Universidade Federal da Paraíba - UFPB
Centro de Informática
Programa de Pós-Graduação em Informática
Dissertação de Mestrado

Expandindo a Área de Cobertura de um Sistema Multi-Robôs através de redes Multicamadas e um Middleware de Base de Dados em Tempo Real

Leandro José Caetano

João Pessoa - Paraíba
Agosto de 2015

Expandindo a Área de Cobertura de um Sistema Multi-Robôs através de redes Multicamadas e um Middleware de Base de Dados em Tempo Real

Leandro José Caetano

Dissertação de Mestrado apresentada à banca examinadora do Programa de Pós-graduação em Informática da Universidade Federal da Paraíba como requisito para obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Tiago P. Nascimento.

João Pessoa - Paraíba

Agosto de 2015

Caetano, Leandro José.

Expandindo a Área de Cobertura de um Sistema Multi-Robôs através de redes Multicamadas e um Middleware de Base de Dados em Tempo Real

85 páginas

Dissertação (Mestrado) - Universidade Federal da Paraíba - Centro de Informática - Programa de Pós-Graduação em Informática.

1. Redes Multicamadas
2. Redes Ad Hoc
3. ZigBee
4. *Middleware* em tempo real
5. Multi-Robôs

I. Universidade Federal da Paraíba - Centro de Informática - Programa de Pós-Graduação em Informática.

Expandindo a Área de Cobertura de um Sistema Multi-Robôs através de redes Multicamadas e um Middleware de Base de Dados em Tempo Real

Leandro José Caetano

Dissertação de Mestrado apresentada à banca examinadora do Programa de Pós-graduação em Informática da Universidade Federal da Paraíba como requisito para obtenção do título de Mestre em Informática, aprovada em 31 de agosto de 2015.

Banca Examinadora:

Prof. Dr. Tiago P. Nascimento (Orientador)
UFPB

Prof. Dr. Iguatemi Eduardo da Fonseca
UFPB

Prof. Dr. Alisson Vasconcelos de Brito
UFPB

Prof. Dr. Frederico Miguel Santos
Instituto Politécnico de Coimbra

Dedico este trabalho à minha família: Clóvis, Tânia, Anderson e Andressa pelo amor e incentivo aos estudos.

Agradecimentos

À Deus, por ter iluminado os meus caminhos e me guiado em todas as decisões a serem tomadas desde o início da jornada.

Ao meu sobrinho Alex Caetano que, nos proporcionando momentos de felicidade, chegou para iluminar e alegrar a nossa família.

À minha mãe Tânia Caetano, que sempre deixou preenchida minha xícara de café, e por todo amor e dedicação.

Ao meu pai Clóvis Caetano, em quem sempre me espelhei e que, como eu, ama o mundo da tecnologia.

Ao meu irmão Anderson Caetano, que sempre transmitiu todo o seu apoio e motivação.

À minha esposa Andressa Caetano, que dedicou todo o seu carinho e contribuição, compreendendo as minhas ausências e as noites em claro.

Ao professor Dr. Tiago Nascimento, pela confiança concedida e por todas as orientações dadas a esse trabalho.

À equipe do Centro de Tecnologia da UFPB, pelo espaço e conhecimento compartilhado no dia-a-dia.

A todos os professores, pela dedicação, atenção e grandes ensinamentos prestados durante esse período.

*"A **curiosidade** e a **exploração***
devem ser prioridades para quem
decidir trabalhar com uma plataforma
***OpenSource**".*
Desconhecido

Resumo

A comunicação entre robôs é tarefa fundamental e de grande importância na robótica. É a partir dela que robôs conseguem descobrir e repassar informações sobre obstáculos encontrados no caminho, transmitir informações sobre um alvo que outro robô não consiga detectar ou determinar, em conjunto com outros robôs, a melhor rota a ser seguida, entre outros. Uma das grandes dificuldades, no contexto em que se tem a comunicação de robôs em uma rede sem fio, está em distanciá-los. Um deles, por exemplo, pode migrar para uma outra área a fim de realizar uma tarefa específica, distanciando-se, assim, da sua formação inicial e ultrapassando o limiar de cobertura de rede. Ao distanciar um robô do outro, os limites conferidos pelo meio acarretarão uma perda de sinal comprometendo a comunicação entre eles. Deste modo, conhecer a área de cobertura de uma rede sem fio onde robôs estarão conectados é um processo importante para a implantação de um sistema de comunicação entre robôs. Este trabalho propõe expansão da área de cobertura de uma rede sem fio, a ser utilizada em sistemas multi-robôs, usando um *Middleware* de Base de Dados em Tempo Real - RTDB, em redes multicamadas que utilizam os padrões IEEE 802.11 e 802.15.4, analisando o comportamento dessa expansão em relação a conectividade dos robôs, área de cobertura e de interferência de Redes Sem Fio. Os resultados obtidos demonstram a robustez e confiabilidade do cenário proposto e de acordo com os experimentos realizados, comprovou-se que o uso de redes multicamadas proporciona um maior alcance na área de cobertura de uma formação de robôs.

Palavras-chave: Redes Multicamadas, Redes Ad Hoc, ZigBee, *Middleware* em tempo real, Multi-Robôs

Abstract

The communication between robots is essential and of great interest in the robotics research field. It is from it that the robots can discover and pass on information about obstacles in their way, pass on to the other robots information about a target that other robot can not detect, determine together which is the best route to follow, among others, until the goals are achieved. The major difficulty in the context where robot communication exists in an Ad Hoc network is distancing the robots from each other. So that one of them for example, may migrate to another area leaving their initial formation, completing a specific task and return to the starting point, in addition to distancing the whole formation of robots from a central computer. By distancing a robot from the limits imposed by other means will entail a loss of signal compromising communication between them. This way, knowing a wireless network coverage area where robots will be connected is an important process for the implementation of a system of communication between robots. The aim of this work is to demonstrate the feasibility of expanding the coverage area of a wireless network, to be used in multi-robot systems, of a formation of robots that use a Real Time Database (RTDB) in multilayer networks that using the IEEE 802.11 and 802.15.4, analyzing the behavior of this expansion in relation to connectivity of robots, coverage area and interference in wireless networks. According to the performed experiments, the results demonstrate the robustness and reliability of the proposed scenario and it was shown that the use of multilayer networks provides a greater range in the coverage area of a robot training.

Keywords: Multilayer Networks, Ad Hoc Networks, ZigBee, Real Time, Middleware, Multi-Robots.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os robôs estão cada vez mais inseridos em diferentes áreas da atividade humana. Pode-se citar, como exemplo, projetos espaciais, robótica, controle de processos, mercado financeiro, telecomunicações e sistemas de controle de tráfego aéreo [5]. Na robótica, a utilização de robôs móveis autônomos trabalhando de forma cooperativa tornou-se um fato essencialmente importante [6] e, naturalmente, pesquisadores estão empenhados em investigar a melhor forma desses robôs realizarem a cooperação [7]. Trabalhando em conjunto e de forma cooperativa esses robôs conseguem finalizar uma tarefa de forma mais rápida e eficiente do que trabalhando sozinhos. A utilização de múltiplos robôs com capacidade de cooperação pode ser vantajosa ou mesmo obrigatória em várias tarefas tais como o transporte de objetos volumosos, cobertura de áreas abrangentes (e.g., para limpeza), vigilância de grandes espaços (e.g., para detecção de incêndios), seguimento de plumas de poluição ou de objetos, etc. A Figura 1.1 mostra um exemplo da utilização de equipes de robôs para a exploração de ambientes desconhecidos [6] [8].

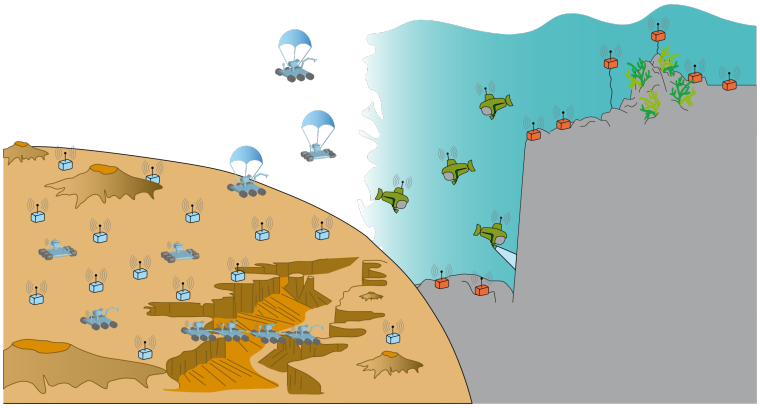


Figura 1.1: Exemplos de sistemas multi-robôs em monitoramento de ambientes desconhecidos.

O uso desses sistemas, denominados de multi-robôs, é ainda justificado pelas notórias vantagens que eles oferecem em relação aos robôs individuais, como: uma maior **escalabilidade**, que age como facilitadora na solução de problemas maiores nos quais muitas vezes podem ser resolvidos adicionando mais robôs à equipe; a **redundância**, em que os sistemas multi-robôs podem funcionar mesmo quando um membro da equipe está danificado ou com defeito; e proporcionando uma maior **força de trabalho**, em que a divisão de tarefas entre vários elementos simplifica as ações e diminuem o tempo de execução delas [7], isso devido ao fato de que, trabalhando de forma conjunta, vários robôs podem compartilhar informações entre si, auxiliar um ao outro e trocar responsabilidades [8].

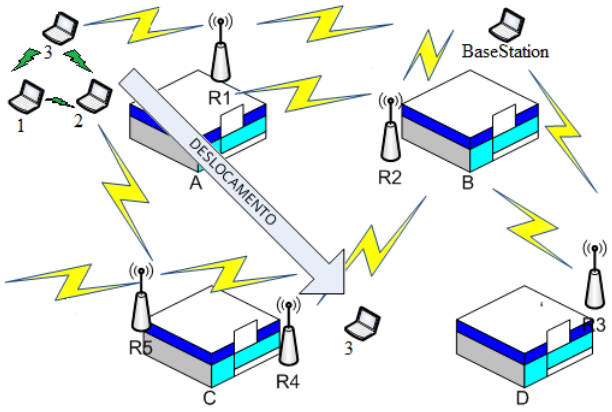


Figura 1.2: Deslocamento de estação: Nesse exemplo evidencia-se o deslocamento de um robô (indicado pelo número 3) saindo de sua formação.

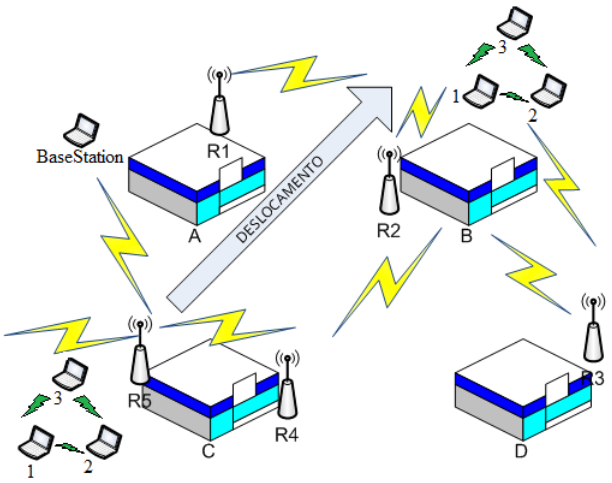


Figura 1.3: Deslocamento da formação: Nessa figura enfatiza-se o deslocamento do grupo de robôs para uma outra posição.

Contribuindo um com o outro, no intuito de ultrapassar as barreiras, alcançar

objetivos e manter a cooperação entre si, de forma que possam se deslocar em formação em um ambiente qualquer, esses robôs conseguem finalizar uma tarefa de forma mais eficaz do que trabalhando sozinhos. Ao distanciar um robô do outro, os limites conferidos pelo meio acarretarão uma perda de sinal, comprometendo a comunicação entre eles (Figuras 1.2 e 1.3). Gradativamente os sistemas autônomos necessitam de comunicações mais robustas e, sendo assim, observa-se que garantir a qualidade da transferência dos dados entre esses robôs é importante.

Neste sentido, a base de dados em tempo real (RTDB - *Real-Time Data Base*) [9] é uma das soluções existentes para esse propósito, uma vez que sua estrutura é formada por um banco de dados não persistente, o que possibilita mais robustez no tráfego dos dados. No entanto, garantir e manter essa qualidade pode se tornar uma tarefa bem desafiadora, uma vez que vários elementos podem interferir em uma transmissão, como o distanciamento entre os robôs, a concorrência nas mensagens, a perda de pacotes, a sincronização das mensagens e os atrasos derivados de fatores externos ou internos à rede com RTDB [10] [11].

Por conseguinte, conhecer a área de cobertura e a qualidade de uma rede sem fio em que os robôs estarão conectados é um processo importante para a implantação de um sistema de comunicação entre robôs. Conforme estudos realizados em [9], viu-se que as diferentes funcionalidades, e o dinamismo dos robôs, permitem uma variedade de maneiras de se configurar a rede de computadores para atendê-los, podendo ela ser mais robusta, mais econômica, centralizadora ou distribuída. O conhecimento e análise dessas redes é facilitada com o auxílio do *Received Signal Strength Indication* (RSSI), um estimador de potência que pode ser utilizado para identificar fontes de interferência e intensidade de sinais.

1.1 Objetivos

Considerando os desafios citados, o objetivo geral deste trabalho é *propor a expansão da área de cobertura de uma rede sem fio, a ser utilizada em sistemas multi-robôs, usando um Middleware de Base de Dados em Tempo Real - RTDB, em redes multicamadas.*

Em continuidade, os seguintes pontos foram considerados com o intuito de efetivar o objetivo geral descrito acima, e são compreendidos como os objetivos específicos da pesquisa:

- Garantir mobilidade, em espaços de maior distância, para sistemas multi-robôs;
- Expandir a área de atuação de equipes de robôs através da expansão da cobertura de redes sem fio;
- Utilizar equipes de robôs para monitoramento de ambientes;
- Utilizar equipes de robôs para realizar análise da área de cobertura e de varreduras de redes sem Fio.

Esta pesquisa, realizada com a ajuda de financiamento apoiado pela CHAMADA UNIVERSAL MCTI/CNPq N° 14/2014, é parte integrante de um projeto maior, que visa criar um sistema de controle de formação aplicado a diversas equipes de robôs móveis (sistemas multi-robôs ou SMR) terrestres e aéreos destinados a monitorar o perímetro aberto (na área de estacionamento, acessos e mata) e fechado (dentro de edificações) do Campus V da Universidade Federal da Paraíba, evitando possíveis acessos de intrusos não pertencentes à comunidade acadêmica, podendo ser utilizados também para a recepção de alunos e visitantes.

1.2 Organização da Dissertação

Quanto a sua estrutura, esta dissertação está organizada em cinco capítulos que discutem inicialmente a introdução, motivação e objetivo geral, bem como os objetivos específicos deste trabalho. No Capítulo 2 é apresentado o Referencial Teórico no qual são expostos os Padrões de Rede e Comunicação utilizados além de alguns trabalhos relacionados que, direta ou indiretamente, possuem relação com esta pesquisa, além de disponibilizar uma fundamentação teórica com o intuito de familiarizar o leitor com alguns tópicos que são discutidos durante este trabalho. No Capítulo 3, é descrita a metodologia aplicada nas atividades planejadas inerentes à realização de cada tarefa primordial para alcançar objetivo geral do trabalho. Baseado em toda a fundamentação teórica e na metodologia explanada anteriormente, o Capítulo 4 expõe os resultados atingidos divididos em dois experimentos, nos quais são apresentadas as dificuldades e soluções encontradas. Finalmente, no Capítulo 5, é apresentado um resumo do trabalho, expondo as conclusões finais e sugestões para estudos futuros.

Além destes Capítulos, alguns materiais foram anexados a esta dissertação. No Apêndice A, é apresentado um Tutorial de como instalar a RTDB em sistema operacional Linux. Já no Apêndice B, encontra-se o Tutorial que explica como implementar e utilizar do ZigBee com a RTDB. Nos Apêndice C e D, é mostrado os scripts de automatização dos tutoriais já citados. Apêndice E mostra a API de integração da RTDB com software de controle dos robôs. O Apêndice F mostra as especifições técnicas do roteador sem fio usado. No Apêndice G é mostrado um Quadro Comparativo entre ZigBee, *Bluetooth* e Wi-Fi.

Capítulo 2

Padrões de Rede e Comunicação

Neste capítulo discorre-se uma pesquisa sobre os principais temas que, de certa forma, estão relacionadas ao objetivo desta pesquisa. Inicialmente, é apresentando referências sobre a tecnologia de rede Mesh destacando-a como uma alternativa para estender as redes locais sem fio e como se pode utilizá-la em comunicação de sistemas multi-robôs (Seção 2.1). Posteriormente, o Zigbee é exporto como outra opção de comunicação sem fio ressaltando o seu uso em aplicações que não necessitem de enviar altas taxas de dados, mas que necessitam de baixa latência e baixo consumo energético (Seção 2.2). Por fim, é discutido um tipo de comunicação em tempo real entre robôs móveis utilizando um *Middleware* de base de dados em tempo real (RTDB - *Real-Time Data Base*) (Seção 2.3).

O capítulo apresenta os padrões de rede utilizados nas redes que fazem parte da topologia proposta. Para que dois times de robôs possam patrulhar ambientes diferentes em um mesmo prédio (i.e. cada time num andar do prédio), partilhando dados entre robôs de um mesmo time porém sem a necessidade de partilhar dados entre os times, foi utilizado, na camada de rede proposta para a comunicação apenas entre robôs do mesmo time, o padrão IEEE 802.15.4. Para essa camada de rede entre robôs, foram utilizados módulos Xbee de comunicação.

Além da comunicação entre robôs do mesmo time, ambos os times deverão também partilhar apenas dados essenciais de posição e bateria de cada robô com um sistema supervisor central, denominado de *Basestation*, que monitora o comportamento de todos os robôs envolvidos, diferenciando os robôs por time e local onde se encontram. Para essa segunda camada de rede foram utilizados roteadores Mesh sem fio com padrão IEEE 802.11 de comunicação.

Para garantir a robustez, sincronismo e consistência nos dados transmitidos em ambas as camadas, fora implantada uma única RTDB que aloca dados provenientes das

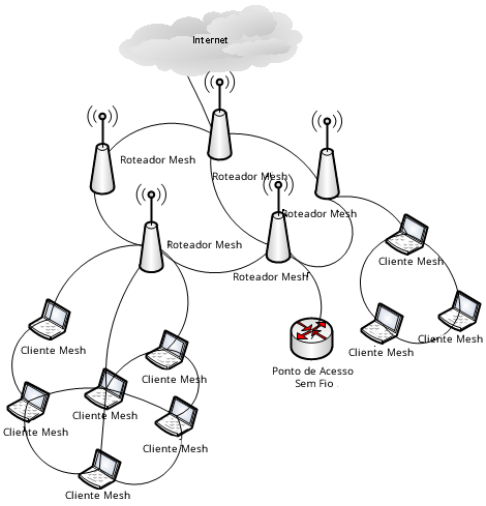
duas camadas de rede, utilizando para isso três estruturas diferentes onde uma serviria apenas para a camada entre os robôs, a segunda estrutura seria compartilhada na camada superior da rede entre os robôs e o supervisor e a terceira seria comum para ambas as camadas de rede. As próximas subseções irão destrinchar a relevância de cada camada de rede e da RTDB para a formação da topologia de rede proposta.

2.1 Redes Mesh

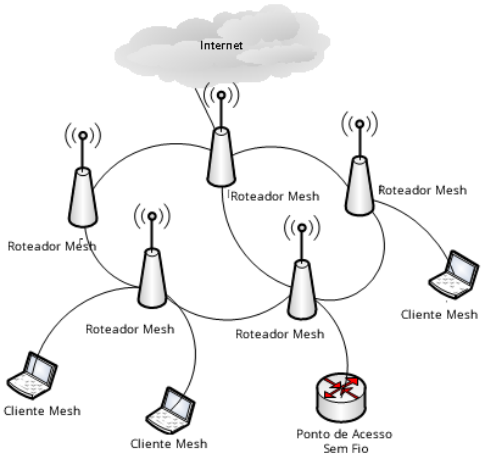
As redes sem fio Mesh (WMNs) são tecnologias que estão emergindo [12] e estão se tornando uma alternativa para estender as redes locais sem fio (WLANs) usadas atualmente. As redes locais sem fio (WLANs) baseiam-se na presença de uma infraestrutura de conectividade de acesso com fio possibilitando a conexão de terminais sem fio. Por outro lado uma rede Mesh independe da presença de infraestrutura com fio, ou seja, não há necessidade da existência de cabeamento, pois os nós sem fio se comunicam um com os outros sem a necessidade dessa infraestrutura, criando automaticamente uma comunicação ponto a ponto e gerando a conectividade em malha[13][12]. Dessa forma, as redes Mesh são utilizadas em diversas aplicações, tais como: aplicações empresariais, automação predial, ampliação de cobertura de prestação de serviço, redes sem fio comunitárias dentre outras [14].

A eficiência das WMNs depende de alguns aspectos de projetos e configurações, tais como: número de saltos, número do canal, local das antenas, requisitos de *throughput* e de latência, dentre outros. Além de várias empresas terem percebidos o potencial desse novo paradigma, e já disponibilizarem produtos para redes Mesh, para que essas redes possam desfrutar de todo potencial proposto, são necessários alocar esforços na investigação e aprimoramento em alguns pontos, a exemplo dos protocolos de controle de acesso ao meio (MAC - *Medium Access Control*) e dos roteamentos não escaláveis, que interferem significantemente no rendimento deste tipo de rede sempre que houver o aumento do número de nós ou de saltos [14][15]. Quanto a classificação, em [15] e [12] é possível organizar as arquiteturas das redes Mesh em três categorias, como exposto na Figura 2.1: *Infrastructure/Backbone WMN, Client WMN e Hybrid WMN.*

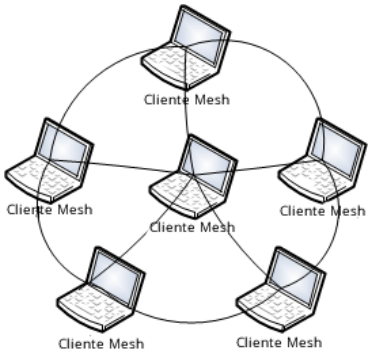
a) ***Infrastructure/Backbone WMN*:** nessa arquitetura os roteadores são conectados entre si (com ou sem fio), formando uma infraestrutura para os clientes. Essa infraestrutura ode ser construída utilizando diversos tipos de tecnologias de rádio, porém



(a) *Hybrid WMN.*



(b) *Infrastructure WMN.*



(c) *Client WMN.*

Figura 2.1: Classificação das arquiteturas de Redes Mesh.

a mais utilizada é o IEEE 802.11. É possível que clientes convencionais conectem a esse (*backbone*) e se integre a WMNs já existentes, através da funcionalidade de *gateway/bridge*. A arquitetura pode ser visualizada na Figura 2.1b;

b) *Client WMN*: arquitetura baseada na comunicação ponto-a-ponto entre os clientes. Os clientes formam uma rede real realizando funcionalidades de configurações e roteamento, sendo dispensado, nessa arquitetura, o uso do roteador. Em comparação a arquitetura "*Infraestrutura/Backnone WMNs*", os requisitos dos nós finais são maiores, pois os mesmos desenvolvem funcionalidade adicionais, tais como a de roteamento e auto-configuração. A arquitetura pode ser visualizada na Figura 2.1c;

c) *Hybrid WMN*: a arquitetura híbrida, conforme Figura 2.1a, é uma combinação das arquiteturas de Infraestrutura e a Cliente WMNs. Os clientes Mesh podem acessar a rede através de roteadores Mesh, bem como é possível a conexão diretamente com outros clientes da malha. Nessa arquitetura o *backbone* fornece conectividade com

outras redes, a exemplo de Wi-fi, WiMAX, Celular e Rede de Sensores Sem Fio. A capacidade de roteamento dos clientes Mesh proporciona melhor conectividade e cobertura.

