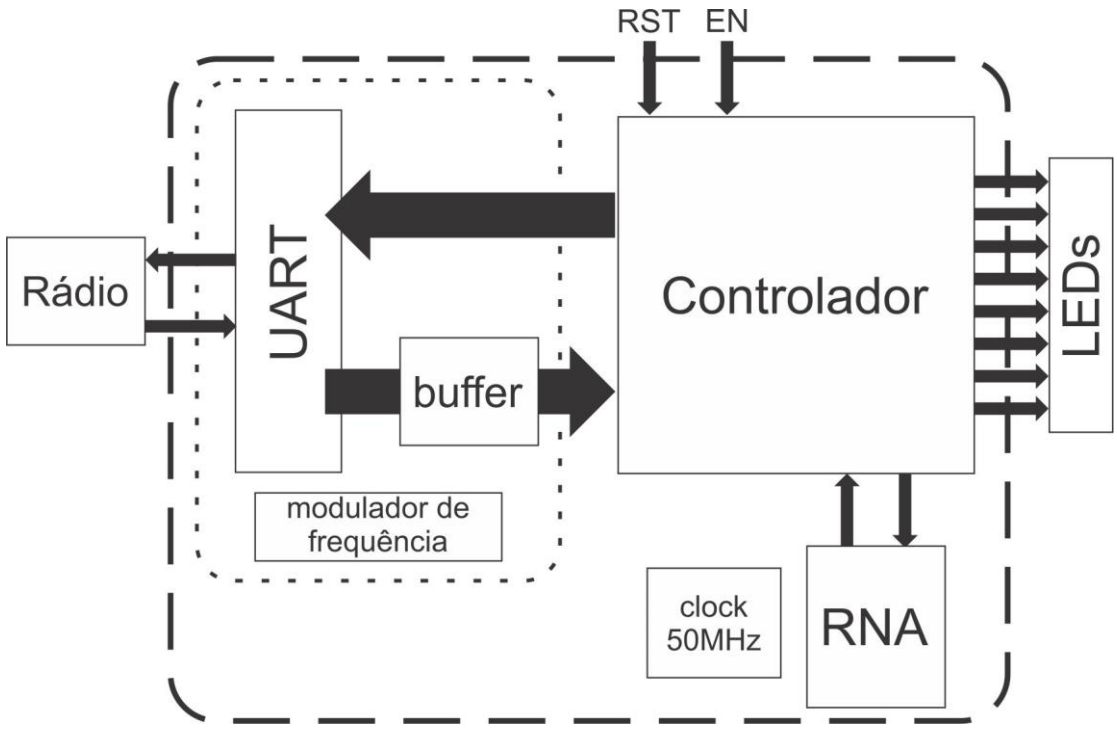

Figura 4.4: Diagrama de blocos da descrição do hardware do modelo no Quartus II.

Ainda na Figura **4.4**, do lado esquerdo, está o sub módulo de comunicação, que consiste no componente modulador de frequência, no componente receptor e emissor entre UART e o controlador (e também o rádio) e em um componente buffer do tipo "primeiro byte a entrar é o primeiro byte a sair" (*first in first out – FIFO)*. Ao centro, o componente controlador, responsável pelo gerenciamento de todos os sub módulos (o algoritmo da máquina de estados). Abaixo está o componente que representa a RNA. O componente de clock 50MHz fornece o sincronismo entre todo o sistema, inclusive para o modulador de frequência, o qual fornece uma frequência diferente de 50MHz para a UART.

O controlador possui oito portas de saída associadas aos *LEDs* (*LEDnport*, onde n = 1, ..., 8.) existentes no FPGA, os quais sinalizam, através de codificação binária destes *LEDs*, qual estado está sendo executado em qual momento, facilitando o entendimento do processo da máquina de estados finitos. Desta forma, ficou fácil entender o comportamento do método de localização durante os testes de depuração preliminares aos experimentos práticos, permitindo a verificação em tempo real do sistema. Na depuração, o controle de passagem entre os estados foi controlado por um dos botões do FPGA (porta de nome *EN*, na Figura **4.4**), pois a passagem natural (em 50MHz) entre os estados seria impossível de se visualizar a olho nu.

O *buffer* FIFO garante que os dados recebidos pelo componente da UART sejam armazenados até o momento em que o controlador precise utilizá-los.

O controlador permanece em estado de IDLE (ocioso, mas em escuta) frequentemente checando se o *buffer* está com alguma nova informação recebida. Quando o FIFO recebe alguma informação do componente receptor da UART, o controlador checa a identificação desta mensagem para verificar se o processo de localização se inicia, ou seja, se a informação de chegada no buffer é a mensagem enviada do alvo para os sensores âncoras.

O rádio que realiza a interface de comunicação sem fio (XBEE) conecta-se no FPGA DE0-nano, através de portas de propósito geral do FPGA, de acordo com a Tabela **4.2** a seguir.

Tabela 4.2: Associação entre os pinos do FPGA DE0-nano com o rádio Xbee

FPGA DE0-nano (Conector GPIO-0)	**Xbee**
Pino 2, referenciado como D3 no datasheet.	Pino 3, receptor UART do Xbee.
Pino 1, referenciado como A8 no datasheet.	Pino 2, transmissor UART do Xbee.
Pino 29, referenciado como VCC3P3.	Pino 1, alimentação de 3.3V do Xbee.
Pino 30, referenciado como GND.	Pino 10, aterramento do Xbee.

A conexão apresentada é bastante simples e é equivalente para qualquer módulo de rádio que siga o mesmo padrão de comunicação.

Após a implementação dos módulos no FPGA, passou-se para a verificação da estimativa de localização do alvo, considerando uma quantidade finita de pontos de localização deste alvo na área experimentada. Para estas estimativas, também foram incluídos os casos que consideram falhas no terceiro e quarto sensores âncoras. O resultado desta verificação é apresentado na seção seguinte.

4.3 Diferença entre as Respostas considerando a Posição do Alvo e a Aproximação da RNA

Para a implementação da rede neural, os parâmetros livres e os valores de *threshold* da RNA foram representados no controlador como valores constantes, como apresentado no **Capítulo 3** anterior, Figura **3.11**. Para esta representação, foram utilizados 16 bits para

representar a parte inteira (10 bits) e a parte decimal (6 bits) dos parâmetros livres gerados. Esses valores reais gerados após o treinamento foram convertidos diretamente em valores binários em complemento de dois. Ressalta-se que foi nesta etapa de conversão dos parâmetros livres e *thresholds* para a sua respectiva representação binária em complemento de dois que foi gerado o erro de quantização que afeta diretamente a resposta da RNA, causando uma diferença entre a posição real do alvo e a estimativa gerada pelo método aqui implementado. Para verificar estas diferenças de distâncias, esses erros são apresentados a seguir comparando a diferença de distância entre a posição real do alvo e as estimativas de posição utilizando os 25% das amostras para o teste da RNA, considerando a base de dados apresentada no **Capítulo 3**, para os seguintes casos:

- Utilizando a RNA com **função sigmoide** e com os pesos representados pelos números reais **gerados** durante o treinamento da RNA;
- Utilizando a RNA com **função Elliott** e com os pesos representados pelos números reais **gerados** durante a conversão binária em complemento de dois, e;
- Utilizando a RNA com **função Elliott** e com os pesos representados pelos números reais **quantizados** durante a conversão binária em complemento de dois (caso que reflete a implementação da RNA no FPGA).

Na **Figura 4.5**, verifica-se a diferença, para 25% dos casos da base de teste e na forma de distância em metros, entre a posição real do alvo e a posição estimada pela RNA utilizando a função sigmoide com os pesos gerados pelo treinamento da RNA.

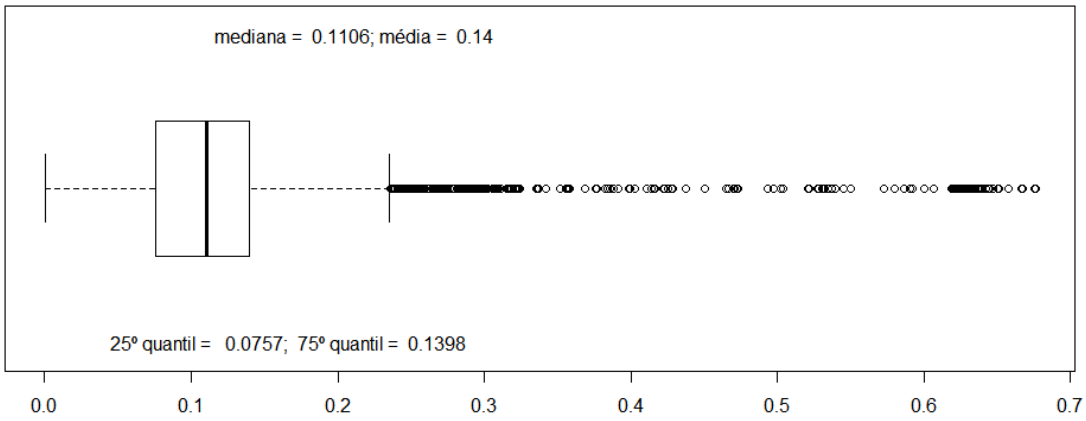

Figura 4.5: Análise estatística das informações considerando a função Sigmoide e os pesos reais gerados durante o treinamento da RNA.