Sistemas multi-robôs móveis são úteis em muitas aplicações críticas, tais como busca e salvamento, vigilância ambiental e ambientes diversificados. A comunicação eficiente entre os robôs em tais sistemas é útil para a coordenação das equipes integrantes, bem como a troca de dados. Uma vez que muitas aplicações para robôs móveis envolvem cenários em que infraestrutura de comunicação podem sofrer interferência ou estar indisponíveis, as equipes de robôs móveis precisam, com frequência, se comunicar uns com os outros através de redes Ad Hoc. Em tais situações, os protocolos de roteamento, para a entrega de mensagens entre os robôs, que possuem um baixo processamento e boa eficiência energética tornam-se um requisito fundamental.

Dentre os trabalhos relacionados à rede Mesh destaca-se o artigo escrito por [16]. Primeiramente os autores propõem e avaliam dois protocolos de roteamento *unicast* adaptados para o uso em redes Ad Hoc formadas por equipes móveis de multi-robôs: *Mobile Robot Distance Vector* (MRDV) e *Mobile Robot Source Routing* (MRSR). Ambos os protocolos buscam explorar as características de mobilidade exclusivamente para as redes de robôs móveis a fim de realizar o roteamento eficiente. Posteriormente, ainda em [16], os autores também propõem e avaliam um eficiente protocolo de roteamento *multicast*, o MRMM (*Mobile Robot Mesh Multicast*) que explora o fato de que os robôs móveis sabem a qual velocidade eles são instruídos se moverem e qual a distância no prédio que a Rede Mesh mantem-se mais eficiente para a comunicação do grupo. Os resultados mostraram que o MRMM forneceu um mecanismo de comunicação em grupo eficiente que poderia ser usado em muitos cenários de aplicação de robôs móveis.

Outro trabalho utilizando redes Mesh foi realizado por [17]. Em seu artigo, os autores consideraram o problema da implantação de uma rede de sensores móveis em um ambiente desconhecido. Uma rede de sensores móvel foi composta de conjunto de nós distribuídos, cada um dos quais tem capacidades de detecção, computação, comunicação e locomoção. Os autores apresentam uma abordagem baseada no potencial de campo para implantação de nós da rede. Os campos foram construídos de tal modo que cada nó foi repelida por ambos os obstáculos e por outros nós, forçando assim a rede para se espalhar por todo o ambiente.

Outro trabalho apresentado por [18] discutiu a prova de conceito que proporcionou a unidades robóticas móveis a capacidade de fornecer uma rede Mesh sem fio móvel,

oferecendo serviços sem fio aos clientes finais e também demonstrou a capacidade de aumentar o rendimento deste sistema Mesh sem fio móvel reduzindo, de forma autônoma, a contagem de saltos necessário para o tráfego na rede. Com isso, o estudo contribuiu para o futuro desenvolvimento de um sistema robusto que pode ser implantado e utilizado em diferentes situações e indústria.

Porém, conforme experimentos pelos autores em [19], é importante mencionar que, em sistemas multi-robôs, o uso puro de uma rede Mesh sem fio não é vantajoso devido ao fato de que nem todos os dados trocados entre robôs tenham que ser necessariamente vistos pelo computador central (*Basestation*) ou por outras equipes de robôs. No controle de formação, por exemplo, é comum compartilhar matriz de covariância de observação de um alvo determinado, dados estes que são inúteis para a *Basestation* [8]. Neste caso, apenas a posição de referência final é necessária para ser transmitida à *Basestation*, permitindo que na rede Mesh trafegue apenas dados úteis, necessários para manter os robôs em formação.

2.2 Redes ZigBee

Na atualidade, há uma uma vasta diversidade de padrões recomendados para estabelecer comunicação em redes sem fio. Dentre os mais conhecidos tem-se o Wi-fi ([20]), Bluetooth ([21]) e Zigbee ([22]), comparados na Tabela 2.1.

As redes ZigBee, mantidas pela *ZigBee Alliance*, estão inseridas no Padrão IEEE 802.15.4, que visa aplicações sem fio para equipamentos que não precisem de alta taxa de dados, mas que necessitam de baixa latência e baixo consumo de energia, o que permite uma maior durabilidade da mesma. A WPAN (*Wireless Personal Area Network*), em que está situada a tecnologia de pequeno alcance (entre 1 e 100 metros), consiste em um padrão para redes locais e suas taxas de transmissão de dados podem variar [23]. As redes ZigBee são um exemplo de WPAN e oferecem uma excelente imunidade contra interferências, e a capacidade de hospedar milhares de dispositivos em uma rede (teoricamente 65.536), com taxas de transferências de dados a 250Kbps [3].

Em uma rede IEEE 802.15.4 há três tipos diferentes de dispositivos: *Network Coordinator*, *Full Function Device* (FFD) e *Reduced Function Device* (RFD). O *Network Coordinato*r contém todo o conhecimento da rede. É o mais sofisticado e utiliza mais memória e processamento. Tem como função coordenar a rede, decide a forma de acesso

Padrões	Zigbee (IEEE 802.15.4)	Wi-Fi (IEEE 802.11b)	Bluetooth (IEEE 802.15.1)
Alcance da transmissão (em metros)	1 – 100	1 – 100	1 – 10
Tempo de vida de baterias (dias)	1 – 10	0.5 – 5	1 – 7
Número de nós	>64000	32	7
Tamanho da pilha de protocolo (KB)	4 – 32	1000	250

Tabela 2.1: Comparação entre os padrões Zigbee, Wi-Fi e Bluetooth [4].

ao meio e permite que novos nós entrem na rede. Já o *Full Function Device* possui toda a funcionalidade do IEEE 802.15.4. Contém uma memória adicional, processamento ideal nas realizações de roteamento de rede e também são utilizados nas margens da rede fazendo uma conexão com o mundo real. O *Reduced Function Device* por sua vez possui funções limitadas e controla custos e complexidades do tráfego da rede. Por não fazerem roteamento, não são usados nas margens das redes [24].

Pelo fato de compartilharem a mesma faixa de frequência, os padrões de comunicação sem fio determinam uma gama de canais ao longo desta faixa, existindo assim uma sobreposição entre os canais definidos por esses padrões [2]. Vê-se na Figura 2.2 os canais estabelecidos pelos padrões IEEE 802.11 e IEEE 802.15.4 na banda de 2,4 GHz, nos quais, apenas três canais (1, 6 e 11) do padrão IEEE 802.11 são mostrados, mas o padrão define 14 canais (apenas 11 permitidos no Brasil) que são dispostos no espectro, de modo que existe sobreposição entre canais vizinhos.

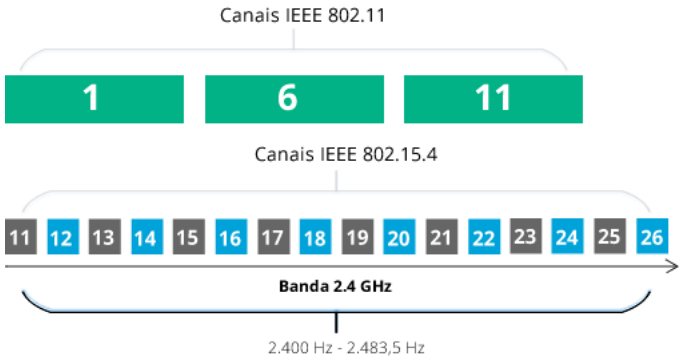


Figura 2.2: Canais definidos pelos padrões IEEE 802.15.4 e IEEE 802.11 na banda de 2,4 GHz.

Já na Figura G.1, no Apendice G, encontra-se um quadro comparativo dos principais padrões em termos das principais características de cada um. O ZigBee é o que tem menor taxa de transmissão de dados, porém tem maior alcance em relação ao Bluetooth, por exemplo. Por outro lado a Wi-Fi apresenta maior taxa de transferência de dados.

2.2.1 A *ZigBee Alliance*

A Rede ZigBee é formada por um consórcio de empresas, tais como Honeywell, Mitsubishi Electric, Philips, Invensis, Samsung e Motorola, ligadas ao ramo de tecnologia. Seus alvos principais são a automação de prédios residenciais, com o intuito de permitir o controle remoto de equipamentos e periféricos, dispositivos eletrônicos, brinquedos, também no monitoramento médico, na agricultura como forma de monitoramento da umidade do ar, do solo, temperatura e vento, no monitoramento automático do afundamento de prédios em tempo real [25]. A *ZigBee Alliance* define os protocolos das camadas de rede e de aplicação, incluindo também seus perfis. Já o padrão IEEE 802.15.4 define as camadas Física e de Acesso Múltiplo ao Meio (MAC), que são as camadas inferiores implementadas em *hardware*. Diferentes companhias e alianças trabalham para desenvolver especificações que abranjam as camadas de rede, perfis de segurança e aplicações de forma a oferecer interoperabilidade e o potencial comercial destas tecnologias [3].

O padrão ZigBee foi direcionado para aplicações que não necessitam de grande largura de banda, que sejam econômicas em relação aos valores de aquisição de *hardware*, quanto também ao consumo de energia, tornando-se bastante útil em aplicações críticas de temporização que necessitam, além das respostas rápidas, de um bom aproveitamento das baterias. Visto que, seu objetivo é ser utilizado em aplicações que não foram utilizadas pelo Bluetooth, ou por outro tipo de padrão sem fio, diferencia-se pelo tipo de alimentação dos dispositivos, que podem ser alimentadas com pilhas alcalinas comuns, criando uma expectativa de anos de duração [3] [24].

Objetivando especificar trabalhos relevantes à área pode-se mencionar [26], que propõe o protótipo de um robô jardineiro, semi-autônomo e móvel, que busca cumprir a exigência de pulverizar uma estufa. O robô utilizado foi equipado com um módulo ZigBee que por sua vez comunicava-se com outros sensores ZigBee distribuídos pela estufa. No estudo realizado, quando uma região específica da estufa necessita de assistência, uma solicitação de pulverização é enviada por mensagem para o robô, via ZigBee. O robô desloca-se até a área requerida e, ao finalizar a tarefa incumbida a ele, permanece no

ponto de operação original aguardando o comando para o próximo serviço.

Em complemento, tem-se o trabalho de [3] cujo o objetivo incide em implementar uma comunicação Ad Hoc sem fios entre membros de uma equipe de robôs móveis, empregando a tecnologia ZigBee, fazendo a integração e o desenvolvimento de funcionalidades do módulo Xbee, utilizando o Arduino como o *hardware* controlador. Esta pesquisa resultou em uma ferramenta útil de investigação, possibilitando a interação e cooperação de um grupo de robôs móveis, especificamente em robótica de enxame, patrulhamento, busca e salvamento, entre outros, com o uso do ZigBee.

2.3 *Real Time DataBase*

A crescente importância da computação em tempo real em inúmeras aplicações e as restrições, relacionadas ao processamento de transações em tempo hábil, que os modelos de base de dados considerados convencionais [27] apresentam - quando submetidos à sistemas de tempo real - motivaram os primeiros estudos sobre o conceito de *Real Time Database* [28]. Este tipo de implementação consiste em uma base de dados distribuída em tempo real, que possibilita que dados locais, não persistentes, sejam compartilhados e acessados entre sistemas móveis, simulando um acesso local, tornando-se diferente das implementações típicas, que se baseiam no modelo cliente-servidor [29] [30]. Por estas características, a RTDB é largamente utilizada em distantes domínios de conhecimento como sistemas aeroespaciais e de defesa, automação industrial, usinas de energia nuclear e robótica [28].

Em sua implementação mais comum, a RTDB é formada por duas partes: a camada superior, formada pela própria RTDB, que é replicada entre todos os participantes do sistema e inclui as interfaces para os dados que serão compartilhados com a camada de comunicação; e a camada inferior, composta por um protocolo de comunicação sem fio, que mantém as múltiplas réplicas da RTDB sincronizadas [31].

A RTDB criada pelo Projeto CAMBADA [32], da Universidade de Aveiro – Portugal, foi desenvolvida para auxiliar a troca de informações entre equipes que participam da competição de futebol de robôs, a *RoboCup Federation* [33]. Nesta base de dados, a camada superior é representada por um *Middleware*, de código aberto, formado por um bloco de memória compartilhada distribuída, denominada de *Blackboard*. O *Blackboard*

opera como a base de dados de cada integrante da RTDB, chamado de agente (*Agent*), compartilhando a informação desejada e possibilitando o acesso aos demais integrantes [30]. O conceito de *blackboard* possui semelhança ao entendimento apresentado em [34], que detém os dados de estado de cada agente, juntamente com as imagens do local dos dados relevantes do estado dos outros membros da equipe.

Esta RTDB foi desenvolvida para sistemas operacionais Linux e é completamente implementado em ANSI C [35], em que o *blackboard* de cada agente é divido em blocos. Um dos blocos é uma área privada reservada para informações locais, ou seja, não é compartilhado com os demais participantes da RTDB, enquanto os demais blocos estão em uma área compartilhada [36], conforme ilustrado na Figura 2.3.

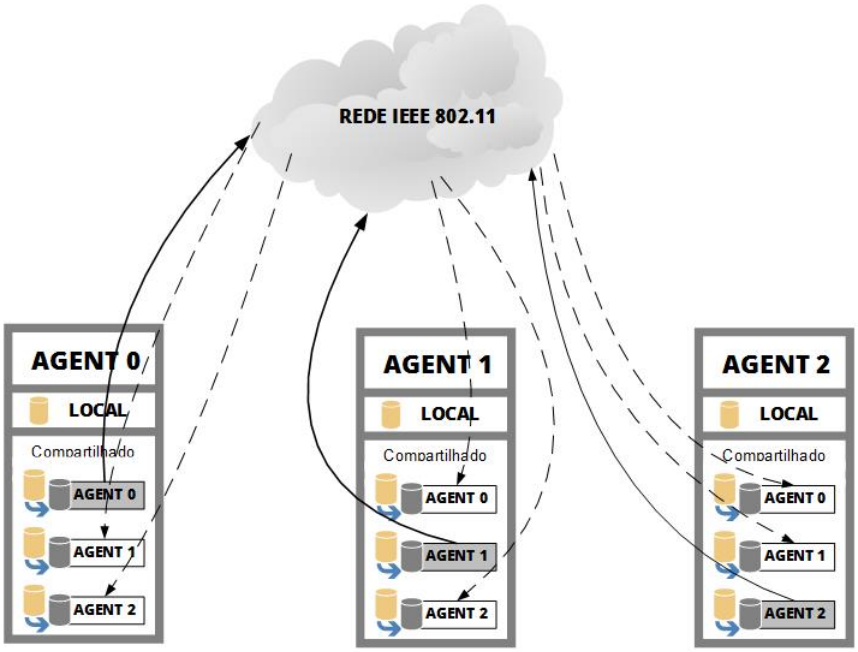


Figura 2.3: Cada agente transmite seu subconjunto de dados que está alocada na memória compartilhada.

A Figura 2.3 mostra que, originalmente, o protocolo de comunicação empregado entre os agentes é uma rede IEEE 802.11, protocolo padrão para redes locais sem fio [37], compartilhado em um único canal e transmitindo as suas variáveis com os demais integrantes da RTDB. Neste caso a largura de banda disponível é dividida entre todos os agentes. A fim de reduzir o número de possíveis colisões, utiliza-se o controle de acesso ao meio TDMA (*Time Division Multiple Access*) adaptativo [36] que funciona dividindo o canal de transmissão em intervalos de tempo distintos [11], o que possibilita, de maneira síncrona, que os agentes passem a ouvir um ao outro sem precisar de uma marcação de tempo centralizada [30]. A entrega das informações é feita utilizando a tecnologia de

redes *multicast*, que é a base de um serviço de rede no qual um único fluxo de dados, proveniente de uma determinada fonte, pode ser enviado simultaneamente para diversos receptores interessados [38].

A alocação do bloco de memória compartilhada é feita através de uma função denominada `DB_init()`, que precisa ser chamada sempre que processo necessita ingressar à RTDB. Em contrapartida, a memória usada pela RTDB é liberada utilizando-se da função `DB_free()`. Após ingressar na RTDB, os agentes precisam de funções específicas para acessar o conteúdo compartilhado. Para isso, utiliza-se funções locais chamadas `DB_put()` e `DB_get()` [39], que possibilitam a escrita e leitura, respectivamente, no Blackboard [6].

A composição dos arquivos quem compõe a RTDB desenvolvida pelo Projeto CAMBADA [32], disponível em sua página oficial [31], contém a seguinte estrutura:

- **Bin**: possui todos arquivos binários;
- **Comm**: possui arquivos relacionados ao sistema de comunicação;
- **Config**: possui arquivos de configuração da RTDB;
- **Include**: possui links simbólicos para todos os arquivos de dependência;
- **Lib**: possui as bibliotecas necessários para o funcionamento da RTDB;
- **Rtdb**: possui arquivos do blackboard.

A configuração da RTDB é composta, basicamente, por três seções: os agentes (*agents*), os itens (*items*), os esquemas (*schemas*) e as atribuições (*assignment*).

A seção de agentes é formada por uma lista de identificadores do agente, separados por vírgulas. A seção de itens é comporta por uma lista de itens. Por sua vez, um item é composto de um identificador, um tipo de dados (*datatype*), o *headerfile* no qual o tipo de dados é declarado (*typedef*). A seção de esquema é formada por um conjunto de itens que podem ser rotulados como compartilhado ou local e a seção de atribuição permite que um esquema possa ser associado a um ou mais agentes [30].

Um exemplo de um arquivo de configuração do RTDB pode ser encontrado na Código 2.1.

Código 2.1: Exemplo de um arquivo de configuração da RTDB.

```
1 # Configuration file for RTDB items.
2 AGENTS = Agent0, Agent1, Agent2;
3 # Item declaration section
```

```
ITEM VALUE {
   datatype = int ;
   headerfile = stdio.h
}
# SCHEMA definition section
SCHEMA A{
   shared = VALUE;
}
# ASSIGNMENT definition section
ASSIGNMENT {
   Schema = A;
   agents = Agent0, Agent1, Agent2;
}
```

Dentre os trabalhos relacionados à RTDB, tem-se como exemplo o trabalho feito por [10], o qual demonstra o início de um estudo para prever o pior caso no tempo de resposta de uma transação em redes RTDB. Segundo o autor, quando uma transação de alta prioridade requer acesso ao disco, irá causar a espera de qualquer transação de nível de prioridade inferior que está acessando o disco naquele momento, gerando um atraso nas transações de nível inferior. Apesar dos resultados não serem completos, a análise apresentada permite que os tempos de resposta de pior caso de transações possam ser calculados e previstos.

Em contrapartida, o artigo de [40] apresenta um protocolo de controle de concorrência chamado TCHP-2PL, que pode garantir a consistência temporal em redes RTDB. Segundo [40], um sistema de base de dados em tempo real (RTDB), não necessita apenas da garantia de transações acabadas nos prazos estabelecidos, mas também garantia de consistência temporal de objetos de dados acessados por transações, ou seja, quando várias transações são executadas simultaneamente e todas são envolvidas no acesso a dados compartilhados, a consistência dos dados pode ser afetada por causa da interferência mútua de transações simultâneas.

Por outro lado, no trabalho desenvolvido por [41] é descrito a ideia de desenvolver um sistema de interação multimodal real que é capaz de reconhecer e reagir a situações que são relevantes para a interação homem-robô. Para isso, foi desenvolvido um Framework, baseado em RTDB, responsável pelo processamento de dados multimodais. Esse Framework permite facilmente a integração de módulos de entrada e saída. Além disso, os resultados possibilitaram que os fluxos de dados assíncronos, gerados a partir de diferentes

fontes, pudessem ser interpretados de uma maneira síncrona.

Já no trabalho realizado por [42] foi proposto uma alteração no protocolo de sincronização da RTDB, permitindo assim o seu funcionamento em redes em Ad Hoc. Essa mudança permitiu também que o fluxo de dados trafegasse, de uma forma eficaz, através da rede Ad Hoc, contribuindo para medição mais precisa da distância entre os agentes móveis e proporcionando uma melhora no alcance dos mesmos. Os resultados mostram que houve uma redução de aproximadamente 3,3 vezes na taxa de insucesso da entrega dos pacotes. Mesmo com os ganhos alcançados, a mobilidade dos agentes ainda continua limitadas ao alcance da rede Ad Hoc.

Capítulo 3

Metodologia

Para atingir os objetivos desta pesquisa, inicialmente criou-se um ambiente de testes no qual teve a finalidade de analisar o comportamento dos agentes, e da RTDB, quando conectados simultaneamente a duas redes de computadores diferentes. A execução desta etapa viabilizou o envio de informações completamente isoladas para duas redes distintas através da RTDB.

Em seguida, as implementações já realizadas no ambiente de testes foram migradas para um ambiente real, com o propósito de realizar, de fato, a implementação da RTDB nos próprios controladores dos robôs RoC (*Robot Controller*), buscando executar uma análise da área de cobertura e uma varredura de redes sem fio existentes.

Por fim, através dos resultados encontrados, o objetivo foi atingido sendo apresentada uma solução para a comunicação para sistemas multi-robôs utilizando o *Middleware* RTDB em Redes multicamadas. Esses resultados possibilitaram a troca de informação de maneira eficiente e escalável e, consequentemente, instituindo um maior alcance na área de cobertura de uma formação de robôs.