Em seguida, a **Figura 4.6** apresenta a diferença de distância entre a posição real do alvo, igualmente para as 25% de amostras da base de teste, mas considerando a RNA com função Elliott e os pesos reais gerados durante o treinamento da RNA.

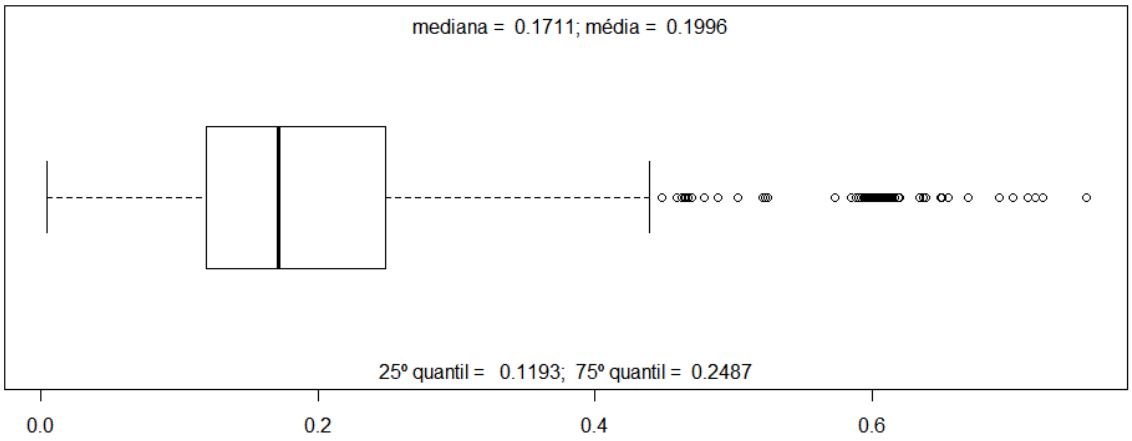

Figura 4.6: Análise estatística das informações considerando a função Elliott e os pesos reais gerados durante o treinamento da RNA.

Por fim, a **Figura 4.7** apresenta os resultados utilizando a função Elliott, mas considerando a conversão binária e consequentemente a quantização dos valores dos pesos gerados durante o treinamento da RNA.

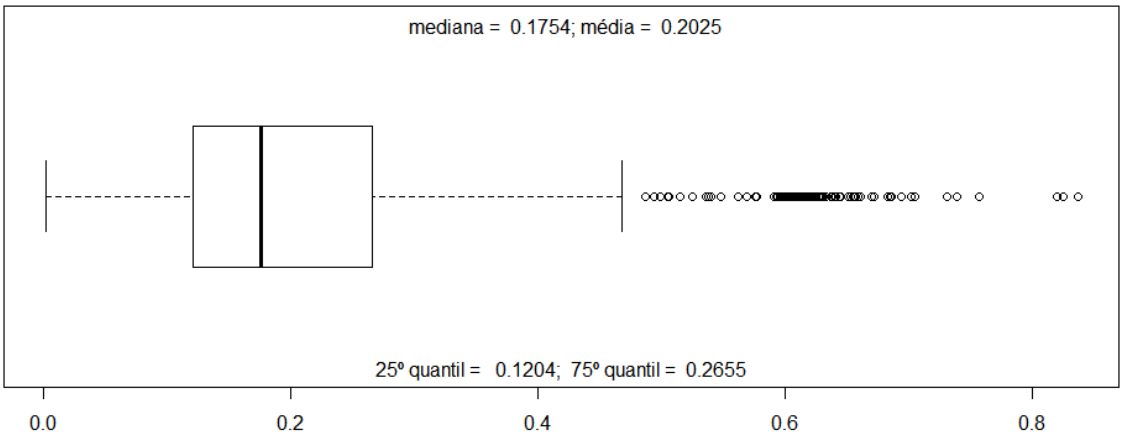

Figura 4.7: Análise estatística das informações considerando a função Elliott e os pesos convertidos para binário em complemento de dois.

Observando a **Figura 4.5**, **Figura 4.6** e **Figura 4.7**, pode-se verificar como a transição entre a RNA desenvolvida na ferramenta RStudio até a sua implementação no FPGA se comporta quanto a perda na precisão da estimativa de posição do sensor alvo. Considerando anteriormente as **Figura 4.5** e **Figura 4.7** em termos práticos, a estimativa de posição do alvo através dos quatro sensores âncoras na implementação difere, em distância média, apenas 0.0625 metros da estimativa da RNA, gerada anteriormente na ferramenta RStudio. Vale ressaltar que nas três figuras aqui comentadas os valores discrepantes são todos positivos devido a diferença de distância entre a posição real do alvo e as estimativas da RNA serem positivas. Estas discrepâncias caracterizam a distribuição apresentada nas três figuras como sendo positivamente simétricas, pois a mediana desta distribuição é afetada pelos valores discrepantes mais à direita. Novamente, em termos práticos, esta última análise mostra que a mediana entre a diferença de posição real e estimada é afetada por valores altos (de distância) com pouca frequência.

A seguir, é realizado um estudo de caso prático utilizando os sensores âncoras para estimar a posição do alvo em 16 localidades simetricamente espaçadas dentro da área de experimento, considerando a RNA implementada no FPGA. Considera-se no próximo tópico também a falta de informações provenientes de alguns dos sensores âncoras, como explicado na descrição do método de localização proposto, no **Capítulo 3**.

4.4 Estudo de caso para os testes e verificação de erros do método

Para realizar este teste, foram escolhidos 16 pontos espaçados simetricamente dentro da área experimentada. Um alvo foi posicionado nesses pontos e os sensores âncoras coletaram a mensagem deste alvo e executaram o método de localização. A escolha desses pontos tentou considerar uma estimativa da aproximação de localização para toda a área do experimento, sem necessariamente utilizar uma quantidade excessiva de pontos de estimativa. A Figura **4.8** ilustra os locais onde o sensor alvo foi posicionado dentro da área do experimento para a execução do método de localização.

Figura 4.8: Setas brancas indicam a posição dos âncoras. Pontos pretos ilustram as 16 posições onde o alvo foi posicionado para testes.

A Figura **4.9** mostra os resultados da estimativa dos pontos de localização na região de experimentação, incluindo os casos para nenhuma falha de sensores âncoras, falha no terceiro âncora e falha no terceiro e quarto âncoras.