3.1 Descrição do Experimento 1

3.1.1 Configuração do Ambiente

Neste experimento criou-se um ambiente de testes no qual foram realizadas as implementações utilizando computadores pessoais simulando os robôs, com o intuito de se assemelhar a um ambiente com múltiplos robôs. Estes computadores possuem configura-

ções de acordo com a descrição da Tabela 3.1.

Processador	**Velocidade NIC**[1]	**Função**
Intel® Core™ i5 2.67GHz	150 Mb/s	*BaseStation*
Intel® Core™ i5 2.50GHz	54 Mb/s	Agente 1
Intel® Core™ i3 2.67GHz	54 Mb/s	Agente 2
Intel® Core™ i5 2.67GHz	11 Mb/s	Agente 3

Tabela 3.1: Descrição física dos computadores utilizados.

Com o intuito de analisar a implementação da RTDB em Ad hoc com rede Mesh esse experimento foi dividido em duas etapas: na primeira parte, o objetivo principal foi replicar as modificações propostas por [42] a fim avaliar a eficiência de uma RTDB implementada em uma rede Ad Hoc em um pequeno grupo de agentes. Analisando a perda e a qualidade dos pacotes a medida que os agentes se distanciavam um do outro e considerando o comportamento, com e sem carga de tráfego adicional, foi analisado as limitações do canal de transmissão e sua sensibilidade aos ruídos, buscando dimensionar o distanciamento máximo entre os agentes sem que que haja falha na comunicação. Esta etapa não será completamente descrita nesta dissertação, porém outros testes com este tipo de cenário podem ser encontrados nos estudos realizados por [42], entretanto utilizando-se de métricas diferentes.

No segundo momento, a RDTB avaliada anteriormente, foi inserida em um ambiente de redes Mesh com o intuito de garantir a sua mobilidade, em espaços de maior distância, assegurando a comunicação das equipes com a *Basestation*, além de atestar uma flexibilidade de troca de informações entre os agentes integrantes das equipes de robôs. A descrição deste ambiente será apresentada na Subseção 3.1.2.

3.1.2 Descrição do Ambiente

Nesta etapa de trabalho foi instalada uma segunda placa de rede nos agentes já utilizados, permitindo, assim, que eles ingressassem em uma segunda rede, paralela à Ad Hoc. Nesta segunda rede, empregou-se o conceito de redes Mesh, especificamente usando a arquitetura *Infrastructure/Backbone WMN*, o que possibilitou uma maior mobilidade dos agentes em toda a área de cobertura da rede, tornando possível a sua expansão com a

[1]NIC - *Network Interface Card* termo em inglês para interface de rede.

instalação de novos roteadores Mesh. Neste caso, foram utilizados dois roteadores, Roteador 1 e Roteador 2, que serviram de *Backbone* para a rede. Para a implementação da rede Mesh, os roteadores utilizados precisaram ter os seus sistemas operacionais atualizados com um projeto de código aberto, que utiliza um sistema Linux baseado em *firmware* para roteadores sem fio e pontos de acesso sem fio, chamado DD-WRT [43].

O objetivo principal desta etapa foi avaliar o comportamento dos agentes e da RTDB quando conectados, simultaneamente, a duas redes de computadores distintas, tornando possível o envio de informações diferentes para cada rede, através da RTDB. A diagramação do ambiente montado pode ser observada na Figura 3.1, em que estão descritos todos os elementos integrantes do ambiente.

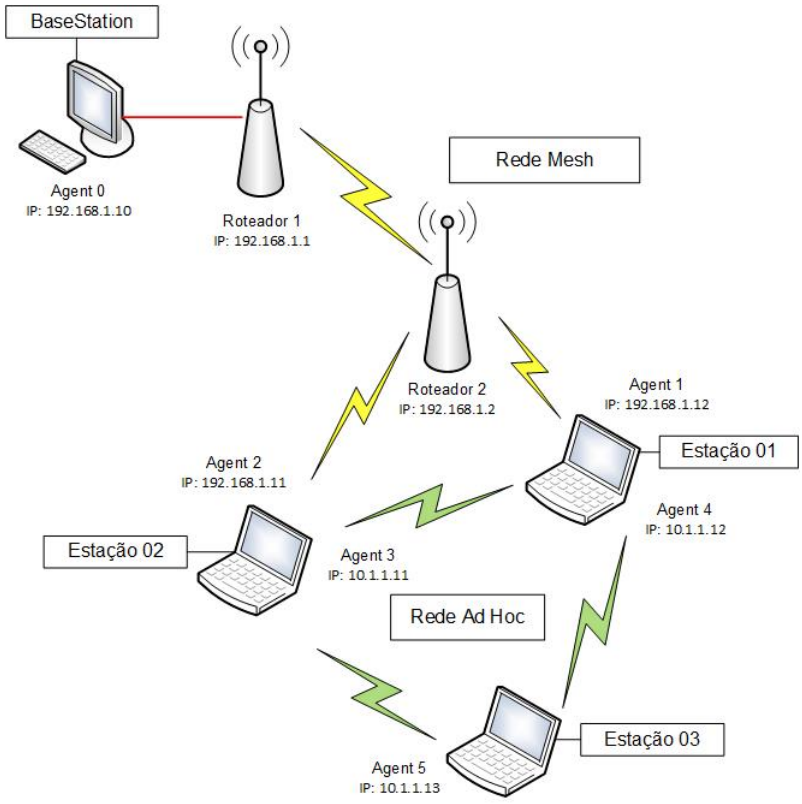


Figura 3.1: Diagramação do ambiente.

Em complemento à Figura 3.1, as Tabela 3.2 descreve detalhadamente o ambiente para este cenário, listando as informações das configurações bem como os seus respectivos endereços de rede.

Elemento da Rede	NIC-1	NIC-2	Agente NIC-1	Agent NIC-2
Basestation	192.168.1.10	X	AGENT=0	X
Roteador 1	192.168.1.1	X	X	X
Roteador 2	192.168.1.2	X	X	X
Estação 01	192.168.1.12	10.10.1.12	AGENT=1	AGENT=4
Estação 02	192.168.1.11	10.10.1.11	AGENT=2	AGENT=3
Estação 03	X	10.10.1.13	X	AGENT=5

Tabela 3.2: Detalhamento dos agentes.

3.2 Descrição do Experimento 2

Nesta etapa, os testes descritos na Seção 3.1, que buscam avaliar o comportamento da RTDB, agora foram ajustados para executarem em conjunto aos controladores dos robôs RoC (*Robot Controller*), objetivando realizar uma análise da área de cobertura e uma varredura de redes sem fio existentes.

3.2.1 Configuração do Ambiente

Os experimentos foram realizados utilizando um computador pessoal central (*BaseStation*) e três robôs (agentes), distribuídos no segundo e terceiro andar do prédio do Centro de Informática da Universidade Federal da Paraíba - CI/UFPB (Ver Figura 3.2).

Figura 3.2: Prédio do Centro de Informática da UFPB.

Tais robôs foram alocados nos corredores dos diferente andares, tais como visto na

Figura 3.3, em que as três cores (amarelo, verde e azul) delimitam as áreas de atuação dos robôs #17, #11 e #12, respectivamente. Essa padronização de cores foi utilizada em todos os gráficos que apresentam os dados dos robôs. Nesses experimentos, os robôs utilizados foram do tipo Turtlebot, Versão 2.

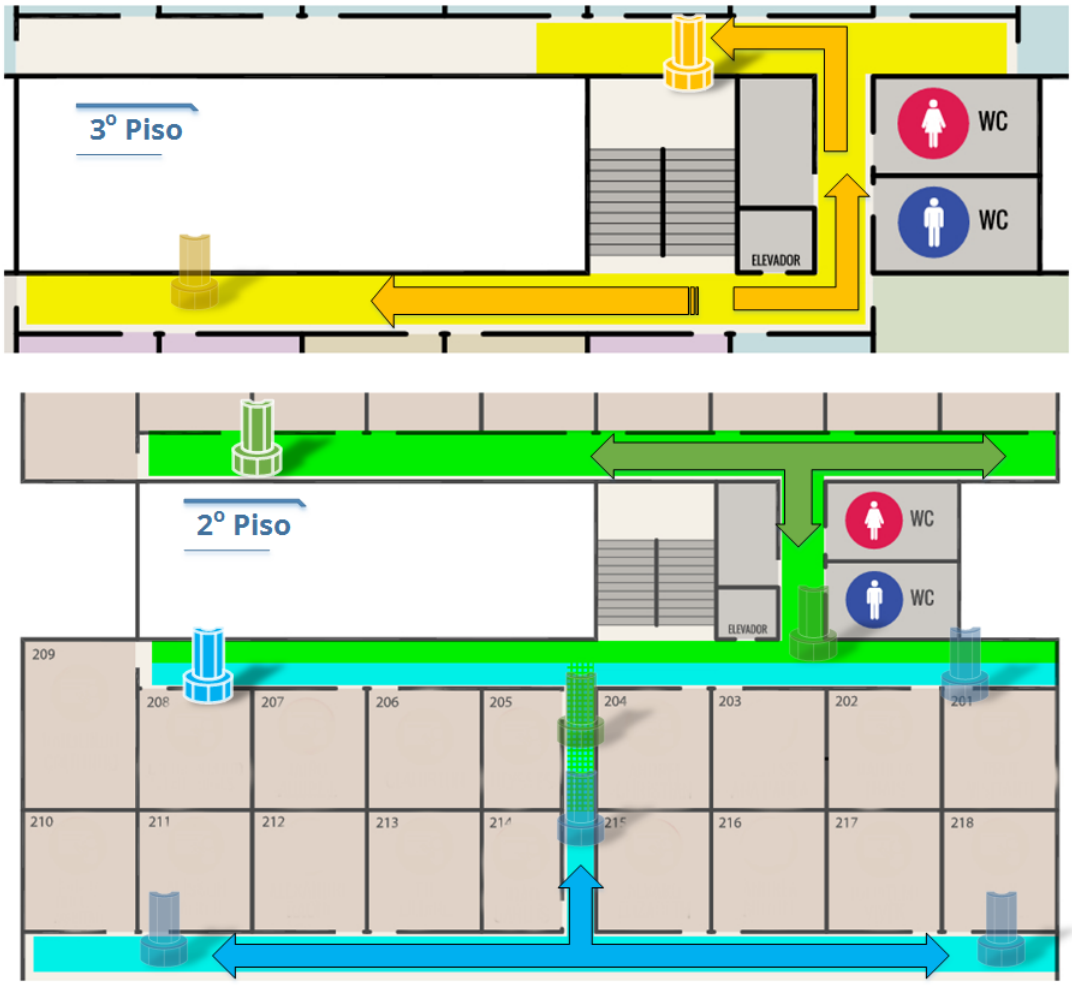


Figura 3.3: Área de análise.

O Turtlebot 2 (base Kobuki[1]) é uma nova versão da bem sucedida plataforma Turtlebot e tem muitas vantagens em relação a seu antecessor: precisão de medição, odometria com protocolo aberto, maior autonomia, maior carga, maior velocidade e maior mobilidade, rodas de maior diâmetro e capacidade para superar obstáculos até 12 mm.

Conforme Figura 3.4, o Turtlebot 2 é equipado com um Kinect possui dimensões de 354 x 354 x 420 mm, peso de 6.3 kg (com netbook), capacidade de carga de 5 kg, autonomia de ate 7 horas e atinge velocidade maxim a de 0,65 m/s.

[1]iClebo Kobuki - http://kobuki.yujinrobot.com/

Figura 3.4: Turtlebot 2 - LASER/UFPB.

Esses robôs foram configurados para atingir uma velocidade de 0,4 m/s e, conforme mostrado na Figura 3.5, foram divididos em dois times, no qual o Time 1, composto por um integrante, ficou localizado no terceiro pavimento, enquanto o segundo Time, formado por outros dois robôs, ficou situado do segundo pavimento. É pertinente mencionar que cada robô dispõe de um *software* denominado RoC (*Robot Controller*), encarregado de realizar o controle de movimento, enquanto o computador central utiliza um *software* chamado *Watchtower*, responsável por realizar o monitoramento de todos os robôs.

3.2.2 Descrição do Ambiente

A proposta de ambiente de comunicação para esse experimento é constituída pela configuração de uma rede composta por duas camadas distribuídas de acordo com a Figura 3.5: uma superior, em que é atribuída a comunicação entre os andares do prédio e outra inferior, encarregada de realizar a comunicação entre os agentes de um mesmo time.

A camada superior foi dada através de uma rede Mesh do tipo *Infrastructure WMN*. Para a implementação desta rede foram utilizados três *Access Point TP-Link TL-WA901ND* (Tabela 3.3) interligados entre si. Este roteadores foram encarregados de propagar uma rede sem fio denominada "ROBOT", configurada para operar o no Canal 9. Esta comunicação ficou responsável por interligar todos os robôs de todas as equipes até a *Basestation*, possibilitando o monitoramento de todas as ações dos robôs no ambiente, como pode ser visto na Figura 3.6.

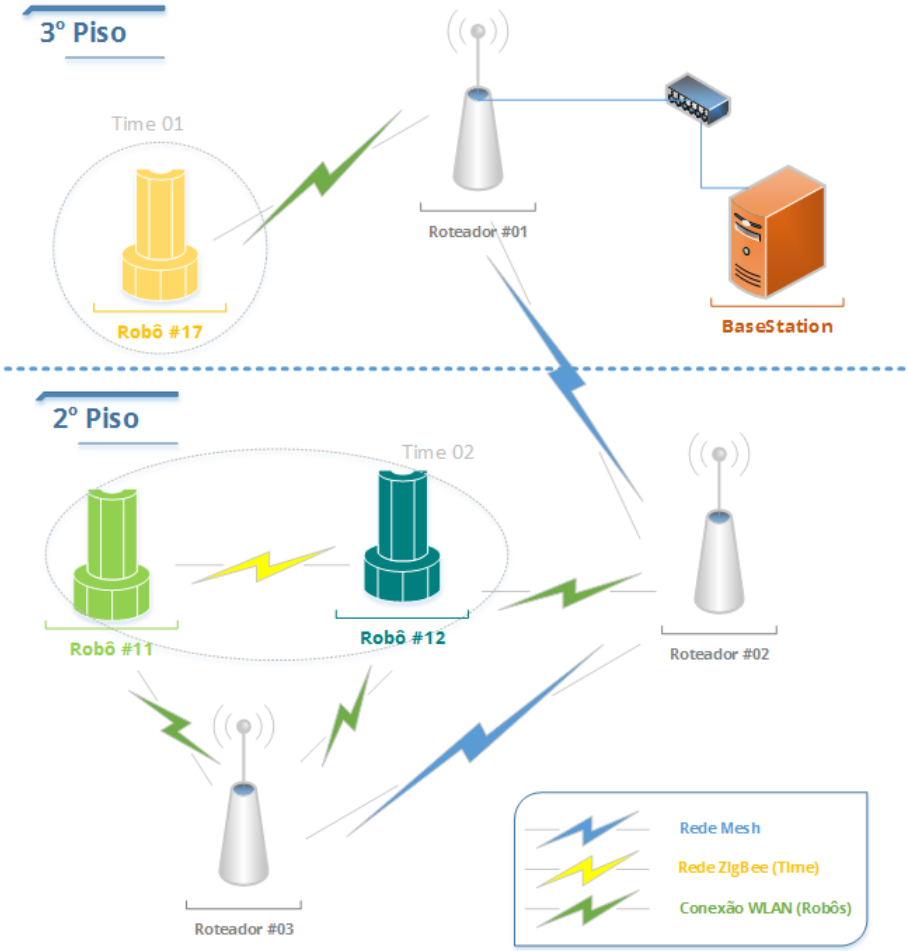


Figura 3.5: Detalhamento do ambiente configurado.

Modelo	TL-WA901ND 300Mbps Wireless N Access
Faixa de Frequência	2.4 ∼ 2.4835GHz
Taxa de dados	11n: até 300Mbps 11g: 54-48-36-24-18-12-9-6M 11b: 11-5.5-2-1M
Sensibilidade @PER	270M:-68dBm@10% PER 108M:-68dBm@10% PER 54M:-68dBm@10% PER 11M:-85dBm@8% PER 6M:-88dBm@10% PER 1M:-90dBm@8% PER

Tabela 3.3: Especificações: *TL-WA901ND Wireless N Access Point.*

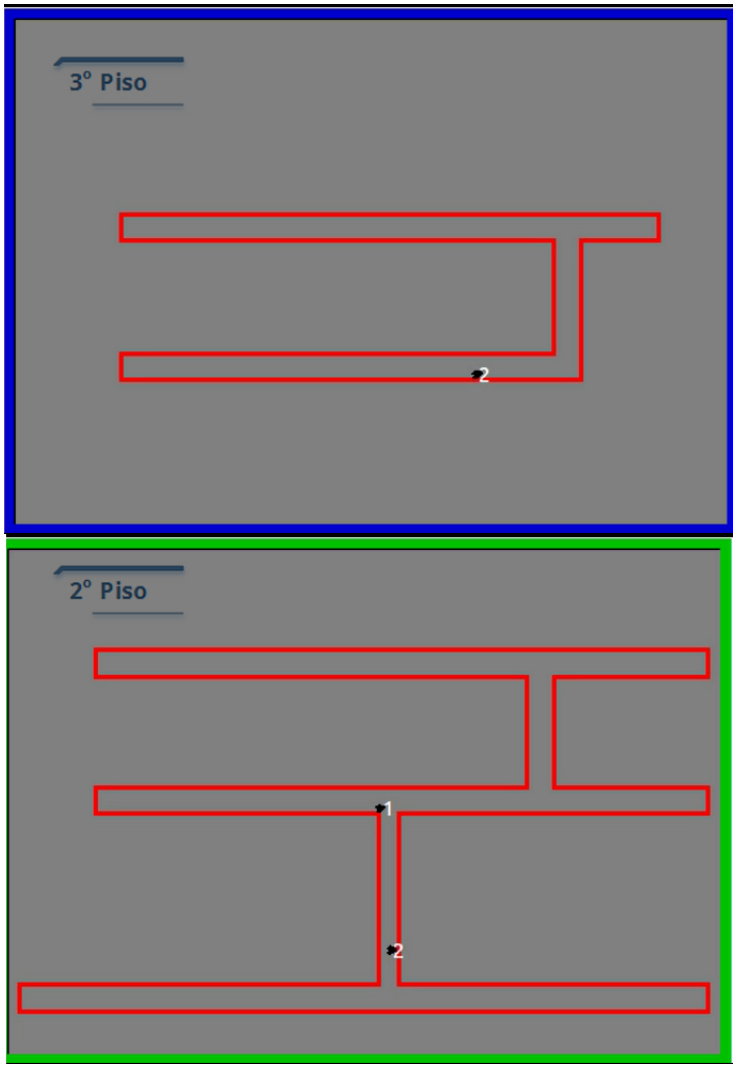


Figura 3.6: Captura de Tela do computador central (*Basestation*) executando o *software WatchTower*.

Já na comunicação entre os robôs de um mesmo time, a camada inferior foi dada por meio de uma rede Ad Hoc, utilizando o padrão ZigBee. Para esta camada foram utilizados módulos Digi XBee S1 (Figura 3.7). Estes módulos foram desenvolvidos com a finalidade de suprirem as necessidades como um baixo custo e baixo consumo de energia das redes de sensores sem fio [3].

A comunicação padrão entre os agentes da RTDB, como visto na Subseção 2.3, é feita pelo protocolo padrão para redes locais sem fio IEEE 802.11. Já a interligação dos Módulos Xbee ao sistema operacional é feita através de uma comunicação na porta serial do sistema [44] que, nativamente, não possui suporte para o envio de pacotes IP em suas transmissões. Como percebido, existem divergências de padronização de transmissão

entre os padrões Serial e IP, tornando-as tecnologias, em princípio, não compatíveis. Com o objetivo de sanar este problema utilizou-se o Protocolo SLIP (*Serial Line Internet Protocol*) que pode ser definido como um método de encapsulamento para transmissão de pacotes IP através de uma conexão serial. Este protocolo é definido pela RFC 1055 recomendada pela IETF® (*The Internet Engineering Task Force*) [45].

Figura 3.7: XBee Serie 1.

Com o propósito de resguardar a comunicação do ZigBee, e visando evitar conflitos de interferência de canal com as redes IEEE 802.11 existentes no ambiente, o módulo XBee foi configurado para trabalhar no canal 26 (hex. 0x1A) e em frequência de 2,480GHz. Estas configurações não geram conflitos com nenhum canal do espectro 802.11, conforme pode ser verificado na Figura 2.2.

Constatado que módulo XBee S1 utilizado possui um `BUFFER` com tamanho limitado, apenas 202 *bytes* [46], com a finalidade de evitar um possível *overflow* no módulo, e como medida de segurança, a RTDB precisou ser modificada para que não enviasse pacotes com mais que 202 *bytes* de dados. A alteração foi realizada no código-fonte do arquivo `rtdb/comm/comm.c` e a constante `BUFFER_SIZE` foi defina para o valor de 202, conforme Figura 3.8:

```
#include "multicast.h"

#include "rtdb_comm.h"


#define BUFFER_SIZE 202

#define TTUP_US 100E3
#define COMM_DELAY_US 150
#define MIN_UPDATE_DELAY_
```

Figura 3.8: Buffer RTDB.

Utilizando o programa X-CTU [44], todos os módulos Xbee foram configurados para estarem conectados em uma rede com o mesmo identificador (ID). Neste caso o

ID=3332. Aos campos *DH Destination Address High* e *DH Destination Address Low* foram atribuídos os valores `0(zero)` e `0x000000000000FFFF`, respectivamente, conforme Figura 3.9. Estes campos representam os endereços inicial e final de comunicação da rede. Os valores inseridos possibilitam que todas as mensagens enviadas por um módulo Xbee, para a rede ID=3332, sejam tratadas como *broadcast* e recebidas por todos os outros integrantes da rede.

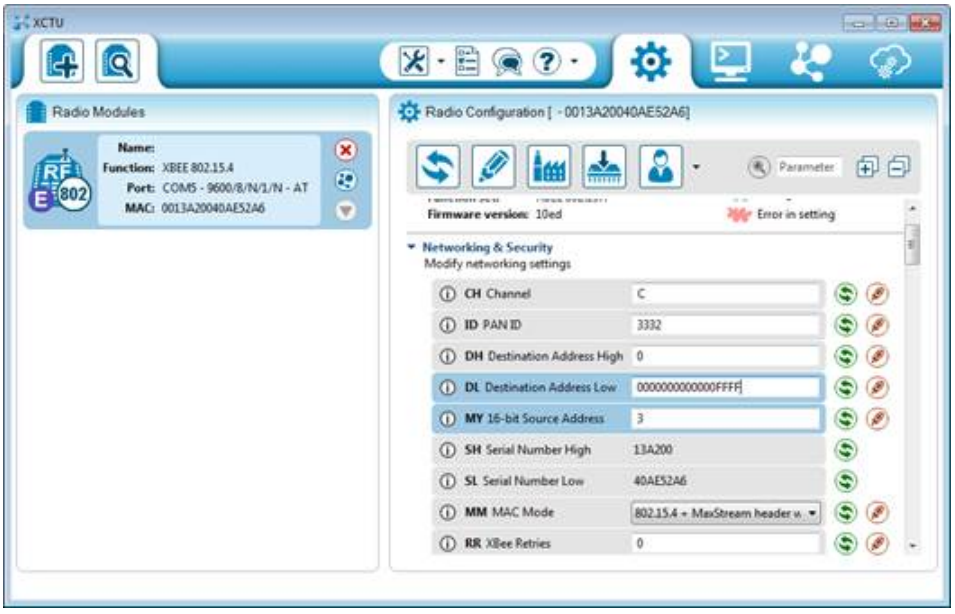


Figura 3.9: Configuração XCTU - ZigBee.

Em continuidade, os seguintes parâmetros de conexão foram utilizados:

- *Baud Rate* = 56800;
- *Data bits* = 8;
- *Flow control* = none.

3.2.3 Captura de Dados

Para realizar a captura de informações relevantes à análise área de cobertura foi criado um *Shell Script*, exposto no Código 3.1, baseado nos valores de RSSI, que busca informações da placa de rede conectada a uma rede sem fio. Fazendo o uso do comando

#iw, que é um utilitário de configuração para dispositivos sem fio, acompanhado de diversos parâmetros, o script inicialmente verifica se o dispositivo está conectado a uma rede sem fio, no caso a rede “ROBOT“. Caso positivo, inicia-se a captura de dados a cada um décimo de segundo que são armazenados em um arquivo de log, juntamente com a hora local do robô.

Código 3.1: Script de Captura do RSSI: rssi_capture.sh

```
#!/bin/bash
################################################################################
#
#   @file:       rssi.sh
#
#   @description: ShellScript
#                 RSSI Log Generator
#
#   @author:     Leandro Caetano
#           leandrocaetano.info
#           leandrojcaetano@gmail.com
#
#   @created:    18.05.2015
#
#   @updated:    19.05.2015
#
################################################################################
#

START=$(date +%s.%N)
DATA=‘date +%d-%m-%Y-%H.%M‘ #pega data atual
THISHOST=$(hostname -f)
FILE=[$THISHOST].log_rssi-$DATA.txt

clear
echo "#####################################"
echo   "######## PROGRAMA LER RSSI ########"
echo   "#####################################"
echo ""
```

```
30
31 conectado=‘iw dev wlan0 link | grep signal‘
32
33 # Se a montagem nÃ£o estiver OK, finaliza o processo e nÃ£o realiza BKP
34 if [ -z "$conectado" ]; then
35       echo "[ERROR] Rede sem fio NÃ£o conectada!!!"
36    echo "Conecte uma das redes disponÃveis:"
37       iwlist wlan0 scan | grep ESSID
38       exit 1
39     else
40    rede=‘iwconfig wlan0 | grep ESSID | awk ’{print $4}’‘
41    echo ""
42       echo "[OK] Conectado Ã  rede sem fio $rede"
43     fi
44
45 echo ""
46 echo "Pressione qualquer tecla para sair..."
47 echo ""
48 echo  "######## LOG RSSI ########" >> $FILE
49 echo "Nome do PC: $THISHOST" >> $FILE
50 echo "" >> $FILE
51 echo "Signal  |  Data e Hora" >> $FILE
52 echo "----------------------------------------" >> $FILE
53 counter=1
54 while :; do
55
56    signal="$(iw dev wlan0 link | grep signal | awk ’{print $2,$3}’)"
57
58    data="$(date +’%T.%N’)"
59
60    echo -en "\rGravando: $counter"
61
62    echo $signal " | " $data >> $FILE
63
64    ((counter++))
65
66 #  sleep 0.5
```

```
67
68    read -n 1 -t 0.1 parar
69
70    if test $? = 0; then
71
72       break;
73
74    fi
75 done
76
77 echo ""
78 echo "."
79 echo "."
80 echo "."
81
82 echo ""
83 END=$(date +%s.%N)
84 DIFF=$(echo "$END - $START" | bc)
85 echo "QTDE de Registros Gravados: $counter" >> $FILE
86 echo "Tempo de coleta: 0$DIFF segundo(s)" >> $FILE
87 echo ""
```

Em contrapartida, para obter informações relevantes às outras redes sem fio que operam no mesmo canal (9) ou as adjacentes à rede "ROBOT", foi utilizado o aplicativo LinSSID (ver Figura 3.10), um *scanner* de redes sem fio com interface gráfica, de código aberto e que pode ser facilmente instalado com o gerenciador de pacotes do Linux, no qual também armazena os dados que foram coletados em um arquivo de log. A detecção de informações com esta ferramenta é limitada aos ruídos provenientes de outras redes e, por consequência, outras fontes geradoras não serão detectadas e levadas em consideração.

3.2.4 Estruturas compartilhadas pela RTDB

Outra ação necessária, foi definição e alocação das informações a serem compartilhadas pela RTDB. Esses dados não poderiam, também, ultrapassar o valor de 202 *bytes*

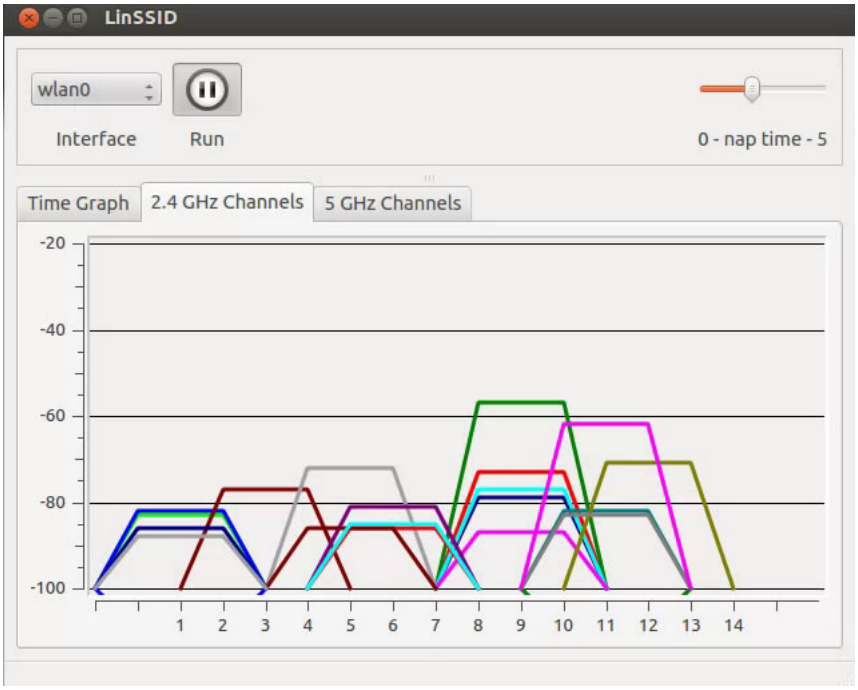


Figura 3.10: *Software* LinSSID.

suportados pelo XBee. Por isso foram divididos em estruturas que totalizando consumiram 96 *bytes* de memória, conforme mostrado no Código 3.2:

Código 3.2: Estruturas compartilhadas pela RTDB

```
1  // Robot Speed
2  struct tRobot
3  {
4     double vx;
5     double vy;
6     double w;
7  };
8
9  // Robot Role
10 struct tRole
11 {
12    float role[20];
13 };
14
15 // Status and battery level (base/robot)
16 struct tWatchtower
```

```
{
   double bat_rob;
   double bat_note;
   double active;
};

// Robot Localization
struct tLoc
{
   double x;
   double y;
   double teta;
};

typedef struct tRobot tRobot ;
typedef struct tRole tRole ;
typedef struct tWatchtower tWatchtower ;
typedef struct tLoc tLoc ;
```

Neste arquivo, cada estrutura possui as seguintes finalidades:

- **tRobot**: Informações referentes à velocidade do robô;
- **tRole**: Informação sobre o comportamento a ser seguido pelo robô;
- **tWatchtower**: Informações referentes ao nível de bateria e status do robô;
- **tLoc**: Informações referentes à localização do robô.

Já o arquivo de configuração, que determina quais estruturas serão compartilhadas com determinados agentes, de acordo com o modelo apresentado na seção 2.3, está exposto no Código 3.3, o qual define dois grupos de informações a serem partilhadas com os diferentes agentes integrantes de cada rede em particular. A fim de garantir uma melhor comunicação dos agentes, caso haja a interrupção de comunicação em uma determinada rede, algumas informações estão replicadas para as duas interfaces configuradas.

Código 3.3: Arquivo de configuração da RTDB: laser.conf

```
# Configuration file for RTDB items.
```

```
AGENTS = BaseStation, Robot1, Robot2, Robot3, Robot4, Robot5, Robot6, Robot7,
    Robot8, Robot9, Robot10, Robot11, Robot12, Robot13, Robot14, Robot15,
    Robot16, Robot17, Robot18, Robot19, Robot20;

# Item declaration section
ITEM ROBOT {
   datatype = tWatchtower;
   headerfile = robot.h;
}
ITEM ZIG {
   datatype = tRobot;
   headerfile = zig.h;
}
ITEM LOC {
   datatype = tLoc;
   headerfile = loc.h;
}
ITEM ROLE {
   datatype = tRole;
   headerfile = role.h;
}

# SCHEMA definition section
SCHEMA WLAN {
   shared = ROLE;
}
SCHEMA ZIGBEE {
   shared = LOC;
   shared = ROBOT;
   shared = ZIG;
}

# ASSIGNMENT definition section
ASSIGNMENT {
   schema = WLAN;
   agents = BaseStation;
}
```

```
37 ASSIGNMENT {
38    schema = ZIGBEE;
39    agents = Robot1, Robot2, Robot3, Robot4, Robot5, Robot6, Robot7, Robot8,
          Robot9, Robot10, Robot11, Robot12, Robot13, Robot14, Robot15, Robot16,
          Robot17, Robot18, Robot19, Robot20;
40 }
```

3.2.5 Métricas de Desempenho

Received Signal Strength Indication

A propagação de ondas eletromagnéticas ao longo de qualquer ambiente, exceto o vácuo, é sempre sitiado por perdas ocasionadas pelo absorvimento de potência pelas partículas do meio, o que acarreta na redução da potência da onda em relação à distância, ou seja a atenuação [3]. O *Received Signal Strength Indication* (RSSI), em português, Indicação da Intensidade do Sinal Recebido, é considerado um estimador de potência, baseado em *hardware*, que pode ser utilizado para identificar fontes de interferência e intensidade de sinais.

A medição de potência de um sinal recebido pode ser realizada pelos nós de uma rede sem a necessidade de transmissão/recepção de pacotes [47]. Esta medição é realizada em dBm (*decibel miliwatt*) para simbolizar a potência do sinal logarítmico, e os valores em decibéis podem ser relacionados ao nível de referência de um 1 mW (*miliwatt*) [1].

A Figura 3.11 mostra a relação matemática entre as medições dBm e seus valores correspondentes em mW. A Equação 3.1 é a utilizada para a conversão.

$$P_{dbm} = 10log_{10}(\frac{P}{1mW}), \tag{3.1}$$

em que:

- **Pdbm**: é a potência em dB;
- **P**: é a potência em watts.

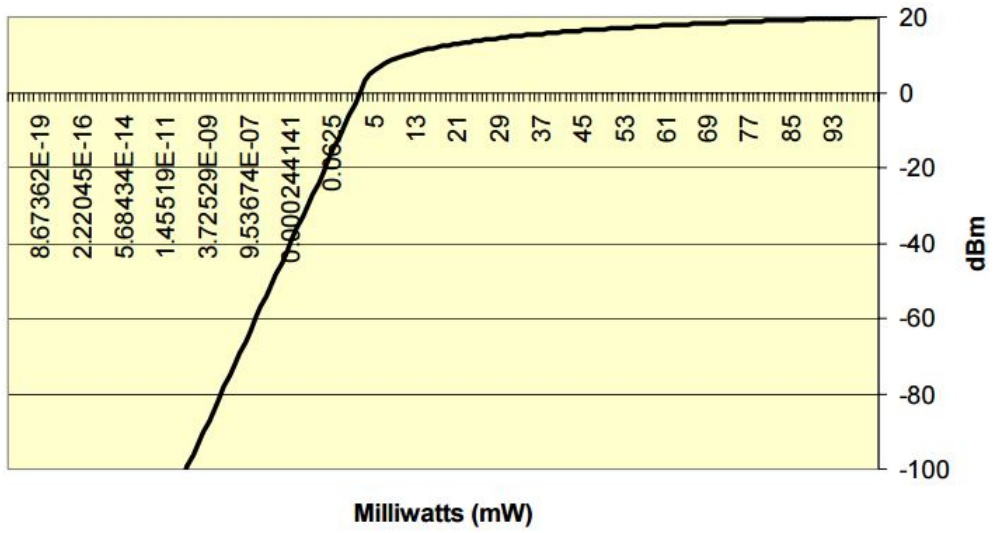


Figura 3.11: Relação entre dBm e *Miliwatts* (mW) [1].

O RSSI também pode ser útil em cálculos que buscam a estimativa de distância e localização [48] e que podem ser obtidas através da Equação 3.2:

$$RSSI_{dbm} = -10nlog10(d) + A_{dbm} \tag{3.2}$$

Logo, tem-se:

- **RSSI**: é o valor do RSSI recebido (dBm);
- **n**: é o expoente de perda do caminho;
- **d**: é a distância;
- **A**: é o valor RSSI a uma distância de referência.

Alguns estudos relacionados, como em [49], expõem pesquisas que visam o controle de robôs com o auxílio do RSSI. Para isso, o link de comunicação entre os robôs foi monitorado e, para garantir uma transmissão estável, foi usado um modelo de força virtual baseada em RSSI para controlar e conduzir os robôs. Os resultados das simulações mostraram que a abordagem baseada em RSSI é mais estável do que a abordagem anterior com base na distância.

No artigo [50], os autores demonstraram o uso do RSSI auxiliando a navegação de robôs móveis. No estudo é apresentado um esquema viável de navegação de robôs móveis

auxiliado por WSN, que inclui localização inicial do robô móvel, ajuste de orientação, planejamento de caminho de camada de rede, controle de movimento e correção baseada em RSSI. O resultado descrito mostra que o esquema proposto apresenta uma boa performance e possibilita, efetivamente, a navegação do robô equipado apenas com sensores simplesmente para a comunicação e desvio de obstáculos.

Já em [51], os autores descrevem um novo algoritmo de planejamento formação de uma rede robô móvel utilizando apenas o nível de sinal RSSI e o sensor direcional. O algoritmo de controle proposto é robusto contra o erro do sensor, mantendo a comunicação de rede constante entre pelo menos dois robôs móveis. A eficácia do algoritmo proposto foi confirmado tanto através de simulação e experimentos reais realizada utilizando vários robôs móveis.

Área de Cobertura

Em redes sem fio Mesh o tamanho da área de cobertura e o seu grau de sobreposição tem um efeito significativo no desempenho da rede. A fim de determinar o tamanho da área de cobertura para cada nó, é importante utilizar a informação sobre a localização de nós vizinhos. No entanto, essas informações nem sempre podem ser obtidas. Estudos atuais buscam um métodos para determinar o tamanho da área de cobertura única, utilizando a informação da distância estimada a partir de intensidade do sinal recebido [52].

Objetivando especificar trabalhos relevantes à área pode-se mencionar [53] que investigou as relações de conflito de escolha entre a área de cobertura e taxa de transferência de comunicação para redes sem fio Mesh. No entanto, a área de cobertura da rede inteira não foi discutida e do tamanho de sua rede investigado foi pequeno (apenas 3-4 nós). Em [54], foi proposto um método para a implementação de nós de modo para determinar o número mínimo de saltos requeridos para proporcionar a terminais de cliente um caminho para os nós de *gateway*. No entanto, assumiu-se que não havia restrições quanto aos locais de nó. Em [55], um algoritmo que foi apresentado maximiza a área de cobertura, forçando os nós sensores para mover tão longe uns dos outros quanto possível a fim de maximizar a sua área de cobertura. Alternativamente, em [56], discute um método para o controle da área de cobertura e de roteamento, baseado na descarga da bateria cada nó da malha. Embora este método pode cobrir toda a área da rede, não se considera sobreposição.

Relação Sinal-Ruído

O ruído e as fontes de interferência são capazes de influenciar significativamente o desempenho de comunicação das redes sem fio [2]. A relação entre o sinal e o ruído é descrita pela nomenclatura SNR (*Signal-to-Noise Ratio*), assim como a própria sigla já se auto justifica, todo sinal recebido será proveniente de um ruído referenciado, ou seja, onde há sinal, há ruido [57]. A diferença entre eles é mostrada em decibéis e o cálculo referente à Relação Sinal-Ruído (SNR), pode ser obtido com uso da Equação 3.3 [58].

$$SNR_{dbm} = Signal_{RSSI} - Noise_{RSSI} \tag{3.3}$$

Diante disso, com os níveis de SNR calculados, pela Equação 3.3, pode-se gerar um gráfico de Distribuição Normal dos dados. As informações são conseguidas através da Função de Gauss (Equação 3.4), em que x é a variável Gaussiana com a média representada por μ e o desvio padrão por σ.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.4}$$

Alguns estudos relacionados, como em [59] e [60], foram realizados com o objetivo de verificar o desempenho de rádios IEEE 802.15.4 sujeitos à interferência de redes IEEE 802.11 (Wi-Fi), *Bluetooth* e fornos microondas. A Figura 3.12 mostra a potência induzida nas componentes de frequência da banda de 2,4 GHz durante experimentos com uma rede IEEE 802.11 operando no canal 6, um forno microondas e uma rede IEEE 802.15.4 operando no canal 13. No cenário representado pela Figura 3.12 é possível observar que o nível de potência médio das fontes de interferência é muito alto em comparação com o nível de potência do rádio IEEE 802.15.4 [59].

Buscando apresentar alguns trabalhos direcionado a análise de ruídos pode ser mencionado [61], em que os autores verificaram as faixas de frequência afetadas por um conjunto de fontes de interferência. Os resultados mostraram que motores de combustão e equipamentos de solda causam interferência apenas abaixo de 1 GHz. No ambiente onde foram realizadas as medições observou-se grande poluição no espectro na faixa de 2,4 GHz devido a outros sistemas de baixo alcance que utilizam essa faixa, como redes Wi-Fi e *Bluetooth*.

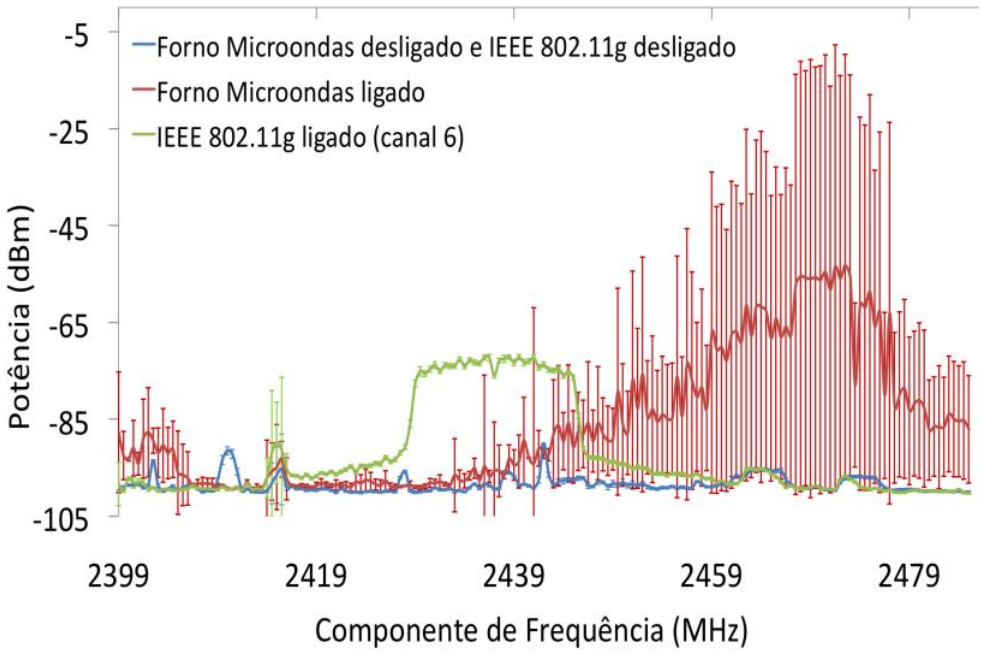


Figura 3.12: Potência induzida nas componentes de frequência [2].

Por outro lado, em [62] os autores realizaram um estudo sobre a intensidade do campo elétrico de trabalho e a distribuição de probabilidade de amplitude (APD) do ruído em uma indústria de papel. A APD é definida como o percentual de tempo em que um sinal impulsivo excede um determinado limite. Os resultados obtidos mostraram que os equipamentos mais comuns que podem causar interferência na faixa de 2,4 GHz são fornos microondas, aquecedores industriais, sistemas de iluminação por radiofrequência e equipamentos de solda. No entanto, esses equipamentos nem sempre estão presentes na indústria.

Capítulo 4

Discussão dos resultados

Esta capítulo tem o objetivo de apresentar as etapas e os resultados alcançados com execução deste projeto. Os resultados expostos na Seção 4.1 foram transcritos em forma de artigo e publicado IV Simpósio Brasileiro de Engenharia de Sistemas Computacionais (SBESC), em novembro de 2014, com o título de "*Expanding the coverage area of a formation of robots through a mesh network and a real time database middleware*" [19].

4.1 Experimento 1

Após a preparação do ambiente de rede, descritos na Seção 3.1, foram iniciados os testes, de fato, com a RTDB. Nesta etapa, implementou-se a RTDB em Rede Mesh e com Rede Ad Hoc, a fim de analisar o comportamento do ambiente proposto. A partir disso, foi realizada a adaptação no código para permitir o uso da RTDB com o ZigBee, buscando minimizar o consumo de energia.

Para que se permitisse o envio de informações distintas através da RTDB, chegou-se à conclusão de que era necessário ser executado ao simultaneamente mais de uma instância deste componente. Este cenário pode ser considerado um problema, pois várias instâncias de uma RTDB em um mesmo computador não eram suportados, conforme estudo realizado por [63]. No primeiro teste, iniciou-se as duas RTDB's com a mesma chave de memória compartilhada para as duas placas de rede. Esta chave de memória compartilhada é definida através da variável de ambiente chamada "AGENT", que recebe como parâmetro um número de identificação do agente. Com essa configuração, a informação foi distribuída para ambas as redes, causando uma redundância do fluxo de dados, o que se distancia do nosso objetivo principal. Talvez esta configuração torne-se interessante

em alguns casos, em que houvesse a necessidade de duplicação das informações trafegadas entre todos os agentes envolvidos.