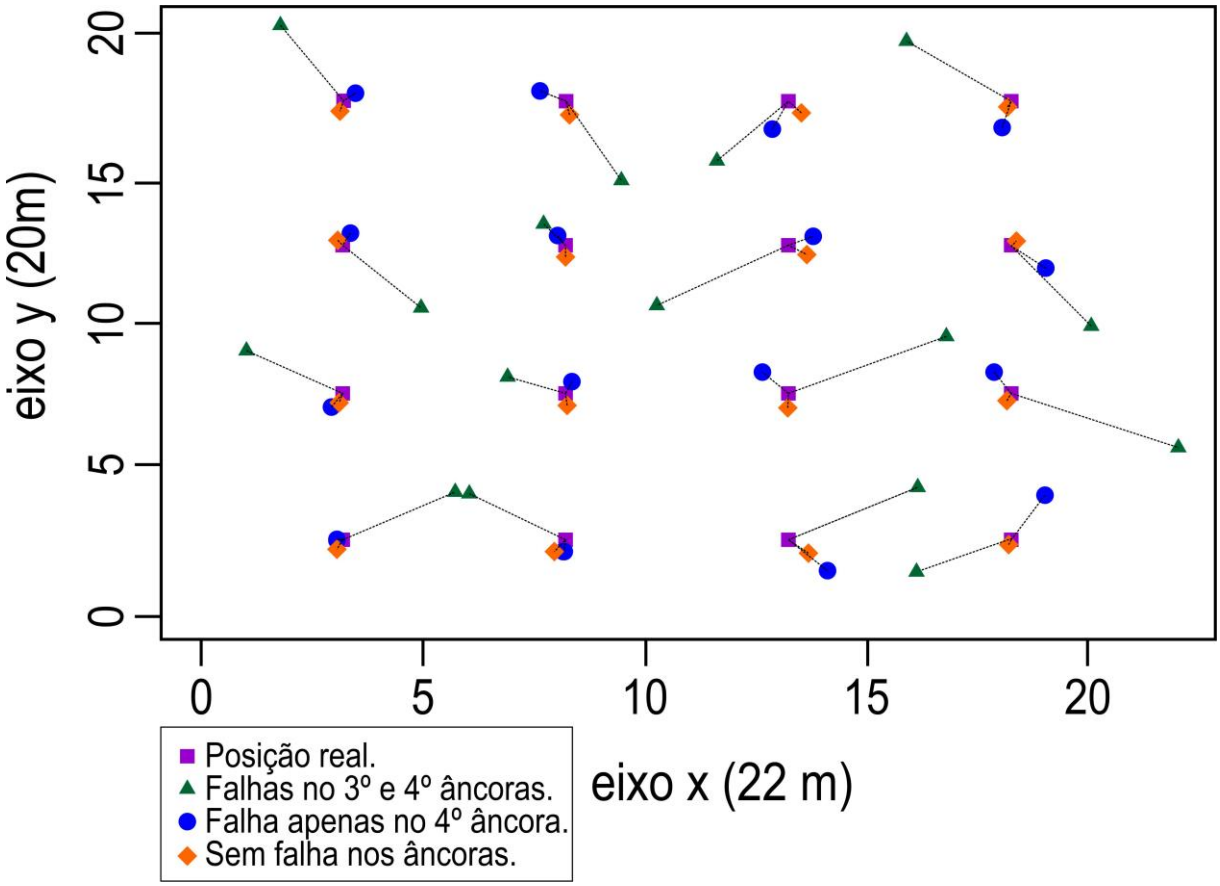

Figura 4.9: Resultados da estimativa da posição do alvo para os 16 pontos no experimento ilustrado na Figura 4.8.

Pode-se observar visualmente na Figura **4.9** que a estimativa sem falhas apresenta mais precisão e acurácia em comparação com os outros casos. A falha de apenas o quarto âncora também oferece, consideravelmente, boa precisão e acurácia, considerando uma aproximação. A falha do terceiro (e, consequentemente, do quarto) âncora tem uma imprecisão considerável, ainda sim, para algumas aplicações este erro pode ser aceitável.

A Figura **4.10** a seguir apresenta a ordenação dos pontos apresentados nos testes na Figura **4.9** com o intuito de apresentar o erro de distância considerando cada posição. Esta análise de erro é apresentada em seguida na **Figura 4.11**.

Figura 4.10: Posições ordenadas do alvo para os casos de teste da Figura 4.9, considerando a análise na Figura 4.11.

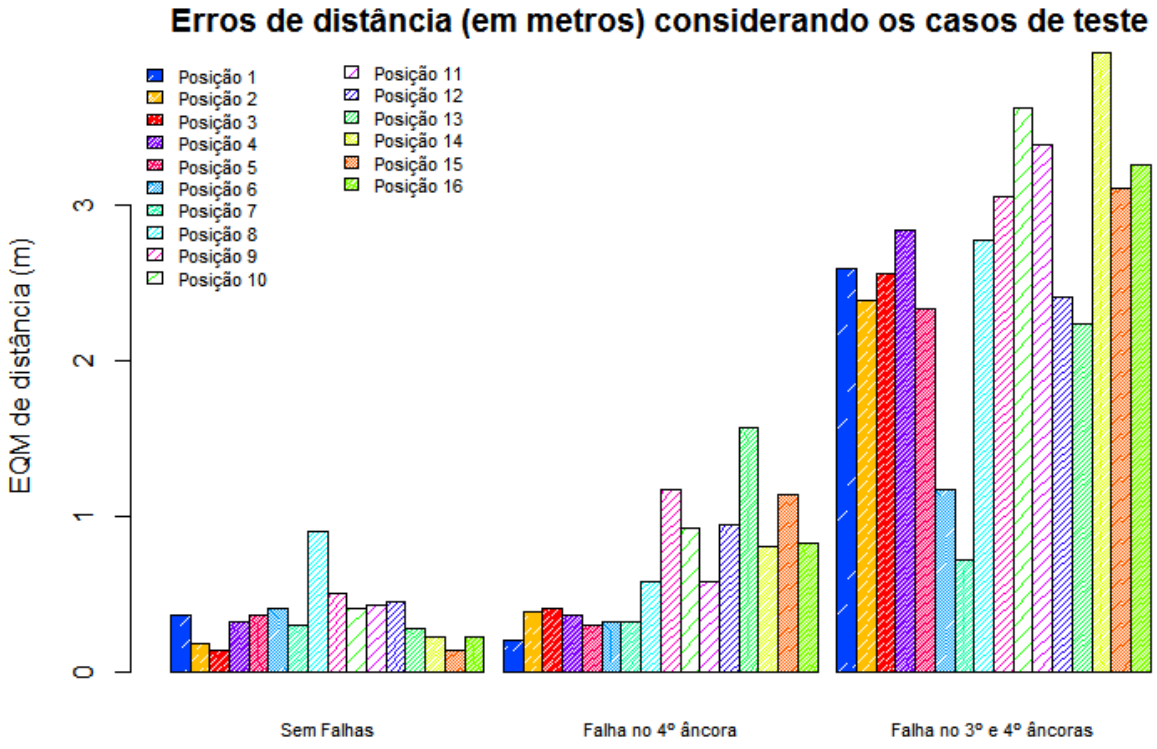

Figura 4.11: Erro de distância considerando todos os 3 casos apresentados na Figura 24.

Na Figura **4.11**, para o caso onde não ocorre falhas nos sensores âncoras, ou seja, os quatro âncoras participam do processo de localização, a estimativa de posição do alvo se mostrou melhor onde os pontos do experimento localizaram-se nas proximidades dos sensores âncoras, ou seja, nas extremidades da região do experimento. Isso ocorre devido ao ruído no sinal propagado do alvo é menor para alguns dos sensores âncoras que estão próximos, contribuindo para uma melhor leitura da potência do sinal e consequentemente, passando um valor mais próximo do real para a RNA. Contudo, em posições extremamente próximas aos sensores âncoras podem ocorrer perda de exatidão de leitura no sinal devido à alta potência do mesmo, a qual não possui muita variação em curtas distâncias. Para o segundo caso, na Figura **4.11**, onde ocorre falha no quarto sensor âncora, o erro de aproximação da posição foi relativamente maior em alguns pontos afastados do quarto sensor âncora. Uma hipótese razoável é que, para estes pontos, a função de aproximação da RNA utiliza principalmente a potência do sinal do quarto sensor âncora como entrada, associada aos parâmetros livres responsáveis por aproximar melhor o resultado nestas regiões. Similarmente, na Figura **4.11**, quando ocorrem falhas no terceiro e quarto sensores âncora, observa-se que mais posições na região de experimento são dependentes do sinal de potência recebido por estes dois sensores. Relembra-se aqui que falhas nos sensores âncoras significa associar zeros na sua respectiva entrada na RNA e que a falta desta informação de entrada para a RNA se sobressai no fato de as posições estarem ou não estarem próximas aos sensores âncoras considerados na aferição da aproximação.

Para uma análise mais detalhada do desempenho da rede neural, a Figura **4.12** apresenta um mapa de regiões de erros, considerando os casos de teste onde os sensores âncoras não falham.

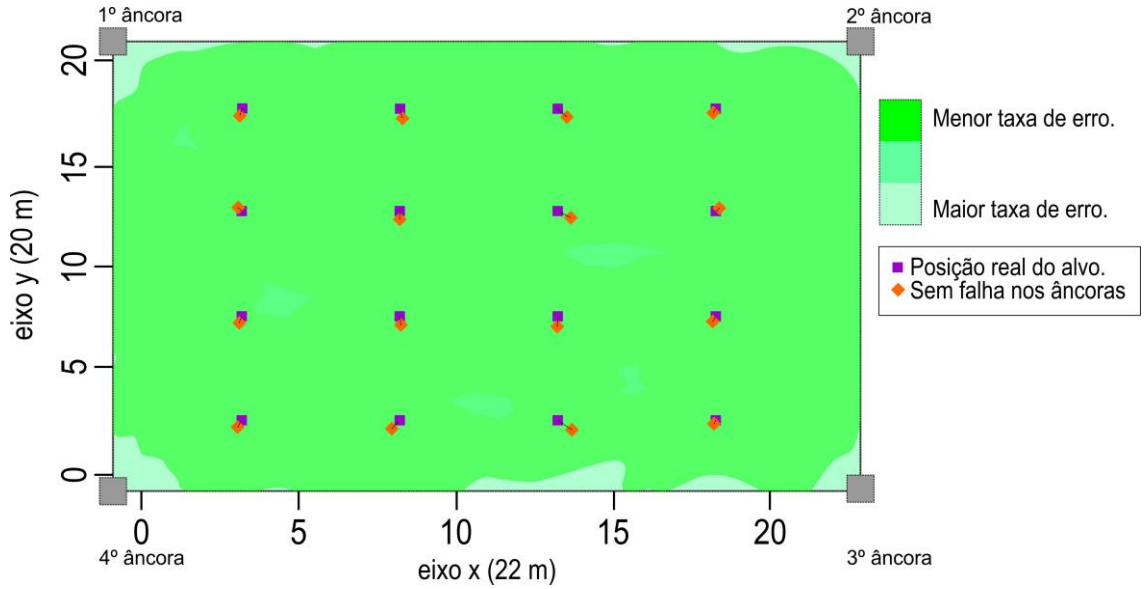

Figura 4.12: Regiões de taxa de erro de resposta sem falhas nos sensores âncora.