Diante do exposto, o principal desafio desta etapa tornou-se realizar alterações no código da RTDB para que as múltiplas instâncias fossem inicializadas no sistema. Baseando-se no estudo descrito por [14], a saída encontrada foi garantir que a chave de memória compartilhada para cada instância RTDB fosse diferente. A solução para este problema deu-se através de uma alteração na forma de comunicação da RTDB, na qual foi imprescindível utilizar diferentes variáveis de ambiente para cada comunicação estabelecida e, com isso, obter de fato o isolamento do fluxo de dados entre as placas de redes. Assim, cada placa de rede passa a receber um contexto de dados distintos.

4.1.1 RTDB em rede Ad Hoc e em rede Mesh

De acordo com a Figura 4.1, como foi explanado anteriormente, vê-se as estações conectadas as duas redes sem fio diferentes - Ad Hoc e Mesh. A *Basestation*, por outro lado, estava conectada, através de um cabo padrão Ethernet, ao Roteador 1, que por sua vez, conectou-se ao Roteador 2 através de rede sem fio, formando, de fato, a rede Mesh.

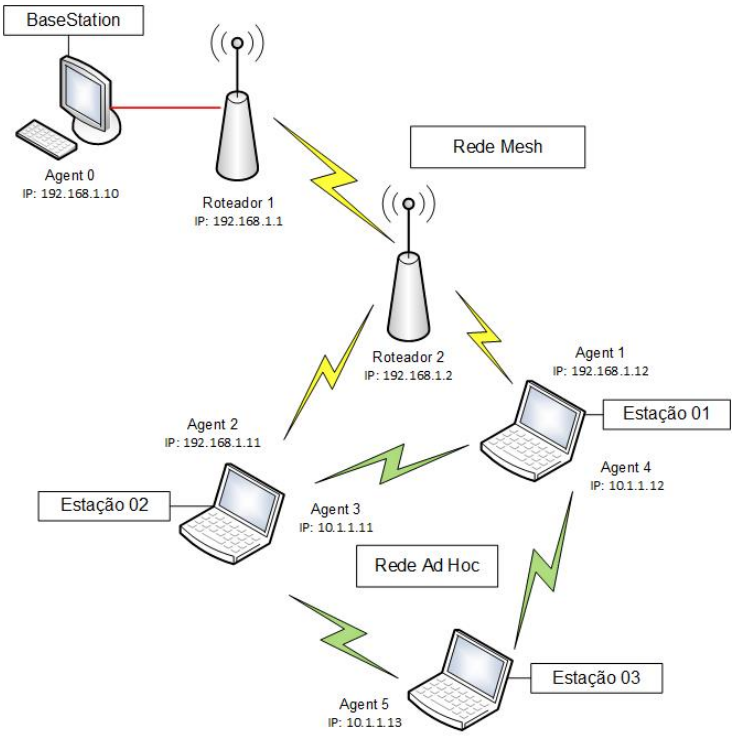


Figura 4.1: Primeiro momento.

Este exemplo possibilita que a *Basestation* leia os dados de uma possível localização referente aos agentes que possuem comunicação com a rede Mesh. Isto permite que a *Basestation* interfira na ação desses agentes, através de escritas na RTDB, mesmo que estejam geograficamente distantes. Já as escritas realizadas pelos integrantes da rede Ad Hoc apenas são ouvidas entre si. Neste caso, estas mensagens não chegam até a *Basestation* o que, de certa foram, torna-se vantajoso por não acarretar no crescimento do tráfego de informações desnecessárias na rede Mesh.

A possibilidade de execução simultânea das duas redes permitiu a expansão da área de alcance dos agentes, na qual foram criados outras possibilidades de comunicação.

4.1.2 RTDB em rede Ad Hoc e em rede Mesh, com deslocamento de um dos agentes entre roteadores

Neste momento, procurou-se analisar o comportamento da RTDB quando submetida a um deslocamento de agentes entre os roteadores que formavam a rede Mesh. Para isso, moveu-se a Estação 02, que estava conectada ao Roteador 2 e à rede Ad Hoc, em direção ao Roteador 1. Como o ambiente onde foi realizado os experimentos possuía vários obstáculos (como paredes e janelas de vidro), a aproximadamente 20 metros de distância do ponto inicial, a comunicação Ad Hoc foi interrompida.

Já a comunicação com o Roteador 2, perdeu-se quando atingiu-se cerca de 50 metros de afastamento. Porém, como esperado, em torno de 10 pacotes de transmissão foram perdidos e a comunicação à rede Mesh foi restabelecida com o auxílio do Roteador 1 em um curto espaço de tempo (aproximadamente 5 segundos). Este processo de transição com mudança do ponto de acesso é chamado de *Handoff* e, comumente, apresenta dois fatores que prejudicam aplicações de tempo real: a latência do processo e o número de pacotes descartados ou atrasados [64].

O tempo obtido não interferiu na eficiência do uso da RTDB, o que torna-se justificável devido à alta capacidade de sincronização deste tipo de base de dados. A configuração deste cenário pode ser encontrada na Figura 4.2.

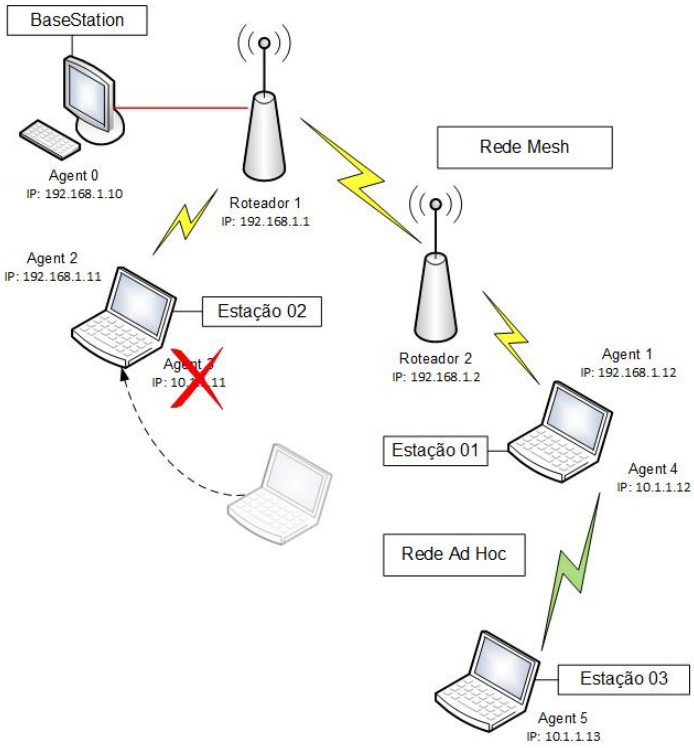


Figura 4.2: Segundo momento.

4.1.3 RTDB em rede Ad Hoc e em rede Mesh, com distanciamento de um dos agentes e com perda de conexão com a rede Mesh

Neste último momento, conforme exposto na Figura 4.3, distanciou-se o agente, denominado de Estação 02, dos roteadores da rede Mesh com o intuito de ocasionar total perda de sinal. Assim como descrito no segundo momento (Seção 4.1.2), devido às barreiras existentes no ambiente, também não foi difícil remeter a esse cenário e a conexão anteriormente estabelecida com o Roteador 2 foi interrompida à aproximadamente 50 metros de distância.

Diante disso, a Estação 02 não conseguiu estabelecer comunicação direta com a *Basestation*, passando a enxergar apenas os agentes integrantes da rede Ad Hoc. Inicialmente, este não seria um cenário ideal, uma vez que a *Basestation* perderia a sua capacidade de controle sobre este agente específico. Porém, a estrutura formada possibilita que as estações que ainda sustentem as duas comunicações, Ad Hoc e Mesh, em paralelo, como por exemplo a Estação 01, admitam que os dados cheguem até a *Basestation*, e vice-versa, fazendo com que estas estações transformassem-se em pontes de comunicação.

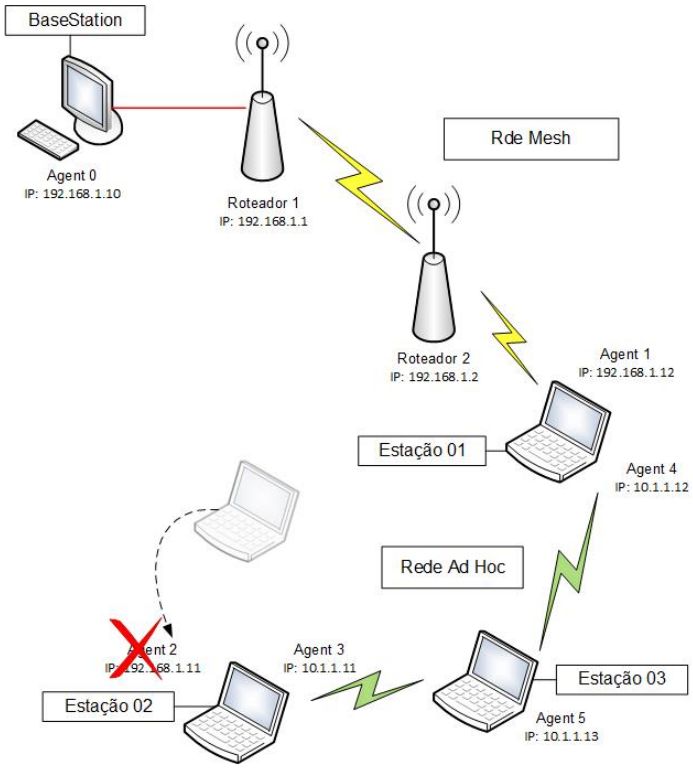


Figura 4.3: Terceiro momento.

Esta contribuição demonstra que a utilização de uma formação de robôs em uma rede Mesh justamente com a rede Ad Hoc possibilita a expansão da área de cobertura dessa formação, permitindo que os agentes participantes consigam se distanciar um do outro e ainda retornar a sua formação inicial, mesmo havendo perda de sinal entre o agente que se distanciou, para efetuar uma tarefa por exemplo, e os outros robôs integrantes de sua formação.

Deste modo, pode-se multiplicar as funções de atuação dos robôs sem o receio que os mesmos se percam um do outro pela quebra de comunicação. Contudo, apesar desse grande benefício, ressalta-se que tudo isso foi possível devido a uma adaptação no método de implementação da RTDB, permitindo assim a sua execução em paralelo, em que uma RTDB é compartilhada entre os agentes via Ad Hoc e a outra com a *Basestation*, via rede Mesh, e possibilitando que uma mesma estação passasse a representar dois agentes distintos.

Porém, esta configuração ainda pode ser otimizada, uma vez que a placa de rede responsável pela comunicação entre os robôs da mesma equipe, na rede Ad Hoc, é subutilizada com a pouca carga de dados enviada pela RTDB (ver Subseção 3.2.4). Como

solução para este problema, foi proposta a utilização das redes ZigBee como interface de comunicação entre estes robôs. Isto interfere diretamente na economia de energia (ver Subseção 2.2), fator muito importante para a autonomia dos agentes moveis. Outro ponto a ser melhorado é a execução em paralelo de duas RTDB's, com diferentes configurações, nos agentes. Este fato implica na necessidade de um maior processamento pelos robôs, o que também gera um desperdício de recursos. A solução a ser apresentada utilizada uma única RTDB, com um único arquivo de configuração (ver Código 3.3), que é compartilhada entre as duas placas de rede.

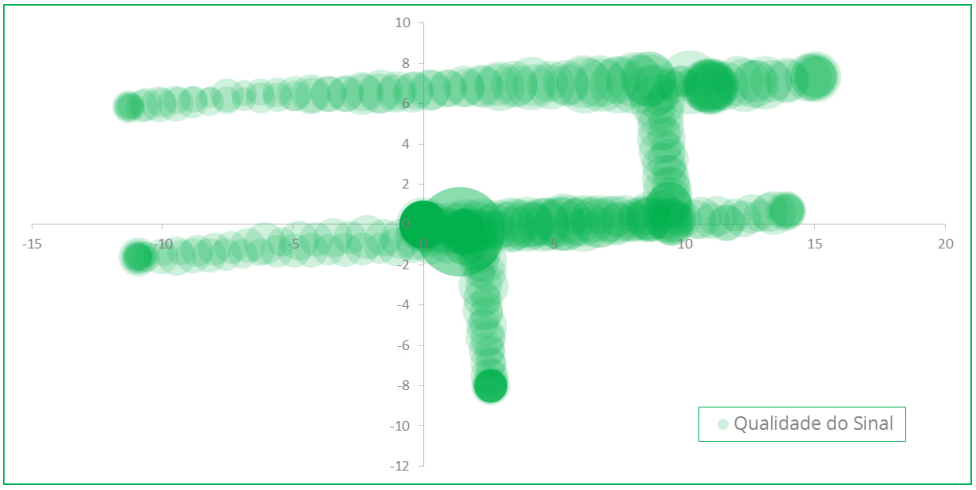
4.2 Experimento 2

Neste experimento, implementou-se a RTDB em robôs, do tipo Turtlebot 2, a fim de analisar o comportamento deles em um ambiente real. A partir disso, foi realizada a adaptação no código para permitir o uso da RTDB com o ZigBee. Os detalhamentos do ambiente, estão descritos na Seção 3.2.

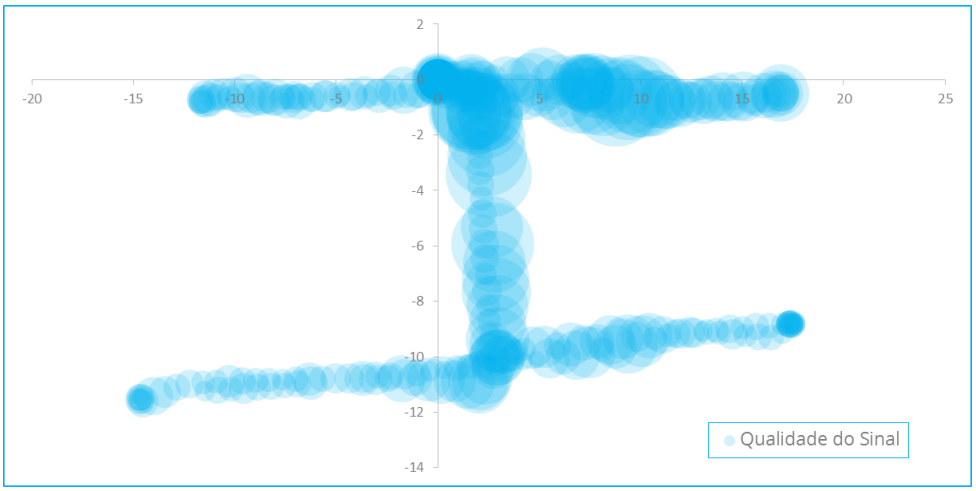
Esses experimentos realizados foram divididos em duas etapas: a primeira parte teve como objetivo principal descobrir a área de cobertura da rede sem fio, denominada "ROBOT", em que os robôs estavam conectados através da placa de rede. Em um segundo momento, foi realizada uma varredura de redes sem fio existentes na qual o sistema aferiu a intensidade do sinal de todos os pontos de acesso dentro do seu alcance, e não apenas aquele que foi designado para os robôs, calculando o valor da Relação Sinal-Ruído (SNR).

Com trajetórias pré-definidas, os Turtlebots percorreram toda a extensão dos corredores do segundo e terceiro andar do prédio do CI-UFPB, realizando assim uma varredura nas áreas de livre acesso de cada pavimento. Na Figura 4.4a, 4.4b e 4.4c, observa-se a trajetória percorrida pelos robôs #11, #12 e #17, respectivamente, em forma de gráfico em um plano cartesiano. A partir da intensidade da potência RSSI foi calculado o percentual de qualidade do sinal do recebido. Este cálculo foi realizado através da Equação 4.1 [65] e está graficalmente representada por bolhas, em que os tamanhos maiores indicam um maior nível de sinal.

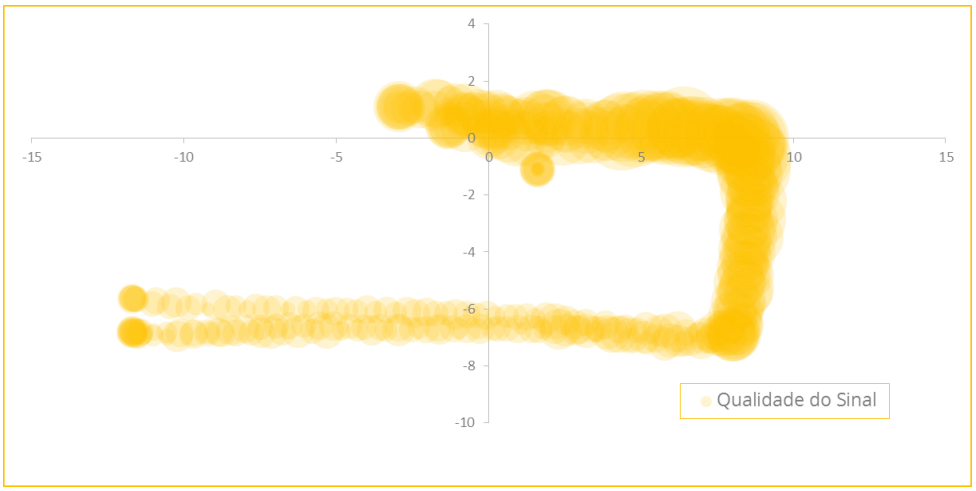
$$Q_{\%} = 2 * (RSSI_{dbm} + 100) \tag{4.1}$$



(a) Robô #11.



(b) Robô#12.



(c) Robô #17.

Figura 4.4: Trajetória dos robôs versus RSSI da rede WiFi.

Todo procedimento pode ser observado através do arquivo “Experimento.mp4“, em anexo, no qual consta o detalhamento da operação de cada robô e da *Basestation* diante das situações impostas. Este experimento teve o tempo total de 6 minutos e 12 segundos. Já a Figura 4.5 apresenta uma captura de tela do conteúdo do anexo.

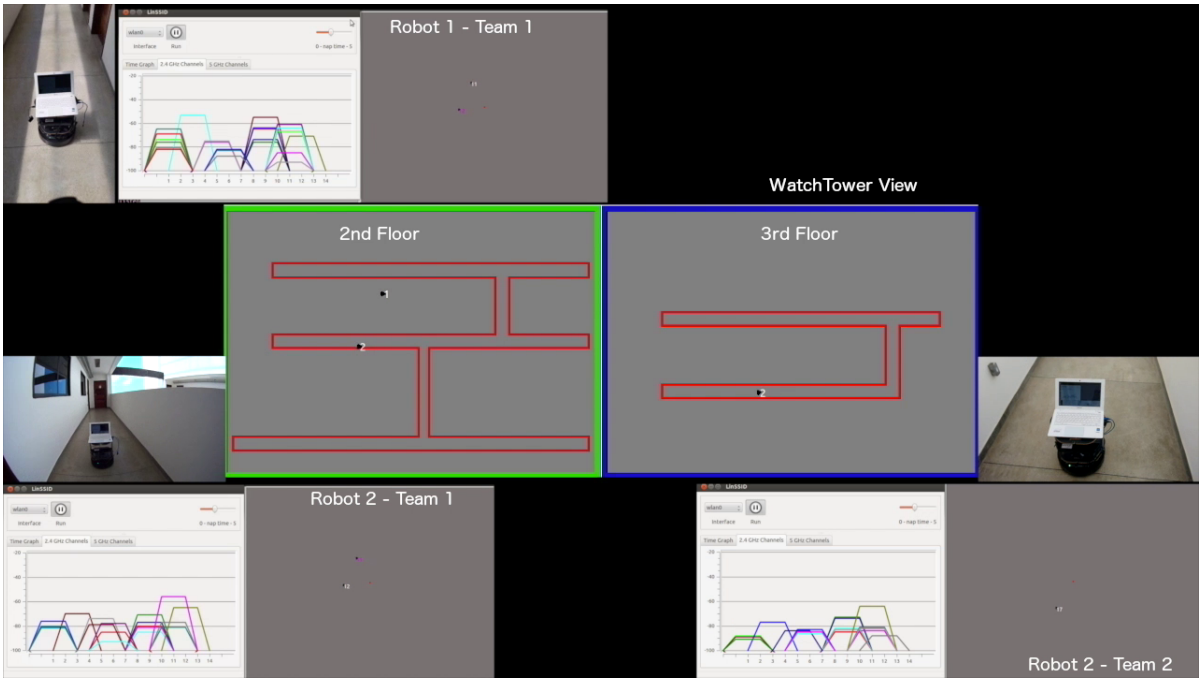


Figura 4.5: Captura de Tela do anexo Experimento.mp4.

4.2.1 Área de Cobertura

Realizando uma interseção dos dados representados pela Figura 4.4, obteve-se os gráficos que representa o mapa de cobertura analisado pelos robôs, o qual é apresentado nas Figuras 4.6 e 4.7. Um desenho como este é habitualmente chamado de mapa de calor, embora que, neste caso, as áreas mais “quentes“ (de maior qualidade de sinal) estejam representadas pela cor que mais se adequada, o verde.

Nestas Figuras 4.6 e 4.7, as áreas onde foram encontrados potência sinal forte (RSSI > -60dbm) estão sinalizadas pela cor verde. Em amarelo, estão indicadas as regiões onde foram identificados níveis de potência aceitáveis (RSSI > -80dbm). Já em zonas que

estão representadas pela cor vermelha encontram-se os níveis de sinais fraco (RSSI > -100dbm), que já merecem atenção. Em contrapartida, os lugares onde houve a perda total de sinal (RSSI <= -100dbm) foram mostrados pela cor preta.

A ausência de sinal justifica-se pela a existência de diversas salas com divisórias que, por sua vez, atenuaram o sinal rede sem fio. Mesmo com a presença dessas barreiras percebe-se que houve alguma penetração do sinal através das salas, contudo tornando-o bastante prejudicado. Essa mesma estrutura afetou a conexão entre os robôs na camada inferior (ZigBee).

No arquivo "Experimento.mp4", todo o experimentos pode ser visto com os pontos de visão simultâneos de cada robô de ambos os times e da *Basestation*. Neste video pode ser visto que a conexão de ambas as camadas de rede é mantida sem nenhum prejuízo até o minuto 2m37s em ambos os pisos. Vamos separar as análises agora entre os pisos 2 e 3.

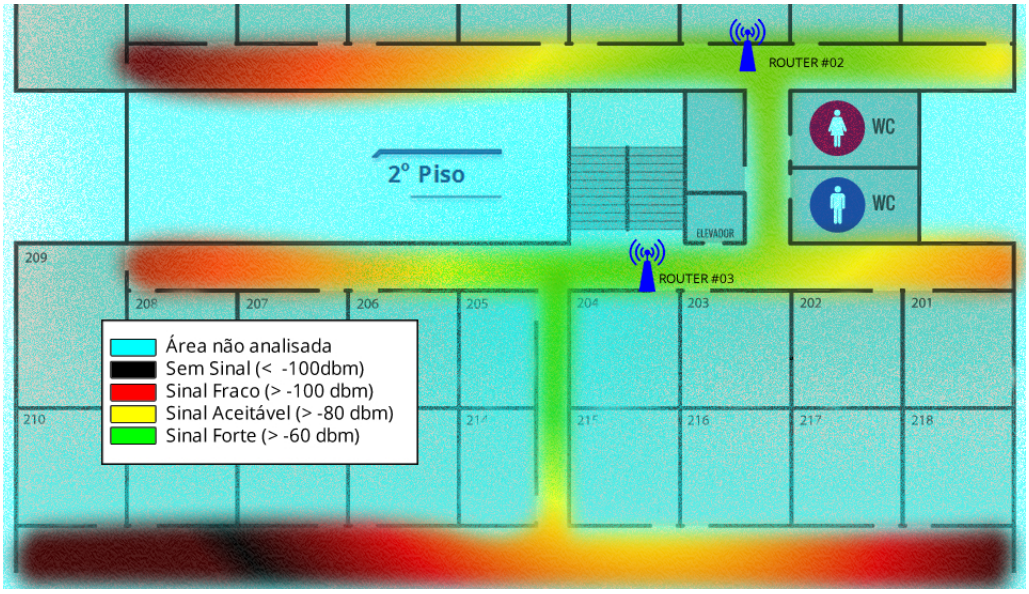


Figura 4.6: Mapa de cobertura do segundo piso.

No piso 2 ocorre o percurso do Time 1 de robôs, contendo os robôs #11 e #12. Neste piso, entre os minutos 2m38s e 3m41s, pode ser visto que a conexão entre os robôs #11 e #12, através da camada inferior (ZigBee), começa a apresentar falhas pois o robô #11 consegue visualizar o movimento do robô #12 e o inverso não ocorre. Por outro lado, a *Basestation* ainda se mantém conectada de forma estável entre ambos robôs. A conexão na camada inferior (ZigBee) entre os robôs #11 e #12 é momentaneamente restabelecida entre os minutos 3m42s e 3m47s devido a passagem de ambos por uma área de corredor aberta (sem obstruções de sinal).

A fase crítica de conexão, tanto na camada inferior (ZigBee) - entre os robôs #11

e #12 - quanto na camada superior (Mesh) - entre os robôs e a *Basestation* -, ocorre entre os minutos 3m48s e 4m23s. Durante esse período, na camada superior (Mesh), a conexão entre a a *Basestation* e o robô #11 é mantida de forma estável, porém a conexão entre a *Basestation* e o robô #12 é perdida culminando na perda de conexão do robô #12 no minuto 4m13s. Neste mesmo intervalo, na camada inferior (ZigBee), houve presença de falhas na conexão, entre os robôs #11 e #12. Durante esse tempo, o robô #11 conseguiu visualizar o movimento do robô #12, porém o inverso não ocorreu. A partir do minuto 4m24s a conexão da malha superior foi restabelecida de forma estável até o final do experimento e a *Basestation* pôde visualizar todo o movimento dos robôs #11 e #12.

Entretanto, durante os minutos 4m24s e 4m46s, a conexão da camada inferior (ZigBee) foi totalmente perdida, fazendo com que nem um dos dois robôs (#11 e #12) pudessem visualizar os movimentos um do outro. A camada de rede inferior (ZigBee) só restabeleceu sua conexão a partir do minuto 4m47s e se manteve estável até o final do experimento.

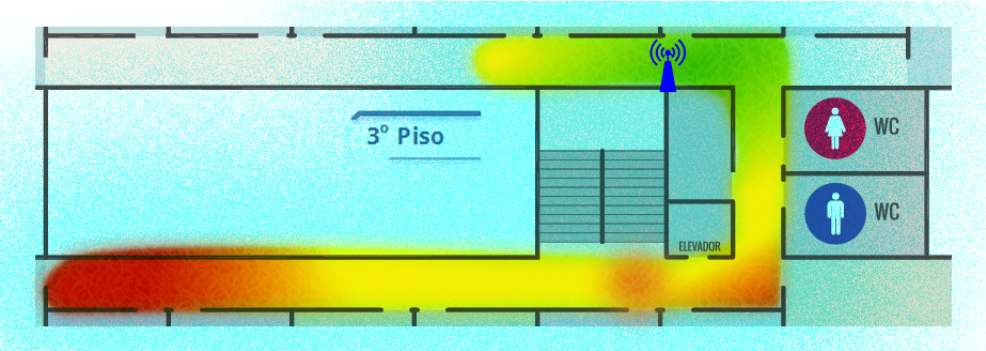


Figura 4.7: Mapa de cobertura do terceiro piso.

No piso 3 a conexão entre o único robô do Time 2 (o robô #17) e a *Basestation* se manteve estável até o minuto 4m15. Entre os minutos 4m15s e 4m23s a conexão ainda se manteve ativa, apesar da visualização do robô #17 ocorrer com um atraso na transmissão. Essa falha foi eliminada após o minuto 4m24s até o fim do experimento.

É notável que toda vez que há perda ou falha na comunicação, em ambas as camadas, os robôs cuja conexão está falhando apresentam-se como se estivessem parados na última informação de localização enviada para a RTDB. Esse fator é fundamental para a robustez do monitoramento dos robôs, pois facilita o restabelecimento da conexão perdida ou colabora com mudança de comportamento de um outro robô, que esteja com a comunicação ativa, para que haja o resgate do robô "perdido". Outro fator importante observado é que todas as falhas de conexão da camada superior (Mesh), vistas no segundo andar do prédio, podem ser fundamentadas pela passagem dos robôs em regiões pretas

ou vermelhas demonstradas nas Figuras 4.6 e 4.7, o que demonstra a necessidade de um terceiro roteador no terceiro corredor por onde houve maior falha de conexão.

Quanto à rede da camada inferior, mostrou-se que uma quebra de formação poderia ser feita fazendo robôs do mesmo time percorrer os dois corredores internos do segundo andar. Porém, para que não haja perda de conexão nesta camada (ZigBee), durante o patrulhamento do terceiro corredor, recomenda-se que o time permaneça em conjunto ou que ao menos um robô esteja localizado no corredor que interliga o terceiro corredor com a parte interna do segundo andar, como foi o caso ocorrido no minuto 4m48s.

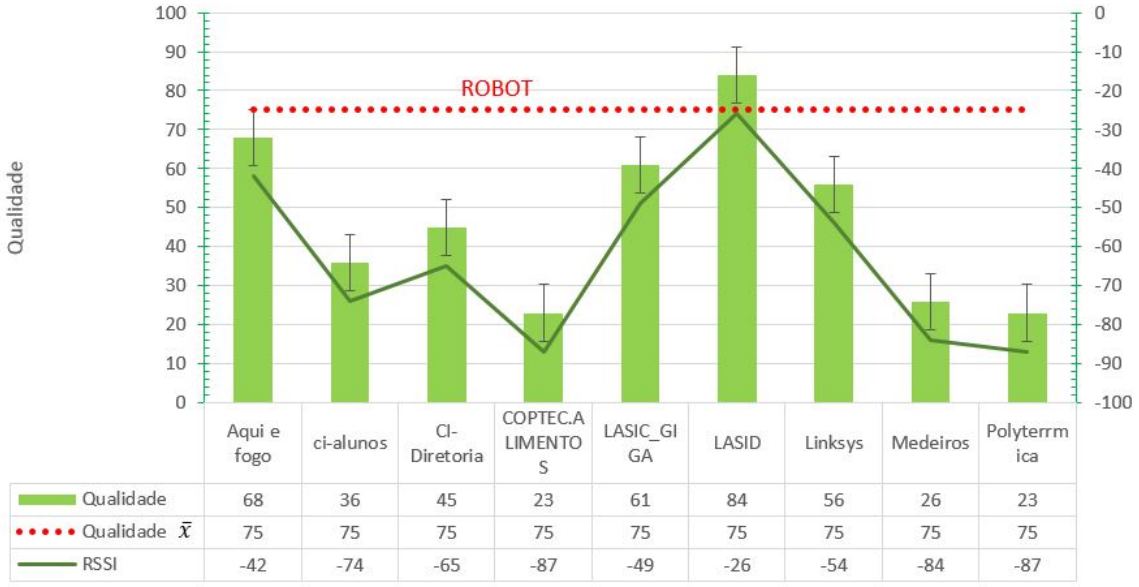
4.2.2 Varredura de Redes Sem Fio

A varredura das redes sem fio existentes foi realizada em comparação com a rede principal dos robôs ("ROBOT") que, conforme mencionado na Seção 4.2, está configurada para operar no Canal 9. Com isso, a análise foi filtrada considerando apenas as redes que operam no mesmo Canal (9) ou em canais próximos (7 a 11).

Na Figura 4.8 pode ser visto um gráfico com a lista de redes que que foram escaneadas por todos os robôs, apresentando o nível máximo de Qualidade de Sinal e RSSI e a média da Qualidade de Sinal da rede "ROBOT". Assim, para ajudar a avaliar possíveis problemas de interferência de canal e calcular o provável impacto do ruído na rede "ROBOT" foi usada uma característica da especificação do padrão 802.11, vista na Equação 4.1, que desempenha, a partir da intensidade da potência RSSI recebida, o cálculo da qualidade de uma rede. Os valores obtidos são apresentados em decibéis.

No cenário representado pela Figura 4.8 é possível observar que o nível de Qualidade de Sinal e o RSSI médio das redes encontrada são considerados baixos quando comparados ao nível de potência médio do rádio que emite o sinal da rede "ROBOT". Considerando o desvio padrão de cada rádio emissor escaneado pelos robôs, observa-se que apenas a redes denominadas de "Aqui e fogo" e "LASID" atingem um nível equivalente à rede "ROBOT". Por apresentarem valores equivalentes, ou superiores, essas redes escaneadas no ambiente podem chegar a interferir na qualidade do sinal emitido pelos radios responsáveis pela "ROBOT".

Diante disso, os níveis da Relação Sinal-Ruído (SNR - *Signal-to-Noise Ratio*) foram calculados por cada robô, gerado, assim, um gráfico de Distribuição Normal dos dados. As informações foram conseguidas através da Função de Gauss, apresentada na Equação 3.4,



	Aqui e fogo	ci-alunos	CI-Diretoria	COPTEC.ALIMENTOS	LASIC_GIGA	LASID	Linksys	Medeiros	Polyterrmica
Qualidade	68	36	45	23	61	84	56	26	23
Qualidade $\bar{x}$	75	75	75	75	75	75	75	75	75
RSSI	-42	-74	-65	-87	-49	-26	-54	-84	-87

Figura 4.8: Lista de redes escaneadas.

e os valores obtidos foram apresentados na Figura 4.9, na qual o gráfico da Gaussiana foi dada por meio das curvas de sino mostradas.

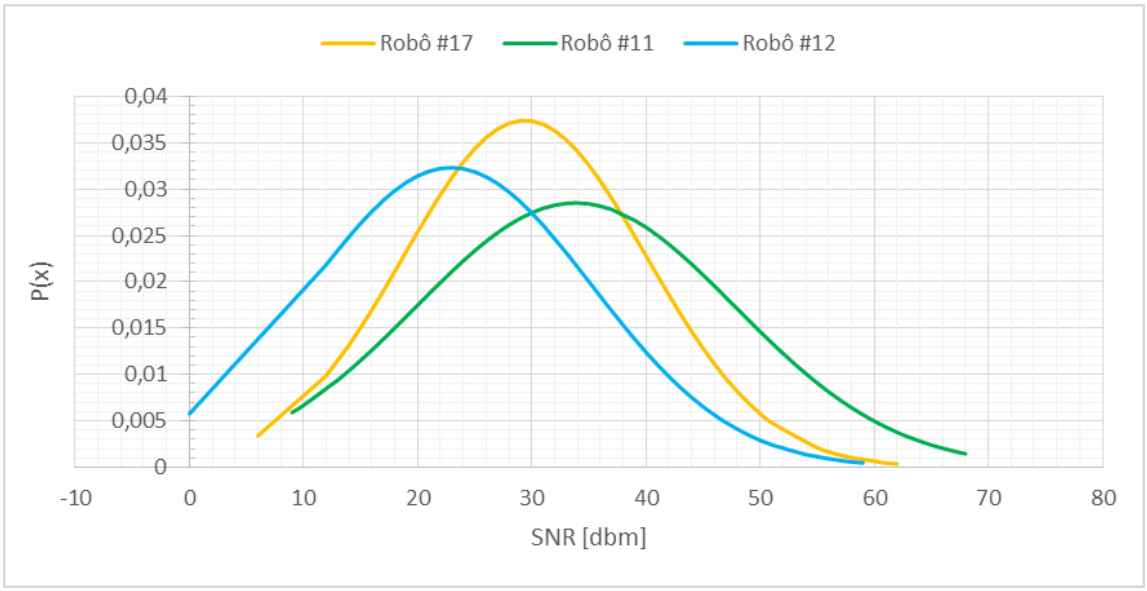


Figura 4.9: Gráfico da Distribuição Normal Relação Sinal-Ruído (SNR).

Com isso, em comparação ao valores expostos na na Tabela 4.1 [66], tem-se que quanto mais alta for a Relação Sinal-Ruído melhor a qualidade do sinal, a taxa de transmissão e menor é o efeito do ruído de fundo sobre a detecção ou medição do sinal [67]. Com os valores médios do SNR em **33**, **22** e **29**, obtidos pelos robôs #11, #12 e #17 respectivamente, qualifica-se as medidas do SNR conseguidas no experimento como acei-

táveis e com um nível não impactante para a desempenho da rede principal dos robôs, a "ROBOT".

Tabela 4.1: SNR (dB) vs. taxa de dados (Mbps).

SNR necessário (dB) para atingir a taxa de dados (Mbps).

Taxa de Dados	6	9	12	18	24	36	48	54
SNR	6,02	7,78	9,03	10,79	17,04	18,80	24,05	24,56

Com o propósito de prevenir possíveis interferências de canal com a rede IEEE 802.15.4 (ZigBee), conforme explanado na Seção 3.2.2, o módulo XBee foi configurado para trabalhar no Canal 26 (hex. 0x1A) e em frequência de 2,480GHz. A utilização desta configuração descarta a possibilidade de produzir de conflitos com os demais canais da rede IEEE 802.11, uma vez que o canal 26 do ZigBee encontra-se fora da faixa de canais e frequências autorizadas para o 802.11 (Canal 14 : 2,477 Mhz) [68], como pode ser visto na Figura 2.2.

Capítulo 5

Conclusão

Esta dissertação demonstrou uma forma de expandir a área de cobertura de uma rede sem fio, a ser utilizada em sistemas multi-robôs, usando um *Middleware* de Base de Dados em Tempo Real - RTDB, em redes multicamadas. A contribuição garantiu a comunicação de robôs integrantes de um sistema multi-robôs, permitindo, assim, um maior alcance na área de cobertura da formação, o que possibilitou aos robôs distanciarem-se uns dos outros e retornarem à sua formação inicial com segurança.

Os resultados obtidos, em um primeiro momento, mostraram que a utilização de uma formação de robôs utilizando a RTDB em uma rede em malha (Mesh) e em uma rede Ad hoc permite um maior alcance na área de cobertura da formação. Esta rede de duas camadas proporcionou outras oportunidades de comunicação, em que uma RTDB é compartilhada entre robôs através Ad Hoc e a outra com a um computador central (*Basestation*) através da rede em malha (Mesh), admitindo que uma mesma estação pudesse representar dois agentes distintos, permitindo o sincronismo de dados entre entre as formações de robôs e a *Basestation* e, também, entre os robôs que estão conectado à mesma rede Ad Hoc. Ainda foi obtido um valor de handoff de 5 segundos, em média, quando um robô se desconectava de um nó e se conectava a outro, na rede Mesh.

No segundo experimento, os resultados obtidos com o estudo realizado demonstram a robustez e confiabilidade do cenário proposto no qual tem-se, novamente, o uso da RTDB em rede multicamadas, agora utilizando o padrão IEEE 802.15.4 (ZigBee) na camada inferior. Este benefício permitiu que os robôs conseguissem realizar com sucesso uma análise da Área de Cobertura e uma varredura de Redes Sem Fio. Para isso, foi executado uma exploração do ambiente observando o nível de RSSI (*Received Signal Strength Indication*) obtido pelos robôs e uma varredura buscando redes adjacentes à rede em que robôs estavam conectados. Os experimentos foram realizados com robôs reais, do tipo

Turtlebot 2, que utilizaram a RTDB como forma de compartilhar informações entre os membros de uma mesma equipe e um computador de monitoramento central (*BaseStation*) de maneira isolada. Além disso, pôde-se multiplicar as funções de atuação dos robôs sem o receio que eles se percam de sua formação por causa da perda de comunicação, uma vez que o monitoramento acontece em tempo real.

Em todos os experimentos, o sucesso só foi possível devido à junção de topologias de rede que utilizaram a RTDB em rede Ad Hoc ou Ad Hoc ZigBee na camada inferior e uma rede em malha (Mesh), na camada superior.

Salienta-se que os resultados aqui apresentados podem conter divergências quando submetidos a situações diferentes das descritas neste experimento, tais como: alteração da configuração de *hardware* dos equipamentos utilizados, condições climáticas, layout do ambiente entre outros.

5.1 Perspectiva de Trabalhos Futuros

Como trabalhos futuros almeja-se realizar estudos e implementações onde os robôs TurtleBot poderão se beneficiar ainda mais da captura de informações do RSSI, passando a realizar uma estimativa de localização no mapa, permitindo assim a realização automatizada de um *Site Survey* no ambiente em que estejam inseridos. Em outra possibilidade de melhoria, serão acrescentados mais nós na rede em malha utilizando algoritmos de alocação de roteadores na camada superior da rede para expandir a rede para todos os andares do prédio do Centro de Informática-UFPB, potencializando o uso do sistemas Multi-robôs com o menor número possível de roteadores. Já em outro ponto, a rede superior também pode ser melhorada alterando a sua arquitetura de comunicação que passaria a utilizar o HLA (*High-level architecture*), o que permitiria uma melhor interação com diversas outras plataformas de computação, como as simulações por exemplo. Por fim, através de estudos referentes a problemas de alocação de recursos na rede, pode-se utilizar a determinação do mapa de interferência e área de cobertura como porta de entrada para implementação de técnicas de alocação otimizada destes recursos.

Referências Bibliográficas

[1] J. Bardwell, "Converting signal strength percentage to dbm values," em *VP of Professional Services*, Nov 2002. [Online]. Disponível em: http://madwifi-project.org/attachment/wiki/UserDocs/RSSI/Converting_Signal_Strength.pdf

[2] R. Gomes, M. Alencar, I. Fonseca, e A. L. Filho, "Desafios de redes de sensores sem fio industriais," *Revista de Tecnologia da Informação e Comunicação*, vol. 4, no. 1, pp. 16–27, Dez 2013. [Online]. Disponível em: http://dx.doi.org/10.12721/2237-5112.v04n01a03

[3] A. S. L. Fernandes, "Comunicação Ad Hoc em Equipas de Robôs Móveis Utilizando a Tecnologia ZigBee," Dissertação de Mestrado, University of Coimbra, Portugal, 2012.

[4] M. Y. d. O. Camada, "Masim: Uma ferramenta para simulaçÃo de agentes móveis em redes de sensores sem fio," Dissertação de Mestrado, Universidade Federal de Santa Catarina, Brasil, 2009.

[5] K.-J. Lin e F. Jahanian, "Requirements and issues on real-time database systems," em *Real-Time Database Systems*, ser. The Springer International Series in Engineering and Computer Science, A. Bestavros, K.-J. Lin, e S. Son, Eds. Springer US, 1997, vol. 396, pp. 17–38. [Online]. Disponível em: http://dx.doi.org/10.1007/978-1-4615-6161-3_2

[6] T. P. Nascimento, A. P. Moreira, e A. G. S. Conceição, "Multi-robot nonlinear model predictive formation control: Moving target and target absence," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1502 – 1515, 2013. [Online]. Disponível em: http://www.sciencedirect.com/science/article/pii/S0921889013001310

[7] G. Corrente, J. Cunha, R. Sequeira, e N. Lau, "Cooperative robotics: Passes in robotic soccer," em *Autonomous Robot Systems (Robotica), 2013 13th International Conference on*, Abr 2013, pp. 1–6.

[8] A. Ahmad, T. P. Nascimento, A. G. S. Conceição, A. P. Moreira, e P. Lima, "Perception-Driven Multi-Robot Formation Control," em *Proceedings of the 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany*, 2013, pp. 1851–1856.

[9] F. Santos, L. Almeida, P. Pedreiras, L. S. Lopes, e T. Facchinetti, "An adaptive tdma protocol for soft real-time wireless communication among mobile computing agents," em *Proceedings of the Workshop on Architectures for Cooperative Embedded Real-Time Systems (satellite of RTSS 2004*, 2004, pp. 5–8.

[10] J. Byun, A. Burns, R. Davis, e A. Wellings, "A worst-case behaviour analysis for hard real-time transactions," em *Real-Time Database Systems*, ser. The Springer International Series in Engineering and Computer Science, A. Bestavros, K.-J. Lin, e S. Son, Eds. Springer US, 1997, vol. 396, pp. 235–249. [Online]. Disponível em: http://dx.doi.org/10.1007/978-1-4615-6161-3_14

[11] F. Santos, L. Almeida, e L. Lopes, "Self-configuration of an adaptive tdma wireless communication protocol for teams of mobile robots," em *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, 2008, pp. 1197–1204.

[12] M. Rabbi, M. Rahman, M. Uddin, e G. Salehin, "An efficient wireless mesh network: A new architecture," em *Communication Technology, 2006. ICCT '06. International Conference on*, 2006, pp. 1–5.

[13] S. Faccin, C. Wijting, J. Kenckt, e A. Damle, "Mesh wlan networks: concept and system design," *Wireless Communications, IEEE*, vol. 13, no. 2, pp. 10–17, 2006.

[14] S. Liese, D. Wu, e P. Mohapatra, "Experimental characterization of an 802.11b wireless mesh network," em *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, ser. IWCMC '06. New York, NY, USA: ACM, 2006, pp. 587–592. [Online]. Disponível em: http://doi.acm.org.ez15.periodicos.capes.gov.br/10.1145/1143549.1143666

[15] I. Akyildiz e X. Wang, "A survey on wireless mesh networks," *Communications Magazine, IEEE*, vol. 43, no. 9, pp. S23–S30, 2005.