Na Figura **4.12** foram considerados valores de *threshold* de 0,5m para o menor erro, entre 0,5m e 1m para erros medianos e, acima de 1m, classificou-se como maior taxa de erros. Seguindo similarmente esta mesma classificação na Figura **4.13**, para o caso onde ocorre falha apenas no quarto sensor âncora.

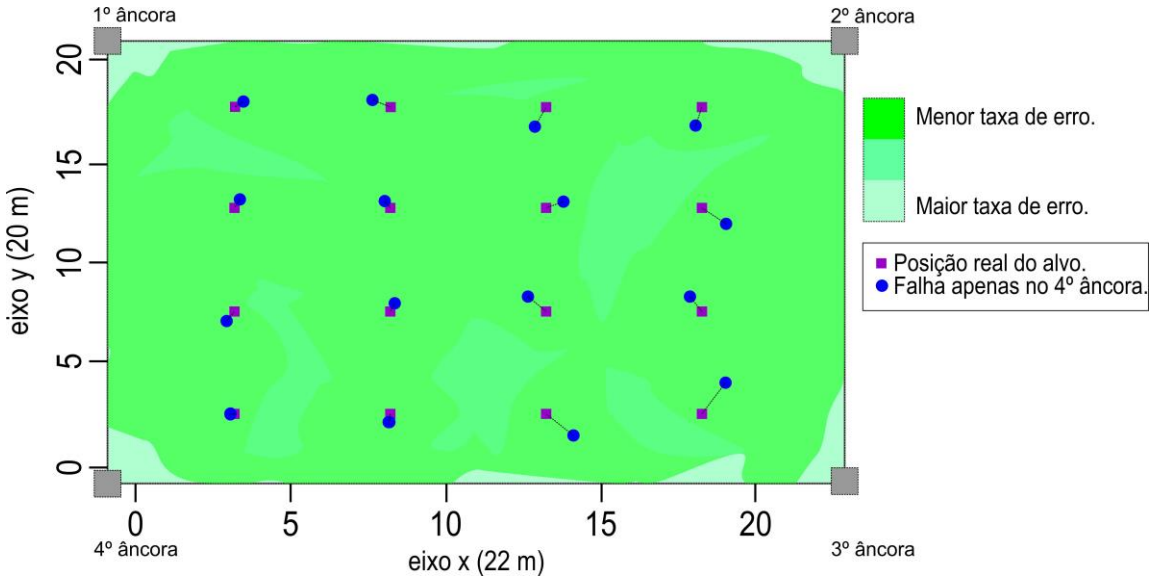

Figura 4.13: Regiões da taxa de erro de resposta considerando falha apenas no quarto sensor âncora.

Continuando, para o caso de falhas no terceiro e no quarto sensor âncora, Figura **4.14**, a melhor aproximação ocorre nas regiões de menor taxa de erro, ou seja, para as posições que não dependem tanto destes sensores para estimar a posição do alvo.

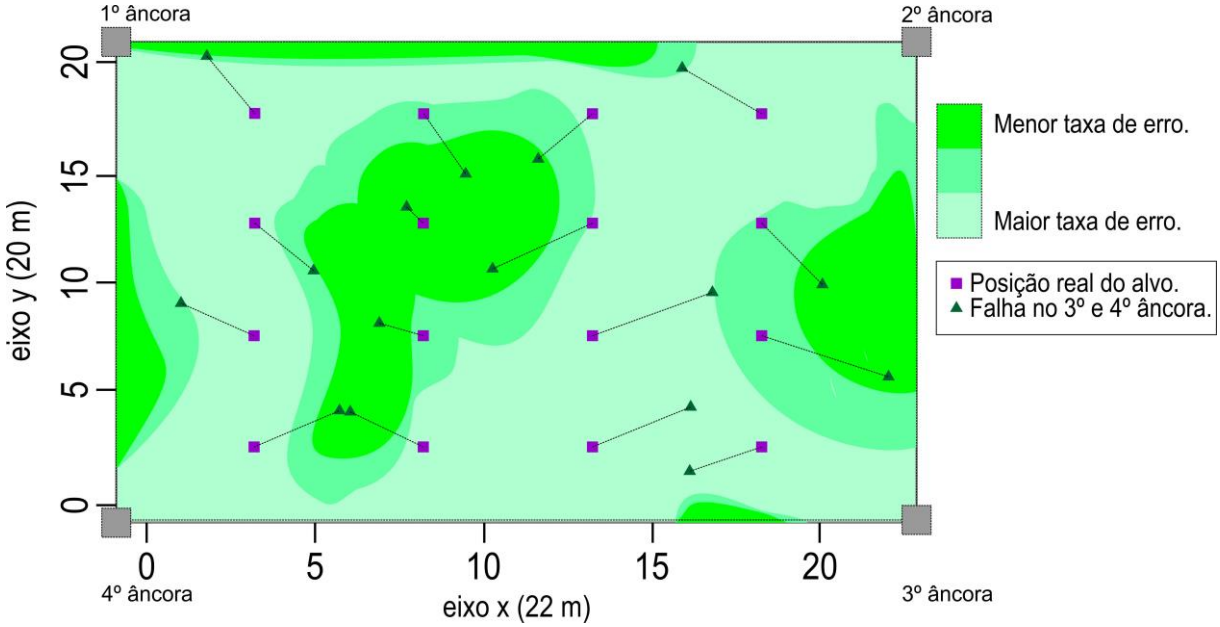

Figura 4.14: Regiões de taxa de erro de resposta, considerando falhas no 3º e 4º sensor âncora.

Além disso, será observado a seguir que a maior parte do EQM total está associado a representação binária dos pesos da RNA no FPGA e que este EQM pode ser diminuído considerando o aumento da quantidade de bits utilizada nesta representação.

Considerando o estudo de caso descrito acima, calculou-se o erro quadrático médio de distância Euclidiana (EQM) para os 16 pontos estimados chegando-se aos resultados apresentados na **Tabela 4.3**.

Tabela 4.3: Erros em metros, dadas as falhas em alguns âncoras par ao estudo de caso apresentado.

	Sem falhas nos âncoras.	*Falha no quarto âncora.*	*Falha no terceiro e, consequentemente, no quarto âncora.*
Erro no eixo X.	0,189077	0,199093	0,203451
Erro no eixo Y.	0,177593	0,168486	0,194702

A seguir, uma discussão e análise sobre o tempo de execução do método será apresentada, através da simulação do método descrito no FPGA e de considerações acerca da propagação do sinal entre os sensores que participam da estimativa de localização, evidenciando que o tempo de execução dependerá da distância entre os sensores âncoras na aplicação.

4.5 Simulação e Tempo de Execução da RNA

Aqui serão observados o tempo de execução de cada neurônio, da RNA e do controlador.

Relacionado com o tempo de execução, o neurônio foi simulado utilizando a ferramenta *ModelSim*^{TM}, da *Altera*^{TM}. O tempo execução para as camadas com dois e com quatro neurônios foi de 452 pico segundos. Este tempo de processamento é a mesma para cada camada, a qual executa todos os neurônios em paralelo. O tempo total de execução da RNA foi de 1,4 nano segundos, conforme a Figura **4.15**.

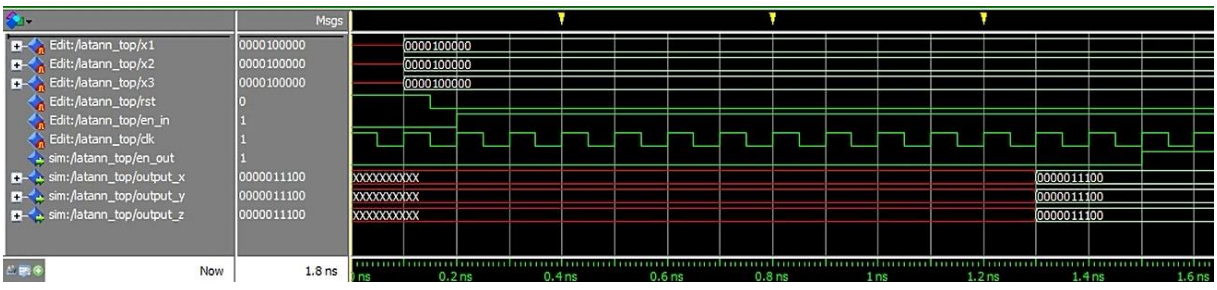

Figura 4.15: Temo de execução da RNA implementada.

Contudo, somados o tempo de execução da máquina de estados do controlador mais o tempo de propagação do sinal considerando a distância entre os sensores âncoras do experimento, o tempo médio total de execução do método proposto foi observado na simulação como em torno de 4 segundos. De acordo com o *datasheet* do rádio Xbee e através dos testes práticos realizados em laboratório, o tempo em que o rádio Xbee retorna a potência do sinal transmitido é de, em média, 1 segundo. Observou-se empiricamente que esse tempo de resposta do rádio, em cada sensor âncora, foi em média de 700 milissegundos. Diante desta análise, consideramos como trabalho futuro uma possível leitura da potência do sinal diretamente via hardware em contrapartida com a forma de aquisição via *firmware* (do rádio) utilizada em nosso experimento. Se este procedimento de leitura de potência for financeiramente viável na aplicação, a potência pode ser lida diretamente de um registrador no dispositivo de rádio sem a necessidade de uma requisição via software – e, consequentemente, ganho em tempo de execução. Contudo, uma análise sobre o *tradeoff* entre o tempo de execução e o custo de um recurso adicional de hardware para coletar esta potência precisaria ser realizado. O tempo de propagação do sinal do rádio no meio físico é um problema inerente relacionado com qualquer método em problemas de localização. Quanto maior a distância entre o alvo e os sensores âncoras que participam do processo de localização, maior será o tempo necessário para a estimativa de posição.

Considerando os resultados alcançados neste trabalho, principalmente quanto aos EQMs, ao tempo de execução e tolerância a falhas do método de localização explorado, no capítulo seguinte, realiza-se os comentários quanto as considerações finais e quanto aos trabalhos futuros.

5 Considerações Finais e trabalhos futuros

Este trabalho buscou desenvolver um novo método de localização distribuída abordando a utilização de FPGAs com RNA treinadas para lidar com falta de informações de entrada, emulando a falha de certos sensores âncoras. Quanto a implementação do método de localização, observou-se que não houve considerável complexidade além do esperado na fase planejamento do pré-projeto, considerando o desenvolvimento do método, dos testes e da análise dos resultados. Pode-se ressaltar que a parte mais trabalhosa foi a do experimento onde foram coletadas as amostras dos 16 pontos do sensor alvo.

Quanto aos resultados alcançados, o trabalho mostrou como a RNA pode ser aplicada em problemas de localização em RSSF. Foram apresentados diferentes resultados considerando os erros gerados devido a falhas nos sensores âncoras e o tempo de execução do método de localização. Quanto ao tempo de execução, mostrou-se através de simulações na plataforma FPGA que tempo de resposta o método de localização está diretamente dependente da distância entre os sensores e ao tempo de propagação do sinal. Com relação a capacidade de aproximação da posição real do alvo, os resultados foram apresentados na forma de erros de distância entre a localização real do alvo e a localização estimada pelo método, considerando os casos de falhas citados. Mesmo para casos críticos como a falha de dois sensores âncoras, observamos que o erro de distância pode ser controlado de forma eficiente aumentando a quantidade de bits que representam os pesos da RNA no FPGA. Além disso, mesmo considerando falhas em dois sensores âncoras, a resposta do método de localização recai sobre a área do experimento. Observou-se também que a relação do crescimento do erro de distância com a proporção de falhas dos sensores âncoras existe quando esses âncoras possuem uma participação maior na função de aproximação representada pela RNA, durante a estimativa da posição do alvo – ainda sim – utilizando três âncoras em LoS com o alvo são suficientes para uma ótima aproximação de posição.

5.1 Publicações geradas a partir deste trabalho

Três publicações foram geradas relacionadas a este trabalho. Em (MACHADO, LARRAT e MONTEIRO, 2013) foram realizados testes e análises com três algoritmos para estimar a localização de sensores – KNN, lateração e RNA - dos quais a RNA mostrou-se com melhores

resultados. Uma outra análise similar, contudo, mais refinada, considerando um pré-processamento das informações *a priori*, foi publicada em (E SILVA, MACHADO e MONTEIRO, 2013), apresentando a RNA como a melhor opção de escolha para o caso estudado. Os resultados de implementação, de testes e de análise deste trabalho foram publicados em (E SILVA, 2015).

5.2 Trabalhos futuros

De todas as melhorias citadas a mais importante está relacionada com o aumento da representação de bits utilizada para descrever os pesos da RNA no FPGA. Observou-se que esta melhoria diminuiria o valor do EQM total, para um valor quase igual a zero. Esta melhoria deve ser aplicada diretamente no modelo do neurônio descrito no FPGA e na forma de ajustes no tamanho das palavras de bits utilizada na comunicação entre os módulos do sistema.

Outra melhoria importante é considerar a comunicação entre os sensores âncoras em LoS, mas não necessariamente em sequência ordenada, permitindo que a RNA considere em seu treinamento fornecer uma resposta para casos de falhas, por exemplo, no 2º e 4º sensores âncoras. O limite que este trabalho alcançou foi a estimativa utilizando pelo menos dois âncoras, mas devido a análise dos resultados, uma pesquisa está sendo avaliada considerando alvos móveis e apenas um sensor âncora, baseada na variação de potência entre dois sensores e a posição conhecida entre um deles.

O método proposto pode ser estendido para vários canais, se possível, permitindo a execução de múltiplas instâncias do mesmo método de localização no FPGA para alvos diferentes. No caso de não haver mais de um canal no rádio (limitação de recursos de hardware) mais de uma RNA pode ser descrita no mesmo FPGA, utilizando o ID de cada alvo para direcionar a sua potência para a respectiva RNA. No **Apêndice 6F**, é possível verificar que pouquíssimo recurso do FPGA foi utilizado na implementação do método de localização aqui descrito, possibilitando a implementação desta melhoria.

O número de âncoras pode ser aumentado e agrupados para estimar a posição de alvos em *clusters*, operando como *cluster heads*, por exemplo. Contudo, em uma análise superficial considerando alvos móveis, RNA precisaria realizar um treinamento em tempo de execução para poder considerar a mudança de região dos *cluster heads* durante o seu agrupamento dinâmico na rede.

Uma alternativa para aproximar mais ainda o resultado da estimativa de localização do resultado valor real pode ser através do cálculo de média entre a estimativa dos valores das RNAs em cada âncora. Para tal, cada sensor âncora enviaria um byte a mais na sua transmissão, onde este byte adicional seria o resultado da sua estimativa de posição. Como exemplo, o segundo sensor âncora pode enviar ao terceiro sensor âncora a sua estimativa e este último, enviar ao quarto sensor âncora a sua estimativa e a estimativa do segundo sensor. O quarto sensor âncora pode então, realizar uma média aritmética entre as estimativas, diminuindo consideravelmente o EQM. Este método poderia ser considerado também diante da falha de sensores âncoras e a média de estimativas sendo aplicada apenas aos sensores que não falharam.

A flexibilidade do FPGA na implementação da RNA permite que esse trabalho seja estendido a outros novos experimentos para obter resultados superiores, tal qual realizar múltiplas localizações de alvos, aplicando este mesmo método para executar esta tarefa.

6 Bibliografia

ALHMIEDAT, T.; SALEM, A. O. A.; TALEB, A. A. AN IMPROVED DECENTRALIZED APPROACH FOR TRACKING MULTIPLE MOBILE TARGETS THROUGH ZIGBEE WSNS. **International Journal of Wireless & Mobile Networks (IJWMN)** , v. 5, n. 3, June 2013.

BEIU, V. E. A. Closse Approximations of Sigmoid Functions by Sum of Step for VLSI Implementation of Neural Networks. **The Scientific Annals, Section: Informatics**, v. 40, n. 1, p. 5-34, 1994.

BHARDWAJ, S. ANN for Node Localization in Wireless Sensor Network. **International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 5**, v. 2, 2013.

BOUKERCHE, A. E. A. Localization systems for wireless sensor networks. **Wireless Communications, IEEE**, v. 14, n. 6, p. 6-12, 2007.

CHAN, F.-K.; WEN, C.-Y. **Adaptive AOA/TOA Localization Using Fuzzy Particle Filter for Mobile WSNs**. Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd. [S.l.]: IEEE. 2011. p. 1-5.