[16] S. M. Das, Y. C. Hu, C. S. G. Lee, e Y.-H. Lu, "Mobility-Aware Ad Hoc Routing Protocols for Networking Mobile Robot Teams," *Journal of Communications and Networks*, vol. 9, no. 3, pp. 296–311, 2007.

[17] A. Howard, M. Matarić, e G. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," em *Distributed Autonomous Robotic Systems 5*, H. Asama, T. Arai, T. Fukuda, e T. Hasegawa, Eds. Springer Japan, 2002, pp. 299–308. [Online]. Disponível em: http://dx.doi.org/10.1007/978-4-431-65941-9_30

[18] C. Q. Nguyen, B.-C. Min, E. T. Matson, A. H. Smith, J. E. Dietz, e D. Kim, "Using mobile robots to establish mobile wireless mesh networks and increase network throughput," *International Journal of Distributed Sensor Networks*, vol. 2012, no. 614532, pp. 1–13, 2012.

[19] L. Caetano, T. Nascimento, M. Oliveira, e G. Rocha, "Expanding the coverage area of a formation of robots through a mesh network and a real time database middleware," em *Computing Systems Engineering (SBESC), 2014 Brazilian Symposium on*, Nov 2014, pp. 138–143.

[20] "Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[21] "Ieee standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements. - part 15.1: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpans)," *IEEE Std 802.15.1-2005 (Revision of IEEE Std 802.15.1-2002)*, pp. 1–580, 2005.

[22] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 15.4: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpans)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, pp. 1–320, Sept 2006.

[23] G. Aggélou, "Wireless mesh networking: [with 802.16, 802.11 and zigbee]," 2009.

[24] A. V. Brito, G. S. Oliveira, e L. J. Caetano, "Uma análise da implementação zigbee pela tecnologia sun spot," em *CSBC 2009 - WIM (IX Workshop de Informática Médica)*, jul 2009. [Online]. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/wim/2009/011.pdf

[25] A. Leonov, "As redes zigbee™ protegem os prédios contra a ameaça de afundamento," em *A Revista de Micro e Nanotecnologia do Pólo Industrial de Manaus (MINAPIM)*, 2007.

[26] C. Chang e J. Jhu, "Zigbee-assisted mobile robot gardener," em *Automatic Control Conference (CACS), 2013 CACS International*, Dez 2013, pp. 41–46.

[27] S. Son, C. Iannacone, e M. Poris, "Rtdb: a real-time database manager for time-critical applications," em *Real Time Systems, 1991. Proceedings., Euromicro '91 Workshop on*, Jun 1991, pp. 207–214.

[28] E. Pavlova e D. V. Hung, "A formal specification of the concurrency control in real-time databases," em *Software Engineering Conference, 1999. (APSEC '99) Proceedings. Sixth Asia Pacific*, 1999, pp. 94–101.

[29] M. Z. Silva. (2014, Nov.) Modelo cliente-servidor. (Acessado em: 15-Dez-2014). [Online]. Disponível em: http://www.test.org/doe/

[30] L. Almeida, F. Santos, T. Facchinetti, P. Pedreiras, V. Silva, e L. Lopes, "Coordinating distributed autonomous agents with a real-time database: The cambada project," em *Computer and Information Sciences - ISCIS 2004*, ser. Lecture Notes in Computer Science, C. Aykanat, T. Dayar, e b. Körpeo?lu, Eds. Springer Berlin Heidelberg, 2004, vol. 3280, pp. 876–886.

[31] Página da rtdb. (Acessado em: 13-Jun-2014). [Online]. Disponível em: https://code.google.com/p/rtdb/

[32] C. P. IEETA. (Acessado em: 02-Fev-2015). [Online]. Disponível em: http://wiki.ieeta.pt/wiki/index.php/CAMBADA

[33] Página da robocup. (Acessado em: 11-Jun-2014). [Online]. Disponível em: http://www.robocup.org

[34] H. Kopetz, *Real-Time Systems: Design Principles for Distributed Embedded Applications*, 1st ed. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[35] A. M. d. S. Barros. (2003, Fev.) Ansi c para quem tem pressa. (Acessado em: 05-Dez-2014). [Online]. Disponível em: http://www.dei.isep.ipp.pt/~abarros/docs/ANSI_C.pdf

[36] F. Santos, L. Almeida, P. Pedreiras, e L. Lopes, "A real-time distributed software infrastructure for cooperating mobile autonomous robots," em *Advanced Robotics, 2009. ICAR 2009. International Conference on*, 2009, pp. 1–6.

[37] O. C. Branquinho, "Administração e projetos de redes: Tecnologia de redes sem fio - arquiteturas de redes ieee 802.11." Escola Superior de Redes RNP, 2009.

[38] R. N. de Ensino e Pesquisa). (2013) Multicast - rnp. (Acessado em: 07-Abr-2014). [Online]. Disponível em: http://www.rnp.br/_arquivo/documentos/div0113.pdf

[39] Antonio J. R. Neves, Jose Luis Azevedo, Bernardo Cunha, Nuno Lau, Joao Silva, Frederico Santos, Gustavo Corrente, Daniel A. Martins, Nuno Figueiredo, Artur Pereira, Luis Almeida, Luis Seabra Lopes, Armando J. Pinho, Joao Rodrigues and Paulo Pedreiras, *CAMBADA Soccer Team: from Robot Architecture to Multiagent Coordination.* Robot Soccer, Vladan Papi (Ed.), 2010. [Online]. Disponível em: http://www.intechopen.com/books/robot-soccer/cambada-soccer-team-from-robot-architecture-to-multiagent-coordination

[40] Y. yuan Xiao, H. Zhang, e F. yu Wang, "Maintaining temporal consistency in real-time database systems," em *Convergence Information Technology, 2007. International Conference on*, Nov 2007, pp. 1627–1633.

[41] F. Wallhoff, T. Rehrl, J. Gast, A. Bannat, e G. Rigoll, "Multimodal data communication for human-robot interactions," em *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on*, Jun 2009, pp. 1146–1149.

[42] L. Oliveira, L. Almeida, e F. Santos, "A loose synchronisation protocol for managing rf ranging in mobile ad-hoc networks," em *RoboCup 2011: Robot Soccer World Cup XV*, ser. Lecture Notes in Computer Science, T. Röfer, N. Mayer, J. Savage, e U. Saranl?, Eds. Springer Berlin Heidelberg, 2012, vol. 7416, pp. 574–585. [Online]. Disponível em: http://dx.doi.org/10.1007/978-3-642-32060-6_49

[43] Página do dd-wrt. (Acessado em: 15-Jun-2014). [Online]. Disponível em: http://www.dd-wrt.com

[44] D. I. Inc., "User's guides: X-ctu configuration & test utility software," Nov 2014. [Online]. Disponível em: http://ftp1.digi.com/support/documentation/90001003_A.pdf

[45] J. Romkey, "Nonstandard for transmission of IP datagrams over serial lines: SLIP," RFC 1055 (INTERNET STANDARD), Internet Engineering Task Force, Jun. 1988. [Online]. Disponível em: https://tools.ietf.org/html/rfc1055

[46] D. I. Inc., "Product manual: Ieee®802.15.4 rf modules by digi international," Nov 2014. [Online]. Disponível em: https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf

[47] L. Tang, K.-C. Wang, Y. Huang, e F. Gu, "Channel characterization and link quality assessment of ieee 802.15.4-compliant radio for factory environments," *Industrial Informatics, IEEE Transactions on*, vol. 3, no. 2, pp. 99–110, Mai 2007.

[48] Q. Zhang, Q. Di, G. Xu, e X. Qiu, "A rssi based localization algorithm for multiple mobile robots," em *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on*, vol. 4, Aug 2010, pp. 190–193.

[49] H.-J. Im, C.-E. Lee, Y.-J. Cho, e S. Kim, "Rssi-based control of mobile cooperative robots for seamless networking," em *Control, Automation and Systems (ICCAS), 2012 12th International Conference on*, Out 2012, pp. 980–982.

[50] N. Zhou, X. Zhao, e M. Tan, "Rssi-based mobile robot navigation in grid-pattern wireless sensor network," em *Chinese Automation Congress (CAC), 2013*, Nov. 2013, pp. 497–501.

[51] T. Komatsu, T. Ohkubo, K. Kobayashi, K. Watanabe, e Y. Kurihara, "A study of rssi-based formation control algorithm for multiple mobile robots," em *SICE Annual Conference 2010, Proceedings of*, Aug 2010, pp. 1127–1130.

[52] G. Hasegawa, S. Takemori, Y. Taniguchi, e H. Nakano, "Determining coverage area using voronoi diagram based on local information for wireless mesh networks," em *Information Technology: New Generations (ITNG), 2012 Ninth International Conference on*, April 2012, pp. 71–76.

[53] J.-H. Huang, L.-C. Wang, e C.-J. Chang, "Throughput-coverage tradeoff in a scalable wireless mesh network," *Journal of Parallel and Distributed*

Computing, vol. 68, no. 3, pp. 278–290, mar 2008. [Online]. Disponível em: http://dx.doi.org/10.1016/j.jpdc.2007.10.003

[54] S. V. R. M. W. S. M. Allen, S. Hurley, "Assessing coverage in wireless mesh networks," 2005.

[55] S. Poduri e G. Sukhatme, "Constrained coverage for mobile sensor networks," em *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 1, April 2004, pp. 165–171 Vol.1.

[56] C. Ma, M. Ma, e Y. Yang, "A battery aware scheme for energy efficient coverage and routing in wireless mimo mesh networks," em *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, March 2007, pp. 3603–3608.

[57] R. Barion. Qual a relação entre o sinal e o ruído? (Acessado em: 15-Jun-2015). [Online]. Disponível em: http://www.entelco.com.br/blog/tag/o-que-e-snr/

[58] B. Krishnamachari, A. L. Murphy, e N. Trigoni, Eds., *Wireless Sensor Networks*. Springer International Publishing, 2014. [Online]. Disponível em: http://dx.doi.org/10.1007/978-3-319-04651-8

[59] A. Lima-Filho, R. Gomes, M. Adissi, T. Borges da Silva, F. Belo, e M. Spohn, "Embedded system integrated into a wireless sensor network for online dynamic torque and efficiency monitoring in induction motors," *Mechatronics, IEEE/ASME Transactions on*, vol. 17, no. 3, pp. 404–414, June 2012.

[60] R. Gomes, M. Spohn, A. Lima, E. Gomes dos Anjos, e F. Belo, "Correlation between spectral occupancy and packet error rate in ieee 802.15.4-based industrial wireless sensor networks," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 10, no. 1, pp. 1312–1318, Jan 2012.

[61] P. Angskog, C. Karlsson, J. Coll, J. Chilo, e P. Stenumgaard, "Sources of disturbances on wireless communication in industrial and factory environments," em *Electromagnetic Compatibility (APEMC), 2010 Asia-Pacific Symposium on*, April 2010, pp. 281–284.

[62] J. Chilo, C. Karlsson, P. Angskog, e P. Stenumgaard, "Emi disruptive effect on wireless industrial communication systems in a paper plant," em *Electromagnetic Compatibility, 2009. EMC 2009. IEEE International Symposium on*, Aug 2009, pp. 221–224.

[63] E. F. Pedrosa, "Simulated environment for robotic soccer agents," Dissertação de Mestrado, Universidade de Aveiro, 2010. [Online]. Disponível em: http://hdl.handle.net/10773/3704

[64] A. O. da Silva e Sérgio Colcher, "Evolução da otimização do handoff no mobile ip," Monografias em Ciência da Computação, Pontificia Universidade Catolica do Rio de Janeiro (PUC-Rio), 2008. [Online]. Disponível em: ftp://ftp.inf.puc-rio.br/pub/docs/techreports/08_27_silva.pdf

[65] M. Windows Dev Center. Wlan_association_attributes structure. (Acessado em: 05-Jun-2015). [Online]. Disponível em: https://msdn.microsoft.com/en-us/library/windows/desktop/ms706828(v=vs.85).aspx

[66] M. Vieira, R. Govindan, e G. Sukhatme, "Towards autonomous wireless backbone deployment in highly-obstructed environments," em *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 5369–5374.

[67] M. A. Vieira, R. Govindan, e G. S. Sukhatme, "An autonomous wireless networked robotics system for backbone deployment in highly-obstructed environments," *Ad Hoc Networks*, vol. 11, no. 7, pp. 1963 – 1974, 2013, theory, Algorithms and Applications of Wireless Networked RoboticsRecent Advances in Vehicular Communications and Networking. [Online]. Disponível em: http://www.sciencedirect.com/science/article/pii/S1570870512001473

[68] K. (en.kioskea.net). How to choose the best wifi channel. (Acessado em: 15-Jun-2015). [Online]. Disponível em: http://ccm.net/faq/network-11#2142

Apêndice A

Tutorial: Como instalar a RTDB - Real-Time Database

Este tutorial tem como objetivo auxiliar na instalação e configuração da RTDB - Real-Time Database (https://code.google.com/p/rtdb/), que consiste em uma base de dados distribuída em tempo real, que possibilita que dados locais, não persistentes, sejam compartilhados e acessados entre sistemas móveis, simulando um acesso local. Como pré-requisito deve-se obter uma instalação de um sistema operacional Linux. Neste caso utilizou-se a distribuição Debian 7. Inicialmente, transfira para o sistema o arquivo chamado rtdb.tar.bz2, anexado a esse tutorial.

Vamos ao tutorial passo a passo:

1. Abra uma janela com um terminal. (OBS.: A partir deste momento, todos os comandos deverão ser realizados com o usuário "root", que pode ser obtido através do comando "su").

2. Acesse a pasta para onde o arquivo foi movido e descompacte-os. Digite o seguinte comando e tecle Enter.

```
root@debian:/home/leandro/Downloads# tar -xjf rtdb.tar.bz2
```

Figura A.1: Comando 1

3. Após isso, uma pasta chamada "rtdb" será criada. Acesse a pasta e observe se os arquivos foram realmente descompactados, através dos comandos a seguir:

4. Para possibilitar a instalação do RTDB, devemos instalar alguns outros pacotes auxiliares. No caso do Debian, deve-se instalar os pacotes "build-essential" e "cmake", conforme comando a seguir:

```
root@debian:/home/leandro/Downloads# cd rtdb
root@debian:/home/leandro/Downloads/rtdb# ls -la
total 104
drwxr-xr-x 9 leandro leandro  4096 Nov 27 19:17 .
drwxr-xr-x 3 leandro leandro  4096 Mar 10 22:21 ..
drwxr-xr-x 2 leandro leandro  4096 Nov 27 19:15 bin
-rw-r--r-- 1 root    root    11332 Mar 10 23:05 CMakeCache.txt
drwxr-xr-x 5 root    root     4096 Mar 10 23:05 CMakeFiles
-rw-r--r-- 1 root    root     1923 Mar 10 23:05 cmake_install.cmake
-rw-r--r-- 1 leandro leandro   279 Nov 27 19:14 CMakeLists.txt
drwxr-xr-x 3 leandro leandro  4096 Mar 11 05:35 comm
drwxr-xr-x 2 leandro leandro  4096 Mar 11 05:35 config
-rw-r--r-- 1 leandro leandro 35147 Feb 28  2010 COPYING
drwxr-xr-x 3 leandro leandro  4096 Mar 11 05:35 example
drwxr-xr-x 2 leandro leandro  4096 Nov 27 19:15 lib
-rw-r--r-- 1 root    root     5151 Mar 10 23:05 Makefile
-rw-r--r-- 1 leandro leandro  1690 Nov 27 19:17 README.TXT
drwxr-xr-x 3 leandro leandro  4096 Mar 11 05:35 rtdb
root@debian:/home/leandro/Downloads/rtdb#
```

Figura A.2: Comando 2

```
root@debian:/home/leandro/Downloads/rtdb# apt-get install build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  binutils dpkg-dev fakeroot g++ g++-4.7 gcc gcc-4.7 libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libc-bin libc-dev-bin
  libc6 libc6-dev libc6-i686 libdpkg-perl libfile-fcntllock-perl libitm1
  libstdc++6-4.7-dev linux-libc-dev make manpages-dev
Suggested packages:
  binutils-doc debian-keyring g++-multilib g++-4.7-multilib gcc-4.7-doc
  libstdc++6-4.7-dbg gcc-multilib autoconf automake1.9 libtool flex bison gdb
  gcc-doc gcc-4.7-multilib libmudflap0-4.7-dev gcc-4.7-locales libgcc1-dbg
  libgomp1-dbg libitm1-dbg libquadmath0-dbg libmudflap0-dbg libcloog-ppl0
  libppl-c2 libppl7 binutils-gold glibc-doc libstdc++6-4.7-doc make-doc
The following NEW packages will be installed:
  binutils build-essential dpkg-dev fakeroot g++ g++-4.7 gcc gcc-4.7
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libc-dev-bin libc6-dev libdpkg-perl libfile-fcntllock-perl libitm1
  libstdc++6-4.7-dev linux-libc-dev make manpages-dev
The following packages will be upgraded:
  libc-bin libc6 libc6-i686
3 upgraded, 20 newly installed, 0 to remove and 28 not upgraded.
Need to get 39.4 MB of archives.
After this operation, 86.9 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

Figura A.3: Comando 3

Isto instalará as dependências necessárias para o RTDB funcionar.

5. Para instalar, de fato, o RTDB execute os comandos a seguir:

   Observe se as mensagens de retorno são idênticas as apresentadas nas figuras anteriores.

6. Para uma simples verificação do sucesso da instalação, execute o programa respon-

```
root@debian:/home/leandro/Downloads/rtdb# apt-get install cmake
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  cmake-data emacsen-common libcurl3 libxmlrpc-core-c3
The following NEW packages will be installed:
  cmake cmake-data emacsen-common libcurl3 libxmlrpc-core-c3
0 upgraded, 5 newly installed, 0 to remove and 28 not upgraded.
Need to get 6,694 kB of archives.
After this operation, 16.1 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

Figura A.4: Comando 4

```
root@debian:/home/leandro/Downloads/rtdb# cmake .
-- The C compiler identification is GNU 4.7.2
-- The CXX compiler identification is GNU 4.7.2
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Looking for include file pthread.h
-- Looking for include file pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found.
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /home/leandro/Downloads/rtdb
```

Figura A.5: Comando 5

sável pela comunicação e compare a mensagem retornada.

```
root@debian:/home/leandro/Downloads/rtdb# make
Scanning dependencies of target rtdb
[ 20%] Building C object rtdb/CMakeFiles/rtdb.dir/rtdb_api.c.o
Linking C static library ../lib/librtdb.a
[ 20%] Built target rtdb
Scanning dependencies of target comm
[ 40%] Building C object comm/CMakeFiles/comm.dir/multicast.c.o
[ 60%] Building C object comm/CMakeFiles/comm.dir/comm.c.o
Linking C executable ../bin/comm
[ 60%] Built target comm
Scanning dependencies of target read
[ 80%] Building C object example/CMakeFiles/read.dir/read.c.o
Linking C executable ../bin/read
[ 80%] Built target read
Scanning dependencies of target write
[100%] Building C object example/CMakeFiles/write.dir/write.c.o
Linking C executable ../bin/write
[100%] Built target write
```

Figura A.6: Comando 6

```
root@debian:/home/leandro/Downloads/rtdb/bin# AGENT=0 ./comm eth0
**** Using device eth0 -> Ethernet 10.0.2.15

 ****communication: STARTED in sync mode...
```

Figura A.7: Comando 7

Apêndice B

Tutorial: Como implementar e utilizar do ZigBee com a RTDB

Este relatório tem como objetivo auxiliar a implementação da Rede ZigBee na RTDB - Real-Time Database (https://code.google.com/p/rtdb/), que consiste em uma base de dados distribuída em tempo real, que possibilita que dados locais, não persistentes, sejam compartilhados e acessados entre sistemas móveis, simulando um acesso local. Originalmente, a comunicação entre os agentes da RTDB é feita pelo protocolo padrão para redes locais sem fio (LAN), o IEEE 802.11. Já a distribuição das informações é realizada via socket utilizando a tecnologia de redes *multicast*. Com o uso do ZigBee, a rede passa a ser classificada como PAN (Personal Area Network) e o padrão utilizado é o IEEE 802.15.4 que visa aplicações sem fio para equipamentos que não precisem de alta taxa de dados, mas que necessitam de baixa latência e baixo consumo de energia. Como pré-requisito, deve-se obter a instalação de um sistema operacional Linux e seguir os passos presentes no Tutorial de Instalação da RTDB. Neste caso utiliza-se a distribuição Debian 7.

B.1 Configurando o ZigBee

Neste procedimento foram utilizados módulos ZigBee modelo Xbee S1 (Figura B.1) e todos tiveram seus Firmwares atualizados para a versão mais recente

A conversação do ZigBee com o sistema operacional é feita através por uma comunicação porta serial e suas configurações podem ser realizadas utilizando um terminal ou pelo software X-CTU [44]. Neste caso utilizou-se o X-CTU. Os parâmetros de conexão utilizados foram:

Figura B.1: Xbee Serie 1.

- Baud Rate 56800
- Data bits = 8
- Flow control = none

Utilizando o programa X-CTU, todos os módulos ZigBee foram configurados para estarem conectados em uma rede com o mesmo identificador (ID). Neste caso o ID=3332. Aos campos DH Destination Address High e DH Destination Address Low foram atribuídos os valores `0(zero)` e `0x000000000000FFFF`, respectivamente, conforme Figura 02. Estes campos representam os endereços inicial e final de comunicação da rede. Os valores inseridos possibilitam que todas as mensagens enviadas por um módulo ZigBee, para a rede ID=3332, sejam tratadas como broadcast e recebidas por todos os outros integrantes da rede.