CHENG, L. E. A. A Survey of Localization in Wireless Sensor Network. **International Journal of Distributed Sensor Networks**, v. 2012, n. Article ID 962523, p. 12, 2012.

CHO, I. **Hardware and software architectures for energy-and resource-efficient signal processing systems**. [S.l.]: [s.n.], 2014.

CHU, P. P. **RTL Hardware Design Using VHDL:** Coding for Efficiency, Portability, and Scalability, capítulo 6.1.2 e 6.1.3. 1. ed. [S.l.]: Wiley Interscience, 2006.

CSÁJI; CSANÁD, B. Approximation with artificial neural networks. **Faculty of Sciences, Etvs Lornd University, v.24**, Hungary, 24, 2001.

E SILVA, M. R. L. F. E. A. Distributed Target Location in Wireless Sensors Network: An Approach Using FPGA and Artificial Neural Network. **Wireless Sensor Network**, v. 7, n. 5, p. 35, 2015.

E SILVA, M. R. L. F.; MACHADO, L. S.; MONTEIRO, D. C. Performance Evaluation of Lateration, KNN and Artificial Neural Networks techniques Applied to Real Indoor and Outdoor Location in WSN. **Journal of Communication and Computer**, v. 10, p. 72-81, 2013.

FAN, J.; ZHANG, B.; DAI, G. D3D-MDS: A Distributed 3D Localization Scheme for an Irregular Wireless Sensor Network Using Multidimensional Scaling. **International Journal of Distributed Sensor Networks**, 2015. 10.

FRERY, A. C. E. A. Data Driven Performance Evaluation ofWireless SensorNetworks. **Sensors**, v. 10, n. 3, p. 2150 - 2168, 2010.

GHADIMI, E. E. A. **A metric for opportunistic routing in duty cycled wireless sensor networks**. Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on.. [S.l.]: IEEE. 2012. p. 335-343.

GHOLAMI, M.; CAI, N.; BRENNAN, R. W. An artificial neural network approach to the problem of wireless sensors network localization. **RoboticsandComputer-IntegratedManufacturing, ELSEVIER**, v. 29, n. 1, p. 96-109, 2013.

GYBENKO, G. Approximations by superpositions of sigmoidal functions. **Mathematics of Control, Signals, and Systems**, v. 2, n. 4, p. 303-314, 1989.

HAN, K. E. A. Algorithm Design for Data Communications in Duty-Cycled Wireless Sensor Networks: A Survey. **Communications Magazine, IEEE** , v. 51, n. 7, p. 107 - 113, 2013.

HORNIK, K. Approximation Capabilities of Multilayer Feedforward Networks. **Neural Networks**, v. 4, n. 2, p. 251–257, 1991.

HSU, H.-C. E. A. **TOA estimation with DLC receivers for IEEE 802.15.4a UWB systems**. Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on. [S.l.]: IEEE. 2011. p. 424–428.

HUANG, G. et al. Trends in extreme learning machines: A review. **Neural Networks**, v. 61, p. 32-48, 2015.

I. DEL CAMPO, R. F. J. E. A. K. B. Controlled Accurary Approximation of the Sigmoid Function for Efficient FPGA-based Implementation of Artificial Neuron. **Electronics Letters, IEEE**, v. 49, n. 25, p. 1598 - 1600, December 2010.

JAMEL et al. **FPGA Based Neural Wireless Sensor Network**. The 13th International Arab Conference on Information Technology. [S.l.]: [s.n.]. 2012.

KUMAR, S.; LEE, S. R. **Localization with RSSI Values for Wireless Sensor Networks:** An Artificial Neural Network Approach. International Electronic Conference on Sensors and Applications. [S.l.]: Multidisciplinary Digital Publishing Institute. 1 June 2014.

KUMAR, S.; LEE, S.-R. Localization With RSSI values for Wireless Sensor Networks: An Artificial Neural Network Approach. **Sensors**, 2014.

LAI, S. **Duty-Cycled Wireless Sensor Networks:** Wakeup Scheduling, Routing, and Broadcasting. [S.l.]: Virginia Polytechnic Institute and State University, 2010.

MACHADO, L.; LARRAT, M.; MONTEIRO, D. **Performance Evaluation of Lateration, KNN and Artificial Neural Networks Techniques Applied to Real Indoor Localization in WSN**. INNOV 2013 : The Second International Conference on Communications, Computation, Networks and Technologies. Portugal: [s.n.]. 2013.

MAMUN, Q. A Qualitative Comparison of Different Logical Topologies for Wireless Sensor Networks. **Sensors**, v. 12, n. 11, p. 14887-14913, 2012.

MANISH PANICKER, C. B. Efficient FPGA Implementation of Sigmoid and Bipolar Sigmoid. **IOSR Journal of Engineering (IOSRJEN)**, 2, n. 6, 2012. 1352-1356.

MOHAMMAD; PARK, Y.; KI-DOO, K. RSS-Based Indoor Localization Algorithm for Wireless Sensor Network Using Generalized Regression Neural Network. **Arab J Sci Eng (2012)**, n. 37, p. 1043–1053, 2012.

NORDIO, A.; CHIASSERINI, C.-F.; MUSCARIELLO, A. **Signal compression and reconstruction in clustered sensor networks**. Communications, 2008. ICC'08. IEEE International Conference on.. [S.l.]: IEEE. 2008. p. 925–929.

PAL, A. Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges. **Network Protocols and Algorithms**, v. 2, n. 1, p. 45-73, 2010.

PATHAK, A.; PRASANNA, V. K. Energy-Efficient Task Mapping for Data-Driven Sensor Network Macroprogramming. **Computers, IEEE Transactions on**, v. 59, n. 7, p. 955-968, 2010.

PRASAD, P. Recent trend in wireless sensor network and its applications: a survey. **Sensor Review**, v. 35, n. 2, p. 229-236, 2015.

RAVINDRA, S.; JAGADEESHA, S. N. TIME OF ARRIVAL BASED LOCALIZATION IN WIRELESS SENSOR NETWORKS: A LINEAR APPROACH. **Signal & Image Processing : An International Journal (SIPIJ)**, v. 4, n. 4, August 2013.

REZAEI; ZAHRA; MOBININEJAD, S. Energy Saving in Wireless Sensor Networks. **International Journal of Computer Science & Engineering Survey (IJCSES)**, v. 3, n. 1, Fevereiro 2012.

RIEDMILLER, M.; BRAUN, H. **RPROP-A fast adaptive learning algorithm**. Proceedings of the International Symposium on Computer and Information Science VII. [S.l.]: [s.n.]. 1992.

SAAD, A.-M. H. Y.; ALHADY, S. S. N. **Embedded Neural Network for Distance Recognition Using Distance Sensor**. 1st International Conference of Recent Trends in Information and Communication Technologies, IRICT (2014). Universiti Teknologi Malaysia, Johor, Malaysia: [s.n.]. 2014. p. 182-191.

SABTO, N. A.; AL MUTIB, K. Autonomous mobile robot localization based on RSSI measurements using an RFID sensor and neural network BPANN. **Journal of King Saud University-Computer and Information Sciences**, v. 25, n. 2, p. 137–143, October 2013.

SAHOO, B. P. S.; RATH, S.; PUTHAL, D. Energy Efficient Protocols for Wireless Sensor Networks: A Survey and Approach. **International Journal of Computer Applications**, v. 44, n. 18, p. 43-48, 2012.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. **Neural Networks**, v. 61, p. 85-117, 2015.

SIBI, P.; JONES, S. A.; SIDDARTH, P. Analysis of different activation functions using back propagation neural networks. **Journal of Theoretical and Applied Information Technology**, v. 4, n. 3, p. 1264-1268, 2013.

SINGH, H.; BISWAS, B. Comparison of CSMA based MAC protocols of wireless sensor networks. **International Journal of AdHoc Network Systems**, v. 2, n. 2, 8 Abril 2012.

SUNG, Y.; POOR, H. V.; YU, H. How much information can one get from a wireless ad hoc sensor network over a correlated random field? **Information Theory, IEEE Transactions on,** v. 55, n. 6, p. 2827–2847, 2009.

TANG et al. **Efficient power management for Wireless Sensor Networks:** A data-driven approach. Local Computer Networks, 2008. LCN 2008. 33rd IEEE Conference on. IEEE, 2008. [S.l.]: [s.n.]. 2008. p. 106-113.

XIE, Z. **A Non-linear Approximation of the Sigmoid Function based on FPGA**. IEEE fifth InternationalConference onAdvanced Computational Intelligence(ICACI). Nanjing, Jiangsu, China: IEEE. 2012. p. 221-223.

XU, J. E. A. ,Distance Measurement Model Based on RSSI in WSN. **Wireless Sensor Network**, v. 2, n. 8, p. 606, 2010. ISSN 8.

ZEN, K. E. A. Intelligent Coordinator Selection Mechanism (ICSM) for IEEE802. 15.4 Beacon-Enabled MAC Protocol in Mobile Wireless Sensor Networks. **International Review on Computers and Software (IRECOS)** , v. 10, n. 2, p. 164-173, 2015.

ZHOU, N.; ZHAO, X.; TAN, M. **Deployment and Routing Method for Fast Localization Based on RSSI in Hierarchical Wireless Sensor Network**. Mobile Ad-Hoc and Sensor Systems (MASS), 2013 IEEE 10th International Conference on. [S.l.]: IEEE. 2013. p. 614-619.

APÊNDICE

A. UART

Na comunicação serial, cada bit da mensagem é enviado por vez, necessitando de apenas um canal de comunicação. Quando um grupo de bits que compõe uma informação na mensagem chega ao processador (ou seja, é armazenado em um registrador), o processador ou controlador pode interpretar tal informação e realizar uma operação. Esse procedimento pode ser representado, sem profundidade de detalhes, na Figura A.**0**.**1**, a seguir.

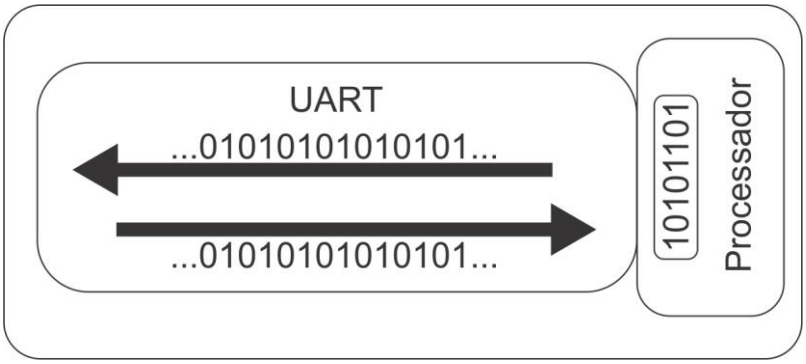

Figura A.0.1: UART transmite/recebe os bits serialmente. O Processador agrupa esses bits em uma unidade de informação dentro de um Registrador.

Para que o sistema detecte que um grupo de bits está sendo recebido, é necessário que uma sinalização ocorra no canal. Na UART, o sinal do canal assume o nível lógico 1

(representado por um valor de tensão elétrica na linha) para informar que o canal está ocioso. Quando ocorre uma transmissão serial de grupo de bits, o canal é sinalizado pelo nível lógico 0 (alteração no valor de tensão elétrica no canal), indicando o bit de início da mensagem. Em seguida, um número conhecido de bits é transmitido em sequência ordenada, os quais representam a informação em um registrador dentro do sistema. Para finalizar a transmissão deste grupo de informação, uma alteração lógica no final da mensagem sinaliza o final da transmissão/recepção de dados (bit de parada). Opcionalmente, um bit de paridade pode ser adicionado à informação serial, indicando se ocorreram erros na mensagem. A **Figura A.2** a seguir apresenta a estrutura de uma unidade de informação serial da UART.

Bit de início	Bit de dados 0	Bit de dados 1	Bit de dados 2	Bit de dados 3	Bit de dados 4	Bit de dados 5	Bit de dados 6	Bit de dados 7	Bit de Paridade	Bit de parada

Figura A.0.2: Unidade de informação enviada/recebida serialmente pela UART.

B. Oversampling

A taxa de transmissão de dados da UART, conhecida como *baud rate*, comumente assume valores padronizados. A taxa de valores considerada neste trabalho é a de 115.200 bits por segundo (bps), que é a maior taxa de bits que é padronizada. A frequência de transmissão/recepção de bits associada a esta taxa de 115.200 bps é de aproximadamente 1.8MHz. O FPGA opera globalmente em 50MHz. Para que este módulo UART possa ser adequadamente agregado ao sistema, a modulação de frequência é necessária. Para tal, utilizou-se a técnica conhecida como *oversampling*, que consistem em amostrar cada bit em 1.8MHz da UART um número fixo de vezes, considerando os 50MHz do FPGA. Em palavras mais simples, cada bit da UART é amostrado várias vezes e apenas uma destas amostras é considerada no FPGA. Para diminuir o erro da amostra no recebimento do bit, evitando considerar que o sinal da UART passe à próxima amostra, o FPGA considera a amostra tomada no meio do tempo em que o bit da UART está sendo recebido. O *oversampling* considerado neste trabalho amostra 16 vezes cada bit da UART recebido no sistema. Tomando o valor do bit no meio das 16 amostras deste bit, ou seja, na oitava amostra, o oversampling reduz as chances para 1/16 de o valor do bit adquirido ultrapassar para fora de seus limites de transição de estado.

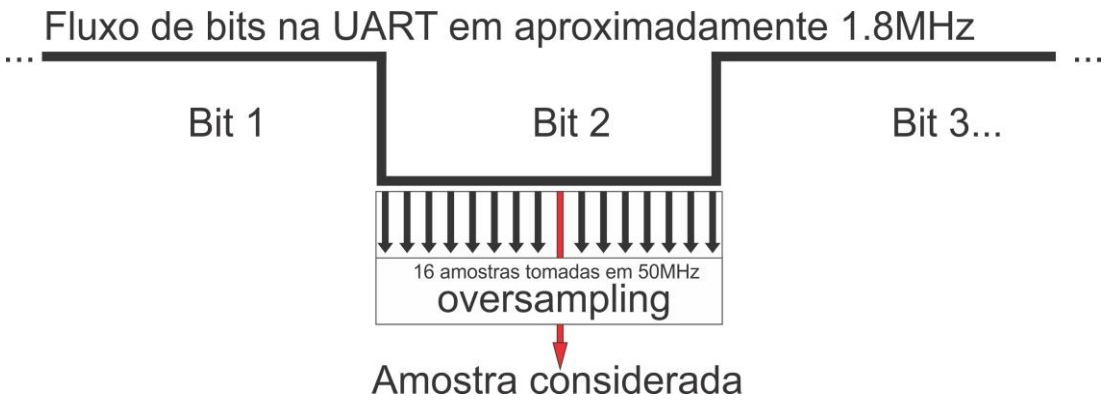

Figura B.1: Ilustração do oversampling. O fluxo de bits da UART (1.8MHz) é amostrado 16 vezes em 50MHz (FPGA). A amostra adquirida no meio de cada bit é a considerada, reduzindo o erro do bit em 1/16.

Na **Figura B.1**, cada uma das 16 amostras deve ocorrer a cada 27 ciclos de 50MHz. Na lógica de conversão 50 MHz (FPGA) para 1.841MHz (UART 115200 bps), um bit é transmitido em 0,000008681 segundos. Se cada bit for amostrado 16 vezes, cada amostra deve ocorrer em 0,000000543 segundos. Isso equivale a 1841620,626151013 Hertz, ou 1.841,620MHz ~ 1.841 MHz. Para que cada uma das 16 amostras (*oversampling*) de um bit ocorram em 0,000000543 segundos (ou 1.841 MHz), em um *clock* de placa de 50MHz, precisamos esperar 50MHz/1.841MHz ciclos (ou 27,1 ciclos) da placa para ler cada uma das 16 amostras de cada bit.

C. Buffer

Antes de o controlador processar os dados recebidos pela UART tais dados são armazenados em um *buffer* do tipo FIFO (*first* in *first* out – primeiro a entrar primeiro a sair). Já que o tempo de recepção de mensagens e o tempo de processamento do algoritmo são diferentes, a medida em que o controlador precisa de informações, este busca as mesmas no buffer, checando se a informação está disponível, conforme o padrão no formato das mensagens trocadas entre os sensores. A quantidade de memória utilizada para o *buffer* neste trabalho foi 256 bytes. Este valor foi considerado em testes empíricos, para que não houvesse perda de informações transmitidas e para checar se o aumento de memória afetaria consideravelmente os recursos da plataforma. Contudo, outros valores apenas para testes experimentais tais como 32 bytes e 64 *bytes* também foram usados para dimensionar o *buffer* sem a ocorrência de perda de informação. Cada vez que o controlador lê do *buffer* a mensagem do alvo e, em sequência, a mensagem do sensor âncora imediatamente antecessor, o controlador realiza um *reset* no *buffer*. Isso evita que o *buffer* estoure a capacidade de recepção de informações. O *buffer* é integrado entre o

receptor da UART e o controlador e opera na frequência de 50MHz juntamente com todo o sistema.

D. Estados definidos no Controlador

A descrição de cada estado é apresentada na **Tabela D.1** a seguir.

Tabela D.1: Descrição da máquina de estados que represente ao controlador.