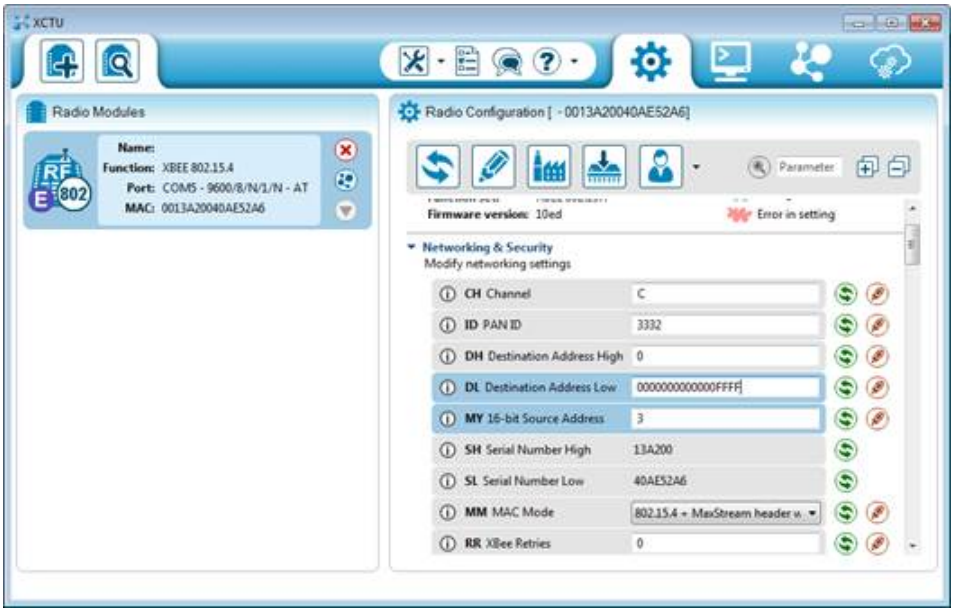


Figura B.2: Configuração XCTU - ZigBee

B.2 Configurando rede para o ZigBee

Como percebido, a comunicação do ZigBee é dada via porta serial e da RTDB é feita através de pacotes TCP/IP que, por padrão, não são tecnologias compatíveis. Com o objetivo de sanar este problema utilza-se o Protocolo SLIP (Serial Line Internet Protocol). O SLIP pode ser definido como um método de encapsulamento para transmissão de pacotes IP através de uma conexão serial. Para associar uma interface de rede a um dispositivo serial utiliza-se a função do "`slattach`", caso nenhum outro programa que esteja usando a porta serial. Neste caso associa-se o dispositivo "/dev/ttyS3" (terceira porta serial) à interface `sl0`. Já a opção "&" faz com que o comando seja executado em background no sistema.

```
root@rtdb:/home/leandro# slattach -L -s 9600 /dev/ttyS3 &
[1] 3834
```

Figura B.3: Executado o comando e o processo número 3834 foi criado

Nesse ponto a interface está ativa, mas a máquina ainda não conhece nada sobre a rede ou como alcançar os demais integrantes. Pra isso foi necessário atribuir um IP para a interface sl0, Utilizando o seguinte comando:

Feito isso a máquina está com o endereço de rede válido e o processo foi repetido nos demais integrantes (agentes) do sistema. Os passos descritos acima foram os mesmos, apenas alterando os endereços IP's a serem usados (Ex.: 10.0.0.2, 10.0.0.3...).

B.3 Configurando a RTDB para ZigBee

Conforme descrito na documentação oficial da digi.com (fabricante do Xbee), o BUFFER suportado pelo módulo utilizado neste experimento (Xbee S1) é de 202 bytes. Caso o Buffer do módulo atinja um valor maior que o valor padrão poderá ocorrer falhas na conexão do dispositivo. Com o objetivo de evitar este possível *overflow* [ref] no ZigBee, a RTDB precisou ser modificada para que não enviasse mais que 202 bytes de Buffer para o módulo. A alteração foi realizada no código-fonte do arquivo `rtdb/comm/comm.c` e a constante `BUFFER_SIZE` foi defina para o valor de 202, conforme Figura B.5:

Isto refletiu nas demais funções que utilizam-se desta constante, como por exemplo o bzero(), responsável pela alocação de memória do sistema e a função recv(), utilizada para receber/ler uma mensagem de um socket. Realizada esta alteração no código, a

```
root@rtdb:/home/leandro# ifconfig sl0 10.0.0.1 netmask 255.255.255.0
root@rtdb:/home/leandro# ifconfig
eth0      Link encap:Ethernet  Endereço de HW 08:00:27:41:2a:3f
          inet end.: 192.168.1.15  Bcast:192.168.1.255  Masc:255.255.255.0
          endereço inet6: fe80::a00:27ff:fe41:2a3f/64 Escopo:Link
          UP BROADCASTRUNNING MULTICAST  MTU:1500  Métrica:1
          RX packets:7678 errors:0 dropped:0 overruns:0 frame:0
          TX packets:286 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:1000
          RX bytes:4011275 (3.8 MiB)  TX bytes:30073 (29.3 KiB)

lo        Link encap:Loopback Local
          inet end.: 127.0.0.1  Masc:255.0.0.0
          endereço inet6: ::1/128 Escopo:Máquina
          UP LOOPBACKRUNNING  MTU:16436  Métrica:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 txqueuelen:0
          RX bytes:480 (480.0 B)  TX bytes:480 (480.0 B)

sl0       Link encap:SLIP VJ
          inet end.: 10.0.0.1  P-a-P:10.0.0.1  Masc:255.255.255.0
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:296  Métrica:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
             compactados:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          colisões:0 compactados:0 txqueuelen:10
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Figura B.4: Comando Ifconfig

```
#include "multicast.h"

#include "rtdb_comm.h"


#define BUFFER_SIZE 202

#define TTUP_US 100E3
#define COMM_DELAY_US 150
#define MIN_UPDATE_DELAY_
```

Figura B.5: Buffer RTDB

RTDB precisou ser recompilada (Passo 5 do Tutorial de Instalação) e a comunicação pode ser iniciada através da interface de comunicação via Serial/ZigBee.

```
-rwxr-xr-x 1 root    root   27285 Jul  1 23:18 comm
-rwxr-xr-x 1 root    root   17455 Jun  4 21:53 read
-rwxr-xr-x 1 leandro users  42017 Fev 28  2010 rtdb_parser
-rwxr-xr-x 1 root    root   17368 Jun  4 21:53 write
root@rtdb:~/rtdb/bin# AGENT=0 ./comm sl0
**** Using device sl0 -> Ethernet 10.0.0.1

 ****communication: STARTED in sync mode...
```

Figura B.6: RTDB inicializada

Apêndice C

Script: Instalação da RTDB

Código C.1: Script de Instalação da RTDB: rtdb_install.sh

```
#!/bin/bash

# creating symbolic links for directory utils

echo "Creating Include Links..."
echo

for i in rtdb comm example;
do
   (cd include ; find ../$i/ -name \*.h -exec ln -sf '{}' . \; )
done

echo "Cleaning old files..."
echo

make clean

echo "Compiling..."
echo
cmake .

make
```

Apêndice D

Script: Inicialização do ZigBee no Linux: zigbee_startcom.sh

Código D.1: zigbee_startcom.sh

```
#!/bin/sh
#############################################################################
#
#   @file:       starcom_zigbee.sh
#
#   @description: ShellScript
#                 ZigBee (SLIP Configuration)
#
#   @author:     Leandro Caetano
#          leandrocaetano.info
#          leandrojcaetano@gmail.com
#
#   @created:    11.05.2015
#
#   @updated:    14.05.2015
#
#############################################################################
#


clear
echo "\n#########################################"
```

```
23 echo   "######## CONFIGURACAO ZIGBEE ########"
24 echo   "########################################\n"
25
26
27 echo "Selecione qual operacao deseja realizar:"
28 echo ""
29 echo "1 - Ativar ZigBee no sistema"
30 echo "2 - Desativar ZigBee no sistema"
31 echo "\nOpÃ§Ã£o:" '\c'
32 read selecao
33 case $selecao in
34   1)     clear
35      echo "\n################################"
36      echo   "### Ativar ZigBee no sistema ###"
37      echo   "################################"
38
39      var="$(ps axf | grep slattach | grep -v grep | awk '{print $1}')"
40      if [ -n "$var" ]; then
41         echo "\n[INFO] JÃ¡ existem dispositivos ZigBee ativados no sistema"
42         echo "\n[INFO] Desativando dispositivos ZigBee conectado..."
43         sleep 1
44         sudo ps axf | grep slattach | grep -v grep | awk '{print "sudo kill -9
               " $1}' | sh
45      fi
46
47
48      if ! command -v setserial >/dev/null 2>&1 ; then
49         echo "\n[INFO] Verificando dependencias necessÃ¡rias..."
50         echo ""
51              # Necessario acesso Ã  internet
52                sudo apt-get install setserial
53
54           if [ $? != 0 ];
55           then
56               echo "\n[ERROR] Dependencias NÃ£o instaladas... Verifique sua
                     conexÃ£o com a internet\n"
57           exit
```

```
58            fi
59
60          #clear
61          sleep 1
62               echo "\n[OK] Dependencias instaladas...\n"
63      fi
64
65
66
67      echo ""
68      echo "Lista de dispositivos disponiveis:"
69      echo ""
70          sudo setserial -g /dev/ttyUSB[0-3]
71
72      echo "\nSelecione qual desses dispositivos Ã© o ZigBee conectado:"
73      echo "\n/dev/ttyUSB"'\c'
74      read opcao
75          echo "#########################################"
76          echo "Voce Selecionou o dispositivo /dev/ttyUSB$opcao"
77          echo "#########################################"
78          echo "\nExecutando..."
79
80             sudo slattach -L -s 38400 /dev/ttyUSB$opcao &
81
82             sleep 1
83             echo ""
84
85             var4="$(ps axf | grep slattach | grep -v grep | awk '{print $1}')"
86
87                if [ -n "$var4" ]; then
88                   echo "[OK] Processo criado. PID=$var4"
89                else
90                    echo "\n[ERROR] OpÃ§Ã£o invÃ¡lida...\n"
91                   exit
92                fi
93          echo "\n#########################"
94          echo  "Qual o nÃºmero deste AGENT?"
```

```
95          echo  "############################"
96          echo "\nAGENT="'\c'
97          read opcao2
98
99          sudo ifconfig sl0 10.0.0.$opcao2 netmask 255.255.255.0 mtu 200
100
101         echo "[OK] Inicialize a RTDB com o seguinte comando:\n AGENT=$opcao2
                ./comm sl0"
102
103     ;;
104
105     2)
106        clear
107        echo  "\n#####################################"
108        echo    "### Desativar ZigBee no sistema ###"
109        echo    "#####################################"
110
111        echo  "\n[INFO] Desativando dispositivos ZigBee conectado"
112        sleep 1
113        sudo ps axf | grep slattach | grep -v grep | awk '{print "sudo kill -9 "
               $1}' | sh
114     ;;
115
116 esac
117 echo  "\n\n[OK] OperaÃ§Ã£o realizada com sucesso... \n\nPrecione qualquer tecla
      para sair."
118
119 read x
120 echo "Tchau! =D"
```

Apêndice E

API de Integração RTDB - RoC

Código E.1: API de Integração RTDB - RoC: rtdb_api.pas

```
1  unit rtdb_api;
2
3  {$link librtdb.a}
4  {$linklib c}
5
6  interface
7  uses CTypes;
8
9  // ************************
10 // DB_init: Aloca acesso a base de dados
11 //
12 // Saida:
13 //    0 = OK
14 //    -1 = erro
15 //
16 //      int DB_init (void);
17 function DB_init(agentNumber: integer): integer; cdecl; external;
18
19 // ************************
20 // DB_free: Liberta acesso a base de dados
21 //
22 //void DB_free (void);
23
24 procedure DB_free; cdecl; external;
```

```
25
26
27 // ************************
28 // DB_put: Escreve na base de dados do proprio agente
29 //
30 // Entrada:
31 //    int _id = identificador da 'variavel'
32 //    void *_value = ponteiro com os dados
33 // Saida:
34 //    0 = OK
35 //    -1 = erro
36 //
37 //int DB_put (int _id, void *_value);
38
39 function DB_put(_id: integer; _value: pointer): integer; cdecl; external;
40
41
42 // ************************
43 // DB_get: Le da base de dados
44 //
45 // Entrada:
46 //    int _agent = numero do agente
47 //    int _id = identificador da 'variavel'
48 //    void *_value = ponteiro para onde sao copiados os dados
49 // Saida:
50 //    int life = tempo de vida da 'variavel' em ms
51 //       -1 se erro
52 //
53 // int DB_get (int _from_agent, int _id, void *_value);
54
55 function DB_get(_from_agent: integer; _id: integer; _value: pointer): integer;
       cdecl; external;
56
57 // ************************
58 // Whoami: identifica o agente onde esta a correr
59 //
60 // Saida:
```

```
61 //    int agent_number = numero do agente
62 //
63 //int Whoami(void);
64
65 function Whoami: integer; cdecl; external;
66
67
68 implementation
69
70 end.
```

Apêndice F

Espcificações: TL-WA901ND Wireless N Access Point

General	
Standards and Protocols	IEEE 802.3, 802.3u, 802.11n, 802.11b and 802.11g, TCP/IP, DHCP
Safety & Emission	FCC、CE
Ports	One 10/100M Auto-Negotiation LAN RJ45 port, supporting passive PoE
Cabling Type	10BASE-T: UTP category 3, 4, 5 cable (maximum 100m) EIA/TIA-568 100Ω STP (maximum 100m) 100BASE-TX: UTP category 5, 5e cable (maximum 100m) EIA/TIA-568 100Ω STP (maximum 100m)
Wireless	
Frequency Band	2.4~2.4835GHz
Radio Data Rate	11n：up to 300Mbps（Automatic） 11g：54/48/36/24/18/12/9/6M（Automatic） 11b：11/5.5/2/1M（Automatic）
Frequency Expansion	DSSS(Direct Sequence Spread Spectrum)
Modulation	DBPSK, DQPSK, CCK, OFDM, 16-QAM, 64-QAM
Security	WEP/WPA/WPA2/WPA2-PSK/WPA-PSK
Sensitivity @PER	270M: -68dBm@10% PER 108M: -68dBm@10% PER; 54M: -68dBm@10% PER 11M: -85dBm@8% PER; 6M: -88dBm@10% PER 1M: -90dBm@8% PER
Antenna Gain	4dBi
Physical and Environment	
Working Temperature	0℃~40℃ (32℉~104℉)
Working Humidity	10% ~ 90% RH, Non-condensing
Storage Temperature	-40℃~70℃(-40℉~158℉)
Storage Humidity	5% ~ 90% RH, Non-condensing

Figura F.1: Espcificações: TL-WA901ND Wireless N Access Point

Apêndice G

Quadro Comparativo entre ZigBee, Bluetooth e Wi-Fi.

	ZigBee	Bluetooth	Wi-Fi
Especificação IEEE	802.15.4	802.15.1	802.11/a/b/g
Aplicações	Controlo e monitorização	Alternativa aos cabos	Web, vídeo e e-mail
Banda de Frequência	868 MHz – Europa 915 MHz – América 2.4 GHz – Global	2.4 GHz	2.4GHz; 5GHz
Máxima taxa de dados	250 Kbps	1Mbps	54 Mbps
Alcance Nominal	10-100 Metros	10 Metros	100 Metros
Número de canais de Rádio Frequência (RF)	1/868 MHz 10/915 MHz 16/2.4 GHz	79	14(2.4 GHz)
Largura de banda do canal	0.3 MHz 0.6 MHz 2 MHz	1 MHz	22 MHz
Tipo de Modulação	BPSK (+ASK), O-QPSK	GFSK	BPSK, QPSK COFDM, CCK, M-QAM
Espalhamento de frequências	DSSS	FHSS	DSSS, CCK, OFDM
Topologia básica	Estrela	Piconet	BSS
Extensão da topologia básica	Topologia em árvore	Scatternet	ESS
Nº máximo de nós	65536	8	2007
Encriptação	AES (CRT, modo contador)	E0 stream cipher	RC4 stream cipher (WEP)
Autenticação	CBC – MAC	Shared secret	WPA2 (8022.11i)
Protecção de dados	16-bits CRC	16-bits CRC	32-bits CRC
Tempo de vida da bateria	Anos	Dias	Horas
Vantagens	Baixo consumo de energia Muitos dispositivos Baixo overhead Baixo custo	Amplamente aplicada à Electrónica de consumo	Domina as redes WLAN Amplamente disponíveis Alta taxa de transferência de dados
Desvantagens	Baixa taxa de transferência de dados	Consumo de energia, médio Poucos dispositivos na piconet	Alto consumo de energia

Figura G.1: Quadro comparativo entre ZigBee, Bluetooth e WiFi [3].

Apêndice H

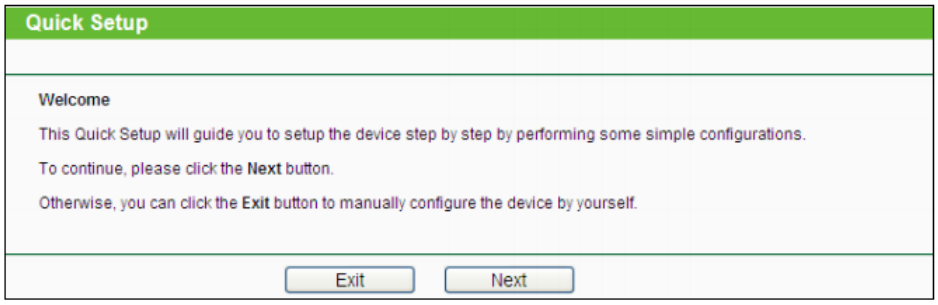
Configurar Rede Mesh TP-Link TL-WA901ND

H.1 Roteador 1

Inicialmente deve-se configurar o roteador que ficará conectado à rede cabeada (que possui acesso à internet). Este roteador deve ser configurado como modo Access Point:

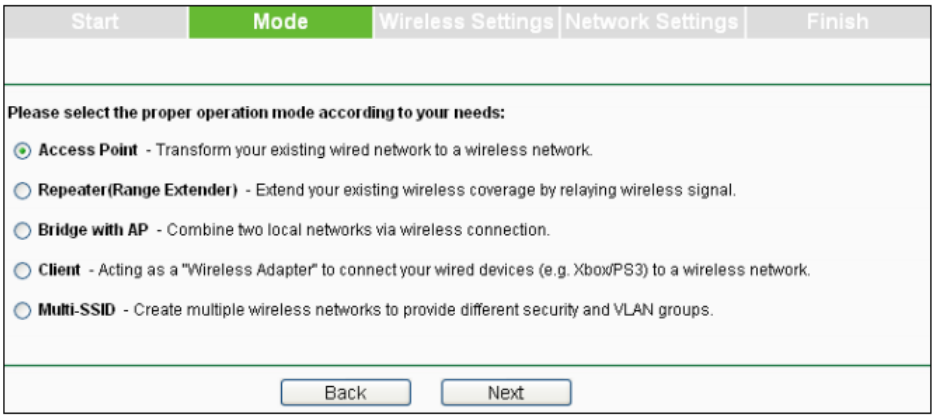
1. Efetue login na área administrativa do roteador pelo ip: 192.168.1.254 e com usuário e senha padrão: admin e senha: admin

2. Depois de logado, clique em Quick Setup. Clique em Next para continuar.



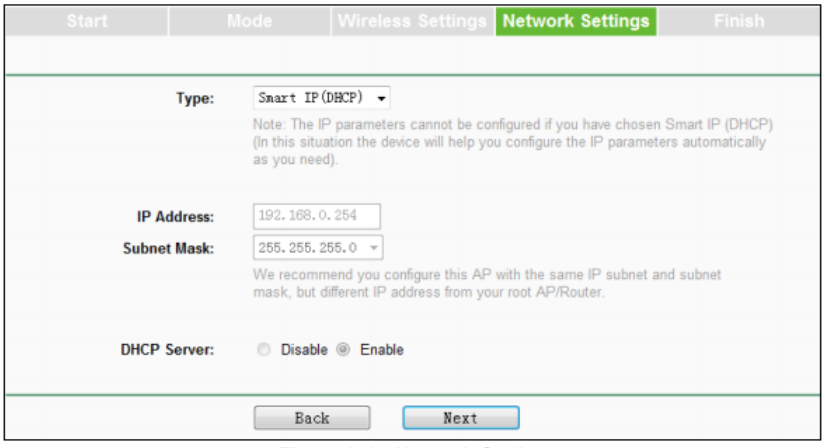
3. Ao aparecer a página do Quick Setup. Clique em Next para continuar.

4. Selecione a opção Access Point:



5. Configure o nome da rede e a senha de acesso.

6. Na página Network Settings é recomendado deixar as configurações padrão.



Pronto, o roteador está configurado para funcionar como ponto de acesso.

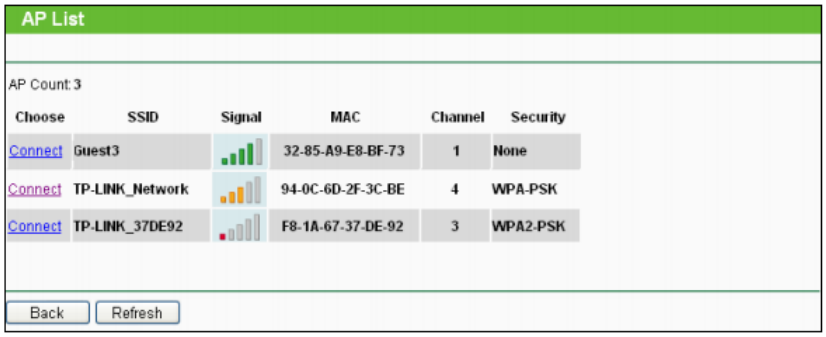
H.2 Roteador 2

1. Agora com o segundo roteador acesse o Quick Setup da mesma forma do roteador do ponto de acesso.

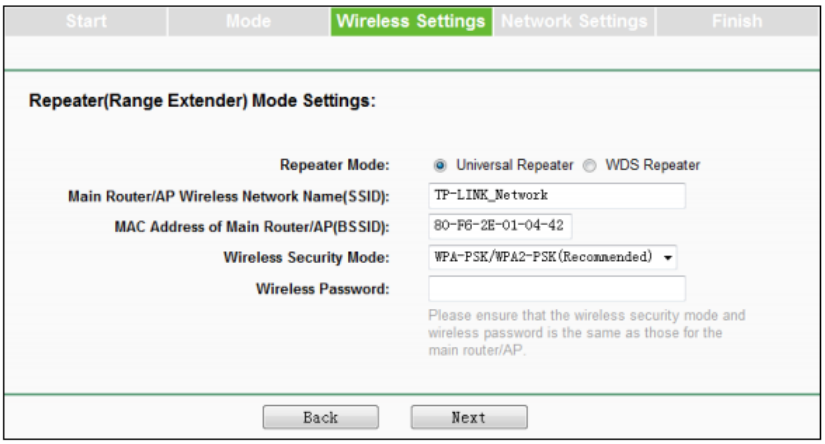
2. Porém defina o modo de operação para Repeater (Range Extender). No passo seguinte,

o roteador irá enxergar as redes sem fio que estão ao seu alcance.

Selecione a rede que foi configurada no Roteador 1.



3. Selecione o tipo de segurança configurado no Roteador 1 e insira a senha de acesso.



4. Na página Network Settings é recomendado deixar as configurações padrão.

Pronto, o roteador está configurado e a rede Mesh está formada. Caso haja mais roteadores a serem inseridos na rede, basta seguir o procedimento referente ao Roteador 2.