Número	Nome do estado	Descrição
1	IDLE	Estado ocioso do controlador, enquanto a mensagem *broadcast* do alvo não foi detectada.
2	BUFFER_READ_ON	O controlador ativa a leitura de um byte do *buffer*.
3	BUFFER_READ_OFF	O controlador desativa a leitura de um byte do *buffer*.
4	BUFFER_CHK_SYM	O controlador checa se a mensagem é proveniente do alvo.
5	BUFFER_CLR_ON	O controlador ativa a limpeza do *buffer*.
6	BUFFER_CLR_OFF	O controlador desativa a limpeza do *buffer*.
7	BUFFER_CHK_CLR	O controlador checa se o *buffer* está limpo.
8	AT_CMD_MODE_ON	O controlador coloca o rádio em modo de comando para futuramente solicitar a potência da última mensagem recebida. No XBEE PRO, o comando "+++", representado pelo hexadecimal 0x2B, é enviado para a porta UART do rádio.
9	AT_CMD_WAIT_RESP	O controlador espera a resposta do rádio para saber se este entrou em modo de comando, para poder aceitar a solicitação da potência da última mensagem recebida.
10	AT_CMD_DELAY	*Delay* de um segundo utilizado no estado 9.
11	AT_CMD_BUFFER_CLR_ON	O controlador ativa a limpeza o buffer após receber a resposta do rádio confirmando a entrada em modo de comando.
12	AT_CMD_BUFFER_CLR_OFF	Desativa a limpeza no estado 11.
13	AT_CMD_BUFFER_CHK_CLR	O controlador checa se as operações nos estados 11 e 12 foram executadas.
14	ATDB_CMD_A	O controlador envia o byte referente ao caractere ASCII "A", relativo ao comando ATDB, par ao rádio XBEE PRO.

15	ATDB_CMD_T	O controlador envia o byte referente ao caractere ASCII ”T”, relativo ao comando ATDB, para o rádio XBEE PRO.
16	ATDB_CMD_D	O controlador envia o byte referente ao caractere ASCII ”D”, relativo ao comando ATDB, para o rádio XBEE PRO.
17	ATDB_CMD_B	O controlador envia o byte referente ao caractere ASCII ”B”, relativo ao comando ATDB, para o rádio XBEE PRO.
18	ATDB_CMD_r	O controlador envia o byte referente ao caractere ASCII ”/r”, relativo ao comando ENTER após o envio do comando ATDB, para o rádio XBEE PRO.
19	BUFFER_ATDB_RESP	O controlador checa no buffer se o rádio enviou a resposta ao comando ATDB, caso contrário, retorna ao estado 1.
20	BUFFER_RD_RSSI_ON	O controlador lê do *buffer* a resposta do rádio, a saber, o RSSI proveniente da mensagem *broadcast* do alvo.
21	BUFFER_RD_RSSI_OFF	Desativa a função executada no estado 20.
22	ASCII_HEX_RSSI	Converte o RSSI representado em ASCII para um valor hexadecimal.
23	BUFFER_WAIT_RSSI_A1	O controlador do segundo âncora aguarda o RSSI proveniente do primeiro âncora.
24	BUFFER_WAIT_RSSI_A1A2	O controlador do terceiro âncora aguarda o RSSI proveniente do primeiro e do segundo âncora.
25	BUFFER_WAIT_RSSI_A1A2A3	O controlador do quarto âncora aguarda o RSSI proveniente do primeiro, do segundo e do terceiro âncora.
26	BUFFER_RD_RSSI_ON	O controlador lê o RSSI da mensagem do âncora imediatamente antecessor. Conforme o ID que identifica o âncora, este estado é executado de forma particular, aguardando uma quantidade diferente de RSSIs na mensagem.
27	BUFFER_RD_RSSI_OFF	Desativa o estado 26.
28	ANN_A1A2_ACT	O controlador envia o RSSI do primeiro âncora e do segundo âncora para a rede neural.
29	ANN_A1A2_WAIT	O controlador espera a resposta da rede neural com as entradas relacionadas ao RSSI do primeiro e do segundo.
30	ANN_A1A2_RESP	O controlador retorna a posição do alvo estimada com as entradas do primeiro e do segundo âncora.

31	ANN_A1A2A3_ACT	O controlador envia o RSSI do primeiro, do segundo e do terceiro âncora para a rede neural.
32	ANN_A1A2A3_WAIT	O controlador espera a resposta da rede neural com as entradas relacionadas ao RSSI ao primeiro, do segundo e do terceiro âncora.
33	ANN_A1A2A3_RESP	O controlador retorna a posição do alvo estimada com as entradas do primeiro do segundo e do terceiro âncora.
34	ANN_A1A2A3A4_ACT	O controlador envia o RSSI do primeiro, do segundo, do terceiro e do quarto âncora para a rede neural.
35	ANN_A1A2A3A4_WAIT	O controlador espera a resposta da rede neural com as entradas relacionadas ao RSSI ao primeiro, do segundo, do terceiro e do quarto âncora para a rede neural.
36	ANN_A1A2A3A4_RESP	O controlador retorna a posição do alvo estimada com as entradas do primeiro, do segundo, do terceiro e do quarto âncora para a rede neural.
37	A1_TO_A2	O controlador envia o RSSI do primeiro âncora para o segundo âncora.
38	A2_TO_A3	O controlador envia o RSSI do primeiro e do segundo âncora para o terceiro âncora.
39	A3_TO_A4	O controlador envia o RSSI do primeiro, segundo e terceiro âncora para o quarto âncora.

E. Figura da descrição final do sistema na ferramenta de desenvolvimento

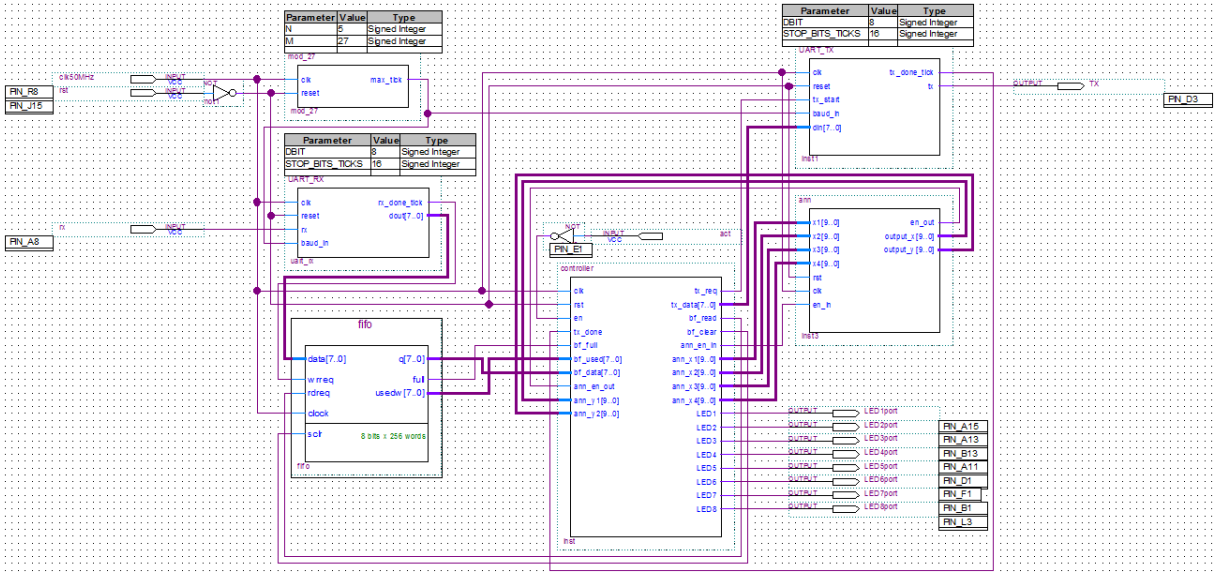

Figura E.1:Implementação do método de localização realizada no software Quartus II, versão 14, Altera

F. Consumo de recursos deste método de localização na plataforma FPGA

Flow Summary	
Flow Status	Successful - Tue Sep 15 13:07:47 2015
Quartus II 64-Bit Version	15.0.0 Build 145 04/22/2015 SJ Full Version
Revision Name	WSN
Top-level Entity Name	CONTROLLER_TX_RX
Family	Cyclone IV E
Device	EP4CE22F17C6
Timing Models	Final
Total logic elements	493 / 22,320 (2 %)
Total combinational functions	432 / 22,320 (2 %)
Dedicated logic registers	263 / 22,320 (1 %)
Total registers	263
Total pins	13 / 154 (8 %)
Total virtual pins	0
Total memory bits	2,048 / 608,256 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

Figura F.1: Recursos do FPGA utilizados considerando o experimento realizado neste trabalho.